

Computergrafik 2: Filtern im Ortsraum

Prof. Dr. Michael Rohs, Dipl.-Inform. Sven Kratz

michael.rohs@ifi.lmu.de

MHCI Lab, LMU München

Folien teilweise von Andreas Butz, sowie von Klaus D. Tönnies
(Grundlagen der Bildverarbeitung. Pearson Studium, 2005.)

Vorlesungen

Datum	Thema
24.4.	Einführung, Organisatorisches (Übungen, Klausur)
1.5./8.5.	keine Vorlesungen (wegen 1. Mai und CHI-Konferenz)
15.5.	Abtastung von Bildern, Punktbasierte Verfahren der Bildverbesserung
22.5.	Licht, Farbe, Farbmanagement
30.5.	Konvolution, Filterung im Ortsraum (Verschiebung wegen Pfingstdienstag)
5.6.	Fouriertransformation: Grundlagen
12.6.	Filterung im Frequenzraum
19.6.	Kanten, Linien, Ecken
26.6.	Segmentierung
3.7.	Segmentierung, Morphologische Operationen
10.7.	Klassifikation
17.7.	Image Matching
24.7.	Klausur (Hörsaal M 018 im Hauptgebäude, 14-16 Uhr)

Themen heute

- Konvolution und Korrelation
- Lineare Filterung
- Nichtlineare Filterung
- Interpolation

Filterung im Ortsraum

- Lineare Filterung
- $m \times n$ Filtermaske
- Lokale Umgebung
- Vorgegebene Operation auf Pixeln in lokaler Umgebung
- Skalarprodukt
 $f(x-1, y-1) * w(-1, -1) +$
 $\dots + f(x, y) * w(0, 0) +$
 $\dots + f(x+1, y+1) * w(1, 1)$

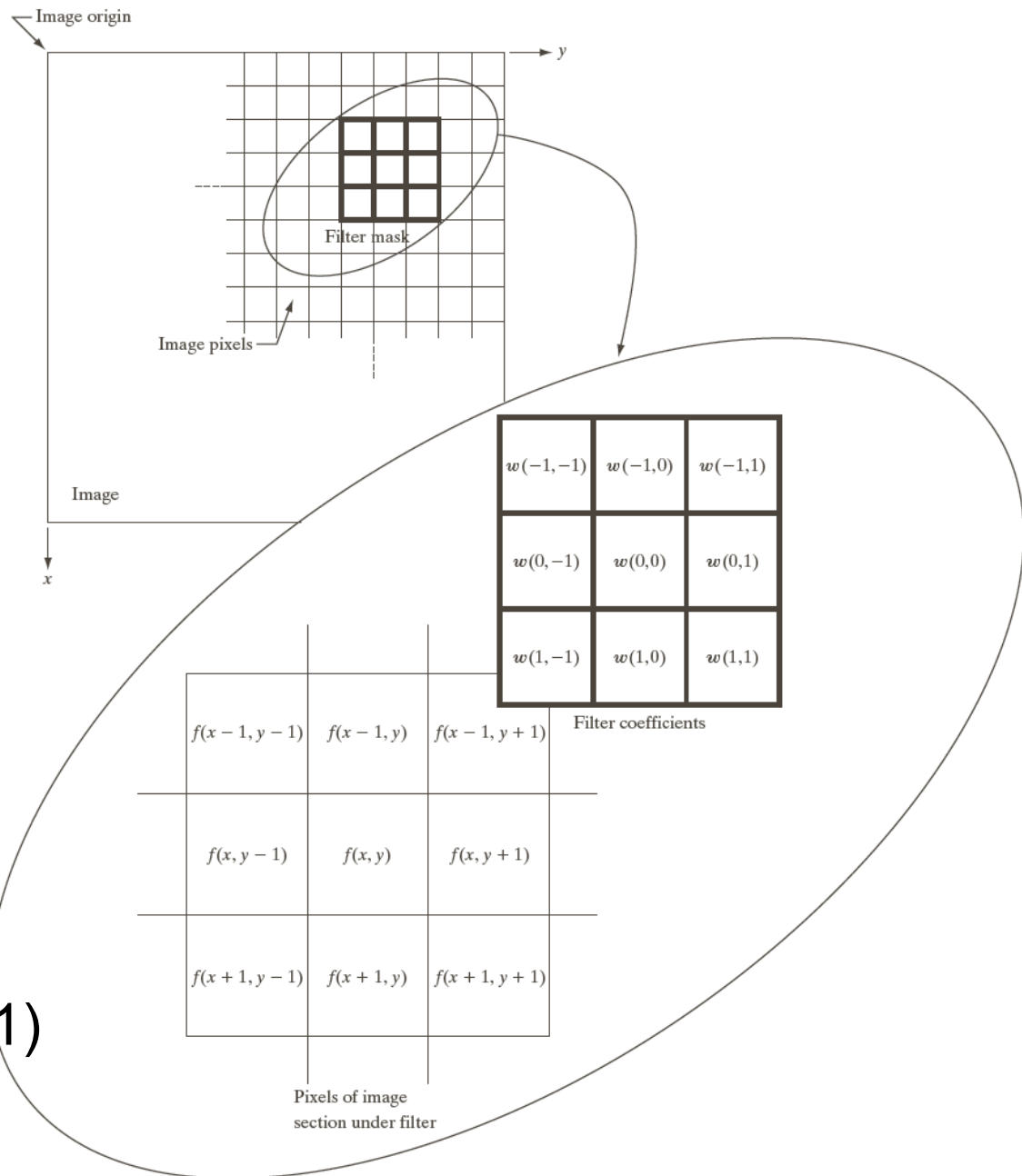


Abbildung: © R. C. Gonzalez & R. E. Woods, Digital Image Processing

Filtern im Ortsraum

- Filterung als gewichtetes Mittel

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

- $m \times n$ Filtermaske mit $m = 2a+1$, $n = 2b+1$
- $M \times N$ Bild mit M Zeilen und N Spalten
- Üblicherweise
 - Filtermaske begrenzt, Gewicht normalisiert (sum=1)
 - Seitenlänge des Filters ungerade

Filtergrößen

- Stärke der Glättung als Effekt der Filtergröße
 - Bild $M \times N = 500 \times 500$
 - quadratisches Boxfilter



Abbildung: © R. C. Gonzalez & R. E. Woods, Digital Image Processing

Tiefpass: Wirkung



$$* \begin{array}{|c|c|c|} \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline \end{array} =$$



Nützlich z.B. gegen Rauschen und Alias-Effekte

Hochpass: Wirkung



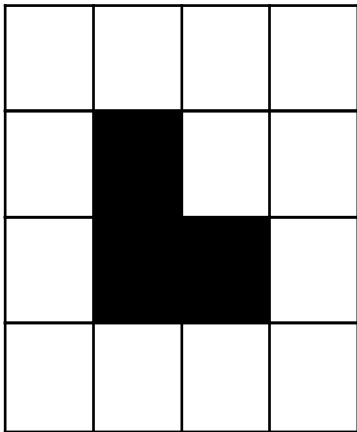
$$* \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 9 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} =$$



Nützlich zum Scharfzeichnen und Kanten finden

Hintereinanderschreiben von Pixeln

- 2-dimensionales Bild (M Zeilen, N Spalten) wird zu MN Spaltenvektor
- manchmal von Vorteil beim Rechnen



$$f = \begin{pmatrix} f(0,0) \\ f(1,0) \\ \vdots \\ f(N-1,0) \\ f(0,1) \\ \vdots \\ f(N-1,1) \\ \vdots \\ \vdots \\ f(0,M-1) \\ \vdots \\ f(N-1,M-1) \end{pmatrix}$$

Lineare Operatoren

- Definition eines linearen Operators $O(\cdot)$

$$O(\alpha \vec{f} + \beta \vec{g}) = \alpha O(\vec{f}) + \beta O(\vec{g})$$

für alle Skalare α, β

- Superpositionsprinzip: komplexe Signale können in einfachere Komponenten zerlegt werden

Nicht-Lineare Operatoren?

- Definition eines linearen Operators $O(\cdot)$

$$O(\alpha \vec{f} + \beta \vec{g}) = \alpha O(\vec{f}) + \beta O(\vec{g})$$

für alle Skalare α, β

- Beispiel eines nicht-linearen Operators: Square

$$\text{sq}(\alpha f + \beta g) \stackrel{?}{=} \alpha \text{sq}(f) + \beta \text{sq}(g)$$

Gegenbeispiel: $\alpha = 1, \beta = 1, f = \{1\}, g = \{3\}$

$$\text{sq}(\{1\} + \{3\}) = \text{sq}(\{4\}) = \{16\}$$

$$\text{sq}(\{1\}) + \text{sq}(\{3\}) = \{1\} + \{9\} = \{10\}$$

Linearer Operator als Matrix

Linearer operator O lässt sich auch als Matrix ausdrücken:

Seien x, y die hintereinandergeschriebenen Pixel zweier Bilder, dann ist

$$\vec{y} = A\vec{x}$$

Jeder Eintrag $A_{i,j}$ gibt an, mit welchem Gewicht Pixel j aus x auf Pixel i in y abgebildet wird

Verschiebungsinvariante Operatoren

- ... sind lineare Operatoren, deren Wirkung unabhängig vom Ort ist
- Beispiel: Filtermaske wirkt überall im Bild gleich
- Beispiel: gleichmäßige Unschärfe im Bild durch Bewegung der Kamera

Konvolution (Faltung)

- Seien f, g abgetastete Bilder mit unendlicher Größe, m, n , Skalare

$$(g * f)(m, n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} g(i, j) f(m - i, n - j)$$

Heißt Konvolution der Funktion f mit g

- g heißt die Konvolutionsfunktion
- Funktioniert so nur für unendlich große Bilder

Eigenschaften der Konvolution

- Linear & verschiebungsinvariant
- Kommutativ & assoziativ

$$[g_1 * g_2](m, n) = [g_2 * g_1](m, n)$$

$$g_1 * ([g_2 * g_3](m, n)) = [g_1 * g_2](m, n) * g_3(m, n)$$

- D.h. wir können mehrere Konvolutionen vorab kombinieren und dann gemeinsam anwenden

Konvolution vereinfacht

Hat die Konvolutionsfunktion g nur einen begrenzten Bereich, in dem $g \neq 0$, dann heißt dieser Bereich Kern von g (kernel)

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	-1	0	0	0
0	0	0	1	0	-1	0	0	0
0	0	0	1	0	-1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Konvolution anschaulich

1	0	-1	0	0	0	0
1	0	-1	1	0	0	0
1	0	-1	1	1	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

	1	2	0	-2	-1	
	2	2	0	-2	-2	
	3	2	0	-2	-3	
	3	2	0	-2	-3	
	2	2	0	-2	-2	

Kernel wird um 180° rotiert

Worst case Aufwand im Ortsraum: N^4 für Kantenlänge N des Bildes

Verwendung der Konvolution

- Sobel-Operator: Kanten finden



*

1	0	-1
2	0	-2
1	0	-1

=



- „erste Ableitung“, „Differenzenquotient“

Konvolution des Dirac-Impulses

Dirac-Impuls
= Einheitsimpuls
= unit impulse

	Origin ↙	$f(x, y)$			
0	0	0	0	0	
0	0	0	0	0	$w(x, y)$
0	0	1	0	0	1 2 3
0	0	0	0	0	4 5 6
0	0	0	0	0	7 8 9

= Punktverteilungsfunktion
= Point Spread Function (PSF)

Abbildung: © R. C. Gonzalez & R. E. Woods, Digital Image Processing

Konvolution des Dirac-Impulses

Dirac-Impuls
= Einheitsimpuls
= unit impulse

↙ Origin	$f(x, y)$			
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

$w(x, y)$		
1	2	3
4	5	6
7	8	9

Padded f

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

↙ Rotated w

9	8	7	0	0	0	0	0	0	0
6	5	4	0	0	0	0	0	0	0
3	2	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Full convolution result

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	3	0	0	0	0
0	0	0	4	5	6	0	0	0	0
0	0	0	7	8	9	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Cropped convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

Punktantwort
= Punktverteilungs-
funktion
= Point Spread Func-
tion (PSF)

Abbildung: © R. C. Gonzalez & R. E. Woods, Digital Image Processing

Punktantwort (Point Spread Function, PSF)

- Faltung eines einzelnen Dirac-Impulses
 - Bild mit einem einzigen weißen Pixel
- Abgetastetes Bild =
Folge von Dirac-Impulsen * Pixelhelligkeiten
- → Gesamtwirkung durch PSF vollständig beschrieben
- → falls PSF umkehrbar, kann Wirkung rückgängig gemacht werden
- PSF kann manchmal experimentell bestimmt werden

Punktantwort (Point Spread Function, PSF)

- Alternative Repräsentation der Faltung

$$(f * h)(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cdot h(x, \alpha, y, \beta) \left(= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cdot h^*(x - \alpha, y - \beta) \right)$$

- Einheitsimpuls an Pixel (a,b)

$$\delta(x - a, y - b) = \begin{cases} 1 & x = a \wedge y = b \\ 0 & \text{sonst} \end{cases}$$

- Punktantwort

$$(\delta * h)(\alpha, \beta) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \delta(x - a, y - b) \cdot h(x, \alpha, y, \beta) = h(a, \alpha, b, \beta)$$

„Bild“ = Einheitsimpuls an (a,b)

PSF

Punktantwort (Point Spread Function, PSF)

- Digitales Bild als Summe seiner Pixel

$$f(x, y) = \sum_{a=0}^{N-1} \sum_{b=0}^{M-1} f(a, b) \cdot \delta(x - a, y - b)$$

