Karlsruhe Institute of Technology

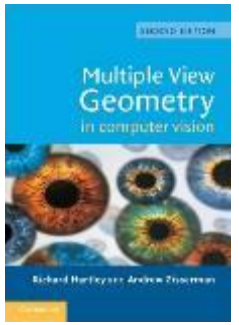
Robotik I – Bildverarbeitung

Tamim Asfour, Rüdiger Dillmann

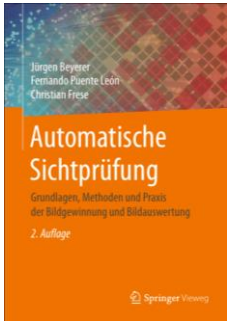
KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H^2T)

Literatur

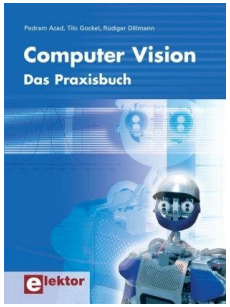
- Multiple view geometry in computer vision
  - R. Hartley und A. Zisserman
  - ISBN 978-0-521-54051-3



- Automatische Sichtprüfung
  - J. Beyerer, F. Puente León und C. Frese
  - ISBN 978-3-662-47785-4



- Computer Vision – Das Praxisbuch
  - Pedram Azad, Tilo Gockel und Rüdiger Dillmann
  - ISBN 978-3-89576-165-2

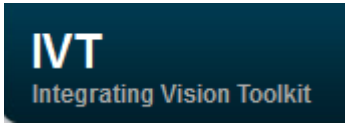


Programmbibliotheken

- OpenCV
  - http://opencv.org
  - Facedetection, Optical Flow, GPU Computing

- Point Cloud library (PCL)
  - http://pointclouds.org
  - Pointcloud processing, RANSAC primitive fitting, ICP

- Integrating Vision Toolkit (IVT)
  - http://ivt.sourceforge.net
  - Firewire, image formats, visualization, image processing



Motivation

- Das Sehvermögen ermöglicht die Wahrnehmung der Umwelt

- Nutzbarkeit in einem technischen System
  - Visuellen Informationen müssen aufgenommen werden
    - gute Qualität
    - digitales Format
  - Relevante Informationen müssen aus den Daten extrahiert werden

- **Bilderfassung:** Hardware
- **Bildverarbeitung:** überwiegend Software

Lena-Bild

Das ursprüngliche Lena-Bild stammt aus der US-amerikanischen November-Ausgabe des Playboy-Magazins des Jahres 1972.

Es zeigt das schwedische Playmate **Lena Söderberg** (vom Playboy „Lenna Sjööblom“ genannt).

https://de.wikipedia.org/wiki/Lena_(Testbild)

Kamerabeispiele

PointGrey Flea3 USB3

CMU MultiSense S7

Karlsruhe Humanoid Head

Intel RealSense R200

roboception (Prototyp)

MultiSense S7 and S7S

https://www.youtube.com/watch?v=75A8iFeVgh8

Bildrepräsentation

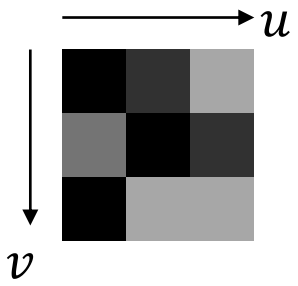
- Bilder müssen im Computer/Roboter repräsentiert werden
- Ein Bild ist ein 2D Gitter von diskreten Punkten (**Pixel**)
- **Bildkoordinaten** (hier):
  - *u* (horizontal)
  - *v* (vertikal)
  - Ursprung ist oben links
  - Einheiten: Pixel
- Die **Farbe** eines Bildpunktes kann auf unterschiedliche Weise repräsentiert werden
- **Graustufenbilder**: Für jeden Pixel wird ein Helligkeitswert abgelegt
  - Normalerweise ein Byte pro Pixel, i.A. Werte in $[0, 255]$

Bildrepräsentation II

Monochrombild: Diskrete Funktion

$$Img: [0 \cdots n-1] \times [0 \cdots m-1] \rightarrow [0 \cdots q]$$

$$(u, v) \mapsto Img(u, v)$$



0 für schwarz
255 für Weiß

Üblich:

$$q = 255$$

$$n = 640, \quad m = 480 \text{ (VGA)}$$

oder

$$n = 768, \quad m = 576 \text{ (PAL)}$$

$$n = 1280, \quad m = 960 \text{ (QVGA)}$$

Bildrepräsentation III

Farbbild:

- Verschiedene Farbmodelle für unterschiedliche Anwendungen
- Klassifikation nach erreichbarem Farbraum

Beispiele:

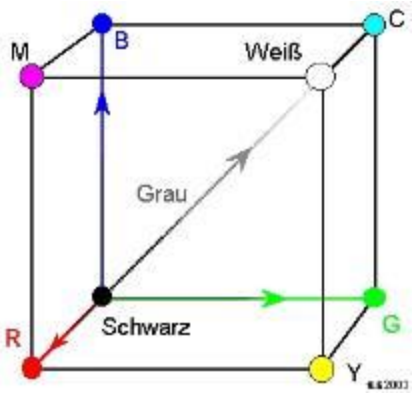
- RGB-Modell (**R**ot-**G**rün-**B**lau-Modell): speziell für Monitore (Phosphor-Kristalle)

$$Img(u,v) = (r,g,b)^T \epsilon \mathbb{R}^3$$

- HSI (Hue, Saturation, Intensity): geeignet für Farbsegmentierung
- CIE: physikalisch (Wellenlänge)
- CMYK-Modell (Cyan, Magenta, Yellow, Schwarzanteil „Key“): Farbdrucker (subtraktive Farbmischung)
- YIQ: Analoges Fernsehermodell

Bildrepräsentation IV

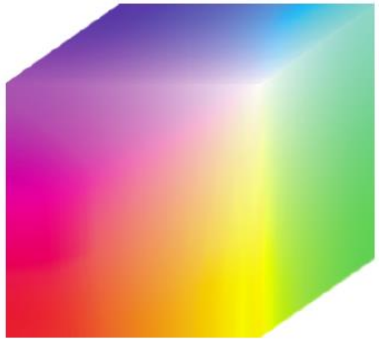
RGB Farbraum



- Additive Farbmischung
- Drei Farbwerte: *Rot*, *Grün*, *Blau*
- *RGB24*
  - Ein Pixel wird durch 3 Bytes repräsentiert (rot, grün, blau)
  - Jedes Byte entspricht 8 Bits
    → 256 Abstufungen für jeden Farbwert
  - $2^8 \times 2^8 \times 2^8$ = 16,8 Mio. Farben darstellbar

$$Img{:}[0\cdots n\text{-}1]\times[0\cdots m-1]\rightarrow[0\cdots R]\times[0\cdots G]\times[0\cdots B]$$

$$(u,v)\mapsto Img(u,v)=(r,g,b)^T$$



24-Bit RGB Farbwürfel

Bildrepräsentation V

HSI (HSV) Farbraum

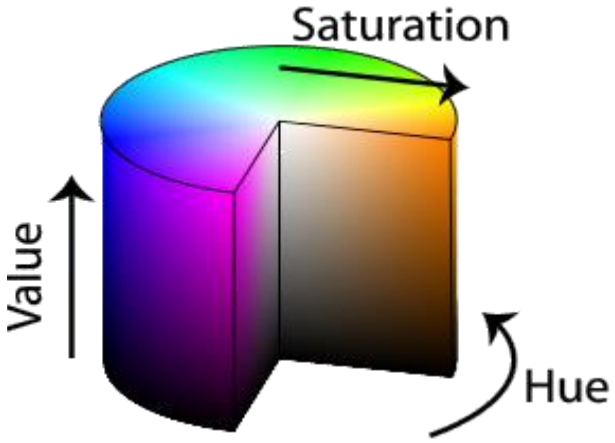
- *Hue* (Farbnuance), *Saturation* (Sättigung), *Intensity/Value* (Helligkeit)
- Kodiert der Farbinformation getrennt von Helligkeit und Sättigung der Farbe
  ⇨ unempfindlich gegen Beleuchtungsänderungen
- Umrechnung von RGB nach HSI
  - H undefiniert, falls R = G = B
  - S undefiniert, falls R = G = B = 0

$$H = \begin{cases} \theta, & falls\ B < G \\ 360 - \theta, & sonst \end{cases}$$

