03-05-H
-709.53

Echtzeitbildverarbeitung (6)

Prof. Dr. Udo Frese

Faltungsoperationen

Linien- bzw. Kantendetektion

Kontrastnormalisierter Kantenfilter

Was bisher geschah

- **k-D Baum für m-Nearest Neighbour Farbsegmentierung**
  - Klassifiziere eine Farbe gemäß der absoluten Mehrheit der m nächsten Trainingsvektoren (oder weise sie zurück)
  - Bilde Binärbaum der entlang abwechselnder Dimensionen jeweils halbiert (Aufbau mit Sortieren, Median und Rekursion)
  - Suche rekursiv, steige zuerst in die nähere Hälfte ab, dann in die fernere, falls der m-t nächste bisher gefundene weiter entfernt ist als die Grenze
- **Tabelliere das Ergebnis der Klassifikation für alle Farben**
- **Vermeide Tabelle größer als Cache, dann lieber etwas rechnen**
- **Bildverarbeitung GermanTeam im Sony Four-legged RoboCup**
  - Tabellenbasierte Farbsegmentierung als Basis
  - Projektion auf den Boden durch bekannte Kamerapose
  - Lokalisation mit Partikelfiltern
  - Viele spezielle Tricks um Effizienz und Robustheit zu verbessern

Faltungsoperationen

Pixel mit seinen Nachbarn verknüpfen

- **Bisher: Pixel alleine (Schwellwert / Farbsegmentierung)**
- **Jetzt: Ein Pixel und seine Nachbarn**
  - ein Hilfsbild ausrechnen
  - Pixel im Hilfsbild hängt von Nachbarn im Originalbild ab
  - im Hilfsbild Pixel einzeln betrachten
- **Allgemeine Erfahrung:**
  Lineare Operationen sind
  - strukturiert
  - leicht zu verstehen
  - oft die Basis nichtlinearer Operationen

Faltungsoperationen

Lineare Abbildung von Bildern

- **Axiome:**
  - $f(p_1+p_2) = f(p_1)+f(p_2)$
  - $f(\lambda p)=\lambda f(p)$
- **Ergebnispixel $f(p)_{x',y'}$ ist gewichtete Summe der Eingangspixel $p_{x,y}$.**

$$f(p)_{x',y'} = \sum_{x,y} \alpha_{x,y,x',y'} p_{x,y}$$

- **zu allgemein, weil für 1024×768 Bild $(1024 \cdot 768)^2 = 6.2 \cdot 10^{11}$ Koeffizienten α**
- **Axiom: Translationsinvariant**
  - f(Trans(p)) = Trans(f(p))
- **Wirkt an jedem Pixel gleich.**
- **Koeffizient $\alpha_{x,y,x',y'}$ hängt nur von der Relativposition x-x‘, y-y‘ ab.**

