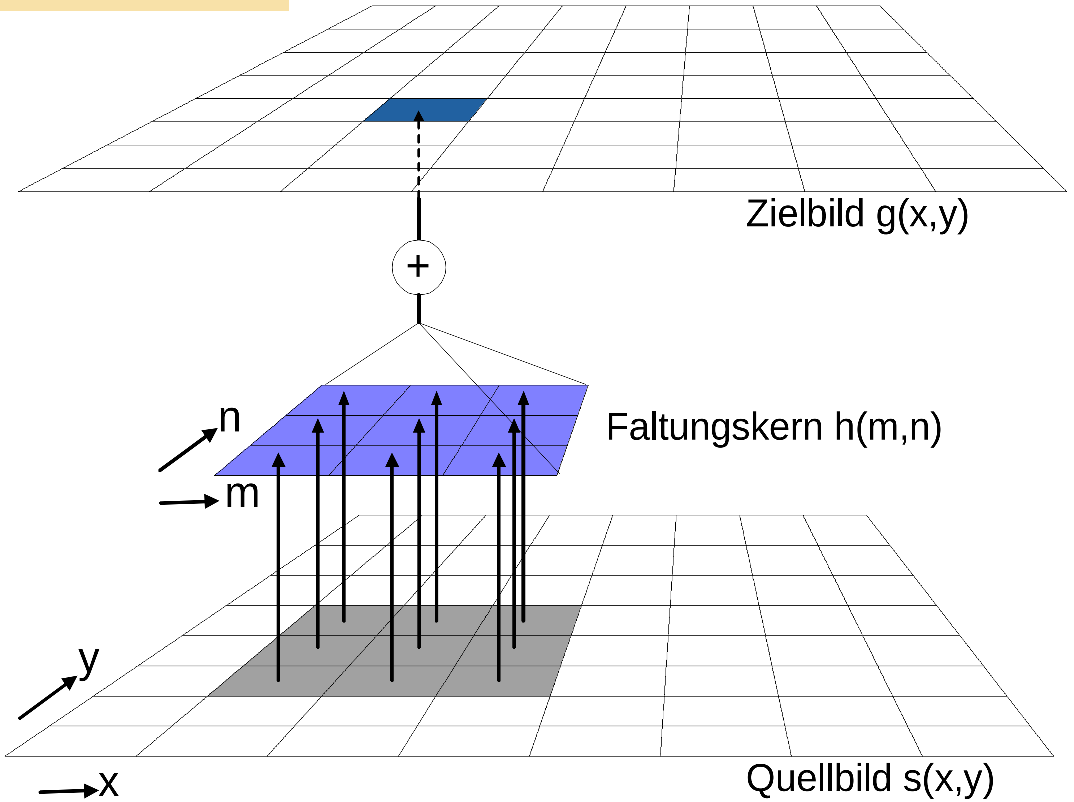
3.3 Filterung im Ortsbereich

3.3.1. Faltung mit einem Faltungskern

$$g(x,y) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} h(m,n) \cdot s(x-m, y-n)$$

$h(m,n)$: Faltungskern

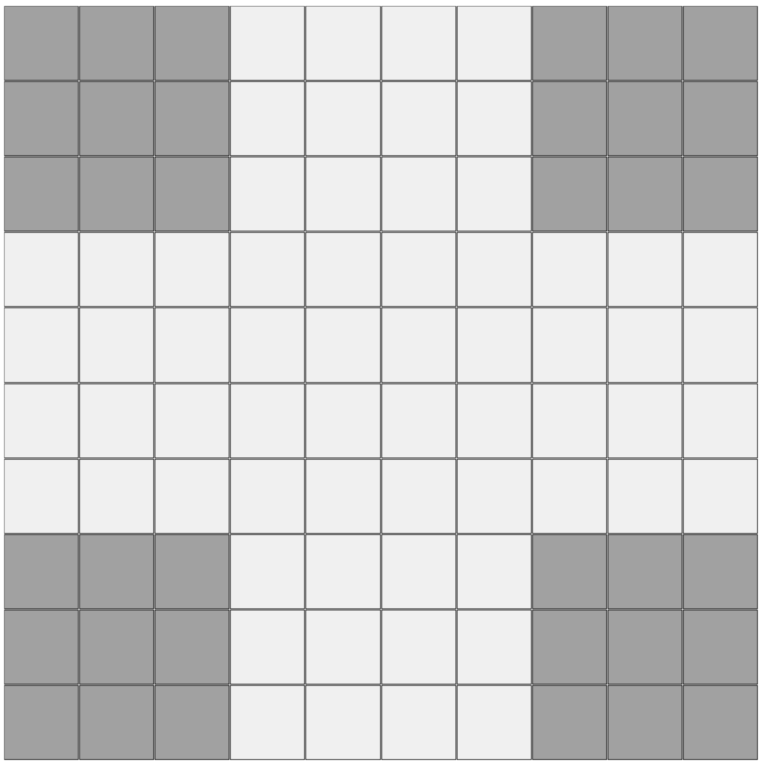


Anwendungsbeispiele:

- Bildglättung (Kap. 3.4)
- Bildschärfung
- Kantenfilter (Kap. 3.5)

ÜBUNG: Faltung mit einem 3x3-Faltungskern

Falten Sie das Bild (Grauwerte 0 und 1) mit den nebenstehenden Faltungskernen.



-1	-1	-1
-1	8	-1
-1	-1	-1

a)

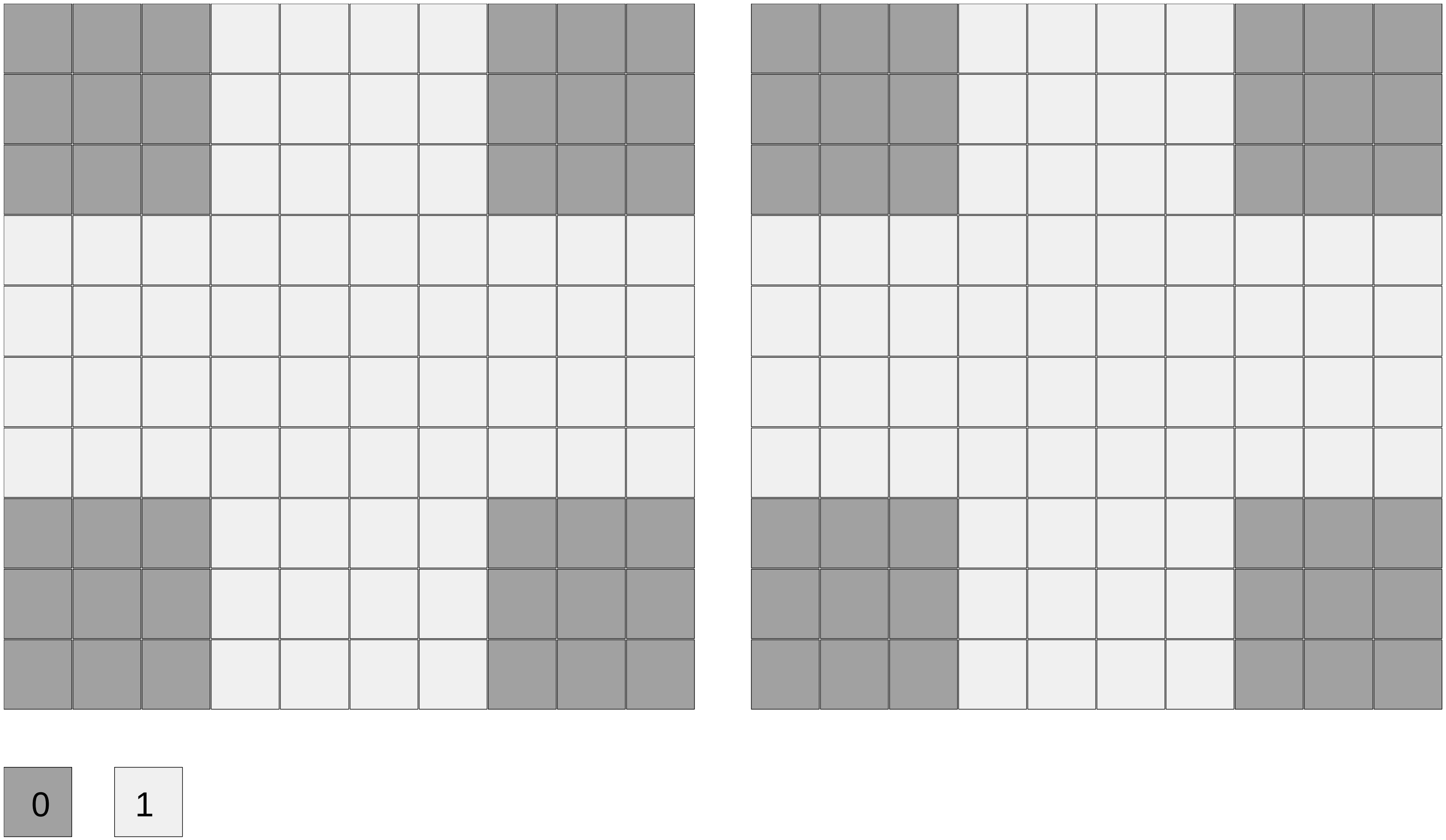
1	0	-1
1	0	-1
1	0	-1

b)

Angenommen es handelt sich um ein 8-bit-Grauwertbild. Welchen Größt- und Kleinstwert könnte das Ergebnis annehmen?

0 1

Übungsblatt: Faltung



3.3.2 Separierbare Filterkerne

Ein Faltungskern $\underline{K}$ ist dann separierbar, wenn der Faltungskern in Form eines dyadischen Produktes zweier Vektoren darstellbar ist.

Das ist dann der Fall, wenn die Zeilen des Faltungskerns paarweise linear voneinander abhängig sind [d.h. rang($\underline{K}$) =1].

$$\underline{K} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

Ist ein Faltungskern separierbar, dann kann die zweidimensionale Faltung durch zwei eindimensionale Faltungen ersetzt werden.

Vorteil: Anzahl der Multiplikationen sinkt

a) 2-dim. Faltung : M*N * n²

b) 2 * 1dim. Faltung : M*N * 2* n

mit Bildgröße M*N und Faltungskerngröße n*n

ÜBUNG: Separierbarkeit

Welche Faltungskerne sind separierbar?
Wie sehen die 1-dim. Kerne der separierbaren Faltungskerne aus?

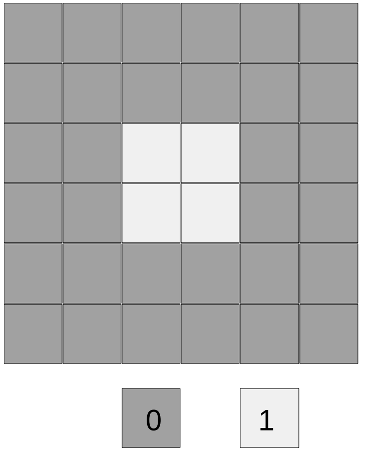
$$\underline{A} = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \qquad \underline{C} = \begin{pmatrix} 1 & 3 & 1 \\ -2 & -6 & -2 \\ 1 & 3 & 1 \end{pmatrix}$$

$$\underline{B} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

ÜBUNG: Separierbarkeit

1. Zeigen Sie am Beispiel des nebenstehenden Bildes und dem angegebenen Filterkern, dass die Faltung tatsächlich durch zwei eindimensionale Filteroperationen durchgeführt werden kann (Separierbarkeit).

1	2	1
2	4	2
1	2	1



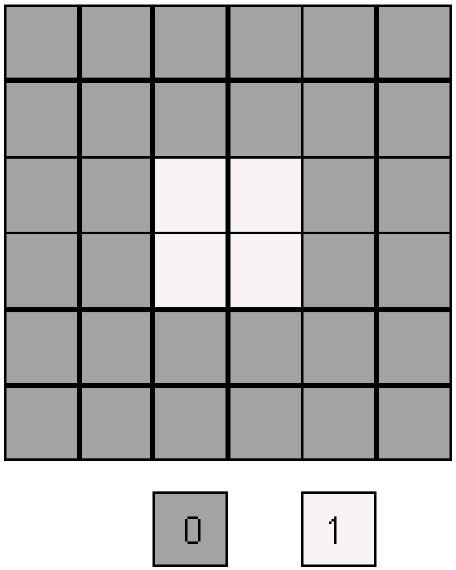
2. Wieviele Additionen/Multiplikationen sind pro Bildpunkt notwendig, wenn

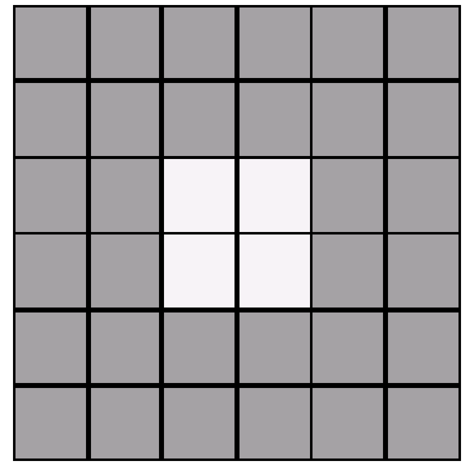
   a) das Ergebnis durch Faltung mit einem zweidimensionalen Faltungskern berechnet wird,

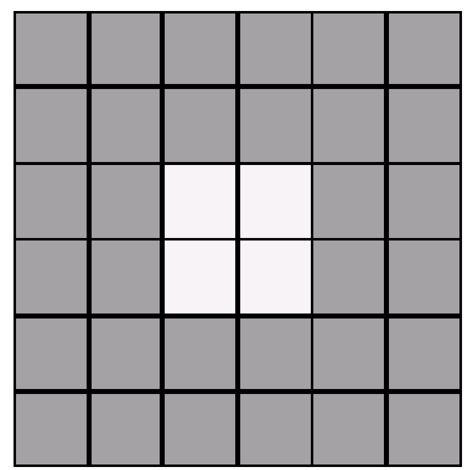
   b) das Ergebnis durch zwei eindimensionale Faltungen berechnet wird (im Falle separabler Filter, z.B. dem Binomialfilter).

Anm.: Bild habe die Größe 1000x1000, der Faltungskern sei 5x5

Übungsblatt: Separierbarkeit







3.4 Bildglättung mit Unschärfeoperationen

3.4.1 Rechteckfilter

Ziel: Filtern zur Bildglättung
(= Entfernung hochfrequenter Bildanteile).

Grundgedanke:

Es wird aus mehreren Bildpunkten der Mittelwert gebildet.

Faltungsmasken (Beispiele):

1/9 x

1	1	1
1	1	1
1	1	1

1/25 x

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

Anmerkung:

- Faltungskern ist *separierbar* .
- Filter ist *anisotrop* (richtungsabhängig)

Beispiel: Anisotropie des Rechteckfilters

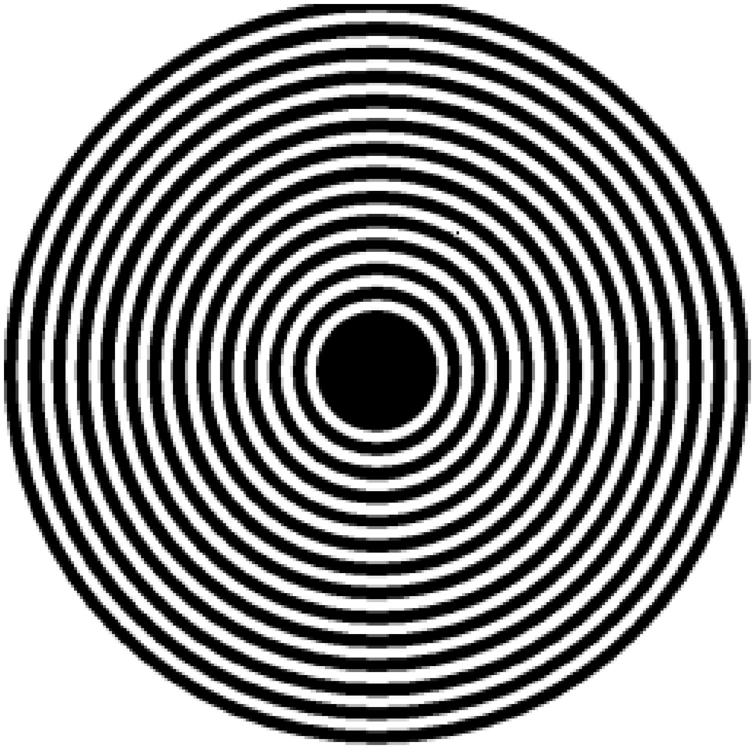
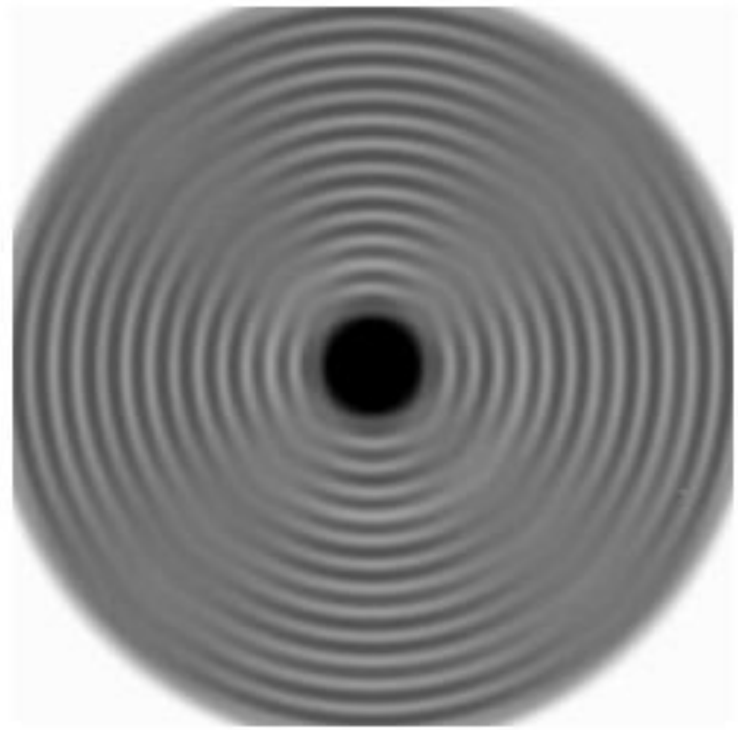


Bild 256 x 256



nach Anwendung einer 11x11-Rechteckmaske

Beispiel: Frequenzverhalten des Rechteckfilters

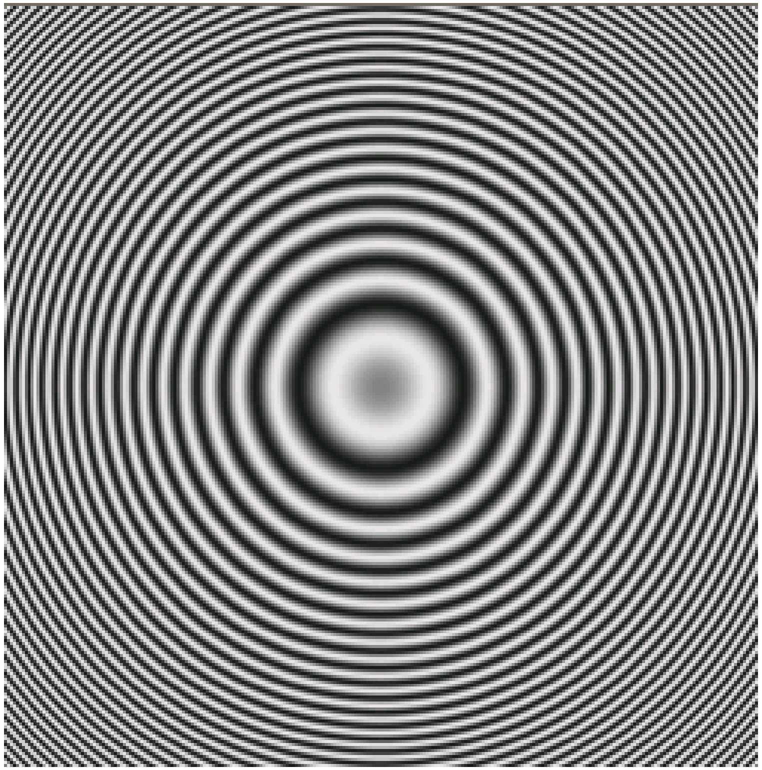
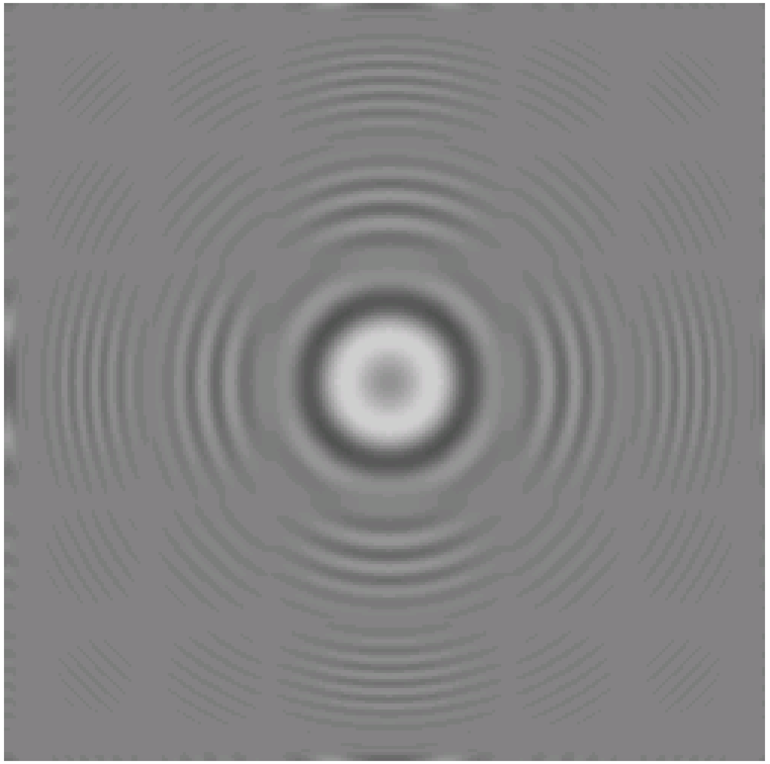


Bild 256 x 256



nach Anwendung einer 13x13-Rechteckmaske

3.4.2 Binomialfilter

Ziel: Filter für die Bildglättung, jedoch mit besseren Eigenschaften als das Rechteckfilter.

Vorteile gegenüber Rechteckfilter:

- Hohe Frequenzanteile werden besser unterdrückt.
- Weitgehend *isotrop*, d.h. die Filterwirkung ist in allen Richtungen gleich.

Faltungsmasken (Beispiele):

1/16 x

1	2	1
2	4	2
1	2	1

1/256 x

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

Anmerkung:

- Faltungskern ist separierbar (s.u.).

Entwurf von Binomialmasken

Die Filterkoeffizienten können anhand des *Pascalschen Dreiecks* abgeleitet werden (= *Binomialkoeffizienten*).

Den zweidimensionalen Faltungskern erhält man aus dem Matrixprodukt eines vertikalen mit einem horizontalen 1D-Binomialkern.

Beispiel (3x3-Binomialkern):

1/4 x 1/4 x

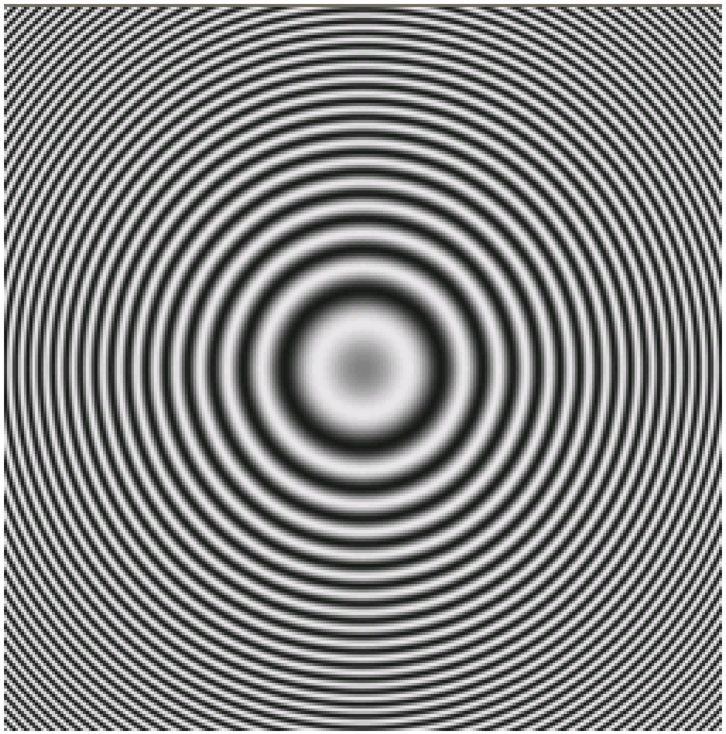
	1	2	1
1	1	2	1
2	2	4	2
1	1	2	1

n	Koeffizienten	Korr.
1	1 1	1
2	1 2 1	1/4
3	1 3 3 1	1/8
4	1 4 6 4 1	1/16
5	1 5 10 10 5 1	1/32
6	1 6 15 20 15 6 1	1/64
7	1 7 21 35 35 21 7 1	1/128
8	1 8 28 56 70 56 28 8 1	1/256

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

k = 0, 1, 2, 3, ...
k = Index innerhalb einer Zeile

Beispiel: Richtungs- und Frequenzverhalten des Binomialfilters



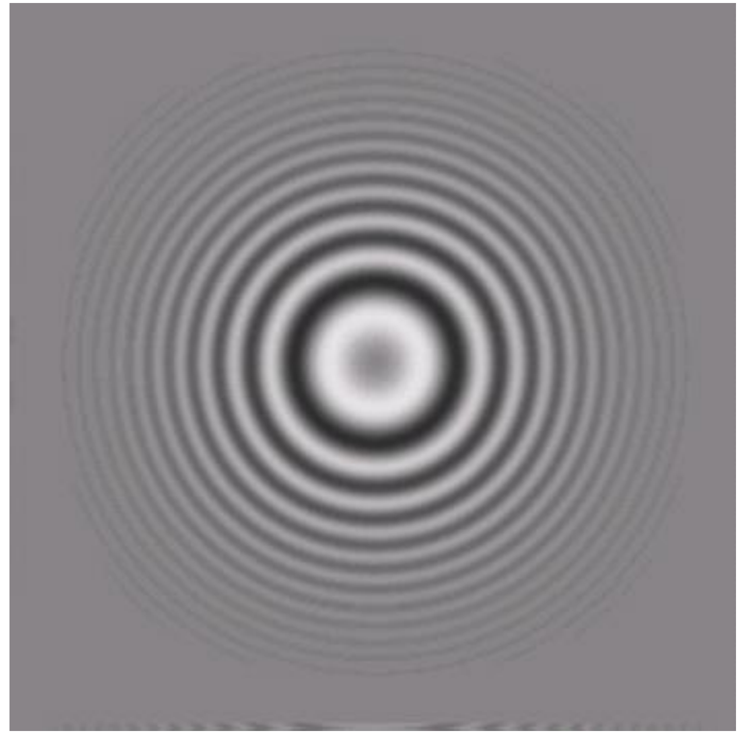


Bild 256 x 256

nach Anwendung eines 17x17-Binomialfilters

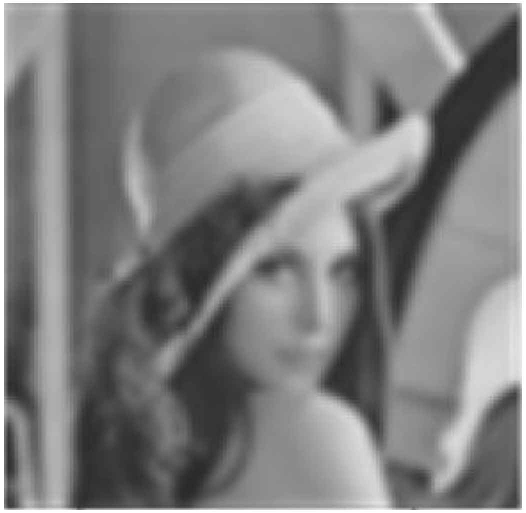
3.4.3 Gauss-Tiefpass

3.4.3.1 Grundlagen

Ziel: Standardfilter für die Bildglättung (ähnlich Binomialfilter, aber universeller). Separierbarer Filterkern.

