

Grundlagen der Künstlichen Intelligenz

19 Wahrnehmung: Basics & Low-Level Vision

Bildverstehen als inverses Problem

Phasen des Bildverstehens

Glättung und Kantenerkennung

Volker Steinhage

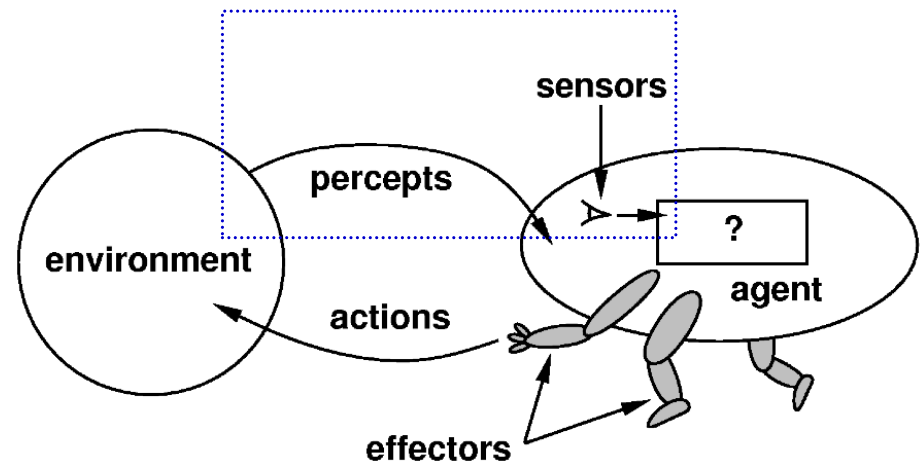
Inhalt

- Wahrnehmung (Perception) als inverses Problem
- Bildentstehung und Bildaufbau
- Phasen des Bildverstehens
- Strategien des Bildverstehens
- Low-Level Vision
 - Konvolution
 - Glättung
 - Kantenerkennung

Wahrnehmung/Perzeption

Die **Wahrnehmung** (Perception)

- bietet dem Agenten Information über seine Umwelt
- wird durch Sensoren initiiert



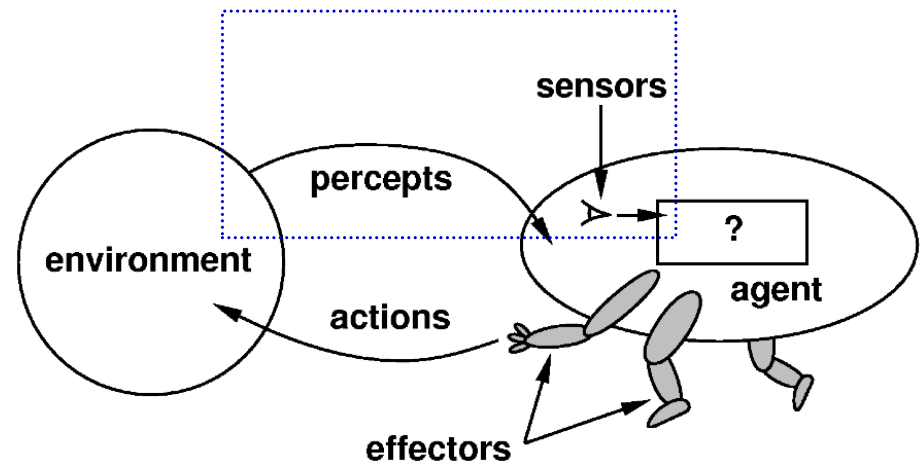
Sensoren sind

- alles, was einen oder mehrere Aspekte der Umgebung aufzeichnet und als Eingabe für ein Agentenprogramm geeignet ist
- Können einfache 1-Bit-Sensoren für eine Schaltererkennung sein oder komplexe Sensoren wie die menschliche Netzhaut

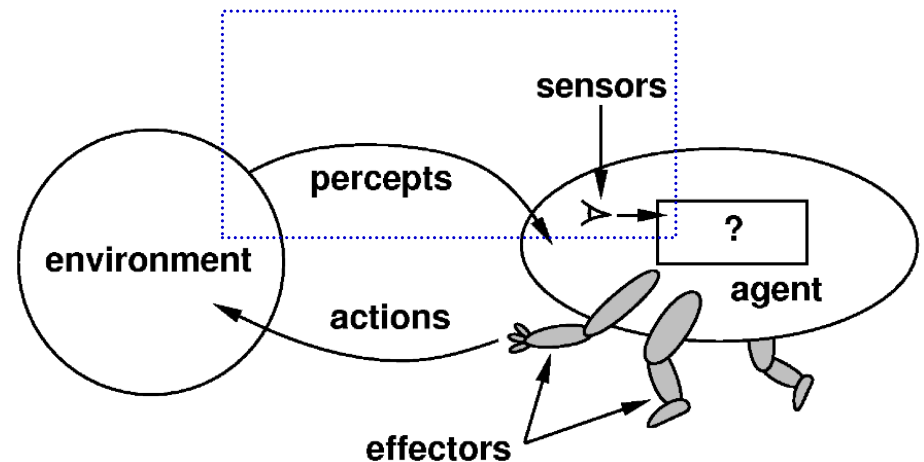
Interpretation von Wahrnehmung (1)

Der Agent

- **erkennt** bestimmte **Merkmale** in seiner **sensorischen Eingabe**
- **interpretiert** die **Merkmale** zu einer **Informationsbeschreibung**
- **kombiniert** diese **Information** mit vorhandener **Wissensbasis**,
- **wählt** eine **Aktion** aufgrund der aktualisierten **Wissensbasis** aus.



Interpretation von Wahrnehmung (2)



Bspl. 1: der **Wumpus-Agent** hatte **5 binäre Sensoren** für Geruch (j/n), Luftzug (j/n), Glitzern von Gold (j/n), Stoß gegen die Wand (j/n), Schrei (j/n)

Bspl. 2: **Stubenfliegen** können bestimmte Funktions**merkmale** aus dem optischen Strom (Perzepte) erkennen und direkt an die für die Flugsteuerung verantwortlichen Muskeln (Effektoren) weitergeben.

Bildverstehen als inverses Problem (1)

- Ausgangspunkt ist die Annahme einer Funktion f , welche die *relevante Umwelt W* auf den *Sensorstimulus S* abbildet:

$$S = f(W).$$

Bei der visuellen Wahrnehmung ist diese durch Physik und Optik definiert und i.W. durch die *Computergrafik* gelöst.

- Das *Bildverstehen* (engl. *Computer Vision*) ist im gewissen Sinne die Umkehrung der Computergraphik:

berechne die abgebildete Welt W aus gegebenem Funktional f und Sensorstimulus S nach

$$W = f^{-1}(S).$$

Daher wird die Computer Vision auch als 'Inverse Computergrafik' bezeichnet

Bildverstehen als inverses Problem (2)

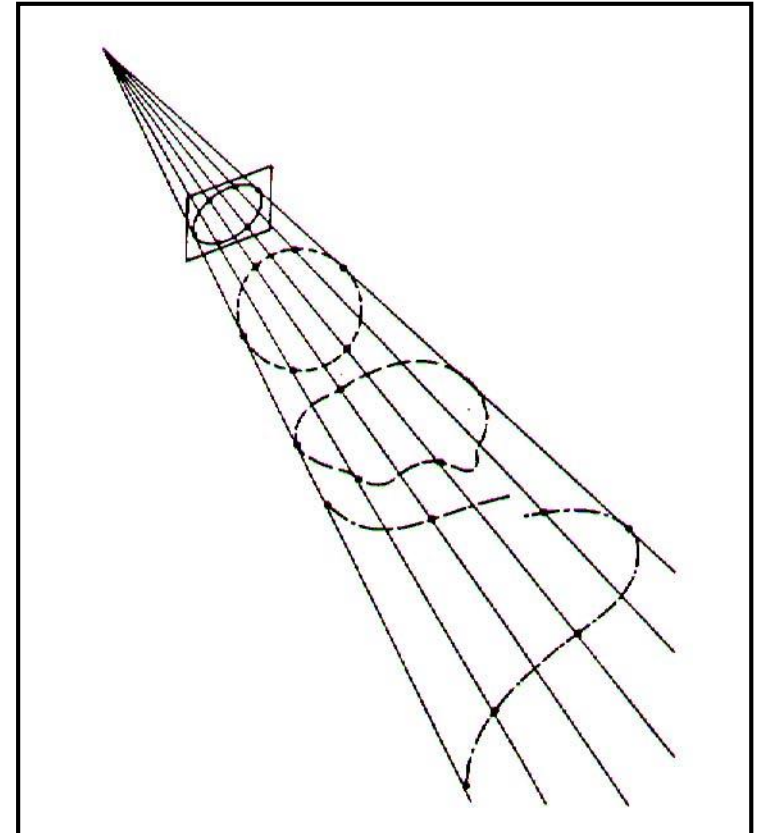
Bildverstehen als Rekonstruktion der abgebildeten Welt W für ein gegebenes **Abbildungsfunktional** f und einen Sensorstimulus S nach

$$W = f^{-1}(S)$$

beschreibt ein ***inverses Problem***: aus der beobachteten *Wirkung* eines Systems (die Abbildung) wird auf die zugrunde liegenden *Ursachen* (abgebildete Welt) geschlossen.

Dieses Interpretationsproblem ist i.A. ***unterbestimmt*** (engl. ***ill-posed***), da die Interpretation f^{-1} generell ***mehrdeutig*** ist.

Das Bspl. skizziert drei räumliche Interpretationen einer Bildbeobachtung.



Bildverstehen als inverses Problem (3)

Mehrdeutigkeit der Interpretation haben verschiedene Ursachen:

- Verlust der Tiefeninformation durch Projektion von 3D-Objekten auf 2D-Bilder,
- perspektivische Verzeichnung,
- ggf. radiale Linsenverzeichnung (im Extrem: Fischaugenobjektiv),
- begrenzte Auflösung von Fotofilm oder CCD-Chip,
- Reduktion der Farben auf begrenzte Farbtiefe der Bilder (im Extrem auf Grauwerte oder Schwarz-Weiß-Bilder),
- mangelnde Kontraste,
- teilweise oder vollständige Verdeckungen,
- ähnliche Erscheinungen verschiedener Objekte,
- ... und Verlust des Kontextes.



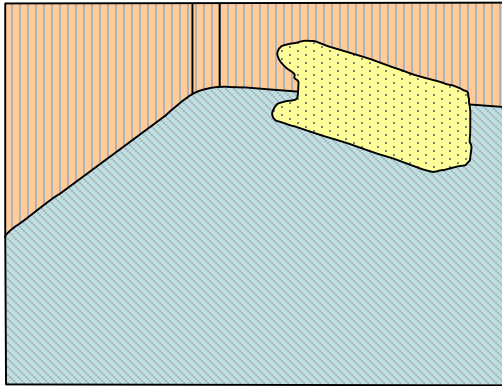
Zeichnung des Cartoonisten William Ely Hill (1887–1962)

Phasen des Bildverstehens (1)

Die Verarbeitung der visuellen Information wird in drei Phasen untergliedert:

- 1) *Early Vision* oder *Low-Level Vision*: (a) *Glättung* des Rohbildes um Rauschen zu eliminieren. (b) Hervorhebung relevanter *Bildpunkte*, häufig von *Konturpunkten* von Objektgrenzen.
- 2) *Mid-Level Vision*: Gruppierung extrahierter Konturpunkte zu *Konturlinien*. Die Konturlinien zerlegen das Bild wiederum in flächenhafte Bereiche, die sog. *Bildsegmente*. Alternativ Gruppierung ähnlicher Bildpunkte zu Segmenten.
- 3) *High-Level Vision*: Bildsegmente werden als Objekte der abgebildeten Szene erkannt. Eine inhaltliche Beschreibung als Interpretation des Bildes ist damit ableitbar.

