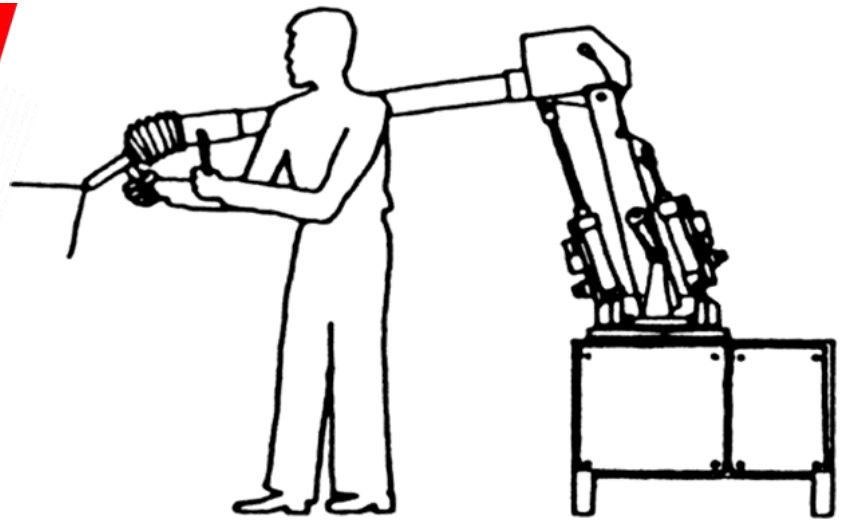
RRLAB
THE ROBOTICS RESEARCH LAB

Roboterprogrammierung



Prof. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

Inhalt

- Programmierung von Industrierobotern
- Online/Offline-Verfahren
- Programmierarten
- Umweltmodellierung

- Foliensatz z.T. von
  - Dr. R. Lafrenz, Universität Stuttgart
  - Prof. Zühlke PAK, TU KL

Programmierung von Industrierobotern

- Muss frei programmierbar sein
- Folge von anzufahrenden Punkten
- Punktfolge beliebig oft anfahrbar
- Freie Wahl der Punkte eingeschränkt durch …
  - Hindernisse
  - Konstruktive Beschränkungen des Roboters

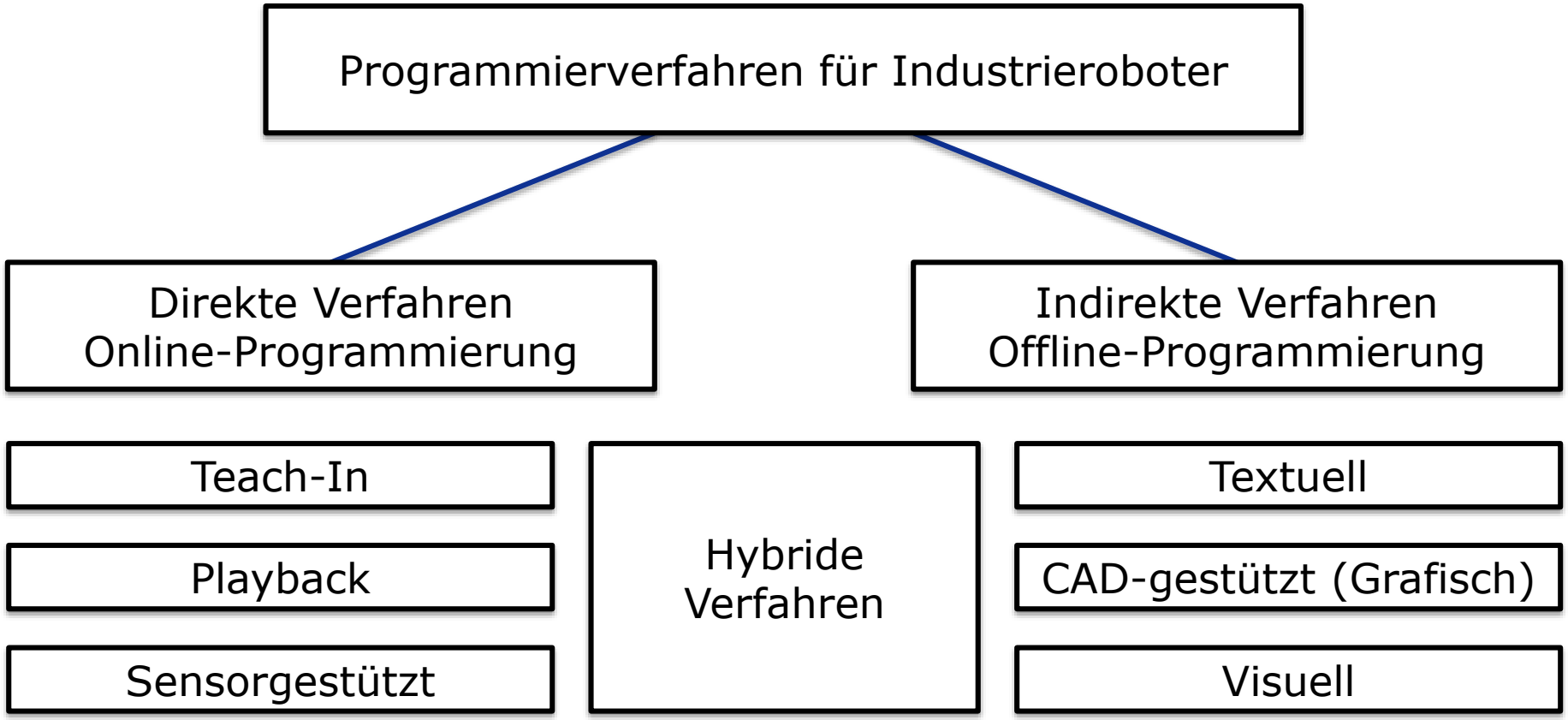
Komponenten der Programmierung

- Betriebssystem
  - Echtzeitfähig
  - Schnittstelle zur Robotersteuerung
- Programmiersprache
  - Roboterspezifische Sprachen (VAL, …)
- Bibliotheken für Standardsprachen (RCCL für C, …)

Komponenten der Programmierung

- Roboterorientierte Routinen
  - Besondere Datentypen (Matrizen)
  - Kinematik- und Dynamik-Routinen
  - Bewegungsbefehle (kartesisch, Gelenkraum)
  - Effektorbefehle
- Aufgabenorientierte Routinen
  - Wissensbasis mit Umweltmodell
  - Regelbasis zur Aufgabenzerlegung
  - Planungsalgorithmen
  - Erteilung komplexer Aufgaben

Überblick Programmierverfahren

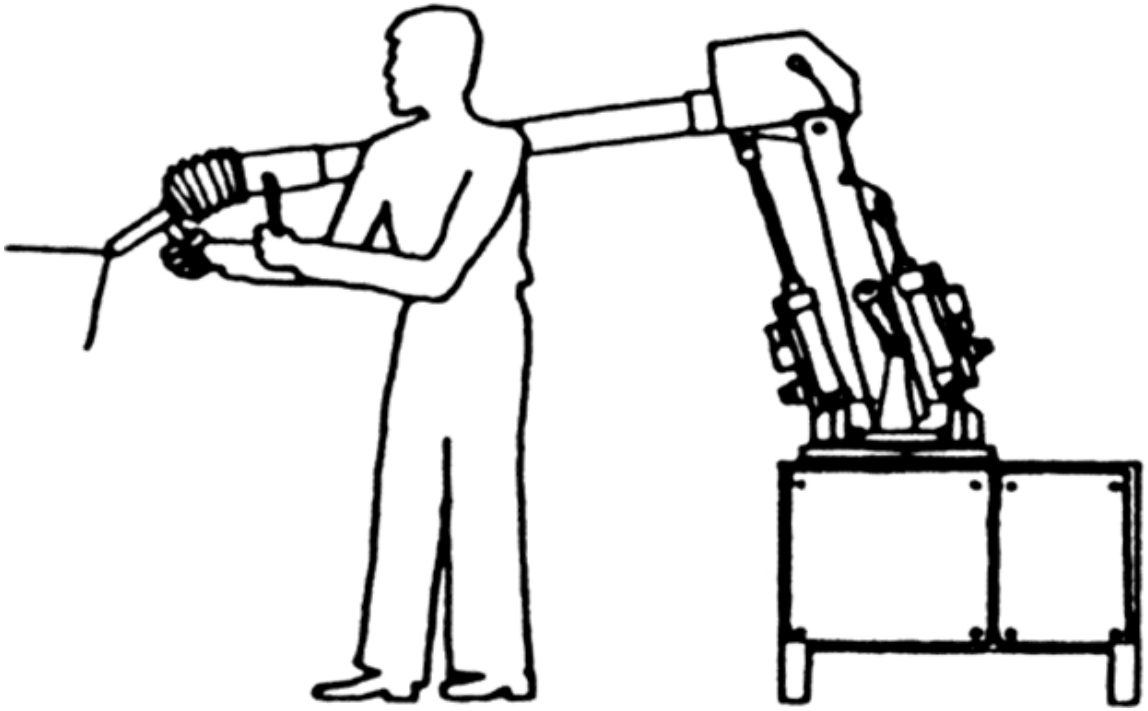


Online-Verfahren: Teach-In

- Positionierung und Konfiguration des Roboters mit speziellen Steuerungsgeräten
- Steuerungsgeräte
  - Teachbox
  - Joystick
  - Maus
  - Teach-Kugel
- Anwendungen
  - Punkt-zu-Punkt-Steuerung
  - Mehrpunkt-Steuerung (MP)

