

# Kapitel 3: Filter im Ortsraum

Prof. Ingrid Scholl  
Bildverarbeitung WS 2018/2019

# 3. Kapitel: Filter im Ortsraum

---

- Was ist ein Filter
- Glättungsfilter (Summenfilter)
- Kantenfilter (Differenzenfilter)
- Nicht lineare Filter

# Was ist ein Filter (im Ortsraum)?

**Beispiel:** Glättung eines Bildes f

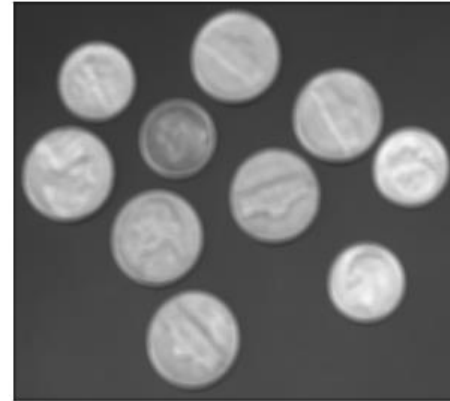


**Glättung:**  
Berechnung eines  
Mittelwertes aus  
einer ROI

Größe des Filters bestimmt das räumliche Ausmaß der ROI  
(z.Bsp.  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ , ...,  $(2n+1) \times (2m+1)$ ), muß nicht  
quadratisch sein

Filterkoeffizienten werden je nach Anwendung berechnet

# Was ist ein Filter (im Ortsraum)?



**Glättung:**  
Berechnung eines  
Mittelwertes aus  
einer (5x5)-ROI

- Eine Filterung ist keine Punktoperation.
- Ein 2D-Filter berücksichtigt eine Menge von benachbarten Pixeln um den Bildpunkt  $(x,y)$ . Diese Nachbarschaft (ROI) ist i.d.R. um den Bildpunkt  $(x,y)$  herum und hat eine Größe von  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$  und allgemein  $(2n+1) \times (2m+1)$ .
- Der Filter besteht aus Filterkoeffizienten. Diese werden je nach Anwendung definiert.
- Die Filterung wird für jeden Bildpunkt  $(x,y)$  ausgeführt und liefert einen gefilterten Wert, der in einem neuen Bild an der Stelle  $(x,y)$  gespeichert wird.

# Faltung – Beispiel Glättung

Filtermatrix  $h$  der Größe  $(2k+1) \times (2k+1)$  wird auf jedes Pixel  $(x,y)$  im Bild  $f$  angewendet:

**Faltung**

$$(f * h)(x, y) = \sum_{v=-k}^k \sum_{u=-k}^k f(x-u, y-v) \cdot h(u, v)$$

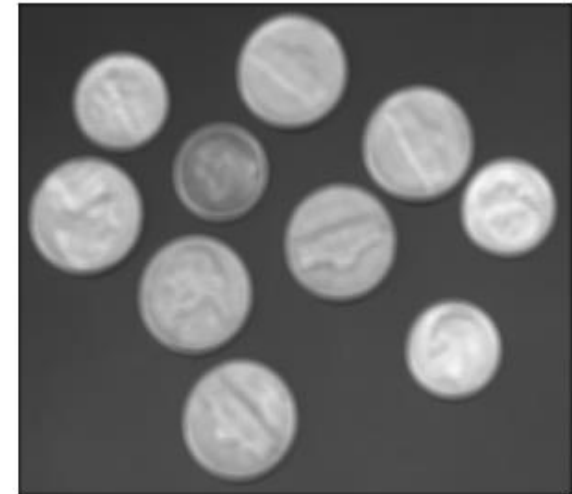
Nachteil: gefiltertes Bild ist kleiner  
(Randprobleme der Dicke  $k$ )



Original  $f$

$$* \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} =$$

5x5-Glättungs-  
filter  $h$



Geglättetes Bild  
 $f * h$

# Faltung (Konvolution) im Ortsraum

Faltung erzeugt ein neues Bild  $f_{neu}$  durch eine gewichtete Summe von Bildelementen in  $f$ :

$$f_{neu} = (f * h)(x, y) = \sum_{v=-k}^k \sum_{u=-k}^k f(x-u, y-v) \cdot h(u, v)$$

Originalbild      Gewichtungsfunktion  
Filtermaske

Die Gewichtungsfunktion  $h$  heißt **Faltungsfunktion**  
(oder **Konvolutionsfunktion** oder auch **Filtermaske**)

\* ist der **Faltungsoperator**

u=	-1	0	1	v=
				-1
		x		0
				1

# Woher kommt die Rotation in der Faltung?

Berechnen Sie die Faltung eines 3x3 Bildes  $f$  mit einer 3x3 Filtermaske  $h$  für den Bildpunkt  $x=2$  und  $y=2$ :

$$f_{neu} = (f * h)(x + k + 1, y + k + 1) = \sum_{v=1}^{2k+1} \sum_{u=1}^{2k+1} f(x + k + 1 - u, y + k + 1 - v) \cdot h(u, v)$$

$$\left( \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} * \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \right) (2,2) = \sum_{v=1}^3 \sum_{u=1}^3 f(x + k + 1 - u, y + k + 1 - v) \cdot h(u, v)$$

mit  $x=2, y=2, k=1$   
bei einem 3x3 Filter  $h$



$$\begin{aligned} &= \sum_{v=1}^3 \sum_{u=1}^3 f(2 + 2 - u, 2 + 2 - v) \cdot h(u, v) \\ &= \sum_{v=1}^3 \sum_{u=1}^3 f(4 - u, 4 - v) \cdot h(u, v) \end{aligned}$$

# Woher kommt die Rotation in der Faltung?

$$\left( \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} * \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \right) (x, y) = \sum_{v=1}^3 \sum_{u=1}^3 f(x+2\underbrace{-u}_{\text{red box}}, y+2\underbrace{-v}_{\text{red box}}) \cdot h(u, v)$$

Bewirkt in der Formel  
die Drehung um 180°

$$h = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

$$h_{180^\circ} = \begin{pmatrix} h_{33} & h_{32} & h_{31} \\ h_{23} & h_{22} & h_{21} \\ h_{13} & h_{12} & h_{11} \end{pmatrix}$$

$$= \sum_{v=1}^3 \sum_{u=1}^3 f(4-u, 4-v) \cdot h(u, v)$$

$$= \sum_{v=1}^3 f(3, 4-v) \cdot h(1, v) + f(2, 4-v) \cdot h(2, v) + f(1, 4-v) \cdot h(3, v)$$

$$\begin{aligned} &= f(3,3) \cdot h(1,1) + f(2,3) \cdot h(2,1) + f(1,3) \cdot h(3,1) \\ &+ f(3,2) \cdot h(1,2) + f(2,2) \cdot h(2,2) + f(1,2) \cdot h(3,2) \\ &+ f(3,1) \cdot h(1,3) + f(2,1) \cdot h(2,3) + f(1,1) \cdot h(3,3) \end{aligned}$$

$$\begin{aligned} &= f_{11} \cdot h_{33} + f_{12} \cdot h_{32} + f_{13} \cdot h_{31} \\ &+ f_{21} \cdot h_{23} + f_{22} \cdot h_{22} + f_{23} \cdot h_{21} \\ &+ f_{31} \cdot h_{13} + f_{32} \cdot h_{12} + f_{33} \cdot h_{11} \end{aligned}$$

## Praktische Anwendung der Faltung:

Hier kann man erkennen, dass der Filter  $h$  um 180° gedreht auf das Bild gelegt wird und die übereinanderliegenden Bild- und Filterkoeffizienten miteinander multipliziert und aufsummiert werden.



