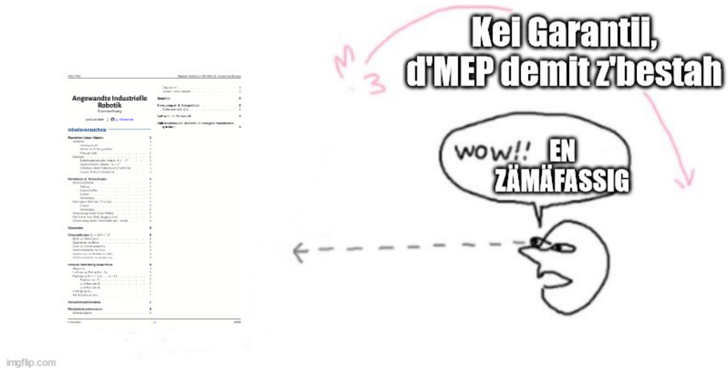
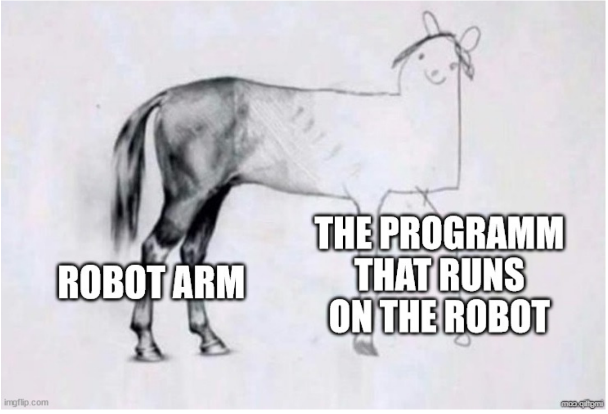
Angewandte Industrielle Robotik

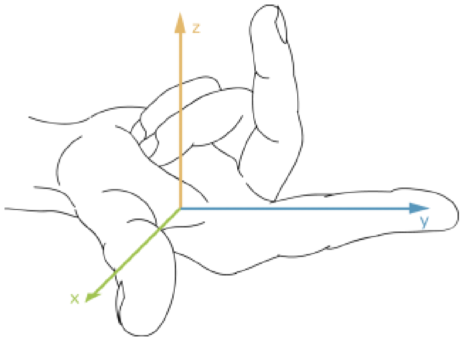
Zusammenfassung

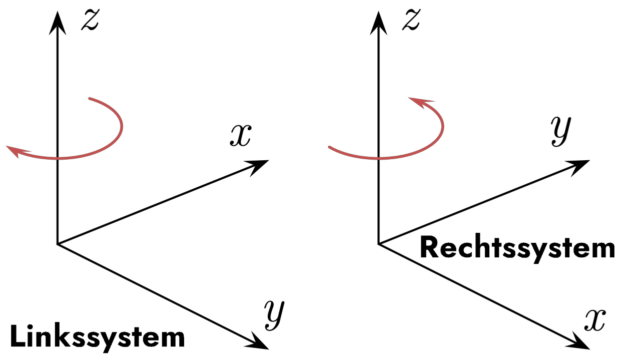
Joel von Rotz / Quelldateien

Inhaltsverzeichnis





> **ℹ Rechtssystem**
>
> Robotereffektoren, Armteile, etc. werden im kartesischen Koordinatensystem dargestellt anhand des **Rechtssystems**.
>
> 
>

> **ℹ Drehung mit der Rechtenhandregel**
>
> Bei der Rotation von Rechtssystemen kann die rechte Handregel angewendet werden. Der Daumen zeigt in die gleiche Richtung der Drehachse (vom Nullpunkt nach aussen) und die rechstlichen Finger zeigen die Drehrichtung an: **Gegenunzeigersinn**!
>
> 
>

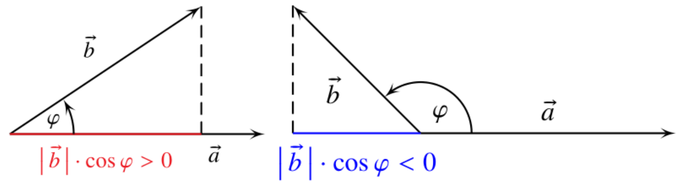
Repetition Linear Algebra

Für das meiste die Linear Algebra Zusammenfassung anschauen.

Aber sonst das Wichtigste

Vektoren

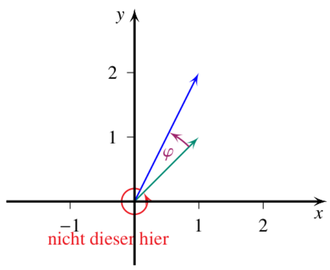
Skalarprodukt



Das Skalarprodukt entspricht der Multiplikation der Projektion $\vec{b_a}$ auf $\vec{a}$ mit $\vec{a}$

$$\vec{a} \bullet \vec{b} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \bullet \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = a_1 \cdot b_1 + \cdots + a_n \cdot b_n = \sum_{i=1}^{n} a_i \cdot b_i$$

Winkel & Orthogonalität



Beim Berechnen des Winkels zwischen zwei Vektoren

$$\varphi = \arccos \frac{x \bullet y}{\|x\| \cdot \|y\|}$$

Es gilt:

- $\vec{a} \bullet \vec{b} > 0 \quad \text{wenn} \quad \varphi < \frac{\pi}{2}$
- $\vec{a} \bullet \vec{b} < 0 \quad \text{wenn} \quad \varphi > \frac{\pi}{2}$

> **! Definition Orthogonalität**
>
> Sind zwei Vektoren *orthogonal / senkrecht* zueinander, ergibt das Skalarprodukt
>
> $$\vec{a} \bullet \vec{b} = 0 \quad \text{und} \quad \varphi = \frac{\pi}{2}$$

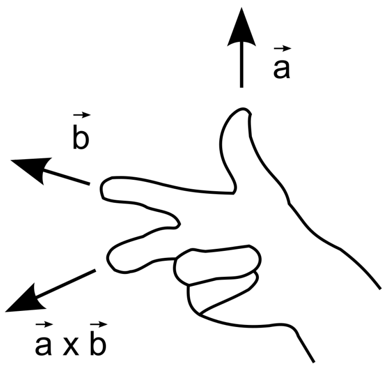
> **ℹ Richtungswinkel in $\mathbb{R}^3$**
>
> $$\cos\alpha = \frac{a_x}{a} \;\&\; \cos\beta = \frac{a_y}{a} \;\&\; \cos\gamma = \frac{a_z}{a}$$
>
> $$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1$$

