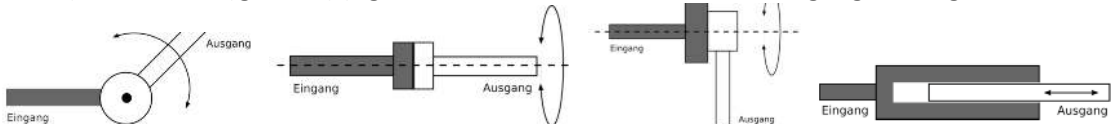


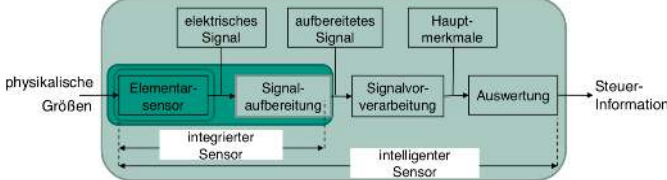
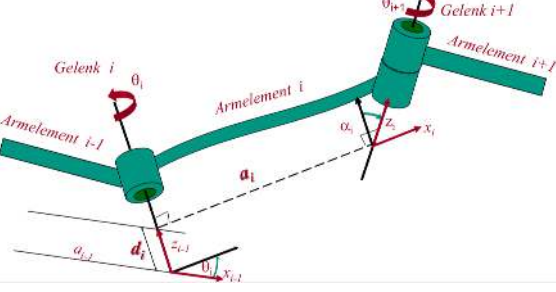


Zusammenfassung

Robotik I (Karlsruher Institut für Technologie)

