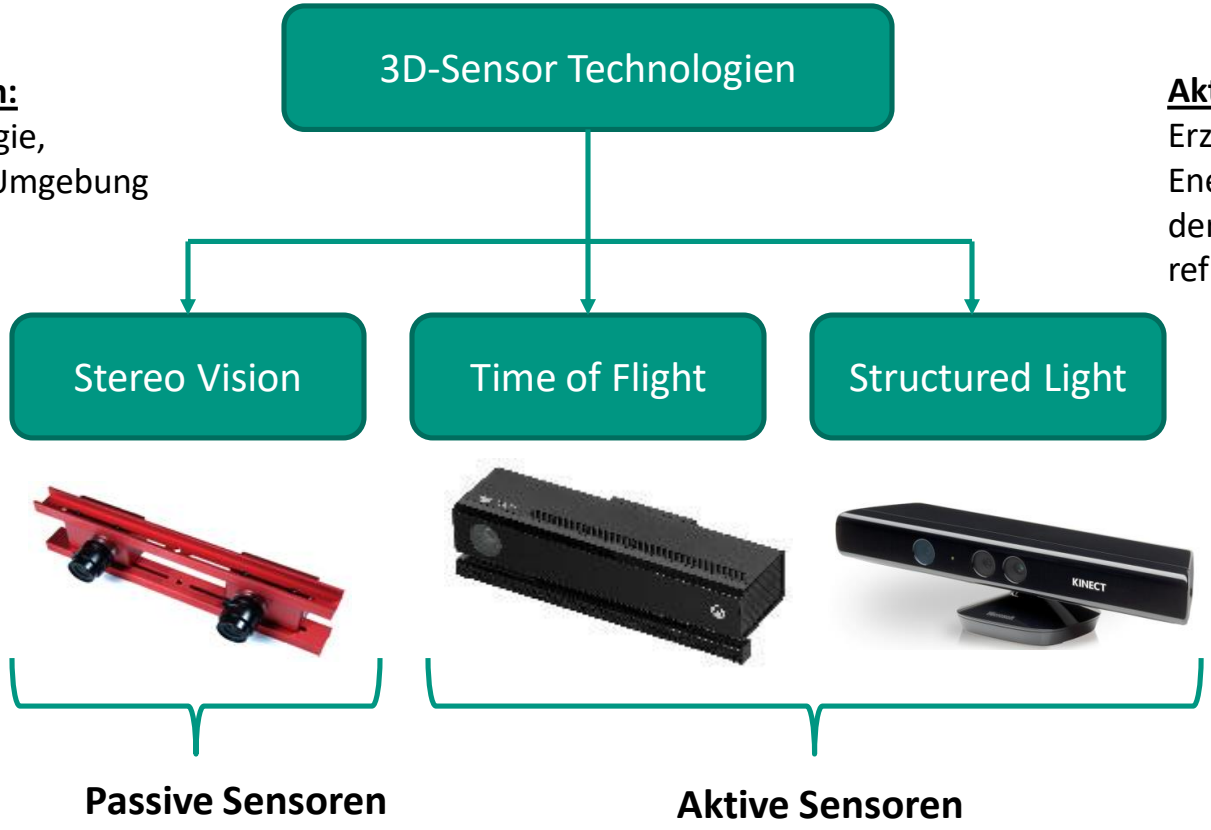
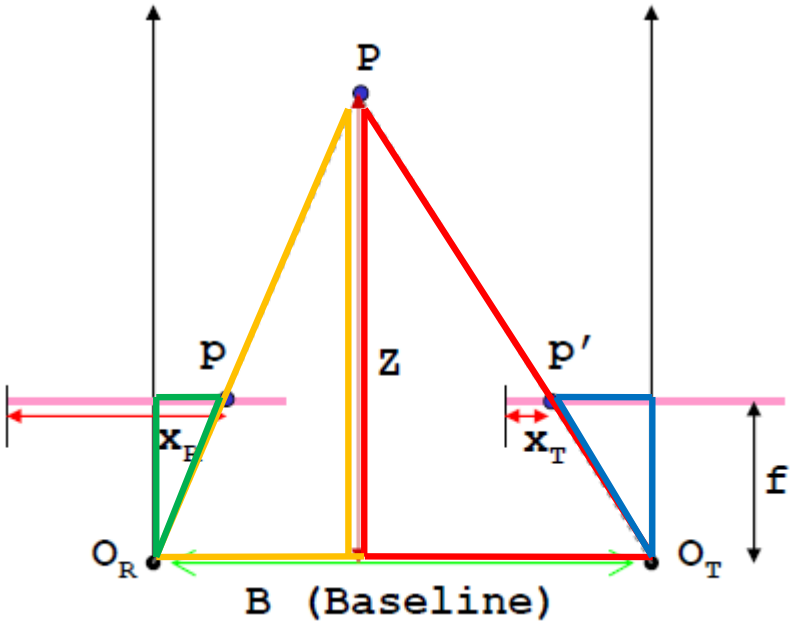
Tiefenkameras

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

Objektdistanz Z

Objektpunkt P

Bildpunkt $p = (x_R, y_R)^T, p' = (x_T, y_T)^T$

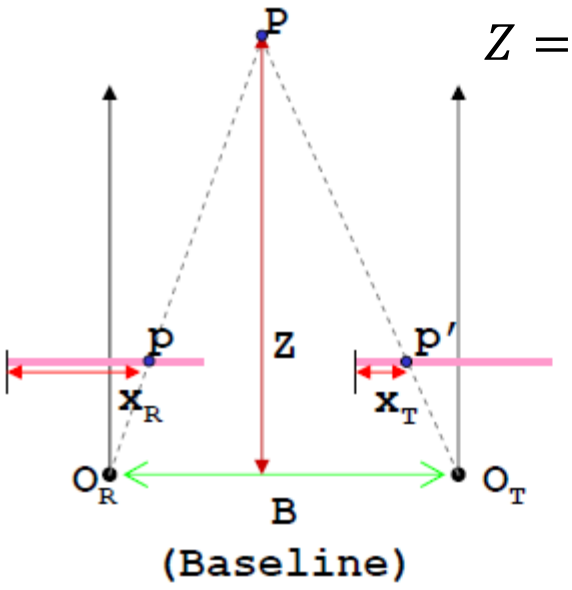
Linke Kamera

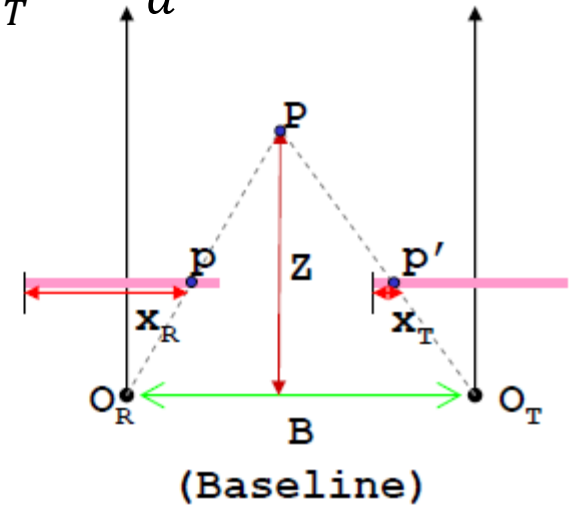
Rechte Kamera

$$Z = \frac{b \cdot f}{x_R - x_T} = \frac{b \cdot f}{d}$$

Stereo Vision II

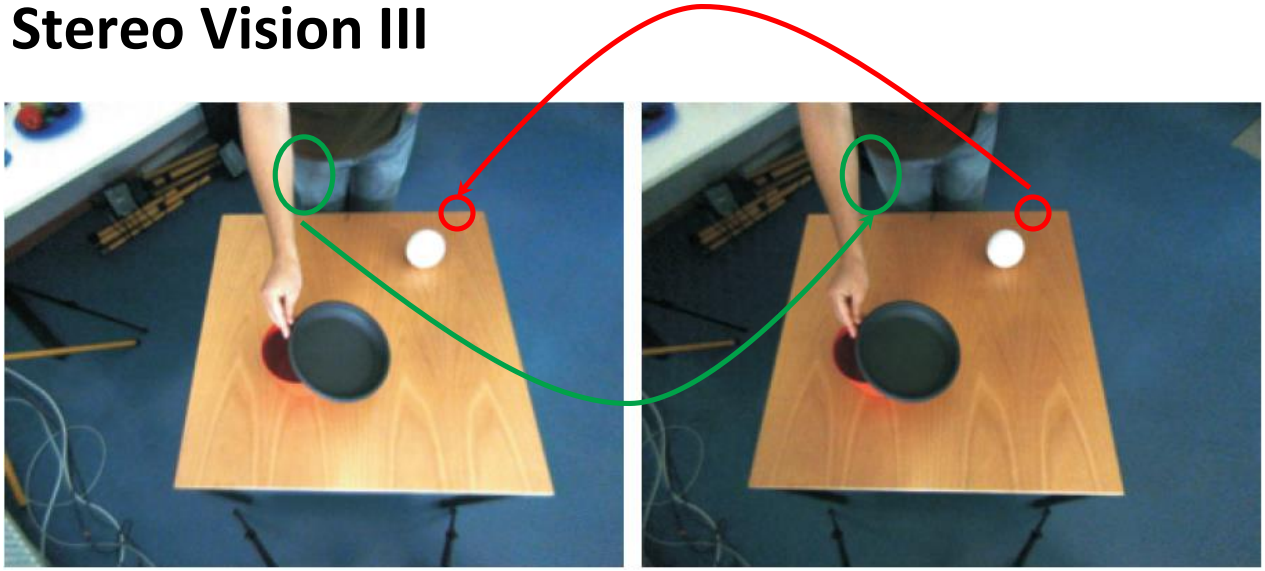
$$Z = \frac{b \cdot f}{x_R - x_T} = \frac{b \cdot f}{d}$$





Disparitäten für **nähere** Objekten haben eine **geringere** Varianz.
Für **entfernte** Objekte **steigt** diese Varianz, was zu **ungenauen** Tiefenmessungen führt.

Stereo Vision III



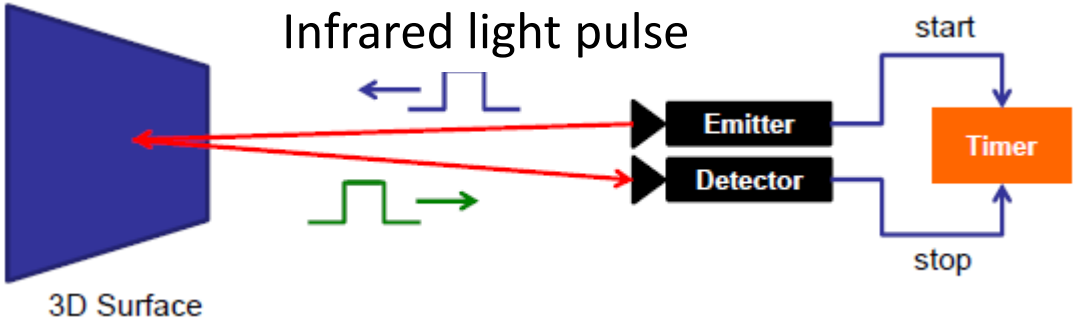
Vorteile

- Einstellbare Brennweiten/Baseline
- Keine explizite Lichtquelle erforderlich

Nachteile

- Mindestens zwei Kameras
- Korrespondezproblem bei homogenen Flächen
- Okklusion

Time of Flight



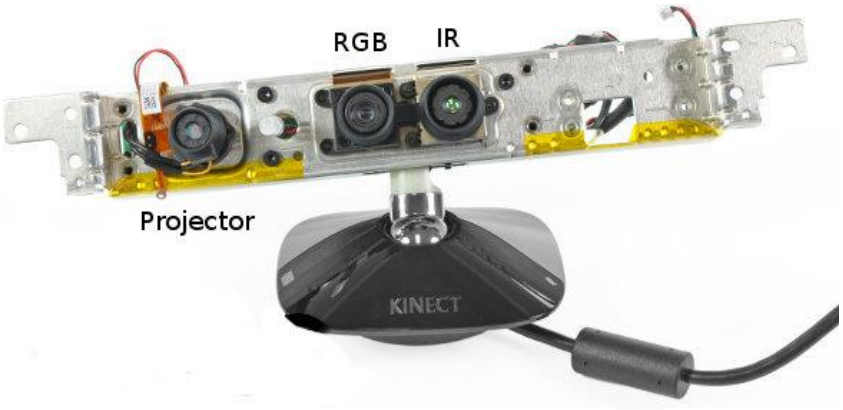
Vorteile

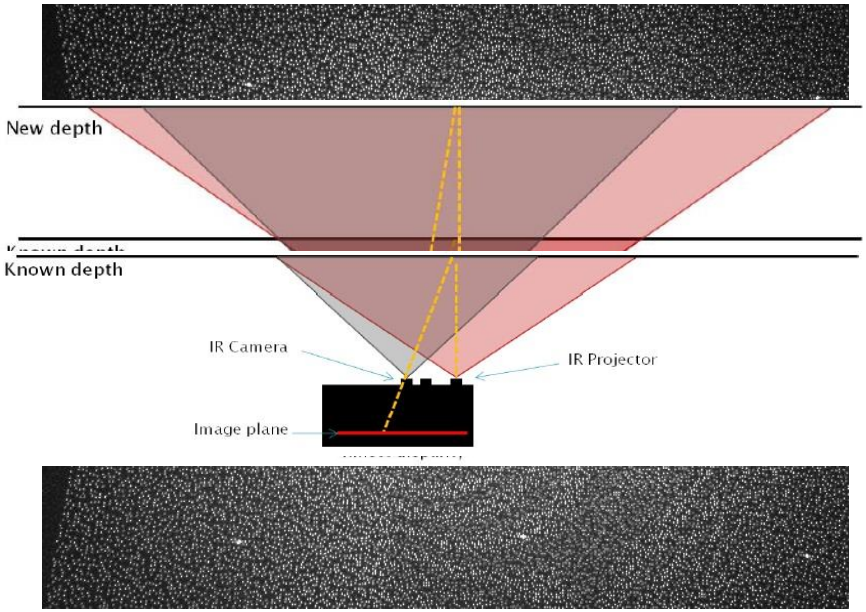
- Keine explizite Tiefenberechnung notwendig
- Hohe Bildrate (bis zu 100 fps)
- Robust gegenüber den meisten Oberflächen

