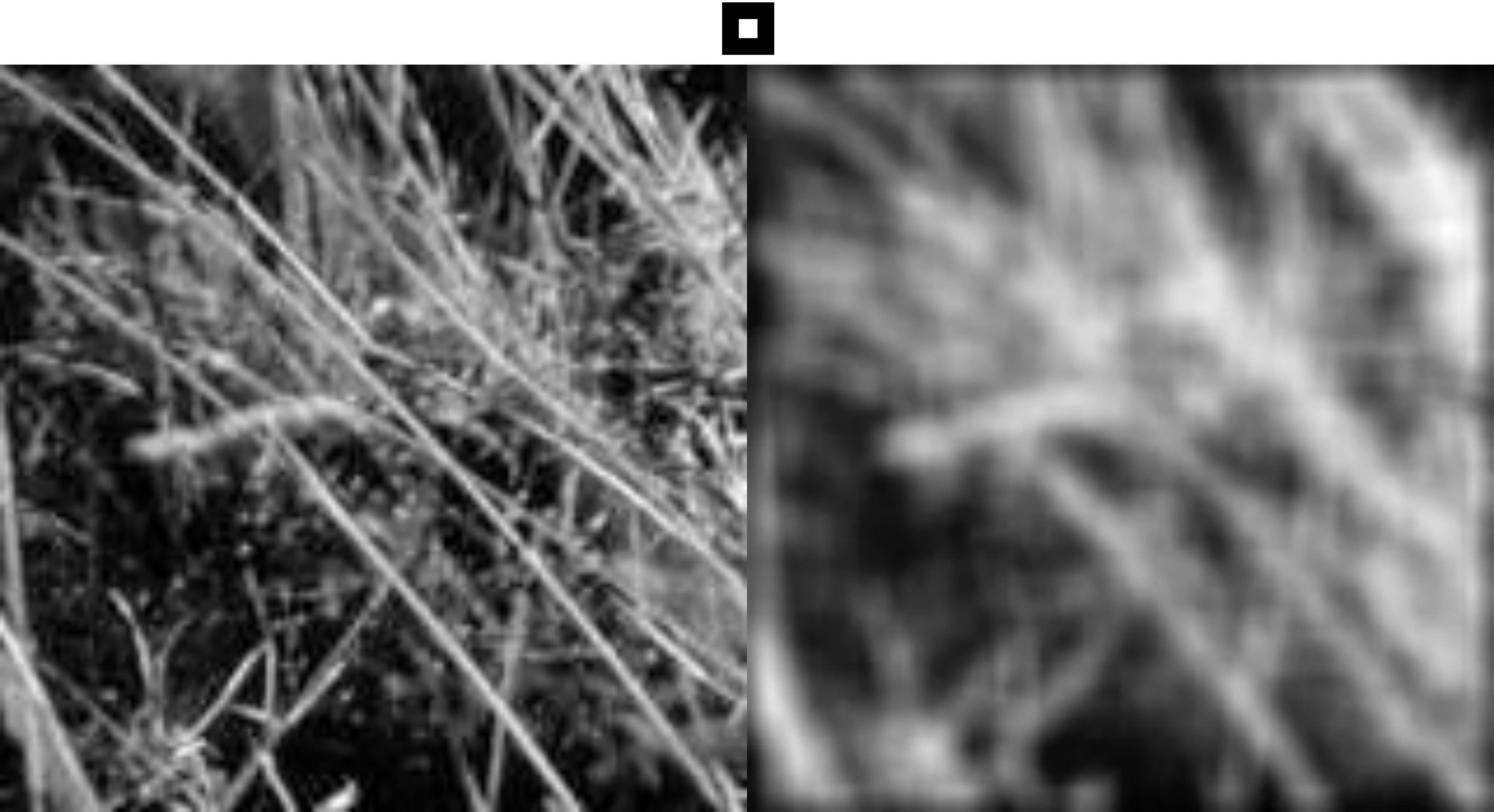
Convolution and Image Derivatives

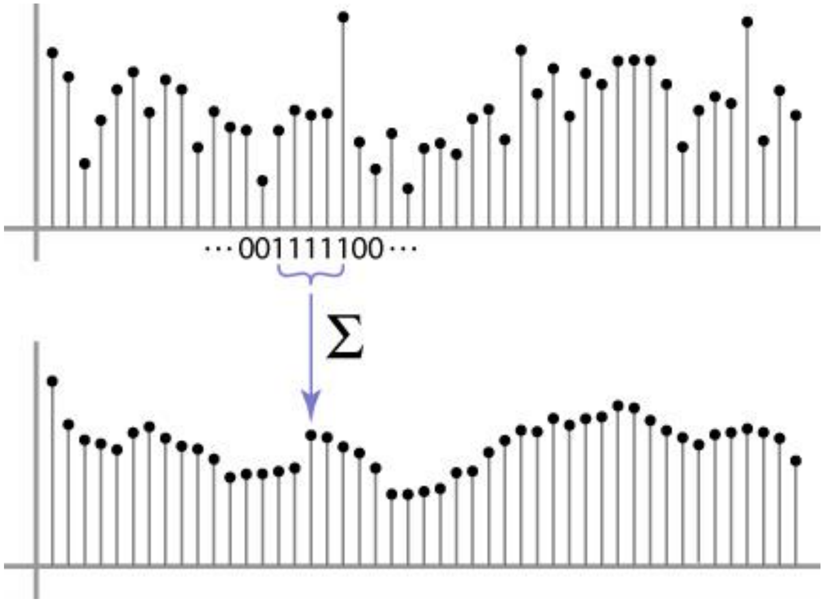
CS180: Intro to Comp. Vision and Comp. Photo
Alexei Efros & Angjoo Kanazawa, UC Berkeley, Fall 2023

box filter is not the best…



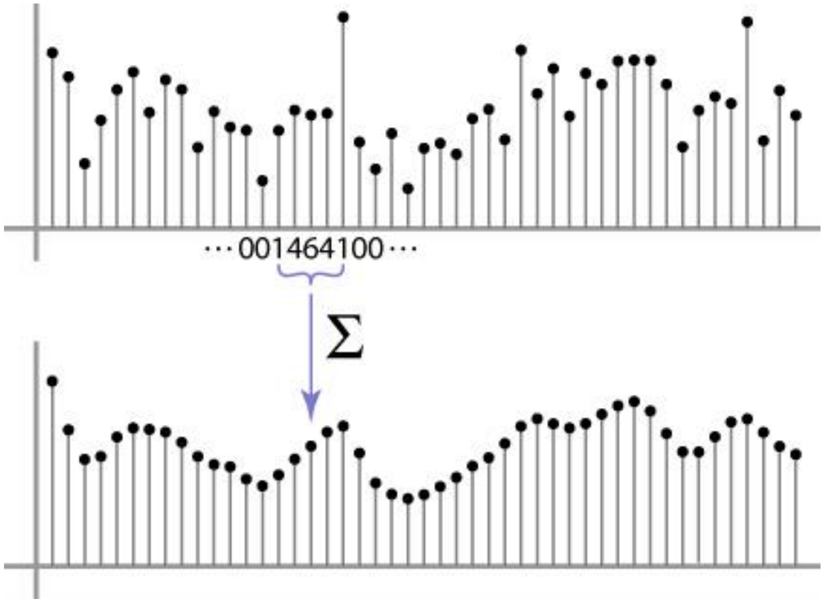
Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5



Weighted Moving Average

- bell curve (gaussian-like) weights [..., 1, 4, 6, 4, 1, ...]



Moving Average In 2D

What are the weights H?

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

$F[x, y]$

$H[u, v]$

 Slide by Steve Seitz

Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

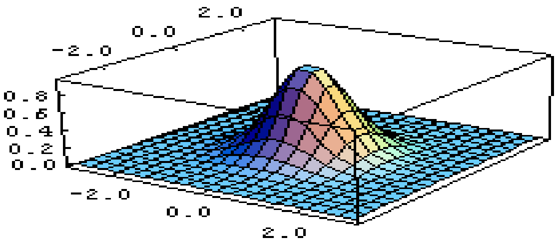
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

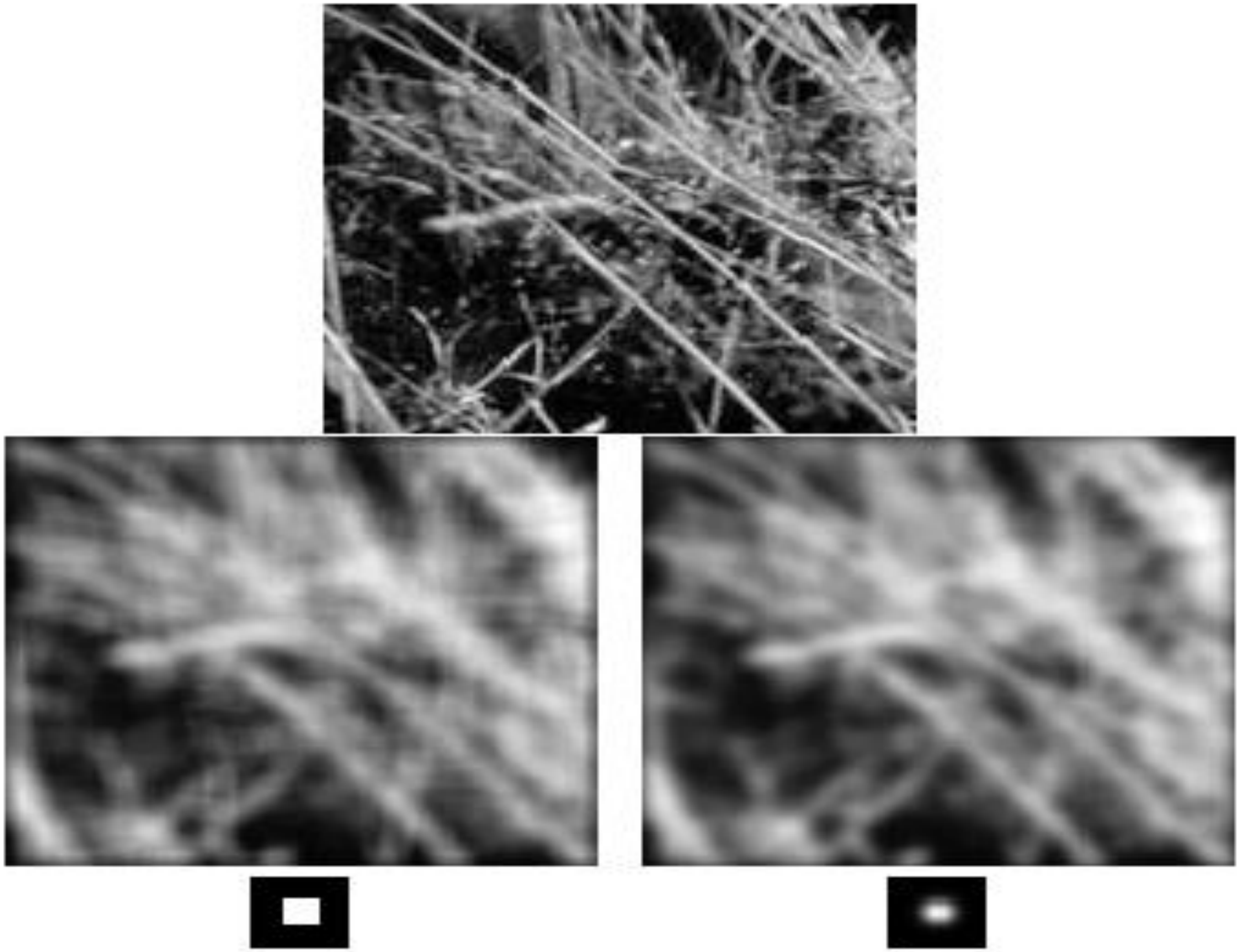


$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

This kernel is an approximation of a Gaussian function:

