Image processing

370: Intro to Computer Vision

Subhransu Maji

February 27 & March 4

College of INFORMATION AND COMPUTER SCIENCES

UMASS AMHERST

Overview of the next two lectures

Digitizing an image

Image processing

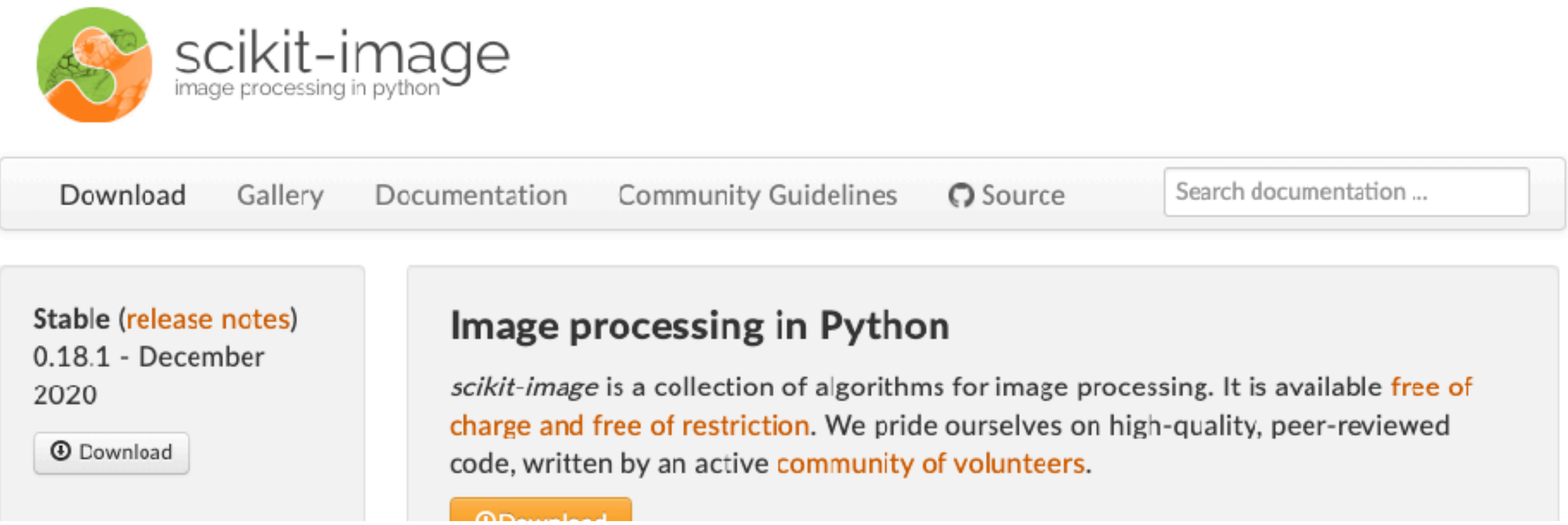
- Example: Improving contrast

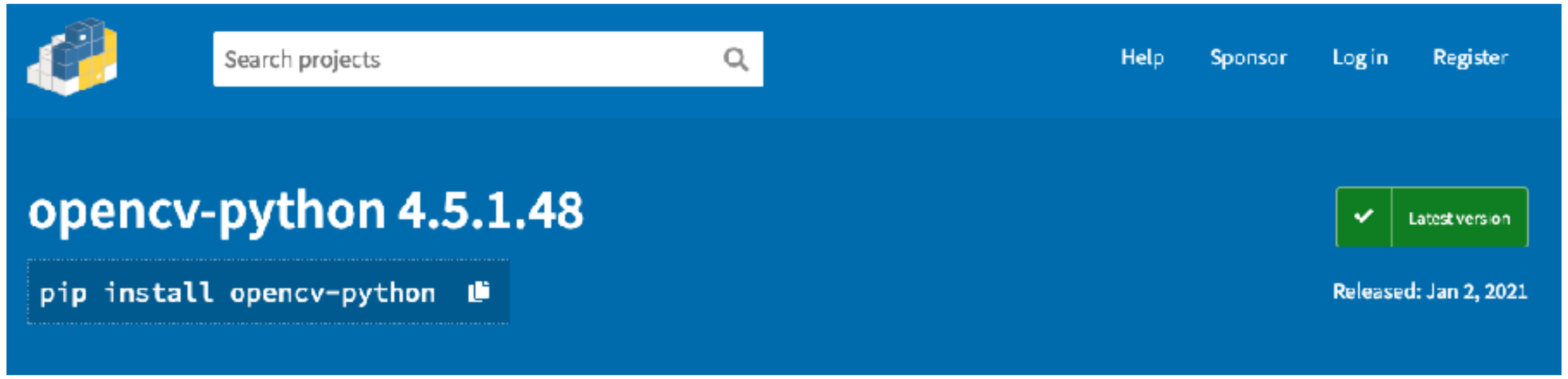
Convolution and filtering

- Mathematical model
- Implementation details

Applications

- Denoising
- Sharpening
- Edge detection





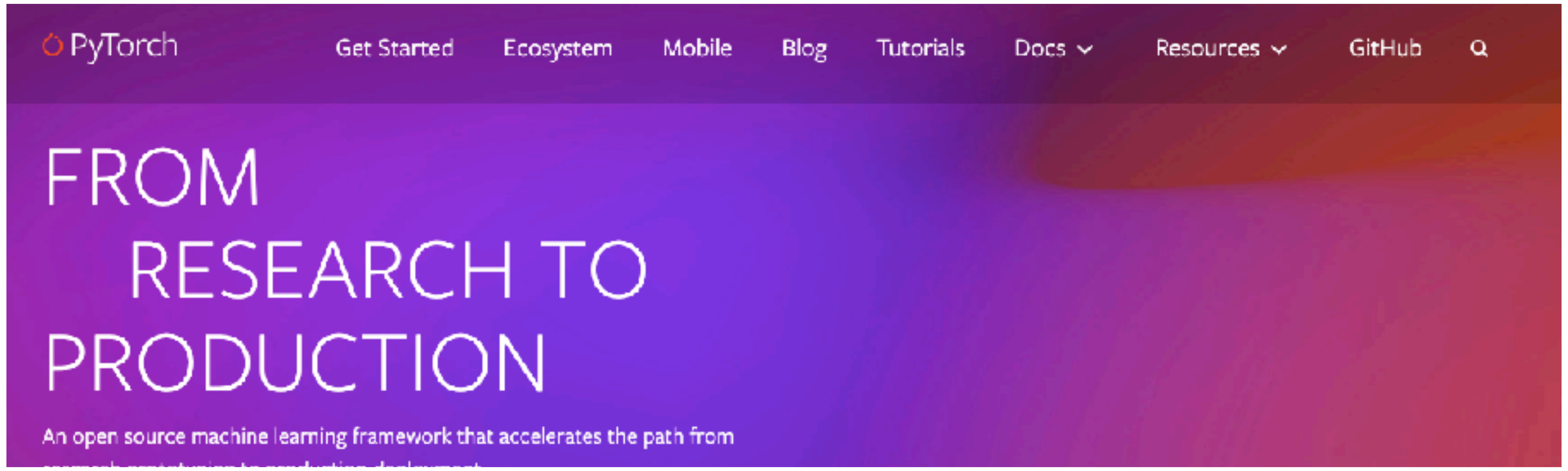
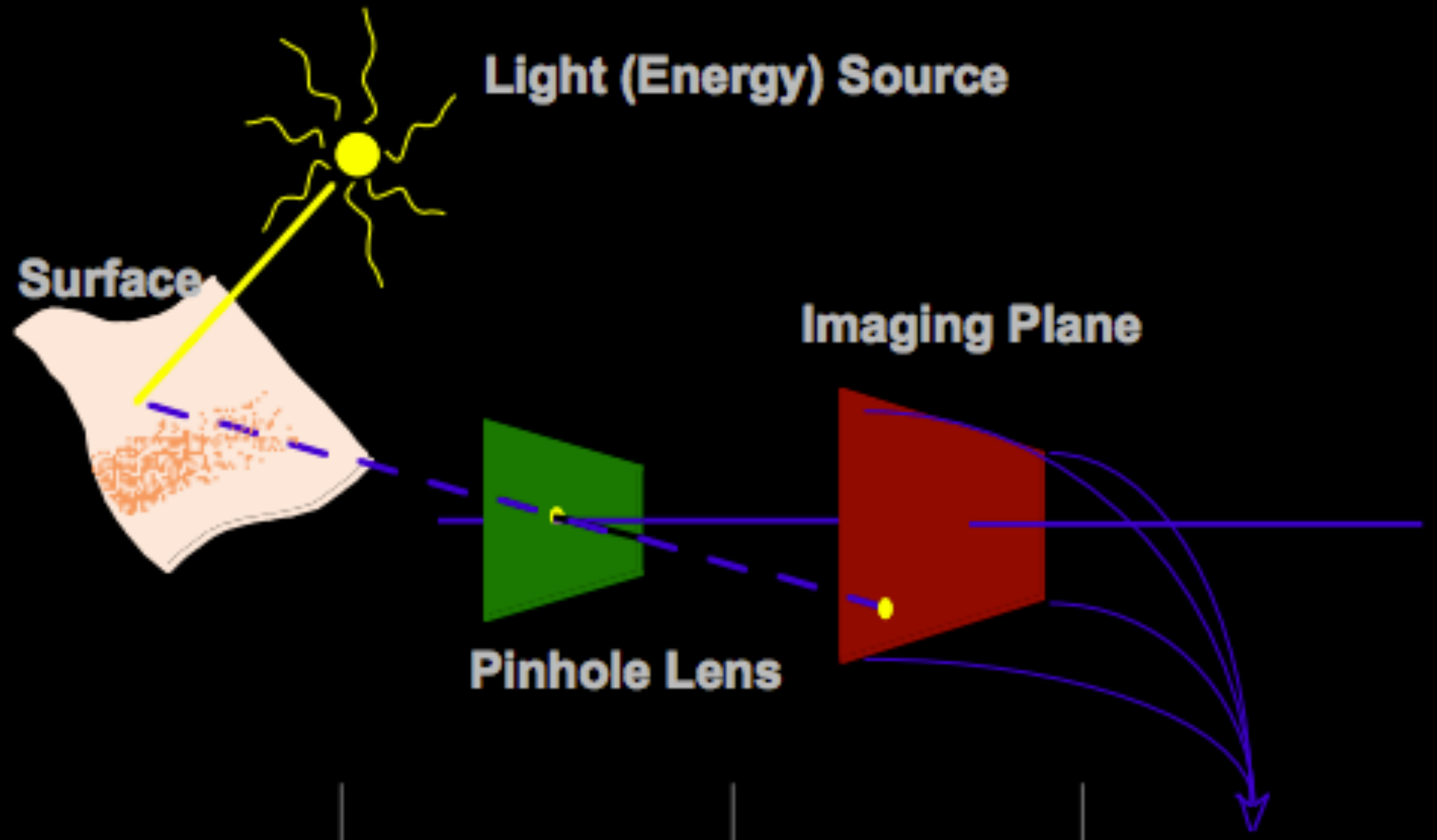


Image formation



World	Optics	Sensor	Signal
		B&W Film	Silver Density
		Color Film	Silver density in three color layers
		TV Camera	Electrical

Pre-digitization image

What is an image before you digitize it?

- Continuous range of wavelengths
- 2-dimensional extent
- Continuous range of power at each point

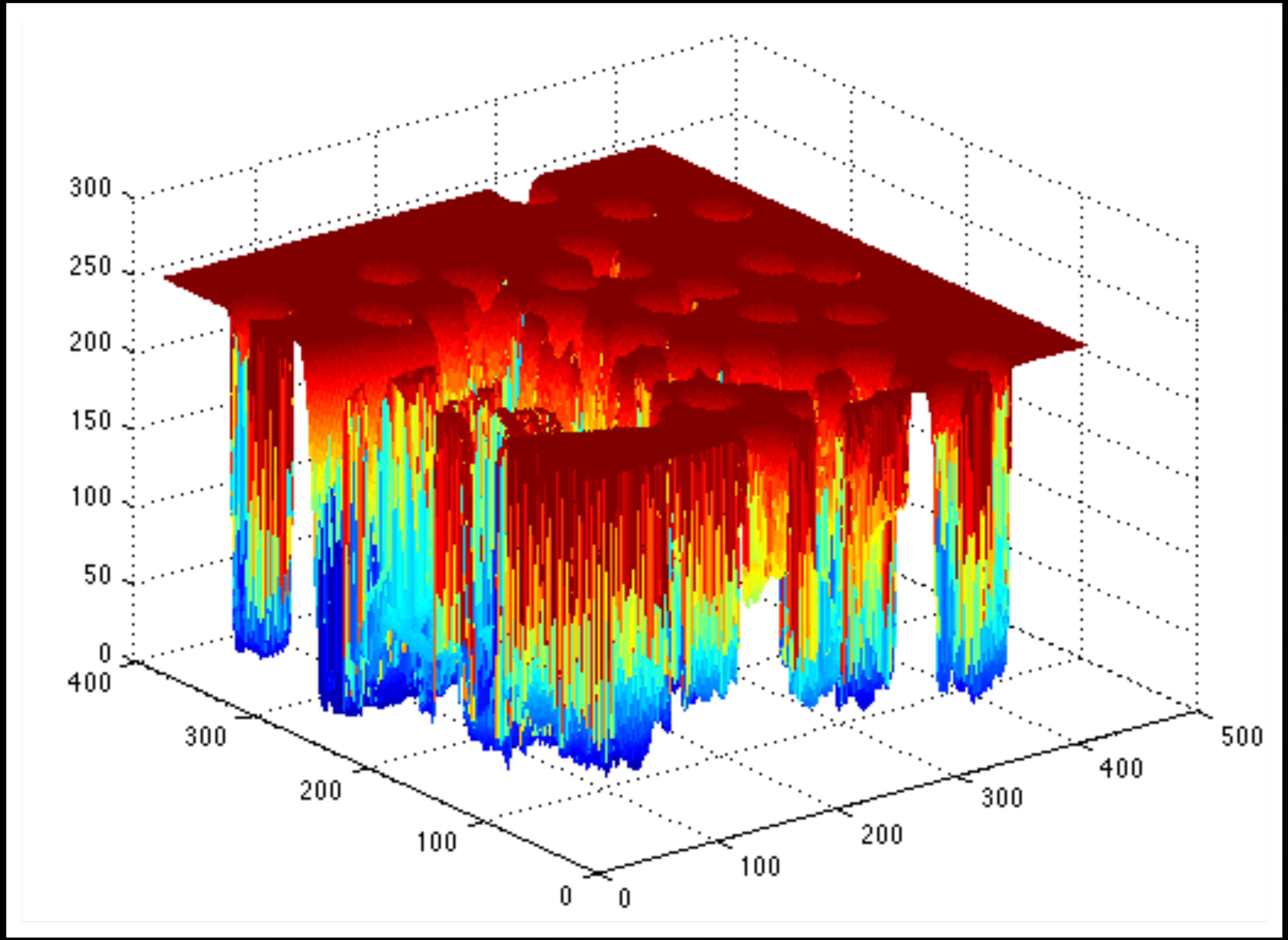
Brightness images

To simplify, consider only a brightness image

- Two-dimensional (continuous range of locations)
- Continuous range of brightness values

This is equivalent to a two-dimensional function over a plane

An image as a surface



How do we represent this continuous two dimensional surface efficiently?

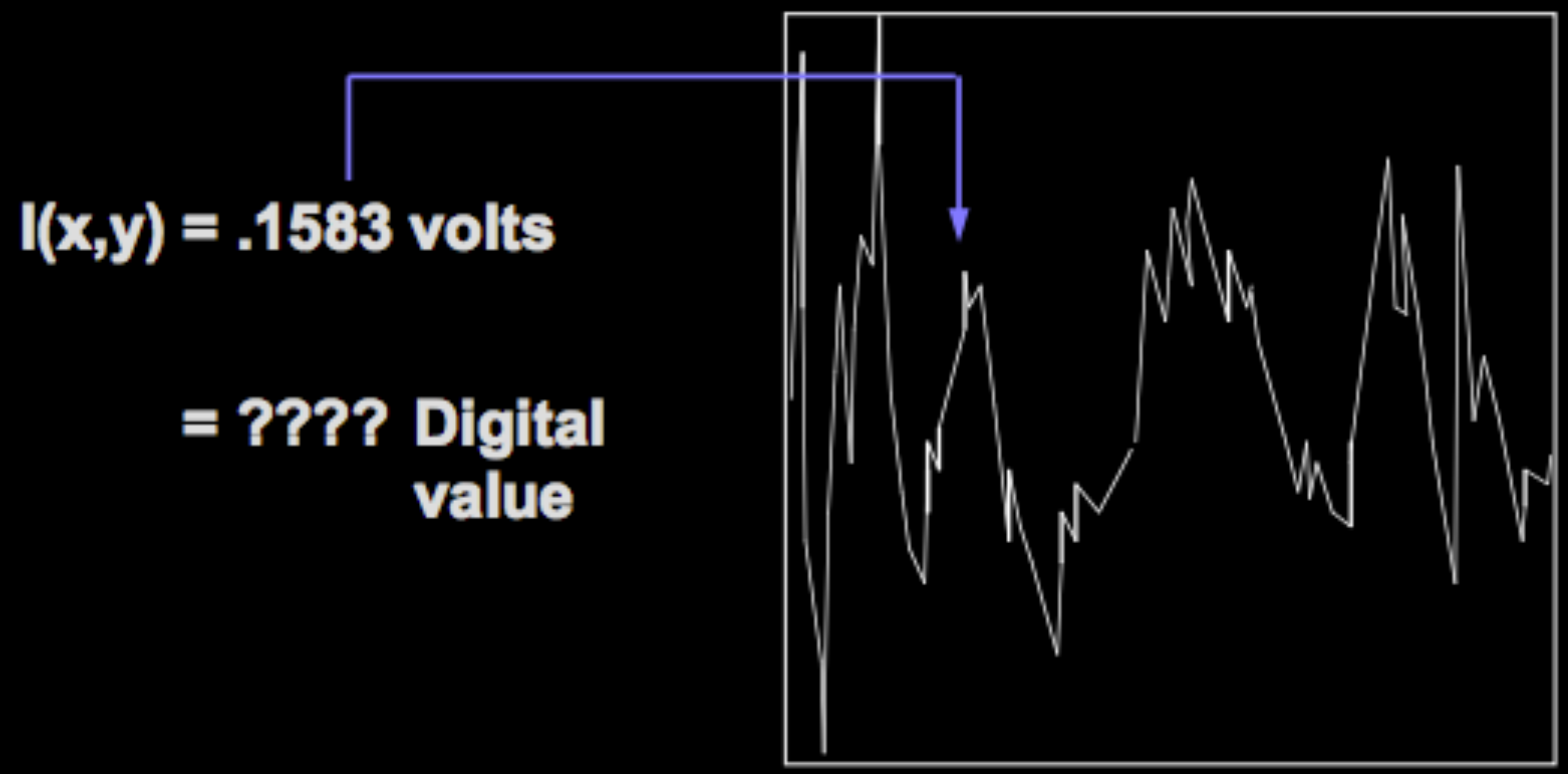
Discretization

Sampling strategies

- Spatial sampling
  - How many pixels?
  - What arrangement of pixels?
- Brightness sampling
  - How many brightness values?
  - Spacing of brightness values?
- For video, also the question of time sampling.

Signal quantization

Goal: determine a mapping from a continuous signal (e.g. analog video signal) to one of K discrete (digital) levels.



Quantization

I(x,y) = continuous signal: $0 \leq I \leq M$

Want to quantize to K values 0,1,....K-1

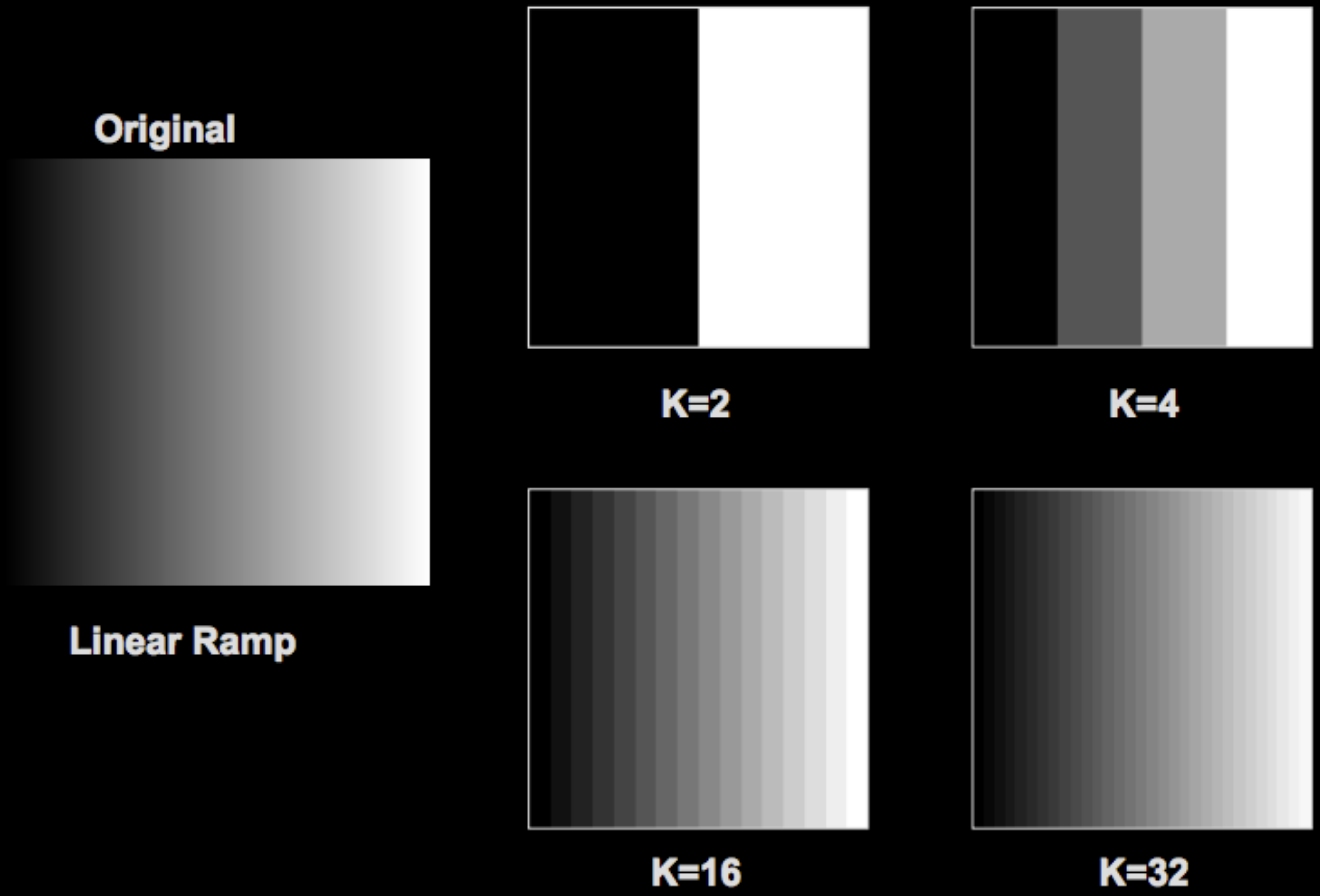
K usually chosen to be a power of 2:

K: #Levels	#Bits
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8

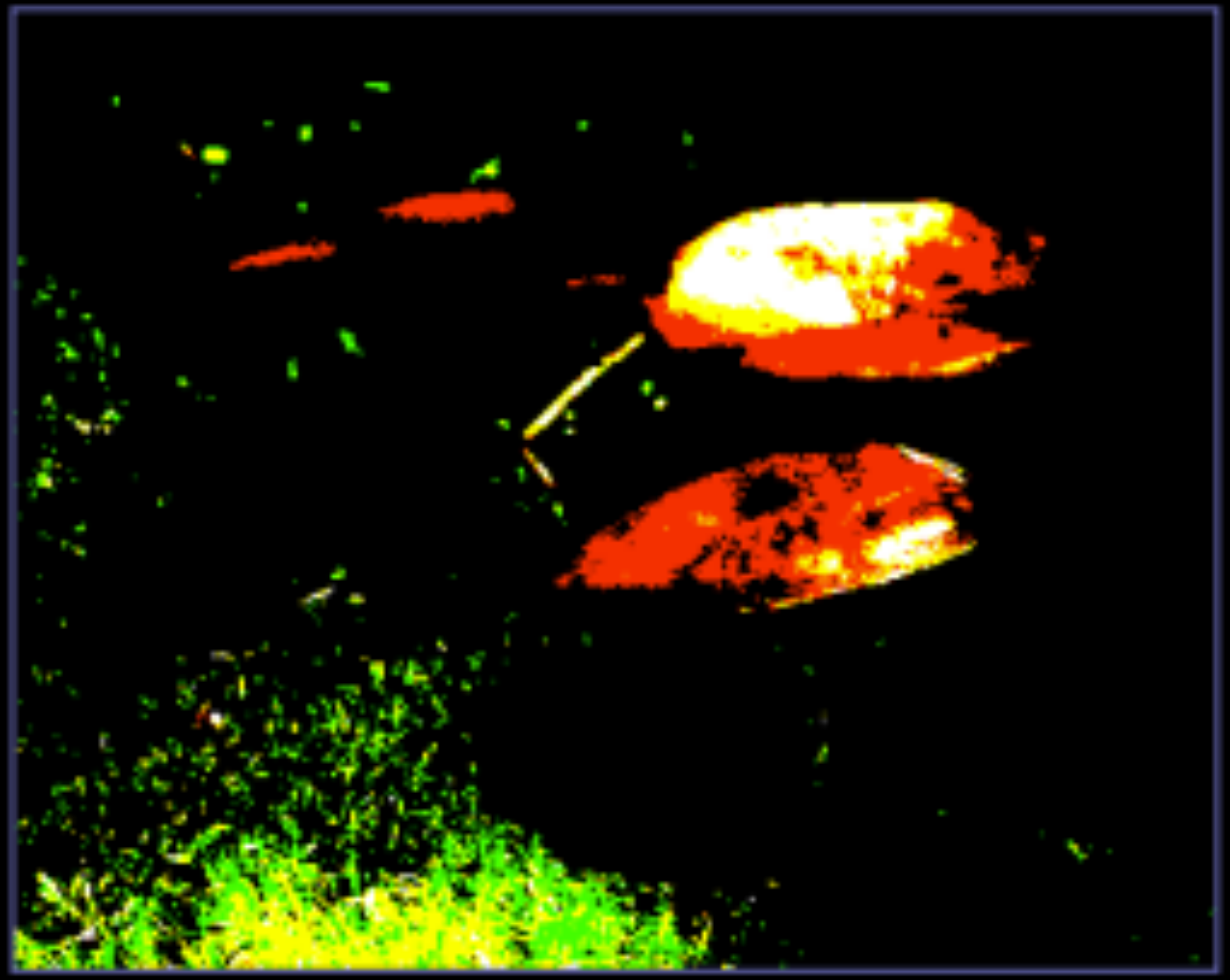
Mapping from input signal to output signal is to be determined.

Several types of mappings: uniform, logarithmic, etc.

Choice of K



Choice of K



K=2 (each color)

K=4 (each color)

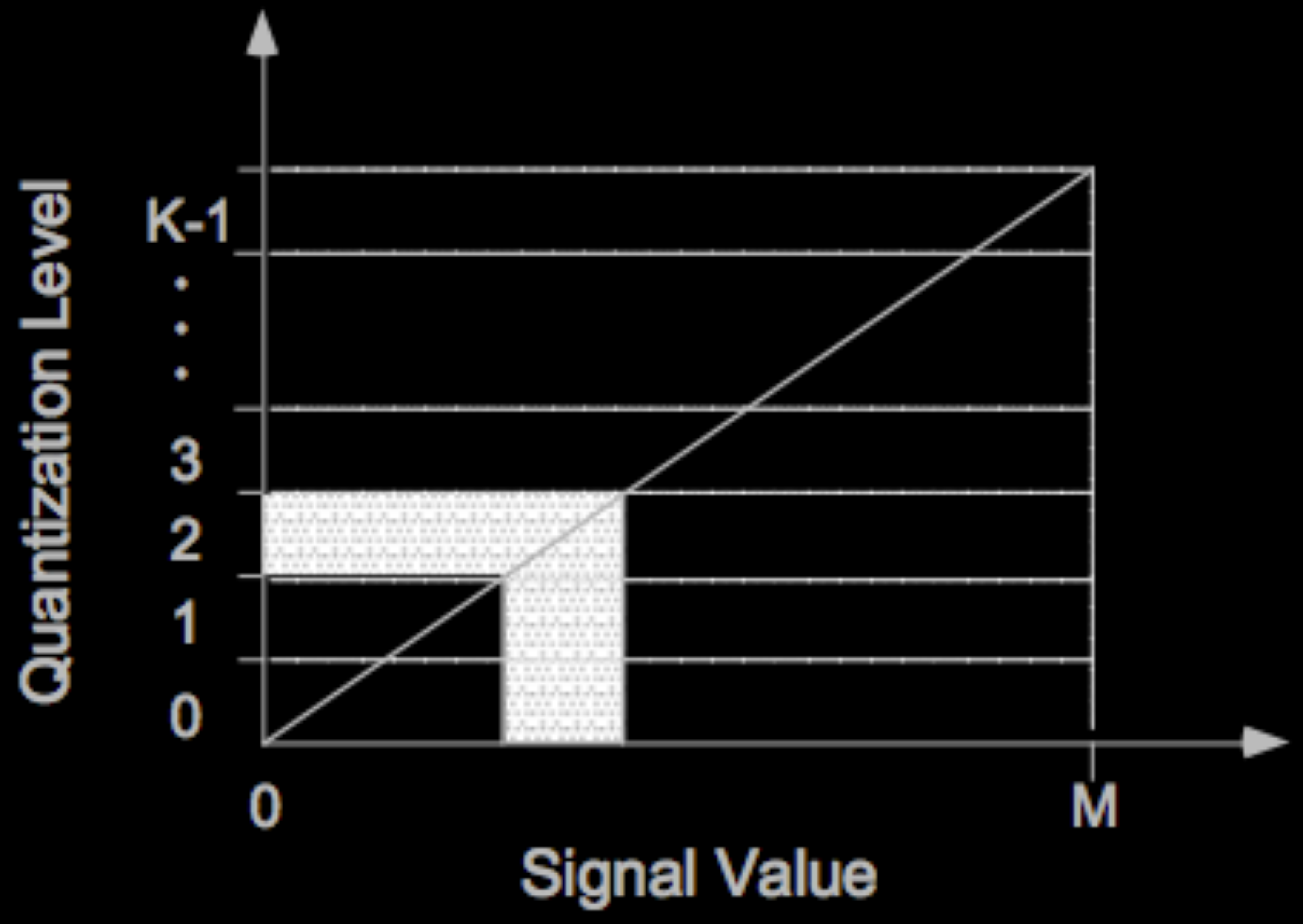
Choice of the function: uniform

Uniform sampling divides the signal range [0-M] into K equal-sized intervals.

The integers 0,...K-1 are assigned to these intervals.

All signal values within an interval are represented by the associated integer value.

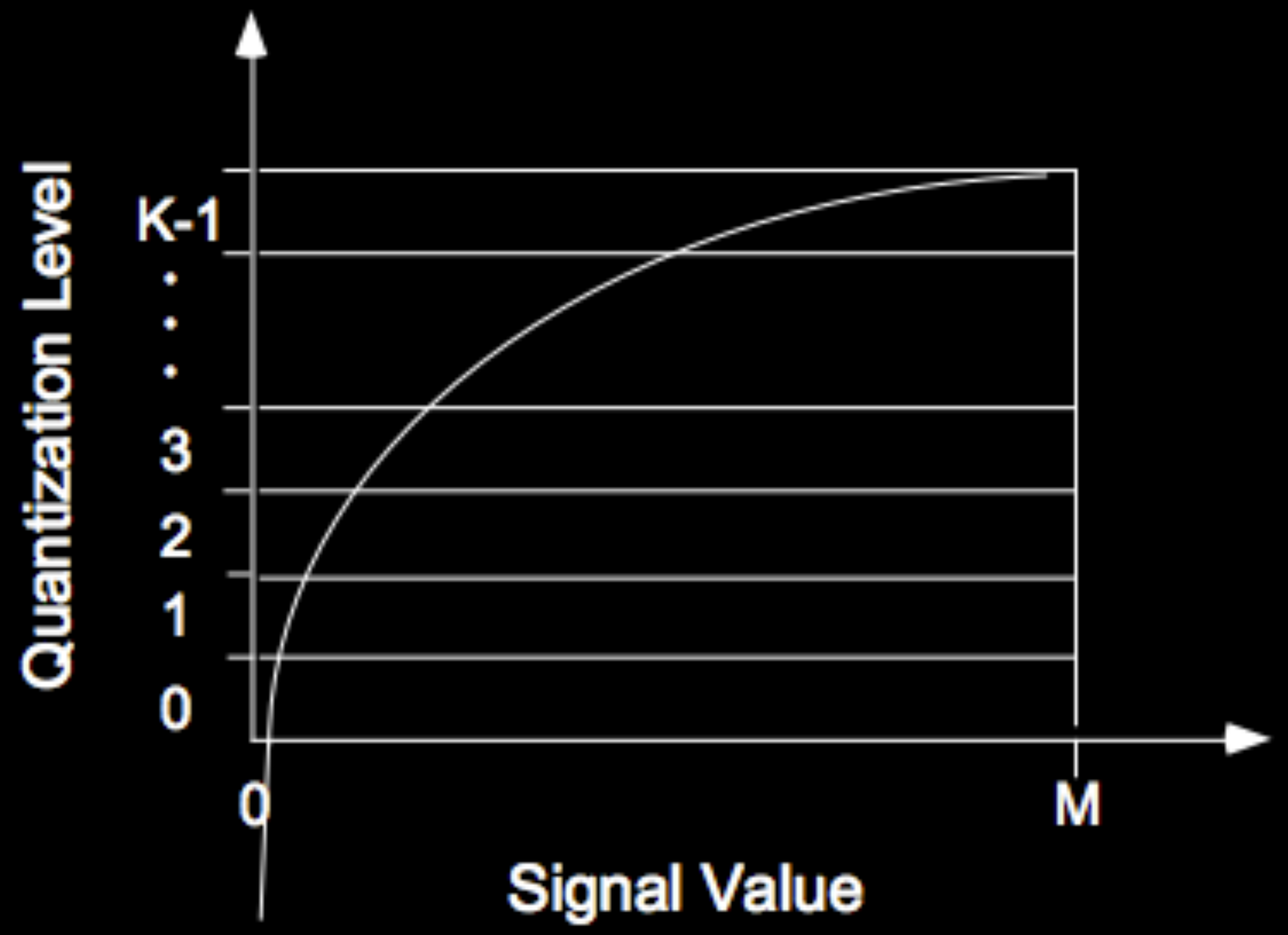
Defines a mapping:



Logarithmic quantization

Signal is: log I(x,y)

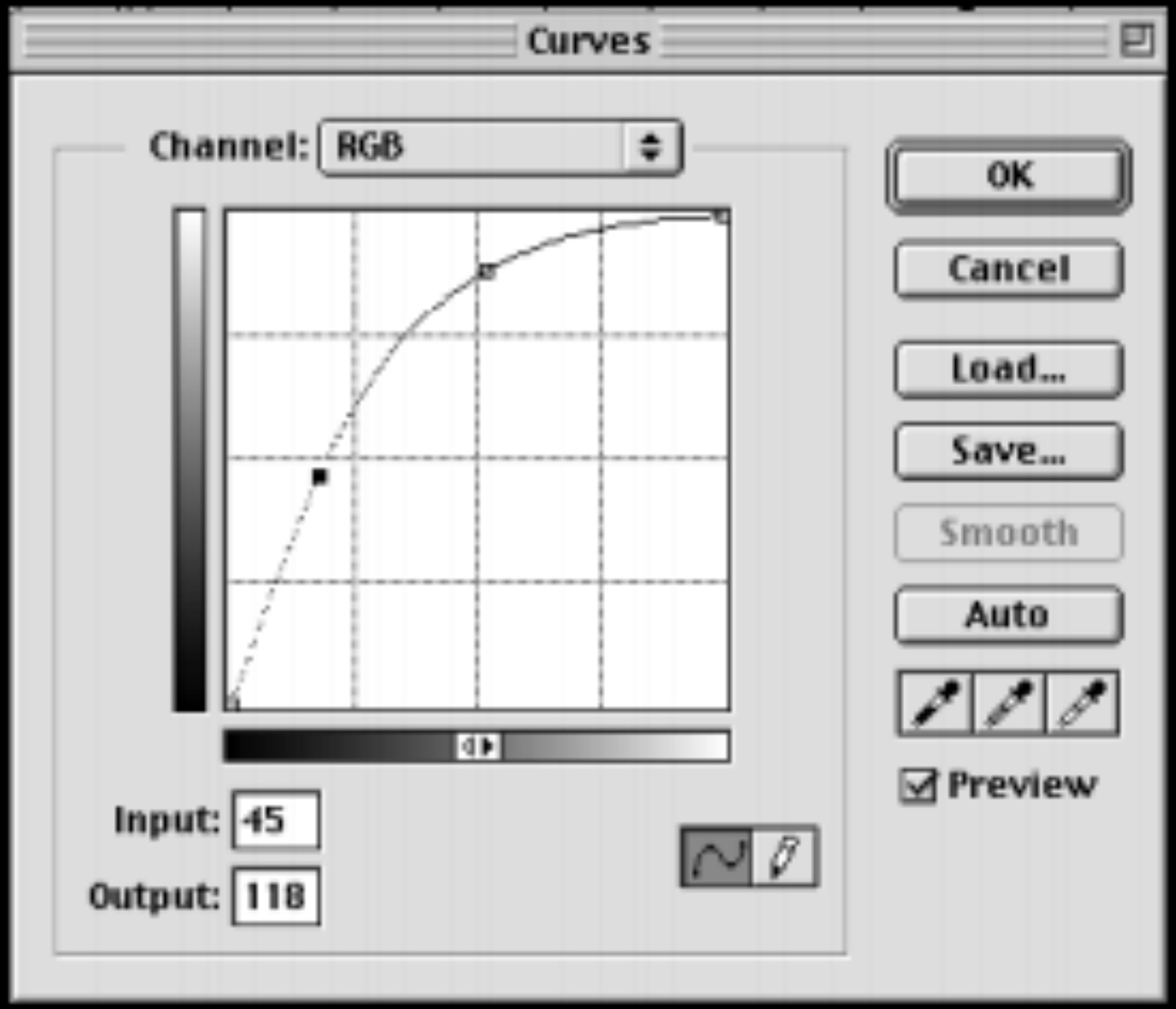
Effect is:



Detail enhanced in the low signal values at expense of detail in high signal values.