# Beispiel: Faltung berechnen

```
>> h180 = imrotate(h, 180)
```

```
h180 =
```

5	6	7
4	9	8
3	2	1

```
>> I.*h180
```

```
ans =
```

50	300	560
48	459	656
42	104	85

```
>> sum(sum(ans))
```

```
ans =
```

2304

```
>> h = [1 2 3; 8 9 4; 7 6 5]
```

```
h =
```

1	2	3
8	9	4
7	6	5

```
>> I = [10 50 80; 12 51 82; 14 52 85]
```

```
I =
```

10	50	80
12	51	82
14	52	85

```
>> f = imfilter(I,h,'conv')
```

f =	565	1350	1237
	1006	2304	1998
	971	2144	1720

**Normalerweise ist h normiert,  
d.h.  $\text{sum}(\text{sum}(h)) = 1.0$**

## Randproblem:

Durch die Faltung mit einem Filter gibt es an den Bildrändern Probleme, da die Berechnung der Faltung davon ausgeht, daß die Filtermaske mittig auf das zu faltende Pixel gesetzt wird.

## Lösung des Randproblems:

- Bild wird um 2k Spalten und 2k Zeilen kleiner
- Randbereiche werden auf konstantem Wert gesetzt
- Randbereiche erhalten den Originalwert
- Bild wird periodisch fortgesetzt

# Lineare Filter in MATLAB

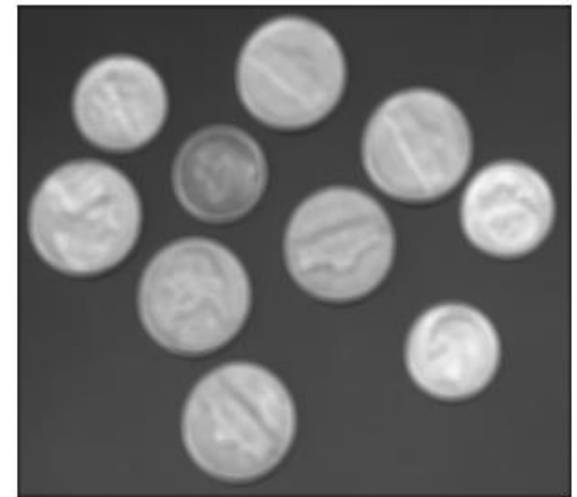
```
I = imread('coins.png'); % Einlesen des Bildes coins.png in I
h = ones(5,5) / 25;      % erzeugt einen 5x5-Filter mit Koeff.=1/25
I2 = imfilter(I,h, 'conv'); % Faltung des Bildes I mit dem Filter h
imshow(I), title('Original Image'); % Bild anzeigen
imshow(I2), title('Geglaettes Bild');
```



Original

$$* \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} =$$

5x5-Glättungs-  
filter



Geglättetes Bild

# Eigenschaften der Faltung

Faltungsoperator  $*$ , Funktionen  $f_1$  und  $f_2$ , Skalare  $a_1$ ,  $a_2$ ,  $a$  und  $b$

Linear: 
$$h * (a_1 f_1 + a_2 f_2) = a_1 \cdot h * f_1 + a_2 \cdot h * f_2$$

Verschiebungsinvariant (ortsunabhängig):

$$h * f(m + a, n + b) = [h * f](m + a, n + b)$$

Kommutativ (Reihenfolge nicht wichtig):

$$[h_1 * h_2](m, n) = [h_2 * h_1](m, n)$$

Assoziativ (steigert die Effizienz):

$$h_1 * ([h_2 * h_3](m, n)) = [h_1 * h_2](m, n) * h_3(m, n)$$

Assoziativ (steigert die Effizienz):

$$h_1 * ([h_2 * f](m, n)) = [h_1 * h_2](m, n) * f(m, n)$$

Seien  $h_1, h_2$  Filter der Größe  $K \times K$  und  $f$  ein Bild der Größe  $M \times N$ .  
Wieviele Multiplikationen fallen nach dem Assoziativgesetz an?

$$h_2 * f \rightarrow K \cdot K \cdot M \cdot N = K^2 \cdot M \cdot N$$

$$h_1 * (h_2 * f) \rightarrow K^2 \cdot M \cdot N + K^2 \cdot M \cdot N = 2K^2 \cdot M \cdot N$$

$$h_1 * h_2 \rightarrow K \cdot K \cdot K \cdot K = K^2 \cdot K^2 = K^4$$

$$(h_1 * h_2) * f \rightarrow K^4 + K^2 \cdot M \cdot N$$

$$K^4 + K^2 \cdot M \cdot N < 2K^2 \cdot M \cdot N$$

$$K^4 < K^2 \cdot M \cdot N$$

$$K^2 < M \cdot N$$

$$\text{Filtergrösse} \ll \text{Bildgrösse}$$

- Rechteckfilter:  
Mittelwertsfilter:

$$h = h_{col} \cdot h_{row} = \frac{1}{3} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \frac{1}{3} (1 \quad 1 \quad 1) = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Gauß- bzw. Binomialfilter:

$$h = h_{col} \cdot h_{row} = (B_n)^T \cdot B_n$$

$$h = h_{col} \cdot h_{row} = (B_2)^T \cdot B_2 = \frac{1}{4} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \cdot \frac{1}{4} (1 \quad 2 \quad 1) = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- Binomialfilter und Rechteckfilter sind **separabel**, d.h. anstatt einer 2D-Filterung kann man zwei 1D-Filterungen ausführen:

$$B_2 \cdot (B_2)^T = \frac{1}{4} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \cdot \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- Schnelle Berechnung durch Ausnutzung des separablen Filters:**  $(f * B_2) * B_2^T$

Normierungsfaktoren für Binomialfilter  $B_n$  der Ordnung  $n$  sind  $2^{-n}$  für 1D-Filter und  $2^{-2n}$  für 2D-Filter. Anstatt einer Abbildung auf Fließkommazahlen, Filterung und Division kann die Filterung vollständig in ganzen Zahlen durchgeführt werden. Die Division wird dann als Shift-Operation um  $n$  bzw.  $2n$  Stellen durchgeführt.