$$\theta = \arccos \frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}$$

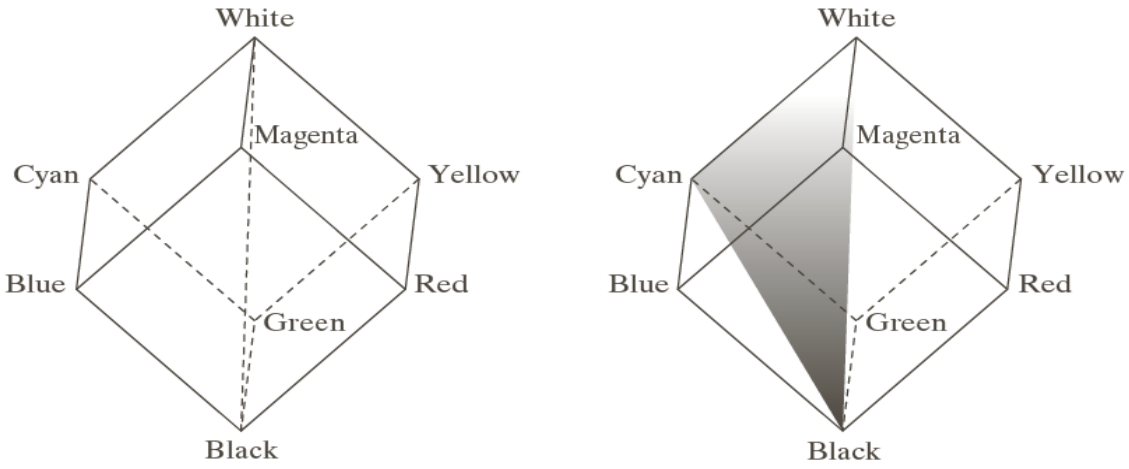
$$S = 1 - \frac{3}{R+G+B} \min(R, G, B)$$

$$I = \frac{1}{3}(R + G + B)$$



Bildrepräsentation V

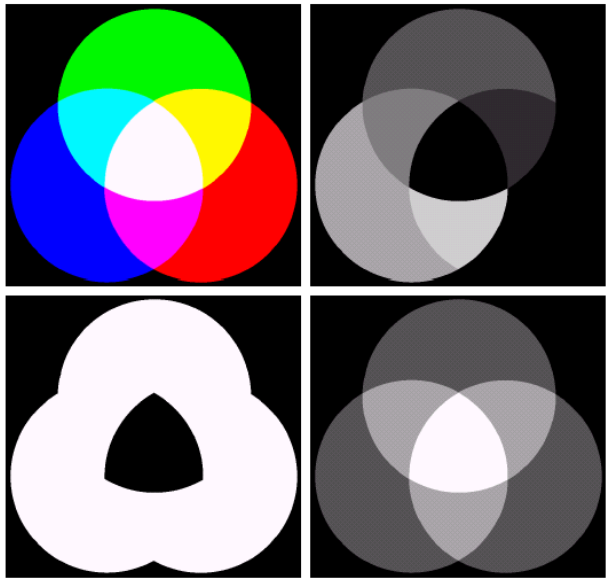
Wie bestimmt man HSI Werte in einem RGB Würfel?



- Im RGB Format enstpricht die ***Helligkeitsachse*** (*Intensity*) der Linie durch die Ecken Schwarz (0,0,0) und Weiß (1,1,1)!
- Die ***Sättigung*** *(Saturation)* eines Farbpunkts auf der Sättigungachse ist Null und erhöht sich mit dem Abstand zur Helligkeitssachse.
- Jeder Punkt im Dreieck hat die selbe ***Farbnuance*** (*Hue*) aber unterschiedliche Helligkeits- und Sättigungswerte, da die Schwarz und Weiß-Komponmente nicht die Farbnuance ändert. Die Farbnuance wird geändert indem das Dreieck um die Helligkeitsachse rotiert wird.

Bildrepräsentation V

RGB



Farbnuance

Sättigung

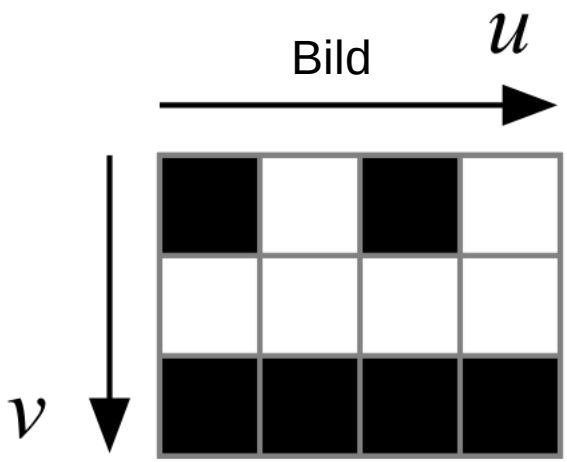
Helligkeit

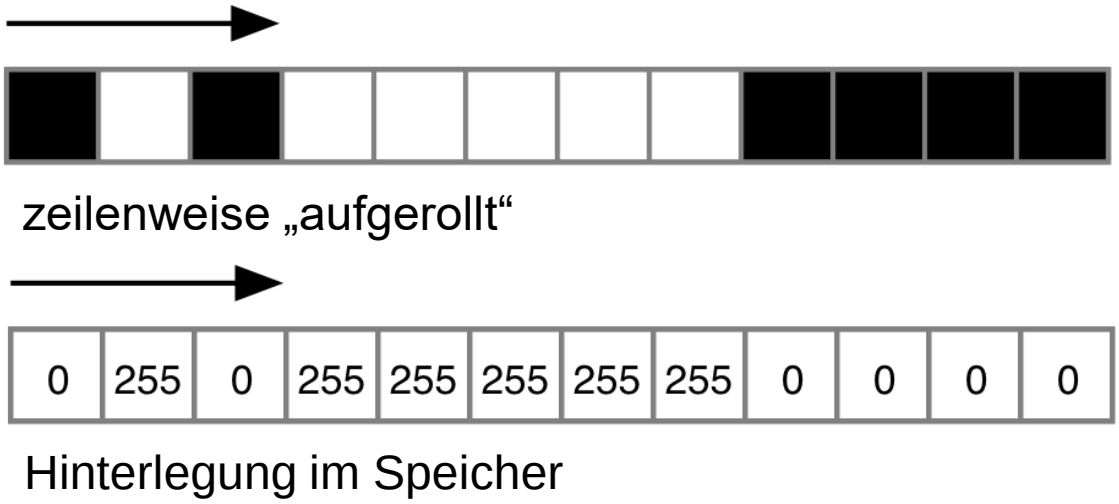
- Die **Hauptunterschiede** im Vergleich zu dem RGB Modell: Das HSV-Farbmodell **entkoppelt Farbwerte** von den **Helligkeitswerten** und erlaubt **unabhängige** Änderung von Farbnuance-, Sättigung- und Helligkeitswerten!

Bildrepräsentation VI

Repräsentation eines 8-Bit Graustufen-Bildes im Speicher

- Pixel werden zeilenweise, linear abgelegt
  - von oben links nach unten rechts
  - Achtung: z.B. bei Bitmaps von unten links nach oben rechts
- Graustufen-Kodierung:
  - Ein Byte pro Pixel
  - 0 schwarz, 255 weiß, dazwischen Graustufen

