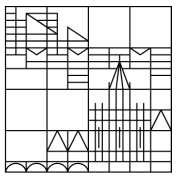
Chapter 1

Images, Noise and Filters



Lecture “Image Analysis and Computer Vision”
Winter semester 2014/15

Bastian Goldlücke

Overview

Overview

Continuous greyscale image

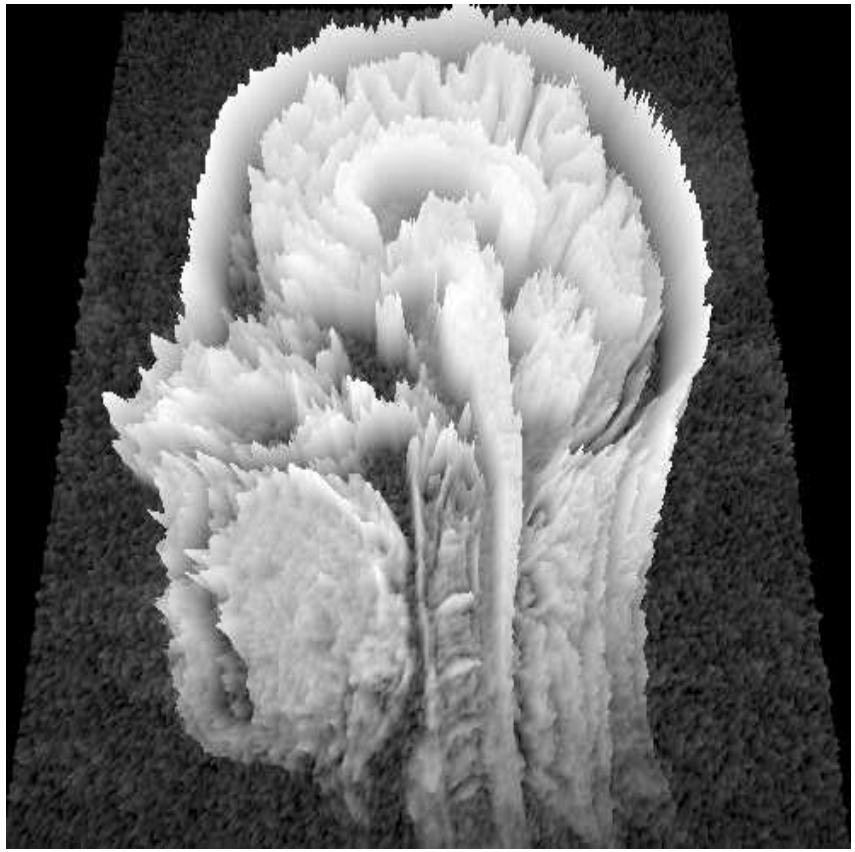
- mapping f from a rectangular domain $\Omega = (0, w) \times (0, h)$ to a co-domain $\mathbb{R}$,

$$f : \mathbb{R}^2 \supset \Omega \to \mathbb{R}$$

- domain Ω is called *image domain* or *image plane*
- co-domain specifies grey value
- Usually low grey values mean dark and high grey values bright.

Continuous greyscale image: visualization

Magnetic resonance (MR) image of a human head



Representation as a function f over a rectangular image domain Ω

Sampling

- discretisation of the **domain** Ω
- Image data are only given on a rectangular point grid within the image domain Ω
- creates a *digital image*

$$(f_{i,j})_{i=1,\ldots,N,\ j=1,\ldots,M}$$

- The grid point or cell (i, j) is called *pixel* (picture element).
- 2D images often have equal pixel distances in both directions.
- Image processing people often normalise these grid sizes to 1. This can be dangerous and is not recommended!
- If the sampling is too coarse, the image quality degrades severely.

Visualization of different sampling rates

256 $\times$ 256 pixels

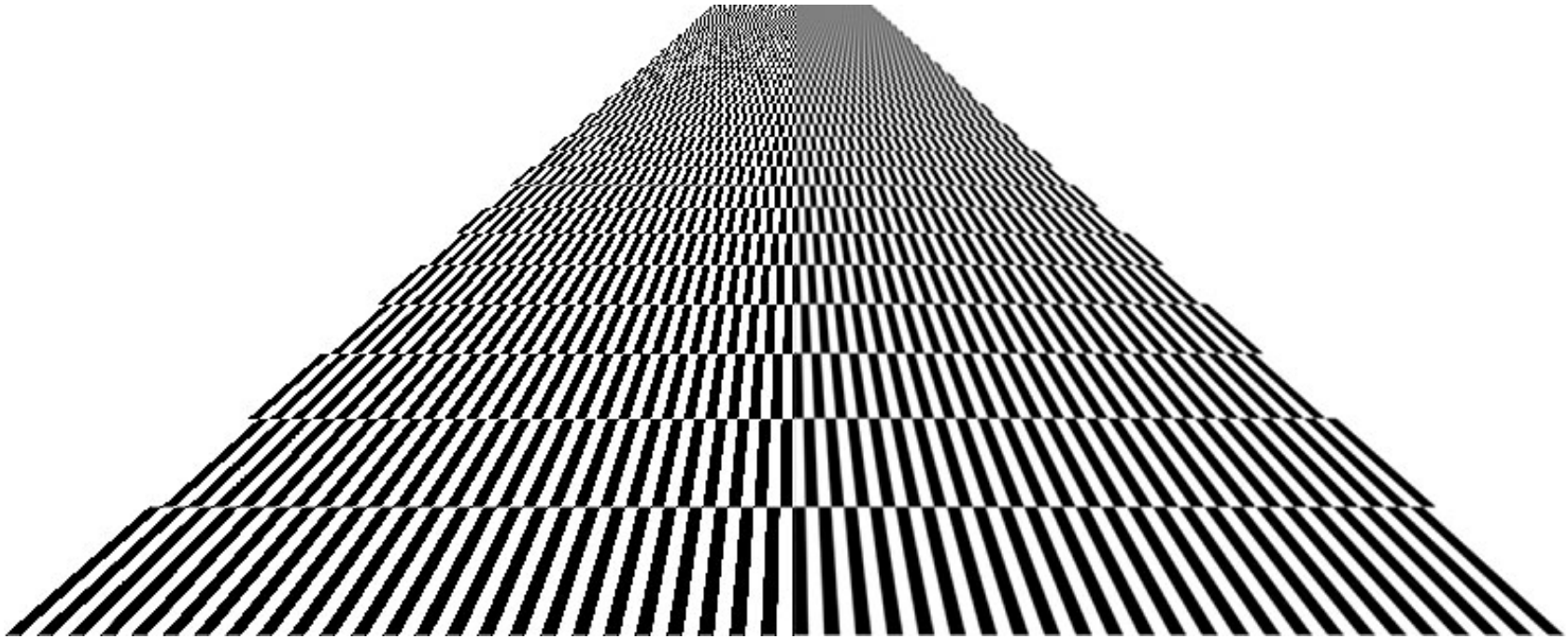
128 $\times$ 128 pixels

64 $\times$ 64 pixels

32 $\times$ 32 pixels

Slide: A. Bruhn, Images: J. Weickert (2000).

A sampling rate which is too low creates *aliasing* ...



- Aliasing means signals become indistinguishable when sampled
- Reminder (maybe) from signal processing: sampling rate must be above the *Nyquist rate*, which is two times the bandwidth of the signal, to have an aliasing-free result
- We will get back to that at a later point, when we discuss the Fourier transform of an image

Image: *zone-10.com*

Quantisation

- discretisation of the **co-domain**
- saves disk space
- example: if a grey value is coded by a single byte, the discrete co-domain is given by

$$\{0, 1, \ldots, 255\}$$

- *Binary images* have the co-domain $\{0, 1\}$.
- Humans can distinguish only 40 greyscales.
- They are even very good in recognising the content of binary images.

Visualization of quantisation effects

256 greyscales

32 greyscales

8 greyscales

2 greyscales

m-Dimensional Images

- generalization: domain in $\mathbb{R}^m$
- $m = 1$: signals
- $m = 2$: (two-dimensional) images
- $m = 3$: three-dimensional images
  - important in medical imaging, e.g.
    computerised tomography (CT, Computertomographie),
    magnetic resonance imaging (MRI, Kernspintomographie).
  - image points/cells in 3-D are called *voxels* (volume elements).
  - voxel dimensions usually differ in different directions!

Rendering of an ultrasound image of a fetus in its 10th week

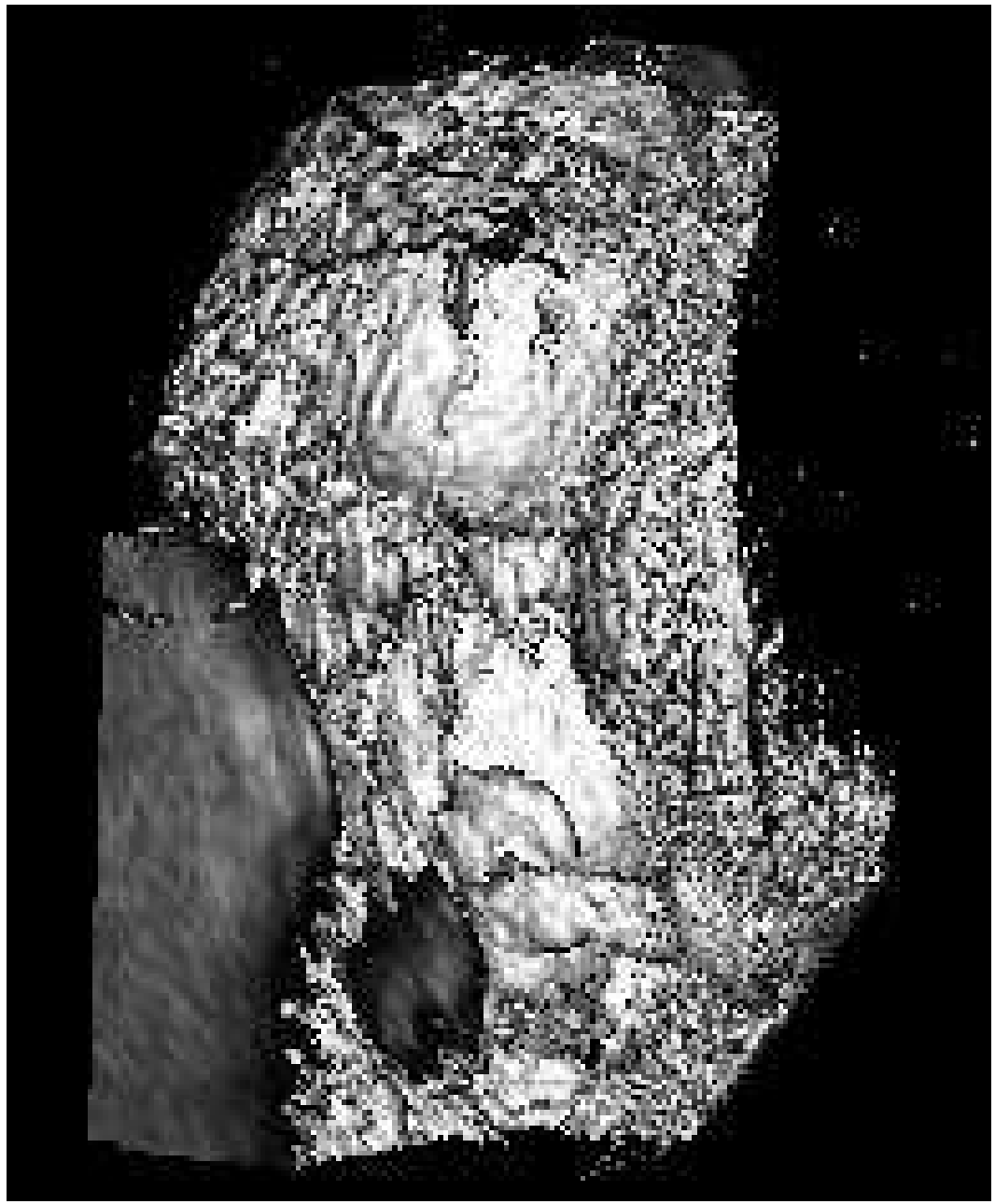


Image: J. Weickert, K. Zuiderveld, B.M. ter Haar Romeny, W. Niessen (1997).

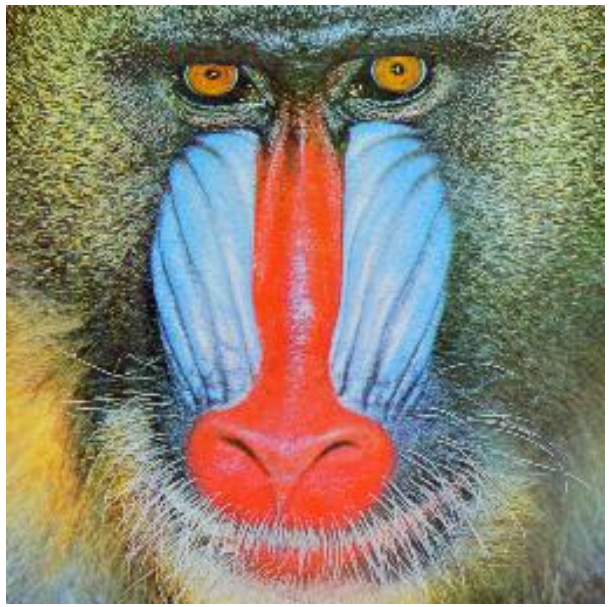
Vector-valued images

- generalization: co-domain in $\mathbb{R}^n$, containing n channels

- Example 1: Colour images
  - Three channels: R (red), G (green), B (blue).
  - Humans can distinguish 2.000.000 colours!
  - We will learn much more about color perception in a later lecture

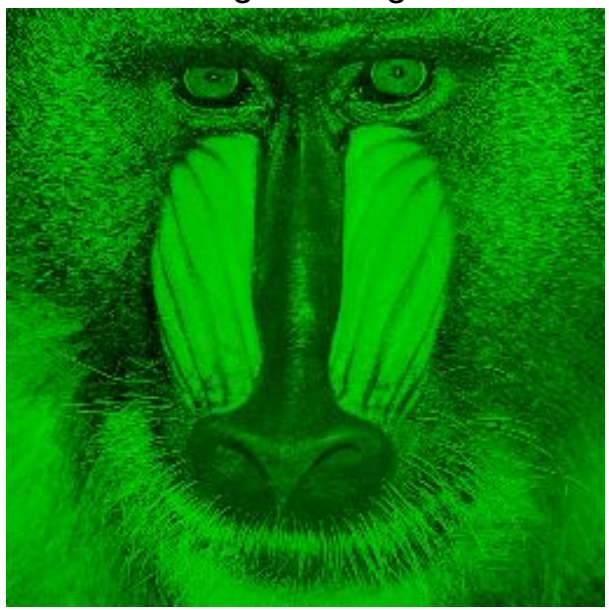
- Example 2: Multispectral image (e.g. satellite images)
  - numerous channels that represent different frequency bands

