What does blurring remove?

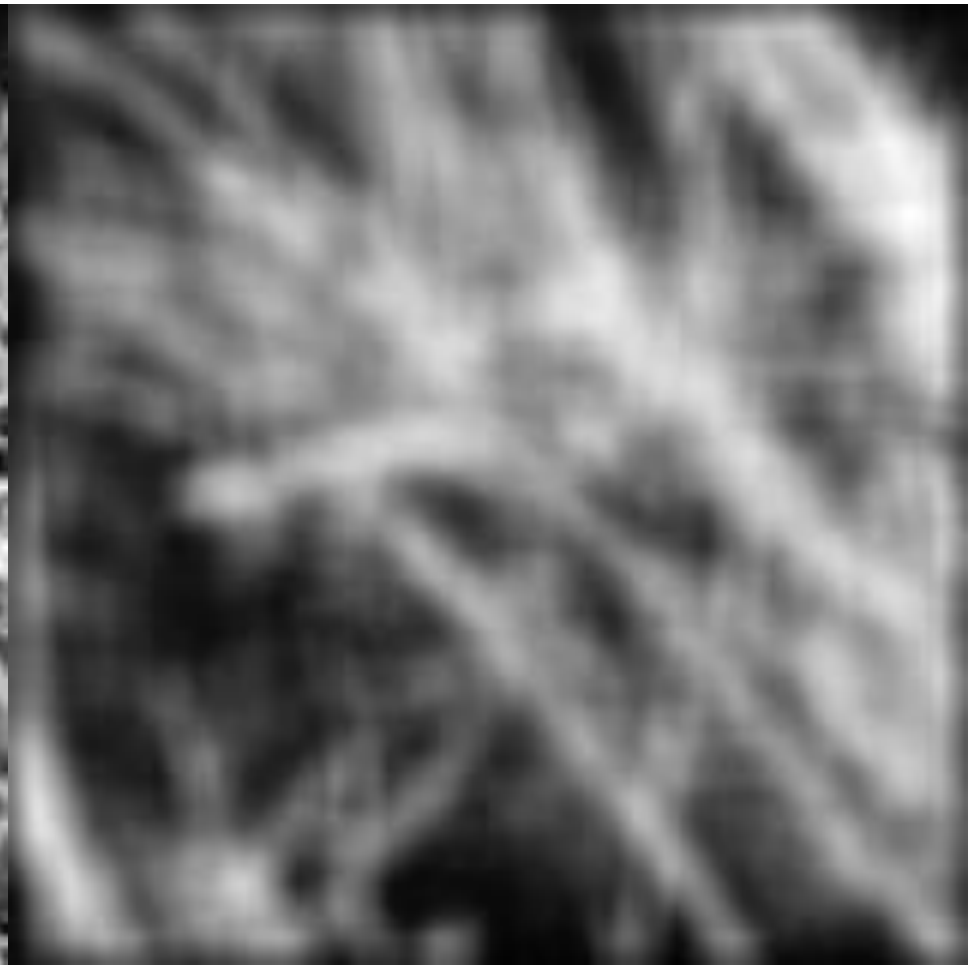
Now let's add it back……

Smoothing with box filter

David Lowe

$$f[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



A few more filters

original

3x3 box filter

do you see
any problems
in this image?

Example: Smoothing by Averaging

Gaussian filters

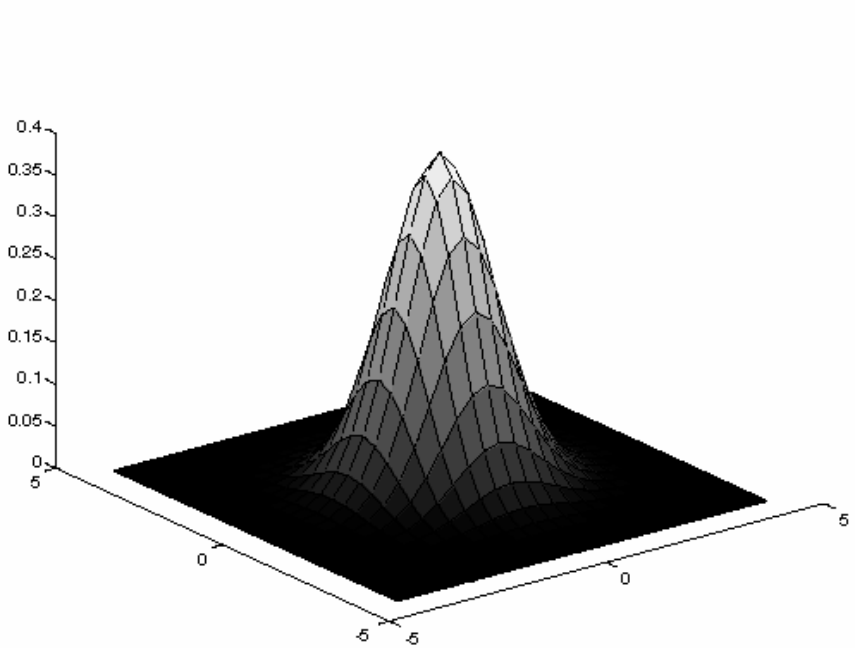
- Remove “high-frequency” components from the image (a low-pass filter)
  - Images become more smooth
- Gaussian convolved with Gaussian…

  …is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving twice with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

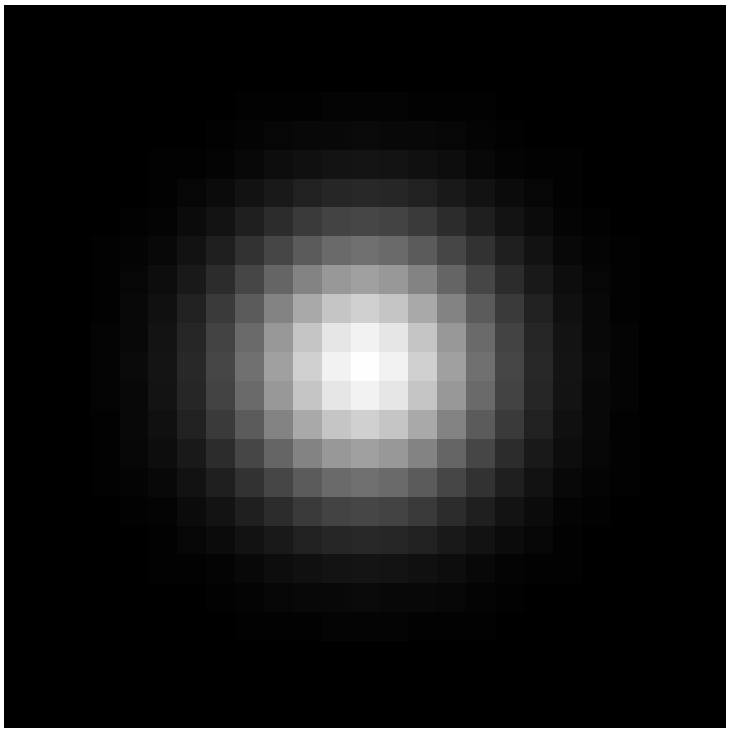
Gaussian Averaging

- Rotationally symmetric.
- Weights nearby pixels more than distant ones.
  - This makes sense as probabalistic inference.



- A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian



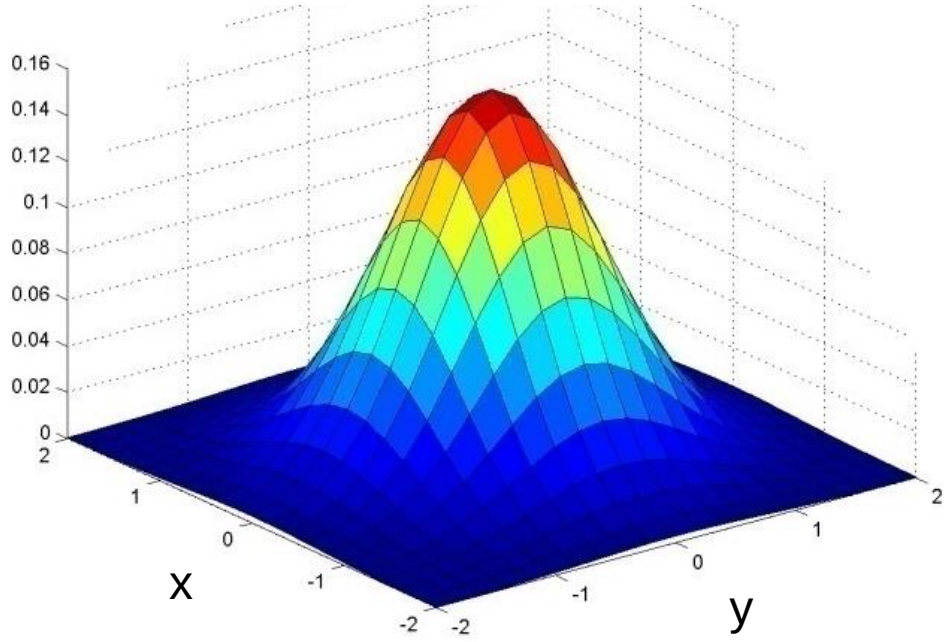
- The picture shows a smoothing kernel proportional to

$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Important filter: Gaussian

Weight contributions of neighboring pixels by nearness

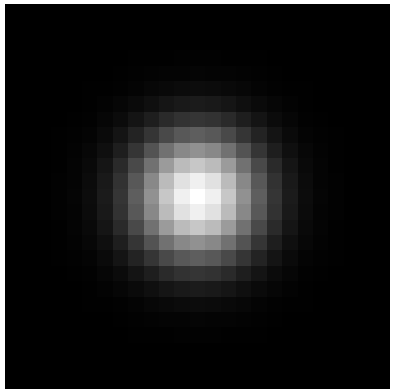


x

y					
	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

Kernel size 5 x 5,
Standard deviation $\sigma = 1$

Viewed from top



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

Slide credit: Christopher Rasmussen

Separability of the Gaussian filter

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

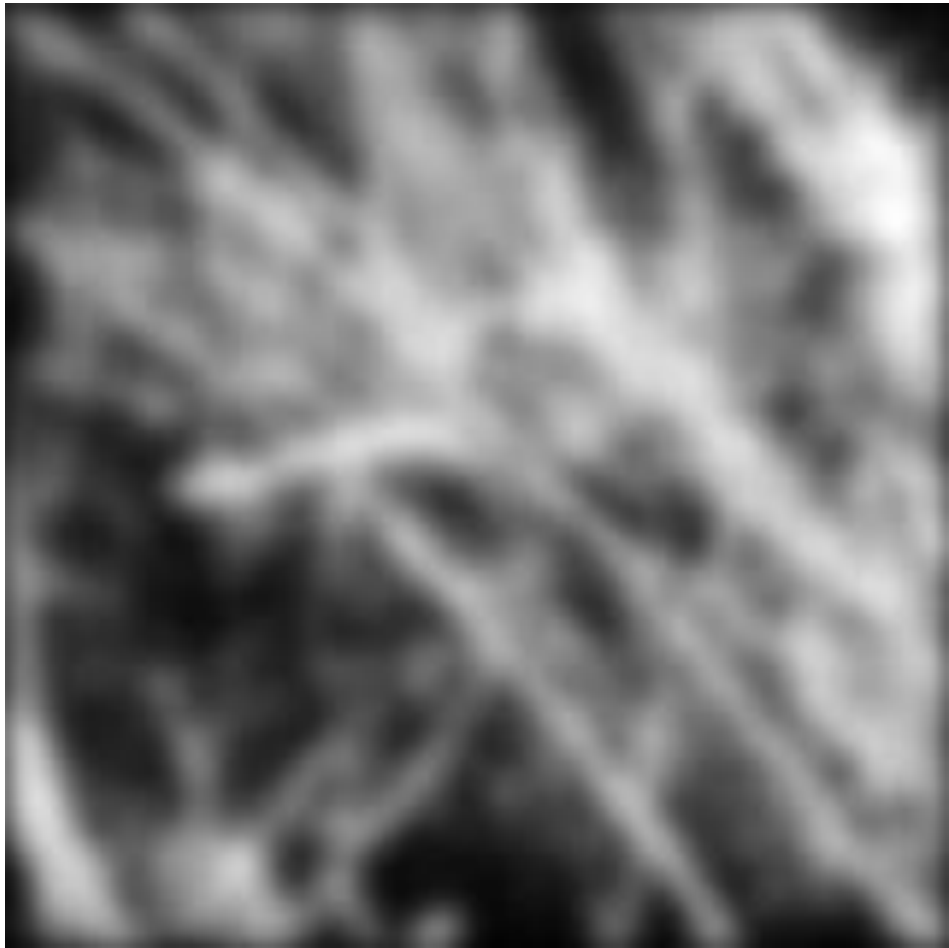
$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

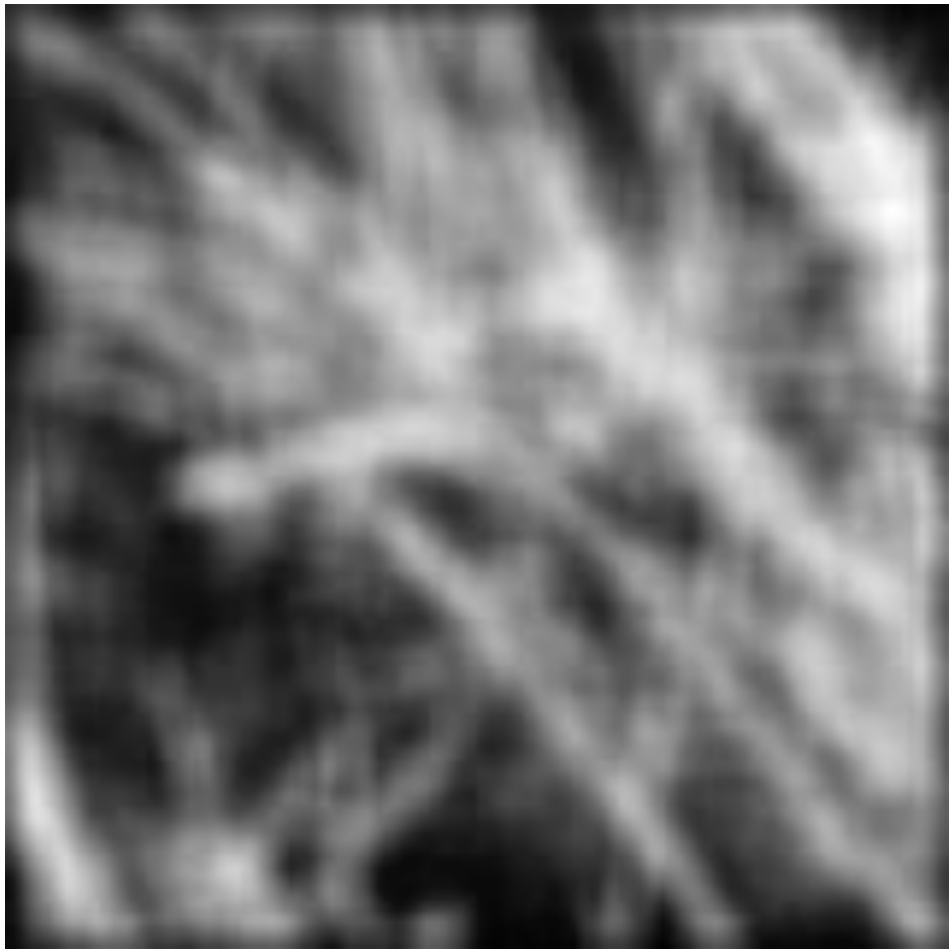
In this case, the two functions are the (identical) 1D Gaussian

