Robotik

Prof. Dr. Mark Ross

ross@hs-koblenz.de

WS 2018/19
Stand: 3. September 2018

Normale Slides mit Hyperlinks

Modalitäten

Modul:	E497 Robotik
	Technisches WPF, Semester 4-6, Bachelor ET/IT/MT
Umfang:	4 SWS / 5 CP
Vorkenntnisse:	Kinematik (Technische Physik I), Vektoren & Matrizen (Mathe), C++
Kontakt:	ross@hs-koblenz.de
Leistungsnachweis:	Klausur (60 min, keine Hilfsmittel)
Material:	www.hs-koblenz.de/ross

Inhalt

Literatur

[Her12] Hertzberg, Lingemann, Nüchter, *Mobile Roboter*
Springer-Vieweg, 2012

[Lan10] Prof. Thomas Langhoff, *Robotik-Glossar*
Robotik Initiative Niedersachsen, 2010, www.robini-hannover.de/robini_glossar

[Lin15] Prof. Linnemann, *Skript Robotik*
Beuth Hochschule für Technik Berlin, 2015

[Web17] Wolfgang Weber, *Industrieroboter - Methoden der Steuerung und Regelung*
3. Aufl, Carl-Hanser-Verlag, 2017

[Wey13] Prof. Michael Weyrich, Andreas Martini, *Delta-Roboter*
Uni-Siegen, 2013,
wiki.zimt.uni-siegen.de/fertigungsautomatisierung/index.php/Delta-Roboter

[Wue14] Prof. Klaus Wüst, *Grundlagen der Robotik*
Technische Hochschule Mittelhessen, 2014,
homepages.thm.de/ hg6458/Robotik/Robotik.pdf

1. Einführung

Anwendungen und Beispiele I

- Produktion: Montage, Heben von Lasten, Positionieren, ...
- Logistik, Transport
- Service/Assistenz: Rasenmähroboter, Piloted Parking
- Animatronic: TV, Hotelreception
- Bionic:
  `Bionic Kanguru`,
  `Binoic Ants`

Ziele/Motivation: Unterstützung, Komfort $\rightarrow$ Wohlstand

Risiko/Gefahr: Fehlfunktion/Gefährdung, Arbeitsplatzabbau $\rightarrow$ Armut

Roboterarten: Einteilung nach Verwendungszweck

- Industrieroboter
- Militär- und Erkundungsroboter: Konflikte, Katastrophen, Planeten
- Medizinroboter: in Chirurgie, ferngesteuert durch Arzt
- Personal Robot: Kommunikation und Interaktion mit Personen
- Serviceroboter: Rasenmähen, Staubsaugen
- Spielzeugroboter: Lego-Mindstorms, Nao (Robocup)
- Transportroboter
- Mikroroboter
- Humanoide Roboter

Definition: Industrieroboter

VDI Richtlinie 2860 Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegung [...] frei programmierbar und ggf. sensorgeführt ist. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar.

DIN EN ISO 8373 automatisch geführte, mit drei oder mehr frei programmierbaren Bewegungsachsen ausgerüstete Mehrzweckmanipulatoren, die ortsfest oder mobil in industriellen Anwendungen eingesetzt werden. Sie führen Greifer oder Werkzeuge.

Wikipedia Roboter sind stationäre oder mobile Maschinen, die nach einem bestimmten Programm festgelegte Aufgaben erfüllen.
(Anm.: Also auch eine Waschmaschine?)

Joseph Engelberger (Entwickler des ersten Industrieroboters „Unimate“, 1961) Ich kann nicht genau sagen, was ein Roboter ist, aber ich weiß, dass es einer ist, wenn ich einen sehe.

Abgrenzung: Industrieroboter versus ...

... Bagger

- ☺ hat mehrere Achsen
- ☺ mit Greifer (Schaufel) oder Werkzeug (Bohrhammer) ausrüstbar
- ☹ i.d.R. nicht programmierbar

... Portalkran

- ☺ hat drei Linearachsen
- ☹ i.d.R. nicht programmierbar

... Platinenbohrer

- ☺ hat drei Linearachsen
- ☺ programmierbar

... CNC-Fräse

- ☺ hat mehrere Achsen
- ☺ programmierbar

Industieroboter: Kinematik

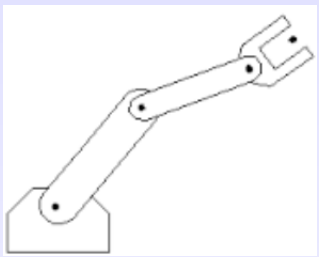
			
Knickarm Seriell	Knickarm Hybrid	Tripod Hybrid	Hexapod Parallel

[?]

Begriffe: Effektor

- Teil des Roboterarmes, das mit Umgebung in Kontakt tritt
- Roboterarm hat Aufgabe, Effektor geeignet im Raum zu führen
- Bsp.: Greifer, Werkzeug, Messspitze

- Tool Center Point (TCP): charakteristischer Punkt des Effektors, dessen Lage (Position und Ausrichtung) die Lage des Effektors eindeutig beschreibt
  z.B. Messspitze, Zentrum der Greifbacken



Art der Greifer: Elektrisch, Magnetisch, Pneumatisch, Vakuum
Beispiele: Parallelgreifer,
Universal Jamming Gripper,
Sushi Hand,
Giant Hand

Begriffe: Armglieder, Gelenke

Armteil: bewegliches Glied bei (stationärem) Roboter

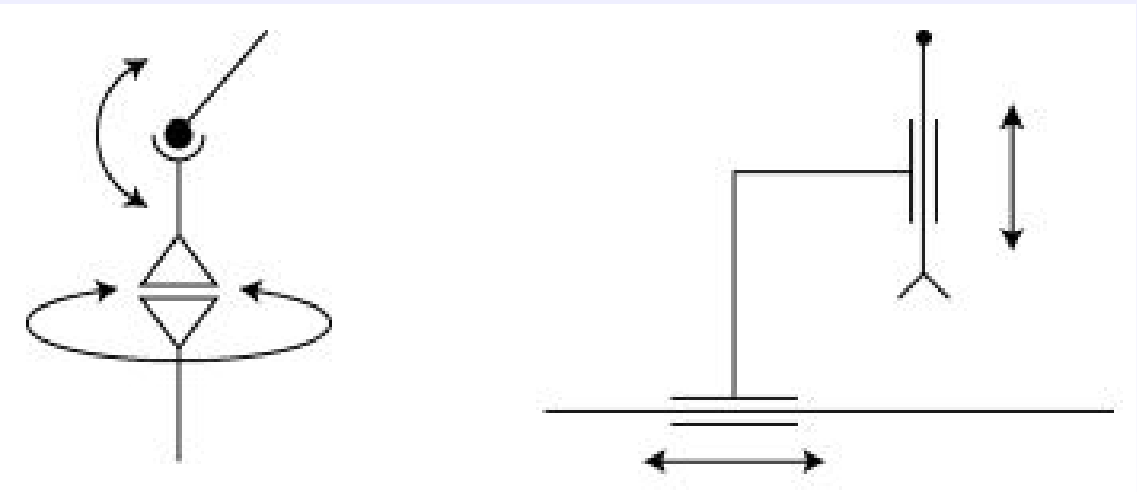
Antrieb: bewegliche Komponenten zur Fortbewegung bei (mobilen) Robotern

Seriellkinematik: (Offene kinematische Kette) jedes Armteil mit beweglichem Gelenk an vorhergehendem Armteil montiert, Effektor ist letztes Glied der Kette, alle Gelenke bilden eine serielle Kette, jedes Gelenk bewegt eigenständig die folgenden Armteile

Parallelkinematik: Mehrere Gelenke bewegen gemeinsam die folgenden Armteile oder Effektor

Translatorische Gelenke: (Linearachse) nächstes Armglied bewegt sich auf gerader Bahn, Montage auf Gleitschienen (wie Pneumatikzylinder) oder Rollschienen (wie Laufkatze bei Portalkran)

Rotatorische Gelenke: (Drehachse) nächstes Armglied dreht sich um Rotationsachse, Montage auf Gelenkbolzen oder Rollenlager (vgl. Mensch, Bagger)

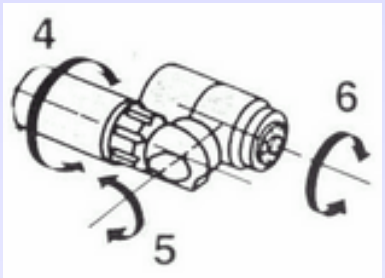


DIN 2861: Symbole für Robotergelenke
[?]

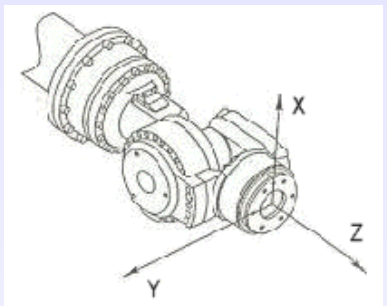
Orientierung des Effektors

Pose: Position + Orientierung

Winkelhand: Achsen 4 und 5 schneiden sich, Drehachse 6 mit Versatz vorgelagert, einfach/kostengünstig, ungünstiger Arbeitsraum, aufwendige Kabelführung, komplizierte Berechnung der Achsstellungen in kartesische Koordinaten



Zentralhand: drei Drehachsen haben gemeinsamen Schnittpunkt, günstige Bauform, einfache Kabelführung, einfache Berechnung der Achsstellungen in kartesische Koordinaten

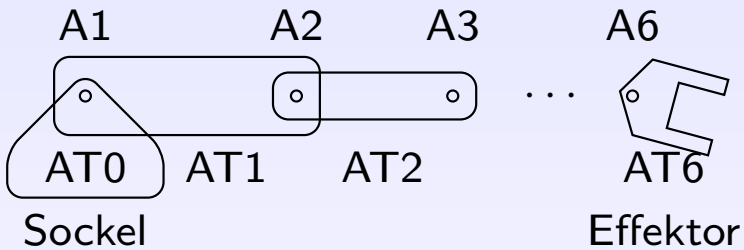


[?]

Typische Bauarten von Industrierobotern I

Aufbau: 6-achsiger Roboter

- 3 Hauptachsen (Grundachsen, Achse 1-3) steuern Position des Effektors im Raum
- Bezeichnung der Bauart beschreibt Grundachsen: RRT, TRR, RRR (Knickarm)
- 3 Handachsen (Nebenachsen, Achse 4-6) steuern Orientierung des Effektors, geringer Beitrag zur Position, rotatorisch
- Der Greifer wird nicht als Achse gezählt
- Offene kinematische Kette: Sockel (Armteil 0) - Achse 1 - Armteil 1 - . . . - Armteil 5 - Achse 6 - Effektor



3-5 Achsen

- i.d.R. 3 Hauptachsen und weniger Nebenachsen, eingeschränkte Orientierung
- Anwendungen: Pick & Place, Bohren

mehr als 6 Achsen

- auch nur 6 Freiheitsgrade, redundante Achsen (parallele T- oder R-Achsen)
- bessere Beweglichkeit, erhöhte Kollisionsvermeidung

Typische Bauarten von Industrierobotern II

Translationsachsen

- Bewegungen für Menschen gut vorstellbar
- Einfache Berechnung der Koordinaten, Orientierung der Folgeachsen unverändert
- Achskoordinaten passen gut zu kartesischen Weltkoordinaten

Rotationsachsen

- gutes Verhältnis von Arbeits- zu Kollisionsraum
- kleines Gelenkspiel, gute Steifigkeit
- hohe Arbeitsgeschwindigkeit

Seriellroboter

- Knickarmroboter (RRR)
- Schwenkarmroboter (RRT, TRR, SCARA - Selective Compliance Assembly Robot Arm), z.B. Roboter zum Bohren von Platinen mit $f = 3$ (x,y,z, Orientierung fest)
- Portalroboter (TTT)
  Karthesisches KS, große Arbeitsräume, hohe Traglasten möglich

Parallelroboter

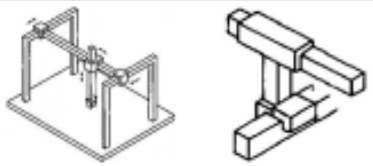
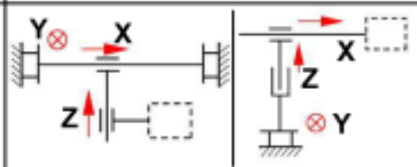
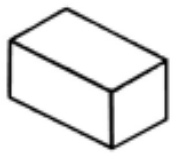
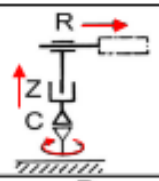
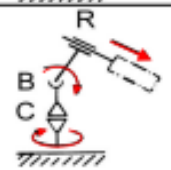
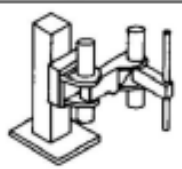
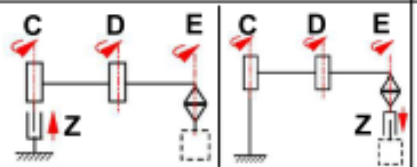
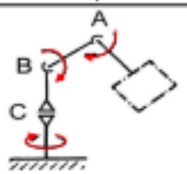
- Delta-Roboter, Tripod, Hexapod

Hybride Kinematik

- z.B. Delta-Roboter mit Zentralhand

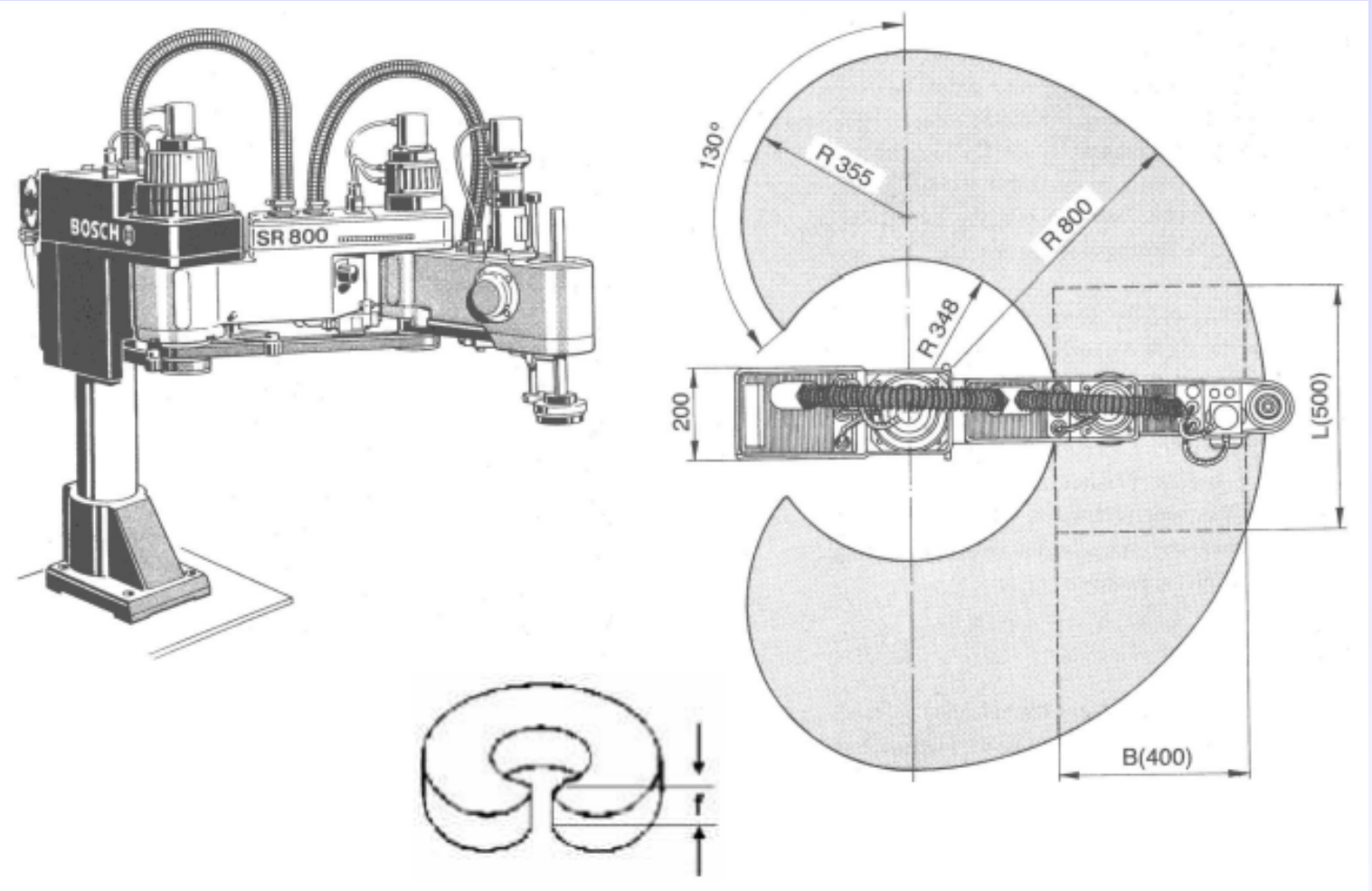
Arbeitsraum, Kollisionsraum

- Arbeitsraum = nutzbarer Raumbereich, der mit TCP erreicht werden kann
- Kollisionsraum = Raumbereich, der bei Bewegung benutzt wird, inklusive aller Teile
- Kollisionsraum $\geq$ Arbeitsraum

Bezeichnung	Anordnung	Kinematisches Ersatzbild	Arbeitsraum
Kartesisches Gerät			
Zylinder-koordinatengerät			
Kugel-koordinatengerät			
Horizontales Knickarmgerät			
Vertikales Knickarmgerät			

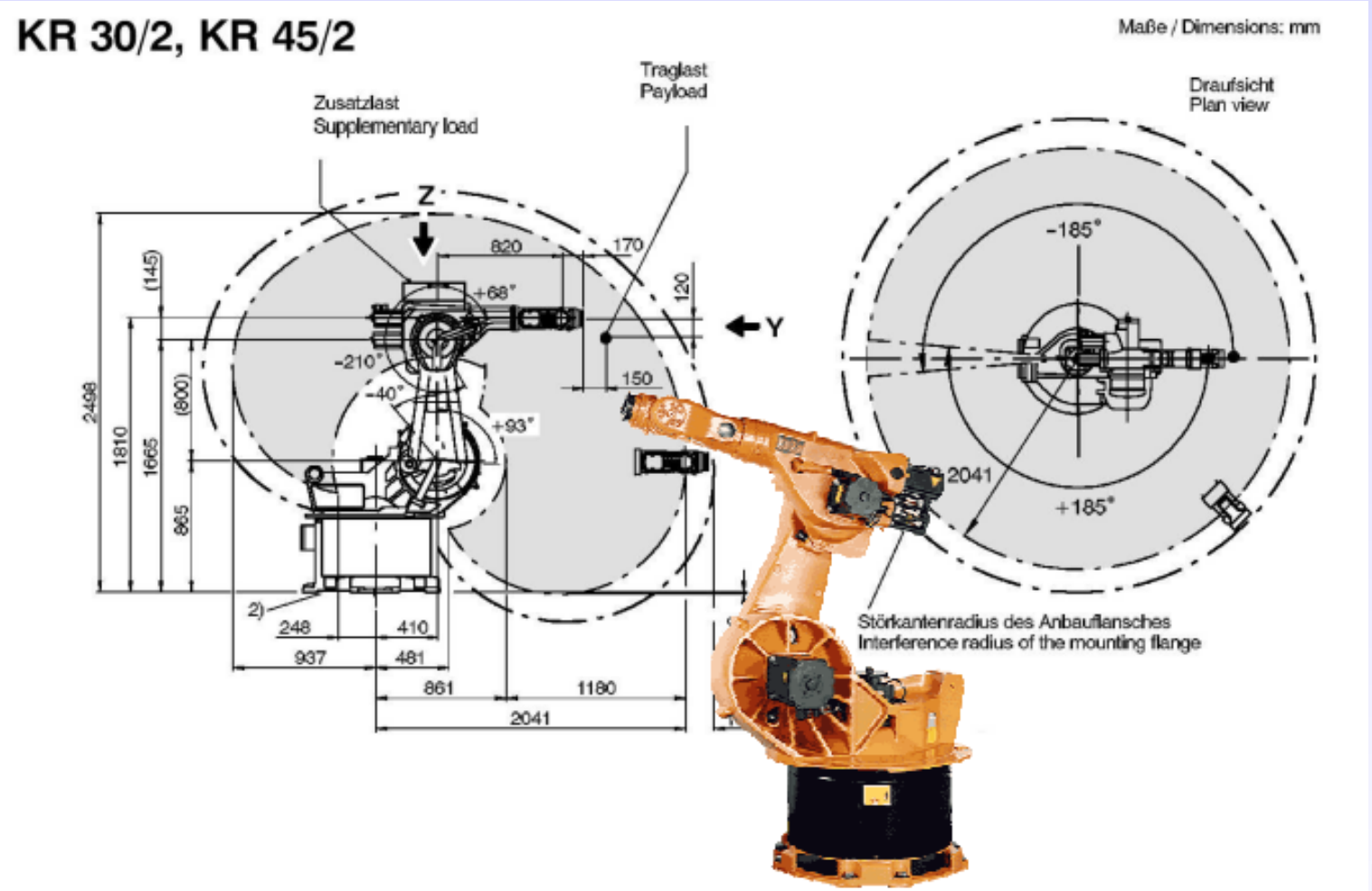
[Lin15]

Arbeitsraum Bosch SR 800



[Lin15]

Arbeitsraum Kuka KR2



[Lin15]

Bewegungsachse, Freiheitsgrad

Bewegungsachse: geführte Verbindung unabhängig voneinander angetriebener Glieder
I.d.R Drehgelenke oder Schubgelenke.
Kreuz-/Kugelgelenke entsprechen zwei/drei Drehgelenken mit Abstand Null

Freiheitsgrad f: Anzahl möglicher unabhängiger Bewegungen (Translation, Rotation) eines Körpers gegenüber Bezugssystem

Bsp.:
- Massenpunkt im Raum: $f =$
- Baggerarm mit Schaufel: $f =$
- Werkstück auf Förderband: $f =$
- Greifer im Raum: $f =$

Getriebefreiheitsgrad F: Anzahl unabhängiger Achsen, die zu eindeutiger Bewegung führen
Geschickte Achsanordnung führt bei $F = 6$ zu maximalem Freiheitsgrad $f = 6$
$F > 6$ (redundante Kinematik) verbessert u.U. Feinbewegung

Bsp.: „Getriebefreiheitsgrad“ des menschlichen Armes ohne „Greifer“

Schulter	Kugelgelenk	$F =$
Ellbogen	Drehgelenk	$F =$
Unterarm	Drehgelenk	$F =$
Handgelenk	Doppeldrehgelenk	$F =$
Summe		$F =$

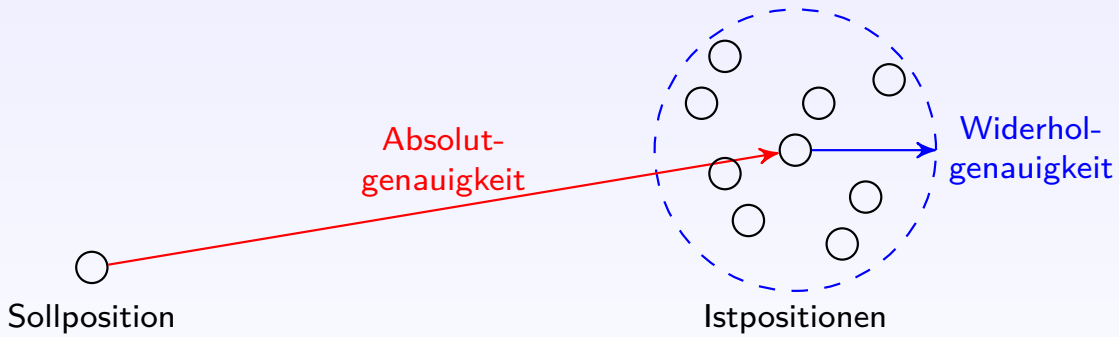
Redundanz: bei fixierter Hand bleibt Ellbogen beweglich

Wichtige Systemparameter

- Traglast: max. Lastmasse mit zulässigem Kraftarm des Schwerpunktes
- Verhältnis Lastmasse zu Eigenmasse: typisch 1:10, bei Leichtbaurobotern auch 1:1
- Arbeitsbereich, Kollisionsbereich
- Getriebefreiheitsgrad: oft F=6
- Geschwindigkeit
- Genauigkeit: Position und Bahn [1]
- Einbauart: Boden, Decke, Wand, Mobil
- Preis, Eigengewicht, Anschlussbedingungen

[1] Unterscheide Absolutgenauigkeit und Wiederholgenauigkeit

Absolutgenauigkeit: Berechnete Position $\stackrel{?}{=}$ Istposition, kann durch Kalibrierung verbessert werden
Wiederholgenauigkeit: Reproduzierbarkeit, Maß für Streuung, Ursache z.B. Getriebespiel



Herausforderungen

Allgemeine Aufgaben und Ziele:

- Genaue Positionierung, Lageregelung auch bei schwankenden Lasten
- Genaue Kraftdosierung, z.B. Anpressdruck beim Fügen
- Begrenzung von Momenten, Geschwindigkeiten, Anlaufströmen: Sanftanlauf, Anhalten ohne Überschwingung
- Genaue Bahnführung, z.B. bei Linienschweißen oder Bearbeitung bewegter Werkstücke auf Förderband
- Sicherheit für Menschen
- Vermeidung von Kollisionen mit Umgebung und Kollabieren von Gelenken

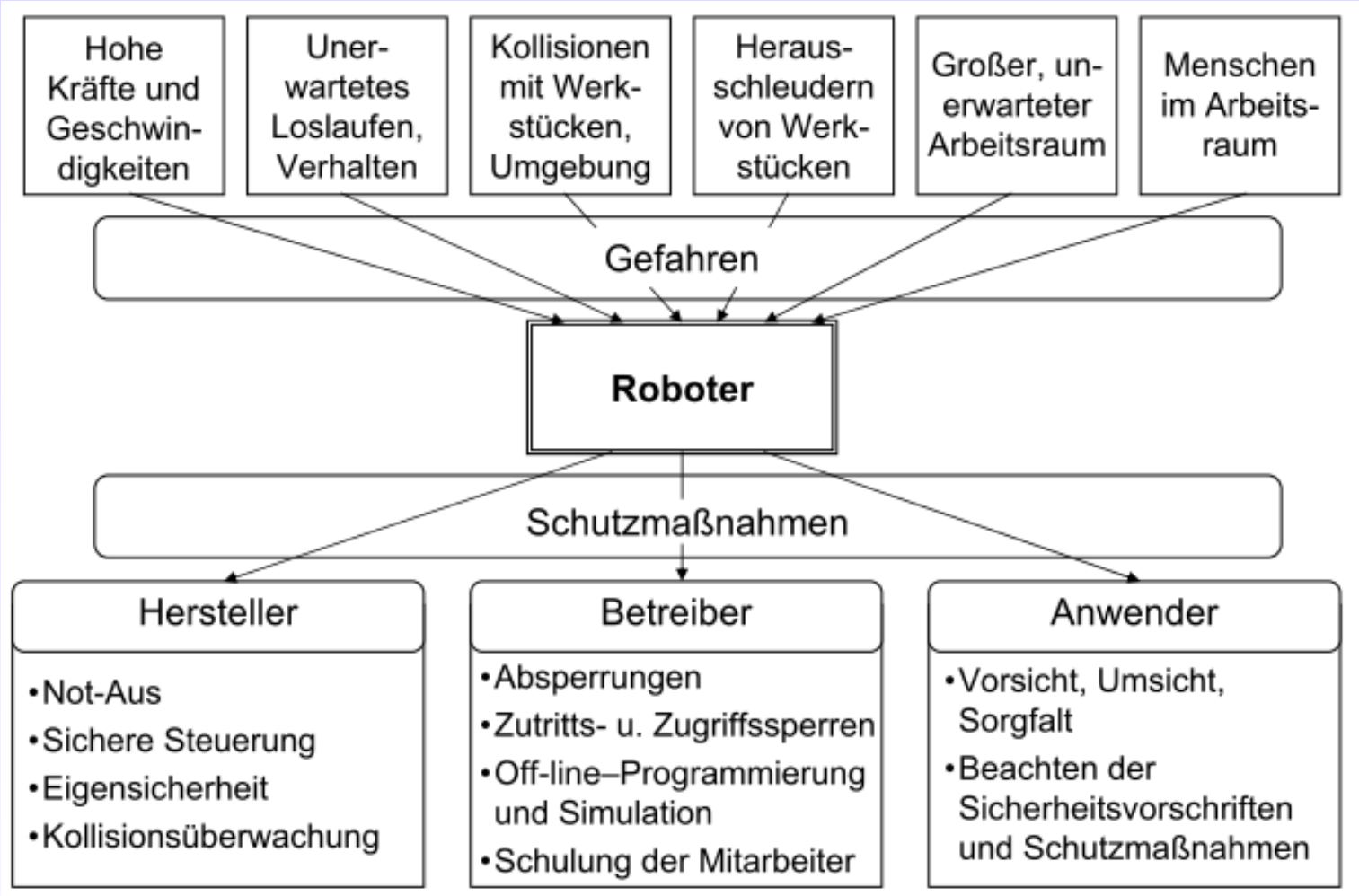
Strategien:

- Servomotoren: geregelte Motoren mit eigener Sensorik
- Aufwendige Regelungsalgorithmen
- Sicherheitstechnik: Lichtschranken, berührungslose Näherungssensoren, Kameratechnik, Einhausung mit Türkontakten

Komponenten von Robotersystemen

- Mechanik mit Aktoren (el. Antriebe, Servomotoren, Pneumatik) und Sensoren (Achsposition, Drehmomente)
- Sensorik zur Erfassung der Umwelt
- Anschluss: Strom, Druckluft, Netzwerk
- Intelligenz/Rechnertechnik: Steuerung und Regelung
- Vernetzung: Mechanik, Rechner, HMI, Programmiergerät, Prozessleitsystem
- Sicherheit
- Material: Trend zu Leichtbau, Stahl $\rightarrow$ Alu $\rightarrow$ Kunststoff

Sicherheitstechnik



[Lin15]

Sicherheitstechnik

Zum Sicherheitsteil einer Robotersteuerung gehören unter anderem:

- Hard- und Softwareendschalter für alle Achsen
- Betriebsartenwahlschalter (Schlüsselschalter)
- Überwachung der Schaltsysteme für Zustimmungsschalter, Notausschalter
- Überwachung der Geschwindigkeiten im Einrichtbetrieb
- Einschaltdiagnose für alle kontaktbehafteten Schaltvorgänge, die Sicherheitsfunktionen übernehmen (z.B. Antriebe ausschalten)
- Watch-Dog-Funktionen
- Spannungs- und Temperaturüberwachung
- Fail Save - Verhalten

Sicherer Umgang mit Robotern

- Komplexe Kinematik (von sechs Achsen) erschwert Abschätzung, wie schnell Roboter sich wohin bewegen wird
- Menschen (Programmierer) glauben oft, Roboterbewegung genau vorhersagen zu können. Jedoch:
  - Defekt an Achsmotor/Sensorik kann zu unerwarteten Bewegungen führen
  - Fehler im Programm
  - Einrichtbetrieb mit kleinen Geschwindigkeiten erzeugt falsche Vorstellung von späteren Geschwindigkeiten im Automatikbetrieb.
- Gewöhnung im Umgang mit Robotern $\rightarrow$ zunehmender Leichtsinn.
- Einrichten eines Roboters stets mit zweiter Person am Not-Aus-Knopf

2. Serielle Industrieroboter

Begriffe

Dynamikmodell: Beschreibt Kräfte (Antrieb, Schwerkraft, Reibung, Trägheit), Drehmomente und daraus resultierende Bewegungen

Kinematikmodell: Vereinfachung des Dynamikmodells.
Beschreibt geometrischen Zusammenhang von mechanischer Struktur und resultierenden Bewegungen

Direkte Kinematik: Vorwärtskinematik.
Berechnet Position des Effektors aus Einstellparametern aller Gelenke.
Position ist immer eindeutig.