Besonderheiten:

1. Mit dem Parameter σ wird der Glättungsgrad eingestellt (s.u.).
2. Ein größeres σ erfordert einen größeren Faltungskern.
3. Varianten: a) Floatingpoint-Kern b) Ganzzahlkern



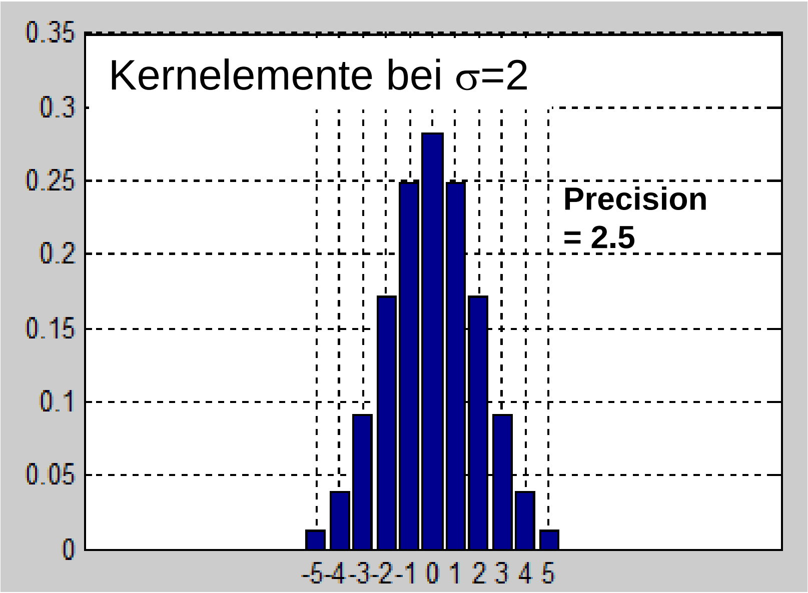
Grundlage für die Berechnung der Faltungskernelemente ist die *Gaussfunktion*:

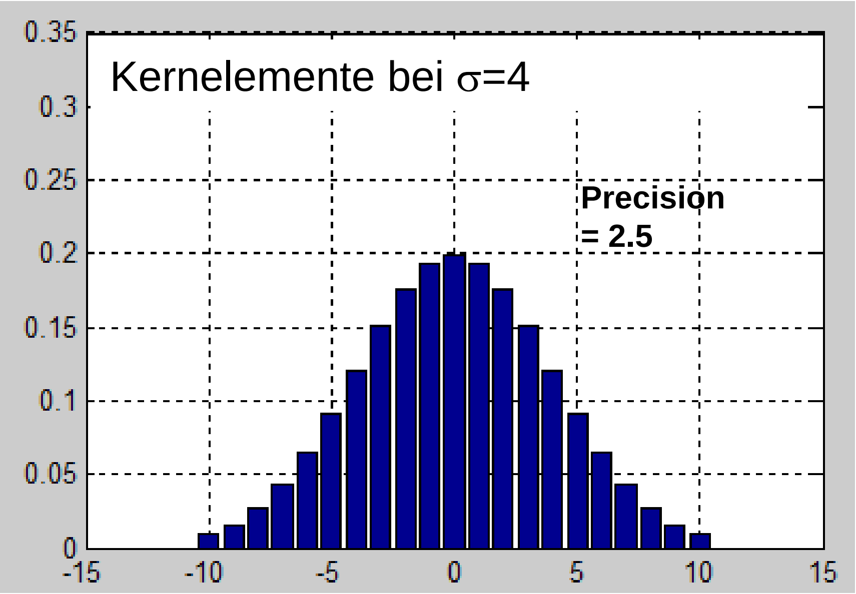
$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

Die gewünschte σ (Funktionsbreite) und Näherungsgüte (*Precision*) legen die die Größe des Faltungskerns fest:

mit `k = int ( Precision * σ )`
und `Precision =` 2.0 ... 3.0 (größerer Wert vergrößert den Kern)

→ Faltungskerngöße 2k+1:





3.4.3.2 Floatingpointkern : Berechnung des Faltungskerns

$$\vec{G}^* = \left[g(-k), \quad g(-k+1) \quad .... \quad g(-1), \quad g(0), \quad g(1) \quad ... \quad g(k-1), \quad g(k) \right]$$

Beispiel: k=2 $\vec{G}^* = \left[g(-2), \quad g(-1) \quad , \quad g(0), \quad g(+1), \quad g(+2) \right]$

Normierung, so dass die Summe der Faltungskernelemente = 1 ist.

$$\vec{G} = \frac{\vec{G}^*}{\sum_{n=-k}^{+k} g(n)}$$

Beispiel: σ = 0.750, Precision = 2.5 → k = int (0.75*2.5) = 1

$$\vec{G}^* = \left[g(-1), \quad g(0), \quad g(+1) \right] = [0.219, \quad 0.532, \quad 0.219]$$

$$mit \quad \sum g(n) = 0.219 + 0.532 + 0.219 = 0.97$$

$$\vec{G} = [0.226, \quad 0.549, \quad 0.226]$$

3.4.3.3 Ganzzahlkern : Berechnung des Faltungskerns

$$\vec{G}^* = \left[g(-k), \;\; g(-k+1) \;\; .... \;\; g(-1), \;\; g(0), \;\; g(1) \;\; ... \;\; g(k-1), \;\; g(k) \right]$$

Beispiel: G* = [0.0619 0.2414 0.3799 0.2414 0.00619]

Normierung, so dass die Randelemente des Faltungskerns = 1 sind.

$$\vec{G}^{**} = round\left(\frac{\vec{G}^*}{g(k)} \right)$$

Beispiel: G** = [1 4 6 4 1]

Berechnung des Normierungsfaktors Nf, mit dem die Summe der Faltungskern-elemente auf 1 normiert werden kann.

$$Nf = \sum_{n=-k}^{k} g^{**}(n)$$

Beispiel: G = 1/16 * [1 4 6 4 1]

Beispiele: Faltungskerne bei verschiedenen σ

σ = 0.750, Precision = 2.5 → k = 1
G = [1 2 1]

σ = 1.050, Precision = 2.5 → k = 2
G = [1 4 6 4 1]

σ = 1.222, Precision = 2.5 → k = 3
G = [1 5 15 20 15 5 1]

σ = 1.350, Precision = 3.0 → k = 4
G = [1 7 27 61 81 61 27 7 1]

→ vergl. mit Binomialkern

3.4.3.4 2D-Gauss-Faltungskern

Im Allg. wird man die die Filterung als 2 x 1D Filterung (hor. / vert.) vornehmen (wg. der Separationsfähigkeit des Gaussfilters).

Dies entspricht der Faltung mit dem 2D-Faltungskern: $\underline{K} = \vec{G}^T \cdot \vec{G}$

Beispiel: G = 1/16 * [1 4 6 4 1]

$\underline{K} =$ 1/256 x

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

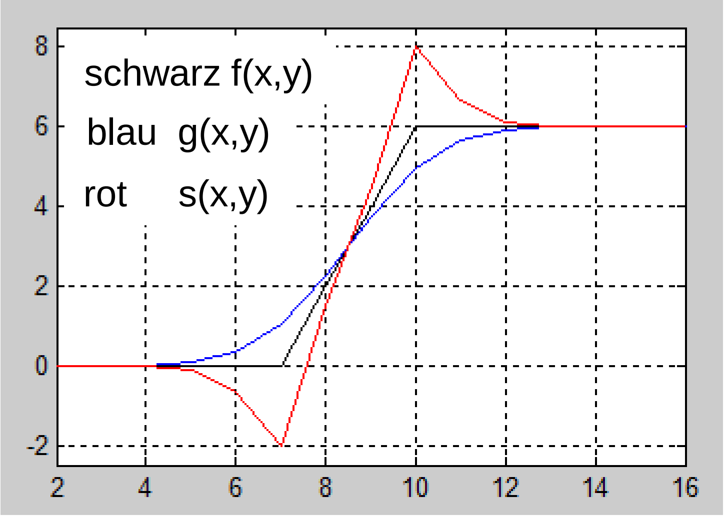
<u>*3.4.3.5 Bildschärfung mit dem Gaussfilter → unsharp masking*</u>

Eine Bildschärfung kann durch gewichtete Subtraktion eines geglätteten Bildes ***g(x,y)*** (Gauss) vom Originalbild ***f(x,y)*** erreicht werden:

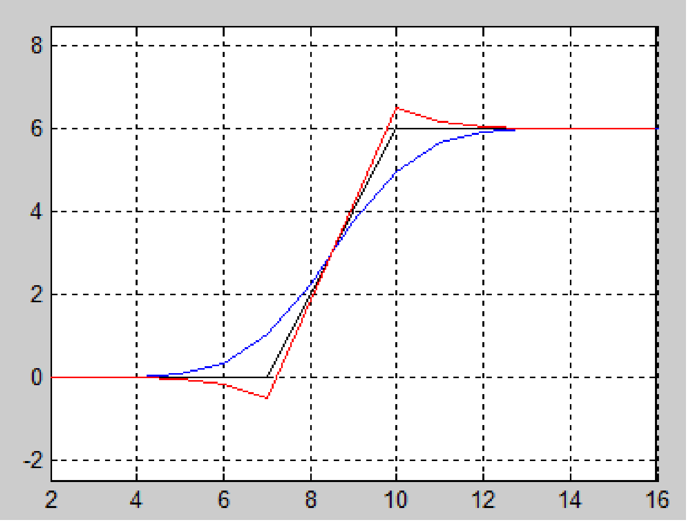
$$s(x,y) = \frac{c}{2c-1} \cdot f(x,y) - \frac{1-c}{2c-1} \cdot g(x,y)$$

mit c = 0.6 ... 0.8 <u>Anm.</u>: je kleiner, desto schärfer

c = 0.6



c = 0.75



Beispiel: unsharp masking

← schärfer, aber mehr Bildrauschen

3.5 Kantendetektion

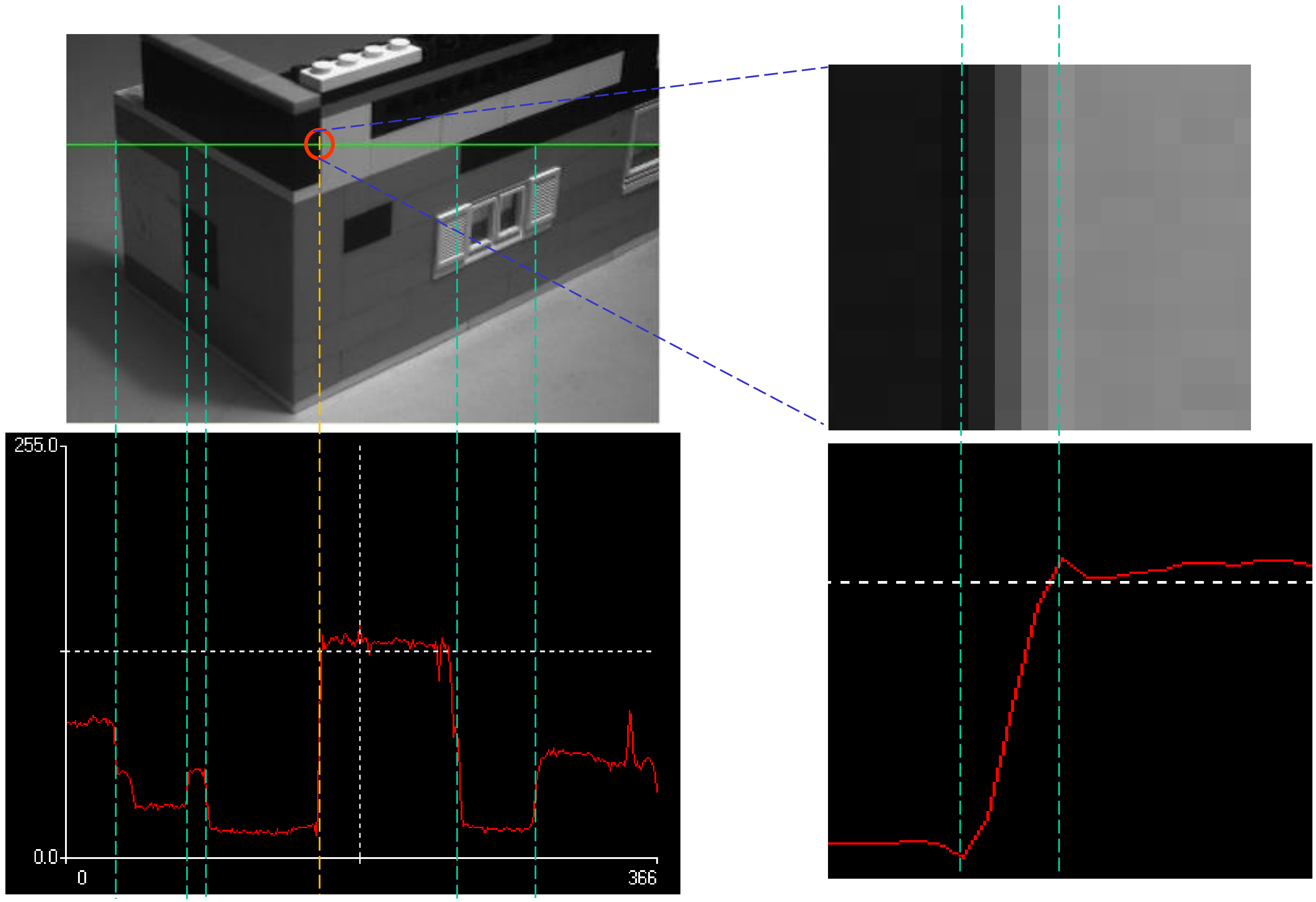
3.5.1 Einführung

3.5.1.1 Ziel

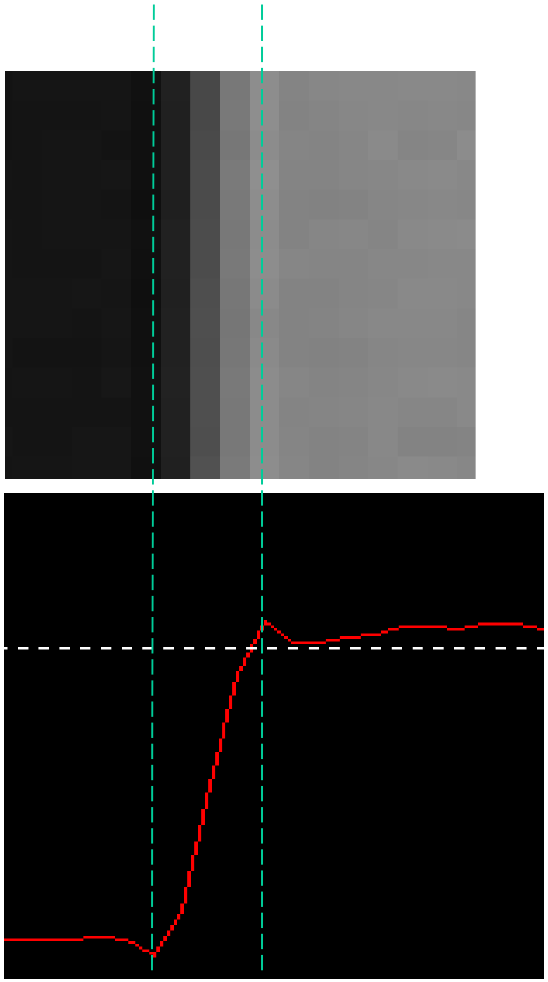
Quellbild	Zielbild
Homogene Bildbereiche	= 0
Grauwertänderungen	≠ 0

→ Hervorhebung von Grauwertsprüngen (Intensitätskanten)

3.5.1.2 Grauwertverlauf im Kantenbereich (von Kamerabildern)



3.5.1.3 Kantenort bestimmen



Wie legt man den Kantenort fest ?

Möglichkeit 1:

→ Wendepunkt im Grauwertverlauf

- sehr empfindlich

Möglichkeit 2:

→ Ort der größten Steigung

- hohe Steigung = hohe Kantenrelevanz

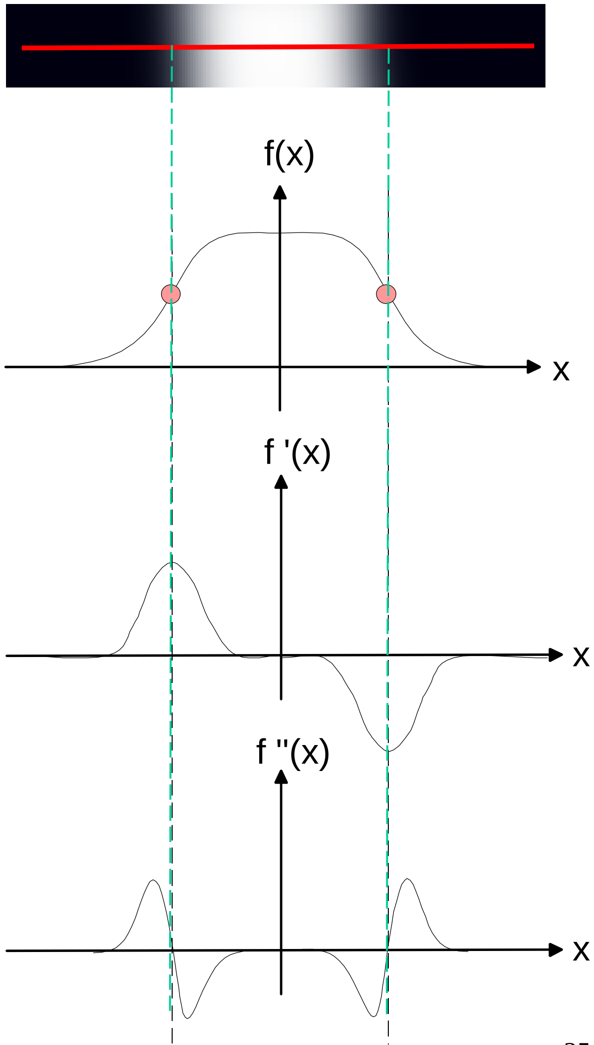
3.5.2 Laplace-Filter

3.5.2.1 Kantenort = Helligkeitswendepunkt

Grundgedanke:

1. Ein Wendepunkt in einer Funktion erzeugt einen Nulldurchgang in der zweiten Ableitung dieser Funktion.
2. Betrachtet man ein Bild als Grauwert-gebirge f(x,y), so deuten Helligkeits-wendepunkte auf Kantenverläufe hin.

Fazit: Kantenpunkte lassen sich somit finden, indem Nulldurchgänge in der 2. Ableitung gesucht werden (*zerocrossings*).



Lösungsansatz:

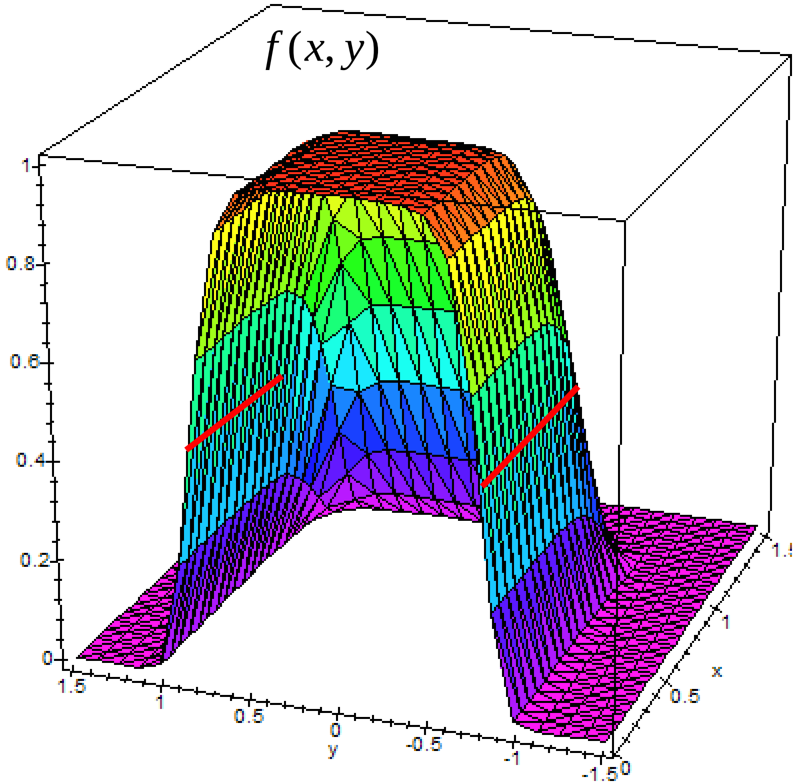
Bei Funktionen mit mehreren Variablen f(x,y) wird die zweite Ableitung mit dem *Laplace*-Operator berechnet:

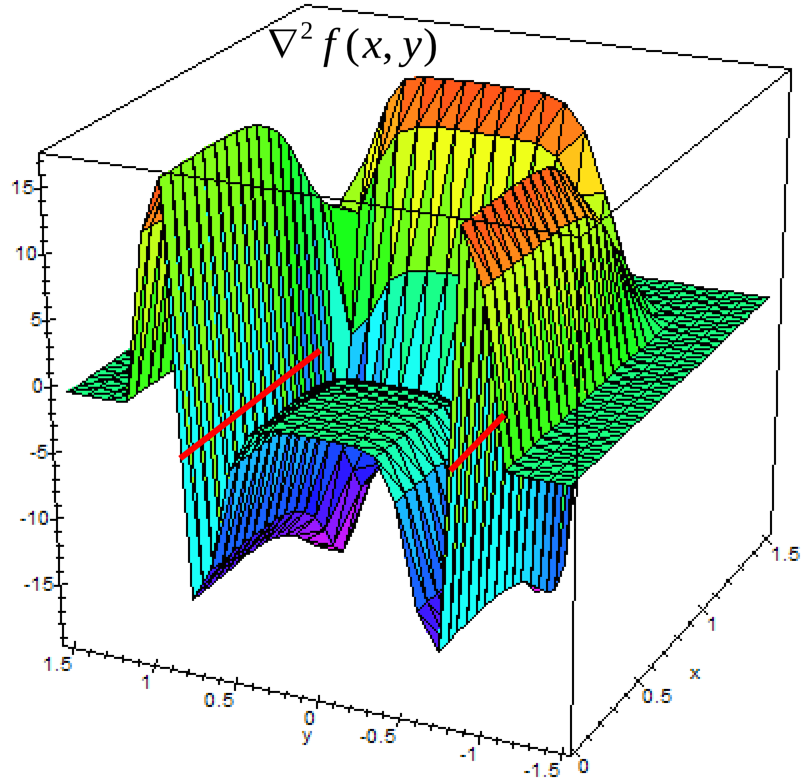
3.5.2.2 Mathematische Grundlagen

Den Laplace-Operator einer skalaren Funktion f(x,y) erhält man mit:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad x, y \in \mathbf{R}$$

Das Ergebnis des Laplace-Operators ist ein **Skalar** !





