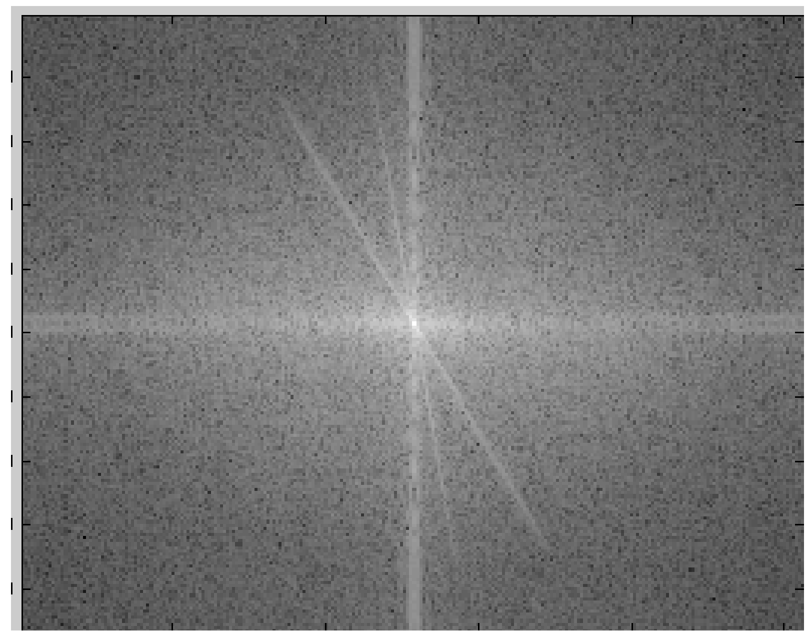
Convolution, Edge Detection, Sampling

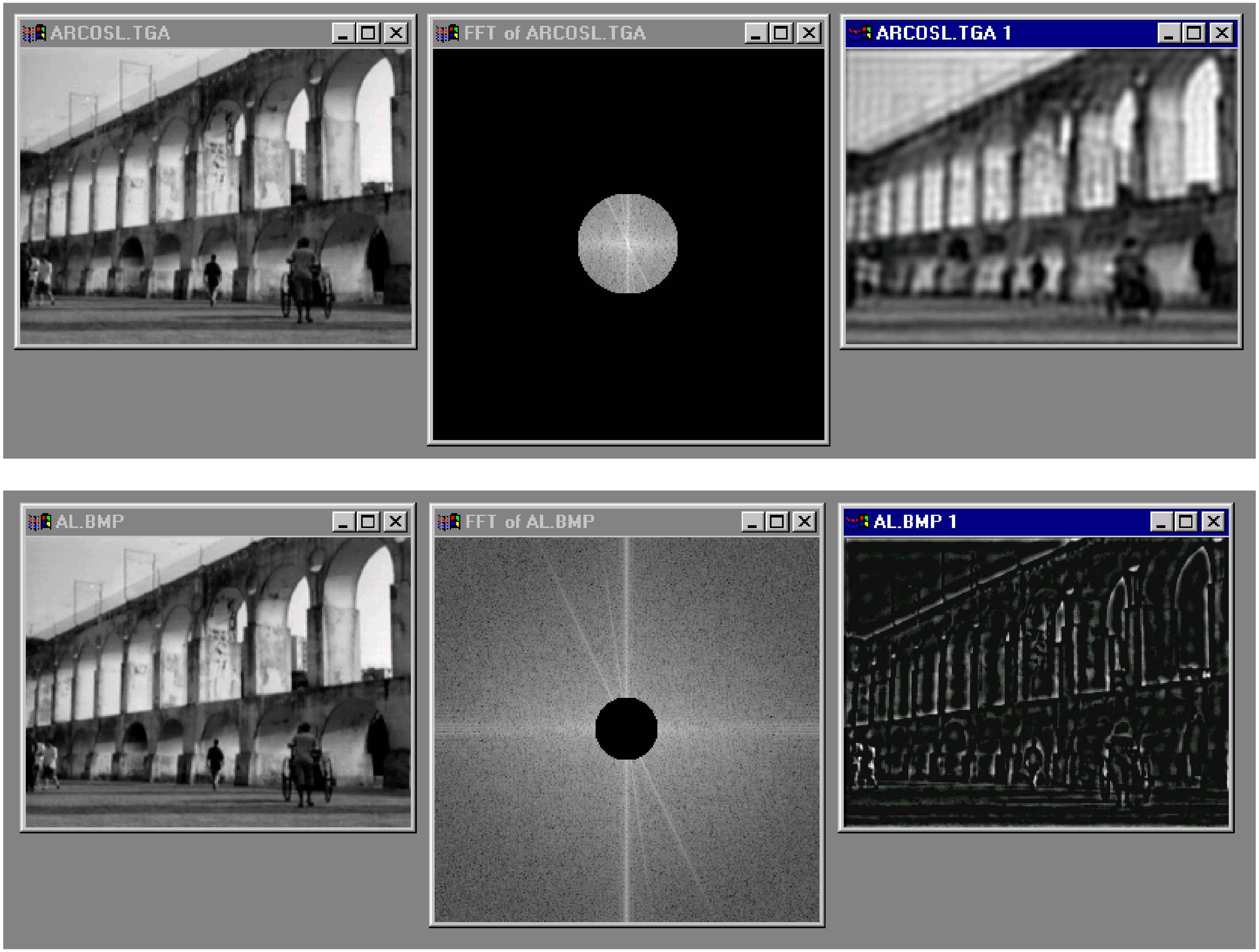
15-463: Computational Photography
Alexei Efros, CMU, Fall 2006

Some slides from Steve Seitz

Fourier spectrum



Fun and games with spectra



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

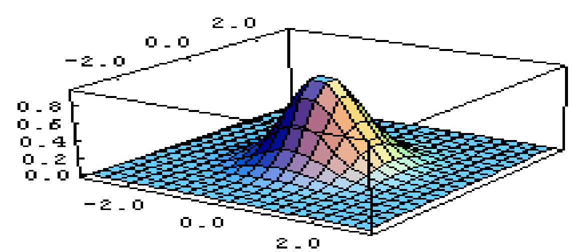
$F[x, y]$

$$\frac{1}{16}$$

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:

Convolution

Remember **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:

$$G = H \star F$$

Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

The Convolution Theorem

The greatest thing since sliced (banana) bread!

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\,\mathrm{F}[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

