Lecture 2:
Foundations II: Degradations in Digital Images

Contents

Noise

- very common in digital images
- can have many reasons, e.g.
  - image sensor of a digital camera
  - grainy photographic films that are digitised
  - specific acquisition methods:
    e.g. ultrasound imaging always creates ellipse-shaped speckle noise
  - atmospheric perturbances during wireless transmission
- our goal today: classify and simulate noise
- Simulating noise is important for the evaluation of image denoising methods, since one knows both the original and the noisy image.
- Algorithms for denoising will be discussed in later lectures.
- Noise is modelled in a stochastic way.

Noise (2)

M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

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.

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

Noise (3)

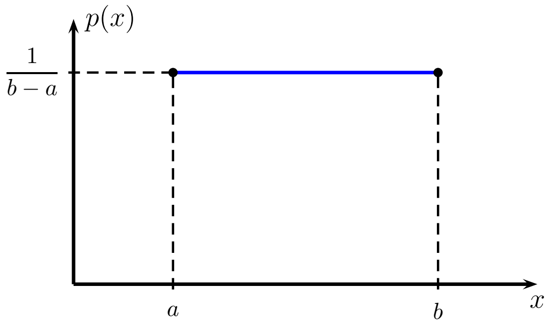
M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

Uniform Noise

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

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a,b], \\ 0 & \text{else.} \end{cases}$$

- appears e.g. in connection with quantisation (cf. Assignment H1, Problem 4)



Density function for uniform noise. Author: M. Mainberger.

Noise (4)

How Can One Simulate Uniform Noise ?

- A random variable U with uniform distribution in $[a, b]$ can be simulated in C as follows:

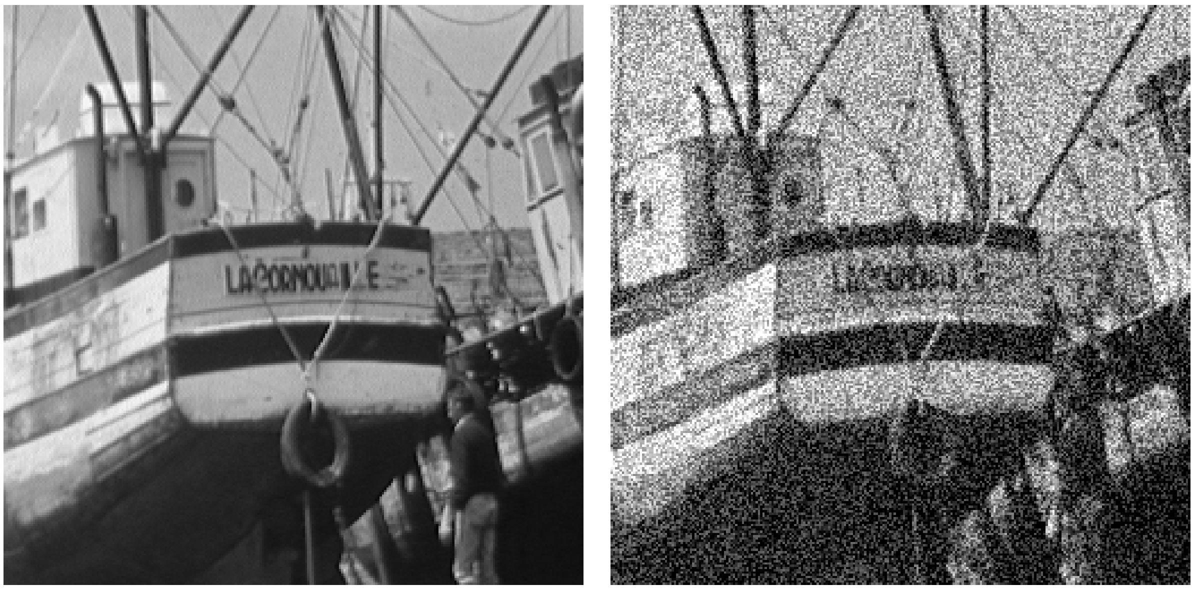
```
U = a + (float)rand() / RAND_MAX * (b - a);
```

  where `(float)rand() / RAND_MAX` creates a uniformly distributed random number in $[0, 1]$.

- Degrading some image g with uniform noise is easy:
  Replace g by $g + U$ with some uniformly distributed random variable U.

- Usually one chooses a symmetric interval w.r.t. 0, i.e. $[-b, b]$ with $b > 0$.

Noise (5)



Left: Original image, 256×256 pixels, grey value range $[0, 255]$. **Right:** After adding noise with uniform distribution in $[-70, 70]$. Resulting grey values outside $[0, 255]$ have been cropped. Author: J. Weickert.

Noise (6)

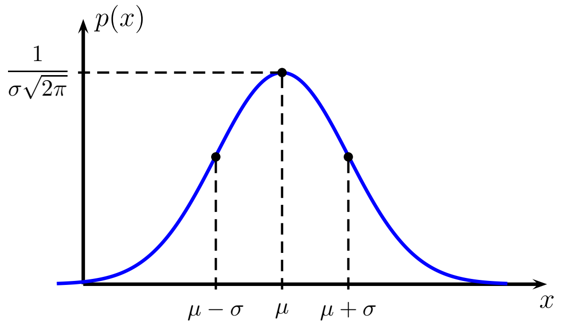
Gaussian Noise (White Noise)

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

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

  where μ is the mean and σ the standard deviation
- $p(x)$ has inflection points at $x = \mu - \sigma$ and $x = \mu + \sigma$.

Noise (7)



Density function for Gaussian noise. Author: M. Mainberger.

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

$$\begin{aligned} P(\mu-\sigma \le X \le \mu+\sigma) &\approx 68\% \\ P(\mu-2\sigma \le X \le \mu+2\sigma) &\approx 95.5\% \\ P(\mu-3\sigma \le X \le \mu+3\sigma) &\approx 99.7\% \end{aligned}$$

  Hence, Gaussian noise lives almost completely in the interval $[\mu-3\sigma,\, \mu+3\sigma]$.