Nachteile

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

Structured Light





Vorteile

- Preiswert
- Kein Korrespondenzproblem auf homogenen Flächen

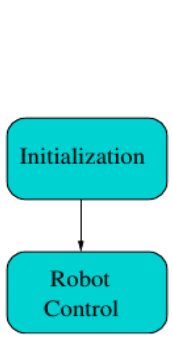
Nachteile

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

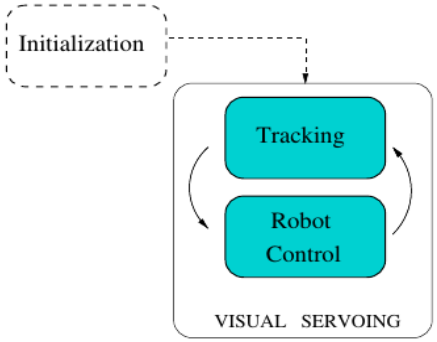
Visual Servoing - Motiviation

- Der Ausdruck **Visual Servoing** beschriebt 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



open-loop Regelung



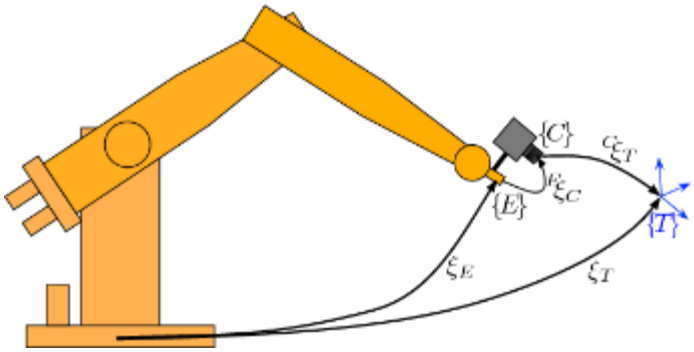
Visual Servoing

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

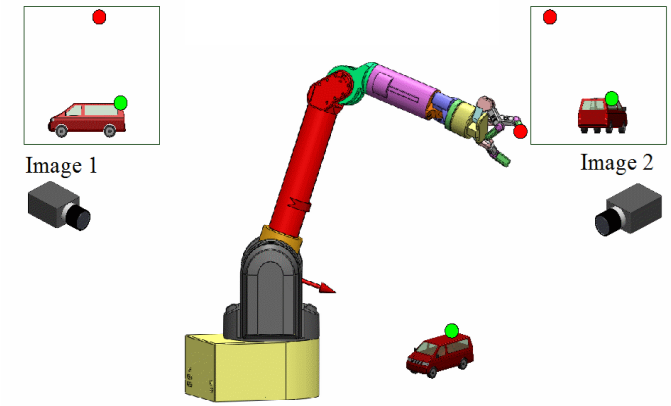
Danica Kragic and Henrik I Christensen, *Survey on Visual Servoing for Manipulation*, Techn. Report, KTH, 2002

Visual Servoing - Systemaufbau

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

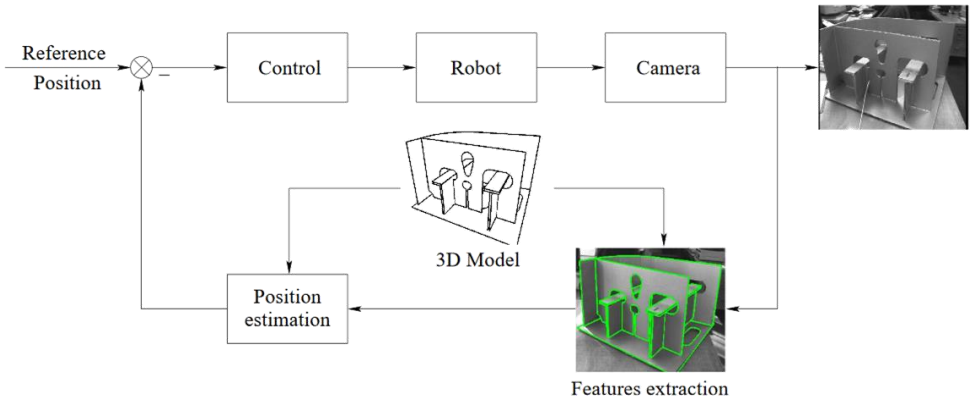


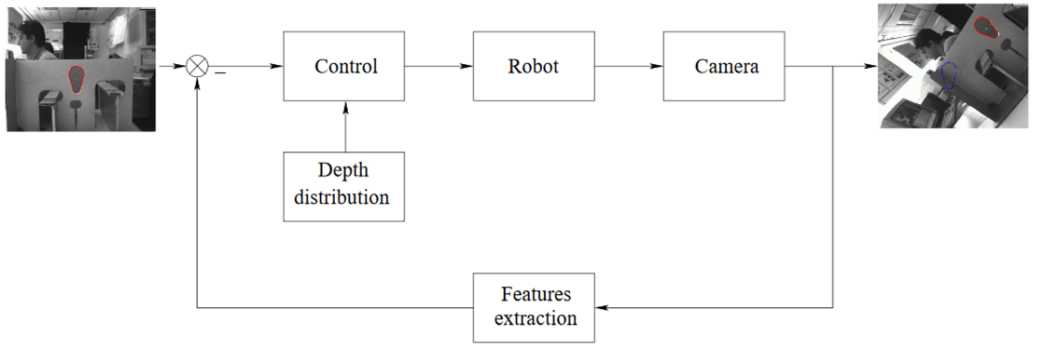
- Externes (festes) Kamera System (eye-to-hand)
  - Externes Kamerasystem wird zur Beobachtung der Bewegung genutzt



Visual Servoing - Verfahren

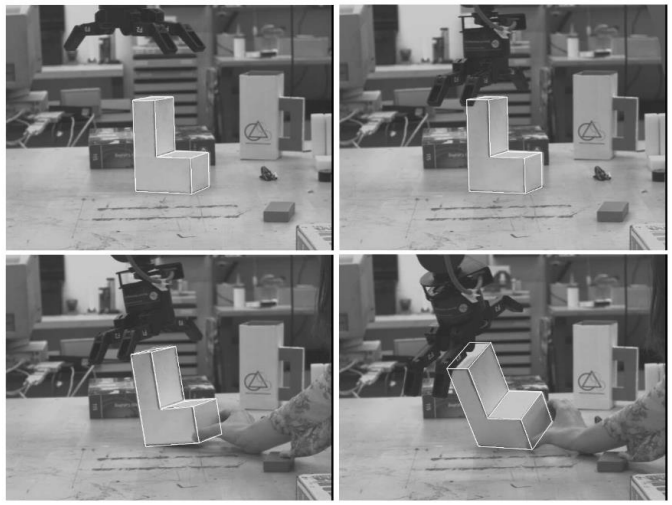
- Positionsbasiertes Visual Servoing
  - Position-based visual servo (PBVS)
  - Einfache Regler
  - 3D Rekonstruktion aufwendig
- Bildbasiertes Visual Servoing
  - Image-based visual servo (IBVS)
  - Merkmalsextraktion einfach
  - Komplexere Regelungsvorschriften
- Hybride Verfahren
  (z.B. 2,5D visual servo)

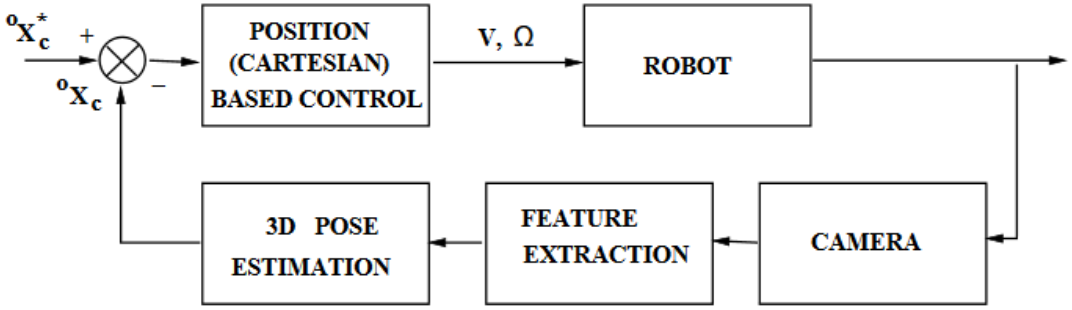




Positionsbasiertes Visual Servoing

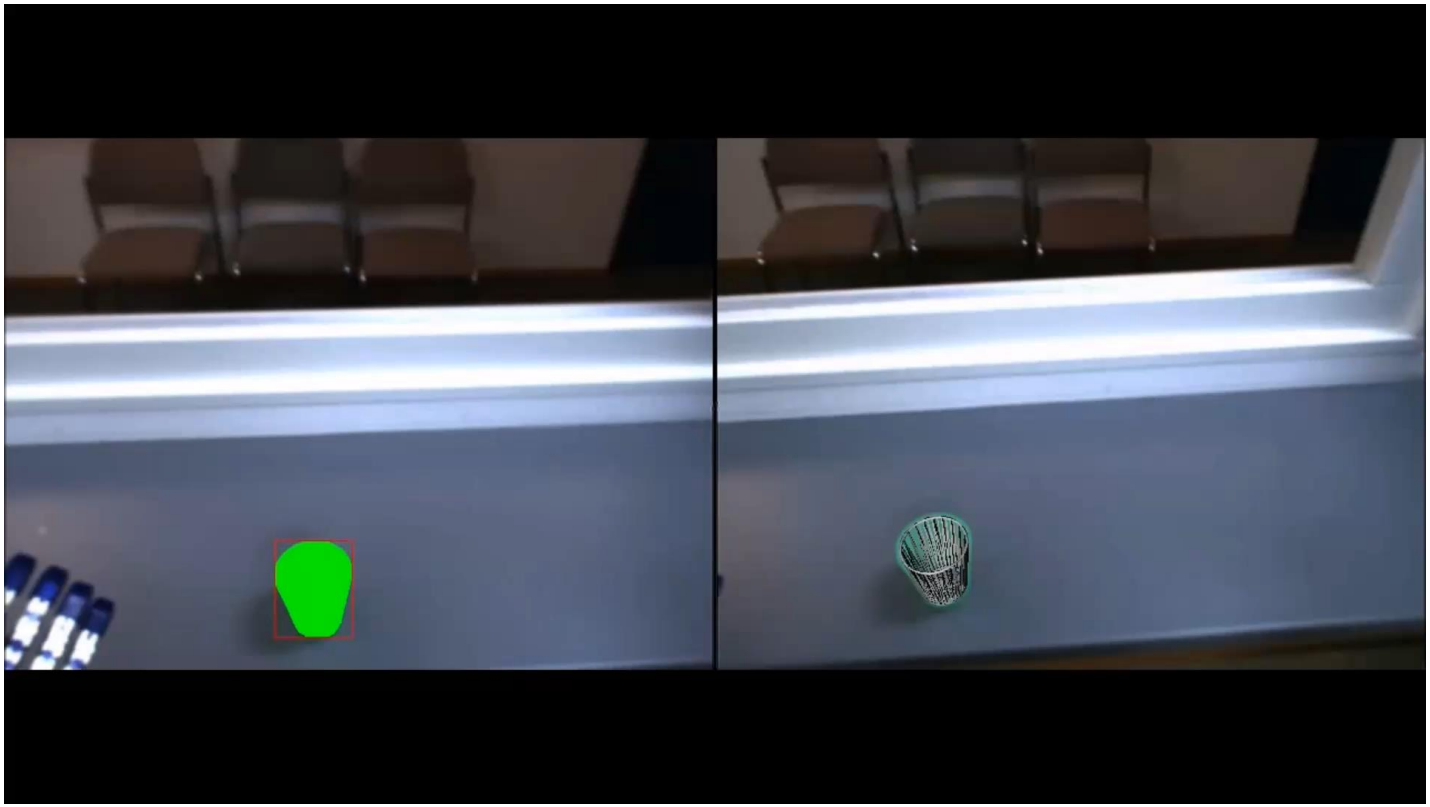
- Zielpose X_g ist vorgegeben
- Ablauf der Regelschleife
  - 3D Lage Schätzung:
    Aktuelle Pose X_c des Zielobjektes (bzw. Hand) wird aus Bildmerkmalen extrahiert
  - Regelvorgabe (kartesisch):
    Differenz $\Delta X = X_g - X_c$
  - Kartesischer Regler führt ΔX aus
  - Ziel erreicht:
    Distanz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





Positionsbasiertes Visual Servoing

- Beispiel auf ARMAR-III

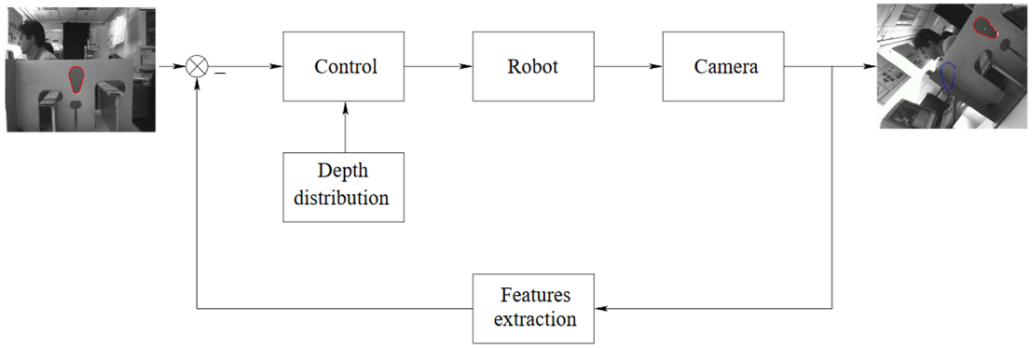


Bildbasiertes Visual Servoing

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

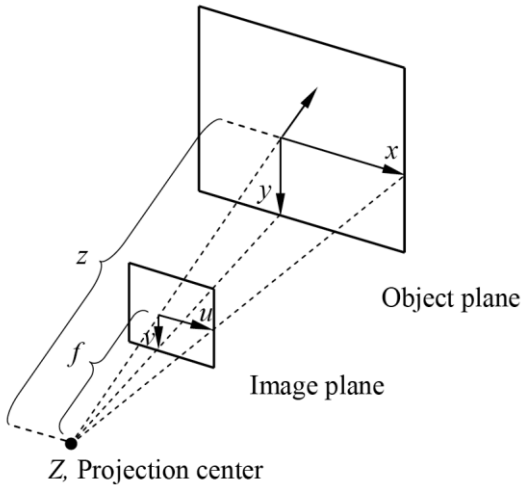
- **Bildmerkmale**
  Bildbverarbeitungsmethoden zur Extraktion von Bildmerkmalen

- **Regelung**
  Geschwindigkeitsvorgaben werden direkt aus der aktuellen und gewünschten Stellung der Bildmerkmale erzeugt



Bildbasiertes Visual Servoing

- **Bildmerkmal**
  Ein Bildmerkmal $s = (u, v)^T$ ist die Projektion eines 3D Punktes $P = (x, y, z)^T$ im Kamerabild.



