Lecture 3

Image Sampling, Pyramids, and Edge Detection

Motivation

This image is too big to fit on the screen.

How can we shrink it to half its size?

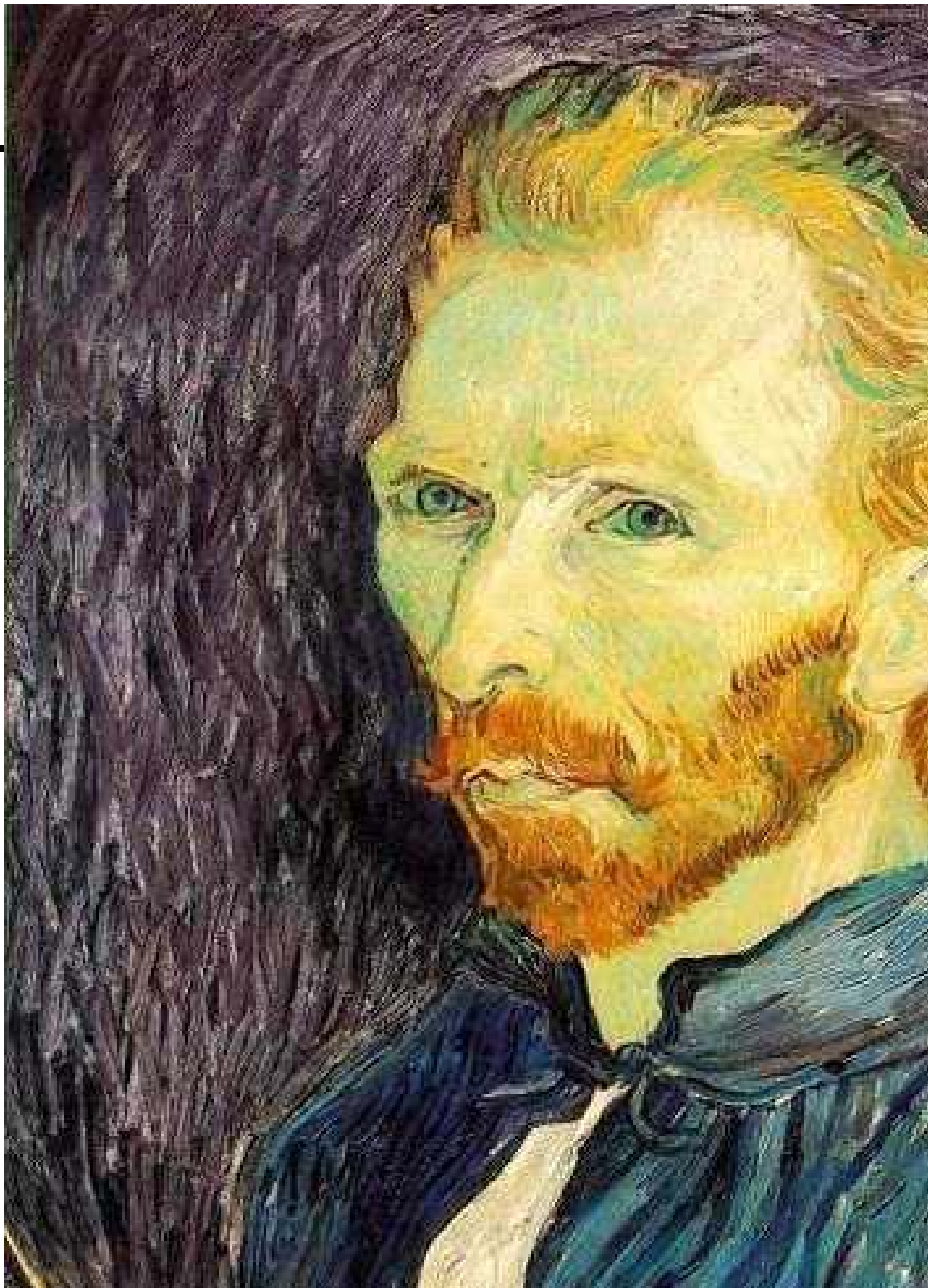
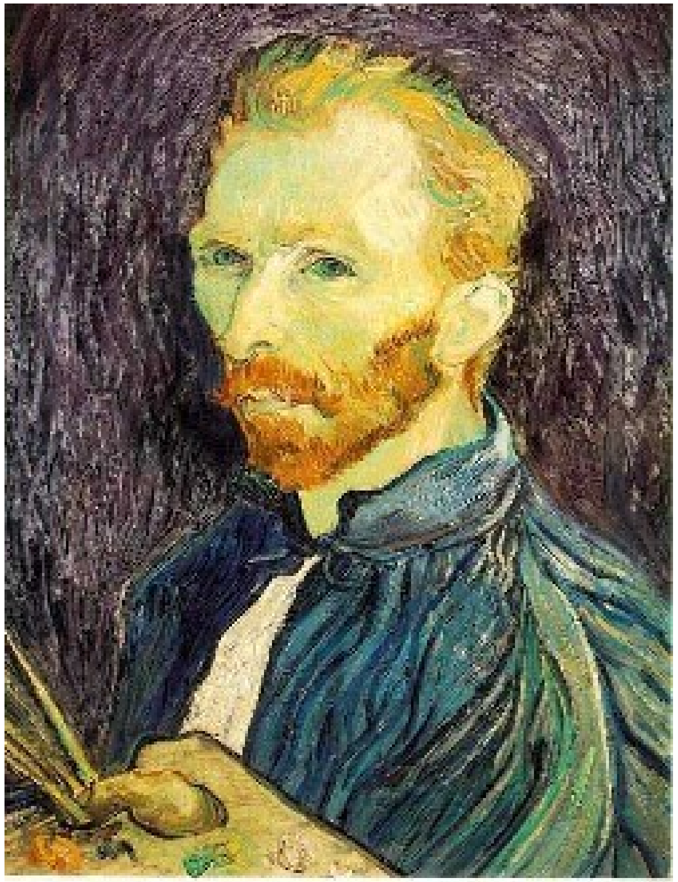
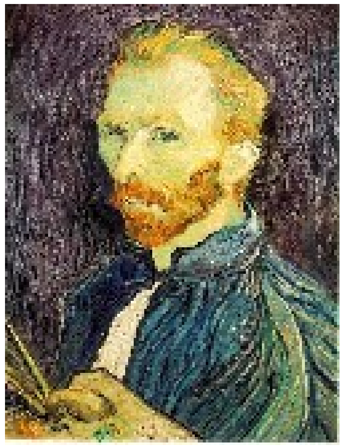


Image sub-sampling

Throw away every other row and column to create a smaller image
- called *image sub-sampling*



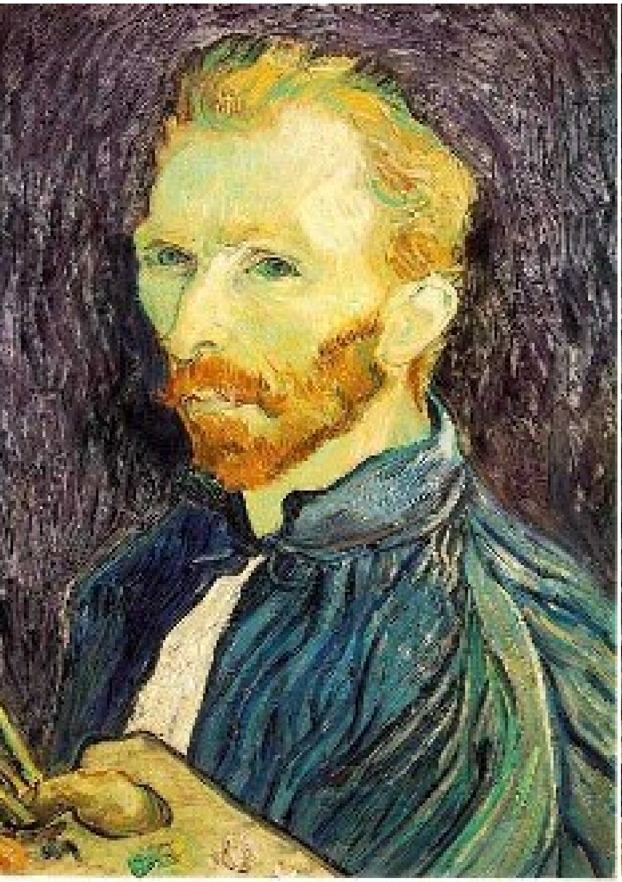
1/2

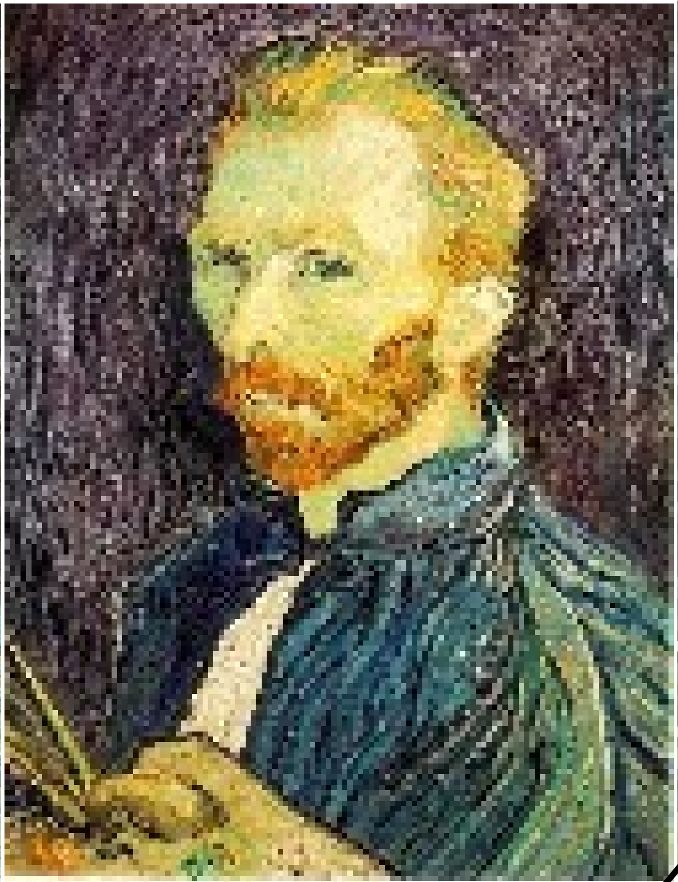


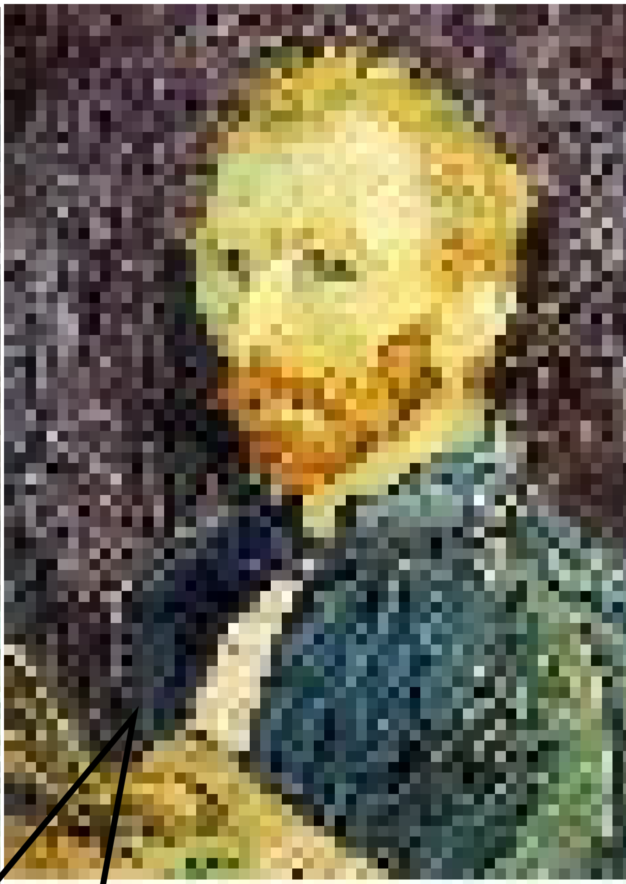
1/4

1/8

Image sub-sampling


1/2


1/4 (2x zoom)

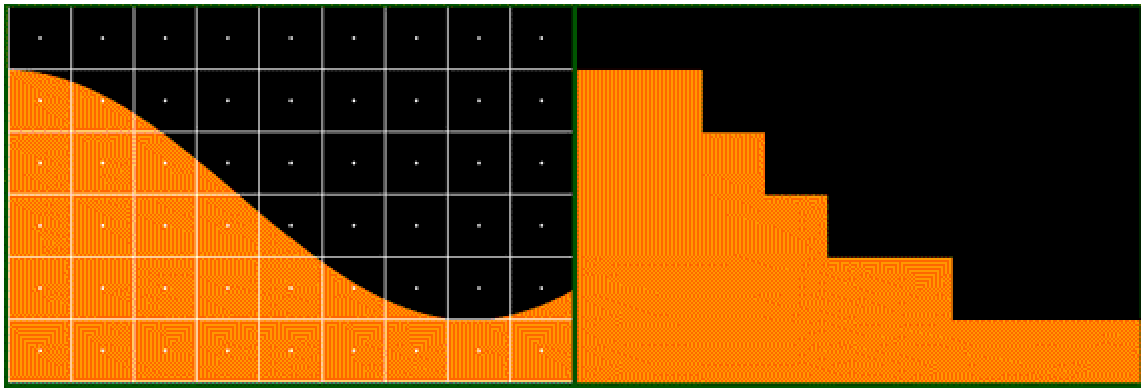

1/8 (4x zoom)

Waarom zo jagged kijk ik?*

*Dutch for “Why do I look so jagged?”