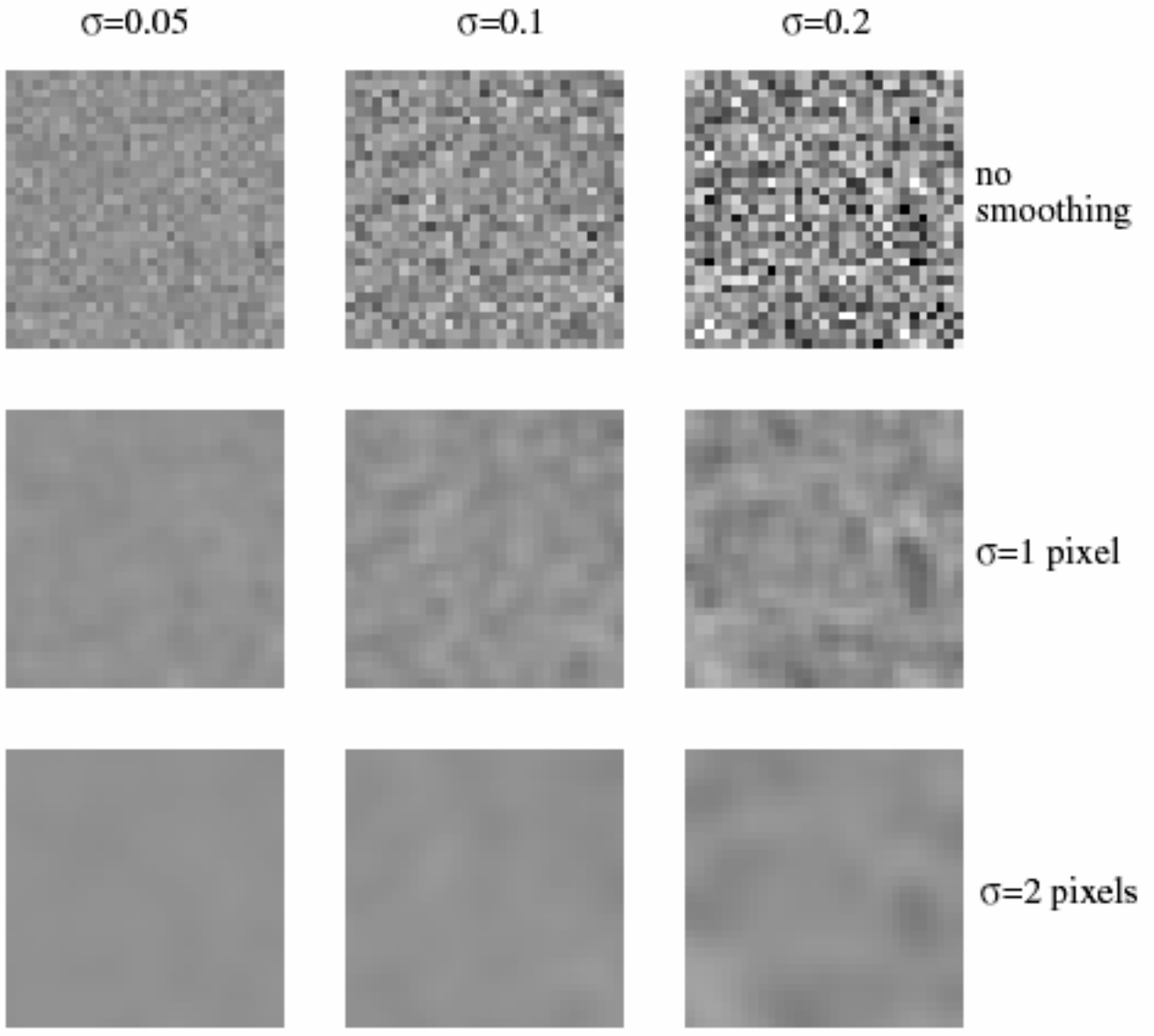
Source: D. Lowe

Smoothing with Gaussian filter



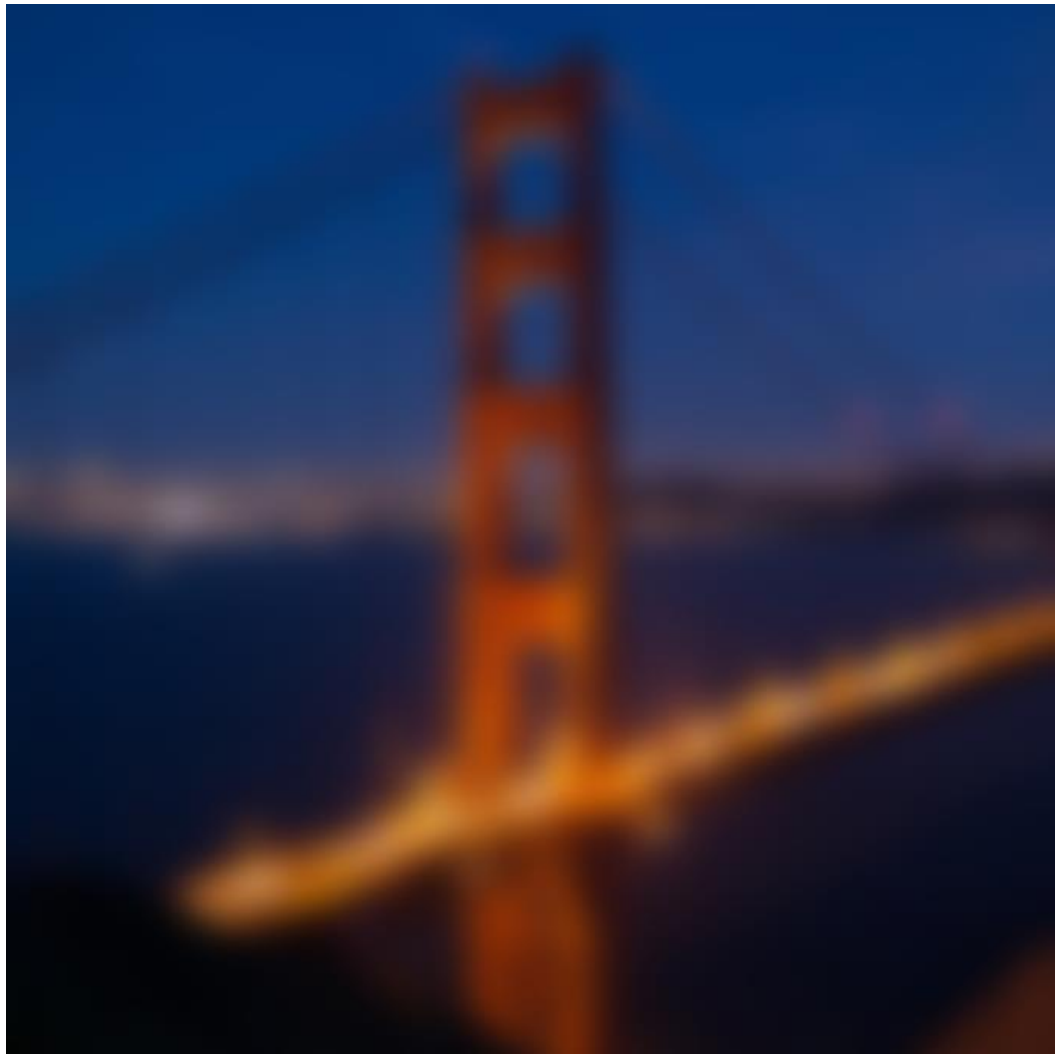
Smoothing with box filter



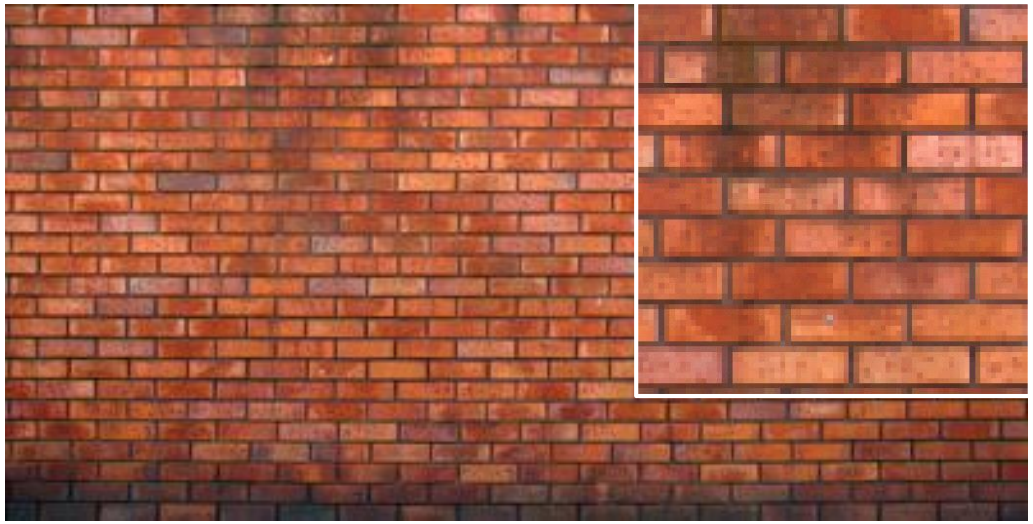


The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

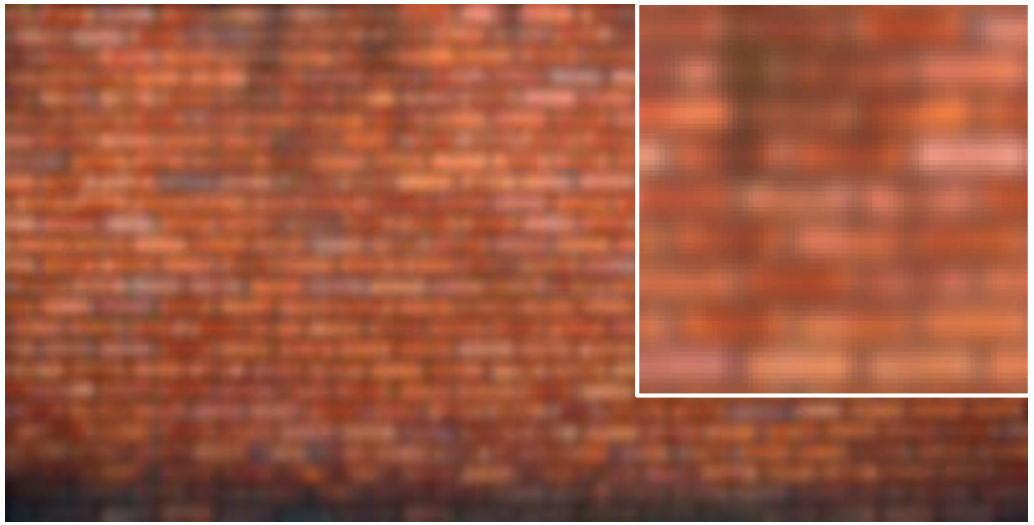
Gaussian filtering example



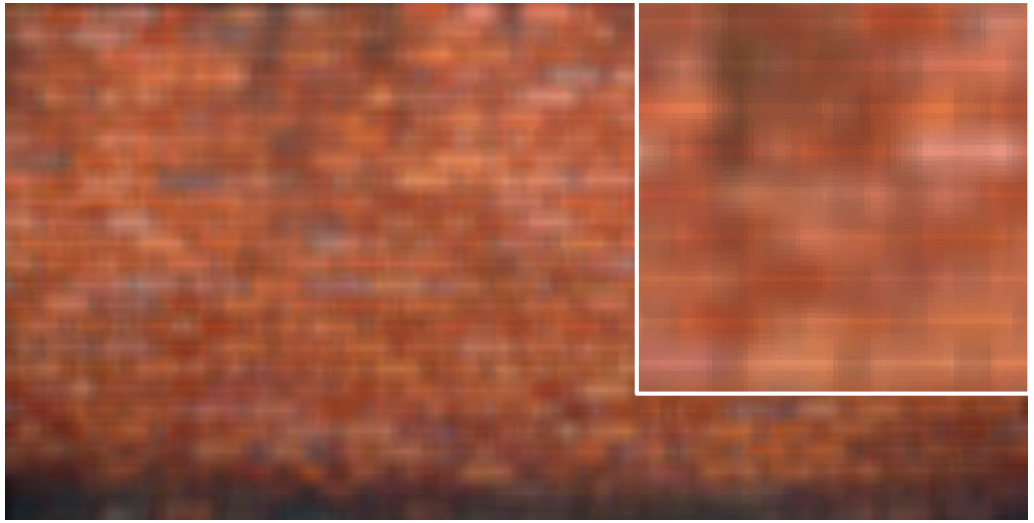
Gaussian vs box filtering



original



7x7 Gaussian

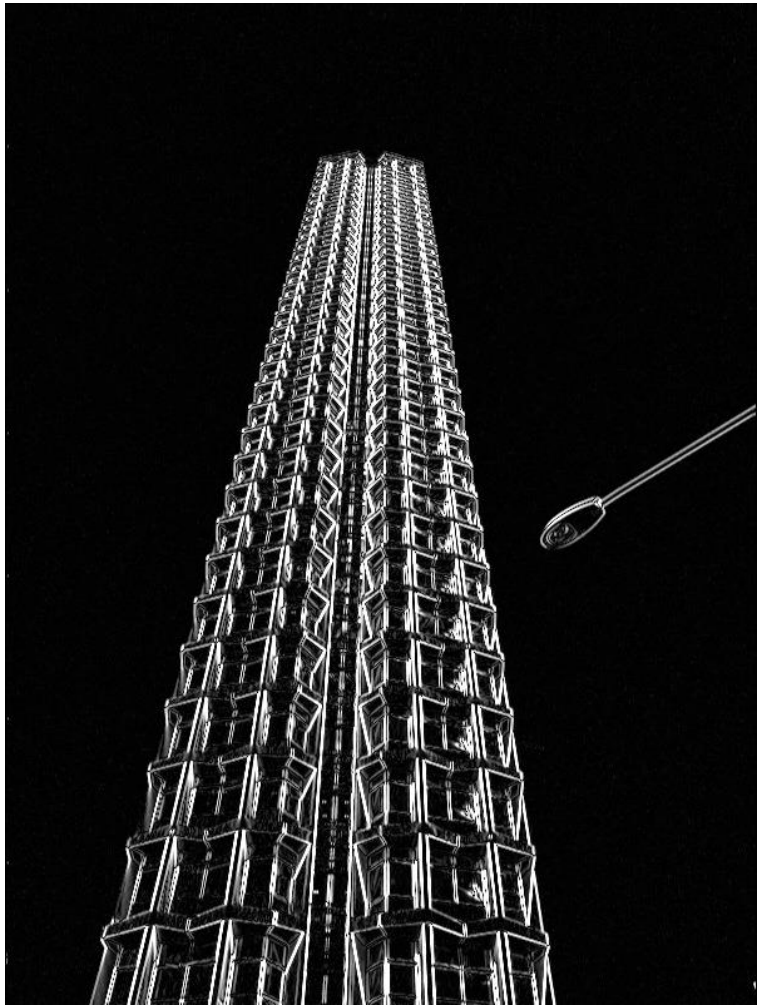


7x7 box

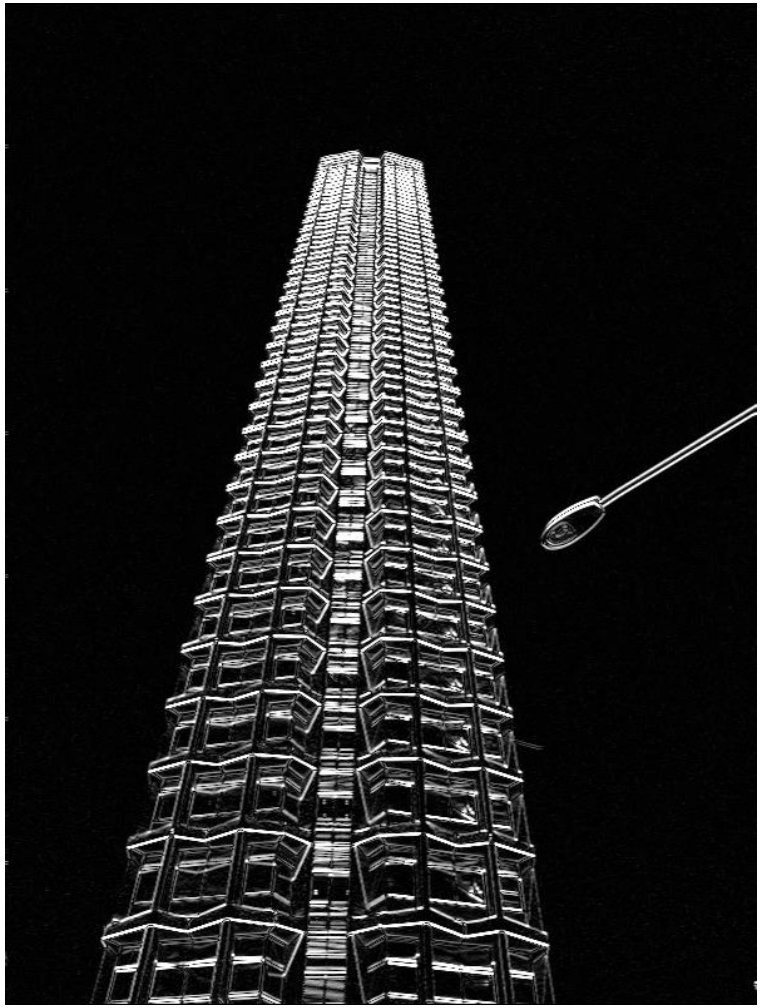
Which blur do you like better?

Sobel filter example

original



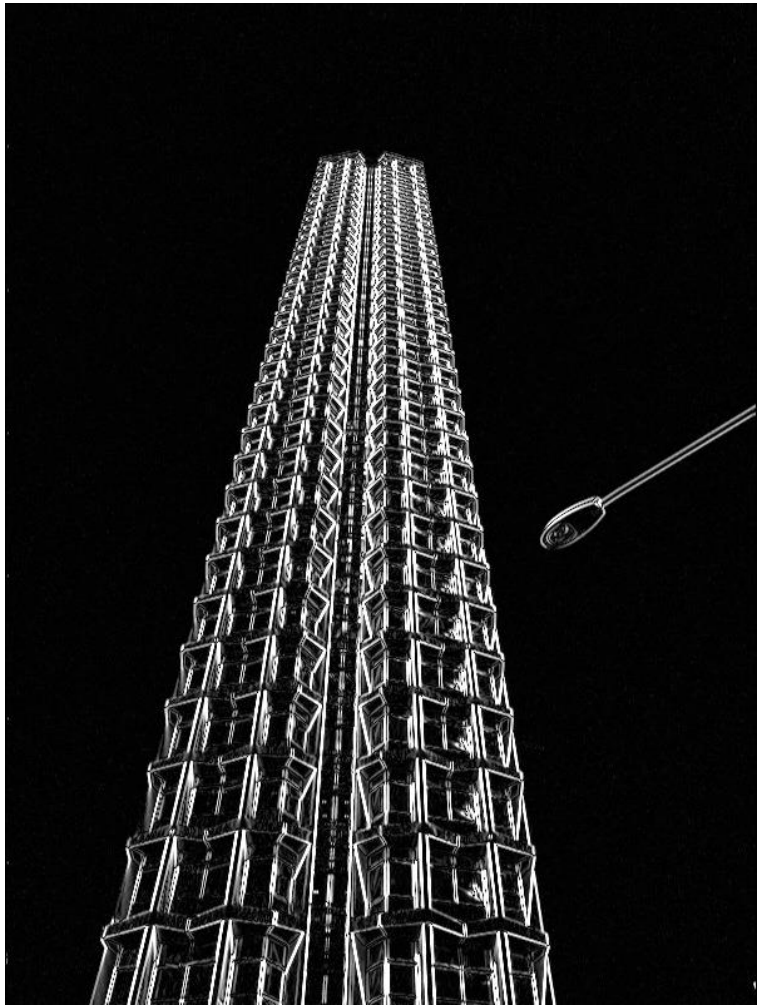
which Sobel filter?



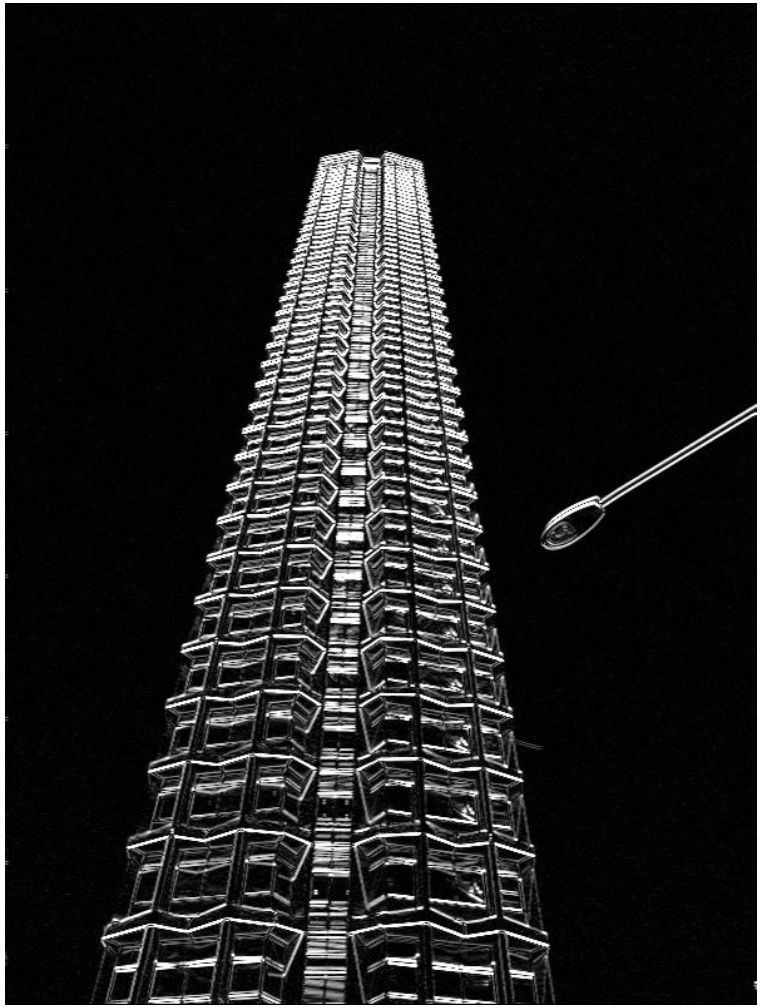
which Sobel filter?

Sobel filter example

original

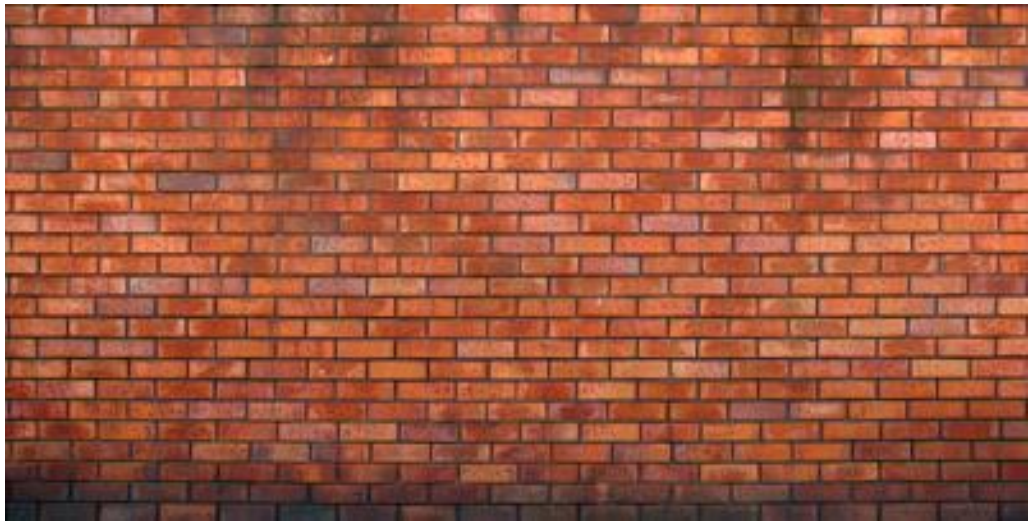


horizontal Sobel filter



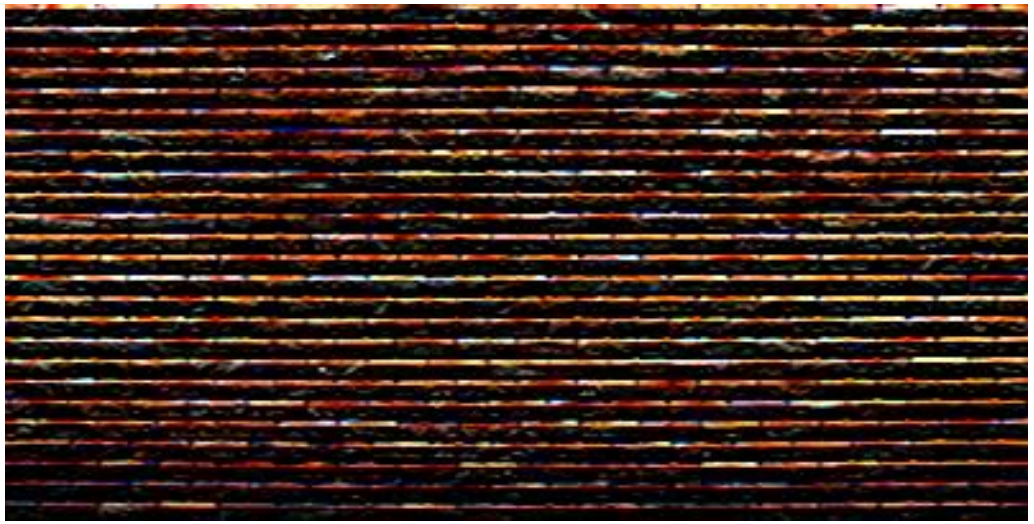
vertical Sobel filter

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

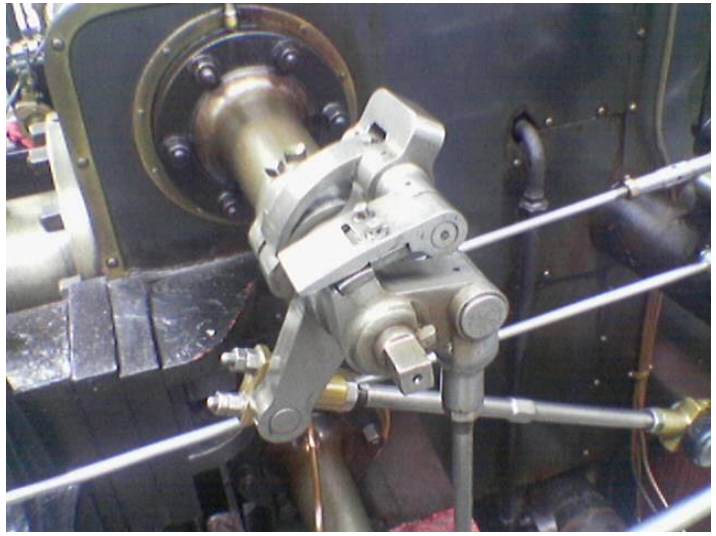
3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

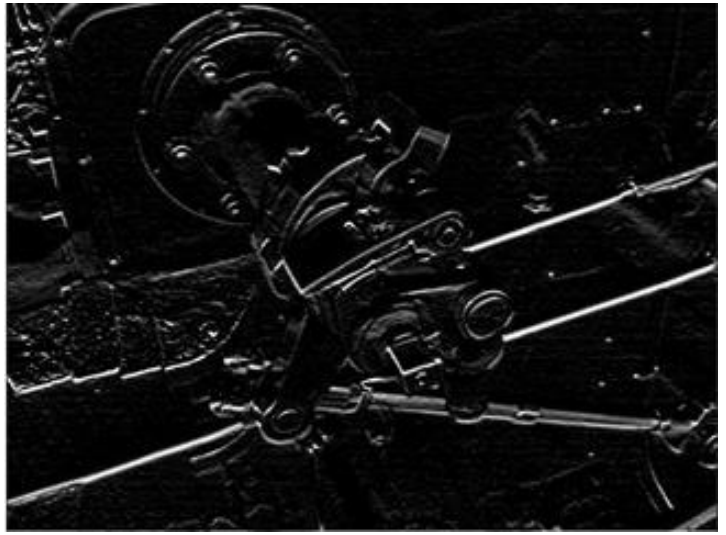
gradient direction amplitude

Image gradient example

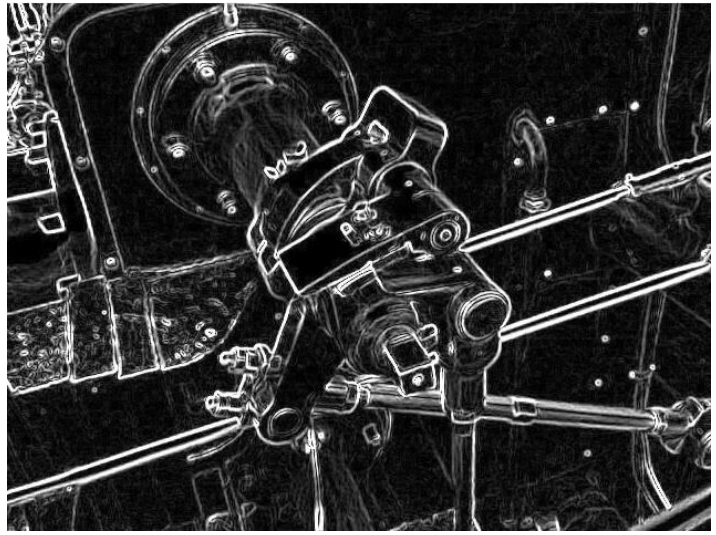
original



vertical derivative



gradient amplitude

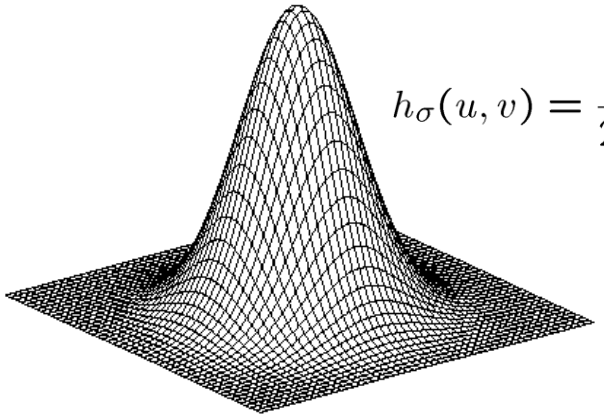


horizontal derivative

How does the gradient direction relate to these edges?

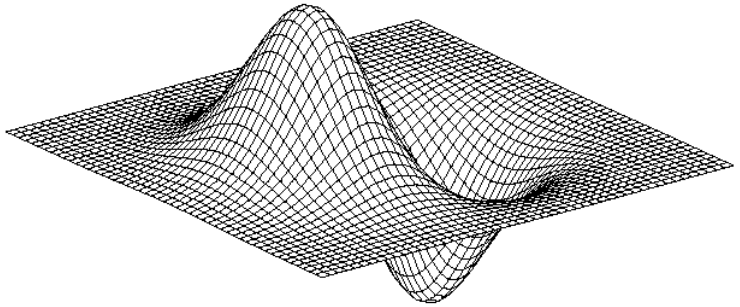
2D Gaussian filters

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



Gaussian

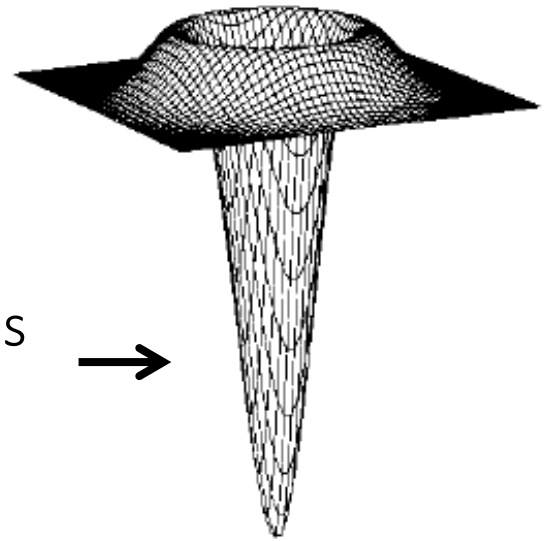
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Derivative of Gaussian

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

how does this relate to this lecture's cover picture?