Kinematik	Analysiert die Geometrie eines Manipulators oder Roboters. Das essentielle Konzept ist die Position.								
Dynamik	Analysiert die Kräfte und Momente, die durch Bewegung und Beschleunigung eines Mechanismus und einer zusätzlichen Last entstehen.								
Kinematische Kette	Beschreibt einen Satz von Gliedern (Körpern) die durch Gelenke kinematisch verbunden sind.								
Freiheitsgrade	Anzahl der unabhängigen Parameter, die zur kompletten Spezifikation der Position eines Mechanismus/Objekts benötigt werden. (degrees of freedom DoF)								
Rotationsmatrizen	$\begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$ $R_z \quad R_y \quad R_x$								
Verkettung von Rotationen	<u>Rotation um lokale Achsen:</u> $((R_{z,\alpha} * R_{y,\alpha}) * R_{x,\alpha}) * a$ <u>Rotation um globale Achsen:</u> $R_{z,\alpha} * (R_{y,\alpha} * (R_{x,\alpha} * a))$								
Bewertung Rotationsmatrizen	<table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td></td><td>Redundanz: neun Werte für eine Rotationsmatrix</td></tr> <tr> <td></td><td>Nicht geeignet für maschinelles Lernen da Einträge voneinander abhängig sind</td></tr> <tr> <td></td><td>Schwierige Interpolation</td></tr> </tbody> </table>	Pro	Contra		Redundanz: neun Werte für eine Rotationsmatrix		Nicht geeignet für maschinelles Lernen da Einträge voneinander abhängig sind		Schwierige Interpolation
Pro	Contra								
	Redundanz: neun Werte für eine Rotationsmatrix								
	Nicht geeignet für maschinelles Lernen da Einträge voneinander abhängig sind								
	Schwierige Interpolation								
Rotationsmatrix zu Eulerwinkel	$R = R_z(\alpha) \cdot R_x(\beta) \cdot R_y(\gamma) = \text{Multiplikation der Rotationsmatrizen} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}$ $\alpha = \arctan\left(-\frac{a_x}{a_y}\right) \quad \beta = \arccos(a_z) \quad \gamma = \arctan\left(\frac{n_z}{o_z}\right)$								
Rotationsmatrix zu RPY (XYZ) Winkel	$R = R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha) = \text{Multiplikation der Rotationsmatrizen} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}$ $\alpha = \arctan\left(\frac{o_z}{a_z}\right) \quad \beta = \arcsin(-n_z) \quad \gamma = \arctan\left(\frac{n_y}{n_x}\right)$								
Gimbal Lock	Bei bestimmten Winkeln werden Achsen voneinander abhängig und ein Freiheitsgrad geht verloren.								
Roll Pitch Yaw und Eulerwinkel Bewertung	<table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Kompakter als Rotationsmatrizen</td><td>Nicht eindeutig</td></tr> <tr> <td>Aussagekräftiger als Rotationsmatrizen</td><td>Nicht kontinuierlich bei kontinuierlicher Rotation</td></tr> <tr> <td></td><td>Gimbal Lock</td></tr> </tbody> </table>	Pro	Contra	Kompakter als Rotationsmatrizen	Nicht eindeutig	Aussagekräftiger als Rotationsmatrizen	Nicht kontinuierlich bei kontinuierlicher Rotation		Gimbal Lock
Pro	Contra								
Kompakter als Rotationsmatrizen	Nicht eindeutig								
Aussagekräftiger als Rotationsmatrizen	Nicht kontinuierlich bei kontinuierlicher Rotation								
	Gimbal Lock								
Eulerwinkel vs. Roll Pitch Yaw	<table border="1"> <thead> <tr> <th>Eulerwinkel</th><th>Roll Pitch Yaw</th></tr> </thead> <tbody> <tr> <td>Multiplikation von links nach rechts</td><td>Multiplikation von rechts nach links</td></tr> <tr> <td>Jede Drehung in Bezug auf neues KS</td><td>Jede Drehung in Bezug auf BKS</td></tr> <tr> <td>Drehung jeweils um veränderte Achsen</td><td>Drehung um jeweils unveränderte Achsen</td></tr> </tbody> </table>	Eulerwinkel	Roll Pitch Yaw	Multiplikation von links nach rechts	Multiplikation von rechts nach links	Jede Drehung in Bezug auf neues KS	Jede Drehung in Bezug auf BKS	Drehung jeweils um veränderte Achsen	Drehung um jeweils unveränderte Achsen
Eulerwinkel	Roll Pitch Yaw								
Multiplikation von links nach rechts	Multiplikation von rechts nach links								
Jede Drehung in Bezug auf neues KS	Jede Drehung in Bezug auf BKS								
Drehung jeweils um veränderte Achsen	Drehung um jeweils unveränderte Achsen								
Rotationsmatrix zu Quaternion	<u>Rotationsachse</u> $(R - I) \cdot x = 0 \rightarrow \text{Ausmultiplizieren zu LGS und dann lösen} \rightarrow \text{Rotationsachse } x$ <u>Rotationswinkel</u> Methode 1: $\alpha = \arccos\left(\frac{\text{Spur}(R)-1}{2}\right)$ Methode 2: Vektor v so, dass $v \cdot x = 0 \rightarrow v' = R \cdot v \rightarrow \alpha = \arccos\left(\frac{v \cdot v'}{\ v\ \cdot \ v'\ }\right)$ <u>Umwandlung in Quaternion</u> $q = \left(\cos\left(\frac{\alpha}{2}\right), x \cdot \sin\left(\frac{\alpha}{2}\right)\right) \rightarrow 4 \text{ Werte}$								
Transformation einer Transformationsmatrix	$T_{trans} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad t_x, t_y \text{ \& } t_z \text{ beschreibt Translation um } \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \text{ eines Punktes.}$ Rotation lässt sich bestimmen durch die blaue 3 x 3 Matrix gemäß Rotationsmatrizen.								
Transformationsmatrix Anwendung	$v = (1, 2, 3)^T \rightarrow v = (1, 2, 3, h)^T \rightarrow v' = T \cdot v$								

