

Routiges Folhel:

WY

$$R = \exp([a]_x) = I + \sin(\|a\|) \left[\frac{a}{\|a\|} \right]_x + (1 - \cos(\|a\|)) \left[\frac{a}{\|a\|} \right]^2_x$$

achs symmetrisch

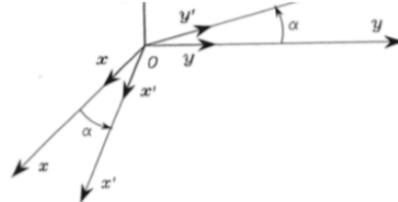
$$[a]_x = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

Ihre Hauptanwendung liegt darin, dass das Ergebnis einer Drehung um die Achse a mit Winkel $\|a\|$ als Matrix beschreibt.

WY

Elementar Rotations-Matrizen:

$$R_z(\alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Corresponding rotation matrices for rotations β about the y axis and γ about the x axis:

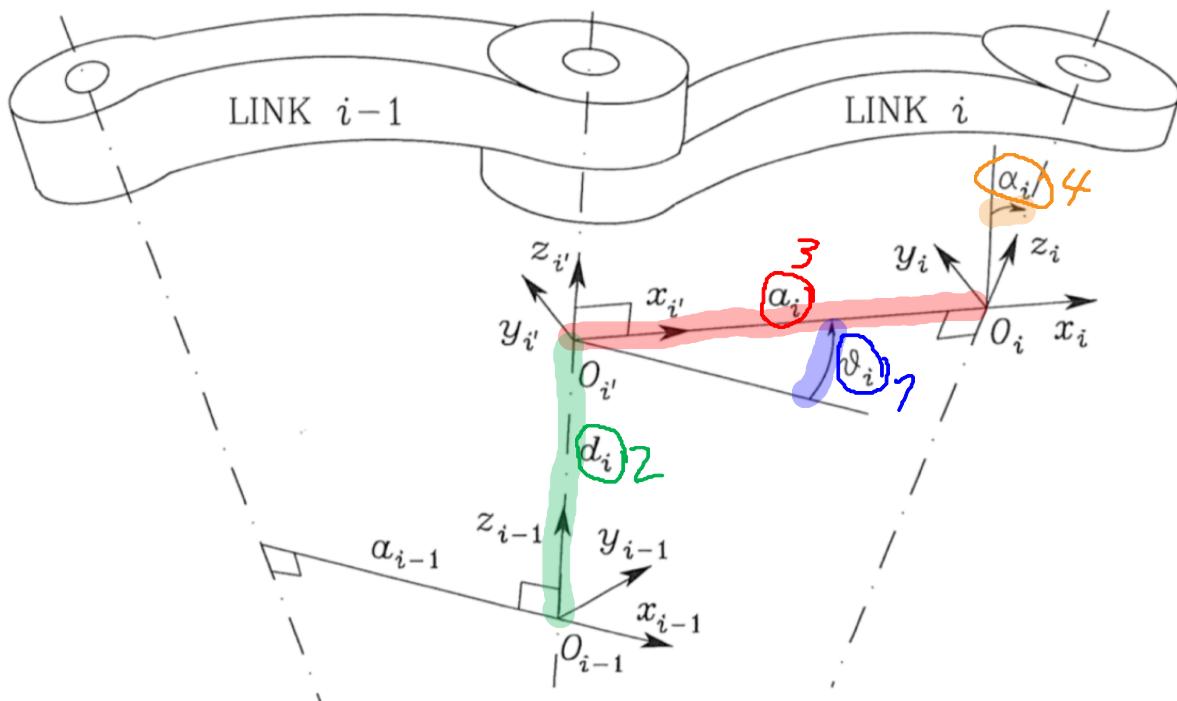
$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{bmatrix}$$

DH-Transformation

JOINT $i-1$

JOINT i

JOINT $i+1$



Eine festgelegte Konvention für eine kinematische Beschreibung von kinematischen Ketten. Hauptzweck dieser Konvention ist eine einheitliche kinematische Beschreibung festzulegen, die jeder sofort nachvollziehen kann. Eine DH-Transformation besteht aus einer festgelegten Verkettung folgender Transformationen (Siehe Abbildung):

- 1: Rotation um z_{i-1} Achse um θ_i grad
- 2: Translation entlang z_{i-1} -Achse um d_i
- 3: Translation entlang x_i' -Achse um a_i
- 4: rotation um x_i' -Achse um α_i grad

- > Drehrichtungen und Koordination nach rechter Handregel
- > Aus den 4 Transformationen lässt sich eine allgemeine homogene Matrix aufstellen
- > Handelt es sich um ein rotationsgelenk, ist θ variabel
- > Handelt es sich um ein translatorisches Gelenk ist d variabel

Newton:

Newton1: Ein kräftefreier Körper bleibt in Ruhe oder bewegt sich geradlinig mit konstanter Geschwindigkeit. oder rotiert mit konstant omega.

