Robotik – Autonomes Greifen

R. Haschke, J. Steil

letzte Änderung 23. Januar 2018

Einleitung

Das vorliegende Skript ist als Begleitung zur Vorlesung “Autonomes Greifen” an der Technischen Fakultät der Universität Bielefeld gedacht. Das Skript erhebt weder Anspruch auf Vollständigkeit noch auf absolute Korrektheit. Korrekturen und andere Verbesserungsvorschläge nehme ich gerne entgegen (rhaschke@techfak.uni-bielefeld.de).

Bielefeld, 23. Januar 2018 Robert Haschke

Inhaltsverzeichnis

1 Screw-Theorie

1.1 Die Lage eines festen Körpers

Definition 1.1 Ein fester Körper (rigid body) ist eine Menge von Punkten, deren relative Lage sich nicht ändern kann. Die Lage all dieser Punkte ist daher relativ zu einem körperfesten Koordinatensystems (KS) zeitlich konstant.

Die Lage eines festen Körpers (und damit all seiner Punkte) im dreidimensionalen Raum ist durch den Ursprung und die Orientierung seines körperfesten KS ($B_{p,u,v,w}$) relativ zu einem festen Welt-KS ($A_{0,x,y,z}$) bestimmt (Abb. 1).

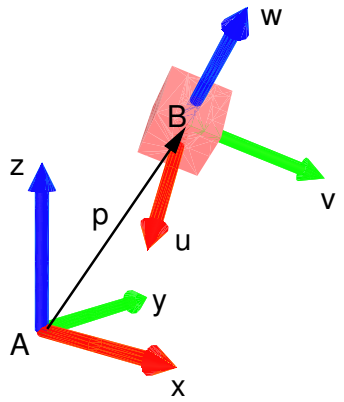


Abbildung 1: Körper mit körperfesten KS B, relativ zum festen Welt-KS A

Die homogene Transformation, die das Welt-KS A in das körperfeste KS B überführt, ist gegeben durch:

$$T_{ab} = \begin{pmatrix} xu & xv & xw & p_x \\ yu & yv & yw & p_y \\ zu & zv & zw & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{ab} & \vec{p}_{ab} \\ 0 & 1 \end{pmatrix}$$

Eine solche homogene Transformation kann auf zwei Arten interpretiert werden. Zum einen können wir $T_{ab} : B \to A$ als Koordinaten-Transformation von einem KS B in ein anderes KS A auffassen (Gl. 1.1). Dabei werden die homogenen Koordinaten eines Punktes Q bzgl. des (körperfesten) KS B ($\bar{q}_{u,v,w} = [q_u, q_v, q_w, 1]^t$) in die zugehörigen Koordinaten desselben Punktes bzgl. des Welt-KS A ($\bar{q}_{x,y,z} = [q_x, q_y, q_z, 1]^t$) transformiert:

$$\bar{q}_{x,y,z} = T_{ab}\, \bar{q}_{u,v,w} \qquad \text{bzw.} \qquad \vec{q}_a = R_{ab}\, \vec{q}_b + \vec{p}_{ab}\,. \tag{1.1}$$

Zum anderen kann $T_{ab} : A \to A$ auch als *Bewegung* aufgefasst werden, die das Koordinatenkreuz A in das Koordinatenkreuz B überführt, wobei *beides mal* auf das KS A Bezug genommen wird und sich durch die Bewegung ein Punkt Q verändert:

$$\bar{q}'_a = T_{ab}\bar{q}_a$$

Bemerkung: Die homogenen Koordinaten von Richtungsvektoren werden anstatt mit einer Eins mit einer Null ergänzt, so dass bei der Transformation der translatorische Anteil $\vec{p}$ ignoriert wird. Im Gegensatz zu Punktvektoren haben Richtungsvektoren keine Verankerung im Raum, sondern können beliebig verschoben werden.

Die perspektivische Transformation und die Skalierung würden die Geometrie von festen Körpern verändern, so dass sie keine gültigen Transformationen für Festkörper darstellen.

1.2 Repräsentation von Rotationen

Definition 1.2 Die Menge $SO(3) = \{R \in \mathbb{R}^{3\times 3} \,|\, RR^t = \mathbf{1},\ \det R = 1\}$ heißt spezielle orthogonale Gruppe (bzgl. der Matrizenmultiplikation). Sie enthält die Menge der Rotationsmatrizen, die die Rechtshändigkeit von KS erhalten.

Im folgenden suchen wir eine geeignete Repräsentation für solche Matrizen. Offenbar genügen drei freie Parameter (Freiheitsgrade), um Rotationsmatrizen zu beschreiben.

1.2.1 Basis-Rotationen

Jede Rotation kann durch Aneinanderreihung von Elementar-Rotationen um die x-, y- oder z-Achse erzielt werden:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \quad R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

1.2.2 Rotation um eine beliebige Achse

Um die Rotationsmatrix für die Drehung um eine beliebige Achse

$$\vec{\omega} = \begin{pmatrix} \cos\phi\cos\psi \\ \sin\phi\cos\psi \\ -\sin\psi \end{pmatrix} \tag{1.2}$$

drehen wir diesen Vektor zunächst um $-\phi$ um die z-Achse (dann liegt er in der x-y-Ebene) und dann um die y-Achse um den Betrag von $-\psi$ (damit liegt der Vektor auf der x-Achse). Hier können wir die Rotation um den Winkel θ durchführen und anschließend die x-Achse wieder zurück transformieren:

$$R_{\vec{\omega}}(\theta) = R_z(\phi)R_y(\psi)\ R_x(\theta)\ R_y^{-1}(\psi)R_z^{-1}(\phi) \tag{1.3}$$

Gegeben einen Einheitsvektor $\vec{\omega} = (\omega_1, \omega_2, \omega_3)$ und die Bezeichnungen $c_\theta = \cos\theta$, $s_\theta = \sin\theta$, $\eta_\theta = 1 - \cos\theta$, dann gilt für $R_{\vec{\omega}}(\theta)$:

$$\begin{aligned} R_{\vec{\omega}}(\theta) &= \begin{pmatrix} \omega_1^2\eta_\theta + c_\theta & \omega_1\omega_2\eta_\theta - \omega_3 s_\theta & \omega_1\omega_3\eta_\theta + \omega_2 s_\theta \\ \omega_1\omega_2\eta_\theta + \omega_3 s_\theta & \omega_2^2\eta_\theta + c_\theta & \omega_2\omega_3\eta_\theta - \omega_1 s_\theta \\ \omega_1\omega_3\eta_\theta - \omega_2 s_\theta & \omega_2\omega_3\eta_\theta + \omega_1 s_\theta & \omega_3^2\eta_\theta + c_\theta \end{pmatrix} \\ &= c_\theta \mathbf{1} + s_\theta\hat{\omega} + \eta_\theta\vec{\omega}\vec{\omega}^t \qquad \text{(Rodrigues-Formel)} \end{aligned}$$

Satz 1.3 (Euler) Jede Rotationsmatrix $R \in SO(3)$ ist äquivalent zu einer Rotation um eine feste Achse $\vec{\omega} \in \mathbb{R}^3$, $\|\vec{\omega}\| = 1$ und einen Drehwinkel $\theta \in [0, 2\pi)$.

Beweis:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \stackrel{!}{=} R_{\vec{\omega}}(\theta) = \begin{pmatrix} \omega_1^2\eta_\theta + c_\theta & \omega_1\omega_2\eta_\theta - \omega_3 s_\theta & \omega_1\omega_3\eta_\theta + \omega_2 s_\theta \\ \omega_1\omega_2\eta_\theta + \omega_3 s_\theta & \omega_2^2\eta_\theta + c_\theta & \omega_2\omega_3\eta_\theta - \omega_1 s_\theta \\ \omega_1\omega_3\eta_\theta - \omega_2 s_\theta & \omega_2\omega_3\eta_\theta + \omega_1 s_\theta & \omega_3^2\eta_\theta + c_\theta \end{pmatrix}$$

Für die Spur der beiden Matrizen gilt:

$$\begin{aligned} \operatorname{sp} R = r_{11} + r_{22} + r_{33} &\stackrel{!}{=} 3\cos\theta + (1-\cos\theta)\sum \omega_i^2 = 1 + 2\cos\theta \\ \Rightarrow \quad \theta &= \cos^{-1}\left(\frac{\operatorname{sp} R - 1}{2}\right) \in [0, \pi]\,. \end{aligned}$$

Diese Gleichung kann nach θ aufgelöst werden, weil die Eigenwerte λ_i von R den Betrag Eins haben und daher gilt:

$$-1 \leq \operatorname{sp} R = \sum \lambda_i \leq 3\,.$$

Um die Rotationsachse zu bestimmen, nutzen wir die restlichen Matrixeinträge:

$$\left.\begin{array}{rcl} r_{32} - r_{23} &=& 2\omega_1 s_\theta \\ r_{13} - r_{31} &=& 2\omega_2 s_\theta \\ r_{21} - r_{12} &=& 2\omega_3 s_\theta \end{array}\right\} \quad \stackrel{\theta\neq 0}{\Rightarrow} \quad \vec{\omega} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \cdot \frac{1}{2s_\theta} \tag{1.4}$$

Falls $R = \mathbf{1}$ ist, folgt $\operatorname{sp} R = 3$ und daher $\theta = 0$. In diesem Fall ist $\vec{\omega}$ beliebig, da $R_{\vec{\omega}}(0) = \mathbf{1}$. □

Bemerkung: Der Winkel θ ist nicht eindeutig bestimmt, da $\theta \pm 2\pi n$ und $-\theta \pm 2\pi n$ denselben cosinus besitzen. Wählt man $-\theta + 2\pi \in [\pi, 2\pi]$, muss die Drehachse $\vec{\omega}$ umgedreht werden. D.h. zu $R \neq \mathbf{1}$ gibt es zwei verschiedene Paare $(\vec{\omega}, \theta)$ mit $\theta \in [0, 2\pi)$, so dass $R = R_{\vec{\omega}}(\theta)$. Solche Singularitäten sind typisch für jede 3-parametrige Repräsentationen von Rotationen.

Die Singularität bei $R = \mathbf{1}$ kann vermieden werden, wenn man die Trennung von normierter Drehachse $\vec{\omega}$ und Drehwinkel θ aufhebt und stattdessen beliebige $\vec{\omega} \in \mathbb{R}^3$ betrachtet für deren Norm gilt: $0 \leq \|\vec{\omega}\| \leq \pi$. Allerdings bleibt die Singularität bei $\|\vec{\omega}\| = \pi$ (Drehung um 180°) erhalten.

1.2.3 Euler Winkel

Eine Rotation R_{ab} zwischen den KS A und B kann auch durch Aneinanderreihung von anderen Standard-Rotationen erreicht werden, z.B.

1. Rotiere um den Winkel ψ um die x-Achse (yaw)
2. Rotiere um den Winkel θ um die alte y-Achse (pitch)
3. Rotiere um den Winkel ϕ um die alte z-Achse (roll)

$$R_{ab} = R_{z,\phi} \cdot R_{y,\theta} \cdot R_{x,\psi}$$

- $(\phi, \theta, \psi) \mapsto R$ ist surjektiv, d.h. für jedes R gibt es geeignete Euler-Winkel.
- Singularitäten: $\theta = \pm\pi/2$.

Je nach Auswahl der Rotationsachsen für die Standard-Rotationen erhält man unterschiedliche Sets von Euler-Winkeln. Die Angabe der Rotationsachsen (und der Anwendungsreihenfolge – weltfest oder körperfest) ist daher zwingend notwendig, wenn man über Euler-Winkel spricht. Insgesamt gibt es 24 unterschiedliche Euler-Winkel-Sets, da direkt aufeinander folgende Rotationen unterschiedliche Rotationsachsen verwenden müssen.

1.2.4 Quaternionen

Quaternionen sind 4-dimensionale Vektoren

$$\begin{aligned} Q &= (q_0, \vec{q}) && q_0 \in \mathbb{R}, \quad \vec{q} \in \mathbb{R}^3 \\ &= q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k}, && q_i \in \mathbb{R}\,, \end{aligned}$$

deren Addition komponentenweise erfolgt. Man kann sich 1, $\hat{i}$, $\hat{j}$ und $\hat{k}$ als Einheitsvektoren dieses 4-dim. Raumes vorstellen. Die distributive und assoziative Multiplikation ist entsprechend folgender Regeln definiert:

- $\hat{i}^2 = \hat{j}^2 = \hat{k}^2 = \hat{i} \cdot \hat{j} \cdot \hat{k} = -1$
- $\hat{i} \cdot \hat{j} = -\hat{j} \cdot \hat{i} = \hat{k}, \qquad \hat{j} \cdot \hat{k} = -\hat{k} \cdot \hat{j} = \hat{i}, \qquad \hat{k} \cdot \hat{i} = -\hat{i} \cdot \hat{k} = \hat{j}$ (zyklisch)

Weiterhin gilt:

- $Q^* = (q_0, -\vec{q})$ (konjugiertes Quaternion zu $Q = (q_0, \vec{q})$)
- $\|Q\|^2 = QQ^* = Q^*Q = \sum q_i^2$ (Betrag)
- $Q^{-1} = \frac{Q^*}{\|Q\|^2}$ (inverses Quaternion)
- $Q \cdot P = (q_0, \vec{q}) \cdot (p_0, \vec{p}) = (q_0p_0 - \vec{q} \cdot \vec{p}, q_0\vec{p} + p_0\vec{q} + \vec{q} \times \vec{p})$ (Multiplikation)
- $Q \cdot (0, \vec{p}) \cdot Q^* = (0, R_{\vec{\omega}}(\theta)\vec{p})$

Eine Rotation $R_{\vec{\omega}}(\theta)$ wird durch das Einheitsquaternion $Q = (\cos\frac{1}{2}\theta, \vec{\omega}\sin\frac{1}{2}\theta)$ mit Betrag $\|Q_{ab}\| = 1$ repräsentiert. Dann gilt für $R_{ab}R_{bc} = R_{ac}$ auch $Q_{ab}Q_{bc} = Q_{ac}$. Die Quaternion-Darstellung erlaubt eine besonders effiziente Berechnung von Verkettungen von Rotationen. Übung

Die Darstellung durch Einheitsquaternionen (mit vier Komponenten) ist frei von Singularitäten. Die Einheitsquaternionen beschreiben die Oberfläche einer 4d-Kugel mit Radius 1. Ähnlich wie jeder 3d-Punkt auf der Erdoberläche (idealisiert als Kugel mit Radius 1) eine 2d-Orientierung beschreibt (Längen- und Breitengrade sind quasi das äquivalent zu den Euler-Winkeln), beschreibt ein 4d-Punkt auf der 4d-Kugel nun eine 3d-Orientierung.

Beachten Sie, dass Q und $-Q$ zwar unterschiedliche Quaternionen sind, aber dieselbe Rotation beschreiben, denn $Q(\theta{+}360°) = -Q$ und erst $Q(\theta{+}720°) = Q$. Daher ist es bei der Quaternion-Interpolation (Spherical Linear Interpolation, SLERP) wichtig, den "kurzen" Weg auf der Kugeloberfläche zu nehmen, d.h. dass beide Quaternionen

auf derselben (z.B. positiven) Hemispere liegen. Die Interpolation zwischen Q_1 und Q_2 erfolgt dann folgendermaßen (Slerp):

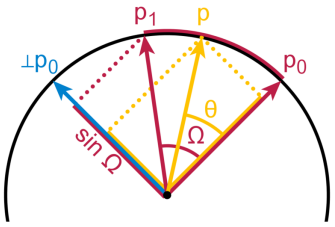
$$\begin{aligned} Q(Q_1, Q_2, t) &= Q_1(Q_1^{-1}Q_2)^t \\ &= Q_2 \cdot (Q_2^{-1}Q_1)^{1-t} \\ &= (Q_1Q_2^{-1})^{1-t} \cdot Q_2 \\ &= (Q_2Q_1^{-1})^t \cdot Q_1 \end{aligned}$$

wobei $Q^t = (\cos\frac{1}{2}\theta t, \vec{\omega}\sin\frac{1}{2}\theta t)$ mit $t \in [0,1]$.

Der Slerp ist auch unabhängig von Quaternionen definiert und überführt beliebige Vektoren p_o, $p_1 \in \mathbb{R}^n$ entlang einem Großkreis ineinander:

$$\text{slerp}(p_0, p_1, t) = \frac{\sin((1-t)\Omega)}{\sin\Omega}p_0 + \frac{\sin(t\Omega)}{\sin\Omega}p_1$$

$$\Omega = \arccos(p_o \cdot p_1) \qquad \text{Drehwinkel}$$



1.2.5 Exponentielle Koordinaten

Notation: Das Kreuzprodukt $\vec{a} \times \vec{b}$ kann auch als lineare Abbildung $\hat{a} \in \mathbb{R}^{3\times3}$ aufgefasst werden, die auf den Vektor $\vec{b}$ angewandt wird:

$$\vec{a} \times \vec{b} = \hat{a} \cdot \vec{b} \qquad \text{wobei} \qquad \hat{a} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} = -\hat{a}^t\,.$$

Es gelten die folgenden Identitäten: Übung

$$\hat{\omega} \cdot \vec{\omega} = \vec{\omega} \times \vec{\omega} = 0 \qquad \text{und} \qquad \hat{\omega}^2 = \vec{\omega} \cdot \vec{\omega}^t - \|\vec{\omega}\|^2\mathbf{1}$$

Wir betrachten eine Rotation um eine beliebige Drehachse $\vec{\omega}$ ($\|\vec{\omega}\| = 1$) und den

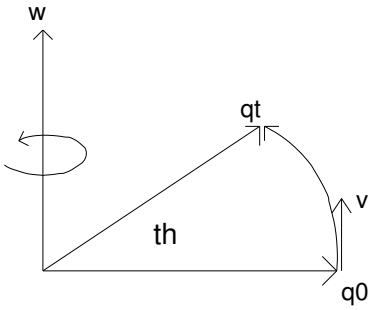


Abbildung 2: Drehung um die Drehachse $\vec{\omega}$ und den Winkel θ.

Winkel θ (s. Abb. 2). Wir motivieren die weiteren Berechnungen basierend auf der Geschwindigkeit eines Punktes q als Teil des rotierenden Körpers. Wenn die Rotation mit konstanter Einheits-Geschwindigkeit um die Achse $\vec{\omega}$ erfolgt lässt sich die Geschwindigkeit schreiben als:

$$\dot{\vec{q}}(t) = \vec{\omega} \times \vec{q}(t) = \hat{\omega}\,\vec{q}(t)\,.$$

Integration dieser linearen Differentialgleichung ergibt:

$$\vec{q}(\theta) = \int_0^\theta \hat{\omega} q(t) dt = e^{\hat{\omega}\theta}\vec{q}(0) = R(\hat{\omega}, \theta)\vec{q}(0)$$

wobei $e^{\hat{\omega}\theta}$ die Exponentialfunktion einer Matrix ist:

$$\begin{aligned} e^{\hat{\omega}\theta} &= \mathbf{1} + \hat{\omega}\theta + \frac{(\hat{\omega}\theta)^2}{2!} + \frac{(\hat{\omega}\theta)^3}{3!} + \dots \\ &= \mathbf{1} + \hat{\omega}\sin\theta + \hat{\omega}^2(1 - \cos\theta) \end{aligned}$$

Satz 1.4 Die Matrix $R(\hat{\omega}, \theta) = e^{\hat{\omega}\theta}$ ist eine Rotationsmatrix.

Definition 1.5 Die Menge der antisymmetrischen Matrizen (skew-symmetric) bezeichnen wir mit $so(3) = \{S \in \mathbb{R}^{3\times 3} \mid S^t = -S\}$.

Die Abbildung $\vec{\omega} \mapsto \hat{\omega}$ bildet Vektoren $\vec{\omega} \in \mathbb{R}^3$ bijektiv auf antisymmetrische Matrizen $\hat{\omega} \in so(3)$. Die Matrix-Exponentialfunktion $\exp : so(3) \to SO(3)$ bildet antisymmetrische Matrizen auf Rotationsmatrizen ab. Frage: Ist diese Abbildung auch surjektiv, d.h. gibt es zu jeder Rotationsmatrix $R \in SO(3)$ eine antisymmetrische Matrix $\hat{\omega}\theta$, so dass $\mathrm{e}^{\hat{\omega}\theta} = R$?

1.3 Twist-Koordinaten homogener Transformationen

Eine allgemeine Bewegung setzt sich aus einer Translation und einer Rotation zusammen. Für eine allgemeine Rotation, bei der die Rotationsachse nicht durch den Ursprung verläuft, sondern durch einen beliebigen Punkt $\vec{p}$, lässt sich die homogene Transformation wie folgt berechnen:

$$T_{\vec{\omega},p} = \begin{pmatrix} \mathbf{1} & +\vec{p} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_{\vec{\omega}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & -\vec{p} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{\vec{\omega}} & (\mathbf{1} - R_{\vec{\omega}})\vec{p} \\ 0 & 1 \end{pmatrix}$$

Satz 1.6 (Chasles) Jede homogene Transformation $T \in SE(3)$ kann aus einer Translation und einer Rotation um dieselbe Achse (die Screw-Achse $\vec{\omega}$) zusammengesetzt werden.

Beweis:

$$T = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \delta\vec{\omega} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_{\vec{\omega}} & (\mathbf{1} - R_{\vec{\omega}})\vec{p} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{\vec{\omega}} & (\mathbf{1} - R_{\vec{\omega}})\vec{p} + \delta\vec{\omega} \\ 0 & 1 \end{pmatrix}$$

$\vec{\omega}$: kann wie oben aus $R = R_{\vec{\omega}}$ bestimmt werden.
δ: $\vec{\omega} \cdot ((\mathbf{1} - R_{\vec{\omega}}\vec{p} + \delta\vec{\omega}) = \vec{\omega} \cdot \vec{t} = \delta\|\vec{\omega}\|^2 = \delta$
$\vec{p}$: Löse $(\mathbf{1} - R_{\vec{\omega}})\vec{p} = \vec{t} - \delta\vec{\omega} \quad \Rightarrow \quad \vec{p} = \vec{p}_0 + \lambda\vec{\omega}$.
□

Wir wollen die für Rotationen eingeführte Exponentialdarstellung nun auf homogene Transformationen verallgemeinern. Daher definieren wir:

Definition 1.7 Die Menge $SE(3) = \left\{ \left(\begin{smallmatrix} R & \vec{p} \\ 0 & 1 \end{smallmatrix} \right) \mid \vec{p} \in \mathbb{R}^3, R \in SO(3) \right\}$ heißt spezielle euklidische Gruppe.

In Korrespondenz zur Definition von $so(3)$ definieren wir:

Definition 1.8 Ein Element $\hat{\xi}$ der Menge

$$se(3) = \left\{ \begin{bmatrix} \hat{\omega} & \vec{v} \\ 0 & 0 \end{bmatrix} \mid \vec{v} \in \mathbb{R}^3, \quad \hat{\omega} \in so(3) \right\}$$

heißt *Twist* und die Darstellung von $\hat{\xi}$ als 6-dimensionaler Vektor $\vec{\xi} = (\vec{v}, \vec{\omega})$ nennen wir *Twist-Koordinaten*.

Im folgenden zeigen wir, dass die Exponentialfunktion die Menge $se(3)$ der Twists surjektiv auf die Menge $SE(3)$ der homogenen Transformationen abbildet – genauso wie antisymmetrische Matrizen auf Rotationsmatrizen abgebildet werden. Zunächst ist aber zu zeigen, dass $\mathrm{e}^{\hat{\xi}}$ überhaupt eine homogene Transformation erzeugt.

Satz 1.9 Für jedes $\hat{\xi} \in se(3)$ liefert $\mathrm{e}^{\hat{\xi}}$ eine homogene Transformation aus $SE(3)$.

Beweis:

Fall 1) $\vec{\omega} = 0$ (reine Translation)

$$\hat{\xi}^2 = \left(\begin{smallmatrix} 0 & \vec{v} \\ 0 & 0 \end{smallmatrix} \right) \cdot \left(\begin{smallmatrix} 0 & \vec{v} \\ 0 & 0 \end{smallmatrix} \right) = 0 \qquad \Rightarrow \qquad \hat{\xi}^n = 0 \quad \text{für alle } n \geq 2$$

$$\mathrm{e}^{\hat{\xi}\theta} = \sum_{n=0}^{\infty} \frac{1}{n!} (\hat{\xi}\theta)^n = \mathbf{1}_{4\times 4} + \hat{\xi}\theta = \begin{pmatrix} \mathbf{1}_{3\times 3} & \theta\vec{v} \\ 0 & 1 \end{pmatrix} \in SE(3)$$

Fall 2) $\vec{\omega} \neq 0$, $\|\vec{\omega}\| = 1$ (dies lässt sich durch Skalierung von θ erreichen), $\hat{\xi} = \left(\begin{smallmatrix} \hat{\omega} & \vec{v} \\ 0 & 0 \end{smallmatrix} \right)$ (Rotation und Translation)

Wir transformieren $\hat{\xi}$ zunächst mittels $T = \begin{pmatrix} \mathbf{1} & \vec{\omega} \times \vec{v} \\ 0 & 1 \end{pmatrix}$ in eine geeignetere Form:

$$\hat{\xi}' = T^{-1}\hat{\xi}T = \begin{pmatrix} \mathbf{1} & -\vec{\omega} \times \vec{v} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \hat{\omega} & \vec{v} \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{1} & \vec{\omega} \times \vec{v} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \hat{\omega} & \vec{v} \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{1} & \vec{\omega} \times \vec{v} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \hat{\omega} & \hat{\omega}^2\,\vec{v} + \vec{v} \\ 0 & 0 \end{pmatrix}$$

$$\boxed{\hat{\omega}^2 = \vec{\omega}\vec{\omega}^t - \mathbf{1}} \qquad = \begin{pmatrix} \hat{\omega} & \vec{\omega}\vec{\omega}^t\vec{v} - \vec{v} + \vec{v} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \hat{\omega} & \vec{\omega}\vec{\omega}^t\vec{v} \\ 0 & 0 \end{pmatrix}$$

Für die Potenzen von $\hat{\xi}'$ gilt:

$$\hat{\xi}'^2 = \begin{pmatrix} \hat{\omega} & \vec{\omega}\vec{\omega}^t\,\vec{v} \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \hat{\omega} & \vec{\omega} \cdot \vec{\omega}^t \cdot \vec{v} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \hat{\omega}^2 & \hat{\omega} \cdot \vec{\omega} \cdot \vec{\omega}^t \cdot \vec{v} \\ 0 & 0 \end{pmatrix} \overset{\hat{\omega}\cdot\vec{\omega}=0}{=} \begin{pmatrix} \hat{\omega}^2 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\hat{\xi}'^3 = \begin{pmatrix} \hat{\omega}^2 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \hat{\omega} & \vec{\omega} \cdot \vec{\omega}^t \cdot \vec{v} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \hat{\omega}^3 & 0 \\ 0 & 0 \end{pmatrix} \quad \Rightarrow \quad \hat{\xi}'^n = \begin{pmatrix} \hat{\omega}^n & 0 \\ 0 & 0 \end{pmatrix} \quad \text{für alle } n \geq 2$$

$$\Rightarrow \quad e^{\hat{\xi}'\theta} = \begin{pmatrix} e^{\hat{\omega}\theta} & \vec{\omega}\vec{\omega}^t\vec{v}\theta \\ 0 & 1 \end{pmatrix}$$

Damit kann die Reihe der Exponentialfunktion berechnet werden. Es ergibt sich:

$$e^{\hat{\xi}\theta} = e^{T\cdot\hat{\xi}'\theta\cdot T^{-1}} = \sum_{n=0}^{\infty}\frac{1}{n!}(T\cdot\hat{\xi}'\theta\cdot T^{-1})^n = T\cdot\left[\sum_{n=0}^{\infty}\frac{1}{n!}(\hat{\xi}'\theta)^n\right]\cdot T^{-1} = T\cdot e^{\hat{\xi}'\theta}\cdot T^{-1}$$
$$= T\cdot\begin{pmatrix} e^{\hat{\omega}\theta} & \vec{\omega}\vec{\omega}^t\vec{v}\theta \\ 0 & 1\end{pmatrix}\cdot T^{-1} = \begin{pmatrix} e^{\hat{\omega}\theta} & (\mathbf{1}-e^{\hat{\omega}\theta})\vec{\omega}\times\vec{v}+\vec{\omega}\vec{\omega}^t\vec{v}\theta \\ 0 & 1\end{pmatrix}\in SE(3) \tag{1.5}$$

Die Paare $T^{-1}T$ innerhalb der Potenzen $(T\cdot\hat{\xi}'\theta\cdot T^{-1})^n$ fallen weg und es bleiben jeweils nur das äußere T und T^{-1} übrig, die aus der Summe herausgezogen worden sind. Die erhaltene 4×4-Matrix ist eine homogene Transformation. □
Es bleibt zu zeigen, dass zu jeder homogenen Transformation ein erzeugender Twist existiert (Surjektivität der Abbildung $\exp : se(3)\to SE(3)$).

