# Lecture 16:
# Nonlinear Filters III:
# Nonlinear Diffusion Filtering



**Contents**



© 2000–2019 Joachim Weickert





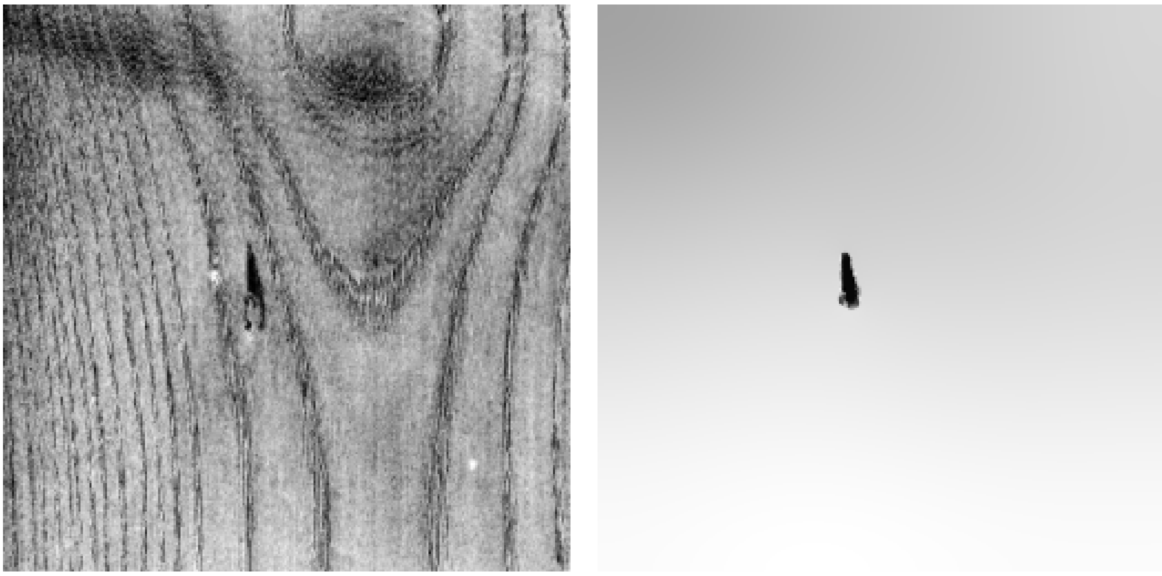
# Motivation

## Nonlinear Diffusion Filters

- class of denoising methods that offer many advantages:
  - smooth within a region and preserve edges (in contrast to linear shift-invariant filters)
  - preserve the average grey value (in contrast to morphological methods, bilateral filters, and NL-means)
  - are shift-invariant and do not suffer from over- and undershoots (in contrast to classical wavelet shrinkage)
  - often give good results
  - can be designed in a flexible way
- have also some limitations:
  - somewhat slower than fast filters such as wavelet shrinkage
  - mathematically a bit more demanding: require differential equations

