Computer Vision & Cognitive Systems

SCQ5109806 - LM CS,DS,CYB,PD

Foundations: image filtering, edges

Prof. Lamberto Ballan

University of Padova - A.Y. 2025/26 October 14, 2025

DIPARTIMENTO MATEMATICA
Department of Mathematics “Tullio Levi-Civita”

Summary

- Image sampling and quantization
- Image histograms
- Images as functions
- Filters (linear systems)
- Convolution and (cross-)correlation

Recap: linear filters

- We define a filter (system) as a unit that converts an input function $f[n, m]$ into an output (response) function $g[n, m]$, where (n, m) are the independent variables

- In the case of images, (n, m) represents the **spatial position** in the image

$$f[n, m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n, m]$$

Recap: moving average filter

- 2D moving average over a 3×3 window of neighborhood (this is also referred as to *box filter*)

$$g[n,m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k,l]$$

$$= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]$$

h

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Image Kernel

↓

$$(f * h)[n,m] = \sum_{k,l} f[k,l]\, h[n-k, m-l]$$

Convolution

Recap: moving average filter

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

$$(f * h)[n, m] = \sum_{k, l} f[k, l]\, h[n-k, m-l]$$

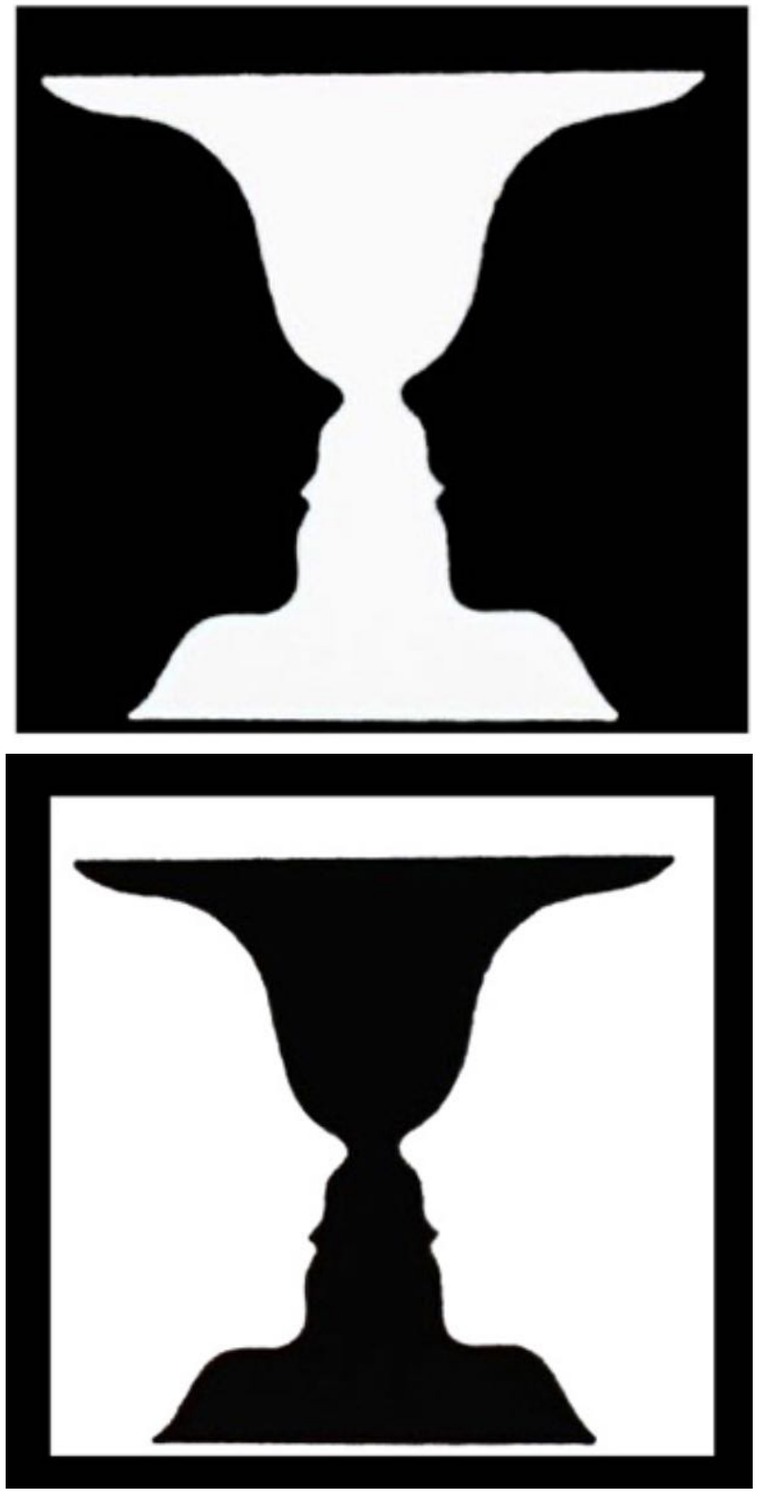
Convolution

Image filtering recap

- Compute a function of the local neighborhood at each pixel in the image
  - The function is specified by a “filter” saying how to combine values from neighbors

- Applications of image filtering:
  - extract information (edges, corners, blobs, …)
  - detect patterns (template matching)
  - de-noising, super-resolution, in-painting

What's an edge?

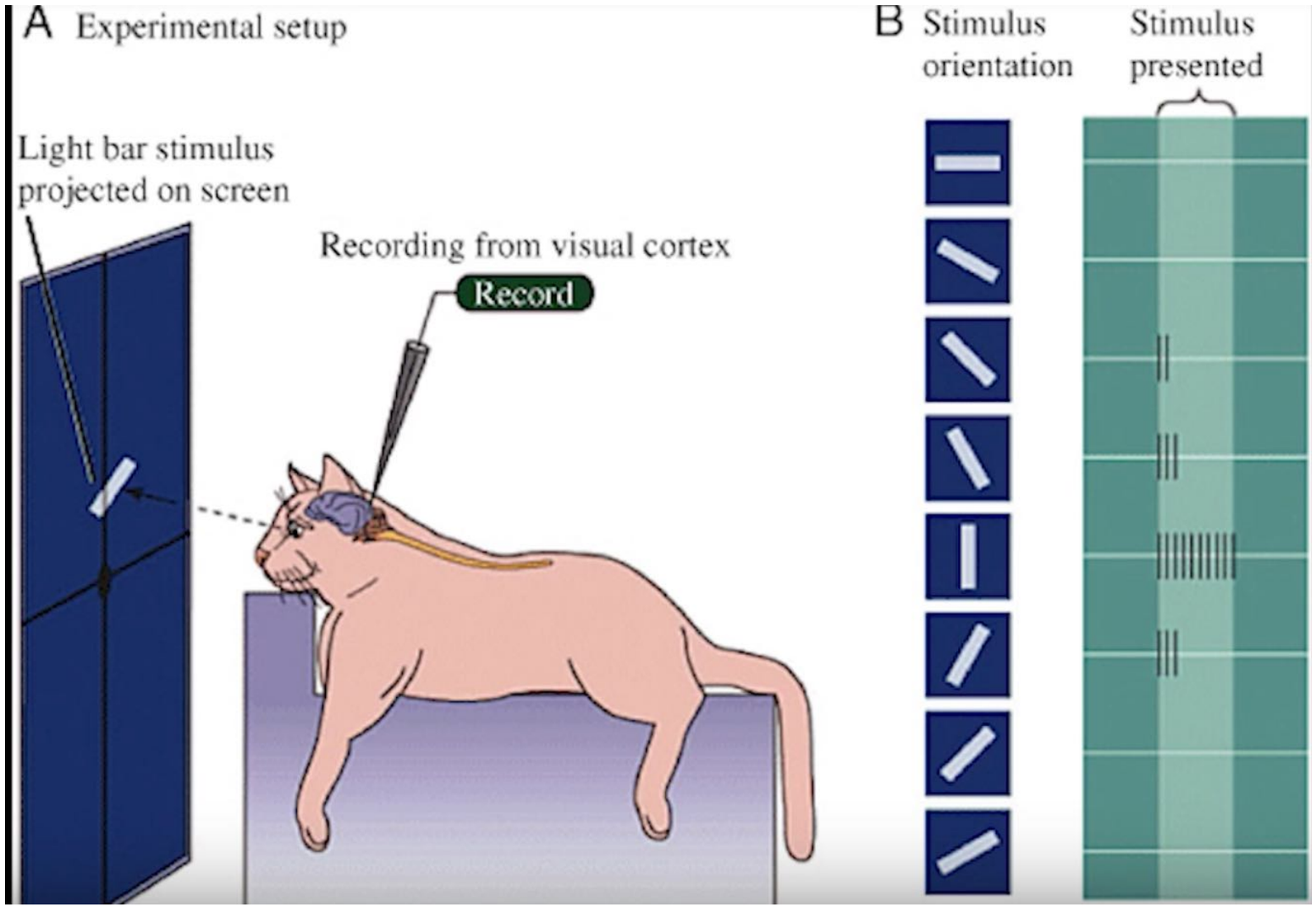


Cave painting (Chauvet, France, ~30,000 BC)

Shen Zhou's "Poet on a mountain top" (China, ~1,500 AD)

What's an edge?