Satz 1.10 Zu jedem $T\in SE(3)$ existiert ein Twist $\hat{\xi}\in se(3)$ und $\theta\in\mathbb{R}$, so dass $\mathrm{e}^{\hat{\xi}\theta} = T$ gilt.

Beweis:
Fall 1) reine Translation ($R=\mathbf{1}$): Setze $\hat{\xi} = \begin{pmatrix} 0 & \vec{p}/\|\vec{p}\| \\ 0 & 0\end{pmatrix}$ und $\theta = \|\vec{p}\|$.
Dann gilt (s.o.) $e^{\hat{\xi}\theta} = \begin{pmatrix}\mathbf{1} & \vec{p}\\ 0 & 1\end{pmatrix}$.

Fall 2) allgemeine Bewegung ($R\neq\mathbf{1}$)
Unter Verwendung des Ergebnisses aus (1.5) müssen wir die Gleichung $T = \mathrm{e}^{\hat{\xi}\theta}$ lösen und bestimmen dazu zunächst $\vec{\omega}$ und θ wie in Satz 1.3. Dann lösen wir die Gleichung für den Positionsanteil

$$\vec{p} = (\mathbf{1}-e^{\hat{\omega}\theta})\hat{\omega}\vec{v}+\theta\vec{\omega}\vec{\omega}^t\vec{v} = M\vec{v} \qquad \text{mit} \quad M = (\mathbf{1}-e^{\hat{\omega}\theta})\hat{\omega}+\theta\vec{\omega}\vec{\omega}^t\,.$$

Es lässt sich zeigen [**?**], dass die Matrix M invertierbar ist, falls $\theta\in(0,2\pi)$. Dies folgt aus der Tatsache, dass die beiden Matrizen, aus denen M besteht zueinander orthogonale Null-Räume haben, wenn $\theta\neq 0$ und $R\neq\mathbf{1}$, also $M\vec{v}=0\Leftrightarrow\vec{v}=0$.In diesem Fall existiert also zu jedem $\vec{p}$ ein Vektor $\vec{v} = M^{-1}\vec{p}$. □

Bemerkung: Die Abbildung ist – wie auch schon $\exp : so(3)\to SO(3)$ – nicht eindeutig. Ausserdem ist der Vektor $\vec{v}$ im Twist i.a. nicht als Translationsvektor interpretierbar; er beinhaltet sowohl Komponenten von $\vec{p}$ als auch von R, wie das folgende Beispiel zeigt und wie man schon der Matrix M entnehmen kann, die ja auch rotatorische Anteile $(\vec{\omega},\theta)$ enthält.

Beispiel 1.11 Wir betrachten die homogene Transformation T_{ab}, die das feste KS A mittels folgender elementarer Transformationen in das KS B überführt (Abb. 3):

- Translation um l_1 entlang der y_A-Achse
- Rotation um α um die neue z-Achse
- Translation um l_2 entlang der neuen y-Achse

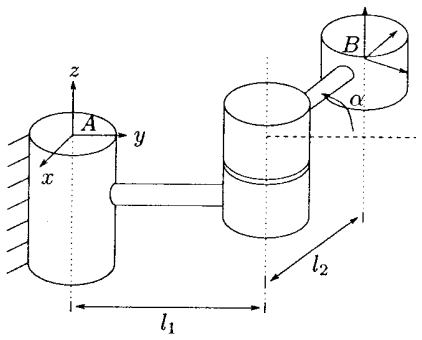


Abbildung 3: Rotation um eine feste Achse im Raum

$$T_{ab}(\alpha) = T_{l_1,y} R_{\alpha,z} T_{l_2,y} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & -l_2\sin\alpha \\ \sin\alpha & \cos\alpha & 0 & l_1 + l_2\cos\alpha \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{ab} & \vec{p}_{ab} \\ 0 & 1 \end{pmatrix}$$

Um den zugehörigen Twist zu finden, berechnen wir zunächst den Rotationsteil $e^{\hat{\omega}\theta} = R_{ab}$. Die Rotation findet offenbar um die z-Achse statt, so dass folgt: $\vec{\omega} = [0,0,1]^t$ bzw. $\hat{\omega} = \left(\begin{smallmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ und $\theta = \alpha$. Um den Anteil $\vec{v}$ zu bestimmen, müssen wir die Gleichung

$$M\vec{v} = [(\mathbf{1} - e^{\hat{\omega}\alpha})\hat{\omega} + \vec{\omega}\vec{\omega}^t\alpha]\vec{v} = \vec{p}_{ab} = \begin{pmatrix} -l_2\sin\alpha \\ l_1 + l_2\cos\alpha \\ 0 \end{pmatrix}$$

lösen. Für M gilt:

$$M = \begin{pmatrix} 1-\cos\alpha & \sin\alpha & \\ -\sin\alpha & 1-\cos\alpha & \\ & & 1-1 \end{pmatrix} \hat{\omega} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \alpha = \begin{pmatrix} \sin\alpha & \cos\alpha - 1 & 0 \\ 1-\cos\alpha & \sin\alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}$$

so dass folgt:

$$\vec{v} = M^{-1}\vec{p}_{ab} = \begin{pmatrix} \frac{\sin\alpha}{2(1-\cos\alpha)} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sin\alpha}{2(1-\cos\alpha)} & 0 \\ 0 & 0 & \frac{1}{\alpha} \end{pmatrix} \begin{pmatrix} -l_2\sin\alpha \\ l_1 + l_2\cos\alpha \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(l_1 - l_2) \\ \frac{\sin\alpha}{2(1-\cos\alpha)}(l_1 + l_2) \\ 0 \end{pmatrix}$$

Der $\vec{v}$-Anteil hat also eine sehr komplizierte Form. Die Darstellung von $\vec{v}$ wird deutlich einfacher, wenn wir nur die relative Transformation der Drehung betrachten:

$$T_R(\theta) = T_{ab}(\theta) \cdot T_{ab}^{-1}(0)\,.$$

Die absolute Transformation $T_{ab}(\theta) = T_R(\theta) \cdot T_{ab}(0)$ setzt sich also aus der (relativen) Drehung $T_R(\theta)$ und der initialen Transformation $T_{ab}(0)$ zusammen und gibt als Ganzes die momentane (von θ abhängige) Lage des KS B relativ zu A an. Dies kann man folgendermaßen interpretieren: Punkte, die relativ zum KS B angegeben sind, werden mittels $T_{ab}(0) : B \to A$ zunächst ins KS A abgebildet und dann mittels $T_R(\theta) : A \to A$ auf ihre endgültigen Koordinaten (immer noch relativ zum KS

A) transformiert. Für $T_R(\theta) = e^{\hat{\xi}_R\theta}$ erhalten wir die wesentlich einfacheren Twist-Koordinaten

$$\xi_R = (\vec{v}, \vec{\omega}) = (l_1, 0, 0,\ 0, 0, 1)^t = (-\vec{\omega} \times \vec{p},\ \vec{\omega}) \quad \text{mit } \vec{p} = (0, l_1, 0)^t\,. \tag{1.6}$$

Die Rechnung dazu:

$$M^{-1} \begin{pmatrix} l_1 \sin\alpha \\ l_1(1-\cos\alpha) \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{l_1 \sin^2\alpha}{2(1-\cos\alpha)} + \frac{1}{2} l_1(1-\cos\alpha) \\ -\frac{1}{2} l_1 \sin\alpha + \frac{\sin\alpha}{2(1-\cos\alpha)} \cdot \cancel{l_1(1-\cos\alpha)} \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix}$$

Es kann gezeigt werden, dass jede Rotationsbewegung um eine Drehachse $l = \{\vec{p} + \lambda\vec{\omega} \,|\, \lambda \in \mathbb{R}\}$ im Raum einen solchen einfachen Twist besitzt. Damit gilt:

für Rotationen:	$\xi_R = (-\vec{\omega} \times \vec{p},\ \vec{\omega})$
für Translationen:	$\xi_T = (\vec{\omega},\ 0)$

1.4 Screw-Koordinaten

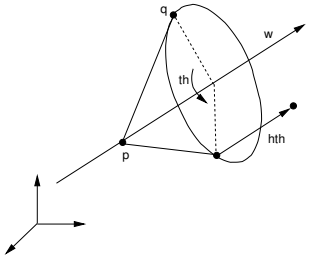


Abbildung 4: Bewegung eines Punktes $\vec{q}$ bei Anwendung eines Screw.

Wir wollen nun die Rotation $R_{\vec{\omega}}(\theta)$ um die Achse $\vec{\omega}$ und die Translation $T_{\vec{\omega}}(h\theta)$ entlang der Achse $\vec{\omega}$ in einer Bewegung zusammenfassen (Abb. 4). Da die Translation entlang der Rotationsachse erfolgt, kommutieren die beiden Bewegungen (gilt nicht im Allgemeinen!) und wir erhalten eine Schraubenbewegung $S(\theta)$, die von dem Twist $\hat{\xi}_S\,\theta = (\hat{\xi}_R + h\,\hat{\xi}_T)\theta$ erzeugt wird:

$$S(\theta) = R_{\vec{\omega}}(\theta) \cdot T_{\vec{\omega}}(h\theta) = T_{\vec{\omega}}(h\theta) \cdot R_{\vec{\omega}}(\theta) = \begin{pmatrix} R_{\vec{\omega},\theta} & -R_{\vec{\omega},\theta}\vec{p} + \vec{p} + h\theta\vec{\omega} \\ 0 & 1 \end{pmatrix} = \mathrm{e}^{(\hat{\xi}_R + h\hat{\xi}_T)\theta}$$

Definition 1.12 Ein Screw besteht aus

- einer Achse $\{l = \vec{p} + \lambda\vec{w},\ \lambda \in \mathbb{R}\}$,
- der Steigung (Pitch) h
- und dem Betrag M.

Er beschreibt eine Rotation um die Achse l um den Winkel $\theta = M$ bei gleichzeitiger Translation entlang dieser Achse um den Betrag $d = h\theta$. Falls $h = \infty$ handelt es sich um eine reine Translation entlang der Achse l um den Betrag M.

Wie wir gesehen haben, hat der zugehörige Twist die Form $\xi_S\,\theta = (\xi_R + h\xi_T)\theta = (-\vec{\omega} \times \vec{p} + h\vec{\omega},\ \vec{\omega})\theta$. Umgekehrt können aus Twist-Koordinaten $\xi\,\theta = (\vec{v}, \vec{\omega})\theta$ sehr leicht die Screw-Koordinaten berechnet werden:

	$\vec{\omega} \neq 0$ (allgemein)	$\vec{\omega} = 0$ (reine Translation)
Achse l	$\left\{ \frac{\vec{\omega} \times \vec{v}}{\|\vec{\omega}\|^2} + \lambda\vec{\omega} \,\middle|\, \lambda \in \mathbb{R} \right\}$	$\{0 + \lambda\vec{v} \,\mid\, \lambda \in \mathbb{R}\}$
Steigung h	$\frac{\vec{\omega}^t\vec{v}}{\|\vec{\omega}\|^2}$	∞
Betrag M	$\|\theta\vec{\omega}\|$	$\|\theta\vec{v}\|$

Damit besteht eine 1:1-Korrespondenz zwischen Twists und Screws. Wir unterscheiden folgende spezielle Screws:

- Steigung-Null Screw: reine Rotation
- Steigung-Unendlich Screw: reine Translation

Zusammenfassung: Zu jeder homogenen Transformation $T \in SE(3)$ existiert ein Twist (bzw. ein Screw), der diese Transformation erzeugt. Wie wir in Beispiel 1.11 gesehen haben, ist der Twist i.d.R. abhängig vom Umfang der Bewegung, im Beispiel dem Drehwinkel α. Im Falle von „echten“ Schraubenbewegungen kann der Betrag $M = \theta$ jedoch ausgeklammert werden: $\xi = \theta\bar{\xi}$, so dass $\bar{\xi}$ für alle Werte von θ dieselbe Form hat. Dieser Umstand macht die Schraubenbewegungen so interessant für die Robotik.

1.5 Vorwärts-Kinematik

Wir betrachten kinematische Ketten mit Drehgelenken (Winkel $\theta_i \in Q_i \subseteq [0, 2\pi)$) und prismatischen Gelenken (Verschiebung $\theta_i \in Q_i \subseteq \mathbb{R}$). Der Gelenkwinkelraum eines Manipulators mit k Gelenken ist dann definiert durch $Q = Q_1 \times Q_2 \times \cdots \times Q_k$. Gegeben ein ortsfestes Basis-KS S und ein „am Endeffektor befestigtes“ Tool-KS T, gibt die Vorwärtskinematik $T_{st} : Q \to SE(3)$ die relative Lage von T gegenüber S in Abhängigkeit von der gewählten Gelenkkonfiguration $\theta \in Q$ an.

Diese Transformation kann man bestimmen durch:

1. Angabe von Relativ-Transformationen (z.B. DH-Konvention)

   $$T_{st}(\theta_1, \ldots, \theta_k) = T_{s1}(\theta_1) \cdot T_{12}(\theta_2) \cdot \ldots \cdot T_{k-1\,k}(\theta_k) \cdot T_{kt}$$

   Dies erfordert die Festlegung von lokalen KS L_i für jedes Segment und die Bestimmung der Relativ-Transformation.

2. Angabe der Twists, die die Bewegung der Gelenke erzeugen.

Die zweite Methode ist meist einfacher, da der erzeugende Twist i.d.R. direkt abgelesen werden kann. Die DH-Konvention erfordert hingegen einige Anstrengungen, um die Parameter für die Relativ-Transformationen zu bestimmen.

Die Idee der Twist-Darstellung beruht darauf, die Twists der durch die Gelenke erzeugten (Schrauben)bewegungen in einer möglichst einfachen Referenzkonfiguration des Manipulators, in der alle Gelenkparameter $\theta_i = 0$ sind (Nullstellung), abzulesen und die zugehörige Koordinaten-Transformation $T_{st}(0) : T \to S$ zu bestimmen.

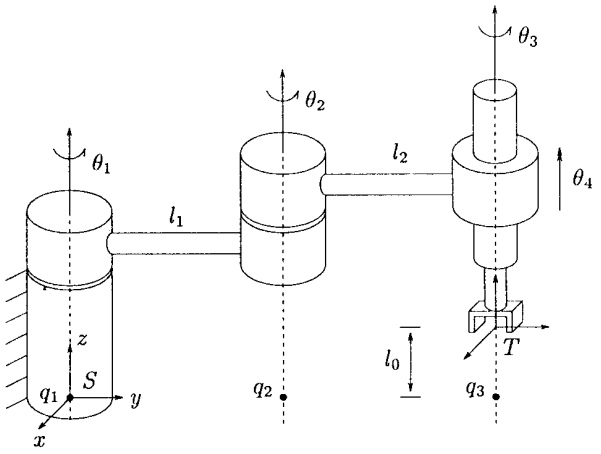


Abbildung 5: SCARA-Manipulator in einer einfachen Referenzkonfiguration

Beginnend beim letzten Gelenk θ_k wird nun die Vorwärtskinematik aufgebaut, indem wir die resultierende Gesamttransformation bestimmen, die sich durch Bewegung eines einzelnen Gelenks θ_i ergibt:

$$\begin{aligned}
&\text{Bewege } \theta_k\text{:} && T_{st}(\theta_k) = \mathrm{e}^{\hat{\xi}_k\theta_k}\, T_{st}(0)\\
&\text{Bewege } \theta_{k-1}\text{:} && T_{st}(\theta_{k-1},\theta_k) = \mathrm{e}^{\hat{\xi}_{k-1}\theta_{k-1}}\, T_{st}(\theta_k) = \mathrm{e}^{\hat{\xi}_{k-1}\theta_{k-1}}\, \mathrm{e}^{\hat{\xi}_k\theta_k}\, T_{st}(0)\\
& && \dots\\
&\text{Bewege } \theta_1\text{:} && T_{st}(\theta_1,\dots,\theta_k) = \mathrm{e}^{\hat{\xi}_1\theta_1}\, T_{st}(\theta_2,\dots,\theta_k) = \mathrm{e}^{\hat{\xi}_1\theta_1}\, \mathrm{e}^{\hat{\xi}_2\theta_2} \dots \mathrm{e}^{\hat{\xi}_k\theta_k}\, T_{st}(0)
\end{aligned}$$

Da die Bewegung immer relativ zum Basis-KS S ausgeführt wird, müssen die neuen Transformationen von links multipliziert werden (im Gegensatz zur DH-Konvention, die die Gelenkbewegungen in umgekehrter Reihenfolge ausführt). Die so bestimmte Vorwärtskinematik ist natürlich unabhängig von der Reihenfolge der Gelenkbewegungen, das „Aufrollen“ von Hinten stellt jedoch die einfachste Herleitung dar. Beginnt man beim ersten Gelenk, verändern sich natürlich die ursprünglich abgelesenen Twists $\hat{\xi}_i$ und man muss durch eine geeignete Transformation erst neue Twists $\hat{\xi}_i'$ daraus bestimmen (Übung).

Die Vorwärtskinematik ist also gegeben durch:

$$T_{st}(\theta_1,\dots,\theta_k) = \mathrm{e}^{\hat{\xi}_1\theta_1}\, \mathrm{e}^{\hat{\xi}_2\theta_2} \dots \mathrm{e}^{\hat{\xi}_k\theta_k}\, T_{st}(0)$$

Die Twists $\hat{\xi}_i$ hängen lediglich von der Wahl des Basis-KS S und der Referenzkonfiguration ab. (Sowohl die Richtung der Gelenkachsen $\vec{\omega}$ als auch deren Unterstützungspunkte $\vec{p}$ verändern sich mit der Referenzkonfiguration und der Wahl des Basis-KS S.) Für Dreh- bzw. Schiebegelenke hatten wir:

$$\xi_i^R = (-\vec{\omega}_i \times \vec{p}_i, \vec{\omega}_i) \qquad \text{bzw.} \qquad \xi_i^T = (\vec{\omega}_i, 0)$$

Beispiel 1.13 (SCARA) Als Beispiel betrachten wir einen SCARA-Manipulator (Selective Compliance Assembly Robot Arm) wie in Abb. 5. Die Achsen lassen sich leicht ablesen:

$$\vec{\omega}_i = (0,0,1)^t \qquad \vec{q_1} = (0,0,0)^t \quad \vec{q_2} = (0,l_1,0)^t \quad \vec{q_3} = (0,l_1+l_2,0)^t$$

Daraus bestimmen wir die Twist-Koordinaten

$$\xi_1 = (-\vec{\omega}_1 \times \vec{q}_1, \vec{\omega}_1) = (0,0,0,\ 0,0,1)^t \quad \xi_2 = (-\vec{\omega}_2 \times \vec{q}_2, \vec{\omega}_2) = (l_1,0,0,\ 0,0,1)^t$$
$$\xi_3 = (-\vec{\omega}_3 \times \vec{q}_2, \vec{\omega}_3) = (l_1+l_2,0,0,\ 0,0,1)^t \quad \xi_4 = (\vec{\omega}_4, 0) = (0,0,1,\ 0,0,0)^t$$

und schließlich die einzelnen Bewegungen:

$$T_{st}(0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_1+l_2 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathrm{e}^{\hat{\xi}_1\theta_1} = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathrm{e}^{\hat{\xi}_2\theta_2} = \begin{pmatrix} c_2 & -s_2 & 0 & l_1 s_2 \\ s_2 & c_2 & 0 & l_1(1-c_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\mathrm{e}^{\hat{\xi}_3\theta_3} = \begin{pmatrix} c_3 & -s_3 & 0 & (l_1+l_2)s_3 \\ s_3 & c_3 & 0 & (l_1+l_2)(1-c_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathrm{e}^{\hat{\xi}_4\theta_4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \theta_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ausmultiplizieren ergibt:

$$T_{st}(\theta) = \begin{pmatrix} R(\theta) & \vec{p}(\theta) \\ 0 & 1 \end{pmatrix}$$
$$R(\theta) = \begin{pmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$\vec{p}(\theta) = \begin{pmatrix} -l_1\sin\theta_1 - l_2\sin(\theta_1+\theta_2) \\ l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \\ l_0+\theta_4 \end{pmatrix}$$

1.6 Geschwindigkeiten

Wir wollen nun auch Geschwindigkeiten von festen Körpern in Form von Twists ausdrücken. Bei der Herleitung der Exponentialdarstellung von Rotationen sind wir von der Gleichung

$$\dot{q}_a = \vec{\omega}^a \times \vec{q}_a = \hat{\omega}^a \vec{q}_a$$

ausgegangen, die die Geschwindigkeit eines Punktes $\vec{q}$ bezüglich des ortsfesten KS A bei Anwendung einer Rotation $\vec{\omega}^a$ angibt. Wir wollen nun $\hat{\omega}^a$ durch R_{ab} ausdrücken:

$$\vec{q}_a(t) = R_{ab}(t)\vec{q}_b$$

Ableiten:
$$\begin{aligned} \dot{q}_a(t) &= \dot{R}_{ab}(t)\vec{q}_b + \overbrace{R_{ab}(t)\dot{\vec{q}}_b}^{0} \\ &= \dot{R}_{ab}(t) \cdot R_{ab}^{-1}(t) \cdot R_{ab}(t) \cdot \vec{q}_b \\ &= \dot{R}_{ab}(t) \cdot R_{ab}^{-1}(t) \cdot \vec{q}_a(t) \end{aligned}$$

Wir können also $\hat{\omega}^a$ mit $\dot{R}_{ab}(t) \cdot R_{ab}^{-1}(t)$ identifizieren, wobei beide Matrizen antisymmetrisch sind (Übung). Die Matrix $\dot{R}$ selbst ist wenig aussagekräftig, da sie Ortskoordinaten – ausgedrückt im körperfesten KS ($\vec{q}_b$) – in Geschwindigkeiten – ausgedrückt im Welt-KS ($\dot{q}_a$) – überführt und daher auf zwei verschiedene KS Bezug nimmt. Außerdem ist $\dot{R}$ weder Element von $SO(3)$ noch von $so(3)$.

Definition 1.14 Wir definieren: $\hat{\omega}^a_{ab} := \dot{R}_{ab} \cdot R^{-1}_{ab}$ und $\hat{\omega}^b_{ab} := R^{-1}_{ab} \cdot \dot{R}_{ab}$. Während $\vec{\omega}^a_{ab}$ die Rotationsgeschwindigkeit im *ortsfesten Welt-KS A* angibt, beschreibt $\vec{\omega}^b_{ab}$ die Rotationsgeschwindigkeit in Koordinaten des *sich bewegenden Körper-KS B*. **In beiden Fällen wird die Bewegung von B relativ zu A beschrieben!**

Um zu zeigen, dass die Definition von $\hat{\omega}^b_{ab}$ sinnvoll ist, d.h. dass der Vektor $\vec{\omega}^b_{ab} = R^{-1}_{ab}\vec{\omega}^a_{ab}$ tatsächlich die im KS B ausgedrückte Version von $\vec{\omega}^a_{ab}$ ist, beweisen wir zunächst folgendes allgemeine Lemma:

Lemma 1.15 Für beliebiges $\vec{\omega} \in \mathbb{R}^3$ und $R \in SO(3)$ gilt:

$$\vec{\omega}\,' = R\vec{\omega} \qquad \Leftrightarrow \qquad \hat{\omega}' = R \cdot \hat{\omega} \cdot R^{-1} \qquad \Leftrightarrow \qquad \widehat{R\vec{\omega}} = R \cdot \hat{\omega} \cdot R^{-1}$$

Beweis: Wir zeigen, dass die Matrizen $\hat{w}\,' = \widehat{R\vec{\omega}}$ und $R\cdot\hat{\omega}\cdot R^{-1}$ identische Wirkung auf beliebige Vektoren $\vec{p}$ haben:

$$\widehat{R\vec{\omega}} \cdot \vec{p} = (R\vec{\omega}) \times \vec{p} = (R\vec{\omega}) \times (RR^{-1}\vec{p}) = R \cdot (\vec{\omega} \times R^{-1}\vec{p}) = R \cdot \hat{\omega} \cdot R^{-1} \cdot \vec{p}$$

Dabei haben wir die Linearität des Kreuzproduktes genutzt, um die Matrix R auszuklammern. □

Damit folgt aus $\vec{\omega}^b_{ab} = R^{-1}_{ab}\vec{\omega}^a_{ab}$:

$$\hat{\omega}^b_{ab} = R^{-1}_{ab} \cdot \hat{\omega}^a_{ab} \cdot R_{ab} = R^{-1}_{ab} \cdot \dot{R}_{ab} \cdot R^{-1}_{ab} \cdot R_{ab} = R^{-1}_{ab} \cdot \dot{R}_{ab}$$

Die Geschwindigkeit eines Punktes $\vec{q}$ kann also entweder bzgl. des Welt-KS A oder bzgl. des körperfesten KS B angegeben werden:

$$\vec{v}_a(t) = \hat{\omega}^a_{ab}\, R_{ab}(t)\, \vec{q}_b = \hat{\omega}^a_{ab}\, \vec{q}_a(t) = \vec{\omega}^a_{ab} \times \vec{q}_a(t)$$
$$\vec{v}_b(t) = R^{-1}_{ab}(t)\vec{v}_a(t) = R^{-1}_{ab}(t)\hat{\omega}^a_{ab}\, R_{ab}(t)\, \vec{q}_b = \hat{\omega}^b_{ab}\vec{q}_b = \vec{\omega}^b_{ab} \times \vec{q}_b$$

Allgemeine Transformation Für eine allgemeine homogene Transformation

$$T_{ab}(t) = \begin{pmatrix} R_{ab}(t) & p_{ab}(t) \\ 0 & 1 \end{pmatrix} \in SE(3)$$

betrachten wir analog:

$$\dot{T}_{ab}T^{-1}_{ab} = \begin{pmatrix} \dot{R}_{ab} & \dot{\vec{p}}_{ab} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} R^{-1}_{ab} & -R^{-1}_{ab}\,\vec{p}_{ab} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \dot{R}_{ab}\, R^{-1}_{ab} & -\dot{R}_{ab}\, R^{-1}_{ab}\vec{p}_{ab} + \dot{p}_{ab} \\ 0 & 0 \end{pmatrix}$$
$$=: \begin{pmatrix} \hat{\omega}^a_{ab} & \vec{v}^a_{ab} \\ 0 & 0 \end{pmatrix} = \hat{V}^a_{ab} \in se(3)$$

und

$$T^{-1}_{ab}\dot{T}_{ab} = \begin{pmatrix} R^{-1}_{ab} & -R^{-1}_{ab}\,\vec{p}_{ab} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{R}_{ab} & \dot{\vec{p}}_{ab} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R^{-1}_{ab}\,\dot{R}_{ab} & R^{-1}_{ab}\dot{p}_{ab} \\ 0 & 0 \end{pmatrix}$$
$$=: \begin{pmatrix} \hat{\omega}^b_{ab} & \vec{v}^b_{ab} \\ 0 & 0 \end{pmatrix} = \hat{V}^b_{ab} \in se(3)$$

Beide Größen können also als Twist aufgefasst werden!