Noise (8)

How Can One Simulate Gaussian Noise ?

- **Box–Muller algorithm** for creating two random variables with *normal distribution* (Gaussian distribution with $\mu = 0$, $\sigma = 1$):
  - Create independent random variables U and V with uniform distribution in [0,1]. Implementation in C:

    ```
    U = (float)rand() / RAND_MAX;
    V = (float)rand() / RAND_MAX;
    ```

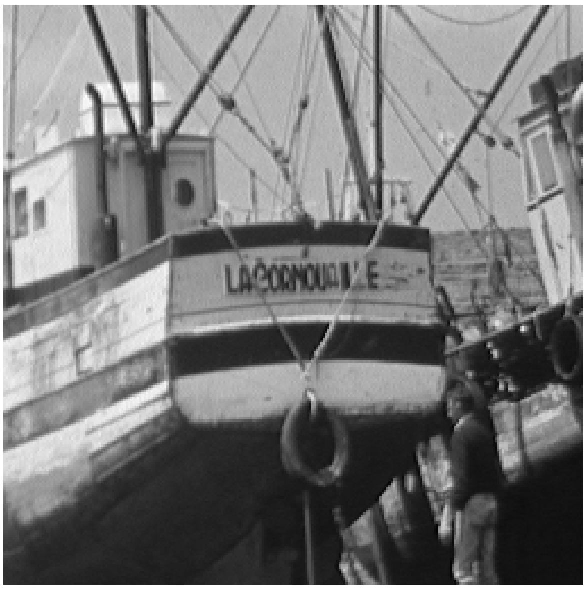
  - If $U > 0$ compute

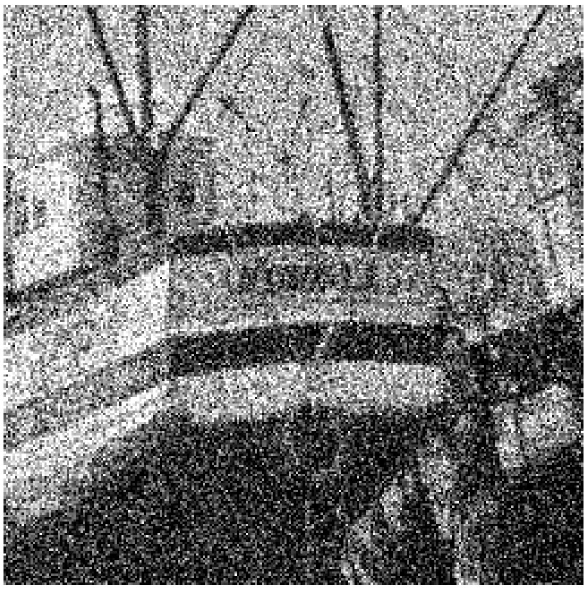
    $$\begin{aligned} N &= \sqrt{-2\ln U}\,\cos(2\pi V), \\ M &= \sqrt{-2\ln U}\,\sin(2\pi V). \end{aligned}$$

    In the tutorial your tutor will prove that N and M are independent random variables with normal distribution (cf. Assignment C1, Problem 3).

- How can one degrade a grey value f by additive Gaussian noise with mean 0 and standard deviation σ ?
  Take some random variable N with normal distribution and replace f by $f + \sigma N$.

Noise (9)





Left: Original image, 256×256 pixels, grey value range: $[0, 255]$. **Right:** After adding Gaussian noise with $\sigma = 64.48$. Grey values outside $[0, 255]$ have been cropped. Author: J. Weickert.

Noise (10)

Be Careful !

- Images with byte-wise coding have grey values in $[0, 255]$.
- By adding Gaussian noise, one may leave this interval.
- Byte-wise coding either crops these values or misinterprets them.
- In both cases no real Gaussian noise is obtained.
- Two remedies, if real Gaussian noise is required:
  - Store the noisy image in floating point precision.
  - Alternatively, avoid storing the noisy image, and add Gaussian noise during testing.
- Similar considerations apply for uniform noise.

Noise (11)

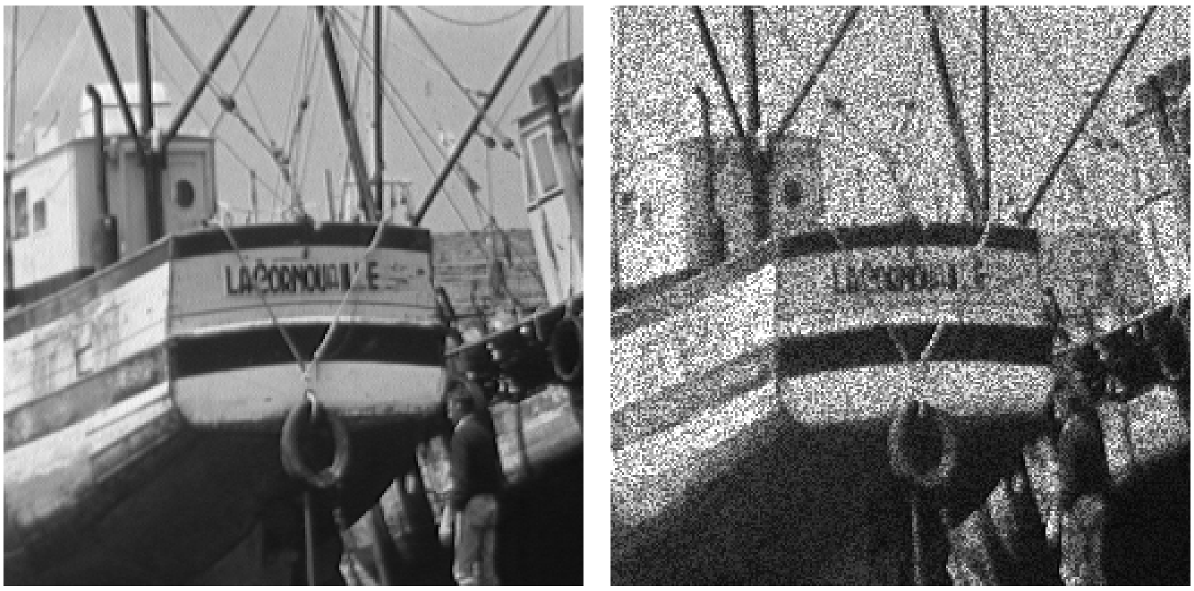
Multiplicative Noise

- Multiplicative noise is signal-dependent (unlike additive noise).
- Usually one assumes that it is proportional to the grey value:

$$\begin{aligned} f_{i,j} &= g_{i,j} + n_{i,j}\, g_{i,j} \\ &= (1 + n_{i,j})\, g_{i,j} \end{aligned}$$

- Example:
  Noise caused by the grains of a photographic emulsion in analog photos.

