

Robotik I – Bildverarbeitung in der Robotik

Tamim Asfour

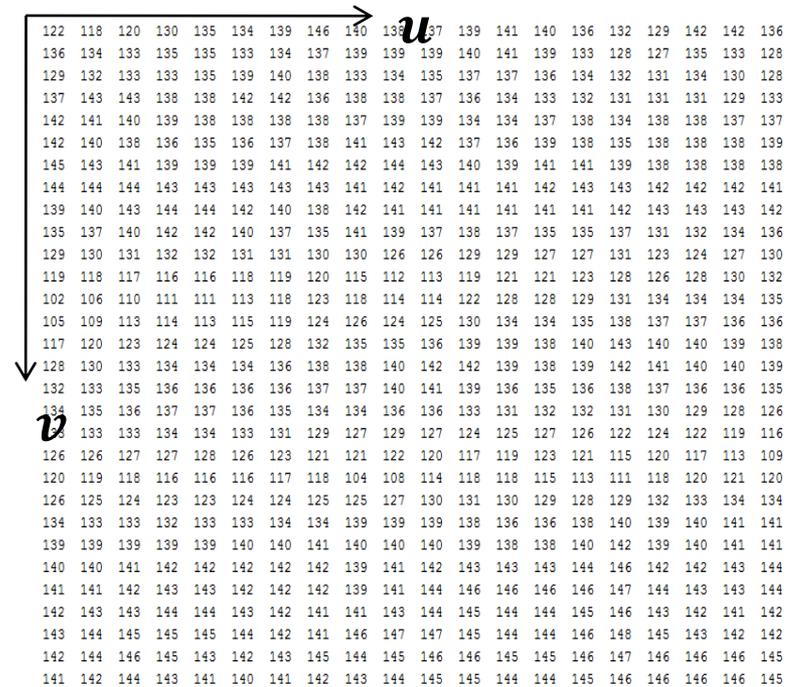
KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)



Bildverstehen



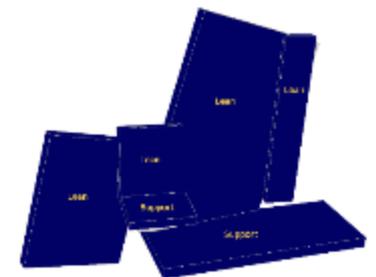
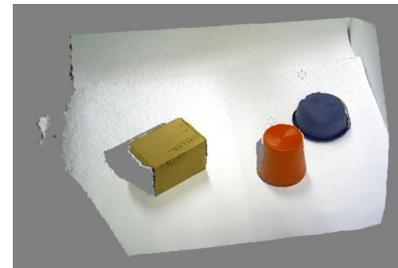
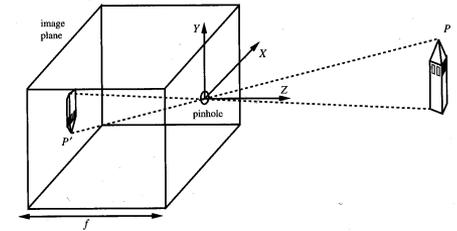
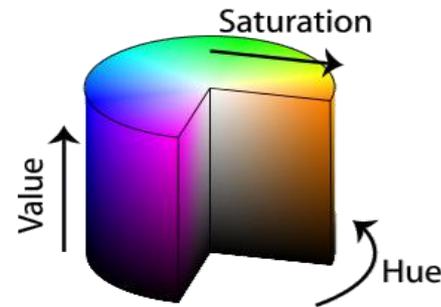
Was wir sehen!



Was ein Roboter sieht!

Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



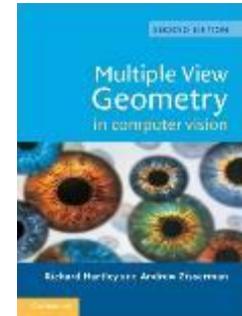
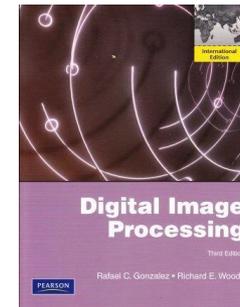
Literatur

- 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. Puento León und C. Frese

- Computer Vision – Das Praxisbuch
 - Pedram Azad, Tilo Gockel und Rüdiger Dillmann



Programmbibliotheken

■ OpenCV

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



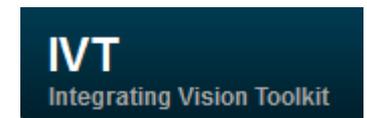
■ Point Cloud library (PCL)

- <http://pointclouds.org>
- Pointcloud processing, RANSAC primitive fitting, ICP



■ Integrating Vision Toolkit (IVT)

- <http://ivt.sourceforge.net>
- Image processing, Feature matching



Motivation

Was ist das Problem?

- Wie können wir einem Roboter „Sehen“ ermöglichen?

Warum ist *Computer Vision* schwer?

- Computer Vision ist nicht nur Bildübertragung von Kamera auf den Computer sondern auch deren **Verarbeitung** und **Interpretation**
- Bei der Übertragung (3D Welt → 2D Bild) geht eine Dimension verloren

Unterschiedliche Aspekte

- **Photometrie:** Wahrnehmung abhängig von Beleuchtung und Materialeigenschaften
- **Dynamik:** Bewegung in der Szene bzw. in der Kamera
- **Okklusion:** Ein Objekt kann z.B. nicht vollständig sichtbar sein
- Interpretation Abhängig von der aktuellen Aufgabe

Motivation

- Das Sehvermögen ermöglicht die Wahrnehmung der Umwelt
- Nutzbarkeit in einem technischen System
 - Visuellen Informationen müssen aufgenommen werden
 - gute Qualität
 - digitales Format
 - Relevante Informationen müssen aus den Daten extrahiert werden
- **Bilderfassung:** Hardware
- **Bildverarbeitung:** überwiegend Software



Lena-Bild

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

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



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

Kamerabeispiele



PointGrey Flea3 USB3



CMU MultiSense S7



Karlsruhe Humanoid Head



Intel RealSense R200



Roboception rc visard

MultiSense S7 and S7S



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

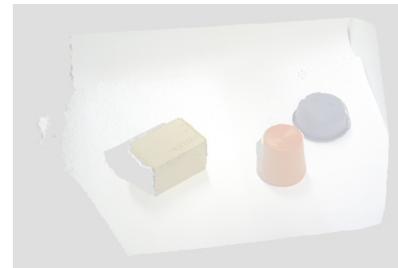
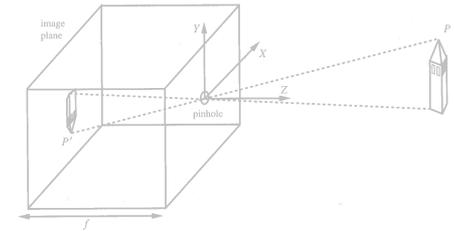
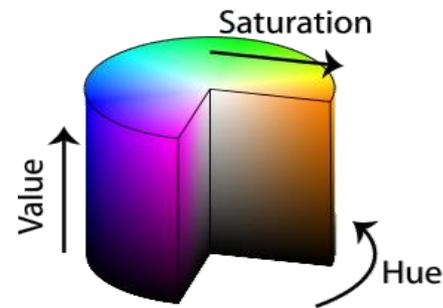
Roboception: rc visard 65 - Bin picking application



<https://www.youtube.com/watch?v=-leZbn1aA8Q>

Inhalt

- **Bildrepräsentation**
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Bildrepräsentation

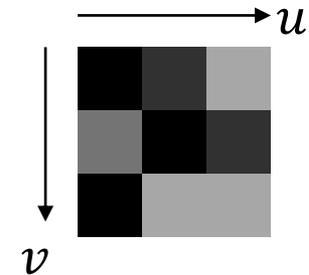
- Bilder müssen im Computer/Roboter repräsentiert werden
- Ein Bild ist ein 2D Gitter von diskreten Punkten (**Pixel**)
- **Bildkoordinaten** (hier):
 - u (horizontal)
 - v (vertikal)
 - Ursprung ist oben links
 - Einheiten: Pixel
- Die **Farbe** eines Bildpunktes kann auf unterschiedliche Weise repräsentiert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

Bildrepräsentation - 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:

$$q = 255$$

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

oder $n = 1920, \quad m = 1080 \text{ (1080p „Full HD“)}$

$$n = 3840, \quad m = 2160 \text{ (2160p „Ultra HD“ oder 4K UHD)}$$

Datenmenge

- Schwarz/Weiss mit 30 Hz, 640x480 Pixel und 1 Byte pro Pixel

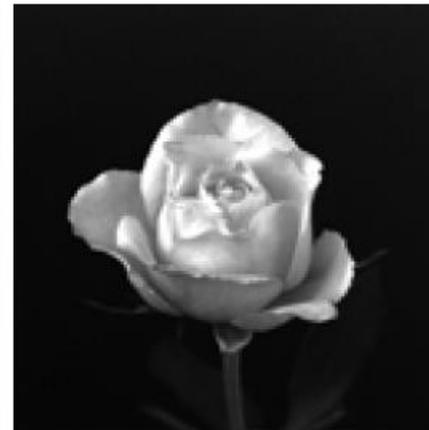
$$30\text{Hz} * 640 * 480 * 1\text{Byte} \approx 8.79 \text{ MB/s}$$

- Farbbilder mit 30 Hz, 640x480 Pixel und 3 Byte pro Pixel

$$30\text{Hz} * 640 * 480 * 3\text{Byte} \approx 26.37 \text{ MB/s}$$

Bildrepräsentation - Auflösung

- Die Abtastung ist der Hauptfaktor, der die räumliche Auflösung eines Bildes bestimmt.
- Die Auflösung ist das kleinste erkennbare Detail in einem Bild.



32

64

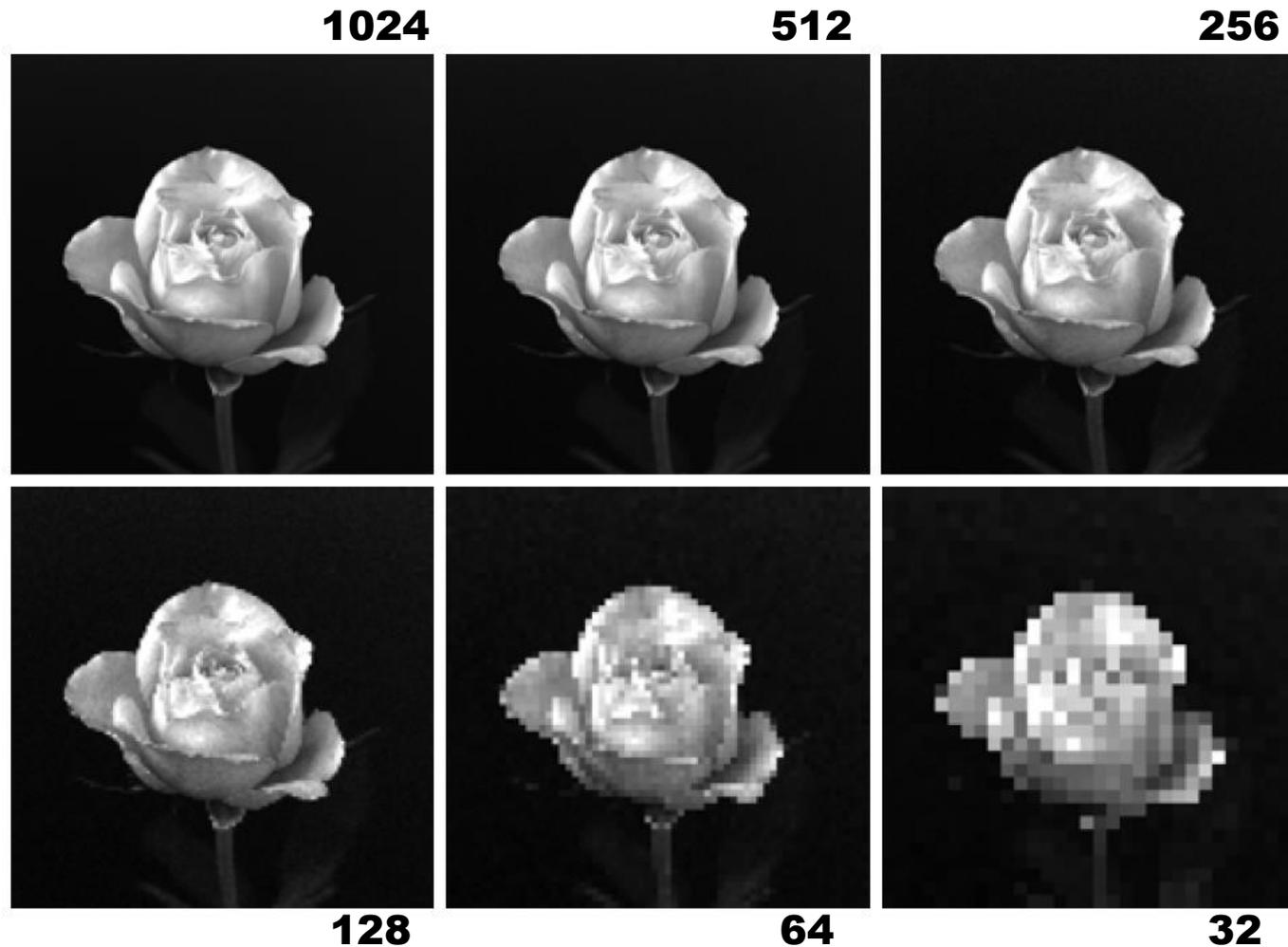
128

256

512

1024

Bildrepräsentation - Auflösung II



- Wenn man die Anzahl der Graustufen konstant hält und die räumliche Auflösung verringert, ergibt sich ein Schachbretteffekt!

Bildrepräsentation - Farbbild

Farbbild:

- Verschiedene Farbmodelle für unterschiedliche Anwendungen
- Klassifikation nach erreichbarem Farbraum

Beispiele:

- *RGB*-Modell (Rot-Grün-Blau-Modell): speziell für Monitore (Phosphor-Kristalle)

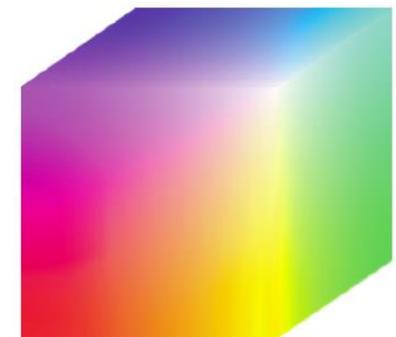
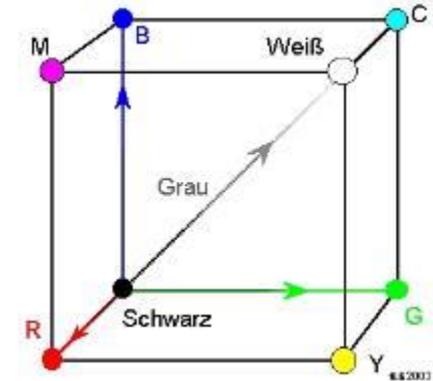
$$\text{Img}(u, v) = (r, g, b)^T \in \mathbb{R}^3$$

- *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

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



24-Bit *RGB* Farbwürfel

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

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

Bildrepräsentation – HSI Farbraum

HSI (HSV) Farbraum

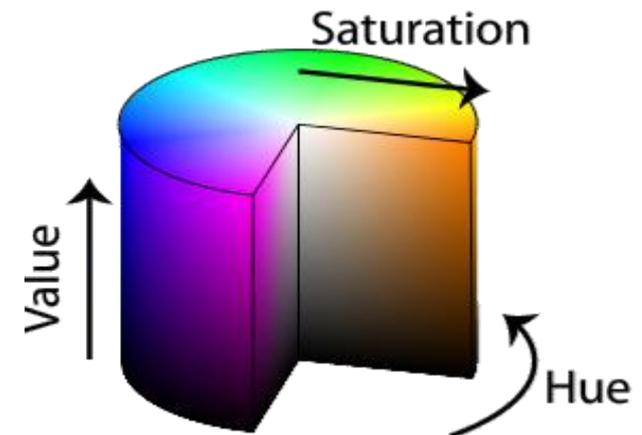
- Hue (Farbnuance), Saturation (Sättigung), Intensity/Value (Lichtintensität/Helligkeit)
- Kodiert der Farbinformation **getrennt** von Helligkeit und Sättigung der Farbe
 - ⇒ unempfindlich gegen Beleuchtungsänderungen
- Umrechnung von *RGB* nach *HSI*
 - *H* undefiniert, falls $R = G = B$
 - *S* undefiniert,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

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

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

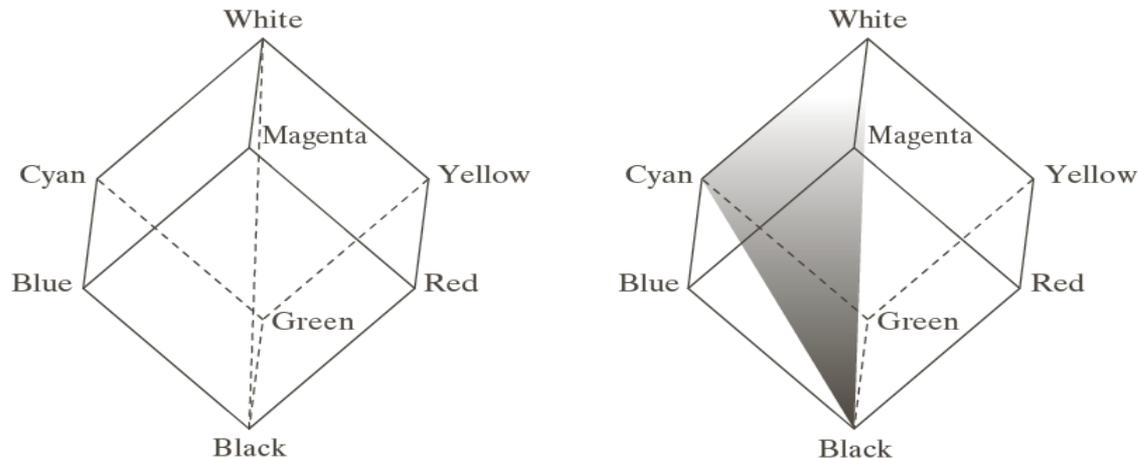
$$S = 1 - \frac{3}{R + G + B} \min(R, G, B)$$

$$I = \frac{1}{3}(R + G + B) \quad V = \max(R, G, B)$$



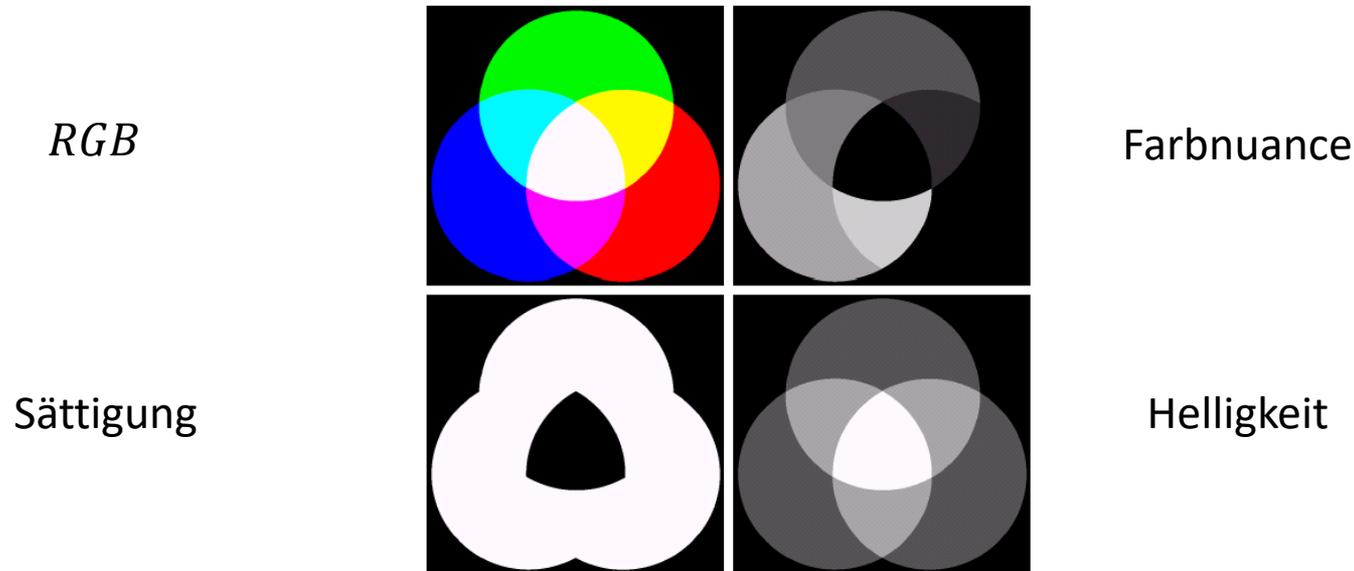
Bildrepräsentation – HSI nach RGB

Wie bestimmt man *HSI* Werte in einem *RGB* Würfel?



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

Bildrepräsentation - Unterschiede



- 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!

Robotik I – Bildverarbeitung in der Robotik

Fabian Paus, Tamim Asfour

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)



Bildrepräsentation

Repräsentation eines 8-Bit Graustufen-Bildes im Speicher

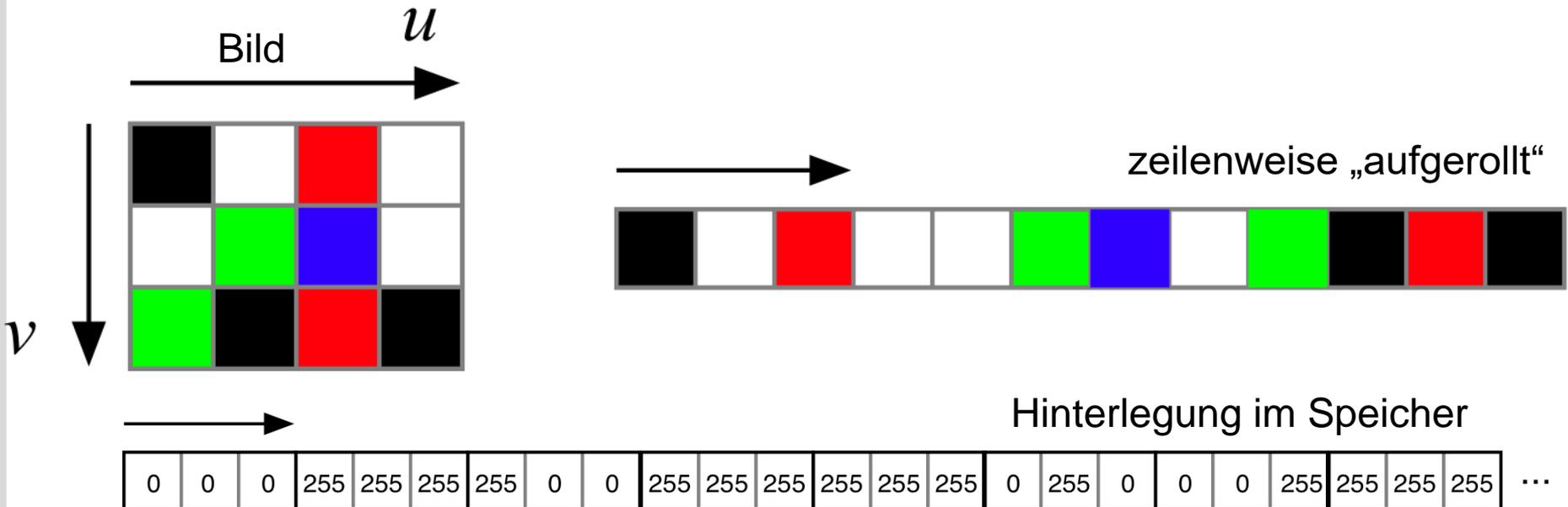
- Pixel werden zeilenweise, linear abgelegt
 - von oben links nach unten rechts
 - Achtung: z.B. bei Bitmaps von unten links nach oben rechts
- Graustufen-Kodierung:
 - Ein Byte pro Pixel
 - 0 schwarz, 255 weiß, dazwischen Graustufen



Bildrepräsentation

Repräsentation eines *RGB24* Farbbildes im Speicher

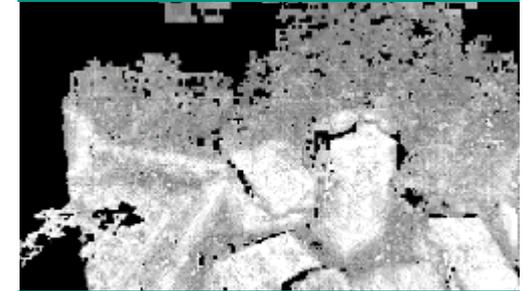
- Pixel werden zeilenweise, wie beim Graustufen-Bild, abgelegt
- Farbkodierung:
 - Drei Bytes pro Pixel
 - Für jeden Kanal gilt: 0 minimale, 255 maximale Intensität



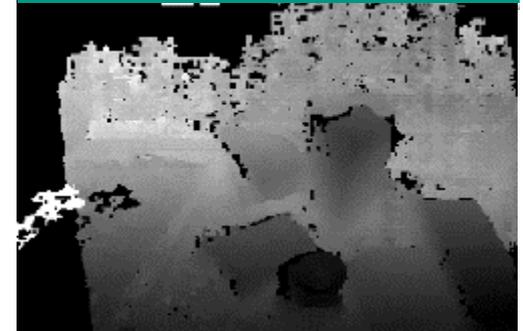
Beispiel: Bildrepräsentation bei rc visard 160

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

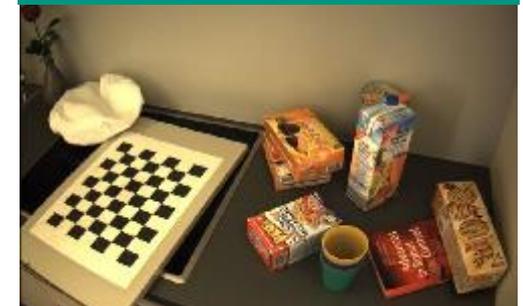
Konfidenzbild



Tiefenbild

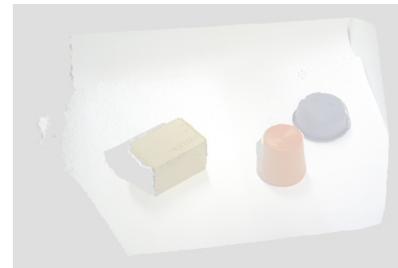
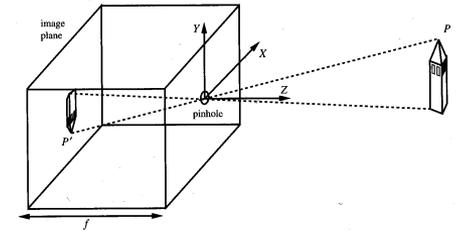
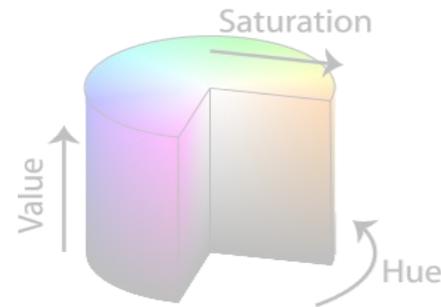


Linke Kamera



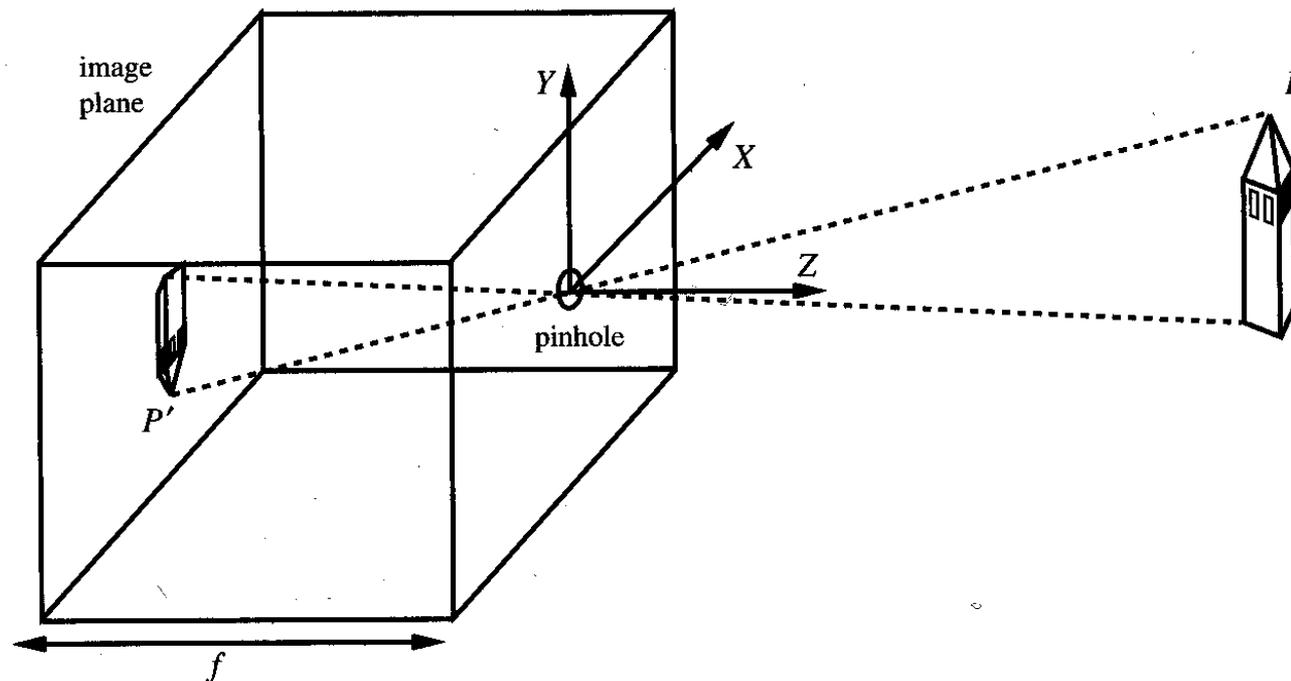
Inhalt

- Bildrepräsentation
- **Kameramodell**
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Bildgenerierung: Lochkamera