- We know edges are special from human (mammalian) vision studies

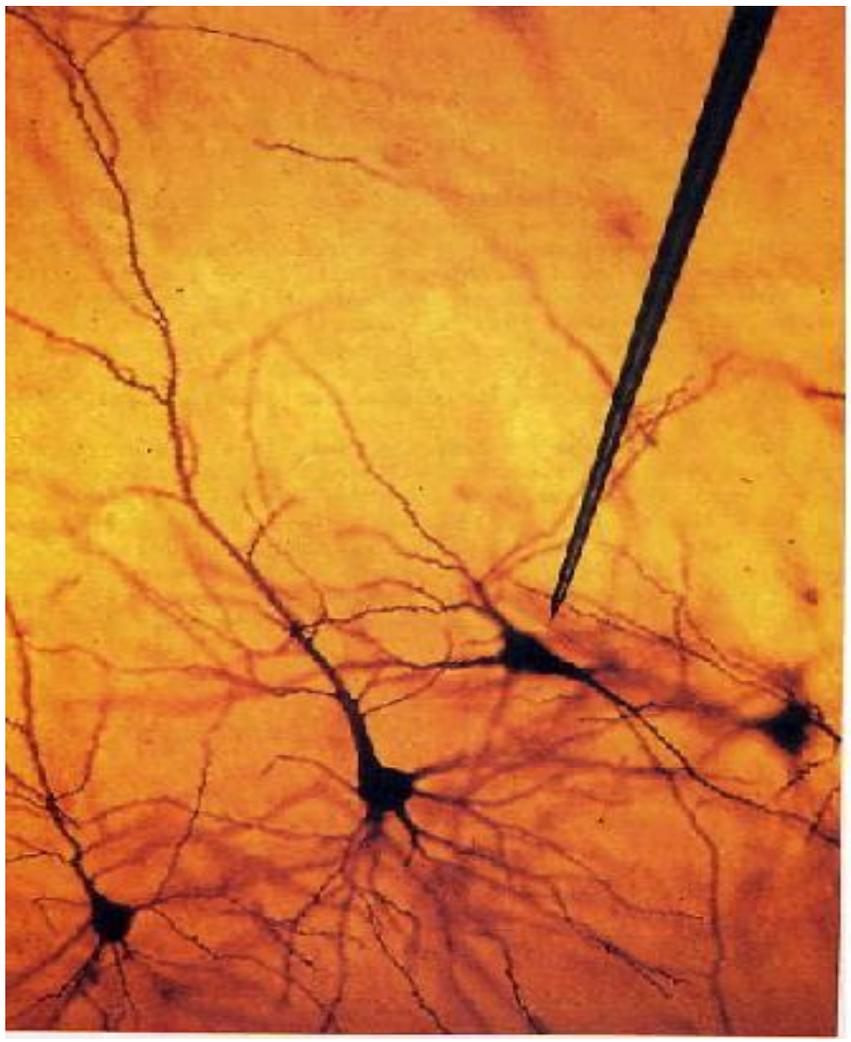


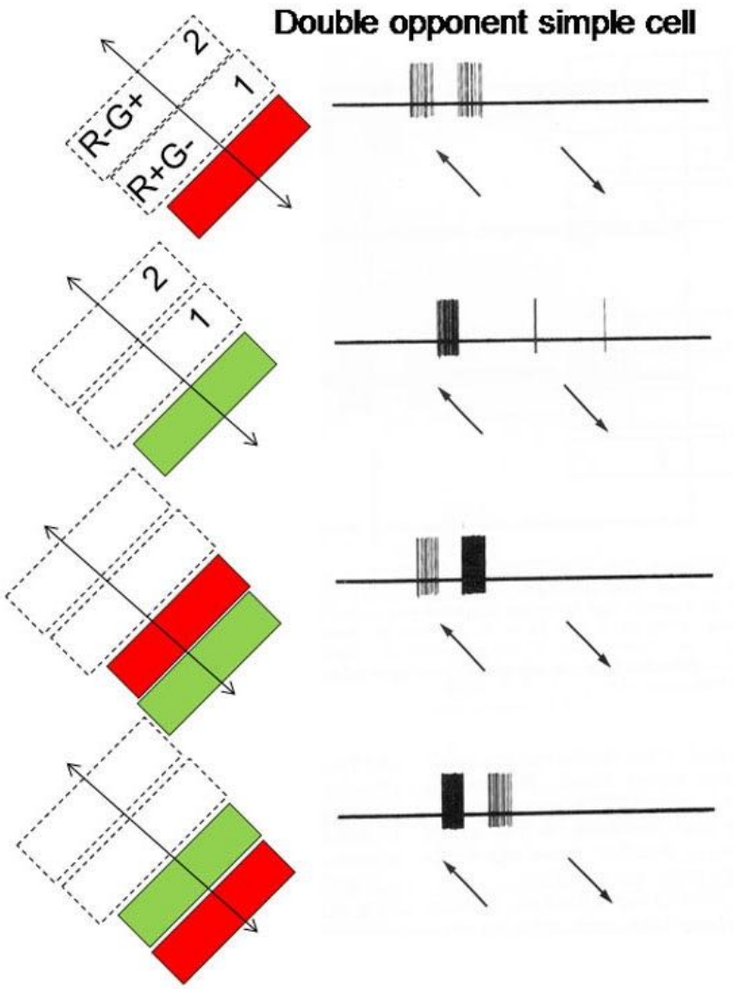
Hubel and Wiesel, 1960s

What's an edge?

- We know edges are special from human (mammalian) vision studies

Hubel and Wiesel, 1960s





https://www.youtube.com/watch?v=8VdFf3egwfg&t=18s

Edge detection

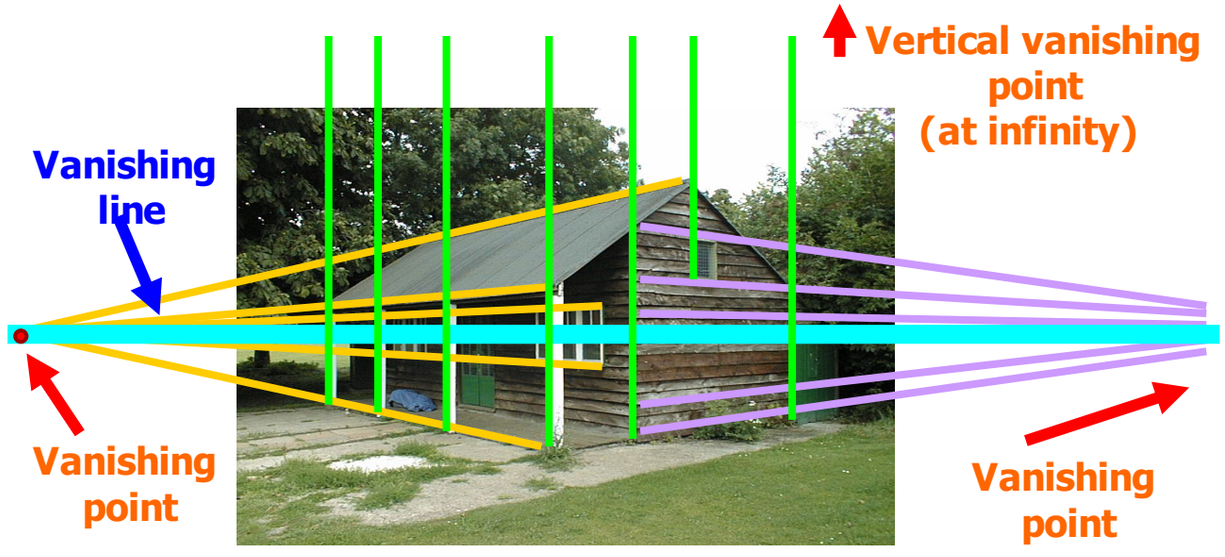
- **Goal:** identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

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

Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint

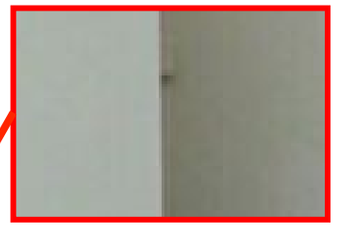


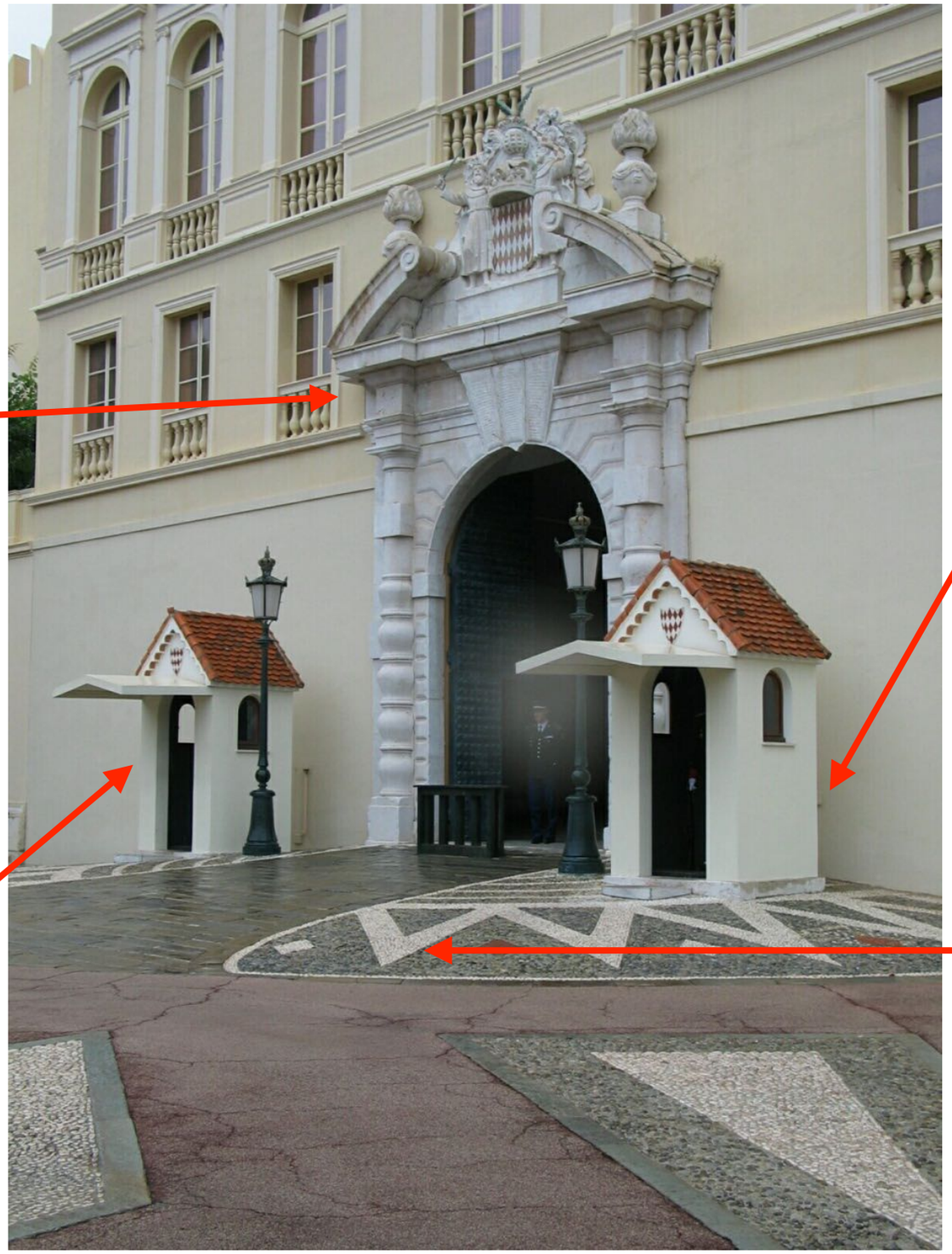
Slide credit: J. Hayes

What causes an edge?