Slide by Steve Seitz

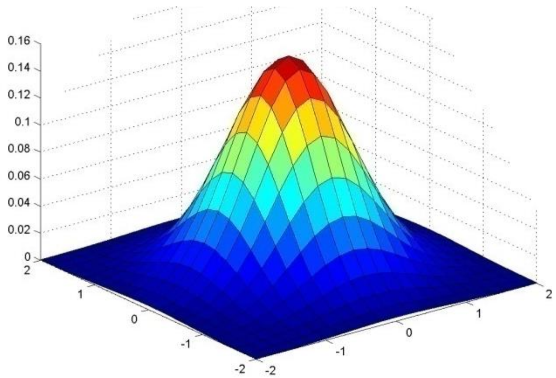
Mean vs. Gaussian filtering

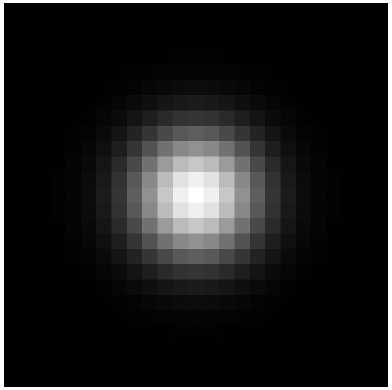


Slide by Steve Seitz

Important filter: Gaussian

Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

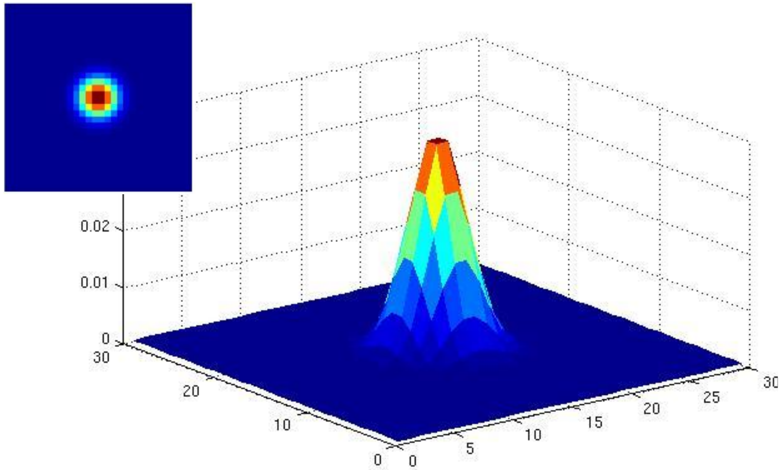
5 x 5, σ = 1

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

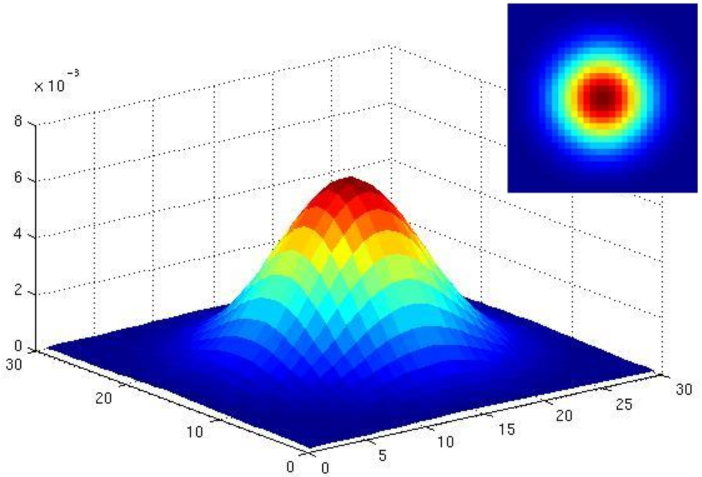
Slide credit: Christopher Rasmussen

Gaussian Kernel

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



σ = 2 with 30 x 30 kernel

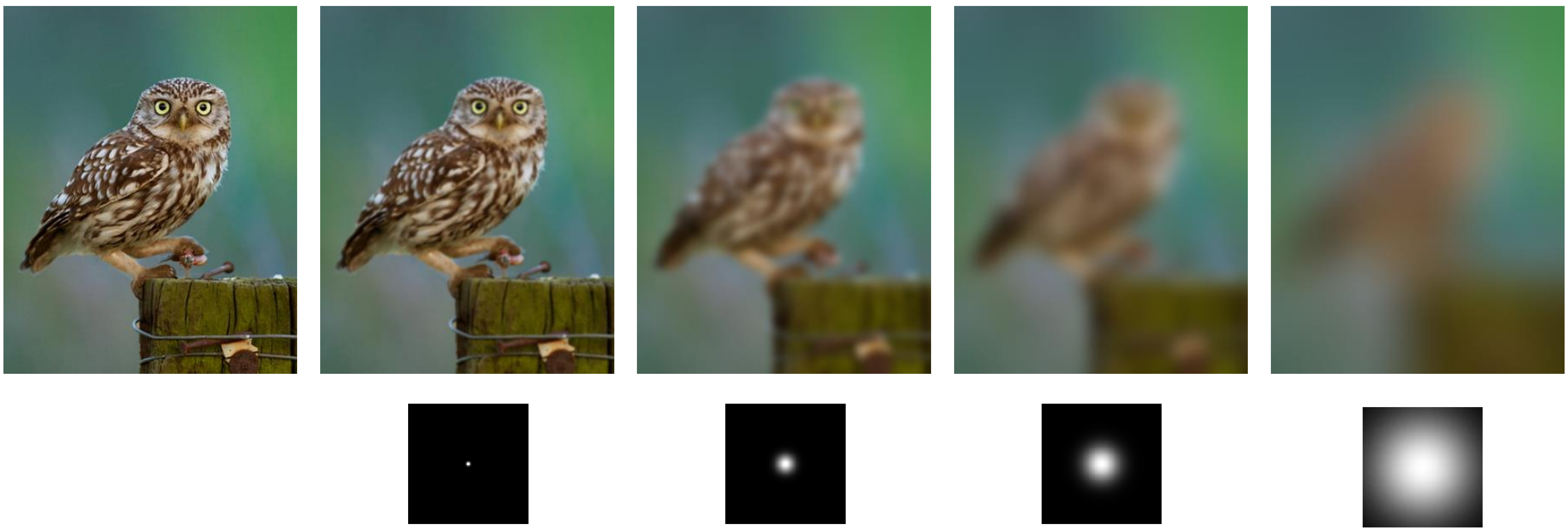


σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

Source: K. Grauman

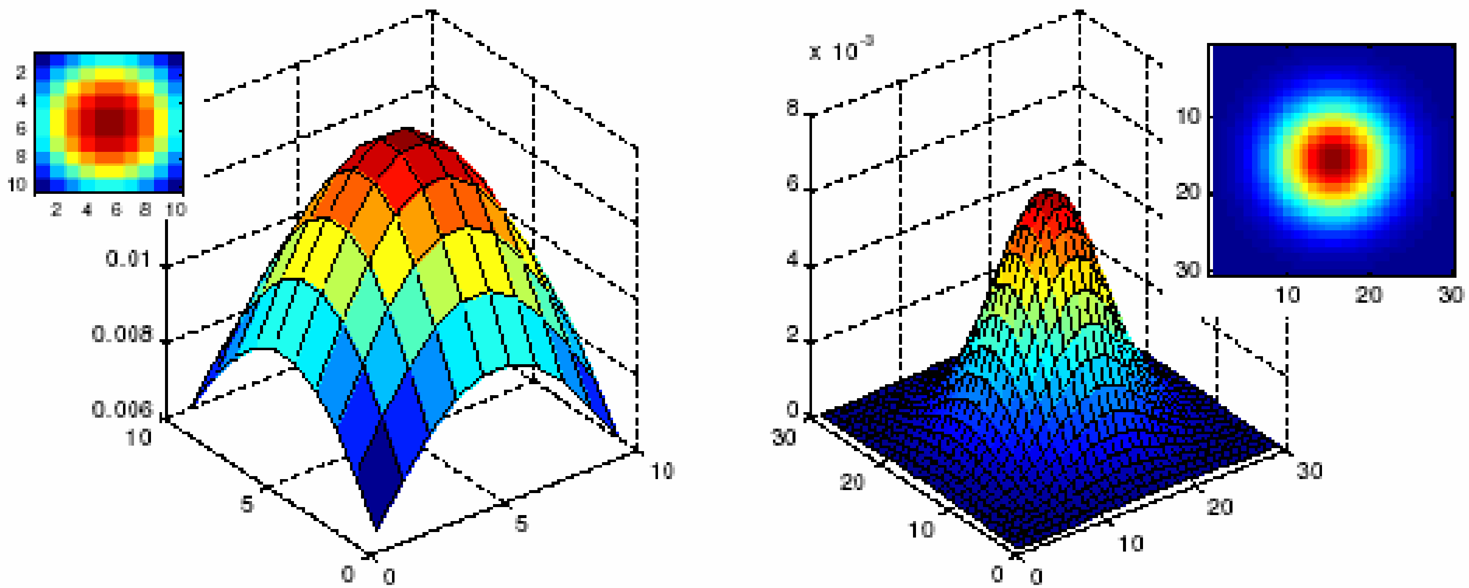
Gaussian filters



σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels



σ = 5 with 10x10 kernel

σ = 5 with 30x30 kernel

Source: K. Grauman

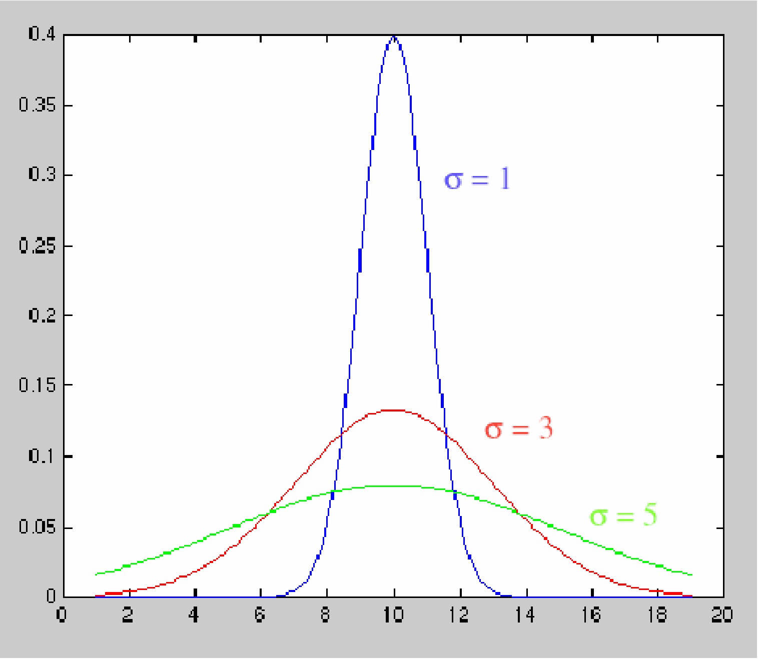
Practical matters

How big should the filter be?

Values at edges should be near zero

Rule of thumb for Gaussian: set filter half-width to about 3 σ

Effect of σ



Side by Derek Hoiem

Cross-correlation vs. Convolution

cross-correlation: $$G = H \otimes F$$

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

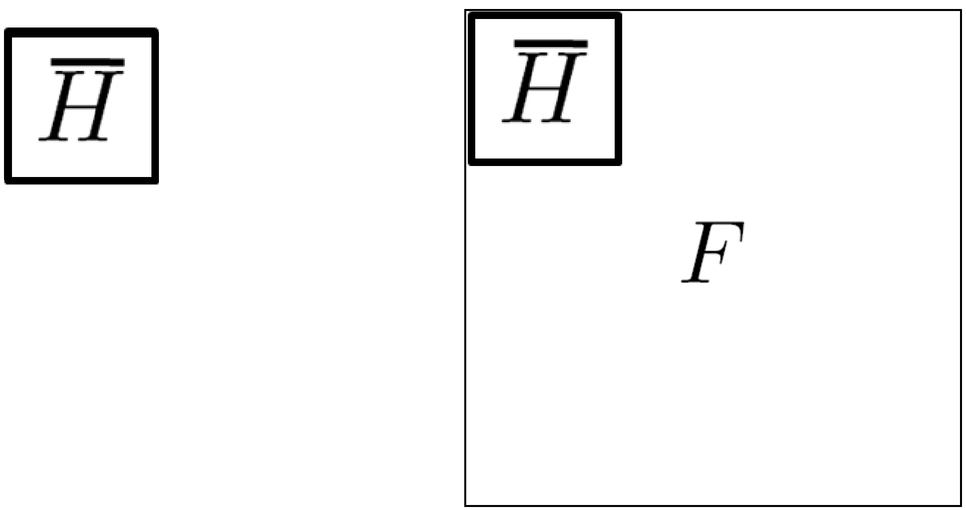
It is written:

$$G = H \star F$$

Convolution is **commutative** and **associative**

Slide by Steve Seitz

Convolution



Adapted from F. Durand

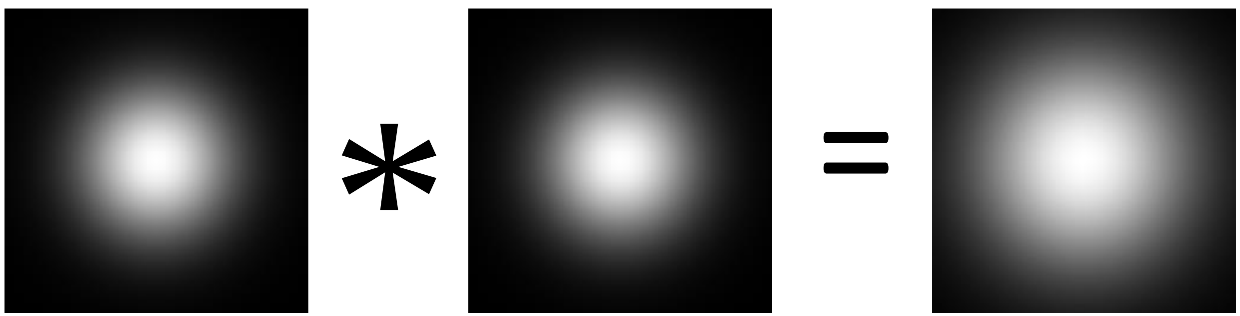
Convolution is nice!

- Notation: $b = c \star a$
- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - identity: unit impulse *e* = [..., 0, 0, 1, 0, 0, ...]

    $$a \star e = a$$
- Conceptually no distinction between filter and signal
- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: a * $(b_1 * b_2 * b_3)$

Gaussian and convolution

- Removes "high-frequency" components from the image (low-pass filter)
- Convolution with self is another Gaussian



  – Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

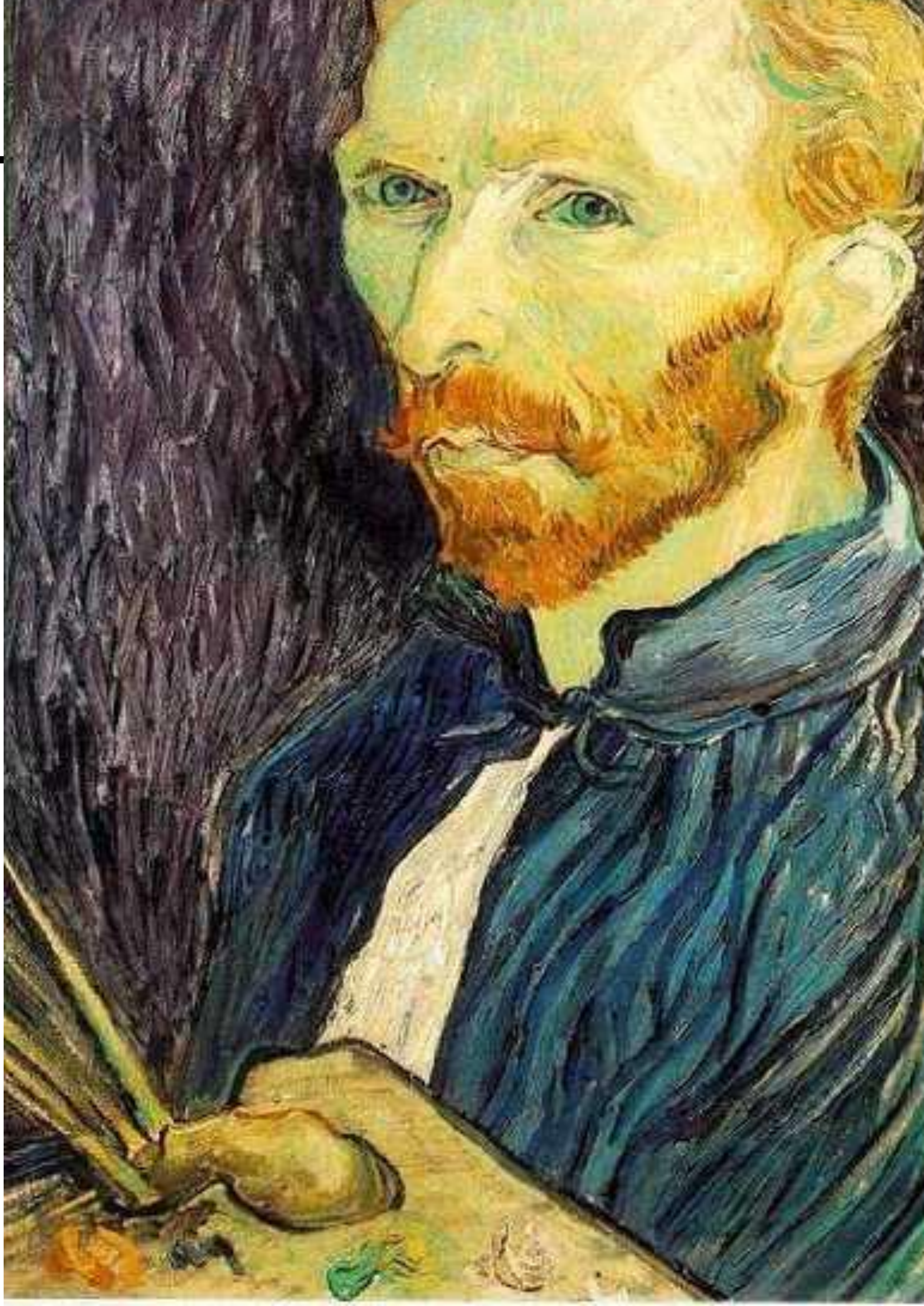
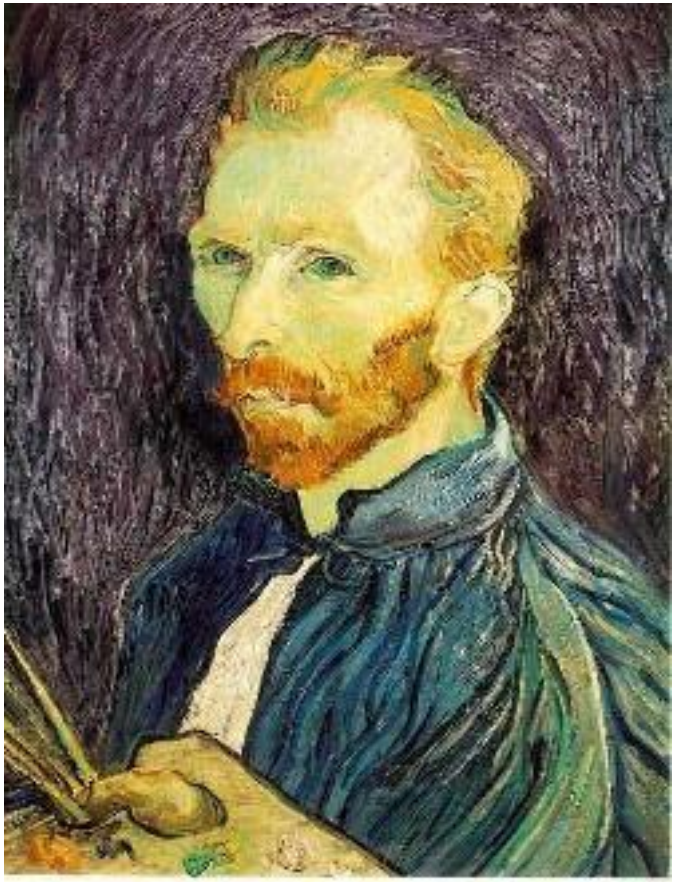


Image sub-sampling



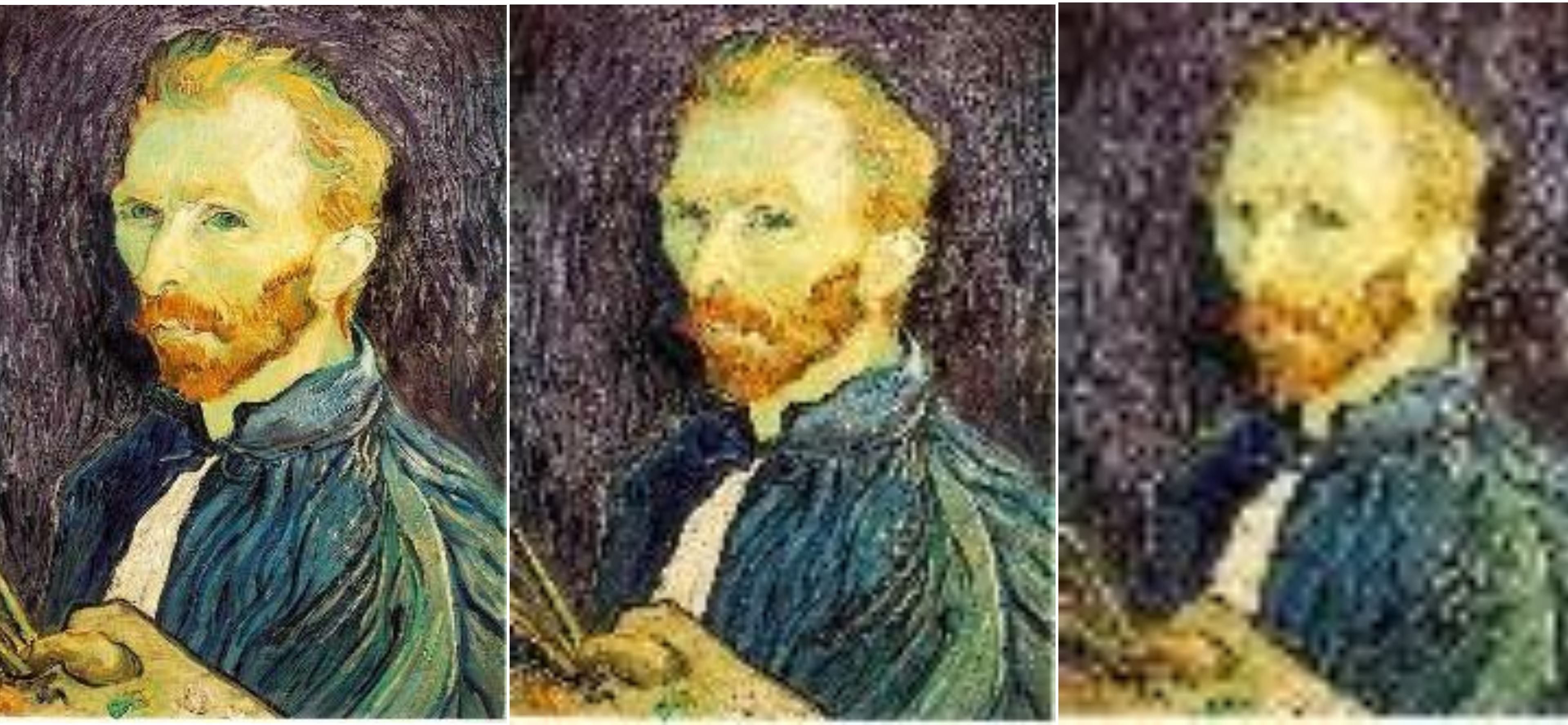
1/4

1/8

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

Slide by Steve Seitz

Image sub-sampling

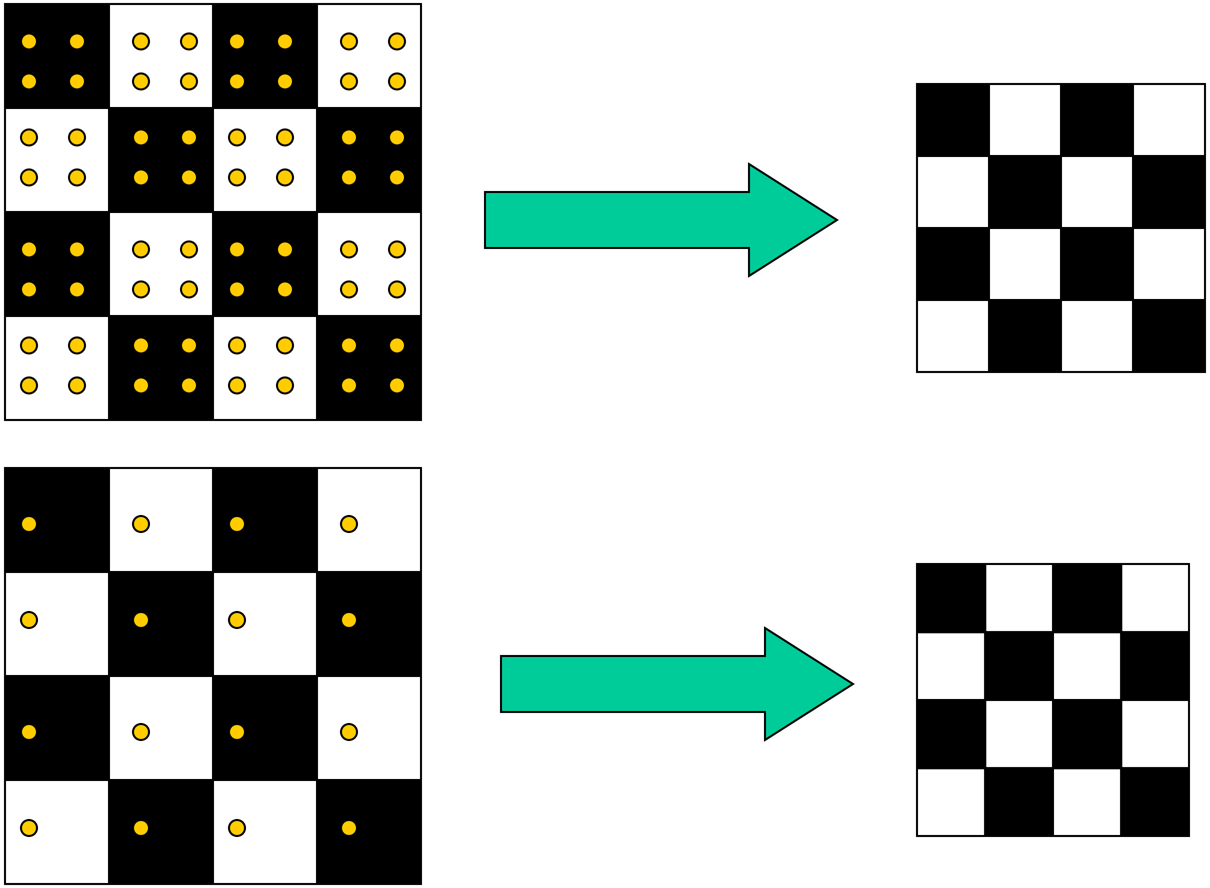


1/2 1/4 (2x zoom) 1/8 (4x zoom)

Aliasing! What do we do?