ÜBUNG: Berechnung der Laplace-Funktion

Gegeben sei folgende Funktion (zweier Variabler):

$$f(x, y) = e^{-(x^4 + y^4)}$$

Berechnen Sie den Wert der Laplace-Funktion $\nabla^2 f(x, y)$ an den Stellen:

a) x=0.0, y=0

b) x=1.0, y=0

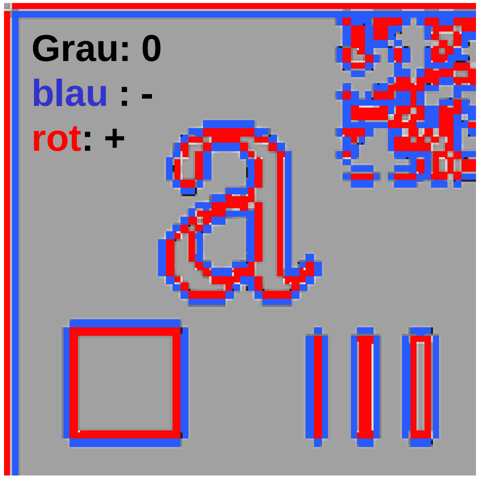
3.5.2.3 Anwendung auf diskrete Funktionen $x, y \in \mathbf{G}$

Da bei Bildern x und y diskret sind, lässt sich der Differential-quotient nur annähern (durch den Differenzenquotienten) und man erhält für die Teilableitungen: → s. Tafel

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) \approx \frac{\partial}{\partial x}\left[f(x+0.5, y) - f(x-0.5, y)\right]$$
$$\approx f(x+1, y) - 2f(x, y) + f(x-1, y)$$

Für die 2. Ableitung nach y verfährt man analog.
Damit erhält man die Faltungskerne:

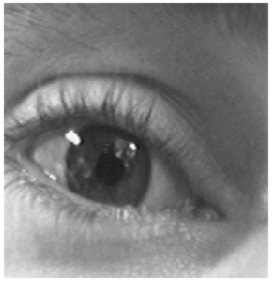
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



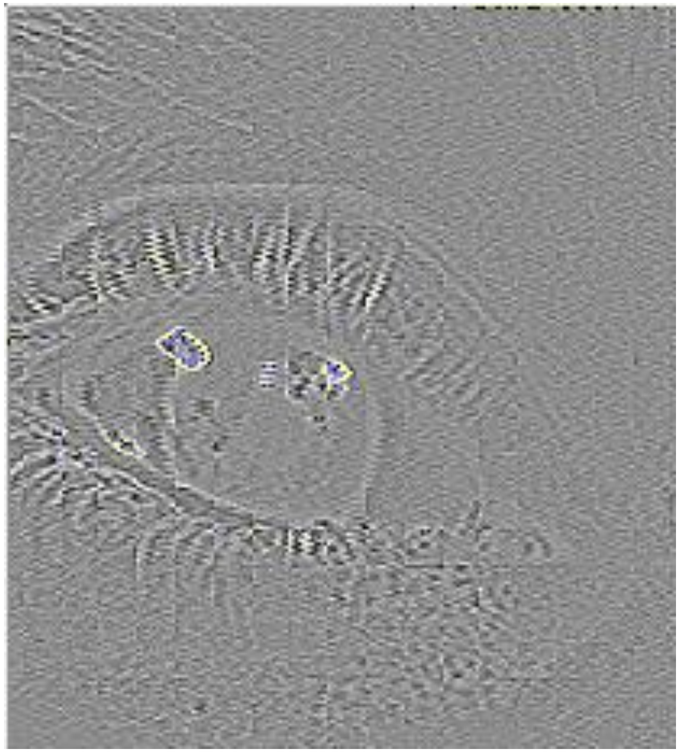
Nachteil des Laplace-Operators:

- verstärkt Bildrauschen

Beispiel zur Rauschempfindlichkeit des Laplace-Filters

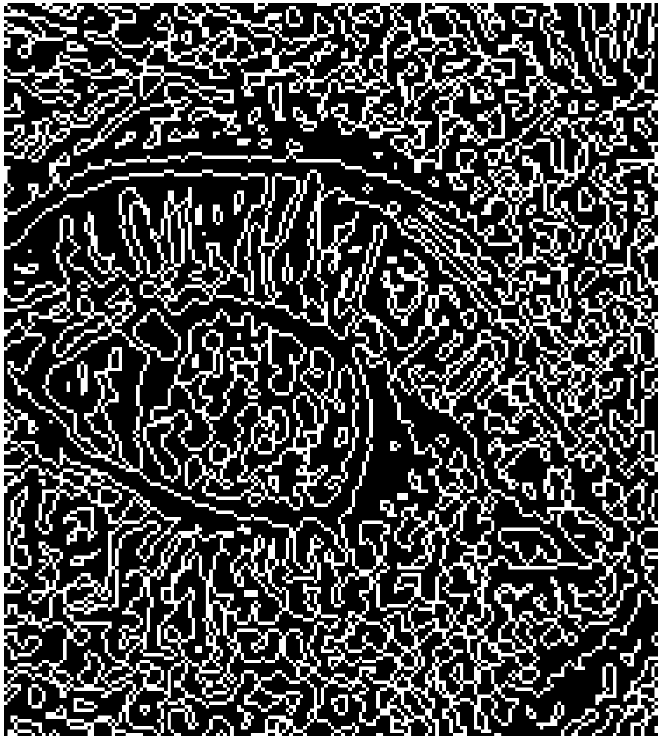


Laplace



→ 0
hellere Werte → +
dunklere Werte → -

zerocrossings



Nulldurchgänge der 2. Ableitung des zuvor (etwas) geglätteten Bildes.

3.5.2.4 Bildschärfung mit dem Laplace-Filter

Durch Subtraktion des Laplace-gefilterten Bildes vom Originalbild kann die Bildschärfe verbessert werden.
(siehe nächste Seite)

$$f_{sharp}(x,y) = f(x,y) - \nabla^2 f(x,y)$$

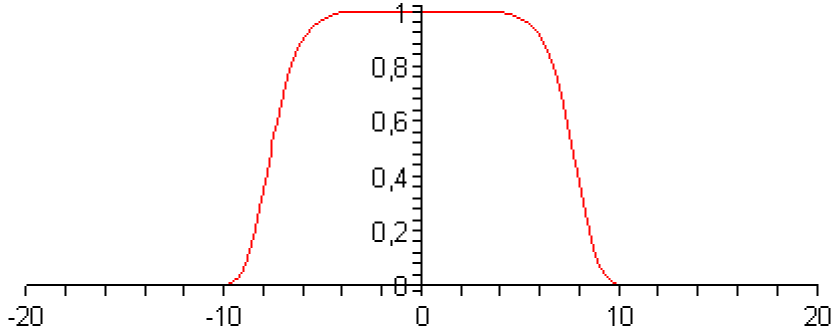
Anm.: Es wird aber auch das Bildrauschen verstärkt.

Beispiel:

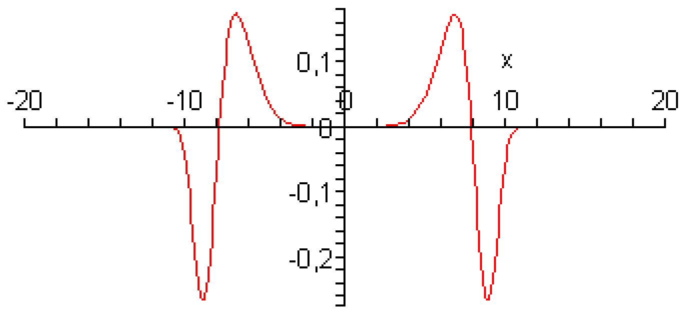
$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \; - \; \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \; = \; \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Prinzip des Schärfungsoperators:

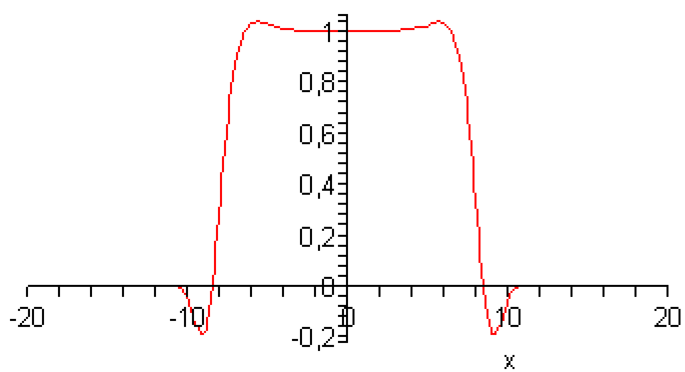
f(x)



-f“(x)



f(x)-f“(x)



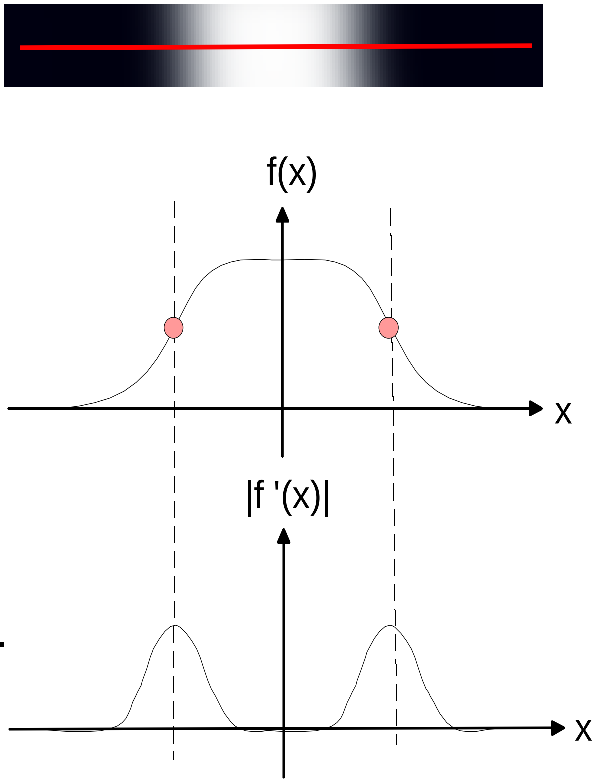
3.5.3 Sobel-Filter

3.5.3.1 Kantenort = maximale Helligkeitssteigung

Ziel: Detektion von Diskontinuitäten (z.B. Kanten), ohne den Nachteil der starken Rauschverstärkung (wie z.B. beim Laplace-Filter).

Grundgedanke:

1. Eine Kante im Bild erzeugt eine hohe Flankensteigung im Grauwertgebirge.
2. Die Höhe der ersten Ableitung ist ein Maß für die Steigung.



Lösungsansatz:

Bei Funktionen mit mehreren Variablen f(x,y) wird die erste Ableitung durch den *Gradient* beschrieben.

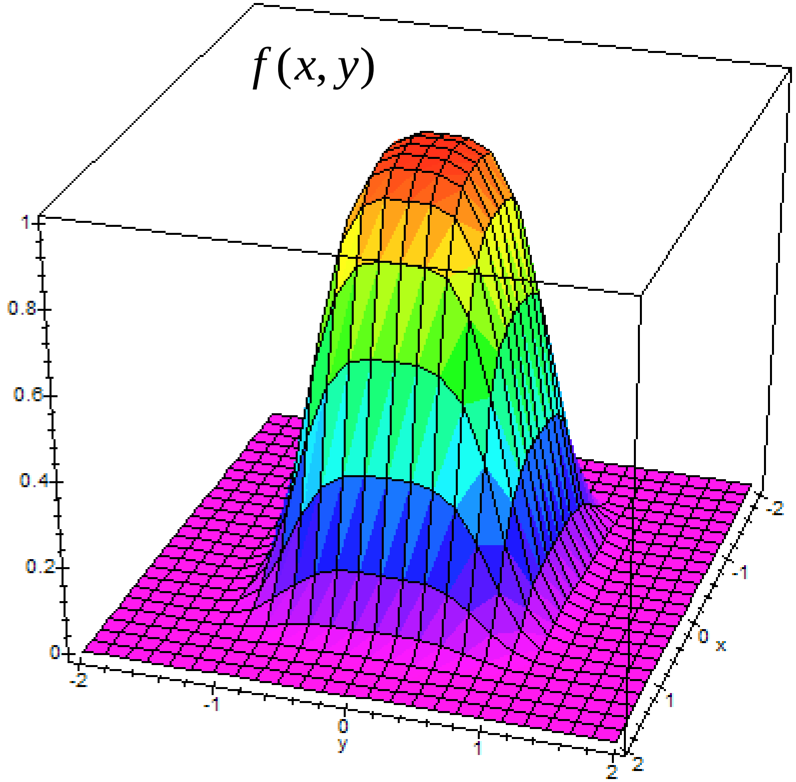
Der *Gradient* ist ein Vektor, der die Richtung und Stärke der Steigung im Grauwertgebirge angibt.

3.5.3.2 Mathematische Grundlagen

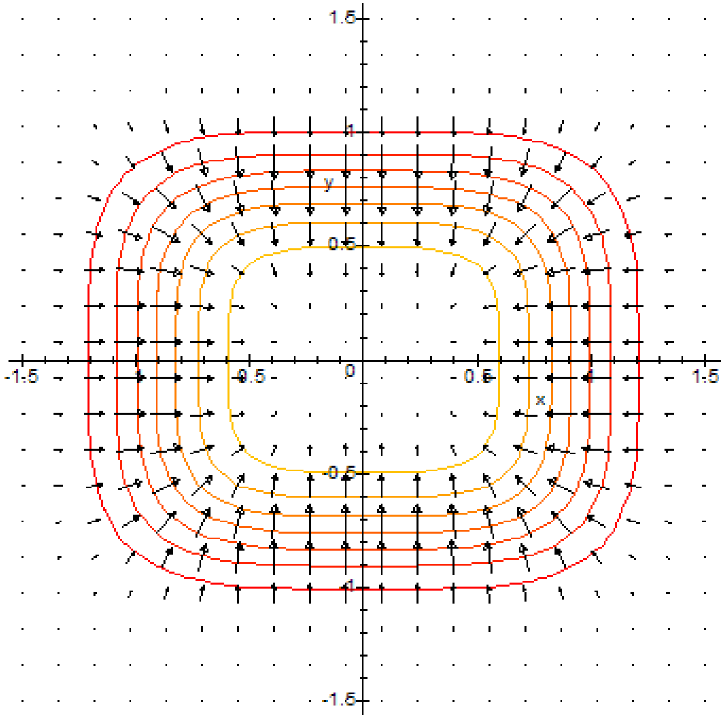
Gradient: Den Gradienten einer skalaren Funktion f(x,y) erhält man mit:

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right]^T \quad x, y \in \mathbf{R}$$

Der Gradient ist ein **Vektor**(-feld) !!



Gradientenfeld von f(x,y)



ÜBUNG: Berechnung des Gradienten

Gegeben sei folgende Funktion (zweier Variabler):

$$f(x,y) = e^{-(x^8+y^8)}$$

Berechnen Sie den Gradienten $\nabla f(x,y)$ an den Stellen:

a) x=0.0, y=0.0

b) x=1.0, y=0.0

c) x=0.0, y=1.0

d) x=1.0, y=1.0

3.5.3.3 Anwendung auf diskrete Funktionen $x, y \in \mathbf{G}$

Da bei Bildern x und y diskret sind, wird der Differentialquotient im Gradient zum Differenzenquotienten und man erhält für die Ableitungen: → s. Tafel

$$G_x = \frac{\partial f}{\partial x} \approx \frac{f(x+1) - f(x-1)}{2}$$

$$G_y = \frac{\partial f}{\partial y} \approx \frac{f(y+1) - f(y-1)}{2}$$

Damit erhält man die Faltungsmasken für die Ableitungen in x- und y-Richtung:

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

$\mathbf{G_x}$

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

$\mathbf{G_y}$

Anmerkung: Zur besseren Rauschunterdrückung wird über 3 Ableitungen gemittelt.

==> *Sobel*-Operator

Den Betrag des Gradienten erhält man mit:

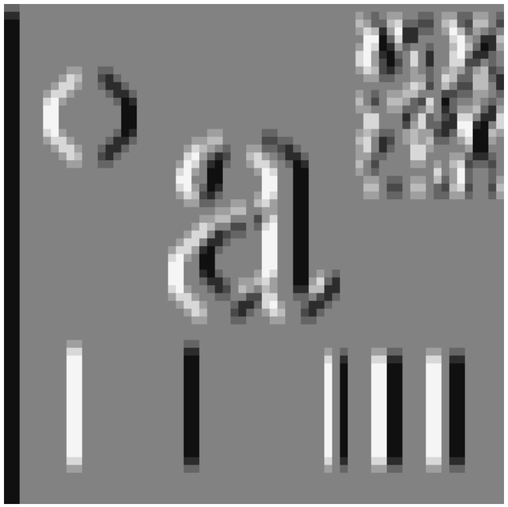
$$G = \sqrt{G_x^2 + G_y^2}$$

die Richtung des Gradienten mit:

$$\alpha = \arctan\left(\frac{G_y}{G_x}\right)$$

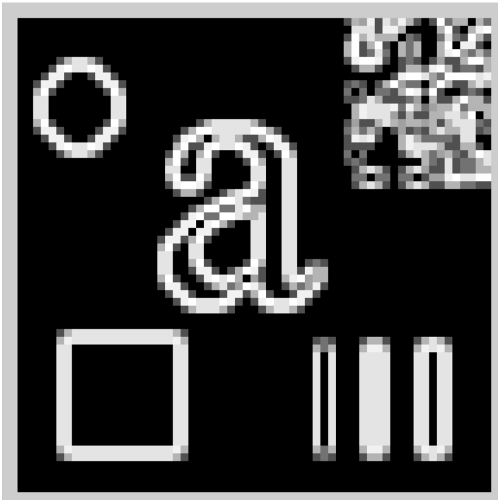
Beispiele:

$G_x(x,y)$



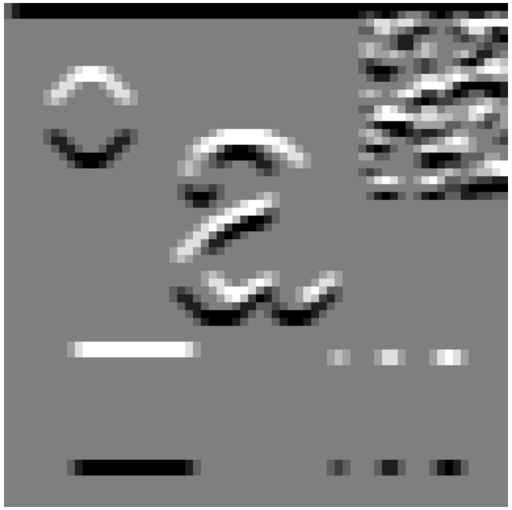
Grau: 0
dunkel: -
hell: +

f (x,y)



|G (x,y)|

$G_y(x,y)$



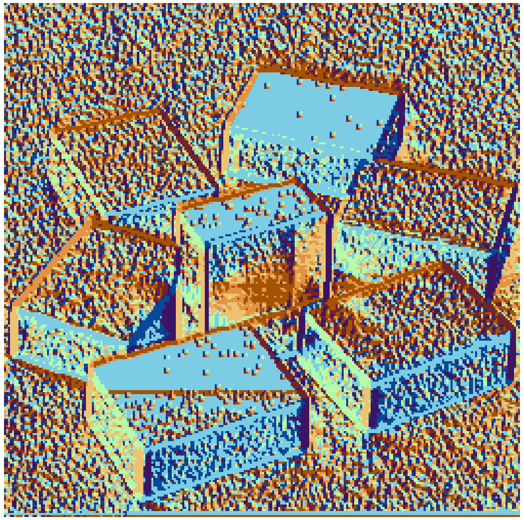
3.5.3.4 Abschließende Bemerkungen

- *Sobel-Filter* ist einfacher Kantenfilter.
- Neben dem Gradientenbetrag kann auch die Gradientenrichtung als Bild ausgegeben werden. Dies kann hilfreich für nachfolgende Verfahren sein (z.B. Liniendetektion mit Hilfe der Houghtransformation).
- Varianten: Robinson-Filter,
  Kirsch-Filter,
  Prewitt-Filter. (s. "Methoden der digitalen Bildsignalverarbeitung", Zamperoni, Vieweg)

Originalbild

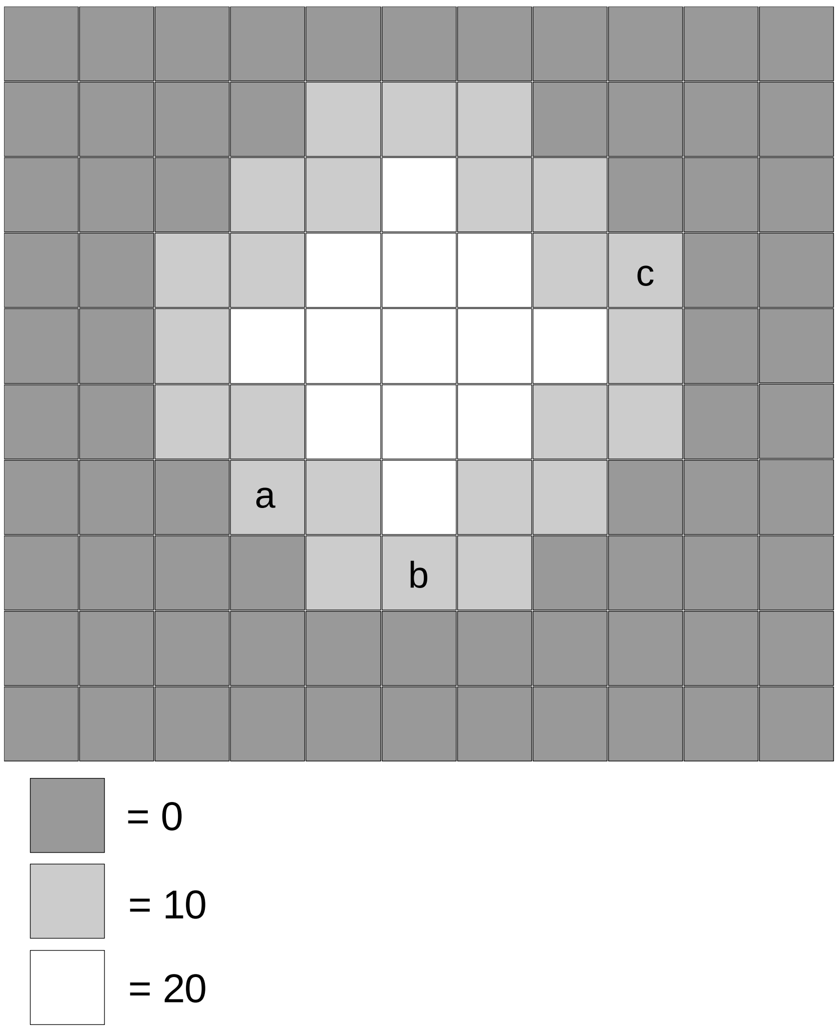
Gradientenbetrag

Gradientenrichtung



Anm.: Unterschiedliche Farben codieren unterschiedl. Richtungen.

ÜBUNG: Sobel-Filter



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

*1/4

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

*1/4

Geben Sie den Gradienten und die Gradientenrichtung für die Punkte a, b und c an.

Was passiert im homogenen Bereich?

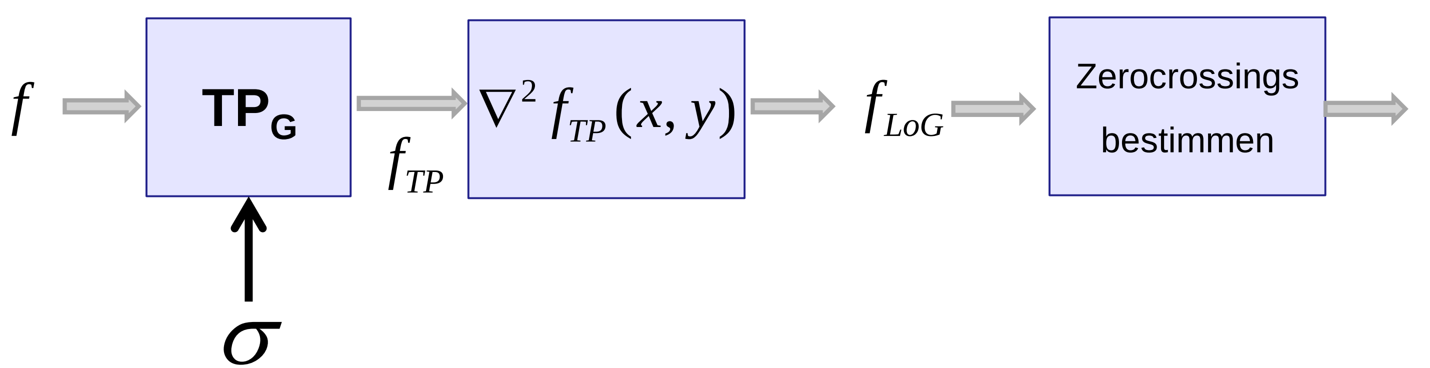
3.5.4 LoG-Filter: Laplace - of - Gaussian

3.5.4.1 Ziele

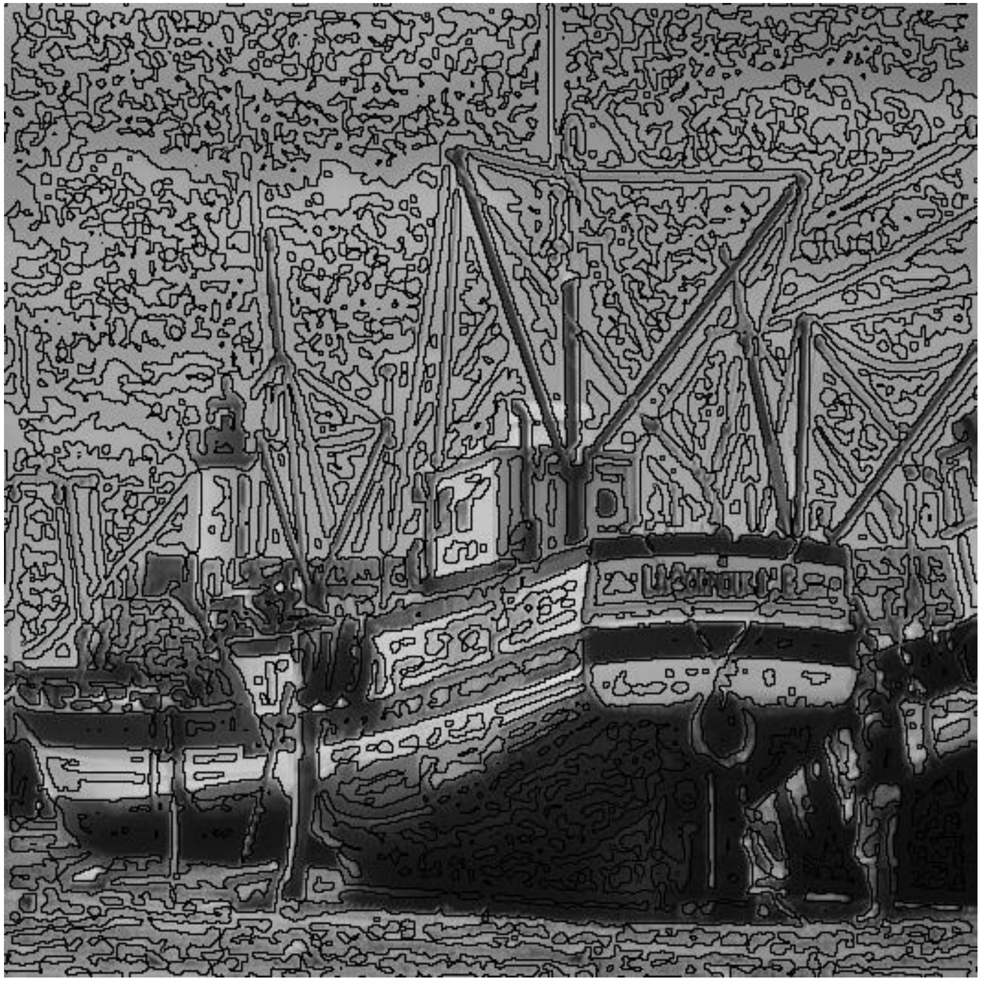
- Einfluß des Bildrauschens vermindern
- einstellbarar Detailgrad

Ansatz: 1. Bild glätten (Gaussfilter)
2. Laplace-Filter (Helligkeitswendepunkte sind Kanten)
3. Zerocrossings bestimmen

→ *LoG-Filter (mexican-hat-operator)*.