Definition 1.16 Der Twist $\hat{V}^a_{ab} = \dot{T}_{ab}\, T^{-1}_{ab}$ heißt Welt-Geschwindigkeit und gibt die Geschwindigkeit des sich bewegenden KS B relativ zum Welt-KS A in Koordinaten von A an. Der Twist $\hat{V}^b_{ab} = T^{-1}_{ab}\, \dot{T}_{ab}$ heißt Körper-Geschwindigkeit und drückt dieselbe Geschwindigkeit in Koordinaten von B aus. In Twist-Koordinaten erhalten wir die Darstellung als 6-dim. Vektor:

$$V^a_{ab} = \begin{bmatrix} \vec{v}^a_{ab} \\ \vec{\omega}^a_{ab} \end{bmatrix} = \begin{bmatrix} -\dot{R}_{ab}\, R^{-1}_{ab} \vec{p}_{ab} + \dot{p}_{ab} \\ (\dot{R}_{ab}\, R^{-1}_{ab})^\vee \end{bmatrix}$$
$$V^b_{ab} = \begin{bmatrix} \vec{v}^b_{ab} \\ \vec{\omega}^b_{ab} \end{bmatrix} = \begin{bmatrix} R^{-1}_{ab} \dot{p}_{ab} \\ (R^{-1}_{ab}\, \dot{R}_{ab})^\vee \end{bmatrix}$$

Der Operator $^\vee$ macht dabei die Operation $^\wedge$ rückgängig.

Für die Geschwindigkeit eines Punktes $\vec{q}$ des Körpers gilt also in homogenen bzw. in 3D-Koordinaten:

$$\bar{v}_a(t) = \hat{V}^a_{ab}\, \bar{q}_a(t) \qquad \text{bzw.} \qquad \vec{v}_a(t) = \vec{\omega}^a_{ab} \times \vec{q}_a(t) + \vec{v}^a_{ab}$$
$$\bar{v}_b(t) = \hat{V}^b_{ab}\, \bar{q}_b \qquad \text{bzw.} \qquad \vec{v}_b(t) = \vec{\omega}^b_{ab} \times \vec{q}_b + \vec{v}^b_{ab}$$

Bemerkung: Für eine reine Translation oder eine reine Rotation erhält man die intuitiv erwarteten Geschwindigkeiten V^a_{ab}:

Translation: $V^a_{ab} = (\dot{\vec{p}}_{ab}, 0)$
Rotation: $V^a_{ab} = (0, \vec{\omega}^a_{ab})$

Im allgemeinen gibt die Komponente $\vec{\omega}^a_{ab}$ ebenfalls die Rotationsgeschwindigkeit an, die Bedeutung der linearen Komponente $\vec{v}^a_{ab}$ ist jedoch nicht mehr intuitiv klar: Sie gibt die Geschwindigkeit eines im KS B befestigten (imaginären) Punktes an, der sich zur Zeit t durch den Ursprung des KS A bewegen würde.

Zwischen den beiden Darstellungen der Geschwindigkeits-Twists gilt folgende Relation:

$$\hat{V}^a_{ab} = \dot{T}_{ab}\, T^{-1}_{ab} = T_{ab}\, T^{-1}_{ab}\, \dot{T}_{ab}\, T^{-1}_{ab} = T_{ab}\, \hat{V}^b_{ab}\, T^{-1}_{ab}$$

Um die Transformation direkt auf Basis der Twist-Koordinaten zu machen, benötigen wir die folgende

Definition 1.17 Zu einer gegebenen Transformation $T = \begin{pmatrix} R & \vec{p} \\ 0 & 1 \end{pmatrix}$ definieren wir die Adjungierte Ad_T:

$$Ad_T = \begin{pmatrix} R & \hat{p}\, R \\ 0 & R \end{pmatrix} \in \mathbb{R}^{6\times 6}\,.$$

Für die Inverse bzw. die Transponierte der Adjungierten gilt:

$$Ad^{-1}_T = \begin{pmatrix} R^t & -R^t\, \hat{p} \\ 0 & R^t \end{pmatrix} = Ad_{T^{-1}} \quad \neq \quad Ad^t_T = \begin{pmatrix} R^t & 0 \\ -R^t\, \hat{p} & R^t \end{pmatrix}\,.$$

Die Adjungierte transformiert nun Twist-Koordinaten bzgl. verschiedener KS (hier von B nach A):

$$Ad_{T_{ab}} V^b_{ab} = \begin{pmatrix} R_{ab} & \hat{p}_{ab}\, R_{ab} \\ 0 & R_{ab} \end{pmatrix} \begin{pmatrix} \vec{v}^b_{ab} \\ \vec{\omega}^b_{ab} \end{pmatrix} = \begin{pmatrix} R_{ab}\, \vec{v}^b_{ab} + \hat{p}_{ab}\, R_{ab}\, \vec{\omega}^b_{ab} \\ R_{ab}\, \vec{\omega}^b_{ab} \end{pmatrix} \overset{Def}{=} \begin{pmatrix} \dot{\vec{p}}_{ab} + \vec{p}_{ab} \times \vec{\omega}^a_{ab} \\ \vec{\omega}^a_{ab} \end{pmatrix}$$
$$= V^a_{ab}\,.$$

Dies gilt auch für beliebige Twists:

Lemma 1.18 Sei $\hat{\xi} \in se(3)$ ein Twist und $T \in SE(3)$ eine homogene Transformation. Dann ist $\hat{\xi}' = T\,\hat{\xi}\,T^{-1}$ ebenfalls ein Twist mit den Koordinaten $\xi' = Ad_T\,\xi$. (Übung)

Die Wirkung von $Ad_{T_{ab}}$ kann wieder passiv und aktiv aufgefasst werden. In der passiven Version wird ein Twist von der Darstellung ξ_b im KS B in die Darstellung ξ_a im KS A überführt. In der aktiven Variante wird ein Twist ξ_a mittels der Transformation T_{ab} in einen anderen Twist ξ_a' – beide im KS A dargestellt – überführt.

Beispiel 1.19 Wir wollen die Welt-Geschwindigkeit V_{ab}^a der Transformation

$$T_{ab}(t) = \begin{pmatrix} \cos\theta(t) & -\sin\theta(t) & 0 & -l_2\sin\theta(t) \\ \sin\theta(t) & \cos\theta(t) & 0 & l_1 + l_2\cos\theta(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

aus Beispiel 1.11 bestimmen. Wir lassen die Zeitabhängigkeit und die Indizes ab der Übersichtlichkeit halber weg. Nach Definition gilt: $\vec{v}_{ab}^a = -\dot{R}R^t\vec{p} + \dot{\vec{p}}$ und $\hat{\omega}_{ab}^a = \dot{R}R^t$.

$$\dot{R}R^t = \begin{pmatrix} -\dot\theta\sin\theta & -\dot\theta\cos\theta & 0 \\ \dot\theta\cos\theta & -\dot\theta\sin\theta & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -\dot\theta & 0 \\ \dot\theta & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\vec{\omega}_3^a & 0 \\ \vec{\omega}_3^a & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Es folgt also:

$$\vec{\omega}_{ab}^a = \begin{pmatrix} 0 \\ 0 \\ \dot\theta(t) \end{pmatrix} \quad \text{und} \quad \vec{v}_{ab}^a = \begin{pmatrix} 0 & \dot\theta & 0 \\ -\dot\theta & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -l_2\sin\theta \\ l_1 + l_2\cos\theta \\ 0 \end{pmatrix} + \begin{pmatrix} -l_2\dot\theta\cos\theta \\ -l_2\dot\theta\sin\theta \\ 0 \end{pmatrix} = \begin{pmatrix} l_1\dot\theta(t) \\ 0 \\ 0 \end{pmatrix}$$

Wir erhalten also wieder den Twist $\xi = (l_1, 0, 0,\ 0, 0, 1)^t$, der im Beispiel 1.11 die eigentliche Bewegung $\mathrm{e}^{\hat{\xi}\theta(t)}$ erzeugte (s. Gl. 1.6).

Bemerkung: Dies gilt auch für allgemeine Schraubenbewegungen der Form $T(t) = \mathrm{e}^{\hat{\xi}\theta(t)} \cdot T(0)$. Für die Ableitung folgt nämlich:

$$\dot{T}(t) = \dot\theta\hat{\xi} \cdot \mathrm{e}^{\hat{\xi}\theta(t)} \cdot T(0) + \mathrm{e}^{\hat{\xi}\theta(t)} \cdot 0$$
$$\hat{V}_{ab}^a = \dot{T}(t) \cdot T^{-1}(t) = \dot\theta\hat{\xi} \cdot \mathrm{e}^{\hat{\xi}\theta(t)} \cdot T(0) \cdot T^{-1}(0) \cdot \mathrm{e}^{-\xi\theta(t)} = \dot\theta\hat{\xi}$$

Die Geschwindigkeit $\hat{V}_{ab}^a$ stimmt demnach mit der Ableitung des erzeugenden Twists $\frac{d}{dt}\hat{\xi}\theta(t) = \hat{\xi}\dot\theta(t)$ überein.

Lemma 1.20 (Übung) Für drei sich gegeneinander bewegende KS (A, B und C) gelten folgende Relationen für die Welt- und Körpergeschwindigkeiten:

$$V_{ac}^a = V_{ab}^a + Ad_{T_{ab}}V_{bc}^b$$
$$V_{ac}^c = Ad_{T_{bc}^{-1}}V_{ab}^b + V_{bc}^c$$
$$V_{ab}^b = -V_{ba}^b = -Ad_{T_{ba}}V_{ba}^a$$

Warum drücken wir Geschwindigkeiten mit so vielen Indizes aus? Die Lage eines Körpers im Raum ist zunächst unabhängig von der Wahl eines KS – erst durch die Angabe von Koordinaten legt man sich auf *ein* KS fest, z.B. $\vec{q}_a$ bzgl. A oder $\vec{q}_b$ bzgl.

B. Geschwindigkeiten hingegen beschreiben immer Relativ-Geschwindigkeiten zwischen zwei Systemen, weshalb man die beiden Bezugskoordinatensysteme als untere Indizes V_{ab} angibt. Der obere Index gibt an, bzgl. welchen Koordinatensystems der Geschwindigkeitsvektor angegeben ist. Dies kann auch bezüglich eines dritten, völlig unabhängigen Koordinatensystems geschehen.

Wir haben gesehen, dass wir sowohl Geschwindigkeiten als auch homogene Transformationen durch Twists ausdrücken können. Dabei werden die homogenen Transformationen durch Anwendung der Exponential-Funktion aus einem Twist *erzeugt*, sind aber selbst *kein* Twist. Geschwindigkeiten können hingegen direkt als Twist interpretiert werden.

1.7 Jacobi-Matrix

Die Jacobi-Matrix bildet bekanntermaßen Gelenkgeschwindigkeiten $\dot{\theta}_i(t)$ auf Tool-Geschwindigkeiten V_{st} ab. Die Berechnung der Jacobi-Matrix aus der Vorwärtskinematik erfordert die Betrachtung sog. differentieller Bewegungen. Mit der Twist-Darstellung ergibt sich:

$$\hat{V}_{st}^s = \dot{T}_{st}(\theta) T_{st}^{-1}(\theta) = \sum_{i=1}^{k} \left(\frac{\partial T_{st}}{\partial \theta_i} \dot{\theta}_i \right) T_{st}^{-1}(\theta) = \sum_{i=1}^{k} \left(\frac{\partial T_{st}}{\partial \theta_i} T_{st}^{-1}(\theta) \right) \dot{\theta}_i$$

Für die einzelnen Differentiale erhalten wir:

$$\begin{aligned}
\frac{\partial T_{st}}{\partial \theta_i} T_{st}^{-1}(\theta) &= \mathrm{e}^{\hat{\xi}_1 \theta_1} \dots \mathrm{e}^{\hat{\xi}_{i-1} \theta_{i-1}} \cdot \frac{\partial}{\partial \theta_i} \left(\mathrm{e}^{\hat{\xi}_i \theta_i} \right) \cdot \mathrm{e}^{\hat{\xi}_{i+1} \theta_{i+1}} \dots \mathrm{e}^{\hat{\xi}_k \theta_k} \cdot T_{st}(0) \cdot T_{st}(\theta)^{-1} \\
&= \mathrm{e}^{\hat{\xi}_1 \theta_1} \dots \mathrm{e}^{\hat{\xi}_{i-1} \theta_{i-1}} \cdot \left(\hat{\xi}_i \right) \mathrm{e}^{\hat{\xi}_i \theta_i} \cdot \mathrm{e}^{\hat{\xi}_{i+1} \theta_{i+1}} \dots \mathrm{e}^{\hat{\xi}_k \theta_k} \cdot T_{st}(0) \cdot T_{st}(\theta)^{-1} \\
&= \mathrm{e}^{\hat{\xi}_1 \theta_1} \dots \mathrm{e}^{\hat{\xi}_{i-1} \theta_{i-1}} \cdot \left(\hat{\xi}_i \right) \mathrm{e}^{-\hat{\xi}_{i-1} \theta_{i-1}} \dots \mathrm{e}^{-\hat{\xi}_1 \theta_1} = \hat{\xi}_i'
\end{aligned}$$

Jede einzelne Spalte der Jacobi-Matrix ist also wieder als Twist interpretierbar:

$$\left(\frac{\partial T_{st}}{\partial \theta_i} T_{st}^{-1}(\theta) \right)^{\vee} = \xi_i' = Ad_{\mathrm{e}^{\hat{\xi}_1 \theta_1} \dots \mathrm{e}^{\hat{\xi}_{i-1} \theta_{i-1}}} \xi_i \,,$$

und zwar als der in die aktuelle Konfiguration transformierte Twist ξ_i der Referenzkonfiguration (aktive Interpretation von Ad_*). Offenbar hängt diese Transformation nur von den vorangehenden Gelenken $\theta_1, \dots, \theta_{i-1}$ ab. Die Jacobi-Matrix bekommt also die einfache Form

$$J_{st}^s(\theta) = [\xi_1', \xi_2', \dots, \xi_k'] \quad \text{mit} \quad \xi_i' = Ad_{\mathrm{e}^{\hat{\xi}_1 \theta_1} \dots \mathrm{e}^{\hat{\xi}_{i-1} \theta_{i-1}}} \xi_i$$

und es gilt:

$$V_{st}^s = J_{st}^s(\theta) \cdot \dot{\theta}$$

Analog kann man auch die Jacobi-Matrix $J_{st}^t(\theta)$ für die Geschwindigkeit des Tool-KS aus Sicht dieses sich bewegenden KS herleiten (Übung). Es gilt:

$$V_{st}^t = J_{st}^t(\theta) \cdot \dot{\theta} \quad \text{mit} \quad J_{st}^t(\theta) = Ad_{T_{st}(\theta)}^{-1} \cdot J_{st}^s(\theta)$$

Beispiel 1.21 (SCARA-Manipulator) Die Achsen des Manipulator bleiben fest, lediglich die Unterstützungsvektoren verschieben sich:

$$\vec{q}_1' = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \vec{q}_2' = \begin{pmatrix} -l_1 \sin\theta_1 \\ l_1 \cos\theta_1 \\ 0 \end{pmatrix} \qquad \vec{q}_3' = \begin{pmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ 0 \end{pmatrix}$$

Wir rechnen die Twist-Koordinaten mit der bekannten Formel $\xi_i' = (-\vec{\omega}_i \times \vec{q}_i', \vec{\omega}_i)$ für Drehgelenke aus:

$$J_{st}^s = \begin{pmatrix} 0 & l_1 \cos\theta_1 & l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & 0 \\ 0 & l_1 \sin\theta_1 & l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

1.7.1 Klassische Bestimmung der Jacobi-Matrix

Die Spalten der Jacobi-Matrix J_{st} werden klassisch wie folgt bestimmt:

$$J_i = \begin{cases} [\vec{\omega}_i^s,\ 0] \\ [\vec{\omega}_i^s \times (p^s - q_i^s),\ \vec{\omega}_i^s] \end{cases} \neq J_i^s = \begin{cases} [\vec{\omega}_i^s,\ 0] & \text{prismatic joint} \\ [-\vec{\omega}_i^s \times q_i^s,\ \vec{\omega}_i^s] & \text{revolute joint} \end{cases} \tag{1.7}$$

wobei p^s die Endeffektorposition und q_i^s sowie $\vec{\omega}_i^s$ die Gelenkposition bzw. die Bewegungsachse des i-ten Gelenks sind. Alle Größen sind in Welt-Koordinaten beschrieben. Offenbar widerspricht diese Berechnung unserer Herleitung. Der Unterschied entspricht der Twist-Transformation mittels

$$Ad_{p^s}^{-1} = \begin{pmatrix} \mathbf{1} & -\mathbf{1}\hat{p}^s \\ 0 & \mathbf{1} \end{pmatrix}. \tag{1.8}$$

D.h. diese Jacobi-Matrizen drücken die Geschwindigkeit bzgl. eines Koordinatensystems aus, dass zwar in das Tool-KS verschoben ist, aber noch die Orientierung des Welt-KS beibehält. Dies hat den Vorteil, dass die lineare Geschwindigkeit, die Geschwindigkeit im Ursprung des Tool-KS angibt und nicht die instantane Geschwindigkeit im Ursprung des Welt-KS. Die lineare Geschwindigkeit ist aber trotzdem bzgl. der Orientierung des Welt-KS angegeben.

1.7.2 Iterative Bestimmung der Jacobi-Matix

Die Bestimmung der Jacobi-Matrix über die Adjungierten $Ad_{e^{\hat{\xi}_1\theta_1}\cdots e^{\hat{\xi}_{i-1}\theta_{i-1}}}$ ist recht aufwendig, da viele Matrixmultiplikationen durchzuführen sind. Ein iterativer Ansatz auf Basis der klassischen Bestimmung spart hier Rechenaufwand.

Die Kinematik eines Roboters mit zwei Armen und (mehrfingrigen) Händen kann als Baumstruktur beschrieben werden. Jeder Knoten des Baumes enthält ein Gelenk(typ) und die Transformationsmatrix zum nächsten Gelenk. Gelenke vom Typ `none` erlauben die Definition von festen Relativtransformationen z.B. für das Basis- oder Tool-KS.

Die Vorwärtskinematik lässt sich dann in klassischer Weise durch Verkettung der Relativtransformationen bestimmen:

$$T_{st} = T_{s0} \cdot \prod_{i=1}^{n} \underbrace{T_i(\theta_i) T_{i\,i+1}}_{A_i(\theta_i)} \cdot T_{nt} \tag{1.9}$$

Per Konvention wählt man häufig als Bewegungsachse die z-Achse, so dass

$$T_i(\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{bzw.} \qquad T_i(\theta_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \theta_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.10}$$

Die Bestimmung der Vorwärtskinematik (Position + Geschwindigkeit) in einem kin. Baum erfolgt iterativ, vom Blatt (Endeffektor/Fingerspitze) bis zur Wurzel (Welt-KS):

$T = \mathbf{1}$
node = Endeffektor node
repeat
  $A \equiv [Q, q] = A_{node}(\theta_{node})$ – lokale Transformation des aktuellen Gelenks
  $T \equiv [R, p] = A \cdot T$ – Transformation vom aktuellen Gelenk-KS zum Endeffektor

  if jointType(node) != `none` **then**
    $\xi = [\vec{v},\ \vec{\omega}] = \begin{cases} [0,0,1,\ 0,0,0]^t & \text{prismatic} \\ [0,0,0,\ 0,0,1]^t & \text{revolute} \end{cases}$
    Transformiere ξ in das Tool-KS und setze, die zum Gelenk gehörige Spalte von J:
    $J_{node} = \xi' = Ad_{T^{-1}}\xi = \begin{pmatrix} R^{-1} & -R^{-1}\hat{p} \\ 0 & R^{-1} \end{pmatrix} \xi \quad = \quad [R^{-1}\vec{v} - R^{-1}(p \times \vec{\omega}),\ R^{-1}\vec{\omega}]$
  end if
  node = node.parent
until node = NULL
Transformiere Twists vom Tool-KS in das Welt-KS (nur Rotation):
$J_{st} = \begin{pmatrix} R & \\ & R \end{pmatrix} J$

1.8 Inverse Kinematik

Die Gleichung der Geschwindigkeitskinematik kann invertiert werden:

$$V_{st}^x = J_{st}^x \cdot \dot{\theta} \qquad \dot{\theta} = J^{-1} \cdot V_{st}^x \,. \tag{1.11}$$

Diese Invertierung ist aber nicht immer möglich (für redundante Manipulatoren ist J nicht quadratisch, Singularitäten). In diesen Fällen verwendet man die Pseudoinverse:

$$J^{\#} = \begin{cases} J^t(JJ^t)^{-1} & \text{falls Zeilen von J linear unabhängig (*)} \\ (J^tJ)^{-1}J^t & \text{falls Spalten von J linear unabhängig} \\ \lim_{\delta\to 0} J^t(JJ^t + \delta\mathbf{1})^{-1} & = \lim_{\delta\to 0}(J^tJ + \delta\mathbf{1})^{-1}J^t \quad \text{sonst} \end{cases} \tag{1.12}$$

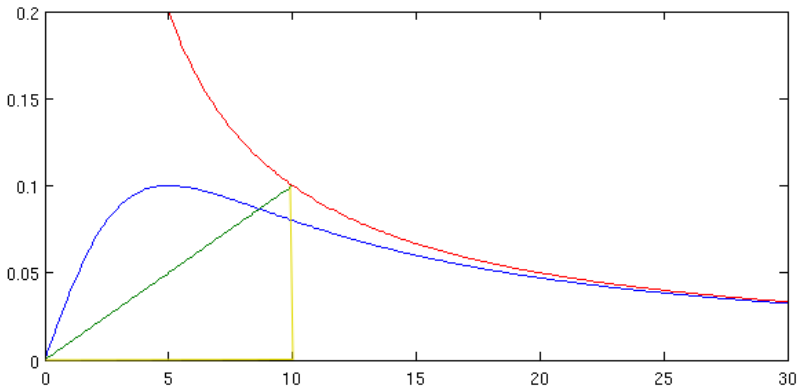


Abbildung 6: Invertierung der Singulärwerte σ_i: σ^{-1}, $\frac{\sigma}{\sigma^2+\lambda^2}$, clipping, smooth clipping

In beiden Fällen gilt: $JJ^\#J = J$ und $J^\#JJ^\# = J^\#$. Bei n Gelenken und m Endeffektor-Freiheitsgraden gilt:

$$J \in \mathbb{R}^{m\times n} \qquad JJ^t \in \mathbb{R}^{m\times m} \qquad J^tJ \in \mathbb{R}^{n\times n} \tag{1.13}$$

Die Pseudoinverse minimiert den Fehler $\|J\dot\theta - V\|^2$ und findet die norm-minimale Lösung ($\min \|\dot\theta\|^2$):

$$\begin{aligned} \min_{\dot\theta}\|J\dot\theta - V\|^2 &= \min_{\dot\theta}(J\dot\theta - V)^t(J\dot\theta - V) \\ \Leftrightarrow \nabla_{\dot\theta}\|J\dot\theta - V\|^2 &= 2J^t(J\dot\theta - V) \stackrel{!}{=} 0 \\ \Leftrightarrow \dot\theta &= (J^tJ)^{-1}J^tV \end{aligned}$$

Eine numerisch stabile (aber aufwendige) Berechnung der Pseudoinversen bietet die Singulärwertzerlegung (SVD):

$$J_{m\times n} = U_{m\times m}\cdot\Sigma_{m\times n}\cdot V^t_{n\times n} \qquad U^tU = \mathbf{1} \tag{1.14}$$

$$J^\# = V\cdot\Sigma^{-1}\cdot U^t \qquad V^tV = \mathbf{1} \tag{1.15}$$

$$\Sigma = \mathrm{diag}(\sigma_1,\ldots,\sigma_k,0,\ldots,0) \tag{1.16}$$

Auch damit ist man vor Singularitäten ($\sigma_i \approx 0$) nicht gefeit. Daher wird eine Regularisierung eingeführt:

$$J^\# = J^t(JJ^t + \lambda\mathbf{1})^{-1} \quad \Leftrightarrow \quad \sigma^{-1} = \frac{\sigma}{\sigma^2+\lambda^2} \quad \Leftrightarrow \quad \min\|J\dot\theta - V\|^2 + \lambda\|\dot\theta\|^2 \tag{1.17}$$

Da dies auch in normalen Situationen zu nicht vernachlässigbaren Fehlern in der Lösung des inversen Problems führt, kann man stattdessen zu kleine Singulärwerte auch clippen:

$$\sigma^{-1} = \begin{cases} 0 & \text{falls } \sigma < \lambda\sigma_{max} \\ \sigma^{-1} & \text{sonst} \end{cases} \tag{1.18}$$

Dies wiederum führt an der Grenze $\lambda\sigma_{max}$ zu unstetigem Verhalten, weshalb man am besten smooth clipping verwendet.

