

Mathematische Grundlagen

Kinematik: analysiert Geometrie eines Manipulators / Roboters.
Essentielles Konzept: Position

Statik: Kräfte / Momente auf ruhendem Mechanismus
Essentielles Konzept: Steifigkeit

Dynamik: analysiert Kräfte / Momente, die durch Bewegung / Beschleunigung eines Mechanismus und einer zusätzlichen Last entstehen

Kinematische Kette: Satz an Gliedern, durch Gelenke verbunden
Gelenk, z.B. Ellbogen

Punkt 2DOF
Armelement / Glied, z.B. Oberarm
Endeffektor, z.B. Hand, Greifer

Körper 3D 6DOF

Freiheitsgrade: Anzahl unabh. Parameter, die zur kompletten Spezifikation der Position eines Mechanismus / Objekts benötigt werden

SO(3) - Spezielle Orthogonale Gruppe: repr. Rotationen
 3×3 Matrizen $\in \mathbb{R}^{3 \times 3}$, $R^T R = I$, $\det(R) = 1$

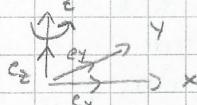
\rightarrow Isomorphismus nicht abelsch!

SE(3) - Spezielle Euklidische Gruppe: repr. Starrkörper beweg. \rightarrow Orientierung bleibt erhalten
Elemente der Form (p, R) , $p \in \mathbb{R}^3$, $R \in SO(3)$

Euklidischer Raum: \mathbb{R}^3 mit Standardskalarprodukt

\sim rechtsdrehendes Koordinatensystem:

$$\begin{aligned} e_x \times e_y &= e_z \\ x \times y &= z \end{aligned}$$



Endomorphismus: lin. Abb. $\phi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, bilden euki. Raum auf sich selbst ab

$$\sim \phi(a) = A \cdot a \quad A \in \mathbb{R}^{3 \times 3}$$

Basiswechselmatrix von e_x, e_y, e_z nach e'_x, e'_y, e'_z

$$A = (e'_x \ e'_y \ e'_z) \cdot (e_x \ e_y \ e_z)^{-1}$$

Isomorphismus: bijektiver Endomorphismus

Rotationen: 2D: lineare Transformation $R_\alpha(x) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \cdot x$

bei Rotation um $C \neq (0,0)$: verschiebe Ebene um $-C$
 \hookrightarrow Rotation $\rightarrow +C$

$$R_{C, \alpha}(x) = R_\alpha(x - C) + C = R_\alpha(x) + (-R_\alpha(C) + C)$$

3D: Rot. in 2D $\hat{=} \text{Rot. um } z\text{-Achse}$

$$R_{z, \alpha} = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{x, \alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}$$

$$R_{y, \alpha} = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix}$$

Inverse Rotationsmatrix: $R_{x, \theta}^{-1} = R_{x, -\theta}$

Eulerwinkel: jede Rotation \sim 3 Rot. um 3 koord.achsen

α, β, γ Eulerwinkel \rightarrow Rotationsmatrix

$$R_{z, \alpha} R_{x, \beta} R_{z, \gamma} = \begin{pmatrix} \cos \alpha \cos \beta - \sin \beta \cos \alpha & -\sin \alpha \cos \alpha - \cos \beta \cos \alpha \sin \alpha & \sin \beta \sin \alpha \\ \sin \alpha \cos \beta + \sin \beta \cos \alpha & -\sin \beta \sin \alpha + \cos \beta \cos \alpha \cos \alpha & -\sin \beta \cos \alpha \\ \sin \beta \sin \beta & \cos \beta \sin \beta & \cos \beta \end{pmatrix}$$

\sim (1) Drehung um α um z -Achse des BKS

(2) Drehung um β um neue x -Achse x'

(3) Drehung um γ um neue yz -Achse z''

Roll - Pitch - Yaw

- (1) x -Achse des BKS um α
- (2) y -Achse des BKS um β
- (3) z -Achse des BKS um γ

$$\left. \begin{array}{l} R_{x, \alpha} \\ R_{y, \beta} \\ R_{z, \gamma} \end{array} \right\} R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha)$$

Affine Transformation affiner Raum = erweiterter eukl. Raum

$$a = (a_x, a_y, a_z, h) \quad h \in \{0, 1\}$$

\sim kombiniere Translation und lineare Transformation

$$\mathbf{b} = A\mathbf{x} + \mathbf{t}$$

$$\Leftrightarrow \mathbf{b} = \begin{pmatrix} b \\ 1 \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} + \begin{pmatrix} t \\ 0 \end{pmatrix} = \begin{pmatrix} A & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix}$$

fehl = Vektor

$$\rightarrow \text{homogene } 4 \times 4 \text{ Matrizen } T = \begin{pmatrix} A & t \\ 0^T & 1 \end{pmatrix}$$

Translationsmatrix: Verschiebung des Oks nach (t_x, t_y, t_z) im BKS

$$T_{\text{trans}} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \leftarrow \text{Verschiebung}$$

$$\text{Translationsdistanz } \| (T_1) - (T_2) \| = \Gamma \dots$$

Basisrotationsmatrizen:

$$T_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{y,\beta} = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{z,\gamma} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \leftarrow \text{Rotation}$$

Abb. des Ortsvektors pos im OKS ins BKS:

$$p_{\text{BKS}} = T p_{\text{OKS}}$$

$$\text{mit } T = \begin{pmatrix} n & 0 & a & u \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

\leftarrow Ursprung des OKS

mit

$$T^{-1} = \begin{pmatrix} n^T u & -n^T u & -o^T u \\ R_{3 \times 3}^T & 0 & -a^T u \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Einheitsvekt. des OKS bzgl. BKS

Rotationsdiff. $\Delta\alpha$

$$R_{\text{ist,ziel}} = R_{\text{ist}}^{-1} R_{\text{ziel}} = R_{\text{ist}}^T R_{\text{ziel}}$$

$$\text{Spur}(R) = 1 + 2 \cos \alpha$$

$$\text{Spur}(R_{\text{ist,ziel}}) = 1 + 2 \cos(\Delta\alpha)$$

Quaternionen $|H|$

$$H = c + ci + cj + ck, q \in H : q = (a, u) = a + u_i i + u_2 j + u_3 k$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$\text{realteil} \rightarrow a \in \mathbb{R}, \text{imaginärteil} \rightarrow \begin{matrix} u \in \mathbb{R}^3 \\ u = u_1 i + u_2 j + u_3 k \end{matrix}$$

$$\text{Rechenregeln: } q = (a, u)^T, r = (b, v)^T$$

$$p = \begin{pmatrix} s \\ 1 \\ 0 \\ 0 \end{pmatrix} \text{ als quat.}$$

$$\rightarrow v = 0 + s i + u_1 j + u_3 k$$

$$\rightarrow q + r = (a+b, u+v)^T \rightarrow q$$

$$\rightarrow \langle q | r \rangle = a \cdot b + \langle u | v \rangle \quad \text{Skalarprodukt}$$

$$\rightarrow q \cdot r = (a + u_1 i + u_2 j + u_3 k) (b + v_1 i + v_2 j + v_3 k) \quad ! \text{ nicht kommut.}$$

$$\rightarrow \text{konj. } q^* = (a, -u)^T$$

$$\rightarrow \text{Norm } |q| = \sqrt{q \cdot q^*} = \sqrt{a^2 + u_1^2 + u_2^2 + u_3^2}$$

$$\rightarrow \text{Inverse } q^{-1} = \frac{q^*}{|q|^2}$$

$$\rightarrow \text{Vektor als Quaternion: } p = (x, y, z)^T \rightarrow q = (0, p)$$

$$\rightarrow \text{Skalar } s = 0 \rightarrow q = (s, 0)^T$$

Rot. Matrix \rightarrow Quaternion: (1) $R x = R \rightarrow$ Rotachse x ist EV zum EW 1

\rightarrow GLS nach x lösen

(2) Rot. Winkel berechnen: bestimme v sodass $v \cdot x = 0$

\rightarrow berechne $v' = R \cdot v$

Rot. Winkel ist Winkel zw. v, v'

$$\rightarrow \cos \alpha = \frac{\langle v, v' \rangle}{\|v\| \|v'\|}$$

$$\rightarrow q = \left(\cos \frac{\alpha}{2}, \vec{v} \cdot \frac{\vec{x}}{\|\vec{x}\|} \sin \frac{\alpha}{2} \right)^T$$

(3)

Rotationen Drehachse a mit $\|a\|=1$
Drehwinkel θ

$$q = (\cos \frac{\theta}{2}, a \sin \frac{\theta}{2}) \text{ Einheitsquaternion}$$

> Punkt v rotieren: $v' = q v q^{-1} = q v q^*$

> Konsat. zweier Rotationen von v mit Quat. q, r

→ Rot. mit $p = q \cdot r$

- Vorgehen:
- (1) Punkt p als Quaternion darstellen
 - (2) Rot. quat. q berechnen
 - (3) q^* berechnen
 - (4) Rot. berechnen
 - (5) Darstellen als Punkt
- (p, a, θ gegeben)

Interpolation: SLERP

$$\text{Slerp}(q_1, q_2, t) = \frac{\sin((1-t)\theta)}{\sin\theta} q_1 + \frac{\sin(t\theta)}{\sin\theta} q_2$$

$$\text{mit } \langle q_1, q_2 \rangle = \cos\theta$$

→ Rotation mit konst. Winkelgeschw.

Duale Quaternionen duale Zahlen der Form $d = p + \epsilon \cdot s$ mit $\epsilon^2 = 0$

$$DQ = (d_1, d_2, d_3, d_4)$$

$$d_i = d_p i + \epsilon \cdot d_s;$$

realer Skalarparte enthält Winkelwert $\theta/2$
imag. $-i -$ Translationsgr. d
restl. Dualzahlen beschr. belieb. gerichtete norm. Gerade

Rotation um Achse a , Winkel θ :

$$q_r = (\cos(\frac{\theta}{2}), a \sin(\frac{\theta}{2})) \mid + \epsilon \cdot (0, 0, 0, 0)$$

Translation mit Vektor $t = (t_x, t_y, t_z)$

$$q_t = (1, 0, 0, 0) + \epsilon \cdot (0, \frac{tx}{2}, \frac{ty}{2}, \frac{tz}{2})$$

Kombination: Transformation $q_T = q_t \cdot q_r$

Anwendung auf Punkt p (duales Quat.):

$$p' = q_T p q_T^* \quad q_T^* = (q_t q_r)^* = q_r^* q_t^*$$

Konjugieren: $q = p + \epsilon \cdot s \rightarrow q^* = p^* - \epsilon \cdot s^*$

Teilsysteme

Gelenktypen

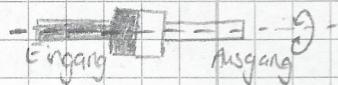
Rotationsgelenk: Drehachse bildet rechten Winkel mit Achsen der Glieder

z.B. Ellbogengelenk



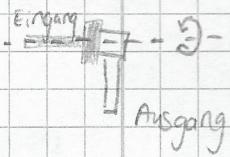
Torsionsgelenk: Drehachse verläuft parallel zu Achsen der Glieder

z.B. Unterarmdrehung

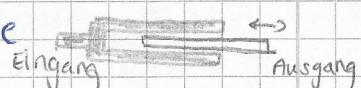


Revolvergelenk: Eingangsglied parallel zur Drehachse
Ausgangsglied rechter Winkel zur Drehachse

z.B. Schultergelenk (Arm nach vorne)



Lineargelenk: gleitende Bewegung entlang der Achse



Arbeitsraum

Punkte im 3D, die von Roboterhand angefahren werden können
≤ 6 Freiheitsgrade, also mind. 3 Gelenke erforderlich

Grundform: Arbeitsraum, der sich ergeben würde, wenn man gegenseitige Behinderung aller Arme und Begrenzung oder Gelenkwinkel nicht berücksichtigt

- > Kartesischer Roboter: Quader → nur Lineargelenke
- > Cylindrical Robot: 2 Lineargelenke + 1 Torsionsgelenk
- > Spherical robot: Kugel Lineargelenk + Torsionsg. + Rot.gel.