- Das Resultat einer linearen Filterung ist die Überlagerung der Punktantworten

$$g(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cdot h(x, \alpha, y, \beta)$$

$$= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \sum_{a=0}^{N-1} \sum_{b=0}^{M-1} f(a, b) \cdot \delta(x - a, y - b) \cdot h(x, \alpha, y, \beta)$$

Korrelation

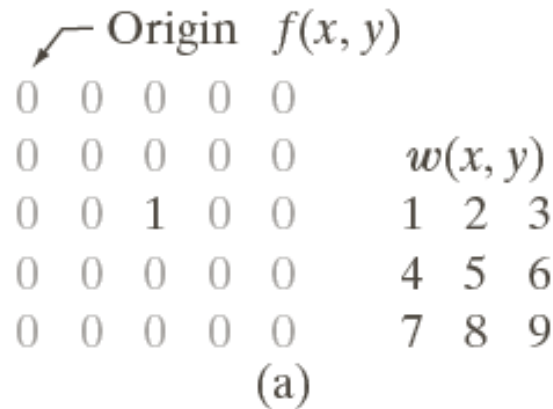
- Seien f, g abgetastete Bilder mit unendlicher Größe, m, n , Skalare

$$(g * f)(m, n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} g(i, j) f(i + m, j + n)$$

Heißt Korrelation der Funktion f mit g

- g heißt die Korrelationsfunktion
- Funktioniert so nur für unendlich große Bilder

Korrelation



Kern nicht gedreht, sonst wie bei Konvolution

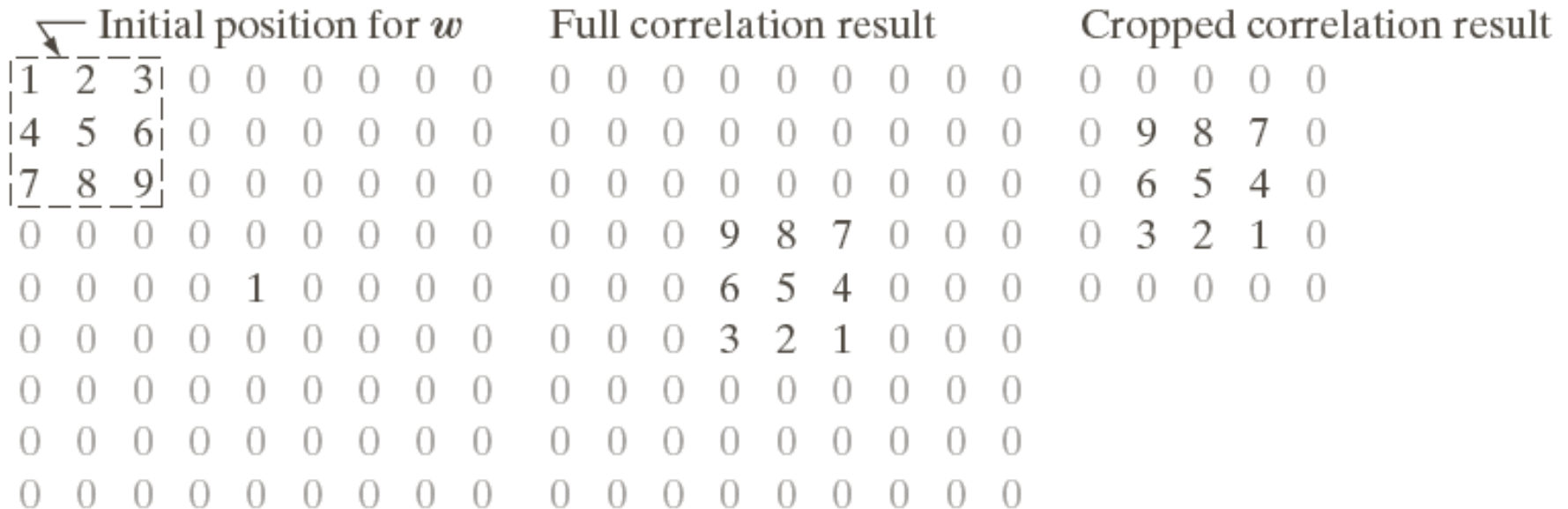
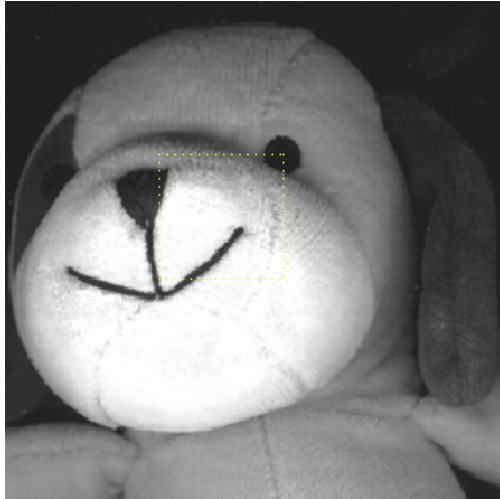
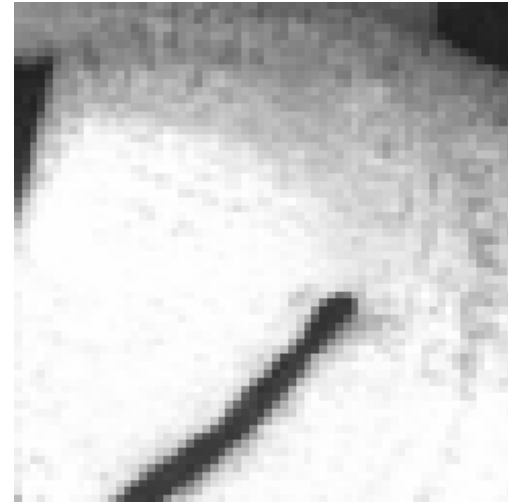


Abbildung: © R. C. Gonzalez & R. E. Woods, Digital Image Processing

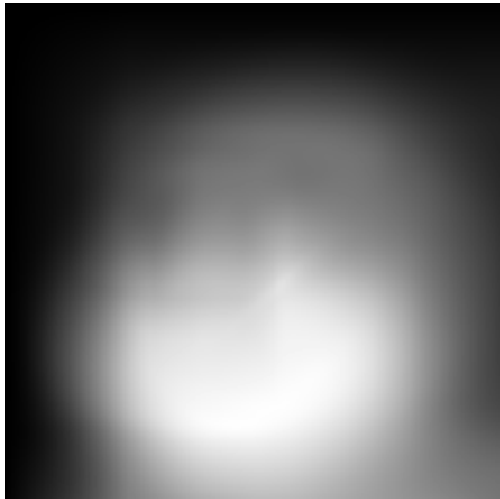
Verwendung der Korrelation



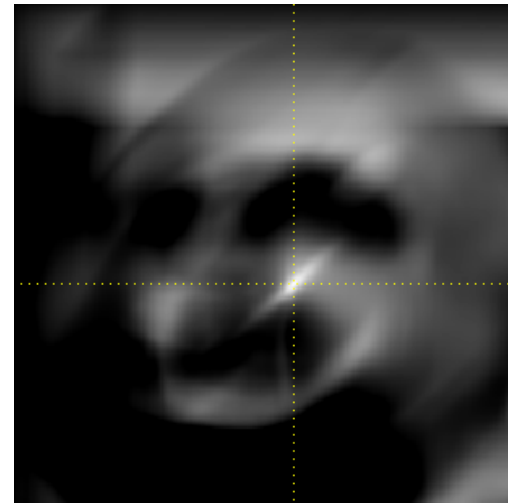
Ausgangsbild (f)



Korrelationsfunktion (g)



Ergebnis (f * g)



...normalisiert

Unterschiede Konvolution / Korrelation

- Konvolution und Korrelation sind zwei eng verwandte Filter-Operationen
- Gleich, falls der Kern von g symmetrisch unter 180° Rotation
- Beispiele:

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Tiefpass

-1	-1	-1
-1	9	-1
-1	-1	-1

Hochpass

Separierbarkeit linearer Filter

- Ein zweidimensionales Filter ist separierbar, falls Punktantwort durch Hintereinanderausführung zweier eindimensionaler Impulsantworten darstellbar
- Strategie: zerlege 2D Filter in einen x- und y-Kern die hintereinander angewandt werden
- Separierbarkeit in h_x und h_y :

$$g(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cdot h(x, \alpha, y, \beta)$$

$$= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cdot h_x(x, \alpha) \cdot h_y(y, \beta)$$

Separierbarkeit linearer Filter

- Verarbeitung zeilenweise

$$g(\alpha, \beta) = \sum_{y=0}^{M-1} h_y(y, \beta) \sum_{x=0}^{N-1} f(x, y) \cdot h_x(x, \alpha)$$

- Verarbeitung spaltenweise

$$g(\alpha, \beta) = \sum_{x=0}^{N-1} h_x(x, \alpha) \sum_{y=0}^{M-1} f(x, y) \cdot h_y(y, \beta)$$

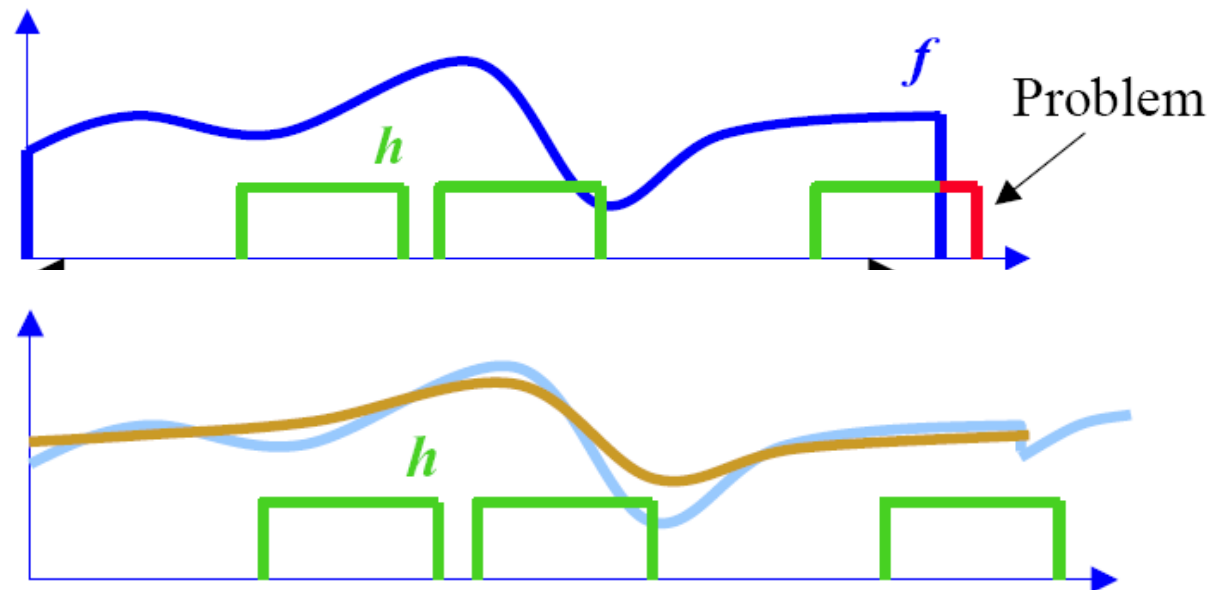
- Reduziert Rechenaufwand von $O(NM)$ auf $O(N+M)$

- Beispiel:

$$h = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, h_x = \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, h_y = \frac{1}{3} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Filterung am Bildrand

- Lösung 1: periodische Fortsetzung des Bildes
- Lösung 2: Spiegeln am Rand
- Lösung 3: Rand konstant fortsetzen
- Lösung 4: Werte außerhalb des Bildes auf 0 setzen („padding“)
- etc...



Filterung am Bildrand



Flächenbasierte Bildverbesserung

- Rauschen kann durch Integration einer Signalfolge reduziert werden
- Konstante Signalfolge
 - Integration über eine zeitliche Folge
 - Integration über eine homogene Fläche
- Lineare verschiebungsinvariante Operatoren
 - Konvolutionmethoden

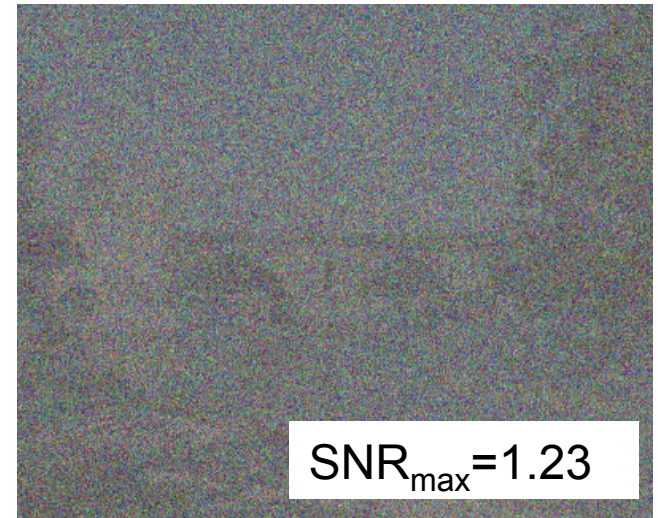
Zeitliche Folge

- Annahmen
 - Aufnahme mehrerer Bilder $g_i, i=1, l$ über einen gegebenen Zeitraum
 - Bild verändert sich über den Zeitraum nicht (keine Bewegung, keine Beleuchtungsänderung)
 - Erwartungswert E des Rauschens („noise“) n ist 0
- Näherung an die unverrauschte Funktion f
 - $$E\{g(m,n)\} = E\{f(m,n)\} + E\{n(m,n)\}$$
$$= E\{ f(m,n) \} + 0 = f(m,n)$$
 - Abschätzung von $E\{g(m,n)\}$ durch Integration über die Bilder



