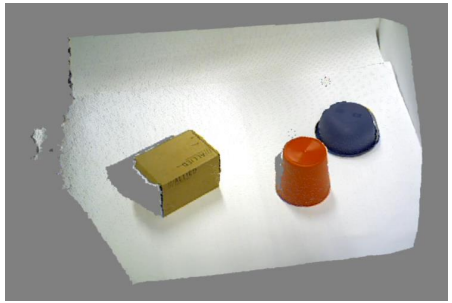
Karlsruher Institut für Technologie

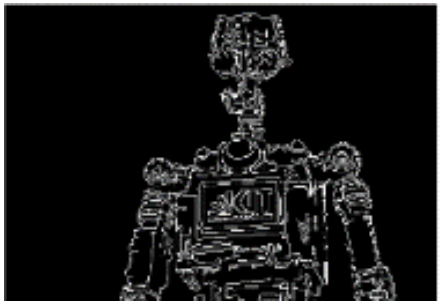
Hochperformante Humanoide Technologien

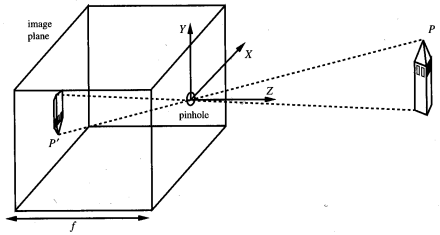
Robotik I: Einführung in die Robotik
Kapitel 9 – Bildverarbeitung in der Robotik

Tamim Asfour

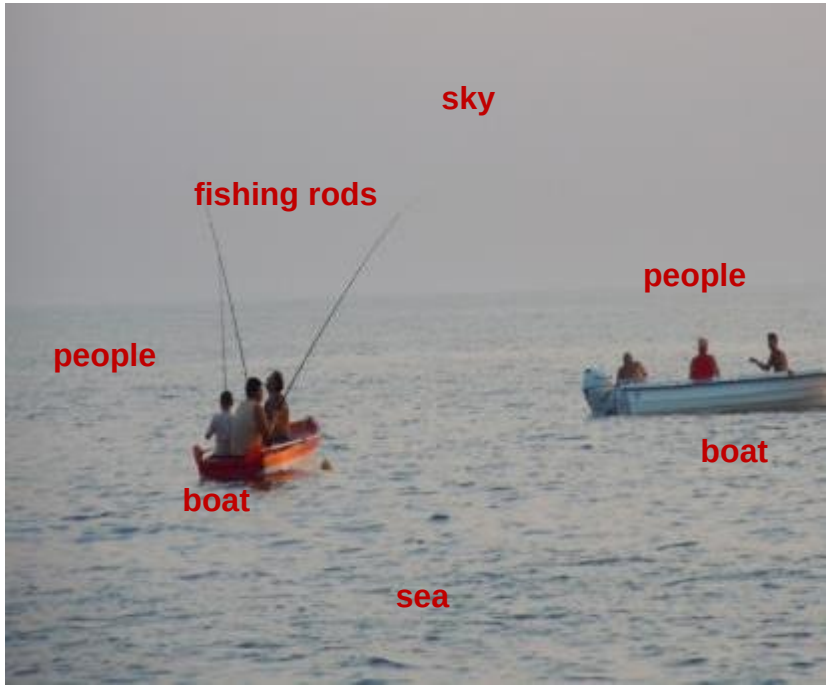
http://www.humanoids.kit.edu



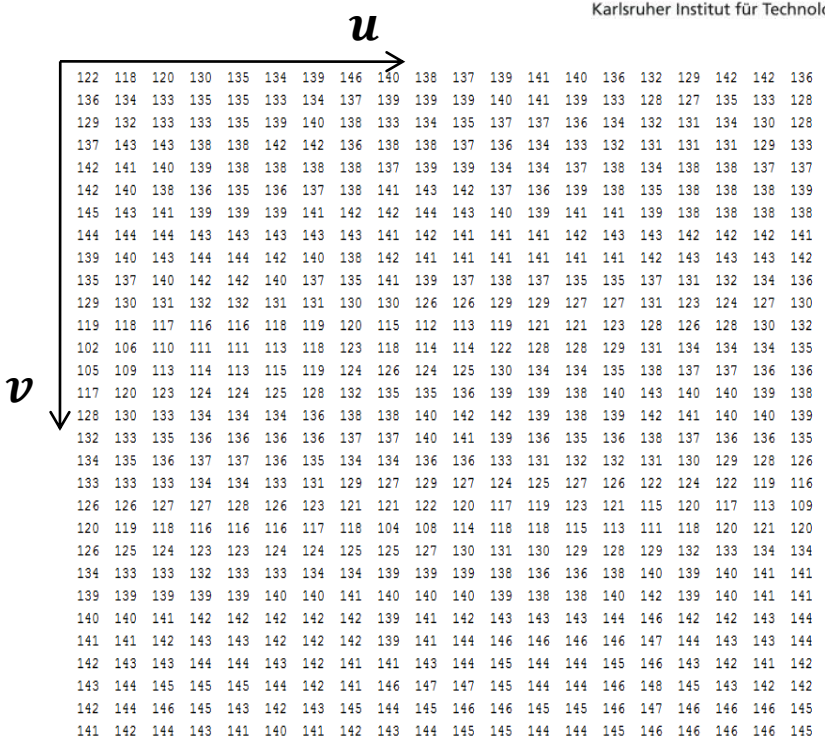




Motivation - Bildverstehen



Was wir sehen!



Was ein Roboter sieht!

Was ist Computer Vision?

Karlsruher Institut für Technologie

Semantics

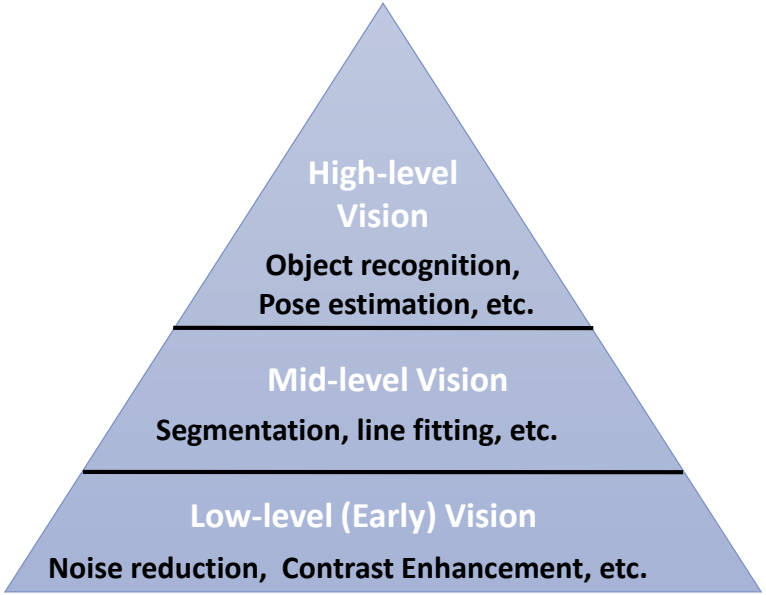
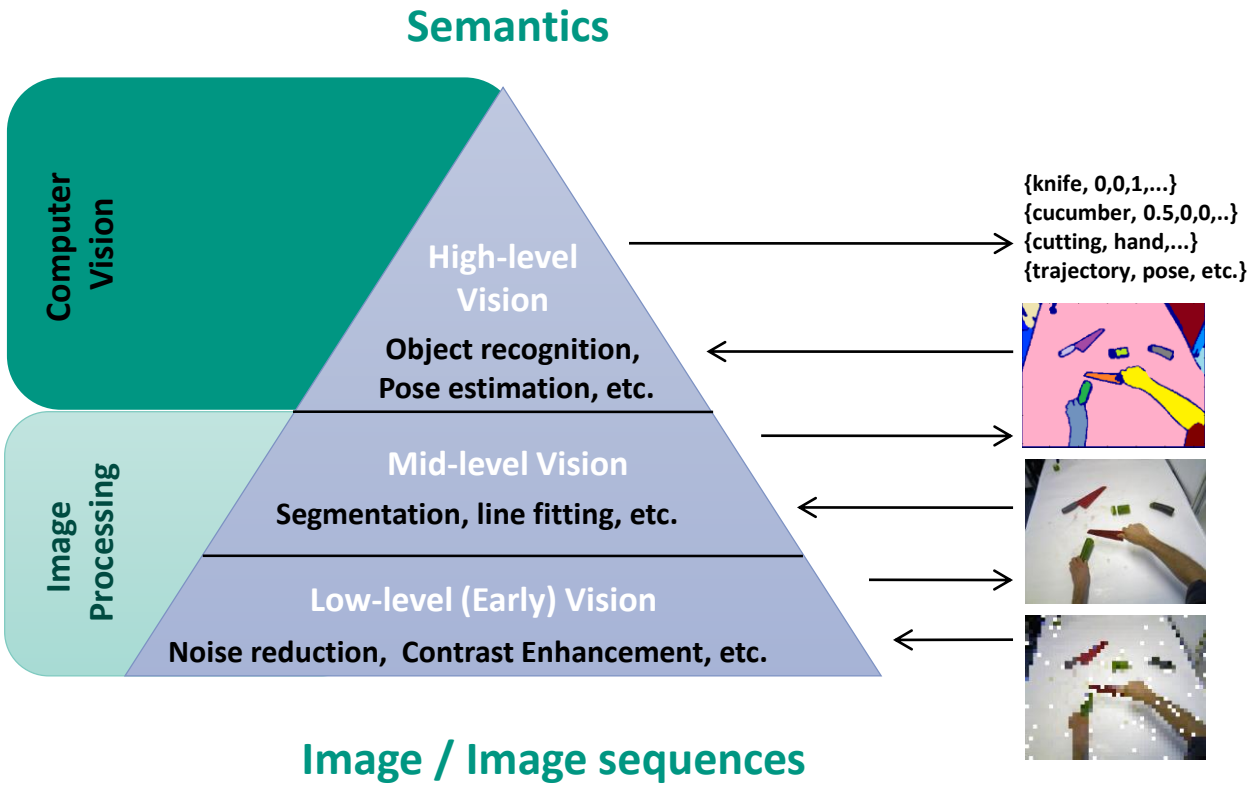


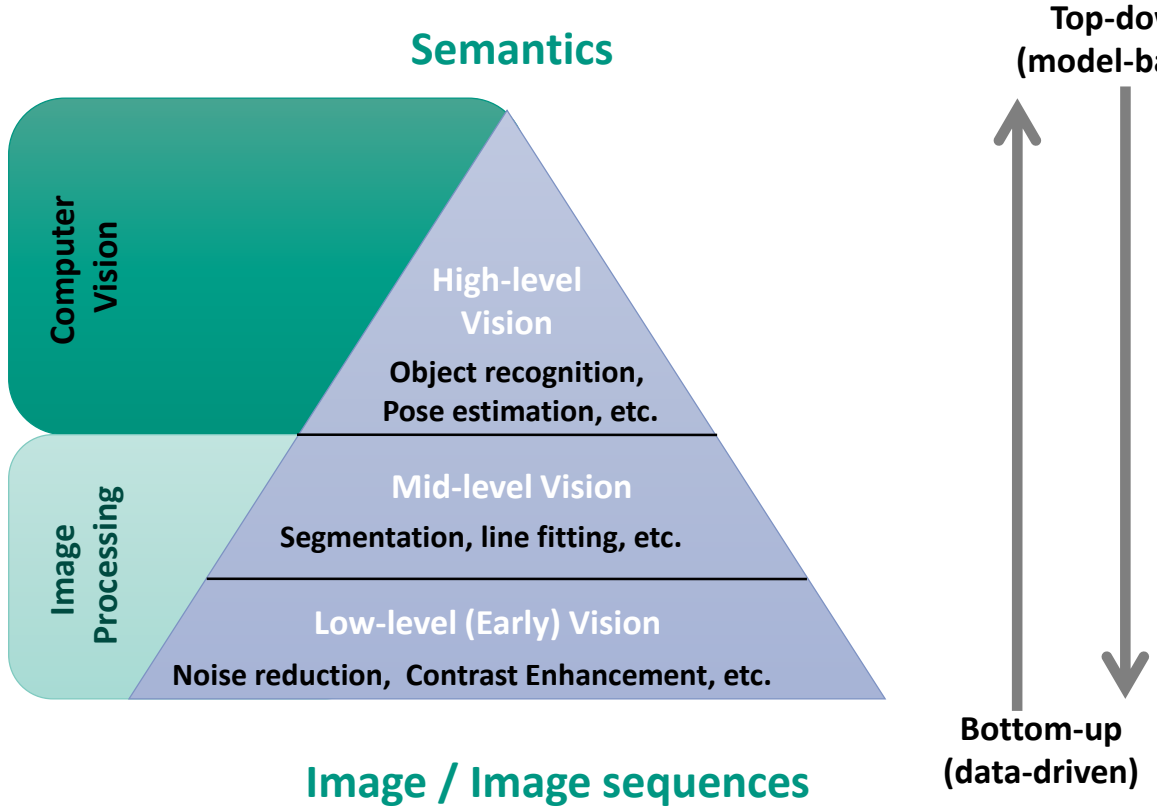
Image / Image sequences

H2T

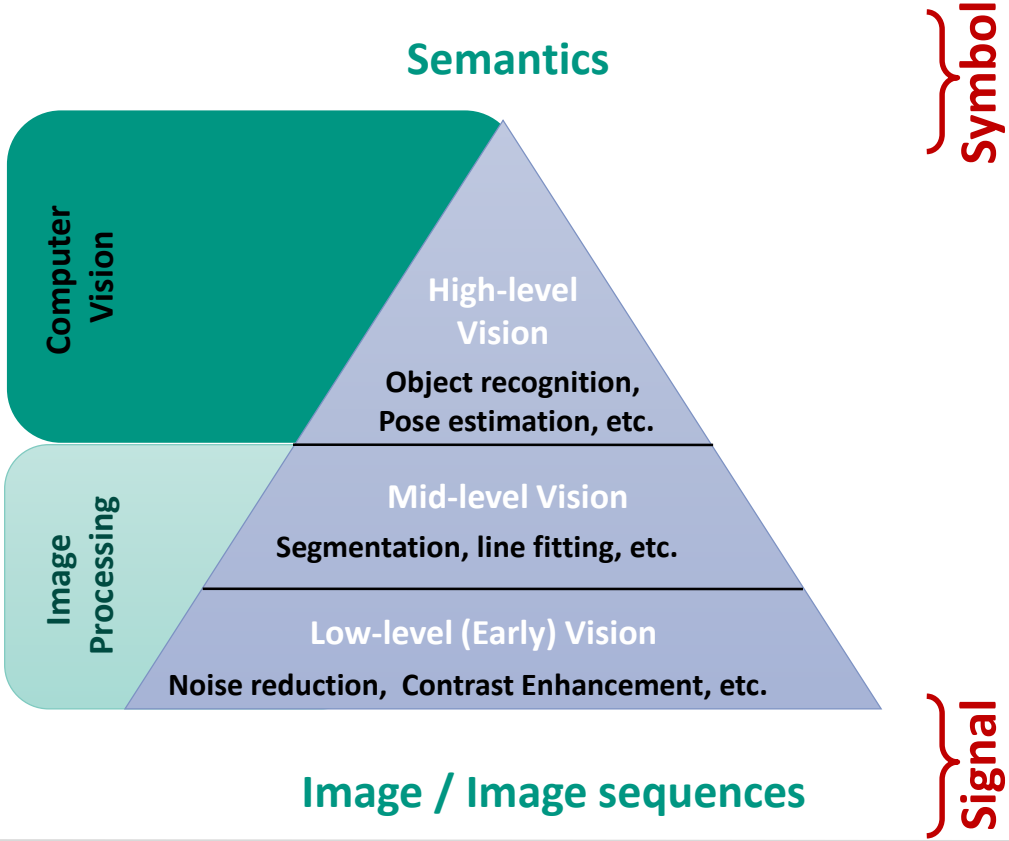
Was ist Computer Vision?

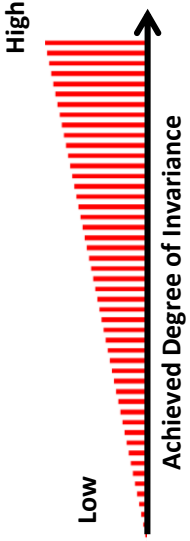


Was ist Computer Vision?

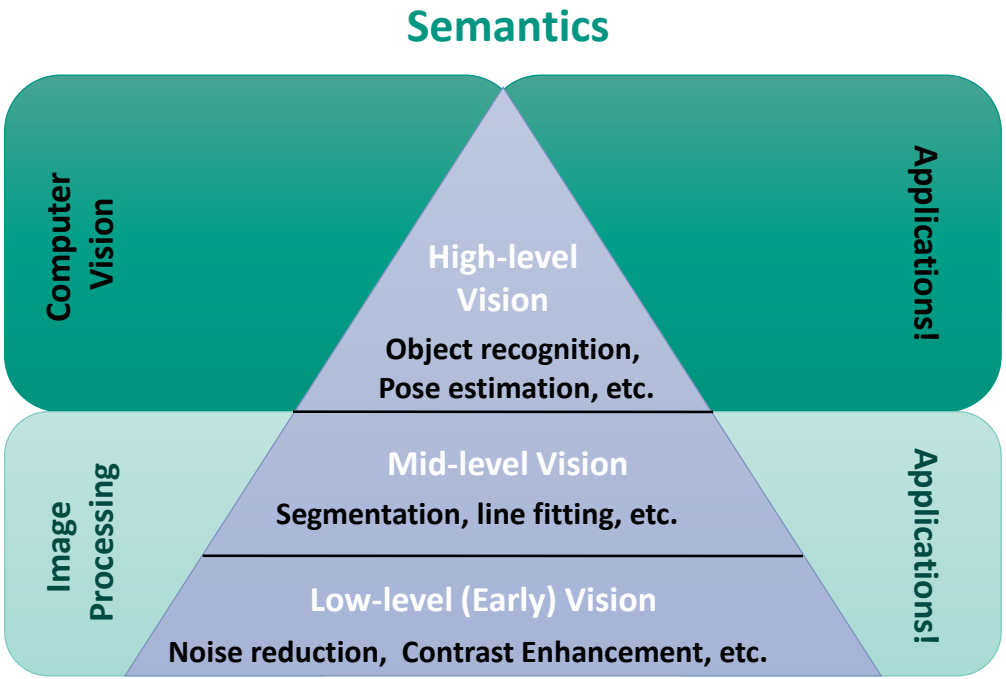


Was ist Computer Vision?





Was ist Computer Vision?



Robotersehen (Robot Vision)

Karlsruher Institut für Technologie

Was ist das Problem?

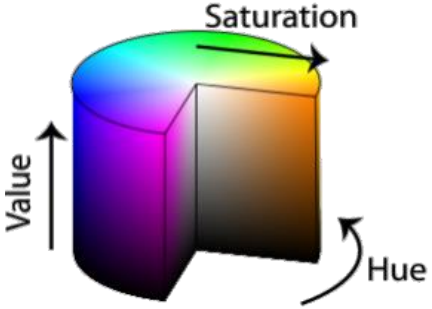
- Wie können wir einem Roboter das „Sehen“ ermöglichen?

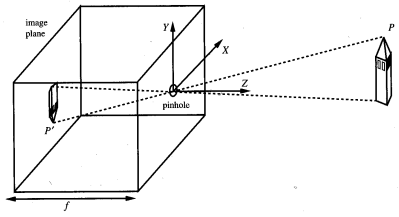
Warum ist ***Robotersehen*** schwer?

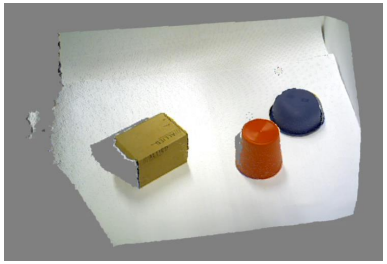
- Computer Vision ist nicht nur Bildübertragung von Kamera auf den Computer sondern auch deren **Verarbeitung** und **Interpretation (Extraktion von Wissen aus Bildern)**
- Bei der Übertragung (3D Welt → 2D Bild) geht eine Dimension verloren
- **Robotersehen:**
  - **Dynamik:** Bewegung in der Szene bzw. in der Kamera
  - **Verdeckungen**: Ein Objekt kann z.B. nicht vollständig sichtbar sein
  - Interpretation abhängig von der aktuellen **Aufgabe**
  - **Photometrie:** Abhängigkeit von Beleuchtung und Materialeigenschaften

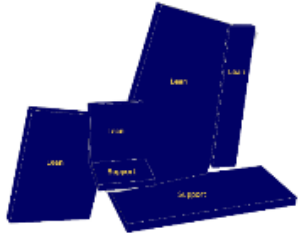
Inhalt

- Bilderzeugung
  - Bildrepräsentation in 2D
  - Bildrepräsentationen in 3D
  - Kameramodell
- Operationen auf Bildern
  - Filteroperationen
  - Morphologische Operatoren
- Merkmalsextraktion und Mustererkennung
  - Segmentierung
  - Canny-Kantendetektor
  - Visual Servoing
  - Registrierung von Punktwolken
- Beispielanwendungen H²T



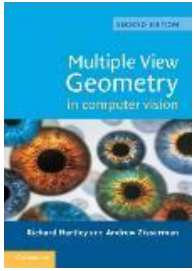


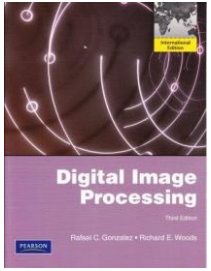


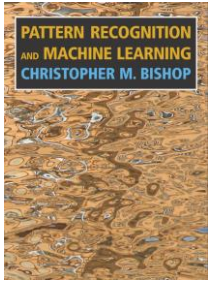


Literatur

- **Multiple View Geometry in Computer Vision**; R. Hartley und A. Zisserman
- **Digital Image Processing**, Rafael C. Gonzalez and Richard E. Woods
- **Automatische Sichtprüfung**; J. Beyerer, F. Puente León und C. Frese
- **Pattern Recognition and Machine Learning;** Christopher M. Bishop; Download
- **Computer Vision – Das Praxisbuch**; Pedram Azad, Tilo Gockel und Rüdiger Dillmann
- **Computer Vision: Algorithms and Applications**; Richard Szeliski (http://szeliski.org/Book)







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
  - Image processing, Feature matching

Motivation

- Das Sehvermögen ermöglicht die Wahrnehmung der Umwelt

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

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

Bildserie: "Peppers" Testbild 4.2.07 der USC-SIPI Bilddatenbank im Original und mit darauf angewandten Filtern

Testbilder

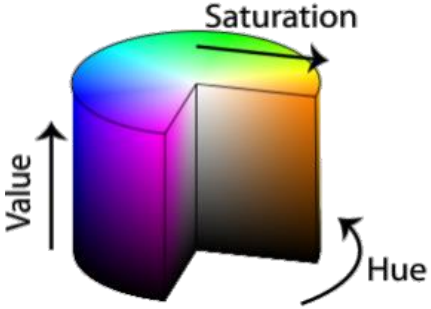
Die USC-SIPI Bilddatenbank wurde erstmals 1977 herausgegeben, um Forschung in der Bildverarbeitung zu unterstützen. Das “Peppers”-Bild ist eines der Bilder der Datenbank.

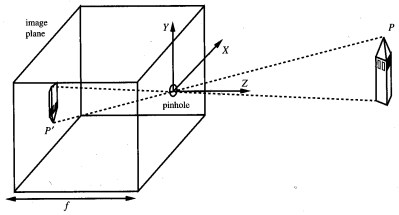
Frühe Publikationen verwendeten daraus oft das “Lena”-Bild, das aus dem Magazin “Playboy” stammt. Das Bild steht daher im Gegensatz zu Zielen wie Gleichstellung und Respekt, sodass von seiner Verwendung abgeraten wird.