- Einfachstes Modell: **Lochkamera-Modell**

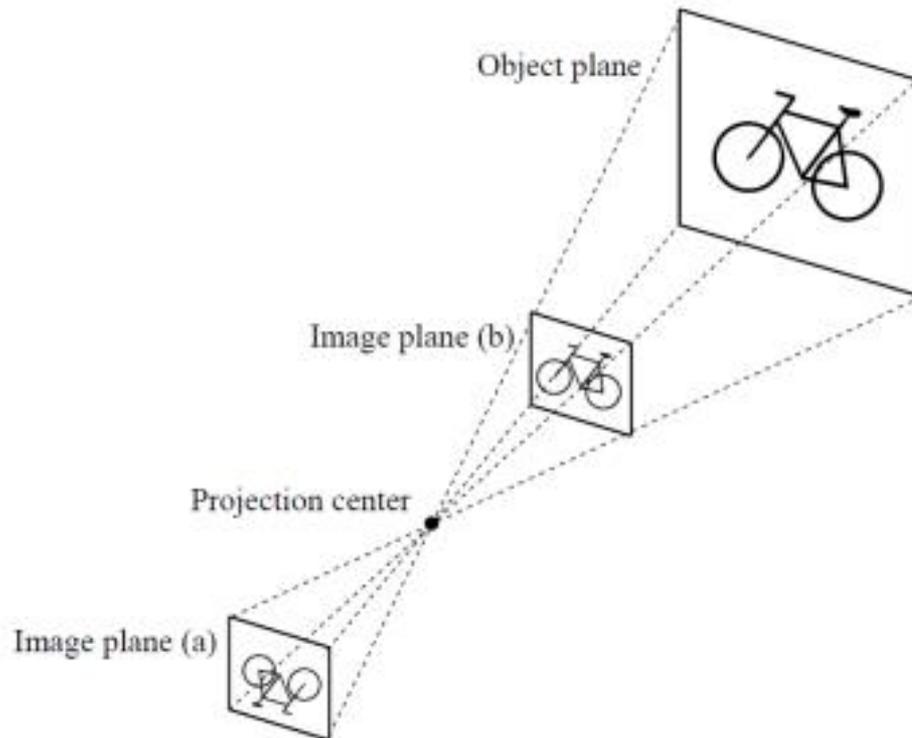


Interner Parameter: **Brennweite f** („Fokalabstand“)

Bildgenerierung: Klassisches Lochkamera Modell

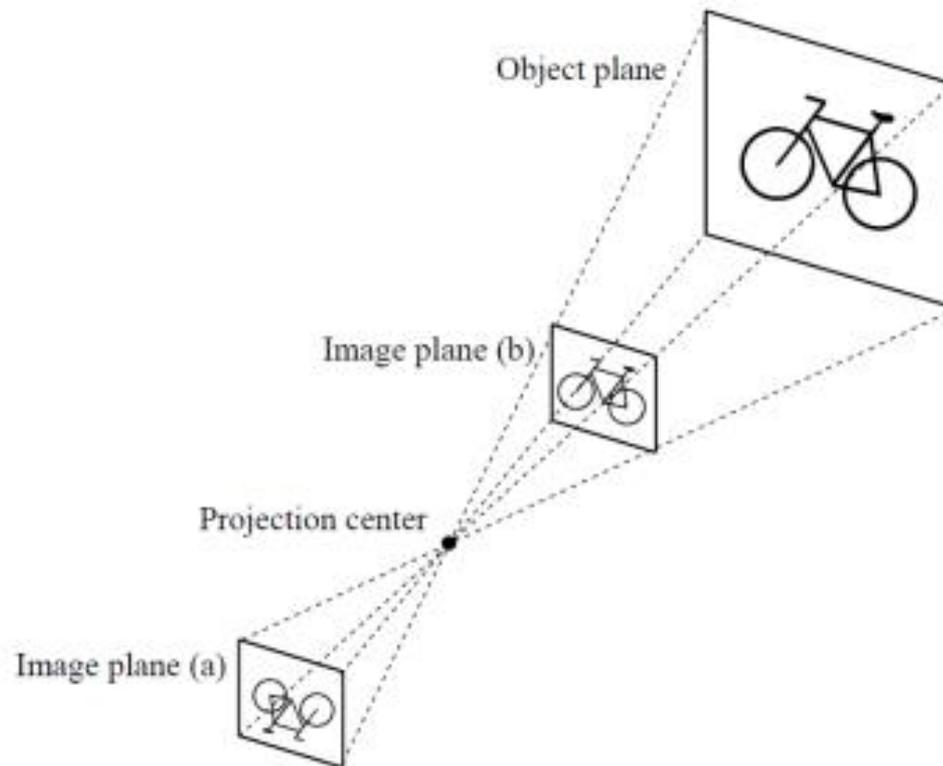
■ Klassisches Lochkamera-Modell

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



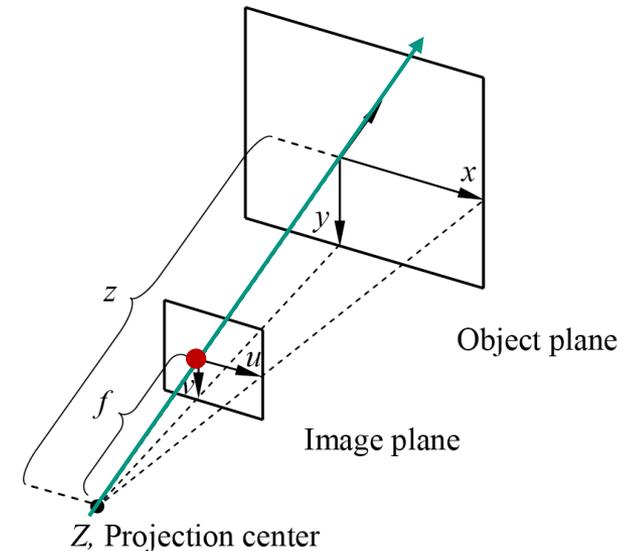
Bildgenerierung: Lochkamera in Positivlage

- Oft verwendete Variante: Lochkammermodell 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

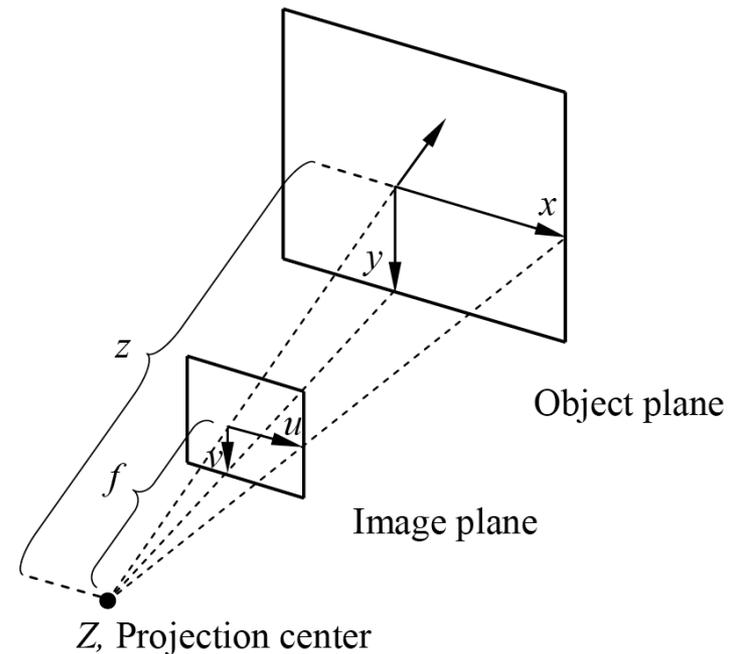
Zweiter Strahlensatz zur Projektion eines Szenenpunktes (x, y, z) auf einen Bildpunkt (u, v) :

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

Bei der Projektion geht die z -Komponente verloren!

Rückprojektion:

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



Kameraparameter

- Parameter, die das Kameramodell vollständig beschreiben
- Man unterscheidet zwischen **intrinsischen** und **extrinsischen** Kameraparameter
 - **Intrinsische Parameter:** Sie sind kameraspezifisch; unabhängig von der Auswahl des Weltkoordinatensystems. Sie bleiben konstant bei Veränderung des Aufbaus
 - **Extrinsische Parameter:** Modellieren die Transformation von Weltkoordinatensystem in das Kamerakoordinatensystem. Sie sind bei einer neuen Lage der Kamera neu zu bestimmen.
- Bisher:
 - Im Lochkameramodel nur ein intrinsischer Kameraparameter (f)
 - Kein Weltkoordinatensystem berücksichtigt, deshalb keine extrinsischen Parameter
 - Annahmen: Hauptpunkt im Ursprung des Bildkoordinatensystem; Pixel sind exakt quadratisch; Keine Linsenverzerrung.

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

- 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

- 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

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

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

- 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

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

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

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

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

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:

$$P = K (R|t)$$

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

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

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

Kamerakalibrierung

- 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

- Die beiden Gleichungen lassen sich weiter umstellen zu

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

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

Kamerakalibrierung

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

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 \\ \dots & \dots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n \end{pmatrix}$$

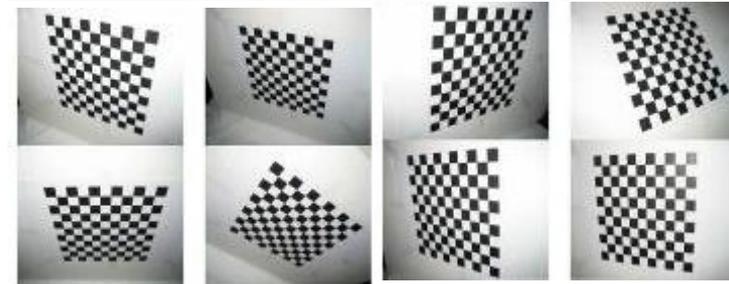
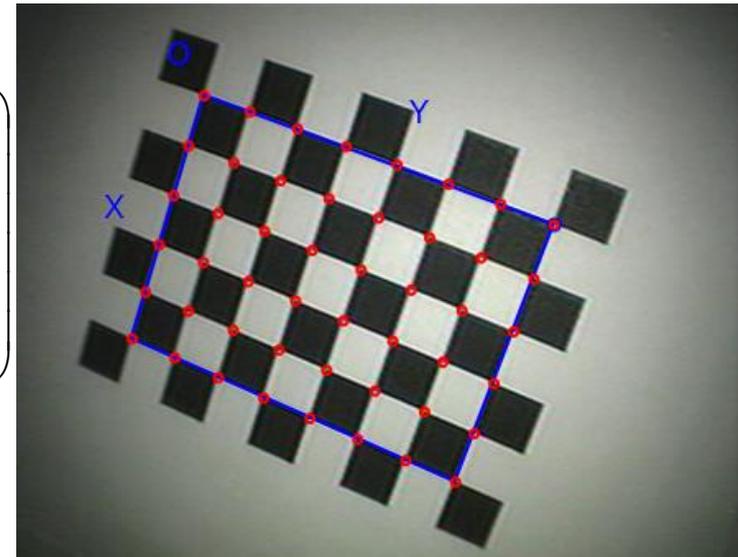
$$\mathbf{x} = \begin{pmatrix} p_1 \\ \dots \\ p_{11} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} u_1 \\ v_1 \\ \dots \\ u_n \\ v_n \end{pmatrix}$$

Kamerakalibrierung

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

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 \\ \dots & \dots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} p_1 \\ \dots \\ p_{11} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} u_1 \\ v_1 \\ \dots \\ u_n \\ v_n \end{pmatrix}$$



Kamerakalibrierung

- Die optimale Lösung x^* mit Hilfe der Kleinste-Quadrate-Methode für ein solches **überbestimmtes lineares Gleichungssystem** ergibt sich aus der Lösung von

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

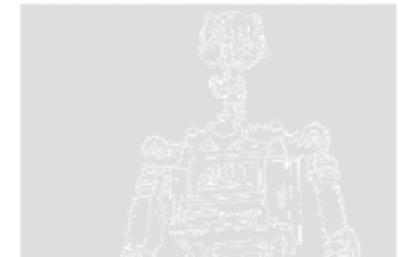
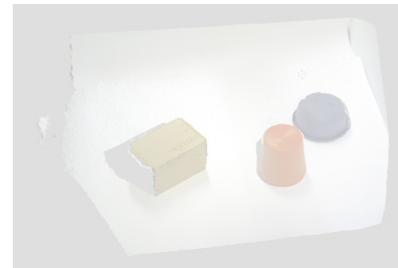
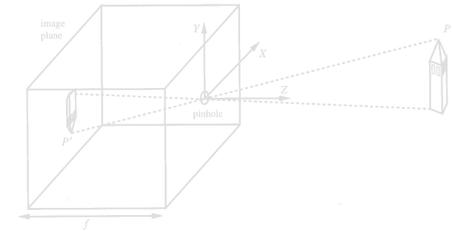
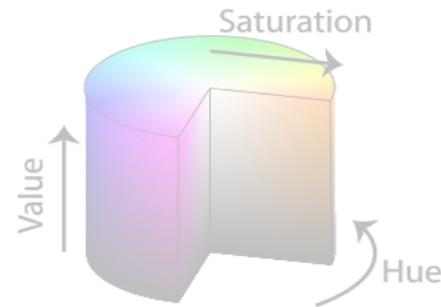
was sich auflösen lässt zu

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

wobei $(A^T A)^{-1}$ mit der **Moore-Penrose Pseudoinversen** bestimmt werden kann

Inhalt

- Bildrepräsentation
- Kameramodell
- **Filteroperationen**
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Filteroperationen

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

$$f(x + y) = f(x) + f(y)$$

$$f(\alpha x) = \alpha f(x)$$



additiv



homogen

Filter

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

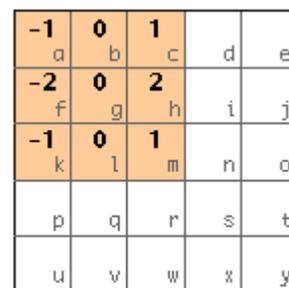
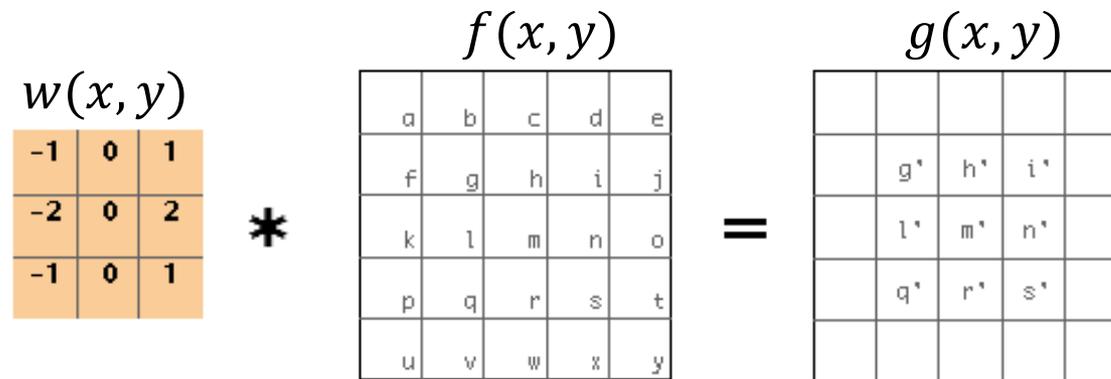
- Beispiel:

$$\begin{array}{|c|c|c|} \hline w(x,y) & & \\ \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|c|} \hline f(x,y) & & & & \\ \hline a & b & c & d & e \\ \hline f & g & h & i & j \\ \hline k & l & m & n & o \\ \hline p & q & r & s & t \\ \hline u & v & w & x & y \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline g(x,y) & & & & \\ \hline & & & & \\ \hline & g' & h' & i' & \\ \hline & l' & m' & n' & \\ \hline & q' & r' & s' & \\ \hline & & & & \\ \hline \end{array}$$

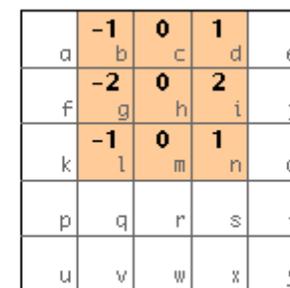
Filter

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

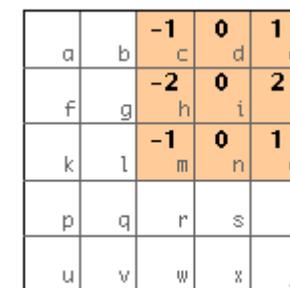
- Beispiel:



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



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



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

Filteroperationen - Grundlagen

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

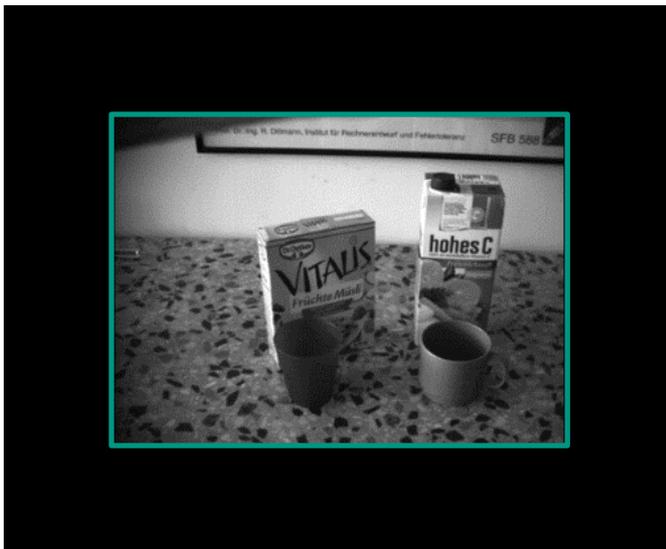
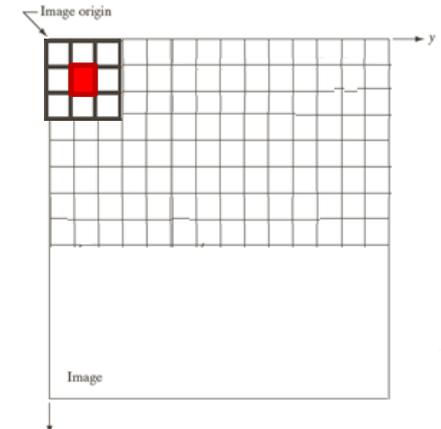
$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) \cdot f(x + s, y + t)$$

- Die **Faltung** ist der gleiche Vorgang, nur dass der 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 - Ränder

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



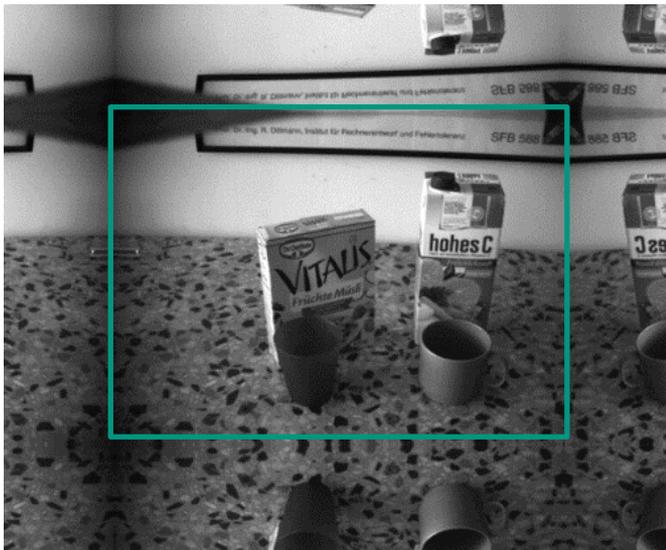
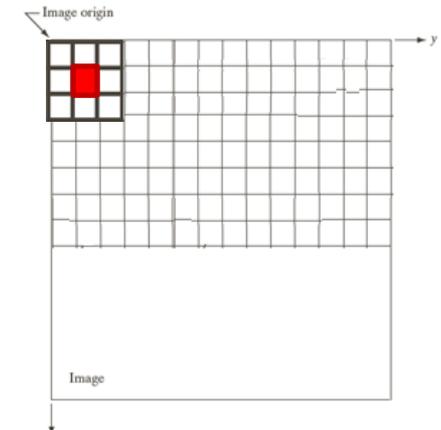
Constant



Wrap

Filteroperationen - Ränder

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



Mirror



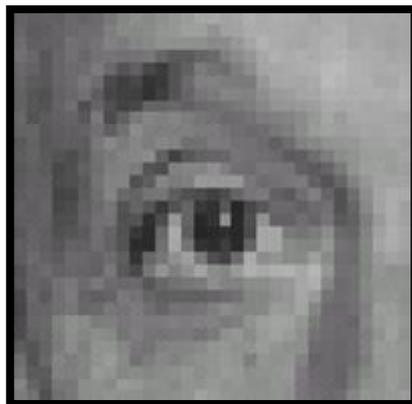
Clamp

Filteroperationen – Design (1)

Wie werden **Maskenkoeffizienten** definiert?

- Hängt davon ab, was der 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



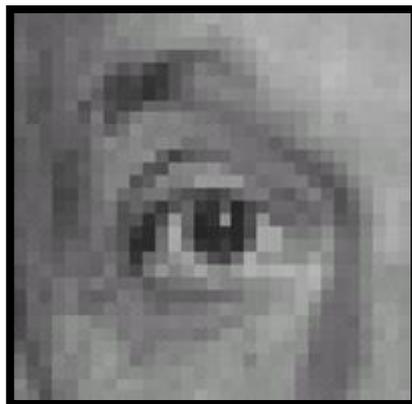
Ergebnis
(Löschen)

Filteroperationen – Design (2)

Wie werden **Maskenkoeffizienten** definiert?

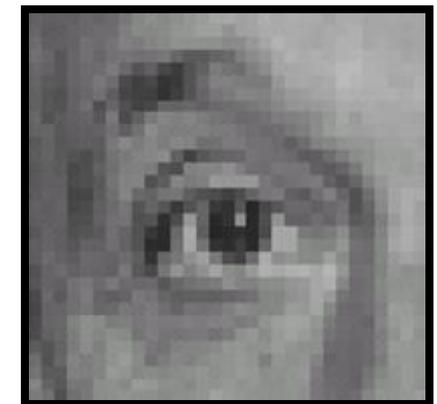
- Hängt davon ab, was der Filter tun soll!

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



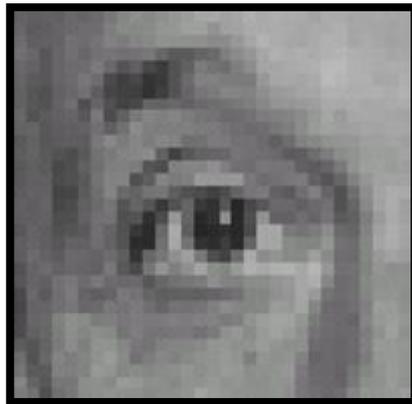
Ergebnis
(Identität)

Filteroperationen – Design (3)

Wie werden **Maskenkoeffizienten** definiert?

- Hängt davon ab, was der 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

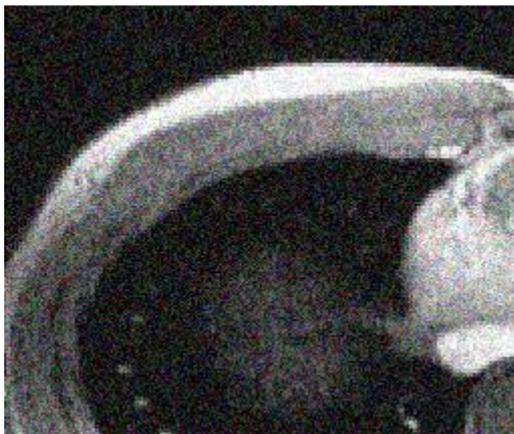
- **Tiefpassfilter:** Glättung, Rauschelimination
 - Medianfilter
 - Mittelwertfilter
 - Gauß-Filter

- **Hochpassfilter:** Kantendetektion
 - Prewitt
 - Sobel
 - Laplace

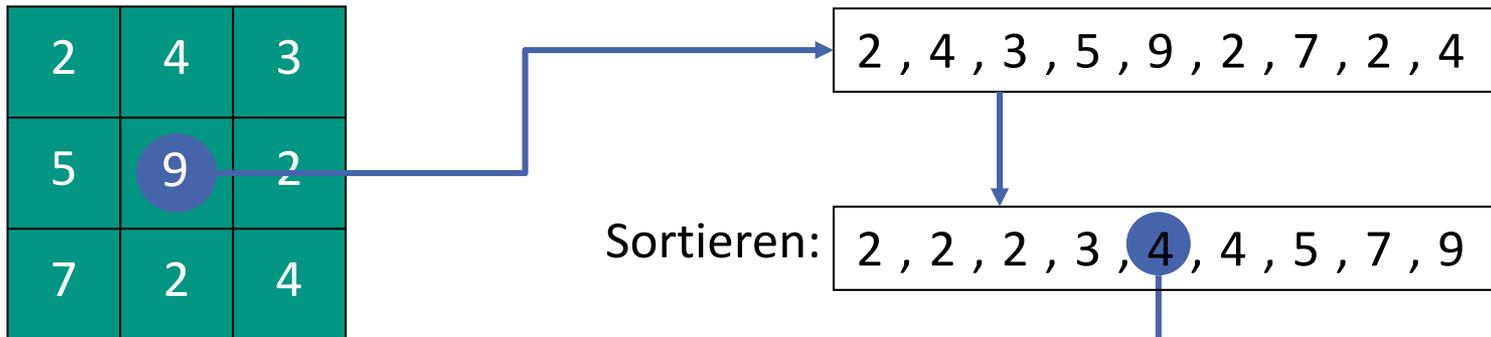
- **Kombinierte Operatoren**
 - Laplacian of Gaussian

Medianfilter

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

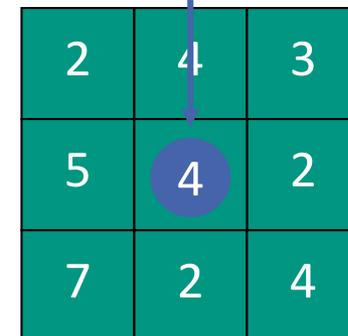


Medianfilter: Beispiel



Sortieren: 2 , 2 , 2 , 3 , 4 , 4 , 5 , 7 , 9

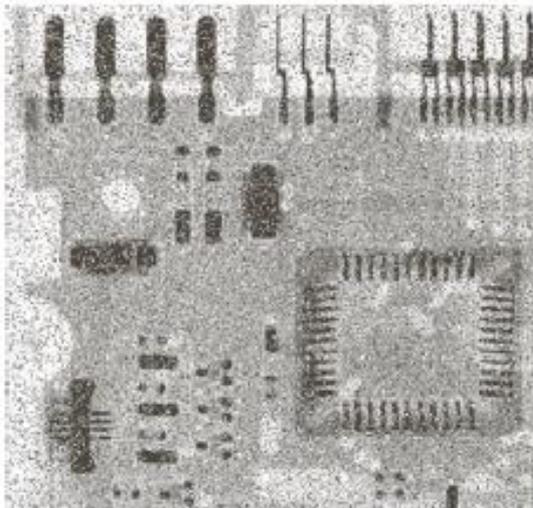
Neu:



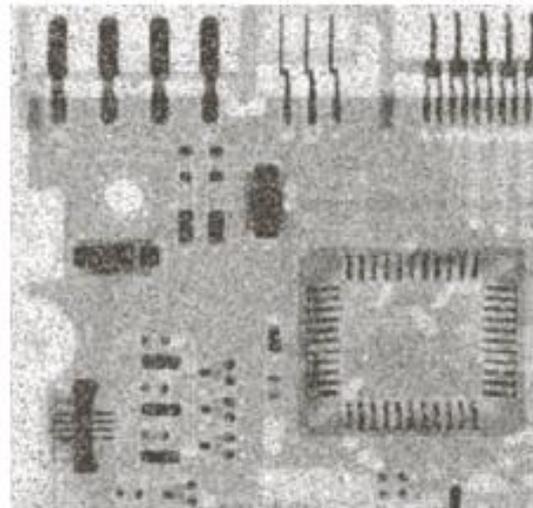
Anstelle des Medianwertes können wir den Maximalwert (**Max-Filter**) wählen, um die hellsten Punkte in einem Bild zu finden. Ebenso kann der **Min-Filter** für die dunkelsten Punkte verwendet werden. Beides sind nichtlineare Filter.