Beispiel: Integration über die Zeit

- Einzelne Aufnahme mit normalverteiltem Rauschen ($\text{SNR} \approx 1.2$)
- $\text{SNR}_{\max} = \text{max. Signalamplitude} / \text{Standardabweichung des Rauschens}$
- Addition von 10 bzw. 50 Aufnahmen



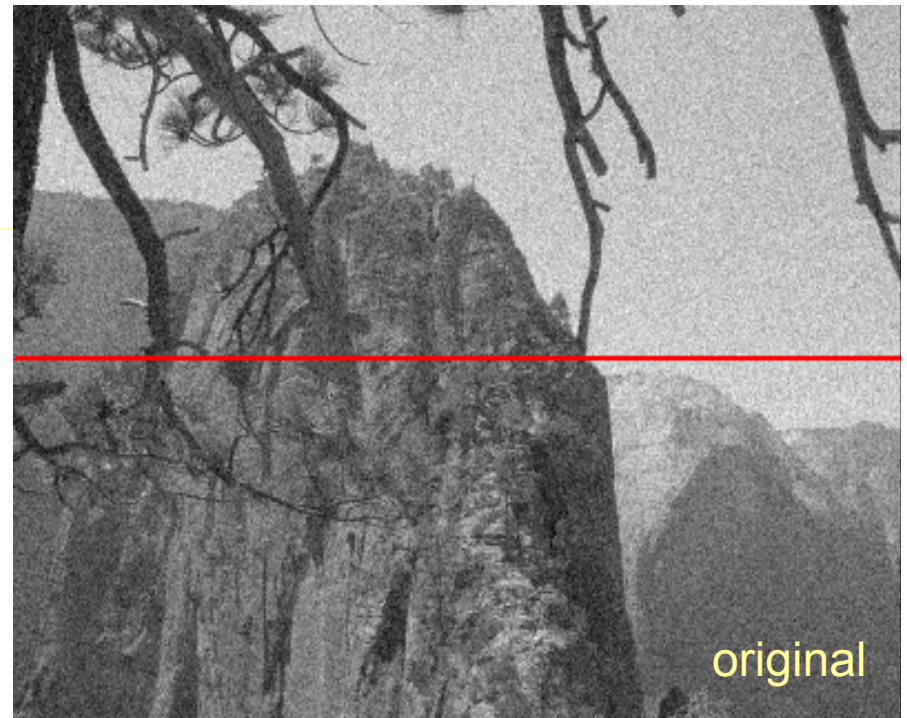
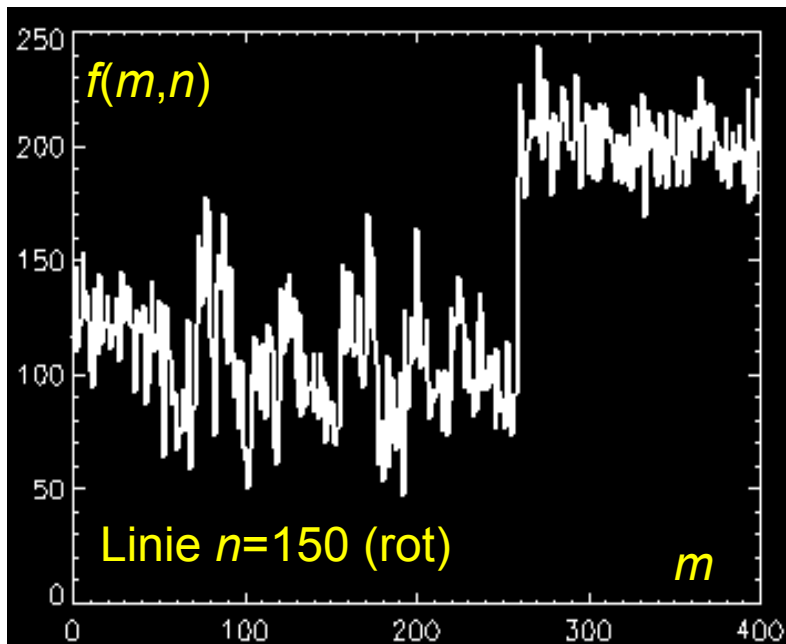
Integration über die Fläche

- Falls für eine Reihe von Bildpunkten (p_0, \dots, p_n) gilt, dass $f(p_i) = \text{const}$, dann kann Rauschen n mit $E\{n\} = 0$ durch Addition der gemessenen Funktionswerte $g(p_i)$ reduziert werden
- Annahmen:
 - Bild besteht aus homogenen Bereichen
 - Benachbarte Punkte haben den gleichen Grauwert
- Rauschunterdrückung:
 - Mittelwertbildung über vorgegebene Nachbarschaft



Mittelwertbildung durch Konvolution

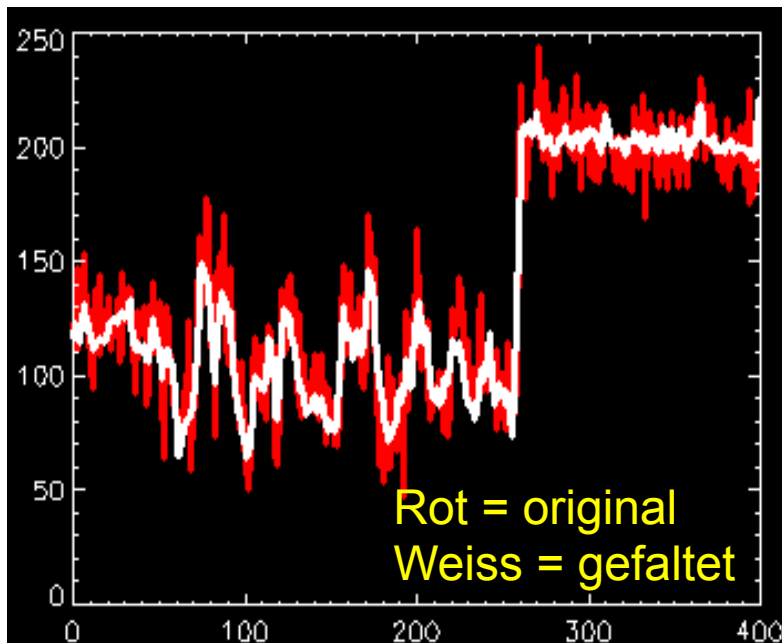
Konvolutionskern: Gleichmäßige Gewichtung der Pixel in einer gegebenen Nachbarschaft



3x3 Boxcar-Filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

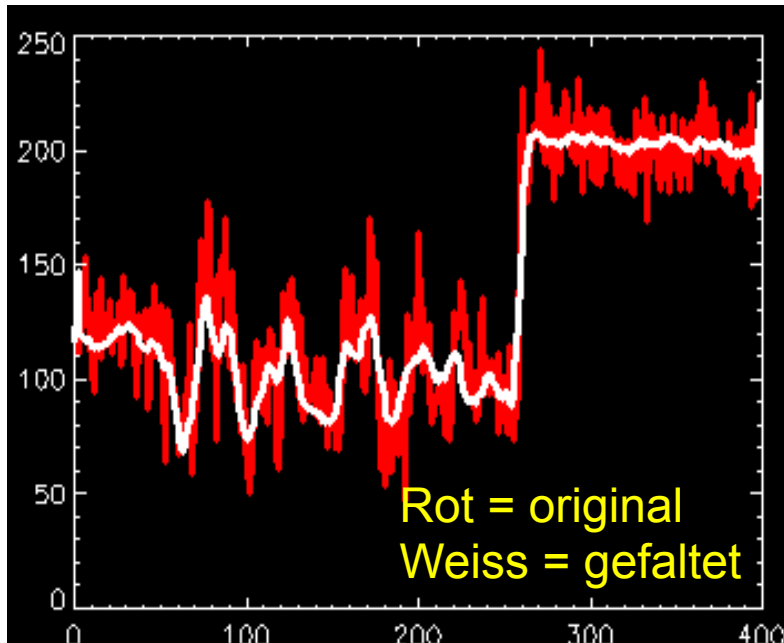
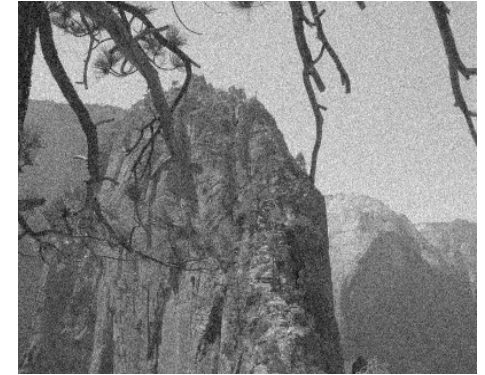
Filterkern



7x7 Boxcar-Filter

Beobachtung: Kanten werden degradiert

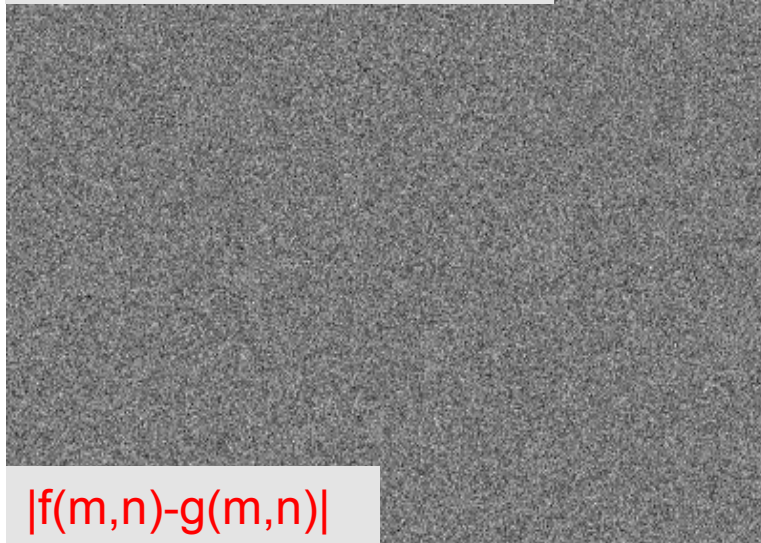
Grund: Annahme konstanter Funktionswerte ist nicht wahr



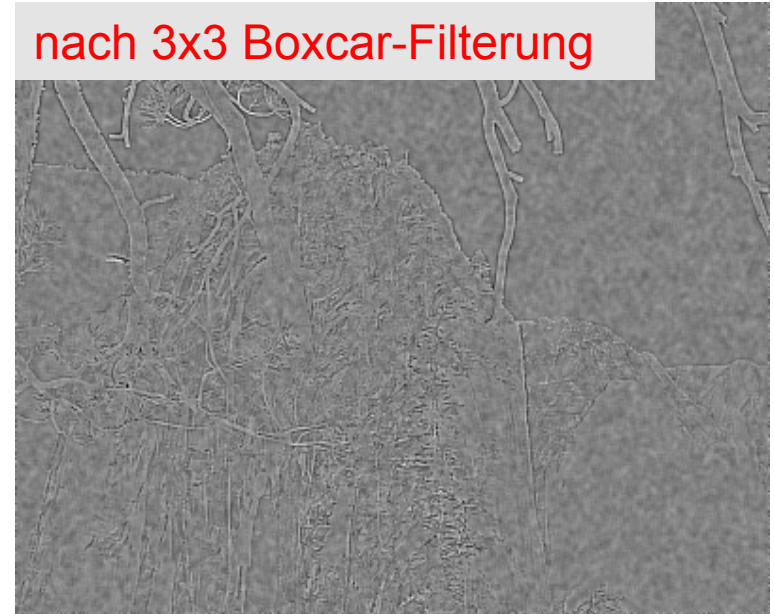
Verhalten an Kanten

- Differenz zwischen Original und Boxcar-gefiltertem Bild
 - Elimination von Strukturen