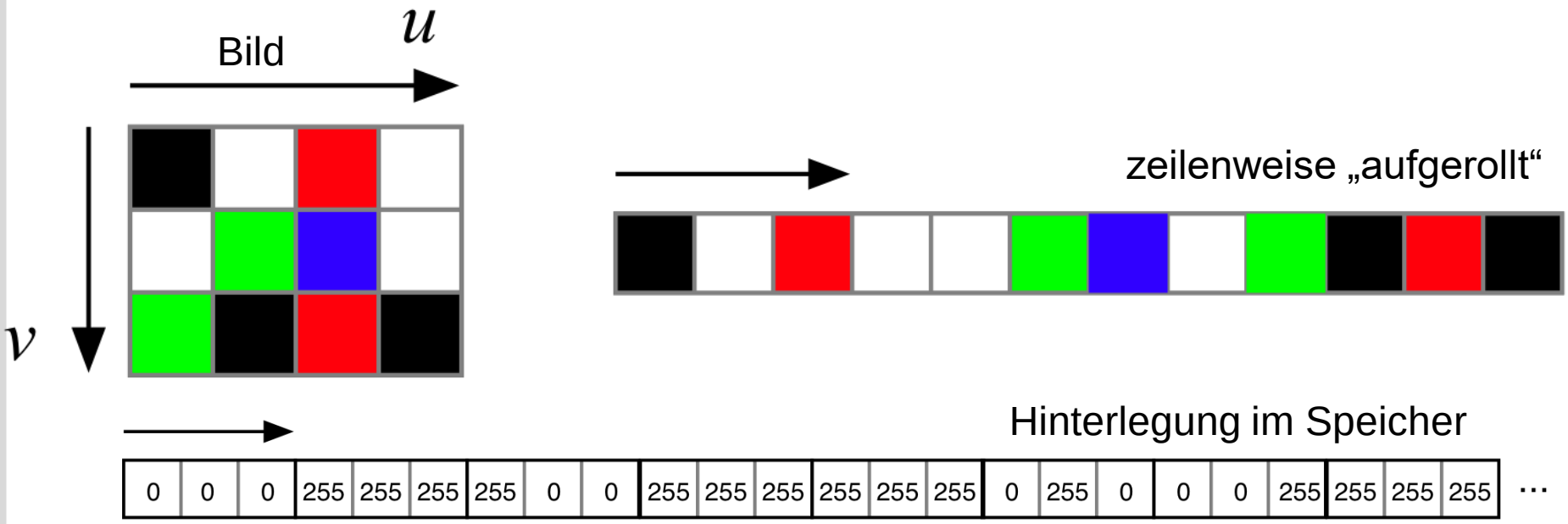




Bildrepräsentation VI

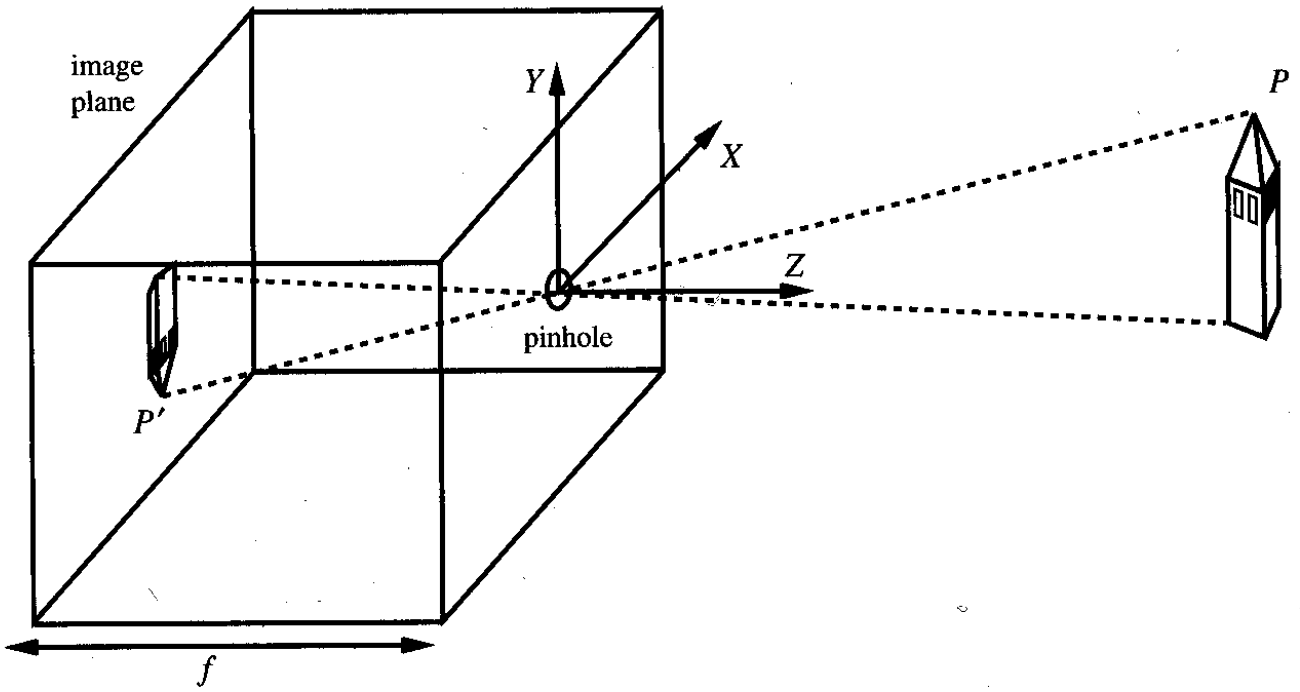
Repräsentation eines RGB24 Farbbildes im Speicher

- Pixel werden zeilenweise, wie beim Graustufen-Bild, abgelegt
- Farbkodierung:
  - Drei Bytes pro Pixel
  - Für jeden Kanal gilt:
    0 minimale, 255 maximale Intensität



Bildgenerierung: Lochkamera

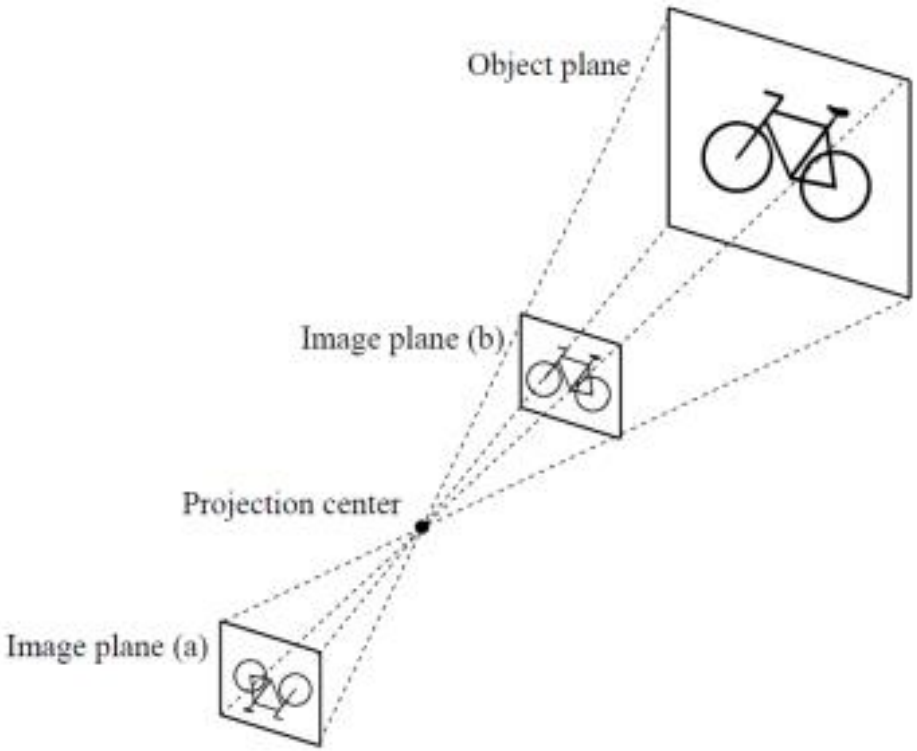
- Einfachstes Modell: Lochkameramodell



Interner Parameter: Brennweite f („Fokalabstand“)

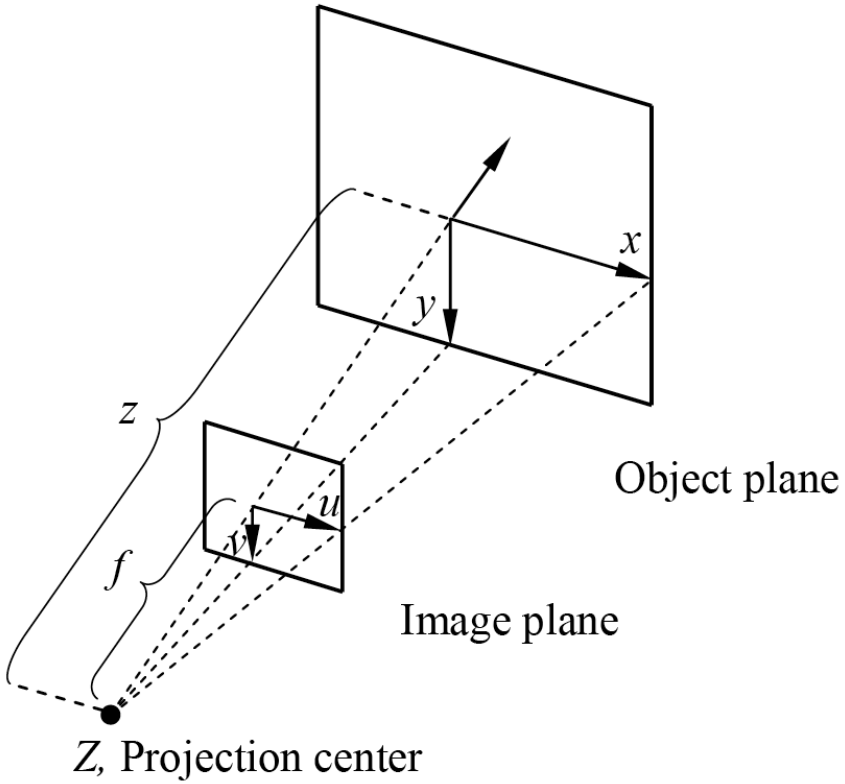
Bildgenerierung: Lochkamera II

- Klassisches Lochkameramodel
  - Projektionszentrum liegt vor der Bildebene, also zwischen Szene und Bildebene
  - Dadurch: Horizontal und vertikal gespiegelte Abbild



Bildgenerierung: Lochkamera III

- Oft verwendete Variante: Lochkameramodell in Positivlage
  - Projektionszentrum liegt hinter der Bildebene
  - Dadurch: keine Spiegelung (Minuszeichen entfallen)

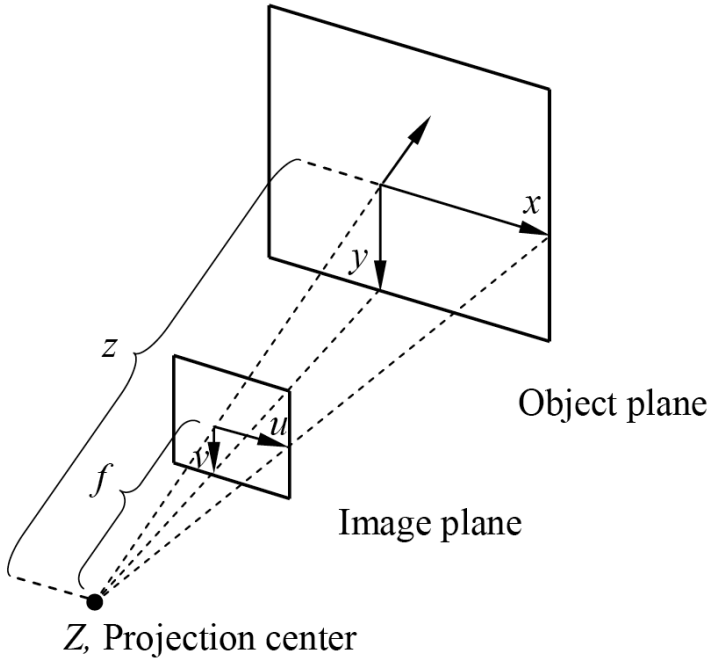


Bildgenerierung: Lochkamera IV

Zweiter Strahlensatz zur Projektion eines Szenenpunktes (x, y, z) auf einen Bildpunkt (u, v):

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix}$$

Bei der Projektion geht
die z-Komponente verloren!