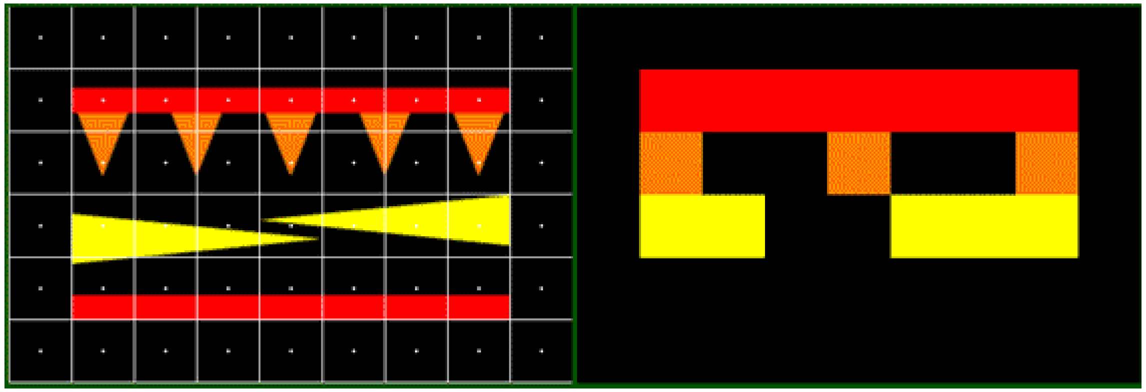
Artifacts arising from bad sampling

Jagged profiles

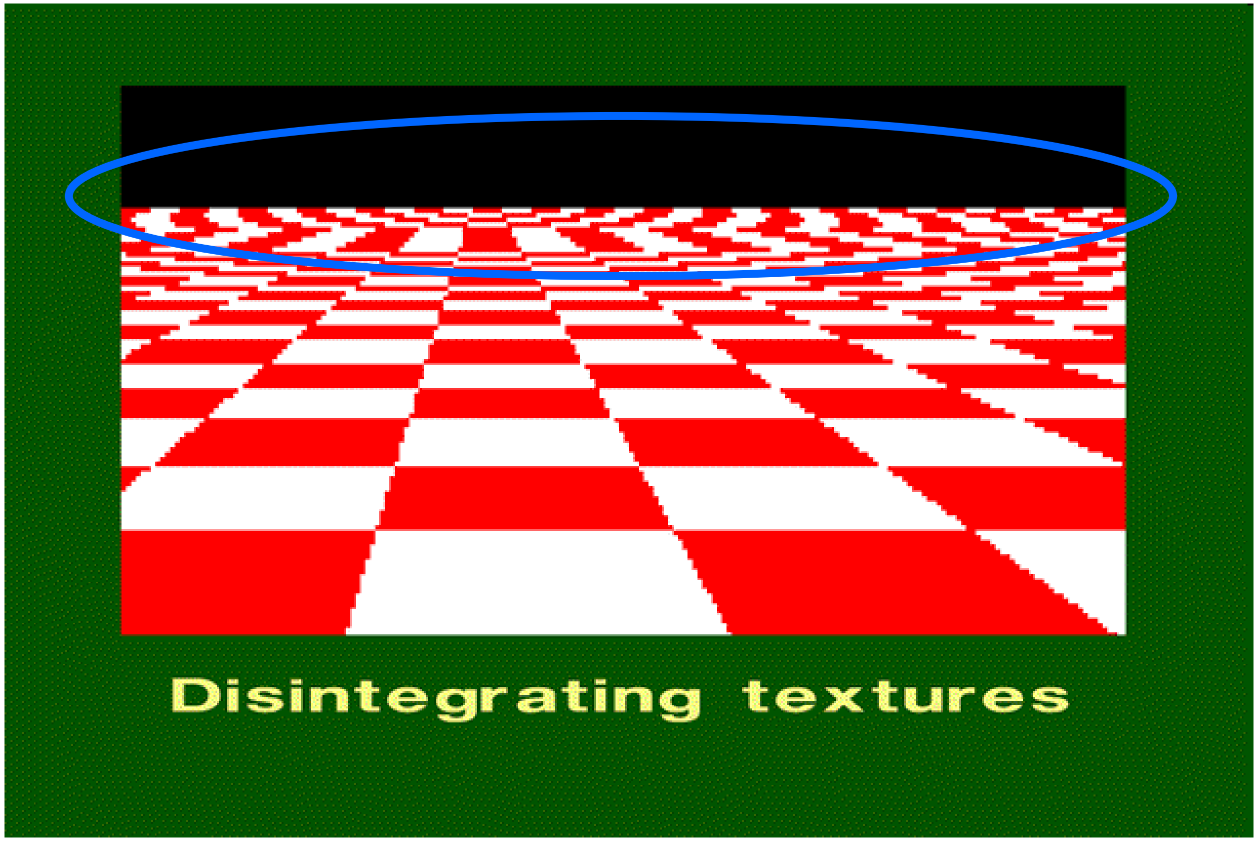


· represents a sampled location

Missing details

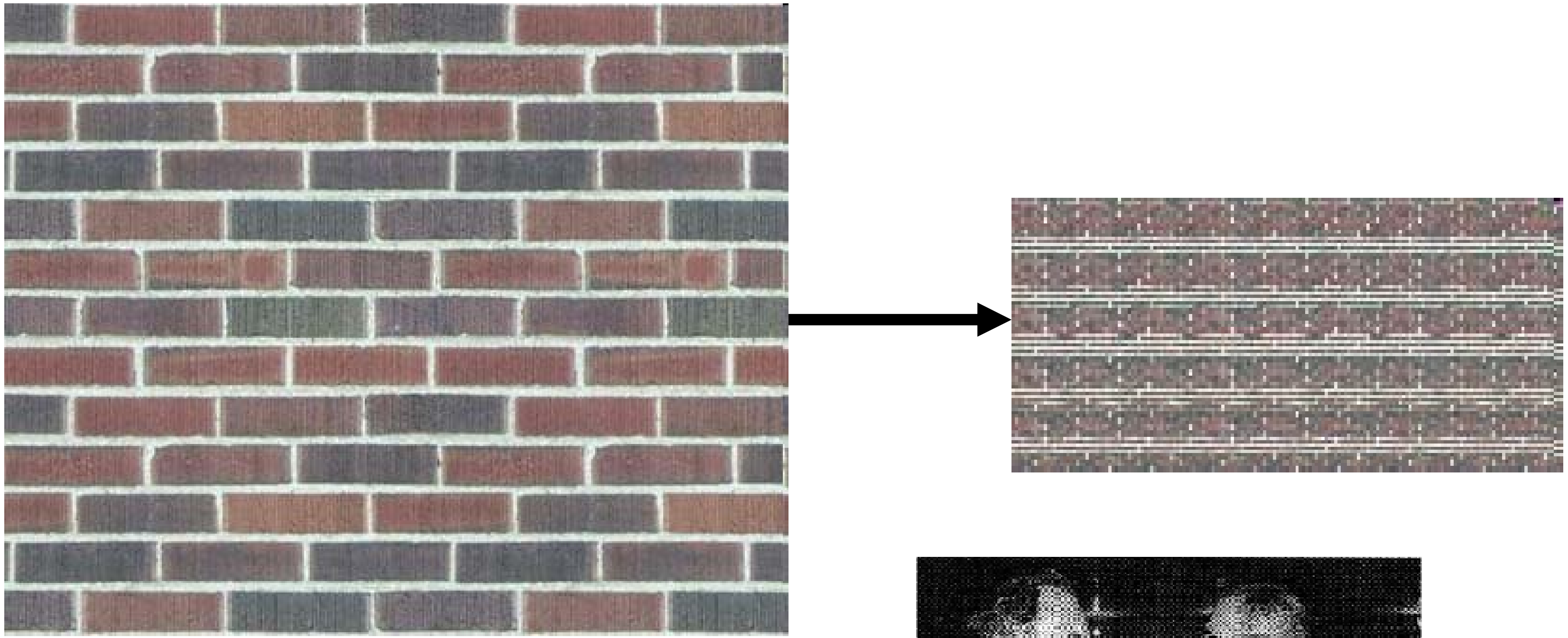


Artifacts arising from bad sampling

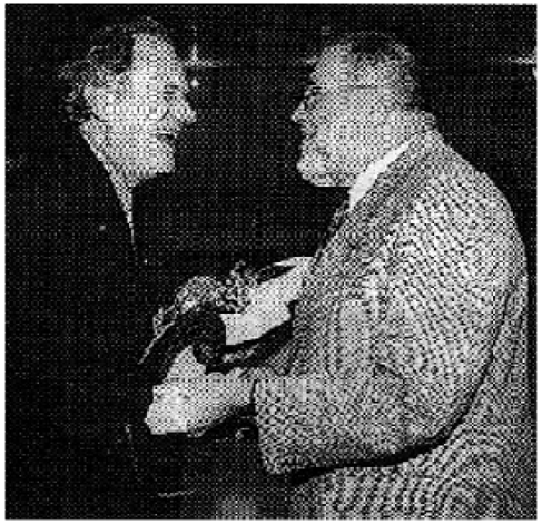


Checkers should get smaller as distance increases

Moire patterns

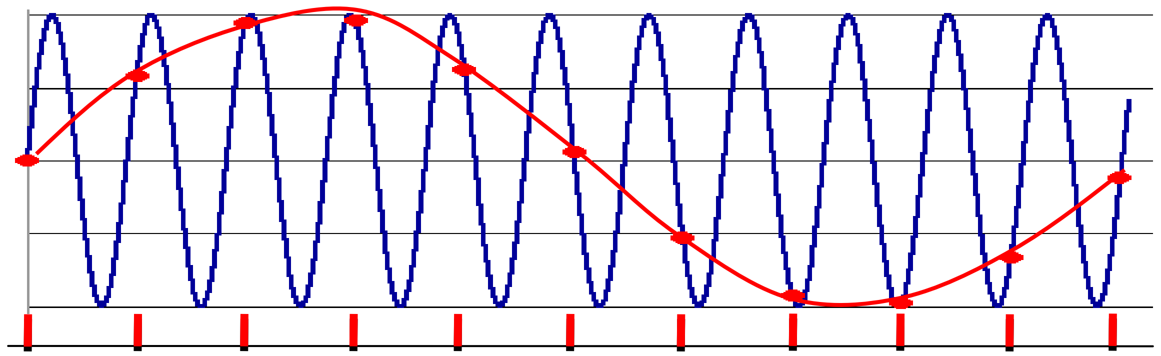


http://www.wfu.edu/~matthews/misc/DigPhotog/alias/



http://www.daube.ch/docu/glossary/moiree.html

Aliasing and the Nyquist rate



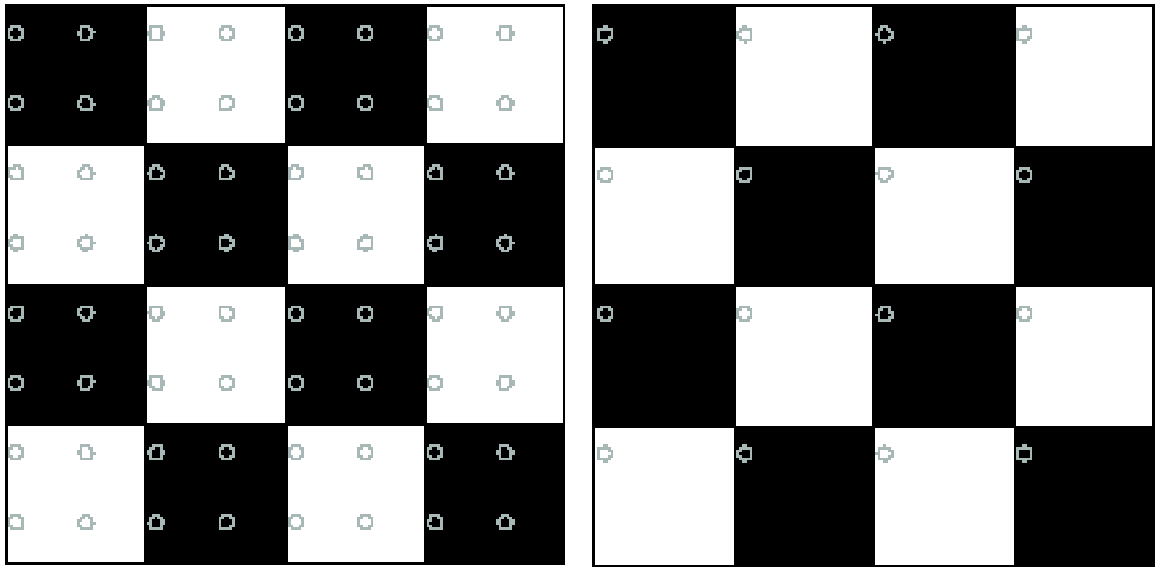
Aliasing can arise when you sample a continuous signal or image

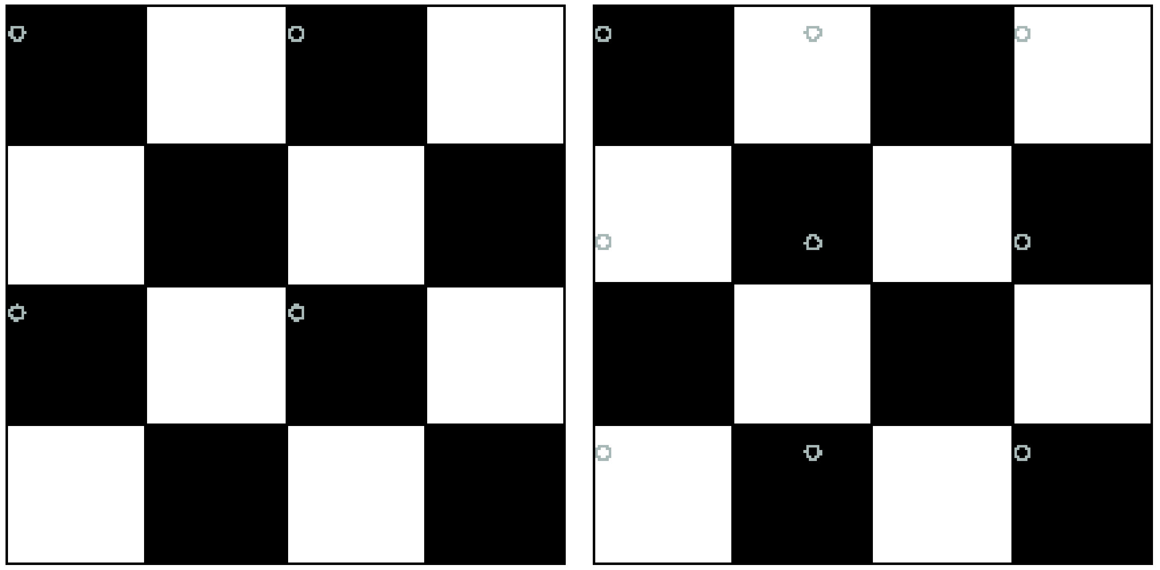
- occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*
- formally, the image contains structure at different scales
  - called “frequencies” in the Fourier domain
- the sampling rate must be high enough to capture the highest frequency in the image