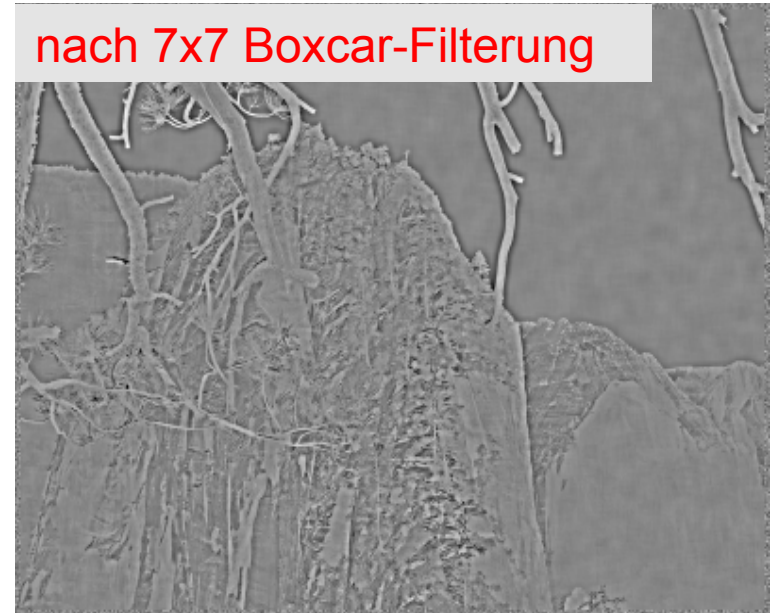
tatsächliches Rauschen



nach 3x3 Boxcar-Filterung



nach 7x7 Boxcar-Filterung



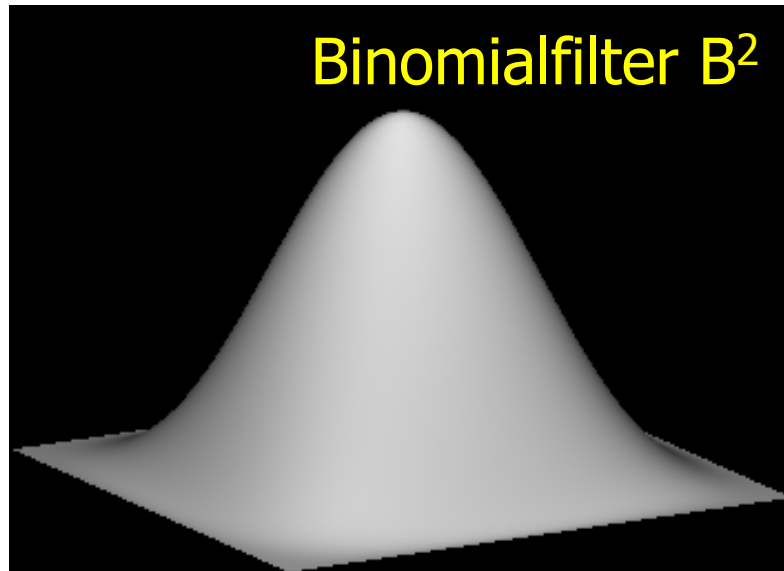
Tiefpassfilter zur Rauschunterdrückung



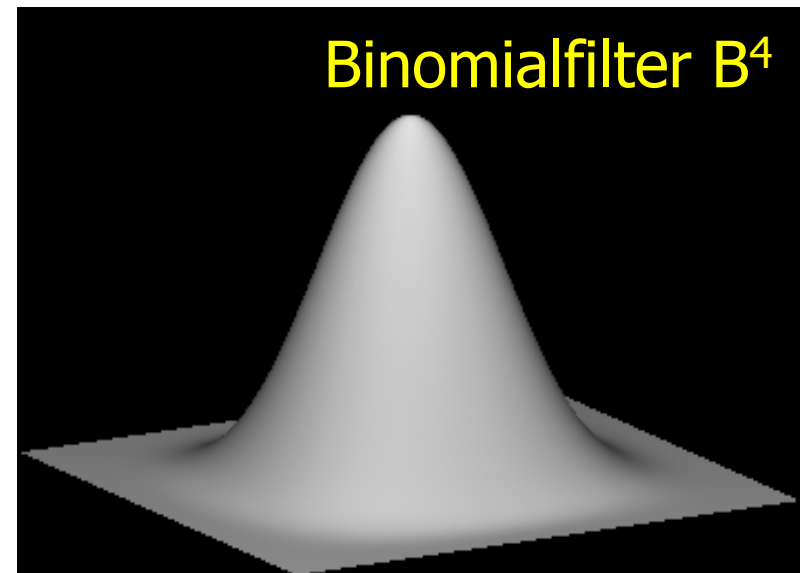
„Ringing Artefakte“



Transferfunktion des Binomialfilters



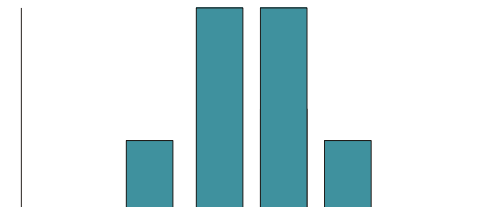
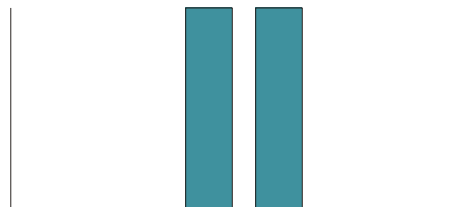
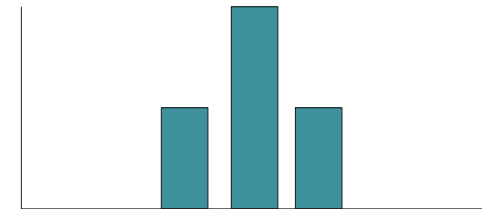
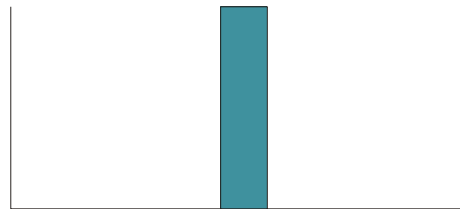
Weniger „Ringing Artefakte“
an den Kanten



Binomialfilter

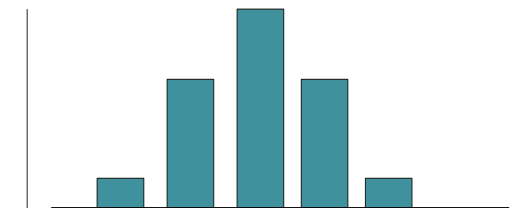
Eindimensionales Binomialfilter B^p :

$$\begin{aligned}
 B^0 &= 1^{-1} \cdot [1] \\
 B^1 &= 2^{-1} \cdot [1 \ 1] \\
 B^2 &= 4^{-1} \cdot [1 \ 2 \ 1] \\
 B^3 &= 8^{-1} \cdot [1 \ 3 \ 3 \ 1] \\
 B^4 &= 16^{-1} \cdot [1 \ 4 \ 6 \ 4 \ 1] \\
 &\dots
 \end{aligned}$$



Zweidimensionales Binomialfilter $\mathbf{B}^p = (B^p)^T * (B^p)$:

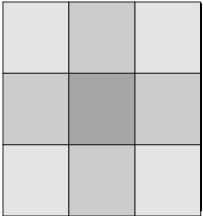
$$\mathbf{B}^2 = 4^{-1} \cdot [1 \ 2 \ 1]^T \cdot 4^{-1} \cdot [1 \ 2 \ 1] = 16^{-1} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$



Zweidimensionale Binomialfilter

$$\mathbf{B}^2 = 1/16 \cdot [1 \ 2 \ 1]^T \cdot [1 \ 2 \ 1] = 1/16 \cdot$$

1	2	1
2	4	2
1	2	1

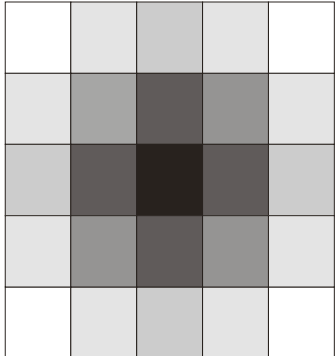


$$\mathbf{B}^3 = 1/64 \cdot [1 \ 3 \ 3 \ 1]^T \cdot [1 \ 3 \ 3 \ 1] = 1/64 \cdot$$

1	3	3	1
3	9	9	3
3	9	9	3
1	3	3	1

$$\mathbf{B}^4 = 1/256 \cdot$$

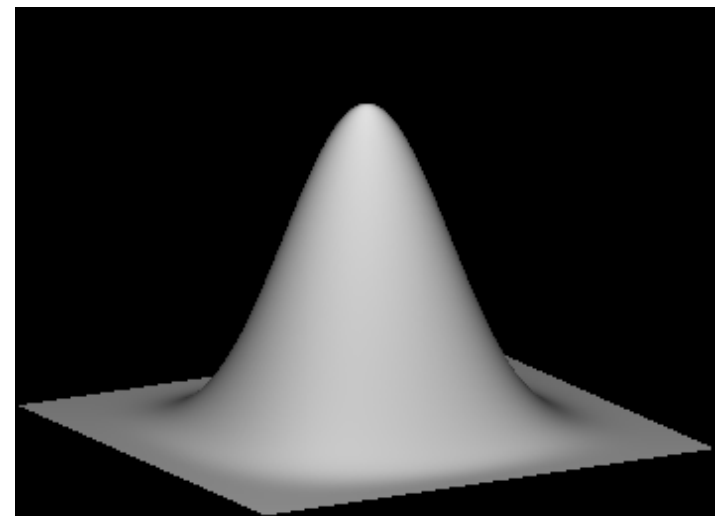
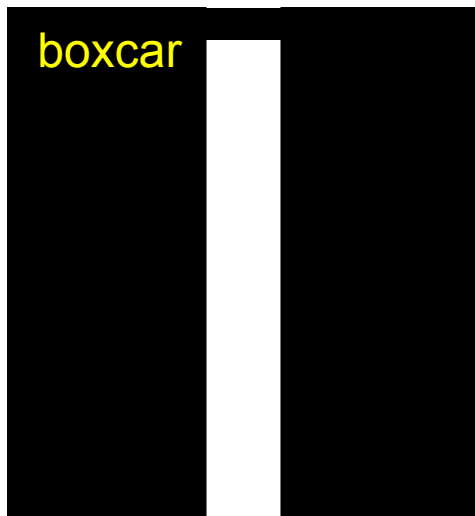
1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1



Binomialfilter und Gaußfunktion

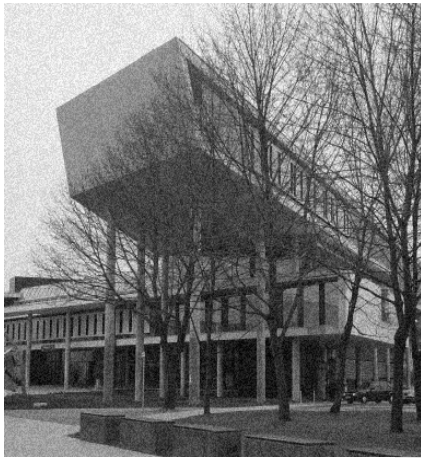
- Für immer größere Filterkerne nähert sich das Binomialfilter der **Gaußschen Glockenkurve** an

$$G(x, y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$



Filterung mit 2D Gaußfilter

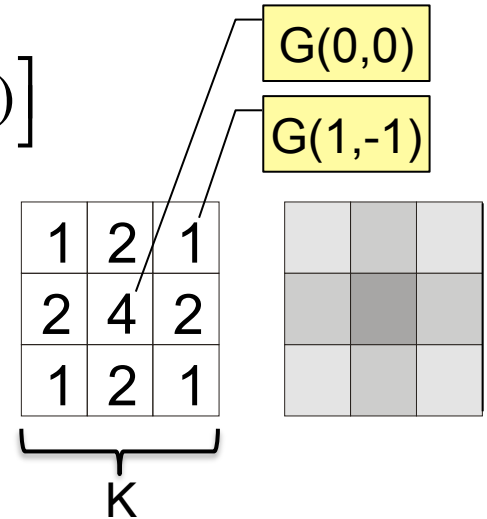
Die Gaußfunktion ist separierbar, so dass die Filterung durch zwei 1D Konvolutionen erfolgen kann



Separierbarkeit der Gaußfunktion

$$G(x, y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right] = a \exp[-b(x^2 + y^2)]$$

$$= a \exp[-bx^2] \exp[-by^2]$$



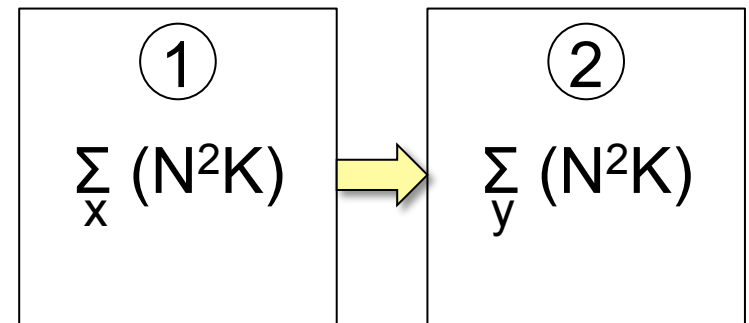
- Konvolution in lokaler Umgebung:

$$\sum_y \sum_x I_{xy} G(x, y) = \sum_y \sum_x I_{xy} a \exp[-bx^2] \exp[-by^2]$$

$$= a \underbrace{\sum_y \exp[-by^2]}_{\textcircled{2}} \underbrace{\sum_x I_{xy} \exp[-bx^2]}_{\textcircled{1}}$$

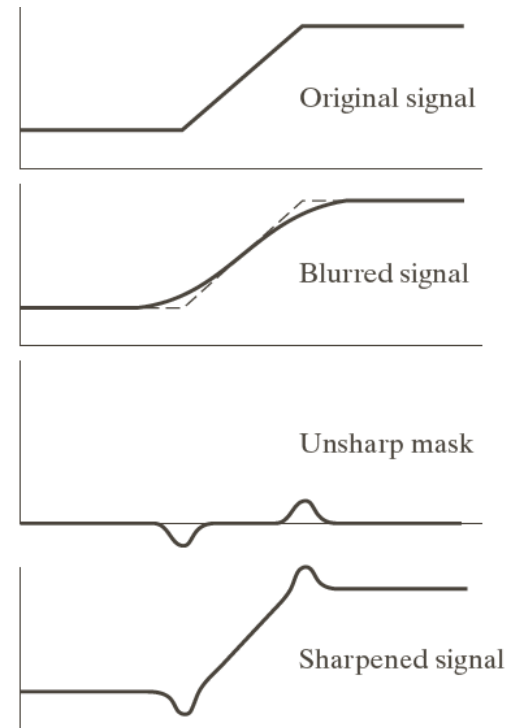
- Original: N^2K^2 Multiplikationen

- Separiert: $2N^2K$ Multiplikationen



Unscharfes Maskieren

- Originalbild unschärfer machen (Tiefpass filtern)
- Unscharfes Bild vom Original subtrahieren (ergibt die unscharfe Maske)
- Maske (nach Skalierung) zum Original addieren
- $k > 1$: Highboost filtering



Highboost filtering



Grenzen



Impulsrauschen
(Salt & Pepper
Noise) kann durch
lineare Filterung
nicht entfernt
werden



Nichtlineare Filterung




- Rauschen und Kanten haben im Frequenzbereich ähnliche Attribute
- Ist ein nichtlineares Filter denkbar, das für Rauschen und Kanteneigenschaften unterschiedlich sensitiv ist?
 - Rauschen sind räumlich gleichverteilte Grauwertvariationen
 - Grauwertvariationen an Kanten sind nicht räumlich gleichverteilt
 - Filter muss diesen Unterschied berücksichtigen

Rangordnungsfilter

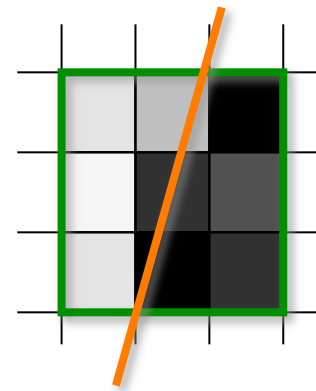
- Vorgehen
 - Sortierung der Elemente in einer Filtermaske
 - Auswahl des an einer bestimmten Stelle einsortierten Werts
 - Eintragung des ausgewählten Werts in die zentrale Position
- Eigenschaften
 - Es entstehen keine neuen Werte
 - Filter ist nichtlinear, nicht kommutativ, nicht assoziativ

Gebräuchlichstes
Rangordnungsfilter
ist das Medianfilter

26 3.	132 8.	112 5.
25 2.	102 4.	142 9.
17 1.	122 7.	117 6.