2: Kraft gleich Masse mal Beschleunigung (Vectorschreibweise) oder Massenträgheit mal Winkelbeschleunigung.

3: Kraft gleich Gegenkraft: Eine Kraft | Drehmoment von Körper A auf Körper B geht immer mit einer gleich großen, aber entgegen gerichteten Kraft oder Moment von Körper B auf Körper A einher.

$$1: \quad \vec{F}_{\text{all}} = 0, \quad \text{dann } \vec{v} = \text{const}, \quad \vec{\omega} = 0$$

$$M = 0, \quad \text{dann} \quad \vec{\omega} = \text{const}, \quad 0$$

$$2: \quad \vec{F} = m \cdot \vec{a}$$

$$M = I \cdot \vec{\alpha}$$

$$3: \quad \vec{F}_A = \vec{F}_B, \quad M_A = M_B$$

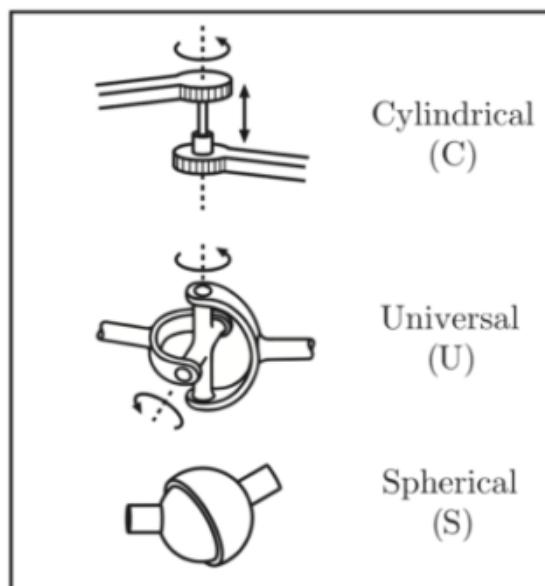
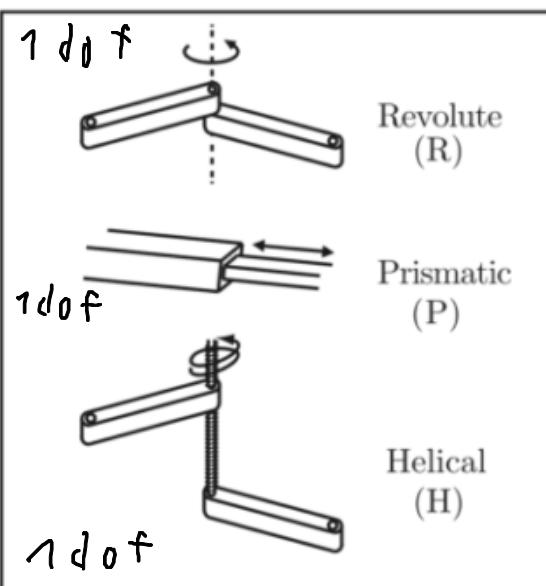
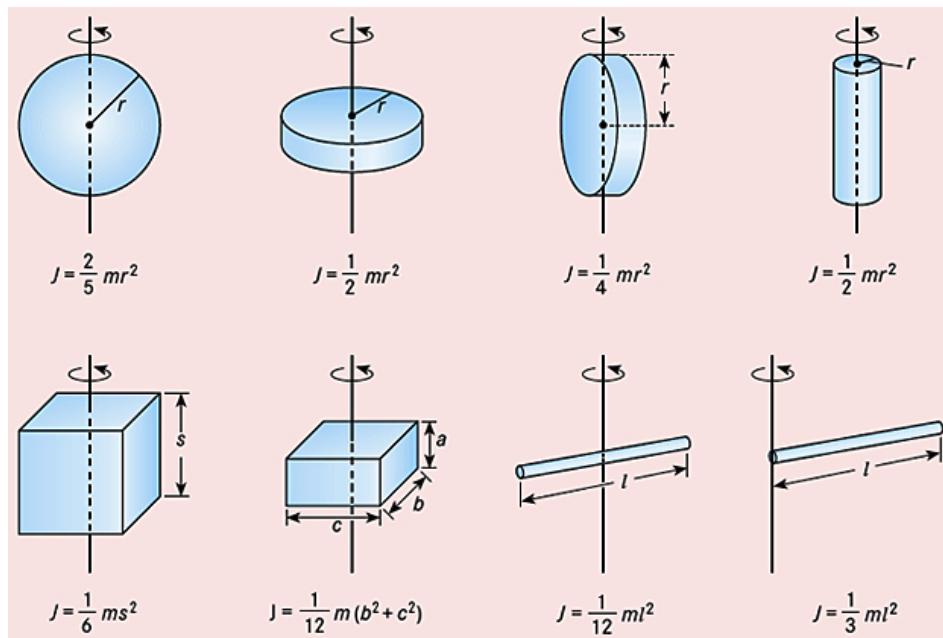
Grüblers Formel

