Robotik II

- Lernende und Planende Roboter -

Prof. Dr.-Ing. R. Dillmann
Dr.-Ing. Sven R. Schmidt-Rohr
Dr.-Ing. Rainer Jäkel

Karlsruher Institut für Technologie

SS 2015

Institut für Anthropomatik (IFA)

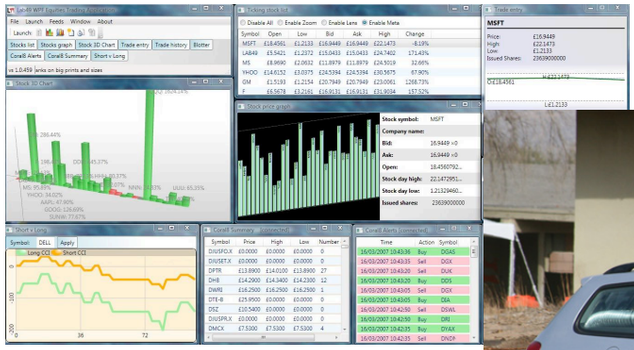
Motivation

- Die Börsenhändlerin und der Taxifahrer,
  die Schreinerin und der Krankenpfleger

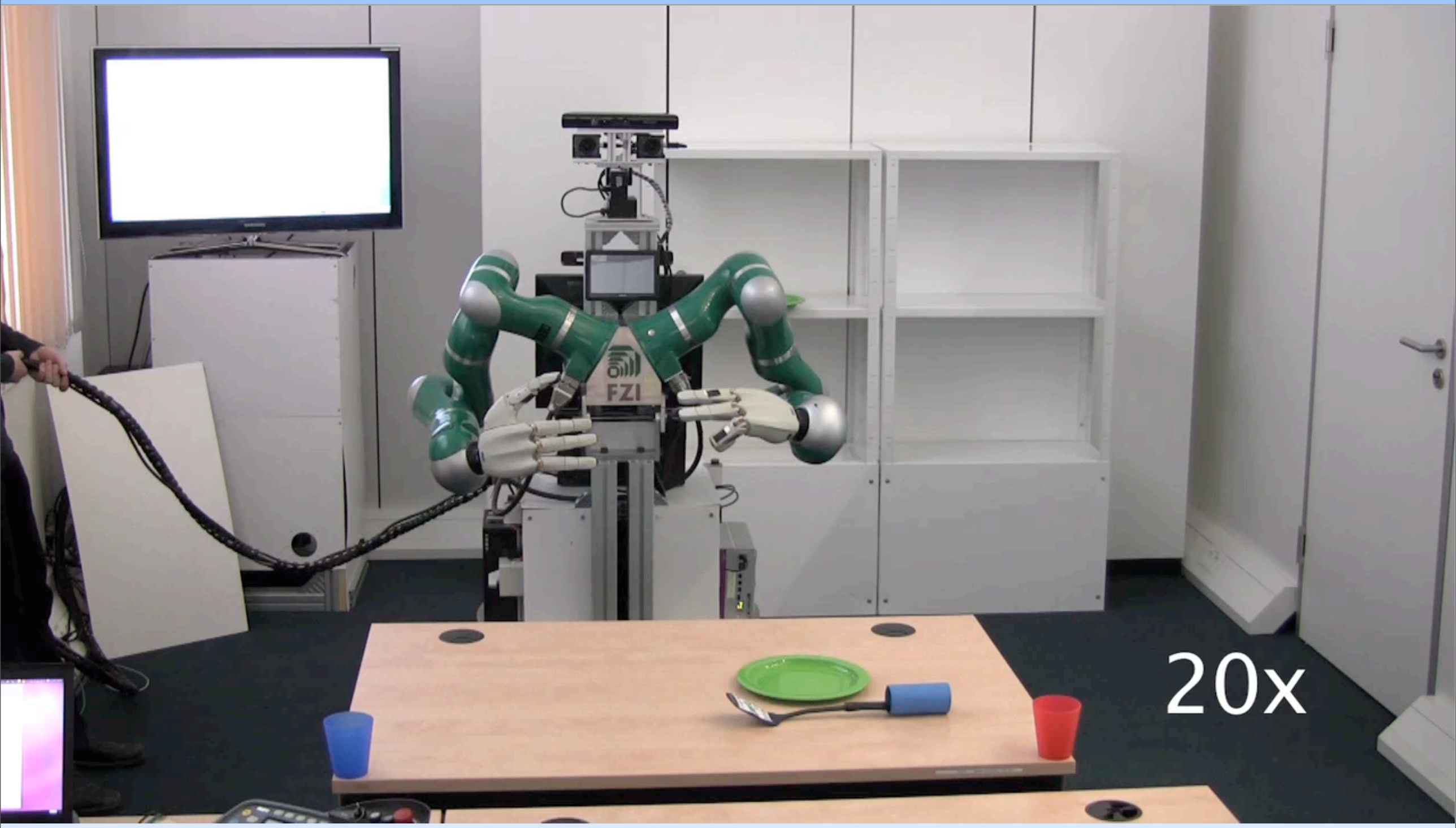
  werden von ähnlichen Algorithmen ersetzt werden

- Volumen dieser Industrie?

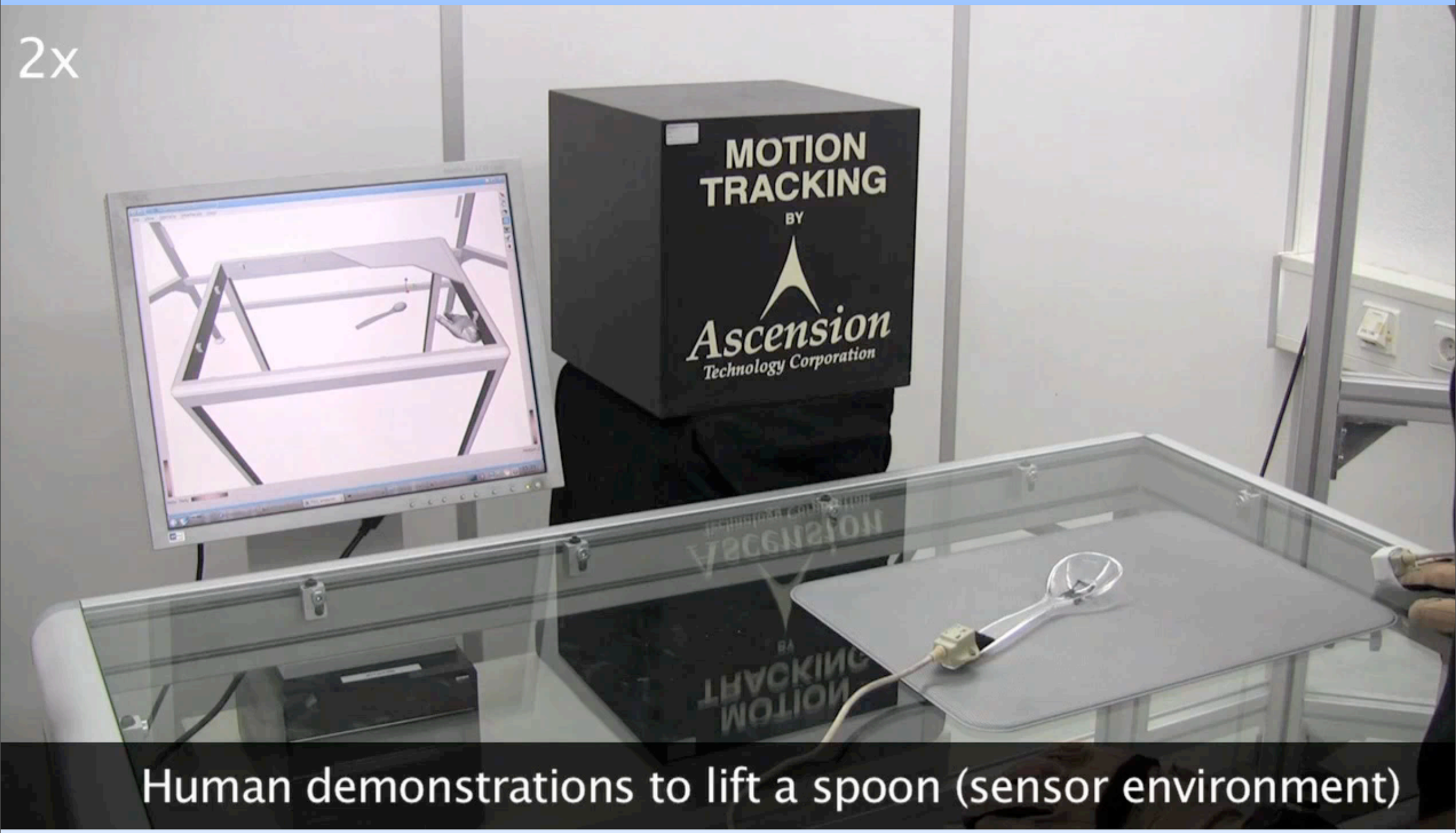
... so groß, wie der menschliche Arbeitsmarkt!



Motivation



Motivation



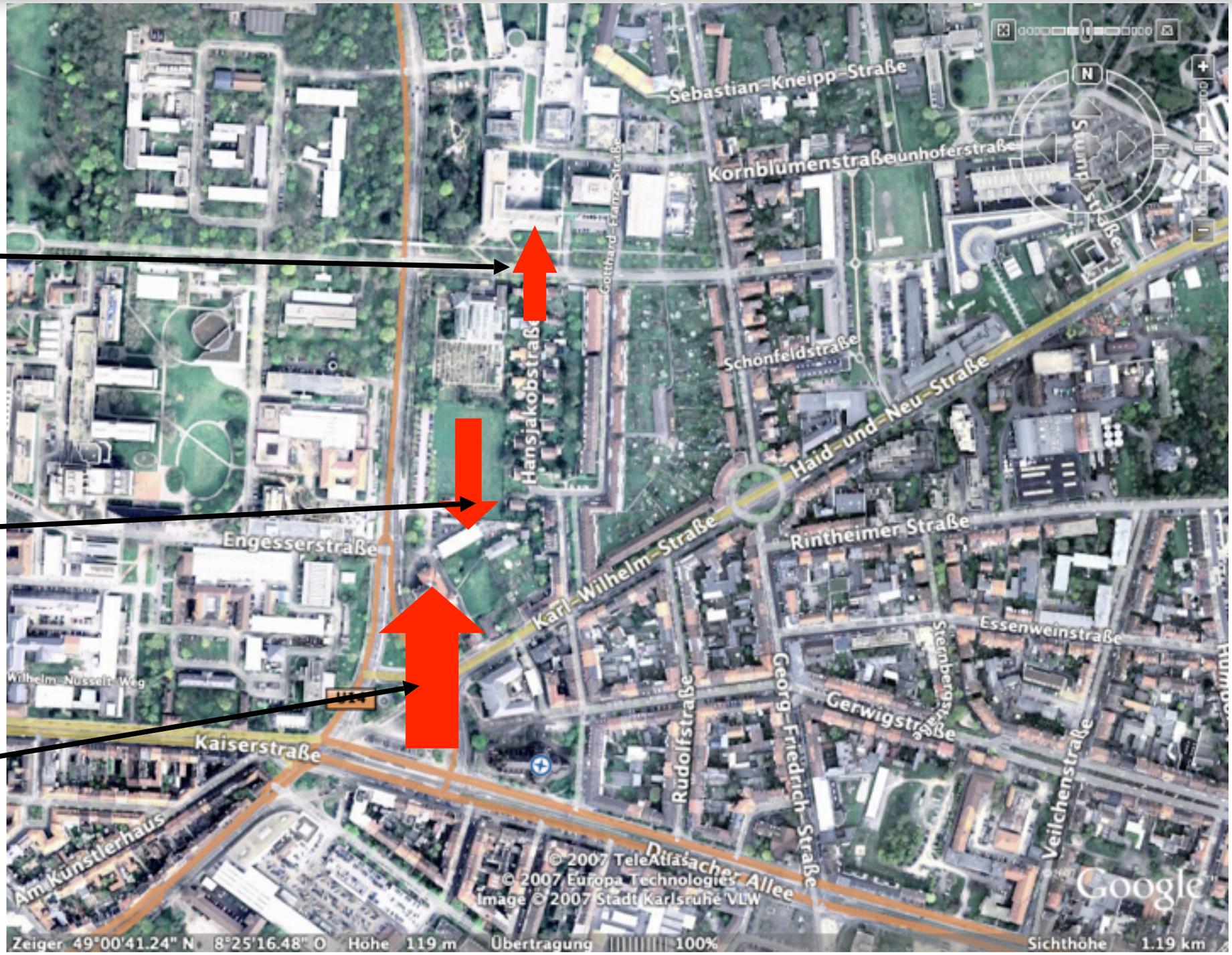
Human demonstrations to lift a spoon (sensor environment)

Wo findet man uns?

Hörsaal

Robotik II
Betreuer

HIS-
Sekretariat



HIS: Forschung Roboterprogrammierung

Höchstspannende Abschlussarbeiten:

www.robotik-abschlussarbeiten.de

Organisatorisches

- Folien: (***nach*** jeder Vorlesung !)
  - ILIAS
  - Password: HisRob2!

- Skript: Existiert seit Sommer 2007, 192 Seiten
  - Erwerb im HIS Sekretariat, Gebäude 50.20
  - Komplementär zu den Folien

- Hinweise zur Prüfungsvorbereitung in den Übungen

Vorlesungstermine

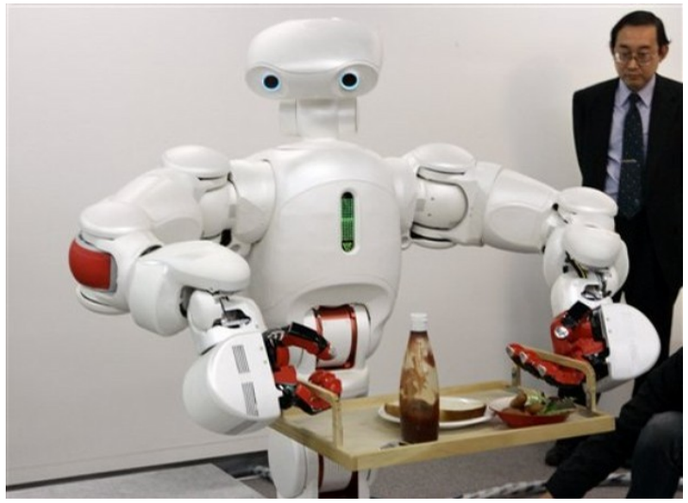
- **Fast** jeden Werk-Montag bis zum 13. Juli = 11 VL-Termine
  - **Keine VL:** 4. Mai, 15. Juni !
- Letzter Termin ist voraussichtlich keine Institutsführung, sondern reguläre Vorlesung!