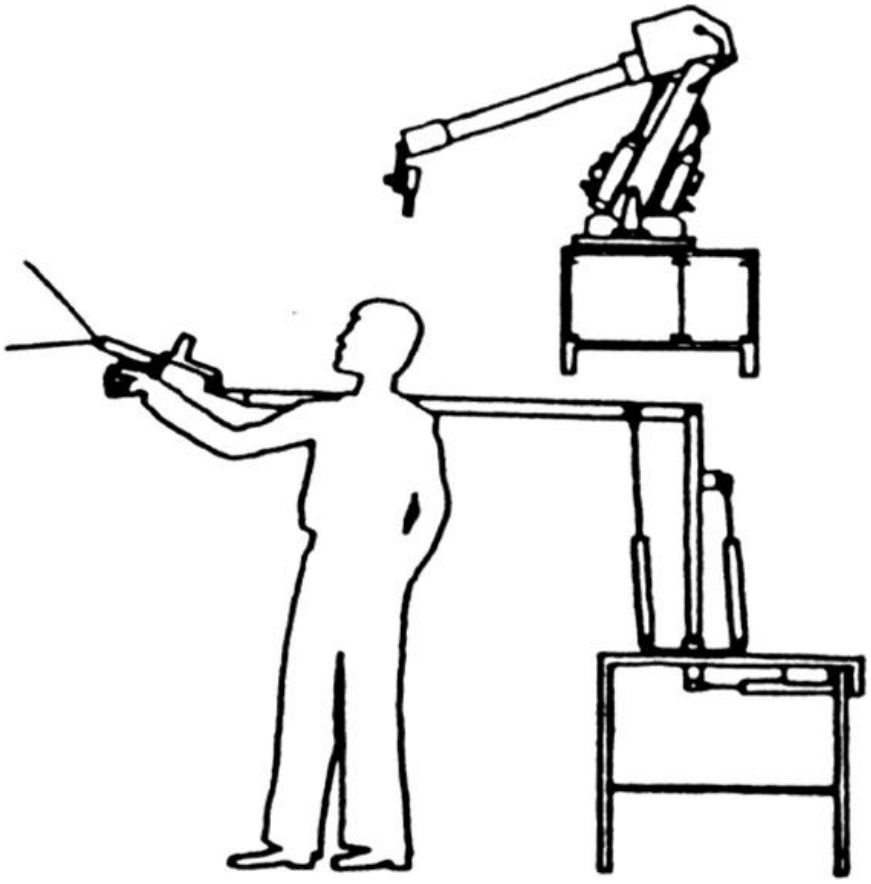
Online Verfahren: Playback, manuelle Führung

- Manuelle Führung des schwerkraftfreien Roboters
- Führung auch im zero-gravity-mode schwierig
- Heute nur noch bei anthropomorphen Roboterarmen



Online Verfahren: Playback, Master-Slave

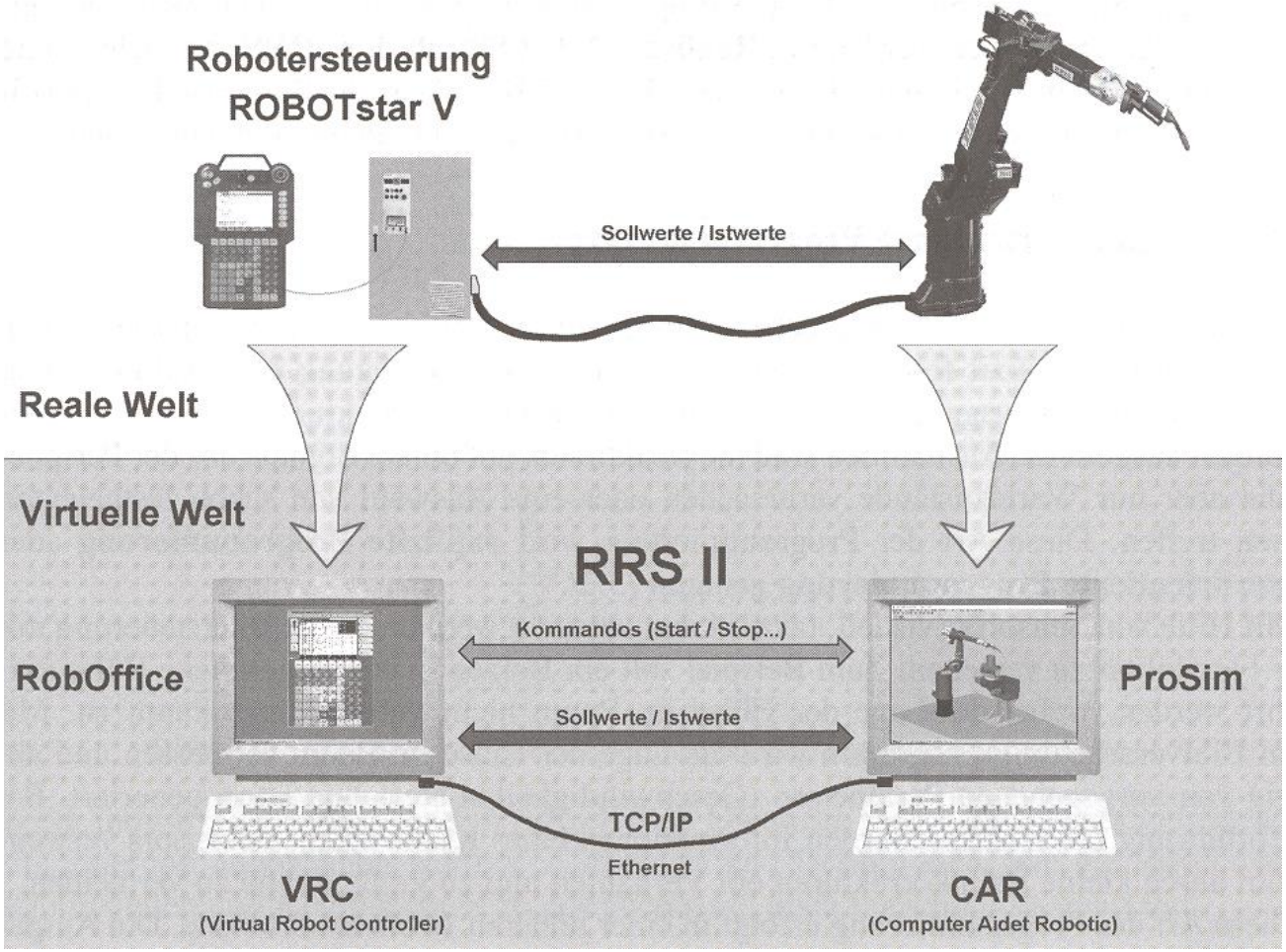
- Master-Slave-Systeme mit möglichst ident. Kinematik
- Manuelle Führung des Masters Einsatz nur bei Teleoperation
- Rückmeldung von Kräften (virtuelle Realität)
- Übertragungsverzögerungen
- Teuer wegen Master-System



