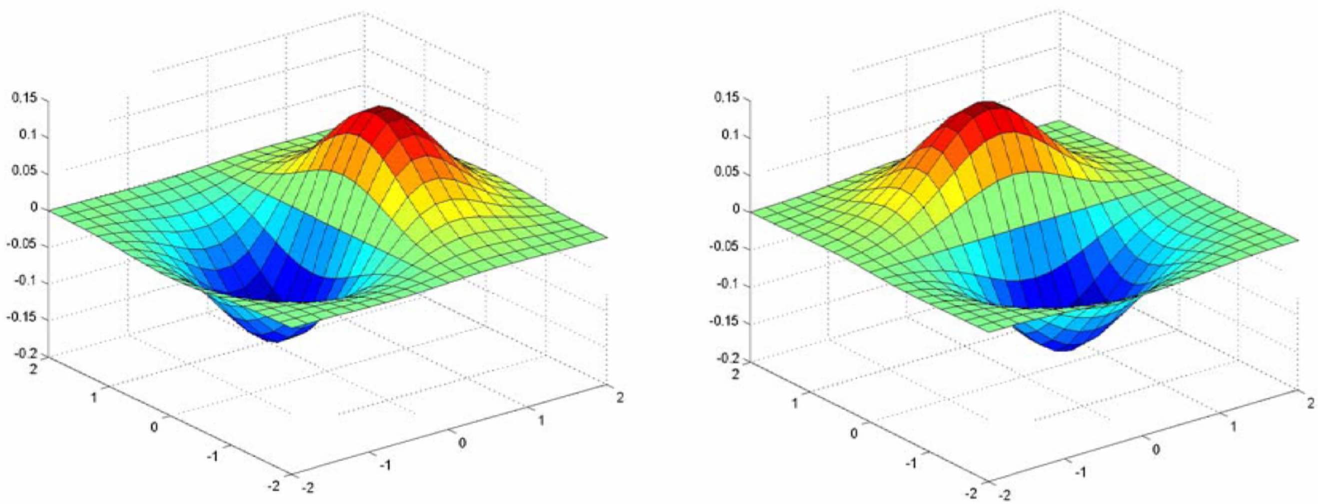
CSCI 497/597P: Computer Vision

Scott Wehrwein

Edge Detection, Continued



Reading

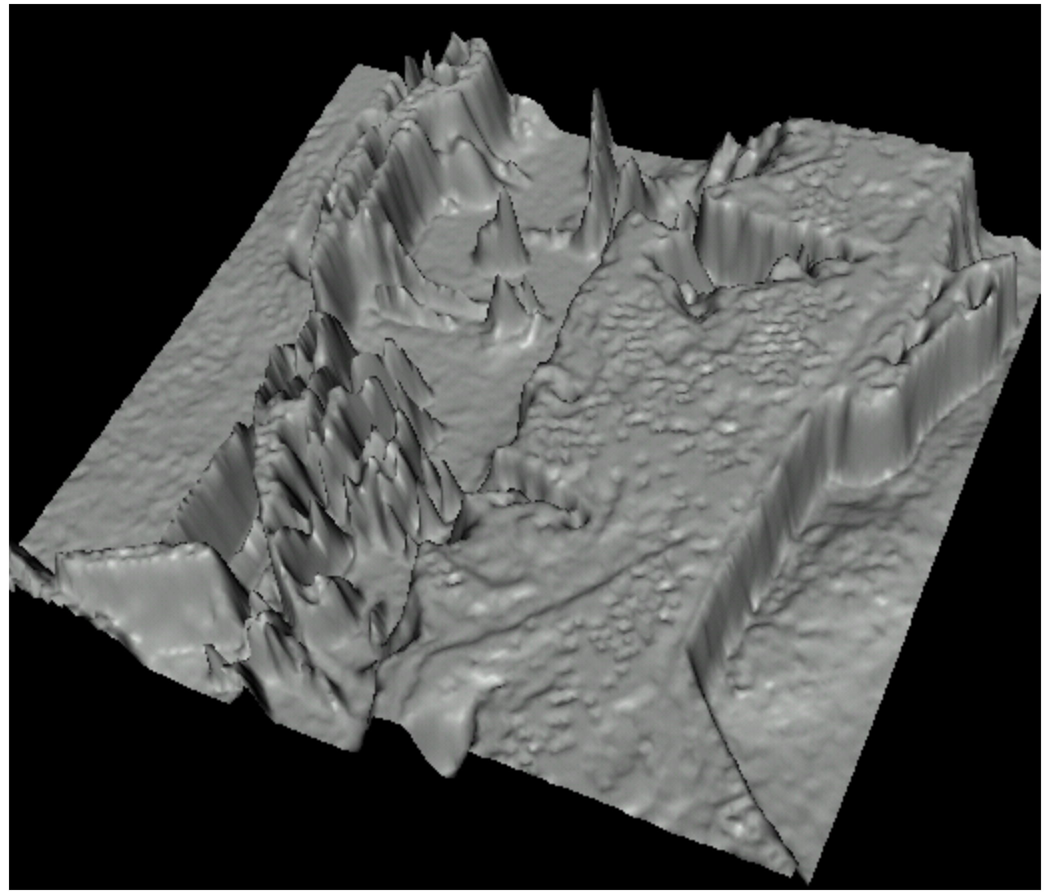
- Szeliski, Chapter 4.2

Announcements

Goals

- Understand the basics of edge detection:
  - The sobel operator as an approximation of the image gradient in the presence of noise.
- Understand the use of non-maximum suppression to localize edges from smoothed gradients.

Images as functions…

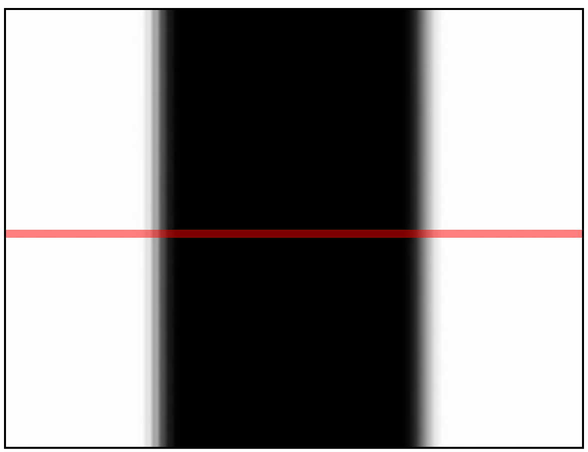


- Edges look like steep cliffs

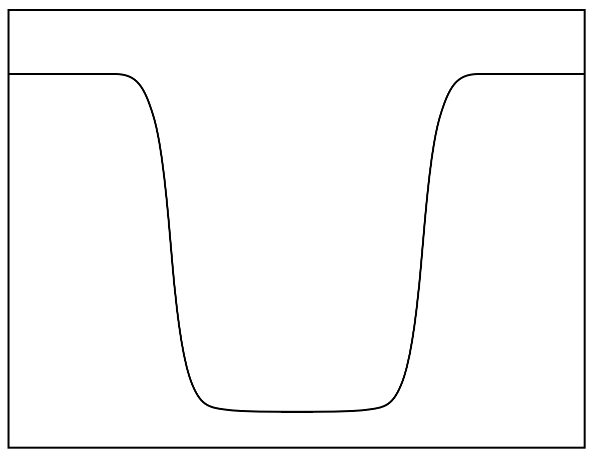
Characterizing edges

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

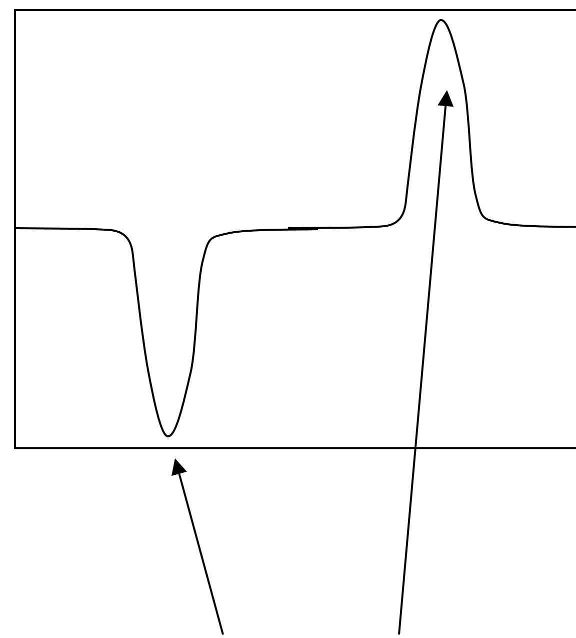
image



intensity function
(along horizontal scanline)



first derivative



edges correspond to
extrema of derivative

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?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

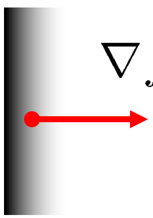
H_y

Source: S. Seitz

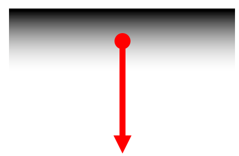
Image gradient

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

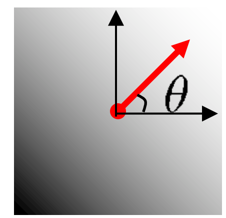
The gradient points in the direction of most rapid increase in intensity



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



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



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

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

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

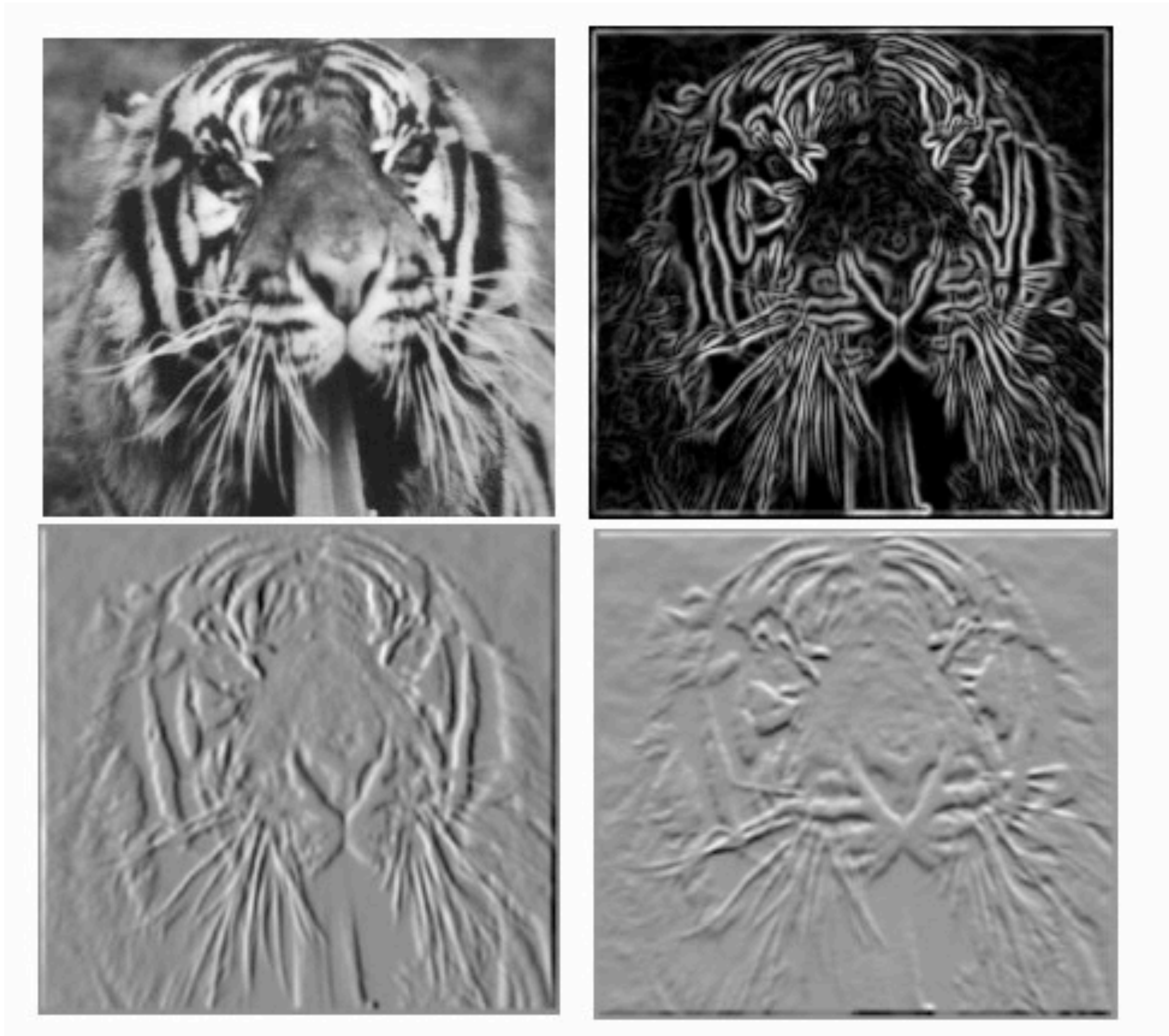
The gradient direction is given by:

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

- how does this relate to the direction of the edge?