Inverse Kinematik: Rückwärtskinematik.
Berechnet Gelenkparameter um bestimmte Position/Orientierung zu erreichen.
Keine, eine oder mehrere Lösungen möglich

Vektoren

Ortsvektor beschreibt Position im Raum (Koordinaten x, y, z) bezogen auf Ursprung (0,0,0)

Freier Vektor beschreibt Richtung im Raum, kann auf Wirklinie verschoben werden

Beispiele
Ortsvektor:
Freier Vektor:

Vektoren

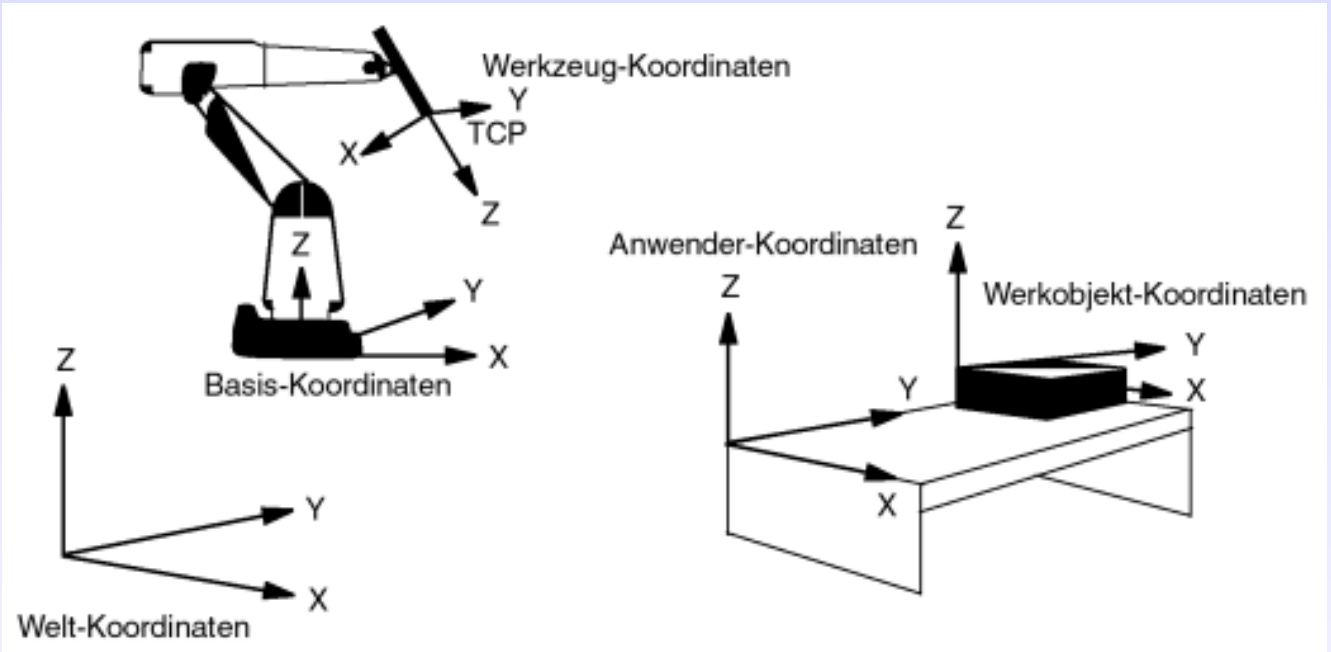
Ortsvektor beschreibt Position im Raum (Koordinaten x, y, z) bezogen auf Ursprung (0,0,0)

Freier Vektor beschreibt Richtung im Raum, kann auf Wirklinie verschoben werden

Beispiele
Ortsvektor:
Freier Vektor:

Koordinatentransformation - Motivation

Verschiedene Koordinatensysteme



Drehsinn bei Rotationen

Drehrichtung um Achse: rechter Daumen in Achsrichtung, positive Drehung in Richtung gekrümmte Finger (Rechte-Hand-Regel)

Lineare Abbildungen I

Beliebige lineare Abbildung eines Ortsvektors $(x\ y)$ in der Ebene:

$$x' = a_{11}x + a_{12}y + a_{13}$$
$$y' = a_{21}x + a_{22}y + a_{23}$$

Erweiterung um dritte Gleichung

$$1 = 0x + 0y + 1$$

ergibt Homogene Matrix (Frame):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y + a_{13} \\ a_{21}x + a_{22}y + a_{23} \\ 0x + 0y + 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Translation um $(dx\ dy)$:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + dx \\ y + dy \\ 1 \end{pmatrix}$$

Rotation um α:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Lineare Abbildungen II

Andere Transformationen (z.B. beliebige Rotationsachsen, DH-Transformation) durch Verkettung obiger Elementartransformationen

Achtung: Bei *freien* Vektoren darf keine Konstante addiert werden:

$$\begin{pmatrix} x' \\ y' \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ \mathbf{0} \end{pmatrix}$$

Lineare Abbildungen III

Im 3D entsteht Homogene Matrix durch vierte Gleichung, also 4×4-Matrix.

Translation
$$T(v) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um x-Achse
$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um y-Achse
$$R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um z-Achse
$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Koordinatentransformation I

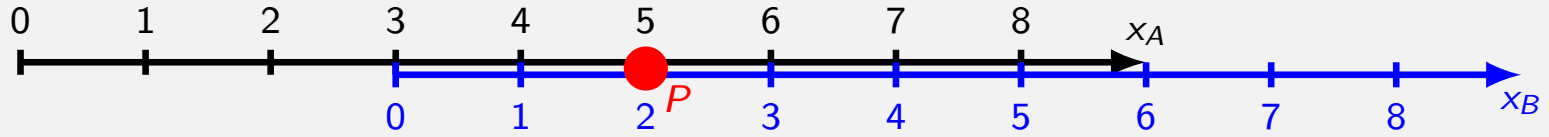
Transformation von Achsen $\vec{e}_x, \vec{e}_y$ und Ursprung $\vec{0}$ von K1 nach K0 ergibt

$$\begin{pmatrix} \vec{e}_x & \vec{e}_y & \vec{0} \\ 0 & 0 & 1 \end{pmatrix}_{\mathbf{K1}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{\mathbf{K1}} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}_{\mathbf{K0}}$$

Wenn die Transformation(smatrix) T das KS K_A ins KS K_B überführt ($K_B = T\ K_A$), dann transformiert T auch Punkte von K_B in K_A ($P_A = T\ P_B$).

Koordinatentransformation II

Beispiel: K_B ist gegenüber K_A um 3 Einheiten in x-Richtung verschoben.



Transformation des Punktes P_B:

$$P_A = T\ P_B = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

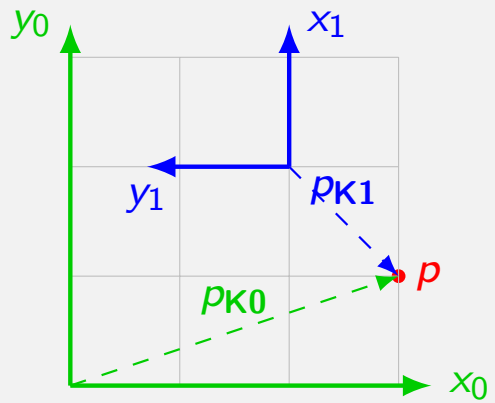
Transformation des KS K_A:

$$K_B = T\ K_A = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Koordinatentransformation III

Geg.: Punkt in K1 $p_{\mathbf{K1}} = (-1, -1)$.
Ges.: Transformationsmatrix und $p_{\mathbf{K0}}$

Lsg.: $p_{\mathbf{K0}} = \begin{pmatrix} & \\ & \end{pmatrix} \cdot \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}_{\mathbf{K1}} = \begin{pmatrix} \\ \end{pmatrix}_{\mathbf{K0}}$



Notation von Transformationsmatrizen

. . . sind leider nicht einheitlich

A) Notation nach [**?**]:

Punkt p im KS_i	${}^{i}p$
Transformation vom KS_i ins KS_k	${}^{k}p = {}^{k}T_i \, {}^{i}p$
Kinematische Kette	${}^{k}T_i = {}^{k}T_{k+1} \, {}^{k+1}T_{k+2} \ldots {}^{i-1}T_i$

B) Notation nach [**?**]:

Punkt p im KS_i	p^i
Transformation vom KS_i ins KS_k	$p^k = {}^{i}_{k}T p^i$
Kinematische Kette	${}^{k}_{i}T = {}^{i+1}_{i}T \cdot {}^{i+2}_{i+1}T \ldots \cdot {}^{k}_{k-1}T$

C) Notation nach [**?**]:

Punkt p im KS_i	wie A)
Transformationsmatrizen	wie B)

2. Serielle Industrieroboter

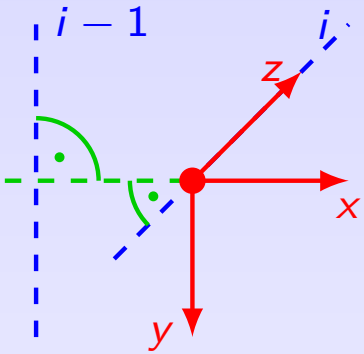
DH-Konventionen

- nach Denavit und Hartenberg, USA 1955
- Methode zur Beschreibung der Beziehungen zwischen zwei Gelenken mit nur 4 DH-Parametern
- Erleichtern Durchführung der kinematischen Vorwärts- und Rückwärtstransformation
- Bestehen aus:
  - DH-Konventionen zur Festlegung der KO-Systeme
  - DH-Transformationen zur Erzeugung der KO-Systeme
  - DH-Parametern für die Transformationen
- Festlegung der KO-Systeme nach festen Regeln, jedoch teilweise mit Wahlfreiheit
- Nutze Wahlfreiheit so, dass möglichst viele DH-Parameter Null werden

DH-Konventionen der KO-Systeme

Windschiefe Achsen = nicht-parallel und nicht-schneidend

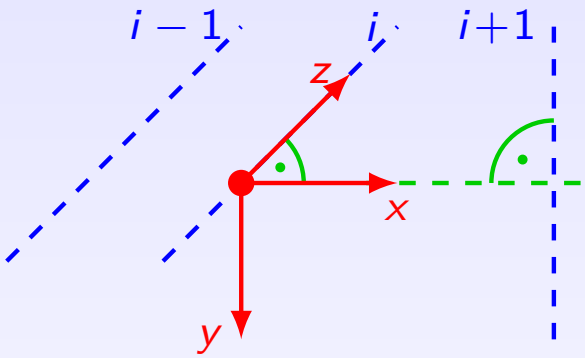
1. Suche gemeinsame Normale zwischen Gelenkachsen
2. Ursprung 0_i= Schnittpunkt von Normale und Gelenkachse i
3. z_i-Achse = Gelenkachse (2 Möglichkeiten)
4. x_i-Achse in Richtung Normale, von 0_{i-1} wegweisend
5. y_i-Achse zum Rechtssystem ergänzen



Parallele Achsen:

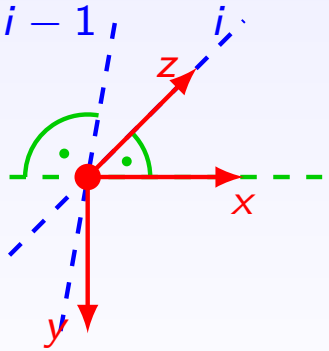
1. Suche Normale zur *nächsten* Gelenkachse $i+1$

... wie oben



Sich schneidende Achsen:

1. Ursprung 0_i= Schnittpunkt der beiden Achsen
2. z_i-Achse = Gelenkachse (2 Möglichkeiten)
3. x_i-Achse senkrecht zu beiden Gelenkachsen legen
4. y_i-Achse zum Rechtssystem ergänzen



DHT - Denavit-Hartenberg-Transformation I

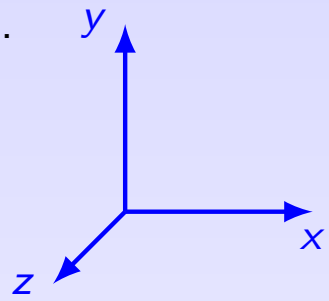
- Nach Festlegung der KS werden die DHTs bestimmt.
- Sie erzeugen jedes KS aus dem vorhergehenden.
- Es ergibt sich eine kinematische Kette der Armteile:
  AT0-AT1-...-AT6
- Jede DHT besteht aus vier elementaren Transformationen

	Parameter	Drehgelenk	Lineargelenk
1. Rotation um x-Achse	α_i	konst.	konst.
2. Translation entlang x-Achse	a_i	konst.	konst.
3. Translation entlang z-Achse	d_i	konst.	var.
4. Rotation um z-Achse	θ_i	var.	konst.

  (konst. $\hat{=}$ konstruktiver Parameter, var. $\hat{=}$ Gelenkstellung)
- Die DHT einer Achse ist $DH_{i-1\to i} = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i)$, wobei θ_i die Drehung um und d_i die Verschiebung entlang der Gelenkachse beschreibt
- Die gesamte DHT eines seriellen 6-Achs-Roboters umfasst also
  - 24 Multiplikationen mit 4×4-Matrizen
  - mit 6 Variablen (Gelenkstellungen)
  - und 18 geometrischen Konstanten
- Oft sind einige der 18 Konstanten gleich Null, z.B. die Translationen bei Zentralhand

DHT - Denavit-Hartenberg-Transformation II

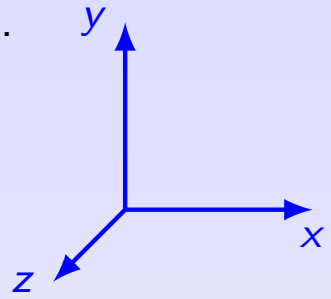
0.



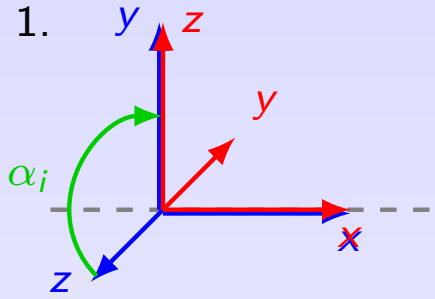
Schritt	Parameter

DHT - Denavit-Hartenberg-Transformation II

0.



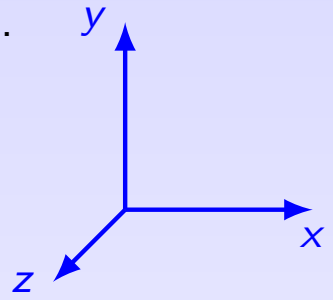
1.



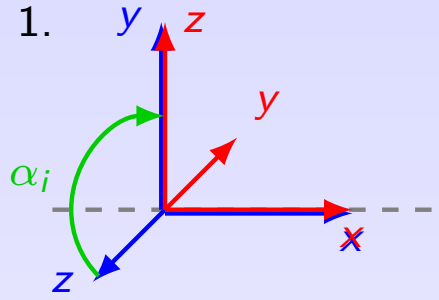
	Schritt	Parameter
1.	Rotation um x-Achse	α_i

DHT - Denavit-Hartenberg-Transformation II

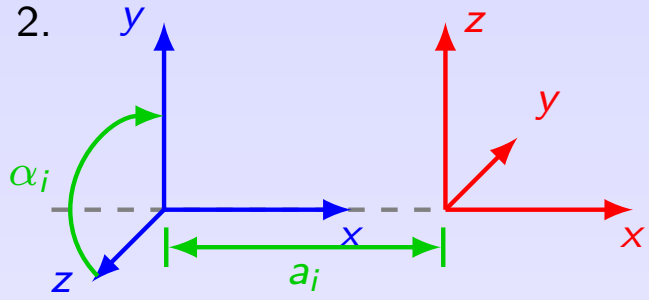
0.



1.



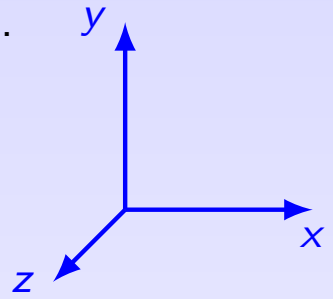
2.



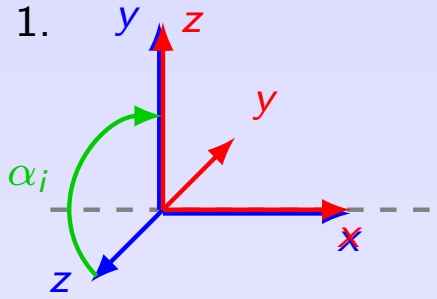
	Schritt	Parameter
1.	Rotation um x-Achse	α_i
2.	Translation entlang x-Achse	a_i

DHT - Denavit-Hartenberg-Transformation II

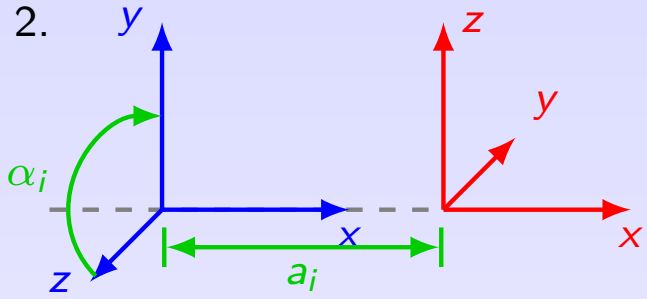
0.



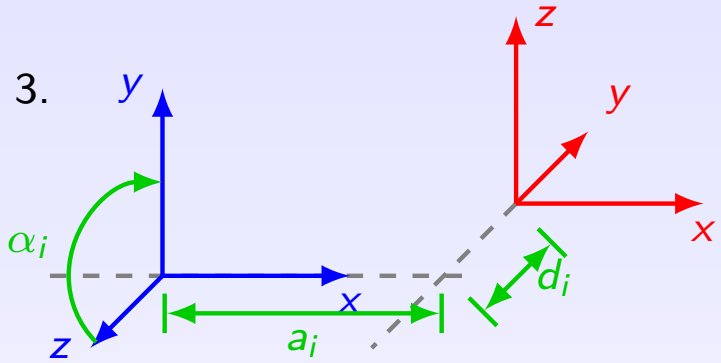
1.



2.



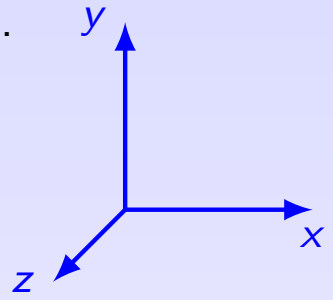
3.



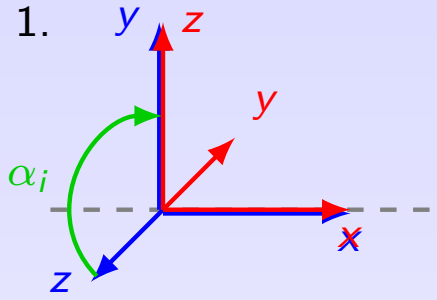
	Schritt	Parameter
1.	Rotation um x-Achse	α_i
2.	Translation entlang x-Achse	a_i
3.	Translation entlang z-Achse	d_i

DHT - Denavit-Hartenberg-Transformation II

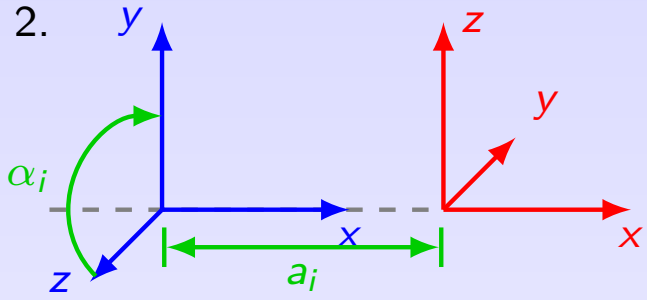
0.



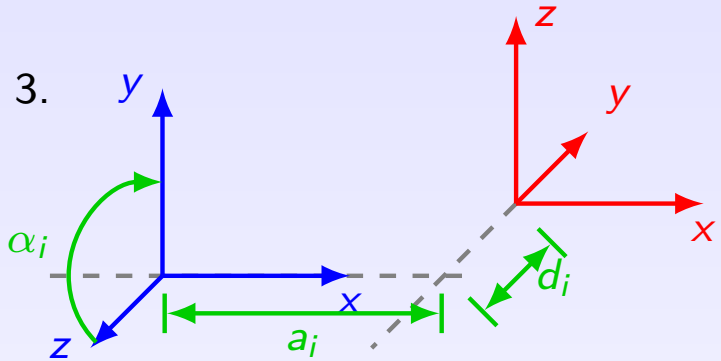
1.



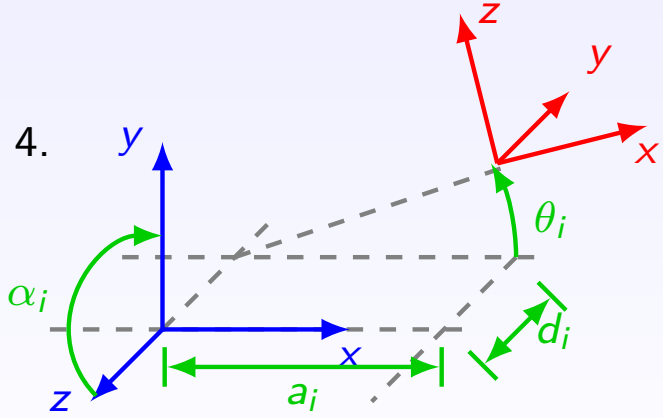
2.



3.



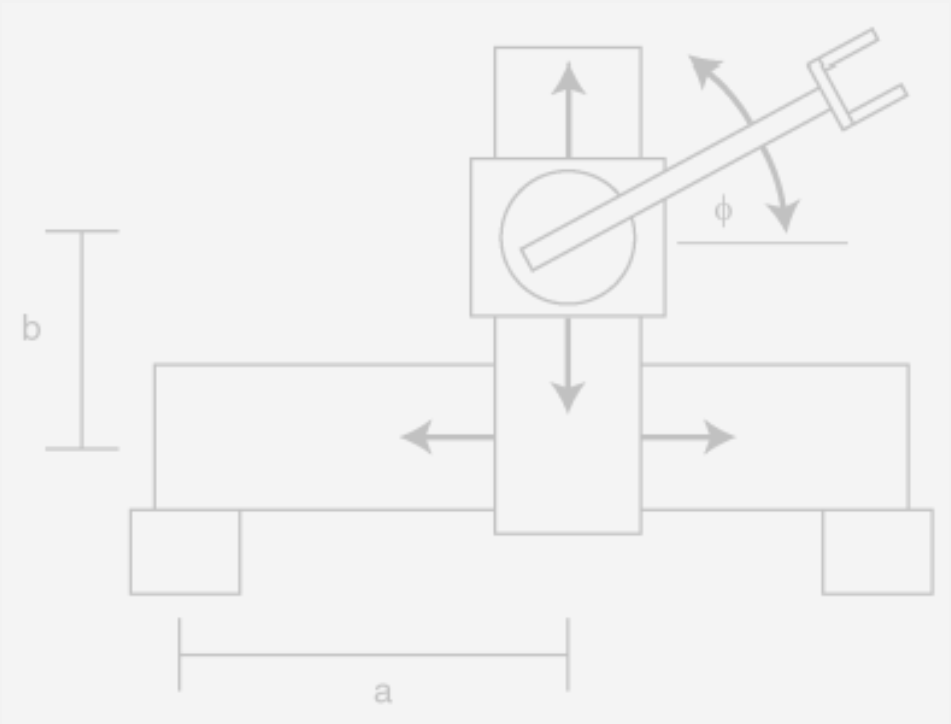
4.



	Schritt	Parameter
1.	Rotation um x-Achse	α_i
2.	Translation entlang x-Achse	a_i
3.	Translation entlang z-Achse	d_i
4.	Rotation um z-Achse	θ_i

Beispiel [?, S. 124]

a) Legen Sie für den TTR-Roboter in jedes Armteil gemäß DH-Konventionen ein KO-System. TCP = Fußpunkt der Greiferbacken.



b) Die DH-Parameter zur Umrechnung der KO-Systeme sind:

Armteil	θ_i	d_i	a_i	α_i
1				
2				
3				

Beispiel [?, S. 124]

c) Geben Sie die drei DH-Transformationsmatrizen an.

$$A_1 = \begin{pmatrix} & \\ & \end{pmatrix}$$

$$A_2 = \begin{pmatrix} & \\ & \end{pmatrix}$$

$$A_3 = \begin{pmatrix} & \\ & \end{pmatrix}$$

d) Wie lautet die Gleichung der Vorwärtstransformation?

$$A_1 A_2 A_3 = \begin{pmatrix} & \\ & \end{pmatrix}$$

Beispiel [?, S. 124]

e) Im Folgenden gelte: a=3, b=2, $\phi = 45°$, $r = \sqrt{2}$, $\sin\phi = \cos\phi = \sqrt{1/2}$. Damit erhält man die Vorwärtstransformation zu

$$A = A_1 A_2 A_3 = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

f) Die Position des TCP in Koordinaten des Koordinatensystems K_0 ist

$$P_{\mathrm{TCP}} = A_1 A_2 A_3 \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(3)} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix} \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(3)} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(0)}$$

g) Berechnen Sie die Koordinaten des Punktes, der im Effektorkoordinatensystem K_3 die Koordinaten $(2, 2, 1)_3$ hat, im Fußkoordinatensystem K_0.

$$P = A_1 A_2 A_3 \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(3)} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(0)}$$

Beispiel [?, S. 124]

h) Berechnen Sie die Koordinaten des Punktes, der im Fußkoordinatensystem K_0 die Koordinaten (4,-3,4) hat, im Effektorkoordinatensystem K_3.
Dazu wird zunächst die Inverse berechnet (z.B. mit Matlab)

$$A^{-1} = (A_1 A_2 A_3)^{-1} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Der gesuchte Punkt ist:

$$P = A^{-1} \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(0)} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}_{(3)}$$

Wiederholung Kinematik

- Allgemeine, geradlinige Bewegung

Weg	s
Geschwindigkeit	$v = \frac{ds}{dt} = \dot{s}$
Beschleunigung	$a = \frac{dv}{dt} = \frac{d^2s}{dt^2} = \ddot{s}$

- Gleichmäßig beschleunigte Bewegung

Weg	$s = \frac{1}{2} a_0 t^2 + v_0 t + s_0$
Geschwindigkeit	$v = a_0 \cdot t + v_0$
Beschleunigung	$a = a_0 =$konst

- Rampenprofil ($s_0 = 0, v_0 = 0$):

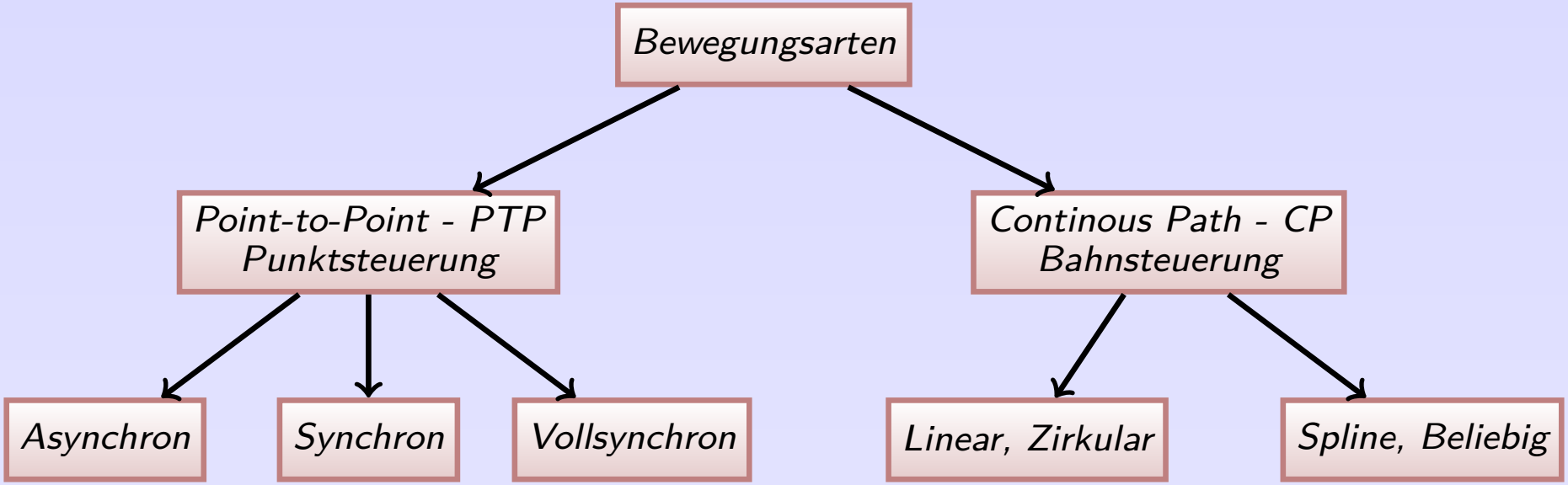
Beschleunigung	Unbeschleunigt	Bremsen
$t \in [t_0, t_1]$	$t \in [t_1, t_2],\ t' = t - t_1$	$t \in [t_1, t_2],\ t'' = t - t_2$
$a = +a_0$	$a = 0$	$a = -a_0$
$v = a_0 t$	$v = v_1 =$ konst	$v = v_2 - a_0 t''$
$s = \frac{1}{2} a_0 t^2$	$s = s_1 + v_1 t'$	$s = s_2 + v_2 t'' - \frac{1}{2} a_0 t''^2$

