Zusammenfassung - Robotik

Julian Shen

9. Juni 2023

1 Mathematische Grundlagen

Kinematik ist die reine geometrische Beschreibung von Bewegung eines Manipulators oder Roboters. Das essentielle Konzept ist die **Position**.

Statik behandelt Kräfte und Momente, die sich auf einen ruhenden Mechanismus auswirken. Das essentielle Konzept ist die **Steifigkeit**.

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

Freiheitsgrade (DoF) ist die Anzahl unabhängiger Parameter, die zur kompletten Spezifikation der Lage eines Objekts benötigt werden, z.B. Starrkörper hat in 2D 3 DoF und in 3D 6 DoF.

Starrkörperbewegungen werden durch zwei Eigenschaften charakterisiert:

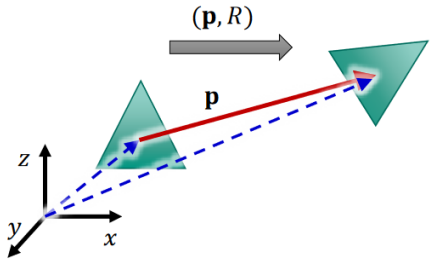
1. Distanz zweier beliebiger Punkte ist konstant
2. Orientierungen im Körper bleiben erhalten

SO(3) und SE(3):

- SO(3): **Spezielle Orthogonale Gruppe**, die **Rotationen** repräsentiert
- SE(3): **Spezielle Euklidische Gruppe**, die **Transformationen** repräsentiert
- Elemente aus SO(3) werden als reale 3×3 orthogonale Matrizen R (Zeilen- und Spaltenvektoren orthonormal) beschrieben und erfüllen

$$R^\top R = 1 \qquad \text{mit} \qquad \det(R) = 1$$

- Elemente aus SE(3) sind von der Form $(\mathbf{p}, R)$ mit $\mathbf{p} \in \mathbb{R}^3$ und $R \in \mathrm{SO}(3)$ und beschreiben Verknüpfungen von Rotationen und Translationen



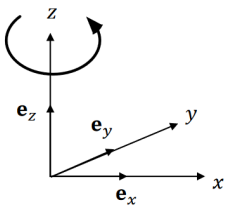
Euklidischer Raum: Vektorraum $\mathbb{R}^3$ mit dem Skalarprodukt.

- Punkt **a** im euklidischen Raum wird durch Vielfache der Einheitsvektoren $\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z}$ beschrieben
- Wir benutzen **rechtsdrehende Koordinatensysteme**

Rechtsdrehendes Koordinatensystem

$\mathbf{e}_x \times \mathbf{e}_y = \mathbf{e}_z$

$\mathbf{x} \times \mathbf{y} = \mathbf{z}$

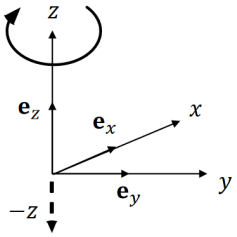


Linksdrehendes Koordinatensystem

$\mathbf{e}_x \times \mathbf{e}_y = -\mathbf{e}_z$

$\mathbf{x} \times \mathbf{y} = -\mathbf{z}$

× : Kreuzprodukt



Lineare Abbildungen (Transformationen), die den euklidischen Raum auf sich selbst abbilden, nennt man **Endomorphismen**:

$$\phi(\cdot)\colon \mathbb{R}^3 \to \mathbb{R}^3$$

- Endomorphismen können durch quadratische Matrizen repräsentiert werden:

$$\phi(\mathbf{a}) = A \cdot \mathbf{a}, \qquad A \in R^{3\times 3}$$

- A beschreibt einen Basiswechsel zwischen den originalen Basisvektoren $\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z}$ und den neuen Basisvektoren $\mathbf{e'_x}, \mathbf{e'_y}, \mathbf{e'_z}$:

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

Bijektive Endomorphismen nennt man **Isomorphismen**.

- Eigenschaften:
  1. Winkel bleiben erhalten
  2. Längen bleiben erhalten
  3. Händigkeit beleibt erhalten
- Eine spezielle Art von Isomorphismen ist die **Rotationsgruppe** SO(3)

Rotationsgruppe SO(3):

- SO(3) ist nicht kommutativ: $A \cdot B \cdot \mathbf{x} \neq B \cdot A \cdot \mathbf{x}$ mit $\mathbf{x} \in \mathbb{R}^3$ und $A, B \in \mathrm{SO}(3)$
- Für alle $R \in \mathrm{SO}(3)$ ist $R^{-1} = R^\top$, die Inverse kann also leicht berechnet werden

Rotationen in 2D:

- Rotation in der xy-Ebene um $(0,0)$ ist eine **lineare Transformation**
- **Rotationsmatrix**: $R_\alpha(\mathbf{x}) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \mathbf{x}$ mit $RR^\top = R^\top R = I$ und $\det(R) = 1$
- Rotation um einen Punkt $\mathbf{c} \neq (0,0)$ ist keine lineare Transformation. Verschiebe dafür die Ebene um $-\mathbf{c}$, rotiere und verschiebe wieder um $+\mathbf{c}$ zurück:

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

- $R_{\mathbf{c},\alpha}$ ist eine nichtlineare Transformation und heißt **affine Transformation**. Sie unterscheidet sich von R_α nur durch das Addieren einer Konstante

Rotationen in 3D:

- Eine 2D Rotation in der xy-Ebene ist eine 3D Rotation um die z-Achse:

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

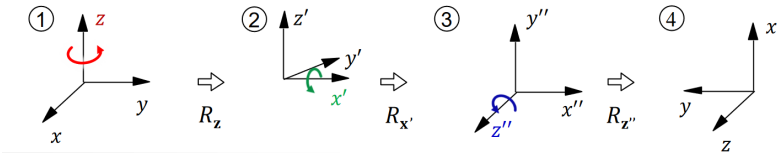
- Rotationen können verkettet werden: $\phi_{\mathbf{z},\gamma}\left(\phi_{\mathbf{y},\beta}\left(\phi_{\mathbf{x},\alpha}(\mathbf{a})\right)\right), \qquad \mathbf{a} \in \mathbb{R}^3$

Probleme mit Rotationsmatrizen:

- **Redundanz**: Neun Werte für eine Rotationsmatrix
- Probleme im Bereich des maschinellen Lernens

Eulerwinkel:

- Es ist möglich jede Rotation durch drei Rotationen um jeweils eine Rotationsachse darzustellen
- **Euler-Konvention**: **z x′ z″** (lokale Drehung, Drehung verändert Achsen) oder **x y z** (globale Drehung, Drehung um feste Achsen)
- Winkel α, β, γ sind **Eulerwinkel** und beschreiben den Grad der Drehungen



- **Vorteile**: Kompakter und aussagekräftiger als Rotationsmatrizen

- **Nachteile**:
  - Nicht eindeutig: In der Euler-Konvention $\mathbf{x}\ \mathbf{y}'\ \mathbf{z}''$ beschreiben die Eulerwinkel $(45°, -90°, 45°)$ und und $(30°, -90°, 60°)$ die gleiche Rotation
  - Nicht kontinuierlich: Kleine Änderung in der Orientierung können zu großen Änderungen der Eulerwinkel führen
  - **Gimbal Lock**: Bei bestimmten Winkeln werden zwei Achsen voneinander abhängig ⇒ Ein Freiheitsgrad geht verloren

Bewertung der Darstellung von Orientierung mit 3×3-Matrizen:

- **Vorteil**: Vektor und Rotationsmatrix sind anschaulich ⇒ übliche Form der Eingabe von Posen
- **Nachteil**: Darstellung als $(\mathbf{p}, R)$ mit $\mathbf{p} \in \mathbb{R}^3$ und $R \in \mathrm{SO}(3)$ führt dazu, dass Translation und Rotation getrennt durchgeführt werden müssen

→ **Ziel**: Geschlossene Darstellung von Rotation und Translation in einer Matrix

Affine Transformationen:

- Der **affine Raum** ist eine Erweiterung zum euklidischen Raum
- Beinhaltet Vektoren, die in **erweiterten, homogenen Koordinaten** ausgedrückt werden: $a = (a_x \quad a_y \quad a_z \quad h)^\top, h \in \{0, 1\}$, wobei a für $h = 0$ einen Ortsvektor und für $h = 1$ einen Richtungsvektor beschriebt
- Für Rotationsmatrix R und Translation $\mathbf{t}$ gilt nun:

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

  womit also nun Translation und Rotation als eine allgemeine homogene 4×4-Matrix beschrieben werden kann:

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \qquad T \in \mathrm{SE}(3) \text{ mit } \mathbf{t} \in \mathbb{R}^3 \text{ und } R \in \mathrm{SO}(3)$$

- Eine **Translationsmatrix**, die eine Verschiebung um $\mathbf{t} = (t_x \quad t_y \quad t_z)^\top$ beschreibt, ist demnach:

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

- Es gilt $T^{-1} = \begin{pmatrix} R^\top & -R^\top \cdot \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}$. Diese bildet $\mathbf{b} = R \cdot \mathbf{x} + \mathbf{t}$ wieder auf $\mathbf{x}$ ab

- **Interpretationen von homogenen 4×4-Matrizen**:
  - **Lagebeschreibung**: ${}^{A}P_{B}$ beschreibt die Lage des Koordinatensystems B relativ zum Koordinatensystem A
  - **Transformationsabbildung** zwischen Koordinatensystemen:

    $$ {}^{A}T_{B} \colon {}^{B}P \to {}^{A}P, \qquad {}^{A}P = {}^{A}T_{B} \cdot {}^{B}P $$

    transformiert Koordinatensystem B in Koordinatensystem A
  - **Transformationsoperator** innerhalb eines Koordinatensystems:

    $$ T \colon {}^{A}P_{1} \to {}^{A}P_{2}, \qquad {}^{A}P_{2} = T \cdot {}^{A}P_{1} $$

    transformiert einen Punkt P_1 in einen Punkt P_2 innerhalb des Koordinatensystems A

  Beispiele 1/58-59
- Lagebeschreibungen können als Matrixprodukt verkettet werden, z.B.

$$ {}^{\text{BKS}}T_{O_3} = {}^{\text{BKS}}T_{O_1} \cdot {}^{O_1}T_{O_2} \cdot {}^{O_2}T_{O_3} $$

Quaternionen:

- Repräsentation ohne Nachteile von Rotationsmatrizen und Eulerwinkeln
- Menge der **Quaternionen** $\mathbb{H}$ ist definiert durch:

$$ \mathbb{C} + \mathbb{C}j \qquad \text{mit } j^2 = -1 \text{ und } i \cdot j = -j \cdot i $$

  wobei i die imaginäre Einheit ist
- Ein Element $\mathbf{q} \in \mathbb{H}$ hat die Form:

$$ \mathbf{q} = (a, \mathbf{u})^{\top} = a + u_1 i + u_2 j + u_3 k \qquad \text{mit } a \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^3 \text{ und } k = i \cdot j $$

- a heißt **Realteil** und $\mathbf{u} = (u_1, u_2, u_3)^{\top}$ heißt **Imaginärteil**
- **Rechenregeln**:

$\cdot$	$\mathbf{1}$	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\mathbf{1}$	1	i	j	k
$\boldsymbol{i}$	i	-1	k	$-j$
$\boldsymbol{j}$	j	$-k$	-1	i
$\boldsymbol{k}$	k	j	$-i$	-1