Noise (12)



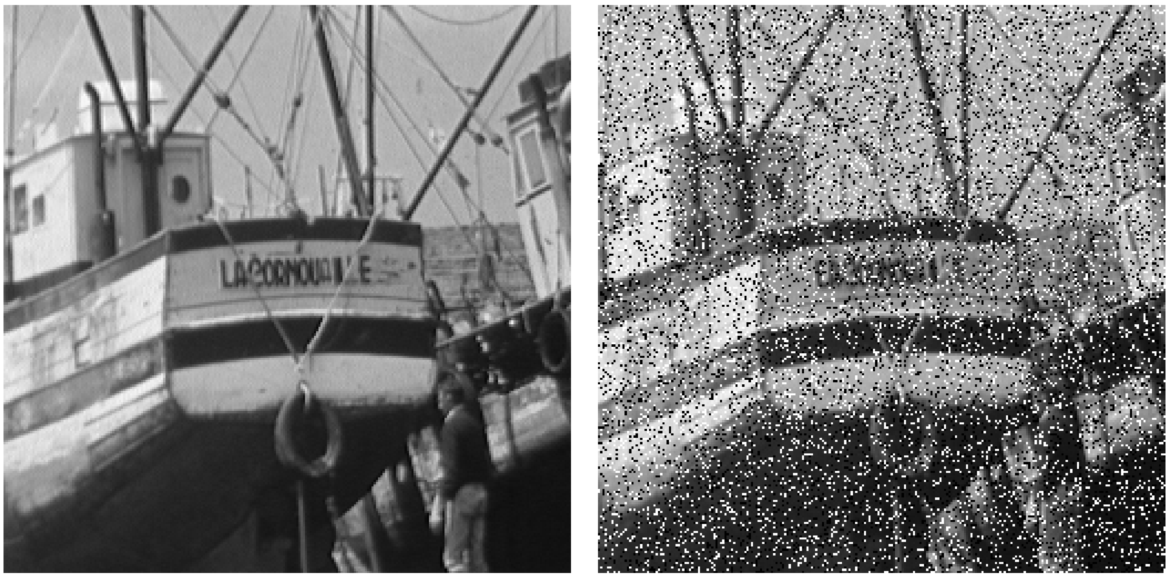
Left: Original image, 256×256 pixels, grey value range: $[0, 255]$. **Right:** After applying multiplicative noise where n has uniform distribution in $[-0.5, 0.5]$. Resulting grey values outside $[0, 255]$ have been cropped. Note that darker grey values are less affected by noise than brighter ones. Author: J. Weickert.

Noise (13)

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 the sensor chip of a digital camera
- *Unipolar* impulse noise gives degradations having only one grey value. *Bipolar* noise attains two grey values.
- Bipolar noise with the highest and lowest grey values is called *salt-and-pepper noise.*

Noise (14)



Left: Original image, 256×256 pixels. **Right:** 20 % of all pixels have been degraded by salt-and-pepper noise, where bright and dark values have equal probability. Author: J. Weickert.

Noise (15)

Measuring Noise

- Let $g = (g_{i,j})$ be the undegraded image (without noise) with $M \times N$ pixels.
- Its *mean (average grey value, Mittelwert)* is given by

$$\mu(g) \; := \; \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} g_{i,j} \, .$$

- The *variance (Varianz)* measures the average quadratic deviation from the mean:

$$\sigma^2(g) \; := \; \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (g_{i,j} - \mu)^2 \, .$$

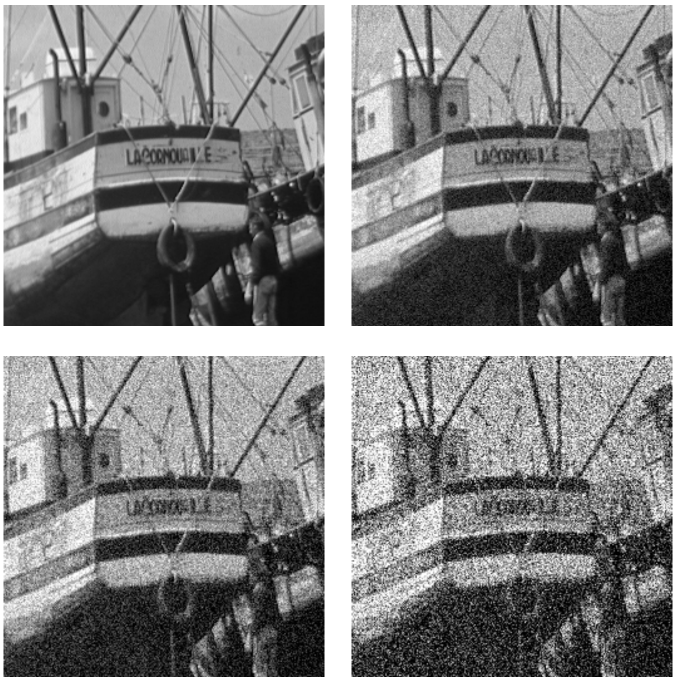
- Its square root $\sigma(g)$ is called *standard deviation (Standardabweichung, Streuung).*

Noise (16)

- Let $f = (f_{i,j})$ be a noisy version of g.
  The *mean squared error (MSE, mittlerer quadratischer Fehler)* between f and g is given by
  $$\mathrm{MSE}\,(f,g) \;:=\; \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{i,j}-g_{i,j})^2.$$
  The smaller, the better.