1.8.1 Redundanzauflösung

Bei redundanten Manipulatoren $n > m$ und in Singularitäten von J gibt es einen kontinuierlichen Nullraum $N(J)$. Geschwindigkeiten im Nullraum führen per Definition zu keiner Bewegung des Endeffektors:

$$J\cdot\dot\theta_N = 0 \quad \text{für} \quad \dot\theta_N \in N(J) \tag{1.19}$$

Damit ist die Bewegung des Roboters nicht eindeutig bestimmt, man sagt der Roboter ist nicht konservativ, d.h. eine *geschlossene* Trajektorie im Task-Raum führt nicht unbedingt zu einer geschlossenen Trajektorie im Gelenkwinkel-Raum. Man benötigt also ein weiteres Kriterium, um die Redundanz aufzulösen und deterministisch eine bestimmte Postur anzufahren. Ein Beispiel wäre die Präferenz für eine bestimmte Postur $\bar{\theta}$:

$$H = \|\theta - \bar{\theta}\|^2 \tag{1.20}$$

Ein Gradientenabstieg auf dieser zus. Kostenfunktion H wird dann auf den Nullraum projiziert und der Task-Bewegung hinzuaddiert:

$$\dot{\theta}_H = -\alpha \nabla H \tag{1.21}$$

$$\dot{\theta} = J^{\#} \cdot V + N(J) \cdot \dot{\theta}_H \tag{1.22}$$

$$N(J) = (\mathbf{1} - J^{\#} J) = V_N \cdot V_N^t \qquad \text{Nullraumprojektor} \tag{1.23}$$

1.8.2 Hierarchie von Tasks

Dieser Prozess kann im Prinzip wiederholt werden, um mehrere Tasks $V_1, ...V_k$ entsprechend dieser Reihenfolge zu priorisieren:

$$\dot{\theta} = J_1^{\#} \cdot V_1 + N(J_1) \cdot \left(J_2^{\#} \cdot V_2 + N(J_2) \cdot \left(\cdots J_{k-1}^{\#} \cdot V_{k-1} + N(J_{k-1}) J_k^{\#} \cdot V_k \right) \right)$$

Dieser naive Ansatz hat aber einen Haken: Die Optimierung in den Unterräumen berücksichtigt nicht die Einschränkung auf den jeweiligen Unterraum, so dass eine gefundene Lösung durch den Nullraum-Projektor evtl. komplett "gepruned" wird. Stattdessen sollte die optimale Lösung jeweils im entsprechenden Unterraum gefunden werden. Eine solche Lösung hat Siciliano '91 vorgeschlagen:

$$\begin{array}{lll} \text{Löse} & J_1 \dot{\theta}_1 = V_1 & J_2 \dot{\theta}_2 = V_2 \quad |_{P_1} \\ & \min_{\dot{\theta}_1} \|J_1 \dot{\theta}_1 - V_1\| & \min_{\dot{\theta}_2} \|J_2 P_1 \dot{\theta}_2 - (V_2 - J_2 \dot{\theta}_1)\| \\ & \dot{\theta}_1 = J_1^{\#} V_1 & \dot{\theta}_2 = (J_2 P_1)^{\#} (V_2 - J_2 \dot{\theta}_1) \end{array}$$

Die Ergänzung um P_1 sorgt dabei für die Beschränkung der Optimierung auf den Nullraum von Task 1. Der Term $-J_2 \dot{\theta}_1$ reduziert die Sollgeschwindigkeit V_2 um den Anteil, der bereits von Task 1 "erledigt" wird. Diese Richtung ist orthogonal zum erlaubten Nullraum und könnte vom Optimierer ohnehin nicht erfüllt werden. Es ergibt sich folgendes rekursive Schema:

$$\begin{aligned} \dot{\theta}_0 &= 0 \\ \dot{\theta}_i &= \dot{\theta}_{i-1} + (J_i P_{i-1})^{\#} (V_i - J_i \dot{\theta}_{i-1}) \\ P_{i-1} &= \mathbf{1} - (J_{i-1}^A)^{\#} J_{i-1}^A \end{aligned} \qquad J_{i-1}^A = \begin{pmatrix} J_1 \\ \vdots \\ J_{i-1} \end{pmatrix}$$

Im i-ten Task muss dabei auf den Nullraum *aller* bisherigen Tasks projiziert werden, so dass die augmentierte Jacobi-Matix J_{i-1}^A verwendet wird.

1.8.3 Gewichtung im Task- und Gelenkraum

Möchte man eine gewichtete Norm (z.B. Mahalanobis-Distanz) für die Bestimmung von $\|J\,\dot{\theta} - V\|$ oder von $\|\dot{\theta}\|$ verwenden, ist das sehr leicht möglich:

$$\begin{aligned}\min \|J\,\dot{\theta} - V\|^2_{M_x} &\equiv \|M_x \cdot (J\,\dot{\theta} - V)\|^2 \\ \min \|\dot{\theta}\|^2_{M_q} &\equiv \|M_q^{-1} \cdot \dot{\theta}\|^2\end{aligned}$$

Mit $\dot{q} = M_q^{-1}\dot{\theta}$ ergibt sich:

$$\begin{aligned}\min \|J\,\dot{\theta} - V\|^2_{M_x} &\equiv \min \|M_x\,J\,M_q\,\dot{q} - M_x\,V\|^2 \\ \Leftrightarrow \quad \dot{q} &= (M_x\,J\,M_q)^{\#} \cdot M_x\,V \\ \Leftrightarrow \quad \dot{\theta} &= M_q \cdot (M_x\,J\,M_q)^{\#} \cdot M_x\,V\end{aligned}$$

Damit kann z.B. die Bewegung einzelner Gelenke beschränkt werden ($M_q \to 0$) oder die Kontrolle einzelner Task-Dimensionen aufgegeben werden ($M_x \to 0$).

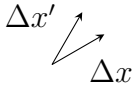
1.8.4 Jacobian-Transpose Methode

Da die Invertierung der Jacobi-Matrix Probleme bei Singularitäten hat, wird häufig auch die Transponierte verwendet:

$$\Delta\theta = \alpha J^t \Delta x$$

Diese Lösung liefert keine exakte Lösung, liefert aber eine Lösung, die in eine ähnliche Richtung zeigt wie Δx:

$$\langle \overbrace{J \cdot \alpha J^t \Delta x}^{\Delta x'}, \Delta x\rangle = \alpha \cdot \langle J^t\Delta x, J^t \Delta x\rangle = \alpha \cdot \|J^t \Delta x\|^2 \geq 0$$



Die Konvergenz ist daher langsamer. Wie wählt man die Schrittweite α?
Idee: Minimiere den verbleibenden Fehler $\|\Delta x' - \Delta x\|^2$:

$$\frac{\partial}{\partial \alpha}\|\Delta x' - \Delta x\|^2 = 2(\alpha J J^t \Delta x - \Delta x) \cdot J J^t \Delta x \stackrel{!}{=} 0 \quad \to \quad \alpha = \frac{\langle J J^t \Delta x, \Delta x\rangle}{\|J J^t \Delta x\|^2}$$

1.9 Wrenches: Kräfte und Drehmomente

Eine verallgemeinerte Kraft umfasst eine lineare Kraft $\vec{f} \in \mathbb{R}^3$ und ein Drehmoment $\vec{\tau} \in \mathbb{R}^3$. Ein solches Paar nennen wir *Wrench* und schreiben:

$$F = (\vec{f}, \vec{\tau}) \in \mathbb{R}^6\,.$$

Ein Wrench F_o wird – genau wie Geschwindigkeiten und Positionsvektoren – immer bzgl. eines KS angegeben. Die Kraft und das Drehmoment sollen jeweils im Ursprung dieses KS angreifen. Im folgenden betrachten wir, wie Wrench-Koordinaten sich von einem Kontakt-KS C in das Objekt-KS O transformieren.

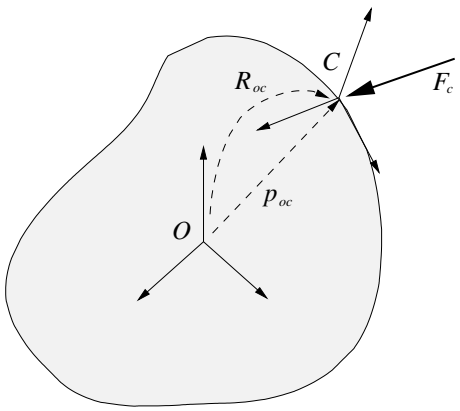


Abbildung 7: Transformation eines Wrench F_c vom Kontakt-KS C ins Objekt-KS O

Fall 1: reine Kraft $F_c = (\vec{f_c}, 0)$

$$F_o = \begin{pmatrix} \vec{f_o} \\ \vec{\tau_o} \end{pmatrix} = \begin{pmatrix} R_{oc}\vec{f_c} \\ \vec{p}_{oc} \times (R_{oc}\,\vec{f_c}) \end{pmatrix}$$

Fall 2: reines Drehmoment $F_c = (0, \vec{\tau_c})$

$$F_o = \begin{pmatrix} \vec{f_o} \\ \vec{\tau_o} \end{pmatrix} = \begin{pmatrix} 0 \\ R_{oc}\vec{\tau_c} \end{pmatrix}$$

Allgemein: $F_c = (\vec{f_c}, \vec{\tau_c}) = (\vec{f_c}, 0) + (0, \vec{\tau_c})$

$$F_o = \begin{pmatrix} \vec{f_o} \\ \vec{\tau_o} \end{pmatrix} = \begin{pmatrix} R_{oc}\vec{f_c} \\ \vec{p}_{oc} \times (R_{oc}\,\vec{f_c}) + R_{oc}\vec{\tau_c} \end{pmatrix} = \begin{pmatrix} R_{oc} & 0 \\ \hat{p}_{oc} \cdot R_{oc} & R_{oc} \end{pmatrix} \cdot \begin{pmatrix} \vec{f_c} \\ \vec{\tau_c} \end{pmatrix} = Ad^t_{T_{oc}^{-1}} \cdot F_c \tag{1.24}$$

Im Vergleich dazu die Transformation von Twists:

$$V^o_{oc} = Ad_{T_{oc}} V^c_{oc} = \begin{pmatrix} R_{oc} & \hat{p}_{oc}R_{oc} \\ 0 & R_{oc} \end{pmatrix} \begin{pmatrix} \vec{v}^c_{oc} \\ \vec{\omega}^c_{oc} \end{pmatrix} = \begin{pmatrix} R_{oc}\vec{v}^c_{oc} + \vec{p}_{oc} \times (R_{oc}\vec{\omega}^c_{oc}) \\ R_{oc}\vec{\omega}^c_{oc} \end{pmatrix}.$$

In beiden Fällen werden die Richtungsvektoren $(\vec{v}, \vec{\omega}, \vec{f}, \vec{\tau})$ mittels der Rotationsmatrix R_{oc} im neuen KS ausgedrückt. Hinzu kommen eine zusätzliche Drehmoment-Komponente $\vec{p}_{oc} \times \vec{f}_o$, die durch die Kraft $\vec{f_c}$ hervorgerufen wird, bzw. eine zusätzliche Geschwindigkeitskomponente $\vec{p}_{oc} \times \vec{\omega}^o_{oc}$, die durch die Rotationsgeschwindigkeit $\vec{\omega}^c_{oc}$ hervorgerufen wird.

Um Gleichung 1.24 allgemein zu beweisen, betrachten wir die instantane Arbeit δW (d.h. die momentane Leistung) die erbracht wird, wenn ein Wrench F_c eine Geschwindigkeit V^c_{ac} hervorruft. Die Geschwindigkeit des Körpers drücken wir relativ zum weltfesten KS A aus.

$$\delta W = V^c_{ac} \cdot F_c = (V^c_{ac})^t \cdot F_c = \left(Ad^{-1}_{T_{oc}} V^o_{ao} + \cancel{V^c_{oc}}\right)^t \cdot F_c = (V^o_{ao})^t \cdot \left(Ad^t_{T_{oc}^{-1}} \cdot F_c\right) = V^o_{ao} \cdot F_o$$

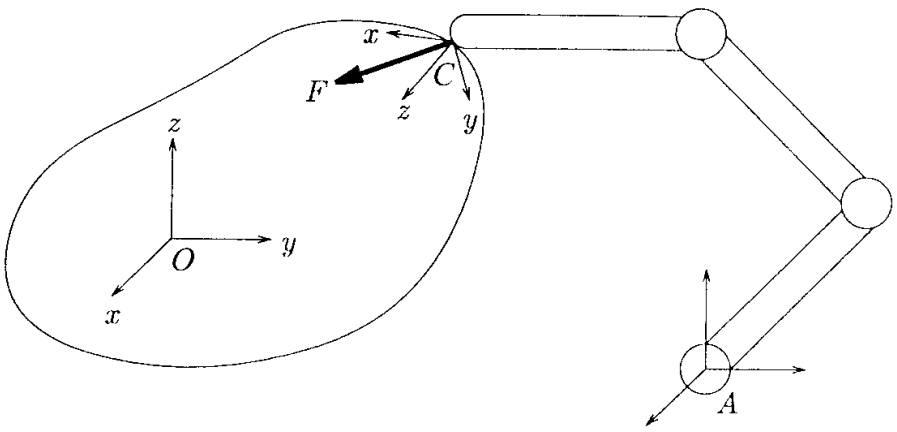


Abbildung 8: Transformation von Wrench-Koordinaten von C nach O

1.9.1 Screw-Koordinaten von Wrenches

Wie schon Twists, können wir auch Wrenches als Screw auffassen und umgekehrt. Der dem Screw (l, h, M) (angegeben bzgl. einem KS A) zugeordnete Wrench F_a entsteht durch Anwendung einer Kraft vom Betrag M entlang der Achse l und einem Drehmoment vom Betrag hM entlang dieser Achse:

$$F_a = M \begin{pmatrix} \vec{\omega} \\ -\vec{\omega} \times \vec{p} + h\vec{\omega} \end{pmatrix} \quad \text{falls } h < \infty \qquad \text{bzw.} \qquad F_a = M \begin{pmatrix} 0 \\ \vec{\omega} \end{pmatrix} \quad \text{falls } h = \infty$$

(Beim zugehörigen Twist wurde eine Rotation um den Betrag M und eine Translation um den Betrag hM durchgeführt – die Rollen von Rotations- und Translationsanteil sind also wieder genau vertauscht.) Umgekehrt können wir zu einem Wrench $F_a = (\vec{f}, \vec{\tau})$ den zugehörigen Screw, durch Lösung obiger Gleichung finden:

	Betrag M	Achse $\vec{\omega}$	Stützpunkt $\vec{p}$	Steigung h
$\vec{f} = 0$ (reines Drehmoment)	$\|\vec{\tau}\|$	$\frac{\vec{\tau}}{\|\vec{\tau}\|}$	0	∞
$\vec{f} \neq 0$ (allgemein)	$\|\vec{f}\|$	$\frac{\vec{f}}{\|\vec{f}\|}$	$\frac{\vec{f} \times \vec{\tau}}{\|\vec{f}\|^2}$	$\frac{\vec{f}^t \vec{\tau}}{\|\vec{f}\|^2}$

1.9.2 Dualität von Twists und Wrenches

Auch diese Gleichungen lassen die Dualität von Twists und Wrenches erkennen. Twists und Wrenches stellen Vektoren in 6-dim. Raum dar, die dual zueinander sind.

Definition 1.22 Ein Twist V und ein Wrench F heißen reziprok zueinander, wenn ihr Skalarprodukt $\delta W = F \cdot V$ Null ist, die beiden Vektoren also orthogonal zueinander stehen. In diesem Falle werden auch die zugeordneten Screws als reziprok bezeichnet. Eine Menge von Twists $V_1, \ldots, V_n$ bzw. eine Menge von Wrenches $F_1, \ldots, F_n$ spannen einen linearen Unterraum der Dimension $m \leq n$ des $\mathbb{R}^6$ auf. Der dazu orthogonale Unterraum mit Dimension $o = 6 - m$ enthält dazu orthogonale Wrenches bzw. Twists.

- Die zu $\{V_1, \ldots, V_n\}$ orthogonalen Wrenches stellen Kräfte dar, die keine Bewegung im Unterraum der Twists erzeugen können.
- Die zu $\{F_1, \ldots, F_n\}$ orthogonalen Twists stellen Bewegungen dar, denen die Wrenches nicht widerstehen können.

1.10 Transponierte Jacobi-Matrix

Die Transponierte der Jacobi-Matrix bildet Kräfte (Wrenches) am Endeffektor eines Manipulators auf Gelenk-Drehmomente ab. Wir leiten diesen Zusammenhang wieder mit Hilfe der geleisteten Arbeit W her:

$$\begin{aligned} & \int_{t_1}^{t_2} \dot{\theta}^t \tau dt = W = \int_{t_1}^{t_2} (V_{st}^s)^t \cdot F_s dt \quad \text{für alle } t_1 \leq t_2 \in \mathbb{R} \\ \Leftrightarrow \quad & \dot{\theta}^t \tau = (J_{st}^s \cdot \dot{\theta})^t \cdot F_s \quad \text{für alle } \dot{\theta} \\ \Leftrightarrow \quad & \tau = (J_{st}^s)^t \cdot F_s \end{aligned}$$

Mittels dieser Gleichung können demnach die Gelenk-Drehmomente berechnet werden, die notwendig sind, um einen bestimmten Wrench am Endeffektor auszuüben.

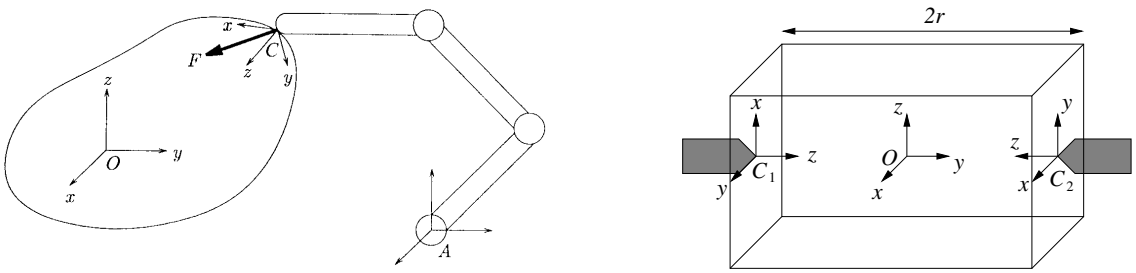


Abbildung 9: Kontaktpunkte an einem zu greifenden Objekt

2 Mehrfingriges Greifen

Motivation

1. Vielzahl von Objekten mit einem Endeffektor greifen.
2. Feine Bewegungen sind einfacher mit kleinen Fingern als mit mehreren großen Armen realisierbar (Dynamik und Trägheit).
3. Manipulierbarkeit erfordert "große" Anzahl von individuell steuerbaren "Fingern", um auch umgreifen zu können.

2.1 Kinematik eines Griffs

Wir untersuchen zunächst die Frage, wie ein Griff definiert ist und welche Zusammenhänge zwischen Kräften und Bewegungen der Finger und des Objektes bestehen.

2.1.1 Kontaktmodelle

Ein Griff zeichnet sich in erster Linie durch eine bestimmte Anzahl von Kontakten an einem (zu greifenden) Objekt aus. Wir definieren für jeden Kontakt i ein KS C_i, dessen z-Achse ins Objektinnere entlang der Kontakt-Normalenrichtung zeigt. Im Ursprung dieser KS greift jeweils ein Wrench F_{c_i} an. Die Lage dieser KS ist durch die Transformation T_{oc_i} relativ zu einem körperfesten KS O angegeben (sinnvollerweise im Schwerpunkt des KS). Die Objektlage selbst ist relativ zu einem Basis-KS P (Palm) festgelegt (T_{po}). Für jeden Kontakt müssen wir nun noch ein Reibungsmodell festlegen. Damit definieren wir die Menge der übertragbaren Kräfte F_{c_i} durch Angabe einer Wrench-Basis und eines Reibungskegels FC_i.

- **Punktkontakt ohne Reibung** Kräfte können nur entlang der Normalenrichtung (z-Achse von C_i) übertragen werden. Bei Anwendung tangentialer Kräfte „rutscht" der Kontaktpunkt einfach weg.

$$F_c = [0, 0, f, 0, 0, 0]^t = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot f = B_c \cdot \vec{f_c} \qquad FC_c = \{f \geq 0\}$$

- **Punktkontakt mit (Haft-)Reibung** Hat man zusätzlich Haftreibung, können auch tangentiale Kräfte ausgeübt werden. Deren maximaler Betrag

hängt nach dem Coulomb-Gesetz linear mit der Kraft in Normalenrichtung zusammen (Abb. 9):

$$\|\vec{f_t}\| \leq \mu f_n \,,$$

wobei μ der (Haft-)reibungskoeffizient ist und von den Materialien der Objekte in Kontakt abhängt. Die Menge der übertragbaren Kräfte liegt damit innerhalb eines Kegels, dessen Achse mit der Normalenrichtung übereinstimmt und dessen Öffnungswinkel $\alpha = \arctan \mu$ ist. Damit ergibt sich:

$$F_c = [f_1, f_2, f_3,\, 0, 0, 0]^t = \begin{pmatrix} 1&0&0 \\ 0&1&0 \\ 0&0&1 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = B_c \cdot \vec{f_c}$$

$$FC_c = \{\vec{f_c} \in \mathbb{R}^3 \,|\, 0 \leq \sqrt{f_1^2 + f_2^2} \leq \mu \vec{f_3}\}$$

- **Soft-Finger-Kontakt** Realistischer ist ein Kontakt, der auch ein Drehmoment entlang der Normalenrichtung ausüben kann, und daher ein einfaches Modell für einen Flächenkontakt darstellt. Der maximale Betrag des Drehmoments ist wieder linear abhängig von der Normalenkraft ($|\tau| = |f_4| \leq \gamma f_3$), wobei γ der torsionale Reibungskoeffizient ist.

$$F_c = [f_1, f_2, f_3,\, 0, 0, f_4]^t = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = B_c \cdot \vec{f_c}$$

$$FC_c = \{\vec{f_c} \in \mathbb{R}^4 \,|\, 0 \leq \sqrt{f_1^2 + f_2^2} \leq \mu f_3,\ |f_4| \leq \gamma f_3\}$$

Ein Kontaktmodell wird also durch Angabe einer Basis $B_c \in \mathbb{R}^{p \times m}$ von m linear unabhängigen Wrench-Komponenten $(f_1, \ldots, f_m)$ sowie eines Reibungskegels FC_c dargestellt, wobei FC_c die Menge der ausübbaren Kräfte durch Angabe einer Reibungsbedingung weiter einschränkt. Da diese Menge konvex ist, gilt folgende Bedingung für die Kräfte:

$$\forall \vec{f_1}, \vec{f_2} \in FC_c \quad \forall \alpha, \beta > 0 \quad : \quad \alpha \vec{f_1} + \beta \vec{f_2} \in FC_c$$

2.1.2 Die Greif-Matrix

Wir fassen die (statischen) Komponenten eines Griffs nun geeignet zusammen. Seien also eine Objektkonfiguration T_{po} und eine Menge von k Kontakten mit Koordinatensystemen C_i, Basen B_{c_i} und Reibungskegeln FC_{c_i} gegeben. Die Wrenches an den einzelnen Kontaktpunkten sind also gegeben durch:

$$F_{c_i} = B_{c_i} \vec{f}_{c_i} \,.$$

Um den Netto-Wrench zu bestimmen, der im Objekt-Schwerpunkt O angreift, müssen wir die Wrenches in das Objekt-KS transformieren und summieren:

$$F_o = \sum_{i=1}^{k} Ad^t_{T_{oc_i}^{-1}} F_{c_i} = \sum_{i=1}^{k} \begin{pmatrix} R_{oc_i} & 0 \\ \hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{pmatrix} B_{c_i} \vec{f}_{c_i} \qquad \vec{f}_{c_i} \in FC_{c_i}$$

Wir definieren die Abbildung von Kontaktkräften $\vec{f_c}$ auf den Objekt-Wrench:

$$G_i := Ad^t_{T^{-1}_{oc_i}} B_{c_i}$$

und erhalten:

$$F_o = \sum_{i=1}^{k} G_i \vec{f_{c_i}} = [G_1, \ldots, G_k] \cdot \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix} = G \cdot \vec{f_c} \qquad \vec{f_c} \in FC$$

wobei

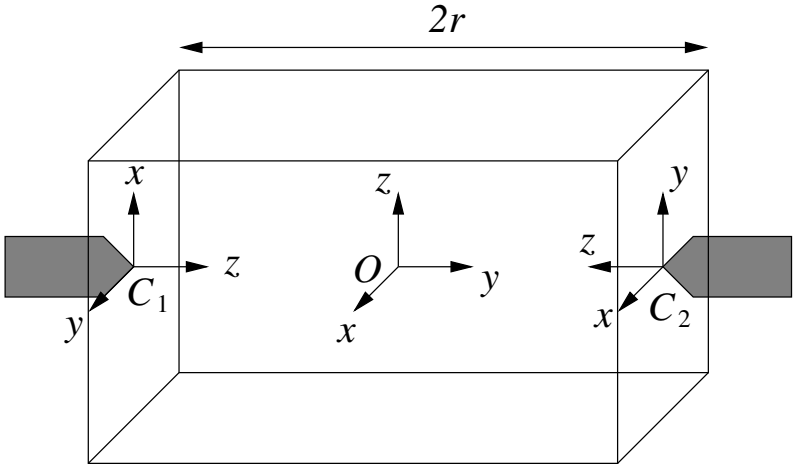
$$\vec{f_c} = (\vec{f_{c_1}}, \ldots, \vec{f_{c_k}}) \in \mathbb{R}^m \qquad m = \textstyle\sum_{i=1}^{k} m_i$$
$$FC = FC_{c_1} \times \cdots \times FC_{c_k} \subset \mathbb{R}^m$$

Definition 2.1 Die Matrix $G = [Ad^t_{T^{-1}_{oc_1}} B_{c_1}, \ldots, Ad^t_{T^{-1}_{oc_k}} B_{c_k}]$ heißt Greifmatrix. Ein Griff ist damit vollständig definiert durch Angabe der Greifmatrix G und der Menge $FC \subset \mathbb{R}^m$ der gültigen Kontaktkräfte. Die Menge der ausübbaren Wrenches auf das Objekt ist damit definiert durch

$$F_o = G\vec{f_c} \quad \text{mit} \quad \vec{f_c} \in FC\,.$$

Bemerkung: Diese Definition von G erlaubt es, unterschiedliche Reibungsmodelle für die verschiedenen Kontakte C_i anzunehmen. Die Kontaktkräfte innerhalb der Reibungskegel können im Prinzip beliebig groß werden – in der Realität benötigt man daher auch noch eine Begrenzung der Kontaktkraft f_n nach oben. Diese ist letztlich durch die Mechanik der Finger und die maximal ausübbaren Drehmomente in den Gelenken bestimmt.