-  erster Rang
-  mittlerer Rang (Median)
-  letzter Rang

Medianfilter



- Annahmen
 1. Grauwerte auf beiden Seiten der Kante jeweils (nahezu) konstant
 2. Kantensignal größer als das Rauschsignal
 3. Kante im Filterbereich (nahezu) gerade

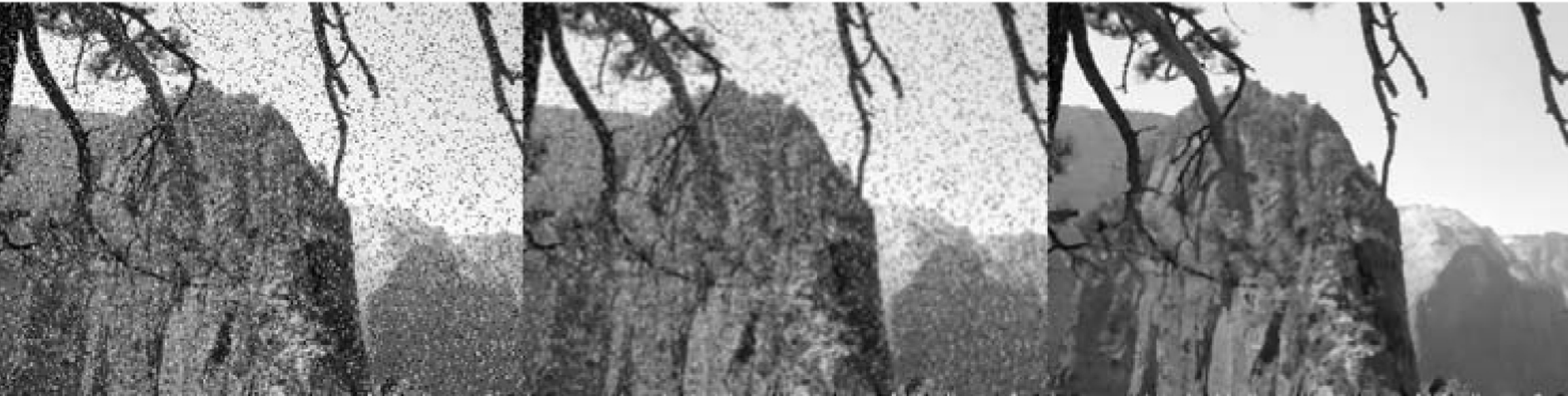
k_D (k_H) = sortierte Folge der Pixelwerte der dunkleren (helleren) Seite
- Kante verläuft durch Filterbereich
 - wegen (2) : alle k_D vor k_H
 - wegen (3): mehr k_D , falls Zentrum in k_D (und umgekehrt)
 - also Median von der Seite, zu der Pixel im Zentrum gehört (kantenerhaltend), Artefakte bei Ecken
- Keine Kante im Filterbereich
 - Median nähert sich dem Erwartungswert mit Anzahl der Stichproben (rauschunterdrückend)

Medianfilter



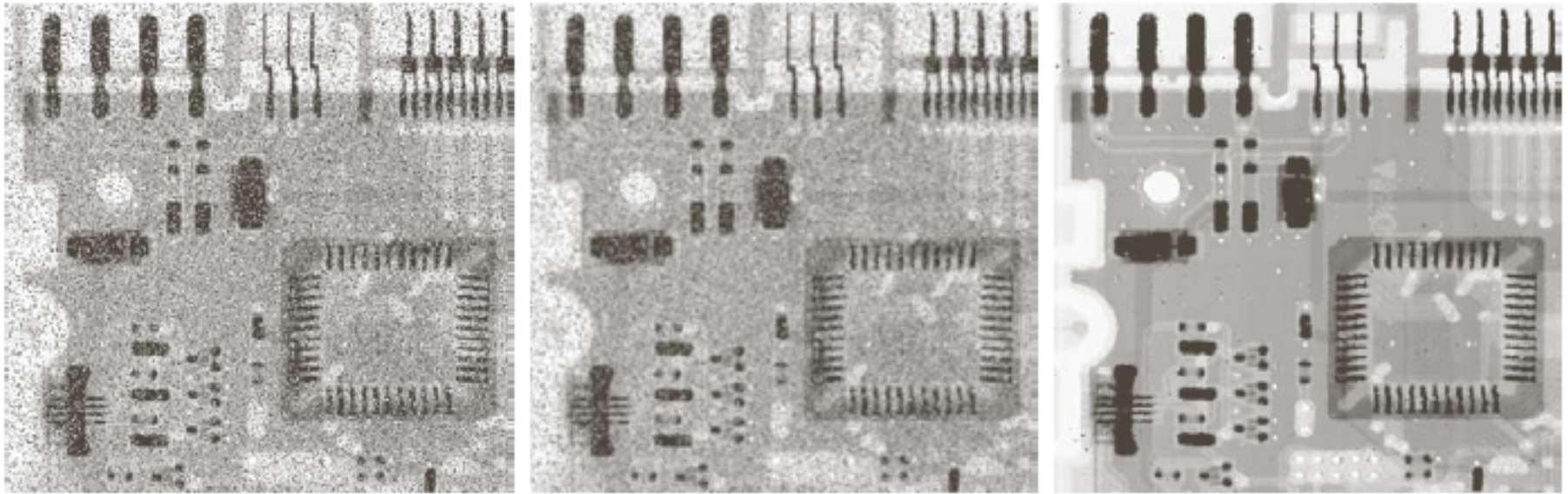
Median zur
Entfernung von
Pepper-Noise in
einer 3x3
Umgebung

Medianfilter



- Durch Medianfilterung (rechts) kann Impulsrauschen im Gegensatz zur Mittelwertfilterung (Mitte) beseitigt werden

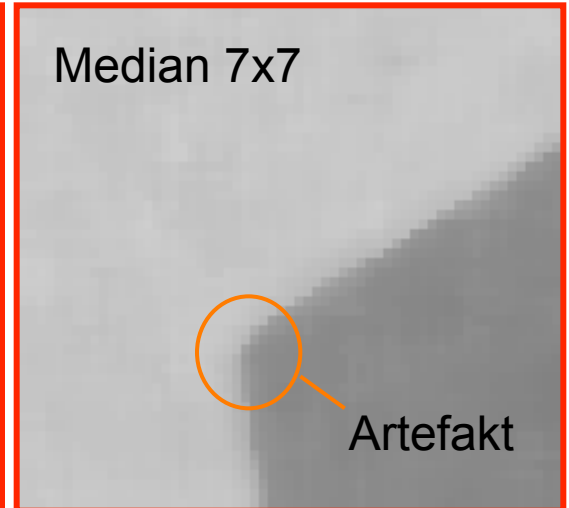
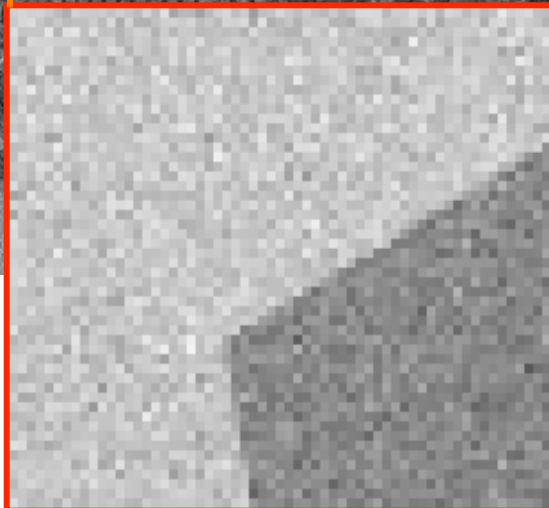
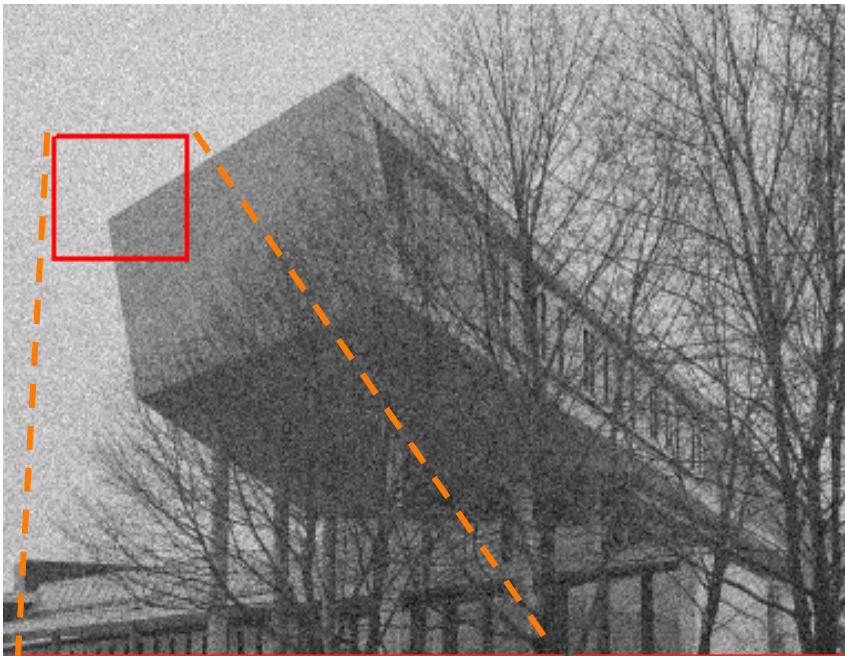
Medianfilter



a b c

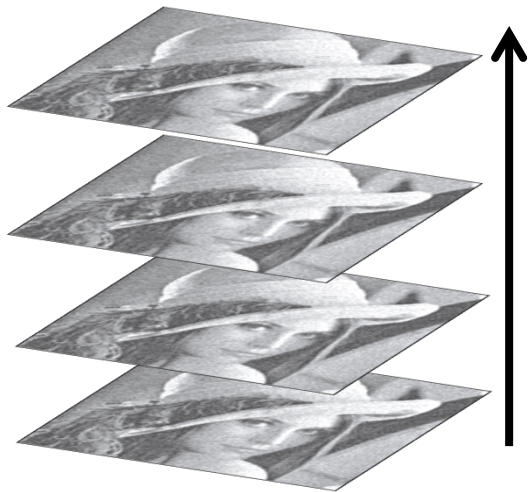
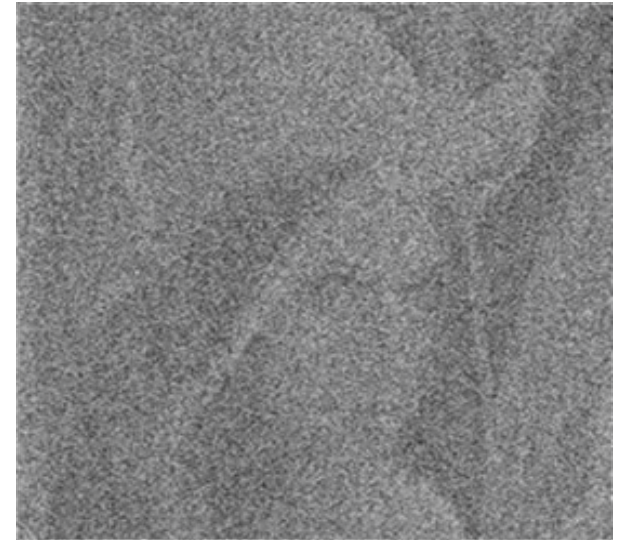
FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Vergleich Median vs. Mittelwert