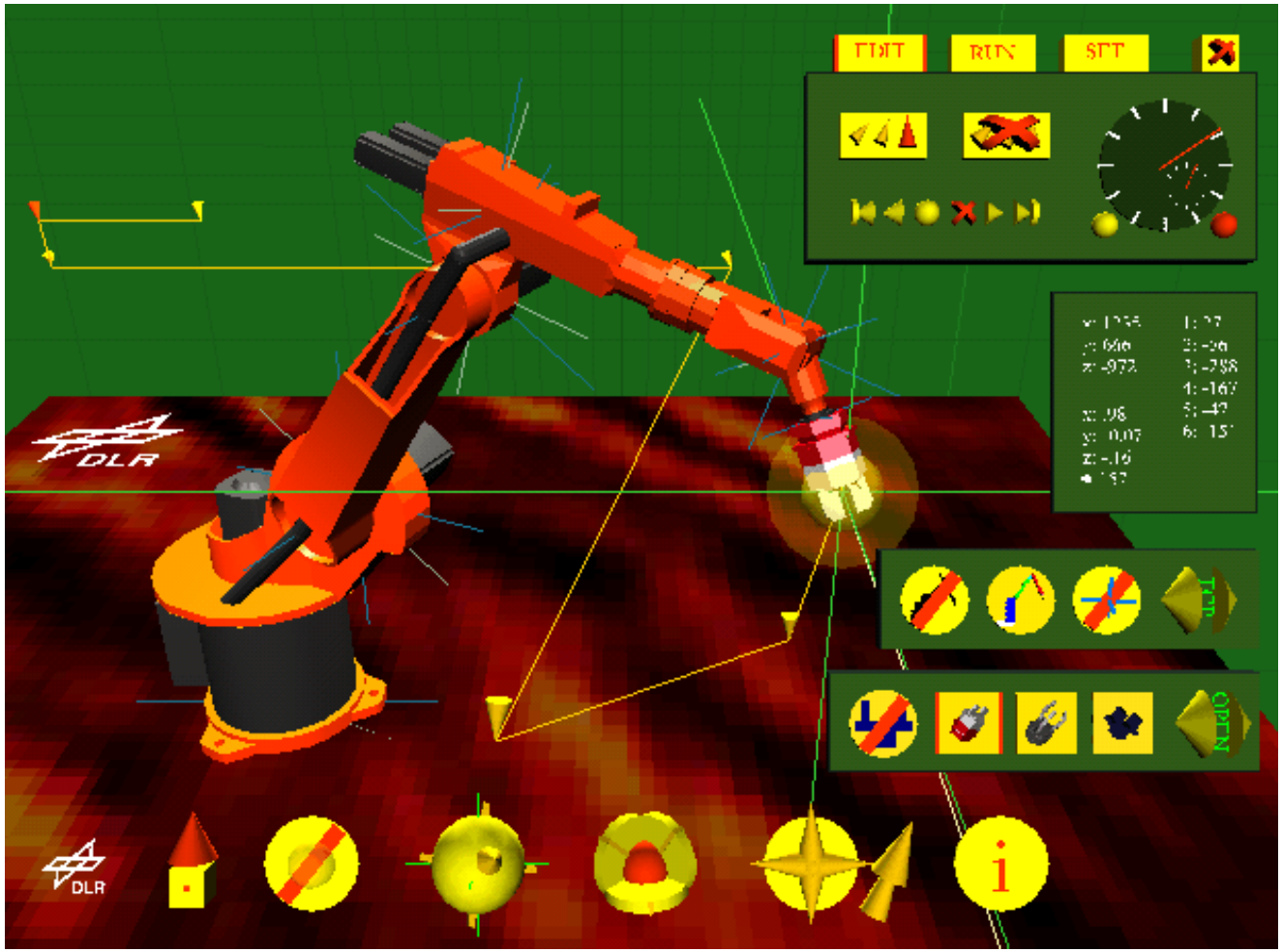
Offline Verfahren: CAD Robotersimulation

- Softwaresimulation (Kinematik und Dynamik) von Roboterzelle und Roboter
- Erstellung der Simulation aufwändig
- Vergleichsweise billige Optimierung von Bewegungsabläufen
- Gefahr von falscher bzw. unvollständiger Simulation
  - Kinematische und dynamische Parameter müssen möglichst exakt sein, da sonst Schäden am Roboter möglich sind

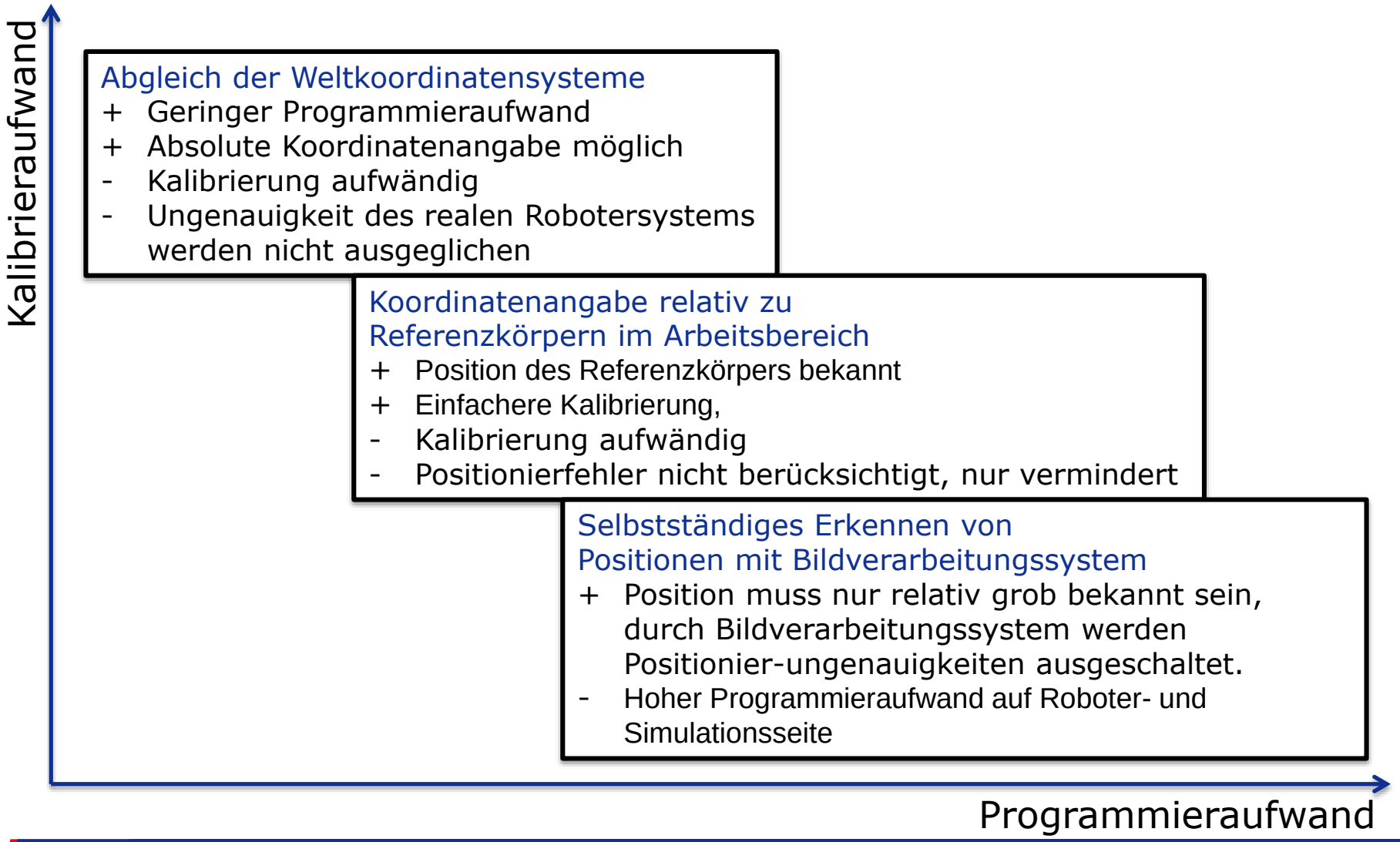
Robotersimulation: Realistic Robot Simulation



Robotersimulation: VRML 2.0 DLR und KUKA



Abgleich zwischen Simulation und Roboter



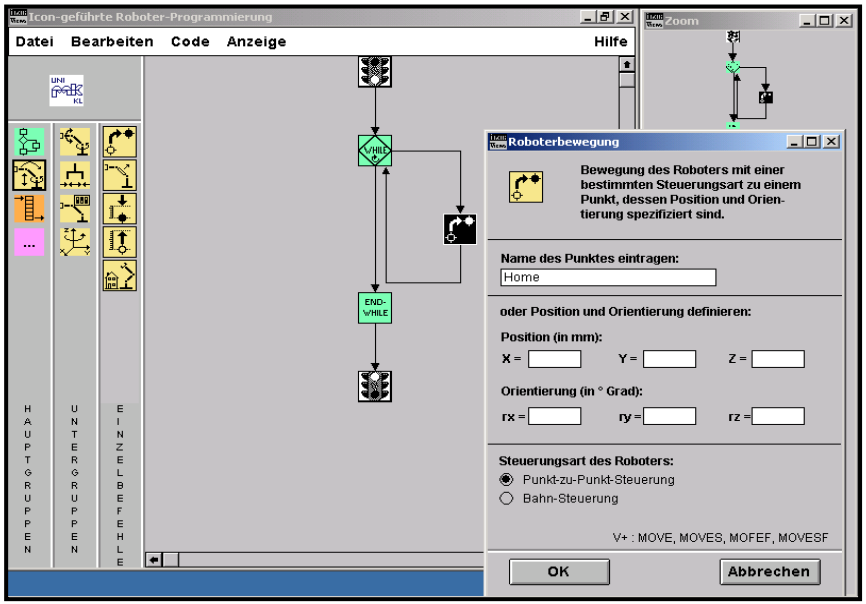
Abgleich der Weltkoordinatensysteme
+ Geringer Programmieraufwand
+ Absolute Koordinatenangabe möglich
- Kalibrierung aufwändig
- Ungenauigkeit des realen Robotersystems werden nicht ausgeglichen