Color image as example for a vector-valued image

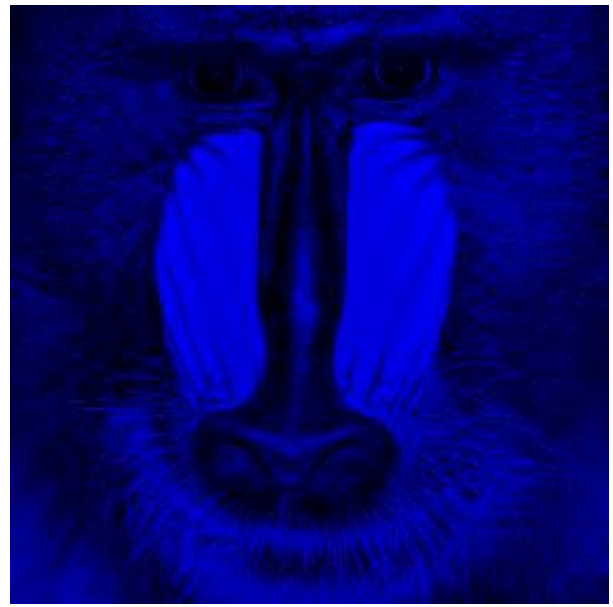


Original image

Red channel



Green channel

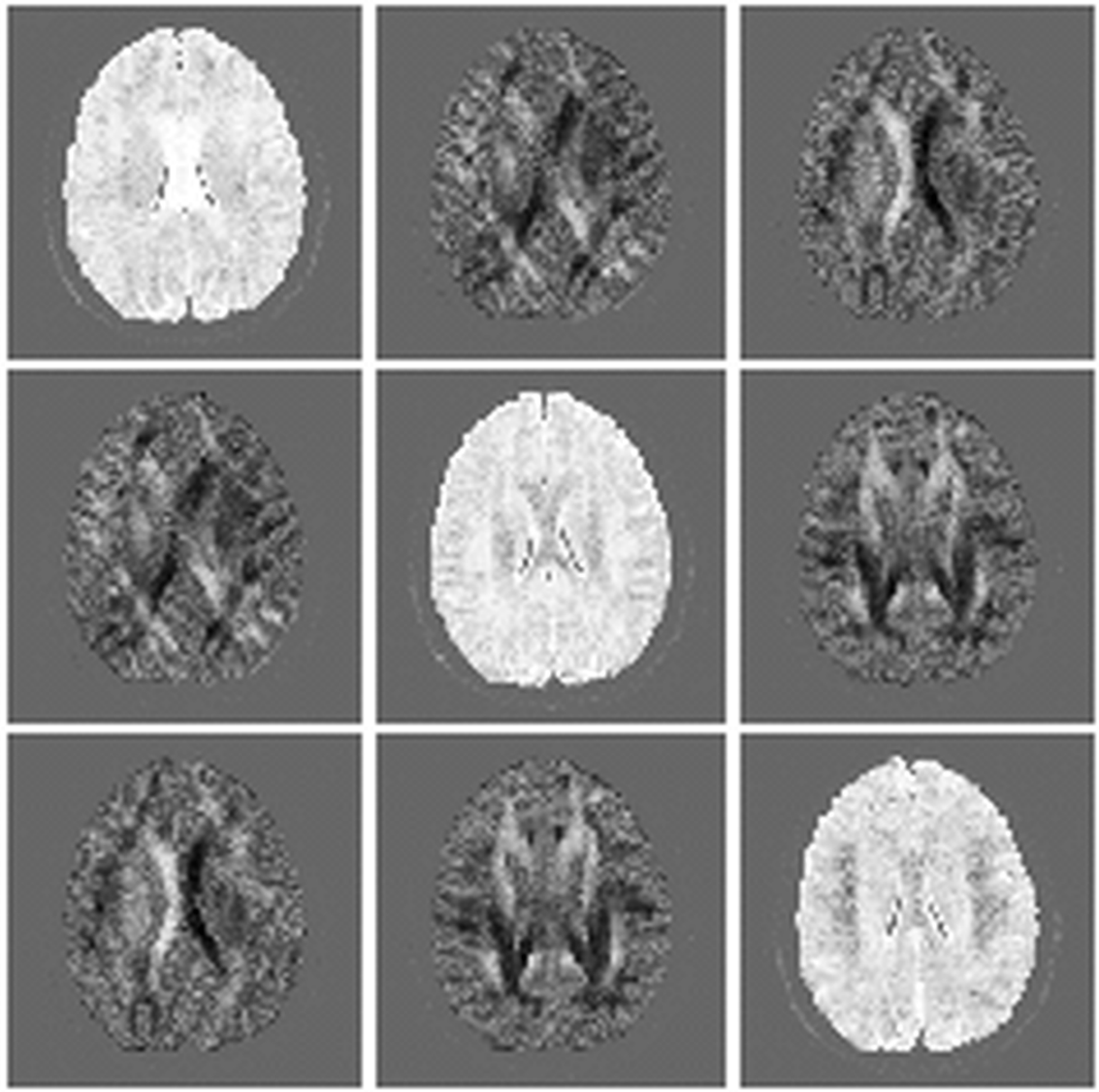


Blue channel

Matrix-valued images (sometimes: “tensor”)

- co-domain in $\mathbb{R}^{n \times n}$
- Example: diffusion tensor MRI (DT-MRI, “Diffusionstensor-Kernspintomographie”): measures in each voxel of a 3D image domain a symmetric positive definite 3×3 matrix
- may create additional constraints, e.g.: A reasonable filter should not destroy relevant properties such as positive definiteness of the matrix field.

Nine coefficients of a DT-MRI data set of a human brain



Since the diffusion matrix is a symmetric 3×3 matrix, only 6 out of 9 images differ.

Slide: A. Bruhn, Image: D. Weinstein, G. Kindlmann, E. Lundberg (1999).

Image sequences

- One can consider image sequences for any of the above mentioned image types.
- This increases the dimensionality of the domain from m to $m + 1$.
- Example 1: A color video sequence can be regarded as a 3D image with 3D co-domain.
- Example 2: 3D ultrasound can create a sequence of 3D scalar-valued images, which can be regarded as 4D image.

Four subsequent frames of an image sequence

Slide: A. Bruhn, Images from the famous Hamburg taxi scene, 256 $\times$ 190 pixels. Source: http://i21www.ira.uka.de/image_sequences/

Relevant images in this class

- We mainly focus on 2D scalar or RGB images ($m = 2$, $n = 1$ or $n = 3$).
- The gradient of a greyscale image yields another important vector-valued image ($m = 2$, $n = 2$).
- Tensors are lurking in the shadows of the second chapter.
- Many methods can also be generalised to images in higher dimensions (see the Mathematical Image Analysis lecture).

Notation and conventions: digital images

- A digital grayscale image will be usually denoted with

$$f = (f_{i,j}),$$

  where we assume $f_{i,j} \in \mathbb{R}$.

- Pixel coordinates (i, j) can be anywhere in $\mathbb{Z}^2$, but to make everything well-defined, we'll implicitly assume that an image always has "compact support", i.e. $f_{i,j} = 0$ outside a bounded region $\Omega \subset \mathbb{Z}^2$ (the usually rectangular image domain).

- Sometimes I write $f(i, j)$ instead of $f_{i,j}$ if the formulas become more easy to read that way.

- If the image has multiple channels (i.e. RGB), I make that clear by e.g. writing

$$\mathbf{f} : \mathbb{Z}^2 \to \mathbb{R}^3$$

  to indicate the co-domain. I also use a bold-faced letter $\mathbf{f}$.

Overview

Image noise

Original image

Noisy image

- How can one characterize noise, what **types of noise** are there?
- How can one get rid of noise?

Additive noise

- Most important type of noise
- Grey values and noise are assumed to be independent:

$$f_{i,j} = g_{i,j} + n_{i,j}$$

where

$g = (g_{i,j})$ is the original image,
$n = (n_{i,j})$ the noise, and
$f = (f_{i,j})$ the noisy image.

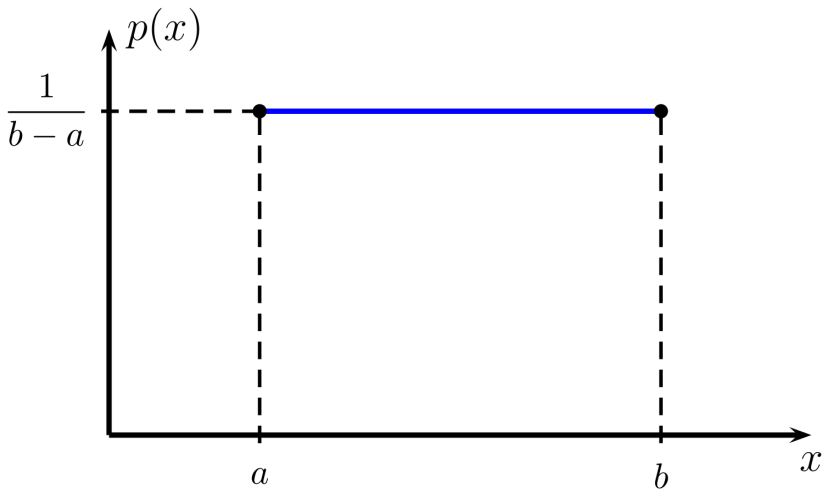
- The noise n may have different distributions, e.g.
  - uniform distribution (very simple)
  - Gaussian distribution (very common)

Uniform distribution

- often not the most realistic noise model, but easy to simulate
- has a constant density function within some interval $[a, b]$:

$$p(x) = \begin{cases} b - a & \text{for } x \in [a, b], \\ 0 & \text{else.} \end{cases}$$

- appears e.g. in connection with quantisation



Density function for uniform noise. Author: M. Mainberger (2008).

Uniform additive noise

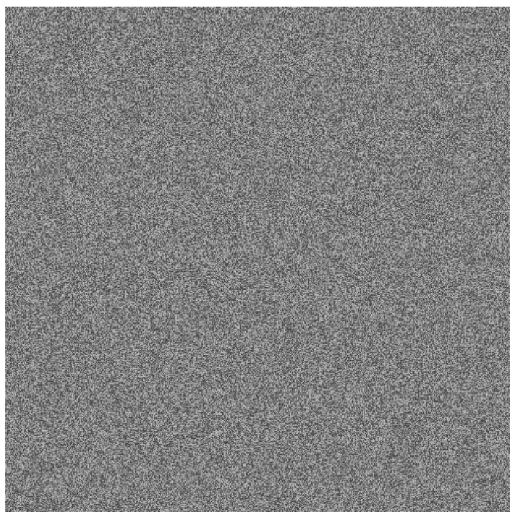
Original image

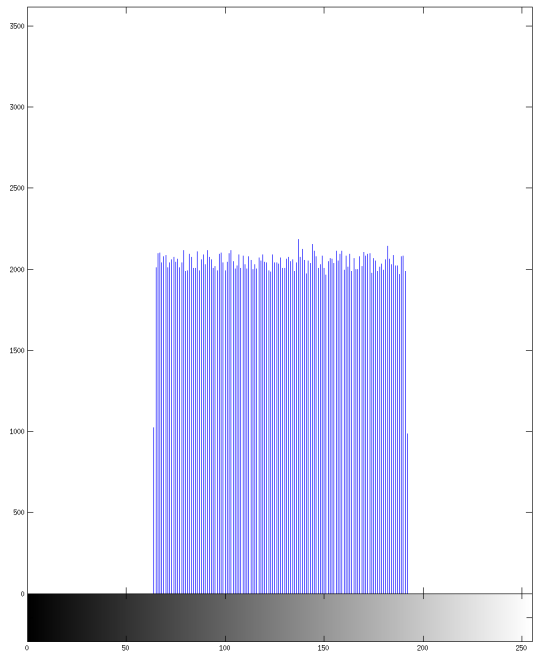
Noisy, $(b - a) = 0.5 f_{max}$

Uniform additive noise

```
g = ones([512,512])*128;
a = -64; b = 64;
n = a + (b-a)*rand(size(image));
f = g + n;

figure,
subplot(1,2,1)
imshow(f / 255.0),
subplot(1,2,2)
imhist(uint8(f))
```





Noise function and histogram

Gaussian distribution

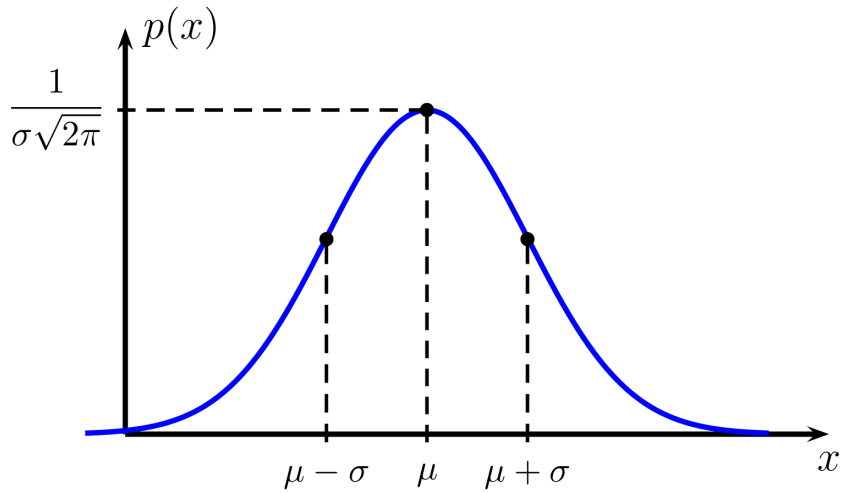
- most important noise model
- good approximation in many practical situations, e.g.
  - thermal sensor noise in CCD cameras
  - circuit noise caused by signal amplifications
- has density function

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

  where μ is the mean and $\sigma > 0$ the standard deviation.

- Note: σ^2 is also called the "variance" of the distribution.

Gaussian distribution



Density function for Gaussian noise. Author: M. Mainberger (2008).

- For a Gaussian-distributed random variable X the following probabilities hold:

$$P(\mu - \sigma \leq X \leq \mu + \sigma) \approx 68\%$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 95.5\%$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 99.7\%$$

- Hence, Gaussian noise lives almost completely in the interval $[\mu - 3\sigma, \mu + 3\sigma]$.
- This is useful to know for discretization, since the distribution technically has infinite support.