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

Bildbasiertes Visual Servoing

- **Interaction Matrix / Image Jacobian**
  Die interaction matrix L beschreibt die Beziehung zwischen der Bewegung eines Bildmerkmals $s = (u, v)^T$ und des entsprechenden 3D Punktes $P = (x, y, z)^T$

$$\dot{s} = L\dot{P}$$

- Aus den Projektionsvorschriften (Lochkameramodell) ergibt sich

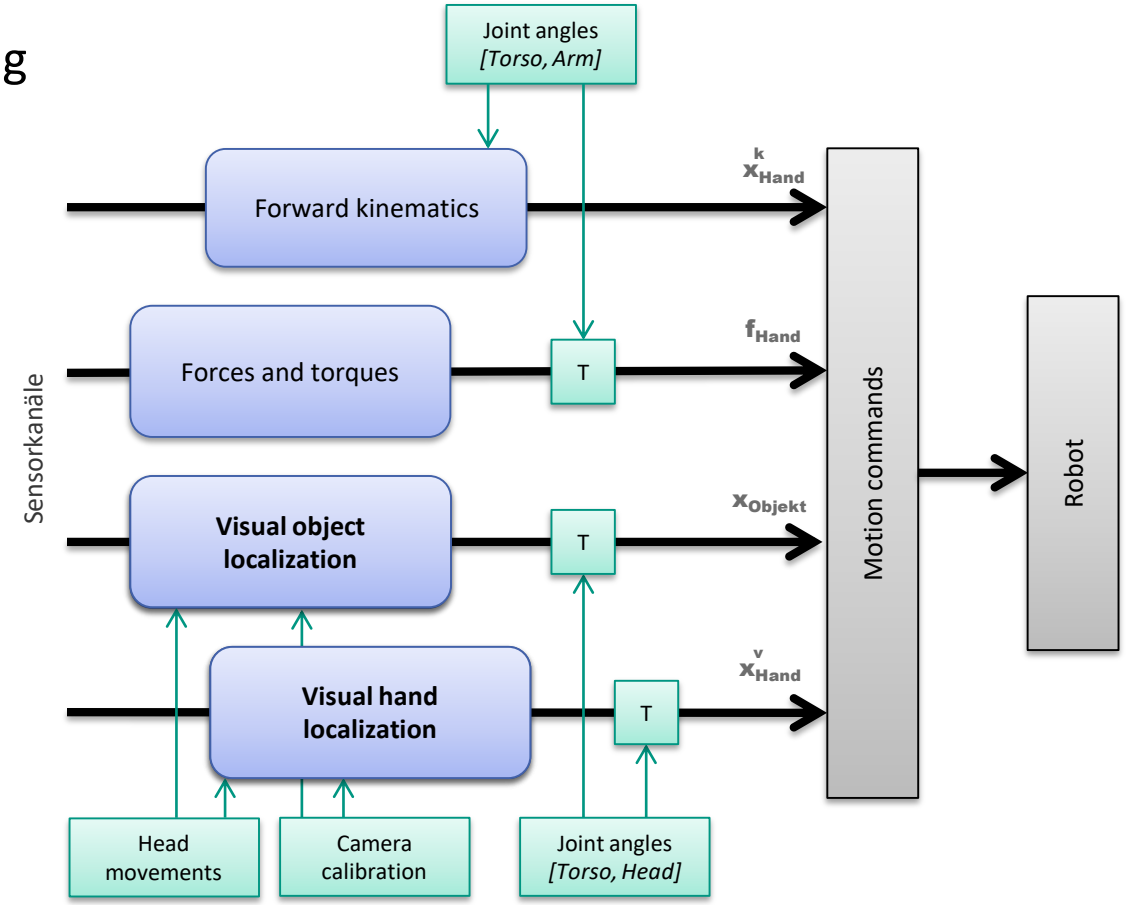
$$L = \begin{pmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{pmatrix}$$

Bildbasiertes Visual Servoing

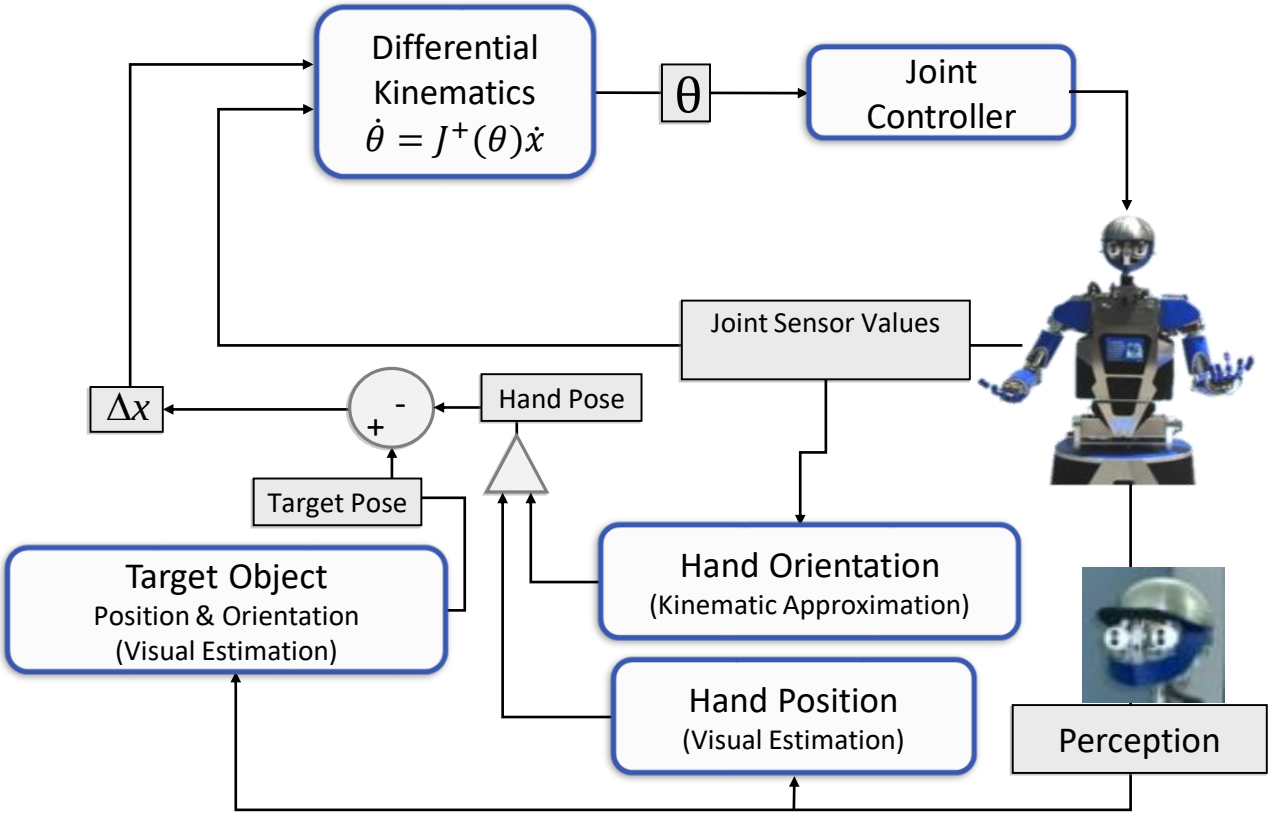
- **Invertierung der Interaction Matrix**
  - Distanz z wird geschätzt
  - Mehrere Merkmale werden beobachtet (mind. 3)
  - Annahme: Kamera-In-Hand System
    Bewegung der 3D Punkte korrespondiert mit Bewegung der Kamera
    (Geschwindigkeit der Kamera v_C)
  - Daraus folgt:
    $$\dot{e} = L v_C$$
  - Mit $\dot{e} = -\lambda e$ ergibt sich
    $$v_C = -\lambda L^+ e$$
  - Hierbei ist L^+ die Pseudoinverse von L

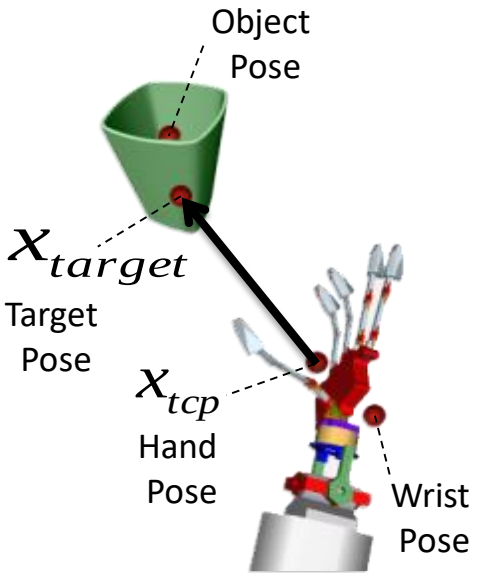
Visual Servoing für ARMAR-III

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



Positionsbasiertes Visual Servoing auf ARMAR-III





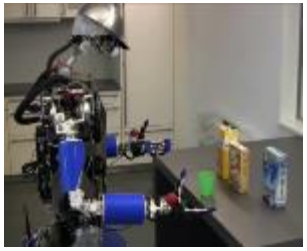
$$\dot{\theta} = J^{+}(\theta)\dot{x}$$

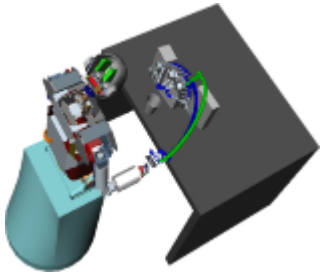
$$\partial^{t} = x^{t}_{vision} - x^{t}_{kinematic}$$

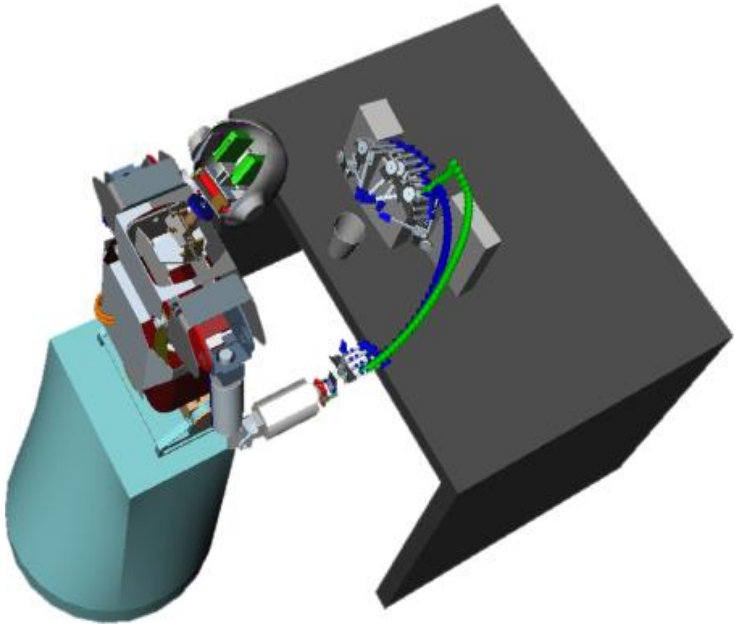
$$x^{t+1}_{tcp} = x^{t+1}_{kinematic} + \partial^{t}_{tcp}$$

Ausführung von Manipulationsaufgaben

Grasping Task → Scene Representation → Motion Planning → **Sensor-based Execution**





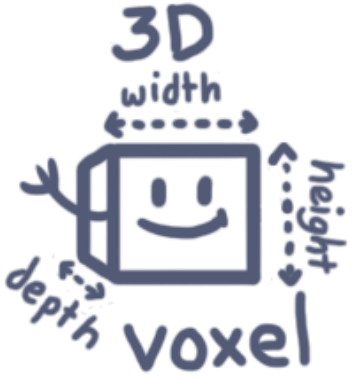


Point Clouds

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

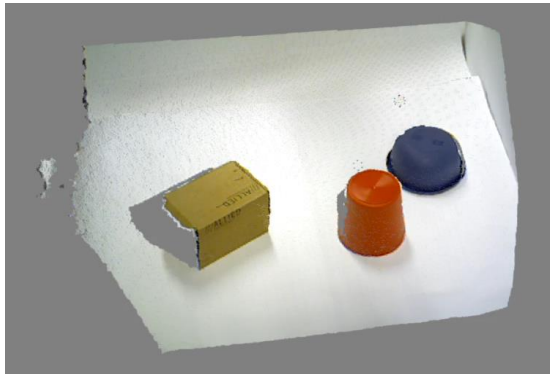
Pixel = Picture Elements

VS.