Links + Ground

$$\text{dof} = m(N - 1 - J) + \sum_{i=1}^J f_i.$$

2 joints

Joints
summe aller Freiheitsgrade die alle joints hergeben



Jacobi-Matrix

Jacobi Matrix aufstellen:

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} J_{P_1}(q) & \dots & J_{P_n}(q) \\ J_{O_1}(q) & \dots & J_{O_n}(q) \end{bmatrix} \dot{q},$$

Index wird so in die Frames gelegt:
i=0 ist der Origin, bzw. das MCS!
i=n ist letztes Frame, also unser TCP, bzw. der Endeffektor

Herkunft:
 $P(t) = f(\theta(t))$
 $V = \frac{d}{dt} f(\theta(t))$
 $V = \frac{df}{d\theta}(\theta) \cdot \dot{\theta}$

chain-rule

$\mathbf{J}(q) = \begin{bmatrix} z_{i-1} \\ \mathbf{0} \\ z_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \end{bmatrix}$

Ist also die letzte Spalte der Rot-Matrix, die die Rotation des Frames *i-1* relativ zum Origin beschreibt

$\mathbf{R}_{i-1}^0 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

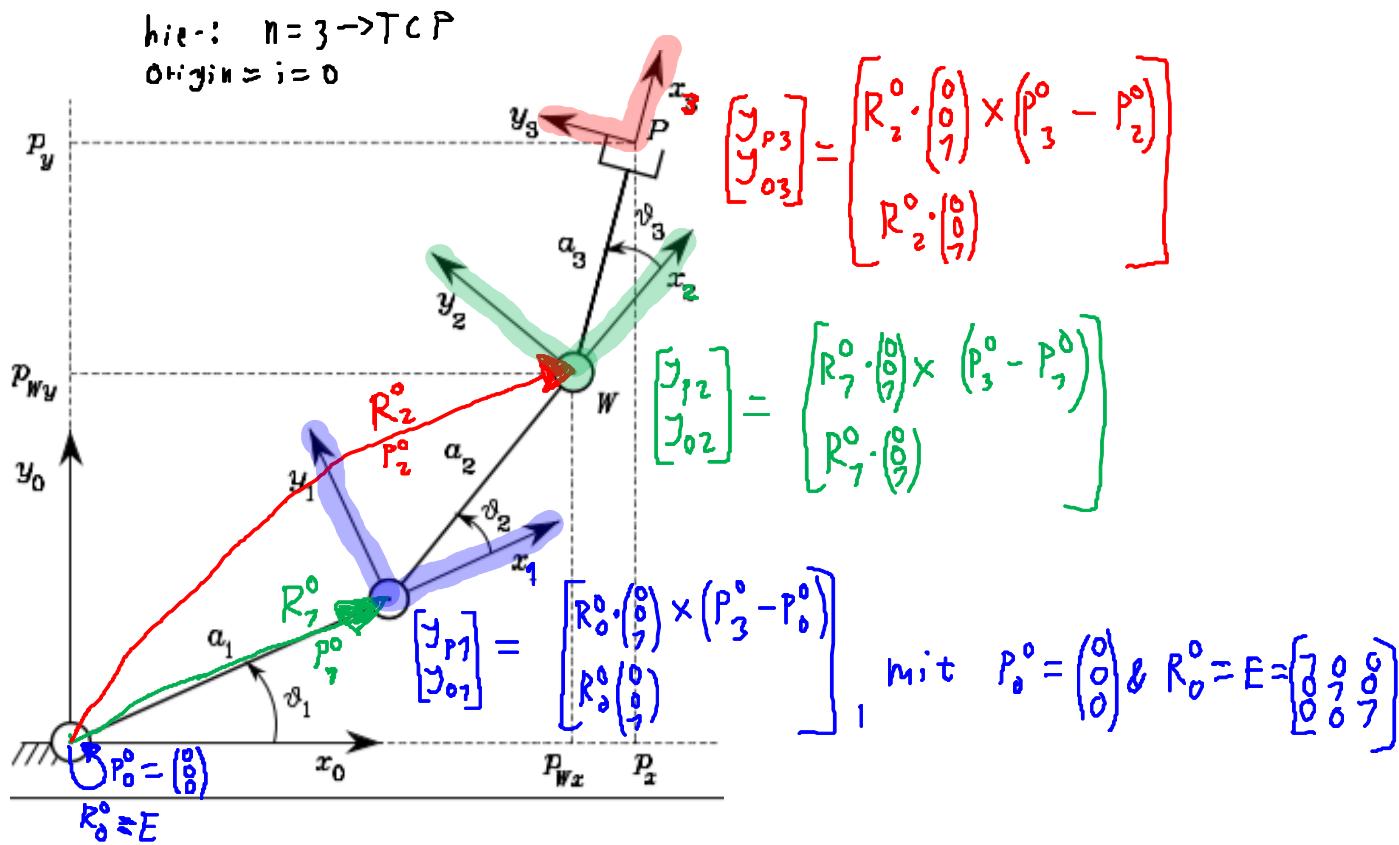
Rot-Matrix, Frame *i-1*, relativ zu Frame 0

translatorisches Gelenk

rotatorisches gelenk

Position des Endeffektors
 $\hookrightarrow \mathbf{p}_n^0$

Position von Frame *i-1* rel zu Frame 0
 $\hookrightarrow \mathbf{p}_{i-1}^0$



Singularität:

Wenn \mathbf{J} keinen vollen Rank hat, ist es eine Singularität (An einer spezifischen Roboterarmorientierung) -> Eine Matrix hat genau dann einen vollen Rank, wenn $\det(M)=0$