Laplacian of Gaussian

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

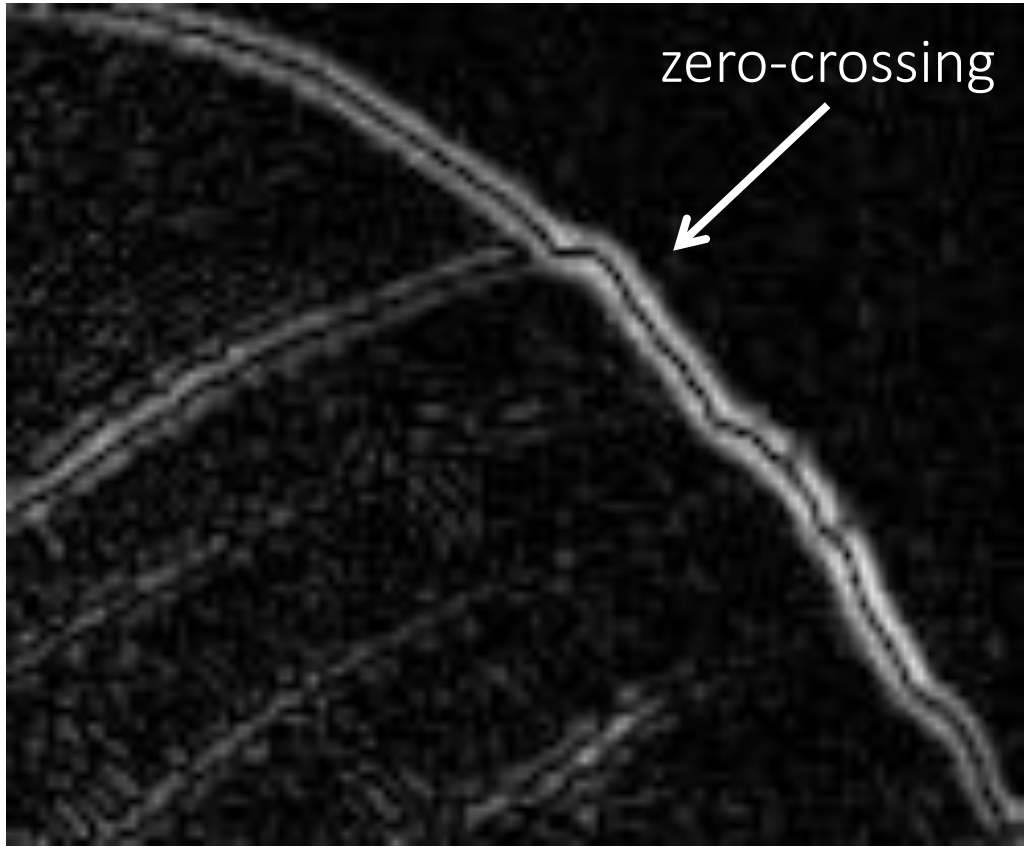
Laplace filtering

Laplacian of Gaussian vs Derivative of Gaussian

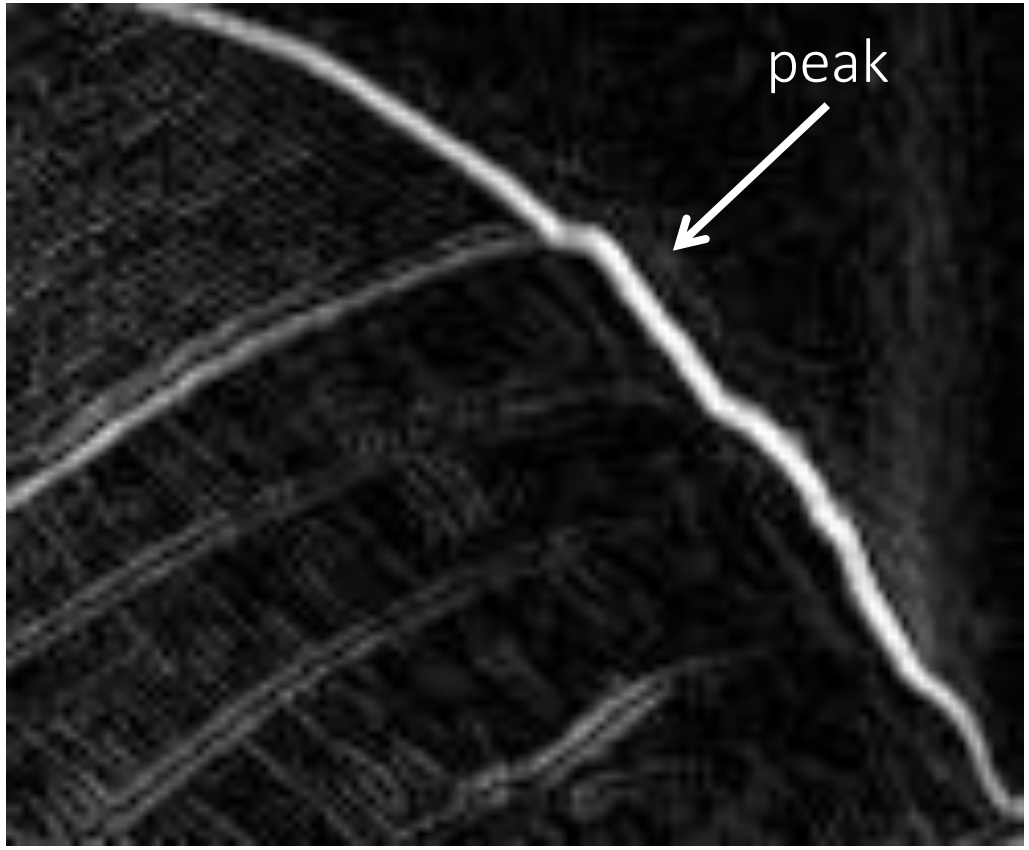
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian



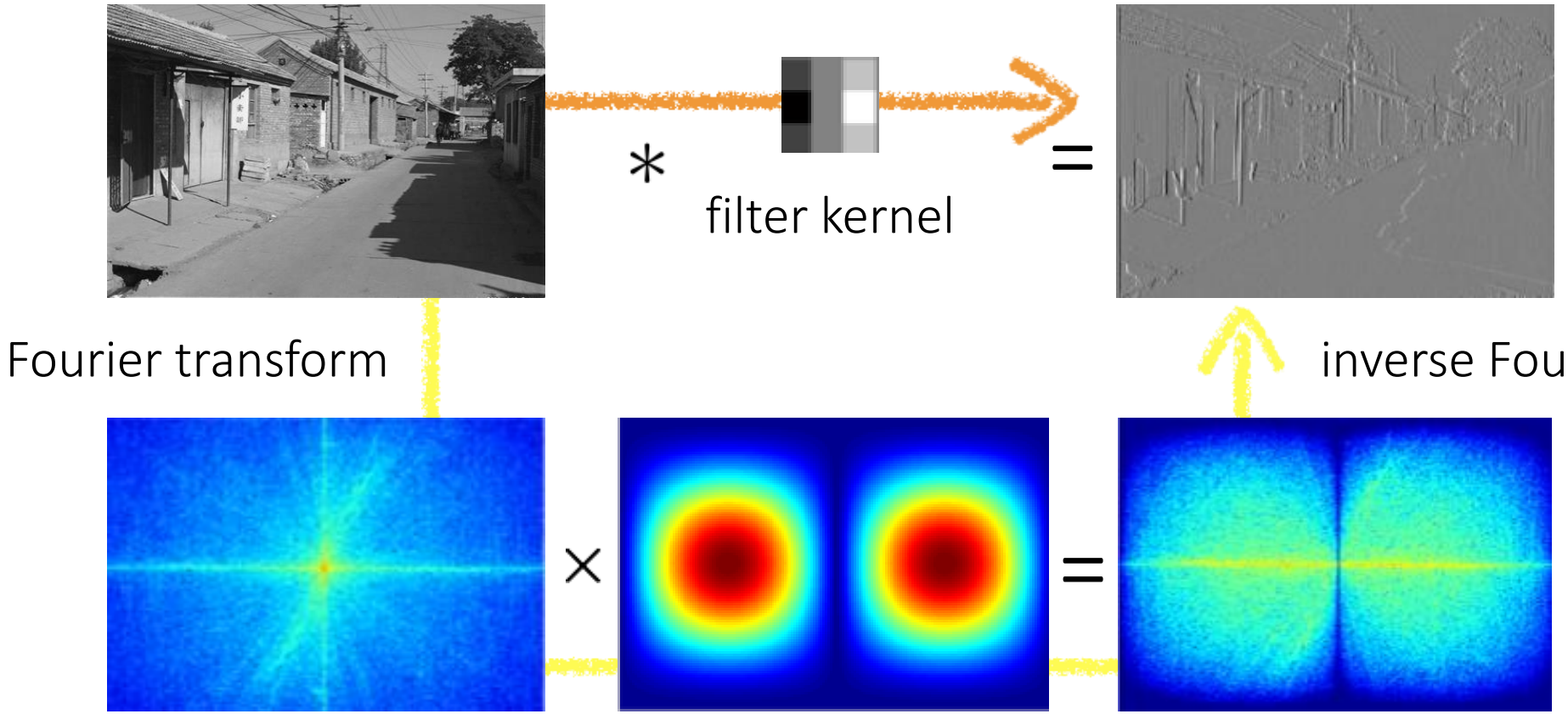
Laplacian of Gaussian filtering



Derivative of Gaussian filtering

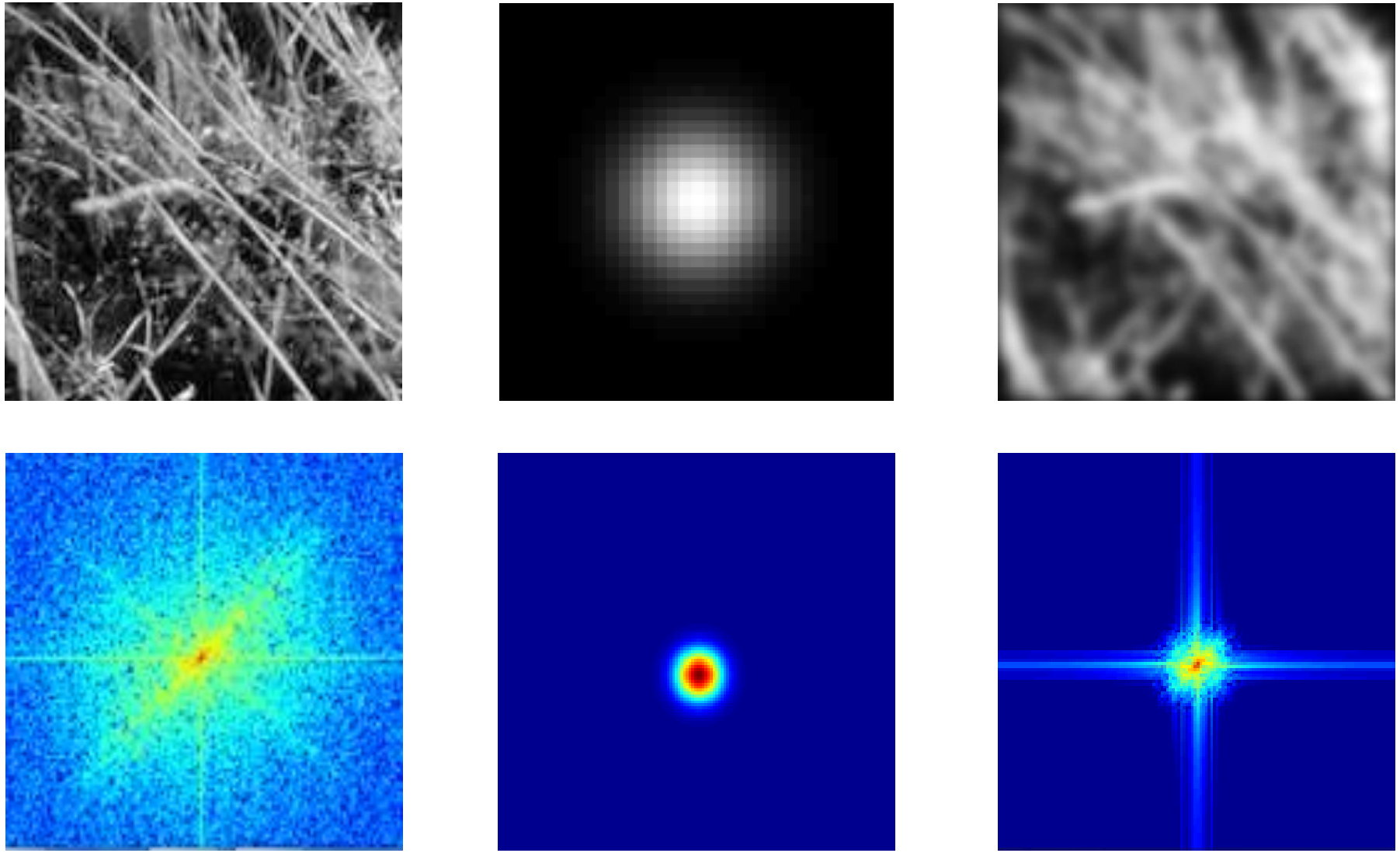
Zero crossings are more accurate at localizing edges (but not very convenient).

Spatial domain filtering

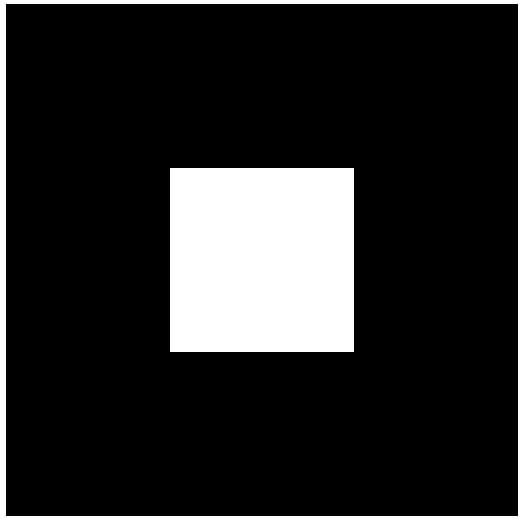
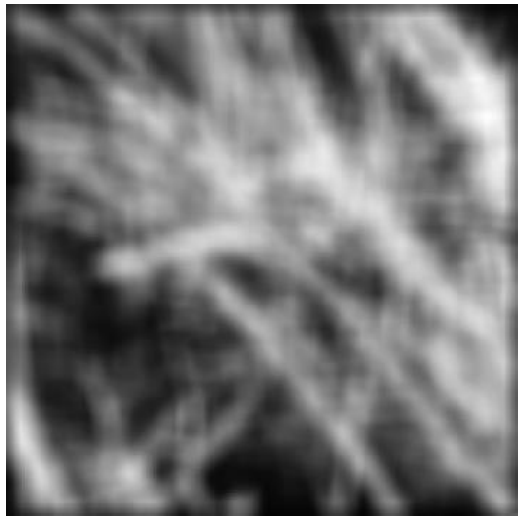
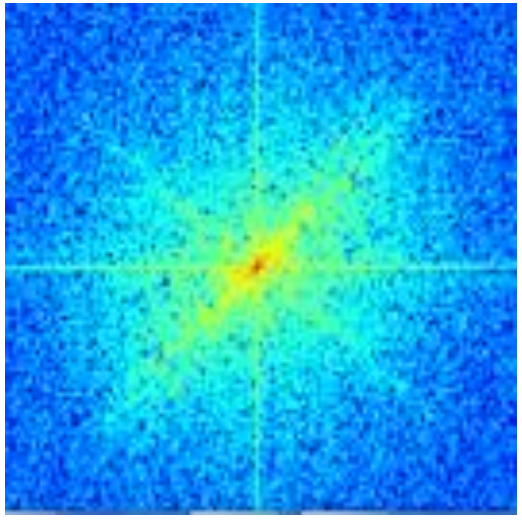
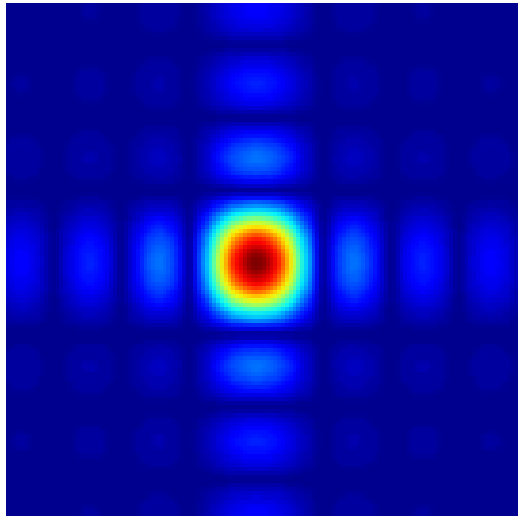
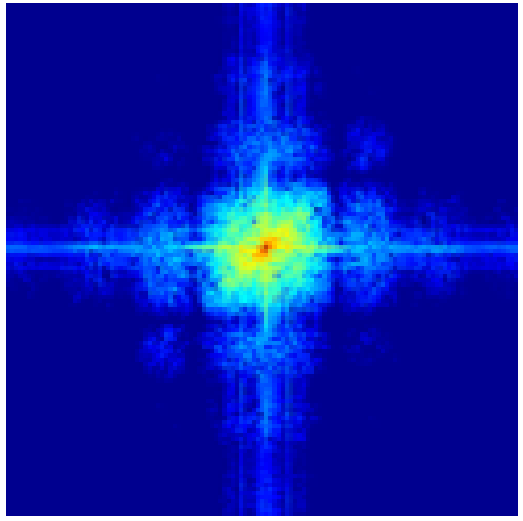


Frequency domain filtering

Gaussian blur

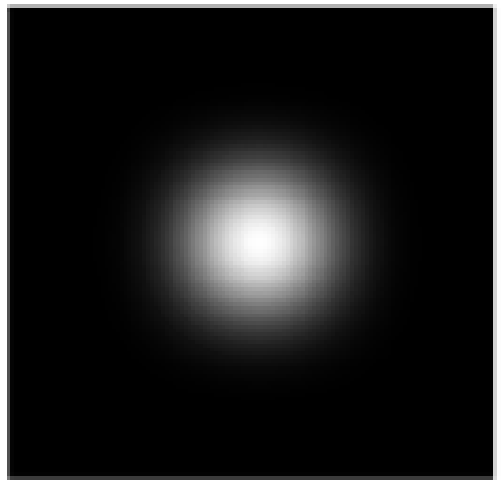


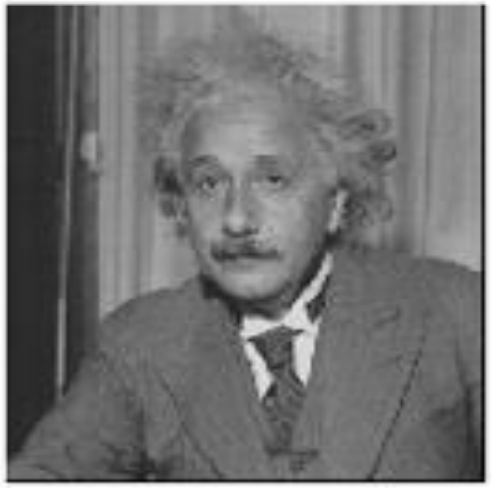
Box blur

More filtering examples

?

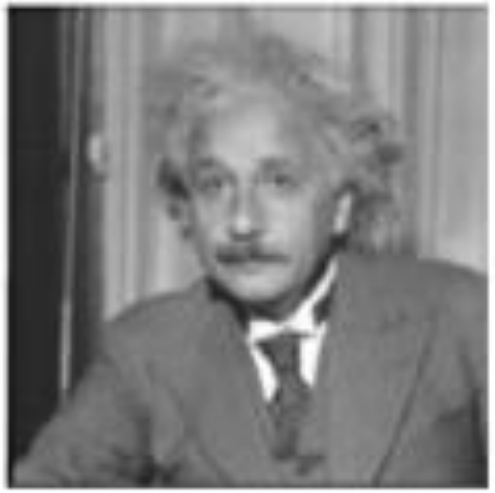




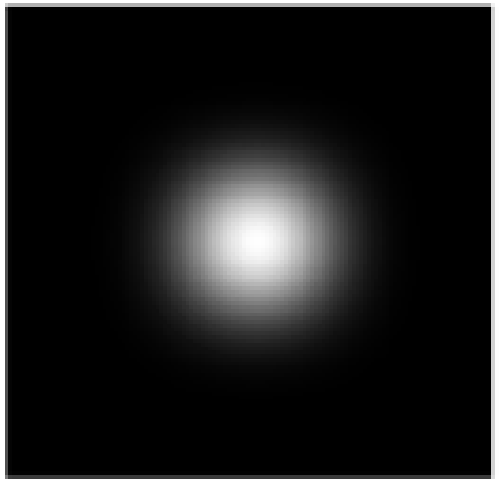
?