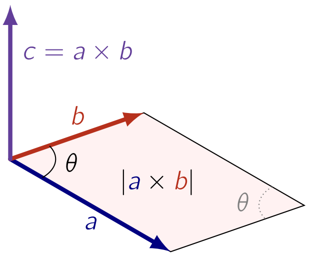
Kreuzprodukt

Mit dem Kreuzprodukt kann ein **orthogonaler** Vektor Teil eines Rechtssystems bestummen werden.

$$\vec{c} = \vec{a} \times \vec{b} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \begin{bmatrix} \mathbf{a}_y \cdot \mathbf{b}_z - \mathbf{a}_z \cdot \mathbf{b}_y \\ \mathbf{a}_z \cdot \mathbf{b}_x - \mathbf{a}_x \cdot \mathbf{b}_z \\ \mathbf{a}_x \cdot \mathbf{b}_y - \mathbf{a}_y \cdot \mathbf{b}_x \end{bmatrix}$$

Der Mittelfinger der beiden Vektoren ist das Kreuzprodukt. Je nach Betrachtung des Systems zeigt der Normalvektor in die andere Richtung





Matrizen

Schiefsymmetrische Matrix: $A = -A^T$

$$A^T = -A \quad \Rightarrow \quad \begin{bmatrix} \mathbf{0} & 7 & 23 \\ -7 & \mathbf{0} & -4 \\ -23 & 4 & \mathbf{0} \end{bmatrix}^T = \begin{bmatrix} \mathbf{0} & -7 & -23 \\ 7 & \mathbf{0} & 4 \\ 23 & -4 & \mathbf{0} \end{bmatrix}$$

Asymmetrische Matrix: $A = A^T$

$$A^T = A \quad \Rightarrow \quad \begin{bmatrix} \mathbf{0} & 1 & 2 \\ 1 & \mathbf{0} & 3 \\ 2 & 3 & \mathbf{0} \end{bmatrix}^T = \begin{bmatrix} \mathbf{0} & 1 & 2 \\ 1 & \mathbf{0} & 3 \\ 2 & 3 & \mathbf{0} \end{bmatrix}$$

Selektion Untermatrizen oder Vektoren

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

$$A(1:2, 1:2) \qquad A(:, 3)$$

Inverse Matrix/Kehrmatrix

Für 2×2 Matrizen:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

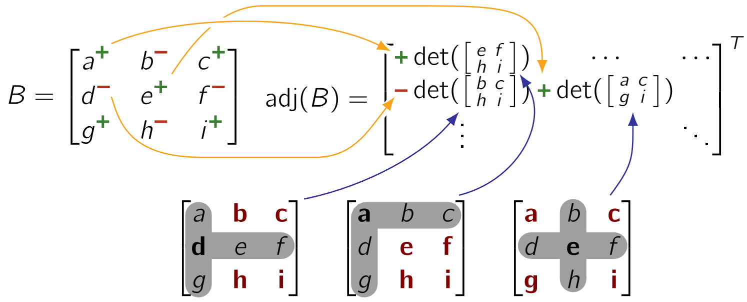
$$A^{-1} = \frac{1}{\det A} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Anhand der Adjunkte $\mathrm{adj}(B)$ um 3×3 Matrizen zu invertieren (auch grössere möglich).

$$B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\mathrm{inv}(B) = B^{-1} = \frac{1}{\det(B)} \cdot \mathrm{adj}(B) \qquad \mathrm{adj}(B) = [\mathrm{cof}(B)]^T$$

Bevor diese Berechnung gemacht werden kann, muss die Matrix auf die Invertierbarkeit geprüft werden $\rightarrow$ Determinante $\det(B) \neq 0$



$$E = \begin{bmatrix} -2 & 4 & 1 \\ 4 & -1 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

$$\mathrm{inv}(E) = E^{-1} = \frac{\mathrm{adj}(E)}{\det(E)}$$

Rotationen & Translationen

Rotationsmatrix

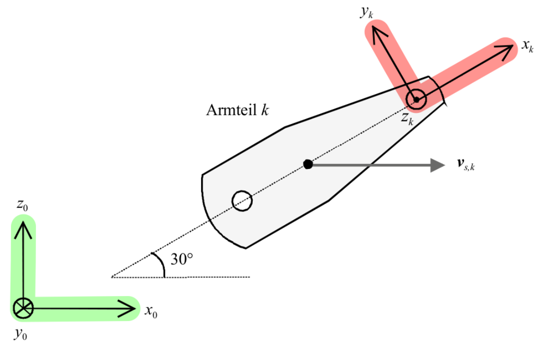
Rotationsmatrix beschreibt eine **Rotation** und wird in Form ${}^a_bR$ dargestellt. Das ${}^a_b$ beschreibt Rotationsmatrix von **a** nach **b**.

Aufbau

Die Matrix wird folgend beschrieben.

$$ {}^a_bR = \begin{bmatrix} x_a^{(b)} & y_a^{(b)} & z_a^{(b)} \end{bmatrix} $$

Beispiel



$$ v_{s,k}^{(k)} = {}^0_kT \cdot v_{s,k}^{(0)} $$

Vektor $v_{s,k}^{(0)}$ wird im 0 Koordinatensystem dargestellt, nun wird es mit ${}^0_kT$ multipliziert. Das Endprodukt ist immer noch der gleiche Vektor, einfach nun im Bezug zum Koordinatensystem k

Eigenschaften

- Zeilen- & Spaltenvektoren sind **orthogonal** zueinander
- Determinante $\det\left({}^k_0A\right) = 1$
- Betrag von Spalten & Zeilen = 1

Inverse

$$ \left({}^k_0A\right)^{-1} = {}^0_kA = {}^k_0A^T $$

Verkettung

$$ {}^k_iA = {}^{i+1}_iA \cdot {}^{i+2}_{i+1}A \cdot \dots \cdot {}^{k-1}_{k-2}A \cdot {}^k_{k-1}A $$

und einer inverse Verkettung:

$$ \begin{aligned} {}^i_kA = \left[{}^k_iA\right]^T &= {}^k_{k-1}A^T \cdot {}^{k-1}_{k-2}A^T \cdot \dots \cdot {}^{i+2}_{i+1}A^T \cdot {}^{i+1}_iA^T \\ &= {}^{k-1}_kA \cdot {}^{k-2}_{k-1}A \cdot \dots \cdot {}^{i+1}_{i+2}A \cdot {}^i_{i+1}A \end{aligned} $$

Beispiel

$$ {}^3_1A = {}^2_1A \cdot {}^3_2A $$

$$ {}^1_3A = {}^3_2A^T \cdot {}^2_1A^T = {}^2_3A \cdot {}^1_2A $$

Homogene Matrizen (Frames)

Damit die Lage des Effektors im Raum eindeutig bestimmbar ist, wird die Rotationsmatrix mit einem Verschiebungsvektor erweitert. Dadurch ist **Orientierung** und **Position** bestimmbar.