- In image compression one often uses the *peak-signal-to-noise ratio (PSNR, Spitzen-Signal-Rausch-Abstand).*
  For a grey value range $[0, 255]$, the PSNR is defined as
  $$\mathrm{PSNR}\,(f,g) \;:=\; 10\,\log_{10}\left(\frac{255^2}{\mathrm{MSE}\,(f,g)}\right)$$
  Its unit is *decibel (dB)*. The higher the better.
  Noise with PSNR values ≥ 30 dB is hardly visible.

Noise (17)



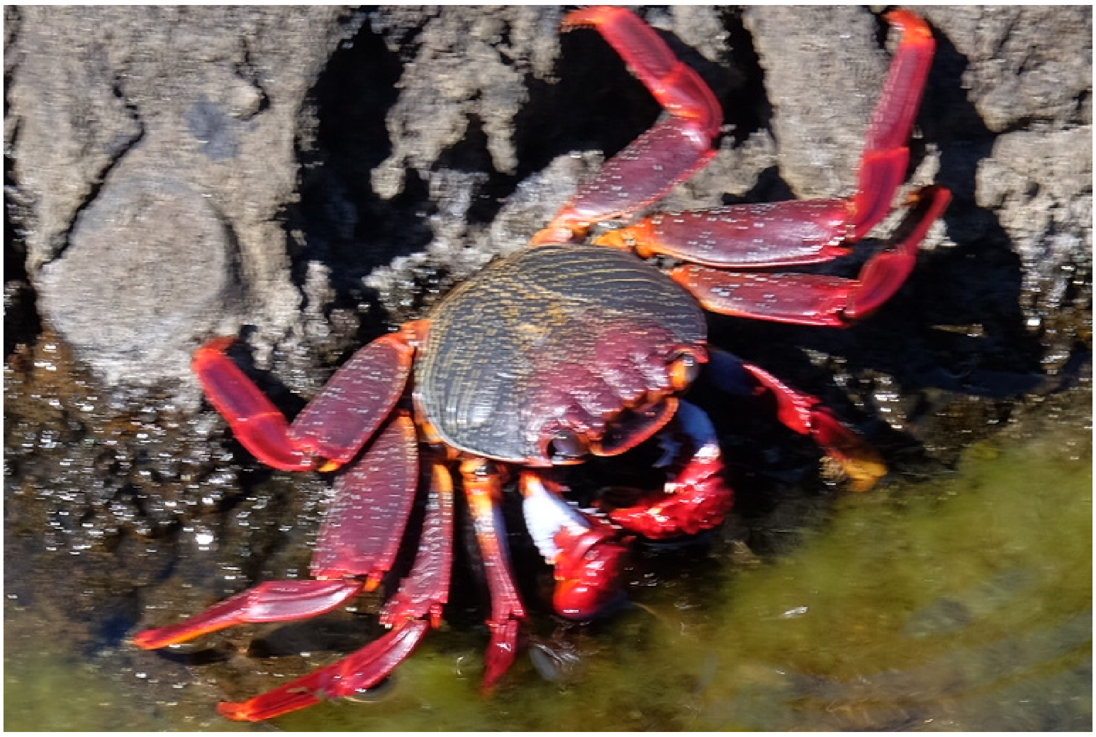
Top left: Original image, 256×256 pixels. **Top right:** Adding Gaussian noise with $\sigma = 15$ gives $\mathrm{MSE} = 226.06$ and $\mathrm{PSNR} = 24.59$ dB. **Bottom left:** $\sigma = 30$ yields $\mathrm{MSE} = 904.24$ and $\mathrm{PSNR} = 18.57$ dB. **Bottom right:** $\sigma = 60$ yields $\mathrm{MSE} = 3616.95$ and $\mathrm{PSNR} = 12.55$ dB. Grey values outside $[0, 255]$ are cropped for visualisation. Author: J. Weickert.

Blur (1)

Blur (Unschärfe)

- second source of image degradations (besides noise)
- caused e.g. by defocussing, imperfections of the optical system, atmospheric disturbances, or motion during image acquisition.
- For simplicity, let us assume that the blurring effect is identical at all locations *(shift-invariant blur model).*
- can be regarded as weighted averaging of grey values within a certain neighbourhood
- weights for averaging and shape of neighbourhood depend on the source of degradation
- The mathematical tool to describe such a weighted averaging is the so-called convolution.

Blur (2)


A real-world example for motion blur: Zoom into a photo of a crab (grapsus adscenionis, Rote Felsenkrabbe) that has been degraded by camera motion. Image size: 768×512 pixels. Can you recognise the direction of the motion? Photo: J. Weickert.

Blur (3)

Convolution (Faltung)

One-Dimensional Convolution

- *discrete convolution* of two 1-D signals $g = (g_i)_{i\in\mathbb{Z}}$ and $w = (w_i)_{i\in\mathbb{Z}}$:

$$(g * w)_i \;:=\; \sum_{k\in\mathbb{Z}} g_{i-k}\, w_k .$$

- The components of w can be regarded as (mirrored) weights for averaging the components of g.

- *continuous convolution* of two 1-D signals $g, w : \mathbb{R} \to \mathbb{R}$:

$$(g * w)(x) \;:=\; \int_{\mathbb{R}} g(x-x')\, w(x')\, dx' .$$

Blur (4)

Example

- Let g_i be a stock market price (Börsenkurs) at day i.