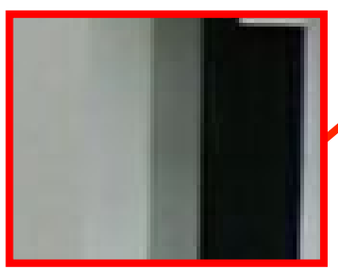
Illumination discontinuity:
cast shadows

Depth discontinuity:
object boundary





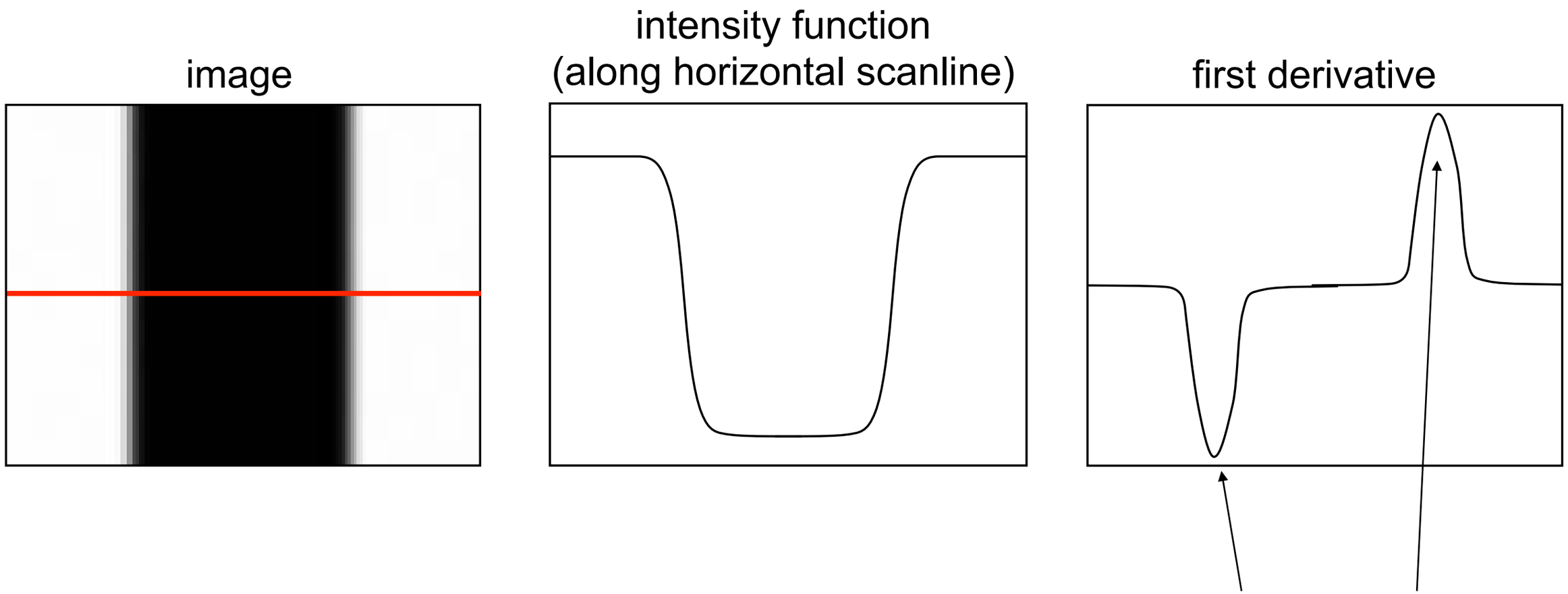
Change in surface
orientation: shape



Surface color
discontinuity

Edges and image gradients

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



This could be the sketch of an edge detection algorithm

Derivatives: a quick recap

- Derivative in 1D:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

A simple example:

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

- Discrete derivative in 1D:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x - 1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x - 1) = f'(x)$$

we can approximate using finite differences

Derivatives: a quick recap

- Types of Discrete derivative in 1D:

  Filter

  - Backward: $\frac{df}{dx} = f(x) - f(x-1) = f'(x)$ `[0 1 -1]`

  - Forward: $\frac{df}{dx} = f(x) - f(x+1) = f'(x)$ `[-1 1 0]`

  - Central: $\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$ `[1 0 -1]`

Derivatives: a quick recap

- 1D discrete derivate example:

$$f(x) = \underset{\text{-1}\quad\ \text{1}\quad\ \text{0}}{10\ \ 15\ \ 10}\ \ 10\ \ 25\ \ 20\ \ 20\ \ 20$$

$$f'(x) = \ 0\ \ 5\ \ -5\ \ 0\ \ 15\ \ -5\ \ 0\ \ 0$$

- What's the derivative filter used in this example?

☑ Backward ☐ Forward ☐ Central

Filter `[0 1 -1]`

Derivatives: a quick recap

- 1D discrete derivate example:

$$f(x) = \begin{matrix} & 0 & 1 & -1 & & & & & \\ & 10 & 15 & 10 & 10 & 25 & 20 & 20 & 20 \end{matrix}$$

$$f'(x) = \begin{matrix} 0 & 5 & 0 & -15 & 5 & 0 & 0 & 0 \end{matrix}$$

- What's the derivative filter used in this example?

☐ Backward ☑ Forward ☐ Central

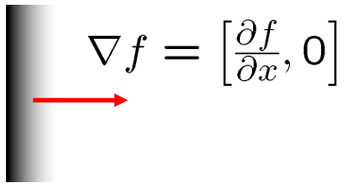
Filter `[-1 1 0]`

2D discrete derivative: image gradient

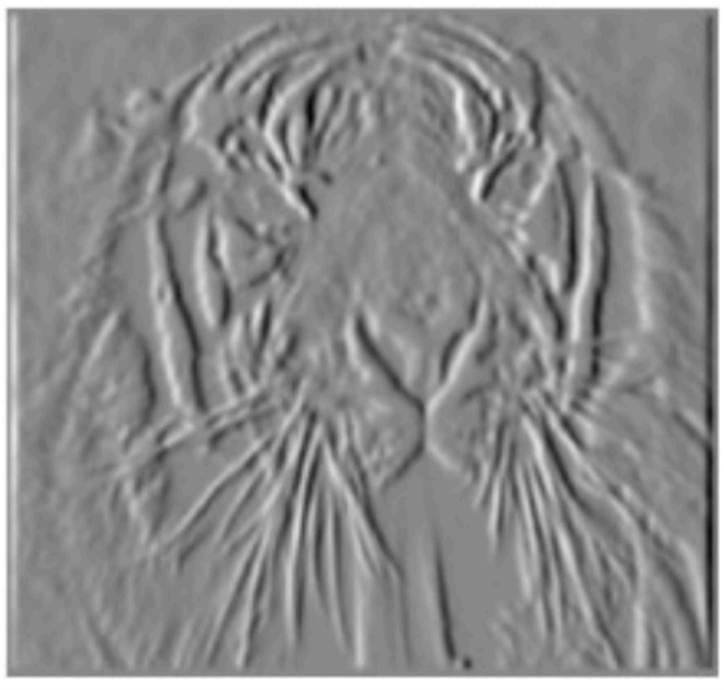
Given function $f(x, y)$

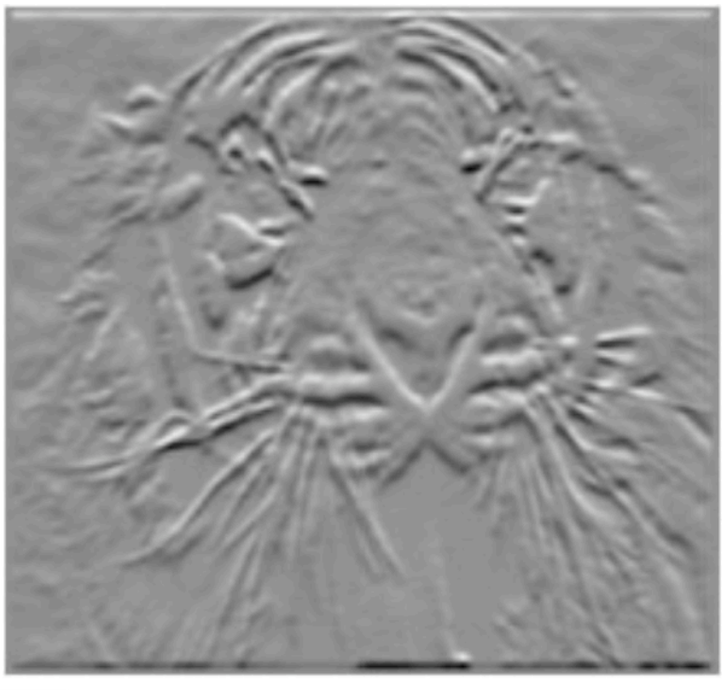
Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

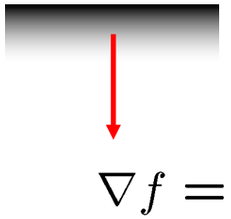
Which one is the gradient in the x-direction? How about y-direction?



x-direction







y-direction

Slide credit: S. Seitz

2D discrete derivate example

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \underbrace{\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}}_{\textit{Filter}} = \; ?$$

What happens when we apply this filter?

2D discrete derivate example

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \underbrace{\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}}_{\textit{Filter}} =$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2D discrete derivate example

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = ?$$

Filter

What happens when we apply this filter?