Voxel = Volumetric Pixel

2D Bild



3D Punktwolke

Point Clouds

- Punktwolke $P = \{(X, C) \big| X \in \mathbb{R}^3, C \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$
  - $X = (x, y, z)$ Ortsinformation
  - $C = (r, g, b)$ Farbinformation
  - Es können weitere (Sensor-)informationen abgespeichert werden

- Zwei verschiedene Arten der Repräsentation

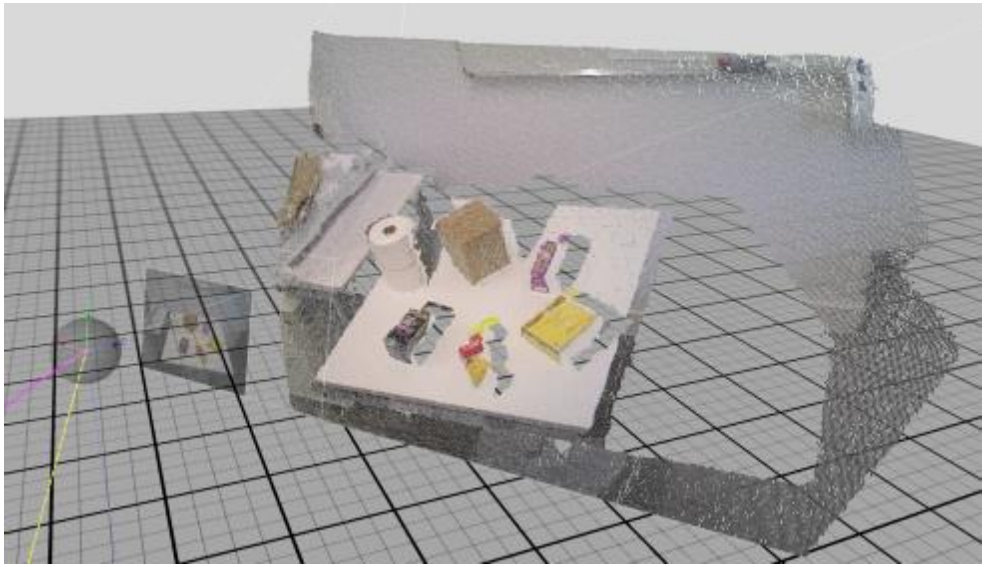
  - **Organisierte** Punktwolke (2D Array Format, Größe muss vorab bekannt sein)
  - **Unorganisierte** Punktwolke (Vector Format)

Normalenschätzung in 3D Punktwolken

Ziel:

- Zusätzliche Oberflächeninformation durch Einbeziehung von lokalen Nachbarpunkten

- Grundlage für weiterführende Algorithmen
  - Segmentierung
  - Deskriptoren
  - Objekterkennung
  - Oberflächen-modellierung



Normalenschätzung in 3D Punktwolken

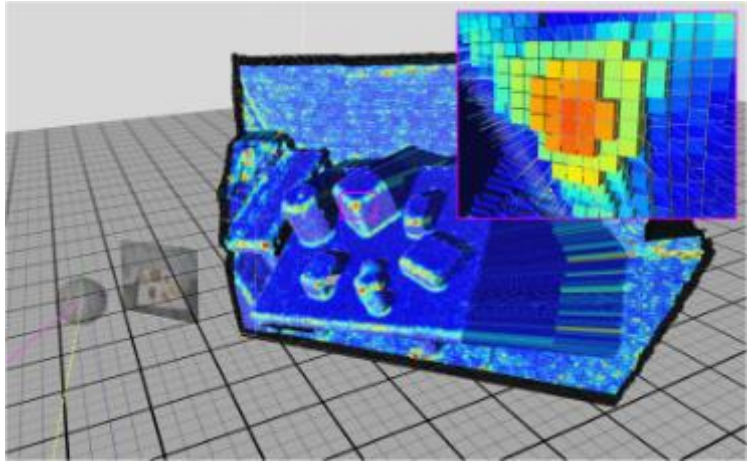
- PCA-basierter Ansatz
- Erstelle die Kovarianzmatrix C der k-Punktnachbarschaft für jeden Punkt p

$$C = \frac{1}{k}\sum_1^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T$$

p_i k Nachbarpunkte

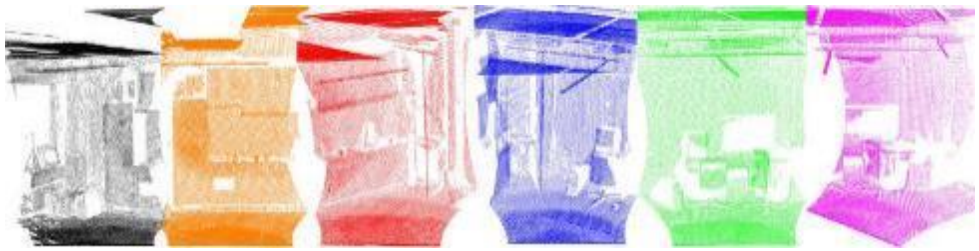
$\bar{p}$ Mittelpunkt aller k Nachbarn

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

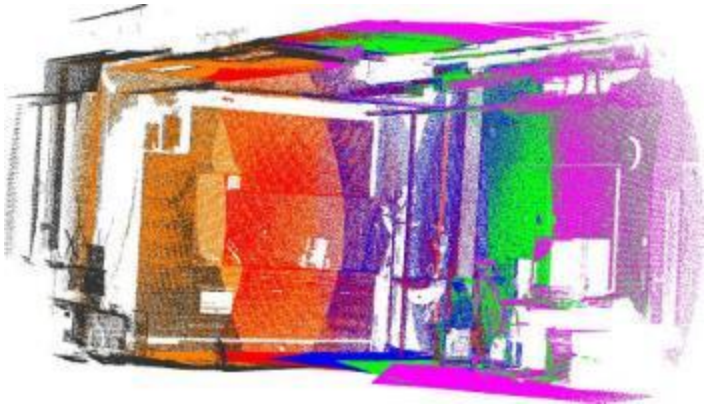


Registrierung von Punktwolken

- **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



Punktwolken eines Raumes aus unterschiedlichen Perspektiven



Registrierte Punktwolke

Iterative Closest Point

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

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

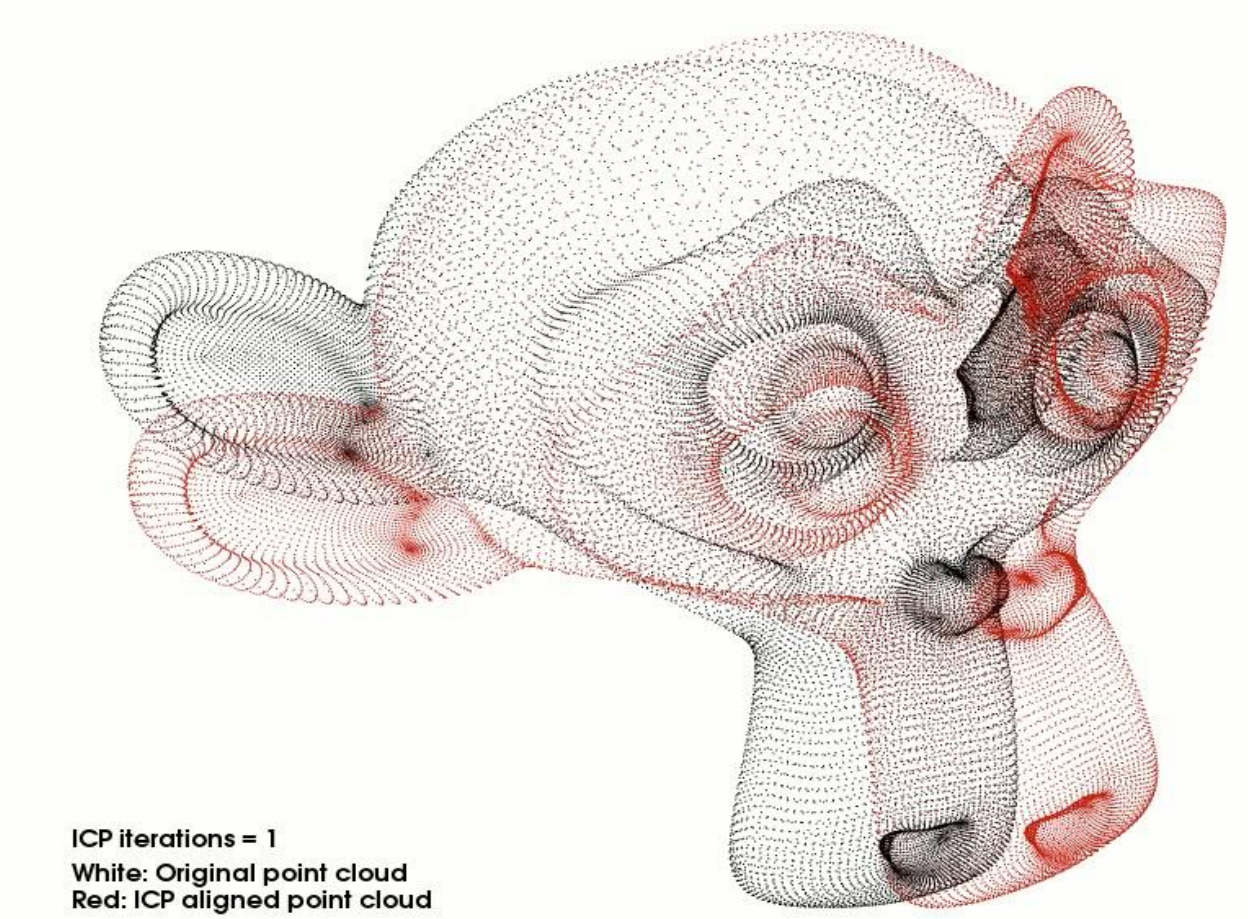
$$D_k = \sum_i \|a_i - T_k \cdot b_i\|^2$$

  - D_k Kombiniert Translation und Rotation
  - Wende Transformation T_k auf alle Punkte aus B an (Update)
  - 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



RANSAC (Random Sample Consensus)

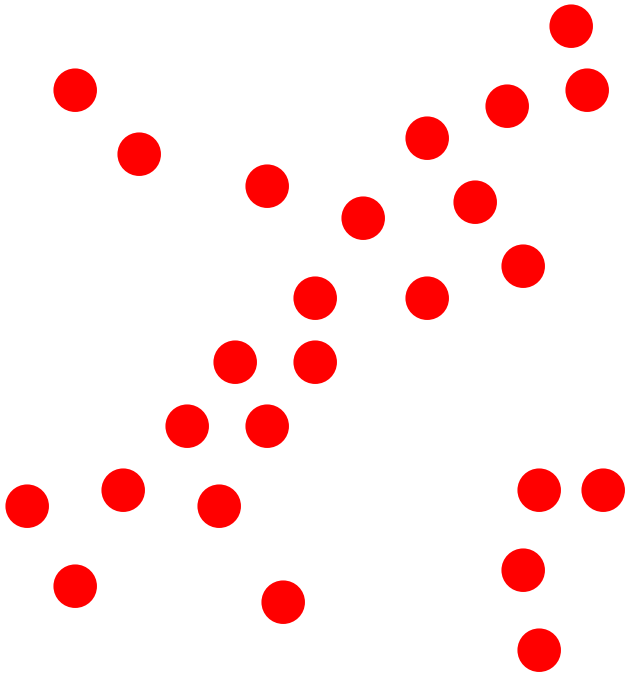
- **RANSAC** ist eine **iterative Methode** zur Schätzung von Modellparametern aus Datenpunkten
- RANSAC ist ein **nicht-deterministischer** Algorithmus
- Robust gegenüber Ausreißern und fehlenden Datenpunkten
- Anwendung in der Bildverarbeitung
  - Schätzung von Linien in 2D Bildern
  - Schätzung von 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)

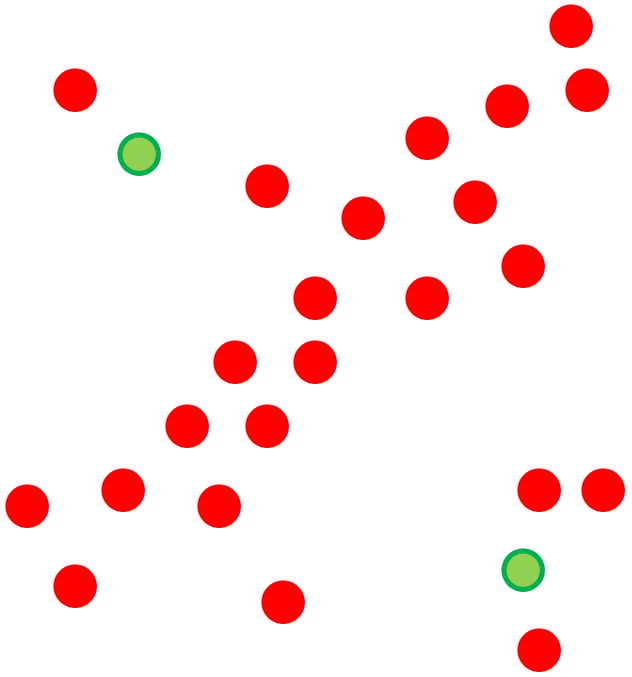
- Beispiel:
  - Line fitting in 2D Datenpunkte



- 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)

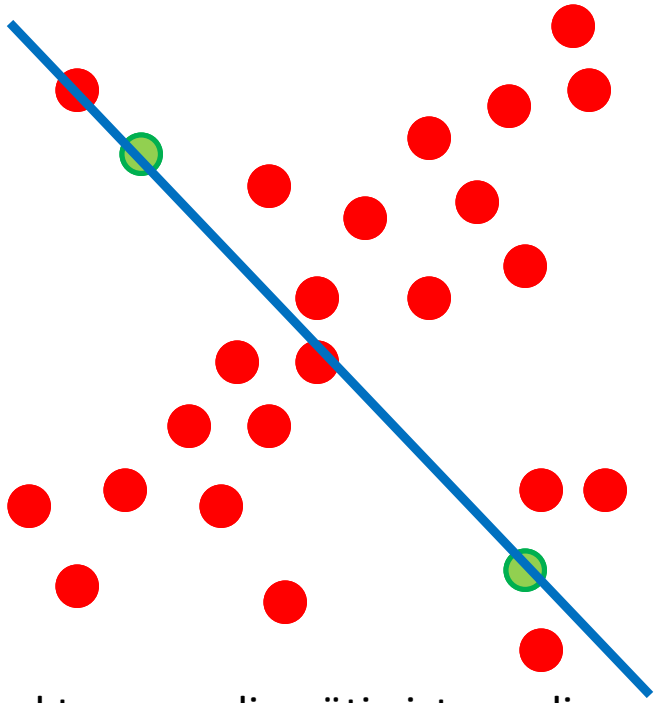
- Beispiel:
  - Line fitting in 2D Datenpunkte



- 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)