2. Serielle Industrieroboter

Bahn

- Bahn = Bewegungsablauf zur Erreichung bestimmter Pose
- Direkte Vorgabe von Zielposition (PtP):
  unvorhersehbare Bahn $s(t)$, Geschwindigkeit $v(t)$ und Beschleunigung $a(t)$,
  Keine Kollisionsvermeidung,
- Bahnvorgabe durch Interpolation (CP): Vorgabe von Zwischenpunkten
- Überschleifen: Übergabe eines neuen Zwischenpunktes noch vor Erreichung der Vorposition
  $\rightarrow$ ruckfreie, schnelle Bewegung, Zwischenpunkte werden nicht exakt angefahren

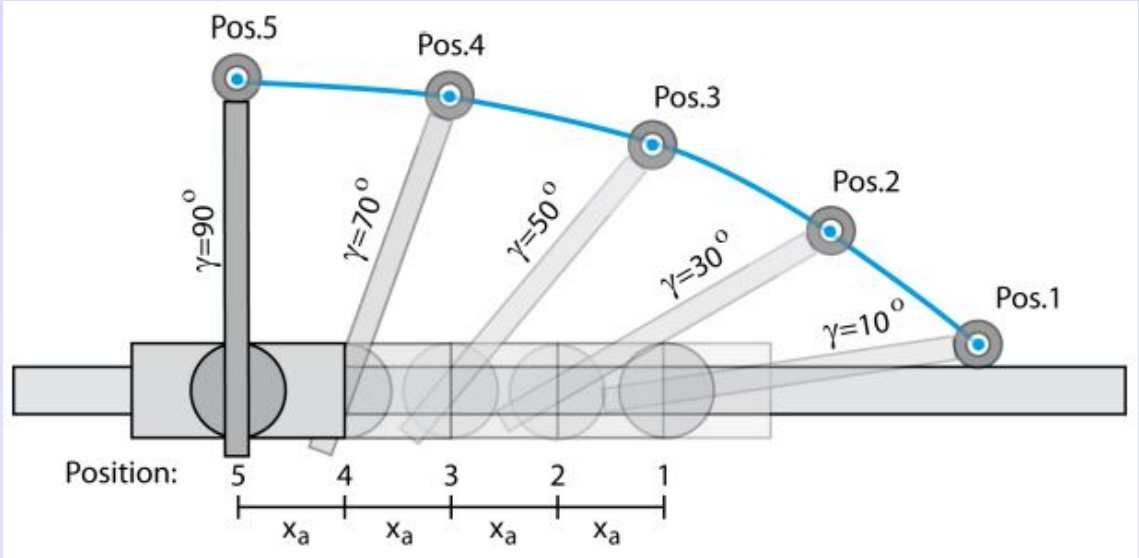
Bewegungsarten

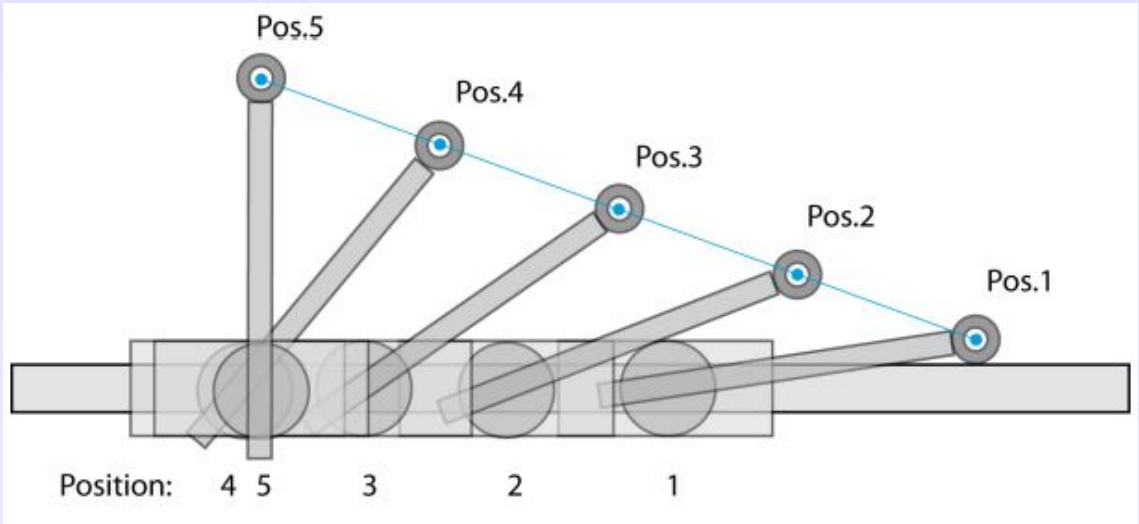


- Nur Start- und Zielpose fest, Bahn beliebig/unvorhersehbar
- Anwendung: Punktschweißen
- \+ zeitoptimales Verfahren, geringe Beschleunigung und Verschleiß

- Ziel: exakte Einhaltung einer vorgegebenen Bahn
- Anwendung: Linienschweißen, Lackieren
- − Rechenintensiv, abrupte Richtungsänderungen, hoher Verschleiß, regt zu Schwingung an

Bewegungsarten: PTP vs. CP





[?]

Vergleiche: $dx(t)$ und $d\gamma(t)$

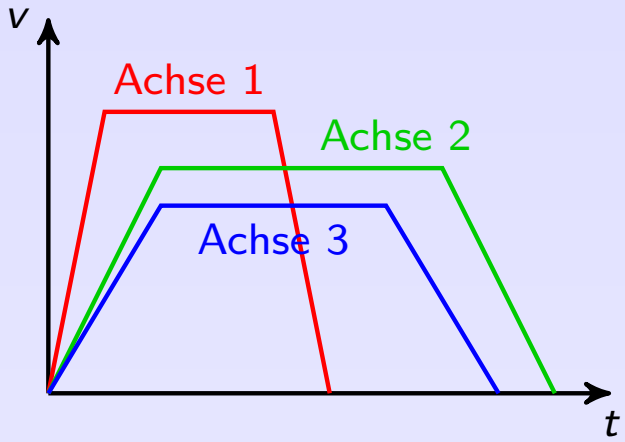
Versehentliche PTP-Steuerung kann zu Kollision führen

Bewegungsarten: PTP - Point-to-Point I

- Steuerung von Start zu Ziel bei beliebiger, unbekannter Bahn
- Aus Zielpose wird durch inverse Kinematik Zielparameter der Gelenke ermittelt
- Zwischenpositionen werden aus Achskoordinaten berechnet
- Keine Kollisionsvermeidung

Asynchrone Bewegung

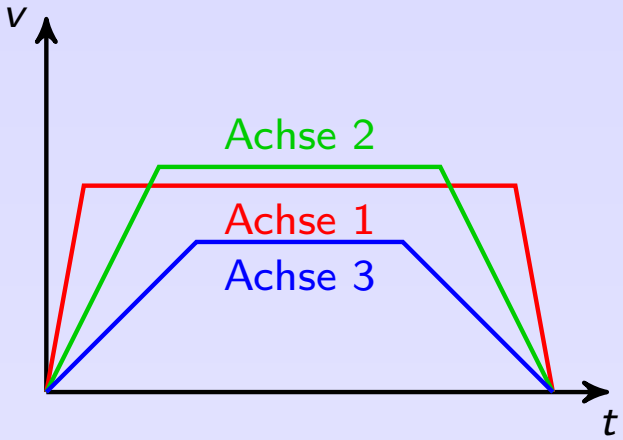
- Alle Achsen starten gleichzeitig mit jeweils max. Geschwindigkeit
- Achsen erreichen nacheinander das Ziel
- Bahn weicht oft stark von direkter Verbindungslinie ab
- Langsamste Achse bestimmt Gesamtdauer



Bewegungsarten: PTP - Point-to-Point II

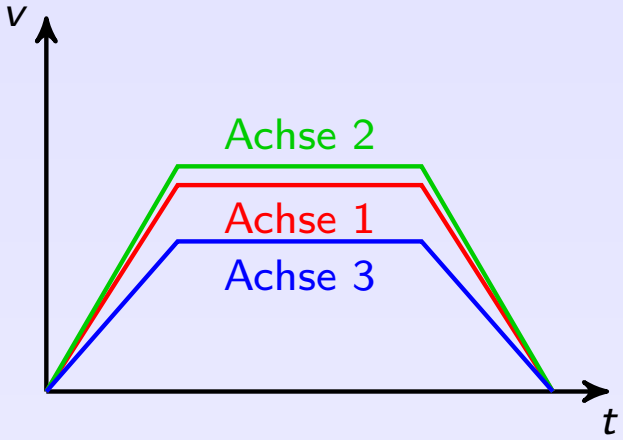
Synchrone Bewegung

- Ziel: Alle Achsen starten und stoppen gleichzeitig
- Leitachse = Achse mit größter Bewegungsdauer, übrige Achsen werden auf gleiche Dauer verlangsamt
- Nicht alle Achsen müssen mit max. Geschwindigkeit fahren
- Bahn weicht oft weniger stark von direkter Verbindungslinie ab
- Langsamste Achse bestimmt Gesamtdauer



Vollsynchrone Bewegung

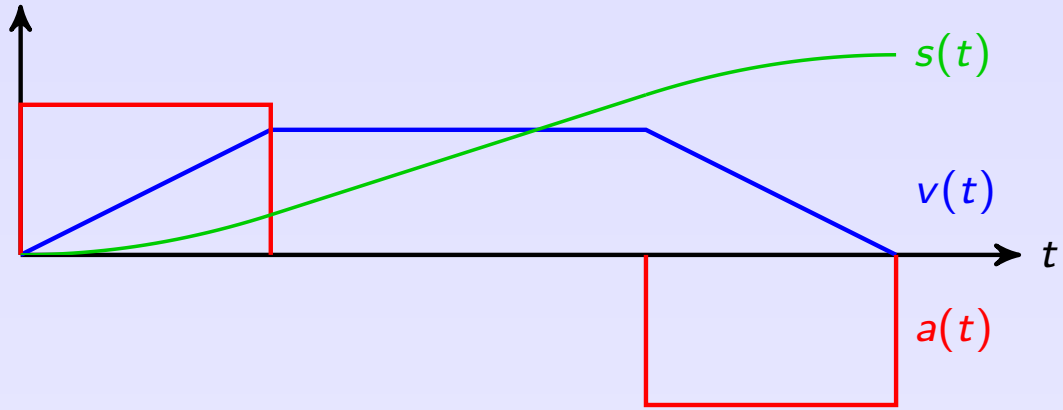
- Gesamtdauer und Beschleunigungsphasen an Leitachse angepasst



Bewegungsprofile I

Rampenprofil

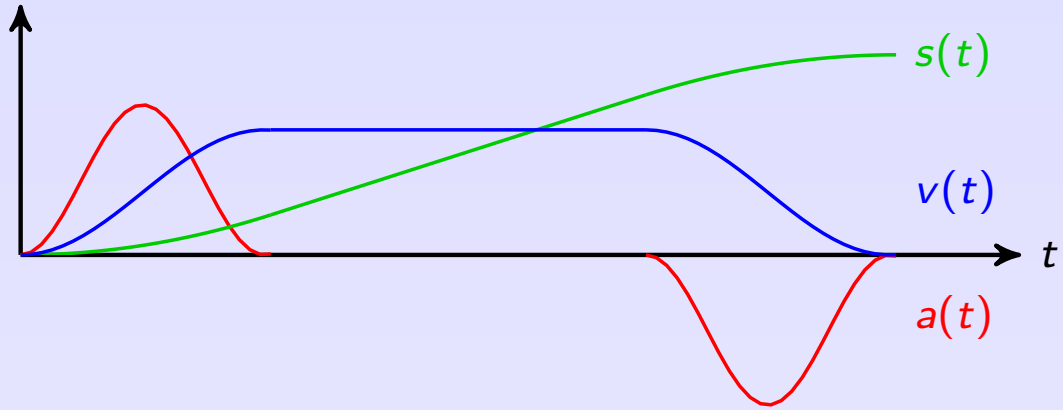
- konstante Beschleunigung auf maximale Geschwindigkeit, dann konstante Geschwindigkeit, dann konstante Negativbeschleunigung
- Beschleunigung und Kraft ($F = m \cdot a$) ändern sich sprunghaft/unstetig von Null auf Max und zurück
- hohe mechanische Belastung



Bewegungsprofile II

Sinoidenprofil

- stetige Beschleunigung nach $\sin^2$-Profil bis Maximum und zurück
- Veränderung der Kraft stetig/weich
- Bewegung ruckfrei/flüssig.



Polynome höherer Ordnung

- Ab Polynom 4. Ordnung werden ebenfalls weiche Bewegungen realisiert
- Kräfte werden als Polynom 2. Ordnung beschrieben
- dadurch weiche Übergänge zwischen beliebigen Geschwindigkeiten

Bewegungsart: CP - Continous Path

- Ziel: exakte Einhaltung einer vorgegebenen Bahn
- Aufwändige Berechnung der Bahnpunkte
- Viele Interpolationspunkte, dazwischen PTP-Bewegung
- Bahn i.d.R. Gerade oder Kreisbogen im karthesischen Raum $\rightarrow$ Kartesische Bahnsteuerung
- Auch beliebige (Spline-)Kurven möglich

Positionsüberschleifen von Zwischenstellungen

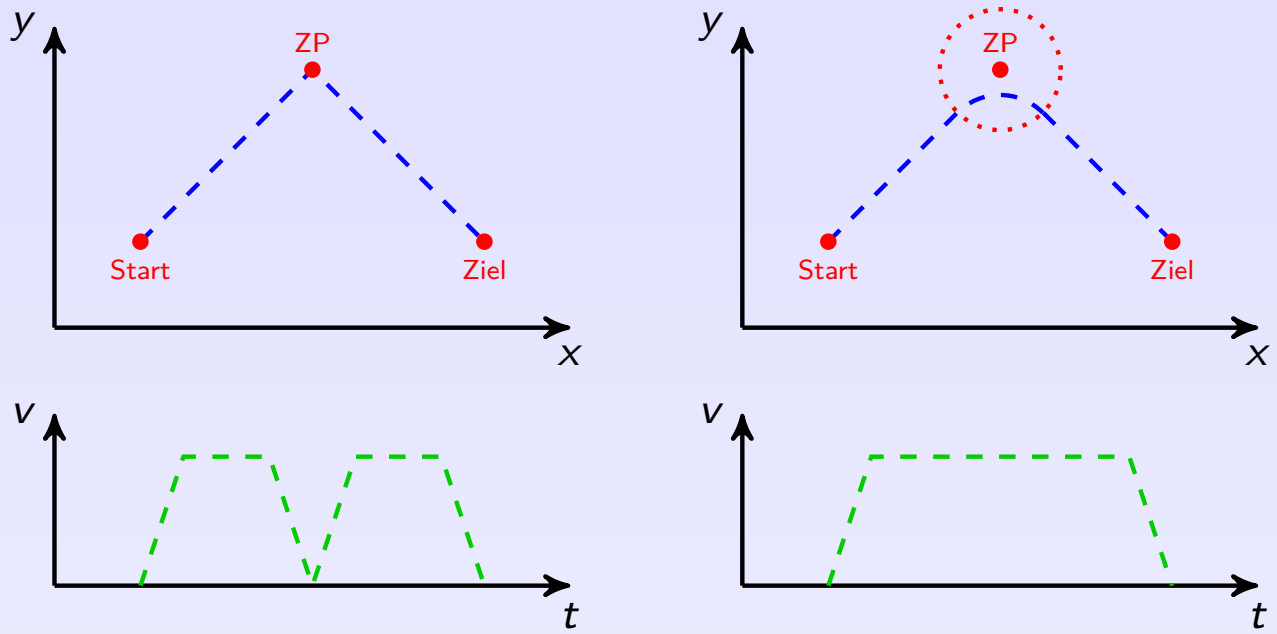
Zwischenpunkt mit Halt/Stillstand

- Werkzeugwechsel, Bearbeitung eines Werkstückes, Warten

Zwischenpunkt ohne Halt

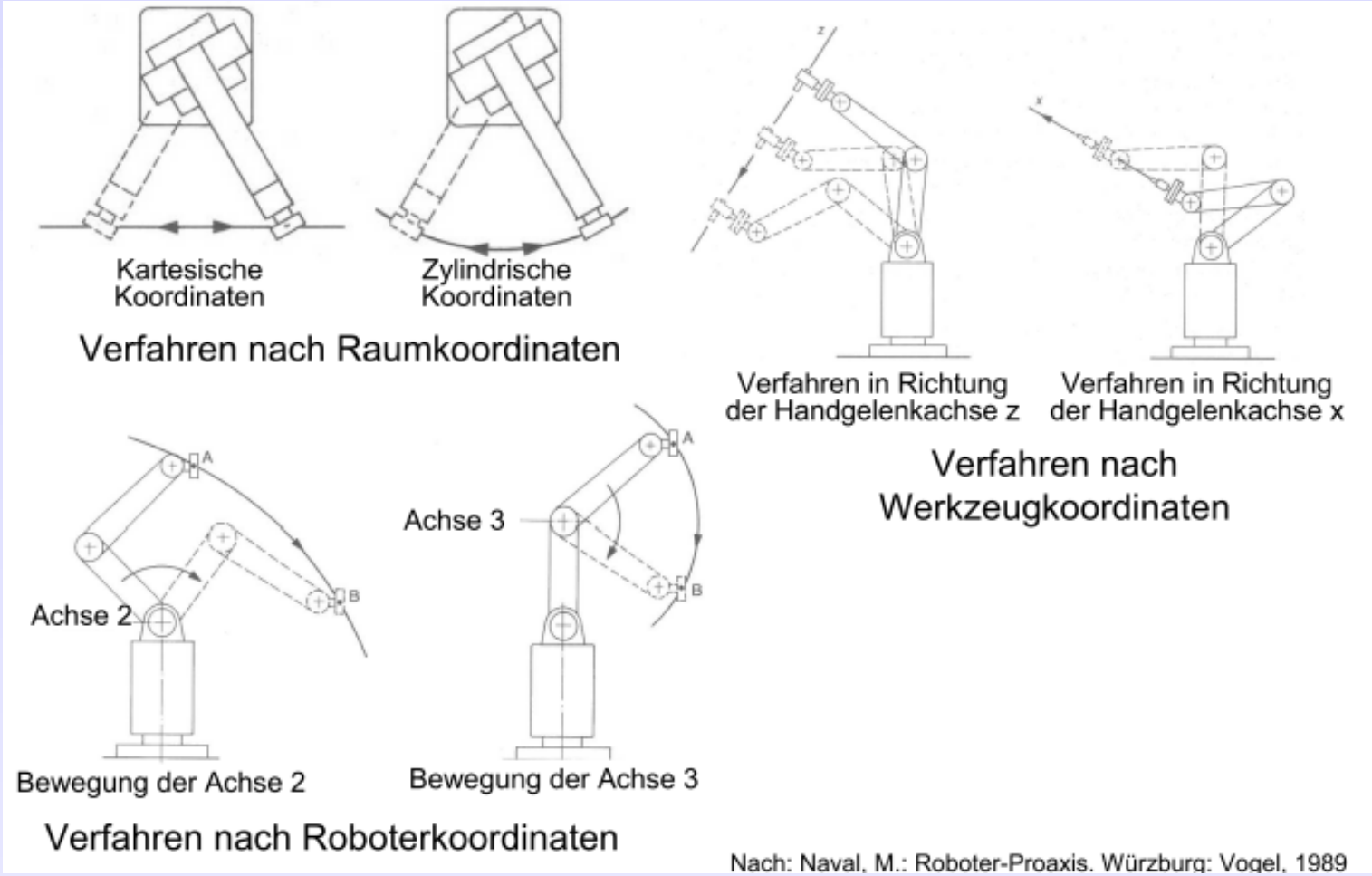
- Zwischenpunkt wurde eingefügt, um Hindernis zu umfahren
- Zwischenpunkt muss nicht exakt angefahren werden
- Stillstand unerwünscht
- abrupte Richtungswechsel, Abbrems- und Beschleunigungphasen vermeiden

→ Positionsüberschleifen, Überschleifkugel mit Radius r, Bewegung schneller



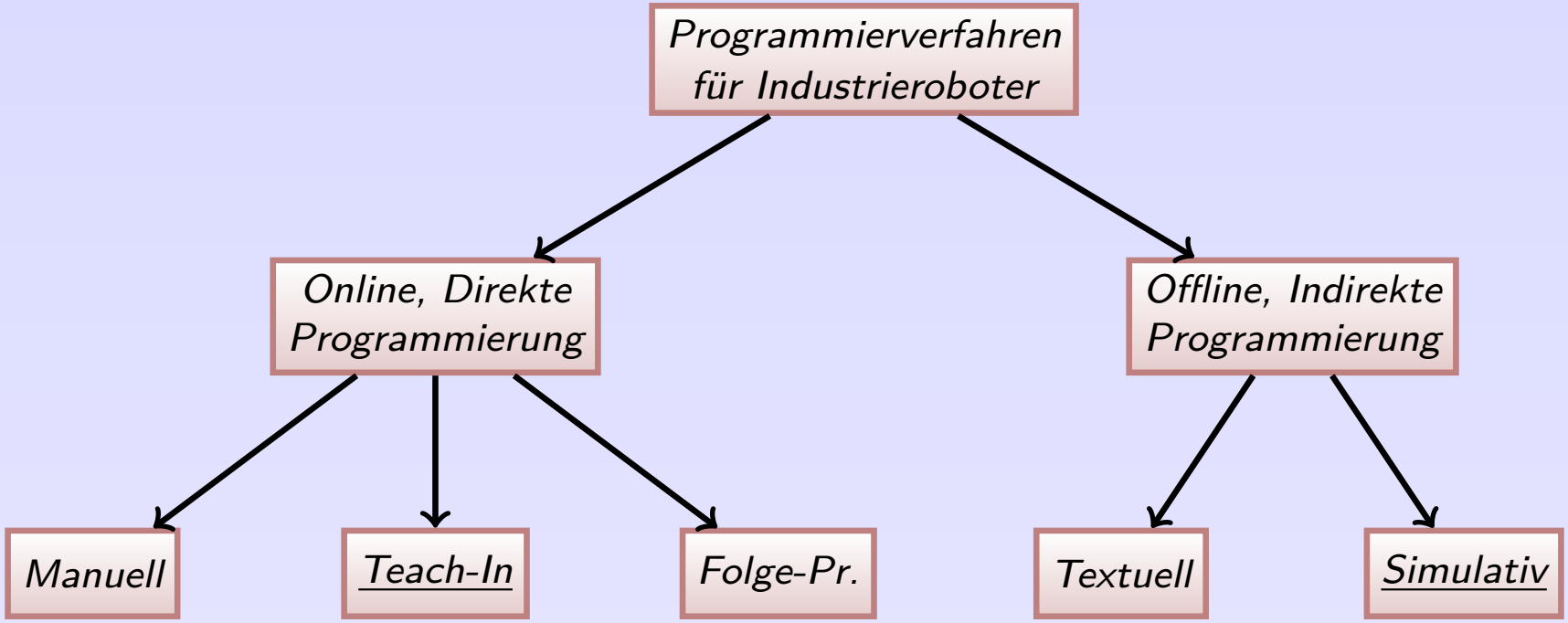
3. Programmierung

Koordinatensysteme für Roboterprogrammierung



[Lin15]

Programmierverfahren für Industrieroboter



Online/Direkte Programmierung: Roboter wird am Einsatzort programmiert und ist eingeschaltet (online)

Offline/Indirekte Programmierung: Programm wird am Schreibtisch erstellt und später auf Roboter portiert

Online-Programmierung I

- IR wird am Einsatzort programmiert und ist eingeschaltet (online)

+ Einfaches Verfahren, keine lange Einarbeitungszeit
+ Erfahrung des Bedienpersonals kann unmittelbar einfließen
+ Programmierung in Arbeitsumgebung ist anschaulich
+ Keine Vermessung von Umgebung, Werkstück oder Roboter
+ Sofortiges Testen und Korrigieren möglich

– Teure Stillstandzeiten, nur manuelle Genauigkeit
– Code schwer portierbar auf andere Robotertypen oder Hersteller
– Integration von Sensorik schwierig
– Erstellung komplexer Programme schwierig

Online-Programmierung II

Manuell

- Haltepunkte der Bewegungsachsen durch Anschläge $\rightarrow$ veraltet

Teach-In (häufigstes Verfahren)

- Manuelles Anfahren von Sollpositionen mit Programmiergerät
- Speichern von (Zwischen)-Stellungen mit Bewegungsparametern
  - Gelenkkoordinaten oder Pose des Effektors
  - Greiferzustand
  - Bewegungsart: PTP, CP-Linear, CP-zirkular
  - Halt oder Überschleifen, Überschleifradius
  - Geschwindigkeit
  - Beschleunigung
- Programm kann später nachbearbeitet werden (Wiederholungen)

Online-Programmierung III

Folgeprogrammierung

- Bediener bewegt Effektor direkt manuell
- Roboter zeichnet Bewegungsfolge auf
- nur bei Leichtbaurobotern möglich
- Anwendung: Lackieren

\+ Einfach, Intuitiv, kurze Einarbeitungszeit

Master-Slave-Programmierung

- Spezielle Folgeprogrammierung mit kleinem Modell (Master) des Roboters
- Roboter (Slave) folgt den Bewegungen des Modells

\+ Einfach, Intuitiv, kurze Einarbeitungszeit

– Relativ ungenaues Verfahren

Offline-Programmierung

Textuell

- Textuelle Befehle in herstellerspezifischer Programmiersprache

\+ Keine Stillstandzeiten, keine Hardware nötig

– Problem der genauen Koordinatenbestimmung

– Programmierfehler werden erst an Anlage erkannt

Simulativ

- Roboter, Werkstück, Werkzeug und Umgebung werden simuliert (Virtual Reality)
- Graphische, interaktive Programmerstellung in Simulation

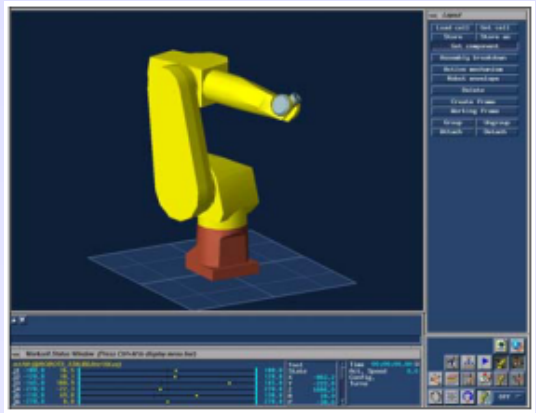
\+ keine Stillstände

\+ Gefahrloses Testen durch Modell

– Modellbildung u.U. aufwändig

– Teures Computersystem nötig

– Anpassung der erstellten Programme an reale Umgebung notwendig wegen Modellungenauigkeiten und -fehlern



[?]

Textuelle Programmierung

Problemorientiert/Explizit/Roboterorientiert

- Explizites Anfahren von Posen (kartesisch oder in Gelenkkoordinaten)
- Sichtweise: WIE soll sich Roboter bewegen?

