Image Processing and Computer Graphics

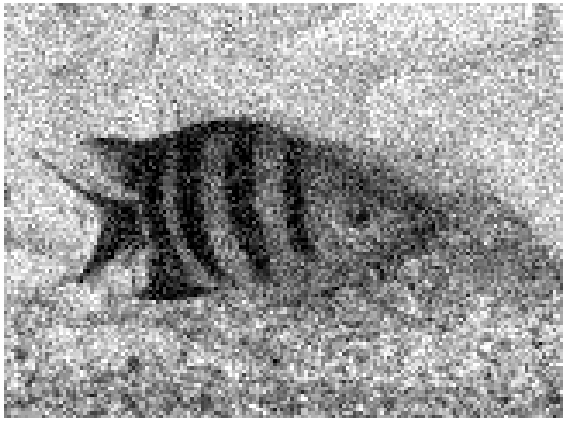
Image Processing

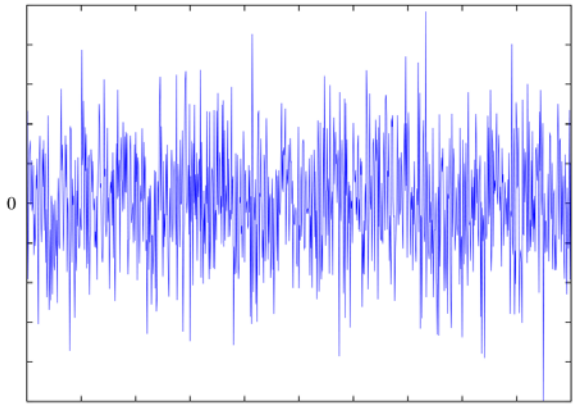
Class 3

Noise, basic operations, and filters

What is noise?

- Noise is a disturbance of the image data by image acquisition or the transmission of images
- Sometimes the term noise is also used more generally for the component of the data that does not fit the underlying model (model noise)
- One classical goal of image enhancement: removal of noise
- Computer vision: rather than removing noise, choose a model that can deal with noise

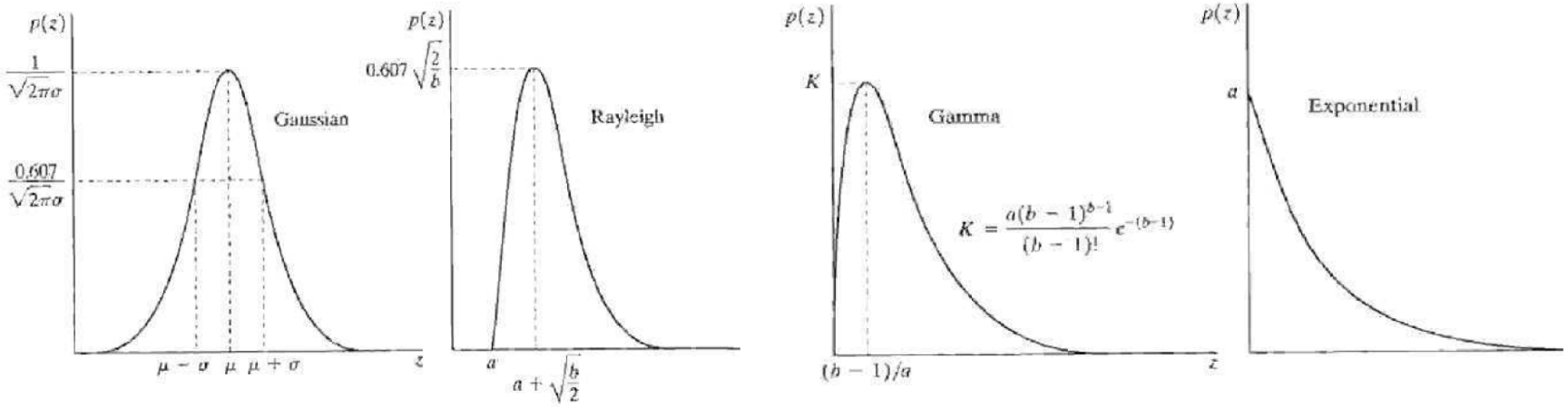




Additive noise

- Assumption: gray values and noise are independent: $I_{ij} = I^*_{ij} + n_{ij}$