$$\alpha_{x,y,x',y'} = \alpha_{x-x',y-y'}$$

$$f(p)_{x',y'} = \sum_{x,y} \alpha_{x-x',y-y'} p_{x,y}$$

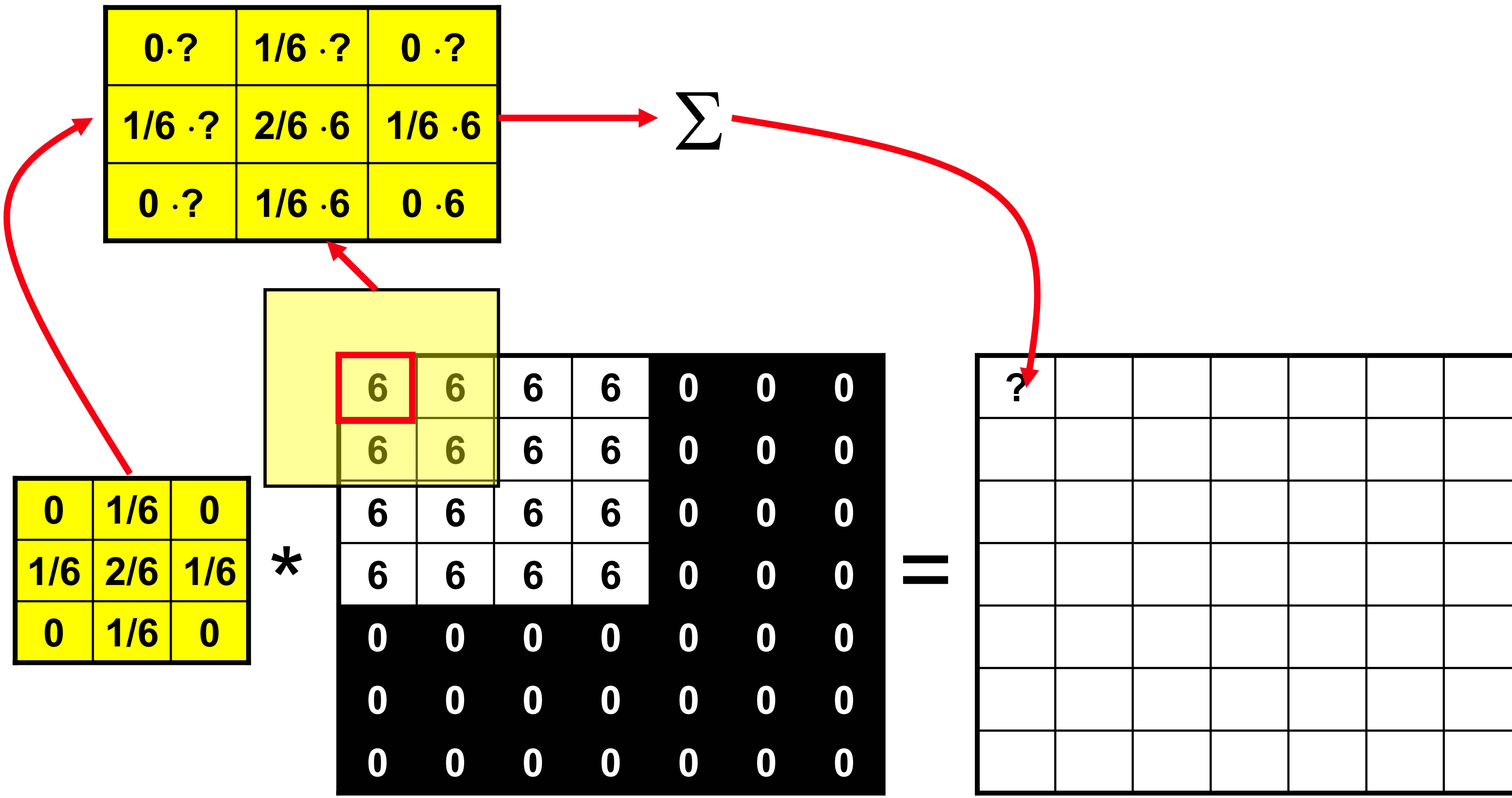
Faltungsoperationen

Lineare Abbildung von Bildern

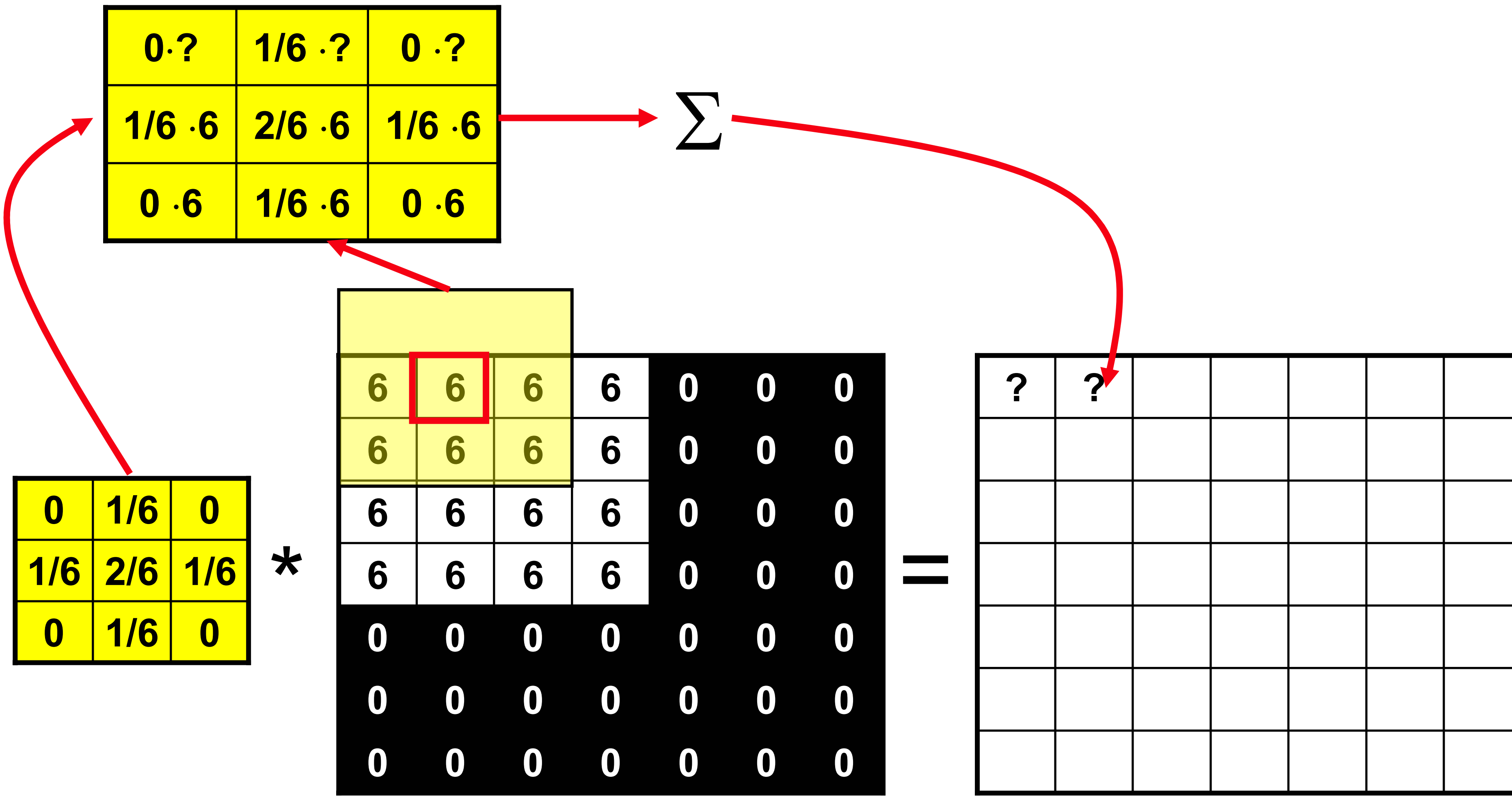
$$p' = \alpha * p$$

- **Faltung, Konvolution, Operator ***
- **α Filter, Filterkern oder Faltungsmaske**
- **Filter α ist wie ein Bild, * verknüpft also Bilder**
- **Oft ist α lokal**
  - 0 für große x-x‘, y-y‘
  - Ergebnispixel wir nur von kleiner Nachbarschaft des Eingangspixels beeinflusst

Faltungsoperationen



Faltungsoperationen



Faltungsoperationen

0·6	1/6 ·6	0 ·6
1/6 ·6	2/6 ·6	1/6 ·6
0 ·6	1/6 ·6	0 ·6

$\sum$

0	1/6	0
1/6	2/6	1/6
0	1/6	0

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	6					

Faltungsoperationen

0·6	1/6 ·6	0 ·6
1/6 ·6	2/6 ·6	1/6 ·6
0 ·6	1/6 ·6	0 ·6

$\sum$

0	1/6	0
1/6	2/6	1/6
0	1/6	0

*

6	6	6	6	0	0	0
6	6	$\boxed{6}$	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	6	6				

Faltungsoperationen

0·6	1/6 ·6	0 ·0
1/6 ·6	2/6 ·6	1/6 ·0
0 ·6	1/6 ·6	0 ·0

$\sum$

0	1/6	0
1/6	2/6	1/6
0	1/6	0

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	6	6	5			

Faltungsoperationen

0·6	1/6 ·0	0 ·0
1/6 ·6	2/6 ·0	1/6 ·0
0 ·6	1/6 ·0	0 ·0

$\sum$

0	1/6	0
1/6	2/6	1/6
0	1/6	0

$*$

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$=$

?	?	?	?	?	?	?
?	6	6	5	1		

Faltungsoperationen

$$\begin{bmatrix} 0 \cdot 6 & 1/6 \cdot 0 & 0 \cdot 0 \\ 1/6 \cdot 6 & 2/6 \cdot 0 & 1/6 \cdot 0 \\ 0 \cdot 6 & 1/6 \cdot 0 & 0 \cdot 0 \end{bmatrix} \rightarrow \sum$$

$$\begin{bmatrix} 0 & 1/6 & 0 \\ 1/6 & 2/6 & 1/6 \\ 0 & 1/6 & 0 \end{bmatrix} * \begin{bmatrix} 6 & 6 & 6 & 6 & 0 & 0 & 0 \\ 6 & 6 & 6 & 6 & 0 & \boxed{0} & 0 \\ 6 & 6 & 6 & 6 & 0 & 0 & 0 \\ 6 & 6 & 6 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? & ? & ? & ? \\ ? & 6 & 6 & 5 & 1 & 0 & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix}$$

Faltungsoperationen

- **Frage an das Auditorium: Wie würde man sprachlich die Wirkung des Filters beschreiben?**

0	1/6	0
1/6	2/6	1/6
0	1/6	0

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	6	6	5	1	0	?
?	6	6	5	1	0	?
?	5	5	4	1	0	?
?	1	1	1	0	0	?
?	0	0	0	0	0	?
?	?	?	?	?	?	?

Faltungsoperationen

- **Frage an das Auditorium: Wie würde man sprachlich die Wirkung des Filters beschreiben?**
- **Das Bild wird geglättet oder verwischt oder unscharf.**

0	1/6	0
1/6	2/6	1/6
0	1/6	0

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	6	6	5	1	0	?
?	6	6	5	1	0	?
?	5	5	4	1	0	?
?	1	1	1	0	0	?
?	0	0	0	0	0	?
?	?	?	?	?	?	?

Faltungsoperationen

0·6	0 ·6	0 ·0
0 ·6	-1/2 ·6	1/2 ·0
0 ·6	0 ·6	0 ·0

$\sum$

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

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	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	-3			

Faltungsoperationen

Verschiebung des Wertebereiches um negative Ergebnisse zu vermeiden.

0·6	0 ·6	0 ·0
0 ·6	-1/2 ·6	1/2 ·0
0 ·6	0 ·6	0 ·0

Σ

+3

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

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	3	3	0			

Faltungsoperationen

- **Frage an das Auditorium: Wie würde man sprachlich die Wirkung des Filters beschreiben?**

+3

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

*

6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	3	3	3	0	3	?
?	3	3	3	0	3	?
?	3	3	3	0	3	?
?	3	3	3	3	3	?
?	3	3	3	3	3	?
?	?	?	?	?	?	?

Faltungsoperationen

- **Frage an das Auditorium: Wie würde man sprachlich die Wirkung des Filters beschreiben?**
- **Vertikale Kanten werden erkannt.**

+3

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

*

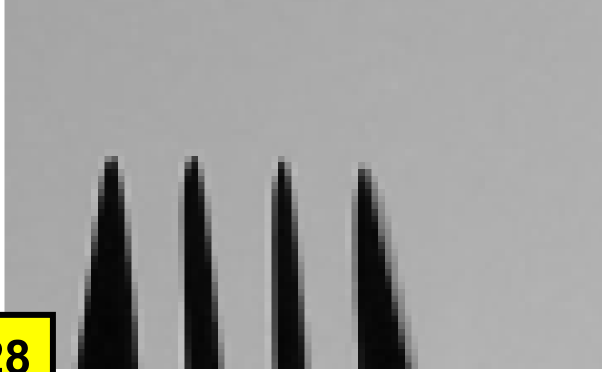
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
6	6	6	6	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

=

?	?	?	?	?	?	?
?	3	3	3	0	3	?
?	3	3	3	0	3	?
?	3	3	3	0	3	?
?	3	3	3	3	3	?
?	3	3	3	3	3	?
?	?	?	?	?	?	?

- **Frage an das Auditorium: Welches Bild gehört zu welchem Filter?**

0	0	1/28	0	0
0	2/28	3/28	2/28	0
1/28	3/28	4/28	3/28	1/28
0	2/28	3/28	2/28	0
0	0	1/28	0	0