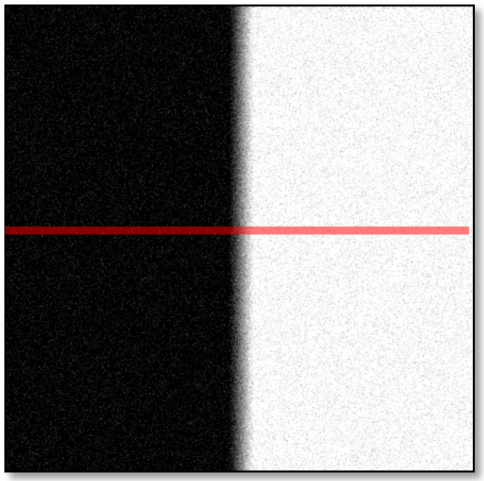
Source: Steve Seitz

Image gradient



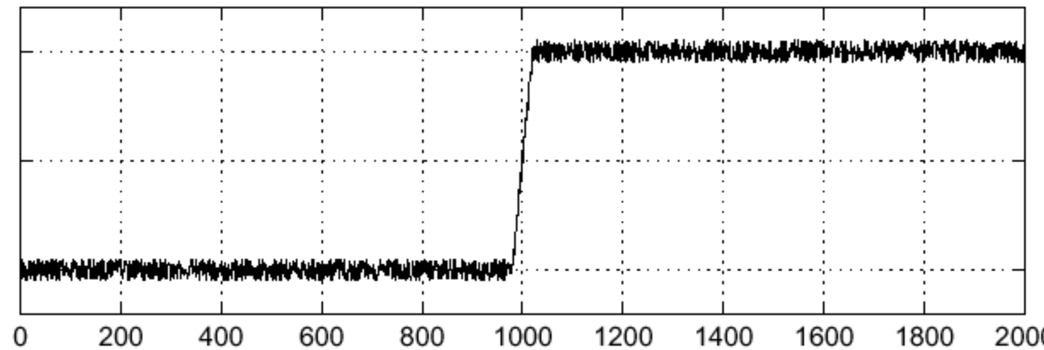
Source: L. Lazebnik

Effects of noise

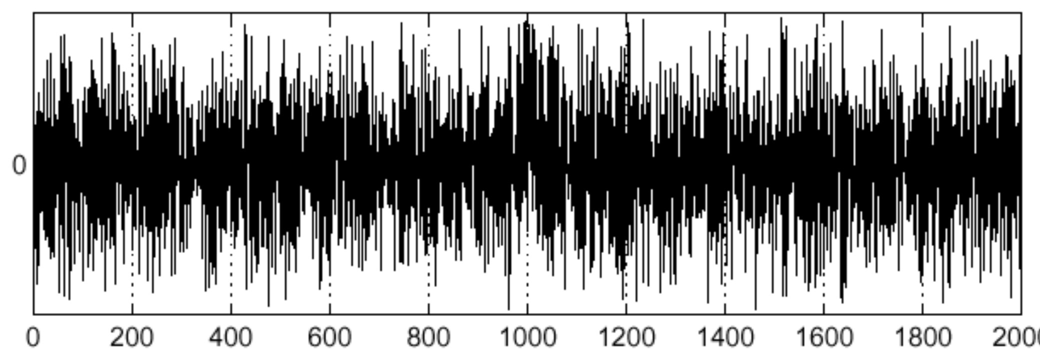


Noisy input image

$$f(x)$$



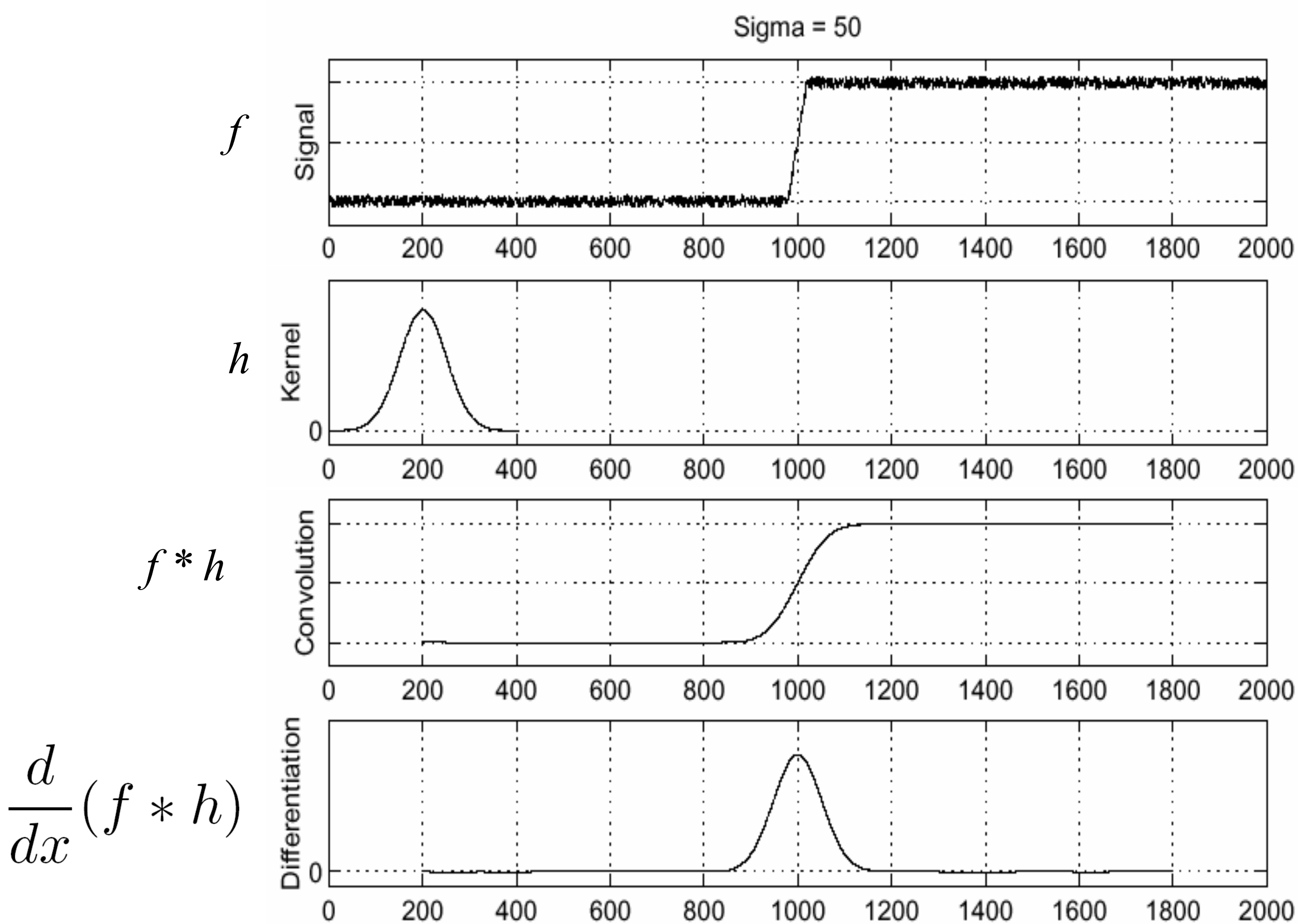
$$\frac{d}{dx}f(x)$$



Where is the edge?

Source: S. Seitz

Solution: smooth first



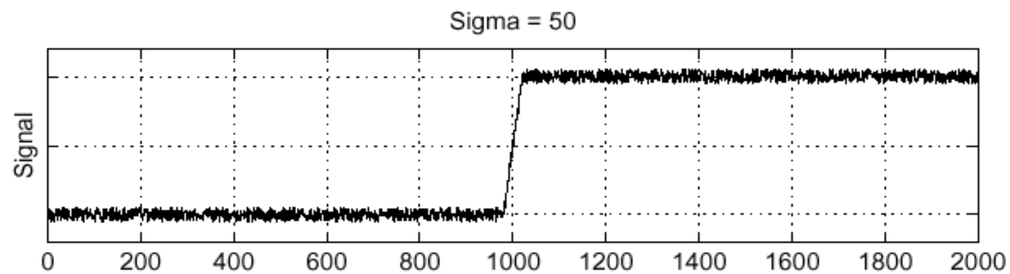
To find edges, look for peaks in $\frac{d}{dx}(f * h)$

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

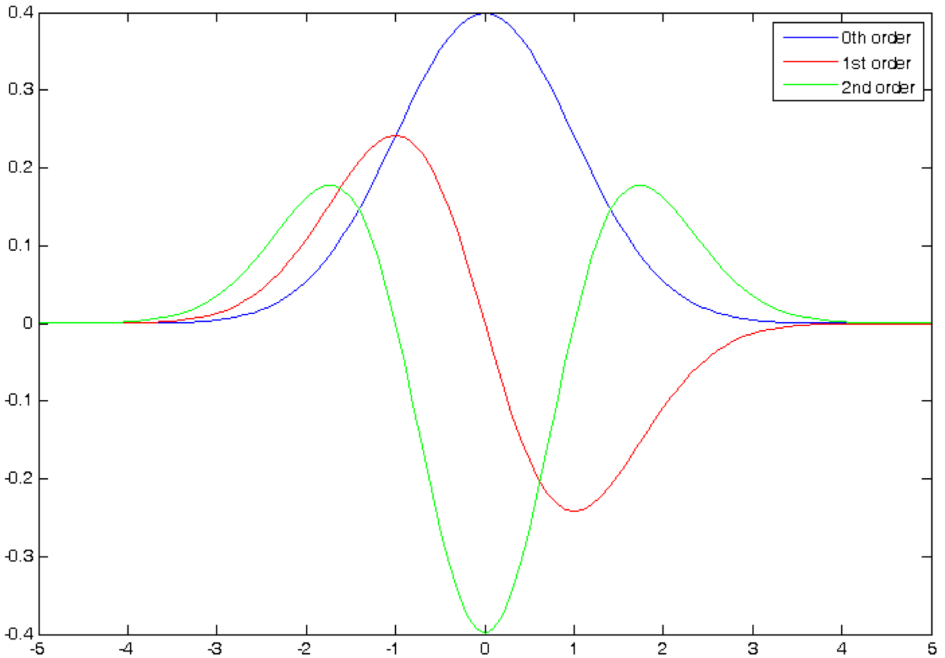


Source: S. Seitz

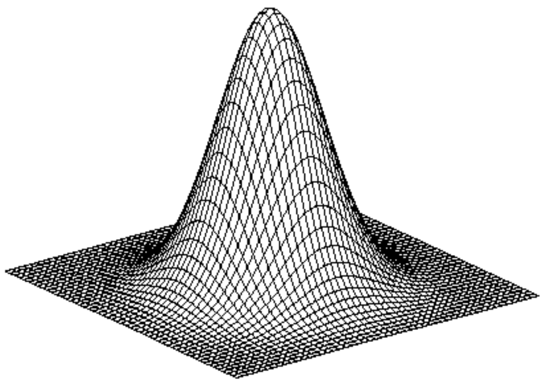
The 1D Gaussian and its derivatives

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

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

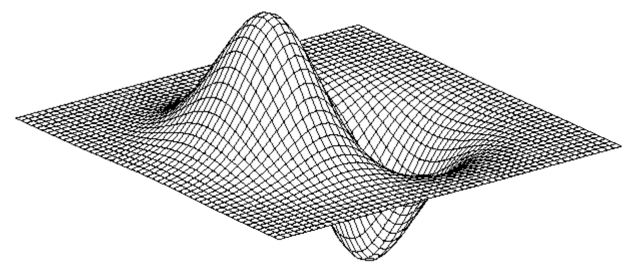


2D edge detection filters



Gaussian

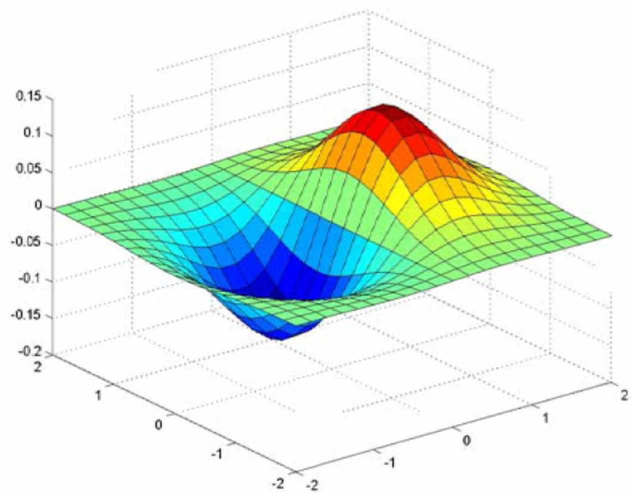
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



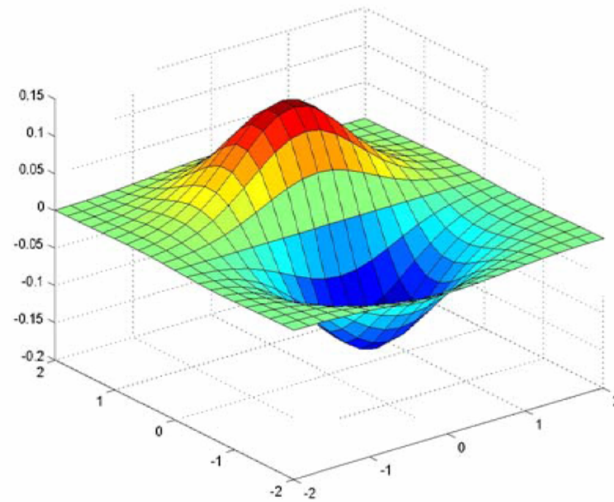
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