Lernziel

- Aufbau, und Funktionsweise eines Robotersystems
- Ansätze zu Programmiermethoden für Robotersysteme
- Modellierung von Umwelt, Handlungs-, Aufgaben- und Problemlösungswissen
- Planungs- und Entscheidungsmethoden für autonome Robotersysteme
- Geeignete Verfahren und Robotersysteme für eine gegebene Aufgabe auswählen und anwenden können

Robotertypen in Robotik II

- In der Vorlesung werden quasi nur Manipulatoren betrachtet!
- Autonome Drohnen (Land/Wasser/Luft) oder kognitive Automobile nicht im Mittelpunkt



Gesamtüberblick

- Teilsysteme eines Roboters:
  - Mechanik (Robotik I)
  - Kinematik (Robotik I)
  - Achsregelung und Antrieb (Robotik I + III)
  - Sensoren (Robotik III)
  - Programmierung (Robotik II)
  - Planung und Modellierung (Robotik II)

Historische Entwicklung I

- 1946 Devol entwickelt Steuergerät, welches mechanische Maschinen durch elektrisch aufgezeichnete Signale steuert
- 1952 Einsatz von Teleoperatoren zur Handhabung radioaktiver Materialien
- 1959 erster kommerzieller Roboter der Firma Planet
- 1960 erster Industrieroboter „Unimate“ (hydraulisch)
- 1961 Einsatz der Unimate Roboter bei Ford
- 1961 Sprache APT für numerisch gesteuerte Maschinen
- 1973 erste Roboterprogrammiersprache WAVE am SRI entwickelt

Historische Entwicklung II

- 1974 Programmiersprache AL am SRI
- 1974 erste elektrisch angetriebene Roboter IRb6 von ASEA
- 1978 erste kommerzielle Robotersprache VAL für den Unimation-Roboter PUMA
- 1983 SRL Sprache in Karlsruhe entwickelt
- 1985 Veröffentlichung von PasRo
- 1988 Entwicklung der Sprache ROBOT-Tools

Historische Entwicklung III

Roboter erster Generation (1975)

- Anfahren fester Haltepunkte
- keine Sensoren
- einfache Aufgaben

Roboter zweiter Generation (1980)

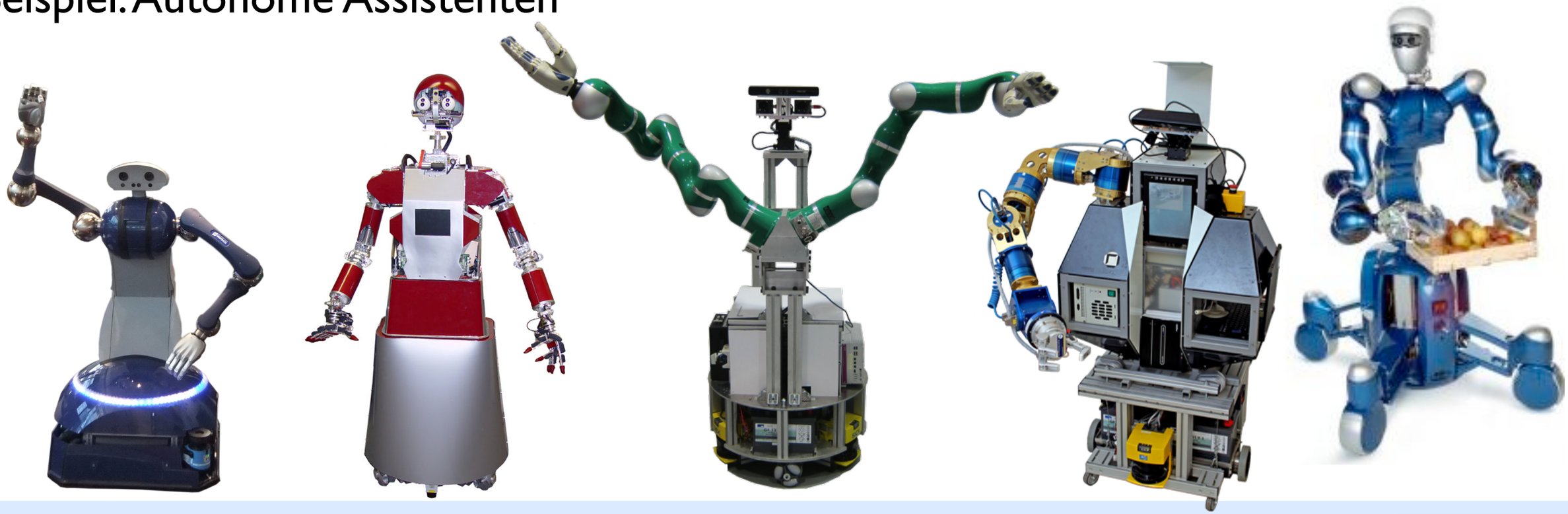
- verschiedene Bewegungsverfahren
- Steuerrechner
- höhere Programmiersprachen
- Sensorfunktionen in der Sprache
- explizite Programmierung

Historische Entwicklung III

Historische Entwicklung IV

Roboter dritter Generation (z.T. heute, Zukunft)

- Multisensorik
- adaptives Verhalten (Methoden der KI)
- Aufgabenorientierte Programmierung
- Autonomes Entscheiden und Planen
- Beispiel: Autonome Assistenten



Serviceroboter

- Entwicklung autonomer Robotersysteme
- Serviceroboter (teilweise als Prototypen realisiert)
  - Reinigungsmaschinen, autonome Staubsauger (Kärcher, Elektrolux ...)
  - autonomer Transport von Post, Waren (Universitäten, Industrie)
  - Mechatronisch anspruchsvolle Roboter (Universitäten, Japan: Automobilhersteller)
  - intelligente Assistenten (Universitäten)

Vision Haushaltsroboter

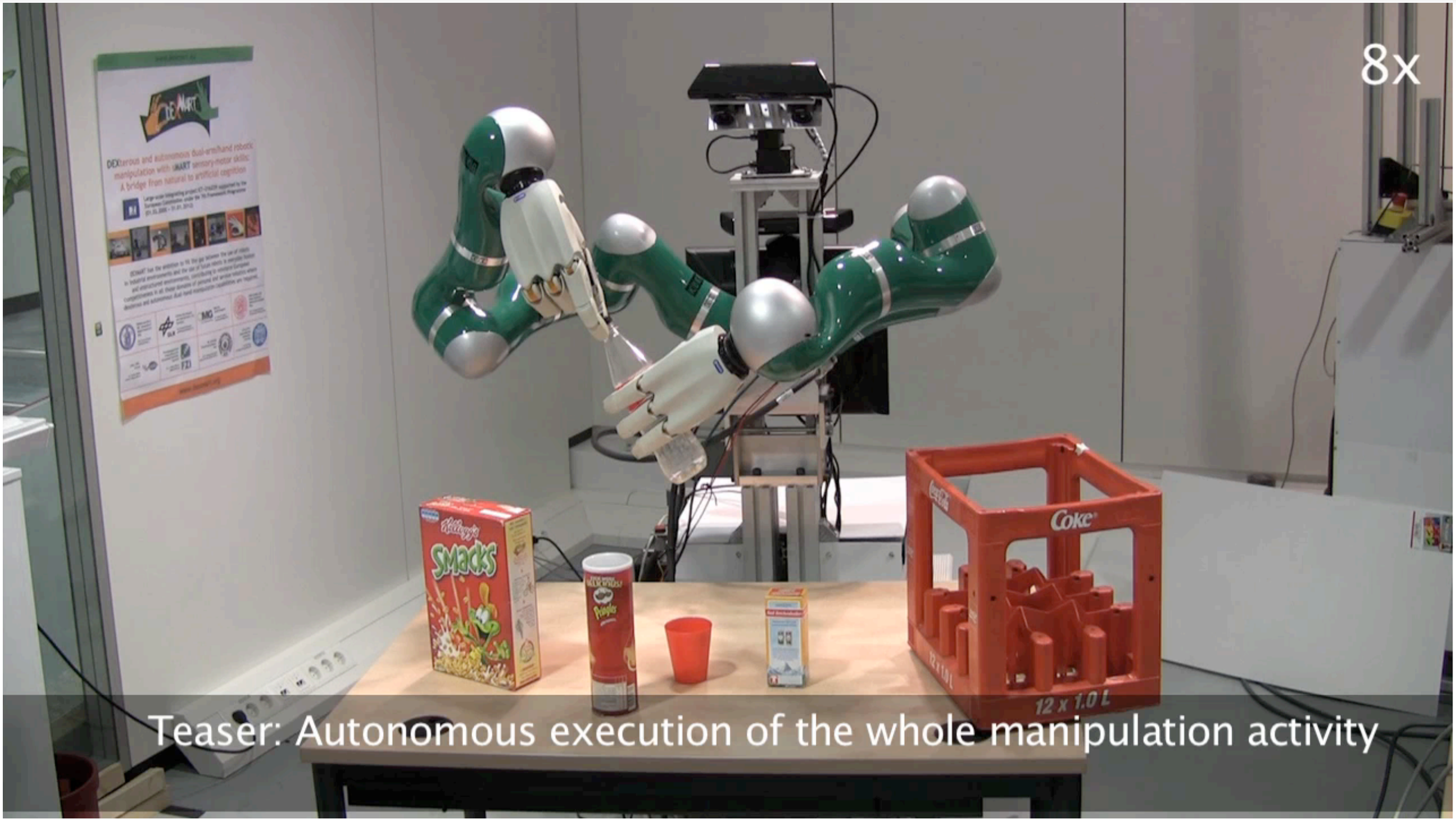
Unterstützung im Haushalt:

- Spülmaschine einräumen
- Tisch decken
- Wohnung Aufräumen

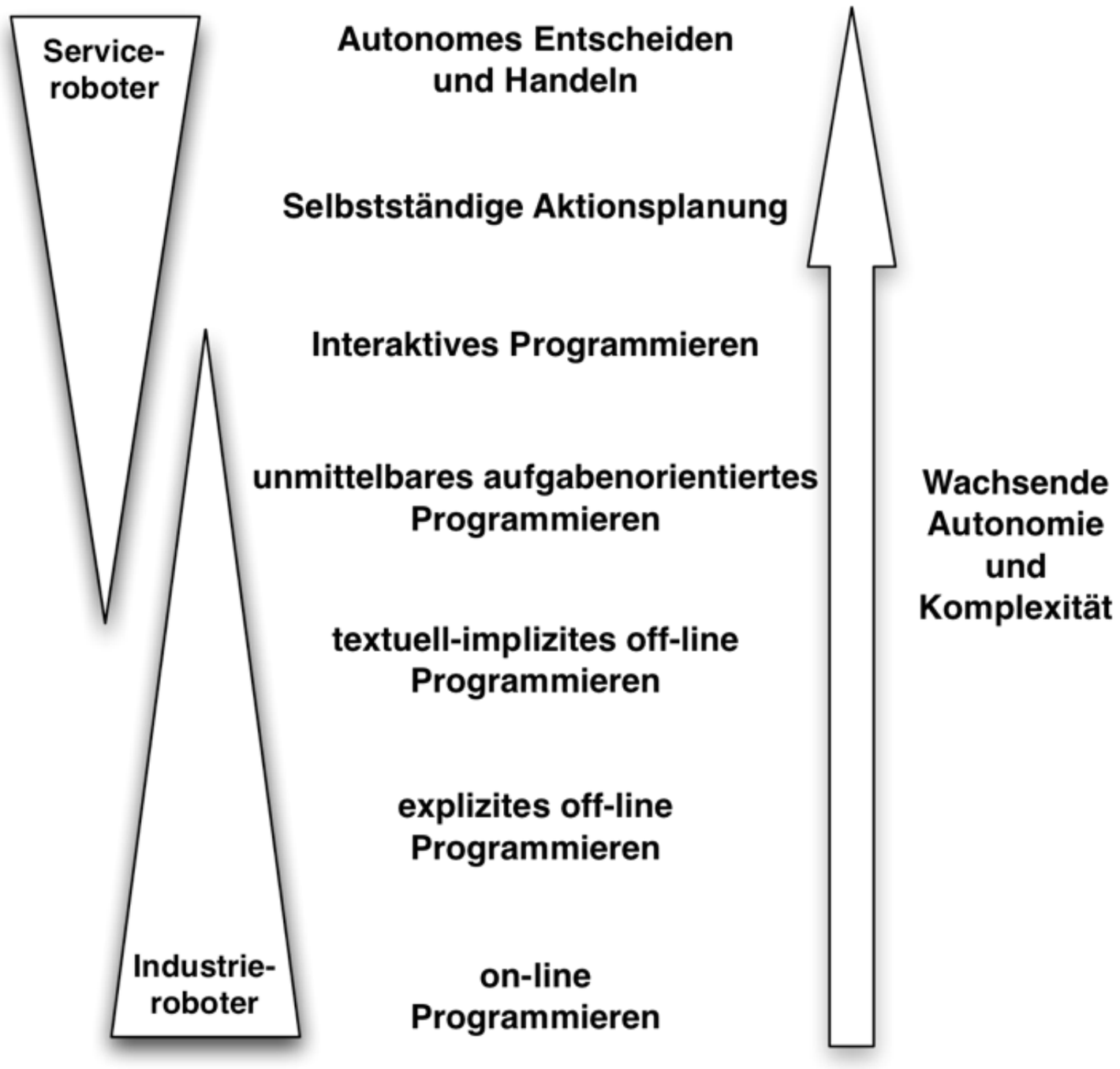
Fähigkeiten:

- Lernen von Tätigkeiten
- Flexibilität
- gute Perzeption (Sensorik)
- Zugriff auf Datenbanken
- Autonomes Entscheiden

Vision Haushaltsroboter



Strukturierung der Vorlesung



Inhalt der Vorlesung

- Einführung und Grundlagen (1.VL)
- Umweltmodellierung und Aufgabenmodellierung (2.VL)
- Programmierung durch Vormachen von Manipulationsaufgaben (3.-5.VL)
- Übung I (5.VL)
- Aktionsplanungsverfahren (6.VL)
- Probabilistisches Entscheiden (7.-9.VL)
- Übung II (10.VL)
- Stand der Technik bei Autonomen Roboterassistenten (11.VL)

Weiterführende Veranstaltungen

- Robotik III - Sensoren in der Robotik
- Biologisch motivierte Systeme
- Roboterpraktikum

- Bachelor-/Masterarbeiten
- HiWi-Jobs
- Fachpraktika

I

Grundlagen zur Programmierung von Robotern

Grundlagen

- Klassifizierung der Verfahren zur Roboter-programmierung
- Direkte Programmierung
- Roboterorientierte Programmierung

Klassifizierung der Roboterprogrammierverfahren

Kriterien:

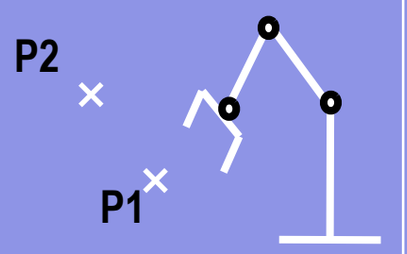
1. Programmierort
2. Art der Programmierung
3. Abstraktionsgrad der Programmierung

Klassifizierung der Roboterprogrammierverfahren

on-line

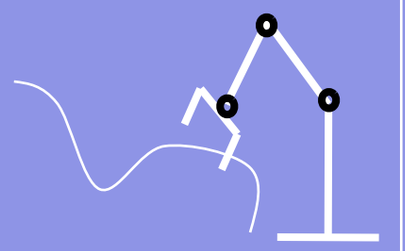
direkt

Teach-In



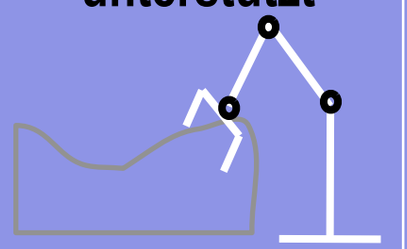
Punkte anfahren u. abspeichern

Play-Back



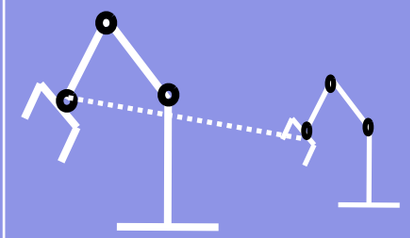
Bahn anfahren u. abspeichern

Sensor-unterstützt



Werkstückkontur erfassen

Master-Slave



Einsatz kinematischen Modells

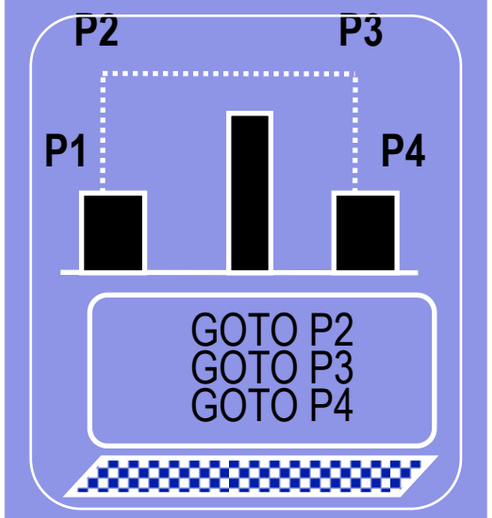
Hybride Verfahren

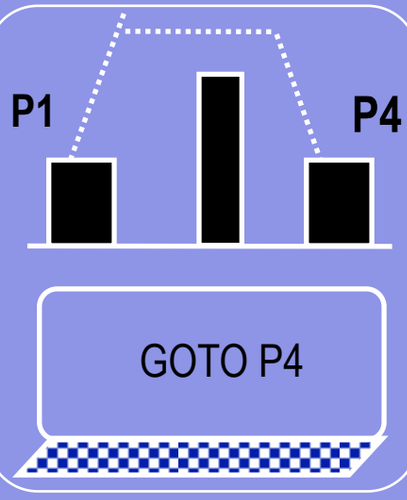
Interaktive Verfahren

PdV

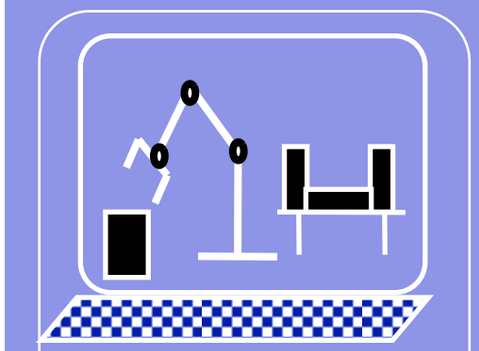
off-line

textuell

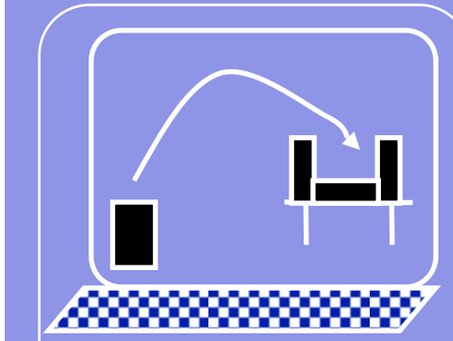




graphisch



Bewegungs-orientiert



Aufgaben-orientiert

1. Programmierort

on-line

- Die Programmierung erfolgt direkt am Roboter (an der Robotersteuerung)
- In der Literatur auch ***direkte*** oder prozessnahe Programmierung genannt

Hybride Verfahren

off-line

- Die Programmierung erfolgt ohne den Roboter mit Hilfe textueller, graphischer, interaktiver Methoden
- In der Literatur auch ***indirekte*** oder prozessferne Programmierung genannt

Klassifizierung der Roboterprogrammierverfahren

Kriterien:

1. Programmierort
2. Art der Programmierung
3. Abstraktionsgrad der Programmierung

2. Art der Programmierung

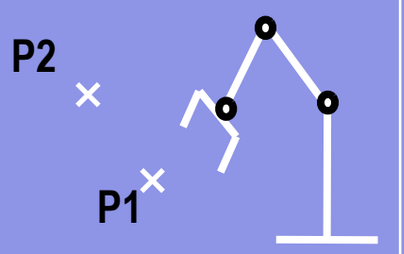
- 2.1 Direkte Programmierung
- 2.2 Textuelle Verfahren
- 2.3 Graphische/Gemischte Verfahren

2.1 Direkte Programmierung

on-line

direkt

Teach-In

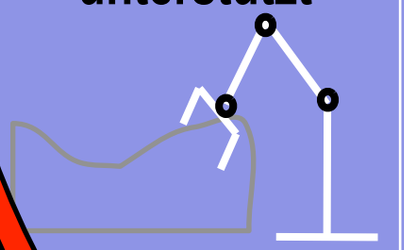


Punkte anfahren u. abspeichern

Play-Back

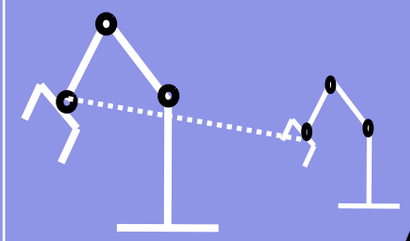
Bahn anfahren u. abspeichern

Sensor-unterstützt



Werkstückkontur erfassen

Master-Slave



Einsatz kinematischen Modells

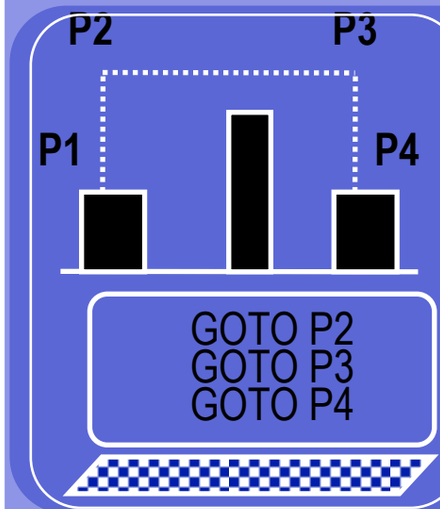
Hybride Verfahren

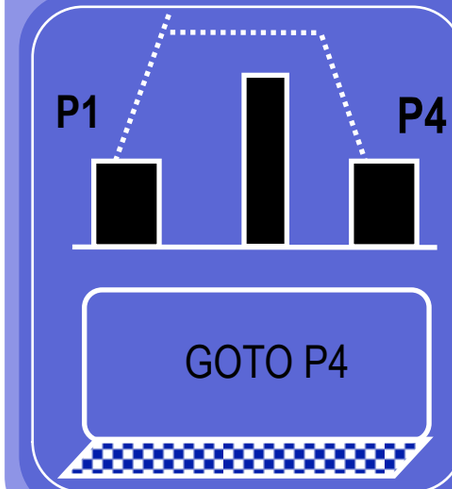
Interaktive Verfahren

PdV

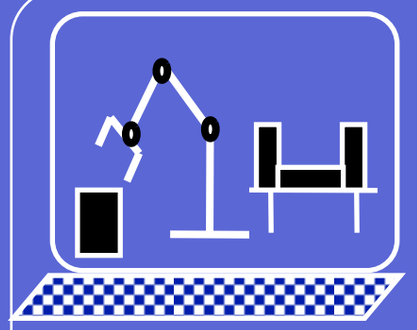
off-line

textuell

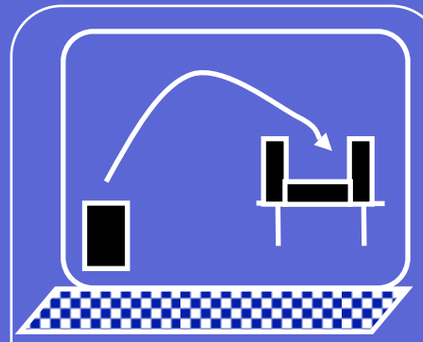




graphisch



Bewegungs-orientiert



Aufgaben-orientiert

explizit

implizit

2.1 Direkte Programmierung

Einstellen des Roboters

ältestes Programmierverfahren.

Der Bewegungsbereich jedes Gelenks wird durch Stopper eingeschränkt.

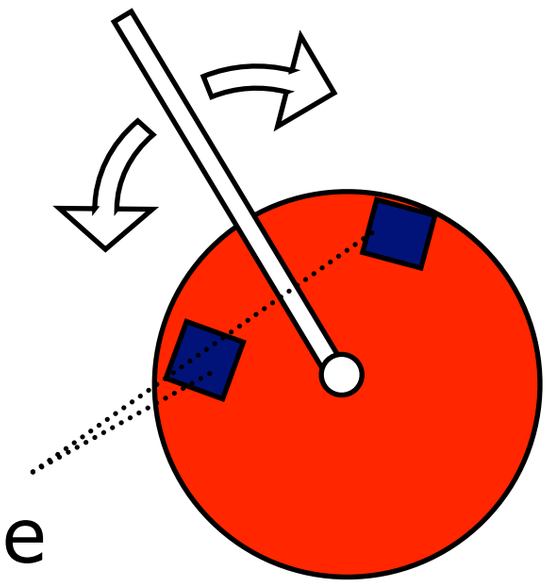
Die Bewegungen erfolgen für jedes Gelenk einzeln bis zum Anschlag.

Zuordnung zwischen Anfahrpunkten zu Stopper kann mit Codiermatritzen erfolgen.

sog.: „Bang-Bang-Robot“.

Nachteil:

sehr kleine Menge von Anfahrpunkten.



Anschläge

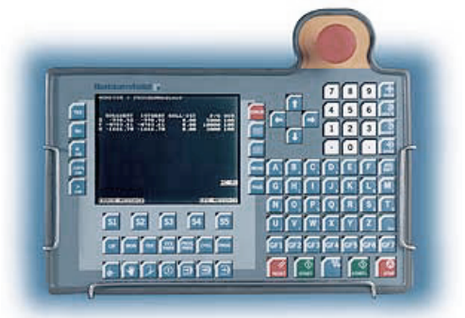
2.1 Direkte Programmierung

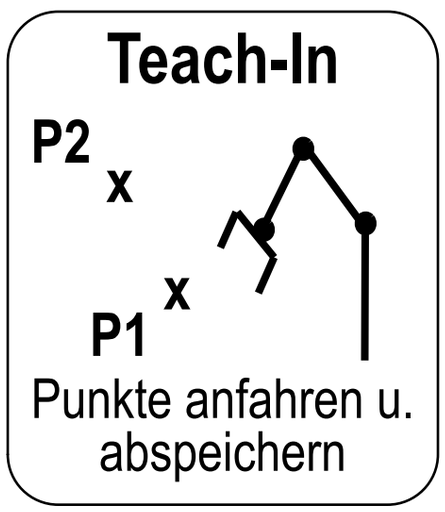
Teach-In Programmierung

Anfahren markanter Punkte der Bahn mit manueller Steuerung
(Teach Box, Teach Panel, weitere: Spacemouse, Teach-Kugel)

Funktionalität einer Teach Box:

- Einzelbewegung der Gelenke
- Bewegung des Effektors in 6 Freiheitsgraden
- Speichern / Löschen von Anfahrpunkten
- Eingabe von Geschwindigkeiten
- Eingabe von Befehlen zur Bedienung des Greifers
- Starten / Stoppen ganzer Programme





2.1 Direkte Programmierung

Vorgehensweise beim Teach-In

- Anfahren markanter Punkte der Bahn
- Bahn = Folge von Zwischenpunkten
- Speichern der Gelenkwerte
- Ergänzung der gespeicherten Werte
  um Parameter wie Geschwindigkeit, Beschleunigung usw.