Koordinatenangabe relativ zu Referenzkörpern im Arbeitsbereich
+ Position des Referenzkörpers bekannt
+ Einfachere Kalibrierung,
- Kalibrierung aufwändig
- Positionierfehler nicht berücksichtigt, nur vermindert

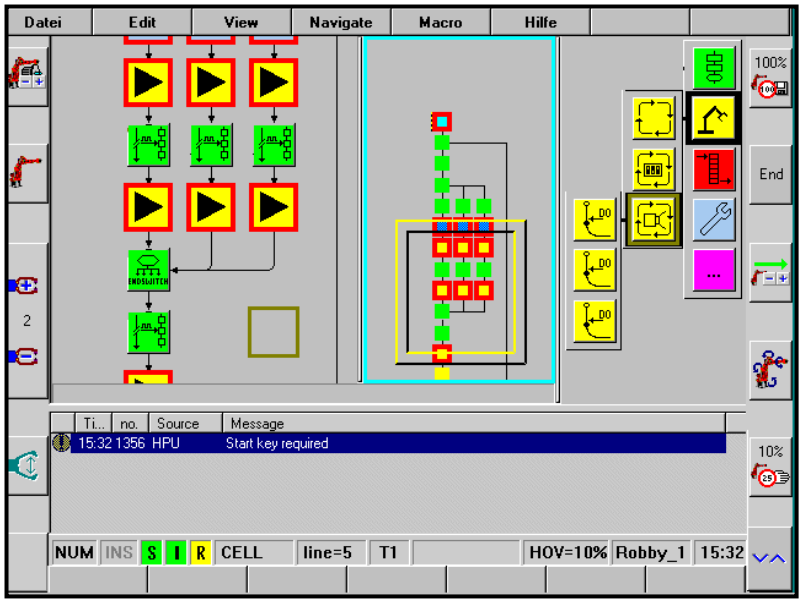
Selbstständiges Erkennen von Positionen mit Bildverarbeitungssystem
+ Position muss nur relativ grob bekannt sein, durch Bildverarbeitungssystem werden Positionier-ungenauigkeiten ausgeschaltet.
- Hoher Programmieraufwand auf Roboter- und Simulationsseite

Visuelle Roboterprogrammierung

- Programme mit zwei- oder höherdimensionierte Strukturen
- Elemente: Grafiken, Diagramme, Ikonen, Animationen



Grafische Roboterprogrammierung, Prototyp des PAK



Umsetzung des PAK-Prototyps für die Verwendung mit dem KUKA-Handbediengerät

Zweite Klassifizierung der Programmierung

- Programmierung durch Beispiele
  - Einstellen des Roboters
  - Manuelle Programmierung
  - Teach-In-Programmierung
  - Master-Slave Programmierung
- Programmierung durch Training
- (Textuelle) Programmierung
  - Roboterorientiert
  - Aufgabenorientiert

Programmierung durch Beispiele: Einstellen

- Diskrete Stellungen, keine kontinuierliche Regelung
- Gelenkeinstellung mit mechanischen Schaltern und Stoppern
- Aufgabe der Robotersteuerung: Signale an Stellglieder senden, damit zum Zielzeitpunkt Stopperstellung aktiv wird
- Nur kleine Anfahrpunktmenge kann zu Programm zusammengefasst werden
- Freie Programmierung stark eingeschränkt

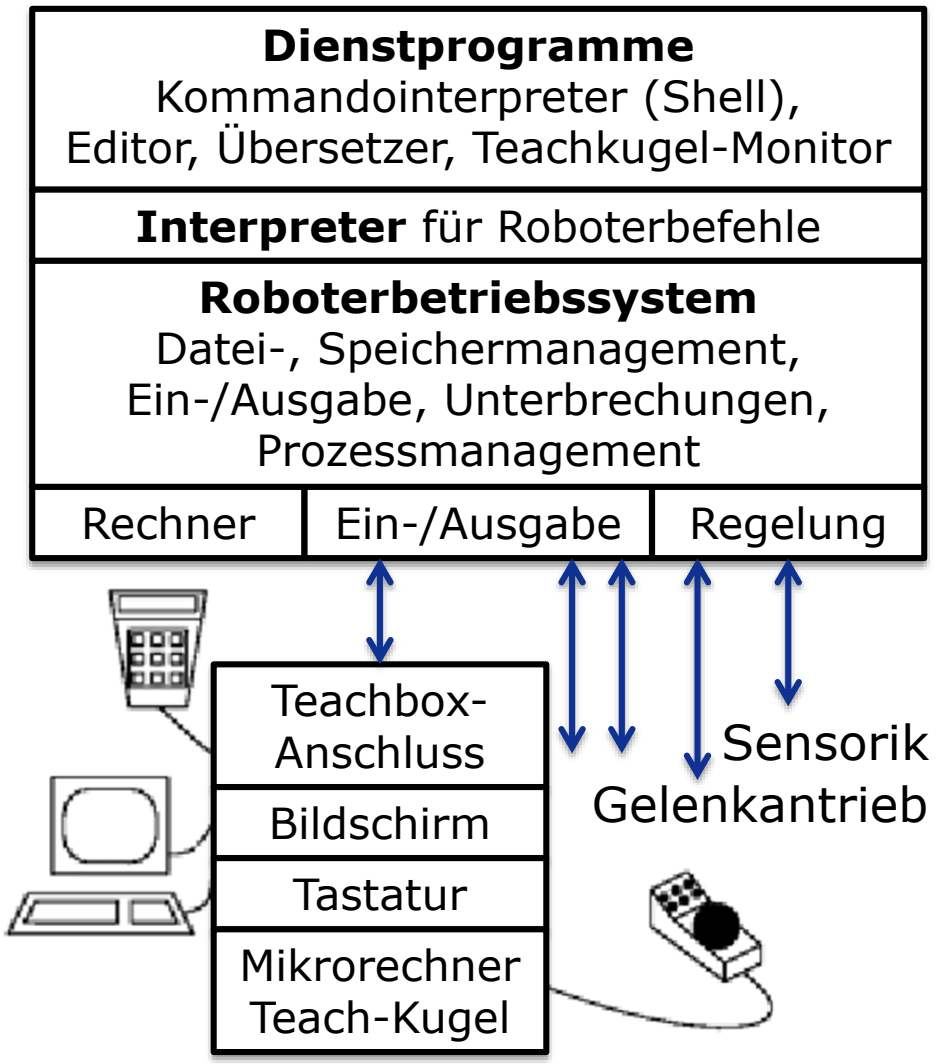
Programmierung durch Beispiele: Manuell

- Gelenkmotoren und Bremsen so eingestellt, dass Roboter manuell bewegt werden kann
- Effektor von Hand entlang erwünschter Bahn führen
- Bahn durch Folge von Zwischenpunkten definieren
- Ist ein Zielpunkt erreicht, so werden die Gelenkwinkel auf Tastendruck gespeichert
- Probleme
  - Enge Fertigungszellen verhindern Eingriff an bel. Positionen
  - Schwere Roboter
  - Gefährlich
- Heute selten genutzt

Programmierung durch Beispiele: Teach-In

- Spezielles Eingabegerät um Effektor zu positionieren (Teach Box, Teach Pentant, Lehrgerät)
- Drei Möglichkeiten
  - Einzelbewegung der Gelenke
  - Bewegung des Effektors in x-, y-, z-Richtung (Pos.-Einst.)
  - Drehungen um die Winkel O, A, T (Orient.-Einst.)
- Auf Tastendruck …
  - Zielpunkte speichern/löschen
  - Programme starten/abbrechen
  - Geschwindigkeiten einstellen
- Alternative Eingabegeräte
  - Joystick, Maus
  - Teach-Kugel