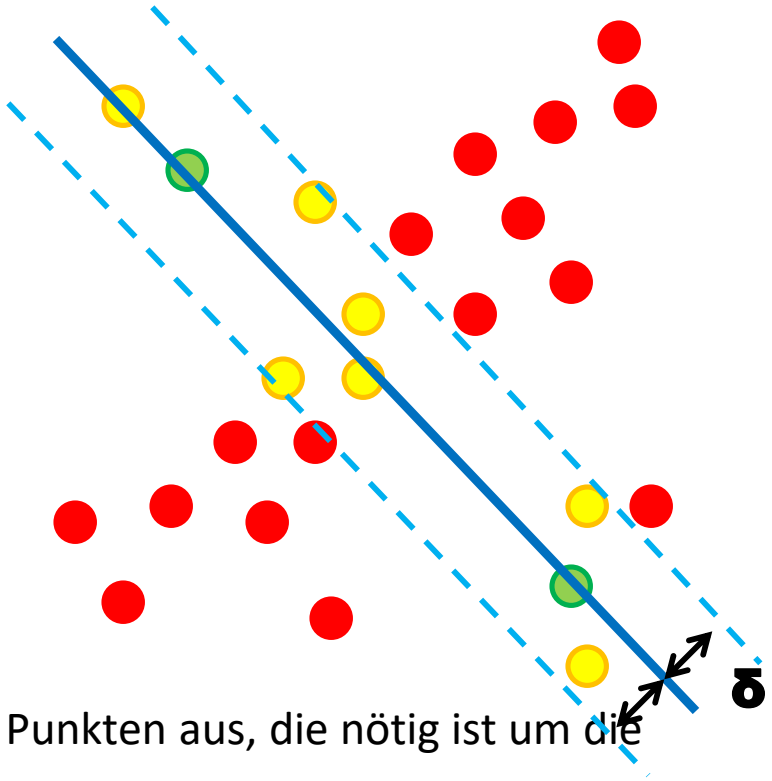
- Beispiel:
  - Line fitting in 2D Datenpunkte



- 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)

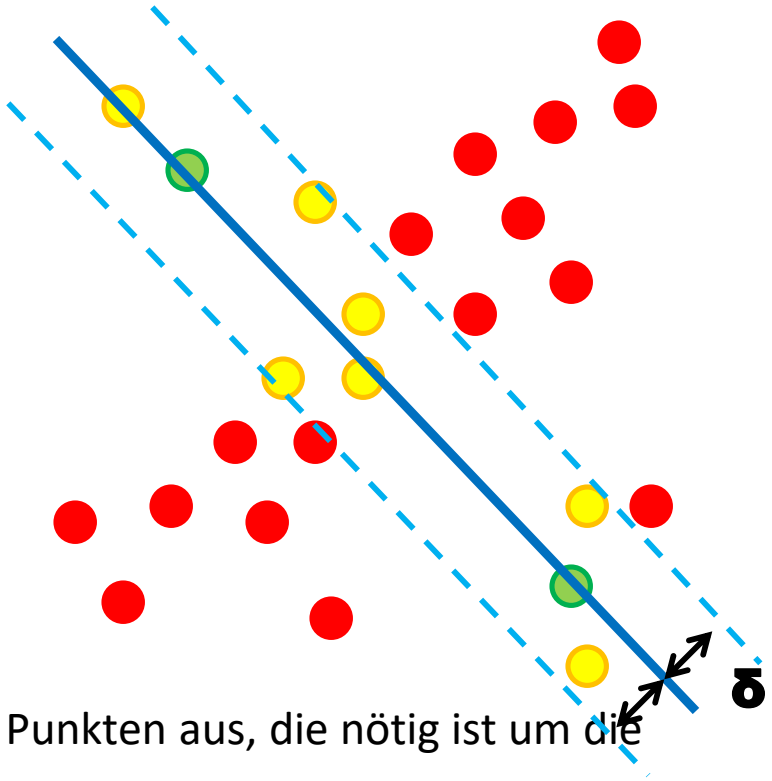
- Beispiel:
  - Line fitting in 2D Datenpunkte



- 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



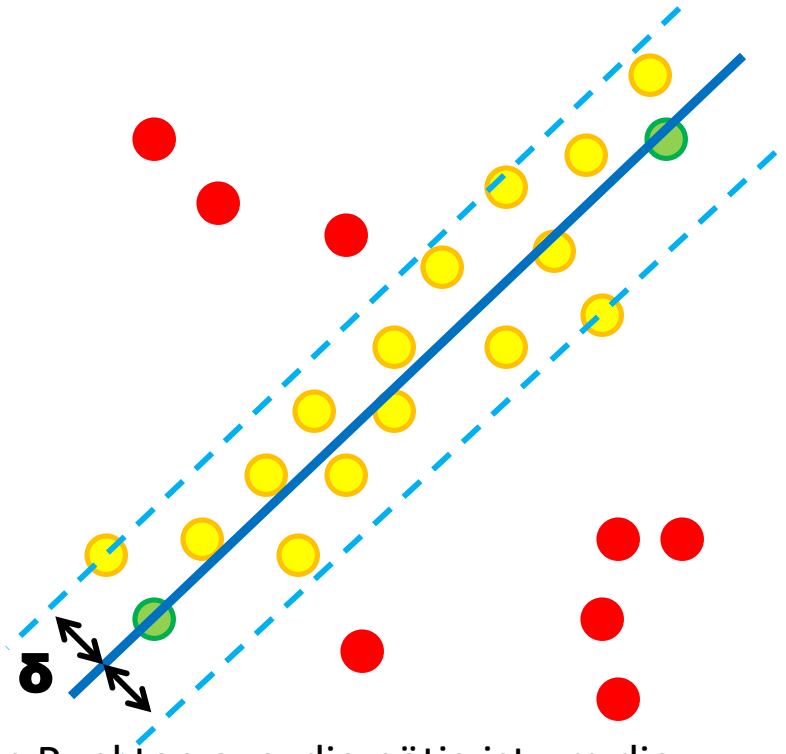
Inlier Anzahl = 9

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**

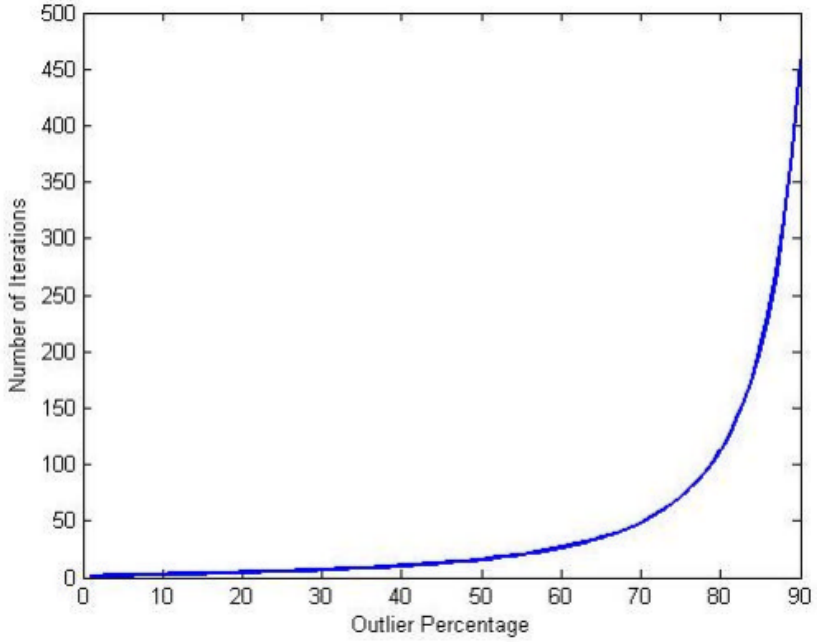
RANSAC (Random Sample Consensus)

Die voraussichtliche **Anzahl an Iterationen** k damit der Algorithmus mit der **Wahrscheinlichkeit** P erfolgreich terminiert kann wie folgt berechnet werden:

- Wahrscheinlichkeit für einen Inlier: $P(inlier) \equiv w$ (w = Anzahl der Inlier/Gesamtzahl der Samples)
- Wahrscheinlichkeit ein Inlier-Modell zu ziehen: $P(subset\ with\ no\ outlier) \equiv w^n$
- Wahrscheinlichkeit ein Outlier-Modell zu ziehen:
  $P(subset\ with\ outliers) \equiv 1 - w^n$
- Wahrscheinlichkeit in k Iterationen ein Outlier-Modell zu ziehen:
  $P(k\ subsets\ with\ outliers) \equiv (1 - w^n)^k$
- Wahrscheinlichkeit für einen erfolglosen Durchlauf : $P(fail) \equiv (1 - w^n)^k$
- Wahrscheinlichkeit für einen erfolgreichen Durchlauf: $P(success) \equiv 1 - (1 - w^n)^k$
- Voraussichtliche Anzahl an Iterationen : $\Rightarrow k = \frac{\log(1-P(success))}{\log(1-w^n)}$

RANSAC (Random Sample Consensus)

- RANSAC benötigt mehr Iterationen im Fall von vielen Outlieren!



- Voraussichtliche Anzahl an Iterationen: $\Rightarrow k = \frac{\log(1-P(success))}{\log(1-w^n)}$
- Minimale Sample Anzahl n muss vorher definiert sein!
- $P(success)$ muss hoch angesetzt werden (d.h. $\geq 95\%$)

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

Simultaneous Localization and Mapping (SLAM)

- Das SLAM Problem:

  Wie kann ein Roboter in einer unbekannten 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 unbekannten 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

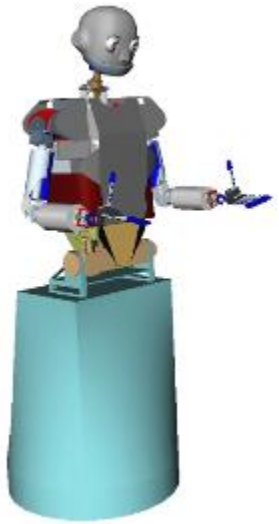
  - SLAM wählt Szenenfeatures als **Orientierungspunkte**

  - Visuelle Features: Punkte, Ecken, Kanten, Texturen, Oberflächen

  - Hinweis: Features müssen unterscheidbar und deskriptiv sein
    (Invariant gegenüber Standpunktwechsel)

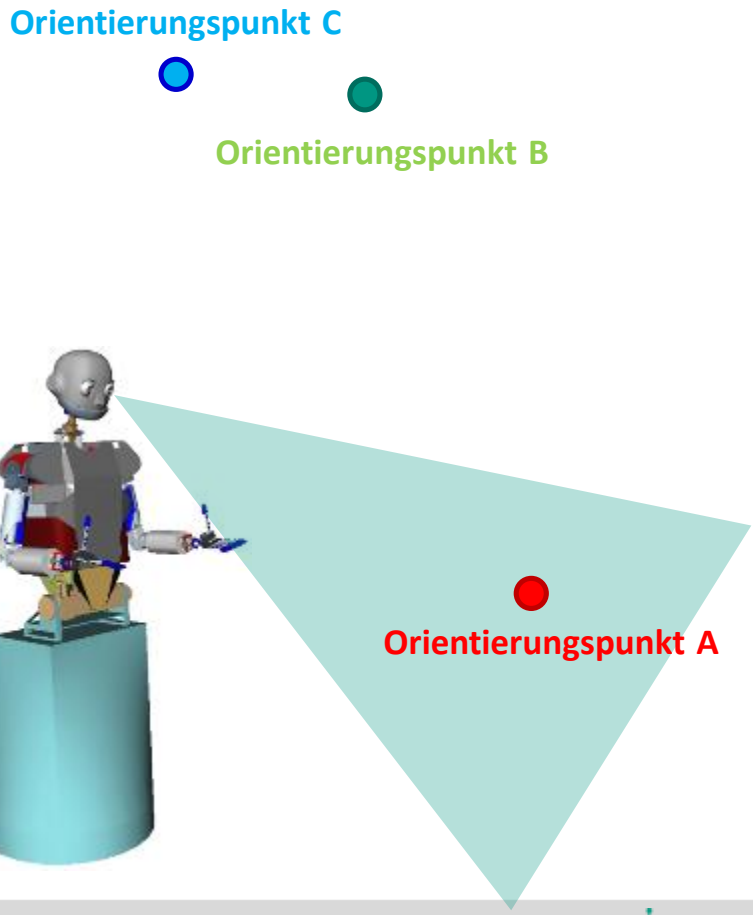
Simultaneous Localization and Mapping (SLAM)

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



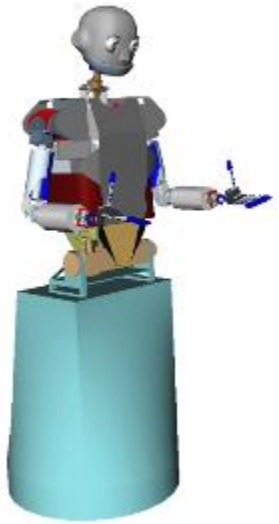
Simultaneous Localization and Mapping (SLAM)

- 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



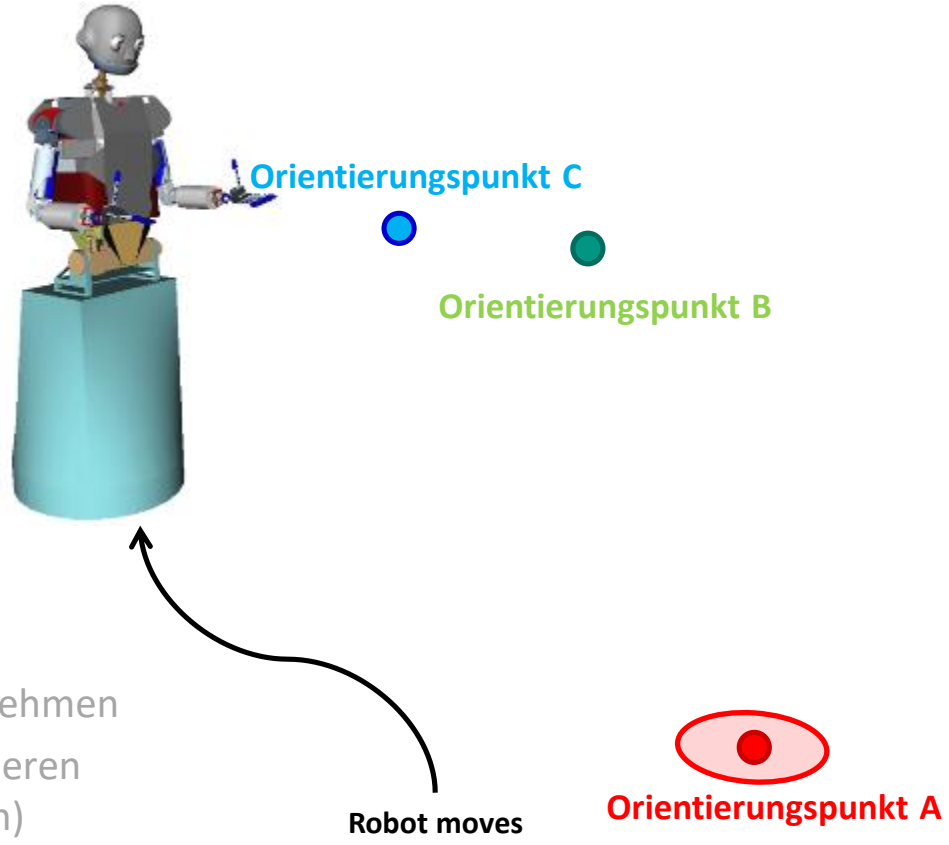
Simultaneous Localization and Mapping (SLAM)