$$\det \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = a_{11} a_{22} a_{33} + a_{12} a_{23} a_{31} + a_{13} a_{21} a_{32} - (a_{31} a_{22} a_{13} + a_{32} a_{21} a_{11} + a_{33} a_{21} a_{12})$$

oder wenn $M^* M^{-1} T = E \rightarrow M^{-1} = M^* T$

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

determinant

Eigenschaften der Jacobi matrix

$$V_{tip} = \dot{J} \dot{\theta}_{joints} = \text{"differential kinematics"}$$

$$J = J^T f_{tip} = \text{"dynamic manipulability"}$$

\tilde{J} = Kräfte/Momente in Gelenke

f_{tip} = Kräfte an TCP

$$M = lin(q)$$

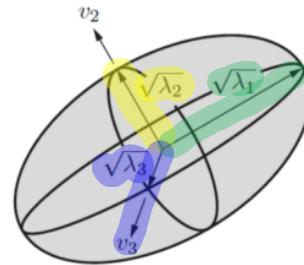
$$\begin{bmatrix} \cdot \\ \cdot \\ \vdots \\ \cdot \end{bmatrix}$$

$\dim(\text{op-space}) = n$

manipulability ellipsoid $J J^T$

- Instead of looking at the shape of the ellipsoid, it is useful to define a single measure
- Semi-axes are defined by singular values of J

$$\sigma_i = \sqrt{\lambda_i(J J^T)}, \text{ for } i = 1, \dots, r$$



$$A = \begin{pmatrix} 3 & -1 & 0 \\ 2 & 0 & 0 \\ -2 & 2 & -1 \end{pmatrix}$$

1.) Berechnen des charakteristischen Polynoms

$$\chi_A(\lambda) = \begin{vmatrix} (3-\lambda) & -1 & 0 \\ 2 & (0-\lambda) & 0 \\ -2 & 2 & (-1-\lambda) \end{vmatrix} = (3-\lambda) \cdot (0-\lambda) \cdot (-1-\lambda) - (-1-\lambda) \cdot 2 \cdot (-1) = -\lambda^3 + 2\lambda^2 + \lambda - 2$$