- We want to compute the average price f_i within the last 200 days:

$$f_i \;=\; \frac{1}{200} \sum_{k=0}^{199} g_{i-k} \, .$$

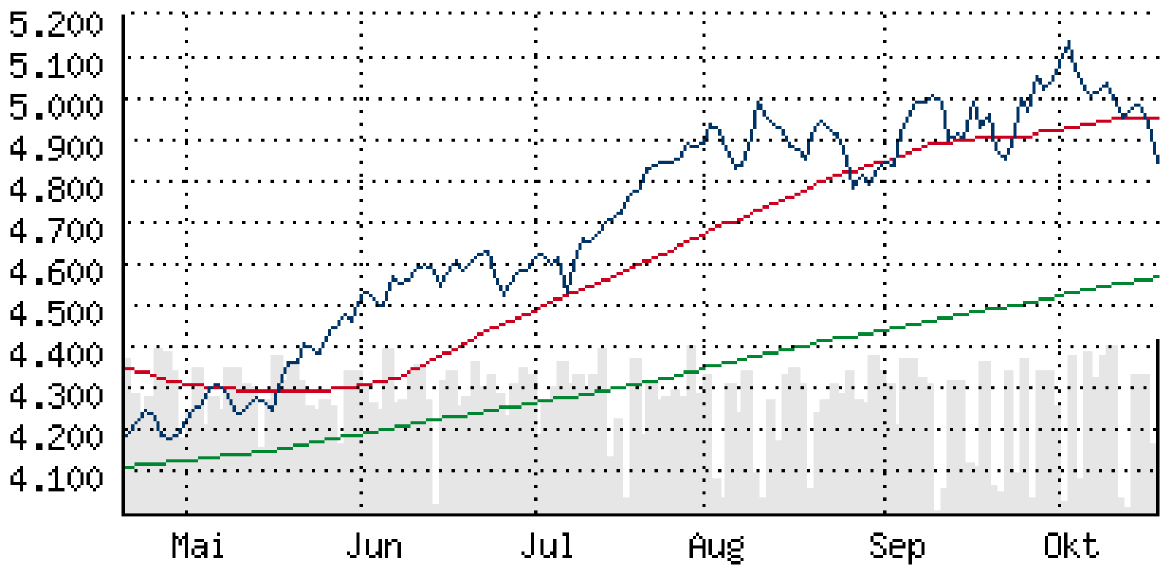
- This is as a discrete convolution between g and a suitable "blur kernel" w:

$$f_i \;=\; \sum_{k\in\mathbb{Z}} g_{i-k}\, w_k$$

with

$$w_k := \begin{cases} \frac{1}{200} & \text{for } k \in \{0, ..., 199\}, \\ 0 & \text{else.} \end{cases}$$

Blur (5)



German stock market index (DAX) on October 20, 2005. **Blue:** Daily values. **Red:** Averaged over the last 38 days. **Green:** Averaged over the last 200 days. Source: `http://www.spiegel.de`.

Blur (6)

Properties of the Convolution

For the continuous convolution, the following properties are not difficult to show (cf. Assignment H1, Problem 3), and they make life easier:

- Commutativity: $f * g \;=\; g * f$.
- Associativity: $(f * g) * h \;=\; f * (g * h)$.
- Distributivity: $(f + g) * h \;=\; f * h \;+\; g * h$,
  $f * (g + h) \;=\; f * g \;+\; f * h$.
- Differentiation: $(f * g)' \;=\; f' * g \;=\; f * g'$.
- Differentiability: If $f \in C^0(\mathbb{R})$ and $g \in C^n(\mathbb{R})$, then $(f * g) \in C^n(\mathbb{R})$.
- Linearity: $(\alpha f + \beta g) * h \;=\; \alpha(f * h) + \beta(g * h)$ with $\alpha, \beta \in \mathbb{R}$.
- Shift Invariance: $(T_b f) * g \;=\; T_b(f * g)$ for all translations T_b with
  $(T_b f)(x) \;:=\; f(x - b)$.

These properties also hold for discrete convolution
(apart from the purely continuous properties “Differentiation” and “Differentiability”).

Blur (7)

Two-Dimensional Convolution

- *discrete convolution* of two images $g = (g_{i,j})_{i,j\in\mathbb{Z}}$ and $w = (w_{i,j})_{i,j\in\mathbb{Z}}$:

$$(g * w)_{i,j} \;:=\; \sum_{k\in\mathbb{Z}} \sum_{\ell\in\mathbb{Z}} g_{i-k,\, j-\ell}\, w_{k,\ell}$$

- *continuous convolution* of two images $g, w : \mathbb{R}^2 \to \mathbb{R}$:

$$(g * w)(x,y) \;:=\; \int_{\mathbb{R}} \int_{\mathbb{R}} g(x-x', y-y')\, w(x',y')\, dx'\, dy'.$$

  The double integral can be computed first with respect to x' and then with respect to y' (Fubini's theorem).

Blur (8)

Example

- Let the convolution kernel $w(x,y)$ be given by

$$w(x,y) \;:=\; \begin{cases} \frac{1}{\pi r^2} & \text{for } x^2+y^2 \le r^2, \\ 0 & \text{else,} \end{cases}$$

- Then $g * w$ describes a smoothing of the image g by averaging all grey values within a disk-shaped neighbourhood of radius r.

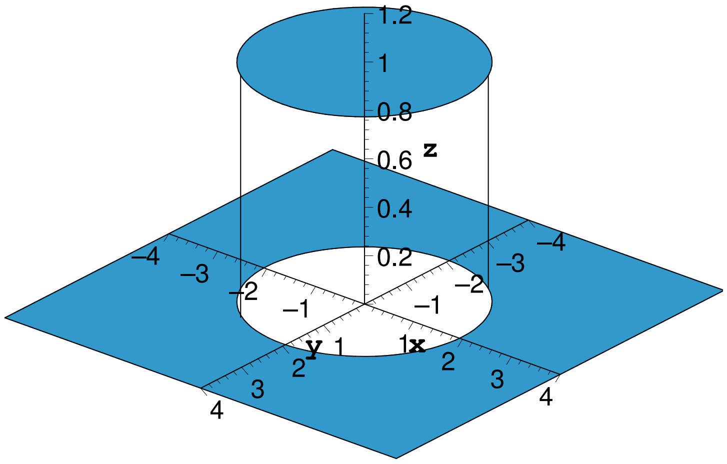
- Remark:
  Computing this convolution by calculating the integral becomes time-consuming when r is large.
  We will soon study a more efficient alternative: the Fourier transform.