Beispiel 2.2 (Griff einer Box)



Wir betrachten den abgebildeten Griff einer Box mit zwei Punktkontakten mit Rei-

bung.

$$F_o = \sum Ad^t_{T^{-1}_{oc_i}} F_{c_i} = \sum Ad^t_{T^{-1}_{oc_i}} B_{c_i} \vec{f}_{c_i}$$

$$p_{oc_1} = (0, -r, 0)^t \quad p_{oc_2} = (0, r, 0)^t \qquad B_{c_i} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^t$$

$$R_{oc_1} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad R_{oc_2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \qquad G = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ -r & 0 & 0 & 0 & r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r & 0 & -r & 0 & 0 \end{pmatrix}$$

$$FC_{c_i} = \{f \mid \sqrt{f_x^2 + f_y^2} \leq \mu_{c_i} f_z\}$$

Der Griff kann also beliebige Wrenches auf die Box ausüben, ausser Drehmomenten entlang der Körper-y-Achse. Dazu wäre mindestens ein Soft-Fingerkontakt notwendig. Natürlich müssen die Kräfte auch die Reibungsbedingung erfüllen, so dass die Normalenkraft groß genug sein muss, um entsprechende Kräfte in der x-z-Ebene zu erzeugen.

2.2 Bewertung von Griffen

2.2.1 Force Closure

Eine wichtige Eigenschaft eines Griffes ist, externen Kräften entgegenwirken zu können, um das Objekt zu halten, bzw. direkt beliebige Kräfte auf das Objekt auszuüben, um dessen Lage zu verändern. Dabei müssen die aufgebrachten Kräfte immer innerhalb der Reibungskegel liegen, damit das Objekt nicht wegrutscht.

Definition 2.3 (Force Closure) Ein Griff (G, FC) heißt force-closure, wenn er jedem externen Wrench F_e widerstehen kann:

$$G(FC) = \mathbb{R}^p \qquad \Leftrightarrow \qquad \forall F_e \in \mathbb{R}^p \quad \exists \vec{f_c} \in FC \ : \ G\vec{f_c} = -F_e$$

Force-closure ist durch die Existenz interner Kräfte charakterisiert, die keinen Nettoeffekt auf das Objekt haben: $Gf_N = 0$ bzw. f_N ist Element des Nullraumes $\mathcal{N}(G)$. Eine Kontaktkraft f_N heißt interne Kraft, falls $f_N \in \mathcal{N}(G) \cap FC$. Liegt f_N sogar im Innern des Reibungskegels FC, heißt sie strikte interne Kraft.

Satz 2.4 Ein Griff ist force-closure, gdw. G surjektiv ist und es strikte interne Kräfte f_N gibt: $f_N \in \mathcal{N}(G) \cap int(FC) \neq \emptyset$.

Beweis: $\Leftarrow$: Wähle $F \in \mathbb{R}^p$. Da G surjektiv, existiert ein f'_c mit $F = Gf'_c$. f'_c liegt jedoch nicht unbedingt innerhalb der Reibungskegel. Durch Hinzunahme von internen Kräften kann man jedoch auch dies erreichen. Sei $f_N \in \mathcal{N}(G) \cap int(FC)$. Dann gilt:

$$\lim_{\lambda \to \infty} \frac{f'_c + \lambda f_N}{\lambda} = f_N \in int(FC)$$

Damit existiert ein $f_c = f'_c + \lambda f_N \in int(FC)$ mit $Gf_c = Gf'_c = F$.

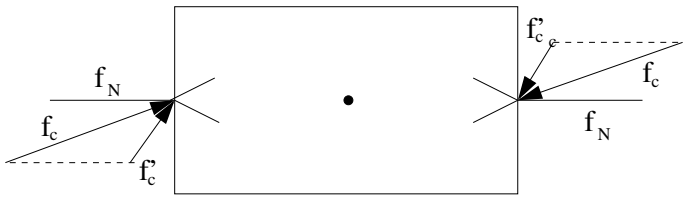


Abbildung 10: Mittels entsprechend hoher interner Kräfte kann jede Kontakt-Kraft in den Reibungskegel „verschoben“ werden.

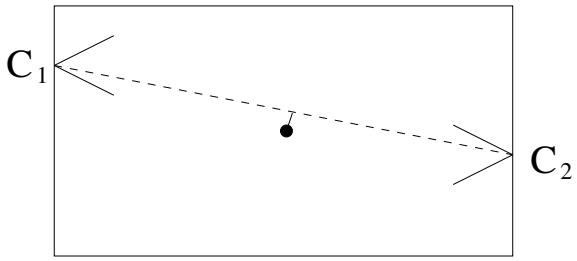
$\Rightarrow$: Der Griff sei force-closure. Die Surjektivität von G ist dann laut Definition erfüllt. Sei F beliebig. Dann existieren $f_1, f_2 \in int(FC)$ mit $Gf_1 = F$ und $Gf_2 = -F$. Es ist aber auch $f_N = f_1 + f_2 \in int(FC)$ und $Gf_N = F - F = 0$. $\square$

Antipodale Griffe

Satz 2.5 Ein planarer Griff mit zwei Punktkontakten mit Reibung ist force-closure gdw. die Verbindungslinie zwischen den Kontaktpunkten innerhalb der Reibungskegel liegt.

Satz 2.6 Ein Griff in 3D mit zwei Soft-Finger-Kontakten ist force-closure gdw. die Verbindungslinie zwischen den Kontaktpunkten innerhalb der Reibungskegel liegt.

Solche Griffe heißen antipodale Griffe (Griffe mit gegenüberliegenden Kontakten).
Beweis: Die Bedingung garantiert, dass sich die beiden Kontakte gegenüber liegen, so dass die Greifmatrix G surjektiv ist. Ausserdem sind Kontaktkräfte entlang der Verbindungslinie strikte interne Kräfte: Einerseits heben sich die Kraftanteile genau auf ($F_1 = -F_2$) und andererseits heben sich auch die Drehmomente auf, da der Hebel für beide derselbe ist – nämlich die kürzeste Verbindung von der Verbindungslinie zum Schwerpunkt.



$\square$

2.2.2 Form closure

Definition 2.7 Ein Griff heißt form-closure, wenn er bereits unter der Annahme von *reibungsfreien Punktkontakten* force-closure ist.

Lemma 2.8 form-closure $\Rightarrow$ force-closure

Form closure ist das strengere Kriterium und ein Griff der form-closure ist, ist auch force-closure mit jeder anderen Reibungsbedingung. Form closure benutzt lediglich die Griff-Geometrie in Form der Grasp-Matrix G, um den Griff zu bewerten. Die Greifmatrix von reibungsfreien Punktkontakten hat die Form:

$$G = \begin{pmatrix} n_{c_1} & \dots & n_{c_k} \\ p_{c_1} \times n_{c_1} & \dots & p_{c_k} \times n_{c_k} \end{pmatrix} \qquad n_{c_i} = R_{oc_i} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$
$$FC = \{ f \in \mathbb{R}^k \mid f_i \geq 0 \}$$

Für force-closure muss wieder gelten: $G(FC) = \mathbb{R}^p$, d.h.

$$\forall\, F_e \in \mathbb{R}^p \quad \exists\, f_i \geq 0 \;:\; G\vec{f_c} = \sum G_i f_i = -F_e$$

Die Spalten von G müssen den gesamten Raum $\mathbb{R}^p$ also *positiv* aufspannen, weil nur drückende Kontaktkräfte $f_i \geq 0$ erlaubt sind.

Lemma 2.9 Sei ein Griff G von reibungslosen Punktkontakten gegeben. Dann sind folgende Aussagen äquivalent:

- Der Griff ist form-closure.
- Die Spalten von G spannen den gesamten $\mathbb{R}^p$ *positiv* auf. (Es sind nur drückende Kontaktkräfte $f_i \geq 0$ erlaubt.)
- Die konvexe Hülle der Spalten von G enthält eine ε-Umgebung des Ursprungs.

Die konvexe Hülle $\mathrm{co}(G)$ ist definiert als

$$\mathrm{co}(G) = \{ F = \sum f_i G_i \mid \sum f_i \leq 1, f_i \geq 0 \}\,.$$

Sie enthält für je zwei Vektoren $v_i, v_j \in \mathrm{co}(G)$ auch deren Verbindungsstrecke: $\lambda v_i + (1-\lambda) v_j \in \mathrm{co}(G)$. Wenn eine ε-Umgebung von Null in $\mathrm{co}(G)$ enthalten ist, kann offenbar in jede Richtung des $\mathbb{R}^6$ ein Wrench positiv aus den G_i erzeugt werden. Dessen Länge ist zwar zunächst auf die ε-Umgebung beschränkt, jedoch erfüllt auch jeder mit $\lambda > 0$ skalierte Wrench die Reibungsbedingung $f_i \geq 0$.

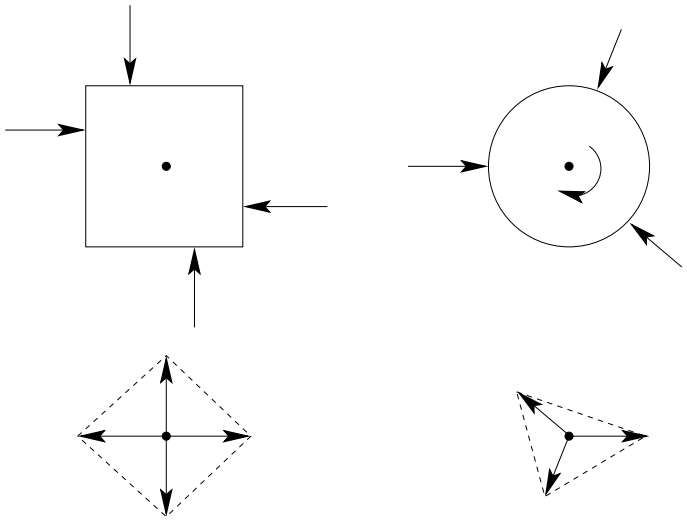


Abbildung 11: Beispiele von 2d-Griffen. Box: form-closure. Kreis: nicht form-closure, da zwar allen externen (linearen) Kräften widerstanden werden kann, aber keine Drehmomente erzeugt werden können. Kugelige Objekte können generell nicht form-closure sein, da die Kontaktkräfte immer auf den Schwerpunkt zeigen und somit kein Hebelarm bleibt, der ein Drehmoment erzeugen könnte.
Unten: Konvexe Hülle der Kräfte – der Ursprung liegt zwar in beiden Fällen innerhalb, aber Drehmomente sind dabei noch nicht berücksichtigt.

Satz 2.10 (Caratheodory) Wenn eine Menge $X = \{v_1, \ldots, v_k\} \subset \mathbb{R}^p$ den gesamten Raum $\mathbb{R}^p$ *positiv* aufspannt, gilt $k \geq p + 1$.

Man braucht also mindestens $p+1 = 7$ reibungsfreie Punktkontakte für einen form-closure-Griff. Nach einem Satz von Steinitz kommt man mit $2p = 12$ Kontakten aus, vorausgesetzt, es existiert überhaupt ein form-closure Griff. Für Kugeln gilt dies z.B. nicht, da alle Kontaktkräfte auf den Schwerpunkt zeigen und daher kein Drehmoment erzeugen können.

2.2.3 Qualitätsmaße

Die Qualität eines Griffes kann durch das Volumen bzw. die Form seines Wrench-Space beschrieben werden $\mathcal{W} \equiv \mathcal{G}(\mathcal{FC})$, d.h. die Menge aller auf das Objekt ausübbaren Kräfte. In der Praxis muss man allerdings die Reibungskegel FC_i noch beschränken, da die Normalkräfte, die über einen Fingerkontakt übertragen werden können, beschränkt sind. Typischerweise beschränkt man alle Normalkräfte auf Betrag Eins (es sind aber auch unterschiedliche Maximalwerte für jeden Fingerkontakt denkbar) und definiert den beschränkten Reibungskegel:

$$FC_{c_i}^{\max} = \{\vec{f} \in FC_{c_i} \mid f_i^n \leq 1\}$$

Je nachdem, ob die Gesamtsumme aller Normalkräfte oder jede einzelne Normalkraft beschränkt wird, ergibt sich eine andere Norm für die Kraftbegrenzung und man erhält die folgenden beiden Wrench-Spaces $\mathcal{W} = \{G\vec{f_c} \mid \vec{f_c} \in FC \text{ und } \|\vec{f_c}\| \leq 1\}$:

1. $\|\vec{f_c}\|_\infty = \max_i |f_i^n| \leq 1 \qquad \mathcal{W}_{L_\infty} = \text{co}\left(\bigoplus_{i=1}^{k} G(FC_{c_i}^{\max})\right)$

2. $\|\vec{f_c}\|_1 = \sum_i |f_i^n| \leq 1 \qquad \mathcal{W}_{L_1} = \text{co}\left(\bigcup_{i=1}^{k} G(FC_{c_i}^{\max})\right)$

$\mathcal{W}_{L_1} \subset \mathcal{W}_{L_\infty}$

Man beachte, dass in die Berechnung der Norm $\|\vec{f_c}\|$ immer nur die Normalenanteile $f_i^n = f_i^z$ eingehen.

Die konvexe Hülle von 3D-Kegeln ist leider nicht so einfach berechenbar, so dass man häufig linear approximiert:

$$FC_{c_i}^{\max} \approx \{\vec{f}_{c_i} = \sum_{j=1}^{n} \alpha_{ij}\vec{f}_{ij} \,|\, \alpha_{ij} \geq 0, \sum_{j=1}^{n} \alpha_{ij} \leq 1\}\,.$$

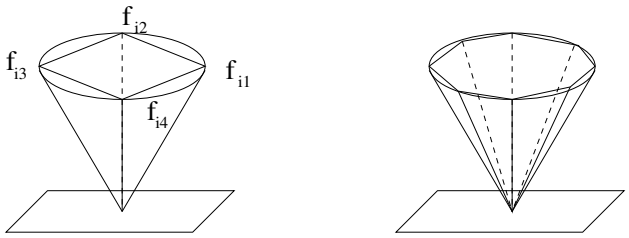


Abbildung 12: Approximation eines Reibungskegels durch eine mehrseitige Pyramide.

Die Kraftvektoren $\vec{f}_{ij}$ liegen auf dem Rand des Reibungskegels des i-ten Kontakts und approximieren ihn mittels einer Pyramide. Man wählt ihren Betrag so, dass der Normalenanteil $f_i^n = \langle f_{ij}, \hat{n}_i\rangle \hat{n}_i$ jeweils Betrag 1 hat – es kann also maximal eine Normalenkraft vom Betrag 1 ausgeübt werden. Jeder Kraftvektor $\vec{f}_{ij}$ erzeugt einen Wrench im Objekt-KS:

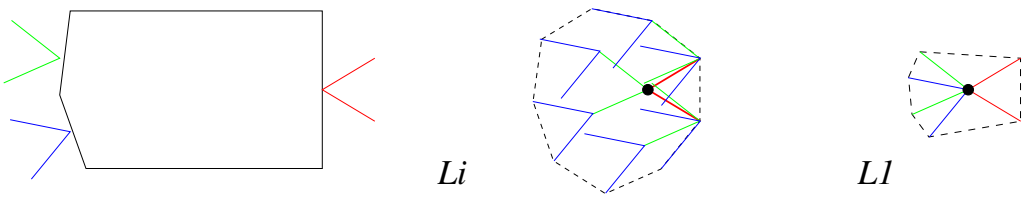
$$F_{ij}^o = Ad_{T_{oc_i}^{-1}}^t B_{c_i} \cdot \vec{f}_{ij}\,.$$

Diese Wrench-Vektoren werden nun zur Approximation des Wrench-Space durch ein Wrench-Polytop verwendet:

1. $\|\vec{f_c}\|_\infty = \max_i |f_i^n| \leq 1 \qquad \mathcal{W}_{L_\infty} = \text{co}\left(\bigoplus_{i=1}^{k} \{F_{i1}^o, \ldots, F_{in}^o\}\right)$

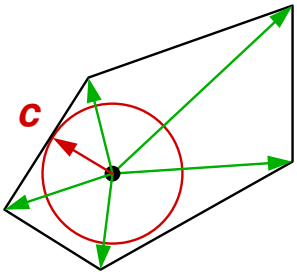
2. $\|\vec{f_c}\|_1 = \sum_i |f_i^n| \leq 1 \qquad \mathcal{W}_{L_1} = \text{co}\left(\bigcup_{i=1}^{k} \{F_{i1}^o, \ldots, F_{in}^o\}\right)$

Beispiel 2.11 Griff eines Rechtecks mit drei Punktkontakten mit Reibung.

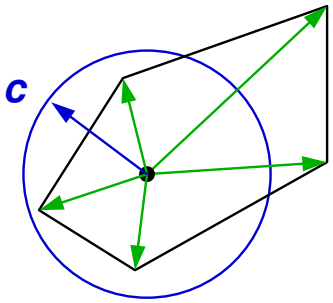


Das Wrench-Polytope bzw. den Wrench-Space $\mathcal{W}$ kann man nun nutzen, um die Qualität des Griffs zu bestimmen:

- Falls der Ursprung innerhalb des Wrench-Polytops liegt, ist der Griff force-closure.
- Bestimme den kleinsten Abstand vom Ursprung zum Rand von $\mathcal{W}$. Dies entspricht dem Radius ε der größten (6-dim.) Kugel um den Ursprung, die gerade noch in das Polytop passt. Die zugehörige Richtung, gibt den Wrench an, dem am schlechtesten widerstanden werden kann bzw. dem gerade noch widerstanden werden kann, wenn die Kontaktkräfte entsprechend der Norm beschränkt werden.



- Bestimme den mittleren Abstand vom Ursprung zu den Grenzflächen von $\mathcal{W}$



Probleme:

- Drehmomente hängen von der Wahl des Objekt-KS O ab. $\curvearrowright$ Benutze den Massenschwerpunkt.
- Drehmomente hängen von der Objektgröße ab. $\curvearrowright$ Skaliere mit mittlerer Objektgröße r und ersetze

$$Ad^t_{T^{-1}_{oc_i}} \quad \Longrightarrow \quad \begin{pmatrix} R_{oc_i} & 0 \\ \frac{1}{r}\hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{pmatrix}$$

- Wrench-Polytop ist konservative Approximation von echtem Wrench-Space $\mathcal{W}$ und abhängig von der Ausrichtung der approximierenden Pyramide.
- Die Anzahl der Wrench-Vektoren in der direkten Summe ($\bigoplus$) ist $N_{L_\infty} = n^k$ und in der Vereinigung ($\bigcup$) $N_{L_1} = n \cdot k$.
  Rechenaufwand für co($\cdot$): $\mathcal{O}(N \log N)$.
- Eine gute Approximation mit $n \geq 8$ Stützpunkten pro Reibungskegel braucht sehr lange. Wähle $n \approx 6$.

Wenn man sich nur für bestimmte Unterräume des 6-dim. Wrench-Raumes interessiert, z.B. weil man nur Kräfte oder nur Drehmomente ausüben will, kann man auch Projektionen der Wrench-polytope W auf diese Räume betrachten und die Maße dort auswerten.

Vortex-Demo

2.3 Die Greif-Bedingung

Bislang haben wir Griffe ohne Rücksicht auf die Finger selbst betrachtet und angenommen, dass Kontaktkräfte innerhalb der FC_i tatsächlich von den Fingern aufgebracht werden können. Nun wollen wir die Manipulierbarkeit des Objektes betrachten, d.h. die Frage welche Objektbewegungen von den Fingern tatsächlich erzeugt werden können. Dazu müssen wir die Hand-Kinematik einbeziehen.

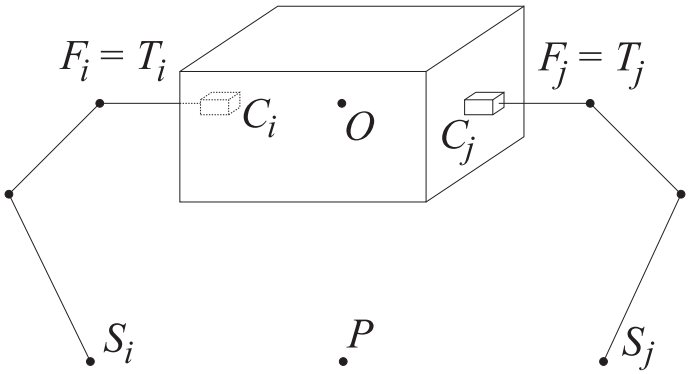


Abbildung 13: Kinematik eines mehrfingrigen Griffes

Wir betrachten folgende Koordinatensysteme (Abb. 13):

- in der Handfläche (Palm): P
- die Fingerbasis S_i für jeden Finger, fest relativ zu P
- das Objekt-KS O
- Kontakt-KS C_i, fest relativ zu O
- Finger-Kontakt-KS F_i im Kontakt, fest relativ zur Fingerspitze

Die Greifbedingung für einen Kontakt/Finger legt fest, entlang welcher Richtungen keine relative Bewegung zwischen Objekt und Finger stattfinden darf (kein Slippage). Für einen reibungsfreien Punktkontakt darf es z.B. keine Relativbewegung zwischen Objekt und Fingerspitze entlang der Kontakt-Normalenrichtung geben, da sonst der Kontakt verloren ginge. Dies kann durch die Bedingung $[0,0,1,0,0,0]^t \cdot V^b_{f_i c_i} = 0$ ausgedrückt werden. Im allgemeinen dürfen keine Verschiebungen entlang der Richtungen stattfinden, entlang derer auch Kräfte ausgeübt werden können. Für einen Kontakt mit Wrench-Basis B_{c_i} muss daher gelten:

$$B^t_{c_i} V^{c_i}_{f_i c_i} = 0\,. \qquad (*)$$

Wir drücken nun $V_{f_i c_i}^{c_i}$ in bekannten Größen aus. Für die Geschwindigkeiten nutzen wir die Transformationsgleichungen:

$$A \to B \to C: \qquad V_{ac}^{c} = Ad_{T_{bc}^{-1}} V_{ab}^{b} + V_{bc}^{c} \tag{1}$$

$$V_{ab}^{b} = -Ad_{T_{ba}} V_{ba}^{a} \tag{2}$$

und erhalten

$$F_i \to P \to C_i: \qquad V_{f_i c_i}^{c_i} \overset{(1)}{=} Ad_{T_{pc_i}^{-1}} V_{f_i p}^{p} + V_{pc_i}^{c_i} \overset{(2)}{=} -Ad_{T_{pc_i}}^{-1} Ad_{T_{pf_i}} V_{pf_i}^{f_i} + V_{pc_i}^{c_i}$$

$$P \to O \to C_i: \qquad V_{pc_i}^{c_i} \overset{(1)}{=} Ad_{T_{oc_i}^{-1}} V_{po}^{o} + \cancel{V_{oc_i}^{c_i}} = Ad_{T_{oc_i}^{-1}} V_{po}^{o}$$

und zusammen:

$$V_{f_i c_i}^{c_i} = -Ad_{T_{pc_i}}^{-1} Ad_{T_{pf_i}} V_{pf_i}^{f_i} + Ad_{T_{oc_i}^{-1}} V_{po}^{o}$$

Dies setzen wir nun in (*) ein:

$$B_{c_i}^{t} Ad_{T_{pc_i}}^{-1} \underbrace{Ad_{T_{pf_i}} V_{pf_i}^{f_i}}_{V_{pf_i}^{p}} = \underbrace{B_{c_i}^{t} Ad_{T_{oc_i}^{-1}}}_{(Ad_{T_{oc_i}^{-1}}^{t} B_{c_i})^t = G_i^t} V_{po}^{o} = G_i^t \cdot V_{po}^{o}$$

Die Tool-Geschwindigkeit $V_{pf_i}^{p}$ relativ zur Handfläche wird durch die Jacobi-Matrix beschrieben:

$$V_{pf_i}^{p} = J_{pf_i}^{p}(\theta_i) \cdot \dot{\theta}_i \,,$$

wobei θ_i der Gelenkwinkel-Vektor des i-ten Fingers ist. Wir erhalten also:

$$B_{c_i}^{t} Ad_{T_{pc_i}}^{-1} \cdot J_{pf_i}^{p}(\theta_i) \cdot \dot{\theta}_i = G_i^t \cdot V_{po}^{o} \; (= B_{c_i}^{t} V_{pc_i}^{c_i})$$

Diese Gleichung hat eine sinnvolle Interpretation: Die rechte Seite ist die Geschwindigkeit $V_{pc_i}^{c_i}$ des i-ten Kontakt-KS relativ zur Handfläche, maskiert mit den relevanten Bewegungsrichtungen B_{c_i}. Die linke Seite gibt die Tool-Geschwindigkeit F_i relativ zu P an, transformiert diese mittels der Adjungierten $Ad_{T_{pc_i}}^{-1}$ ins C_i-KS und maskiert sie wiederum mit B_{c_i}.

Wir fassen die Bedingungen für alle k Finger zusammen:

Definition 2.12 Die Hand-Jacobi-Matrix für k Kontakte und Finger (mit n_i DOFs) ist definiert als:

$$J_H(\theta, T_{po}) = \begin{pmatrix} B_{c_1}^{t} Ad_{T_{pc_1}}^{-1} J_{pf_1}^{p}(\theta_1) & & \\ & \ddots & \\ & & B_{c_k}^{t} Ad_{T_{pc_k}}^{-1} J_{pf_k}^{p}(\theta_k) \end{pmatrix} \in \mathbb{R}^{m \times n} \qquad \begin{matrix} n = \sum n_i \\ m = \sum m_i \end{matrix}$$

$$= \begin{pmatrix} B_{c_1}^{t} Ad_{T_{s_1 c_1}}^{-1} J_{s_1 f_1}^{s_1}(\theta_1) & & \\ & \ddots & \\ & & B_{c_k}^{t} Ad_{T_{s_k c_k}}^{-1} J_{s_k f_k}^{s_k}(\theta_k) \end{pmatrix}$$

Damit können wir die Greif-Bedingung für die gesamte Hand schreiben als:

$$J_H(\theta, T_{po}) \cdot \dot{\theta} = G^t \cdot V_{po}^{o} \qquad \in \quad \mathbb{R}^m \,. \tag{2.1}$$

Bemerkung: Die Greif-Bedingung erfasst nur Kontakte mit Fingerspitzen und es kommen auch nur solche Finger vor, die überhaupt Kontakt haben. Die Vektoren $\vec{v} = J_H \cdot \dot{\theta} = G^t \cdot V_{po}^{o} \in \mathbb{R}^m$ fassen die Relativ-Geschwindigkeiten aller Kontakt-KS relativ zur Palm ($V_{pc_i}^{b}$) zusammen und betrachten dabei nur die relevanten Richtungen (maskiert mit B_{c_i}), d.h. $\vec{v} = (B_{c_1}^{t} V_{pc_1}^{c_1}, \ldots, B_{c_k}^{t} V_{pc_k}^{c_k})$.