# Gauß- und Binomialfilter

n	f	Binomialkoeffizienten (Pascal'sches Dreieck)	$\sigma^2$
0	1	1	0
1	1/2	1 1	1/4
2	1/4	1 2 1	1/2
3	1/8	1 3 3 1	3/4
4	1/16	1 4 6 4 1	1
5	1/32	1 5 10 10 5 1	5/4
6	1/64	1 6 15 20 15 6 1	3/2

$$h = h_{col} \cdot h_{row} = B_n \cdot (B_n)^T$$

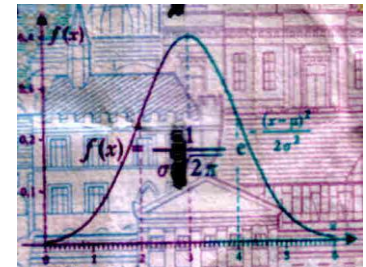
**Beispiel:** n=2

$$\leftarrow (B_2)^T = \frac{1}{4} \cdot \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

$$B_2 = \frac{1}{4} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

**1D-Gaußkurve:**  
 $n \rightarrow \infty$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \cdot \left(\frac{x-\mu}{\sigma}\right)^2}$$





# 2D Gaußkurve

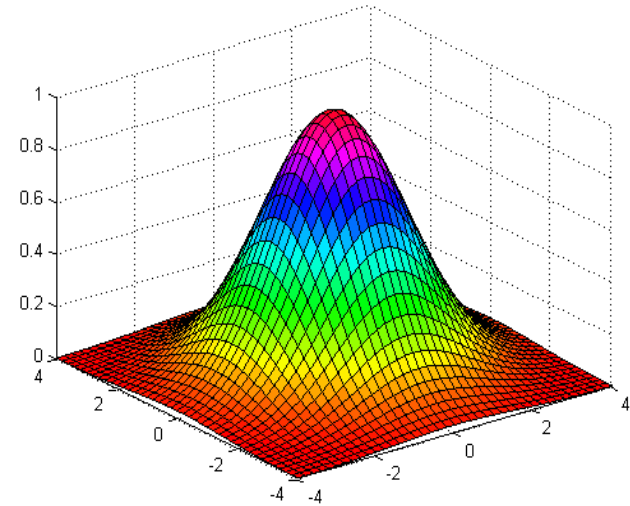
- 2D-Gaußfunktion entsteht durch die Matrixmultiplikation zweier 1D-Gaußfunktionen mit  $\eta = 0$ , somit auch separabel

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

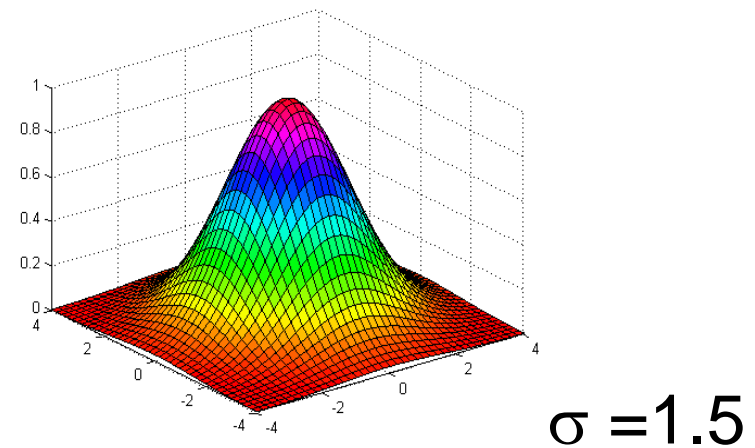
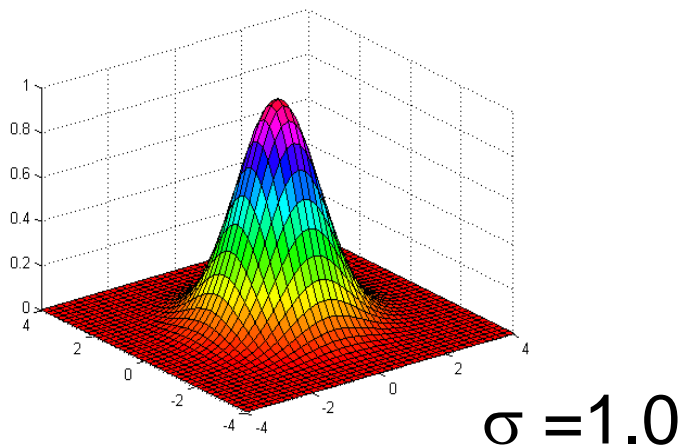
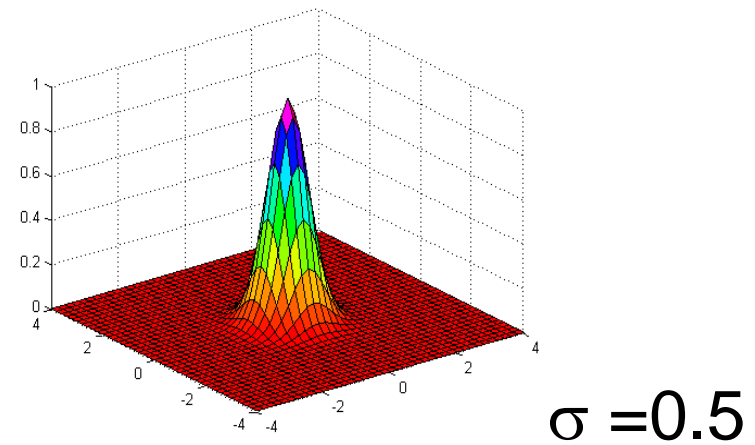
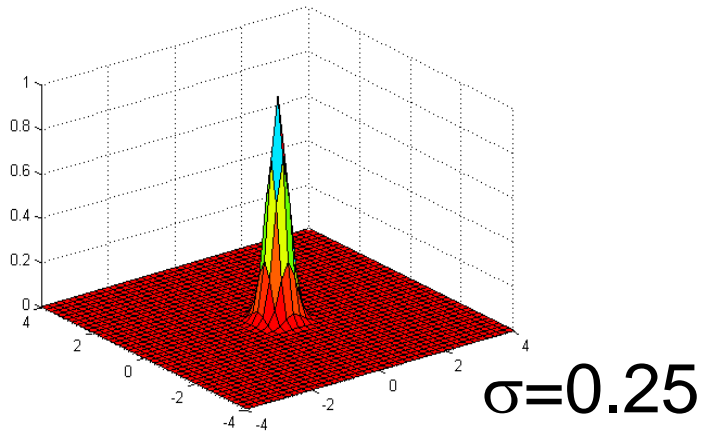
- Filtergröße  $K \times K$  mit:

$$K = 2 \cdot \lceil 3\sigma \rceil + 1$$

$\sigma$  bestimmt den Radius der glockenförmigen Gaußfunktion



# 2D-Gausskurven



# MATLAB: Filter erzeugen und Bild filtern

```
x = [1 4 6 4 1];           % Zeilenvektor
y = [1; 4; 6; 4; 1];       % Spaltenvektor
h = y*x                     % Spaltenvektor*Zeilenvektor
h = h/(16*16)
sum(h)
sum(sum(h))                % Kontrolle: Ergebnis = 1.0

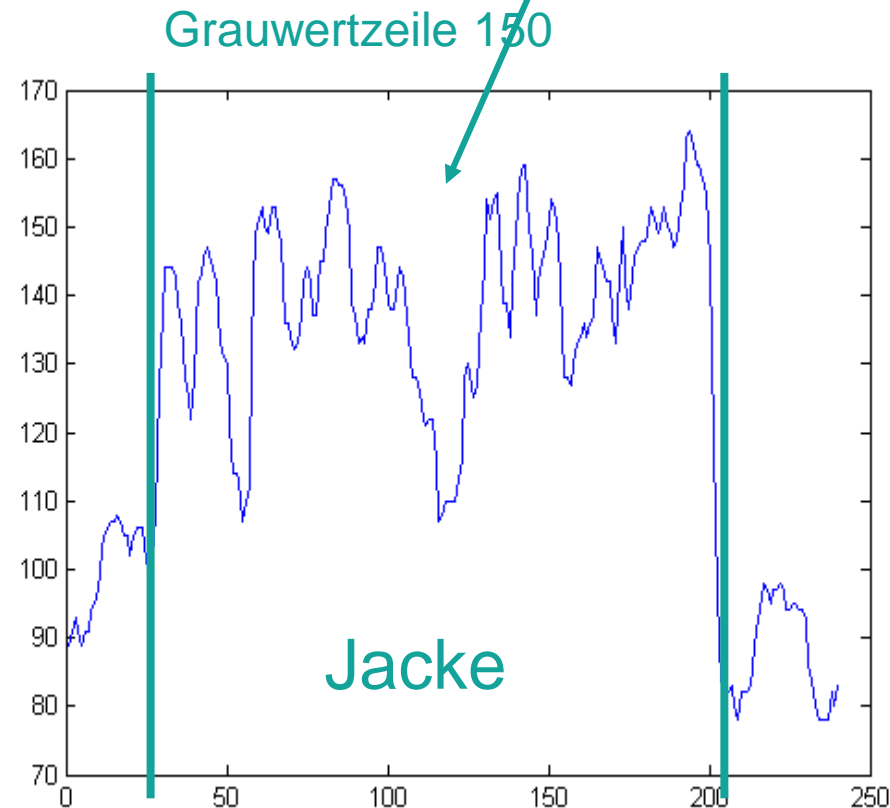
% Faltung Bild I mit Filter h, 3.ter Parameter 'conv'
% steht für "convolution" = Faltung
I3 = imfilter(I,h,'conv');

figure; imshow(I3);        % Anzeigen des Bildes
```

# Kantenextraktion

Logische Kette: „*Kanten erkennen bedeutet ...*“

- ⇒ Grauwertunterschiede
- ⇒ Änderungen der Grauwerte
- ⇒ Grauwertdifferenzen
- ⇒ Grauwertdifferenzen mit verschiedenen Richtungen
- ⇒ Richtungsabhängige Ableitungen in Bildern



# Kanten- und Liniendetektion durch lineare Filter

- Glättungsfilter unterdrücken hohe Grauwertunterschiede
- Kantenfilter verstärken die hohen Grauwertunterschiede (kleine Strukturen)
- Kantenfilter basieren auf der n-ten Ableitung. Hier: Ableitungsfilter 1. Ordnung im Diskreten:

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x, y) - f(x - \Delta x, y)}{\Delta x}$$

Rückwärtsgradient

$$\approx \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

Vorwärtsgradient

$$\approx \frac{f(x + \Delta x, y) - f(x - \Delta x, y)}{2\Delta x}$$

Symmetrischer Gradient



# Kanten- und Liniendetektion durch lineare Filter

- **Beispiel:** Erzeuge einen symmetrischen Gradienten, der vertikale Kanten erkennt:

$$\begin{aligned}\frac{\partial f(x, y)}{\partial x} &\approx \frac{f(x + \Delta x, y) - f(x - \Delta x, y)}{2\Delta x} \\ &\approx \frac{f(x + 1, y) - f(x - 1, y)}{2} \\ &\approx \frac{1}{2} (1 \cdot f(x + 1, y) + 0 \cdot f(x, y) + (-1) \cdot f(x - 1, y))\end{aligned}$$

Symmetrischer Gradient  $h = \frac{1}{2} \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$

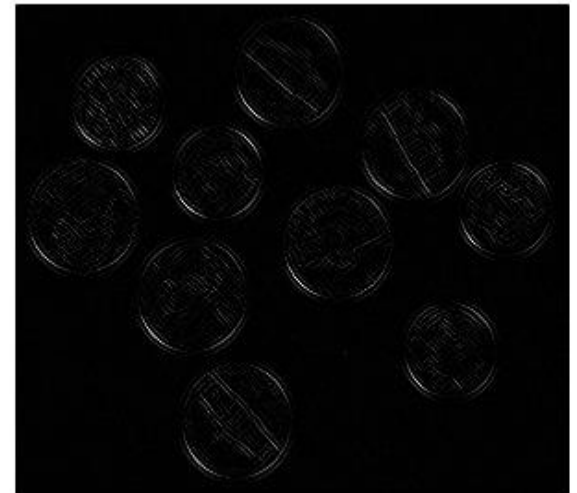


Laplacefilter basiert auf der 2.ten Ableitung:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial}{\partial x} \frac{\partial f}{\partial x} \approx \frac{f(x + \Delta x) - f(x) - [f(x) - f(x - \Delta x)]}{(\Delta x)^2} \\ &= \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{(\Delta x)^2}\end{aligned}$$

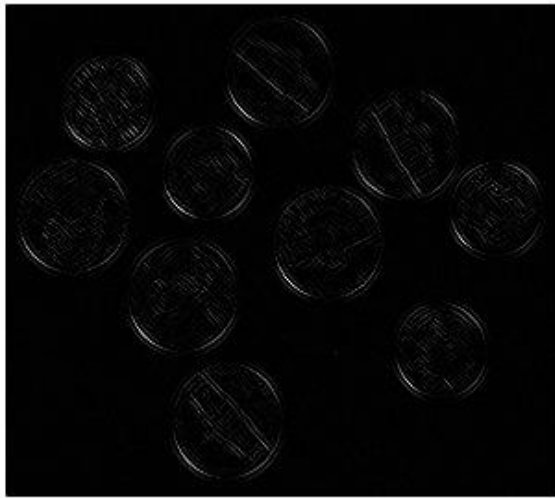
**Beispiel Laplace-Operator:**

$$L = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

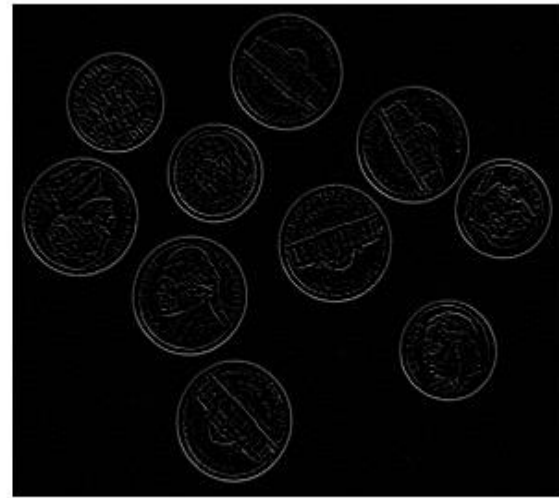


- Glättung mit der Binomialmaske und anschließende Subtraktion vom Originalbild

$$L' = 4(B_2 - I) = \frac{1}{4} \left[ \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right] = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$



$f * L$




$f * L'$




- **Sobeloperatoren:**

extrahieren Kanten mit Vorzugsrichtungen und gleichzeitiger Glättung

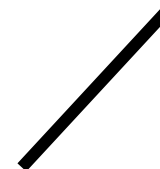

$$S_1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$$




$$S_2 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

$$S_4 = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$



1. Marr-Hildreth edge detector
2. Canny edge detector

## **Berücksichtigen zusätzlich:**

- Eigenschaften der Kanten
- Rauschen im Bild

[1980] Marr, D and Hildreth, E.: „Theory of Edge Detection“,  
Proc. R. Soc. Lond., vol. B207, pp. 187-217, 1980.