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**

2D example


Good sampling


Bad sampling

Avoiding aliasing in sub-sampling

To avoid aliasing, sampling rate ≥ 2 * max frequency in the image (this is called the **Nyquist rate)**

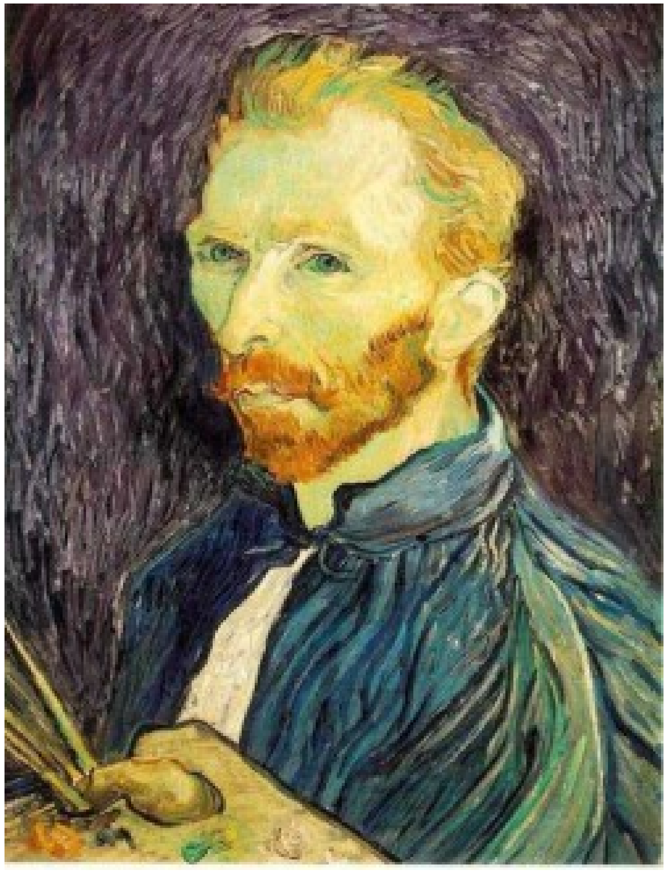
Therefore, can avoid aliasing in sub-sampling by reducing the max frequency in the image.

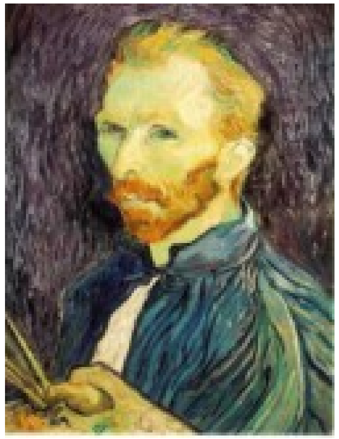
How?

Sub-sampling with Gaussian pre-filtering

Blur the image (low pass filter) the image, *then* subsample

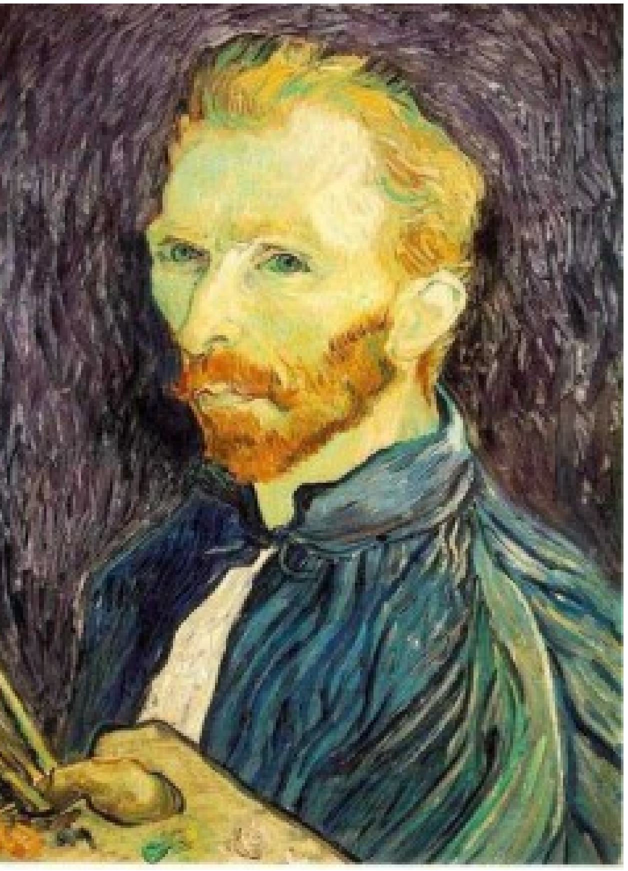
- Blur using Gaussian filter
- Filter size should double for each ½ size reduction (Nyquist)

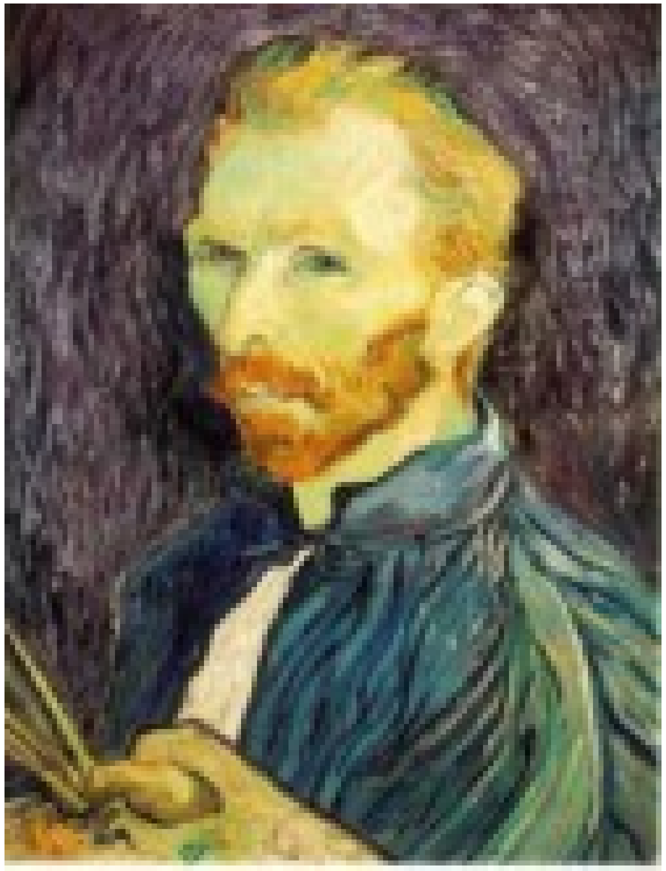

Gaussian 1/2

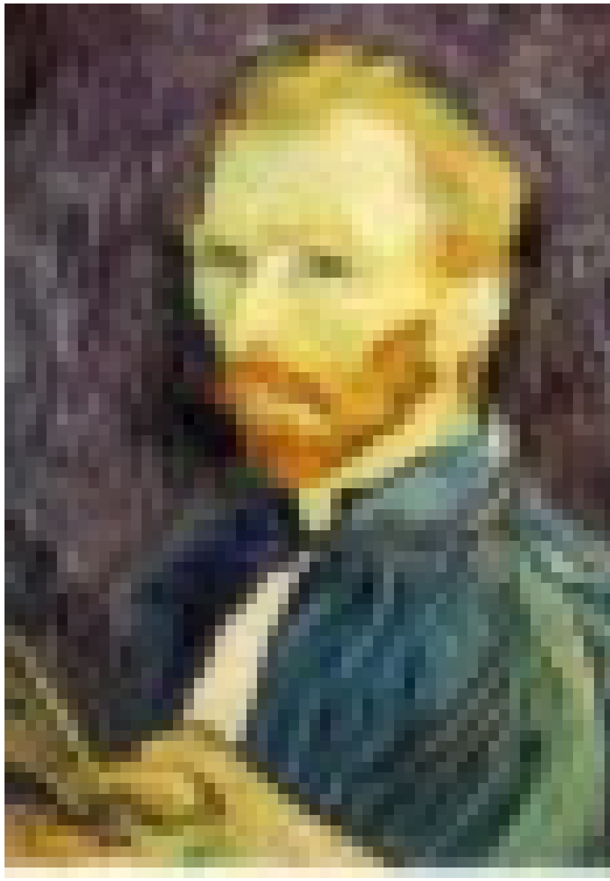

G 1/4

G 1/8

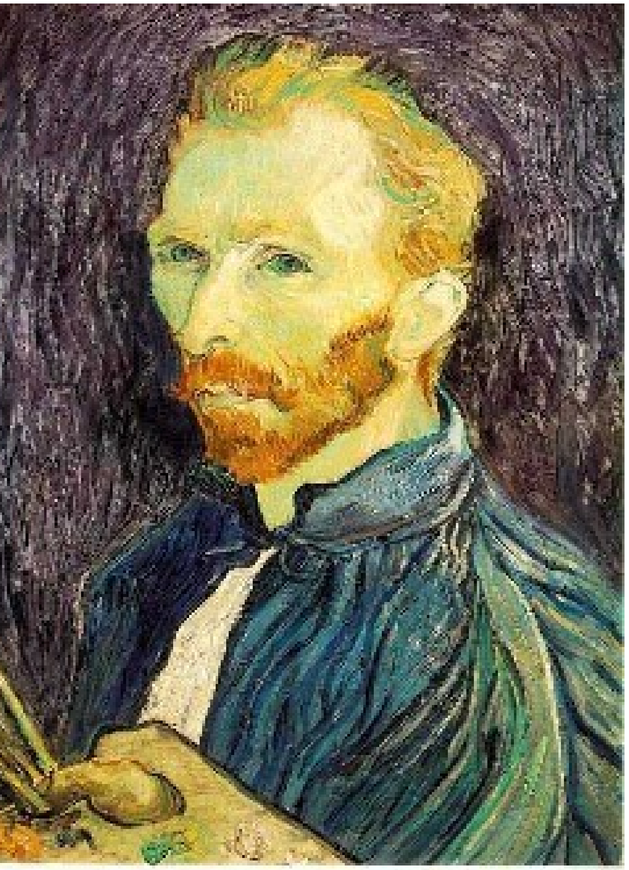
Sub-sampling with Gaussian pre-filtering

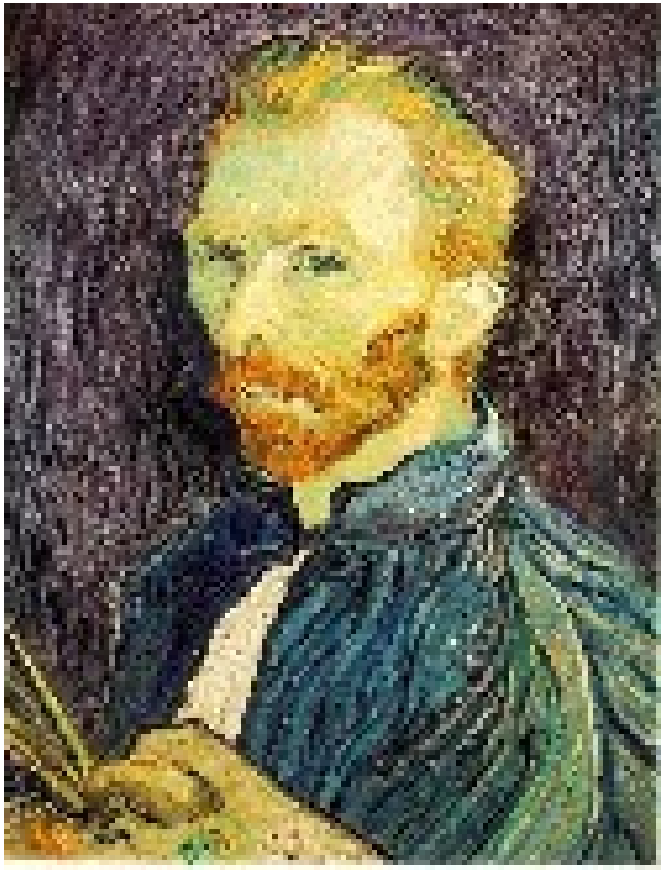

Gaussian 1/2

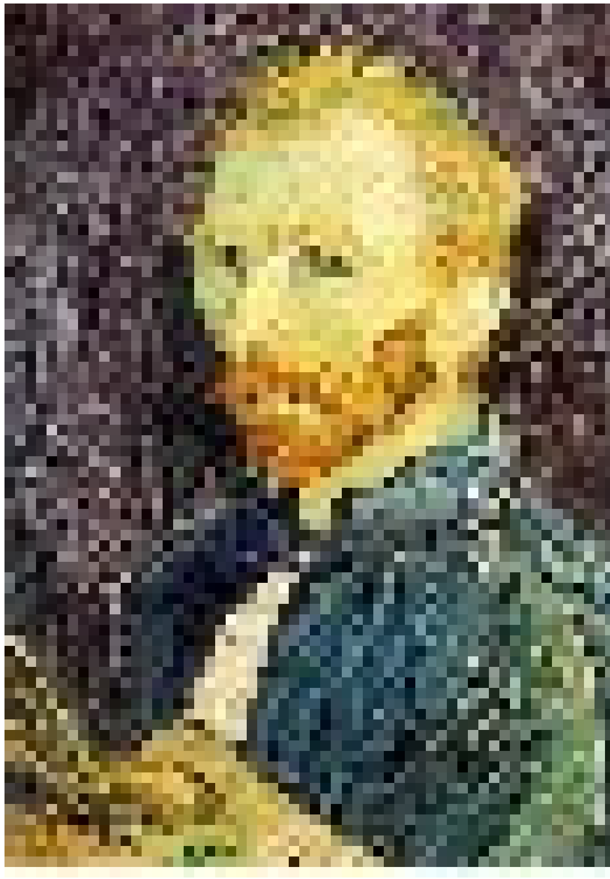

G 1/4 (2xzoom)