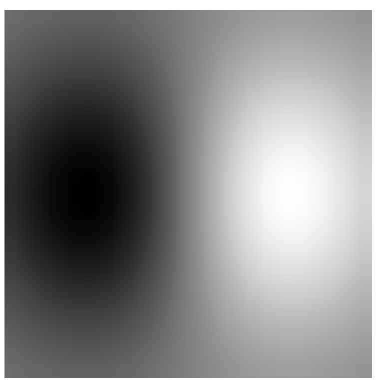
Derivative of Gaussian filter

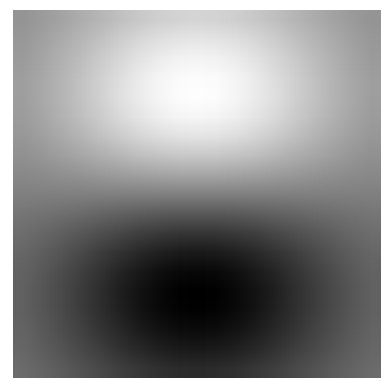


x-direction



y-direction





The Sobel operator

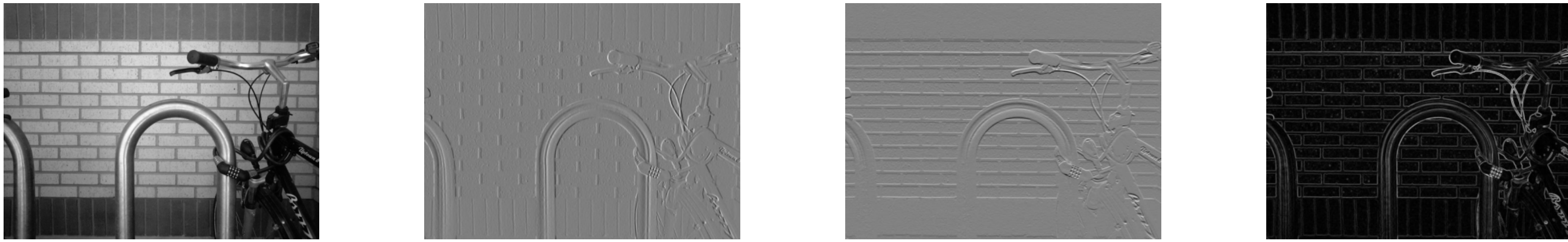
- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$s_x \qquad\qquad s_y$$

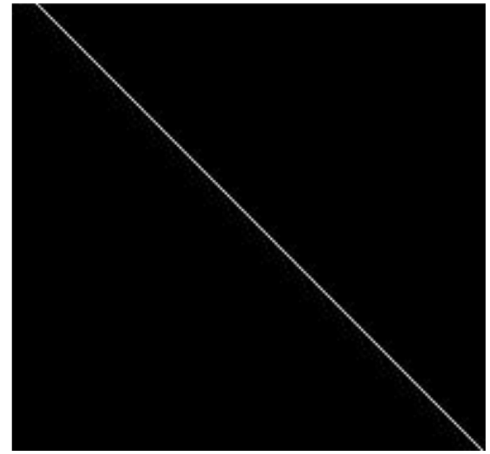
- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example



Source: Wikipedia

Image with Edge



Edge Location

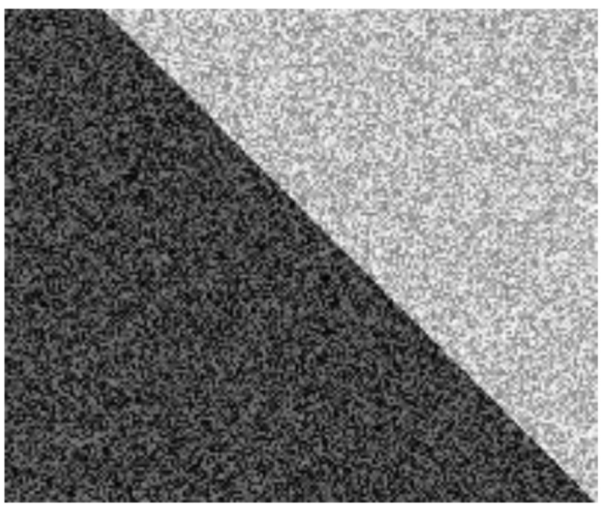
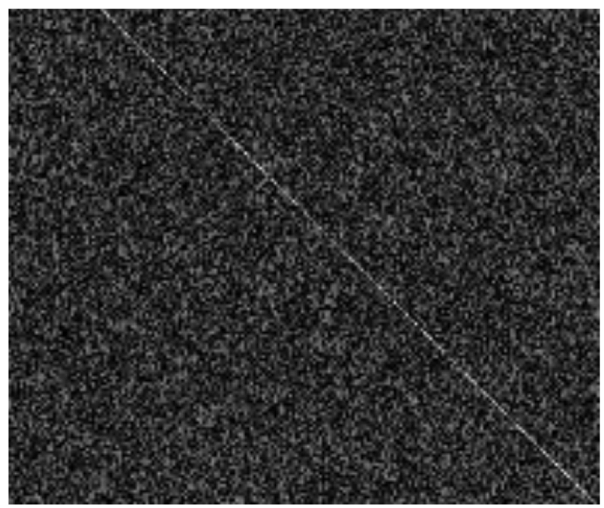
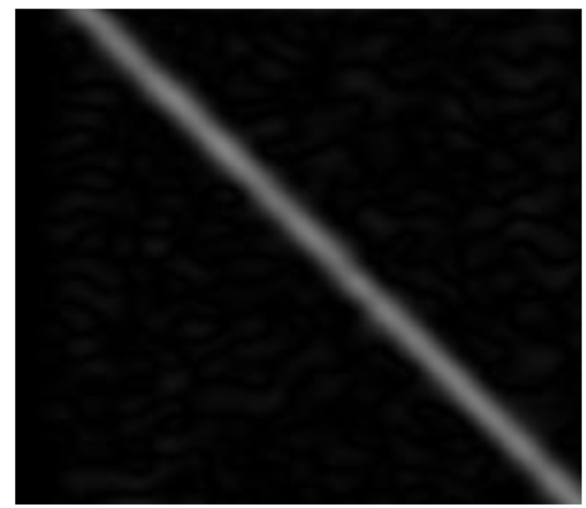


Image + Noise



Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

Criteria for a good boundary detector

- Criteria for a good boundary detector:
  - **Good detection:** Fire only on real edges, not anywhere else
  - **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

Source: L. Fei-Fei

Canny edge detector

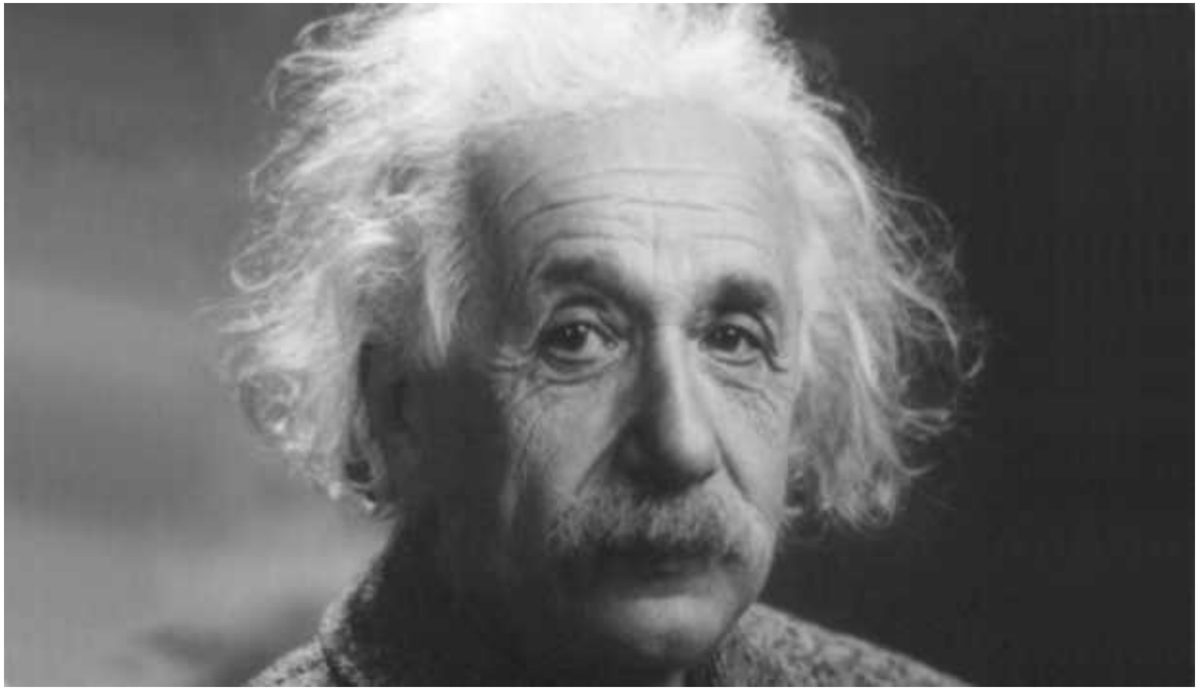
- The classic edge detector
- Baseline for all later work on grouping
- Theoretical model: step-edges corrupted by additive Gaussian noise

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

22,000 citations!

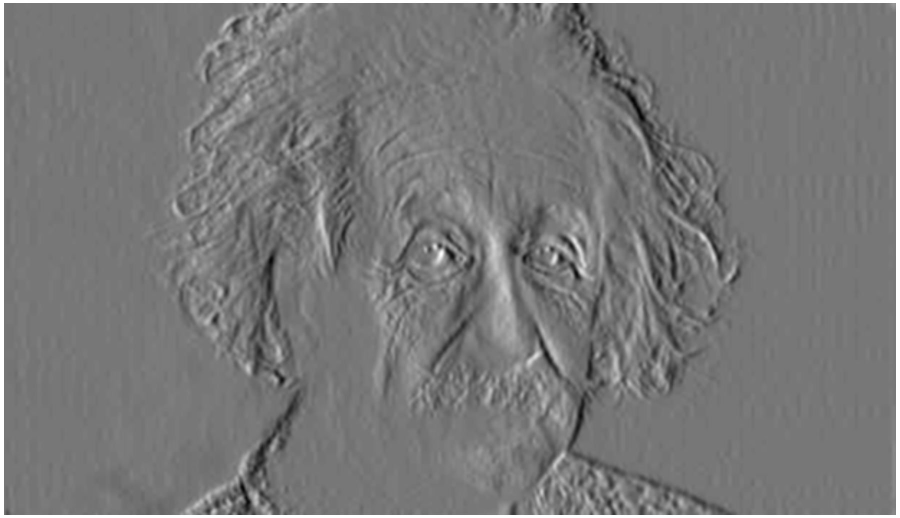
Source: L. Fei-Fei

Example

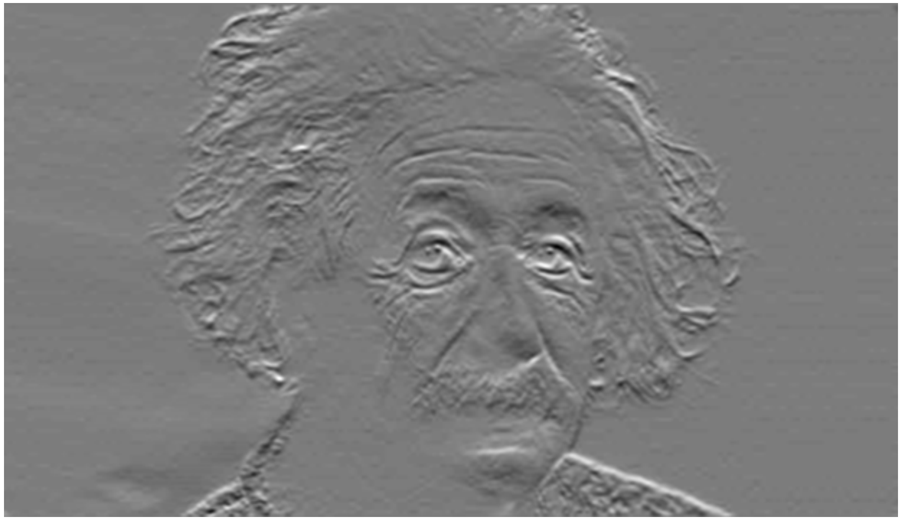


original image

Compute Gradients (DoG)