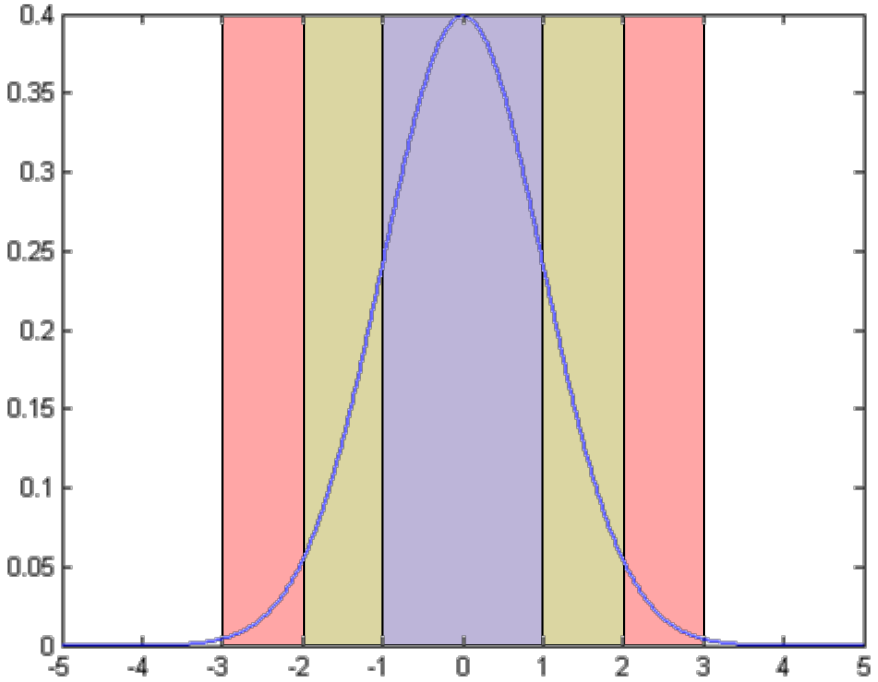
- Noise distribution depends on the sensor
  - Poisson noise, Gaussian noise (CCD cameras)
  - Rayleigh distribution (radar)
  - Gamma distribution (laser imaging)
  - Exponential distribution (laser imaging)



From Gonzales-Woods 2002

Gaussian noise and Gaussian distribution

- Density function $G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- Usually zero-mean Gaussian noise
- Good approximation in many practical situations
- In particular: thermic sensor noise in CCD cameras
- 1σ interval: 68% of values
- 2σ interval: 95.5% of values
- 3σ interval: 99.7% of values



Gaussian with $\sigma = 1$.

Gaussian noise

Lena test image without noise

Gaussian noise with $\sigma = 20$ added

Multiplicative noise

- Signal dependent: $I_{ij} = I^*_{ij}(1 + n_{ij})$
- More difficult to handle in image enhancement algorithms
- Useful trick:
  - Transform the image by applying the logarithm

    $$\log I_{ij} = \log I^*_{ij} + \log(1 + n_{ij})$$

  - Noise becomes additive noise (can be removed by standard algorithms)
  - Apply backtransform to denoised image

    $$I^*_{ij} = \exp \log I^*_{ij}$$

Impulse noise

- A certain percentage of pixels is replaced by one (unipolar impulse noise) or two (bipolar impulse noise) fixed values
- Caused, for instance, by pixel defects of CCD chips
- Special case salt-and-pepper noise: some pixels replaced by white or black values

Original

5% salt-and-pepper noise

20% salt-and-pepper noise

Uniform noise

- A certain percentage of pixels is replaced by uniformly distributed random variables
- Very unpleasant noise, no a-priori knowledge in the noise model

Original

5% uniform noise

20% uniform noise

Evaluation, Signal-to-noise ratio (SNR)