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

## Three Examples Demonstrating the Flexibility of Diffusion Filtering

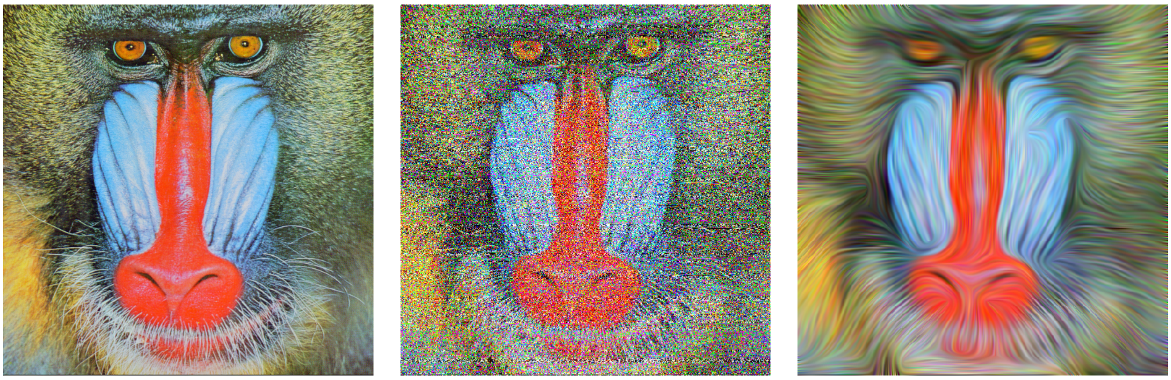


Defect detection in a wood surface. **Left:** Wood surface, $256 \times 256$ pixels. **Right:** After nonlinear diffusion filtering (model of Catté et al.). Author: J. Weickert.



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

Fingerprint enhancement. **Left:** Fingerprint image, $300 \times 300$ pixels. **Right:** Filtered with an advanced nonlinear diffusion method (coherence-enhancing anisotropic diffusion). Author: S. Grewenig et al.



Robustness of vector-valued diffusion filtering with respect to noise. **Left:** Mandrill, $512 \times 512$ pixels. **Middle:** With additive Gaussian noise. Noise variance per channel: four times the signal variance. **Right:** Coherence-enhancing anisotropic diffusion filtering. Author: J. Weickert.





# Mathematical Background

## What is a Differential Equation?

- *Algebraic equations* state relations between an unknown *number* and its powers.

  Example:
  $$x^2 - 8x + 15 = 0.$$

  Two solutions: $x_1 = 3$ and $x_2 = 5$.

- *Differential equations* state relations between an unknown *function* and its derivatives.

  Example:
  $$\frac{du(t)}{dt} = 5\,u(t).$$

  Infinitely many solutions: $u(t) = a\,e^{5t}$ with arbitrary $a \in \mathbb{R}$.

  An additional *initial condition* such as $u(0) = 2$ can make the solution unique:
  $$u(t) = 2\,e^{5t}.$$

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

## Mathematical Background (2)



### Ordinary and Partial Differential Equations

- If the unknown function $u$ depends only on a single variable (e.g. $u = u(t)$):
  - Only ordinary derivatives appear.
  - yields an *ordinary differential equation (ODE, gewöhnliche Differentialgleichung).*
  - The previous equation $\frac{du(t)}{dt} = 5\,u(t)$ is an ODE.
- If $u$ is a function of multiple variables (e.g. $u = u(x,t)$):
  - Partial derivatives such as $\frac{\partial u}{\partial x}$ or $\frac{\partial u}{\partial t}$ may occur.
  - This gives a *partial differential equation (PDE, partielle Differentialgleichung).*

## Mathematical Background (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

### Example of a PDE

- Consider the *linear one-dimensional diffusion equation*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

  for some unknown function $u(x,t)$.
  Assume that at time $t = 0$, the solution coincides with a known function $f(x)$:

$$u(x,0) = f(x).$$

- It can be shown that this problem has a unique solution.
  It is given by convolution with a Gaussian $K_\sigma$ with standard deviation $\sigma = \sqrt{2t}$:

$$u(x,t) = (K_{\sqrt{2t}} * f)(x).$$

- Such nice analytical solutions only exist for a few simple PDEs.
  Usually numerical approximations are required.

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

# Physical Background: What is Diffusion?

- Diffusion equilibrates concentration differences by redistributing mass.
  It preserves the total mass.

- (Isotropic) Diffusion processes are described by the PDE

$$\partial_t u \;=\; \operatorname{div}(g \cdot \boldsymbol{\nabla} u)$$

  where the divergence and the nabla operator involve the spatial derivatives only, not the temporal ones.

- The longer the diffusion time $t$, the more the concentration differences are equilibrated.

- The diffusivity function $g$ may depend on the location:
  At locations where $g$ is larger, the diffusion proceeds faster.