Aufgabenorientiert/Implizit

- Höhere Abstraktionsebene, z.B. Wechsele Werkzeug, Greife Objekt, Füge zusammen
- Sichtweise: WAS soll Roboter tun?
- Benötigte Komponenten:
  - Sensorik, z.B. Kamera
  - Umweltmodell: Hindernisse, Werkstück, ...
  - Synchronisation: Roboteraktionen mit Zuständen der Umwelt
  - Aufgabenwissen: Automatische Erzeugung von expliziten Anweisungen aus impliziten Aufgaben

Simulative Programmierung

Modellierung der Roboterzelle

- Geometrisches CAD-Modell der Roboterzelle: Roboter, Werkstücke, Werkzeug, Umgebung, Hindernisse
- System enthält Auswahl gängiger Robotertypen und Umgebungen

Programmierung

- in roboterspezifischer Sprache oder systemeigener Programmiersprache, die dann in Code der jeweiligen Robotersteuerung übersetzt wird.
- Explizite Eingabe von Positionen und Bahnen oder virtuelles Teach-In
- Basierend auf CAD-Daten des Werkstücks $\rightarrow$ theoretisch hohe Arbeitsgenauigkeit

Animation des Modells

- Animation = dreidimensionale synthetische Laufbildsequenzen
- Realistische Darstellung des Programmablaufes aus beliebigen Entfernungen und Blickwinkeln
- Funktionen wie Zeitlupe, Zeitraffer, Schrittmodus und virtuelle Kamera an Roboterhand möglich
- Automatische, risikolose Kollisionsdetektion möglich
- Realitätsnahe Erreichbarkeits- und Taktzeitstudien möglich

Programmiersprachen für Roboter

Steuerprogramme für Robotersysteme können auf unterschiedlichen Sprachebenen dargestellt werden

- Maschinenorientierte Programmiersprachen (Assembler): VDI 2863, Blatt 1: 1987 Programmierung numerisch gesteuerter Handhabungseinrichtungen: IRDATA, Allgemeiner Aufbau, Satztypen und Übertragung. (Norm wurde zurückgezogen!)
- Problemorientierte Programmiersprachen: Industrial Robot Language (IRL).
- Objektorientierte Programmiersprachen
- Regelbasierte Programmiersprachen

Beispiele:

- IRL (Industrial Robot Language): herstellerunabhängige Programmiersprache für Industrieroboter, ab 1996 in DIN 66312, Norm wurde wieder zurückgezogen
- Rapid von ABB
- KRL (Kuka Robot Language) von Kuka
- VAL,V+ von Unimation/Stäubli

Versuch einer Normung bisher gescheitert, überwiegend Herstellersprachen

Grundelemente von Robotersprachen

- Vorgabe von Position und Orientierung, Bewegungsart (PTP,Linear,Zirkular), Geschwindigkeit, Beschleunigung
- Ansteuerung des Effektors: Greifer oder Werkzeug
- Schnittstelle zu Sensorsignalen für Messaufgaben und Synchronisieren
- Kontrollstrukturen: Verzweigungen, Wiederholungen, Unterprogramme
- Variablen, Felder, arithmetische Ausdrücke, logische Operationen
- Kommentare

Beispiel in Pseudocode

```
1    MOVE(20,10,0,0,0,0,PTP,UEB=NEIN)
2    TOOL ON
3    MOVE(30,10,0,0,0,0,LIN,UEB=NEIN,SPEED=10)
4    ...
5    TOOL OFF
6    HOME, PTP
```

Weitere Konzepte von Robotersprachen

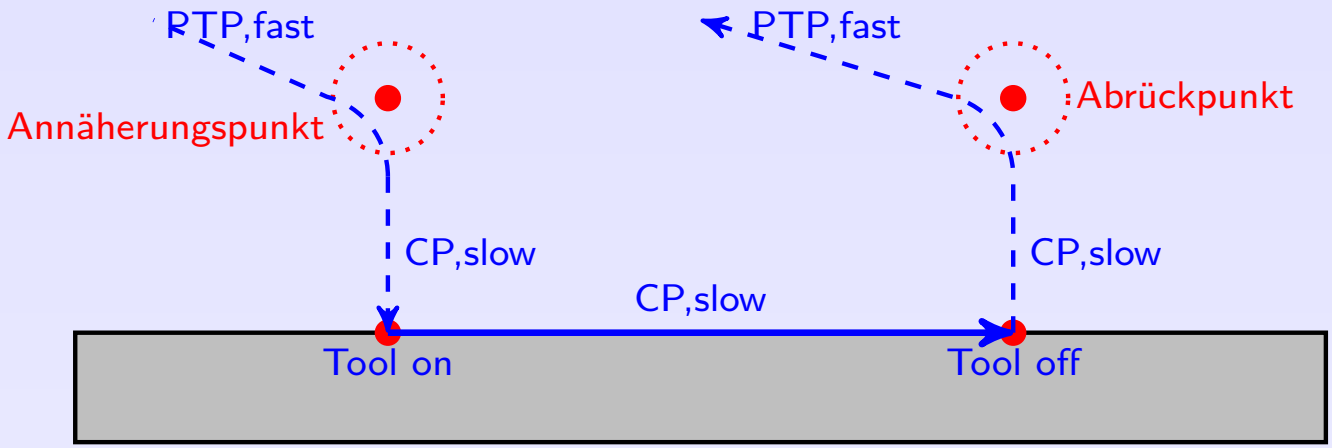
- Automatische Umrechnung von Welt- in Achskoordinaten
- Definition eigener Koordinatensysteme, z.B. Werkstück-KS
- Spezielle Datentypen: Vektoren, Homogene Matrizen für Rotation und Translation mit Konstruktoren
- Arithmetik für Matrizen
- Automatische Berechnung von An- und Abrückpunkten

3. Programmierung

Praxis: Annäherungspunkte, Bewegungsart

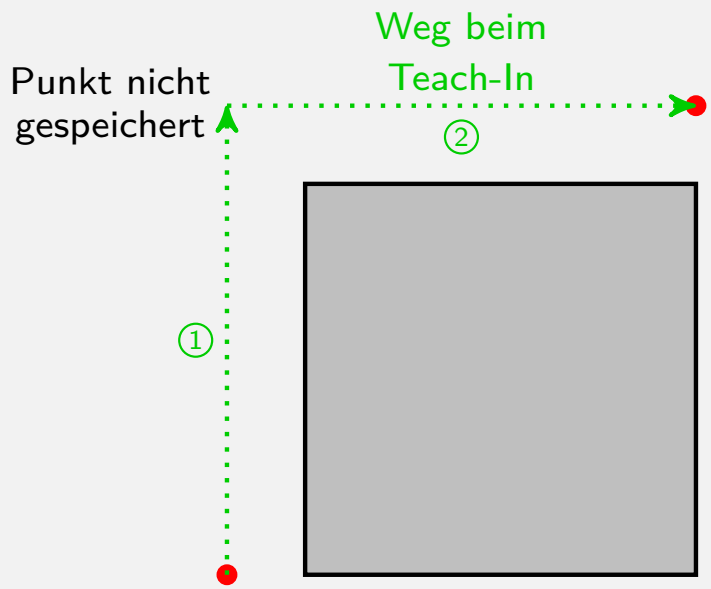
Übliche Bearbeitungsfolge eines Werkstückes, z.B. Lackieren, Fräsen, Schweißen o.ä.:

- Bewegung im freien Raum: schnelle PTP bis Annäherungspunkt
- Im Annäherungspunkt bereits richtige Orientierung/Greiferstellung
- Um Annäherungspunkt kleine Überschleifzone, kein Stopp
- Langsames, senkrechtes Anfahren an Werkstück auf CP-Bahn
- Bearbeitung: CP, kontrollierte Geschwindigkeit
- Fertig: langsames, senkrechtes Entfernen auf CP-Bahn bis Abrückpunkt
- Abrückpunkt ohne Stopp mit Überschleifen durchfahren, Wechsel auf hohe Geschwindigkeit und PTP



Praxis: Genügend Teachpunkte

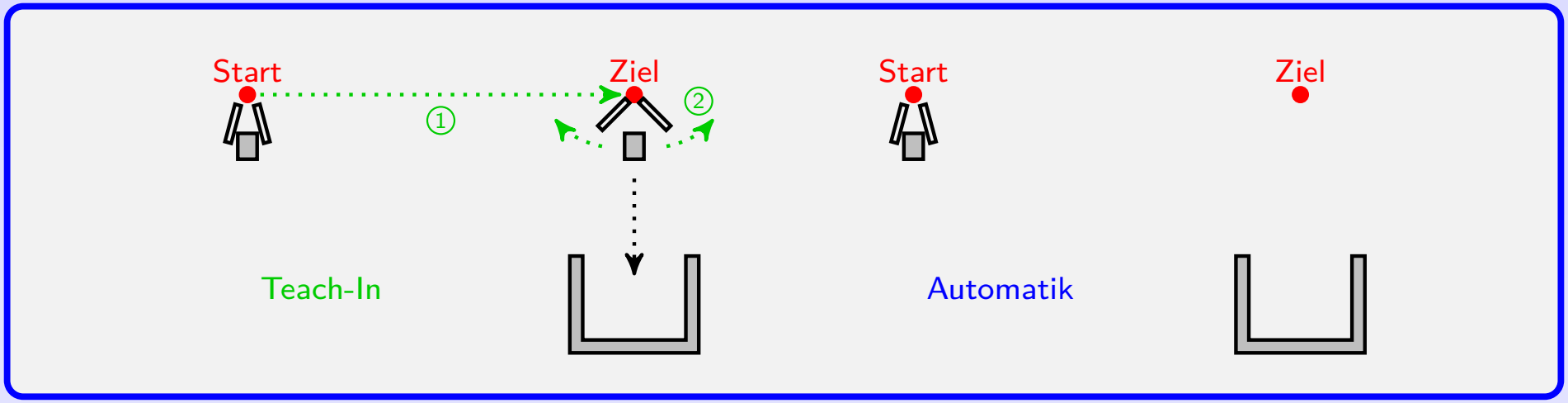
- Roboter fährt nur gespeicherte (!) Zwischenpunkte an
- Wurde Zwischenpunkt zur Kollisionsvermeidung manuell angefahren aber nicht gespeichert, kann es zu Crash kommen



- Auch Reduktion von Bahnabweichungen bei PTP durch zusätzliche Zwischenpunkte

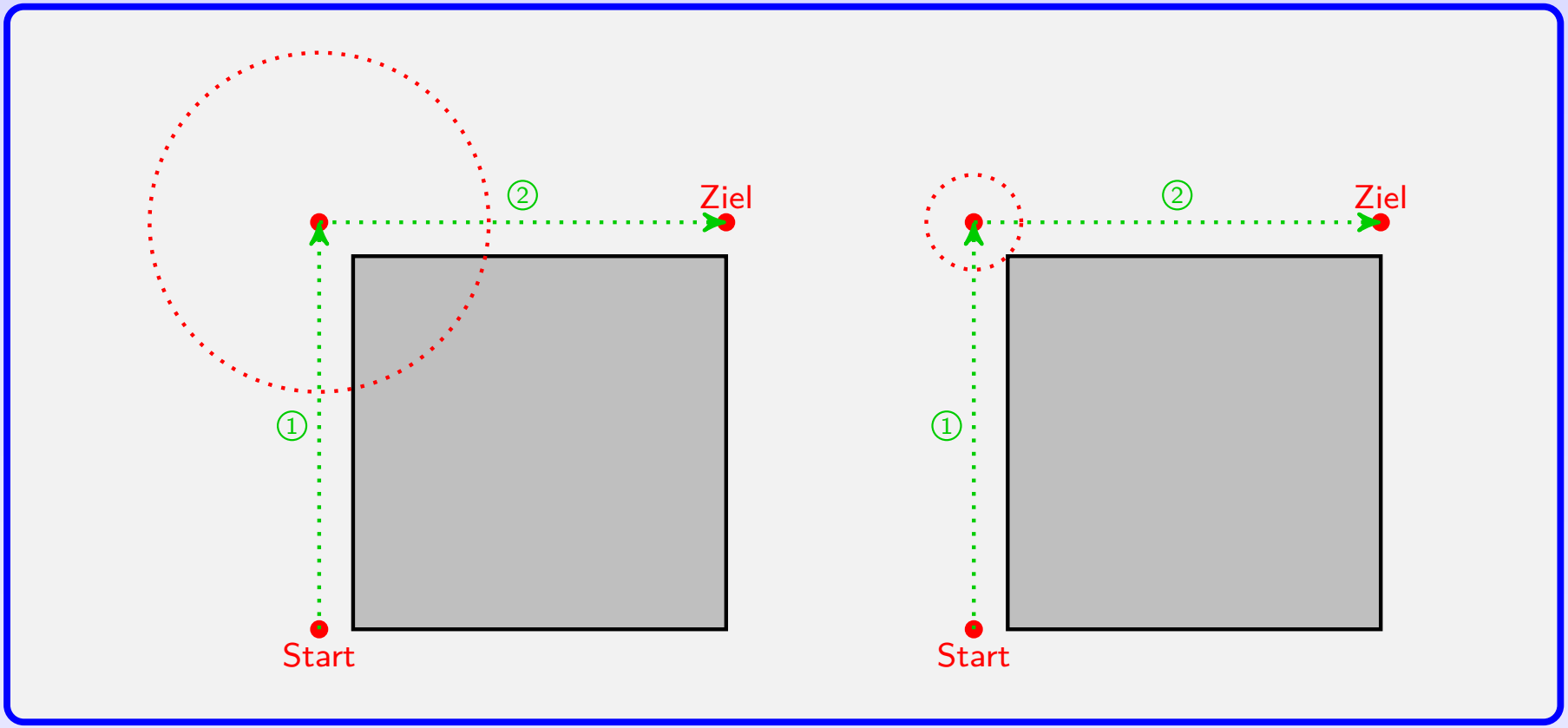
Praxis: Greifer synchronisieren

- Weitere Zwischenpunkte zur Synchronisation von Greifer/Tool mit Bewegung



Praxis: Größe der Überschleifzone

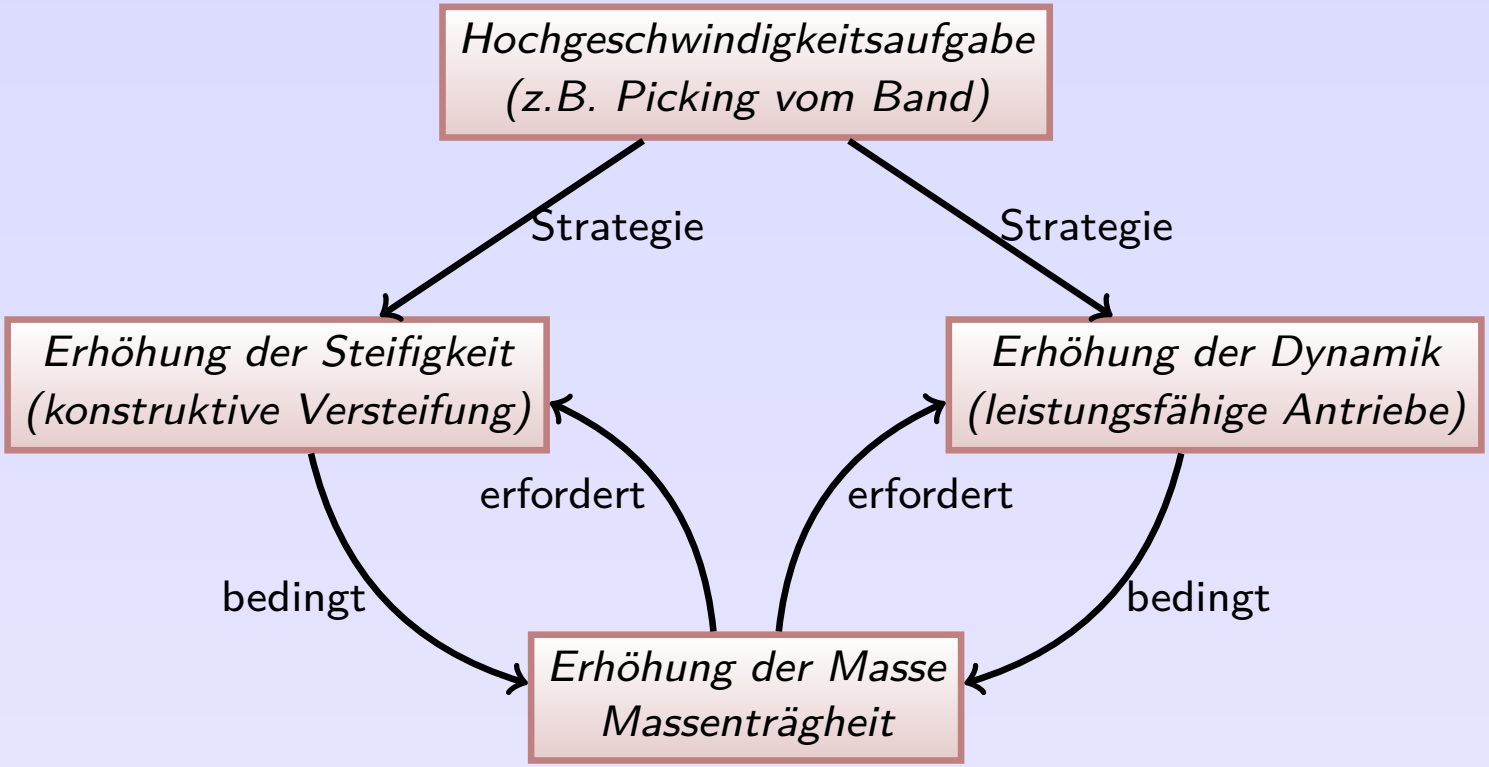
- Falls Überschleifzone zu groß, kann es zu Crash kommen



- Überschleifzone darf Hindernisse nicht überlappen

4. Industrieroboter mit Parallelkinematik

Motivation: Leistungsgrenzen serieller Roboter

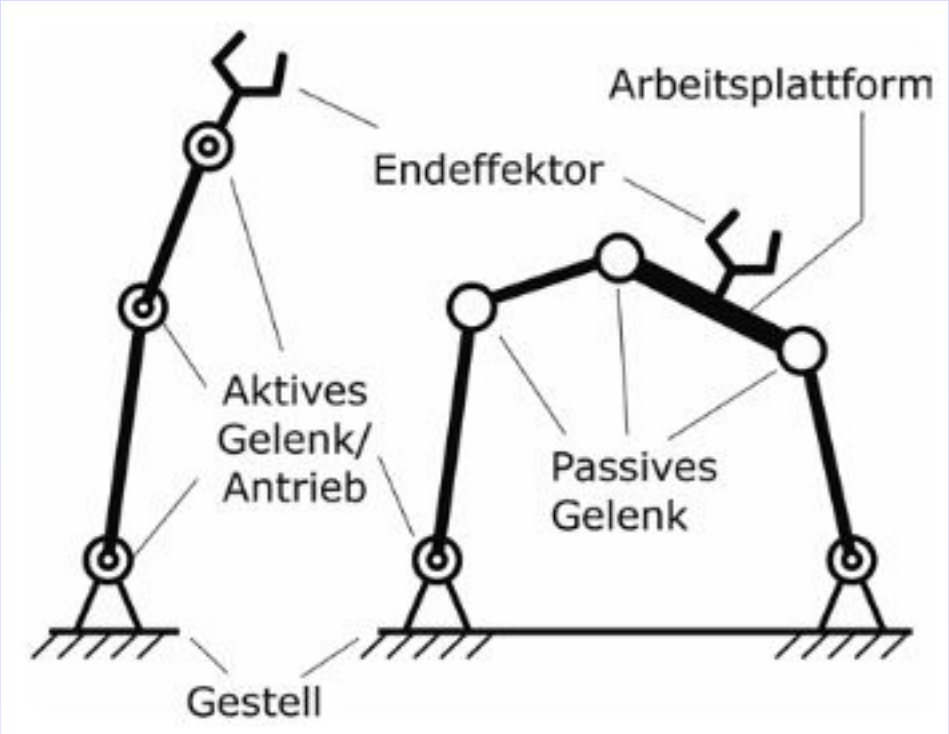


Weiteres Problem: beide Strategien sind kostenintensiv

Fazit:

- Geringe bewegte Masse → hohe Dynamik
- Hohe Steifigkeit → hohe Präzision
- Leistungsgrenzen von seriellen Kinematiken für hochdynamische Aufgaben

Vergleich: Seriell vs. Parallel



- Bei serieller Kinematik werden auch aktive Antriebe mitbewegt.
- Parallele Kinematik verwendet auch passive Gelenke
- Geschlossene Kette erhöht Steifigkeit und reduziert Schwingung

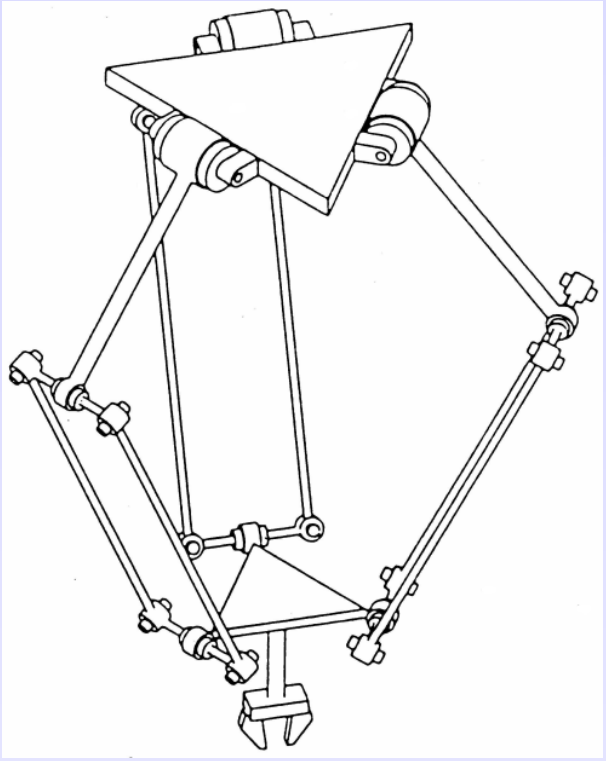
Eigenschaften Paralleler Strukturen:

+ hohe Steifigkeit, Wiederholgenauigkeit, Geschwindigkeit und Gewichtsreduktion
– beschränkter Arbeitsraum im Verhältnis zum Bauraum, oft geringer Schwenkbereich, hoher rechnerischer Aufwand

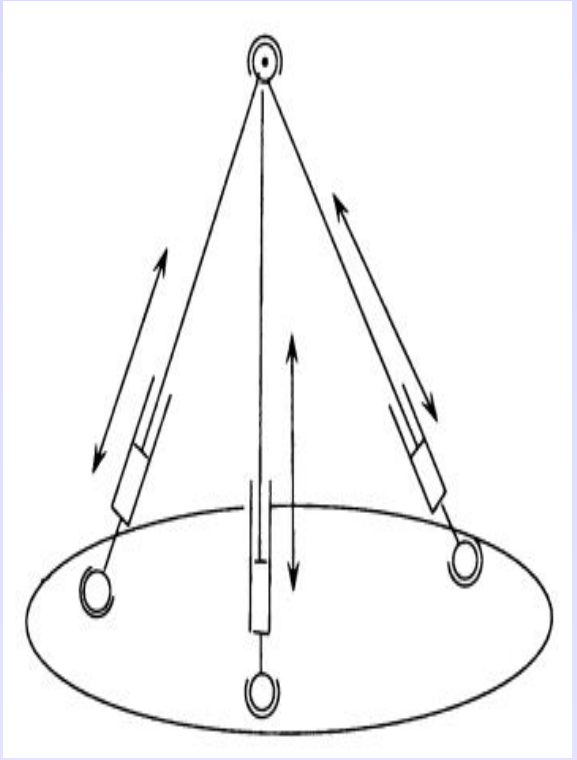
Vergleich: Seriell vs. Parallel

	Seriell	Parallel
Absolutgenauigkeit		besser
Wiederholgenauigkeit		besser
Steifigkeit		besser
Last/Massen-Verhältnis		besser
Arbeitsraum (Größe)	besser	
Hindernisvermeidung	besser	
Flexibilität (Anpassung an Aufgabe)	besser	
Berechnungsaufwand	einfacher	

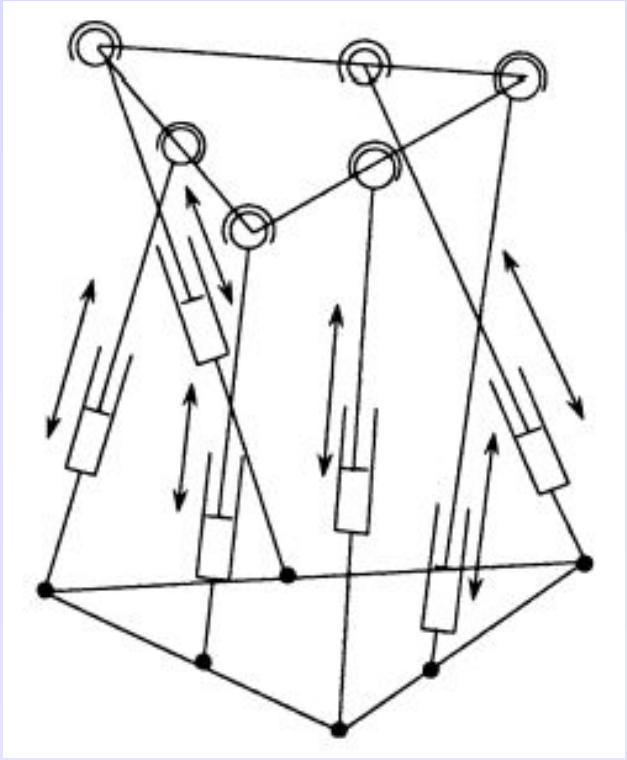
Typische Bauformen



Delta-Roboter
3 rotatorische Motoren
12 passive Gelenke
3 Freiheitsgrade
Delta-Roboter
2-Achs-Delta



Tripod
3 translatorische Antriebe
6 passive Gelenke
3 Freiheitsgrade
Tripod



Hexapod
6 translatorische Antriebe
12 passive Gelenke
6 Freiheitsgrade
Hexapod

Kennwerte

- Freiheitsgrade (für Pick-and-Place genügt meist $F = 3$)
- Arbeitsraum A / Bauraum B: $\eta = \frac{A}{B}$
- Geschwindigkeit des TCP
- Beschleunigung des TCP
- Kraft (max. Nutzlast)
- Massenträgheit
- Steifigkeit/Schwingungsverhalten
- Genauigkeit (Absolut$\sim$/Wiederhol$\sim$)
- Energetischer Wirkungsgrad (Energieverbrauch)

4. Industrieroboter mit Parallelkinematik

Delta-Roboter: Prinzip

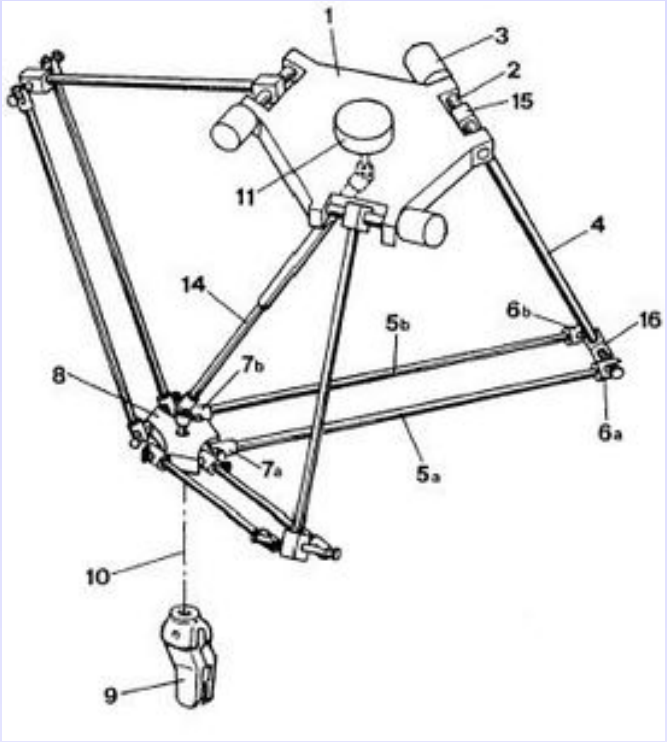
- meist hängend montiert, drei (identische) Arme um 120° versetzt
- angetriebener Oberarm an Grundplatte montiert, Unterarm verbindet passiv Oberarme mit Arbeitsplatte (Effektor)
- meist viergelenkige Unterarmstruktur aus parallelen Stäben unterdrückt rotatorische Freiheitsgrade → Arbeitsplatte parallel zur Grundplatte

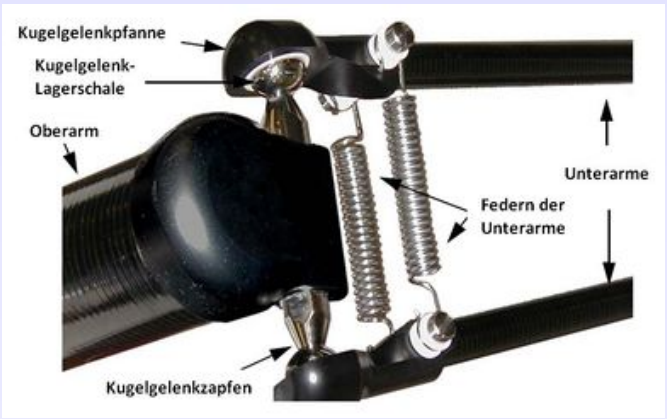
\+ Geringe bewegte Masse:
→ hohe Geschwindigkeit und Beschleunigung

– geringe Lastaufnahme, kleiner Arbeitsraum im Vergleich zum Bauraum

– komplexe Berechnung der translatorischen Bewegung aus rotatorischen Antrieben

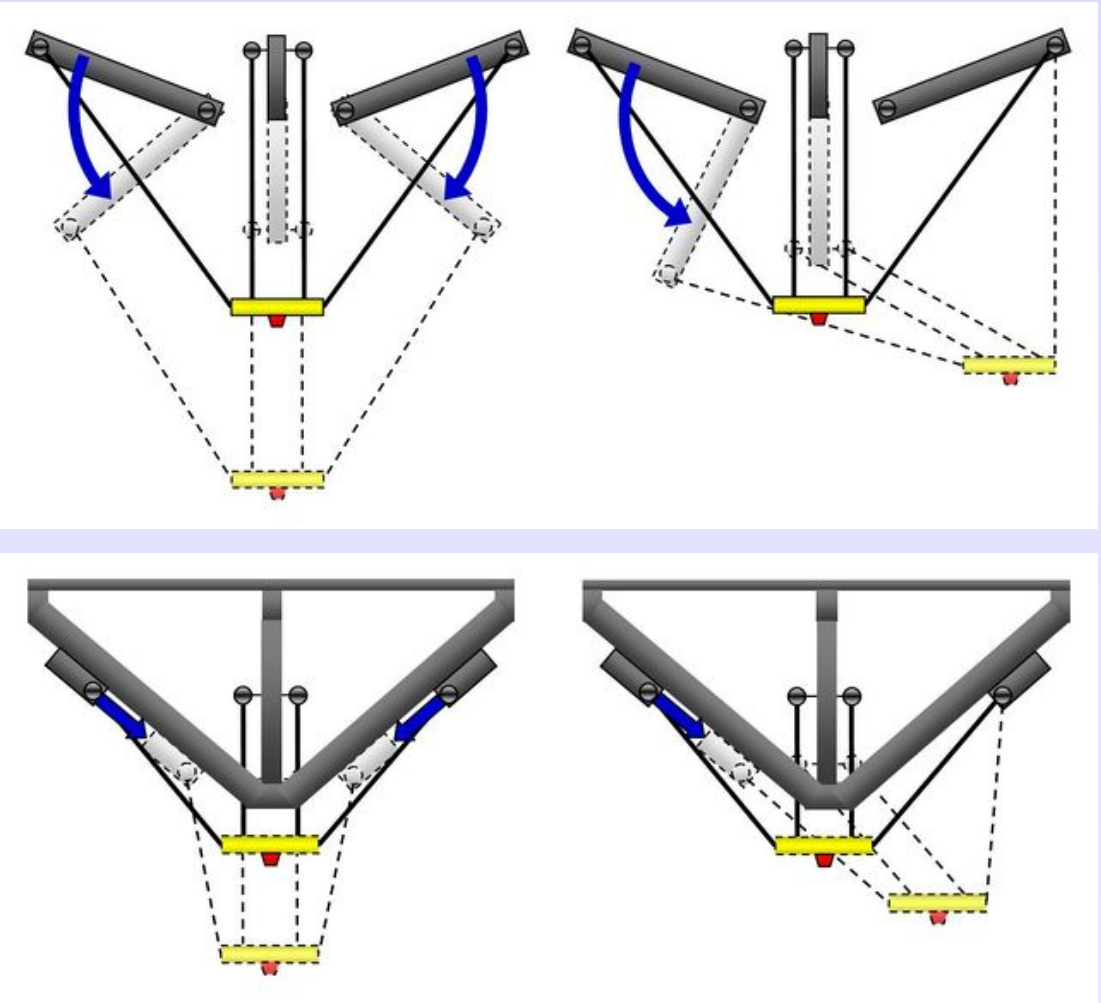
- Anwendung: Pick-and-Place, High-Speed
- Patent von R. Clavel
- Varianten: Linearantriebe, 4 Arme
- [Wey13]



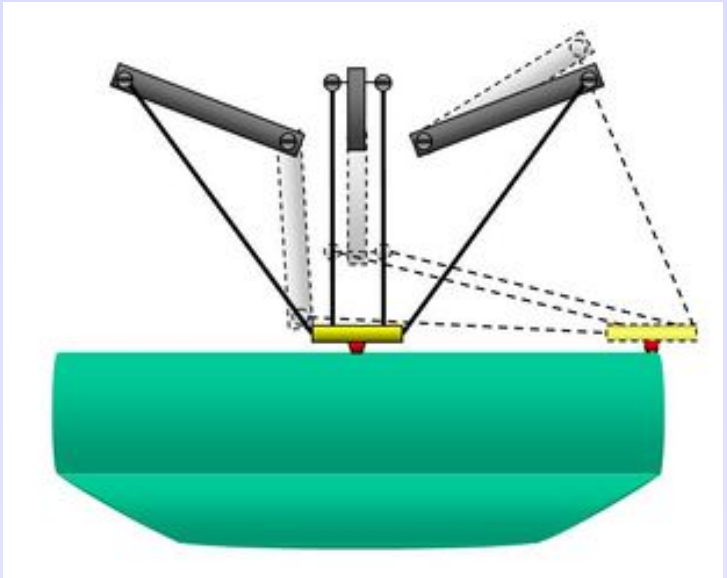


Delta-Roboter: Antriebe und Arbeitsraum [Wey13]