Paralleler Roboter: Stewart - Plattform



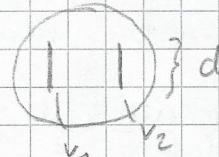
- 6 Freiheitsgrade

~ 3 rotatorisch, 3 translatorisch

Radkonfigurationen

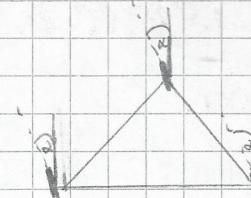
Differentialantrieb:

- Geradeaus und Kurvenfahrten
- Drehen auf der Stelle
- Vorwärts- & Rückwärtsfahren identisch
- ⊕ einfache Mechanik
- ⊖ Radregelung in Echtzeit



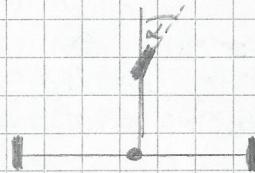
Synchro-Drive:

- Geradeaus- und Kurvenfahrten
- Vorw. - & Rückwärtsfahren identisch
- Plattform dreht nicht mit
- ⊕ einfache Regelung, Geradeausfahrt mechanisch garantiert
- ⊖ mechanische Komplexität



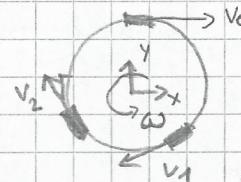
Dreirad-Antrieb:

- Geradeaus- und Kurvenfahrten
- Vorw. - & Rückwärtsfahren untersch.
- ⊕ einfache Mechanik
- ⊖ eingeschränkte Manövrierfähigkeit



Mecanum-Antrieb:

- ⊕ uneingeschränkte Beweglichkeit in Richtungen x, y, ω
- ⊖ mechanische Komplexität aufwendige Regelung



Antriebe

Fluidischer Antrieb

↳ fließgeschw. des Mediums

Linearantrieb: Kolbengeschw. $v(t) = f(t) / A$ \leftarrow Grundfläche Kolben

Kolbenkraft: $F(t) = P(t) \cdot A$

\leftarrow Druck des Mediums

Schaufelrad: Winkelgeschw. Kolben: $\omega(t) = 2 \cdot f(t) / ((R^2 - r^2) \cdot h)$
Höhe Schaufelrad
äußerer Radius, innerer Radius

Drehmoment Kolben: $T(t) = 0,5 \cdot P(t) \cdot h (R - r) (R + r)$

Muskelartiger Antrieb

Pneumatischer Antrieb:

- > Stellenergie: komprimierte Luft bewegt Kolben, keine Getriebe
- > Vorteile: billig, einfacher Aufbau, schnelle Reaktionszeit
- > Nachteile: laut, keine Steuerung der Geschw. bei Bewegung, schlechte Positioniergenauigkeit
- > Einsatz: kleinere Roboter mit schnellen Arbeitszyklen und wenig Kraft

Hydraulischer Antrieb:

- > Stellenergie: Öldruckpumpe + steuerbare Ventile
- > Vorteile: große Kräfte, mittlere Geschwindigkeit
- > Nachteile: laut, Verunreinigungen, schlechte Reaktionszeit und Positionier-/Wiederholgenauigkeiten
- > Einsatz: große Roboter, z.B. zum Schweißen

Elektrischer Antrieb

- > Stellenergie: Schritt-/Servomotoren
- > Vorteile: wenig Platzbedarf, leise, gute Regelbarkeit, hohe Positionier-/Wiederholgenauigkeit
- > Nachteile: wenig Kraft, langsam

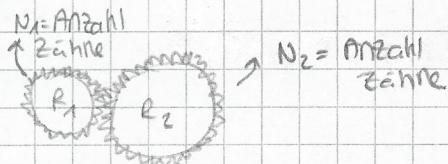
Gelenke Beispiele

Stirnradgetriebe

$$\text{Untersetzung } n = N_1 / N_2$$

$$\text{Winkelgeschw. } \omega_2 = n \omega_1$$

$$\text{Drehmoment } T_2 = T_1 / n$$



Schrauben- & Spindelgelenke

- > Lineargeschw. $v(t) = p \cdot \omega(t)$ $p = \text{Steigungskonst.}$, Entfernung welche Schraube bei Umdrehung zurücklegt
- > Kraft $F = \frac{2 \cdot \frac{1}{4} \pi \cdot d_m \cdot \mu \cdot p \sec \beta}{d_m \cdot p + \mu \cdot \pi \cdot d_m \cdot \sec \beta}$
 - Drehmom. \uparrow
 - winkelgeschw. \uparrow
 - mittl. Durchmesser der Schraube \leftarrow
 - Reibungskoeff. \uparrow
 - Gewindesteigung \uparrow

Harmonic Drive

- gutes Übersetzungsverhältnis
- sehr genaue Bewegung
- hohe Positioniergenauigkeit

mehr Beispiele:

Planetengetriebe, Schneckenradgetr., Leitspindel + Nutter, Seilzug, ...

Sensoren

Umwandlung phys. Größen und deren Änderungen in elektr. Signale

Ziel: Erfassung der Umwelt in nicht fest definierten / sich verändernden Umgebungen

Probleme: Signalverarbeitung, Sensorik liefert nur partielle Infos, Fusion der Messwerte bei mehreren Sensoren

Interne Sensoren: kein Kontakt zur Umwelt

- Bestimmung von Lage und Position
- Stellung der Gelenke
- Geschw. der Gelenke
- Kräfte + Momente, die auf Gelenke wirken

z.B. Neigungsmesser, Tachogenerator

Externe Sensoren: Information aus der Umwelt

- Bestimmung von Lage und pos. in Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos
- Entfernungsmessung
- Lage von Objekten
- Kohärenz von Objekten

z.B. aktiv: Ultraschall, Infrarot
passiv: Kameras, Mikros

Aktive Sensoren:

Simulation der Umwelt durch Eintrag von Energie, Messen und Auswerken der Antwort

Passive Sensoren: in Umwelt vorhandene Signale werden ausgemessen und ausgewertet

Kinematik

Robotermodellierung:

- (1) Geometrische Modellierung: mathem. Beschreibung der Form von Körpern
- (2) Kinematische Modellierung: Lehre der geom. und analytischen Beschreibung der Bewegungszustände mechanischer Systeme
- (3) Dynamische Modellierung: Untersuchung der Bewegung von Körpern als Folge der auf sie wirkenden Kräfte und Momente

Kinematisches Modell

- beschreibt zsm. hängt zw. Raum der Gelenkwinkel und Raum der Lage des Endeffektors in Weltkoordinaten

→ Einsatz für: Kollisionserkennung, Selbstkollisionserkennung, Erreichbarkeitsanalyse

Vorwärtskinematik:

direktes kinem. Problem: Bestimmung der Lage des Endeffektors aus Gelenkwinkelstellungen des Roboters

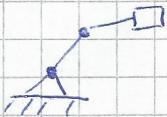
Inverser Kinematik:

Bestimmung der Gelenkwinkelstellungen zu gewünschter Lage des Endeffektors

Kinematische Kette: gebildet von durch Gelenke kinematisch verbundenen Körpern

Typen: offene kin. Kette

geschlossene kin. Kette



Kinematische Parameter:

→ Gelenkparameter: rot.-gelenk: Rot. Achse; Schubgelenk: Transl. Achse; ...

→ Spezifikation der Lage der Gelenke zueinander:

- feste Transformation zw. Gelenken

- Definiert lokale Koord.-sys. der Gelenke

- Transf. von Gelenk $i-1$ zu Gelenk i durch Transf.-matrix $i^{-1}T_i$

→ 6 Parameter pro Glied der kin. Kette (3 transl., 3 rot.)

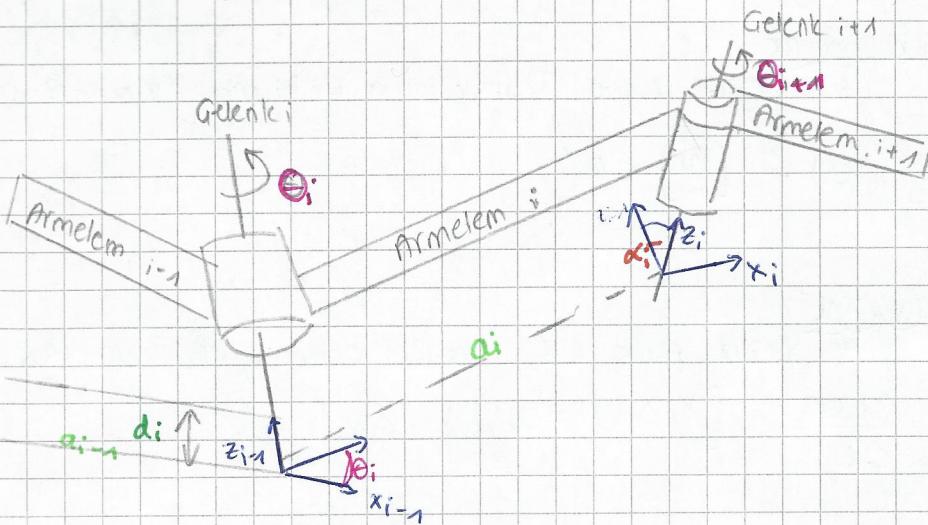
Denavit-Hartenberg Konvention (DH)

Reduktion der Anzahl Params von 6 auf 4

→ systematische Beschreibung der Beziehung (transl., rot.) zw. benachbarten Gelenken

Parameter des Armeelements

- jedes Armelement i durch Gelenke $i, i+1$ eingebunden
- z_i liegt entlang der Gelenkkarze $i+1$
- Armelementlänge a_i : Abstand von z_{i-1} zu z_i
- x_i entlang der Normalen von z_{i-1} zu z_i , zeigt weg von z_{i-1}
- Armelementverwindung α_i : Winkel von z_{i-1} zu x_i um z_i
- Gelenkabstand d_i : Abst. zw. x_{i-1} und x_i -Achse entlang z_{i-1} -Achse
- Gelenkwinkel θ_i : Winkel von x_{i-1} zu x_i um z_{i-1}



DH Parameter bestimmen
z-Achse immer in
Richtung 'Gelenk'
des Gelenks'

Transformation OKS_{i-1} zu OKS_i

- (1) Rotation θ_i um z_{i-1} -Achse → x_{i-1} parallel zu x_i -Achse
 - (2) Translation d_i entlang z_{i-1} -Achse
 - (3) Translation a_i entlang x_i -Achse
 - (4) Rotation α_i um x_i -Achse
- } alles in 4D!

→ Ergebnis $A_{i-1,i}$:

→ inverse Transformation OKS_i zu OKS_{i-1}: $A_{i-1,i}^{-1} = A_{i,i-1}$

→ Durch Verkettung der Matrizen: Lage einzelnes US bzgl. Bezugs.KS bestimmbbar:

Lage von USM bzgl. Basis:

$$S_{\text{Basis},m}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \dots \cdot A_{m-1,m}(\theta_m)$$

z.B. Endeffektorstellung ermitteln: $S_{\text{Basis},n} = \text{Greifer}$

→ Gelenkwinkel $\theta_1, \dots, \theta_n$ vorgegeben → einsetzen

Jacobi-Matrix: $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$J_f(a) = \left(\frac{\partial f_i}{\partial x_j}(a) \right)_{i,j} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \dots & \frac{\partial f_1}{\partial x_n}(a) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \dots & \frac{\partial f_m}{\partial x_n}(a) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

End-Effektor Geschwindigkeiten

$$\dot{x}(t) = J_f(\theta(t)) \cdot \dot{\theta}(t)$$

Kräfte und Moment am End-Effektor

$$\tau(t) = J_f^T(\theta(t)) \cdot F(t)$$

Orehmom. in Gelenken kraft-Momenten-Vektor am End-Effektor

Berechnung der Jacobi-Matrix

(1) Translationsgelenk

Annahme: j -tes Gelenk führt Translation in Richtung EV $v_i \in \mathbb{R}^3$ durch

$$\Rightarrow \frac{\partial f}{\partial \theta_j}(\theta) = [v_{ij}] \in \mathbb{R}^6$$

(2) Rotationsgelenk

Annahme: j -tes Gelenk führt Rot. um Rot.achse $v_j \in \mathbb{R}^3$ an Pos. $p_j \in \mathbb{R}^3$ durch

$$\Rightarrow \frac{\partial f}{\partial \theta_j}(\theta) = [v_j \times (f(\theta) - p_j)] \in \mathbb{R}^6$$

Singularitäten

- kinematische Kette ist in singularer Konfiguration wenn J_f nicht vollen Rang hat
(= zwei od. mehr Spalten lin. abh.)
- J_f nicht invertierbar \rightarrow bestimmte Bewegungen unmöglich
- in Umgebung von Singularitäten große Gelenkgeschw. nötig um Endeffektor-Geschw. zu halten

Manipulierbarkeit - Maß für Bewegungsfreiheit des Endeffektors

Manipulierbarkeits-Ellipsoid: abh. von Gelenkwinkelkonfiguration

\rightarrow bilden mit $J(\theta)$ Einheitskreis der Gelenkwinkel-Geschw.