Blur (9)

Modelling Blur by Convolutions

- **Defocussed Optical System:**

  modelled by a cylinder–shaped convolution kernel (“pillbox kernel”)



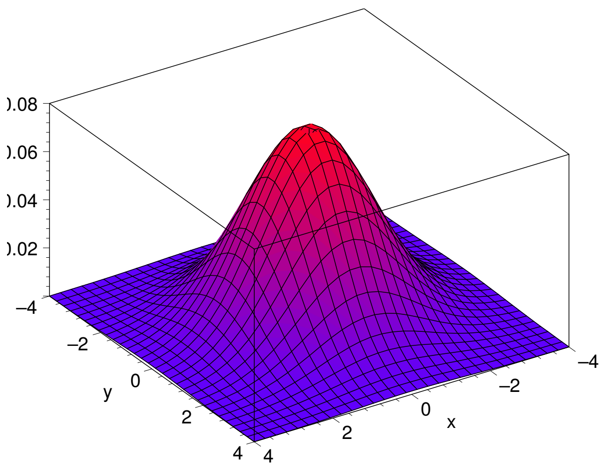
Cylinder-shaped convolution kernel. Author: B. Burgeth.

Blur (10)

- **Atmospheric Perturbations (e.g. Telescopes):**

  can be approximated by a 2-D Gaussian (product of two 1-D Gaussians):

$$w(x,y) \;=\; \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x-\mu_1)^2-(y-\mu_2)^2}{2\sigma^2}\right)$$



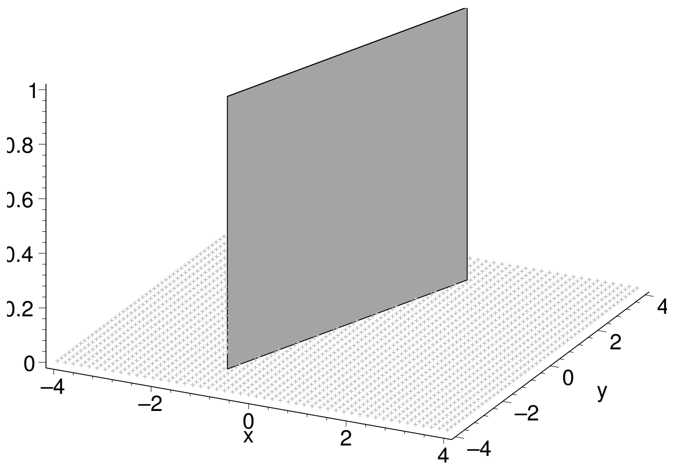
2-D Gaussian. Author: B. Burgeth.

Blur (11)

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

- **Motion Blur:**

  in the simplest case (uniform motion, all objects in equal distance to camera):
  1-D box function along the direction of motion

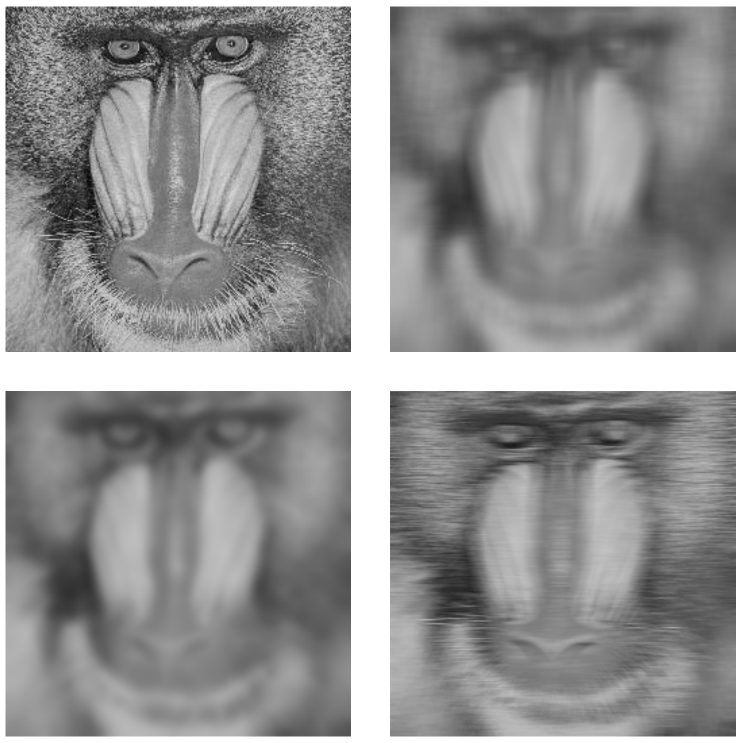


Kernel for a convolution with a 1-D box function. Author: B. Burgeth.

Blur (12)

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

original image | out-of-focus blur



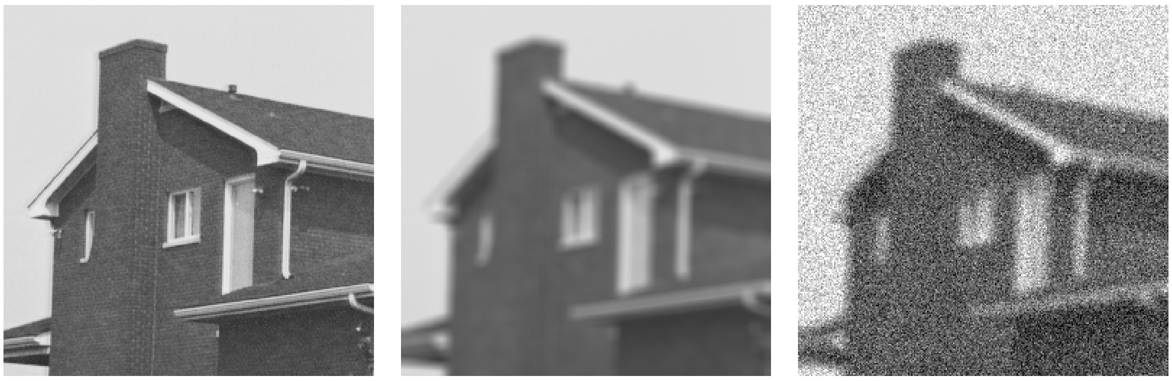
Gaussian blur | horizontal motion blur

Simulation of different types of blur.