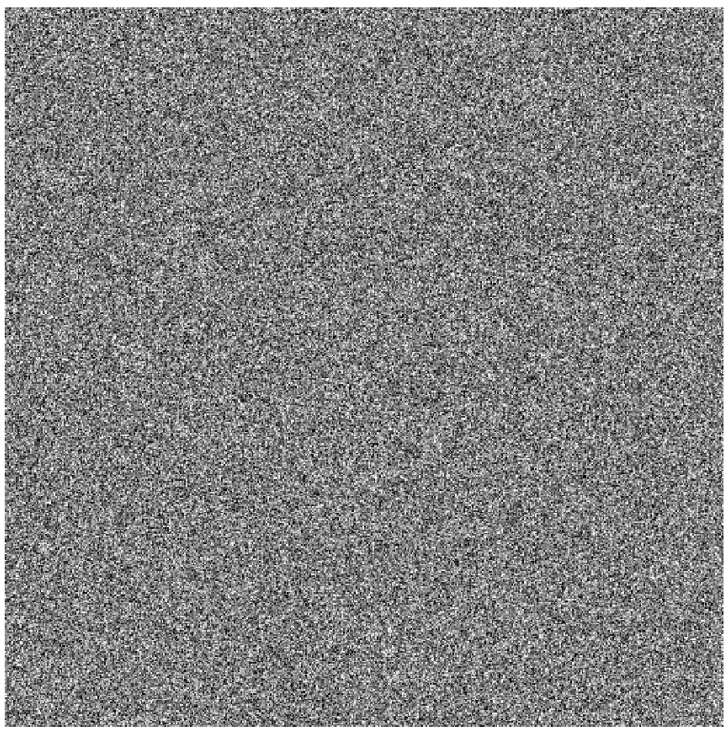
Gaussian additive noise

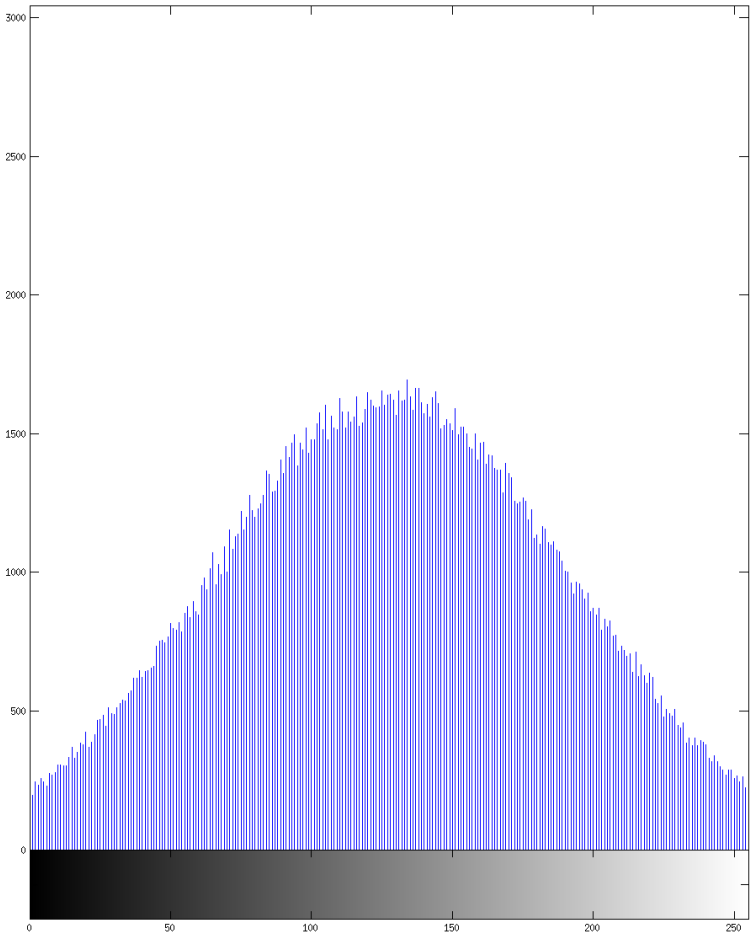
Original image

Additive Gaussian noise, $\sigma = 0.25 f_{max}$

Gaussian additive noise

```
g = ones([512,512])*128;
sigma=64;
n = sigma*randn(size(image));
f = g + n;
```





Noise function and histogram

Multiplicative noise

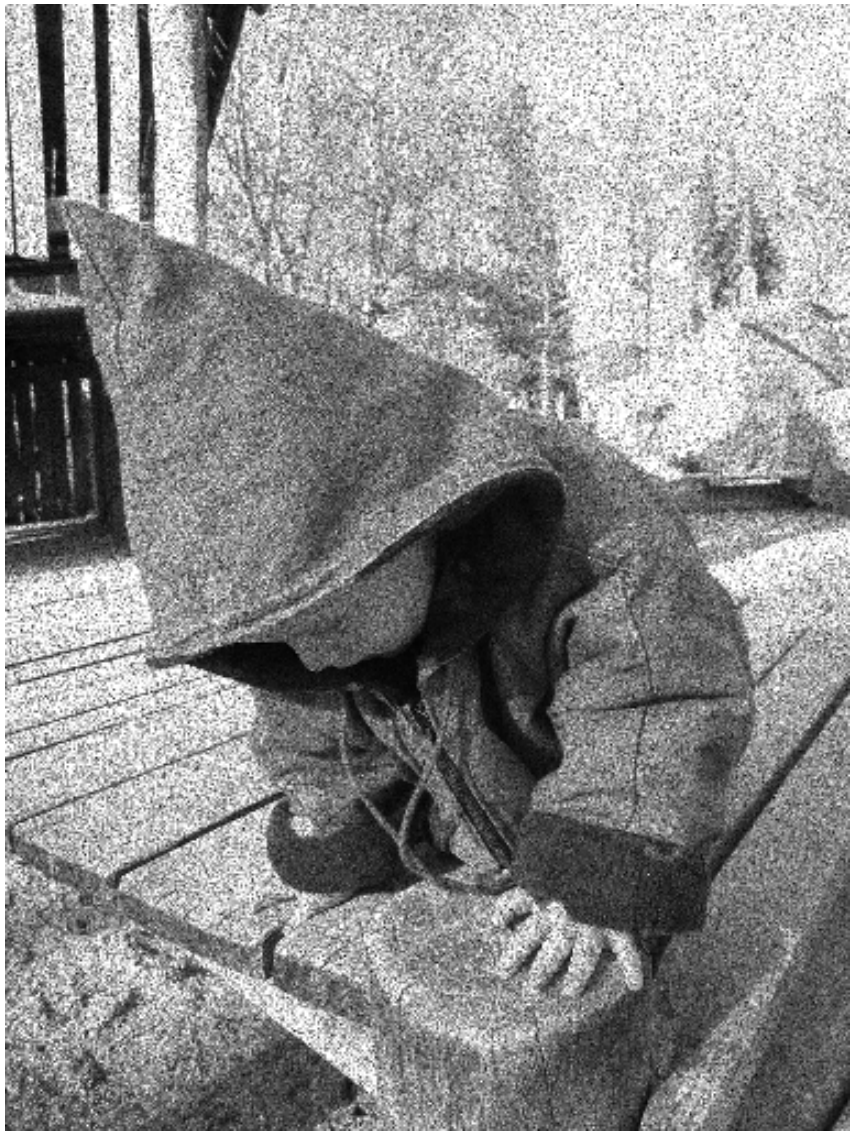
- Multiplicative noise is signal-dependent (unlike additive noise).
- Frequently, it is proportional to the grey value:

$$f_{i,j} = g_{i,j} + n_{i,j} g_{i,j} = (1 + n_{i,j}) g_{i,j}.$$

- The noise n can again follow different distributions.
- Multiplicative noise is prevalent in radar, ultrasound and tomographic images.

Multiplicative Gaussian noise

Original image



Multiplicative Gaussian noise, $\sigma = 0.25 f_{max}$

Impulse Noise

- degrades the image at **some (!)** pixels where erroneous grey values are created, in contrast to additive or multiplicative noise that affects all pixels
- Example: pixel defects in a CCD chip of a digital camera
- **Unipolar impulse noise** gives degradations having only one grey value, whereas **bipolar noise** attains two grey values.
- Bipolar noise with highest and lowest grey values is called **salt-and-pepper noise**.

Salt-and-pepper noise

Original image

Salt-and-pepper noise, 20 percent of pixels affected

We now have many ways to add interesting noise to images ...
but how to get rid of it again?

Overview

Question: noise reduction

- Given a camera and a still scene, how can you reduce noise?

Question: noise reduction

- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

Question: noise reduction

- Given a camera and a still scene, how can you reduce noise?
- Take lots of images and average them!

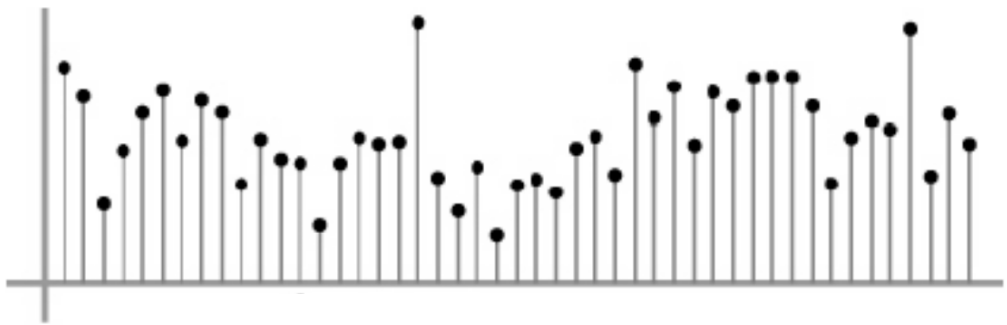
- What's the next best thing?

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

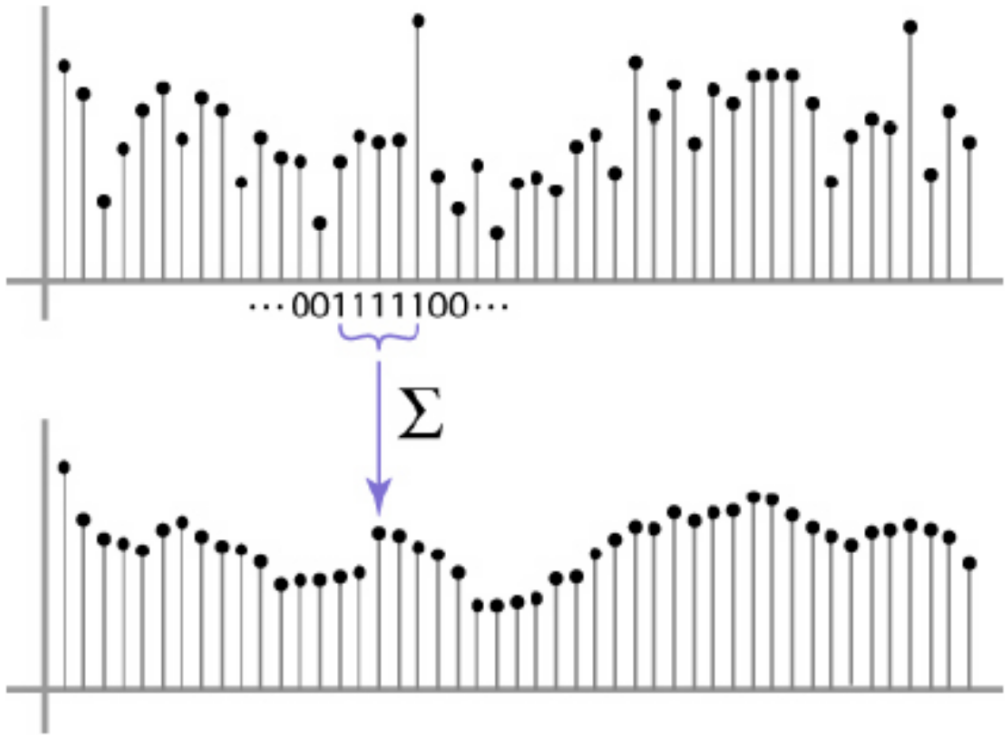
Moving average in 1D

- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



Moving average in 1D

- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- "Kernel" for moving average in 1D: $\frac{[1,1,1,1,1]}{5}$.



Example: stock market indices

German stock market index (DAX) on April 17, 2014. **Blue**: Daily values. **Green**: Averaged over the last 38 days. **Red**: Averaged over the last 200 days.

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0								

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Moving average in 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Generalization: the correlation

- Involves weighted combinations of pixels in small neighborhoods.
- The output pixel's value is determined as a weighted sum of input pixel values

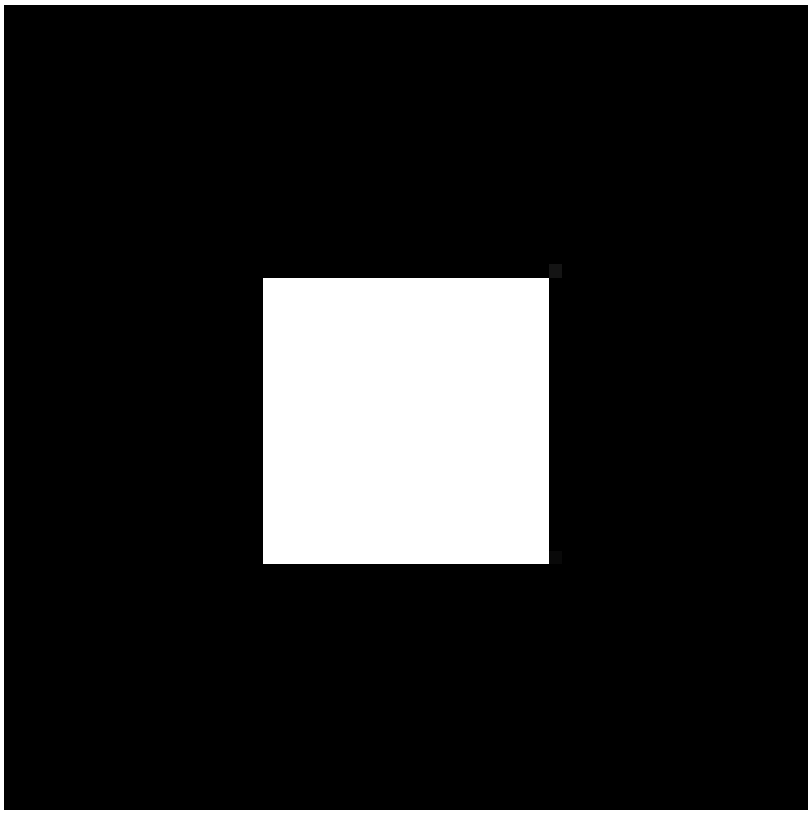
$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i+k, j+l)\, h(k,l)$$

- The entries of the **weight kernel** or **mask** $h(k,l)$ are often called the filter coefficients.
- The operator defined in this way is the **correlation operator** and denoted as

$$g = f \otimes h.$$

Visualizing a 2D kernel

The 3×3 box filter:
moving average

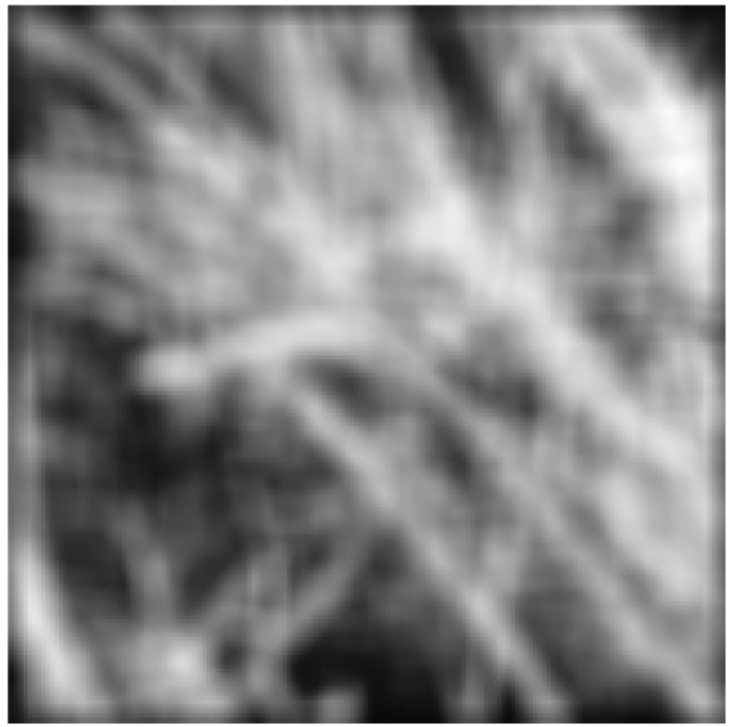


$$
\begin{array}{ccc}
 & k = 0 & \\
 & \vdots & \\
\cdots & \begin{array}{|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
\end{array} & \cdots \quad l = 0 \\
 & \vdots &
\end{array}
$$