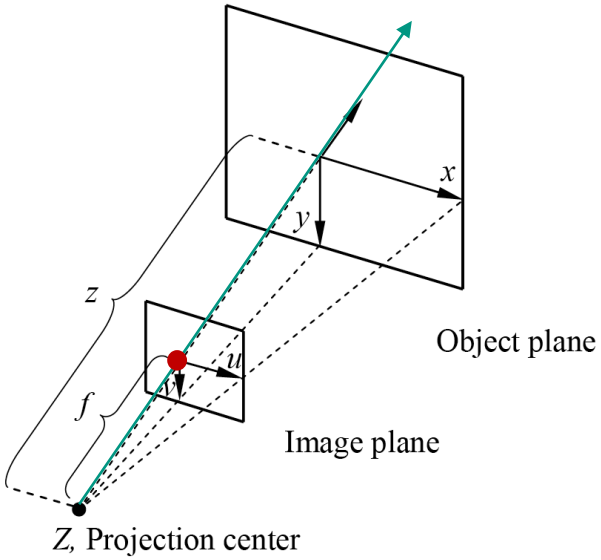


Rückprojektion:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{z}{f} \begin{pmatrix} u \\ v \end{pmatrix}$$

Koordinatensysteme

- Hauptachse (*principal axis*): Gerade durch das Projektionszentrum, rechtwinklig zur Bildebene
- Hauptpunkt (*principal point*): Schnitt der Hauptachse mit der Bildebene
- Bildkoordinaten: 2D Koordinaten *(u, v)* eines Punktes im Bild. Einheit: Pixel



- Kamerakoordinatensystem: 3D Koordinaten *(x, y, z)* eines Punktes relativ zur Kamera. Der Ursprung liegt im Projektionszentrum, die x- und y-Achse ist parallel zur u- bzw. v-Achse in der Bildebene. Einheit: mm
- Weltkoordinatensystem: 3D Basiskoordinatensystem in der Welt. Einheit: mm

Kameraparameter

- Parameter, die das Kameramodell vollständig beschreiben
- Man unterscheidet zwischen intrinsischen und extrinsischen Kameraparameter
  - Intrinsische Parameter: Sie sind unabhängig von der Auswahl des Weltkoordinatensystems. Sie bleiben konstant bei Veränderung des Aufbaus
  - Extrinsische Parameter: modellieren die Transformation von Weltkoordinatensystem in das Kamerakoordinatensystem. Sie sind bei einer neuen Lage der Kamera neu zu bestimmen.

- Bisher:
  - Im Lochkameramodel nur ein intrinsischer Kameraparameter (f)
  - Kein Weltkoordinatensystem berücksichtigt, deshalb keine extrinsischen Parameter
  - Annahmen: Hauptpunkt im Ursprung des Bildkoordinatensystem; Pixel sind exakt quadratisch ; Keine Linzenverzerrung.

Filteroperationen

- **Filter** in der Bildverarbeitung
  auch *spatial filters, spatial masks, kernels, windows*

- Ein Filter besteht aus
  - einer Nachbarschaft (meist ein kleines Rechteck)
  - einer vordefinierten Operation

- Ein Filter wird auf alle Pixel des Bildes angewendet
  - Berechnung eines neuen Pixelwertes durch Anwendung der Filteroperation unter Berücksichtigung der Nachbarschaftspixel

- **Lineare Filter** Eigenschaften

$$f(x+y) = f(x) + f(y) \quad \Rightarrow \quad \text{additiv}$$
$$f(\alpha x) = \alpha f(x) \quad \Rightarrow \quad \text{homogen}$$

Filteroperationen

- **Tiefpassfilter**: Glättung, Rauschelimination
  - Mittelwertfilter
  - Gauß-Filter
- **Hochpassfilter**: Kantendetektion
  - Prewitt
  - Sobel
  - Laplace
  - Roberts
- **Kombinierte** Operatoren
  - Laplacian of Gaussian

Mittelwertfilter

- Ziel: Rauschunterdrückung
  - Beispiel: Durchschnitt aus einem Pixel und seine 8 Nachbarn;
    Größe beliebig wählbar
  - 3x3 Mittelwertfilter: Filtermaske

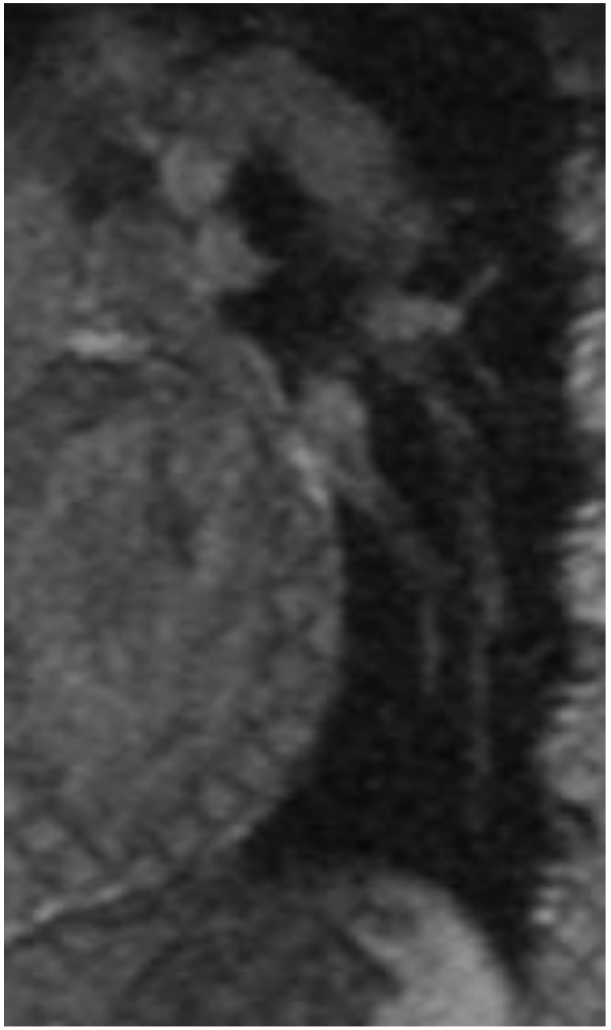
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

13	25	30	34	40	46	60	76
45	(1/9)*50	(1/9)*52	(1/9)*55	65	67	87	77
34	(1/9)*45	(1/9)*55 => 52	(1/9)*60	54	45	56	65
45	(1/9)*50	(1/9)*52	(1/9)*48	45	65	65	45
34	34	54	56	57	58	67	70
45	46	46	46	45	53	52	60
50	68	69	60	70	70	78	79
68	70	78	78	80	80	80	90

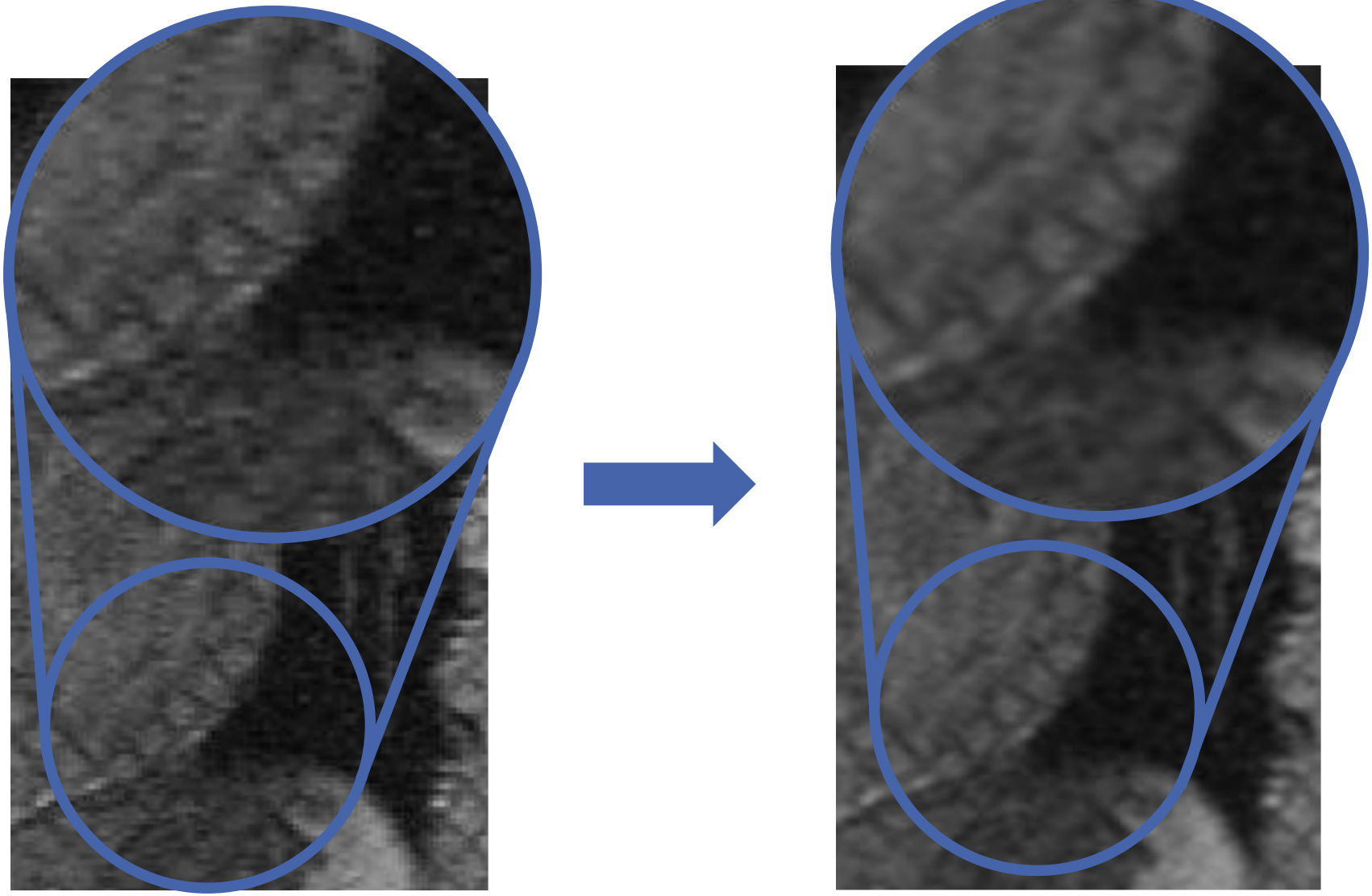
$$F_{Mittelwert} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$m'(x,y) = \sum_{j=-1}^{1}\sum_{i=-1}^{1} m(i,j) \cdot p(x-j, y-i)$$