2.4 Manipulierbarkeit

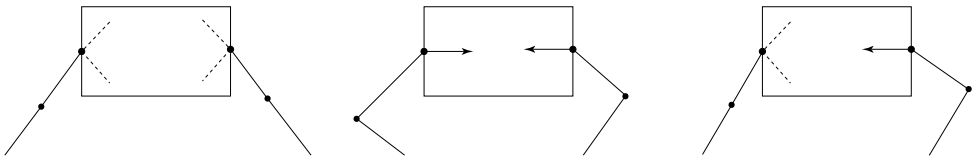
Definition 2.13 Ein mehrfingriger Griff $\big(G, FC, J_H(\theta, T_{po})\big)$ heißt manipulierbar, falls jede Objektbewegung $V_{po}^o \in \mathbb{R}^p$ durch entsprechende Gelenkbewegungen $\dot{\theta}$ erzeugt werden kann, so dass Gleichung (2.1) erfüllt ist.

Satz 2.14 Ein Griff ist manipulierbar gdw. $\mathcal{R}(G^t) \subseteq \mathcal{R}(J_H\big(\theta, T_{po}\big))$.

Beweis:
Das Bild von G^t ist eine Teilmenge des Bildes von J_H, d.h. alle Vektoren $\vec{v} \in \mathbb{R}^m$, die G^t erzeugen kann, kann J_H ebenfalls erzeugen.

$\Rightarrow$: Sei $\vec{v} \in \mathcal{R}(G^t)$, d.h. es gibt V_{po}^o mit $\vec{v} = G^t \cdot V_{po}^o$. Da der Griff manipulierbar ist, gibt es dazu auch ein $\dot{\theta}$ mit $\vec{v} = G^t \cdot V_{po}^o = J_H \cdot \dot{\theta} \in \mathcal{R}(J_H)$.
$\Leftarrow$: Sei V_{po}^o beliebig, und sei $\vec{v} = G^t \cdot V_{po}^o \in \mathbb{R}^m$ der zug. Bildvektor. Da $\vec{v}$ nach Voraussetzung auch ein Bildvektor von J_H ist, gibt es also auch ein $\dot{\theta}$ mit $\vec{v} = J_H \cdot \dot{\theta}$.
□
Bemerkungen:

- Manipulierbarkeit erfordert nicht, dass J_H invertierbar ist. Vielmehr können viele Gelenkbewegungen $\dot{\theta}$ existieren, die dieselbe Objektbewegung V_{po}^o hervorrufen.
- Falls $n > m$ (Redundanz), existieren z.B. *interne Bewegungen* $\dot{\theta} \in \mathcal{N}(J_H)$, die keine Objekt-Bewegung (in relevanten Richtungen B_{c_i}) erzeugen.
- force-closure und Manipulierbarkeit sind unabhängige Konzepte:



force closure,
nicht manipulierbar

nicht force closure,
manipulierbar

nicht force closure,
nicht manipulierbar

- Die Forderung (*) $B_{c_i}^t V_{f_i c_i}^{c_i} = 0$ ist sehr streng. Sie erlaubt keinerlei Objekt-Finger-Bewegung entlang der Richtungen, in die auch Kontaktkräfte übertragen werden können. Insbesondere also kein Gleiten auf der Kontaktoberfläche (bei Coloumb-Reibung) und keine Rotation um die Kontaktnormale (bei Softfingerkontakt). Wichtig ist eigentlich nur, dass der Kontakt nicht verloren geht. Dies wird bereits durch die folgende Bedingung sichergestellt:

$$[0, 0, 1, 0, 0, 0]^t V_{f_i c_i}^{c_i} = 0\,. \tag{2.2}$$

2.5 Strukturelle Kräfte

Wir können mittels der Hand-Jacobi-Matrix auch Kontakt-Kräfte in Gelenk-Kräfte umrechnen. Die Herleitung erfolgt mittels der geleisteten Arbeit W:

$$\int_{t_1}^{t_2} \dot{\theta}_i^t \tau_i \, dt = W = \int_{t_1}^{t_2} \left(V_{pc_i}^{c_i}\right)^t \cdot F_{c_i} \, dt \quad \text{für alle } t_1 \leq t_2 \in \mathbb{R}$$

$$\Leftrightarrow \quad \begin{aligned} \dot{\theta}_i^t \cdot \tau_i &= \left(Ad_{T_{oc_i}^{-1}} V_{po}^o\right)^t \cdot B_{c_i} \vec{f}_{c_i} = \left(B_{c_i}^t Ad_{T_{oc_i}^{-1}} V_{po}^o\right)^t \cdot \vec{f}_{c_i} \\ &= \left(G_i^t V_{po}^o\right)^t \cdot \vec{f}_{c_i} = \left(J_H^i \cdot \dot{\theta}_i\right)^t \cdot \vec{f}_{c_i} = \dot{\theta}_i^t \cdot \left(J_H^i\right)^t \cdot \vec{f}_{c_i} \end{aligned} \quad \text{für alle } \dot{\theta}$$

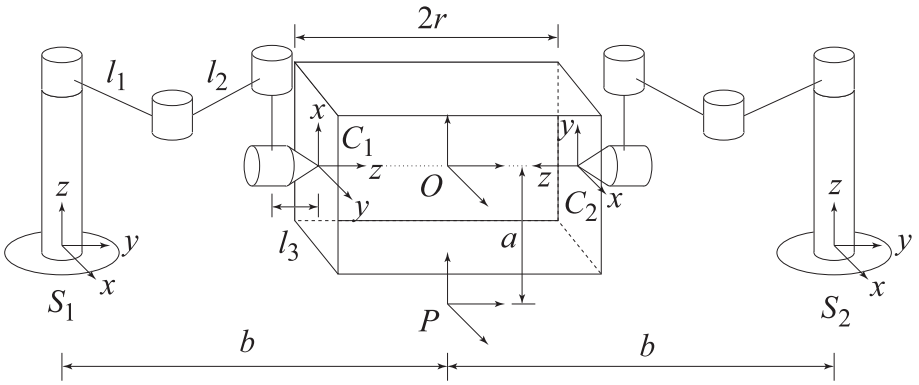
Fassen wir wieder alle Finger $i = 1, \ldots, k$ zusammen, erhalten wir:

$$\tau = J_H^t \cdot \vec{f}_c \qquad \vec{f}_c \in FC$$

Bemerkungen:

- Kontaktkräfte $\vec{f}_c^s \in \mathcal{N}(J_H^t)$ im Nullraum von J_H^t heißen strukturelle Kräfte. Diese können nicht durch die Gelenk-Drehmomente τ aktiv erzeugt werden, sondern werden passiv von der Hand-Konstruktion aufgebracht. Der Griff kann in diesem Fall trotzdem manipulierbar sein.

Beispiel 2.15 (Zwei SCARAs greifen eine Box) mittels Soft-Finger-Kontakten.



Zur Greif-Matrix G aus Beispiel 2.2 kommt noch die Soft-Finger-Komponente hinzu:

$$G = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -r & 0 & 0 & 0 & 0 & r & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & r & 0 & 0 & -r & 0 & 0 & 0 \end{pmatrix} \quad B_{c_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \operatorname{rang}(G) = 6$$

Die Jacobi-Matrix des SCARA-Manipulators ist aus Beispiel 1.21 bekannt. Zur Vereinfachung setzen wir die Länge der Fingerspitze $l_3 = 0$.

$$J_{s_i f_i}^{s_i} = \begin{pmatrix} 0 & l_1 \cos\theta_1 & l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & 0 \\ 0 & l_1 \sin\theta_1 & l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Wir betrachten nur die (einfache) Lage des Objektes in der Zeichnung – für eine allgemeine Lage würde die Greif-Bedingung sehr komplex werden.

$$R_{po} = \mathbf{1} \qquad \vec{p}_{po} = [0, 0, a]^t$$

Die Adjungierten $Ad_{T_{s_i c_i}^{-1}}$ lesen wir direkt ab: Mit $\vec{p}_{s_1 c_1} = [0, b-r, a]^t$ und $\vec{p}_{s_2 c_2} = [0, r-b, a]^t$ folgt:

$$R_{s_1 c_1}^t = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad -\hat{p}_{s_1 c_1} = \begin{pmatrix} 0 & a & r-b \\ -a & 0 & 0 \\ b-r & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad Ad_{T_{s_1 c_1}}^{-1} = \begin{pmatrix} R_{s_1 c_1}^t & \begin{bmatrix} b-r & 0 & 0 \\ 0 & a & r-b \\ -a & 0 & 0 \end{bmatrix} \\ 0 & R_{s_1 c_1}^t \end{pmatrix}$$
$$\vec{p}_{s_1 c_1} = [0, +b-r, a]^t$$

$$R_{s_2 c_2}^t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \quad -\hat{p}_{s_2 c_2} = \begin{pmatrix} 0 & a & b-r \\ -a & 0 & 0 \\ r-b & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad Ad_{T_{s_2 c_2}}^{-1} = \begin{pmatrix} R_{s_2 c_2}^t & \begin{bmatrix} 0 & a & b-r \\ r-b & 0 & 0 \\ a & 0 & 0 \end{bmatrix} \\ 0 & R_{s_2 c_2}^t \end{pmatrix}$$
$$\vec{p}_{s_2 c_2} = [0, -b+r, a]^t$$

Die beiden Teile $J_i = B_{c_i}^t \cdot Ad_{T_{s_i c_i}}^{-1} J_{s_i f_i}^{s_i}(\theta_i)$ der Hand-Jacobi-Matrix $J_H = \begin{pmatrix} J_1 & \\ & J_2 \end{pmatrix}$ sehen nun wie folgt aus:

$$J_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ -b+r & -b+r+l_1 c_1 & -b+r+l_1 c_1 + l_2 c_{12} & 0 \\ 0 & l_1 s_1 & l_1 s_1 + l_2 s_{12} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$J_2 = \begin{pmatrix} +b-r & +b-r+l_1 c_1 & +b-r+l_1 c_1 + l_2 c_{12} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -l_1 s_1 & -l_1 s_1 - l_2 s_{12} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Der Griff ist force-closure, weil er antipodal ist und Soft-Finger-Kontakte benutzt. Er ist aber nicht manipulierbar, da

$$G^t \cdot [0, 0, 0,\ 0, 1, 0]^t = [0, 0, 0, 1,\ 0, 0, 0, -1]^t \ \notin \ \mathcal{R}(J_H)\,.$$

Mit Coloumb-Kontakten wäre er nicht force-closure, aber manipulierbar!

Der Nullraum $\mathcal{N}(J_H^t)$ wird von folgenden strukturellen Kräften aufgespannt:

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

Diese entsprechen den Objekt-Wrenches $F_o = G\vec{f}_c^s$:

$$F_o = [0, 0, 0,\ 0, \pm 1, 0]^t \quad \text{Drehmoment um Objekt-}y\text{-Achse}$$
$$F_o = [0, 0, 0,\ \pm 1, 0, 0]^t \quad \text{Drehmoment um Objekt-}x\text{-Achse}$$

Diese Drehmomente können vom Griff nicht *aktiv* erzeugt aber mittels der Soft-Finger-Kontakte und der Konstruktion *passiv* aufgebracht werden, um externen Störungen zu widerstehen.

2.6 Zusammenfassung

	Finger		Kontakt		Objekt
Geschwindigkeiten	$\dot{\theta}$	$\xrightarrow{J_H}$	$B_c^t V_{pc}^c$	$\xleftarrow{G^t}$	V_{po}^o
Kräfte	τ	$\xleftarrow{J_H^t}$	$\vec{f}_c$	$\xrightarrow{G}$	F_o

Eigenschaft	Beschreibung	in Formeln
force closure	jedem Wrench F_e kann (passiv) widerstanden werden	$G(FC) = \mathbb{R}^p$
manipulierbar	jede Objektbewegung kann (aktiv) erzeugt werden – entlang der Kraftrichtungen	$\mathcal{R}(G^t) \subseteq \mathcal{R}(J_H)$
interne Kraft	Kontaktkräfte $\vec{f}_N$, die keinen Nettoeffekt auf das Objekt haben	$\vec{f}_N \in \mathcal{N}(G) \cap int(FC)$
interne Bewegung	Gelenk-Bewegungen $\dot{\theta}$, die keine Objekt-Bewegung V_{po}^o erzeugen	$\dot{\theta}_N \in \mathcal{N}(J_H)$
strukturelle Kraft	Kontaktkräfte $\vec{f}_c^s$, die ohne Gelenk-Drehmomente τ erzeugt werden	$\vec{f}_c^s \in \mathcal{N}(J_H^t)$

2.7 Optimierung von Kontaktkräften $\vec{f}_c$

Wir suchen im folgenden optimale Griffe. Man kann sich verschiedene Optimalitätskriterien und Nebenbedingungen vorstellen:

- stabiler Griff: $\vec{f}_c \in FC$
- force-closure oder manipulierbarer Griff
- Begrenzung der Kontaktkräfte: $\|\vec{f}_c\| < M_f$ (bei zerbrechlichen Objekten)
- Begrenzung der Gelenk-Drehmomente: $\tau_i^L \leq \tau_i \leq \tau_i^U$
- minimale Kontaktkräfte: min $\|\vec{f}_c\|$
- minimale Gelenk-Drehmomente: min $\|\tau\|$
- hohe Griffsicherheit: große Entfernung zum Rand des Reibungskegels FC

Eine Lösung erfolgt i.d.R. numerisch, wobei das Hauptproblem die nichtlinearen (quadratischen) Reibungsbedingungen $\vec{f}_c \in FC$ sind. Ihre Formulierung mittels linearer Matrixungleichungen (LMI) ermöglicht die Anwendung effizienter (polynomiell konvergenter) Optimierungsverfahren (z.B. Interior-Point-Optimierung).

Einschub: Lineare Matrixungleichungen (LMI)

Definition 2.16 Sei $S = S^t \in \mathbb{R}^{n \times n}$ eine symmetrische Matrix. Die Schreibweisen

$$\begin{array}{ll} S > 0 & \text{„}S \text{ ist positiv [definit]“ und} \\ S \geq 0 & \text{„}S \text{ ist positiv semidefinit“} \end{array}$$

verallgemeinern die entsprechenden skalaren Ungleichungen und stehen für

$$\begin{array}{lll} S > 0 & \Leftrightarrow & \vec{x}^t S \vec{x} > 0 \quad \text{für alle } \vec{x} \neq 0 \\ & \Leftrightarrow & \text{alle (reellen) Eigenwerte } \lambda_i \text{ von } S \text{ sind positiv: } \lambda_i > 0 \\ S \geq 0 & \Leftrightarrow & \vec{x}^t S \vec{x} \geq 0 \quad \text{für alle } \vec{x} \neq 0 \\ & \Leftrightarrow & \text{alle } \lambda_i \geq 0 \end{array}$$

Definition 2.17 Eine lineare Matrixungleichung ist eine Gleichung der Form

$$P(x) = S_0 + x_1 S_1 + x_2 S_2 + \cdots + x_k S_k > 0\,,$$

wobei die S_i symmetrisch sind. Die x_i sind gesuchte *skalare* Variablen, um die Ungleichung zu erfüllen. Beachten Sie, dass die x_i linear in die Ungleichung eingehen.

Satz 2.18 Sei eine LMI der Form $P(x) = S_0 + \sum_{i=1}^{m} x_i S_i$ und eine beliebige affine Transformation $x = Az + b$ mit $A \in \mathbb{R}^{m\times l}$ und $b \in \mathbb{R}^m$ gegeben. Dann ist $P'(z) := P(Az + b)$ wieder eine LMI in der neuen Variablen $z \in \mathbb{R}^l$.

Beweis:

$$P'(z) = S_0 + \sum_{i=1}^{m} S_i \Big(\underbrace{\sum_{j=1}^{l} a_{ij} z_j + b_i}_{x_i = A_i z + b_i} \Big) = \underbrace{S_0 + \sum_{i=1}^{m} b_i S_i}_{S'_0} \quad + \quad \sum_{j=1}^{l} z_j \Big(\underbrace{\sum_{i=1}^{m} a_{ij} S_i}_{S'_j} \Big)$$

Die S'_j sind symmetrisch, da sie Summen von symmetrischen Matrizen sind. □
Ende Einschub.

Satz 2.19 Die Reibungskegel-Bedingungen FC können als LMI ausgedrückt werden.

Beweis:

- reibungsfreier Punktkontakt: $P_i = f \geq 0$
- Punktkontakt mit Reibung: $P_i = \begin{pmatrix} \mu f_z + f_x & f_y \\ f_y & \mu f_z - f_x \end{pmatrix} \geq 0$
- Softfinger-Kontakt: $P_i = \begin{pmatrix} f_z & 0 & 0 & \alpha f_x \\ 0 & f_z & 0 & \alpha f_y \\ 0 & 0 & f_z & \beta f_\tau \\ \alpha f_x & \alpha f_y & \beta f_\tau & f_z \end{pmatrix} \geq 0 \qquad \alpha = \mu^{-1} \quad \beta = \gamma^{-1}$

Wir betrachten lediglich den Punktkontakt mit Reibung. Für die Eigenwerte von P_i gilt: $\lambda_\pm = \mu f_z \pm \sqrt{f_x^2 + f_y^2}$. Die Bedingung $\lambda_- \geq 0$ ist damit äquivalent zur Reibungsbedingung $\sqrt{f_x^2 + f_y^2} \leq \mu f_z$. Ausserdem kann $P_i \geq 0$ als LMI geschrieben werden:

$$P_i = f_z \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix} + f_x \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + f_y \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \geq 0$$

□
Das simultane Bestehen der LMI $P_i \geq 0$ für alle k Kontakte kann in einer einzelnen LMI zusammengefasst werden:

$$\vec{f_c} \in FC \qquad \Leftrightarrow \qquad P(\vec{f_c}) = \begin{pmatrix} P_1(\vec{f}_{c_1}) & & & \\ & P_2(\vec{f}_{c_2}) & & \\ & & \ddots & \\ & & & P_k(\vec{f}_{c_k}) \end{pmatrix} \geq 0$$

Da jeder Matrixblock P_i eine Linearkombination aus mehreren (drei) Basismatrizen ist, ergibt sich eine entsprechende Darstellung auch für die gesamte Matrix:

$$P(\vec{f_c}) = f^z_{c_1} \begin{pmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + f^x_{c_1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \cdots + f^y_{c_k} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Bemerkungen:

- Die triviale Lösung $\vec{f_c} = 0$ erfüllt die Reibungsbedingung.
- Wenn die Kontaktkräfte $\vec{f_c}$ strikt im Inneren der Reibungskegel liegen sollen, fordert man $P_i > 0$ anstatt $P_i \geq 0$ – die triviale Lösung $\vec{f_c} = 0$ gibt es dann nicht mehr.
- In die LMI $P(\vec{f_c})$ können verschiedene Kontaktmodelle eingehen.

Skalare Ungleichungen können durch geeignete Diagonalmatrizen in LMI-Form ausgedrückt werden:

$$m \leq x \leq M \qquad \Leftrightarrow \qquad \begin{matrix} -m + x \geq 0 \\ M - x \geq 0 \end{matrix} \qquad \Leftrightarrow \qquad \begin{pmatrix} -m & \\ & M \end{pmatrix} + \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} x \geq 0$$

Z.B. können wir die ausübbaren Drehmomente auf das Intervall

$$\begin{aligned} & \tau_{\min} \leq \tau = J_H^t \vec{f_c} \leq \tau_{\max} \\ \Leftrightarrow \qquad & \tau_i^{\min} \leq \sum_{j=1}^{m} J_{ji}^H f_j \leq \tau_i^{\max} \end{aligned}$$

beschränken, indem wir zusätzlich die LMI $T = \operatorname{diag}(T_1, \ldots, T_n) \geq 0$ berücksichtigen, wobei:

$$T_i = \begin{pmatrix} -\tau_i^{\min} + \sum J_{ji}^H f_j & \\ & \tau_i^{\max} - \sum J_{ji}^H f_j \end{pmatrix} \geq 0\,.$$

Analog können wir die Kontaktkräfte nach oben beschränken, z.B.

$$\|\vec{f_c}\|_1 = \sum f^n_{c_i} \leq 1 \qquad \Leftrightarrow \qquad 1 - \sum f^n_{c_i} \geq 0\,.$$

Wir können nun verschiedene Optimierungsprobleme mittels der LMIs beschreiben.

Force-closure Falls G surjektiv ist, bestimme eine Basis $N := [n_1, \cdots, n_l] \in \mathbb{R}^{m \times l}$ des Nullraumes von G. Jeder Vektor $\vec{f_N} \in \mathcal{N}(G) \subset \mathbb{R}^m$ kann dann als $\vec{f_N} = Nz$ mit $z \in \mathbb{R}^l$ geschrieben werden. Falls eine Lösung z der resultierenden LMI gefunden werden kann (feasibility problem), ist der Griff force-closure. Die Idee ist also, nur potentielle innere Kräfte $\vec{f_N} \in \mathcal{N}(G)$ zu betrachten, und die Reibungsbedingungen (LMI) für diese umzuschreiben.

Minimierung der Kontaktkräfte Bei gegebenem externen Wrench F_e werden minimale Kontaktkräfte $\vec{f_c}$ gesucht, die dem Wrench noch widerstehen können. Wir bestimmen zunächst eine beliebige Lösung $\vec{f_0}$ der Gleichung

$$G\vec{f_c} = -F_e\,.$$

Wir können aber auch beliebige Kräfte $\vec{f_N} \in \mathcal{N}(G)$ addieren:

$$\mathcal{M} = \{\vec{f_c} \in \mathbb{R}^m \,|\, G\vec{f_c} = -F_e\} = \{\vec{f_0} + Nz \,|\, z \in \mathbb{R}^l\}$$

wobei $N \in \mathbb{R}^{m\times l}$ wieder eine Basis des Nullraumes von G ist. Wir können nun die LMI wieder umschreiben: $P(\vec{f_c}) = P(\vec{f_0} + Nz)$, so dass die Reibungsbedingungen bzgl. der $z \in \mathbb{R}^l$ ausgedrückt sind. Wir minimieren die Summe der Fingerkräfte:

$$\min \Psi(\vec{f_c}) = w^t \cdot \vec{f_c} \qquad \vec{f_c} \in FC$$

wobei der Vektor $w = [w_1, \ldots, w_k]^t$ die Gewichtung der einzelnen Kraftkomponenten angibt, z.B.: $w_i = [0, 0, 1]^t$ für Punktkontakte mit Reibung, um nur die Normalkräfte zu berücksichtigen. Das Optimierungsproblem können wir also schreiben als:

$$\min \Psi(z) = \Psi(\vec{f_0} + Nz) = \underbrace{w^t \cdot \vec{f_0}}_{const} + (w^t N) \cdot z \qquad P(z) > 0\,.$$

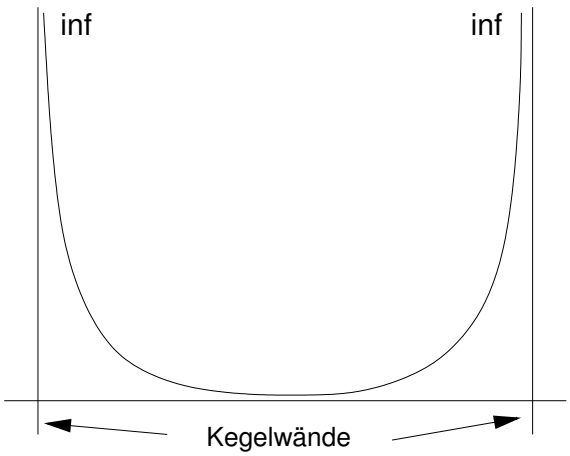
Die optimale Lösung einer solchen linearen Funktion unter konvexen Nebenbedingungen (FC) liegt immer auf dem Rand der Reibungskegel. Daher: möglichst großer Abstand zum Rand der Reibungskegel:

$$\min \Psi(\vec{f_c}) = w^t \cdot \vec{f_c} + \underbrace{\log(\det P^{-1}(\vec{f_c}))}_{-\sum \log(\lambda_i)} \qquad \vec{f_c} \in FC\,.$$

Die Eigenwerte λ_i von $P(\vec{f_c})$ bilden ein Maß dafür, wie weit $\vec{f_c}$ innerhalb der Reibungskegel FC liegt, denn

$$FC = \{\vec{f_c} \,|\, P(\vec{f_c}) > 0\} = \{\vec{f_c} \,|\, \lambda_i(P(\vec{f_c})) > 0\}\,,$$

d.h. $-\sum \log(\lambda_i)$ hat die Gestalt einer Barrierefunktion, die im Innern der FC klein ist, aber zum Rand steil ansteigt:



Die Lösung des Optimierungsproblems liefert optimale Kontaktkräfte $\vec{f_c}$, die dem gewählten externen Wrench F_e widerstehen können.

2.8 Anwendungsorientierung und Virtuelle Kontakte

2.8.1 Task-Ellipsoid

In Anwendungen wollen wir i.d.R. Kräfte nur in bestimmten Richtungen ausüben können, z.B. Schieben eines Objektes über den Tisch oder Anheben eines Objektes. In dieser Richtung wollen wir dann eine besonders große Kraft ausüben können, wohingegen uns Störungen in anderen Richtungen nicht so sehr interessieren.

Eine solche Aufgabe kann als Task-Ellipsoid beschrieben werden:

$$\mathcal{E}_\varepsilon = \{(\vec{x}-\vec{\mu})^t\,\Sigma^{-1}(\vec{x}-\vec{\mu}) \le \varepsilon \,|\, \Sigma \in \mathbb{R}^{6\times 6}, \vec{x}, \vec{\mu} \in \mathbb{R}^6\}\,,$$

wobei die Matrix $\Sigma = U\,\mathrm{diag}(\sigma_1,\ldots,\sigma_6)U^t$ die Form des Ellipsoids beschreibt: Die Richtungen $U = [u_1,\ldots,u_6]$ im Körper-KS werden entsprechend ihrer Bedeutung mit $0 < \sigma_i < 1$ gewichtet. Das Zentrum des Ellipsoids kann mittels $\vec{\mu}$ verschoben werden, um besonderes Gewicht auf eine bestimmte Bewegungsrichtung zu legen. Gesucht ist nun das (volumenmäßig) größte Ellipsoid, das noch in das zulässige Wrench-Polytop passt:

$$\begin{aligned} Q &= \max_{\varepsilon\in\mathbb{R}^+} \{\varepsilon \,|\, \mathcal{E}_\varepsilon \subseteq \mathcal{W}_{L_\infty}\} \\ &= \max_{\varepsilon\in\mathbb{R}^+} \{\varepsilon \,|\, \forall F \in \mathcal{E}_\varepsilon\ \exists \vec{f_c} \in FC\ :\ G\vec{f_c} = F \text{ und } \|\vec{f_c}\|_\infty \le 1\} \end{aligned}$$

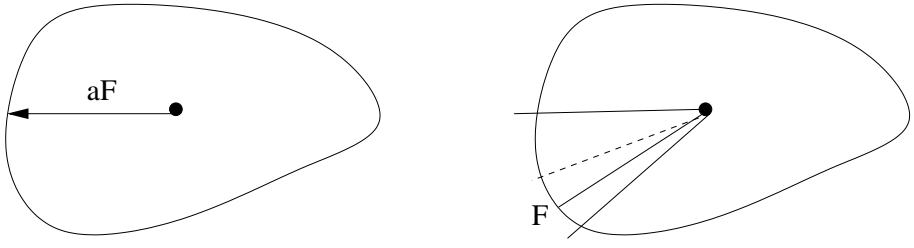
2.8.2 Virtuelle Kontakte

Wir suchen die maximale Kraft αF ($\max \alpha$), die wir in Richtung F ausüben können:

$$G\vec{f_c} = \alpha F \qquad \Leftrightarrow \qquad \begin{bmatrix} G & -F \end{bmatrix} \cdot \begin{bmatrix} \vec{f_c} \\ \alpha \end{bmatrix} = 0 \qquad \vec{f_c} \in FC \text{ und } \alpha > 0$$

Wir fügen also $-F$ als virtuellen Kontakt in die Greifmatrix $G' = [G\ -F]$ ein und suchen nach Kontaktkräften $[\vec{f_c^t}\ \ \alpha]^t \in \mathcal{N}(G') \cap int\, FC'$ mit $FC' = FC \times \{\alpha \ge 0\}$. Der virtuelle Kontakt wird also behandelt wie ein Punktkontakt ohne Reibung.