Medianfilter: Beispiel

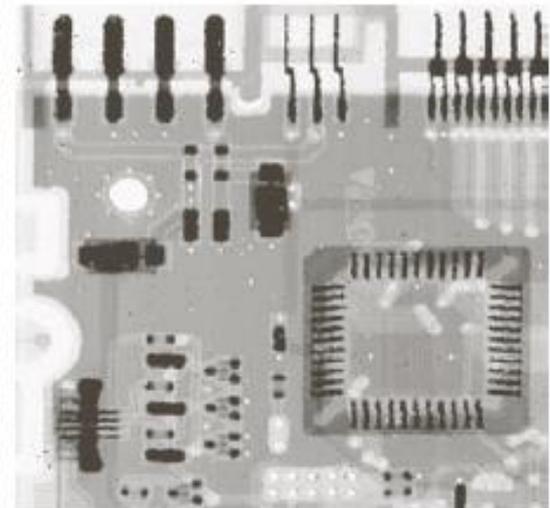
- Für bestimmte Arten von Rauschen, wie z.B. **Salz- und Pfefferrauschen**, bieten Medianfilter ausgezeichnete Rauschunterdrückungseigenschaften mit weniger Unschärfe als lineare Filter ähnlicher Größe!
 - Geeignet für Salz- und Pfefferrauschen
 - Nicht geeignet für Gaußsches Rauschen
- Bewahrt Kanten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



Original Bild beschädigt durch
Salz- und Pfefferrauschen



3 X 3 Mittelwertfilter



3 X 3 Medianfilter

Mittelwertfilter

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

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

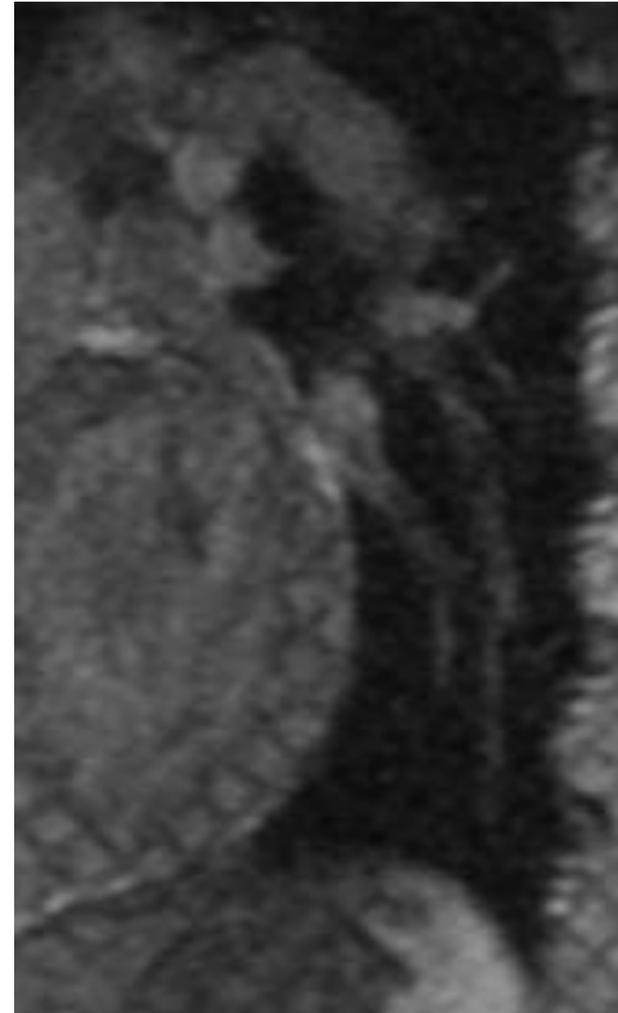


13	25	30	34	40	46	60	76
45	$(1/9)*50$	$(1/9)*52$	$(1/9)*55$	65	67	87	77
34	$(1/9)*45$	$(1/9)*55 \Rightarrow 52$	$(1/9)*60$	54	45	56	65
45	$(1/9)*50$	$(1/9)*52$	$(1/9)*48$	45	65	65	45
34	34	54	56	57	58	67	70
45	46	46	46	45	53	52	60
50	68	69	60	70	70	78	79
68	70	78	78	80	80	80	90

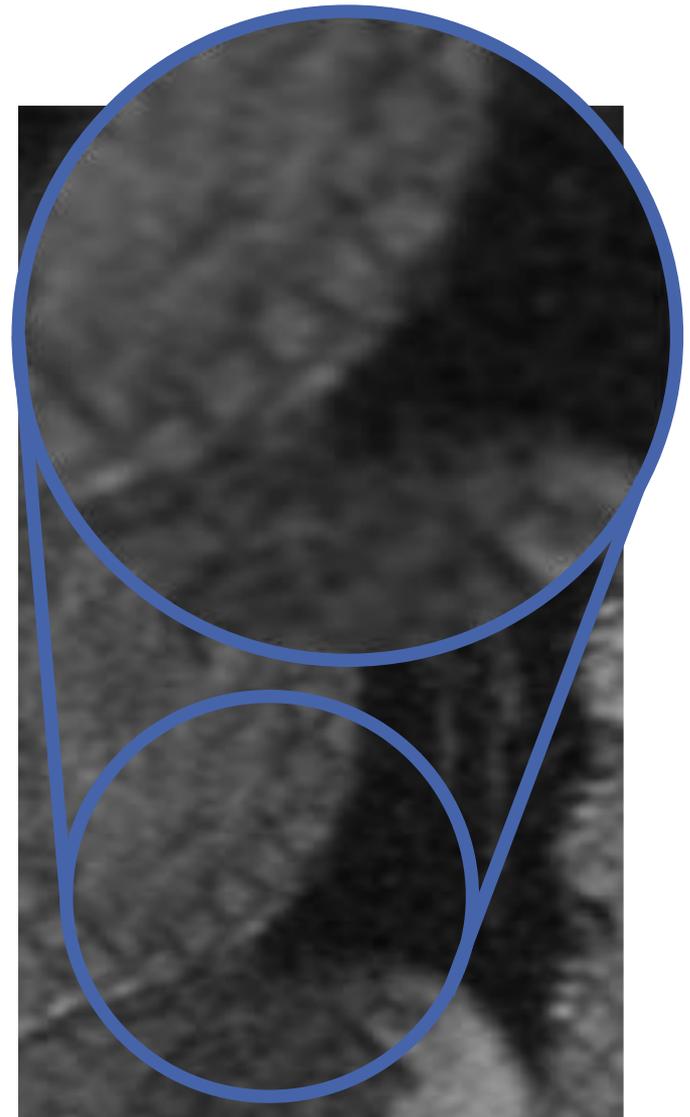
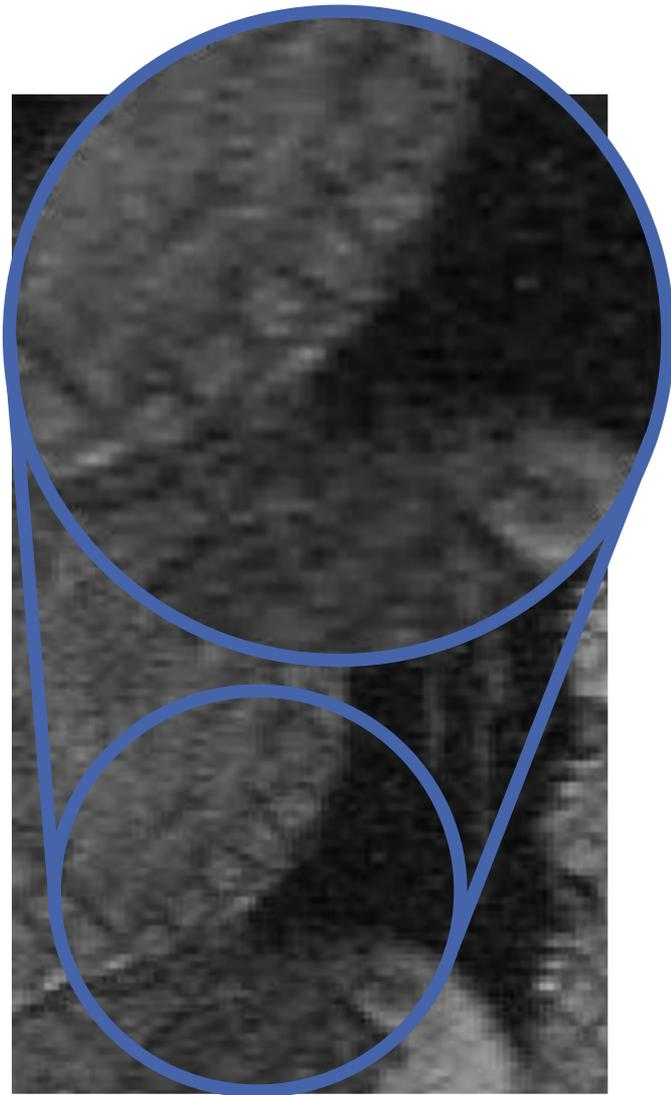
$$F_{\text{Mittelwert}} = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

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

Mittelwertfilter: Beispiel



Mittelwertfilter: Beispiel



Gauß-Filter

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

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

Ortsbereich

Frequenzbereich

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

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

Gauß-Filter II

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



Originalbild



$$\sigma^2 = 4$$



$$\sigma^2 = 16$$

Filteroperationen

- **Tiefpassfilter:** Glättung, Rauschelimination
 - Medianfilter
 - Mittelwertfilter
 - Gauß-Filter

- **Hochpassfilter:** Kantendetektion
 - Prewitt
 - Sobel
 - Laplace

- **Kombinierte Operatoren**
 - Laplacian of Gaussian

Filter – Prewitt

■ Prewitt-X Filter

- Approximiert durch

$$P_x = \frac{\partial f(x,y)}{\partial x}$$

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

■ Prewitt-Y Filter

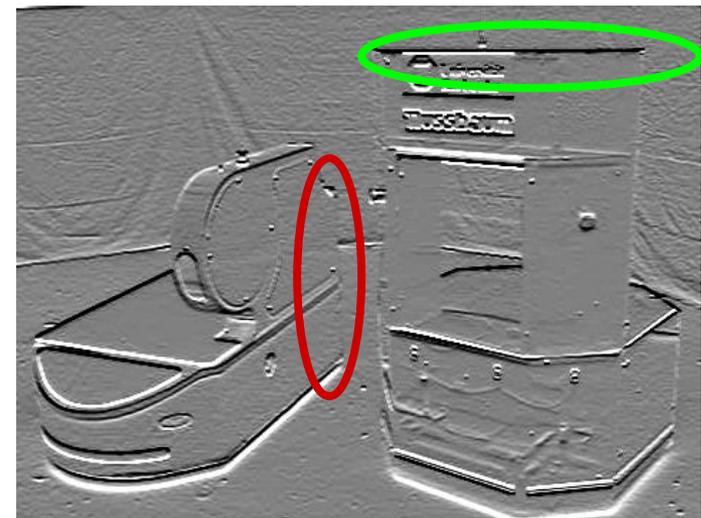
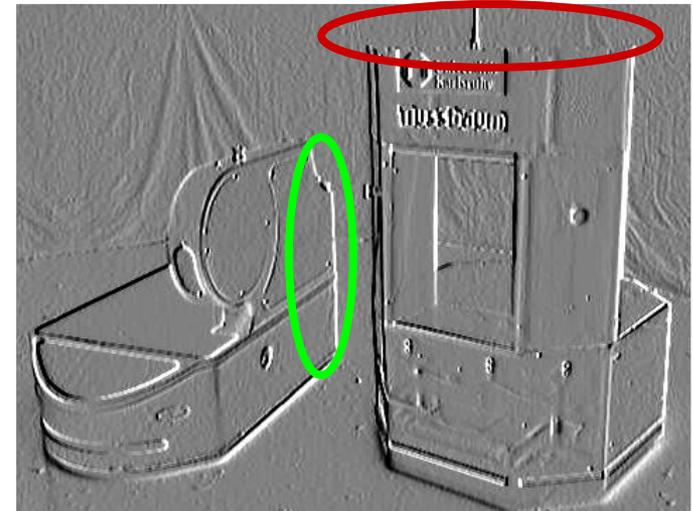
- Approximiert durch

$$P_y = \frac{\partial f(x,y)}{\partial y}$$

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

■ Eigenschaften:

- Gute Ergebnisse bei Detektion von vertikalen bzw. horizontalen Kanten

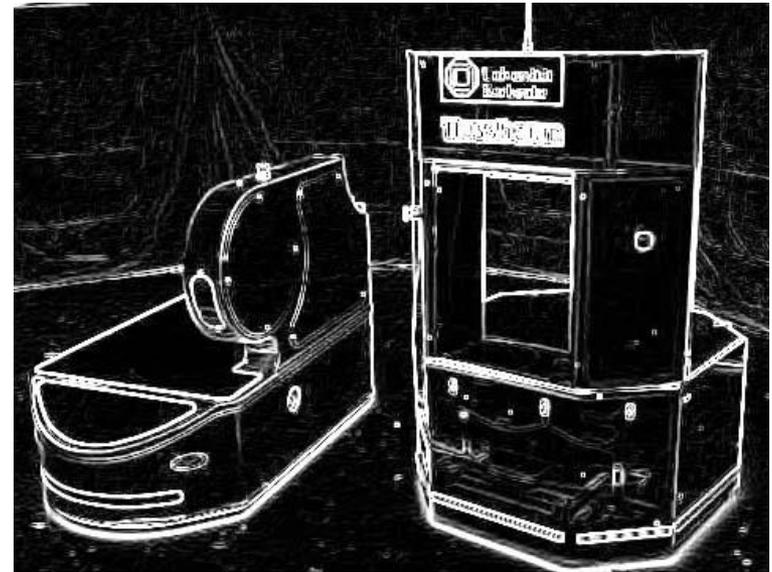


Filter – Prewitt (2)

- 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
- Sobelfilter ist die Verkettung von Gaußfilter über Ableitung und Differenzenquotienten

■ Sobel-X Filter

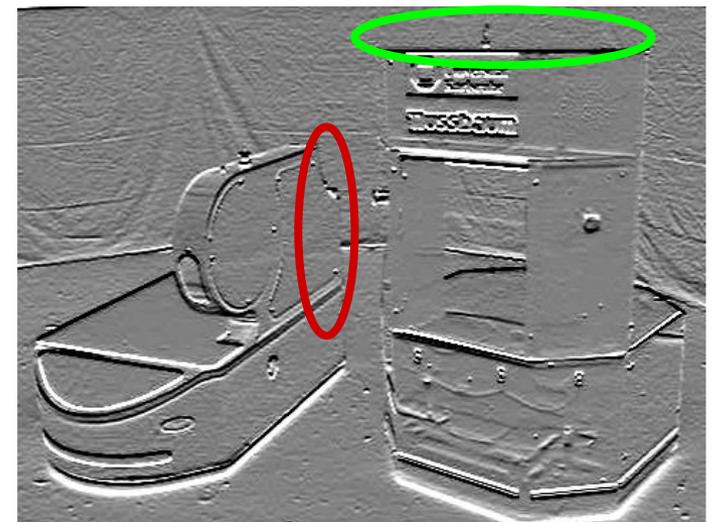
Approximiert durch

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

■ Sobel-Y Filter

Approximiert durch

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

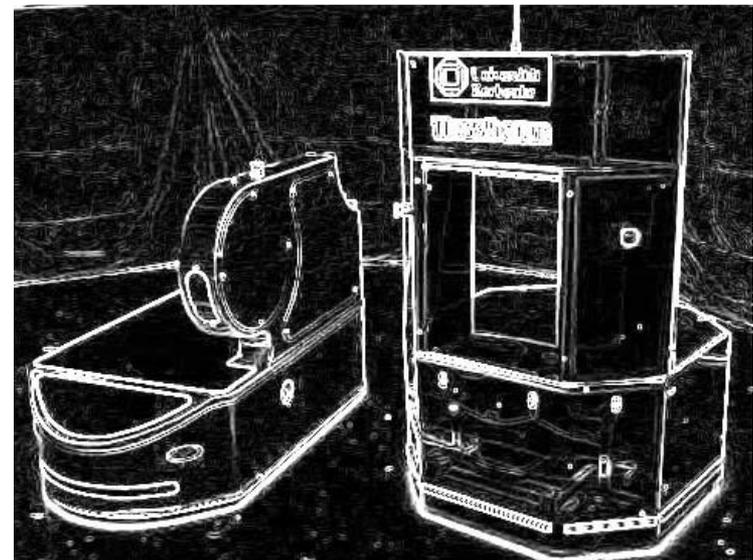


Filter – Sobel (2)

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

$$M \approx \sqrt{S_x^2 + S_y^2}$$

- Danach: Schwellwertfilterung

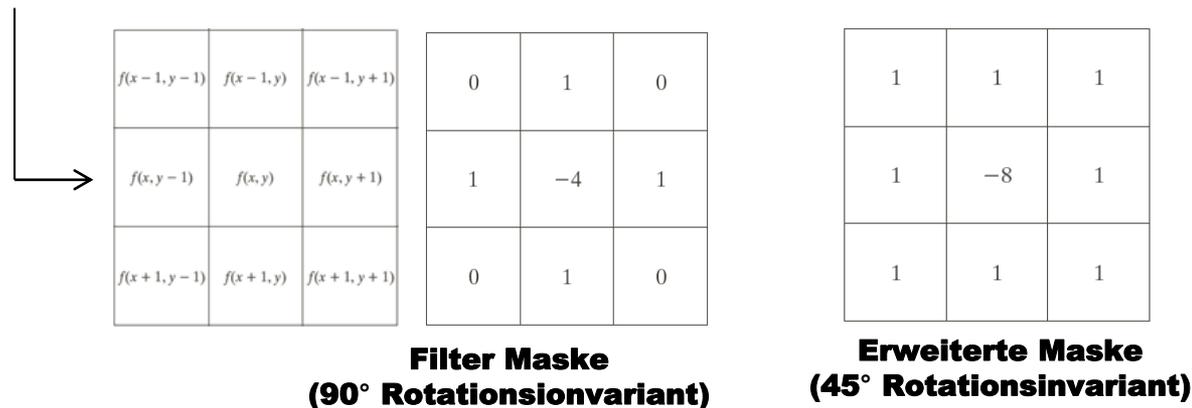


Filter – Laplace (1)

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

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \left\{ \begin{array}{l} \frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \\ \frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \end{array} \right.$$

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



die Koeffizienten der Maskensumme sind gleich Null, was darauf hindeutet, dass die Antwort in den Bereichen konstanter Intensität gleich Null ist.

Filter – Laplace (2)

Laplace-Operator:

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

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



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

Filteroperationen

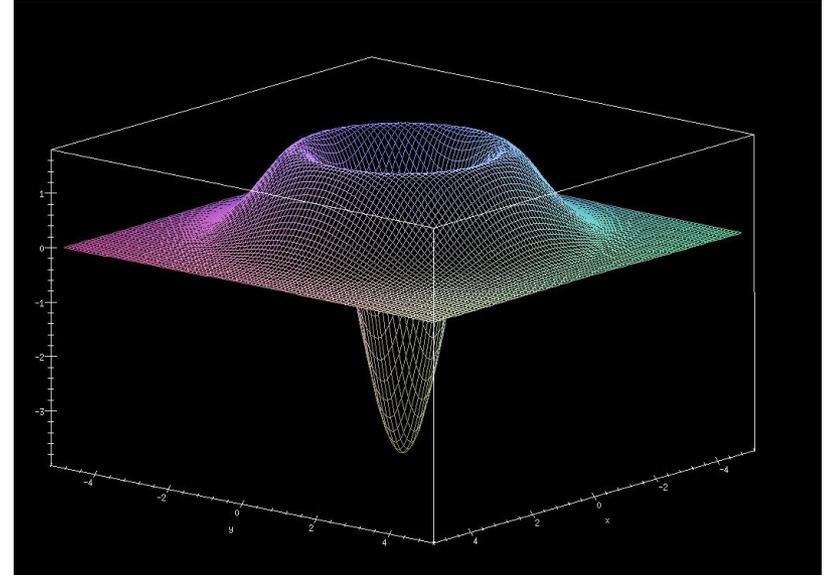
- **Tiefpassfilter:** Glättung, Rauschelimination
 - Medianfilter
 - Mittelwertfilter
 - Gauß-Filter

- **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:



$$LoG(f(x, y)) = \Delta (f(x, y) * g(x, y))$$

g bezeichnet die Filterfunktion eines Gauß-Filters.

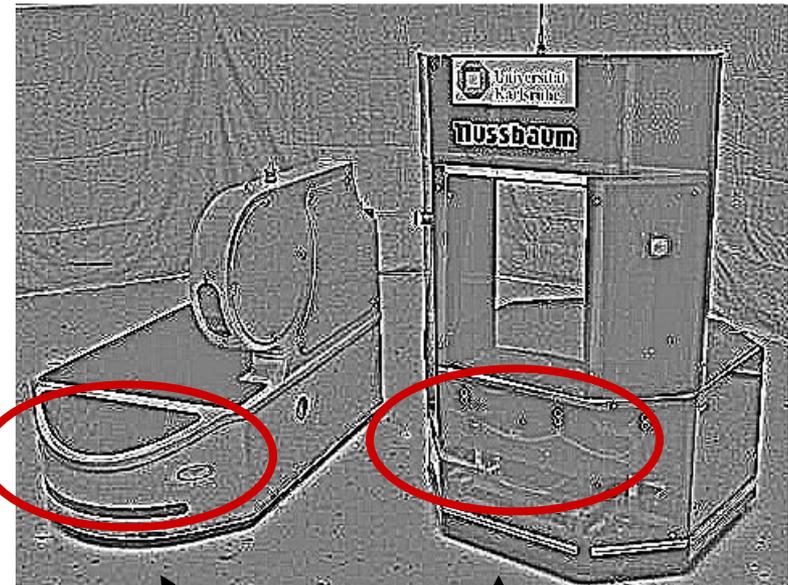
Hauptmerkmale des LoG-Operators

- Gaußsche Teil des Operators ist ein **Tiefpassfilter**, der das Bild verwischt (glättet) 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 bei der Gaußschen Filter unwahrscheinlich, dass Artefakte wie z.B. "Treppenhaus"-Effekte auftreten, die im Originalbild nicht vorhanden sind.
- Der Laplace-Operator (die zweite Ableitung) ist invariant zur 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.

Approximation

Faltung mit der Matrix

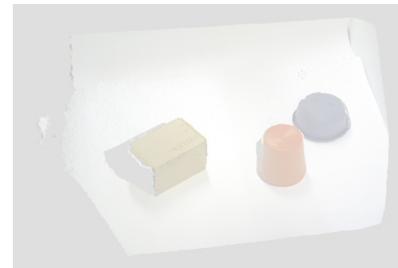
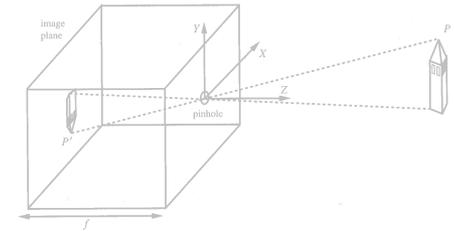
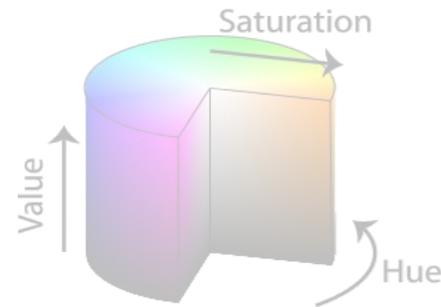
$$\Delta F(x, y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$



Stärkere Kanten,
Weniger Rauschen

Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- **Segmentierung**
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Segmentierung

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

Mögliche Verfahren:

- Schwellwertfilterung
- Clustering
- Kantenextraktion
- Region-Growing



Segmentierung: Schwellwertfilterung (1)

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

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



Segmentierung: Schwellwertfilterung (2)

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

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

$$\text{Img}'(u, v) = \begin{cases} 255, & \text{falls } \begin{cases} H_{max} \geq \text{Img}_H(u, v) \geq H_{min}, \\ S_{max} \geq \text{Img}_S(u, v) \geq S_{min}, \\ V_{max} \geq \text{Img}_V(u, v) \geq V_{min} \end{cases} \\ 0, & \text{sonst} \end{cases}$$

- Problem:
 - Wechselnde Lichtbedingungen
 - Reflexionen, Schattenwürfe

Einschub: Probleme von Lichtverhältnissen

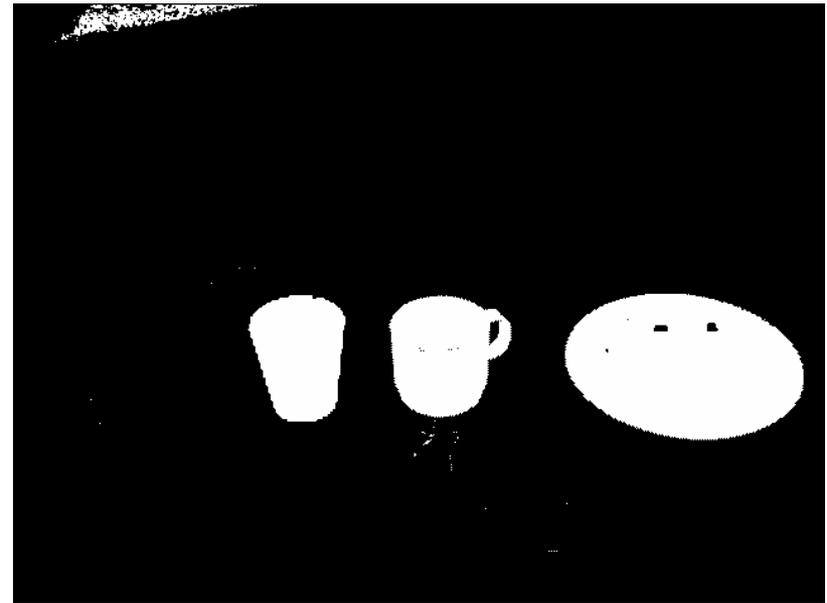


Segmentierung: Beispielanwendung

- Beispielanwendung: Objekterkennung und -lokalisierung



Eingabebild



Ergebnis der Segmentierung

Anwendungsbeispiel : Farbsegmentierung

- Segmentierung der Hautfarbe und Verfolgung von Gesicht und Händen

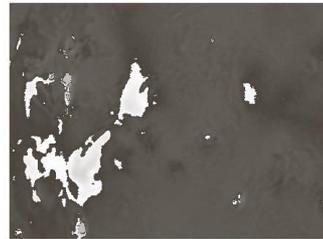


HSV-Segmentierung Wie kann die rote Region extrahiert werden? (HSV)

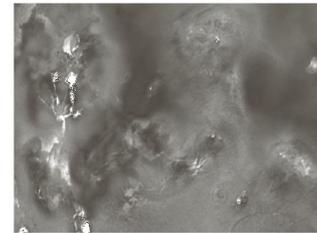
Original Bild



Farbnuance



Sättigung



Helligkeit



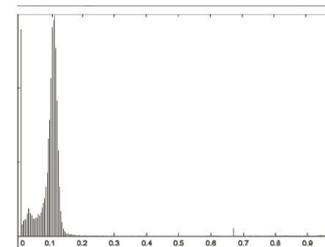
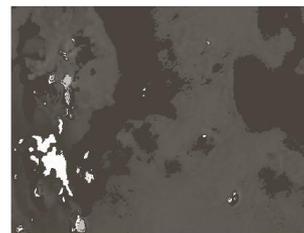
HSV

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



Binärmaske durch Schwellwertfilterung des **Sättigungsbildes** (>10%)

Farbnuancenwerte des Bildes werden mit der Maske multiplizieren.



Schwellwertfilterung
Anwenden auf das Histogramm!

Berechnung des **Histogramms**

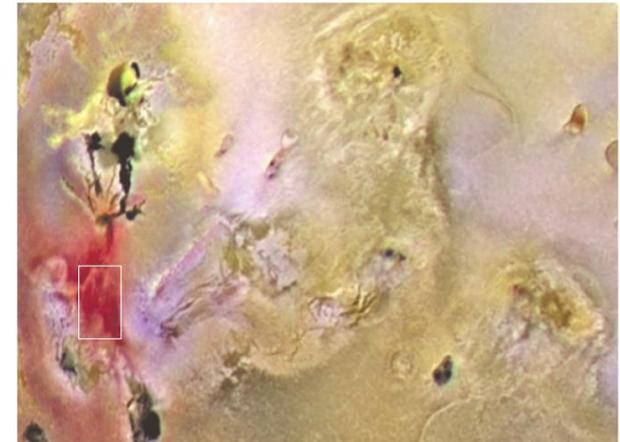


Segmentiertes Bild

RGB Segmentierung

Original Bild

Wie kann die rote Region extrahiert werden? (RGB)

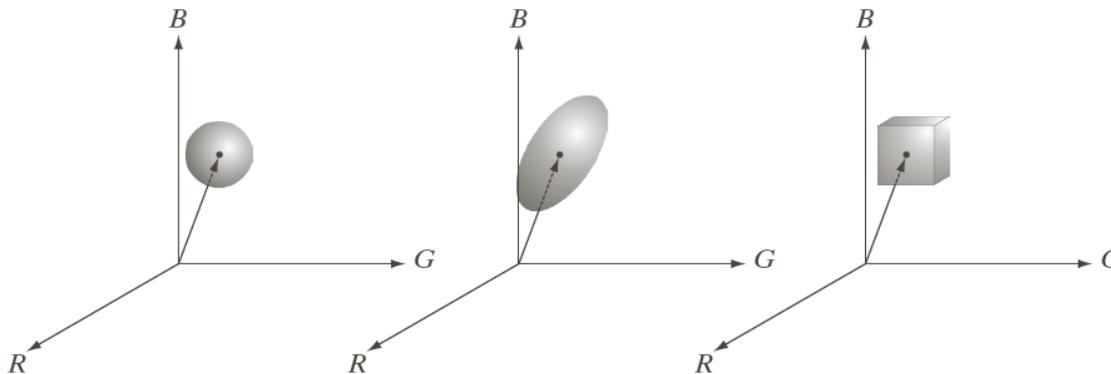


Definiere region of interest (ROI).

Euklidische Distanz
der beiden Farbvektoren!