+128

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

+128

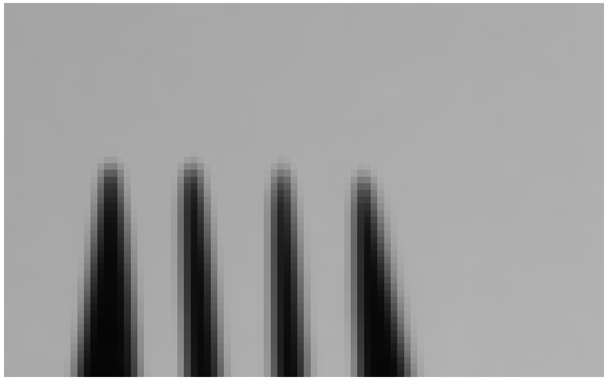
1	0	-1
1	0	-1
1	0	-1

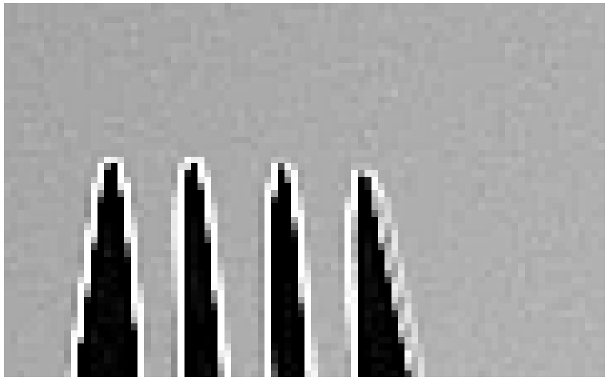
0	1/8	0
1/8	4/8	1/8
0	1/8	0

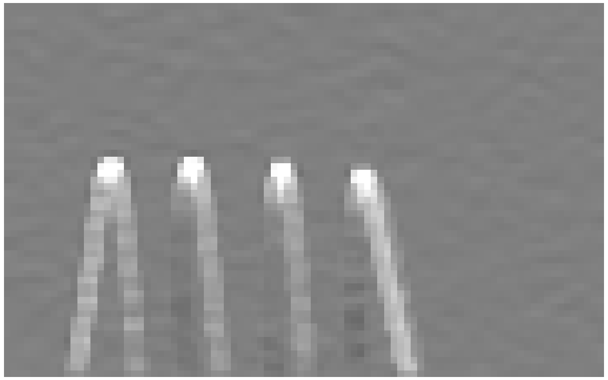
+128

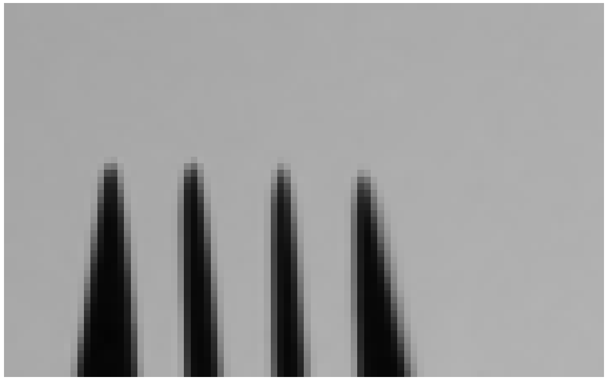
-1/2	-1/2	-1/2
-1/2	8/2	-1/2
-1/2	-1/2	-1/2

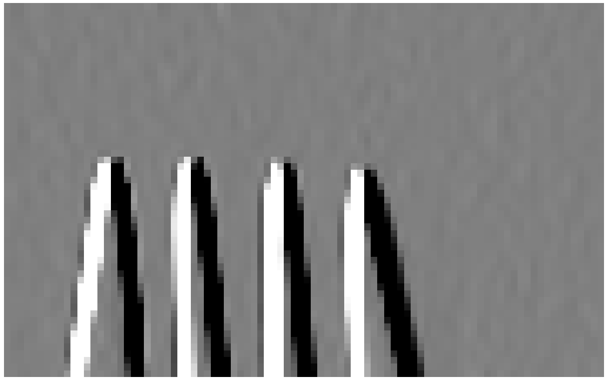
-1/2	-1/2	-1/2
-1/2	10/2	-1/2
-1/2	-1/2	-1/2

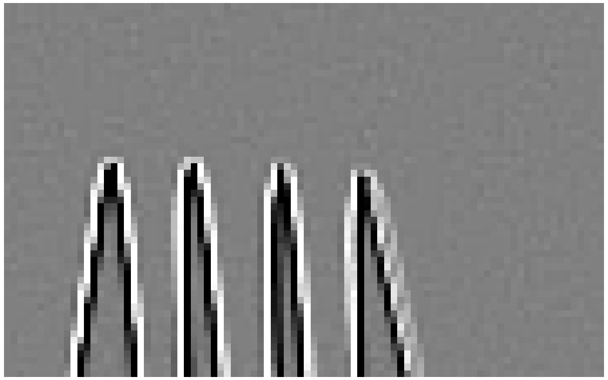






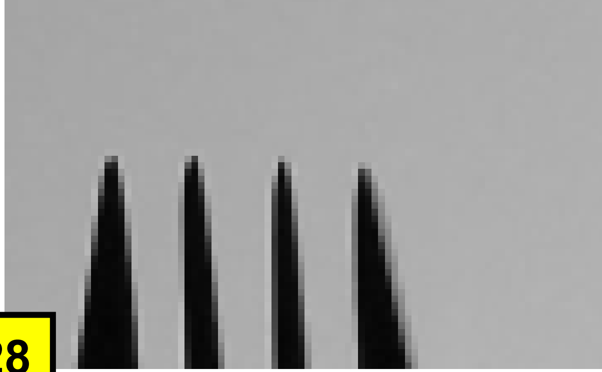






- **Frage an das Auditorium: Welches Bild gehört zu welchem Filter?**

0	0	1/28	0	0
0	2/28	3/28	2/28	0
1/28	3/28	4/28	3/28	1/28
0	2/28	3/28	2/28	0
0	0	1/28	0	0



+128

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

+128

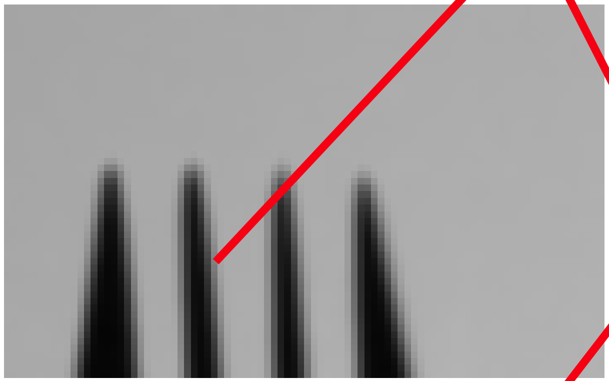
1	0	-1
1	0	-1
1	0	-1

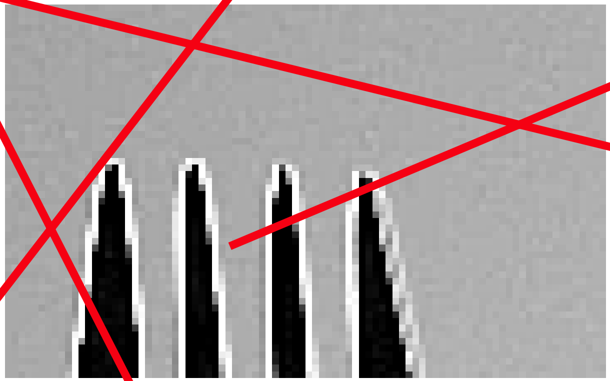
0	1/8	0
1/8	4/8	1/8
0	1/8	0

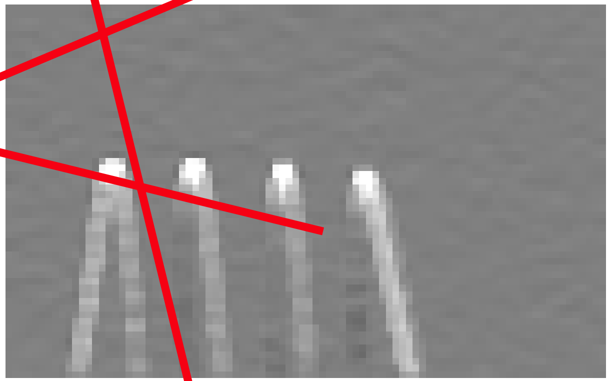
+128

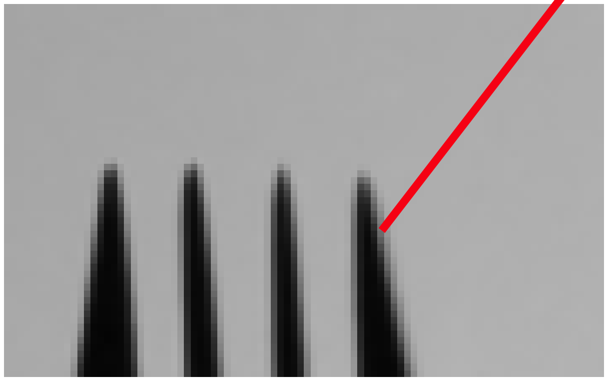
-1/2	-1/2	-1/2
-1/2	8/2	-1/2
-1/2	-1/2	-1/2

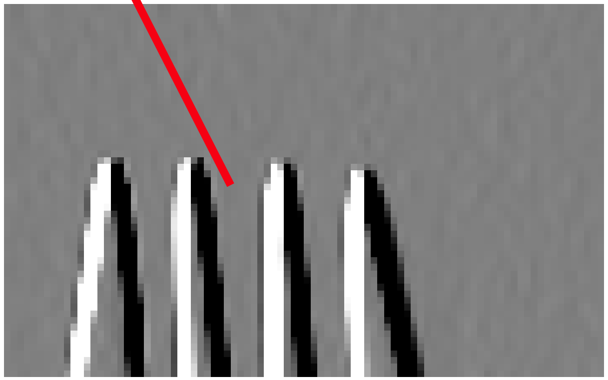
-1/2	-1/2	-1/2
-1/2	10/2	-1/2
-1/2	-1/2	-1/2