Smooting with a box (local mean) filter

depicts box filter:
white = high value, black = low value

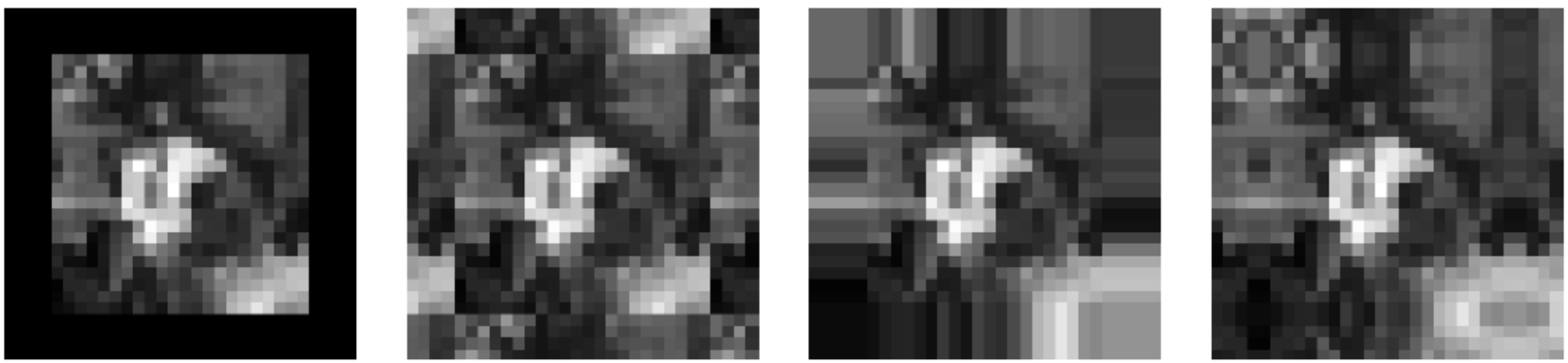
original



filtered

Boundary conditions

- The results of filtering the image in this form will lead to a darkening of the corner pixels.
- The original image is effectively being padded with 0 values wherever the convolution kernel extends beyond the original image boundaries.
- A number of alternative padding or extension modes have been developed.



zero wrap clamp mirror

The Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image (low-pass filter).

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

$F[x, y]$

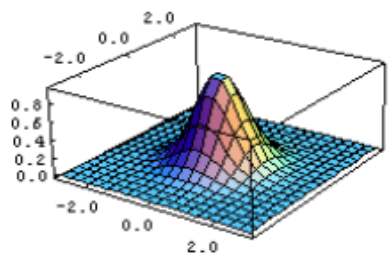
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

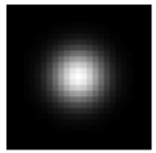
$H[u, v]$

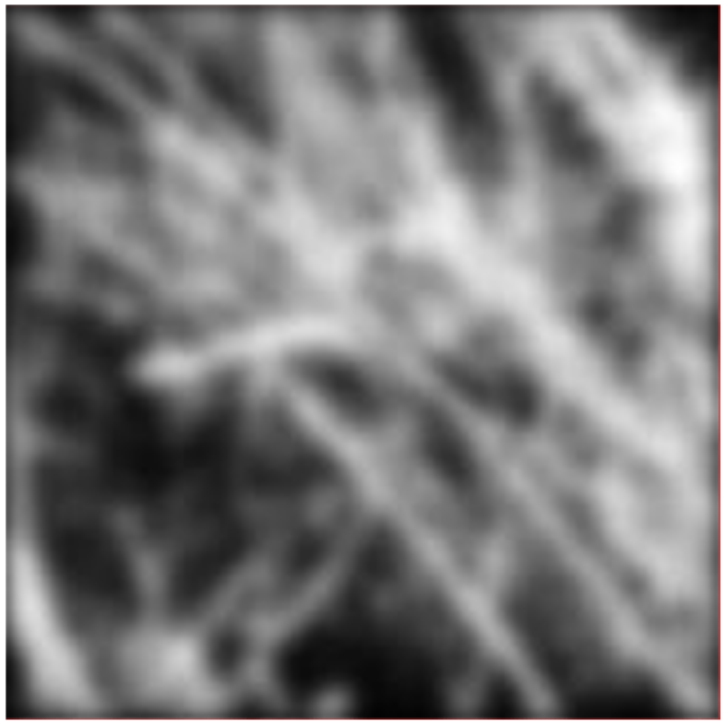
This kernel is an approximation of a 2d Gaussian function:

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

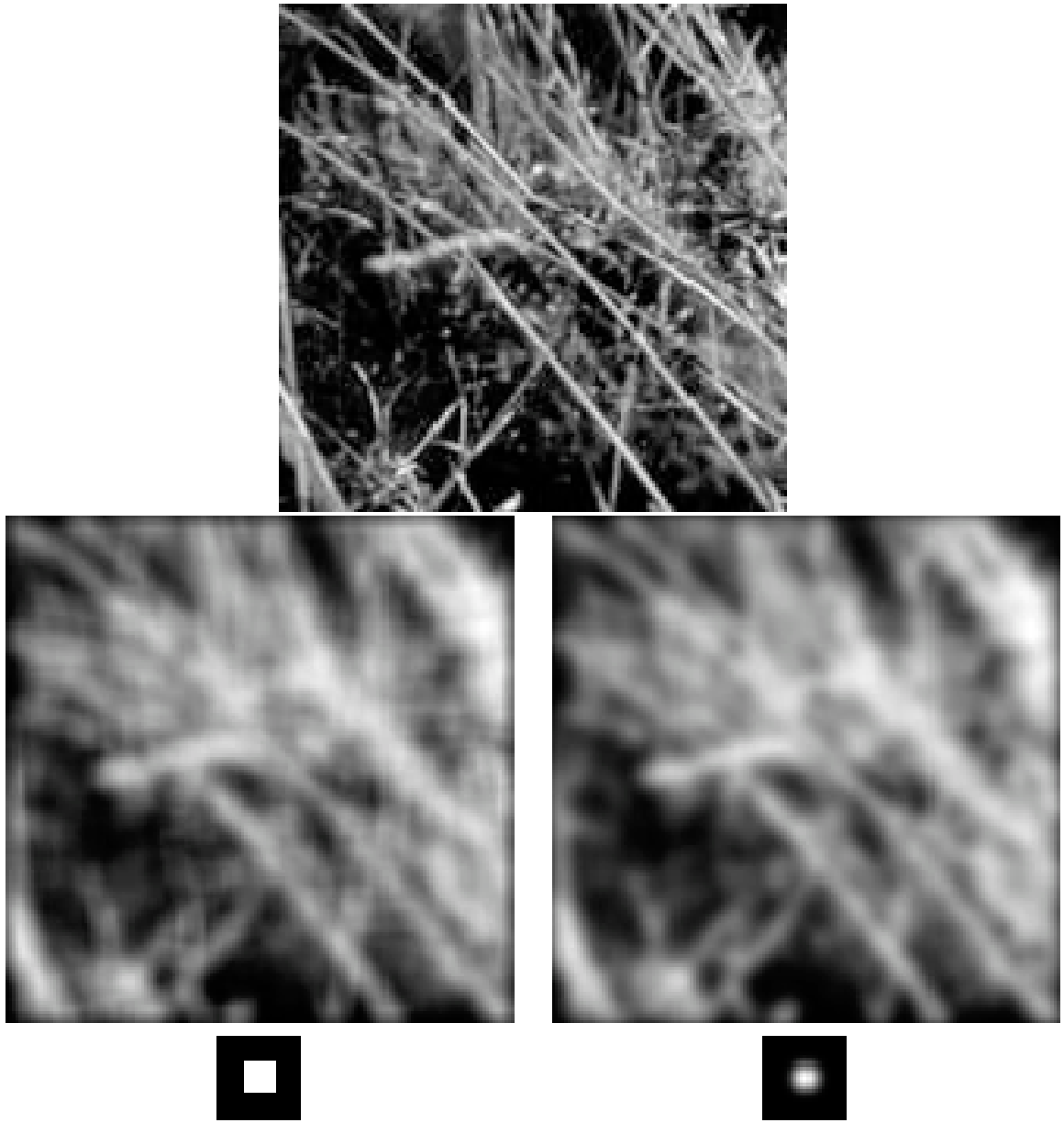


Smoothing with a Gaussian filter



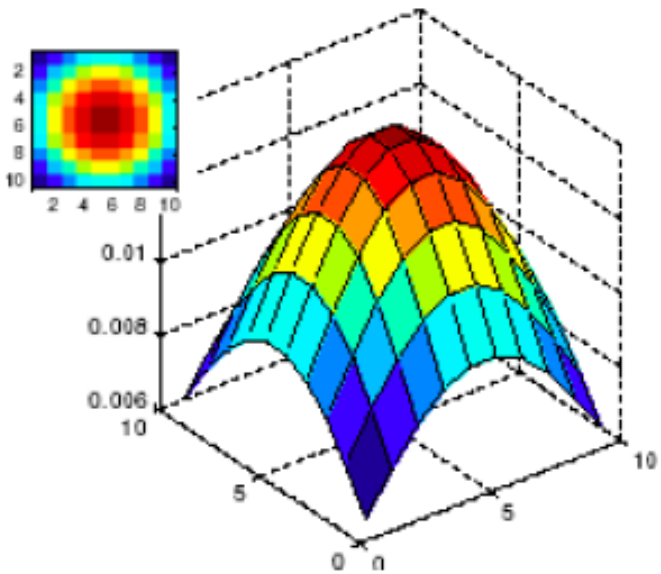


Mean vs. Gaussian

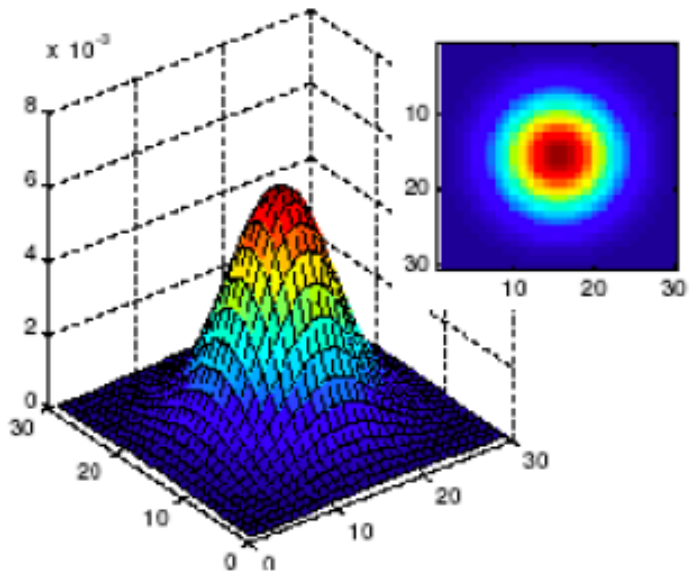


Gaussian filter parameters

- **Size of kernel or mask**: Gaussian function has infinite support, but discrete filters use finite kernels.



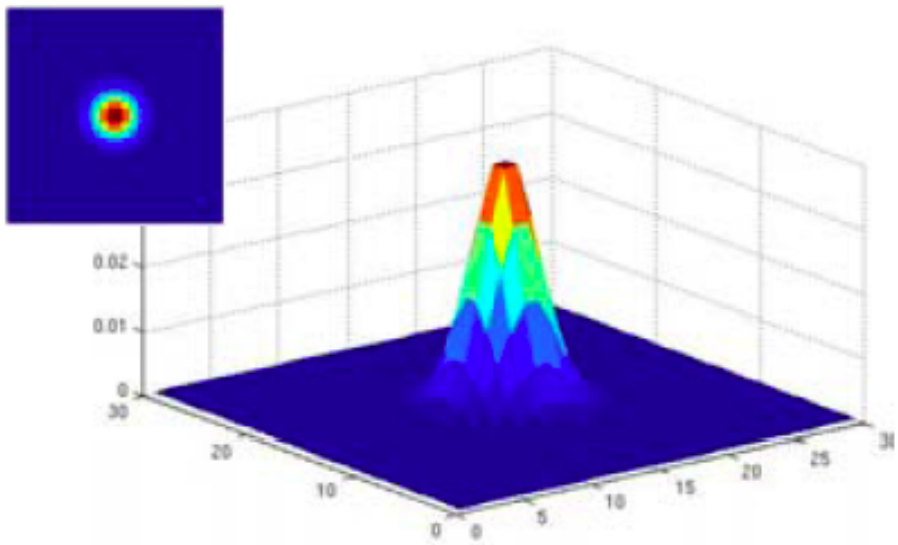
σ = 5 with 10 x 10 kernel



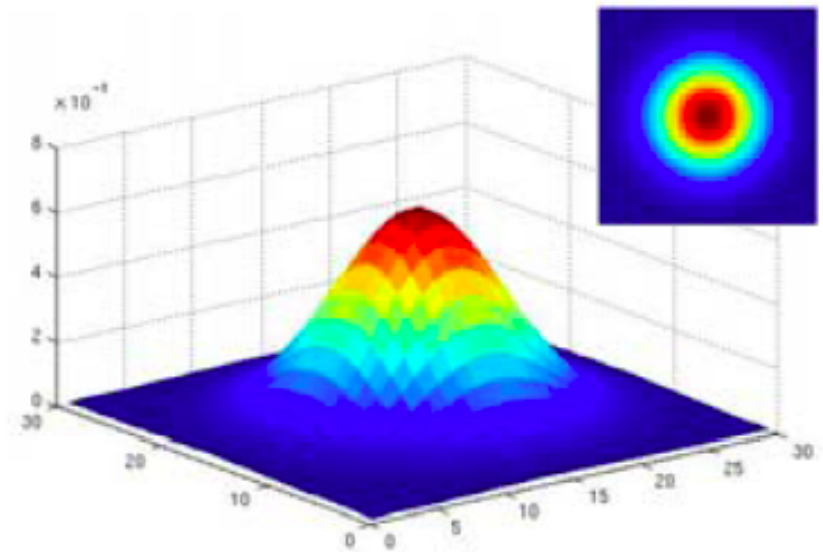
σ = 5 with 30 x 30 kernel

Gaussian filter parameters

- **Standard deviation of the Gaussian**: determines extent of smoothing.