- **Rechenoperationen**: Seien $\mathbf{q} = (a, \mathbf{u})^\top, \mathbf{r} = (b, \mathbf{v})^\top$ zwei Quaternionen
  - **Addition**: $\mathbf{q} + \mathbf{r} = (a + b, \mathbf{u} + \mathbf{b})^\top$
  - **Skalarprodukt**: $\langle \mathbf{q} \mid \mathbf{r} \rangle = a \cdot b + \langle \mathbf{v} \mid \mathbf{u} \rangle = a \cdot b + v_1 \cdot u_1 + v_2 \cdot u_2 + v_3 \cdot u_3$
  - **Multiplikation**: $\mathbf{q} \cdot \mathbf{r} = (a + u_1 i + u_2 j + u_3 k) \cdot (b + v_1 i + v_2 j + v_3 k)$
  - **Konjugierte** Quaternion: $\mathbf{q}^* = (a, -\mathbf{u})^\top$
  - **Norm**: $|\mathbf{q}| = \sqrt{\mathbf{q} \cdot \mathbf{q}^*} = \sqrt{\mathbf{q}^* \cdot \mathbf{q}} = \sqrt{a^2 + u_1^2 + u_2^2 + u_3^2}$
  - **Inverse**: $\mathbf{q}^{-1} = \dfrac{\mathbf{q}^*}{|\mathbf{q}|^2}$
- **Einheitsquaternion** $\mathbb{S}^3 = \{\mathbf{q} \in \mathbb{H} \mid \|q\|^2 = 1\}$
- Beschreibung eines Vektors $\mathbf{p} \in \mathbb{R}^3$ als Quaternion $\mathbf{q}$: $\mathbf{q} = (0, \mathbf{p})^\top$
- Beschreibung eines Skalars $s \in \mathbb{R}^3$ als Quaternion $\mathbf{q}$: $\mathbf{q} = (s, \mathbf{0})^\top$
- Sei eine Rotation beschrieben durch eine Drehachse $\mathbf{a}$ mit $\mathbf{a} = 1$ und einen Drehwinkel θ, dann existiert hierfür eine Repräsentation als Quaternion: $\mathbf{q} = \left(\cos\frac{\theta}{2}, \mathbf{a}\sin\frac{\theta}{2}\right)$
- Ein Punkt $\mathbf{v}$ wird mit einer Quaternion $\mathbf{q}$ rotiert durch: $\mathbf{v'} = \mathbf{q}\mathbf{v}\mathbf{q}^{-\mathbf{1}} = \mathbf{q}\mathbf{v}\mathbf{q}^*$, wobei die letzte Gleichheit gilt, weil $\mathbf{q}$ ein Einheitsquaternion ist
- Verkettung von Rotationen $f \circ h$: $f(h(\mathbf{v})) = \mathbf{q}(\mathbf{r}\mathbf{v}\mathbf{r}^*)\mathbf{q}^*$

Beispiel 1/72

Bewertung von Quaternionen:

- **Vorteile**: Kompakt, Anschaulich, Kein Gimbal Lock, Verkettung möglich, Stetige Repräsentation
- **Nachteil**: Nur Beschreibung von Rotation, keine Translation

SLERP Interpolation:

- SLERP Interpolation von $\mathbf{q_1}$ nach $\mathbf{q_2}$ mit Parameter $t \in [0, 1]$:

$$\text{SLERP}(\mathbf{q_1}, \mathbf{q_2}, t) = \frac{\sin((1-t)\cdot\theta)}{\sin\theta} \cdot \mathbf{q_1} + \frac{\sin(t\cdot\theta)}{\sin\theta} \cdot \mathbf{q_2} \qquad \text{mit } \langle \mathbf{q_1} \mid \mathbf{q_2} \rangle = \cos\theta$$

- Ergebnis: Rotation mit konstanter Winkelgeschwindigkeit
- **Problem**: Orientierungen in SO(3) werden durch Einheitsquaternionen doppelt abgedeckt, weil $\mathbf{q}$ und $-\mathbf{q}$ der gleichen Rotation entsprechen

  $\Rightarrow$ SLERP berechnet deshalb nicht immer die kürzeste Rotation. Es muss geprüft werden, ob die Rotation von $\mathbf{q_1}$ zu $\mathbf{q_2}$ oder $-\mathbf{q_1}$ zu $\mathbf{q_2}$ kürzer ist

Duale Quaternionen:

- Erlauben es auch Translationen zu berücksichtigen
- **Duale Zahlen** sind Zahlen der Form:

$$d = p + \varepsilon \cdot s, \quad \text{wobei } \varepsilon^2 = 0$$

  mit **Primärteil** p, **Sekundärteil** s
- **Rechenoperationen**: Seien $d_1 = p_1 + \varepsilon \cdot s_1$ und $d_2 = p_2 + \varepsilon \cdot s_2$ duale Zahlen
  - **Addition**: $d_1 + d_2 = p_1 + p_2 + \varepsilon \cdot (s_1 + s_2)$
  - **Multiplikation**: $d_1 \cdot d_2 = p_1 \cdot p_2 + \varepsilon \cdot (p_1 \cdot s_2 + p_2 \cdot s_1)$
- **Duale Quaternionen**:

$$DQ = (d_1, d_2, d_3, d_4), \qquad d_i = dp_i + \varepsilon \cdot ds_i$$

  - Primärteil dp_i enthält den Winkelwert $\theta/2$
  - Sekundärteil ds_i enthält die Translationsgröße $d/2$
- **Multiplikationstabelle für duale Einheitsquaternionen**:

$\cdot$	$\mathbf{1}$	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon i}$	$\boldsymbol{\varepsilon j}$	$\boldsymbol{\varepsilon k}$
$\mathbf{1}$	1	i	j	k	ε	εi	εj	εk
$\boldsymbol{i}$	i	-1	k	$-j$	εi	$-\varepsilon$	εk	$-\varepsilon j$
$\boldsymbol{j}$	j	$-k$	-1	i	εj	$-\varepsilon k$	$-\varepsilon$	εi
$\boldsymbol{k}$	k	j	$-i$	-1	εk	εj	$-\varepsilon i$	$-\varepsilon$
$\boldsymbol{\varepsilon}$	ε	εi	εj	εk	0	0	0	0
$\boldsymbol{\varepsilon i}$	εi	$-\varepsilon$	εk	$-\varepsilon j$	0	0	0	0
$\boldsymbol{\varepsilon j}$	εj	$-\varepsilon k$	$-\varepsilon$	εi	0	0	0	0
$\boldsymbol{\varepsilon k}$	εk	εj	$-\varepsilon i$	$-\varepsilon$	0	0	0	0

- Rotation um eine Achse $\mathbf{a}$ mit dem Winkel θ: $\mathbf{q_r} = \left(\cos\frac{\theta}{2}, \mathbf{a}\sin\frac{\theta}{2}\right) + \varepsilon \cdot (0,0,0,0)$
- Translation mit dem Vektor $\mathbf{t} = (t_x, t_y, t_z)$: $\mathbf{q_t} = (1,0,0,0) + \varepsilon \cdot (0, \frac{t_x}{2}, \frac{t_y}{2}, \frac{t_z}{2})$
- Kombination zu einer Transformation T: $\mathbf{q_T} = \mathbf{q_t q_r}$
- Eine Transformation $\mathbf{q_T}$ wird auf einen Punkt $\mathbf{p}$ als duale Quaternion wie folgt angewendet: $\mathbf{p'} = \mathbf{q_T p q_T^*}$, mit $\mathbf{q_T^*} = (\mathbf{q_t q_r})^* = \mathbf{q_r^* q_t^*}$
- Konjugieren von $\mathbf{q} = \mathbf{p} + \varepsilon \cdot \mathbf{s}$: $\mathbf{q}^* = \mathbf{p}^* - \varepsilon \cdot \mathbf{s}^*$

Beispiel 1/83-85

Bewertung von dualen Quaternionen:

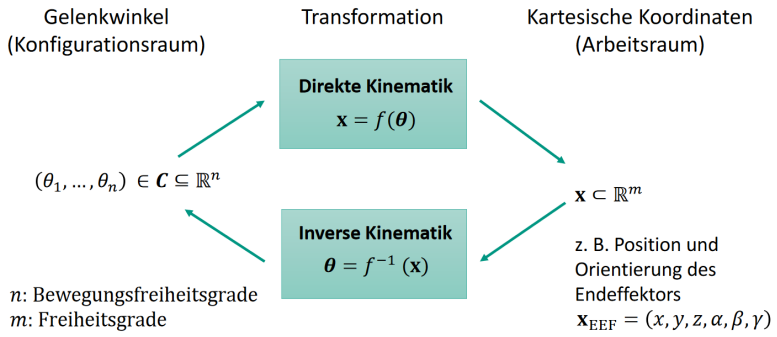
- **Vorteile**: Erlauben Lagebeschreibung und Transformationen, Geringere Redundanz (nur 8 statt 12 Werte bei homogener Matrix)
- **Nachteile**: Lage durch Angabe einer Dualquaternion zu beschreiben ist relativ schwierig, Komplexe Verarbeitungsvorschriften

2 Kinematik

Kinematisches Modell: Beschreibt Zusammenhänge zwischen **Gelenkwinkelraum** (Konfigurationsraum) und **Posenraum des Endeffektors** (Arbeitsraum)

Direkte und Inverse Kinematik:

- **Direkte Kinematik**:
  - Eingabe: Gelenkwinkelstellungen des Roboters
  - Ausgabe: Pose des Endeffektors
  - z.B. Wo befindet sich meine Hand?
- **Inverse Kinematik**:
  - Eingabe: Zielpose des Endeffektors
  - Ausgabe: Gelenkwinkelstellungen
  - z.B. Wie bewege ich meine Hand zum Ziel?



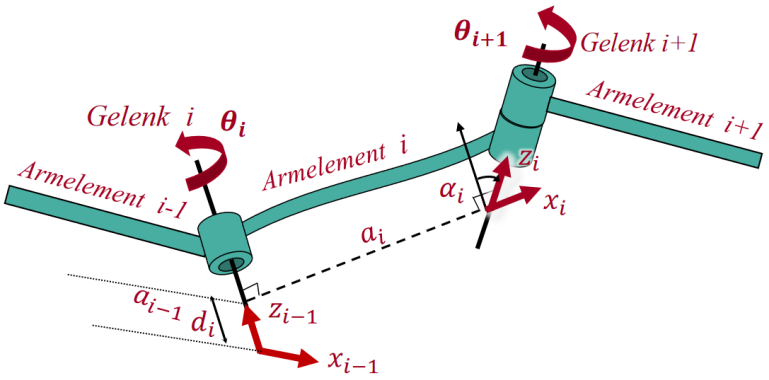
Kinematische Kette wird von mehreren Körpern gebildet, die durch Gelenke kinematisch verbunden sind (z.B. Roboterarm). Unterscheidung zwischen:

- **Offene** kinematische Kette: Nur ein Ende ist fest, anderes Ende frei bewegbar
- **Geschlossene** kinematische Kette: Beide Enden der Kette sind fest

Für jedes Glied müssen **6 Parameter** für die Transformation zwischen Gelenken bestimmt werden (3 Rotationsparameter, 3 Translationsparameter)

Denavit-Hartenberg (DH) Konvention:

- Durch die geschickte Wahl der Koordinatensysteme lassen sich die Parameter zur Beschreibung eines Armelements auf 4 reduzieren
- **Regeln für Koordinatensysteme**:
  - z_{i-1}-Achse liegt entlang der Bewegungsachse des i-ten Gelenks
  - x_i-Achse verläuft entlang der gemeinsamen Normalen (Kreuzprodukt von z_{i-1} und z_i) von z_{i-1} und z_i
  - y_i-Achse vervollständigt das Koordinatensystem entsprechend der Rechte-Hand-Regel
- **Parameter des Armelements (DH-Paramerter)**:
  - **Armelementlänge** a_i beschreibt den Abstand von z_{i-1} zu z_i entlang x_i
  - **Armelementverdrehung** α_i beschreibt den Winkel von z_{i-1} zu z_i um x_i
  - **Gelenkabstand** d_i ist der Abstand zwischen der x_{i-1}-Achse und x_i-Achse entlang der z_{i-1}-Achse
  - **Gelenkwinkel** θ_i ist der Winkel von x_{i-1} zu x_i um z_{i-1}