- 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)



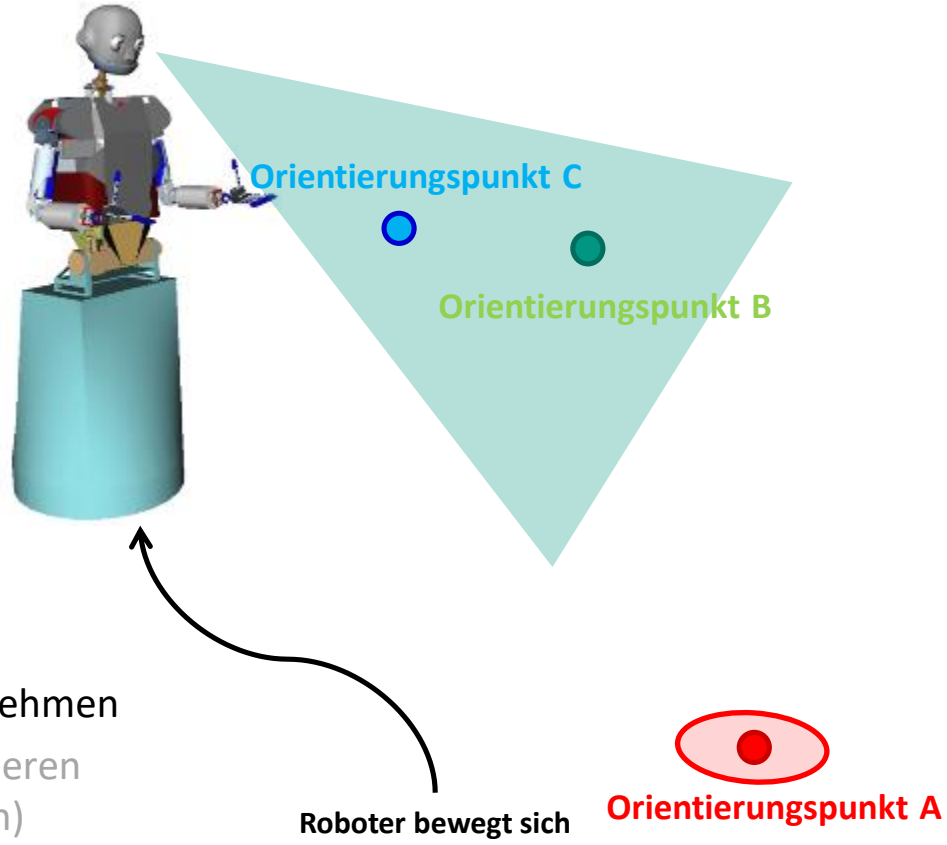
Simultaneous Localization and Mapping (SLAM)

- 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)



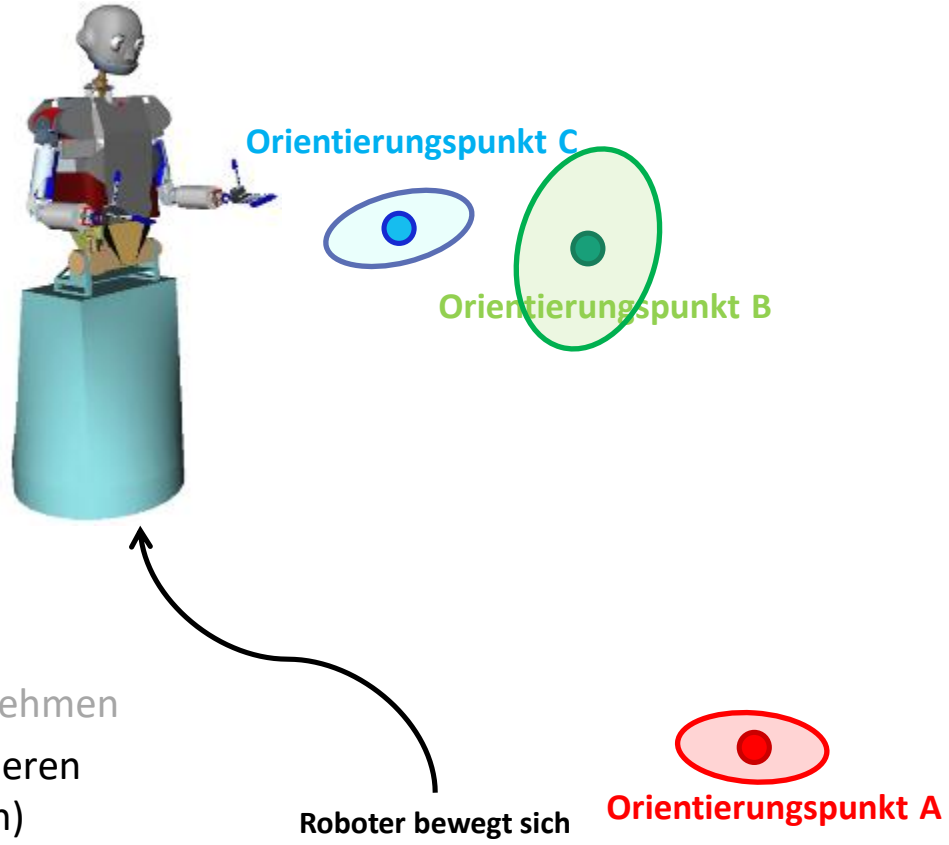
Simultaneous Localization and Mapping (SLAM)

- 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)



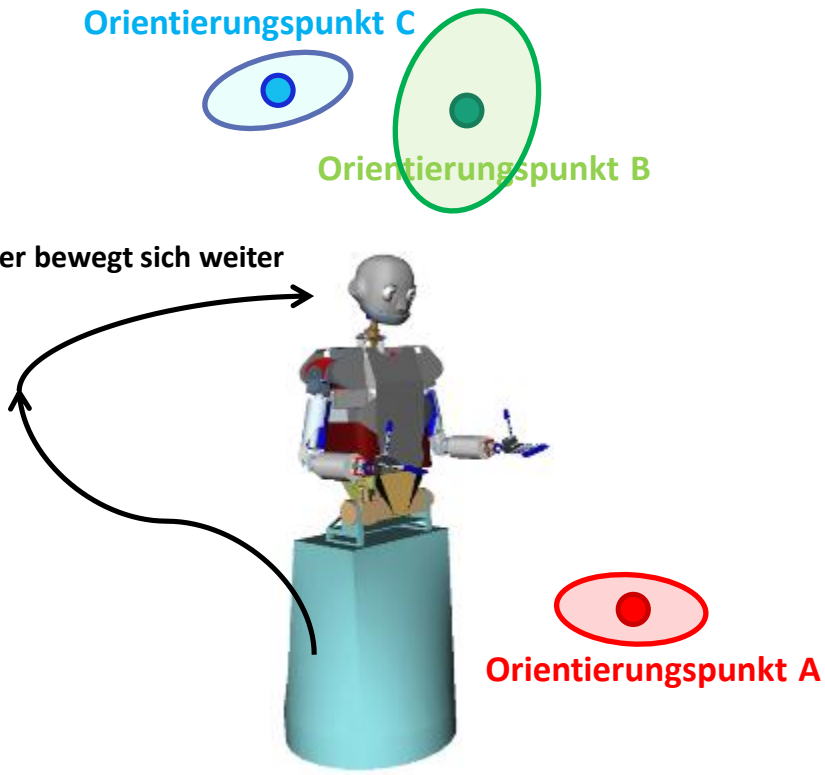
Simultaneous Localization and Mapping (SLAM)

- 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)

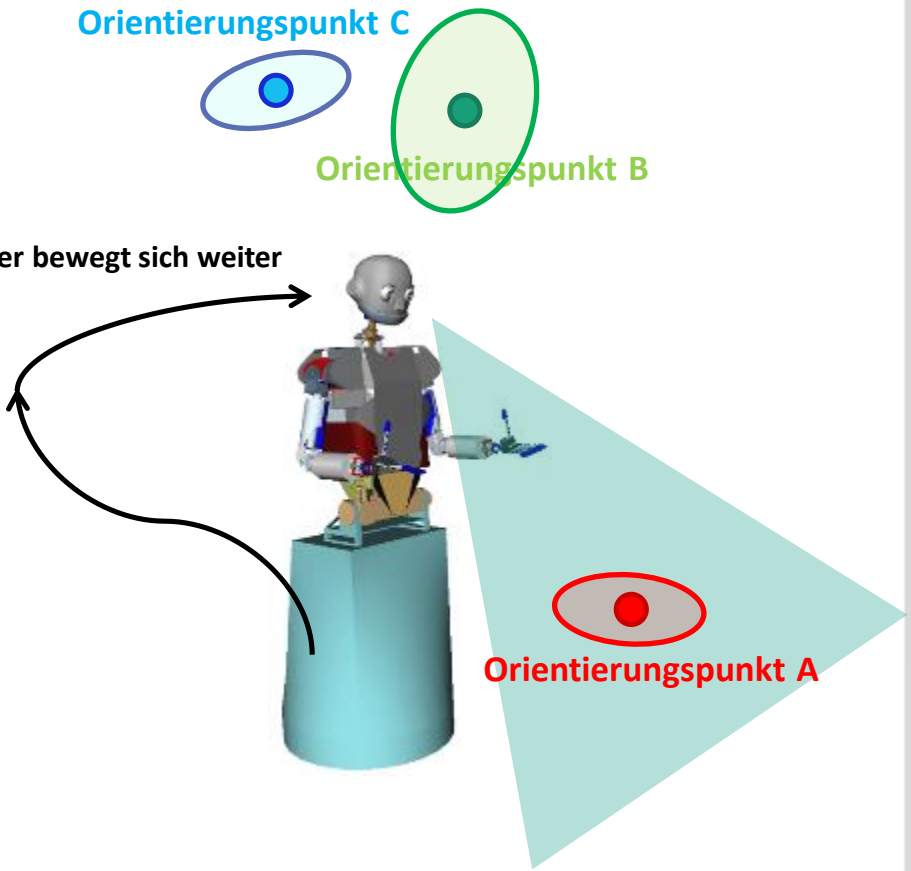
- Wie funktioniert SLAM?
  - Interne Repräsentation für:
    - Position der Orientierungspunkte
    - Parameter der Kamera
  - In jedem Frame:
    - Vorhersage, wie viel sich der Roboter bewegt hat
    - Neue Orientierungspunkte aufnehmen
    - Interne Repräsentation aktualisieren (Messungenauigkeiten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