Logarithmic quantization

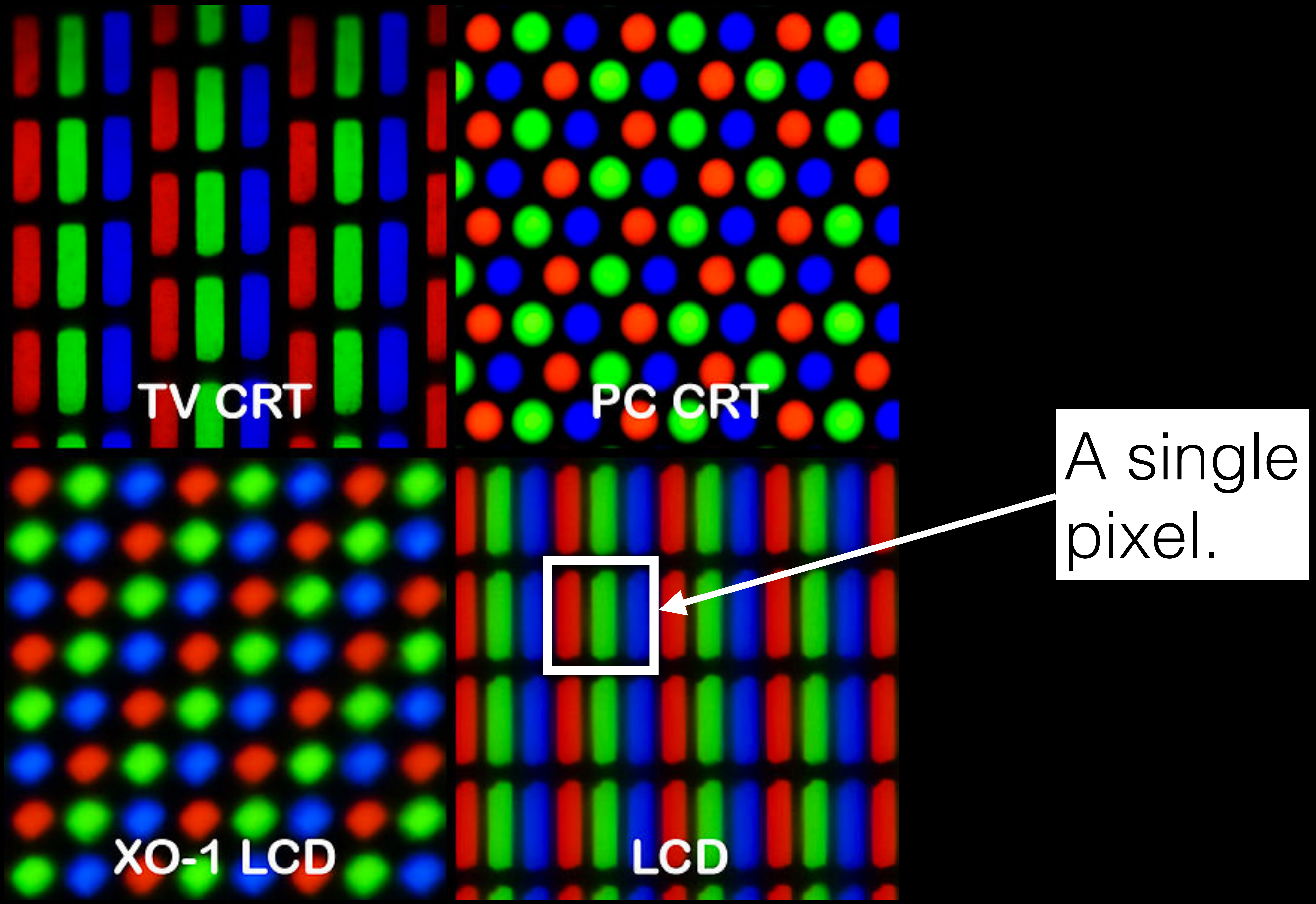
Quantization Curve



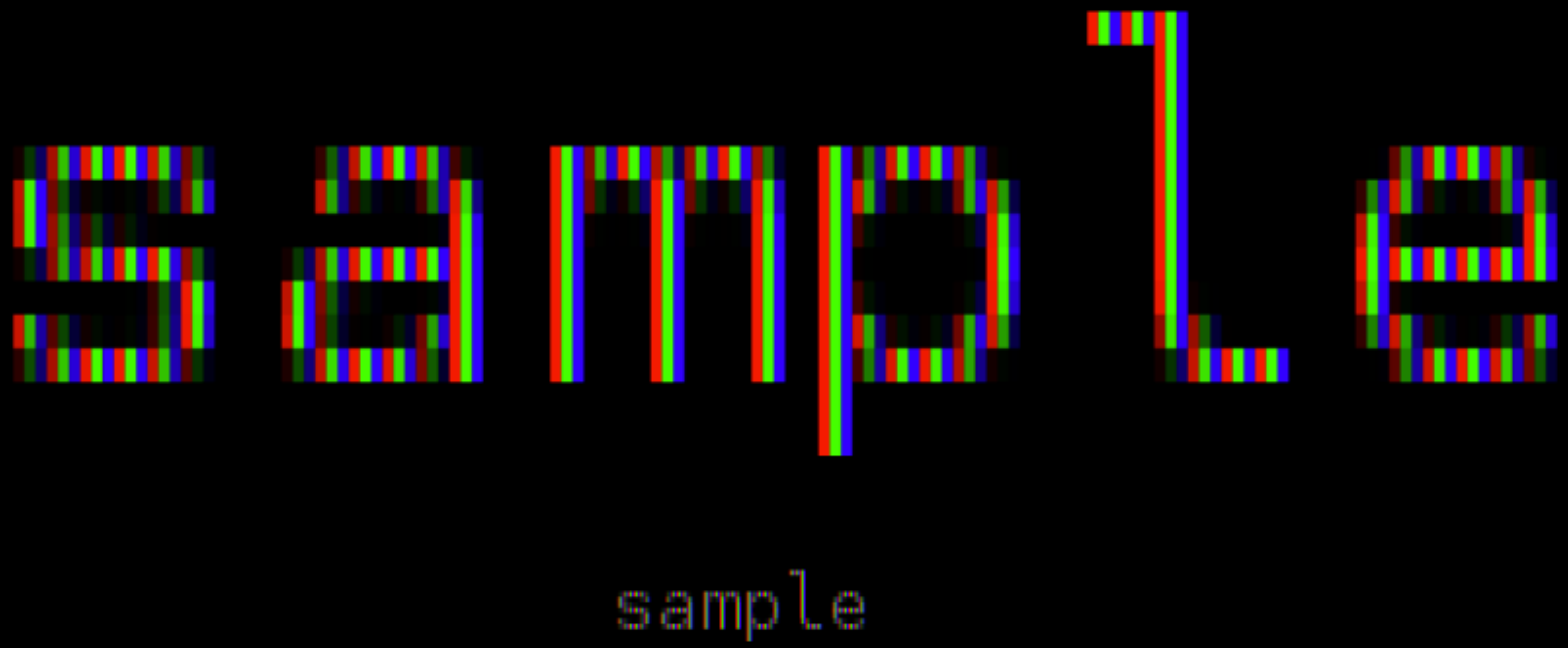
Color displays

Given a 24 bit color image (8 bits for R, G, B)

- Turn on 3 subpixels with power proportional to RGB values



“White” text on color display



http://en.wikipedia.org/wiki/Subpixel_rendering

Lookup tables

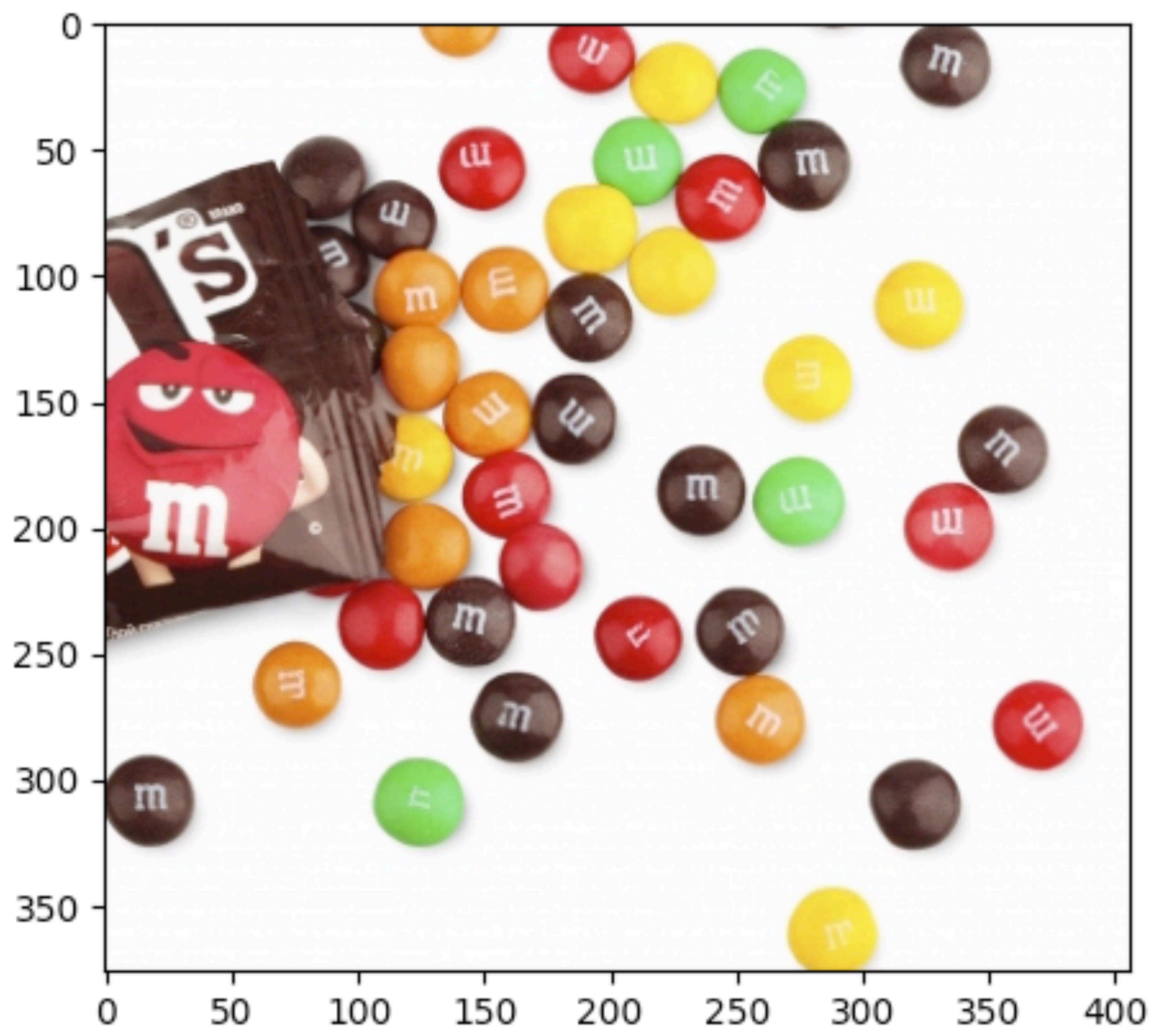
8 bit image: 256 different values.

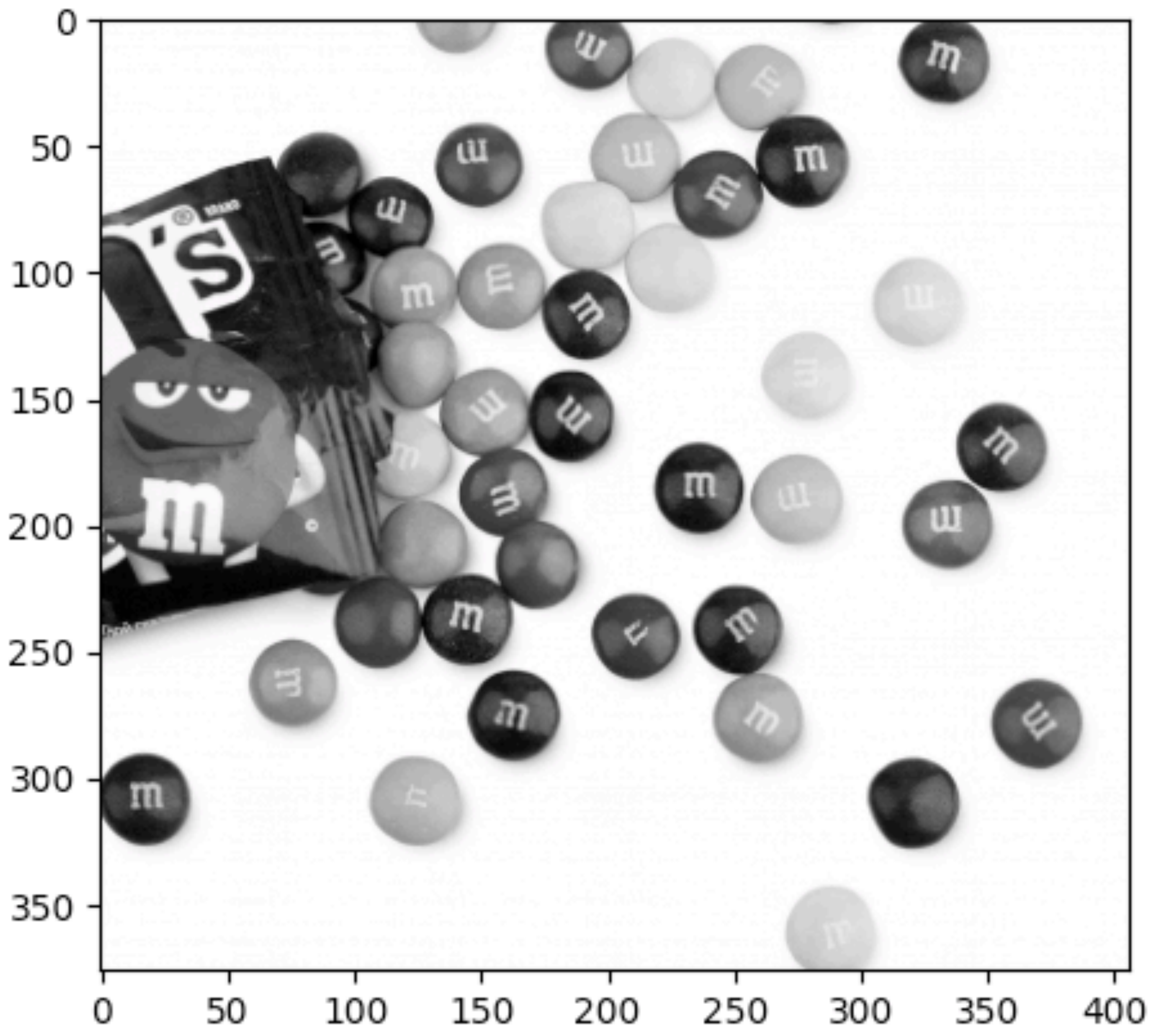
Simplest way to display: map each number to a gray value:

- 0 → (0.0, 0.0, 0.0) or (0,0,0)
- 1 → (0.0039, 0.0039, 0.0039) or (1,1,1)
- 2 → (0.0078, 0.0078, 0.0078) or (2,2,2)
- …
- 255 → (1.0, 1.0, 1.0) or (255,255,255)

This is called a grayscale mapping.

Color to gray and colormaps






```
File Edit Options Buffers Tools Python Help
from skimage import io
import matplotlib.pyplot as plt

im = io.imread('mnms.jpeg');
plt.figure(1);
plt.imshow(im)

gray = im[:,:,0]*0.3 + im[:,:,1]*0.6 + im[:,:,2]*0.1;

plt.figure(2);
plt.imshow(gray, cmap='gray')
plt.show()
```

R : G : B :: 0.3 : 0.6 : 0.1

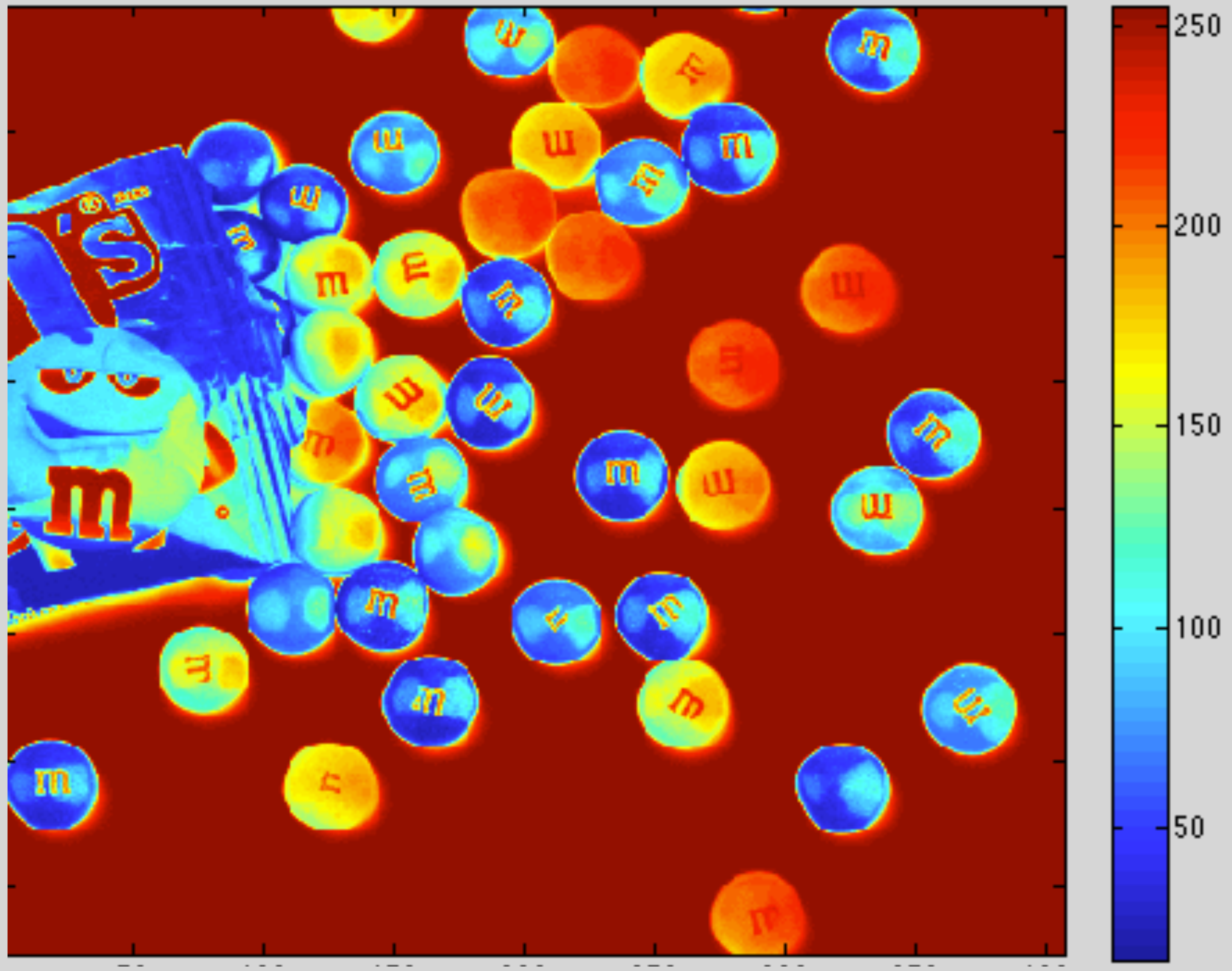
Non-gray lookup tables

We can also use other mappings:

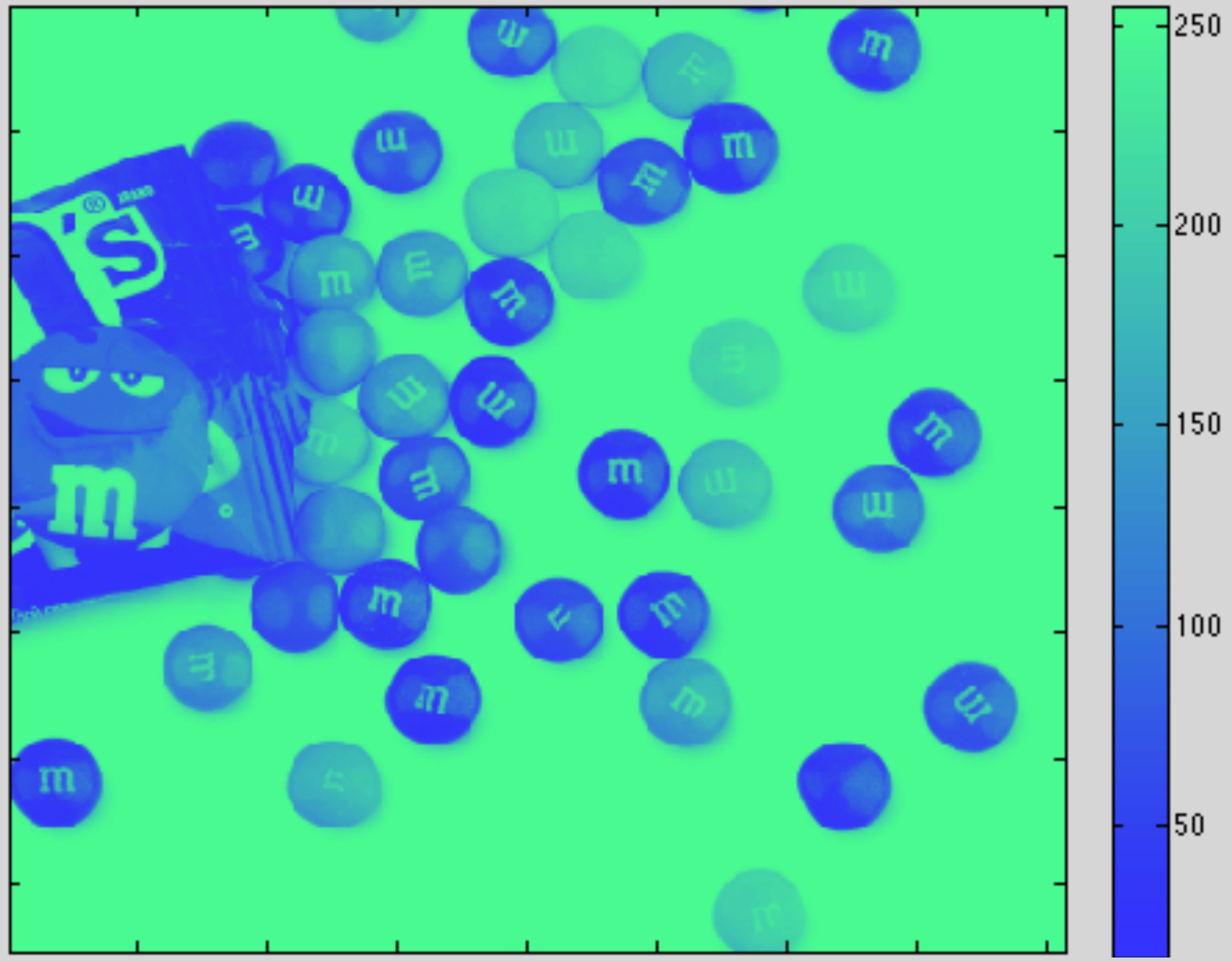
- 0 → (17, 25, 89)
- 1 → (45, 32, 200)
- ...
- 255 → (233,1,4)

These are called lookup tables.

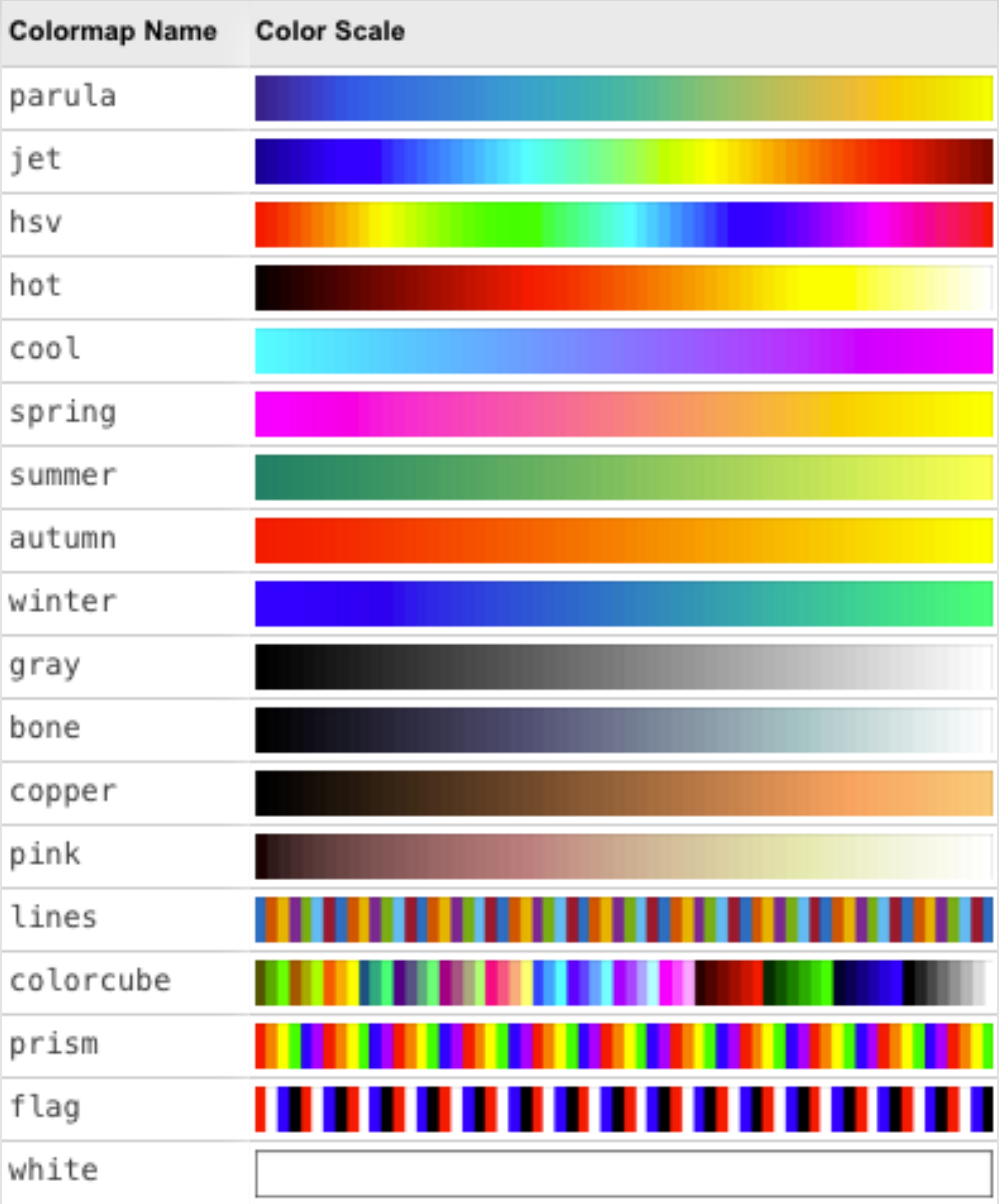
More colormaps



jet



winter



Enhancing images

What can we do to “enhance” an image after it has already been digitized?

- We can make the information that is there easier to visualize.
- We can guess at data that is not there, but we cannot be sure, in general.

Increasing the contrast

Removing motion blur

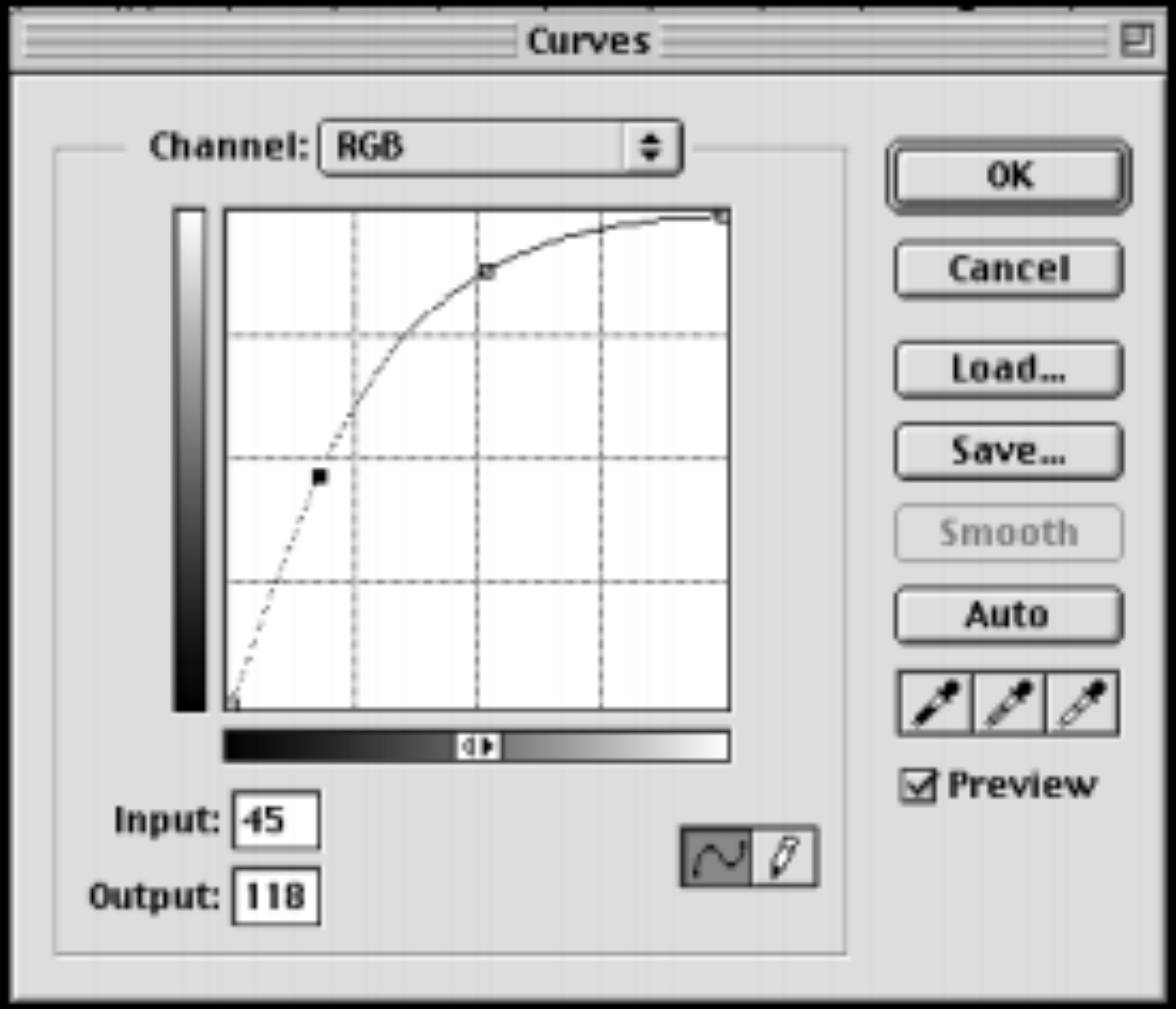
Contrast enhancement

Two methods:

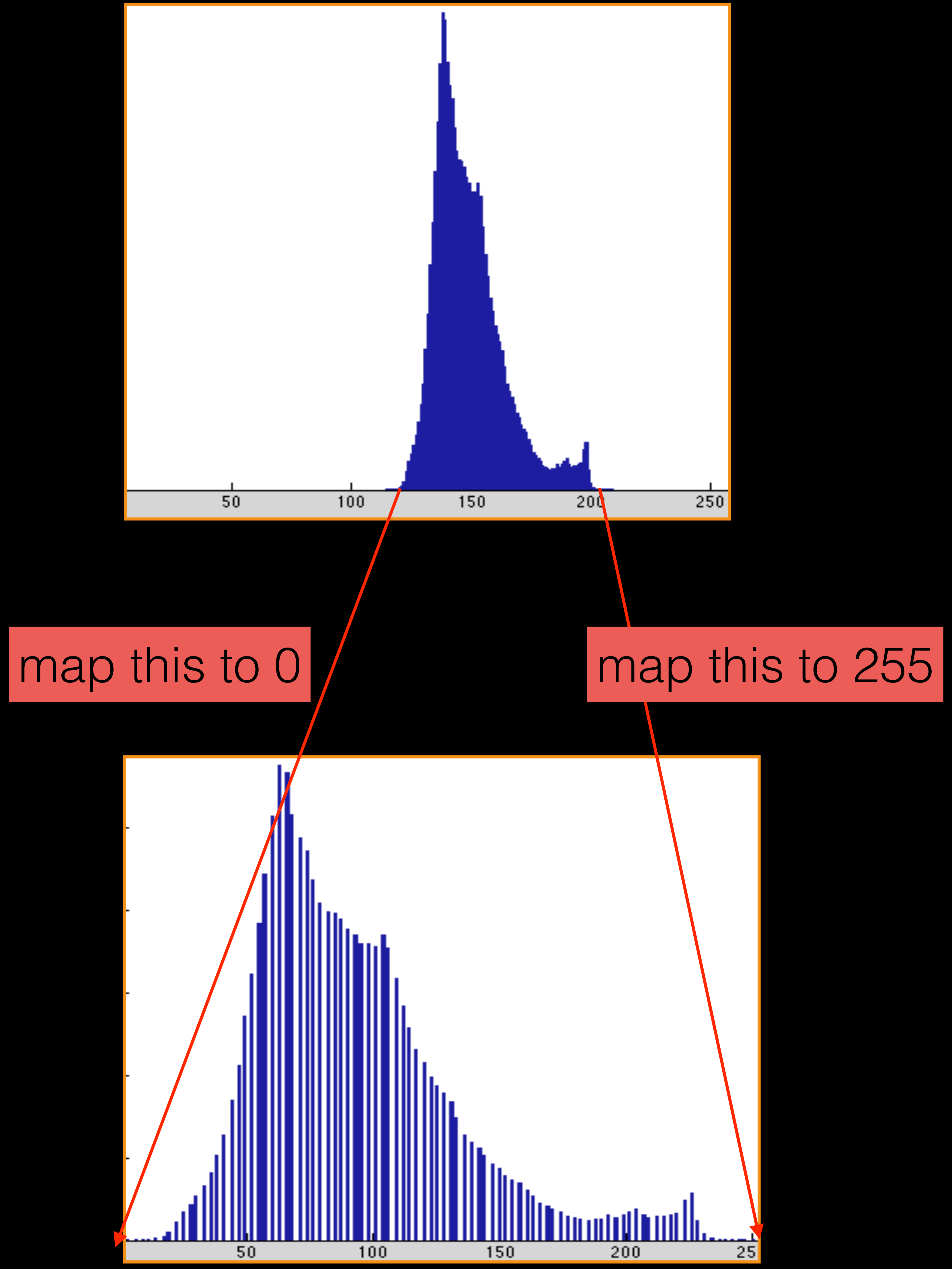
- Normalize the data (non-linear mapping, contrast stretching)
- Transform the data (histogram equalization)

Logarithmic quantization

Quantization Curve

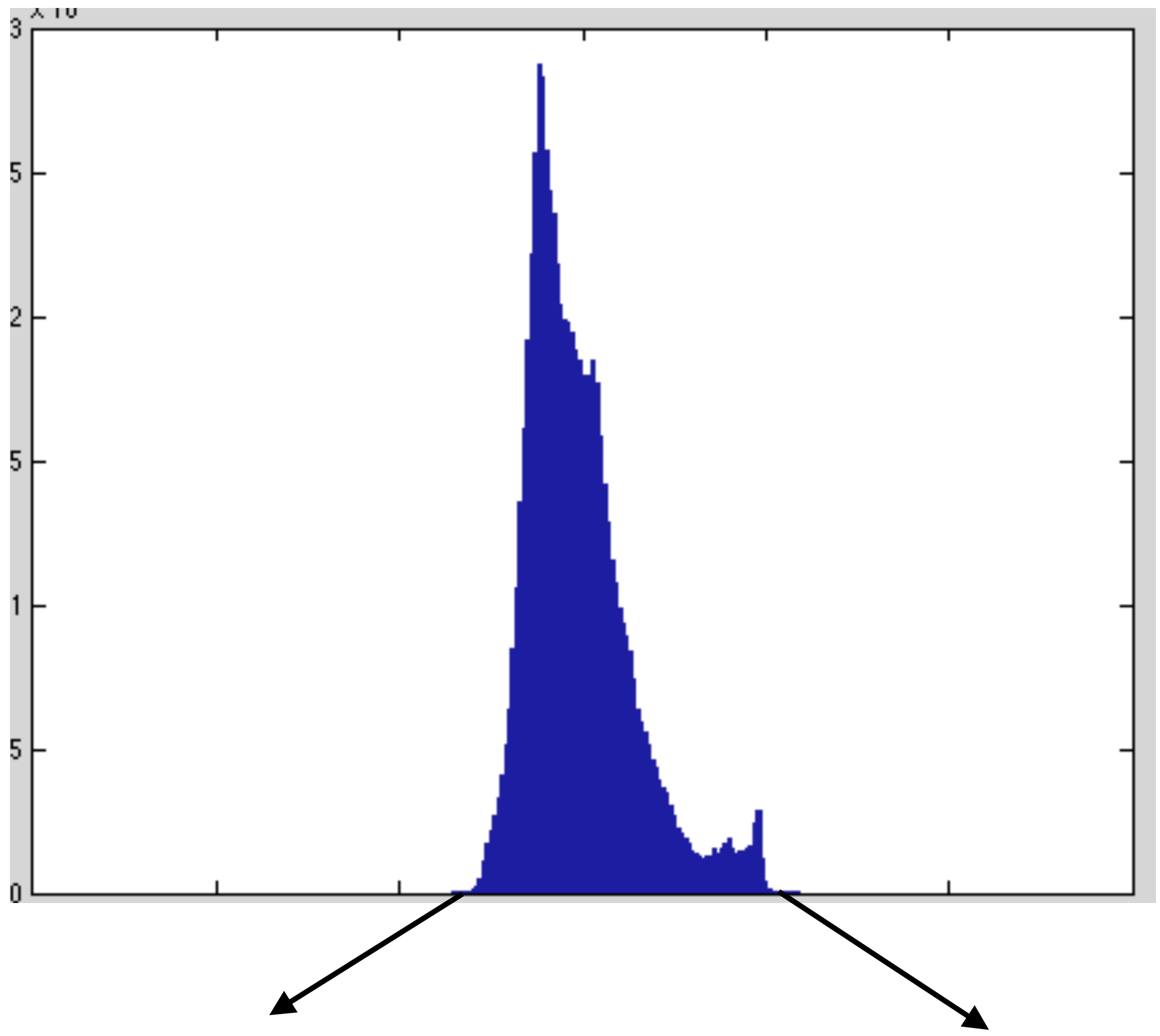


Contrast stretching



Contrast stretching

Basic idea: scale the brightness range of the image to occupy the full range of values



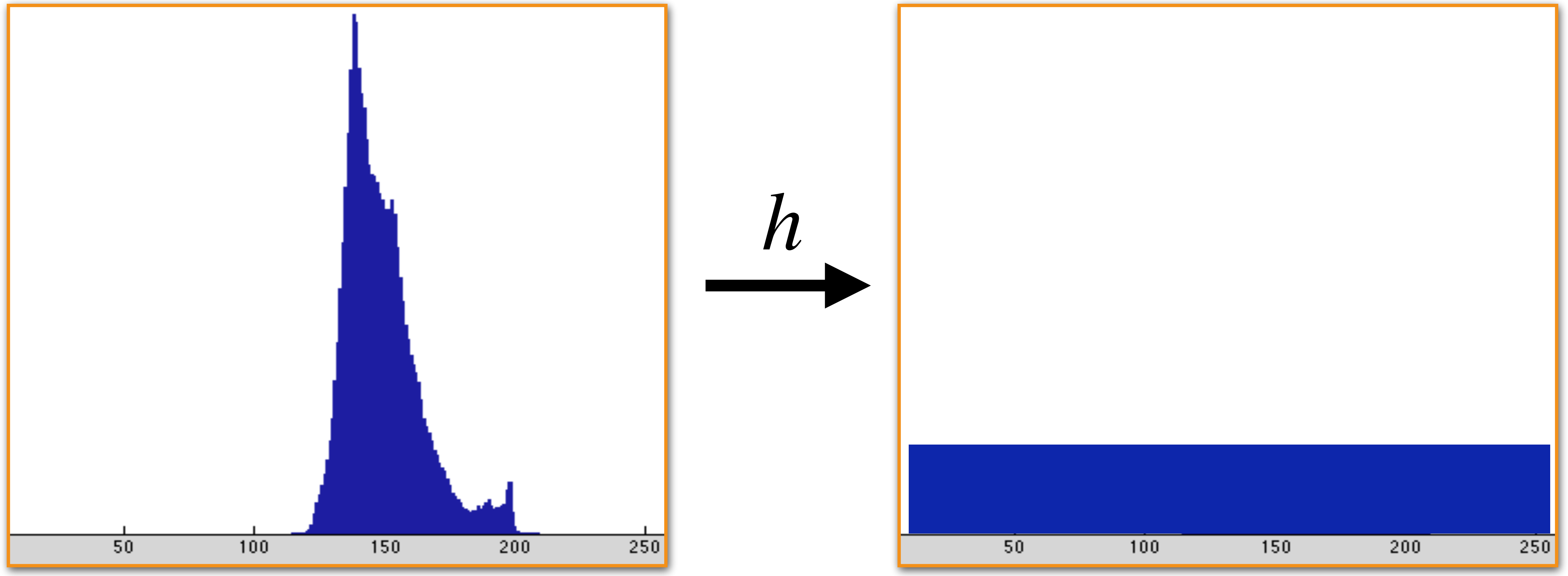
map this to 0

map this to 255

$$I \leftarrow floor\left(\frac{I - \min(I)}{\max(I) - \min(I)} \times 255\right)$$

Question: When is contrast stretching not effective?