X-Derivative of Gaussian



Y-Derivative of Gaussian

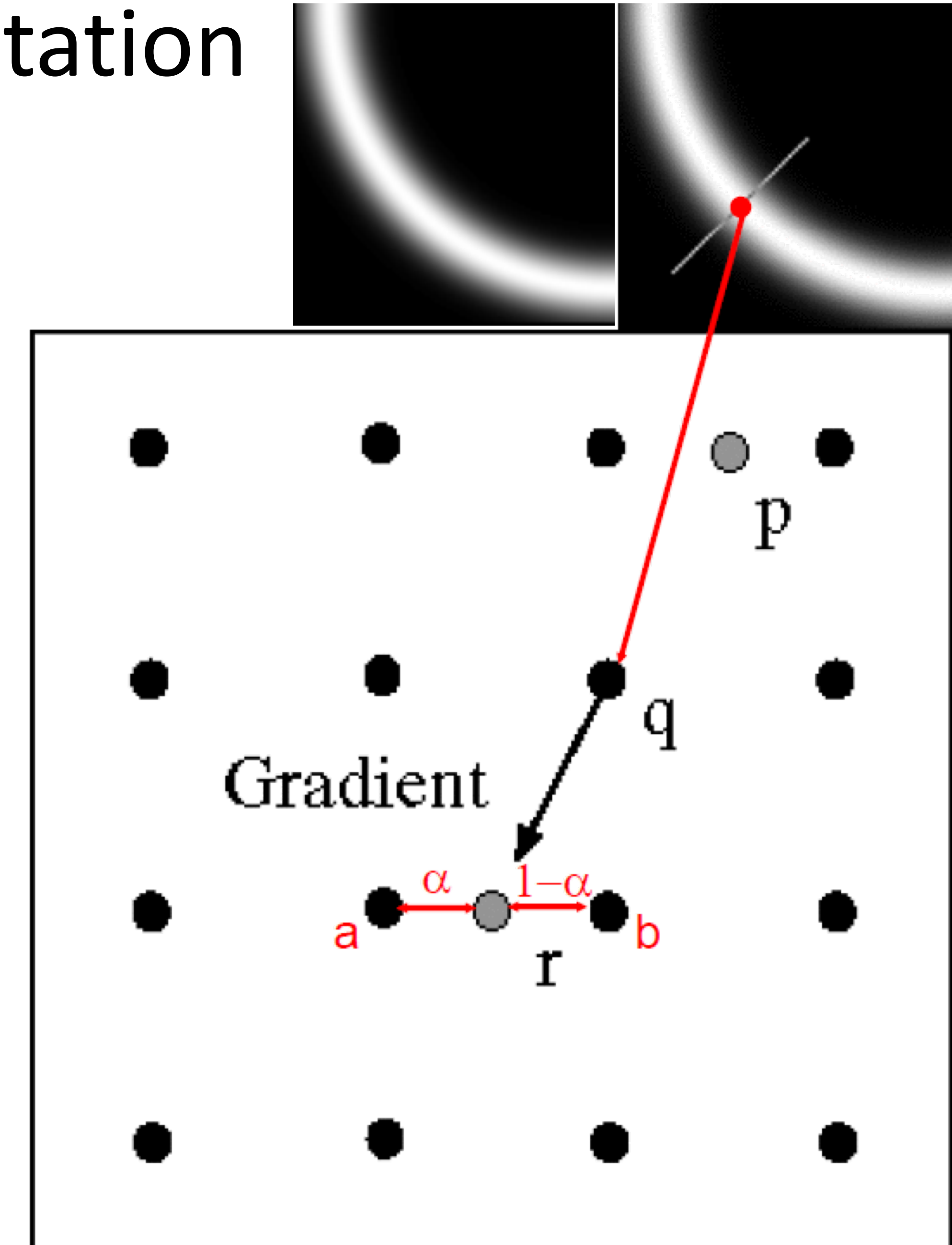
Gradient magnitude and orientation

- Orientation is undefined at pixels with 0 gradient

Magnitude

Orientation
theta = numpy.arctan2(gy, gx)

Non-maximum suppression for each orientation



Pixel q is a maximum if it is larger than p and r

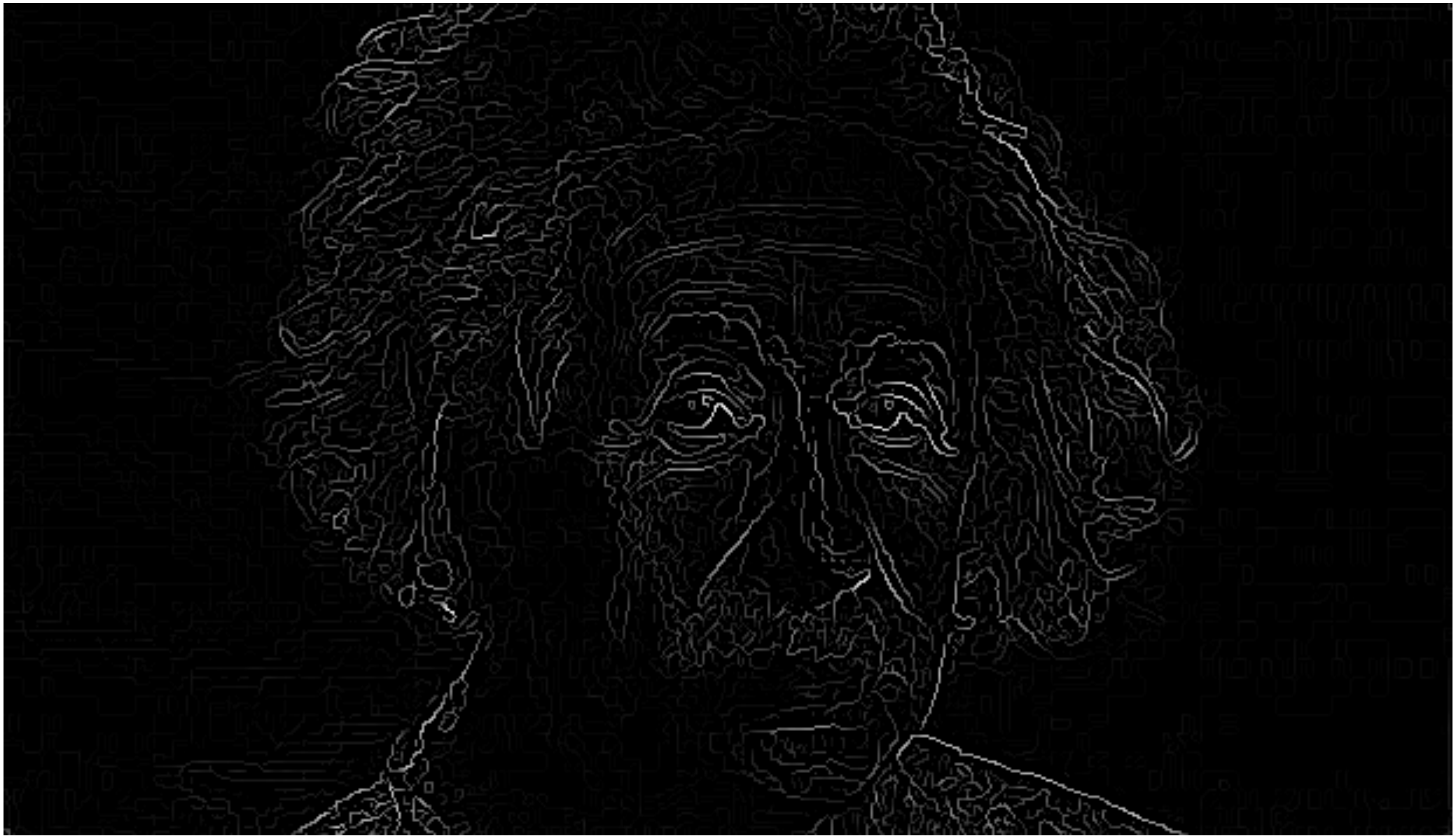
p and r’s locations are determined by the gradient orientation

Their intensity is determined by linear interpolation.

Source: D. Forsyth

Before Non-max Suppression

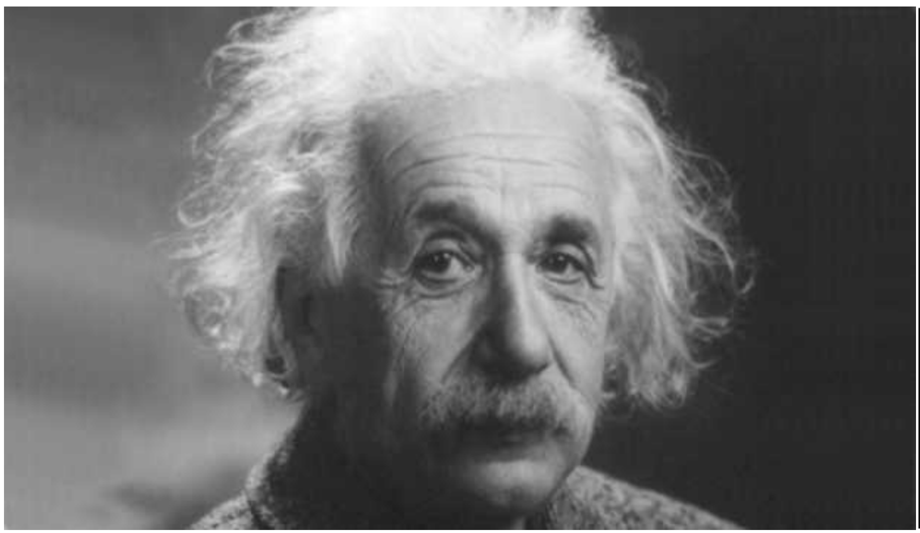
After Non-max Suppression



Hysteresis thresholding

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

Final Canny Edges

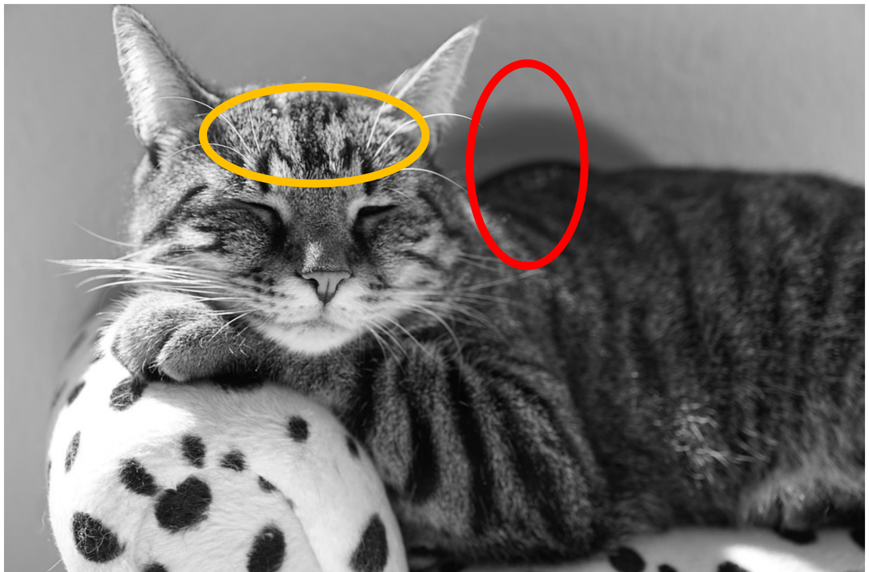
 

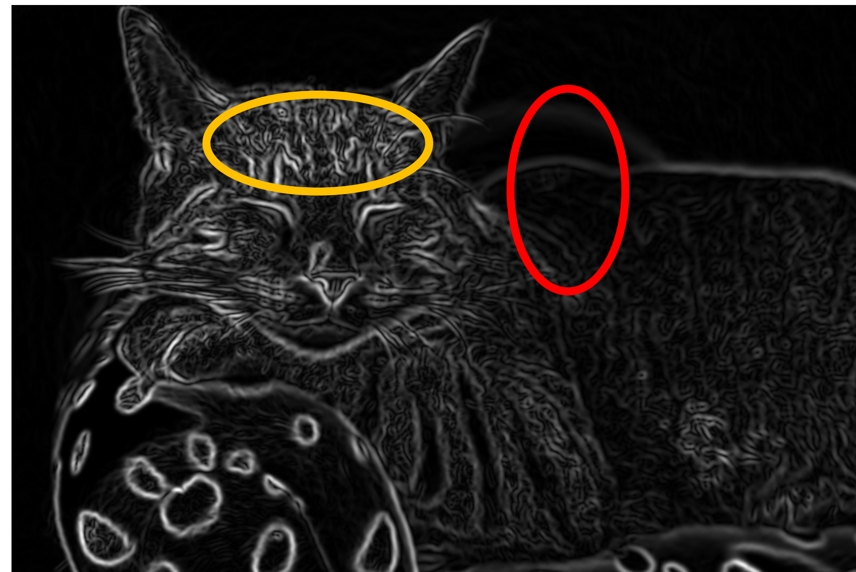
Canny edge detector

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

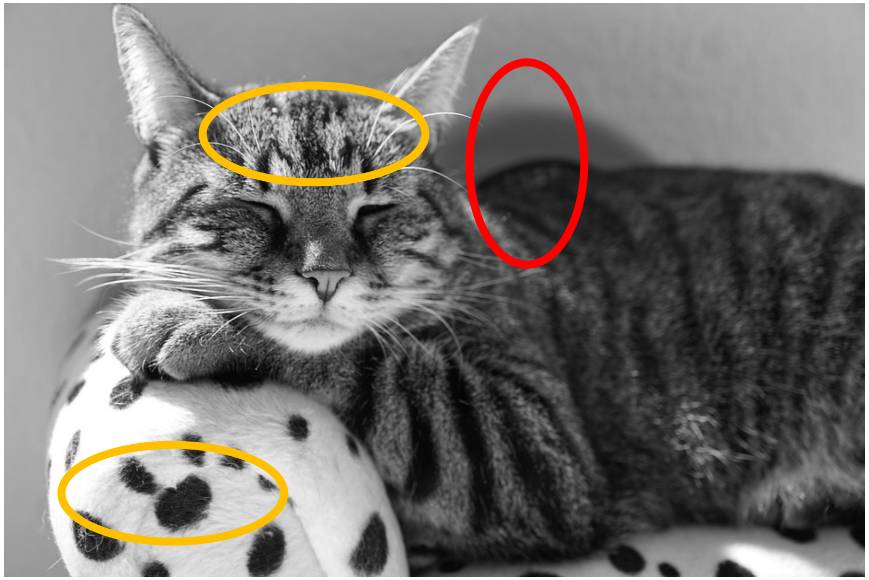
Source: D. Lowe, L. Fei-Fei

Does Canny always work?