\rightarrow Kreis: Bewegung des Endeff. in alle Richtungen uneingeschr. mögl.
Degenerierte Fälle (Linie): Endeff. bew. eingeschränkt

Eigenwertanalyse: $A(\theta) = J(\theta) \cdot J(\theta)^T \rightarrow$ symm., quadr., pos. def., invertierbar

λ : Eigenwerte, v : Eigenvekt. von A

$$\Rightarrow A v = \lambda v$$

$$(\lambda I - A)v = 0$$

$$\text{Singularwerte } \sigma_i = \sqrt{\lambda_i}$$

- > kleiner Singularwert: $\mu_1(\Theta) = \sigma_{\min}(A(\Theta))$
- > Inverse Kondition: $\mu_2(\Theta) = \frac{\sigma_{\min}(A(\Theta))}{\sigma_{\max}(A(\Theta))}$
- > Determinante: $\mu_3(\Theta) = \det A(\Theta)$

Geometrisches Modell

Einsatzbereiche: graphische Darstellung von Körpern,
 Ausgangspunkt der Abstandsmessung & Kollisionserkennung
 Grundlage zur Berechnung der Bewegungen von Körpern
 Grundlage zur Ermittlung der wirkenden Kräfte & Momente

- > Klassifizierung nach
 - Raum: 2D / 3D / ... Modelle
 - Grundprimitiven: Kanten- (Drahtmodelle) / Volumenmodell / Flächen- (oberfl.-) modell

Blockwelt: Körper als einhüllender Quader dargestellt
 ↳ ersten Schritten der Kollisionsvermeidung
 Klasse: 2,5D / Volumen / Flächen

Kantenmodell: Körper als Polygonzüge (Kanten)
 ↳ schnelle Visualisierung
 Klasse: 3D / Kanten / Flächen

Volumenmodell: Körper genau darstellen
 ↳ Ermittlung genauer Werte der Kollisionserkennung
 Klasse: 3D / Volumen

Inverse Kinematik

Bestimmung der Gelenkwinkelstellung
 zu gewünschter Lage des Endeffektors

Problem: geg. $P_{TCP} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$= \frac{B_{IK}}{TCP}(\Theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \dots \cdot A_{n-1,n}(\theta_n)$$

ges.: Θ Gelenkwinkel

Geometrische Methode nutzt geom. Beziehungen um Θ aus T_{TCP} zu bestimmen
 ~ trigon. Fkt., Sin / Cos Sätze

Werkzeug: Substitution $u = \tan\left(\frac{\Theta}{2}\right) \rightarrow \cos\Theta = \frac{1-u^2}{1+u^2}$
 $\sin\Theta = \frac{2u}{1+u^2}$

Algebraische Methoden Gleichsetzen TCP Pose, Transformation

$$P_{TCP} = \overset{\text{BGS}}{T_{TCP}(\theta)}$$

→ Koeffizientenvergleich → Gleichungen aufstellen + Lösen
(4 trivial $0=0, 1=1 \rightarrow 12$ nicht-triviale)

Vorgehensweise: $P_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \dots \cdot A_{5,6}(\theta_6) = *$

- (1) invertiere $A_{0,1}(\theta_1)$, multiplizierte beide Seiten von * mit $A_{0,1}^{-1}$
- (2) Gleichung mit 1 unbekannten finden → Lösen
- (3) einsetzen, Gleichung mit 1 unbekannten ...
- (4) falls so nicht weiter: weitere Matrix invertieren
- (5) wiederholen bis alle θ ermittelt

Numerische Methoden

(1) Vorwärtssubstitution als Funktion $x(t) = f(\theta(t))$

TCP Pose $\in \mathbb{R}^6$ $\xrightarrow{\quad}$ \mathbb{R}^n Gelenkwinkelstellung

(2) Ableitung nach der Zeit

$$\frac{dx(t)}{dt} = \dot{x}(t) = J_f(\theta) \dot{\theta}(t)$$

$\xleftarrow{\quad}$ Gelenkgeschw.
 $\xrightarrow{\quad}$ TCP Geschw.

(3) Übergang zum Differenzenquotienten

$$\Delta x \approx J_f(\theta) \Delta \theta$$

$\xrightarrow{\quad}$ Fehler in TCP Pose $\xleftarrow{\quad}$ Fehler in Gelenkstellungen

(4) Umkehrung $\Delta \theta \approx J_f^\#(\theta) \Delta x$

$$\text{Pseudoinverse } J_f^\# \quad A^\# = A^T (A A^T)^{-1}$$

$$(A^\#)^\# = A ; \quad (A^T)^\# = (A^\#)^T ; \quad (\lambda A)^\# = \lambda^{-1} A^\# \quad (\lambda \neq 0)$$

iteratives Vorgehen beginne bei Initialkonfiguration θ_0 und $x_{TCP,0}$

- (1) berechne $x_{TCP,t}$ in Iteration t aus Gelenkust. θ_t
- (2) berechne Fehler Δx aus $x_{TCP,\text{soll}}$, berechne $x_{TCP,t}$
- (3) benutze approx. inverses kin. Modell F um $\Delta \theta$ zu berechnen
- (4) $\theta_{t+1} = \theta_t + \Delta \theta$
- (5) nächste Iteration t+1

Dynamik

Dynamisches Modell: beschreibt Zusammensetzung von Kräften, Momenten, Bewegungen, welche in mech. Mehrkörpersystem auftreten
Zweck: Analyse der Dynamik, Synthese mech. Strukturen, Modellierung elastischer Strukturen, Regerentwurf

Allgemeines Modell

Robo besteht aus n Partikeln mit Masse m_i und Pos. r_i :

$$F_i = m_i \cdot \ddot{r}_i \quad i=1, \dots, n$$

→ Partikel nicht unabh. voneinander beweglich (Verbindungen, Gelenke)
 $\leq k$ Constraints der Form $g_j(r_1, \dots, r_n) = 0 \quad j=1, \dots, k$
= holonome Einschränkungen
→ wirken auf Robo als constraint forces

Generalisierte Koordinaten min. Satz unabh. Koord., der aktuellen Systemzustand vollst. beschreibt

$$q_1, \dots, q_m \quad (m=3n-k)$$

gesucht: Funktionen für Pos. der Massenpunkte

$r_i = f_i(q_1, \dots, q_m) \quad i=1, \dots, n$
die gleichzeitig constraints $g_j(r_1, \dots, r_k) = 0$ enthalten

Bewegungsgleichung

$$\tau = M(q) \cdot \ddot{q} + c(q, \dot{q}) + g(q)$$

τ : $n \times 1$, generalisierte Kräfte

$M(q)$: $n \times n$, Massenträgheitsmatrix

$c(q, \dot{q})$: $n \times 1$, Zentripetale-, Corioliskomponenten

$g(q)$: $n \times 1$, Gravitationskomponenten

q, \dot{q}, \ddot{q} : $n \times 1$, generalisierte Koord. (Pos., Geschw., Besch.)

→ direktes dyn. Problem: geg. $\tau(t)$, $q(t_0)$, $\dot{q}(t_0)$, $\ddot{q}(t_0)$
ges. $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$

$$\begin{aligned} \tau &= M(q) \cdot \ddot{q} + c(q, \dot{q}) + g(q) \\ \rightarrow &\text{Lösen nach } q(t), \dot{q}(t), \ddot{q}(t) \end{aligned}$$

Inverses dyn. Problem: geg. $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$

ges. $\tau(t)$
→ Lösen durch einsetzen

Modellierung: Lagrange

Lagrange-Fkt.: $L(q, \dot{q}) = E_{kin}(q, \dot{q}) - E_{pot}(q)$

Bewegungsgl.: $\ddot{\tau}_i = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i}$ i-te Komponente

→ Ermitteln der Bew.gl. für jedes Gelenk i

$$\begin{aligned} 3D: (x, y, z) \\ \Rightarrow \frac{1}{2} m(x^2 + y^2 + z^2) \\ E_{kin} = \frac{1}{2} m v^2 \end{aligned}$$

(1) E_{kin}, E_{pot} berechnen

(2) als gen. koord. ausdrücken $L(q, \dot{q}) = \dots$

(3) Ableitungen berechnen $\ddot{\tau}_1 = \dots, \ddot{\tau}_2 = \dots$

$$\ddot{\tau} = H(q) \ddot{q} + C(q, \dot{q}) + g(q)$$

④ einfache, geschlossenes Modell, analytisch auswertbar

⑤ Berechnung in $O(n^3)$ → umfangreich, nur Antriebsmomente berechnet

$$E_{pot} = mg h$$

Modellierung: Newton-Euler

$$\text{Kraft } F_i = \underbrace{\frac{d}{dt} (m_i \cdot v_{s,i})}_{\text{Impuls}} = m_i \ddot{v}_{s,i}$$

$$\text{Drehmoment } N_i = \underbrace{\frac{d}{dt} (I_i \omega_{s,i})}_{\text{Drehimpuls}} = I_i \ddot{\omega}_{s,i}$$

Vorwärtsgleichungen: Beschl. $v_{s,i}, \omega_{s,i}$ von Armelem. i abh. von Beschl. vorhergehender Armelem.

bestimmt werden: Winkelgeschw. ω_i , Geschw. v_i , Beschl. \ddot{v}_i der Basis der koord.g.v.

Geschw. $v_{s,i}$, Beschl. $\ddot{v}_{s,i}$ der Massmittelpunkte der Armelem.

Rekursive Berechnung von Basis zum Greifer:

$$w_{i+1} = G_1(w_i, q_{i+1}, \dot{q}_{i+1})$$

$$\dot{w}_{i+1} = G_2(w_i, \dot{w}_i, q_{i+1}, \dot{q}_{i+1}, \ddot{q}_{i+1})$$

$$v_{i+1} = G_3(v_i, w_i, q_{i+1})$$

$$\ddot{v}_{i+1} = G_4(\ddot{v}_i, \ddot{w}_i, \ddot{w}_i, \ddot{q}_{i+1})$$

$$v_{s,i+1} = G_5(v_{i+1}, w_{i+1})$$

$$\dot{v}_{s,i+1} = G_6(\dot{v}_{i+1}, w_{i+1}, \dot{w}_{i+1})$$

Rückwärtsgleichungen: Kraft F_i , Drehmoment N_i auf Armelem. i abh. von \ddot{v}_{i+1} nachfolgenden Armelem.

bestimmt werden: F_i Kraft im Schwerpunkt von Armelem. i

Drehimpuls N_i

Kraftvektor f_i von Armelem. $i-1$ auf i

Momentvektor n_i

Skalares Drehm. τ_i am i -ten Gelenk

Kraft-Momenten-Satz: $F_i = m_i \ddot{v}_i = f_i - f_{i+1}$

Drehimpulssatz: $N_i = I_i \cdot \ddot{\omega}_i = n_i - n_{i+1} + (-s_i - p_i) \times f_i - p_i \times f_{i+1}$
Trägheitsensor

Rek. Ber. von Greifer zur Basis: $F_i = H_1(v_{s,i})$, $N_i = H_2(w_i, \dot{w}_i, q_i, \dot{q}_i)$

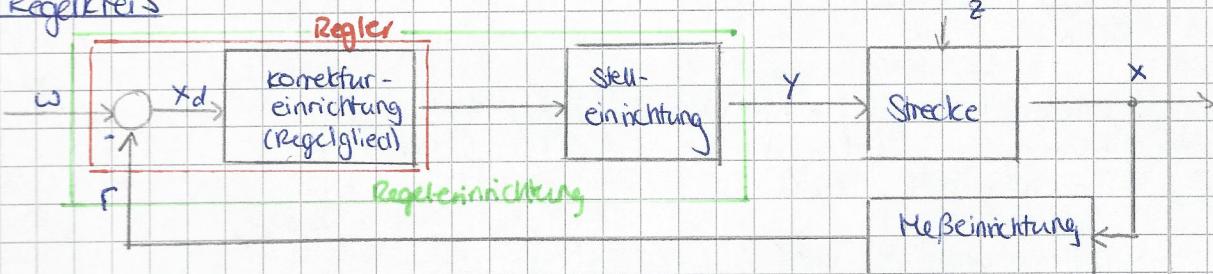
$$f_i = H_3(f_{i+1}, F_i, q_{i+1}), n_i = H_4(n_{i+1}, f_i, F_i, N_i, q_{i+1})$$

$$\tau_i = H_5(N_i, q_i)$$

Regelung

Regelung = Anordnung, durch welche bei unvollst. bekannter Strecke, unvollst. Kenntnis der Störgröße, die Regelgröße (= Ausgangsgröße der Strecke) laufend erfasst und mit Führungsgröße verglichen wird, um mittels der so gebildeten Differenz die Regelgröße an den Sollverlauf anzugelichen