Programmierung durch Beispiele: Teach-In

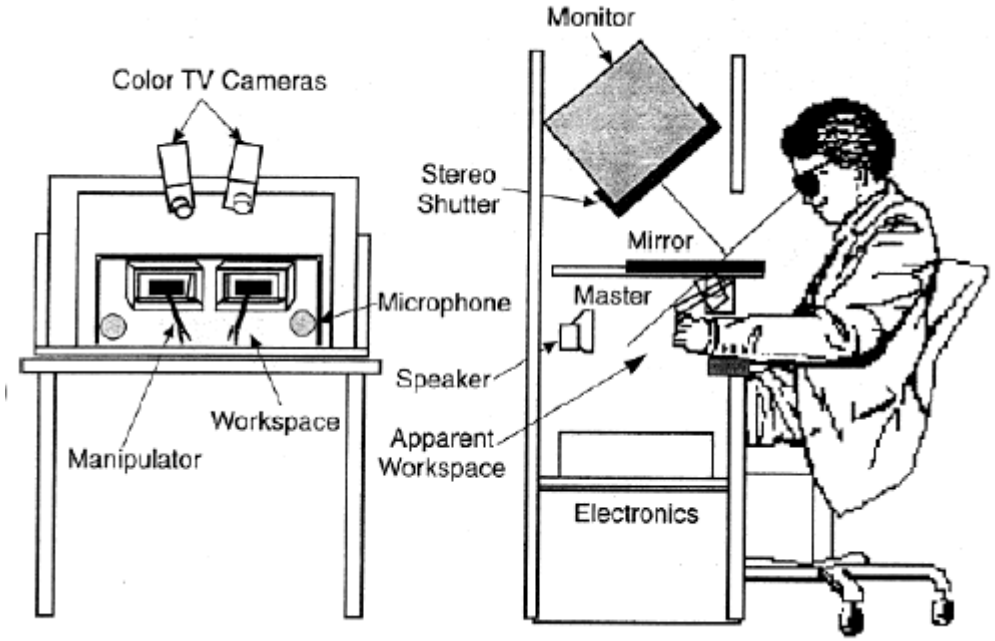


Programmierung durch Beispiele: Master-Slave

- Manuelle Programmierung schwerster Roboter
- Führung des kleinen und leichten Master-Roboters
- Übertragung der Führung an Slave-Roboter
- Teuer, da zwei Roboter benötigt
- (Fast nur) Verwendung in der „Teleoperation“

Programmierung durch Beispiele: Teleoperation

- Einsatz: Aufenthalt für Menschen schwierig/gefährlich
- Einsatzgebiete: Verstrahlte Räume, unter Wasser, Weltraum
- Wie Master-Slave-Programmierung (ohne Zwischenpunkte)
- Situation am Slave wird meist per Kamera übertragen
- Sehr viele Probleme
  - Kamerabildübertragung
  - Kräfte
  - Übertragungszeiten
  - …



Programmierung durch Beispiele: Vor/Nachteile

- Vorteile
  - Keine Programmierkenntnisse erforderlich
  - Keine weiteren Rechner zur Programmierung notwendig
  - Keine Arbeitsraumvermessung
    (WKS-Stellung nicht benötigt)
  - Programmierung erfolgt direkt mit realen Robotern
  - Berücksichtigung aller konstruktiven Ungenauigkeiten
  - Berücksichtigung aller Störgrößen
- Nachteile
  - Einbezug von Sensoren nicht möglich
  - Bahnkorrektur durch Sensorinformation nicht möglich
- Keine Verwendung bei (heutigen) intelligenten Robotern

Programmierung durch Training

- Auszuführende Aktion wird dem Roboter vorgeführt
- Aufnahme der Aktion durch Sensoren
- Roboter wiederholt die Aktion (Training), bis sie Gütekriterien genügt (Genauigkeit, Schnelligkeit, …)
- Externe Sensoren erfassen Abweichung von der Zielvorgabe
- Programmverbesserung durch Korrektur (Selbstanalyse)

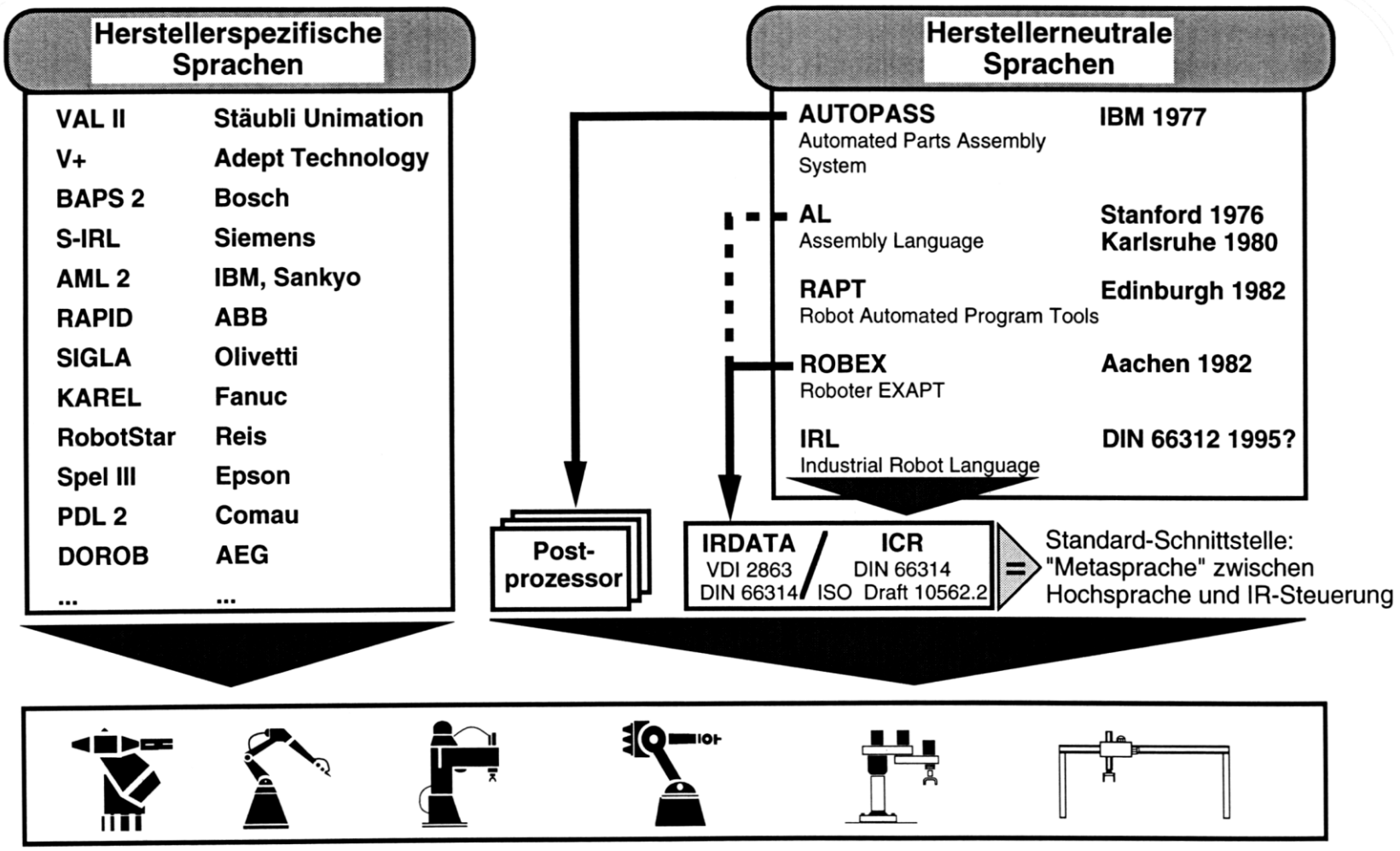
Programmierung durch Training

- Forschungsthema
  - Noch nicht im realen Einsatz
  - Heute einfache Programme möglich
    (Klötzchen umstellen, manuelles Einfügen)
- Bildverarbeitung und Interpretation
  - Objekterkennung
  - Positions- und Orientierungsbestimmung von Objekten
  - Verfolgung bewegter Objekte
- Sensorintegration: Auswertung von vielfältigen Sensoren
- Vorgangsanalyse: Aus beobachteter Aktionen muss Sequenz elementarer Handlungen extrahiert werden