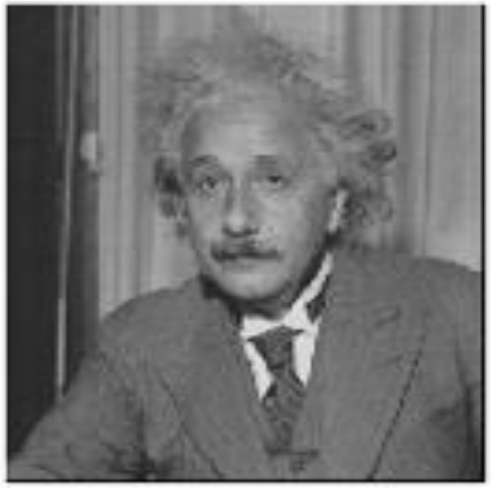
filters shown in frequency-domain

More filtering examples



low-pass





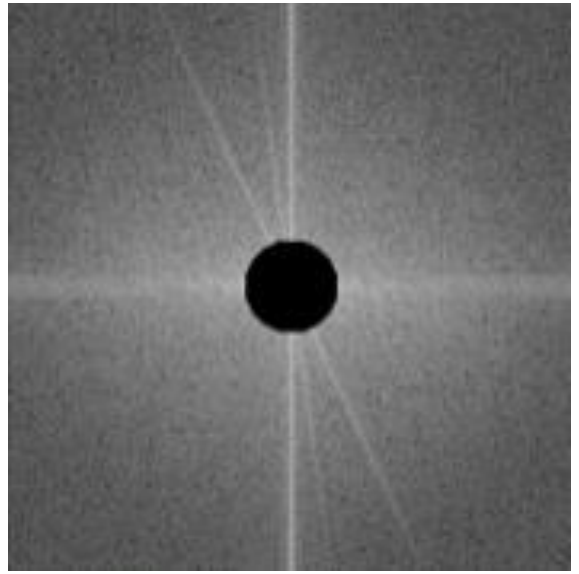
band-pass

filters shown in frequency-domain

More filtering examples

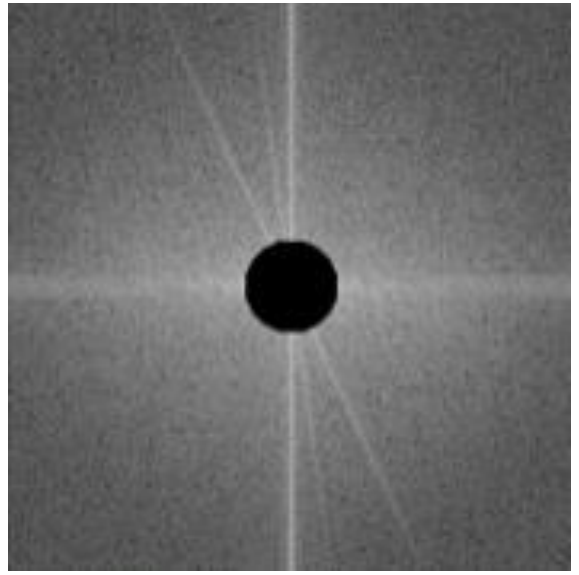
?

high-pass



More filtering examples

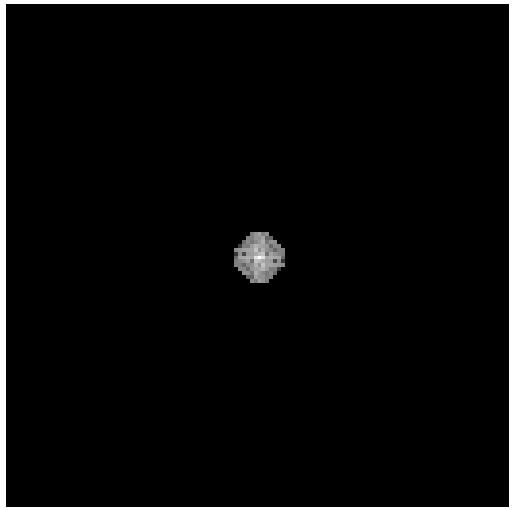
high-pass



More filtering examples

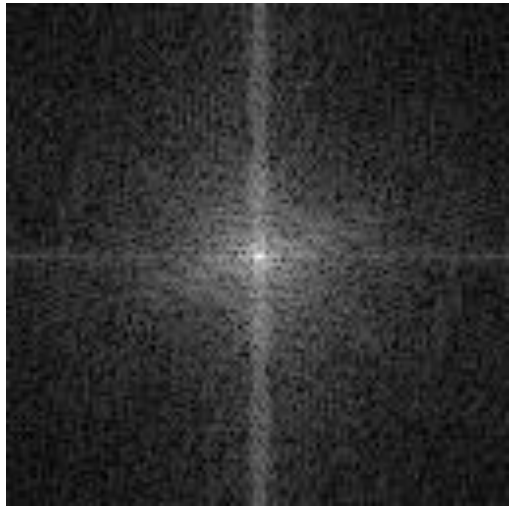
original image

low-pass filter



?

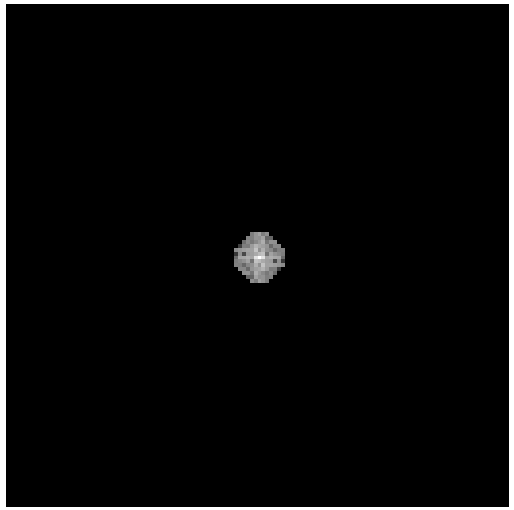
frequency magnitude



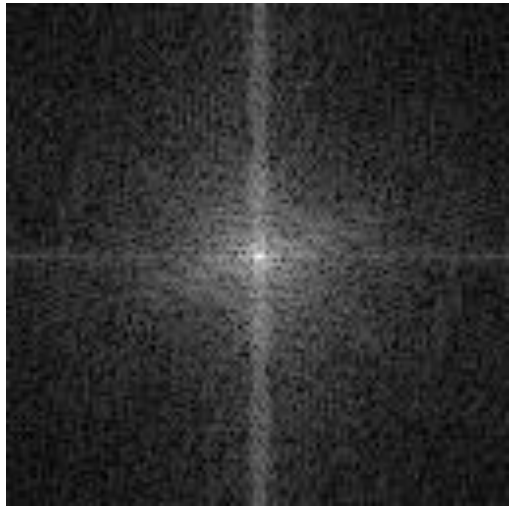
More filtering examples

original image

low-pass filter



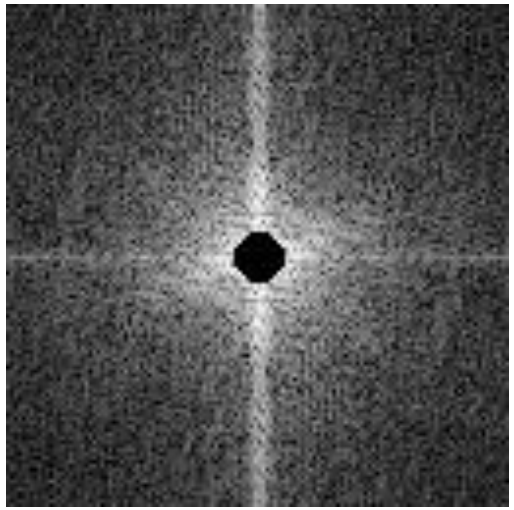
frequency magnitude



More filtering examples

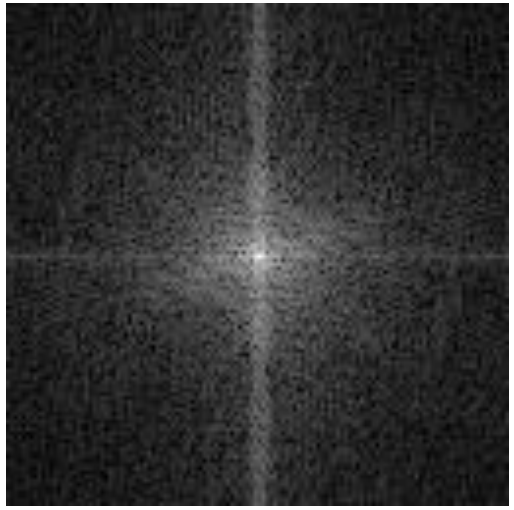
original image

high-pass filter



?

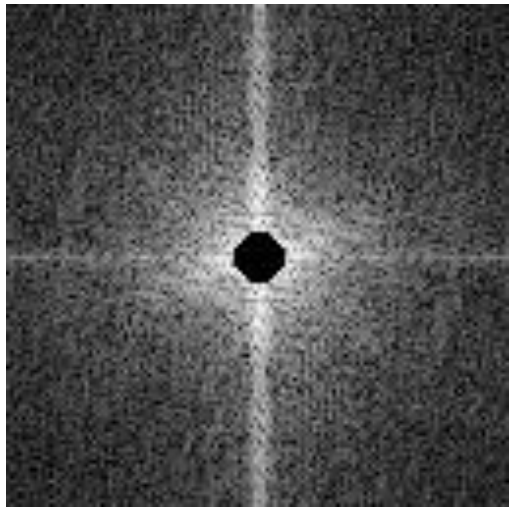
frequency magnitude



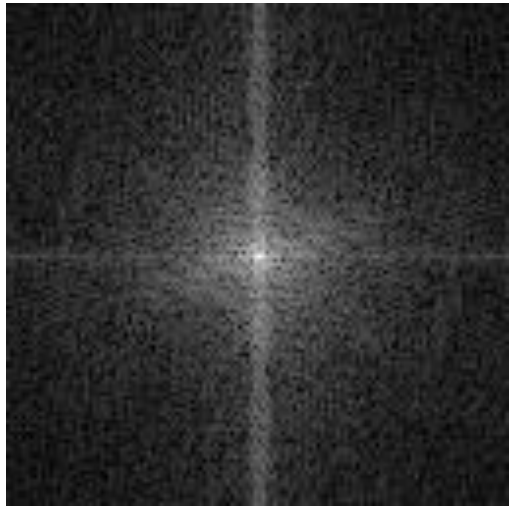
More filtering examples

original image

high-pass filter



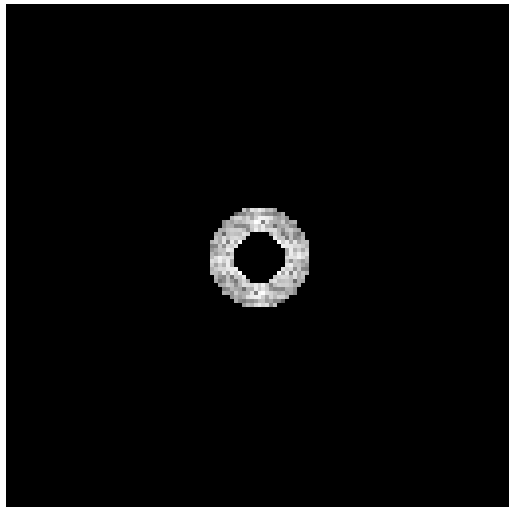
frequency magnitude

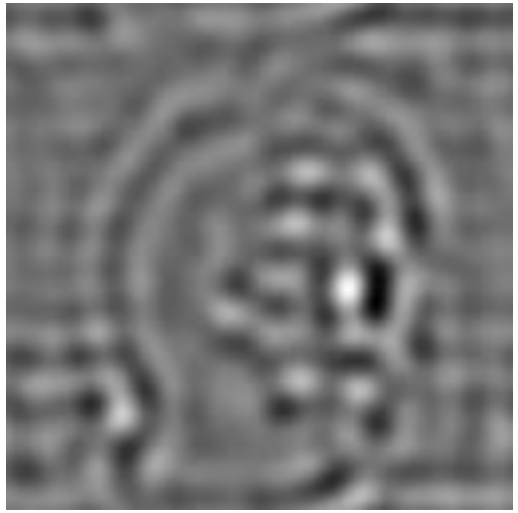


More filtering examples

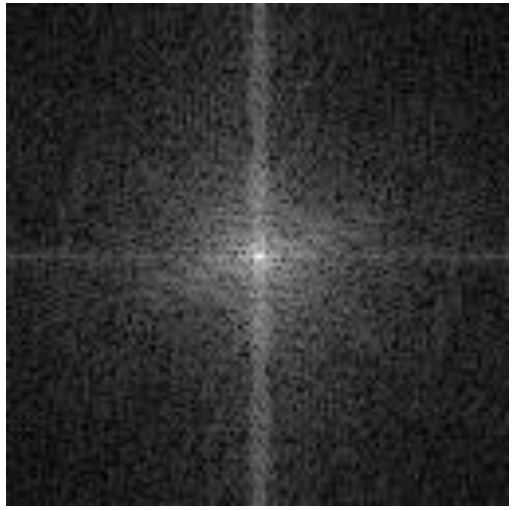
original image

band-pass filter





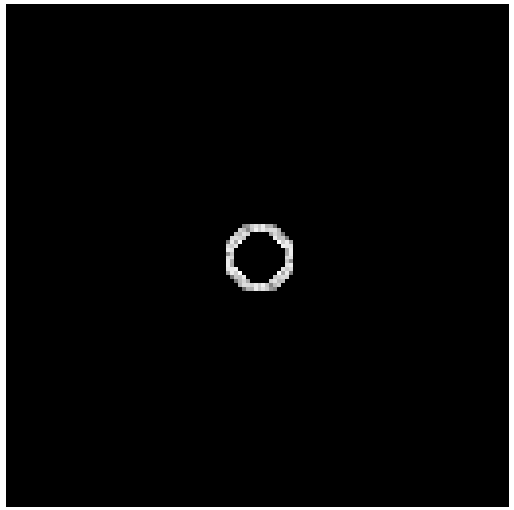
frequency magnitude

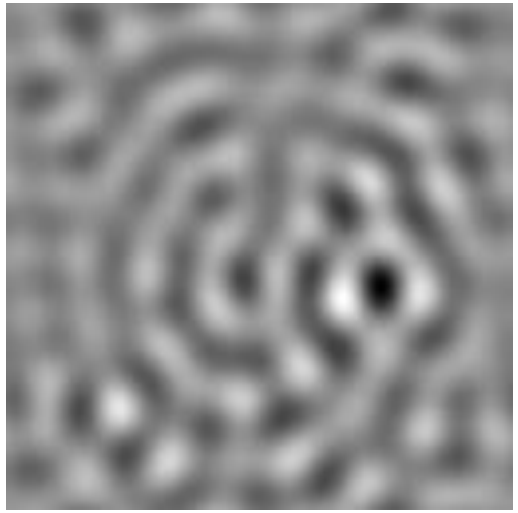


More filtering examples

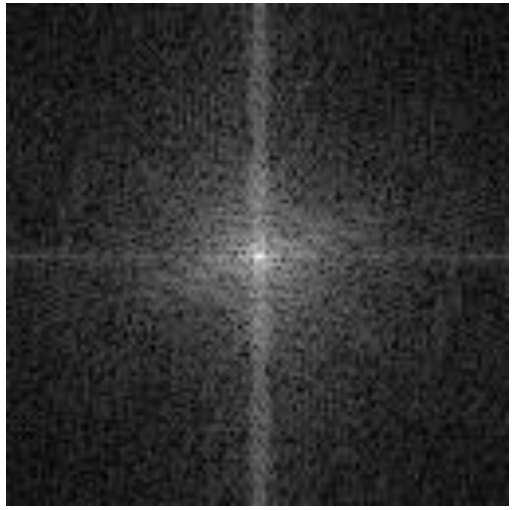
original image

band-pass filter





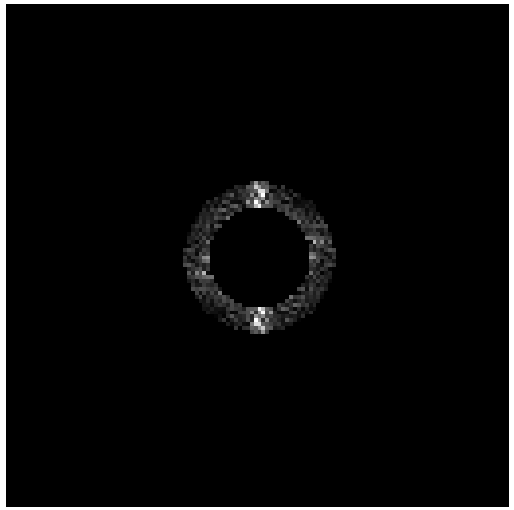
frequency magnitude



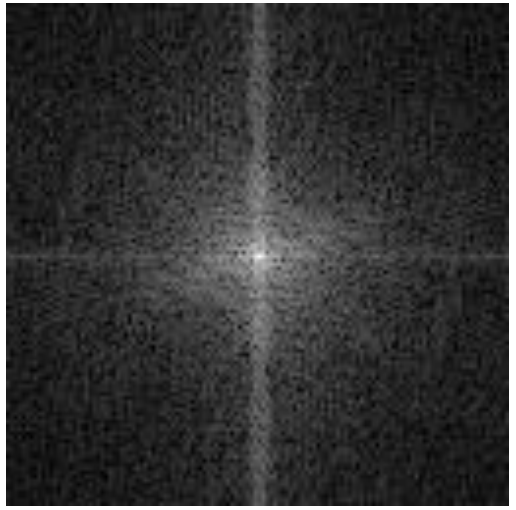
More filtering examples

original image

band-pass filter



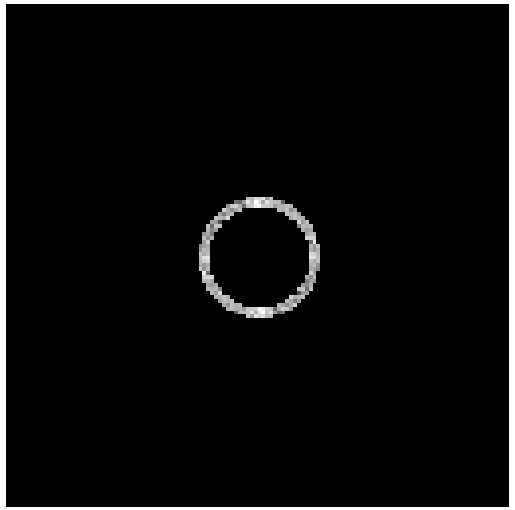
frequency magnitude

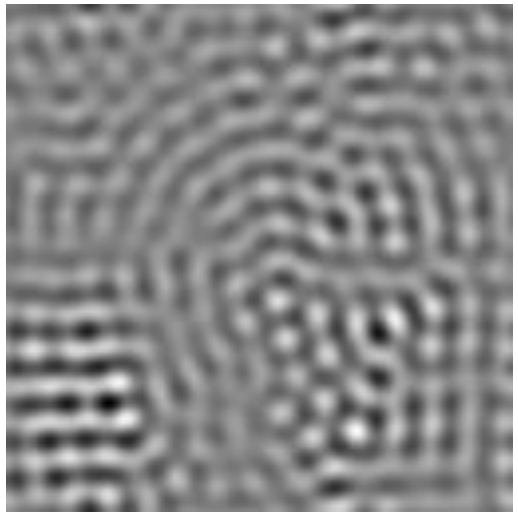


More filtering examples

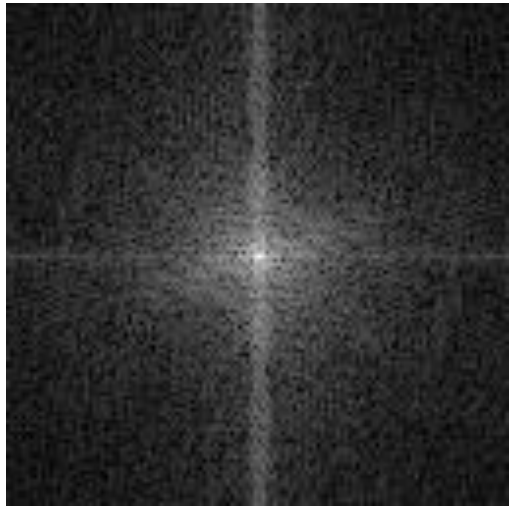
original image

band-pass filter





frequency magnitude



Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Original

Salt and pepper noise

Impulse noise

Gaussian noise

Source: S. Seitz

Filtering to reduce noise

- Noise is what we're not interested in.
  - We'll discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

- I = S + N. Noise doesn't depend on signal.
- We'll consider:

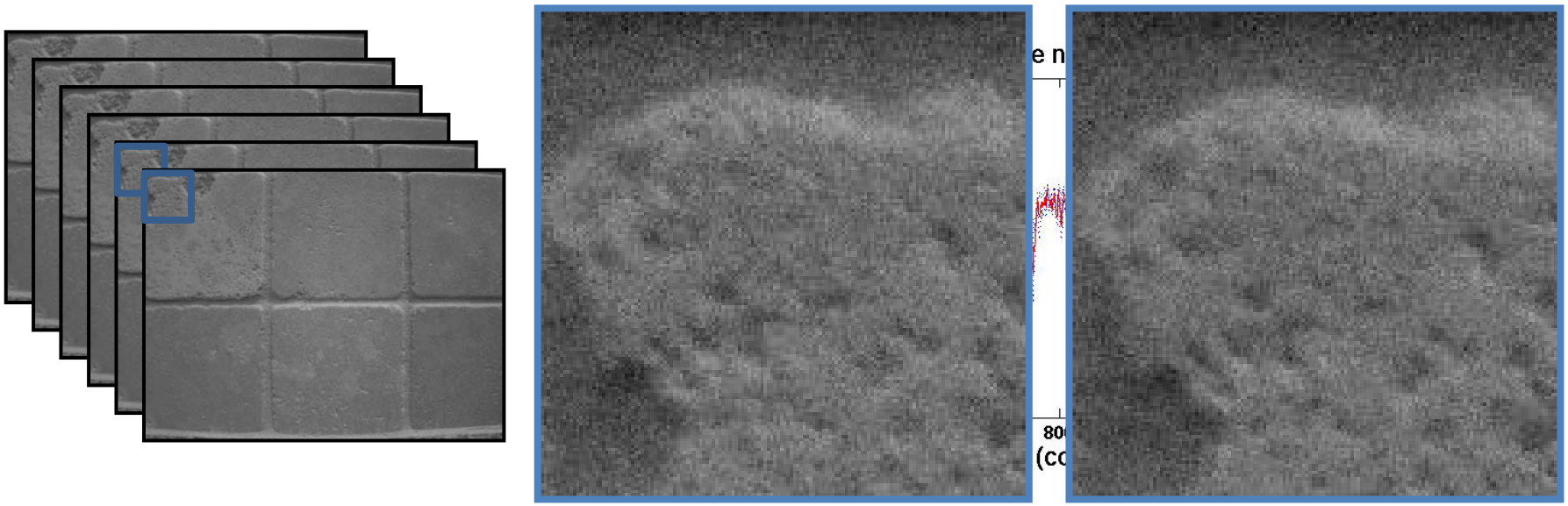
$$I_i = s_i + n_i \text{ with } E(n_i) = 0$$

s_i deterministic.

n_i, n_j independent for $n_i \neq n_j$

n_i, n_j identically distributed

Motivation: noise reduction



- We can measure **noise** in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

Average Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(Camps)

Does it reduce noise?