Regelkreis



w = Führungsgröße

Sollwert von x: x_s

y = Stellgröße

r = Rückführgröße = $k_j \times (k_j > 0, \text{const.})$

x_d = Regelabweichung

x = Regelgröße

z = Störgröße

Wahl der Führungsgröße * $w = k_j \times s$

$$\Rightarrow x_d = w - r = k_j (x_s - x)$$

zunächst $x = x_s \Rightarrow x_d = 0$ (Regelung in Reihe)

z größer $\Rightarrow x$ abweichen $\Rightarrow r$ abgesenkt $\Rightarrow x_d$ angehoben

$\Rightarrow y$ angehoben $\Rightarrow x$ angehoben mit Tendenz x_s wieder anzunehmen

→ Störgröße wird ausgeregelt \sim Führungsgröße wird eingeregelt

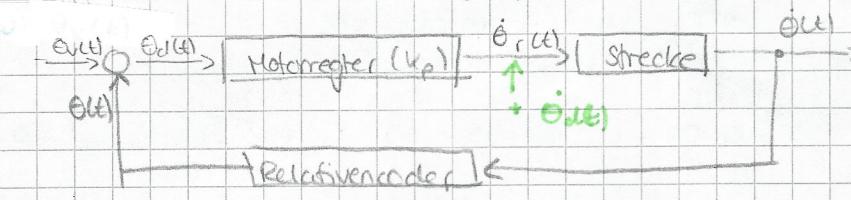
\Rightarrow Umkehr der Wirkungsrichtung im Soll-Y-Stwertvergleich

Geschwindigkeitsregelung

im Gelenkraum: kontinuierliche Vorgabe von Gelenkgeschw.

Prop. Regelung mit Faktor k_p $\dot{\theta}_r(t) = k_p \cdot (\theta_v(t) - \theta(t)) + \dot{\theta}_d(t)$

Regeldiff.



Nachteil: $\dot{\theta}_d = 0 \rightarrow$ keine Gelenkbewegung
 \rightarrow Lsg: Geschw. Vorsteuerung $\dot{\theta}_d(t)$

Laplace-Transformation

$$\mathcal{L}\{f(t)\} = f(s) = \int_0^\infty f(t)e^{-st} dt$$

$$s \in \mathbb{C}$$

$$s := \sigma + j\omega; f(t) = 0, t < 0$$

Gleichungslösung im Frequenzbereich statt im Zeitbereich

Ableitungsfunktion: Ann.: $\lim_{t \rightarrow \infty} e^{-st} f(t) \rightarrow 0$

$$e^{-st} = e^{-st}(\cos \omega t - i \sin \omega t)$$

$$\mathcal{L}\{f(t)\} = s \cdot f(s) - f(0)$$

Integral einer Funktion

$$L\left[\int_0^t f(t') dt'\right] = \frac{1}{s} f(s)$$

Impulsfunktion (Dirac) $L[S(t)] = 1$

existiert nur für $s > 0$

$$\text{Sprungfunktion } \sigma(t) \quad L[\sigma(t)] = \left[\frac{1}{s} \cdot e^{-st} \right]_0^\infty = \lim_{s \rightarrow 0^+} \frac{1}{s} = \frac{1}{s}$$

$$e^{-st} = \begin{cases} 0 & , s > 0 \rightarrow \text{Schwingung klingt ab} \\ 1 & , s = 0 \rightarrow \text{Oberschwingung} \\ -\infty & , s < 0 \rightarrow \text{Schwingung klingt auf} \end{cases}$$

Regeln

> Linearitätsatz $L\{\alpha f_1(t) + \beta f_2(t)\} = \alpha f_1(s) + \beta f_2(s)$

$$(f_1 + f_2)(t) \\ = \int_{-\infty}^{\infty} f_1(x) f_2(t-x) dx$$

> Faktionsatz $L\{f_1(t) * f_2(t)\} = f_1(s) * f_2(s)$

> Grenzwertsatz $f_1(t=0) = \lim_{s \rightarrow \infty} s * f(s)$

> Differenziationsatz $L\left\{\frac{d}{dt} f(t)\right\} = sF(s)$

> Integrationsatz: $L\left\{\int f(t) dt\right\} = \frac{1}{s} F(s)$

> Verschiebung: $L\{f(t-\tau)\} = e^{-\tau s} F(s)$

$$L\{e^{-at}\} = \frac{1}{s-a}$$

$$L\{t^n\} = \frac{n!}{s^{n+1}} \quad (n=1, 2, \dots)$$

$$L\{\sin(\omega t)\} = \frac{\omega}{s^2 + \omega^2}$$

$$L\{\cos(\omega t)\} = \frac{s}{s^2 + \omega^2}$$

Übertragungsglieder

PID - Regelung Proportional-Integral-Derivative Controller

$$T = K_p \theta_d + K_i \int \theta_d(t) dt + K_d \dot{\theta}_d$$

K_p = virtuelle Felder, die Positionsfehler reduzieren

K_d = virtuelle Dämpfer, der Geschw. fehler reduziert

K_i = reduziert Regelabweichungen

Stabilität einer Regelung Ziel: Regelabw. geht mit der Zeit gegen 0

Annahmen: Roboter bewegt sich in horizontaler Ebene

Zeitposition halten bei konstantem θ_d

$$\rightarrow \ddot{\theta}_d + 2\zeta\omega_n \dot{\theta}_d + \omega_n^2 \theta_d = 0 \rightarrow s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

3 Lösungstypen:

(1) $\zeta > 1$ aperiodische Lösung: $\theta_d(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \rightarrow$ keine Schwingung
 $s_{1,2} = -\frac{1}{\zeta}, \pm j\sqrt{\omega_n^2 - \zeta^2 \omega_n^2}$

(2) $\zeta = 1$ aper. Grenzfall: $\theta_d(t) = (c_1 + c_2 t) e^{-\zeta \omega_n t} \rightarrow \tau = \frac{1}{\zeta \omega_n}$
 \rightarrow gerade keine Schwingung

(3) $\zeta < 1$ gedämpfte Schwingung: $\theta_d(t) = (c_1 \cos(\omega_n t) + c_2 \sin(\omega_n t)) e^{-\zeta \omega_n t} \rightarrow$ Überschwingt

Testfunktionen

Impulsfkt., Sprungfkt., Anstiegsfkt., Harmonische Fkt.

Elementare Übertragungsglieder (2)

| Benennung | Funktionalbeziehung | Symbol |
|------------------------------|--------------------------------|--------|
| S-Glied Summenglied | $y(t) = \pm u_1(t) \pm u_2(t)$ | |
| KL-Glied Kennlinienglied | $y(t) = K \cdot F(u(t))$ | |
| M-Glied Multiplizierglied | $y(t) = K \cdot u_1(t)u_2(t)$ | |

Elementare Übertragungsglieder (1)

| Benennung | Funktionalbeziehung | Symbol |
|--|--|--------|
| P-Glied Proportionalglied | $y(t) = K \cdot u(t)$ | |
| I-Glied Integrierglied | $y(t) = K \cdot \int_0^t u(\tau)d\tau$ | |
| D-Glied Differenzierglied | $y(t) = K \cdot \dot{u}(t)$ | |
| T _t -Glied Totzeit-Glied | $y(t) = K \cdot u(t - T_t)$ | |

Regler

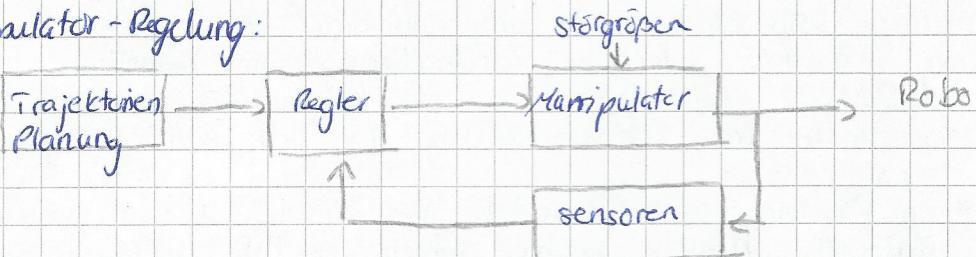
Zustandsregler: Verbessertes Regelverhalten \rightarrow Regelabweichung + (alle) Zustandsgrößen der Regelstrecke verfügbar

Kaskadenregler: Manipulator = Mehrgrößensystem
unabh. lineare Einzelregelkreise der einzelnen Gelenke

Adaptive Regelung: Lageabh. und somit zeitveränderliche Systemteile werden als Parameterschwankungen aufgefasst

Regelungskonzepte für Manipulatoren

Manipulator-Regelung:



\rightarrow Regelung von Manipulatoren beinhaltet auch Einbeziehung von Umwelteinflüssen

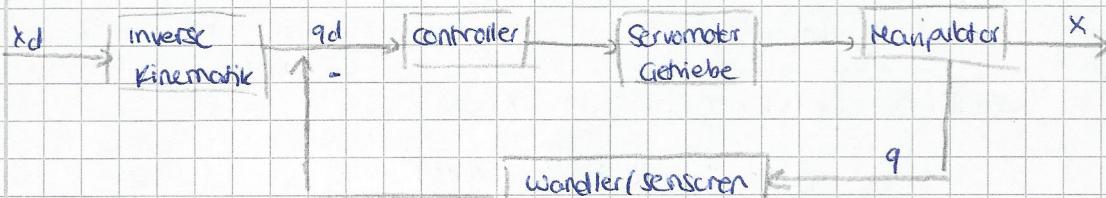
\rightarrow Einbeziehen von Gravitations-, Zentrifugal-, Coriolis-, Reibungskräfte, -momente auf Gelenke

$$\text{Stellkräfte } \vec{Q} \rightarrow \vec{Q} = M(\vec{q}) \ddot{\vec{q}} + n(\dot{\vec{q}}, \vec{q}) + g(\vec{q}) + R\vec{q}$$

$n \times 1$ aktig
 $n \times n$ Trägheit
 $n \times 1$ Zentrifugal-
 $1 \times n$ Wkomp.

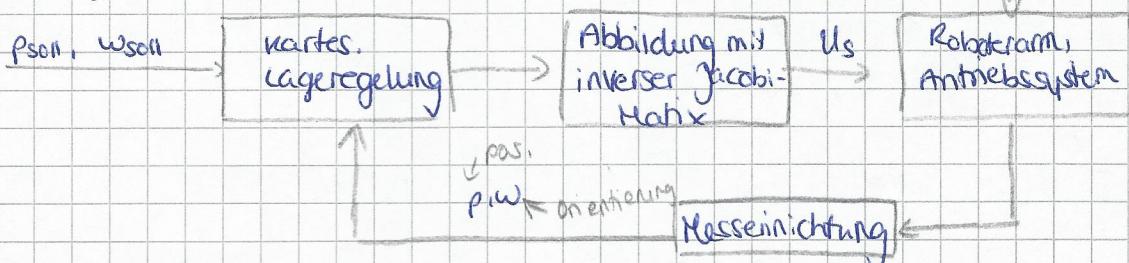
 $n \times n$ Diag. Matrix Reibungskräfte
 $n \times 1$ Winkellagen des Manipulators
 $n \times 1$ Gravitationscomp.