Mittelwertfilter II



Mittelwertfilter II



Gauß-Filter

- **Ziel:** Rauschunterdrückung, Glättung
- Definiert durch zweidimensionale Gauß-Funktion

- $f(x,y) = \frac{1}{2\,\pi\sigma^2} e^{-\frac{x^2+y^2}{2\,\sigma^2}}$ $\quad\longleftrightarrow\quad$ $F(u,v) = e^{-\frac{u^2+v^2}{2}\sigma^2}$

Ortsbereich $\qquad$ Frequenzbereich

- Approximation von $f(x)$ durch einen 3×3-Filter für $\sigma = 0.85$:

$$F_{Gauß} = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Gauß-Filter II

- Die Stärke der Glättung ist ausschließlich durch den Parameter σ bestimmt: Je größer σ, umso stärker die Glättung.
- Die Größe $n \times n$ der Filtermaske beeinflusst die Güte der Approximation des Filters

Originalbild

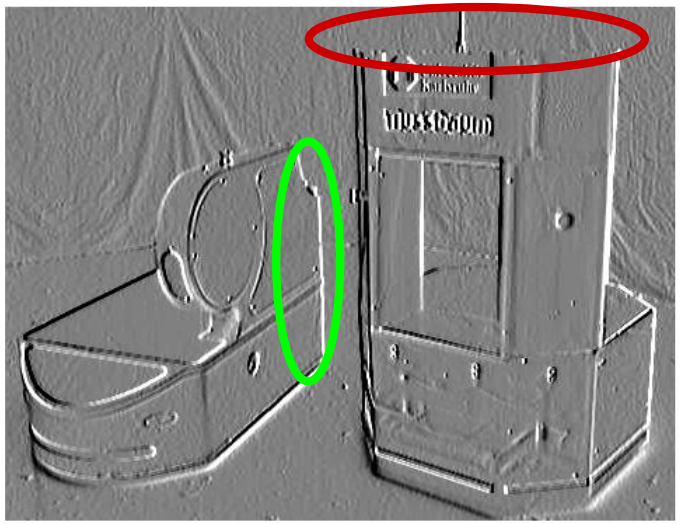
$\sigma^2 = 4$

$\sigma^2 = 16$

Filterung – Prewitt

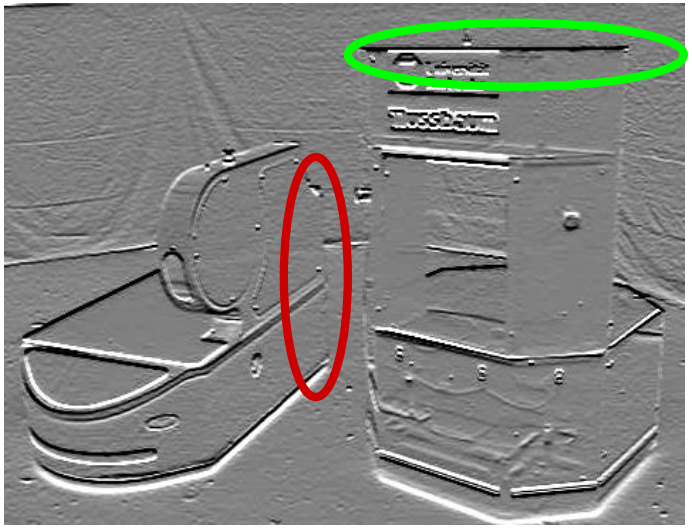
- **Prewitt-X Filter** $P_x = \frac{\partial g(x,y)}{\partial x}$
  - Approximiert durch

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



- **Prewitt-Y Filter** $P_y = \frac{\partial g(x,y)}{\partial y}$
  - Approximiert durch

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



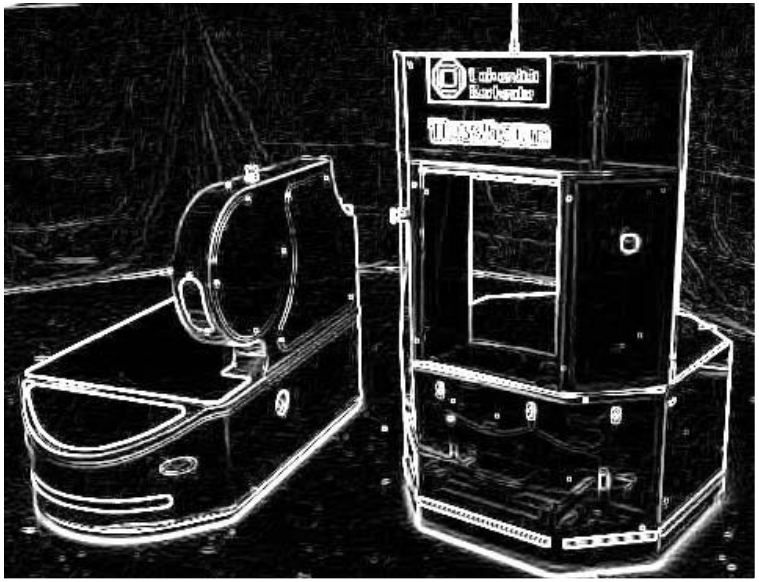
- **Eigenschaften**:
  - Gute Ergebnisse bei Detektion von vertikalen bzw. horizontalen Kanten

Filterung – Prewitt III

- Kombination der Prewitt-Filter zur Bestimmung des Gradientenbetrags M

$$M \approx \sqrt{P_x^2 + P_y^2}$$

- Danach: Schwellwertfilterung

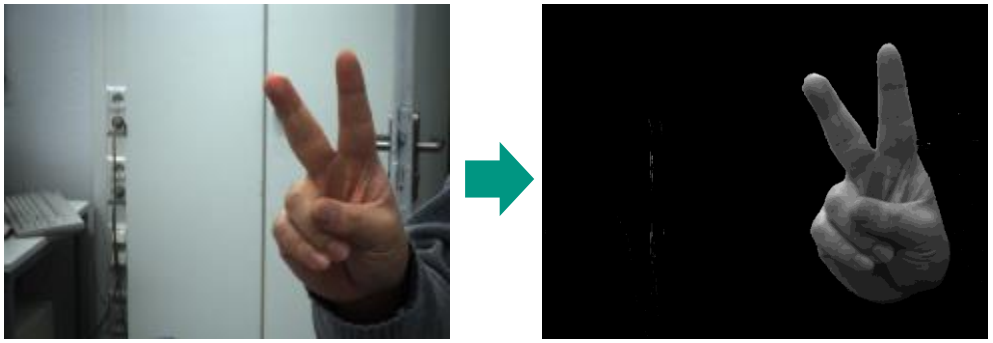


Segmentierung

- **Segmentierung** ist die Aufteilung eines Bildes in aussagekräftige Segmente
- Jedes Pixel wird mindestens einem Segment zugeordnet
- Identifikation von interessante Bildregionen für die Analyse, Erkennung und Klassifikation

Mögliche Verfahren:

- Schwellwertfilterung
- Clustering
- Kantenextraktion
- Region-Growing

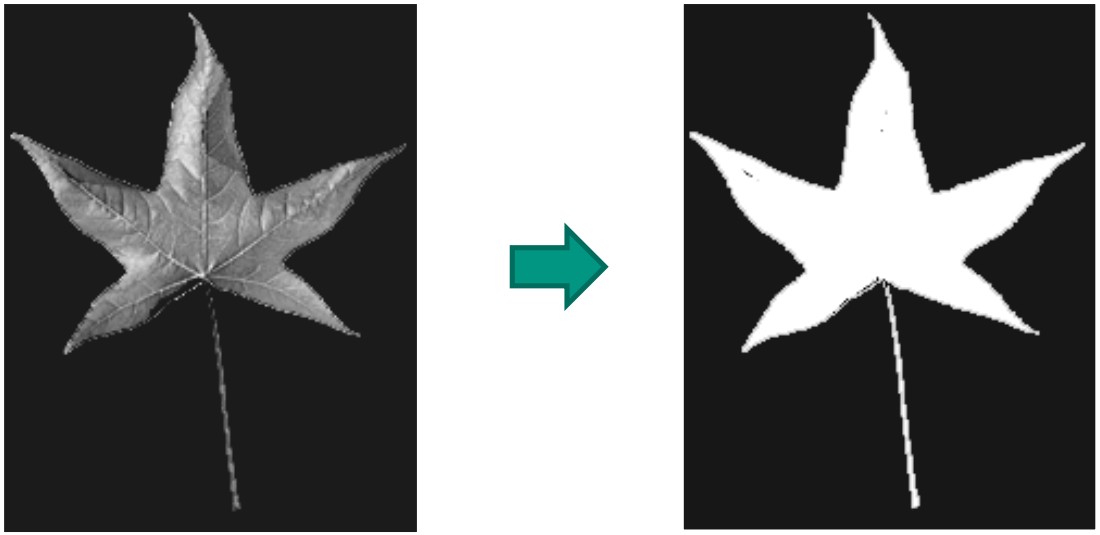


Segmentierung: Schwellwertfilterung I

- Schwellwertfilterung zur Konvertierung eines Grauwertbildes in ein binäres Bild

- Intensität von jedem Pixel (u, v) wird mit einem vordefinierten Schwellwert T abgeglichen

$$Img'(u,v) = \begin{cases} 255 & falls\ Img(u,v) > T \\ 0, & sonst \end{cases}$$



Segmentierung: Schwellwertfilterung II

- Oft können Objekte über ihre Farbe segmentiert werden:
  - Menschliche Haut
  - Einfarbige Objekte

- Beispiel:
  - Intervallschranken im HSV-Farbraum:

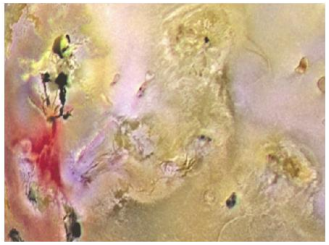
$$Img'(u,v) = \begin{cases} 255 & falls \quad \begin{array}{l} H_{max} \geq Img_H(u,v) \geq H_{min}, \\ S_{max} \geq Img_S(u,v) \geq S_{min}, \\ V_{max} \geq Img_V(u,v) \geq V_{min} \end{array} \\ 0 & sonst \end{cases}$$

- Problem:
  - Wechselnde Lichtbedingungen
  - Reflexionen, Schattenwürfe

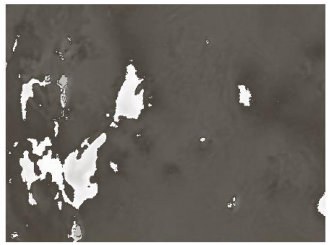
HSV Segmentierung

Wie kann die rote Region extrahiert werden? (HSV)

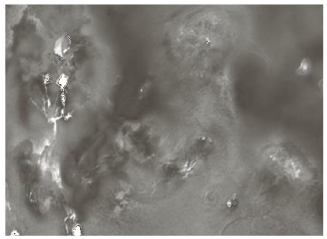
Original Bild



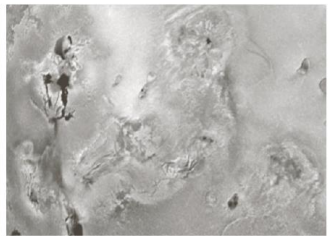
Farbnuance



Sättigung



Helligkeit

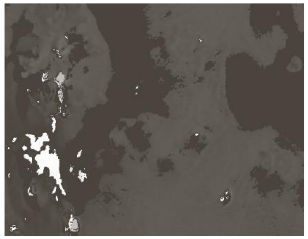


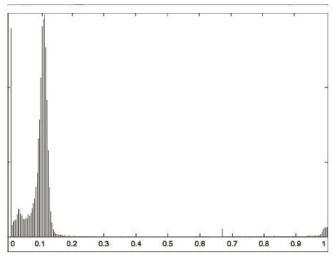
HSV

Segmentierung ist die **Unterteilung** eines Bildes in **plausible** Regionen

Binärmaske durch Schwellwertfilterung des **Sättigungs**bildes (>10%)

Farbnuancenwerte des Bildes werden mit der Maske multiplizieren.





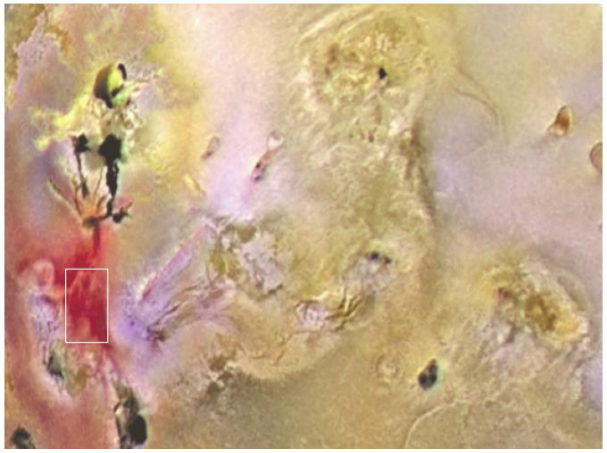
Schwellwertfilterung Anwenden auf das Histogramm!

Berechnung des **Histogramms**

Segmentiertes Bild

RGB Segmentierung

Original Bild



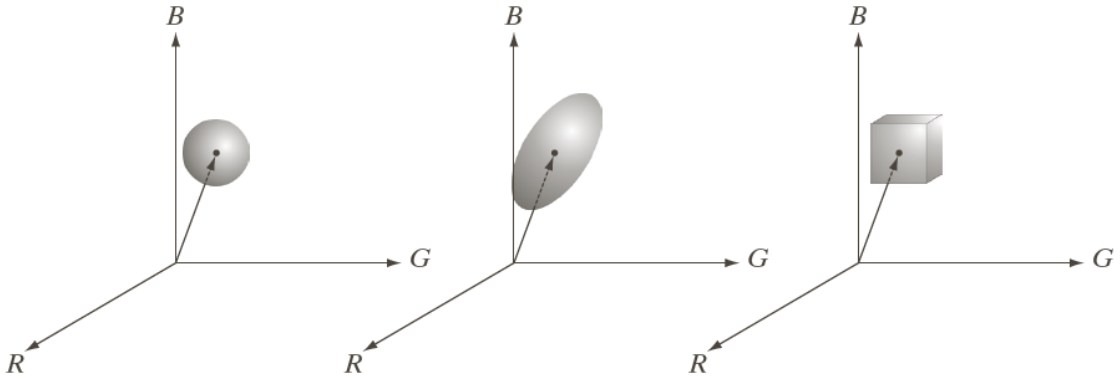
Wie kann die rote Region extrahiert werden? (RGB)

Definiere region of interest (**ROI**).

Euklidische Distanz
der beiden Farbvektoren!

$$D(z,a) = \|z-a\| = [(z_R - a_R)^2 + (z_G - a_G)^2 + (z_B - a_B)^2]^{\frac{1}{2}}$$

$$D(z,a) \leq D_0$$

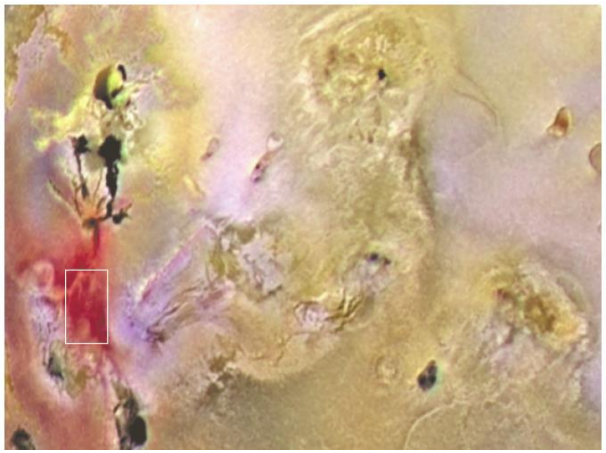


Unterschiedliche Bereiche!

Klassifiziere jedes RGB Pixel, ob der Farbwert in einem spezifizierten Bereich liegt oder nicht!