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

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

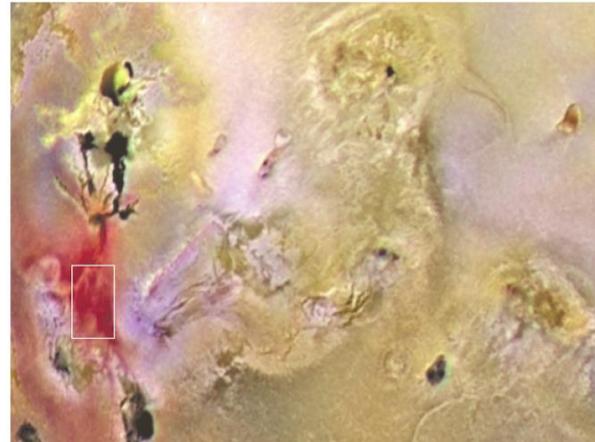


Unterschiedliche Bereiche!

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

RGB Segmentierung

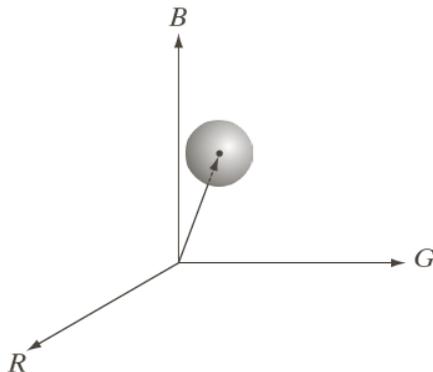
Wie kann die rote Region extrahiert werden? (*RGB*)



Original Bild



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



Segmentiertes Bild

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

Robotik I – Bildverarbeitung in der Robotik

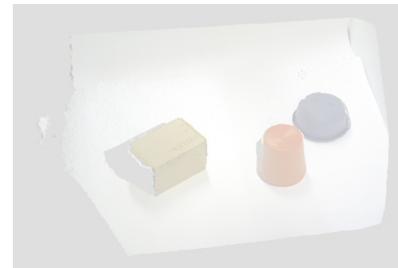
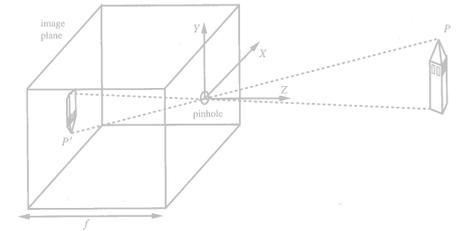
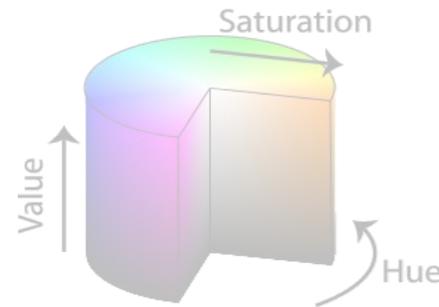
Peter Kaiser, Tamim Asfour

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)



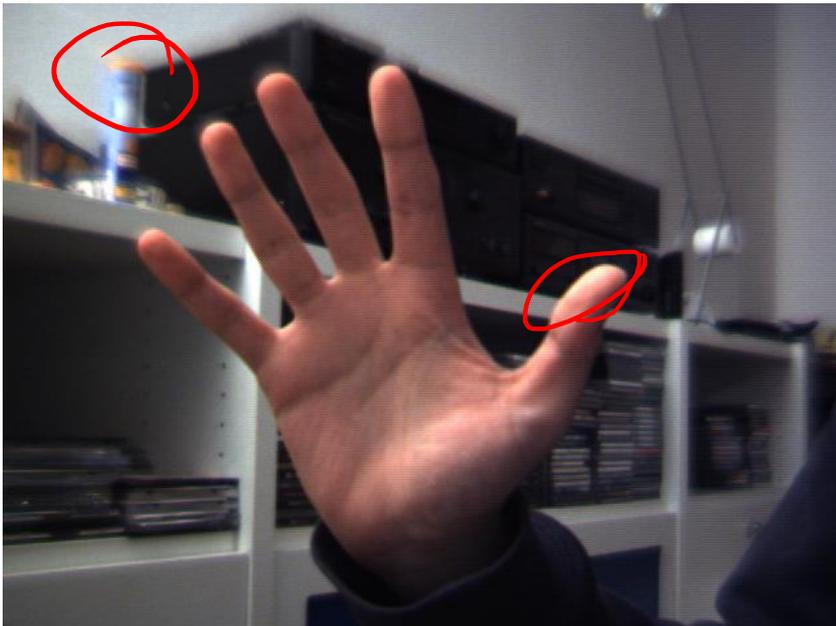
Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- **Morphologische Operatoren**
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel

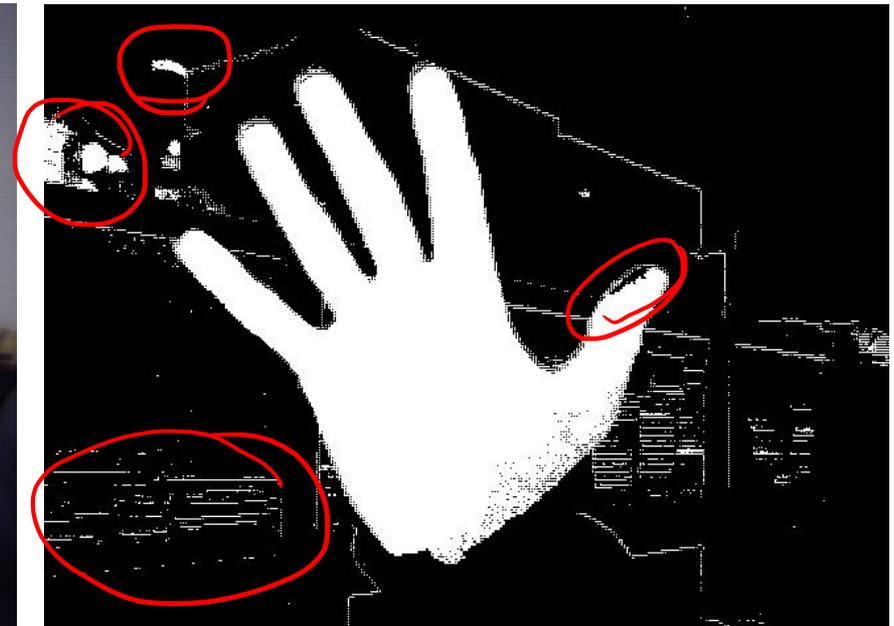


Motivation: Farbsegmentierung

- Problem bei der Segmentierung: Artefakte, unvollständige Segmentierung



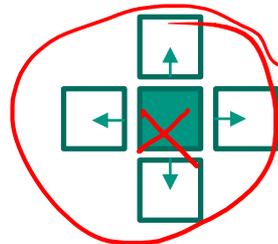
Eingabebild



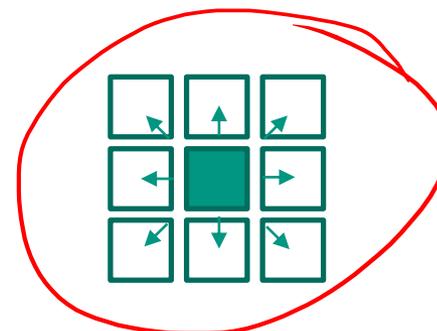
Ergebnis Segmentierung

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
 - **Erosion**
Die Erosion entfernt vereinzelte Pixel und schwach zusammenhängende Pixelgruppen
- Der Effekt eines morphologischen Operators hängt von der Größe und der Form der berücksichtigten Pixelnachbarschaft ab. Definiert durch das strukturierende Element/Kernel



4er-Nachbarschaft



8er-Nachbarschaft

Morphologische Operatoren: Erosion

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

$k := \text{div}((n - 1), 2)$

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 $I(u + j, v + i) \neq q$ then

goto NEXT

end if

end for

end for

$I'(u, v) = q$

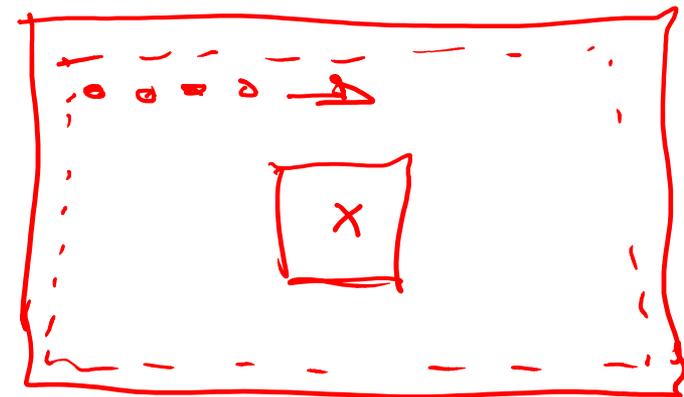
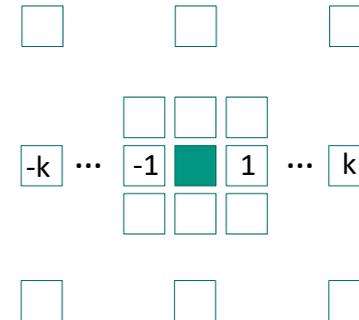
NEXT:

end if

end for

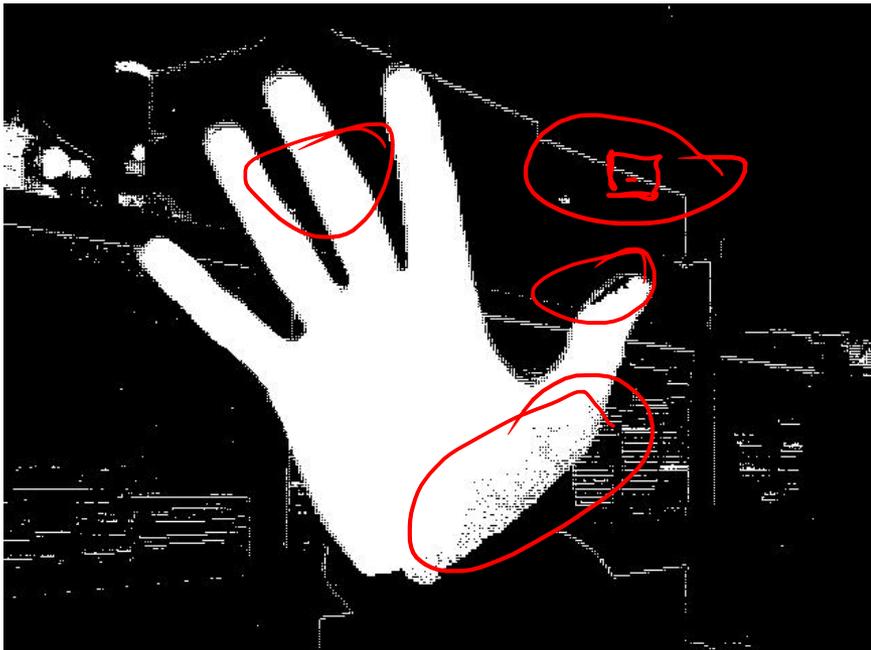
end for

Bildhöhe h ,
Bildbreite w

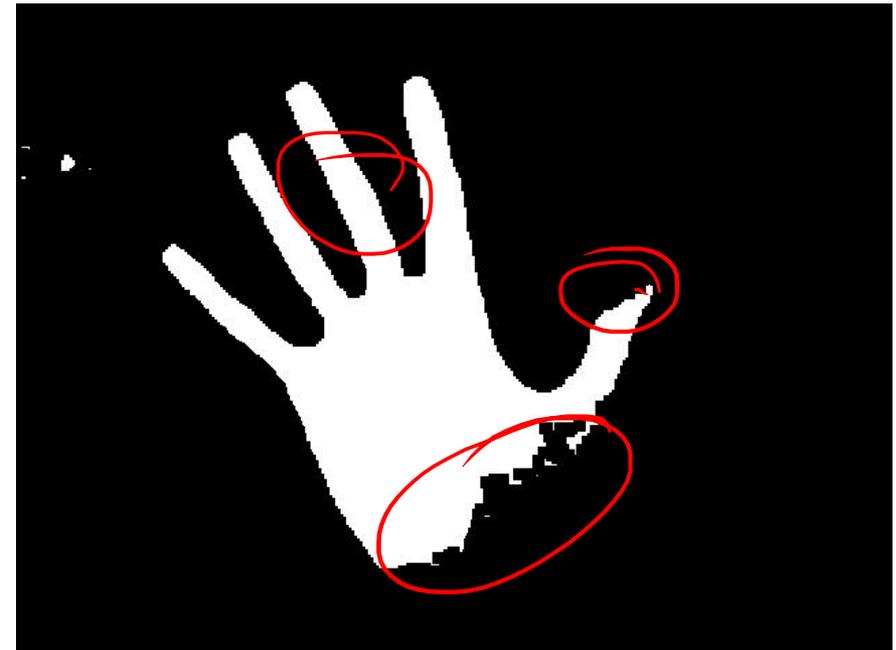


Morphologische Operatoren: Erosion, Beispiel

- Beispielanwendung einer Erosion



Eingabebild



Ergebnisbild

Morphologische Operatoren: Dilation

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

$k := \text{div}((n - 1), 2)$

for all pixels (u, v) in I' do

$I'(u, v) := 0$

end for

for $v := k$ to $h - k - 1$ do

for $u := k$ to $w - k - 1$ do

if $I(u, v) = q$ then

for $i := -k$ to k do

for $j := -k$ to k do

$I'(u + j, v + i) = q$

end for

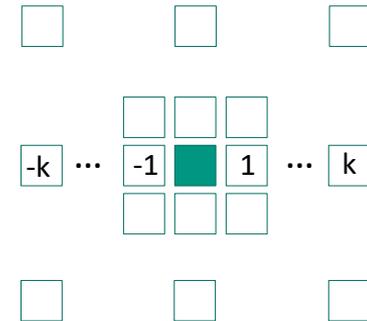
end for

end if

end for

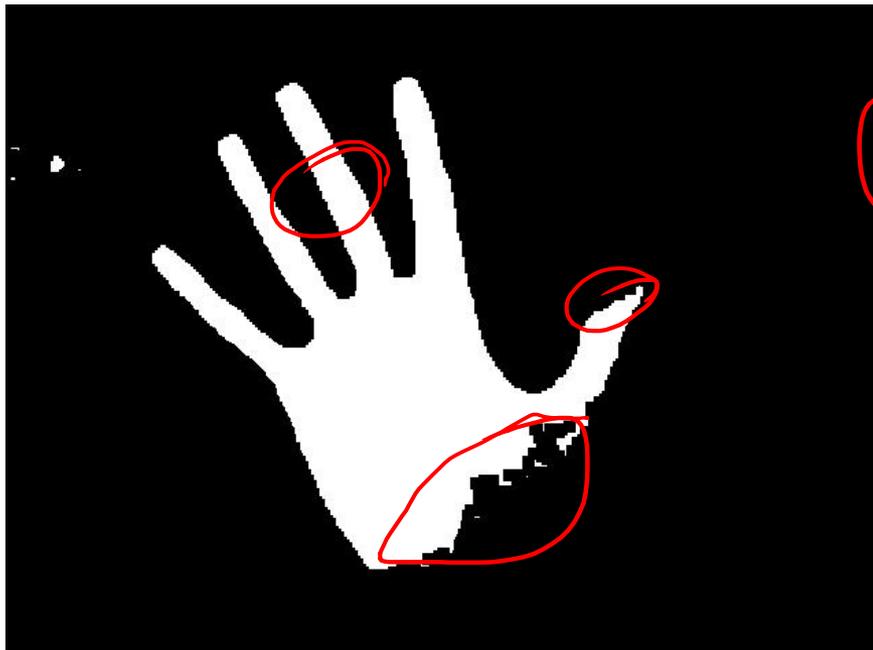
end for

Bildhöhe h ,
Bildbreite w

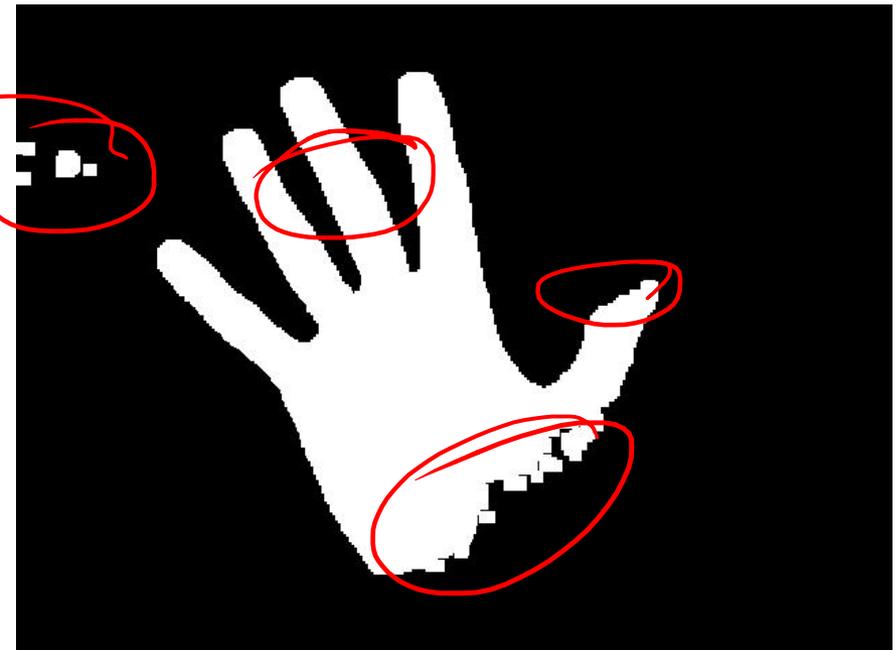


Morphologische Operatoren: Dilatation, Beispiel

- Beispielanwendung einer Dilatation



Eingabebild



Ergebnisbild

Morphologische Operatoren: Öffnen & Schließen

Öffnen:

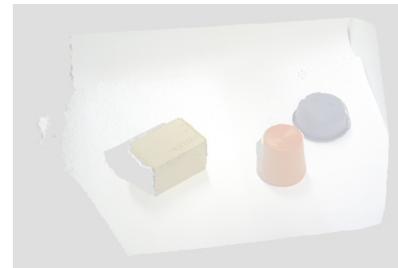
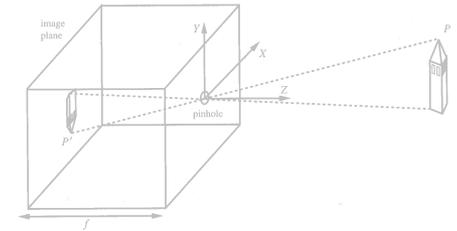
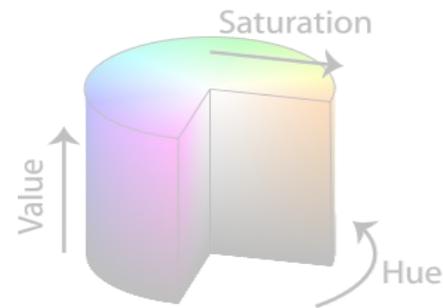
- Anwendung von Erosion danach Dilatation
- Entfernt dünne Stege oder kleine außenliegende Objekte

Schließen:

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

Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- **Canny-Kantendetektor**
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Canny-Kantendetektor

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

Canny-Kantendetektor

Das Prinzip basiert auf drei Grundsätzen:

1. **Geringe Fehlerrate:** Alle Kanten sollen gefunden werden und detektierte Kanten sollten so nah wie möglich an den realen Kanten sein.
2. **Kantenpunkte sollen gut detektiert werden:** Distanz zwischen detektierten Kantenpunkten und dem Zentrum der realen Kanten soll minimal sein.
3. **Eindeutigkeit:** Der Detektor soll nur ein Punkt, nicht mehrere Kantenpunkte, zu einem realen Kantenpunkt liefern.

Canny-Kantendetektor: Algorithmus

1. Rauschunterdrückung: Gauß-Filter
2. Berechnung der Gradienten in horizontaler und in vertikaler Richtung
 $Prewitt_x / Prewitt_y$ oder $Sobel_x / Sobel_y$

a) Berechnung der Richtung: $\phi = \text{atan}\left(\frac{g_y}{g_x}\right)$

b) Einteilung der Richtung in vier (oder acht) Quadranten:

1 : $[-67.5^\circ, -22.5^\circ)$

2 : $[-22.5^\circ, 22.5^\circ)$,

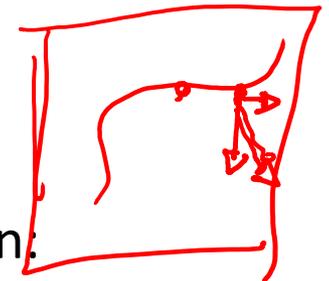
3 : $[22.5^\circ, 67.5^\circ)$

4 : $[-90^\circ, -67.5^\circ)$ oder $[67.5^\circ, 90^\circ)$

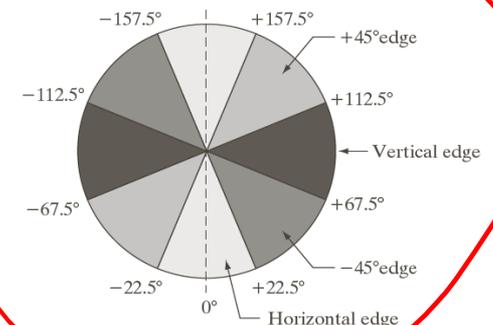
c) Magnitude $M = \sqrt{g_x^2 + g_y^2}$

3. Non-Maximum Supression

4. Hysterese-Schwelwertverfahren

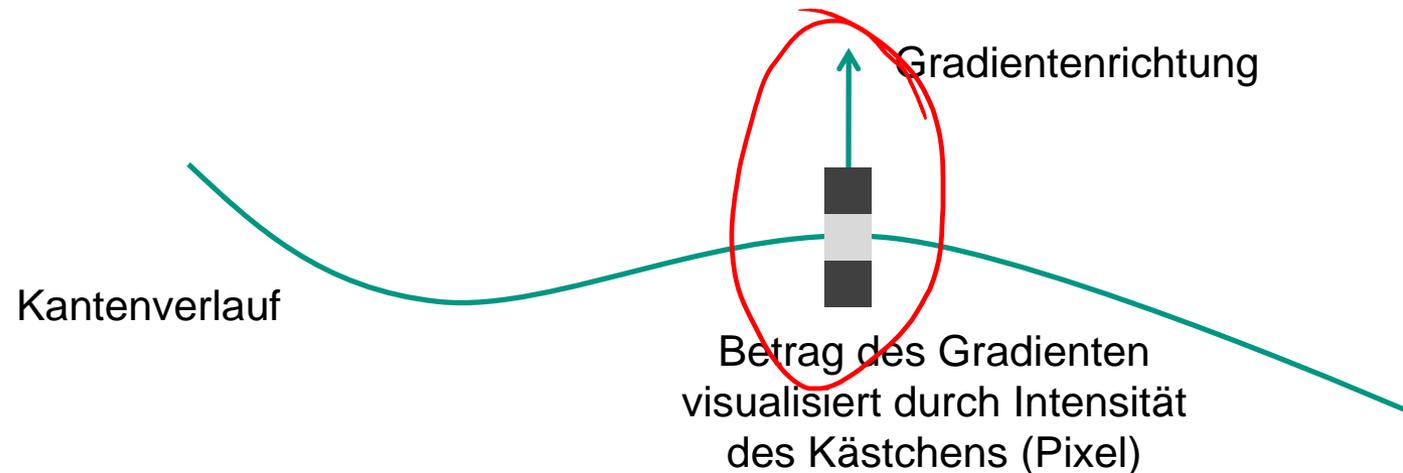


Einteilung in 8 Quadrante:



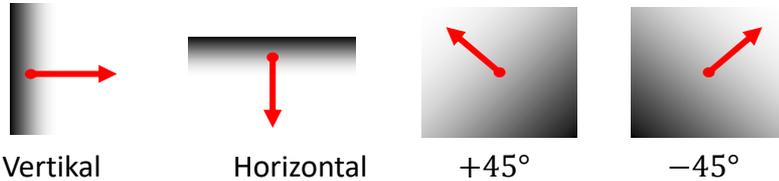
Canny-Kantendetektor: Non-Maximum Suppression

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



Canny-Kantendetektor : Non-Maximum Suppression

Definiere vier Kantenorientierung

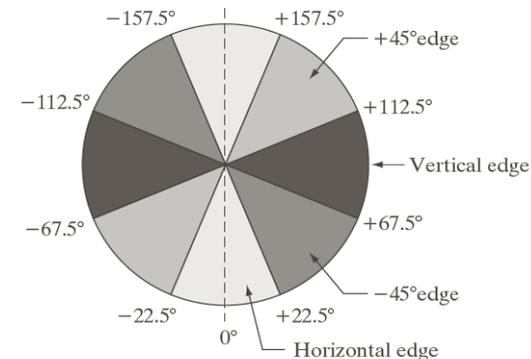
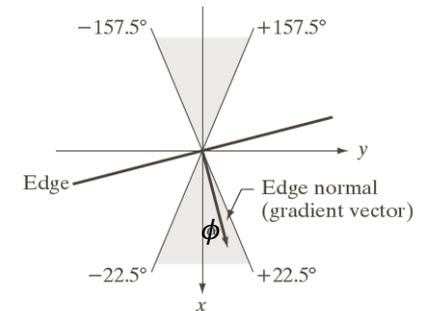
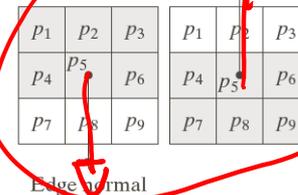


Definiere einen Bereich um alle möglichen Kantenrichtungen in vier Richtungen zu unterteilen

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

1. Definiere eine 3×3 Region (p_1 bis p_9) um jeden Punkt (x, y) in $\phi(x, y)$.
2. Bestimme die Richtung der Kante welche nah bei $\phi(x, y)$ ist. (In diesem Beispiel ist p_5 horizontal.)
3. Wenn der Wert im Zentrum der Magnitude M kleiner als einer der beiden Nachbarn in Richtung der Kantennormale ist (im Beispiel p_2 und p_8), dann setze $M = 0$ (Suppression!)

Beispiel: $p_1 \dots p_9$



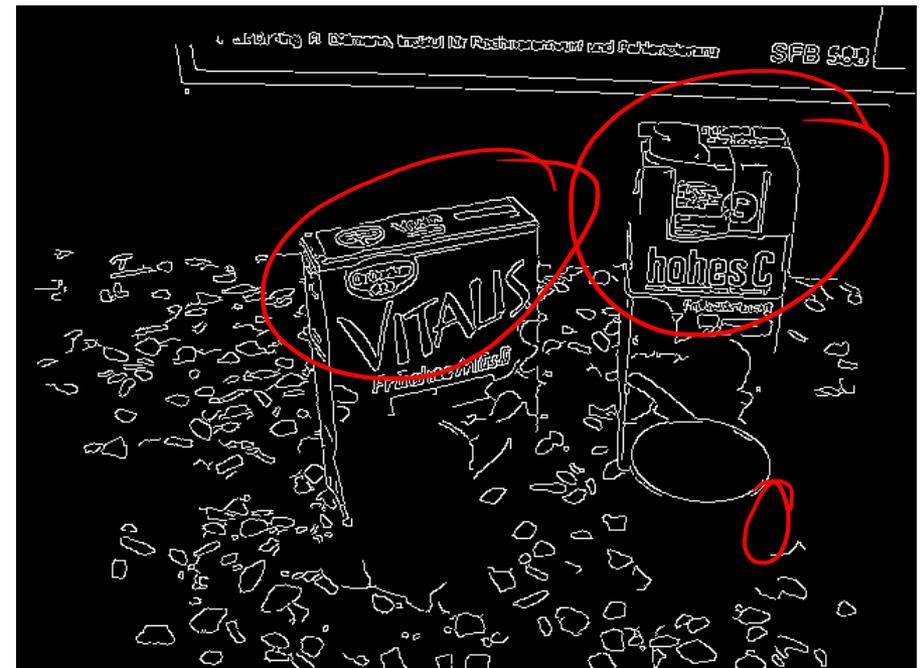
Canny-Kantendetektor: Schwellwertverfahren

■ Hysterese-Schwellwertverfahren

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

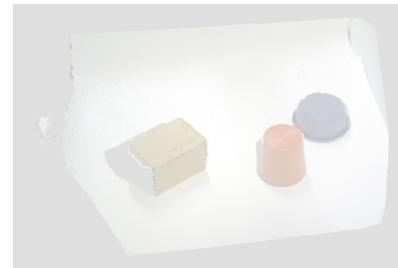
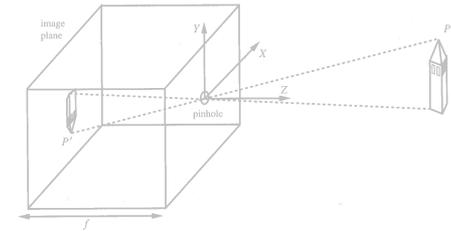
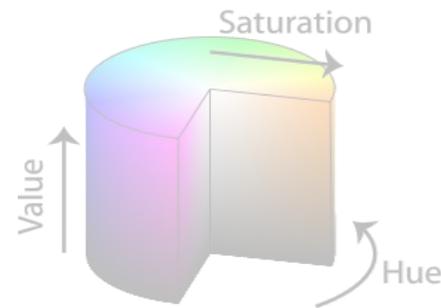
Anmerkung: Lokale Maximalitätsbedingung aus Schritt 3 muss in jedem Fall erfüllt sein!