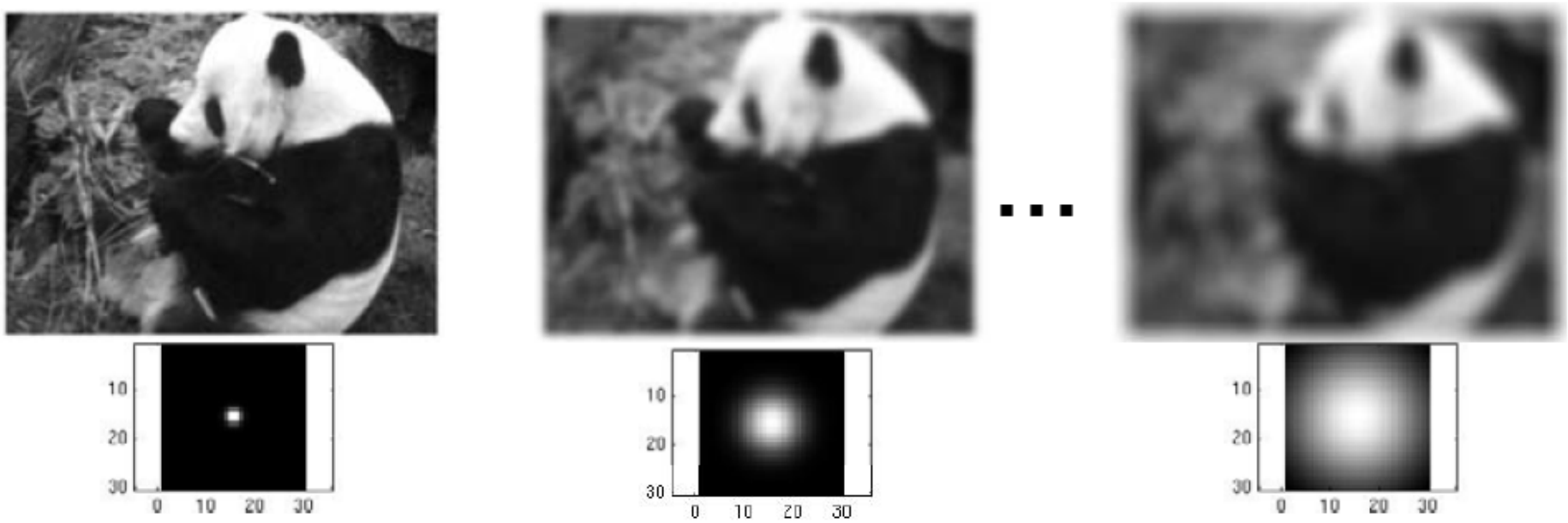


σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

Gaussian filter parameters

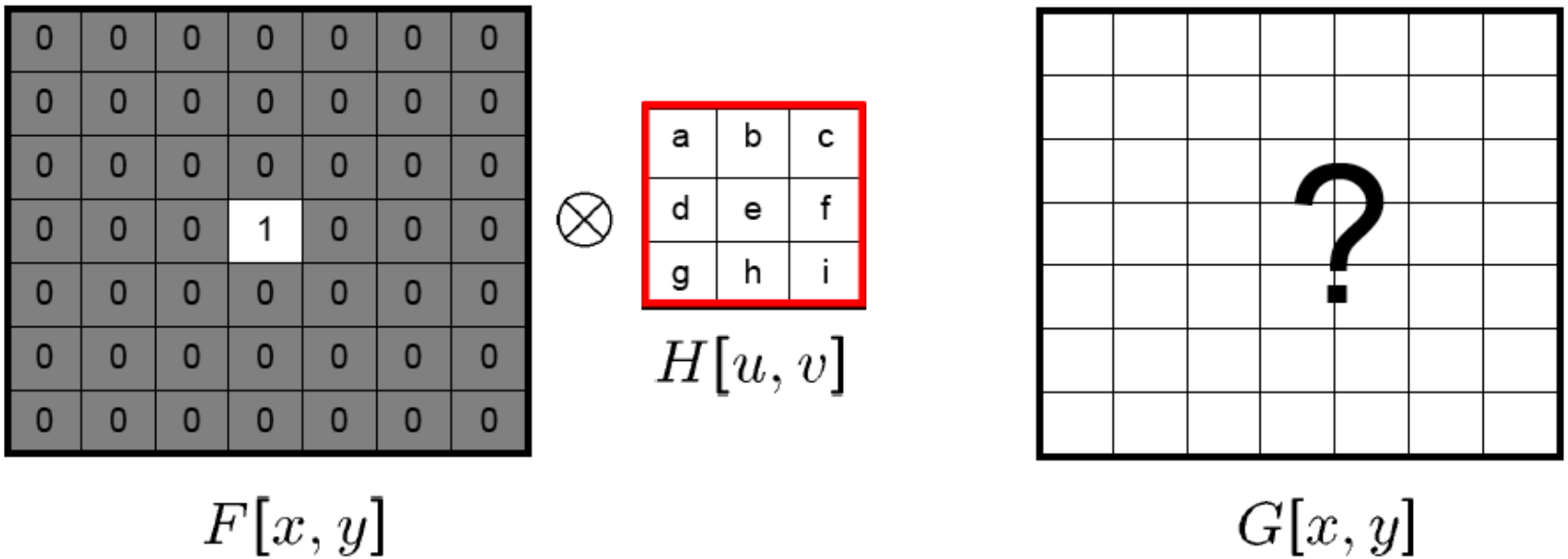



```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

Impulse response of correlation

- Impulse signal δ:

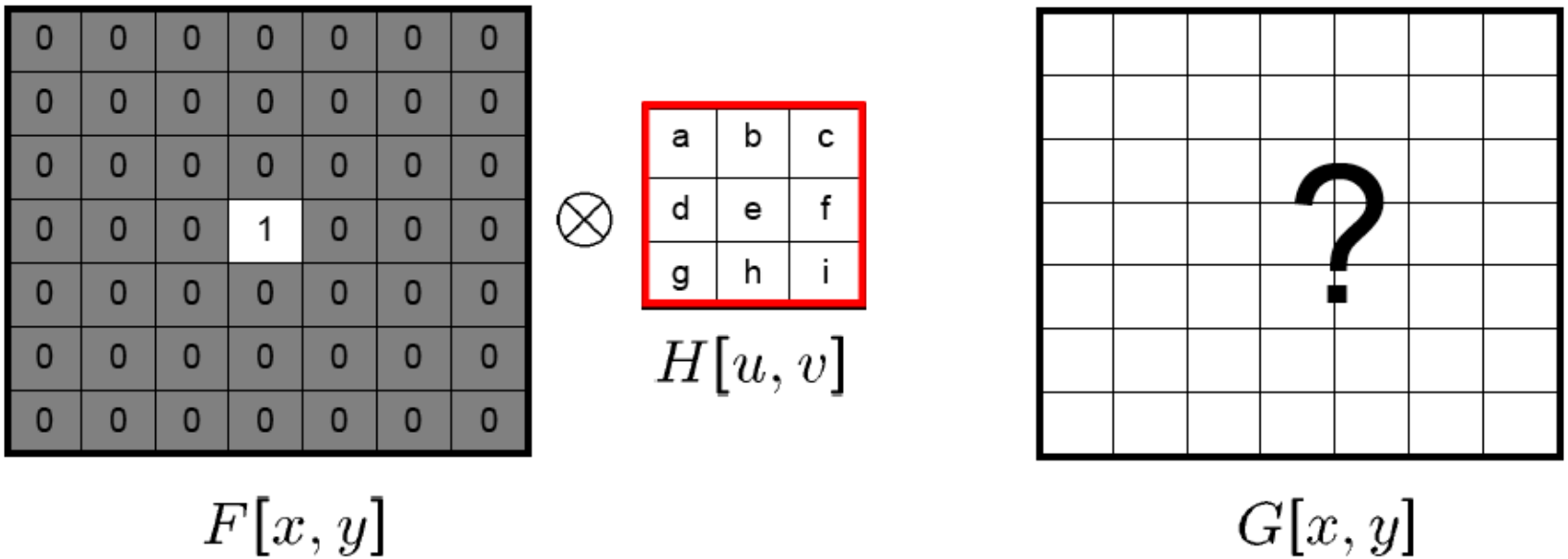
$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0, \\ 0 & \text{else.} \end{cases}$$



Impulse response of correlation

- Impulse signal δ:

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0, \\ 0 & \text{else.} \end{cases}$$



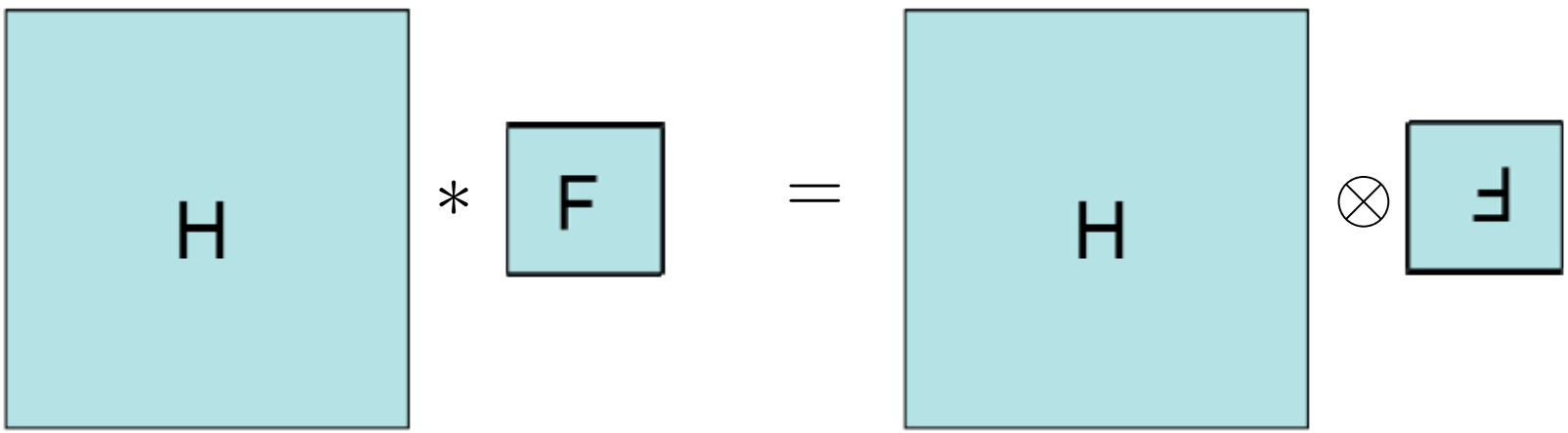
- The filter will be flipped, mathematicians do not like that ...
  - no left-side “identity element” for the correlation operation?
  - not commutative?

Convolution

- The convolution $g = f * h$ of two images f, h is defined as

$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i-k, j-l)\, h(k,l).$$

- Since now $\delta * h = h$, the kernel h is sometimes called the **impulse response function**.
- Equivalent to flipping the filter in both dimensions (bottom to top, right to left) and apply cross-correlation.



Convolution vs. Correlation

- Convolution (Matlab: `conv2`)

$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i-k, j-l)\, h(k,l)$$

$$g = f * h$$

- Correlation (Matlab: `imfilter`)

$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i+k, j+l)\, h(k,l)$$

$$g = f \otimes h$$

- for filters which are symmetric in both dimensions, the output is the same (e.g. Gaussian, box filters)
- for correlation, visualization of the filter is much more intuitive

Convolution vs. Correlation

- Convolution (Matlab: `conv2`)

$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i-k, j-l)\, h(k,l)$$

$$g = f * h$$

- Correlation (Matlab: `imfilter`)

$$g(i,j) = \sum_{k,l \in \mathbb{Z}} f(i+k, j+l)\, h(k,l)$$

$$g = f \otimes h$$

- for filters which are symmetric in both dimensions, the output is the same (e.g. Gaussian, box filters)
- for correlation, visualization of the filter is much more intuitive
- **... that's why they are very frequently mixed up - be aware**

Useful properties of the convolution

- **Commutative**: $f * h = h * f$
- **Associative**: $(f * h) * g = f * (h * g)$
- **Identity element**: $\delta * h = h * \delta = h$
- **Linear (superposition principle)**:

$$f * (h_1 + h_2) = f * h_1 + f * h_2$$
$$(f + g) * h = f * h + g * h$$
$$(\alpha f) * h = \alpha (f * h)$$

- Mathematicians say: $(\{f : \mathbb{Z}^2 \to \mathbb{R}\}, +, *)$ is an "unital commutative algebra over $\mathbb{R}$" (you may want to remember that to impress your friends).