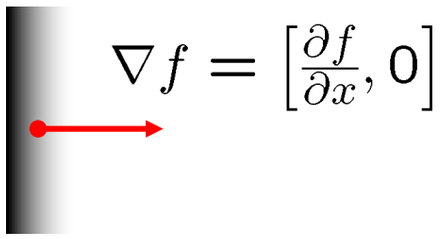
2D discrete derivate example

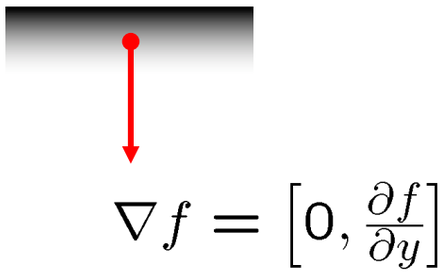
$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} * \underbrace{\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{\textit{Filter}} =$$

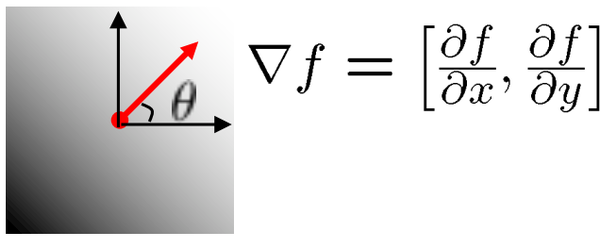
$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image gradient

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



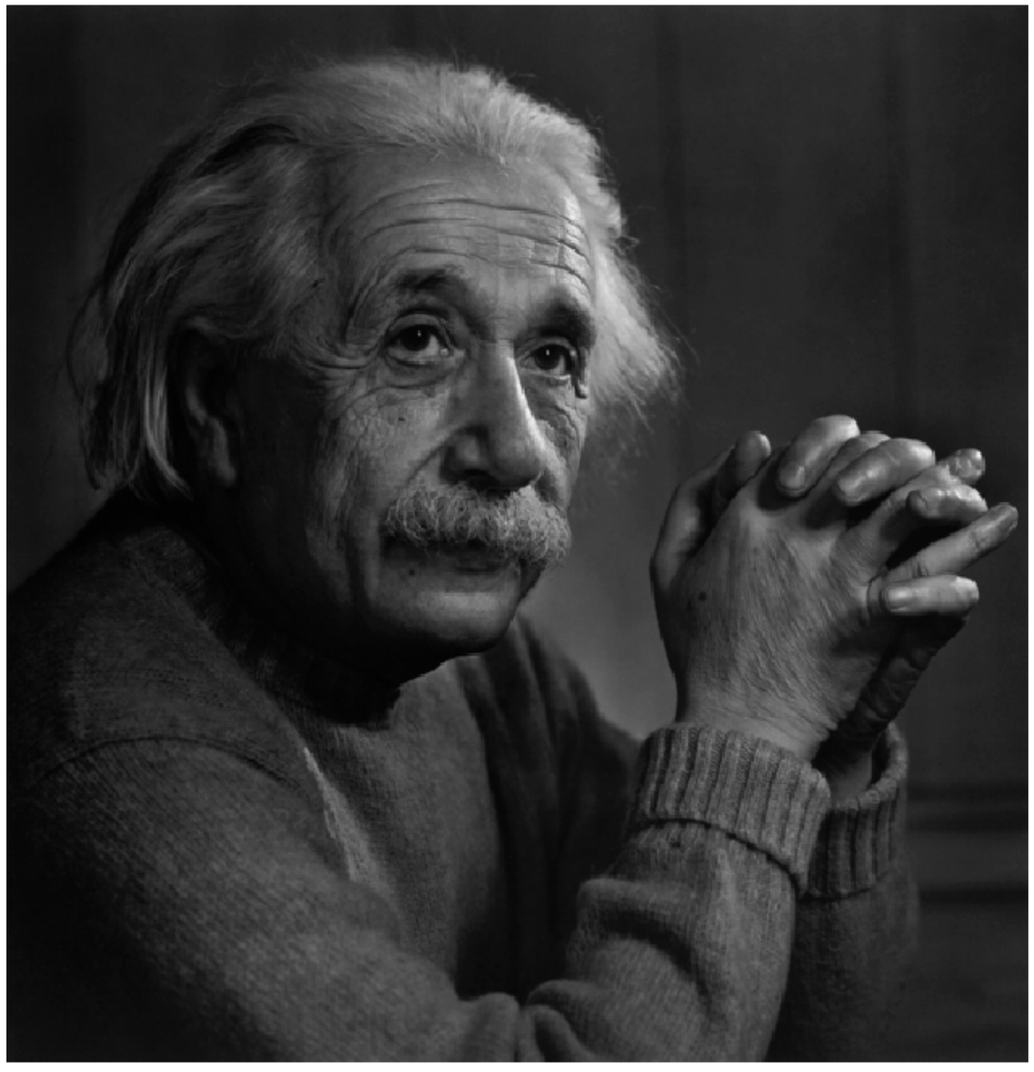


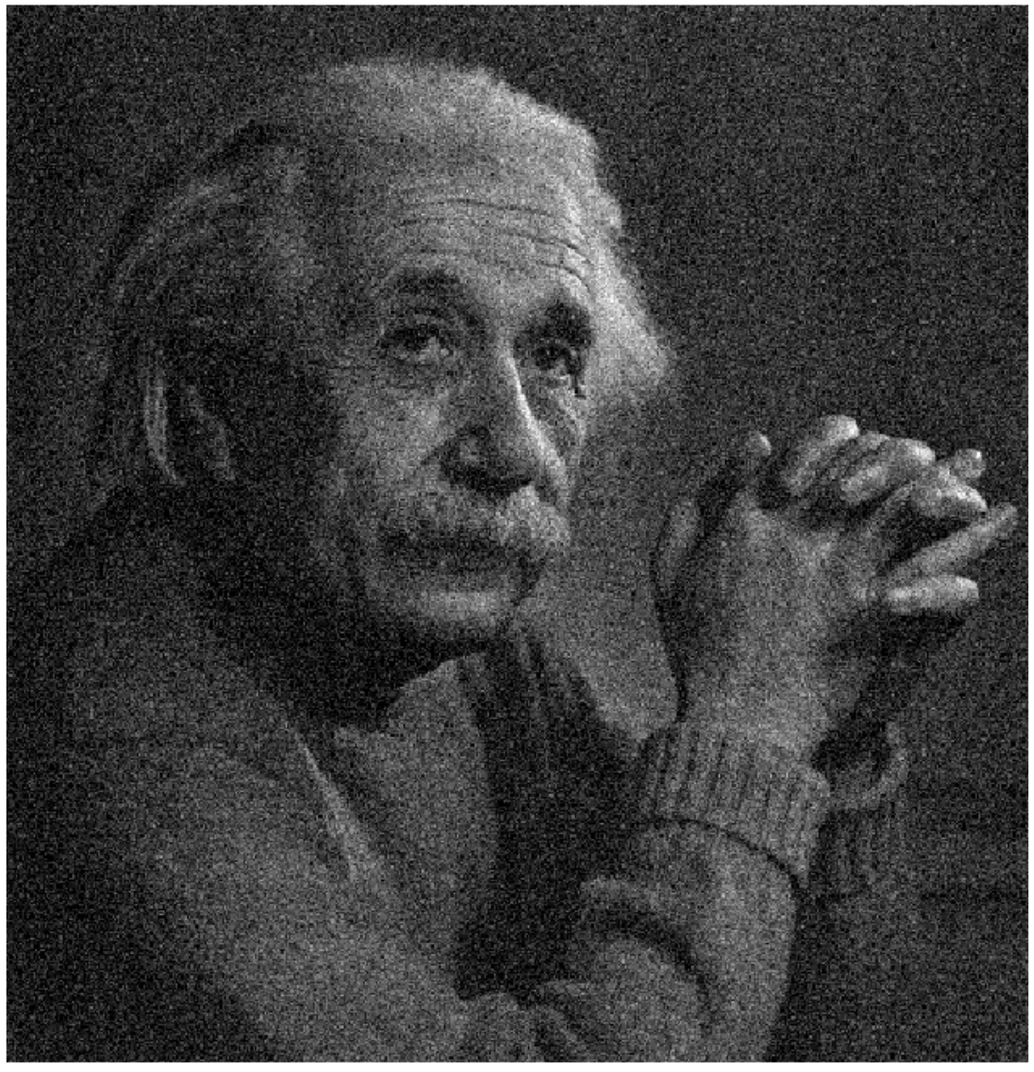


  - The gradient vector points in the direction of most rapid increase in intensity
  - The gradient direction is given by: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - The *edge strength* is given by the gradient magnitude:

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

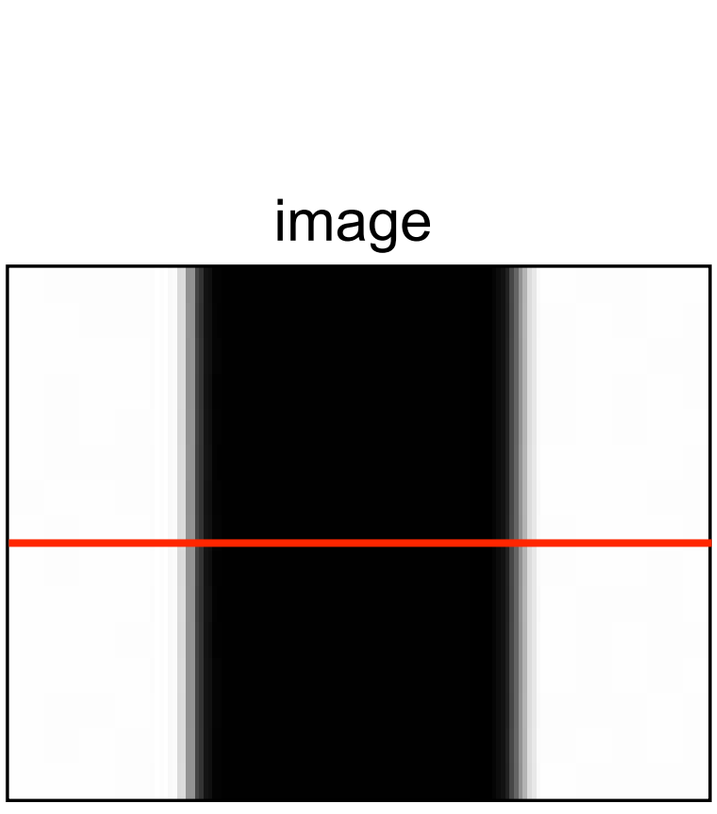
Effects of noise

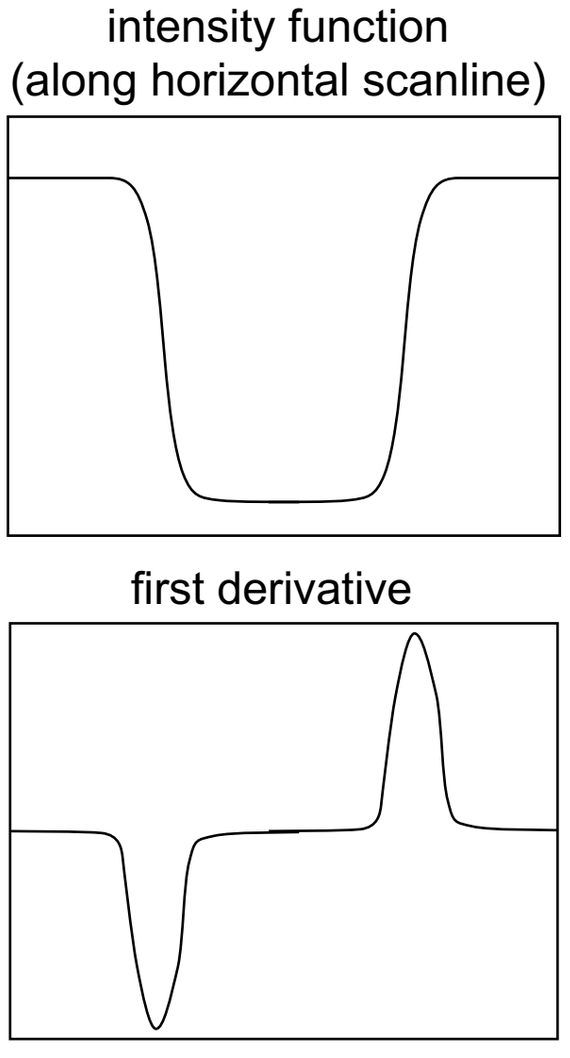




Intensity profile

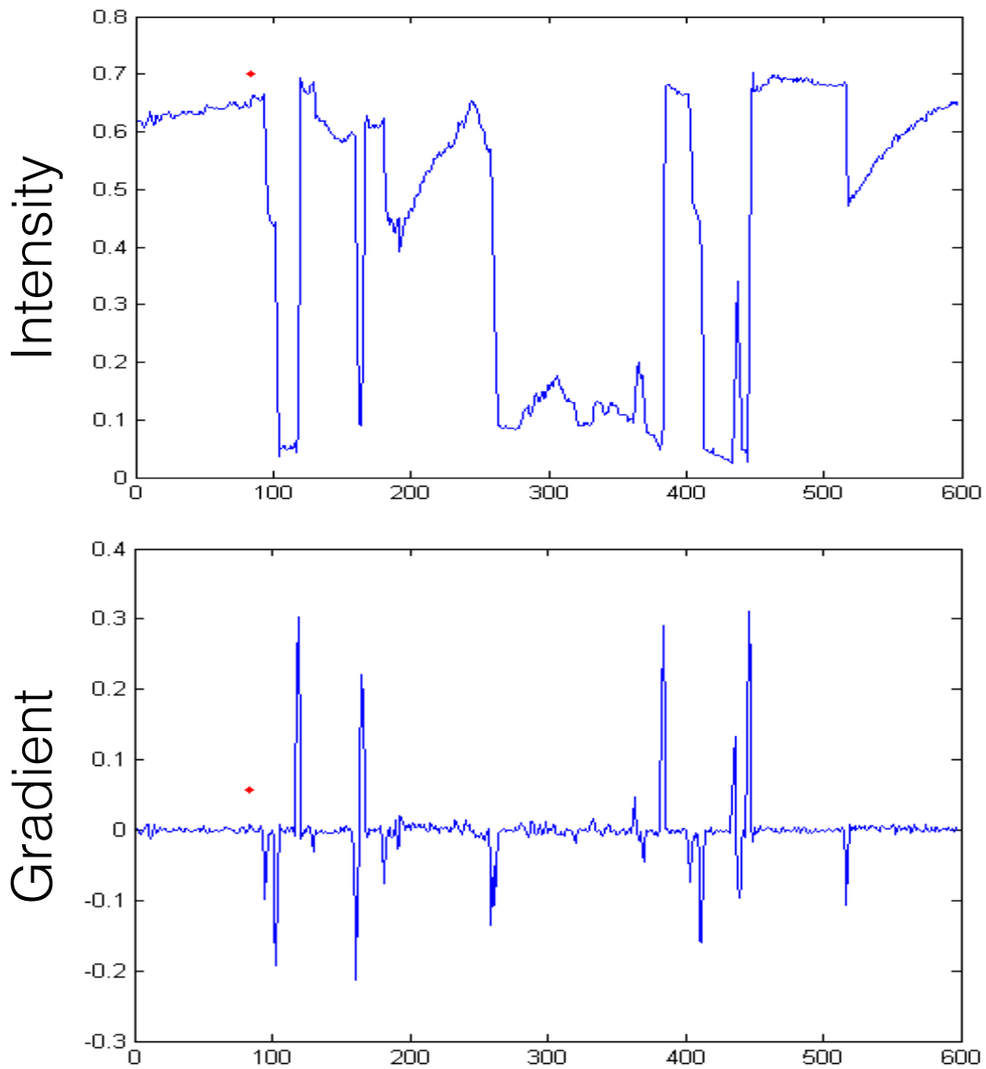
- Consider a single row or column of the image
  - Plot intensity as a function of position gives a signal





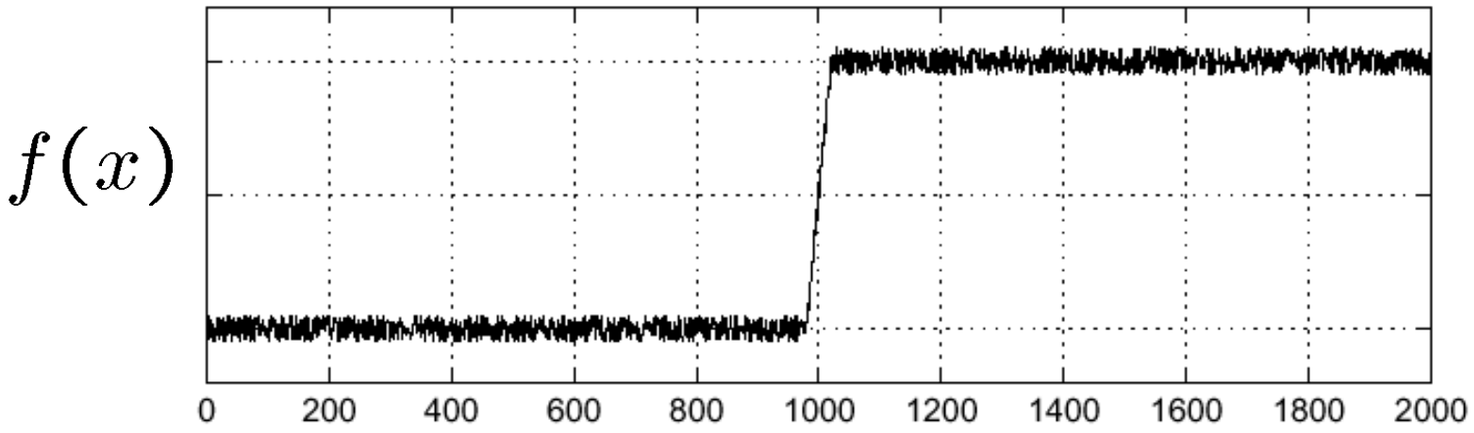
Intensity profile

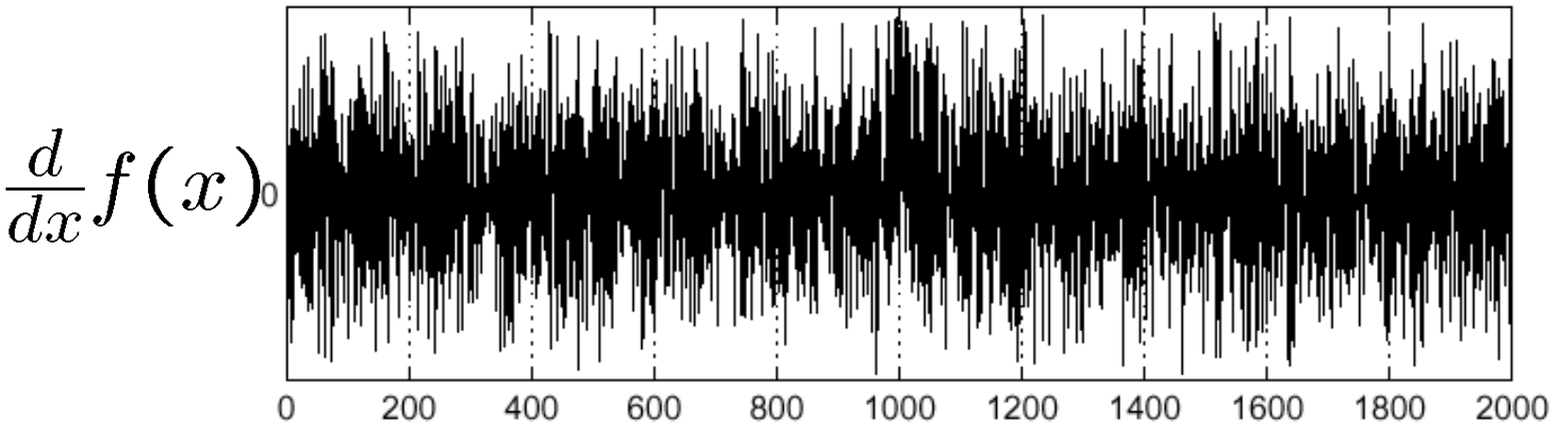
- Consider a single row or column of the image
  - Plot intensity as a function of position gives a signal



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?