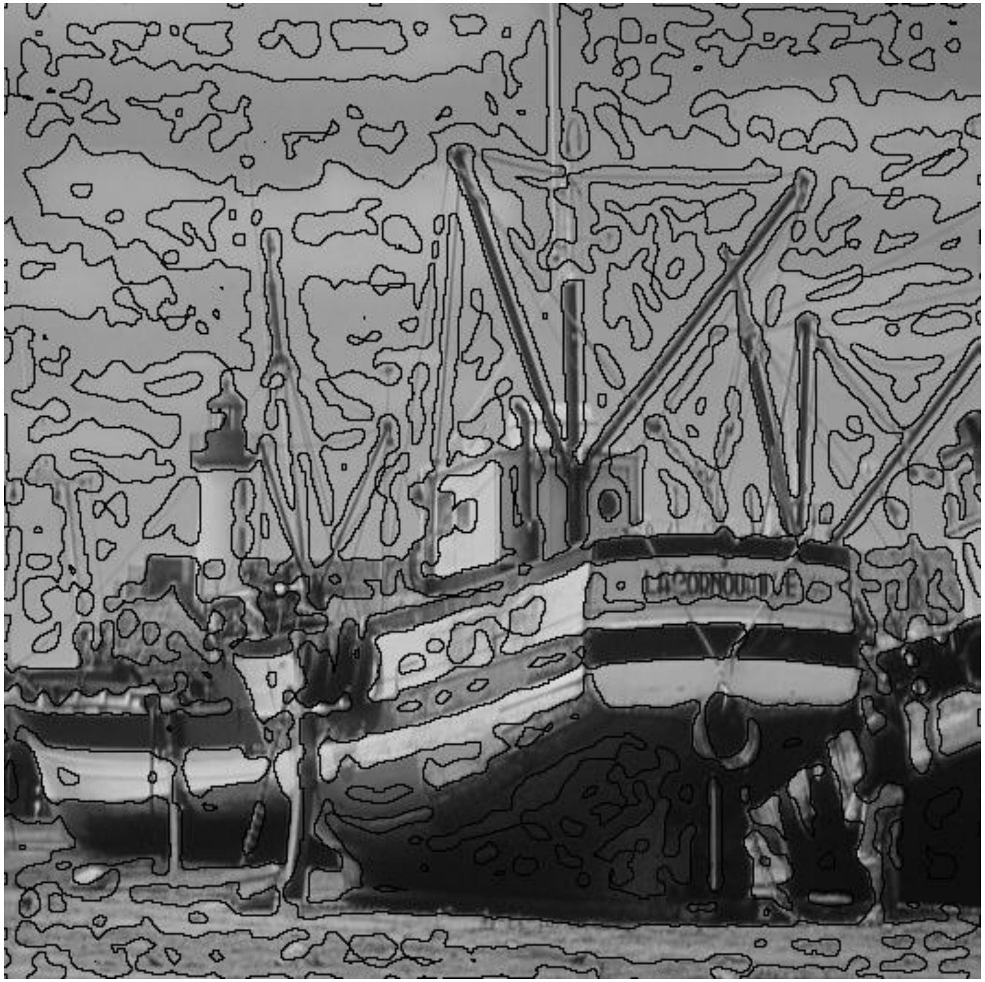
Anwendung: Kantenfilter mit einstellbarem Detailgrad



Original mit eingezeichneten *Zerocrossings*

= Helligkeitswendepunkte im Bild

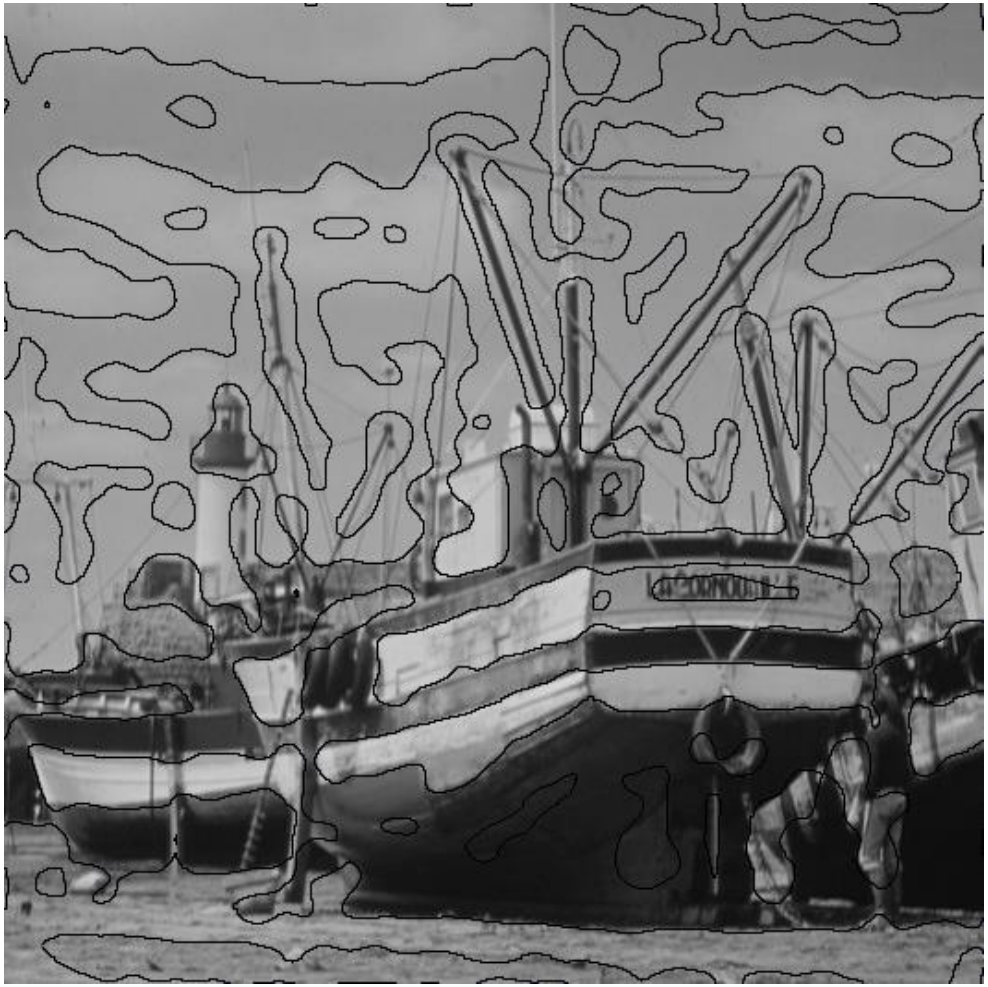
σ=2.0



Original mit eingezeichneten *Zerocrossings*

= Helligkeitswendepunkte im Bild

σ=4.0



Original mit eingezeichneten *Zerocrossings*

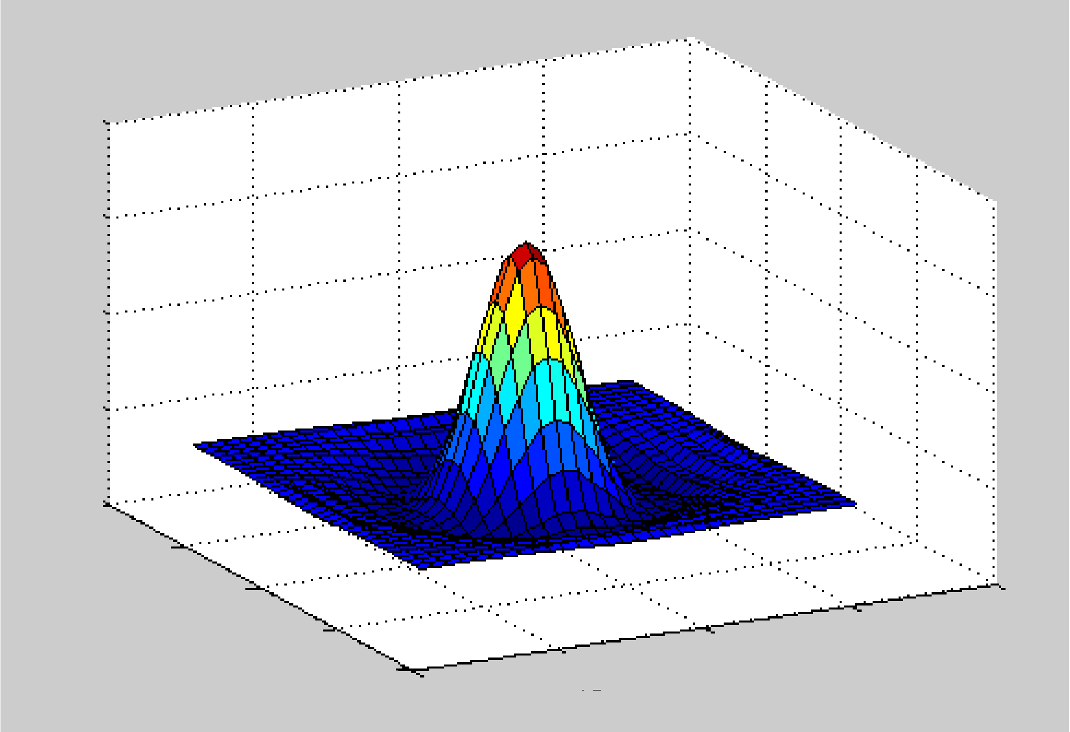
= Helligkeitswendepunkte im Bild

σ=8.0

3.5.4.2 Durchführung der Filterschritte

1. Gaussfilter anwenden
   a) 2x1D-Filterung (Separationsfähigkeit nutzen)
   b) Floatingpointkern nutzen (höhere Genauigkeit)
2. Laplace-Filter anwenden

Die Zusammenfassung der Filterschritte führt auf Faltungskerne der Form:

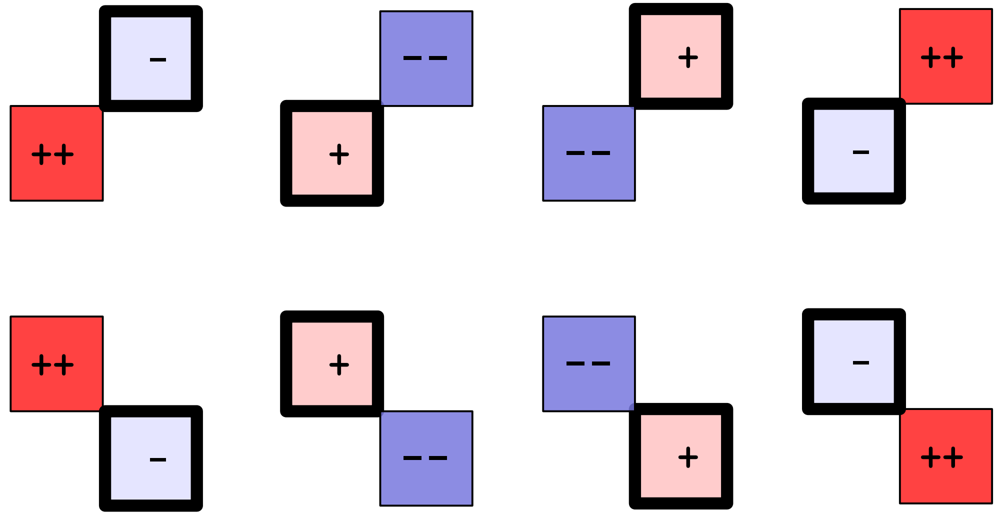


→ mexican-hat-Operator

3.5.4.3 Bestimmung der Zerocrossings (pixelgenau)

Nebenstehende Vorzeichenwechsel diagonaler Nachbarbildpunkte werden abgeprüft.

Tritt ein Vorzeichenwechsel auf, so wird dem Bildpunkt mit dem betrags-mäßig kleineren Wert (+ oder -) das Label „zerocrossing“ zugewiesen (**Anm.**: dick umrandet)



for each Bildpunkte $f_{LoG}(x,y)$ des Bildes

  if $f_{LoG}(x,y) * f_{LoG}(x+1,y+1) < -Threshold$

    if $| f_{LoG}(x,y) | < | f_{LoG}(x+1,y+1) |$ **then** $f(x,y)$ = ZEROCROSSING

    else $f(x+1,y+1)$ = ZEROCROSSING

  if $f_{LoG}(x,y) * f_{LoG}(x+1,y-1) < -Threshold$

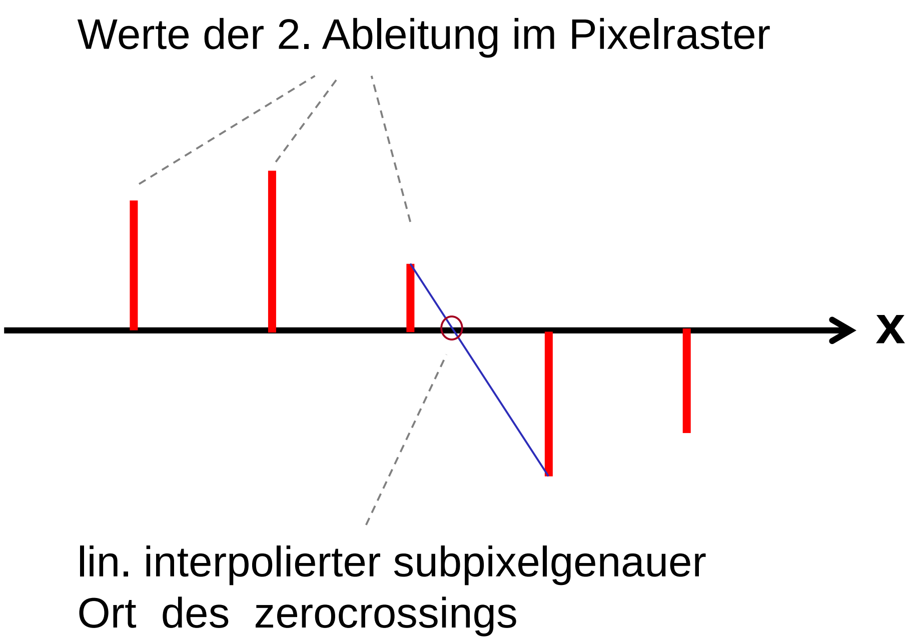
    if $| f_{LoG}(x,y) | < | f_{LoG}(x+1,y-1)|$ **then** $f(x,y)$ = ZEROCROSSING

    else $f(x+1,y-1)$ = ZEROCROSSING

3.5.4.4 Abschließende Anmerkungen

LoG-Filter spielt eine große Rolle beim menschlichen Stereosehen (s. Marr, Poggio, Hildreth, Grimson).

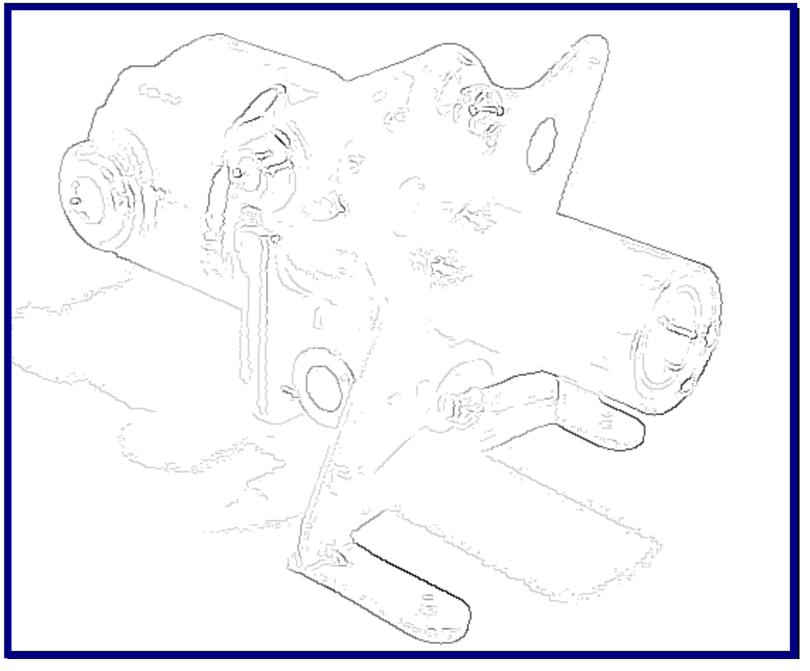
Die zerocrossings können subpixelgenau verfeinert werden.



3.5.5 Canny-Filter

3.5.5.1 Ziele

- Detektion aller relevanten Kanten (Grauwertgradient-basiert, vgl. Sobel-Operator).
- Der detektierte Kantenort soll möglichst nah am tatsächlichen Kantenort liegen.
- Eine Bildkante sollte auch nur eine (ein Pixel breite) detektierte Kante zur Folge haben, insbesondere auch im Hinblick auf Bildrauschen.



Weitere Eigenschaften:

- Detaillierungsgrad einstellbar
- Kantenrelevanz spiegelt sich in der Intensität wider

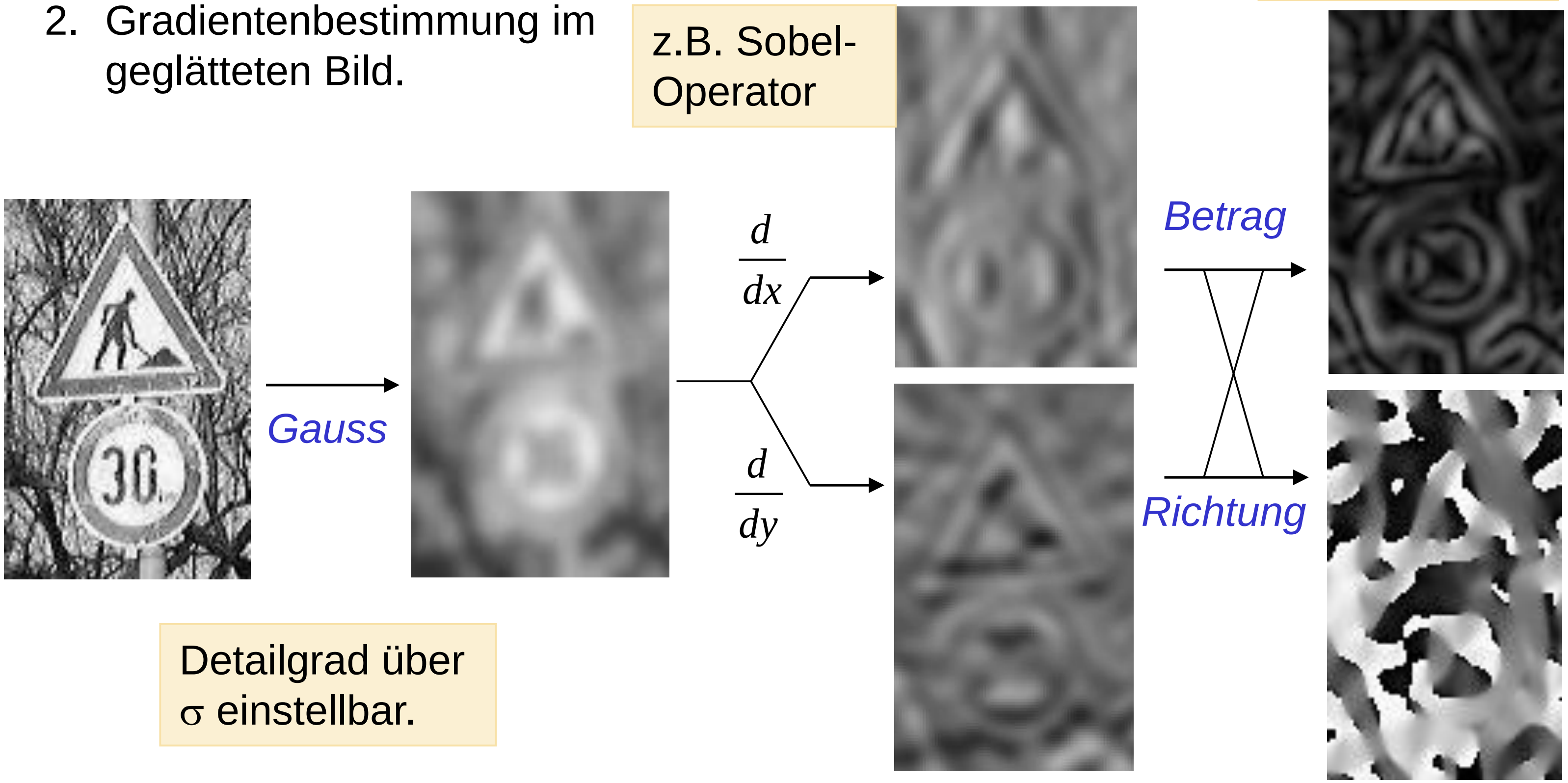
→ einer der meistverwendeten Kantenoperatoren

3.5.5.2 Prinzip

Die Kanten werden in mehreren Verarbeitungsschritten extrahiert:

1. Bildglättung mit Hilfe des Gauss-Operators,
2. Gradientenbestimmung im geglätteten Bild.

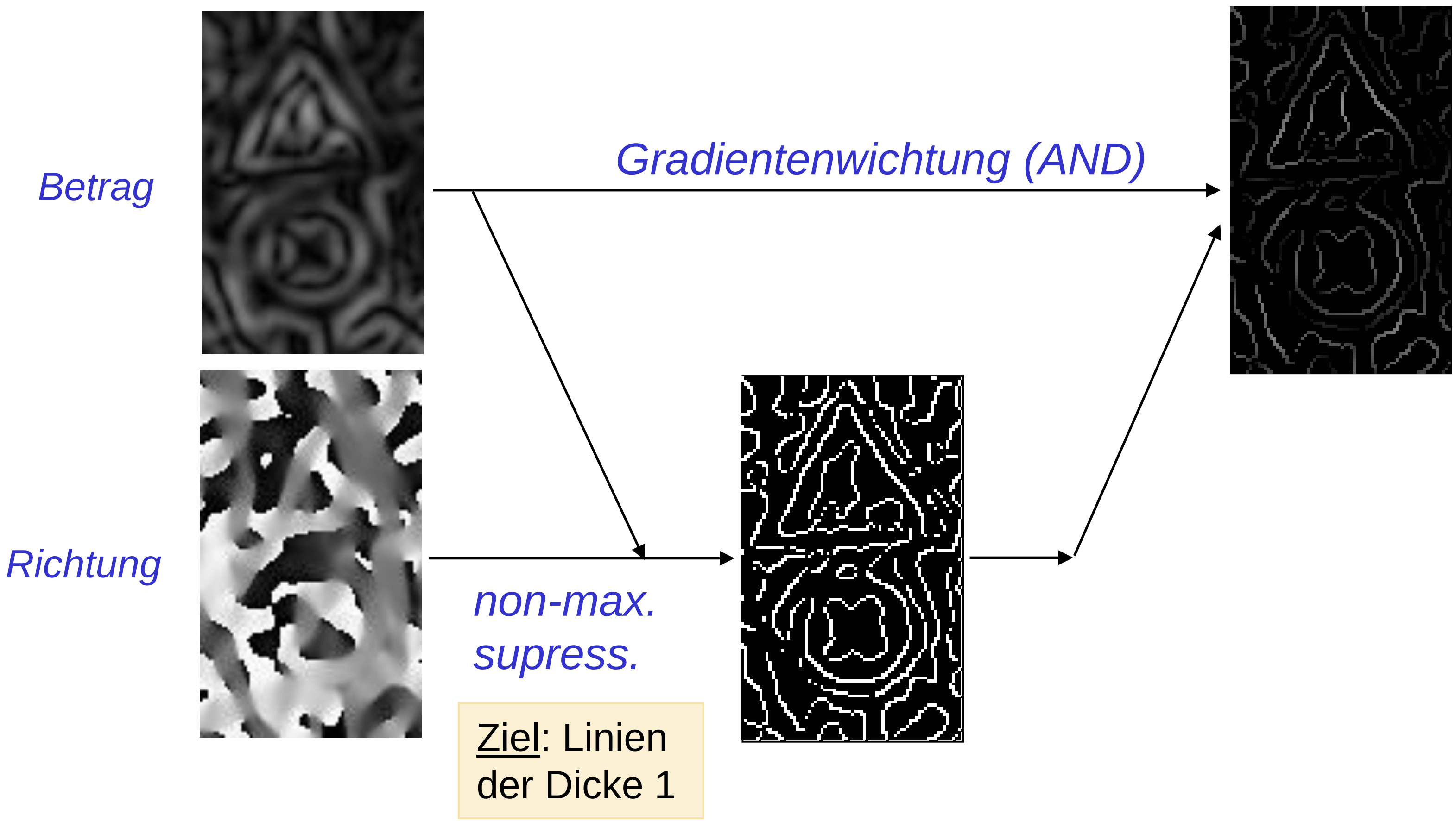
$$G = \sqrt{G_x^2 + G_y^2}$$



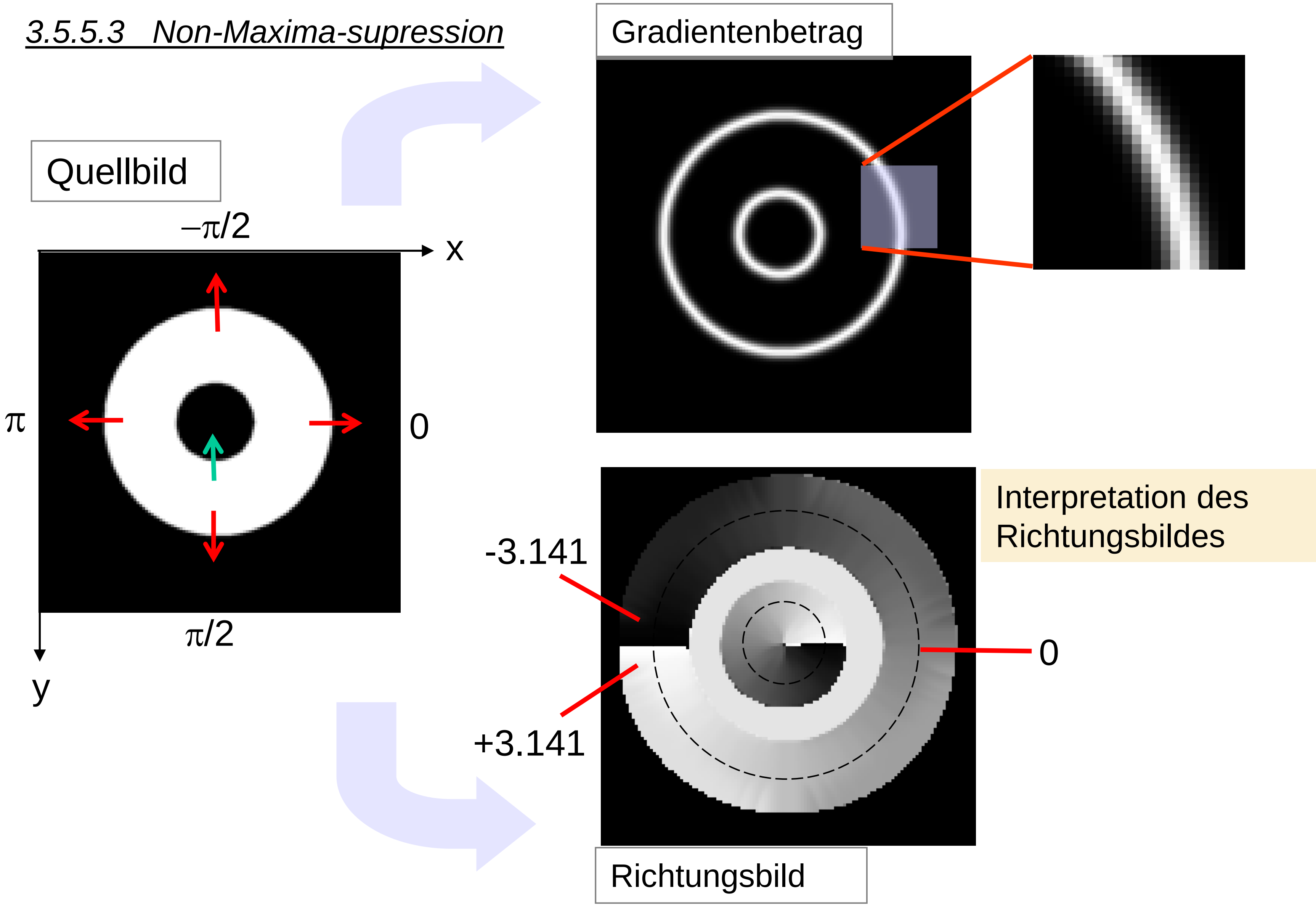
Detailgrad über σ einstellbar.