Regelung im Gelenkwinkelraum



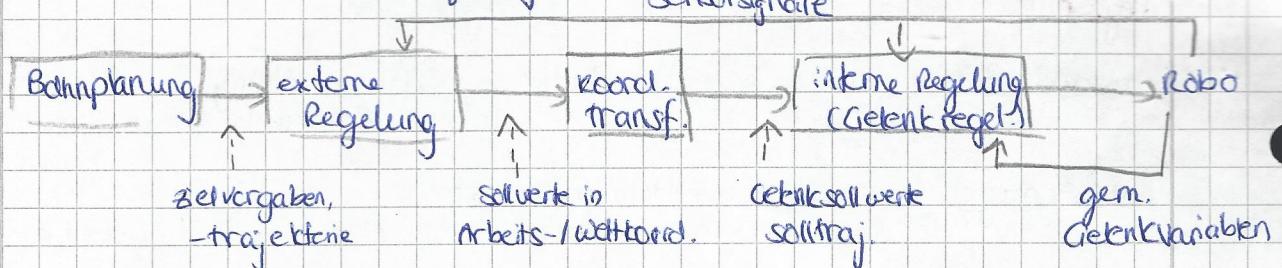
Koordinatentransformation: Soll-Trajektorien im Gelenkwinkelraum

Regelung im kartesischen Raum



höhere Komplexität des Algorithmus
aber: direkte gezielte Beeinflussung der einzelnen Raumkoord.

Struktur einer Roboterregelung



Kraft-/Positionssregelung

zur Ausführung von Aufgaben, die Interaktionskräfte berücksichtigen müssen

Problem: Positionen + Kräfte eng miteinander verknüpft

Lösung: erweiterte natürliche Randbedingungen um künstliche

→ Reine Kraft- oder Positionssregelung für jede kart. Bewegungsrichtung des Arms

» Achsen Regelung: Einschließen einer Box

- kartesische Achsen unabh. geregelt → PID-Regler

- Problem: Reibung berücksichtigen

» Achsenüberlagerte Regelung: Spiralsuche

- Kraft- und Positionssregelung auf X und Y Achsen

Impedanz-Regelung

Regelt dynamische Beziehung zw. Kraft + Pos. im Kontaktfall

→ Interaktion Robo ↔ Umwelt ähnlich Feder-Dämpfer-Masse-System

$$f(t) = d \cdot x(t) + b \cdot \dot{x}(t) + m \cdot \ddot{x}(t)$$

↓ Lapt. - Transf.

$$F(s) = (d + b \cdot s + m \cdot s^2) \cdot X(s)$$

Impedanz des
Euler-D. - K. Syst.

Bahnsteuerung

Zustände darstellbar im

> Gelenkwinkelraum \mathbb{R}^n

⊕ näher an Steuerung der Teilsysteme des Roboters

> kartesischen Raum \mathbb{R}^3

⊕ näher an zu lösender Aufgabe

⊖ Lösen inverser Kinematik nötig

Interpolation der Weltkoord.:

Start-, Zielkoord.

→ Interpolation

Interpol. Koord.

→ Inverse Kinematik

Interpol.

Gelenkwinkel

Interpol. der Gelenkwinkel:

Start-, Zielkoord.

→ Inverse Kinematik

Gelenkwinkel

→ Interpolation

Interpol.

Gelenkwinkel

Bahnsteuerung im Gelenkwinkelraum → Funktion der Gelenkwinkelzustände

Abfahren der punktweise spezifizierten Trajektorien:

> asynchron: Achsensteuerung unabh. voneinander

> synchron: achsinterpoliert Steuerung → Bewegung aller Achsen beginnt, endet
zum gleichen Zeitpunkt
→ Leitachse

⊕ Gelenkansteuerung einfacher, Trajektorie eindeutig und berücksichtigt Gelenkwinkelgrenzen
⊖ Interpolation für mehrere Gelenke, Formulierung der Trajektorie umständlich

Kartesischer Raum → Funktion der Zustände des Roboters

Endeffektor folgt in Lage und Orientierung einer definierten Bahn

⊕ Bahn einfacher zu formulieren, Interpolation einfacher

⊖ Inverse Kinematik für jeden Trajektoriepunkt lösen, geplante Trajektorie nicht immer ausführbar

Interpolationsarten

Punkt-zu-Punkt-Steuerung (PTP)

Sequenz von Gelenkwinkelvektoren

$$q(t_j) = (q_1(t_j), q_2(t_j), \dots, q_n(t_j))^T$$

⊕ Gelenkwinkeltraj. ber. einfach

keine Probleme mit Singularitäten

Randbed.: Start, Zielzustand bekannt

Gelenkwinkelbereiche, Geschw., Beschl. begrenzt
Geschw., Beschl., ... zu Beginn und Ende = x

- (1) Berechnung zu fahrbare Strecke / Winkelstrecke
- (2) Modifikation von v_m, b_m , Berechnung t_e, t_b, t_v , Interpolation
- (3) Ermittlung Gelenksollwerte

Interpol. mit Rampenprofil: ruckartige Beschle., kann zu Eigenschwingungen mech. Teile führen

mit Sigmoidprofil: längere Beschle./Bremsphase, Robo weniger Belastung

synchrone PTP-Bahnen: PTP-Params für jedes Gelenk;
 ~ bestimme $t_e = t_{e,\max} = \max(t_{e,i})$
 (Achse mit max. Fahrzeit)
 => Leitachse
 ~ setze $t_{e,i} = t_e$ für alle Gelenke
 ~ alle Gelenke beginnen/beenden Bewegung gemeinsam

asynchrone PTP-Bahnen:
 jedes Gelenk sofort mit max. Beschl. angesteuert
 ~ Gelenkbew. enden unabh. voneinander

Approximierte Bahnsteuerung

Bahninkpol. = Bahn verläuft durch alle Stützpunkte der Trajektorie
 Bahnapprox. = Kontrollpunkte beeinflussen Bahnverlauf, werden approx.

Überschleifen

zum Zeitpunkt $t_j - \varepsilon$ beginnen, Parameter der Teiltraj. $j \geq 1$ auf Params der Teiltraj. j zu überführen

> Geschw.überschl.:

- Beginn wenn Geschw. Minim.wert unterschreitet
- (1) Abh. vom Geschw.profil

> Pos.überschl.:

- Beginn wenn TCP in Überschleifkugel eintritt
- auf/berhalb: Bahn exakt einhalten
- (4) gnt kontrollierbar

Konfiguration $q \in C$ = Zustand eines Robotos
 Lage + Orientierung im eukl. Raum
 Gelenkwinkelvel. in Gelenkws.raum
 ? od.

Bewegungsplanung

Geg.: Konfigurationsraum C
 Startkonf. $q_{\text{start}} \in C$
 Zielkonf. $q_{\text{ziel}} \in C$

Ges.: stetige Traj. $\tau: [0, 1] \rightarrow C$
 mit $\tau(0) = q_{\text{start}}$
 $\tau(1) = q_{\text{ziel}}$

unter Berücksichtigung von: Gütekriterien, Neben-, Rand-, Zwangsbed.

Arbeitsraum: ω kartesischer Raum \mathbb{R}^6 , Tool center Point (TCP)

Konfigurationsraum C : $C_{\text{free}} = \text{kollisionsfreie Konfigurationen} = C \setminus C_{\text{obs}}$ } $C = C_{\text{free}} \cup C_{\text{obs}}$
 \hookrightarrow Raum aller mögl. Konf. $C_{\text{obs}} = \text{Konf. die zu Kollision führen}$
 $=$ Hindernisraum: Menge aller Konfigurationsraumhindernisse
 $C_{\text{obs}} = \bigcup C_H$

> Globale Randbed.: limitieren gültigen Konfigurationsraum
 > Lokale ...: limitieren Übergänge zw. Konfigurationen

> Vollständiger Algo: findet für spezielle Planungsprobleme mind. 1 Lsg / erkennt in endl. Zeit, dass keine Lsg. existiert

> Randomisierter Algo: verwenden Zufallsgrößen um Ablauf zu steuern

> Auflösungsvollst. Algo: approx. Algo für diskretisierte Problemstellung vollständig

> Probabilistisch-vollst. Algo: findet mind. 1 Lsg falls existent, keine Aussage falls keine Lsg.

Pfadplanung für mobile Roboter

Geg.: 2D Weltmodell, q_{start}, q_{ziel}

Ges.: günstigste Verbindung von q_{start} nach q_{ziel}

(1) konstruiere Wegennetz W

(2) Suche in W

(1) Voronoi-Diagramme Region = Menge aller Punkte, deren Abstand zum Zentrum geringer als zu allen anderen Zentren

Geg. Punktmenge P → teilen in gleichgroße P₁, P₂ → ... →

Fall 1: 2 Punkte

Mittelsenkrechte

Fall 2: 3 Punkte

Mittelsenkr. aller Punktpaare um Schnittpunkt abschneiden

→ verbinde nächste Nachbarn aus P₁, P₂ entlang Trennungslinie

→ einzeichnen, abschneiden neuer Mittelsenkrechten

⊕ max. Abst. zu Hindernissen, Wegprüfung mit Abstandssensoren einfach

⊖ i.d.R. nicht kürzester Weg, wenige Hindernisse → wenige Wege

Sichtgraphen Verbinde jedes Paar von Eckpunkten auf Rand von C_{free} durch Linie falls Segment kein Hindernis schneidet

⊕ kürzester Weg, Methode exakt

⊖ Wege nicht zwingend kollisionsfrei → Hinderniskanten evtl. wegsegmente

→ Erweiterung der Hindernisse um Roboform → nicht kürzeste Wege

zellzerlegung zerlege C_{free} in Zellen, stelle Nachbarschaft in Graph dar
suche opt. Weg im Graphen

(1) exakte Zellzerlegung: Zellen überlappen nicht, $\bigcup_i z_i = C_{free}$

(2) approx. Zellzerlegung: Zellen von vorgefesterter Form, Zelle nicht vollst. in C_{free}?
→ verringe Größe, zerlege Zelle weiter

(2)

Baumsuche Darstellung des Kontrahumes als Quadtree

→ rek. Unterteilung in Kacheln, Kacheln entw. frei od. Hindernis

→ finde freie Kacheln von Start nach Ziel

kürzeste Weg /
@P/...

A*-Algorithmus Bestensuche, finde opt. Pfad von Start nach Ziel (Pfeilkosten min.)

O = open set = zu besuchende Knoten

C = closed set = besuchte Knoten

Update für K_i bes. Knoten v_n:

pred(v_n), akt. Kum. Kosten um v_n zu erreichen g(v_n), Heuristik für erwartete Kosten zum Ziel h(v_n)

Initialisierung:

O = {v₀}, C = {}, g(v₀) = ∞ (1 ≤ i ≤ k), g(v₀) = 0

while O ≠ ∅:

- bestimme zu erweiternden Knoten: v_i ∈ O mit min. f(v_i) = g(v_i) + h(v_i)
- v_i = v_{ziel} → Lsg. gefunden
- O.remove(v_i), C.add(v_i)
- Update für alle Nachfolger v_j von v_i: v_j ∈ C → überspringe v_j

21

Init

$$O = \{v_S\}, C = \{\}, g(v_i) = \infty \quad (1 \leq i \leq k), h(v_S) = 0$$

while $O \neq \emptyset$:

- bestimme zu erweiternden Knoten: $v_i \in O$ mit min. $f(v_i) = g(v_i) + h(v_i)$

- $v_i = v_{\text{ziel}}$? \rightarrow Lsg. gefunden

- O.remove(v_i)

C.add(v_i)

- Update für alle Nachfolger v_j von v_i :

- $v_j \in C$? überspringe v_j

- $v_j \notin O$? O.add(v_j)

- $g(v_i) + \text{cost}(v_i, v_j) < g(v_j)$?

$$\cdot g(v_j) = g(v_i) + \text{cost}(v_i, v_j)$$

$$\cdot h(v_j) = \text{heuristic}(v_j, v_{\text{ziel}}) \rightarrow \text{meist eukl. Distanz}$$

$$\cdot \text{pred}(v_j) = v_i$$

\rightarrow findet opt. Lsg. wenn Heuristik zulässig (überschätzt min. Kosten nicht)
optimale Effizienz

Potentialfeld-Methode Robo bewegt sich unter Einfluss von Kräften eines Potentialfelds

$$U: C_{\text{free}} \rightarrow \mathbb{R}$$

$$\text{Kraft in Punkt } q \text{ des Pot. feldes: } F(q) = -\nabla U(q)$$

Ablösendes Potential erzeugt von Tendenzen, Einfluss nur falls Abstand zu Robo $\leq p_0$

$$U_{\text{ab}}(q) = \begin{cases} \frac{1}{2} V \left(\frac{1}{p(q, q_{\text{obs}})} - \frac{1}{p_0} \right)^2 & \text{für } p(q, q_{\text{obs}}) \leq p_0 \\ 0 & \text{sonst} \end{cases}$$