Nicht-lokale Mittelwertbildung

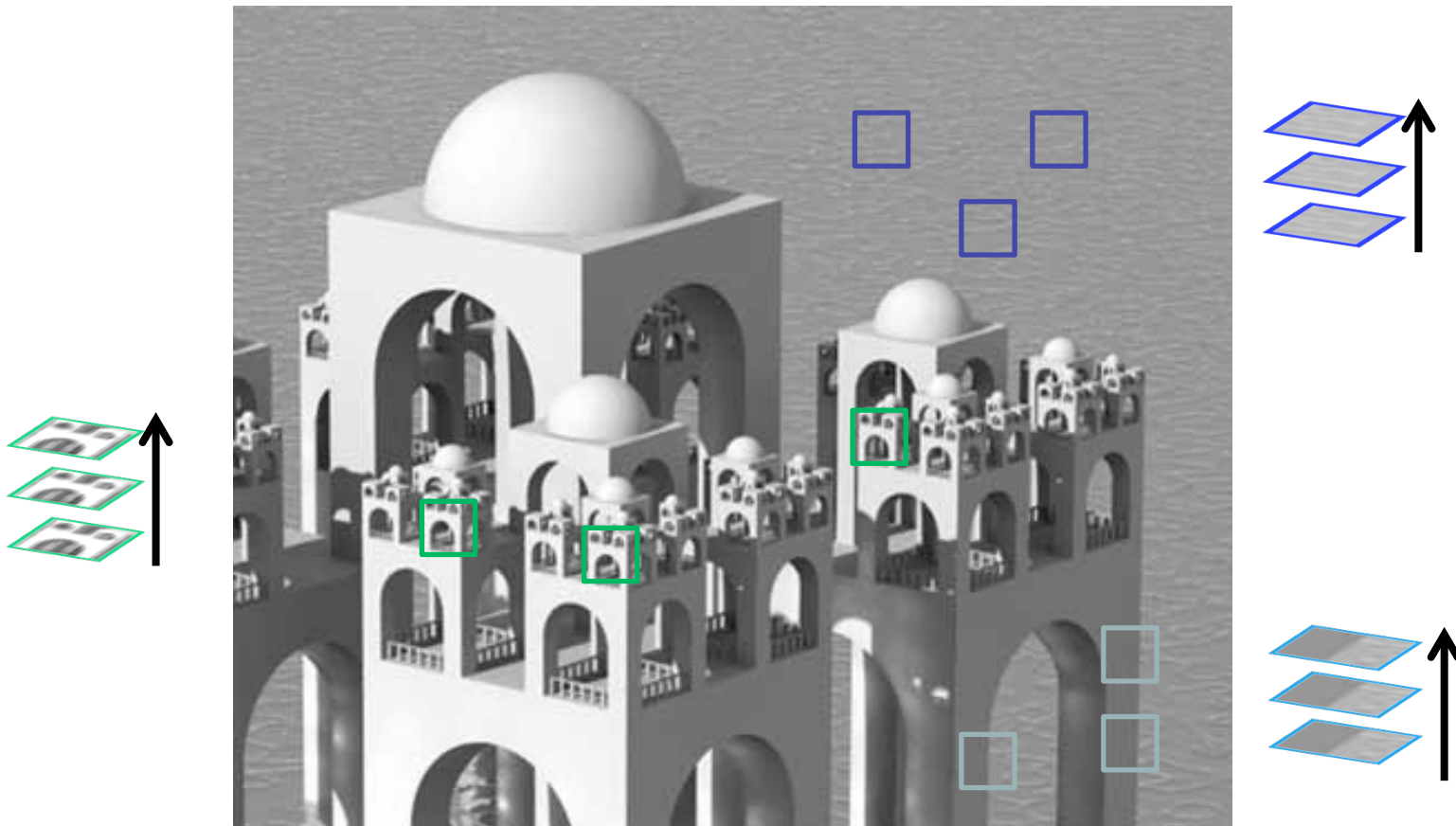
- Mehrere verrauschte Bilder einer statischen Szene
- Rauschen hat Mittelwert 0
- Mittelwert bilden



Buades, Coll, Morel: A Review of Image Denoising Algorithms, With a New One. Multiscale Modeling and Simulation, vol. 4, no. 2, pp. 490–530, 2005.
Figures from: Shahar Kovalsky, Alon Faktor: A Tour of Image Denoising, Slides

Nicht-lokale Mittelwertbildung

- Idee: Redundanz in natürlichen Bildern ausnutzen



Figures from: Shahar Kovalsky, Alon Faktor: A Tour of Image Denoising, Slides

Buades et al.: Gewichteter Mittelwert durch Selbst-Ähnlichkeit

- Verrauschtes Bild:

$$v = \{v(i) \mid i \in I\}, \quad v(i) = u(i) + n(i)$$

- Nicht-lokaler Mittelwert:

$$NL[v](i) = \sum_{j \in I} w(i, j) v(j)$$

- Gewichtsfunktion

$$w(i, j) = \frac{1}{Z(i)} \exp\left(-\frac{\|v(N_i) - v(N_j)\|_{2,a}^2}{h^2}\right)$$

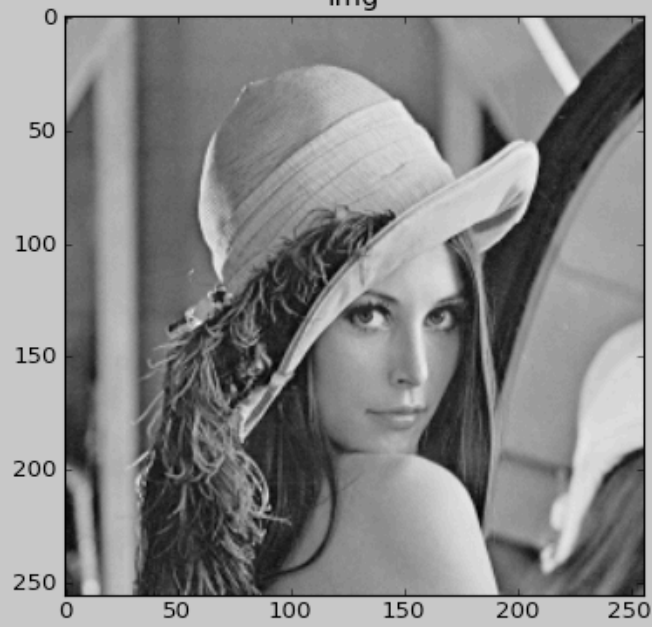
$$Z(i) = \sum_j \exp\left(-\frac{\|v(N_i) - v(N_j)\|_{2,a}^2}{h^2}\right)$$



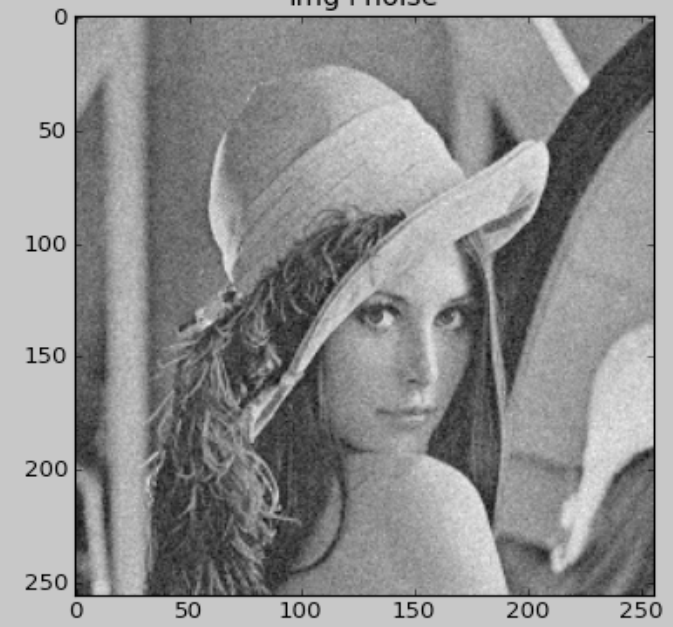
Figures from: Shahar Kovalsky, Alon Faktor:
A Tour of Image Denoising, Slides

Buades, Coll, Morel: A Review of Image Denoising Algorithms, With a New One. Multiscale Modeling and Simulation, vol. 4, no. 2, pp. 490–530, 2005.