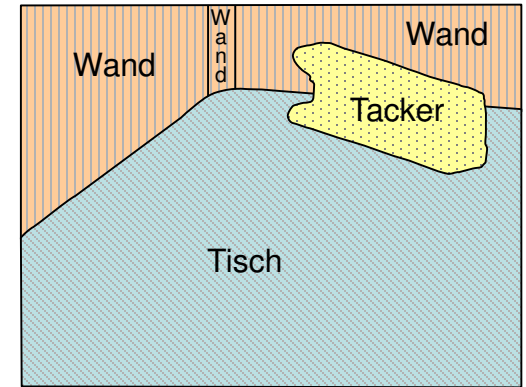
Phasen des Bildverstehens (2)



High-Level Vision



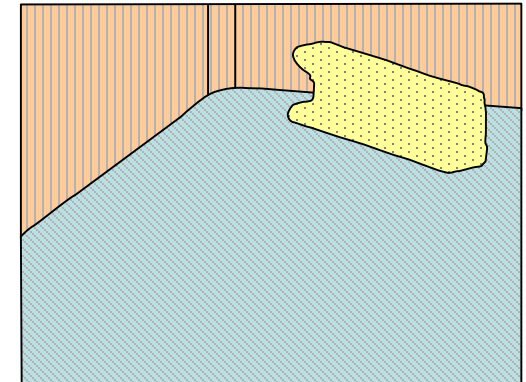
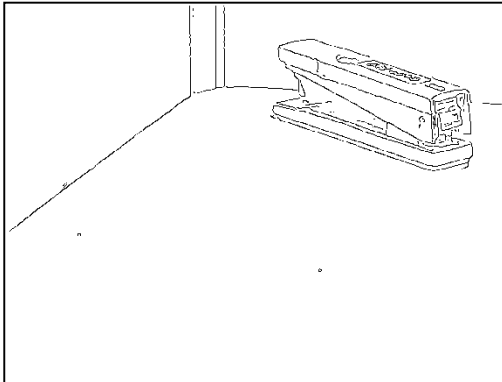
Segmente → Semantik



Mid-Level Vision



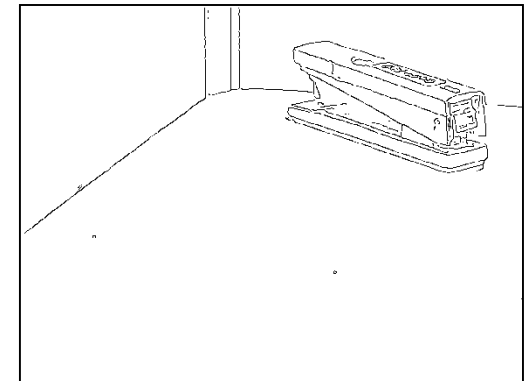
Raster → Segmente



Low-Level Vision



Raster → Raster



Alternative Nomenklatur (1)

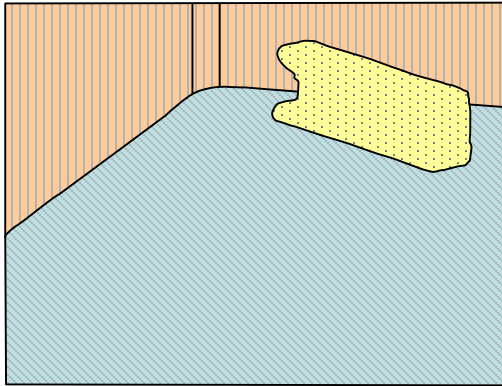
Die Sichtweise von Russel und Norvig ist eine **KI-Sicht** auf das Bildverstehen.

Alternativ wird **Computersehen** (*Computer Vision*) als umfassender Begriff verstanden.

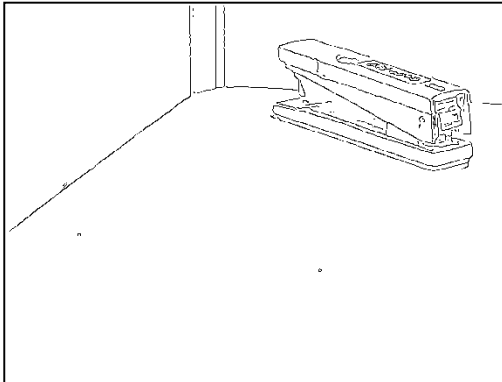
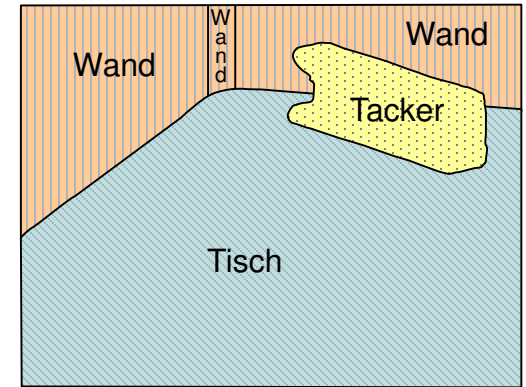
- **Bildverstehen** (engl. *Image Understanding*) entspricht dann im engeren Sinne der High-Level Vision: Zuordnung von Semantik, wissensbasierte Verfahren sowie Inferenz- und Lernverfahren.
- Mid-Level Vision: weitestgehend Verfahren der **Segmentierung** zur Ableitung von geometrisch beschreibbaren Segmenten.
- Low-Level Vision: Verfahren der **Bildverarbeitung** (engl. *Image Processing*) zur Verbesserung, Manipulation und Transformation von Rasterbildern eingesetzt.



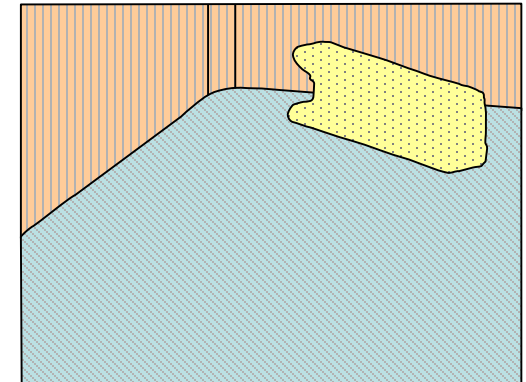
Phasen der Computer Vision



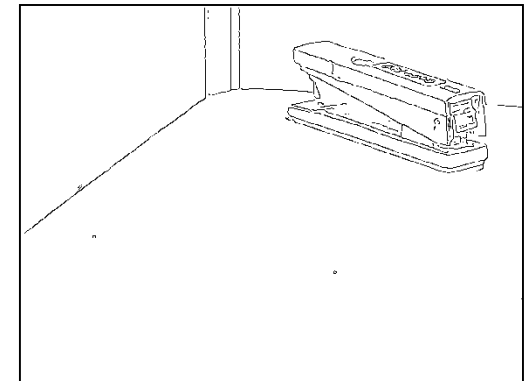
Bildverstehen
Image Understanding



Segmentierung
Segmentation



Bildverarbeitung
Image Processing



Grundsätzliche Strategieansätze (1)

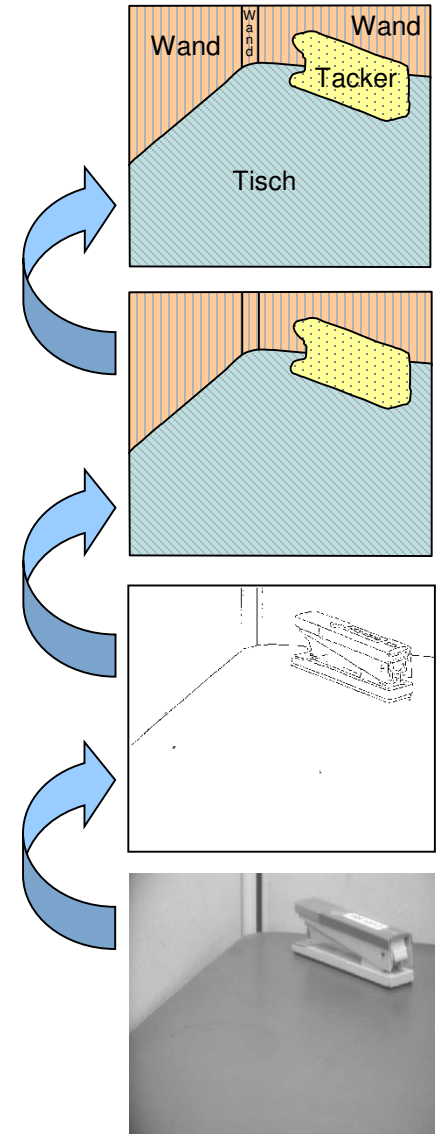
Datengetriebener Ansatz:

Allgemeine Regeln erzeugen Bildsegmente aus den Pixeln durch

- Gruppierung gleichartiger Pixel
- Abgrenzung ungleichartiger Pixel

Messung von Gleich- bzw. Ungleichartigkeit von Pixeln über Helligkeit, Farbe, Umgebungstextur u. a. Kriterien.

Wegen Start mit Sensordaten auf niedriger Modellierungsebene spricht man auch von *Bottom-up*-Ansatz



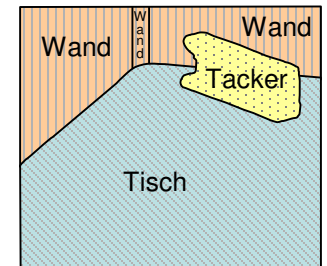
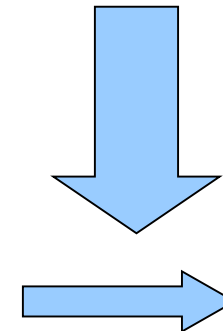
Grundsätzliche Strategieansätze (2)

Modellgetriebener Ansatz:

Modelle relevanter Weltobjekte und ihren inhaltlichen und räumlichen Beziehungen steuern die Prozesse von Mid- und Low-Level-Vision.



Bspl.: Auf einem Schreibtisch suchen wir nach Schreibpapier, Kugelschreibern, ... und Tackern.



Wegen Start mit Modellen der höchsten Modellierungsebene spricht man auch von *Top-down*-Ansatz.

Grundsätzliche Strategieansätze (3)

Heterarchischer Ansatz:

In der Praxis werden Aufgaben nicht nur durch rein *hierarchische Ansätze*, also rein datengetriebene oder rein modellgetriebene Ansätze, gelöst.

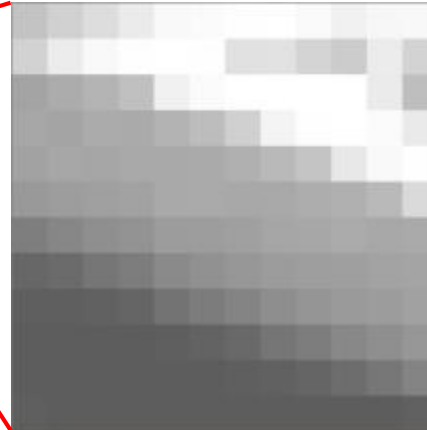
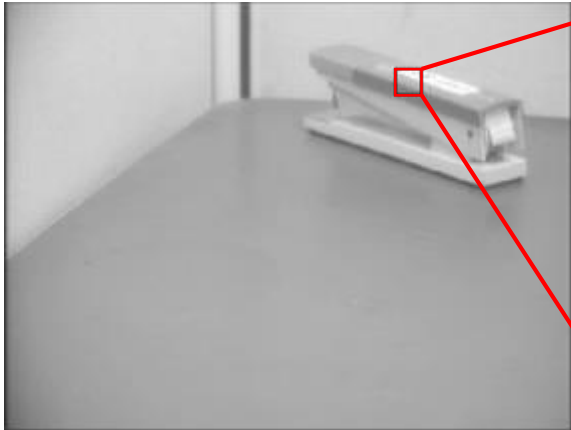
Vielmehr werden Top-down- und Bottom-up-Prozesse anwendungsspezifisch in sog. *heterarchischen* Ansätzen kombiniert.

In den folgenden Vorlesungen werden typische Methoden von Low-Level, Mid-Level- und High-Level-Vision in dieser Reihenfolge vorgestellt.

Damit beginnen wir bei den „Rohbildern“; genauer: wie diese aussehen und entstehen, also dem sog. *Bildaufbau*.

Bildaufbau: Grauwertbild

Was ist ein Bild?