Inverse Transformationsmatrix	$T^{-1} = \begin{pmatrix} n_x & n_y & n_z & -n^T v \\ o_x & o_y & o_z & -o^T v \\ a_x & a_y & a_z & -a^T v \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
Berechnung von Transformationsmatrizen	<u>Transformation:</u> siehe Transformation einer Transformationsmatrix <u>Rotation:</u> Bestimme die Rotationsmatrix und setze diese in T ein	
Verkettung Transf.	$T_{\text{final}} = T_{\text{init}} \cdot T_1 \cdot T_2 \cdot T_3 \dots$	
Affine Transformation Bewertung	Pro	Contra
	Drehachse und -reihenfolge implizit enthalten	12 nichttriviale Kenngrößen
		Redundanz wegen Orthogonalität
Translations- und Rotationsdifferenz Transf.matr.	Translationsdifferenz: $\Delta t = \ t_{\text{Goal}} - t_{\text{TCP}}\ = \sqrt{(t_{G,1} - t_{T,1})^2 + (t_{G,2} - t_{T,2})^2 + (t_{G,3} - t_{T,3})^2}$ Rotationsdifferenz: $\Delta \alpha = \arccos\left(\frac{\text{Spur}(R_{\text{TCP}}^T R_{\text{Goal}}) - 1}{2}\right)$	
Quaternion allg.	$q = (a, u)^T = a + u_1 i + u_2 j + u_3 k$ mit $a \in \mathbb{R}, u \in \mathbb{R}^3$ und $k = i * j$	
Punkt zu Quaternion	Punkt $p = (u_1, u_2, u_3) \rightarrow$ Quaternion $(a, p) = a + u_1 i + u_2 j + u_3 k$ ($a=0$ da keine Rotation)	
Rotations-quaternion Winkel ϕ und Achse a	Quaternion $q = (\cos \frac{\phi}{2}, v \sin \frac{\phi}{2})$ wobei v die Achse ist beschrieben durch $(v_1, v_2, v_3)^T$ $\rightarrow \cos \frac{\phi}{2} + i \cdot v_1 \sin \frac{\phi}{2} + j \cdot v_2 \sin \frac{\phi}{2} + k \cdot v_3 \sin \frac{\phi}{2}$ Konjugiertes Quaternion $q^* = (a, -u)$	
Transformation eines Punktes mittels Quaternion	$v' = q \cdot v \cdot q^*$	
SLERP Interpolation	$\theta = \arccos(q_1 \cdot q_2) \rightarrow \text{slerp}(q_1, q_2, t) = \frac{\sin((1-t)\theta)}{\sin \theta} * q_1 + \frac{\sin(t)\theta}{\sin \theta} * q_2$	
Quaternion Bewertung	Pro	Contra
	Kompakte & Anschauliche Darstellung	Nur Rotation, keine Translation
	Konkatenation möglich	
	Kein Gimbal Lock	
	Berechnung der Inversen Kinematik	
	Repräsentation stetig	
Duale Quaternionen	Ermöglichen die bei Quaternionen fehlende Translation mittels Dualzahlen (und auch Rotation). Allerdings schwer für Anwender und komplexe Verarbeitung.	
Arbeitsraum	Der Arbeitsraum besteht aus den Punkten im 3D Raum, die von der Roboterhand angefahren werden können. Hierzu sind 3 DoF in der Bewegung, als mindestens 3 Gelenke erforderlich.	
Gelenktypen	Rotationsgelenk (R): Drehachse bildet rechten Winkel mit den Achsen der Glieder Torsionsgelenk (T): Drehachse parallel zu Achse der beiden Glieder Revolvergelenk (V): Eingangsglied parallel zu Drehachse, Ausgangsglied 90° zu Drehachse Linear(Translations-)gelenk (L): gleitende oder fortschreitende Bewegung entlang Achse 	
Muskelartige Antriebe	<u>Pneumatische Antriebe:</u> Komprimierte Luft bewegt Kolben (Schnelle Zyklen mit wenig Kraft) <u>Hydraulischer Antrieb:</u> Öldruckpumpe und steuerbare Ventile. Große, starke Roboter.	
Elektrische Antriebe	Elektrische in mechanische Energie. Stromdurchflossener Leiter von Magnetfeld abgelenkt und durch Wechsel der Polarität Drehbewegung. Kraft proportional zu Strom und Magnetfeldstärke \rightarrow kompakt, leise und hohe Positionierungsgenauigkeit aber nur wenig Kraft	