$p = \|q - q_{\text{obs}}\|$ $(F_{\text{ab}} = -\nabla U_{\text{ab}})$

Anziehendes Potential möglichst nur ein Min. in q_{ziel}

> Lineare Fkt.: $U_{\text{an}}(q) = k \cdot \|q - q_{\text{ziel}}\|$

\rightsquigarrow für kleine Distanzen
große Kraft

$$F_{\text{an}}(q) = -k \cdot \frac{q - q_{\text{ziel}}}{\|q - q_{\text{ziel}}\|}$$

> quadr. Fkt.: $U_{\text{an}} = k \cdot \frac{1}{2} \|q - q_{\text{ziel}}\|^2$

} oft Kombin.:
lin. falls weit weg vom Ziel
quadr. falls nah am Ziel

$$F_{\text{an}}(q) = -k \cdot (q - q_{\text{ziel}})$$

\Rightarrow Summe der Kräfte bestimmt Bewegungsrichtung

$$U(q) = U_{\text{an}}(q) + U_{\text{ab}}(q)$$

$$\rightsquigarrow F(q) = F_{\text{an}}(q) + F_{\text{ab}}(q)$$

Bewegungsplanung für Manipulatoren

Probabilistic Roadmaps (PRM)

(1) Vorverarbeitung: Erzeugung eines kollisionsfreien Graphen durch wählen zufälliger Punkte (Sampling)

(2) Anfrage: verbinde Start, Ziel mit Graphen, suche Weg \rightarrow z.B. mit A*

⊕ mehrere Anfrage effizient

⊖ stark abh. vom Verwendeten Sampling,
nur approximativ

Dynamic Roadmaps (DRM)

Vorverarbeitung: Approx. Konfig. Raum durch Roadmap

Approx. Arbeitsraum durch Voxel

Abb. QWC von Voxel \rightarrow Roadmap

Anfrage: ermittle Voxel mit Hindernis \rightarrow lösche zugeh. Knoten, Kanten aus

Anpassen der Roadmap \rightarrow Roadmap, verbinde Start, Ziel mit Graph
Planen in angepasster Roadmap

Distance Aware - DRM: alle Voxel in Sicherheitsabst. löschen
nahe Kanten mit höherem Gewicht versehen

Rapidly-exploring Random Trees (RRTs)

probabilist., vollst., rand. Algo

Algo zur Einmalanfrage
keine Vorverarbeitung

\rightarrow Form von CoBs unbekannt

Init.: leerer Baum T , füge q_{start} in T ein

Iteration:

- (1) erzeuge zufälligen Punkt q_s
- (2) bestimme nächsten Nachbar q_{nn} in T
- (3) Füge Punkte auf Verbindung q_s, q_{nn} in T ein
 - mit Schrittweite d
 - prüfe jeden Teilpfad auf Kollision mit CoBs
 - steigere bei Kollision

(4) gehe zu 1

\rightarrow prüfe in jedem 4-ten Schritt ob Ziel mit T verbindbar

Kollisionsprüfung: CCD (continuous collision detection)

Sampling basiert: einzelne Punkte auf Pfad prüfen

Bidirektionale RRTs: zwei Bäume aufbauen, T_1 von q_{start}

T_2 von q_{ziel}

\rightarrow zufällige Punkte erweitern T_1, T_2

\rightarrow Lsg. gefunden falls beide Bäume mit q_s verbunden

Nachbearbeitung: zufällige Wahl zw. Knoten im Lsg. Weg: Verbindung kollisionsfrei? Knoten verbinden, zwischenliegende Knoten löschen
 \rightarrow glattere Trajektorie

Constrained RRT

Projiziere Stichprobe q_s auf q_s' die Nebenbed. erfüllt

Randomized Gradient Descent:

- Toleranzwert für Nebenbed.: α
- Zufällige Bestimmung von n Nachbarn von q_s (in Hypersphäre mit Rad. d_{max})
- Distanz eines Nachbarn zu C_{UB} kleiner als Dist. q_s zu C_{UB} : ersetze q_s mit Nachbarn
- Wdh. bis max. Iterationszahl / Distanz q_s zu $C_{UB} \leq \alpha$

First Order Retraction:

$$q_s' = q_s - J(q_s)^{\#} \Delta x_s$$

Abst. von q_s zu C_{UB} im Arbeitsraum

RRT* optimiert Suchbaum iterativ \rightarrow asympt. Optimalität

- (1) $q_s = \text{randomConf}(C)$ // zufällige Konf.
- (2) $q_{nn} = \text{NearestNeighbor}(q_s, T)$
- (3) $q_{new} = \text{steer}(q_{nn}, q_s, d)$ // von q_{nn} mit Schrittweite d in Richtung q_s
- (4) if ! CollisionFreePath(q_{nn}, q_{new}) \rightarrow (1)
- (5) $Q_{near} = \text{Near}(T, q_{new}, r)$ // alle Knoten aus T mit Abstand $\leq r$ zu q_{new}
- (6) $q_{min} = \text{MinCostPath}(Q_{near}, q_{new})$ // $q_{min} \in Q_{near}$ sodass $\text{Cost}(q_{min}) + \text{Cost}(q_{min}, q_{new})$ min von Start zu q_{min}
- (7) AddPath(T, q_{min}, q_{new})
- (8) Rewire(T, q_{new}, Q_{near}) // $\forall q_{near} \in Q_{near}$:
 $\text{cost}(q_{new}) + \text{cost}(q_{new}, q_{near}) < \text{cost}(q_{near})$:
 \rightarrow ersetze Verbindung
- (9) if ! Timeout \rightarrow (1)

Eng Passagen Ideal: Sampling nur in sichtbarer Voronoi-Region

Dynamic Domain: Approx. durch Kugel mit Radius r

Bridge Sampling: wähle zielgerichtete Punkte in engen Passagen für nächste Stichprobe

- (1) wähle gleichverteilten zufälligen Punkt $q_1 \in C_{obs}$
- (2) wähle zweiten Punkt $q_2 \in C_{obs}$ nahe q_1
- (3) Mittelpunkt q_s zw. $q_1, q_2 \in C_{free} \rightarrow$ verwenden als neuen Punkt für RRT

Greifplanung

menschliche Hand: 27 DoF

Grifftaxonomie:

- Vereinfachung der Griffsynthese (Kontaktpunkte auf Objekt)
- Benchmark für Evaluation von Roboterhänden
- Grundlagen für das Design von Roboterhänden
- Einsatz bei autonomer Greifplanung

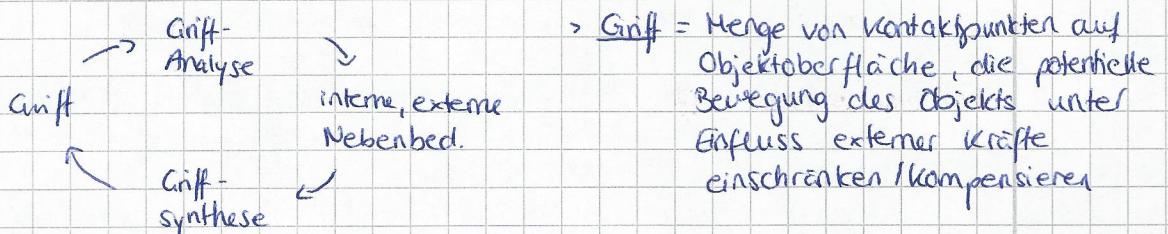
Skizze Grifftaxonomie:

• 16 Griffarten

• Hierarchiebaum: fasst Griffarten zu Gruppen zusammen

• erste Ebene: untersch. Kraft-, Präzisionsgriffe

→ aus der Beobachtung von Mechaniken → Fokus auf Verwendung von Werkzeugen



> Greifanalyse:
Geg: Objekt + Griff als Menge von Kontaktpunkten
Ges: Aussagen zu Stabilität des Griffes unter Berücksichtigung von Nebenbed.

> Greifsynthese:
Geg: Objekt + Menge von Nebenbed.
Ges: Menge von Kontaktpunkten (Griff)

Kontaktmodelle

Fingerspitzenkontakte mit der Objektoberfläche

$$\text{Reibungswinkel } \varphi_{\text{eff}} \\ \beta = \arctan(\mu)$$

> Punktkontakt ohne Reibung: auf Fläche angreifende Kraft wirkt nur normal zur Fläche

> starrer Punktkontakt mit Reibung: angreifende Kraft wirkt normal und tangential zur Fläche, Kräfte über Coulombsches Reibungsgesetz verknüpft

> soft-Kontakt = nicht starrer Punktk. mit Reibung: angreifende Kraft wirkt normal und tangential. Zusätzl. wirken axiale Momente. es gilt das Coulomb. Reib. g.

Wrenchvektor Kräfte f_i auf Kontaktpunkt p_i , Momente T_i ; $i \in \{x, y, z\}$ als Vektor zusammenfassen:

> planarer Griff: $w = (f_x, f_y, T_z)^T \in \mathbb{R}^3$ $T = d \times f$ $d =$ Vektor vom Schwerpunkt zum Kontaktpunkt

> räumlicher Griff: $w = (f_x, f_y, f_z, r_x, r_y, r_z)^T \in \mathbb{R}^6$

abh. vom Typ des von p_i folgen Wrenchvektoren, die an p_i wirkenden normalen (n), tangentiale (t), axialen (a) Kräfte/Momente beschreiben

→ w_n, w_t, w_a (skalare: c_n, c_t, c_a)

Greifmatrix für räuml. Griff: wrenchvektoren als Spalten einer Matrix

$$G = [{}^1w_n, {}^1w_t, {}^1w_o, \dots, {}^m w_n, {}^m w_t, {}^m w_o] \in \mathbb{R}^{6 \times 3m}$$

m = Anz. Kontaktpunkte

G repräsentiert geom. + physik. Eig. eines Fingerspitzengriffs

Gleichgewichtsgriß Summe aller Kräfte und Momente auf gegr. Objekt = 0

Kraftgeschlossener Griff Griff kompensiert alle externen Kräfte + Momente

- ohne Reibung: planarer Griff: mind. 4 Kontaktpunkte \rightarrow Ursprung innerhalb der konvexen Hülle des Wrenches, ϵ -Metrik Vorgehen:
- Wrenches am Kontaktpunkt berechnen 3D Griff: max. 12 Kontaktpunkte
- GWS bestim. mit Reibung: planarer Griff: 3 Kontaktpunkte min. Abstand zum Rand $\epsilon > 0$ (ϵ -Metrik)
- min. Abstand 3D: mind. 6 Kontaktpunkte d.h. Ursprung nicht auf dem Rand
- Ursprung, Rand GWS

Formgeschlossener Griff für jeden Kontaktpunkt ausschl. Nichtdurchdringungseig. co-lineär zum korresp. externen Oberflächen-Normalenvektor berücksichtigt

\rightarrow wechs. Normal-/Tangentialkräfte / Drehmomente u.a. von Reibung berücksichtigt

zu Kontaktpunkten korresp. externen Oberfl.-Normalenvektoren spezifizieren Kontakt geometrie des Fingerspitzengriffs

$$\tilde{G} = [{}^1w_n, {}^2w_n, \dots, {}^mw_n] \in \mathbb{R}^{3 \times m}$$

planar: mind. 4 Kontaktpunkte

3D: mind. 7 Kontaktpunkte

\Rightarrow Kraftschluß: Kinematik der Hand kann aktiv Kräfte erzeugen um externen Störung zu widerstehen

Formschluss:kontakte an sich verhindern Objektbewegung

\rightarrow Formschl. \Rightarrow Kraftschl.

Gleichgew. Kraftschl. mit weniger Kontaktpunkten mögl. \rightarrow Präzisionsgriffe



Stabile Griffe einbeziehen von Fingerkräften Potentialfunktion $V: \mathbb{R}^6 \rightarrow \mathbb{R}$ spezifiziert in Griff gespeicherte pot. Energie in Abh. von Lage, Orientierung des gegriffenen Objekts

$\delta q = (s_x, s_y, s_z, s_\alpha, s_\beta, s_\gamma) \in \mathbb{R}^6 \neq 0$ infinitesimale Lageänderung des Objekts

δV = daraus resultierende Veränderung der pot. Energie

dann: Griff stabil, falls $\nabla \delta q \in \mathbb{R}^6: \delta V > 0$

Suchraum Griffplanung: Suchraum hat Dimension $6+n$
6 für Pos. + Orientierung der Hand im Raum
n Anzahl konf. param. der Handfinger

Griffsynthese durch Vorwärtsplanung

- (1) Hand-, Objektmodell in Simulationsumgebung
- (2) Erzeuge Griffkandidaten
- (3) Evaluation der Griffkandidaten (Kraftschluss-Metrik: Griffqualität)

Kraftschluss-Metrik: Wie gut kann Griff externen Kräften widerstehen?