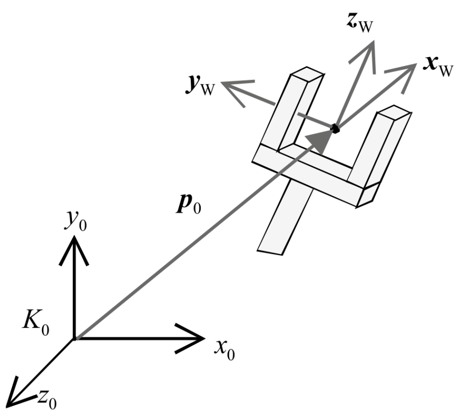
$$ a^{(k)} = {}^0_kT \cdot a^{(0)} $$

*Vom Bergspitz **0** aus den See **a** anschauen, dann via dem Wanderweg ${}^0_kT$ zum Bergspitz **k** und von dort auf denselben See **a** schauen.*

Die homogene Matrix/Frame besteht aus einer **Rotation**, einer **Translation** und einem sehr markanten **1**.

$$ {}^k_iT = \begin{bmatrix} x_k^{(i)} & y_k^{(i)} & z_k^{(i)} & p_{ik}^{(i)} \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} $$

Die Verschiebung wird genutzt um ein Frame auf eine Position zu setzen, z.B. am TCP des Werkzeugs.



Beispiel

Vektor a wird vom Koordinatensystem K_k ins System K_i überführt. Bei **Freien** Vektor wird der Wert in der vierten Zeile auf **0** gesetzt, da die Position des Vektors nicht wichtig ist → Rotation würde genügen

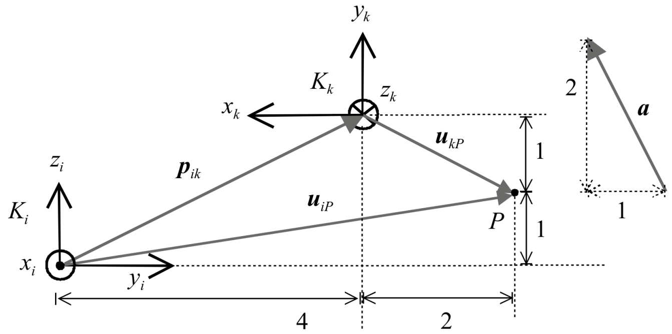
$$ {}^k_iT = \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad a^{(k)} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ \mathbf{0} \end{bmatrix} $$

$$ a^{(i)} = {}^k_iT \cdot a^{(k)} = \begin{bmatrix} 0 & 0 & -1 & \mathbf{0} \\ -1 & 0 & 0 & \mathbf{4} \\ 0 & 1 & 0 & \mathbf{2} \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 0 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 2 \\ 0 \end{bmatrix} $$

Der Nutzen des Frames ist bei Ortsvektoren ersichtlich. Bei diesen Vektoren wird der vierte Wert auf **1** gesetzt. Vektor $u_{kP}^{(k)}$ zeigt auf Punkt P und wird nun im Bezug zu K_i in $u_{iP}^{(i)}$ transformiert.

$$u_{kP}^{(k)} = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

$$u_{iP}^{(i)} = {}^k_iT \cdot u_{kP}^{(k)} = \begin{bmatrix} 0 & 0 & -1 & \mathbf{0} \\ -1 & 0 & 0 & \mathbf{4} \\ 0 & 1 & 0 & \mathbf{2} \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} -2 \\ -1 \\ 0 \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 1 \\ 1 \end{bmatrix}$$



Während ein freier Vektor sich nicht verändert, ändern sich Ortsvektoren zu komplett neuen Vektoren ($|u_{kP}^{(k)}| \neq |u_{iP}^{(i)}|$).

Inverse

Zu **jeder** homogenen Matrix ist **immer** die inverse homogene Matrix gegeben und es gilt speziell:

$${}^i_kT = ({}^k_iT)^{-1} = \begin{bmatrix} {}^k_iA^T & -{}^k_iA^T \cdot p_{ik}^{(i)} \\ [0 \quad 0 \quad 0] & 1 \end{bmatrix}$$

Verkettung

$${}^k_iT = {}^{i+1}_{i}T \cdot {}^{i+2}_{i+1}T \cdot \dots \cdot {}^{k-1}_{k-2}T \cdot {}^{k}_{k-1}T$$

und einer inverse Verkettung:

$$\begin{aligned} {}^i_kT = \left[{}^k_iT\right]^{-1} &= {}^{k}_{k-1}T^{-1} \cdot {}^{k-1}_{k-2}T^{-1} \cdot \dots \cdot {}^{i+2}_{i+1}T^{-1} \cdot {}^{i+1}_{i}T^{-1} \\ &= {}^{k-1}_{k}T \cdot {}^{k-2}_{k-1}T \cdot \dots \cdot {}^{i+1}_{i+2}T \cdot {}^{i}_{i+1}T \end{aligned}$$

⚠ $]^{-1}$ anstatt $]^T$

Da die homogene Matrix nicht einfach so transponiert werden darf, muss die spezielle Umformung verwendet werden!