Rotatorische Antriebe und Linearantrieb



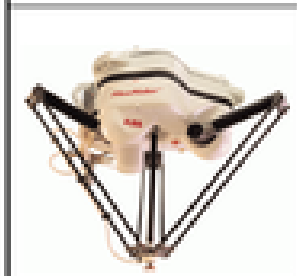
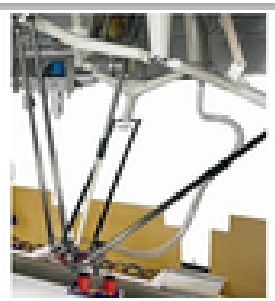
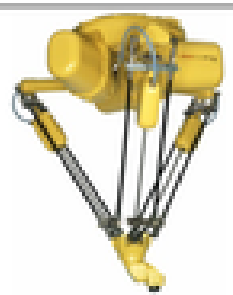
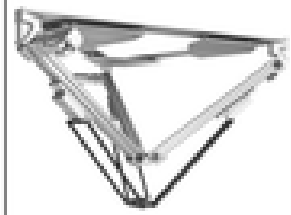
Arbeitsraum vs. Bauraum



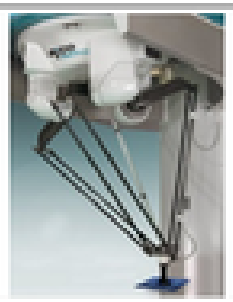
Delta-Roboter: Eigenschaften [Wey13] I

Vorteile	Nachteile
+ geringe bewegte Massen ➔ geringes Trägheitsmoment	- geringe Lastaufnahme
+ hohe Geschwindigkeiten und Beschleunigungen erreichbar ➔ hoher Durchsatz möglich	- kleiner Arbeitsraum im Verhältnis zum Bauraum
+ kompakte Motoren ➔ geringer Energiebedarf	- rotatorische Bewegung bzw. Schwenkwinkel beschränkt
+ hohe Steifigkeit durch parallele Struktur	- erhöhter Aufwand für Steuerung
+ hohe Wiederholgenauigkeit	
+ hohe Flexibilität	
+ hohe Zuverlässigkeit	
+ hohe Hygienefreundlichkeit	

Delta-Roboter: Überblick [Wey13] I

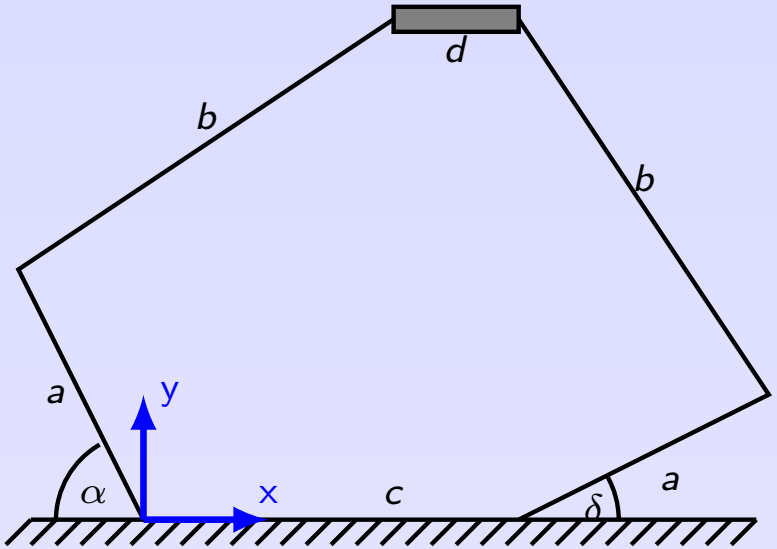
	ABB	adept	BOSCH	FANUC ROBOTICS	FESTO
					
Bezeichnung	**Flexpicker IRB 360**	**Quattro s650HS**	**Sigpack Systems Delta Roboter XR3**	**M-3iA**	**TRIPOD**
Kinematik	Delta-Kinematik	4-armige Parallelkinematik	Delta-Kinematik	Delta-Kinematik	Delta-Kinematik mit Linearantrieben
Anzahl Achsen	3 oder 4	4	4	4 oder 6	3 oder 4
Traglast	bis 3 kg	bis zu 6 kg	bis zu 2 kg	bis zu 6 kg	bis zu 5 kg
Durchmesser Arbeitsbereich	bis zu 1600 mm (bei 1 kg Traglast)	bis zu 1300 mm	bis zu 1200 mm	bis zu 1350 mm	bis zu 1250 mm
Beschleunigung max.	ca. 10 g	ca. 15 g	ca. 13 g	keine Angabe	ca. 10 g
Wiederholgenauigkeit	0,05 mm	0,1 mm	0,1 mm	0,1 mm	0,1 mm
Picks pro Minute	bis zu 200	bis zu 300	bis zu 185	keine Angabe	bis zu 150
Gewicht	120-145 kg	117 kg	75 kg	140-153 kg	150 kg
Anwendungen	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie
Quellen und weitere Informationen	www.abb.de	www.adept.de	www.boschpackaging.com	www.fanucrobotics.de	www.festo.de

Delta-Roboter: Überblick [Wey13] II

	FREI	Kawasaki	manz automation	Schneider Electric	SIGMATEK
					
Bezeichnung	**Asycube**	**YF003N**	**DR 1200**	**PacDrive P4**	**SDR 100**
Kinematik	Delta-Kinematik	Delta-Kinematik	Delta-Kinematik	Delta-Kinematik	Delta-Kinematik
Anzahl Achsen	3 oder 4	4	3 oder 4	3 oder 4	4
Traglast	bis zu 0,35 kg	keine Angabe	bis zu 2 kg	bis zu 1,5 kg	bis zu 1 kg
Durchmesser Arbeitsbereich	bis zu 350 mm	bis zu 1300 mm	bis zu 1200 mm	bis zu 1200 mm	bis zu 1100 mm
Beschleunigung max.	ca. 10 g	keine Angabe	ca. 5 g	ca. 10 g	ca. 10 g
Wiederholgenauigkeit	0,002 mm	0,1 mm	0,15 mm	0, 1mm	0,1 mm
Picks pro Minute	bis zu 240	bis zu 175	keine Angabe	bis zu 182	bis zu 150
Gewicht	keine Anagbe	135-145 kg	75 kg	keine Angabe	65 kg
Anwendungen	Mikrosystemtechnik Uhrenindustrie Medizintechnik	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie	Lebensmittelindustrie Pharmaindustrie Kosmetikindustrie	Lebensmittelindustrie Photovoltaikindustrie Elektronikindustrie
Qellen und weitere Informationen	www.frei-technik.de	www.kawasakirobot.de	www.manz-automation.com	www.elau.de	www.sigmatek-automation.com

Vorwärts-Kinematik eines 2D-Delta-Roboters (Geometrisch)

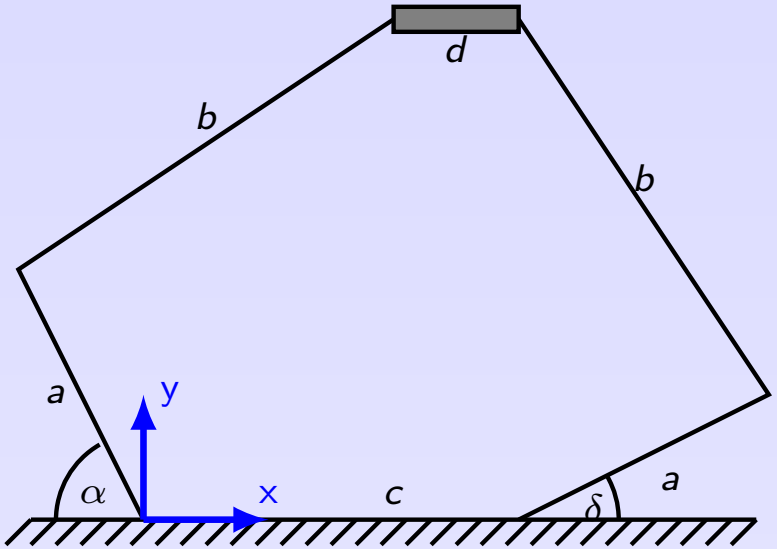
Geg.: Längen der symmetrischen Arme (a, b), Winkel der Oberarme (α, δ), Geometrie von Grund- und Arbeitsplatte (c, d)

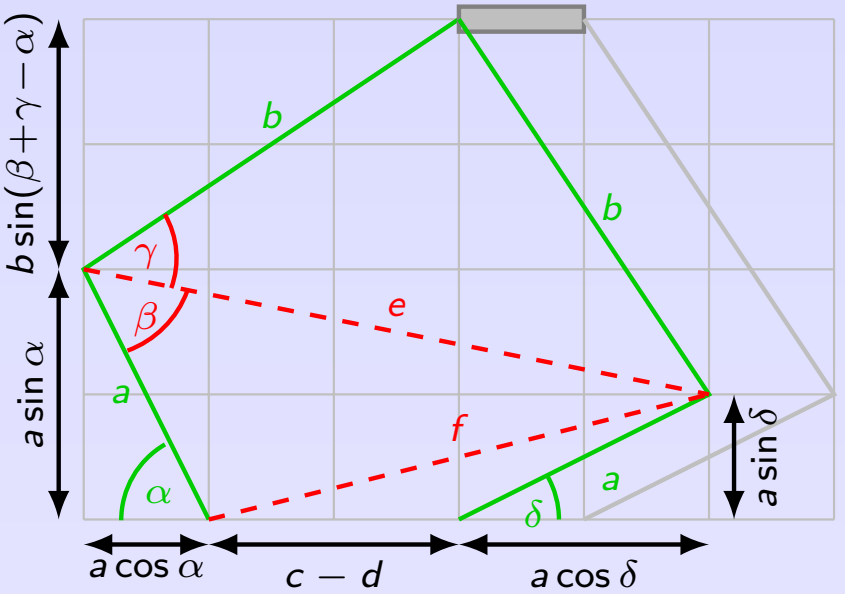


1. KO-Ursprung in Gelenk 0 eines Armes legen

Vorwärts-Kinematik eines 2D-Delta-Roboters (Geometrisch)

Geg.: Längen der symmetrischen Arme (a, b), Winkel der Oberarme (α, δ), Geometrie von Grund- und Arbeitsplatte (c, d)

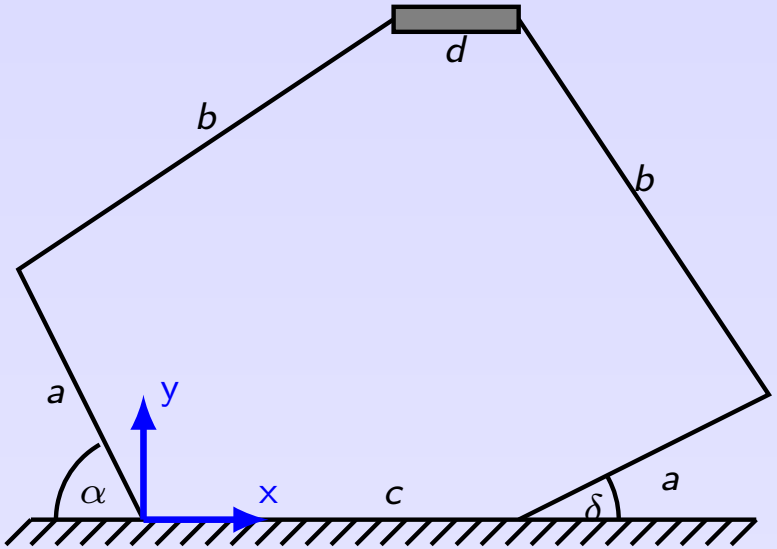


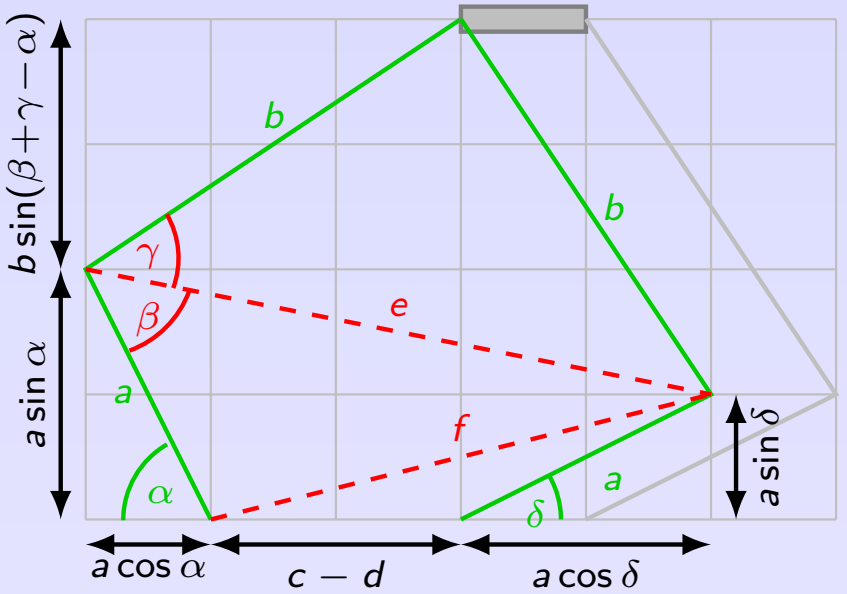


1. KO-Ursprung in Gelenk 0 eines Armes legen

Vorwärts-Kinematik eines 2D-Delta-Roboters (Geometrisch)

Geg.: Längen der symmetrischen Arme (a, b), Winkel der Oberarme (α, δ), Geometrie von Grund- und Arbeitsplatte (c, d)



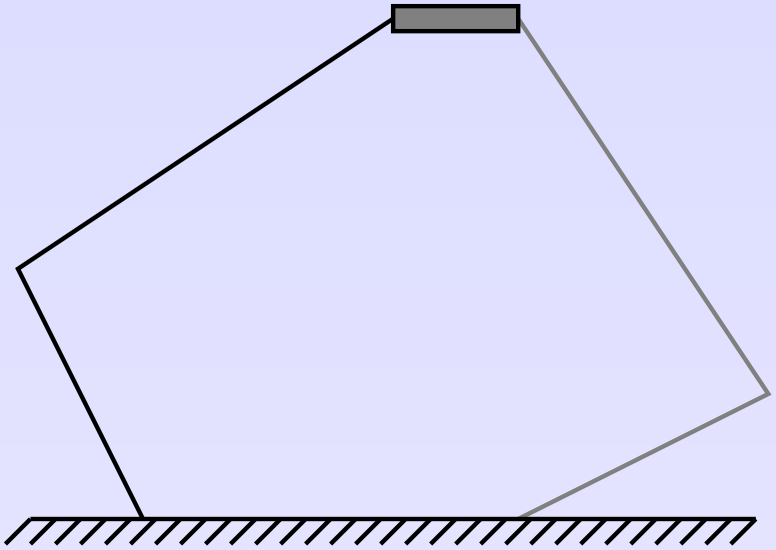


1. KO-Ursprung in Gelenk 0 eines Armes legen
2. Translation des anderen Armes um Länge d der Arbeitsplatte
3. $f = \sqrt{((c-d) + a\cos\delta)^2 + (a\sin\delta)^2}$ | Pythagoras
4. $e = \sqrt{(a\cos\alpha + c - d + a\cos\delta)^2 + (a\sin\alpha - a\sin\delta)^2}$ | Pythagoras
5. $\cos\beta = \frac{a^2+e^2-f^2}{2ae}$ | Kosinussatz
6. $\cos\gamma = \frac{b^2+e^2-b^2}{2be} = \frac{e}{2b}$ | Kosinussatz
7. $y = a\sin\alpha + b\sin(\beta+\gamma-\alpha)$

Vorwärts-Kinematik eines 2D-Delta-Roboters (DH-Trafo)

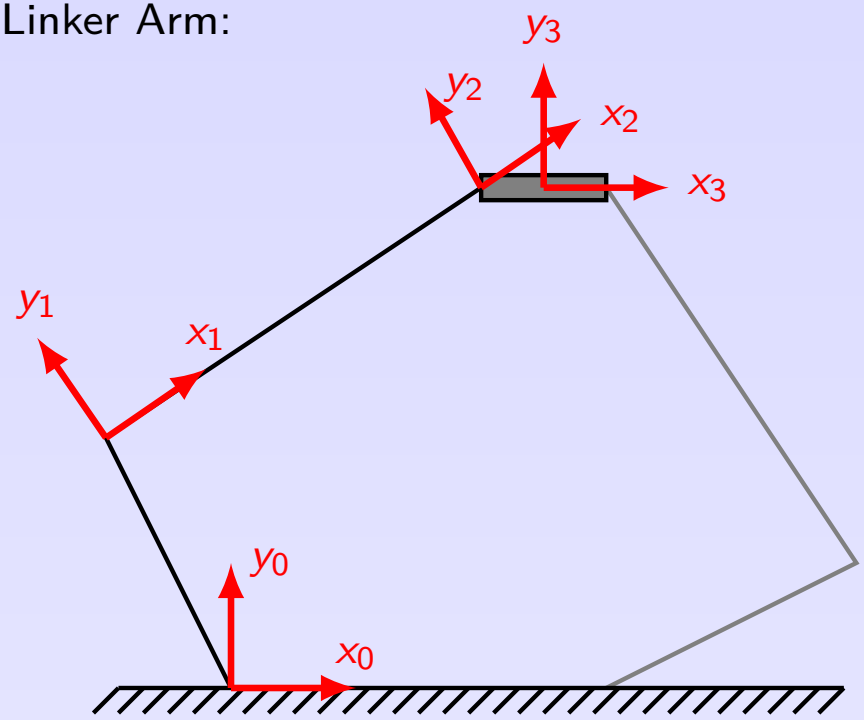
Linker Arm:

1. DH-KO festlegen



Vorwärts-Kinematik eines 2D-Delta-Roboters (DH-Trafo)

Linker Arm:

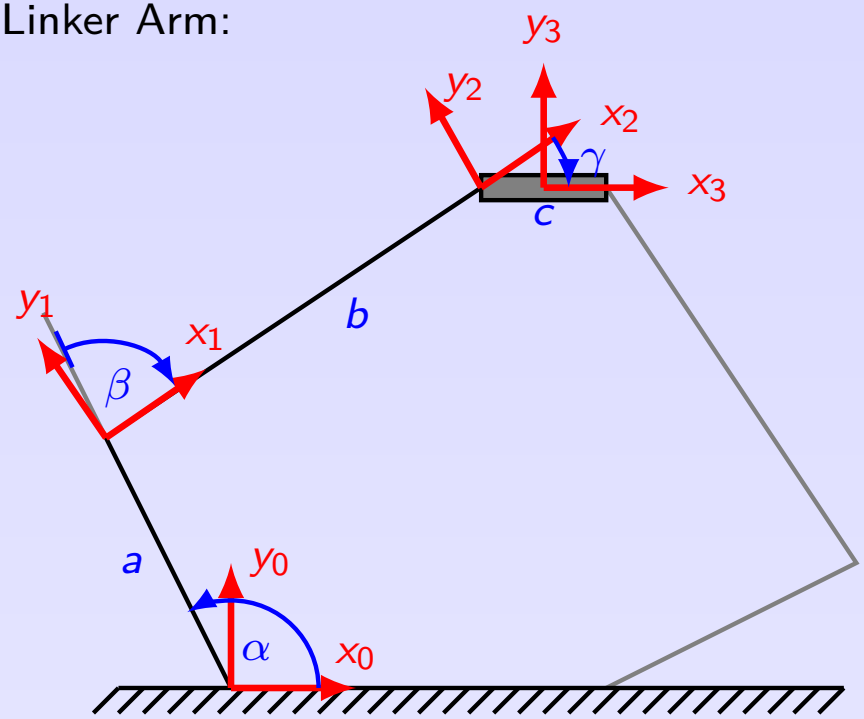


1. DH-KO festlegen
2. DH Parameter bestimmen

	Rot_x	Tr_x	Tr_z	Rot_z
1				
2				
3				

Vorwärts-Kinematik eines 2D-Delta-Roboters (DH-Trafo)

Linker Arm:



1. DH-KO festlegen
2. DH Parameter bestimmen

	Rot_x	Tr_x	Tr_z	Rot_z
1				
2				
3				

3. Transformationsgleichung A_L aufstellen

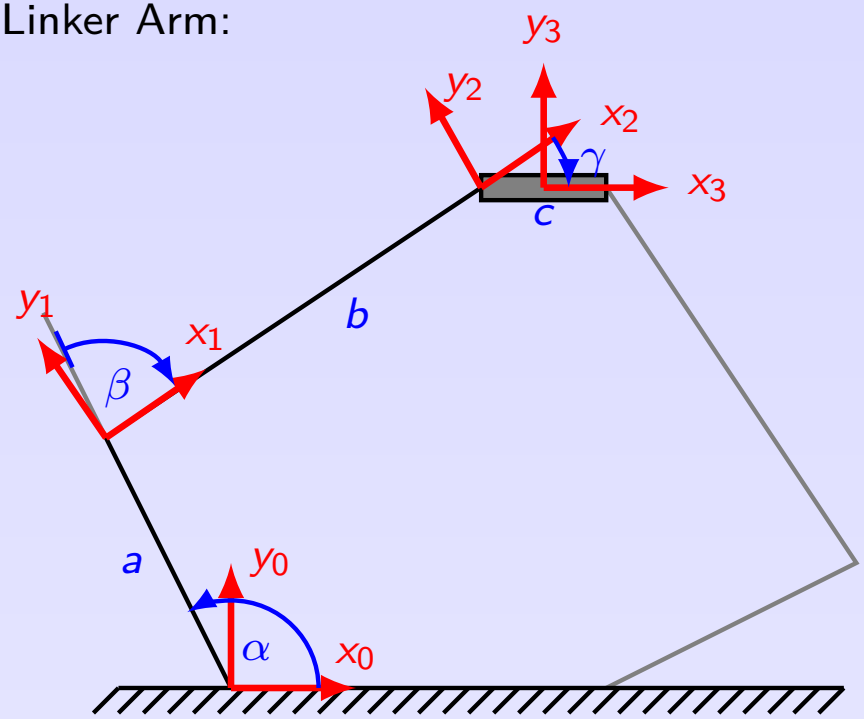
$$A_1 = \text{Tr}_x(a)\text{Rot}_z(\alpha) = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\alpha & -s_\alpha & 0 & 0 \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_\alpha & -s_\alpha & 0 & a \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit $s_\alpha = \sin\alpha$ und $c_\alpha = \cos\alpha$

$$A_L = A_3 A_2 A_1 = \begin{pmatrix} c_\gamma & -s_\gamma & 0 & \frac{c}{2} \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\beta & -s_\beta & 0 & b \\ s_\beta & c_\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\alpha & -s_\alpha & 0 & a \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Vorwärts-Kinematik eines 2D-Delta-Roboters (DH-Trafo)

Linker Arm:



1. DH-KO festlegen
2. DH Parameter bestimmen

	Rot_x	Tr_x	Tr_z	Rot_z
1				
2				
3				

3. Transformationsgleichung A_L aufstellen
4. Analog für rechten Arm $\rightarrow A_R$
5. Durch Koeffizientenvergleich von A_L und A_R Unbekannte (β) bestimmen und in Transformationsgleichung einsetzen

$$A_1 = \text{Tr}_x(a)\text{Rot}_z(\alpha) = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\alpha & -s_\alpha & 0 & 0 \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_\alpha & -s_\alpha & 0 & a \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit $s_\alpha = \sin\alpha$ und $c_\alpha = \cos\alpha$

$$A_L = A_3 A_2 A_1 = \begin{pmatrix} c_\gamma & -s_\gamma & 0 & \frac{c}{2} \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\beta & -s_\beta & 0 & b \\ s_\beta & c_\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_\alpha & -s_\alpha & 0 & a \\ s_\alpha & c_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_L = \begin{pmatrix} c_\gamma & -s_\gamma & 0 & \frac{c}{2} \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\alpha c_\beta - s_\alpha s_\beta & -s_\alpha c_\beta - c_\alpha s_\beta & 0 & ac_\beta + b \\ c_\alpha s_\beta + s_\alpha c_\beta & -s_\alpha s_\beta + c_\alpha c_\beta & 0 & as_\beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \dots$$

$$\begin{pmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\beta c_\gamma - c_\alpha s_\beta s_\gamma + s_\alpha c_\beta s_\gamma & -s_\alpha c_\beta c_\gamma - c_\alpha s_\beta c_\gamma + s_\alpha s_\beta s_\gamma - c_\alpha c_\beta s_\gamma & 0 & c_\gamma(ac_\beta + b) - as_\beta s_\gamma + \frac{c}{2} \\ c_\alpha c_\beta s_\gamma - s_\alpha s_\beta s_\gamma + c_\alpha s_\beta c_\gamma + s_\alpha c_\beta c_\gamma & -s_\alpha c_\beta s_\gamma - c_\alpha s_\beta s_\gamma - s_\alpha s_\beta c_\gamma + c_\alpha c_\beta c_\gamma & 0 & s_\gamma(ac_\beta + b) + as_\beta c_\gamma \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Konstruktionsbedingt $\gamma = \alpha - \beta$, α bekannt $\Rightarrow$ 1 Unbekannte (β)
Analog: $A_R = \dots \Rightarrow$ 1 Unbekannte
Koeffizientenvergleich von A_L und $A_R \Rightarrow \beta = \dots$
Berechnung des TCP (KO_3)

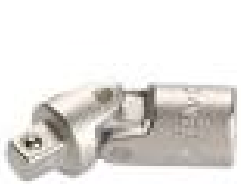
4. Industrieroboter mit Parallelkinematik

Konstruktionsmöglichkeiten

- Anzahl der Freiheitsgrade
- Veränderliche Stablängen oder Fußpunkte
- Translatorische oder rotatorische Antriebe
- Rein parallele oder hybride Struktur (z.B. Delta-Roboter mit Zentralhand)

Gelenkarten

	Gelenkart	F	Beispiel
D_1	Drehgelenk, Scharnier	1	Türscharnier
D_2	Kardangelenk, Kreuzgelenk	2	Handgelenk
D_3	Kugelgelenk	3	Schulter
S_1	Schubgelenk	1	Schubkastenauszug
TTR	Plattengelenk	3	Tischbein auf Boden
TR	Drehschubgelenk	2	Enger Ring auf Stange




Getriebefreiheitsgrad, Gleichung nach Grübler

- (Getriebe-)Freiheitsgrad F einer Anordnung: statisch bestimmt $F = 0$, überbestimmt $F < 0$, unterbestimmt $F > 0$
- Unterbestimmung: Bewegungen können eingeleitet werden
- Bei $F = 1$ sind alle Gliedlagen bestimmt, gdw. Antrieb bestimmt ist
- Ein Getriebe ist *zwangläufig*, wenn Stellung der Antriebsglieder die Stellungen der übrigen Glieder eindeutig festlegt
- Gleichung nach Martin Fürchtegott Grübler (1917) beschreibt Beweglichkeit von durch Gelenke verbundene Getriebeteile
- Unterscheide Bewegungen/Getriebetyp in Ebene, Sphäre oder Raum

$$F = T \cdot (n - 1 - g) + \sum_{i=1}^{g} b_i = T \cdot (n - 1) - \sum_{i=1}^{g} (T - b_i) \tag{1}$$

T Getriebetyp (Raum T=6, Sphäre oder Ebene T=3), n Anzahl der Getriebeglieder, g Anzahl der Gelenke, b_i: Beweglichkeit des Gelenks i (z.B. Dreh- oder Schubgelenk $b = 1$)
Anschauung: Jedes Glied (außer Gestell) besitzt 6 FG $\rightarrow$ 6(n-1)
Jedes Gelenk i nimmt 6 FG minus seiner Beweglichkeit $b_i \rightarrow -\sum(6 - b_1)$