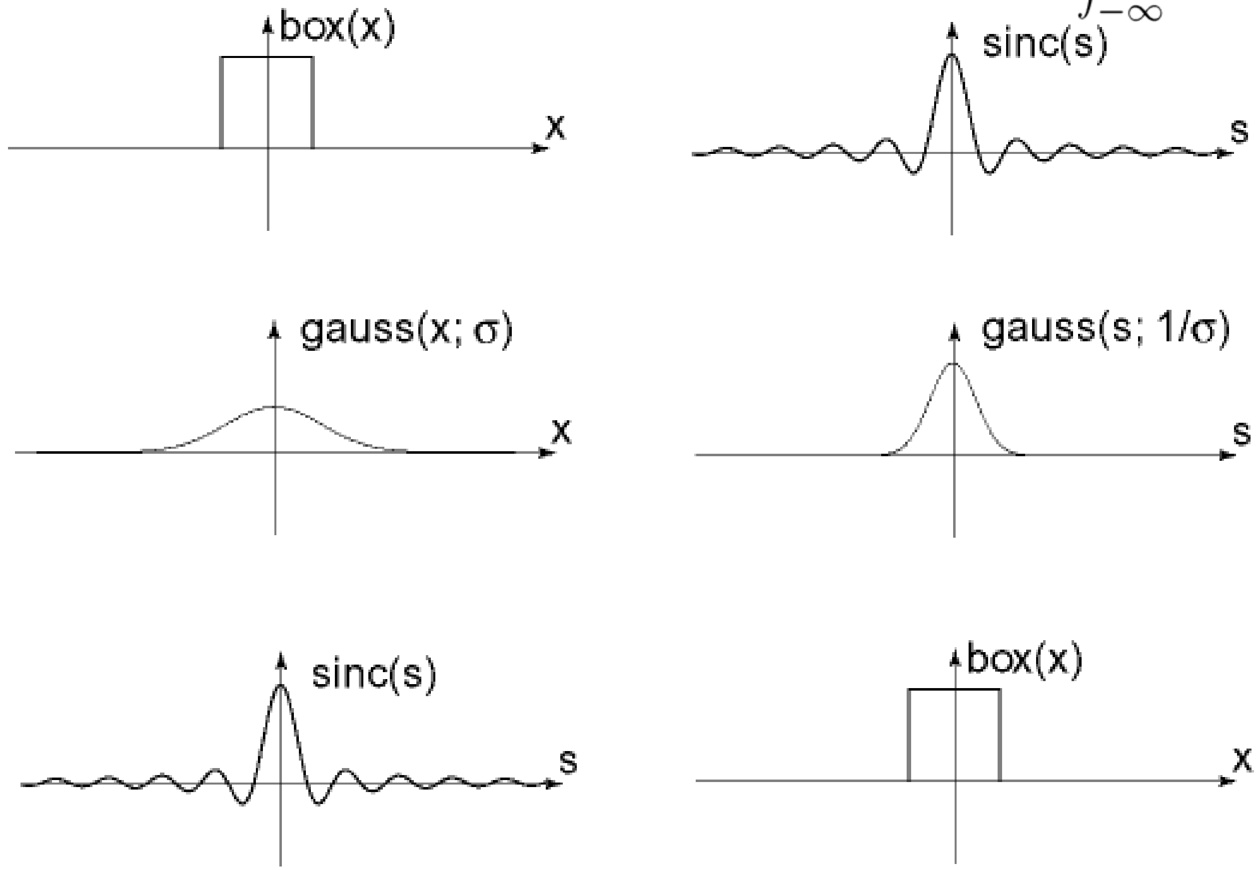
Fourier Transform pairs

Spatial domain

$f(x)$

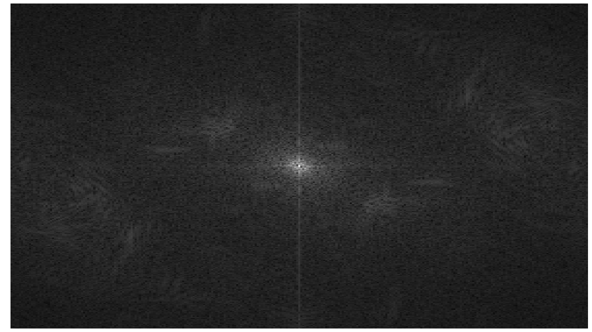
Frequency domain

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$



2D convolution theorem example

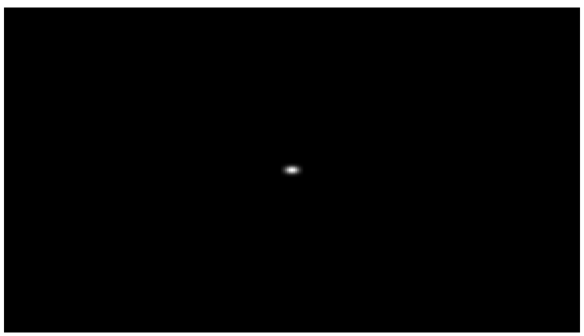
$f(x,y)$

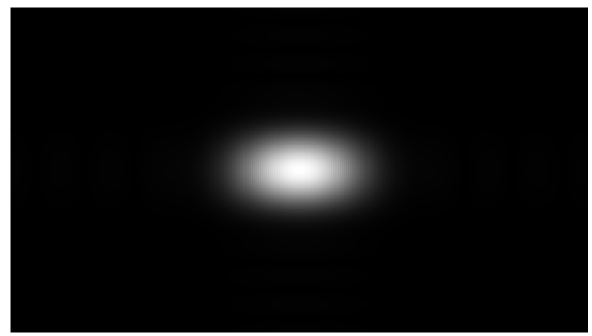


$|F(s_x,s_y)|$

$*$ $\times$

$h(x,y)$

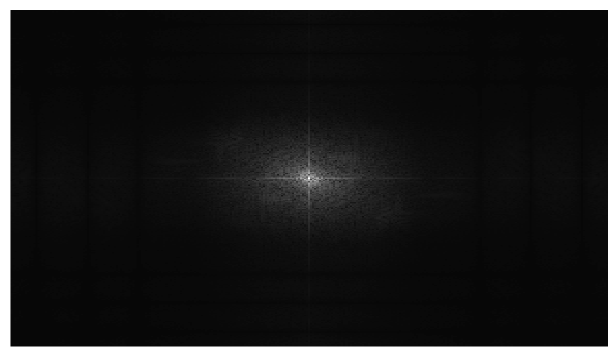




$|H(s_x,s_y)|$

$\Downarrow$ $\Downarrow$

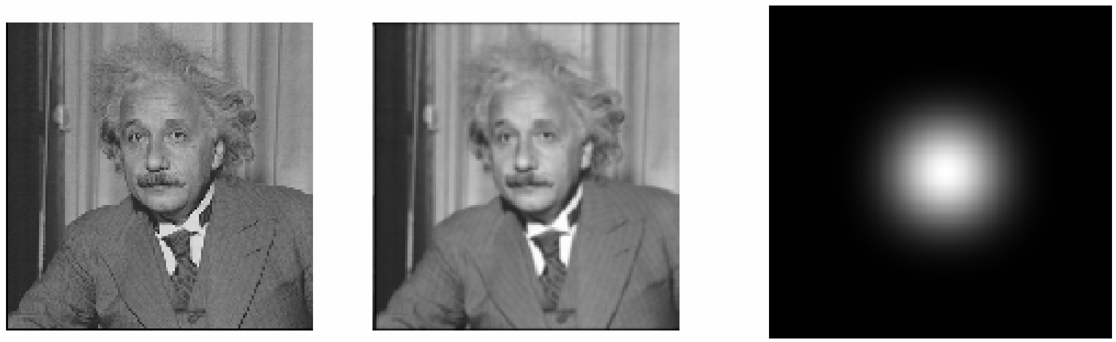
$g(x,y)$



$|G(s_x,s_y)|$

Low-pass, Band-pass, High-pass filters

low-pass:



band-pass:

what’s high-pass?

Edges in images

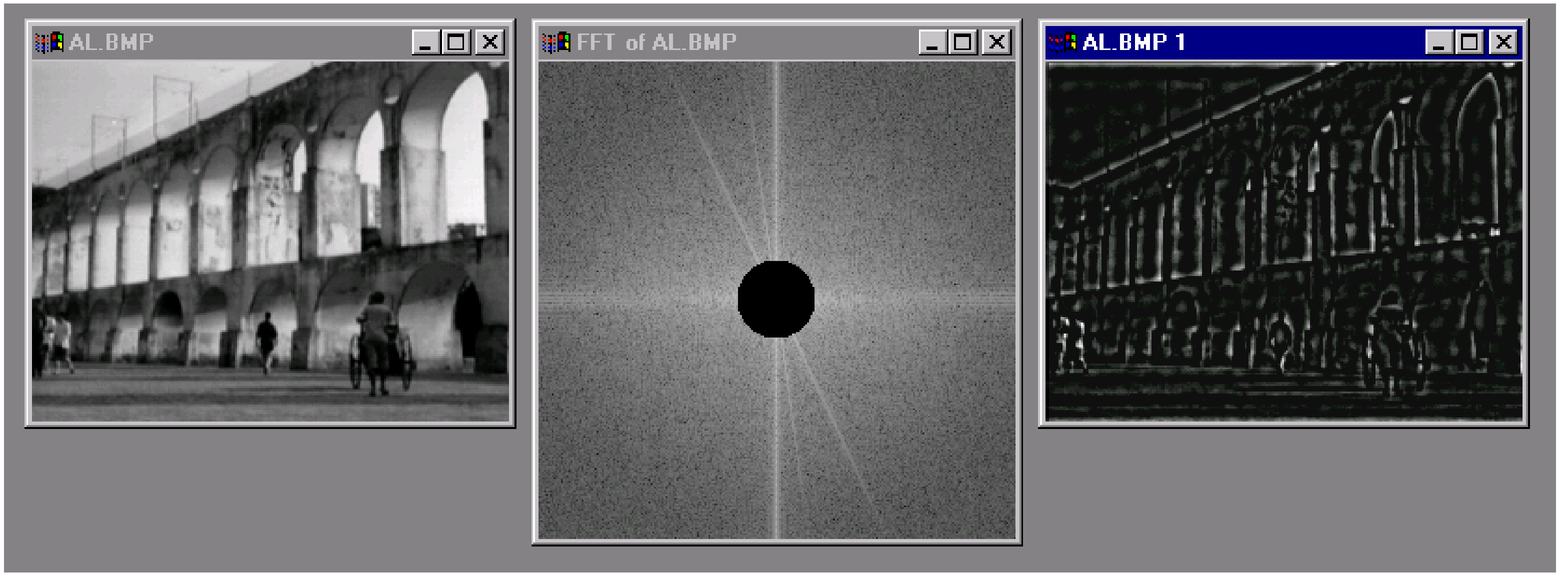
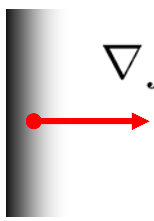


Image gradient

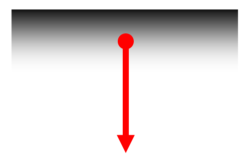
The gradient of an image:

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

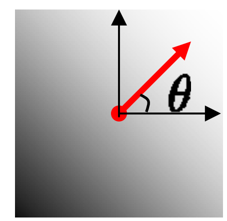
The gradient points in the direction of most rapid change in intensity



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



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



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

The 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

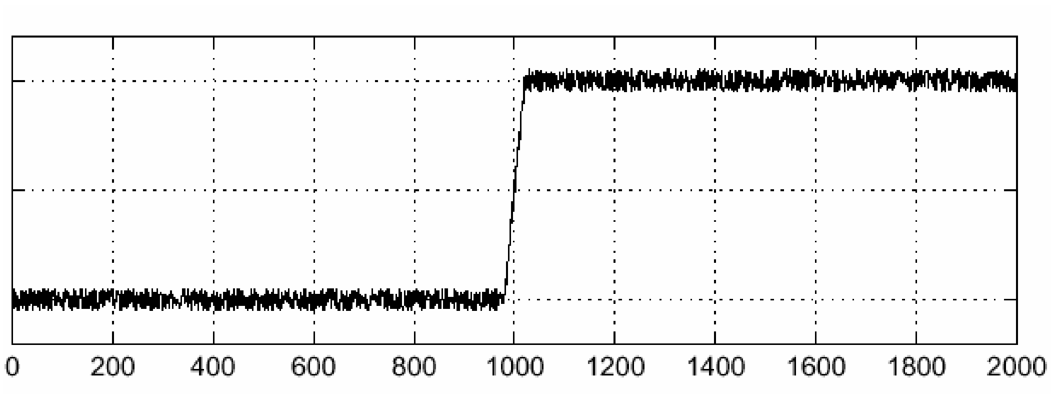
$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effects of noise

Consider a single row or column of the image

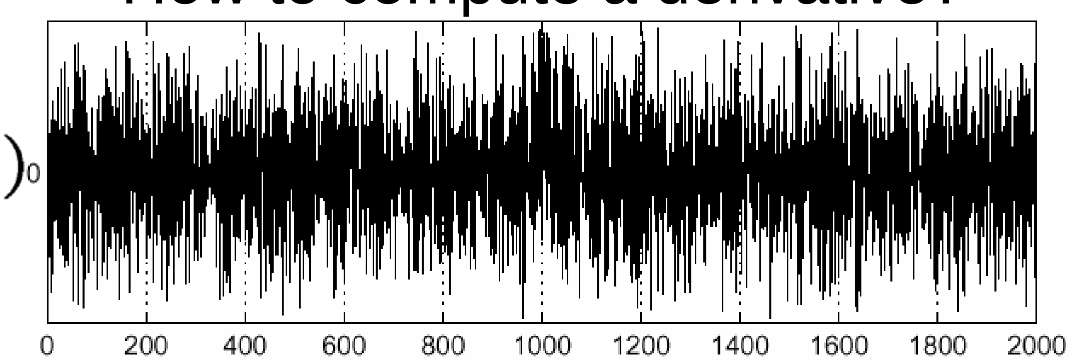
- Plotting intensity as a function of position gives a signal

$f(x)$



How to compute a derivative?

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



Where is the edge?