Rad-konfigurationen	<p><u>Differentialantrieb</u>: Geradeaus- & Kurvenfahrten, Drehen auf Stelle, Vorwärts- und Rückwärtsfahren identisch → Einfache Mechanik vs. Radregelung in Echtzeit</p> <p><u>Dreirad-Antrieb</u>: Geradeaus- & Kurvenfahrten, Vorwärts- & Rückwärts unterschiedlich (unterschiedliches α) → Einfache Mechanik vs. Eingeschränkte Manövrierfähigkeit</p> <p><u>Synchro-Antrieb</u>: Geradeaus- & Kurvenfahrten, Vorwärts- und Rückwärts identisch, Plattform dreht nicht → einfache Regelung, geradeaus garantiert vs. mechan. komplex</p> <p><u>Mecanum-Antrieb</u>: uneingeschränkte Beweglichkeit in Richtung x, y und w vs. mechanisch komplex, aufwendige Regelung</p>								
Sensoren	<p><u>Informationsfluss</u></p>  <p><u>Klassifizierung</u></p> <table border="1" data-bbox="359 795 1492 1220"> <thead> <tr> <th>Interne Sensoren</th><th>Externe Sensoren</th></tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> - kein Kontakt zur Umwelt - Bestimmung von Lage und Position durch Neigung, Orientierung, Drehrichtung, Beschleunigung, Lenkwinkel <p><i>Aufgabe</i>: Stellung der Gelenke, Geschwindigkeit der Bewegung der Gelenke, Kräfte und Momente auf Gelenke</p> </td><td> <ul style="list-style-type: none"> - Information aus der Umwelt - Bestimmung von Position und Orientierung und Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos <p><i>Aufgabe</i>: Entfernung, Lage von Positionsmarken und Objekten, Kontur von Objekten, Pixelbilder der Umwelt</p> </td></tr> <tr> <th>Aktive Sensoren</th><th>Passive Sensoren</th></tr> <tr> <td>Simulation der Umwelt durch Eintrag von Energie, Messen & Auswerten der Antwort</td><td>Umwelt vorhandene Signale werden gemessen und ausgewertet</td></tr> </tbody> </table>	Interne Sensoren	Externe Sensoren	<ul style="list-style-type: none"> - kein Kontakt zur Umwelt - Bestimmung von Lage und Position durch Neigung, Orientierung, Drehrichtung, Beschleunigung, Lenkwinkel <p><i>Aufgabe</i>: Stellung der Gelenke, Geschwindigkeit der Bewegung der Gelenke, Kräfte und Momente auf Gelenke</p>	<ul style="list-style-type: none"> - Information aus der Umwelt - Bestimmung von Position und Orientierung und Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos <p><i>Aufgabe</i>: Entfernung, Lage von Positionsmarken und Objekten, Kontur von Objekten, Pixelbilder der Umwelt</p>	Aktive Sensoren	Passive Sensoren	Simulation der Umwelt durch Eintrag von Energie, Messen & Auswerten der Antwort	Umwelt vorhandene Signale werden gemessen und ausgewertet
Interne Sensoren	Externe Sensoren								
<ul style="list-style-type: none"> - kein Kontakt zur Umwelt - Bestimmung von Lage und Position durch Neigung, Orientierung, Drehrichtung, Beschleunigung, Lenkwinkel <p><i>Aufgabe</i>: Stellung der Gelenke, Geschwindigkeit der Bewegung der Gelenke, Kräfte und Momente auf Gelenke</p>	<ul style="list-style-type: none"> - Information aus der Umwelt - Bestimmung von Position und Orientierung und Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos <p><i>Aufgabe</i>: Entfernung, Lage von Positionsmarken und Objekten, Kontur von Objekten, Pixelbilder der Umwelt</p>								
Aktive Sensoren	Passive Sensoren								
Simulation der Umwelt durch Eintrag von Energie, Messen & Auswerten der Antwort	Umwelt vorhandene Signale werden gemessen und ausgewertet								
Kinematisches Modell	<p>Das kinematische Modell eines Roboters beschreibt die Zusammenhänge zwischen dem Raum der Gelenkwinkel und dem Raum der Lage des Endeffektors in Weltkoordinaten.</p>								
Vorwärtskinematik	<p><u>Direktes kinematische Problem</u>: Bestimmung der Lage des Endeffektors aus den Gelenkwinkelstellungen des Roboters. (Wo ist meine Hand? Hier!) $f: \mathbb{R}^n \rightarrow \mathbb{R}^m, x = f(\theta)$</p>								
Rückwärtskinematik	<p><u>Inverses kinematisches Problem</u>: Bestimmung der Gelenkwinkelstellungen zu einer gewünschten Lage des Endeffektors. (Wie erreiche ich den Becher?) $F: \mathbb{R}^m \rightarrow \mathbb{R}^n, \theta = F(x)$</p>								
Denavit-Hartenberg (DH) Konvention	 <p><u>z_{i-1}-Achse</u> liegt entlang der Bewegungsachse des i-ten Gelenks</p> <p><u>x_i-Achse</u> verläuft entlang der gemeinsamen Normalen von z_{i-1} & z_i & zeigt weg von z_{i-1}</p> <p><u>Die y_i-Achse</u> vervollständigt KS entsprechend der Rechte-Hand-Regel</p> <p><u>Armement i</u> zwischen Gelenk i & $i+1$</p> <p>z_i liegt entlang Gelenkachse $i + 1$</p> <p><u>Armementlänge a_i</u> liegt beschreibt den Abstand von z_{i-1} zu z_i</p> <p><u>Armementverwindung α_i</u> beschreibt den Winkel zwischen z_{i-1} zu z_i um x_i</p> <p><u>Gelenkabstand d_i</u> ist der Abstand zwischen x_{i-1}-Achse und x_i Achse entlang z_{i-1}-Achse</p> <p><u>Gelenkwinkel θ_i</u> ist der Winkel von x_{i-1} zu x_i um z_{i-1}</p>								