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

# A Continuous Diffusion Filter

## Diffusion in Image Processing

| | |
|---|---|
| grey values: | are interpreted as concentrations |
| image domain: | rectangular domain $\Omega := (0, a_1) \times (0, a_2)$ |
| image: | bounded function $f : \Omega \to \mathbb{R}$ |

## Diffusion Filter

Computes a filtered version $u(x, y, t)$ of $f(x, y)$ as solution of the diffusion equation

$$\partial_t u = \operatorname{div}(g \; \boldsymbol{\nabla} u)$$

with the original image as initial condition,

$$u(x, y, 0) = f(x, y),$$

and reflecting boundary conditions
($\boldsymbol{n}$: unit normal vector at image boundary $\partial\Omega$, and $\partial_{\boldsymbol{n}} u := \boldsymbol{n}^\top \boldsymbol{\nabla} u$):

$$\partial_{\boldsymbol{n}} u = 0.$$

## A Continuous Diffusion Filter (2)

### How is the Diffusivity $g$ Chosen ?

- We want to reduce the diffusion at edges in order to preserve them.
- $|\boldsymbol{\nabla} u| = \sqrt{(\partial_x u)^2 + (\partial_y u)^2}$ is an indicator for edges:
  Edges are locations where $|\boldsymbol{\nabla} u|$ is large (cf. Lecture 13).
- Choose a diffusivity that is decreasing in $|\boldsymbol{\nabla} u|$, e.g. the *Charbonnier diffusivity*

$$g(|\boldsymbol{\nabla} u|) \;:=\; \frac{1}{\sqrt{1 + |\boldsymbol{\nabla} u|^2/\lambda^2}}$$

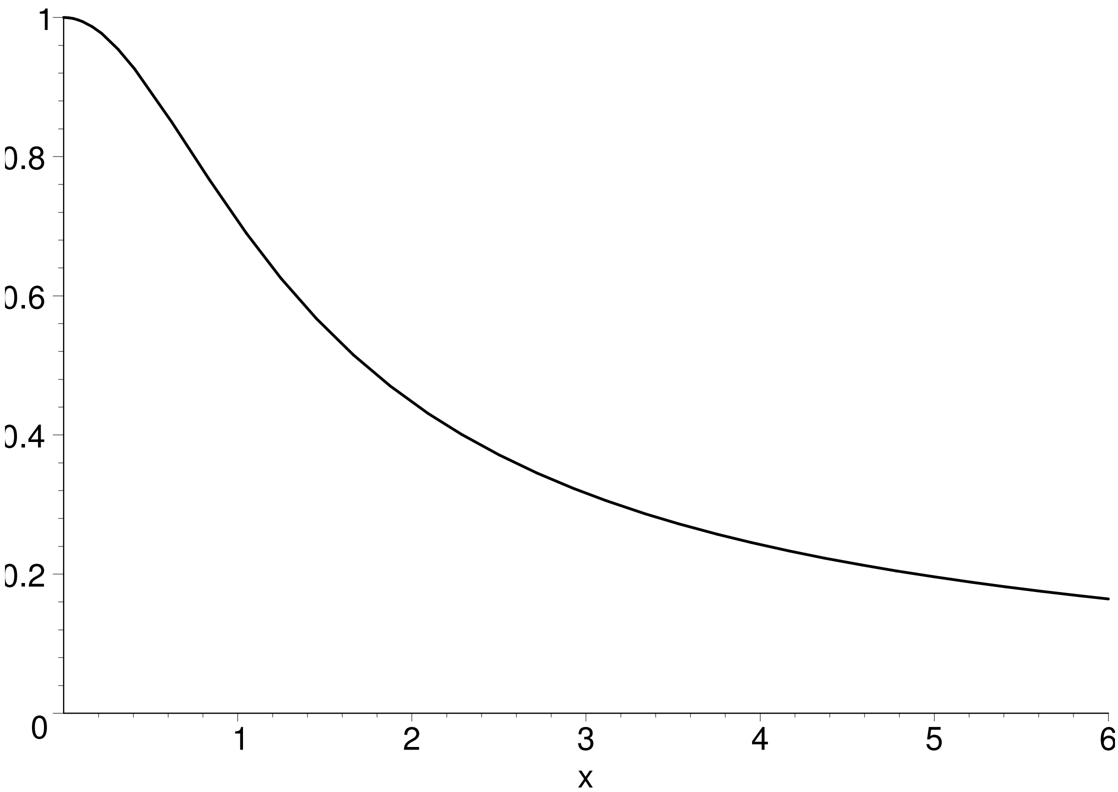
  or the faster decreasing *Perona–Malik diffusivity* (allowing edge enhancement)