Orienterung durch Euler-Winkel

Eine Drehung eines Frames im Bezug zu einem anderen ist durch **drei** Winkel angegeben. Diese drei Winkel werden **Euler-Winkel** genannt, wenn diese nacheinander ausgeführt werden.

$$R_{ZYX} = {}^W_RA = \begin{pmatrix} x_W^{(R)} & y_W^{(R)} & z_W^{(R)} \end{pmatrix} = R_Z \cdot R_Y \cdot R_X$$

$$= \begin{bmatrix} C_A & -S_A & 0 \\ S_A & C_A & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_B & 0 & S_B \\ 0 & 1 & 0 \\ -S_B & 0 & C_B \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_C & -S_C \\ 0 & S_C & C_C \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{C_A \cdot C_B} & -S_A \cdot C_C + C_A \cdot S_B \cdot S_C & S_A \cdot S_C + C_A \cdot S_B \cdot C_C \\ \mathbf{S_A \cdot C_B} & C_A \cdot C_C + S_A \cdot S_B \cdot S_C & -C_A \cdot S_C + S_A \cdot S_B \cdot C_C \\ \mathbf{-S_B} & \mathbf{C_B \cdot S_C} & \mathbf{C_B \cdot C_C} \end{bmatrix}$$

$$\begin{aligned} A &= \operatorname{atan2}(\mathbf{R_{21}}, \mathbf{R_{11}}) \\ B &= \arcsin(-\mathbf{R_{31}}) \\ C &= \operatorname{atan2}(\mathbf{R_{32}}, \mathbf{R_{33}}) \end{aligned}$$

! Stäubli vs. KUKA

Stäubli haben eine X, Y', Z''-Reihenfolge ($R_X(\alpha) \cdot R_Y(\beta) \cdot R_Z(\gamma)$)...

$$R_{XYZ} = {}^W_RA = \begin{pmatrix} x_W^{(R)} & y_W^{(R)} & z_W^{(R)} \end{pmatrix} = R_X(A) \cdot R_Y(B) \cdot R_Z(C)$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_A & -S_A \\ 0 & S_A & C_A \end{bmatrix} \cdot \begin{bmatrix} C_B & 0 & S_B \\ 0 & 1 & 0 \\ -S_B & 0 & C_B \end{bmatrix} \cdot \begin{bmatrix} C_C & -S_C & 0 \\ S_C & C_C & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{C_B \cdot C_C} & \mathbf{-C_B \cdot S_C} & \mathbf{S_B} \\ C_A \cdot S_C + S_A \cdot S_B \cdot C_C & C_A \cdot C_C - S_A \cdot S_B \cdot S_C & \mathbf{-S_A \cdot C_B} \\ S_A \cdot S_C - C_A \cdot S_B \cdot C_C & C_A \cdot S_B \cdot S_C + S_A \cdot C_C & \mathbf{C_A \cdot C_B} \end{bmatrix}$$

$$\begin{aligned} A &= \operatorname{atan2}(-\mathbf{R_{23}}, \mathbf{R_{33}}) \\ B &= \arcsin(\mathbf{R_{13}}) \\ C &= \operatorname{atan2}(-\mathbf{R_{12}}, \mathbf{R_{11}}) \end{aligned}$$

...und **KUKA** haben eine Z, Y', X''-Reihenfolge ($R_Z(\gamma) \cdot R_Y(\beta) \cdot R_Z(\alpha)$). Die Gleichungen für diese ist oben angegeben.

Beispiel

Folgend ein Beispiel, welches die Darstellung von

$$\begin{array}{cccc} {}^1_0A = & \begin{bmatrix} -0.9397 \\ 0 \\ -0.3420 \end{bmatrix} & \begin{bmatrix} -0.3420 \\ 0 \\ 0.9397 \end{bmatrix} & \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} x_0 \\ y_0 \\ z_0 \end{matrix} \\ & \mathbf{x}_1^{(0)} & \mathbf{y}_1^{(0)} & \mathbf{z}_1^{(0)} \end{array}$$

$${}^1_0A = R_Z(A) \cdot R_Y(B) \cdot R_X(C)$$

$$A_{1\to 0} = \text{atan2}(0, -0.9397) = 180°$$
$$B_{1\to 0} = \arcsin(-(-0.3420)) \approx 20°$$
$$C_{1\to 0} = \text{atan2}(-0.9397, 0) = -90°$$

Tipp

Da cos eine gerade Funktion ist, ergeben $\cos(\pm\alpha)$ den gleichen Wert, z.B. $\cos(\pm 45°) = \frac{1}{\sqrt{2}}$
sin ist ungerade und daher ist der sin-Wert immer verkehrt $\sin(\pm\alpha) = \mp a$

$$\text{atan2}(x, y) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{wenn } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{wenn } x < 0 \text{ und } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{wenn } x < 0 \text{ und } y < 0, \\ +\frac{\pi}{2} & \text{wenn } x = 0 \text{ und } y > 0, \\ -\frac{\pi}{2} & \text{wenn } x = 0 \text{ und } y < 0, \\ \text{undefiniert} & \text{wenn } x = 0 \text{ und } y = 0. \end{cases}$$

Roll Pitch Yaw (Roll Neigung Gier)

- Drehung um die x-**Basis**achse mit dem Winkel ψ (**yaw**)
- Drehung um die y-**Basis**achse mit dem Winkel θ (**pitch**)
- Drehung um die z-**Basis**achse mit dem Winkel ϕ (**roll**)

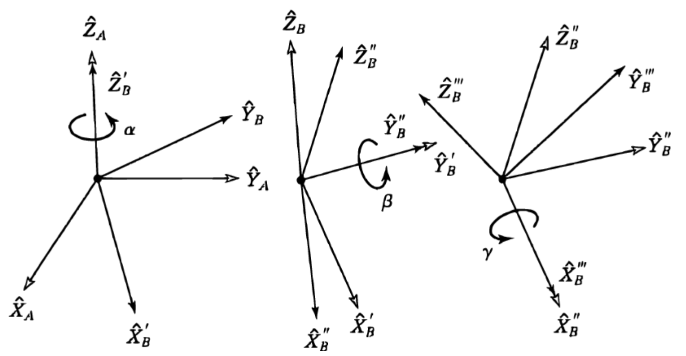
$$\mathbf{R}(\psi, \theta, \phi) = {}^W_R A = \begin{pmatrix} x_W^{(R)} & y_W^{(R)} & z_W^{(R)} \end{pmatrix}$$
$$= \begin{bmatrix} C_\phi \cdot C_\theta & -S_\phi \cdot C_\psi + C_\phi \cdot S_\theta \cdot S_\psi & S_\phi \cdot S_\psi + C_\phi \cdot S_\theta \cdot C_\psi \\ S_\phi \cdot C_\theta & C_\phi \cdot C_\psi + S_\phi \cdot S_\theta \cdot S_\psi & -C_\phi \cdot S_\psi + S_\phi \cdot S_\theta \cdot C_\psi \\ -S_\theta & C_\theta \cdot S_\psi & C_\theta \cdot C_\psi \end{bmatrix}$$

$$\phi = \text{atan2}(R_{21}, R_{11})$$
$$\theta = \arcsin(-R_{31})$$
$$\psi = \text{atan2}(R_{32}, R_{33})$$

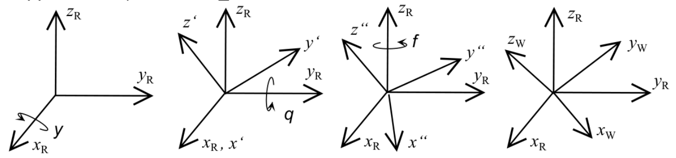
Unterschied RPY & Euler

Während bei Euler Drehungen immer vom neuen Koordinatensystem gedreht wird (**intrinsisch**), werden bei RPY Drehungen immer vom Ursprünglichen Koordinatensystem gedreht (**extrinsisch**).