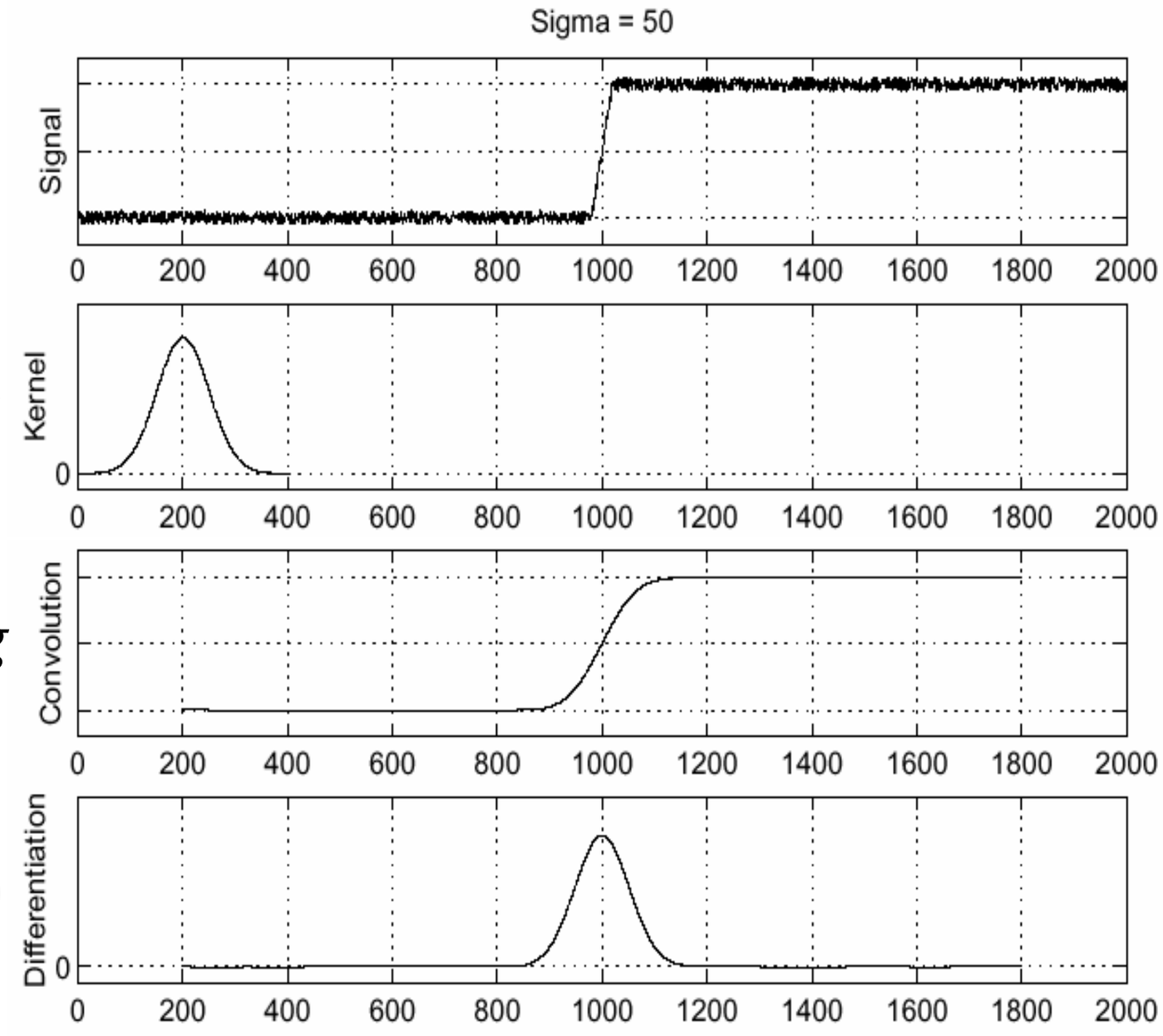
Slide credit: S. Seitz

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - The larger the noise, the stronger the response

- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (*=noise pixels?*) to look more like neighbors

Solution: smooth first



f

g

$f * g$

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

Where is the edge?

Look for peaks in $\frac{d}{dx}(f * g)$

Slide credit: S. Seitz

Derivative theorem of convolution

- Differential property of convolution:

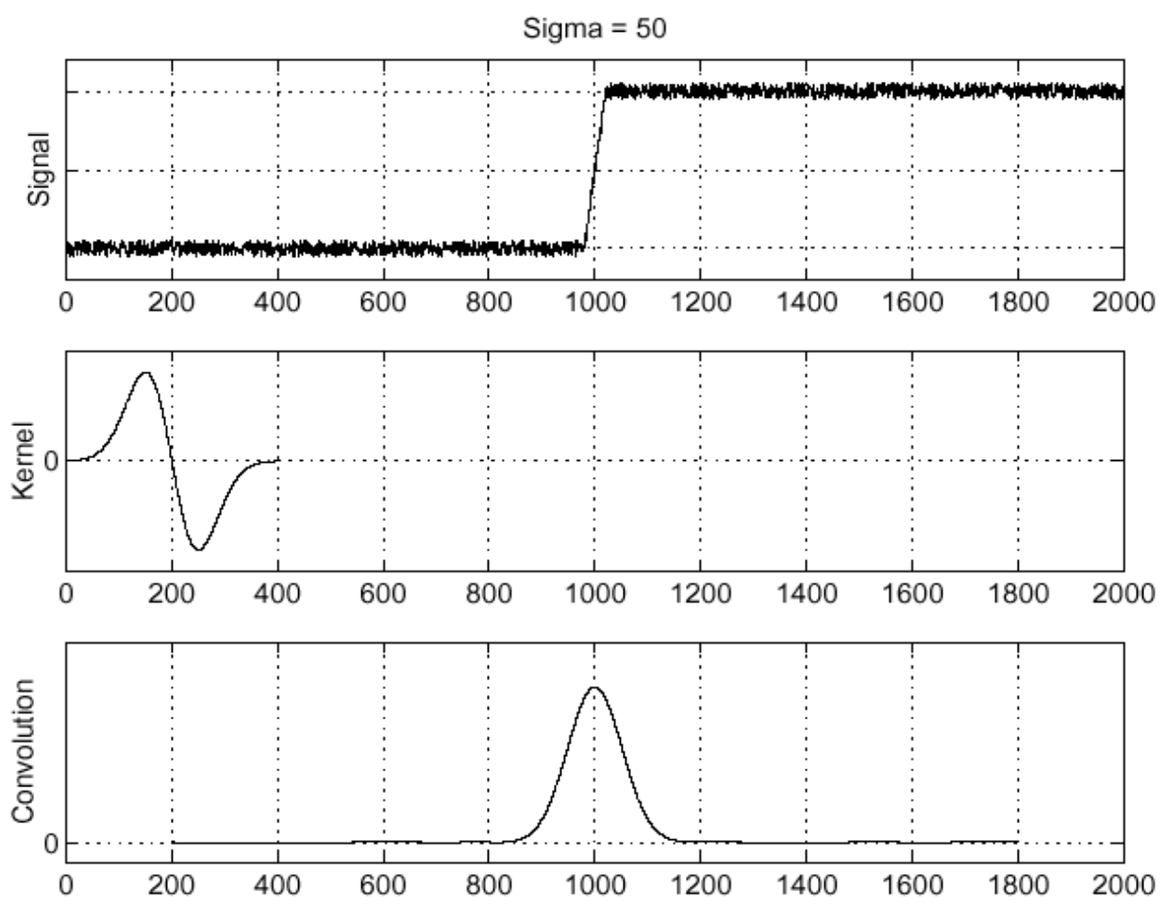
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

  - This is a useful property and save us one operation:

f

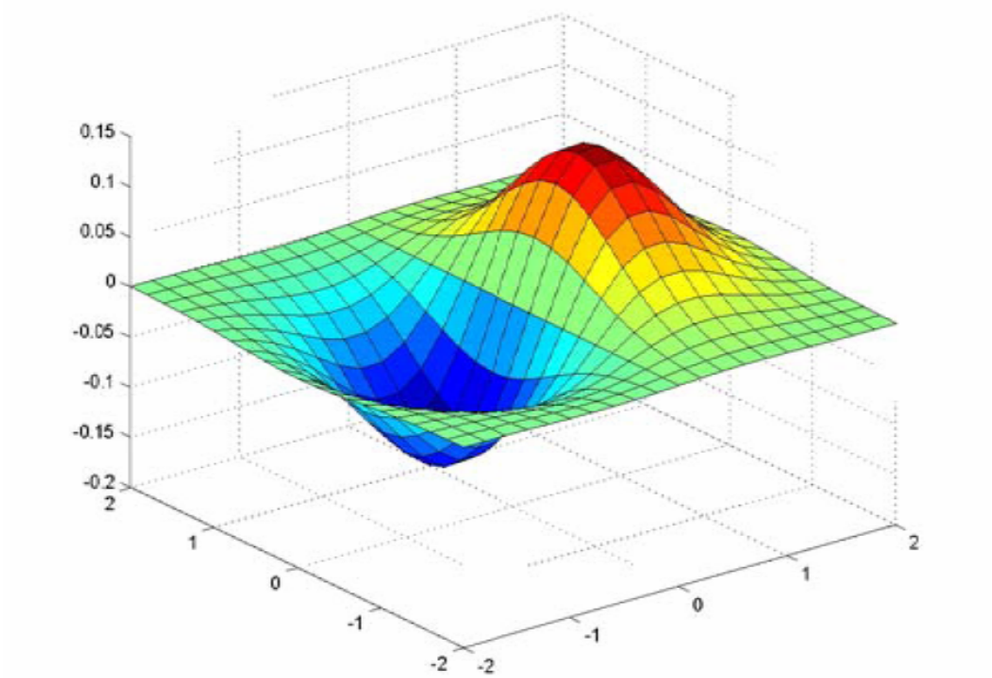
$\frac{d}{dx} g$

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

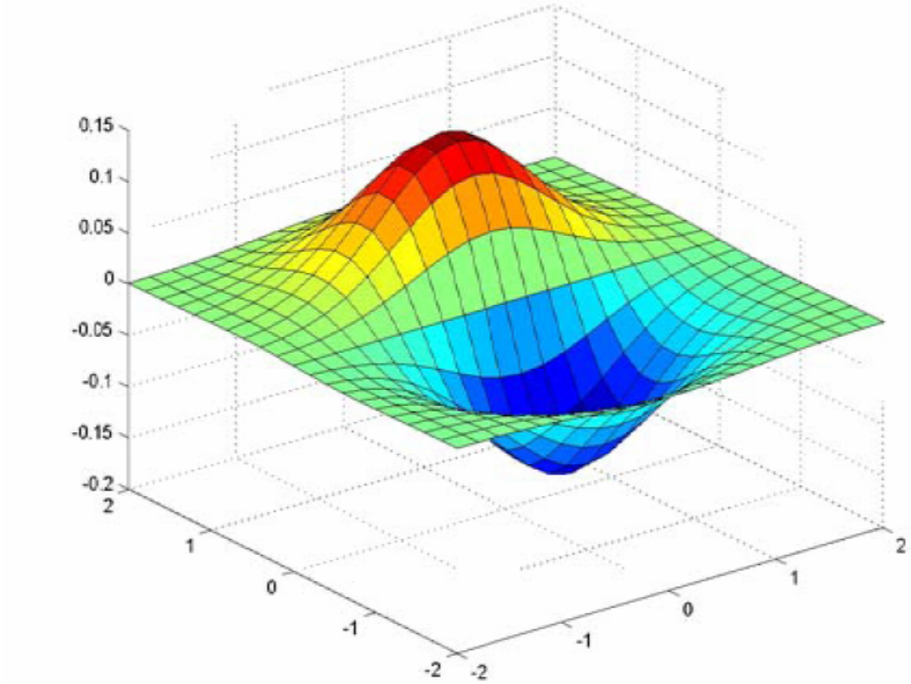


Slide credit: S. Seitz

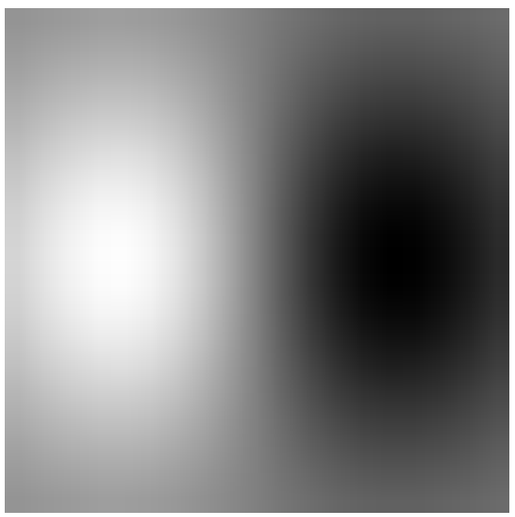
Derivative of Gaussian filter

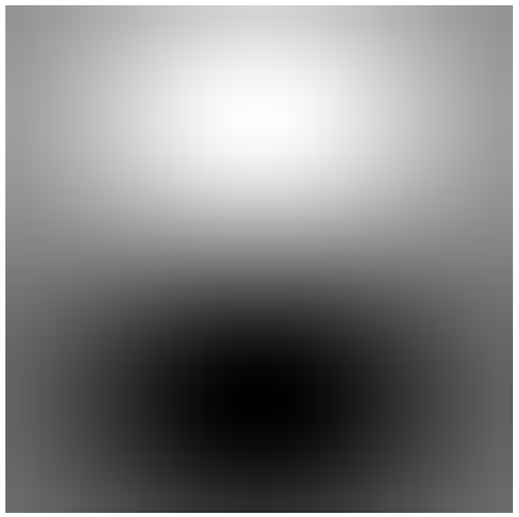


x-direction

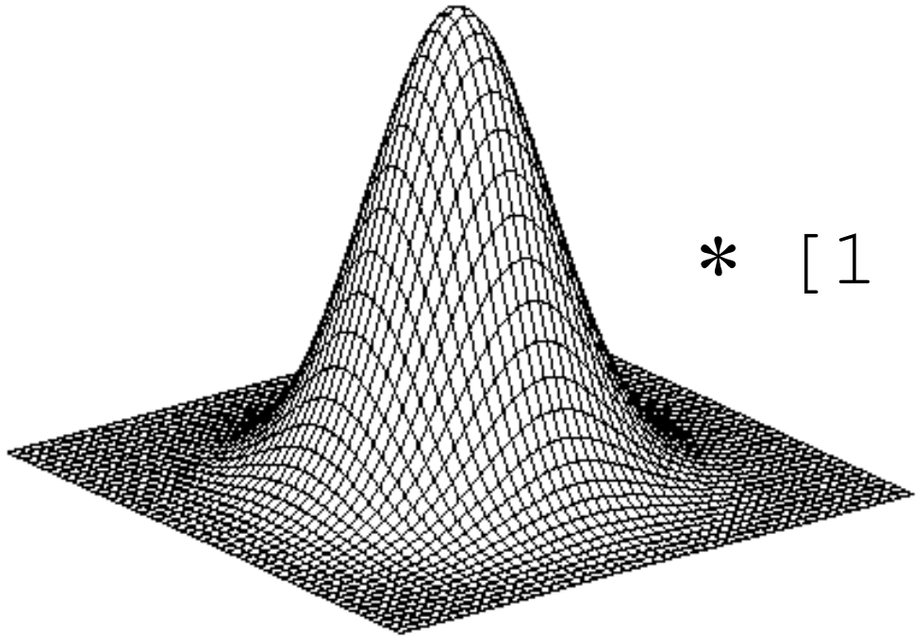


y-direction

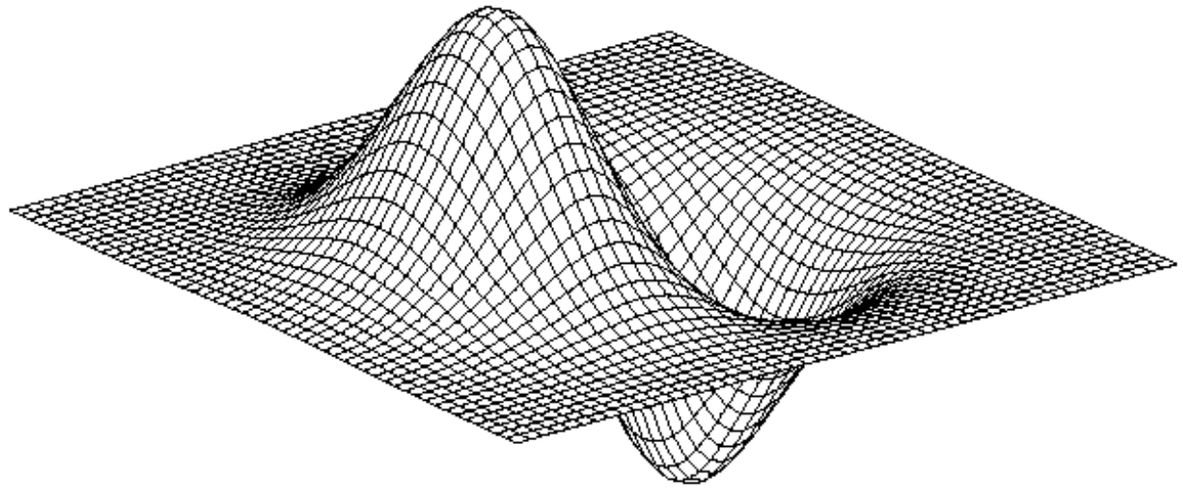




Derivative of Gaussian filter



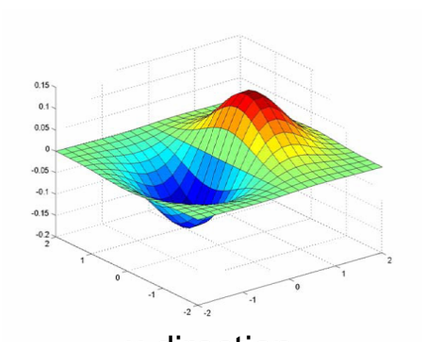
* [1 0 -1] =



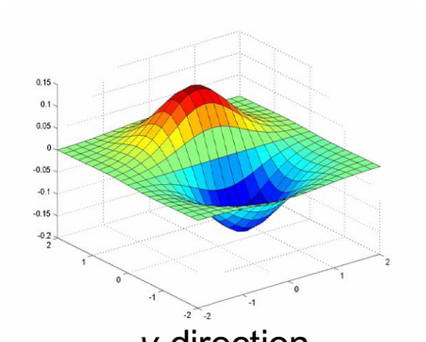
2D Gaussian

x-derivative

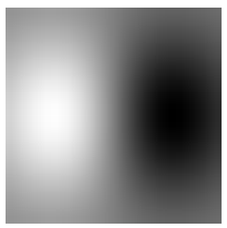
Derivative of Gaussian filter

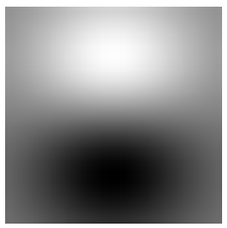


x-direction



y-direction





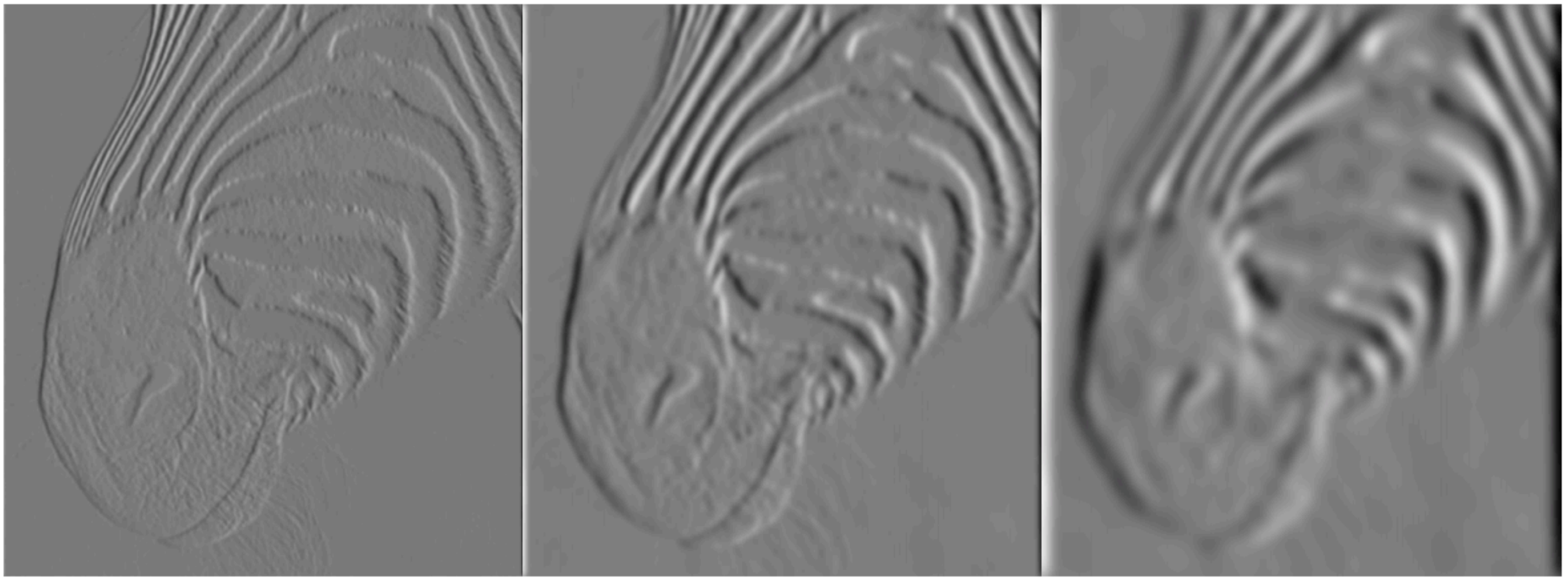
Note: separability of the Gaussian Filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