- **Shift-invariant**:

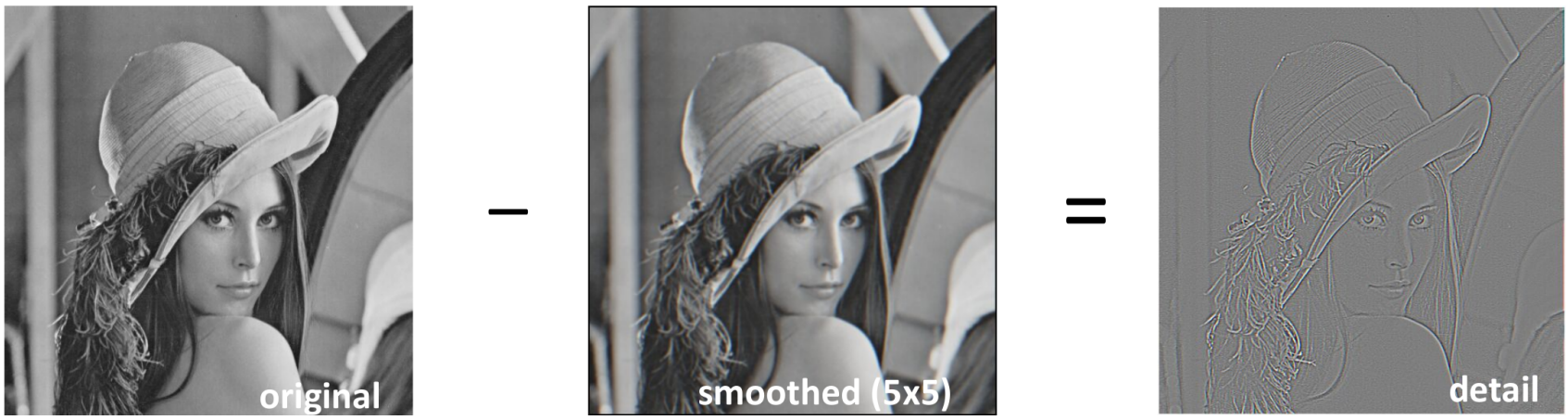
$$(\text{shifted } f) * h = (f * h) \text{ shifted.}$$

This is correct because shifting an image can be expressed by a convolution with a shifted impulse.

We have seen smoothing ... what about sharpening?

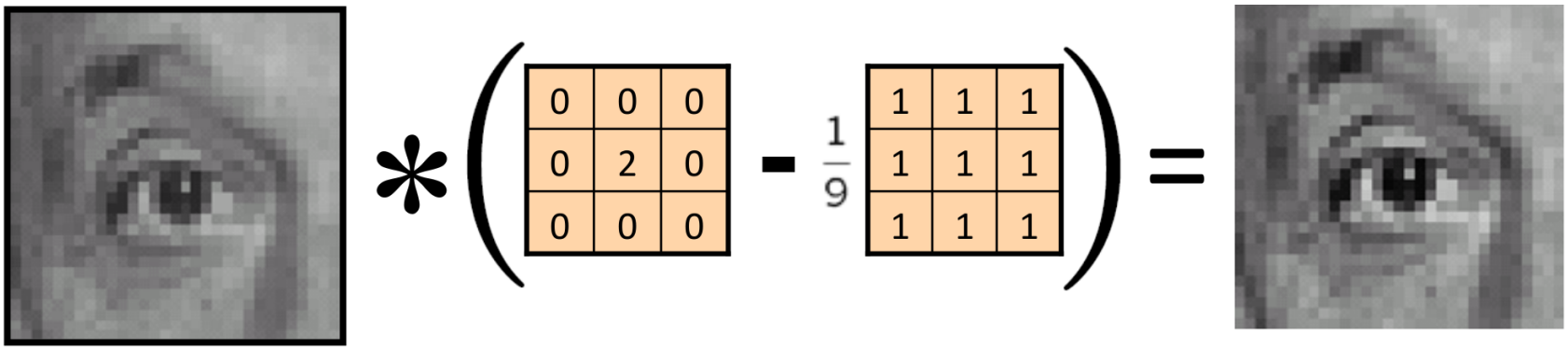
Sharpening idea

- What does blurring take away?



- Let's add it back!

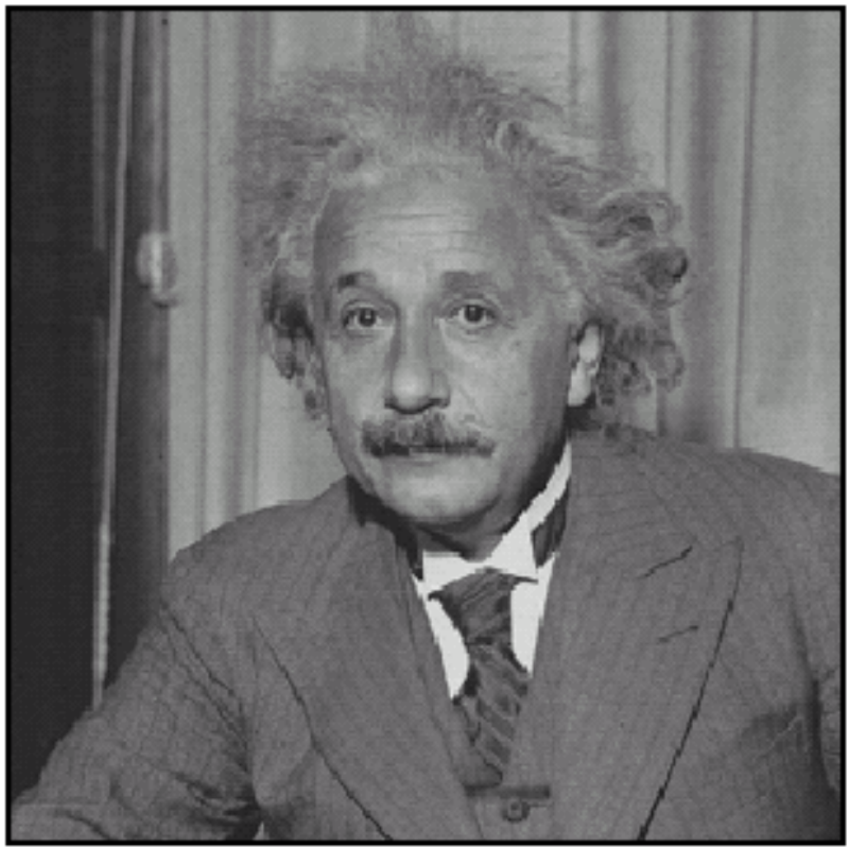
Sharpening filter



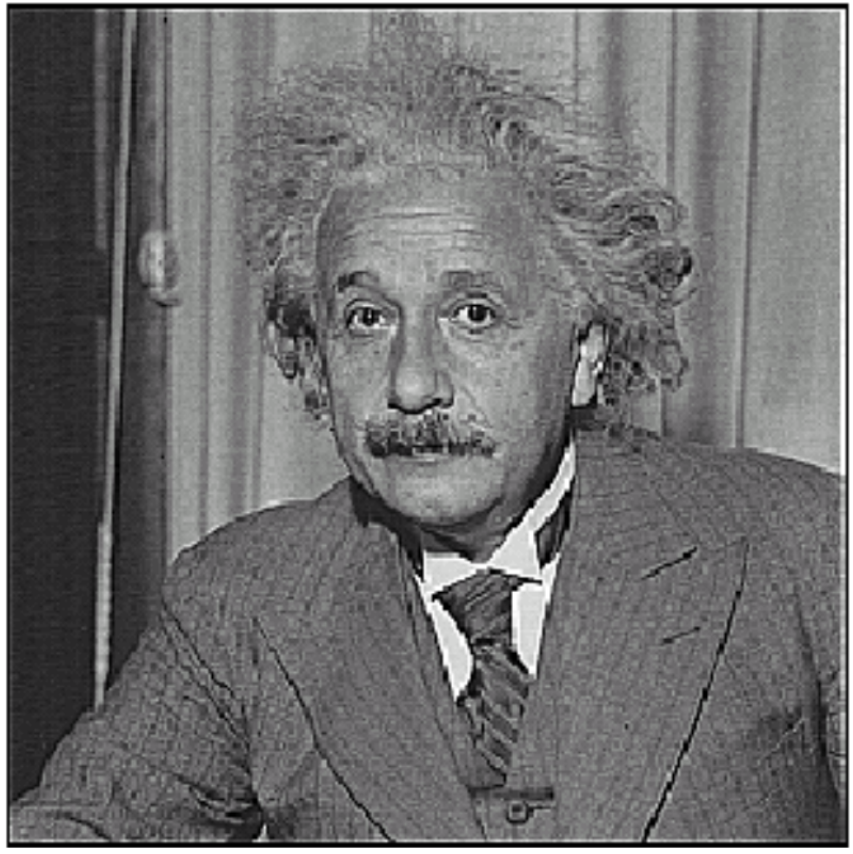
Original

Sharpening filter
(accentuates edges)

Sharpening filter: results



input

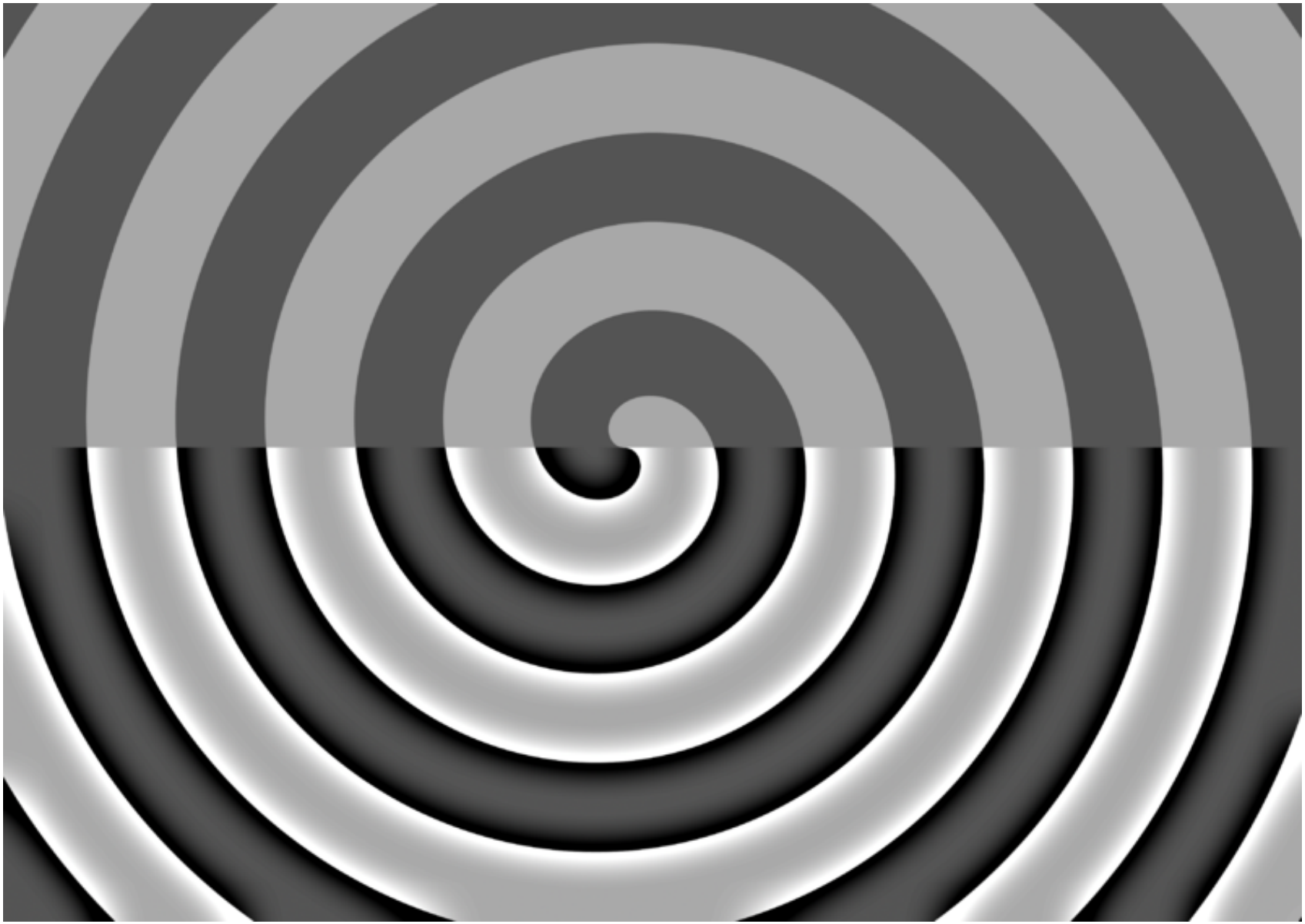


sharpened

Source: D. Lowe

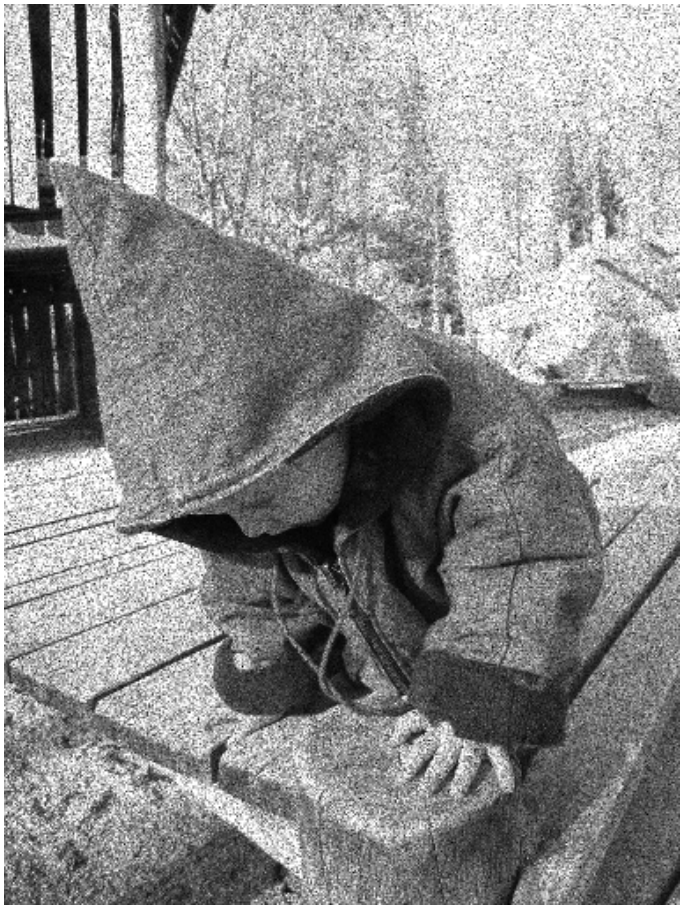
Sharpening filter: results

original

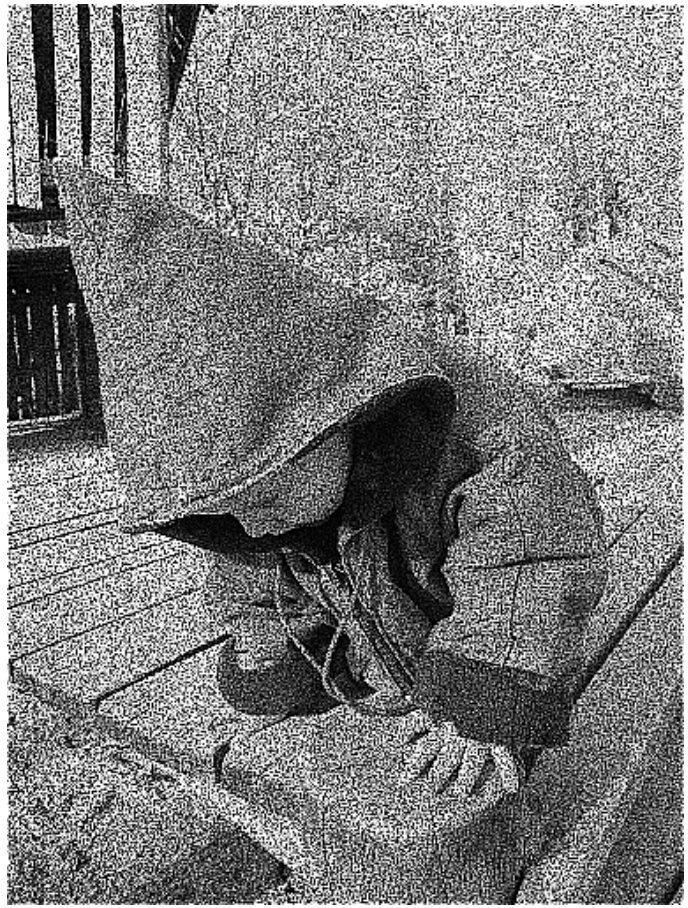


sharpened

Sharpening noisy images ... bad idea



Input with Gaussian noise $\sigma = 0.25 f_{max}$



Result of sharpening, $\alpha = 3.0$

Note: noise is “high-frequency” content, i.e. detail.

Convolution frequently occurs in optical systems

“Optical” convolution

- Camera shake or motion blur

=

*

“Optical” convolution

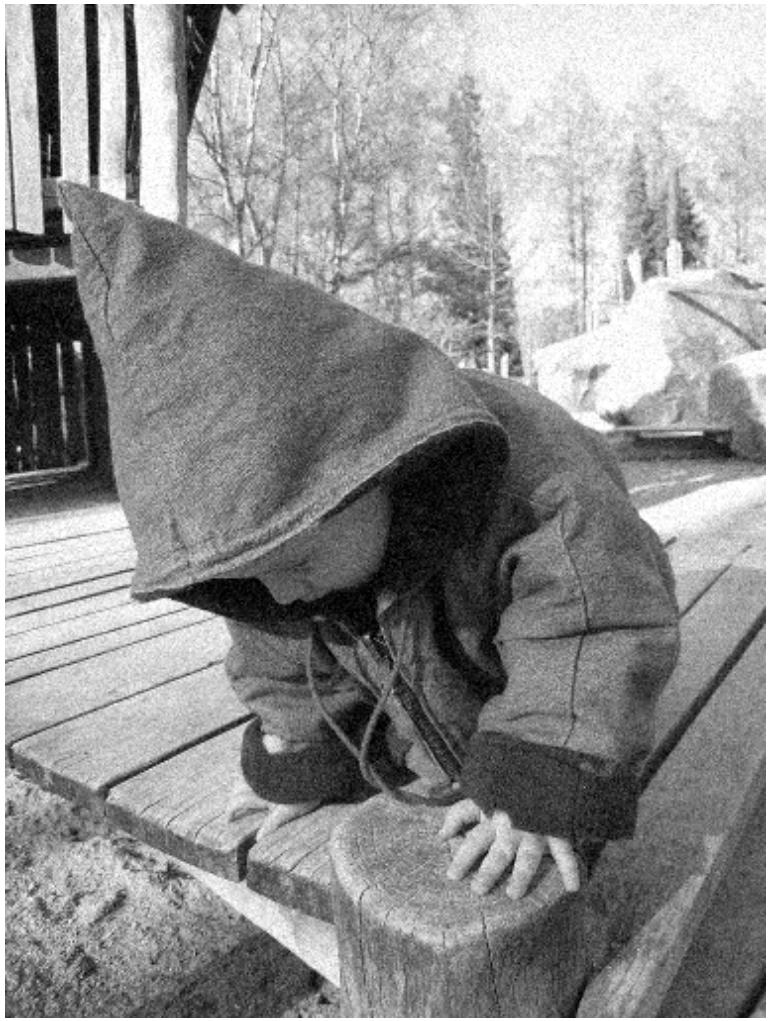
- Camera shake or motion blur


- Blur in out-of-focus regions of an image
  (related to shape of the aperture, see later chapter).


Overview

that was all nice and well, but remember the original goal -
what is the actual denoising performance of the filters?

Denoising Gaussian noise with a Gaussian filter



Gaussian additive noise, $\sigma = 5\% f_{max}$

Gaussian filter, $\sigma = 1.0$

Does not look too good subjectively - no preservation of edges.
But what is the best choice of parameter?

How to measure denoising quality? Two common measures.

Note: we often use the notation $(f_n)_{n=1,\dots,N}$ for all pixels in an image, if it is not important to distinguish the dimensions.

Mean square error (MSE)

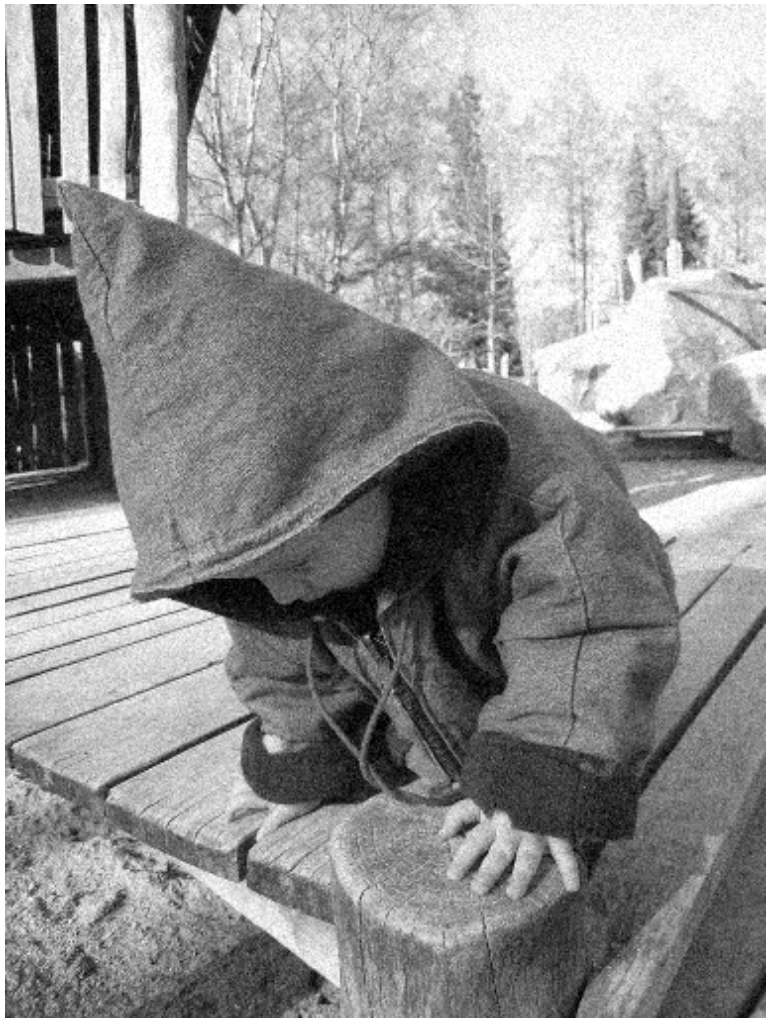
$$MSE(f,g) = \frac{1}{N}\sum_{n=1}^{N}(f_n - g_n)^2.$$

Peak signal-to-noise ratio (PSNR). For a maximum greyscale value of V,

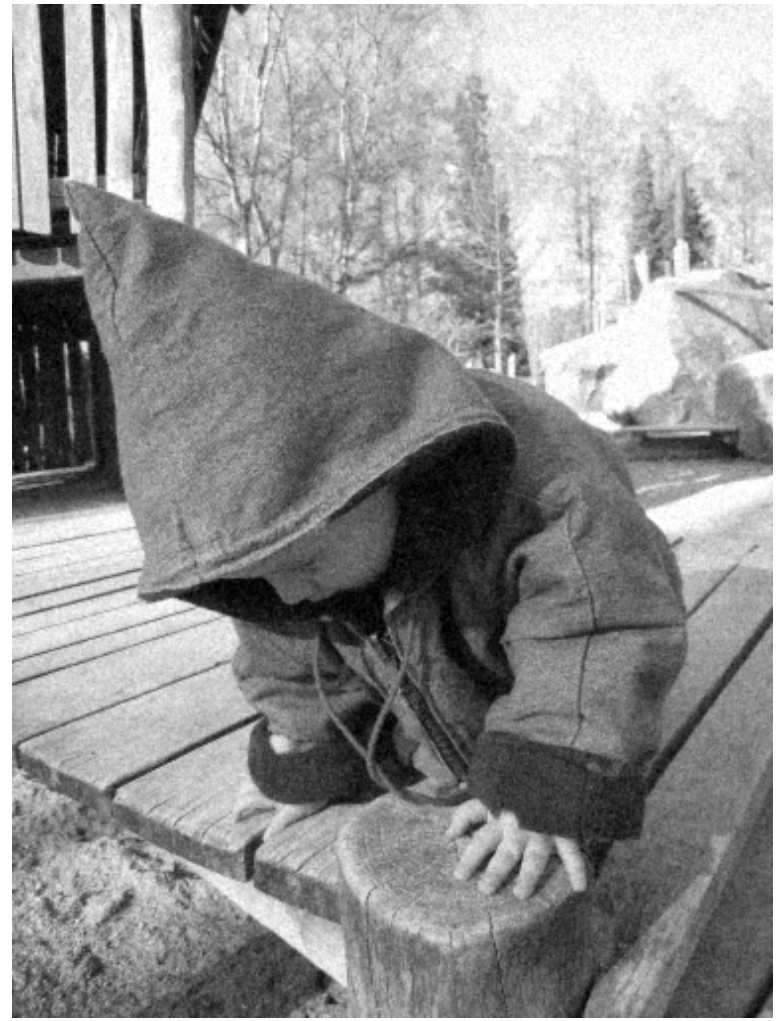
$$PSNR(f,g) = 10\ \log_{10}\left(\frac{V^2}{MSE(f,g)}\right) \quad \leftarrow \frac{\text{peak signal}^2}{\text{noise}^2}$$

Note: both do not correspond very well to human perception - check out “structural similarity” (SSIM) for something better.

Denoising with Gaussian filter - best result



Gaussian additive noise, $\sigma = 5\% f_{max}$, PSNR=27.28



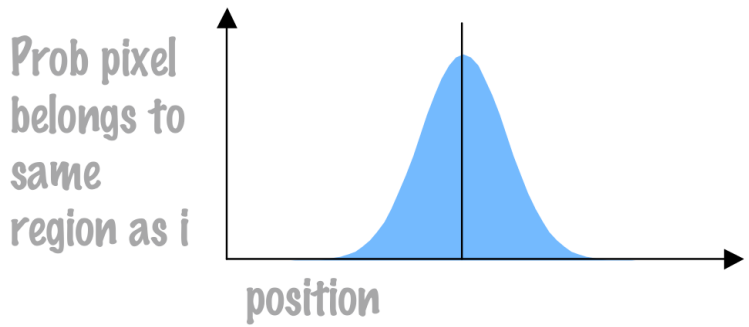
Gaussian filter, $\sigma = 0.5$, PSNR=29.18