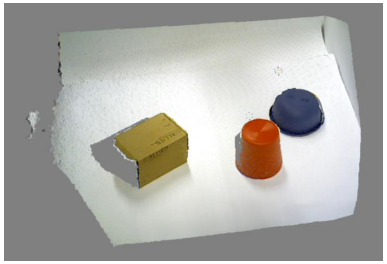
[1] USC-SIPI image database, https://sipi.usc.edu/database/
[2] On alternatives to Lenna. Journal of Modern Optics, Jul. 2017
[3] A note on the Lena image. Nature Nanotech, vol. 13, no. 12, Art. no. 12, 2018. (“affects all Nature Research journals”)

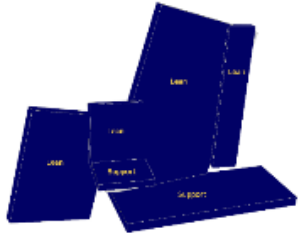
Inhalt

- Bilderzeugung
  - **Bildrepräsentation in 2D**
  - Bildrepräsentationen in 3D
  - Kameramodell
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung









Bildrepräsentation

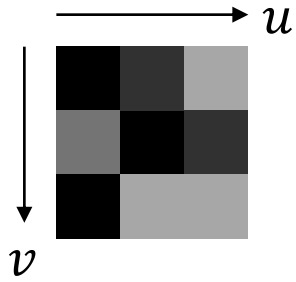
- Bilder müssen im Rechner repräsentiert werden
- Ein Bild ist ein 2D Gitter von diskreten Punkten (**Pixel**)
- **Bildkoordinaten** (hier):
  - u (horizontal)
  - v (vertikal)
  - Ursprung ist oben links
  - Einheiten: Pixel
- Die **Farbe** eines Bildpunktes kann auf unterschiedliche Weise repräsentiert werden
  - **Graustufenbilder**: Für jedes Pixel wird ein Helligkeitswert abgelegt: normalerweise **ein Byte pro Pixel**, i.A. Werte in $[0, 255]$
  - **Farbbilder**: Für jedes Pixel wird ein separater Wert für z.B. **R** (Rot), **G** (Grün) und **B** (Blau) abgelegt ➔ **3 Byte pro Pixel**

Bildrepräsentation - Monochrombild

Monochrombild: Diskrete Funktion

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

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

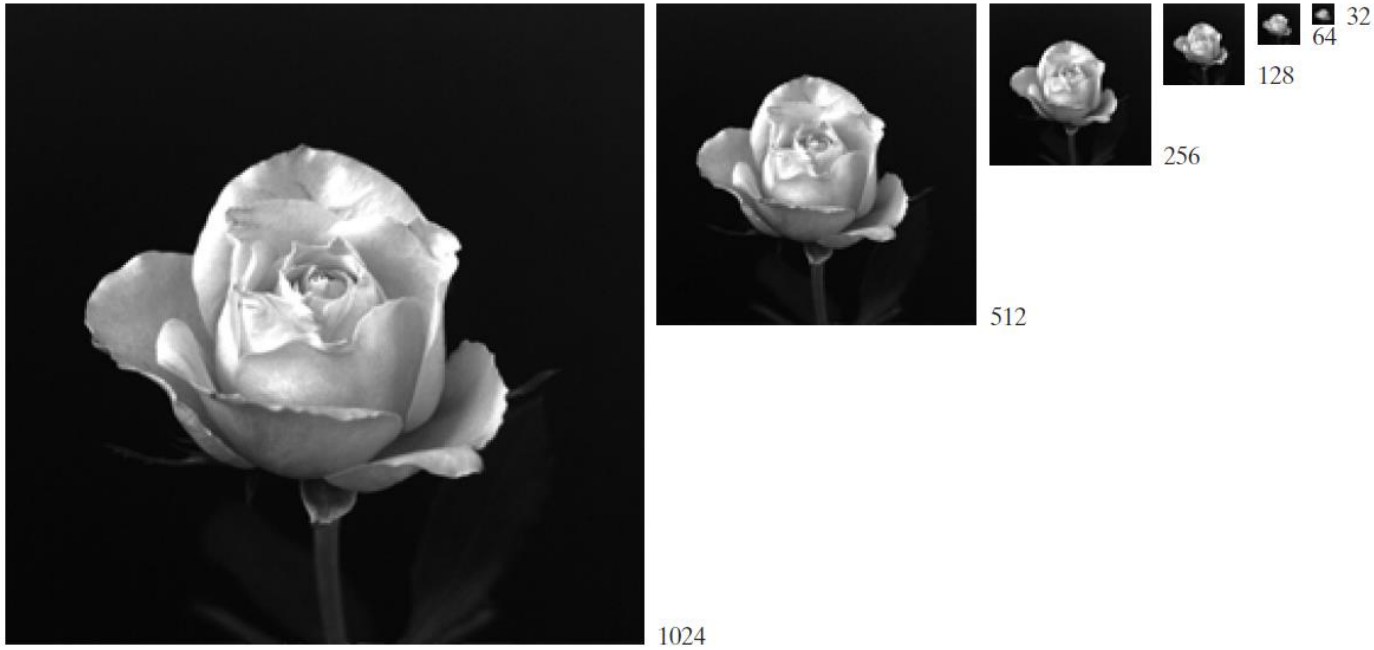


0 für schwarz
255 für weiß

Üblich: $q = 255$
$n = 640, \quad m = 480$ (VGA)
$n = 1920, \; m = 1080$ (1080p „Full HD“)
$n = 3840, \; m = 2160$ (2160p „Ultra HD“ oder 4K UHD)

Bildrepräsentation - Auflösung

- Die Auflösung gibt an, wie viel Pixel im Bild vorhanden sind



Bildrepräsentation - Farbbild

- Verschiedene Farbmodelle für unterschiedliche Anwendungen
- Klassifikation nach Farbraum

Beispiele:

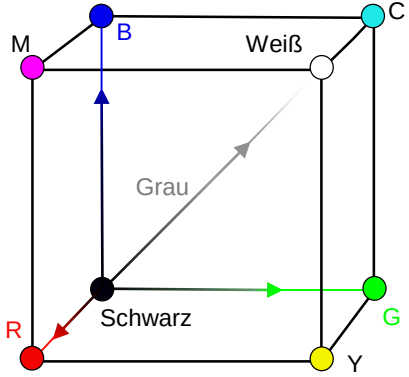
- RGB-Modell (**R**ot-**G**rün-**B**lau-Modell): speziell für Monitore (früher: Phosphor-Kristalle)

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

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

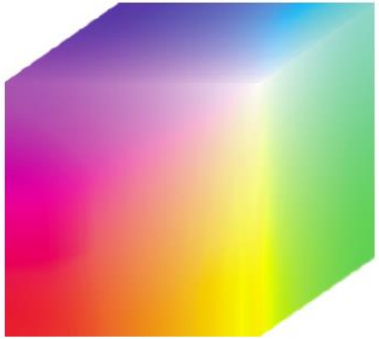
Bildrepräsentation - RGB Farbraum

- 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



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

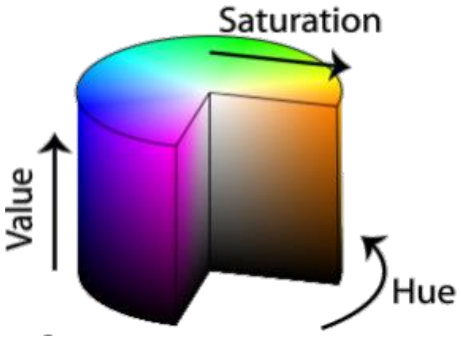
$$(u, v) \mapsto Img(u, v) = (r, g, b)^T$$



24-Bit RGB Farbwürfel

Bildrepräsentation – HSI (HSV) Farbraum

- *Hue* (Farbnuance/Farbton), *Saturation* (Sättigung), *Intensity/Value* (Lichtintensität/Helligkeit)
- Kodiert die 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 $R = G = B = 0$



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

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

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

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

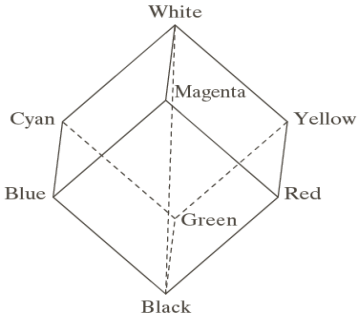
Bildrepräsentation - HSI nach RGB

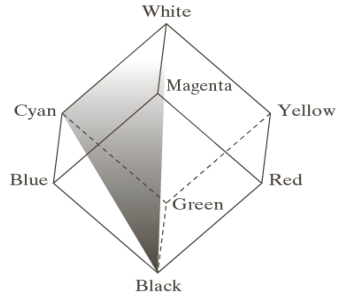
Wie bestimmt man HSI Werte in einem RGB Würfel?

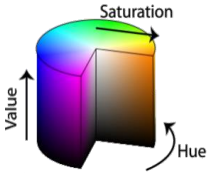
- Im RGB Format enstpricht die ***Helligkeitsachse*** (*Intensity*) der Linie durch die Ecken Schwarz (0,0,0) und Weiß (1,1,1)

- Die ***Sättigung*** *(Saturation)* eines Farbpunkts auf der Sättigungsachse 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.

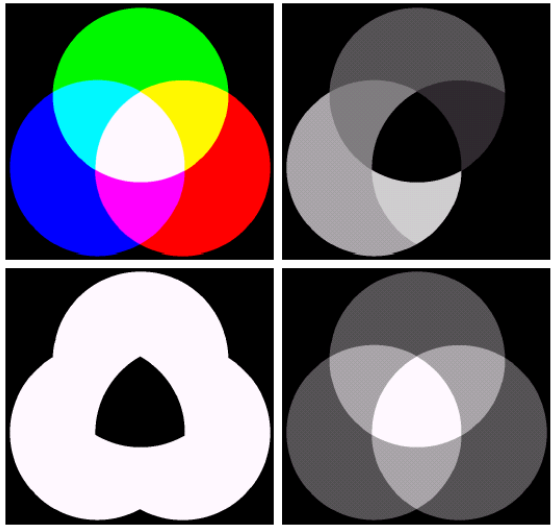






Bildrepräsentation – Unterschiede zu RGB

RGB



Farbnuance

Sättigung

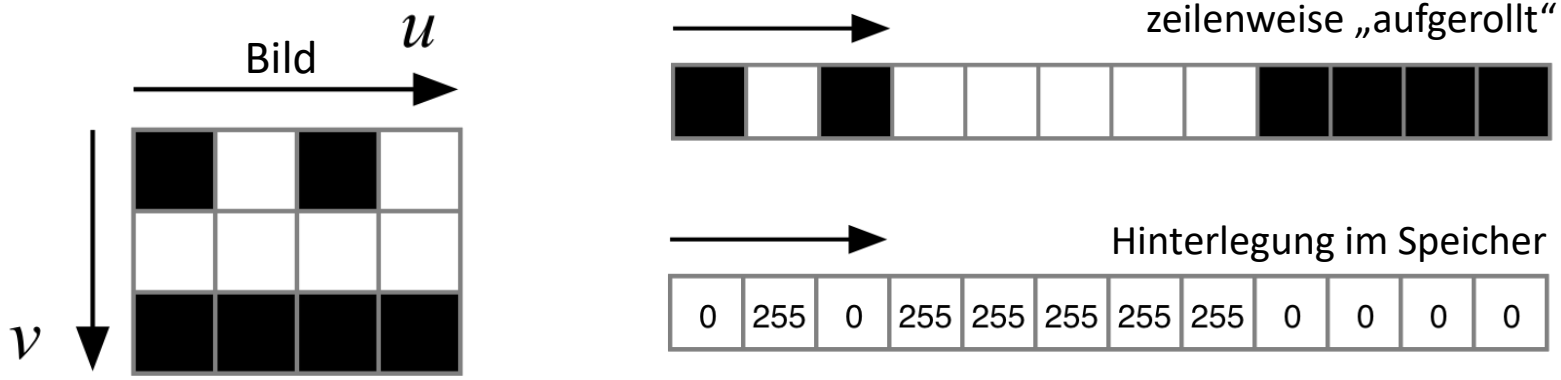
Helligkeit

- 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 - Repräsentation im Speicher

Repräsentation eines 8-Bit Graustufen-Bildes im Speicher

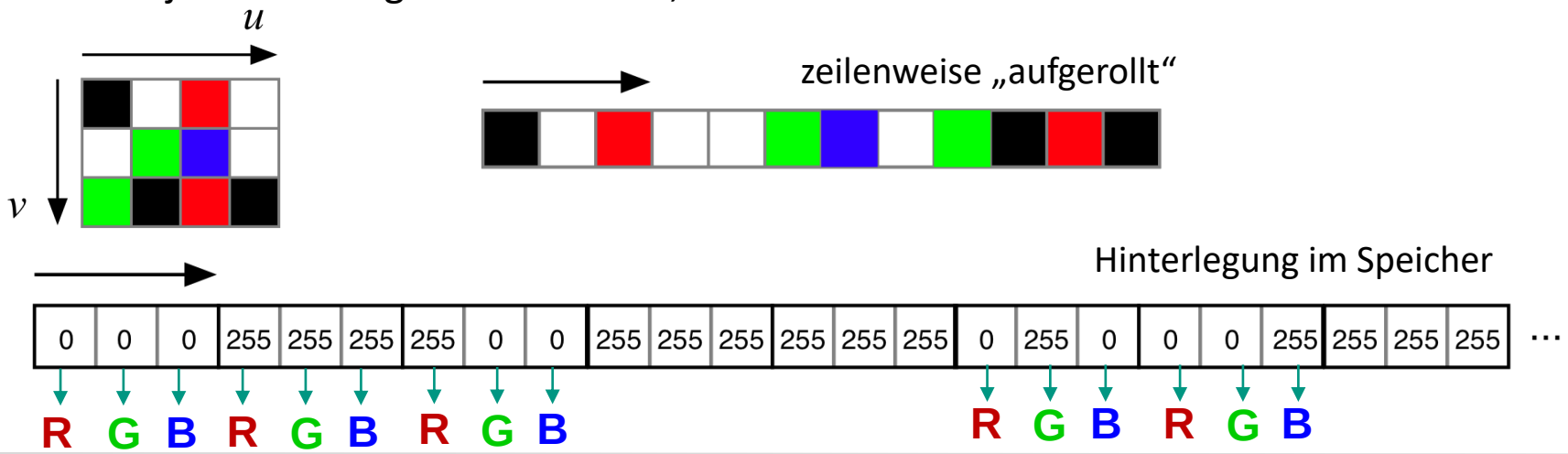
- Pixel werden zeilenweise, linear abgelegt
  - von oben links nach unten rechts
  - Achtung: z.B. bei Bitmaps von unten links nach oben rechts
- Graustufen-Kodierung:
  - Ein Byte pro Pixel
  - 0 schwarz, 255 weiß, dazwischen Graustufen



Bildrepräsentation - Repräsentation im Speicher (2)

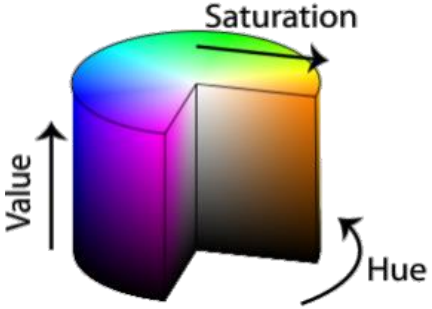
Repräsentation eines $RGB24$ Farbbildes im Speicher

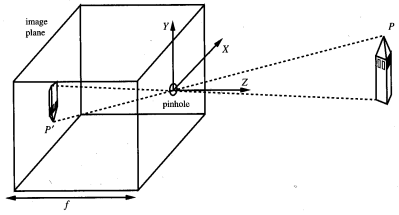
- Pixel werden zeilenweise, wie beim Graustufen-Bild, abgelegt
- Farbkodierung:
  - Drei Bytes pro Pixel
  - Für jeden Kanal gilt: 0 minimale, 255 maximale Intensität

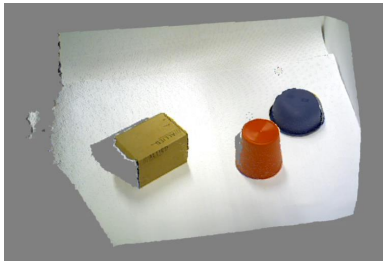


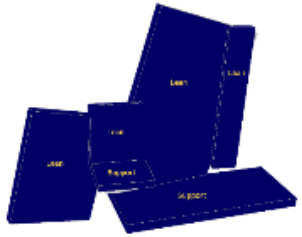
Inhalt

- Bilderzeugung
  - Bildrepräsentation in 2D
  - **Bildrepräsentationen in 3D**
  - Kameramodell
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung





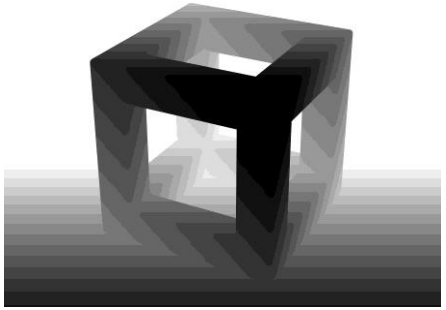




Bildrepräsentation – Tiefenbilder

Tiefenbild:

- Monochrom- und Farbbilder haben keine Informationen über räumliche oder geometrische Verhältnisse im Bild
- Zusätzlich zur visuellen Information kann die Entfernung zum Sensor pro Pixel gespeichert werden



https://en.wikipedia.org/wiki/Depth_map

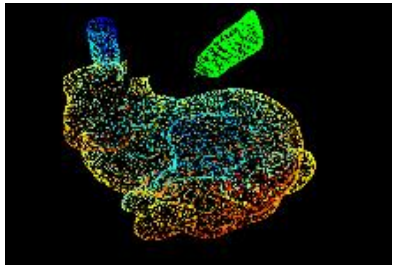
Beispiele:

- $RGB\boldsymbol{D}$-Modell (**R**ed-**G**reen-**B**lue-**D**epth Modell) für Tiefenkameras:

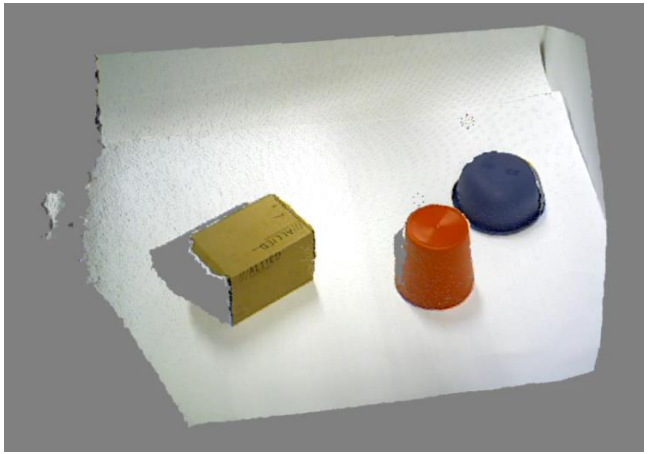
$$Img(u,v) = (r,g,b,d)^T \rightarrow \{d \in \mathbb{R}, (r,g,b) \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$$

Bildrepräsentation – Punktwolken

- Eine **Punktwolke** ist eine **diskrete** Menge von **3D-Punkten** mit einem festen Koordinatensystem.



2D Bild



3D Punktwolke

Bildrepräsentation – Punktwolken (2)

- Punktwolke $P = \{(X, C) \big| X \in \mathbb{R}^3, C \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$

  $X = (x, y, z)$ Ortsinformation

  $C = (r, g, b)$ Farbinformation

  Es können weitere (Sensor-)informationen abgespeichert werden (z.B. Labels, Normale, …)

- Zwei verschiedene Arten der Repräsentation

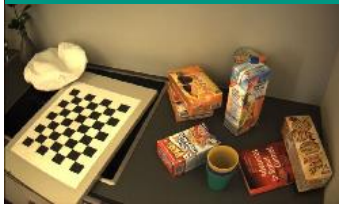
  - **Organisierte/geordnete** Punktwolke
    (2D Array Format, Größe muss vorab bekannt sein)
  - **Unorganisierte/ungeordnete** Punktwolke
    (Vector Format)

Beispiel – Bildrepräsentation bei Roboception-Kamera

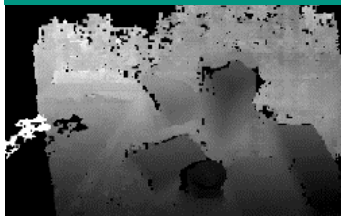
- Roboception Kamera (rc visard 160): Stereo-Kamera System mit *160 mm* Baseline (Abstand der beiden Kameras)
- **Kamerabilder**: Bildauflösung 1280 x 960 Pixel (≈1.3 MegaPixel).
- **Tiefenbild:** Distanz der Umgebung wird aus dem linken und rechten Kamerabild relativ zum Sensor berechnet (Stereovision, später)
- **Konfidenzbild (Spezialfall):** Abschätzung für das *Vertrauen* in die Messwerte des Tiefenbilds.

https://roboception.com

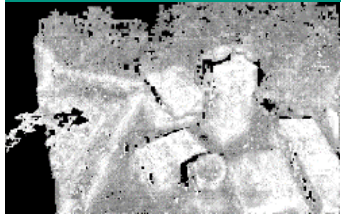
Linke Kamera (RGB)



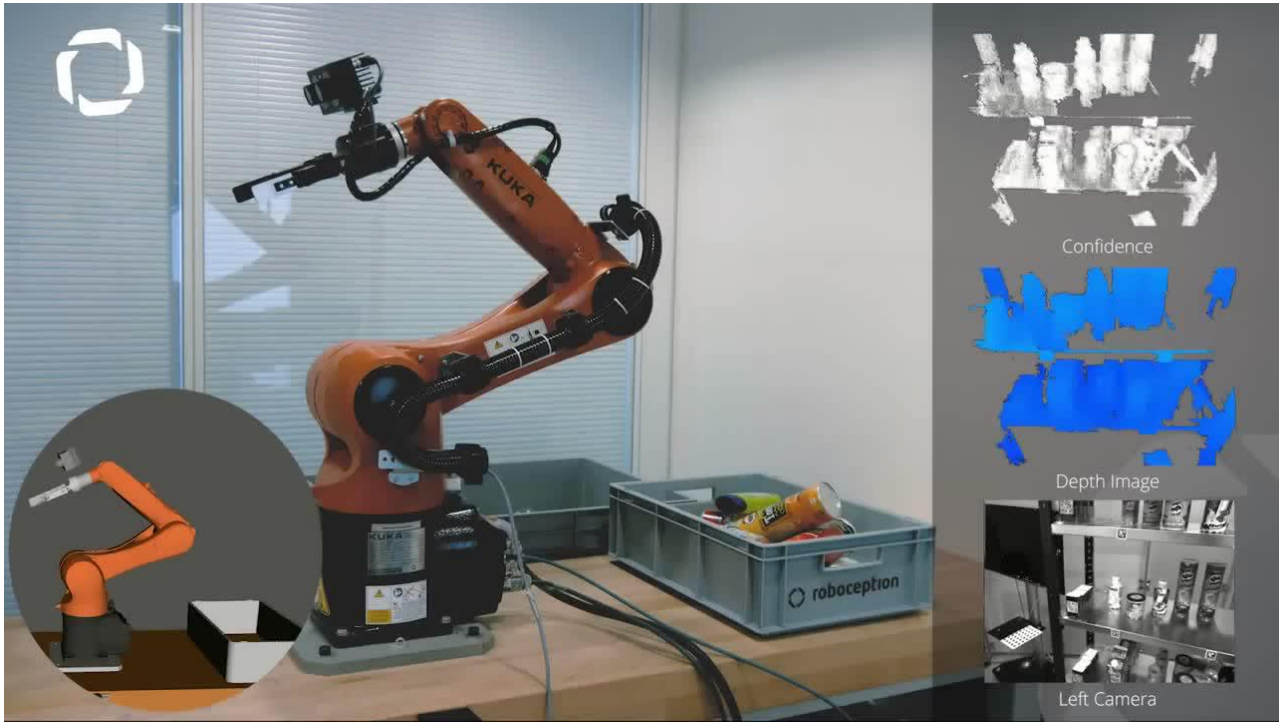
Tiefenbild



Konfidenzbild



Roboception: rc visard 65 – Bin Picking Application



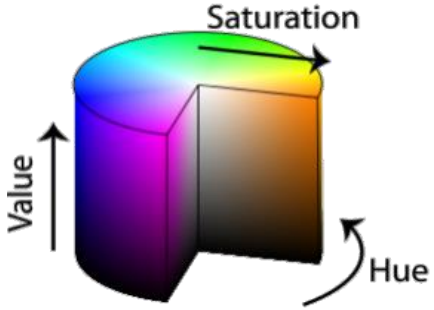
https://www.youtube.com/watch?v=-IeZbn1aA8Q

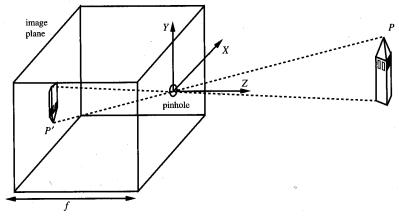
Microsoft Azure Kinect

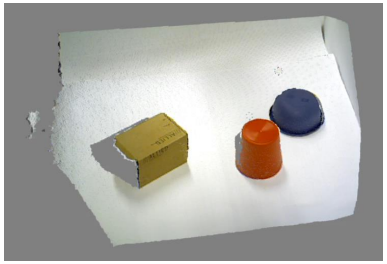
https://www.youtube.com/watch?v=jJglCYFiodI

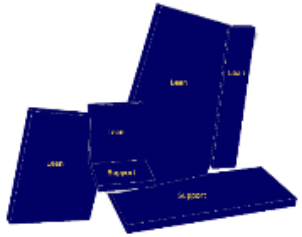
Inhalt

- Bilderzeugung
  - Bildrepräsentation in 2D
  - Bildrepräsentationen in 3D
  - **Kameramodell**
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung



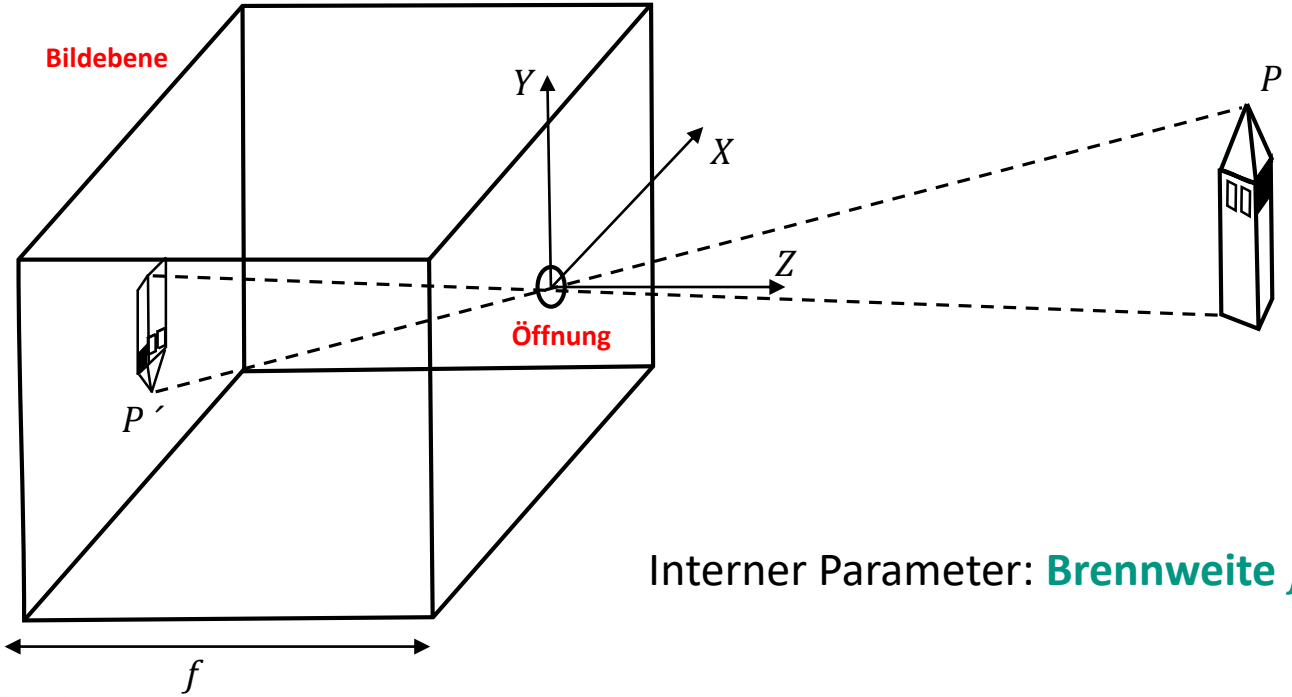






Bildgenerierung – Lochkamera

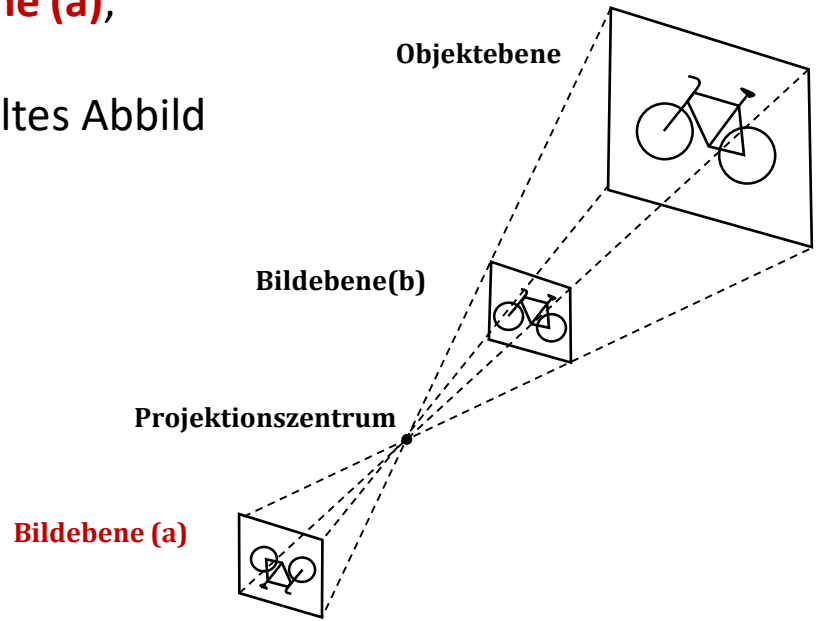
- Einfachstes Modell: **Lochkamera-Modell**



Interner Parameter: **Brennweite f („Fokalabstand“)**

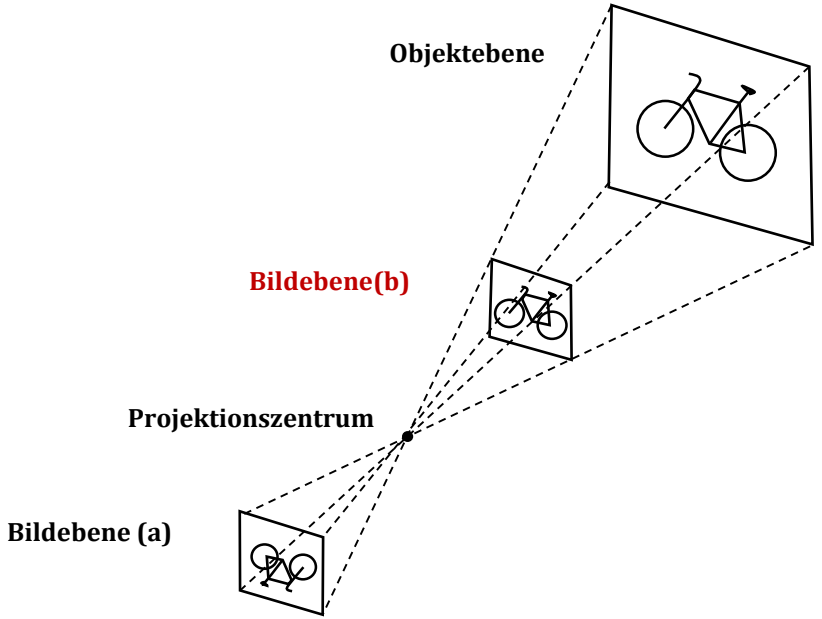
Bildgenerierung – Klassisches Lochkamera Modell

- Klassisches Lochkamera-Modell
  - **Projektionszentrum liegt vor der Bildebene (a)**, also zwischen Szene und Bildebene
  - Dadurch: Horizontal und vertikal gespiegeltes Abbild



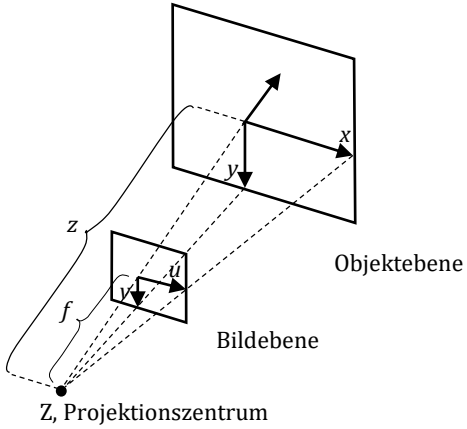
Bildgenerierung – Lochkamera in Positivlage

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



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. Hier Einheit: mm
- **Weltkoordinatensystem:** 3D-Basiskoordinatensystem in der Welt.