Euler Winkel
$R_Z \rightarrow R_{Y'} \rightarrow R_{X''}$

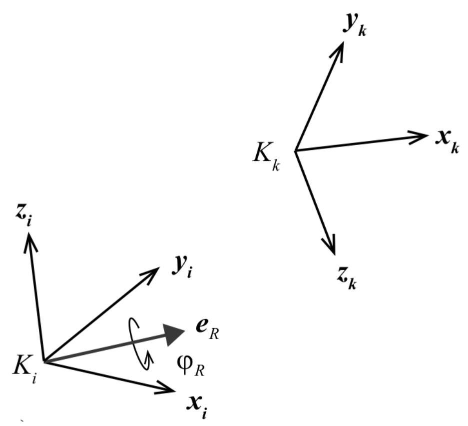


Roll Pitch Yaw
$R_X \rightarrow R_Y \rightarrow R_Z$



Orientierung durch Drehvektor und -winkel

Drehung wird um einen Drehvektor gemacht → den *direkten* Weg



$$\varphi_R = 2 \cdot \arccos(qt_1) \qquad e_R = \begin{bmatrix} qt_2 \\ qt_3 \\ qt_4 \end{bmatrix} / \sin(0.5 \cdot \varphi_R)$$

Da die Ermittelung des Drehvektors e_R und des Drehwinkels φ_R aufwendig ist, wird meistens mit Euler-Winkeln gearbeitet. Ein **Vorteil** von Drehvektor & -winkel bei der Bahnsteuerung/kontinuierliche Veränderung → Dies wird auch mit **Quarternions** gemacht!

Quaternion

Mit Quaternion können Drehungen in kompakter Form dargestellt werden, analog zu komplexen Zahlmultiplikation für Rotationen. Zusätzlich gibt es kein *Gimbal-Lock* (Zwei Rotationsachsen sind gleich und daher hat man ein Freiheitsgrad weniger). In AROB wird mit Einheitsquaternion gearbeitet: $\sqrt{qt_1^2 + qt_2^2 + qt_3^2 + qt_4^2} = 1$.

$$q = \begin{bmatrix} qt_1 \\ qt_2 \\ qt_3 \\ qt_4 \end{bmatrix} = \underbrace{qt_1}_{\text{real/Skalar Komp.}} \underbrace{+\, qt_2\ i + qt_3\ j + qt_4\ k}_{\text{imaginär/Vektor Komp.}}$$

Umwandlungen $Q \leftrightarrow EA \leftrightarrow {}^k_iA$

Euler zu Quartenion

$A = \psi \qquad B = \theta \qquad C = \phi$

$$qt = \begin{bmatrix} \cos(\phi/2)\cdot\cos(\theta/2)\cdot\cos(\psi/2) + \sin(\phi/2)\cdot\sin(\theta/2)\cdot\sin(\psi/2) \\ \sin(\phi/2)\cdot\cos(\theta/2)\cdot\cos(\psi/2) - \cos(\phi/2)\cdot\sin(\theta/2)\cdot\sin(\psi/2) \\ \cos(\phi/2)\cdot\sin(\theta/2)\cdot\cos(\psi/2) + \sin(\phi/2)\cdot\cos(\theta/2)\cdot\sin(\psi/2) \\ \cos(\phi/2)\cdot\cos(\theta/2)\cdot\sin(\psi/2) - \sin(\phi/2)\cdot\sin(\theta/2)\cdot\cos(\psi/2) \end{bmatrix}$$

Quartenion zu Euler

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}(2\cdot(qt_1\cdot qt_2 + qt_3\cdot qt_4),\, qt_1^2 - qt_2^2 - qt_3^2 + qt_4^2) \\ \arcsin(2\cdot(qt_1\cdot qt_3 - qt_2\cdot qt_4)) \\ \operatorname{atan2}(2\cdot(qt_1\cdot qt_4 + qt_2\cdot qt_3),\, qt_1^2 + qt_2^2 - qt_3^2 - qt_4^2) \end{bmatrix}$$

Euler zu Rotationsmatrix

Von Euler Rotation abhängig, folgend mit intrinsisch ZYX:

$$R_{ZYX} = {}^W_RA = \begin{pmatrix} x_W^{(R)} & y_W^{(R)} & z_W^{(R)} \end{pmatrix} = \underbrace{R_Z(A)}_{3.}\cdot\underbrace{R_Y(B)}_{2.}\cdot\underbrace{R_X(C)}_{1.}$$

$$= \underbrace{\begin{bmatrix} C_A & -S_A & 0 \\ S_A & C_A & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Z-Rotation}} \cdot \underbrace{\begin{bmatrix} C_B & 0 & S_B \\ 0 & 1 & 0 \\ -S_B & 0 & C_B \end{bmatrix}}_{\text{Y-Rotation}} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_C & -S_C \\ 0 & S_C & C_C \end{bmatrix}}_{\text{X-Rotation}}$$

$$= \begin{bmatrix} C_A\cdot C_B & -S_A\cdot C_C + C_A\cdot S_B\cdot S_C & S_A\cdot S_C + C_A\cdot S_B\cdot C_C \\ S_A\cdot C_B & C_A\cdot C_C + S_A\cdot S_B\cdot S_C & -C_A\cdot S_C + S_A\cdot S_B\cdot C_C \\ -S_B & C_B\cdot S_C & C_B\cdot C_C \end{bmatrix}$$