Canny-Kantendetektor: Beispiel



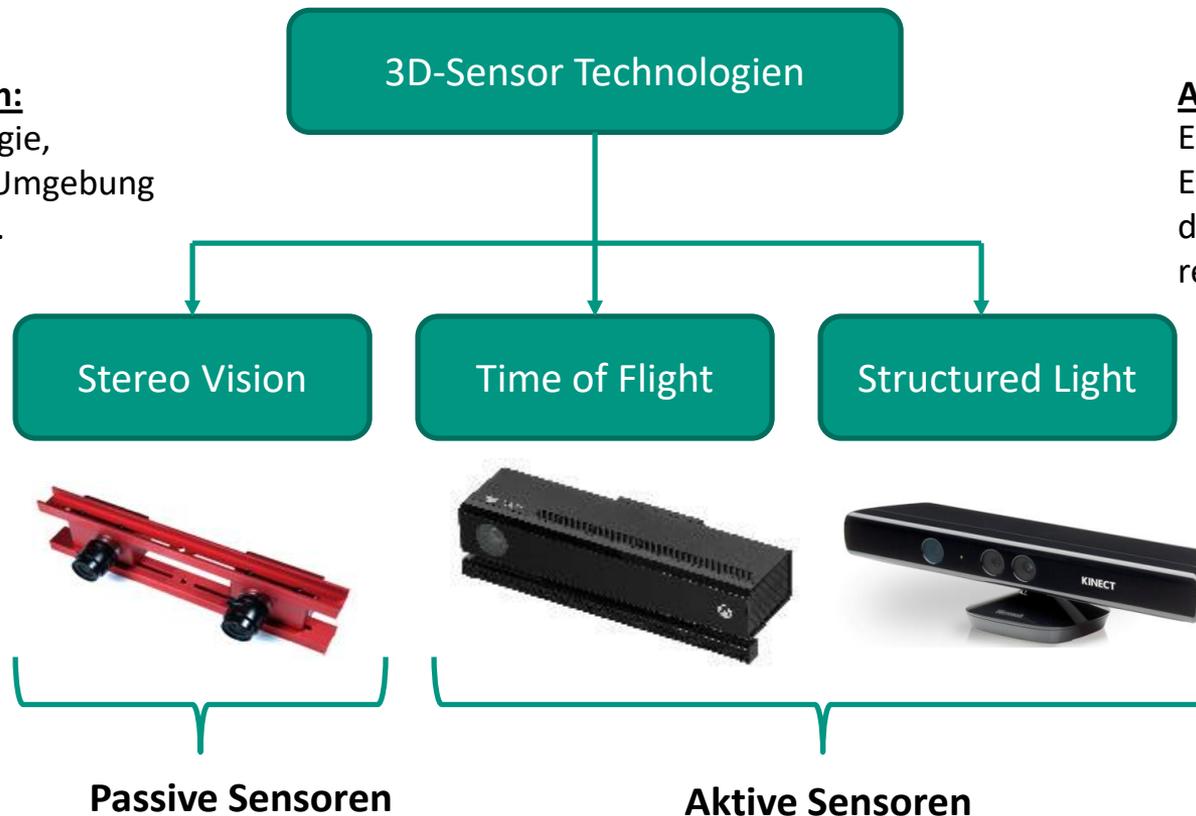
Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- **Tiefenkameras**
- Visual Servoing
- Punktwolken
- SLAM
- Anwendungsbeispiel



Passive Sensoren:
Empfangen Energie,
welche von der Umgebung
abgestrahlt wird.

Aktive Sensoren:
Erzeugen und senden
Energie aus, die von
der Umgebung
reflektiert wird.



Stereo Vision I

- Brennweite f
- Baseline $B = b$

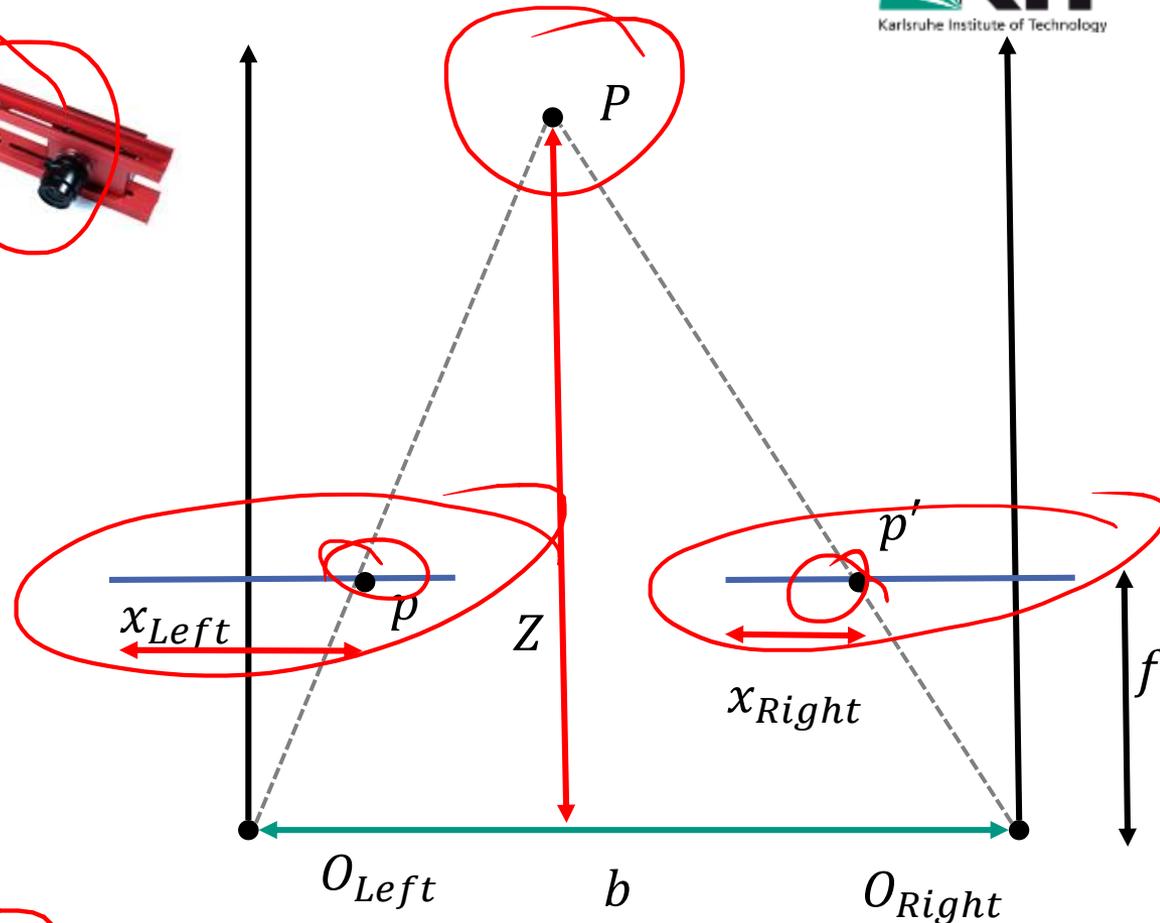
- Objektdistanz Z
- Objektpunkt P
- Bildpunkte

- $p = (x_{Left}, y_{Left})^T$,

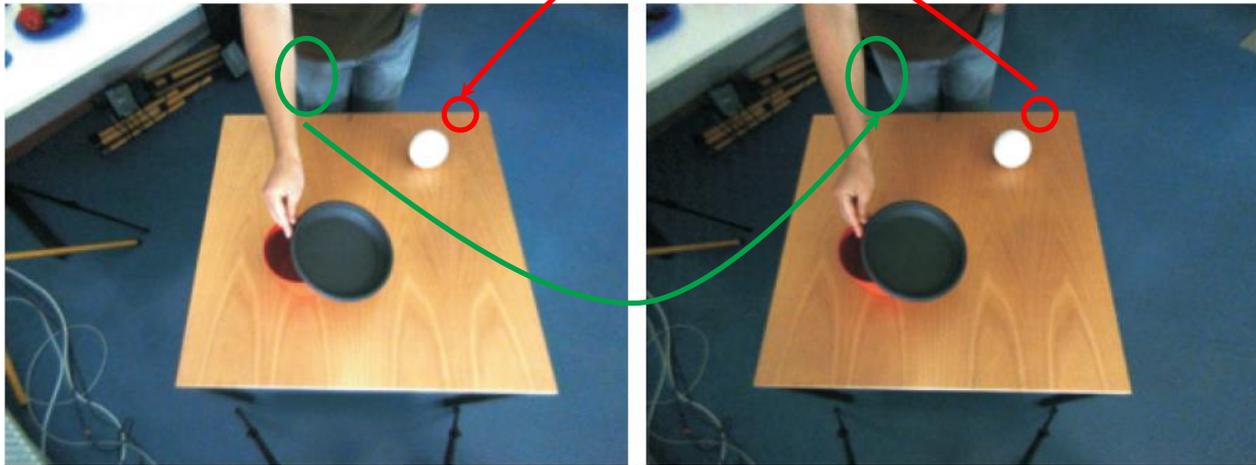
- $p' = (x_{Right}, y_{Right})^T$

- Disparität d

$$Z = \frac{b \cdot f}{x_{Left} - x_{Right}} = \frac{b \cdot f}{d}$$



Stereo Vision III



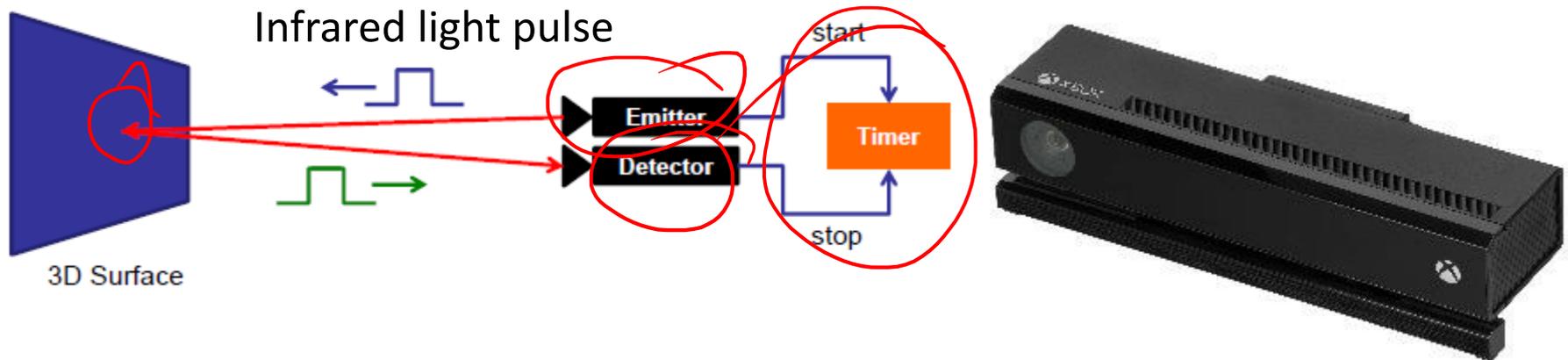
+ Vorteile

- Einstellbare Brennweiten/Baseline
- Keine explizite Lichtquelle erforderlich

- Nachteile

- Mindestens zwei Kameras
- Korrespondenzproblem bei homogenen Flächen

Time of Flight



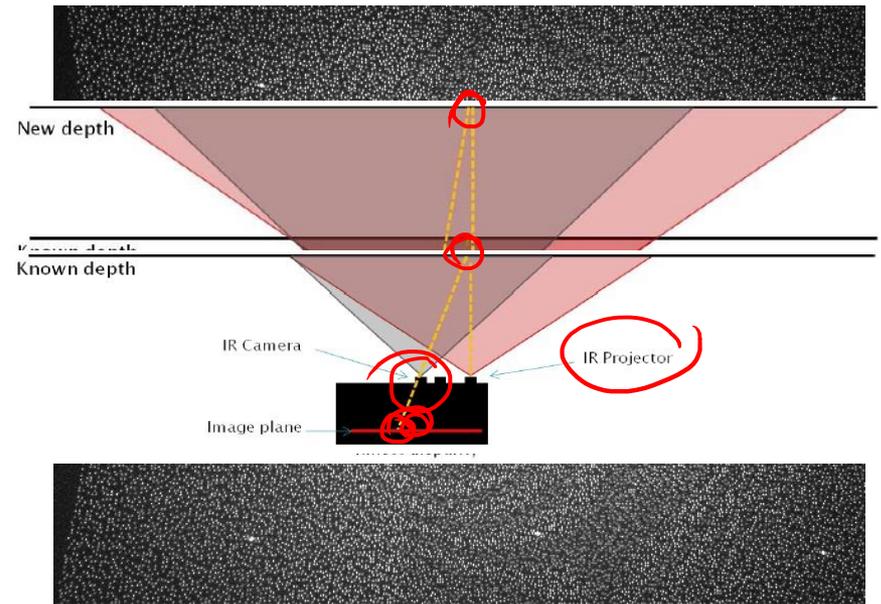
Vorteile

-  Keine explizite Tiefenberechnung notwendig
-  Hohe Bildrate (bis zu 100 fps)
-  Robust gegenüber den meisten Oberflächen (Kein Korrespondenzproblem)

Nachteile

-  Niedrige Auflösung (512 x 424 px)
-  Interferenz mit anderen Lichtquellen und Reflektionen

Structured Light



Vorteile

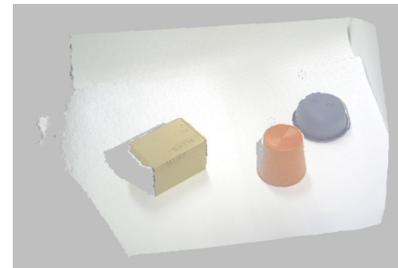
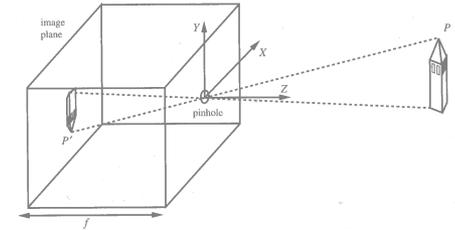
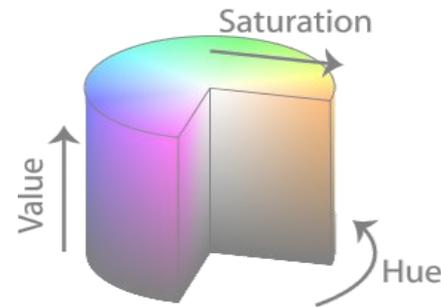
-  Preiswert
-  Kein Korrespondenzproblem auf homogenen Flächen

Nachteile

-  Eingeschränkte Reichweite (2 m)
-  Erfordert bestimmte Lichtverhältnisse (nur für den Innenraum geeignet)

Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- **Visual Servoing**
- Punktwolken
- SLAM
- Anwendungsbeispiel

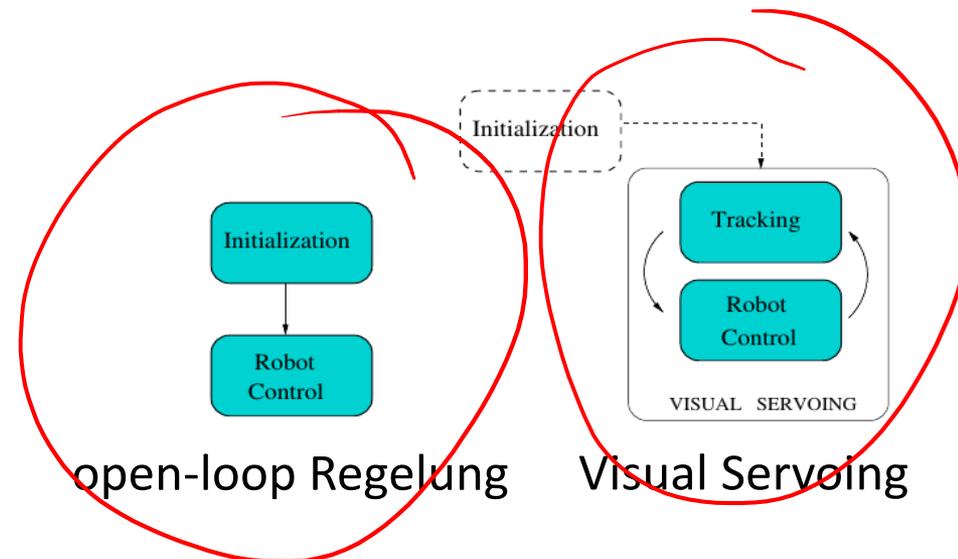


Visual Servoing - Motivation

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

Motivation

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



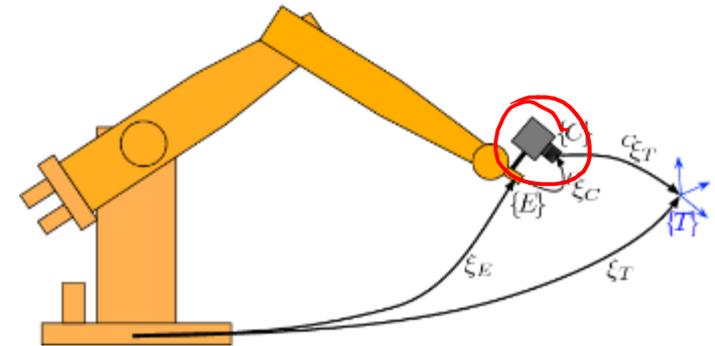
F Chaumette, S Hutchinson, *Visual servo control - Part I - Basic approaches*, IEEE Robotics & Automation Magazine 13 (4), 82-90, 2006

Danica Kragic and Henrik I Christensen, *Survey on Visual Servoing for Manipulation*, Techn. Report, KTH, 2002

Visual Servoing - Systemaufbau

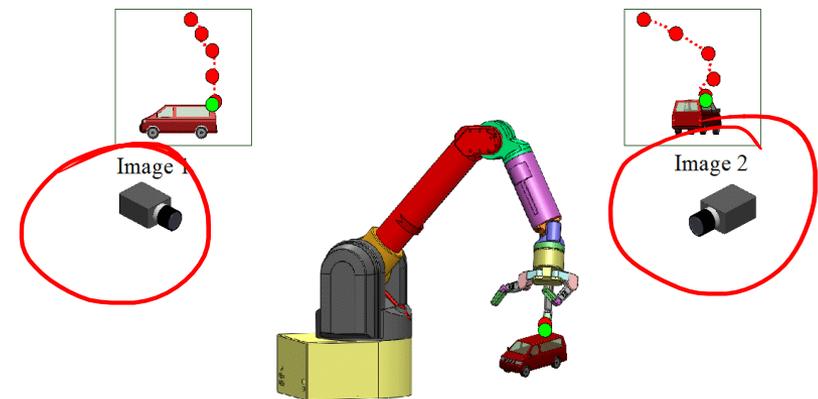
■ Kamera-in-Hand (eye-in-hand)

- Kamera ist an dem Manipulator angebracht
- Bewegungen des Manipulators beeinflussen die Pose der Kamera



■ Externes (festes) Kamera System (eye-to-hand)

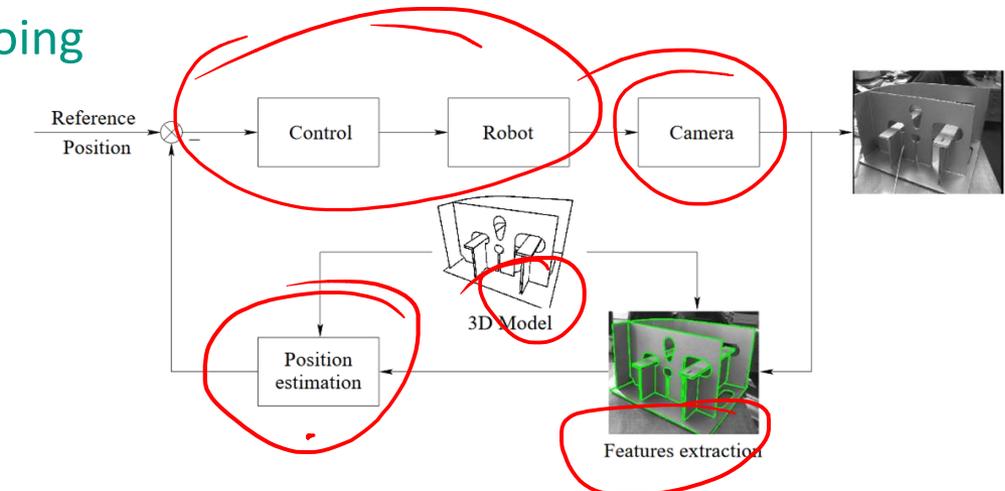
- Externes Kamerasystem wird zur Beobachtung der Bewegung genutzt



Visual Servoing - Verfahren

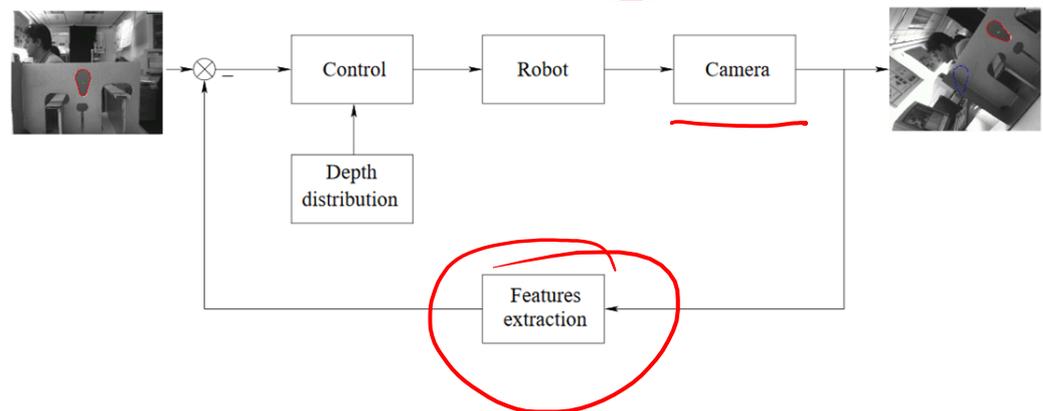
■ Positionsbasiertes Visual Servoing

- Position-based visual servo (PBVS)
- Einfache Regler
- 3D Rekonstruktion aufwendig



■ Bildbasiertes Visual Servoing

- Image-based visual servo (IBVS)
- Merkmalsextraktion einfach
- Komplexere Regelungsvorschriften

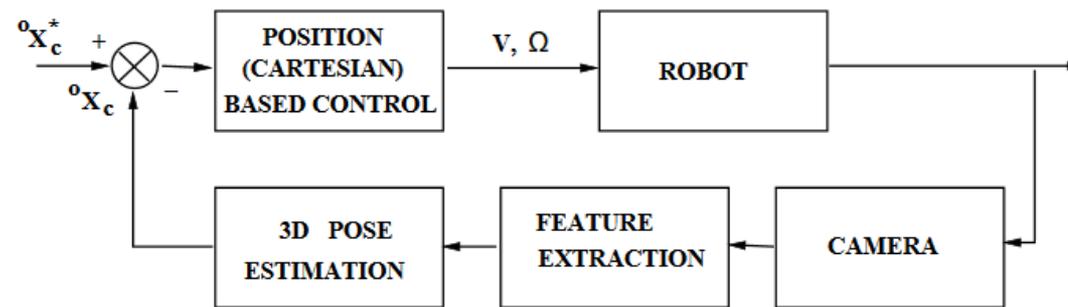
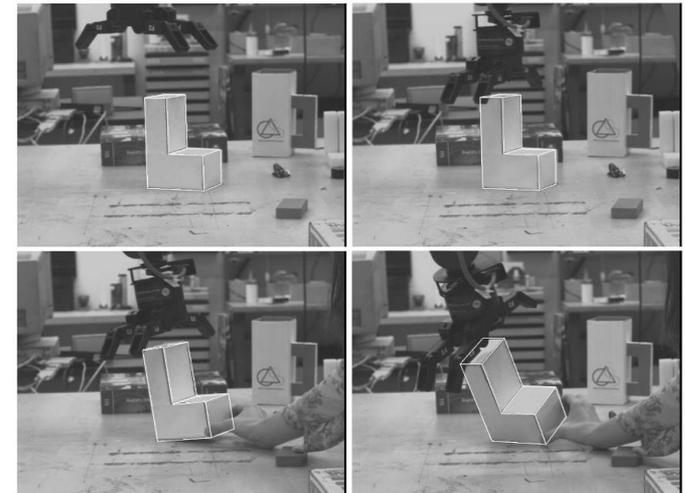


■ Hybride Verfahren

(z.B. 2,5D visual servo)

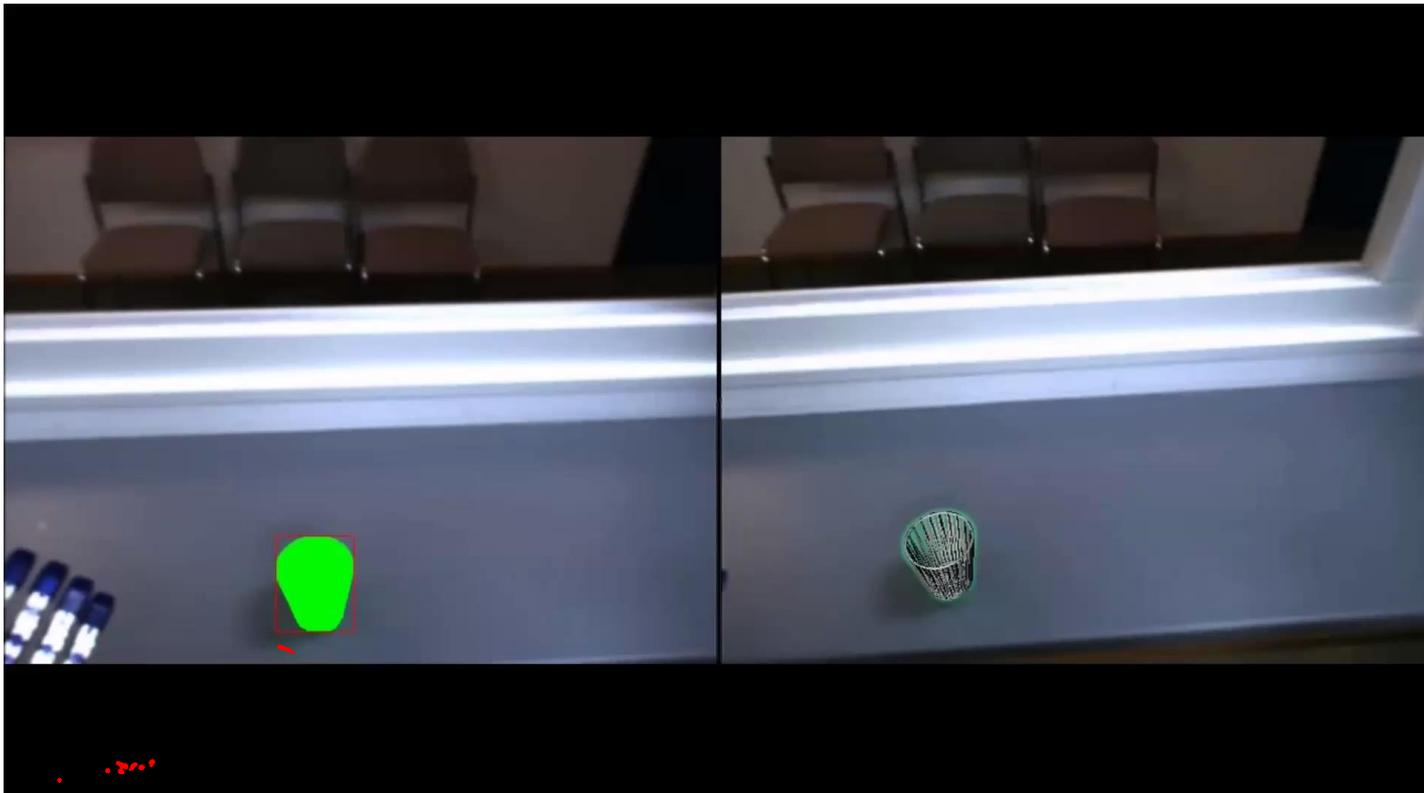
Positionsbasiertes Visual Servoing

- Zielpose X_g ist vorgegeben
- Ablauf der Regelschleife
 - 3D Lage Schätzung:
Aktuelle Pose X_c des Zielobjektes (bzw. Hand) wird aus Bildmerkmalen extrahiert
 - Regelvorgabe (kartesisch):
Differenz $\Delta X = X_g - X_c$
 - Kartesischer Regler führt ΔX aus
 - Ziel erreicht:
Distanz ΔX kleiner als Schwellwert



Positionsbasiertes Visual Servoing

- Beispiel auf ARMAR-III



Bildbasiertes Visual Servoing

■ Ansatz

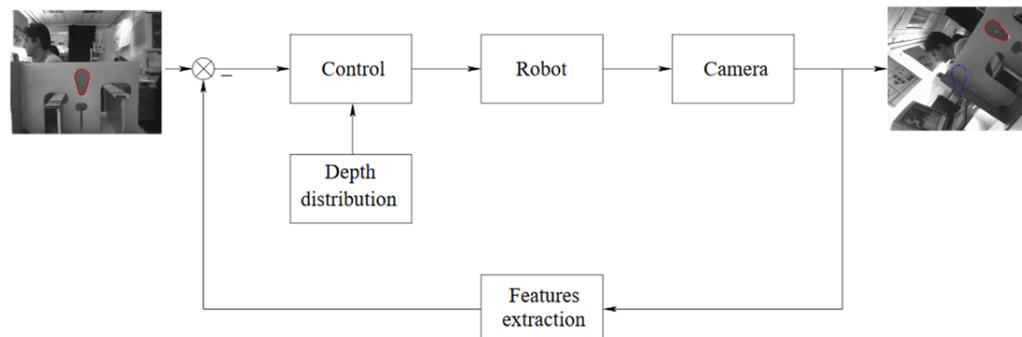
Die Bewegung des Manipulators ergibt sich aus der aktuellen und gewünschten Position von Bildmerkmalen