Einheit in dieser Vorlesung: mm

Bildgenerierung – Lochkamera-Projektion

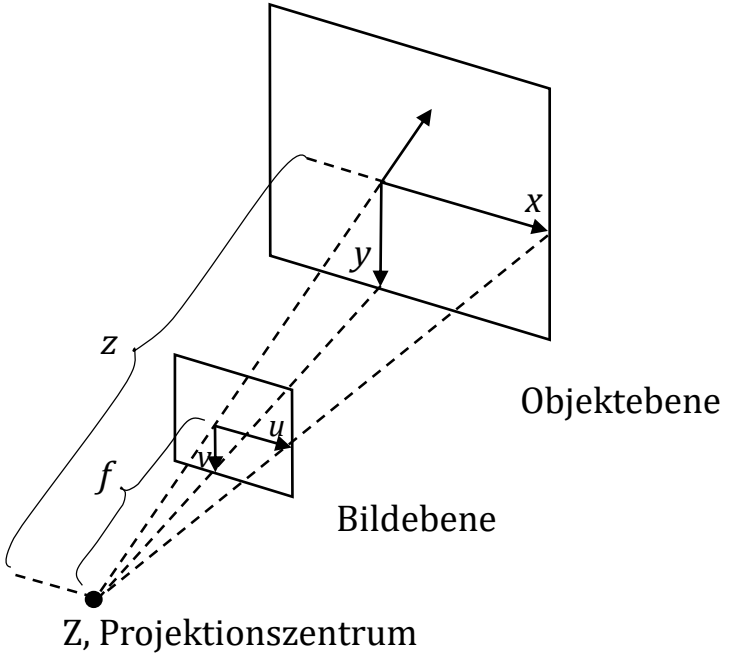
- Zweiter Strahlensatz zur Projektion eines Szenenpunktes (x, y, z) auf einen Bildpunkt (u, v):

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

- Bei der Projektion geht die z-Komponente verloren!

- Rückprojektion:

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



Bildgenerierung – Vom RGBD-Bild zur Punktwolke

- Mit einer Lochkamera ist es nicht ohne weiteres möglich Korrespondenzen von einem Pixel zu einem Punkt im Raum zu bestimmen

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

- Wenn der Sensor in der Lage ist Tiefenwerte d für jedes Pixel zu bestimmen (z.B. LiDaR, Stereo-Kamera, ToF, …), kann eine einfache Korrespondenz hergestellt werden

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{d}{f} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Erweitertes Kameramodell

- Bisheriges Klassisches Lochkameramodel ist zu einfach für Anwendungen
- **Erweiterung des Modells**
  - **Unabhängige Brennweiten f_x and f_y** in u und v Richtung (**rechteckige Pixel**)

$$f = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

  - Hauptpunkt (c_x, c_y), ist nicht identisch mit dem Ursprung des Kamerakoordinatensystems
  - Projektion von Kamera- zu Bildkoordinaten kann nun erweitert werden zu:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{z} \begin{pmatrix} f_x \cdot x \\ f_y \cdot y \end{pmatrix}$$

- *Vorher*: Einfaches Lochkamera-Modell:

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

Erweitertes Kameramodell (2)

- Unter Verwendung homogener Koordinaten kann dies in Form einer Matrixmultiplikation geschrieben werden:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{z} \begin{pmatrix} f_x \cdot x \\ f_y \cdot y \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Erweitertes Kameramodell (3)

- Unter Verwendung homogener Koordinaten kann dies in Form einer Matrixmultiplikation geschrieben werden:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Erweitertes Kameramodell (4)

- Unter Verwendung homogener Koordinaten kann dies in Form einer Matrixmultiplikation geschrieben werden:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{K}} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- K ist die **Kalibriermatrix der Kamera**

$$\begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Erweitertes Kameramodell (5)

$$\begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = K^{-1} \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix}$$

$$K^{-1} = \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{pmatrix}$$

Intrinsische und Extrinsische Kameraparameter

- Die Parameter in der Kalibriermatrix K werden als **intrinsische Parameter** bezeichnet.
- Die Beziehung zwischen Kamera-Koordinatensystem und Weltkoordinatensystem wird durch die **extrinsischen Parameter** beschrieben: **eine Rotation R und eine Translation t im Raum.**
- R und t geben die Transformation vom Weltkoordinatensystem zum Kamerakoordinatensystem so an, dass für ein Weltpunkt x_w die Kamerakoordinaten x_c bestimmt werden durch:

$$x_c = R\, x_w + t$$

- Durch die Kombination von extrinsischen und intrinsischen Parametern wird die Projektion eines Punktes von Weltkoordinaten auf Bildkoordinaten durch die **Projektionsmatrix** P gegeben:

$$P = K \cdot (R \mid t)$$

Kamerakalibrierung

- **Kamerakalibrierung** bedeutet die Bestimmung der extrinsischen und intrinsischen Parameter der Kamera
- Dies erfordert **mindestens 6 Korrespondenzen** zwischen nicht-koplanaren Weltpunkten und ihren Projektionen in die Bildebene. Für jede Korrespondenz gilt die Beziehung

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \text{ mit } P = (K \cdot R \mid K \cdot t) = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{pmatrix}$$

- Hier gilt es, die unbekannten Parameter p_1 bis p_{12} zu bestimmen

Kamerakalibrierung (2)

- Die Gleichung kann aufgelöst werden nach

$$u = \frac{p_1 X + p_2 Y + p_3 Z + p_4}{p_9 X + p_{10} Y + p_{11} Z + p_{12}}$$

$$v = \frac{p_5 X + p_6 Y + p_7 Z + p_8}{p_9 X + p_{10} Y + p_{11} Z + p_{12}}$$

- Da homogene Koordinaten verwendet werden, ändert die Multiplikation von P mit einem Faktor (ungleich Null) nichts. Daher kann die Gleichung normalisiert werden und z.B. $p_{12} = 1$ gesetzt werden (also P mit $1/p_{12}$ multiplizieren)

Kamerakalibrierung (3)

- Die beiden Gleichungen lassen sich weiter umstellen zu

$$p_1X + p_2Y + p_3Z + p_4 = u \cdot p_9X + u \cdot p_{10}Y + u \cdot p_{11}Z + u$$
$$p_5X + p_6Y + p_7Z + p_8 = v \cdot p_9X + v \cdot p_{10}Y + v \cdot p_{11}Z + v$$

$$u = p_1X + p_2Y + p_3Z + p_4 - u \cdot p_9X - u \cdot p_{10}Y - u \cdot p_{11}Z$$
$$v = p_5X + p_6Y + p_7Z + p_8 - v \cdot p_9X - v \cdot p_{10}Y - v \cdot p_{11}Z$$

Kamerakalibrierung (4)

- Jede Korrespondenz zwischen Welt- und Bildpunkt ergibt **2 lineare Gleichungen**.
- Mit Hilfe von $n \geq 6$ Korrespondenzen ergibt sich das lineare Gleichungssystem $A\,x = b$ mit

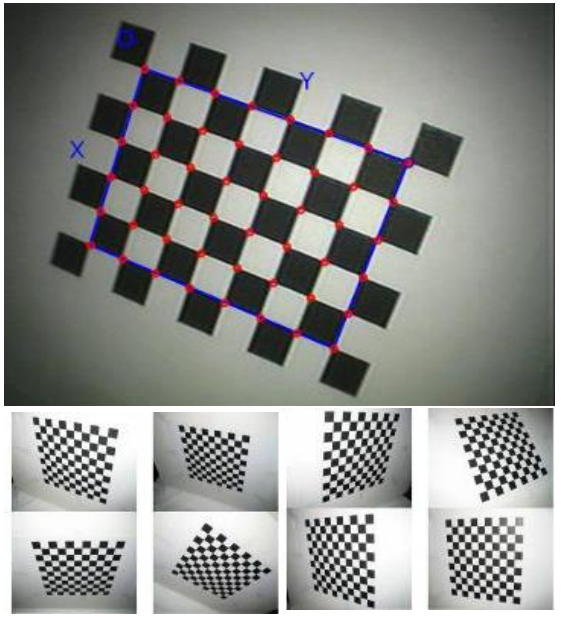
$$\underbrace{\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_{10} \\ p_{11} \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} u_1 \\ v_1 \\ \cdots \\ u_n \\ v_n \end{pmatrix}}_{b}$$

Kamerakalibrierung (5)

- Jede Korrespondenz zwischen Welt und Bildpunkt ergibt 2 lineare Gleichungen.
- Mit Hilfe von n Korrespondenzen ergibt sich das lineare Gleichungssystem $A\,x = b$ mit

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n \end{pmatrix}$$

$$x = \begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_{10} \\ p_{11} \end{pmatrix} \qquad b = \begin{pmatrix} u_1 \\ v_1 \\ \cdots \\ u_n \\ v_n \end{pmatrix}$$



Kamerakalibrierung (6)

- Die optimale Lösung x^* mit Hilfe der Kleinste-Quadrate-Methode für ein solches **überbestimmtes lineares Gleichungssystem** $(Ax = b)$ ergibt sich aus der Lösung von

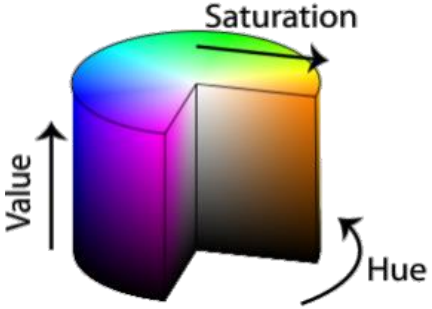
$$A^T A\, x^* = A^T b$$

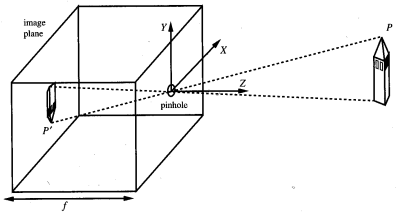
$$x^* = (A^T A)^{-1} A^T b$$

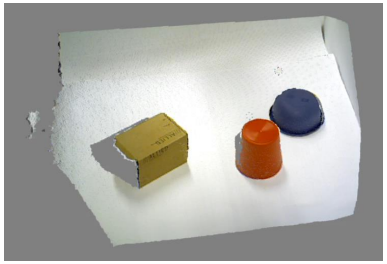
$(A^T A)^{-1} A^T$: **Moore-Penrose Pseudoinversen**

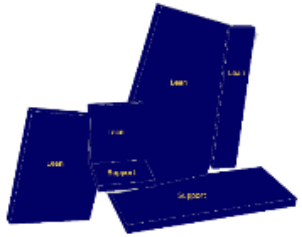
Inhalt

- Bilderzeugung
- Operationen auf Bildern
  - **Filteroperationen**
  - Morphologische Operatoren
- Merkmalsextraktion und Mustererkennung









Filteroperationen

- **Filter** in der Bildverarbeitung
  auch *spatial filters, spatial masks, kernels, windows*
- Ein Filter ist eine **Operation** auf einer Menge benachbarter Pixel, also
  - vordefinierte Operation
  - Nachbarschaft (meist ein kleines Rechteck)
- Ein Filter wird auf alle Pixel des Bildes angewendet
  - Berechnung eines neuen Pixelwertes durch Anwendung der Filteroperation unter Berücksichtigung der Nachbarschaftspixel
- Eigenschaften eines **linearen Filters**

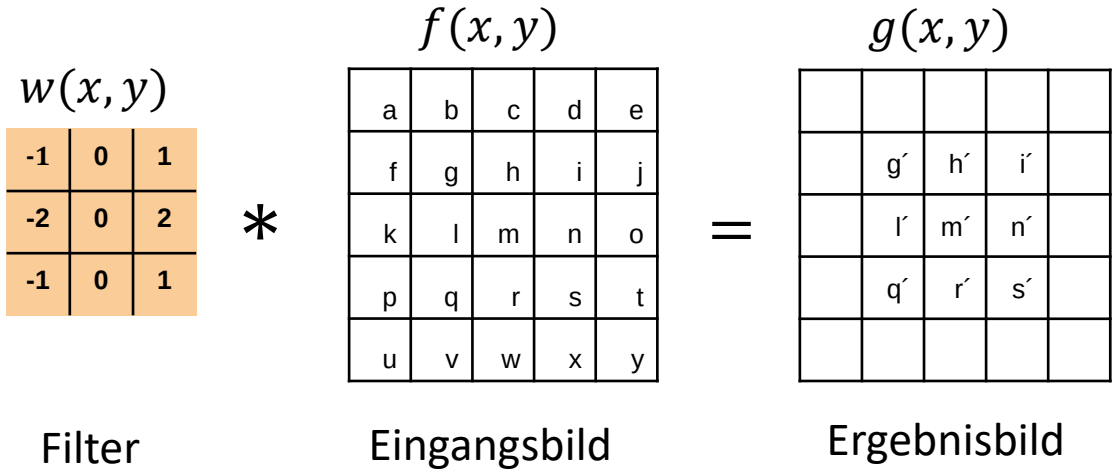
$$f(x+y) = f(x) + f(y) \qquad \text{additiv}$$

$$f(\alpha x) = \alpha f(x) \qquad \text{homogen}$$

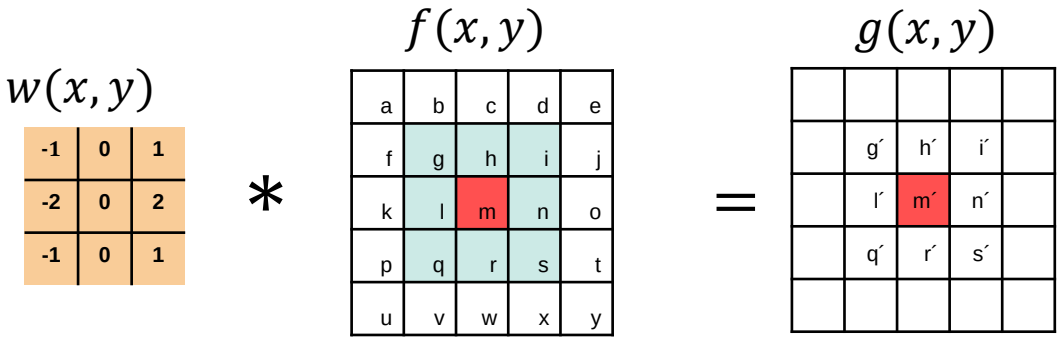
Das Filter

- Die Anwendung eines Filters auf ein Bildelement wird dadurch realisiert, dass die Filtermaske auf dieses Bildelement gelegt wird, die Maskenwerte mit den darunter liegenden Pixelwerten multipliziert werden und das Ergebnis aufsummiert wird.

- Beispiel:

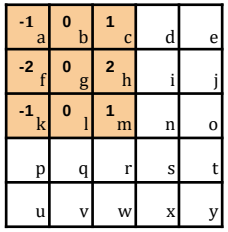


Das Filter (2)

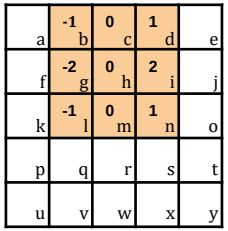


$$m' = -1 \cdot g + 0 \cdot h + 1 \cdot i - 2 \cdot l + 0 \cdot m + 2 \cdot n - 1 \cdot q + 0 \cdot r + 1 \cdot s$$

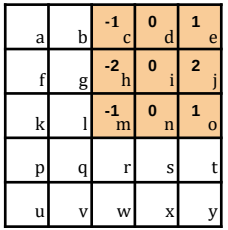
- Beispiel:



$$g' = -a + c - 2f + 2h - k + m$$

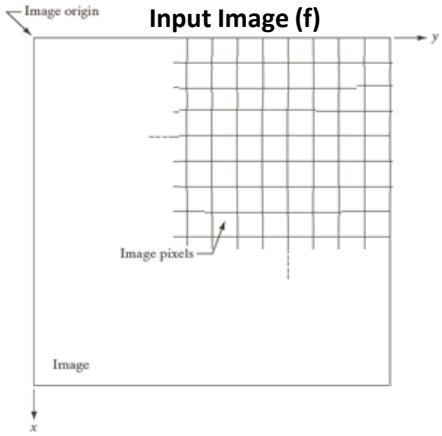


$$h' = (d + 2i + n) - (b + 2g + l)$$



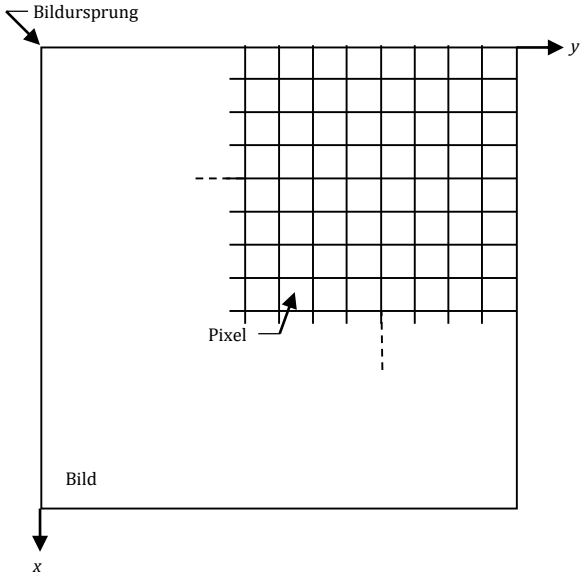
$$i' = (e + 2j + o) - (c + 2h + m)$$

Filter über jedes Pixel

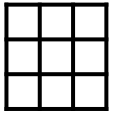


Filter

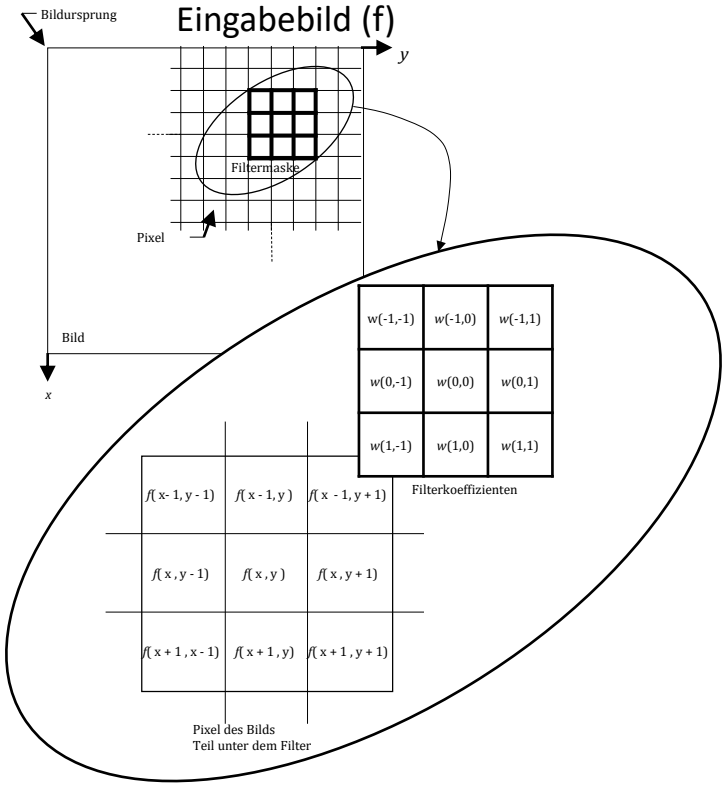
Filteroperation



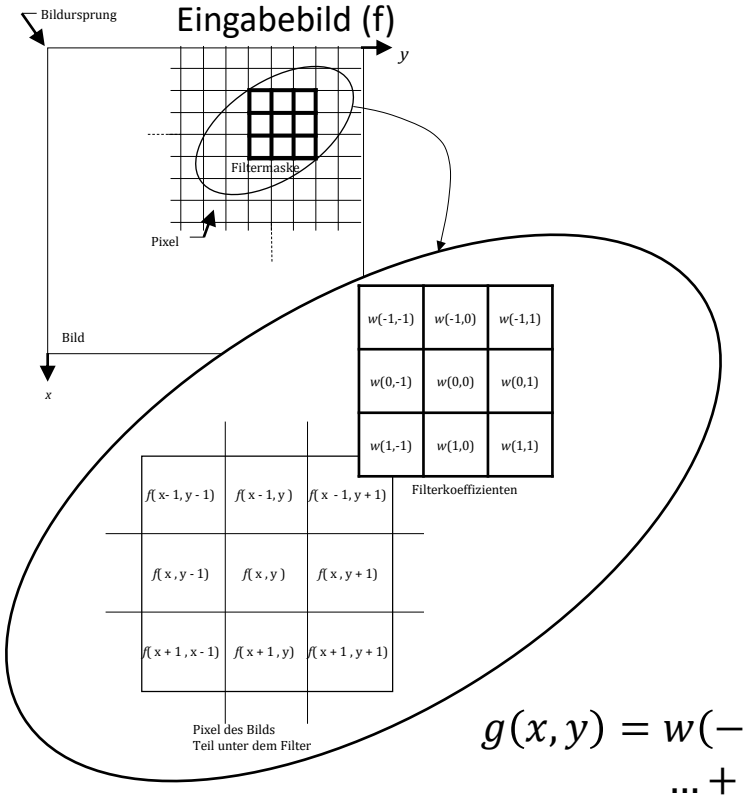
Filter

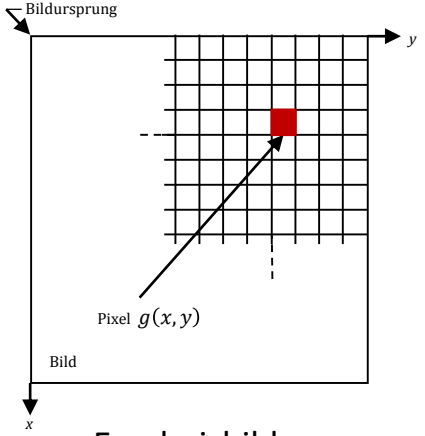


Filteroperation



Filteroperation – Korrelation



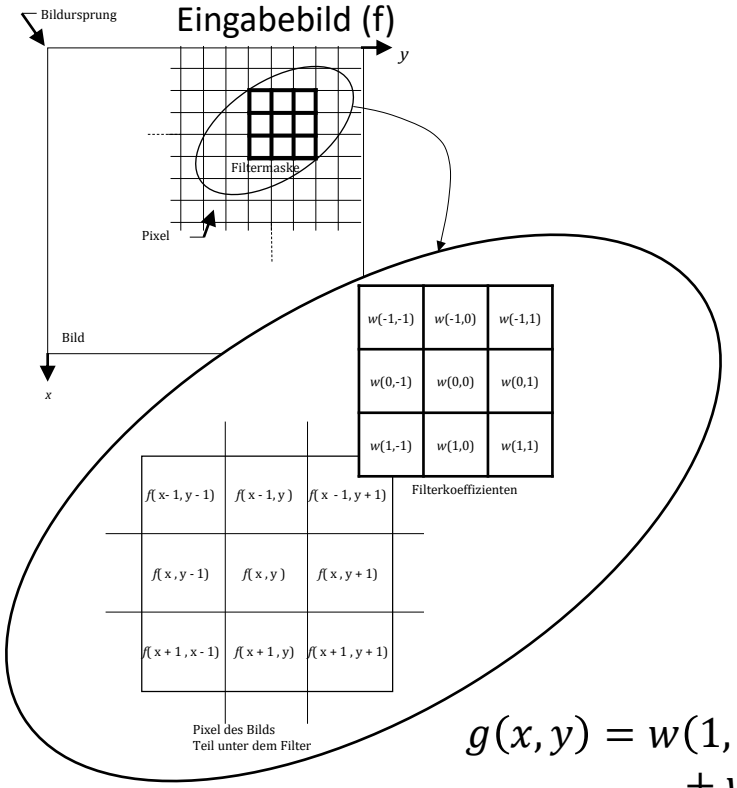


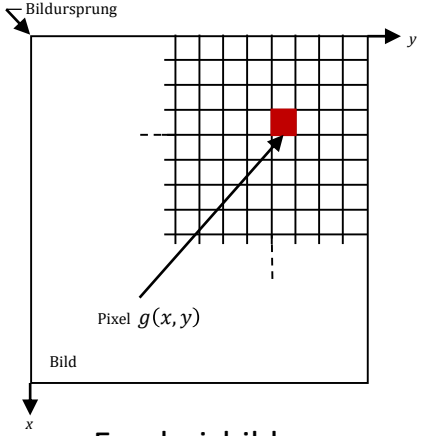
Ergebnisbild:

$$\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\cdot f(x+s,y+t)$$

$$g(x,y) = w(-1,-1)\cdot f(x-1,y-1) + w(-1,0)\cdot f(x-1,y) + \\ \ldots + w(0,0)\cdot f(x,y) + \cdots + w(1,1)\cdot f(x+1,y+1)$$

Filteroperation – Faltung





Ergebnisbild:

$$\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\cdot f(x-s,y-t)$$

$$g(x,y) = w(1,1)\cdot f(x-1,y-1) + w(1,0)\cdot f(x-1,y) + \\ \ldots + w(0,0)\cdot f(x,y) + \cdots + w(-1,-1)\cdot f(x+1,y+1)$$

Filteroperationen – Grundlagen

- **Korrelation** ist eine Filteroperation, bei der eine Filtermaske über das Bild bewegt und die Summe der Produkte an jedem Bildpunkt berechnet wird. Korrelation ist eine Funktion der Verschiebung des Filters

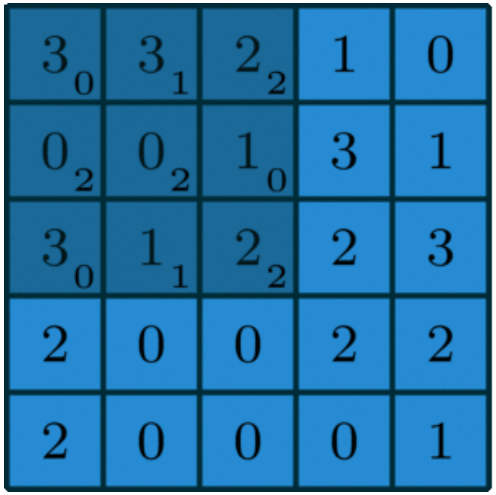
$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$$

- Die **Faltung** ist eine Filteroperation, bei der das Filter zuerst um 180° gedreht wird

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x-s, y-t)$$

Filteroperationen – Bildbereich

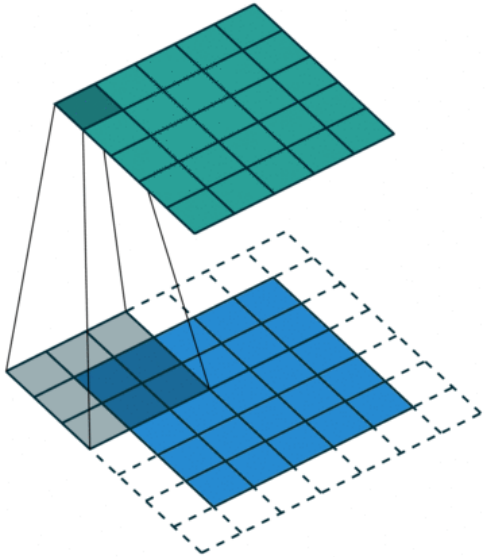
- Standard-Filteroperationen können die Auflösung des Bildes verändern. Wenn nur gültige Pixel betrachtet werden, wird das **Bild verkleinert** – die Ränder werden abgeschnitten, da die Randpixel nie im Zentrum eines Filters liegen.