G 1/8 (4xzoom)

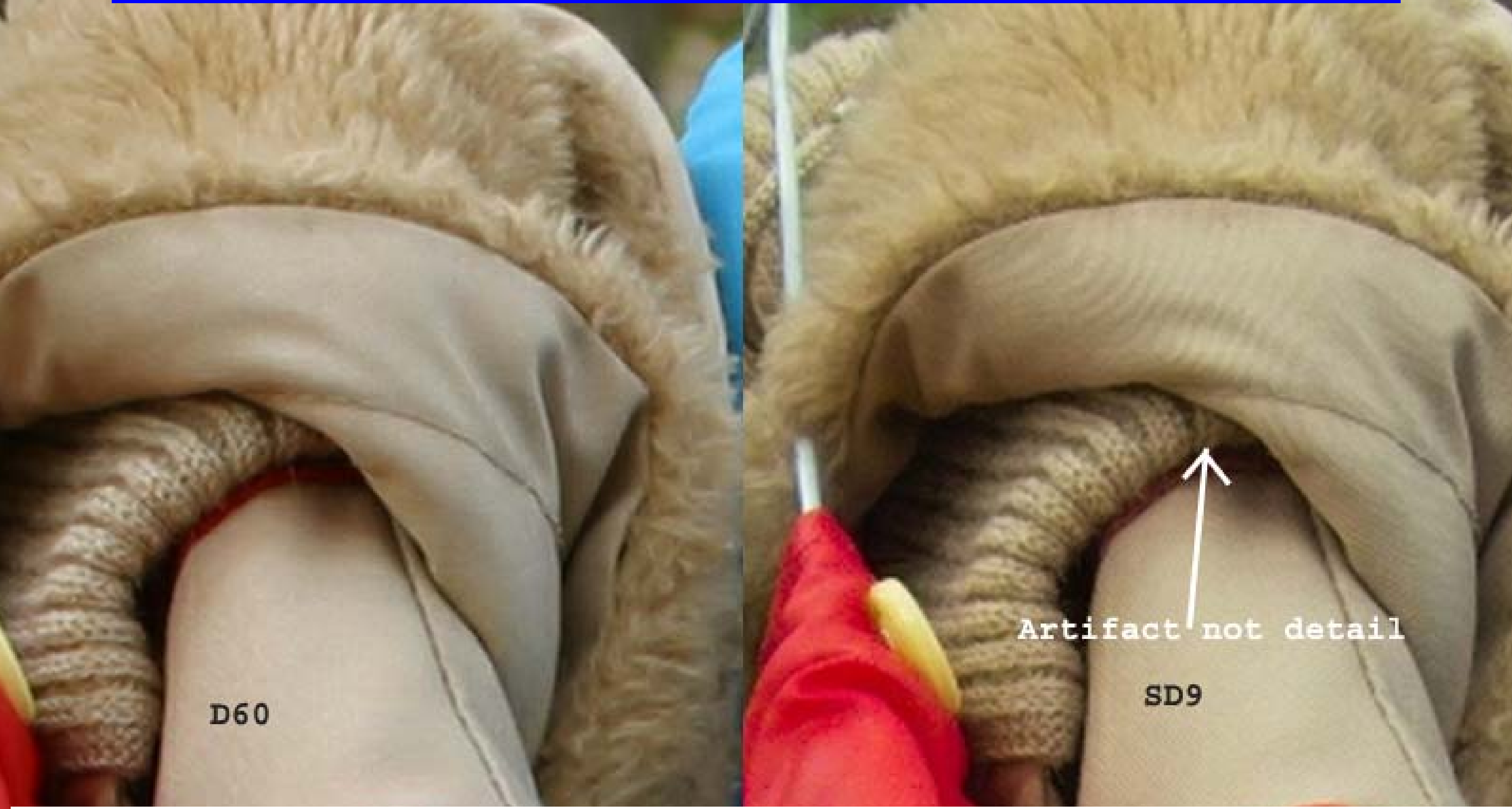
Compare with...


1/2


1/4 (2x zoom)

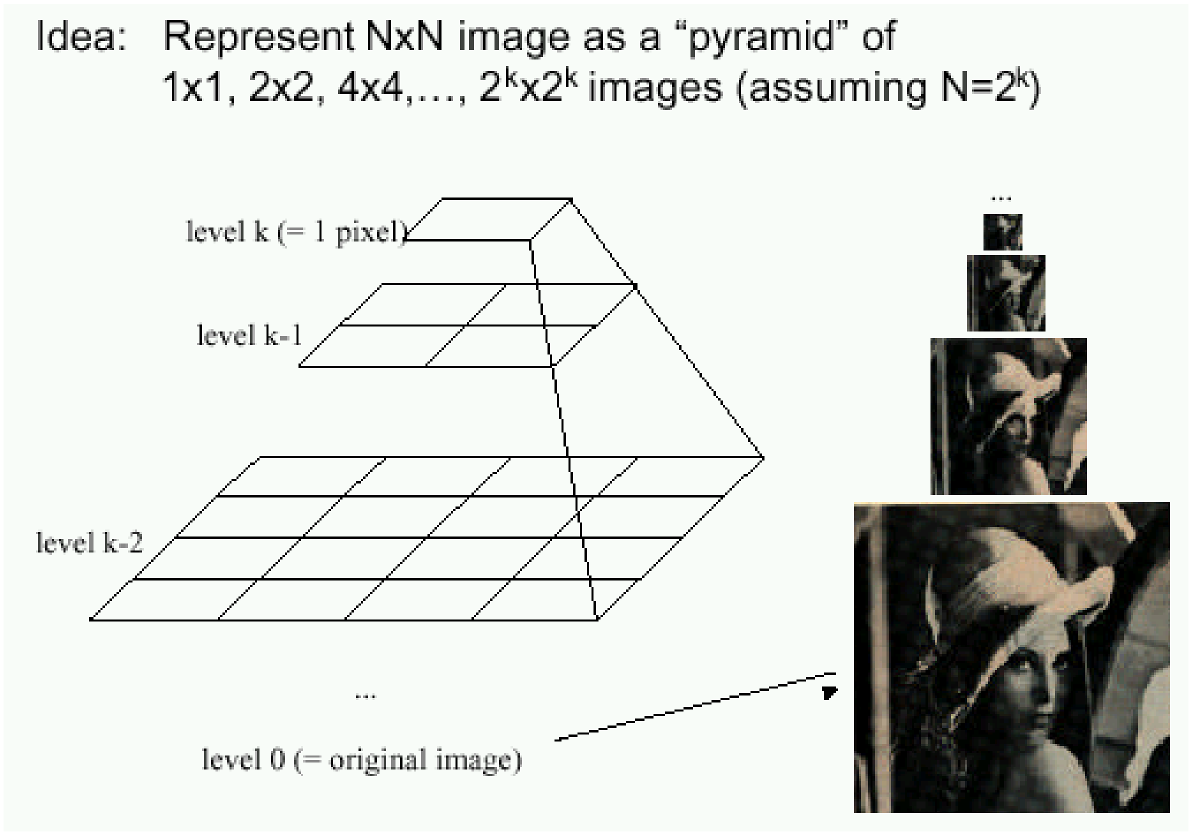

1/8 (4x zoom)

Aliasing and Moire patterns in real images



Images by Dave Etchells of Imaging Resource using the Canon D60 (with an antialias filter) and the Sigma SD-9 (which has no antialias filter). The bands below the fur in the image at right are the kinds of artifacts that appear in images when no antialias filter is used. Sigma chose to eliminate the filter to get more sharpness, but the resulting apparent detail may or may not reflect features in the image.

Some times we want many resolutions


Idea: Represent NxN image as a "pyramid" of 1x1, 2x2, 4x4,..., 2^kx2^k images (assuming N=2^k)
level k (= 1 pixel)
level k-1
level k-2
...
level 0 (= original image)
...

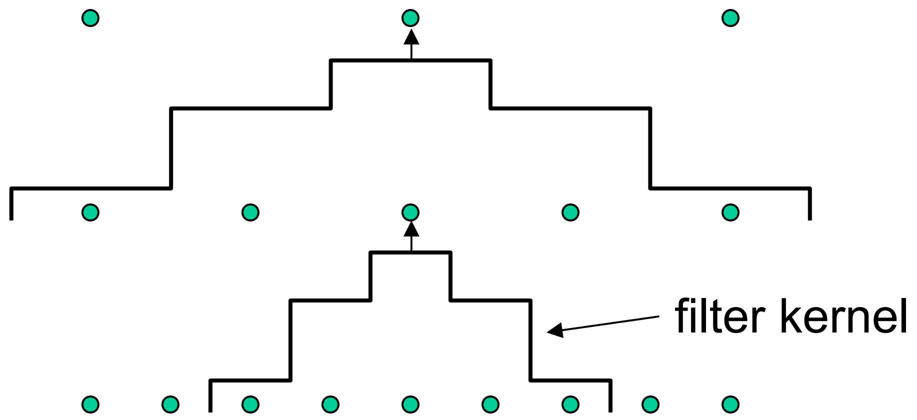
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

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

Gaussian Pyramids have all sorts of applications in computer vision

- Texture synthesis, compression, feature detection, object recognition

Gaussian pyramid construction



Repeat

- Filter w/ Gaussian
- Subsample

Until minimum resolution reached

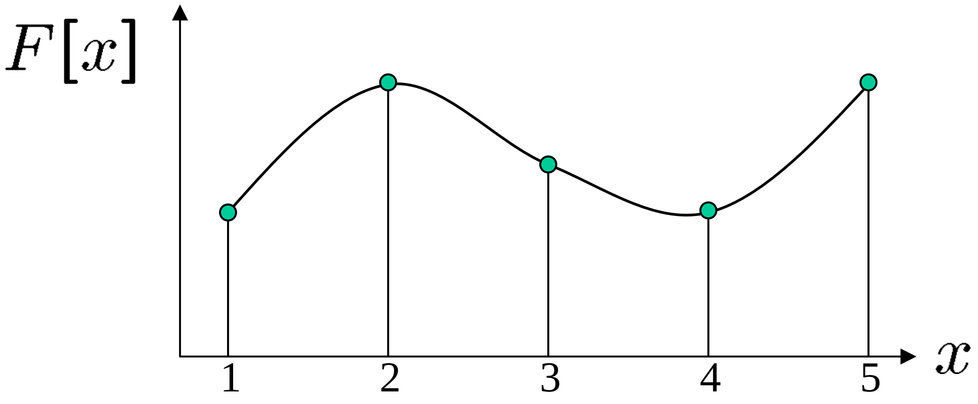
- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?



d = 1 in this example

Recall how a digital image is formed

$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Image resampling and interpolation

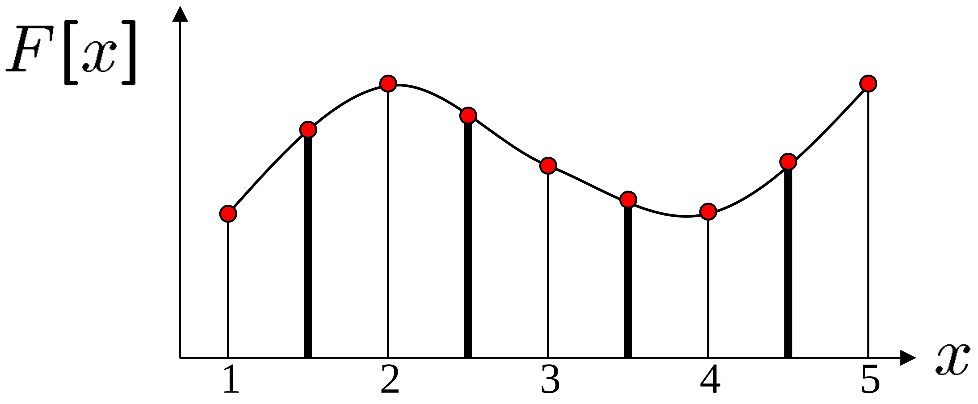
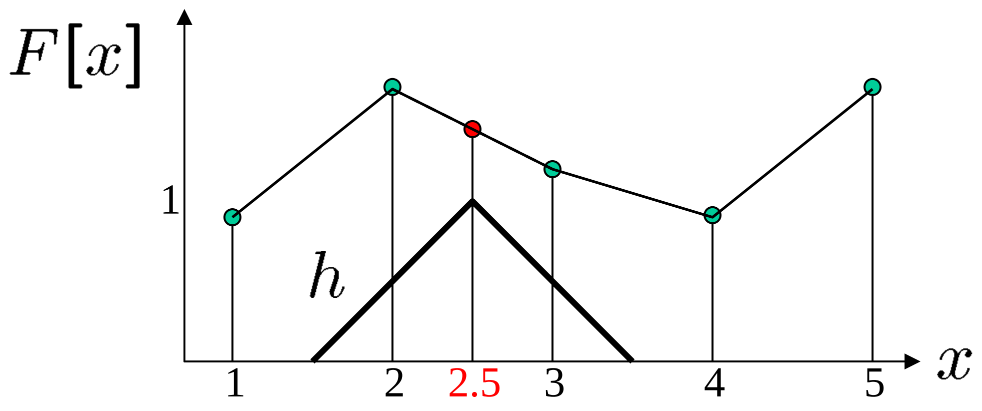