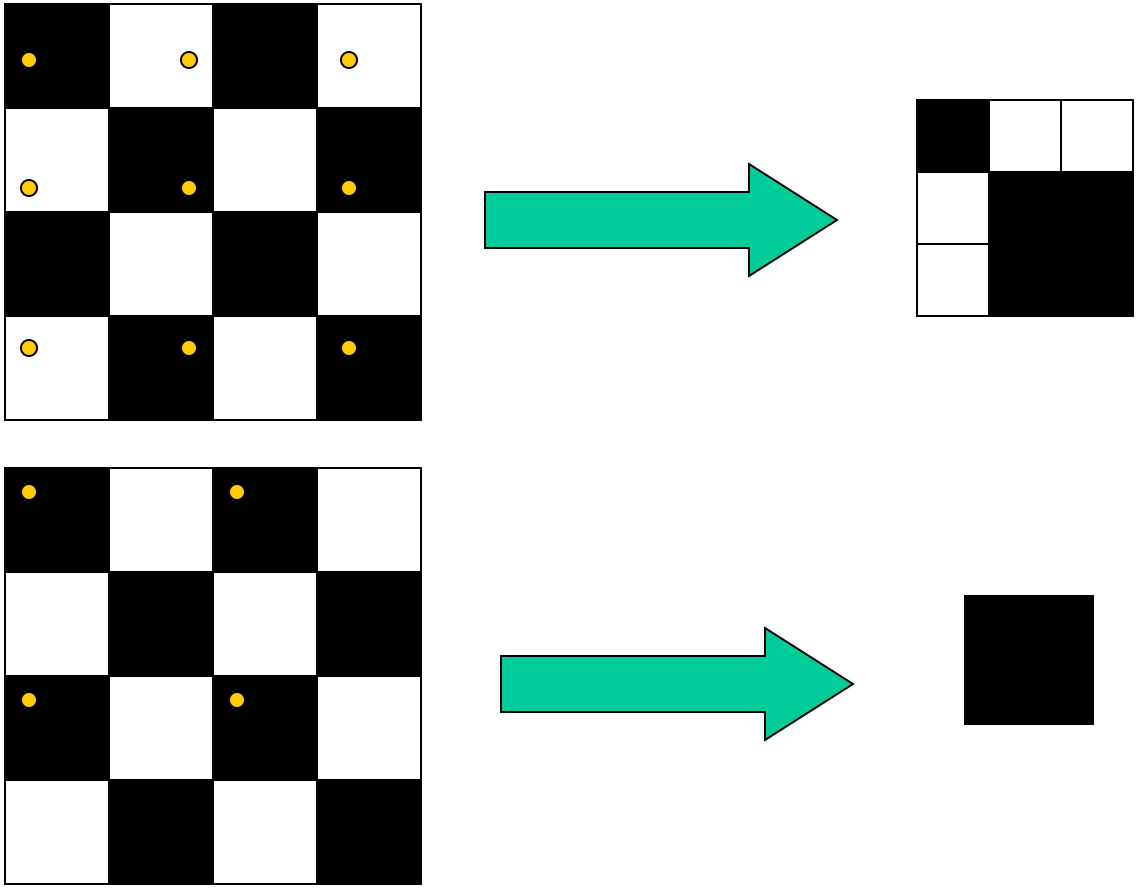
Slide by Steve Seitz

Sampling an image



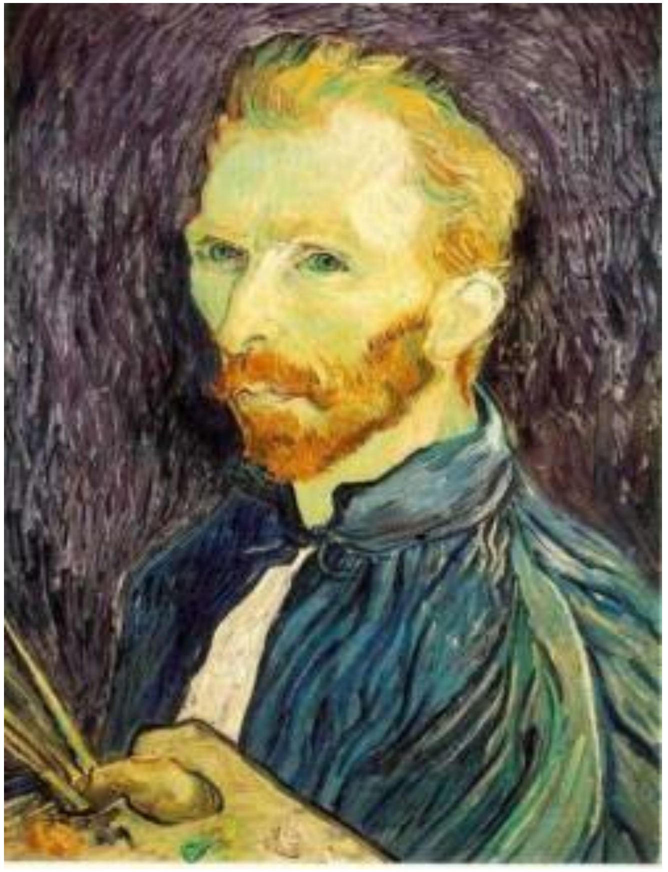
Examples of GOOD sampling

Undersampling



Examples of BAD sampling -> Aliasing

Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

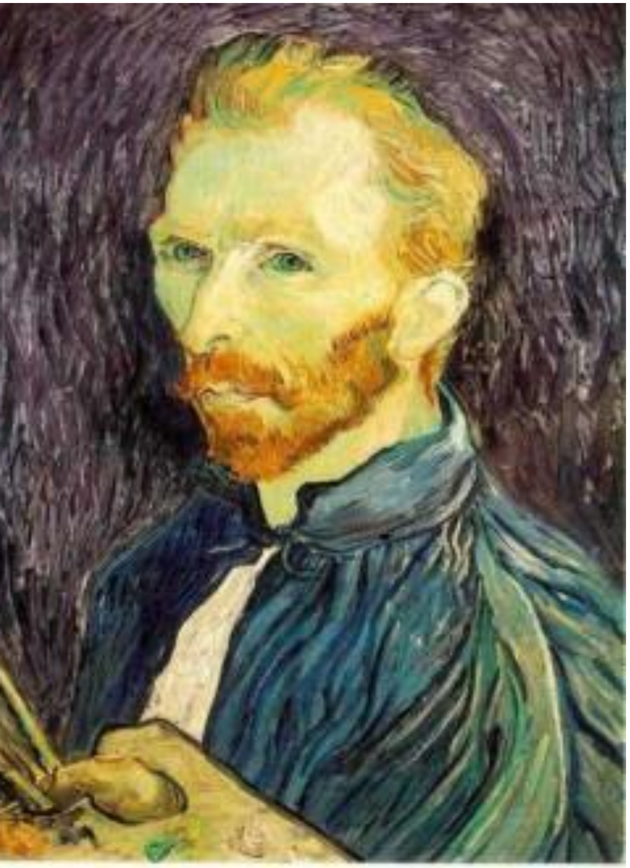
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering

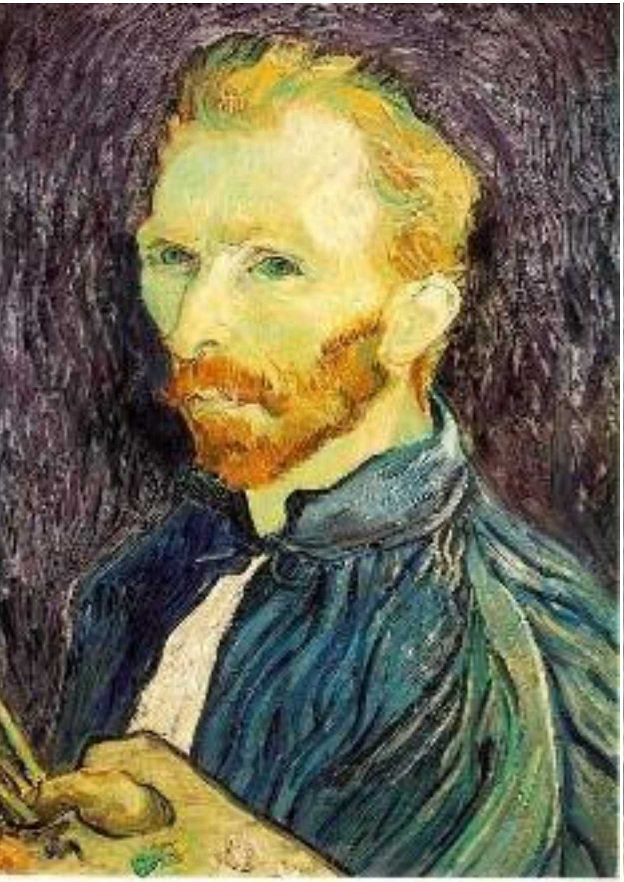

Gaussian 1/2


G 1/4


G 1/8

Slide by Steve Seitz

Compare with...



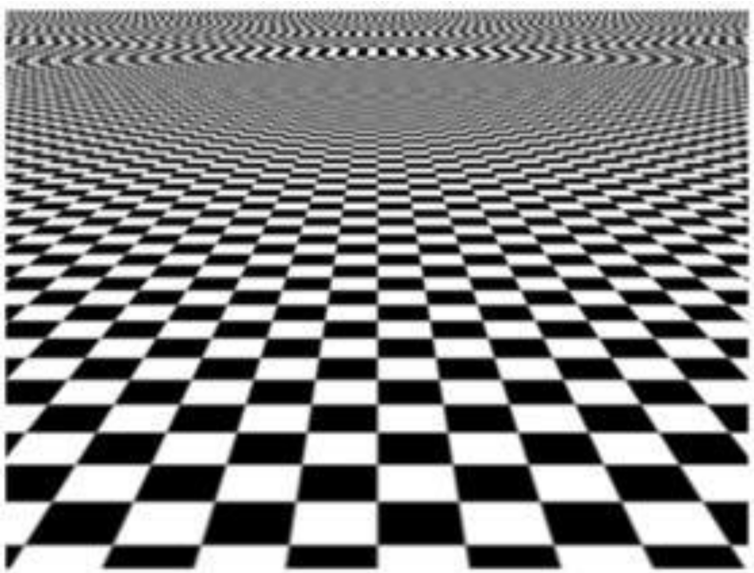
1/2

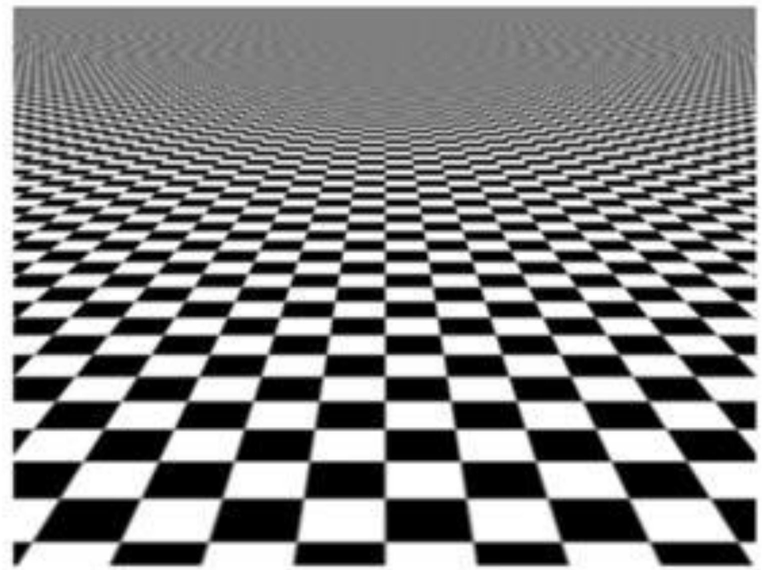
1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

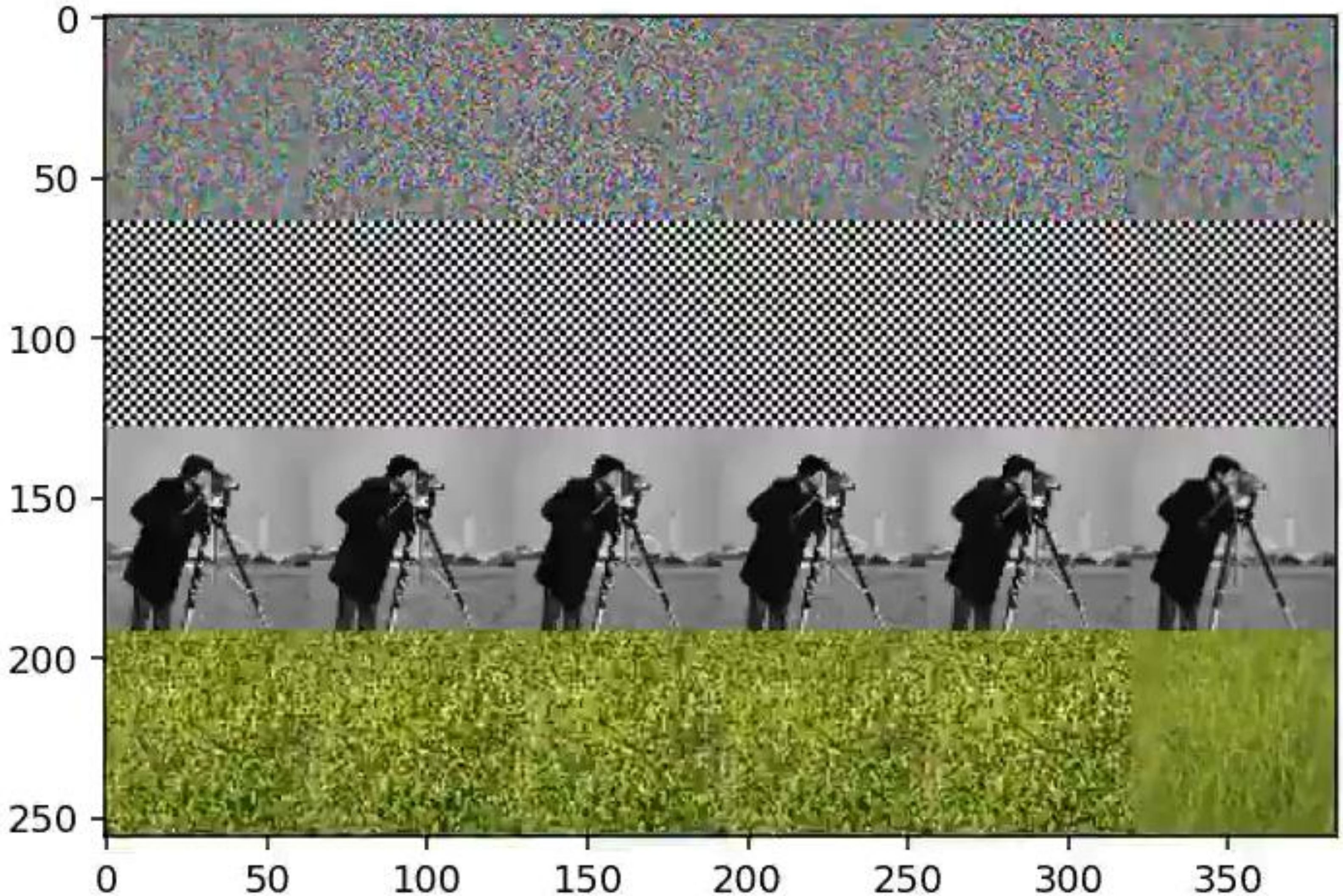
More Gaussian pre-filtering





A real problem!

128 x 128 → 64x64



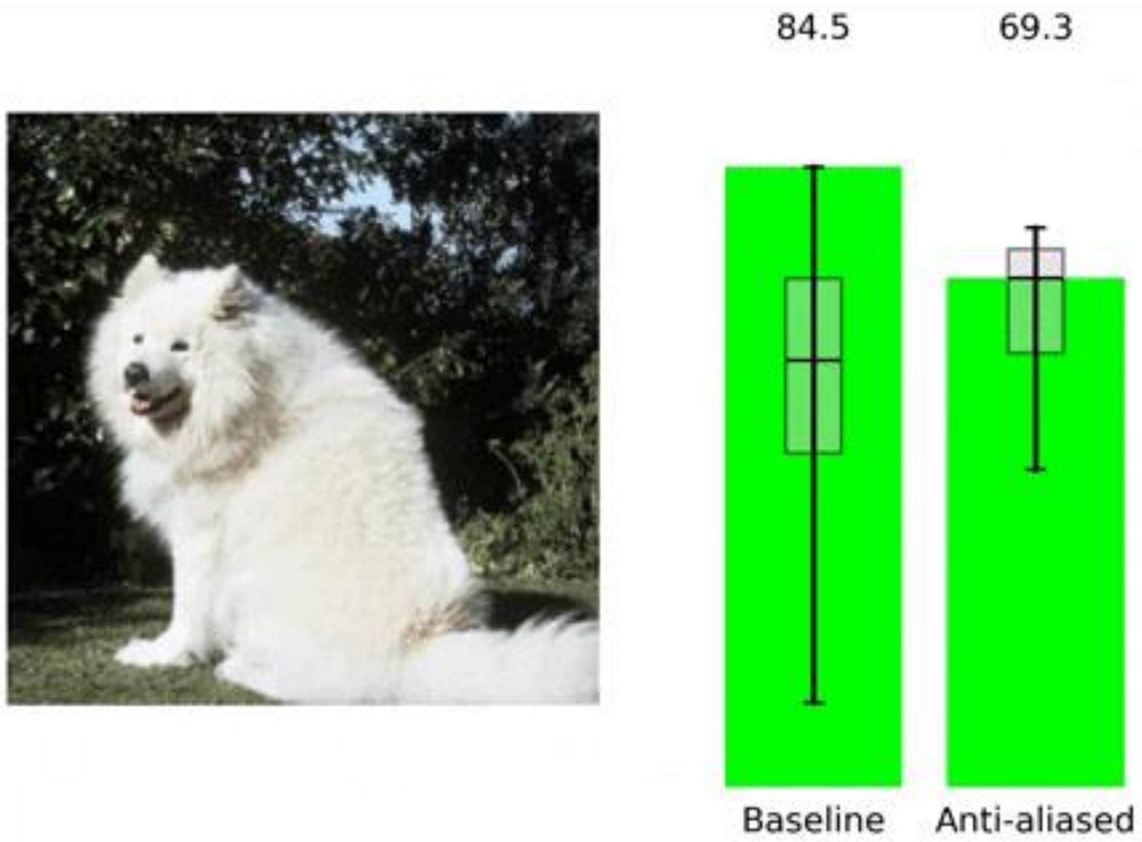
Open-CV:
default, bicubic, Lanczos4

Pytorch:
bilinear, bicubic

PIL: Lanczos

[Credit: @jaakkolehtinen]()