Histogram equalization

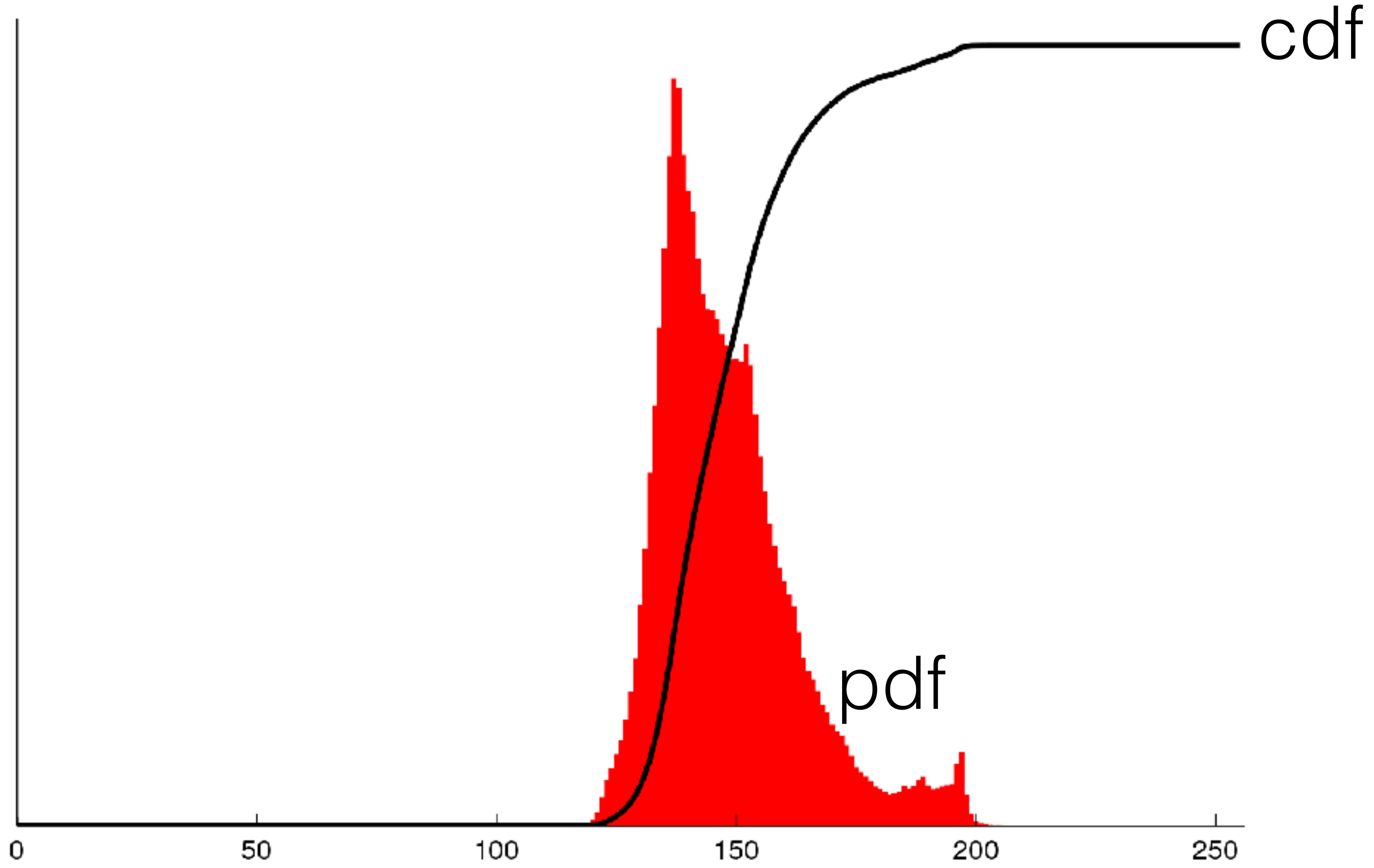


Remap data to create a uniform distribution
Why is this good?

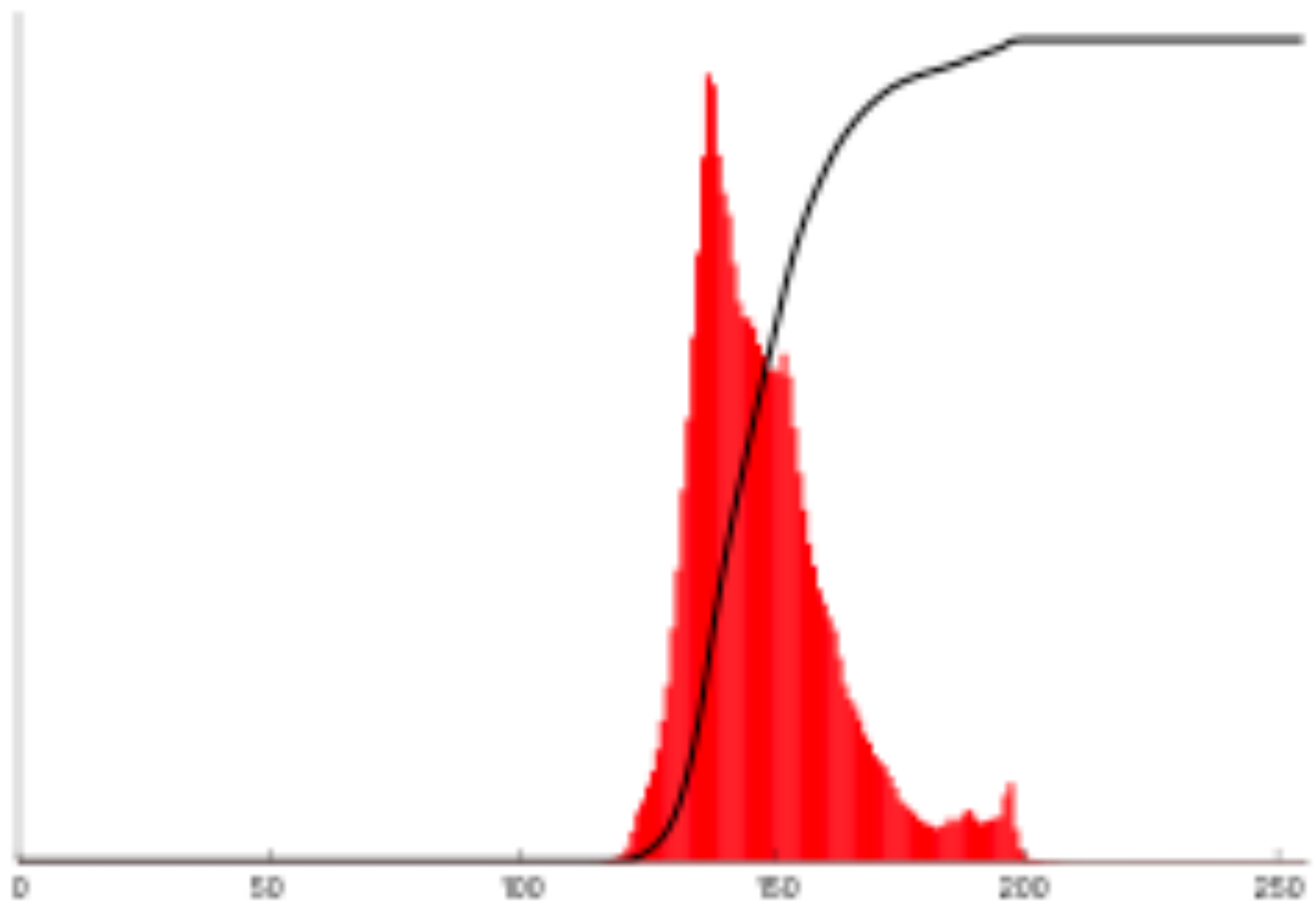
https://en.wikipedia.org/wiki/Histogram_equalization

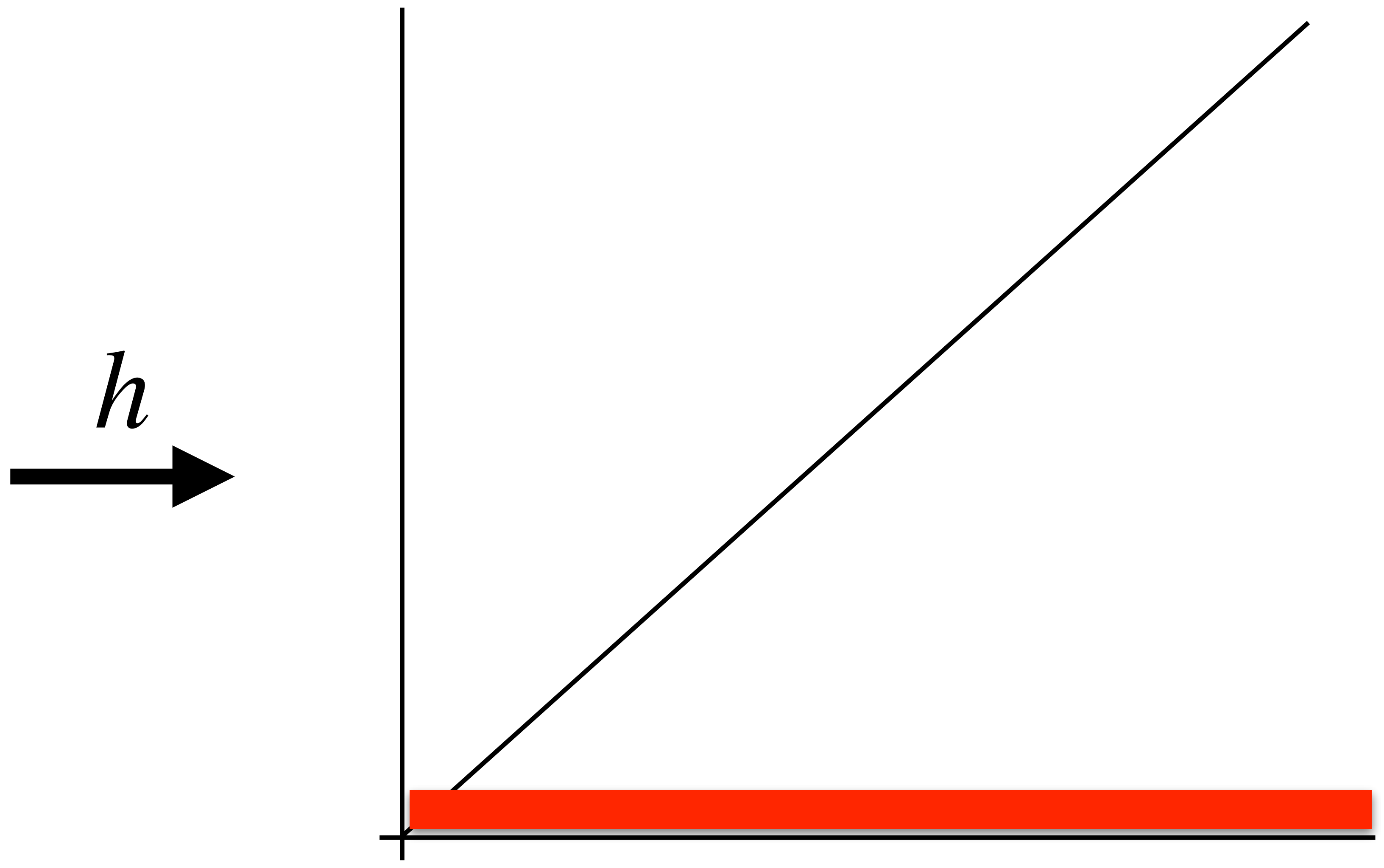
Cumulative distribution function

pdf(v) = #pixels with value = v	Probability density function (aka histogram)
cdf(v) = #pixels with value <= v	Cumulative distribution function



Histogram equalization …



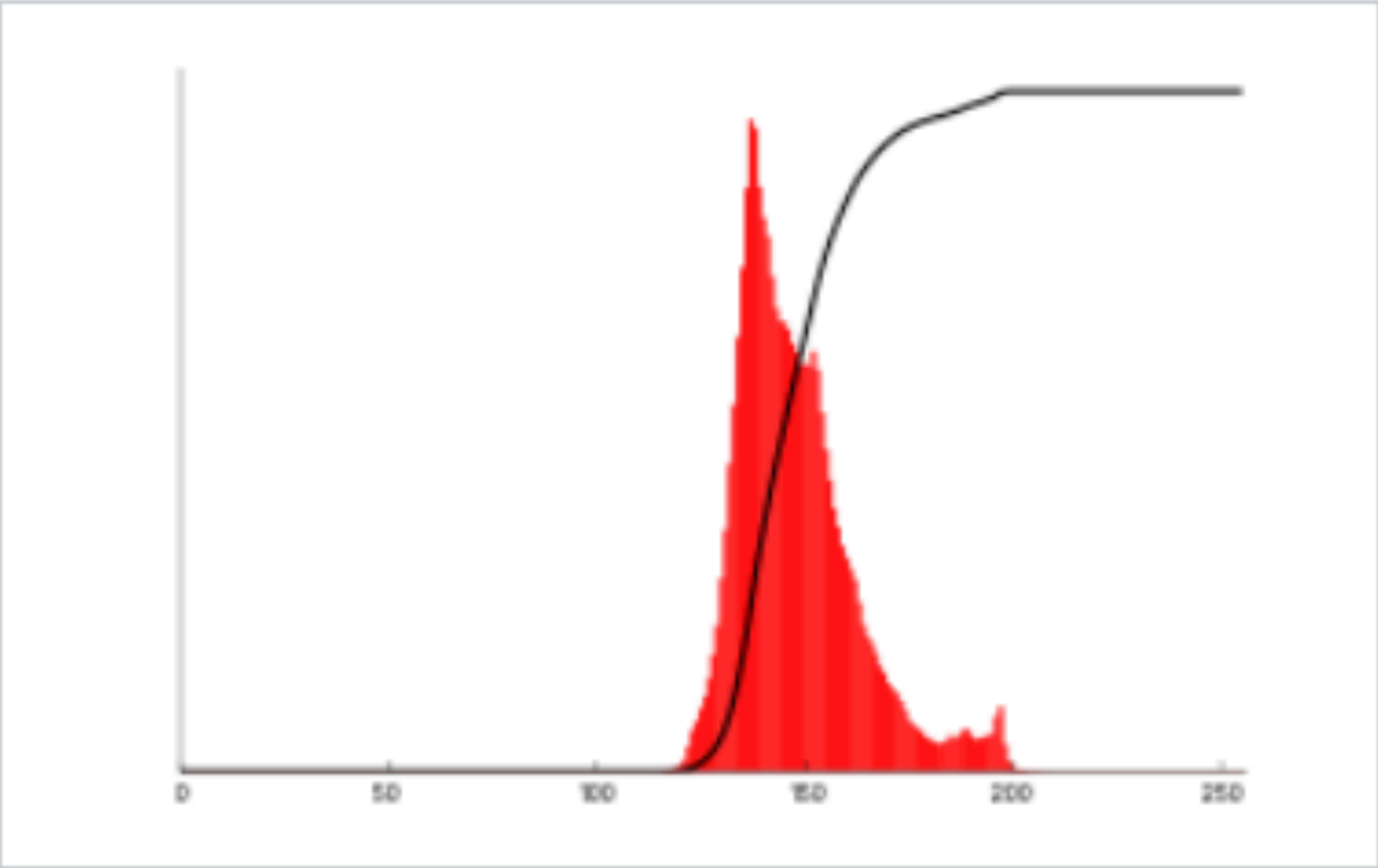


What happens to the ***cdf*** after equalization?
What value should pixels=v be mapped to?

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

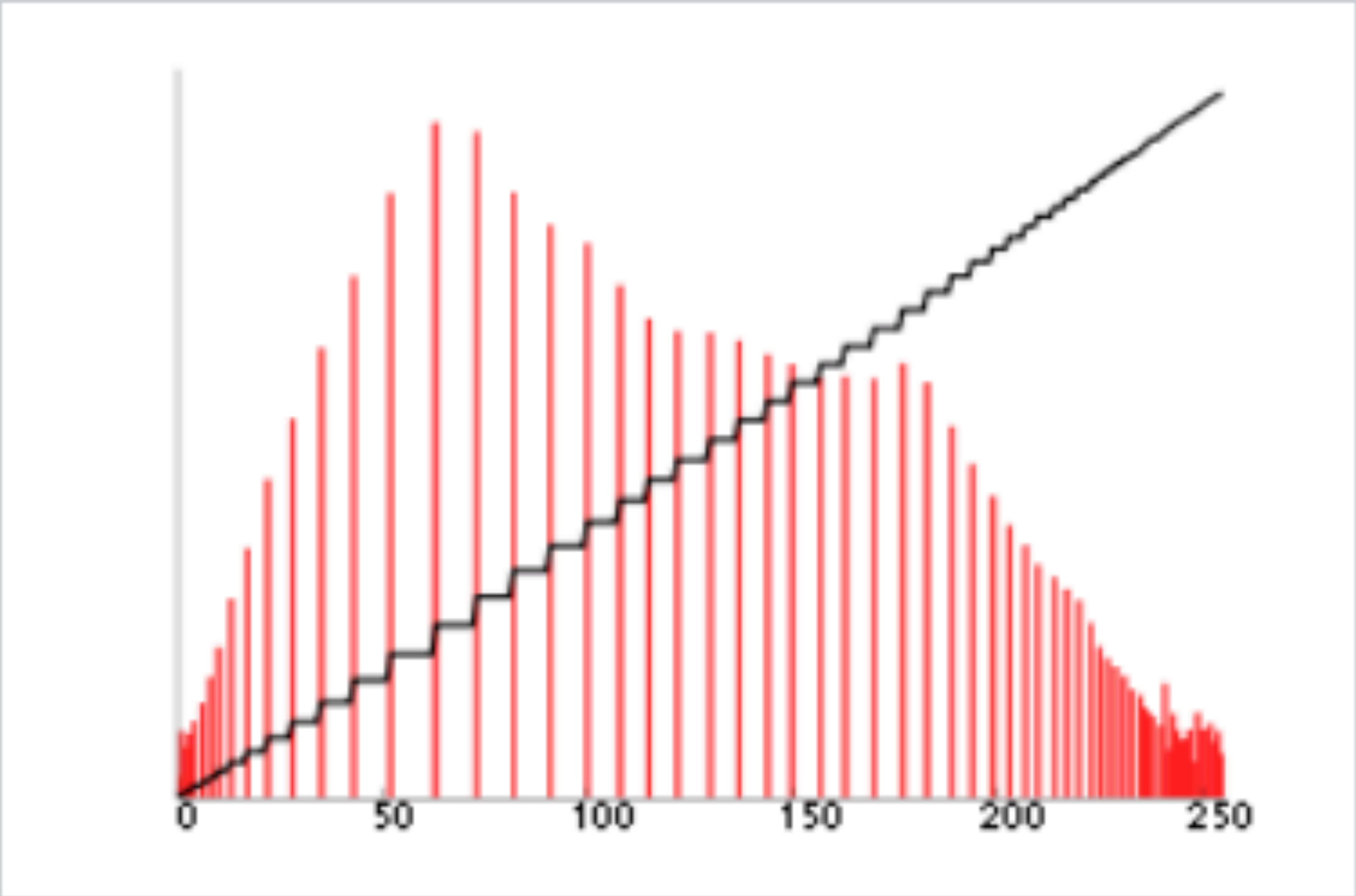
M x N pixels
L levels

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

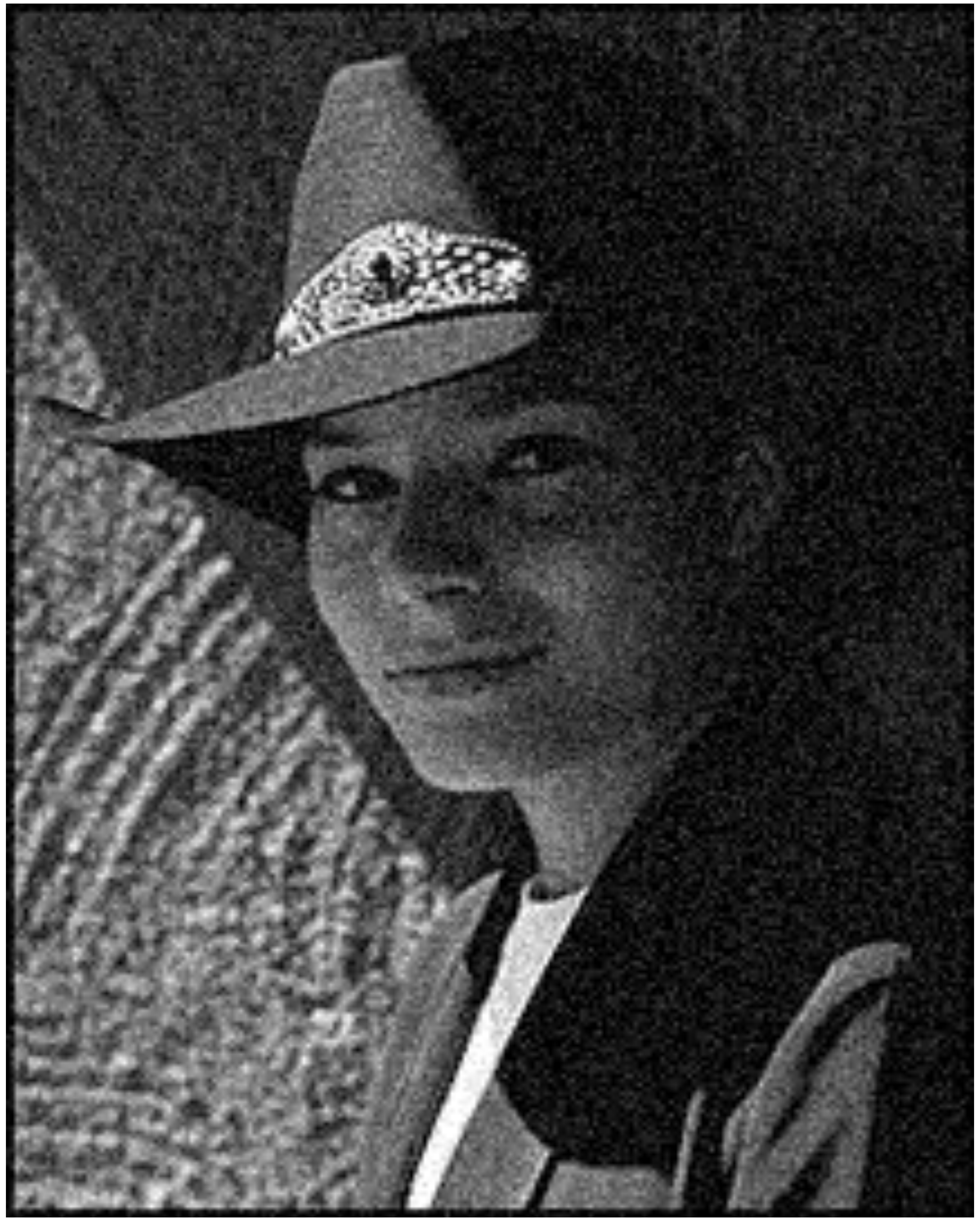
After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Denoising

How can we reduce noise in a photograph?



Moving average

Let's replace each pixel with a *weighted* average of its neighborhood

The weights are called the *filter kernel*

Weights for the average of a 3x3 neighborhood

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

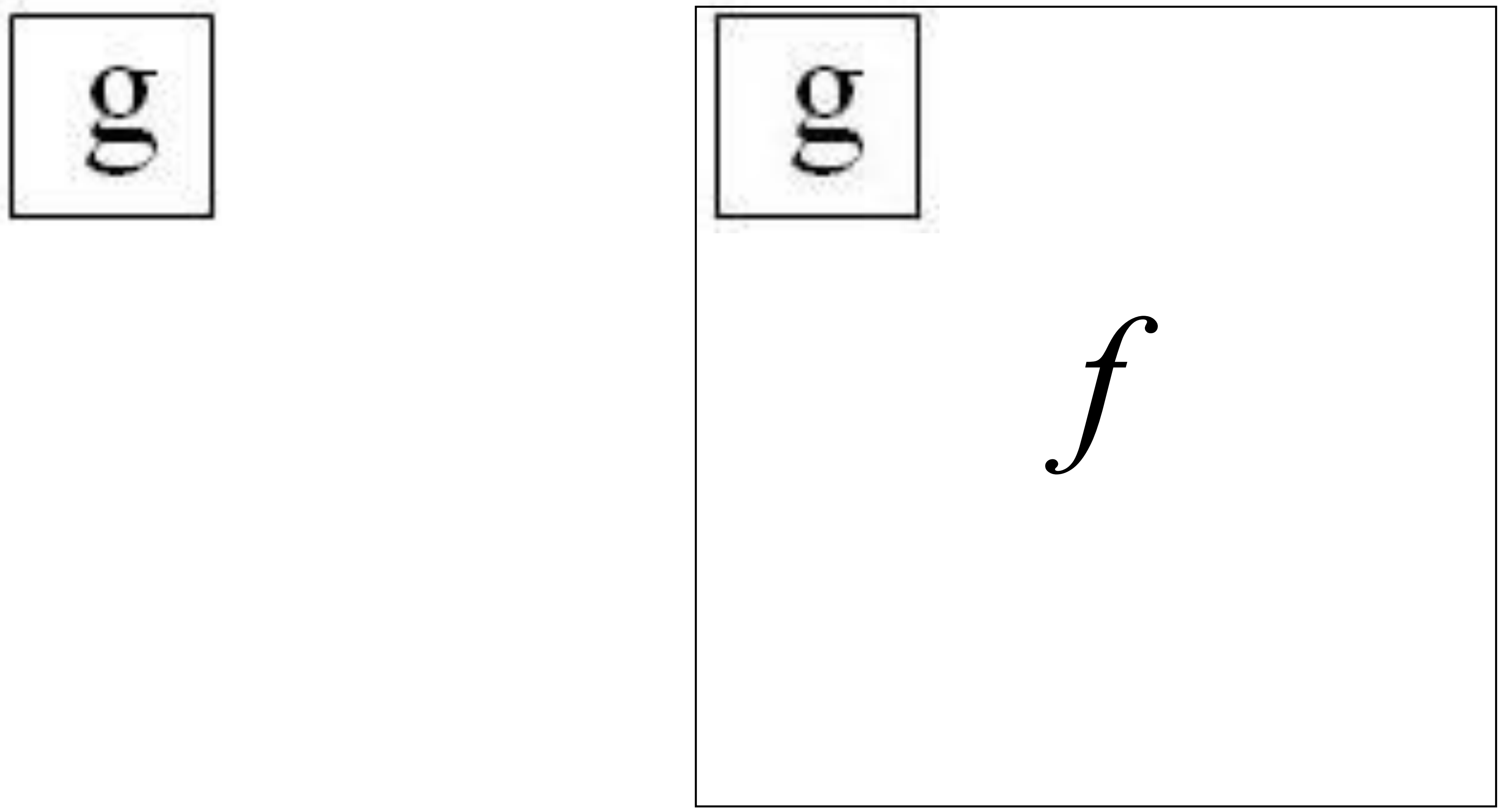
"box filter"

Source: D. Lowe

Filtering

Let f be the image and g be the kernel. The output of filtering f with g denoted $f*g$ is given by:

$$(f * g)[m,n] = \sum_{k,l} f[m+k, n+l]g[k,l]$$



Filtering computes the correlation between the g and f at each location

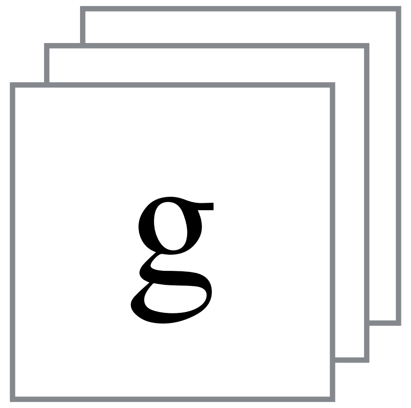
Convolution is filtering with a flipped g (by notation)

Source: F. Durand

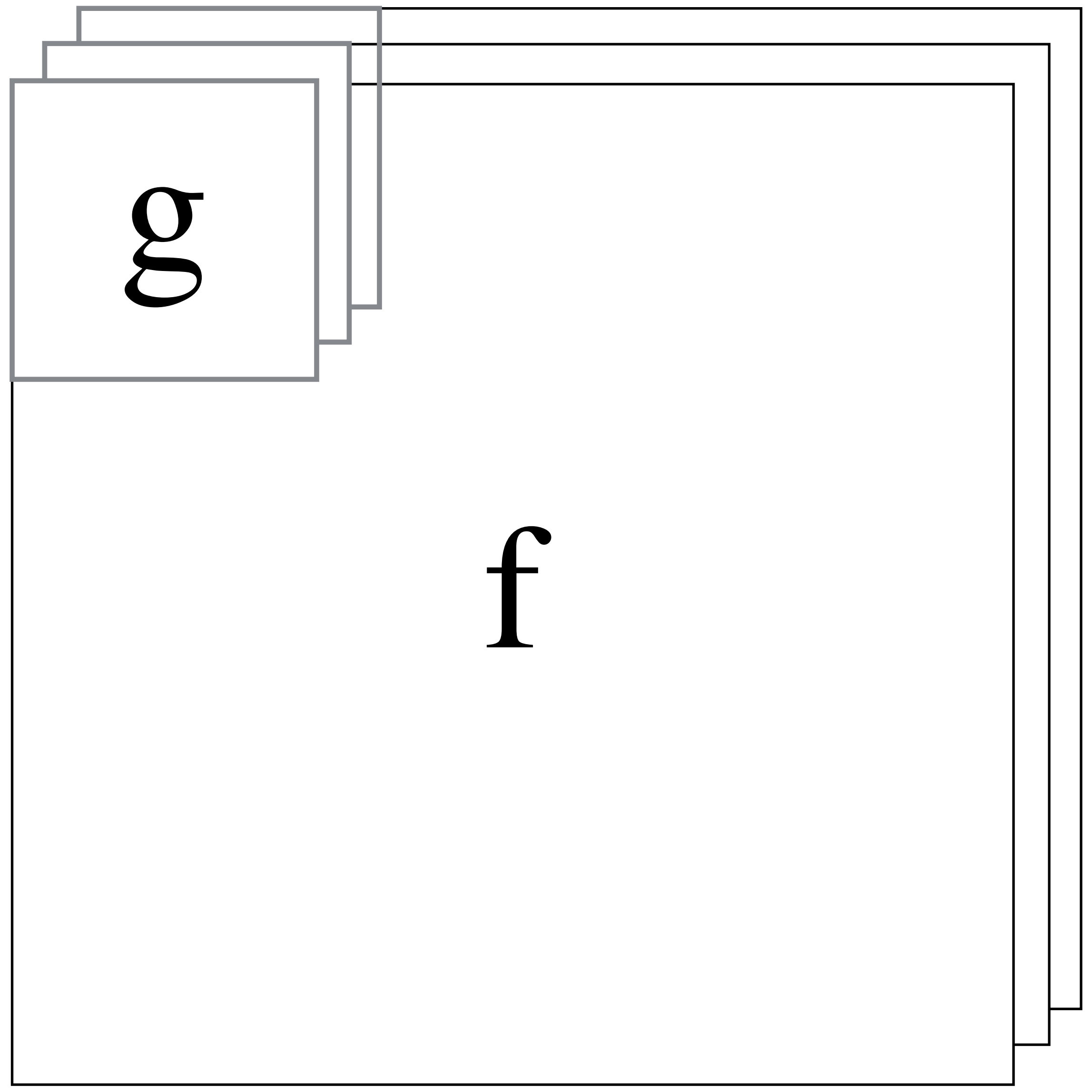
Filtering: multi-channel case

Let f be the image and g be the kernel. The output of filtering f with g denoted $f * g$ is given by:

$$(f * g)[m, n] = \sum_{k,l,c} f[m + k, n + l, c]g[k, l, c]$$



multi-channel



multi-channel

Key properties

Linearity: filter(f_1 + f_2) = filter(f_1) + filter(f_2)

Shift invariance: same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))

<u>Theoretical result:</u> any linear shift-invariant operator can be represented as a convolution

Properties in more detail

Commutative: $a * b = b * a$

- Conceptually no difference between filter and signal

Associative: $a * (b * c) = (a * b) * c$

- Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
- This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

Scalars factor out: $ka * b = a * kb = k (a * b)$

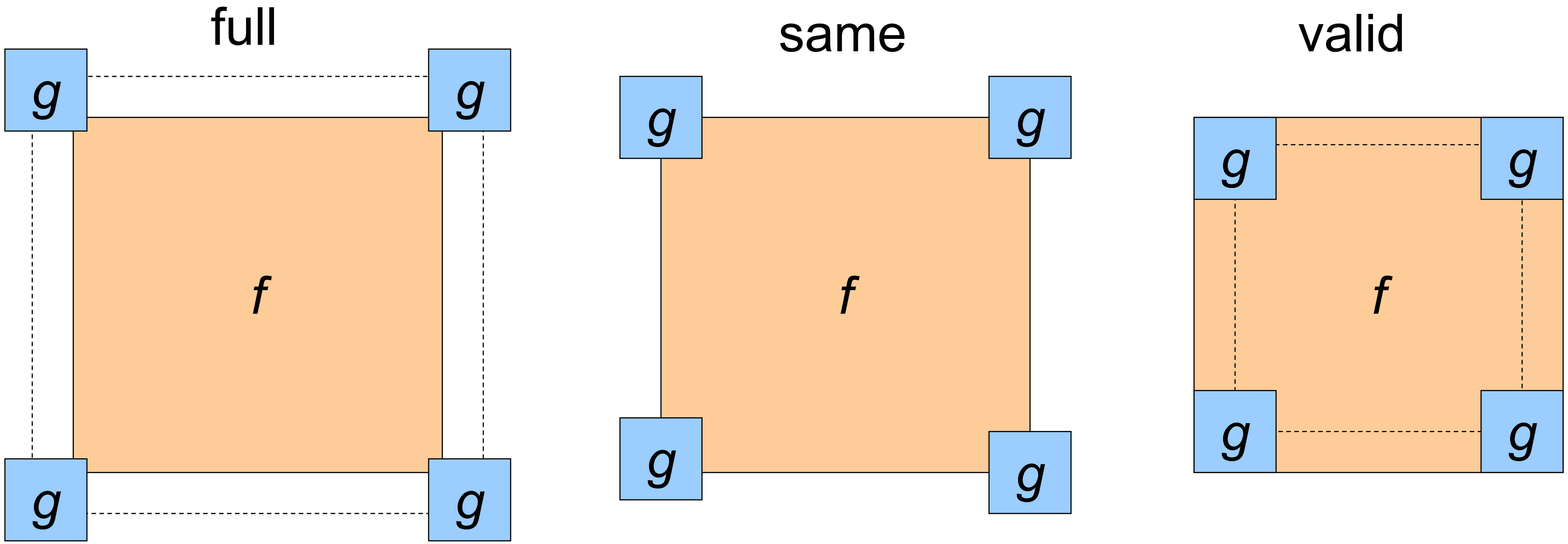
Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Annoying details

What is the size of the output?