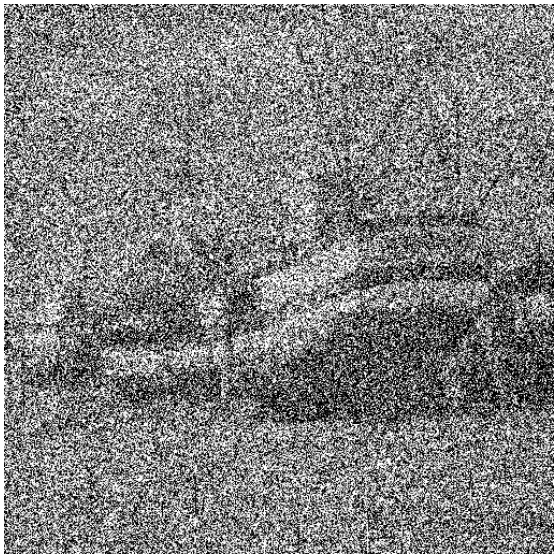
- Quantitative measure for the degradation of an image I versus a noise-free version I^* (**ground truth**)
- Based on the variance of the image versus the variance of the noise
- Variance of the image: $\sigma_I^2 = \frac{1}{N}\sum_i (I_i^* - \mu)^2$
- Additive, zero-mean noise model: $I_i = I_i^* + n_i$
- Variance of the noise: $\sigma_n^2 = \frac{1}{N}\sum_i (I_i^* - I_i)^2$
- **Signal-to-noise ratio**: $\mathrm{SNR} = 10 \log_{10}\left(\frac{\sigma_I^2}{\sigma_n^2}\right) = 10 \log_{10}\left(\frac{\sum_i (I_i^* - \mu)^2}{\sum_i (I_i^* - I_i)^2}\right)$
- **Peak signal-to-noise ratio**: $\mathrm{PSNR} = 10 \log_{10}\left(\frac{N(\max_i I_i^* - \min_i I_i^*)^2}{\sum_i (I_i^* - I_i)^2}\right)$
- Their unit is decibel (dB), the higher the better

Peak signal-to-noise-ratio

Noise free image

$\sigma_n = 2$, PSNR = 42.1

$\sigma_n = 10$, PSNR = 28.15



$\sigma_n = 200$, PSNR = 7.63

Point operations

- The most straightforward way to enhance an image is to treat each pixel independently:

$$u_{ij} = f(I_{ij})$$

- This kind of operation mainly transforms intensities in a way that relevant structures are in a range that can be well observed.

- Example: changing brightness

$$u(x, y) = I(x, y) + b$$

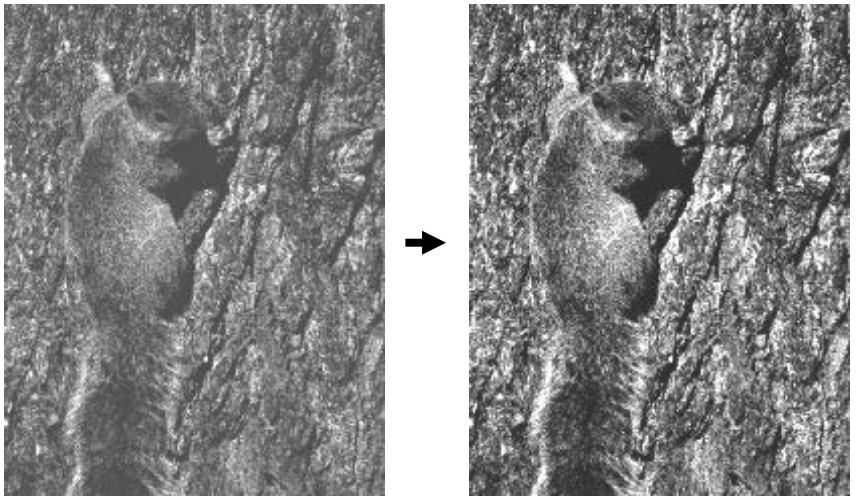
  - Darkening for $b < 0$
  - Values that exceed the allowed range must be clipped

Contrast enhancement and Gamma correction

- Contrast enhancement

$$u(x,y) = aI(x,y)$$

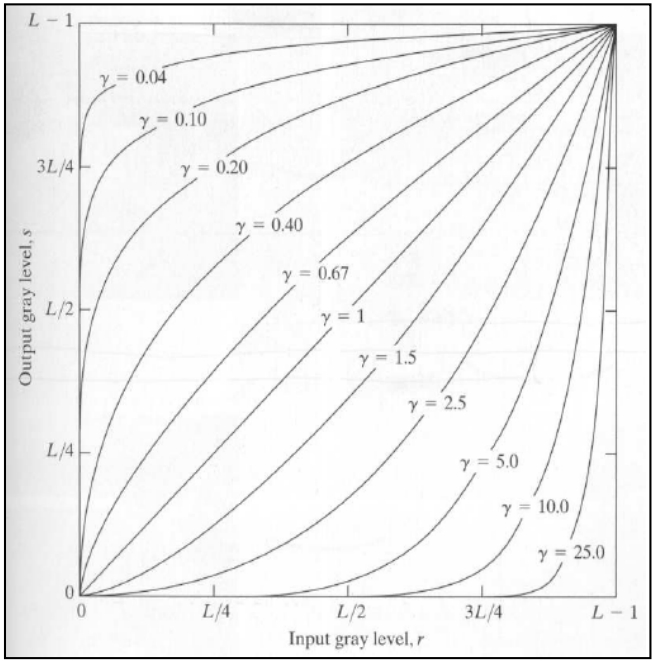
  - $a > 1$
  - Contrast attenuation for $a < 1$
  - Clipping of values that exceed the allowed range