DH Transformationsmatrix OKS_{i-1} zu OKS_i	$A_{i-1,i} = R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) = \begin{pmatrix} \cos \theta_i & -\sin \theta_i \cdot \cos \alpha_i & \sin \theta_i \cdot \sin \alpha_i & a_i \cdot \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cdot \cos \alpha_i & -\cos \theta_i \cdot \sin \alpha_i & a_i \cdot \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$ $A_{i,i-1} = \begin{matrix} & -a_i \\ & -d_i \sin \alpha_i \\ & -d_i \cos \alpha_i \\ 0 & 0 & 0 & 1 \end{matrix} \quad \text{Transponierte Matrix}$	
Endeffektor-Geschwindigkeit	$\dot{x}(t) = J_f(\theta(t)) \cdot \dot{\theta}(t)$	
Endeffektor Kraft	$\tau(t) = J_f^T(\theta(t)) \cdot F(t)$	
Berechnung der Jacobi Matrix	$J_f = \left(\frac{\partial f}{\partial \theta_1} \dots \frac{\partial f}{\partial \theta_n} \right) \rightarrow \text{Translationsgel.: } \frac{\partial f(\theta)}{\partial \theta_j} = \begin{bmatrix} v_j \\ 0 \end{bmatrix} \quad \text{Rotationsgel.: } \frac{\partial f(\theta)}{\partial \theta_j} = \begin{bmatrix} v_j \times (f(\theta) - p_j) \\ v_j \end{bmatrix}$	
Singularitäten	Eine kinematische Kette ist in einer singulären Konfiguration wenn die zugehörige Jacobi Matrix nicht vollen Rang hat (zwei oder mehr Spalten in J_f linear abhängig). Dadurch dass die Jacobi Matrix dann nicht invertierbar ist sind bestimmte Konfigurationen nicht möglich. Für quadratische Matrizen gilt dann, dass die Determinanten der Jacobi Matrix gleich null ist.	
Erreichbarkeit	Es existiert mindestens eine Gelenkwinkelkonfiguration, so dass der TCP in Gitterzelle liegt.	
Manipulierbarkeit	Maximaler Manipulierbarkeitswert einer Gitterzelle	
Geometrische Modelle	<u>Verwendung:</u> Kollisions- und Kontaktberechnung, Simulation, Darstellung, Kräfte & Moment <u>Arten:</u> Einhüllende Quader (Kollision), Polygonzüge (Visualisierung), Volumenmodell (exakt)	
Inverse Kinematik Geomet. Methode	Anwendung von trigonometrischen Funktionen & Sinus- und Kosinussätze Kosinussatz: $c^2 = a^2 + b^2 - 2ab \cdot \cos \gamma$ wobei γ der Winkel zwischen a und b ist	
Inverse Kinematik Algebraische Methode	Gleichsetzen von P_{TCP} und der Transformation ${}^{BKS}T_{TCP}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \rightarrow 16$ (12 nicht triviale) Gleichungen die gelöst werden müssen 1. Invertiere $A_{0,1}(\theta_1)$ und multipliziere beide Seite mit $A_{0,1}^{-1}$ 2. Versuche aus dem neuen Gleichungssysteme, eine Gleichung zu finden mit nur einer Unbekannten und löse diese Gleichung nach der unbekannten. 3. Versuche Gleichung zu finden die durch Substitution der im letzten Schritt gefundenen Lösung nach einer Unbekannten lösbar ist 4. Falls keine Lösung mehr gefunden wird, muss eine weitere Matrix invertiert werden. 5. Wiederhole 1. bis 4. bis alle Gelenkwinkel ermittelt wurden	
Inverse Kinematik Numerische Methode	<u>Pseudoinverse</u> 1. Vorwärtskinematik: $x(t) = f(\theta(t))$ 2. Ableitung nach der Zeit: $\frac{\partial x(t)}{\partial t} = \dot{x}(t) = J_f(\theta)\dot{\theta}(t)$ 3. Übergang zum Differenzenquotient $\Delta x \approx J_f(\theta)\Delta\theta$ 4. Umkehrung: $\Delta\theta \approx J_f^\#(\theta)\Delta x$ Pseudoinverse: $A^\# = A^T(AA^T)^{-1}$ (nahe Singularitäten instabil) <u>Damped Least Squares</u> Minimiere: $\min_{\Delta\theta} \left\ J_f(\theta)\Delta\theta - \Delta x \right\ _2^2 + \lambda^2 \left\ \Delta\theta \right\ _2^2 \rightarrow \Delta\theta = J^T(JJ^T + \lambda^2 E)^{-1} \Delta x$ wobei Dämpfungskonstante λ anwendungsspezifisch gewählt werden muss (groß genug für Stabilität in Umgebung von Singularitäten aber klein genug für schnelle Konvergenz)	
Inverse Kinematik Bewertung	Allgemeine Verfahren	Spezielle Verfahren
	Numerische Verfahren	Trigonometrische graphische Verfahren
	Allg. Lösungsverfahren für Gleichungssysteme	
	→ hoher Aufwand	→ schnell
	→ lange Zeitdauer	→ nur für spezielle Robotertypen
Dynamisches Modell	Das dynamische Modell beschreibt den Zusammenhang von Kräften, Momenten und Bewegungen, welche in einem mechanischen Mehrkörpersystem auftreten. <u>Bewegungsgleichung</u>	