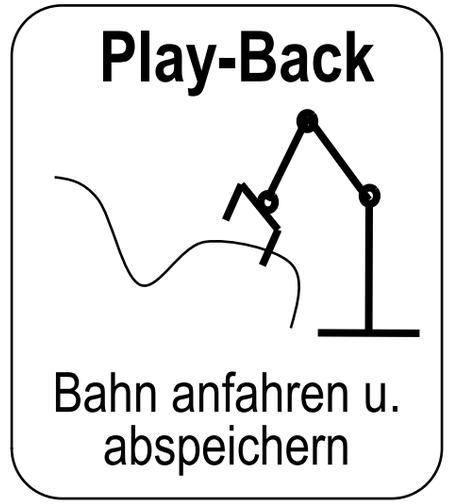
Anwendung:

- in der Fertigungsindustrie (Punktschweißen, Nieten)
- Handhabungsaufgaben (Pakete vom Fließband nehmen)

2.1 Direkte Programmierung

Play-Back- (manuelle) Programmierung

- Einstellung des Roboters auf Zero-Force-Control
  (Roboter kann durch den Bediener bewegt werden)
- Abfahren der gewünschten Bahn
- Speichern der Gelenkwerte:
  - automatisch (definierte Abtastfrequenz) oder
  - manuell (durch Tastendruck)



Anwendung:

- mathematisch schwer beschreibbare Bewegungsabläufe
- Integrierung der handwerklichen Erfahrung
- Typische Einsatzbereiche sind: Lackieren oder Kleben

2.1 Direkte Programmierung

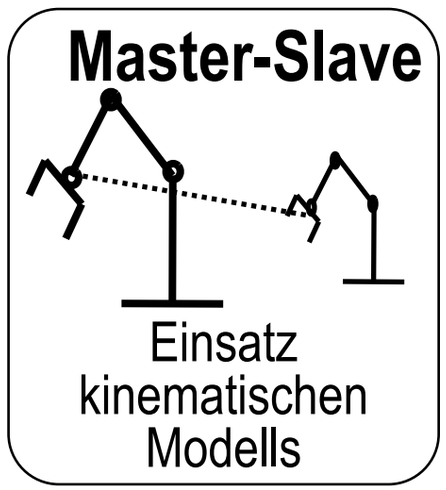
Nachteile der Play-Back-Programmierung:

- schwere Roboter schwierig zu bewegen
- wenig Platz in engen Fertigungszellen für Bediener
  → dadurch Sicherheitsrisiko
- hoher Speicherbedarf (bei hoher Abtastrate)
- schlechte Korrekturmöglichkeiten

2.1 Direkte Programmierung

Master-Slave-Programmierung

- Bediener führt einen kleinen, leicht bewegbaren Master Roboter (entspricht einem kinematischen Modell des Slave-Roboters)
- Bewegung wird auf den Slave-Roboter übertragen
- Bewegungen werden synchron ausgeführt
- Slave-Roboter wirkt als Kraftverstärker



Anwendung:

- Handhabung großer Lasten bzw. großer Roboter

Vor- und Nachteile:

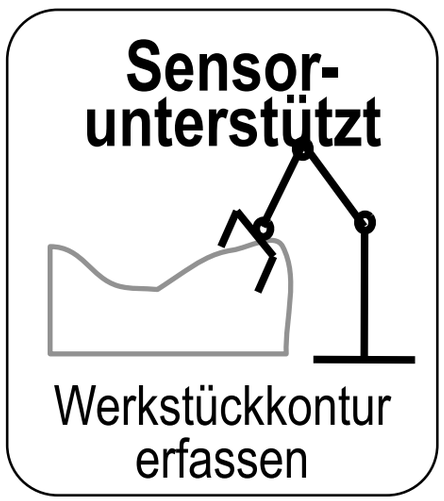
- teuer, da zwei Roboter benötigt werden
- Möglichkeit, auch schwerste Roboter zu programmieren

2.1 Direkte Programmierung

Sensorunterstützte Programmierung

Manuell

- Bediener führt Programmiergriffel (Leuchtstift, Laserstift) entlang der abzufahrenden Bahn
- Erfassung der Bewegung durch externe Sensoren (zB. Kameras, Laserscanner)
- Berechnung der inversen Kinematik
- Abspeichern der Bahn als Folge der Gelenkwinkel



Automatisch

- Vorgabe des Start- und Zielpunktes
- Sensorische Ertastung der Sollkontur (zB. über Kraft-Momenten-Sensor)

Anwendung:

- Schleifen, Entgraten von Werkstücken

2.1 Zusammenfassung direkte Programmierung

Vorteile:

+ schnell bei einfachen Trajektorien
+ sofort anwendbar
+ geringe Fehleranfälligkeit
+ Bediener benötigt keine Programmierkenntnisse
+ kein Modell der Umwelt erforderlich

Nachteile:

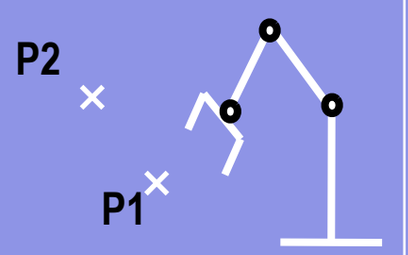
- hoher Aufwand bei komplexen Trajektorien
- nur mit und am Roboter möglich
- spezifisch für einen Robotertyp
- Verletzungsgefahr durch Roboter

Roboterprogrammierverfahren

on-line

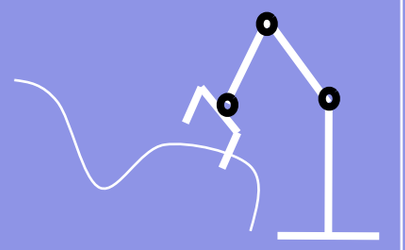
direkt

Teach-In



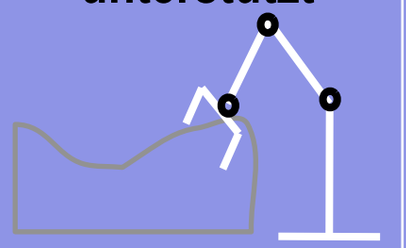
Punkte anfahren u. abspeichern

Play-Back



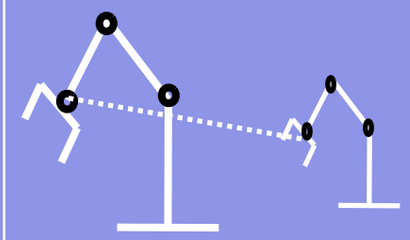
Bahn anfahren u. abspeichern

Sensor-unterstützt



Werkstückkontur erfassen

Master-Slave



Einsatz kinematischen Modells

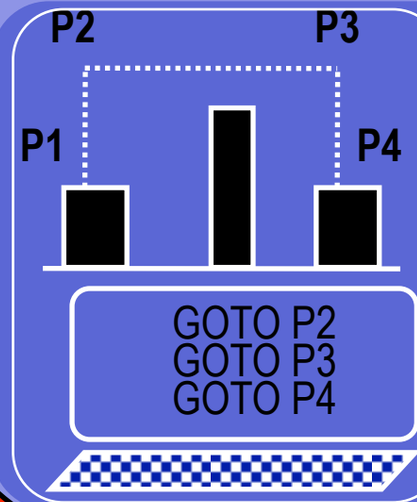
Hybride Verfahren

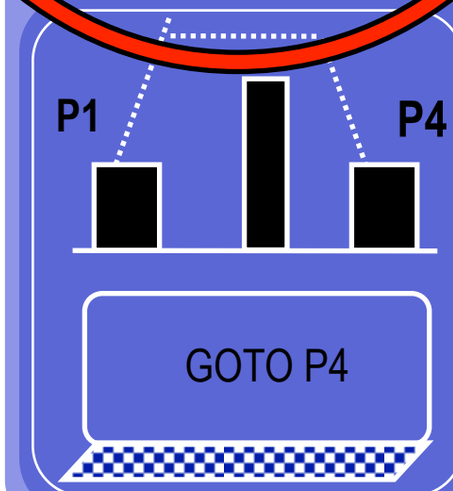
Interaktive Verfahren

PdV

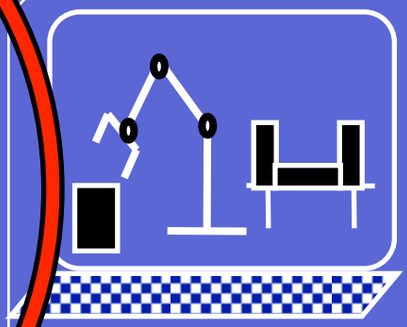
off-line

textuell

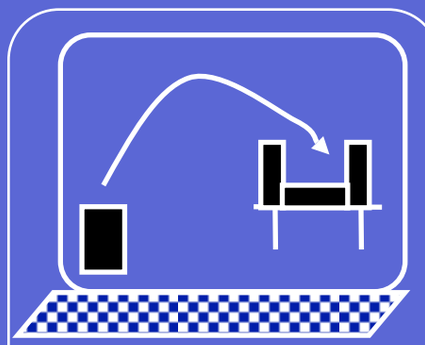




graphisch



Bewegungs-orientiert

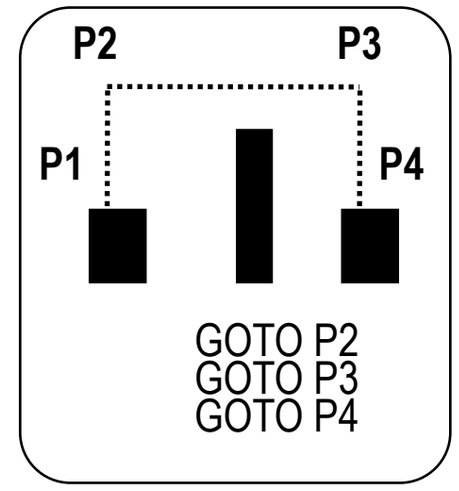


Aufgaben-orientiert

explizit

implizit

2.2 Textuelle Verfahren



- Programmierung erfolgt mittels erweiterter, höherer Programmiersprachen (PasRo, Val, etc.)

  → Robotersteuerprogramm

- Vorteile:
  - Programmierung kann unabhängig vom Roboter erfolgen
  - strukturierte, übersichtliche Programmierlogik
  - Erstellung komplexer Programme (Einbezug von Wissensbasis, Weltmodell, Auswertung von Sensoren)
- Nachteile:
  - Bediener benötigt Programmierkenntnisse
  - keine / schlechte Korrekturmöglichkeiten

2.2 Textuelle Verfahren: Sprache

Sprache: (DIN 66025)
Programm = Menge numerierter Sätze
z.B. N70 G00 X20 Z12
entspricht Werkzeug im Eilgang (G00) an Position X=20 Z=12 bewegen.
(N = Satznummer)

Sprachen:	APT (Automatically Programmed Tools)	1961 MIT
	EXAPT (Extended Subset of APT)	1966 Aachen

z.B.

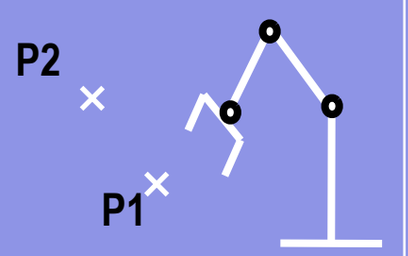
P1=POINT/20,12	Variablendefinition
Rapid	Eilgang
GOTO/P1	Positionierung auf P1

Roboterprogrammierverfahren

on-line

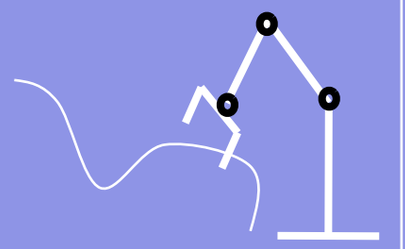
direkt

Teach-In



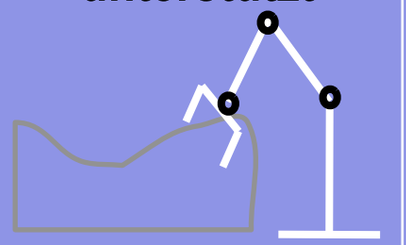
Punkte anfahren u. abspeichern

Play-Back



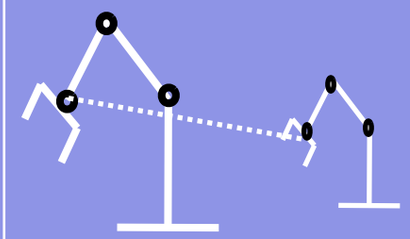
Bahn anfahren u. abspeichern

Sensor-unterstützt



Werkstückkontur erfassen

Master-Slave



Einsatz kinematischen Modells

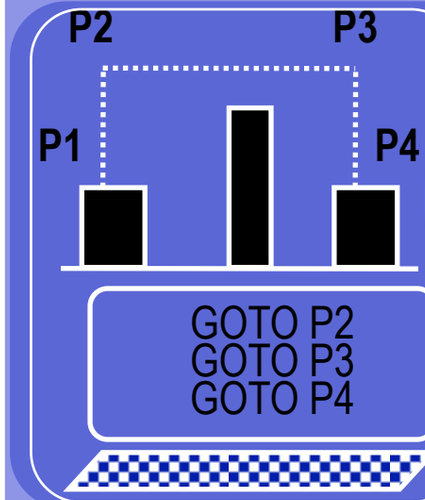
Hybride Verfahren

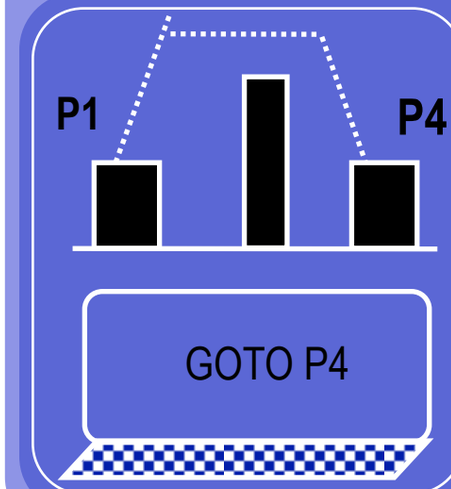
Interaktive Verfahren

PdV

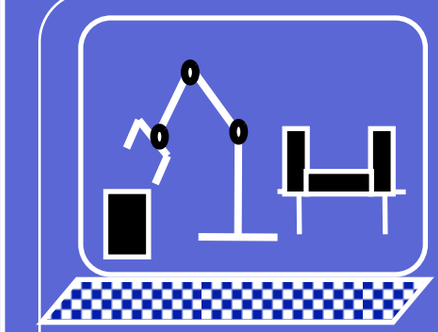
off-line

textuell

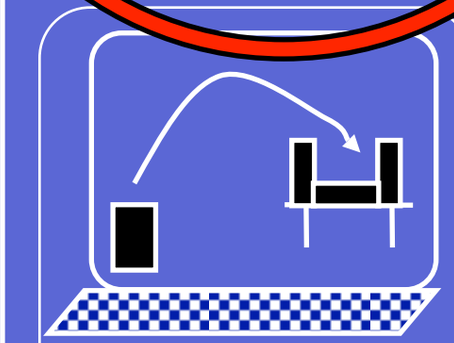




graphisch



Bewegungs-orientiert



Aufgaben-orientiert

explizit

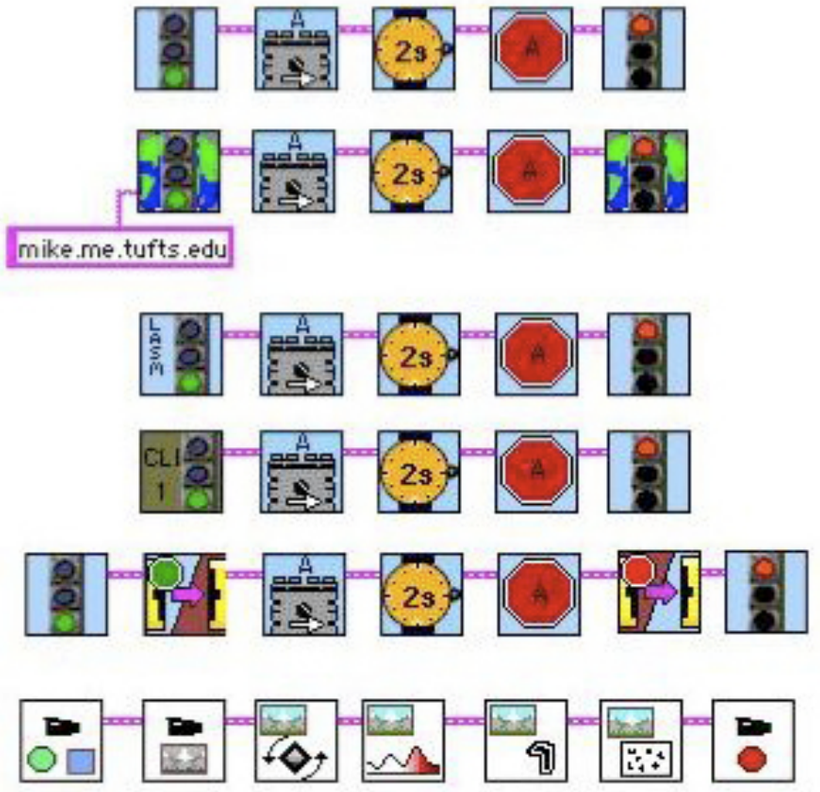
implizit

2.3 Graphische/Gemischte Verfahren

- Graphische Darstellung von Kontrollstrukturen (if/else, Schleifen, Marken…)
- Kopplung mit
  - Visualisierungs-Tool
  - Simulations-Tool (z.B. RobCAD)
- Trajektorien durch Interpolation aus Stützpunkten, Freihandzeichnen, analytisch
- Operationen Icon- oder Menu-gesteuert
- Vorteile: Rapid prototyping, Validierung des Programms durch Simulation
- Nachteil: 2d-Sicht des Anwenders