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Filteroperationen – Ränder

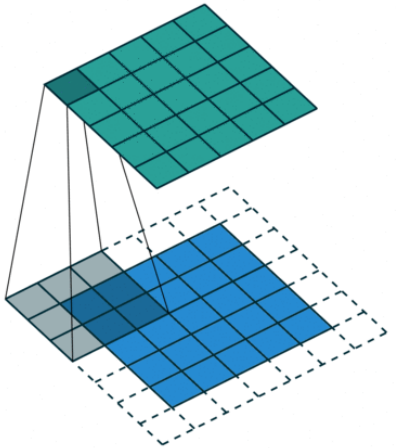
- Um die Größe des ursprünglichen Bildes zu erhalten werden die Ränder oft mit künstlichen Pixeln „aufgefüllt“ → Diese Verfahren werden als **Padding** bezeichnet



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Filteroperationen – Ränder (2)

- Was passiert an den Rändern?
  - Konstanter Wert (**Constant**), z.B. 0:
    Bild wird an den Rändern auf 0 gesetzt.
  - Umschlingen (**Wrap**):
    Bild wird *„fortgesetzt“*

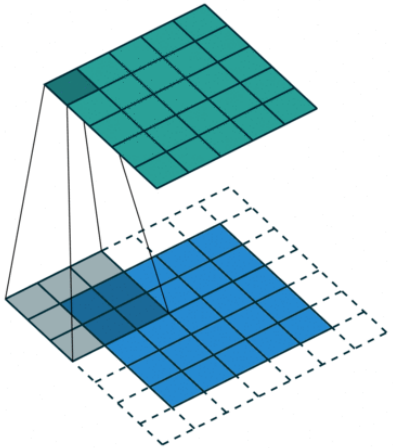


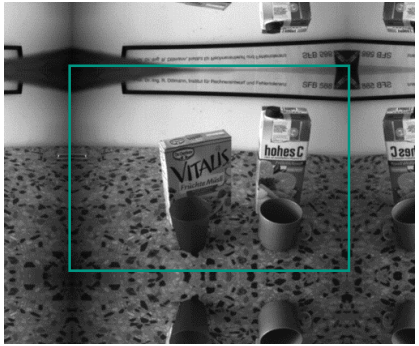
Constant

Wrap

Filteroperationen – Ränder (3)

- Was passiert an den Rändern?
  - Spiegeln (**Mirror**, **Reflect**):
    Bild wird an den Rändern gespiegelt
  - Wiederholen (**Clamp**, **Replicate**):
    Nehme letzten Wert





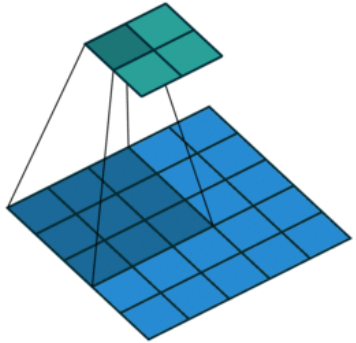
Mirror

Clamp

Filteroperationen – Schrittweite

- Oft ist es nützlich die Auflösung des ursprünglichen Bildes zu verkleinern, um den Informationsgehalt zu verändern. Dies wird erreicht indem die **Schrittweite des Filters** verändert wird.
- **Beispiel:**

  Die Höhe und Breite des Bildes wird ungefähr halbiert, wenn nur jedes 2. Pixel für die Filteroperation betrachtet wird.



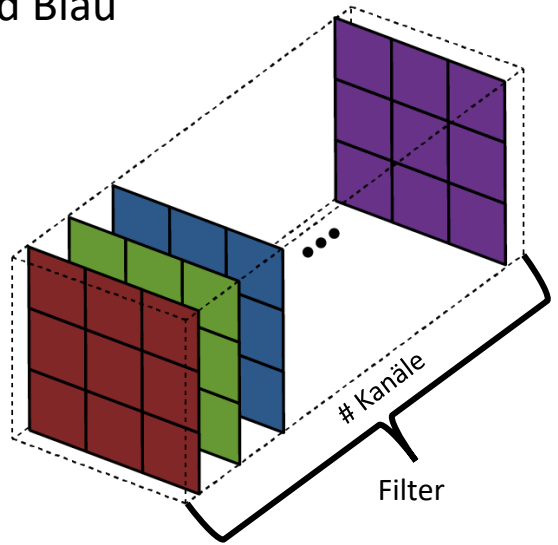
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Filteroperationen - Anwendung auf Farbbilder

- Die Anwendung von Filtern auf Graustufenbilder ist der triviale Fall
  - Das Bild hat nur einen Kanal (0…255 als Grauwert)
- RGB-Bilder sind oft von höherem Interesse als Graustufenbilder
  - Das Bild hat drei Kanäle – jeweils einen für Rot, Grün und Blau

- Generell gilt:

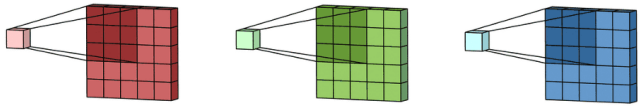
  Jedes Filter hat eine Filtermatrix (**Kernel**) pro Eingabekanal



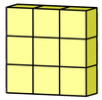
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Filteroperationen - Anwendung auf Farbbilder (2)

- Jeder Kernel wird auf den dazugehörigen Kanal angewendet



- Die Ergebnisse jedes Kernels werden am Ende addiert



- Pro Filter entsteht ein Ausgabekanal:
  - Oft wird zu dem Ausgabewert noch ein Bias-Term addiert (CNNs)

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Filteroperationen - Design

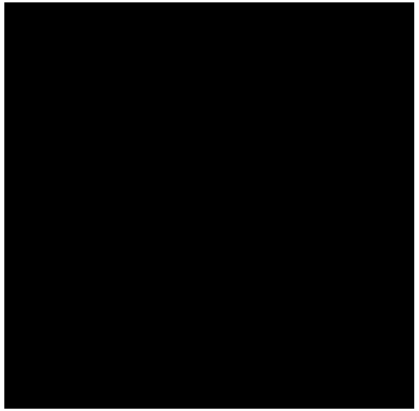
Wie werden **Maskenkoeffizienten** definiert?

- Hängt davon ab, was das Filter tun soll!

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Original

0	0	0
0	0	0
0	0	0



Ergebnis (Löschen)

Filteroperationen - Design (2)

Wie werden **Maskenkoeffizienten** definiert?

- Hängt davon ab, was das Filter tun soll!

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Original

0	0	0
0	1	0
0	0	0

Ergebnis (Identität)

Filteroperationen - Design (3)

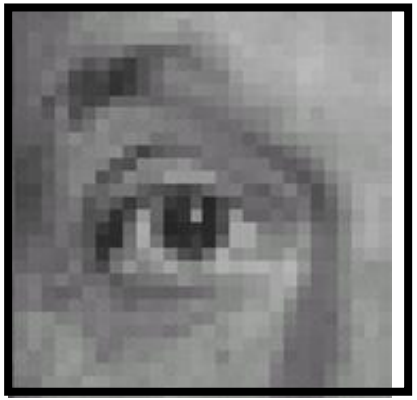
Wie werden **Maskenkoeffizienten** definiert?

- Hängt davon ab, was das Filter tun soll!

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Original

0	0	0
0	0	1
0	0	0



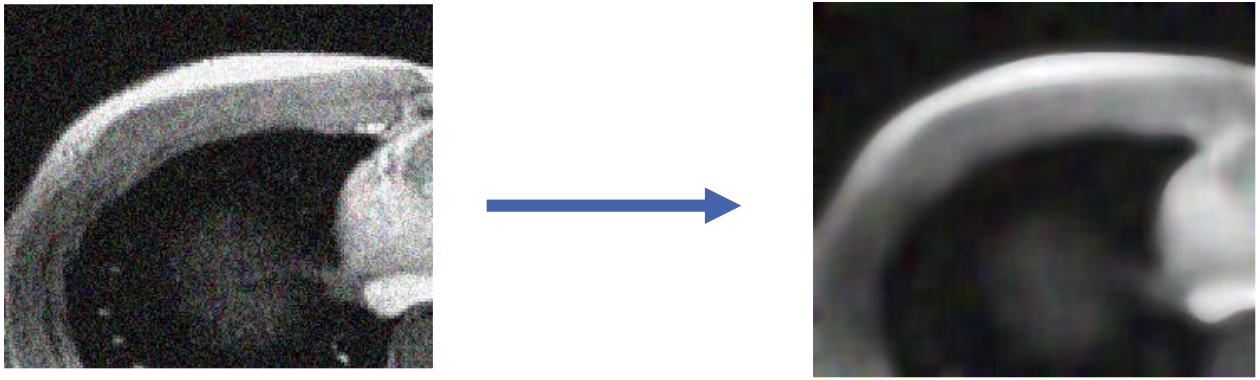
Nach *links*
um 1 Pixel verschoben

Filteroperationen

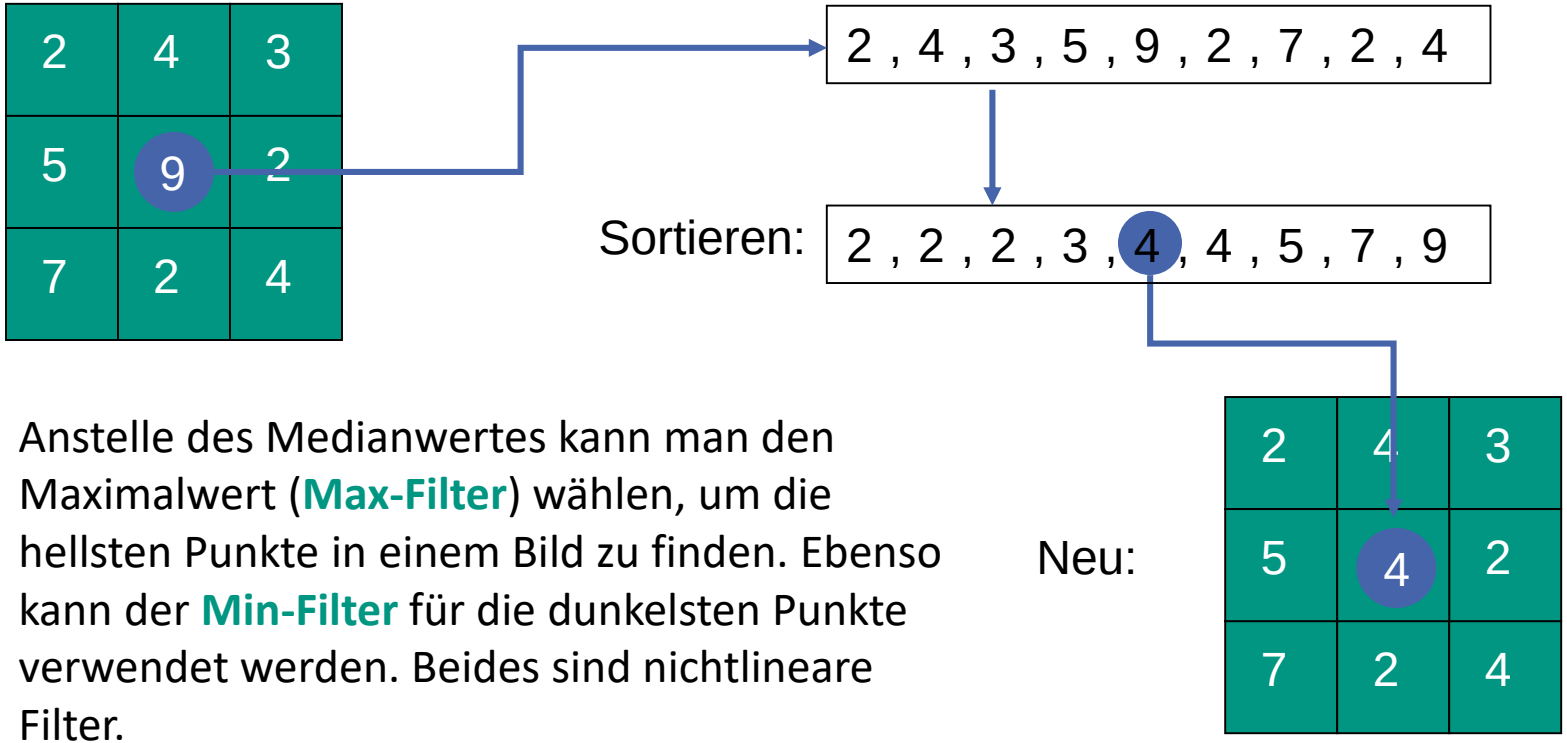
- **Tiefpassfilter**: Glättung, Rauschelimination
  - Medianfilter
  - Mittelwertfilter
  - Gauß-Filter
- **Hochpassfilter**: Kantendetektion
  - Prewitt
  - Sobel
  - Laplace
- **Kombinierte** Operatoren
  - Laplacian of Gaussian

Medianfilter

- **Nichtlineares** Filter zur Rauschunterdrückung
- Die Filterantwort basiert auf der Reihenfolge (Ranking) der Pixel, die in dem vom Filter umschlossenen Bildbereich enthalten sind.
- Schritte:
  - Wähle die Kernelgröße (Maske).
  - Sortiere alle Grauwerte in Bereich des Kernels
  - Bestimme den mittleren Grauwert aus den sortierten Pixeln
  - Wähle das Pixel als neuen Wert aus



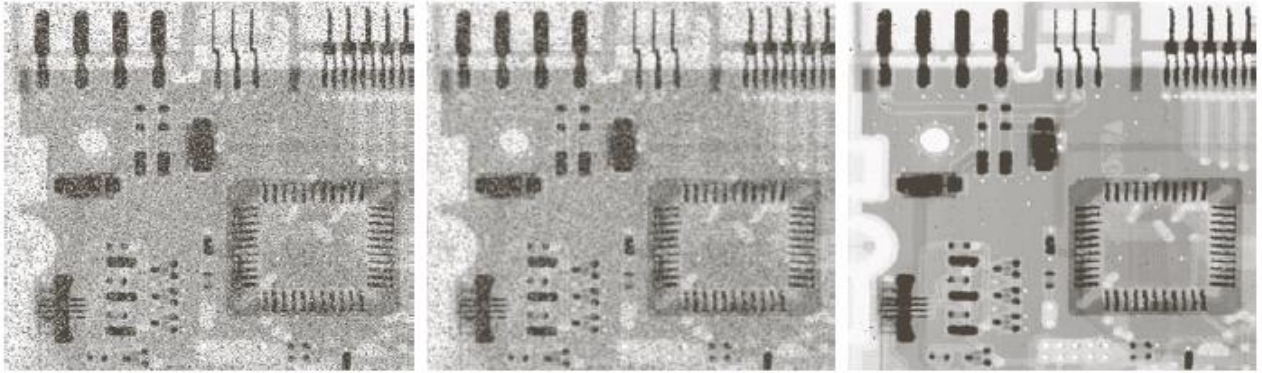
Medianfilter - Beispiel



Anstelle des Medianwertes kann man den Maximalwert (**Max-Filter**) wählen, um die hellsten Punkte in einem Bild zu finden. Ebenso kann der **Min-Filter** für die dunkelsten Punkte verwendet werden. Beides sind nichtlineare Filter.

Medianfilter - Beispiel (2)

- Für bestimmte Arten von Rauschen, wie z.B. **Salz- und Pfefferrauschen**, hat das Medianfilter sehr gute Rauschunterdrückungseigenschaften mit weniger Unschärfe als lineare Filter ähnlicher Größe!
  - Geeignet für Salz- und Pfefferrauschen;
  - Nicht geeignet für Gaußsches Rauschen
- Bewahrt Kanten und entfernt Rauschen im Bild.



Original Bild beschädigt durch *Salz- und Pfefferrauschen*

3 X 3 Mittelwertfilter

3 X 3 Medianfilter

Mittelwertfilter

- **Ziel:** Rauschunterdrückung
  - Durchschnitt aus einem Pixel und seinen 8 Nachbarn
  - Größe beliebig wählbar
  - 3x3 Mittelwertfilter: Filtermaske

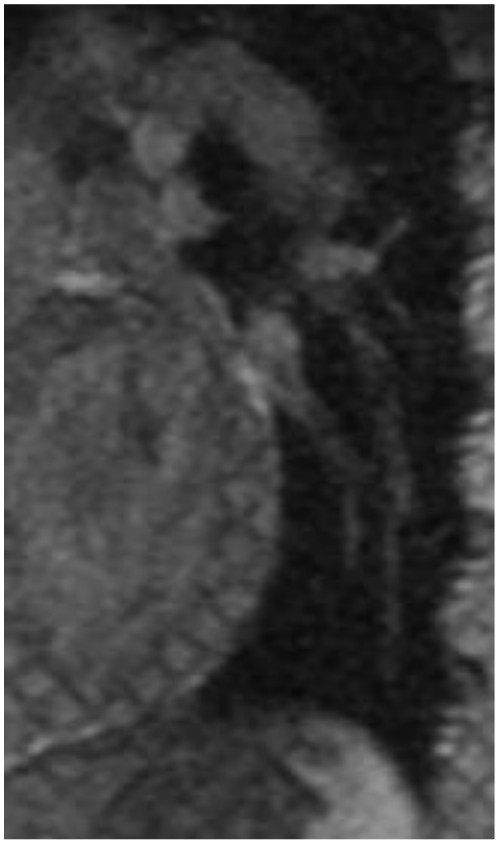
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

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

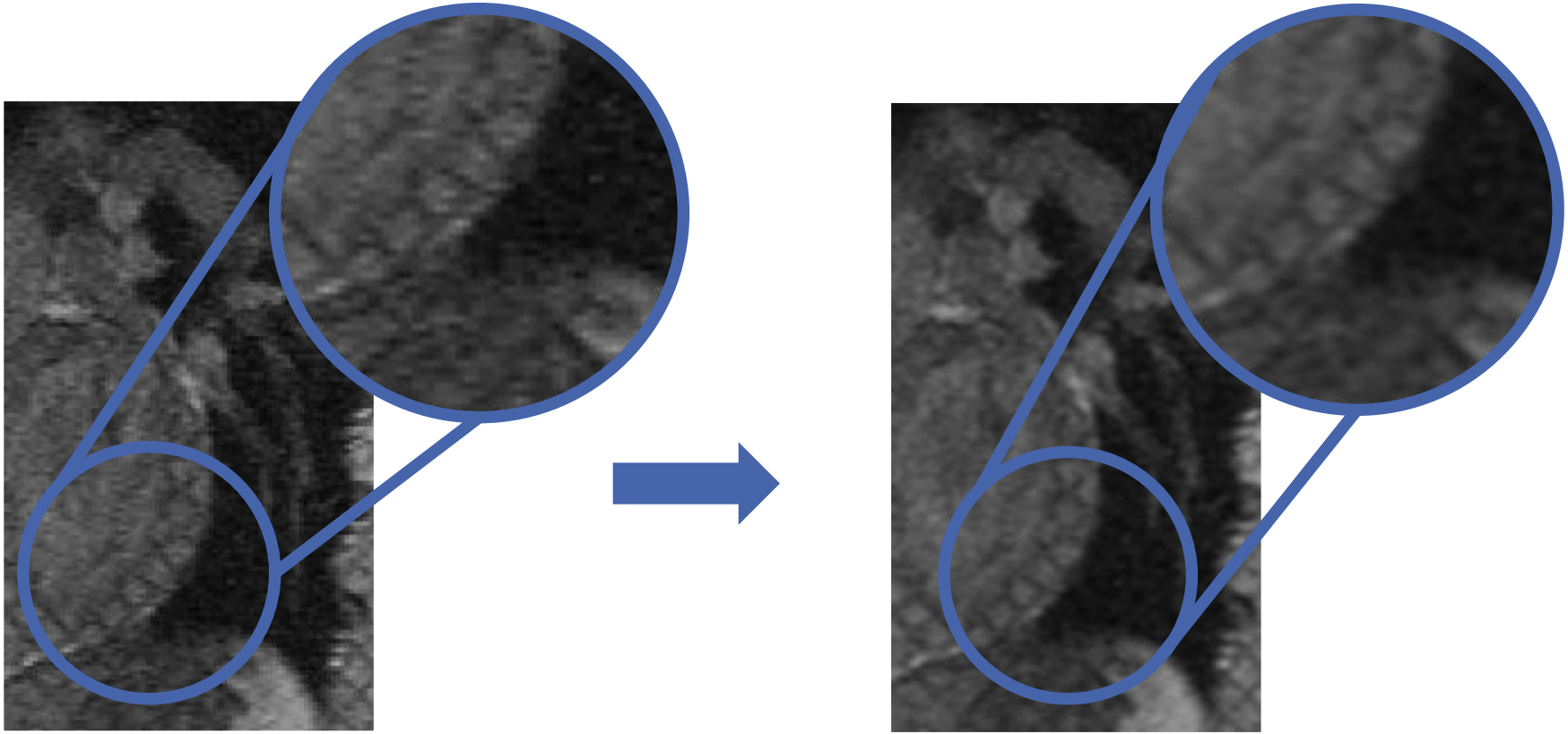
$$W_{Mittelwert} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$g(x,y) = \sum_{s=-1}^{1}\sum_{t=-1}^{1} w(s,t) \cdot f(x+s, y+t)$$

Mittelwertfilter - Beispiel



Mittelwertfilter - Beispiel (2)



Gauß-Filter

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

$$w(x,y) = \frac{1}{2\,\pi\sigma^2} e^{-\frac{x^2+y^2}{2\,\sigma^2}}$$

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

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

Gauß-Filter (2)

- 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

$\sigma^2 = 4$

$\sigma^2 = 16$

Filteroperationen

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

Filter – Prewitt

- **Prewitt-X Filter** $P_x = \frac{\partial f(x,y)}{\partial x}$
  - Gradient in horizontaler Richtung
  - Approximiert durch

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

- **Prewitt-Y Filter** $P_y = \frac{\partial f(x,y)}{\partial y}$
  - Gradient in vertikaler Richtung)
  - Approximiert durch

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

Filter – Prewitt (2)

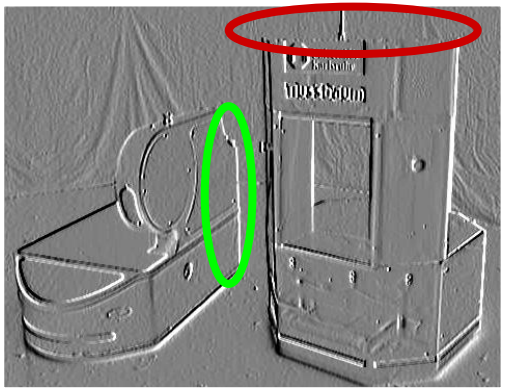
- **Prewitt-X Filter** $P_x = \frac{\partial f(x,y)}{\partial x}$
  - Approximiert durch

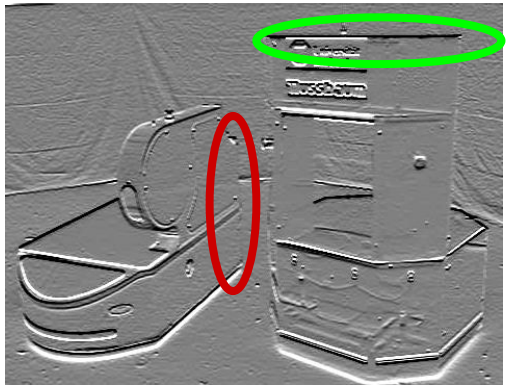
$$p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

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

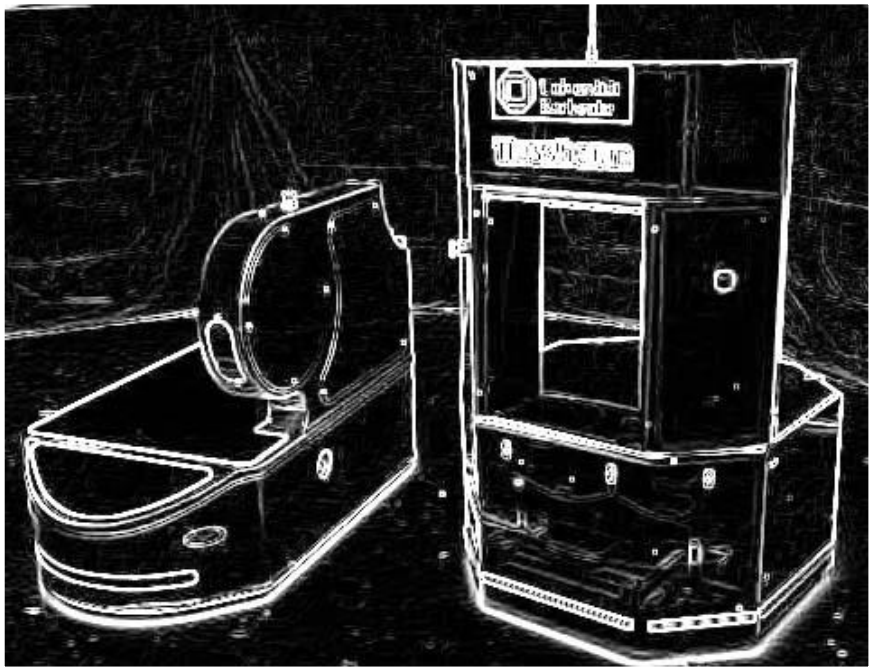




Filter – Prewitt (3)

- Kombination der Prewitt-Filter zur Bestimmung des Gradientenbetrags M

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



- Danach: Schwellwertfilterung

Filter – Sobel

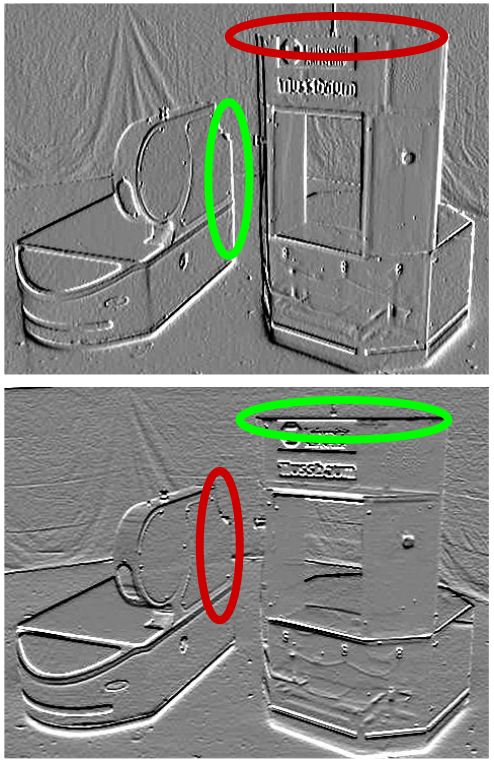
- Ähnlich wie Prewitt, gewichtet aber das Zentrum des betrachteten Fensters stärker
- Unterdrückt Rauschen; behält aber die Kanten
- Sobelfilter ist die Verkettung von Gaußfilter über Ableitung und Differenzenquotienten