- bestimme Kontaktpunkte + -normalen
- bestimme Reibungskoeffizienten jedes Kontaktpunkts
- berechne GWS als konvexe Hülle über alle Reibungskoeffizienten
- E-Metrik: min. Distanz vom Zentrum zum Rand des GWS ist Maß für Griffstabilität

Grasp Wrench Space (GWS): konvexe Hülle über Vereinigung aller Kontakt-Wrenches

Zufallsbasierte & Vorwärts-Greifplanung

- (1) Random. Erzeugung von Greifhypothesen
- (2) Kontaktbestimmung
- (3) Evaluation der Hypothesen (Kraftschluss, Kollision, Robustheit)

Handmodell: Grasp Center Point = definiert Zentrum g und Anfangsrichtung a für Grifftyp

Erzeugung von Greifhypothesen: bestimme Anfangsrichtung: zufällige Wahl eines Oberfl. Punkts, \rightarrow Ermittlung der Oberfl. normalen n

Bestimmung Greifhypothese: Posit. der Hand + Kontaktbestimmung

\rightarrow Speichere valide Griffe

Griffsynthese auf Objektteilen

erzeuge gute Griffkandidaten durch Objektaproximationen

\rightarrow

Formprimitive Objekte durch Formprim. dargestellt
für jedes Prim. untersch. Greifstrategien vordefiniert
 \rightarrow Vorwärtsimulation des Greifprozesses

zB Kugel Zylinder
Box Kegel

Box-basierter Ansatz: Approx. durch Boxen

\rightarrow Greifhyp.- für Boxen erzeugen

\rightarrow Evaluation der Greifhyp. durch GraspIt!

Decomposition Algo: Punktewolke der Oberfl. in minimum volume Bounding Box packen

Allg. von Boxen zu Griffen: 6 Seitenflächen \Rightarrow 6 Greifhypothesen

- Griffpunkt: Mittelpunkt Seitenfläche
- Griffrichtung: entlang Normalen der Seitenfläche
- Handorientierung: u. mögl. orientiert an Kanten der Fläche
- verwirfe unmögl. Griffe direkt (blockierte/verdeckte/zu große Seitenflächen,...)

Greifplanung mit Superquadriken

Superquadrik = parametr. Flächen definieren Form des geom. Objekts

$$x(\eta, \omega) = \begin{pmatrix} a_1 \cos \epsilon_1(\eta) \cos \epsilon_2(\omega) \\ a_2 \cos \epsilon_1(\eta) \sin \epsilon_2(\omega) \\ a_3 \sin \epsilon_1(\eta) \end{pmatrix} \quad -\pi/2 \leq \eta \leq \pi/2 ; -\pi \leq \omega \leq \pi$$

a_1, a_2, a_3 = Größe der Superq. in Richtung der Raumachsen
 $\epsilon_1, \epsilon_2 \in [0, 1]$ bestimmen Schärfe des Kantenverlaufs

- ~ Objektoberfl. als Punktfolge
 - ~ ermittle Superquadrik, deren Oberfl. Punktfolge am besten darstellt
- Decomposition Tree:
- Teile Punktfolge, erstelle 2 Superq. usw. bis akzeptables Resultat

Grieffl. mit Medialen Achsen

approx. Objektform über enthaltene Kugeln mit max. Durchmesser

mediale Achse = Vereinigung der Mittelpunkte aller orth. Kugeln
~ beschreibt topolog. Skelett

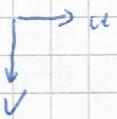
Generierung mögl. Griffen mit Heuristiken

- Einbeziehung von Objektsymmetrie
- mögl. Griffen geometrisch sinnvoll

Bildverarbeitung

Bildrepräsentation

Bildkoordinaten Einheit: Pixel



Graustufenbild Helligkeitswert $\in [0, 255]$ (idle) pro Pixel *

Monochrombild Diskrete Funktion $[0 \dots n-1] \times [0 \dots n-1] \rightarrow [0 \dots q]$
 $(u, v) \mapsto \text{img}(u, v)$

üblich $q = 255$

$n = 640, m = 480 \quad | \quad n = 1920, m = 1080 \quad | \quad n = 3840, m = 2160$

Auflösung = kleinstes erkennbares Detail im Bild

Farbbild

> RGB Farbraum

Ein Pixel = 3 Bytes ≈ 1 Byte je Wert $(u, v) \mapsto (r, g, b)^T$

> HSL Farbraum Hue, Saturation, Intensity / Value

Farbinfo getrennt von Helligkeit / Sättigung \rightarrow unempfindlich gegen Beleuchtungsänder.

RGB \rightarrow HSL

H undef. falls $R = G = B$

S undef. falls $R = G = B = 0$

$$H = \begin{cases} 0 & , B \leq G \\ 360 - \theta & , \text{sonst} \end{cases}$$

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

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

$$I = \frac{1}{3} (R + G + B)$$

$$V = \max(R, G, B)$$

HSL \rightarrow RGB

> I entspricht Linie durch $(0, 0, 0)$ (schwarz) und $(1, 1, 1)$ (weiß)

> S eines Farbpunkts auf der Sättigungsachse = 0, erhöht sich mit Abst. zur Helligkeitsachse

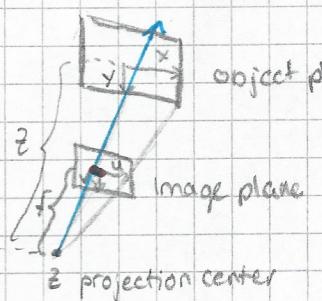
> H wird geändert durch rot. des Dreiecks um Helligkeitsachse

Speicher Pixel zeilenweise, linear abgetragen (von oben links nach unten rechts)

Kameramodell

Lichtkamera interner Parameter Brennweite f

Koordinatensysteme



Hauptachse: gerade durchs Projektionszentrum, rechtwinklig zur Bildebene

Hauptpunkt: Schnitt Hauptachse, Bildebene

Bildgenerierung Abb. Szenepunkt (x, y, z) auf Bildpunkt (u, v)

$$(z \text{ Strahlensatz}) \quad (u) = \frac{f}{z} (x) \quad (\text{Rückprojektion}) \quad (v) = \frac{f}{z} (y)$$

Kameraparameter

> **intrinsische**: kameraspezifisch, unabh. von Weltkoord.

> **extrinsische**: modellieren Transformation wks in Kamerakoord.

Erweitertes Kameramodell

unabh. Brennweiten f_x, f_y in u, v Richtung $f = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$

$$\text{Projektion jetzt: } \begin{pmatrix} u \\ v \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{Kalibriermatrix } K} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (c_x, c_y) = \text{Hauptpunkt}$$

$$K^{-1} = \begin{pmatrix} 1/f_x & 0 & -c_x/f_x \\ 0 & 1/f_y & -c_y/f_x \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Kalibriermatrix } K^{-1}$$

Parameter in K = intrinsische Params

extrins. Params = WKS / KWS Transformation durch Rotation R , Translation t

Weltpunkt x_w , Kamerakoord. x_c : $x_c = Rx_w + t$

Projektionsmatrix $P = K \cdot (R | t)$

Kamerakalibrierung : Bestimmung extr./intr. Params

$$\text{Löse } \begin{pmatrix} u \\ v \\ w \end{pmatrix} = P \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ nach Params } p_{11} \dots p_{12} \text{ in } P$$

Filteroperationen Filter = (Nachbarschaft, operation)

→ wird auf alle Bildpixel angewendet

linearer Filter: $f(x+y) = f(x) + f(y)$, $f(\alpha x) = \alpha f(x)$

Anwendung Filter: Filter auf Bild legen, Maskenwerte mit Pixelwerten multiplizieren, aufsummieren

Korrelation: Summe der Produkte an jedem Punkt

Faltung: = Korrelation, aber Filter erst um 180° drehen

Ränder: Constant / Wrap (Fortsætzen) / Mirror / Replicate

Tiefpassfilter Glättung, Rauschelimination

Medianfilter

Mittelwertfilter

Gauß-Filter

Kombinierte Operatoren
Laplacian of Gaussian

Hochpassfilter Kanten detektion

Prewitt

Sobel

Laplace

altern. Max-Filter
Min-Filter

Medianfilter nicht linear

(1) Kernelgröße wählen (2) Grauwerte im Kernelbereich sortieren

(3) mittleren Grauwert ermitteln → neuer Wert

④ Salz- und Pfefferrauschen → bewahrt Kanten, entfernt Rauschen

⑤ Gaußsches Rauschen

Mittelwertfilter Durchschnitt aus Pixel + Nachbarschaft

$$\text{zB } F = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Gauß-Filter def. durch 2D-Gauß-Fkt.

$$\text{Ortsbereich: } f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{Freq. ber.: } F(u,v) = e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$$\text{zB } F = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \quad ; \text{ je größer } \sigma \text{ umso stärker die Glättung}$$

σ Größe $n \times n$ beeinflusst Approx.-güte

TIEFPASSFILTER

> Prewitt-X Filter: Detektion vertikaler (|) Kanten $P_x = \frac{\partial f(x,y)}{\partial x}$

$$\text{approx. durch } p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

> Prewitt-Y Filter: Detektion horizontaler (-) Kanten $P_y = \frac{\partial f(x,y)}{\partial y}$

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

Kombination zur Bestimmung des Gradientenbetrags $H \approx \sqrt{P_x^2 + P_y^2}$
→ danach: Schwellenwertfilterung

HOCHPASSFILTER

Sobel-X approx. durch

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

Sobel-Y approx. durch

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

$$H \approx \sqrt{s_x^2 + s_y^2} \quad \text{danach Schwellenwertfilterung}$$

Laplace rotat. invariant

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

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

dünnerne Kanten als Prewitt / Sobel

Laplacian of Gaussian (LOG)

Gauß-Filter \rightarrow Laplace-Operator

$$\text{LOG}(f(x,y)) = \Delta(f(x,y) * g(x,y))$$

\leftarrow Gauß-Filterfunktion

Approx.:

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

Segmentierung = Aufteilung eines Bildes in aussagekräftige Segmente

Schwellwertfilterung Konvertierung Grauwertbild \rightarrow binär Bild

Vergleiche Intensität von Pixeln (u,v) mit Schwellwert T

$$\text{img}^1(u,v) = \begin{cases} 255, & \text{img}(u,v) > T \\ 0, & \text{sonst} \end{cases}$$

\sim Farbsegmentierung Bsp. HSV-Farbraum

$$\text{img}^1(u,v) = \begin{cases} 255, & H_{\max} \geq \text{img}_H(u,v) \geq H_{\min} \\ & S_{\max} \geq \text{img}_S(u,v) \geq S_{\min} \\ & V_{\max} \geq \text{img}_V(u,v) \geq V_{\min} \\ 0, & \text{sonst} \end{cases}$$

Problem: wechselnde Lichtbed., Schattenwürfe

Morphologische Operatoren Nachbearbeitung binärer Bilder

\leftarrow Vereinigung Strukturelement

Dilatation $g(m,n) = \bigcup_{(m_k, n_k) \in S} b(m+m_k, n+n_k)$

Ergebnis an Koord des Strukturelement.

vergrößert Pixel zu Bereichen

\leftarrow Schnitt

Erosion $g(m,n) = \bigcap_{(m_k, n_k) \in S} b(m+m_k, n+n_k)$

entfernt einzelne Pixel, schwach zsm hängende Pixelgruppen

Opening Erosion \rightarrow Dilatation, entfernt dünne Stege, kleine aufgelagerte Objekte

Closing Dilatation \rightarrow Erosion, Überbrückung kleiner Distanzen, Schließen innerer Löcher

canny-kantendetektor berechnet binäre Antwort
 (1) Gauß-Filter
 (2) Gradienten M berechnen (Prewitt oder Sobel)
 a) Richtung $\phi = \arctan\left(\frac{g_y}{g_x}\right)$

\emptyset = keine Kante
 255 = Kante

- b) Einteilung der Richtung in 4 Quadranten
 (1) $[-67,5^\circ, -22,5^\circ]$ (3) $[22,5^\circ, 67,5^\circ]$
 (2) $[-22,5^\circ, 22,5^\circ]$ (4) $[67,5^\circ, 90^\circ]$