Combined Blur and Noise

- Often digital images suffer from both blur and noise.
- A typical degradation model combines shift invariant blur with additive noise:

$$f = g * w + n.$$



Left: Original image, 256×256 pixels. **Middle:** Blurred with a pillbox kernel of radius 5 pixels. **Right:** Further degradation by additive Gaussian noise with $\sigma = 30$. Values outside $[0, 255]$ are cropped. The result simulates a digital camera that is out of focus and suffers from sensor noise. Author: J. Weickert.

Summary

- Noise and blur are frequent degradations in digital images.
- Noise can be modelled and simulated in a stochastic way.
  Most important is additive Gaussian noise with zero mean.
  It can be simulated with the Box–Muller algorithm.
  Sometimes also multiplicative and impulse noise is present.
- Shift-invariant blur can be simulated by convolution with a suitable kernel (e.g. pillbox kernel, Gaussian, 1-D box function).
- A frequently used degradation model of some initial image g with a shift-invariant blurring kernel w and additive noise n is given by

$$f = g * w + n.$$

References

- R. C. Gonzalez, R. E. Woods: *Digital Image Processing*. Prentice Hall, Upper Saddle River, International Edition, 2017.
  (Chapter 5 deals with noise and blur models)
- G. E. P. Box, M. E. Muller: A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, Vol. 29, 610–611, 1958.
  (original paper on the Box–Muller method)
- C. Boncelet: Image noise models. In A. Bovik (Ed.): *Handbook of Image and Video Processing*. Academic Press, San Diego, pp. 325–335, 2000.
  (good description of noise models for images)
- H. Tijms: *Understanding Probability*. Cambridge University Press, 2004.
  (elementary introduction to the probabilistic concepts used in this lecture)

Assignment C1 – Classroom Work

Problem 1 (Discrete Convolution)

Consider the discrete signal $\boldsymbol{f} = (f_i)_{i \in \mathbb{Z}}$ defined as

$$f_i = \begin{cases} \frac{1}{2} & (i \in \{0, 1\}), \\ 0 & (\text{else}) \end{cases}$$

and convolve it three times with itself.

(This problem will be useful for approximating discrete Gaussian convolution.)

Problem 2 (Continuous Convolution)

Consider the following two functions.

$$H(x) := \begin{cases} 0 & (x < 0), \\ 1 & (x \geqslant 0) \end{cases} \quad \text{and} \quad K(x) := \begin{cases} 1 + x & (-1 \leqslant x < 0), \\ 1 - x & (0 \leqslant x \leqslant 1), \\ 0 & (\text{else}). \end{cases}$$

Sketch both functions as well as their convolution $(K * H)(x)$.
What effect did the convolution with kernel K have on the function H?

(This shows the effect of a convolution with a smoothing kernel.)

Assignment C1 (2)

Problem 3 (Box-Muller Algorithm)

Show that the random variables created by the Box-Muller algorithm obey a normal distribution. You may use the following guidelines to construct a proof step by step.

(i) Let X and Y be two independent random variables with normal distribution (Gaussian distribution with $\mu = 0$ and $\sigma = 1$). Write down the density function $p_{(X,Y)}(x, y)$ for the 2-D normal distribution. Argue why only the length of the vector $(x, y)^\top$ is relevant for $p_{(X,Y)}$.

(ii) Let R and Q be given by the transformation of X and Y into polar coordinates, i.e. $(R, Q) := (g_1^{-1}(X, Y), g_2^{-1}(X, Y))$ with $\boldsymbol{g}(r, \varphi) = (r \cos \varphi, r \sin \varphi)^\top$, $r \in [0, \infty)$ and $\varphi \in [0, 2\pi)$. Show that the densities of R and Q have the form $p_R(r) = r \exp\left(-\frac{r^2}{2}\right)$ and $p_Q(\varphi) = \frac{1}{2\pi}$.

(iii) Use the transformation/inversion rule and two variables U and V with uniform distribution on $[0, 1]$ to derive the Box-Muller algorithm from your previous results.

Assignment C1 (3)

You can use the following results from probability theory without proof:

- The (cumulative) distribution function of a random variable X with density function $p_X(x)$ is defined as
$$F_X(x) := P\,(X \leqslant x) := \int_{-\infty}^{x} p_X(t)\; dt\,.$$
- For two independent random variables X and Y, it holds $p_{(X,Y)}(x, y) = p_X(x)\, p_Y(y)$.
- The probability densities of (X, Y) and (R, Q) are linked by the relation $p_{(R,Q)} = |\det(\boldsymbol{J_g})| \cdot p_{(X,Y)}$, where $\boldsymbol{J_g}$ is the Jacobian of $\boldsymbol{g}$.
- Transformation/Inversion Rule: Let F be a continuous distribution function on $\mathbb{R}$ with inverse F^{-1}. If U is a uniform $[0, 1]$-distributed random variable, then the random variable $X = F^{-1}(U)$ has the distribution function F. Conversely, if X has the distribution function F, then $F(X)$ is uniformly distributed on $[0, 1]$.

(Since Gaussian noise is the most frequent noise in image processing, it is important to understand how such noise can be created.)

Assignment H1 – Homework

Problem 1 (Peak-Signal-to-Noise Ratio) (1+2+1 = 4 points)

Let the image $f = (f_{i,j})$ be a noisy version of $g = (g_{i,j})$ degraded by additive noise $n = (n_{i,j})$ with zero mean:

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

In the lecture the peak-signal-to-noise ratio (PSNR) is defined as a measure of quality.

(a) In which case do we have PSNR$(f, g) = 0$?
Characterise this case mathematically with respect to the grey values of the images.

(b) Let a filtered version u of f be given such that

$$\text{PSNR}(u, g) = \text{PSNR}(f, g) + 30\,\text{dB} \,.$$

How has the MSE of f and g changed during the filtering?

(c) Assume that $n \to 0$. Calculate PSNR(f, g). How can you interpret this?