The problem are the edges - how can we do better?

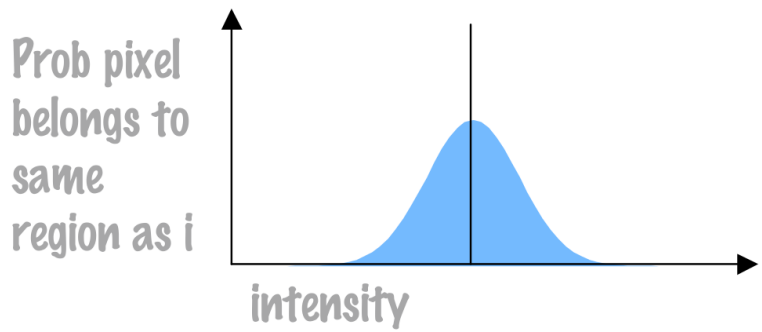
The bilateral filter

- **Key idea:** To preserve edges, average only over pixels which are nearby **and** have similar color.

- Two weights, one according to distance and one according to intensity, both Gaussian:



$N_\sigma(\|x_n - x_m\|)$



$N_\tau(|f_n - f_m|)$

- **Note on notation**: x_n, x_m are the locations of the pixels, $1 \leq n, m \leq N$, and f_n, f_m the corresponding intensities. σ, τ are the standard deviations for the two Gaussian distributions N_σ, N_τ.

The bilateral filter

- For parameters $\sigma, \tau > 0$ and a greyscale image f, the bilateral filter in the nth pixel is defined as

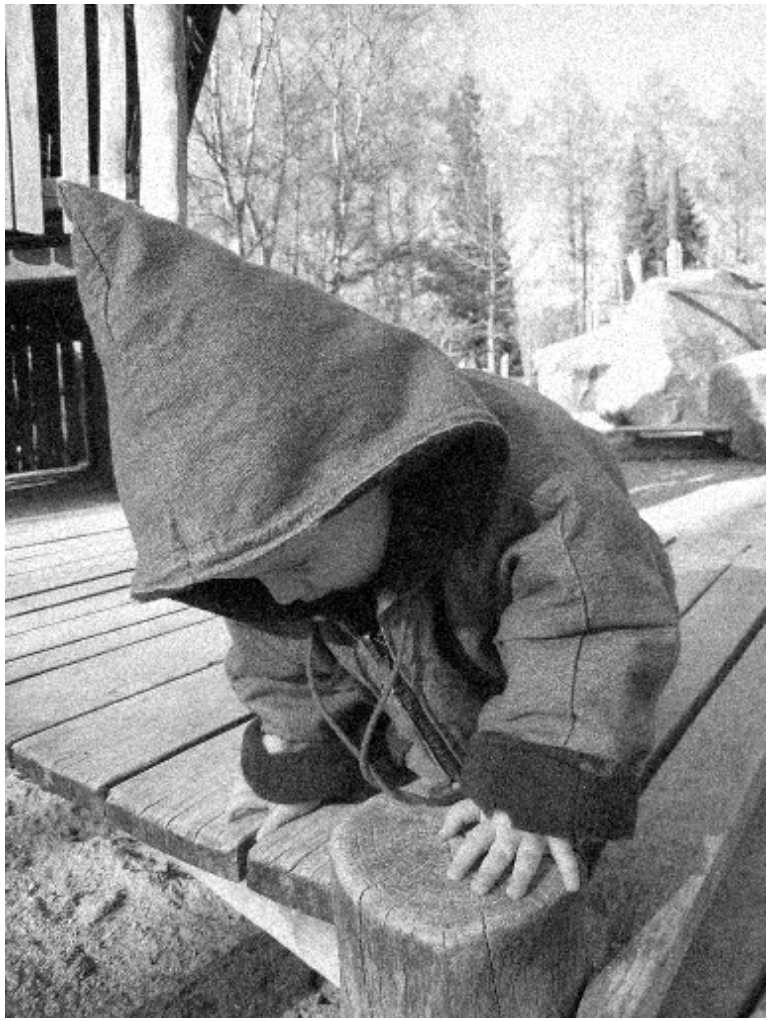
$$g(n) = \frac{1}{K_n} \sum_{m=1}^{N} f_m \, N_\sigma(\|x_n - x_m\|) \, N_\tau(|f_n - f_m|)$$

$$\text{with } K_n = \sum_{m=1}^{N} N_\sigma(\|x_n - x_m\|) N_\tau(|f_n - f_m|).$$

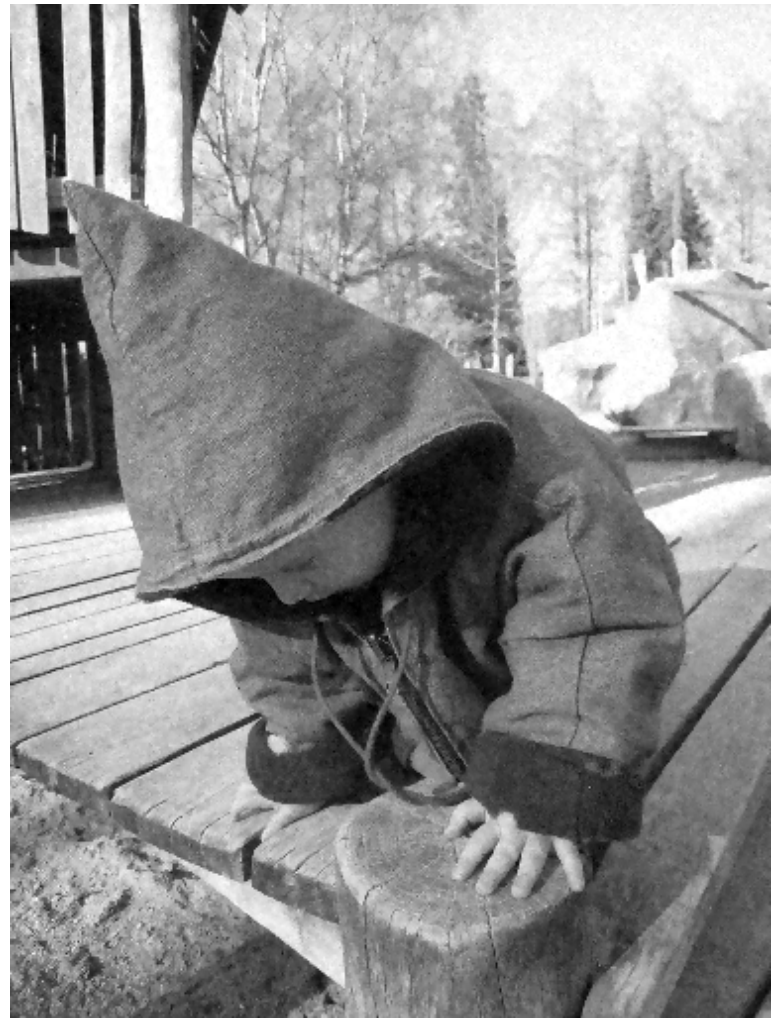
- Note: Although the sum runs over all pixels, in practice, one will only compute it over a small window with a distance of about three times σ around x_n.

- Note: for $\tau \to \infty$, the result approximates a Gaussian filter with standard deviation σ.

Denoising with bilateral filter - best result in terms of PSNR

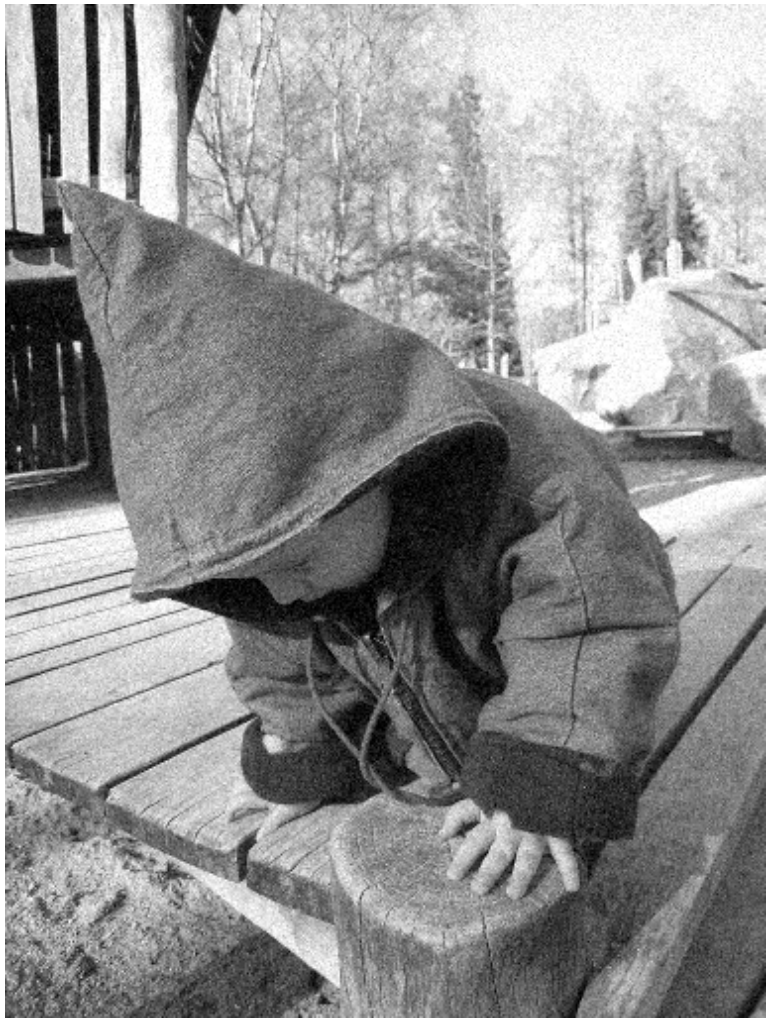


Gaussian additive noise, $\sigma = 5\% f_{max}$, PSNR=27.28

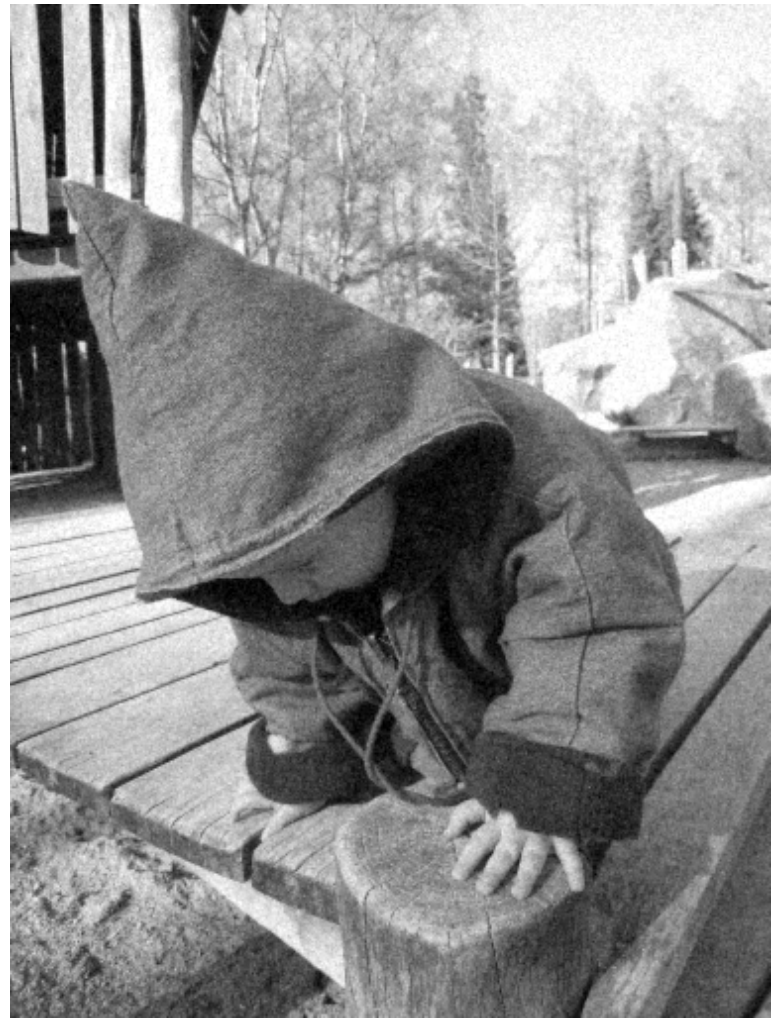


Bilateral filter, $\sigma = 1, 0.1$, PSNR=30.20

Compare to Gaussian filter - best result in terms of PSNR



Gaussian additive noise, $\sigma = 5\% f_{max}$, PSNR=27.28



Gaussian filter, $\sigma = 0.5$, PSNR=29.18

Non-local means

- **Observation:** images have similar patches all over the place
- **Idea:** When averaging, put most weight on the most similar patches
- **In formulas:**
  - around each pixel n, define a $k \times k$ window of pixels $\mathcal{W}_n$, which is written as a vector of pixel indices.
  - define a distance $d_{n,m}$ between pixels as

$$d_{n,m}^2 := \sum_{s=1}^{k^2} \left(f(\mathcal{W}_n(s)) - f(\mathcal{W}_m(s)) \right)^2.$$

  - kernel weights are then given by

$$k_{n,m} := \exp\left(-\frac{d_{n,m}^2}{2\sigma^2} \right),$$

    where $\sigma > 0$ is a parameter.

[Baudes et al. 2005]

Non-local means

- As usual, with all kernel weights defined, the nth pixel in the filtered image g is then defined as

$$g(n) := \frac{1}{\sum_{m=1}^{N} k_{n,m}} \sum_{m=1}^{N} k_{n,m} f_m.$$

- Note: for practical reasons, the sums above are usually also restricted to a window around the nth pixel.