- Gamma correction
  - Camera chips have different response curves than the human eye, usually $I \propto I^{\gamma}$
  - Compensation of these effects by a gamma correction

$$f(I(x,y)) = I_{\max}\left(\frac{I(x,y)}{I_{\max}}\right)^{\frac{1}{\gamma}}, \quad \gamma > 0$$

  - The range $[0, I_{\max}]$ is not affected



From Gonzales-Woods 2002

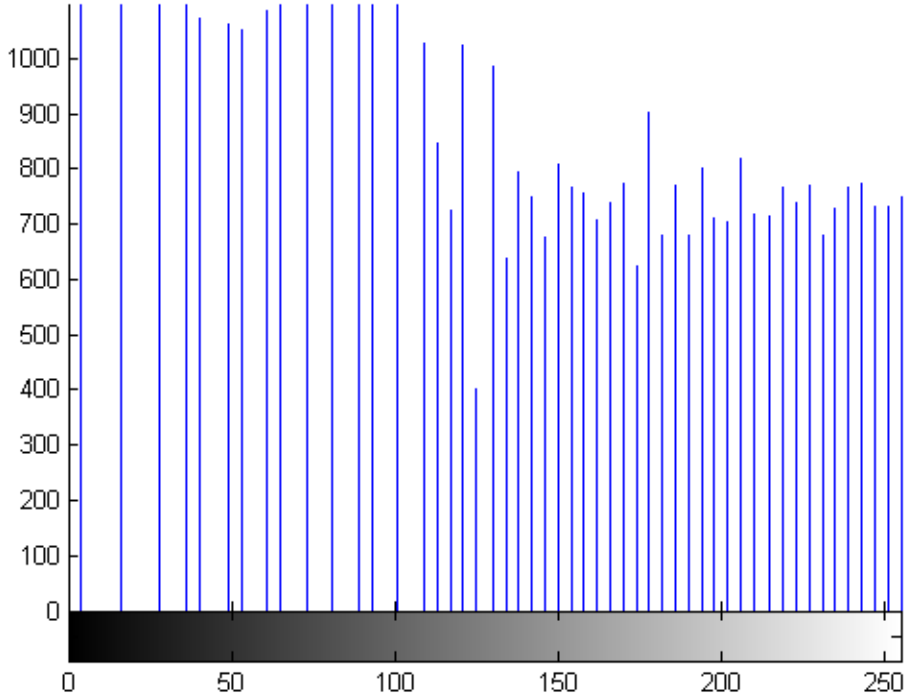
Gamma correction

Dark areas become brighter without leading to oversaturation in the brighter areas

Gray value histogram

- The gray value **histogram** contains the number of pixels in the image that have a certain gray value

Input image



Histogram

- Histogram equalization: transformation such that all gray values are equally frequent

Difference image

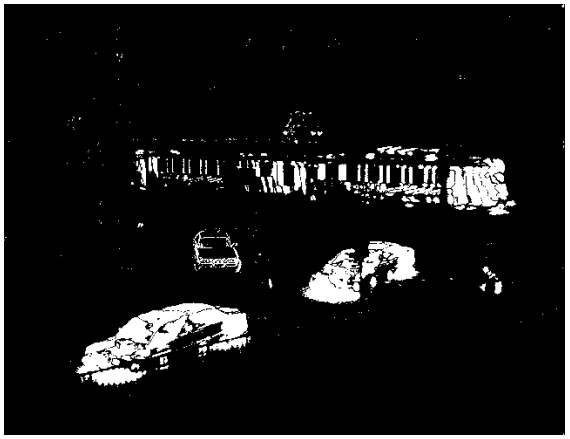
- Subtracting the grayvalues of two images from each other yields a difference image:

$$I_\Delta = |I_1 - I_2|$$

- Can be used for detecting parts of moving objects in static scenes

Input images

Difference image



Difference image after thresholding

Background subtraction

- Special difference image: **background subtraction**
  - Take one image of the static background without the object
  - Difference image to this background image indicates the object
  - Can be used for object tracking and segmentation of a person in front of a static background