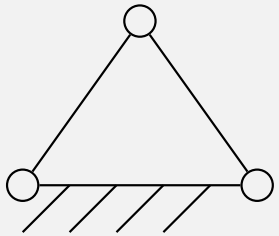
Getriebefreiheitsgrad: Beispiele

1. Beweglich verbundenes Fachwerk mit 3 Gliedern in der Ebene

Getriebetyp $T=$
Getriebeglieder $n=$
Gelenke $g=$
Beweglichkeit $\Sigma b_i=$

$$F = T(n-1-g) + \sum b_i =$$

$\Rightarrow$

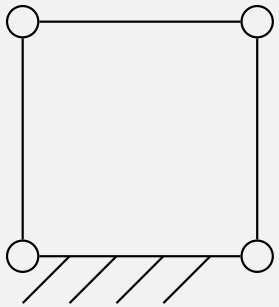


2. Beweglich verbundene Anordnung mit 4 Gliedern in der Ebene

Getriebetyp $T=$
Getriebeglieder $n=$
Gelenke $g=$
Beweglichkeit $\Sigma b_i=$

$$F =$$

$\Rightarrow$



3. Delta-Roboter mit 3 aktiven Drehgelenken und 6 passiven Kardangelenken

Getriebetyp $T=$

Getriebeglieder $n=$

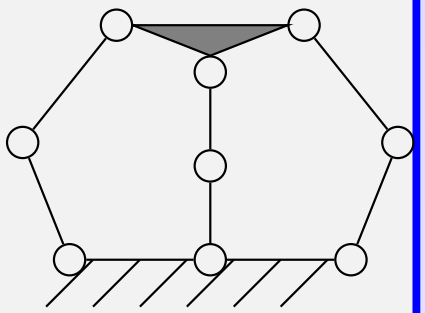
Gelenke $g=$

Beweglichkeit $b_{1\ldots3}=$

Beweglichkeit $b_{4\ldots9}=$

Beweglichkeit $\Sigma b_i=$

$b_{1\ldots3} = 1,\ b_{4\ldots9} = 2$

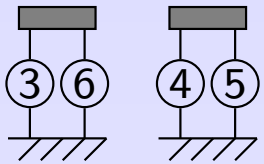
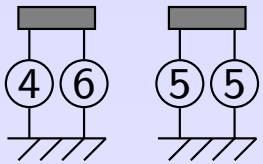
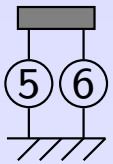
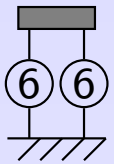
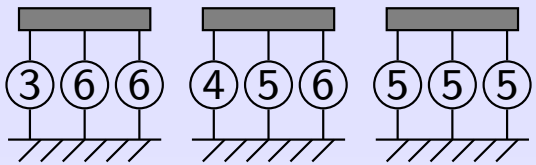
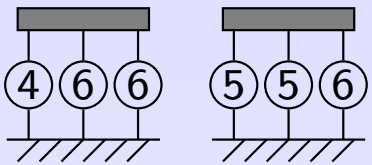
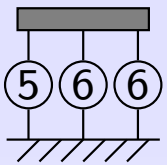
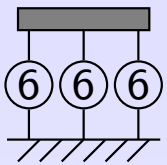
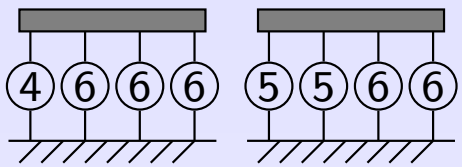
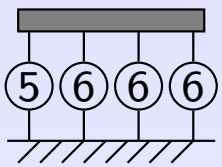
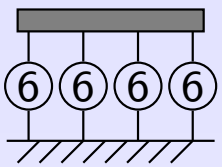
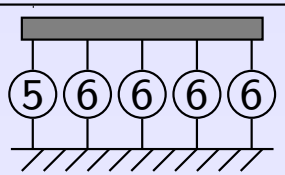
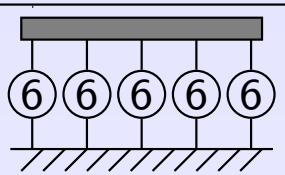
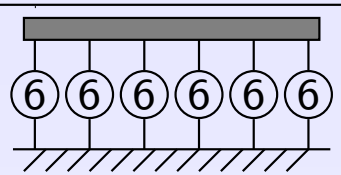


$$F = T(n-1-g) + \sum b_i =$$

$\Rightarrow$

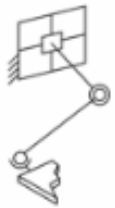
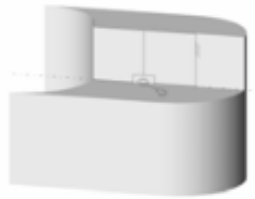
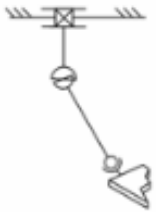
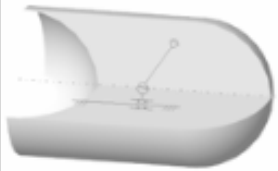
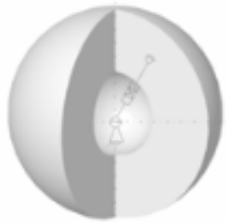
Verteilung der Gelenkfreiheiten

Geg.: gewünschter Freiheitsgrad F der Arbeitsplatte und Anzahl k der Gelenkketten
Ges.: Notwendige Freiheitsgrade F_i der einzelnen Gelenkketten
1. $F_i \geq F$, andernfalls wäre Beweglichkeit der AP eingeschränkt
2. Aus $F = T(n-1-g) + \sum b_i$ und $n = 2+g-k$ folgt $\sum b_i = F + T(k-1)$

k	$F = 3$	4	5	6
2				
3				
4				
5				
6				

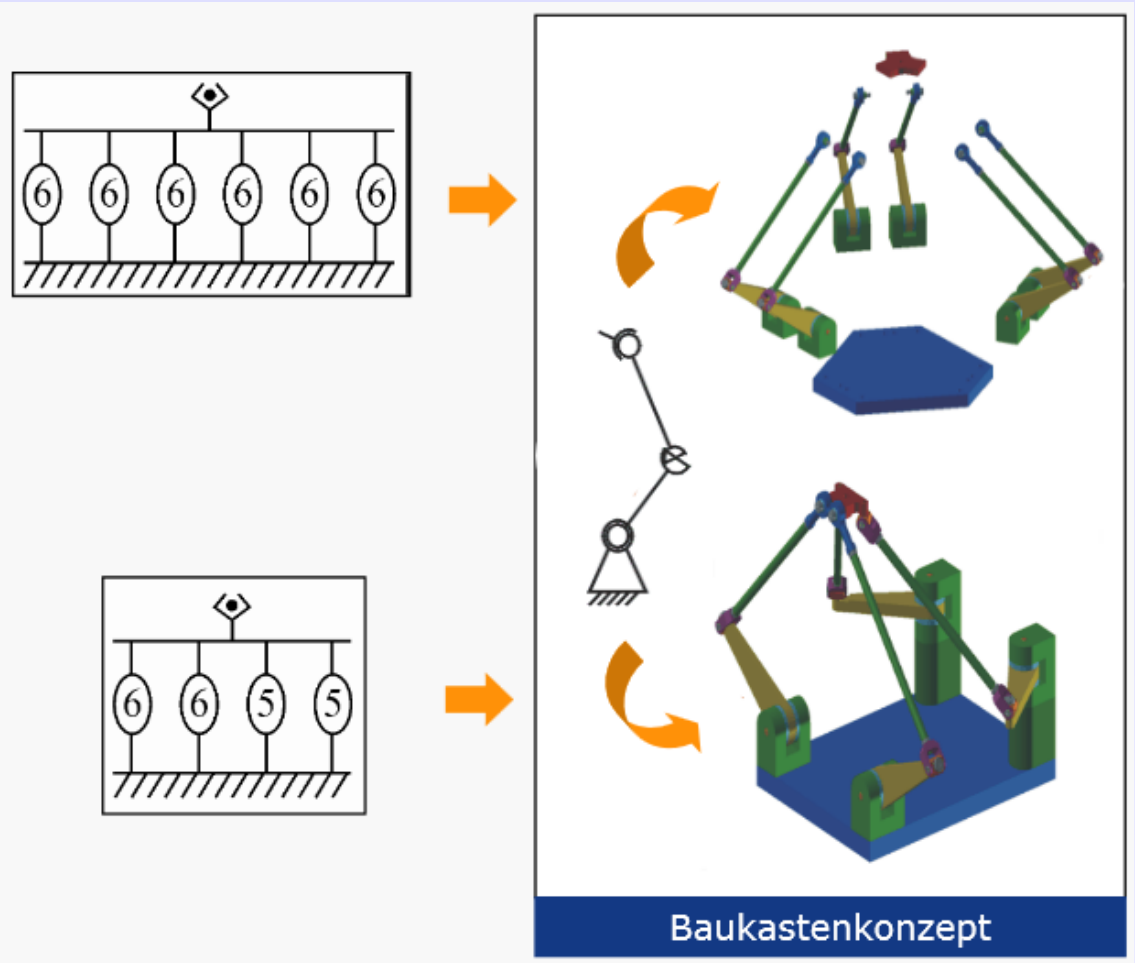
Auswahl der Gelenkketten (nach [?])

- Verwendung von Dreh- (D), Kardan- (D_2), Kugel- (D_3) und Schubgelenken (S) sowie Kreuztische (S_2)
- Ketten mit 3 Gelenkebenen als Kompromiss zwischen Konstruktion und Steifigkeit
- Keine passiven Schubgelenke oder angetriebene Gelenke nahe Gestell

Bezeichnung	Bewegungsgrad	Strukturskizze	Gelenke	Antriebe	Teilarbeitsraum
$D_3 D_2 D$	b = 6 räuml. Gelenkkette		1 Kugelgelenk 1 Kardangelenk 1 Drehgelenk	1 Antrieb: $D_3 D_2 \underline{D}$ (Drehantrieb, gestellfest)	
$D_3 D S_2$	b = 6 räuml. Gelenkkette		1 Kugelgelenk 1 Drehgelenk 1 Kreuztisch	2 Antriebe: $D_3 D \underline{S_2}$ (2x Schubantrieb)	
$D_3 D_2 S$	b = 6 räuml. Gelenkkette		1 Kugelgelenk 1 Kardangelenk 1 Schubgelenk	1 Antrieb: $D_3 D_2 \underline{S}$ (Schubantrieb, gestellfest)	
$D_3 S D_2$	b = 6 räuml. Gelenkkette		1 Kugelgelenk 1 Schubgelenk 1 Kardangelenk	1 Antrieb: $D_3 \underline{S} D_2$ (Schubantrieb, bewegt)	

Auswahl der Gelenkketten (nach [?])

- gegebene Beweglichkeit einer Gelenkkette (z.B. $F_i = 6$)$\Rightarrow$ verschiedene Kombinationsmöglichkeiten von Gelenken ($\Rightarrow$ D_3D_2D, DDDDDD oder ...)
- Für jede Gelenkkette ergibt sich Arm, mehrere Arme werden nach Baukastenkonzept positioniert
- Maßsynthese: Armlängen und Montagepositionen festlegen

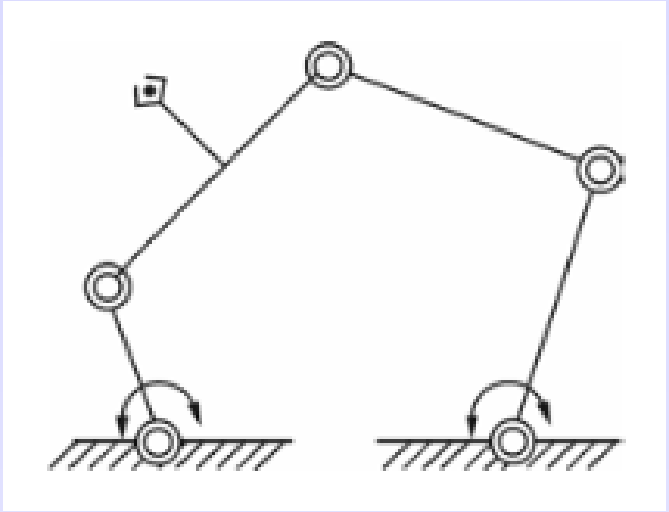


Hexa

- TU Braunschweig [?]
- Geschwindigkeit: 5 m/s
- Beschleunigung: 5g
- Nutzlast: 3 kg

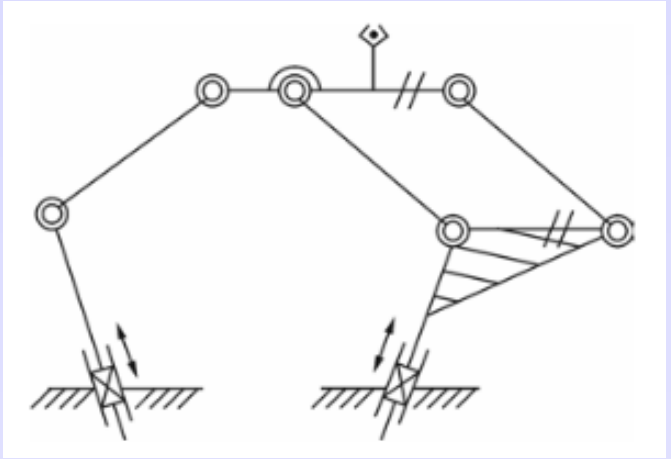
Fünfgelenk

- TU Braunschweig [?]
- 2D-Kinematik
- Aktive Schwingungsunterdrückung im passiven Unterarm durch Adaptronik
- Geschwindigkeit: 4 m/s
- Beschleunigung: 10g
- Nutzlast: 1 kg



Portys

- TU Braunschweig [?]
- 2D-Kinematik
- Parallelführung des Endeffektors
- Geschwindigkeit: 3 m/s
- Beschleunigung: 4g
- Genauigkeit: 10 µm
- Nutzlast: 3 kg



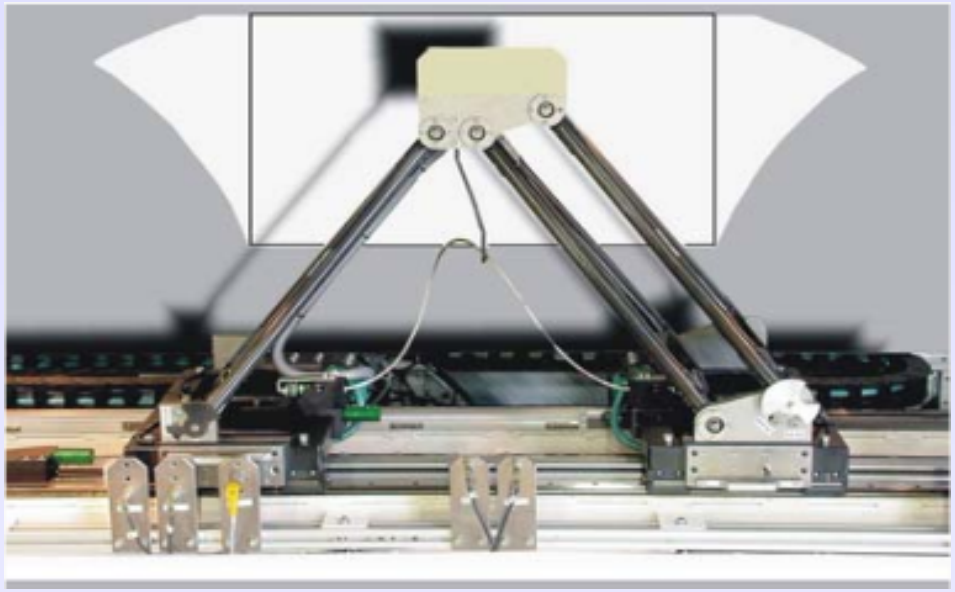
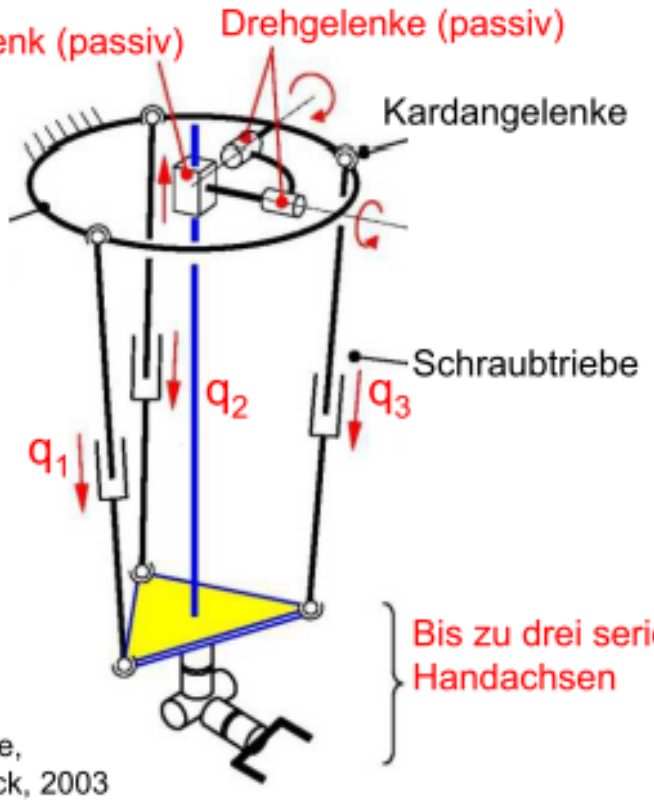


ABB IRB 940 (Tricept)

Typische Einsatzgebiete
- Mechanische Bearbeitung (Trennen, Bohren, Schleifen)
- Schweißen, Brennschneiden

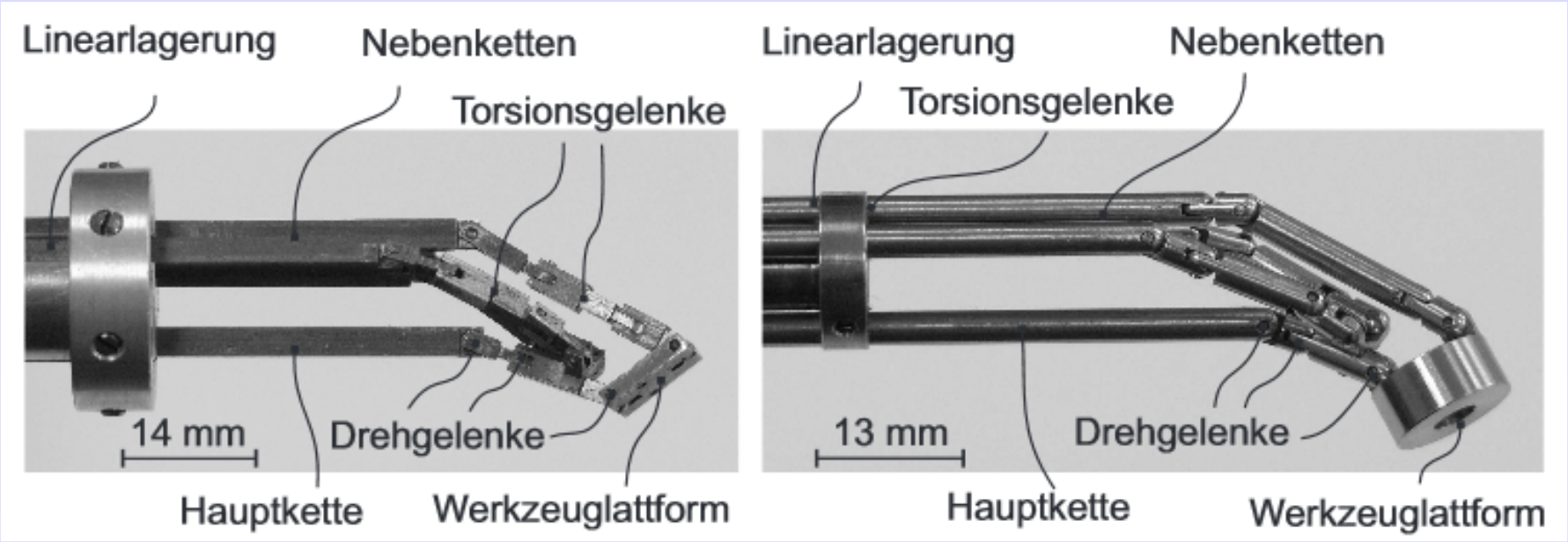
Eigenschaften
- Teilweise geschlossene kinematische Kette
- Günstig für Aufnahme von Bearbeitungskräften



Parallele Hybrid-struktur:

Nach C. Woernle,
iam - Uni Rostock, 2003

Parallelkinematik für Hand



- Anwendung: Medizintechnik
- Ziel: kompakter Aufbau der Nebenachsen, Lineare Antriebe (Motoren oder manuelle Bedienung) in Entfernung vom Werkzeug
- Gelenkfreiheitsgrade f=22, Gesamtfreiheitsgrad F=4

5. Mobile Roboter

Einführung

Anwendungen

- Transport in Produktion: Führerlose Transportsysteme (FTS), alternative zu spurgeführten Wagen
- Transport in Krankenhäusern: Medikamente, Essen, Wäsche
- Reinigung: Wege, Wohnungen (saugen), Solaranlagen
- Haushalt, Garten: Rasenmähen, Saugen
- Museen: Führungen
- Überwachung: Räume, Gebäude, Ausstellungen
- Autonome Autos, Bagger
- Spielzeug: Lego, Roboterhund AIBO von Sony

Mobile Roboter

- sind mit Rädern, Ketten, Beinen, Flügel o.ä. beweglich
- haben prinzipiell unbegrenzten Arbeitsraum mit wechselnden Umweltbedingungen
- müssen Antriebsenergie mitführen (Akku, Benzin) oder erzeugen (Sonne)
- müssen Arbeitsraum mit anderen (Menschen oder Robotern) teilen
- Arbeitsraum kann sich ständig ändern (Hindernisse, Individuen)
- müssen selbständig Umwelt erfassen, benötigen entsprechende Sensoren
- darf niemals bei Bewegung jemanden gefährden

Mobil: Gegenteil von stationär/ortsfest

Autonom: Mobil + selbständig

Asimov'sche Robotergesetze

Anm.: Isaac Asimov: russisch-amerikanischer Science-Fiction-Schriftsteller, 1942

1. Ein Roboter darf keinen Menschen verletzen oder zulassen, dass einem Menschen Schaden zugefügt wird (Sicherheit)
2. Ein Roboter muss den Befehlen eines Menschen gehorchen, außer, dies führt zu einer Verletzung des ersten Gesetzes (Verlässlichkeit)
3. Ein Roboter muss seine eigene Existenz schützen, außer dies führt zum Widerspruch mit den anderen Gesetzen (Verfügbarkeit))

5. Mobile Roboter

DOF - Degree of Freedom, Freiheitsgrade

(Aktiver) Freiheitsgrad

- Summe der translatorischen und rotatorischen Möglichkeiten der Gelenkbewegungen
- Entspricht Getriebefreiheitsgrad

Effektiver Freiheitsgrad

- Entspricht Dimensionalität der Pose (Position + Orientierung)
- Im Raum max. 6, in der Ebene max. 3

Holonome Systeme

- können von jeder Pose direkt jede andere Pose annehmen
- können in jede beliebige Richtung fahren, ohne vorherige Drehung
- Falls effekiver DOF größer als aktiver DOF $\Rightarrow$ nicht-holonom

Aktiver DOF eines Autos =

Effektiver DOF eines Autos =

Auto ist

Antriebsarten

- Räder
- Kettenfahrzeuge: `Chaos Robot Platform`
- Gehen (4 Beine): `Boston Dynamics`
- Gehen (2 Beine): `HRP-4C Miim's Human-like Walking`
- Hüpfen: `Bionic Kangaroo`
- Schlängeln, Kriechen: `Robot Snake`
- Fliegen: `Dronen`
- Klettern: `Climbing Robot (Boston Dynamics)`
- Springen: `Sand Flea`
- ...

→ hier: nur radgetriebene Roboter

Räder

Passive Räder

1 DOF Rad

2 DOF Castorrad

2 DOF Kugelrolle

Angetriebene 2 DOF Räder

Omni-Rad

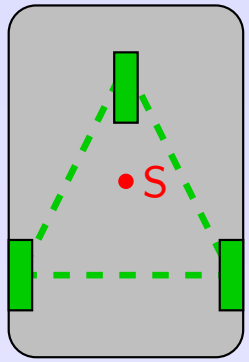
Omni-Rad

Mecanum-Rad

Stabilität radgetriebener Fahrzeuge

2 Räder

- *statisch* stabil falls Schwerpunkt unterhalb Radachse
- andernfalls Regelung zur *dynamischen* Stabilität (Segway)



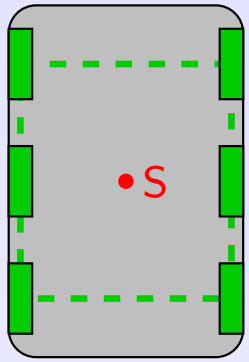
3 Räder

- stabil falls Schwerpunkt oberhalb durch Räder aufgespanntem Dreieck
- bei hohem Schwerpunkt können Fliehkräfte zum Kippen führen

$\rightarrow$ schwere Akkus meist ganz unten

$>$3 Räder

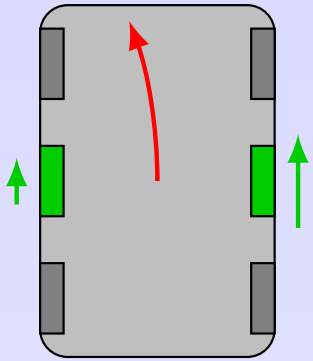
- impliziert größere Stabilität wegen größerer Standfläche (aufgespanntes Polygon)
- statisch überbestimmt (hyperstatisch),
- verlieren Kontakt auf unebenem Boden $\rightarrow$ Wackeln, Probleme bei unabhängig angetriebenen Rädern



Rad-Konfigurationen I

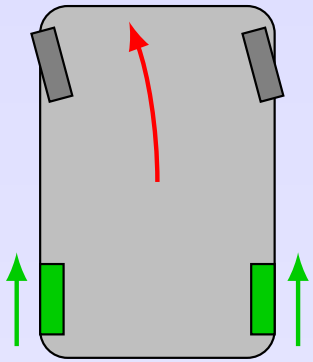
Differentialantrieb

- Alle Räder starr oder frei schwenkbare Stützräder
- Kurvenfahrt durch unterschiedliche links/rechts Geschwindigkeiten
- mind. 2 unabhängig angetriebene Räder $\rightarrow$ 2 Motoren
- nicht holonom



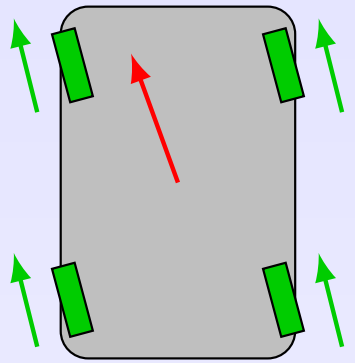
Einachsige Lenkung

- schwenkbare Achse oder schwenkbare Räder (Auto)
- gekoppelte Antriebsräder
- Antrieb + Lenkung $\rightarrow$ 2 Motoren
- nicht holonom



Synchro-Antrieb

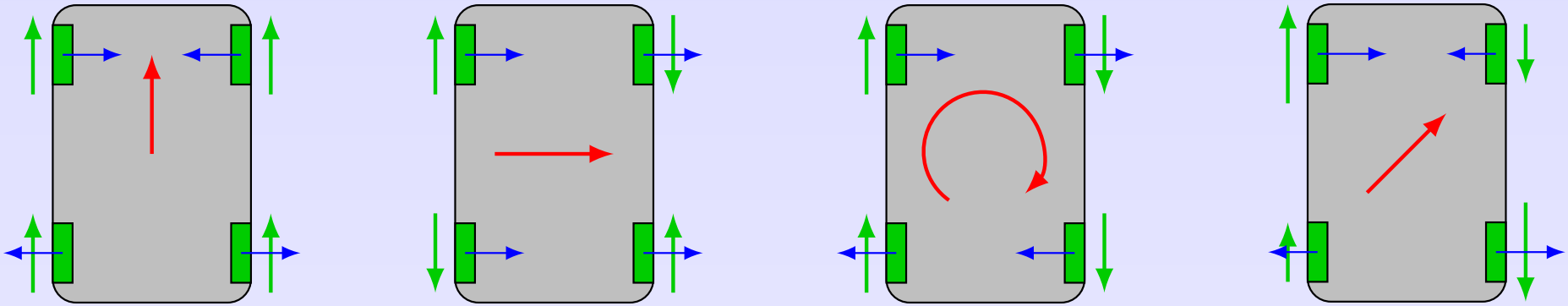
- mind. 3 synchron angetriebene und synchron lenkbare Räder
- Antrieb + Lenkung $\rightarrow$ 2 Motoren
- Effektiver DOF = aktiver DOF = 2 (Translationen)
- Holonom, Drehung der Grundfläche nicht möglich
- Mit drehbarer Plattform: DOF=3 und 3 Motoren



Rad-Konfigurationen II

Omni-Antrieb

- mind. 3 unabhängig angetriebene Omni- oder Mecanum-Räder
- $\rightarrow$ mind. 3 Motoren
- holonom, omnidiraktionale Bewegung
- Drehung von Rad und äußeren Rollen unabhängig



- Drehung von Rad und äußeren Rollen unabhängig $\rightarrow$ pro Rad zwei überlagerte Kräfte: Antriebskraft ↗ und Querkraft ↗
- `Robotnik Mecanum Wheels`
  `Phidgets MURVV` (`Anleitung`)

Anm.: Alle Konfigurationen nur beispielhaft: andere Anzahlen und Positionen von aktiven und passiven Rädern möglich

5. Mobile Roboter

Laserscanner

- präzise Entfernungsmessung nach Time-of-Flight-Prinzip
- Zur Umgebungserkundung und Kollisionsvermeidung
- sendet gepulsten, nichtsichtbaren Laserstrahl aus, der von Objekt reflektiert wird
- Aus Laufzeit folgt Entfernung

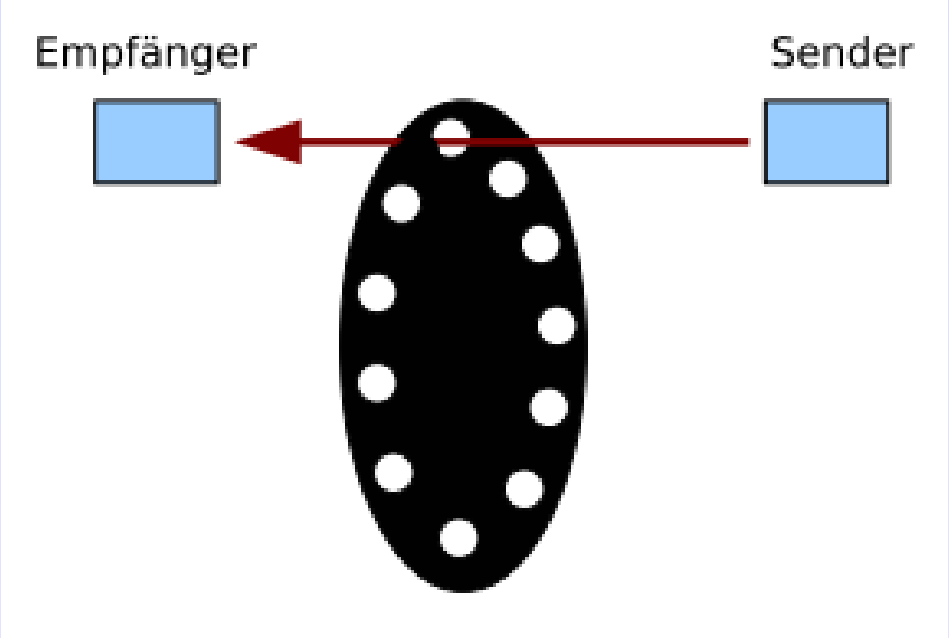
$$s = \frac{1}{2} c \cdot t$$

  Messbereich bis 80m,
- Bsp.: Winkelauflösung 0,5 $\Rightarrow$ 3cm breite Lücken im Abstand 10m

\+ präzise, hohe Taktzeiten

\− teuer

Radencoder



- Berechnung der Geschwindigkeit des Roboters aus Drehzahl der Räder
- Zur Geschwindigkeitsregelung und Selbstlokalisation
- Lochscheibe oder Reflexion
- Probleme: Schlupf, Unterschiedliche Drehzahlen der Räder bei Kurvenfahrt, lastabhängiger Raddurchmesser (Verformung)

\+ preiswert

− ungenau

5. Mobile Roboter

Grundprinzip

- Ziel $\rightarrow$ optimaler Pfad
- Hindernisse $\rightarrow$ Ausweichen, ggf. Pfadkorrektur
- Pfadsuche nutzt Wissen über Standort & Umgebung
- Hindernisdetektion und Pfadkorrektur mit hoher Frequenz: Reaktionen mit $\approx$ 10Hz
- Sicherheitsroutinen mit höchster Frequenz: Reflexe mit $\approx$ 100Hz