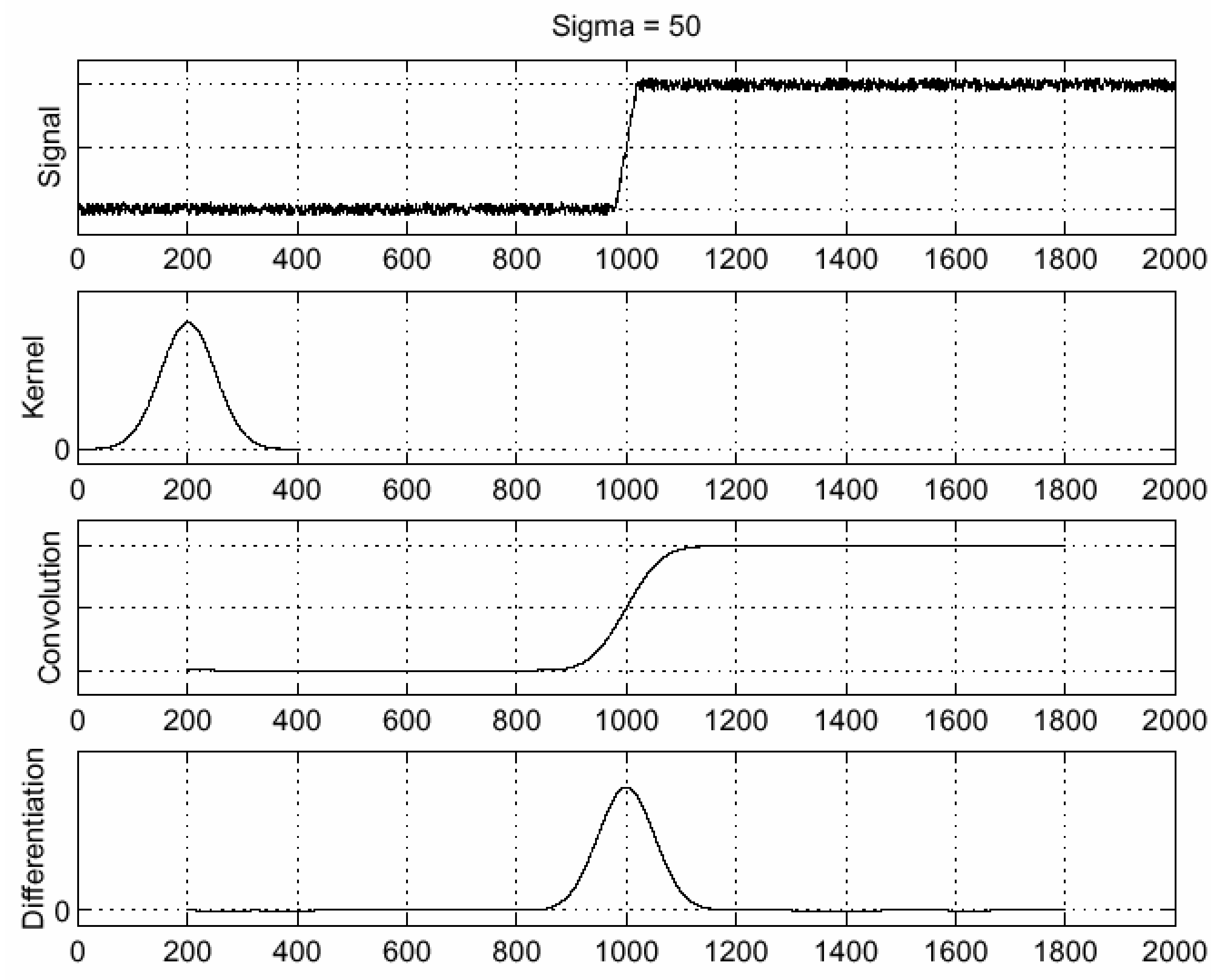
Solution: smooth first

f

h

$h \star f$

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



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

Derivative theorem of convolution

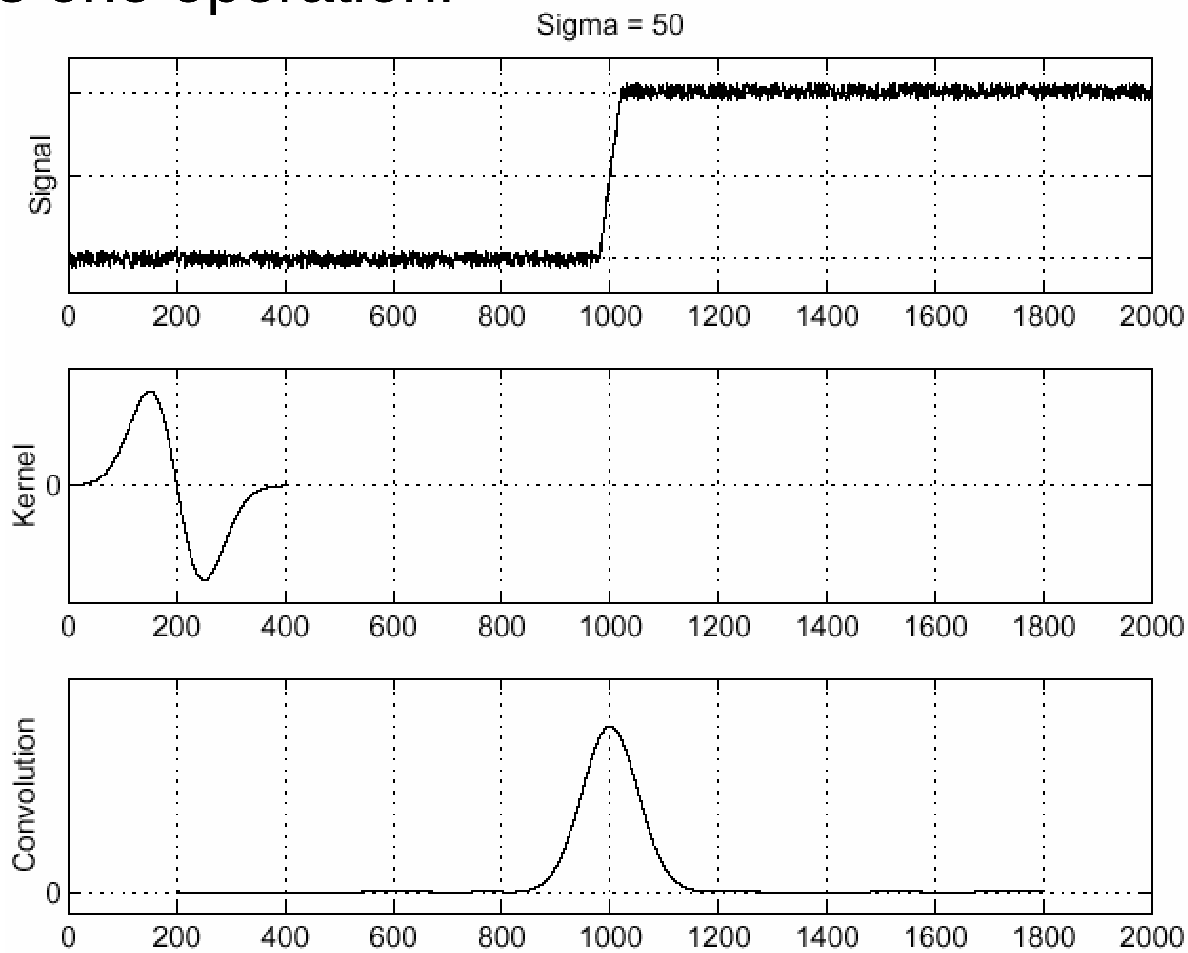
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

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

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



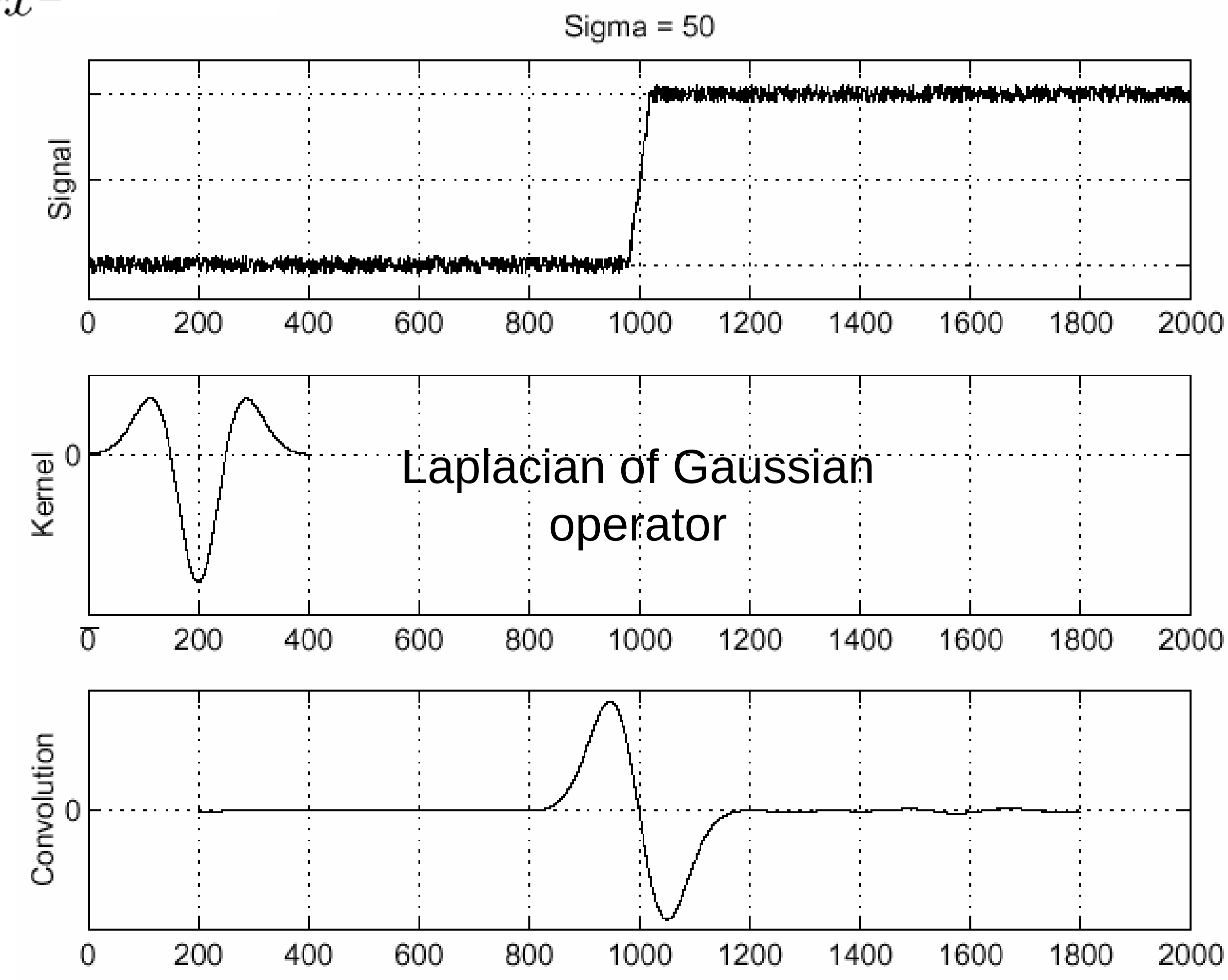
Laplacian of Gaussian

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

f

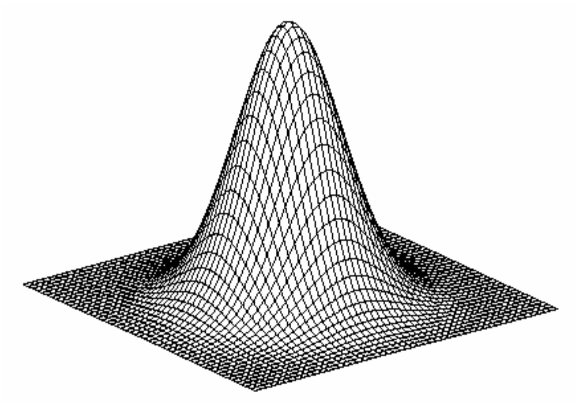
$\frac{\partial^2}{\partial x^2}h$

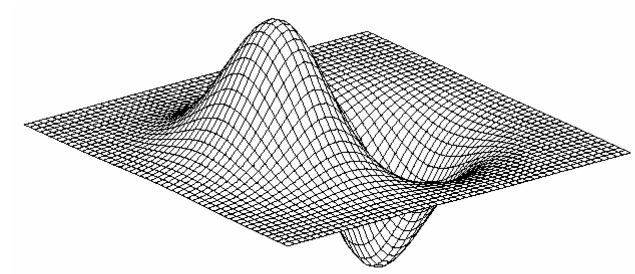
$(\frac{\partial^2}{\partial x^2}h) \star f$