- Difference image of color images: $I_\Delta = \frac{1}{3}\sum_{k=1}^{3} |I_{k,1} - I_{k,2}|$

Input images

Difference image

Difference image after thresholding

Linear filters, convolution theorem

- Filters take neighboring pixels into account to improve the signal

- In signal processing, filters are often designed in the Fourier domain
  → global frequency analysis, takes all pixels into account

- Filtering in the Fourier domain can be translated to filtering in the spatial domain via the **convolution theorem:**

$$\mathcal{F}(f) \cdot \mathcal{F}(h) = \mathcal{F}(f * h)$$

Fourier transform

- In image processing, we usually design the filter h in the spatial domain instead of $\mathcal{F}(h)$ in the Fourier domain because:
  - More interested in a local analysis
  - Easier handling of image boundaries
  - Easier to generalize to nonlinear filters

Convolution

- Convolution with a static filter h is a **linear operation**
  → linear filtering

$$(f * h)(x) := \int h(-x') f(x + x') dx'$$

- Convolution in 2D

$$(I * h)(x, y) := \int h(-x', -y') I(x + x', y + y') dx' dy'$$

- Some general properties of convolution:
  - Linearity $(\alpha f + \beta g) * h = \alpha(f * h) + \beta(g * h), \quad \alpha, \beta \in \mathbb{R}$
  - Shift invariance $f(x) * g(x + \delta) = (f * g)(x + \delta), \quad \forall \delta \in \mathbb{R}$
  - Commutativity $f * g = g * f$
  - Associativity $(f * g) * h = f * (g * h)$

- Correlation: $f(x) \star h(x) := \int h(x') f(x + x') dx'$

Discrete convolution and separability

- Discrete convolution in 1D:

$$(f * h)_i = \sum_{k=1}^{n} f_k h_{i-k} \quad i = 1, ..., N$$

- Discrete convolution in 2D:

$$(f*h)_{i,j} = \sum_{k=1,l=1}^{n,m} f_{k,l} h_{i-k,j-l} \quad i = 1, ..., N; j = 1, ..., M$$

- Separability: A filter is separable if

$$h(x, y) = \delta(x, y) * h_1(x, \cdot) * h_2(\cdot, y)$$

where δ is the Dirac distribution

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & \text{else} \end{cases}, \qquad \int \delta(x) dx = 1$$

- Separable filters can be implemented via successive 1D convolutions (associativity of convolution). This reduces the computational complexity of the convolution from $O(NMnm)$ to $O(NM(n + m))$

Gaussian filtering

- Linear filter for image smoothing
- Convolution with a Gaussian kernel of width σ

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

- The Gaussian filter is separable; the smoothed image can be derived as

$$\tilde{I}(x,y) = I(x,y) * G_\sigma(x,\cdot) * G_\sigma(\cdot,y)$$

- Discrete filter mask usually derived by sampling the continuous Gaussian in the 3σ-interval
- Computational complexity: $O(NM\sigma)$
- Alternative implementation in the Fourier domain is $O(NM\log(NM))$

Gaussian convolution

Input image

$\sigma = 1$

$\sigma = 2$

$\sigma = 4$

Boundary conditions

- The image domain Ω is generally not an infinite domain but has boundaries $\partial\Omega$.

- Different types of boundary conditions
  - **Dirichlet boundary conditions**:

    $$I(x,y) = 0, \quad (x,y) \in \partial\Omega$$

  - **Homogeneous Neumann boundary conditions**:

    $$\frac{\partial}{\partial \mathbf{n}} I(x,y) = 0 \quad (x,y) \in \partial\Omega$$

    where $\mathbf{n}$ is the outer normal vector on $\partial\Omega$ and $\frac{\partial I}{\partial \mathbf{n}} := \mathbf{n}^\top \nabla I$ is the directional derivative

- Usually Neumann boundary conditions are preferred as they have nicer properties (e.g. preservation of average intensity)

- They are obtained by **mirroring the image** at the boundary

Gaussian convolution – noise removal

Noisy input image

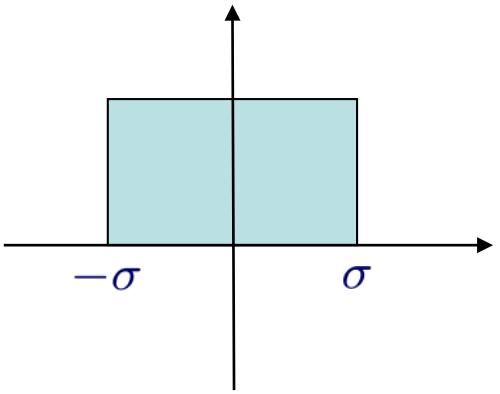
$\sigma = 1$

Satisfied?