Image resampling and interpolation

So what to do if we don't know f

- Answer: guess an approximation $\tilde{f}$
- Can be done in a principled way: filtering



linear interpolation

d = 1 in this example

Image reconstruction

- Define a continuous function based on F

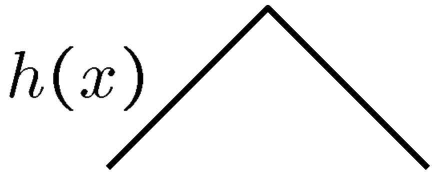
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

- Reconstruct by cross-correlation with filter h:

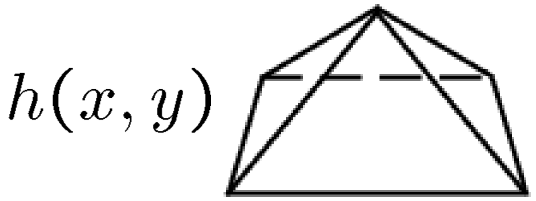
$$\tilde{f} = h \otimes f_F$$

Interpolation filters

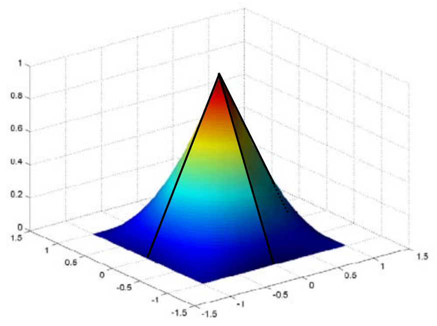
What does the 2D version of this hat function look like?



performs
linear interpolation



(tent function) performs
bilinear interpolation

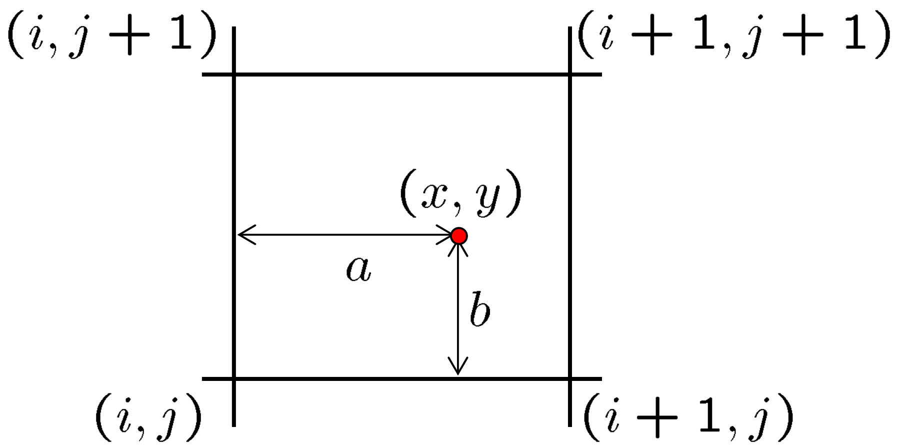


Better filters give better resampled images

- Bicubic is common choice
  - fit 3rd degree polynomial surface to pixels in neighborhood
  - can also be implemented by a convolution

Bilinear interpolation (exercise)

A simple method for resampling images



Fill in the blanks in terms of a and b

$$f(x,y) = \underline{\qquad\qquad} f[i,j]$$
$$\underline{\qquad\qquad} f[i+1,j]$$
$$\underline{\qquad\qquad} f[i+1,j+1]$$
$$\underline{\qquad\qquad} f[i,j+1]$$

Okay, enough about sampling

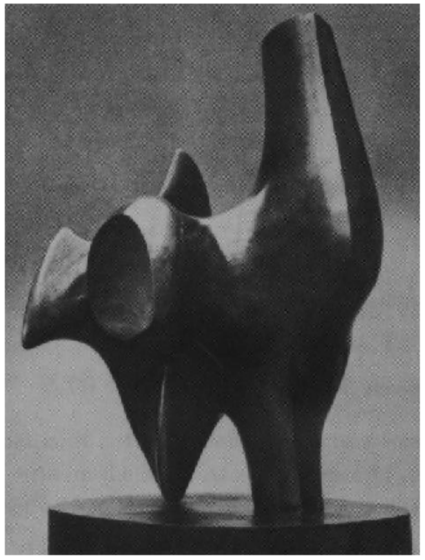
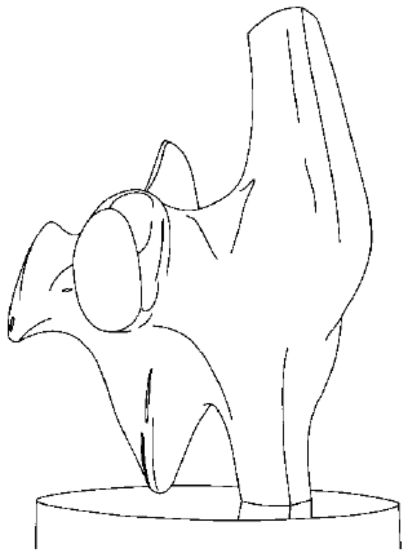
Suppose we want to extract useful features from images

One type of feature: Edges

Why extract edges?

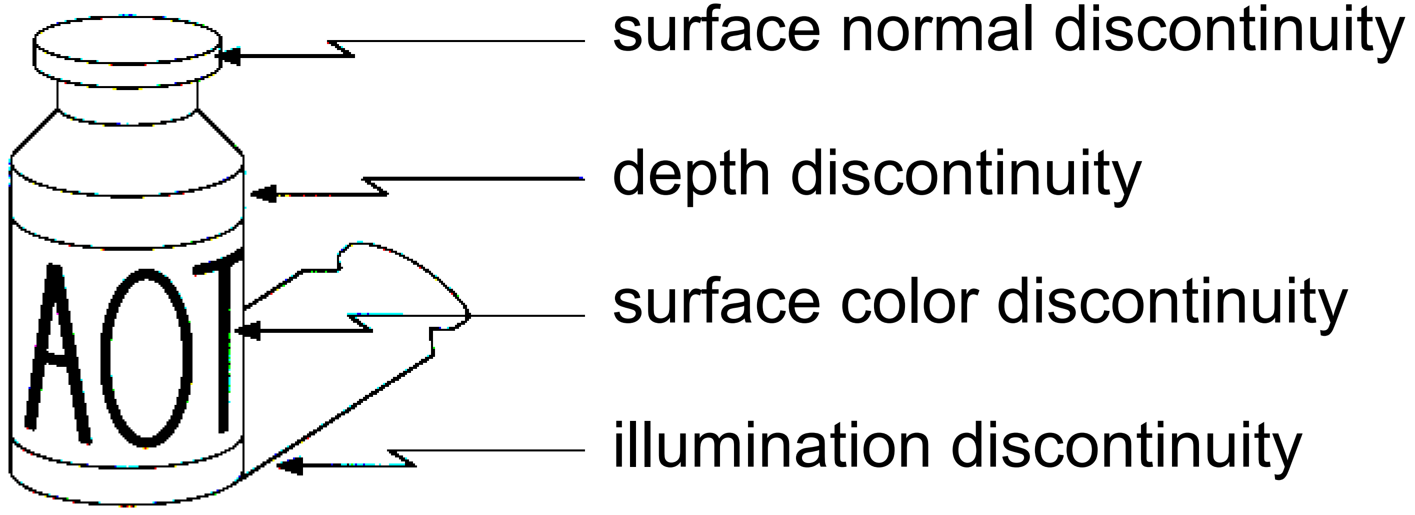
- Edges and lines are used in
  - object recognition
  - image matching (e.g., stereo, mosaics)
  - document analysis
  - horizon detection
  - line following robots
  - and many more apps

- More compact than pixels

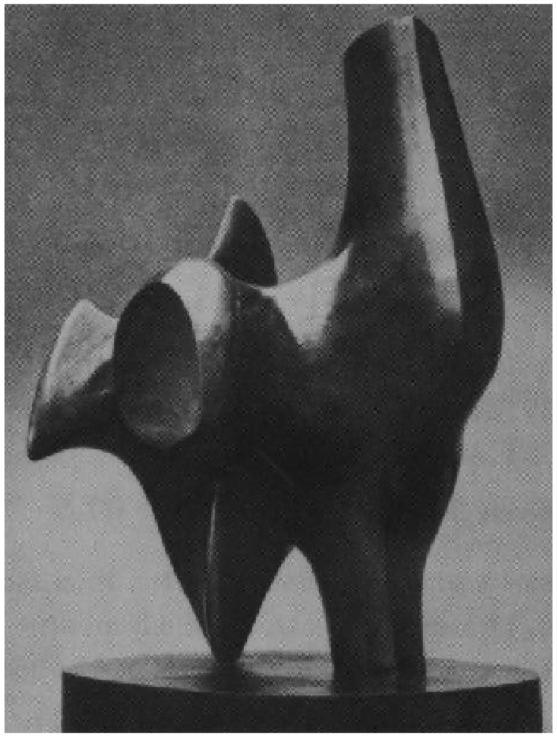
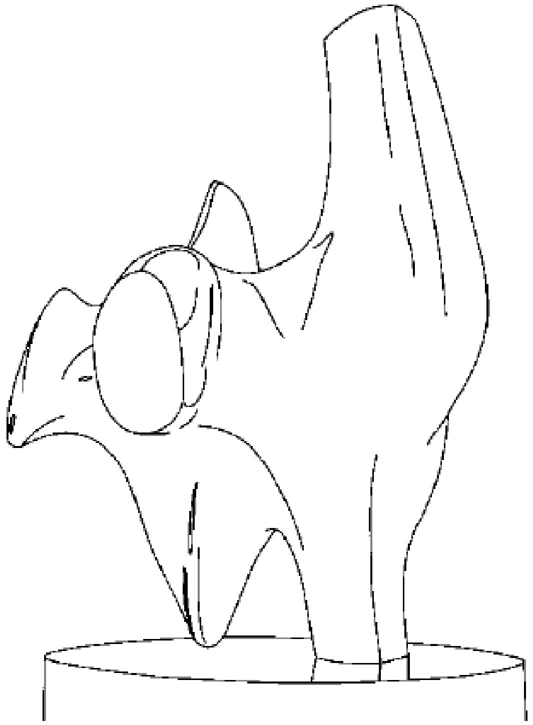



Where do edges come from?

Edges in images are caused by a variety of factors

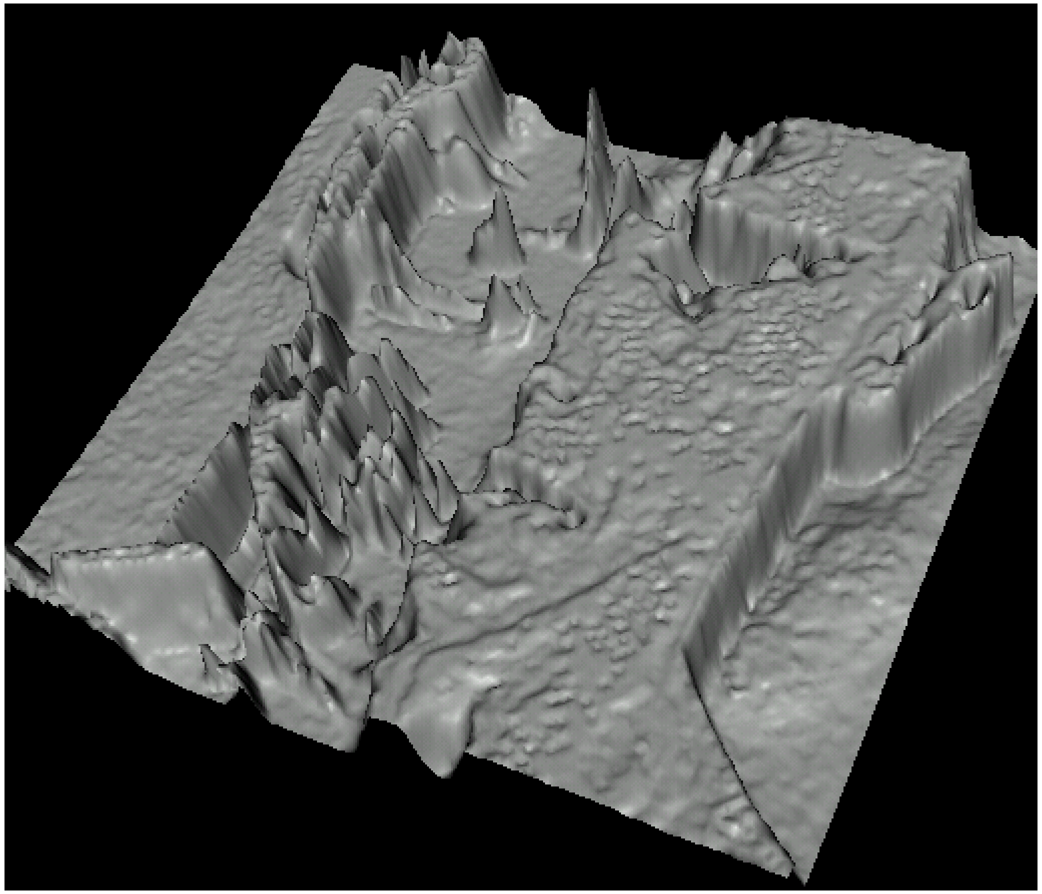

AOT
surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

Edge detection

How can you tell that a pixel is on an edge?

Images as functions…

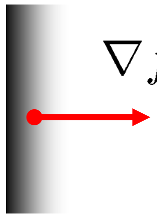


Edges look like steep cliffs

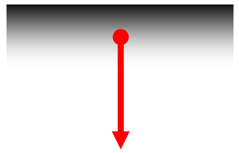
Image gradient

The gradient of an image:

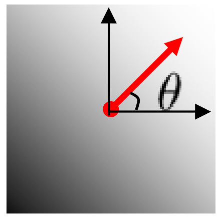
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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



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

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

The gradient direction is given by:

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

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

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 discrete gradient

How can we differentiate a *digital* image F[x,y]?

y

x

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

The discrete gradient

How can we differentiate a *digital* image F[x,y]?

- Answer: 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 cross-correlation?