Python: scipy.ndimage.correlate / convolve

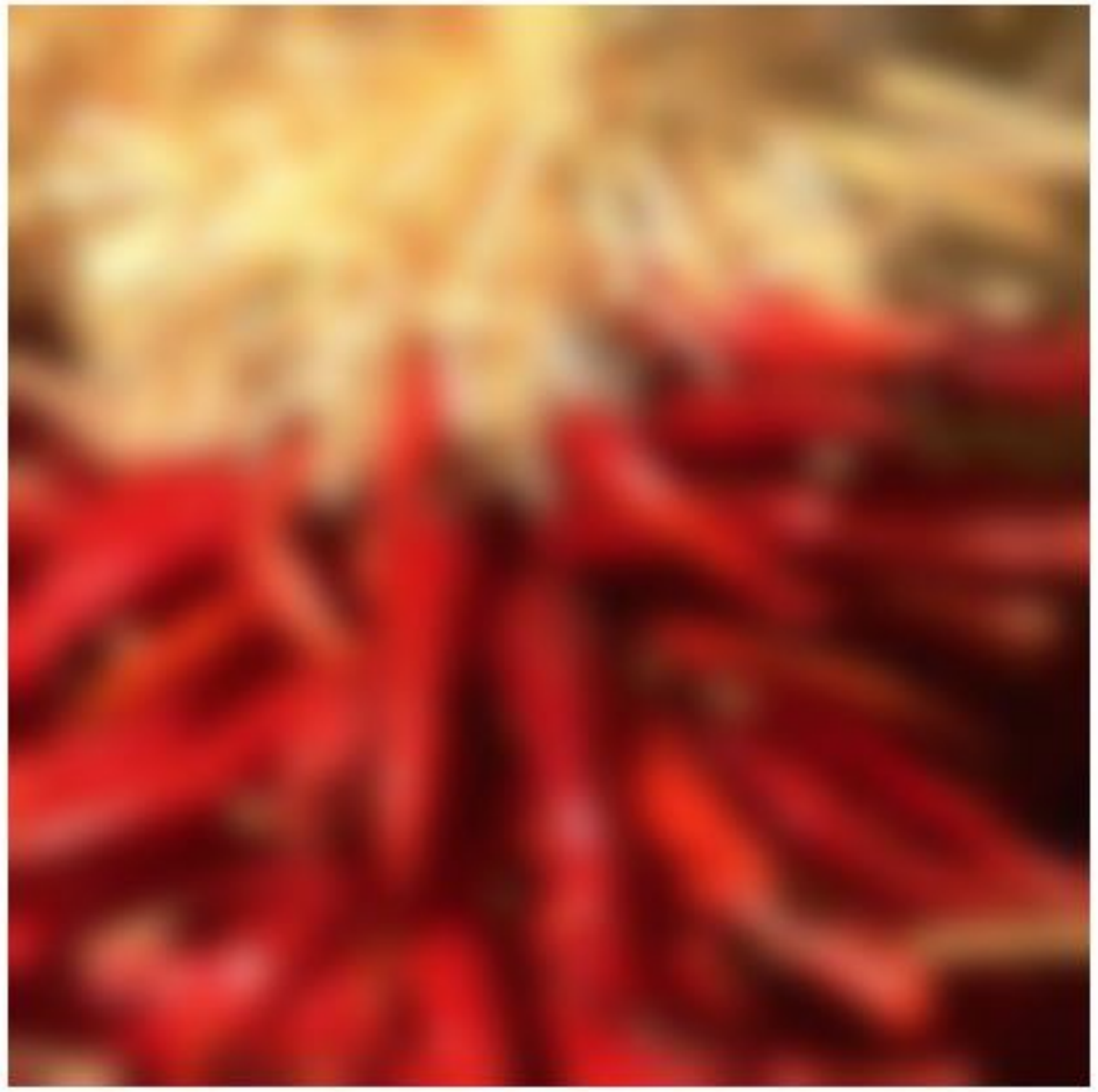
- *shape* = 'full': output size is sum of sizes of f and g
- *shape* = 'same': output size is same as f
- *shape* = 'valid': output size is difference of sizes of f and g



Annoying details

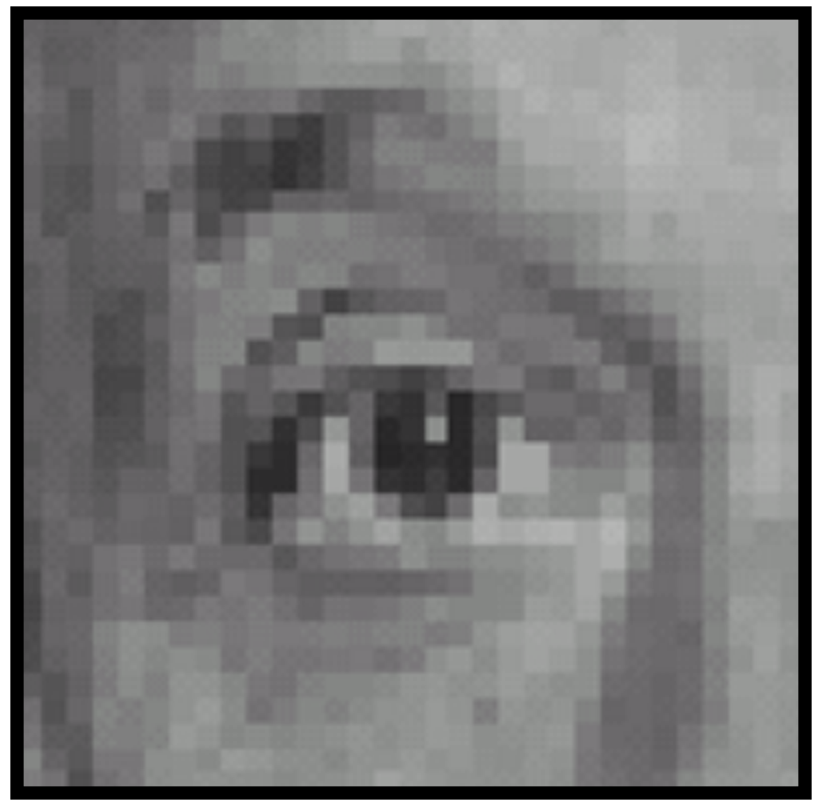
What about near the edge?

- the filter window falls off the edge of the image
- need to extrapolate
- methods:
  - clip filter (black) — correlate(f, g, mode=‘constant’, cval=0.0)
  - wrap around — correlate(f, g, mode=‘wrap’)
  - copy edge — correlate(f, g, mode=‘nearest’)
  - reflect across edge — correlate (f, g, mode=‘reflect’)



Source: S. Marschner

Practice with linear filters



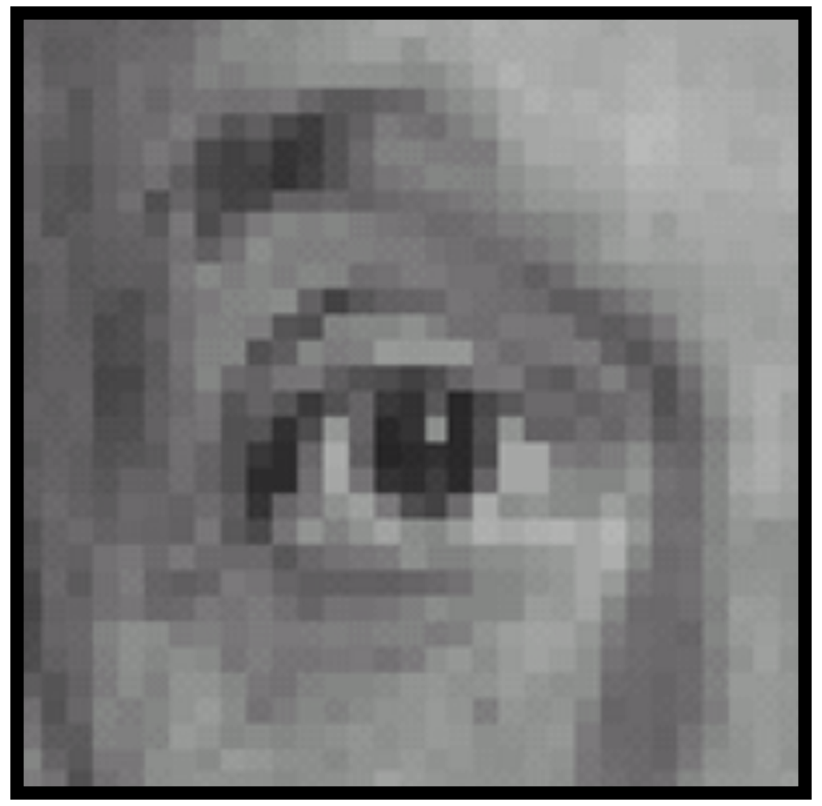
Original

0	0	0
0	1	0
0	0	0

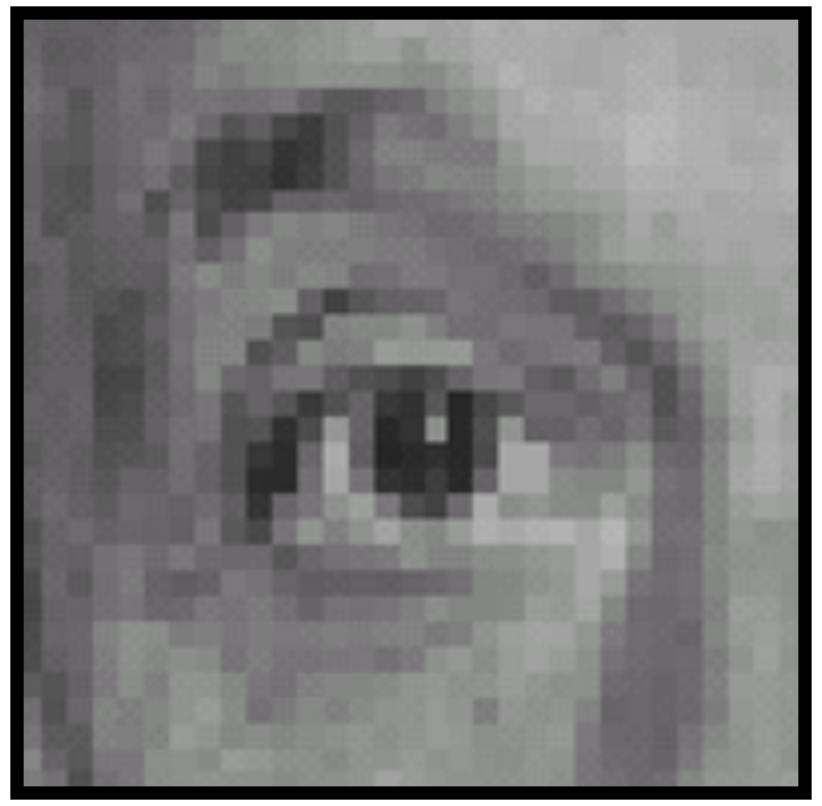
?

Source: D. Lowe

Practice with linear filters



0	0	0
0	1	0
0	0	0

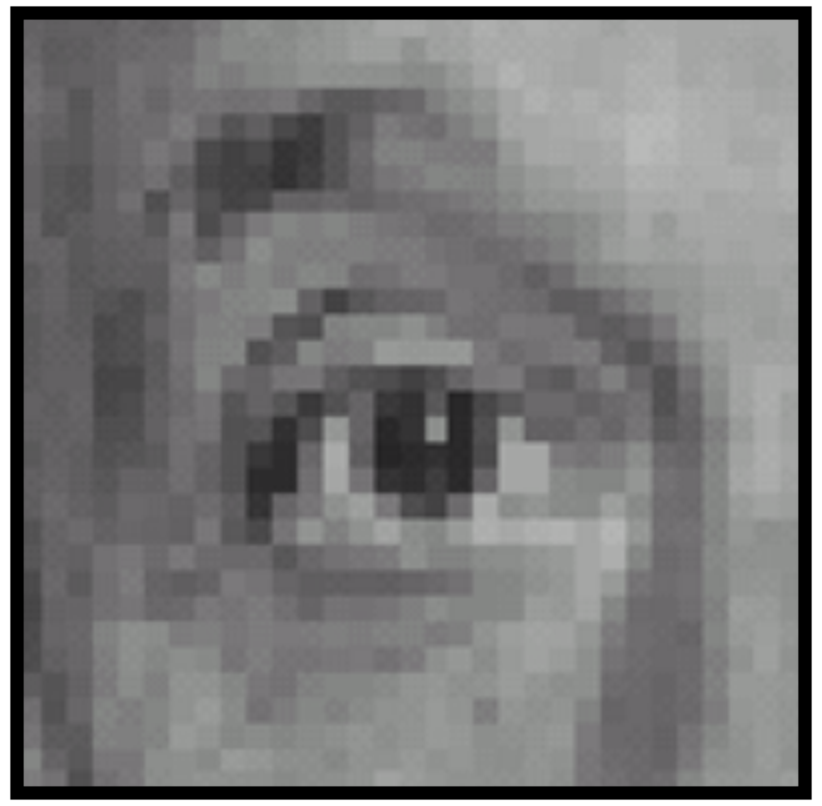


Original

Filtered
(no change)

Source: D. Lowe

Practice with linear filters



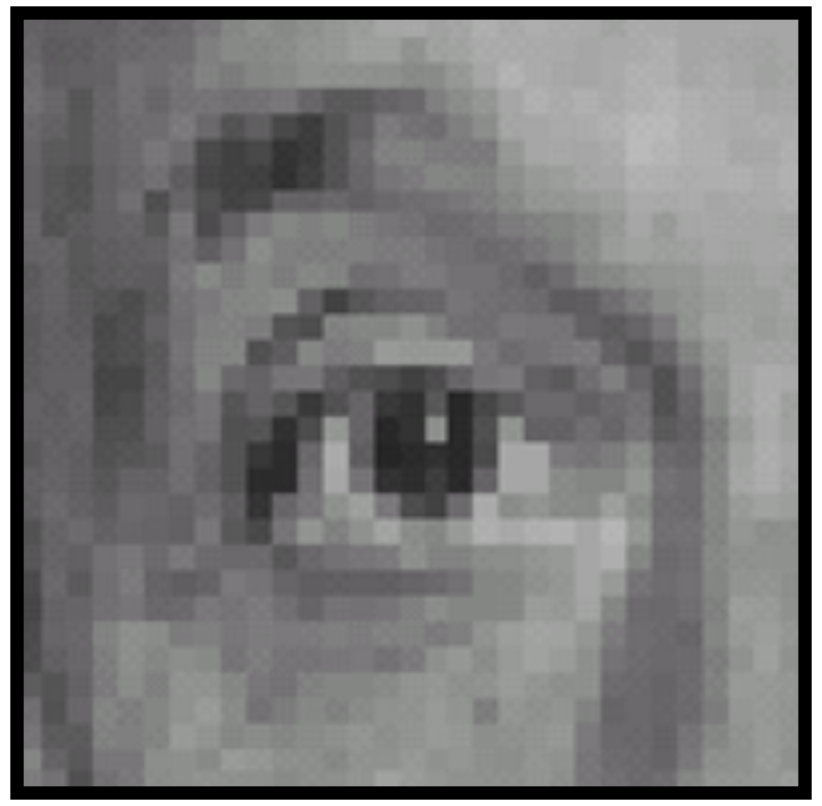
Original

0	0	0
0	0	1
0	0	0

?

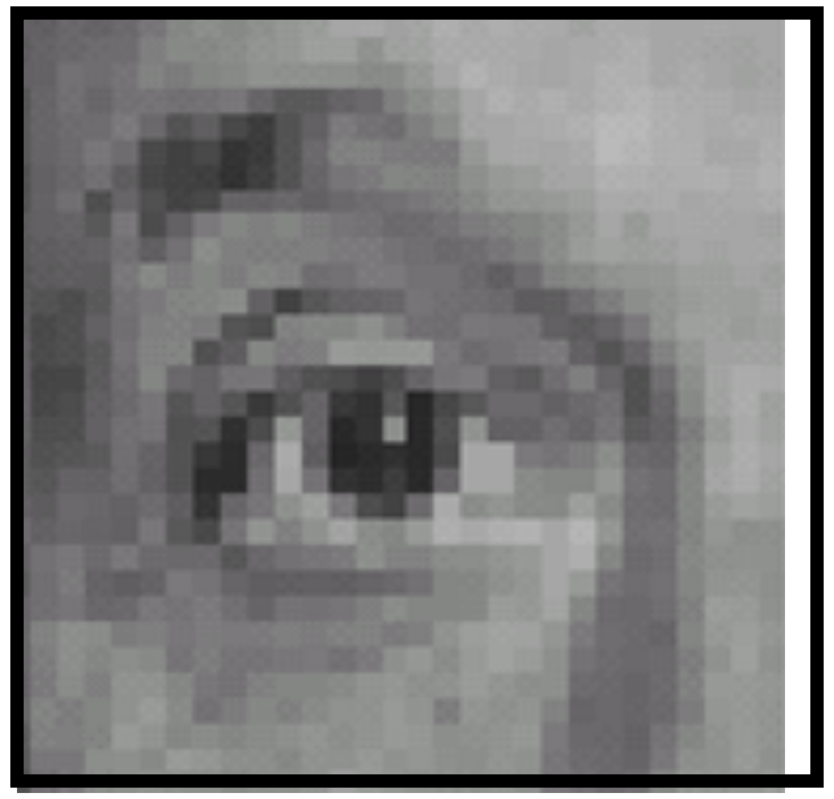
Source: D. Lowe

Practice with linear filters



Original

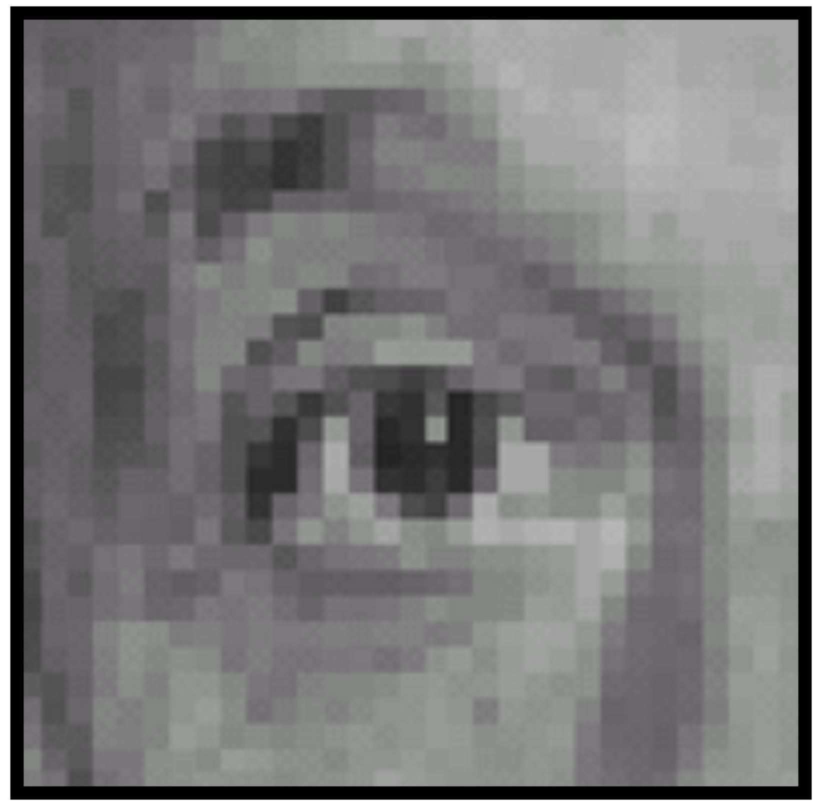
0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters



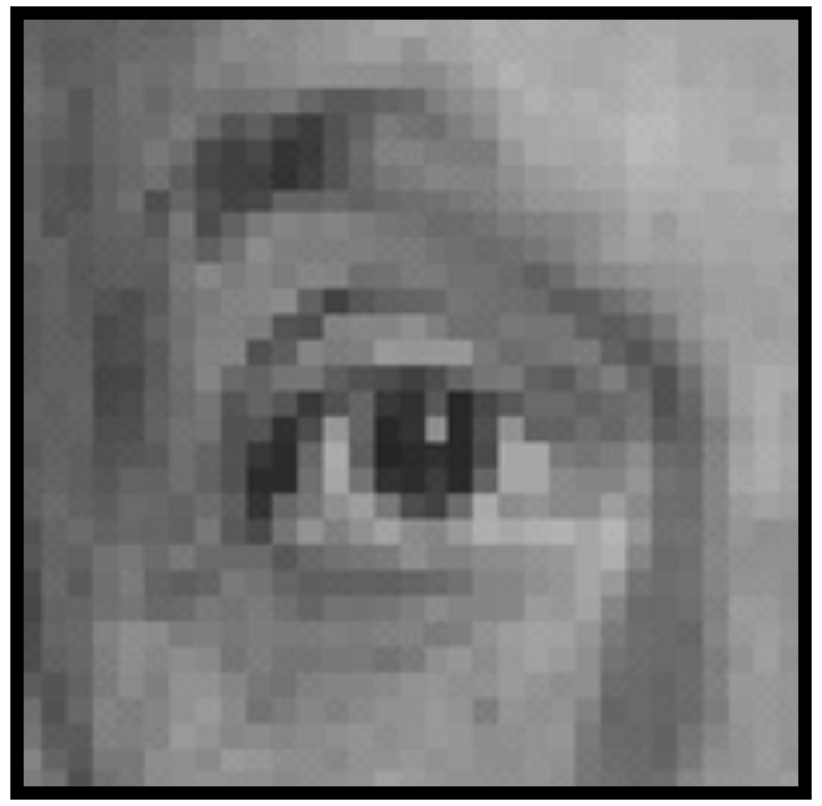
Original

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

?

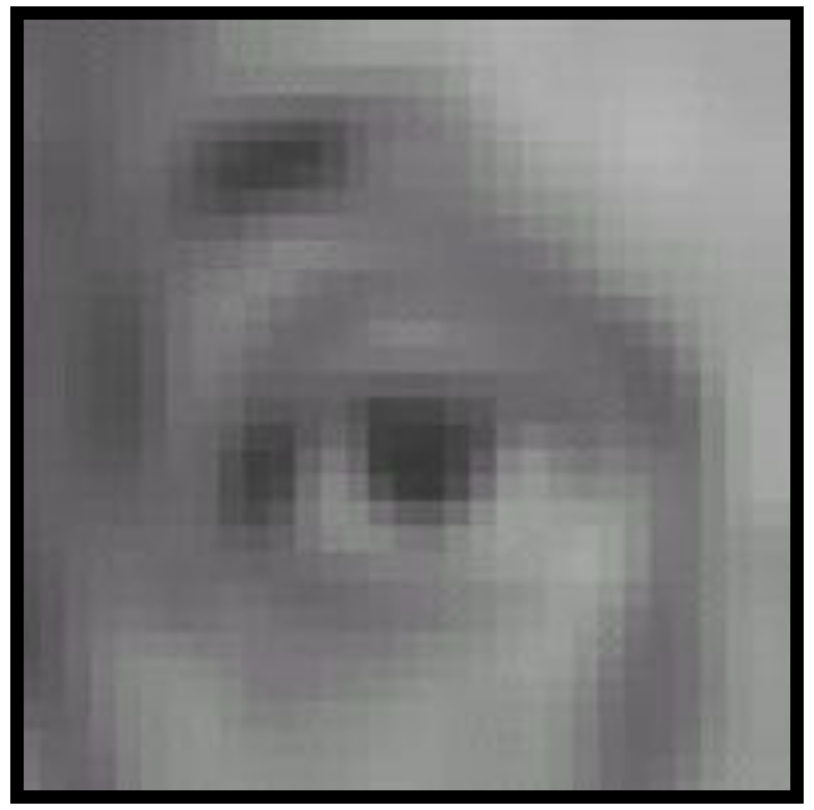
Source: D. Lowe

Practice with linear filters



Original

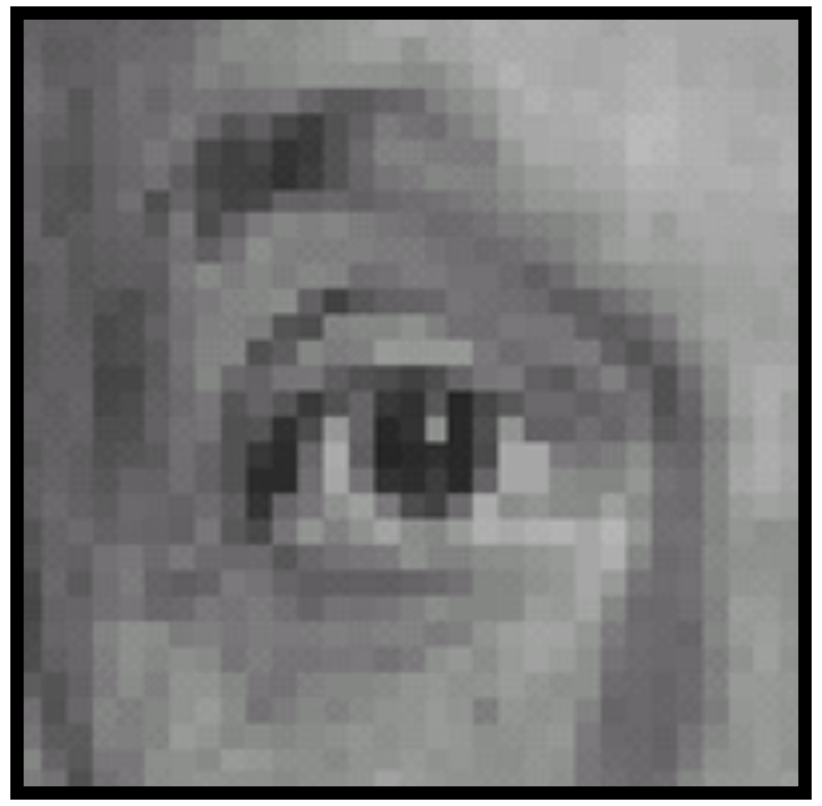
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



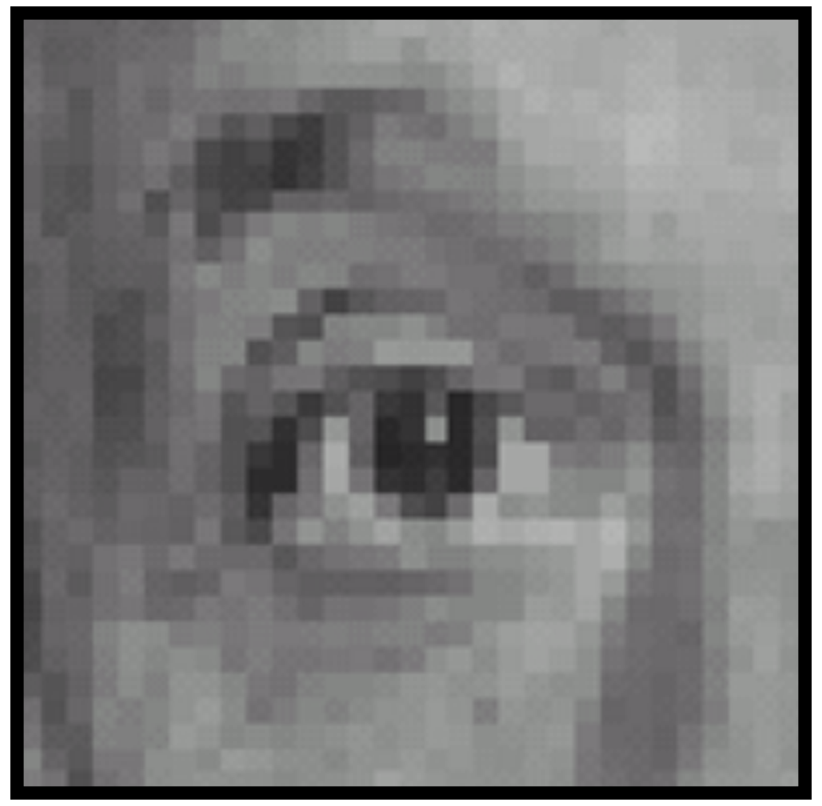
Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters



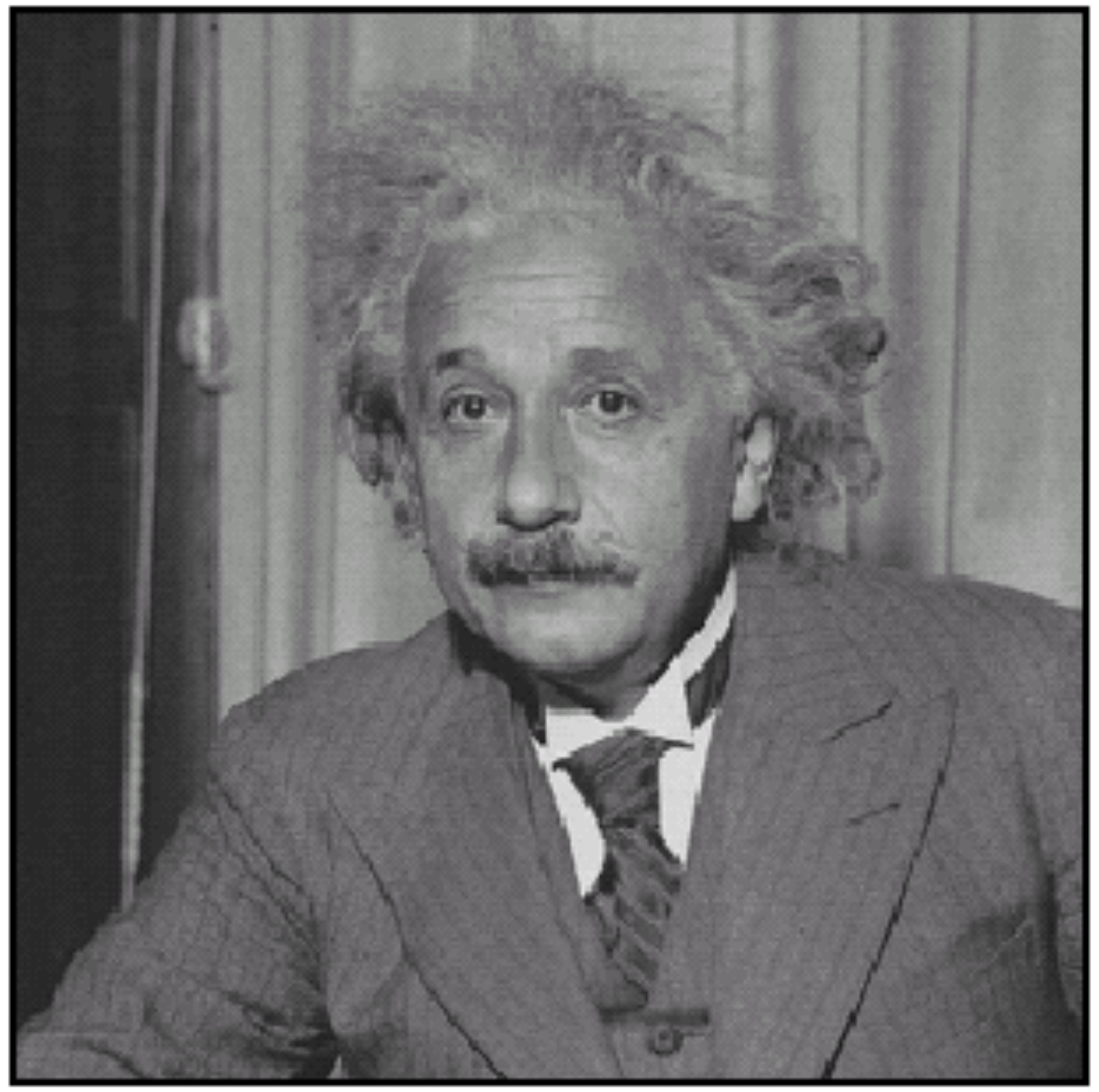
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

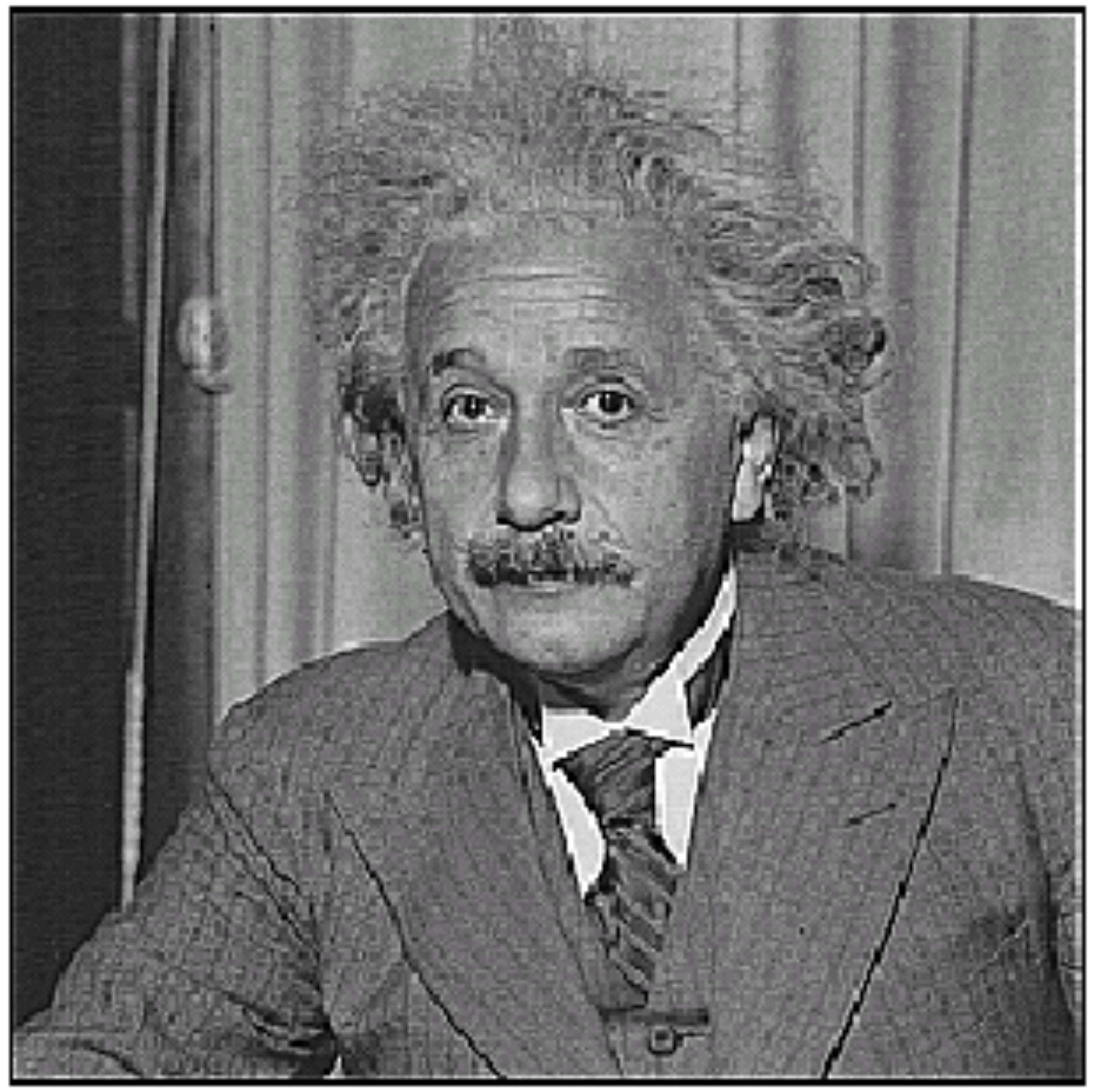
Sharpening filter: accentuates differences with local average

Source: D. Lowe

Sharpening



before



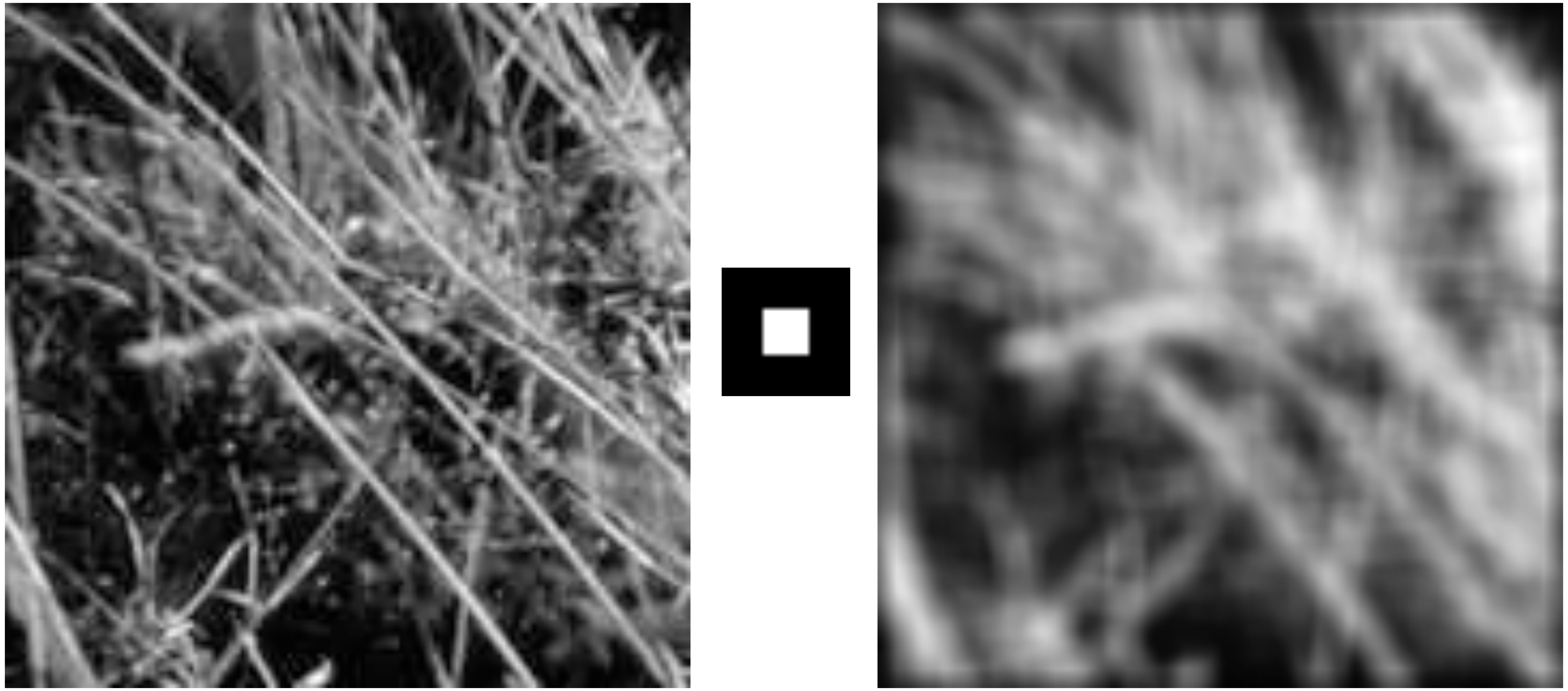
after

Source: D. Lowe

Smoothing with box filter revisited

What's wrong with this picture?