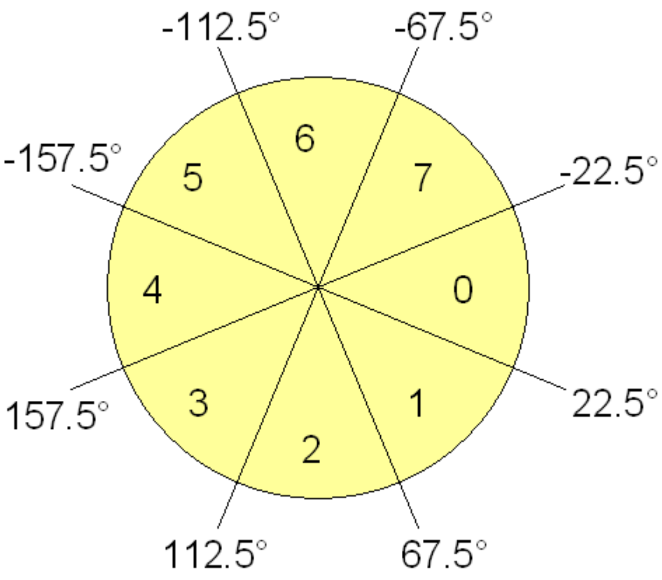
$$\alpha = \arctan\left(\frac{G_y}{G_x}\right)$$

3. Berechnung des Betrags- und Richtungsbildes aus dem Gradienten.
4. Bestimmung aller Maxima im Gradientenbetragsbild senkrecht zur Kantenrichtung (s. nachfolgende Seite).

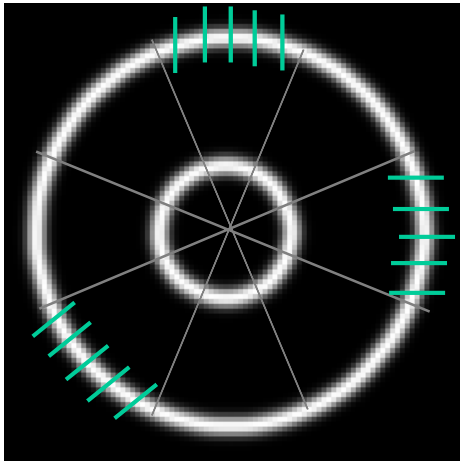


3.5.5.3 Non-Maxima-supression

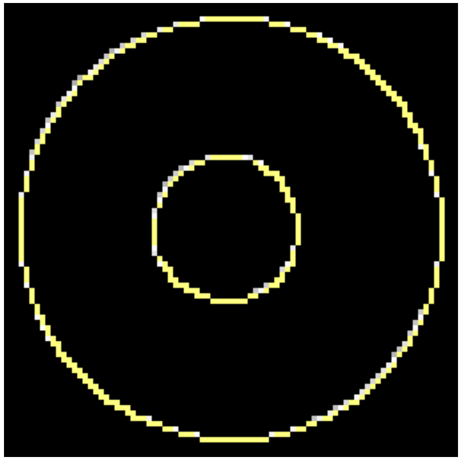




g_5	g_6	g_7
g_4	g_M	g_0
g_3	g_2	g_1



Maxima → 255
sonst → 0



Kantenrichtung	Maximum, wenn im Betragsbild gilt
0 oder 4	$g_M >= g_0$ AND $g_M >= g_4$
1 oder 5	$g_M >= g_1$ AND $g_M >= g_5$
2 oder 6	$g_M >= g_2$ AND $g_M >= g_6$
3 oder 7	$g_M >= g_3$ AND $g_M >= g_7$

Bild

Canny (ungewichtet) mit unterlegtem Bild

3.5.5.4 Nachbereitung: Gabelpunkte auftrennen

Für nachfolgenden Verarbeitungsstufen kann es vorteilhaft sein, zusammenhängende Kantenpunkte einzusammeln und in eine Teilkontur-Datenstruktur zu überführen.

Dies setzt voraus, dass sich Kantenverläufe nicht aufgabeln.

Algorithmus:

Die in den nebenstehenden Masken angegebenen Bild-strukturen werden gesucht.

(X → ungleich 0)

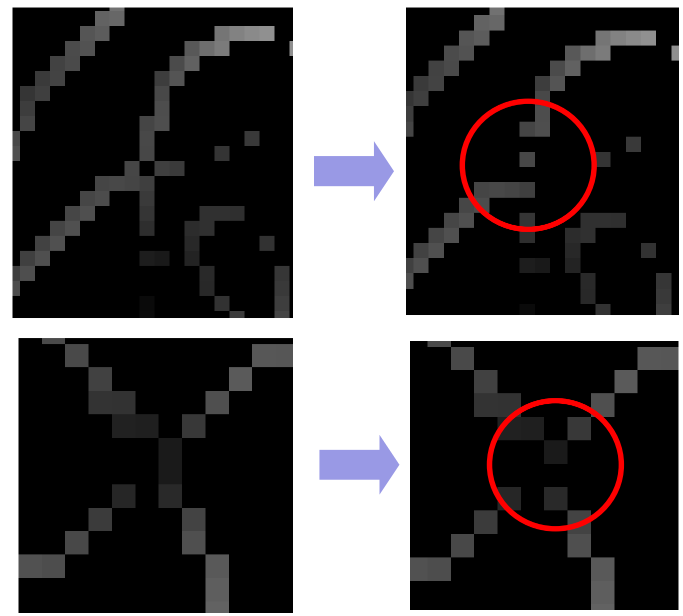
0	X	0
X	X	X
0	0	0

Die rot markierten Grauwerte werden zu 0 gesetzt.

0	X	0
0	X	X
X	0	0

Analog wird mit den um 90°, 180° und 270° verdrehten Masken verfahren.

X	0	X
0	X	0
0	X	0

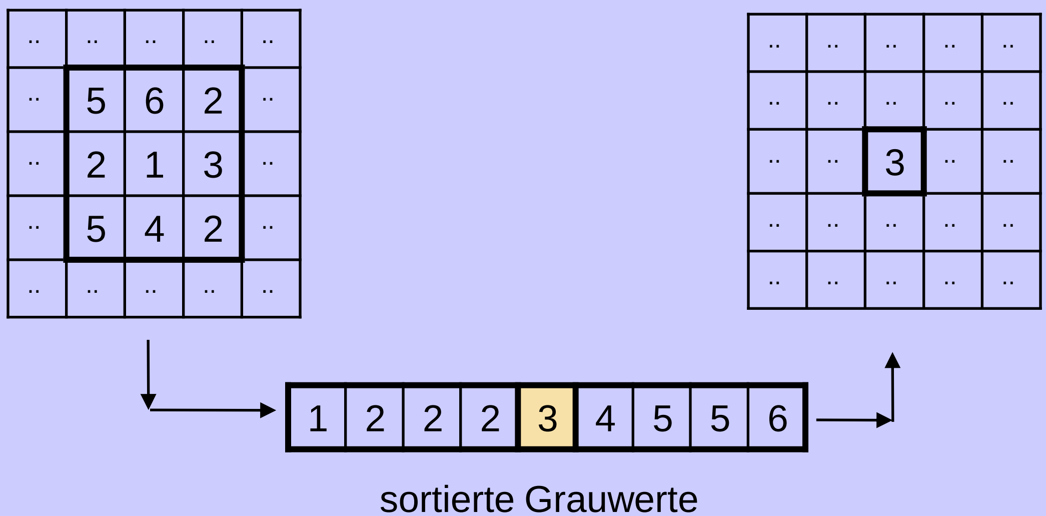


3.6 Rangordnungsoperatoren

3.6.1 Median-Operator

Grundgedanke: Die Grauwerte innerhalb einer Bildpunktumgebung (z.B. quad. Filtermaske) werden der Größe nach sortiert. Der in der in dieser Anordnung in der Mitte stehende Wert wird als Zielgrauwert ausgegeben.

Beispiel: 3x3-Medianfilter



Anwendungen:

- Kantenerhaltende Bildglättung
- Entfernen von "Salt-and-Pepper-Noise"
- Unterdrückung bzw. Hervorhebung von Bildstrukturen einer bestimmten Größe

ÜBUNG: Wirkung des Medianfilters auf verschiedene Bildstrukturen

Diskutieren Sie die Wirkung des Median-Filters auf

a) Einzelpunkte,
b) dünne Linien,
c) Ecken,
d) Kanten.

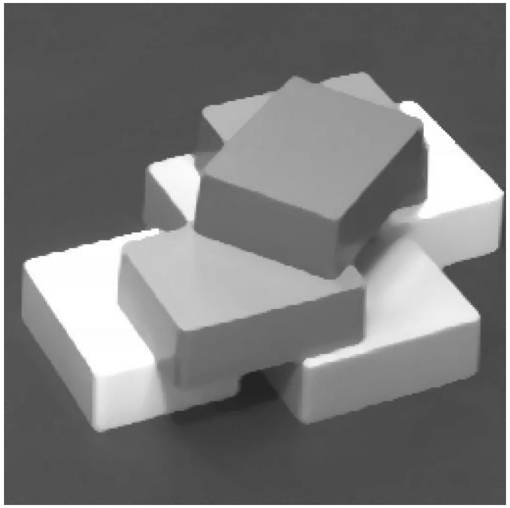
Beispiel: Median 5x5-Maske

Beispiel: Median 9x9-Maske

Beispiel: Median 13x13-Maske

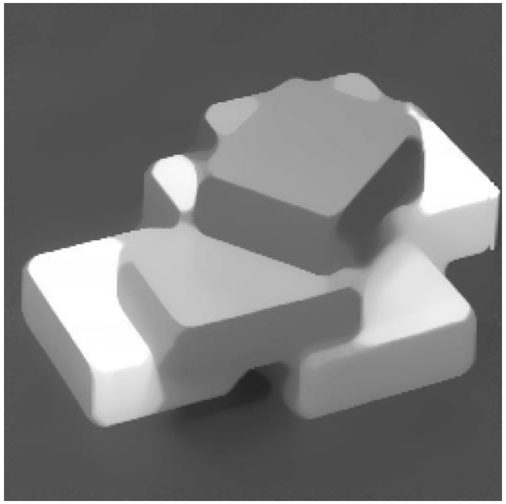
Beispiel: Eckendetektion (einfach und nicht sehr gut)

Median 3x3



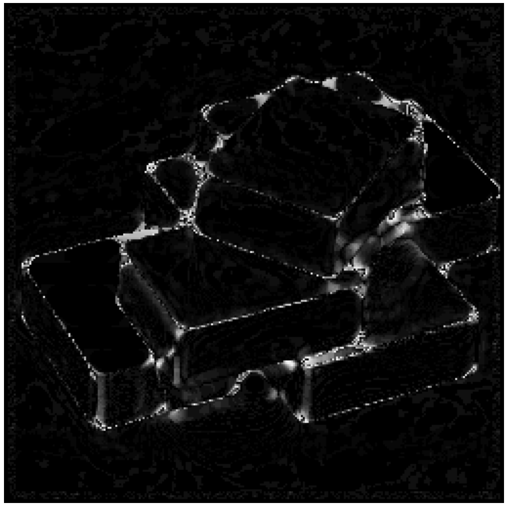
–

Median 7x7

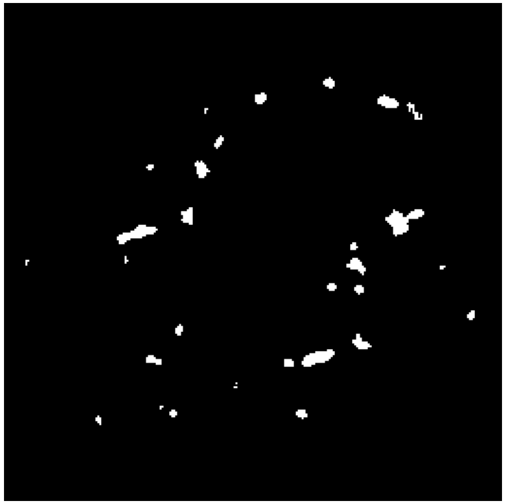


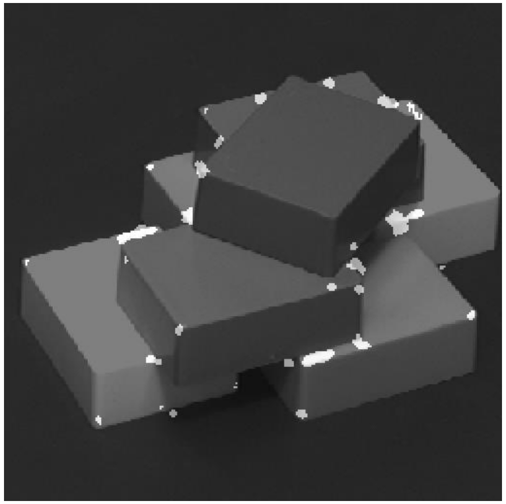
=

Differenzbild



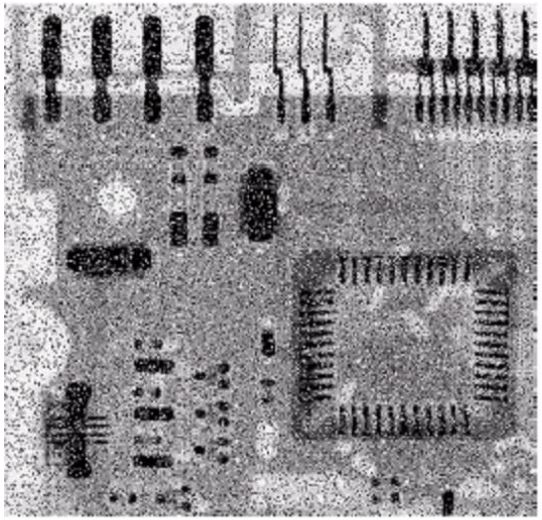
Median 3x3 vom Differenzbild
+ Binarisierung



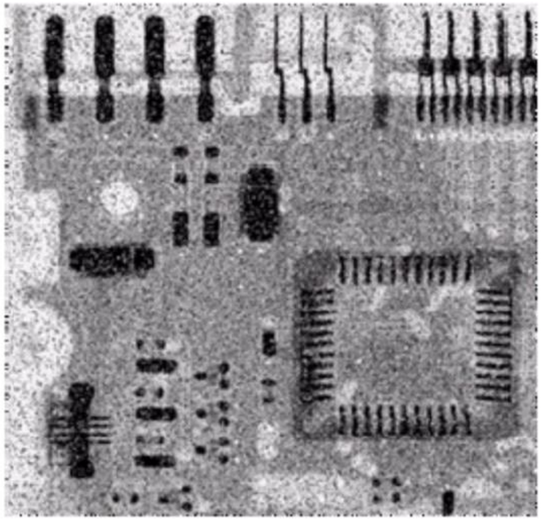


Beispiel: Entfernen von "Salt-and-Pepper"-Rauschen

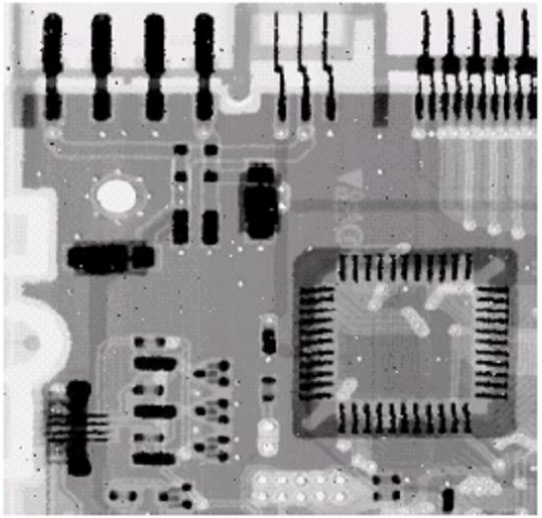
Originalbild



3x3-Rechteckmaske



3x3-Median

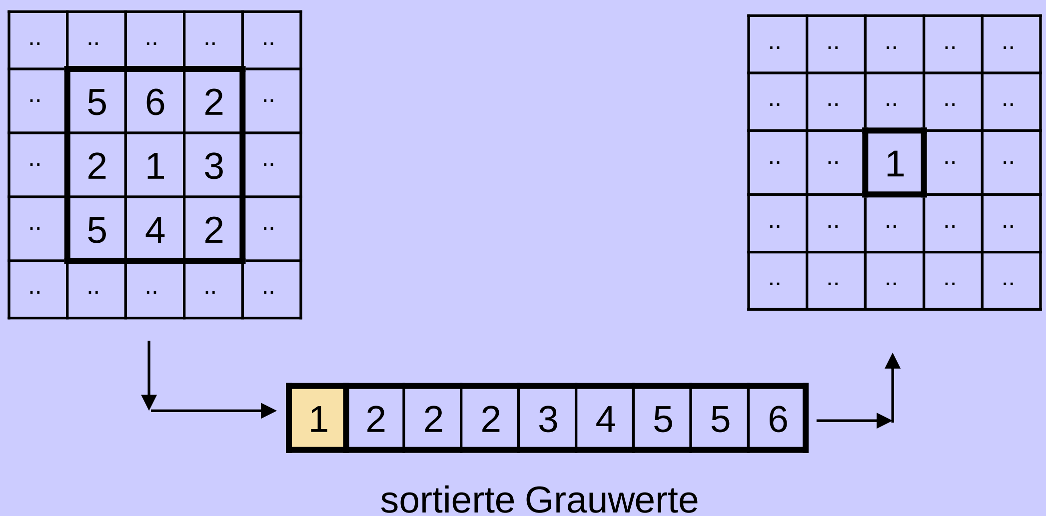


aus "Digital Image Processing", Gonzalez/Woods

3.6.2 Minimum-Operator

Grundgedanke: Innerhalb einer Bildpunktumgebung (z.B. quad. Filtermaske) wird der minimale Grauwert bestimmt. Dieser wird als Zielgrauwert ausgegeben.

Beispiel: 3x3-Minimum-Filter

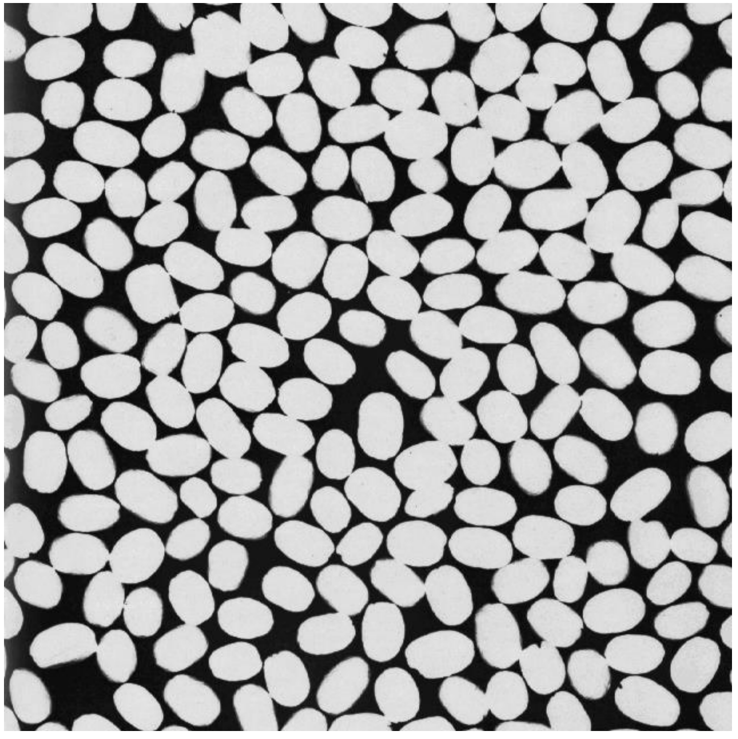


Anwendungen:

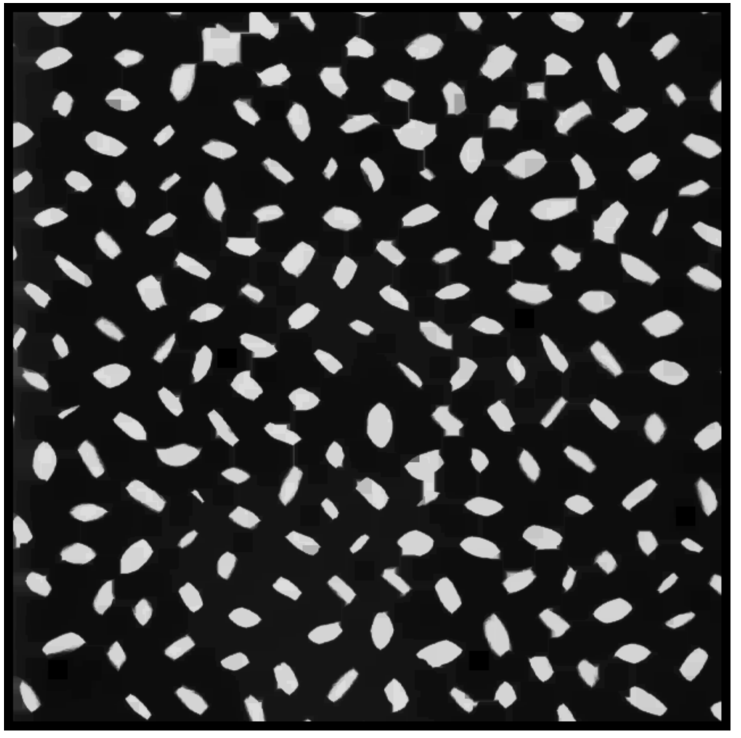
- Unterdrückung bzw. Hervorhebung von Bildstrukturen einer bestimmten Größe
- Basis vieler "Morphologischer Operatoren" (folgt später)

Beispiel: Separation von Partikeln

Partikel (sich berührend)



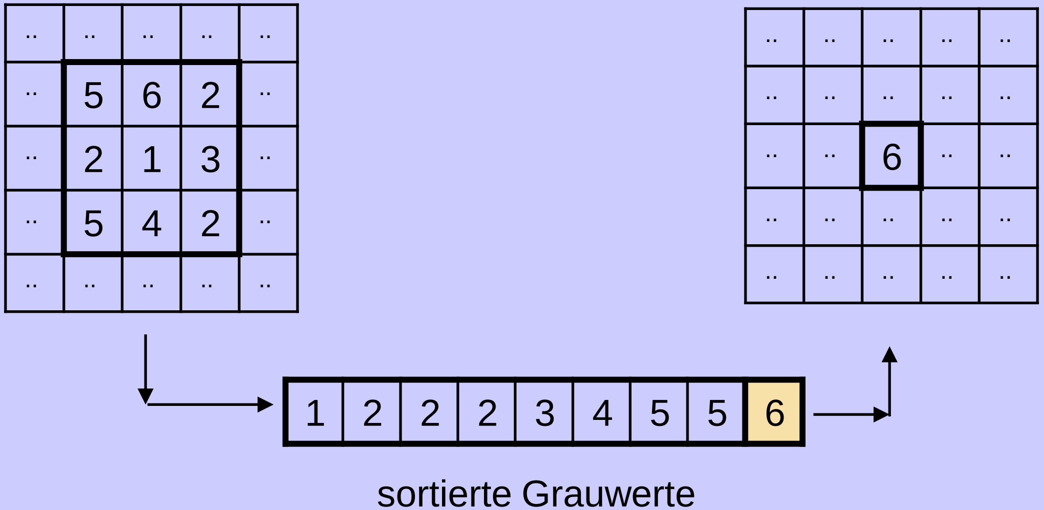
4x Anwendung des 5x5-Minimumop.



3.6.3 Maximum-Operator

Grundgedanke: Innerhalb einer Bildpunktumgebung (z.B. quad. Filtermaske) wird der maximale Grauwert bestimmt. Dieser wird als Zielgrauwert ausgegeben.

Beispiel: 3x3-Maximum-Filter



Anwendungen:

- Unterdrückung bzw. Hervorhebung von Bildstrukturen einer bestimmten Größe
- Basis vieler "Morphologischer Operatoren" (folgt später)

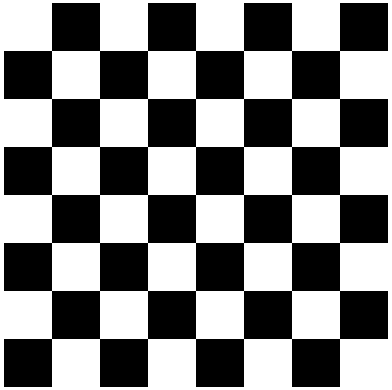
3.7 Geometrische Bildtransformationen

3.7.1. Einführung

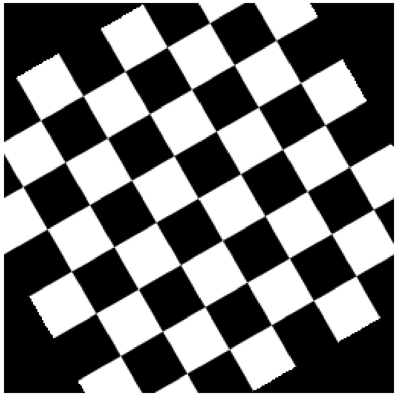
3.7.1.1 Zweck

→ Skalierung, Verschiebung, Drehung, Scherung und andere Verformungen des Bildes.