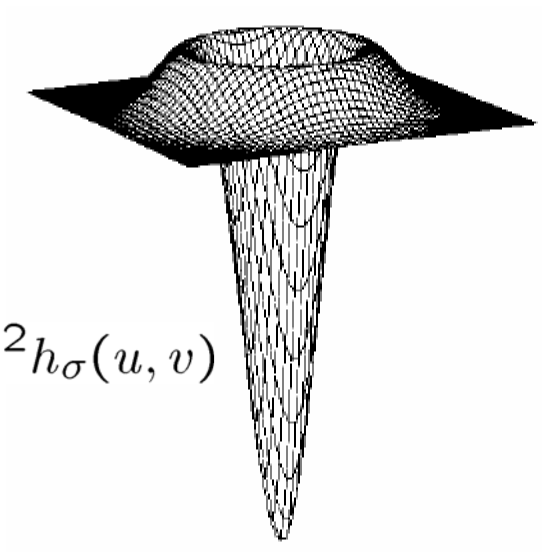
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters





Laplacian of Gaussian



Gaussian

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

derivative of Gaussian

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

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

∇^2 is the **Laplacian** operator:

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

MATLAB demo

```
g = fspecial('gaussian',15,2);
imagesc(g)
surfl(g)
gclown = conv2(clown,g,'same');
imagesc(conv2(clown,[-1 1],'same'));
imagesc(conv2(gclown,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
imagesc(conv2(clown,dx,'same'));
lg = fspecial('log',15,2);
lclown = conv2(clown,lg,'same');
imagesc(lclown)
imagesc(clown + .2*lclown)
```

Image Scaling

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

How to generate a half-sized version?

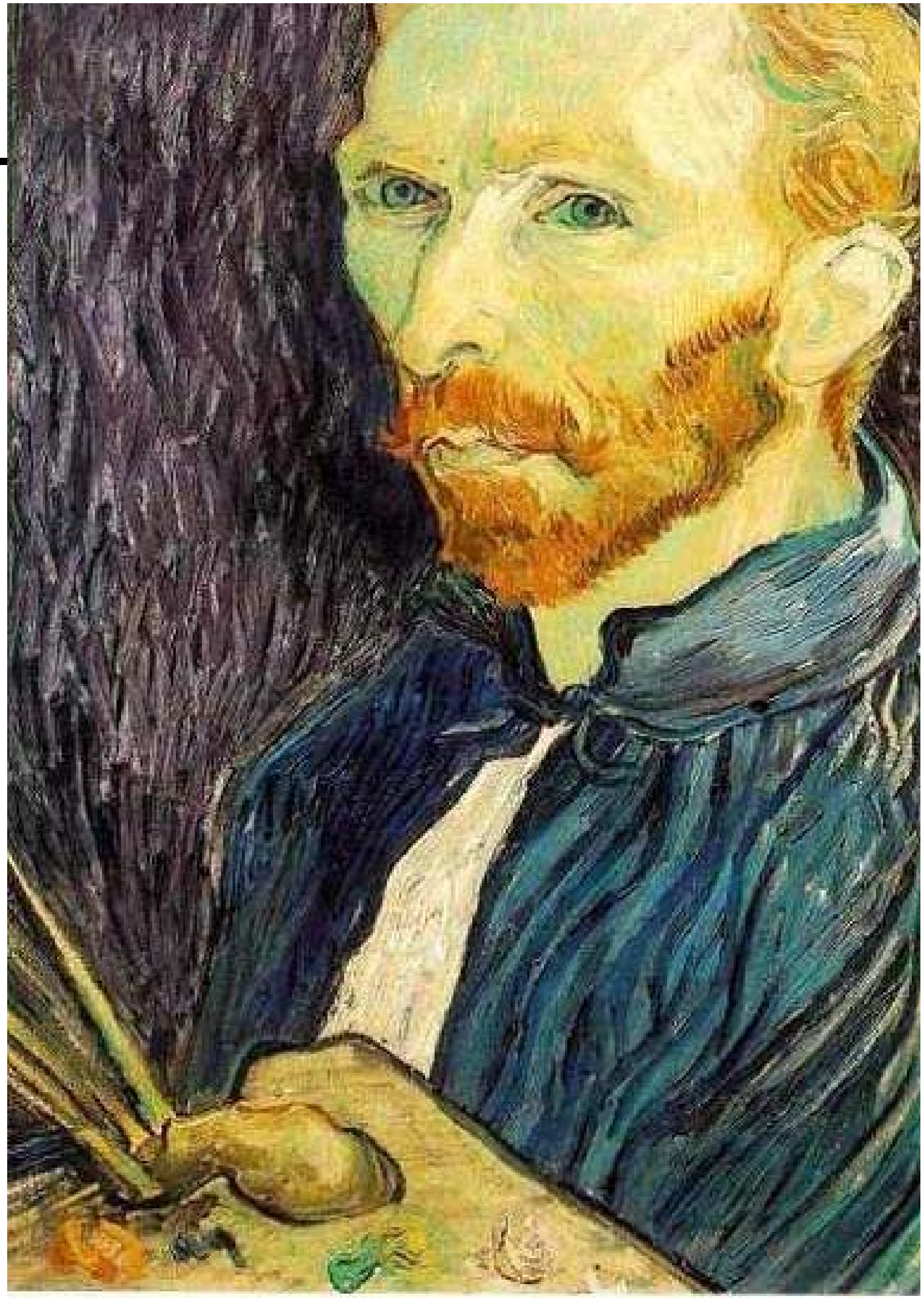
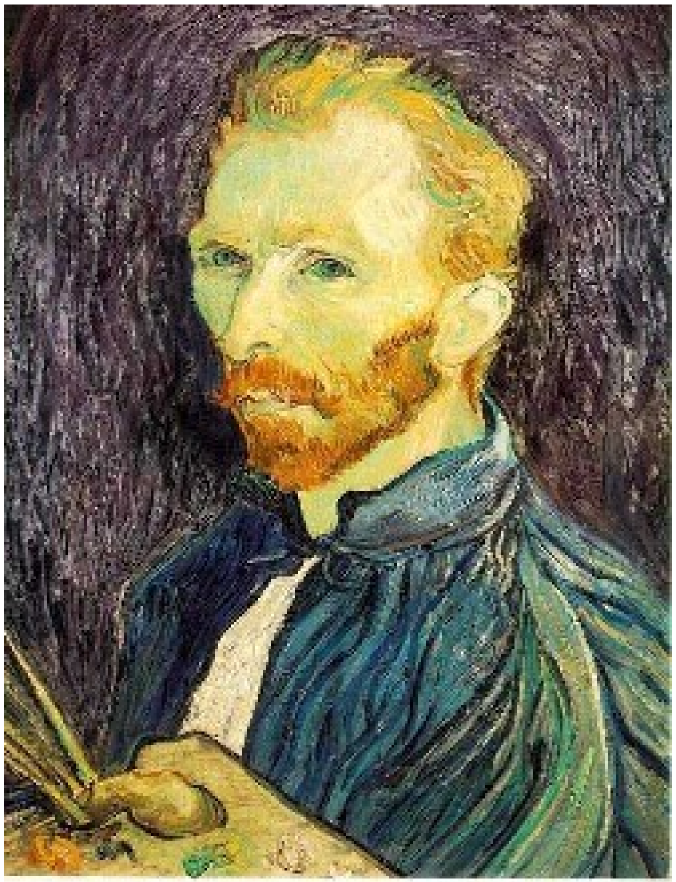
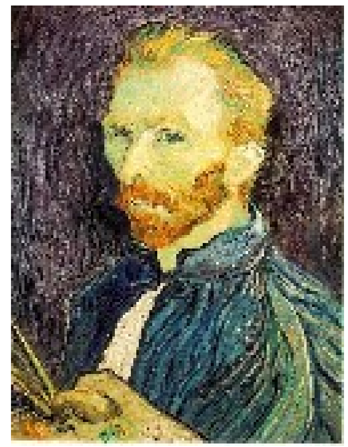