2.3 Graphische/Gemischte Verfahren

- Simple graphische Programmierung: Lego Mindstorms
  - Ikonische Programmierung
  - Leicht verständlich
  - Beschränkte Möglichkeiten



2.3 Graphische/Gemischte Verfahren

- Graphische Programmierung basierend auf sensorieller Erfassung der Benutzervorführung

  → Simulation der Roboterprogramme

- Vorteile:
  - Programmierer benötigt weniger Programmierkenntnisse
  - einfache Programmierung, leichte Fehlererkennung
  - schnelles Erstellen komplexer Programme (rapid prototyping)
- Nachteile:
  - sensorielle Benutzererfassung noch zu ungenau
  - Leistungsfähige Hardware für Signalanalyse, Modellierung, ...
  - Komplexe Modelle benötigt

Klassifizierung der Roboterprogrammierverfahren

Kriterien:

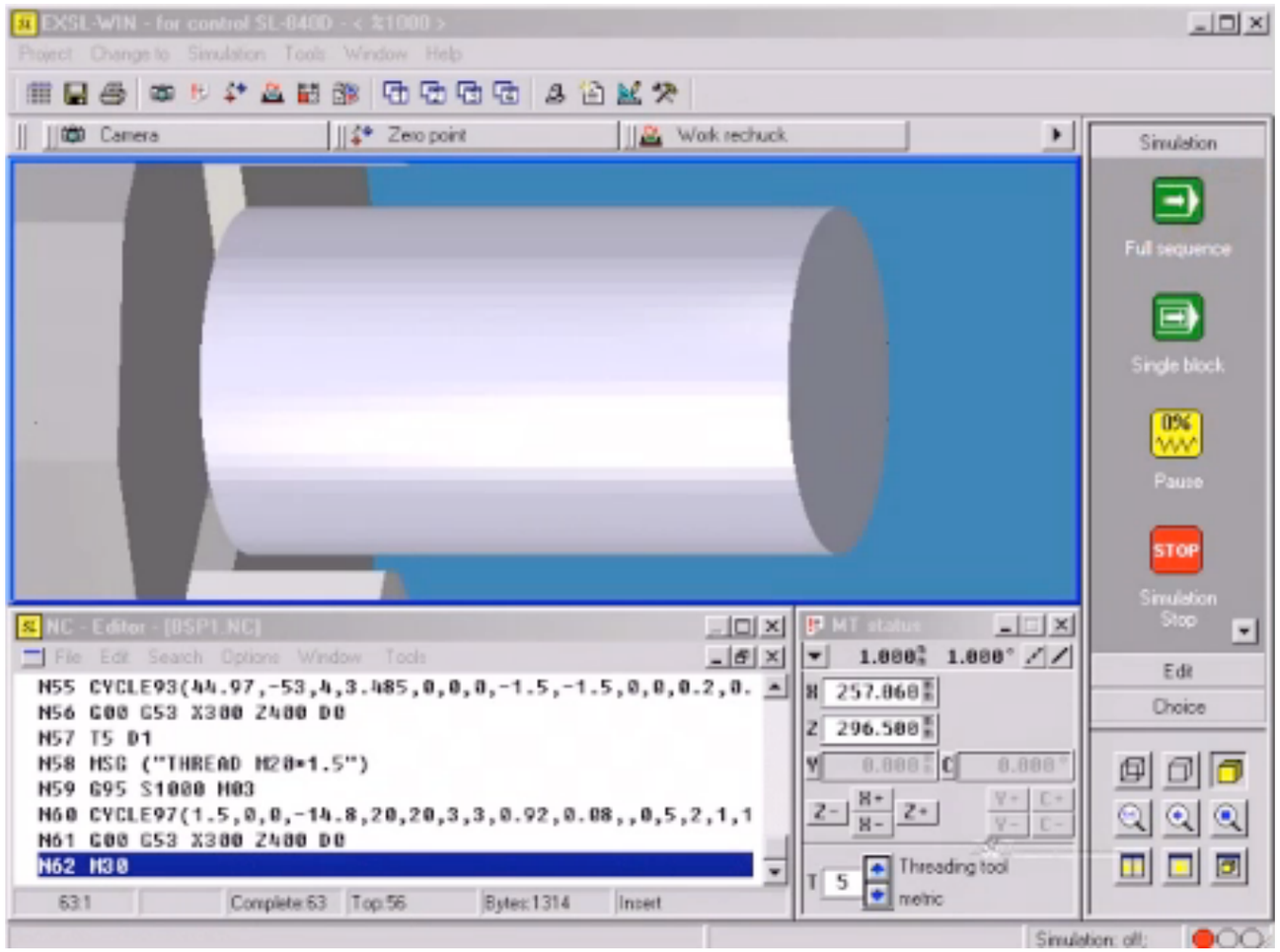
1. Programmierort
2. Art der Programmierung
3. Abstraktionsgrad der Programmierung

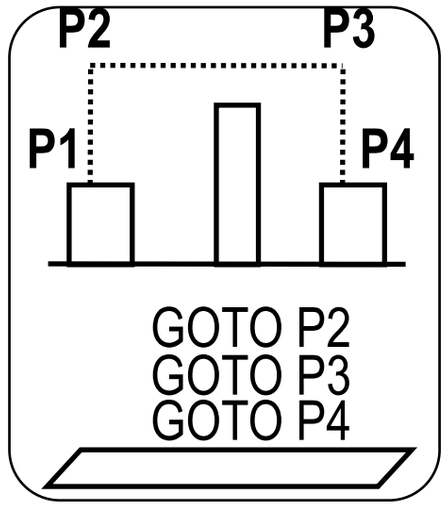
3. Abstraktionsgrad der Programmierung

Explizite oder roboterorientierte Programmierung

Bewegungen und Greiferbefehle sind direkt in eine Programmiersprache eingebunden.

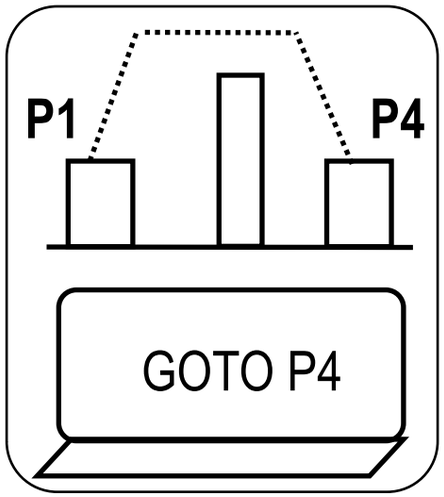
„Wie ist es zu tun?“





3. Abstraktionsgrad der Programmierung

Implizite oder aufgabenorientierte Programmierung



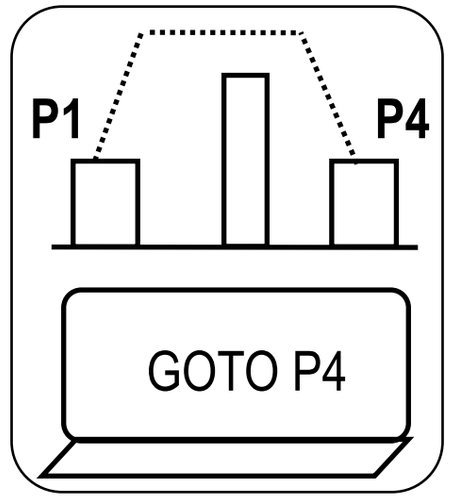
Die Aufgabe, die der Roboter durchführen soll, wird beschrieben, z.B. in Form von Zuständen.

„Was ist zu tun?“

- Abstrakte Form der Programmierung erfolgt in den Phasen
  1. Modellierung der Umwelt
  2. Spezifikation der Aufgaben
  3. Erzeugung der Roboterprogramme
- u. U. erfolgt vor der Ausführung eine Überprüfung des Roboterprogramms (Simulation)