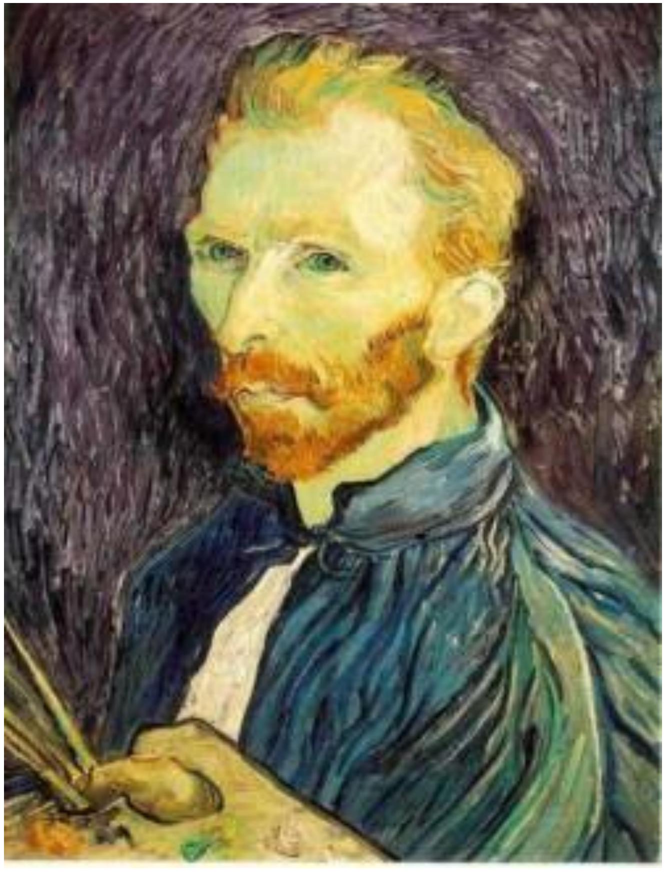
problems in NN too




```
pip install antialiased-cnns
```

Making Convolutional Networks Shift-Invariant Again,
Richard Zhang ICML 2019

Iterative Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

G 1/8

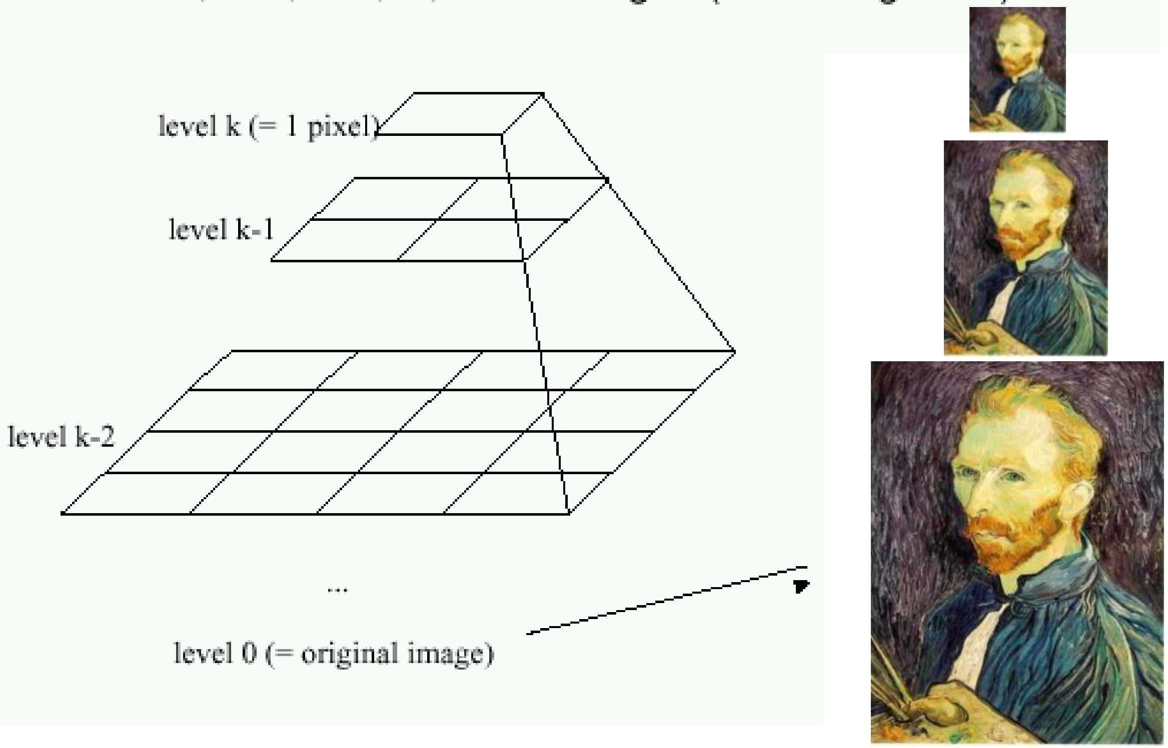
filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Slide by Steve Seitz

Image Pyramids

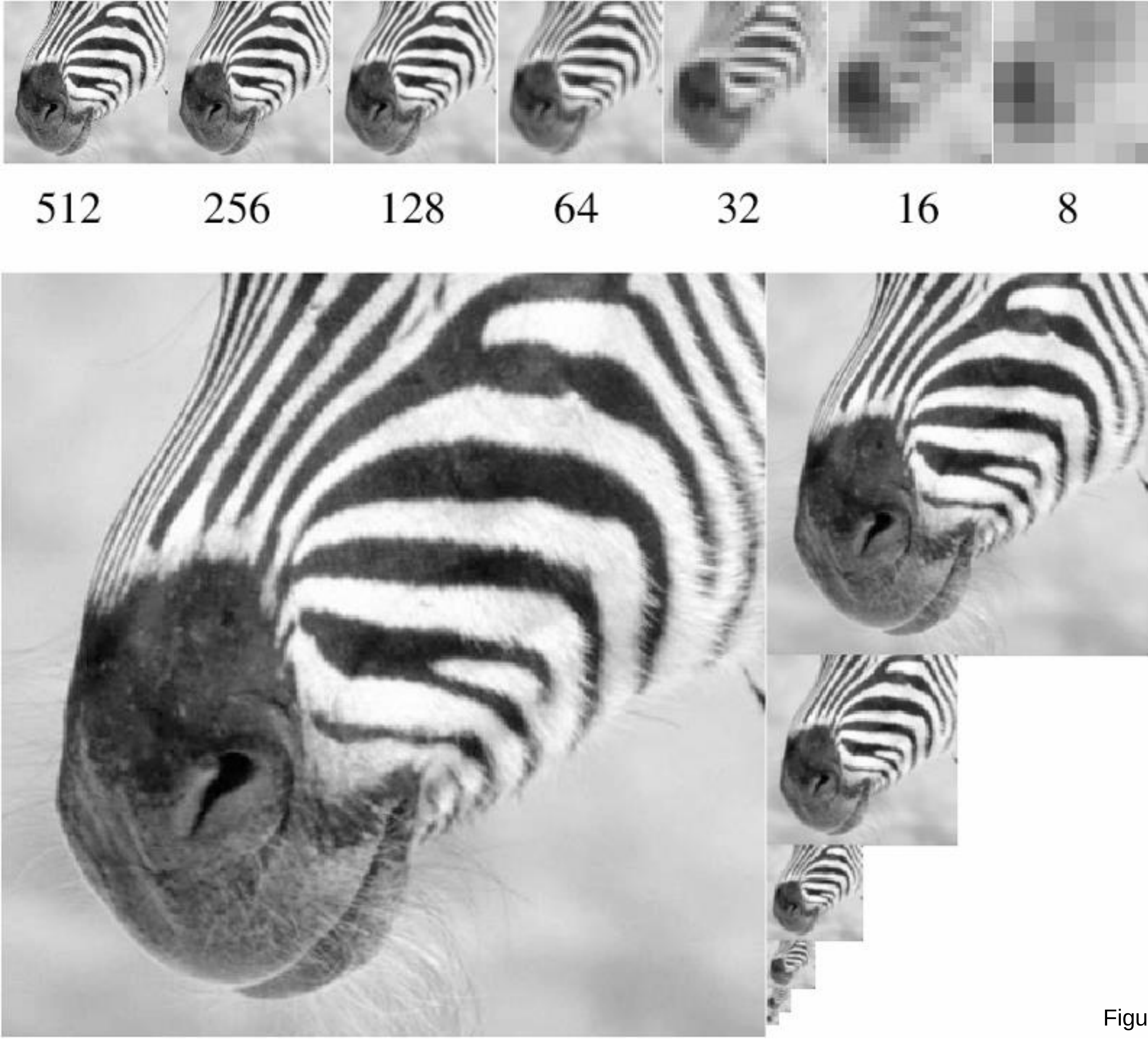
Idea: Represent NxN image as a "pyramid" of $1\text{x}1$, $2\text{x}2$, $4\text{x}4$,…, $2^k\text{x}2^k$ images (assuming $N=2^k$)



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

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

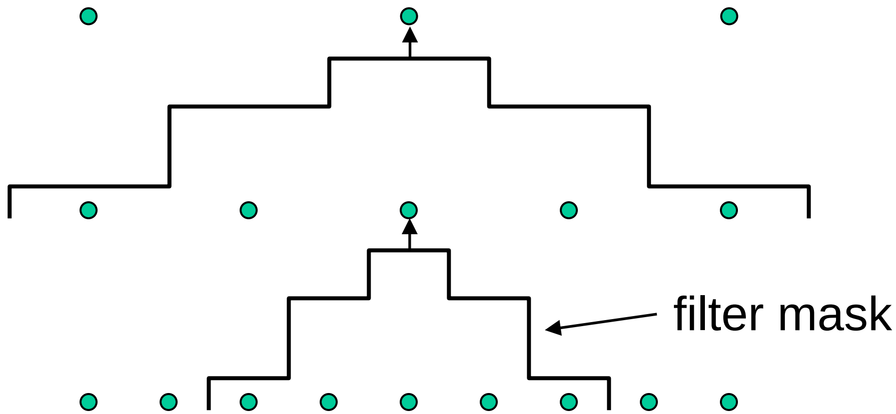
Slide by Steve Seitz

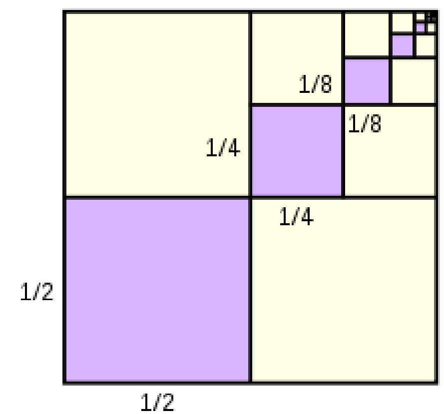


A bar in the big images is a hair on the zebra's nose; in smaller images, a stripe; in the smallest, the animal's nose

Figure from David Forsyth

Gaussian pyramid construction





Repeat

- Filter
- 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!

Slide by Steve Seitz

What are they good for?

Improve Search

- Search over translations
  - Classic coarse-to-fine strategy
  - Project 1!
- Search over scale
  - Template matching
  - E.g. find a face at different scales

What else are convolutions good for?

Taking derivative by convolution

(on board)

Partial derivatives with convolution

Image is function f(x,y)

Remember:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

Approximate:

-1	1

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

Another one:

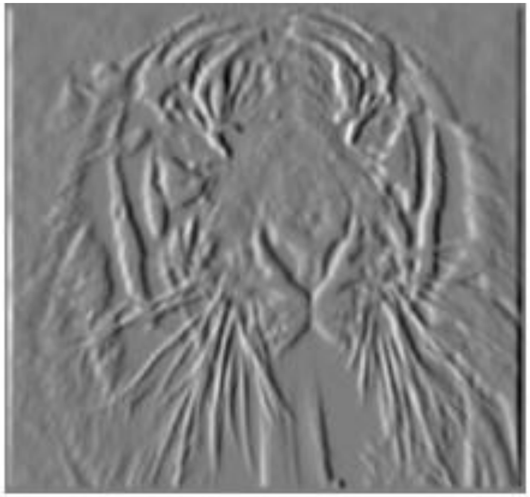
-1	0	1

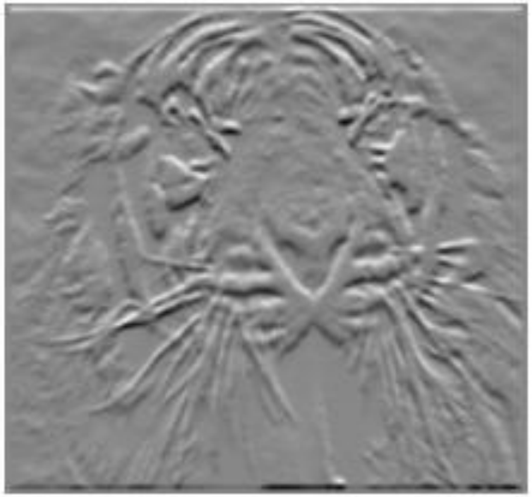
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x-1,y)}{2}$$

Partial derivatives of an image

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

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





-1	**1**

-1
1

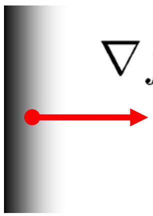
or

1
-1

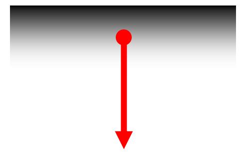
Which shows changes with respect to x?