- Intuitively, takes out small variations.

$$I(i,j) = \hat{I}(i,j) + N(i,j) \text{ with } \mathrm{N(i,\ j)} \sim \mathrm{N}(0,\ \sigma)$$

$$\mathrm{O(i,\ j)} = \frac{1}{\mathrm{m}^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + N(i-h, j-k) =$$

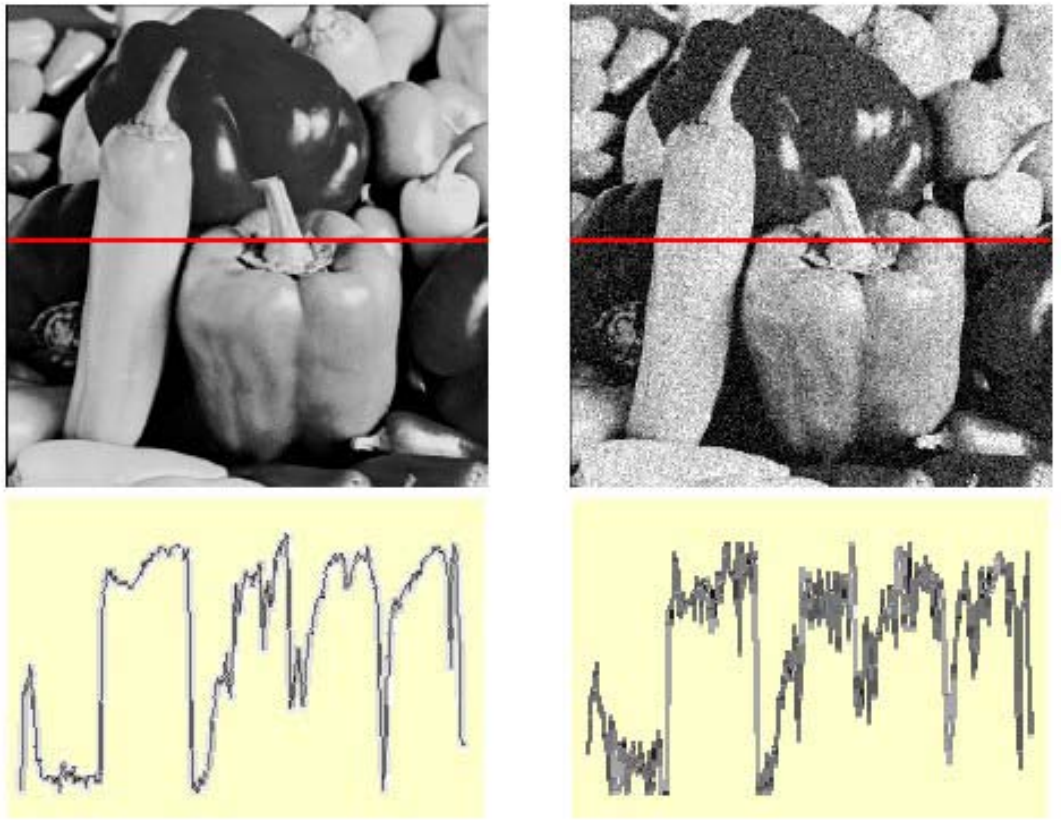
$$= \frac{1}{\mathrm{m}^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + \underbrace{\frac{1}{m^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} N(i-h, j-k)}_{\hat{N}(i,j)}$$

$$E(\hat{N}(i,j)) = 0$$

$$E(\hat{N}^2(i,j)) = \frac{1}{m^2} m\sigma^2 = \frac{\sigma^2}{m} \Rightarrow \hat{N}(i,j) \sim N(0, \frac{\sigma}{\sqrt{m}})$$

(Camps)

Gaussian noise



$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

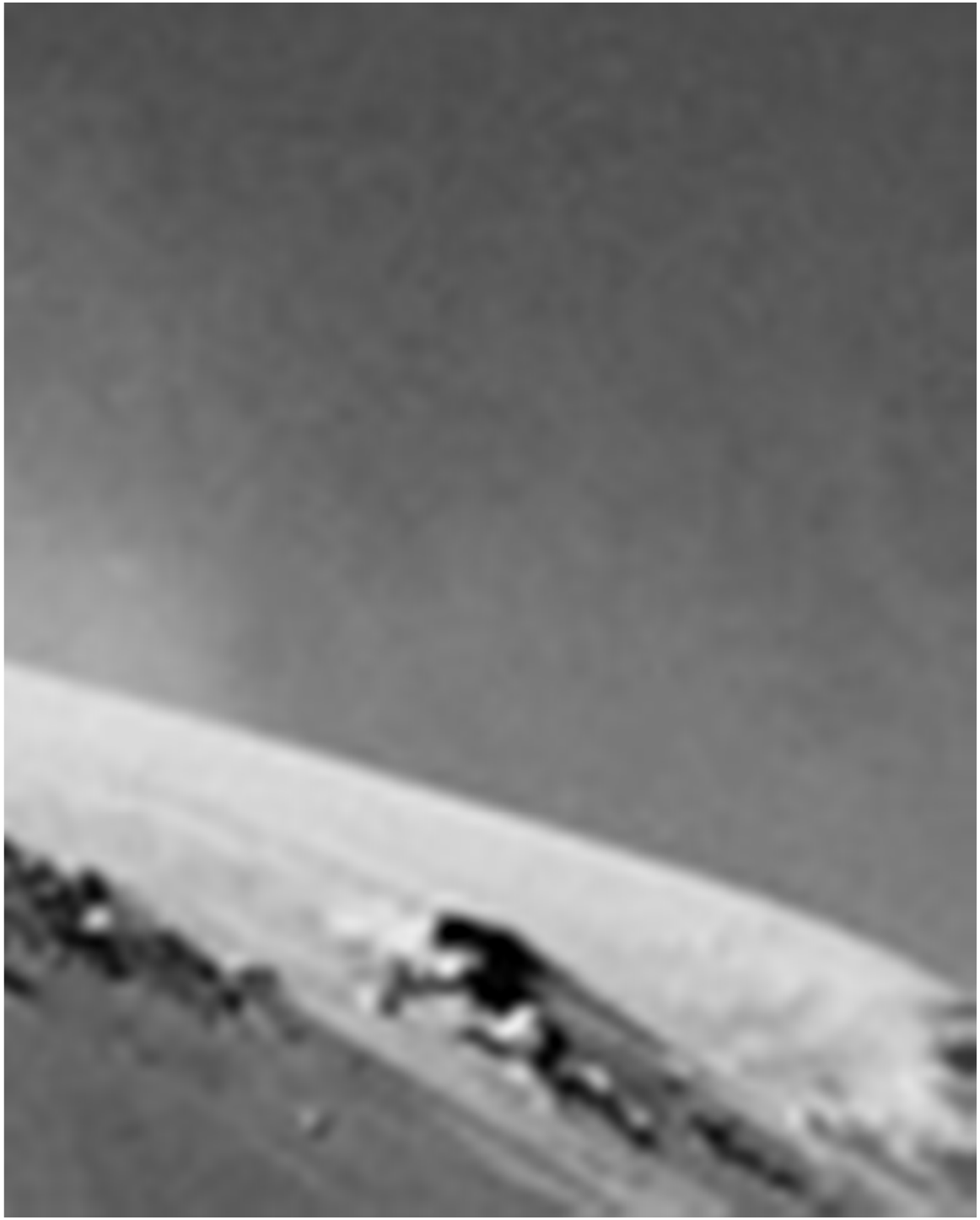
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

```
>> noise = randn(size(im)).*sigma;

>> output = im + noise;
```


Fig: M. Hebert

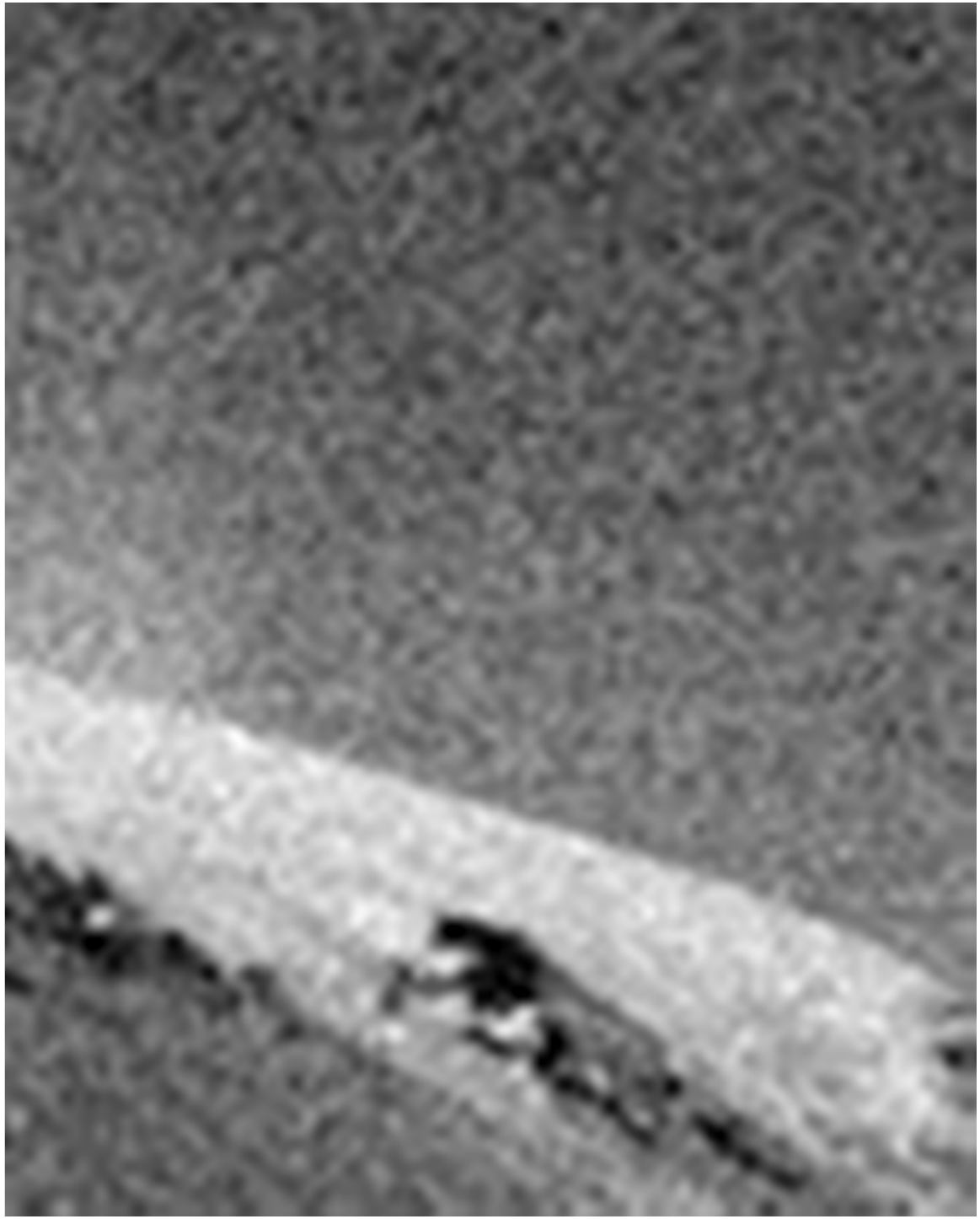
sigma=1



Effect of sigma on Gaussian noise:

This shows the noise values added to the raw intensities of an image.

sigma=16



Effect of sigma on Gaussian noise

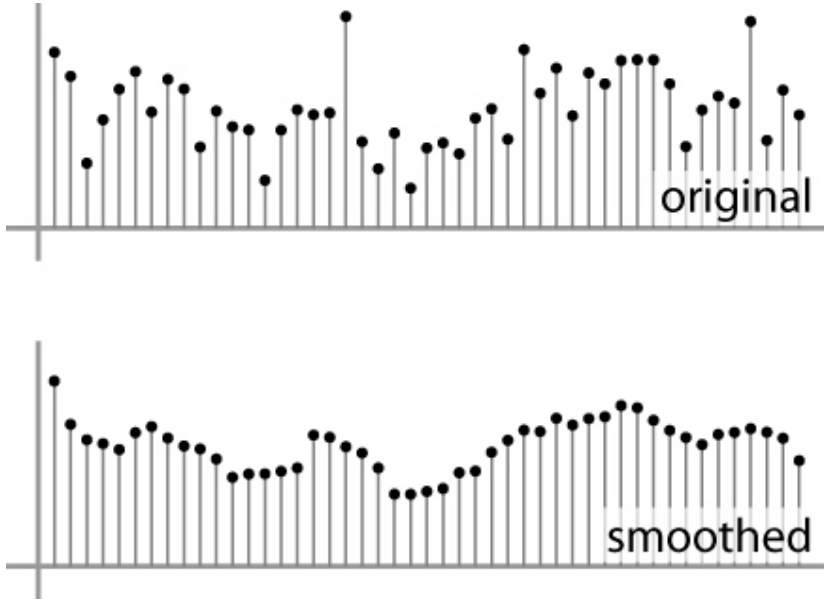
This shows the noise values added to the raw intensities of an image.

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

First attempt at a solution

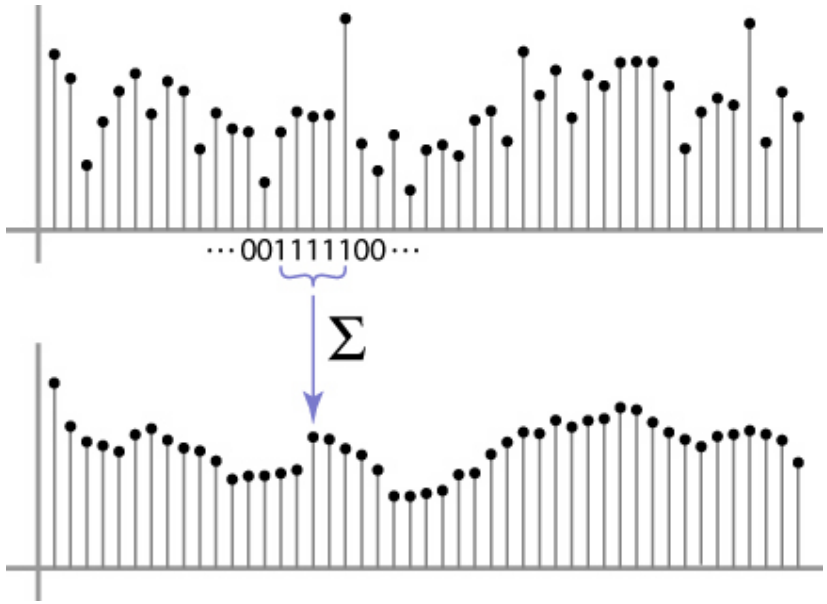
- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



Source: S. Marschner

Weighted Moving Average

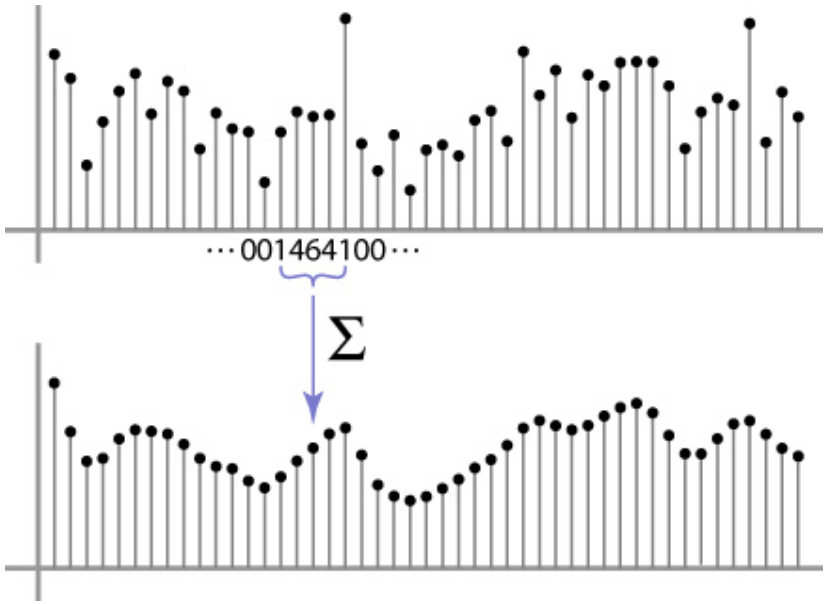
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Source: S. Marschner

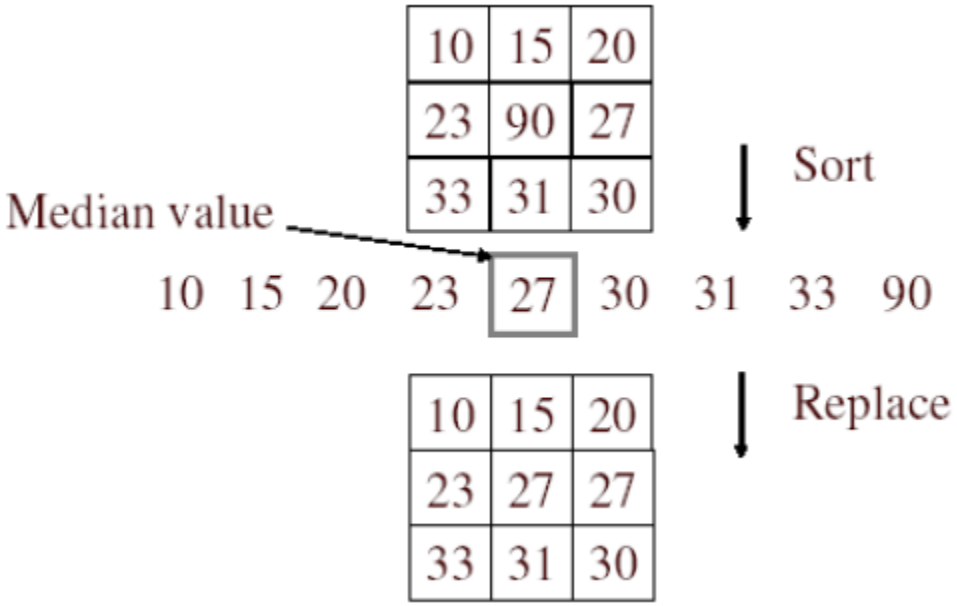
Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

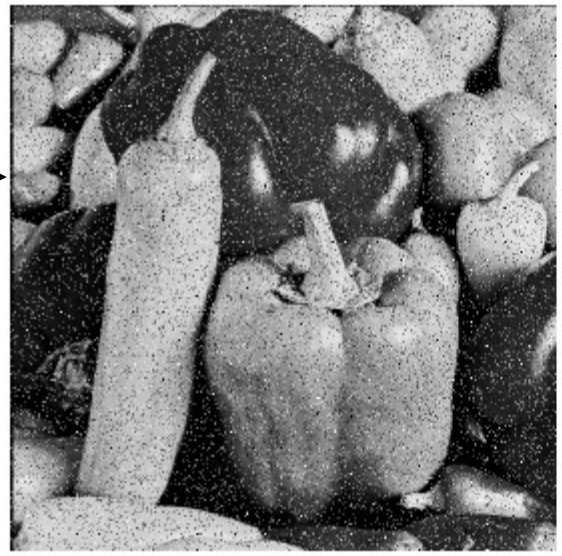
Median filter



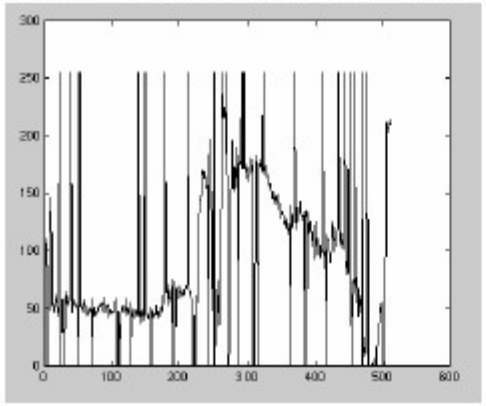
- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise

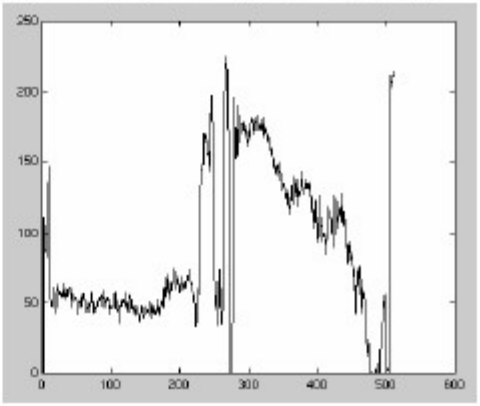
Median filter

Salt and pepper noise



Median filtered



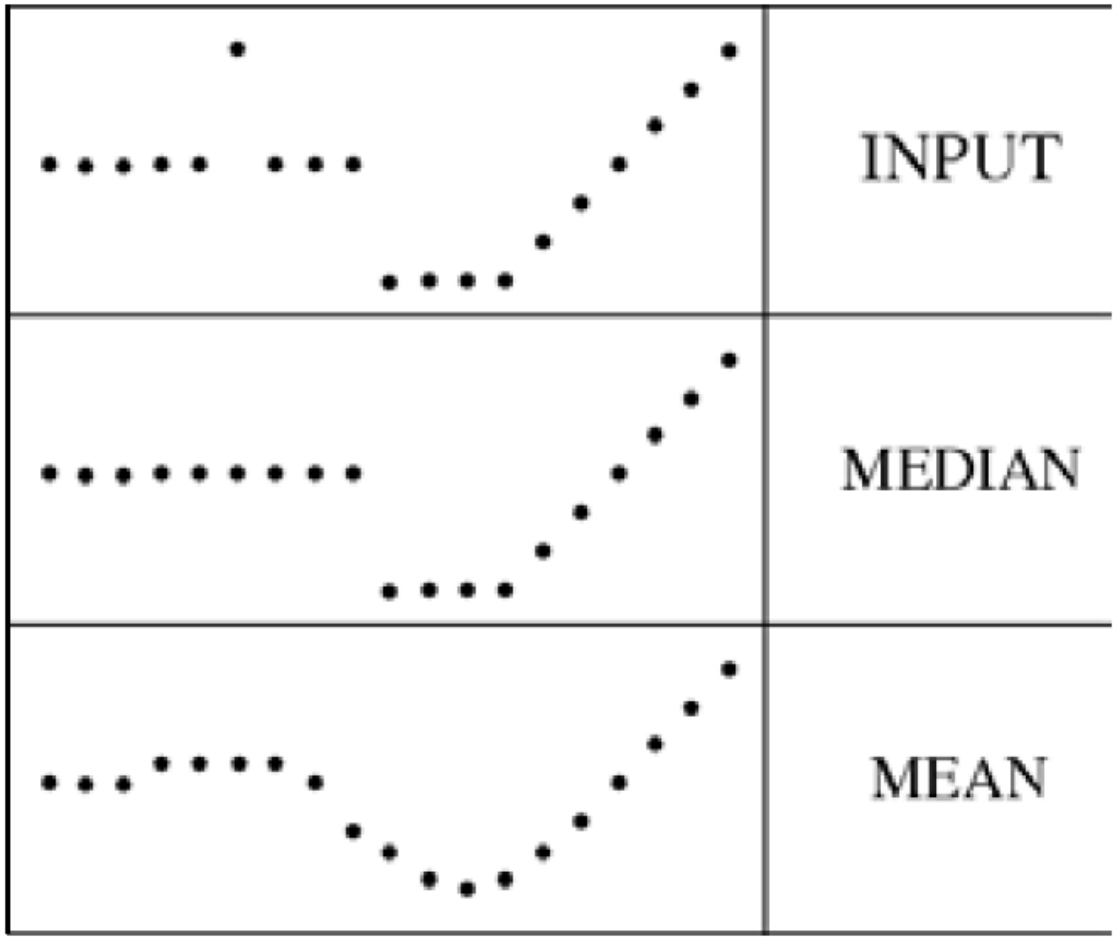


Plots of a row of the image

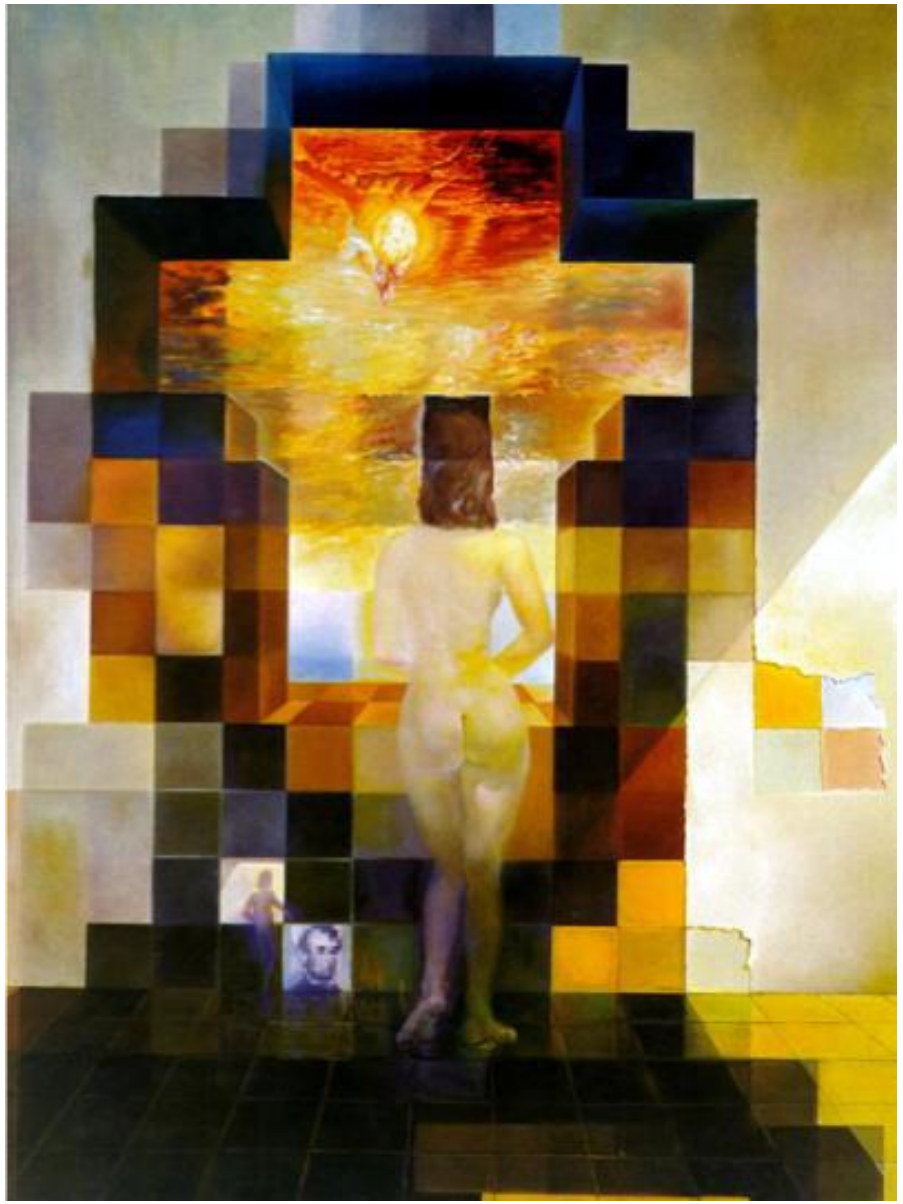
Source: M. Hebert

Median filter

- Median filter is edge preserving



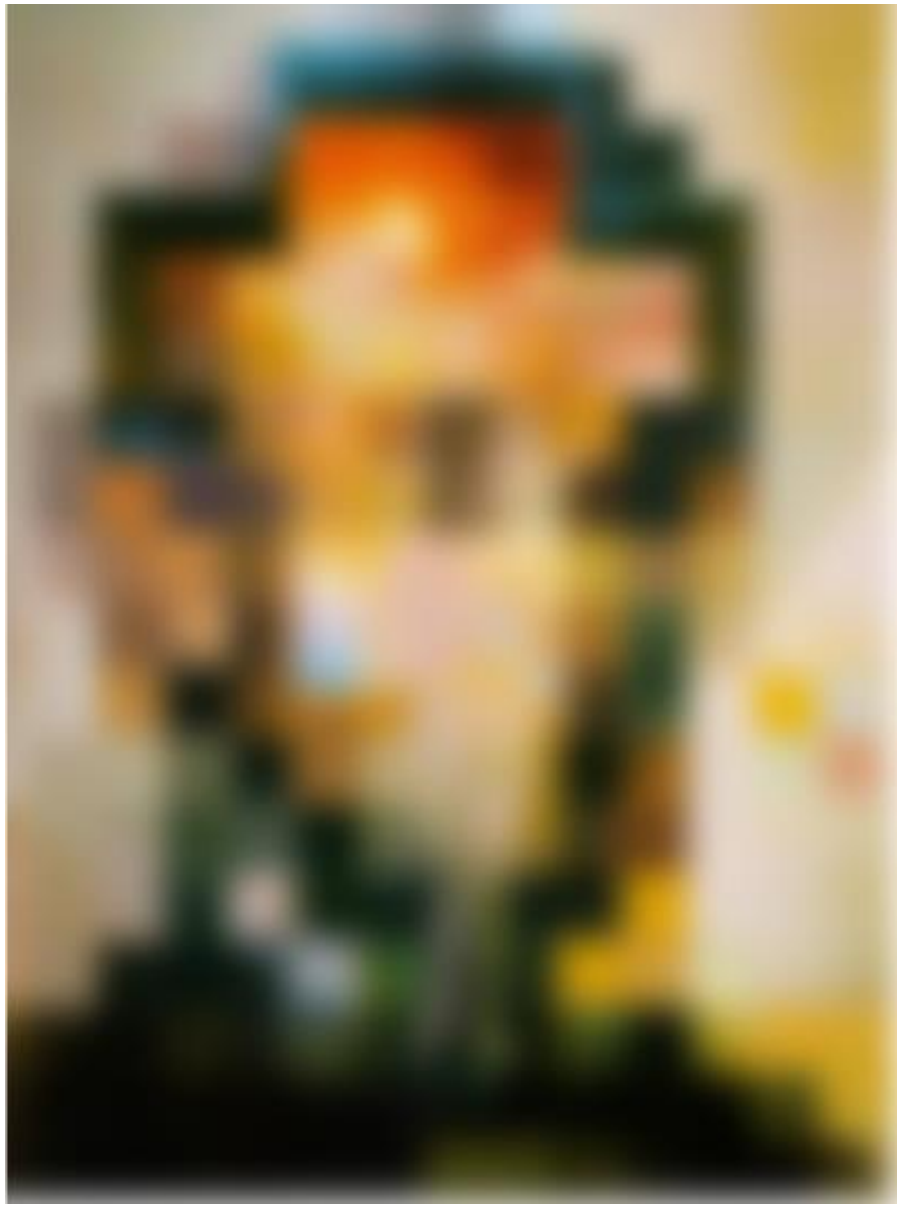
Frequency-domain filtering in human vision



Gala Contemplating the Mediterranean Sea Which at Twenty Meters Becomes the Portrait of Abraham Lincoln (Homage to Rothko)

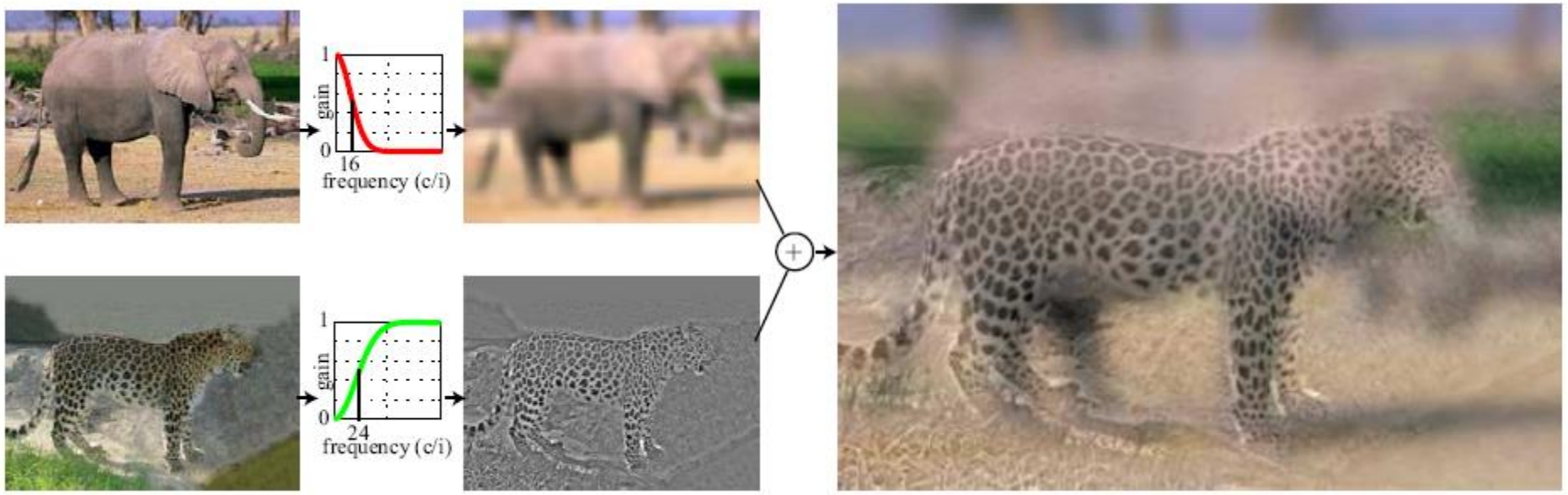
Salvador Dali, 1976

Frequency-domain filtering in human vision



Low-pass filtered version

Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Still used extensively

foreground details enhanced, background details reduced

input image

user-provided mask

Why does a lower resolution image still make sense to us? What information do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/

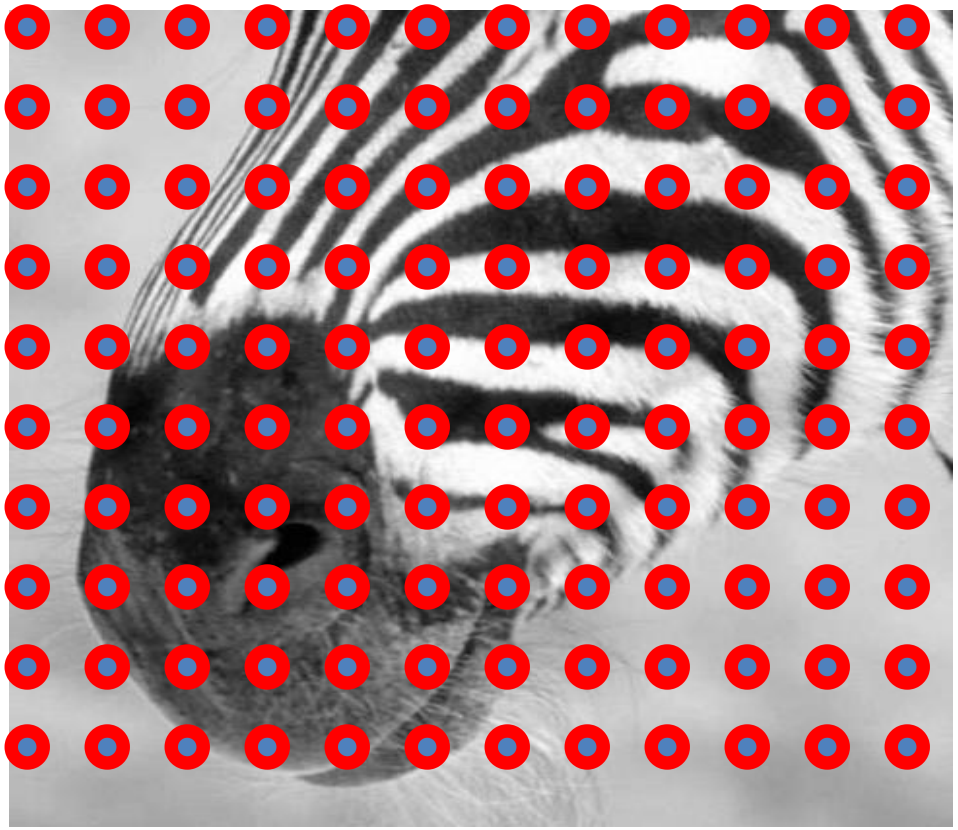
Slide: Hoiem

Sampling

Why does a lower resolution image still make sense to us? What do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/

Subsampling by a factor of 2



Throw away every other row and column to create a 1/2 size image

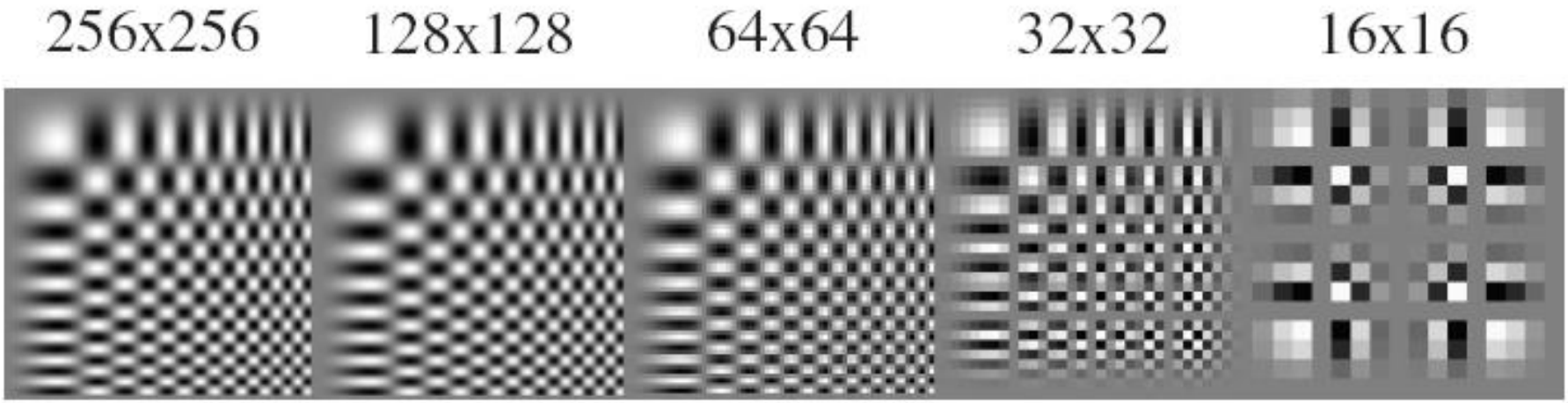
Algorithm for downsampling by factor of 2

1. Start with image of w x h
2. Sample every other pixel
   - im_small = image[::2:, ::2]
3. Repeat until im_small is 1 pixel large.

Numpy syntax:
::2 -> start at 0, end at 'end', increase every 2, until the end.
e.g.,
0,2,4,6,…,w

(if w is not even, then this goes to w-1)

Sampling and aliasing



Image

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

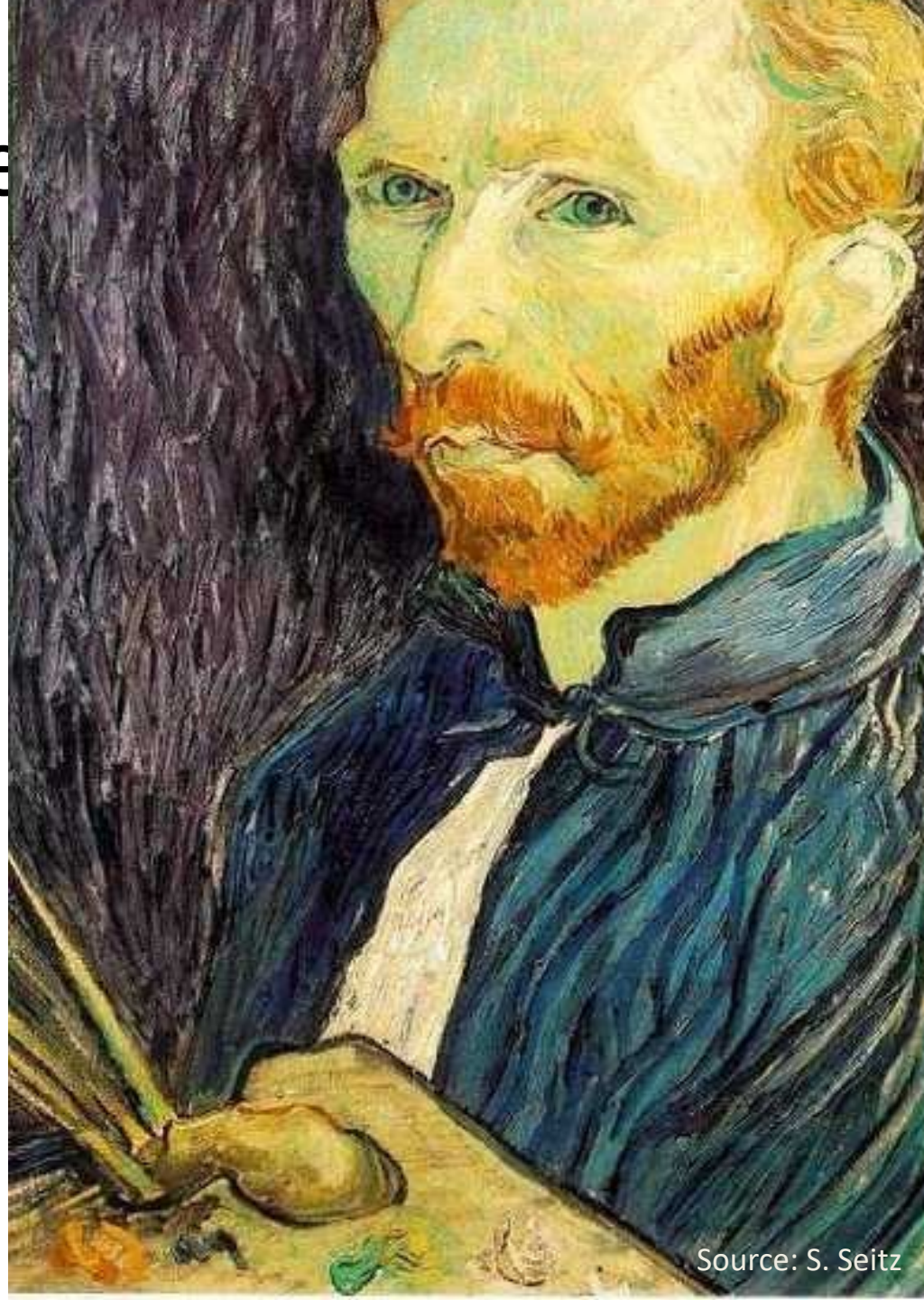
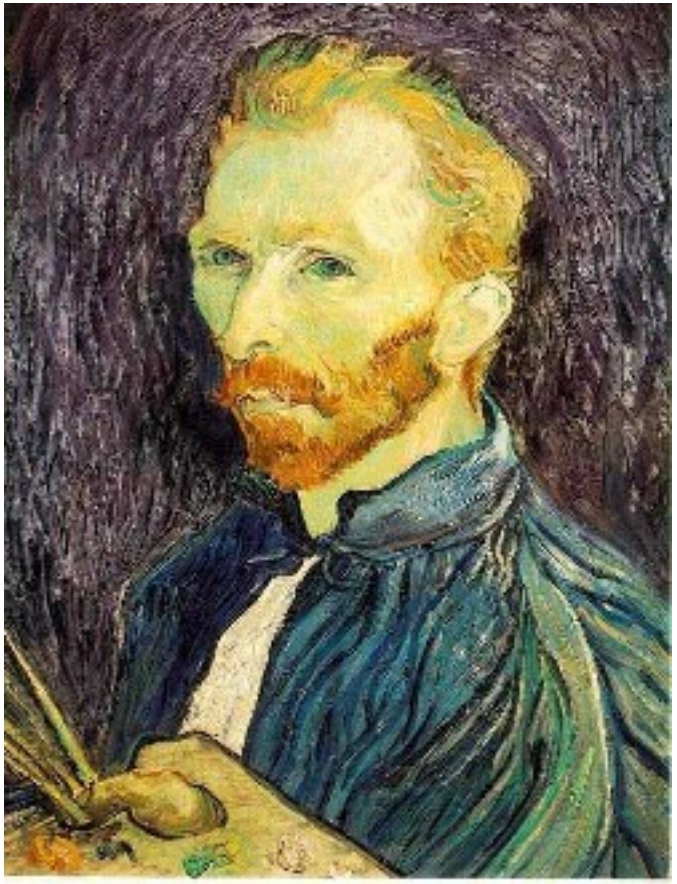


Image sub-sampling



1/4

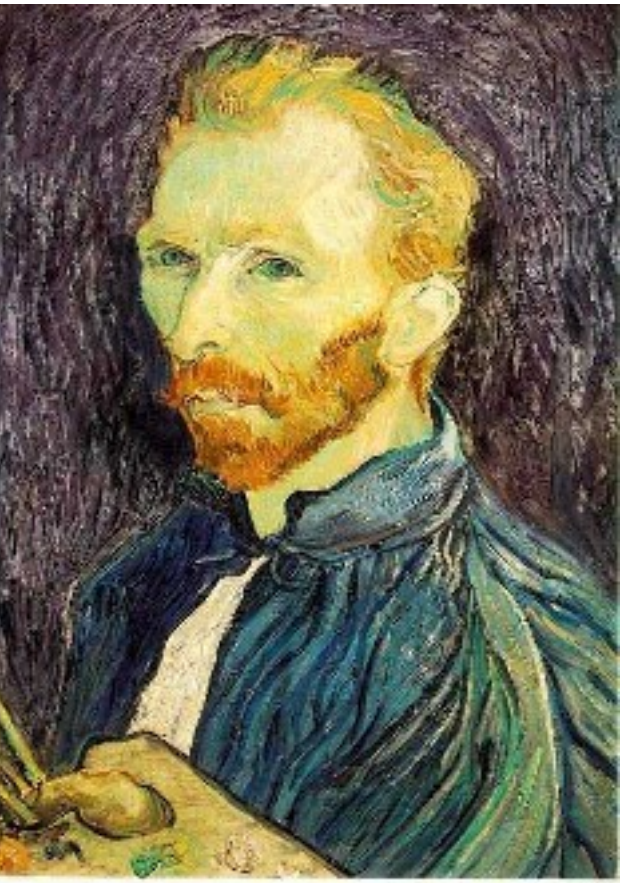
1/8

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)

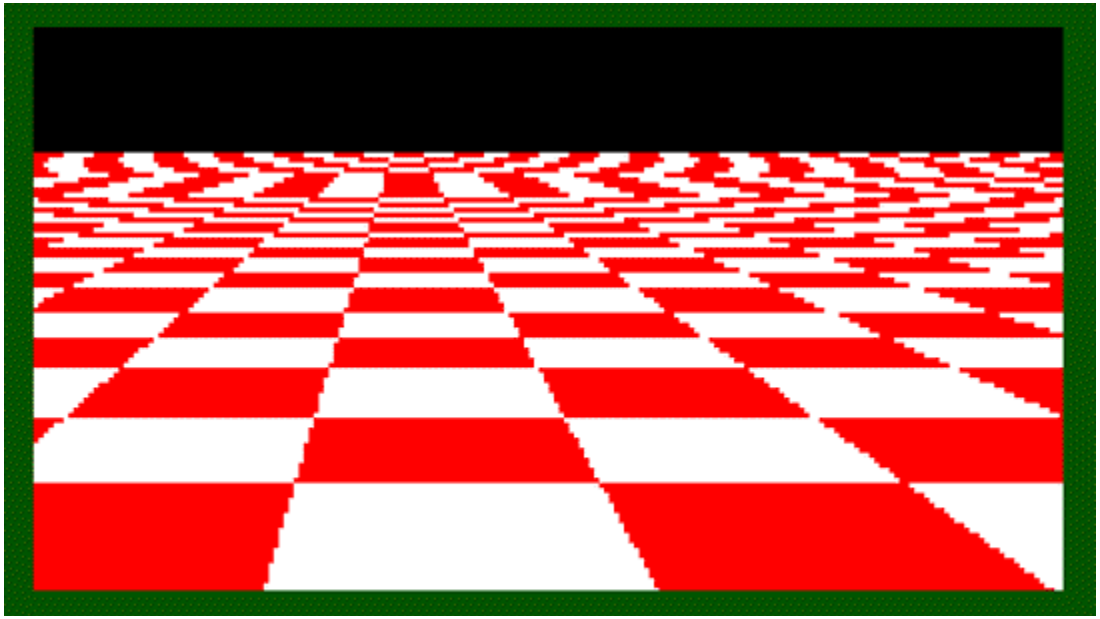
Why does this look so crufty?