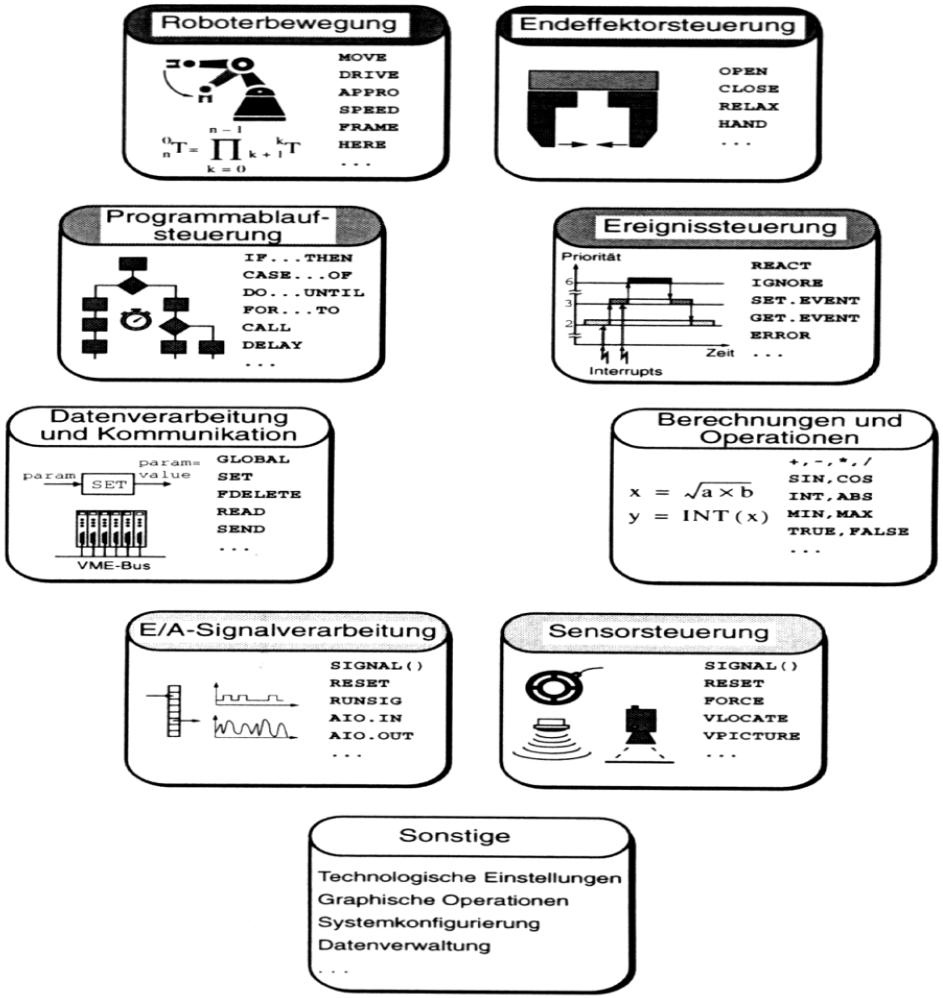
Roboterorientierte Programmierung

- Roboterprogramm mit expliziten Bewegungsbefehlen (z.B.: „Fahre auf gerader Linie zu Punkt B“)
- Textuelle Programmierung in Roboterprogrammiersprache
- Meist Erweiterung von universeller Sprache (z.B. C)
  - Roboterspezifische Datentypen (Transformationsmatrizen, Operatoren)
  - Bewegungsbefehle
  - Effektorbefehle

Roboterorientierte Programmiersprachen

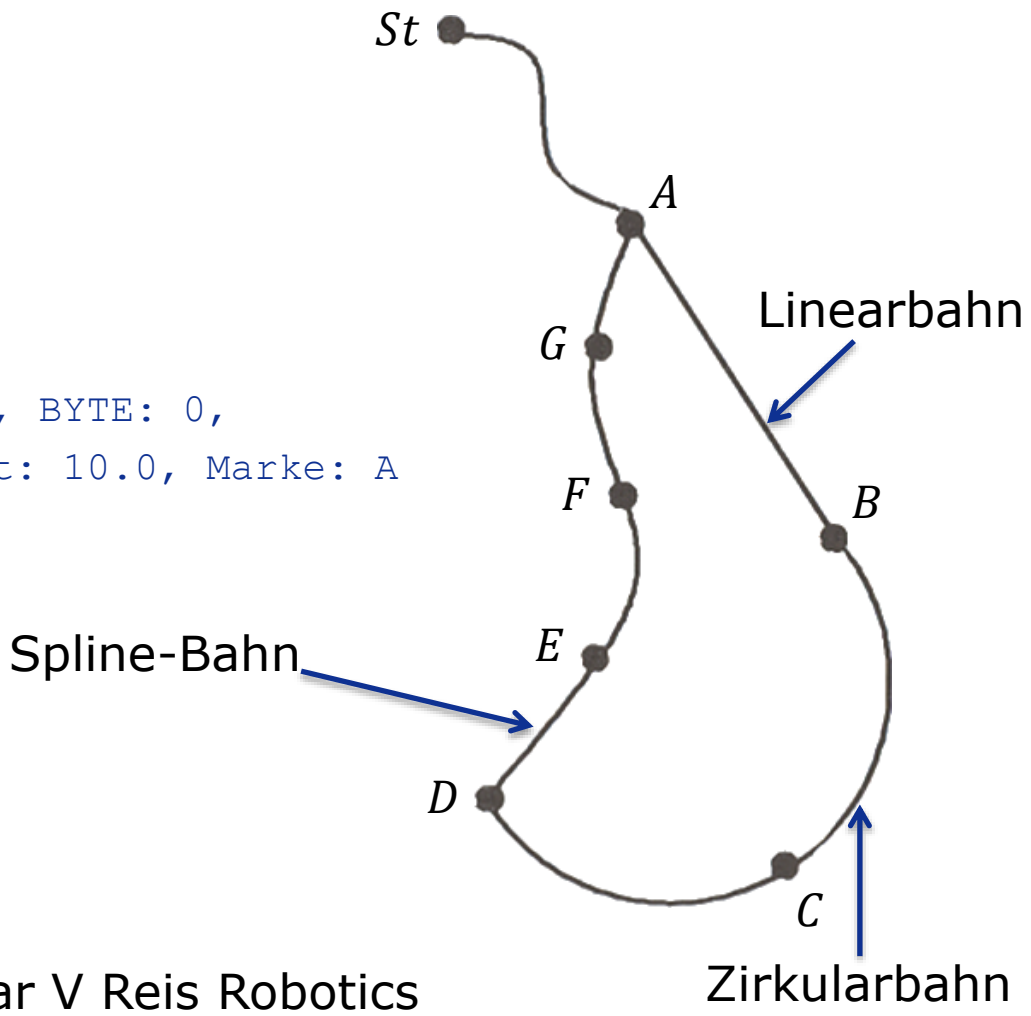


Roboterorientierte Programmierung: V+ (Adept)



Roboterorientierte Programmierung: ROBOTstar

```
   Werkzeugvariable T
   Position     A
   BEWEG_ART    #LINEAR
   BAHN_GESCHW  150
   BAHN_BESCHL  60
   POSITION     B
   BEWEG_ART    PTP
   WARTE_BIT    #EINGANG, Pegel: 1, BYTE: 0,
                Bit_Nr: 5, Max_Zeit: 10.0, Marke: A
   BEWEG_ART    ZIRK
   POSITION     C
   POSITION     D
   BEWEG_ART    #SPLINE
   POSITION     E
   POSITION     F
   POSITION     G
A: POSITION     A
```



Programmierbeispiel zu ROBOTstar V Reis Robotics

Roboterorientierte Programmierung: SRCL

Textuelle Programmierung mit Siemens Robot Control Language

DEF HP 5	Mit DEF-Anweisung Hauptprogramm einrichten (evtl. andere HP-Nr.)
GES BAN 20	Bahngeschwindigkeit setzen, z.B. 20 m/min
GES ALL 50	Mit 50% der maximalen Achsgeschwindigkeit fahren
PTP X1 Y1 Z1	Mit Handverfahrtasten Roboter in
A1 B1 C1	Warteposition fahren und Position teachen
GRF 1 AUF	Greifer öffnen
WRT E1 H	Auf Teil warten (High-Signal an Eingang 1)
PTP X2 Y2 Z2	Teil anfahren
A2 B2 C2	
GRF 1 ZU	Teil greifen
PTP X3 Y3 Z3	Werkzeugmaschine anfahren

Roboterorientierte Programmierung: SRCL

```
LIN  X4  Y4  Z4      Teil einlegen
     A4  84  C4
GRF  1   AUF         Greifer öffnen
LIN  X5  Y5  Z5      Arm aus Maschine herausfahren
     A5  B5  C5
S    IA  1           Maschine starten (Impulsausgang 1 setzen)
WRT  E2  H           Auf Fertigmeldung warten (Eingang 2: (High)-Signal)
LIN  X4  Y4  24      Fertigteil anfahren
     A4  B4  C4
GRF  1   ZU          Fertigteil greifen
LIN  X5  Y5  Z5      Arm aus Maschine herausfahren
     A5  B5  C5
PTP  X6  Y6  Z6      Ablage anfahren
     A6  B6  C6
GRF  1   AUF         Teil ablegen
END  HP  1           Programmende
```