Bei gegebenem Wrench-space W, wird also nach der größten Kraft, gesucht, die in eine bestimmte Richtung F ausgeübt werden kann.



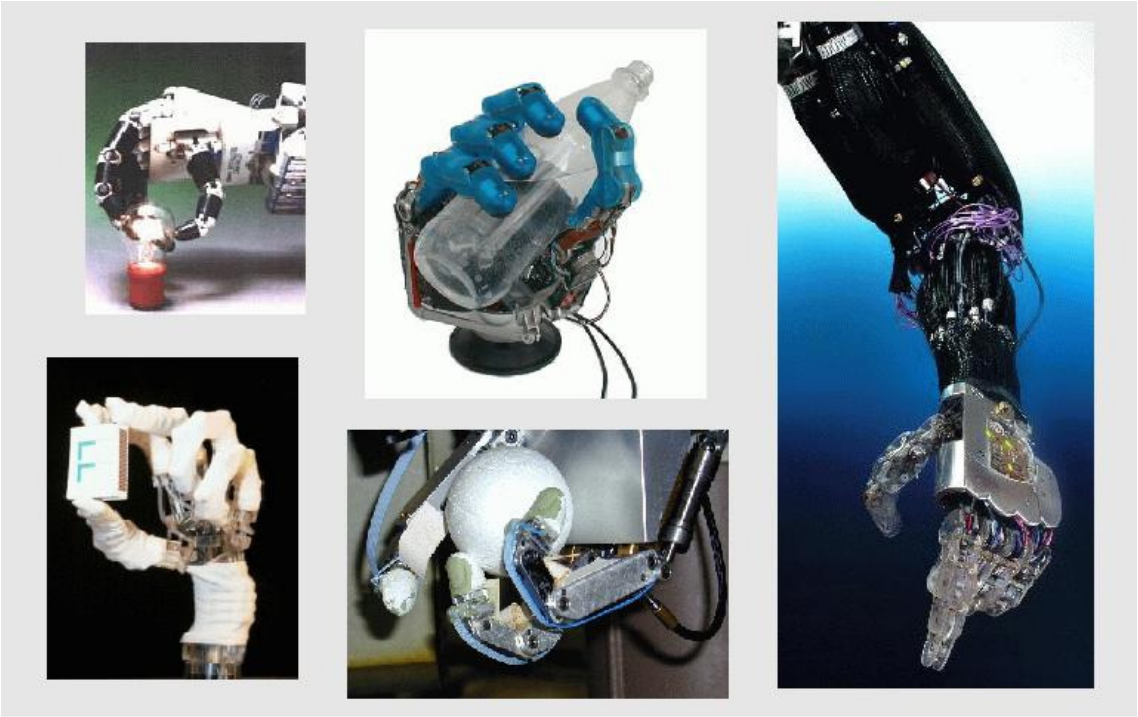
Wenn wir für den virtuellen Kontakt auch einen „Haftreibungskegel“ mit Öffnungswinkel $\tan\alpha = \mu_v$ zulassen, suchen wir in einem ganzen Kegelausschnitt nach der größten ausübbaren Kraft F. Wir optimieren also auch die Kraftrichtung, in der die größte Kraft ausgeübt werden kann.

Ein extern wirkender Wrench F_e, wie z.B. die Gravitation, kann ebenfalls als virtueller Kontakt berücksichtigt werden:

$$G\vec{f_c} = \alpha F - F_e \qquad \Leftrightarrow \qquad \begin{bmatrix} G & -F & F_e \end{bmatrix} \cdot \begin{bmatrix} \vec{f_c} \\ \alpha \\ 1 \end{bmatrix} = 0 \qquad \vec{f_c} \in FC \text{ und } \alpha > 0$$

Wir können also sowohl bevorzugte Kraftrichtungen als auch extern wirkende Wrenches im selben framework beschreiben und lediglich einen Algorithmus zur Optimierung der internen Kräfte verwenden.

3 Antriebskonzepte



- Hirzinger Hand ($4 \cdot 3 = 12$ DOF): (lokale) Elektromotoren
- Bretthauer Hand: Fluid-Aktorik
- Shadow Hand ($5 \cdot 3 + 1 = 16$ DOF): pneumatische Muskeln und Sehnen

4 Greifstrategien

Bislang haben wir Griffe unabhängig von ihrer tatsächlichen Realisierbarkeit betrachtet, d.h. Kontaktpunkte angenommen und Begriffe wie force-closure und Manipulierbarkeit studiert. Im folgenden soll skizziert werden, welche Probleme auftau-

chen, wenn man mit einer realen Hand tatsächlichen einen Griff realisieren möchte. In der Robotikforschung gibt es dazu zwei konkurrierende Ansätze:

Inverse Handkinematik	**Kontaktbasiertes Zugreifen**
– viele Iterationen – inverse Handkinematik nicht analytisch bestimmbar (hochredundantes Problem) – 3D-Geometriebestimmung, z.B. mit Laserscanner	– viele Iterationen + keine inverse Kinematik notwendig – empfindliche (und hochauflösende) Taktilsensorik

Eine technisch motivierte Zerlegung des Greifprozesses:

- Identifikation und Lokalisation des Objektes in der Szene
- Griffplanung in Abhängigkeit von Objekt, Umgebung (Hindernissen) und Aufgabenstellung
- grobe Positionierung des Roboterarms über dem Objekt
- Feinpositionierung der Hand am Objekt mittels Handkamera
- Einstellen einer objekt-spezifischen Pre-Grasp-Postur
- Kontrolliertes Zugreifen, Herstellung von Fingerkontakten
- Lageregelung des Greifobjektes
- haptische Objektidentifikation und Griffbewertung
- Trajektorienplanung für Umgreifprozesse zur Manipulation
- Objekttransport
- Ablegen / Einpassen am Zielort
- Loslassen

Äußerst komplexer Gesamtprozess!

- Greifen ist zu komplex, um es vollständig zu programmieren.
- *Lernen* von Bewegungssequenzen (Greifen, Laufen, Fahrradfahren, Tennis) und Handlungsabfolgen (z.B. Montage) ist erforderlich.
- Klassische Verfahren überwachten Lernens (Neuronale Netze) sind nicht anwendbar, da Input-*Output*-Paare nicht verfügbar sind.
- Lernen muss durch selbständige *Exploration* und *Interaktion* mit der Umgebung oder durch Imitation erfolgen.
- Ein einfaches *Bewertungssignal* muss als Rückmeldung über Güte/Erfolg der Handlung ausreichen

Wir betrachten im folgenden die beiden Lernverfahren:

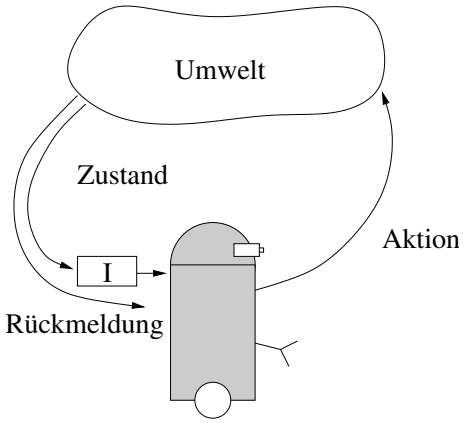
- Reinforcement Learning
- Imitation Learning

5 Reinforcement Learning (Verstärkungslernen)

Das Reinforcement Learning widmet sich Lernaufgaben, bei denen

- keine „Sollausgabe“ vorgegeben werden kann, sondern der Lerner selbständig eine geeignete Handlungsstrategie entwickeln muss.
- ein unmittelbarer zeitlicher Zusammenhang zwischen Aktion und Reaktion nicht besteht, sondern die Folgen einer Aktion möglicherweise erst viel später sichtbar werden. Typisches Beispiel: Labyrinth.

In diesem Fall sind klassische überwachte und gradientenbasisierte Lernverfahren nicht einsetzbar. Ein Reinforcement-Lerner ist ein Agent, der mit seiner Umwelt interagiert und durch trial-and-error versucht, seine Handlungsstrategie zu optimieren, um möglichst viel positives Feedback zu ernten.



Das zugrundeliegende Modell enthält folgende Komponenten:

- Der Zustand des Agenten in seiner Umwelt wird mittels einer *diskreten und endlichen Menge von Umgebungszuständen* $\mathcal{S}$ modelliert, wobei der Agent eventuell nur einen Ausschnitt $I(s)$ davon wahrnehmen kann. (Wir beschränken uns auf den Fall $I(s) = s$.)

- Ausgehend von seinem wahrgenommenen Zustand $s \in \mathcal{S}$ führt der Agent eine Aktion a aus einer *diskreten und endlichen Menge von Aktionen* $\mathcal{A}$ aus, z.B. $\mathcal{A}$ ={linkes/rechtes Rad vor/zurück bewegen}.

- Die Auswahl der Aktion wird durch die Policy $\pi : \mathcal{S} \to \mathcal{A}$ gesteuert, also einer Funktion die zu einem geg. Zustand s_t die nächste auszuführende Aktion $a_t = \pi(s_t)$ liefert. Die Policy ist die Handlungsstrategie des Agenten.

- Die Reaktion der Umwelt auf die Aktion wird durch einen endlichen Zustandsautomat modelliert. Der Folgezustand $s_{t+1} = \delta(s_t, a_t)$ ist also letztlich durch eine Zustandsübergangsfunktion $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ bestimmt.

- Damit der Agent auch lernen kann, erhält er ein Reward-Signal $r_t = r(s_t, a_t)$ oder $r_t = r(s_{t+1})$, dass den Erfolg oder den Misserfolg seiner bisherigen „Mission“ bewertet.

- Das Reward-Signal ist (leider) meistens neutral, also wertlos, und erst am Ende einer Aktionssequenz erhält der Agent eine positive oder negative Rückmeldung.
- Das Ziel des Agenten ist es, seine Strategie oder Policy π so anzupassen, dass er maximalen Reward von der Umgebung erhält.

Die Zustandsübergänge werden von den Gesetzmäßigkeiten der Umwelt diktiert, während die Reward-Funktion $r_t(s_t, a_t)$ die Lernaufgabe bestimmt.

Definition 5.1 Ein so modellierter Prozess wird auch *Markovscher Entscheidungsprozess (MDP)* genannt. Entscheidend dabei ist die Markov-Eigenschaft, die fordert, dass die Zustandsübergänge δ nur vom aktuellen Zustand s_t und der ausgeführten Aktion a_t abhängen, nicht jedoch von der gesamten Handlungsgeschichte $(s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$. Ein MDP-Iterationsschritt aus Sicht des Agenten enthält folgende Punkte:

1. Bestimme aktuellen Zustand s_t.
2. Wähle eine Handlung $a_t = \pi(s_t)$ und führe sie aus.
3. Erhalte Reward $r_t(s_t, a_t)$.
4. Umgebung geht als Reaktion auf a_t in neuen Zustand $s_{t+1} = \delta(s_t, a_t)$ über.

Beispiel 5.2 Ein Agent soll lernen, Fahrrad zu fahren.

Zustände: Neigung des Fahrrads, Geschwindigkeit

Aktionen: Drehung des Lenkers nach rechts, links

Rewards: z.B. abhängig von der Neigung des Rades,
streng negativ (Bestrafung), falls das Rad umstürzt.

Der Agent soll lernen, den kumulierten Reward zu maximieren, d.h. möglichst lange aufrecht zu fahren.

Beispiel 5.3 Ein Agent soll lernen, Schach zu spielen.

Zustände: Alle möglichen Konstellation auf dem Schachbrett (sehr sehr viele)

Aktionen: Alle möglichen Züge (auch sehr viele)

Rewards: Am Ende des Spiels erfährt der Agent ob er gewonnen (+1) oder verloren (-1) hat. Ein unerlaubter Zug wird sehr stark bestraft (-10).

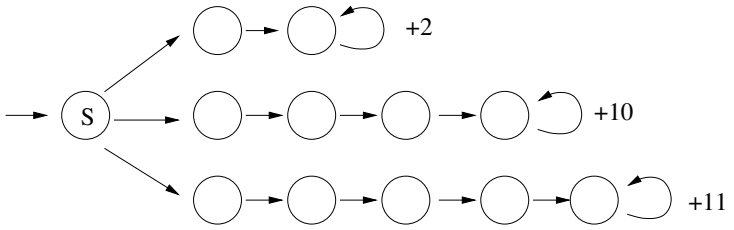
In diesem Fall hat der Agent kaum eine Chance zu lernen, wie man gewinnt, denn den positiven Reward wird er nur sehr selten erhalten... Schon eher könnte er die Schachregeln lernen, um damit die starke Bestrafung zu vermeiden...

Definition 5.4 Der *kumulierte Reward* oder auch *value function* gibt den zu erwartenden Reward an, wenn der Agent ausgehend von einem Zustand s_t seine Policy π verfolgt: $V^\pi : \mathcal{S} \to \mathbb{R}$.

Unterschiedliche Definitionen von V^π beziehen zukünftige Rewards in unterschiedlicher Weise ein:

discounted reward:	$V^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$	exponentiell abnehmende Gewichtung zukünftiger Rewards
finite-horizon reward:	$V^\pi = \sum_{t=0}^{h} r_t$	gleiche Gewichtung zukünftiger Rewards, endlicher Zeithorizont
average reward:	$V^\pi = \lim_{h\to\infty} \frac{1}{h} \sum_{t=0}^{h} r_t$	gleiche Gewichtung aller zukünftigen Rewards

Beispiel 5.5 Wir betrachten folgenden Zustandsübergangsgraphen:



$\gamma = 0.9, h = 4$

Eine Aktionsauswahl besteht eigentlich nur am Anfang. Hat der Agent sich einmal für einen Zweig entschieden, kann er immer nur eine Aktion ausführen, die ihn vorwärts bringt. Erst am Ende jeden Zweiges gibt es einen Reward – dies sind sozusagen die Zielzustände (mit unterschiedlicher Attraktivität).

discounted reward: $V^\pi = \sum_{t=0}^{\infty} \gamma^t r_t = \gamma^n \sum_{t=0}^{\infty} \gamma^t R = \frac{R\gamma^n}{1-\gamma} \approx \{16; 66; 65\}$

finite-horizon reward: $V^\pi = \sum_{t=0}^{4} r_t = \{4; 0; 0\}$

average reward: $V^\pi = \lim_{h\to\infty} \frac{1}{h} \sum_{t=0}^{h} r_t = \{2; 10; 11\}$

Je nach Wahl der Funktion für den kumulierten Reward V^π ergibt sich eine andere optimale Strategie π^*!

Im folgenden werden wir nur den discounted Reward betrachten, weil damit am einfachsten gerechnet werden kann. Der Faktor $\gamma \in (0, 1)$ bestimmt die exponentiell abnehmende Gewichtung zukünftiger Rewards, d.h. man möchte lieber heute positive Rewards sammeln als irgendwann in der Zukunft. Wir können den discounted Reward auch rekursiv berechnen:

$$\begin{aligned} V^\pi(s_0) &:= \sum_{t=0}^{\infty} \gamma^t r_t = r_0(s_0, \pi(s_0)) + \gamma\Big(r_1(s_1, \pi(s_1)) + \gamma\big(r_2(s_2, \pi(s_2)) + \dots\big)\Big) \\ &= r_0(s_0, \pi(s_0)) + \gamma V^\pi(s_1) \\ &= r_0(s_0, \pi(s_0)) + \gamma V^\pi\big(\delta(s_0, \pi(s_0))\big) \end{aligned} \tag{5.1}$$

Der discounted reward im aktuellen Zustand s wird dabei ausgedrückt als Summe des Rewards $r(s, \pi(s))$ aufgrund der Aktion $a = \pi(s)$ im aktuellen Zeitschritt und des zu erwartenden Rewards $V^\pi(s')$ im Folgezustand.

Gleichung (5.1) gilt natürlich für alle möglichen Zustände $s_0 \in \mathcal{S}$, so dass wir es eigentlich mit einem Gleichungssystem von $|\mathcal{S}|$ Gleichungen zu tun haben. Dieses System kann nach den Unbekannten $V^\pi(s), s \in \mathcal{S}$ aufgelöst und damit die value-function V^π zu gegebener Policy π bestimmt werden.

5.1 Nicht-Deterministische Prozesse

Sowohl die Zustandsübergangsfunktion der Weltmodellierung als auch die Policy des Agenten können (unabhängig voneinander) stochastisch sein, d.h. einem Zufallsgesetz folgen. Wir betrachten beide Möglichkeiten zunächst separat.

Stochastische Policy Anstatt deterministisch immer dieselbe Aktion $a = \pi(s)$ auszuwählen, ordnet die Policy π im Zustand s nun jeder Aktion $a \in \mathcal{A}$ eine Wahrscheinlichkeit $P(a|s)$ zu. Entsprechend dieser Wahrscheinlichkeitsverteilung $P(\cdot|s)$ wird die tatsächlich auszuführende Aktion ausgewählt. Die value-function muss nun den Erwartungswert der Rewards betrachten:

$$\begin{aligned} V^\pi(s) &:= \sum_{t=0}^{\infty} \gamma^t \, E(r_t) = E\Big(\sum_{t=0}^{\infty} \gamma^t \, r_t\Big) \overset{(5.1)}{=} E(r(s,\pi(s))) + \gamma E\big(V^\pi(\delta(s,\pi(s)))\big) \\ &= \sum_a P(a|s) \cdot r(s,a) \;+\; \gamma \sum_a P(a|s) \cdot V^\pi(\delta(s,a)) \end{aligned} \tag{5.1'}$$

Stochastische Zustandsübergänge Die Reaktion der Umwelt auf Aktionen des Agenten wird nun durch einen nicht-deterministischen Zustandsautomat beschrieben. Auch hier ordnet die Zustandsübergangsfunktion δ jedem Zustands-Aktions-Paar (s, a) jetzt eine Wahrscheinlichkeitsverteilung $P(s'|s, a)$ zu, die angibt mit welcher Wahrscheinlichkeit der Folgezustand s' erreicht wird. Die value-function muss wieder den Erwartungswert der Rewards betrachten, diesmal gewichtet über alle möglichen Folgezustände s':

$$V^\pi(s) := E\Big(\sum_{t=0}^{\infty} \gamma^t \, r_t\Big) \overset{(5.1)}{=} r(s,a) + \gamma \sum_{s'} P(s'|s,a) \cdot V^\pi(s') \qquad a = \pi(s) \tag{5.1"}$$

Wir werden i.d.R. nur stochastische Übergangsfunktionen δ, nicht jedoch stochastische Policies betrachten. Man beachte, dass sich die deterministischen Gleichungen als Grenzfall der stochastischen ergeben, wenn man die Wahrscheinlichkeitsverteilungen $P(\cdot|s, a)$ bzw. $P(\cdot|s)$ für genau einen Folgezustand s' bzw. für genau eine Aktion a Eins sind.

Für die optimale value-function $V^* \equiv V^{\pi^*}$ gelten die *Bellman'schen Optimalitätsgleichungen:*

$$V^*(s) = \max_a \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \cdot V^*(s') \Big) \quad \forall s \in \mathcal{S}$$

Diese Gleichungen sind wieder rekursiv definiert und beschreiben lediglich, dass die optimale value-function in jedem Zustand s den maximal erreichbaren Reward angibt.

5.2 Policy Iteration

Das Lernproblem besteht nun aber darin, die optimale Policy zu lernen. Die folgende Gleichung liefert eine Iterationsvorschrift um eine bisherige Policy π mit zugehöriger value-function V^π (mittels einer greedy-Strategie) zu verbessern:

$$\pi'(s) := \arg\max_a \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \cdot V^\pi(s') \Big) . \tag{5.2}$$

Die neue Policy π' hat für jeden Zustand s eine höhere Bewertung $V^{\pi'}(s)$. Wir erhielten genau die alte Policy $\pi' = \pi$ und die alte value-function V^π, wenn wir $a = \pi(s)$ wählten. Da aber über alle möglichen Aktionen a maximiert wird, verbessert sich die Policy π' bzw. die zugehörige value-function $V^{\pi'}$. Wir erhalten damit den folgenden Algorithmus, um die optimale Policy π^* zu lernen:

```
Initialisiere die Policy π' zufällig
loop
     π := π'
     Berechne V^π durch Lösung von (5.1-5.1'')
     Verbessere die Policy entsprechend (5.2)
until π = π'
```

Die Berechnung der value-function V^π über das Gleichungssystem (5.1') ist u.U. sehr aufwendig. Das folgende iterative Verfahren ermöglicht eine Approximation von V^π:

```
Initialisiere k = 0, V_k^π = 0
loop
     V_{k+1}^π(s) = r(s, π(s)) + γ Σ_{s'∈S} P(s'|s, π(s)) · V_k^π(s')
     k → k + 1
until V_{k+1}^π ≈ V_k^π
```

5.3 Value Iteration

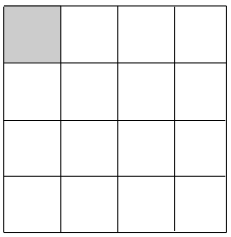
Alternativ kann man direkt die value-function optimieren:

```
Initialisiere k = 0, V_k(s) = 0
loop
     Berechne Q_k(s,a) := r(s,a) + γ Σ_{s'∈S} P(s'|s,a) · V_k(s')
     Update V_{k+1}(s) := max_a Q_k(s,a)
     k → k + 1
until Änderungen von V sind klein genug
```

Es kann gezeigt werden, dass dieser Algorithmus tatsächlich zur optimalen value-function V^* konvergiert. Die optimale Policy kann von dieser einfach abgeleitet werden:

$$\pi^*(s) := \arg\max_a \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \cdot V^*(s') \Big) . \tag{5.3}$$

Beispiel 5.6
Zum Vergleich der beiden Verfahren Policy- und Value-Iteration schauen wir uns das typische Reinforcement-Problem an: Ein Agent kann sich auf einem endlichen Gitter fortbewegen, indem er eine der Aktionen *hoch, runter, rechts, links* auswählt. Am Rand des Gitters sollen „verbotene“ Aktionen unberücksichtigt bleiben, d.h. der Agent erhält keinen negativen Reward, bewegt sich aber auch nicht fort. Ziel des Agenten soll es sein, eines der grauen Felder zu erreichen (und dort zu bleiben), so dass nur bei Erreichen dieser Felder ein positiver Reward von Eins erteilt wird.



Policy Iteration

$\xrightarrow{\pi_0}$

←	←	←	←
→	→	→	↑
↓	←	←	←
→	→	→	↑

$\xrightarrow{V^{\pi_0}}$

100	100	90	81
53.14	59.05	65.61	72.9
0	0	0	0
0	0	0	0

$\xrightarrow{\pi_1}$

←	←	←	←
↑	↑	↑	↑
↑	↑	↑	↑
→	→	→	↑

$\xrightarrow{V^{\pi_1}}$

100	100	90	81
100	90	81	72.9
90	81	72.9	65.61
43.05	47.83	53.14	59.05

$\xrightarrow{\pi_2}$

←	←	←	←
↑	↑	↑	↑
↑	↑	↑	↑
↑	↑	↑	↑

$\xrightarrow{V^{\pi_2}}$

100	100	90	81
100	90	81	72.9
90	81	72.9	65.61
81	72.9	65.61	59.05

$\xrightarrow{\pi_3=\pi_2}$ ...

Nach jeder Verbesserung der Policy muss hier die auf der neuen Strategie basierende Wertefunktion exakt berechnet werden. Dafür ist jeweils eine größere Menge an Iterationen erforderlich.

Value Iteration Im Gegensatz dazu, benötigt Value Iteration pro Iterationsschritt nur wenige Rechenschritte, um die neue Wertefunktion zu bestimmen:

$\xrightarrow{V_0}$

10	10	0	0
10	0	0	0
0	0	0	0
0	0	0	0

$\xrightarrow{V_1}$

19	19	9	0
19	9	0	0
9	0	0	0
0	0	0	0

$\xrightarrow{V_2}$

27.1	27.1	17.1	8.1
27.1	17.1	8.1	0
17.1	8.1	0	0
8.1	0	0	0

$\xrightarrow{V_3}$

34.39	34.39	24.39	15.39
34.39	24.39	15.39	7.29
24.39	15.39	7.29	0
15.39	7.29	0	0

$\xrightarrow{V_4}$

40.95	40.95	30.95	21.95
40.95	30.95	21.95	13.85
30.95	21.95	13.85	6.56
21.95	13.85	6.56	0

$\xrightarrow{V_5}$

46.86	46.86	36.86	27.86
46.86	36.86	27.86	19.76
36.86	27.86	19.76	12.47
27.86	19.76	12.47	5.9

$\xrightarrow{V_6}$

52.17	52.17	42.17	33.17
52.17	42.17	33.17	25.07
42.17	33.17	25.07	17.78
33.17	25.07	17.78	11.22

$\xrightarrow{V_7}$

56.95	56.95	46.95	37.95
56.95	46.95	37.95	29.85
46.95	37.95	29.85	22.56
37.95	29.85	22.56	16

$\rightarrow \cdots \xrightarrow{V_\infty}$

100	100	90	81
100	90	81	72.9
90	81	72.9	65.61
81	72.9	65.61	59.05

Policy Iteration benötigt zwar weniger Iterationen, ist aber pro Iteration langsamer als Value Iteration. Es ist daher nicht klar, welcher Algorithmus der bessere ist, und es gibt Beispiele, für die der eine oder andere Algorithmus schneller konvergiert. Policy Iteration und Value Iteration sind Beispiele von *dynamischer Programmierung*. Sie benutzen die Zustandsübergangsfunktion $\delta(s, a)$ und die Reward-Funktion $r(s, a)$.