What's the solution?



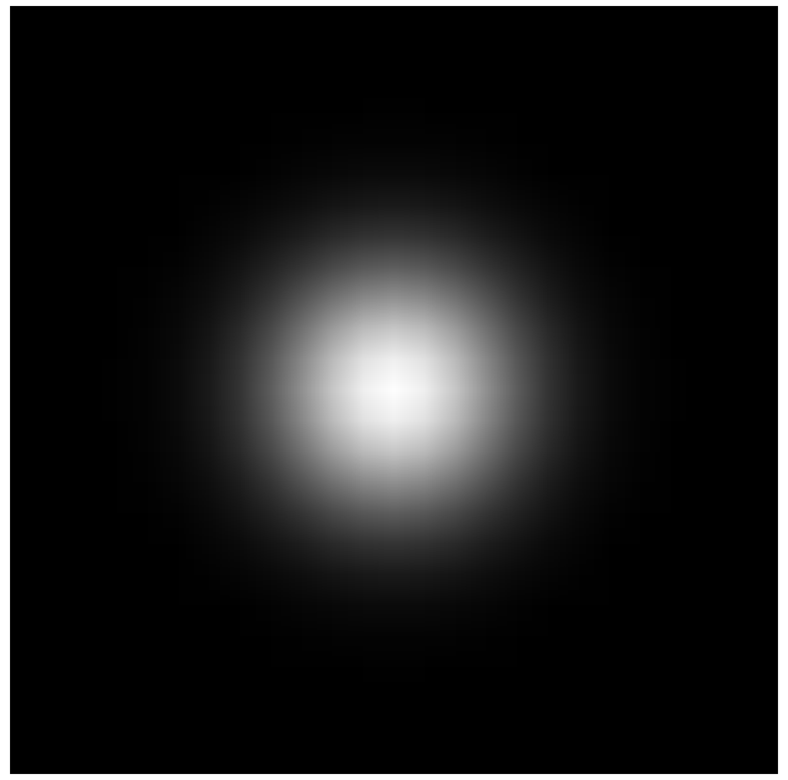
Source: D. Forsyth

Smoothing with box filter revisited

What's wrong with this picture?

What's the solution?

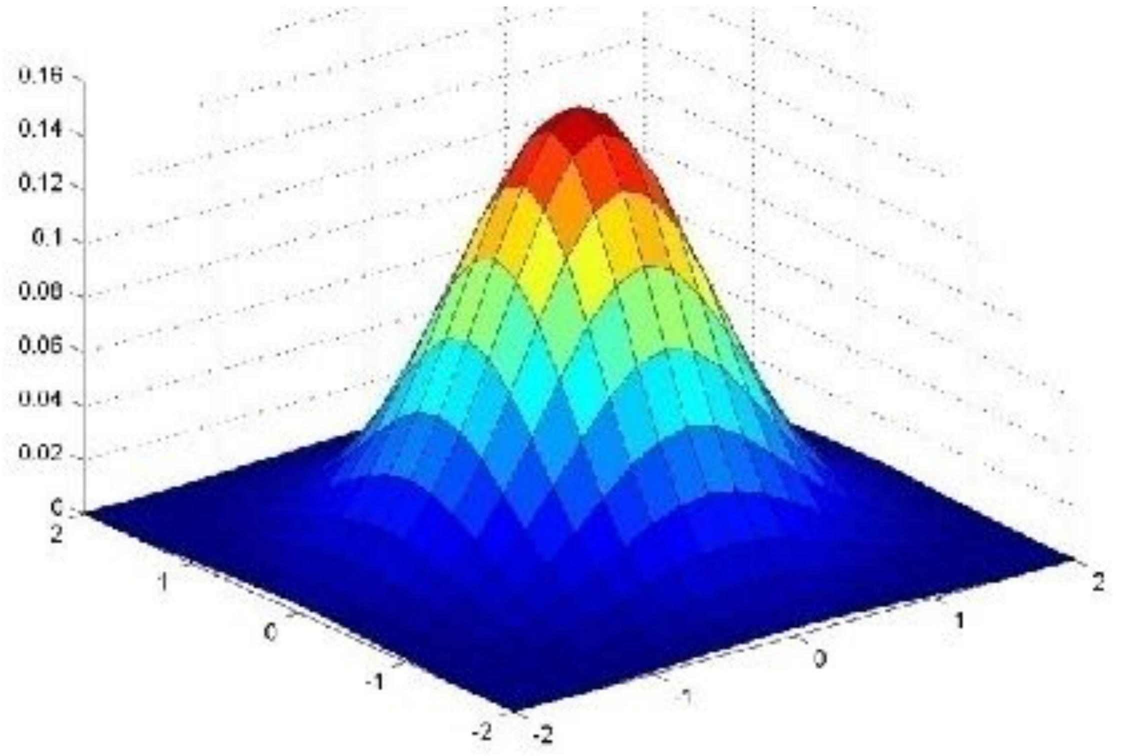
- To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

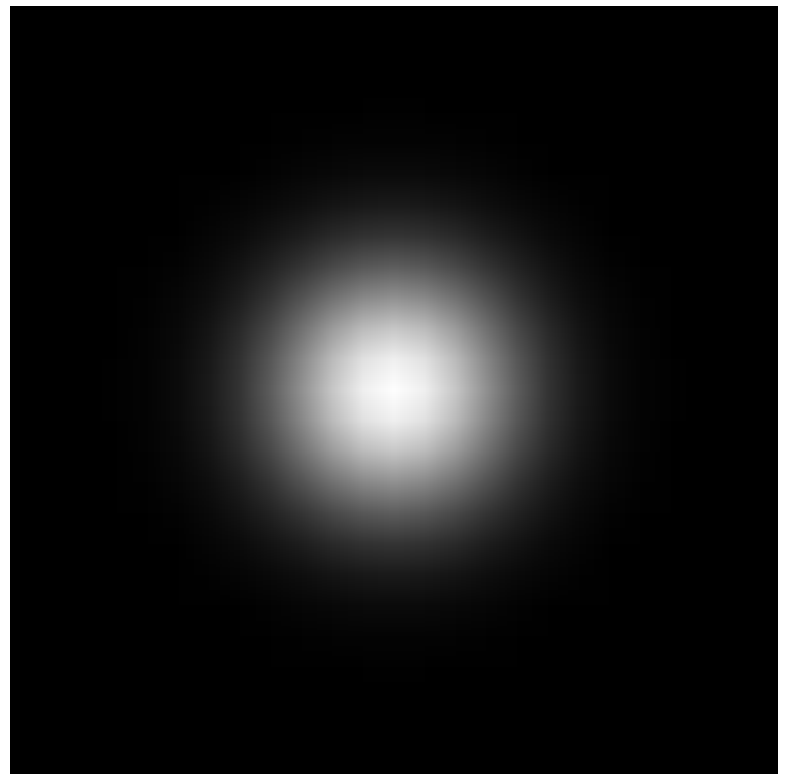


"fuzzy blob"

Gaussian kernel

Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should *renormalize* weights to sum to 1 in any case)





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

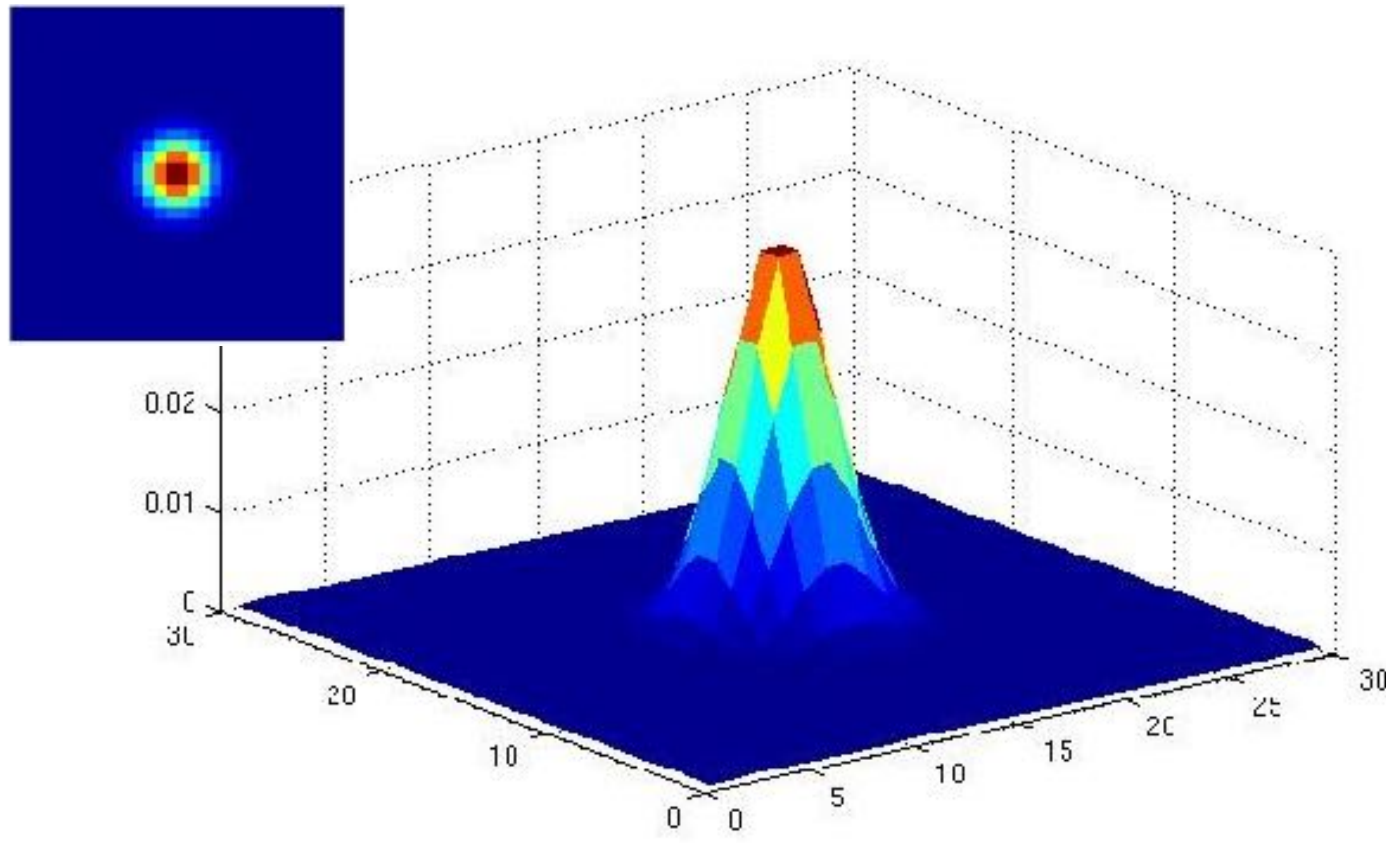
5 x 5, $\sigma = 1$

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

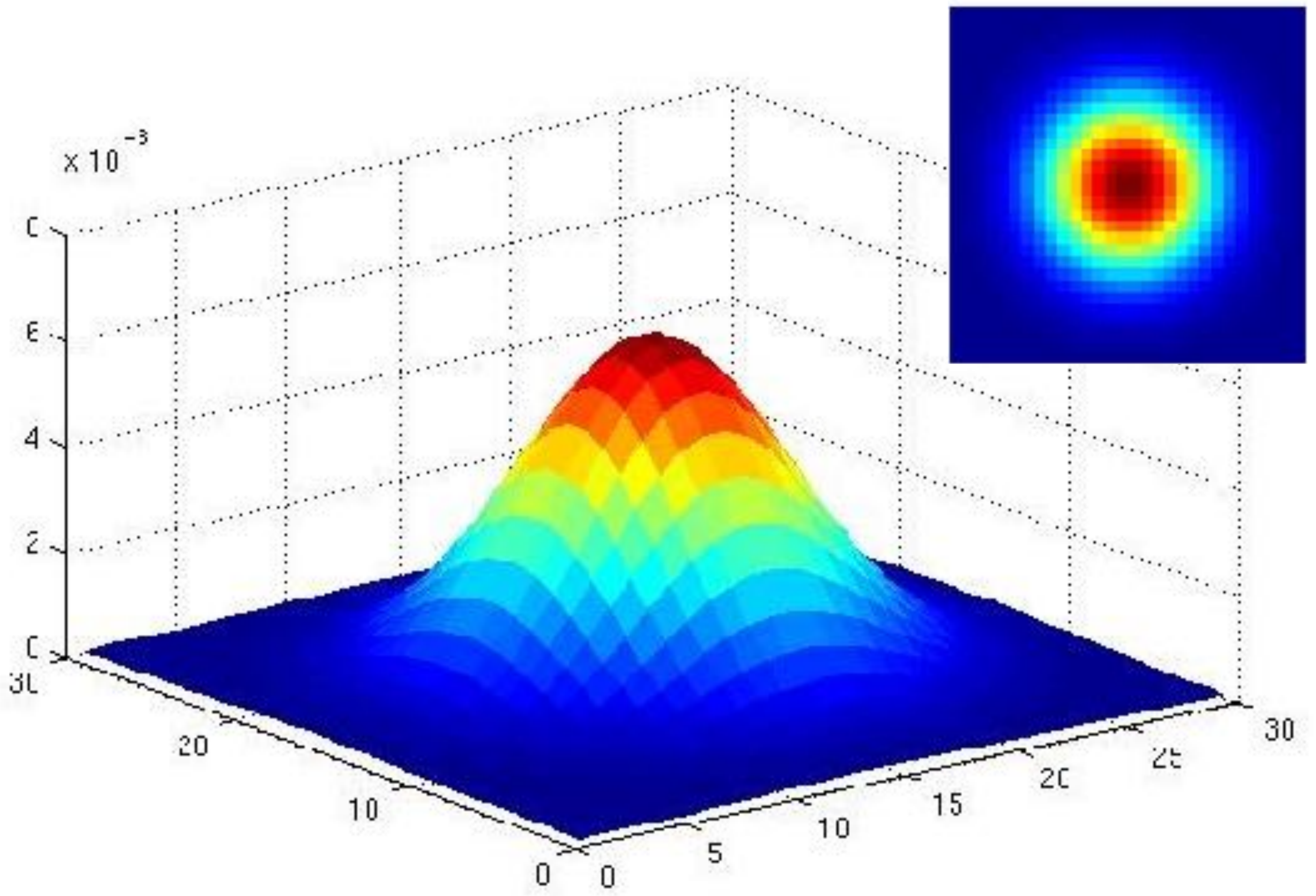
Source: C. Rasmussen

Gaussian kernel

Standard deviation σ: determines extent of smoothing



σ = 2 with 30 x 30 kernel



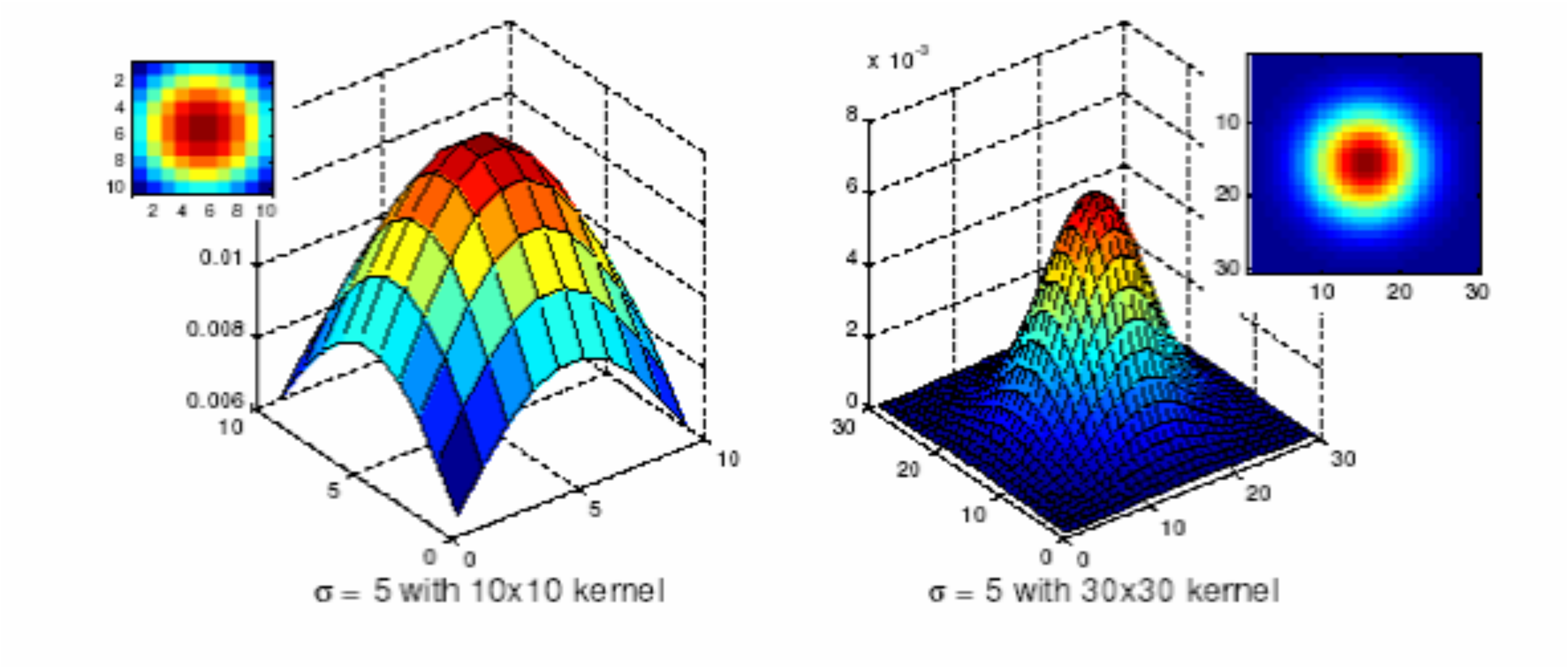
σ = 5 with 30 x 30 kernel

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

Source: K. Grauman

Choosing kernel width

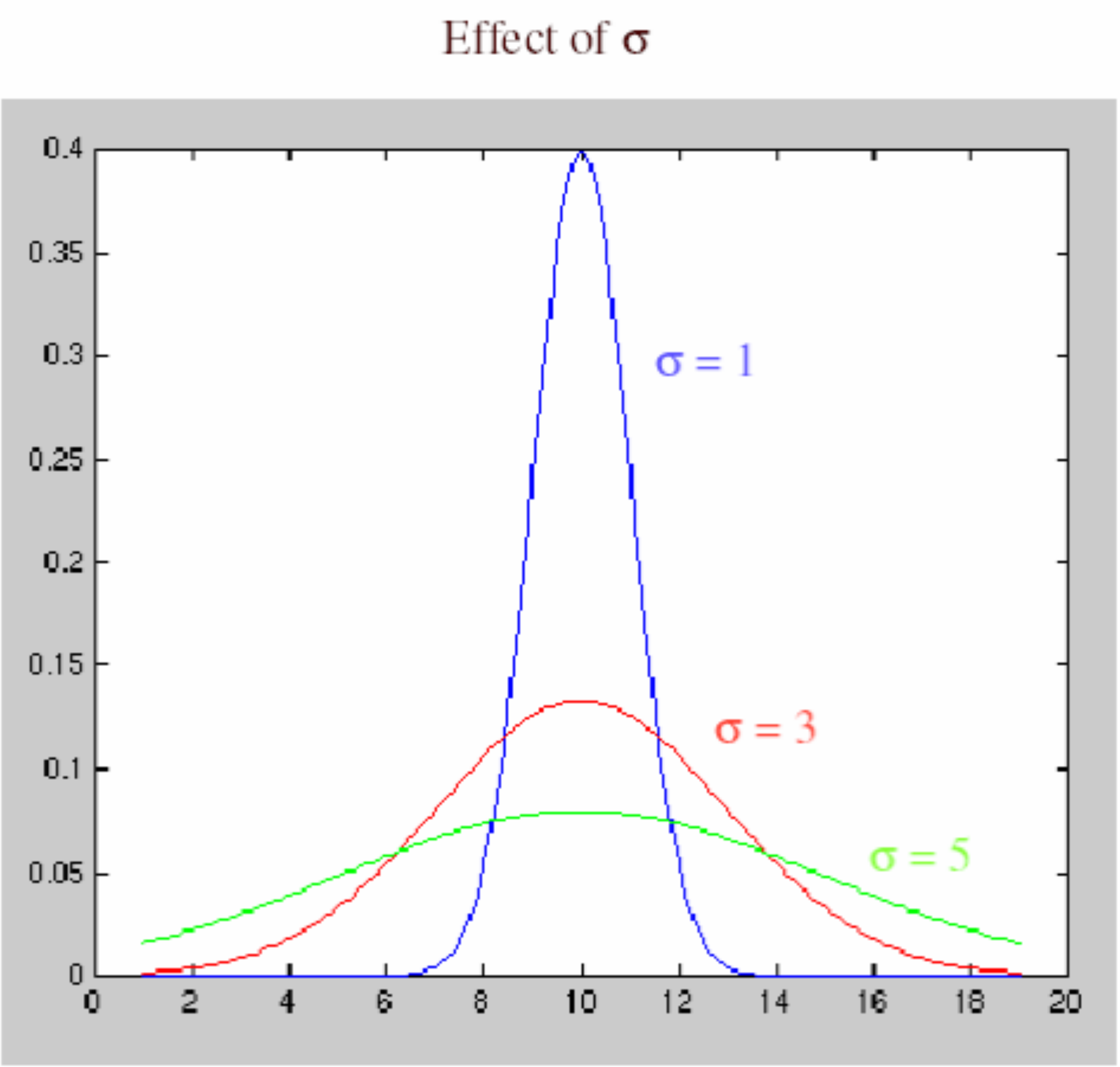
The Gaussian function has infinite support, but discrete filters use finite kernels



Source: K. Grauman

Choosing kernel width

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



Gaussian filters

Remove high-frequency components from the image (*low-pass filter*)

Convolution with self is another Gaussian

- So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
- Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$

Separable kernel

- Factors into product of two 1D Gaussians
- Discrete example:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

scipy.ndimage.gaussian_filter(*input*, *sigma*, *order=0*, *output=None*, *mode='reflect'*, *cval=0.0*, *truncate=4.0***)**

Separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

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

Why is separability useful?

Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among columns)

What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?

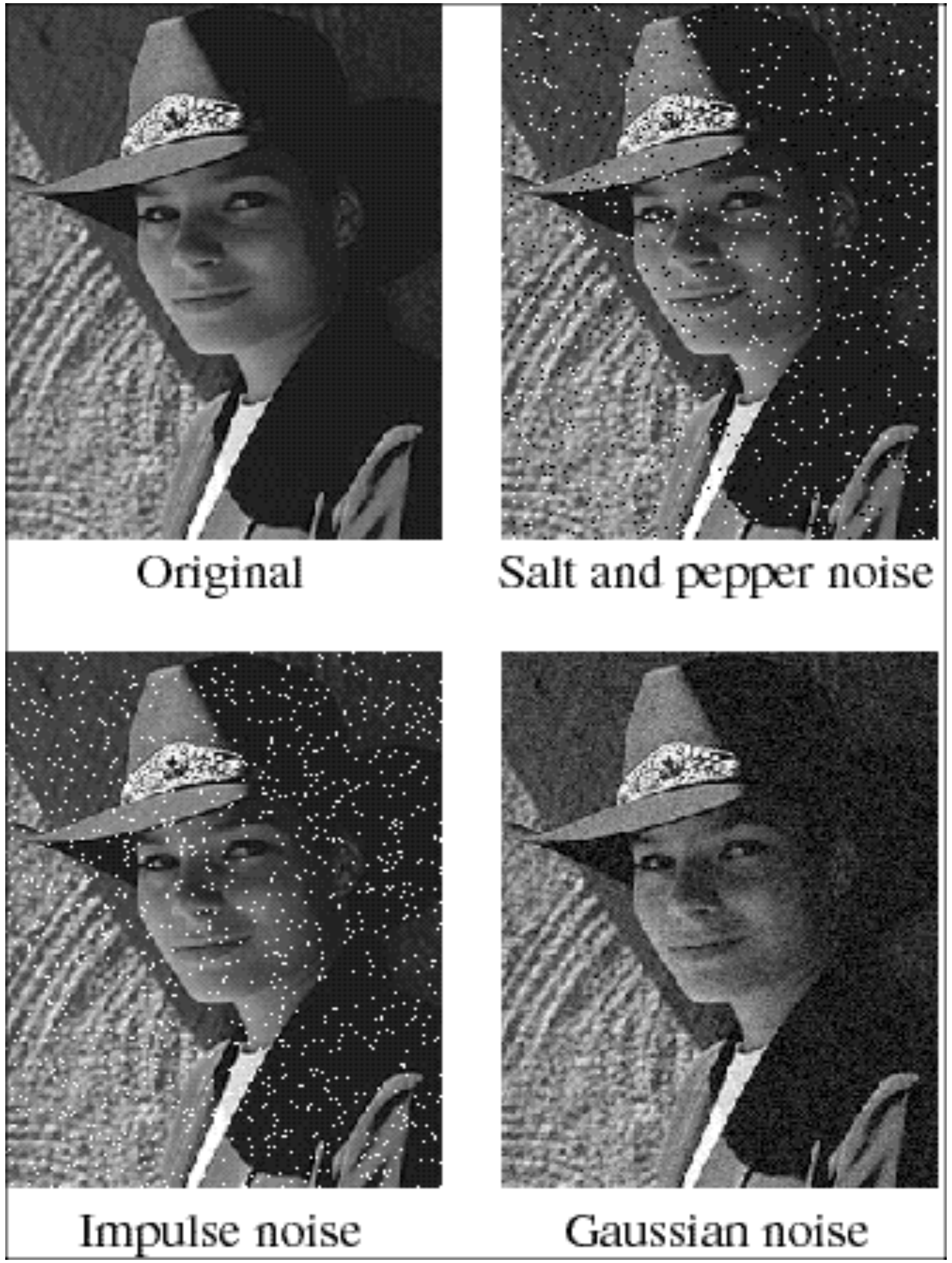
- $O(n^2 m^2)$

What if the kernel is separable?

- $O(n^2 m)$

Question: Is the box filter separable?

Types of noise



Salt and pepper noise: contains random occurrences of black and white pixels

Impulse noise: contains random occurrences of white pixels

Gaussian noise: variations in intensity drawn from a Gaussian normal distribution

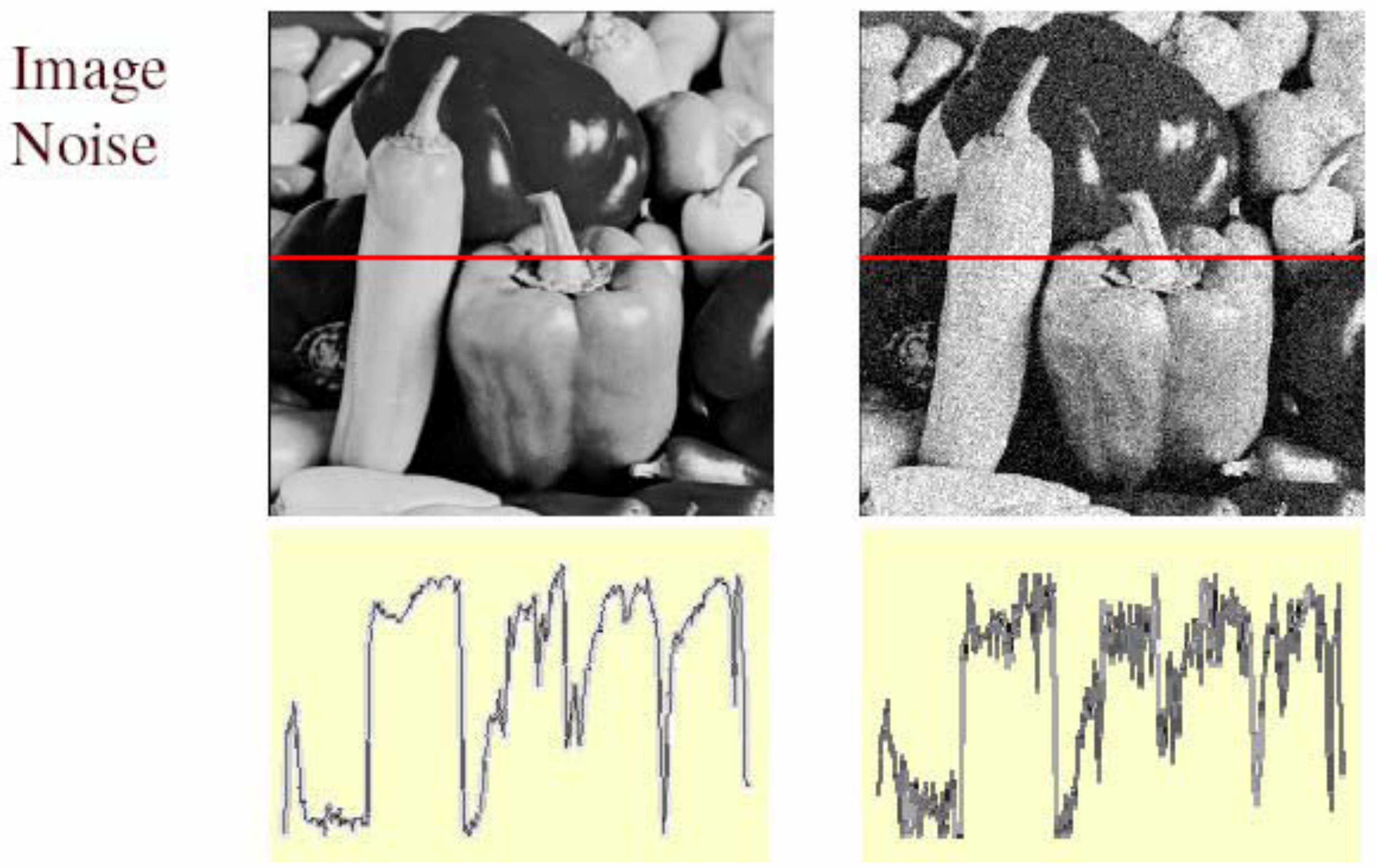
Source: S. Seitz

Gaussian noise

Mathematical model: sum of many independent factors

Good for small standard deviations

Assumption: independent, zero-mean noise

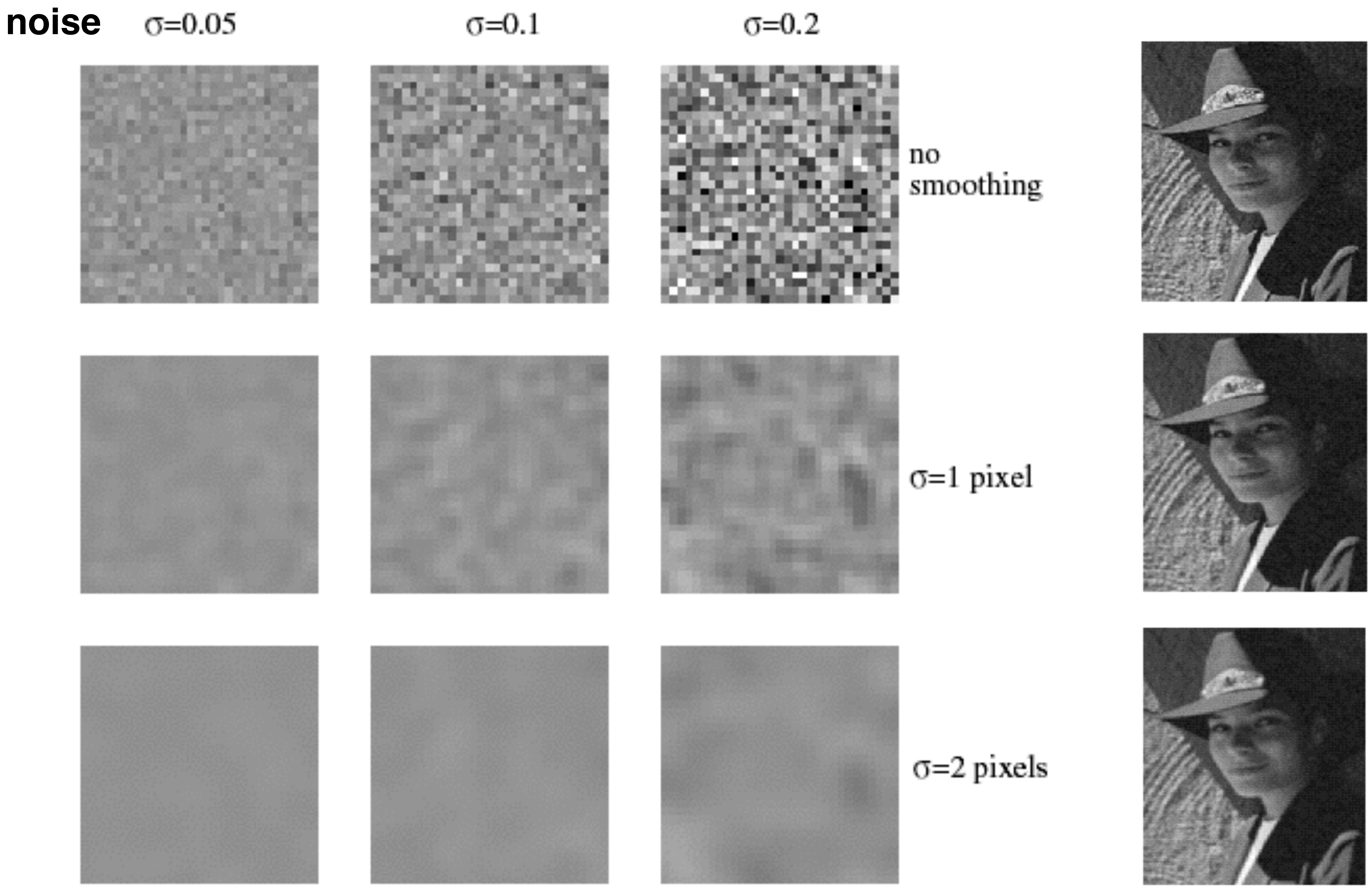


$$f(x,y) = \overbrace{\widehat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

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

Source: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise

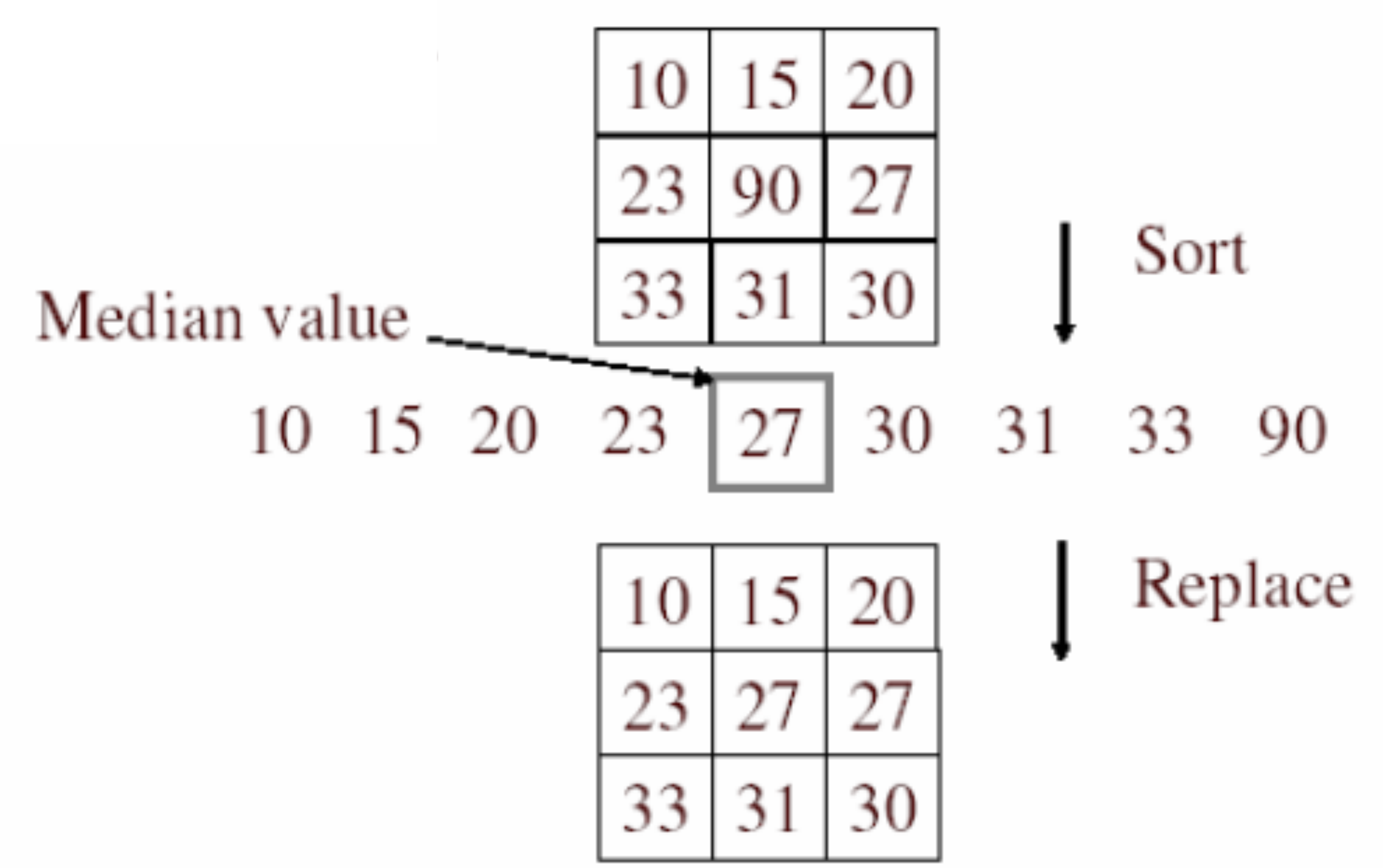
What’s wrong with the results?