Image sub-sampling



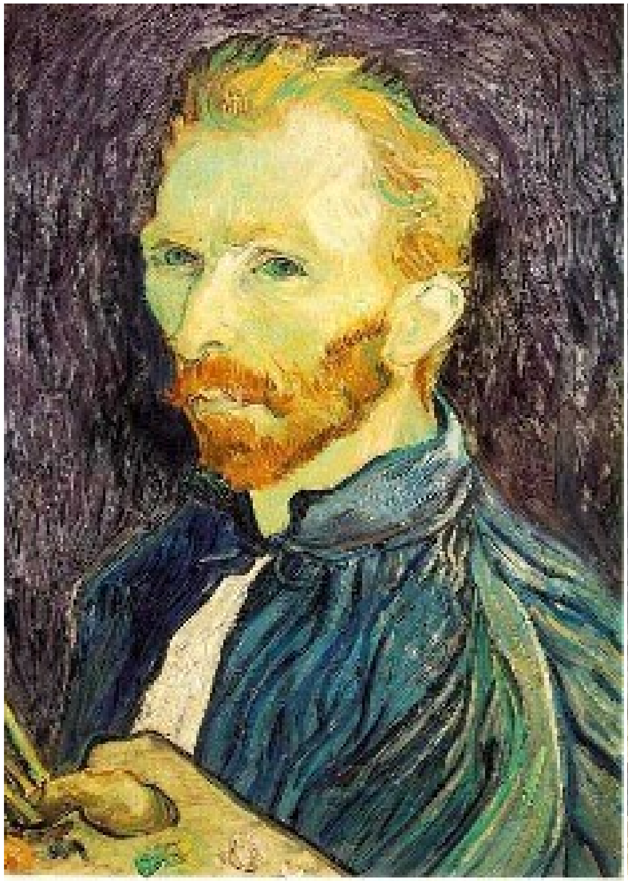


1/4

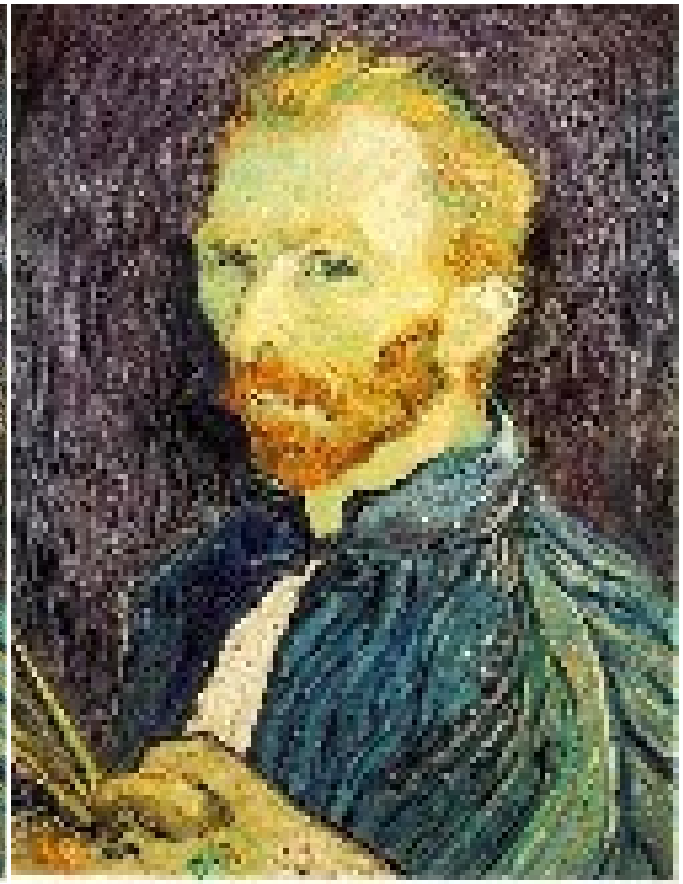
1/8

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

Image sub-sampling



1/2

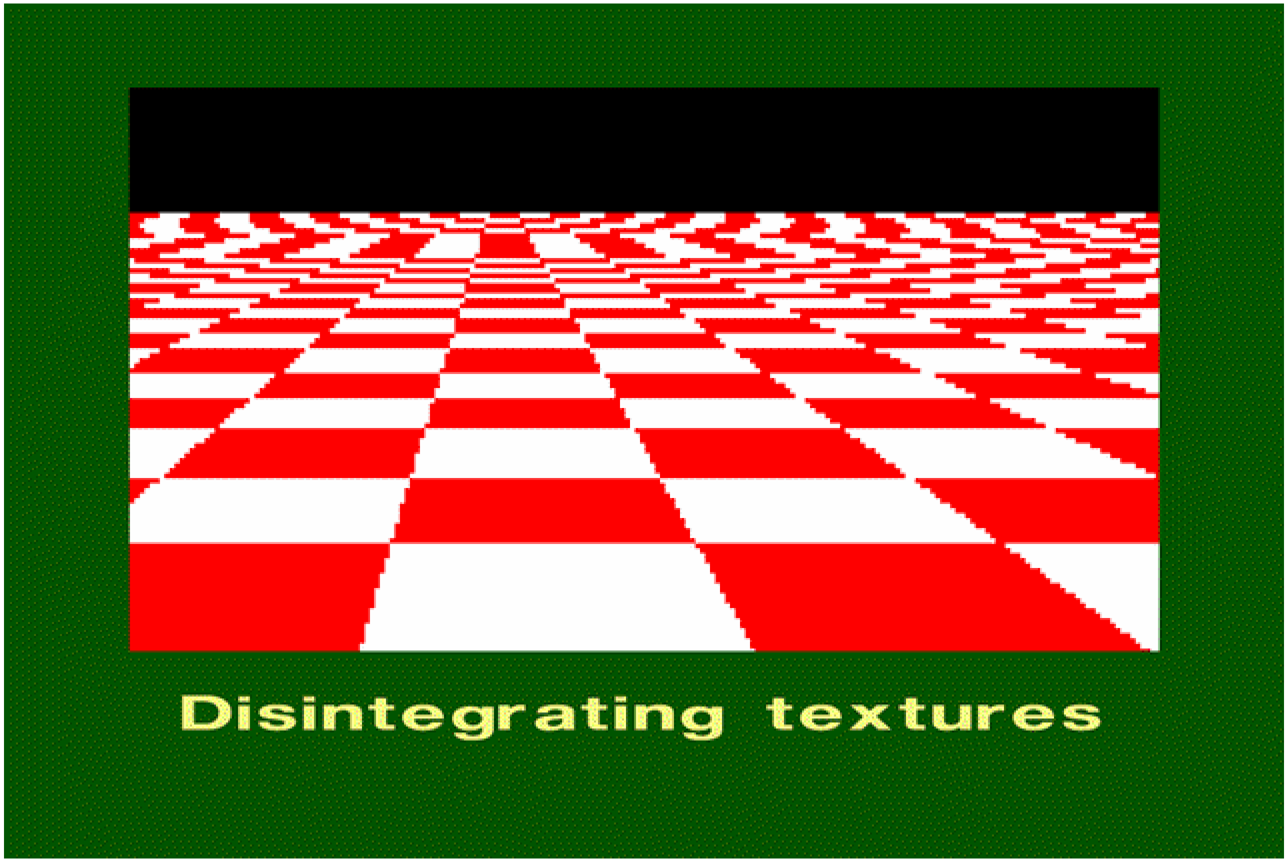


1/4 (2x zoom)

1/8 (4x zoom)

Why does this look so crufty?

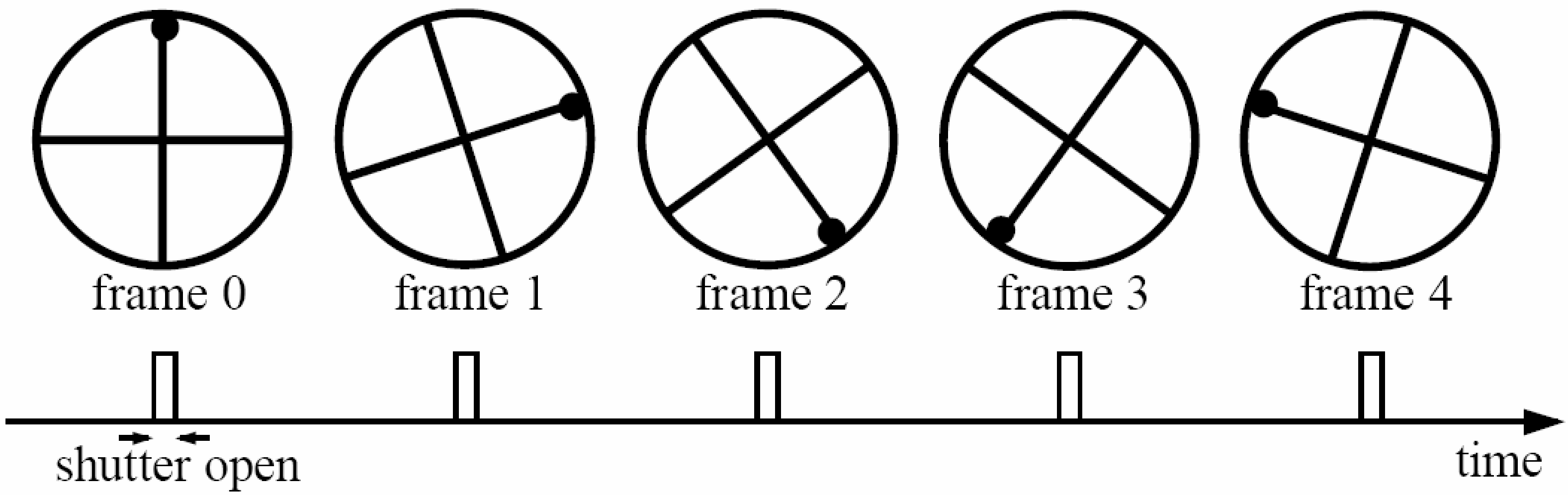
Even worse for synthetic images



Really bad in video

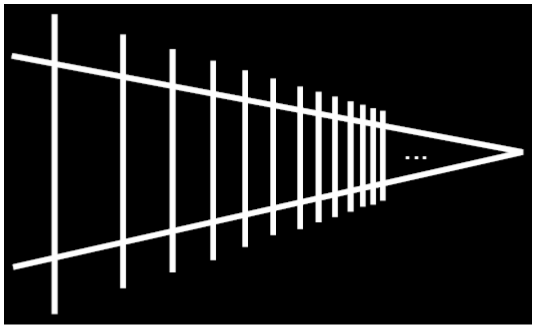
Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

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)

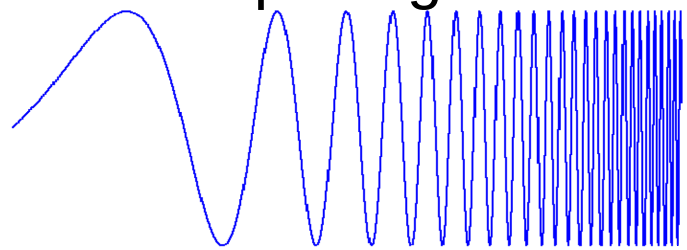
Alias: n., an assumed name



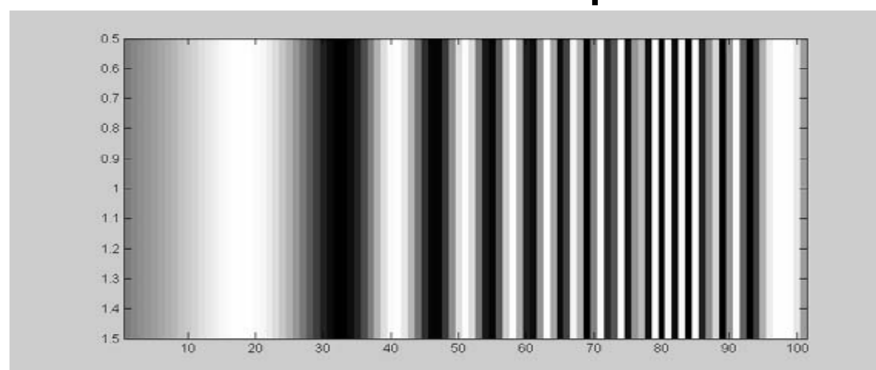
Picket fence receding
Into the distance will
produce aliasing…

WHY?

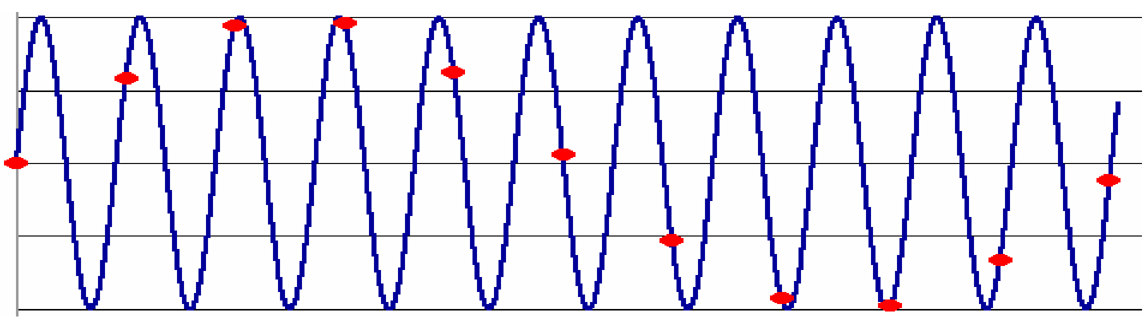
Input signal:



Matlab output:



x = 0:.05:5; imagesc(sin((2.^x).*x))



Aj-aj-aj:
Alias!

Not enough samples

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*

Where can it happen in images?