$$g(|\boldsymbol{\nabla} u|) \;:=\; \frac{1}{1 + |\boldsymbol{\nabla} u|^2/\lambda^2}\,.$$

- The positive parameter $\lambda$ serves as *contrast parameter:*
  – Locations with $|\boldsymbol{\nabla} u| > \lambda$ are regarded as edges.
  – Here the diffusivity is reduced significantly.

MI 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

## A Continuous Diffusion Filter (3)




Plot of the Charbonnier diffusivity for $\lambda = 1$. Author: B. Burgeth.

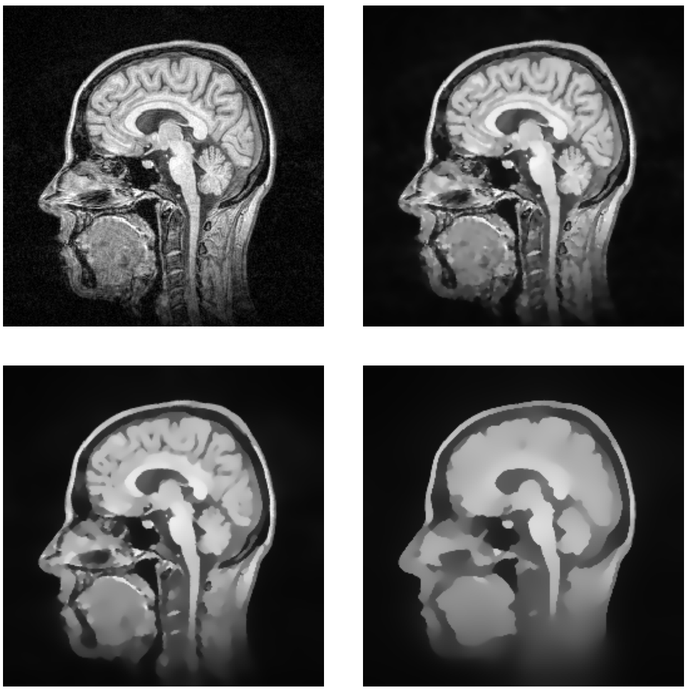
## A Continuous Diffusion Filter (4)

**(a) Left:** Noisy original image, $256 \times 256$ pixels. **(b) Middle:** Nonlinear diffusion with Charbonnier diffusivity, $\lambda = 0.1$, $t = 500$. **(c) Right:** Affine rescaling of the range of (b) to the interval $[0, 255]$. Author: J. Weickert.