Rotationsmatrix zu Euler

Von Euler Rotation abhängig, folgend mit intrinsisch ZYX:

$$R_{ZYX} = {}^W_RA = \begin{pmatrix} x_W^{(R)} & y_W^{(R)} & z_W^{(R)} \end{pmatrix} = \underbrace{R_Z(A)}_{3.}\cdot\underbrace{R_Y(B)}_{2.}\cdot\underbrace{R_X(C)}_{1.}$$

$$= \begin{bmatrix} C_A & -S_A & 0 \\ S_A & C_A & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_B & 0 & S_B \\ 0 & 1 & 0 \\ -S_B & 0 & C_B \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_C & -S_C \\ 0 & S_C & C_C \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{C_A\cdot C_B} & -S_A\cdot C_C + C_A\cdot S_B\cdot S_C & S_A\cdot S_C + C_A\cdot S_B\cdot C_C \\ \mathbf{S_A\cdot C_B} & C_A\cdot C_C + S_A\cdot S_B\cdot S_C & -C_A\cdot S_C + S_A\cdot S_B\cdot C_C \\ \mathbf{-S_B} & \mathbf{C_B\cdot S_C} & \mathbf{C_B\cdot C_C} \end{bmatrix}$$

$$A = \operatorname{atan2}(\mathbf{R_{21}}, \mathbf{R_{11}})$$
$$B = \arcsin(-\mathbf{R_{31}})$$
$$C = \operatorname{atan2}(\mathbf{R_{32}}, \mathbf{R_{33}})$$

Quaternion zu Rotationsmatrix

$${}^W_0A = \begin{bmatrix} \mathbf{x}_W^{(0)} & \mathbf{y}_W^{(0)} & \mathbf{z}_W^{(0)} = \end{bmatrix} \qquad x_W^{(0)} = \begin{bmatrix} 2\cdot(qt_1^2 + qt_2^2) - 1 \\ 2\cdot(qt_2\cdot qt_3 + qt_1\cdot qt_4) \\ 2\cdot(qt_2\cdot qt_4 - qt_1\cdot qt_3) \end{bmatrix}$$

$$y_W^{(0)} = \begin{bmatrix} 2\cdot(qt_2\cdot qt_3 - qt_1\cdot qt_4) \\ 2\cdot(qt_1^2 + qt_3^2) - 1 \\ 2\cdot(qt_3\cdot qt_4 + qt_1\cdot qt_2) \end{bmatrix} \qquad z_W^{(0)} = \begin{bmatrix} 2\cdot(qt_2\cdot qt_4 + qt_1\cdot qt_3) \\ 2\cdot(qt_3\cdot qt_4 - qt_1\cdot qt_2) \\ 2\cdot(qt_1^2 + qt_4^2) - 1 \end{bmatrix}$$

Rotationsmatrix zu Quaternion

$${}^k_iA = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

$$\begin{bmatrix} qt_1 \\ qt_2 \\ qt_3 \\ qt_4 \end{bmatrix} = \begin{bmatrix} 0.5\cdot\sqrt{A_{11} + A_{22} + A_{33} + 1} \\ 0.5\cdot\operatorname{sgn}(A_{32} - A_{23})\cdot\sqrt{A_{11} - A_{22} - A_{33} + 1} \\ 0.5\cdot\operatorname{sgn}(A_{13} - A_{31})\cdot\sqrt{A_{22} - A_{33} - A_{11} + 1} \\ 0.5\cdot\operatorname{sgn}(A_{21} - A_{12})\cdot\sqrt{A_{33} - A_{11} - A_{22} + 1} \end{bmatrix}$$

> **i** Vorzeichenfunktion $\operatorname{sgn}(x)$
>
> $$\operatorname{sgn}(x) = \begin{cases} +1 & \text{für } x > 0 \\ 0 & \text{für } x = 0 \\ -1 & \text{für } x < 0 \end{cases}$$

Denavit-Hartenberg-Konvention

Die DH-Konvention beschreibt die Achsen-Zusammenhänge mit vier Parametern (zwei Rotationen und zwei Translationen), welche in folgender Reihenfolge abgearbeitet wird.

1. $\theta_i/°$: Drehung an der z_i-Achse
2. d_i/m: Verschiebung Richtung z_i-Achse
3. a_i/m: Verschiebung Richtung x_i-Achse
4. $\alpha_i/°$: Drehung an der x_i-Achse

Allgemein

Jedes Armteil $i (i = 0, 1, \ldots, n)$ eines Robotes mit n Gelenken wird mit einem Koordinatensystem K_i versehen. K_i ist mit Armteil i fest verbunden.

Die DH Parameter sind so auszulegen, das man K_i in K_{i+1} überführt werden kann.

Festlegung Weltsystem K_0

- K_0 ist fest mit ruhender Basis (AT0) verbunden
- **Positon von $\mathbf{K_0}$** irgendwo auf 1.Gelenkachse → möglichst nahe AT1
- z_0-**Achse** zeigt entlang 1.Gelenkachse → x_0- & y_0-Achse

frei wählbar solange **Rechtssystem**

Festlegung $K_i (i = 1, 2, \ldots, n-1)$

Allgemein: K_i liegt auf Gelenkachse $i+1$

Position von K_i

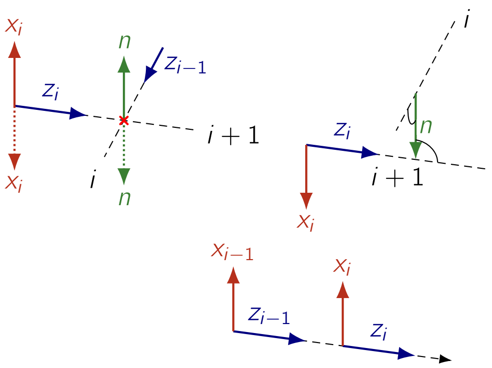
- K_i liegt im Schnittpunkt von Gelenkachse i & $i+1$
- **oder** verlaufen Gelenkachsen i & $i+1$ parallel, dann K_i irgendwo auf Gelenkachse $i+1$
  - sinnvoll: zuerst K_{i+1} festlegen, danach K_i so legen, dass Abstand minimal ist.
- **oder** Gelenkachsen i & $i+1$ **nicht** parallel und **nicht** schneidend, dann K_i auf Schnittpunkt der Normalen der Gelenkachsen setzen.

z_i-Achse von K_i

- z_i-Achse entlang der Gelenkachse $i+1 \rightarrow$ 2 Möglichkeiten

x_i-Achse von K_i

- Wenn z_{i-1} & z_i-Achse schneiden: x_i-Achse parallel zum Kreuzprodukt von $z_{i-1} \times z_i \rightarrow$ 2 Möglichkeiten
- Wenn z_{i-1} & z_i-Achse **nicht** schneiden: x_i-Achse entlang gemeinsame Normalen von Achsen i & $i+1$
- z_{i-1}-Achse auf z_i-Achse: x_i-Achse $= x_{i-1}$-Achse