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

- (3) Non-Maximum Supression
 (4) Hysterese-Schwellwertverfahren

Non-Maximum Supression $\phi(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$ = kanten normale

(1) Def. 3×3 Region (p_1, \dots, p_9) um jeden Punkt (x,y) in $\phi(x,y)$

(2) Bestimme Kantenrichtung nahe $\phi(x,y)$

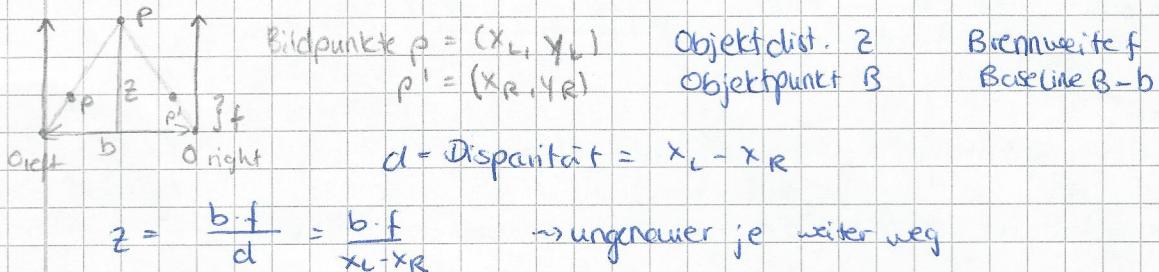
(3) Wert im Zentrum von M kleiner als einer der beiden Nachbarn in Kantenrichtung $\Rightarrow M=0$

Hysterese-Schwellwertverfahren

- (1) 2 Schwellwert low/high
 (2) Beitrag des Grad. für einen Pixel > Schwellwert high \rightarrow Teil der Kante
 (3) $<$ low \rightarrow abgelehnt als Kante
 (4) von akzeptierten Pixeln ~~Kanten~~ Nachbarn verfolgen

Tiefenkameras

Stereo Vision



Visual Servoing

benutze visuelle Eingabedaten zur Robotsteuerung

- > Kamera-in-Händ : Kamera an Robo (Manipulator)
 > Externe Kamera

Positionsbasiert Zielpose x_g vorgegeben

- (1) 3D Lage Schätzung: x_c aktuelle Pos. aus Bild extrahieren
 (2) Regelvorgabe kartesisch $\Delta x = x_g - x_c$
 (3) Kartes. Regler führt Δx aus
 (4) Stoppe wenn Δx kleiner Schwellenwert

(+) einfache Regler

(-) 3D Rekonstr. aufwändig

Bildbasiert

Geschw. vergaben direkt aus aktuelle + gewünschte
 Stellung der Bildmerkmale
 \downarrow aktuelle Pos.

Fehler $e(t) = s(t) - s^*$ \leftarrow gew. Zielpos.

(+) einfache Merkmalsextraktion

(-) komplexere Regelung

Interaktion Matrix Beziehung zw. Bildpunkt $s = (u, v)$ und 3D Punkt (x, y, z)

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

f^2+uv^2
 f
 uv
 f
 u (33)

Schärfedistanz e
Kamerabewegung $v_c \Rightarrow e = L \cdot v_c, \dot{e} = -\lambda e$
 $\Rightarrow v_c = -L^+ e$ Pseudoinverse von L
Regelungseingabe

Punktwolken diskrete Menge von 3D-Punkten

$$P = \{(x, c) | x \in \mathbb{R}^3, c \in [0, 255]^3 \subset \mathbb{N}^3\}$$

$$\begin{array}{ll} x = (x, y, z) & \text{Ort} \\ c = (r, g, b) & \text{Farbe} \end{array}$$

Registrierung: Zusammenführen von Punktwolken, welche gleiches Objekt beschreiben
 \rightarrow überführen in ein Koord.-sys.

Iterative Closest Point (ICP) Registrierung A, B

(3D) für jede Iteration k:

- ✓ Punkte a_i aus A: suche $b_j \in B$, der a_i am nächsten
- Berechne Transformation T_k , sodass D_k minimal
- (2B) $D_k = \sum \|a_i - T_k \cdot b_j\|^2$ \nwarrow kombiniert Translat. + Rot.
- wende T_k auf alle Punkte aus B an
- Abbruch: Schwellwert für $D_{k-1} - D_k$
max Iterationszahl

RANSAC

(1) wähle zufällig min. Anzahl Punkte aus, die zur Modellparam.berechnung nötig
 $\rightarrow 2$ für Linien 2D, 3 für Ebenen 3D

(2) schätze Modell

(3) bewerte Modell \rightarrow berechne Inliers (Teilmenge der Datenpunkte mit Abstand zum Modell kleiner als Schwellwert)

(4) wdh. bis meiste Inlier vor. Anz. Iterationen: $n = \frac{\log(1 - p(\text{success}))}{\log(1 - w^n)}$

$$p(\text{success}) = \frac{n \text{ Punkte}}{w \text{ Inlier}} = \frac{1}{1 - (1 - w)^n}$$

SLAM Simultaneous Localization and Mapping \Leftarrow Karte ableiten aus Menge von Pos. Schätzung Roboter bei geg. Karte

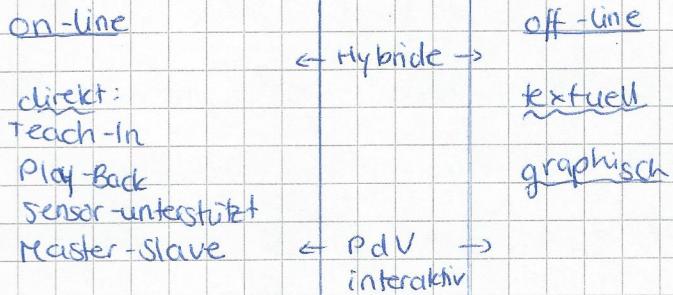
Szenenfeatures als Orientierungspunkte (Ecken, Kanten, Texturen, ...)

In jedem Frame:

- (1) Vorhersage, wie viel sich Robo bewegt hat
- (2) neue Orientierungspunkte aufnehmen
- (3) interne Repr. aktualisieren

Programmierung

Verfahren



on-line = direkt am Roboter / der Steuerung = direkte Progr.
off-line = ohne Robo = indirekte Progr.

Abstraktionsgrad

Explizit (roboterorientiert) = imperativ
Bewegung + Greiferbefehle direkt in Progr. Sprache eingebunden "wie"

Implizit (aufgaborientiert) = deklarativ
Aufgabe wird beschrieben, z.B. durch Zustände "Was"

Direkte Programmierung

Teach-In

- > Anfahren markanter Punkte der Bahn \rightarrow Bahn = Folge von Zwischenpunkten
- > speichern der Gelenkwerte
- > nachträglich: Geschw., Beschl., ...

Play-Back Robo auf Zero-Force-Control

- > Bediener fährt mit Robo Bahn ab
- > Speichern der Gelenkwerte
- ∅ Sicherheitsrisiko, hoher Speicherbedarf, schlechte Korrekturmögl.

Master-Slave Master = kin. Modell des großen Slave

- > Master geführt von Bediener
- > Bewegung auf Slave übertragen \rightarrow Slave $\hat{=}$ Kraftverstärker
- > Bewegungen synchron
- ∅ schwere Rabos progr. bar ∅ teuer (2 Rabos)

Sensor unterstützt

Manuell: Bediener führt Progr. griffel, Bewegung mit Sensoren erfassen
↳ inverse Kinematik \rightarrow Bahn als Folge der Gelenkwinkel

Automatisch: Übergabe Start-, Zielpunkt; sensorische Abtastung der Sollkontur

∅ Fehler bei Erfassung der Bahn

aug. dir. Progr.: (∅) schnell (^{für} einfache Traj.), sofort anwendbar,
Bediener braucht keine Progr.-kenntnisse, kein Umweltmodell

∅ hoher Aufwand für komplexe Traj., nur mit am Robo spezifisch für Robo,
Sicherheitsrisiko, keine Adaption an Umwelt

Textuelle Verfahren ~ mit Robotersteuerprogramm
① unabh. Robo, strukturierte Logik, komplexe Progr. mögl.
② Bediener braucht Progr. Kenntnisse

Graphische Verfahren

> Virtuelles Teach-In: Robomanipulation in 3D Visualisierung

Statecharts: Hierarchisch, Interleavetransitions, Zustandsaktionsphasen (Entry, Exit, throughout)

Erweiterung: ermögliche Datenfluss (Eingabe, Parameter, Ausgabe)

Symbolische Planung

Plan = Sequenz parametr. Aktionen zum Erreichen eines def. Ziels
symbolisch = Weltzustand durch Boolesche Prädikate repr.

STRIPS Sprache zur Planungsbeschreibung

- Zustände, Aktionen, Ziele

Zustand der Welt:

- > Konjunktion positiver aussagenlogischer Literale
- > Einschränkungen: endl. # Literale, keine Variablen (neg. Literale / Funktionen)
- jeder Zustand muss vollst. beschrieben werden können + bekannt sein
(nicht vorkommende Literale = negative Literale)

Aktion: Tripel (Deklaration, Vorbed., Effekte)

> Dekl. = Name + Parameterliste

> Vorbed.: Konj. von Literalen → müssen wahr sein um Operator/Aktion
 $\forall a$ anzuwenden zu können

> Effekte = Auswirkung der Aktion auf Weltzustand
 $\exists z$

Aktion A ausführbar in allen Zust. z, falls z Va erfüllt

Ergebnis: entferne alle negativen Literale des Ea aus z
füge alle pos. Literale des Ea zu z

Programmieren durch Vormachen

Ziel: Intuitive Roboterprogr. ohne Expertenwissen

Drei-Phasen-Modell: Perzeption → Kognition → Aktion

Komplexitätsreduktion im Gegensatz zum Ausprobieren aller Möglichkeiten
↳ trainieren von Robotern

Hauptaufgaben

Wer soll imitiert werden? Lehrerauswahl

Wann soll imitiert werden? Bewegung segmentieren, Start, Ende, aktueller Kontext
der Bewegung

Was soll imitiert werden? Welche Aspekte der Demo interessant?
Irrelevante Eigenschaften filtern

Wie soll imitiert werden? Wie soll gelerntes Verhalten ausgeführt werden

Perception = Arten der Demonstration

- Schritte: (1) Record Mapping: Aktionen des Lehrers aufnehmen
 (2) Embodiment Mapping: Aktion auf Robo abbilden

Teleoperation

- > Lehrer steuert Robo direkt
- > Robosensoren nehmen Demo auf
- keine Nachbearb.
nötig

Aufgenommene Daten direkt → von Robo nutzbar
 Daten nur indirekt verfügbar →

Shadowing

- > Lehrer + Schüler gleches Embodiment
- > Nachahmung des Lehrers durch Beobachtung
- > Aufnahmeder Aktion mit eigenen Sensoren → indirekte Aufnahme

Sensors on Teacher

- > Bewegungen des Lehrers direkt aufnehmen + direkter Zugriff
 → Abbildung Mensch nach Robo nötig

Erfassung menschl. Bewegung:

Marker - basiert: z.B. VICON

Marker an anatom. Landmarken des Menschen, Aufnahme

Marker-frei: Rekonstruktion aus Kamera-Daten

mechanisch: direkte Messung von Gelenkwinkeln

Imitation / externe Beobachtung

- > Beob. durch Robosensoren
- > komplexe Bildverarbeitung
- > Abb. Mensch nach Robo nötig

Kognition lernen einer Fähigkeit

Zwei Ebenen:

- I) Trajektorie: Zuordnung zw. Sensor- und Motorinformationen
- II) symbolisch: Zerlegung in Folge von Aktionen

ziel: Zuordnung Weltzustand, Aktion

→ Segmentierte Demonstration: Bewegungssegmentierung od. Aufgabensegm.

Demonstrationsverarbeitung:

Batch Lernen: Aktion lernen, wenn alle Aktionen / Wdh. aufgenommen

Incrementelles Lernen: Actionsrepr. nach jeder Aktion / Wdh. gelernt / aktualisiert

Embodiment Mapping

direkt benutzt abgeleitet

Teleoperation

Sensors on Teacher

Recording Mapping
direct
indirect
abgeleitet

Shadowing

Imitation

Bewegungen entpr.
körper des Robos
= Demonstration

Möglichkeit auf
Robo notwendig
= Imitation