## A Continuous Diffusion Filter (5)

original, $256 \times 256$ $\quad$ $t = 5$

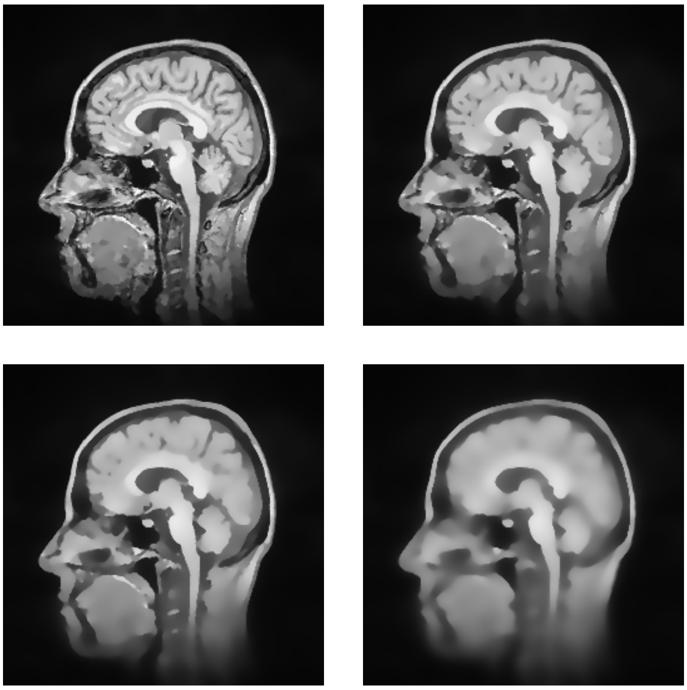


$t = 20$ $\quad$ $t = 80$

Influence of the diffusion time $t$ on the result of a diffusion filter with Perona-Malik diffusivity ($\lambda = 4$). Author: J. Weickert.

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

$\lambda = 2$ $\lambda = 4$



$\lambda = 6$ $\lambda = 10$

Influence of the contrast parameter $\lambda$ on the result of a diffusion filter with Perona-Malik diffusivity ($t = 20$). Author: J. Weickert.



MIA 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

# A Simple Algorithm

## Finite Difference Approximation

- rewrite the continuous diffusion equation $\partial_t u = \operatorname{div}\left(g(|\boldsymbol{\nabla} u|)\, \boldsymbol{\nabla} u\right)$ as

$$\partial_t u \;=\; \partial_x\Big(g(|\boldsymbol{\nabla} u|)\, \partial_x u\Big) \;+\; \partial_y\Big(g(|\boldsymbol{\nabla} u|)\, \partial_y u\Big)$$

- $u_{i,j}^k$ approximates $u\left((i-\frac{1}{2})h_1,\, (j-\frac{1}{2})h_2,\, k\tau\right)$ with spatial grid sizes $h_1$, $h_2$, and time step size $\tau$.
- finite difference approximations in $\left((i-\frac{1}{2})h_1,\, (j-\frac{1}{2})h_2,\, k\tau\right)$:

$$\begin{aligned}
\partial_t u &\approx \frac{u_{i,j}^{k+1} - u_{i,j}^k}{\tau} \\
\partial_x(g\,\partial_x u) &\approx \frac{1}{h_1}\left((g\,\partial_x u)_{i+1/2,j}^k - (g\,\partial_x u)_{i-1/2,j}^k\right) \\
&\approx \frac{1}{h_1}\left(g_{i+1/2,j}^k\, \frac{u_{i+1,j}^k - u_{i,j}^k}{h_1} \;-\; g_{i-1/2,j}^k\, \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1}\right).
\end{aligned}$$

## A Simple Algorithm (2)

- The diffusivity $g_{i+1/2,j}^k$ is approximated by the arithmetic mean of $g_{i,j}^k$ and $g_{i+1,j}^k$.
- $g_{i,j}^k$ can be obtained with central differences:

$$
\begin{aligned}
g_{i,j}^k &= g\left(\sqrt{(\partial_x u)^2 + (\partial_y u)^2}\right)\Big|_{i,j}^k \\
&\approx g\left(\sqrt{\left(\frac{u_{i+1,j}^k - u_{i-1,j}^k}{2h_1}\right)^2 + \left(\frac{u_{i,j+1}^k - u_{i,j-1}^k}{2h_2}\right)^2}\right)
\end{aligned}
$$



## A Simple Algorithm (3)

### Resulting Numerical Scheme

$$
\begin{aligned}
\frac{u_{i,j}^{k+1} - u_{i,j}^k}{\tau} &= \frac{1}{h_1}\left(g_{i+1/2,j}^k \frac{u_{i+1,j}^k - u_{i,j}^k}{h_1} - g_{i-1/2,j}^k \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1}\right) \\
&+ \frac{1}{h_2}\left(g_{i,j+1/2}^k \frac{u_{i,j+1}^k - u_{i,j}^k}{h_2} - g_{i,j-1/2}^k \frac{u_{i,j}^k - u_{i,j-1}^k}{h_2}\right)
\end{aligned}
$$