COMPUTER VISION @H2T

Beispiel: Perceptual Pipeline

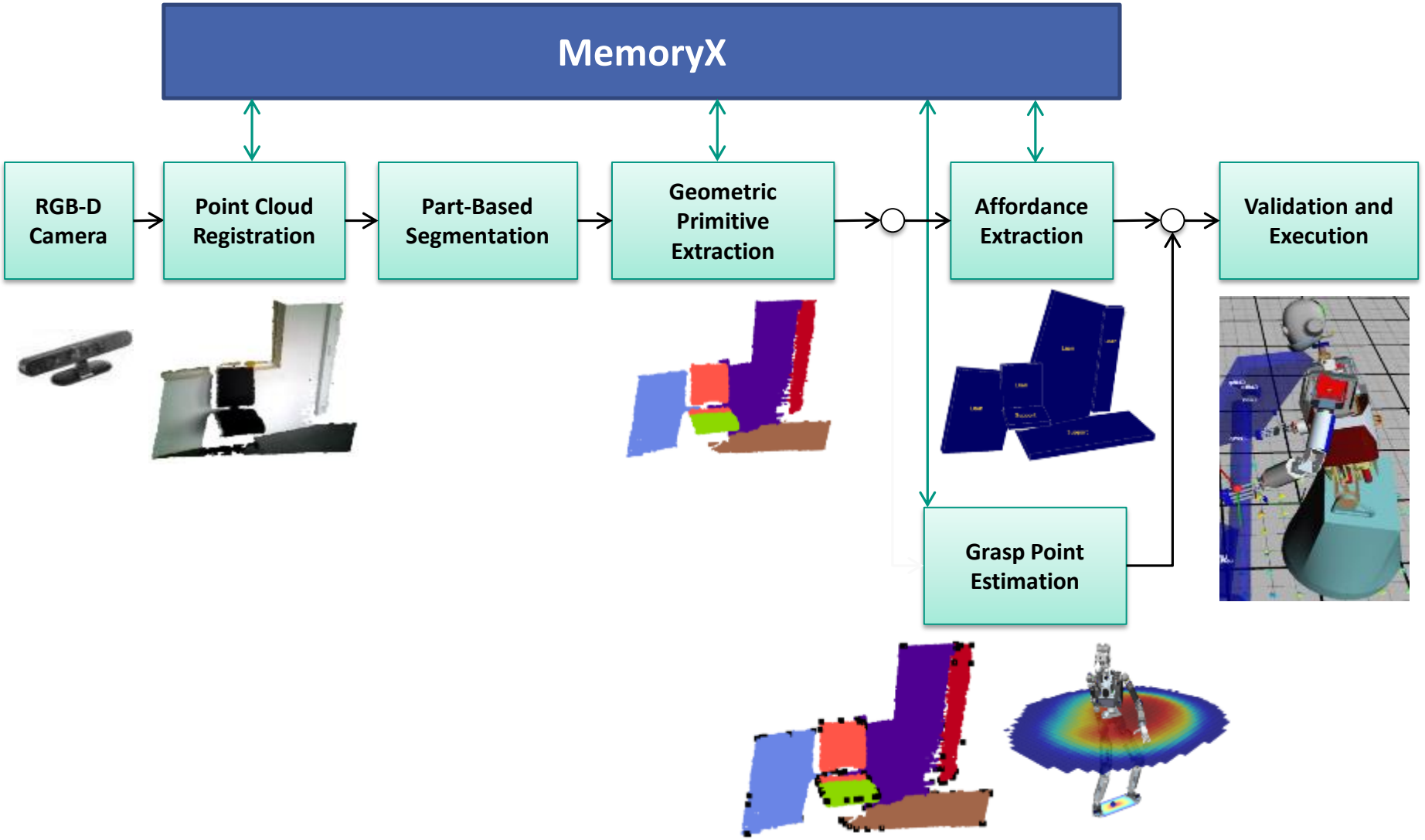
- SLAM
- RANSAC

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

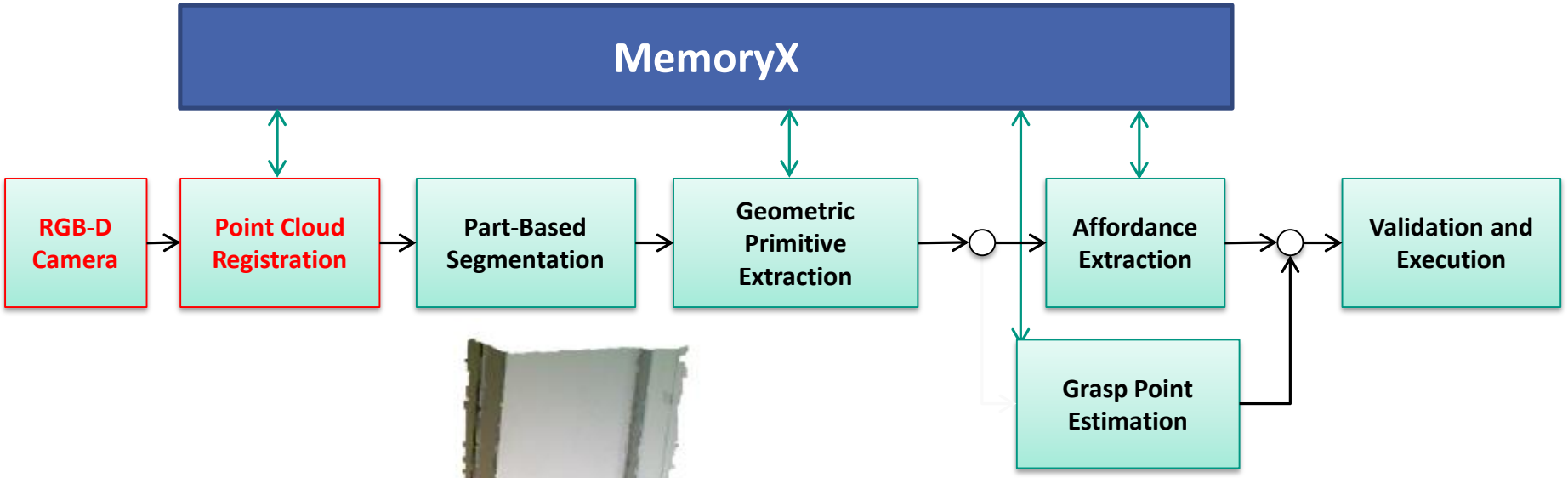
Loco-Manipulation Affordances

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



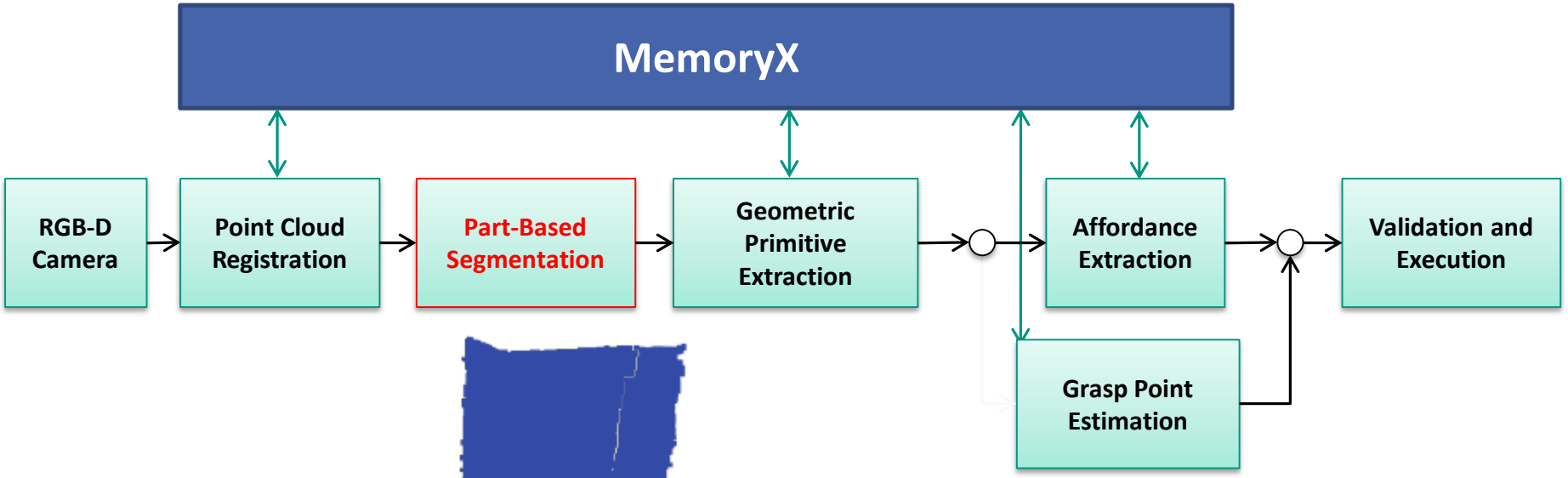
Loco-Manipulation Affordances



- 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.

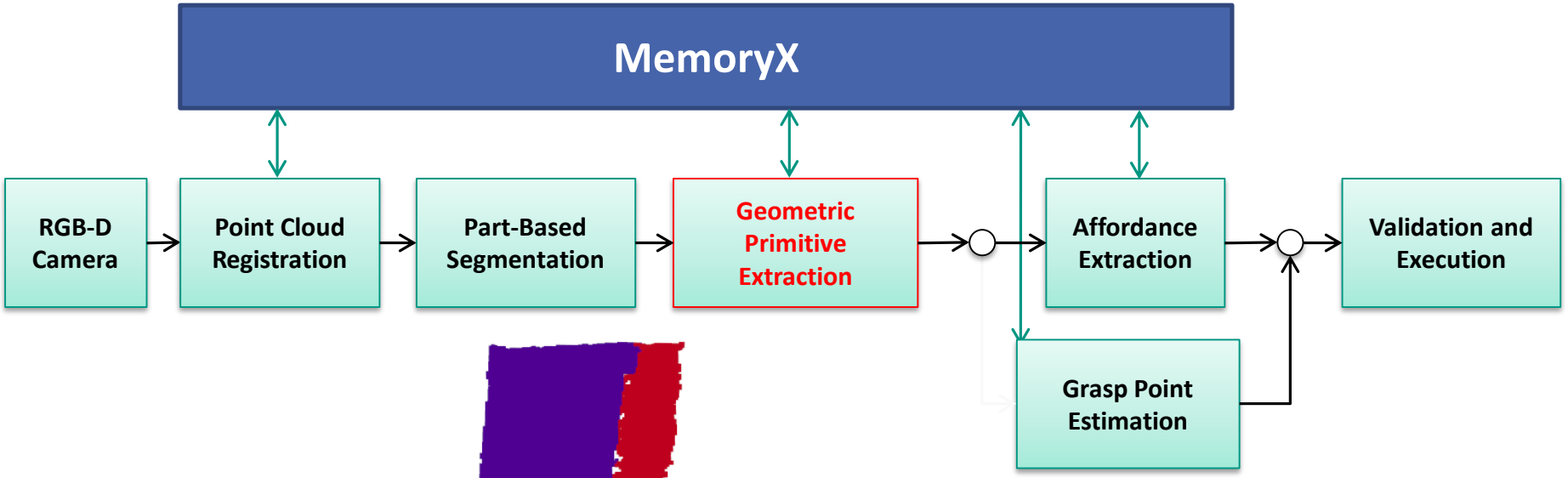
Loco-Manipulation Affordances



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

Segmentation: S. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius, “Convexity based object partitioning for robot applications,” in International Conference on Robotics and Automation ICRA, June 2014.

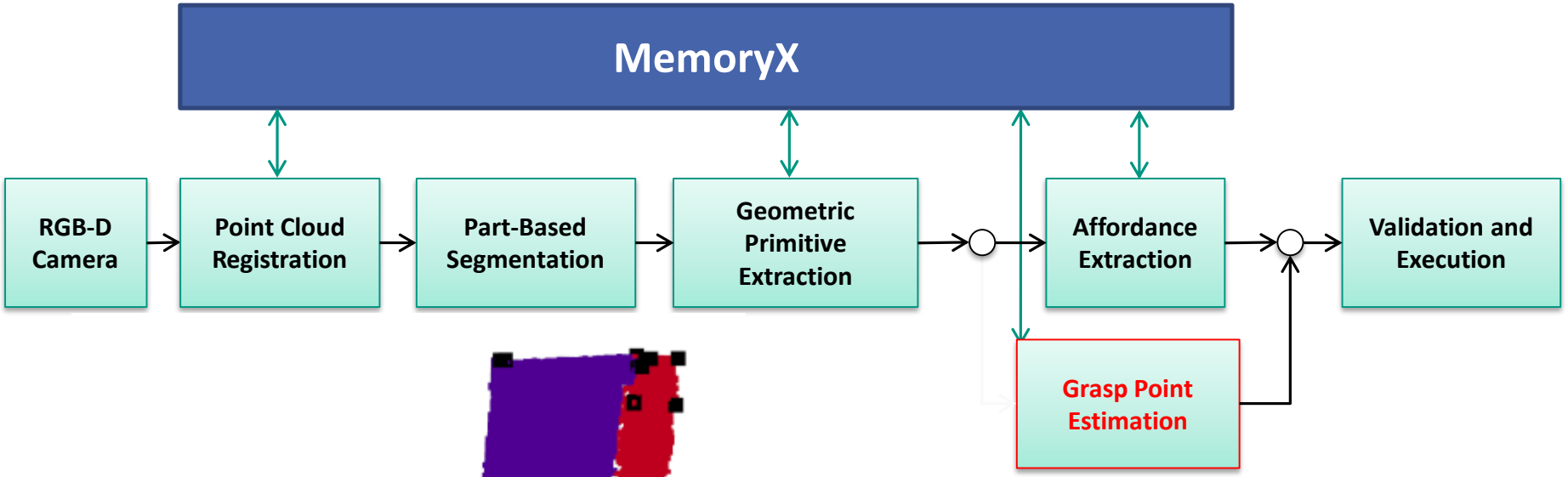
Loco-Manipulation Affordances

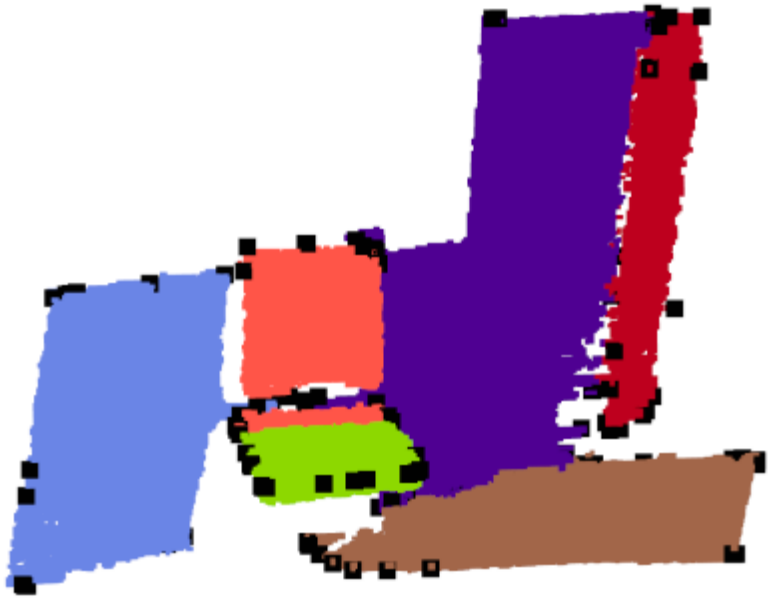


- Fitting von geometrischen Modellen (z.B. Ebenen und Zylinder) an Segmente mit RANSAC
- Wegen Untersegmentierung werden Modelle iterativ an die verbleibenden Outliers gefittet (z.B. Stuhl)

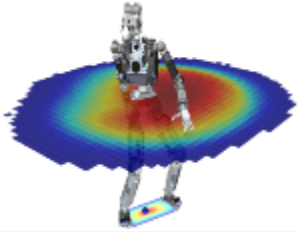
PCL: R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in International Conference on Robotics and Automation, 2011.