2.) Berechnen der Nullstellen des charakteristischen Polynoms

(vgl. Kapitel "Kubische Gleichungen lösen")

$$\lambda_1 = 1, \quad \lambda_2 = 2, \quad \lambda_3 = -1;$$

$$x^2 + px + q = 0$$

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$$

Different measures for manipulability:

- ratio of the longest and shortest semi-axes of the manipulability ellipsoid

$$\mu_1(A) = \frac{\sqrt{\lambda_{\max}(A)}}{\sqrt{\lambda_{\min}(A)}} = \sqrt{\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}} \geq 1$$

- condition number of the matrix A

$$\mu_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \geq 1$$

- volume of the manipulability ellipsoid

$$\mu_3(A) = \sqrt{\lambda_1 \lambda_2 \dots} = \sqrt{\det(A)}.$$

(smaller values are better)

(a larger value is better)

force ellipsoid

$$(J J^T)^{-1}$$

Euler Lagrange:

- Lagrange function:

$$\mathcal{L} = T - U$$

Kinetic Energy Potential Energy

- (Euler-)Lagrange equation for every component (generalized coordinate)

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \xi_i$$

generalized forces
 external forces die auf Segment,
 i = 1, ..., n
 wirken
 generalized coords

ξ_i is the generalized force corresponding to coordinate q_i

- $q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \end{bmatrix}$ Sollte sinnvoll wählen, so dass möglichst simpel
 - dann T aufstellen:
 Dabei stets Massenpunkte betrachten (nacheinander)
 wie dreht und translasiert Massenpunkt A..B...?
 $\hookrightarrow \frac{1}{2} I \dot{q}^2 \quad \hookrightarrow \frac{1}{2} m v^2$
 worum dreht betrachteter Schwerpunkt?
 - dann U aufstellen:
 wieder Massenpunkte nach einem der abhandeln:
 Koordinatensystem beachten & U kann auch negative Terme beinhalten
 - Lagrange Funktion aufstellen, evtl. $x = f(q)$ überführen
 $\hookrightarrow \mathcal{L} = T - U \stackrel{!}{=} f(\vec{q})$ & konstante
 - Lagrange-Gleichung für jedes q_i nacheinander aufstellen.
 gleich Bew-Gleichung
 $\hookrightarrow \xi_i \neq 0 \rightarrow$ welche F (torques) wirken auf Segment?
 - Gleichungen nach Thau_i auflösen. Für jedes Thau_i eine Gleichung.
- | | | | | |
|-------------------------|---|-------------------------------|--------------------------------|-----------------------|
| $B(q)\ddot{q}$ | $+ C(q, \dot{q})\dot{q}$ | $= 0$ | \Rightarrow | only θ 's |
| Inertia term s.above | Nonlinear term, s. above | Damping (viscous friction) | Friction (Coulomb friction) | Gravity-term, s.above |
| $= \tau$ | $J^T(q)h_e$ | $\rightarrow h_e = F_{trp}$ | | |
| Actuator torques | Forces and torques between end effector and the environment, Transformed by corresponding Jacobian | | | |

Newton-Euler

1) Free Body Diagram zeichnen

- > Schnittkräfte(je nach Gelenktyp unterschiedlich)
 - > Aktorkräfte (Richtung=DOFJoint)
 - > äußere Kräfte (g)
 - > Newton 3 beachten

2) Newton 2 anwenden

- > summe aller ... in ...-Richtung
 - > Achtung! jedes freigeschnittene Segment einzeln betrachten. Beispiel translation: x1 von Segment1 hat nichts mehr zu suchen in Gleichungen für Segment2! hier arbeitest du mit x2!