3x3 5x5 7x7

Gaussian smoothing with increasing standard deviation

Alternative idea: Median filtering

A **median filter** operates over a window by selecting the median intensity in the window



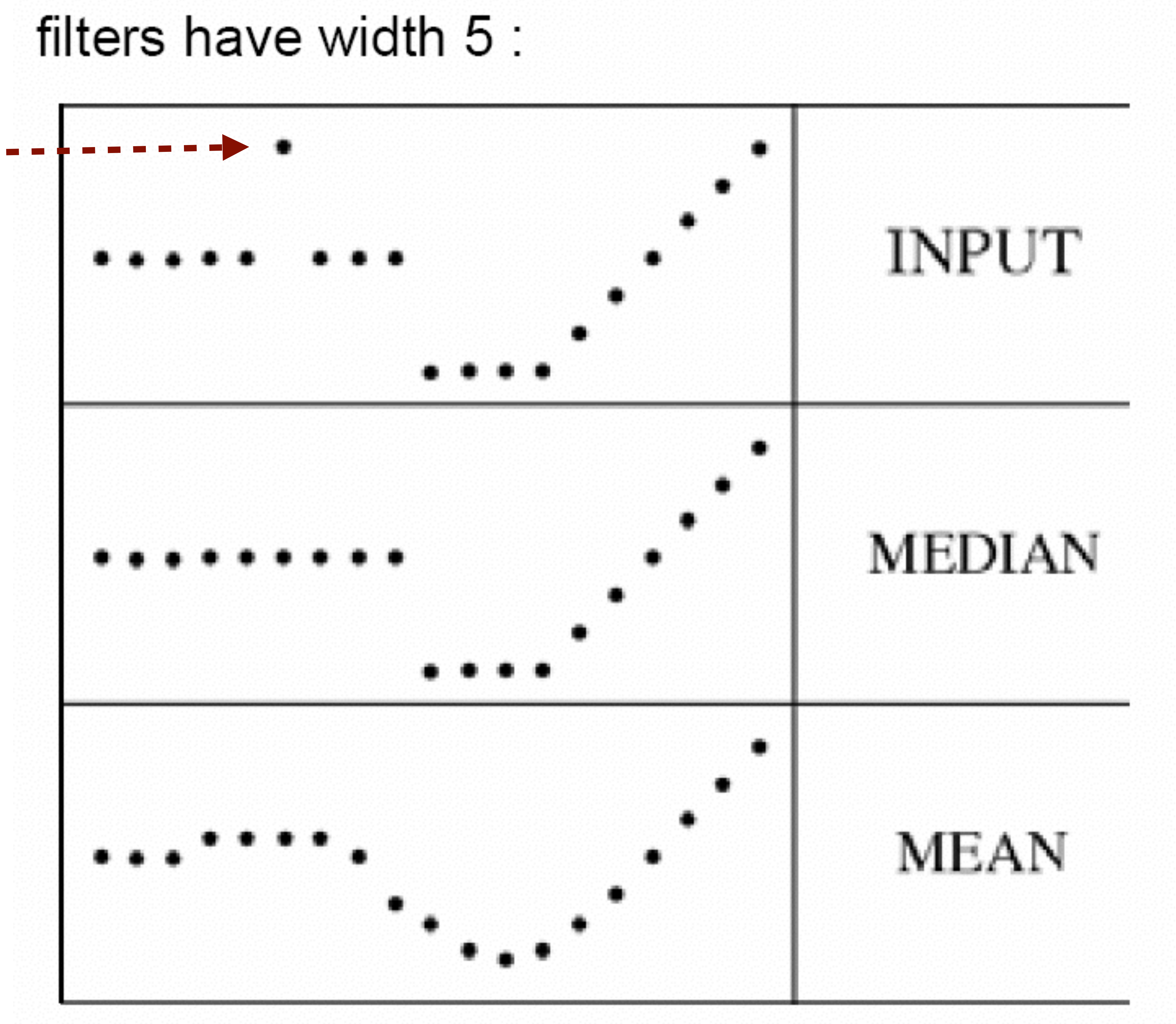
Question: is median filtering linear?

Source: K. Grauman

Median filter

What advantage does median filtering have over Gaussian filtering?

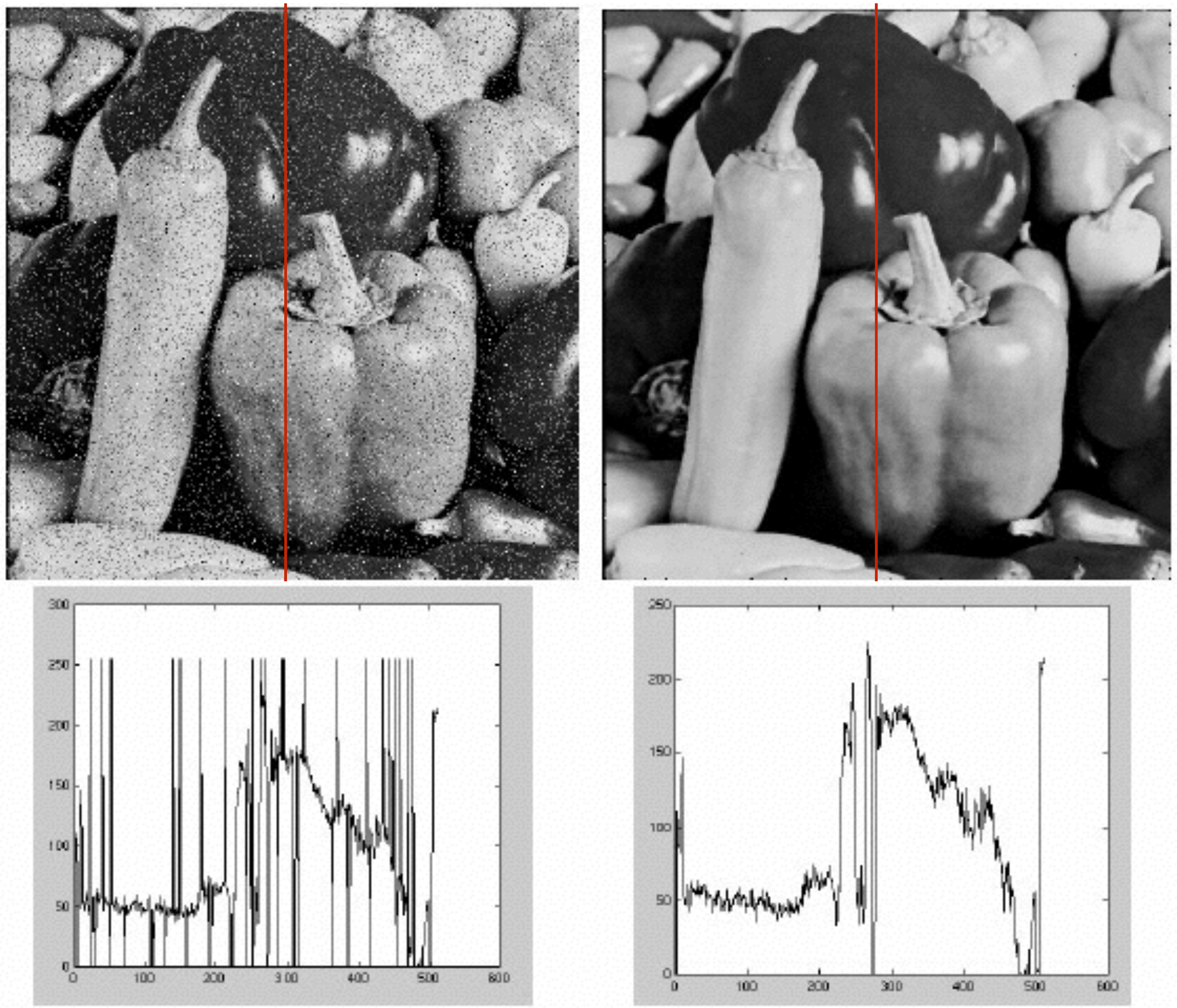
Robustness to outliers



Source: K. Grauman

Median filter

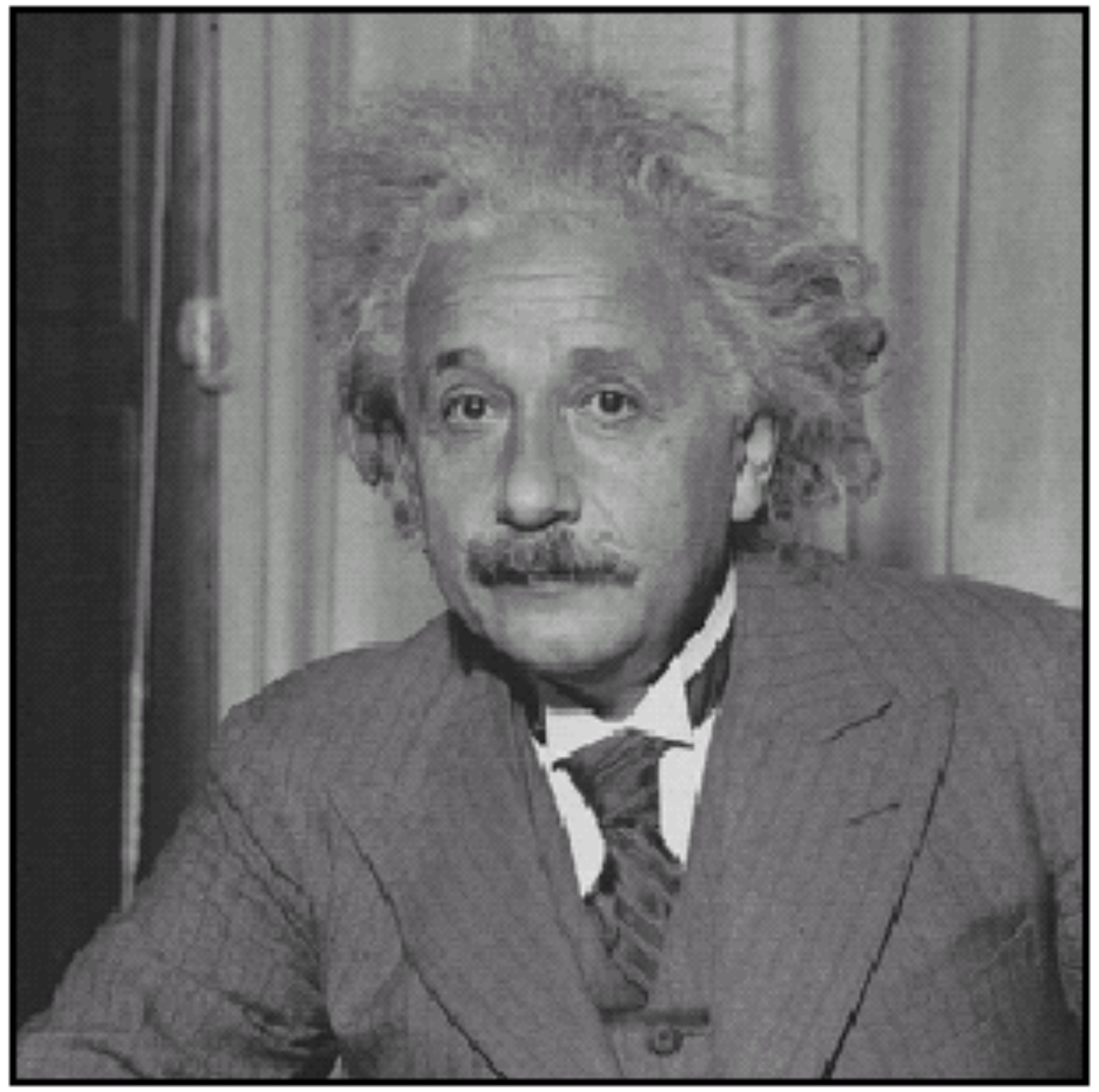
Salt-and-pepper noise　　　　Median filtered



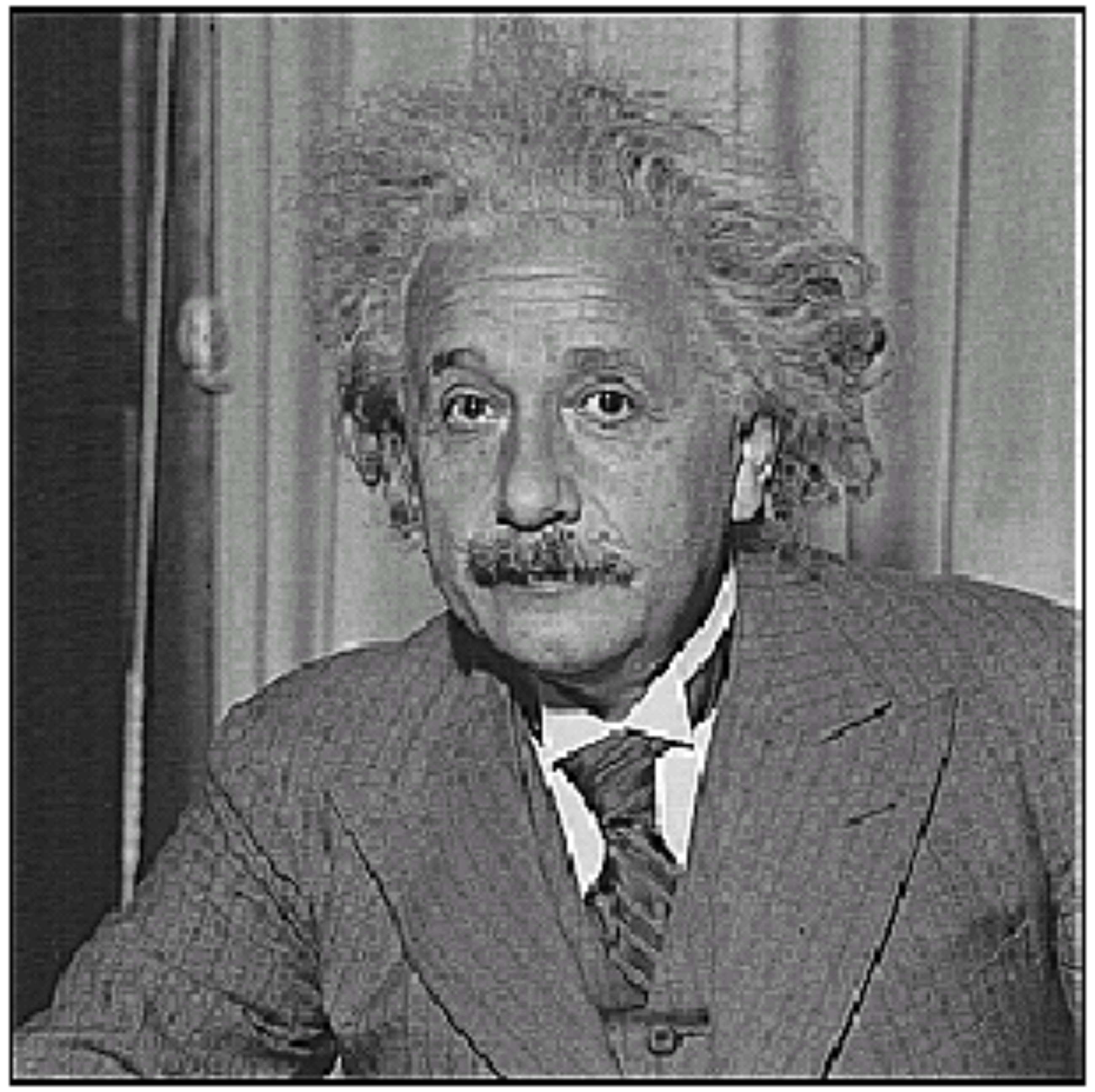
scipy.ndimage.median_filter(input, size=None, footprint=None, output=None, mode='reflect', cval=0.0, origin=0)

Source: M. Hebert

Sharpening



before



after

Source: D. Lowe

Sharpening

What does blurring take away?

–

=

Let's add it back:

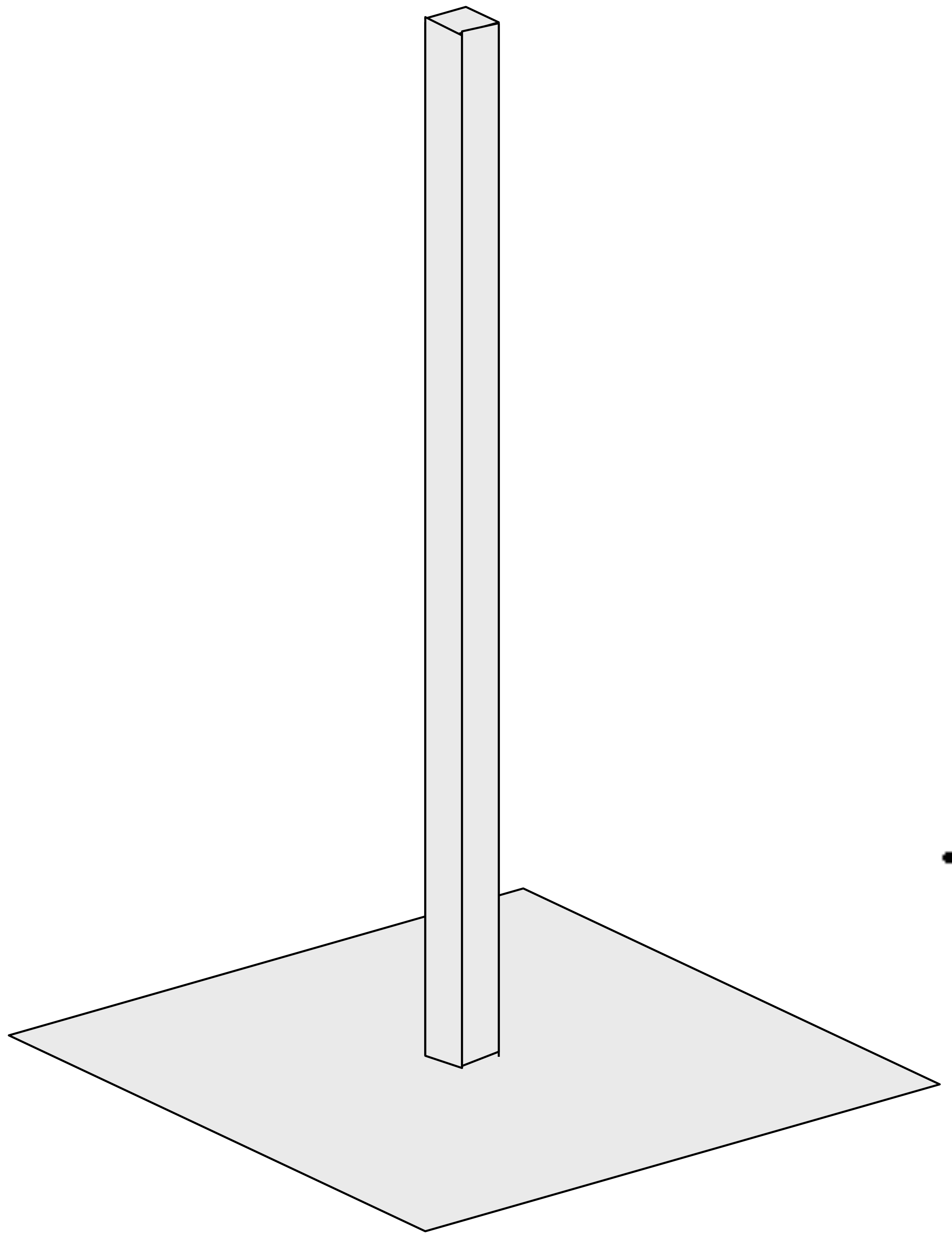
$+ \alpha$

=

Sharpening filter

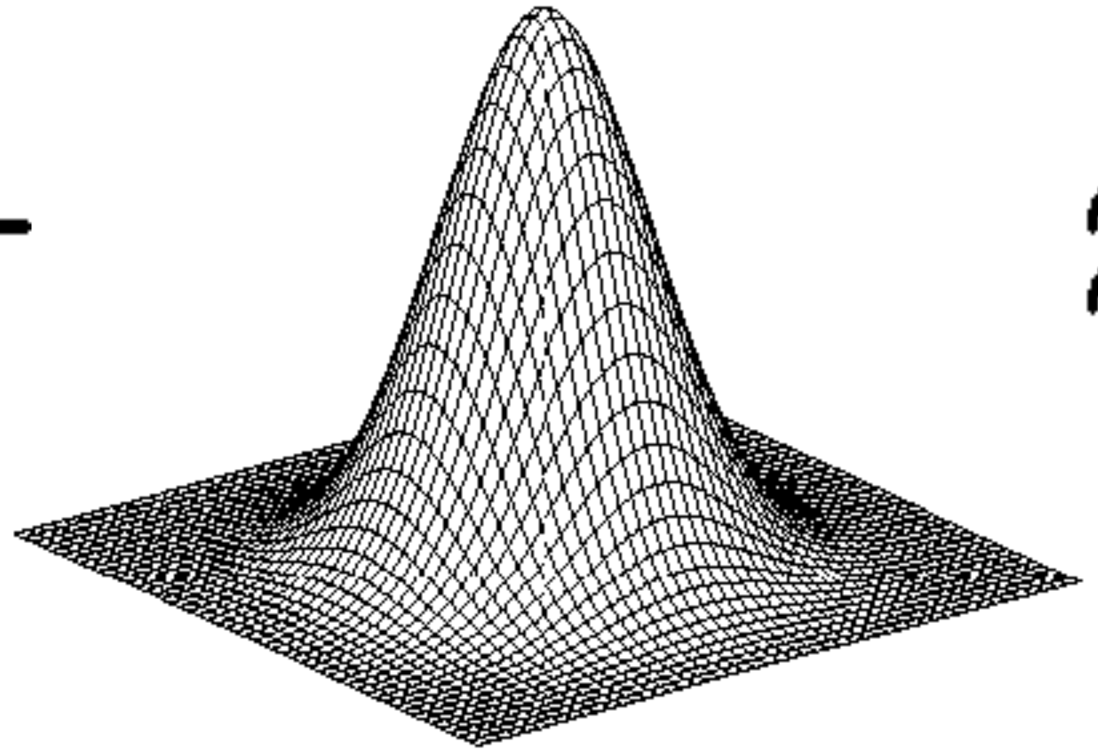
$$I = blurry(I) + sharp(I)$$

$$\begin{aligned} sharp(I) &= I - blurry(I) \\ &= I * e - I * g_\sigma \\ &= I * (e - g_\sigma) \end{aligned}$$



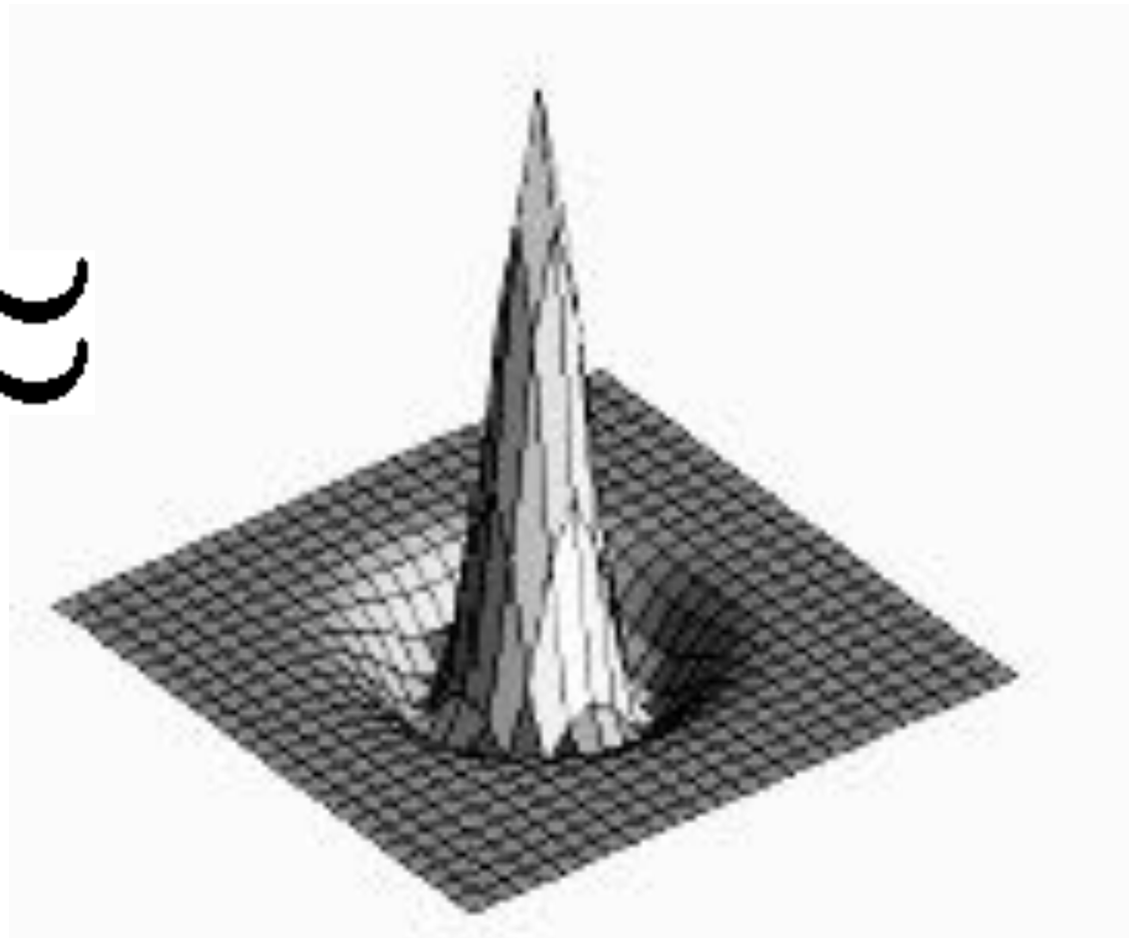
unit impulse

$-$



Gaussian

$\approx$

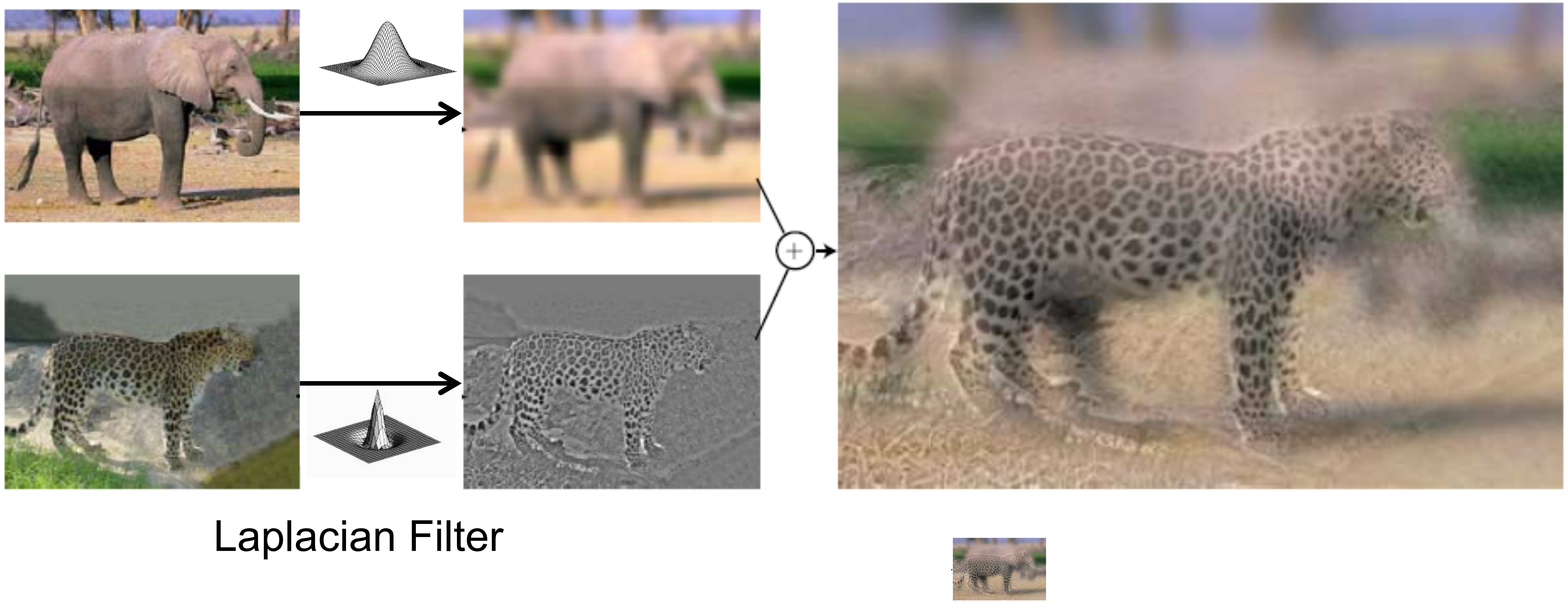


Laplacian of Gaussian

Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns, [“Hybrid Images,”]() SIGGRAPH 2006

Gaussian Filter

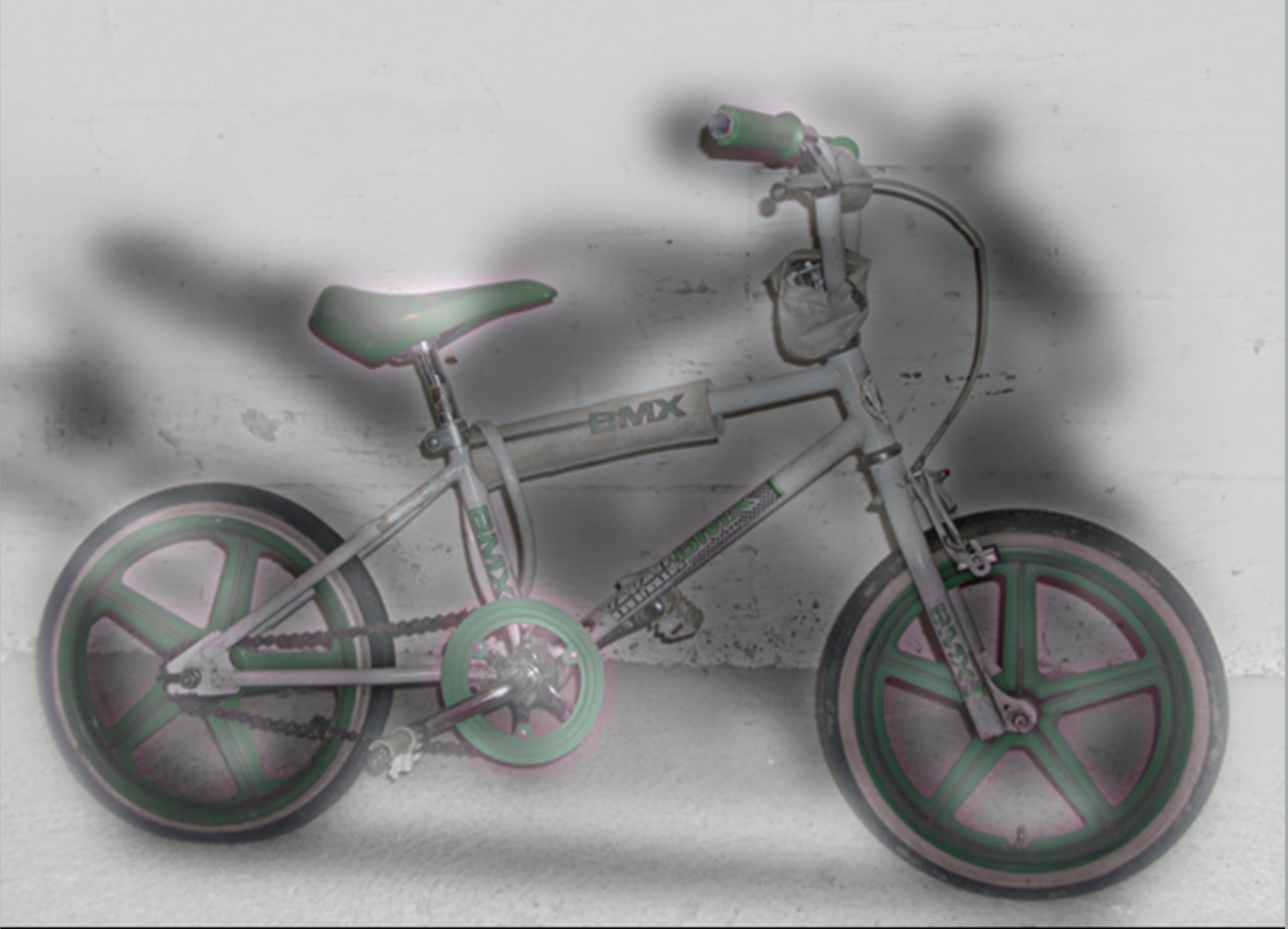


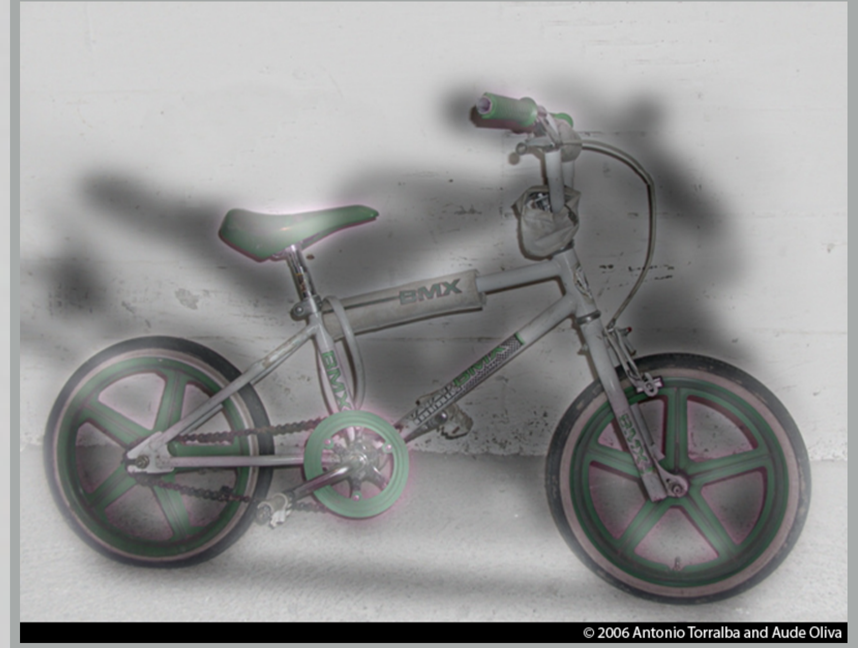
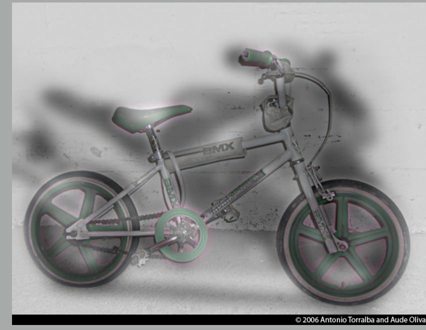
Laplacian Filter