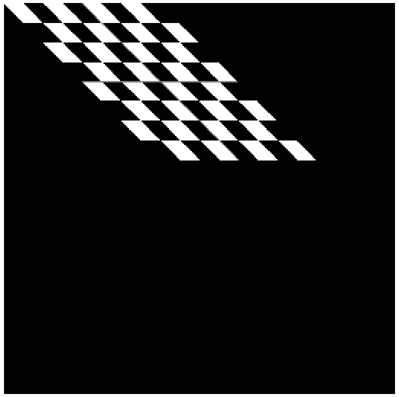
Originalbild



Rotation



Scherung u. Skalierung



Anwendungsbeispiele:

- Korrektur perspektivischer Verzeichnungen
- Korrektur von Linsenverzeichnungen

3.7.1.2 Ansatz 1: Direkte Methode → Source-to-Target Mapping :

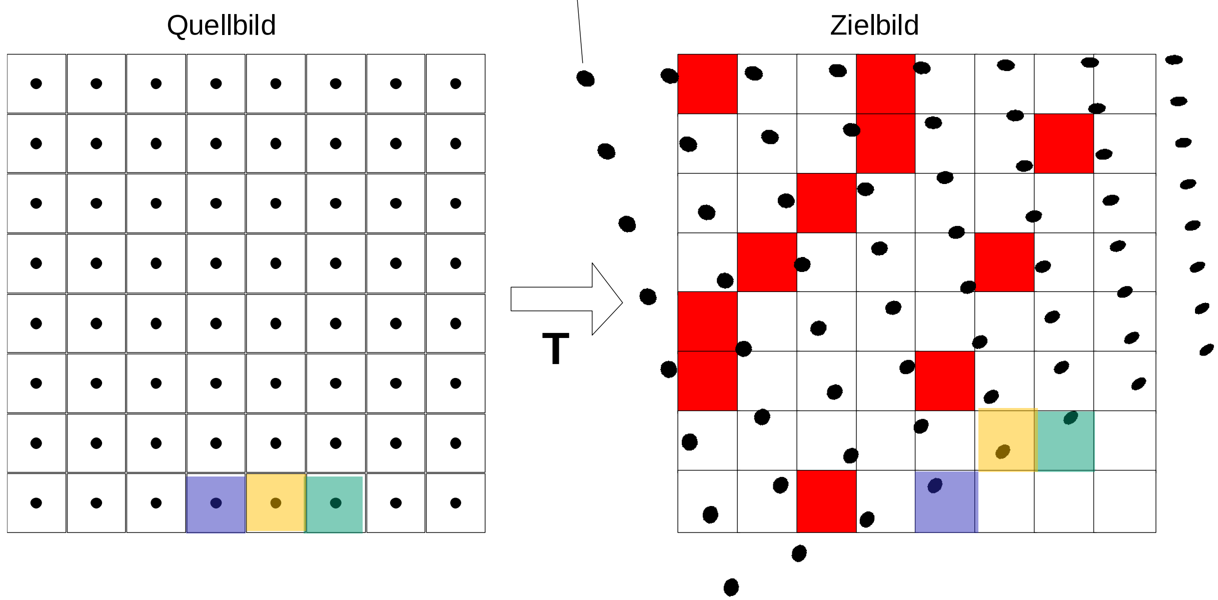
for each Bildpunkt des <u>Quell</u>bildes $q(x_q, y_q)$

$(\tilde{x}_z, \tilde{y}_z) \leftarrow \mathbf{T}\,(x_q, y_q)$ // Zielkoordinaten berechnen

$z(\mathrm{round}(\tilde{x}_z), \mathrm{round}(\tilde{y}_z)) \leftarrow q(x_q, y_q)$ // Grauwert ins Zielbild kopieren

Anm.: $x_q \in \mathbb{N}, \quad \tilde{x}_q \in \mathbb{R}$

transformiertes Quellbildpunktraster

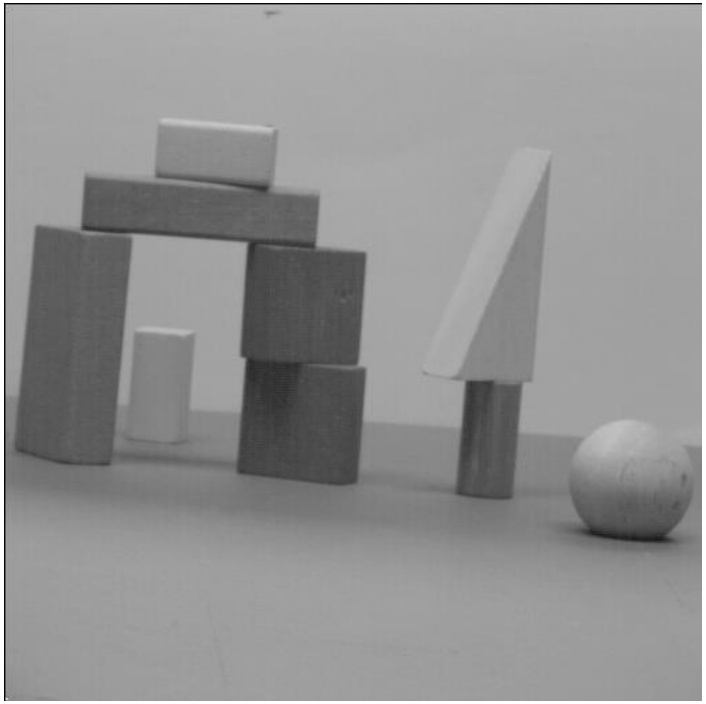


Problem:
Nicht jeder Zielbild-punkt hat einen korrespondierenden Quellbildpunkt.

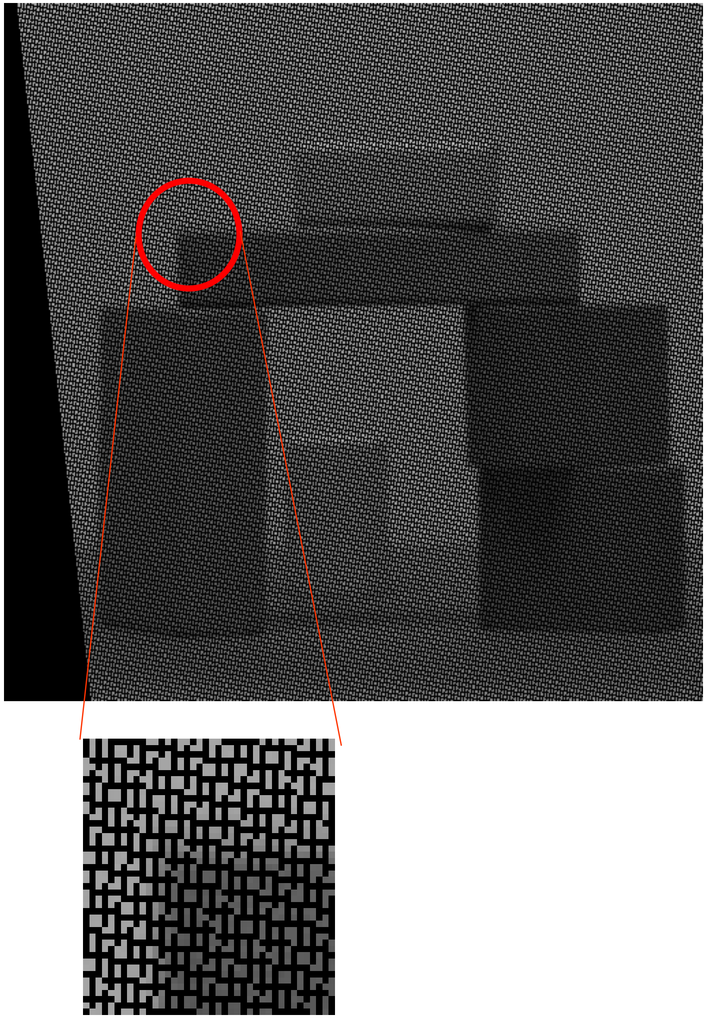
• Quellbildpunkte (diskretes Raster)

Zielbildpunkte ohne Quellbildpunkt

Beispiel: Ergebnis der direkten Transformation (rotiert und vergrößert)



Originalbild



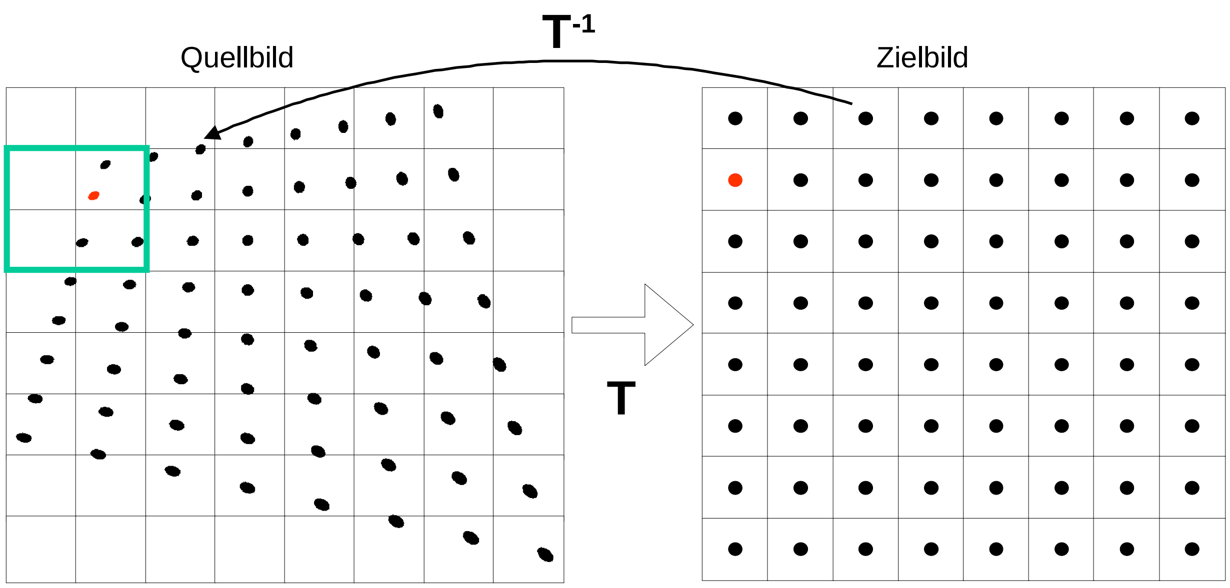
3.7.1.3 Ansatz 2: Indirekte Methode → Target-to-source Mapping:

for each Bildpunkt des <u>Ziel</u>bildes $z(x_z, y_z)$

$(\tilde{x}_q, \tilde{y}_q) \leftarrow \mathbf{T}^{-1}(x_z, y_z)$ // Quellkoordinaten berechnen

$z(x_z, y_z) \leftarrow \text{Interpol}\,(q(\tilde{x}_q, \tilde{y}_q))$ // Grauwert ins Zielbild kopieren

Anm.: $x_q \in \mathbb{N}$, $\tilde{x}_q \in \mathbb{R}$



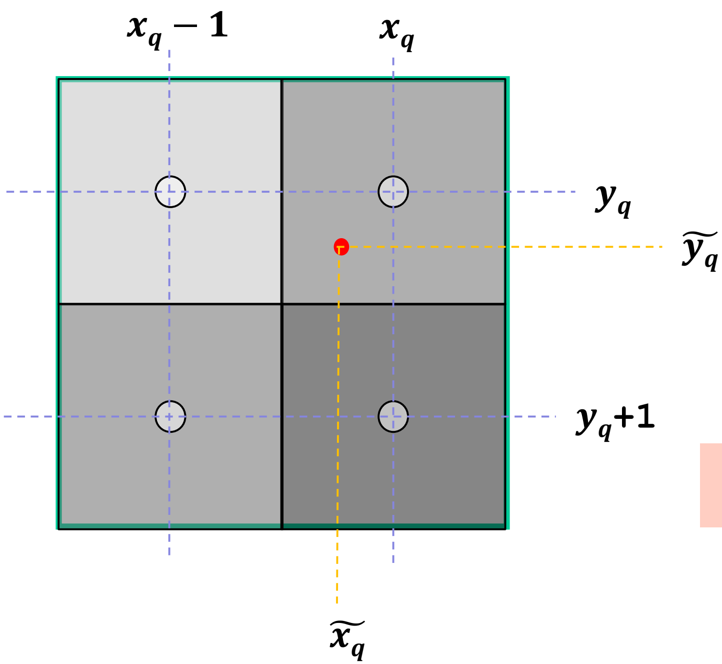
Offene Fragen:

a) Es wird die Umkehrfunktion $\mathbf{T}^{-1}$ der Transformationsvorschrift **T** benötigt.
b) Wie bestimmt man den Zielgrauwert aus dem/den Quellgrauwert(en)?

Bestimmung des Zielgrauwertes aus den Quellgrauwerten

Ansatz 1: „*Rundung*"

Der Zielgrauwert wird aus dem Quellbildpunkt entnommen, dessen Koordinate am dichtesten an der berechneten Quellkoordinate liegt.



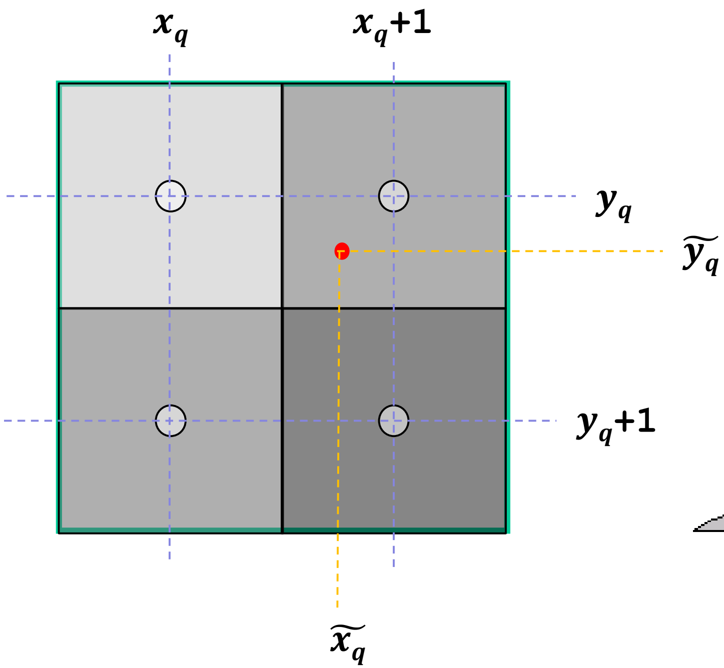
Quellbild-Ausschnitt
(siehe vorherige Seite)

Anm.: $x_q \in \mathbb{N}$, $\tilde{x}_q \in \mathbb{R}$

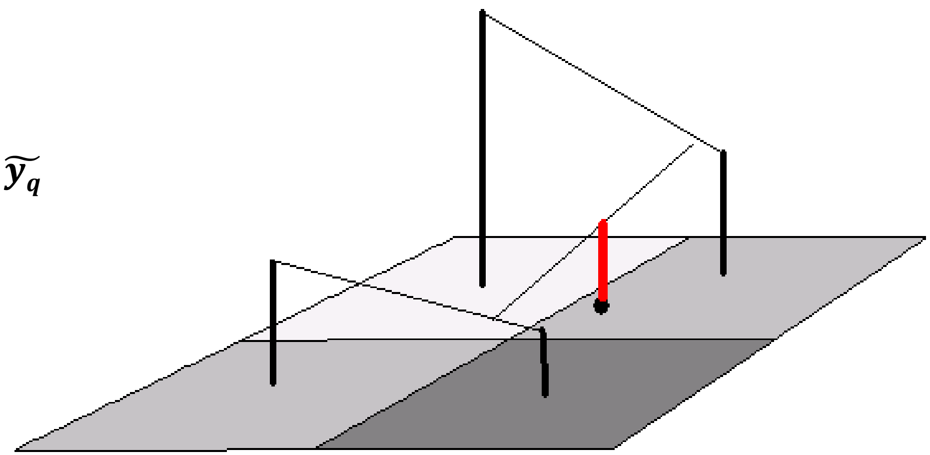
$$\boldsymbol{z}(\boldsymbol{x_z}, \boldsymbol{yz}) \leftarrow \boldsymbol{q}(\mathrm{round}(\tilde{\boldsymbol{x}}_q), \mathrm{round}(\tilde{\boldsymbol{y}}_q))$$

Ansatz 2: "*Bilineare Transformation*"

Der Zielgrauwert wird durch Interpolation aus den vier Quellnachbarn berechnet.



Quellbild-Ausschnitt
(siehe vorherige Seite)



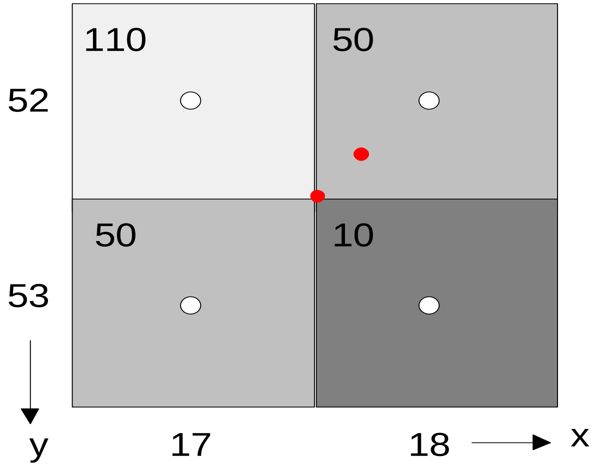
ÜBUNG: Bilineare Interpolation

1. Gegeben seien die Grauwerte f(x) und f(x+1) zweier aufeinanderfolgender Bildpunkte in einem 1-dimensionalen Bild (Zeilenkamera).
   Geben Sie eine Formel an, mit der für die Position

   $$\tilde{x} \ (\text{mit} \quad x <= \tilde{x} <= x+1, \quad \tilde{x} \in \mathrm{R}, \quad x \in \mathrm{N})$$

   der Grauwert interpoliert werden kann.

2. Gegeben sei folgender Quellbildausschnitt (4 Bildpunkte).



Geben Sie für die Positionen

a) $(\tilde{x}, \tilde{y}) = (17.5,\ 52.5)$

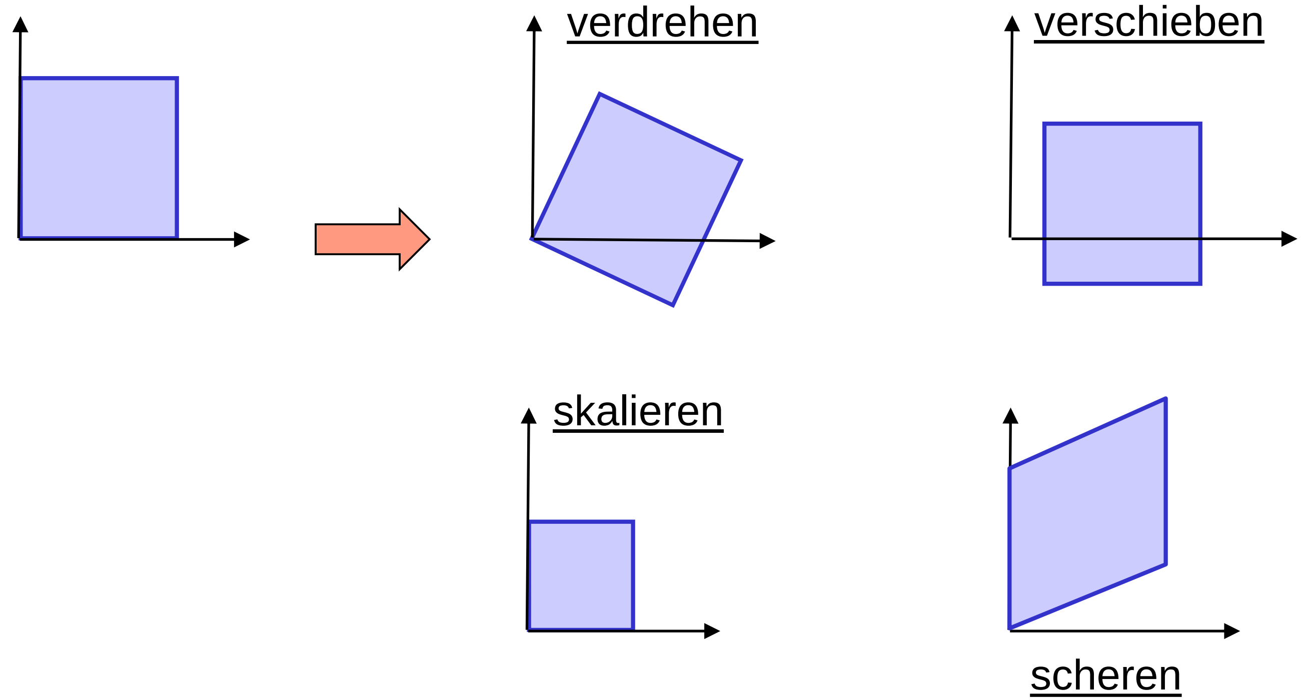
b) $(\tilde{x}, \tilde{y}) = (17.7,\ 52.4)$

mit Hilfe der *bilinearen Interpolation* den interpolierten Grauwert an.

3.7.2 *Affine Transformation*

3.7.2.1 Eigenschaften

Mit der *affinen Transformation* können folgende Transformationen durchgeführt werden:



Die affine Transformation ist Parallelen-erhaltend.

3.7.2.2 Affine Transformation für Source-target-Mapping

Basis der affinen Transformation sind lineare Polynome :

$$x_z = a_0 + a_1 \cdot x_q + a_2 \cdot y_q$$
$$y_z = b_0 + b_1 \cdot x_q + b_2 \cdot y_q$$

bzw. in Matrixform

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \cdot \begin{bmatrix} x_q \\ y_q \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

3.7.2.3 Affine Transformation für Target-source-Mapping

$$x_q = A_0 + A_1 \cdot x_z + A_2 \cdot y_z$$
$$y_q = B_0 + B_1 \cdot x_z + B_2 \cdot y_z$$

bzw. in Matrixform

$$\begin{bmatrix} x_q \\ y_q \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix} \cdot \begin{bmatrix} x_z \\ y_z \end{bmatrix} + \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}$$

3.7.2.4 Verschiedene Spezialfälle

…. am Beispiel des Source-to-target-Mappings

a) Verschiebung (Translation):

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_q \\ y_q \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} x_q \\ y_q \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

b) Rotation:

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} x_q \\ y_q \end{bmatrix}$$

Achtung: Rotation <u>um den Koordinatenursprung</u>, <u>nicht</u> um die <u>Bildmitte</u> !

ÜBUNG: Affine Transformation 1

Geben Sie die Transformationsparameter so an, dass die Quellbildpunkte wie folgt auf die Zielbildpunkte abgebildet werden (Anm.: Bildgröße auf 1 normiert):

$$(x_{q1}, y_{q1}) = (0,0) \quad \Rightarrow \quad (x_{z1}, y_{z1}) = (0,0)$$
$$(x_{q2}, y_{q2}) = (1,0) \quad \Rightarrow \quad (x_{z2}, y_{z2}) = (0.4\ ,\ 0)$$
$$(x_{q3}, y_{q3}) = (1,1) \quad \Rightarrow \quad (x_{z3}, y_{z3}) = (0.9\ ,\ 0.4)$$

a) für das Source-to-target-Mapping

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \cdot \begin{bmatrix} x_q \\ y_q \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

b) für das Target-to-source-Mapping

$$\begin{bmatrix} x_q \\ y_q \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix} \cdot \begin{bmatrix} x_z \\ y_z \end{bmatrix} + \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}$$

ÜBUNG: Affine Transformation 2

Gegeben seien die Parameter des Source-to-target-Mappings (a_0, a_1, … b_2):

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0.5 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_q \\ y_q \end{bmatrix} + \begin{bmatrix} 5 \\ 10 \end{bmatrix}$$

Bestimmen Sie daraus die Parameter des Target-to-source-Mappings.

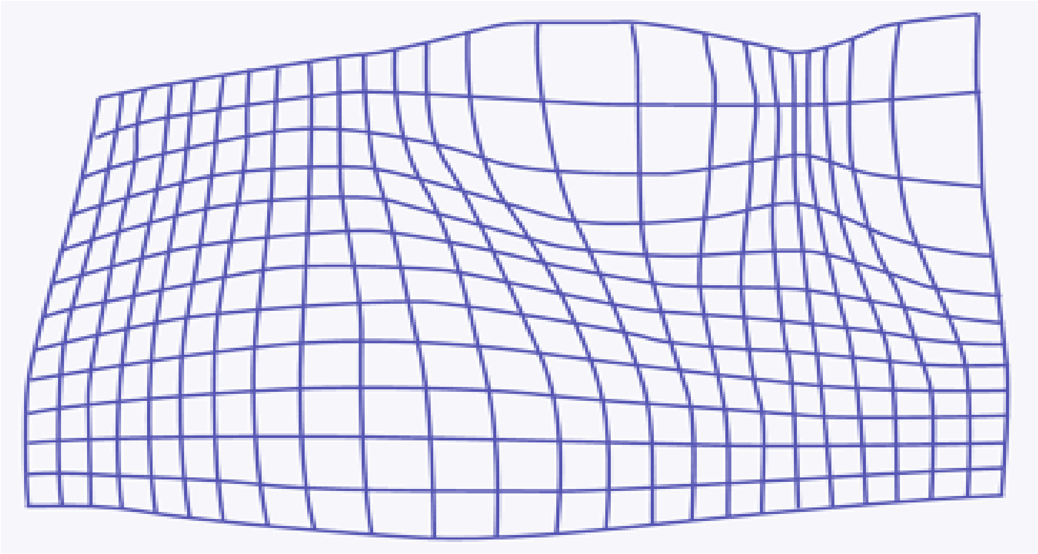
3.7.3 Transformation mit Polynomen

Die *polynomale Transformation* ist eine Erweiterung der affinen Transformation und erlaubt weitergehende geometrische Bildtransformationen.