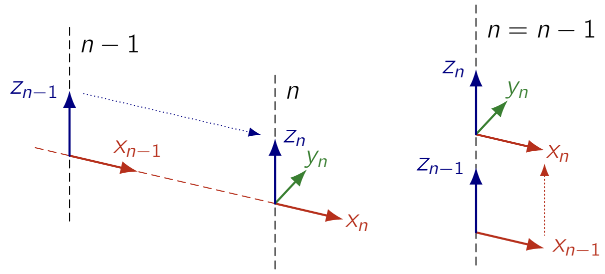
Festlegung K_n

z_n-Achse

Die z_n-Achse wird in Richtung der z_{n-1}-Achsedurch den TCP gelegt.

x_n-Achse

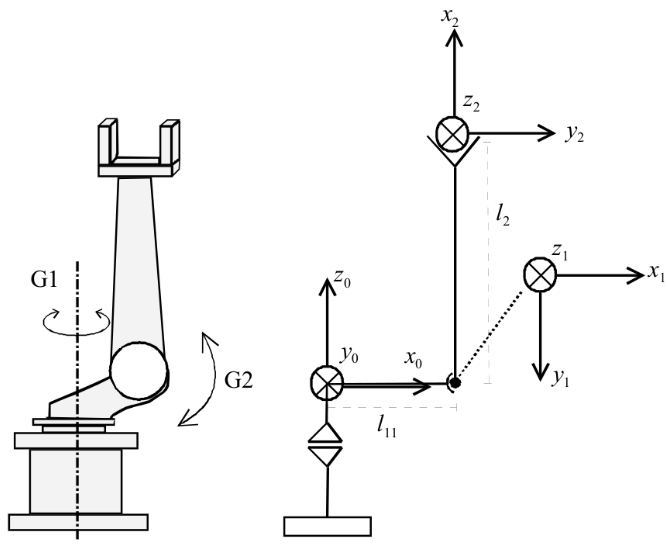
- z_n-Achse separat von z_{n-1}: x_n-Achse zeigt von Gelenkachse $n-1$ zu n.
- z_n-Achse auf z_{n-1}-Achse: x_n-Achse zeigt in gleiche Richtung wie x_{n-1}-Achse



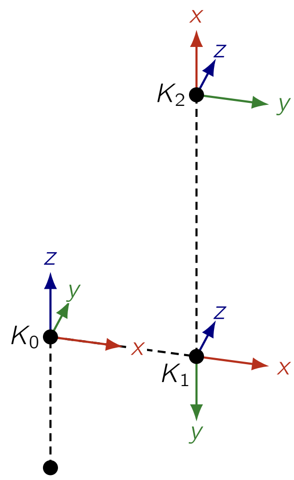
Beispiel

Folgender Roboter wird in DH Parametern beschrieben.

1. Weltsystem K_0 bestimmen (z.B. Ende des 1.Gelenkes)
2.



Gelenk	$\theta_i/°$	d_i/m	a_i/m	$\alpha_i/°$
1	0	0	l_{11}	-90
2	-90	0	l_2	0



Als Homogene Matrix

$$
{}^{i}_{i-1}T = \begin{bmatrix} x_i^{(i-1)} & y_i^{(i-1)} & z_i^{(i-1)} & p_{i-1,i}^{(i-1)} \\ 0 & 0 & 0 & 1 \end{bmatrix}
= \begin{bmatrix} C_{\theta_i} & -S_{\theta_i} \cdot C_{\alpha_i} & S_{\theta_i} \cdot S_{\alpha_i} & a_i \cdot C_{\theta_i} \\ S_{\theta_i} & C_{\theta_i} \cdot C_{\alpha_i} & -C_{\theta_i} \cdot S_{\alpha_i} & a_i \cdot S_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Vorwärtstransformation

Sind die Gelenkkoordinaten (Winkel) bekannt, kann die Lage des TCP und Orientierung des Effektors eindeutig beschrieben werden.

$$
T_W(q) = {}^{n}_{0}T(q) = {}^{1}_{0}T(q_1) \cdot {}^{2}_{1}T(q_2) \cdots {}^{n-1}_{n-2}T(q_{n-1}) \cdot {}^{n}_{n-1}T(q_n)
$$

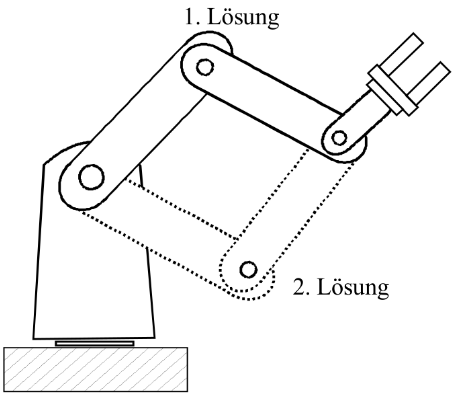
Ist ein Ortsvektor $u_{(i)}$, welcher mit Armteil i verbunden ist und auf Punkt P zeigt, gegeben, kann dieser im Raum berechnet (4.Komponente $= 1$):

$$
u_{0P}^{(0)} = {}^{i}_{0}T(q) \cdot u_{iP}^{(i)} = {}^{1}_{0}T(q_1) \cdot {}^{2}_{1}T(q_2) \cdots {}^{i}_{i-1}T(q_i) \cdot u(i)_{iP}
$$

Rückwärtstransformation

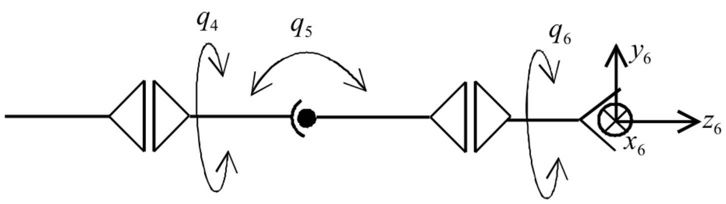
Mehrdeutigkeit

Mehrere Lösungen von Gelenkkoordinaten für die Lage des TCP (Orientierung & Position).



Singularität

Beispiel: q_4 & q_6 verfügen über unendlich viele Lösungen.