## Grundlagen des Kantendetektors nach Marr und Hildreth:

1. Grauwertunterschiede sind nicht von der Bildauflösung abhängig, so daß die Erkennung von Kanten Operatoren von verschiedenen Größen benötigt
2. Ein plötzlicher Grauwertsprung (=Kante) führt zu einem Peak in der 1.ten Ableitung oder zu einem Nulldurchgang (zero crossing) in der 2.ten Ableitung

## Eigenschaften des Marr und Hildreth Kantendetektors:

1. Operator muss sich in der Größe verändern können:
  - Große Filtermasken detektieren verschmierte Kanten,
  - kleine Filtermasken detektieren feine Details.
2. Differenzenoperator der 1.ten oder 2.ten Ableitung

Marr Hildreth Filter ist die 2.te Ableitung der Gaußfunktion oder der Laplace-Operator der Gaußfunktion:

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \quad \text{mit} \quad G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Berechnung der 2.ten Ableitung der Gaußfunktion:

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \quad \text{mit} \quad G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \frac{\partial}{\partial x} \left[ \frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right] + \frac{\partial}{\partial y} \left[ \frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right]$$

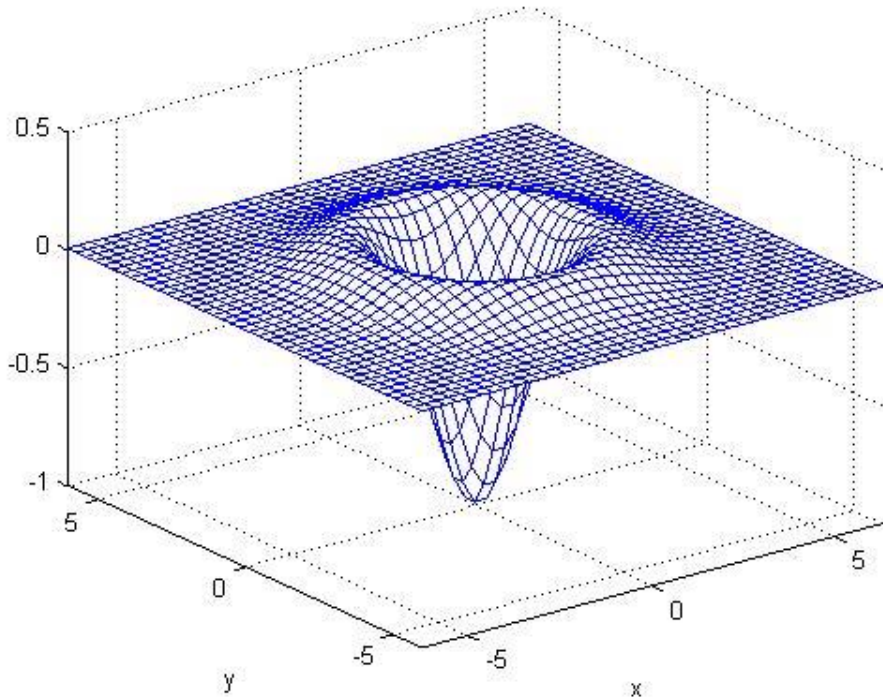
$$= \left[ \frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right] \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[ \frac{y^2}{\sigma^4} - \frac{1}{\sigma^2} \right] \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left[ \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

**Laplacian of Gaussian**  
**LoG**

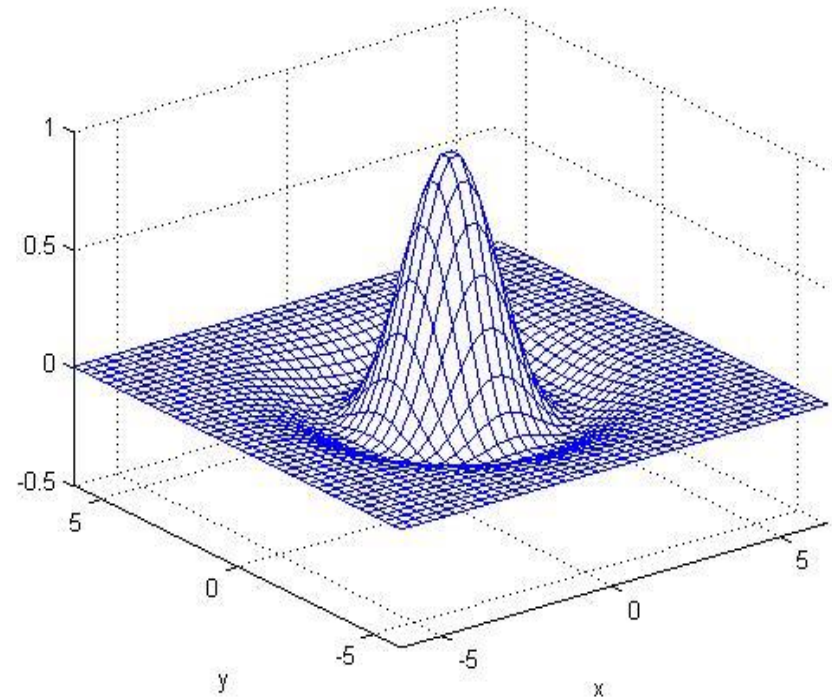
# Marr Hildreth Kantendetektor

$$((x^2+y^2-2\sigma^2)/\sigma^4) \exp(-(x^2+y^2)/(2\sigma^2))$$



LoG mit  $\sigma = \sqrt{2}$

$$(-(x^2+y^2-2\sigma^2)/\sigma^4) \exp(-(x^2+y^2)/(2\sigma^2))$$



Negative Log = Mexican Hat Function

# Marr Hildreth Kantendetektor

Approximation des negativen  
LoG Filter h =

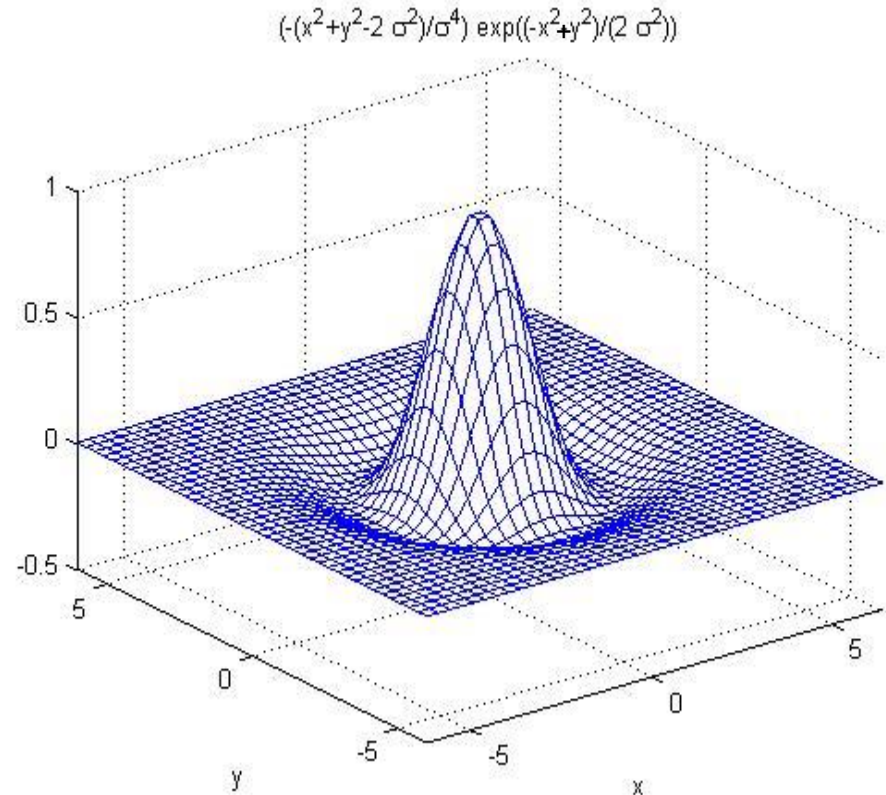
0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