5.4 Q-Lernen

Policy und Value Iteration benutzen sowohl die Reward-Funktion $r(s, a)$ als auch die Wahrscheinlichkeiten für Zustandsübergänge $P(s'|s, a)$, um die optimale Policy zu finden. Diese Daten sind i.d.R. jedoch nicht ad hoc verfügbar und der Agent muss sie durch Exploration der Umwelt erst lernen. Das Q-Lernen umgeht dieses Modell-Wissen durch Nutzung der Q-Funktion (oder auch action-value function genannt):

$$Q^\pi(s, a) := E\big(V^\pi(s) \mid a\big) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot V^\pi(s')$$

Sie gibt den erwarteten Reward an, wenn man in Zustand s **die Aktion** a **auswählt** sowie **im weiteren** der Policy π folgt. Die ursprüngliche value function $V^\pi(s)$ wird also zu einer zwei-parametrigen Funktion $Q^\pi(s, a)$ „aufgebläht“, mit dem Erfolg, dass die auszuführende Aktion a im aktuellen Zustand s von der ansonsten verfolgten Policy (und der damit verbundenen value-function V^π) entkoppelt wird. Man kann also im aktuellen Zustand s nun **alle** möglichen Aktionen betrachten. Die „alte“, von a unabhängige value function erhält man aus Q^π mittels der Beziehung $V^\pi(s) = Q^\pi(s, \pi(s))$, also durch Verwendung der Aktion $a = \pi(s)$. Analog zu Gleichung (5.3) kann die zu Q gehörige Policy π bestimmt werden:

$$\pi(s) = \arg\max_a Q(s, a)\,. \tag{5.4}$$

Mit der Beziehung $V^*(s) = \max_a Q^*(s, a)$ kann man die Bellman'schen Optimalitätsgleichungen für die optimale Q-Funktion Q^* aufstellen:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot \underbrace{\max_{a'} Q^*(s', a')}_{V^*(s')}$$

5.4.1 Basisalgorithmus des Q-Lernen

Diese rekursiven Definition(en) von Q bilden im Prinzip die Grundlage für den Q-Lernalgorithmus. Wir müssen jedoch noch die Weltmodellierung in Form der Übergangswahrscheinlichkeiten $P(s'|s, a)$ loswerden. Aber wir können den Folgezustand s' aufgrund einer ausgewählten Aktion a ja wieder beobachten! Der Lernalgorithmus wird dadurch zu einem (inter)aktiven Lernprozess in der Umwelt:

```
Initialisiere k = 0, Q_k(s,a) zufällig, aber klein
loop
     Bestimme aktuellen Zustand s
     Führe eine Aktion a aus (zufällig oder entsprechend Q_k (5.4))
     Erhalte Reward r(s,a) bzw. r(s,a')
     Bestimme den Folgezustand s'
     Update Q_{k+1}(s,a) := r(s,a) + γ max_{a'} Q(s',a')
     k → k + 1
until Änderungen von Q sind klein genug
```

Das Tupel $\langle s, a, r, s' \rangle$ beschreibt die *Erfahrung*, im Zustand s bei ausgeführter Aktion a im Zustand s' zu landen und den Reward r erhalten zu haben. Das Update wird so gemacht, dass man den höchstmöglichen Q-Wert $\max_{a'} Q(s', a')$ im Folgezustand verwendet (greedy).

5.4.2 Temporal Difference Learning: TD(0)

Eine zentrale Idee des Reinforcement Lernens ist das *temporal difference*-Lernen, kurz TD-Lernen. Es kombiniert zwei wichtige Aspekte:

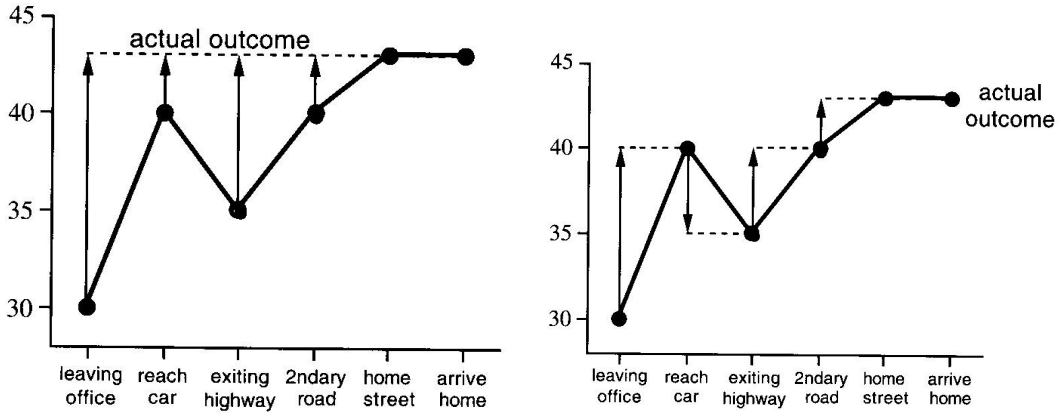
- das direkte Lernen aus den eigenen Erfahrungen in Abwesenheit eines Modells des zu lernenden Problems und
- die Fähigkeit, noch vor Erreichen des Ziels die aktuellen Werte auf der Basis anderer, ebenfalls noch nicht perfekt gelernter Werte, zu verbessern (das sog. *bootstrapping*).

Die Updateformel für TD-Lernen ist nur mithilfe der Value-Funktion $V(s_t)$ definiert:

$$V(s) := V(s) + \alpha\Big(r + \gamma V(s') - V(s)\Big), \tag{5.5}$$

$\alpha \in [0, 1]$ ist dabei die Lernschrittweite.

Beispiel 5.7 Ein Dozent fährt aus der Uni mit dem Auto nach Hause. Beim Verlassen seines Büros registriert er die Uhrzeit, den Wochentag und sonstige relevante Daten. An einem Freitag verläßt er sein Büro um 18 Uhr und schätzt, daß er 30 Minuten bis zu seinem Haus brauchen wird. Beim Betreten des Parkplatzes um 18:05 Uhr beginnt es aber zu regnen, so daß er seine Schätzung auf eine Gesamtreisezeit von 40 Minuten erhöht. Eine Viertelstunde später (also um 18:20 Uhr) hat er die Autobahn verlassen und ist dabei sehr glatt durchgekommen – er revidiert die geschätzte Reisezeit auf 35 Minuten. Dummerweise setzt sich auf der Landstraße ein nicht überholbarer Trecker vor ihn, so daß er erst um 18:40 Uhr in seine Straße einbiegt und um 18:43 Uhr dann glücklich zu Hause ankommt.



Wie aus Gleichung 5.5 ersichtlich, ist die Wahl der Aktionen bei TD-Lernen nicht festgelegt. Hier gibt es zwei prinzipielle Möglichkeiten:

- on-policy-Lernen: Die Policy π wird sowohl zur Verhaltensgenerierung als auch zur Bestimmung der Wertefunktion Q^π im Folgezustand verwendet:

$$Q(s,a) = Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$$

  Die Aktion a', die im Folgezustand s' ausgewählt wird, wird also mittels der Policy bestimmt: $a' = \pi(s')$. Da hierbei das Tupel (s,a,r,s',a') verwendet wird, heißt dieser Algorithmus *Sarsa*.

- off-policy-Lernen: Die Policy π wird nur zur Verhaltensgenerierung im aktuellen Zustand verwendet. Der Wert des Folgezustandes wird aber mittels einer anderen Stratgie (typischerweise greedy) bestimmt:

$$Q(s,a) := Q(s,a) + \alpha\Big(\underbrace{r + \gamma \max_{a'} Q(s',a')}_{Q'(s,a)} - Q(s,a)\Big) \tag{5.6}$$

  Hier erkennt man den Basisalgorithmus des Q-Lernen wieder, der über alle möglichen Aktionen im Folgezustand maximiert. Im Prinzip wird in der Praxis nur dieser Algorithmus verwendet.

Anders als in Kapitel 5.4 wird der aktuelle Wert hier jeweils nur in Richtung des neuen Wertes verändert - man springt nicht mehr sofort zum neuen Wert, sondern erhält einen fließenden Übergang zu diesem Wert. Dies ist insbesondere dann von Vorteil, wenn der Algorithmus momentan einer suboptimalen Strategie folgt. Die Lernschrittweite $\alpha \in [0,1]$ bestimmt, wie groß der Lernschritt in Richtung des neuen Q-Wertes $Q'(s,a)$ sein soll. Für $\alpha = 1$ erhält man wieder die obige Q-Lernregel, für $\alpha = 0$ lernt man gar nichts.

Damit lautet der Algorithmus:

`Initialisiere` $k = 0, Q_k \equiv 0$
`loop`
    `Beobachten aktuellen Zustand` s
    `Wähle Handlung` $a = \pi(s)$
    `Erhalte Reward` r `und beobachte Folgezustand` s'
    `Update` $Q_{k+1}(s,a) := Q_k(s,a) + \alpha\big(r + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a)\big)$
`until` Änderungen von Q sind klein genug

Satz 5.8 Unter folgenden Voraussetzungen konvergiert Q gegen die optimale Q-Funktion Q^*:

- $|r(s,a)| < \infty \qquad \forall s, a,$
- $0 \leq \gamma < 1,$
- Jedes (s,a)-Paar wird unendlich oft besucht.

5.4.3 Temporal Difference Learning: TD(λ)

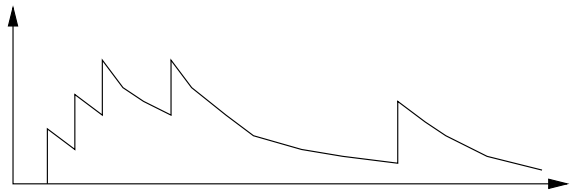
Die Lernregel TD(0) berücksichtigt nur eine einzige Erfahrung (s_t, a_t, r_t) des Agenten:

$$Q(s_t,a_t) := Q(s_t,a_t) + \alpha\Big(r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\Big)\,.$$

Damit führt jeder Iterationsschritt des Agenten nur zu einem einzigen Update der Q-Funktion. Falls der Reward stark verzögert kommt, baut sich die richtige Q-Funktion nur sehr langsam auf. Sinnvoller ist es daher bei einem Update auch vorherige Schritte – also den gesamten Weg zum Ziel – zu berücksichtigen. Dazu führen wir die *Eignung (eligibility)* eines (s,a)-Paares ein:

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{falls } (s,a) = (s_t,a_t) \\ \gamma\lambda e_{t-1}(s,a) & \text{sonst} \end{cases}$$

Dabei bestimmt der Faktor λ, wie stark die vorherigen Zustände gewichtet werden sollen. Falls ein Paar (s,a) im Zeitschritt t besucht wird, erhöht sich die Eignung um 1, sonst nimmt sie exponentiell ab:

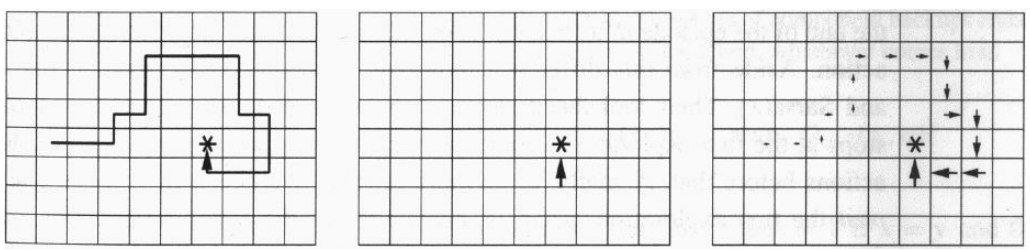


Je höher der Wert $e(s,a)$, desto öfter wurde das Paar besucht, desto weniger in der Vergangenheit liegen diese Besuche und desto wichtiger ist offenbar sein Anteil am erzielten Erfolg (Reward). Dementsprechend nutzen wir folgende Updateregel:

$$\begin{aligned} Q(s,a) &:= Q(s,a) + \alpha \cdot e(s,a) \cdot \Delta(s,a,s') \qquad \text{für alle } (s,a) \in \mathcal{S} \times \mathcal{A} \\ \Delta(s,a,s') &= r + \gamma \max_{a'} Q(s',a') - Q(s,a) \end{aligned}$$

Die alte Lernregel erhalten wir mit $e(s,a) = \delta_{ss_t}\delta_{aa_t}$, also für $\lambda = 0$.

Beispiel 5.9



durchlaufener Pfad 1-Step-Update: TD(0) TD(λ)

5.4.4 Exploration vs. Exploitation

Exploration: meist zufällige Erforschung des Zustandsraumes
Exploitation: optimale Ausnutzung des bisher gewonnenen Wissens (Approximation von V oder Q) zur Bestimmung der Policy.

- Zu ausgiebiges Erforschen bedeutet, dass der Agent auch nach langem Lernen noch ziellos im meist sehr großen Zustandsraum umherwandert. Dadurch werden auch Bereiche intensiv untersucht, die für die Lösung der Aufgabe gar nicht relevant sind.
- Zu frühes Ausbeuten der gelernten Approximation der Q-Funktion bewirkt möglicherweise, dass sich ein suboptimaler Pfad durch den Zustandsraum etabliert, weil dieser zufällig zuerst gefunden wurde und gute Ergebnisse lieferte. Die optimale Lösung wird jedoch nicht mehr gesucht.

Beispiel 5.10 Mehrere einarmige Banditen, deren Ausschüttung und Erfolgsquote zufällig sind. Wie lange soll man bei anderen Automaten nach höheren Ausschüttungsquoten suchen und ab wann soll man sich auf einen Automaten konzentrieren?

Lösung: teilweise zufällige Auswahl der Aktionen.

ε-greedy: Wähle typischerweise die optimale Aktion a^* aus, aber mit Wahrscheinlichkeit $\frac{\varepsilon}{|\mathcal{A}|}$ eine zufällige andere Aktion a.

Boltzmann-Gewichtung: Wähle die Aktionen a zufällig entsprechend der Boltzmann-Verteilung aus:

$$\pi(s,a) = \frac{\mathrm{e}^{\beta(Q(s,a)-Q(s,a^*))}}{\sum_{a'} \mathrm{e}^{\beta(Q(s,a')-Q(s,a^*))}} \quad \text{mit } a^* = \arg\max_a Q(s,a)$$

Die Lernparameter $\varepsilon \in [0,1]$ und $\beta \in [0,\infty)$ sollten langsam abnehmen.

- $\varepsilon = 1$ bzw. $\beta = 0 \Rightarrow$ Gleichverteilung
- $\varepsilon = 0$ bzw. $\beta = \infty \Rightarrow$ greedy

Der wichtigste Unterschied zwischen ε-greedy und Boltzmann-Exploration ist die Auswahl der auszuführenden Aktion im Falle der Exploration: während bei ε-greedy jede suboptimale Aktion, also auch die bisher am schlechtesten bewertete, die gleiche Chance hat, ausgesucht zu werden, werden suboptimale Aktionen bei der Boltzmann-Exploration mit einer Wahrscheinlichkeit ausgewählt, die mit ihrem momentanen Wert korreliert.

5.5 Generalisierung: Kontinuierliche Zustands- und Aktionsräume

Alle Algorithmen sind von diskreten Zustands- und Aktionsräumen ausgegangen und speichern die Funktionen $V(s)$ und $Q(s,a)$ in einer Tabelle. Dies ist sehr ineffizient und im Falle von kontinuierlichen Zustands- und Aktionsräumen nicht mehr praktikabel.

Lösung: Funktionsapproximation für $V(s)$ bzw. $Q(s,a)$.
allg. Ansatz eines Approximators: $\hat{Q}_w(s,a)$.
Die Gewichte w haben je nach Wahl des Approximators unterschiedliche Bedeutung:

- Splines/Polynome: Stützstellen
- Multilagen-Perzeptron (MLP): Gewichte im Netz
- SOM, LLM, PSOM: Stützstellen

Ein guter Funktionsapproximator sollte folgende Eigenschaften haben:

- Gute Generalisierung, d.h. Interpolation und Extrapolation bei unbekannten Zuständen und Aktionen.
- Gute Approximation der tatsächlichen Q-Funktion. Dazu muss die Auflösung der Q-Funktion genügend hoch sein, kann aber in unterschiedlichen Bereichen des Raumes $\mathcal{S} \times \mathcal{A}$ variieren.
- Geringer Speicherbedarf
- Effiziente Berechnung der optimalen Aktion: $\arg\max_{a \in \mathcal{A}} \hat{Q}_w(s,a)$.
  Bei kontinuierlichem Aktionsraum $\mathcal{A}$ stellt dies ein nichtlineares Optimierungsproblem dar, mit all seinen Problemen (lokale Minima = suboptimale Aktionsauswahl)
- Effiziente Updates der Q-Funktion.
- Lokale Lernverfahren.

Einige dieser Eigenschaften widersprechen einander und man muss einen geeigneten Mittelweg finden, der alle Eigenschaften ausreichend berücksichtigt.

5.6 Self-Organizing Map (SOM) / Neural Gas

Die Tabellendarstellung von $Q(s,a)$ weist kontinuierlichen Zustands-Aktions-Paaren zunächst diskrete Werte $(\bar{s}, \bar{a})$ zu und diesen dann den Q-Wert $Q(\bar{s}, \bar{a})$. Die Q-Funktion ist damit als stückweise konstante Funktion (auf den Zellen des Gitters) dargestellt. SOMs stellen eine Alternative zur Vektorquantisierung (VQ) dar, um kontinuierliche Paare (s,a) auf diskrete $(\bar{s}, \bar{a})$ abzubilden.

Wir wollen die SOM im folgenden lediglich zur Quantisierung der Zustände $s \in \mathcal{S}$ benutzen – die Aktionen selbst seien weiterhin diskret.

Die allgemeine SOM besteht aus folgenden Komponenten:

- Diskretes Gitter von Knoten oder Neuronen: $r \in \mathbb{R}^m$.
- Jedem Knoten r ist ein Gewichtsvektor $\vec{w}_r \in \mathbb{R}^I \triangleq \mathcal{S}$ im Input-Raum (Zustandsraum) zugeordnet.
- Zu gegebenem Input $\vec{x} = s$ bestimme den Ausgabe-Knoten (winner):
  $r^* = \arg\min_{r \in \mathbb{R}^m} \|\vec{x} - \vec{w}_r\|$.

- SOM-Lernregel: $\Delta\vec{w}_r = \varepsilon \cdot h(r, r^*) \cdot (\vec{x} - \vec{w}_r)$
- Nachbarschaftsgewichtung mit Gaußglocke: $h(r, r^*) = exp(-\frac{\|r-r^*\|}{2\sigma^2})$.
- Nachbarschaftsbeziehungen auf dem abstrakten Gitter werden auf den Input-Raum übertragen.
- Das abstrakte Gitter kann man sich als Koordinatensystem für die m-dimensionale Hyperfläche im n-dimensionalen Input-Raum vorstellen.
- Anstatt die Nachbarschaftsgewichtung auf Gitter-Abständen $\|r - r^*\|$ zu berechnen, kann man auch direkt die Abstände im Input-Raum verwenden: $\|\vec{w}_r - \vec{w}_{r^*}\|$. Das abstrakte Gitter ist dann überflüssig und das resultierende Modell nennt man *Neural Gas*.
- SOM und Neural Gas sind Vektorquantisierer und können daher als Ersatz für die Q-Tabellen-Quantisierung eingesetzt werden.

Voronoi-Parzellierung Die Voronoi-Parzellierung gibt zu gegebenen Quantisierungsvektoren die zugehörigen Regionen im Input-Raum an, die auf diese Vektoren abgebildet werden (aufgrund des kleinsten Abstandes zu diesen Vektoren).

Nach der Quantisierung mit der SOM können alle tabellenbasierten RL-Algorithmen ohne Änderungen angewandt werden. Der SOM-Algorithmus garantiert eine variierende Dichte der Quantisierungsvektoren im Zustandsraum – proportional zur Häufigkeit mit der die entsprechenden Regionen besucht werden.

Die Nachbarschaftsbeziehungen der SOM können nun auch genutzt werden, um kontinuierlich zwischen den diskreten Q-Werten $Q(\bar{s}, a)$ zu interpolieren. Dazu benutzen wir neben dem winner-Vektor $\bar{s}^*$ auch alle benachbarten Quantisierungsvektoren $\bar{s}_i \in \mathcal{N}$:

$$\hat{Q}(s,a) = \sum_{\bar{s}\in\mathcal{N}} \underbrace{\frac{\|\bar{s}-s\|}{\sum_{\bar{s}\in\mathcal{N}}\|\bar{s}-s\|}}_{w_{\bar{s}}} Q(\bar{s},a) = \sum_{\bar{s}\in\mathcal{N}} w_{\bar{s}}\, Q(\bar{s}_i,a) \qquad \sum_{\bar{s}\in\mathcal{N}} w_{\bar{s}} = 1$$

Der Interpolationswert $\hat{Q}(s, a)$ entsteht also durch Mittelung der Q-Tabellenwerte $Q(\bar{s}_i, a)$ über alle benachbarten Vektoren $\bar{s}_i \in \mathcal{N}$, wobei die Q-Werte entsprechend dem Abstand des Vektors $\bar{s}_i$ zum tatsächlichen Zustandsvektor s gewichtet werden.

Die TD(λ)-Update-Regel erhält folgende Form:

$$Q(\bar{s},a) := Q(\bar{s},a) + \alpha \underbrace{\left(r + \max_{a'} \gamma\, \hat{Q}(s',a') - \hat{Q}(s,a)\right)}_{\Delta(s,a,s')} e(\bar{s},a)$$

Den Fehler $\Delta(s, a, s')$ berechnen wir also mittels der $\hat{Q}$-Approximationen für die tatsächlich beobachteten *kontinuierlichen* Zustände s und s'. Die Eignung $e_t(\bar{s}, a)$ wird nur noch entsprechend dem jeweiligen Anteil $w_{\bar{s}}$ der Quantisierungsvektoren $\bar{s}$ zum Gesamtwert von Q erhöht:

$$e_t(\bar{s},a) = \begin{cases} \gamma\lambda e_{t-1}(\bar{s},a) + w_{\bar{s}} & \text{falls } \bar{s}\in\mathcal{N} \text{ und } a = a_t \\ \gamma\lambda e_{t-1}(\bar{s},a) & \text{sonst} \end{cases}$$

5.7 Verbleibende Probleme

Unvollständige Zustandsinformation erfordert internen Zustand des Agenten, der Historie der Zustände und Aktionen kodiert. Damit ist es möglich Mehrdeutigkeiten in der Zustandsinformation aufzulösen.
Beispiel: Labyrinth. Der Agent sieht aber nur den aktuellen Gang und kann nicht zwischen ähnlichen Gängen unterscheiden.

Schlechte Skalierung mit der Größe (Dimensionalität) des Lernproblems. Zur Lösung komplexer Probleme muss Vorwissen in die Strukturierung der Lernaufgabe eingehen (bias):

Shaping: Präsentation von zunächst einfachen Lernaufgaben, um Basis-Fähigkeiten zu Erlernen. Übergang zu immer komplexeren Aufgaben. Beispiel: Labyrinthe.

Lokale Rewards: Rewards, die in zeitlicher Nähe zur verursachenden Handlung stehen, erhöhen die Lerngeschwindigkeit. Schlecht: Rewards erst bei Erfolg/Misserfolg einer ganzen Handlungssequenz.

Imitation: Einschränkung des Suchraumes durch Nachahmen einer vorgeführten Handlung.

Reflexe: Eine vorprogrammierte Heuristik kann ebenfalls helfen, den Suchraum einzuschränken. Sie stellt eine sinnvolle Start-Policy dar, die durch Erfahrung aber auch verändert werden kann.

Hierarchiches Lernen: Zerlegung des Problems in eine Hierarchie von Teilproblemen, die separat gelernt werden.

6 Imitation Learning

Imitation: Nachahmung einer beobachteten Bewegungssequenz

Das Gebiet ist relativ neu und bislang existieren kaum Anwendungen. Wir zeigen im folgenden ein paar Probleme und Fragestellungen des Imitation Learning auf.

1. Was soll nachgeahmt werden?
2. Wie wird eine Status- bzw. Sensorsequenz in sinnvolle Teilstücke zerlegt?
3. Wie müssen die beobachteten Sequenzen ausgeführt werden, d.h. welche Aktionen führen zur beobachteten Bewegungssequenz?
   Wenn Armbewegungen beobachtet werden, sieht man i.d.R. nur den Effekt der Gelenkwinkeländerungen. Welche Gelenkwinkeländerungen notwendig sind, um diesen Effekt zu erzielen, muss der Lerner selbst explorieren.

6.1 Was soll nachgeahmt werden?

Handlungsequenzen kann man sich auf verschiedenen Abstraktionsebenen vorstellen. Beispiel Greifen:

- Gelenkwinkel: Welche Arm- und Fingerbewegungen hat der Instruktor gemacht?
- Griffpunkte: Wo hat der Instruktor das Objekt gegriffen?
- Griffpostur: Welchen Grifftyp (Pinzettengriff, Kraftgriff) hat der Instruktor ausgewählt, um das Objekt zu greifen und an welcher Stelle des Objekts greift er zu?
- Was hat der Instruktor gegriffen?

Am Beispiel der Manipulation, z.B. Zusammenbau eines Baufix-Flugzeugs, wird klar, dass man sich typischerweise einen *hierarchischen Aufbau* vorstellen kann.

- Gelenkwinkel: Welche Handbewegungen sind notwendig, um zwei Teile aufeinander zu legen, eine Schraubverbindung herzustellen, etc.
- In welcher Reihenfolge müssen welche Teile miteinander verbunden werden?
- Kommt es auf die Farbe oder Länge der Objekte an oder sind diese Eigenschaften irrelevant?

Dies motiviert die hierarchische Aufteilung in Handlungsprimitiven. (behavioural primitives).

Beispiel 6.1 Kinder sollen Armbewegungen nachmachen, bei denen

- ein Fleck auf dem Tisch durch die Hand bedeckt wird,
- kein Fleck auf dem Tisch existiert, aber dieselbe Armbewegung ausgeführt wurde.

Falls ein Fleck zu sehen ist, benutzen die Kinder beide Arme mit gleicher Wahrscheinlichkeit, um den Fleck zu verdecken. Ohne Fleck, benutzen die Kinder immer denselben Arm wie der Instruktor. Das Ziel der Handlung wurde als top-down festgelegt.

Auf welche Eigenschaften der vorgeführten Handlungssequenz(en) es ankommt, kann z.B. mittels einer Invarianz-Analyse festgestellt werden: Welche Teile der Handlungssequenz wiederholen sich immer wieder (scheinen also relevant für die Aufgabe zu sein), welche variieren (und sind daher irrelevant)?

6.2 Wie soll eine Handlungssequenz in sinnvolle Teilstücke (primitives) zerlegt werden?

- Welche Handlungsprimitiven sind überhaupt sinnvoll?
- Welche Primitiva können in hierarchischer Weise zusammengefasst werden?
- Wie können die Primitiva in einer Handlungssequenz detektiert werden? Lösung: Ansätze aus der Spracherkennung

6.3 Imitation auf der Gelenkwinkel-Ebene

Auf unterster Hierarchie-Ebene müssen i.d.R. Gelenkwinkel-Trajektorien reproduziert werden. Es hat sich gezeigt, dass eine direkte Übertragung der beobachteten Gelenkwinkel beim Menschen auf Roboter-Gelenkwinkel (selbst humanoide Roboter) nicht möglich ist, da beide immer eine (leicht) unterschiedliche Kinematik haben – insbesondere variieren Segmentlängen und die Anordnung der Gelenke zueinander.