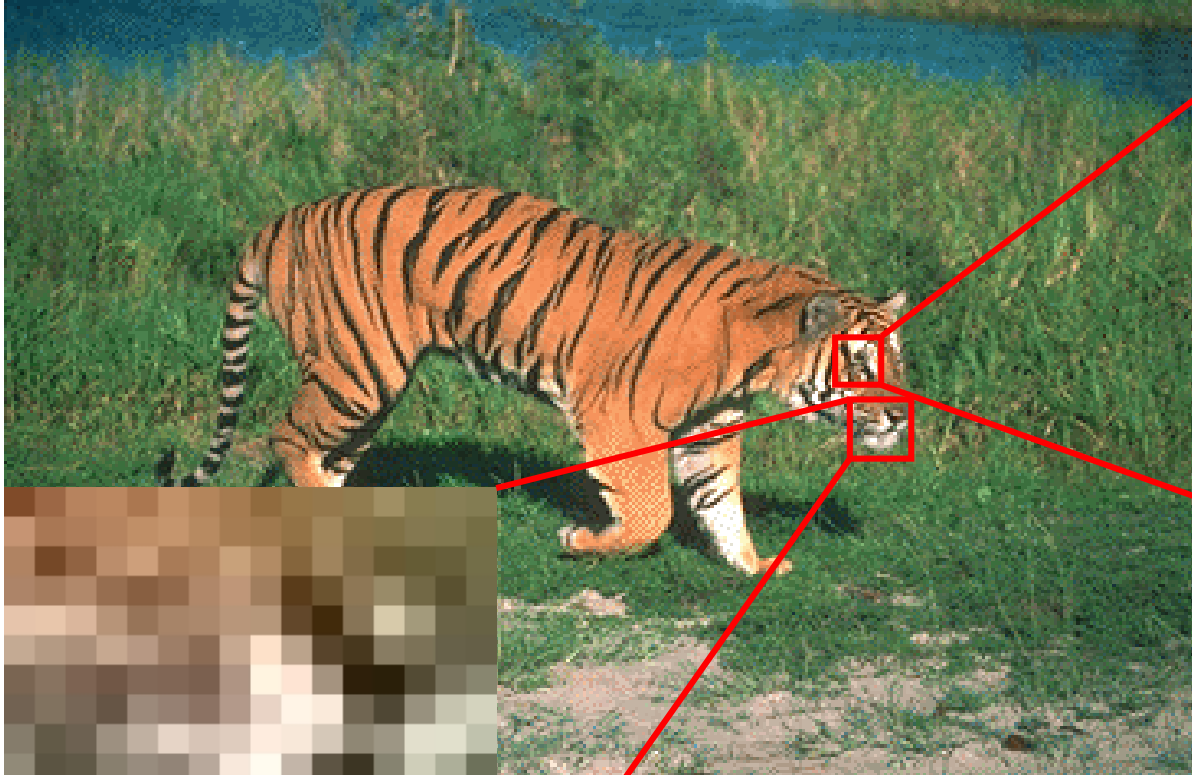


195	209	221	235	249	251	254	255	250	241	247	248
210	236	249	254	255	254	225	226	212	204	236	211
164	172	180	192	241	251	255	255	255	255	235	190
167	164	171	170	179	189	208	244	254	255	251	234
162	167	166	169	169	170	176	185	196	232	249	254
153	157	160	162	169	170	168	169	171	176	185	218
126	135	143	147	156	157	160	166	167	171	168	170
103	107	118	125	133	145	151	156	158	159	163	164
095	095	097	101	115	124	132	142	117	122	124	161
093	093	093	093	095	099	105	118	125	135	143	119
093	093	093	093	093	093	095	097	101	109	119	132
095	093	093	093	093	093	093	093	093	093	093	119

- *Einkanaliges Grauwertbild*: Matrix von Pixeln (engl. Abkürzung für *Picture Elements*). Jedes *Pixel* hat einen Helligkeits- oder *Intensitätswert*.
- Wird jedes Pixel mit 1 Byte kodiert, sind Helligkeitswerte von 0 bis 255 darstellbar.

Bildaufbau: Farbbild

Was ist ein Bild?



RGB-Bilder sind *mehrkanalige Farbwertbilder*. Der Helligkeits- und Farbeindruck ergibt sich durch Addition der drei Intensitätswerte aus dem Rot-, Grün- und Blau-Kanal.

Bildaufbau: Bildkoordinaten

Ein häufig und auch in dieser Vorlesung verwendetes **Bildkoordinatensystem**:

- Ursprung: in der linken unteren Ecke,
- x-Werte für die Bildspalten mit nach rechts steigenden Index.
- y-Werte für die Bildzeilen mit nach oben steigenden Index.



In der Literatur finden sich aber z.T. auch andere Bildkoordinatensysteme, die häufig aus der Nachrichten- und Fernsehtechnik stammen. Zum Beispiel:

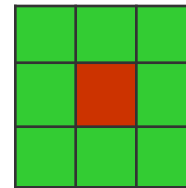
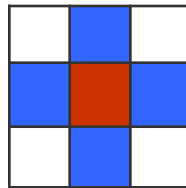
- Ursprung: in der linken oberen Ecke,
- x-Werten für die Bildzeilen mit nach unten steigendem Index,
- y-Werten für die Bildspalten mit nach rechts steigendem Index.



Bildaufbau: Nachbarschaften und Distanzen

Nachbarschaften:

I.A. werden die zu einem **Pixel** *nächsten* Nachbarpixel durch die sog. **4-er Nachbarschaft** oder die sog. **8-er Nachbarschaft** bezeichnet:

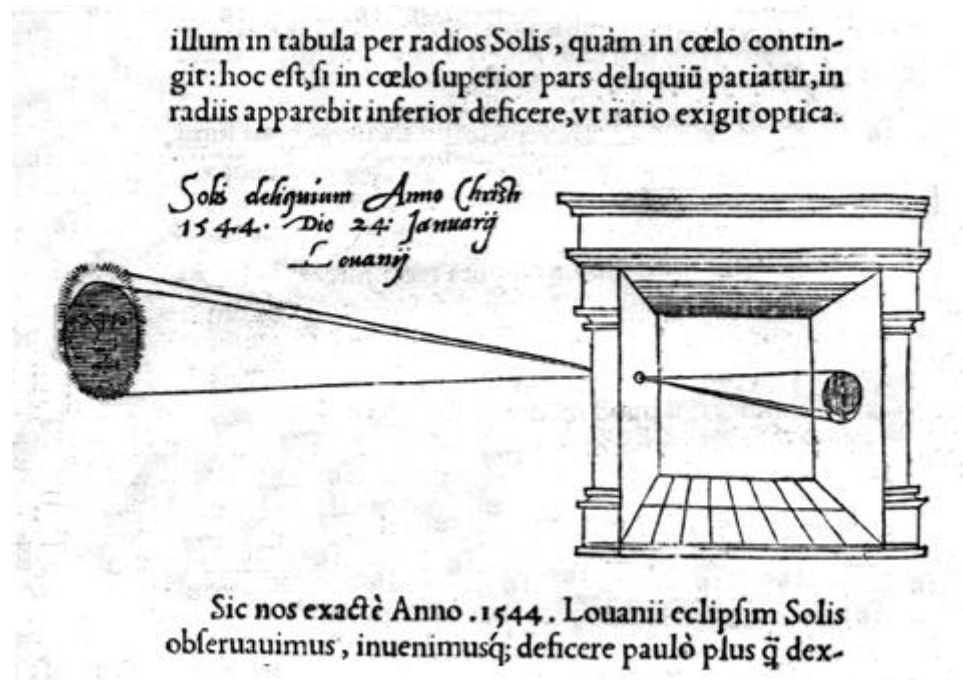


In der Bildanalyse gängige Distanzmaße zwischen Pixeln $p_1 = (x_1, y_1)$ und $p_2 = (x_2, y_2)$ sind die

- **Euklidische Distanz**: $D_e(p_1, p_2) = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2}$,
- **Manhattan-Distanz** oder **City-Block-Distanz**: $D_c(p_1, p_2) = |x_2 - x_1| + |y_2 - y_1|$.

Bildaufbau: Lochkamera (1)

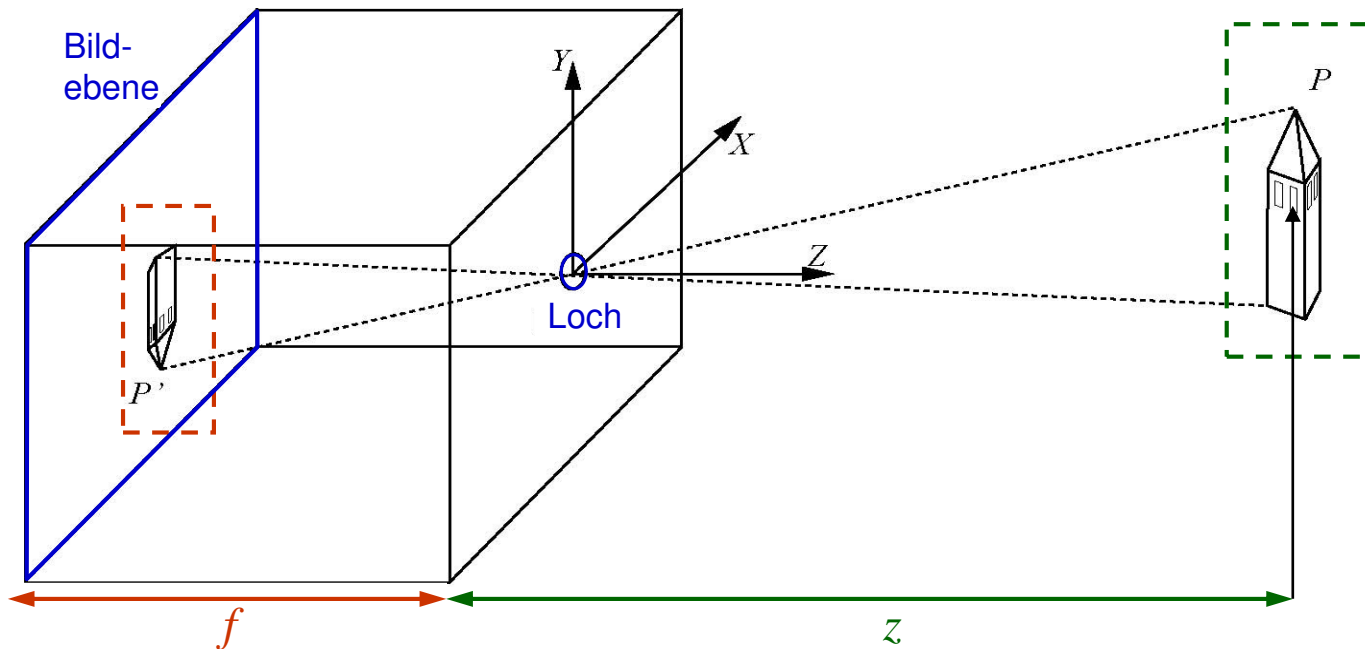
Das einfachste und für viele Anwendungen durchaus hinreichende Modell der Bildgenerierung ist die Verwendung einer **Lochkamera**.



Bildaufbau: Lochkamera (2)

Die Öffnung O definiere den Ursprung des **Kamerakoordinatensystems** (X,Y,Z) .
Dann wird ein **Objektpunkt** $P = (x,y,z)$ auf einen **Bildpunkt** $P' = (x',y',z')$ projiziert.
Mit **fokaler Länge** f und **Objektpunktentiefe** z ergibt sich aus ähnlichen Dreiecken:

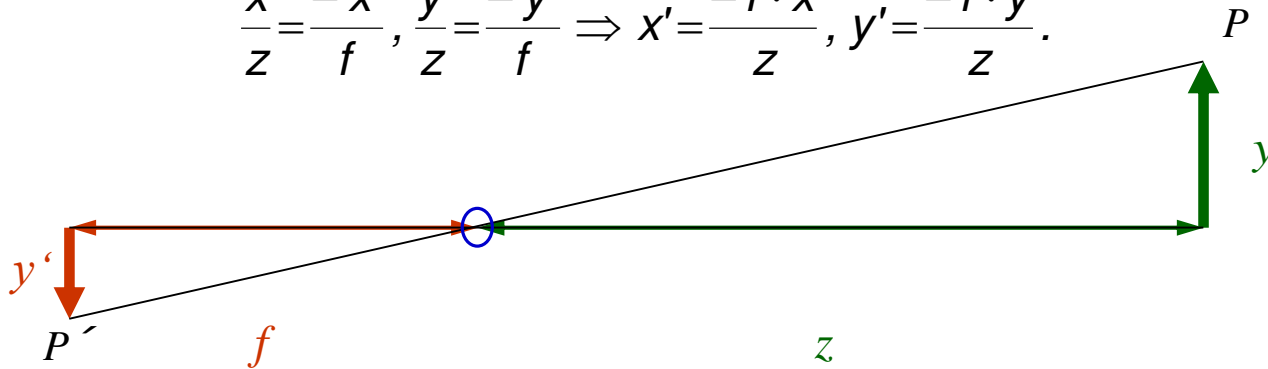
$$\frac{x}{z} = \frac{-x'}{f}, \frac{y}{z} = \frac{-y'}{f} \Rightarrow x' = -\frac{f \cdot x}{z}, y' = -\frac{f \cdot y}{z}.$$



Bildaufbau: Zentralprojektion

Die Abbildungsgleichung definiert die **perspektivische Projektion**:

$$\frac{x}{z} = \frac{-x'}{f}, \frac{y}{z} = \frac{-y'}{f} \Rightarrow x' = \frac{-f \cdot x}{z}, y' = \frac{-f \cdot y}{z}.$$



Eigenschaften der perspektivischen Projektion

- Geraden werden als Geraden abgebildet (**geradentreue** Abbildung).
- Projektionen paralleler Raumgeraden schneiden sich in einem gemeinsamen **Fluchtpunkt**. Man spricht auch von einer **perspektivischen Verzeichnung**. Die Zentralprojektion ist damit **nicht parallelentreu**.
- Die Zentralprojektion entspricht der Abbildung durch ein menschl. Auge und vermittelt einen **natürlichen Bildeindruck**.