The unknown $u_{i,j}^{k+1}$ can be computed *explicitly* from known values at time step $k$:

$$
\begin{aligned}
u_{i,j}^{k+1} = u_{i,j}^k &+ \frac{\tau}{h_1}\left(g_{i+1/2,j}^k \frac{u_{i+1,j}^k - u_{i,j}^k}{h_1} - g_{i-1/2,j}^k \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1}\right), \\
&+ \frac{\tau}{h_2}\left(g_{i,j+1/2}^k \frac{u_{i,j+1}^k - u_{i,j}^k}{h_2} - g_{i,j-1/2}^k \frac{u_{i,j}^k - u_{i,j-1}^k}{h_2}\right).
\end{aligned}
$$

This scheme is called an *explicit scheme.*

## A Simple Algorithm (4)

- The explicit scheme can be rewritten as weighted averaging:

$$
\begin{aligned}
u_{i,j}^{k+1} \;=\;& \frac{\tau g_{i+1/2,j}^k}{h_1^2}\, u_{i+1,j}^k + \frac{\tau g_{i-1/2,j}^k}{h_1^2}\, u_{i-1,j}^k + \frac{\tau g_{i,j+1/2}^k}{h_2^2}\, u_{i,j+1}^k + \frac{\tau g_{i,j-1/2}^k}{h_2^2}\, u_{i,j-1}^k \\
&+ \left(1 - \frac{\tau g_{i+1/2,j}^k}{h_1^2} - \frac{\tau g_{i-1/2,j}^k}{h_1^2} - \frac{\tau g_{i,j+1/2}^k}{h_2^2} - \frac{\tau g_{i,j-1/2}^k}{h_2^2}\right) u_{i,j}^k
\end{aligned}
$$

- At the image borders, introduce an additional one-pixel layer by mirroring. This ensures reflecting boundary conditions.

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

## A Simple Algorithm (5)

**Stencil Notation for $h_1 := h_2 := 1$:**

| | | |
|---|---|---|
| $0$ | $\tau g_{i,j+1/2}^k$ | $0$ |
| $\tau g_{i-1/2,j}^k$ | $-\tau g_{i,j+1/2}^k$<br>$-\tau g_{i-1/2,j}^k + \;1\; - \tau g_{i+1/2,j}^k$<br>$-\tau g_{i,j-1/2}^k$ | $\tau g_{i+1/2,j}^k$ |
| $0$ | $\tau g_{i,j-1/2}^k$ | $0$ |

- space- and time-dependent adaptive averaging where the weights sum up to 1
- central weight becomes negative if $\tau$ is too large $\Longrightarrow$ instability
- For $|g| \leq 1$ we get stable convex combinations if $\tau \leq 0.25$:
  All weights are nonnegative and sum up to $1$.
  In this case no over- and undershoots are possible.

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

# Comparison to Bilateral Filtering

- Nonlinear diffusion is close in spirit to bilateral filtering:
  - Results look similar.
  - The diffusion time $t$ resembles the spatial parameter $\sigma_s$.
  - The contrast parameter $\lambda$ resembles the tonal parameter $\sigma_t$.
- Main differences:
  - Diffusion filtering iterates a small adaptive kernel.
    Bilateral filtering uses one large adaptive kernel.
  - Diffusion filtering preserves the average grey level.
    Bilateral filtering does not.
    (see Assignment C8, Problem 1, and H8, Problem 2)





# Summary

- Nonlinear diffusion filters regard the original image as initial state of a diffusion process.
- The diffusivity is a nonlinear function of the evolving image.
  It is reduced at edges.
- A discretisation leads to an iterated adaptive averaging with a $3 \times 3$ mask.
- The weights of the mask depend on the diffusivity (and the time step size).
- Nonlinear diffusion filtering preserves the average grey value.
- No over- or undershoots appear, i.e. it satisfies a maximum-minimum principle.
- two model parameters: diffusion time $t$, contrast parameter $\lambda$