Too bad, it removes some of the noise, but also part of the signal
Better: Nonlinear diffusion; see Computer Vision, class 2

Why a Gaussian filter and not a box filter?

a) The box filter is not separable



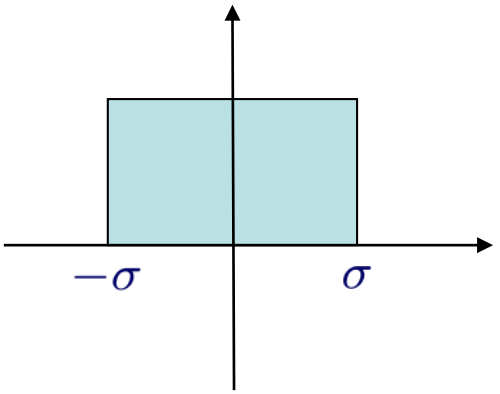
b) The Gaussian filter is rotationally invariant, the box filter is not

c) Of course we prefer a box filter!

Box filter

- Alternative to a Gaussian kernel: box kernel

$$B_\sigma(x) = \begin{cases} \frac{1}{2\sigma} & |x| < \sigma \\ 0 & \text{else} \end{cases}$$



- Simple (unweighted) averaging of neighboring values

- Disadvantages:
  - Result not as smooth (differentiability is increased only by one order)
  - Not rotationally invariant

- Advantage: convolution with a large kernel is much more efficient, since

$$\tilde{f}_{i+1} = \frac{1}{2\sigma} \sum_{j=i+1-\sigma}^{i+1+\sigma} f_j = \frac{1}{2\sigma} \left(\sum_{j=i-\sigma}^{i+\sigma} f_j + f_{i+1+\sigma} - f_{i-\sigma} \right) = \tilde{f}_i + \frac{1}{2\sigma} \left(f_{i+1+\sigma} - f_{i-\sigma} \right)$$

- Filter is separable

- Complexity $O(NM)$ independent of σ

Box filter

Gaussian filter $\sigma = 4$

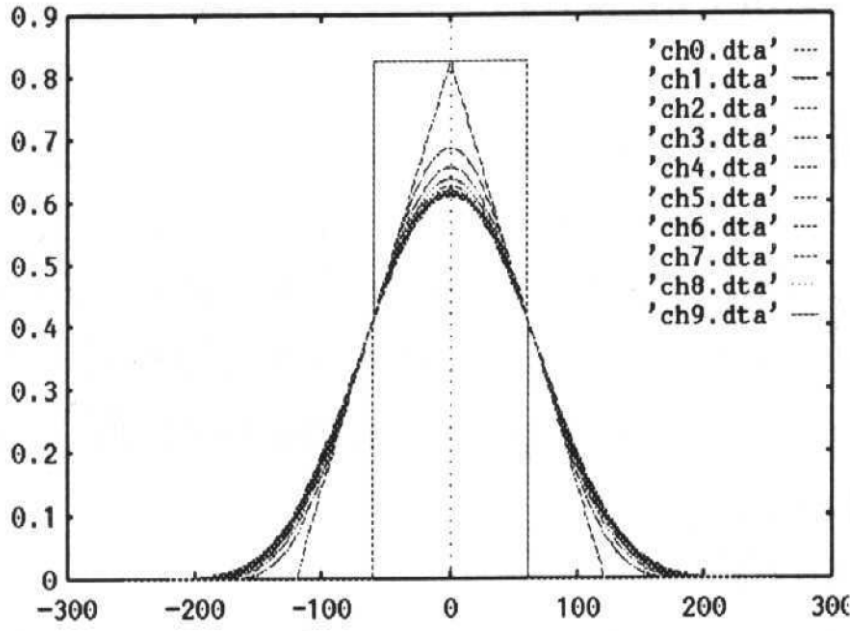
Box filter $\sigma = 6$

Relations to Gaussian filter

- Successive convolution of the box kernel with itself yields filters that resemble more and more the Gaussian kernel