	-1	+1

H

The Sobel operator

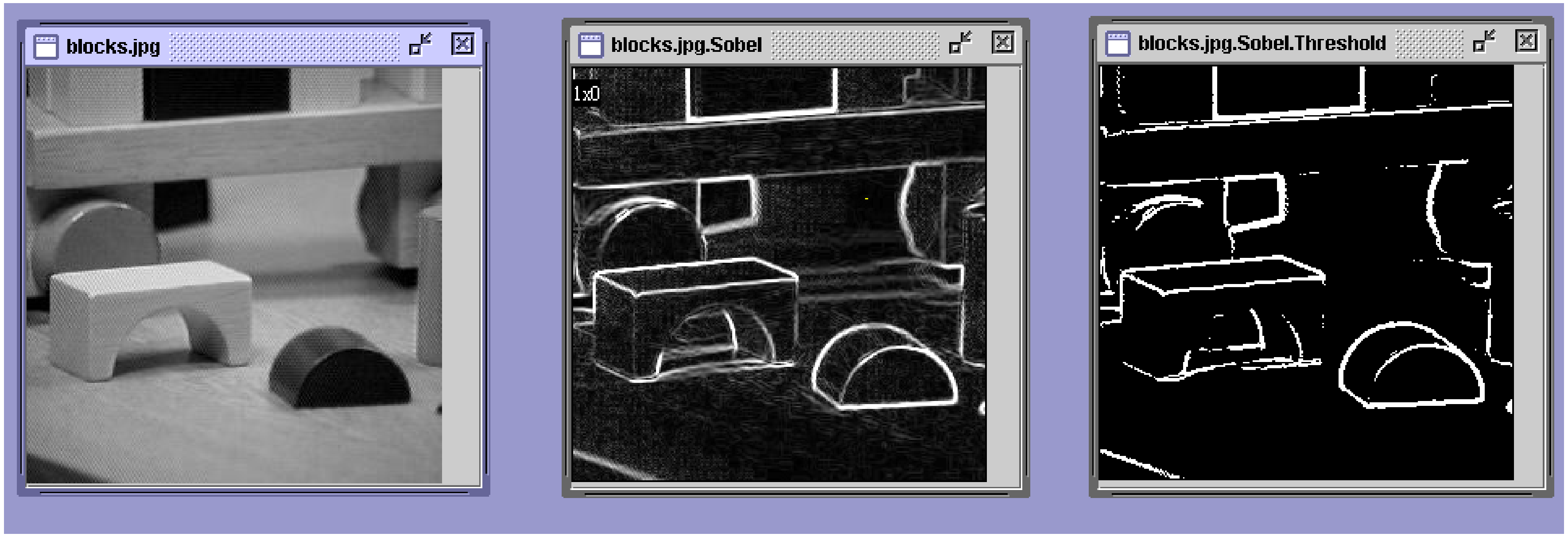
Better approximations of the derivatives exist

- The *Sobel* operators below are very commonly used

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

- 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 value, however

Edge detection using the Sobel operator



original image

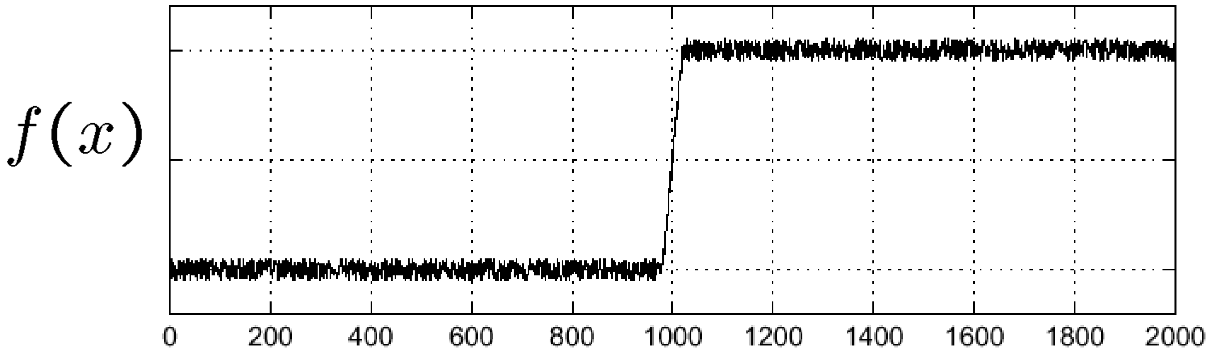
Sobel gradient magnitude

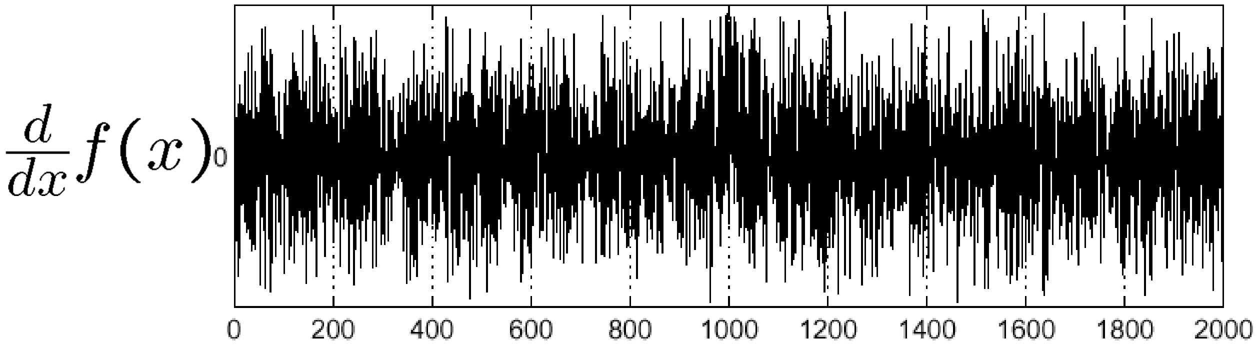
thresholded

Effects of noise

Consider a single row or column of the image

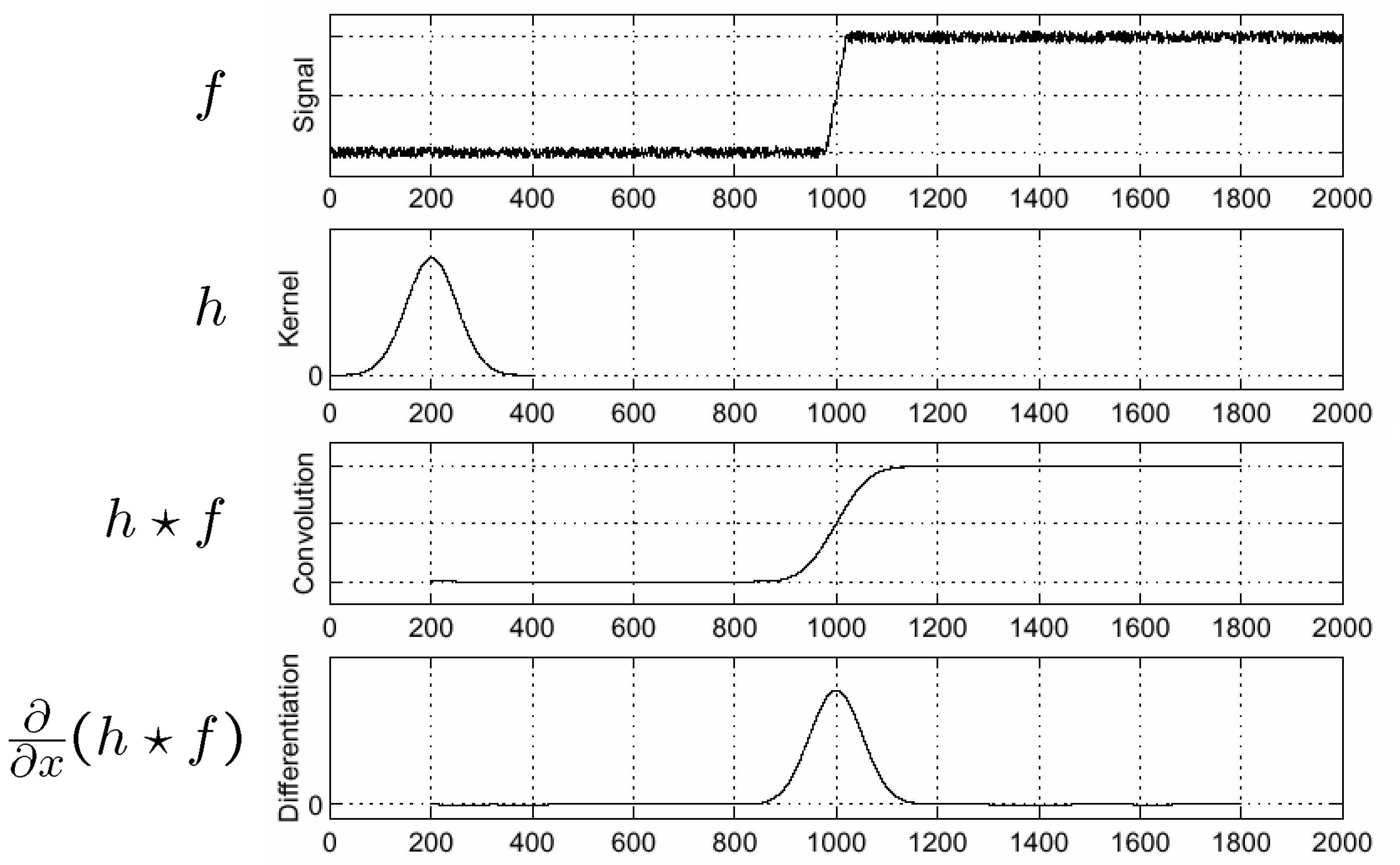
- Plotting intensity as a function of position gives a signal





Where is the edge?

Solution: smooth first



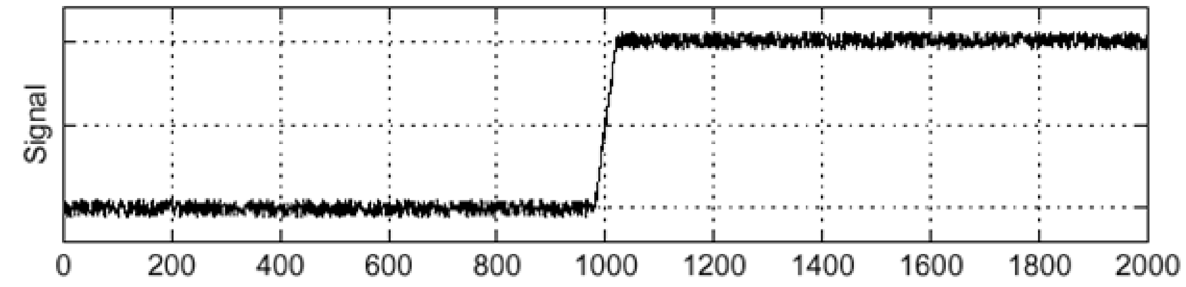
Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

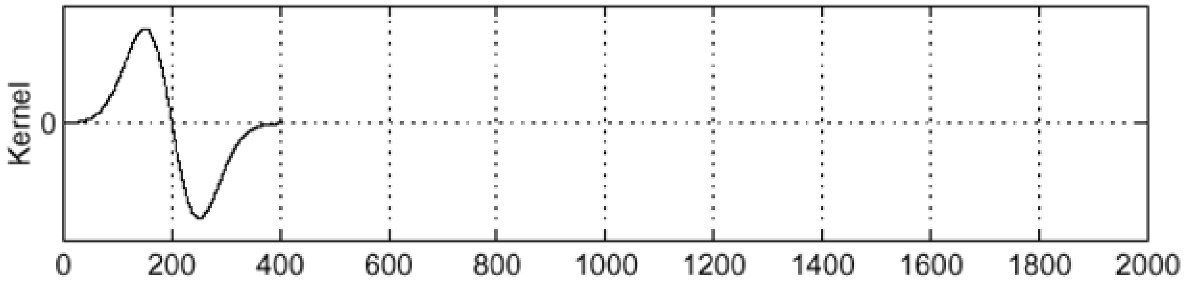
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