- DH-Parameter beschreiben wie aufeinanderfolgende Gelenke ineinander transformiert werden
- **DH-Transformationsmatrizen**: Beschreibung mit homogenen Matrizen

1. Rotation θ_i: $R_{z_{i-1}}(\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

2. Translation d_i: $T_{z_{i-1}}(d_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$

3. Translation a_i: $T_{x_i}(a_i) = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

4. Rotation α_i: $R_{x_i}(\alpha_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- Zusammenführen zu einer Matrix:

$$\begin{aligned} A_{i-1,i} &= R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) \\ &= \begin{pmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos\alpha_i & \sin\theta_i \cdot \sin\alpha_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & -\cos\theta_i \cdot \sin\alpha_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

- Inverse DH-Transformation:

$$A_{i-1,i}^{-1} = A_{i,i-1} = \begin{pmatrix} \cos\theta_i & \sin\theta_i & 0 & -a_i \\ -\cos\alpha_i \cdot \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & \sin\alpha_i & -d_i \cdot \sin\alpha_i \\ \sin\theta_i \cdot \sin\alpha_i & -\sin\alpha_i \cdot \cos\theta_i & \cos\alpha_i & -d_i \cdot \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Durch Multiplikation der DH-Matrizen lässt sich die Lage der einzelnen Koordinatensysteme bezüglich des Bezugskoordinatensystems bestimmen

Direktes kinematisches Problem: Stellung des Endeffektors (EFF) in Bezug auf das BKS ist gegeben durch:

$$S_{\text{Basis, EEF}} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \cdots \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$$

Lösung des Problems ergibt sich aus Einsetzen der Gelenkwinkel in obige Gleichung.

Beispiele 2/38-48

Oft interessiert man sich für verwandte Beziehungen wir z.B. Gelenkwinkelgeschwindigkeiten → Endeffektor-Geschwindigkeit. Dafür muss man die Vorwärtskinematik ableiten → **Jacobi-Matrix**

Jacobi-Matrix: Für eine differenzierbare Funktion $f\colon \mathbb{R}^n \to \mathbb{R}^m$, $f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix}$ und $\mathbf{x} \in \mathbb{R}^n$ ist die Jacobi-Matrix für ein $\mathbf{a} \in \mathbb{R}^n$ wie folgt:

$$J_f(\mathbf{a}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{a}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial f_m}{\partial x_n}(\mathbf{a}) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Problem: Vorwärtskinematik ist matrixwertig → Jacobi-Matrix nicht definiert

Lösung: Vektorwertige Repräsentation wählen, z.B. mit Eulerwinkel

Geschwindigkeitsraum und Kraftraum:

- Annahme: Kinematische Kette bewege sich entlang einer Trajektorie $\theta\colon \mathbb{R} \to \mathbb{R}^n$, wobei θ die Gelenkwinkelstellungen zu einem Zeitpunkt t beschreibt
- Pose des End-Effektors $\mathbf{x}(t) \in \mathbb{R}^6$ zum Zeitpunkt t: $\mathbf{x}(t) = \mathbf{f}(\boldsymbol{\theta}(t))$, wobei f die Funktion für die Vorwärtskinematik ist
- **Beziehung zwischen Endeffektor- und Gelenkwinkelgeschwindigkeiten**:

  $$\dot{\mathbf{x}}(t) = J_{\mathbf{f}}(\boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t)$$

- **Beziehung zwischen Kräfte und Momente am End-Effektor und Drehmomenten in den Gelenken**:

  $$\boldsymbol{\tau}(t) = J_{\mathbf{f}}^{\top}(\boldsymbol{\theta}(t)) \cdot \mathbf{F}(t)$$

  wobei $\mathbf{F}\colon \mathbb{R} \to \mathbb{R}^6$ der Kraft-Momenten-Vektor am End-Effektor und $\boldsymbol{\tau}\colon \mathbb{R} \to \mathbb{R}^n$ die Drehmomente in Gelenken ist
- Im Geschwindigkeits- und Kraftraum lässt sich die Frage nach der Inversen Kinematik durch die Inverse der Jacobi-Matrix lösen, z.B. Welche Gelenkwinkelgeschwindigkeiten sind notwendig, um eine End-Effektor-Geschwindigkeit zu realisieren? → Löse nach $\dot{\boldsymbol{\theta}}(t)$ auf

Berechnung der Jacobi-Matrix:

- Jede Spalte der Jacobi-Matrix gehört zum Gelenk θ_i der kinematischen Kette
- Wenn j-tes Gelenk ein Translationsgelenk ist, das eine Translation in Richtung des Einheitsvektors $\mathbf{z_j} \in \mathbb{R}^3$ durchführt, gilt für die j-te Spalte der Jacobi-Matrix:

  $$J_j(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{z_j} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^6$$

- Wenn j-tes Gelenk ein Rotationsgelenk ist, das eine Rotation um die Rotationsachse $\mathbf{z_j} \in \mathbb{R}^3$ an der Position $\mathbf{p_j} \in \mathbb{R}^3$ durchführt, gilt für j-te Spalte der Jacobi-Matrix:

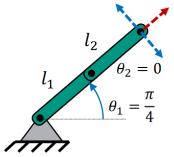
$$J_j(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{z_j} \times (\mathbf{f}(\boldsymbol{\theta}) - \mathbf{p_j}) \\ \mathbf{z_j} \end{bmatrix} \in \mathbb{R}^6$$

Beispiel 2/68-72

Kinematische Singularitäten:

- **Definition**: Wenn sich Roboter in einer Konfiguration $\boldsymbol{\theta}_{\text{singular}} \in C$ befindet, in der er sich nicht instantan in eine oder mehr Richtungen bewegen kann
- Bedeutet, dass Jacobi-Matrix $J(\boldsymbol{\theta})$ keinen vollen Rang bzw. min. zwei linear abhängige Spalten hat, also nicht invertierbar ist

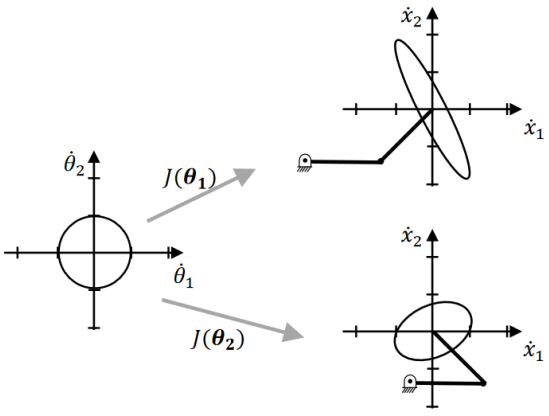
Es existiert keine Gelenkwinkelgeschwindigkeit, die eine Endeffektor-Geschwindigkeit in die **rote Richtung** erzeugt.
⇒ Die Konfiguration ist singulär.



Manipulierbarkeit: Maß für die Bewegungsfreiheit des Endeffektors, bzw. wie nahe eine Konfiguration an einer Singularität ist

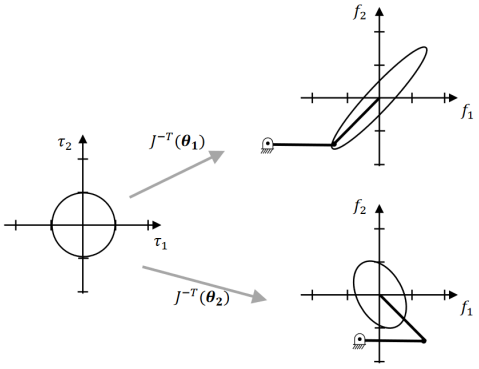
- Nutze $J(\boldsymbol{\theta})$ um Einheitskreis der Gelenkwinkel-Geschwindigkeiten in den Raum der Endeffektor-Geschwindigkeiten abzubilden

  → Resultat: **Ellipsoid der Manipulierbarkeit**
- Kreis: Bewegungen des Endeffektors in alle Richtungen uneingeschränkt möglich
- Degenerierte Fälle (gestauchtes Ellipsoid): Endeffektor-Bewegung ist in bestimmen Richtungen eingeschränkt



- Singulärwerte von $J(\boldsymbol{\theta})$ sind die Wurzeln der Eigenwerte von $J(\boldsymbol{\theta}) \cdot J(\boldsymbol{\theta})^\top$, also $\sigma_i = \sqrt{\lambda_i}$, wobei λ_i ein EW von $J(\boldsymbol{\theta}) \cdot J(\boldsymbol{\theta})^\top$ ist
- **Skalare Maße** für die Manipulierbarkeit:
  - Kleinster Singulärwert $\mu_1(\boldsymbol{\theta}) = \sigma_{\min}(A(\boldsymbol{\theta}))$
  - Inverse Kondition: $\mu_2(\boldsymbol{\theta}) = \dfrac{\sigma_{\min}(A(\boldsymbol{\theta}))}{\sigma_{\max}(A(\boldsymbol{\theta}))}$
  - Determinante: $\mu_3(\boldsymbol{\theta}) = \det A(\boldsymbol{\theta})$
- **Kraft-Ellipsoid**: Selbiges kann man auch für den Kraftraum einführen

$$\boldsymbol{\tau}(t) = J_{\boldsymbol{f}}^{\top}\big(\boldsymbol{\theta}(t)\big) \cdot \boldsymbol{F}(t) \qquad \rightarrow \qquad \boldsymbol{F}(t) = J_{\boldsymbol{f}}^{-\top}\big(\boldsymbol{\theta}(t)\big) \cdot \boldsymbol{\tau}(t)$$



Geometrisches Modell:

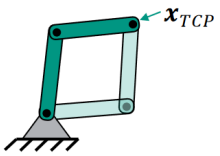
- **Einsatzbereiche**:
  - Abstandsmessung und Kollisionserkennung
  - Graphische Darstellung von Körpern
  - Berechnung der Bewegungen von Körpern
  - Ermittlung der wirkenden Kräfte und Momente
- **Klassifizierung** nach **Raum** (2D oder 3D) oder **Grundprimitiven** (Kanten- bzw. Drahtmodelle, Flächen- bzw. Oberflächenmodelle, Volumenmodell)

3 Inverse Kinematik

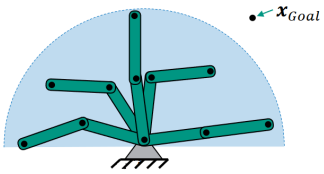
Problemstellung: siehe S. 9

Damit Inverse $\theta = f^{-1}(x)$ existiert, muss die Vorwärtskinematik f bijektiv sein, aber:

- Vorwärtskinematik ist i. A. nicht injektiv

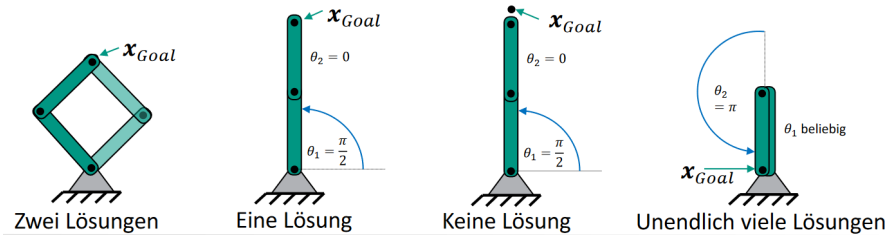


- Vorwärtskinematik ist i. A. nicht surjektiv



Unterschiedliche Fälle der inversen Kinematik:

- Es gibt zwei unabhängige Lösungen (Normalfall).
- Es gibt genau eine Lösung (Rand des Arbeitsraums).
- Es gibt keine Lösung (Außerhalb des Arbeitsraums).
- Es gibt unendlich viele Lösungen (Zielpunkt in der Basis)



Geometrische Methoden:

- Nutze geometrische Beziehungen (z.B. Sinus-/Kosinussatz), um θ aus T_{TCP} zu bestimmen
- Bei mehreren Gelenken kann das aber sehr schwierig werden
- **Beispiel**: *3/23-25*

Algebraische Methoden:

- Führe Koeffizientenvergleich der beiden Matrizen der gewünschten TCP-Pose und der Transformation aus dem kinematischen Modell durch

  $\rightarrow$ 16 Gleichungen bei homogenen Matrizen, von denen 4 trivial sind (letzte Zeile immer (0 0 0 1))

- **Beispiel**: *3/29-32*
- **Problem**: Oft können nicht alle Gelenkwinkel aus den 12 Gleichungen bestimmt werden
- **Lösung**: Kenntnis der Transformationen erhöht Chance, die Gleichungen zu lösen, z.B. sei Gleichung $P_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot A_{2,3}(\theta_3) \cdot A_{3,4}(\theta_4) \cdot A_{4,5}(\theta_5) \cdot A_{5,6}(\theta_6)$ gegeben
  1. Invertiere $A_{0,1}(\theta_1)$ und multipliziere beide Seiten der Gleichung mit $A_{0,1}^{-1}$
  2. Versuche im neuen Gleichungssystem den Wert einer Variablen zu bestimmen
  3. Versuche weitere Gleichung zu finden, die durch Substitution der im letzten Schritt gefundenen Lösung lösbar ist
  4. Falls keine Lösungen gefunden werden kann, so muss eine weitere Matrix $A_{1,2}(\theta_2)$ invertiert werden
  5. Wiederhole die Schritte 1 - 4 bis alle Gelenkwinkel ermittelt sind

  → Große Wahrscheinlichkeit, dass das Gleichungssystem gelöst werden kann

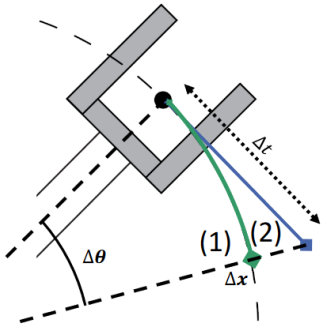
Im Folgenden werden **numerische Methoden** betrachtet:

Gradientenabstieg:

- **Fehlerfunktion** für die Zielpose $\mathbf{x}_{\text{Goal}} \in W$: $e(\boldsymbol{\theta}) = \|\mathbf{x}_{\text{Goal}} - f(\boldsymbol{\theta})\|^2$
- Wähle Startkonfiguration $\boldsymbol{\theta}_0 \in C, i = 0$ und Schrittlänge $\gamma \in \mathbb{R}$
- Solange $e(\boldsymbol{\theta}_i) > e_{\text{Threshold}}$:
  - $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma \cdot 2 \cdot J^T(\boldsymbol{\theta}_i) \cdot (f(\boldsymbol{\theta}_i) - \mathbf{x}_{\text{Goal}})$, wobei J die Jacobi-Matrix der Vorwärtskinematik ist
  - $i = i + 1$
- Je nach gewählter Startkonfiguration erhält man unterschiedliche Lösungen

Pseudoinverse:

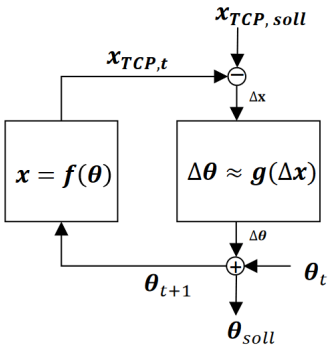
- **Differenzenquotient**:
  - Tatsächliche Bewegung gemäß $\dot{\mathbf{x}}(t) = J(\boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t)$
  - Annäherung mittels Differenzenquotient: $\Delta\mathbf{x} \approx J(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$



- Suche Lösung für das inverse Problem: $\Delta\boldsymbol{\theta} = J_f^{-1}(\boldsymbol{\theta}) \cdot \Delta\mathbf{x}$, aber J_f muss nicht unbedingt invertierbar sein $\rightarrow$ Nutze Pseudoinverse
- Es lässt sich herleiten:

$$\Delta\boldsymbol{\theta} = (J_f^\top J_f)^{-1} J_f^\top \Delta\mathbf{x} = J_f^+(\boldsymbol{\theta})\Delta\mathbf{x}$$

  wobei J_f^+ die **Pseudoinverse** von J_f ist $(A^+ = (A^\top A)^{-1}A^\top)$
- **Algorithmus**:



- **Problem**: Pseudoinverse ist in der Nähe von Singularitäten instabil!
- **Lösung**: **Damped least squares**-Methode

Damped Least Squares:

- Führe Dämpfungskonstante $\lambda > 0$ ein und berechne

$$\Delta\boldsymbol{\theta} = J^\top(JJ^\top + \lambda^2 I)^{-1}\Delta\mathbf{x}$$

- Analyse der Stabilität über **Singulärwertzerlegung**: $J = UDV^T = \sum_{i=1}^{r} \sigma_i \mathbf{u_i}\mathbf{v_i}^\top$, wobei U, V orthogonale Matrizen, D Diagonalmatrix mit Singulärwerten auf der Diagonalen, $\mathbf{u_i}, \mathbf{v_i}$ die i-te Spalte von U bzw. V und r der rang von J ist

- Es lässt sich herleiten:
  - Pseudoinverse: $J^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v_i} \mathbf{u_i}^\top$
  - Damped Least Squares: $J^\top (JJ^\top + \lambda^2 I)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v_i} \mathbf{u_i}^\top$

    $\rightarrow$ Für $\sigma_i \rightarrow 0$ wird die Pseudoinverse instabil, Damped-Least-Squares bleibt aber wohldefiniert. Für große σ_i verglichen mit λ verhält sich Damped-Least-Squares wie die Pseudoinverse.

4 Dynamik

Dynamisches Modell: Beschreibt Zusammenhang zwischen Antriebs- und Kontaktkräfte, welche in einem mechanischen Mehrkörpersystem auftreten und deren resultierenden Beschleunigungen und Bewegungen

Allgemeine Bewegungsgleichung:

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) + \varepsilon(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$$

mit:

- $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$: $n \times 1$ Vektor der generalisierten Koordinaten (Position, Geschwindigkeit und Beschleunigung)
- $\boldsymbol{\tau}$: $n \times 1$ Vektor der generalisierten Kräfte
- $M(\mathbf{q})$: $n \times n$ Massenträgheitsmatrix
- $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$: $n \times 1$ Vektor mit Zentripetal- und Corioliskomponenten
- $g(\mathbf{q})$: $n \times 1$ Vektor der Gravitationskomponenten
- $\varepsilon(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$: $n \times 1$ Nichtlineare Effekte, wie z.B. Reibung

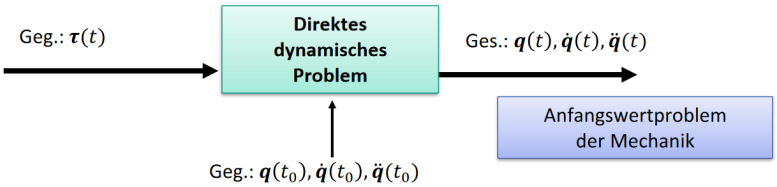
Generalisierte Koordinaten:

- **Definition**: Minimaler Satz an voneinander unabhängigen Koordinaten, der den aktuellen Systemzustand vollständig beschreibt
- Allgemein: Roboter besteht aus N Partikel. Für jeden Ortsvektor eines Partikels braucht man 3 Raumkoordinaten, insgesamt $3N$ Koordinaten, um das System zu beschreiben
- Partikel können sich wegen Verbindungen und Gelenken nicht unabhängig voneinander bewegen $\rightarrow$ Einführung von **Zwangsbedingungen**

- $3N$ Koordinaten lassen sich mit k unabhängigen Zwangsbedingungen auf $3N - k$ unabhängige generalisierte Koordinaten q_i reduzieren
- **Beispiel**: *4/11-15*

Direktes Dynamisches Problem:

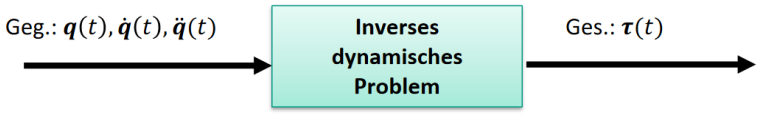
- Gegeben externe Kraft und aktueller Bewegungszustand, was ist die neue Bewegung des Systems?



- Löse Differentialgleichung $\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$ nach $\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)$

Inverses Dynamisches Problem:

- Aus den gewünschten Bewegungsparametern sollen die dazu erforderlichen Kräfte und Momente ermittelt werden



- Berechne rechten Teil der Gleichung $\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$ durch Einsetzen

Modellierung der Dynamik: Versuche Terme der allgemeinen Bewegungsgleichung herzuleiten

Methode nach Lagrange:

1. Ermittle E_{kin} und E_{pot}
2. Drücke E_{kin} und E_{pot} in generalisierten Koordinaten als **Lagrange-Funktion** aus

$$L(\mathbf{q}, \dot{\mathbf{q}}) = E_{\text{kin}}(\mathbf{q}, \dot{\mathbf{q}}) - E_{\text{pot}}(\mathbf{q})$$

3. Für jedes Gelenk i ist die Bewegungsgleichung:

$$\tau_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i}$$

Beispiele: *4/26-35*

Eigenschaften:

- Einfaches Aufstellen der Gleichungen
- Geschlossenes Modell, analytisch auswertbar
- Berechnung sehr umfangreich $\mathcal{O}(n^3)$
- Nur Antriebsmomente werden berechnet

Methode nach Newton-Euler:

- **Newton-Euler Gleichungen**:

$$\mathbf{f} = \boldsymbol{I}\mathbf{a} + \mathbf{v} \times \boldsymbol{I}\mathbf{v} \quad \text{wobei} \quad \mathbf{f} = \begin{pmatrix} \boldsymbol{n}_C \\ \boldsymbol{f} \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_C \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}}_C \end{pmatrix}$$

$\boldsymbol{v}_C$: Lineare Geschwindigkeit des Körpers in Bezug auf CoM
$\dot{\boldsymbol{v}}_C$: Lineare Beschleunigung des Körpers in Bezug auf CoM
f, v, a: 6D Kraft- bzw. Bewegungsvektoren welche alle auf den Körper wirkenden Kräfte und Bewegungen (Geschwindigkeit, Beschleunigung) beschreiben

- Die Beschleunigungen $\ddot{\mathbf{c}}_\mathbf{i}$ und $\dot{\boldsymbol{\omega}}_\boldsymbol{i}$ eines Armelementes i hängen von den Beschleunigungen der vorhergehenden Armelemente ab

  → Beschleunigungen können über kinematisches Modell von der Basis zum Greifer rekursiv berechnet werden → **Vorwärtsgleichungen**

- Die Kraft $\mathbf{f}_i$ und das Drehmoment $\mathbf{n}_{\mathbf{C},\mathbf{i}}$, die auf ein Armelement i wirken, hängen von den nachfolgenden Armelementen ab

  → Kräfte und Momente können vom Greifer zur Basis rekursiv berechnet werden → **Rückwärtsgleichungen**

  → **Rekursiver Newton-Euler Algorithmus** (RNEA)

Rekursiver Newton-Euler Algorithmus:

1. Rekursive Berechnung der Geschwindigkeit und Beschleunigung jedes einzelnen Armelements i von der Basis bis zum Endeffektor:
   - Geschwindigkeit: $\mathbf{v}_\mathbf{i} = \mathbf{v}_{\mathbf{p(i)}} + \boldsymbol{\Phi}_\boldsymbol{i}\dot{\mathbf{q}}_\mathbf{i}$ mit $\mathbf{v}_\mathbf{0} = 0$
     - $\dot{\mathbf{q}}_\mathbf{i}$: Generalisierte Geschwindigkeit des Armelements i
     - $\boldsymbol{\Phi}_\boldsymbol{i}$: $6 \times n$ Bewegungsmatrix (Abhängig vom Gelenktyp)
     - $\mathbf{v}_{\mathbf{p(i)}}$: Geschwindigkeit des Vorgängerelements $p(i)$
   - Beschleunigung: $\mathbf{a}_\mathbf{i} = \mathbf{a}_{\mathbf{p(i)}} + \boldsymbol{\Phi}_\boldsymbol{i}\ddot{\mathbf{q}}_\mathbf{i} + \dot{\boldsymbol{\Phi}}_\boldsymbol{i}\dot{\mathbf{q}}_\mathbf{i}$ mit $\mathbf{a}_\mathbf{0} = -\mathbf{a}_\mathbf{g}$

2. Berechnung der Kräfte/Momente jedes einzelnen Armelements i mithilfe Newton-Euler:

$$\mathbf{f_i^a} = \mathbf{I_i a_i} + \mathbf{v_i} \times \mathbf{I_i v_i}$$

   - $\mathbf{f}_i^a$: Kräfte, welche aufgrund von a_i auf das Armelement i wirken
   - $\mathbf{I_i}$: Trägheitsmoment des Armelements i
   - $\mathbf{v_i}$: Geschwindigkeit des Armelements i (in Schritt 1 berechnet)
   - $\mathbf{a_i}$: Beschleunigung des Armelements i (in Schritt 1 berechnet)

3. Rekursive Berechnung der Kräfte zwischen den Armelementen und der generalisierten Kräfte für den jeweiligen Gelenktyp:

   - $\mathbf{f_i} = \mathbf{f_i^a} - \mathbf{f_i^e} + \sum_{j \in c(i)} \mathbf{f_j}$
   - $\boldsymbol{\tau_i} = \boldsymbol{\Phi_i}^\top \mathbf{f_i}$
     - $\mathbf{f_i}$: Resultierende Kraft am Armelement i
     - $\mathbf{f_i^e}$: Summe aller externen Kräfte, die an i wirken
     - $\mathbf{f_i}$: Kraft eines anliegenden Armelementes j
     - $c(i)$: Menge nachfolgender Armelemente i in kinematischer Kette
     - $\boldsymbol{\Phi_i}$: $6 \times n$ Bewegungsmatrix (Abhängig vom Gelenktyp)
     - $\boldsymbol{\tau_i}$: Generalisierte Kräfte/Momente an i

Eigenschaften der Methode nach Newton-Euler:

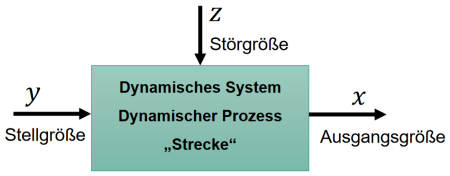
- Beliebige Anzahl von Gelenken
- Belastungen der Armelemente werden berechnet
- Aufwand $\mathcal{O}(n)$
- Rekursion

Herausforderungen der Dynamik:

- Nichtlineare Kräfte (wie z.B. Reibung) können nicht direkt modelliert werden, haben jedoch einen großen Einfluss
- Dynamik eines Roboters kann sich im Laufe der Zeit stark verändern z.B. durch Abnutzung
- Dynamik variiert stark in Abhängigkeit von der auszuführenden Aufgabe

5 Regelung

Regelungstechnik: Lehre von der selbsttätigen, gezielten Beeinflussung dynamischer Prozesse während des Prozessablaufs bei unvollständiger Systemkenntnis, insbesondere bei Störungen

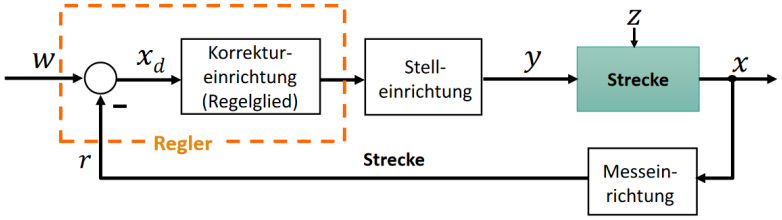


Aufgabe: Ausgangsgröße eines dynamischen Systems soll mittels der Stellgröße ein Sollverhalten gegen den Einfluss einer Störgröße aufzeigen

Prinzip: Strecke ist laufend zu beobachten und Stellgröße ist derart zu verändern, dass trotz der Störgrößeneinwirkung die Ausgangsgröße an den gewünschten Verlauf angeglichen wird.

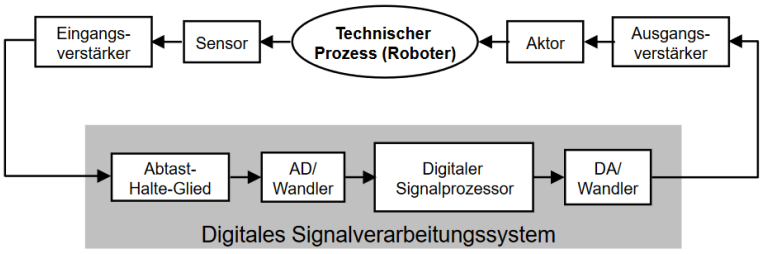
Eine Anordnung, die dies bewirkt heißt **Regelung**.

Aufbau einer Regelung:



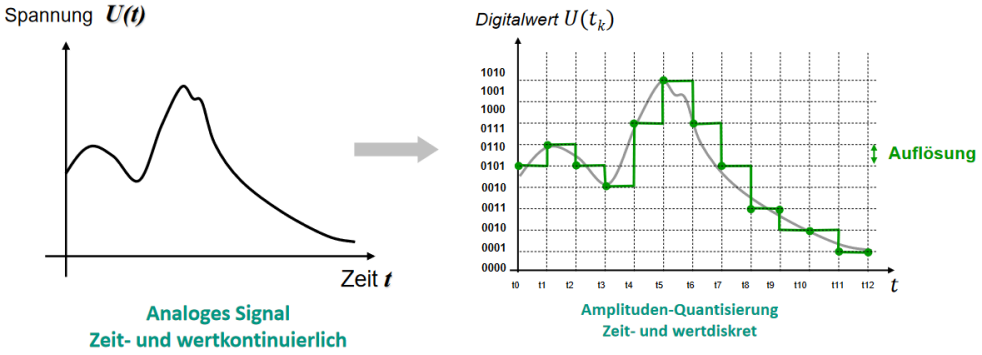
w	Führungsgröße	x_d	Regeldifferenz
y	Stellgröße	x	Regelgröße
r	Rückführgröße	z	Störgröße

Aufbau eines digitalen Signalverarbeitungssystems:

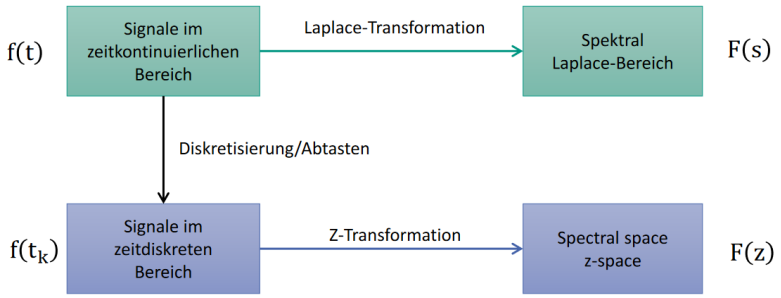


- Abtast-Halte-Glied hält den abgetasteten Wert innerhalb einer Abtastperiode konstant

Kontinuierliche und diskrete Signale:



Beschreibung von Dynamischen Systemen:



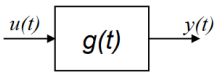
- Für Eingabesignal $u(t)$ und Ausgabesignal $y(t)$ werden die Laplace-Transformierten $U(s)$ und $Y(s)$ berechnet
- **Übertragungsfunktion** $G(s) = \frac{Y(s)}{U(s)}$

Laplace-Transformation:

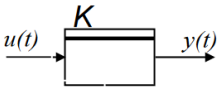
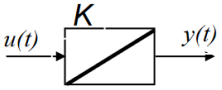
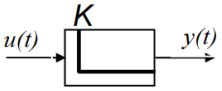
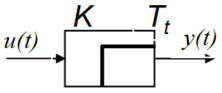
- $L\{f(t)\} = F(s) = \int_0^\infty f(t)e^{-st}\,dt \qquad s \in \mathbb{C}; \quad f(t) = 0, t < 0$
- $L\{1\} = \frac{1}{s}, \quad L\{a\} = \frac{a}{s}, \quad L\{t\} = \frac{1}{s^2}$
- $L\{\dot{f}(t)\} = s \cdot F(s) - f(0)$
- $L\{\int_0^t f(t)\,dt\} = \frac{1}{s}F(s)$
- $L\{e^{-\alpha t}\} = \frac{1}{s+\alpha}$
- Für Dirac-Impuls $\delta(t)$ ist $L\{\delta(t)\} = 1$
- $L\{\alpha f_1(t) + \beta f_2(t)\} = \alpha F_1(s) + \beta F_2(s)$

Weiter Regeln siehe *5/46-47*

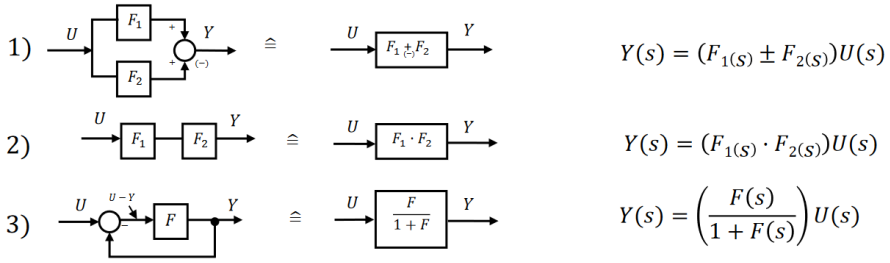
Übertragungsglieder:



- $Y(s) = G(s) \cdot U(s)$, $\quad G(s)$: Übertragungsfunktion
- $y(t) = g(t) * u(t) = \int_0^t g(\tau) \cdot u(t-\tau)\, d\tau$, für $t > 0$
- **Elementare Übertragungsglieder**:

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

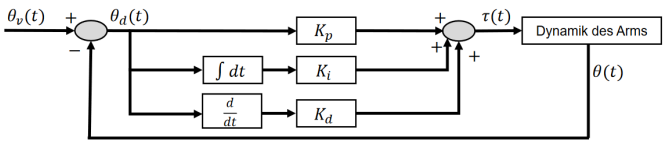
- **Umformregeln für Wirkungspläne**:



Beispiel Erstellung des Strukturbildes des Regelkreis: 5/58-67

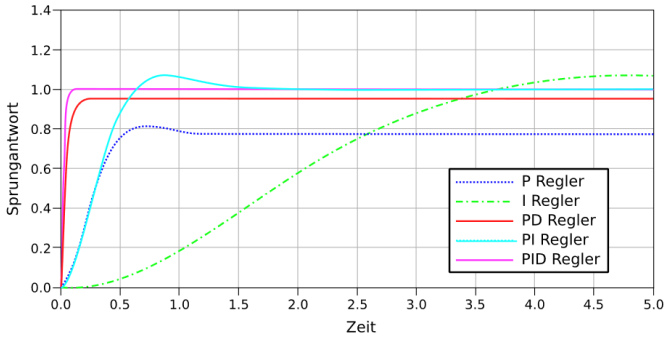
PID-Regler:

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



Laplace-Transformation liefert die Übertragungsfunktion: $G(s) = K_p + K_i \frac{1}{s} + K_d s$

Vergleich von Reglern:

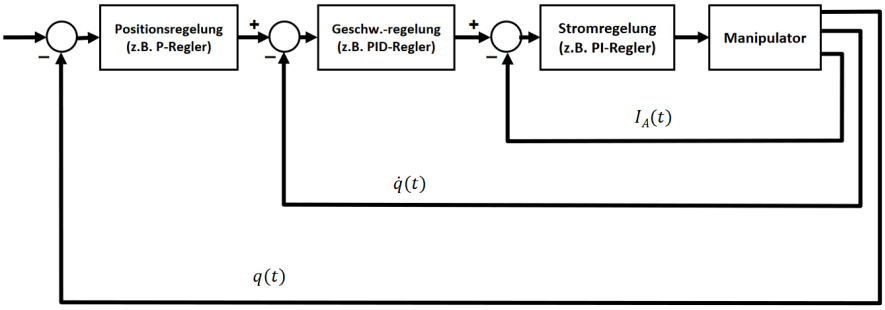


I-Anteil:
Regelabweichungen kompensieren (stationäre Genauigkeit)

D-Anteil:
Dynamik (wie schnell)

Stabilität von Reglern: s. *Beispiel 5/79-88*

Kaskadenregelung:



Regelungskonzepte für Manipulatoren: RELEVANT??? *s. 5/ab 92*

6 Bahnsteuerung

Trajektorie: Bewegungen eines Roboters werden aufgefasst als Zustandsänderungen über die Zeit, die Einschränkungen berücksichtigen.

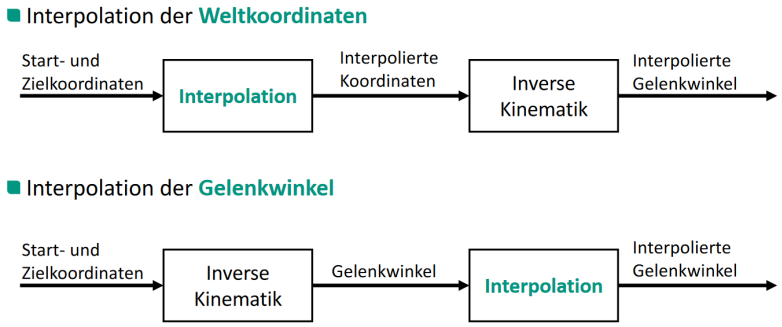
Problem der Bahnsteuerung:

- Gegeben: Zustand S_{Start} zum Startzeitpunkt und Zustand S_{Ziel} zum Zielzeitpunkt
- Gesucht: Zwischenzustände S_i, damit Trajektorie stetig wird

Darstellung der Zustände: Zustände können dargestellt werden im

- Konfigurationsraum: $\mathbb{R}^n$
- Arbeitsraum: $\mathbb{R}^3, \text{SE}(3)$
- Bahnsteuerung im Konfigurationsraum ist näher an der Ansteuerung der Teilsysteme des Roboter

- Bahnsteuerung im Arbeitsraum ist näher an der zu lösenden Aufgabe, zur Steuerung ist das Lösen der inversen Kinematik nötig



Bahnsteuerung im Konfigurationsraum:

- Bahnsteuerung als Funktion der Gelenkwinkelzustände
- Verlauf der punktweise in Gelenkwinkeln spezifizierten Bahn muss im Arbeitsraum nicht definiert sein
- Abfahren dieser punktweise spezifizierten Trajektorien **synchron** oder **asynchron**

Bahnsteuerung im Arbeitsraum:

- Angabe der Trajektorie erfolgt als Funktion der Zustände des Roboters, z.B. Position, Geschwindigkeit, Beschleunigung
- **Continuous Path**: Endeffektor folgt in Position und Orientierung einer definierten Bahn

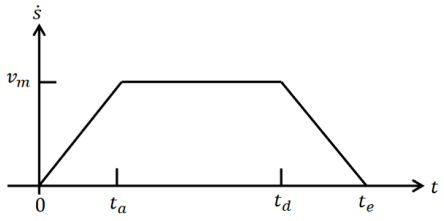
Vor- und Nachteile der Darstellungen:

- **Arbeitsraum**:
  - **Vorteile**: Bahn einfacher zu formulieren, Interpolation ist einfacher
  - **Nachteile**: Inverse Kinematik für jeden Punkt der Trajektorie zu lösen, Geplante Trajektorie nicht immer ausführbar
- **Konfigurationsraum**:
  - **Vorteile**: Ansteuerung der Gelenke einfacher und Trajektorie ist eindeutig und berücksichtigt Gelenkwinkelgrenzen
  - **Nachteile**: Interpolation für mehrere Gelenke, Formulierung der Trajektorie umständlicher

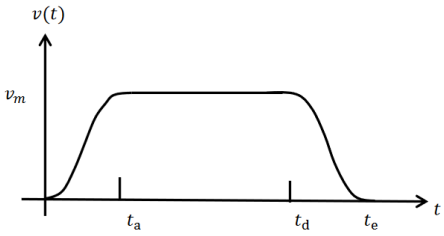
Punkt-zu-Punkt-Steuerung (der Gelenkwinkel):

- Roboter führt Punkt-zu-Punkt-Bewegung aus

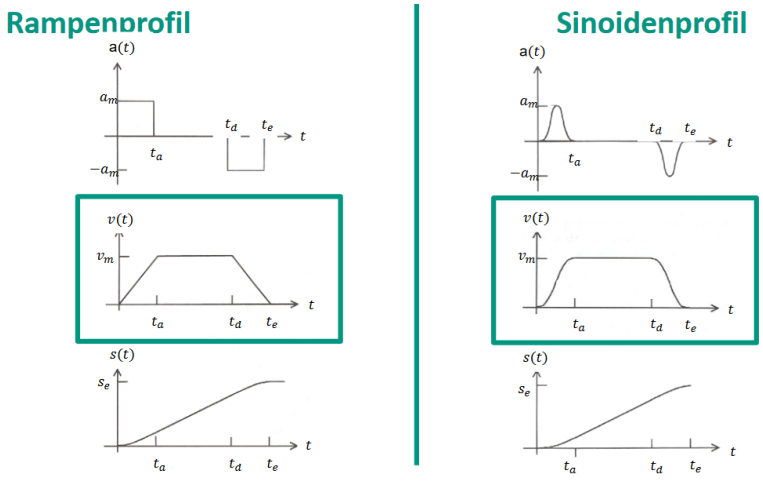
- **Vorteile**: Berechnung der Gelenkwinkeltrajektorie ist einfach, Keine Probleme mit Singularitäten
- Sequenz von Gelenkwinkelvektoren: $\mathbf{q}(t_j) = (q_1(t_j), q_2(t_j), \ldots, q_n(t_j))^\top$ mit $q_i(t_j)$ ist Winkel von Gelenk i zum Zeitpunkt t_j
- **Randbedingungen**:
  - Start- und Zielzustand sind bekannt: $\mathbf{q}(t_0) = \mathbf{q}_{\text{Start}},\ \mathbf{q}(t_e) = \mathbf{q}_{\text{Ziel}}$
  - Geschwindigkeiten zu Beginn und am Ende: $\dot{\mathbf{q}}(t_0) = x,\ \dot{\mathbf{q}}(t_e) = y$
  - Gelenkwinkelbereich sowie Geschwindigkeiten und Beschleunigungen sind begrenzt: $\mathbf{q}_{\min} < \mathbf{q}(t_j) < \mathbf{q}_{\max},\ |\dot{\mathbf{q}}(t_j)| < \dot{\mathbf{q}}_{\max},\ |\ddot{\mathbf{q}}(t_j)| < \ddot{\mathbf{q}}_{\max}$
- Interpolation für PTP mit **Rampenprofil**:
  - **Vorteil**: Einfache Art zur Berechnung der Bahnparameter $s(t)$
  - **Nachteil**: Sprungförmige Aufschaltung der Beschleunigung kann zu Eigenschwingungen von mechanischen Teilen führen



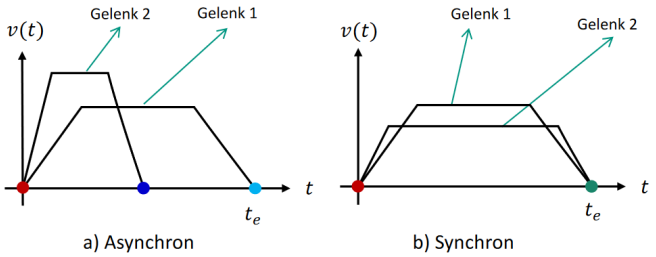
  - **Berechnung der Parameter**: *6/25-28*
- Interpolation für PTP mit **Sinoidenprofil**:
  - **Vorteil**: Roboter wird weniger beansprucht
  - **Nachteil**: Längere Beschleunigungs- und Bremsphase als beim Rampenprofil



  - **Berechnung der Parameter**: *6/31-32*



- **Asynchrone** und **synchrone** PTP-Bahnen:



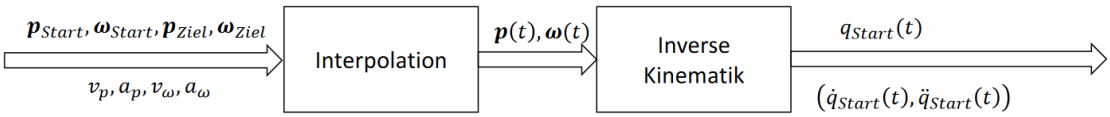
  - Asynchron: Jedes Gelenk wird sofort mit der maximalen Beschleunigung angesteuert. Jede Gelenkbewegung endet unabhängig von den anderen
  - Synchron: Alle Gelenke beginnen und beenden ihre Bewegungen zum gleichen Zeitpunkt

Synchrone PTP-Bahnen:

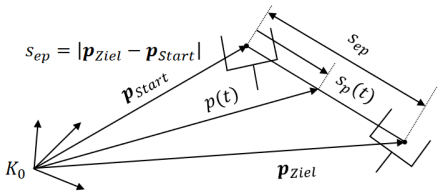
- Setze die maximale Fahrzeit als Fahrzeit für alle Gelenke: $t_{e,i} = t_e = \max(t_{e,i})$
- Bei **vollsynchron**: Zusätzlich Beschleunigungszeit t_a und Bremszeit t_d bei allen Gelenken gleich
- Herleitung Formeln: *5/38*

Steuerung im Arbeitsraum:

- **Continuous Path**: Endeffektor folgt in Position und Orientierung einer definierten Bahn
- Pose des Endeffektors im Arbeitsraum gegeben durch Punkt $\mathbf{p} = (x, y, z)^\top \in \mathbb{R}^3$ und Orientierung $\boldsymbol{\omega} = (\alpha, \beta, \gamma)^\top \in \mathbb{R}^3$

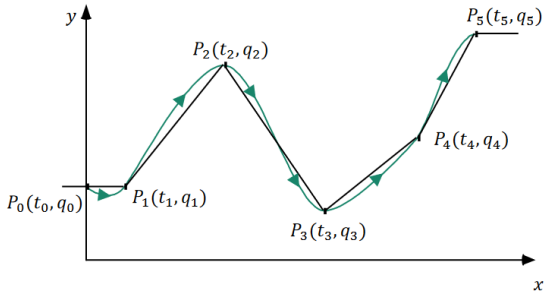


Linearinterpolation:



- $p(t) = \mathbf{p}_{\text{Start}} + \frac{s_p(t)}{s_{ep}} \cdot (\mathbf{p}_{\text{Ziel}} - \mathbf{p}_{\text{Start}})$
- **Berechnung** von $s_p(t)$ und $s_\omega(t)$: s. *5/41-42*

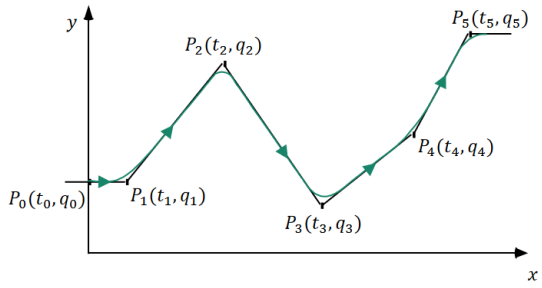
Segmentweise Bahninterpolation mit kubischen Splines:



- Polynom $f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- **Gegeben**: Anfangspunkt $f(0) = s_a$, Endpunkt $f(t_e) = s_e$, Anfangsgeschwindigkeit $\dot{f}(0) = v_a$, Endgeschwindigkeit $\dot{f}(t_e) = v_e$
- **Gesucht**: Parameter $a_0, a_1, a_2, a_3 \in \mathbb{R}$
- Berechnung s. *5/46-50*

Approximierte Bahnsteuerung:

- **Bahninterpolation**: Ausgeführte Bahn verläuft durch alle Stützpunkte der Trajektorie
- **Bahnapproximation**: Kontrollpunkte beeinflussen den Bahnverlauf und werden approximiert



- **Überschleifen** vermeiden „roboterhafte“ Bewegungen

Approximation mit Bernsteinpolynomen:

- **Bézierkurven**: Im Unterschied zu kubischen Splines verlaufen Bézierkurven nicht durch alle Stützpunkte $\mathbf{P}_i$, sondern werden nur von ihnen beeinflusst

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t)\mathbf{P}_i, \qquad \text{wobei } B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

  das i-te **Bernsteinpolynom** vom Grad n ist

- Annäherung an Bézierkurve durch **De-Casteljau-Algorithmus**, der darauf basiert, dass eine Bézierkurve geteilt und durch zwei aufeinanderfolgende Bézierkurven dargestellt wird

7 Bewegungsplanung

Ziel: Erzeugen einer kollisionsfreien Trajektorie unter Berücksichtigung verschiedener Ziele und Einschränkungen

Arbeits- und Konfigurationsraum:

- **Arbeitsraum** W: Kartesischer Raum $\mathbb{R}^6$, in dem sich der **Tool Center Point** (TCP) aufhalten kann
- **Konfiguration** $\mathbf{q} \in C$ beschreibt den Zustand eines Roboters als Gelenkwinkelvektor im Gelenkwinkelraum
- **Konfigurationsraum** $C = C_{\text{free}} \cup C_{\text{obs}}$: n-dimensionaler Raum aller Gelenkwinkelkonfigurationen
- **Freiraum** $C_{\text{free}} = C \setminus C_{\text{obs}}$: Alle kollisionsfreie Konfigurationen. Für komplexere Kinematiken kann C_{free} nicht effizient berechnet werden $\rightarrow$ Verwendung approximativer Verfahren
- **Hindernisraum** C_{obs}: Alle Konfigurationen, die zur einer Kollision führen
- C_{free} und C_{obs} ändern sich während der Ausführung

Bewegungsplanung - Begrifflichkeiten:

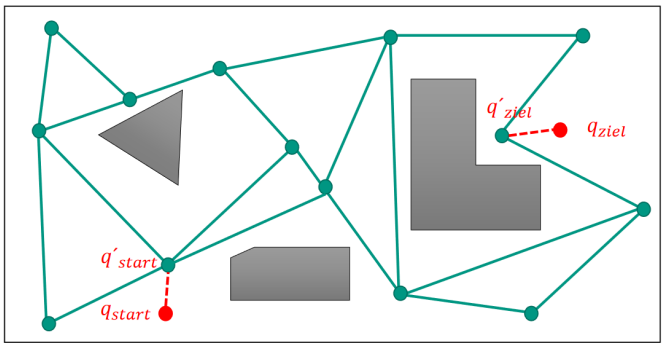
- **Vollständiger Algorithmus**: Findet für ein Problem mindestens eine Lösung oder erkennt in endlicher Zeit, dass keine Lösung existiert
- **Randomisierter Algorithmus**: Verwenden Zufallsgrößen, um den Ablauf zu steuern, wobei oft heuristische Annahmen genutzt werden, um die Berechnung zu beschleunigen
- **Auflösungsvollständiger Algorithmus**: Approximativer Algorithmus, der zusätzlich vollständig ist
- **Probabilistisch-vollständiger Algorithmus**: Findet mindestens eine Lösung falls sie existiert, aber kann nicht erkennen, dass keine Lösung existiert
- Allgemeine Bewegungsplanungsaufgaben sind **PSPACE-vollständig**

Problemklassen der Bewegungsplanung:

- **Klasse a**:
  - **Gegeben**: Vollständiges Weltmodell, vollständige Nebenbedingungen
  - **Gesucht**: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
- **Klasse b**:
  - **Gegeben**: Unvollständiges Weltmodell, unvollständige Nebenbedingungen
  - **Gesucht**: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
  - **Problem**: Kollision mit unbekannten Objekten
- **Klasse c**:
  - **Gegeben**: Zeitvariantes Weltmodell (bewegliche Hindernisse)
  - **Gesucht**: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
  - **Problem**: Hindernisse in Ort und Zeit variant
- **Klasse d**:
  - **Gegeben**: Zeitvariantes Weltmodell
  - **Gesucht**: Trajektorie zu einem beweglichen Ziel
  - **Problem**: Zielzustand in Ort und Zeit beweglich
- **Klasse e**:
  - **Gegeben**: Kein Weltmodell $\rightarrow$ Muss erst erzeugt werden
  - **Gesucht**: Kollisionsfreie Trajektorie vom Start- zum Zielzustand

Ansatz für die Pfadplanung für mobile Roboter:

1. Konstruiere ein Wegenetz W in C_{free}
2. Bilde $\mathbf{q}_{\text{Start}}$ und $\mathbf{q}_{\text{Ziel}}$ auf die nächsten Knoten $\mathbf{q}'_{\text{Start}}$ und $\mathbf{q}'_{\text{Ziel}}$ in W ab
3. Suche in W einen Weg von $\mathbf{q}'_{\text{Start}}$ nach $\mathbf{q}'_{\text{Ziel}}$
4. Finde einen Weg zwischen $\mathbf{q}_{\text{Start}}$ und $\mathbf{q}'_{\text{Start}}$, sowie zwischen $\mathbf{q}'_{\text{Ziel}}$ und $\mathbf{q}_{\text{Ziel}}$

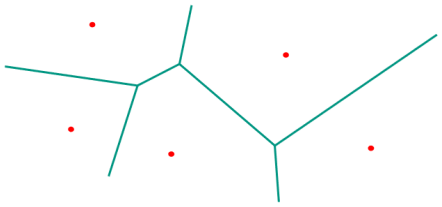


Konstruktion des Wegenetzes W durch Voronoi-Diagramme, Sichtgraphen oder Zellzerlegung.

Suche im Wegenetz durch Baumsuche oder A^*-Algorithmus

Voronoi-Diagramme:

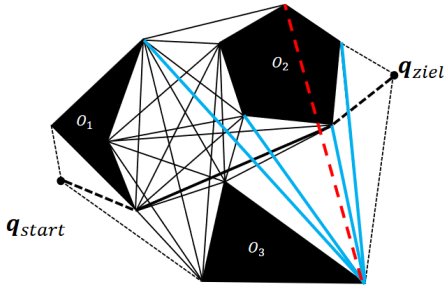
- Region des Voronoi-Diagramms ist definiert als die Menge aller Punkte, deren Abstand zum Hindernis geringer ist als zu allen anderen Hindernissen
- Punkte auf der Grenze zwischen zwei Regionen des Voronoi-Diagramms besitzen den gleichen Abstand zum eigenen und zum benachbarten Hindernis



- **Konstruktion** mit Mittelsenkrechten, s. *7/33-39*
- **Vorteile**: Maximaler Abstand zu Hindernissen
- **Nachteile**: In der Regel ist der gefunden Weg nicht der kürzeste. Bei wenigen Hindernissen werden nur wenige Wege generiert

Sichtgraphen:

- Verbinde jedes Paar von Eckpunkten auf dem Rand von C_{free} durch ein gerades Liniensegment, wenn das Segment kein Hindernis schneidet

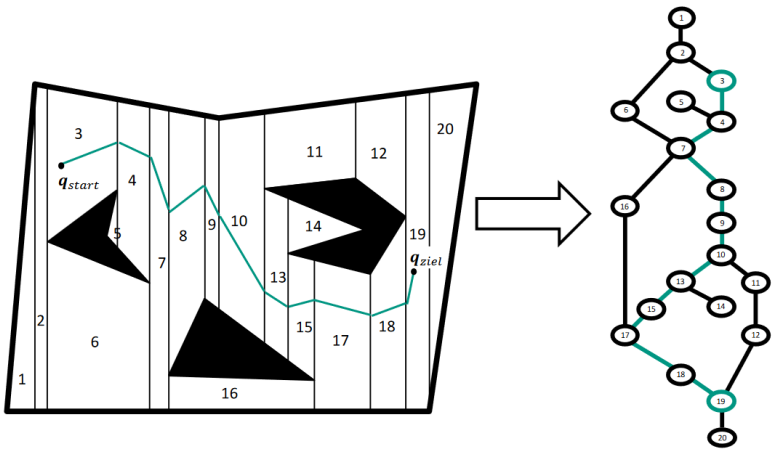


- **Vorteile**: Gefundene Weg ist optimal bei 2D-Problemen
- **Nachteile**: Wege sind nicht zwingend kollisionsfrei, da Hinderniskanten auch Wegsegmente sein können → **Lösung**: Erweiterung der Hindernisse

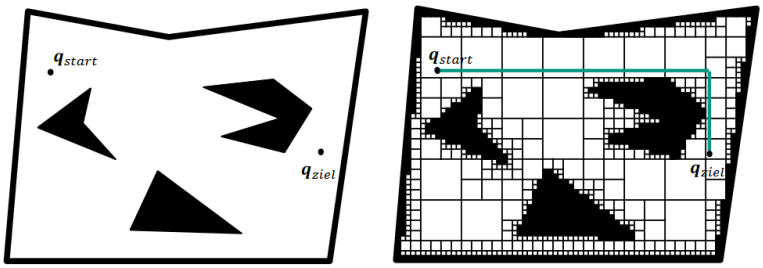
Zellzerlegung:

1. Zerlege C_{free} in Zellen, so dass ein Weg zwischen zwei Konfigurationen innerhalb einer Zelle leicht zu finden ist
2. Stelle die Nachbarschaft in einem Graphen dar
3. Suche den optimalen Weg von $\mathbf{q}_{\text{Start}}$ nach $\mathbf{q}_{\text{Ziel}}$ in dem Graphen

Exakte Zellzerlegung:



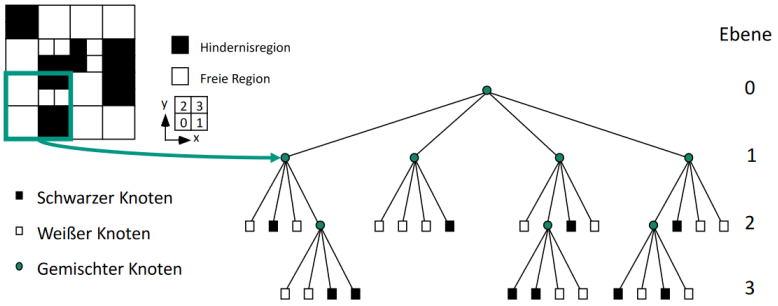
Approximative Zellzerlegung:



- **Vorteil**: Einfache Zerlegung und damit einfachere Wegsuche
- **Nachteil**: Freiraum kann i.A. nur annähernd beschrieben werden

Baumsuche:

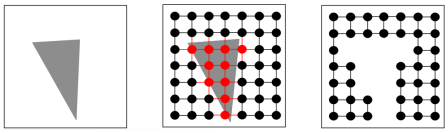
- Darstellung des Konfigurationsraums als **Quadtree**



- Kacheln finden, in denen sich Start- bzw. Zielkonfiguration befinden
- Benachbarte freie Kacheln des Baums vom Start zum Ziel verbinden

A*-Algorithmus:

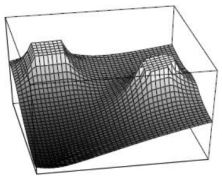
- Liefert kürzesten Pfad von Start nach Ziel
- Diskretisiere kontinuierlichen Raum



- **Kostenfunktion**: $f(x) = g(x) + h(x)$, wobei $g(x)$ Kosten vom Start-Knoten zum Knoten x und $h(x)$ geschätzte Kosten vom x zum Ziel-Knoten entspricht
- **Funktionsweise** ähnlich wie Dijkstra, s. *7/66-76*
- Für zulässige Heuristik h ist A^* optimal effizient und findet optimale Lösung

Potentialfelder:

- Roboter bewegt sich unter dem Einfluss von Kräften, welche ein **Potentialfeld** auf ihn ausübt
- **Potentialfeld** U ist eine Skalarfunktion über dem Freiraum $U\colon C_{\text{free}} \to \mathbb{R}$
- Kraft in einem Punkt $\mathbf{q}$ ist der negative Gradient $F(\mathbf{q}) = -\nabla U(\mathbf{q})$
- Hindernisse erzeugen ein abstoßendes Potential, $\mathbf{q}_{\text{Ziel}}$ ein anziehendes Potential $\rightarrow$ $\mathbf{q}_{\text{Ziel}}$ soll das einzige Minimum sein
- **Potentialfunktion**: *7/82-85*
- Für das Potential- und Kräftefeld gilt: $U(\mathbf{q}) = U_{an}(\mathbf{q}) + U_{ab}(\mathbf{q})$ und $F(\mathbf{q}) = F_{an}(\mathbf{q}) + F_{ab}(\mathbf{q})$

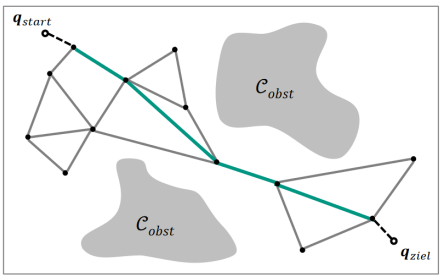


- **Idee**: Roboter läuft entlang eines Tals zum Ziel

Nun folgen Algorithmen für **Bewegungsplanung für Manipulatoren**.

Probabilistic Roadmaps:

- Approximation des Freiraumes durch einen Graphen (**Roadmap**)
- **Schritt 1**: Erzeugung eines kollisionsfreien Graphen durch Wählen zufälliger Punkte (**Sampling**)
  - Mit **lokalen Planer** wird überprüft, ob benachbarte Punkte miteinander verbunden werden können
- **Schritt 2**: Verbinde $\mathbf{q}_{\text{Start}}$ und $\mathbf{q}_{\text{Ziel}}$ mit dem Graphen und suche einen Weg von $\mathbf{q}_{\text{Start}}$ nach $\mathbf{q}_{\text{Ziel}}$ durch den Graphen (z.B. mit A^*)

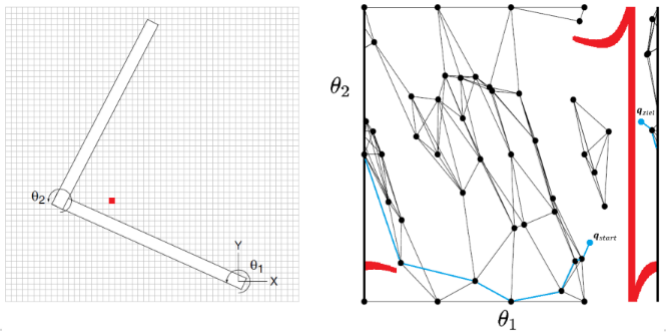


- **Vorteil**: Sehr gut geeignet für statische Umgebungen, Einmalige Konstruktion des Graphen

- **Probleme**: Hängt stark vom verwendeten Sampling ab, Nicht vollständig, da der Graph den Freiraum nur approximiert

Dynamic Roadmaps (DRM):

- Approximation des Konfigurationsraums durch eine **Roadmap**
- Approximation des Arbeitsraums durch **Voxel**
- Abbildung Φ_{WC} von Voxel $\rightarrow$ Roadmap
- **Vorgehen**:
  1. Erzeugung einer selbstkollisionsfreien Roadmap durch Wählen zufälliger Punkte (Sampling)
  2. Erzeugung der Abbildung Φ_{WC} durch Kollisionsüberprüfung zwischen allen Knoten/Kanten und allen Voxeln (sehr rechenaufwändig)
  3. Ermittle alle Voxel mit Hindernis
  4. Lösche alle zugehörigen Kanten und Knoten (mit Φ_{WC} ermittelt) aus der Roadmap
  5. Verbinde $\mathbf{q}_{\text{Start}}$ und $\mathbf{q}_{\text{Ziel}}$ mit dem Graphen und suche einen Weg von $\mathbf{q}_{\text{Start}}$ nach $\mathbf{q}_{\text{Ziel}}$ durch den Graphen

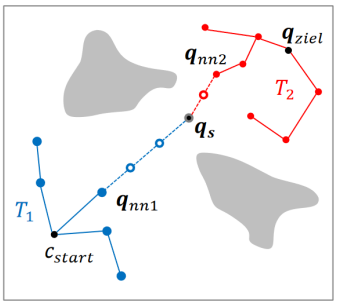


- **Distance Aware Dynamic Roadmap** (DA-DRM) löscht zusätzlich auch alle Voxel in der Nähe von Hindernissen

Rapidly-exploring Random Trees (RRT):

- Algorithmus zur Verarbeitung von einmaligen Anfragen, Keine Vorverarbeitung notwendig
- Probabilistisch vollständiger, randomisierter Algorithmus
- Form von C_{obs} im Konfigurationsraum ist unbekannt $\rightarrow$ Kollisionsprüfung im Arbeitsraum

- **Vorgehen**:
  1. Erzeuge leeren Baum T und füge $\mathbf{q}_{\text{Start}}$ in T ein
  2. Erzeuge einen zufälligen Punkt $\mathbf{q}_S$
  3. Bestimme den nächsten Nachbarn $\mathbf{q}_{nn}$ in T
  4. Füge Punkte auf der Verbindung zwischen $\mathbf{q}_S$ und $\mathbf{q}_{nn}$ in T mit Schrittweite d ein, prüfe jeden der Teilpfade auf Kollision und stoppe, wenn eine Kollision erkannt wurde
  5. Gehe zu 2.
- **Visuelles Beispiel**: *7/109-119*
- **Bidirektionale RRTs**: Baue mit obigem Verfahren 2 Bäume auf, einen von $\mathbf{q}_{\text{Start}}$ und einen von $\mathbf{q}_{\text{Ziel}}$ ausgehend. Eine Lösung ist gefunden, wenn beide Bäume mit $\mathbf{q}_S$ verbunden wurden

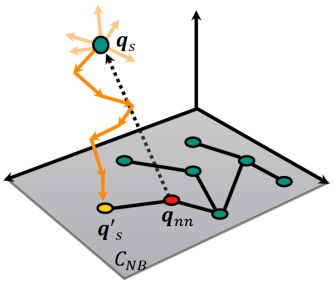


- **Nachbearbeitung (Smoothing)**: Falls Verbindung zwischen zwei zufälligen Konten des Lösungsweges kollisionsfrei ist, dann füge Kante zwischen beiden Knoten hinzu und lösche alle dazwischenliegenden Knoten aus dem Lösungspfad → Erzeugt glattere Trajektorien

Constrained RRT:

- Bei der Bewegungsplanung müssen evtl. Nebenbedingungen erfüllt werden
- **Raum der Nebenbedingungen** $C_{NB} \subseteq C$ können niederdimensionale Gebilde im Konfigurationsraum darstellen
- **Idee**: Projiziere eine Stichprobe $\mathbf{q}_S$ auf eine Konfiguration $\mathbf{q}'_S$, die die Nebenbedingung erfüllt

- **Randomized Gradient Descent**:
  1. Zufällige Bestimmung von n Nachbarn von $\mathbf{q}_S$
  2. Falls die Distanz eines Nachbarn zu C_{NB} kleiner als die Distanz von $\mathbf{q}_S$ zu C_{NB}, ersetze $\mathbf{q}_S$ mit diesem Nachbarn
  3. Wiederholen bis maximale Iterationszahl erreicht oder die Distanz von $\mathbf{q}_S$ zu C_{NB} kleiner ist als α



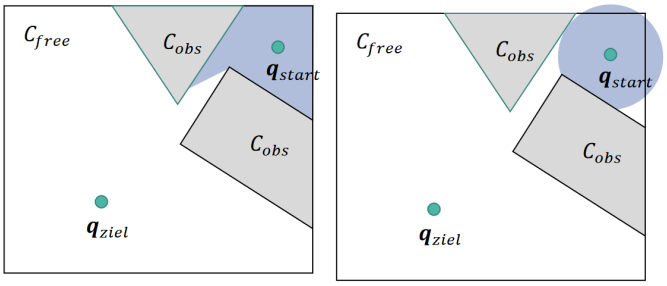
- **First Order Retraction**: Berechne $\mathbf{q}'_S$ mithilfe der Jacobi-Matrix

RRT*:

- RRT* optimiert Suchbaum iterativ während der Suche
- Optimierung des Suchbaums aufgeteilt in zwei Schritte:
  1. Ermittle zu jedem neuen Knoten die Kosten
  2. Rewiring des Suchbaums beim Hinzugefügen neuer Knoten
  3. Siehe *7/135-138*
- **Nachteil**: Längere Laufzeiten, Uni-direktionaler Ansatz

Enge Passagen:

- **Problem**: Klassische RRTs können viel Zeit benötigen, bis eine Lösung für einen Durchgang durch eine enge Passage gefunden wird
- **Ideal**: Nur in sichtbarer Voronoi Region eines Knotens sampeln → **Problem**: Berechnung aufwendig → Approximation durch Kugel mit Radius r (**Dynamic Domain**)



- **Bridge Sampling**:
  1. Wähle gleichverteilt einen zufälligen Punkt $\mathbf{q}_1 \in C_{\text{obs}}$
  2. Wähle nach einer geeigneten Wahrscheinlichkeitsverteilung einen zweiten Punkt $\mathbf{q}_2 \in C_{\text{obs}}$ in der Nähe von $\mathbf{q}_1$
  3. Wenn der Mittelpunkt $\mathbf{q}_S$ zwischen $\mathbf{q}_1$ und $\mathbf{q}_2$ in C_{free} liegt, dann verwende ihn als neue Stichprobe
  4. Wiederhole