RGB Segmentierung

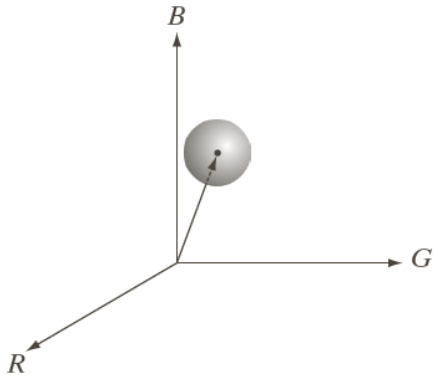
Wie kann die rote Region extrahiert werden? (RGB)



Original Bild

$$D(z,a) \leq D_0$$

Segmentiertes Bild

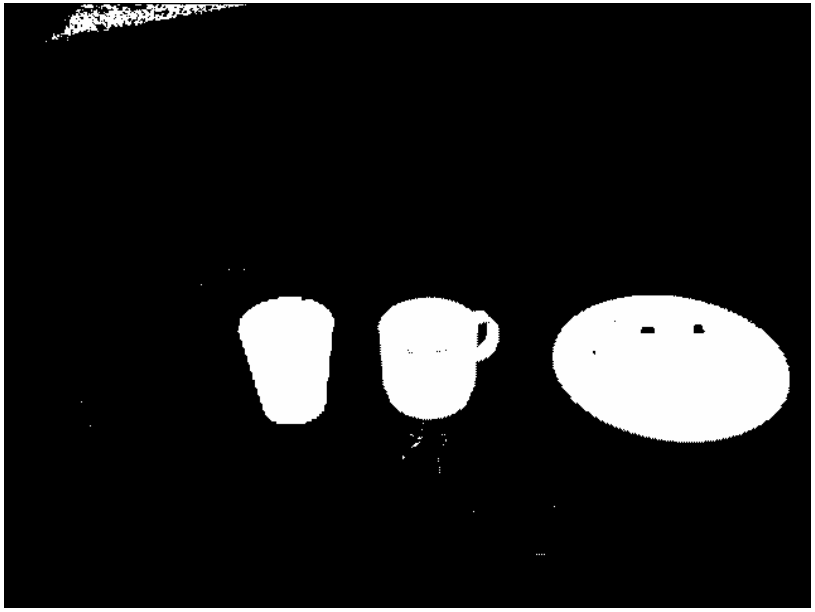


Klassifiziere jedes RGB Pixel, ob der Farbwert in einem spezifizierten Bereich liegt oder nicht!

Segmentierung: Beispielanwendung

- Beispielanwendung: Objekterkennung und -lokalisierung

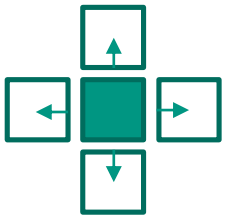
Eingabebild



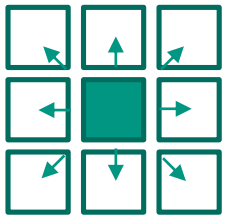
Ergebnis der Segmentierung

Morphologische Operatoren

- Morphologische Operatoren werden oft für die Nachbearbeitung binärer Bilder verwendet (z.B. Ergebnis einer Farbsegmentierung)
- Gängige morphologische Operatoren:
  - **Dilatation**
    Die Dilatation vergrößert Pixel zu größeren Bereichen
  - **Erosion**
    Die Erosion entfernt vereinzelte Pixel und schwach zusammenhängende Pixelgruppen
- Der Effekt eines morphologischen Operators hängt von der Größe und der Form der berücksichtigten Pixelnachbarschaft ab



4er-Nachbarschaft



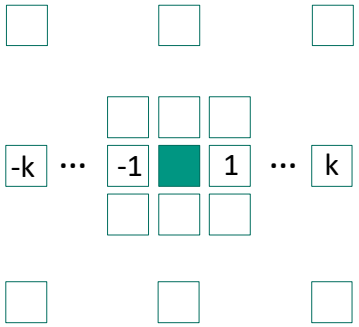
8er-Nachbarschaft

Morphologische Operatoren II

Algorithmus 19 Dilatation$(I, n) \rightarrow I'$

```
k := div((n − 1), 2)
for all pixels (u, v) in I' do
    I'(u, v) := 0
end for
for v := k to h − k − 1 do
    for u := k to w − k − 1 do
        if I(u, v) = q then
            for i := −k to k do
                for j := −k to k do
                    I'(u + j, v + i) = q
                end for
            end for
        end if
    end for
end for
```

Bildhöhe h,
Bildbreite w

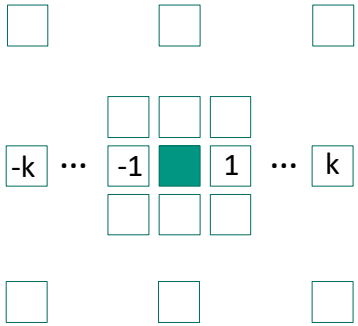


Morphologische Operatoren III

Algorithmus 20 Erosion$(I, n) \rightarrow I'$

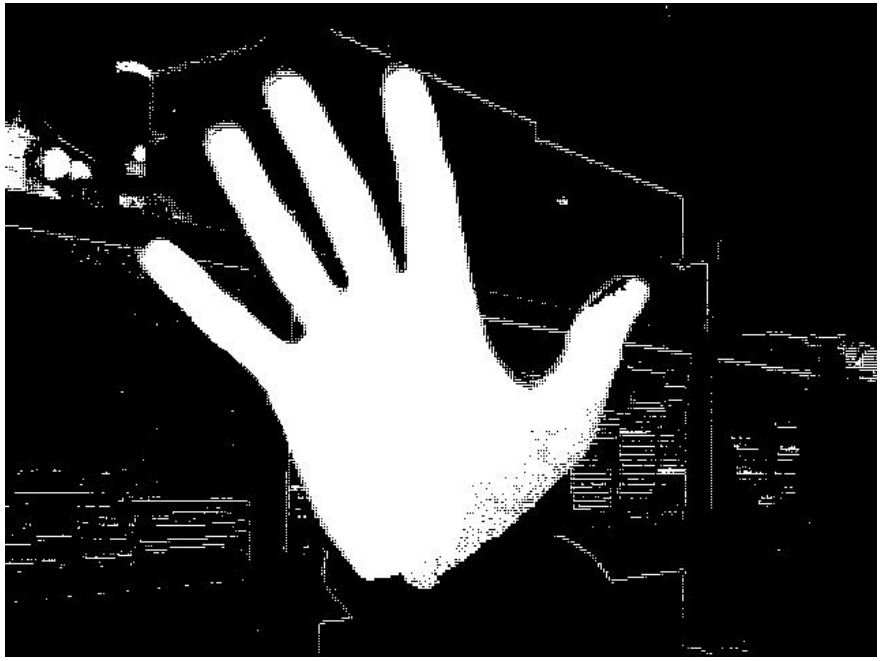
```
k := div((n − 1), 2)
for all pixels (u, v) in I' do
   I'(u, v) := 0
end for
for v := k to h − k − 1 do
   for u := k to w − k − 1 do
      if I(u, v) = q then
         for i := −k to k do
            for j := −k to k do
               if I(u + j, v + i) ≠ q then
                  goto NEXT
               end if
            end for
         end for
         I'(u, v) = q
         NEXT:
      end if
   end for
end for
```