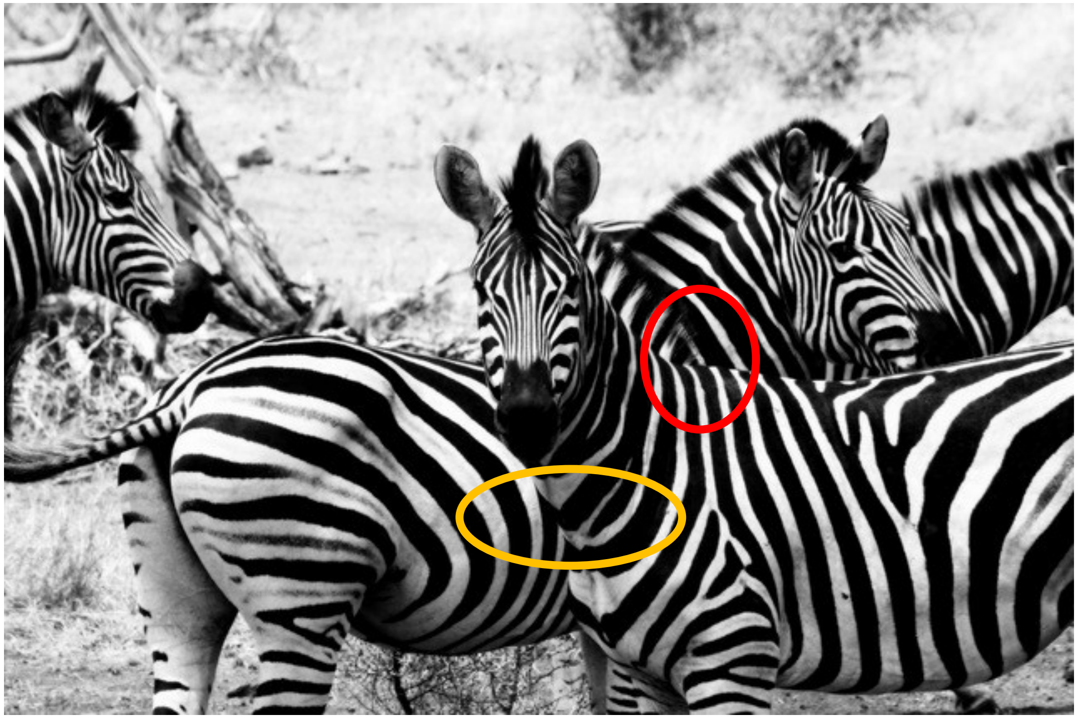


The challenges of edge detection



- Texture
- Low-contrast boundaries

The challenges of edge detection

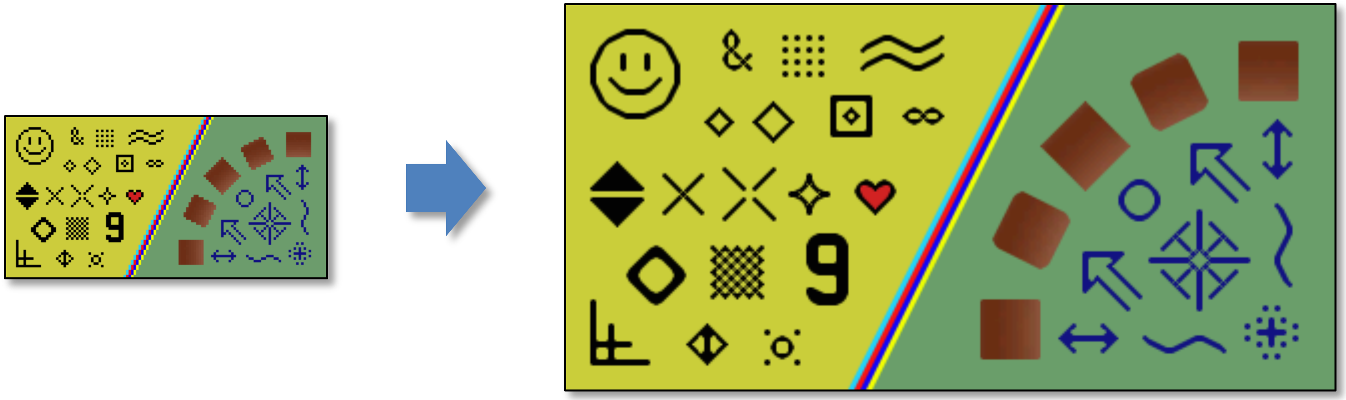


- Higher-level information

CSCI 497/597P: Computer Vision

Scott Wehrwein

Image Resampling & Interpolation



Reading

- Szeliski, Chapter 3.5

Announcements

Goals

- Know how to downsample an image naively
- Gain some intuition for why that's a bad idea
- Know how and why to build a Gaussian Pyramid
- Understand how to upsample an image naively
- Know how to use

Image

This image is too big to fit on the screen. How can we generate a half-sized version?

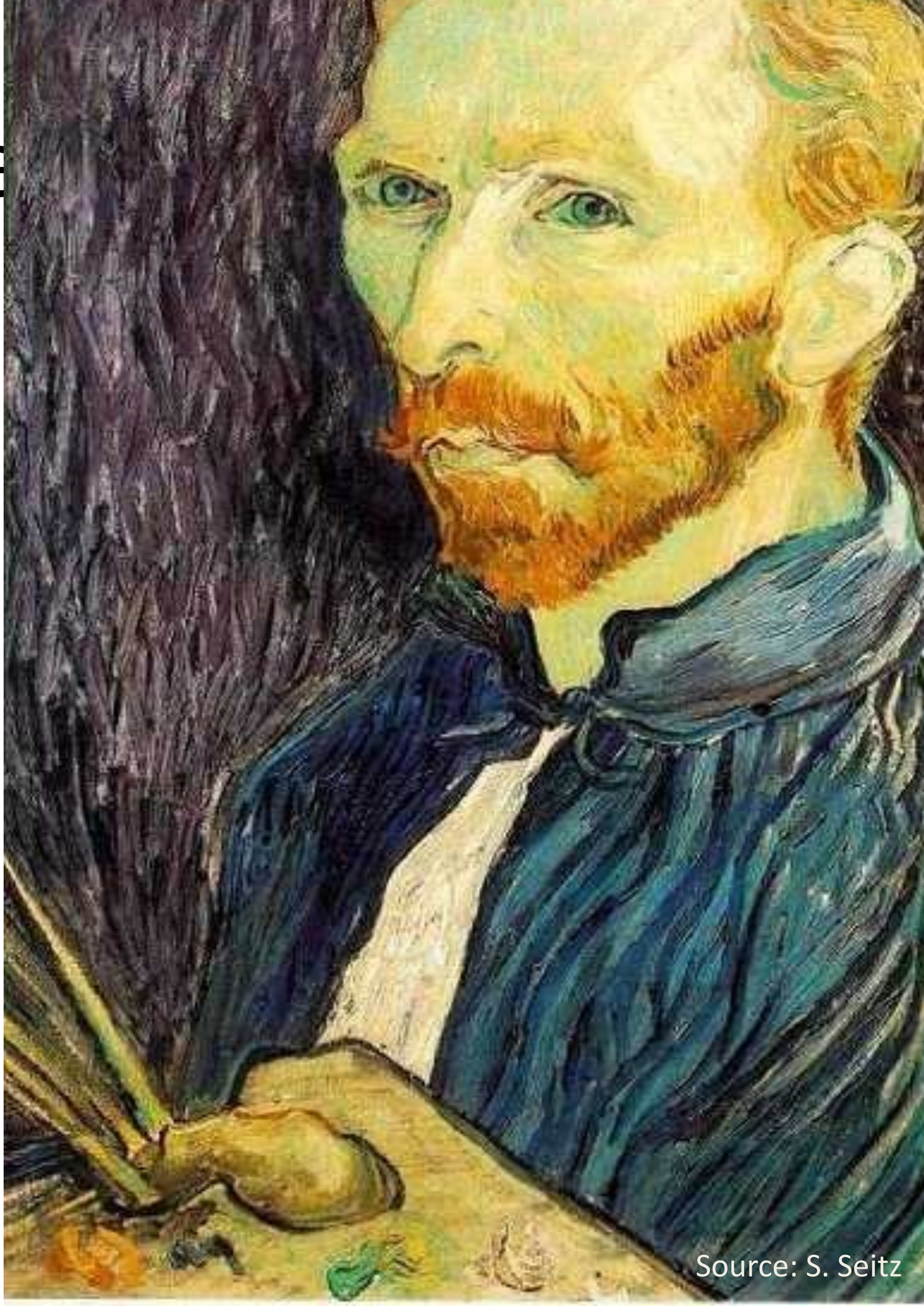
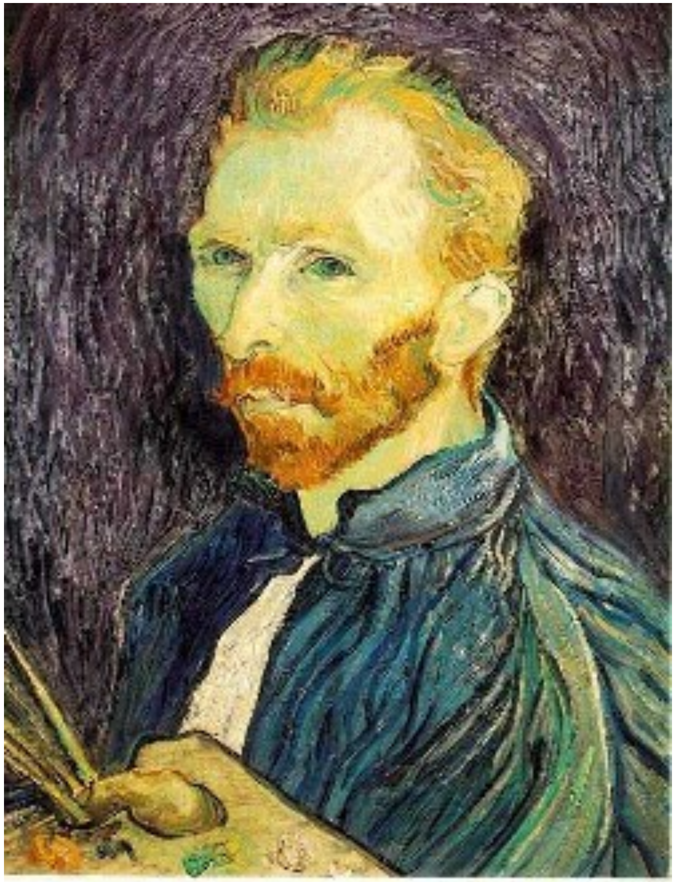


Image sub-sampling



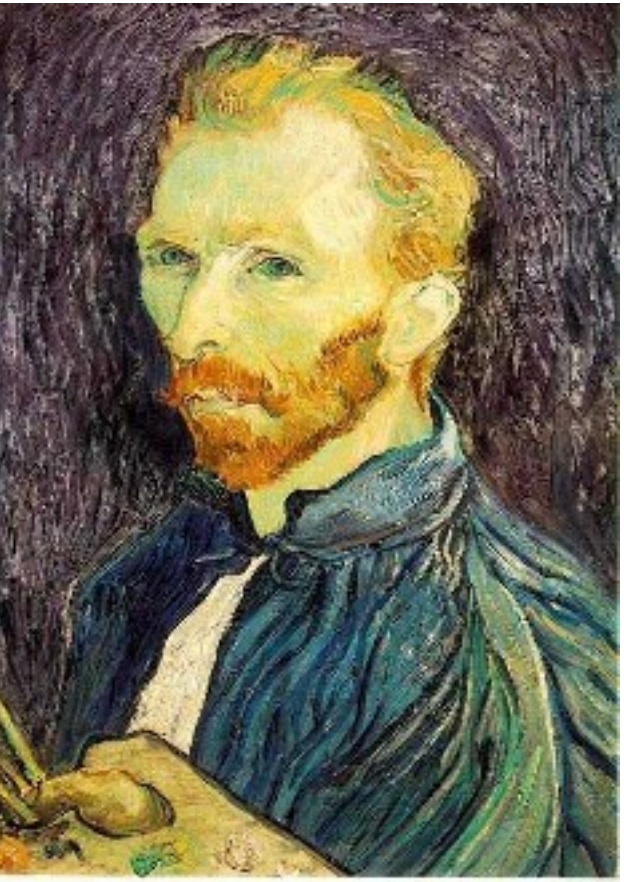
1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Source: S. Seitz

Image sub-sampling



1/2

1/4 (2x zoom)

1/8 (4x zoom)

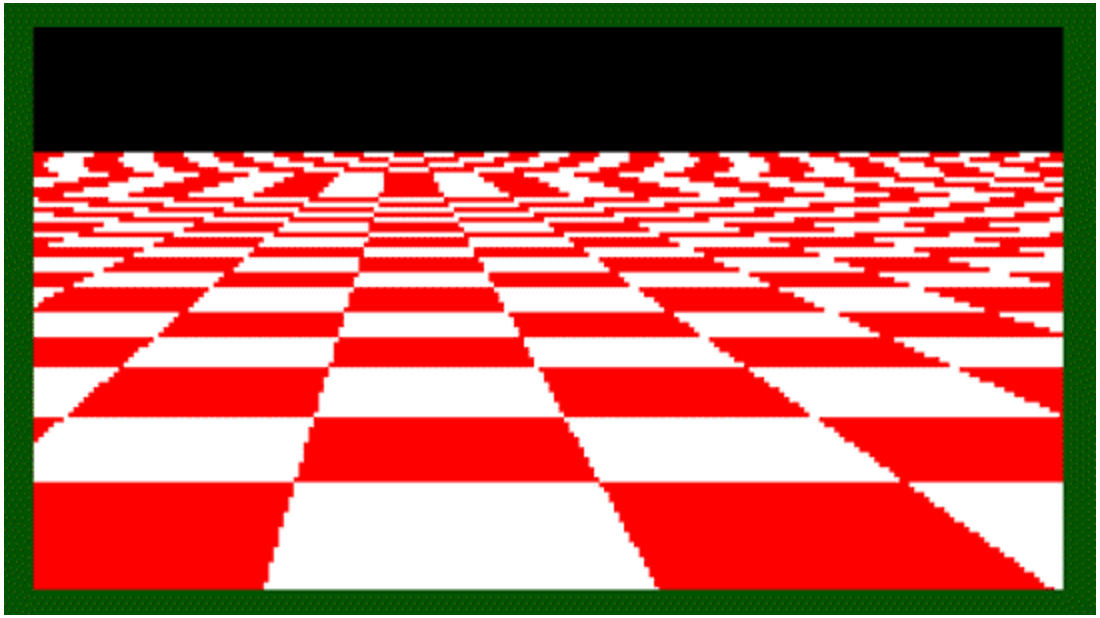
Why does this look so crufty?