Basis der polynomalen Transformation sind Polynome der Form:

$$x_z = a_0 + a_1 x_q + a_2 y_q + a_3 x_q y_q + a_4 x_q^2 + a_5 y_q^2 \ldots.$$
$$y_z = b_0 + b_1 x_q + b_2 y_q + b_3 x_q y_q + b_4 x_q^2 + b_5 y_q^2 \ldots.$$

hier am Beispiel des Source-to-target-Mappings

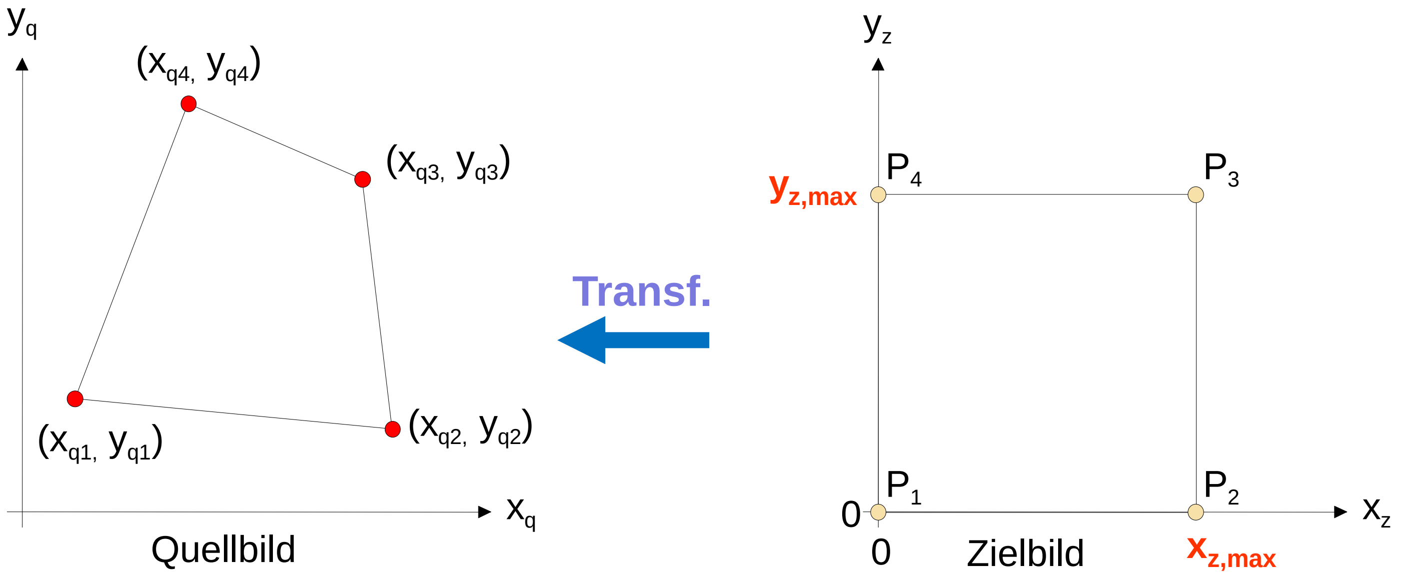


3.7.4 Vier-Punkt-Transformation

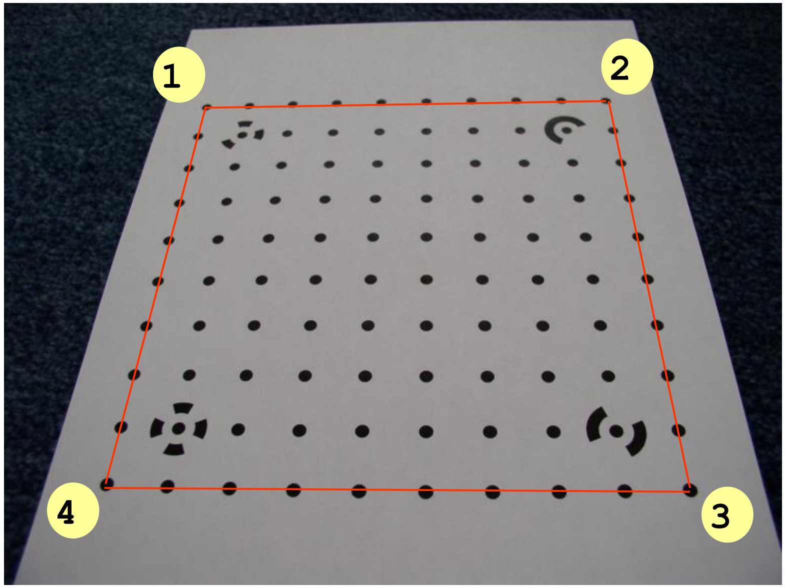
Die *Vier-Punkt-Transformation* (VPT) entzerrt eine beliebige 4-eckige Bildfläche auf eine rechteckige Bildfläche. Die VPT ist eine Target-to-Source-Transformation.

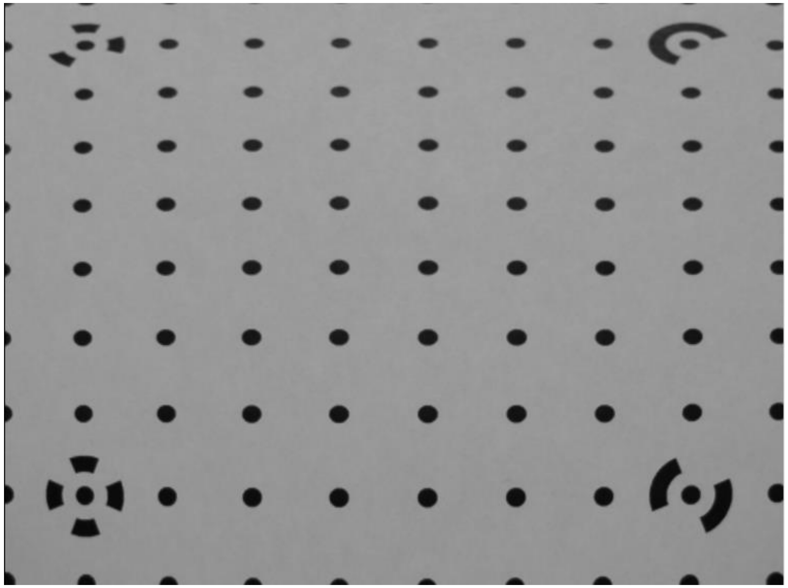
Parameter der Transformation:

- Eckpunkte des Quellbildausschnittes
- Zielbildgröße



Beispiel: Vier-Punkt-Transformation





Quellbild mit den gewünschten Eckpunkten 1..4

Zielbild

Zu einer Zielkoordinate wird die korrespondierende Quellkoordinate wie folgt bestimmt (Target–to-Source):

1. Zielkoordinate normieren

$$\hat{x}_z = x_z / x_{z,\max} \quad \text{und} \quad \hat{y}_z = y_z / y_{z,\max}$$

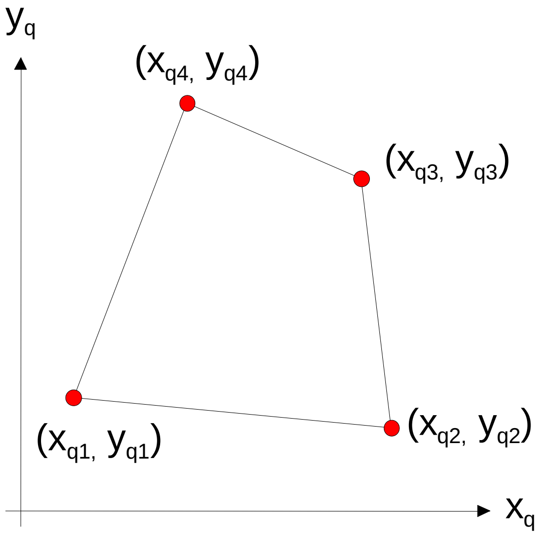
2. Hilfsgrößen $\Phi_1 \ldots \Phi_4$ ausrechnen

$$\begin{aligned}
\Phi_1(\hat{x}_z, \hat{y}_z) &= (1-\hat{x}_z)\cdot(1-\hat{y}_z) \\
\Phi_2(\hat{x}_z, \hat{y}_z) &= \hat{x}_z \cdot (1-\hat{y}_z) \\
\Phi_3(\hat{x}_z, \hat{y}_z) &= \hat{x}_z \cdot \hat{y}_z \\
\Phi_4(\hat{x}_z, \hat{y}_z) &= (1-\hat{x}_z)\cdot \hat{y}_z
\end{aligned}$$

3. Quellkoordinate bestimmen
(von denen der Grauwert geholt wird)

$$x_q = \sum_{i=1}^{4} \Phi_i(\hat{x}_z, \hat{y}_z)\cdot x_{q_i} \quad \text{und}$$

$$y_q = \sum_{i=1}^{4} \Phi_i(\hat{x}_z, \hat{y}_z)\cdot y_{q_i}$$



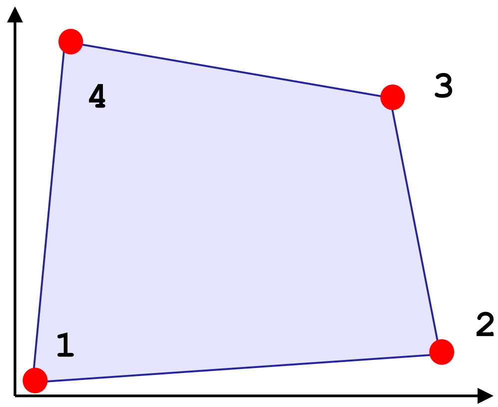
Lit.: Nonlinear Shape Restoration by Transformation Models, Tang/Suen, Int. Conf. on Pattern Recognition, 1990

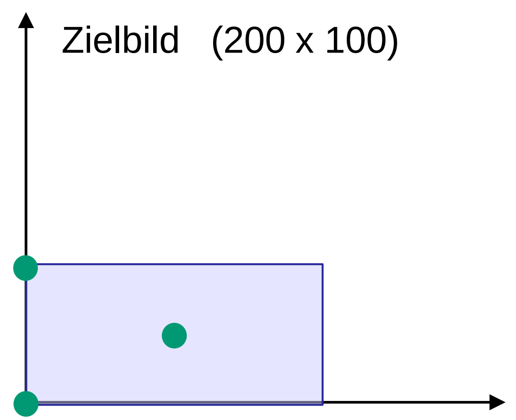
ÜBUNG: Vier-Punkte-Transformation

<u>Quellbildgröße:</u> 600x500 → <u>Zielbildgröße:</u> 200x100

Die folgenden Punkte markieren den Quellbildausschnitt, der auf das Zielbild abgebildet werden soll : ● (10,20), (300,40), (250,220) und (30, 280)

Quellbildausschnitt





a) Berechnen Sie die Parameter $\Phi_1 \ldots \Phi_4$ für die Zielbild-Koordinaten ●

a1) (0, 0)

a2) (0, 99)

a3) (100, 50)

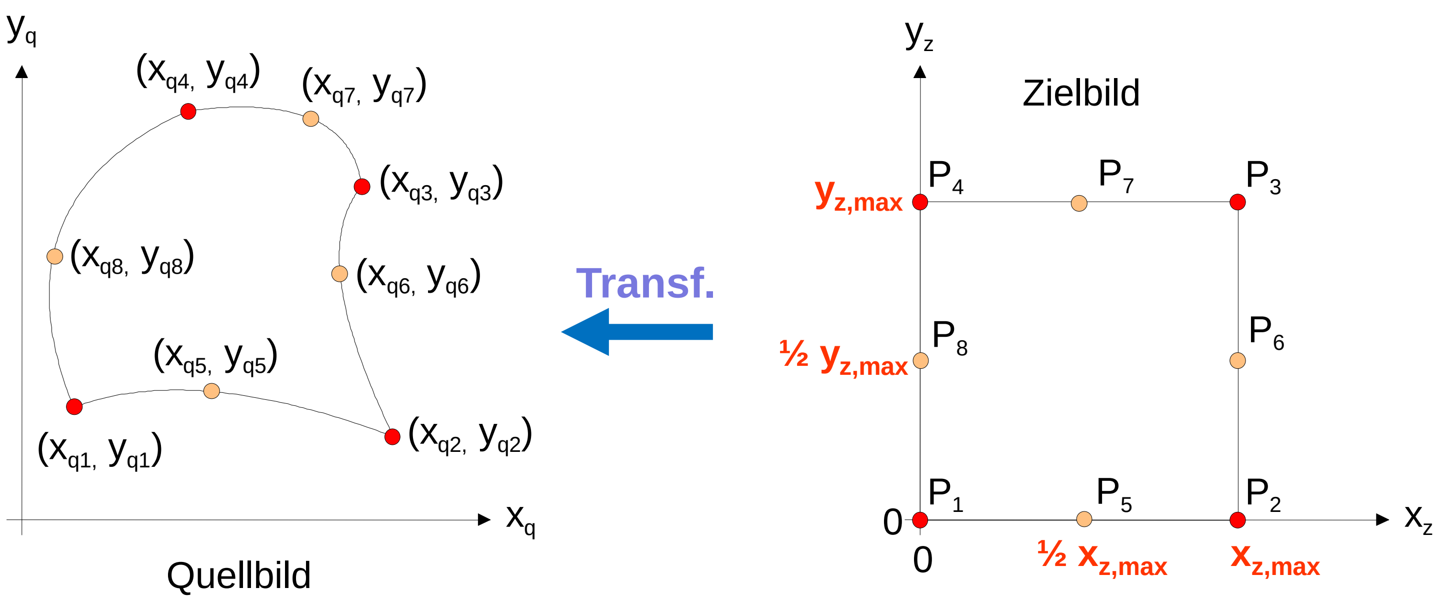
b) Aus welcher Quellbildpunkt-Koordinate wird der Grauwert des Zielbildpunktes (100, 50) genommen?

3.7.5 Acht-Punkt-Transformation (biquadratische Transformation)

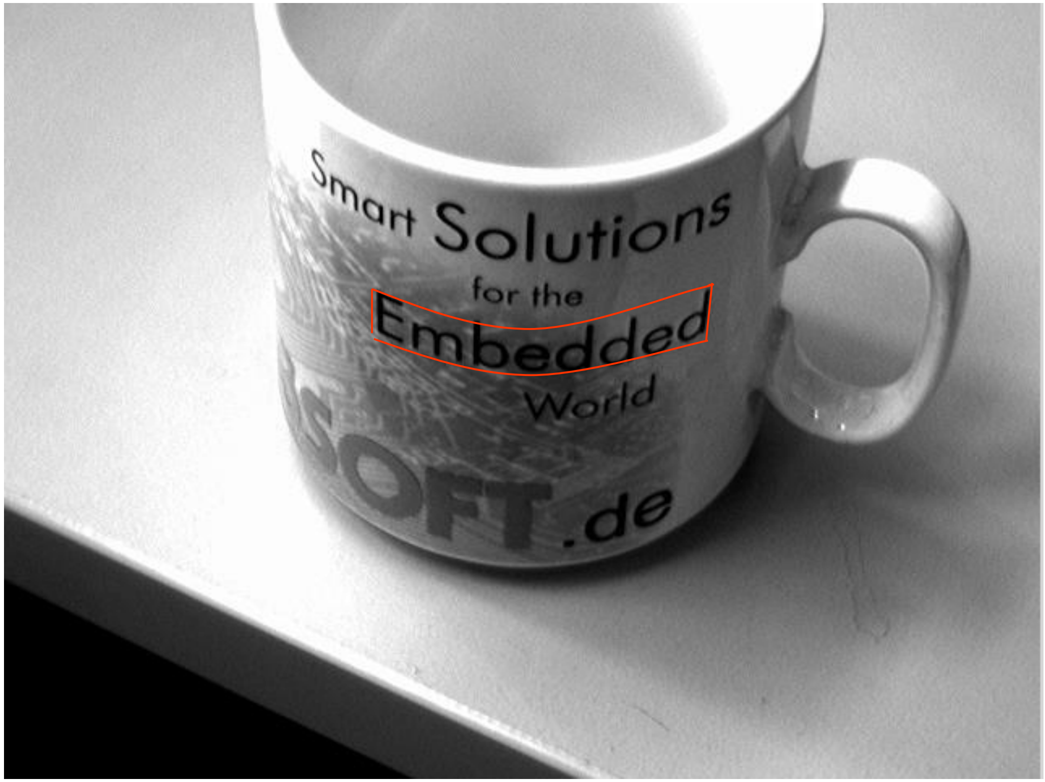
Die *Acht-Punkt-Transformation* entzerrt eine beliebige 8-eckige Bildfläche auf eine rechteckige Bildfläche.

Parameter der Transformation:

- 4 Eckpunkte und 4 Zwischenpunkte des Quellbildausschnittes
- Zielbildgröße



Beispiel: Acht-Punkt-Transformation



Zu einer Zielkoordinate wird die korrespondierende Quellkoordinate wie folgt bestimmt:

1. Zielkoordinate normieren

$$\hat{x}_z = x_z / x_{z,\max} \quad \text{und} \quad \hat{y}_z = y_z / y_{z,\max}$$

2. Hilfsgrößen $\Phi_1 \ldots \Phi_8$ ausrechnen

$$\begin{aligned}
\Phi_1(\hat{x}_z,\hat{y}_z) &= (1-\hat{x}_z)\cdot(1-\hat{y}_z)\cdot(1-2\hat{x}_z-2\hat{y}_z)\\
\Phi_2(\hat{x}_z,\hat{y}_z) &= \hat{x}_z\cdot(1-\hat{y}_z)\cdot(2\hat{x}_z-2\hat{y}_z-1)\\
\Phi_3(\hat{x}_z,\hat{y}_z) &= \hat{x}_z\cdot\hat{y}_z\cdot(2\hat{x}_z+2\hat{y}_z-3)\\
\Phi_4(\hat{x}_z,\hat{y}_z) &= \hat{y}_z\cdot(1-\hat{x}_z)\cdot(2\hat{y}_z-2\hat{x}_z-1)\\
\Phi_5(\hat{x}_z,\hat{y}_z) &= 4\hat{x}_z\cdot(1-\hat{x}_z)\cdot(1-\hat{y}_z)\\
\Phi_6(\hat{x}_z,\hat{y}_z) &= 4\hat{x}_z\cdot\hat{y}_z\cdot(1-\hat{y}_z)\\
\Phi_7(\hat{x}_z,\hat{y}_z) &= 4\hat{x}_z\cdot\hat{y}_z\cdot(1-\hat{x}_z)\\
\Phi_8(\hat{x}_z,\hat{y}_z) &= 4\hat{y}_z\cdot(1-\hat{x}_z)\cdot(1-\hat{y}_z)
\end{aligned}$$

3. Quellkoordinate bestimmen

$$x_q = \sum_{i=1}^{8} \Phi_i(\hat{x}_z,\hat{y}_z)\cdot x_{q_i} \quad \text{und}$$

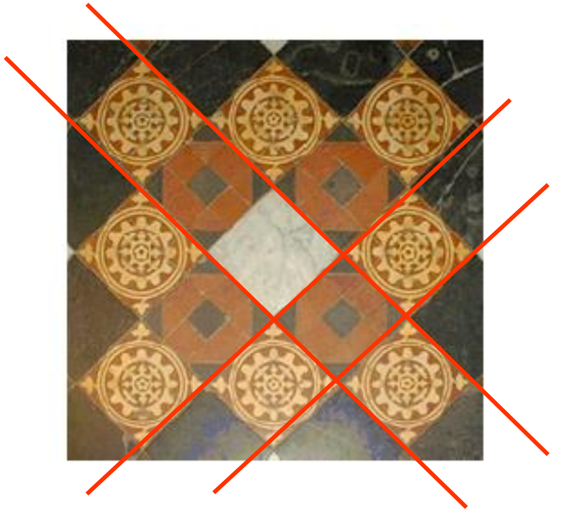
$$y_q = \sum_{i=1}^{8} \Phi_i(\hat{x}_z,\hat{y}_z)\cdot y_{q_i}$$

Lit.: Nonlinear Shape Restoration by Transformation Models, Tang/Suen, Int. Conf. on Pattern Recognition, 1990

3.7.6 Perspektivische Transformation

3.7.6.1 Eigenschaften

Die perspektivische Transformation ermöglicht die Berechnung der Senkrechtansicht einer ebenen Fläche aus einer Schrägansicht.

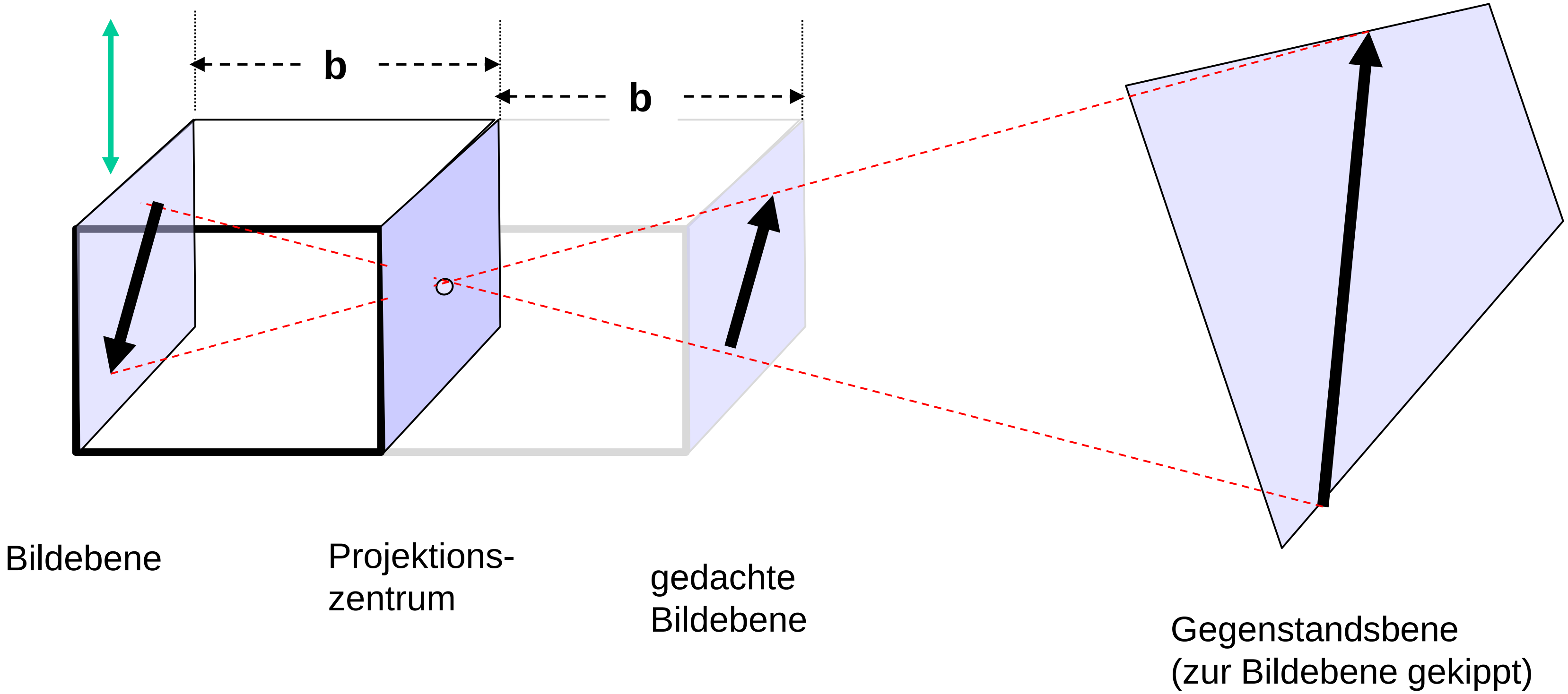


Die perspektivische Transformation ist geradenerhaltend, im Gegensatz zur affinen Transformation aber nicht parallelenerhaltend.

Lochkameramodell

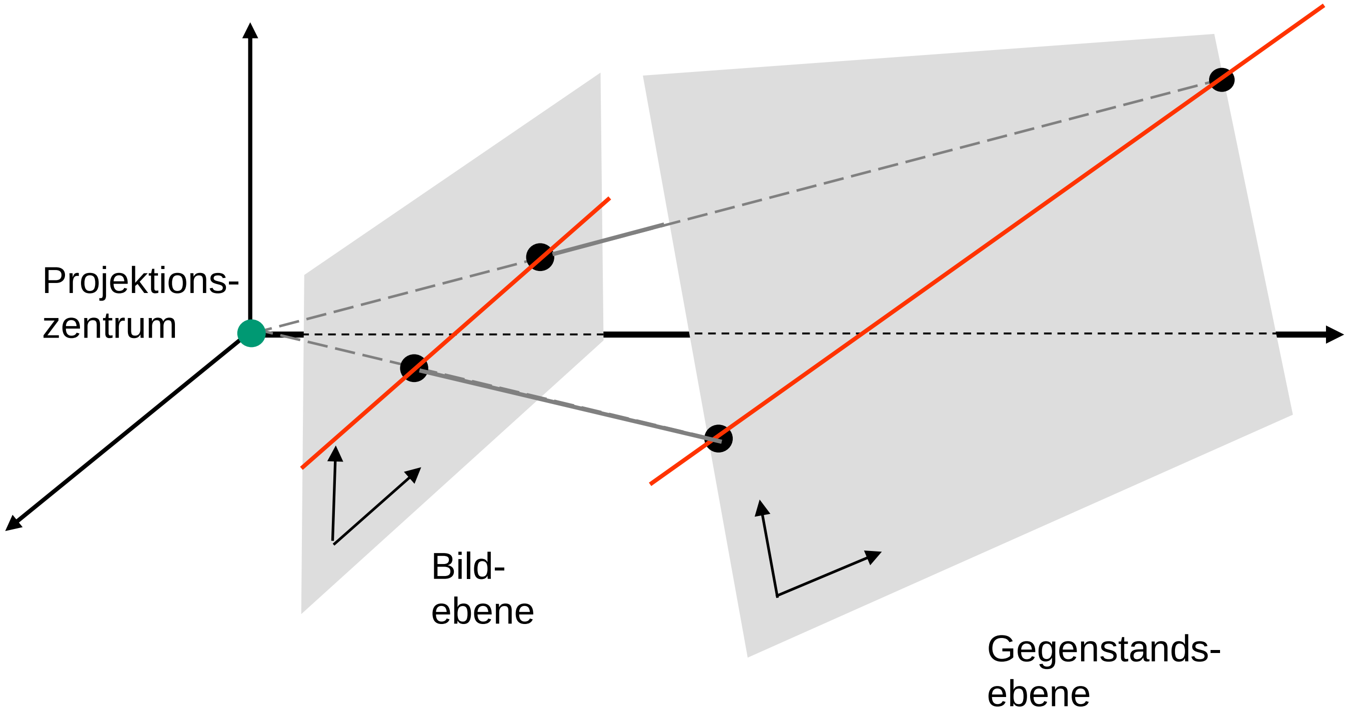
Das Bild wird spiegelverkehrt und auf dem Kopf stehend abgebildet.

b: Bildweite



Aus diesem Grund wird das Bild der „*gedachten Bildebene*“ verwendet, da dort das Bild aufrecht, seitenrichtig und im gleichen Maßstab wie auf der Bildebene abgebildet wird.

Geometrische Situation der perspektivischen Transformation



Die perspektivische Transformation ist eine Ebene-zu-Ebene-Transformation im 3-dimensionalen Raum.

3.7.6.2 Perspektivische Transformation für Target-to-Source-Mapping

Die perspektivische Transformation wird beschrieben durch:

$$x_q = \frac{b_{11}x_z + b_{12}y_z + b_{13}}{b_{31}x_z + b_{32}y_z + 1} \qquad y_q = \frac{b_{21}x_z + b_{22}y_z + b_{23}}{b_{31}x_z + b_{32}y_z + 1}$$

→ 8 Parameter beschreiben die Transformation.