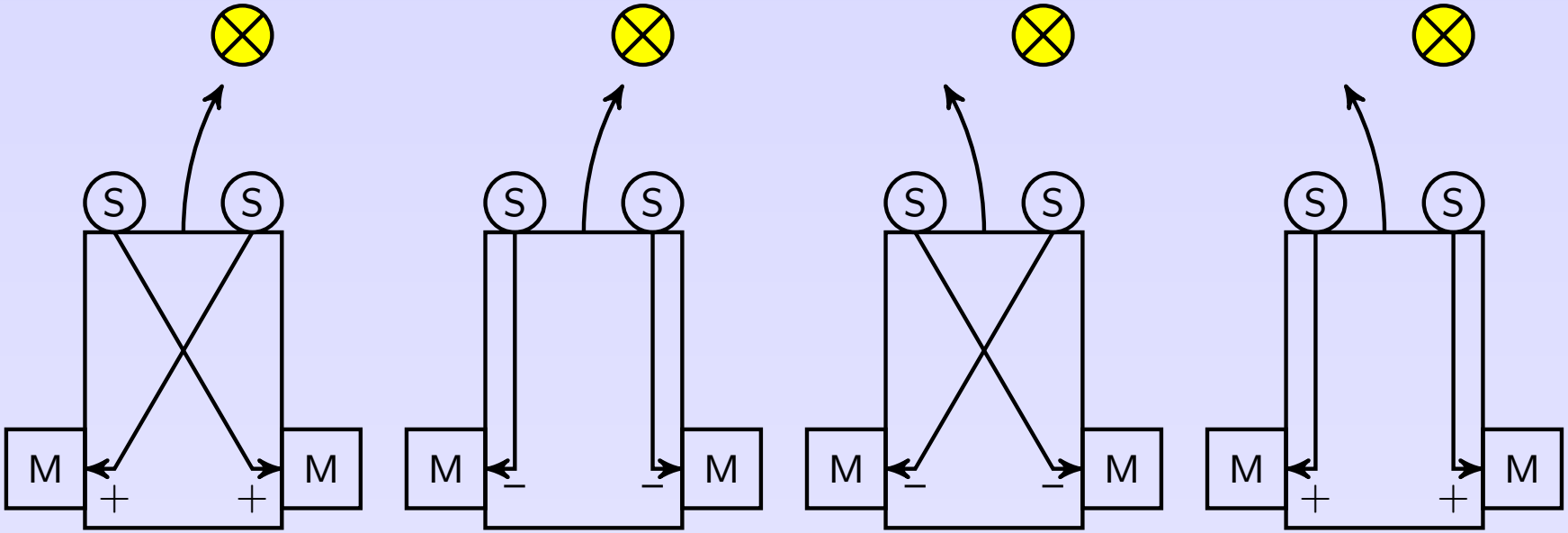
Reflexe

- wandeln Sensordaten am schnellsten in Steuersignale
- überschreiben andere Steuersignale
- teilweise in Hardware realisiert
- Beispiele: Not-Aus, Kontaktsensor

Reaktionen

- nutzen Sensordaten, zum Teil fusioniert oder vor verarbeitet
- Basis für sog. verhaltensbasierte Robotik

Navigation auf Basis von Reflexen



- Idee: Braitenberg-Vehikel (Valentino Braitenberg, 1986)
- Motoren direkt mit Sensoren gekoppelt
- Erhöhtes Sensorsignal erhöht (+) oder reduziert (−) Motordrehzahl
- Mit Helligkeitssensoren: Vehikel navigiert zum Licht hin oder davon weg
- Mit Abstandssensoren: Vehikel navigiert stets in Freiraum hinein
- Auch Kombination unterschiedlicher Sensoren auf einen Aktor möglich

Reaktive Navigation - Konturverfolgung

Naiver Algorithmus zur Navigation entlang einer Wand im Abstand d:

```
WENN (Entfernung zur Wand > d)
     DANN drehe in Richtung Wand;
WENN (Entfernung zur Wand < d)
     DANN drehe von der Wand weg;
```

- Problem: Oszillierendes Verhalten
- Lösung: Regelungstechnik

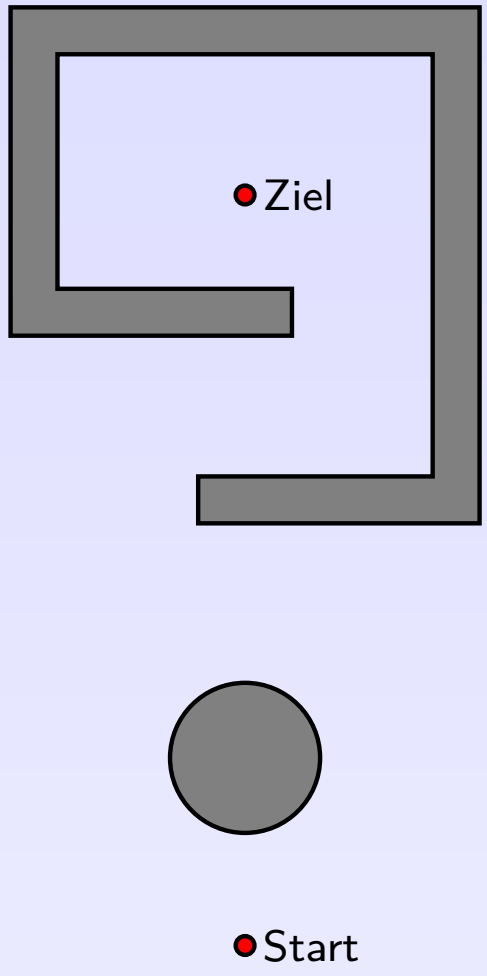
Reaktive Navigation - Bug-Algorithmen I

- Bug = Käfer, Insekt
- Nutzen nur lokales Wissen (Sensordaten) um globales Ziel zu erreichen
- Unterschiedliche Strategien optimieren Pfad, Speicherbedarf oder Rechenleistung

Algorithmus Bug-0

```
1 repeat
2   repeat
3     fahre durch Freiraum aufs Ziel zu
4     terminiere, falls Ziel erreicht
5   until Kontakt mit Hindernis
6   repeat
7     fahre entlang Hinderniskontur
8   until Richtung zum Ziel frei
9 until Ziel erreicht
```

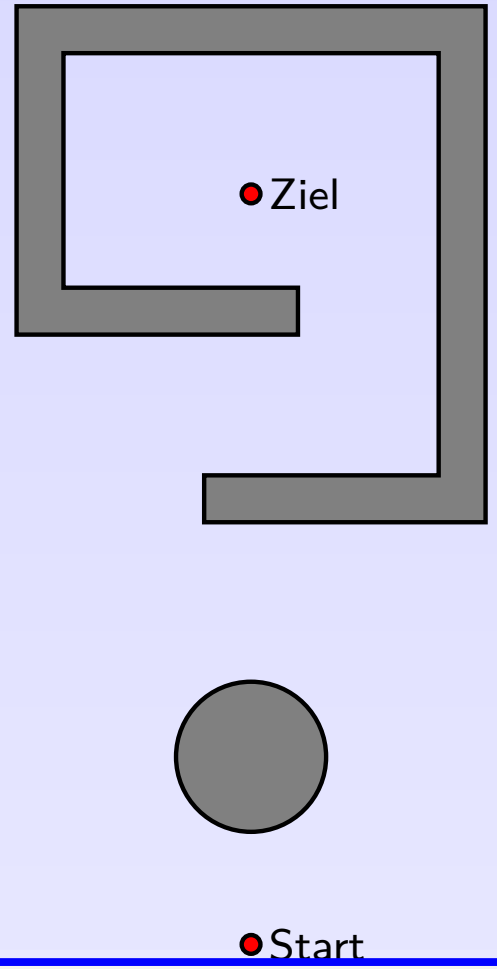
Eigenschaften:



Reaktive Navigation - Bug-Algorithmen II

Algorithmus Bug-1

```
1  repeat
2    repeat
3      fahre durch Freiraum aufs Ziel zu
4      terminiere, falls Ziel erreicht
5    until Kontakt mit Hindernis
6    var start := aktuelle Position
7    var near := aktuelle Position
8    repeat
9      fahre entlang Hinderniskontur
10     var akt := aktuelle Position
11     if akt naeher am Ziel als near then
12       near := aktuelle Position
13     endif
14   until akt = start
15   fahre entlang Hinderniskontur zu near
16 until Ziel erreicht
```

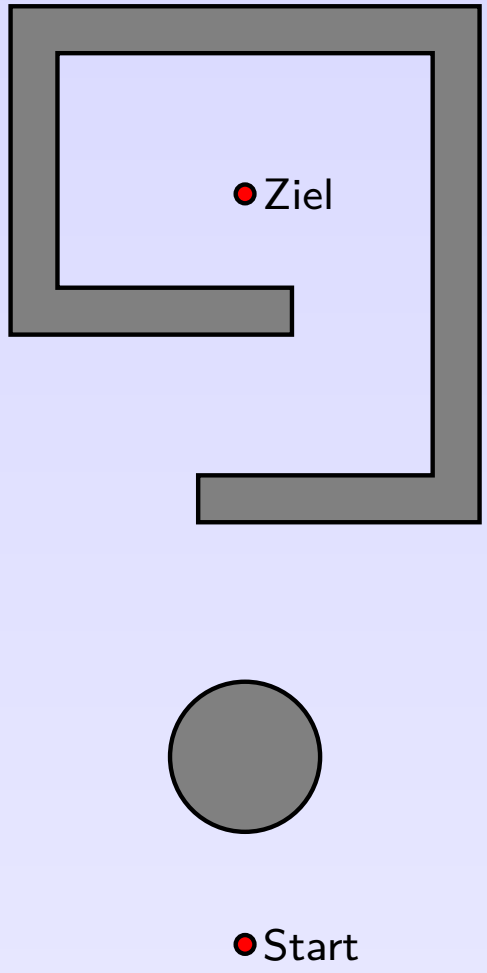


Eigenschaften:

Reaktive Navigation - Bug-Algorithmen III

Algorithmus Bug-3

```
1  repeat
2    repeat
3      fahre durch Freiraum aufs Ziel zu
4      terminiere, falls Ziel erreicht
5    until Kontakt mit Hindernis
6    repeat
7      fahre entlang Hinderniskontur
8    until Richtung zum Ziel frei and
9      schneidet nicht bisherigen Pfad
10 until Ziel erreicht
```



Eigenschaften:

Pfadsuche mit A*-Algorithmus I

- Suchalgorithmus zur Berechnung des kürzesten Pfades zwischen zwei Knoten mit Kantengewichtung/Kosten c
- findet stets optimale Lösung

Kosten c: z.B. Entfernung oder Zeit

Heuristik h: verwendete Schätzfunktion, darf Kosten nie überschätzen, z.B. Luftlinienabstand

Unknown Set $\mathcal{U}$: speichert unbekannte Knoten, deren Weg unbekannt ist

Open Set $\mathcal{O}$: speichert bekannte Knoten, deren möglicherweise suboptimaler Weg mit dazugehörigen Kosten bekannt ist

Closed Set $\mathcal{C}$: speichert fertige Knoten, deren optimaler Weg mit dazugehörigen Kosten bekannt ist

Initial: $\mathcal{U} = \{\text{Alle Knoten außer Start}\}$, $\mathcal{O} = \{\text{Start}\}$, $\mathcal{C} = \{\}$

Pfad: Jeder bekannte oder fertige Knoten enthält Zeiger auf Vorgänger, Rückverfolgung bis Start möglich

`A*-Demo`

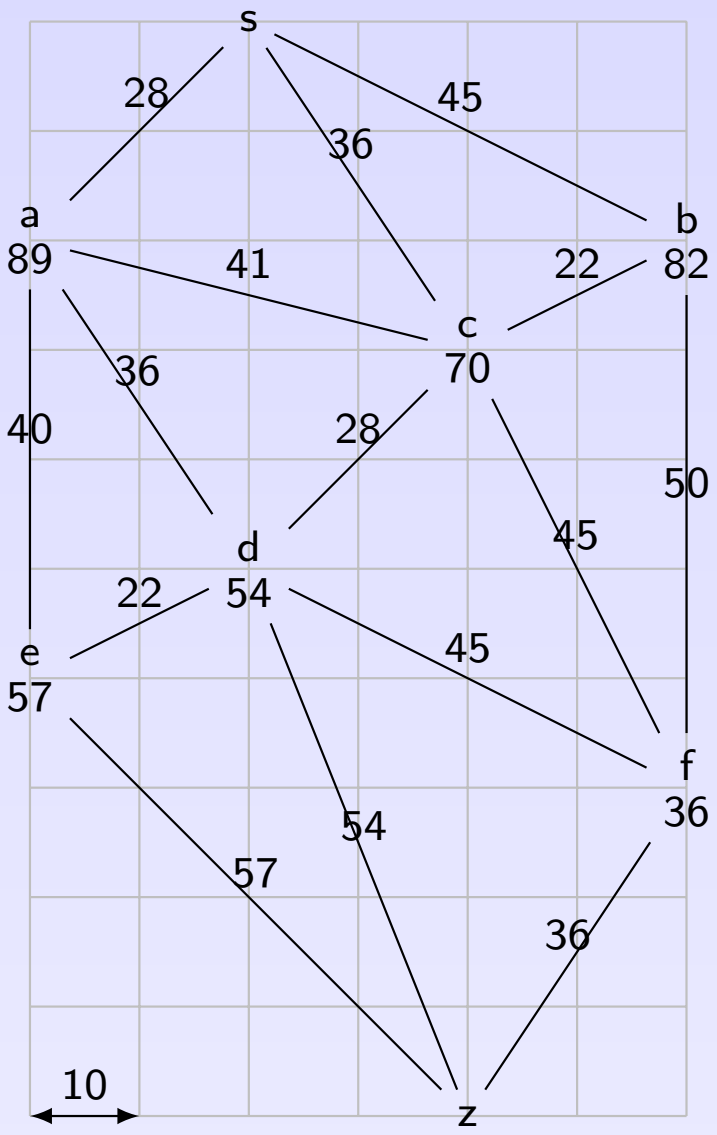
Pfadsuche mit A*-Algorithmus II

```
1  start.cost := 0                          // initialize cost of start
2  start.key := 0                           // initialize key of start
3  open.add(start)                          // add start to open set
4  repeat
5     node := minimum(open)                 // find minimum key of open set
6     if node=goal then                     // if goal is reached ...
7        return true                        // ... then ready
8     open.remove(node)                     // remove node from open set
9     closed.add(node)                      // add node to closed set
10    foreach next of successors(node) // expand node
11       if closed.contains(next) then
12          continue                       // nothing to do
13       c := node.cost + cost(node,next) // calculate costs
14       if open.contains(next) and c>=next.cost then
15          continue                       // nothing to do
16       next.prev := node                 // update prev-pointer
17       next.cost := c                    // update cost value
18       k := c + h(next,goal)             // estimate costs
19       if open.contains(next) then
20          next.key := k                  // update key
21       else
22          open.add(next)                 // add node to open set
23       endif
24    endfor
25 until open.isEmpty()                    // if open set is empty ...
26 return false                            // ... no path possible
```

Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

Heuristik: Luftlinienabstand



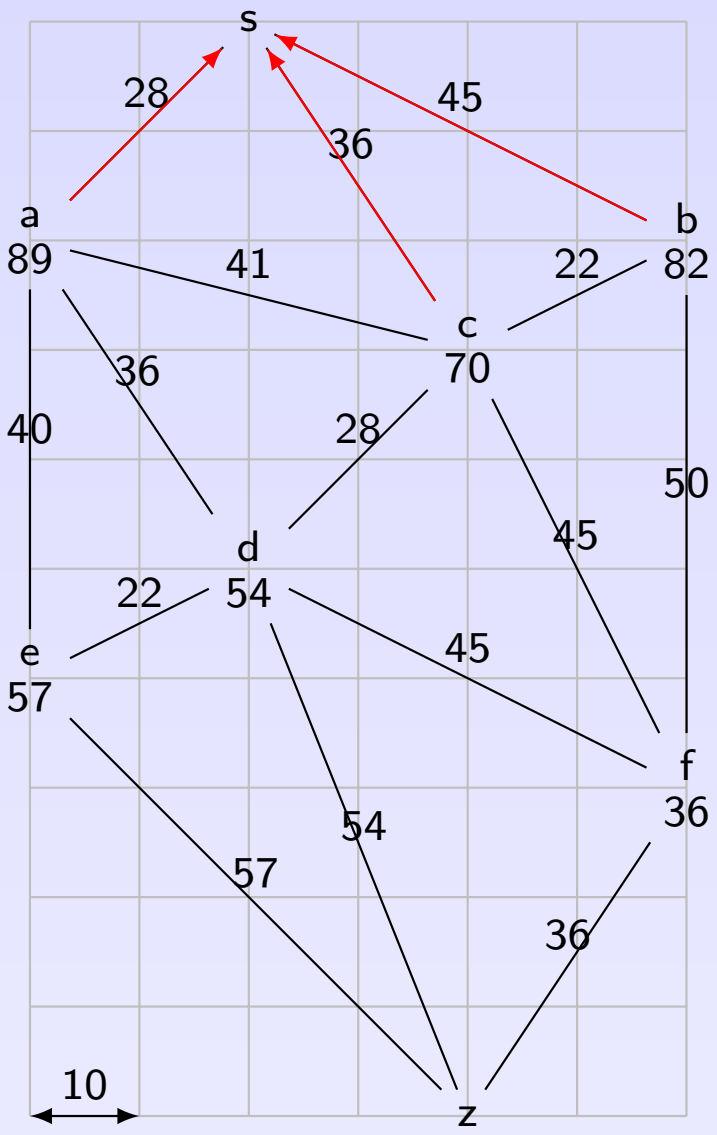
Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$
1. $\min(\mathcal{O}) = s$

	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

Heuristik: Luftlinienabstand



Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

1.

$\min(\mathcal{O}) = s$	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

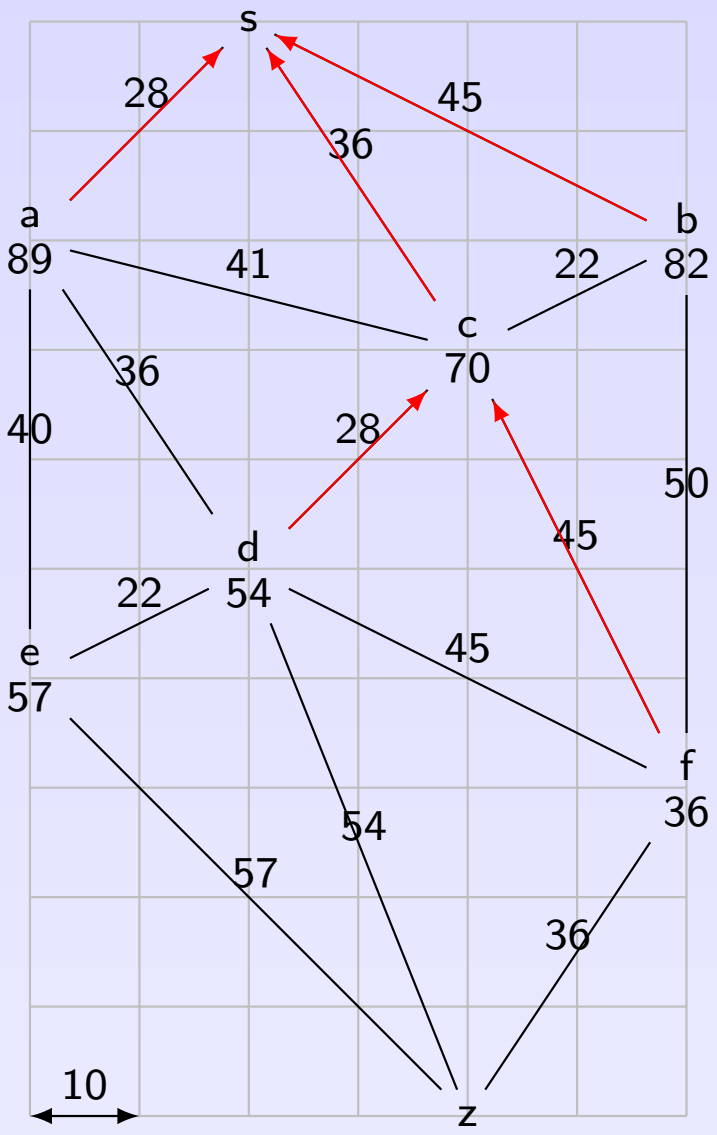
$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

2.

$\min(\mathcal{O}) = c$	cost	key	prev
a	min(36+41,28)=28	107	s
b	min(36+22,45)=45	45	s
d	36+28=64	64+54=118	c
f	36+45=81	81+36=117	c

$\mathcal{C} = \{s, c\}$ $\mathcal{O} = \{a, b, \not{c}, d, f\}$ $\mathcal{U} = \{e, z\}$

Heuristik: Luftlinienabstand



Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

1. $\min(\mathcal{O}) = s$

	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

2. $\min(\mathcal{O}) = c$

	cost	key	prev
a	min(36+41,28)=28	107	s
b	min(36+22,45)=45	45	s
d	36+28=64	64+54=118	c
f	36+45=81	81+36=117	c

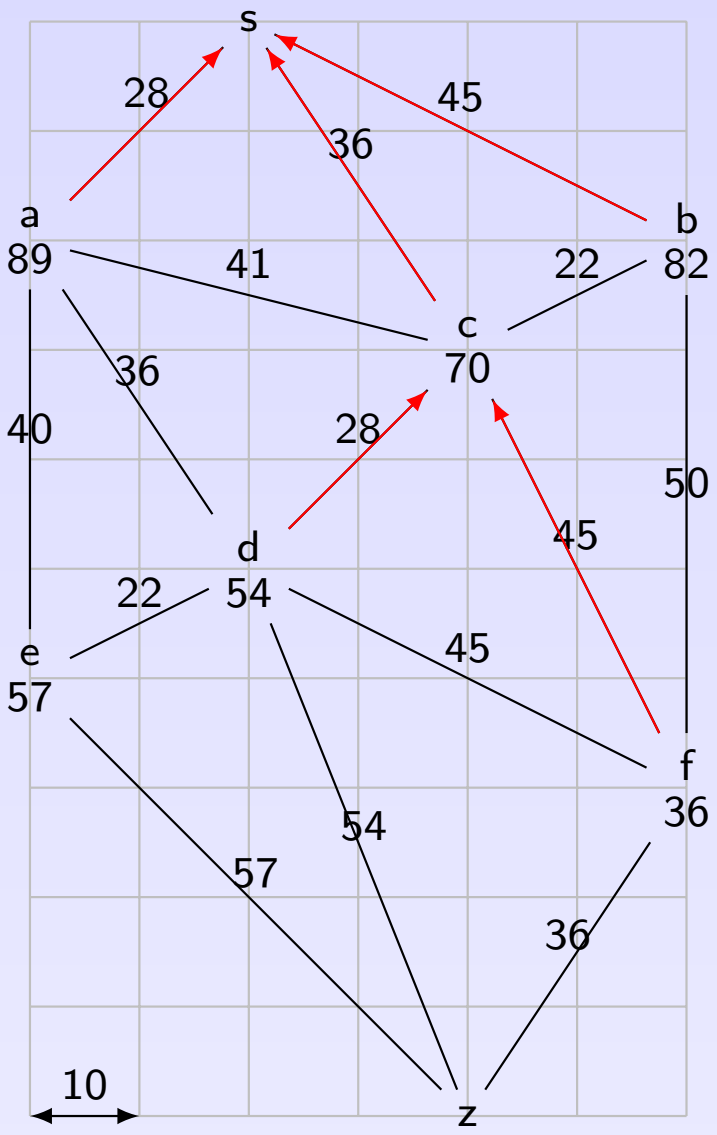
$\mathcal{C} = \{s, c\}$ $\mathcal{O} = \{a, b, \not{c}, d, f\}$ $\mathcal{U} = \{e, z\}$

3. $\min(\mathcal{O}) = b$

	cost	key	prev
f	min(45+50,81)=81	117	c

$\mathcal{C} = \{s, b, c\}$ $\mathcal{O} = \{a, \not{b}, d, f\}$ $\mathcal{U} = \{e, z\}$

Heuristik: Luftlinienabstand



Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

1. $\min(\mathcal{O}) = s$

	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

2. $\min(\mathcal{O}) = c$

	cost	key	prev
a	min(36+41,28)=28	107	s
b	min(36+22,45)=45	45	s
d	36+28=64	64+54=118	c
f	36+45=81	81+36=117	c

$\mathcal{C} = \{s, c\}$ $\mathcal{O} = \{a, b, \not{c}, d, f\}$ $\mathcal{U} = \{e, z\}$

3. $\min(\mathcal{O}) = b$

	cost	key	prev
f	min(45+50,81)=81	117	c

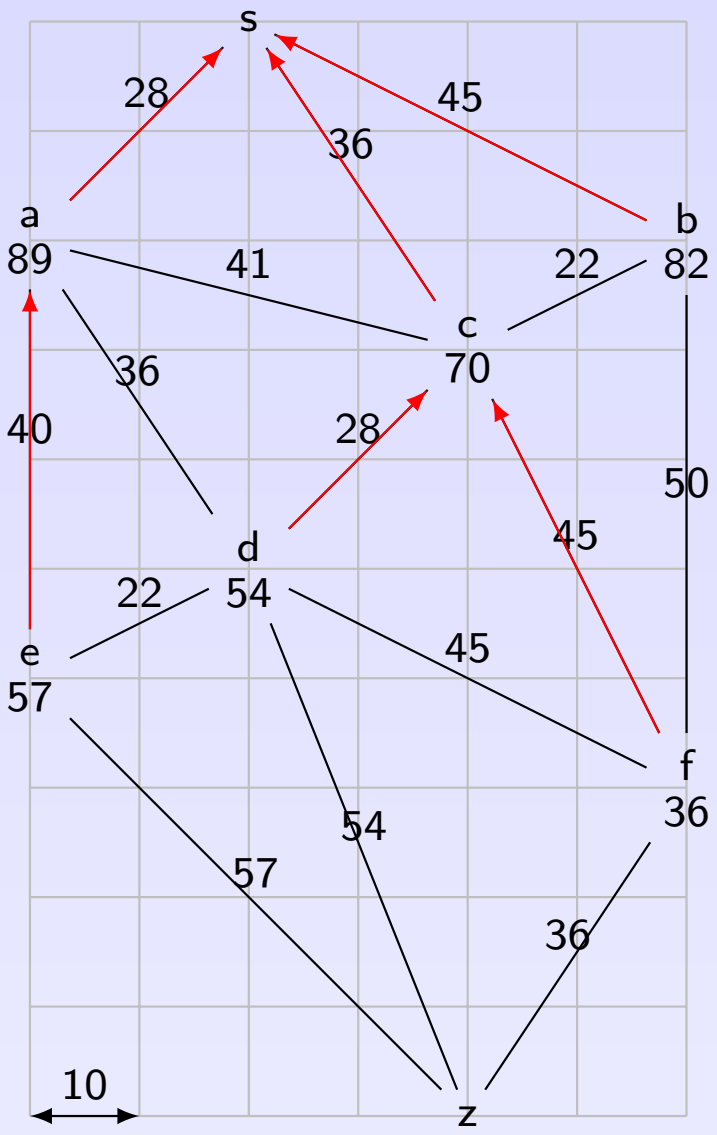
$\mathcal{C} = \{s, b, c\}$ $\mathcal{O} = \{a, \not{b}, d, f\}$ $\mathcal{U} = \{e, z\}$

4. $\min(\mathcal{O}) = a$

	cost	key	prev
d	min(28+36,64)=64	118	c
e	28+40=68	68+57=125	a

$\mathcal{C} = \{s, a, b, c\}$ $\mathcal{O} = \{\not{a}, d, e, f\}$ $\mathcal{U} = \{z\}$

Heuristik: Luftlinienabstand



Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

1. $\min(\mathcal{O}) = s$

	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

2. $\min(\mathcal{O}) = c$

	cost	key	prev
a	min(36+41,28)=28	107	s
b	min(36+22,45)=45	45	s
d	36+28=64	64+54=118	c
f	36+45=81	81+36=117	c

$\mathcal{C} = \{s, c\}$ $\mathcal{O} = \{a, b, \not{c}, d, f\}$ $\mathcal{U} = \{e, z\}$

3. $\min(\mathcal{O}) = b$

	cost	key	prev
f	min(45+50,81)=81	117	c

$\mathcal{C} = \{s, b, c\}$ $\mathcal{O} = \{a, \not{b}, d, f\}$ $\mathcal{U} = \{e, z\}$

4. $\min(\mathcal{O}) = a$

	cost	key	prev
d	min(28+36,64)=64	118	c
e	28+40=68	68+57=125	a

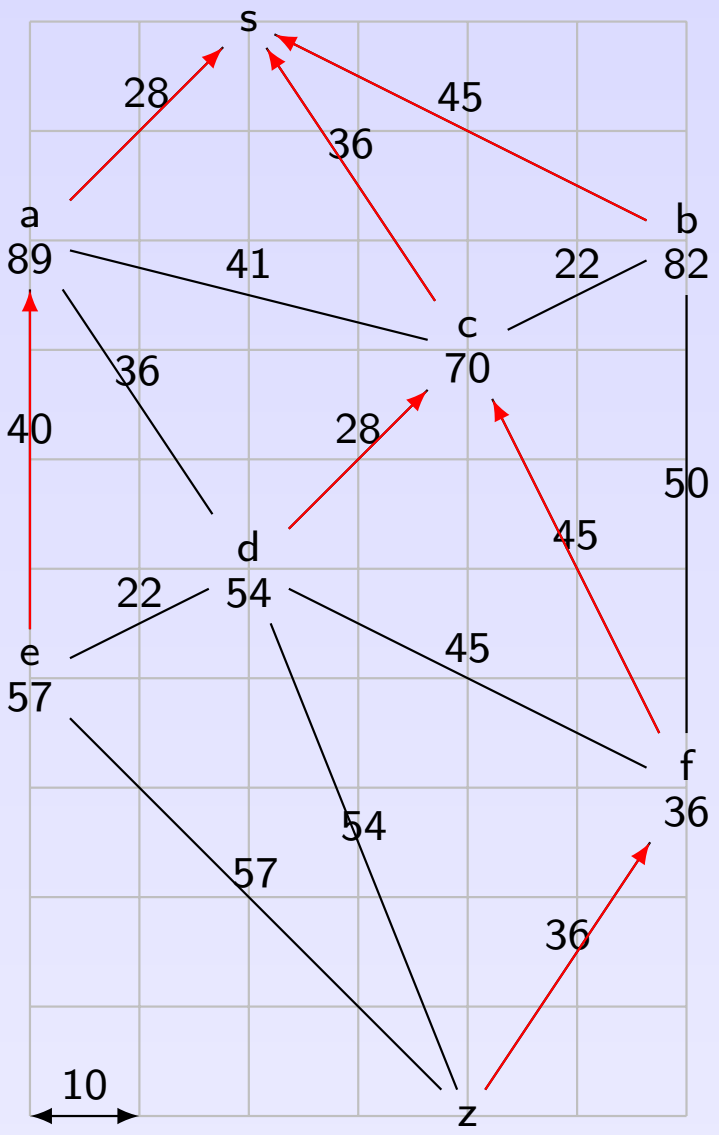
$\mathcal{C} = \{s, a, b, c\}$ $\mathcal{O} = \{\not{a}, d, e, f\}$ $\mathcal{U} = \{z\}$