Roboterorientierte ...: Realisierungsvarianten

- Vollständiger Neuentwurf der Sprache
  - Frei von Sachzwängen und Implementierungsdetails
  - Vermeidung bekannter Schwachpunkte
  - Reichhaltige, roboterorientierte Datentypen
  - AL, VAL, VAL II (Unimation, Puma-Roboter)

Code	Definition
`X.TO.Y`	Name of program
`Open`	Open robot gripper as it approaches point X
`APPRO X,25`	Approach point X within $25mm$
`MOVE X`	Move to point X
`CLOSEI`	Close jaws immediately
`DEPART 25`	Back away from point X $25mm$
`APPRO Y,25`	Approach point Y within $25mm$
`MOVE Y`	Move to point Y
`OPENI`	Open robot gripper immediately
`DEPART 25`	Back away from point $25mm$

Roboterorientierte ...: Realisierungsvarianten

- Weiterentwicklung von Automatisierungs-/Steuerungssprache
  - Weiterentwicklung vorhandener NC-Sprache einfach
  - Leichte Übernahme von vorhandenen Programmen
  - RAPT aus APT für NC-Maschinen, ROBEX aus EXAPT
- Erweiterung allgemeiner Programmiersprache um roboterorientierte Sprachelemente
  - Erweiterung einfacher als Neuentwicklung
  - Verwendung existierender Bibliotheken
  - AUTOPASS eingebettet in PL/1, PASRO in Pascal

Roboterorientierte Progr.: Sprachelemente

- Befehle
  - Für Bewegung eines/mehrerer Roboter
  - Für Betrieb von Greifern/Werkzeugen
  - Für externe Sensoren
  - Zur Ein-/Ausgabe von Daten/Signalen über Schnittstellen
  - Zur Synchronisation/Kommunikation mit Prozessen
  - Zur Parallelverarbeitung
  - Zur Unterbrechungsbehandlung
  - Zur logischen Verkettung von KS
- Anweisungen
  - Zur Berechnung von Ausdrücken
  - Zur Ablaufsteuerung

Roboterorientierte Progr.: Sprachelemente

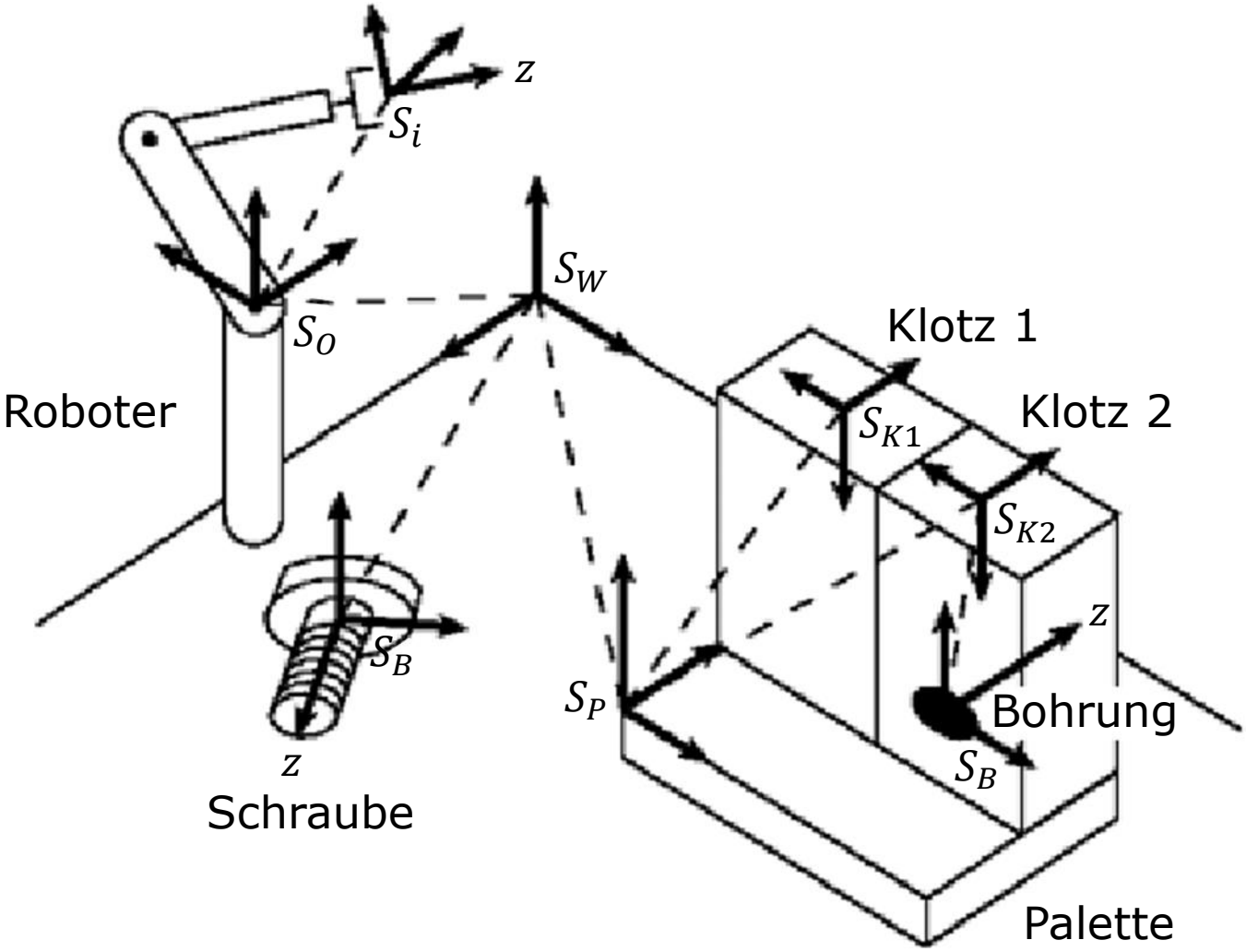
- Echtzeitverarbeitung mit Periode, Dauer und Deadline
- Prozedurkonzept
- Konstruktor- und Selektorbefehle für komplexe, strukturierte Datentypen
- Definition von generischen Operationen durch den Benutzer

Idealfall	Realität
Alle Sprachelemente unterstützt	Nicht alle Sprachelemente in Roboterprogrammiersprachen

Roboterorientierte Progr.: Bsp. für Roboterwelt

- Zulassung von kartesischen Stellungen für Anfahrpunkte
- Vorteile
  - Kenntnis über roboterspezifische Kinematiken auf Gelenkebene nicht benötigt
  - Lesbare Programme
  - Kartesische Stellungen einfacher als Gelenkwinkelangaben
  - Direkte Umsetzung von Konstruktionsdaten in Programme
  - Leichte Übertragung von Programmen auf andere Roboter
  - Art und Anzahl der Gelenke vor Programmierer verborgen
- Nachteile
  - Kartesische Stellung zu manipulierender Objekte benötigt
  - Arbeitsraum muss exakt vermessen sein

Roboterorientierte Progr.: Bsp. für Roboterwelt



Roboterorientierte Progr.: Datentypen

- VECTOR: Punkte im Raum mit homogenen Koordinaten
- RotMatrix: Rotationsmatrizen
- TransMatrix: Homogene Transformationsmatrizen
- FRAME: Frames
- JointPosition: Gelenkwinkel bei Drehgelenken und Schubdistanzen bei Schubgelenken

Roboterorientierte Progr.: Datenobjekte

- STARTPOS oder PARKPOS: Parkposition des Roboters
- WORLD: Frame des WKS
- BASE: BKS-Frame des Roboters bezogen auf das WKS
- HAND: Frame, das die Lage des Flansch-KS angibt
- TOOL: Effektor-Frame (Arbeitspunkt) bezogen auf HAND
- Xvector: Homogene Koordinate (1,0,0,1)
- Yvector: Homogene Koordinate (0,1,0,1)
- Zvector: Homogene Koordinate (0,0,1,1)
- NULLVEKTOR: (0,0,0,1)
- IdRotMatrix, IdTransMatrix: Einheitsmatrizen
- RobError: Variable, enthält letzten Fehler des Programms