■ Bildmerkmale

Bildverarbeitungsmethoden zur Extraktion von Bildmerkmalen

■ Regelung

Geschwindigkeitsvorgaben werden direkt aus der aktuellen und gewünschten Stellung der Bildmerkmale erzeugt



Visual Servoing für ARMAR-III

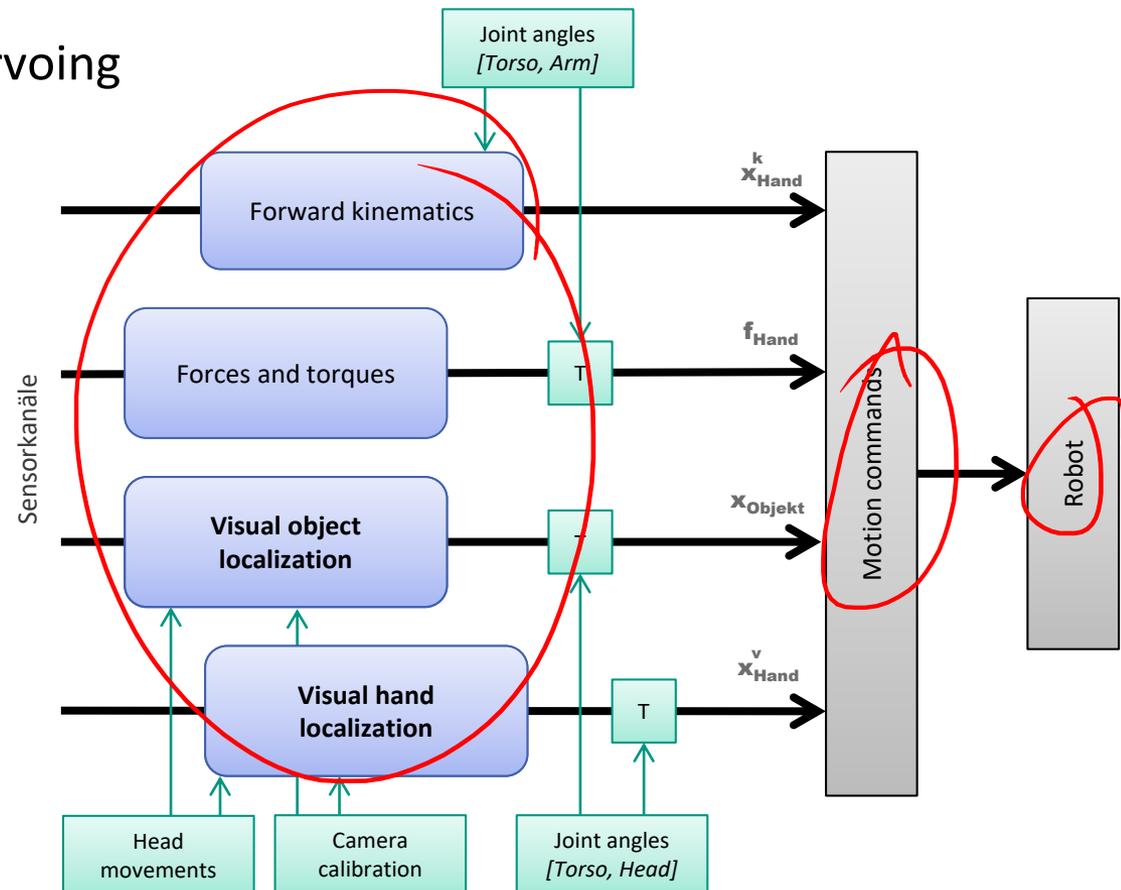
■ Ausführung von Greif und Manipulationsaufgaben

■ Positionsbasiertes Visual Servoing

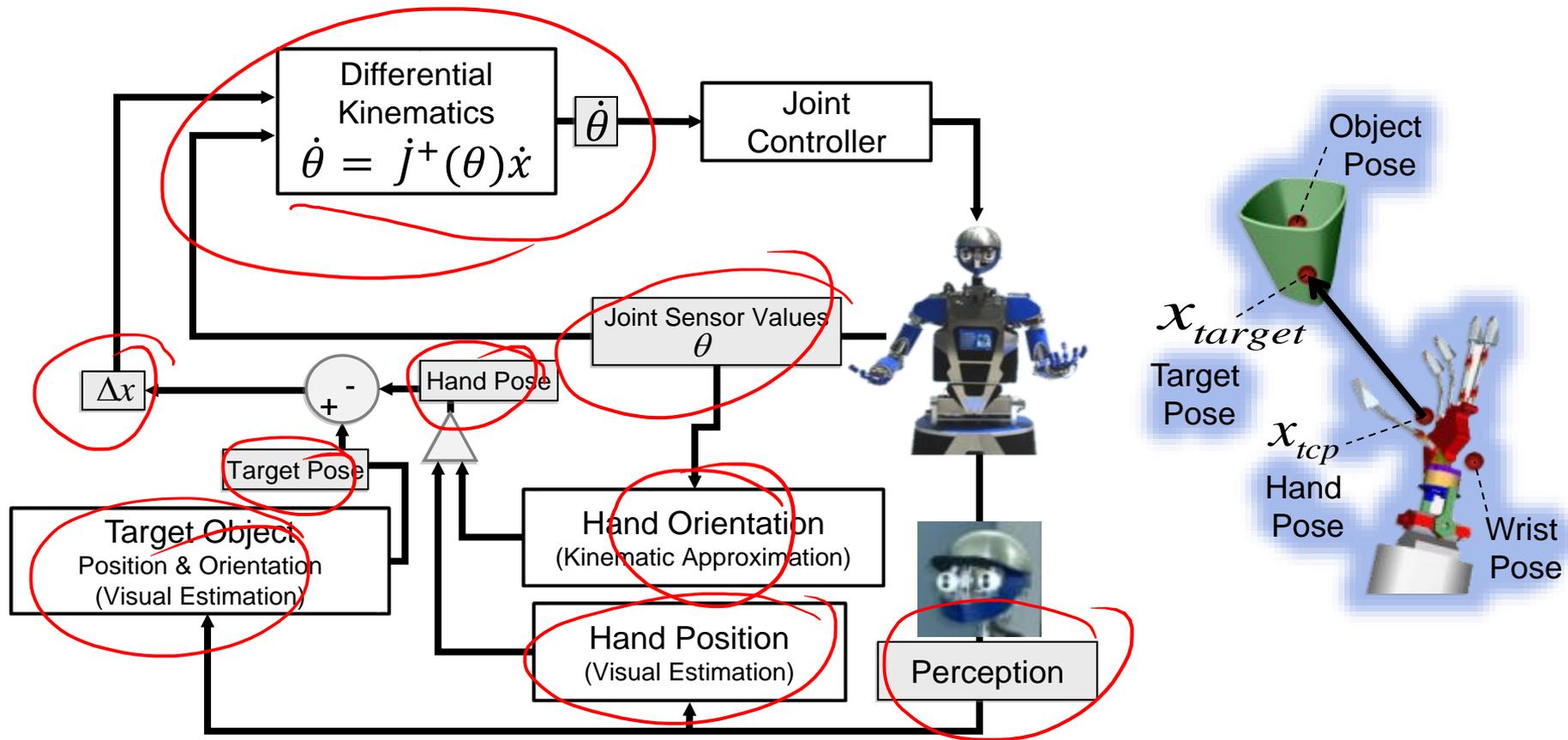
- Modellwissen
- Objektlokalisierung
- Handlokalisierung

■ Sensoren

- Kraft/Kontakt
- Kameras
- Interne Sensoren



Positionsbasiertes Visual Servoing auf ARMAR-III

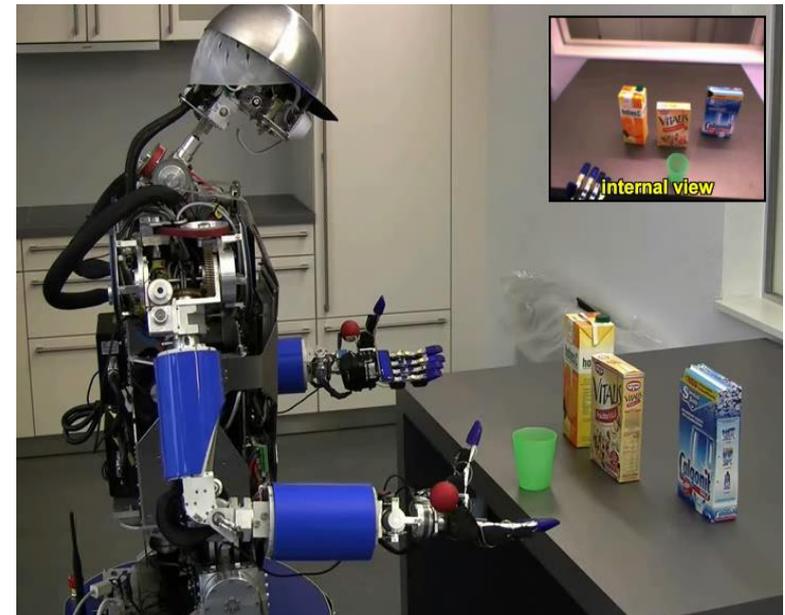
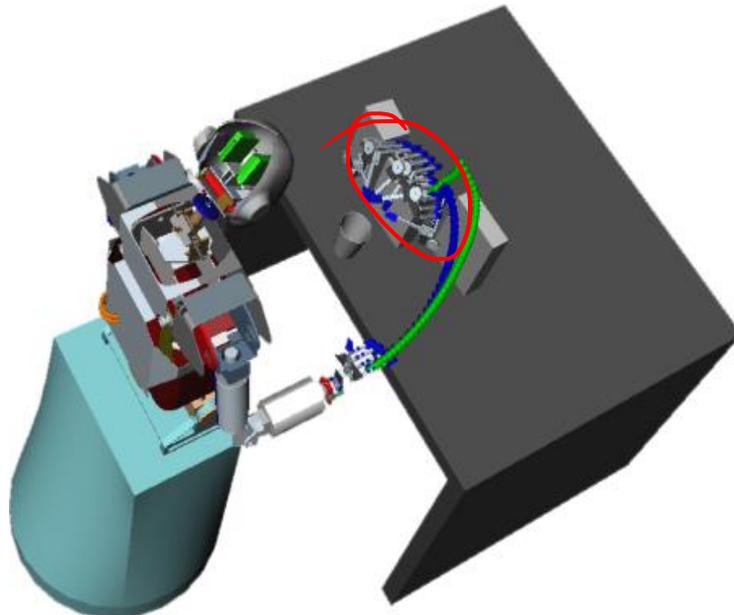
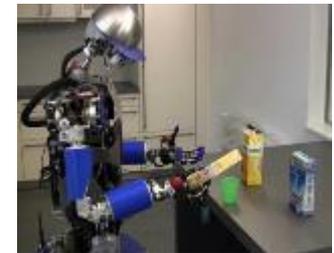
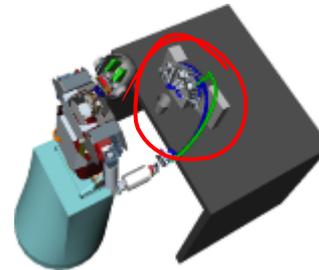
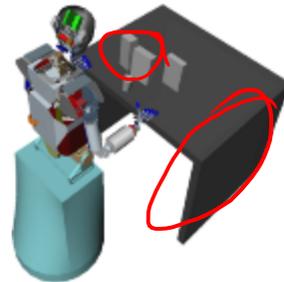
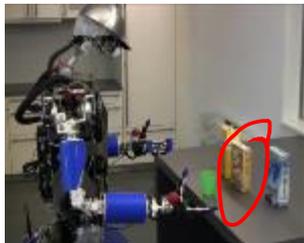
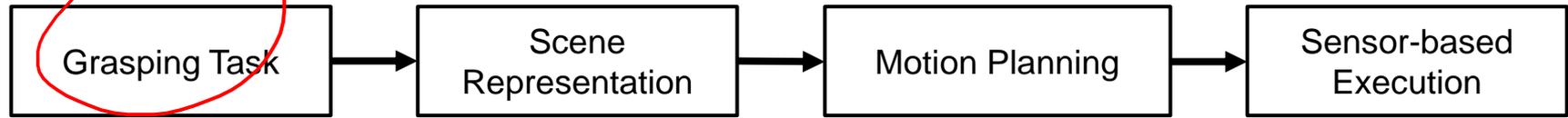


$$\dot{\theta} = j^+(\theta)\dot{x}$$

$$\delta^t = x_{vision}^t - x_{kinematic}^t$$

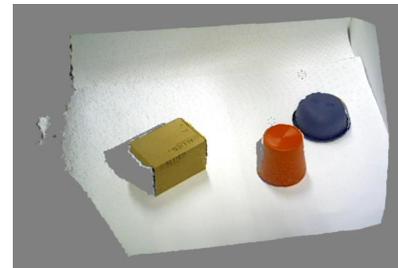
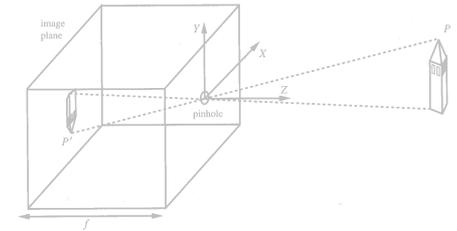
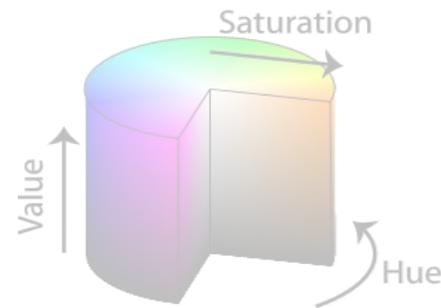
$$x_{tcp}^{t+1} = x_{kinematic}^{t+1} + \delta_{tcp}^t$$

Ausführung von Manipulationsaufgaben



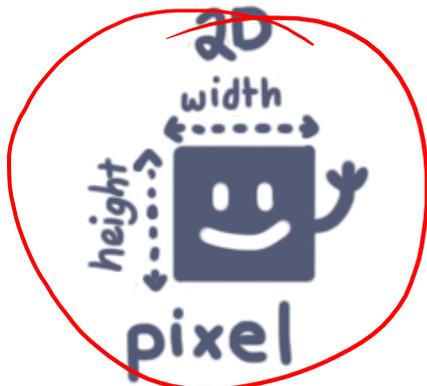
Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- **Punktwolken**
- SLAM
- Anwendungsbeispiel



Point Clouds

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

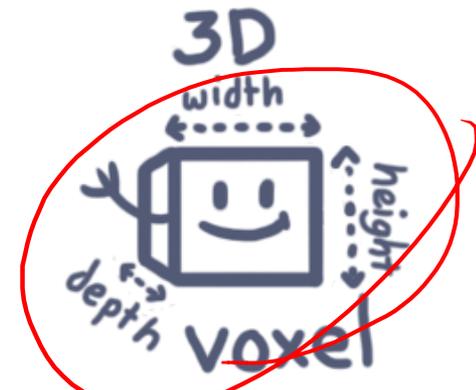


Pixel = Picture Elements

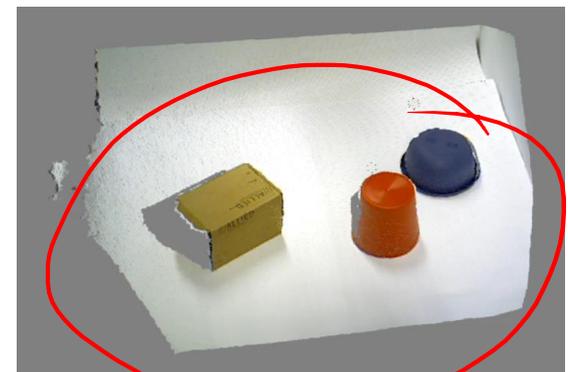


2D Bild

VS.



Voxel = Volumetric Pixel



3D Punktwolke

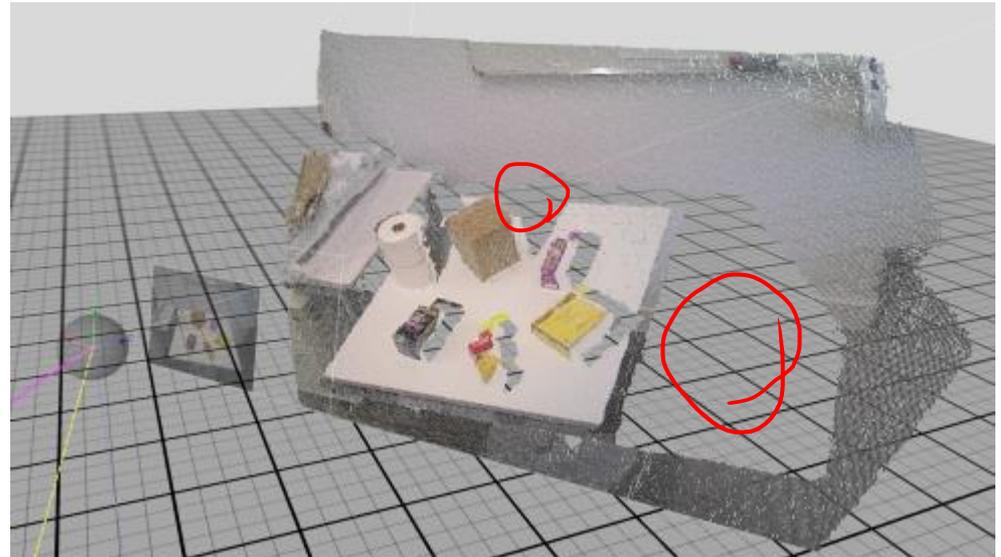
Point Clouds

- Punktvolke $P = \{(X, C) \mid X \in \mathbb{R}^3, C \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$
 - $X = (x, y, z)$ Ortsinformation
 - $C = (r, g, b)$ Farbinformation
 - Es können weitere (Sensor-)informationen abgespeichert werden
- Zwei verschiedene Arten der Repräsentation
 - **Organisierte** Punktvolke (2D Array Format, Größe muss vorab bekannt sein)
 - **Unorganisierte** Punktvolke (Vector Format)

Normalenschätzung in 3D Punktwolken

Ziel:

- Zusätzliche Oberflächeninformation durch Einbeziehung von lokalen Nachbarpunkten
- Grundlage für weiterführende Algorithmen
 - Segmentierung
 - Deskriptoren
 - Objekterkennung
 - Oberflächenmodellierung



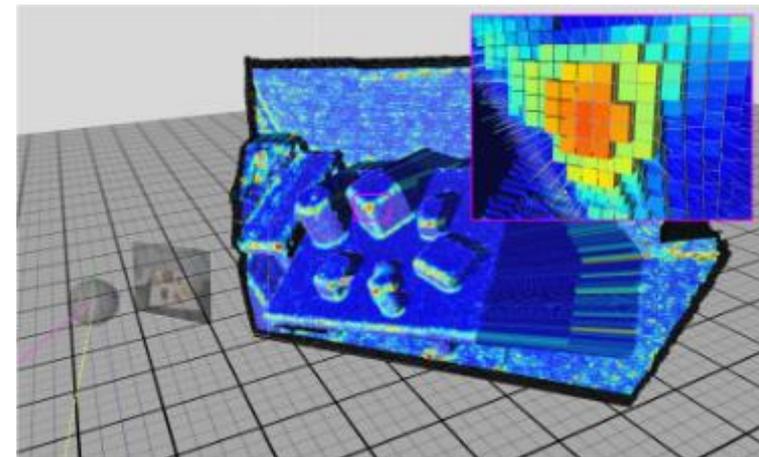
Normalenschätzung in 3D Punktwolken

- 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 mit der 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



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]:

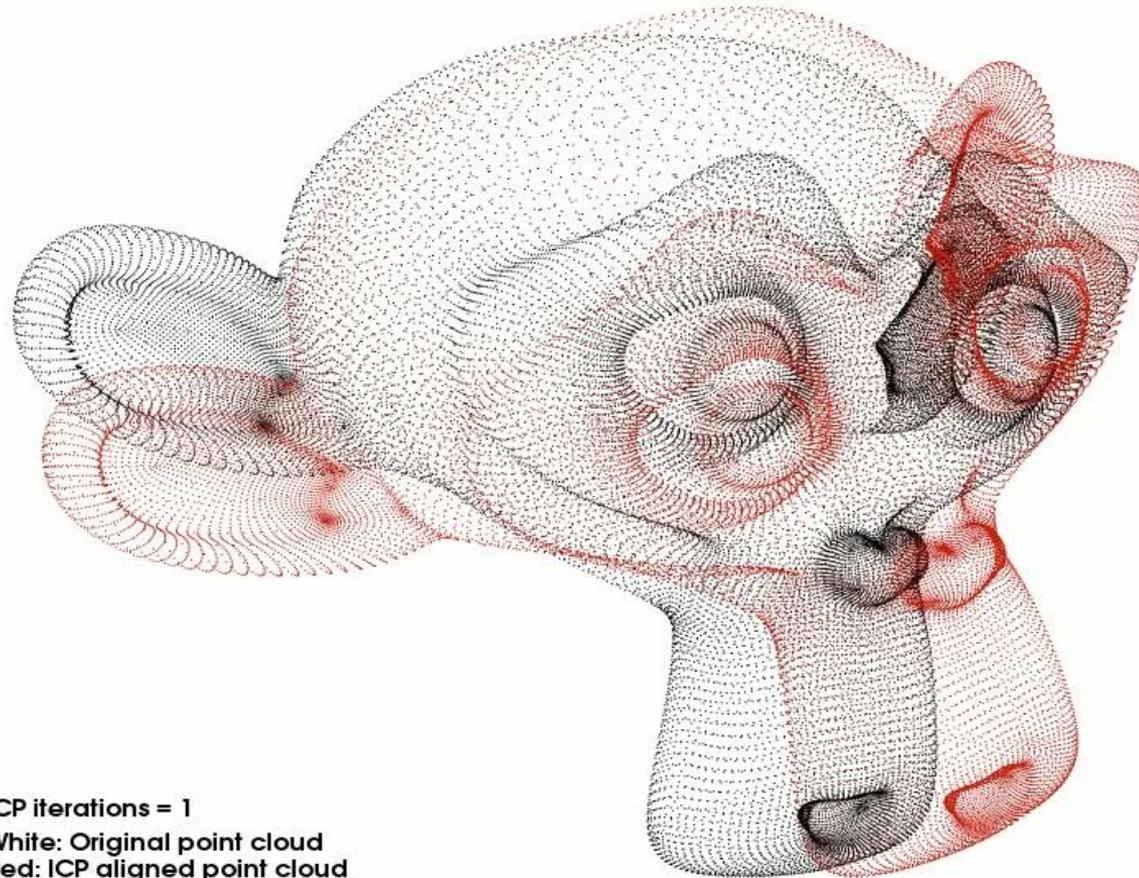
$$D_k = \sum_i \|a_i - T_k \cdot b_i\|^2$$

- D_k Kombiniert Translation und Rotation
- Wende Transformation T_k auf alle Punkte aus B an (Update)
- Abbruchkriterien:
 - Schwellwert für $D_{k-1} - D_k$
 - Maximale Anzahl an Iterationen erreicht

Iterative Closest Point II

- 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
 - Symmetrische Objekte können nicht ohne weiteres registriert werden
 - Konvergenz in ein lokales Minimum möglich
 - Überlappung der Punktwolken erforderlich

Iterative Closest Point III



ICP iterations = 1
White: Original point cloud
Red: ICP aligned point cloud

RANSAC (Random Sample Consensus)

- **RANSAC** ist eine **iterative Methode** zur Schätzung von Modellparametern aus Datenpunkten
- RANSAC ist ein **nicht-deterministischer** Algorithmus
- 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

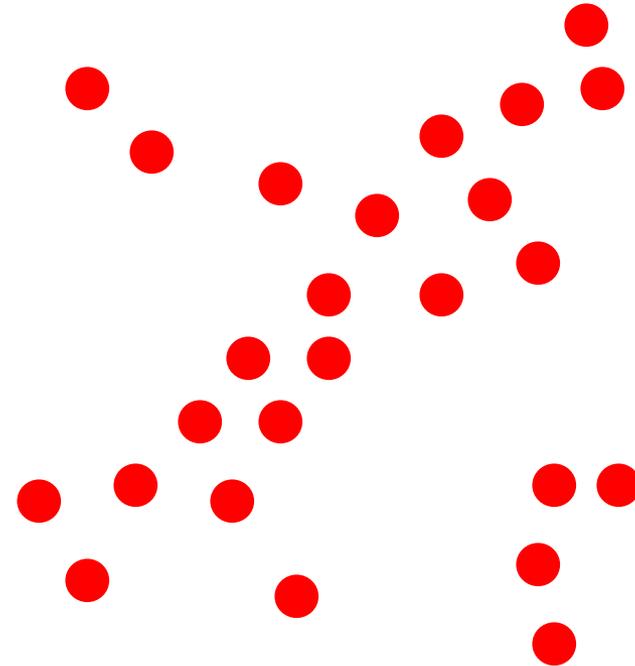
RANSAC (Random Sample Consensus)

■ Der RANSAC Algorithmus:

1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
 - 2 Punkte für Linien in 2D
 - 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)

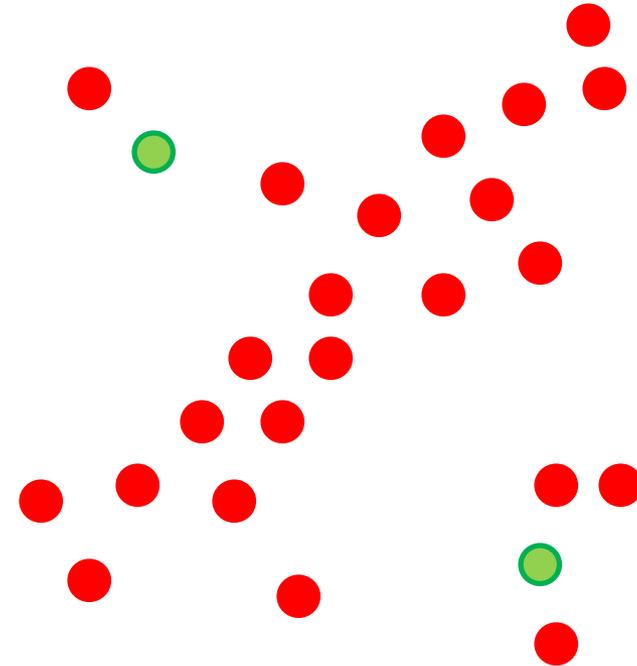
- Beispiel:
 - Line fitting in 2D Datenpunkte



- 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)

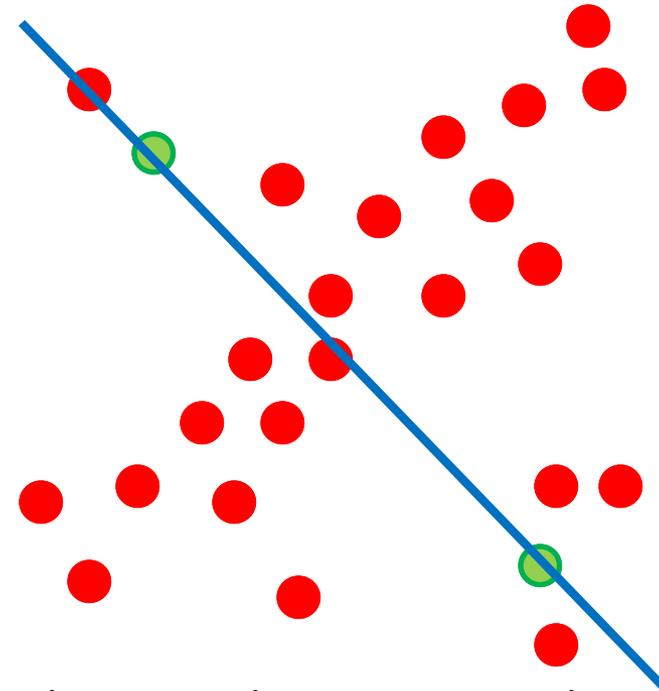
- Beispiel:
 - Line fitting in 2D Datenpunkte



- 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)

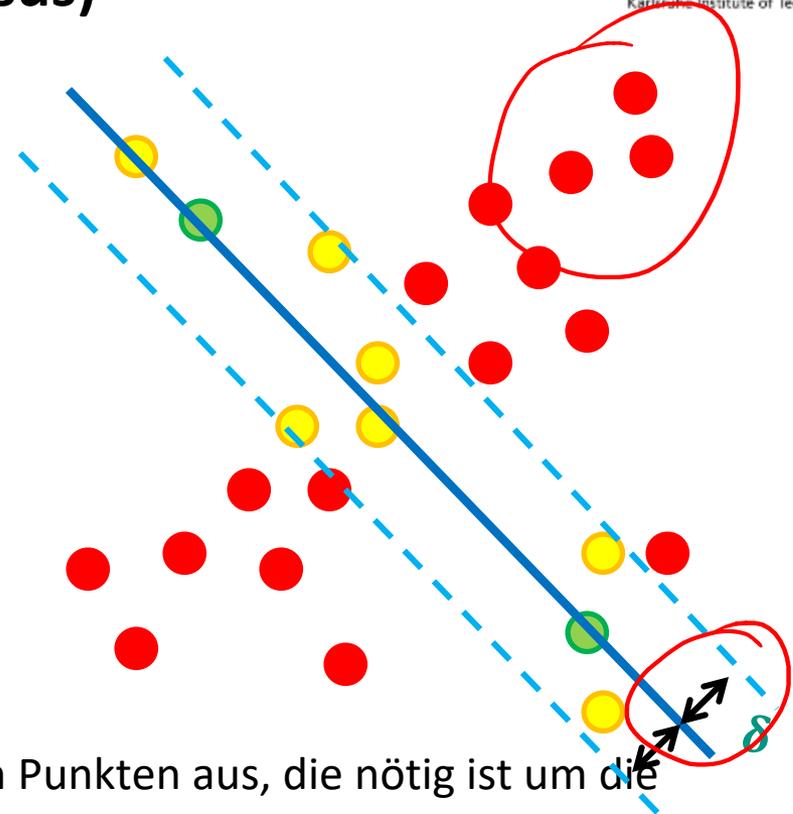
- Beispiel:
 - Line fitting in 2D Datenpunkte