Image gradient

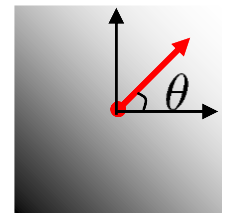
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

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

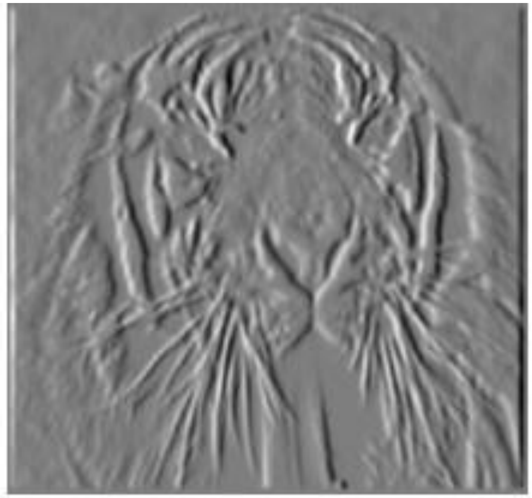
The *edge strength* is given by the gradient magnitude

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

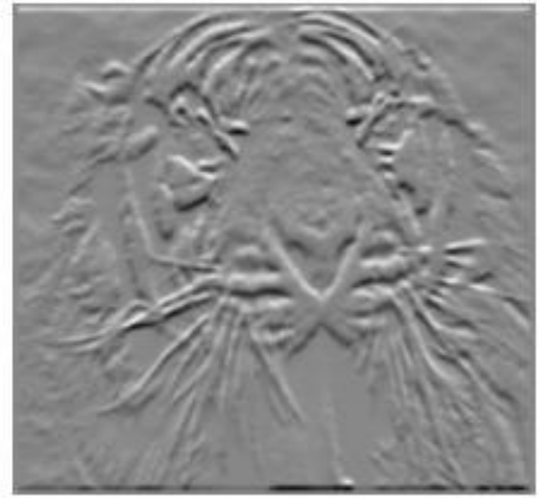
The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

Source: Steve Seitz

Image Gradient



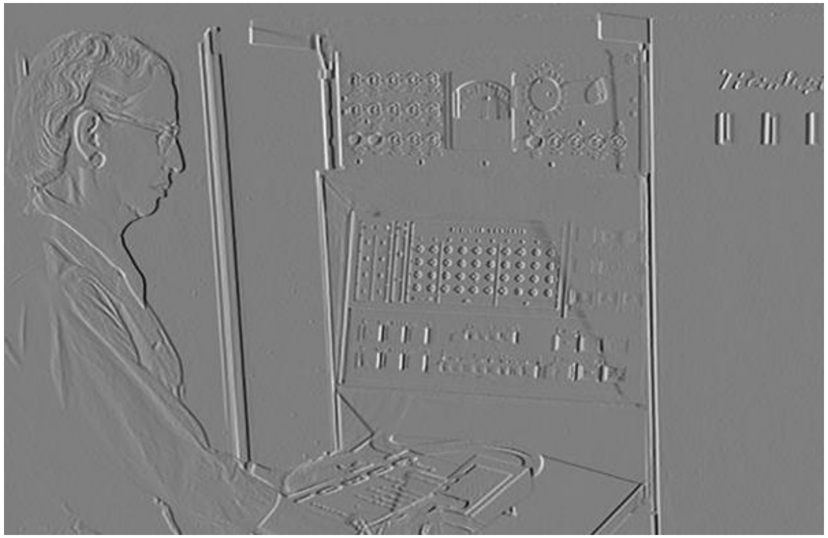
$$\frac{\partial f(x,y)}{\partial x}$$



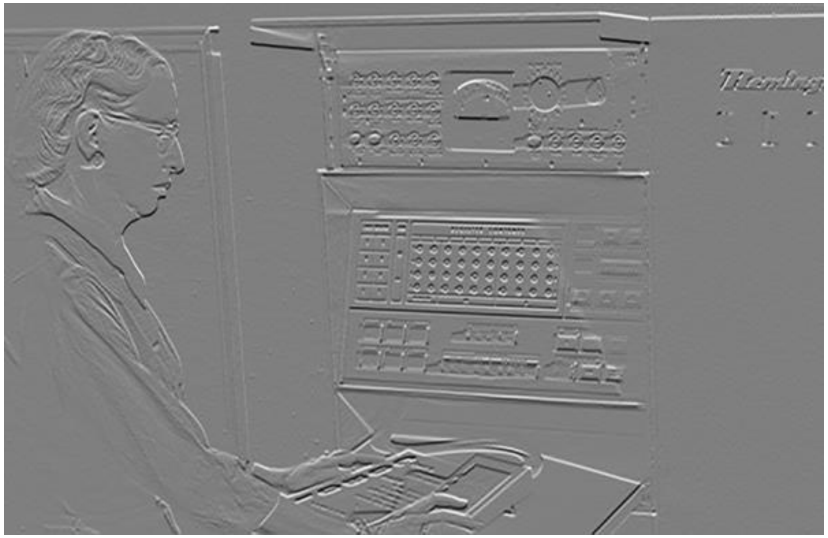
$$\frac{\partial f(x,y)}{\partial y}$$

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

Partial Derivatives



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



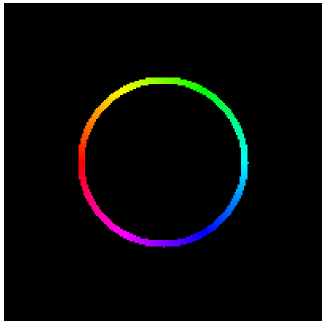
$$\frac{\partial f(x,y)}{\partial y}$$

Gradient magnitude

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

Gradient Orientation

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$ `atan2(dy,dx)`

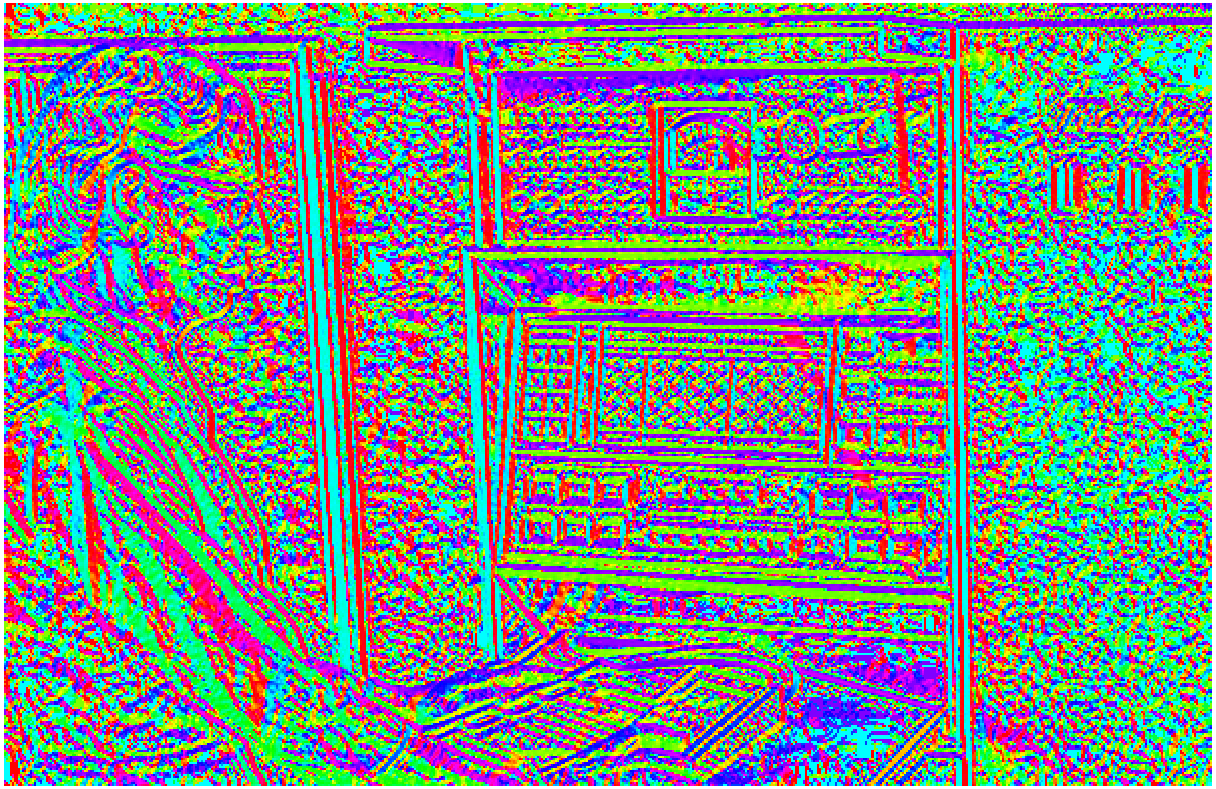


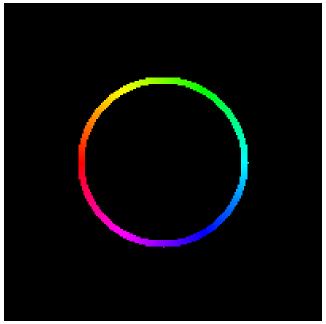
lightness is equal to gradient magnitude

Source: D. Fouhey

Image Gradient

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



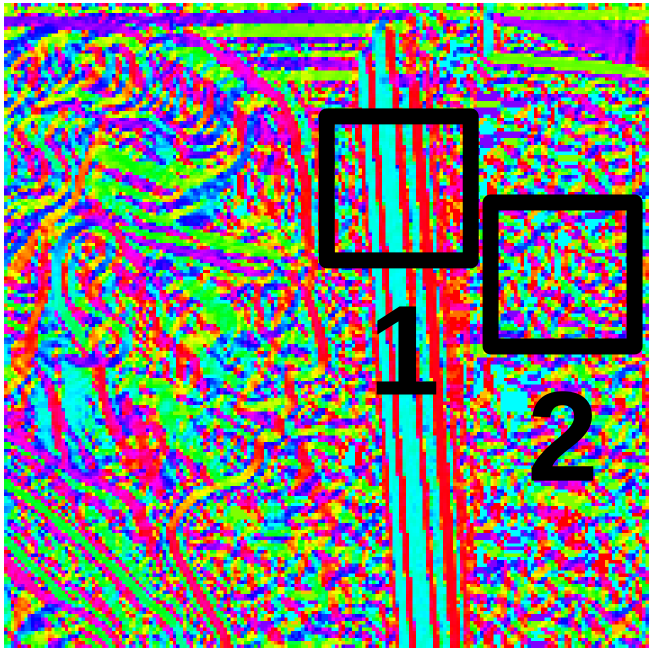


all the gradients

Source: D. Fouhey

Image Gradient

Why is there structure at 1 and not at 2?

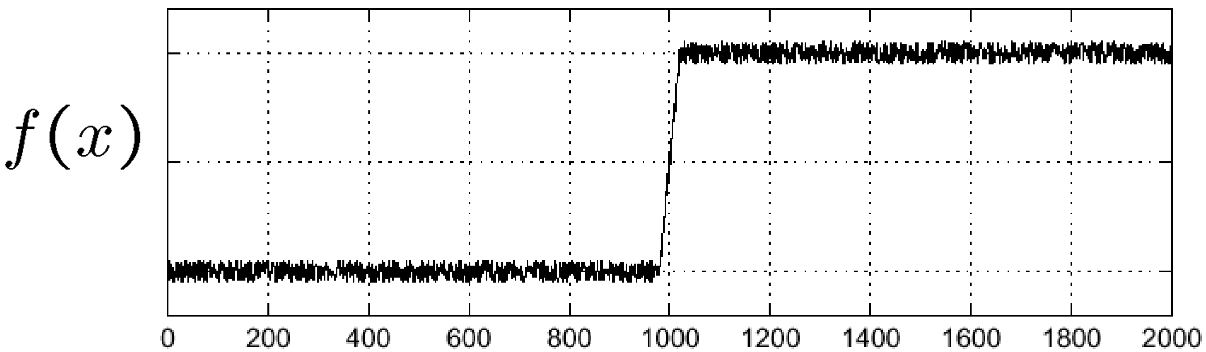


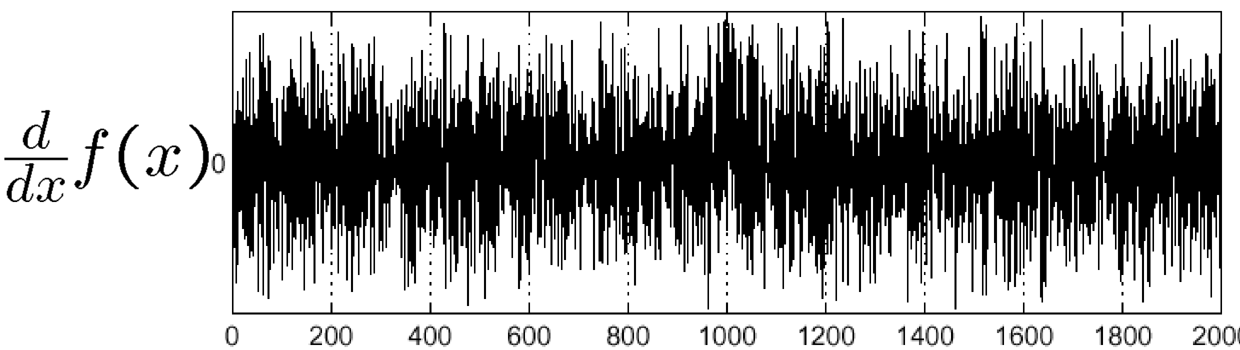
Source: D. Fouhey

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

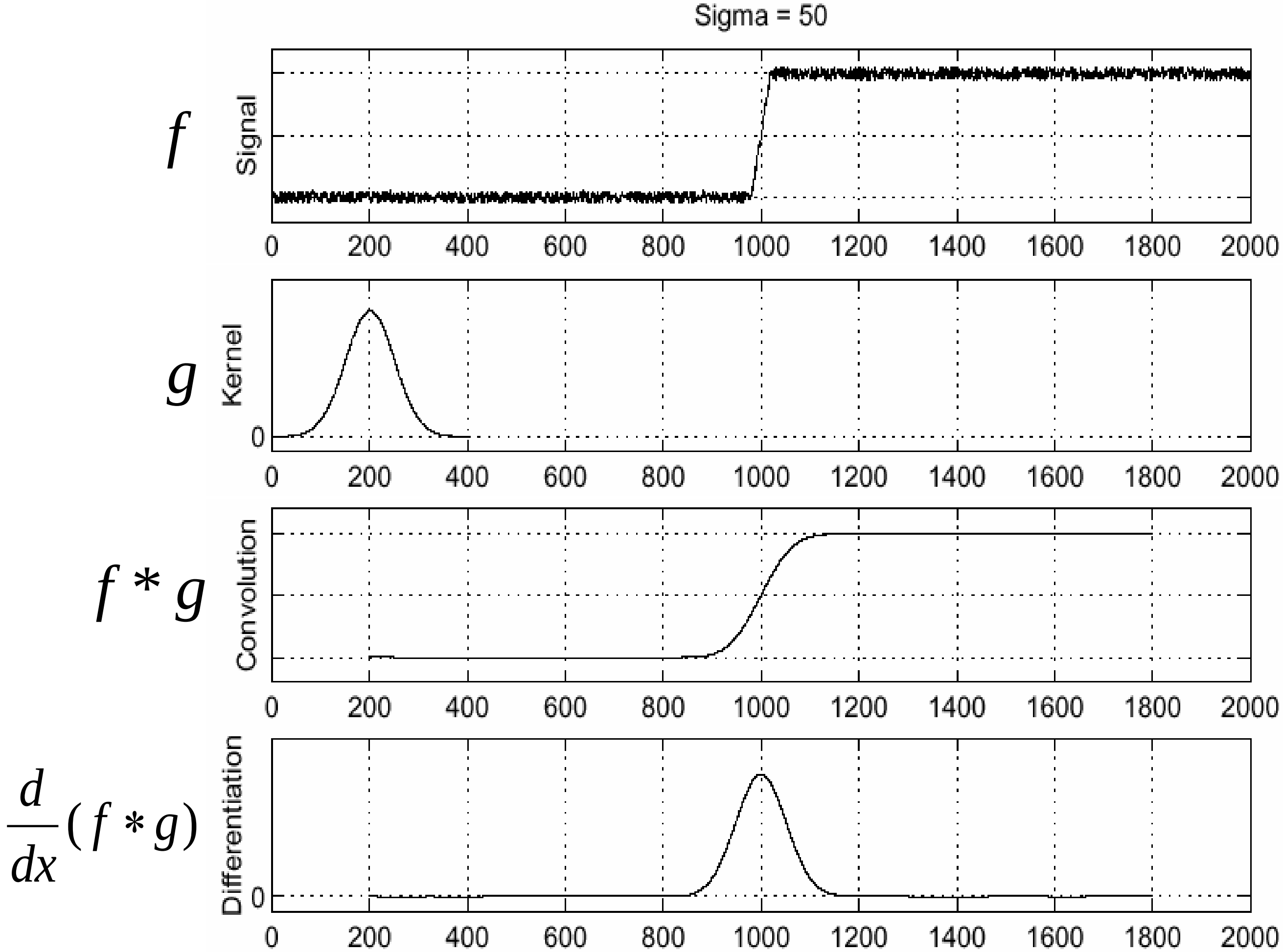




Where is the edge?