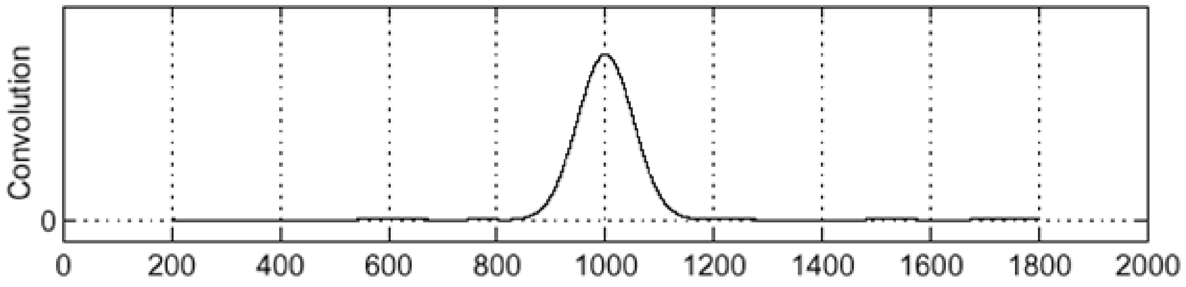
f



$\frac{\partial}{\partial x}h$



$(\frac{\partial}{\partial x}h) \star f$



Need to find (local) maxima of a function

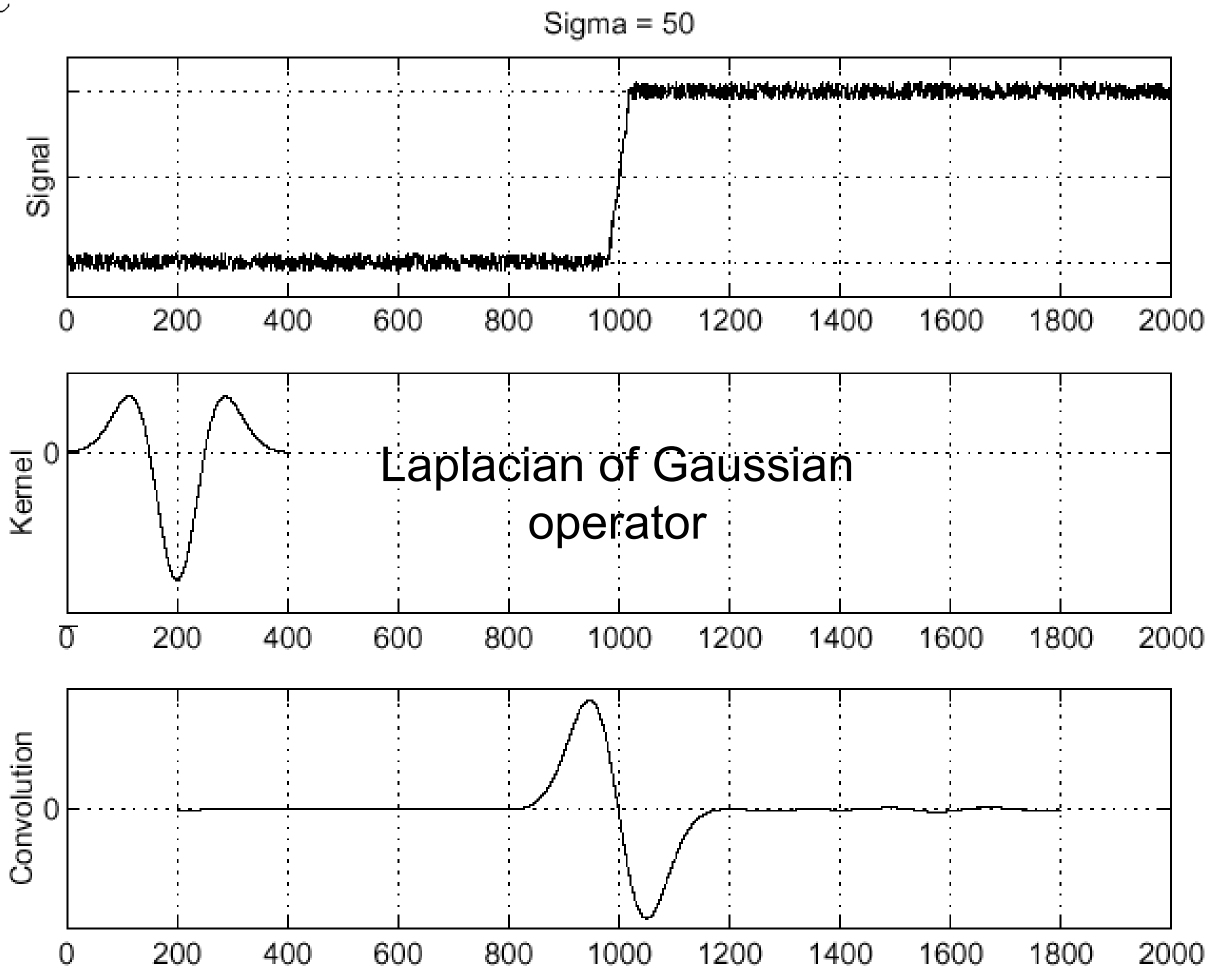
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

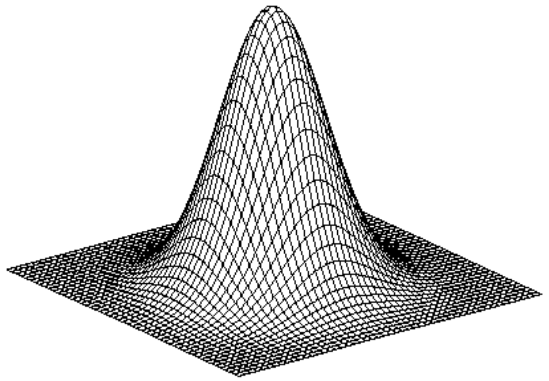
$(\frac{\partial^2}{\partial x^2}h) \star f$



Laplacian of Gaussian operator

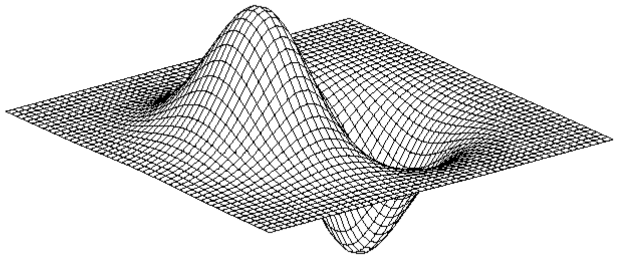
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

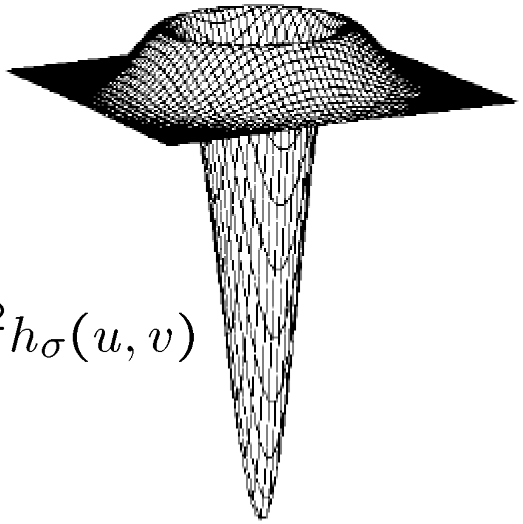
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian (x direction)

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

Laplacian of Gaussian



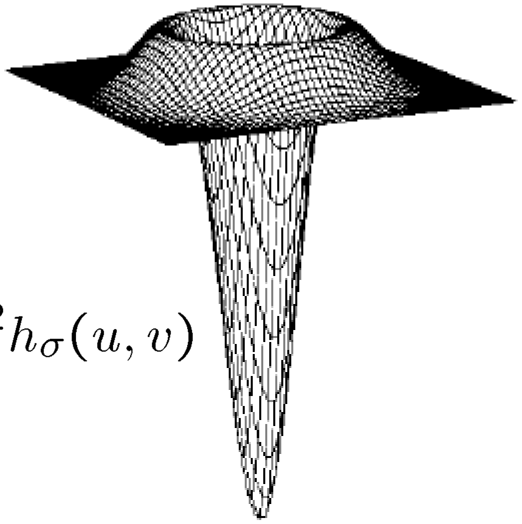
$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

LoG filter

Laplacian of Gaussian



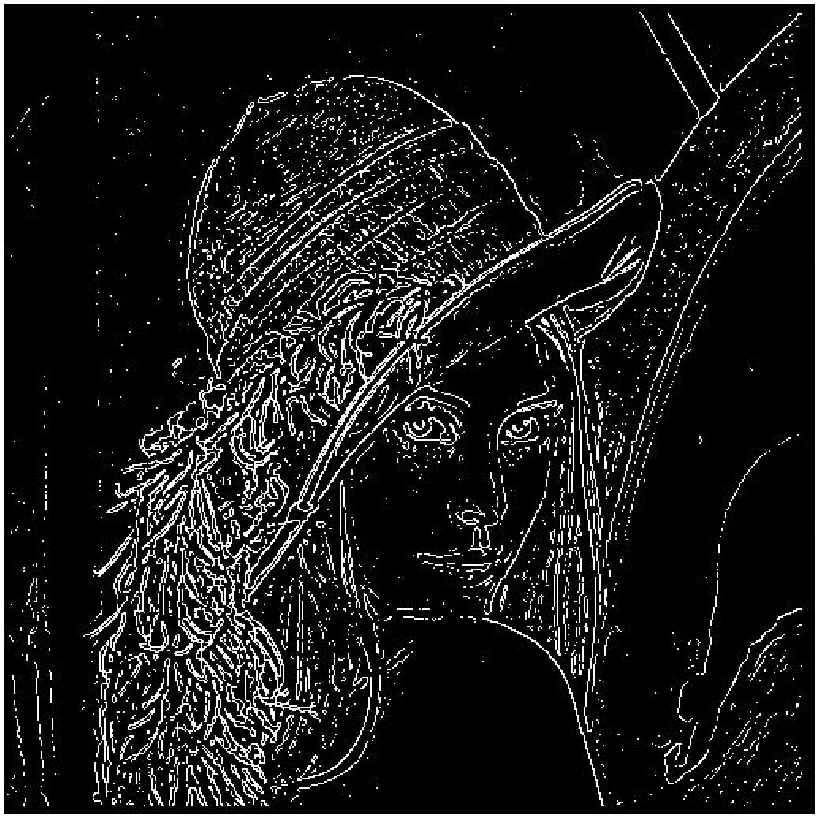
$\nabla^2 h_\sigma(u,v)$

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Discrete approximation with σ = 1.4

Edge detection using LoGs

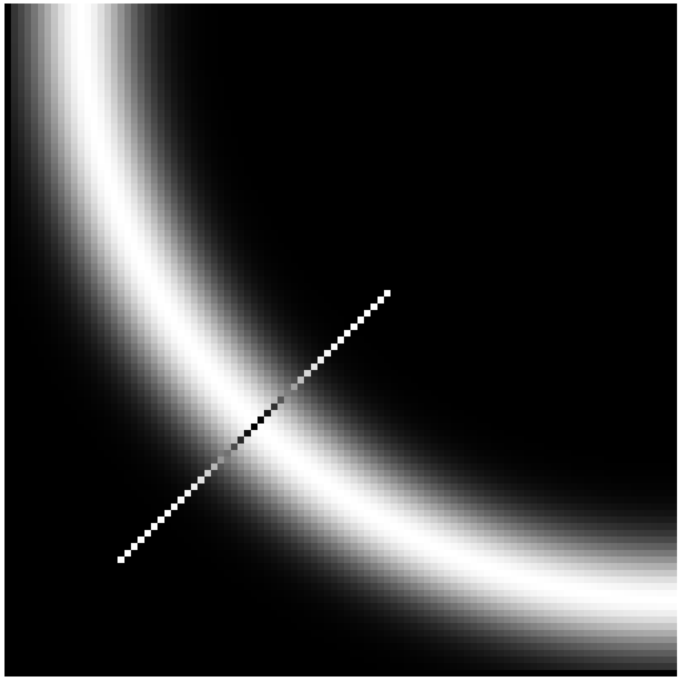
original image (Lena)

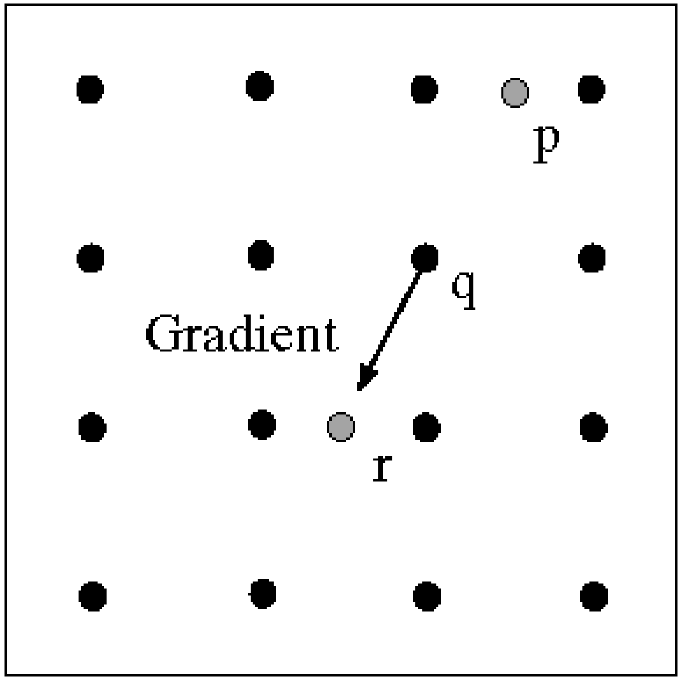

LoG followed by zero crossing detection

Canny Edge Detector

1. Smoothing: Smooth the image with a Gaussian filter with spread σ
2. Gradient: Compute gradient magnitude and direction at each pixel of the smoothed image
3. Thresholding: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. Non-maximum suppression (thinning): Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)

Step 4: Thinning (Non-maximum suppression)





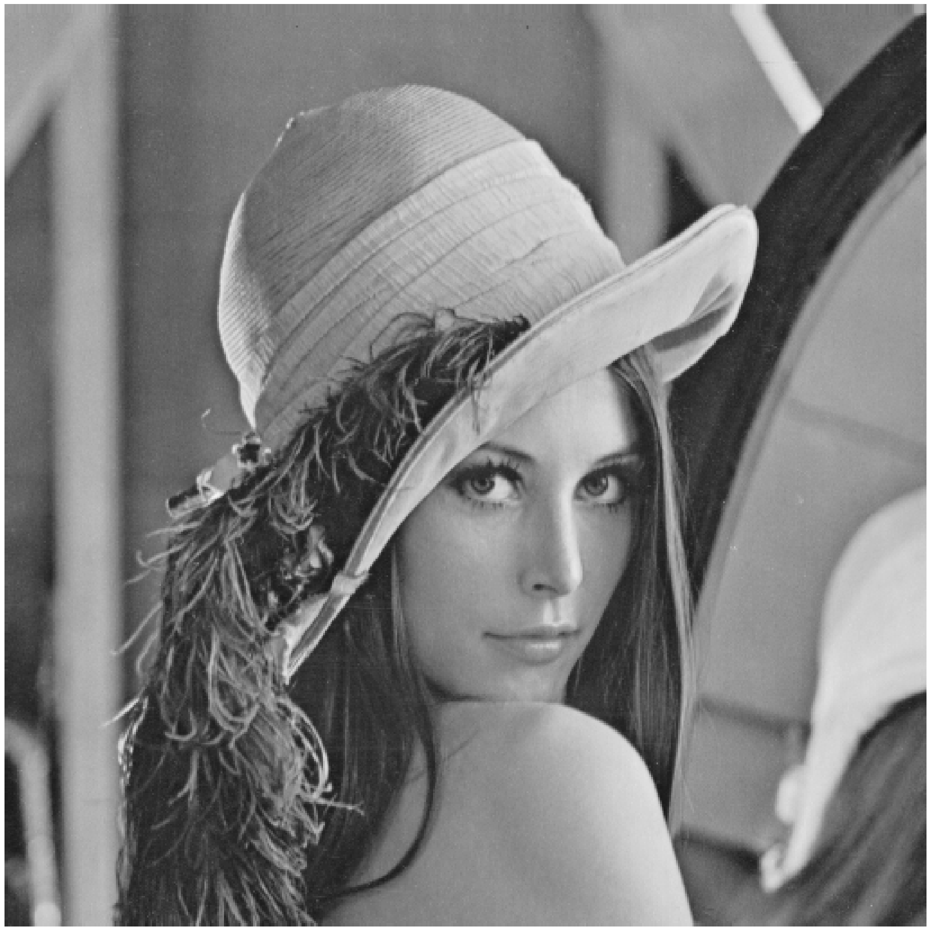
Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