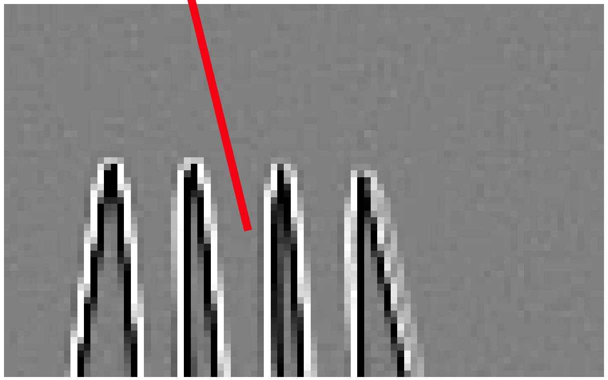






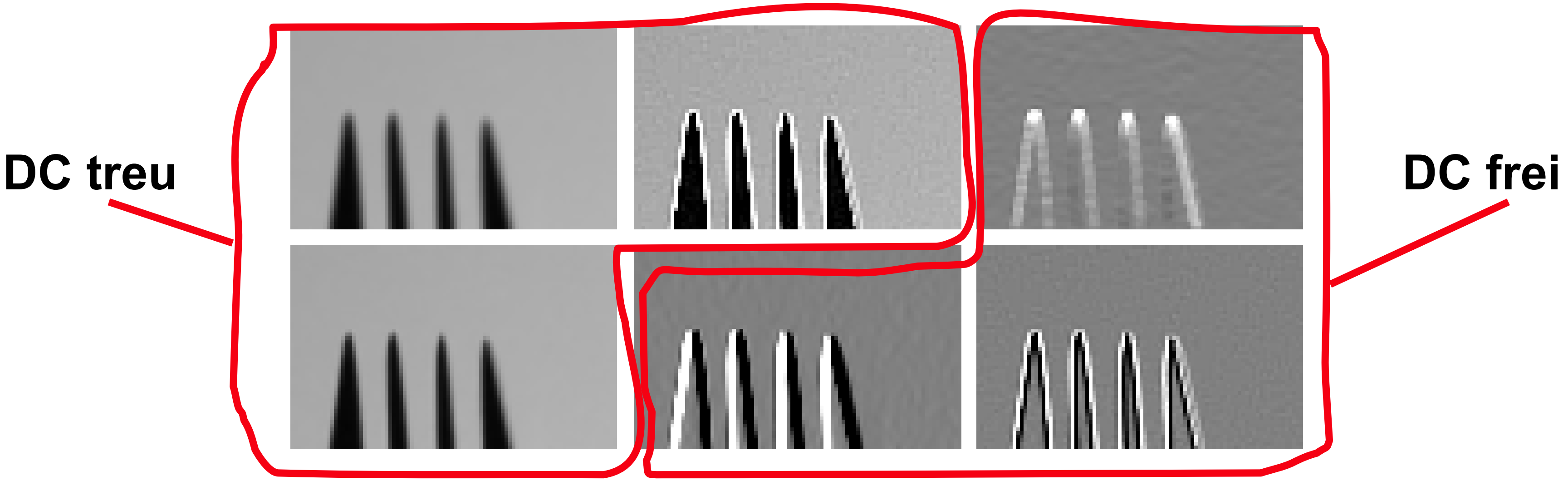






Faltungsoperationen

- Summe der Filterkoeffizienten ist Antwort auf konstantes Bild
- DC Anteil, von Directed Current = Gleichstrom
- DC Anteil 0: Helligkeit „im Großen“ spielt keine Rolle (DC frei).
- DC Anteil 1: Helligkeit „im Großen“ bleibt gleich (DC treu).
- DC Anteil$\neq$0: Filter skalierbar auf DC Anteil =1
- Technisch: In Darstellung +128 (Grau) addieren.



Faltungsoperationen

- **Direkte Implementierung einer Faltung**

```
convolve (Image* dstImg, Image* srcImg, doubleImage* filter)
{
   for (y=0; y<dstImg.height; y++)
      for (x=0; x<dstImg.width; x++) {
         double sum = 0;
         for (y2=0; y2<filter.height; y2++)
            for (x2=0; x2<filter.width; x2++) {
               xSrc = x + x2 - filter.width/2;
               ySrc = y + y2 - filter.height/2;
               if (0<=xSrc && xSrc<srcImg.width &&
                  0<=ySrc && ySrc<srcImg.height)
                  sum += filter(x2, y2) * srcImg(xSrc, ySrc);
         dstImg(x, y) = sum + filter->offset;
      }
}
```

Faltungsoperationen

- **Direkte Implementierung einer Faltung**
- **<u>Frage an das Auditorium</u>: Wie kann man die Routine beschleunigen? Wie für einen speziellen Filter?**

```
convolve (Image* dstImg, Image* srcImg, doubleImage* filter)
{
   for (y=0; y<dstImg.height; y++)
      for (x=0; x<dstImg.width; x++) {
         double sum = 0;
         for (y2=0; y2<filter.height; y2++)
            for (x2=0; x2<filter.width; x2++) {
               xSrc = x + x2 - filter.width/2;
               ySrc = y + y2 - filter.height/2;
               if (0<=xSrc && xSrc<srcImg.width &&
                  0<=ySrc && ySrc<srcImg.height)
                  sum += filter(x2, y2) * srcImg(xSrc, ySrc);
         dstImg(x, y) = sum + filter->offset;
      }
}
```

Faltungsoperationen

Beschleunigung der Faltung durch

- **Allgemeine Faltung**
  - Zeigern statt Koordinaten
  - Rand in äußeren 2 Schleifen berücksichtigen ⇒ innen kein Test
  - rechnen mit `int`, Summe durch ggT teilen
- **Spezieller Filter**
  - inneren 2 Schleifen durch Term ersetzen (!)
  - 0 Koeffizienten auslassen
  - gleiche Koeffizienten zusammenfassen
  - Multiplikation / Division mit Zweierpotenzen durch <<, >> (autom.)
  - konstanten Speicheroffset von 1 für x ausnutzen
- **Bei großen Filtern**
  - Separieren (erst x dann y filtern)
  - Fast Fourier Transformation

- **Frage an das Auditorium:**
  Was sind Merkmalen, die man in Büroumgebungen, hier z.B. das MZH wohl mit Bildverarbeitung erkennen könnte?

Linien- und Kantendetektion

- **Helligkeitsschwellwert:**
  - unmöglich
- **Farbsegmentierung:**
  - rote Türen, blaue Türen, gelbe Schilder?
  - Schwierig wegen wechselnder Beleuchtung.
  - Vielleicht Farbe relativ zur Bodenfarbe bestimmen.
- **Kanten:**
  - Viele gerade Konturen in Büroumgebungen.
  - Oft senkrecht.
  - Ziemlich Beleuchtungsunabhängig.
  - Gutes Merkmal.
  - Aber Verwechselungsgefahr durch viele Kanten

Vertikale Linien

Linien- und Kantendetektion

Linien- und Kantendetektion

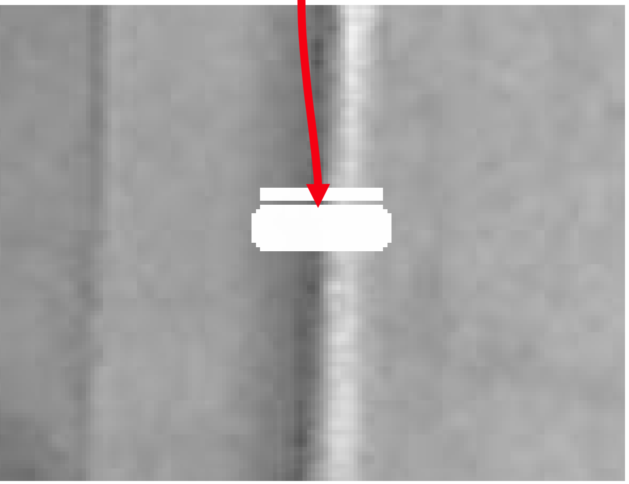
- **Bisher:**
  - Grauwert Schwellwert oder Farbsegmentierung
  - Objekt muss eine durchgängige Region im Bild sein
- **Probleme, wenn**
  - Verdeckung / Überlappung
  - Innere Struktur (Muster, Aufdruck, …)
  - Reflexionen
- **Lösung**
  - Merkmal des Randes nicht der Region
  - geometrische Struktur toleriert Verdeckungen
  - z.B. „Kanten sind gerade“