Source: S. Seitz

Image sub-sampling – another example

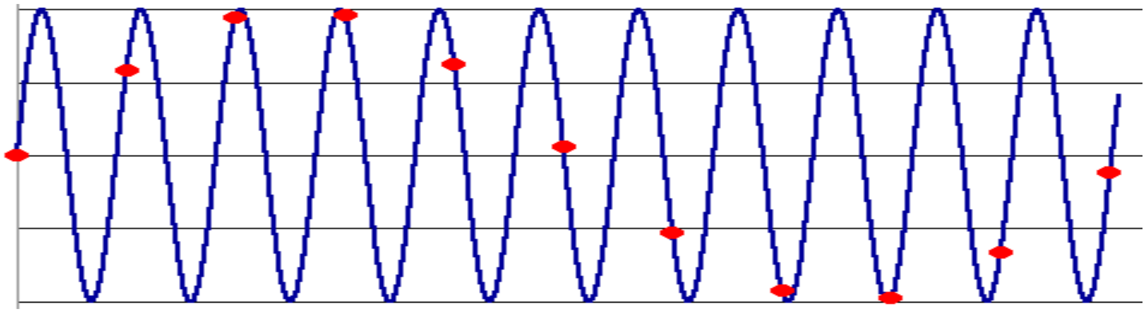
Source: F. Durand

Even worse for synthetic images



Source: L. Zhang

Aliasing



- Occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*

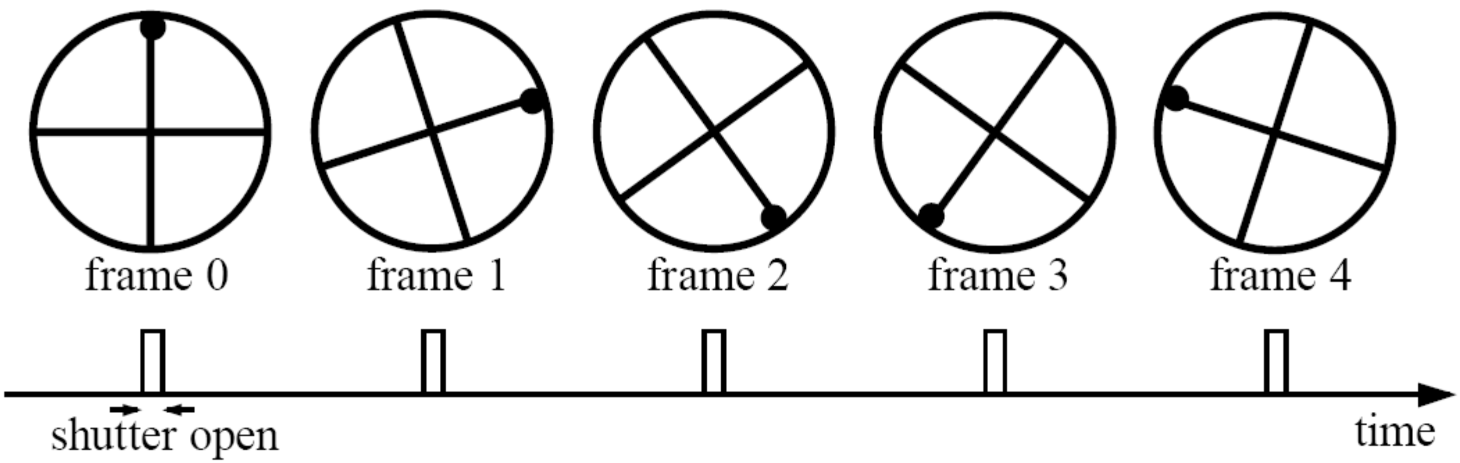
- To do sampling right, need to understand the structure of your signal/image
- Enter Monsieur Fourier…
  - “But what is the Fourier Transform? A visual introduction.” https://www.youtube.com/watch?v=spUNpyF58BY&t=444s
- To avoid aliasing:
  - sampling rate ≥ 2 * max frequency in the image
    - said another way: ≥ two samples per cycle
  - This minimum sampling rate is called the **Nyquist rate**

Source: L. Zhang

Wagon-wheel effect

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

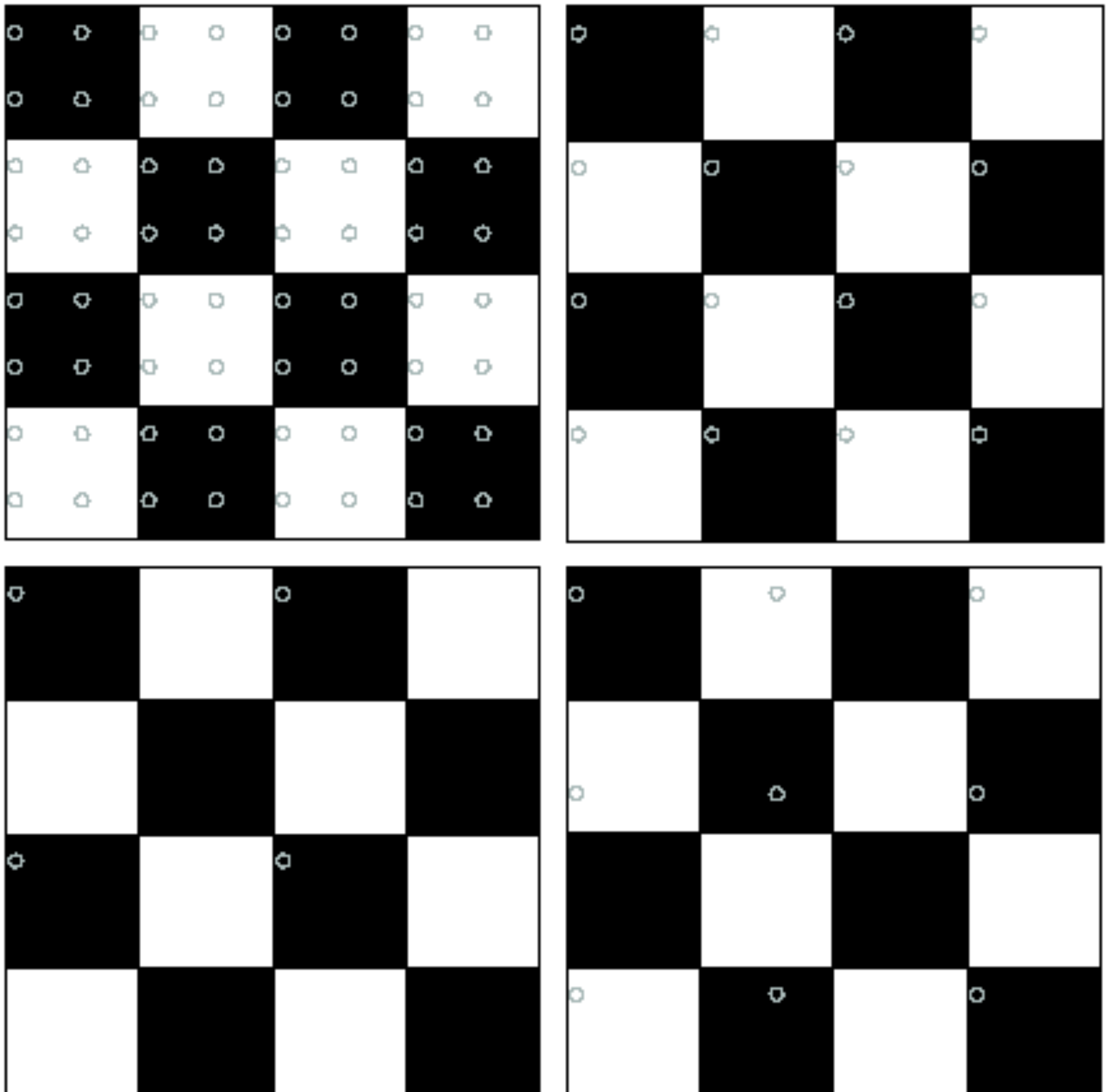


Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

http://www.michaelbach.de/ot/mot-wagonWheel/index.html
https://en.wikipedia.org/wiki/Wagon-wheel_effect

Source: L. Zhang

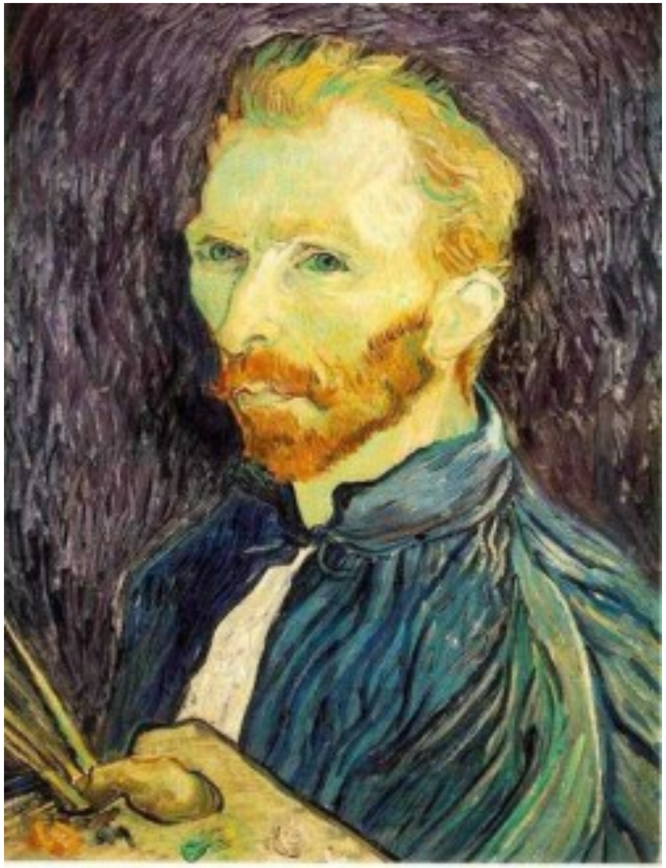
Nyquist limit – 2D example



Good sampling

Bad sampling

Gaussian pre-filtering



Gaussian 1/2

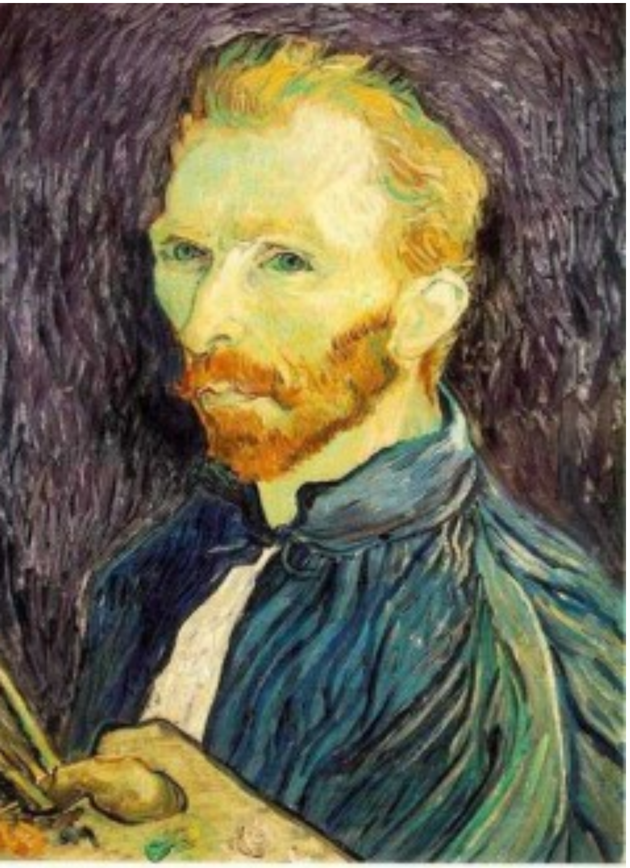
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

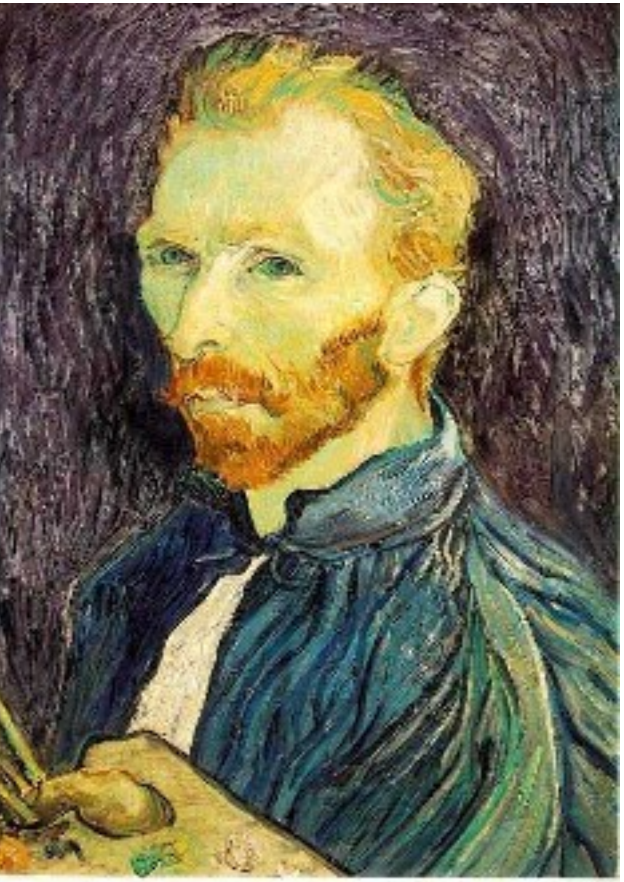
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Compare with...