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

τ : $n \times 1$ Vektor der generalisierten Kräfte
 $M(q)$: $n \times n$ Massenträgheitsmatrix
 $c(\dot{q}, q)$: $n \times 1$ Vektor der Zentripetal- und Corioliskomponenten
 $g(q)$: $n \times 1$ Vektor der Gravitationskomponenten
 q, \dot{q}, \ddot{q} : $n \times 1$ Vektor der generalisierten Koordinaten
 (Position, Geschwindigkeit, Beschleunigung)

Direkts dynamisches Problem (Anfangswertproblem der Mechanik): Aus äußeren Kräften und Momenten sowie Anfangszustand wird unter Verwendung des dynamischen Modells die sich ergebenden Bewegungsänderungen berechnet.

Inverses dynamisches Problem: Aus den gewünschten Bewegungsparametern sollen, unter Verwendung des dynamischen Modells, die dazu erforderlichen Stellkräfte und -momente ermittelt werden.

Methode nach Lagrange

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

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

Algorithmus

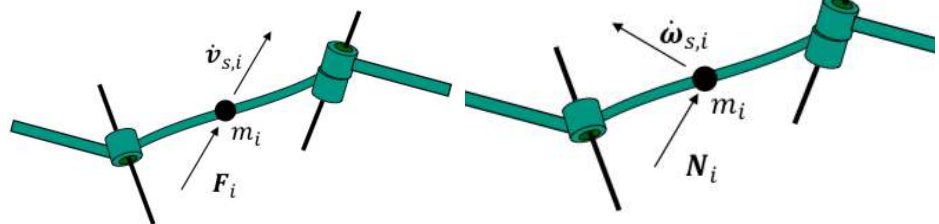
1. Berechne E_{kin} und E_{pot}
2. Drücke E_{kin} und E_{pot} in generalisierten Koordinaten aus $L(q, \dot{q}) = E_{kin}(q, \dot{q}) - E_{pot}(q)$
3. Berechne die Ableitung

Methode nach Newton-Euler

Betrachtung der Massenschwerpunkte eines Armelements.

Kraft = Impuls abgeleitet nach der Zeit $F_i = \frac{d}{dt} (m_i v_{s,i}) = m_i \cdot \dot{v}_{s,i}$

Drehmoment = Drehimpuls abgeleitet nach Zeit $N_i = \frac{d}{dt} (I_i \omega_{s,i}) = I_i \dot{\omega}_{s,i}$



→ Beschleunigungen von der Basis/Greifer zum/zur Greifer/Basis berechnen

Vorwärtsgleichung (von Basis zum Greifer)

Startwerte: $\omega_0, \dot{\omega}_0, v_0, \dot{v}_0$ Eingabe: $q(t), \dot{q}(t), \ddot{q}(t)$

$\omega_{i+1} = G_1(\omega_i, q_{i+1}, \dot{q}_{i+1})$ $\dot{\omega}_{i+1} = G_2(\omega_i, \dot{\omega}_i, q_{i+1}, \dot{q}_{i+1}, \ddot{q}_{i+1})$