- **Sobel-X Filter**
  Approximiert durch

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

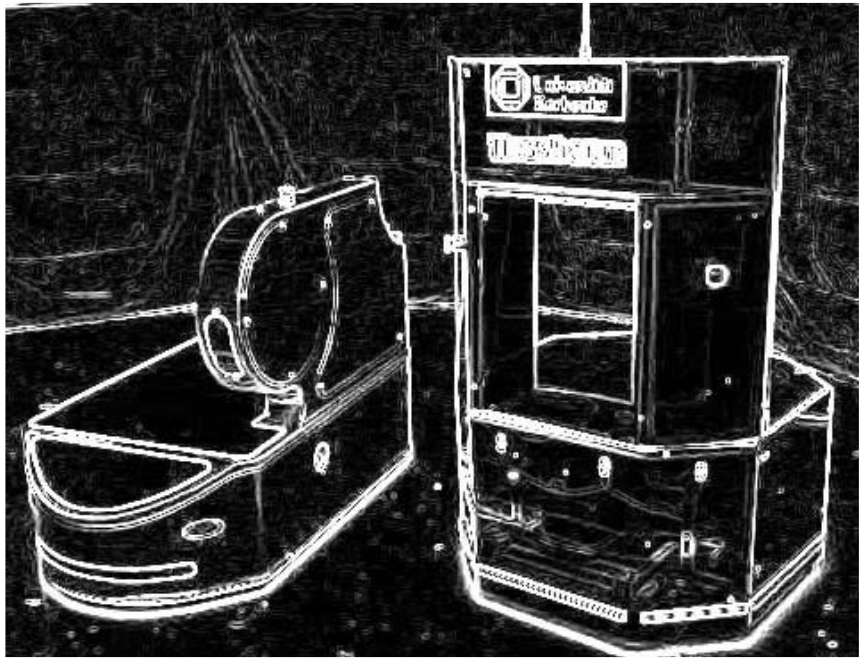
- **Sobel-Y Filter**
  Approximiert durch

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



Filter – Sobel (2)

- Ähnlich wie Prewitt
- Kombination der Sobel-Filter zur Bestimmung des Gradientenbetrags M
- Danach: Schwellwertfilterung



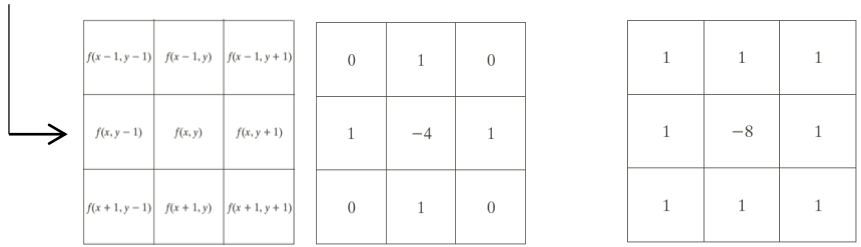
Filter – Laplace (1)

- Der Laplace-Operator ist ein linearer und **rotationsinvarianter** Operator zweiter Ordnung.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \begin{cases} \frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \end{cases}$$

$$\Delta \mathrel{\widehat{=}} \nabla^2$$

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$



Filter Maske (90° Rotationsionvariant)

Erweiterte Maske (45° Rotationsinvariant)

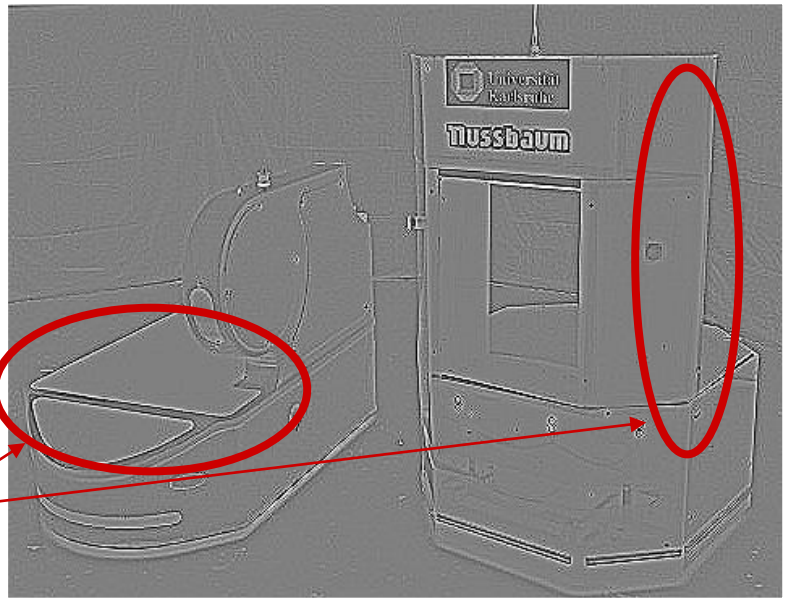
Die Summe der Masken-Koeffizienten ist gleich Null, was darauf hindeutet, dass die Antwort in den Bereichen konstanter Intensität gleich Null ist.

Filter – Laplace (2)

Laplace-Operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 \approx \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



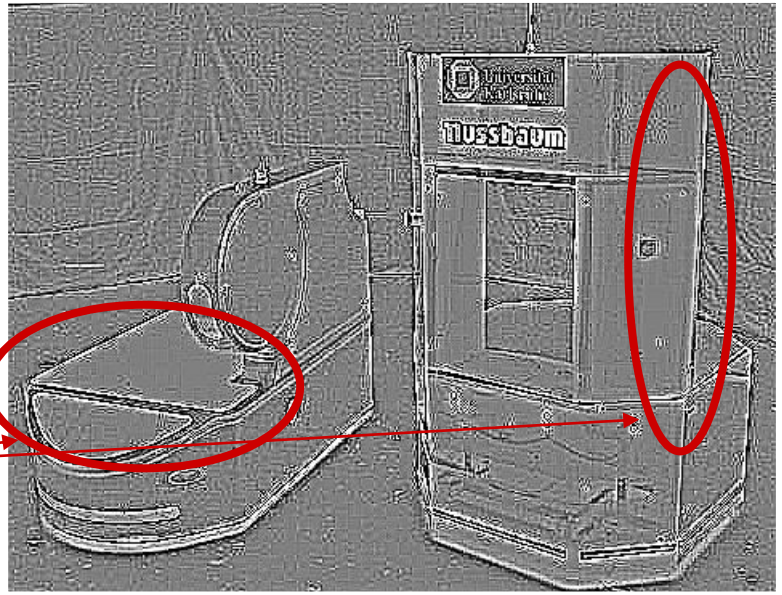
Nulldurchgänge (*Zero-Crossings*) definieren Kanten

Die Kanten sind dünner als bei Prewitt oder Sobel.

Filter – Laplace (3)

Variation des **Laplace-Operator**:

$$\nabla^2 \approx \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



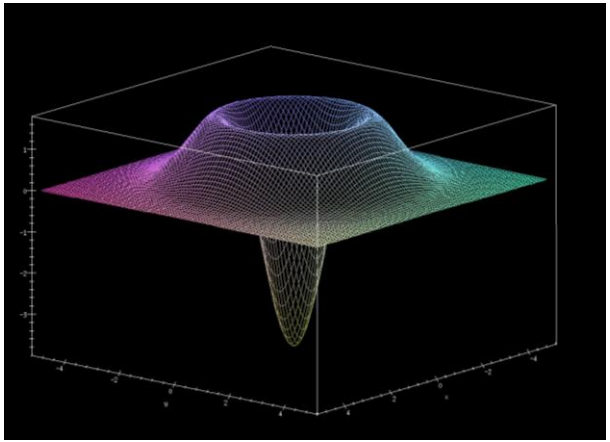
- Stärkere, aber mehr störende Kanten
- Rauschanfällig, deshalb Glättung vor der Anwendung des Laplace-Filters, damit Artefakte nicht als Kanten erkannt werden

Filteroperationen

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

Filter – Laplacian of Gaussian (LoG)

- Der Laplace-Operator ist sehr rauschempfindlich
- Wesentlich bessere Ergebnisse werden erzielt, wenn man das Bild mit einem Gauß-Filter glättet und dann den Laplace-Operator (Laplacian of Gaussian, LoG) verwendet:



$$LoG\big(f(x,y)\big) = \nabla^2\big(f(x,y) * g(x,y)\big) = \Delta\left(f(x,y) * g(x,y)\right) \qquad \Delta \mathrel{\widehat{=}} \nabla^2$$

g bezeichnet die Filterfunktion eines Gauß-Filters.

Hauptmerkmale des LoG-Operators

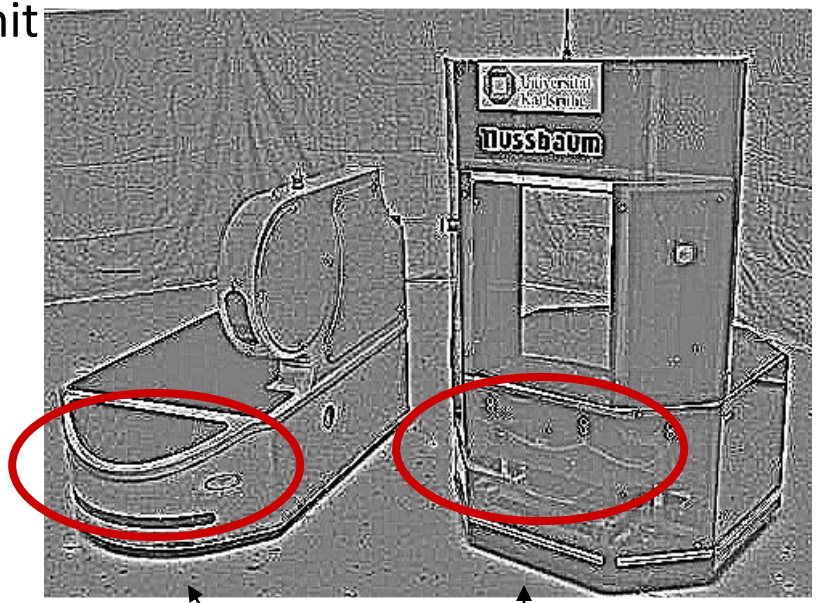
- Gaußscher Teil des Operators ist ein **Tiefpassfilter**, welches das Bild glättet (verwischt) und so die Intensität von Strukturen (z.B. Rauschen) auf Skalen reduziert, die viel kleiner sind als Sigma σ.
- Im Gegensatz zum Mittelwertfilter ist es beim Gauß-Filter unwahrscheinlich, dass Artefakte wie z.B. "*Treppenhaus*"-Effekte auftreten, die im Originalbild nicht vorhanden sind.
- Der Laplace-Operator (die zweite Ableitung) ist invariant bzgl. der Rotation, die gleichermaßen auf Intensitätsänderungen in **jeder Maskenrichtung** reagiert. Auf diese Weise können wir vermeiden, mehrere Masken zu verwenden, um die stärkste Reaktion an jedem Punkt des Bildes zu berechnen.

- Nulldurchgänge (*Zero Crossings*) des Laplacian entsprechen den Kanten.

Filter - Laplacian of Gaussian (LoG) (2)

- Approximation von LoG durch die Faltung mit der Matrix

- $\Delta F(x,y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$

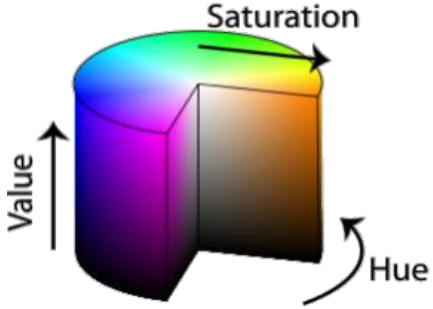


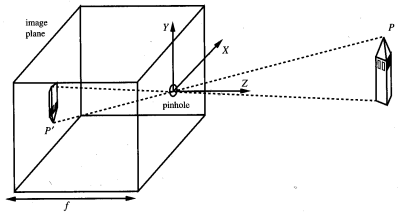
Stärkere Kanten,

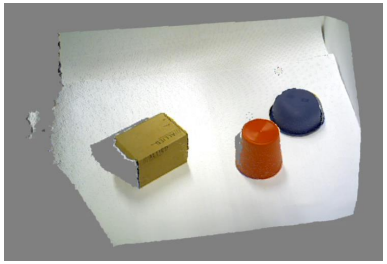
Weniger Rauschen

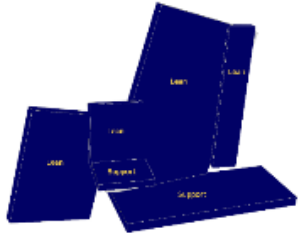
Inhalt

- Bilderzeugung
- Operationen auf Bildern
  - Filteroperationen
  - **Morphologische Operatoren**
- Merkmalsextraktion und Mustererkennung





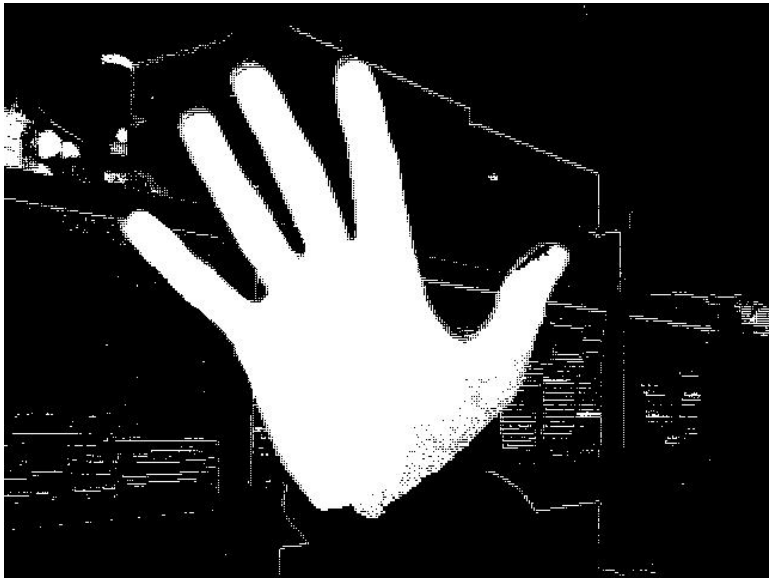




Motivation: Farbsegmentierung

- Problem bei der Segementierung: Artefakte, unvollständige Segementierung

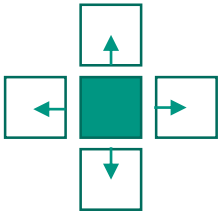
Eingabebild



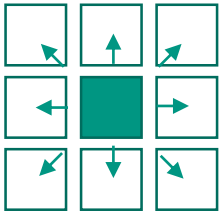
Ergebnis Segmentierung

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; verbindet Strukturen; ODER-Operator
  - **Erosion:** Die Erosion entfernt vereinzelte Pixel und schwach zusammenhängende Pixelgruppen; löst Strukturen auf; UND-Operator
- Der Effekt eines morphologischen Operators hängt von der Größe und der Form der berücksichtigten Pixelnachbarschaft (**Strukturelement**) ab.



4er-Nachbarschaft



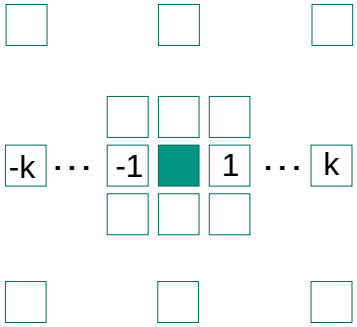
8er-Nachbarschaft

Morphologische Operatoren – Dilatation (ODER)

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
```

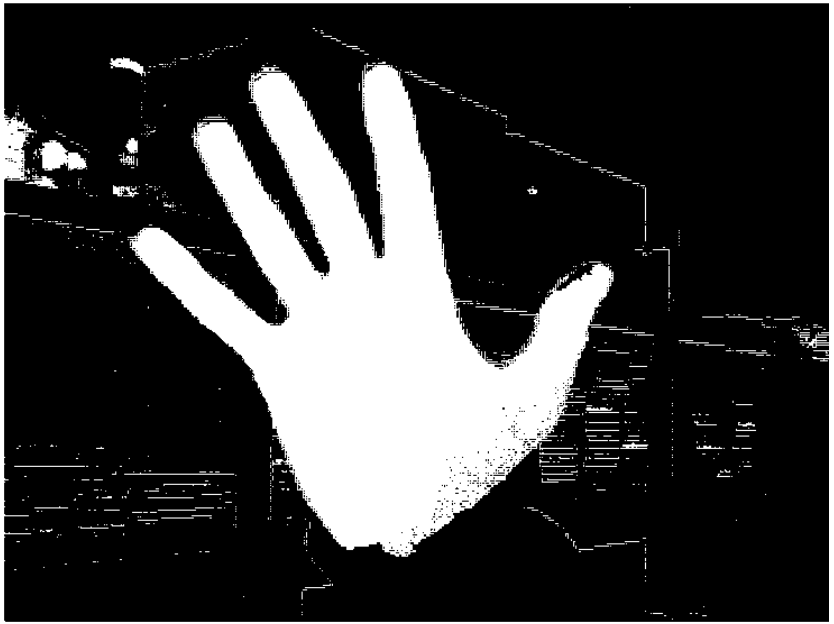
Bildhöhe h,
Bildbreite w



ODER zwischen Strukturelement und Bild

Beispiel: Dilatation

- Verbindet Strukturen; ODER



Eingabebild

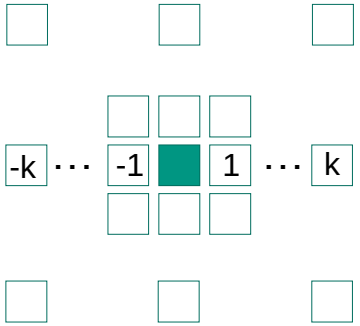
Ergebnisbild

Morphologische Operatoren – Erosion (UND)

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
```

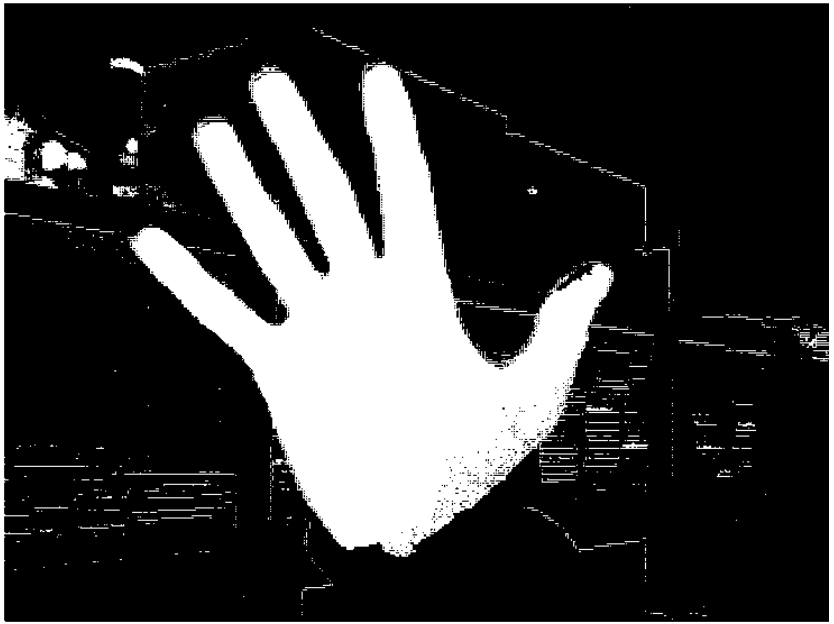
Bildhöhe h,
Bildbreite w



UND zwischen Strukturelement und Bild

Beispiel Erosion

- Löst Strukturen auf; UND



Eingabebild

Ergebnisbild

Morphologische Operatoren – Öffnen & Schließen

Öffnen:

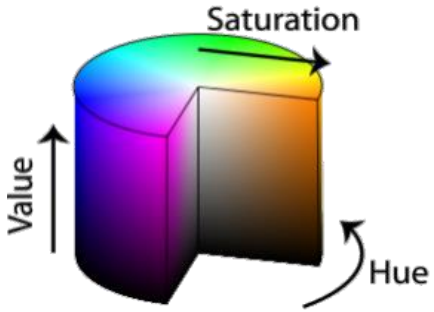
- Anwendung von Erosion danach Dilatation
- Entfernt dünne Linien oder kleine außenliegende Bereiche

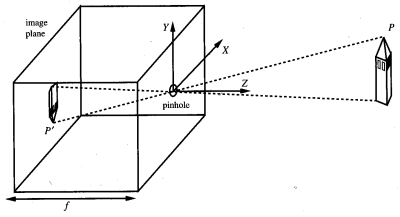
Schließen:

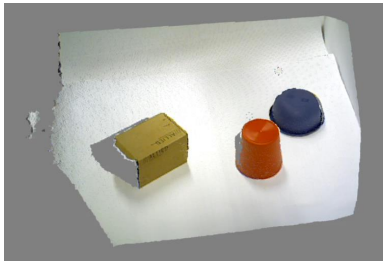
- Anwendung von Dilatation danach Erosion
- Überbrückung kleiner Distanzen und Schließung von inneren Löchern

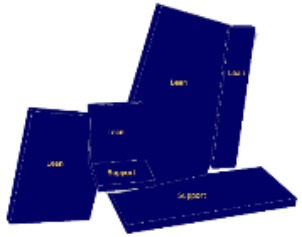
Inhalt

- Bilderzeugung
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung
  - **Segmentierung**
  - Canny-Kantendetektor
  - Visual Servoing
  - Registrierung von Punktwolken
  - Beispielanwendungen H²T





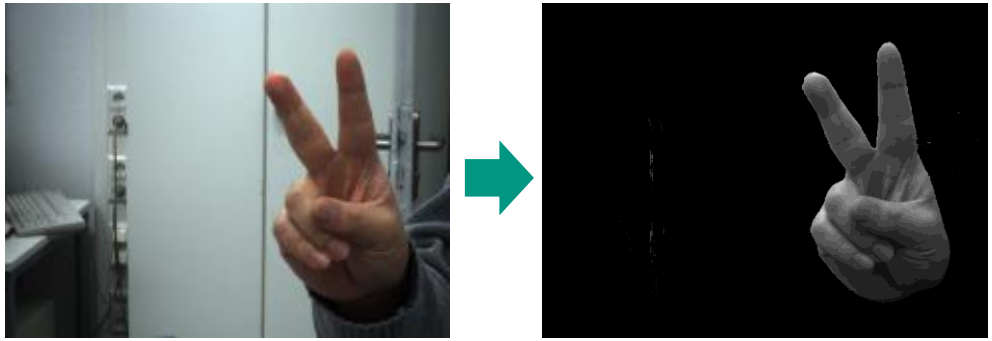




Segmentierung

- **Segmentierung** ist die Aufteilung einer Menge in aussagekräftige Segmente
  - Mögliche Mengen in der Computer Vision: Bilder, Punktwolken, Formen, …
- Jedes Pixel wird mindestens einem Segment zugeordnet
- Identifikation von interessante Bildregionen für die Analyse, Erkennung und Klassifikation

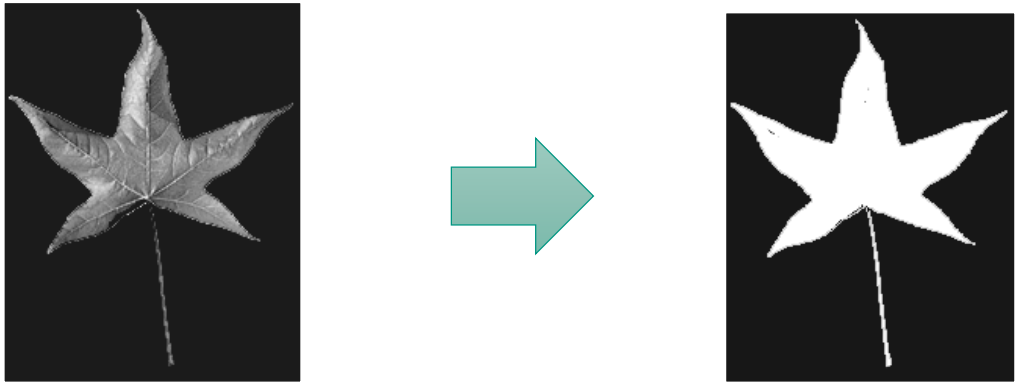
Mögliche Verfahren:
- Schwellwertfilterung
- Clustering
- Kantenextraktion
- Region-Growing



Segmentierung – Schwellenwertfilterung

- Schwellwertfilterung zur Konvertierung eines Grauwertbildes in ein binäres Bild
- Intensität von jedem Pixel (u, v) wird mit einem vordefinierten **Schwellwert** T verglichen

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



Segmentierung – Schwellenwertfilterung (2)

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

- **Beispiel:**
  - Intervallschranken im HSV-Farbraum:

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

- **Problem:**
  - Wechselnde Lichtbedingungen
  - Reflexionen, Schattenwürfe

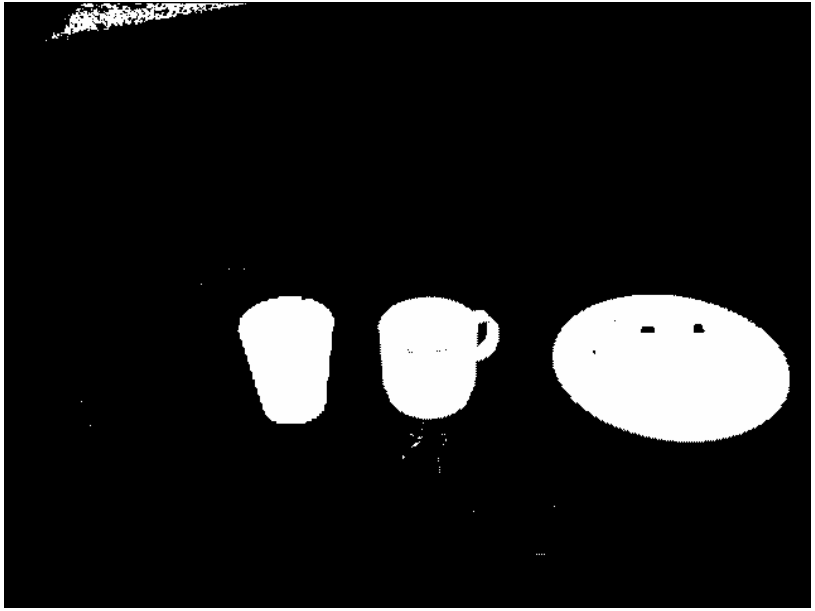
Probleme von Lichtverhältnisse

Ist dieses Kleid weiß & gold oder blau & schwarz?