- During image synthesis:
  - **sampling** continous singal into discrete signal
  - e.g. ray tracing, line drawing, function plotting, etc.
- During image processing:
  - **resampling** discrete signal at a different rate
  - e.g. Image warping, zooming in, zooming out, etc.

To do sampling right, need to understand the structure of your signal/image

Enter Monsieur Fourier…

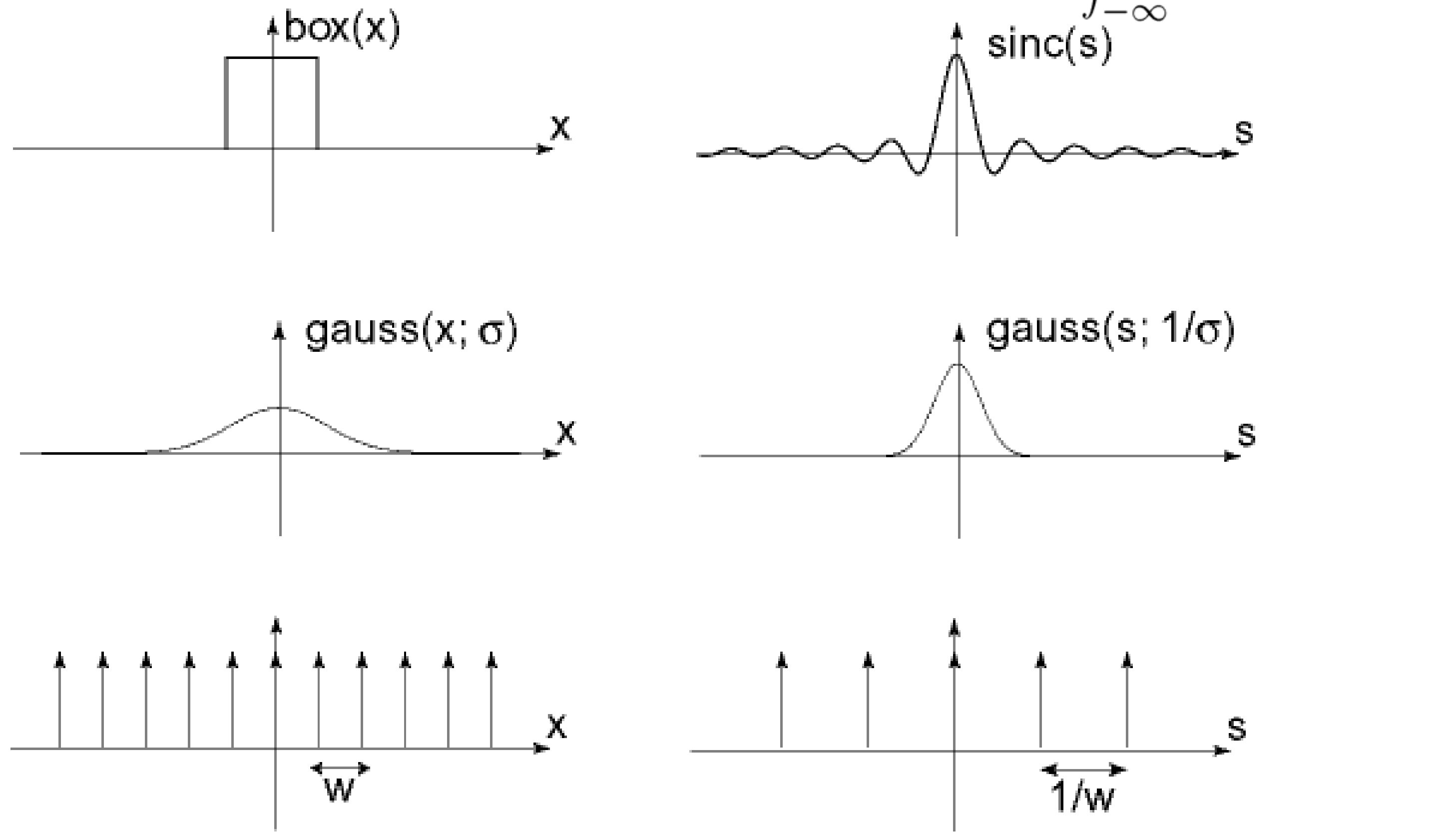
Fourier transform pairs

Spatial domain

$f(x)$

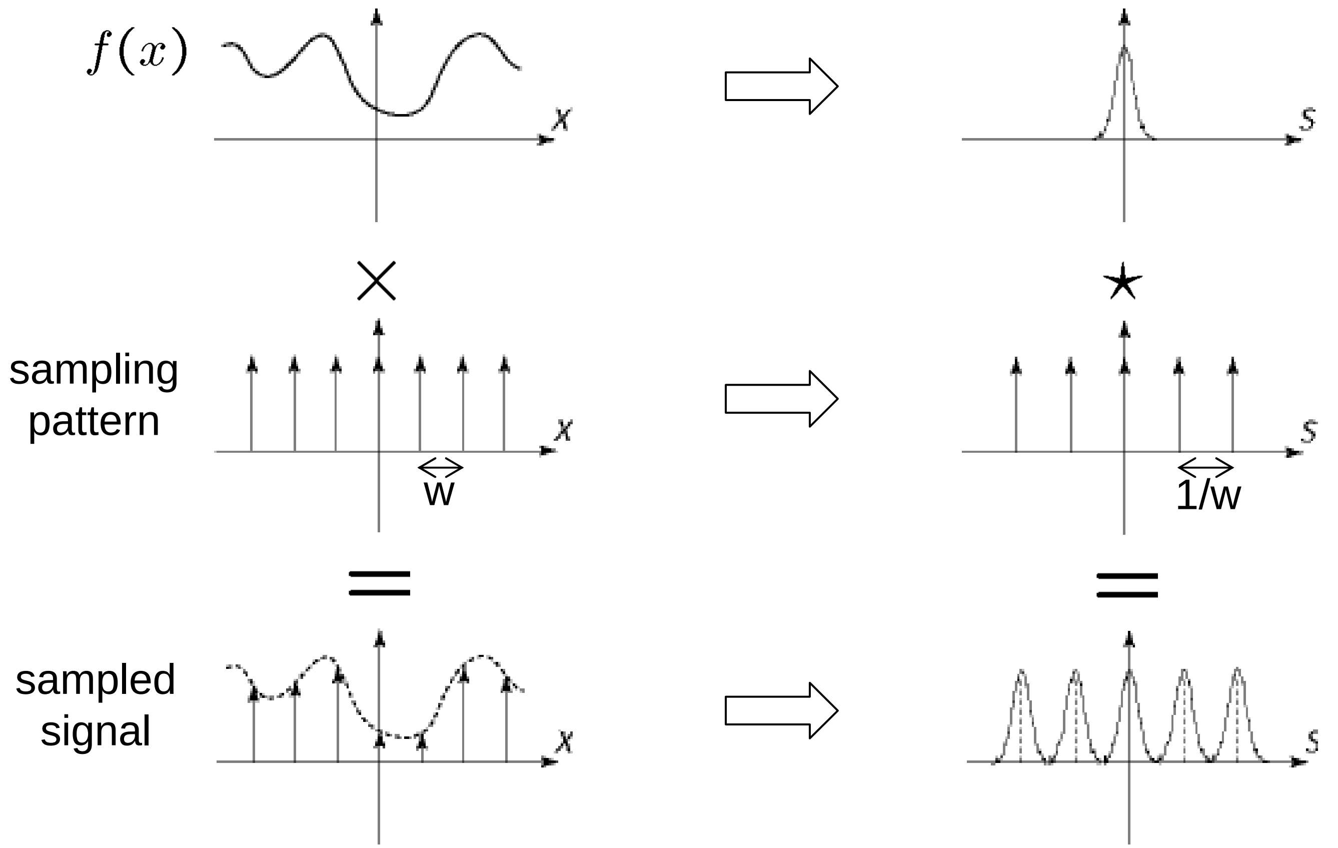
Frequency domain

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$



Sampling

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$