Implizite oder aufgabenorientierte Programmierung

Beispiel: Einschenken unter Berücksichtigung von Hindernissen



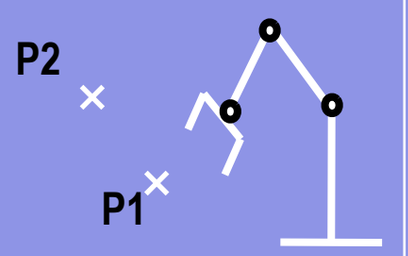
Roboterorientierte Programmierung

Roboterorientierte Programmierung

on-line

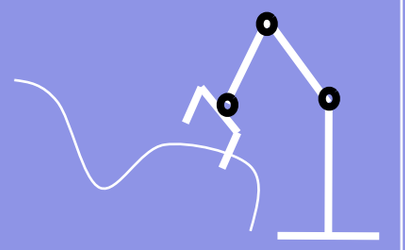
direkt

Teach-In



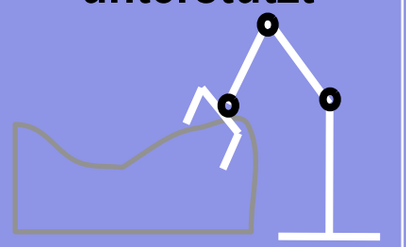
Punkte anfahren u. abspeichern

Play-Back



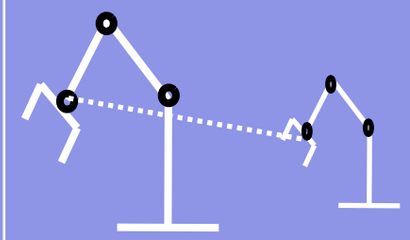
Bahn anfahren u. abspeichern

Sensor-unterstützt



Werkstückkontur erfassen

Master-Slave



Einsatz kinematischen Modells

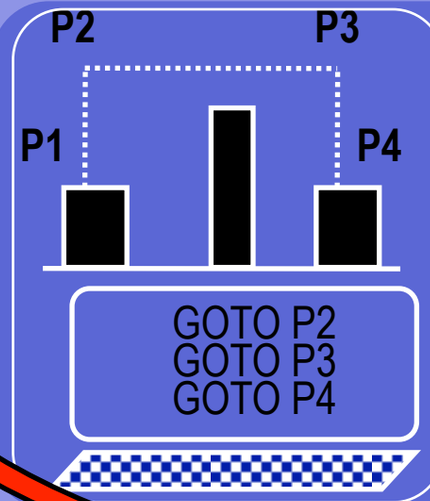
Hybride Verfahren

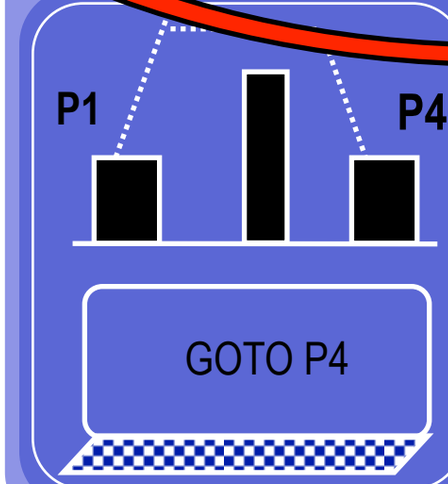
Interaktive Verfahren

PdV

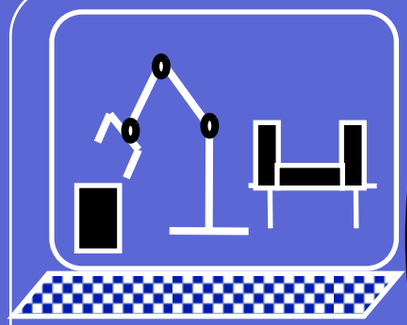
off-line

textuell

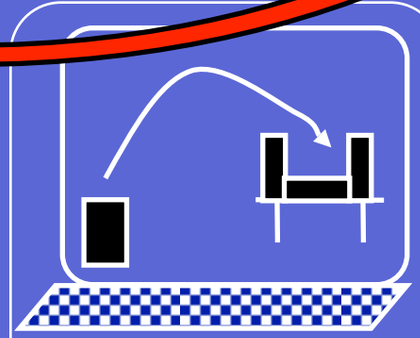




graphisch



Bewegungs-orientiert

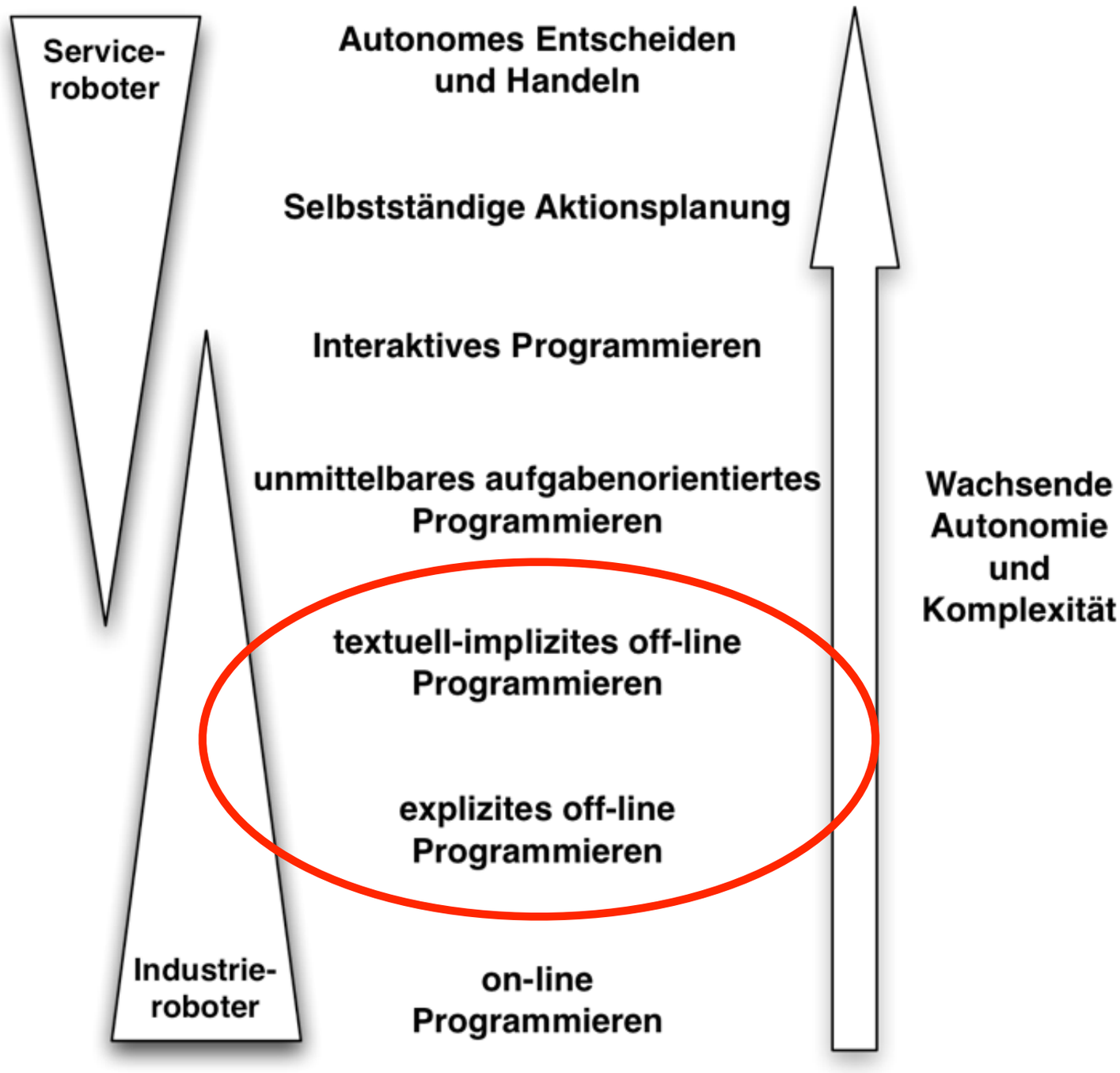


Aufgaben-orientiert

explizit

implizit

Einordnung

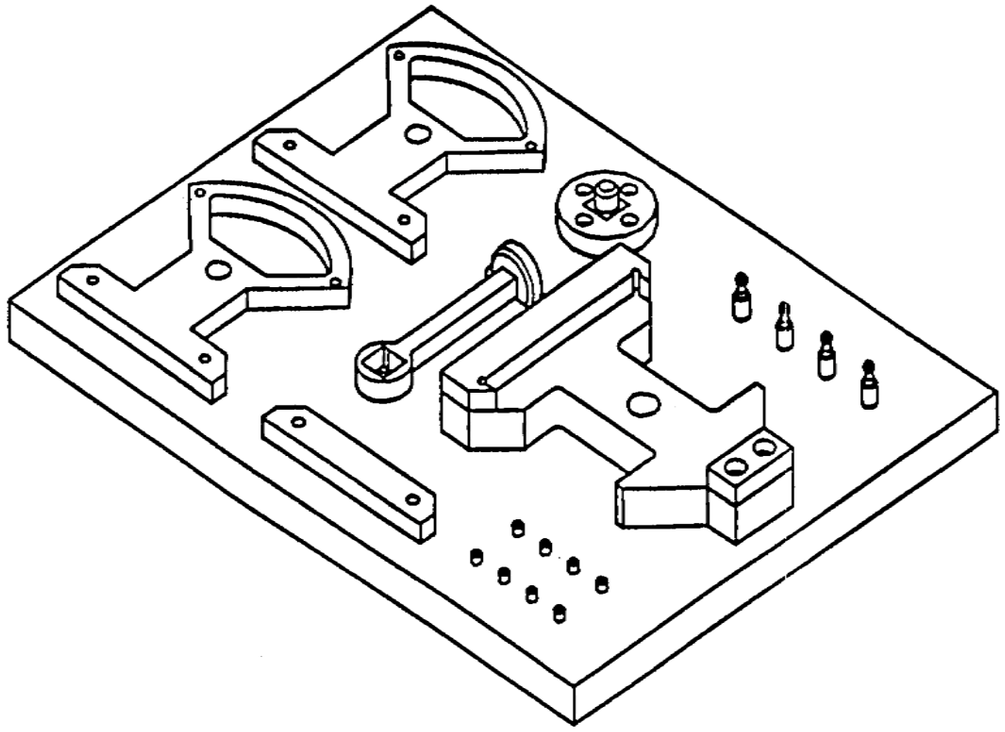

Service-
roboter
Autonomes Entscheiden
und Handeln
Selbstständige Aktionsplanung
Interaktives Programmieren
unmittelbares aufgabenorientiertes
Programmieren
textuell-implizites off-line
Programmieren
explizites off-line
Programmieren
Industrie-
roboter
on-line
Programmieren
Wachsende
Autonomie
und
Komplexität

Motivation

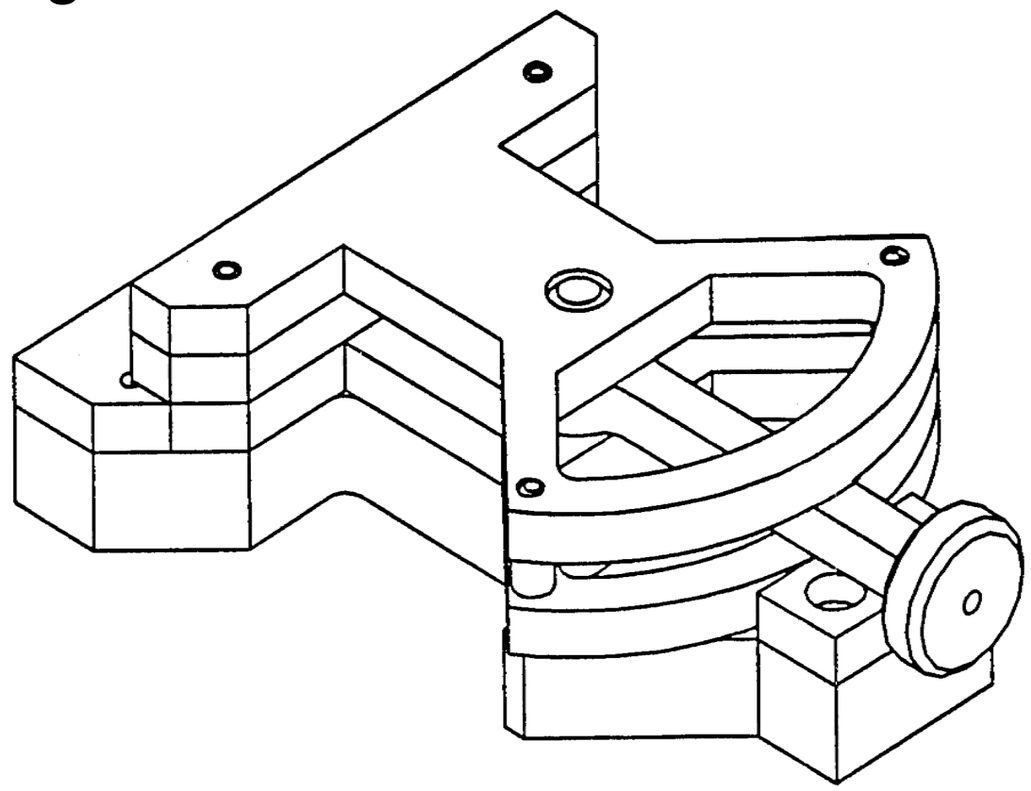
Aufgabenmodell: gegeben durch Anfangs- und Endzustand

(zB. Relationale Darstellung)

Bsp: Der Cranfield-Montage-Benchmark

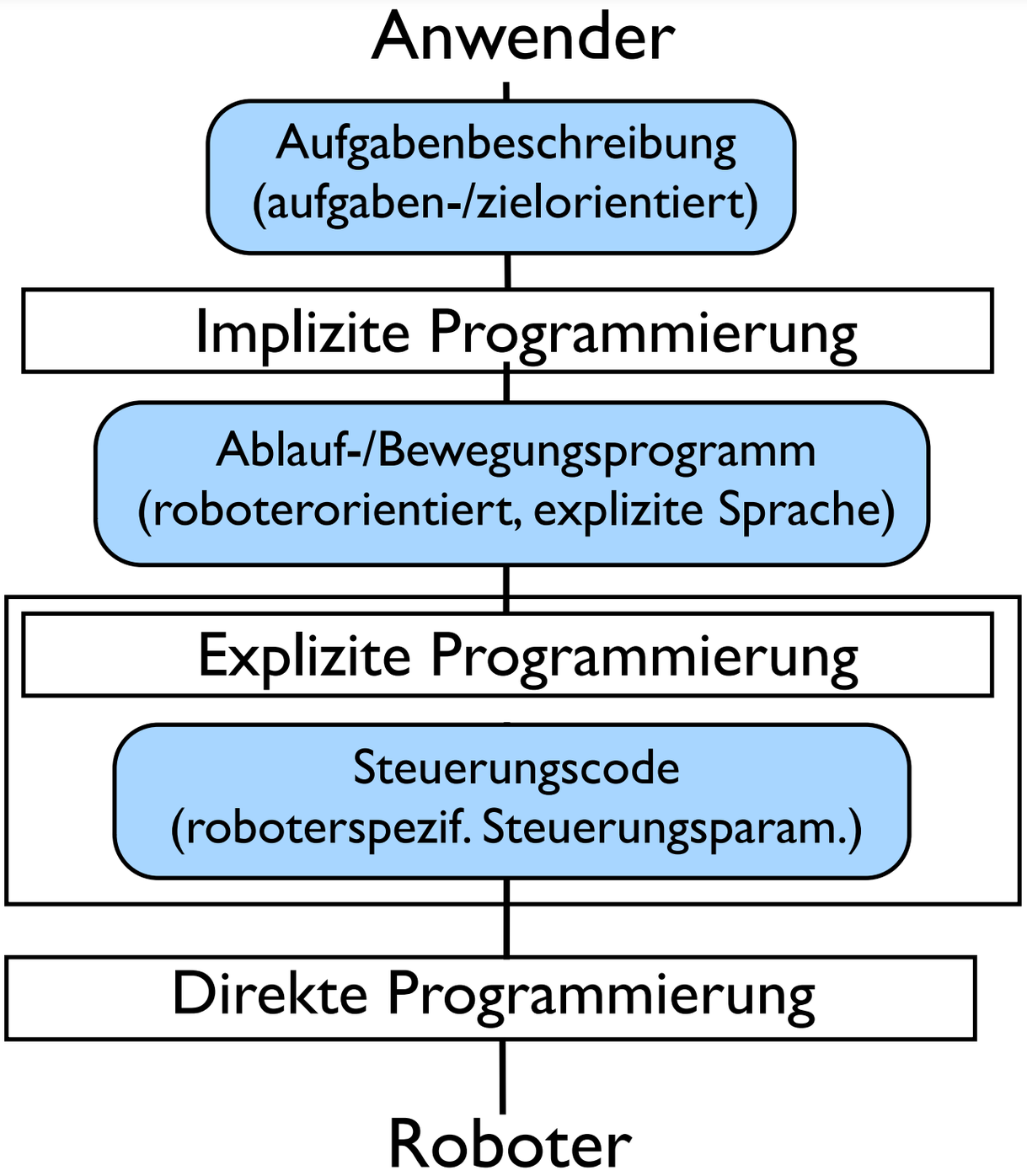


Anfangszustand



Endzustand

Anwender → Roboter

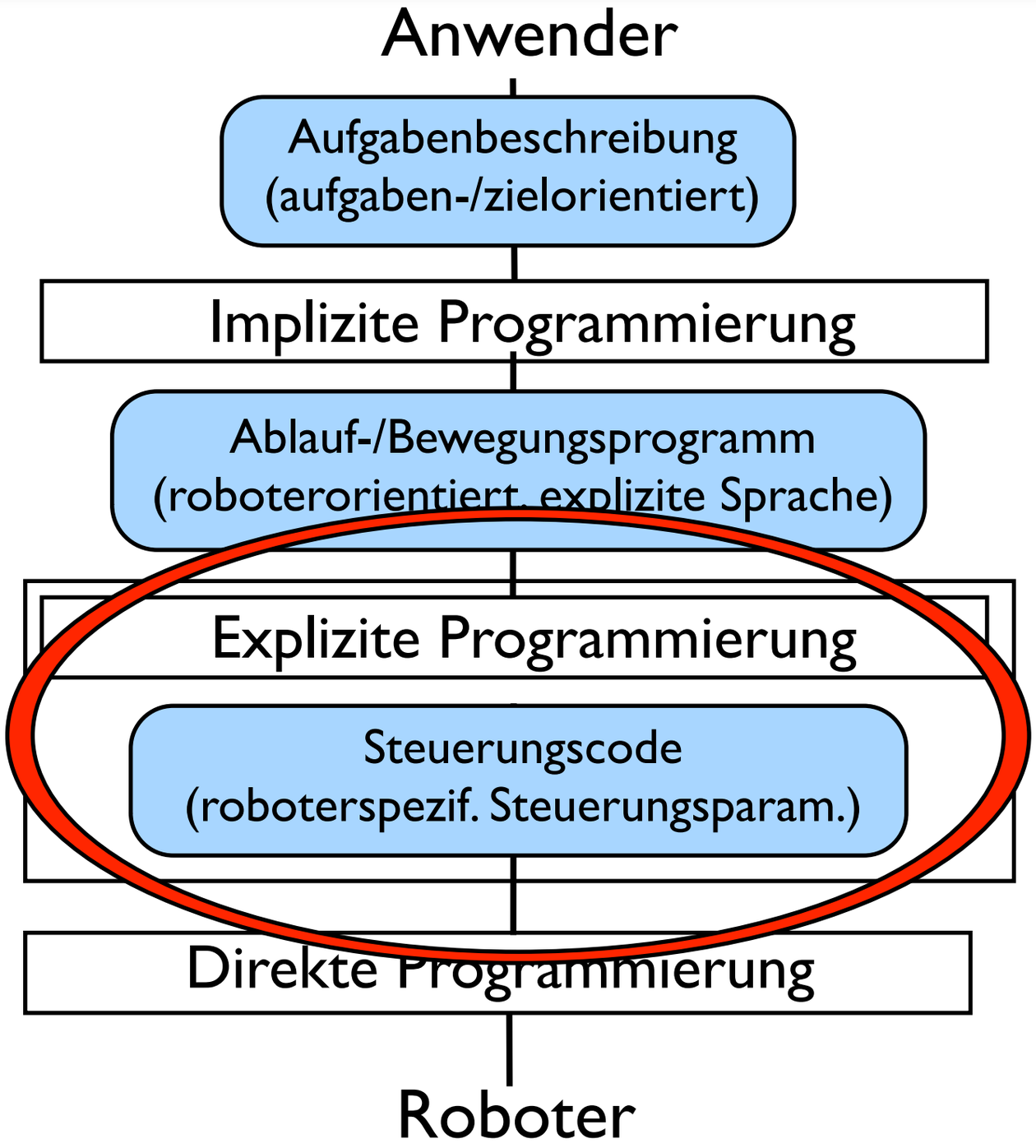


- Aktionsplanungssysteme
- Automatische Roboter-programmiersysteme

- Roboterprogrammierspachen
- Graphische Simulationssysteme

- Teach-In
- Playback-Programmierung

Anwender → Roboter



- Aktionsplanungssysteme
- Automatische Roboter-programmiersysteme

- Roboterprogrammierspachen
- Graphische Simulationssysteme

- Teach-In
- Playback-Programmierung

Einordnung der roboterorientierten Programmierung

Jedes Robotersystem besitzt eine roboterabhängige Steuerungsebene, welche folgende Eigenschaften kapselt:

- Ansteuerung der Hardware (sowohl interne als auch externe)
- Bewegungsaktionen
- Lokale Modelle
- Elementare Operationen (erfordern evtl. Echtzeitregelung)

→ Roboterorientierte Steuerung

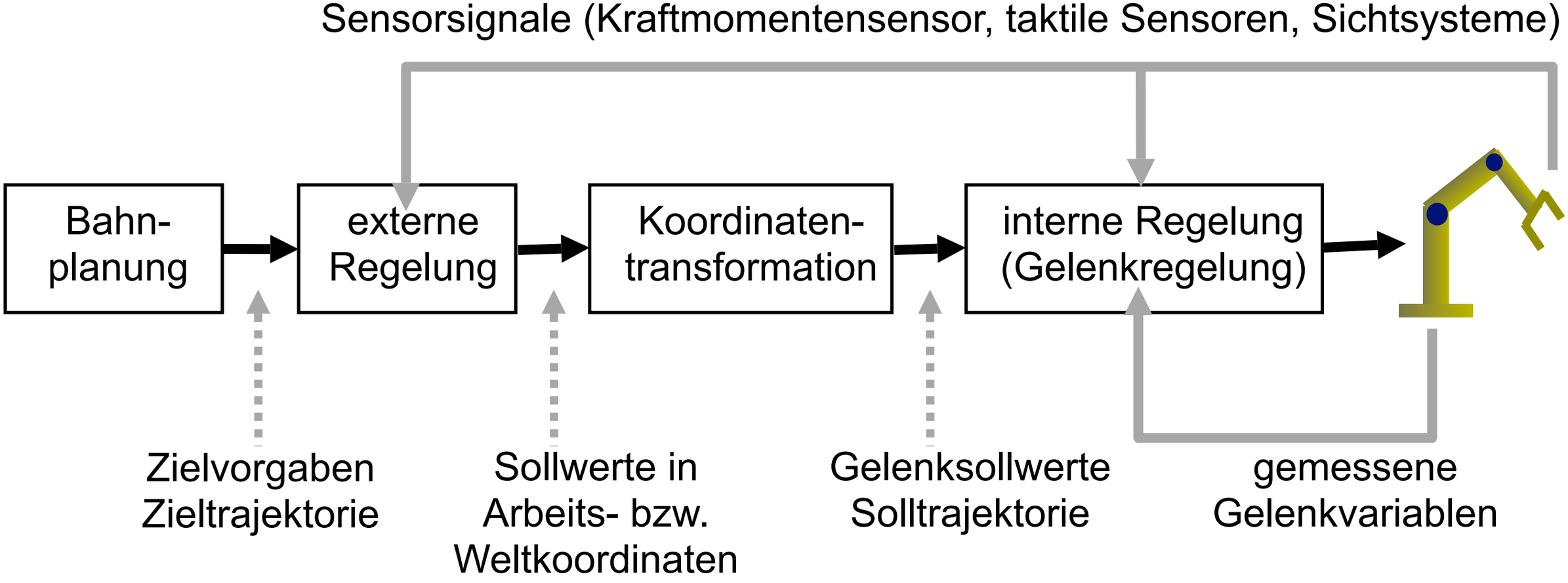
Roboterorientierte Programmierung

Anforderungen:

- Positions-, Geschwindigkeitsregelung der aktiven Komponenten (z.B. Gelenke, Räder)
- Auslesen und Parametrieren der internen und externen Sensorik
- Koordinatentransformationen, direkte und inverse Kinematik
- Sensorabhängige Regelung
- Verfahren auf Trajektorien
- Generierung und Zusammensetzen von Trajektorien zu komplexen Bewegungen
- Verkettung komplexer Bewegungen zu Elementaroperationen

Komponenten der roboterorientierten Programmierung

Regelungszyklus eines Roboters



Sprachelemente von Roboterprogrammiersprachen

Befehle für :

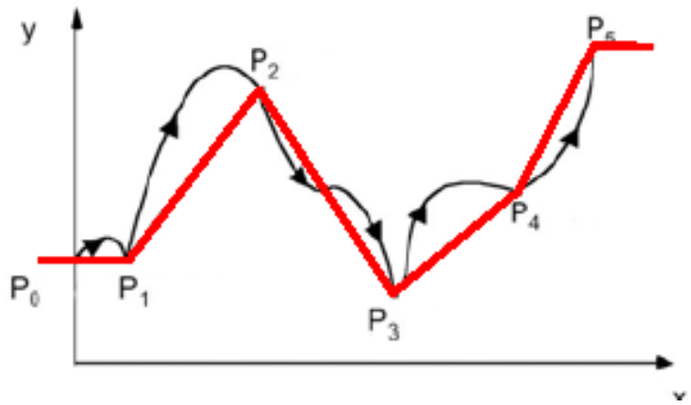
- Bewegung eines oder mehrerer Roboter
- Betrieb von Greifern / Werkzeugen
- Ein-/Ausgabe von Daten / Signalen über Schnittstellen
- Externe Sensoren
- Zur Synchronisation / Kommunikation zwischen Prozessen
- Parallelverarbeitung
- Zur logischen Verkettung von Koordinatensystemen

Anweisungen zur Ablaufsteuerung

Definition generischer Operationen (zB. Armbewegung + Griff → ein Operator)

Bewegungsanweisungen

- Bewegungen im Gelenkwinkelraum
  - Bewege alle Gelenke mit max. Geschwindigkeit
  - Regelung der Geschwindigkeiten mit gleichzeitiger Beendigung der Bewegung aller Gelenke



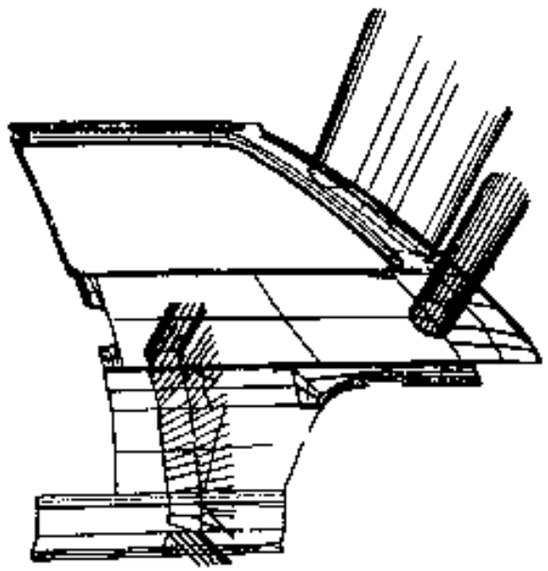
Vorteile:

+ Hohe Einstellgenauigkeit der Position
+ Hohe Wiederholgenauigkeit
+ eindeutige Roboterkonfiguration
+ keine inverse Kinematik erforderlich

Nachteile:

- Abhängigkeit von Robotertyp
- kein Bezug der Gelenkwinkel zur Objektlage

- Kartesische Bewegungen (Stellung des TCPs)
  - Erfordert inverse Kinematik
  - Nutzung von Frames, z.B. relativ zu einem Objekt



- Geometriebezogene Bahndefinition

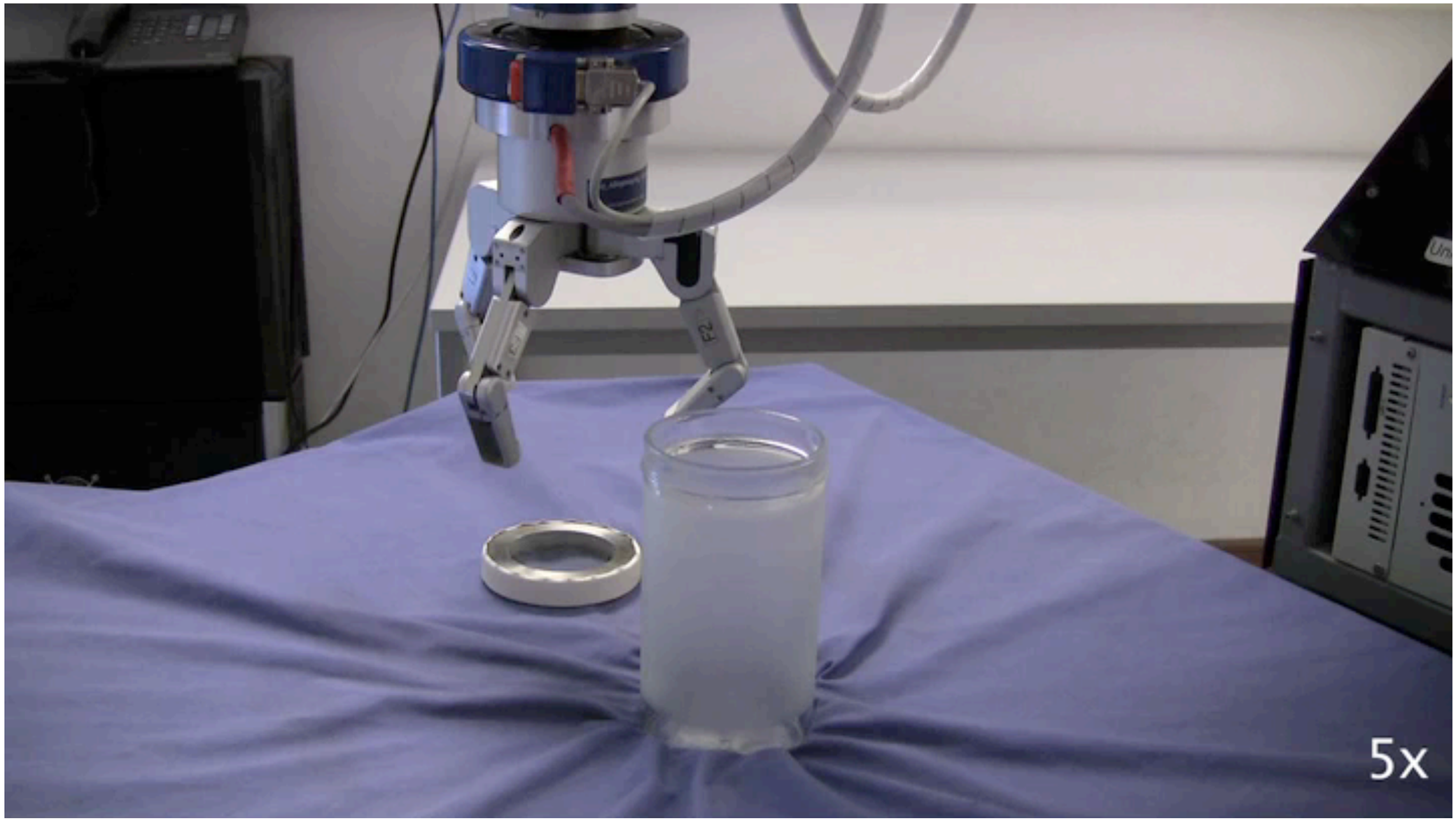
Sprachelemente: Semantik von Greiferbefehlen

- Verschiedene Greifertypen: evtl. mit taktiler und/oder Kraftsensorik
- Backengreifer: Industrie, Forschung