Segmentierung – Beispielanwendung

- Beispielanwendung: Objekterkennung und -lokalisierung

Eingabebild



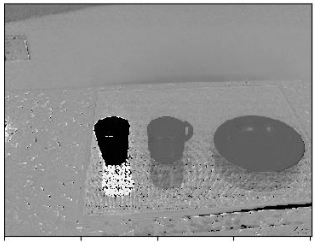
Ergebnis der Segmentierung

HSV-Segmentierung

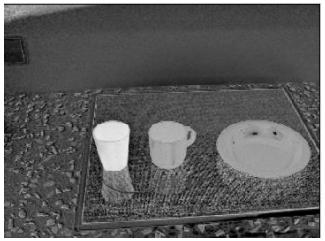
Wie können grüne Regionen bestimmt werden? (HSV)

Original Bild

Farbnuance (H)



Sättigung (S)



Helligkeit (V)

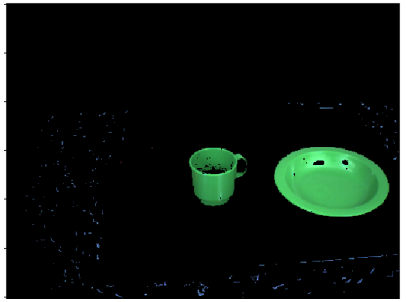
HSV

Schwellwertfilterung anwenden

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

Bereich für HSV Werte angeben. Vorteil: Farbnuance kann von der Helligkeit entkoppelt werden

Maske



Segmentiertes Bild

RGB Segmentierung

Originalbild

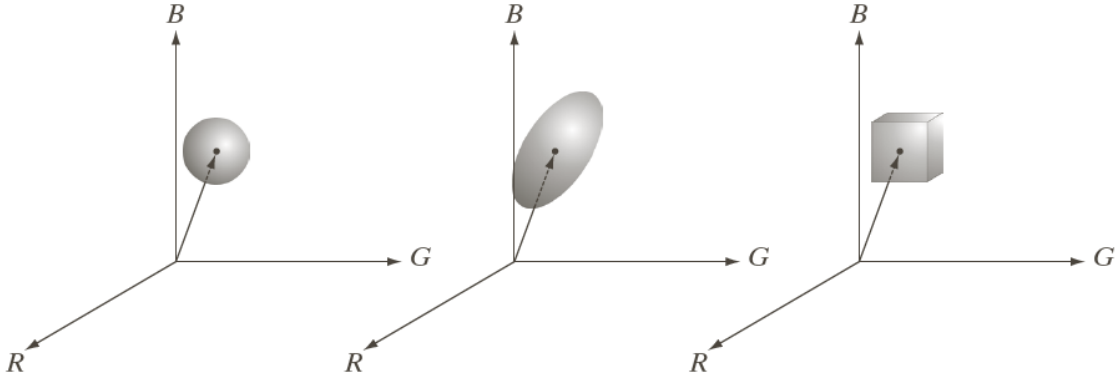
Wie können grüne Regionen bestimmt werden? (RGB)

Euklidische Distanz
der beiden Farbvektoren!

$D(z,a) \leq D_0$

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

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



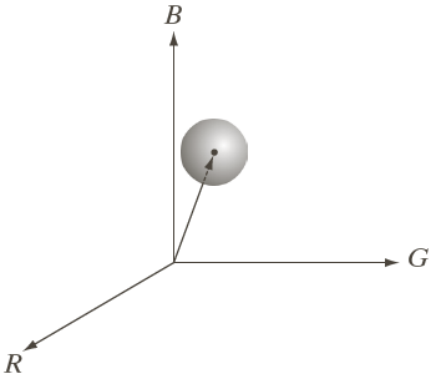
Unterschiedliche Bereiche!

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

RGB Segmentierung

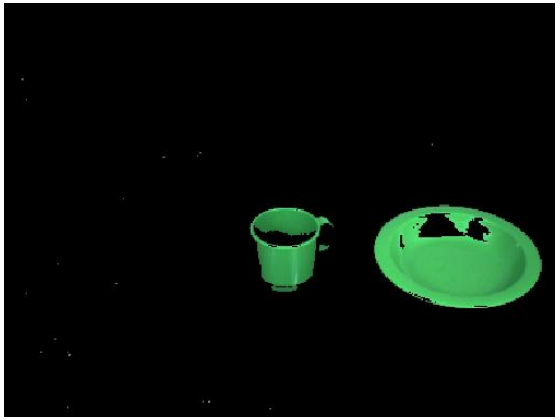
Wie können grüne Regionen bestimmt werden? (RGB)

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



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

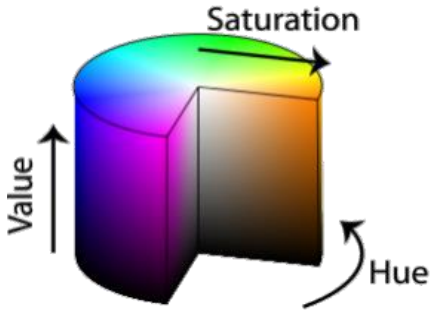
Original Bild

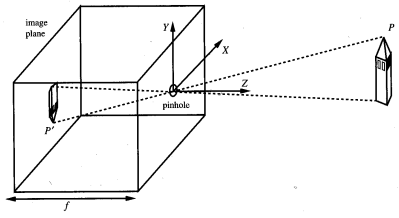


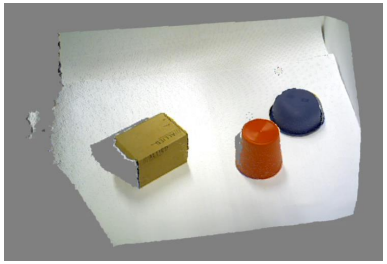
Segmentiertes Bild

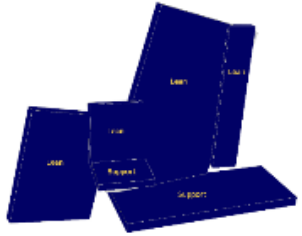
Inhalt

- Bilderzeugung
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung
  - Segmentierung
  - **Canny-Kantendetektor**
  - Visual Servoing
  - Registrierung von Punktwolken
  - Beispielanwendungen H²T









Canny-Kantendetektor

- Nach John F. Canny aus dem Jahre 1986
- Der Canny-Kantendetektor ist weit verbreitet und bezüglich der Leistung anderen Kantendetektoren 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

Canny-Kantendetektor (2)

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 – Algorithmus

1. Gauß-Filter
   → Rauschunterdrückung
2. Berechnung der Intensitätsgradienten
3. Non-Maximum Suppression
   → Unterdrückung von nicht-maximalen „Kantenpixel“
4. Double Threshold
   → Bestimmung von möglichen Kanten
5. Kantenverfolgung mit Hysterese
   → Entferne schwach verbundene Kantenpixel

Canny-Kantendetektor – Intensitätsgradienten

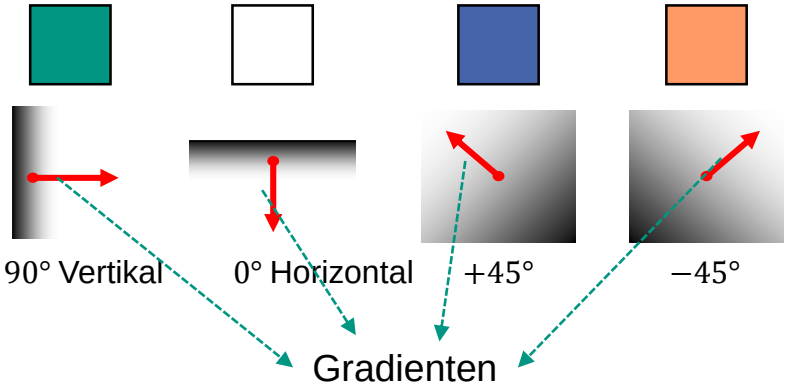
- Berechnung der Gradienten in horizontaler und vertikaler Richtung mit Prewitt- oder Sobel-Filter

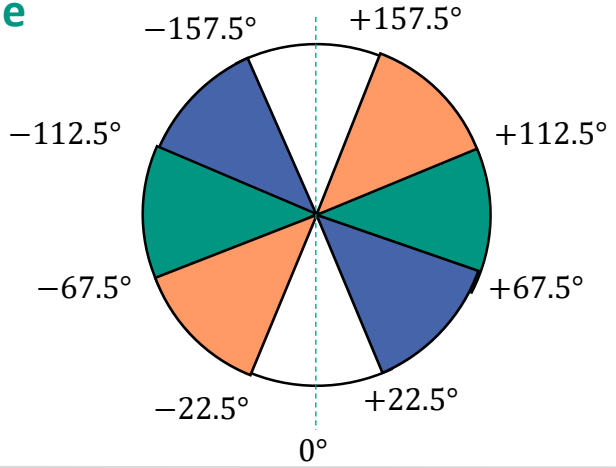
$$g_x = f * s_x, \qquad g_y = f * s_y$$

- Bestimmung der Orientierung θ und des Betrags M des Gradienten

$$\theta = \operatorname{atan2}(g_y, g_x), \qquad \theta \in (-180°, 180°], \qquad M = \sqrt{g_x^2 + g_y^2}$$

Diskretisierung des Winkels θ: Einteilung in **vier Bereiche**

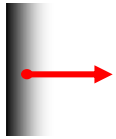




Canny-Kantendetektor – Non-Maximum Suppression

Non-Maximum Suppression: d.h. eine Kantenausdünnung. Alle Pixel, die kein Maximum darstellen, werden unterdrückt.

- Gradient muss lokales Maximum sein
- Betrachtung der zwei direkten Nachbarn entlang der Gradientenrichtung
- Überprüfung erfolgt gemäß der diskreten Gradientenrichtung
  → 3x3 Filter bestimmt benachbarte Pixel zum Vergleich

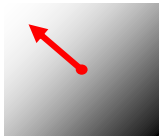


Vertikal

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

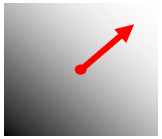
Horizontal

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$



$+45°$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



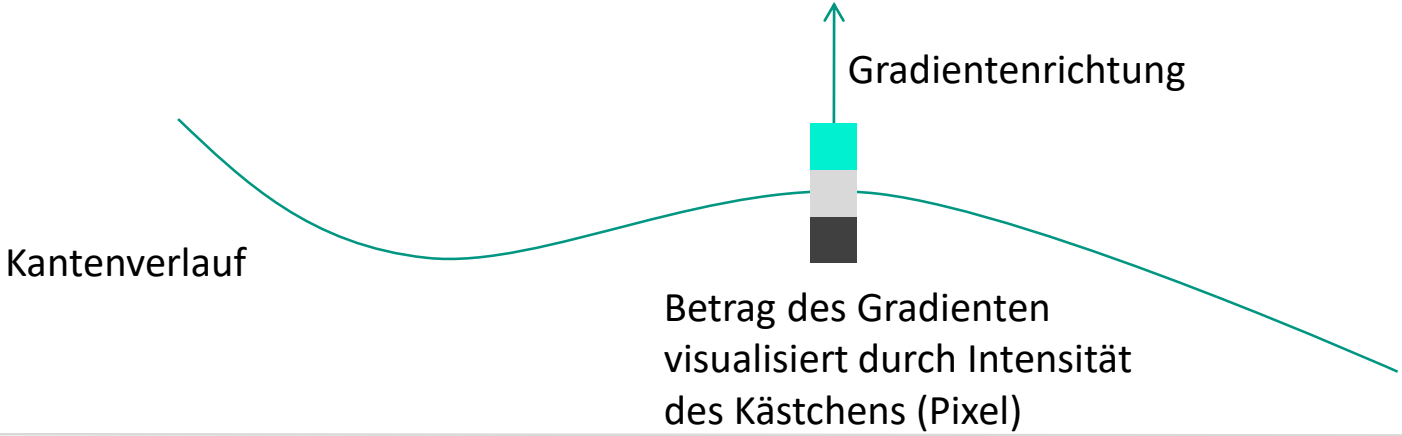
$-45°$

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Canny-Kantendetektor – Non-Maximum Suppression (2)

Non-Maximum Suppression: d.h. eine Kantenausdünnung. Alle Pixel, die kein Maximum darstellen, werden unterdrückt.

- Gradient muss lokales Maximum sein
- Betrachtung der zwei direkten Nachbarn entlang der Gradientenrichtung
- Überprüfung erfolgt gemäß der diskreten Gradientenrichtung



Canny-Kantendetektor – Double Threshold

Double Threshold zur Entfernung schwach verbundener Kantenpixel

- Einteilung der Pixel in **starke**, **schwache** und **keine Kanten**
- Zwei Schwellwerte werden definiert

$$M_{weak}, \qquad M_{strong}$$

- Drei Intervalle für die Betrag M des Gradienten

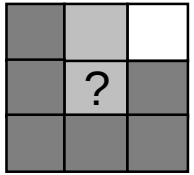
$0 \leq M \leq M_{weak}$	Keine Kante
$M_{weak} \leq M \leq M_{strong}$	Schwache Kante
$M > M_{strong}$	Starke Kante

Canny-Kantendetektor – Kantenverfolgung mit Hysterese

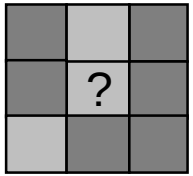
Der Canny-Kantendetektor verwendet **Kantenverfolgung mit Hysterese**

- Starke Kantenpixel bleiben immer erhalten
  - $M > M_{strong}$: starke Kante
- Welche schwachen Kantenpixel werden erhalten bzw. verworfen?
  - $M_{weak} \leq M \leq M_{strong}$: schwache Kante

- Betrachte die 8 Nachbarpixel von jedem schwachen Kantenpixel
  - Befindet sich ein starker Kantenpixel in der Nachbarschaft: Kantenpixel
  - Befindet sich kein starker Kantenpixel in der Nachbarschaft: Kein Kantenpixel



→ Kantenpixel

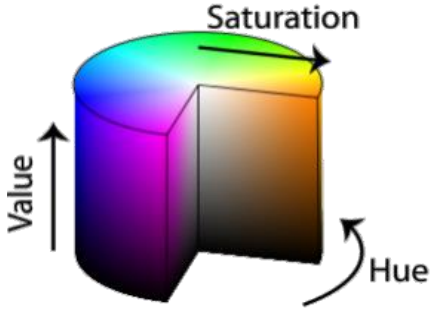


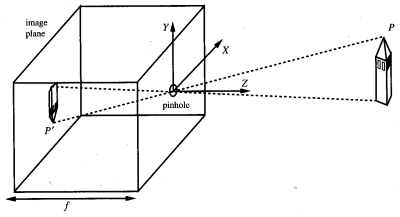
→ Kein Kantenpixel

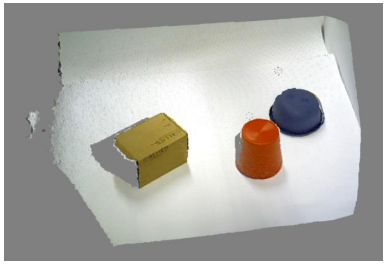
Canny-Kantendetektor – Beispiel

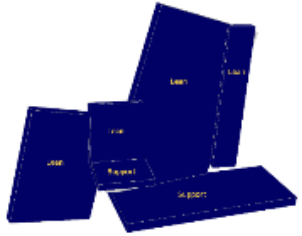
Inhalt

- Bilderzeugung
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung
  - Segmentierung
  - Canny-Kantendetektor
  - **Visual Servoing**
  - Registrierung von Punktwolken
  - Beispielanwendungen H²T





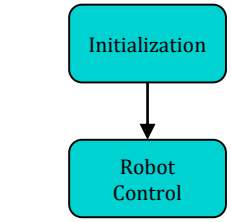




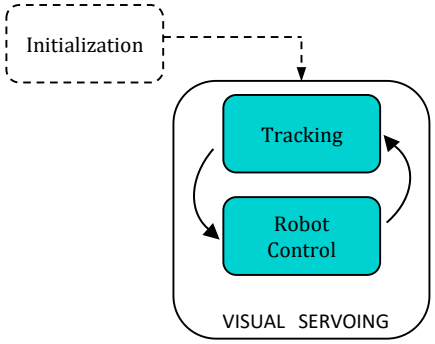
Visual Servoing – Motiviation

- Der Ausdruck **Visual Servoing** beschriebt Verfahren bei denen visuelle Eingabedaten genutzt werden, um die Bewegung eines Roboters zu steuern (**bildgeführte Bewegung**).

- **Motivation**
  - Modellfehler z.B. Kinematik
  - Fehler bei der Ausführung
    z.B. fehlerhafte Positionierungen der Robotergelenke
  - Überwachung der Szene
    z.B. Reaktion auf Kollisionen



open-loop Regelung

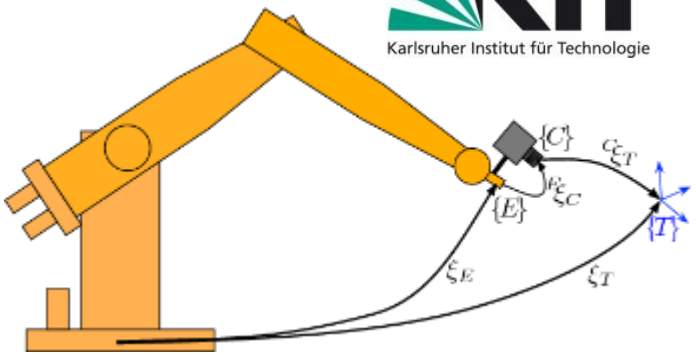


Visual Servoing

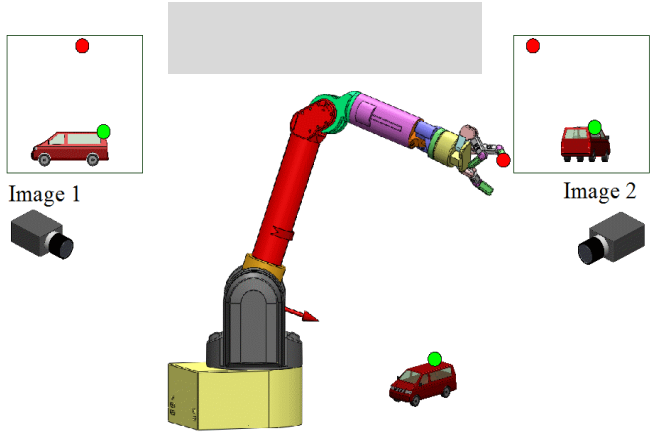
- François Chaumette, Seth Hutchinson, *Visual servo control - Part I - Basic approaches,* IEEE Robotics & Automation Magazine 13 (4), 82-90, 2006
- Danica Kragic and Henrik I Christensen, *Survey on Visual Servoing for Manipulation*, Techn. Report, KTH, 2002

Visual Servoing

- **Kamera-in-Hand** (eye-in-hand)
  - Kamera ist an dem Manipulator angebracht
  - Bewegungen des Manipulators beeinflussen die Pose der Kamera

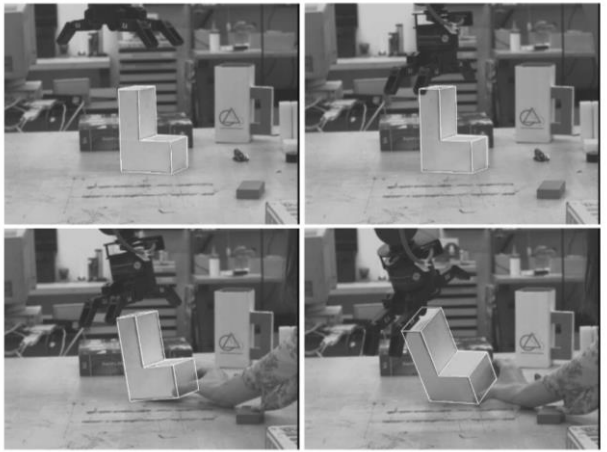


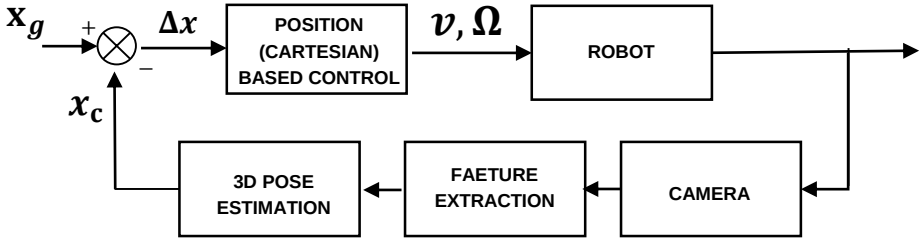
- **Externe (feste) Kamera** (eye-to-hand)
  - Externes Kamerasystem wird zur Beobachtung der Bewegung genutzt



Positionsbasiertes Visual Servoing

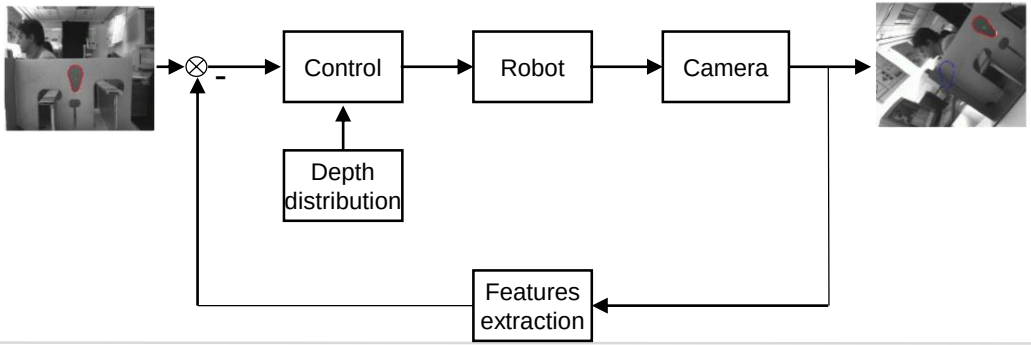
- Zielpose $\boldsymbol{x}_g$ ist vorgegeben
- Ablauf der Regelschleife
  - **3D Lageschätzung**:
    Aktuelle Pose $\boldsymbol{x}_c$ der Hand wird aus Bildmerkmalen extrahiert
  - **Regeldifferenz** (kartesisch) bilden:
    $\Delta \boldsymbol{x} = \boldsymbol{x}_g - \boldsymbol{x}_c$
  - **Kartesischer Regler** zum Ausgleich von $\boldsymbol{\Delta x}$
  - Ziel ist erreicht, wenn Distanz $\boldsymbol{\Delta x}$ kleiner als Schwellwert





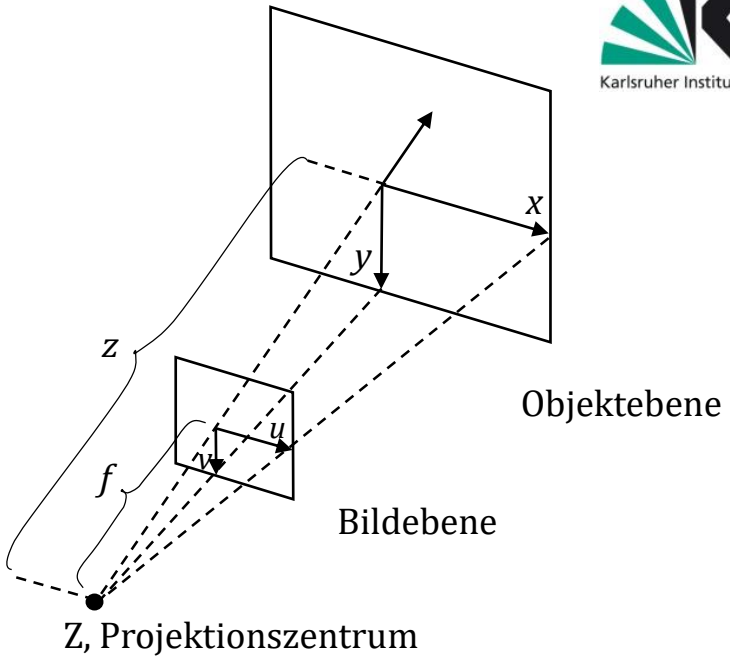
Bildbasiertes Visual Servoing

- **Ansatz:** Die Bewegung des Roboters (Arm) ergibt sich aus aktueller und gewünschter Position von Bildmerkmalen
- **Bildmerkmale:** Bildverarbeitungsmethoden zur Extraktion von Bildmerkmalen
- **Regelung:** Geschwindigkeitsvorgaben werden direkt aus der aktuellen und gewünschten Position der Bildmerkmale erzeugt



Bildbasiertes Visual Servoing (2)

- **Bildmerkmal:**
  Ein Bildmerkmal $s = (u, v)^T$ ist die Projektion eines 3D Punktes $\mathrm{p} = (x, y, z)^T$ im Kamerabild.