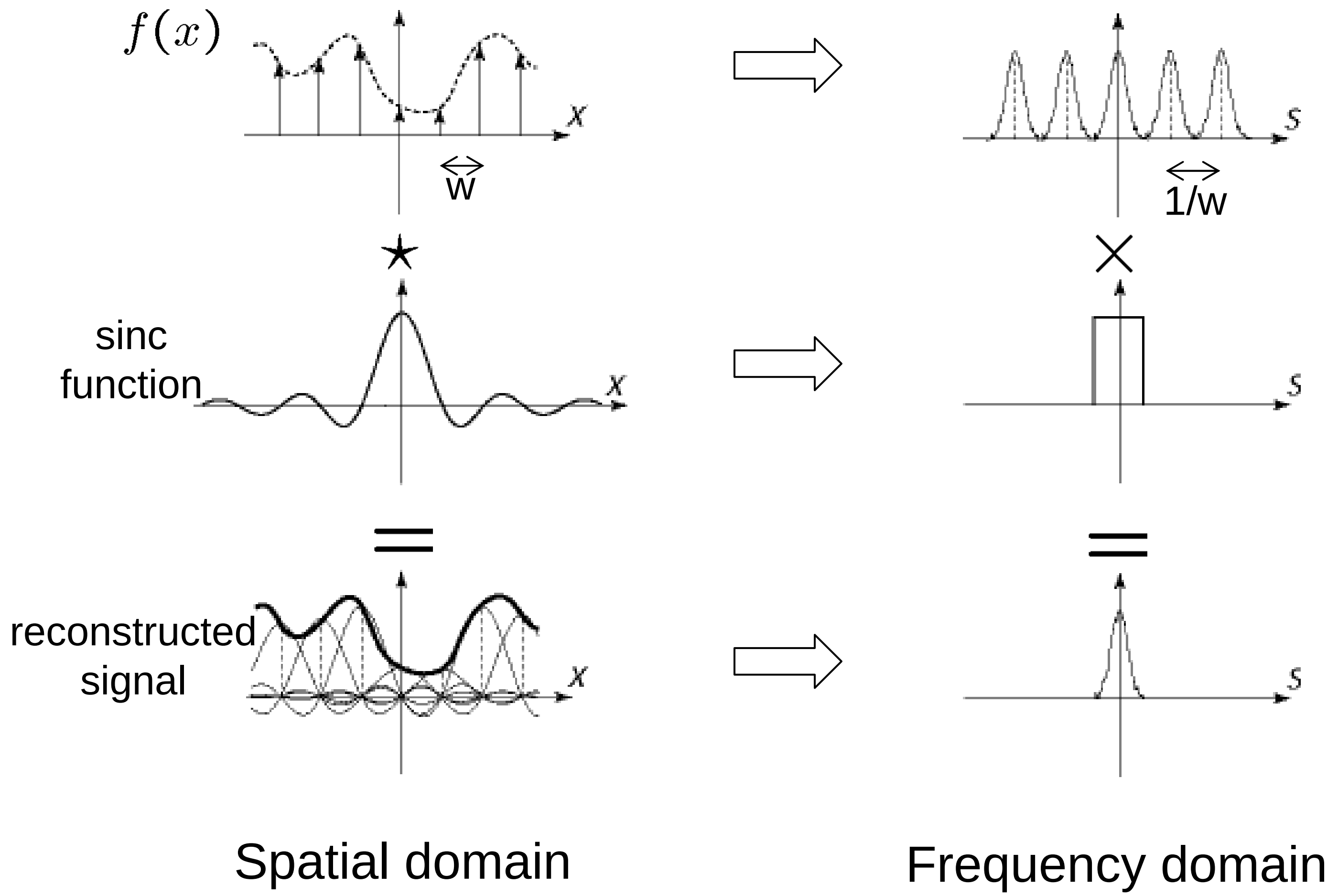


Spatial domain

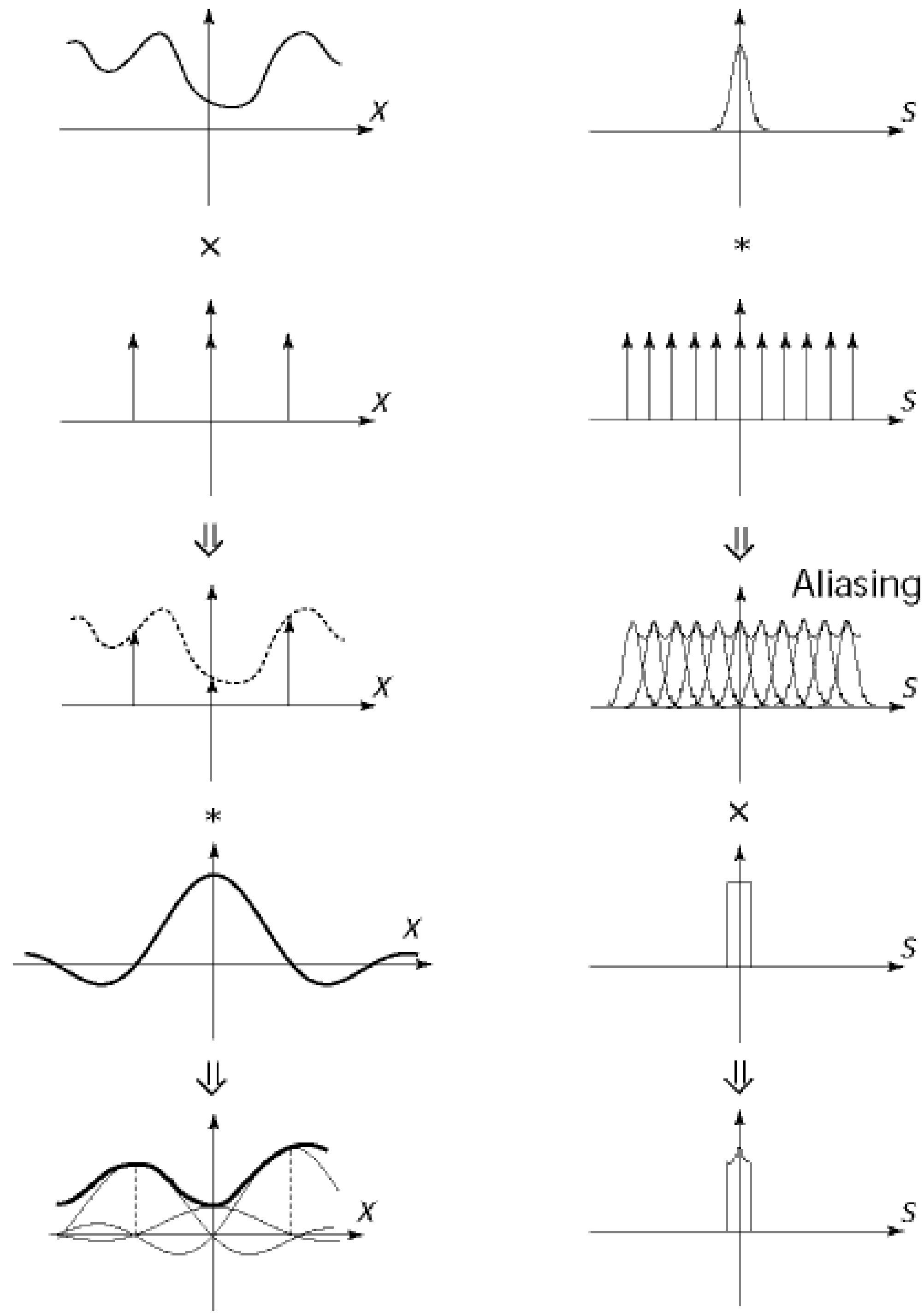
Frequency domain

Reconstruction

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$

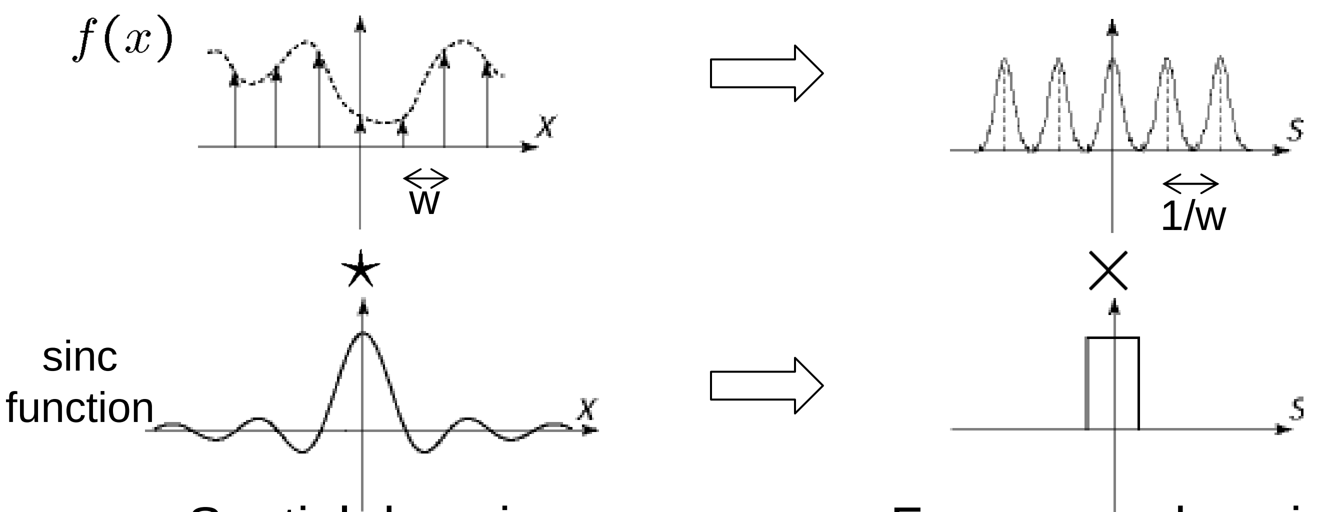


What happens when the sampling rate is too low?



Nyquist Rate

What's the minimum Sampling Rate *1/w* to get rid of overlaps?



Spatial domain

Frequency domain

Sampling Rate ≥ 2 * max frequency in the image

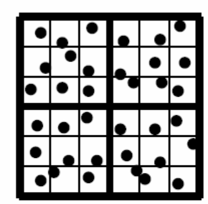
- this is known as the Nyquist Rate

Antialiasing

What can be done?

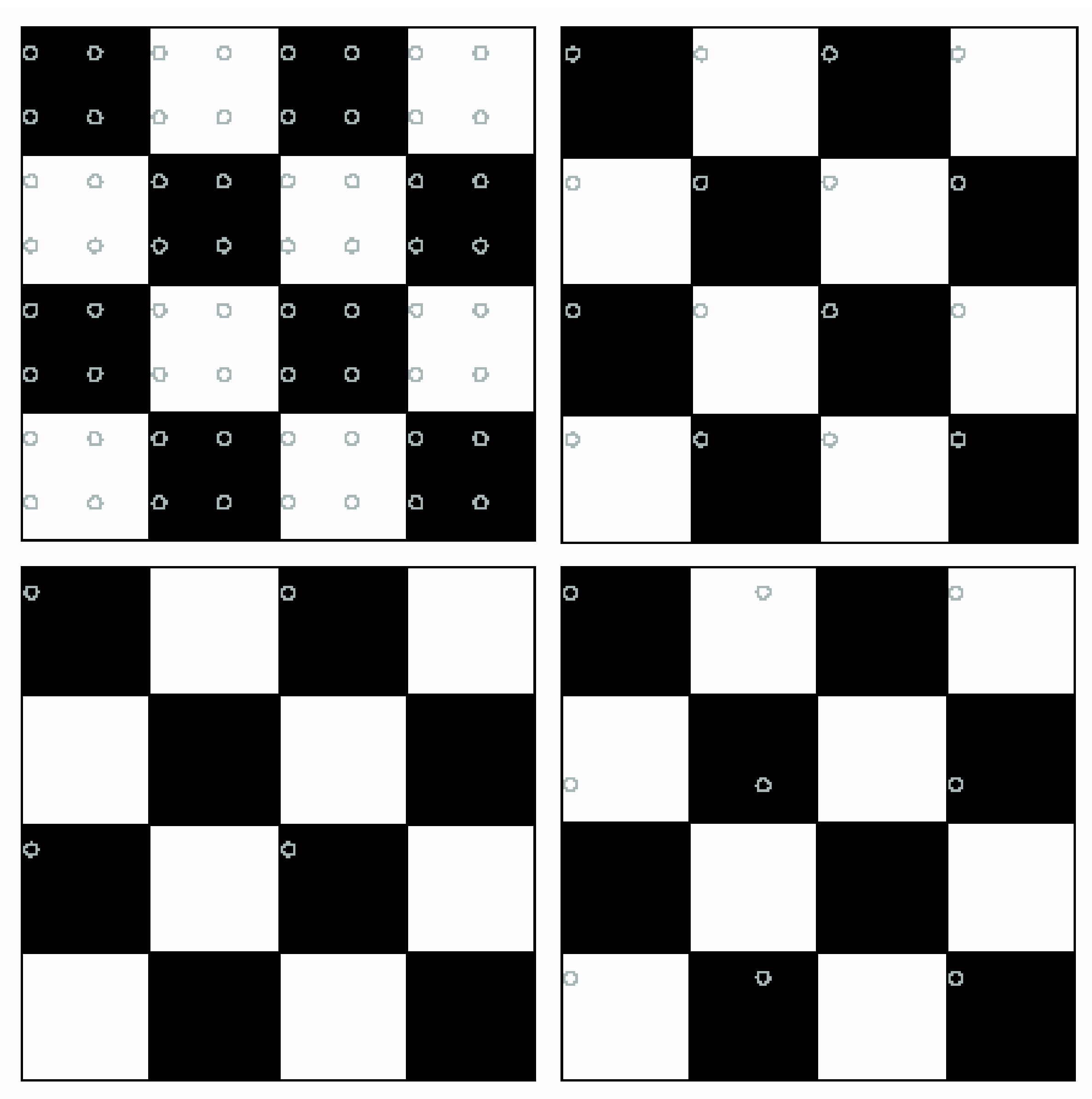
Sampling rate ≥ 2 * max frequency in the image