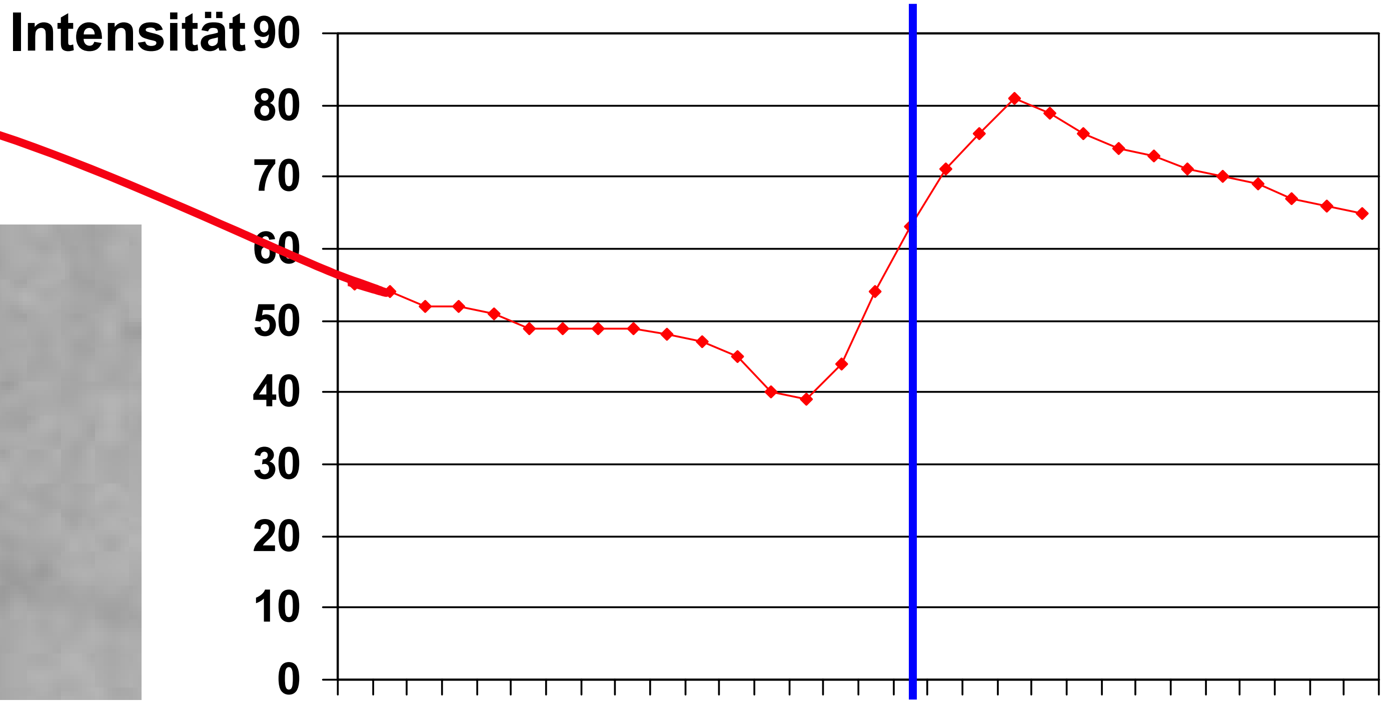
Linien- und Kantendetektion

- **Erster Schritt: Gehört ein Pixel zu einer Kante?**
  - Analog zu „gehört ein Pixel zum Objekt“ bei Schwellwert / Farbsegmentierung
  - Betrachte nicht nur Pixel alleine
  - Betrachte 3×3 Umgebung um Pixel
  - Harte Entscheidung Kante / Nicht-Kante vermeiden
  - Graduelles Ergebnis
- **Zweiter Schritt: Fasse die Kantenpunkte zu Kurven zusammen**
  - Hier: zu Geraden
  - Festes Modell dadurch Toleranz gegenüber Verdeckung
  - Geraden statt Strecken, dadurch nicht Bestimmung der Endpunkte

Linien- und Kantendetektion

Horizontaler Schnitt durch eine vertikale Kante





Idee: Eine Kante ist dort, wo sich die Intensität schnell ändert.

Linien- und Kantendetektion

Sobel-Filter

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

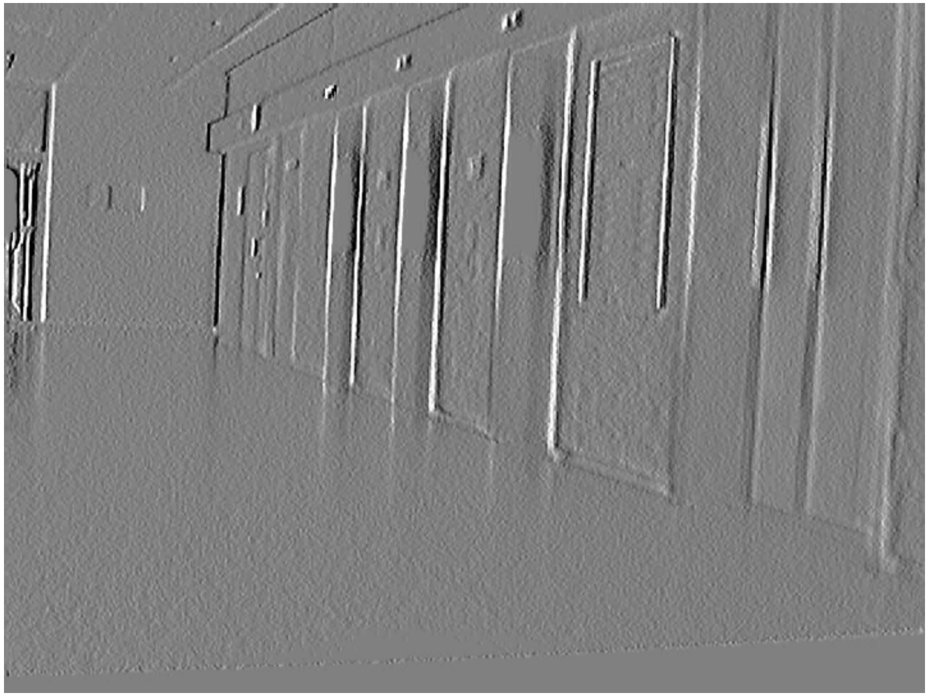
Sobel X
(vertikale Kanten)

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

Sobel Y
(horizontale Kanten)

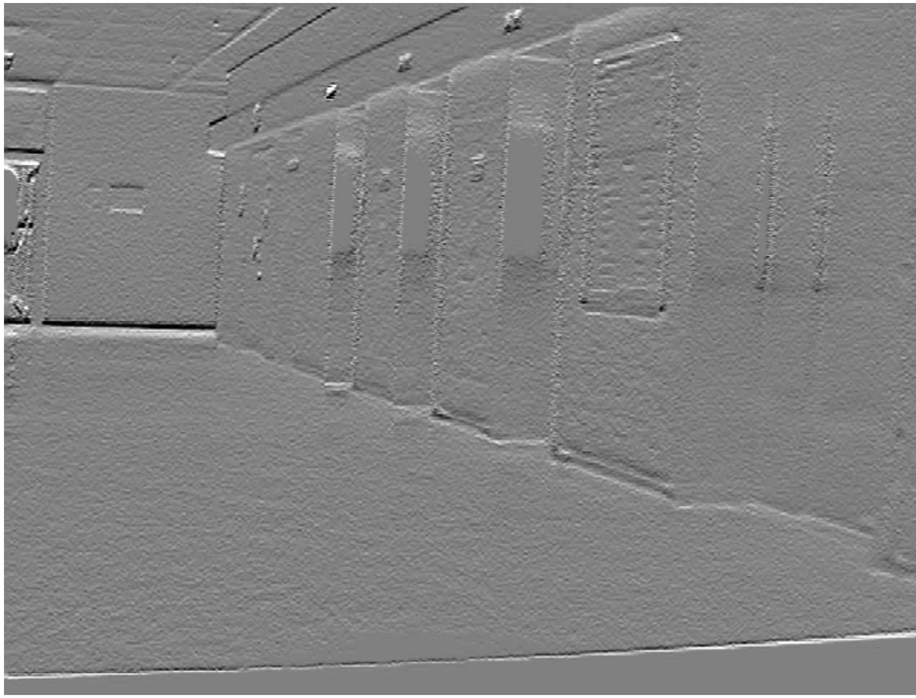
- **Kombination aus Differenz (quer) und Mittelwert (längs)**
- **Auch mit Faktor 1/8 üblich.**
- **Es gibt noch weitere ähnliche Filter.**

Linien- und Kantendetektion



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

Sobel X
(vertikale Kanten)



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

Sobel Y
(horizontale Kanten)

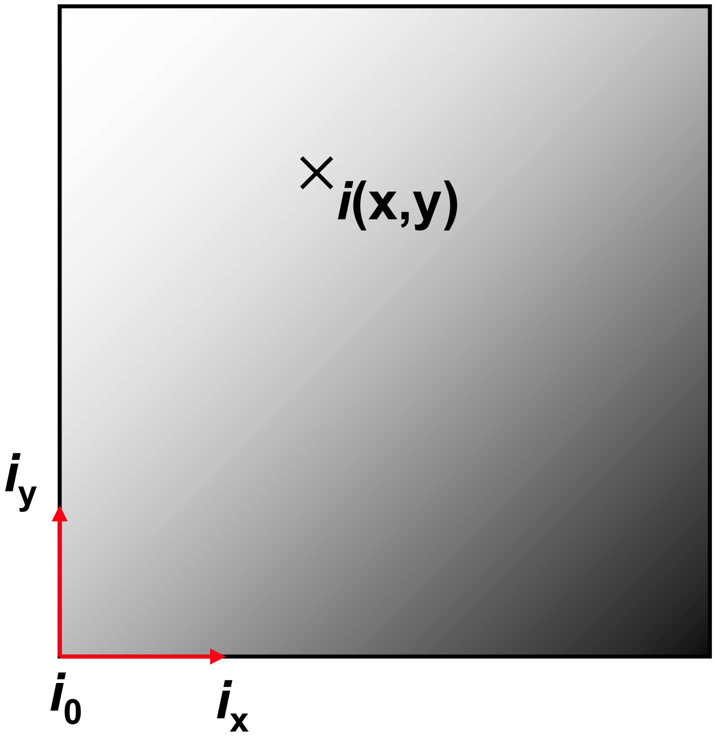
Linien- und Kantendetektion

Was sagt der Sobel-Filter?