Low-Level-Vision: Glättungs- & Kantenfilter

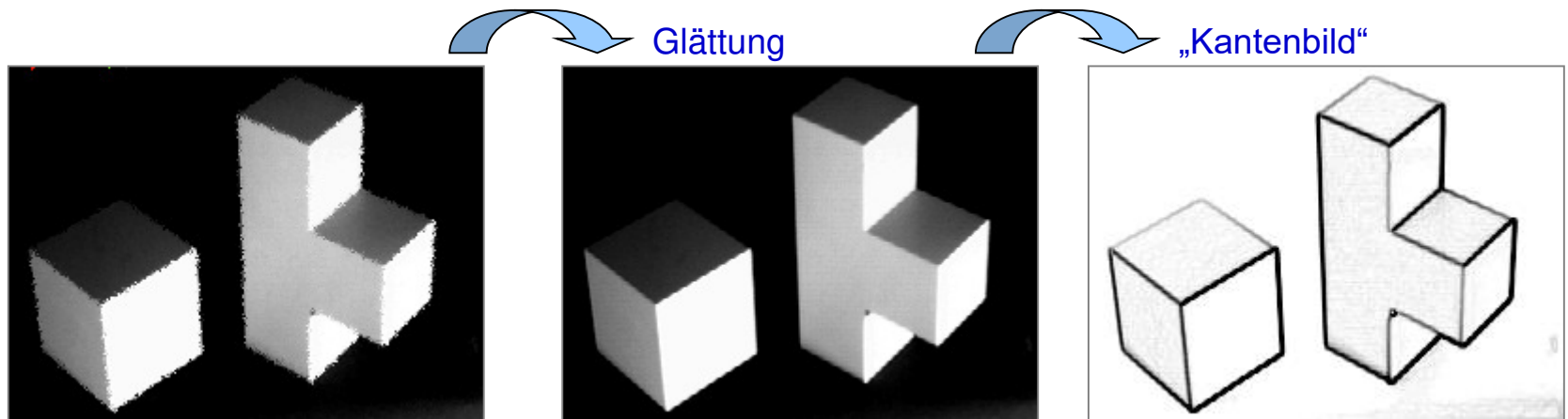
Wir wollen Prozesse der Low-Level-Vision an zwei wichtigen Beispielen darlegen:

1) **Verrauschte** Bilder zeigen Störungen, die dem eigentlichen Bildsignal überlagert sind und den Interpretationsprozess stören können

→ **Glättungsfilter** beseitigen dieses Rauschen

2) Bildsegmente, die Objekten oder Objektteilen entsprechen, zeigen Ähnlichkeiten bzgl. bestimmter Eigenschaften (z.B. Farbe, Intensität, Textur, etc.)

→ **Kantenfilter** suchen Pixel, die einen signifikanten Wechsel bzgl. der Intensität aufweisen und damit zur Abgrenzung von Bildsegmenten dienen



Low-Level-Vision: Konvolution (1)

Glättung und Kantenfindung sind als sog. **Filter** realisierbar. Grundlage solcher Filter ist die sog. **Konvolution** oder **Faltung** von zwei Funktionen.

Die Konvolution ist beschreibbar durch

$$(f * g)(x,y) = \sum_{u \in D} \sum_{v \in D} f(u,v) \cdot g(x-u,y-v).$$

D ist eine Menge von ganzzahligen Verschiebungsvektoren (u,v) so, dass die Positionen $(x-u,y-v)$ i.A. in lokaler Nachbarschaft der zentralen Position (x,y) sind.

Der Funktionswert $(f * g)(x,y)$ in Position (x,y) ist dann die gewichtete Summe der $g(x-u,y-v)$ mit einer nach (x,y) verschobenen Gewichtsfunktion $f(u,v)$.

Die Funktion f heißt **Konvolutionsfunktion** oder **Faltungsfunktion**. Die **Konvolution** oder **Faltung** wird durch das Operatorsymbol „ $*$ “ dargestellt.

Low-Level-Vision: Konvolution (2)

Beispiel für 1-dimensionale Konvolutionen

$$(f * g)(x) = \sum_{u \in D} f(u) \cdot g(x-u).$$

- $g(x)$ beschreibe Werte der Grauwertpixel einer Bildzeile

- $f_1(u)$ sei definiert für $D = \{-1, 0, +1\}$ mit $f_1(-1) = f_1(0) = f_1(+1) = 1/3$

1/3	1/3	1/3
-----	-----	-----

$$\rightarrow (f_1 * g)(x) = f_1(-1) \cdot g(x+1) + f_1(0) \cdot g(x) + f_1(+1) \cdot g(x-1) = [g(x+1) + g(x) + g(x-1)]/3$$

$\rightarrow f_1(u)$ bildet den Mittelwert über Zeilenpixel und deren zwei Nachbarpixel

- $f_2(u)$ sei definiert für $D = \{-1, 0, +1\}$ mit $f_2(-1) = 1$, $f_2(0) = 0$, $f_2(+1) = -1$

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

$$\rightarrow (f_2 * g)(x) = f_2(-1) \cdot g(x+1) + f_2(0) \cdot g(x) + f_2(+1) \cdot g(x-1) = g(x+1) - g(x-1)$$

$\rightarrow f_2(u)$ rechnet die gerichtete Differenz der zwei Nachbarpixel eines Zeilenpixels

○

Low-Level-Vision: Konvolution (3)

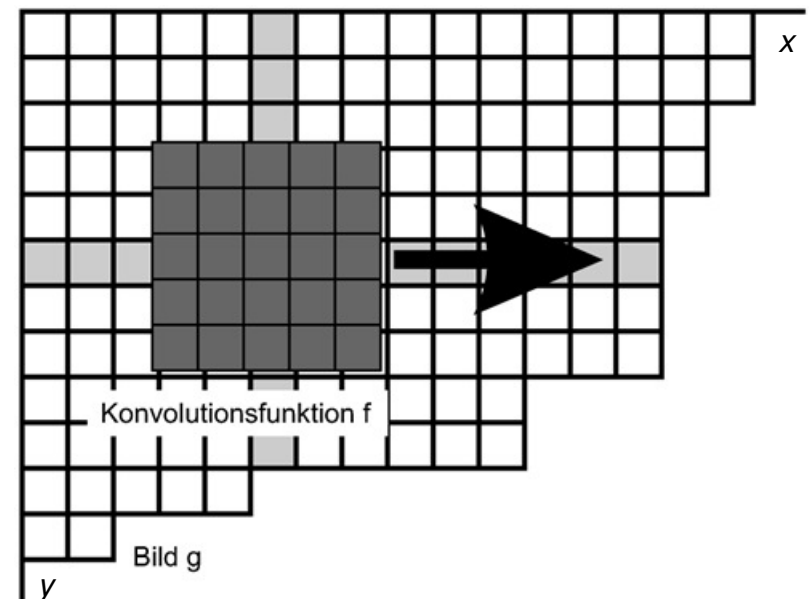
Die 2-dimensionale Konvolution

$$(f * g)(x, y) = \sum_{u \in D} \sum_{v \in D} f(u, v) \cdot g(x - u, y - v),$$

ist auf die Pixelmatrix eines Grauwertbildes übertragbar. Die Funktion $g(x, y)$ steht dann für die Grau- bzw. Intensitätswerte der Bildmatrix.

Die Konvolution der Grauwertfunktion $g(x, y)$ mit einer Konvolutionsfunktion f für Pixel (x, y) ist die mit den Werten der zentral auf (x, y) verschobenen Konvolutionsfunktion f gewichtete Summe der Werte von (x, y) und seinen Nachbarpixeln gemäß der Verschiebungsvektoren aus D .

Für $u, v \in D = \{-2, -1, 0, 1, 2\}$ ergibt sich ein 5×5 -Nachbarschaftsfeld, innerhalb dessen die Verrechnung erfolgt.



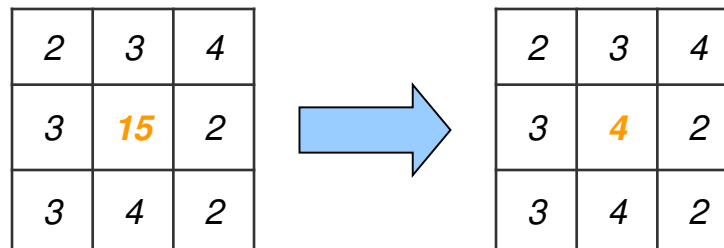
Low-Level-Vision: Bewegter Mittelwert (1)

Einfaches Beispiel für 2-dimensionale Konvolution:

Das **Mittelwertfilter** berechnet für alle Pixel eines Bildes den Mittelwert aus deren 3×3-Nachbarfeldern. Also $u, v \in \mathbf{D} = \{-1, 0, 1\}$.

Das Abspeichern der so ermittelten neuen Pixelwerte als neues Bild resultieren in einem geglätteten Bild.

I.A. werden die durch Filter manipulierten Bilder wieder als Bilder abgespeichert. Daher werden die Ergebnisse oft auf ganzzahlige Grauwerte gerundet:

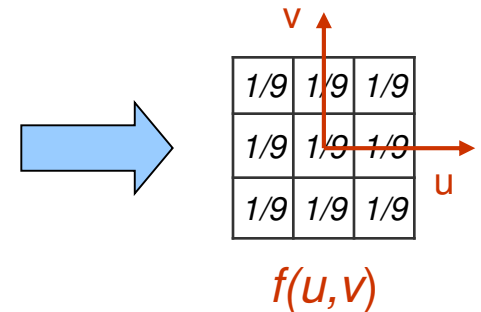


$$(2+3+4+3+15+2+3+4+2)/9 = 38/9 \approx 4.$$

Low-Level-Vision: Bewegter Mittelwert (2)

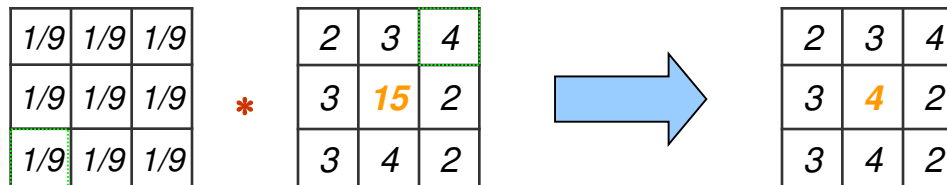
Die diskrete Konvolutionsfunktion $f(u,v)$ bestimmt also die Gewichtung aller Pixel aus dem 3×3 -Nachbarschaftsfeld eines Pixels (x,y) , die in die Berechnung des Mittelwertes eingehen:

$$f(u,v) = \{ (-1, 1) \rightarrow 1/9, (0, 1) \rightarrow 1/9, (1, 1) \rightarrow 1/9, \\ (-1, 0) \rightarrow 1/9, (0, 0) \rightarrow 1/9, (1, 0) \rightarrow 1/9, \\ (-1, -1) \rightarrow 1/9, (0, -1) \rightarrow 1/9, (1, -1) \rightarrow 1/9 \}$$



Die Mittelwertberechnung lässt sich nun als Konvolution schreiben:

$$h(x,y) = \sum_u \sum_v f(u,v) \cdot g(x-u,y-v) \text{ mit } u,v = -1,0,1.$$



Hervorgehoben die korrespondierenden Pixel von Filter und Bildausschnitt für $u=v=-1$

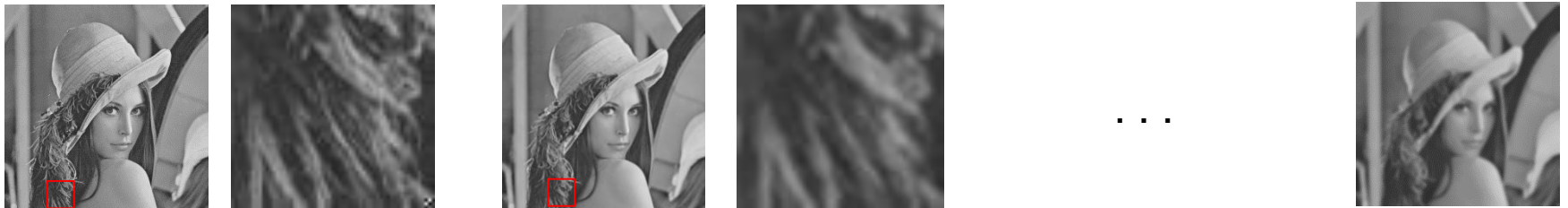
Low-Level-Vision: Bewegter Mittelwert (3)

Das **Mittelwertfilter** ist – wie viele Filter – bzgl. der Größe des Nachbarschaftsfeldes durch den Parameter m der Filtergröße generalisierbar:

$$h(x,y) = \sum_u \sum_v f(u,v) \cdot g(x-u,y-v) \text{ mit } u,v = -(m-1)/2, \dots, (m-1)/2.$$

Das bisherige Mittelwertfilter hatte die Filtergröße $m = 3$.