5. $\min(\mathcal{O}) = f$

	cost	key	prev
d	min(81+45,64)=64	118	c
z	81+36=117	117	f

$\mathcal{C} = \{s, a, b, c, f\}$ $\mathcal{O} = \{d, e, \not{f}, z\}$ $\mathcal{U} = \{\}$

Heuristik: Luftlinienabstand



Beispiel zum A*-Algorithmus

0. Initial: $\mathcal{C} = \{\}$ $\mathcal{O} = \{s\}$ $\mathcal{U} = \{a, b, c, d, e, f, z\}$

1. $\min(\mathcal{O}) = s$

	cost	key	prev
a	0+28=28	28+89=107	s
b	0+45=45	45+82=127	s
c	0+36=36	36+70=106	s

$\mathcal{C} = \{s\}$ $\mathcal{O} = \{\not{s}, a, b, c\}$ $\mathcal{U} = \{d, e, f, z\}$

2. $\min(\mathcal{O}) = c$

	cost	key	prev
a	min(36+41,28)=28	107	s
b	min(36+22,45)=45	45	s
d	36+28=64	64+54=118	c
f	36+45=81	81+36=117	c

$\mathcal{C} = \{s, c\}$ $\mathcal{O} = \{a, b, \not{c}, d, f\}$ $\mathcal{U} = \{e, z\}$

3. $\min(\mathcal{O}) = b$

	cost	key	prev
f	min(45+50,81)=81	117	c

$\mathcal{C} = \{s, b, c\}$ $\mathcal{O} = \{a, \not{b}, d, f\}$ $\mathcal{U} = \{e, z\}$

4. $\min(\mathcal{O}) = a$

	cost	key	prev
d	min(28+36,64)=64	118	c
e	28+40=68	68+57=125	a

$\mathcal{C} = \{s, a, b, c\}$ $\mathcal{O} = \{\not{a}, d, e, f\}$ $\mathcal{U} = \{z\}$

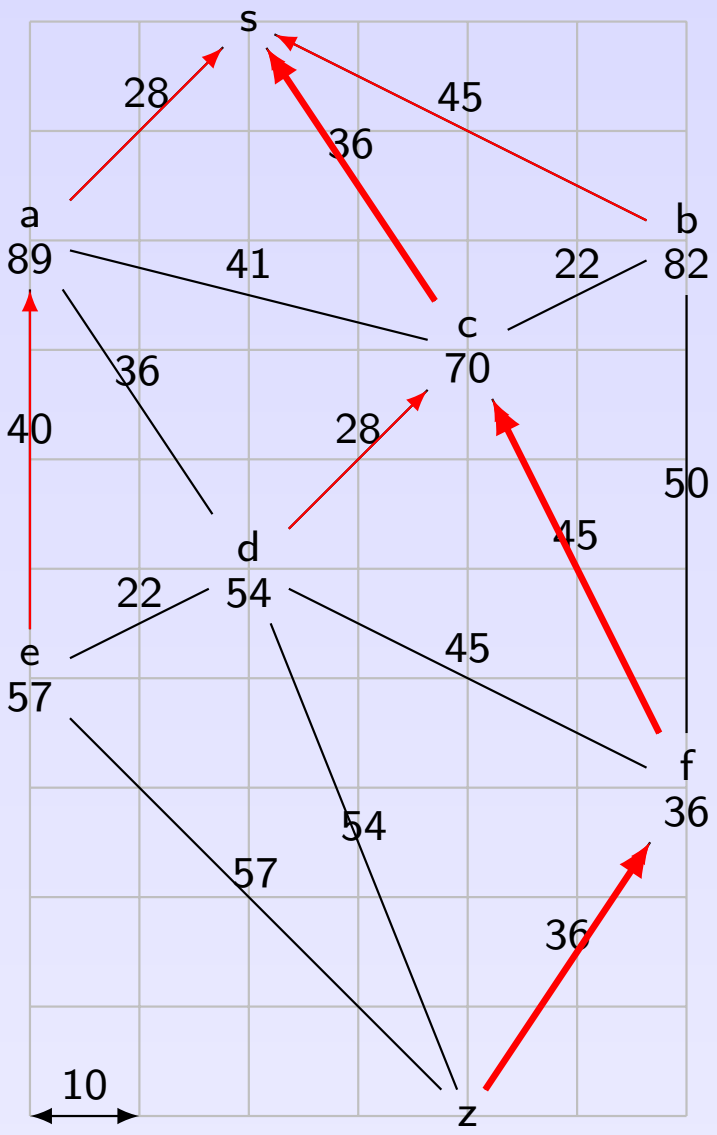
5. $\min(\mathcal{O}) = f$

	cost	key	prev
d	min(81+45,64)=64	118	c
z	81+36=117	117	f

$\mathcal{C} = \{s, a, b, c, f\}$ $\mathcal{O} = \{d, e, \not{f}, z\}$ $\mathcal{U} = \{\}$

6. $\min(\mathcal{O}) = z \rightarrow$ Ready

Heuristik: Luftlinienabstand



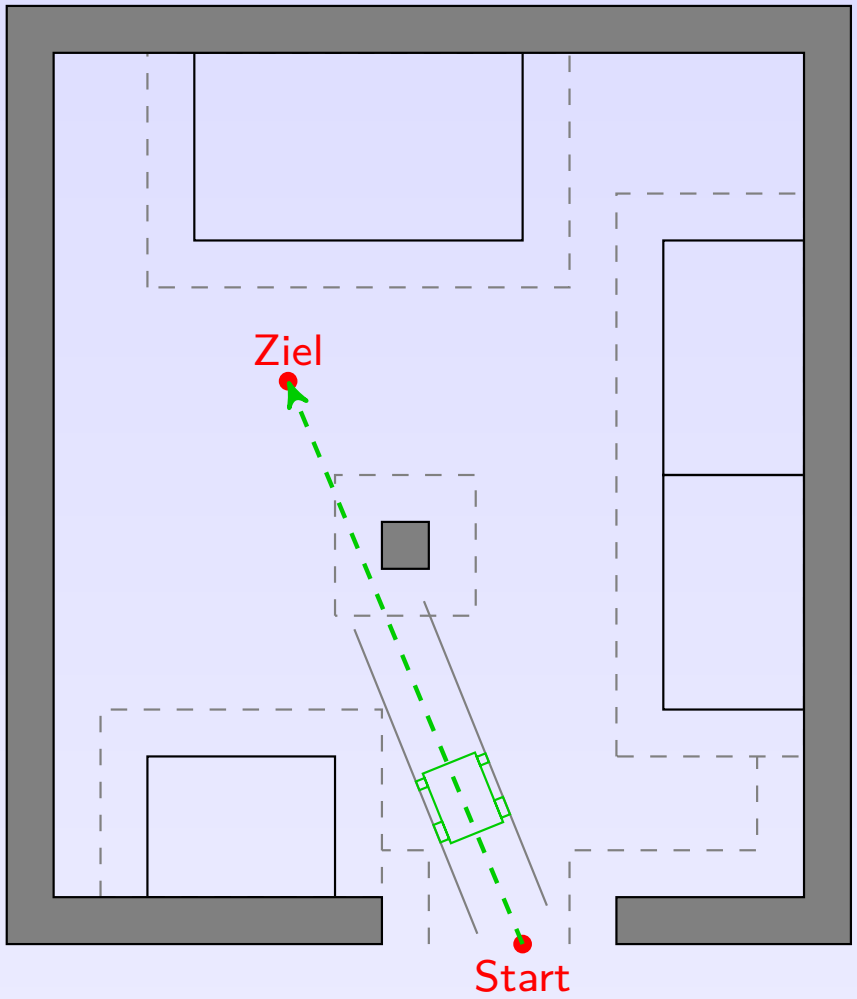
Konfigurationsraum und Arbeitsraum

Konfigurationsraum:

- Raum der Achskoordinaten
- für 6-Achs Knickarmroboter: 6D
- Alternative für stationäre Roboter

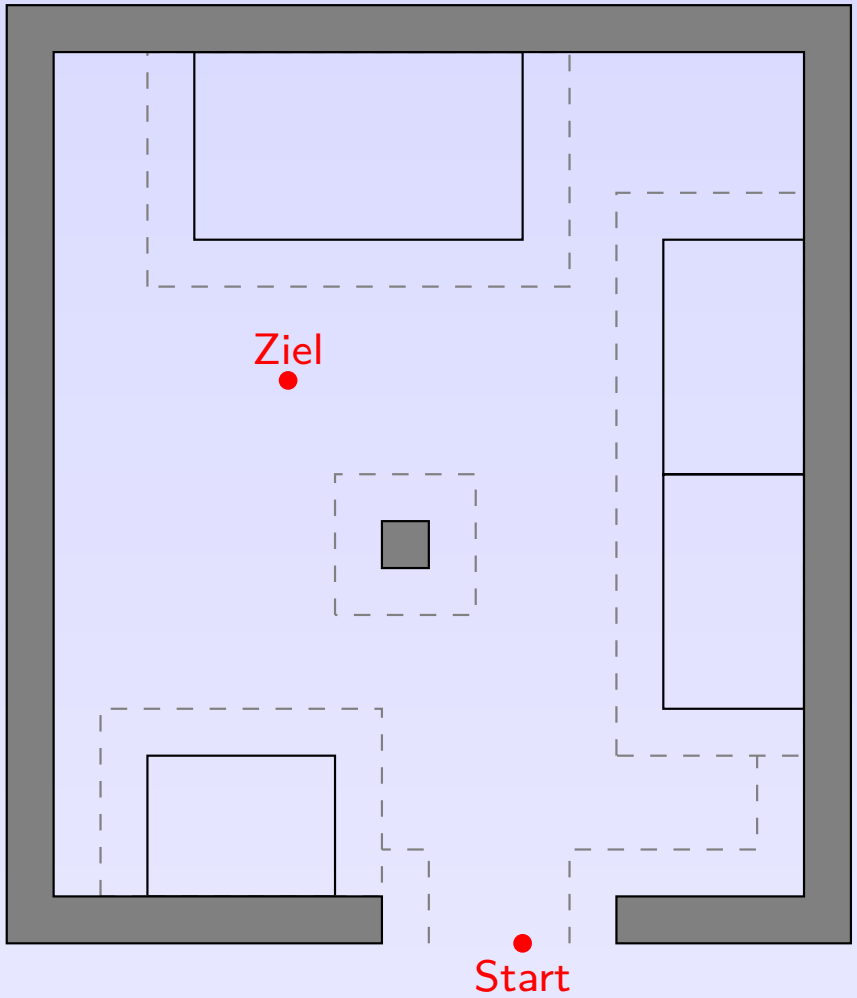
Arbeitsraum:

- Physischer Raum mit physischen Begrenzungen/Hindernissen
- Ebene: 2D-Arbeitsraum
- Üblich bei mobilen Robotern
- Modellierung von Ausdehnung des Roboters schwierig
- Einfacher: Hindernisse um halbe Roboterbreite vergrößern



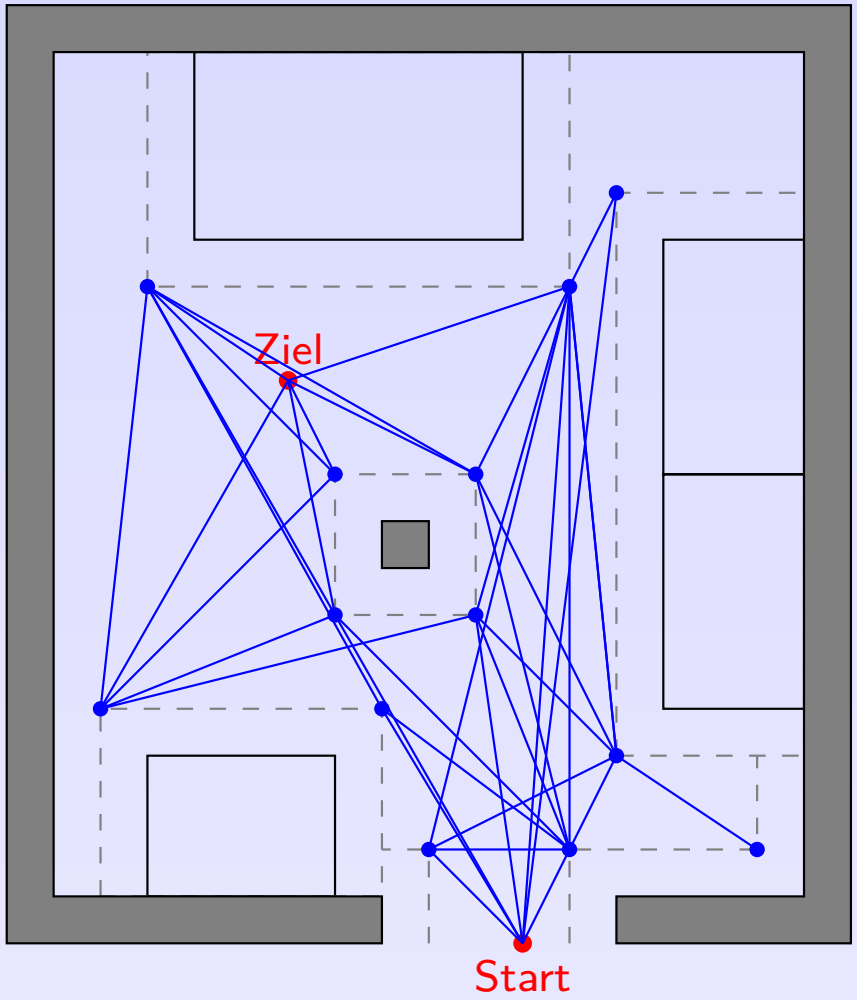
Sichtbarkeitsgraph

- Umgebungskarte = Menge von Polygonen



Sichtbarkeitsgraph

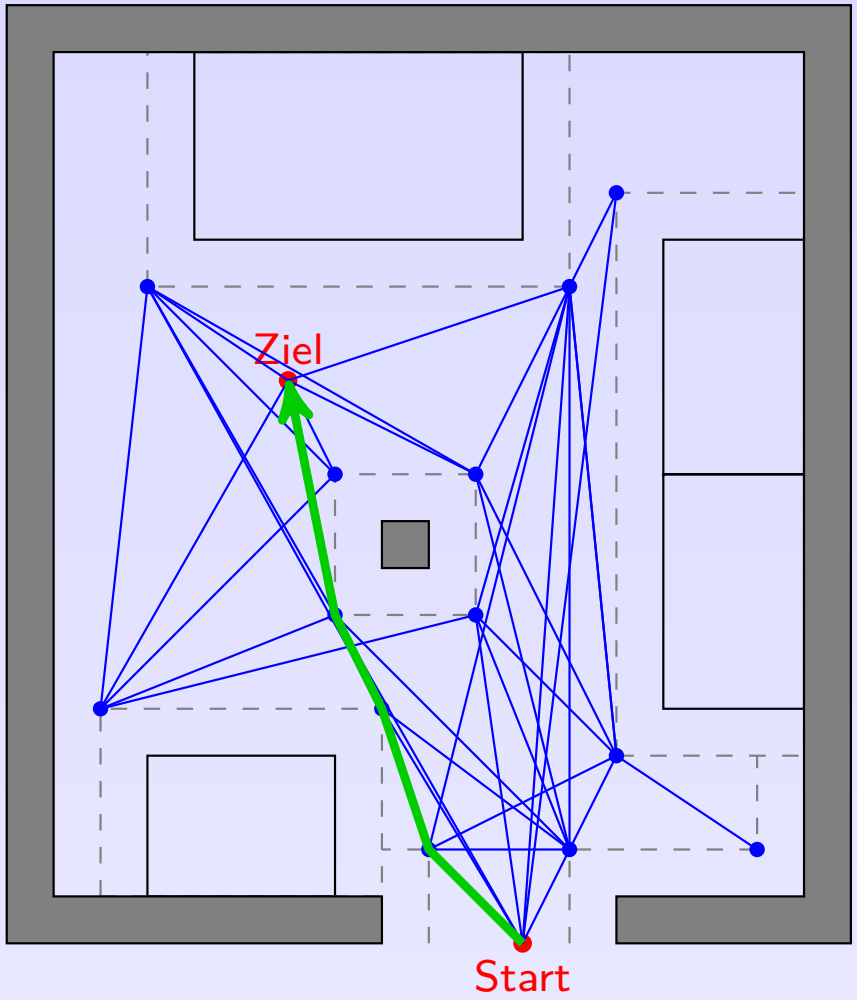
- Umgebungskarte = Menge von Polygonen
- Alle Polygonecken sind Knoten •
- Start und Ziel sind ebenfalls Knoten •
- Kanten ⁄ zwischen zwei Konten mit Sichtkontakt
- A*-Pfadsuche, Heuristik = euklidische Distanz von Knoten zum Ziel



Nicht alle Kanten gezeichnet,
weitere Kanten entlang Objekträndern!

Sichtbarkeitsgraph

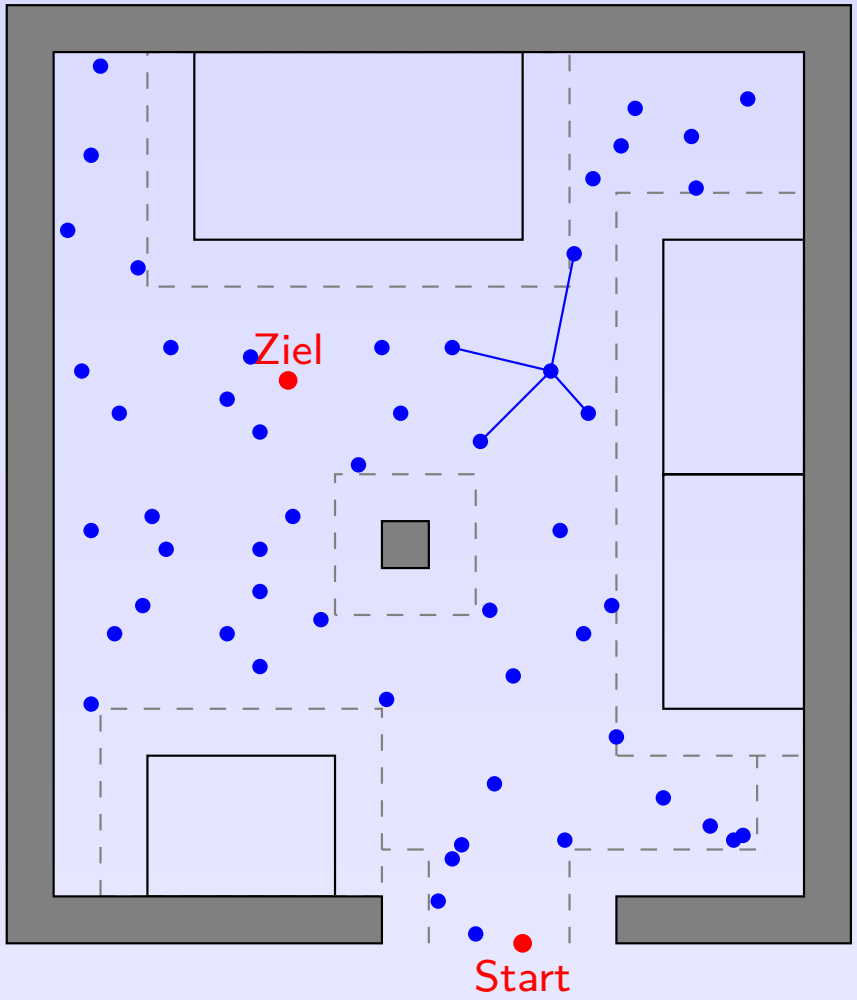
- Umgebungskarte = Menge von Polygonen
- Alle Polygonecken sind Knoten •
- Start und Ziel sind ebenfalls Knoten •
- Kanten ╱ zwischen zwei Konten mit Sichtkontakt
- A*-Pfadsuche, Heuristik = euklidische Distanz von Knoten zum Ziel
- Problem: Pfad ↗ oft dicht an Objektecken und Hindernissen



Nicht alle Kanten gezeichnet,
weitere Kanten entlang Objekträndern!

Probabilistische Straßenkarte

- Umgebungskarte = Menge von vergrößerten Polygonen
- Zufällig verteilte Knoten im freien Raum
- Kanten zu jeweils n nächsten Nachbarn oder zu allen Knoten im Abstand $<d$ (nur jeweils sichtbare Knoten!)
- A*-Pfadsuche, Heuristik = euklidische Distanz von Knoten zum Ziel
- Für große Anzahl Knoten wird kürzester Pfad gefunden
- Für kleine Anzahl Knoten: Effizient aber Zick-Zack-Pfad



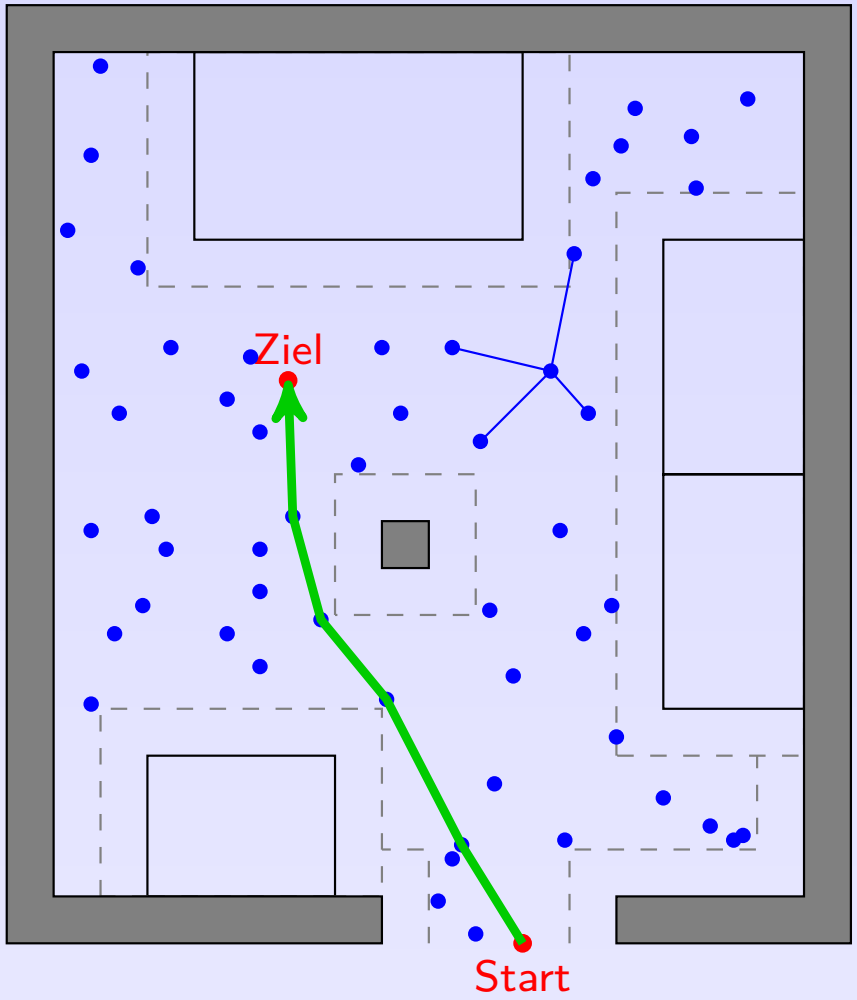
Exemplarisch nur Kanten
für einen Knoten gezeichnet!

Variationen

- Kombination von Polygonecken und zufälligen Punkten
- Knoten nicht zufällig, sondern z.B. auf kartesischem oder hexagonalem Gitter

Probabilistische Straßenkarte

- Umgebungskarte = Menge von vergrößerten Polygonen
- Zufällig verteilte Knoten im freien Raum
- Kanten zu jeweils n nächsten Nachbarn oder zu allen Knoten im Abstand $<d$ (nur jeweils sichtbare Knoten!)
- A*-Pfadsuche, Heuristik = euklidische Distanz von Knoten zum Ziel
- Für große Anzahl Knoten wird kürzester Pfad gefunden
- Für kleine Anzahl Knoten: Effizient aber Zick-Zack-Pfad



Exemplarisch nur Kanten
für einen Knoten gezeichnet!

Variationen

- Kombination von Polygonecken und zufälligen Punkten
- Knoten nicht zufällig, sondern z.B. auf kartesischem oder hexagonalem Gitter

SLAM - Simultaneous Localization and Mapping

- Simultane Lokalisierung und Kartenerstellung
- Falls Karte vorhanden: Roboter kann sich mittels Sensoren darin positionieren
- Falls absolute Position (GPS o.ä.) bekannt: Roboter kann relative Position von Hindernissen messen, damit absolute Position berechnen und somit Karte erstellen
- SLAM = Methode für mobile Roboter zur gleichzeitigen Erstellung einer Umgebungskarte und Schätzung eigener Pose innerhalb dieser Karte
- SLAM = autonome Erkundung der Umgebung um Karte zu erstellen und später zur Navigation zu nutzen
- Roboter sieht i.d.R. nur Teil der Umgebung, daher inkrementeller Aufbau der Karte
- Messwerte von Hindernissen und eigener Position mit Fehlern/Ungenauigkeiten behaftet $\rightarrow$ komplizierte Mapping- und Fehlerausgleichsalgorithmen
- Unterschiedliche Implementierungen/Ansätze (Extended Kalman Filter, Partikelfilter, Expectation Maximization Filter, Graph-basierte Techniken)
- de.wikipedia.org
- Visual SLAM Car Navigation

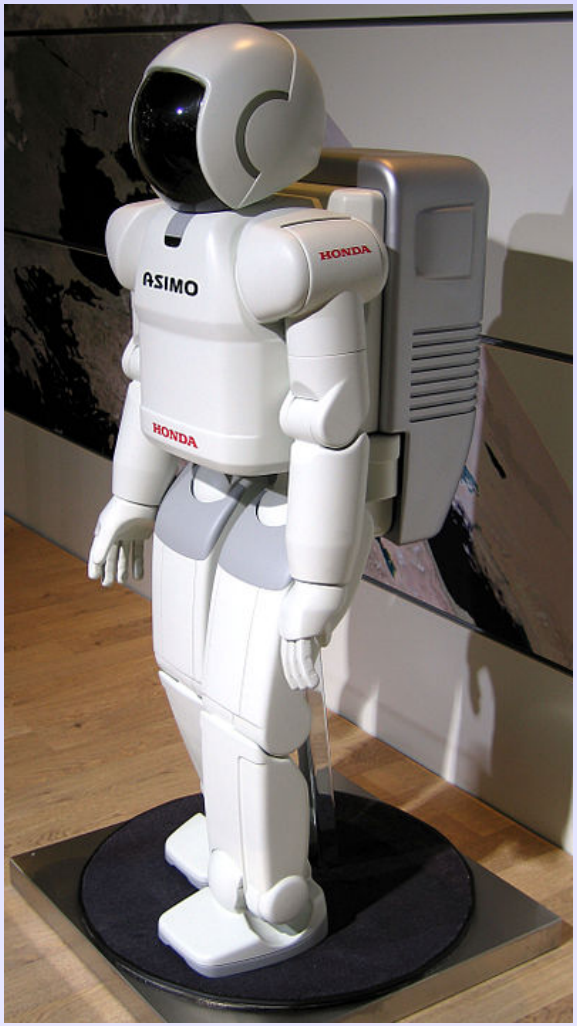
6. Humanoide Roboter

RoboCup

- Roboterfußball-Weltmeisterschaft, seit 1997 (Japan)
- 2015 (China): über 2000 Wissenschaftler und Studenten
- Liegen: Simulation, Fussball (Small, Middle, Humanoid), Sonstige (Rescue, @Home, Logistik)
- Ziel (u.a.): 2050 Turnier gegen Weltmeister
- robocup.org
- Weitere Wettbewerbe: SpaceBot, Euroc (European Robotics Challenge), German RoboCup, DARPA Robotics Challenge...

Asimo

- Asimo = Advanced Step in Innovative Mobility
- von Honda, Entwicklung seit 1999, mehrere Versionen
- am weitesten entwickelter Humanoide
- gehen, rennen (9 km $\perp$ h), Treppen steigen, Gleichgewicht durch Gyroskop und Beschleunigungssensoren
- kann 500g pro Hand heben, 55 kg, 130 cm
- Betriebsdauer 1h,
- 57 Freiheitsgrade (Kopf: 3, Arm: 7×2, Hand: 13×2, Oberkörper: 2, Bein: 6×2)
- Top-Down-Software: keine Lernfähigkeit, Tätigkeiten a-priori programmiert



Nao

- Spielzeugroboter von Aldebaran Robotics (Frankreich), 2006
- 57 cm, 25 Freiheitsgrade
- 2 Kameras, 4 Mikrofone, IR-Sender und -Empfänger, Berührungs- und Entfernungsmesser
- frei programmierbar, Intel ATOM 1.6GHz CPU (Linux)
- RoboCup Standard Plattform League
- Betriebsdauer 1,5h
- de.wikipedia.org

HRP-4C

= Miim

- von Forschergruppe AIST (Japan)
- Sprechen, Spracherkennung, Singen, Gehen, Tanzen, Emotionen
- 38 Motoren, 43 kg, 158 cm
- en.wikipedia.org

Stand der Technik

Sony QRIIO

REEM-C

Albert Hubo

TOPIO 3.0

6. Humanoide Roboter

Kommunikation

- Kommunikation mit menschlich aussehendem Roboter wird menschlich
  - Probleme: Dialekt, Rhetorik, Ironie
  - Enthält nonverbale Kommunikation: Mimik, Gestik, Tonfall
- Roboter hat (einige) überlegene Fähigkeiten: Geschwindigkeit, Sensorik (Laser, Radar), WLAN
- Mensch erwartet (alle) menschliche Fähigkeiten: den Blicken folgen, Zusammenhänge erkennen, Essen abschmecken
- Fazit: Kommunikation mit Humanoiden ist komplex und noch nicht gelöst
  Nicht-Humanoide (Service-Roboter) sollten wie Maschine aussehen, um Erwartungshaltung und Kommunikationsniveau zu reduzieren

Regionale Unterschiede:

- Deutschland: Skepsis gegenüber Humanoiden
- Japan: Hohe Akzeptanz gegenüber Humanoiden