- 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)

- Beispiel:
 - Line fitting in 2D Datenpunkte



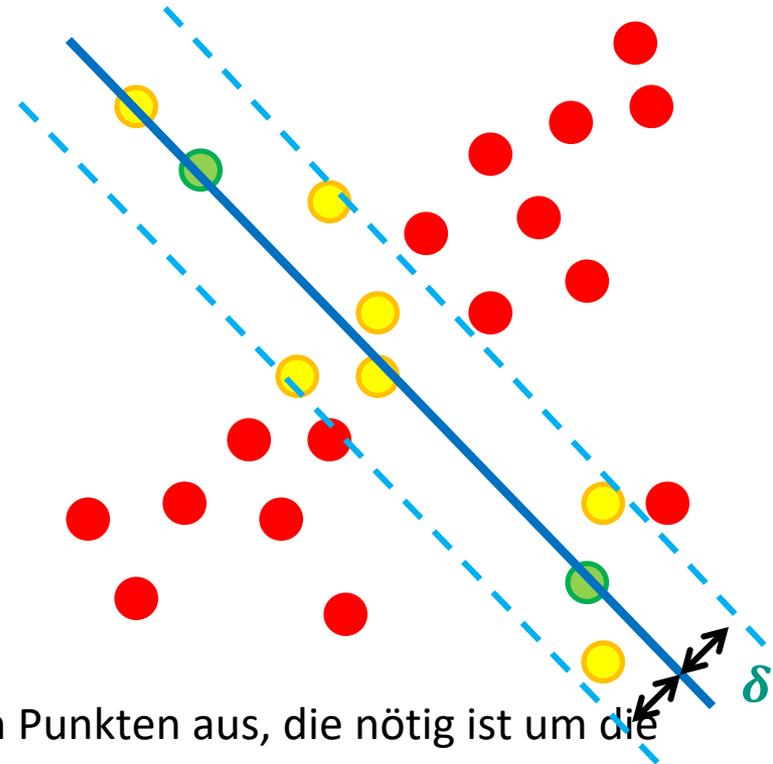
- RANSAC Algorithmus:
 1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
 2. Schätzung des Modell aus dem ausgewählten Datensatz
 - 3. Bewertung der Modellschätzung**
 4. Wiederhole 1-3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC (Random Sample Consensus)

- Beispiel:
 - Line fitting in 2D Datenpunkte

Inlier Anzahl = 9

Outlier Anzahl = 16



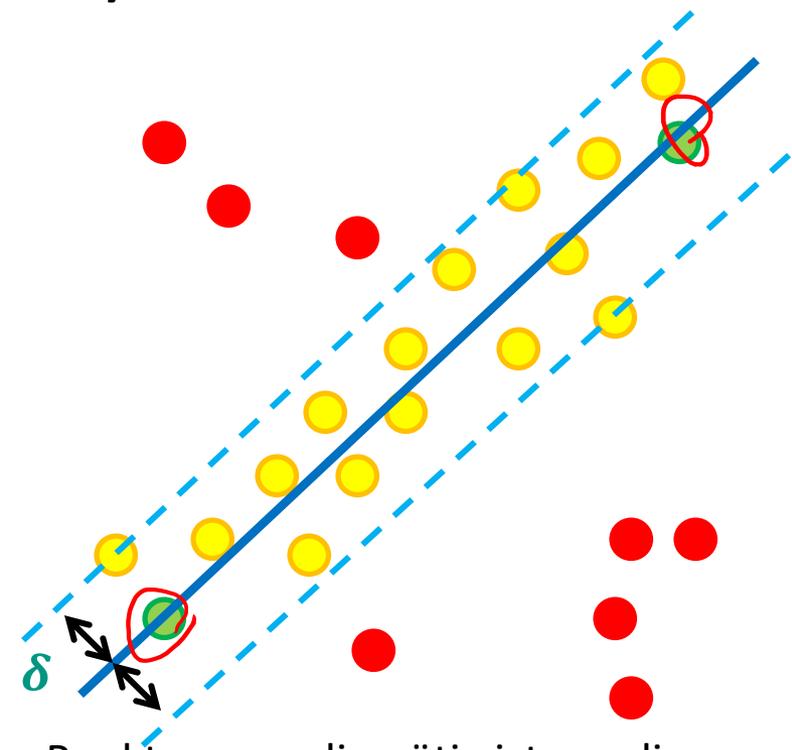
- RANSAC Algorithmus:
 1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
 2. Schätzung des Modell aus dem ausgewählten Datensatz
 - 3. Bewertung der Modellschätzung**
 4. Wiederhole 1-3 bis das Modell mit den meisten Inliers gefunden wird

RANSAC (Random Sample Consensus)

- 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)

■ Vorteile:

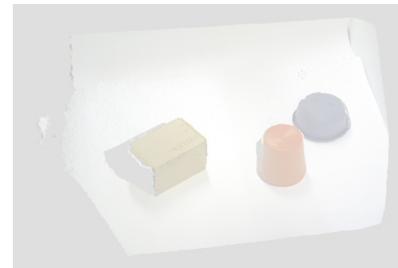
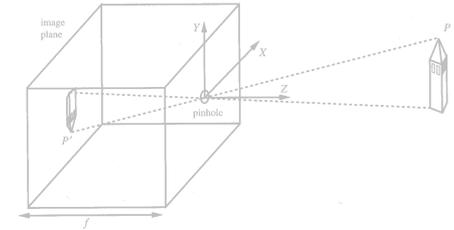
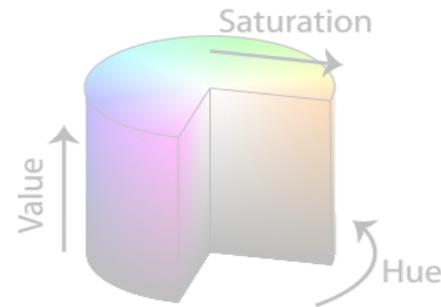
- Simpel, allgemein und einfach zu implementieren
- Robuste Modellschätzung für Daten mit wenigen Ausreißern
- Vielseitig anwendbar

■ Nachteile:

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

Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- **SLAM**
- Anwendungsbeispiel



Simultaneous Localization and Mapping (SLAM)

■ Das SLAM Problem:

Wie kann ein Roboter in einer unbekanntem Umgebung **navigieren** und dabei eine Karte von seiner Umgebung **erstellen** und auch **aktualisieren**?

- SLAM bedeutet die Schätzung der aktuellen Roboterpose als auch der Karte der Umgebung **zur gleichen Zeit**.

■ SLAM wird benötigt

- um voll autonome Roboter zu bauen
- um in einer unbekanntem Umgebung zu überleben
- um zu wissen wo sich der Roboter gerade befindet
- um die Navigation ohne externe Positionsbestimmung (z.B. GPS) zu ermöglichen.

Simultaneous Localization and Mapping (SLAM)

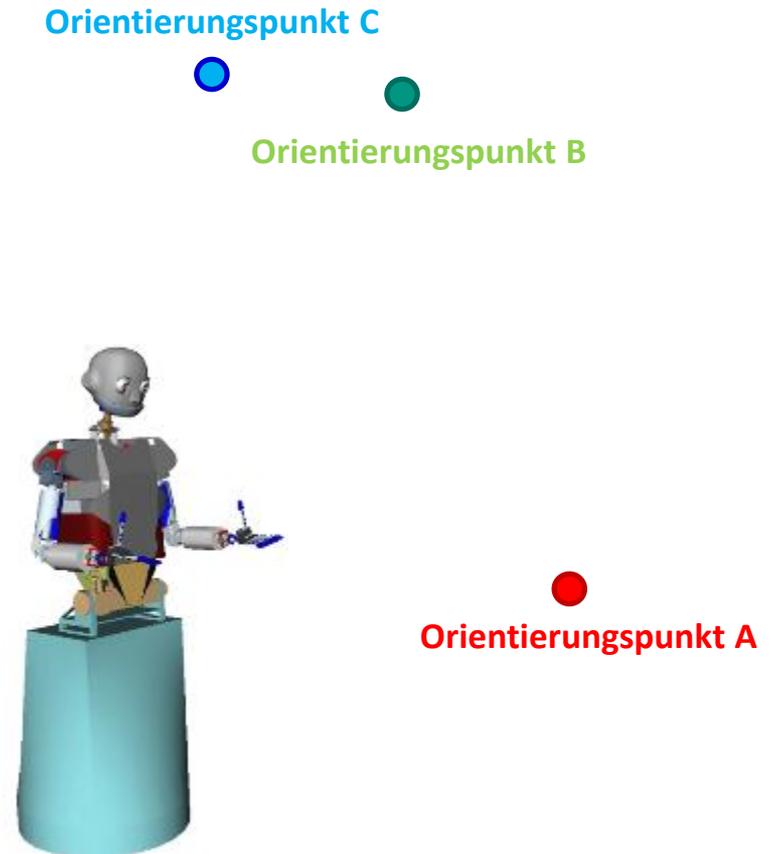
- Es entsteht ein Henne-Ei-Problem. Wir brauchen
 - eine Karte, um den Roboter zu **lokalisieren** (aber SLAM hat kein Vorwissen über die Umgebung)
 - und eine genaue Schätzung der Position, um eine **Karte** zu erstellen
- Begriffe:
 - **Lokalisierung**: Schätzen der Roboterposition bei gegebener Karte
 - **Kartierung**: Aus einer Menge von Positionen eine Karte ableiten
 - **SLAM**: Gleichzeitige Lokalisierung und Kartierung

Simultaneous Localization and Mapping (SLAM)

- Lösung:
Position von Kamera und Roboter während einer Bewegung verfolgen
- Deshalb braucht der Roboter **Features**, die er zuordnen und verfolgen kann
 - SLAM wählt Szenenfeatures als **Orientierungspunkte**
 - Visuelle Features: Punkte, Ecken, Kanten, Texturen, Oberflächen
 - Hinweis: Features müssen unterscheidbar und deskriptiv sein
(Invariant gegenüber Standpunktwechsel)

Simultaneous Localization and Mapping (SLAM)

- Wie funktioniert SLAM?
 - Interne Repräsentation für:
 - Position der Orientierungspunkte
 - Parameter der Kamera
 - In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat

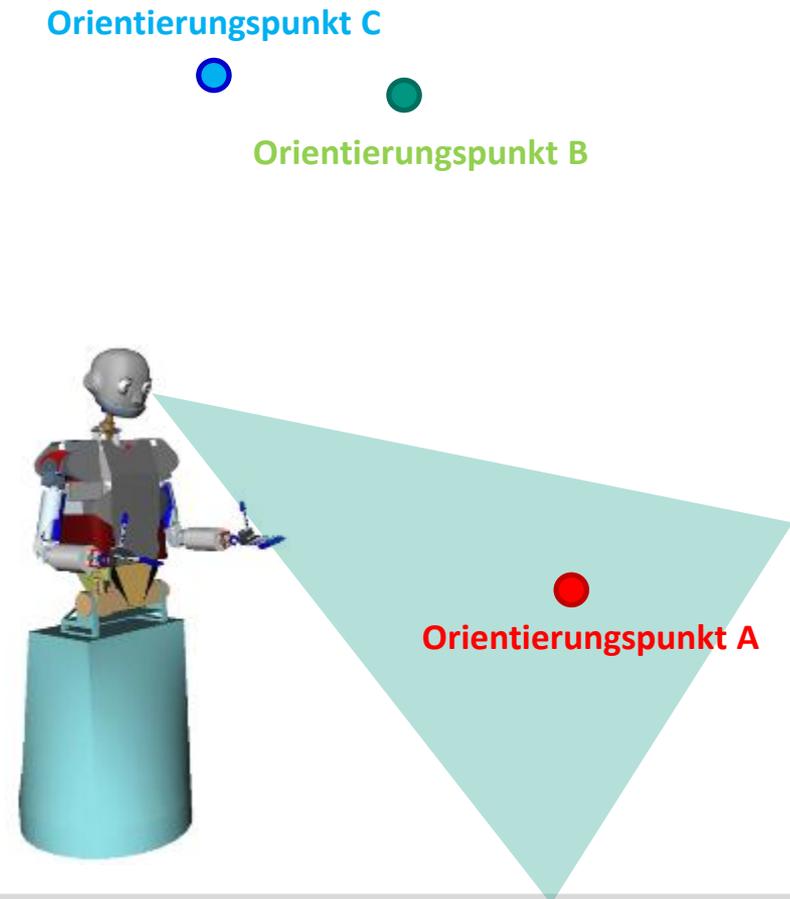


Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkte
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen



Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

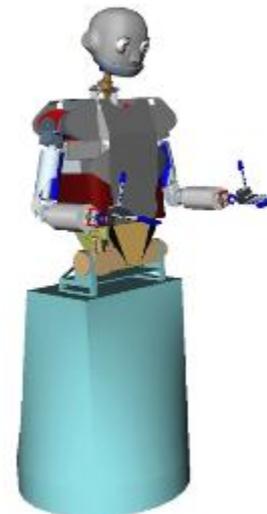
- Interne Repräsentation für:
 - Position der Orientierungspunkte
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)

Orientierungspunkt C



Orientierungspunkt B



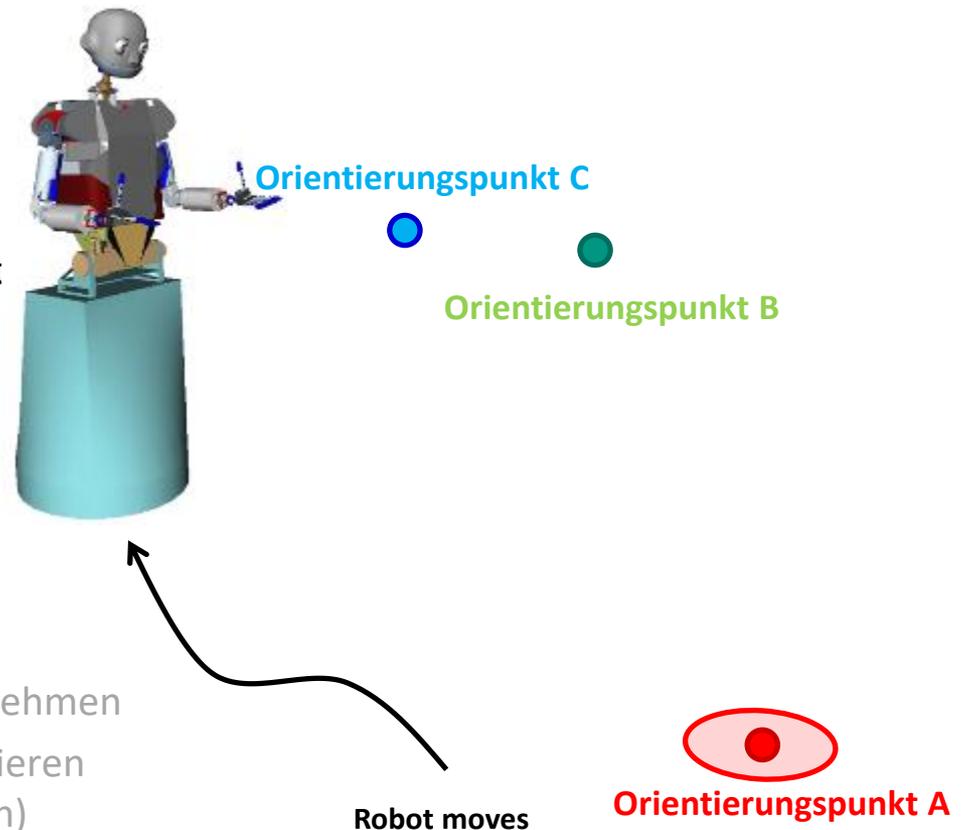
Orientierungspunkt A

Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkt
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)

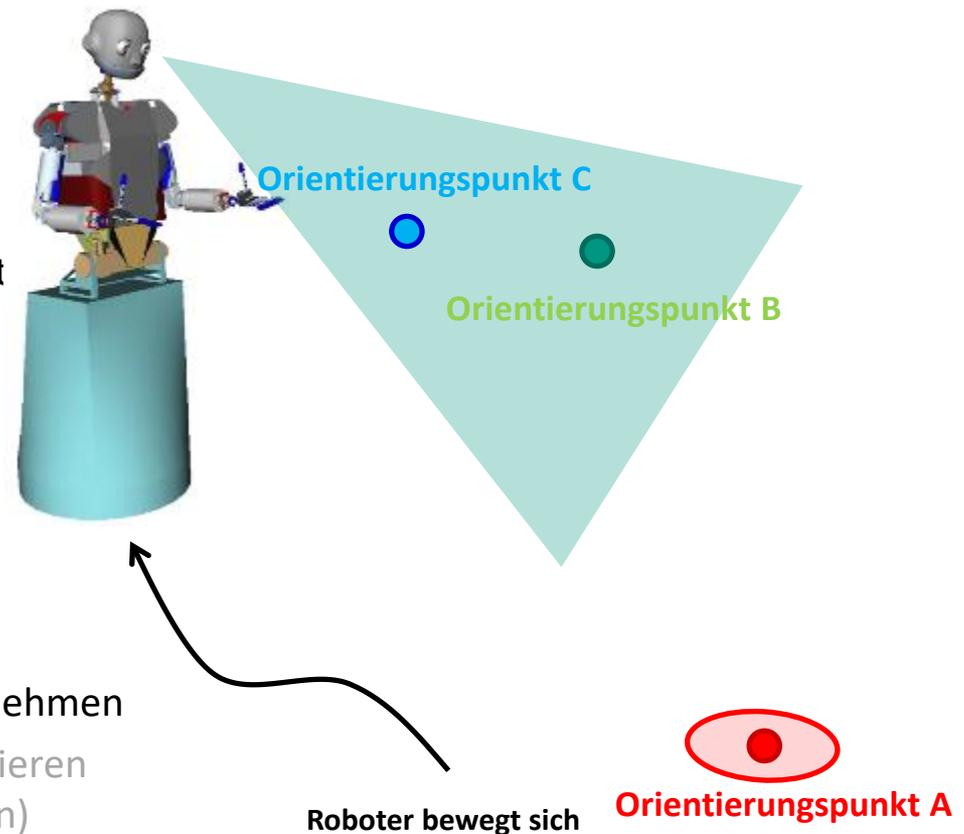


Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkt
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)

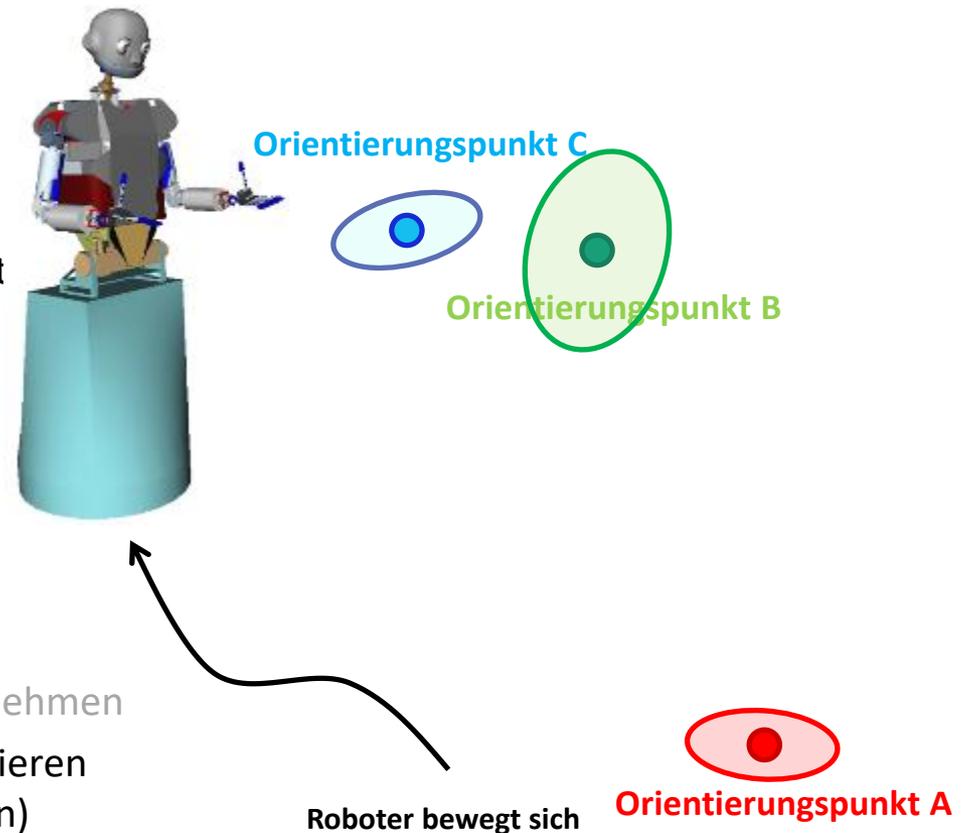


Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkt
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)

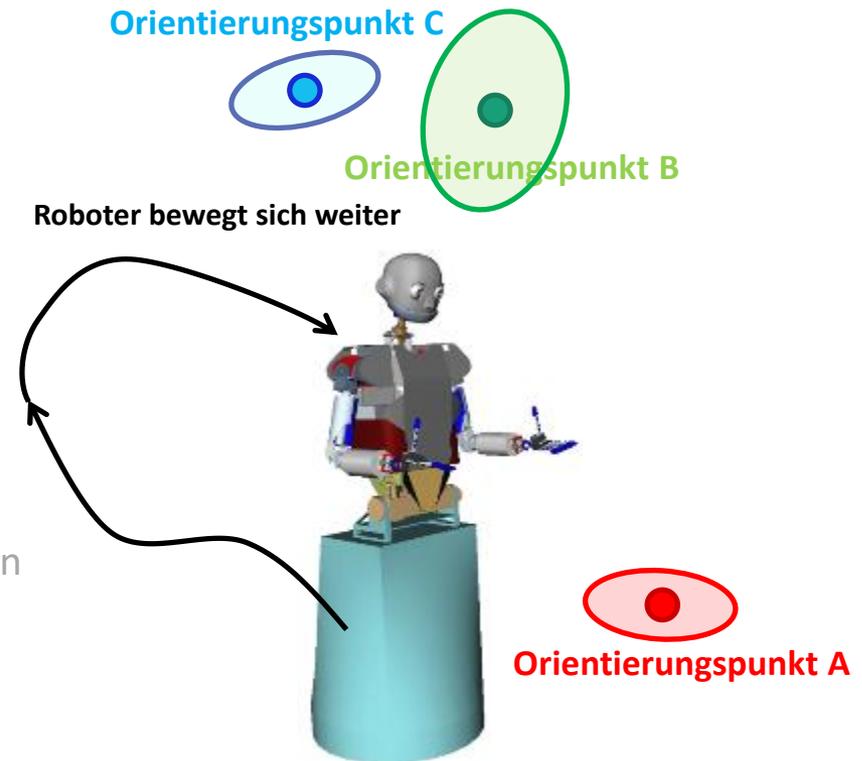


Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkte
 - Parameter der Kamera

- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)



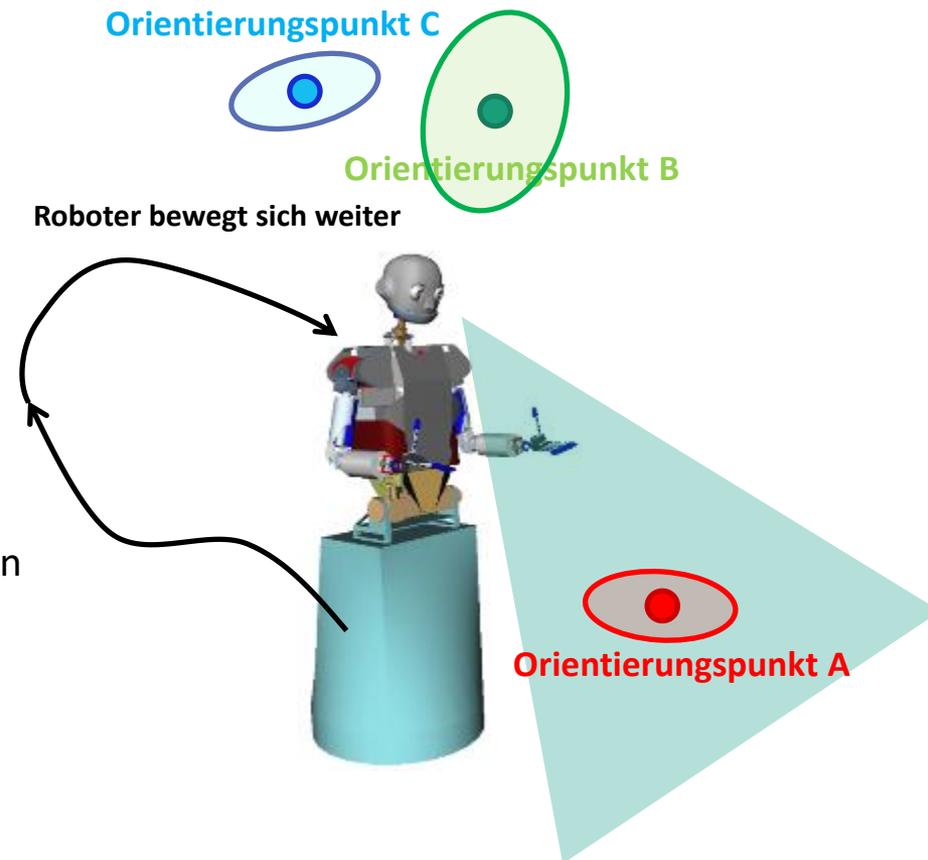
Simultaneous Localization and Mapping (SLAM)

■ Wie funktioniert SLAM?