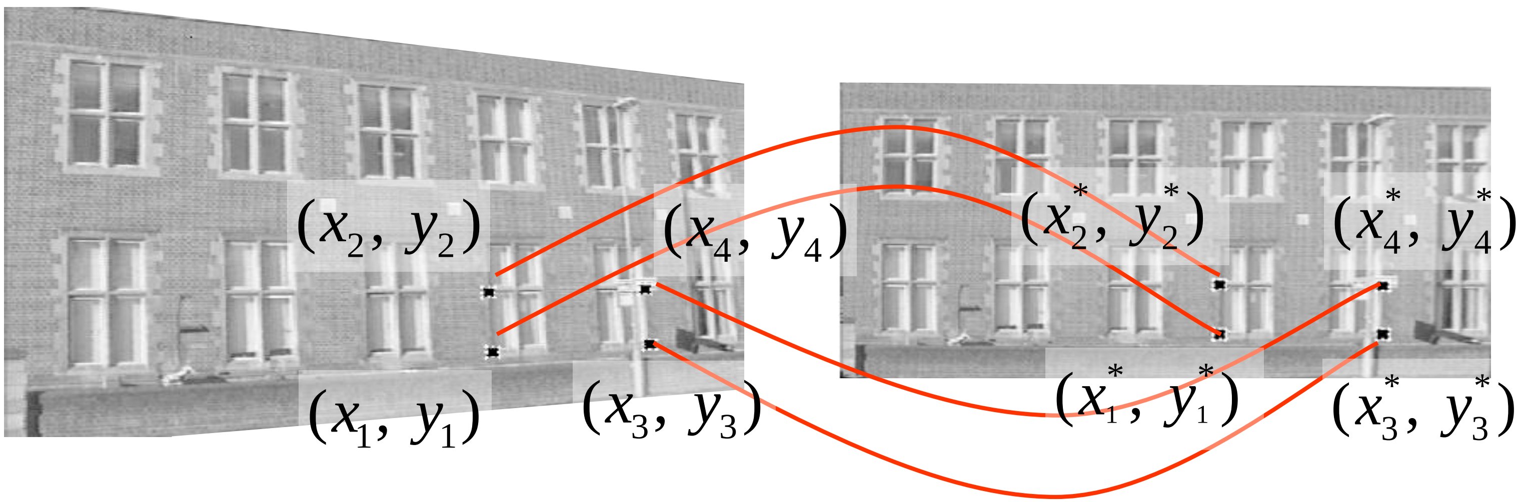
3.7.6.3 Perspektivische Transformation für Source-to-Target-Mapping

$$x_z = \frac{b^*_{11}x_q + b^*_{12}y_q + b^*_{13}}{b^*_{31}x_q + b^*_{32}y_q + 1} \qquad y_z = \frac{b^*_{21}x_q + b^*_{22}y_q + b^*_{23}}{b^*_{31}x_q + b^*_{32}y_q + 1}$$

3.7.6.4 Bestimmung der Transformationsparameter (b-Parameter)

Für die Bestimmung der 8 unbekannten b-Parameter werden mindestens 4 korrespondierende (nicht kollineare) Punkte im Quell- und Zielbild benötigt.

Anm.: 4 Punkte / 2 Gleichungen → 8 Unbekannte



Herleitung der Gleichungssystems zur Bestimmung der b-Parameter

Trennen der Unbekannten $b_{11} \ldots b_{32}$ von den bekannten Größen:

$$x_q = \frac{b_{11}x_z + b_{12}y_z + b_{13}}{b_{31}x_z + b_{32}y_z + 1} \quad \textbf{(1)} \qquad y_q = \frac{b_{21}x_z + b_{22}y_z + b_{23}}{b_{31}x_z + b_{32}y_z + 1} \quad \textbf{(2)}$$

$$x_q \cdot (b_{31}x_z + b_{32}y_z + 1) = b_{11}x_z + b_{12}y_z + b_{13} \quad \textbf{(1`)}$$

$$y_q \cdot (b_{31}x_z + b_{32}y_z + 1) = b_{21}x_z + b_{22}y_z + b_{23} \quad \textbf{(2`)}$$

ausmultiplizieren und alle b-Terme nach rechts

$$x_q = b_{11}x_z + b_{12}y_z + b_{13} - b_{31}x_z x_q - b_{32}y_z x_q \quad \textbf{(1``)}$$

$$y_q = b_{21}x_z + b_{22}y_z + b_{23} - b_{31}x_z y_q - b_{32}y_z y_q \quad \textbf{(2``)}$$

pro Punkt (x_k, y_k) eine Gleichung aufstellen (k = 1...4 oder mehr)

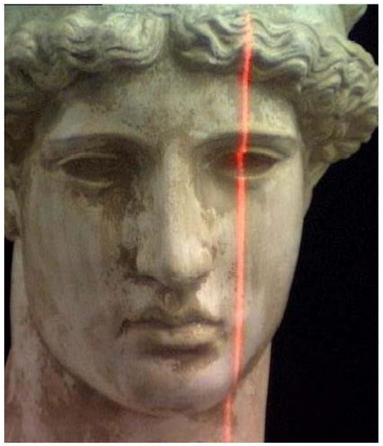
$$x_{q,k} = b_{11}x_{z,k} + b_{12}y_{z,k} + b_{13} - b_{31}x_{z,k}x_{q,k} - b_{32}y_{z,k}x_{q,k} \qquad (1''')$$

$$y_{q,k} = b_{21}x_{z,k} + b_{22}y_{z,k} + b_{23} - b_{31}x_{z,k}y_{q,k} - b_{32}y_{z,k}y_{q,k} \qquad (2''')$$

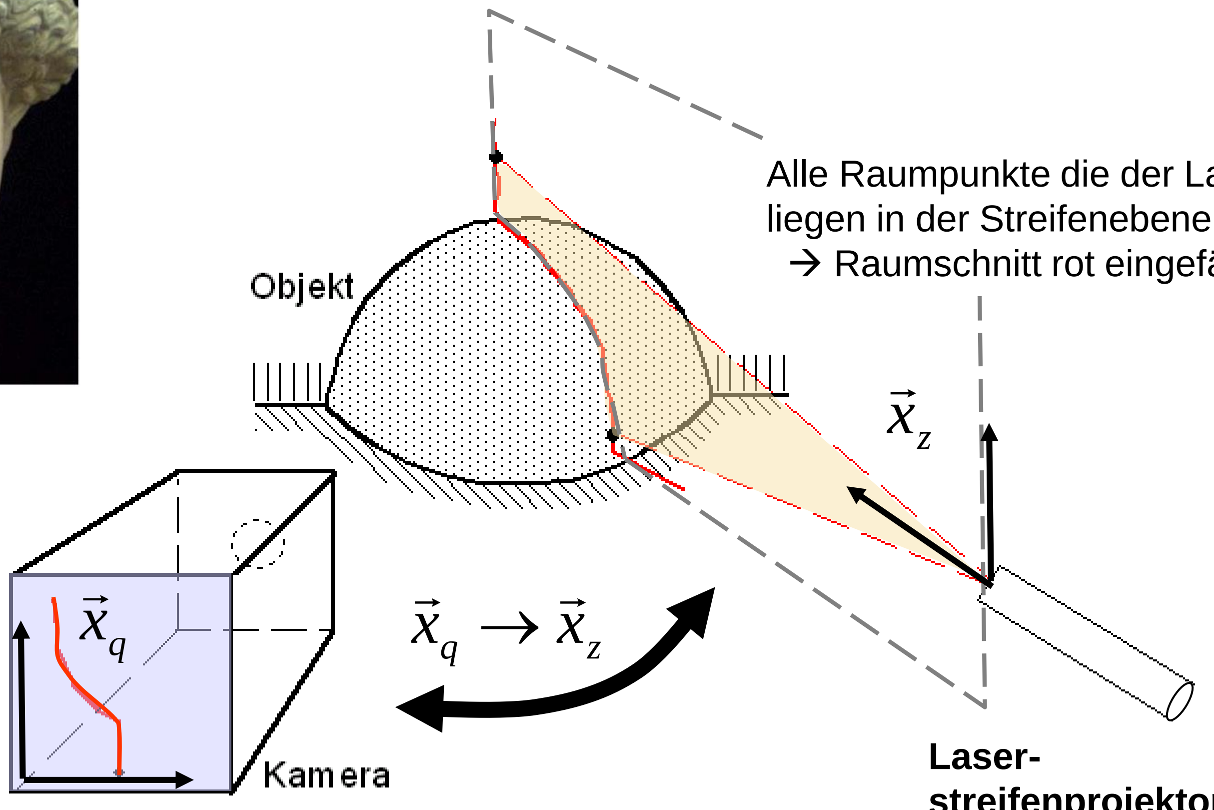
oder in Matrixform

$$\begin{bmatrix} x_{z,1} & y_{z,1} & 1 & 0 & 0 & 0 & -x_{z,1}x_{q,1} & -y_{z,1}x_{q,1} \\ 0 & 0 & 0 & x_{z,1} & y_{z,1} & 1 & -y_{z,1}x_{q,1} & -y_{z,1}y_{q,1} \\ \ldots\ldots & & & & & & & \\ x_{z,4} & y_{z,4} & 1 & 0 & 0 & 0 & -x_{z,4}x_{q,4} & -y_{z,4}x_{q,4} \\ 0 & 0 & 0 & x_{z,4} & y_{z,4} & 1 & -y_{z,4}x_{q,4} & -y_{z,4}y_{q,4} \end{bmatrix} \cdot \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \end{pmatrix} = \begin{pmatrix} x_{q,1} \\ y_{q,1} \\ \ldots. \\ x_{q,4} \\ y_{q,4} \end{pmatrix}$$

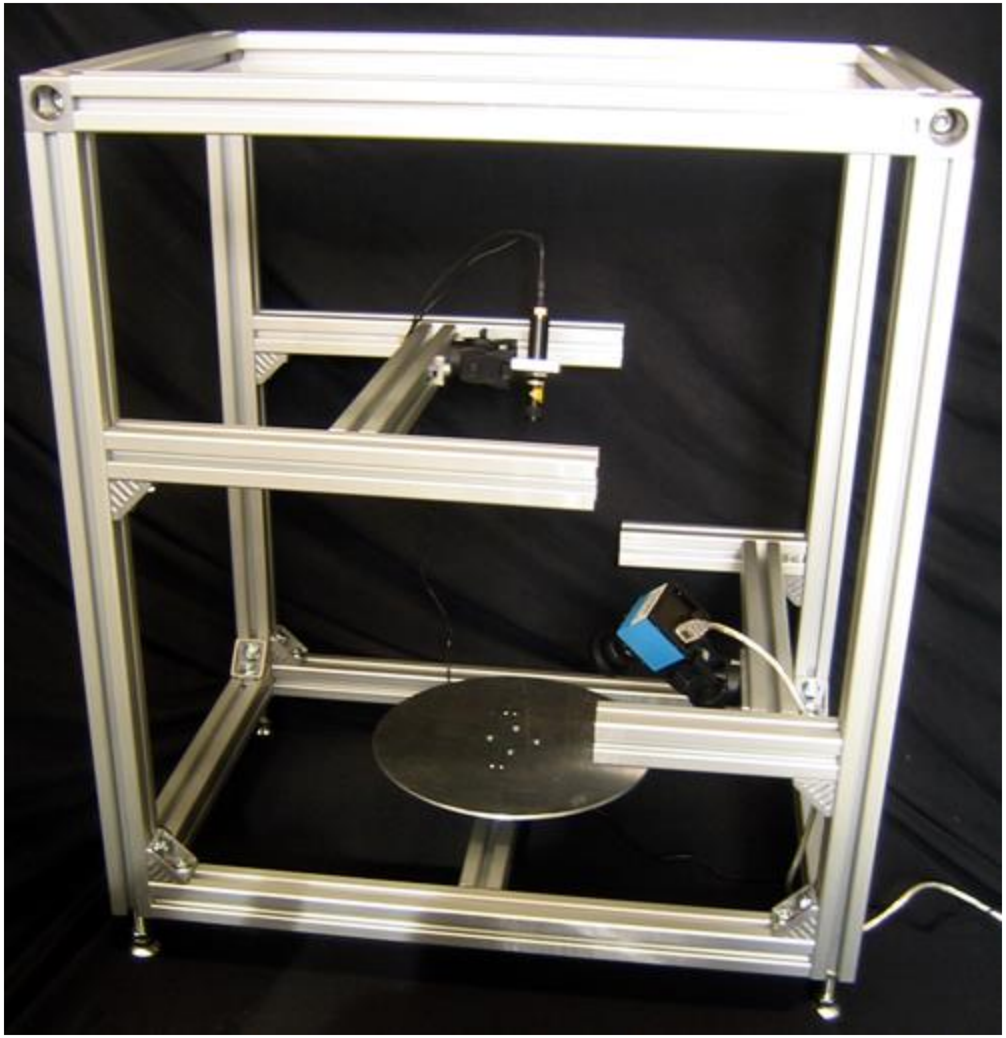
3.7.6.5 Beispielanwendung 1: 3D-Laserscanner



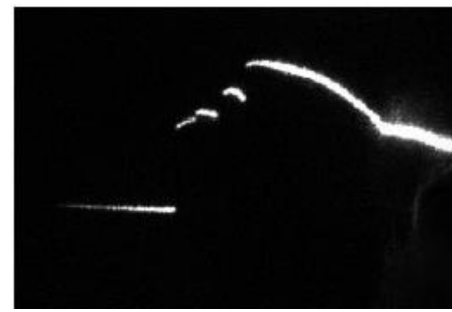
Alle Raumpunkte die der Laser trifft liegen in der Streifenebene.
→ Raumschnitt rot eingefärbt

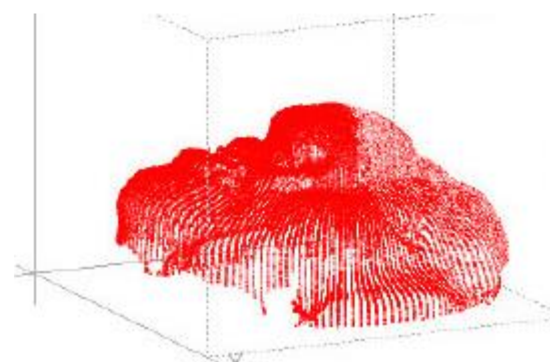


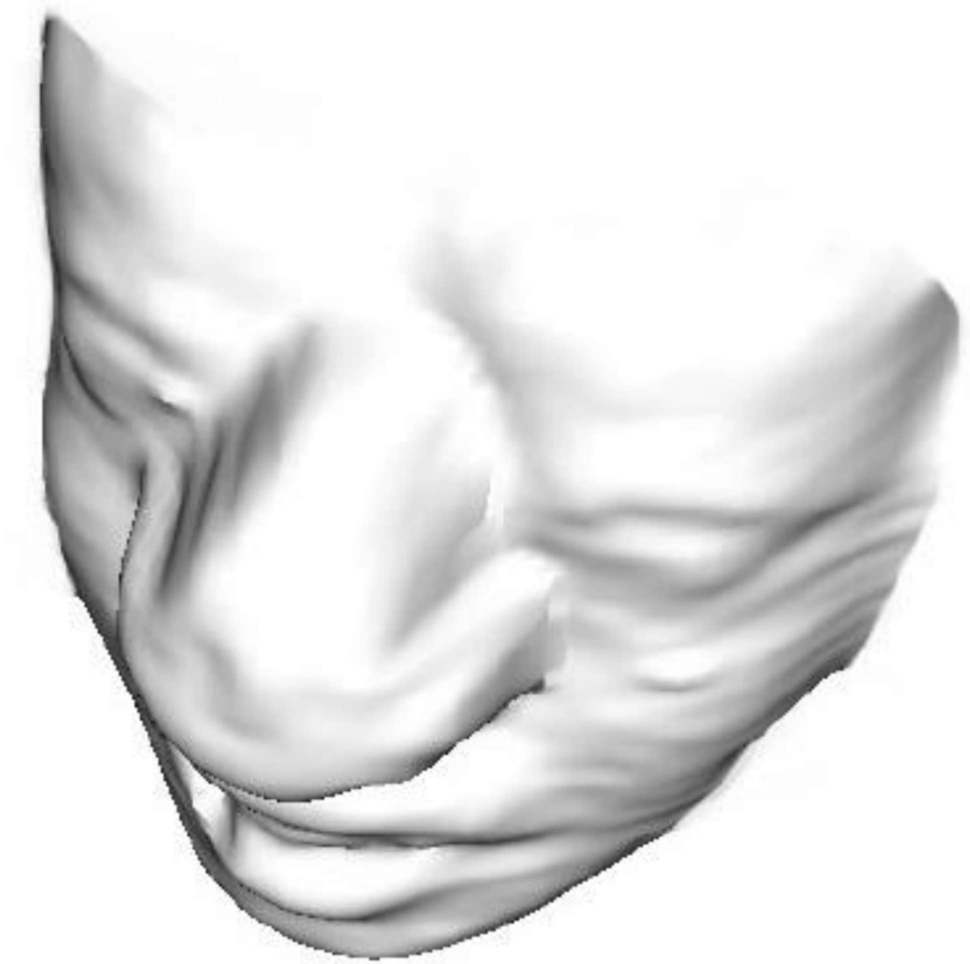
Alle abgebildeten Punkte liegen in der Bildebene. → blau eingefärbt



3D-Laserscanner, HAW, Robot Vision Lab

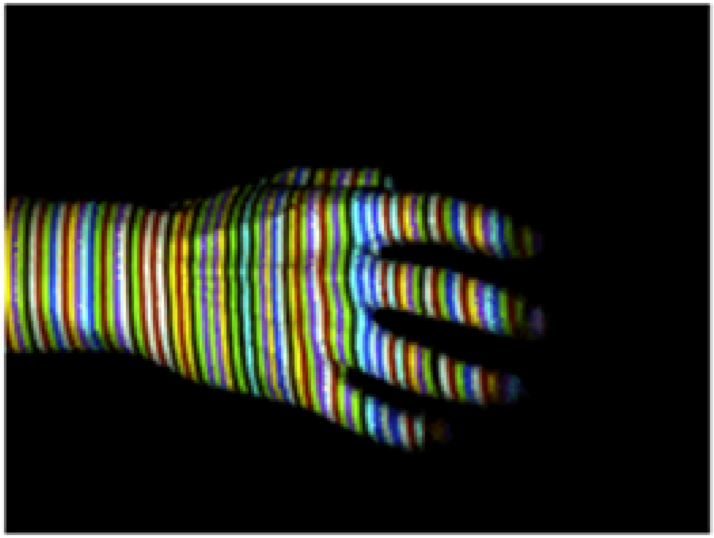




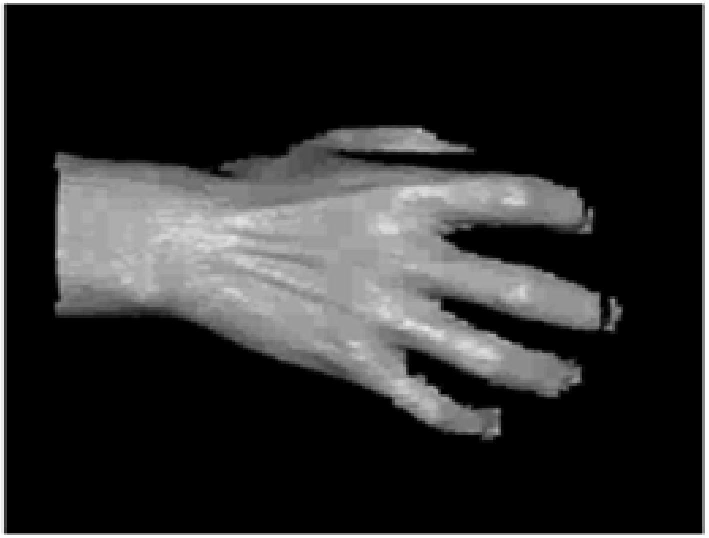


3.7.6.6 Beispielanwendung 1: Coded light Sensoren

Erweiterung des Konzeptes auf mehrere codierte Lichtschnitte zur gleichen Zeit
hier z.B. durch Farbcodierung ....



=>

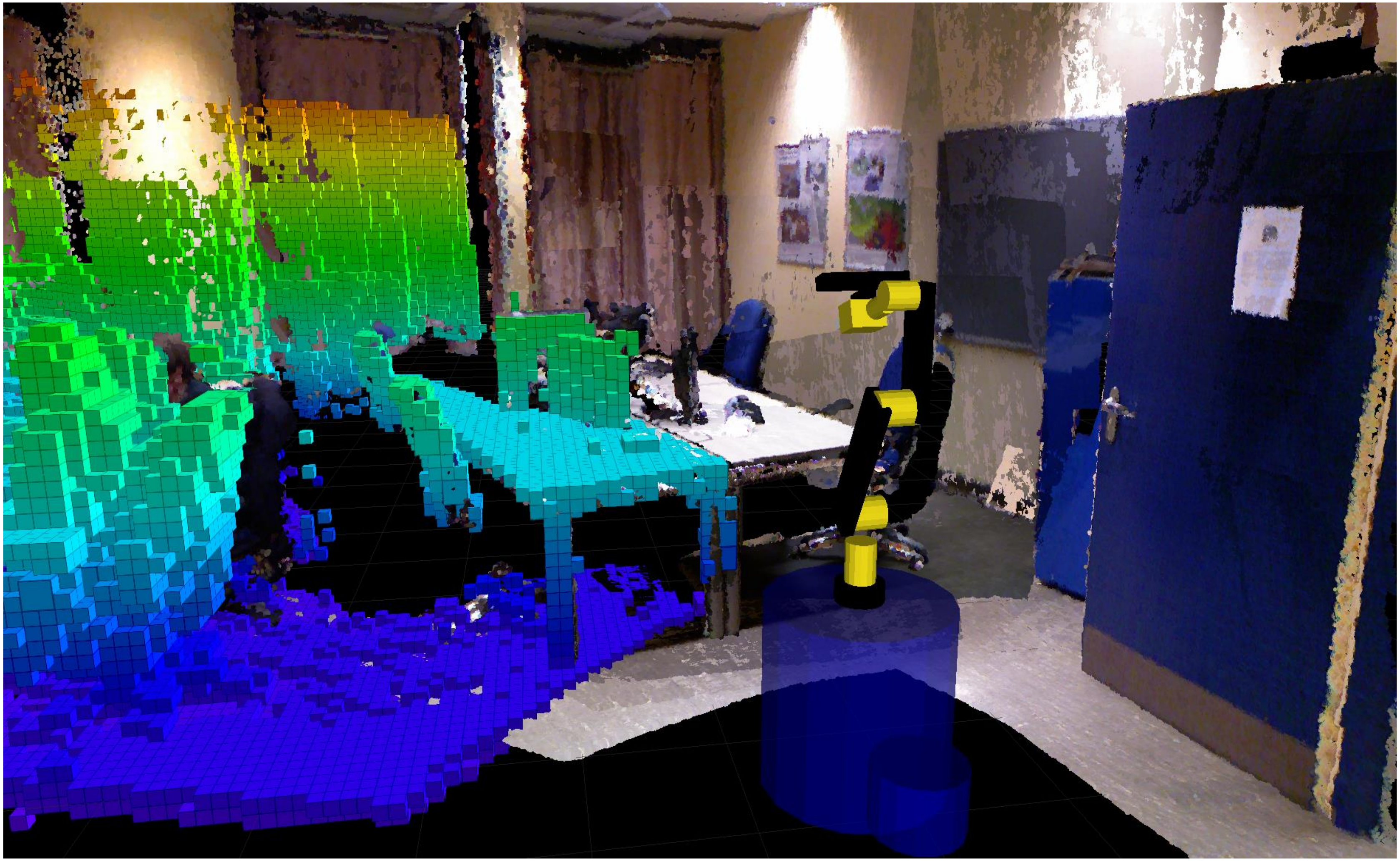


Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming
Li Zhang, Brian Curless, and Steven M. Seitz

.... oder ein codiertes Punktmuster beim Kinect 1

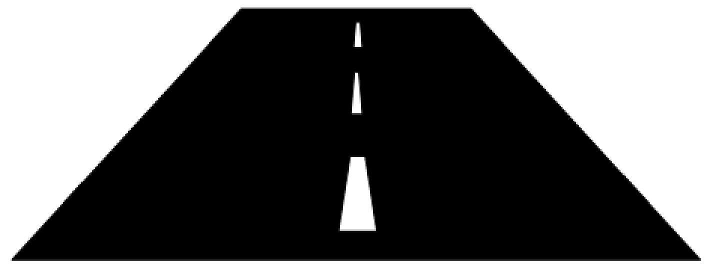
s. hackengineer.com

Robot-Vision-Labor: Mit Kinect erzeugte Octomap zur Roboter-Kollisionsvermeidung



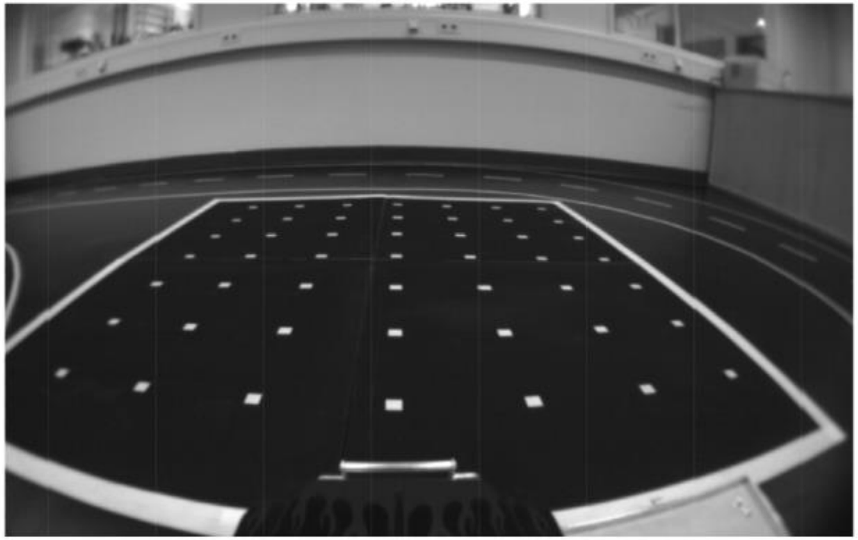
3.7.6.7 Beispielanwendung 2: Carolo-Cup-Fahrzeug

Ziel: Bild in die Straßenebene transformieren:



Vorteil: sehr viel einfachere Spurführung

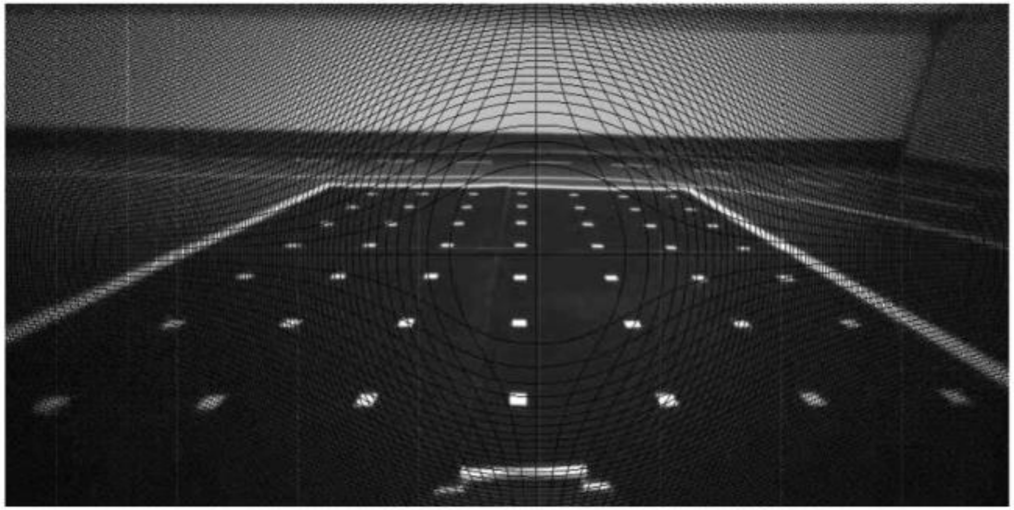
Zusätzliche Aufgaben : Korrektur der Linsenverzeichnung



Kalibrierplatte zur Bestimmung der Verzeichnungsparameter

a) Linsenverzeichnung

b) perspektivische Korrektur



Korrektur der Linsenverzeichnung (Source-to-target)