Bildhöhe h,
Bildbreite w



Morphologische Operatoren IV

- Beispielanwendung einer Erosion

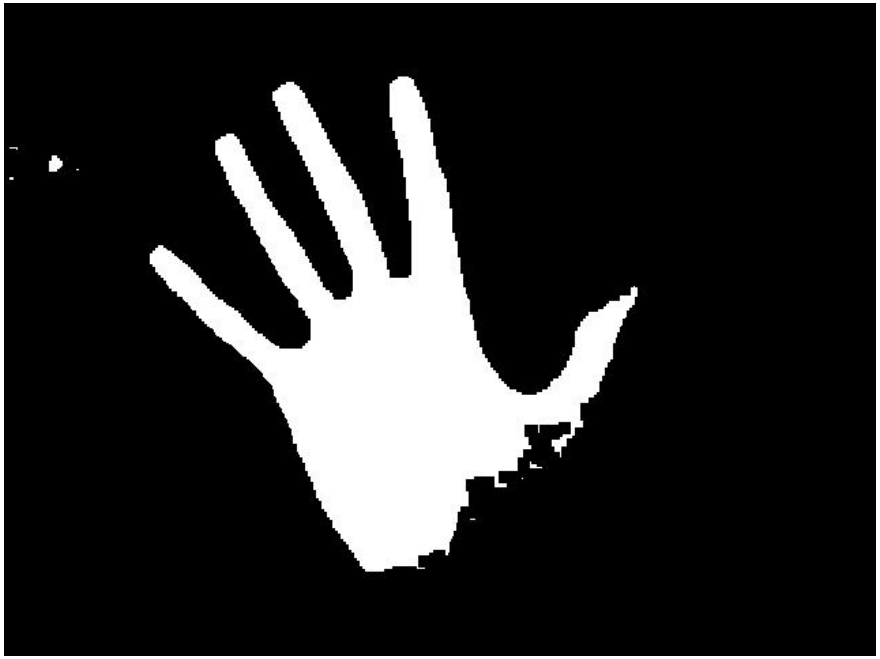


Eingabebild

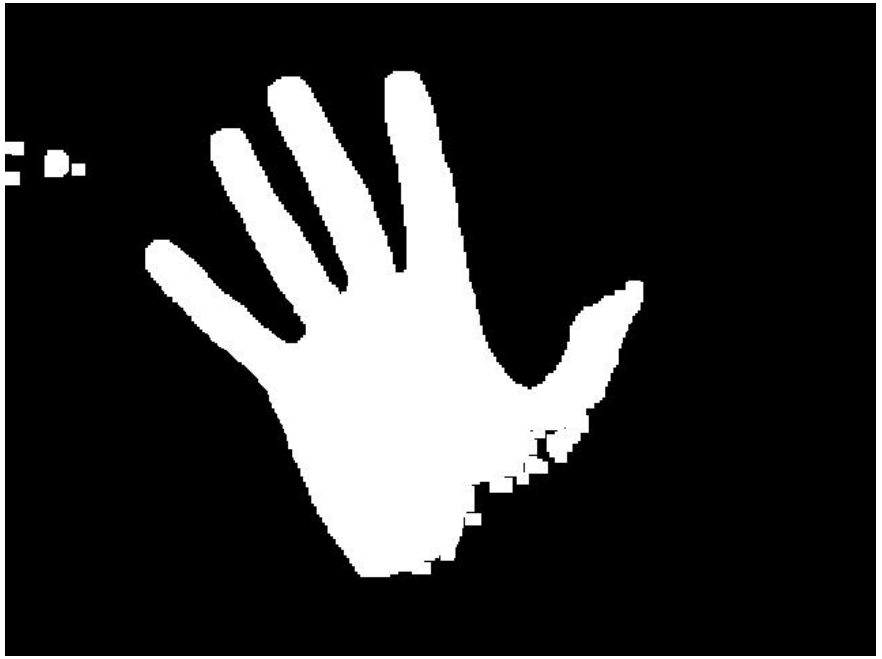
Ergebnisbild

Morphologische Operatoren V

- Beispielanwendung einer Dilatation



Eingabebild



Ergebnisbild

Canny-Kantendetektor

- Nach John F. Canny aus dem Jahre 1986
- Der Canny-Kantendetektor ist weit verbreitet und bezüglich der Leistung im Vergleich zu anderen Kantendetektoren im Allgemeinen überlegen.
- Ziel war es den “optimalen” Kantendetektor zu finden:
  - Gute Detektion
  - Gute Lokalisierung
  - Minimale Antwort (“dünne Linien”)
- Canny-Kantendetektor berechnet binäre Antwort (üblicherweise 0: keine Kante, 255: Kante)
- Subpixelgenauigkeit durch Erweiterung möglich
- Algorithmus besteht aus mehreren Schritten

Canny-Kantendetektor

- Das Prinzip basiert auf drei Grundsätzen:

1. **Geringe Fehlerrate**: Alle Kanten sollen gefunden werden und detektierte Kanten sollten so nah wie möglich an den realen Kanten sein.

2. **Kantenpunkte sollen gut detektiert werden:** Distanz zwischen detektierten Kantenpunkten und dem Zentrum der realen Kanten soll minimal sein.

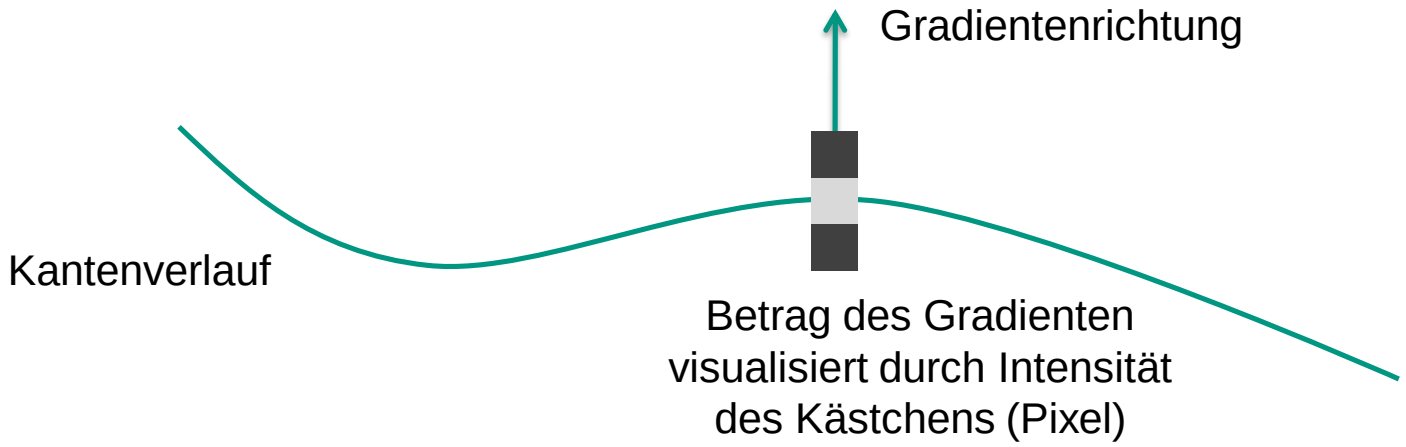
3. **Eindeutigkeit**: Der Detektor soll nur ein Punkt, nicht mehrere Kantenpunkte, zu einem realen Kantenpunkt liefern.

Canny-Kantendetektor II

- Der Algorithmus
  1. Rauschunterdrückung: Gauß-Filter
  2. Berechnung der Gradienten in horizontaler und in vertikaler Richtung Prewitt_X / Prewitt_Y oder Sobel_X / Sobel_Y
     a) Berechnung der Richtung: $\phi = \text{atan}(\frac{g_y}{g_x})$
     b) Einteilung der Richtung in vier Quadranten:
        1 : [-67.5°, -22.5°) 2 : [-22.5°, 22.5°),
        3 : [22.5°, 67.5°) 4 : [-90°, -67.5°) oder [67.5°, 90°]
     c) Magnitude $M = \sqrt{g_x^2 + g_y^2}$
  3. Non-Maximum Supression
  4. Hysterese-Schwellwertverfahren

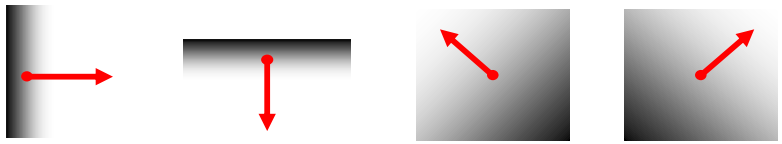
Canny-Kantendetektor III

- Der Canny-Kantendetektor verwendet **Non-Maximum Supression** (d.h. eine Kantenausdünnung)
  - Gradient muss lokales Maximum sein
  - Betrachtung der zwei direkten Nachbarn entlang der Gradientenrichtung
  - Überprüfung erfolgt gemäß dem jeweiligen Quadranten



Canny-Kantendetektor III

Definiere **vier** Kantenorientierung

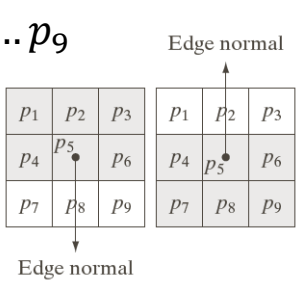


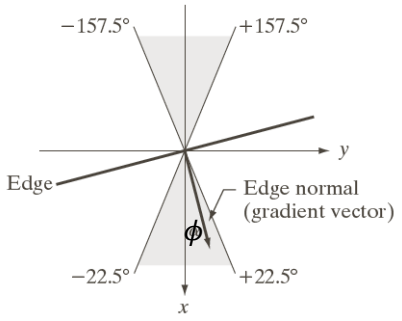
Vertikal | Horizontal | $+45°$ | $-45°$

Definiere einen Bereich um alle möglichen Kantenrichtungen in vier Richtungen zu unterteilen

Kantennormale: $\phi(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$

Beispiel: $p_1 \ldots p_9$

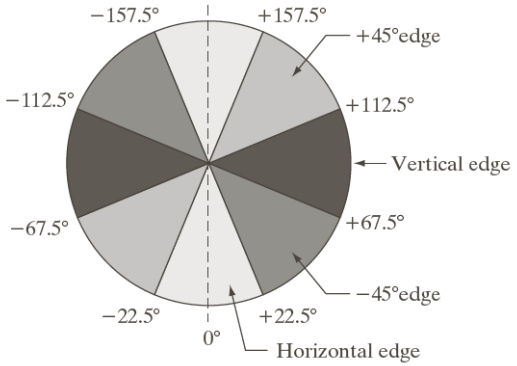




1. Definiere eine 3×3 Region (p_1 bis p_9) um jeden Punkt (x, y) in $\phi(x, y)$.

2. Bestimme die Richtung der Kante welche nah bei $\phi(x, y)$ist. (In diesem Beispiel ist p_5 horizontal.)

3. Wenn der Wert im Zentrum der Mangitude M kleiner als einer der beiden Nachbarn (In diesem Fall p_2 und p_8) in Richtung der Kantennormale ist, dann setze $M = 0$ (Suppression!)



Canny-Kantendetektor IV

- Hysterese-Schwellwertverfahren
  1. Verwendung von zwei Schwellwerten: low / high
  2. Liegt der Betrag des Gradienten für ein Pixel über dem Schwellwert „high“, so wird es auf jeden Fall als Teil einer Kante akzeptiert
  3. Liegt der Betrag des Gradienten für ein Pixel unter dem Schwellwert „low“, so wird es als Teil einer Kante abgelehnt
  4. Ausgehend von den akzeptierten Pixel werden Kanten (rekursiv) verfolgt:
     - Überprüfung der acht direkten Nachbarn
     - Betrag Gradient muss über Schwellwert low liegen

Anmerkung: Lokale Maximalitätsbedingung aus Schritt 3 muss in jedem Fall erfüllt sein!

Canny-Kantendetektor V

- Beispiel