- Central limit theorem in statistics: iterated averaging kernels (= positive kernels) converge to Gaussians

- Smoothing result with filter of order k is k -times differentiable



Authors: Guichard-Morel 1999

Iterated box filter

Gaussian filter $\sigma = 4$

3 iterations of a box filter

Recursive filter

- Another fast alternative to the Gaussian filter (Deriche 1990)

- Idea: recursively propagate information in both directions of the signal

α : smoothness parameter

$$f_i = \frac{\left(1 - e^{-\alpha}\right)^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} \left(I_i + e^{-\alpha}(\alpha - 1) I_{i-1}\right) + 2e^{-\alpha} f_{i-1} - e^{-2\alpha} f_{i-2}$$

$$g_i = \frac{\left(1 - e^{-\alpha}\right)^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} \left(e^{-\alpha}(\alpha + 1) I_{i+1} - e^{-2\alpha} I_{i+2}\right) + 2e^{-\alpha} g_{i+1} - e^{-2\alpha} g_{i+2}$$

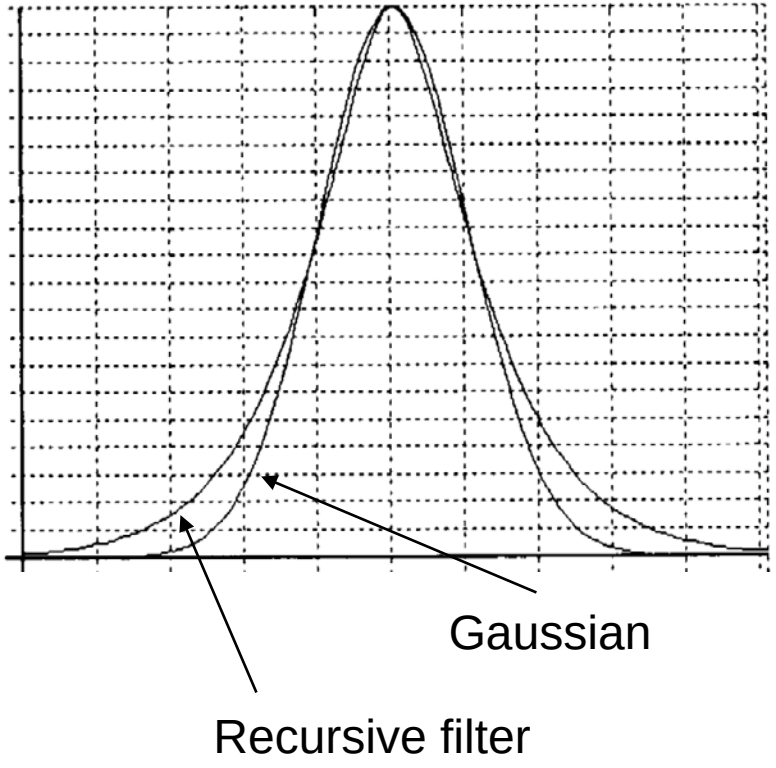
$$\tilde{I}_i = f_i + g_i$$

Recursive filter

- Recursive filter approximates Gaussian convolution
- Relation between α and σ

$$\alpha \cdot \sigma = \frac{5}{2\sqrt{\pi}}$$

- Filter is separable and rotationally invariant
- Complexity $O(NM)$ is independent of α



- Hard (maybe impossible) to implement Neumann boundary conditions

Recursive filter

Gaussian filter $\sigma = 4$

Recursive filter with $\sigma = 4$

Derivative filters

- Important aspect in image processing: measuring the **local change** of intensities (can indicate object boundaries, measure texture, etc.)

- In continuous functions, the local change is given by the derivative $\frac{df}{dx}$ of the function

- Counterpart in multidimensional functions (such as images) is the gradient

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)^\top$$

- Abbreviated notation: $\nabla I = (I_x, I_y)^\top$

- Image must be differentiable

Gaussian derivative

- Differentiability is ensured by combining the derivative with a small amount of Gaussian smoothing → **Gaussian derivatives**

- Convolution and derivatives are both linear operations
  → Applying the derivative to the image or to the Gaussian does not matter

$$\frac{\partial}{\partial x}(I(x) * G_\sigma(x)) = I(x) * \frac{\partial}{\partial x} G_\sigma(x)$$

- The Gaussian function is in $\mathcal{C}^\infty$ → arbitrarily high order derivatives exist