- Interne Repräsentation für:
 - Position der Orientierungspunkte
 - Parameter der Kamera

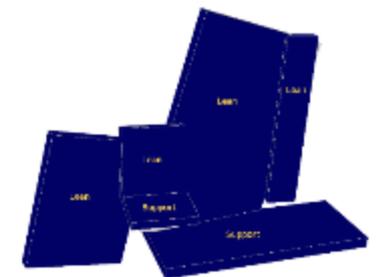
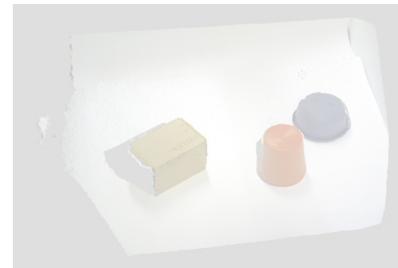
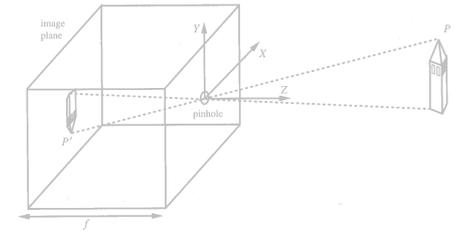
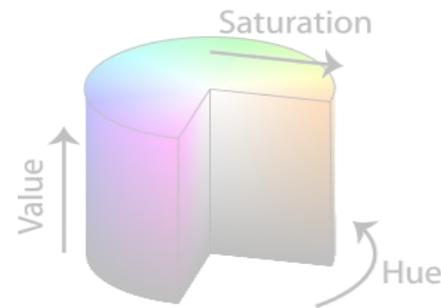
- In jedem Frame:
 - Vorhersage, wie viel sich der Roboter bewegt hat
 - Neue Orientierungspunkte aufnehmen
 - Interne Repräsentation aktualisieren (Messungenauigkeiten beachten)

Roboter sieht bekanntes Feature:
Es kann eine Zuordnung stattfinden



Inhalt

- Bildrepräsentation
- Kameramodell
- Filteroperationen
- Segmentierung
- Morphologische Operatoren
- Canny-Kantendetektor
- Tiefenkameras
- Visual Servoing
- Punktwolken
- SLAM
- **Anwendungsbeispiel**



Beispiel: Perceptual Pipeline

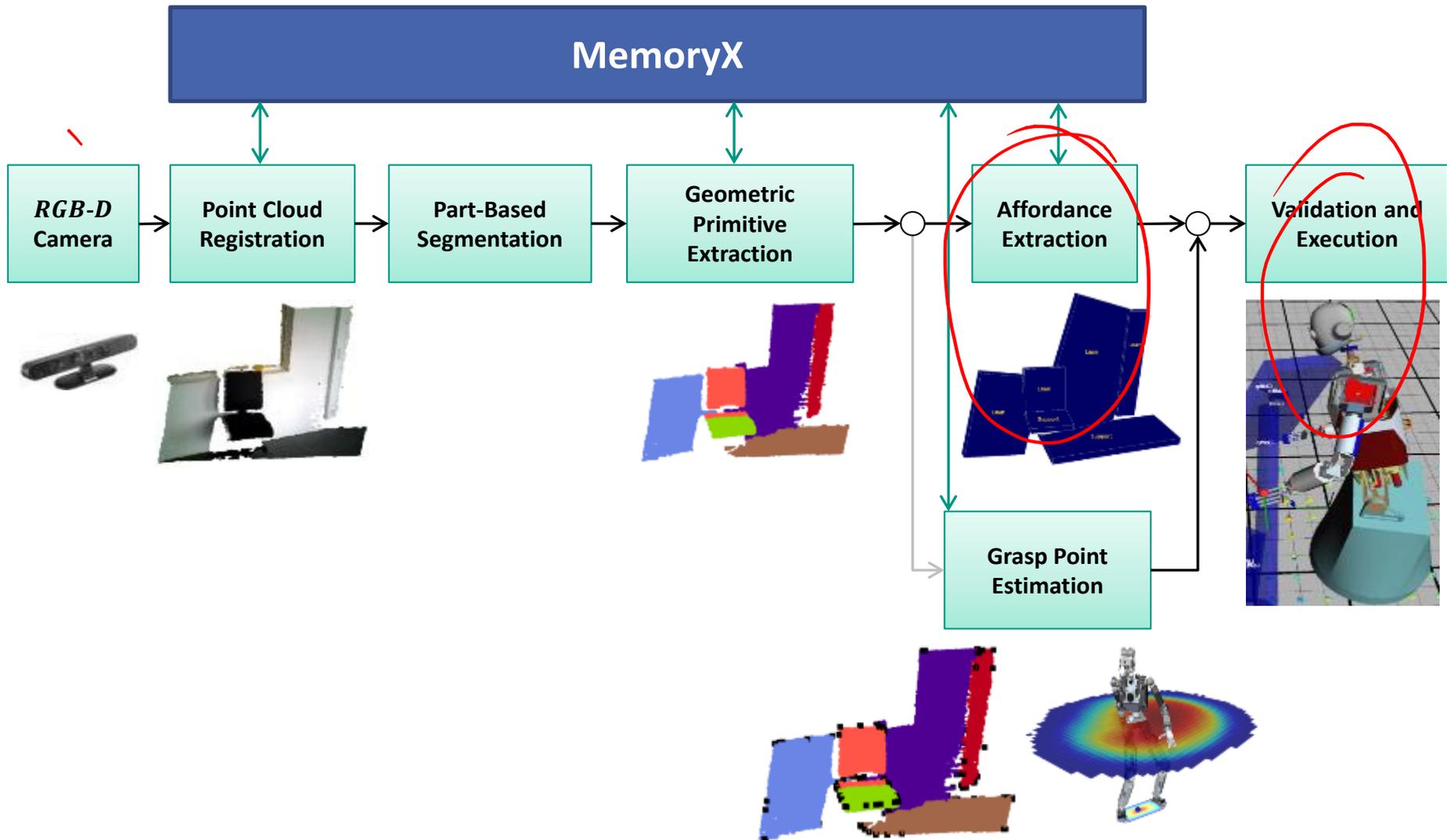
- SLAM
- RANSAC

- Integriert in das Roboter-Softwareframework ArmarX (späteres Kapitel)

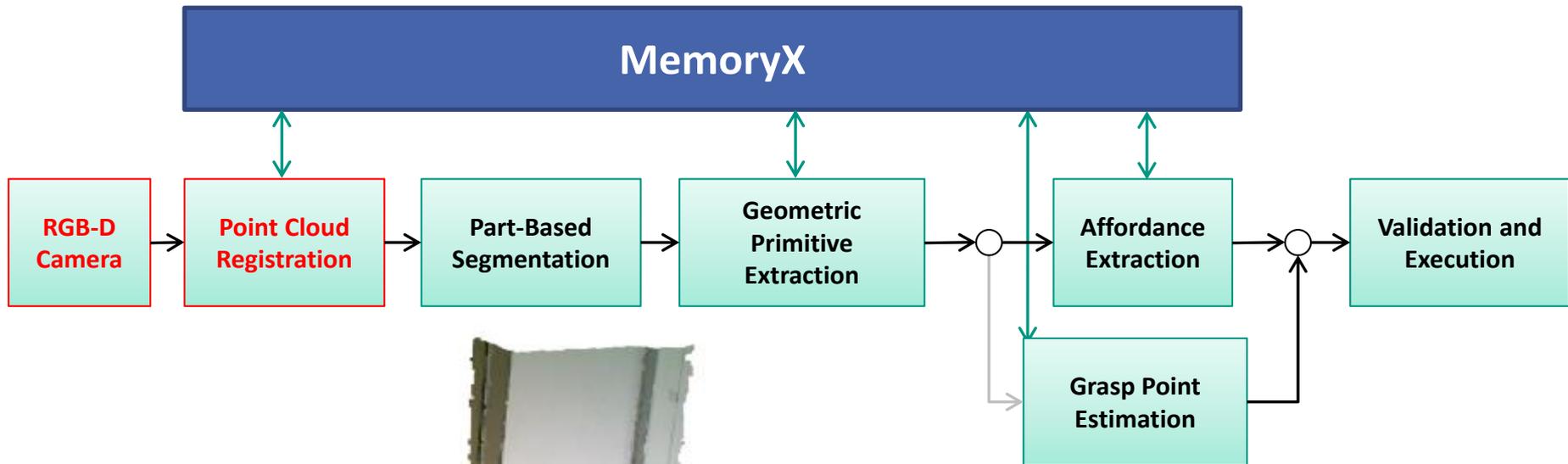
Loco-Manipulation Affordances

Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T.,
“Validation of Whole-Body Loco-Manipulation Affordances for Pushability and
Liftability”, In IEEE-RAS International Conference on Humanoid Robots 2015.

Loco-Manipulation Affordances



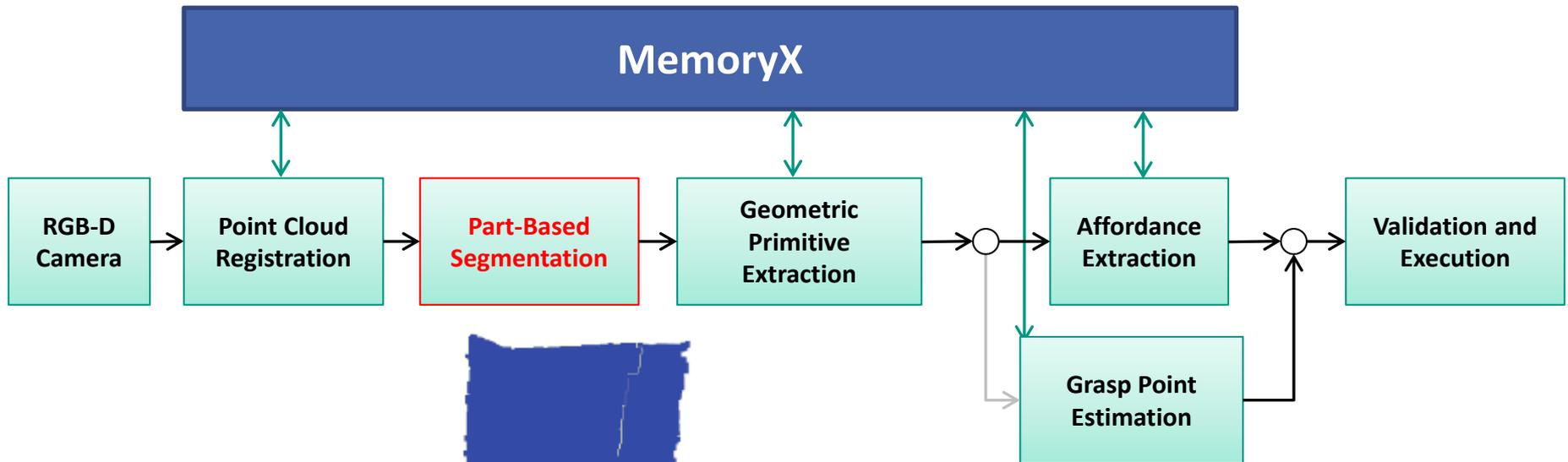
Loco-Manipulation Affordances



- Daten aufgenommen mit der Asus Xtion Pro Kamera
- Registrierung der Kamerabilder mit der OpenSource Bibliothek Real Time Appearance Based Mapping (RTAB-Map).

RTAB-Map: M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2661–2666, IEEE, 2014.

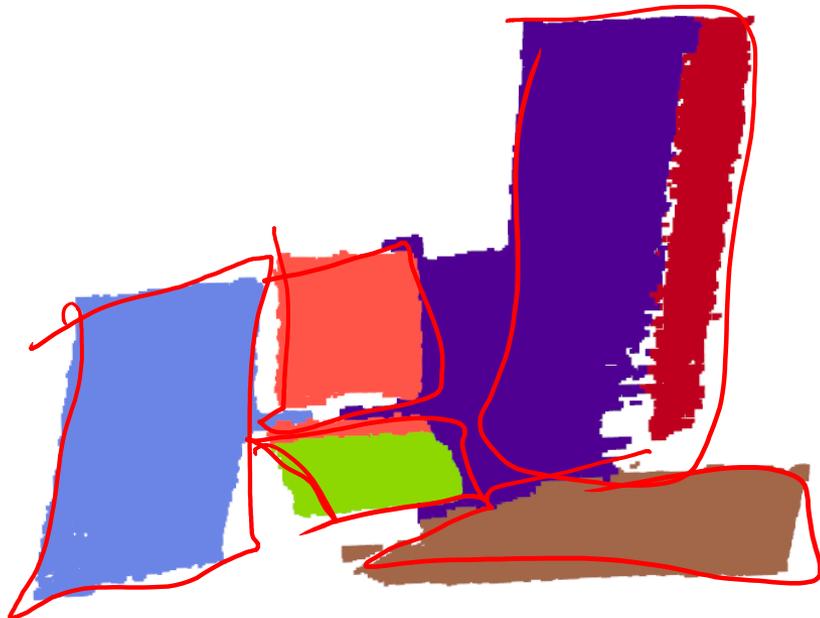
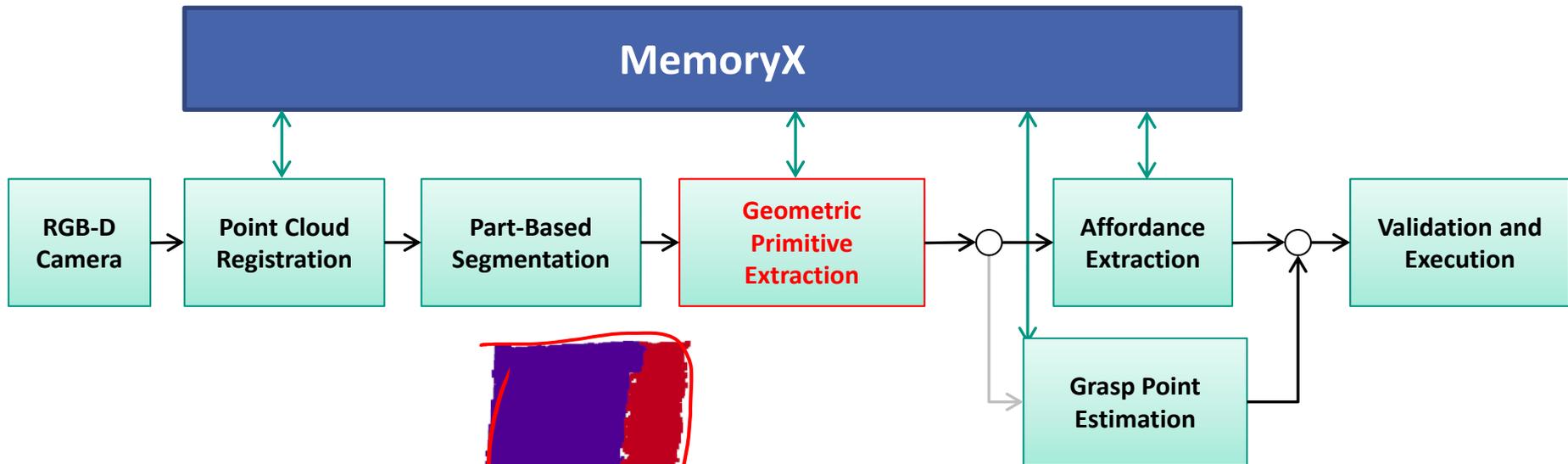
Loco-Manipulation Affordances



- Tiefenbasierte Segmentierung: benötigt für Szenenerlegung
- 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," in International Conference on Robotics and Automation ICRA, June 2014.

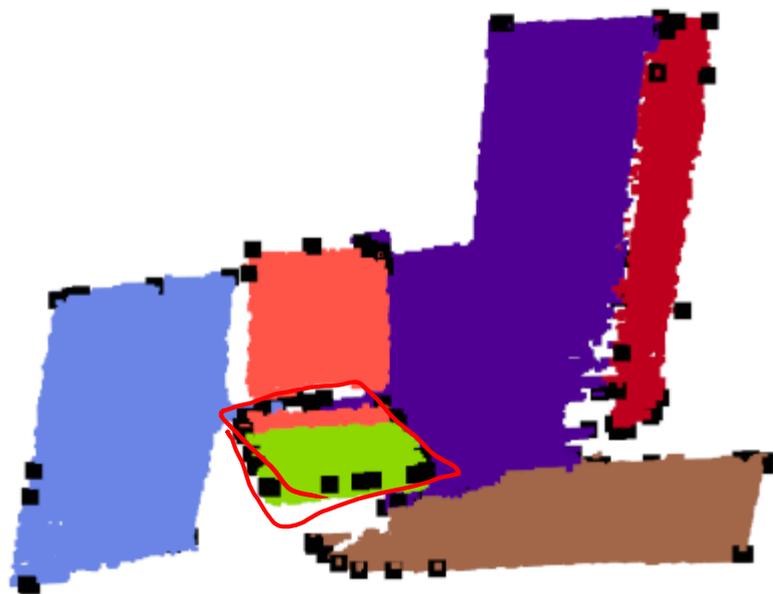
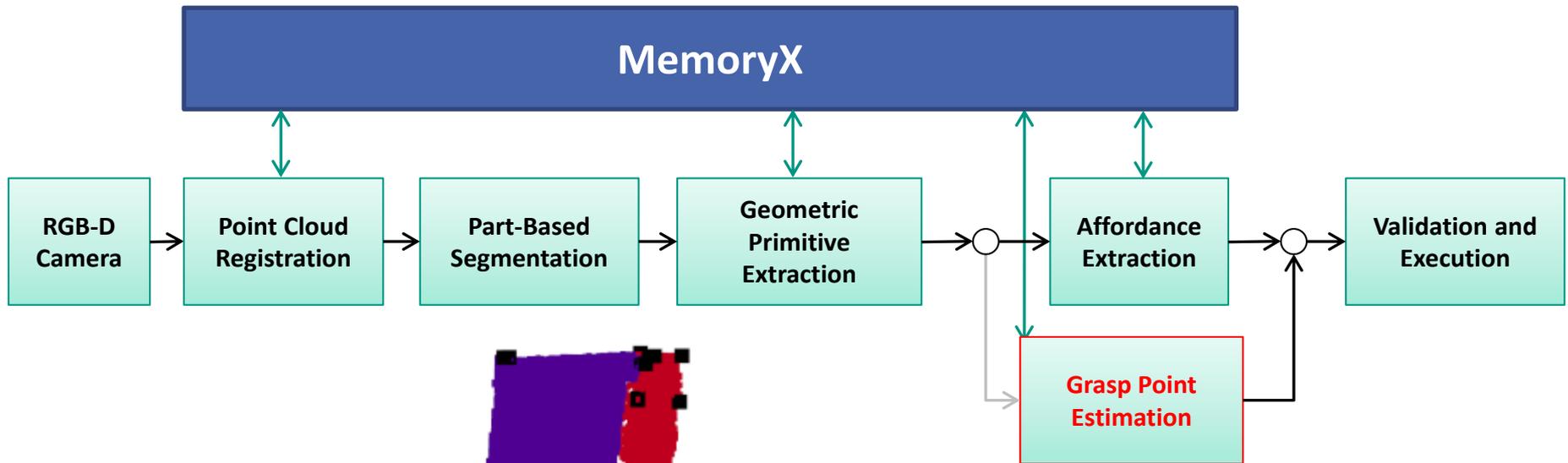
Loco-Manipulation Affordances



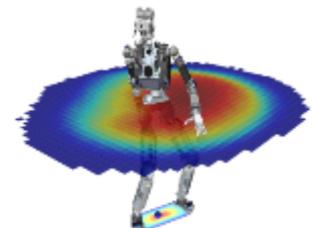
- Fitting von geometrischen Modellen (z.B. Ebenen und Zylinder) an Segmente mit RANSAC
- Wegen Untersegmentierung werden Modelle iterativ an die verbleibenden Outliers gefittet (z.B. Stuhl)

PCL: R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in International Conference on Robotics and Automation, 2011.

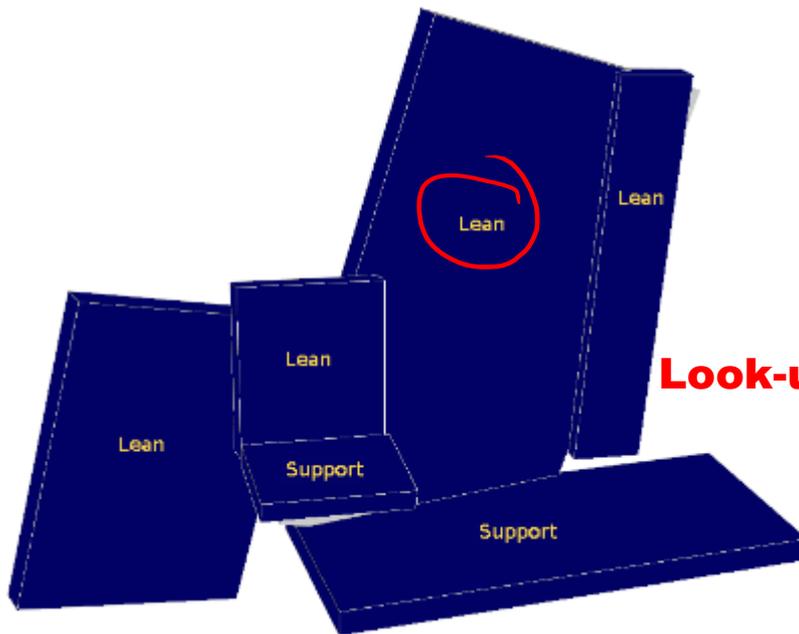
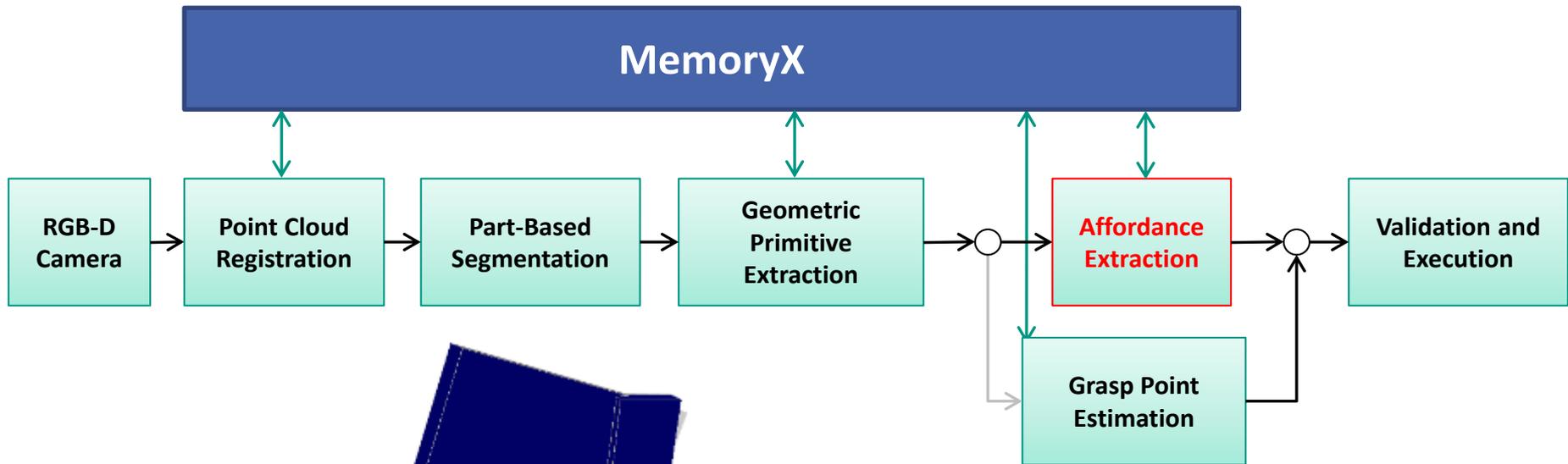
Loco-Manipulation Affordances



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



Loco-Manipulation Affordances



Look-up Tabelle!

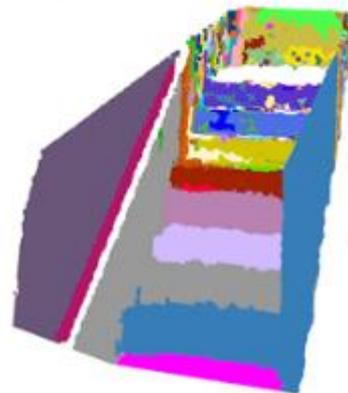
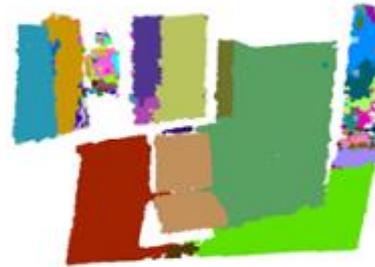
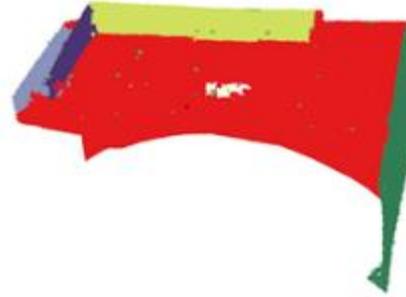
Affordance	Shape	Parameters	Conditions	Valid.
Support	Planar	Normal \mathbf{n} Area a	$\mathbf{n} \uparrow \mathbf{z}_{world}$ $a \geq \lambda_1$	(1a)
Lean	Planar	Normal \mathbf{n} Area a	$\mathbf{n} \perp \mathbf{z}_{world}$ $a \geq \lambda_2$	(1a)
Grasp	Planar	Normal \mathbf{n} Area a	$a \in [\lambda_3,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

Loco-Manipulation Affordances

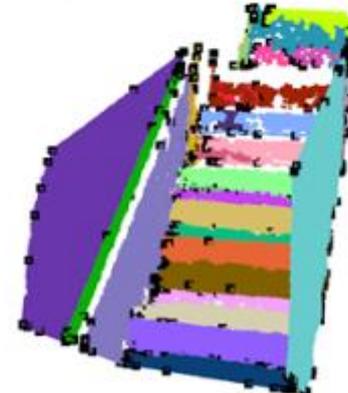
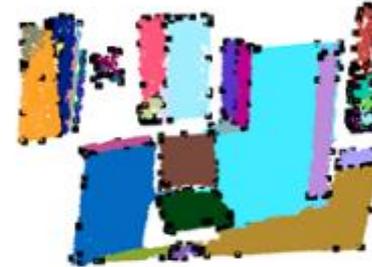
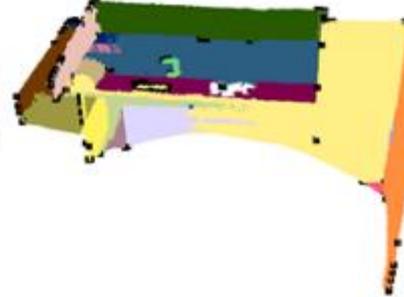
Registered Point Clouds



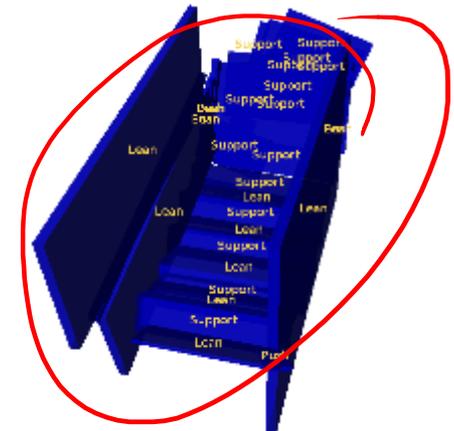
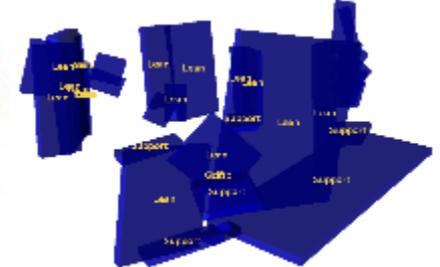
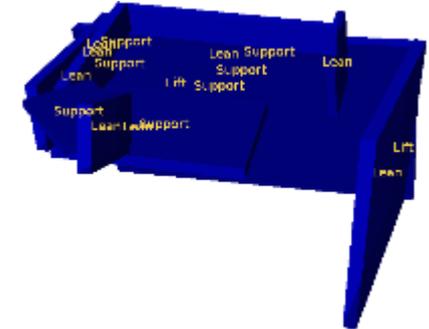
Segmented Point Cloud



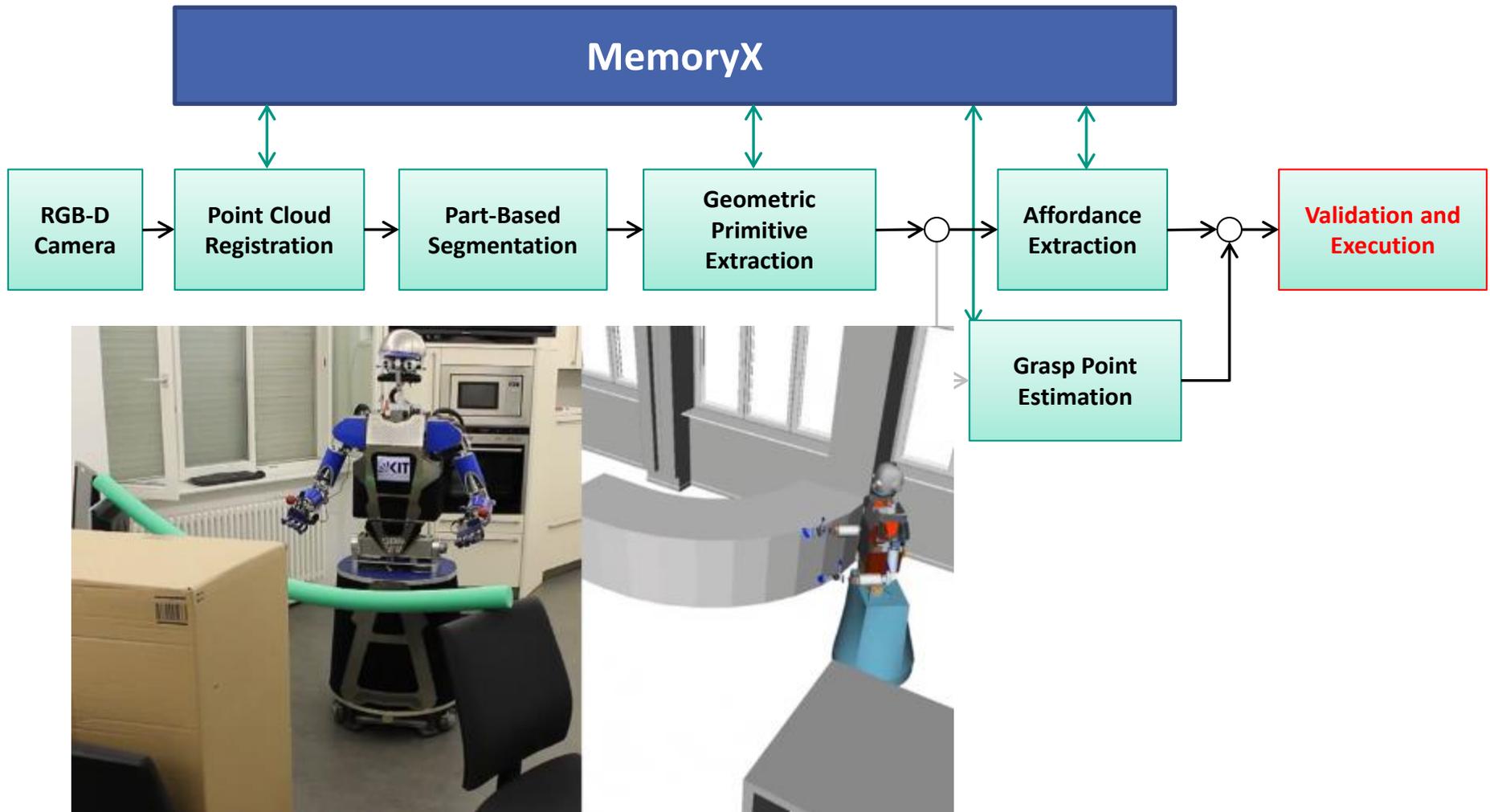
Primitive & Grasp Points



Affordances



Loco-Manipulation Affordances



Loco-Manipulation Affordances



Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability

Peter Kaiser, Markus Grotz, Eren E. Aksoy, Martin Do, Nikolaus Vahrenkamp and Tamim Asfour

Institute for Anthropomatics and Robotics - High Performance Humanoid Technologies Lab (H2T)

KIT – University of the State of Baden-Wuerttemberg and National Laboratory of the Helmholtz Association

www.kit.edu

Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T., “Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability”, In IEEE-RAS International Conference on Humanoid Robots 2015.

Englische Begriffe

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