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

# References


- P. Perona, J. Malik: Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 12, 629–639, 1990.
  *(the first journal paper on nonlinear diffusion filtering)*
- J. Weickert: Nonlinear diffusion filtering. In B. Jähne, H. Haußecker, P. Geißler (Eds.): *Handbook of Computer Vision and Applications, Vol. 2: Signal Processing and Pattern Recognition*. Academic Press, San Diego, pp. 423–450, 1999.
  *(survey paper dealing also with implementational aspects)*
- J. Weickert: *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart, 1998.
  (`http://www.mia.uni-saarland.de/weickert/Papers/book.pdf`)
  *(monograph on continuous and discrete foundations, as well as on modelling aspects)*
- G. Aubert, P. Kornprobst: *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*. Second Edition, Springer, New York, 2006.
  *(for the mathematically inclined reader)*






# Assignment C8 – Classroom Work

**Problem 1 (Bilateral Filtering and NL-Means)**

Consider a filter of the form

$$u_i = \sum_{j=1}^{N} q_{i,j} f_j \qquad \forall i \in \{1, \ldots, N\}$$

where $\boldsymbol{f}$ is a discrete signal of length $N$ and $\boldsymbol{u}$ is its filtered version. Using an $N \times N$ matrix $\boldsymbol{Q}$ with entries $q_{i,j}$ this can be reformulated as

$$\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{f} \qquad\qquad (*).$$

(a) Prove that if all columns of $\boldsymbol{Q}$ have sum 1, then the average grey level is preserved, i.e.

$$\frac{1}{N}\sum_{i=1}^{N} u_i = \frac{1}{N}\sum_{i=1}^{N} f_i\,.$$

(b) Prove that if all rows of $\boldsymbol{Q}$ have sum 1 and all entries of $\boldsymbol{Q}$ are nonnegative, then the filter $(*)$ satisfies the maximum-minimum principle

$$\min_j f_j \le u_i \le \max_j f_j \qquad \forall i \in \{1, \ldots, N\}.$$

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

(c) Show that bilateral filtering and NL-means satisfy the row sum condition in (b).

(d) Consider the discrete signal $\boldsymbol{f} = (1, 0, 0)$ and the weighting function

$$g(x) = \frac{1}{1+x^2}.$$

Show that bilateral filtering and NL-means do not fulfil the column sum condition from (a). You can assume reflecting boundary conditions if necessary.





# Assignment H8 – Homework

**Problem 1 (Multiple Choice)** (1+1+1+1+1+1 Points)

Which of the following statements are true and which ones are false? Justify your answers for each statement.

*Remark: You only get points if your justification is correct. Simply stating true or false is not enough to obtain points in this problem.*

(a) By multiplying the Fourier transform of an image iteratively with a sinc function, one obtains a Gaussian smoothed image in the spatial domain.

(b) Huffman coding is most efficient if a word consists of letters that all occur with the same frequency.

(c) Discrete histogram equalisation is always invertible.

(d) If the discrete cosine transform of a signal $f$ is nonsymmetric, then $f$ has a nonzero imaginary part.

(e) Taking the forward differences of the backward differences of a 1D signal approximates its second order derivative.

(f) Hard wavelet shrinkage without cycle spinning is idempotent.

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

# Assignment H8 (2)

**Problem 2 (Nonlinear Diffusion)** (2+2+2 points)

In the explicit scheme for 1-D nonlinear diffusion filtering, the new value $u_i^{k+1}$ at pixel $i$ can be computed from the known values of the old time step $k$ via

$$u_i^{k+1} = u_i^k + \frac{\tau}{h}\left(g_{i+1/2}^k \frac{u_{i+1}^k - u_i^k}{h} - g_{i-1/2}^k \frac{u_i^k - u_{i-1}^k}{h}\right)$$

where $\tau$ is the time step size, $h$ is the grid size, and $g_{i\pm1/2}^k$ approximates the diffusivity in the intermediate location $i \pm 1/2$ at time level $k$.

(a) How does the corresponding stencil look like?