Source: S. Seitz

Image sub-sampling – another example

Source: F. Durand

Even worse for synthetic images



Source: L. Zhang

Better sensors

Solutions:

- Sample more often

Anti-aliasing

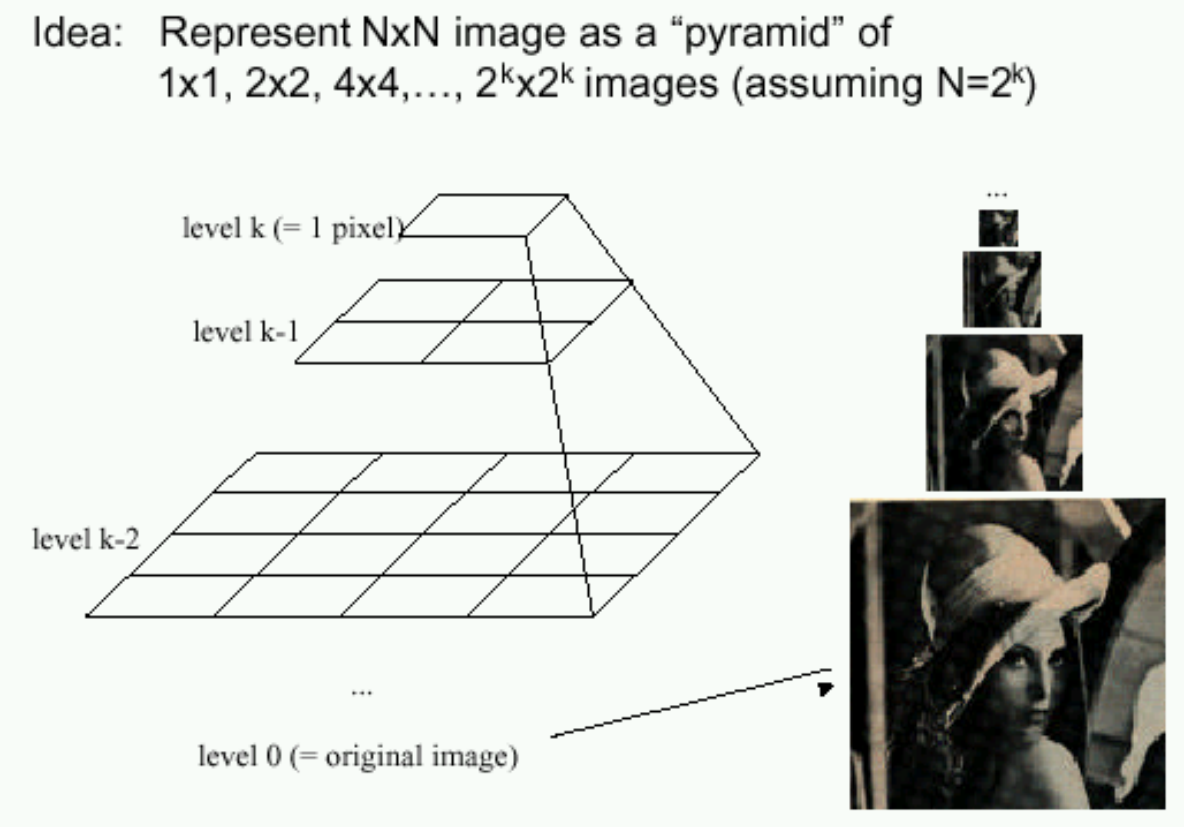
Solutions:

- Sample more often

- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it's better than aliasing
  - Apply a smoothing (*low pass*) filter

Gaussian image pyramid

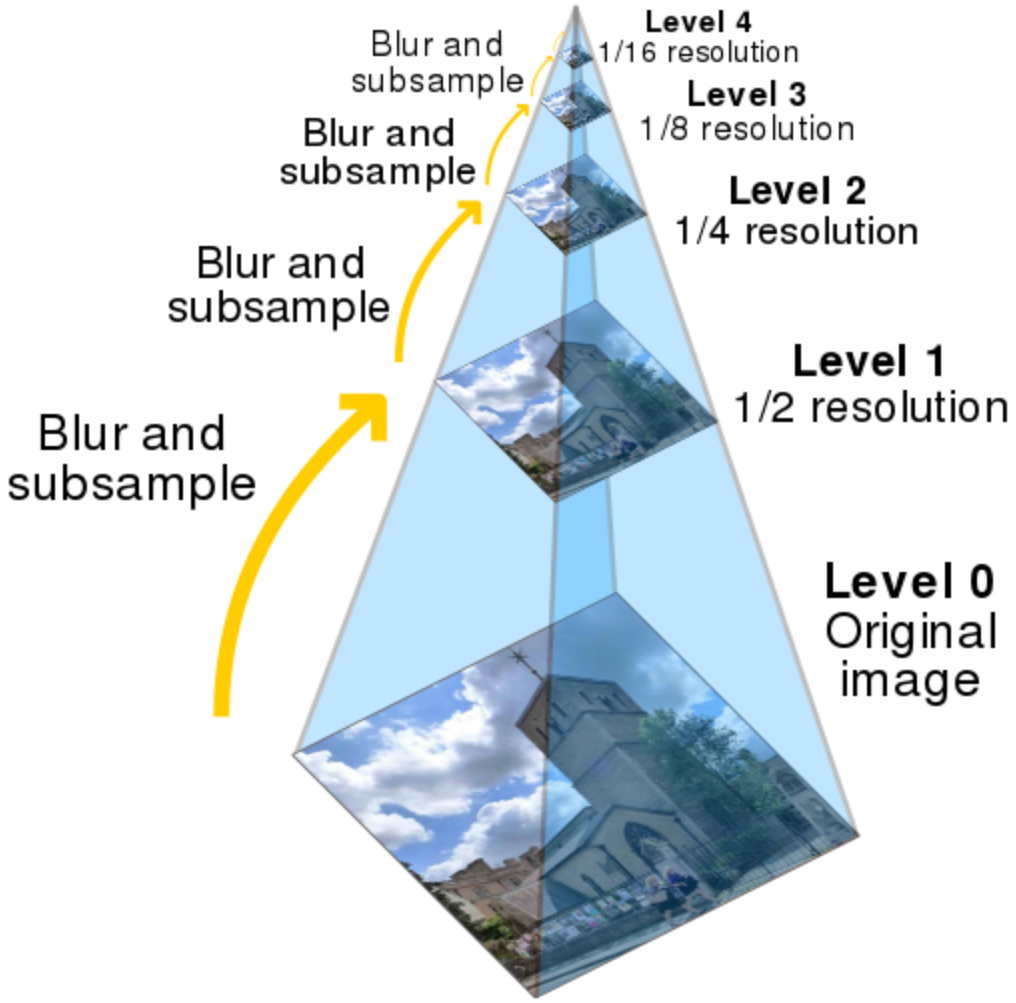
Gaussian pyramids [Burt and Adelson, 1983]



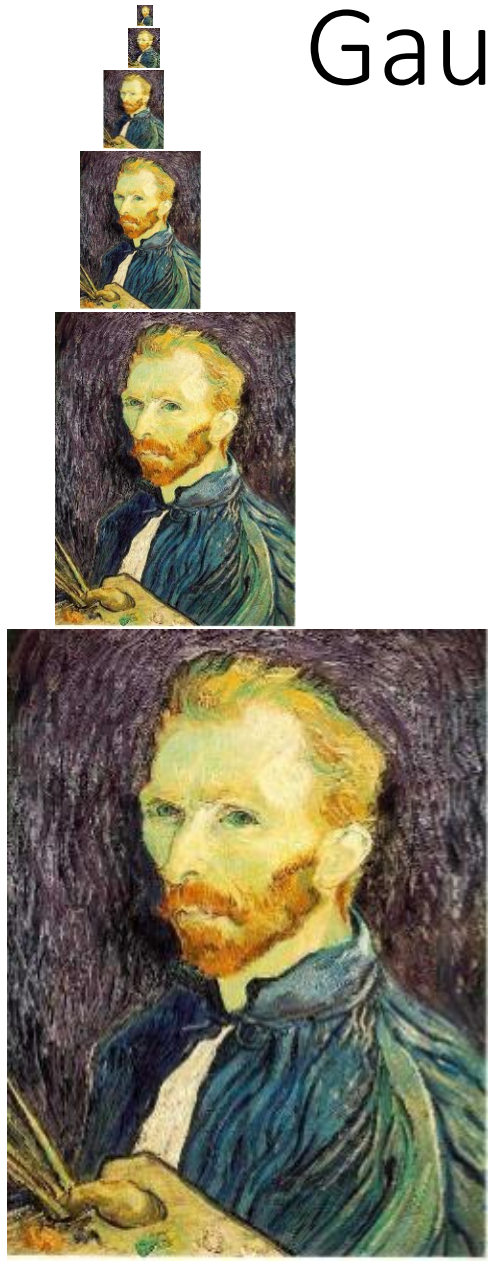
- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz


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

Gaussian image pyramid



The name of this sequence of subsampled images

Constructing a Gaussian pyramid

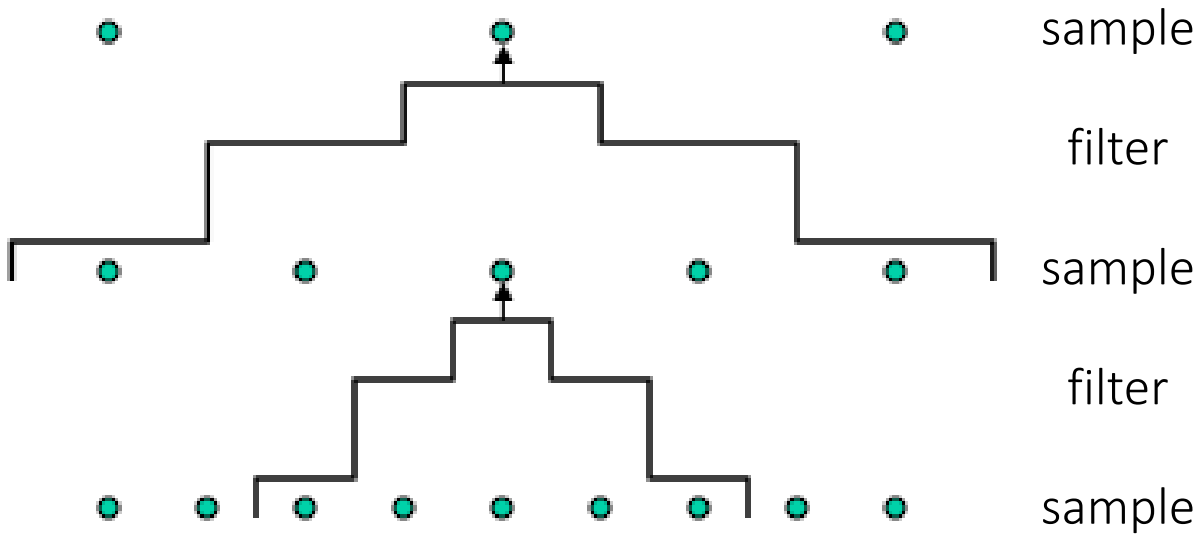
Algorithm

repeat:

filter

subsample

until min resolution reached



Question: How much bigger than the original image is the whole pyramid?

Constructing a Gaussian pyramid

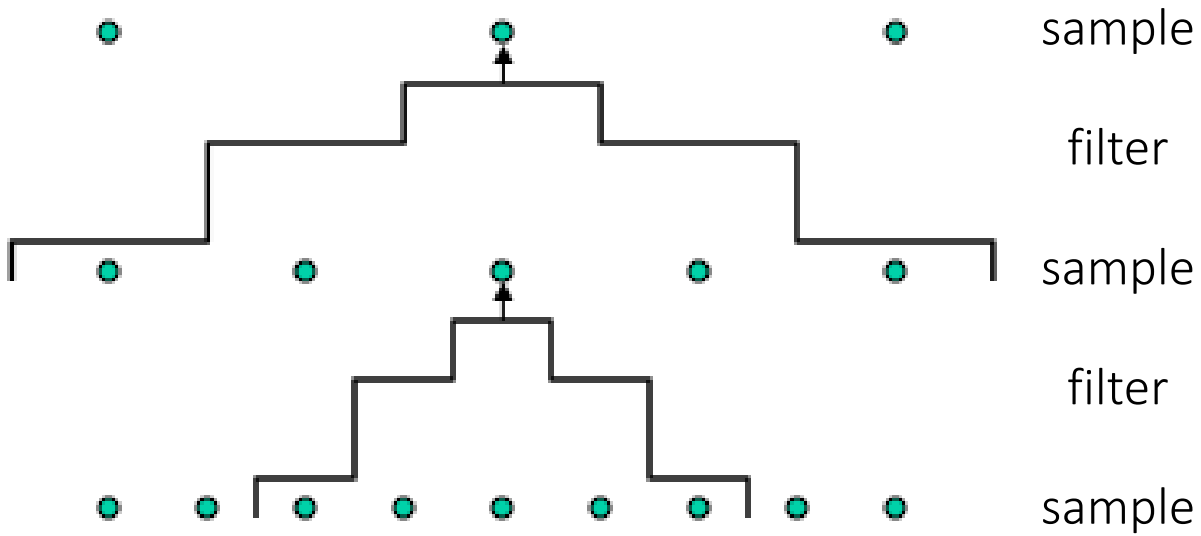
Algorithm

repeat:

filter

subsample

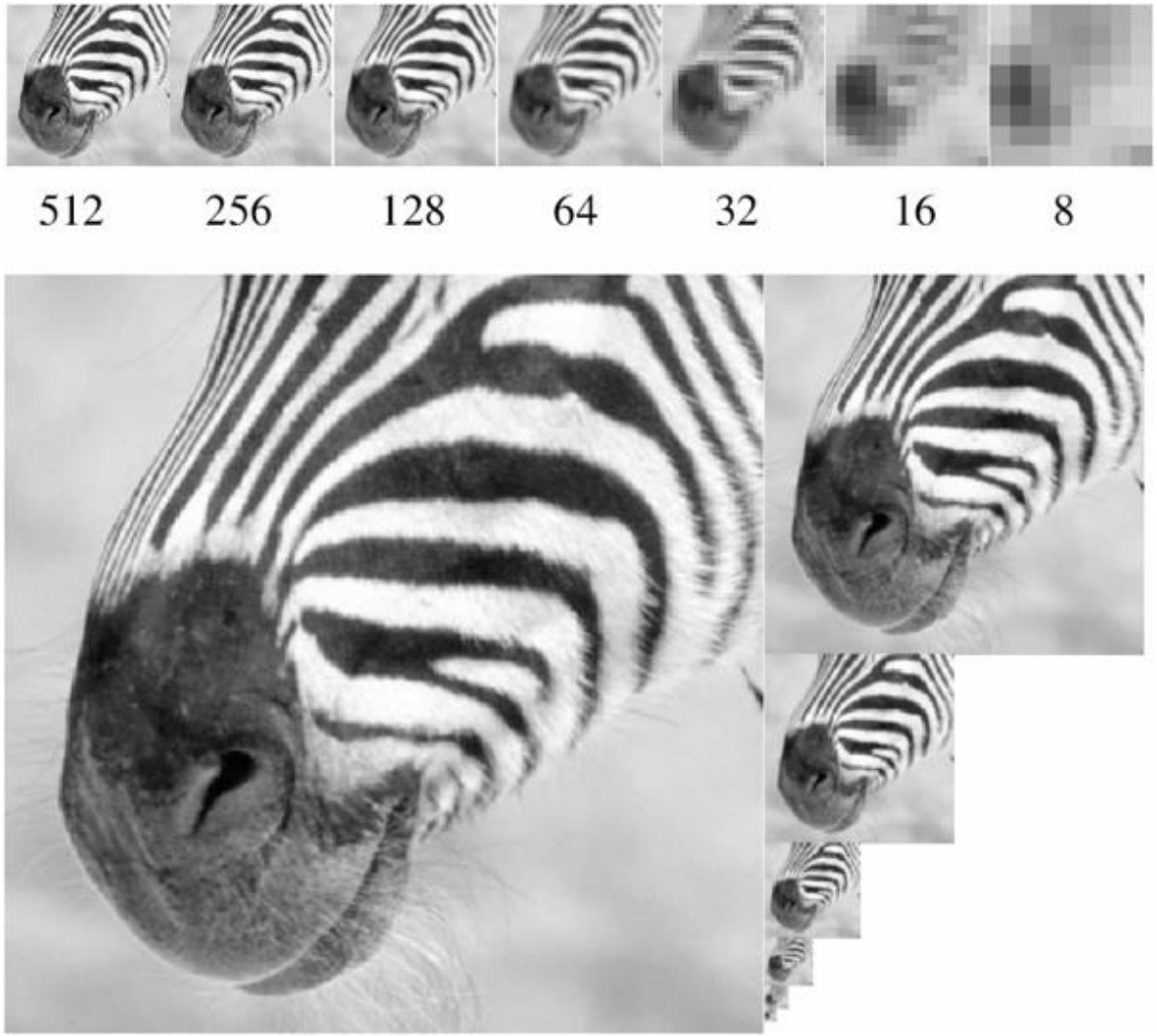
until min resolution reached



Question: How much bigger than the original image is the whole pyramid?

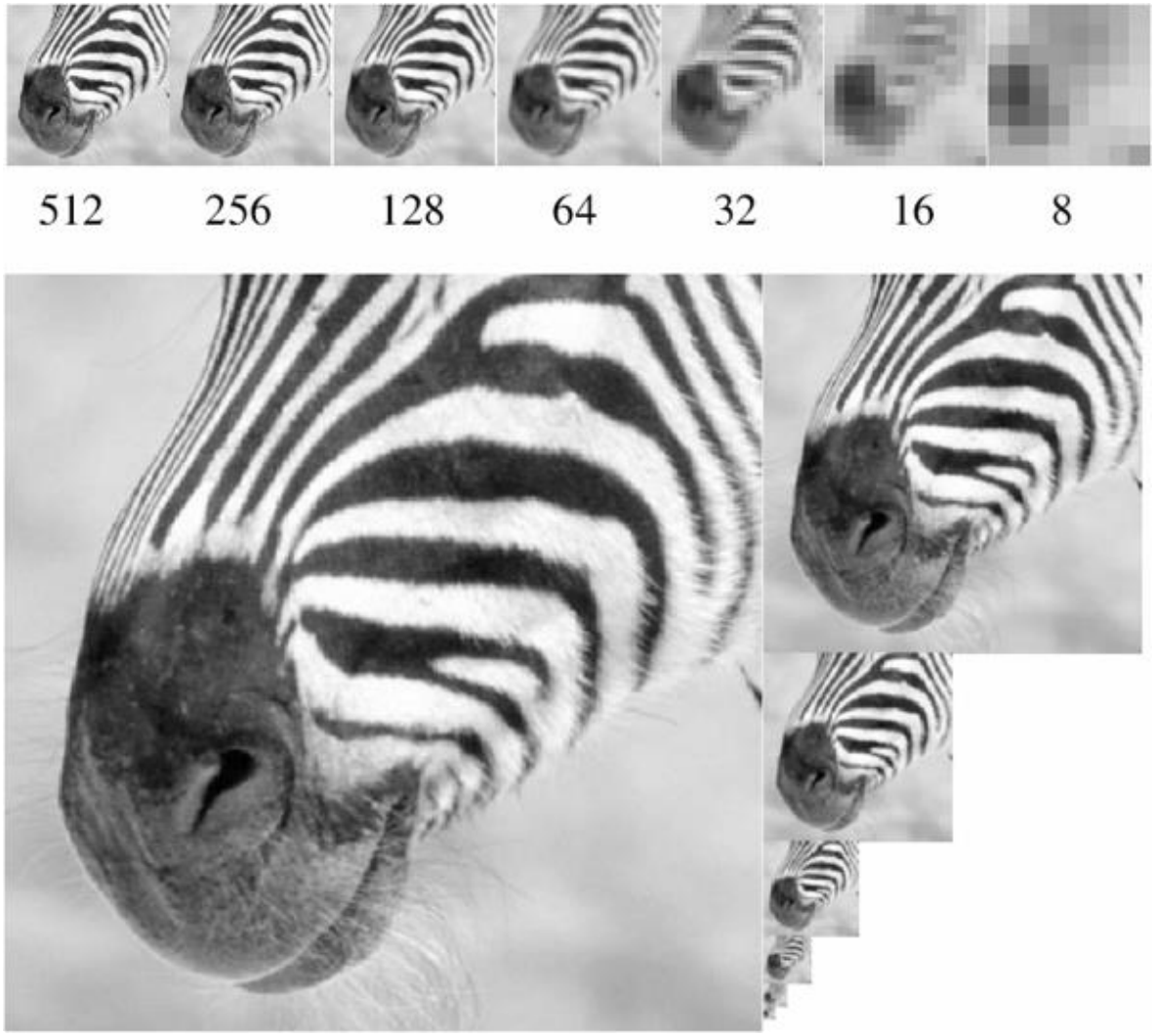
Answer: Just 4/3 times the size of the original image! (How did I come up with this number?)

Some properties of the Gaussian pyramid



What happens to the details of the image?

Some properties of the Gaussian pyramid

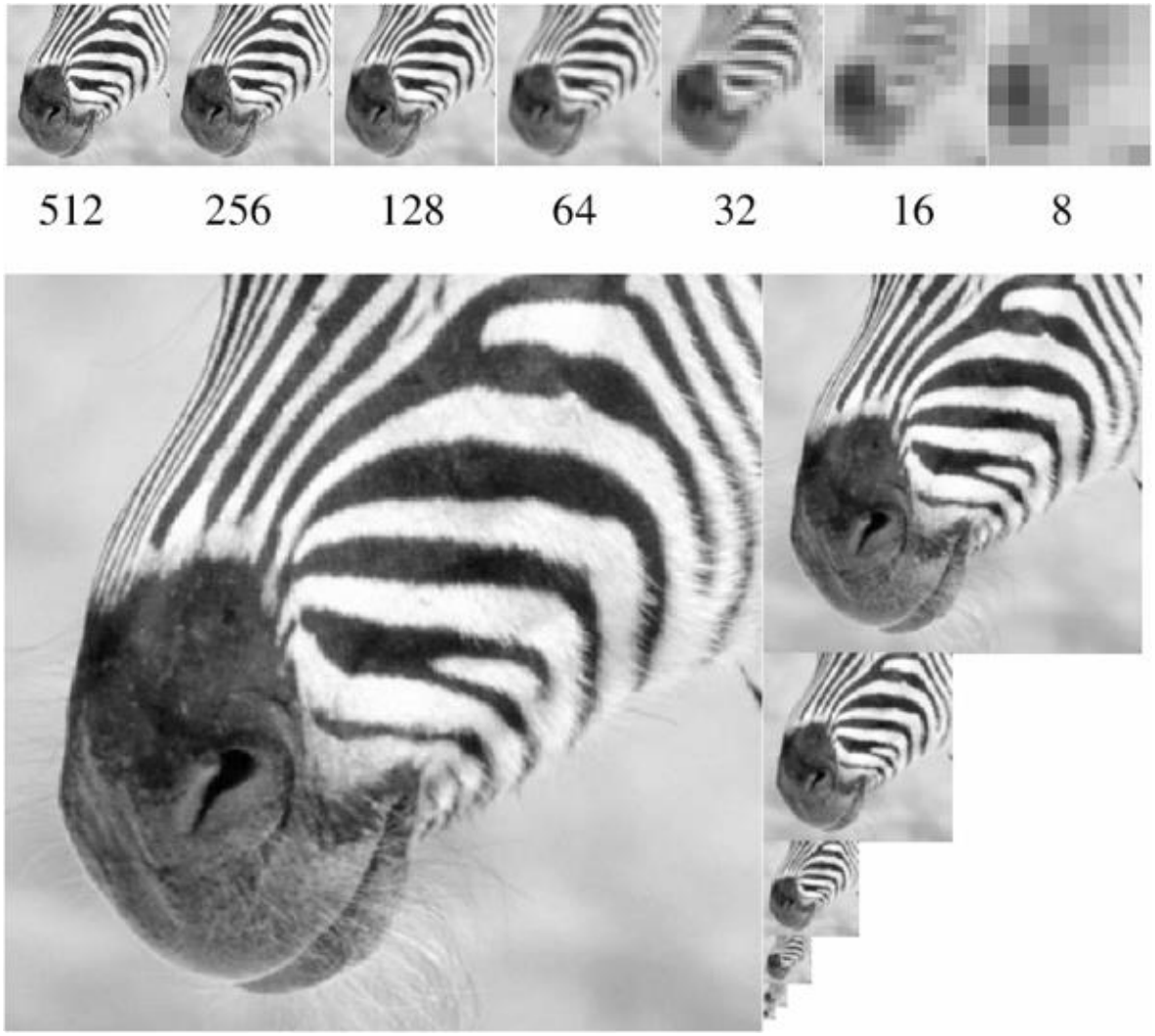


What happens to the details of the image?

- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

Some properties of the Gaussian pyramid



What happens to the details of the image?

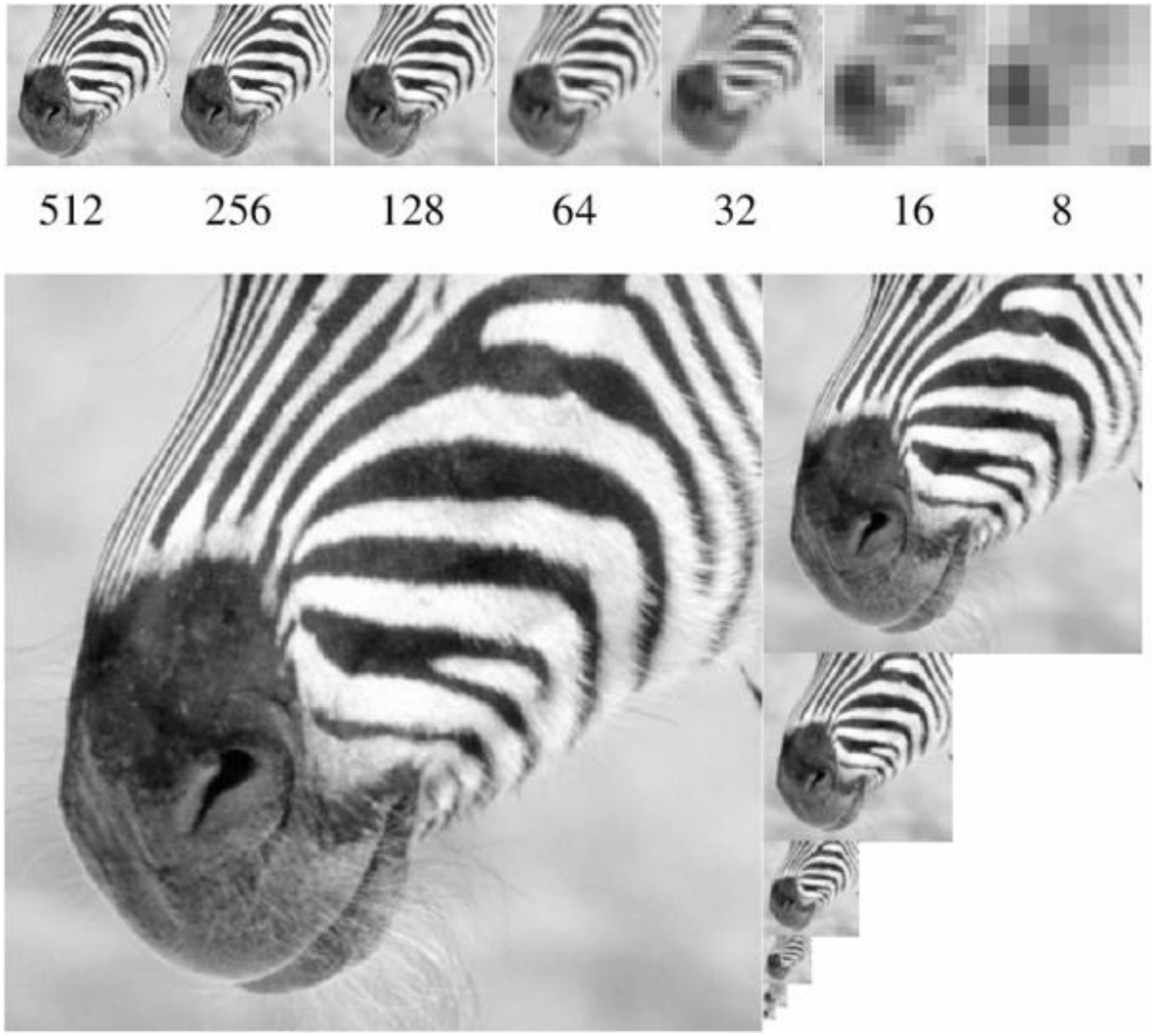
- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

- Mostly large uniform regions in the original image.

How would you reconstruct the original image from the image at the upper level?

Some properties of the Gaussian pyramid



What happens to the details of the image?

- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

- Mostly large uniform regions in the original image.

How would you reconstruct the original image from the image at the upper level?

- That’s not possible.

Blurring is lossy

level 0

-

level 1 (before downsampling)

=

residual

What does the residual look like?

Blurring is lossy

level 0

-

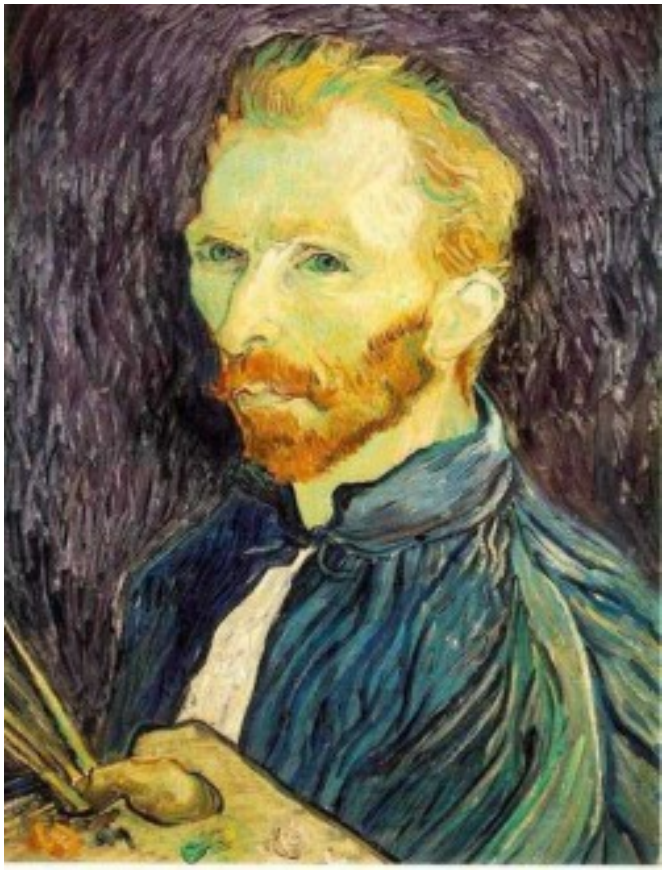
level 1 (before downsampling)

=

residual

Can we make a pyramid that is lossless?

Gaussian pre-filtering



Gaussian 1/2

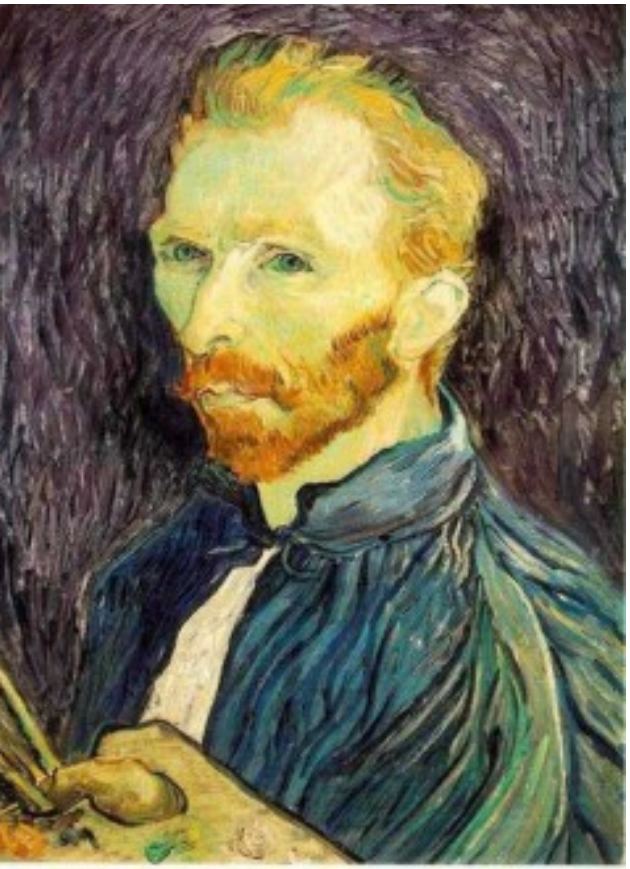
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

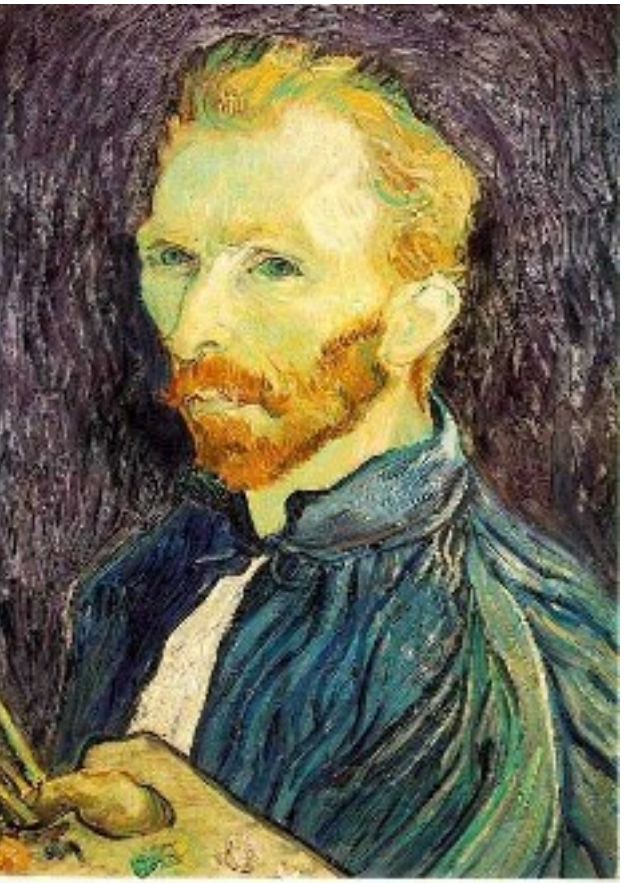
G 1/4

G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Compare with...



1/2

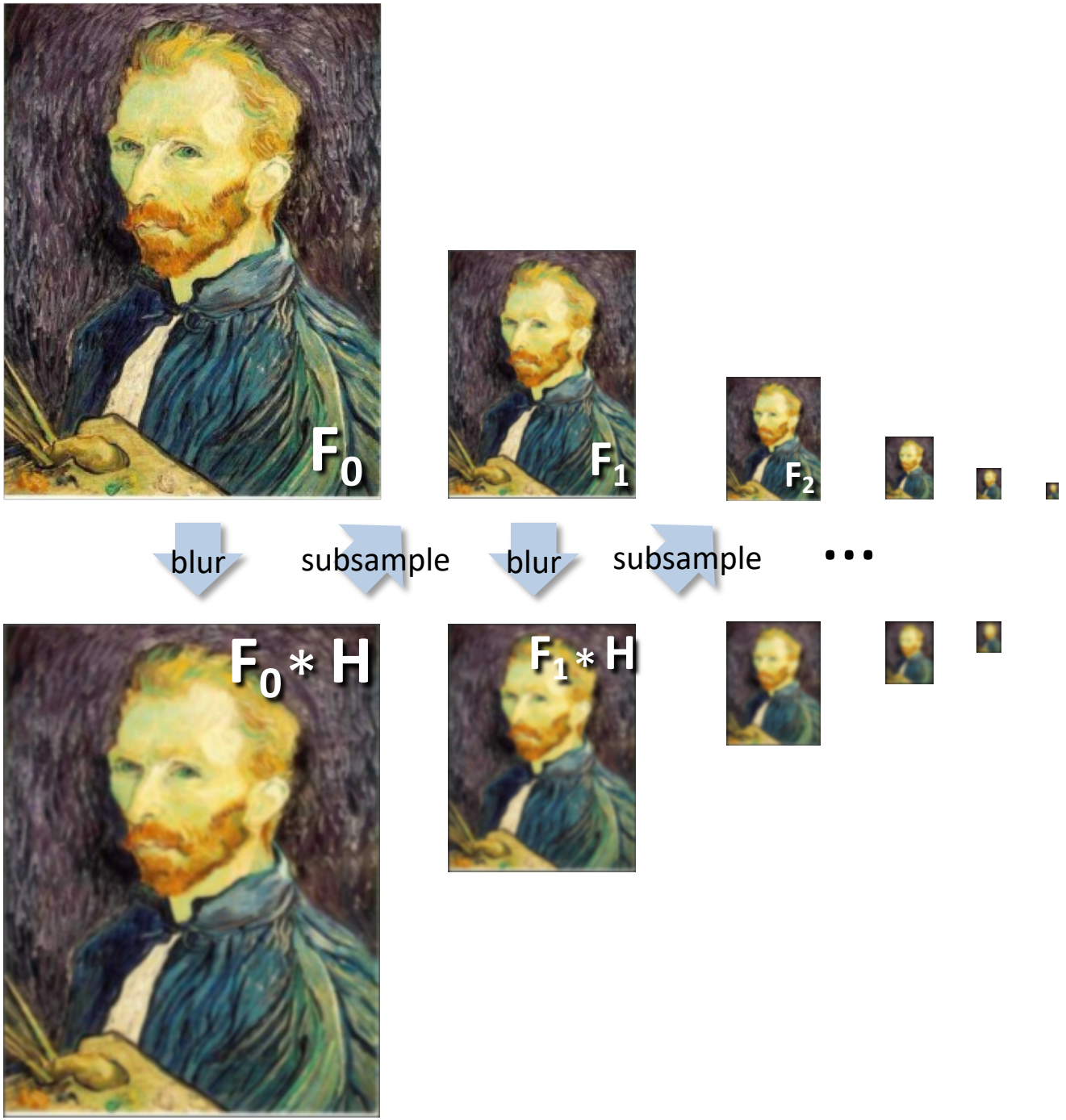
1/4 (2x zoom)

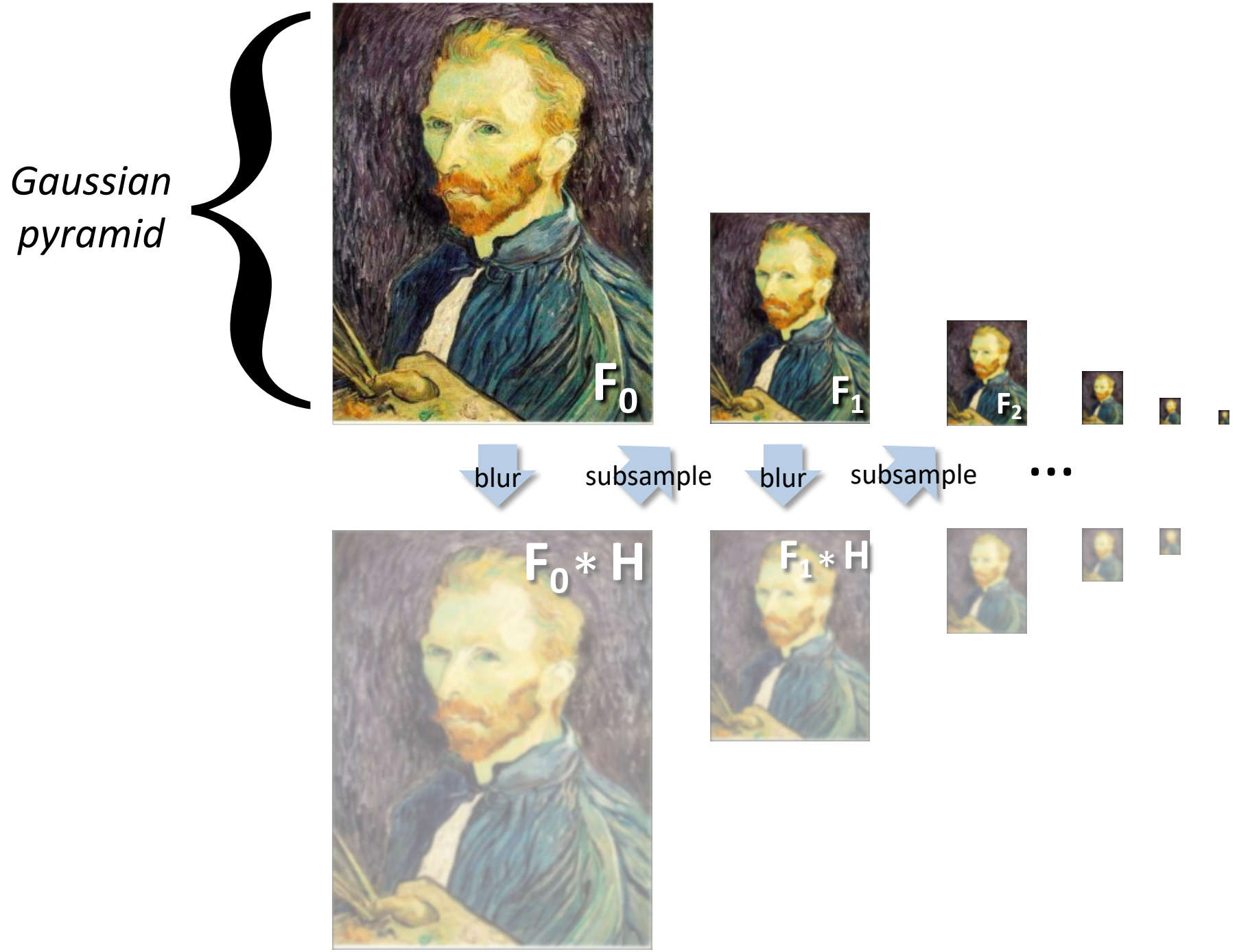
1/8 (4x zoom)

Source: S. Seitz

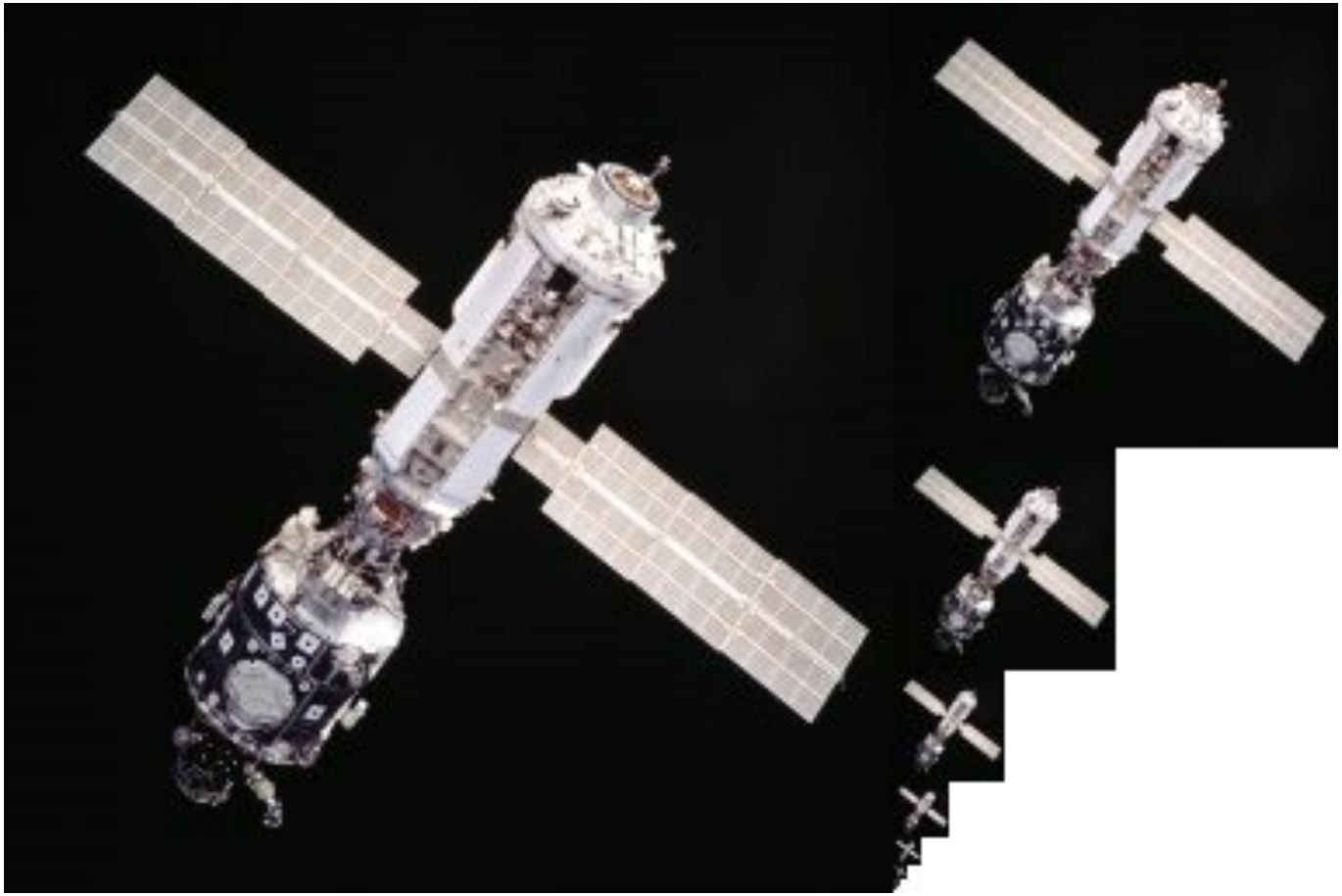
Gaussian pre-filtering

- Solution: filter the image, *then* subsample




Gaussian pyramid
F_0
F_1
F_2
blur
subsample
blur
subsample
• • •
$F_0 * H$
$F_1 * H$

Gaussian Pyramid



What are image pyramids used for?

image compression

multi-scale texture mapping

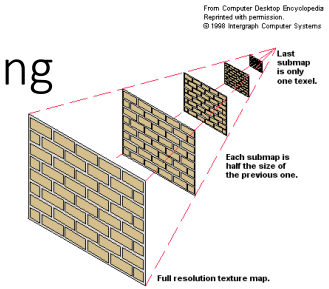
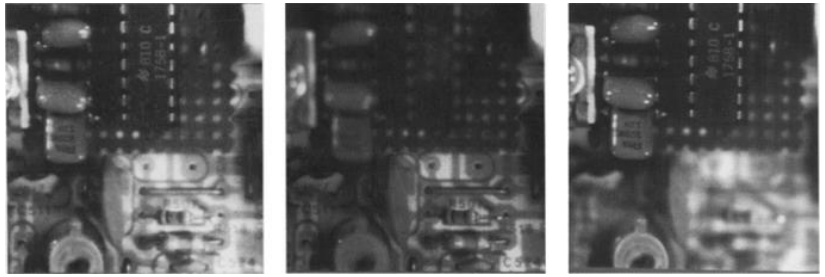
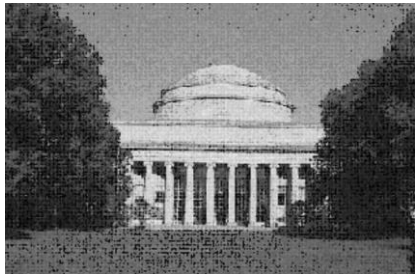


image blending

focal stack compositing



denoising



multi-scale detection

multi-scale registration