1. Raise sampling rate by *oversampling*
   - *Sample at k times the resolution*
   - continuous signal: easy
   - discrete signal: need to interpolate
2. Lower the max frequency by *prefiltering*
   - Smooth the signal enough
   - Works on discrete signals
3. Improve sampling quality with better sampling
   - Nyquist is best case!
   - Stratified sampling (jittering)
   - Importance sampling (salaries in Seattle)
   - Relies on domain knowledge



jittered,
9 samples per pixel

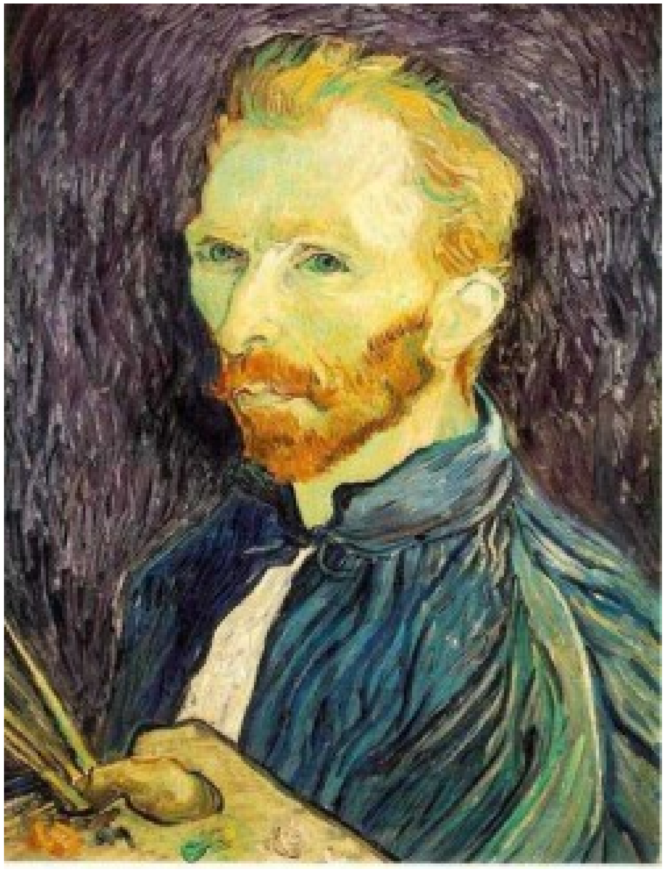
Sampling



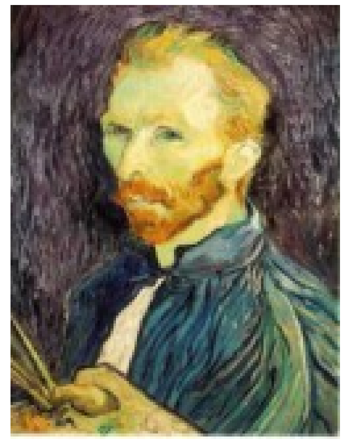
Good sampling:
- Sample often or,
- Sample wisely

Bad sampling:
- see aliasing in action!

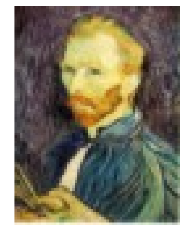
Gaussian pre-filtering



Gaussian 1/2



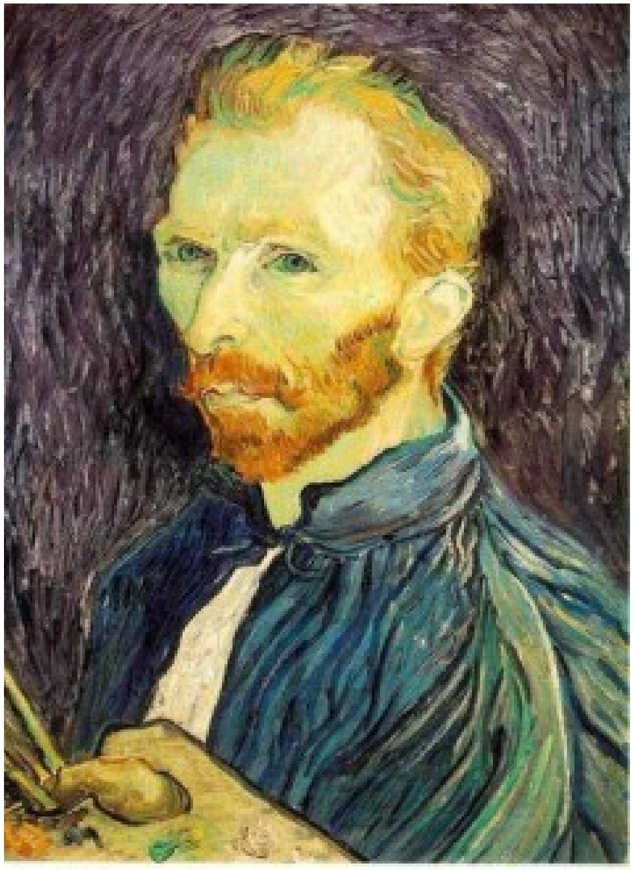
G 1/4



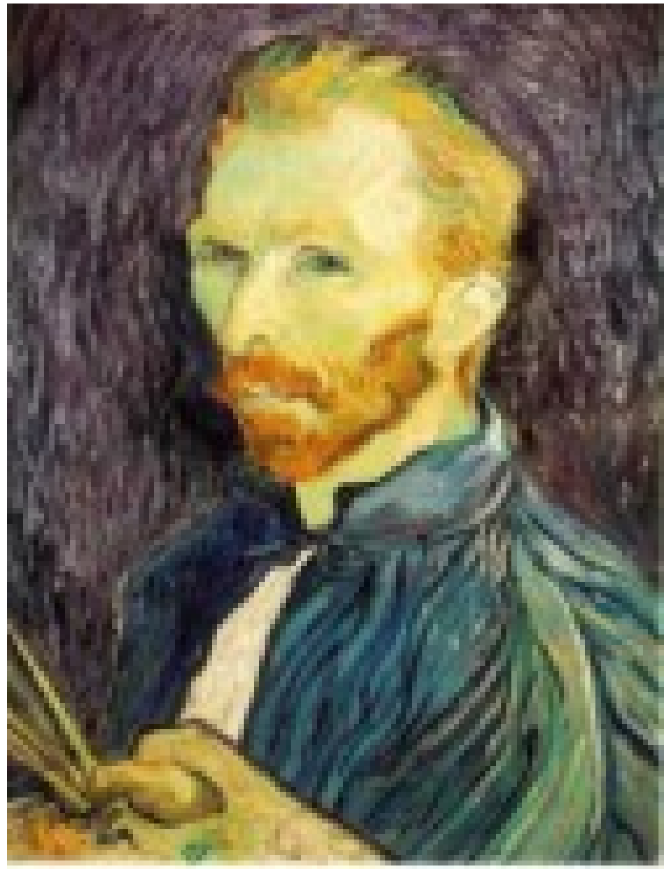
G 1/8

Solution: filter the image, *then* subsample

Subsampling with Gaussian pre-filtering



Gaussian 1/2

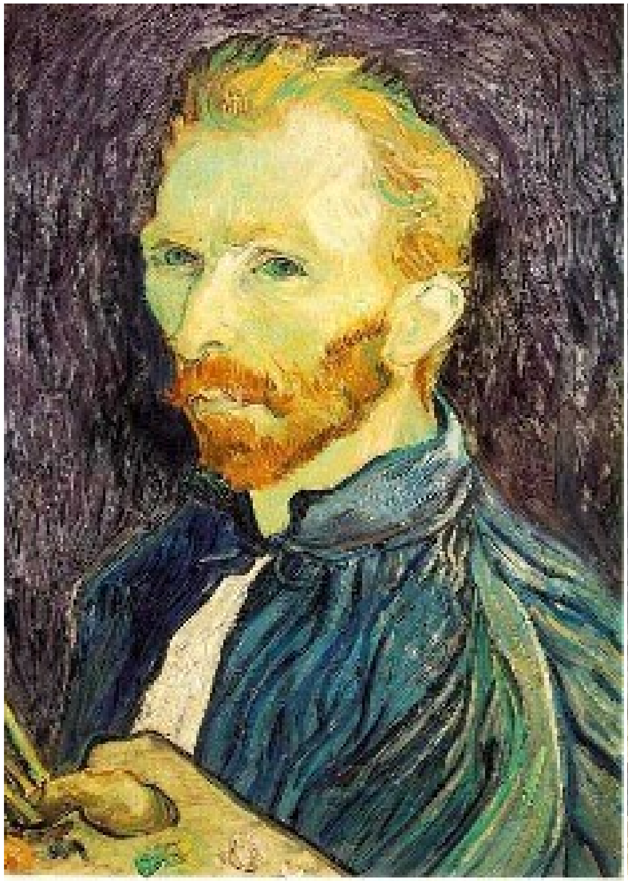


G 1/4

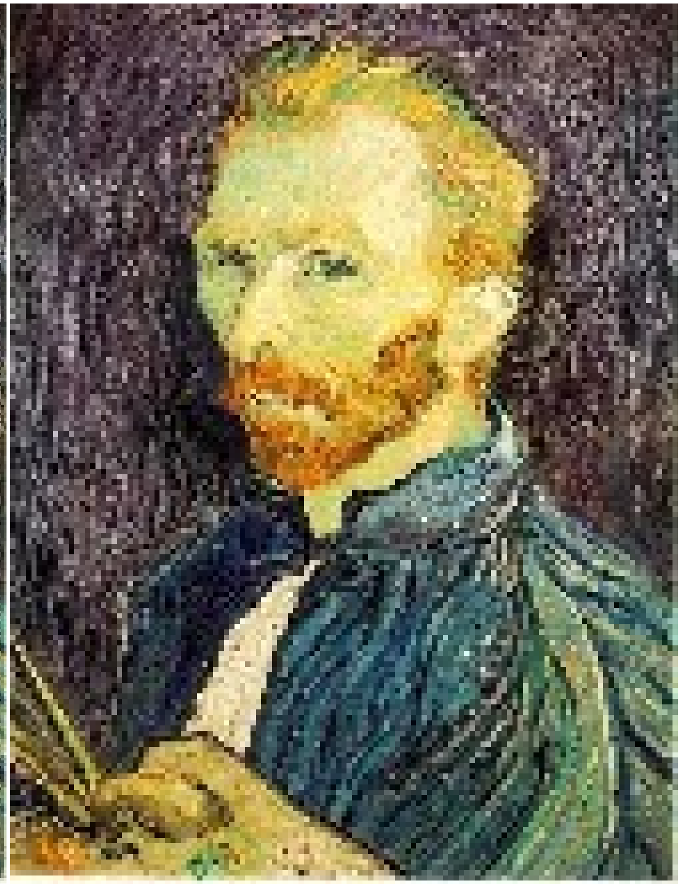
G 1/8

Solution: filter the image, *then* subsample

Compare with...



1/2



1/4 (2x zoom)

1/8 (4x zoom)