$$g(x, y) = [\nabla^2 G(x, y)] * f(x, y)$$

$$g(x, y) = \nabla^2 [G(x, y) * f(x, y)]$$

1. Gauss-Filterung

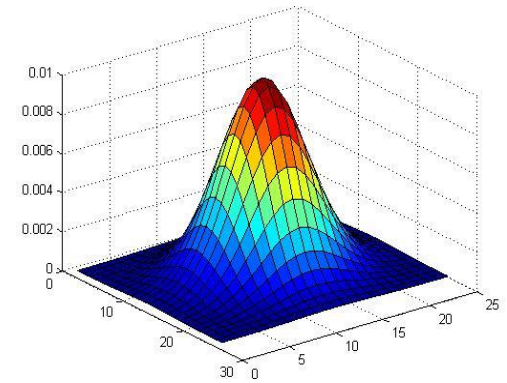
2. Laplace-Filterung  
aus dem Ergebnis der  
Gaussfilterung



Negative Log = Mexican Hat Function

1. Filtere das Originalbild mit einem  $n \times n$  Gauss-Tiefpaßfilter mit  $n > 6\sigma$  (99,7% der Bilddaten sind  $\pm 3\sigma$  um den Mittelwert)
2. Berechne die Laplace-Filterung auf das Ergebnis aus Schritt 1, z.Bsp. mit dem  $3 \times 3$  Laplace-Filter
3. Suche die Nulldurchgänge (Zero Crossings) aus dem Bild aus Schritt 2

25x25 Gauss-  
Filter  
bei  $\sigma=4.0$



3x3 Laplace-Filter  
 $h =$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



## Algorithmus Zero Crossing Pixel:

3. Untersuche die 3x3 Umgebung um ein Pixel.  
Falls einer der folgenden Fälle eintritt, wird das Pixel als Zero Crossing Pixel markiert:

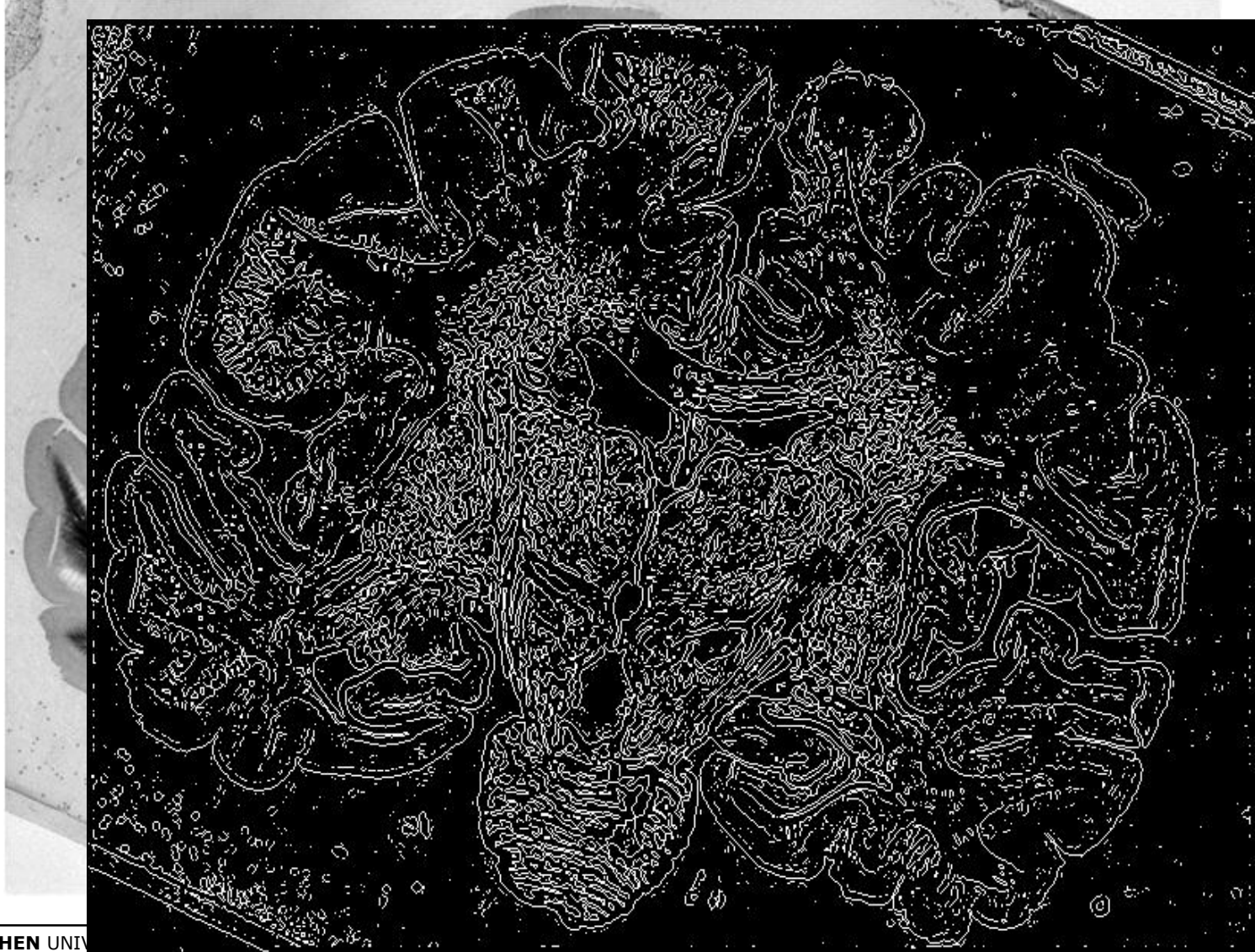
- Fall 1: Vorzeichenwechsel links-rechts
- Fall 2: Vorzeichenwechsel oben-unten
- Fall 3: Vorzeichenwechsel über die Diagonale  
links oben – rechts unten
- Fall 4: Vorzeichenwechsel über die Diagonale  
rechts oben – links unten

```
% Marr Hildreth Edge Detection in Matlab:
```

```
BW = edge(I, 'zerocross', thresh, h)
```

```
[BW, thresh] = edge(I, 'zerocross', ...)
```