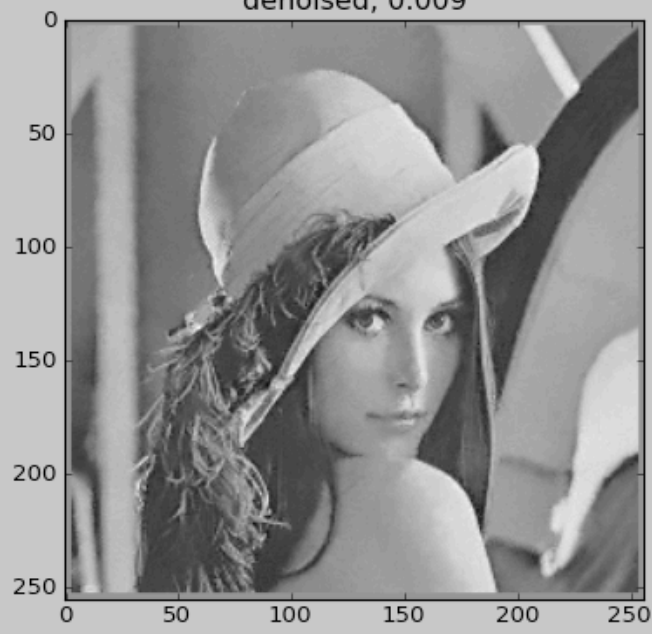
img



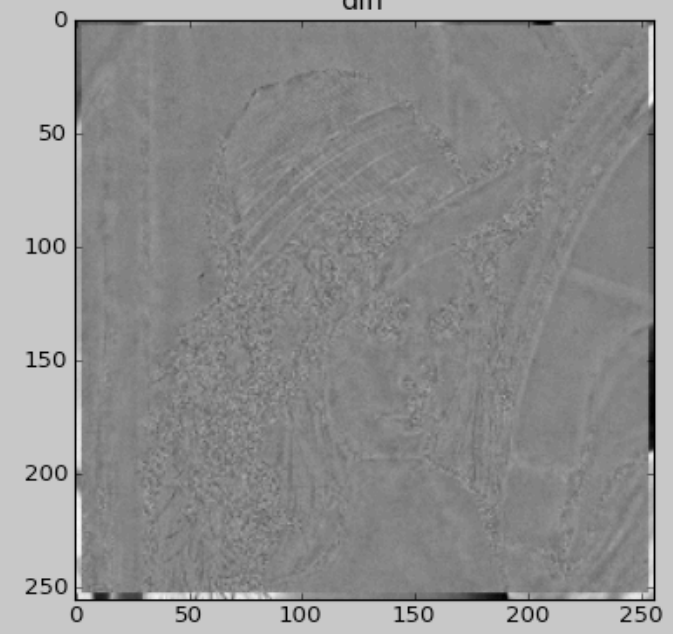
img+noise



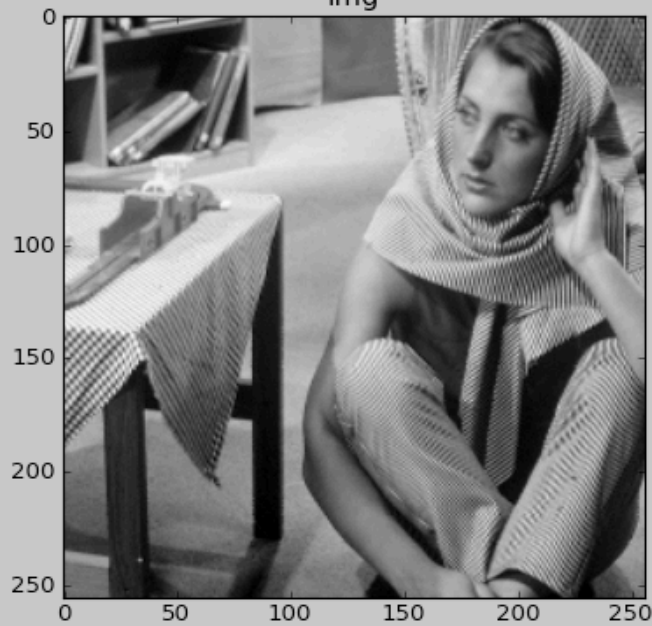
denoised, 0.009



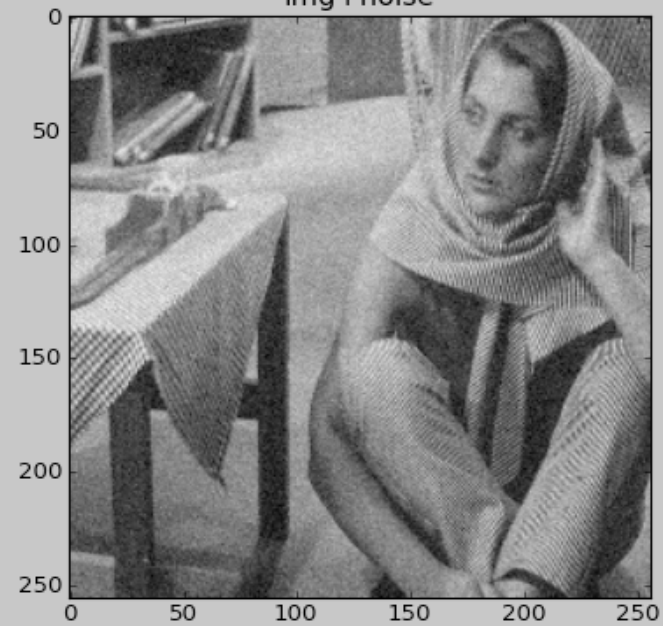
diff



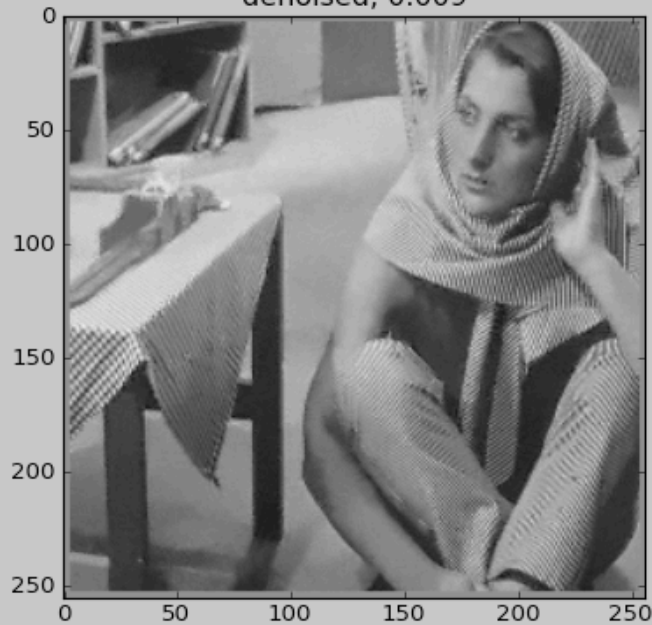
img



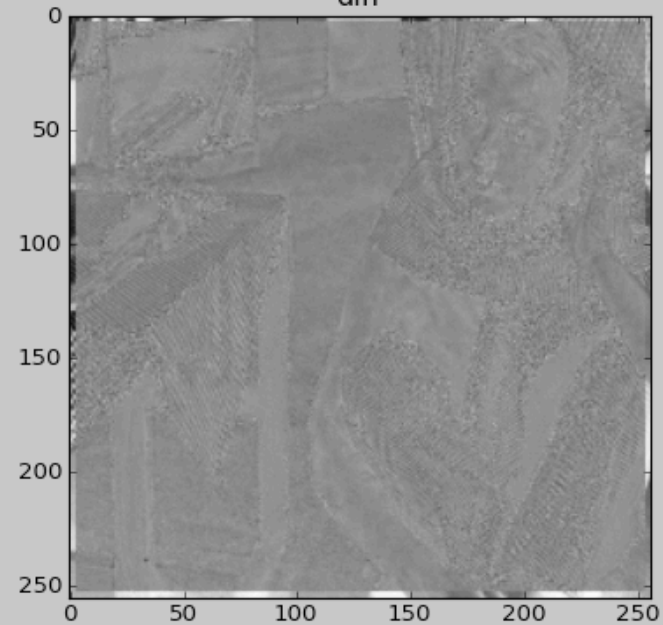
img+noise



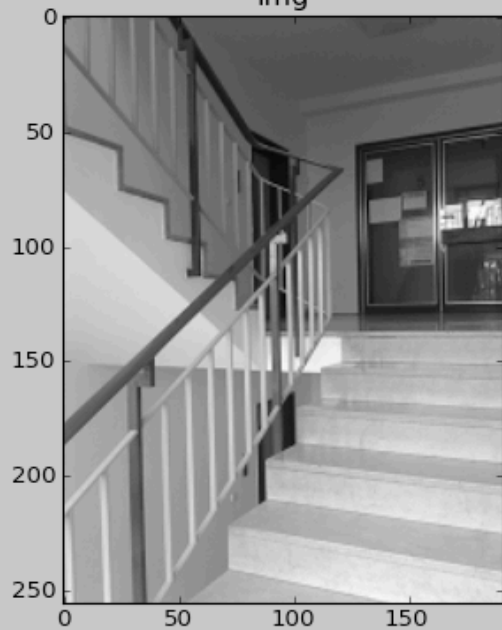
denoised, 0.009



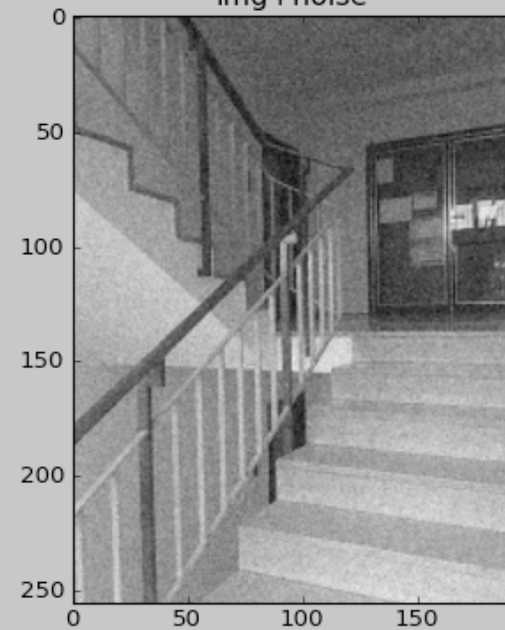
diff



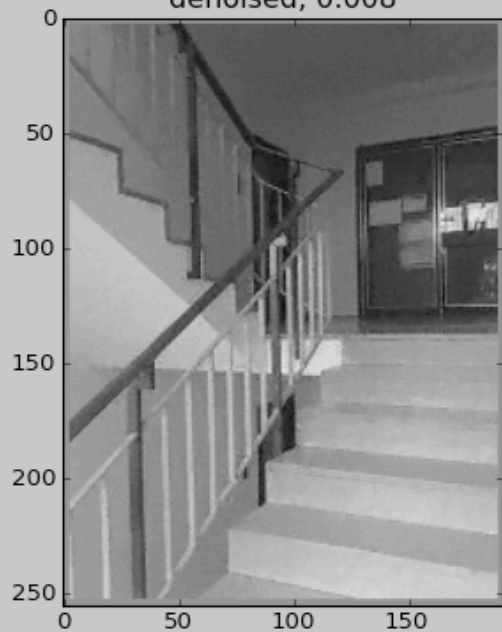
img



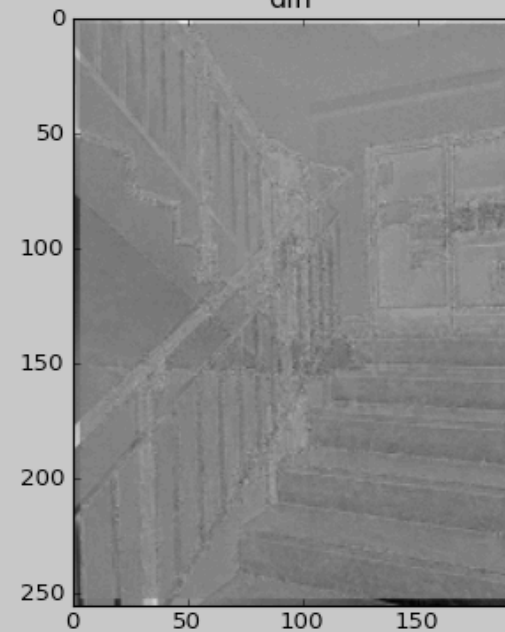
img+noise



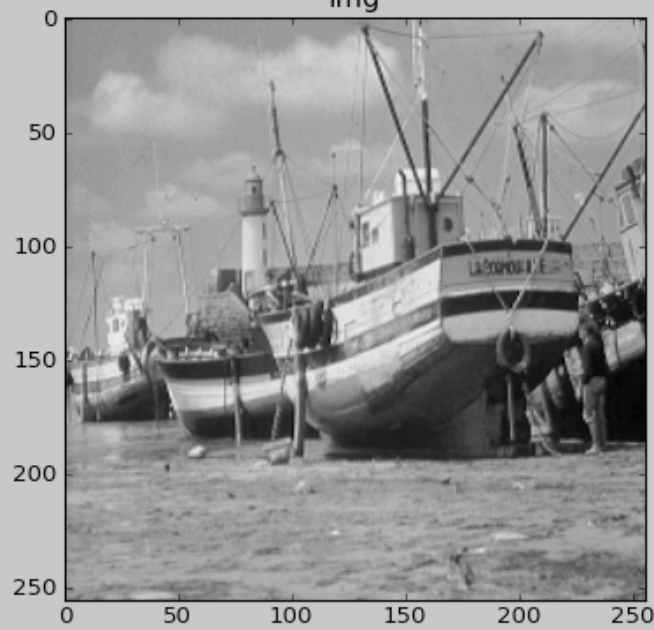
denoised, 0.008



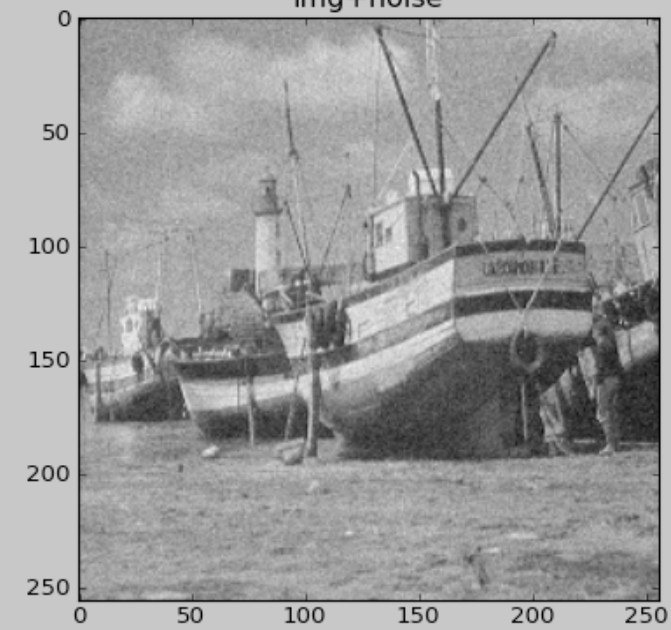
diff



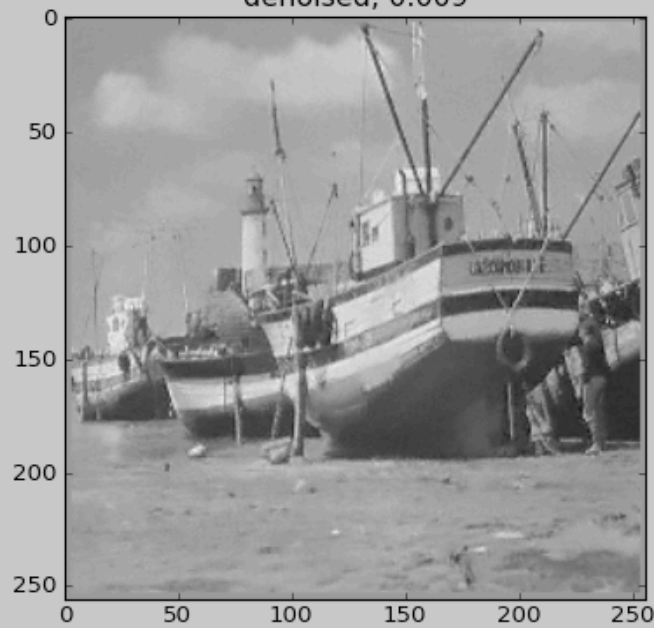
img



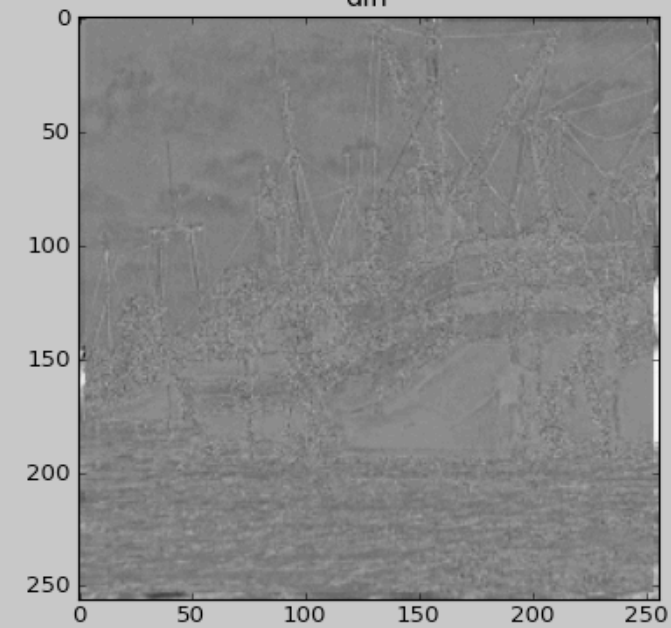
img+noise



denoised, 0.009



diff

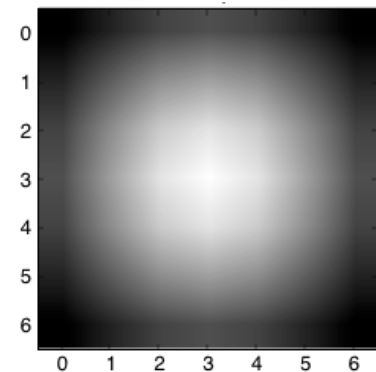


Nicht-lokaler Mittelwert in der Praxis

- Nicht-lokaler Mittelwert $NL[v](i) = \sum_{j \in I} w(i, j)v(j)$
 - In der Praxis beschränkt auf Suchfenster (z.B. Größe 21x21)

- Gewichtsfunktion $w^*(i, j) = \exp\left(-\frac{\|v(N_i) - v(N_j)\|_{2,a}^2}{h^2}\right)$

- Nachbarschaft N_i (z.B. Größe 7x7)
- Differenz gewichtet mit Gaußfunktion
- h abhängig von Stärke des Rauschens

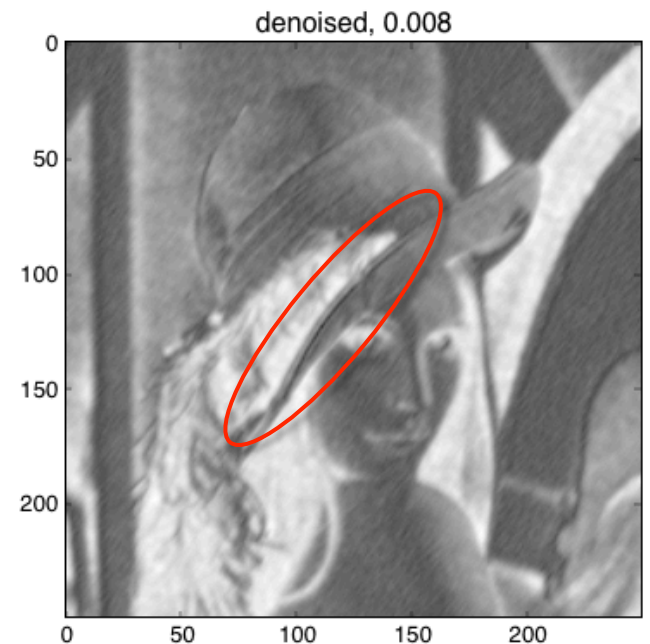
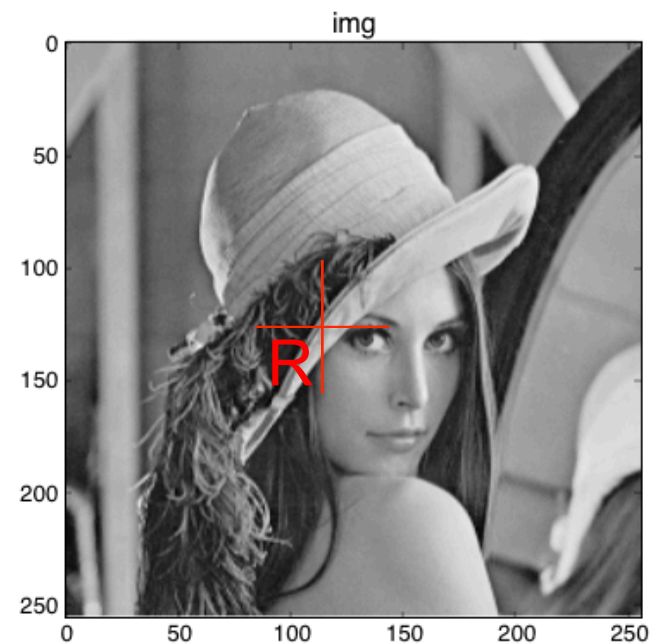


Buades, Coll, Morel: A Review of Image Denoising Algorithms, With a New One. Multiscale Modeling and Simulation, vol. 4, no. 2, pp. 490–530, 2005.