Das Mittelwertfilter ist auch in weiteren Größen wie 5, 7, ... implementierbar. Mit zunehmender Größe „verwäscht“ das Filter dann jedoch auch die eigentlichen Bildinhalte stärker.



Allg. werden die Matrizen der Filter auch als Konvolutionskerne, als Faltungskerne*, (Filter-)Kern und als (Filter-)Masken bezeichnet.

* Faltung als deutsche Bezeichnung für Konvolution.

Low-Level-Vision: Zur Implementierung der Konvolution

Die Anwendung einer Konvolutionsfunktion $f(u,v)$ führt für alle Pixel (x,y) mit Grau- bzw. Intensitätswerten $g(x,y)$ zu neuen Werten $h(x,y)$:

$$h(x,y) = \sum_u \sum_v f(u,v) \cdot g(x-u,y-v) \text{ mit } u,v = -(m-1)/2, \dots, (m-1)/2.$$

Für die Implementierung werden die Grauwerte $g(x,y)$ aus einem Eingabebild $G[x,y]$ gelesen und die errechneten neuen Werte $h(x,y)$ in eine Ausgabebild $H[x,y]$ geschrieben.

Für das Mittelwertfilter enthält $H[x,y]$ also die lokal gemittelten Intensitätswerte der Pixel aus dem Eingabebild $G[x,y]$.

Low-Level-Vision: Konvolution & Korrelation

Für viele Filter lässt sich die Konvolution durch die Korrelation mit einer punktgespiegelten Korrelationmaske ersetzen.

Die Konvolution

$$(f * g)(x, y) = \sum_u \sum_v f(u, v) \cdot g(x - u, y - v) \text{ mit } u, v = -(m-1)/2, \dots, (m-1)/2$$

entspricht dann der Korrelation

$$(f \oplus g)(x, y) = \sum_u \sum_v f(u, v) \cdot g(x + u, y + v) \text{ mit } u, v = -(m-1)/2, \dots, (m-1)/2.$$

Die Zuordnung der Gewichte der Filterkerne zu den Bildpixeln ist jetzt einfacher zu lesen im Gegensatz zur punktgespiegelten Zuordnung bei der Konvolution.

Low-Level-Vision: Gauß-Filter (1)

Das Mittelwertfilter ist das einfachste Glättungsfilter. Bei vielen Glättungen soll der Einfluss der Nachbarpixel mit wachsendem Abstand abnehmen.

Gehen wir zudem von zufällig verteiltem Gaußschem Rauschen aus, so stellt das sog. Gauß-Filter ein geeignetes Filter dar. Als Korrelation geschrieben:

$$\begin{aligned} h_g(x, y) &= \sum_u \sum_v f_{G,\sigma}(u, v) \cdot g(x + u, y + v) \\ &= \sum_u \sum_v \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/2\sigma^2} \cdot g(x + u, y + v). \end{aligned}$$

$\sigma = 1$ ist für das Glätten geringen Rauschens ausreichend. $\sigma = 2$ glättet stärkeres Rauschen, führt aber auch zu stärkerem Detailverlust. Das Gauß-Filter für $\sigma = 1$ und $m = 3$:

0,075	0,124	0,075
0,124	0,204	0,124
0,075	0,124	0,075

Low-Level-Vision: Gauß-Filter (2)

Die Größe des Konvolutionskerns eines Gauß-Filters

- ist so zu wählen, dass die Werte der Gauß-Funktion hinreichend gut approximiert werden,
- ist entsprechend von der Standardabweichung σ der Gauß-Funktion abhängig.

Für eine Filtergröße $m = 2 \cdot \lceil 3\sigma \rceil + 1$ gilt:

- der Funktionswert am Rand des Filterkerns beträgt noch 1% des Maximums der Gauß-Funktion,
- das Maximum in $f_{G,\sigma}(0,0)$ ist $(2\pi\sigma^2)^{-1}$.

Low-Level-Vision: Gauß-Filter (3)

Für alle Glättungfilter muss sich die Summe der Gewichtselemente des Konvolutionskerns zu 1 aufsummieren.

Der Grund: jede von 1 verschiedene Gewichtssumme würde eine Skalierung der Intensitäten der Bildfunktion ergeben.

Wird das Gauß-Filter zunächst aus der analogen Gauß-Funktion erzeugt nach

$$f_{G,\sigma}(u,v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/2\sigma^2},$$

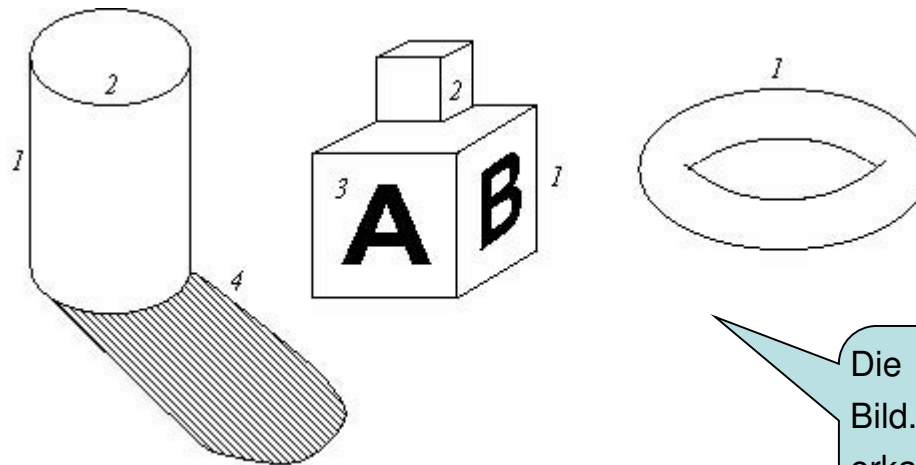
so ist dieser noch zu diskretisieren. Durch die Diskretisierung und den damit verbundenen Quantisierungseffekt wird die Summe nicht mehr 1 ergeben.

Entsprechend muss die Gauß-Funktion durch einen Normierungsfaktor (gleich der Summe aller resultierenden Gewichte) entsprechend angepasst werden.

Low-Level-Vision: Kantenfilter (1)

Als Kanten (Konturen) werden verstanden: gerade Linien oder Kurven, entlang derer eine „wesentliche“ Änderung der Bildhelligkeit auftritt.

Motivation: Kantenkonturen im Bild entsprechen Szenenkonturen, die für Objekterkennung und Szeneninterpretation relevant sind.



Die Kantenerkennung erfolgt im Bild. Die Klassifikation der erkannten Kanten (s. Bild) erfolgt in späteren Interpretationsphasen.

Typen von Szenenkonturen:

- 1) Tiefendiskontinuitäten
- 2) Diskontinuitäten in der Oberflächenorientierung
- 3) Reflexdiskontinuitäten
- 4) Beleuchtungsdiskontinuitäten (Schatten)

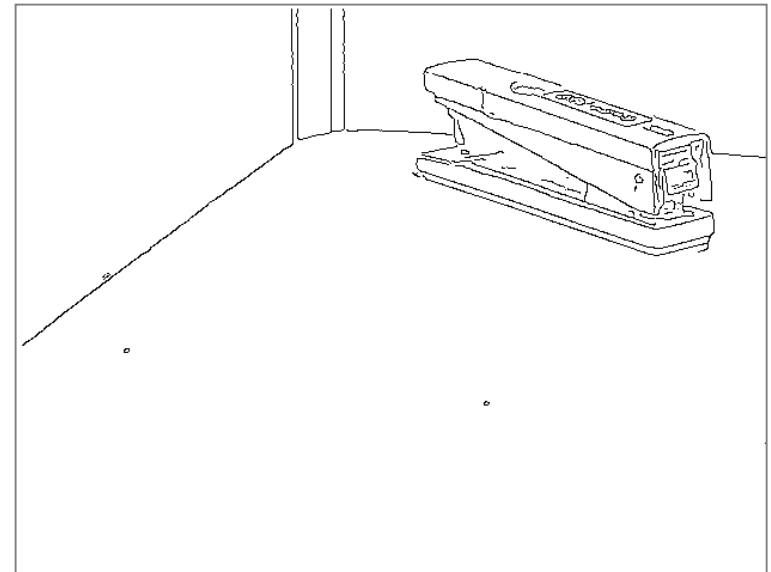
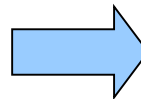
Low-Level-Vision: Kantenfilter (2)

Die Ergebnisse von Kantenfiltern sind i.A. nicht perfekt.

- Gründe sind u. a. mangelnde Kontraste, Rauschen, Verschattungen etc.
- Auswirkungen sind
 - Lücken in den Kantendetektionen (falsch negative Kantenhypotesen)
 - Kanten durch Rauschen, wo keine Szenenkanten sind (falsch positive Hyp.)
 - nicht korrekt ausgerichtete Kanten (Unschärfen)



Grauwertbild



Berechnete Kanten

Low-Level-Vision: Kantenfilter (3)

Welche Idee liegt Kantenfiltern zugrunde?

- Kanten sollen sein: gerade Linien oder Kurven, entlang derer eine „wesentliche“ Änderung der Bildhelligkeit auftritt
- Betrachten wir zunächst ein idealisiertes (Grauwert-)Bild als reellwertige **Intensitätsfunktion**, die jeder Bildkoordinate einen Intensitätswert zuordnet:

$$I: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}, (x,y) \rightarrow I(x,y), I_{\min} \leq I(x,y) \leq I_{\max} .$$

- „Wesentliche“ Änderungen in einer Funktion bedeuten dann hohe Werte in der ersten Ableitung!

Low-Level-Vision: Kantenfilter (4)

Welche Idee liegt Kantenfiltern zugrunde?

- Die Idee ist also:

- differenziere die Bildfunktion $I(x,y)$,

- suche nach hohen Werten der Ableitung

$$\nabla I(x, y) = \begin{pmatrix} \frac{\partial I(x, y)}{\partial x} \\ \frac{\partial I(x, y)}{\partial y} \end{pmatrix}.$$

- Praktisch erfolgt die Umsetzung für die *diskrete* Intensitätsfunktion von Bildern mit diskreten Koordinaten- und quantisierten Intensitätswerten $x \in \{0, 1, \dots, L-1\}$, $y \in \{0, 1, \dots, R-1\}$ und $I(x,y) \in \{I_{min}, I_{min}+1, \dots, I_{max}\}$ über eine Approximation der Ableitung $\nabla I(x,y)$ durch Differenzenquotienten.

Low-Level-Vision: Kantenfilter (5)

Welche Idee liegt Kantenfiltern zugrunde?

- Eine einfache Approximation der Ableitung $\nabla I(x, y) = \begin{pmatrix} \frac{\partial I(x, y)}{\partial x} \\ \frac{\partial I(x, y)}{\partial y} \end{pmatrix}$

sind $\frac{\partial I(x, y)}{\partial x} \approx (-1 \ 0 \ 1) * I(x, y)$

und $\frac{\partial I(x, y)}{\partial y} \approx \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} * I(x, y).$

Low-Level-Vision: Sobel-Operator (1)

Um a priori einer hohen Sensitivität der Kanten hervorhebung gegenüber lokalem Rauschen vorzubeugen, werden die beiden eindim. Kantenfilter zur Gradientenapproximation jeweils mit einem eindim. Glättungsfiler kombiniert.