- Angenommen, wir haben ein Bild mit einem linearen Helligkeitsverlauf, was liefert der Sobel Filter?
- Helligkeit des Pixels x, y ist

$$i(x,y) = i_0 + i_x x + i_y y$$

- i_0: Grundintensität am Pixel 0,0
- i_x: Änderung in X Richtung
- i_y: Änderung in Y-Richtung



Linien- und Kantendetektion

- **Sobel-X (s_x) / Sobel Y (s_y) bestimmen für die Änderung (Gradient) der Helligkeit in X und Y Richtung**

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

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

$$i(x,y) = i_0 + i_x x + i_y y$$

$$\begin{aligned} s_x &= -1\big(i_0 + i_x(x-1) + i_y(y-1)\big) + 1\big(i_0 + i_x(x+1) + i_y(y-1)\big) \\ &\quad + -2\big(i_0 + i_x(x-1) + i_y(y)\big) + 2\big(i_0 + i_x(x+1) + i_y(y)\big) \\ &\quad + -1\big(i_0 + i_x(x-1) + i_y(y+1)\big) + 1\big(i_0 + i_x(x+1) + i_y(y+1)\big) \\ &= i_x + i_x + 2i_x + 2i_x + i_x + i_x \\ &= 8i_x \end{aligned}$$

$$s_y = 8i_y$$

- **Sobel X / Y liefert den Gradienten in X / Y-Richtung**

Linien- und Kantendetektion

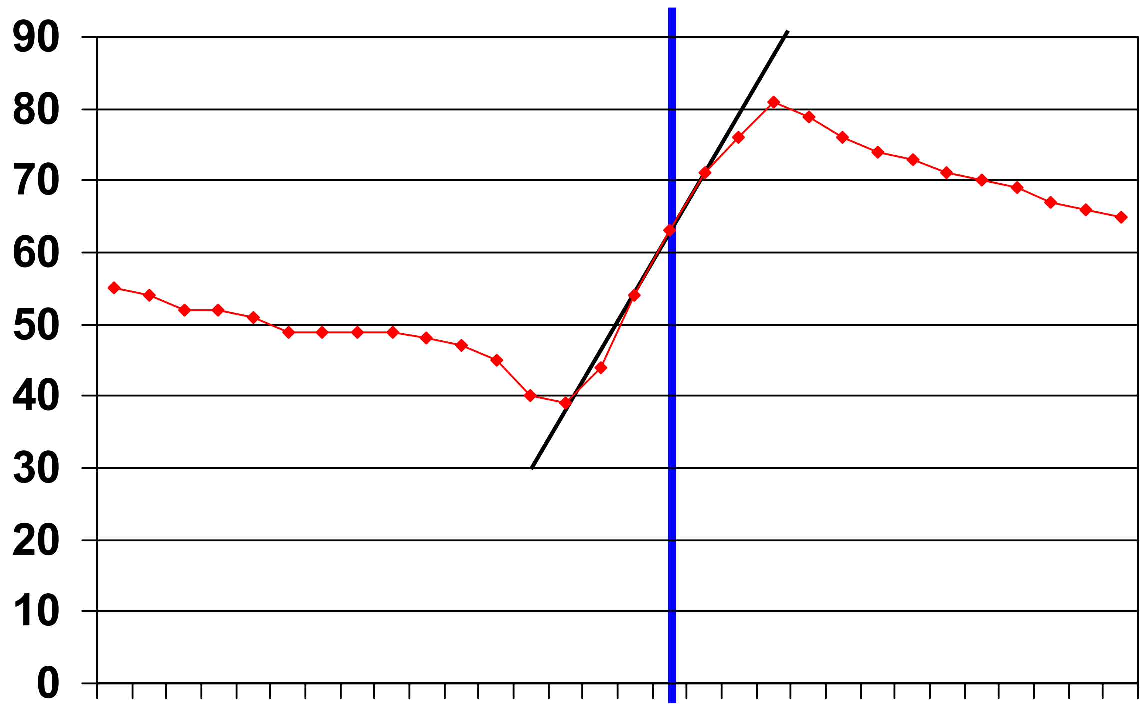
- **Was passiert mit schrägen Kanten?**
  - Sowohl Sobel X als auch Sobel Y spricht an.
  - Je horizontaler die Kante, je mehr Sobel Y
  - Je vertikaler die Kante, je mehr Sobel X
- **Sobel X (s_x) und Sobel Y (s_y) als Vektor zeigen die Richtung einer schrägen Kante an.**
  - Betrag des Vektors gibt Intensität der Kante an
  - Richtung des Vektors gibt Winkel quer zur Kante an (von dunkel nach hell)
- **Im folgenden Herleitung und Erklärung**

$$\sqrt{s_x^{\,2} + s_y^{\,2}}$$

$$\arctan 2\left(s_y, s_x\right)$$

Linien- und Kantendetektion

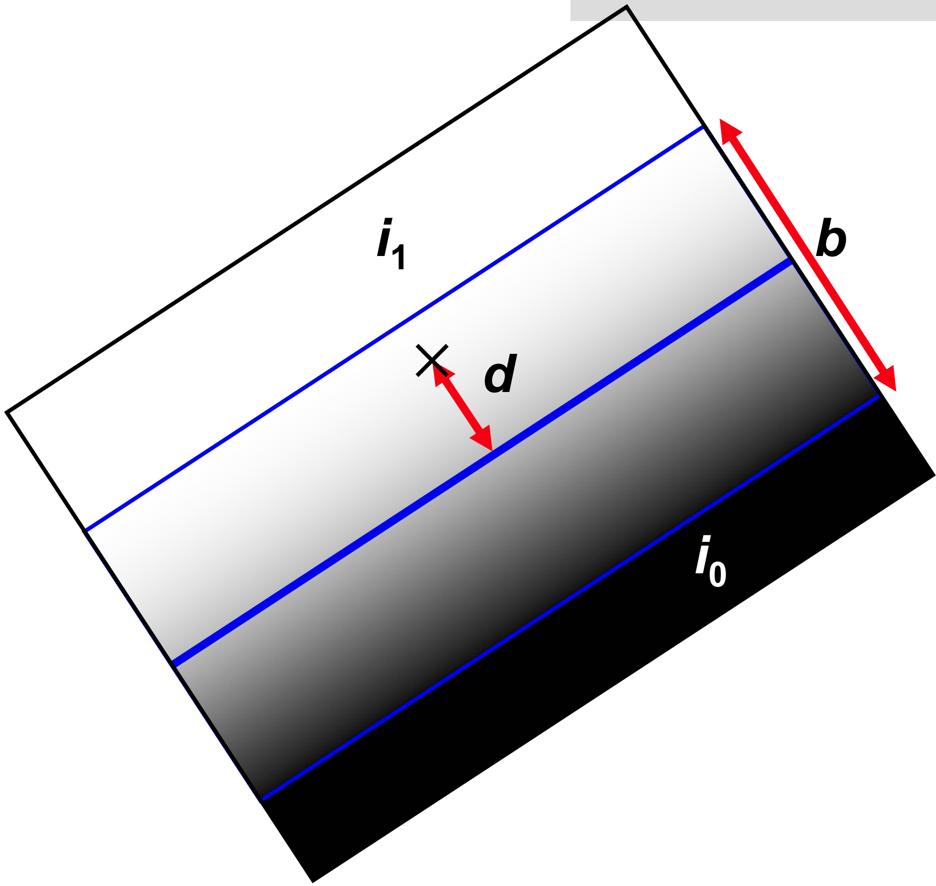
- **Beobachtung: An einer Kante ist ein (kleiner) Streifen linearen Helligkeits-wechsel**