Canny Edge Detector

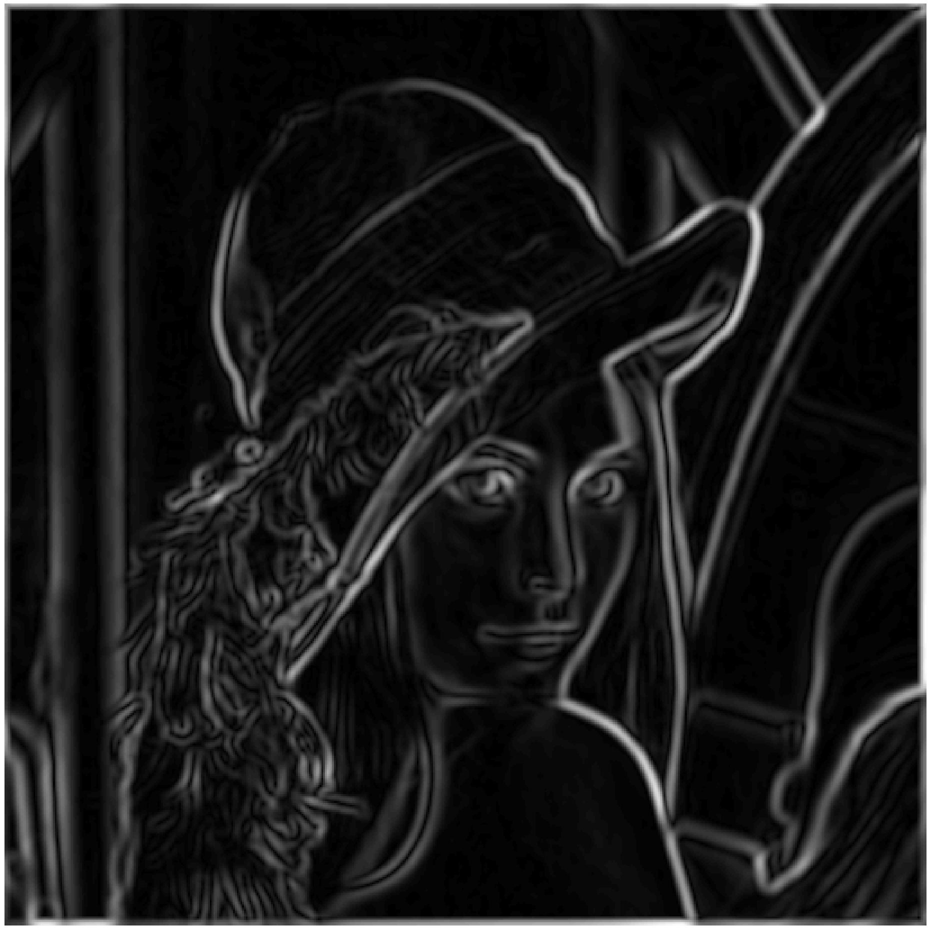
1. Smoothing: Smooth the image with a Gaussian filter with spread σ
2. Gradient: Compute gradient magnitude and direction at each pixel of the smoothed image
3. Thresholding: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. Non-maximum suppression (thinning): Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)
5. Tracing edges: Trace high-magnitude contours and keep only pixels along these contours, so weak little segments go away

The Canny edge detector



original image (Lena)

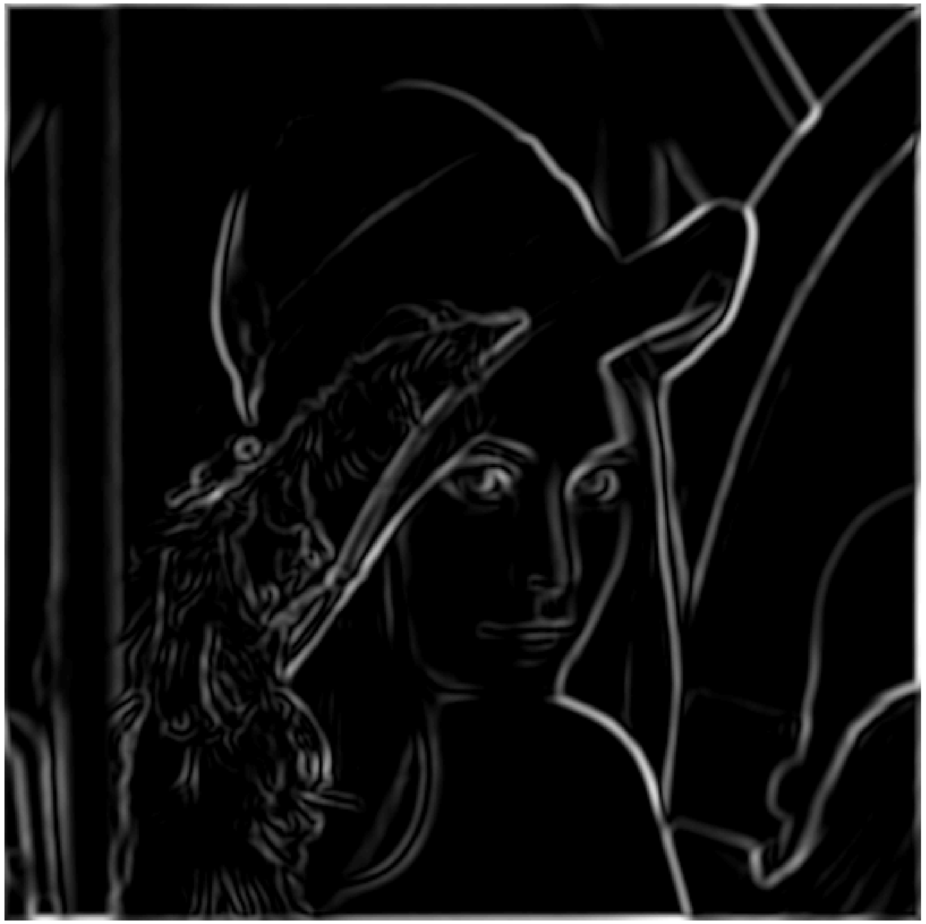
The Canny edge detector: Step 2



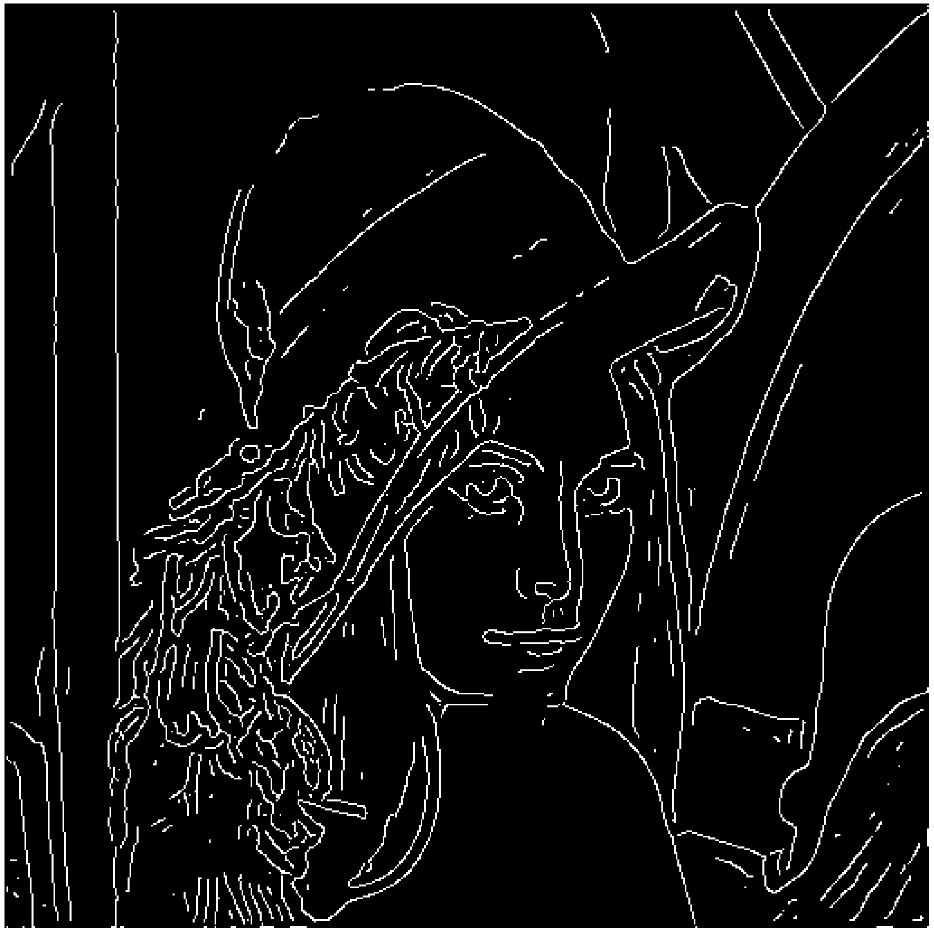
norm of the gradient of smoothed image

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

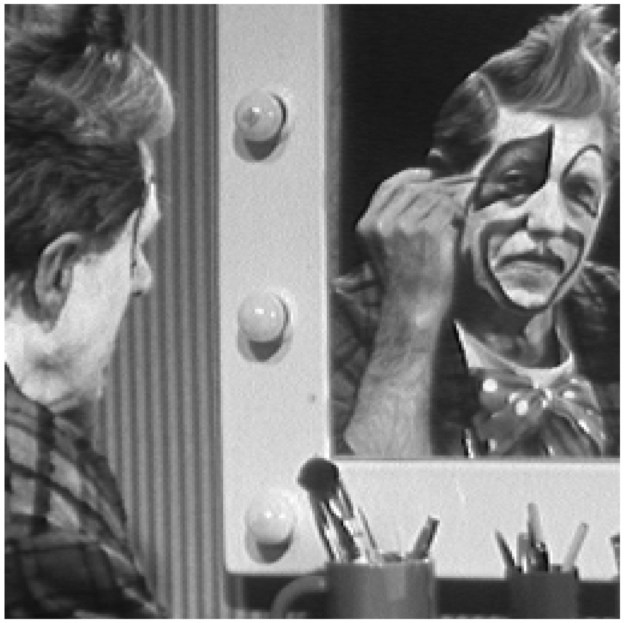
The Canny edge detector: Step 3

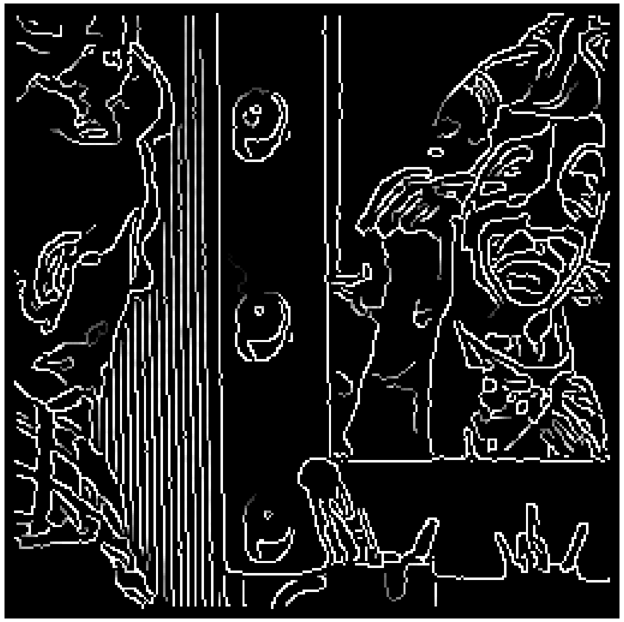

thresholding

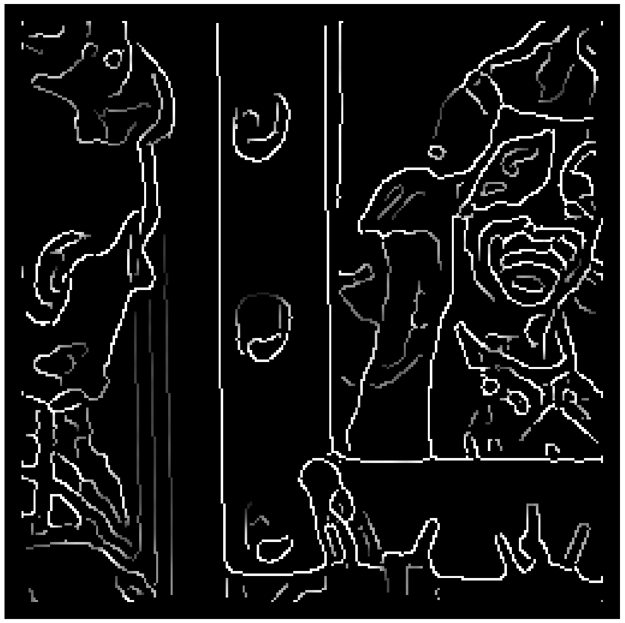
The Canny edge detector: Steps 4 & 5


Thinning (single pixel edges) & tracing edges

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

Next Time: Intelligent image scissors

- Things to do:
  - Read intelligent scissors paper(s) online
  - Project 1 will be assigned next class
  - Prepare for C/C++ programming
  - Visit Vision and Graphics Lab (Sieg 327)
    - Your ID card should open Sieg 327

Lena (1972)

Lena in 1997