Der Glättungsfiler ist eine ganzzahlige Approximation eines eindim. Gaußfilters. Genauer wird dieser als Binomialfilter 1. Ordnung bezeichnet.

$$\begin{aligned} \textbf{Horizontales Sobel - Filter } S_x : \quad & \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & +1 \end{pmatrix} \\ \\ \textbf{Vertikales Sobel - Filter } S_y : \quad & \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} = \begin{pmatrix} +1 \\ 0 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \end{aligned}$$

Low-Level-Vision: Sobel-Operator (1)

Die Anwendung der beiden Sobel-Filter durch die **Konvolution** zeigt die Korrespondenz zwischen den Koeffizienten der Filterkerne einerseits und den Nachbarpixeln des untersuchten Pixels andererseits:

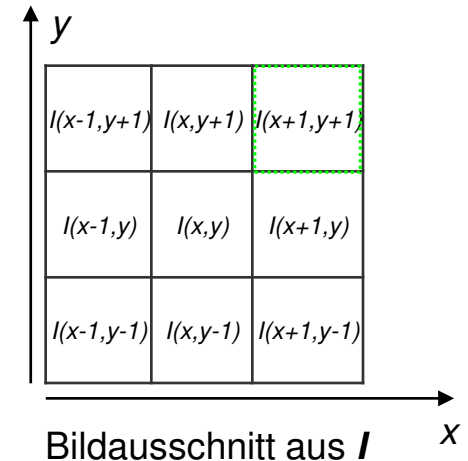
$$(f * g)(x,y) = \sum_u \sum_v f(u,v) \cdot g(x-u,y-v) \text{ mit } u,v = -(m-1)/2, \dots, (m-1)/2.$$

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

Horizontales Sobel-Filter \mathbf{S}_x

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

Vertikales Sobel-Filter \mathbf{S}_y



$$I'_x(x,y) \leftarrow \mathbf{S}_x * I = I(x-1,y-1) + 2 \cdot I(x-1,y) + I(x-1,y+1) - I(x+1,y-1) - 2 \cdot I(x+1,y) - I(x+1,y+1)$$

$$I'_y(x,y) \leftarrow \mathbf{S}_y * I = I(x-1,y-1) + 2 \cdot I(x,y-1) + I(x+1,y-1) - I(x-1,y+1) - 2 \cdot I(x,y+1) - I(x+1,y+1).$$

Low-Level-Vision: Sobel-Operator (2)

Aus den Approximationen der Ableitungen der Intensitätsfunktion in x-Richtung und y-Richtung werden dann Betrag und Richtung des Gesamtgradienten berechnet. Mit

$$\nabla I(x, y) = \text{grad}(I) = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} \frac{\partial I(x, y)}{\partial x} \\ \frac{\partial I(x, y)}{\partial y} \end{pmatrix}.$$

- richtungsunabhängiger **Gradientenbetrag***:

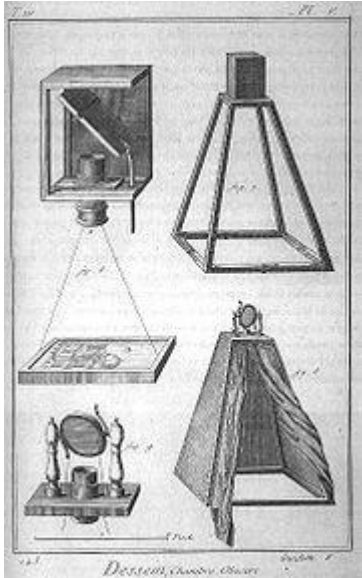
$$|\nabla I(x, y)| = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}.$$

- **Gradientenrichtung**:

$$\Theta = \begin{cases} \arctan(g_y(x, y) / g_x(x, y)) & \text{für } g_x(x, y) \neq 0, \\ 0^\circ & \text{für } g_x(x, y) \neq 0, g_y(x, y) = 0, \\ 90^\circ & \text{für } g_x(x, y) = 0, g_y(x, y) \neq 0. \end{cases}$$

Low-Level-Vision: Sobel-Operator (3)

Beispiel für Anwendung des Sobel-Operators:



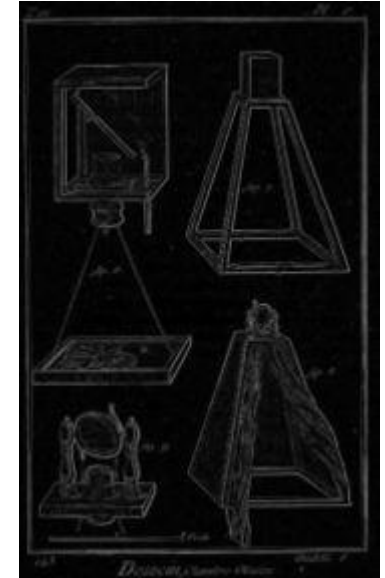
Originalbild
"Camera Obscura"



mit vertikalem
Sobel S_y gefaltet



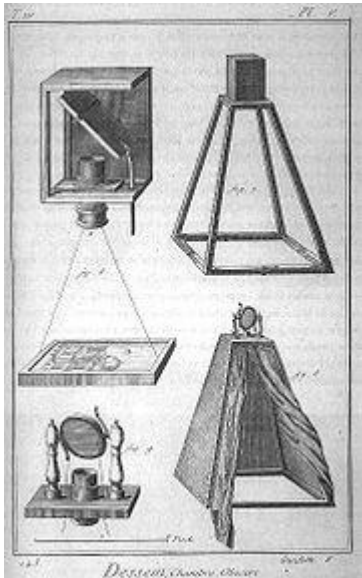
mit horizontalem
Sobel S_x gefaltet



Beide Faltungen
kombiniert

Low-Level-Vision: Skalierung der Intensitätswerte

Generell können Berechnungen auf Pixeln zu neuen Intensitätswerten führen, die nicht mehr durch das vorgegebene Intensitätsspektrum abgedeckt werden.



Originalbild
"Camera Obscura"



mit vertikalem
Sobel S_x gefaltet



mit horizontalem
Sobel S_y gefaltet



Beide Faltungen
kombiniert

So führen die horizontalen und vertikalen Sobel-Filter auch zu negativen Werten. Der Gradientenbetrag der kombinierten Sobel-Filter S_x und S_y kann zu Werten oberhalb der maximal darstellbaren Intensität führen.

Low-Level-Vision: Bildverbesserung

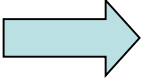
Bevor Bilder mit Konvolutionsfiltern bearbeitet werden, sollten noch grundlegende Verfahren der Bildverbesserung angewendet werden.

Die folgende Grauwertspreizung ist aber auch nutzbar, um Intensitätswerte nach Anwendung des Sobel-Operators wieder in das Intensitätsspektrum zurück zu führen.

Diese basieren i.W. darauf, den Kontrast in kontrastarmen Bildern zu verstärken und über- bzw. unterbelichtete Bilder abzudunkeln bzw. aufzuhellen.

Z.B. nutzen Satellitenbilder den möglichen Dynamikbereich von 256 Grauwerten oft nur unvollständig: die Sensoren sind so ausgelegt, dass sowohl sehr helle (z.B. Schnee) als auch dunkle Flächen in Messwerte umgesetzt werden können.




Kontrast-
verstärkung



Bildquelle: Abteilung Fern-
erkundung der Univ.Trier:
Kursbegleitung *Digitale Bild-
bearbeitung*

Low-Level-Vision: lineare Transformationsfunktionen

Generell werden zur Bildverbesserung Transformationsfunktionen $T(I)$ benutzt, die die Intensitätswerte I eines Eingabebildes auf neue Intensitätswerte $T(I)$ des verbesserten Ausgabebildes abbilden.

Eine **lineare Transformationsfunktion** hat die Form

$$T(I) = (I + c_1) \cdot c_2 .$$

Entsprechend der Parameterbelegung folgt:

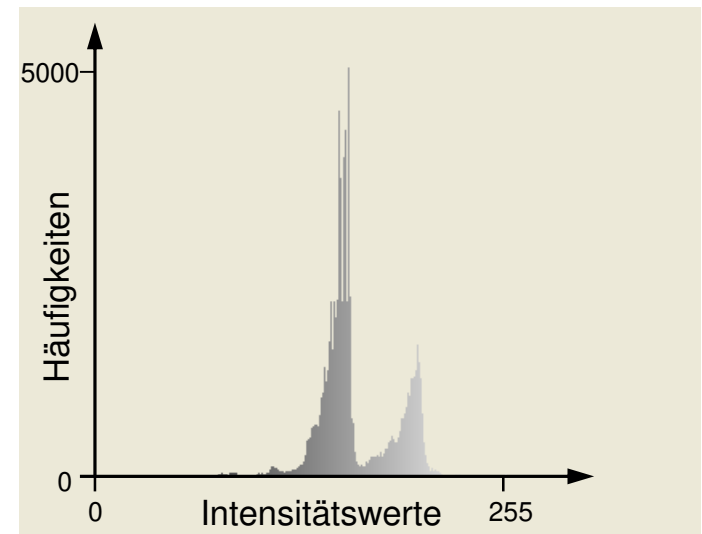
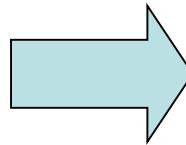
- die identische Abbildung für $c_1 = 0$, $c_2 = 1$,
- Aufhellung für $c_1 > 0$,
- Abdunklung für $c_1 < 0$,
- Kontraststeigerung für $c_2 > 1$
- Kontrastminderung für $c_2 < 1$.

Low-Level-Vision: Intensitätshistogramm

Die Funktionsweise der lineare Transformation $T(I) = (I + c_1) \cdot c_2$ lässt sich gut über das Histogramm eines Bildes bzw. eines Kanals darstellen.

Das **Intensitätshistogramm eines Bildes $I[x,y]$** mit Intensitätsspektrum $\{0, \dots, I_{\max}\}$ ist eine diskrete Funktion, die jedem Intensitätswert I des Spektrums die Anzahl n_I der Pixel im vorliegenden Bild zuweist, die diesen Wert aufweisen.

Das normalisierte Histogramm skaliert die Einträge für jeden Intensitätswert zu $n_I / (\text{Gesamtzahl der Bildpixel})$.

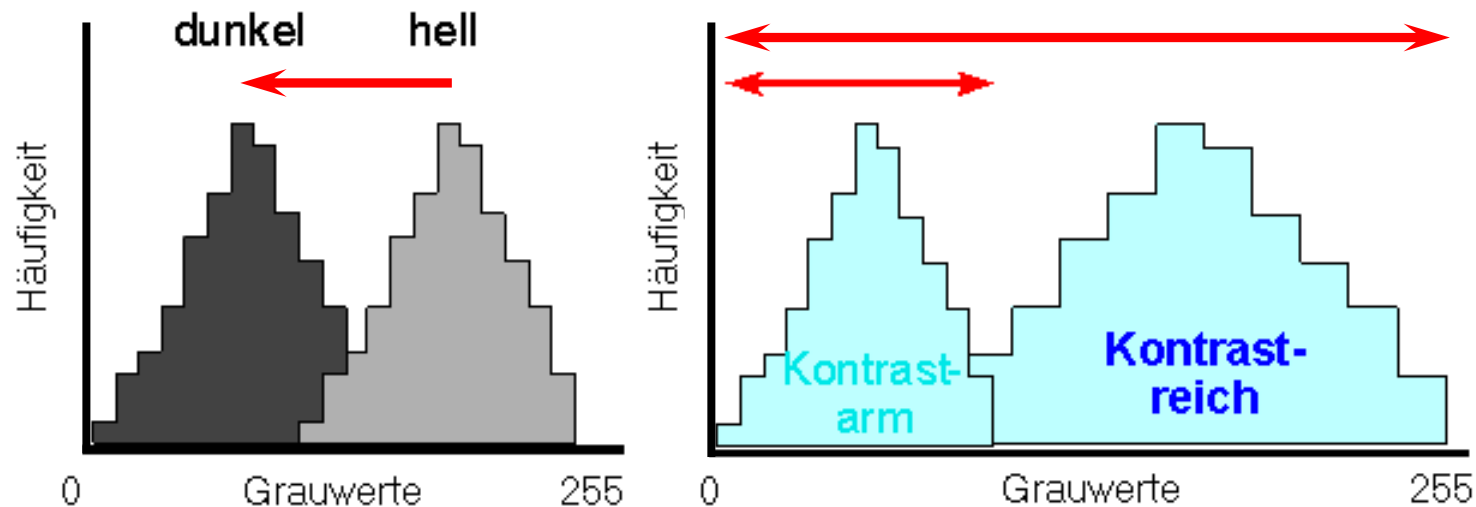


Low-Level-Vision: Lineare Grauwertspreizung (1)

Die lineare Grauwertspreizung basiert auf der linearen Transformation

$$T(I) = (I + c_1) \cdot c_2$$

und bewirkt bei einem zu hellen und kontrastarmen Bild zunächst das Verschieben der belegten hohen Intensitäten durch negatives c_1 und die anschließende Kontrasterhöhung durch Streckung um c_2 :



Low-Level-Vision: Lineare Grauwertspreizung (2)

Der globale Kontrast C_{global} eines Einkanalbildes kann durch die lineare Grauwert- bzw. Intensitätsspreizung optimiert werden, bei der die lineare Transformationsfunktion $T(I) = (I + c_1) \cdot c_2$ wie folgt parametrisiert wird:

- $c_1 = -I_{\text{minGiven}}$,
 - $c_2 = I_{\text{max}} / (I_{\text{maxGiven}} - I_{\text{minGiven}})$,
- $\leadsto T(I) = [I - I_{\text{minGiven}}] \cdot [I_{\text{max}} / (I_{\text{maxGiven}} - I_{\text{minGiven}})]$,

mit maximalen und minimalen Intensitätswerten I_{maxGiven} , I_{minGiven} im gegebenen Bild $I[x,y]$ und maximal darstellbarem Intensitätswert I_{max}^* .