$v_{i+1} = G_3(v_i, \omega_i, q_{i+1})$ $\dot{v}_{i+1} = G_4(v_i, \omega_i, \dot{\omega}_{i+1}, q_{i+1})$

$v_{s,i+1} = G_5(v_{i+1}, \omega_{i+1})$ $\dot{v}_{s,i+1} = G_3(\dot{v}_{i+1}, \omega_{i+1}, \dot{\omega}_{i+1})$

Rückwärtsgleichung (Greifer zur Basis)

Startwerte: $F_n = F_{ex}$ und $N_n = N_{ex}$

$F_i = H_1(\dot{v}_{s,i})$

$N_i = H_2(\omega_i, \dot{\omega}_i, q_i, \dot{q}_i)$

$f_i = H_3(f_{i+1}, F_i, q_{i+1})$

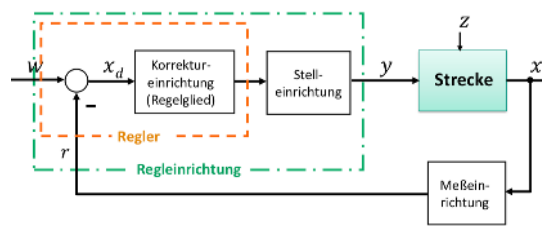
$n_i = H_4(n_{i+1}, f_{i+1}, F_i, N_i, q_{i+1})$

$\tau_i = H_5(n_i, q_i)$

Eigenschaften: Beliebige Anzahl von Gelenken, Belastungen der Armelemente werden berechnet, Aufwand $O(n)$, Rekursion

Modellierung der Dynamik

Definition: Lehre von der selbsttätigen, gezielten Beeinflussung dynamischer Prozesse



während des Prozessablaufs. Der Ausgangsgröße eines dynamischen Systems soll mittels der Stellgröße ein Sollverhalten, d.h. ein gewünschtes Verhalten ausgeprägt werden, & zwar gegen den Einfluss einer Störgröße, die nur unvollständig bekannt ist.

w: Führungsgröße (Soll-Wert)

y: Stellgröße

r: Rückführgröße (Ist-Wert)

x_d : Regeldifferenz

x: Regelgröße

z: Störgröße

Regelungstechnische Grundsituation: Forderung nach selbsttätiger, gezielter Beeinflussung bei unvollständiger Systemkenntnis, insbesondere bei Einwirkung von Störungen.

Regelung: Unter einer Regelung versteht man eine Anordnung, durch welche bei unvollständig bekannter Strecke, insbesondere unvollständiger Kenntnis der Störgröße, die Regelgröße, d.h. die Ausgangsgröße der Strecke, laufend erfasst und mit der Führungsgröße verglichen wird, um mittels der so gebildeten Differenz die Regelgröße an den Sollverlauf anzugleichen.

Regelkreis: Aus physikalischen Gesetzen ermittelt man Gleichungen zwischen zeitveränderlichen Größen des Systems.

Übertragungsglied: Ein Block ordnet jedem Zeitverlauf der Eingangsgröße eindeutig einen Zeitverlauf der Ausgangsgröße zu.

Laplace-Transformation:

$$\mathcal{L}\{f(t)\} = f(s) = \int_0^\infty f(t)e^{-st}dt \text{ und } s := \sigma + j\omega; f(t) = 0, t < 0$$

⇒ Differential- & Integralausdrücke durch algebraische Ausdrücke ersetzen

⇒ Gleichungslösung im Frequenzbereich statt im Zeitbereich

$$\text{Ableitungsfunkt.: } \mathcal{L}\{\dot{f}(t)\} = s \int_0^\infty e^{-st}f(t)dt - f(0) = s \cdot f(s) - f(0) \quad (\lim_{t \rightarrow \infty} e^{-st}f(t) \rightarrow 0)$$

$$\text{Integral einer Funktion: } \mathcal{L}\left[\int_0^t f(\tau)d\tau\right] = \frac{1}{s}f(s)$$

Impulsfunktion (Dirac-Impuls)

$$\delta(t) = \begin{cases} \infty & , \text{ für } t = 0 \\ 0 & , \text{ für } t \neq 0 \end{cases}$$