(b) For the whole signal, this iteration step can be formulated as a single matrix-vector product

$$\boldsymbol{u}^{k+1} = \boldsymbol{Q}(\boldsymbol{u}^k)\ \boldsymbol{u}^k\ .$$

Here, $\boldsymbol{u}^k$ and $\boldsymbol{u}^{k+1}$ are vectors of size $N$, and $\boldsymbol{Q}(\boldsymbol{u}^k)$ is an $N \times N$ matrix.
Which structure has the matrix $\boldsymbol{Q}(\boldsymbol{u}^k)$? How is it related to the previous stencil?
Hint: You can assume that the signal is reflected at the boundaries, i.e. one may use dummy values $u_0^k := u_1^k$ and $u_{N+1}^k := u_N^k$.

(c) Compute the row and column sums of $\boldsymbol{Q}(\boldsymbol{u}^k)$.
What are your findings considering the results proven in C8, Problem 1(a) and 1(b)?



# Assignment H8 (3)

Please download the required files from the webpage

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

into your own directory. You can unpack them with the command `tar xvzf Ex08.tar.gz`.

**Problem 3 (Wavelet Shrinkage)** (3+3 points)

The goal of this assignment is to implement different strategies for wavelet shrinkage and to use them for denoising. The file `wavelet.c` contains an implementation of the 2-D Haar wavelet transform and the corresponding inverse transform. Note that the arrays for the image and its wavelet coefficients have the index range `i=0,...,nx-1` and `i=0,...,ny-1`. The scaling coefficient has the index `(i,j) = (0,0)`.

(a) Supplement the missing code for hard, soft and Garrote shrinkage in the respective routines `hard_shrinkage`, `soft_shrinkage`, and `garrote_shrinkage`.

(b) Compile the program with the command

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

The image `trui-n30.pgm` should be denoised via wavelet shrinkage. Find suitable thresholds for each shrinkage strategy. What do you notice?

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

## Assignment H8 (4)

**Problem 4 (NL-Means)** (2+2+2 points)

The image `trui-n30.pgm` has been created by adding Gaussian noise of standard deviation $30$. Compile the program `nl_means` with the command

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

(a) Option $0$ of `nl_means` provides an implementation of the NL-means filter. There are three parameters to choose:

- The patch size $m$ defines the size of a square patch with dimension $(2m+1) \times (2m+1)$.
- The search window size $n$ gives the size of the square search space with dimension $(2n+1) \times (2n+1)$ around pixel $i$.
- The filter parameter $\sigma$ is the standard deviation of the Gaussian weight function $g$, that is defined as $g(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right)$.

Denoise `trui-n30.pgm` using the NL-means implementation from `nl_means`.

(b) Now try to denoise `trui-n30.pgm` by a Gaussian convolution using option $1$ of the program `nl_means`.

(c) The so-called method noise is given by the difference between noisy image and filtered version. The program `nl_means` automatically outputs a method noise image for every denoising result you produce. It calculates the difference between two images and shifts the result by $127.5$.
Take a look at the method noise images that were generated for your results from (a) and (b). What do you observe? Why is the result shifted by $127.5$? How should the method noise look like in the optimal case? Justify your answers.

MIA 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

## Assignment H8 (5)

**Submission**

Please submit the theoretical Problems 1 and 2 in handwritten form before the lecture. For the practical Problems 3 and 4 submit the files as follows: Rename the main directory `Ex08` to `Ex08_<your_name>` and use the command

```
tar czvf Ex08_<your_name>.tar.gz Ex08_<your_name>
```

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

- The source file `wavelet.c` for Problam 3(a);
- One reasonably denoised image for each shrinkage type from Problem 3(b);
- One reasonably denoised image of Problem 4(a) and one for Problem 4(b);
- The method noise images of Problem 4(c);
- a text file README that contains the parameters which are used for the creation of the submitted images; the answers to the questions of Problem 3(b) and 4(c); information on all people working together for this assignment.

Please make sure that only your final version of the programs and images are included.
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, June 7, 10 am (before the lecture)