- The 2D Gaussian can be expressed as the product of 2 functions (one a function of *x* and the other a function of *y*)
- In this case the two functions are the (identical) 1D Gaussian

Scale of Gaussian derivative filter

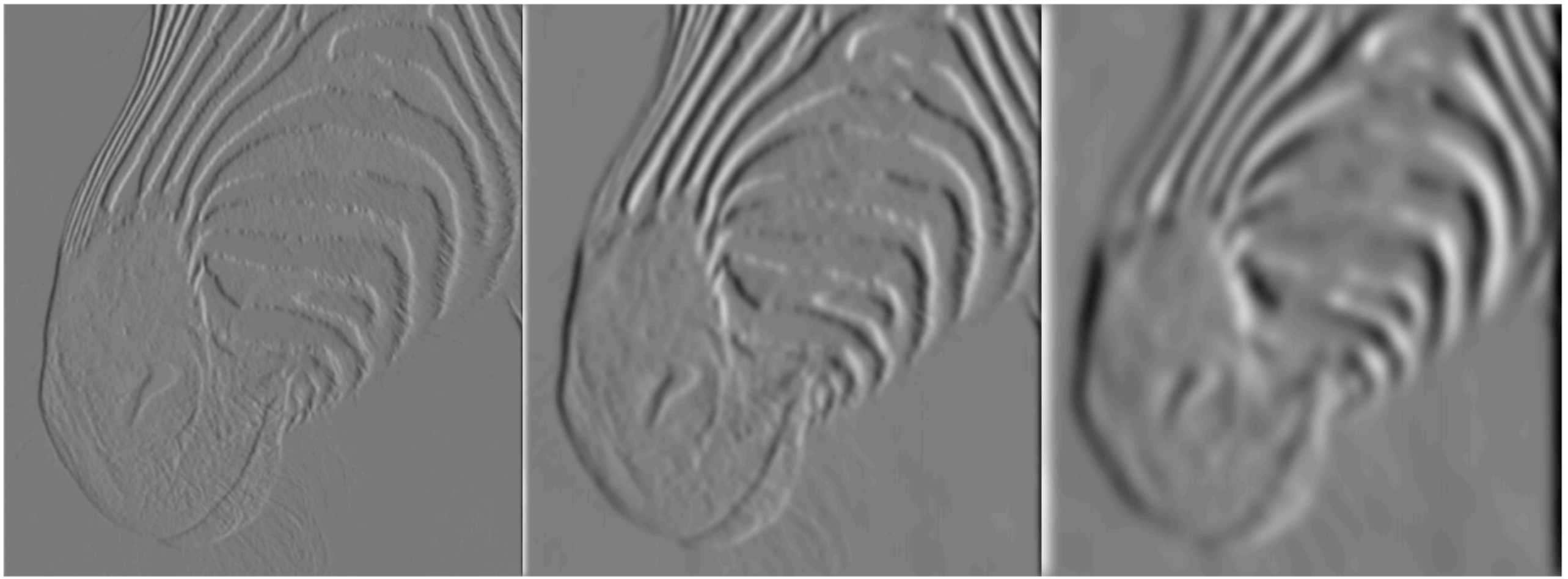


1 pixel 3 pixels 7 pixels

- Note: σ is the *scale* (*width*/*spread*) of the Gaussian kernel
- Smoothed derivative remove noise, but blurs edges

Image credit: D. Forsyth

Scale of Gaussian derivative filter



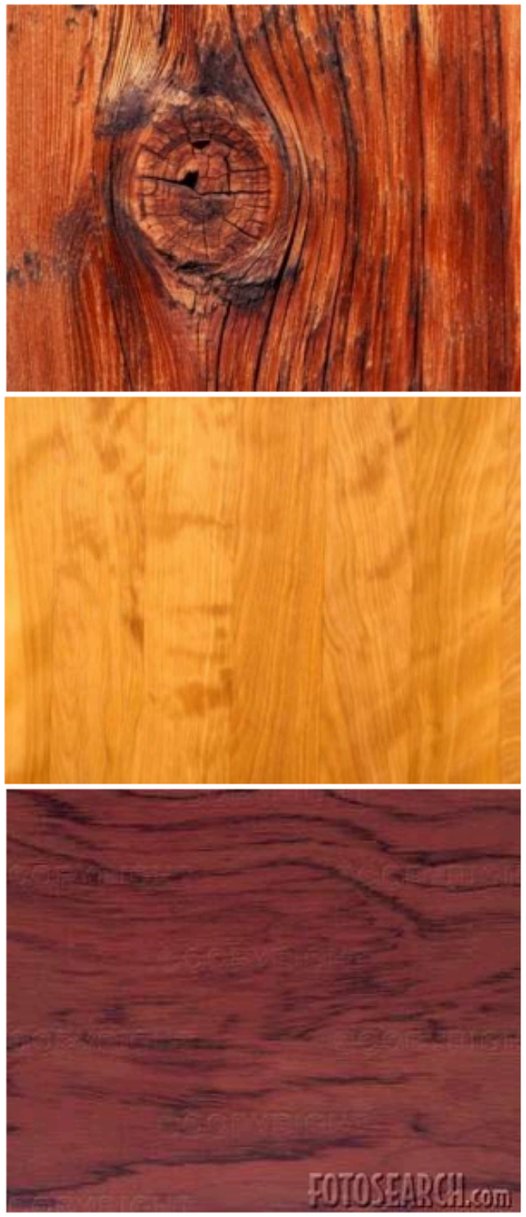
1 pixel 3 pixels 7 pixels

- The apparent structures differ depending on scale σ
  - Larger values: larger scale edges detected
  - Smaller values: finer features detected

Image credit: D. Forsyth

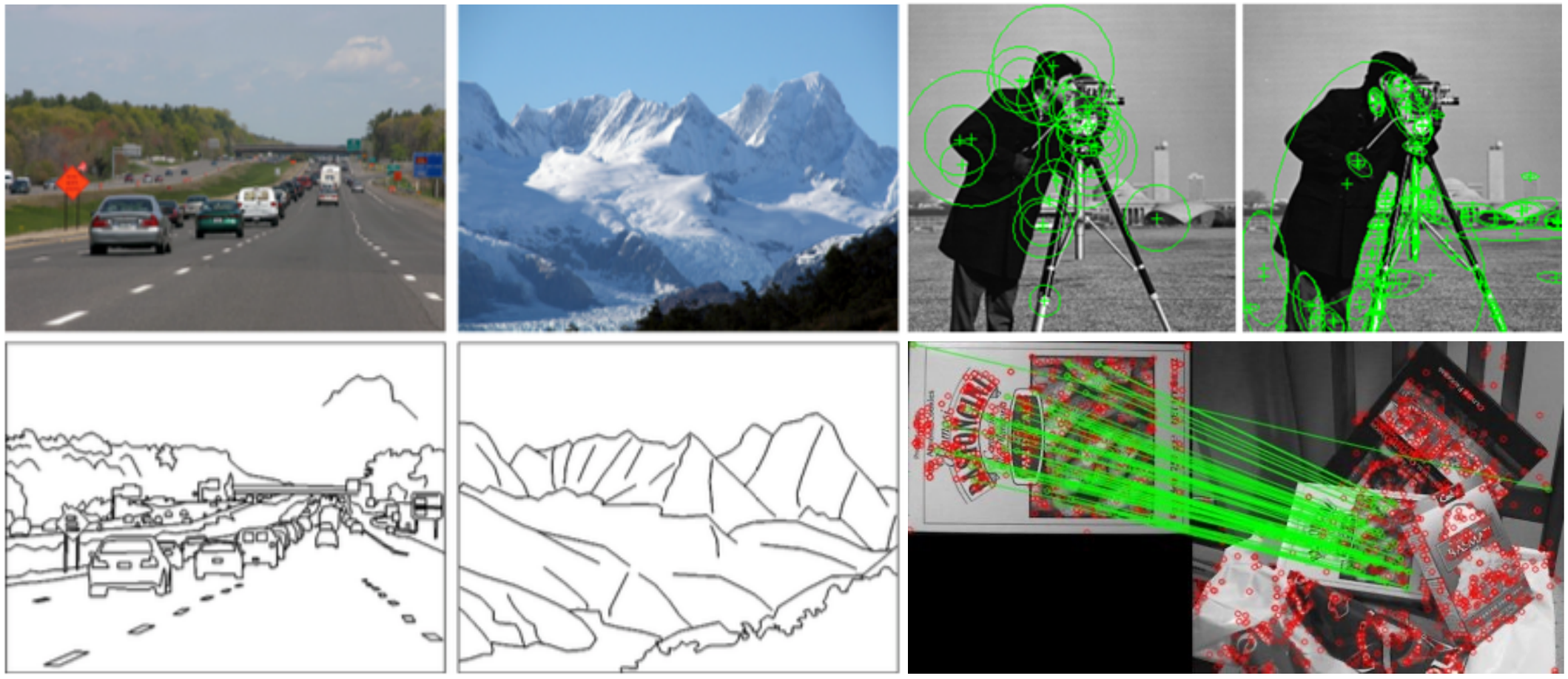
So, what scale to choose?

- It depends what we are looking for…



Coming up

- **Next lectures:**
  - Lab on image filtering
  - Image gradients, edges, feature detectors/descriptors



Contact

- **Office:** Torre Archimede, room 6CD3
- **Office hours** (ricevimento)**:** Friday 9:00-11:00

lamberto.ballan@unipd.it
http://www.lambertoballan.net
http://vimp.math.unipd.it
@ lambertoballan.bsky.social