1/2

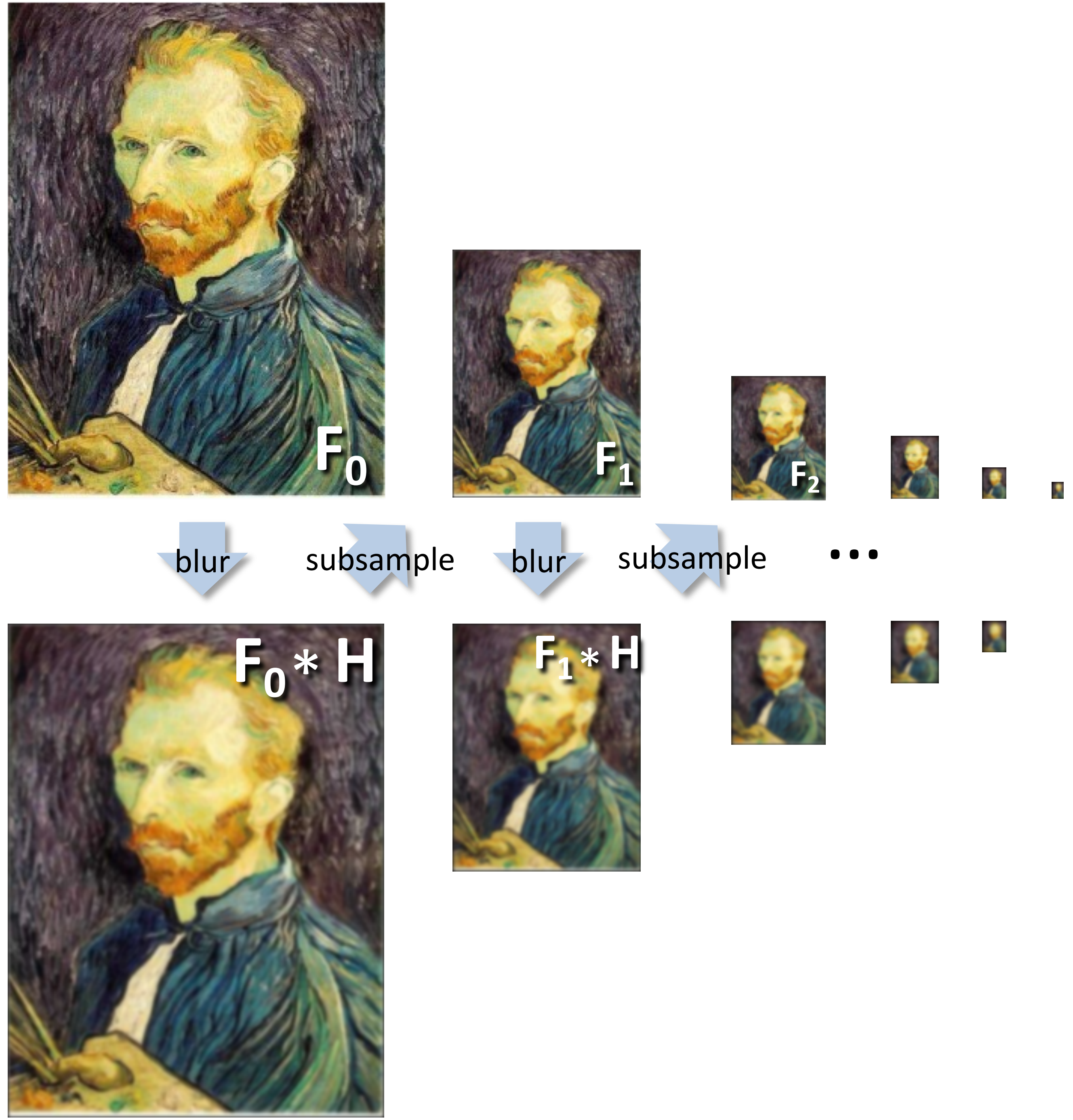
1/4 (2x zoom)

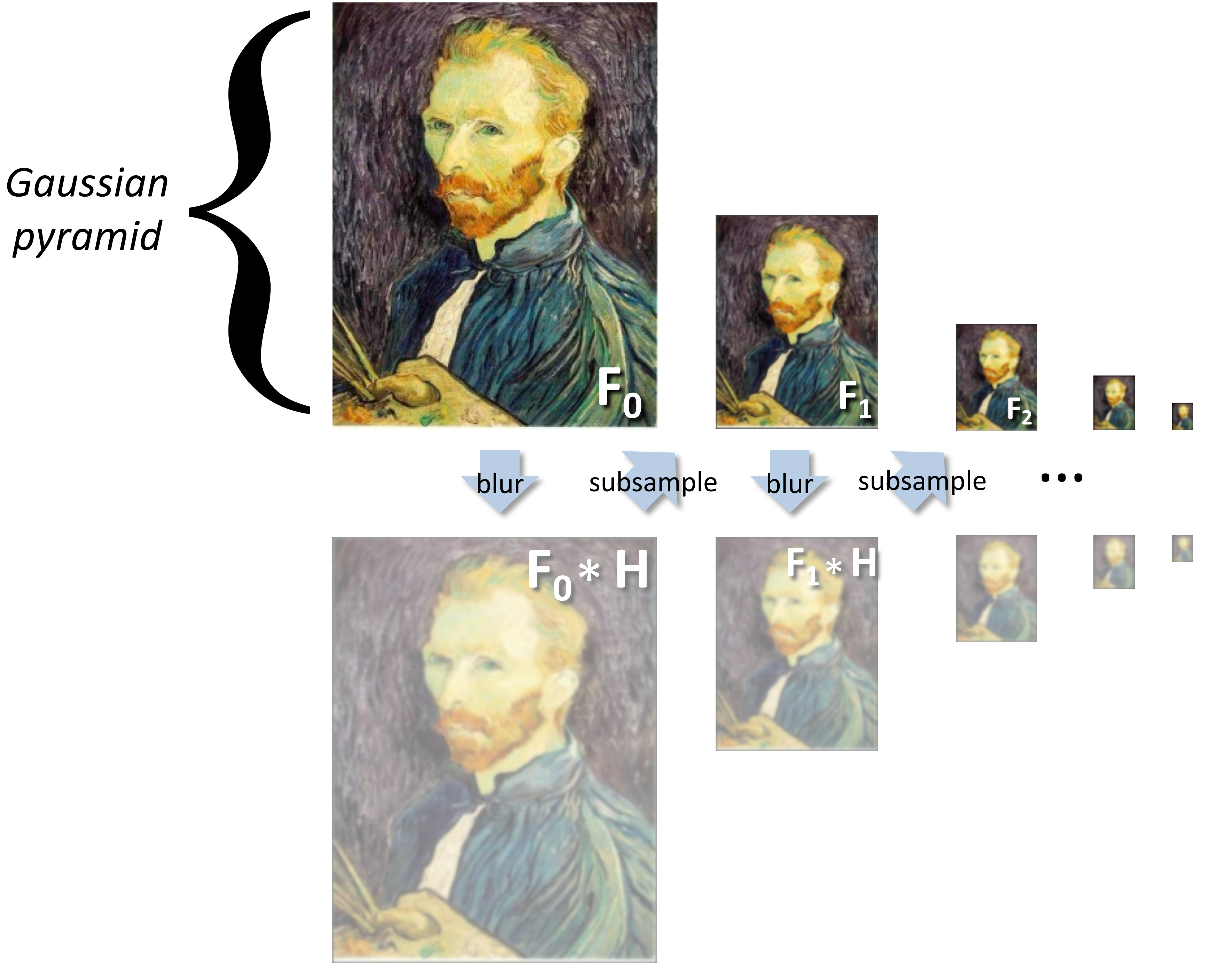
1/8 (4x zoom)

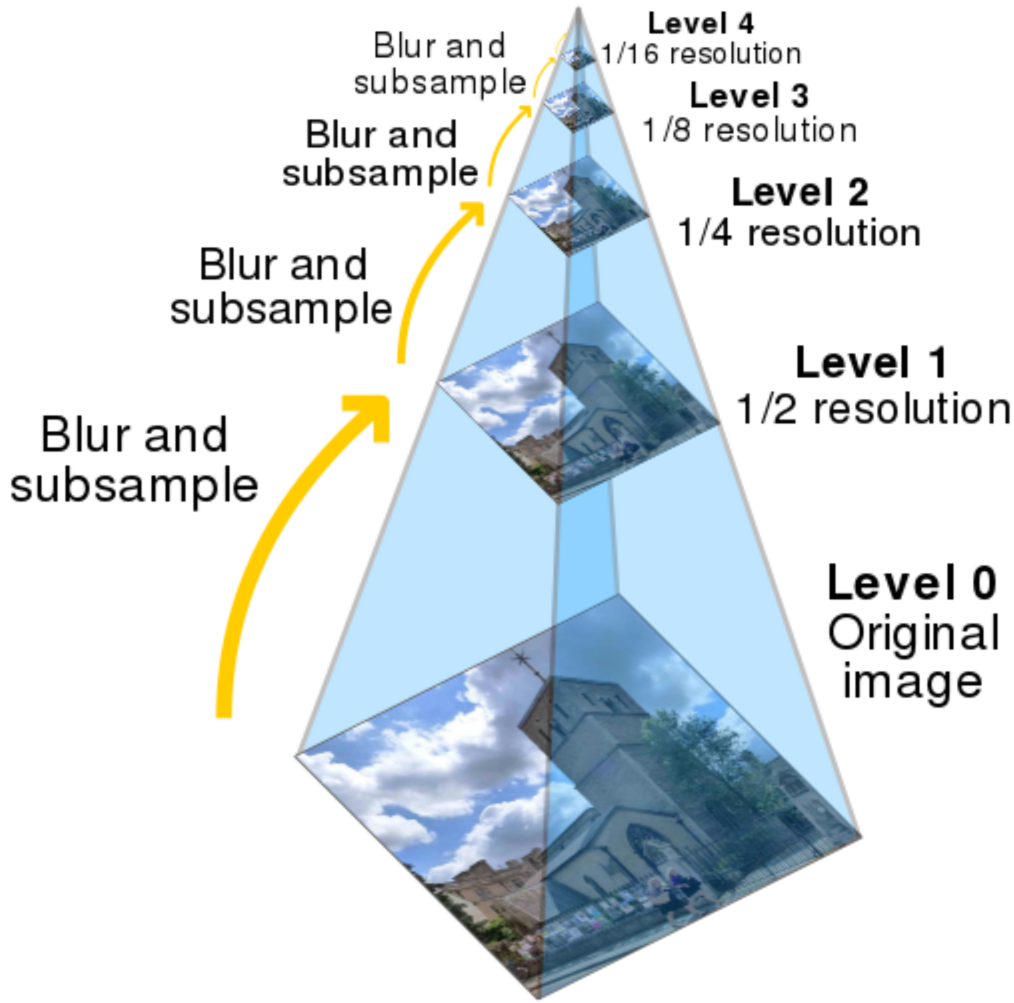
Source: S. Seitz

Gaussian pre-filtering

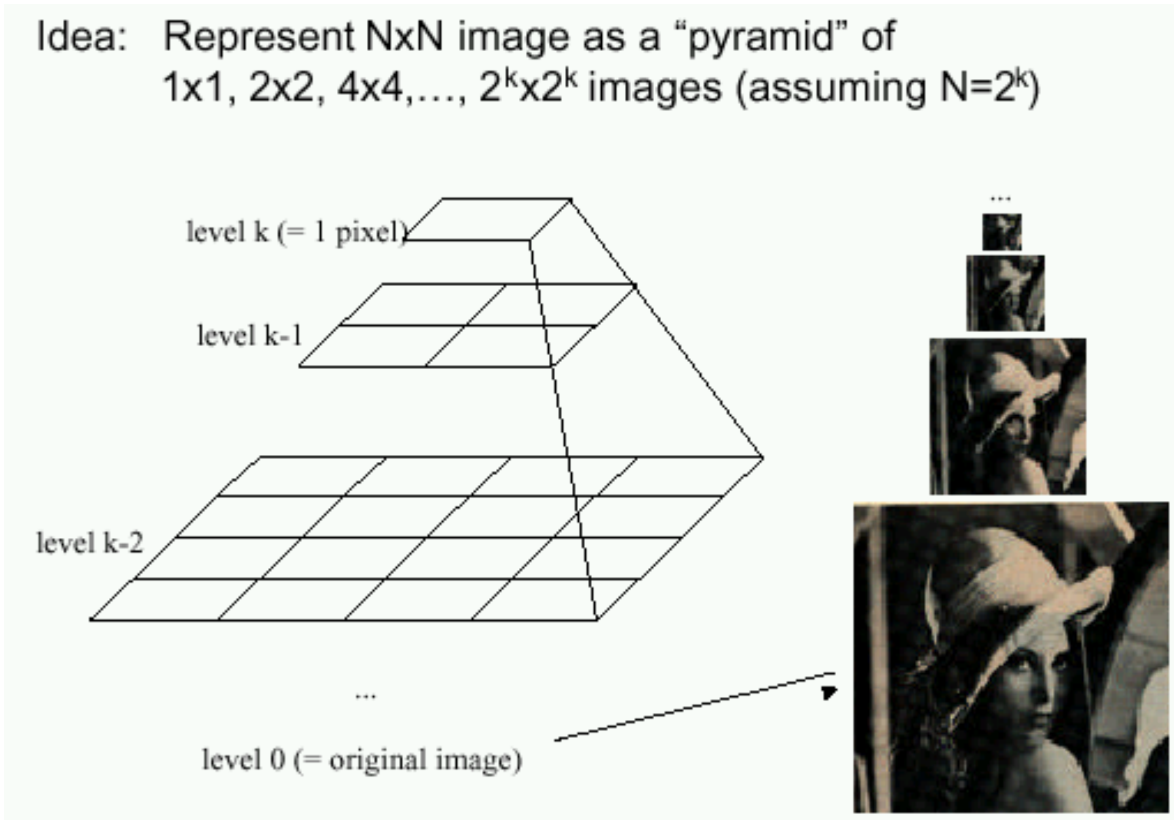
- Solution: filter the image, *then* subsample