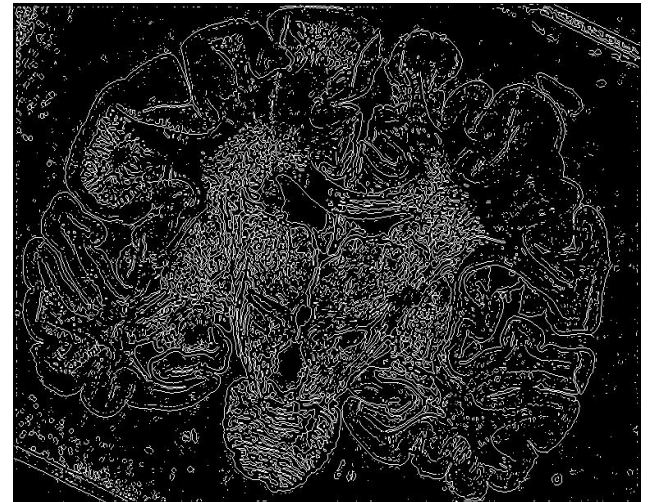
# Marr Hildreth Kantendetektor



# Marr Hildreth Kantendetektor

```
% Bild einlesen und verbessern
I = imread('gehirnschnitt.bmp');
I = rgb2gray(I);
I = imadjust(I, stretchlim(I), []);
figure, imshow(I, []);

% LoG
[BW,thresh] =
edge(I, 'log', [], 6*std2(I)/255);
thresh
figure, imshow(BW);
imwrite(I, 'gehirnschnitt_adjusted.jpg');
imwrite(BW, 'gehirnschnitt_log.jpg');
```



# Canny Edge Detector

[1986] Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986, pp. 679-698

## Algorithmus:

1. Rauschunterdrückung durch eine Gauß-Filterung
2. Berechnung des Gradientenbetrages:

$$g(x, y) = \sqrt{G_x^2 + G_y^2}$$

```
hy = fspecial('sobel');  
hx = hy';  
Gy = imfilter(double(I), hy, 'replicate');  
Gx = imfilter(double(I), hx, 'replicate');  
grad_mag = sqrt(Gx.^2 + Gy.^2);
```

Berechnung der Gradientenrichtung:

$$g(x, y) = \tan^{-1}(G_y / G_x)$$

Ein Kantenpunkt ist ein Pixel, das ein lokales Maximum in der Richtung des Gradienten aufweist.

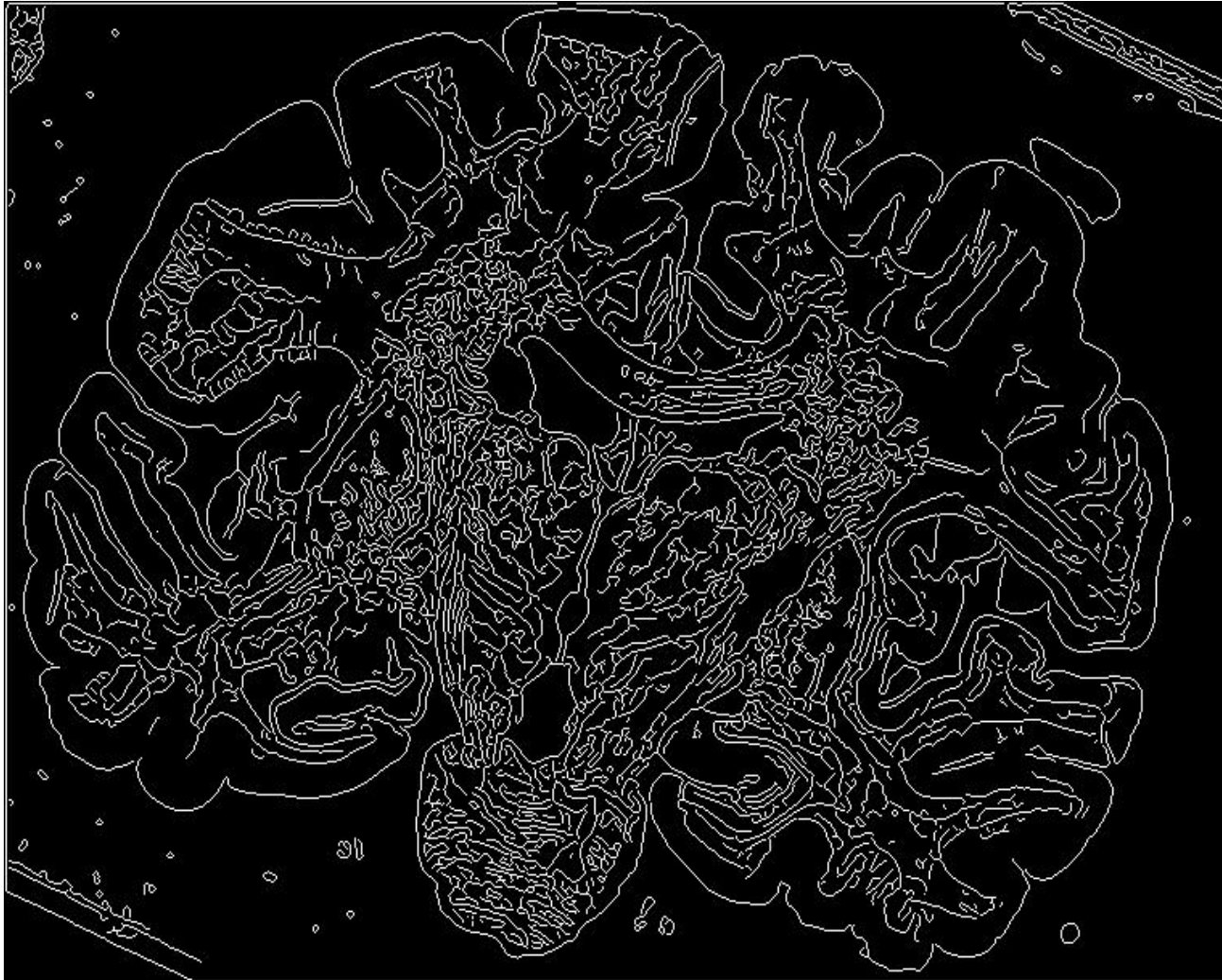
## Algorithmus:

3. Untersuche für jedes aus Schritt 2 detektierte Kantenpixel, ob der Gradientenbetrag ein lokales Maximum ist. Alle anderen Kantenpixel werden auf Null gesetzt. Es bleiben dann nur Kanten mit der Dicke 1 übrig, die entlang der stärksten Gradientengröße laufen. Hierbei berücksichtigt der Algorithmus 2 Schwellwerte  $T1$  und  $T2$ :
  - Starke Kantenpixel: Gradientenbetrag des Grätapixel  $> T2$
  - Schwache Kantenpixel:  $T1 < \text{Gradientenbetrag des Grätapixel} \leq T2$
4. Algorithmus verbindet starke Kantenpixel, wenn es einen Pfad aus der 8-er Nachbarschaft von schwachen Kantenpixeln gibt (Edge linking)



# Canny Edge Detector

---



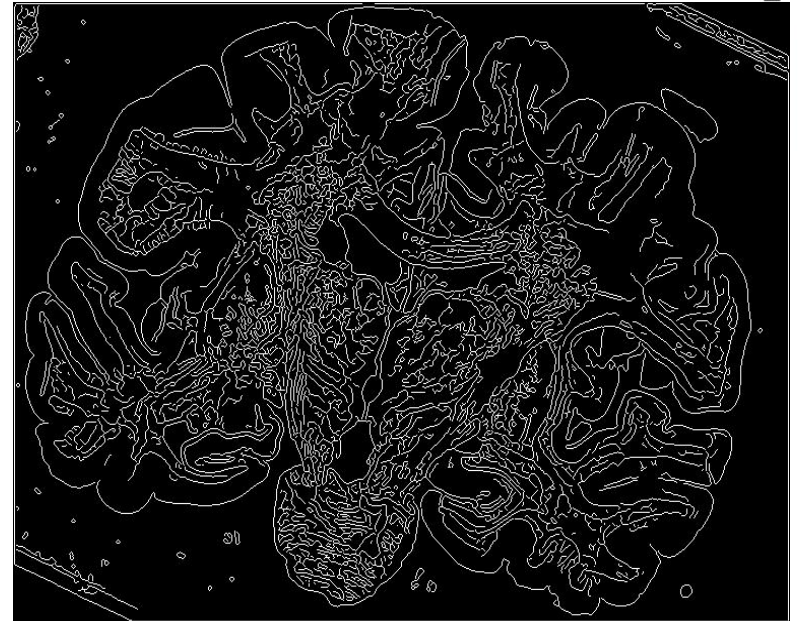
# Canny Edge Detector

```
>> [g, t] = edge(I, 'canny');  
>> t
```

```
t =
```

```
0.0438    0.1094
```

```
>> figure, imshow(g, []);
```