Linien- und Kantendetektion

- **Ideale Linie (dick blau) ist umgeben von einem Bereich linearen Helligkeitsverlaufs (dünn blau)**
- **Helligkeitsverlauf von i_0 nach i_1 hat Breite *b***
- **Helligkeit eines Punktes mit vorzeichenbehaftetem Abstand *d* zur Kante ist**

$$\frac{i_0 + i_1}{2} + \frac{d}{b}\left(i_1 - i_0\right)$$

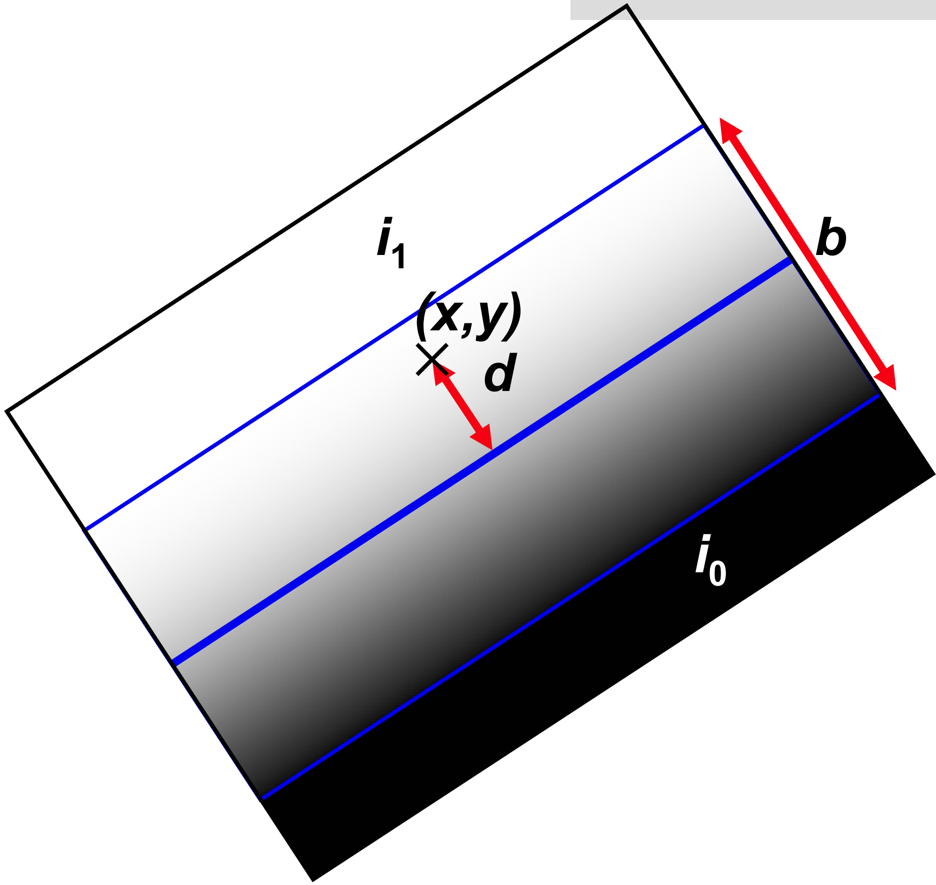


Linien- und Kantendetektion

- **Abstand d ergibt sich aus Koordinaten (x,y) und Hessescher Normalform (p Abstand Kante zum Ursprung, α Richtung Normalenvektor auf i_1)**

$$d = p + \cos\alpha \cdot x + \sin\alpha \cdot y$$

$$i = \frac{i_0 + i_1}{2} + \frac{p + \cos\alpha \cdot x + \sin\alpha \cdot y}{b}(i_1 - i_0)$$



Linien- und Kantendetektion

$$i = \frac{i_0 + i_1}{2} + \frac{p + \cos\alpha \cdot x + \sin\alpha \cdot y}{b}(i_1 - i_0)$$

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

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

- **Anwendung des vorherigen Ergebnisses ergibt**

$$s_x = 8\cos\alpha \frac{i_1 - i_0}{b} \qquad s_y = 8\sin\alpha \frac{i_1 - i_0}{b}$$

$$\sqrt{s_x{}^2 + s_y{}^2} = 8\frac{i_1 - i_0}{b}, \quad \arctan 2(s_y, s_x) = \alpha$$

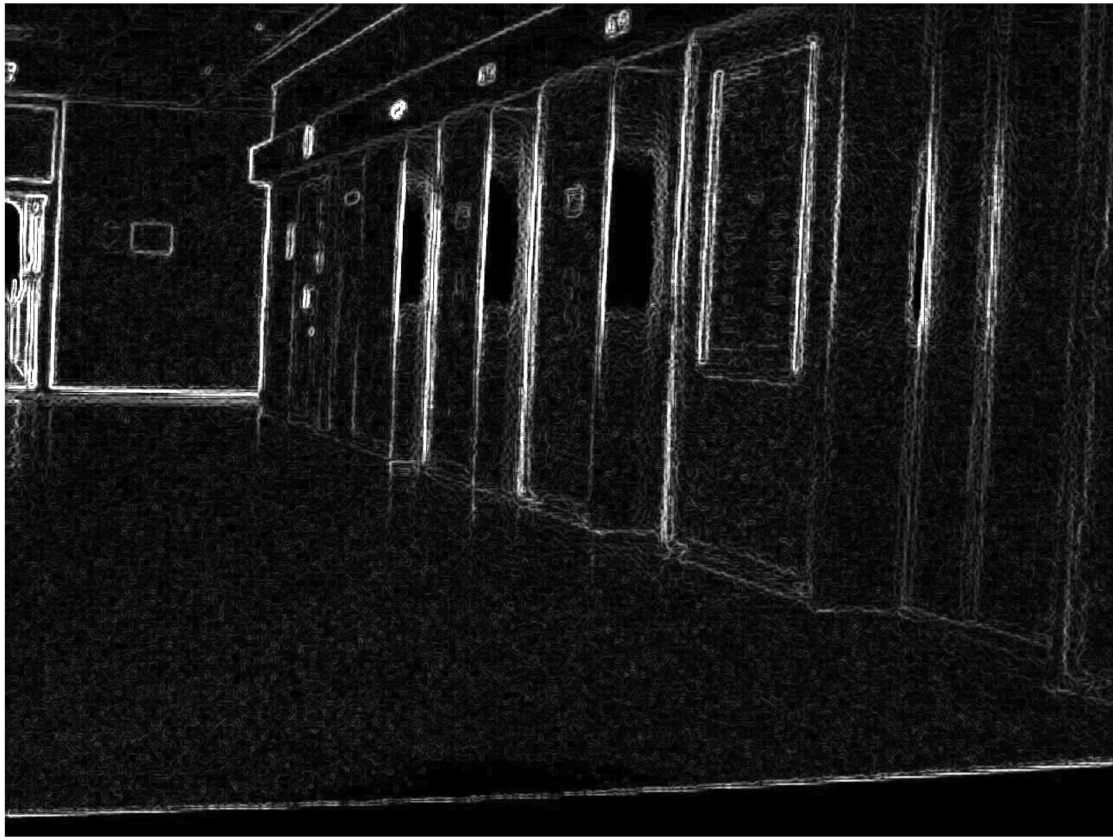
- **Die Sobellänge liefert richtungsunabhängig die Intensität der Kante**
- **Die Sobelrichtung liefert die Richtung senkrecht zur Kante**

Linien- und Kantenerkennung

- **Betrag des Sobelvektors**

$$\sqrt{s_x^{\ 2} + s_y^{\ 2}} = 8\frac{i_1 - i_0}{b}$$

- **zeigt Kanten in alle Richtungen**



Linien- und Kantendetektion

- **Sobel Operator findet Kanten / Linien**
- **Sobel X (s_x) und Sobel Y (s_y) als Vektor geben den Helligkeitsgradienten an**
  - Betrag des Vektors ist Maß für Stärke der Kante
  - Vorhandensein einer Kante mit Schwellwert
  - Richtung des Vektors gibt Richtung der Kante an
  - senkrecht zur Kante zur helleren Seite
- **Linien führen zu zwei hohen Sobelwerten mit umgekehrten Vorzeichen**
  - Daher (s_x,s_y) und $(-s_x,-s_y)$ gleich behandeln
  - Kantenorientierung: 360°, Linienorientierung 180°

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

Sobel X (vertikale Kanten)

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

Sobel Y (horizontale Kanten)