- **Fehlerfunktion:**
  - Aktuelle Position der Merkmale im Kamerabild (Zeitpunkt t): $s(t)$
  - Gewünschte Zielposition der Merkmale im Kamerabild (konstant): s^*
  - Fehler: $e(t) = s(t) - s^*$

Bildbasiertes Visual Servoing (3)

- **Interaction Matrix / Image Jacobian** (L) beschreibt die Beziehung zwischen der Geschwindigkeiten eines Bildmerkmals und der Kamera

$$\dot{s} = L(u, v, z) \cdot \dot{p}$$

$\dot{s}$ ist Geschwindigkeit eines Pixels $(\dot{u}, \dot{v})$

$\dot{p}$ ist Geschwindigkeit der Kamera $(v, \omega) = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$

z Tiefe des Punkts

$L \in \mathbb{R}^{2\times 6}$

Bildbasiertes Visual Servoing (4)

- **Interaction Matrix / Image Jacobian** (L) beschreibt die Beziehung zwischen der Geschwindigkeiten eines Bildmerkmals und eines der Kamera

$$\dot{s} = L(u, v, z) \cdot \dot{p}$$

- Aus den Projektionsvorschriften (Lochkameramodell) ergibt sich

$$L = \begin{pmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{u \cdot v}{f} & \frac{f^2 + u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & -\frac{f^2 + v^2}{f} & \frac{uv}{f} & u \end{pmatrix}$$

Bildbasiertes Visual Servoing (5)

Invertierung der Interaction Matrix

- Distanz z wird geschätzt
- Mehrere (**mindestens 3**) Merkmale werden beobachtet
- Annahme: Kamera-In-Hand System
  Bewegung der 3D Punkte korrespondiert mit Bewegung der Kamera v_C
- Daraus folgt:

$$\dot{e} = L\, v_C$$

- Mit $\dot{e} = -\lambda\, e$ (Fehler soll gegen null gehen) ergibt sich

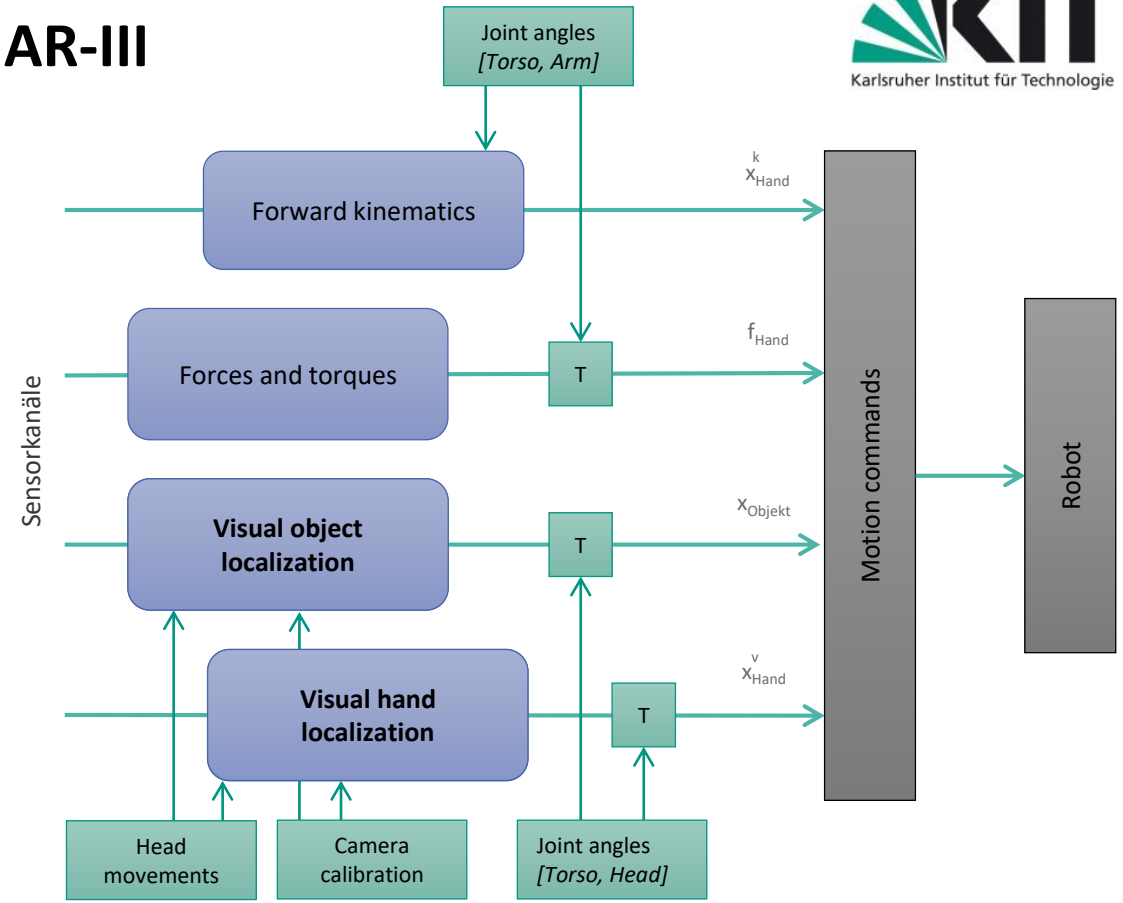
$$-\lambda\, e = L\, v_C \quad \rightarrow \quad v_C = -\lambda\, L^{+}\, e$$

L^{+} ist die Pseudoinverse von L

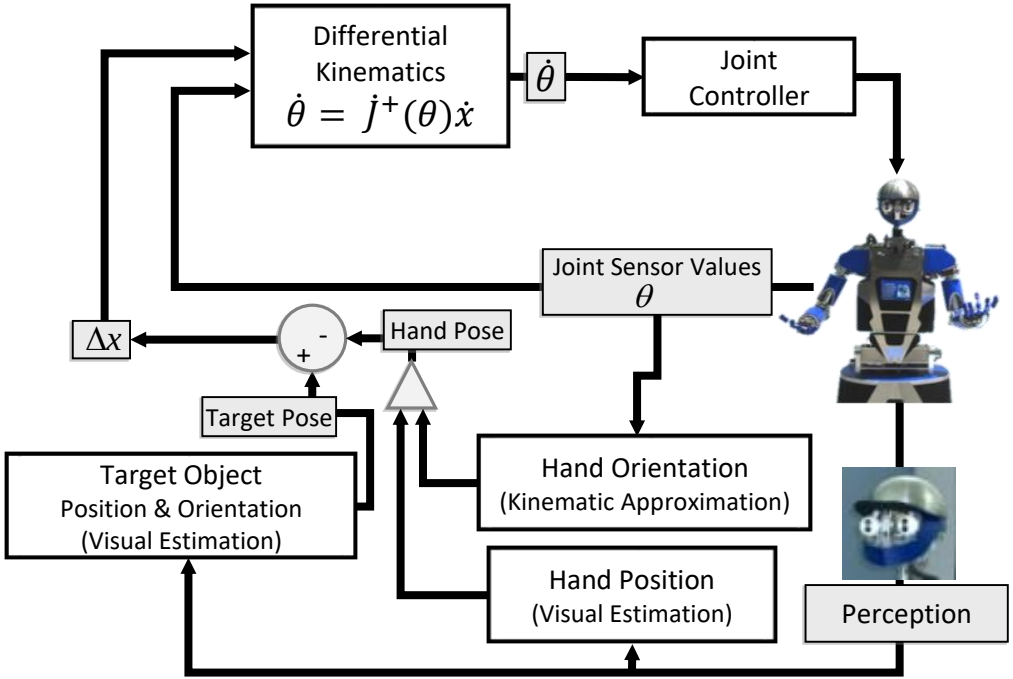
- Regelungseingabe v_C wird aus der Position der Bildmerkmale berechnet

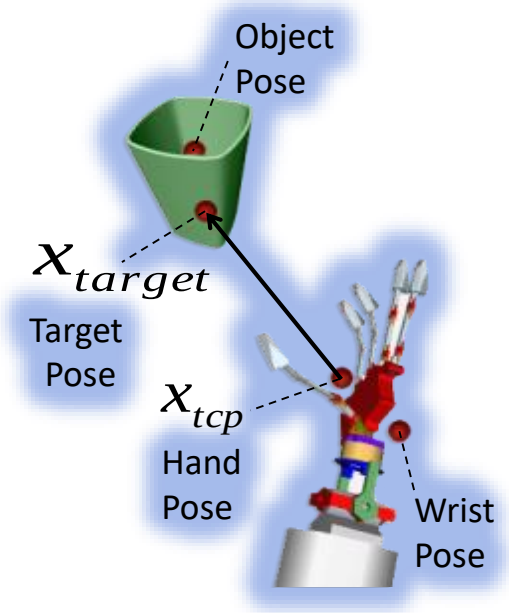
Visual Servoing für ARMAR-III

- Ausführung von Greif und Manipulationsaufgaben
- Positionsbasiertes Visual Servoing
  - Modellwissen
  - Objektlokalisierung
  - Handlokalisierung
- Sensoren
  - Kraft/Kontakt
  - Kameras
  - Interne Sensoren



Positionsbasiertes Visual Servoing auf ARMAR-III



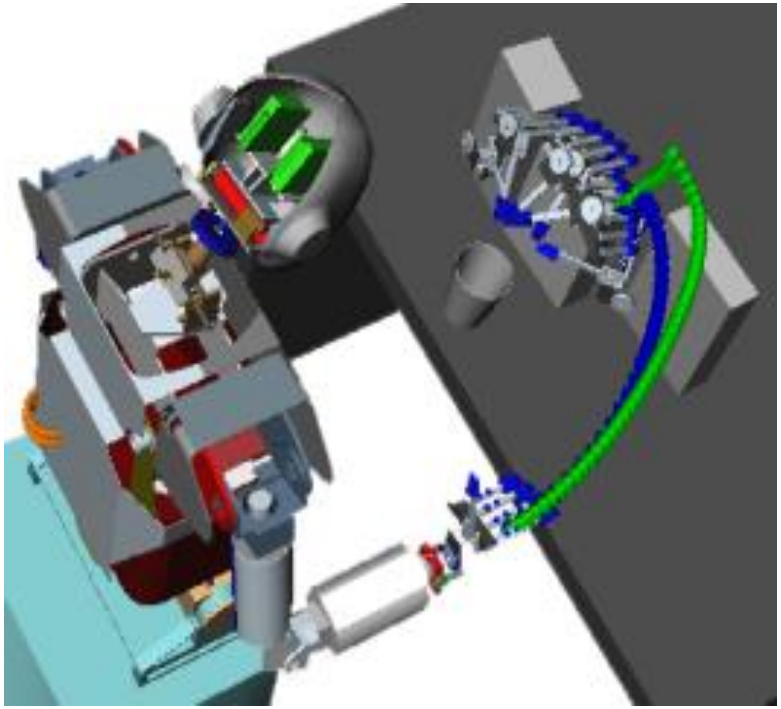


$$\dot{\theta} = j^{+}(\theta)\dot{x}$$

$$\delta^{t} = x^{t}_{vision} - x^{t}_{kinematic}$$

$$x^{t+1}_{tcp} = x^{t+1}_{kinematic} + \delta^{t}_{tcp}$$

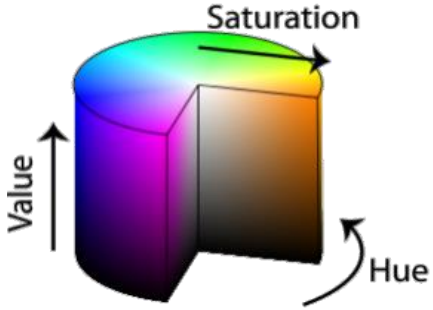
Positionsbasiertes Visual Servoing – ARMAR-III (1)

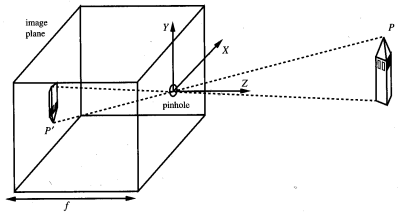


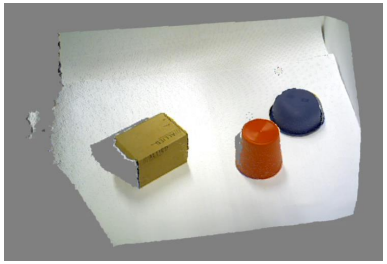
Positionsbasiertes Visual Servoing – ARMAR-III (2)

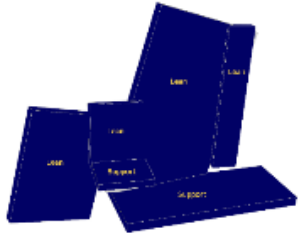
Inhalt

- Bilderzeugung
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung
  - Segmentierung
  - Canny-Kantendetektor
  - Visual Servoing
  - **Registrierung von Punktwolken**
  - Beispielanwendungen H²T







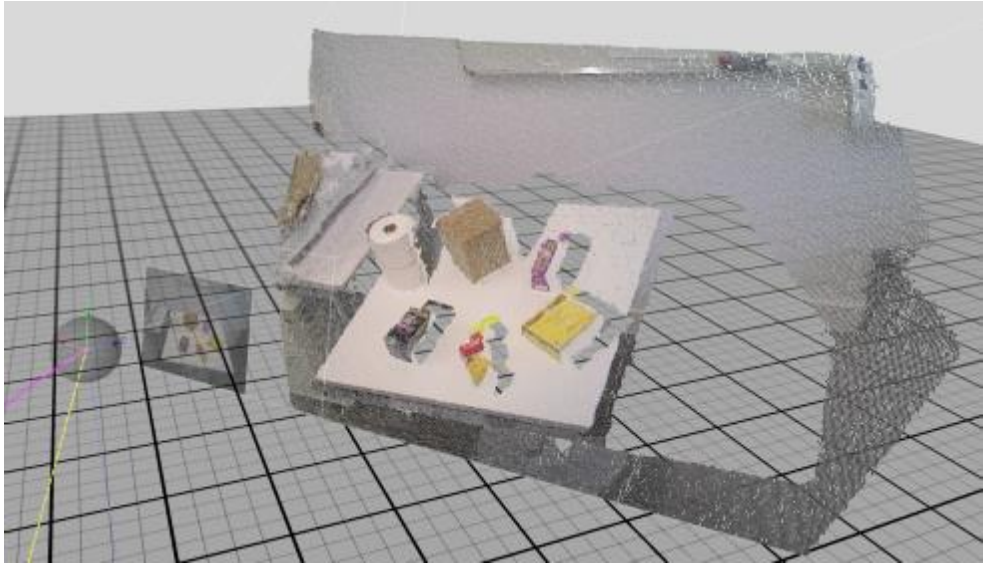


Normalenschätzung in 3D Punktwolken

Ziel:

- Zusätzliche Oberflächeninformation durch Einbeziehung von lokalen Nachbarpunkten

- Grundlage für weiterführende Algorithmen:
  - Segmentierung
  - Deskriptoren
  - Objekterkennung
  - Oberflächenmodellierung



Normalenschätzung in 3D Punktwolken (2)

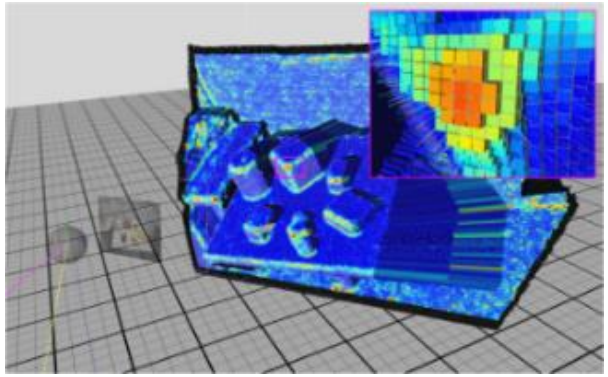
- PCA-basierter Ansatz
- Erstelle die Kovarianzmatrix C der k-Punktnachbarschaft für jeden Punkt p

$$C = \frac{1}{k}\sum_{1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T$$

p_i: k Nachbarpunkte

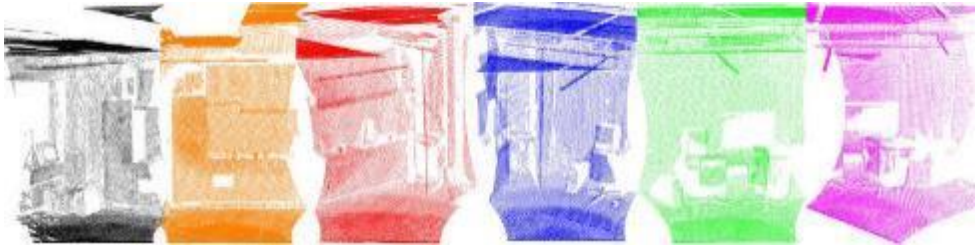
$\bar{p}$: Mittelpunkt aller k Nachbarn

- Bestimme die Eigenwerte und Eigenvektoren von C
  - Hauptkomponentenanalyse (Principle Component Analysis, PCA)
  - **Eigenvektor zu kleinstem Eigenwert korrespondiert zur Normalen**

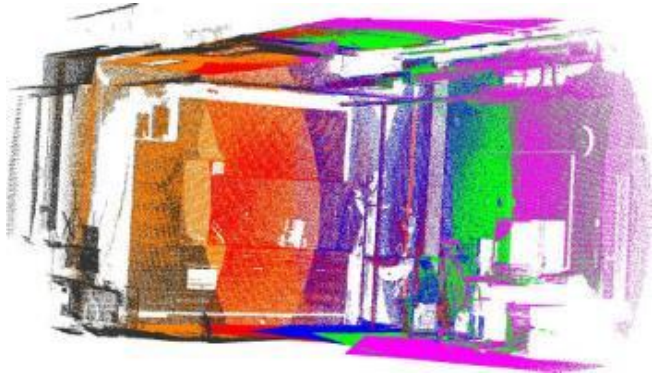


Registrierung von Punktwolken

- **Registrierung**: Zusammenführen von Punktwolken, welche das gleiche Objekt aus unterschiedlichen Ansichten beschreibt
- Überführung in ein übergeordnetes Koordinatensystem (z.B. Weltkoordinatensystem)
- Extrinsische Kalibrierung der Kameras erforderlich



Punktwolken eines Raumes aus unterschiedlichen Perspektiven



Registrierte Punktwolke

Iterative Closest Point

- **Iterative Closest Point** (ICP) ist ein gängiger Algorithmus für die **Registrierung** zweier Mengen A, B mit a priori unbekannter Zuordnung (Besl und McKay, 1992)

- **Beispiel:** Registrierung zweier 3D Punktwolken
  - Für jede Iteration k gilt:
    - Für jeden Punkt a_i aus A suche Punkt b_i aus B, der a_i am nächsten liegt
    - Berechne eine Transformation T_k, so dass D_k minimal wird, z.B. mit (Horn, 1987):

$$D_k = \sum_i \| a_i - T_k \cdot b_i \|^2$$

  - D_k kombiniert Translation und Rotation
  - Wende Transformation T_k auf alle Punkte aus B an (**Update**)
  - Abbruch:
    - Schwellwert für die Differenz $(D_{k-1} - D_k)$ oder
    - Maximale Anzahl an Iterationen erreicht

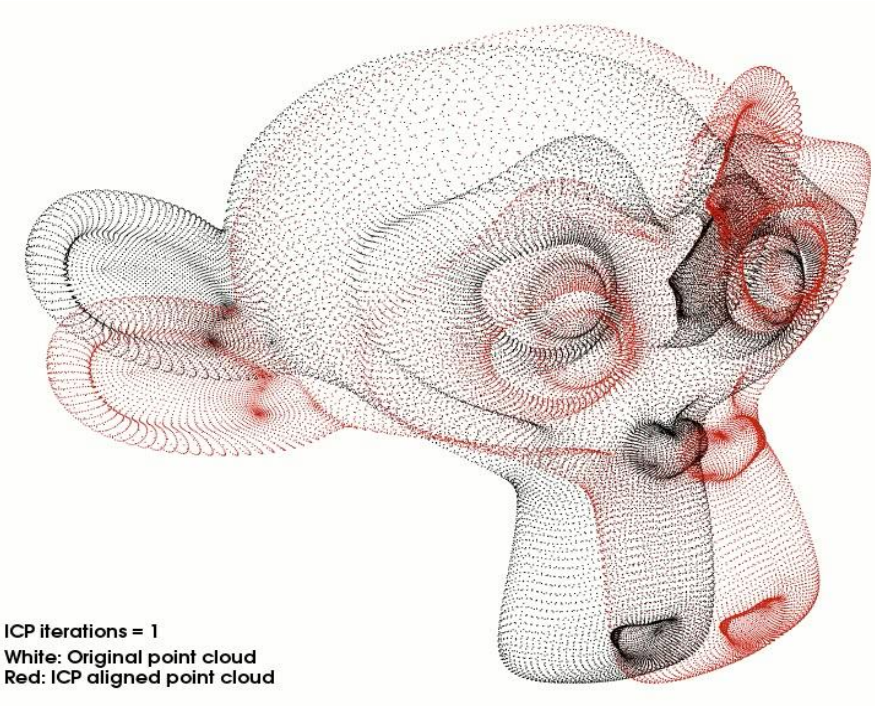
P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239 256, Feb. 1992, DOI: 10.1109/34.121791

Berthold K. P. Horn, Closed-Form Solution of Absolute Orientation Using Unit Quaternions, Journal of the Optical Society of America A 4(4):629-642; April 1987, DOI: 10.1364/JOSAA.4.000629

Iterative Closest Point (2)

- Minimiert die „Distanz“ zwischen zwei Punktwolken
- Sehr gut geeignet für die Rekonstruktion in 2D und 3D
- Vorteile
  - Algorithmus für Punkte, Normalenvektoren und andere Darstellungsformen anwendbar
  - Nur einfache mathematische Operationen notwendig
  - Schnelles Registrierungsergebnis
- Nachteile
  - Überlappung der Punktwolken erforderlich
  - Symmetrische Objekte können nicht ohne weiteres registriert werden
  - Konvergenz in ein lokales Minimum möglich

Iterative Closest Point – Visualisierung



RANSAC – Random Sample Consensus

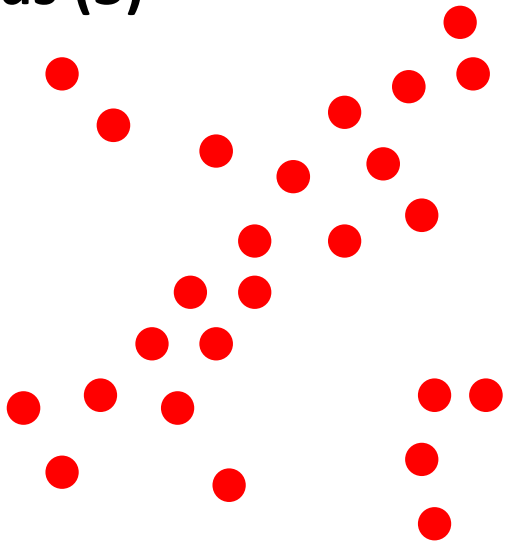
- **RANSAC** ist eine **iterative Methode** zur Schätzung von Modellparametern aus Datenpunkten
- RANSAC ist ein **nicht-deterministischer** Algorithmus, d.h. er liefert nicht immer das gleiche Ergebnis
- Robust gegenüber Ausreißern und fehlenden Datenpunkten
- Anwendung in der Bildverarbeitung
  - Schätzung von Linien in 2D Bildern
  - Schätzung von Ebenen und anderen Primitiven in 3D Punktwolken

RANSAC – Random Sample Consensus (2)

- Der RANSAC Algorithmus:
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist, um die Modellparameter zu berechnen, also
     2 Punkte für Linien in 2D und **3 Punkte** für Ebenen in 3D
  2. Schätze ein Modell aus dem ausgewählten Datensatz
  3. Bewertung der Modellschätzung:
     Berechne die Teilmenge der Datenpunkte (*Inliers*), deren Abstand zum Modell kleiner ist als ein vordefinierter Schwellwert
  4. Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC – Random Sample Consensus (3)

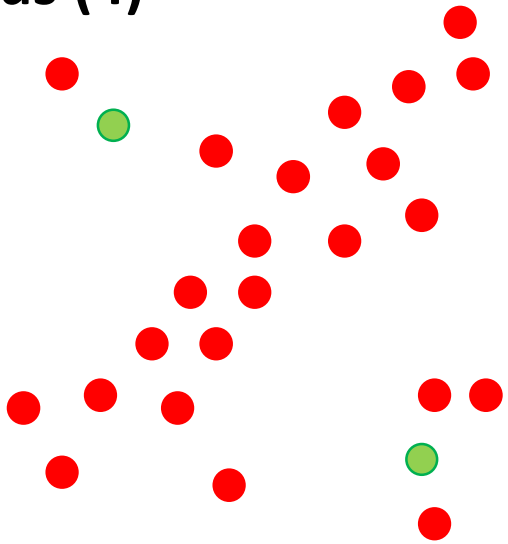
- Beispiel:
  - Line fitting in 2D Datenpunkte



- RANSAC Algorithmus:
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
  2. Schätzung des Modell aus dem ausgewählten Datensatz
  3. Bewertung der Modellschätzung
  4. Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC – Random Sample Consensus (4)

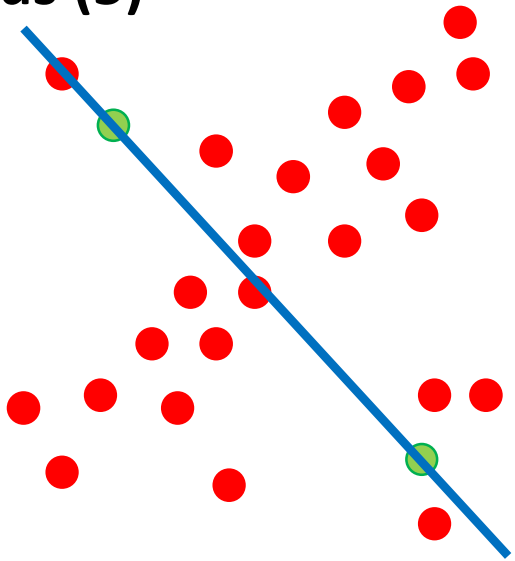
- Beispiel:
  - Line fitting in 2D Datenpunkte



- RANSAC Algorithmus:
  1. **Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen**
  2. Schätzung des Modell aus dem ausgewählten Datensatz
  3. Bewertung der Modellschätzung
  4. Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC – Random Sample Consensus (5)

- Beispiel:
  - Line fitting in 2D Datenpunkte

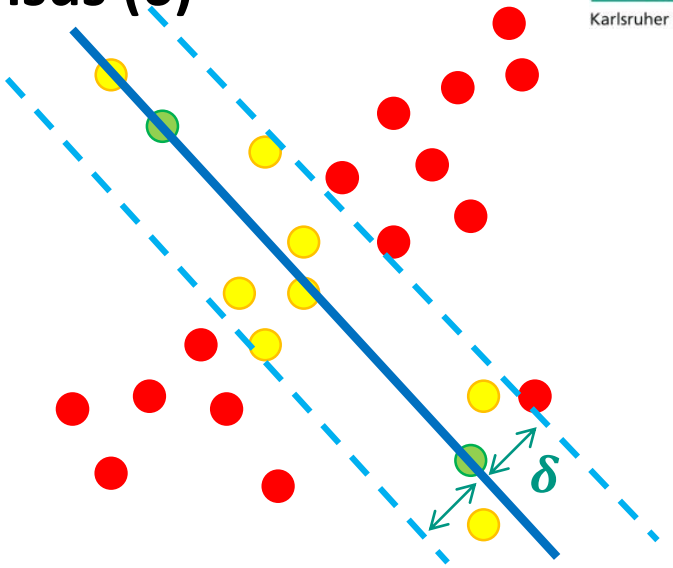


- RANSAC Algorithmus:
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
  2. **Schätzung des Modell aus dem ausgewählten Datensatz**
  3. Bewertung der Modellschätzung
  4. Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC – Random Sample Consensus (6)

- Beispiel:
  - Line fitting in 2D Datenpunkte

Inlier Anzahl = 10
Outlier Anzahl = 15



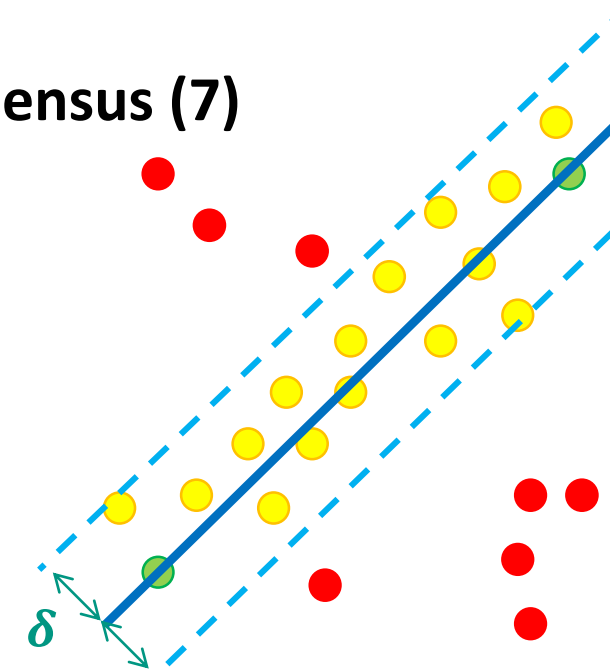
- RANSAC Algorithmus:
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
  2. Schätzung des Modell aus dem ausgewählten Datensatz
  3. **Bewertung der Modellschätzung**
  4. Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC – Random Sample Consensus (7)

- Beispiel:
  - Line fitting in 2D Datenpunkte

Inlier Anzahl = 17

Outlier Anzahl = 8



- RANSAC Algorithmus:
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
  2. Schätzung des Modell aus dem ausgewählten Datensatz
  3. Bewertung der Modellschätzung
  4. **Wiederhole 1–3 bis das Modell mit den meisten Inliers gefunden wird**

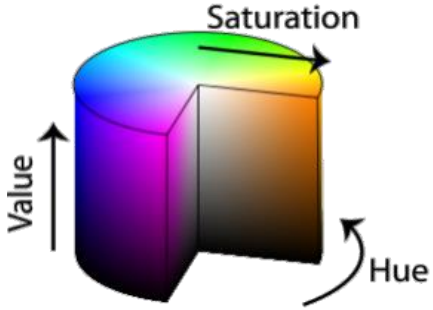
RANSAC – Random Sample Consensus (5)

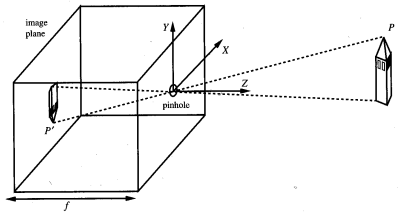
- Vorteile:
  - Allgemein und einfach zu implementieren
  - Robuste Modellschätzung für Daten mit wenigen Ausreißern
  - Vielseitig anwendbar