3) Unbekannte entfernen

- > einsetzen
 - > dgl = $f(q \& \text{Konstanten})!!!$
 - > das gefundene, unsortierte Differentialgleichungssystem kann nun noch sortiert werden, in diese form:

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + F_v\dot{q} + F_s \operatorname{sgn}(\dot{q}) + g(q) =$$

Inertia term
s.above Nonlinear term, s. above Damping (viscous friction) Friction (Coulomb friction) Gravity-term, s.above

$$= \tau - J^T(q)h_e$$

Actuator torques Forces and torques between end effector and the environment ,
 Transformed by corresponding Jacobian

Trajektorienplanung / Path-planning

- Trajectory is the combination of

- The path, i.e. a purely geometric description of the sequence of configurations achieved by the robot,

$$\theta : [0, 1] \rightarrow \Theta \quad (0: \text{start of path}, 1: \text{end of path})$$

- A time scaling, which specifies the times when those configurations are reached.

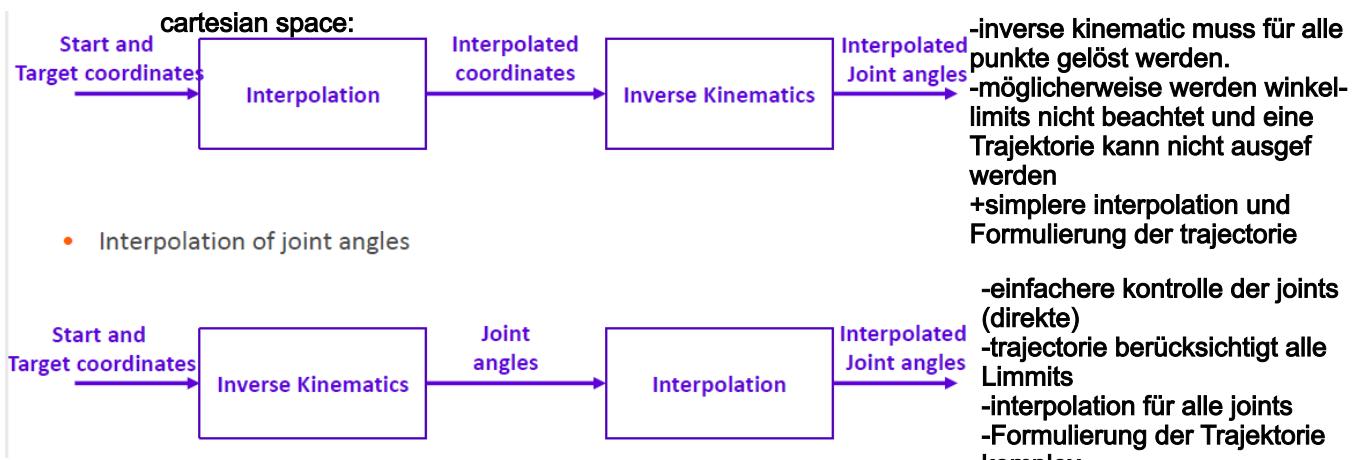
$$s : [0, T] \rightarrow [0, 1]$$

→ theta und s müssen zweimal differenzierbar sein, weil:

damit auch die Beschleunigung eindeutig beschrieben wird

→ zwei Möglichkeiten Trajektorie zu planen:

im joint space (näher an Funktionsweise des Roboters)
im cartesian space, ist näher an der zu lösenden Aufgabe



→ zwei Möglichkeiten Trajektorienplanung im joint space:

- asynchron: Steuerung der Achsen unabhängig voneinander

- synchron: Gibt Leit-Achse, alle Achsen starten und enden gleichzeitig mit ihrer Bewegung. Anwendungen: schweißen, mahlen...

Trajektorie Programmierung

→ nenne Methoden dies da gibt:

- teach in: mit control panel oder so kann Roboter manuell zu key-points angefahren werden. dann PTP movements
- playback: Roboter geht in zero force mode und kann intuitiv per hand an Punkte herangefahren werden. Ist schwierig zu entwickeln so ein no force mode. key-points können nicht sehr exakt angefahren werden. Grad schwerer Roboter aus Sicherheitsgründen schwierig für sowas. Daher fürs ungenaue Kleben oder mahlen geeignet.