Changing expression

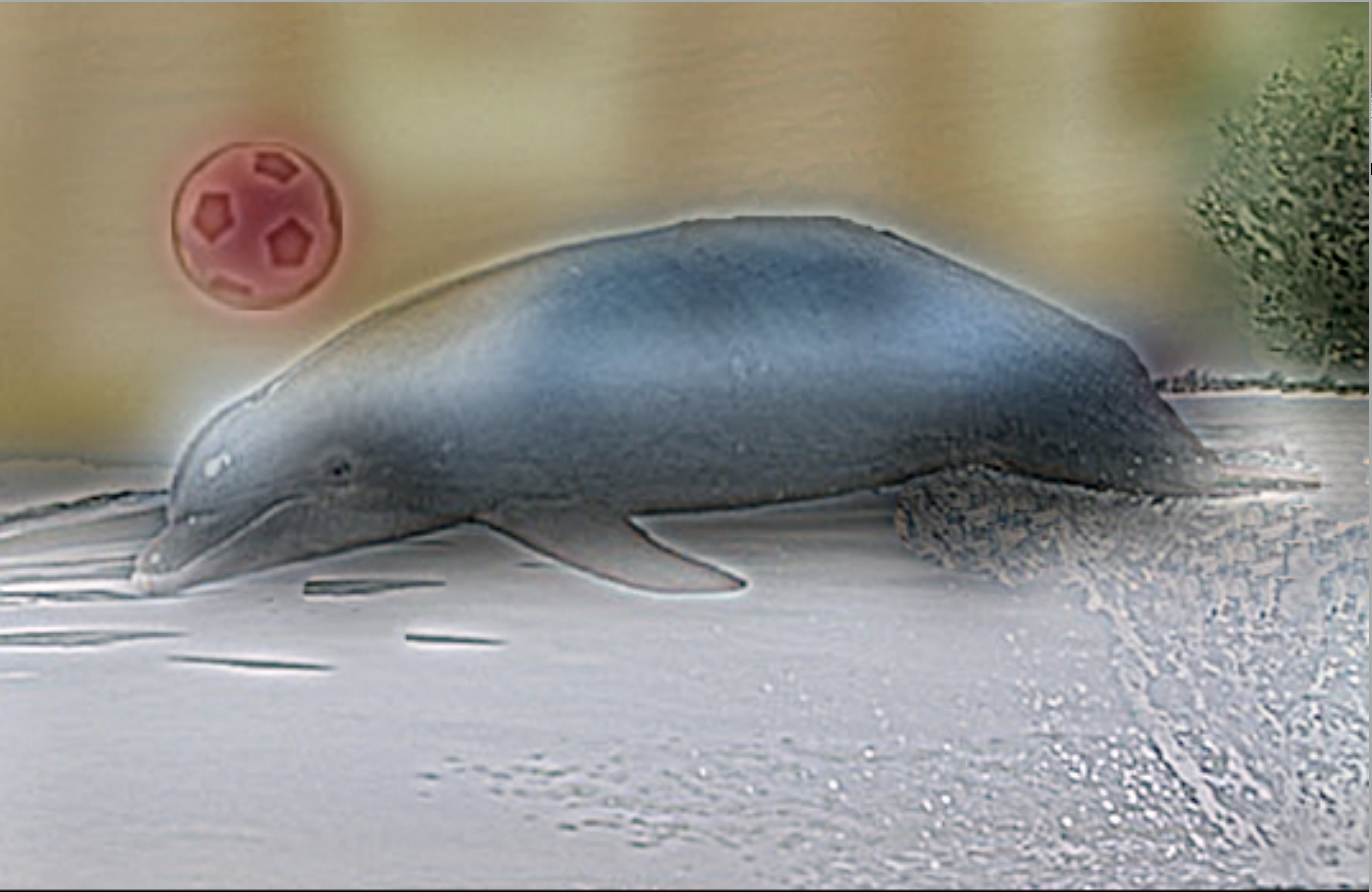
SIGGRAPH2006

Sad ⟷ Surprised



motorcycle and bicycle



dolphin and car

Edge detection

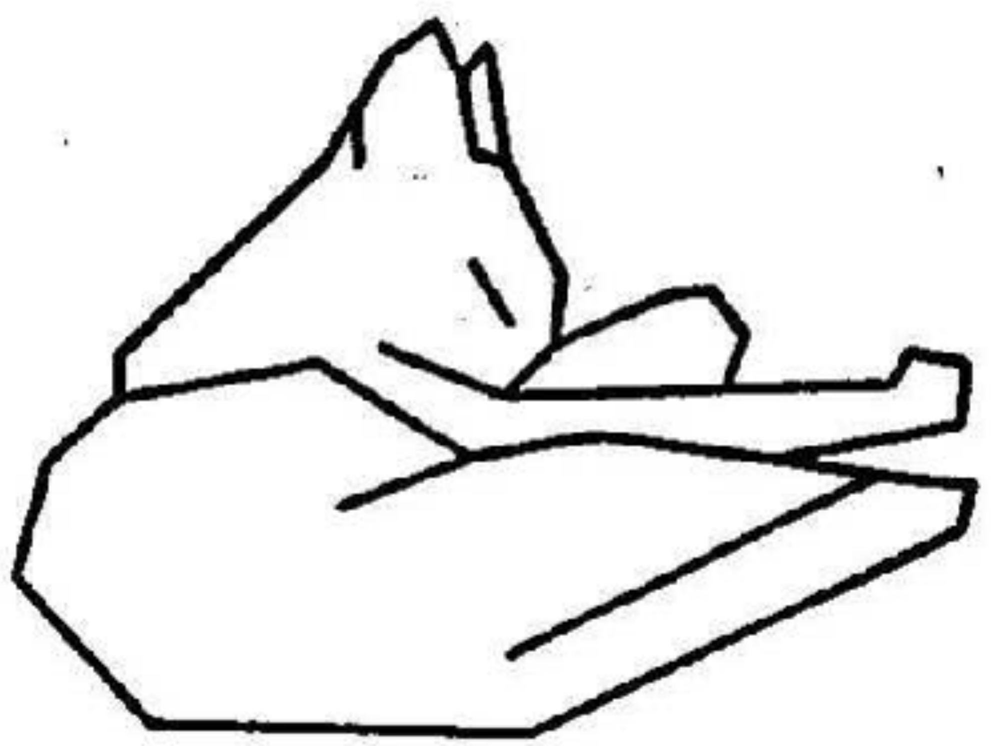
[Winter in Kraków photographed by Marcin Ryczek]

Edge detection

Goal: Identify sudden changes (discontinuities) in an image

- Intuitively, most semantic and shape information from the image can be encoded in the edges
- More compact than pixels

Ideal: artist's line drawing (but artist is also using object-level knowledge)

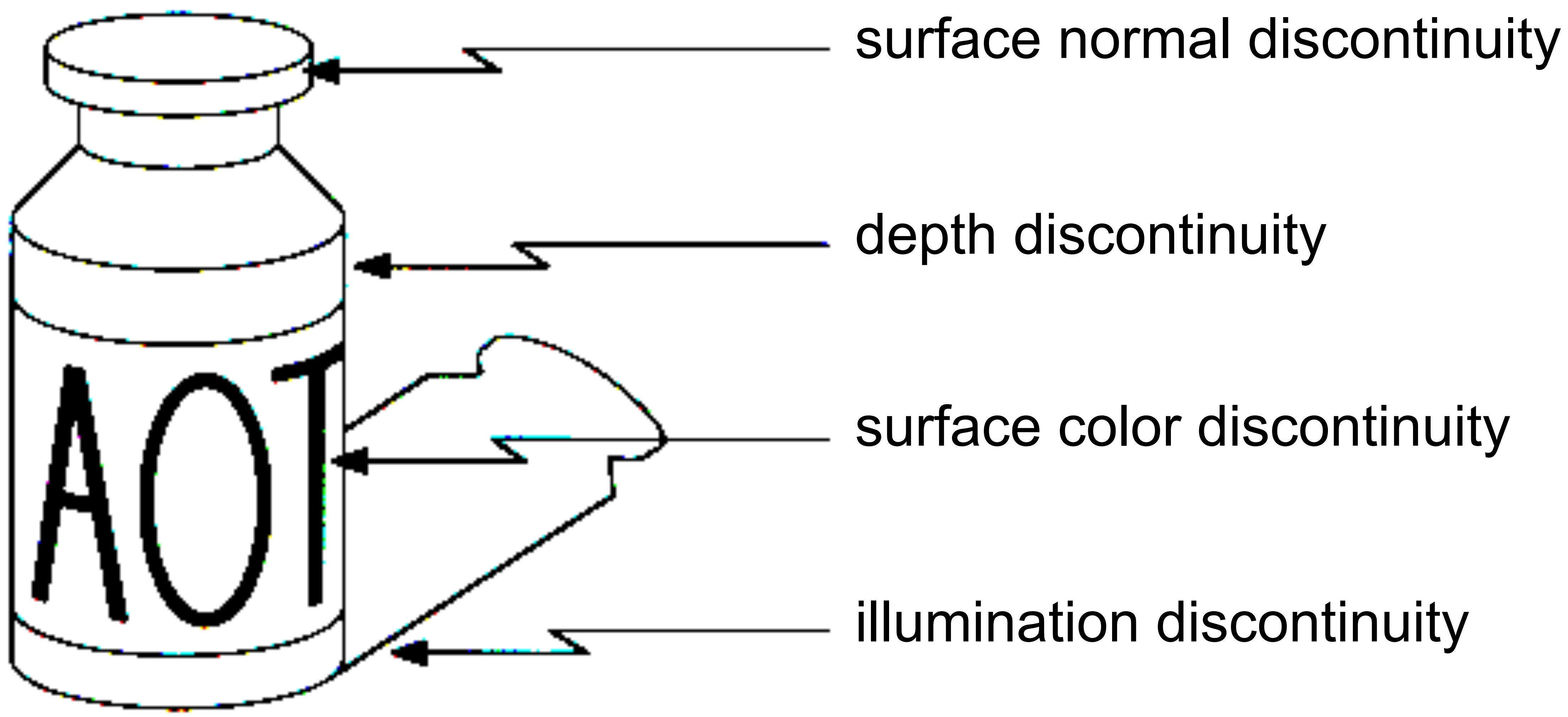


Attneave's Cat (1954)

Source: D. Lowe

Origin of edges

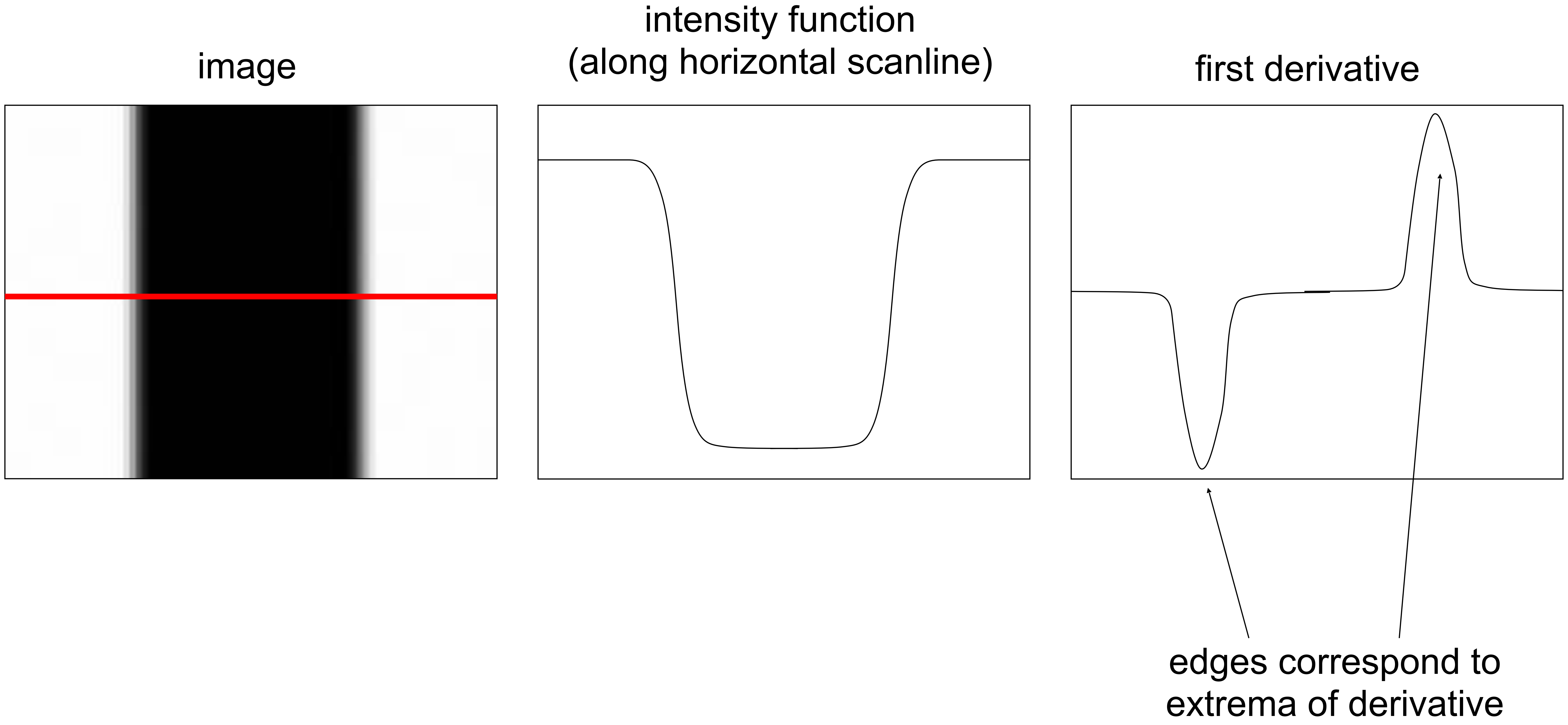
Edges are caused by a variety of factors:



Source: Steve Seitz

Edge detection

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

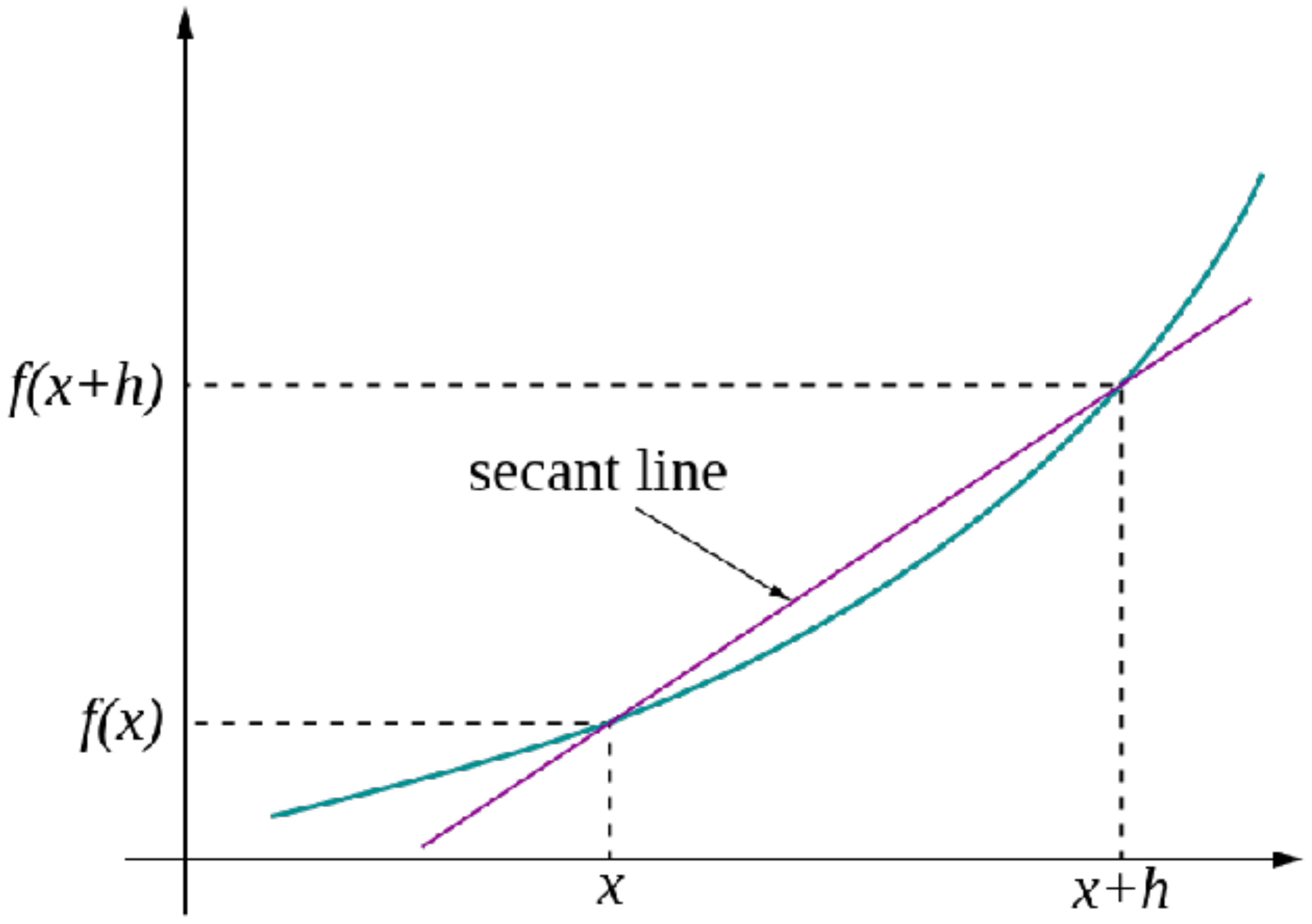


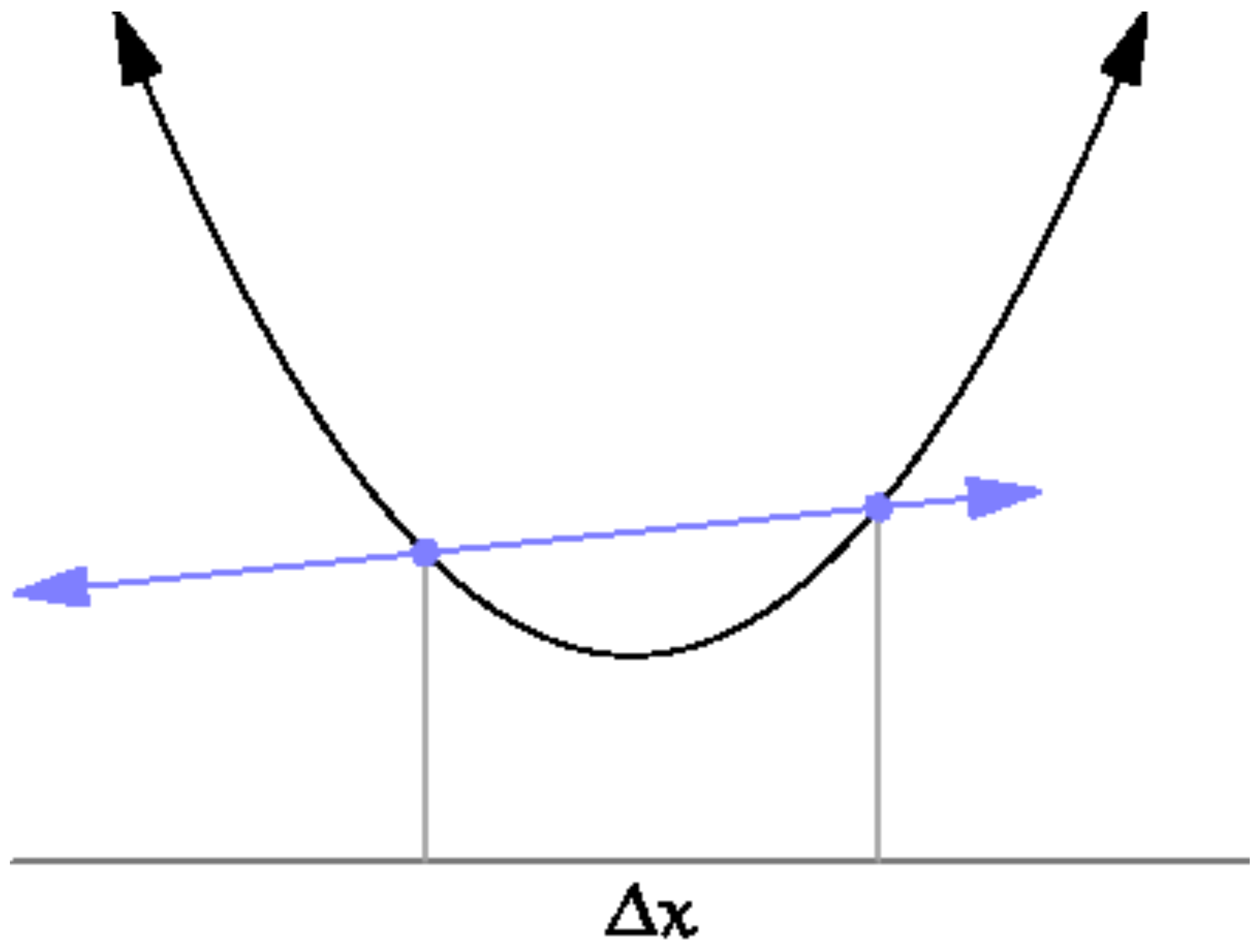
One dimensional derivatives

From Calc101

$$y = f(x) \qquad\qquad \text{Gradient} \;\; m = \frac{\Delta y}{\Delta x}$$

$$m = \frac{f(x+h) - f(x)}{(x+h) - x} = \frac{f(x+h) - f(x)}{h}$$



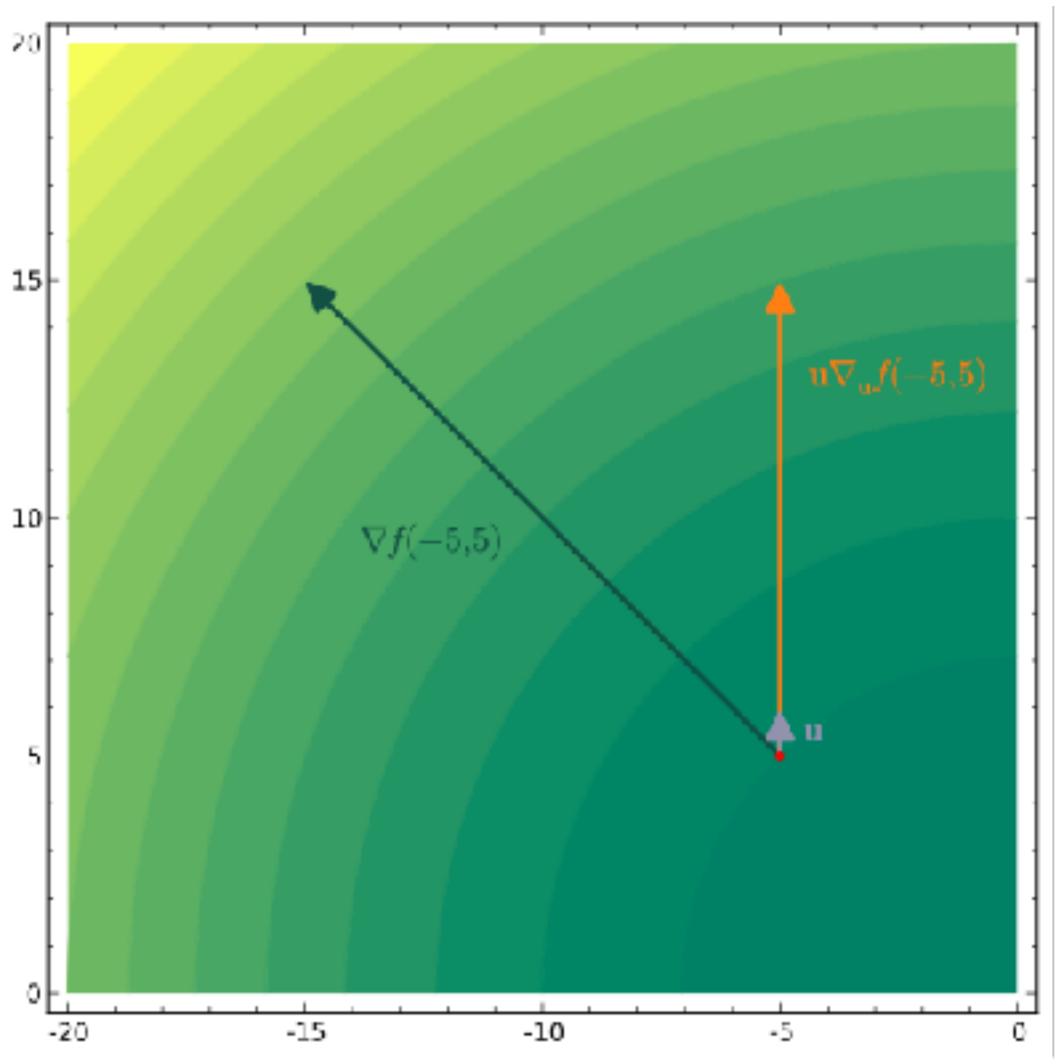


https://en.wikipedia.org/wiki/Derivative

Two dimensional derivatives

For 2D function f(**x**), one can compute a derivative for each direction **v**

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h}.$$



Directional derivatives of the function along the axes are called partial derivatives.
For example the partial derivative with respect to x is:

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

Source: K. Grauman

Partial derivatives with convolutions

For 2D function f(x,y), the partial derivative is:

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

For discrete data, we can approximate using finite differences:

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

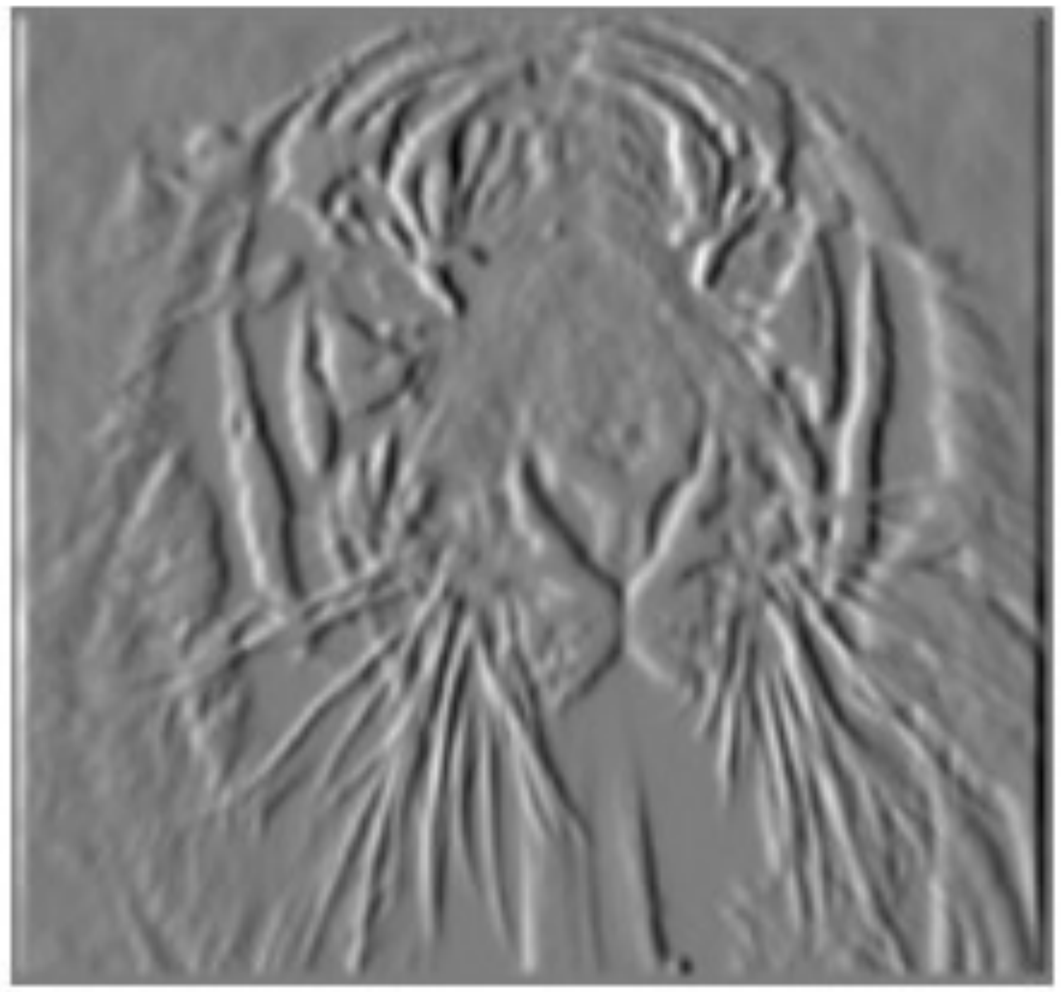
Question: To implement the above as correlation, what would be the associated filter?

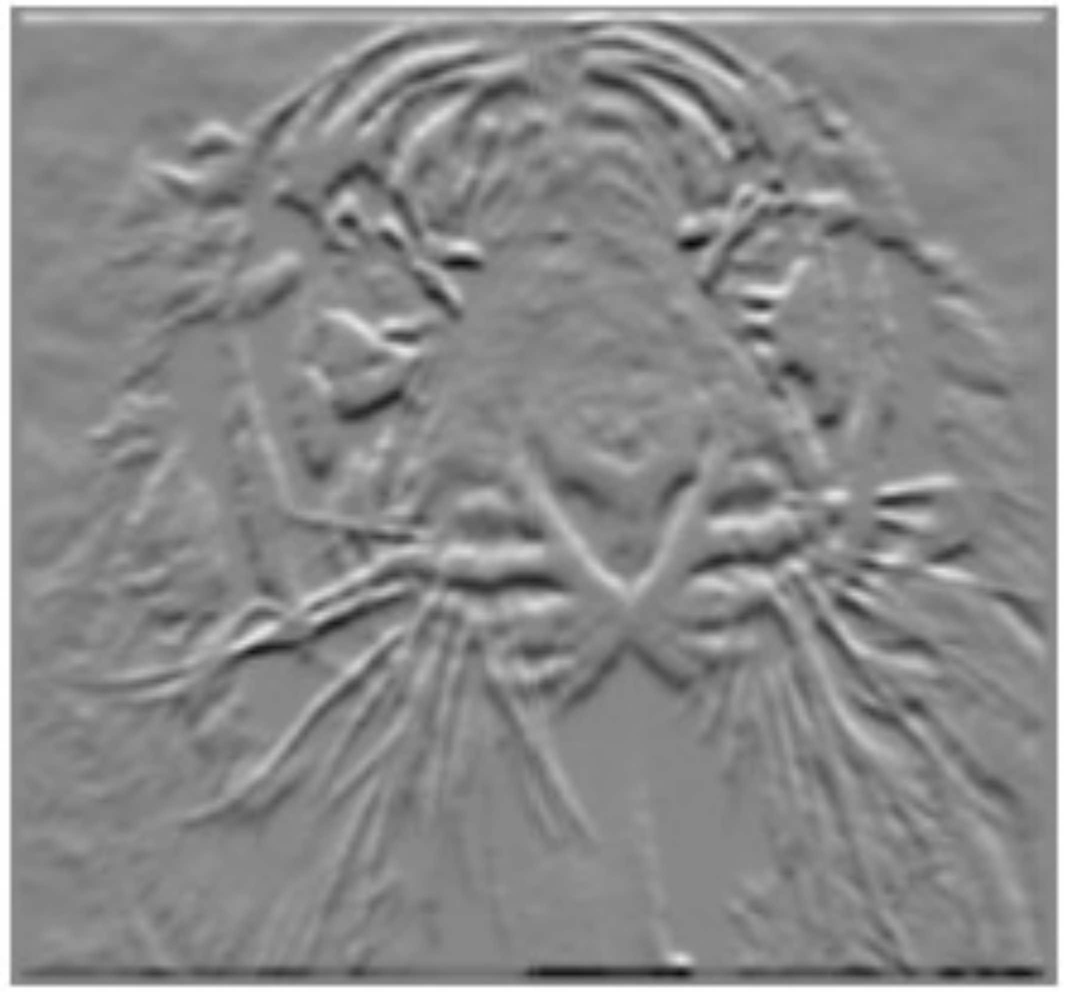
Source: K. Grauman

Partial derivatives of an image

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

-1	1





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

-1
1

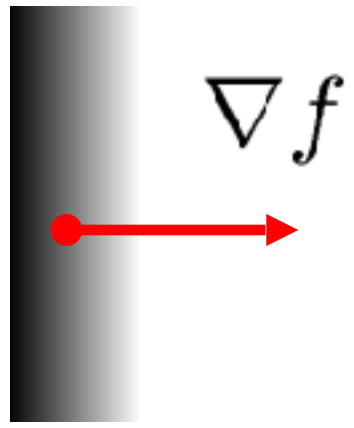
or

1
-1

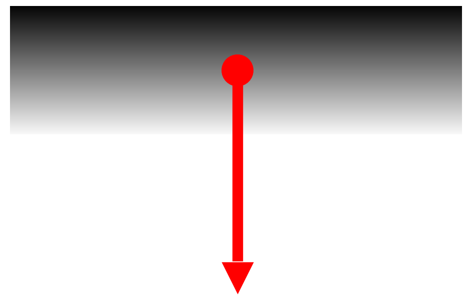
Which one shows changes with respect to x?