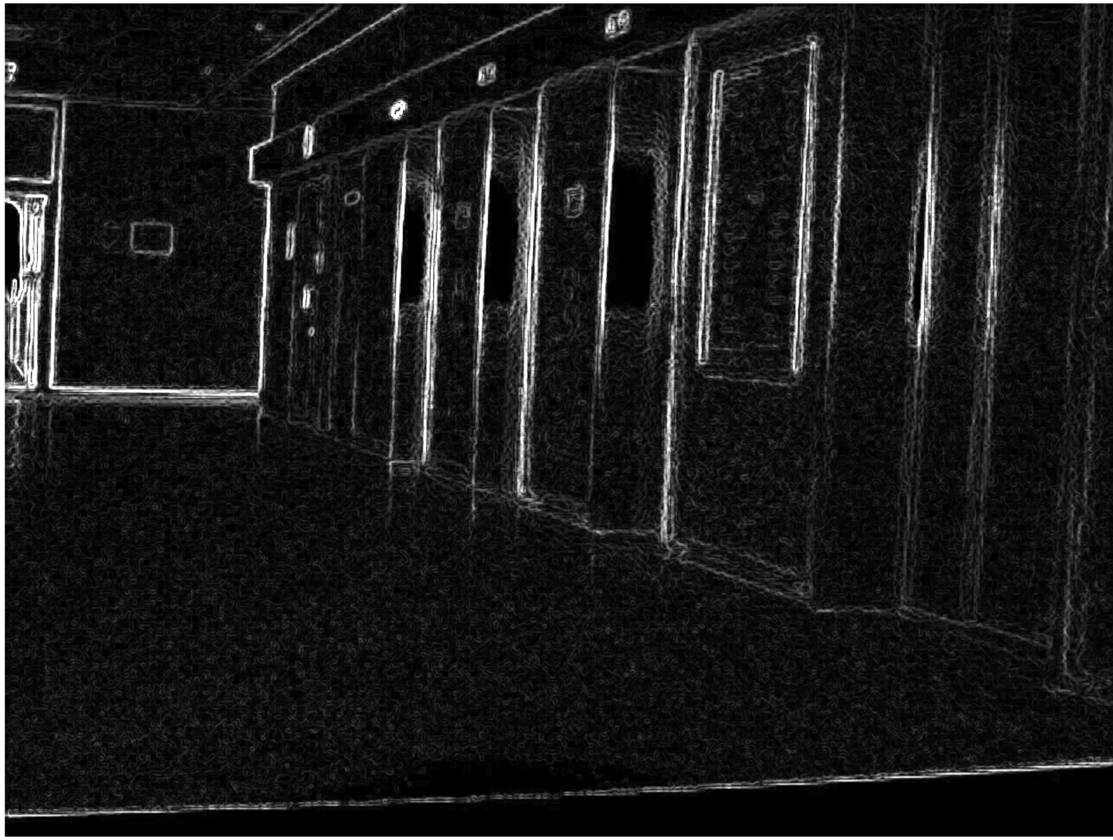
Kontrastnormalisierter Kantenfilter

Kontrastnormalisierter Kantenfilter

- **Betrag des Sobelvektors liefert Helligkeitsänderung quer zur Kante**

$$\sqrt{s_x{}^2 + s_y{}^2} = 8\frac{i_1 - i_0}{b}$$

- **abhängig von Kontrast und Unschärfe**
- **schwer einen Schwellwert zu setzen**
- **gesucht: beleuchtungsunabhängiger Kantenfilter**



Kontrastnormalisierter Kantenfilter

- Frage an das Auditorium: Wie könnte man aus dem Sobelvektor eine Größe konstruieren, die beleuchtungsunabhängig ist?

Kontrastnormalisierter Kantenfilter

- <u>Frage an das Auditorium</u>: Wie könnte man aus dem Sobelvektor eine Größe konstruieren, die beleuchtungsunabhängig ist?
- Vektor normieren
- $\Rightarrow$ nur die Sobelrichtung betrachten
- Nachteil: sagt nichts über „Stärke" der Kante

Kontrastnormalisierter Kantenfilter

Beleuchtungsinvarianter Kantenfilter:

- **betrachte Raum $R^{3\times 3}$ der 3*3 Bildausschnitte**
- **Mitte ist der Punkt, an dem die Kante detektiert werden soll**
- **definiere Skalarprodukt, das die Mitte stärker gewichtet**

$$\langle L, L' \rangle = \sum_{x,y=-1}^{1} w_{xy} L_{xy} L'_{xy} \qquad w = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- **dadurch euklidischer Vektorraum mit Begriffen wie Länge, Winkel, orthonormal, Projektion, etc.**
- **definiere eine Orthonormalbasis $v_{1..9}$, in der Dimensionen vorkommen, die für Kantenerkennung eine Bedeutung haben**

Kontrastnormalisierter Kantenfilter

<u>Beleuchtungsinvarianter Kantenfilter:</u>

$$v_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad v_{2x} = \sqrt{2}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

- v_1 ist additive Beleuchtungsänderung
- v_1 soll ignoriert werden
- v_{2x} ist linearer Gradient in X-Richtung
- v_{2x} soll detektiert werden

$$\langle v_1, v_1 \rangle = 1 \qquad \langle v_{2x}, v_{2x} \rangle = 1 \qquad \langle v_1, v_{2x} \rangle = 0$$

- $v_{1..2}$ sind orthonormal und werden zu Orthonormalbasis ergänzt

Kontrastnormalisierter Kantenfilter

Beleuchtungsinvarianter Kantenfilter:

- **projiziere Bildausschnitt L in die Richtungen der Basis und quadriere**
- **durch $\langle L, v_i \rangle^2$**
- **Ergebnis: 9-Vektor, in dem**
  - 1. Komponente ($\langle L, v_1 \rangle^2$) ignoriert werden soll
  - 2. Komponente ($\langle L, v_{2x} \rangle^2$) Indiz für Kante ist
  - 3.-9. Komponente ($\langle L, v_i \rangle^2$, i=3..9) Indiz gegen Kante ist
  - alle Komponenten zusammen die quadrierte Länge von L ergeben

$$\langle L, L \rangle = \sum_{i=1}^{9} \langle L, v_i \rangle^2$$

Kontrastnormalisierter Kantenfilter

Beleuchtungsinvarianter Kantenfilter:

- **Definiere Detektor: 2. Komponente durch Summe aller außer der 1.**

$$c_x^2 = \frac{\langle L, v_{2x} \rangle}{\sum_{i=2}^{9} \langle L, v_i \rangle^2} = \frac{\langle L, v_{2x} \rangle^2}{\langle L, L \rangle - \langle L, v_1 \rangle^2}$$

- **linear beleuchtungsinvariant ($L \rightarrow \alpha L + \beta v_1$)**
  - additive Änderung ($+\beta v_1$) wird ignoriert, weil $\langle v_1, v_i \rangle = 0$ für i=2..9
  - multiplikative Änderung (αL) kürzt sich raus
- **Nenner ist „lokaler Bildkontrast“ (Varianzformel)**
- **Intuition:**
  „Welcher Anteil des lokalen Bildkontrastes stammt von einem X-Gradienten?“

Kontrastnormalisierter Kantenfilter

<u>Beleuchtungsinvarianter Kantenfilter:</u>

- **Analog für c^2_y Kanten in Y-Richtung (v_{2y})**
- **Analog für Kanten in Richtung α mit**

$$v(\alpha) = \cos\alpha v_{2x} + \sin\alpha v_{2y}$$

$$c_\alpha^2 = \frac{\langle L, \cos\alpha v_{2x} + \sin\alpha v_{2y}\rangle^2}{\langle L, L\rangle - \langle L, v_1\rangle^2} = \frac{\left(\cos\alpha\langle L, v_{2x}\rangle + \sin\alpha\langle L, v_{2y}\rangle\right)^2}{\langle L, L\rangle - \langle L, v_1\rangle^2}$$

- **Trenne Abhängigkeit von α heraus**

$$\left(c_x, c_y\right) = \frac{\left(\langle L, v_{2x}\rangle, \langle L, v_{2y}\rangle\right)}{\sqrt{\langle L, L\rangle - \langle L, v_1\rangle^2}}, \qquad c_\alpha^2 = \left(\cos\alpha c_x + \sin\alpha c_y\right)^2$$

Kontrastnormalisierter Kantenfilter

$$\left(c_x, c_y\right) = \frac{\left(\left\langle L, v_{2x}\right\rangle, \left\langle L, v_{2y}\right\rangle\right)}{\sqrt{\left\langle L, L\right\rangle - \left\langle L, v_1\right\rangle^2}}, \qquad c_\alpha^2 = \left(\cos\alpha c_x + \sin\alpha c_y\right)^2$$

- **Rechnung auf Bild: Skalarprodukte werden zu Filtern**

$$\frac{\left(\frac{\sqrt{2}}{16}\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} * I, \frac{\sqrt{2}}{16}\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix} * I\right)}{\sqrt{\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I^2 - \left(\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I\right)^2}} = \frac{\sqrt{2}\left(\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} * I, \begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix} * I\right)}{\sqrt{16\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I^2 - \left(\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} * I\right)^2}}$$

- **Vektor (c_x,c_y) ist skalierter Sobelvektor**
  - Richtung quer zur Kantenrichtung
  - Beleuchtungsinvariant
  - Länge^2: Welcher Anteil des lokalen Bildkontrastes ist ein linearer Gradient?
  - Gradienten*reinheit* nicht -*stärke*

Zusammenfassung

- **Faltungsoperationen**
  - Lineare, translationsinvariante Abbildung
  - Ergebnispixel ist gewichtete Summe der Pixel in der Umgebung des Eingangspixels
  - Bilder glätten, Kontrast vergrößern, Kanten detektieren
- **Kantendetektion mit dem Sobel Filter (SobelX, SobelY)**
  - SobelX und SobelY geben als Ergebnis einen Vektor
  - Betrag: Kantenstärke
  - Richtung: Richtung der Kante (senkrecht zum Helleren)
- **Kontrastnormalisierter Kantenfilter**
  - Sobel Vektor durch Kontrast dividiert
  - Beleuchtungsinvariant
  - Betrag²: Welcher Anteil des Bildkontrastes ist linearer Gradient
  - Gradienten*reinheit* nicht -*stärke*

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

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