- Sometimes, the process is iterated to get a better estimate with already pre-denoised patches.

- However, in any case, this filter is usually very, very slow.

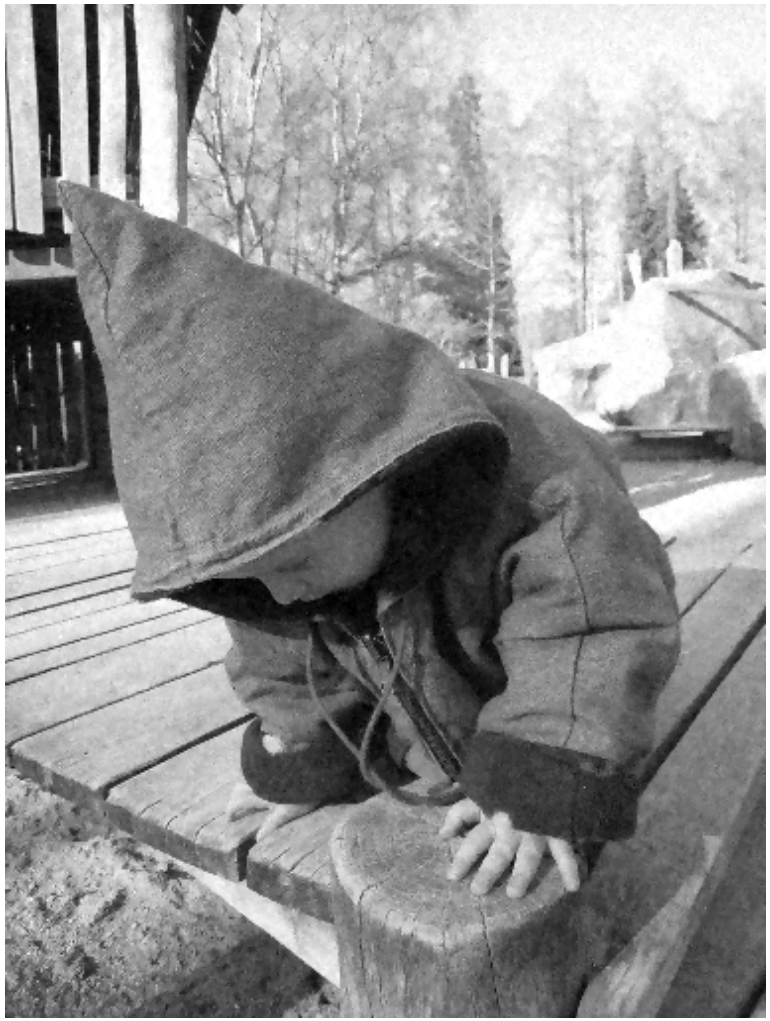
Denoising with NL means - best result in terms of PSNR

Original

NL means, $\sigma = 0.4$, PSNR=30.12

Compare to bilateral filter - best result in terms of PSNR

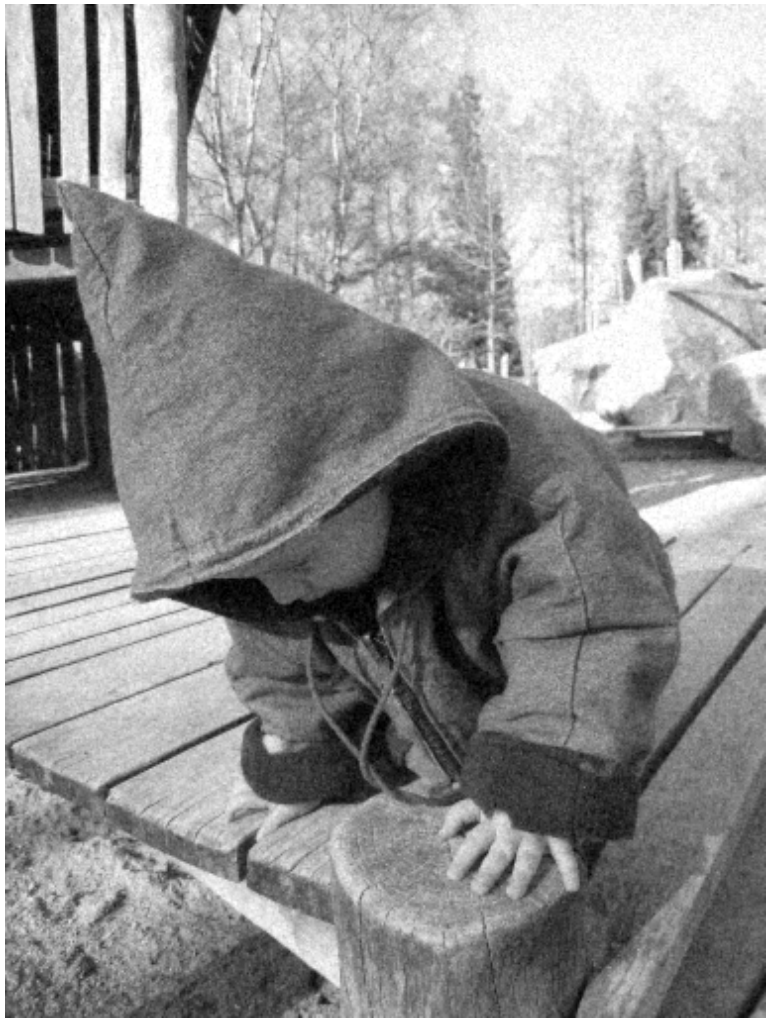
Original



Bilateral filter, $\sigma = 1, 0.1$, PSNR=30.20

Compare to Gaussian filter - best result in terms of PSNR

Original



Gaussian filter, $\sigma = 0.5$, PSNR=29.18

So far, we have only checked Gaussian noise -
what about other types of noise?

Denoising salt and pepper noise

Salt and pepper noise, $p = 0.2$

Local averaging filter, window size 5×5

Denoising salt and pepper noise

Salt and pepper noise, $p = 0.2$

Local averaging filter, window size 5×5

- Not zero mean locally
- Averaging biases the results locally

Median filter

- Very good for removing outliers
- For each pixel n, compute the vector $\mathbf{f}$ consisting of k^2 grayscale values

$$\mathbf{f} = (f_{n,1}, ..., f_{n,k^2})$$

  in a local window of size $k \times k$ around n.

- The filtered result $g(n)$ at that location is defined as the **median value** of this vector. Informal definition: sort all elements in the vector by size and take the middle element of the result (i.e. the one at position $(k^2 + 1)/2$ if k is odd as usual).
- The underlying idea is a piecewise constant image model, we will skip the details here.

Median filter: results

Salt and pepper noise, $p = 0.2$

Median 3×3

Median filter: results

Salt and pepper noise, $p = 0.2$

Median 5×5

Median filter: results

Median 3 $\times$ 3 closeup

Median 5 $\times$ 5 closeup

Iterated median filter

Median 3 $\times$ 3 applied twice

Median 5 $\times$ 5 closeup

Overview

Summary

- In image analysis, **convolution** is by far the most important filter.
- Convolution is linear and shift-invariant.
- For the application of image denoising, the best filters are non-linear and non-local.
- Know your noise, and you will be able to select the appropriate filter for denoising - ignore what noise you have, and failure to remove it properly will be a likely result.
- One can get better by at denoising for example by learning how images should look like from training data - this is however seminar material.
- We will explore a few of the manifold applications of convolution in the next chapter, when we turn towards analyzing image derivatives and local features.