Backengreifer @ PR2, WillowGarage

Sprachelemente: Semantik von Greiferbefehlen

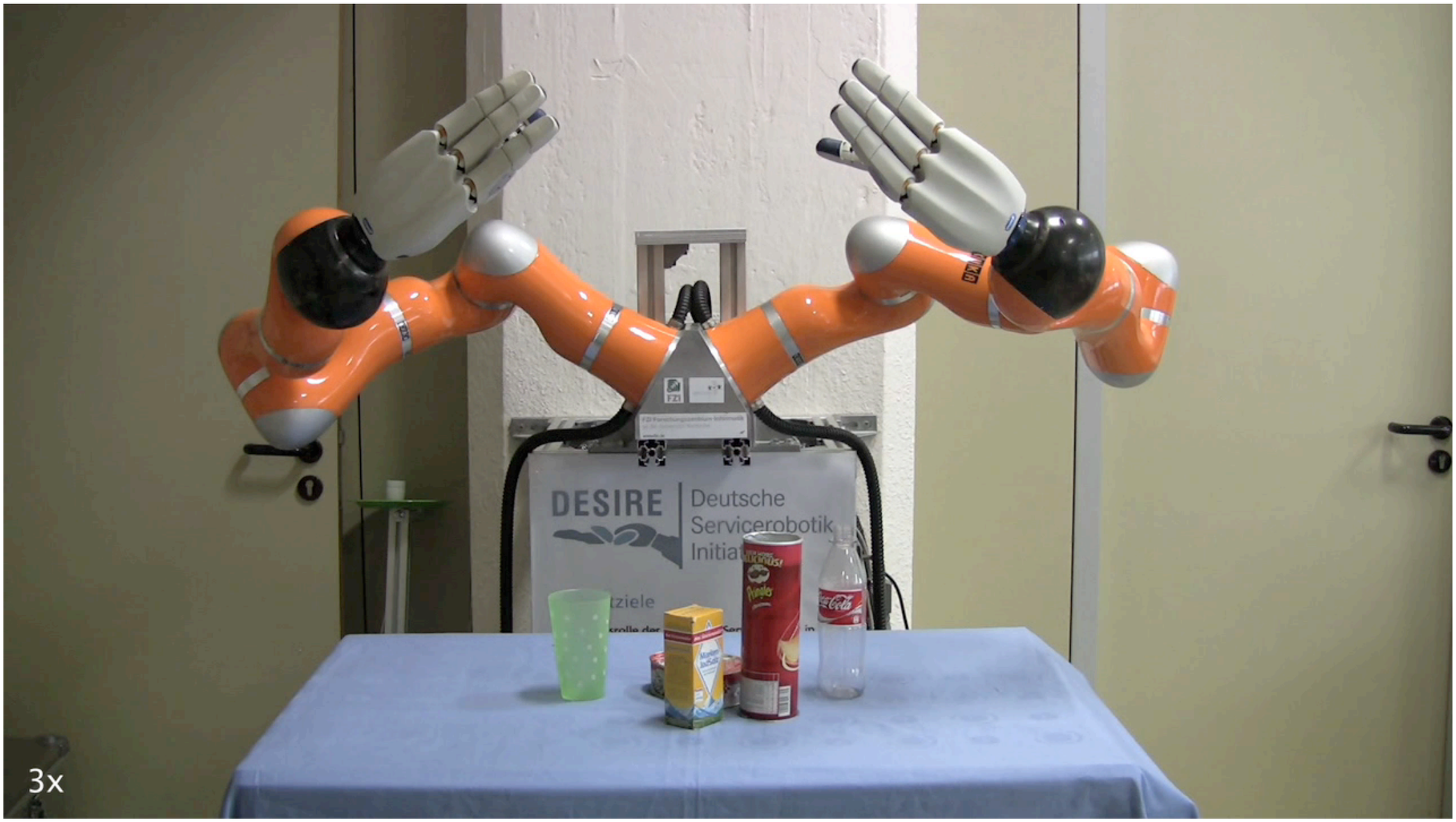
- 3-Finger-Hand: Industrie, Forschung



Barrett-Hand @ Albert II, HIS / FZI, Dillmann

Beispiel komplexe Manipulation

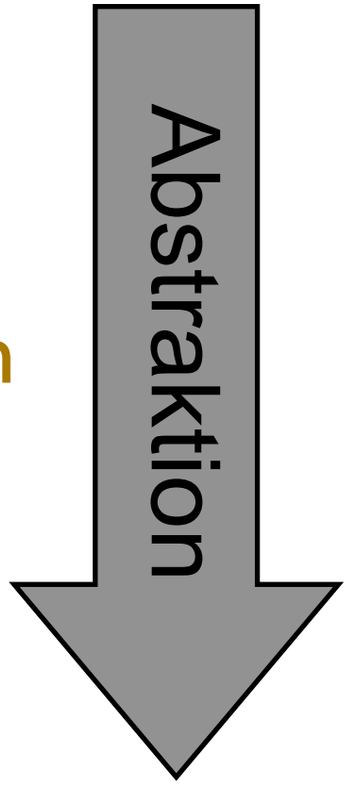
- 4/5-Finger-Hand: Forschung



Schunk Anthropomorphe Hand @ Adero, HIS / FZI, Dillmann

Semantik von Greiferbefehlen

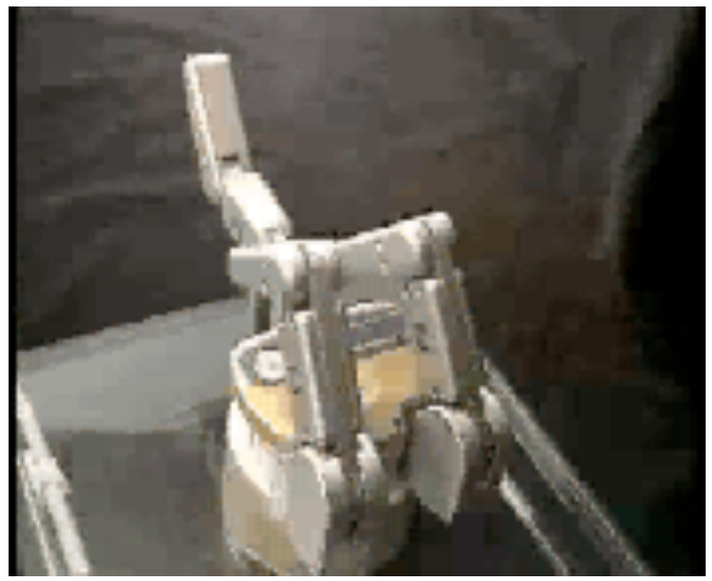
1. Steuerung im Gelenkwinkel-Raum
2. Steuerung im kartesischen/ zylindrischen/… Raum
3. Semantische Steuerung



Semantik von Greiferbefehlen

1. Steuerung im Gelenkwinkel-Raum

- Anzahl der Freiheitsgrade bestimmt Anzahl der Parameter (Backengreifer: 1, menschl. Hand: 22)
- Wenig Kapselungs-Aufwand (keine inverse Kinematik nötig)
- Hand-abhängig
- Kein Bezug zwischen Finger-stellung und Handstellung



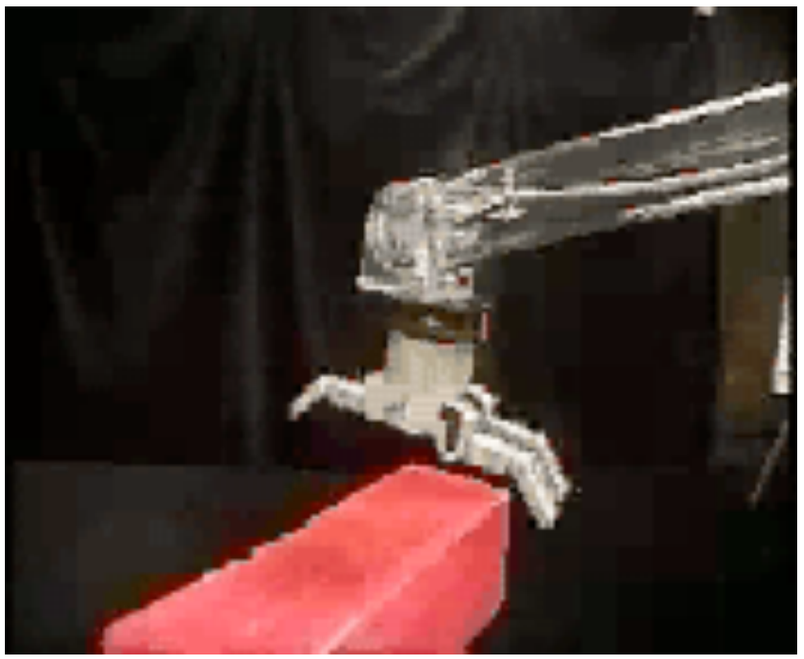
Bsp: BarrettHand(Spread: 0, F1: 15000, F2: 12000, F3: 15000);

Semantik von Greiferbefehlen

2. Steuerung im kartesischen/ zylindrischen/… Raum

- Parameter: Position/Orientierung jedes Fingers/ jeder Fingerspitze
- Inverse Kinematik nötig
- Berechnung aus Greif-/ Bewegungsplanung oder aus menschlicher Vorführung
- Mögliche Konfigurationen (Konfigurationsraum) handabhängig

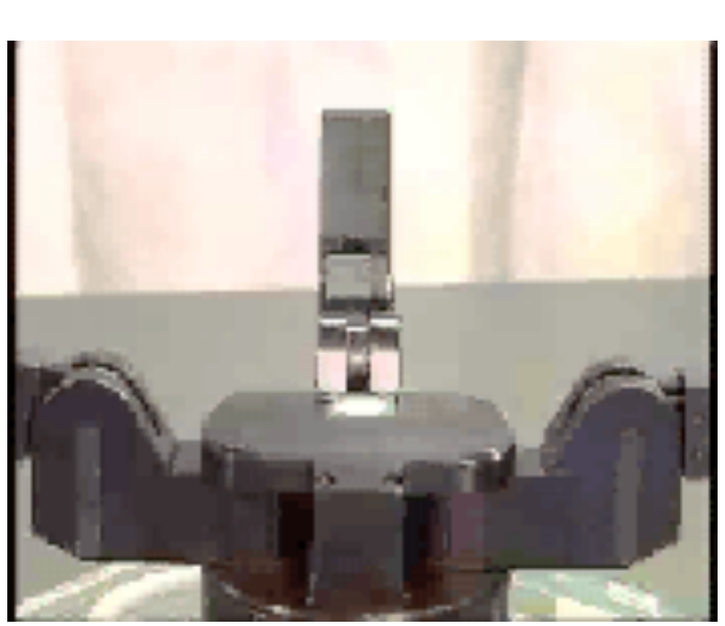
Bsp: BarrettHand(F1: [5,5,0],
F2: [5,0,5],
F3: [0,0,5]);



Semantik von Greiferbefehlen

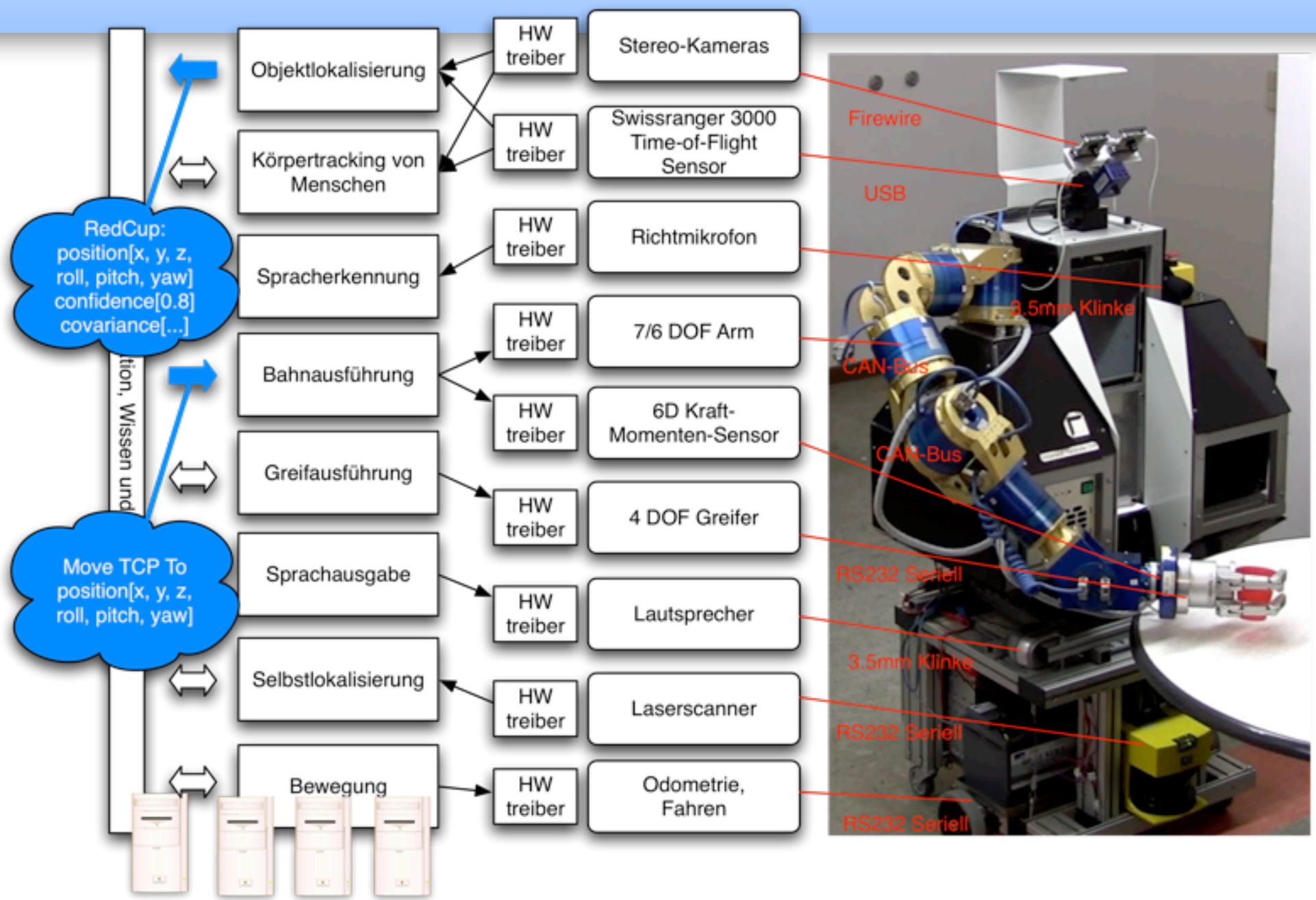
3. Semantische Steuerung

- Parameter: Griff-Form, zu greifendes Objekt, Objektgröße, Objektform, …
- Mapping auf Roboterhand nötig
- Übertragbar auf andere Roboterhände (mögliche Griffe handabhängig)
- Griff-Form vom Menschen lernbar



Bsp: BarrettHand(circular_grasp, 12cm);

Ein-/Ausgabe von Signalen über Schnittstellen

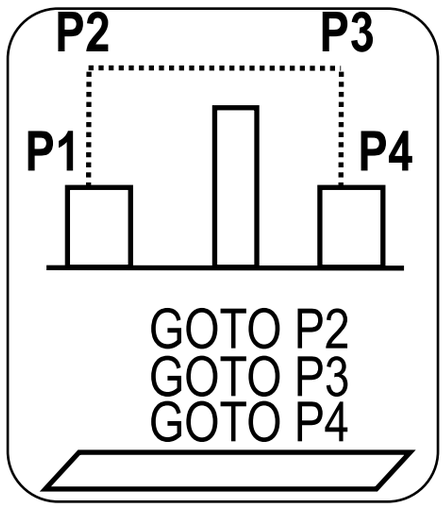


Für die höheren Schichten wird der Roboter als Softwareschnittstelle abgebildet:

Kollektion von Skills: Perzeption und Aktorik = Daten lesen/schreiben

Explizite Programmierung

- Nur in Verbindung mit (abstrakten) Programmiersprachen



Vorteile:

+ beliebig komplexe Bahnen
+ Anbindung von Sensoren
+ reaktive Planung

Nachteile:

- Keine standardisierte Programmiersprache
- Kenntnis der Programmiersprache

Ausblick: nächste Vorlesung

- Umweltmodellierung
- Aufgabenmodellierung

Literatur Robotik II

- R. Dillmann, M. Huck „Informationsverarbeitung in der Robotik“. Springer, 1991.
- Springer Handbook of Robotics; Siciliano, Khatib; 2008, Part A & D
- S. Russel, P. Norvig Artificial Intelligence: A Modern Approach 2nd Edition; 2003, Kapitel 16 & 17
- Probabilistic Robotics; S. Thrun, W. Burgard, D. Fox; 2005, Kapitel 14
- Planning Algorithms, S. M. LaValle; 2006, Kapitel III

- Dissertationen Jäkel, Schmidt-Rohr (PDFs online bei KIT Digitale Bib.)