Image gradient

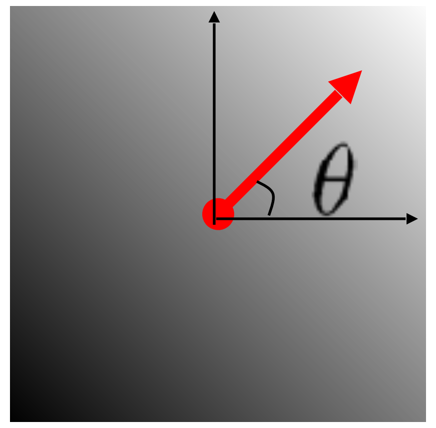
The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



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



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



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

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

How does this direction relate to the direction of the edge? — they are orthogonal

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

The gradient strength is given by the magnitude $$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

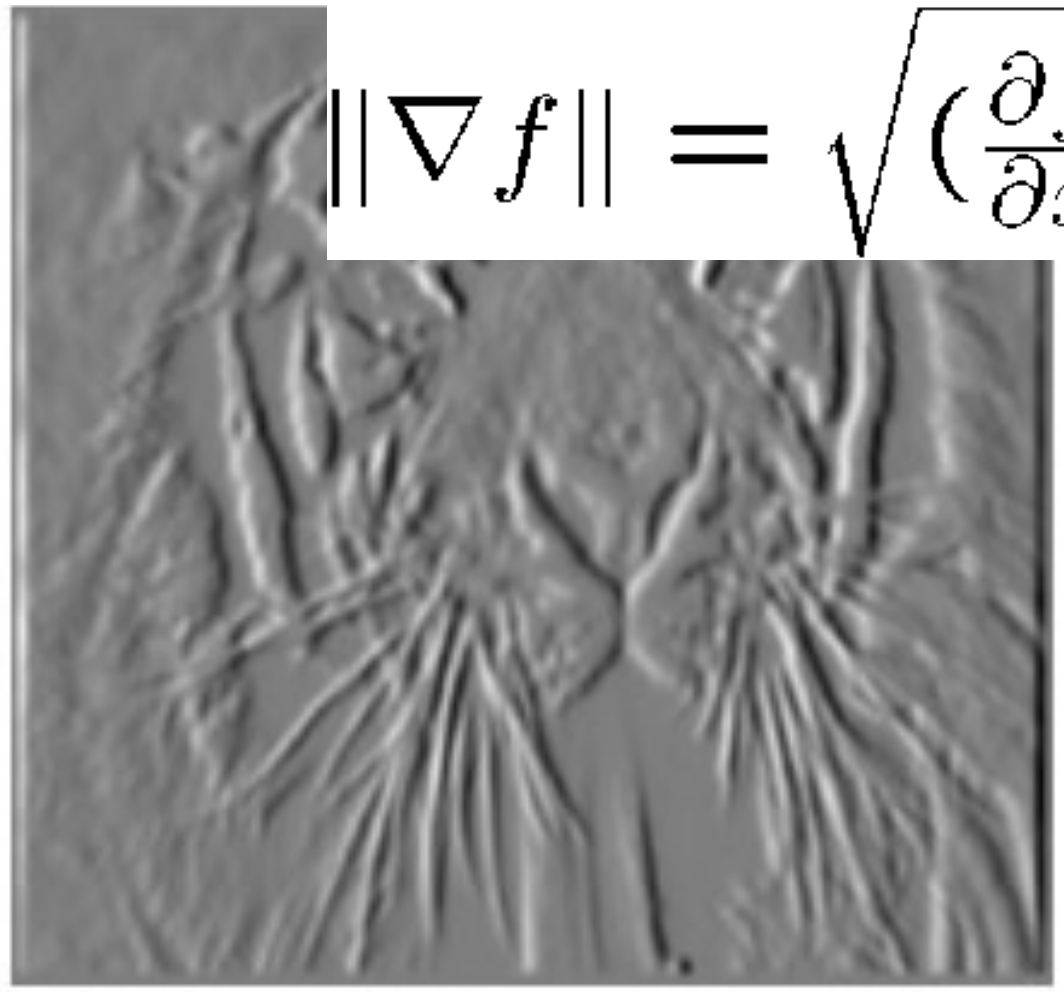
Source: Steve Seitz

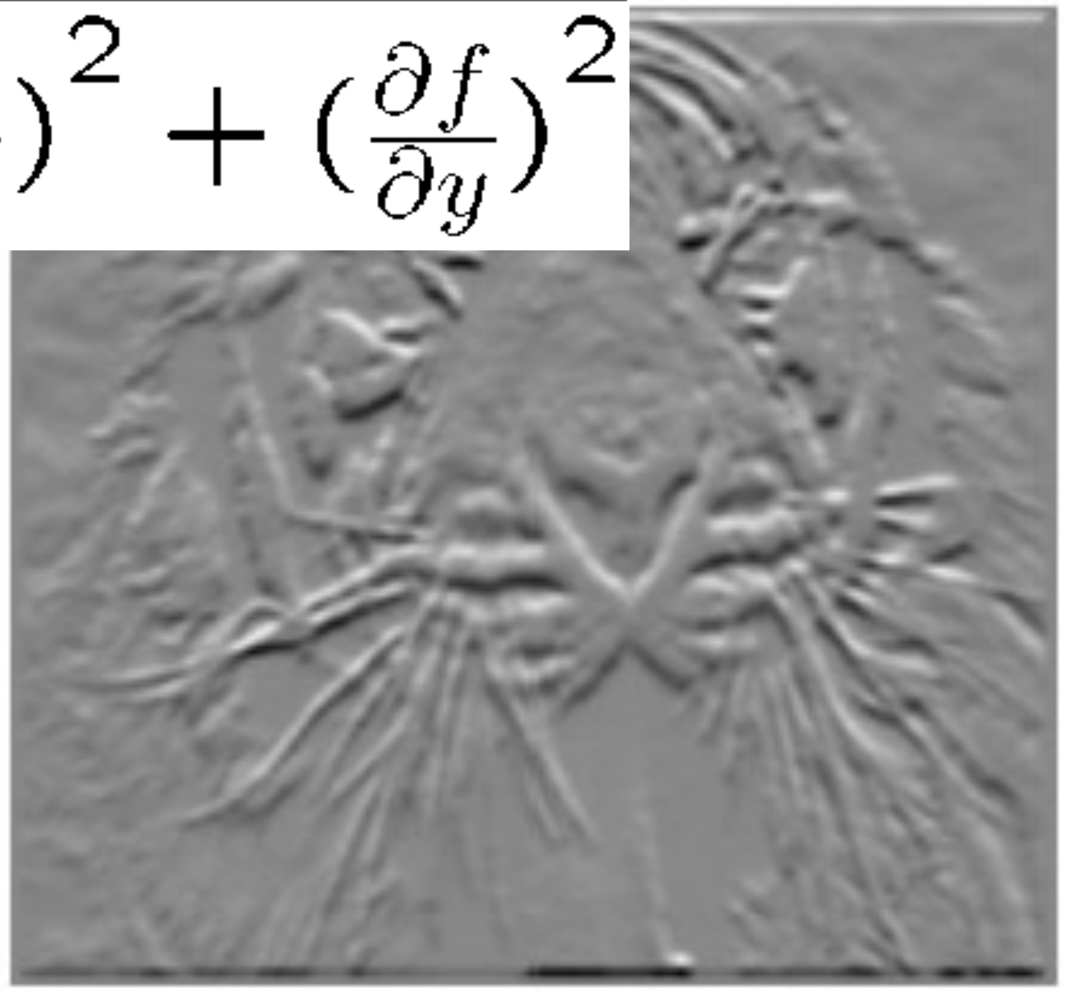
Partial derivatives of an image

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

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

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





-1	1

-1
1

or

1
-1

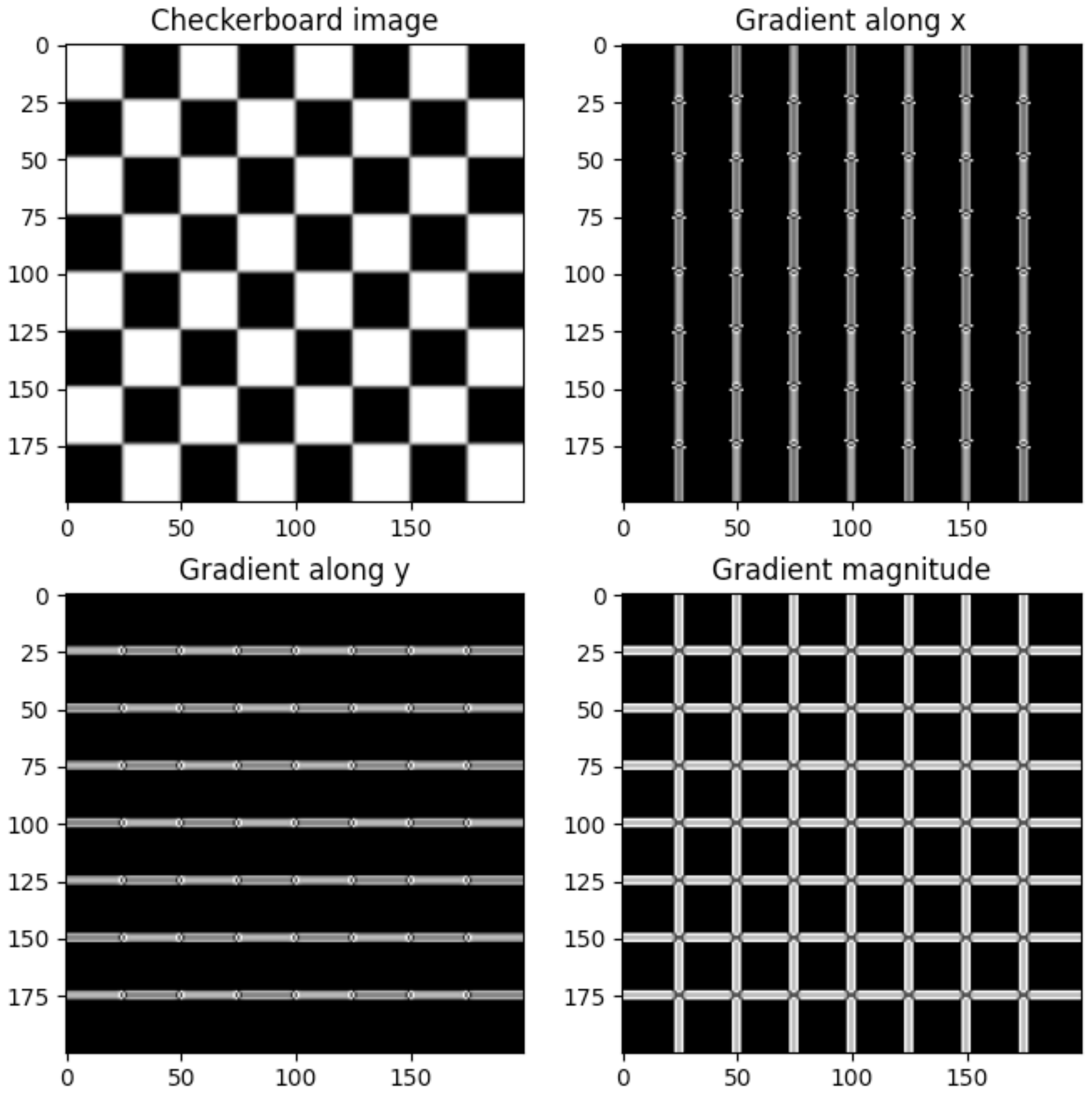
Which one shows changes with respect to x?

Edge detection in Python

~ — IPython: Users/smaji — ipython

File Edit Options Buffers Tools Python Help

```
import numpy as np
import scipy.ndimage as ndi
import matplotlib.pyplot as plt
from skimage import data

im = data.checkerboard()

# Filters along x and y
fx = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
fy = fx.transpose()

# Apply filters and compute magnitude
gx = ndi.correlate(im, fx)
gy = ndi.correlate(im, fy)
mag = np.sqrt(np.maximum(gx**2 + gy**2, 0))

# Optionally convert this to a 0-255 image for display
mag = np.uint8(mag/mag.max()*255)


# Visualize outputs
plt.subplot(2,2,1)
plt.imshow(im, cmap='gray')
plt.title('Checkerboard image')

plt.subplot(2,2,2)
plt.imshow(gx, cmap='gray')
plt.title('Gradient along x')

plt.subplot(2,2,3)
plt.imshow(gy, cmap='gray')
plt.title('Gradient along y')

plt.subplot(2,2,4)
plt.imshow(mag, cmap='gray')
plt.title('Gradient magnitude')
plt.show()
```



Edge detection example

image

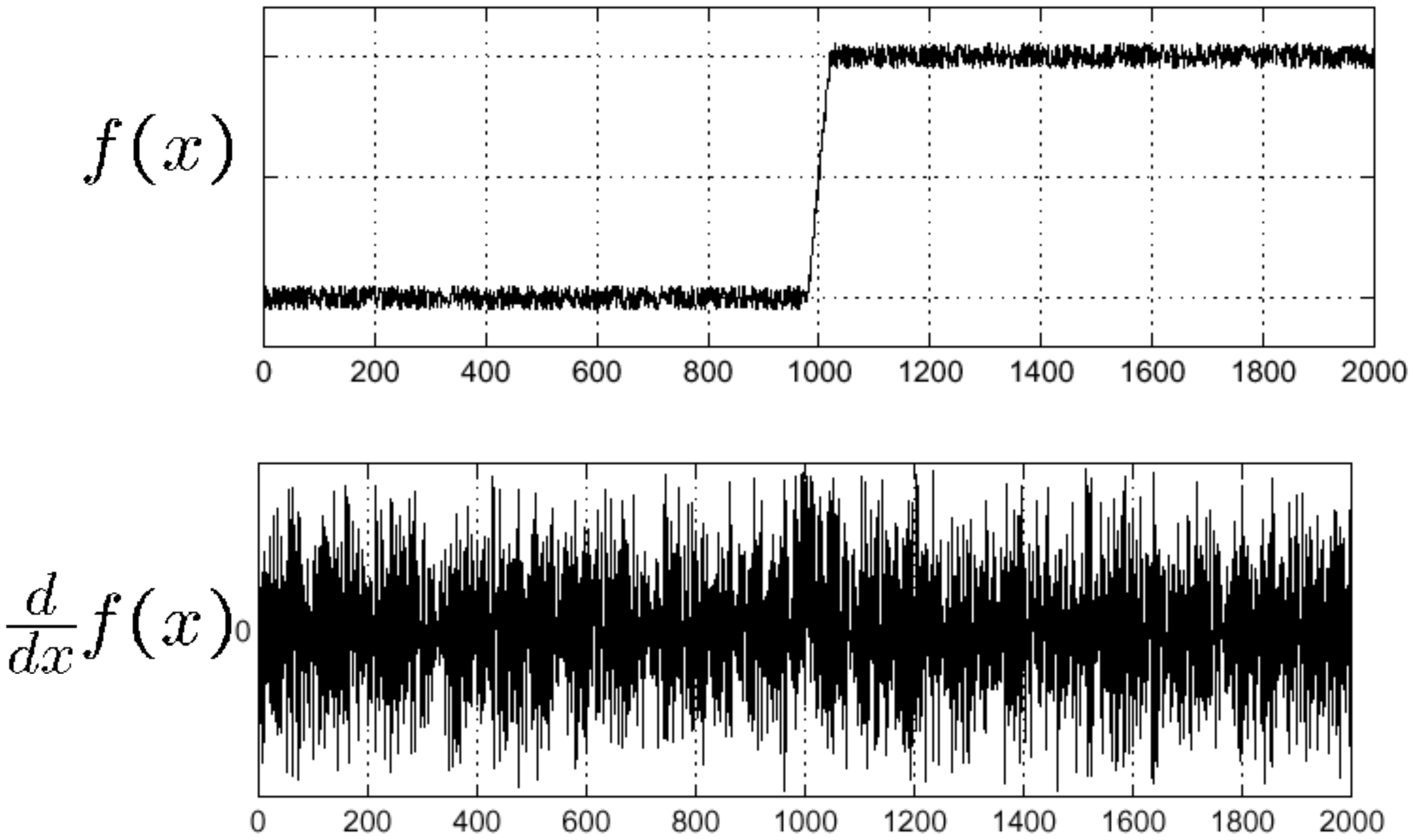
edge magnitude

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A}$$

https://en.wikipedia.org/wiki/Prewitt_operator

Effects of noise

Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

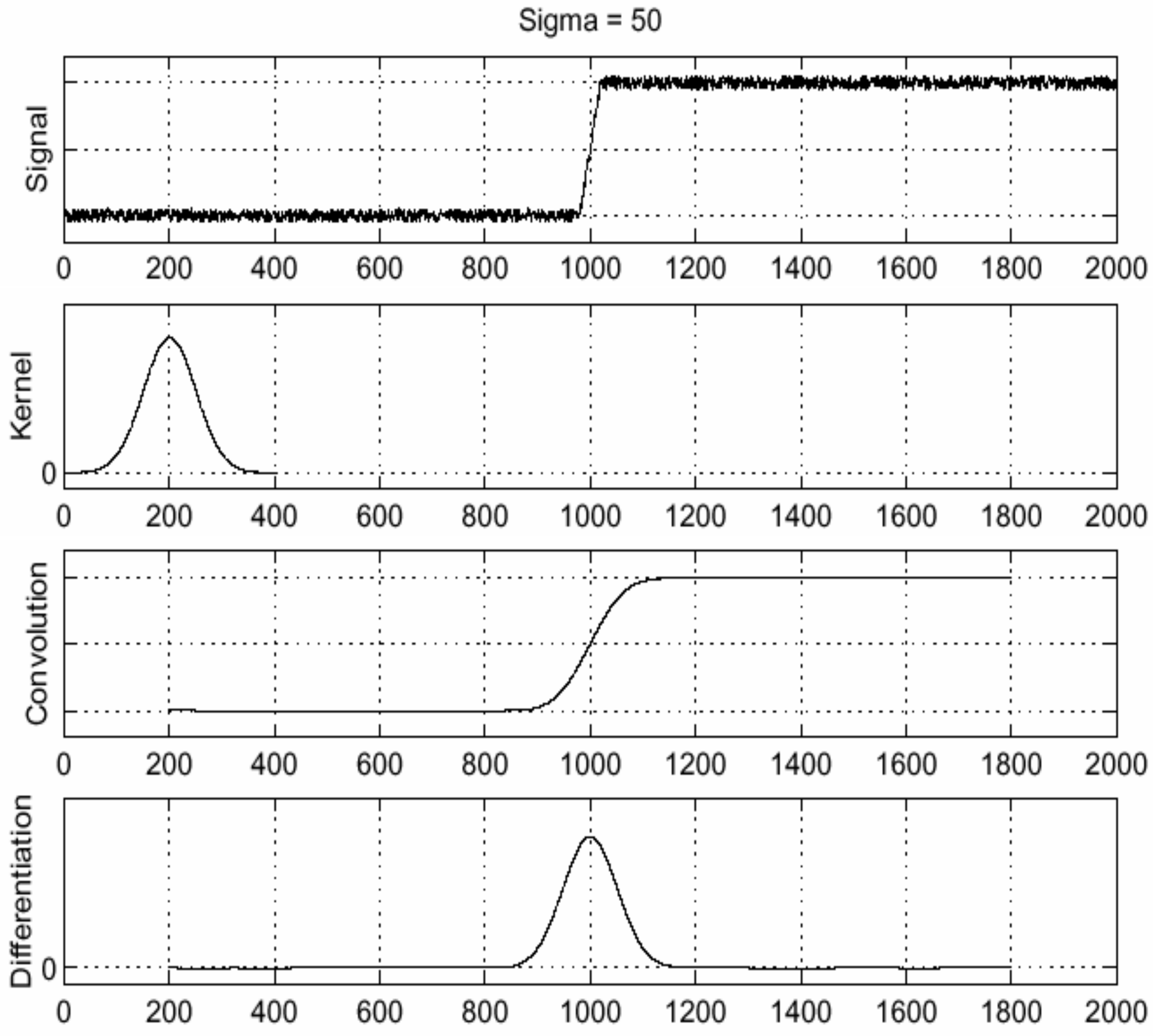
Solution: smooth first

f

g

$f * g$

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



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

Source: S. Seitz

Smooth derivative filters

Differentiation is convolution, and convolution is associative:

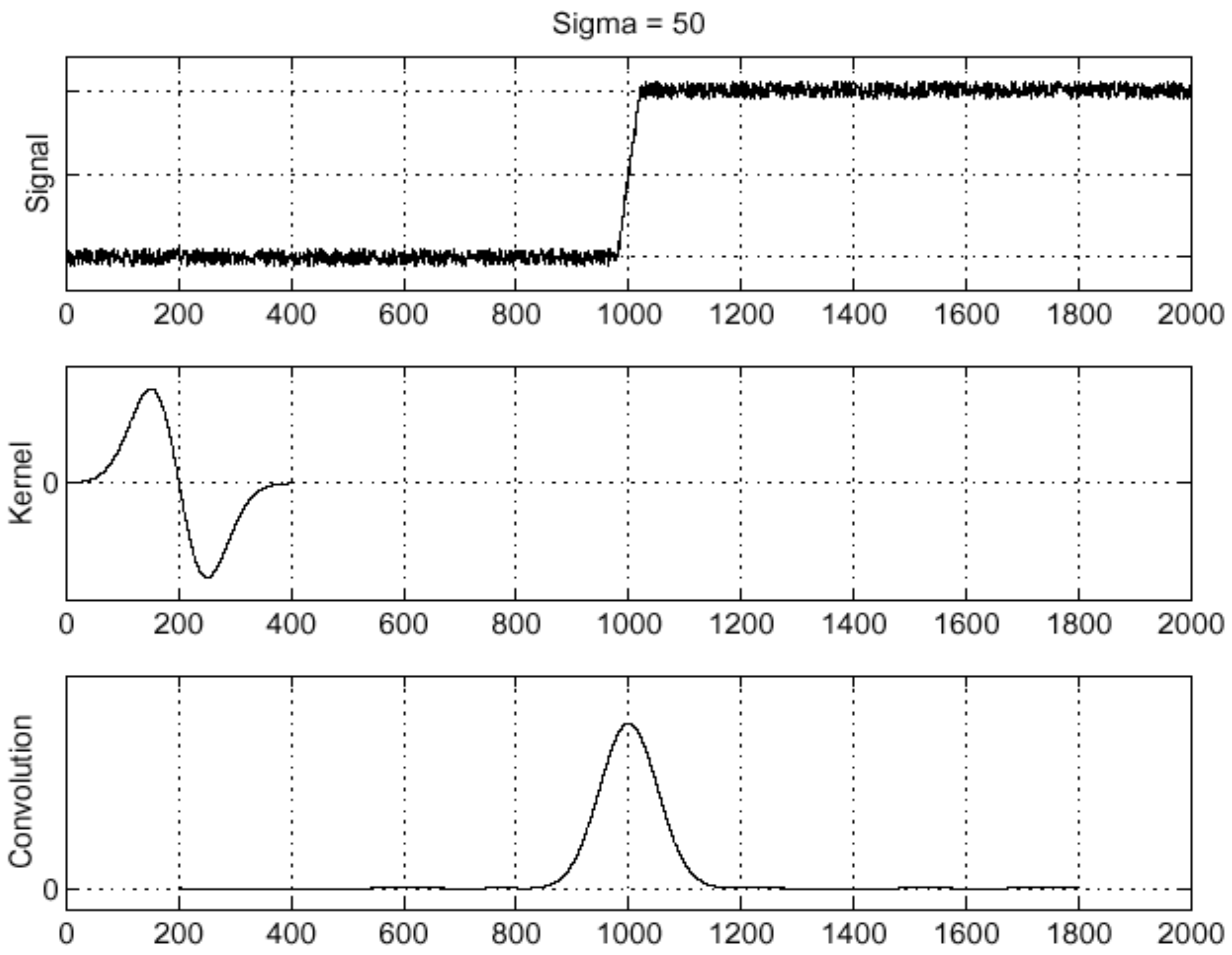
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

This saves us one operation:

f

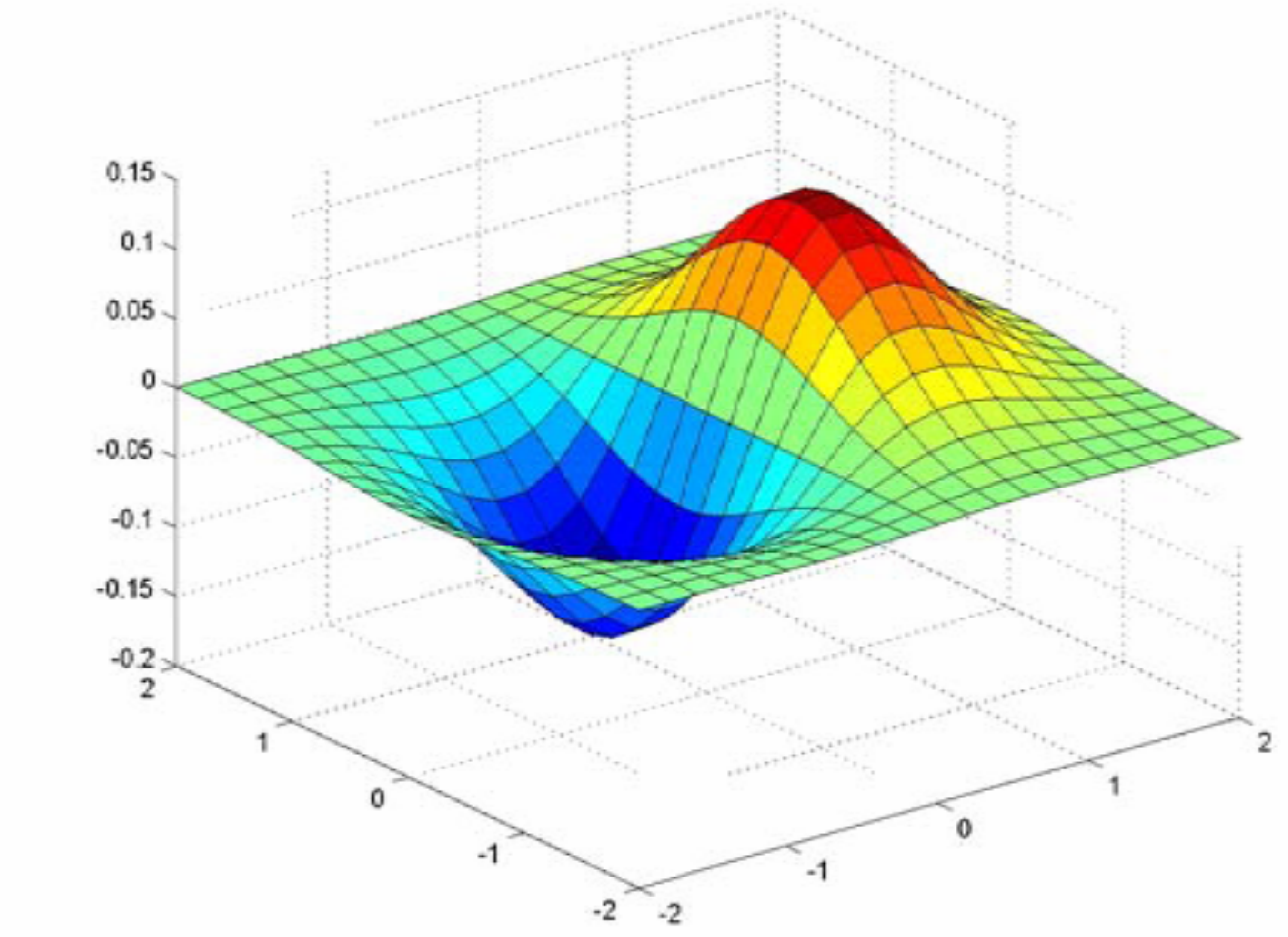
$\frac{d}{dx} g$

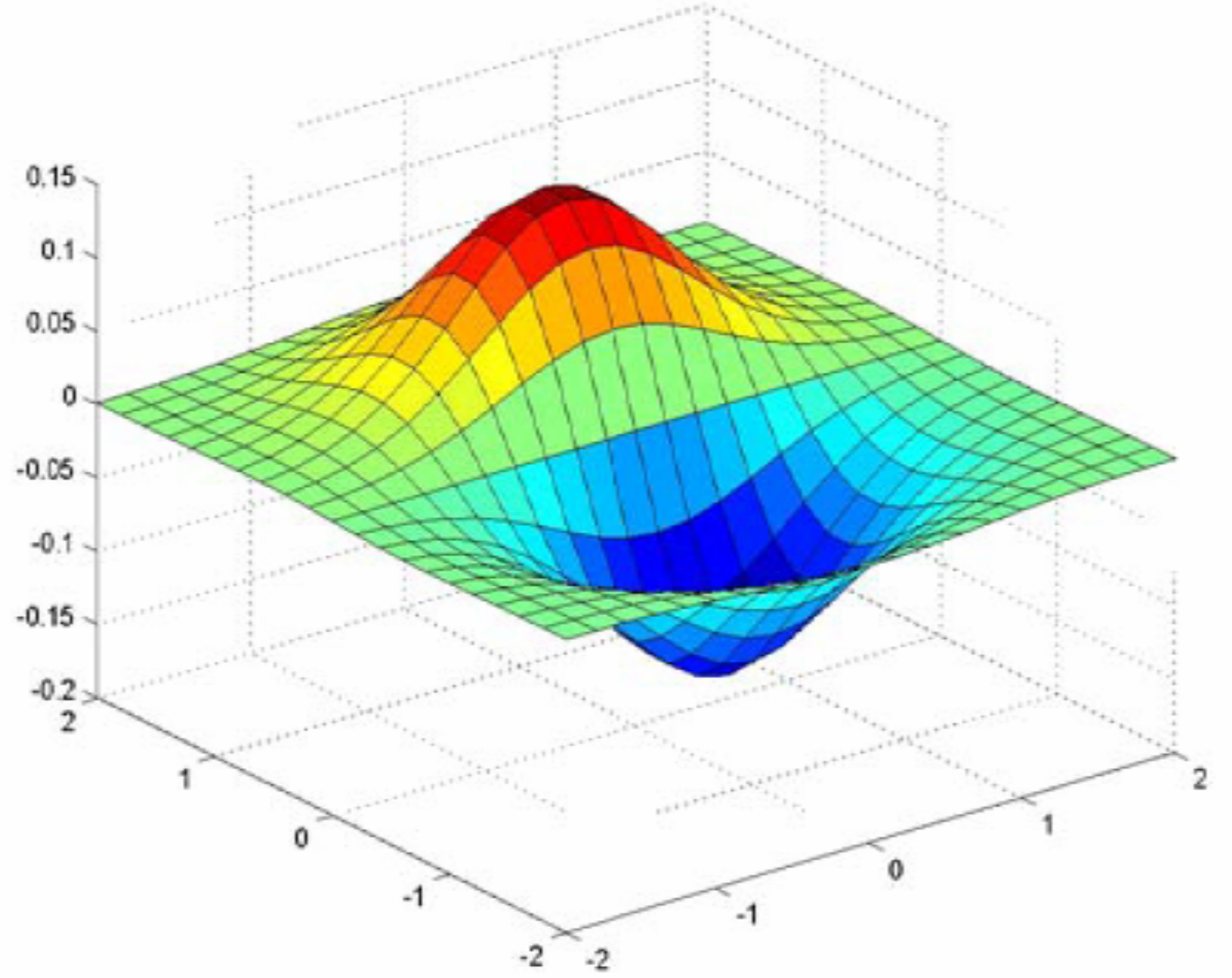
$f * \frac{d}{dx} g$



Source: S. Seitz

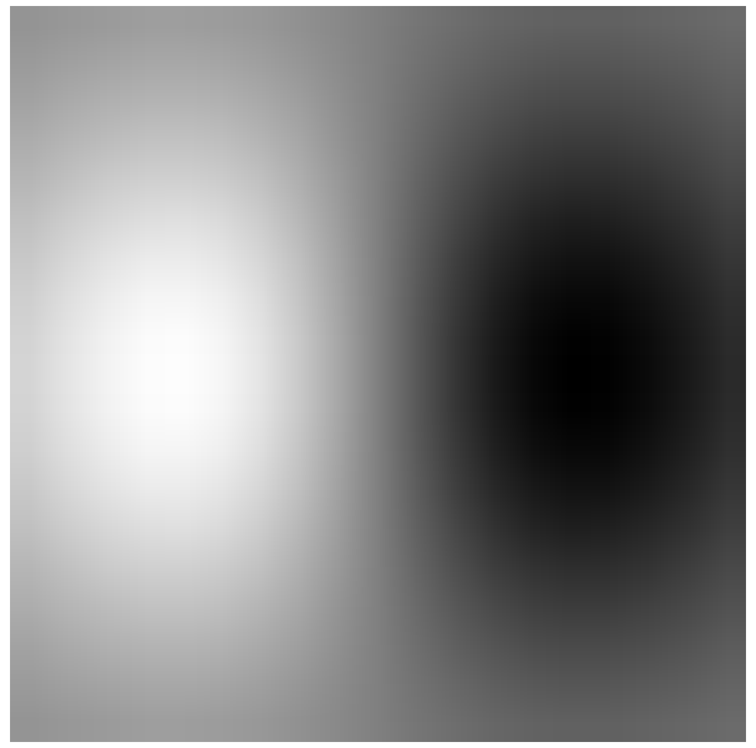
Derivative of Gaussian filters

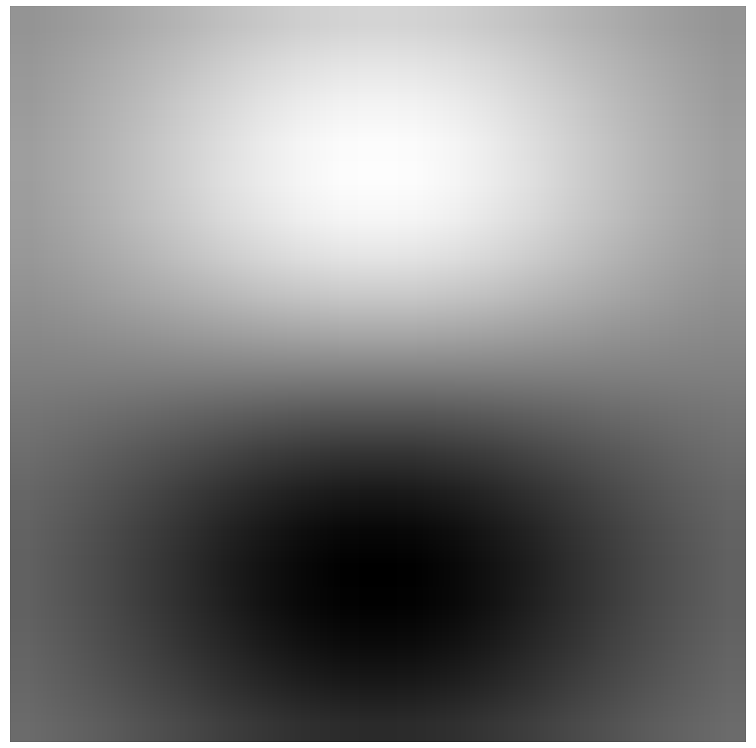




x-direction

y-direction

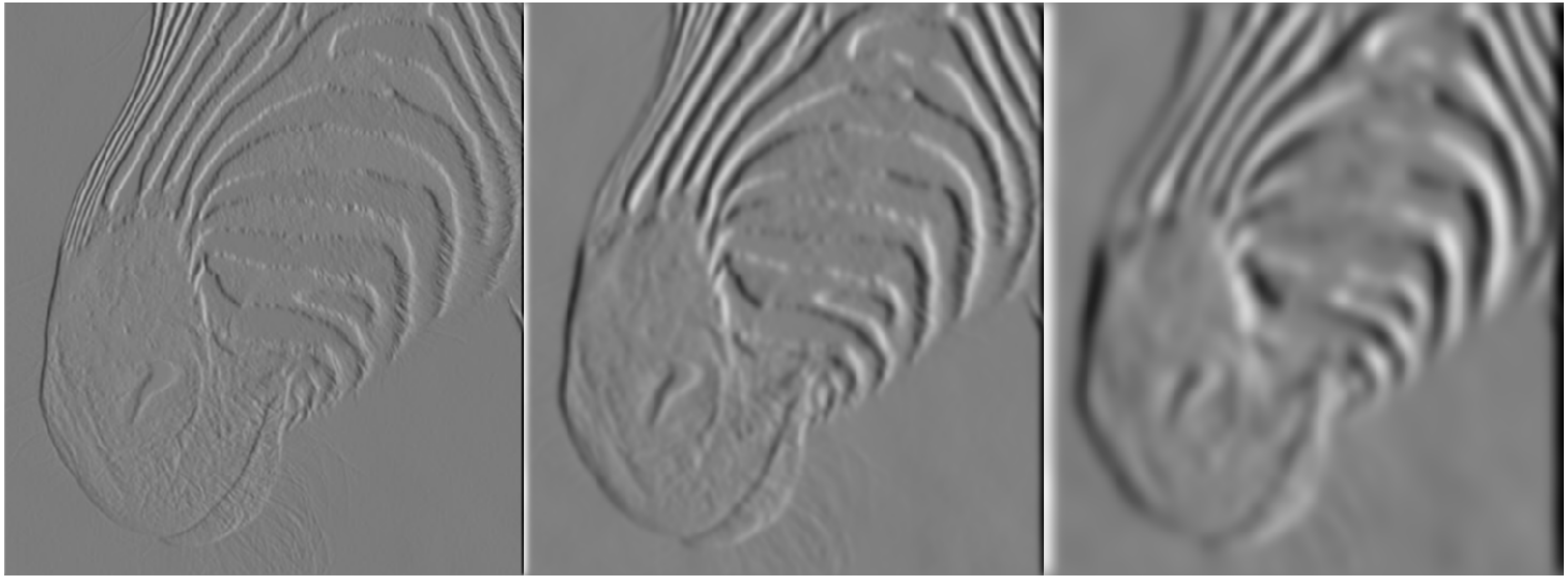




1) Which one finds horizontal edges?

2) Are these filters separable?

Scale of Gaussian derivative filter



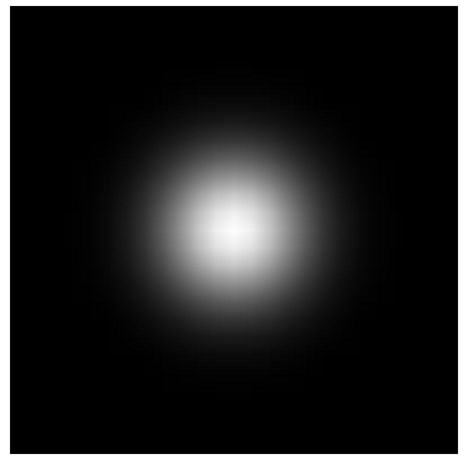
1 pixel 3 pixels 7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different "scales"

Source: D. Forsyth

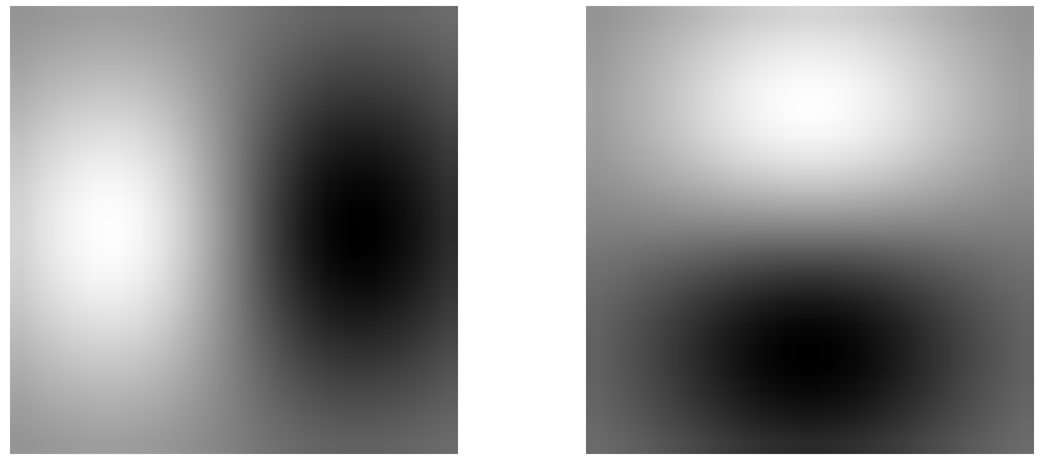
Smoothing and derivative filters

Smoothing filters



- Gaussian: remove "high-frequency" components; "low-pass" filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
    - **One:** constant regions are not affected by the filter

Derivative filters



- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
    - **Zero:** no response in constant regions
- High absolute value at points of high contrast