Gewichtsfunktion

- Gewichtsfunktion
 - Differenz gewichtet mit Gaußfunktion
 - Vergleich aller 7x7-patches mit Referenz-patch R

$$\left\| v(N_i) - v(N_j) \right\|_{2,a}^2$$

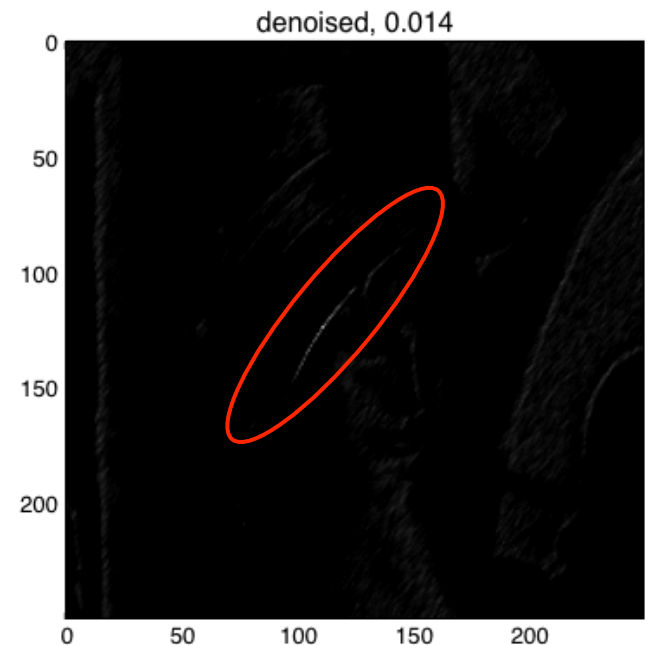
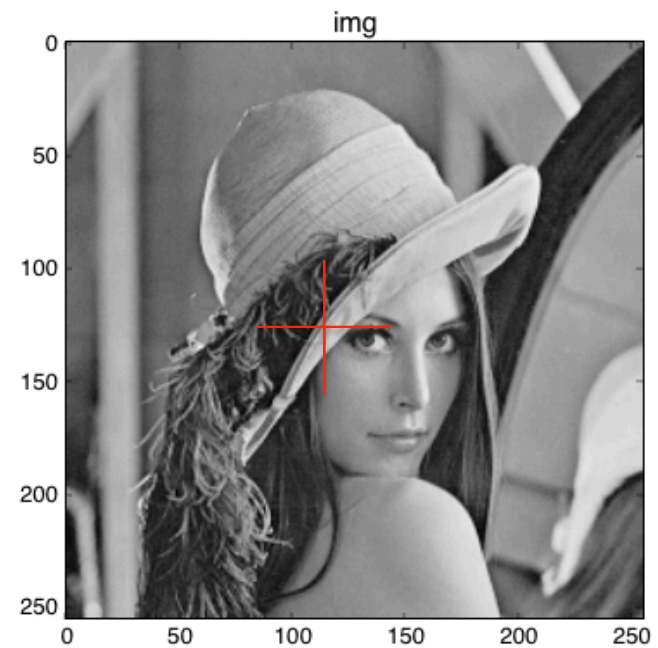


Gewichtsfunktion

- Gewichtsfunktion
 - Differenz gewichtet mit Gaußfunktion
- Exp-Funktion:
 - Maximum bei Differenz 0: $\exp(0) = 1$
 - sehr schneller Abfall mit zunehmender Differenz

$$w^*(i, j) = \exp\left(-\frac{\|v(N_i) - v(N_j)\|_{2,a}^2}{h^2}\right)$$

- Normalisierung: $w(i, j) = \frac{w^*(i, j)}{\sum_j w^*(i, j)}$



TRANSFORMATION UND INTERPOLATION

Transformation und Interpolation

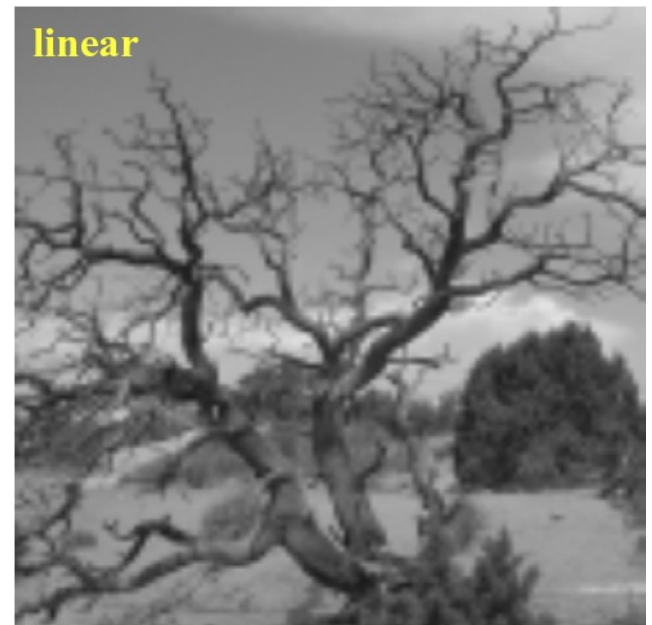
- Die geometrischen Transformationen Translation, Rotation und Skalierung sind auf reellen Zahlen definiert:

$$Rot_{\alpha} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad Tr_{dx,dy} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + dx \\ y + dy \end{pmatrix}, \quad Sc_s \begin{pmatrix} x \\ y \end{pmatrix} = s \begin{pmatrix} x \\ y \end{pmatrix}$$

- Digitale Bilder haben ganzzahligen Definitionsbereich
 - Definitionsbereich = diskrete Pixel
- Nach Transformation ist eine Interpolation der zugehörigen Intensitätswerte notwendig
 - nächster Nachbar
 - bilineare Interpolation
 - bikubische Interpolation

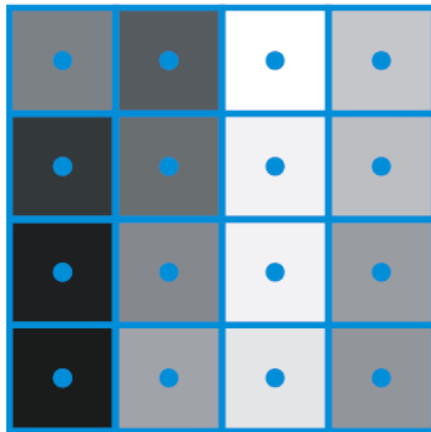
Interpolation

- Konstante Interpolation (Wert des nächsten Nachbarpixels)
- Lineare Interpolation
- Interpolation durch Polynome höheren Grades

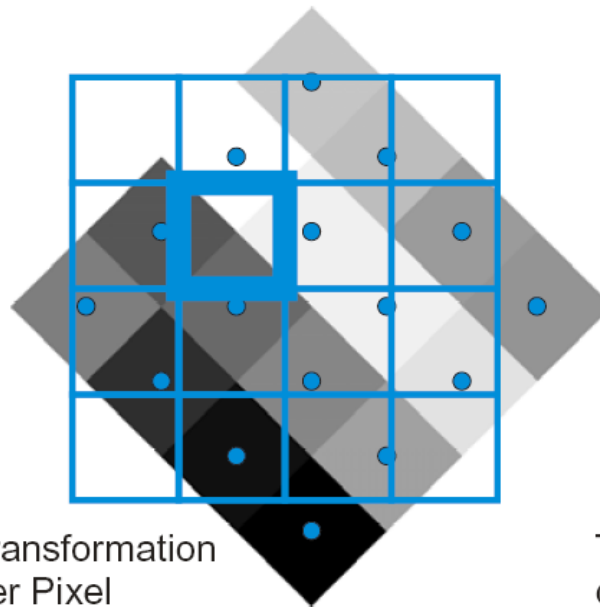


Konstante Interpolation

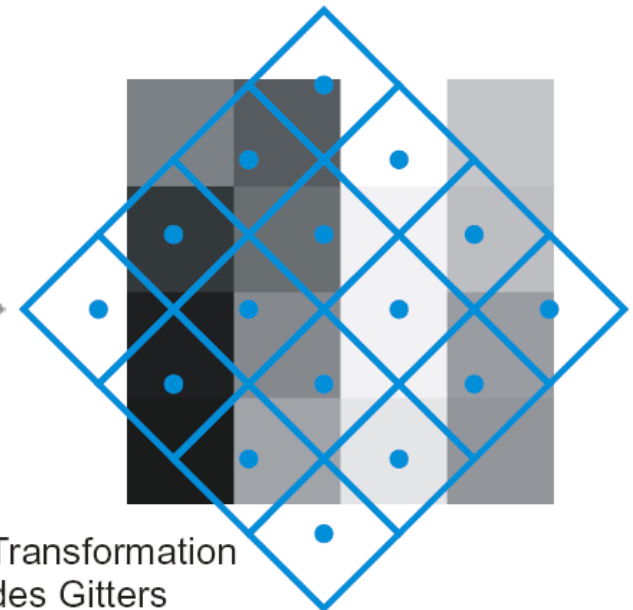
- inverse Transformation: berechne für jedes Pixel im transformierten Bild die Position im Ursprungsbild



Transformation
der Pixel

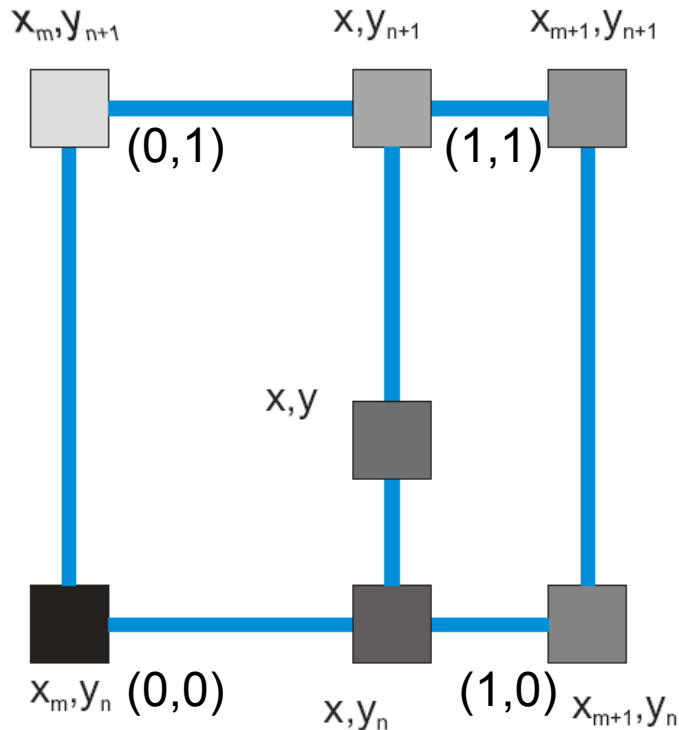


Transformation
des Gitters



Bilineare Interpolation

- Verwendung der 4 nächsten Nachbarn



Erster Schritt:

$$g_1(x_m, y) = \frac{y_{n+1} - y}{y_{n+1} - y_n} f(x_m, y_n) + \frac{y - y_n}{y_{n+1} - y_n} f(x_m, y_{n+1}),$$

$$g_1(x_{m+1}, y) = \frac{y_{n+1} - y}{y_{n+1} - y_n} f(x_{m+1}, y_n) + \frac{y - y_n}{y_{n+1} - y_n} f(x_{m+1}, y_{n+1}).$$

Zweiter Schritt:

$$g(x, y) = \frac{x_{m+1} - x}{x_{m+1} - x_m} g_1(x_m, y) + \frac{x - x_m}{x_{m+1} - x_m} g_1(x_{m+1}, y).$$

- falls die Punkte $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$ vorliegen:

$$g(x,y) \approx (1-x)g_1(0,y) + x g_1(1,y) = \dots$$

Bilineare und Bikubische Interpolation

- Bilineare Interpolation: die 4 nächsten Nachbarn

$$f(x, y) \approx a + bx + cy + dxy$$

- Bestimmung der Koeffizienten?

- Bikubische Interpolation: die 16 nächsten Nachbarn

$$f(x, y) \approx \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$$