Roboterorientierte Progr.: Ausdrücke

SCALAR	$*$	VECTOR	$\rightarrow$	VECTOR
VECTOR	$\pm$	VECTOR	$\rightarrow$	VECTOR
VECTOR	$*$	VECTOR	$\rightarrow$	SCALAR
VECTOR	$\times$	VECTOR	$\rightarrow$	VECTOR
RotMatrix	$*$	VECTOR	$\rightarrow$	VECTOR
RotMatrix	$*$	RotMatrix	$\rightarrow$	RotMatrix
TransMatrix	$\pm$	VECTOR	$\rightarrow$	TransMatrix
TransMatrix	$*$	VECTOR	$\rightarrow$	VECTOR
TransMatrix	$*$	TransMatrix	$\rightarrow$	TransMatrix
FRAME	$\pm$	VECTOR	$\rightarrow$	FRAME
FRAME	$*$	VECTOR	$\rightarrow$	VECTOR
FRAME	$*$	TransMatrix	$\rightarrow$	FRAME

Roboterorientierte Progr.: Kommunikation

- Synchronisation mit Robotern und Geräten
  - Signale
  - Nachrichten
  - Explizite Warteanweisungen: `WAITROBOTER`, `WAITTIME`
- Ermöglicht alle Synchronisationsarten ohne explizite Sprachmitteilungen (Semaphore, Monitor)

Roboterorientierte Progr.: Unterbrechungen

- Echtzeitsprachen müssen Aktivitäten unterbrechen können
- Asynchrone Ausführung einer benutzerdefinierten Unterbrechungsbehandlung
- Unterbrechungsereignisse
  - Nachrichten
  - Signal `AN` und seit letztem `AUS`-Zustand wurde noch keine Unterbrechung ausgelöst
  - Alarm im Programm
  - Alarm von Robotersteuerung
  - Alarm vom Betriebssystem

Roboterorientierte Progr.: Unterbrechungen

- Benutzerdefinierte Unterbrechungsbehandlungsprozedur (UBH-Prozedur) für ein Ereignis definieren
- Zuordnung von UBH-Prozedur und Priorität zum Ereignis
- Unterbrechung ausführen wenn keine UBH-Prozedur aktiv ist oder die aktive Prozedur eine geringere Priorität besitzt
- Blockierte Unterbrechung wird zurückgestellt
- Roboterbetriebssystem organisiert UBH nach Prioritäten
- Sprung zur Unterbrechungsstelle,
  wenn UBH-Prozedur mit `RETURN` beendet wird

Aufgabenorientierte Programmierung

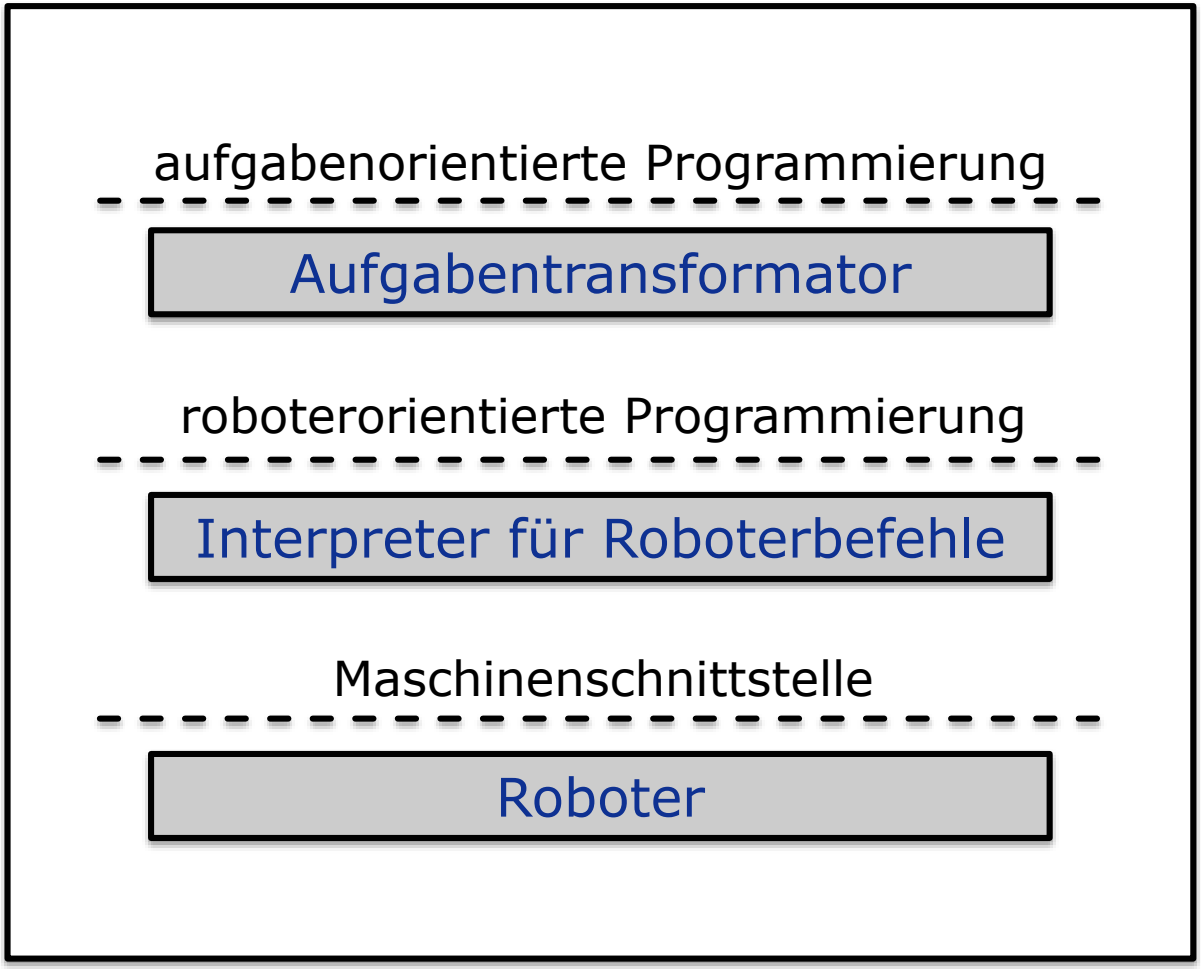
- Durchführung in Abstraktionsebenen
  (abstrakter als normale Roboterprogrammiersprachen)
- Textuelle Programmierung
- Programmierebene meist für intelligente Roboter

Roboterorientiert	Aufgabenorientiert
Wie ein Roboter eine Aufgabe löst	**Was** der Roboter ausführen soll

Aufgabenorientierte Progr.: Aufgabenplaner

- Erzeugt in roboterorientierter Sprache ein Roboterprogramm
- Benötigt
  - Faktenwissen: Wissensbasis mit Umweltmodell (Fabrik, Fertigungszellen, Roboter, Maschinen, …)
  - Operationales Wissen: Wissensbasis mit Regeln zur Zerlegung von Aufgaben in Einzelschritte
  - Verschiedene Algorithmen zur Montage-, Greif- und Bahnplanung sowie Sensorintegration
  - Synchronisationsmuster zur Koordination der Tätigkeit des Roboters mit der Umwelt

...: Aufgabentransformator-Schichtenmodell



Verfahren: Vergleich

Einlernverfahren	Textuell	Grafisch/Interaktiv
Bewegungsorientiert	Ablauforientiert	Bewegungsorientiert
Einfach (erlernbar)	Komplex (Vorkenntnisse)	Einfach/Komplex
Logischer Programmaufbau muss ergänzt werden	Reale Positionswerte müssen ergänzt werden	Positionswerte müssen korrigiert werden
Online-Verfahren (Produktionsmittel blockiert)	Offline-Verfahren	
Keine Dokumentation	Gute Dokumentation	
Schlechte Korrekturmöglichkeiten	Einfache Korrekturmöglichkeiten	
-	-	Kollisionsbehandlung durch Simulation möglich
-	-	Ermittlung von Taktzeiten, usw. durch Simulation
Geringer HW/SW-Aufwand	Mittlerer HW/SW-Aufwand	Hoher HW/SW-Aufwand

Verfahren: Anwendungskriterien

	Einlernverfahren	Textuell	Grafisch/ Interaktiv
Kinematiken	Einfache	Beliebige	
Peripherie	Wenig	Umfangreich	Wenig
Sensorik	Kaum	Beliebig	Kaum
Aufgaben-spektrum	Schmal (bewe-gungsorientiert	Breites Spektrum	
Programmier-aufkommen	Gering	Hoch	Hoch
Qualifikation	Niedrig	Mittel	Mittel
Sonstiges			Für Planungs-aufgaben

Nächste Vorlesung

- Anwendung
  - „humanzentrierte Automatisierung“
  - Robotersystem und Steuerungskonzepte
  - Testergebnisse
- Zusammenfassung und Übersicht