- Sampling the derivative of the Gaussian yields a discrete filter mask for implementation

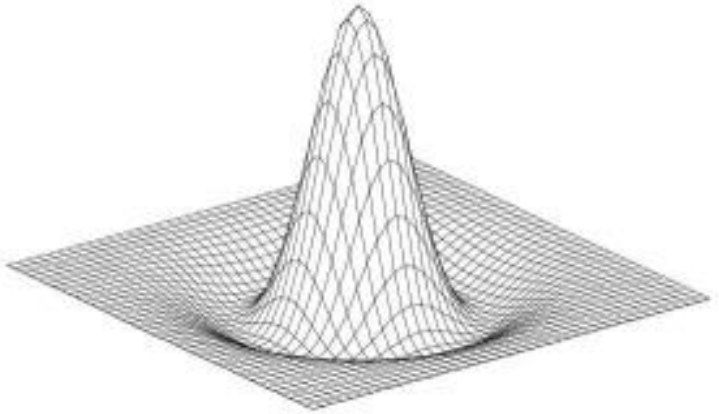
- A common mask for the first derivative is (-1/2, 0, 1/2). This is called **central difference**.

Higher order derivatives

- Sometimes we are interested in higher derivatives than the first derivative

- A popular example is the **Laplace filter**, which is based on second derivatives

$$\Delta I := \frac{\partial^2 I}{\partial^2 x} + \frac{\partial^2 I}{\partial^2 y} = I_{xx} + I_{yy}$$



- Zero-crossings of this filter correspond to image edges

- Another way to detect edges is by the gradient magnitude (based on first derivatives)

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$

Edge detection

Gradient magnitude

Laplacian
(normalized to a [0,255] range)

Summary

- There are different noise models depending on the source of noise
- The quality of an image enhancement method can be measured by the signal-to-noise ratio
- Simple point operations like gamma correction or histogram equalization can make dark structures better visible
- Difference images can (under certain conditions) detect moving objects
- An important smoothing filter is the Gaussian filter
- There are fast approximations for large amounts of smoothing, such as the iterated box filter and the recursive filter
- Image edges can be detected with derivative filters

Literature

- R. Deriche: Fast algorithms for low-level vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 12(1):78-87, 1990.

Programming assignment

- Get used to the very basic programming environment that we will use in this tutorial. Look at the file `Ex01.cpp` in `ImageProcessingEx01.zip` with any editor. Compile it using the included makefile (just type `make`) or with
  `g++ Ex01.cpp NMath.cpp –I. –o Ex01`
  and run the program via
  `./Ex01`
  This will just create a copy of the image `lena.pgm` called `lenaNoisy.pgm`. You can look at images with `display`. We also provide code for displaying images from within your running program.

- Now add Gaussian noise with standard deviation 10 and 20 to the Lena image by filling in the missing code. Use the Box-Muller method, which is described below, to simulate the noise.

  The Box-Muller method creates a Gaussian distributed random variable with $\mu = 0, \sigma = 1$ given uniformly distributed random numbers. The function to generate uniformly distributed random numbers in C/C++ is `rand()`. It generates numbers between 0 and `RAND_MAX`.

  First create two independent random variables U and V with uniform distribution in [0,1]. Then compute $N = \sqrt{-2 \ln U} \cos(2\pi V)$ $M = \sqrt{-2 \ln U} \sin(2\pi V)$
  N and M are independent Gaussian distributed random variables with $\mu = 0, \sigma = 1$

  When saving the degraded image to disk (e.g. PGM format) ensure not to leave the interval [0,255]. Clip the values.

Programming assignment

- Measure the PSNR of the noisy images
- Create a sequence of 50 noisy images. Average these images: $\bar{I} = \frac{1}{N} \sum_{i=1}^{N} I_i$
  Measure the PSNR of the averaged image. What do you find?
- Think of a general sequence of images. Why does the above trick not work with general sequences?
- Compute the difference image of `Sidenbladh.ppm` and `SidenbladhBG.ppm`.
  Color images can be handled with the class `CTensor`.
- Implement the Gaussian filter, the iterated box filter, and the recursive filter. Note that you must mirror the image at the boundaries to have Neumann boundary conditions. Test these filters on `chinaToilet.pgm`. Compare their results and computation times, particularly for large amounts of smoothing.
- What about color images? Is color a problem? Run your favorite filter on `fallingMangoes.ppm`.