Source: S. Seitz

Solution: smooth first



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

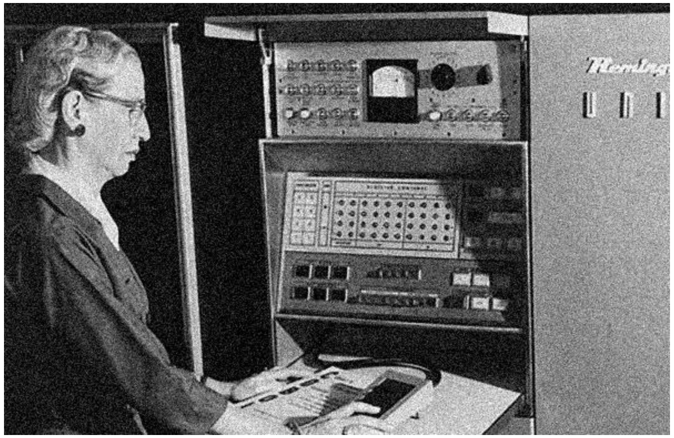
Source: S. Seitz

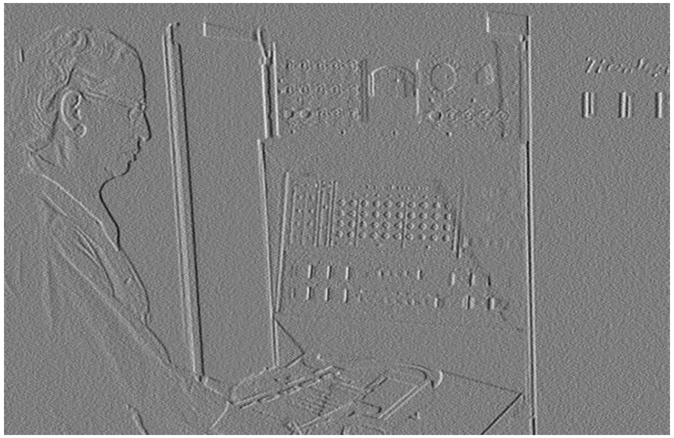
Noise in 2D

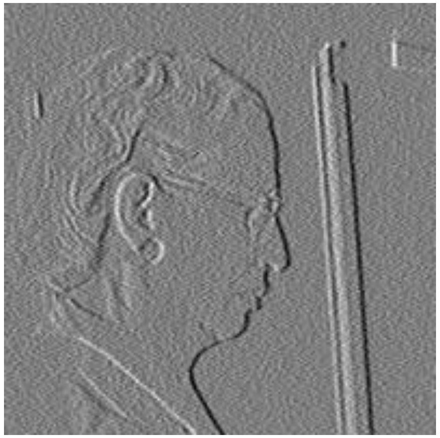
Noisy Input

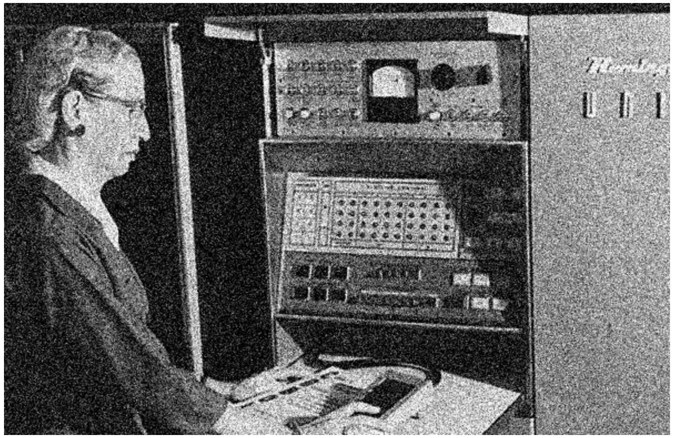
Ix via [-1,01]

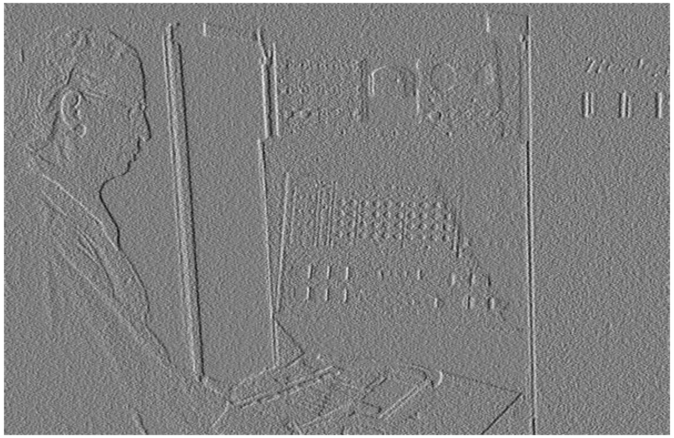
Zoom







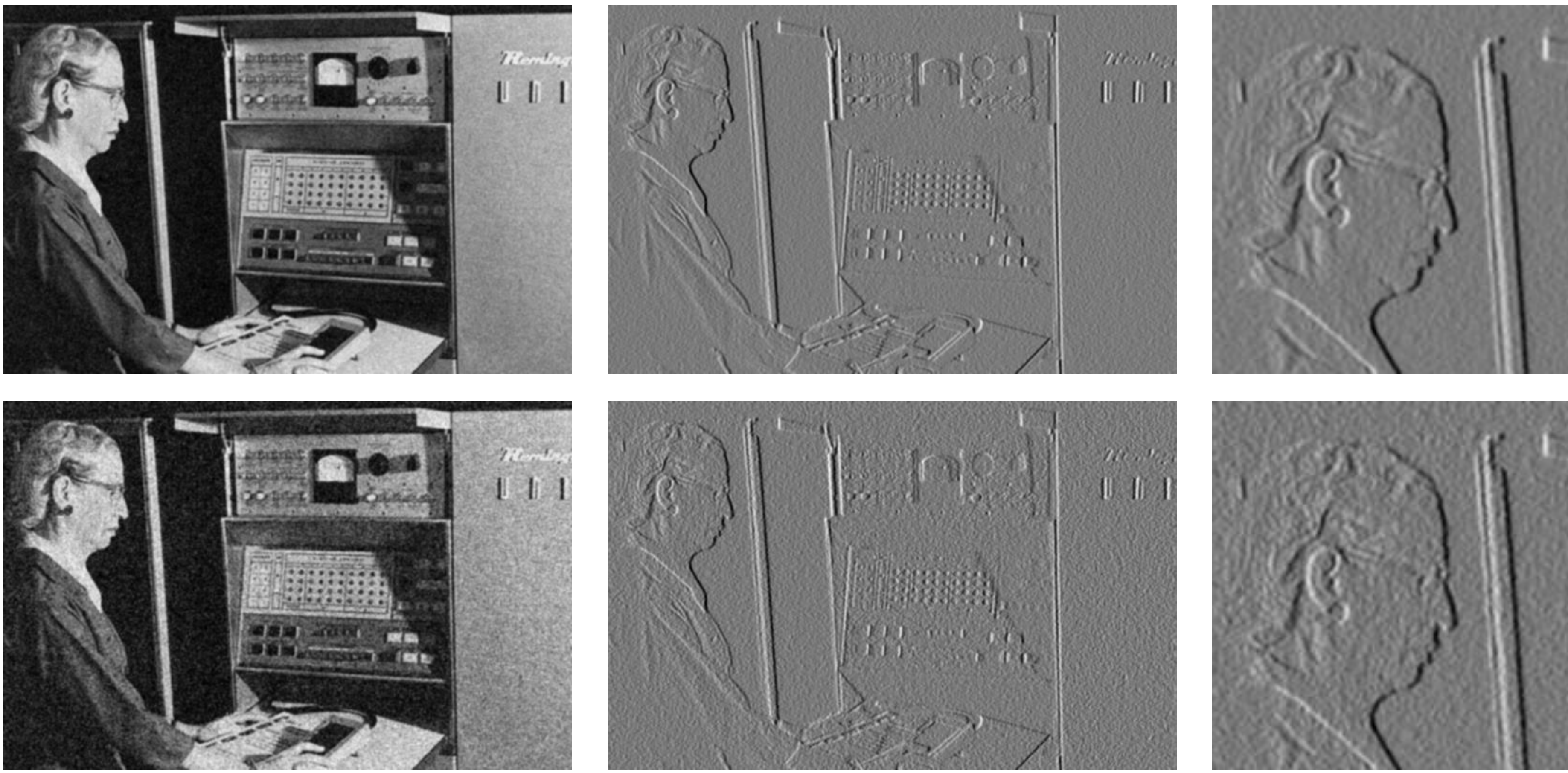




Source: D. Fouhey

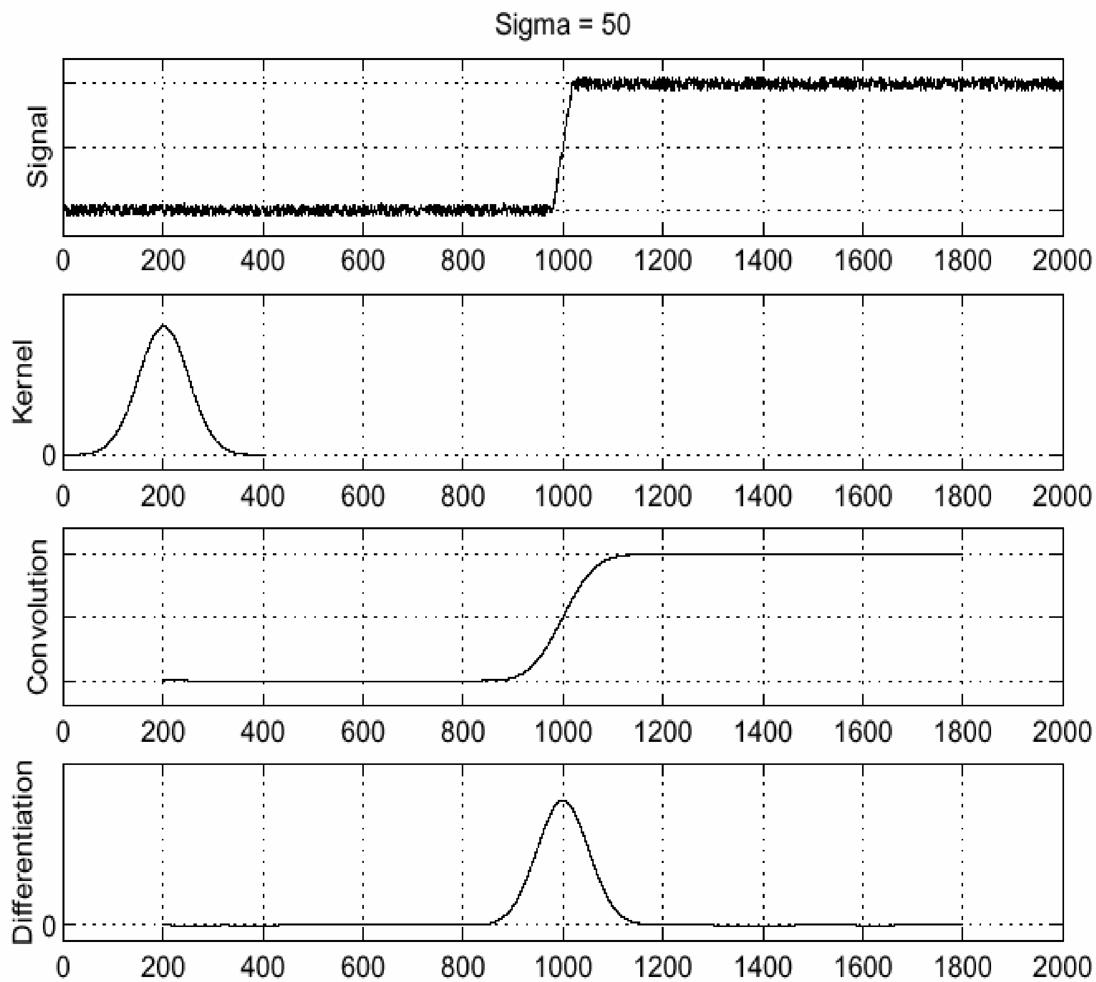
Noise + Smoothing

Smoothed Input | Ix via [-1,01] | Zoom



Source: D. Fouhey

How many convolutions here?



f

g

$f * g$

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

can we reduce this?