Trajektorien-Interpolation

-> Interpolationstypen:

- synchron PTP und asynchron PTP
- linearbahn
- Zirkularbahn
- Spline-Bahn (mehrere punkte, smooth miteinander verbunden, z.B. mit kubischen polynomfunktionen)

-> velocity profiles:

- trapezodial (trapezförmig) profile:

durch die einfach gehaltene Form gibt es Sprünge in den Beschleunigungen. kann zu mechanischen Oszillationen führen. gibt Vmax, amax, tb, tv, te. wird zeit-abschnitt-weise definiert wie schnell ist und wie schnell a sein darf und so.

- sinoid profile (sinoidenförmig?):

ist smoother. keine beschleunigungssprünge, Roboter wird daher besser entlastet, dafür auch längere beschleunigungszeiten.

-> neben interpolation gibt's auch approximation, hier muss der pfad nicht zwingend durch vorgegebene punkte druch gehen

Motion Planning

-> um was getsn do?

im gegensatz zur trajektorienplanung wird auch die dynamic berücksichtigt und nicht nur die kinematic und der Roboter wird mit Kräften oder Beschleunigungen (u) gesteuert.

Given an initial state $x(0) = x_{\text{start}}$ and a desired final state x_{goal} , find a time T and a set of controls $u : [0, T] \rightarrow \mathcal{U}$ such that the motion (10.2) satisfies $x(T) = x_{\text{goal}}$ and $q(x(t)) \in \mathcal{C}_{\text{free}}$ for all $t \in [0, T]$.

-> Verhältnis von Vontrol-inputs zu DOF:

gilt CIP=DOF ist es einfach einen pfad auszuführen. Gilt aber CIP<DOF, wird's schwieriger einem pfad zu folgen. Beispiel: Auto in der Ebene hat nur 3 DOF, nur 2 control inputs.

-> Online/Offline planning:

-ist die Umgebung statisch, kann offline Planung gemacht werden.

-Ändert sich die Umgebung, muss schneller online planer verwendet werden (Autonomes Fahren)

-> Cfrees und CObstacle?

Cfree is der freie configurationsspace, der nicht anfahrbare configurationsspace ist Cobs. Auch Gelenk-Limits werden im C_obs abgebildet, nicht nur obstacles/Hindernisse.

-> Wie kann man Geometrien des Roboters in Cfrees space übertragen/berücksichtigen?

- man erstellt ein "grown"-C-space. Beispiel runder Staubsaugroboter lässt obstacles im workspace einfach um seinen Radius wachsen -> volia, schon hat man den Cfrees space gefunden unter Berücksichtigung seiner eigenen Geometrie.

-> wie kann man Cfrees erstellen mit polygonal Roboter der rotiert?

- jetzt wird aus einem 2d workspace ein 3D Cfrees space. Roboter lässt wieder obstacles wachsen, jedoch in unterschiedlichen winkel-ausichtungen. so entsteht dritte dimension

-> wie kann man Kollisions-erkennung approximieren?

-> komplex geformte Hindernisse können approximiert werden, mit überlappenden Kugeln. So ist Berechnung weniger komplex (Beispiel Lampe)

-> Path planer:

- A*

für diskretisierte Räume, Graphenbasiert.

- Dijkstras Algorithmus (

auch graphenbasiert und diskret)

- Für dijkstra und A* muss search space diskretisiert werden in ein graph. hierfür werden 4er(Manhattan distance) oder 8er(euclidische ditanz)

Nachbarschaften=Bewegungsrichtungen festgelegt um mögliche Bewegungsrichtungen festzulegen. Nachbarn werden nur in OPEN übergeben wenn nix im weg ist.

- complete path planners

hier wird cfree exakt dargestellt. Beispiel:

Zuerst wird free space mit obstacle-growing erzeugt. Dann werden Linien zwischen allen "sich sehenden" Ecken gespannt, Roboter (als punkt dargestellt) eingeschlossen.

- wavefront planner

das gesamte grid wird so verarbeitet das in jeder zelle bereits drinn steht wie nahe es von ihr aus zum Ziel ist. So kann Roboter von vielen Startpunkten aus "hinunter zum "Ziel gelangen, welches quasie ein minimum repräsentiert (Zielzelle=0). Ziel könnte bspw eine Ladestation sein die häufig von unterschiedlichen Startpunkten angefahren werden muss

-> multiresolution grid representation:

ein ungleichmäßig aufgelöstes grid. Gegenden in denen nichts ist, sind grob aufgelöst, Gegenden in denen bspw Hindernisse sind, werden feiner (zum Hindernis hin) aufgelöst.