Es erfolgt also i.A. eine Abdunklung ($c_1 \leq 0$) und eine Kontrastverstärkung ($c_2 \geq 1$), da $I_{\text{minGiven}} \geq 0$ und somit $I_{\text{max}} \geq I_{\text{maxGiven}} - I_{\text{minGiven}}$.



Bildquelle: Abteilung Fernerkundung der Univ.Trier: Kursbegleitung *Digitale Bildbearbeitung*

* Z.B. $I_{\text{max}} = 255$ für 1-Byte-Grauwertbilder.

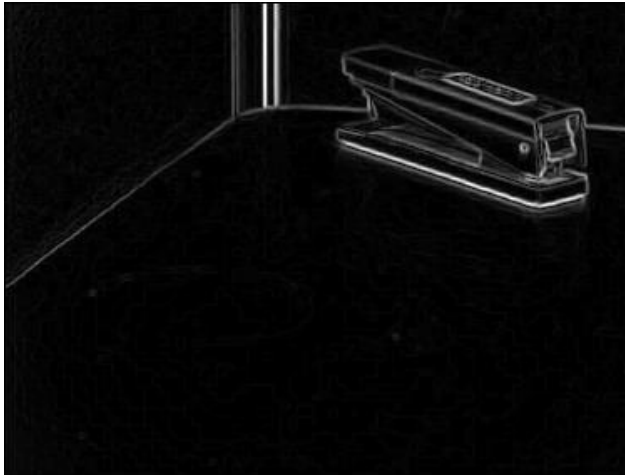
Low-Level-Vision: Binarisierung

Binarisierung bezeichnet eine binäre Klassifikation von Pixeln $I(x,y)$ nach einem sog. Schwellwert (threshold) t_B :

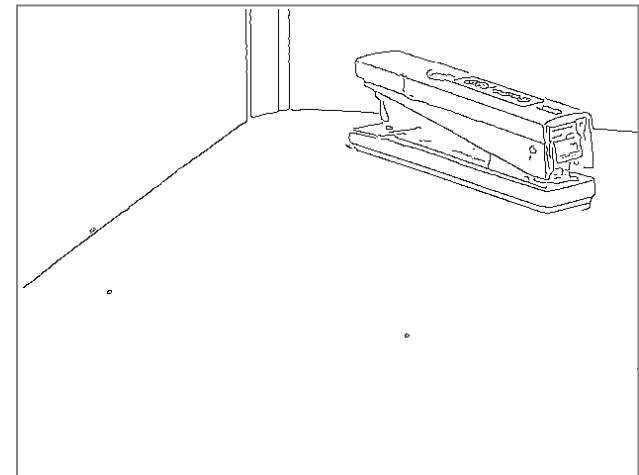
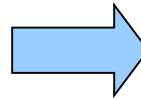
$$I(x,y) \in \begin{cases} \text{Klasse}_1 : I(x,y) \leq t_B, \\ \text{Klasse}_2 : I(x,y) > t_B. \end{cases}$$

Bei der Analyse der Bilder von Textdokumenten kann so z.B. schwarze Schrift von weißem Hintergrund getrennt werden.

Nach der Anwendung eines Sobel-Filters können so alle „Kantenpixel“ selektiert werden, die einen *Gradientenbetrag* oberhalb eines Schwellwertes t_B haben.



Ergebnis des Sobel-Operators



Binarisierung \rightarrow Kantenpixeln

Low-Level-Vision: Canny-Operator (1)

Für die Kantenerkennung kombiniert der Canny-Algorithmus vier Prozesse:

- Glättung durch Gauß-Filter
- Kantenerkennung durch Sobel-Operator
- Verdünnung der Kanten durch Unterdrückung von „Non-Maxima“
- Binarisierung



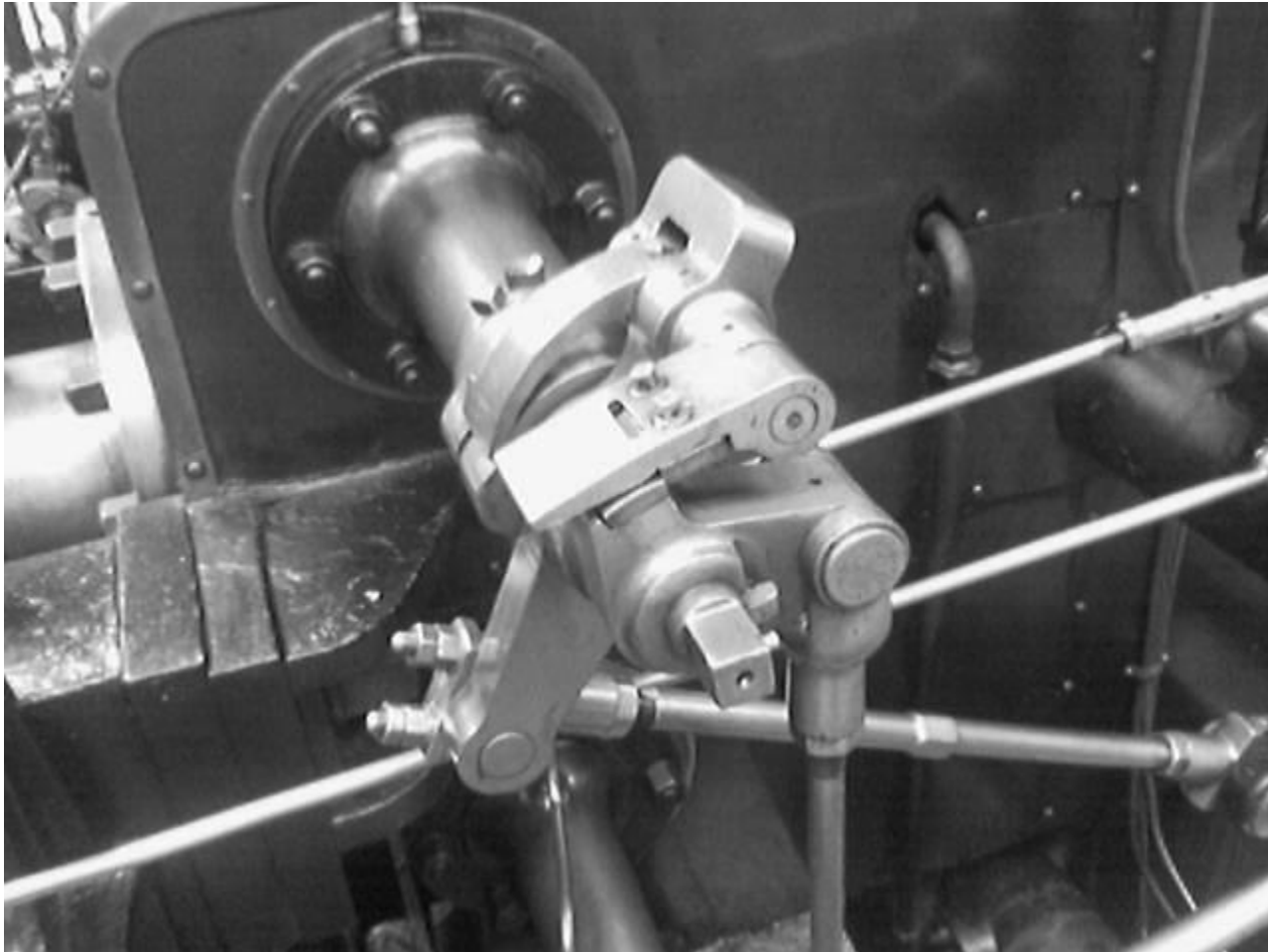
Low-Level-Vision: Canny-Operator (2)

Der Canny-Algorithmus mit drei Parametern σ , T_1 und T_2 :

1. **Glättung** durch **Gauß-Filter** mit Standardabweichung σ .
2. **Kantenerkennung** mit **Sobel-Operator**:
 - 2.1 Gradientenbetrag: $|\nabla I(x,y)| = (g_x(x,y)^2 + g_y(x,y)^2)^{1/2}$,
 - 2.2 Gradientenrichtung: $\Theta = \arctan(g_y(x,y) / g_x(x,y))$
gerundet zu 0° , 45° , 90° und 135° .
3. **Non-Maxima-Unterdrückung**: alle Pixel, die in Gradientenrichtung Θ nicht ein lokales Maximum bilden, werden auf Null gesetzt.
4. **Binarisierung**: Um das Aufbrechen von Kanten durch variierende Kantenstärke zu vermeiden, werden zwei Schwellwerte T_1 und T_2 mit $T_1 < T_2$ verwendet. Man scannt das Bild bis zu einem Pixel mit Gradientenbetrag $> T_2$. Ausgehend von diesem Pixel werden schrittweise alle benachbarten Pixel entlang der Kante mit Gradientenbetrag $> T_1$ als Kantenelement markiert.

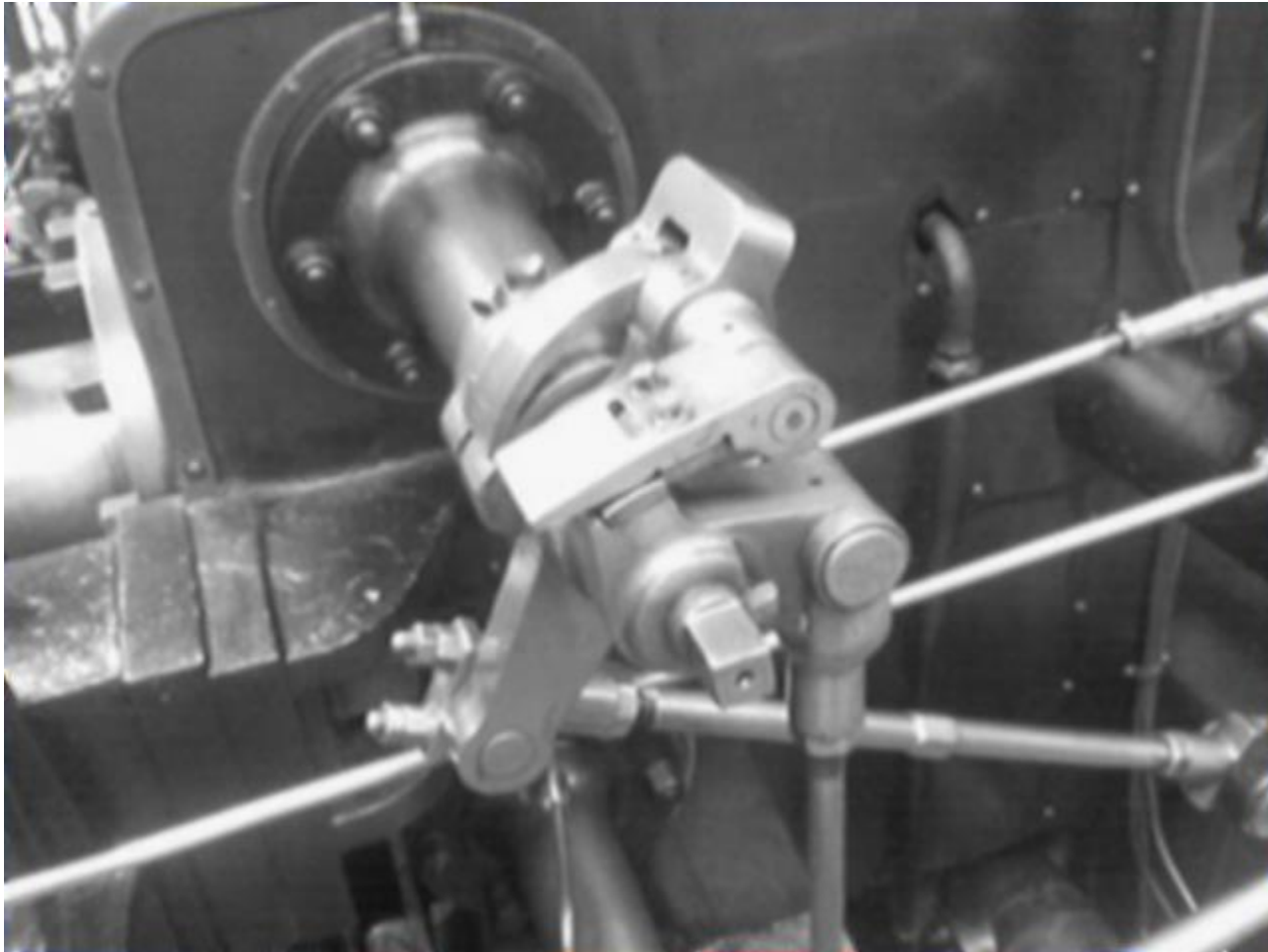
Low-Level-Vision: Canny-Operator (3.1)

Ausgangsbild für Canny-Operator:



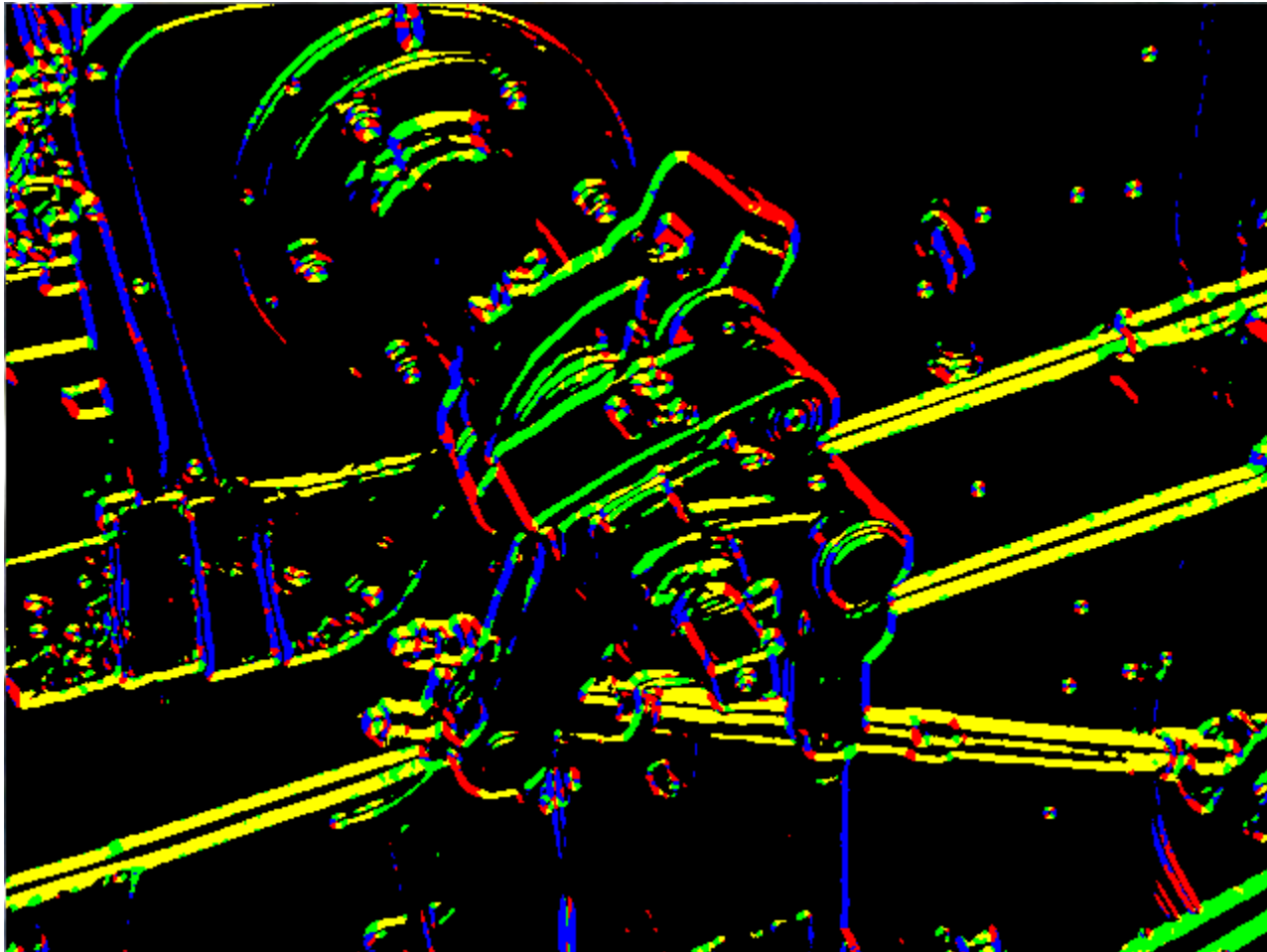
Low-Level-Vision: Canny-Operator (3.2)

Nach Anwendung von 5x5-Gauß-Filter mit $\sigma = 1,4$:



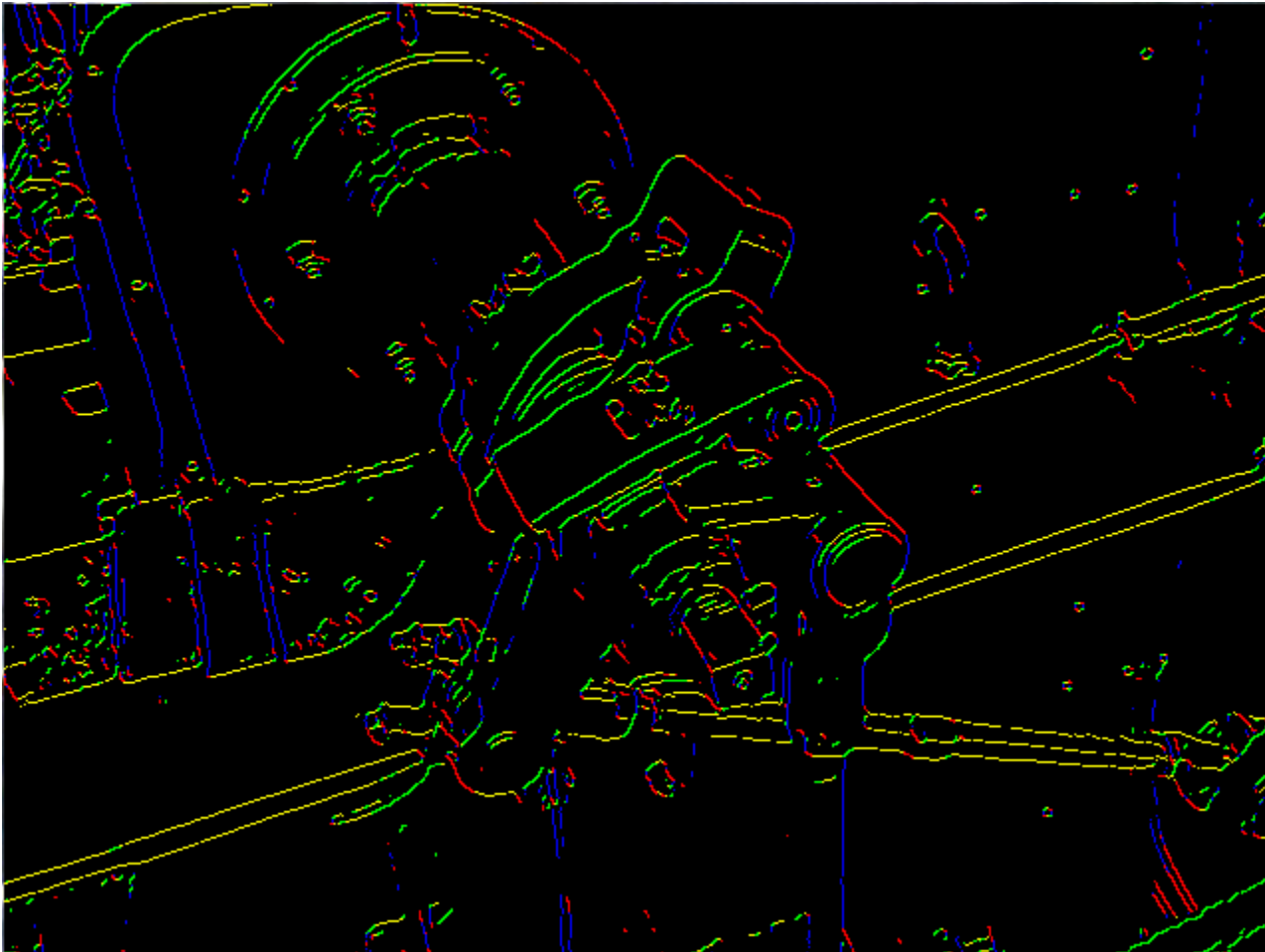
Low-Level-Vision: Canny-Operator (3.3)

Nach Anwendung von Sobel-Filter mit Farbkodierung der Klassifikation nach **Kantenrichtungen**: gelb = 0° , grün = 45° , blau = 90° , rot = 135°



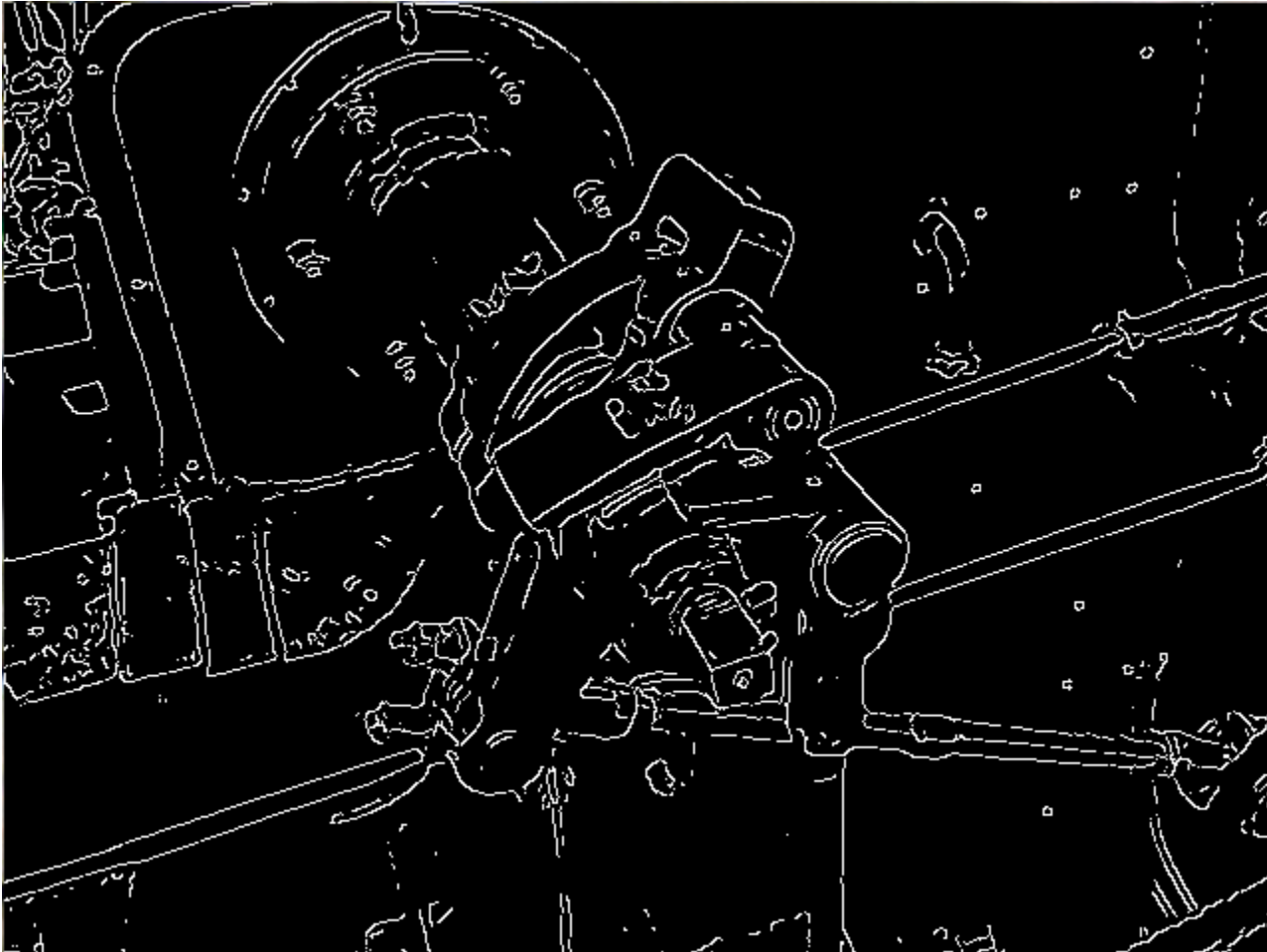
Low-Level-Vision: Canny-Operator (3.4)

Nach Anwendung der Non-Maximum-Unterdrückung in **Gradienten**richtung:



Low-Level-Vision: Canny-Operator (3.5)

Nach Binarisierung:



Zusammenfassung

- Das Verstehen von Bildern beschreibt generell ein unterbestimmtes inverses Problem, das u.a. Interpretationsmehrdeutigkeiten auflösen muss.
- Man unterscheidet Low-, Mid- und High-Level-Vision als Phasen des Bildverstehens.
- Grundlegende Strategieparadigmen des Bildverstehens sind der datengetriebene Ansatz (Bottom-Up), der modellgetriebene Ansatz (Top-Down) sowie der heterarchische Ansatz als Mischform.
- In der Early Vision stellt die lokale Faltung (Konvolution) eine grundlegende Operation dar, mit deren Hilfe sowohl Glättungsoperationen als auch Kantenerkennung umsetzbar sind.