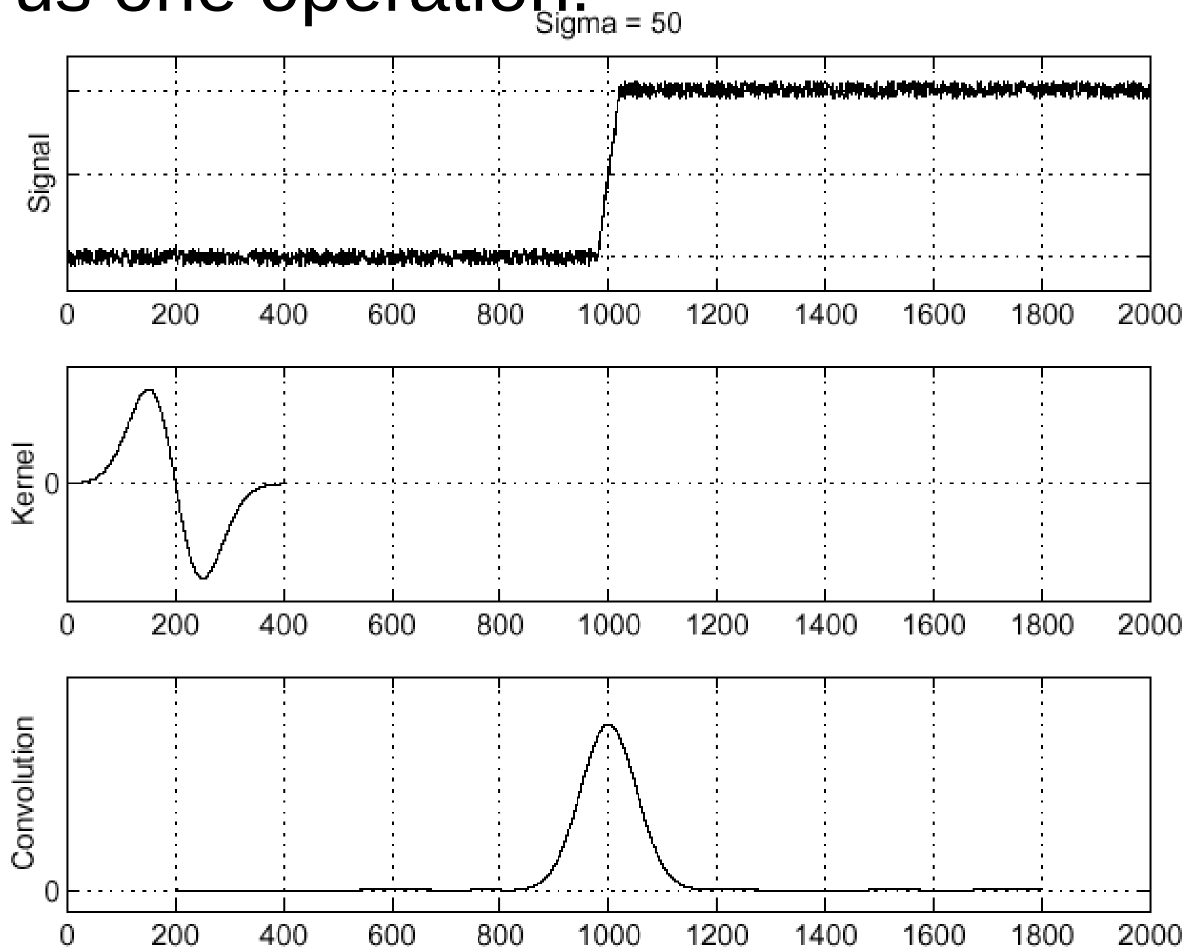
Derivative theorem of convolution

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

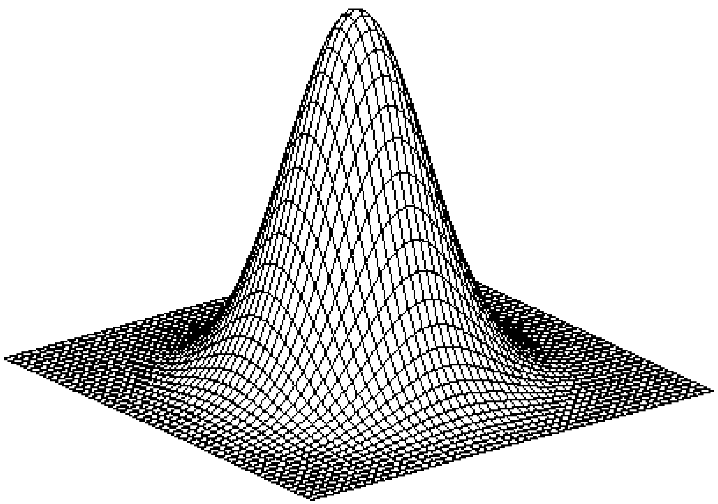
This saves us one operation:

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

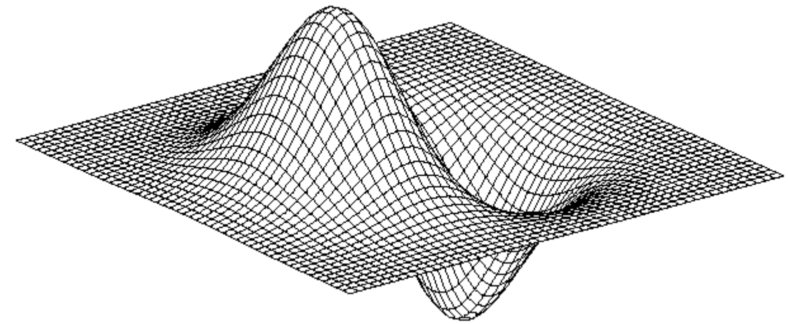
$$(\frac{\partial}{\partial x}h) \star f$$



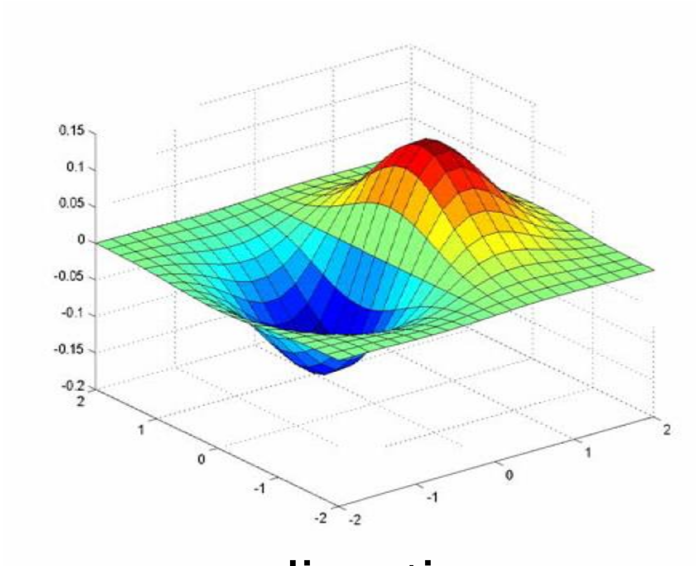
Derivative of Gaussian filter



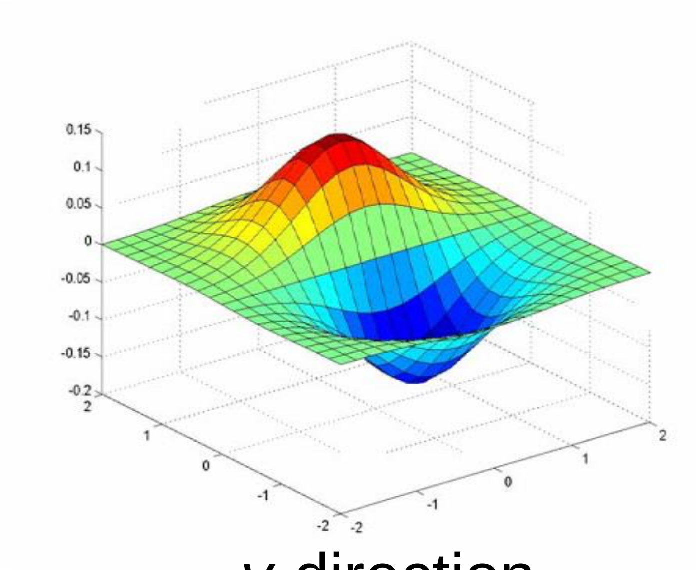
* [1 -1] =



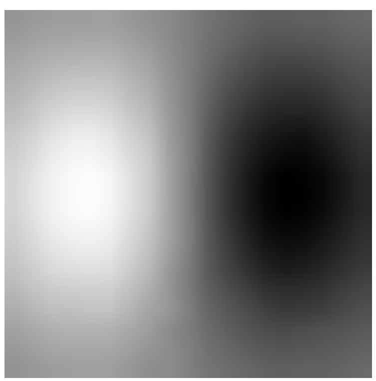
Derivative of Gaussian filter

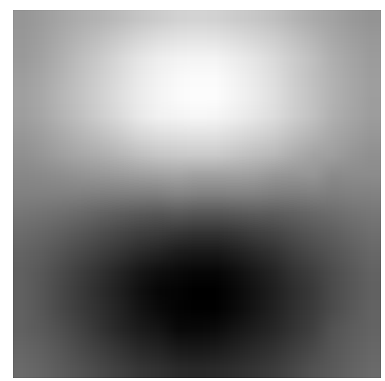


x-direction



y-direction





Which one finds horizontal/vertical edges?

Compare to classic derivative filters

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

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

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

Source: K. Grauman

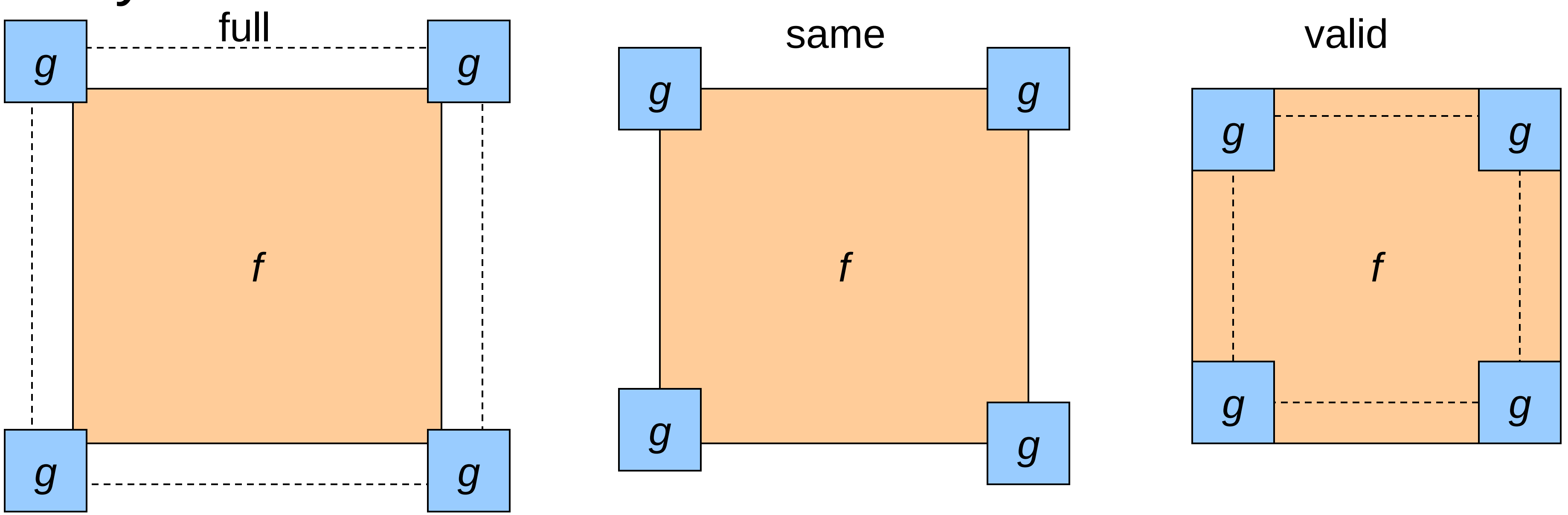
Filtering: practical matters

What is the size of the output?

(MATLAB) filter2(g, f, *shape*) or conv2(g,f,shape)

- *shape* = 'full': output size is sum of sizes of f and g
- *shape* = 'same': output size is same as f
- *shape* = 'valid': output size is difference of sizes of f and g

Pytorch conv2d 'valid' or 'same'

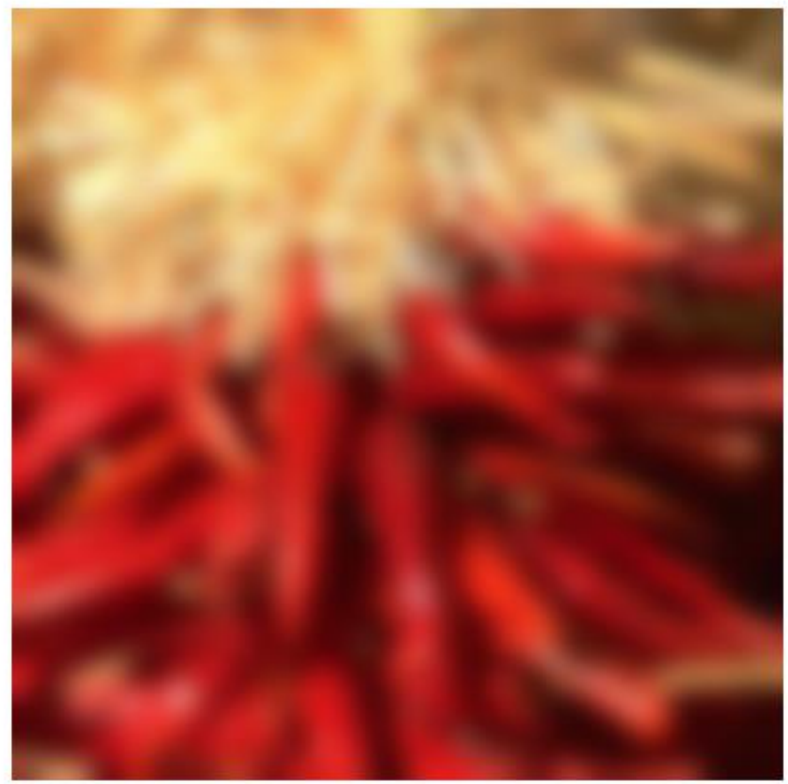


Source: S. Lazebnik

Practical matters

What about near the edge?

- the filter window falls off the edge of the image
- need to extrapolate
- methods:
  - clip filter (black)
  - wrap around (circular)
  - copy edge
  - reflect across edge



Source: S. Marschner