Loco-Manipulation Affordances

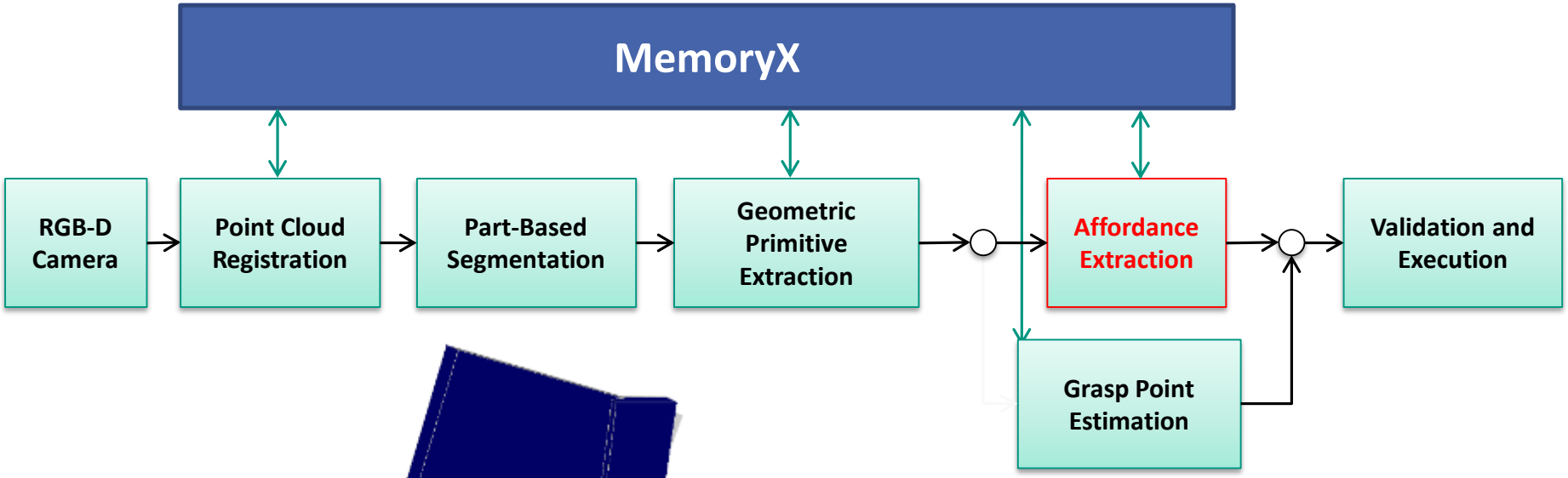


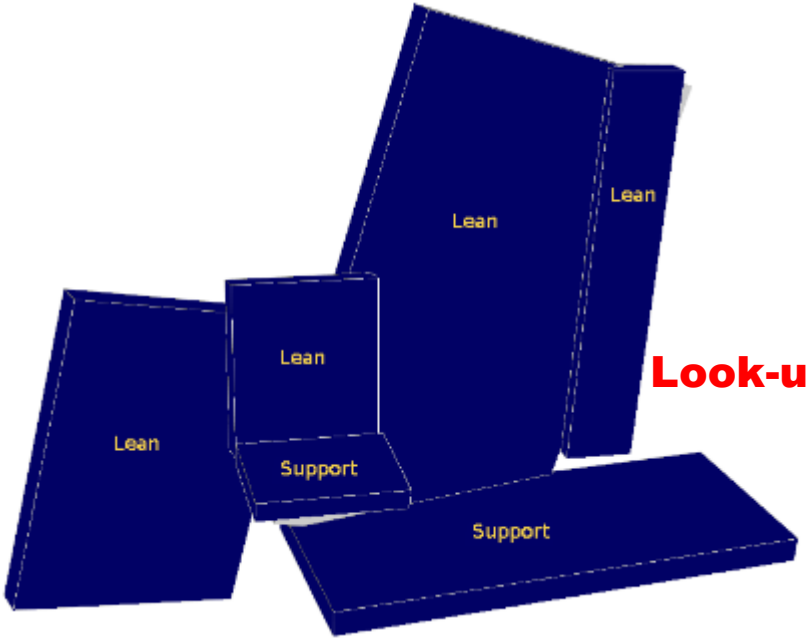


- Berechne konvexe Hülle für die projizierten Inlier Punkte um *Grasp Points* zu schätzen. Vgl. Schwarzen Punkte am Stuhl.
- Invertierte Reachability Map um Punkte zu filtern.



Loco-Manipulation Affordances

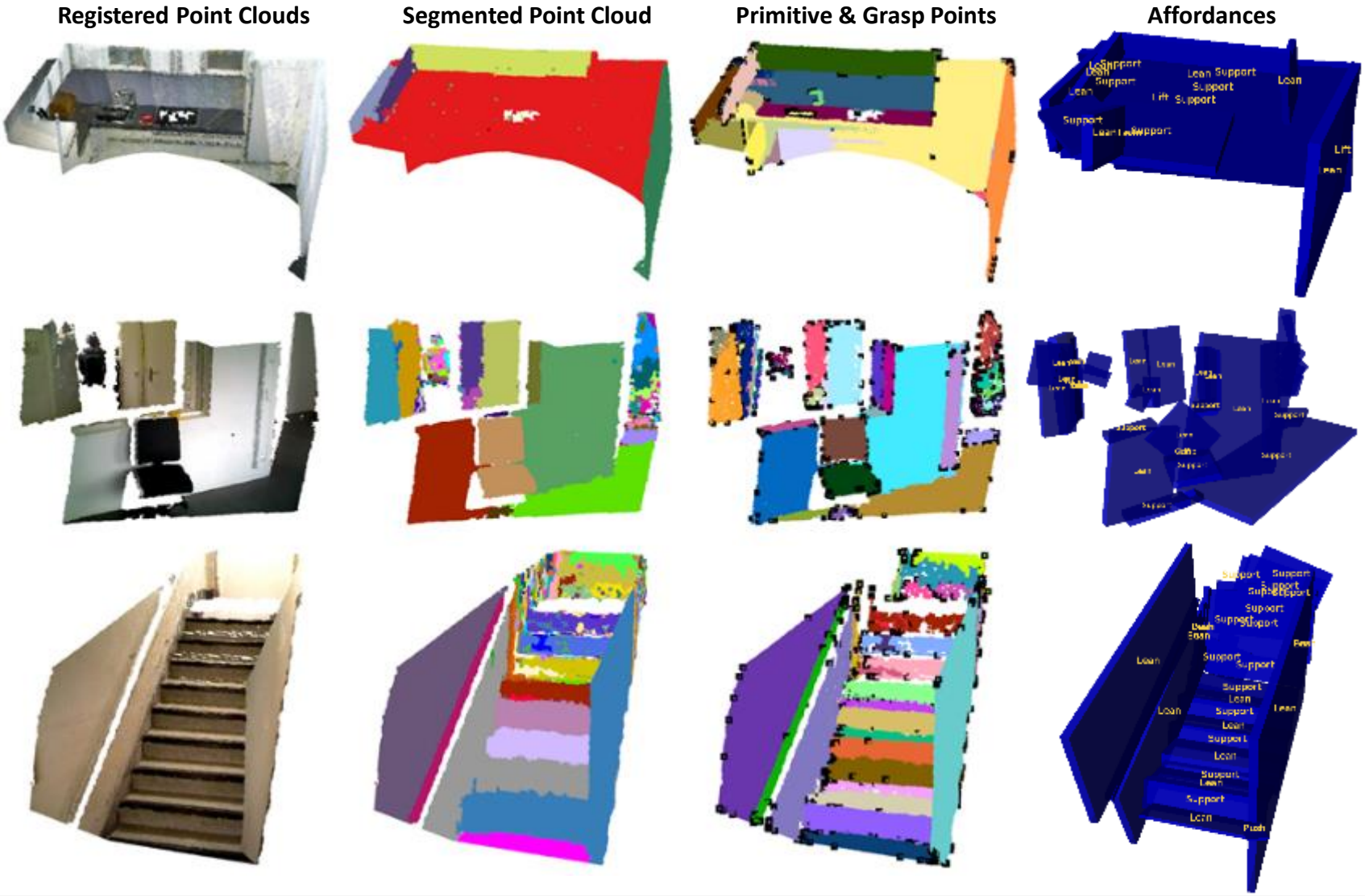




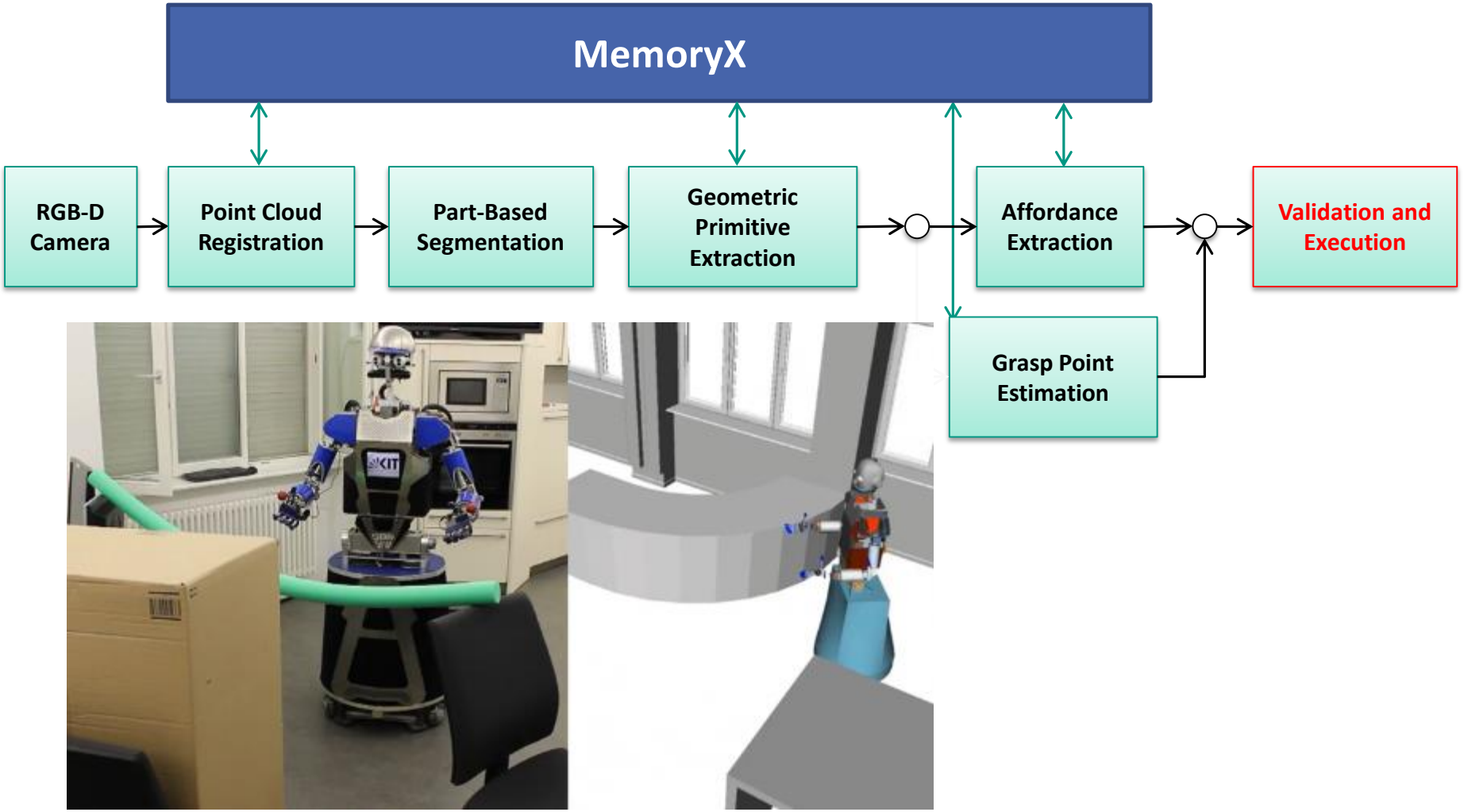
Look-up Tabelle!

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

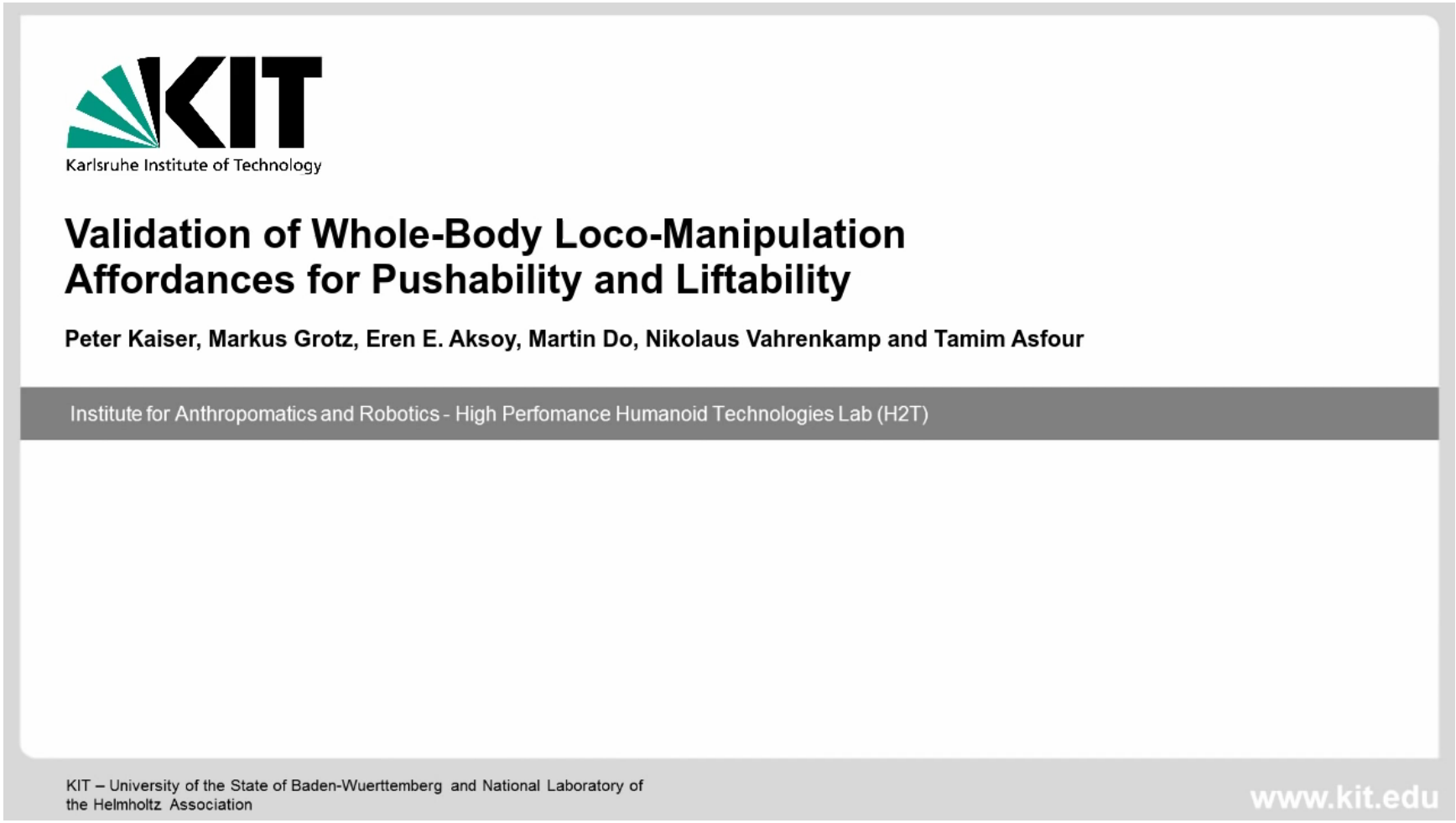
Loco-Manipulation Affordances



Loco-Manipulation Affordances



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.

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