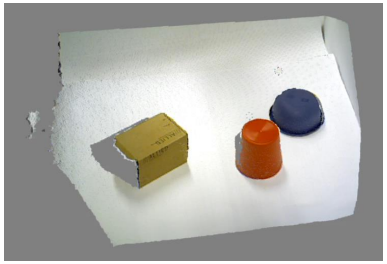
- Nachteile:
  - Nicht-deterministisch
  - Trade-off zwischen Genauigkeit und Laufzeit (benötigt viele Iterationen)
  - Nicht anwendbar wenn das Verhältnis Inliers/Outliers zu klein ist

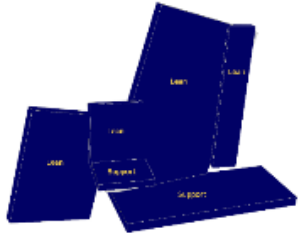
Inhalt

- Bilderzeugung
- Operationen auf Bildern
- Merkmalsextraktion und Mustererkennung
  - Segmentierung
  - Canny-Kantendetektor
  - Visual Servoing
  - Registrierung von Punktwolken
  - **Beispielanwendungen H²T**

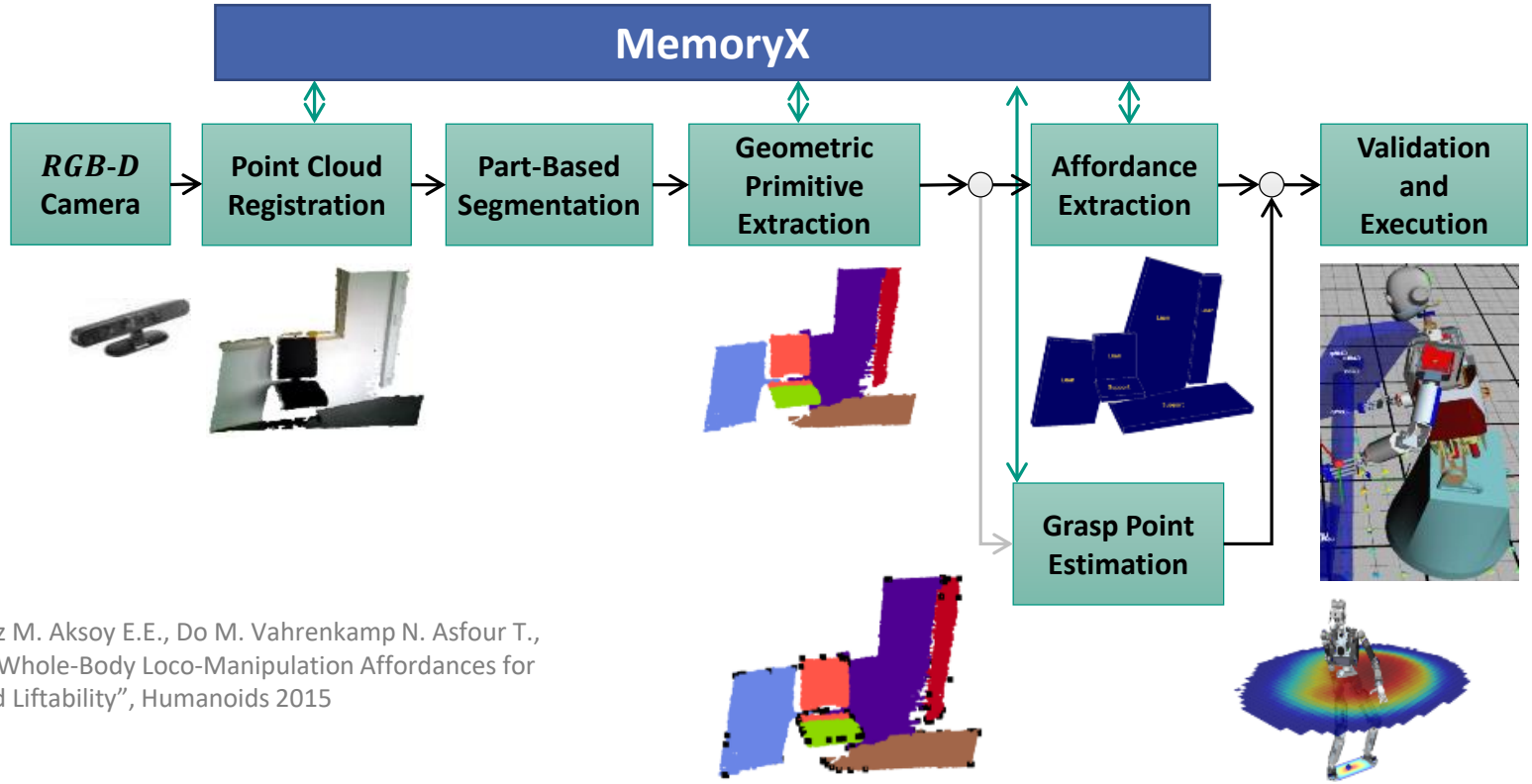






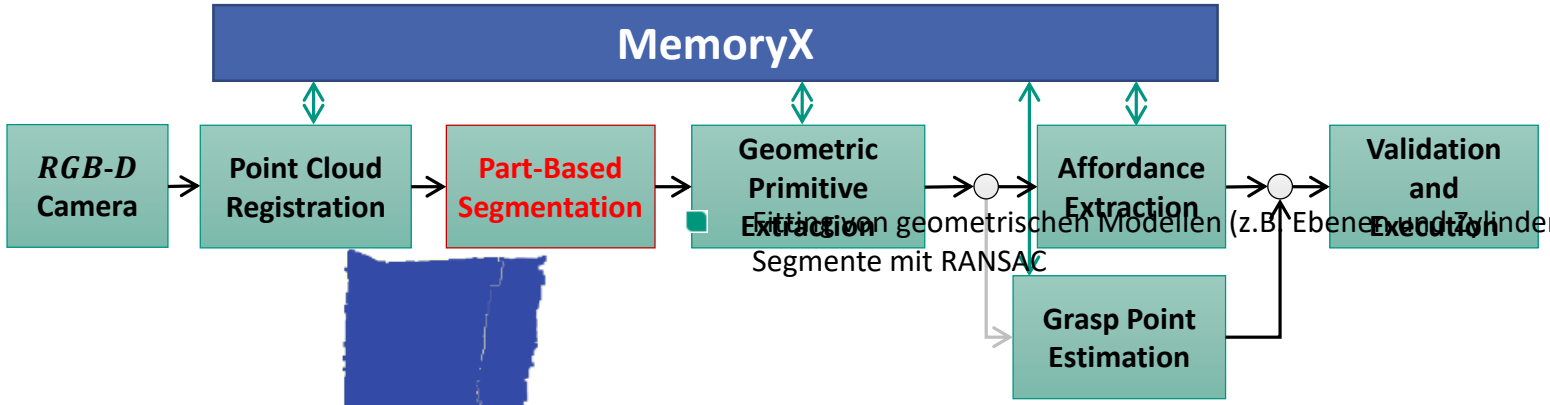


Loco-Manipulation Affordances



Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T., "Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability", Humanoids 2015

Loco-Manipulation Affordances (2)

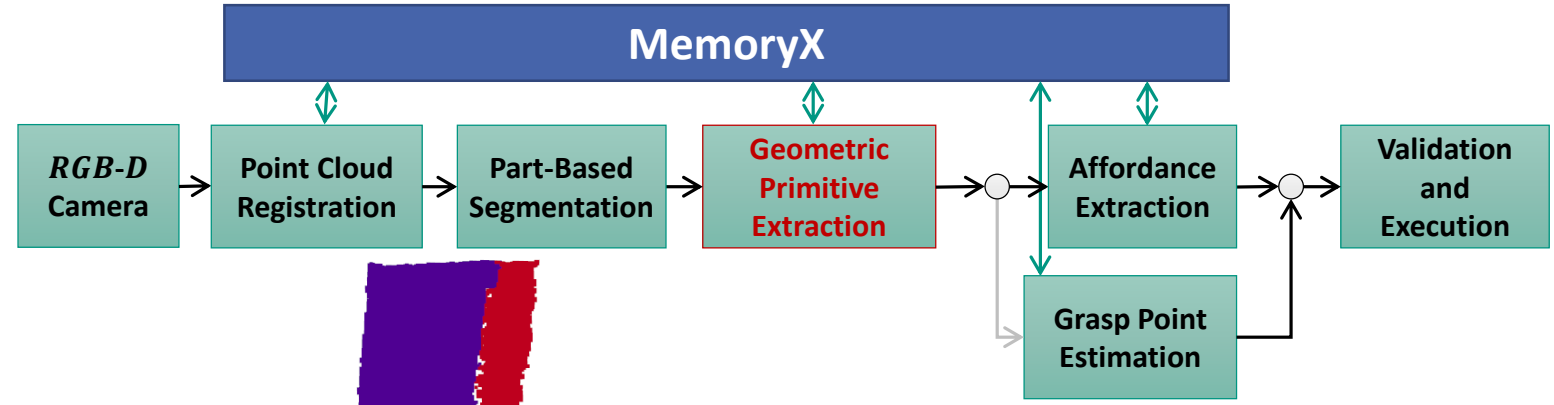


- Fitting von geometrischen Modellen (z.B. Ebenen und Zylinder) in Segmente mit RANSAC

- Tiefenbasierte Segmentierung: benötigt für Szenenzerlegung
- Wir verwenden Locally Convex Connected Patches (LCCP) zur Segmentierung
- Vorteil: Einfach und schnell
- Nachteil: Untersegmentierung (z.B. Stuhl)

Segmentation: S. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius, "Convexity based object partitioning for robot applications," ICRA 2014.

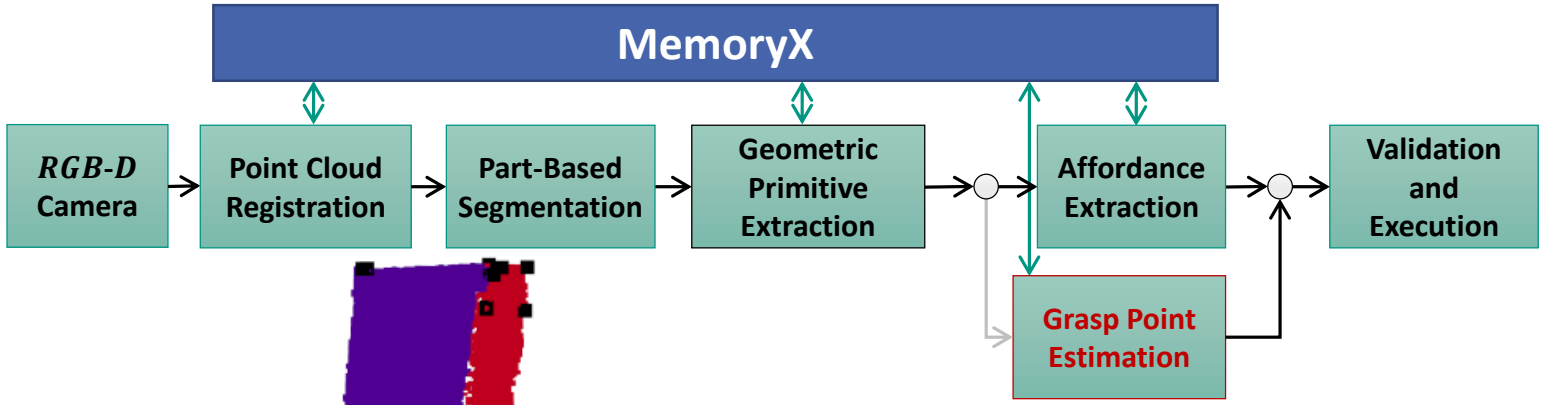
Loco-Manipulation Affordances (3)

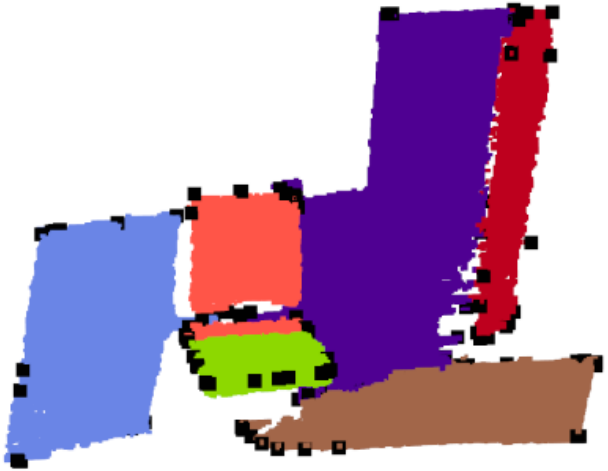


- Fitting von geometrischen Modellen (z.B. Ebenen und Zylinder) in Segmente mit RANSAC
- Wegen Untersegmentierung werden Modelle iterativ an die verbleibenden Outliers gefittet (z.B. Stuhl)

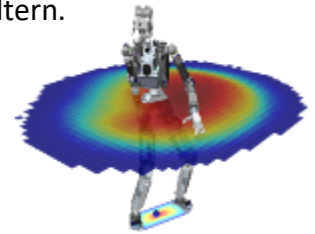
PCL: R. B. Rusu and S. Cousins, “3d is here: Point Cloud Library (PCL),” ICRA 2011

Loco-Manipulation Affordances (4)

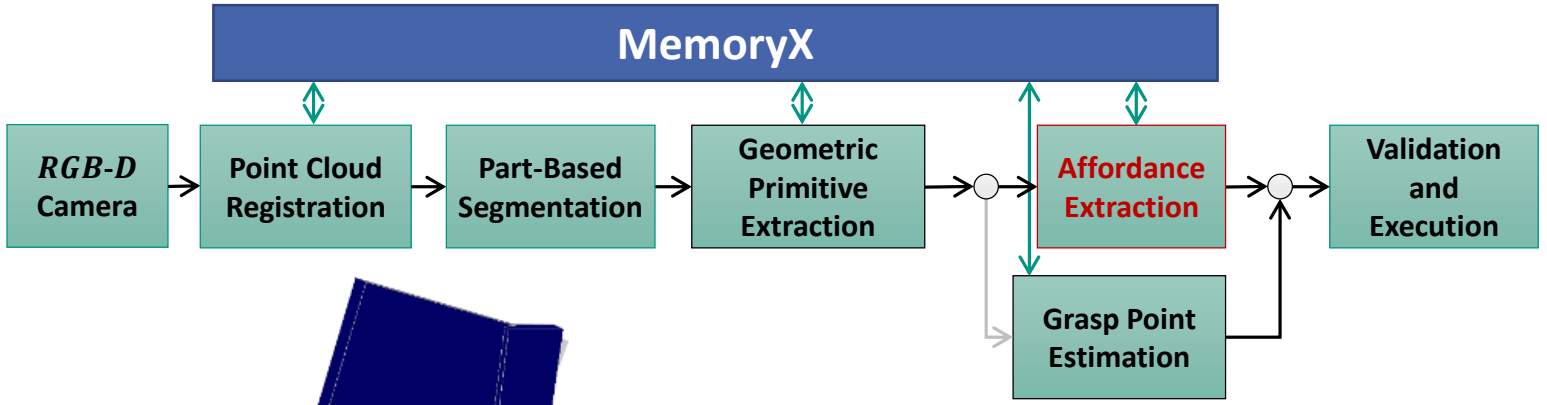


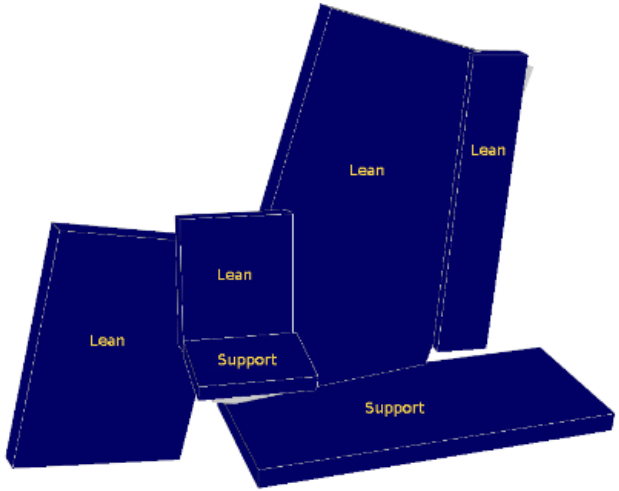


- Berechne konvexe Hülle für die projizierten Inlier Punkte um *Grasp Points* zu schätzen. Vgl. Schwarze Punkte am Stuhl.
- Invertierte Reachability Map um Punkte zu filtern.



Loco-Manipulation Affordances (5)

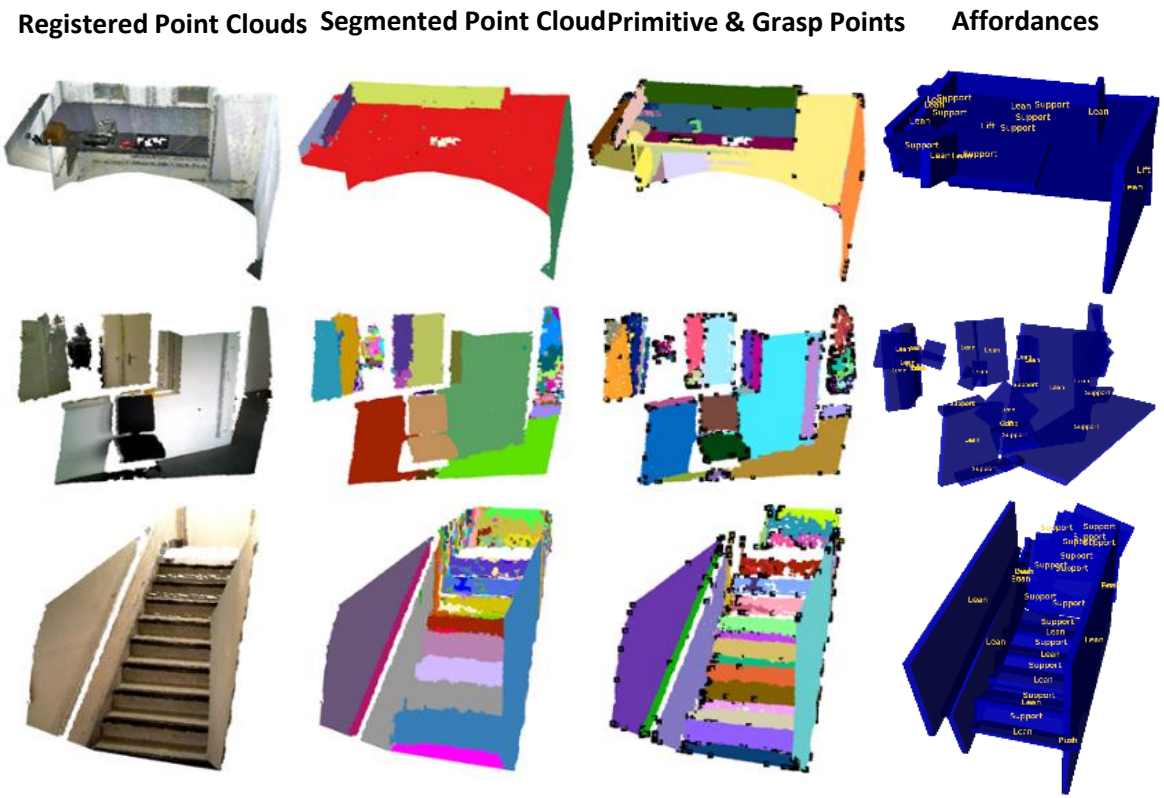




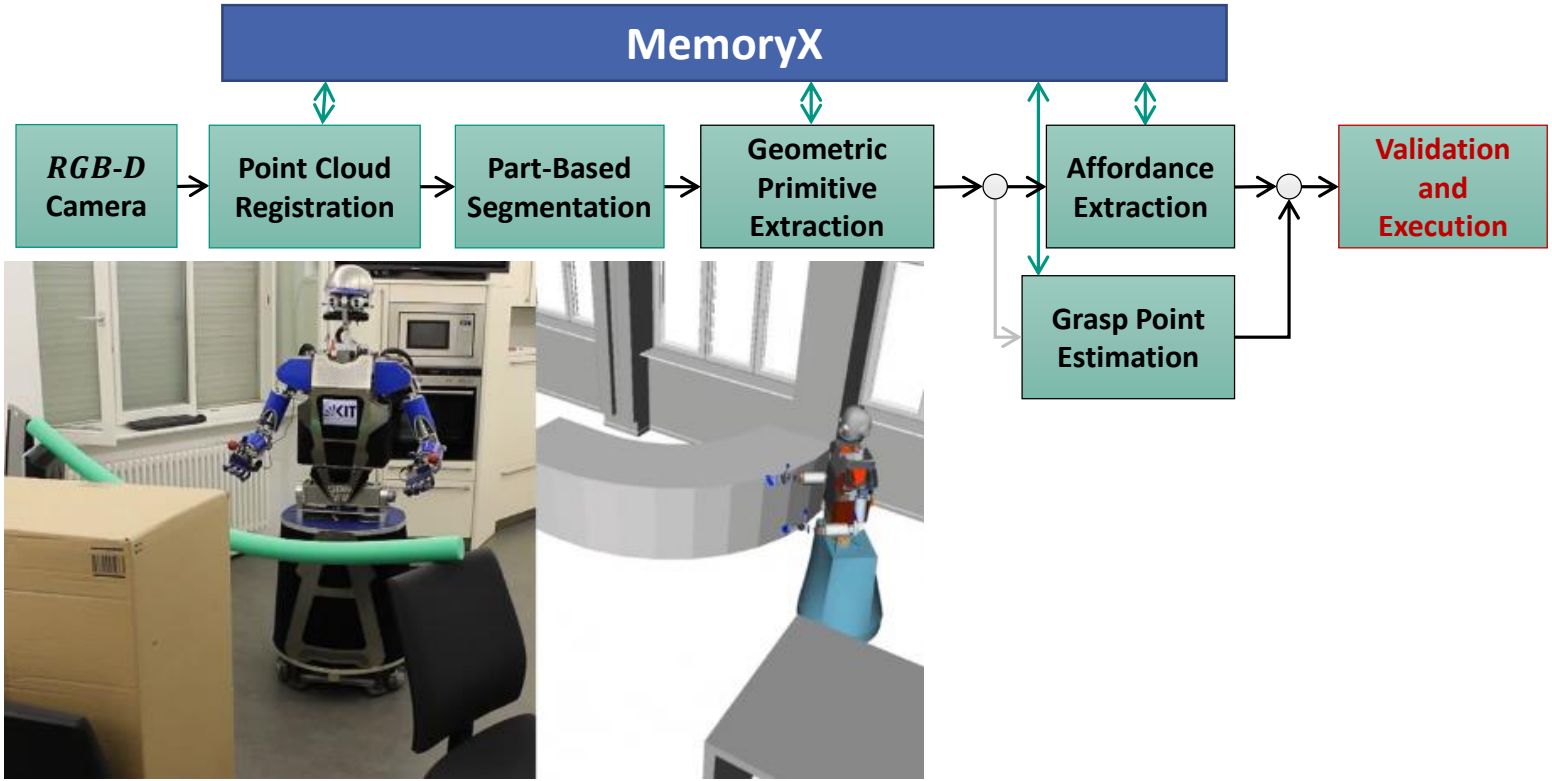
Look-up Tabelle!

Affordance	Shape	Parameters	Conditions	Valid.
Support	Planar	Normal $\boldsymbol{n}$ Area a	$\boldsymbol{n} \uparrow \boldsymbol{z}_{world}$ $a \geq \lambda_1$	(1a)
Lean	Planar	Normal $\boldsymbol{n}$ Area a	$\boldsymbol{n} \perp \boldsymbol{z}_{world}$ $a \geq \lambda_2$	(1a)
Grasp	Planar	Normal $\boldsymbol{n}$ Area a	$a \in [\lambda_3, \lambda_4]$	(3)
	Cylindrical	Radius r Direction $\boldsymbol{d}$	$r \in [\lambda_5, \lambda_6]$ $\|\boldsymbol{d}\| \leq \lambda_7$	
	Spherical	Radius r	$r \in [\lambda_8, \lambda_9]$	
Hold	Cylindrical	Radius r Direction $\boldsymbol{d}$	$r \in [\lambda_{10}, \lambda_{11}]$ $\|\boldsymbol{d}\| \geq \lambda_{12}$	(2a)
Push	Planar	Normal $\boldsymbol{n}$ Area a	$\boldsymbol{n} \perp \boldsymbol{z}_{world}$ $a \leq \lambda_{13}$	(1b)

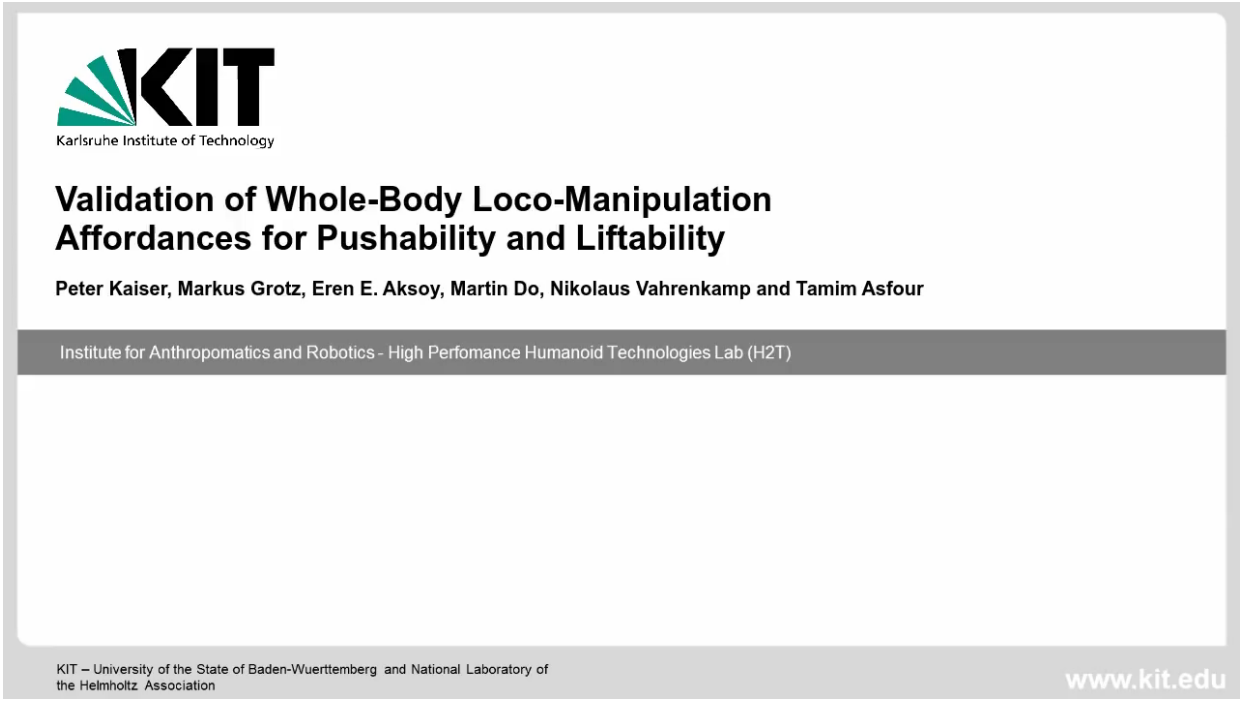
Loco-Manipulation Affordances (6)



Loco-Manipulation Affordances (7)



Loco-Manipulation Affordances (8)



Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T., "Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability", Humanoids 2015

Englische Begriffe

Deutsch	Englisch
Farbnuance	hue
Sättigung	saturation
Helligkeit	intensity/value
Hauptachse	principal axis
Hauptpunkt	principal point
Bildkoordinaten	image coordinates
Kamerakoordinaten	camera coordinates
Weltkoordinatensystem	world coordinate system
Schwellenwertverfahren	thresholding
Punktwolken	point clouds
Kartierung	mapping
Orientierungspunkt	landmark