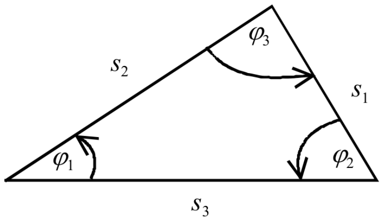


Ein Ansatz wäre das **Gütekriterium**, in welche man eine Lösung nimmt, welche am nächsten zur vorhergehenden Lösung liegt.

$$
f(q_4, q_6) = c_1 \cdot (q_{4,A} - q_4)^2 + c_2 \cdot (q_{6,A} - q_6)^2 \stackrel{!}{=} \text{MIN}
$$

Geometrischer Ansatz

Segmente in Dreiecke umwandeln und danach Kosinussatz anwenden!



$$
s_1^2 = s_2^2 + s_3^2 - 2 \cdot s_2 \cdot s_3 \cdot \cos\varphi_1
$$
$$
s_2^2 = s_1^2 + s_3^2 - 2 \cdot s_1 \cdot s_3 \cdot \cos\varphi_2
$$
$$
s_3^2 = s_1^2 + s_2^2 - 2 \cdot s_1 \cdot s_2 \cdot \cos\varphi_3
$$

Kinematische Transformationen mit der Jacobi-Matrix

Software mit Schneebeli

Offline vs. Online Programmierung

Offline Programmierung

Vorteile

- Fahrwegoptimierung
- Grosse Punktmengen
- Komplexe Geometrien
- Erschliessung neuer Aufgaben für Roboter
- Kollisionsprüfung
- Simulation der Bewegungen
- Fahrbereichsabklärungen beim Auslegen der Anlage
- Änderungen sind einfacher zu handhaben
- Frühzeitiges Erkennen von Planungsfehlern
- Erstellung der Programme ist möglich, während die Anlage noch aufgebaut oder benutzt wird
- Strukturierung der Programme ist einfacher
- Einfachere Dokumentation
- Flächenprogrammierung
- Häufige Änderungen am Prozess können besser abgebildet werden

Nachteile

- Abhängig von der Genauigkeit des Roboters
- Abhängig von der Genauigkeit der CAD-Modelle
- Bewegliche Teile am Roboter können nicht simuliert werden
- Spezielle (teure) Software notwendig
- Qualifizierte Programmierer sind notwendig
- Unterschiedliche Programmiersprachen der Hersteller

Online Programmierung

Vorteile

- Schneller fertig bei einfachen Bewegungen
- Keine Genauigkeitsprobleme, da Wiederholgenauigkeit hoch ist
- Intuitives Verfahren
- Schnelles, direktes Feedback

Nachteile

- Roboter ist während Teach-In nicht produktiv
- Potential für Chaosprogrammierung vorhanden

Qualitätsmerkmale eines guten Roboterprogramms

Lesbarkeit

Das Programm muss so geschrieben sein, dass eine Person – welche etwas von Roboter-Programmierung versteht und eine Ahnung von der Aufgabe des Roboters hat – das Programm mit **vernünftigem Aufwand verstehen** und **ändern** kann.

- Verwendung von sprechenden Namen: `pointHome/pHome`, `dioBulkheadOpen`
- Einhalten von Standards: z.B. Code Guidelines
- Verwendung von sinnvollen Kommentaren: *"Say what you mean, simply and directly."* - Brian Kernighan

Wiederverwendbarkeit

In der Praxis zeigt sich, dass einzelne Teile der Programme oft verwendet werden können, andere für jede Anlage individuell angepasst werden müssen. → Modularisierung

- Was unabhängig voneinander ändern kann, muss getrennt werden. [IDesign Methode]
- Eine gute Strukturierung der Programme hilft zudem, den Überblick zu behalten und beschleunigt auch die Programmierung.

Wenige unterschiedliche Ablaufpfade (tiefe Komplexität)

Eine tiefe Komplexität hilft, dass das Programm verständlich bleibt und dass es einfacher zu erweitern und pflegen ist. Tiefe Komplexität heisst, dass es möglichst wenige unterschiedliche Pfade durch das Programm gibt.

- Ein Roboter sollte eine klar definierte Aufgabe haben.
- Möglichst wenige 'If', 'Switch' und andere Bedingungen einbauen.
- Schachtelung von Unterscheidungsstatements möglichst vermeiden.
- Keinen Copy/Paste-Code verwenden.
- Unterteilung in Submodule
- Arbeiten mit Variablen

Intelligente Bewegungen

Der Roboter soll möglichst natürliche und keine mechanischen Bewegungen ausführen.

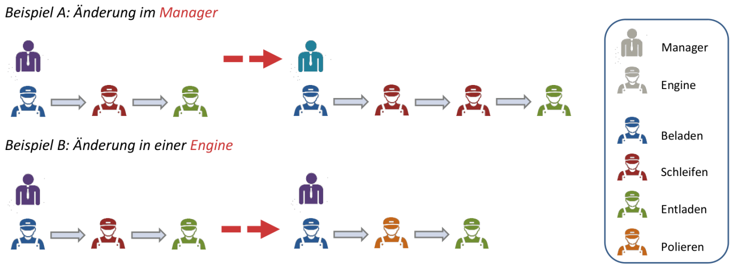
- Ein genereller Ansatz ist, die Beschleunigungen aller Achsen möglichst tief zu halten.
- Weitere Optionen:
  - Kontinuierliche Bewegungen (Punkte und Orientierung)
  - Keine unnötigen Fahrpunkte einbauen
  - Kürzeste Wege suchen
  - Nachbearbeiten von nicht optimalen Bewegungen

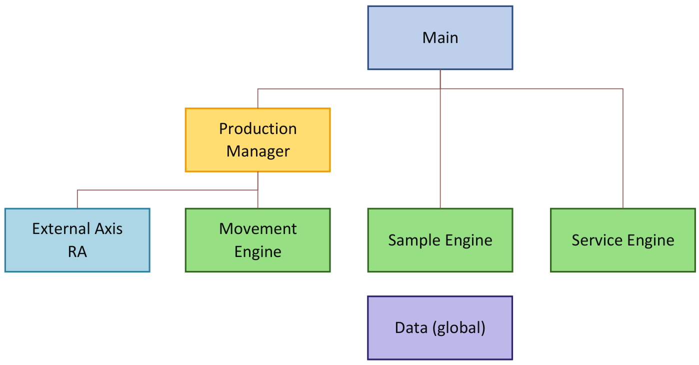
Roboter erfüllt den Auftrag

- Jeder mögliche Ablaufpfad muss getestet werden.
- Qualität des Produktes muss stimmen.
- Kundenzufriedenheit ist wichtig.
- Wow-Effekt ist wertvoll. [Wow-Effekt bedeutet, dass man mit geringem Aufwand dem Kunden einen Mehrwert liefert, indem man das Programm etwas effizienter, genauer, etc. macht]

Struktur

Manager	Engines
Organisieren den Ablauf	Rechnen
Verwalten die Daten	Bewegen
Treffen Entscheidungen	Messen
...	...





Werkzeug Ausrichtung