Gaussian pyramid
F0
F1
F2
blur
subsample
blur
subsample
...
F0 * H
F1 * H


Level 4
1/16 resolution
Level 3
1/8 resolution
Level 2
1/4 resolution
Level 1
1/2 resolution
Level 0
Original image
Blur and subsample
Blur and subsample
Blur and subsample
Blur and subsample

Gaussian pyramids [Burt and Adelson, 1983]

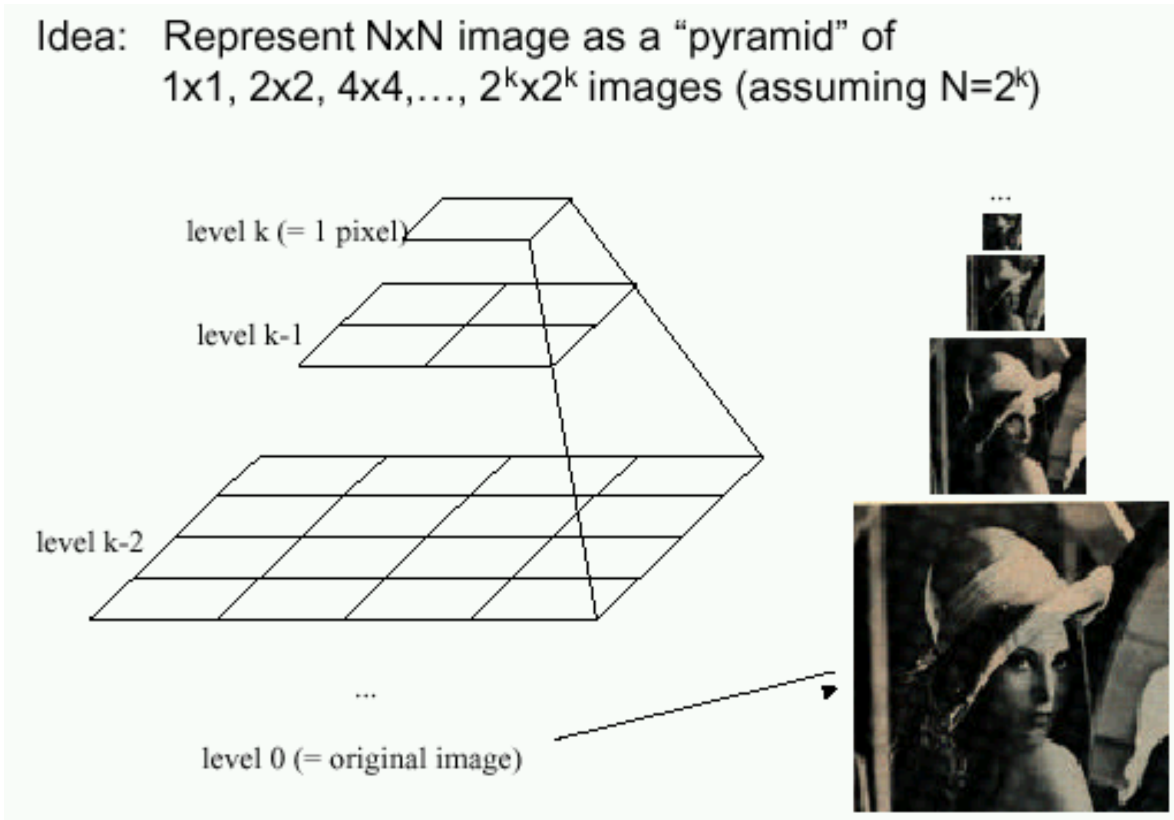


- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz

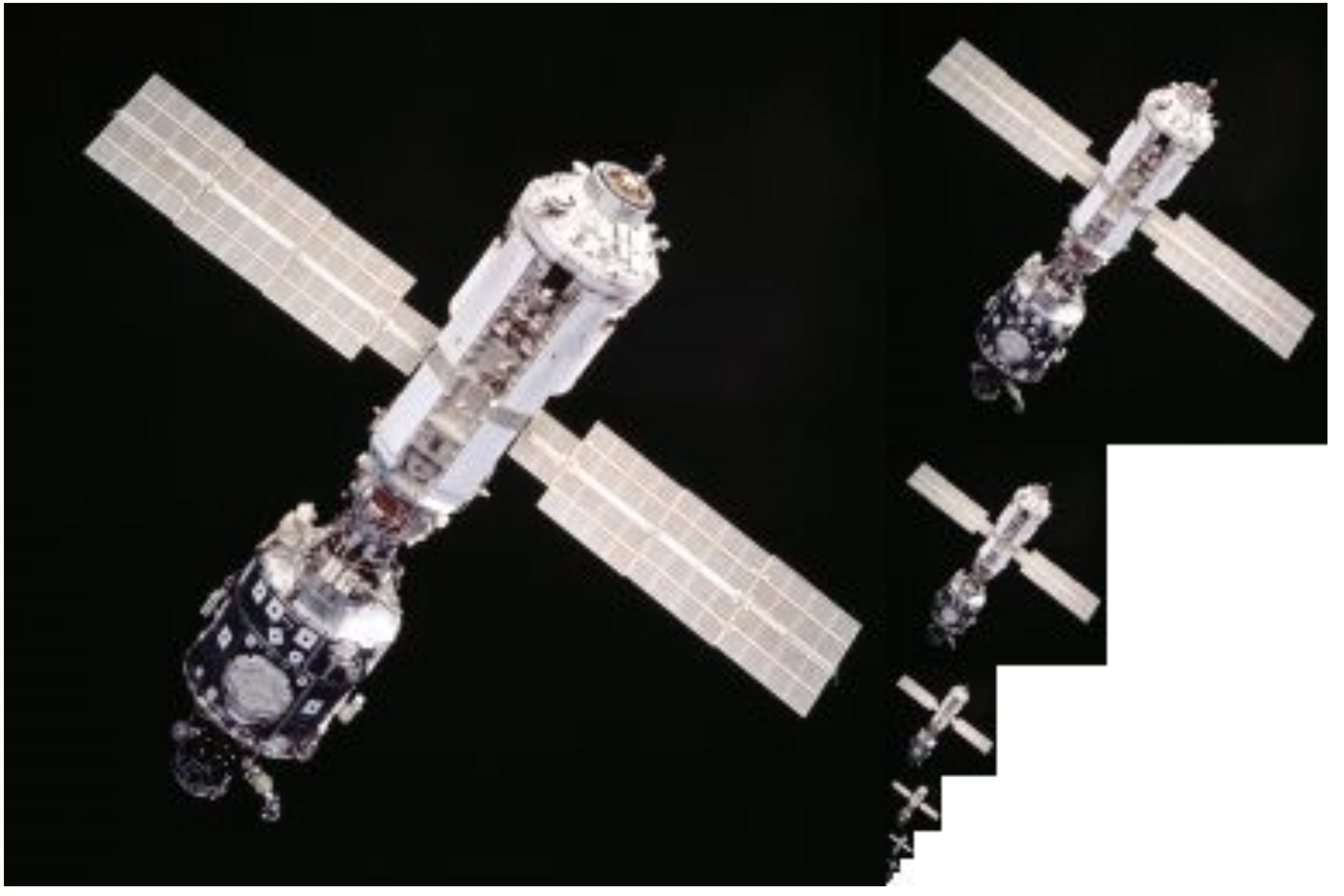
Gaussian pyramids [Burt and Adelson, 1983]



- How much space does a Gaussian pyramid take compared to the original image?

Source: S. Seitz

Gaussian Pyramid



https://en.wikipedia.org/wiki/1/4_%2B_1/16_%2B_1/64_%2B_1/256_%2B_%E2%8B%AF

What are Gaussian Pyramids useful for?

- Operating at multiple **scales**

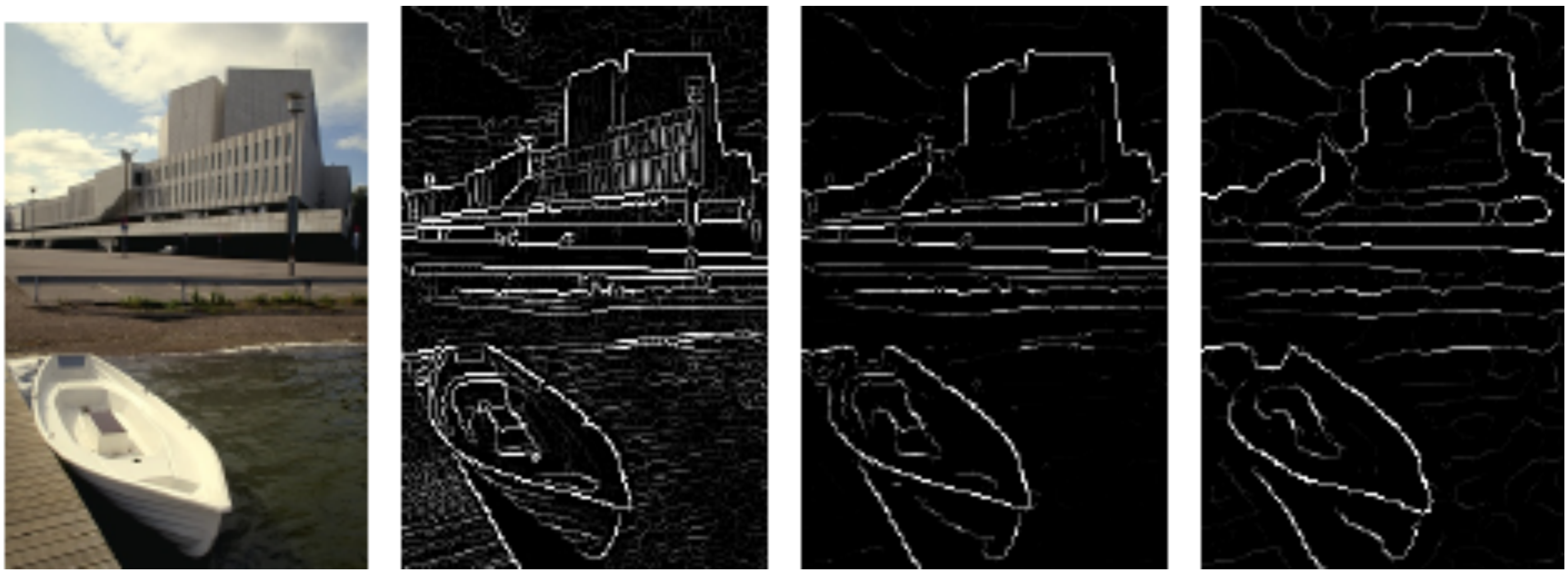
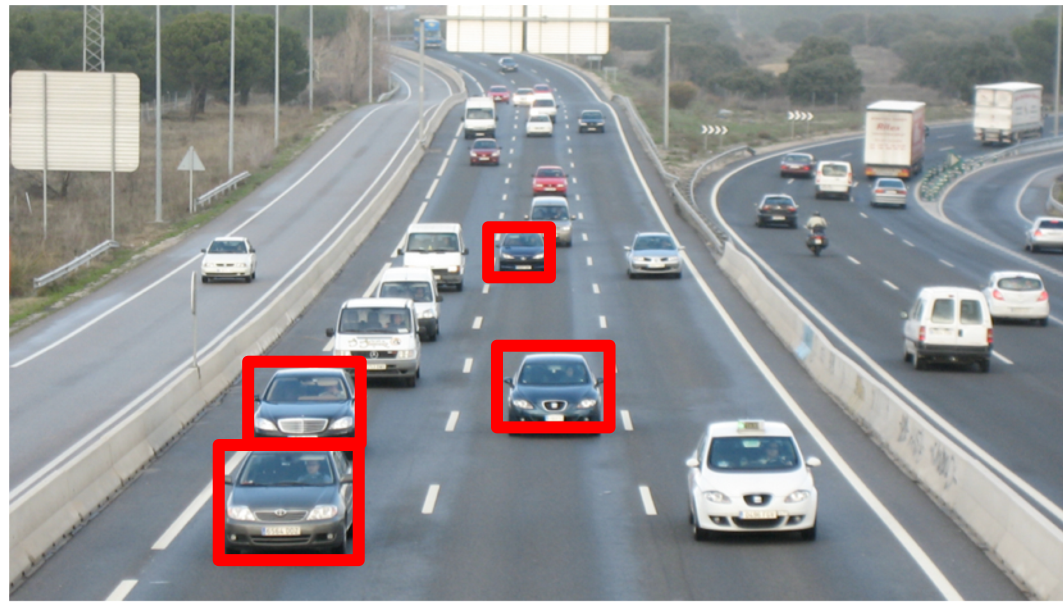


Image source: Baris Sumengen

Operating at multiple **scales**



Operating at multiple **scales**