- Z.Bsp. Median, Maximum, Minimum innerhalb einer lokalen Umgebung
- Keine lineare Multiplikation der Filterkoeffizienten mit dem Bild (keine Faltung möglich)
- Beispiel: Median-Filter der Größe 3x3



# Beispiel: 3x3 Median Filterung

147	176	185	113	74	104	106	174	203	110	58
204	155	162	145	98	118	122	207	233	111	61
211	171	182	163	95	105	116	161	167	96	64
185	142	155	140	89	111	102	136	136	79	68
127	86	128	118	110	119	103	181	195	100	60
87	114	190	118	80	114	118	188	200	100	47
73	155	184	78	57	124	131	117	106	104	79
106	124	117	86	146	164	125	94	78	104	85
127	86	69	178	255	178	100	84	76	50	48
90	77	84	221	246	147	73	58	49	78	137
40	27	74	181	189	121	41	35	109	207	238



**3x3 Umgebung:**

**{147, 204, 211, 176, 155, 171, 185, 162, 182}**

**3x3 Umgebung sortiert:**

**{147, 155, 162, 171, 176, 182, 185, 204, 211}**

**Median:**

**Wert in der Mitte der sortierten Menge = 176**

```
I = imread('klinikum_gray.tif');  
figure; imshow(I);  
I2 = imcrop(I,[166 129 10 10]);  
imtool(I2)  
I3 = medfilt2(I2); // 3x3 Median Filterung  
imtool(I3)
```

0	155	145	98	98	98	106	122	111	61	0
155	176	163	145	105	105	118	167	167	110	61
155	171	155	145	111	105	118	136	136	96	64
127	155	142	128	111	105	116	136	136	96	64
87	128	128	118	114	110	118	136	136	100	60
86	127	118	118	114	114	119	131	117	100	60
87	117	118	117	114	124	124	118	104	100	79
88	117	117	117	146	131	124	100	94	79	50
88	90	88	146	178	147	100	78	78	78	50
40	77	84	181	181	147	84	73	76	78	50
0	40	74	84	147	73	41	41	49	78	0

# Matlab: Nicht lineare Filter

```
% Column filtering example
I = imread('tire.tif');
imshow(I), title('Original')
fun1 = @imagestd;
fun2 = @colstd;
disp('Column Filtering')
```

```
tic
```

```
I2 = nlfilter(I,[3 3],fun1);
```

```
toc
```

```
figure, imshow(I2),title('STD using Non-Linear Filtering')
```

```
tic
```

```
I3 = colfilt(I,[3 3],'sliding',fun2);
```

```
toc
```

```
figure, imshow(I3), title('STD using Column Filtering')
```

Function Handle:  
 fun1 = @imagestd;  
 Die Funktion imagestd ist  
 in der Datei imagestd.m  
 gespeichert

Original



STD using Non-Linear Filtering



# Matlab: Function Handle Funktionen

---

```
function y = imagestd(x)
%IMAGESTD berechnet die Standardabweichung von einem Bildblock
%x ist eine Matrix
y = uint8(round(std2(x)));
```

```
function y = colstd(x)
%COLSTD berechnet die Standardabweichung von einem Block, der
spaltenweise %verarbeitet wird; x ist ein Vektor
y = uint8(round(std(double(x))));
```

# Matlab: Blockweise Filterung

```
% Block processing example of an averaging filter
I = imread('tire.tif');

% Create a handle to the function BLKAVG.
% Where BLKAVG is an M-file function which computes:
% y = uint8(mean2(x)*ones(size(x)));

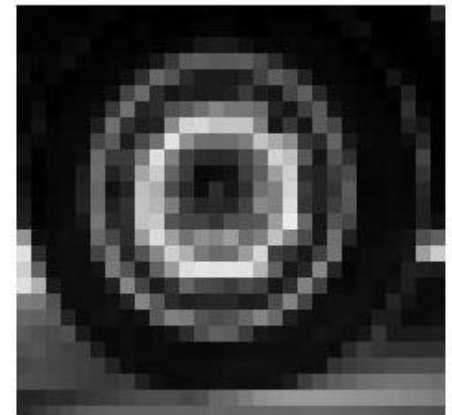
fun = @blkavg;
% For more information on function handles
% type at the command prompt 'doc function_handles'

I2 = blkproc(I,[8 8], fun);
imshow(I), title('Original')
figure
imshow(I2), title('Distinct Blocks Averaged')
```

Original



Distinct Blocks Averaged



# Matlab: Übergabe weiterer Parameter bei Function Handles

```
% Block processing example of an averaging filter
I = imread('tire.tif');
% Function Handle:
fun = @blkavg;
T = 0.5;
% additional input parameter T to the function handle
```

```
I2 = blkproc(I,[8 8], fun, T);
```

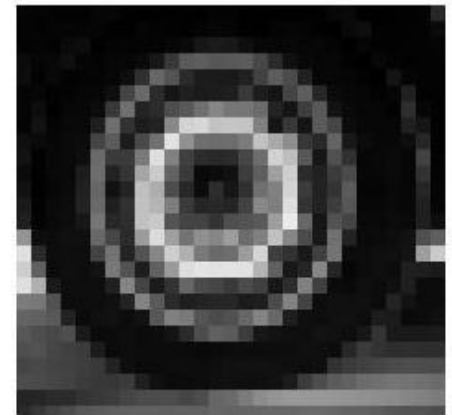
```
imshow(I), title('Original')
figure
imshow(I2), title('Distinct Blocks Averaged')
```

```
% Function Handle blkavg.m:
function y = blkavg(x, T)
y = uint8(mean2(x)*T*ones(size(x)));
```

Original



Distinct Blocks Averaged



# 3. Kapitel: Zusammenfassung

## Filter

### Glättungsfilter

Faltungsoperatoren:

- Mittelwertfilter
- Binomial-  
bzw. Gaussfilter

### Kantenfilter

Faltungsoperatoren:

- Einfache Differenzenfilter
- 1. Ableitung
- Filter 2.ter Ableitung
- Laplace Filter
- Sobel

Algorithmen, die Faltungen verwenden:

- Marr Hildreth Operator
- Canny Edge Operator

### Nicht lineare Filter

- Median
- Minimum
- Maximum  
(nicht über Faltung berechenbar!)

# Vielen Dank für die Aufmerksamkeit !

FH Aachen  
Fachbereich Elektrotechnik und Informationstechnik  
Prof. Ingrid Scholl  
Eupenerstr. 70  
52066 Aachen  
T +49. 241. 6009 52177  
scholl@fh-aachen.de  
www.fh-aachen.de