(*This assignment gives you deeper insights into the meaning of values obtained by the PSNR.)*

Problem 2 (Continuous Convolution) (2+2+2 = 6 points)

(a) Consider the continuous signal

$$f(x) := \begin{cases} \frac{1}{2} & (-1 \le x \le 1), \\ 0 & (\text{else}) \end{cases}$$

and convolve it two times with itself. You will obtain two functions

$$h_1 := f * f, \qquad h_2 := f * f * f.$$

(b) Evaluate h_1 at the points $-2, -1, 0, 1$ and 2, and evaluate h_2 at $-3, -2, -1, 0, 1, 2$ and 3.

(c) Compare these results with the discrete convolutions from Classroom Assignment C1, Problem 1.
What similarities and differences do you notice?

(This problem will be useful for approximating continuous Gaussian convolution.)

Assignment H1 (3)

Problem 3 (Properties of the Convolution) (1+1+1+2+1+1 = 7 points)

Consider the three real-valued functions f, g, and h. Show that the continuous convolution in 1-D satisfies the following properties.

Commutativity:	$f * g = g * f$.
Associativity:	$(f * g) * h = f * (g * h)$.
Distributivity:	$(f + g) * h = f * h + g * h$ and $f * (g + h) = f * g + f * h$.
Differentiability:	If $f \in C^0(\mathbb{R})$ and $g \in C^n(\mathbb{R})$, then $(f * g) \in C^n(\mathbb{R})$. *Hint: Show first that we have* $(f * g)' = f * g'$.
Linearity:	$(\alpha f + \beta g) * h = \alpha(f * h) + \beta(g * h)$ with $\alpha, \beta \in \mathbb{R}$.
Shift Invariance:	$(T_b f) * g = T_b(f * g)$ for all translations T_b with $(T_b f)(x) := f(x - b)$.

(Since convolution is the central model for blur and will also be of fundamental importance in linear system theory, it is very useful to know these rules.)

Assignment H1 (4)

Problem 4 (Noise, Quantisation, and Dithering) (2+2+3 = 7 points)

Please download the required file `Ex01.tar.gz` from the webpage

```
http://www.mia.uni-saarland.de/Teaching/ipcv19.shtml
```

into your own directory. To unpack the data use the command `tar xvzf Ex01.tar.gz`.

Usually the grey values in an image are represented by $2^8 = 256$ different grey values. In this problem, we want to quantise the image `boat.pgm` such that it contains afterwards only 2^q different grey values, with $q < 8$.

(a) In order to reduce the number of grey values we subdivide the co-domain into uniform intervals of size $d = 2^{8-q}$. All values that lie in an interval are then mapped to the mean value of the interval:

$$u_{i,j} := \left(\left\lfloor \frac{u_{i,j}}{d} \right\rfloor + \frac{1}{2} \right) \cdot d$$

where $\lfloor . \rfloor$ denotes the floor function, i.e. $\lfloor x \rfloor$ is the largest integer number that does not exceed x. Consider the file `quantisation.c` and supplement the missing code such that it performs the quantisation described above. Compile the program with

```
gcc -O2 -o quantisation quantisation.c -lm
```

and apply it to the image `boat.pgm` using $q = 1,\ 3$, and 6.

Remark: You can display an image `img.pgm` with the command `xv img.pgm`.

Assignment H1 (5)

(b) Integrate uniformly distributed noise $n_{i,j}$ with zero mean and maximal deviation of 0.5 into the quantisation process:

$$u_{i,j} := \left(\left\lfloor \frac{u_{i,j}}{d} + n_{i,j} \right\rfloor + \frac{1}{2} \right) \cdot d \; ,$$

Use the same choices for q that you tried in (a). How do the results change?

(c) To improve the visual quality when quantising images one can use *Floyd-Steinberg dithering*, a special type of so-called *error diffusion*. It distributes the quantisation error of a pixel to its unprocessed neighbouring pixels as follows:

For each pixel (i, j) from left to right and top to bottom:

1. Quantise $u_{i,j}$ according to (a).
2. Compute the error e between the original value and the quantised value.
3. Distribute the error e to the neighbouring pixels that are not visited yet as follows:
   - $u_{i+1,j} := u_{i+1,j} + \frac{7}{16}e$
   - $u_{i-1,j+1} := u_{i-1,j+1} + \frac{3}{16}e$
   - $u_{i,j+1} := u_{i,j+1} + \frac{5}{16}e$
   - $u_{i+1,j+1} := u_{i+1,j+1} + \frac{1}{16}e$

Supplement the missing code in `floyd-steinberg.c` and compile the program with

```
gcc -O2 -o floyd-steinberg floyd-steinberg.c -lm .
```

Apply it to the image `boat.pgm` and compare your result with the ones obtained in (a) and (b). What are your findings?

Assignment H1 (6)

Submission

Please remember that up to three people from the same tutorial group can work and submit their results together. The theoretical Problems 1, 2, and 3 have to be submitted in handwritten form into the mailbox of your tutorial group **before** the lecture. The mailboxes can be found in Building E1.3 in the corner next to Lecture Hall 001. For the practical Problem 4 please submit files as follows: Rename the main directory `Ex01` to `Ex01_<your_name>` and use the command

```
tar czvf Ex01_<your_name>.tar.gz Ex01_<your_name>
```

to pack the data. The directory that you pack and submit should contain the following files:

- the supplemented source code used in Problem 4.
- one image for each Subproblem 4(a)–(c) using $q = 3$.
- a text file README that contains the answers to the questions of Subproblem 4(b) and 4(c), as well as information on all people working together for this assignment.

Please make sure that only your final version of the programs and images are included. Please do **not** submit any additional files, especially no executables. Submit the file via e-mail to your tutor via the address:

```
ipcv-xx@mia.uni-saarland.de
```

where `xx` is either `t1`, `t2`, `t3`, `t4`, `t5`, `w1`, `w2`, `w3`, or `w4`, depending on your tutorial group.

Deadline for submission: Friday, April 19, 10 am (before the lecture)