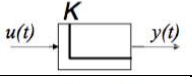
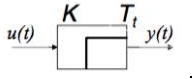
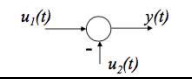
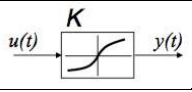
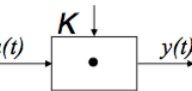
$$\text{Laplace Transformation: } \mathcal{L}[\delta(t)] = \int_0^\infty \delta(t) \cdot e^{-st}dt = 1$$

Einheitssprungfunktion

$$\sigma(t) = \begin{cases} 0 & , \text{ für } t < 0 \\ 1 & , \text{ für } t \geq 0 \end{cases}$$

$$\text{Laplace Transformation: } \mathcal{L}[\sigma(t)] = \frac{1}{s}$$

Lineares zeitinvariantes Übertragungsglied (LZI)	$Y(s) = G(s) \cdot U(s) \Leftrightarrow y(t) = g(t) * u(t) = \int_0^t g(t-\tau) * u(\tau) * d\tau$	
P-Glied (Proportionalglied)	$y(t) = K \cdot u(t)$ $Y(s) = K \cdot U(s)$	
I-Glied (Integrierglied)	$y(t) = K \cdot \int_0^t u(\tau)d\tau$ $Y(s) = K \cdot \frac{1}{s} \cdot U(s)$	

D-Glied (Differenzierglied)	$y(t) = K \cdot \dot{u}(t)$ $Y(s) = K \cdot [sU(s) - u(0)]$	
T _t -Glied (Totzeit-Glied)	$y(t) = K \cdot u(t - T_t)$ $Y(s) = K \cdot e^{-T_t s} \cdot U(s)$	
S-Glied (Summenglied)	$y(t) = \pm u_1(t) \pm u_2(t)$ $Y(s) = \pm U_1(s) \pm U_2(s)$	
KL-Glied (Kennlinienglied)	$y(t) = K \cdot F(u(t))$	
M-Glied (Multiplizierglied)	$y(t) = K \cdot u_1(t)u_2(t)$ $Y(s) = K \cdot U_1(s) * U_2(s)$	

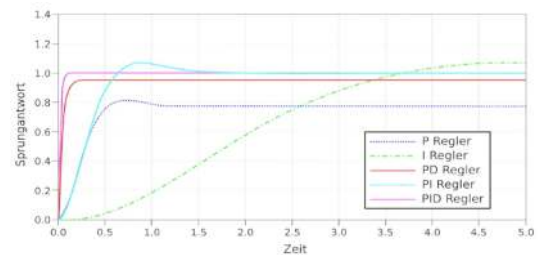
Proportional-Integral-Derivative Controller (PID Regelung)

$$\tau = \underbrace{K_p \theta_d}_{\text{P}} + \underbrace{K_i \int \theta_d(t) dt}_{\text{I}} + \underbrace{K_d \dot{\theta}_d}_{\text{D}}$$

K_p : virtuelle Feder die Positionsfehler senkt

K_d : virtueller Dämpfer der den Geschwindigkeitsfehler reduziert

K_i : reduziert Regelabweichung



Stabilität: Regelabweichung geht gegen 0 mit einem PID Regler

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

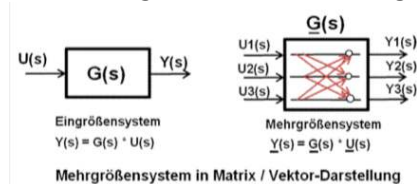
→ Aperiodische Lösung: Zielwert ohne schwingen langsam über Exponentialfunktion erreicht

→ Aperiodischer Grenzfall: Zielwert schnell erreicht und System überschwingt gar nicht

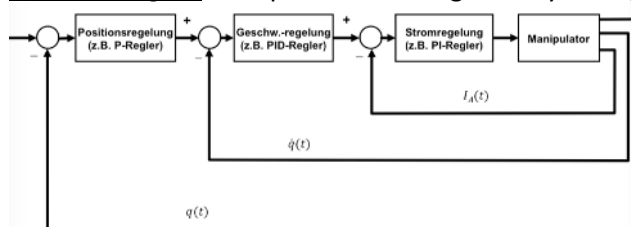
→ Gedämpfte Schwingung: Das System überschwingt

Testfunktionen: Impulsfunktion, Sprungfunktion, Anstiegsfunktion, Harmonische Funktion

Zustandsregler: Verbessertes Regelverhalten (Mehrgrößensysteme)



Kaskadenregler: Manipulator = Mehrgrößensystem (unabhängige Einzelregelkreise Gelenke)



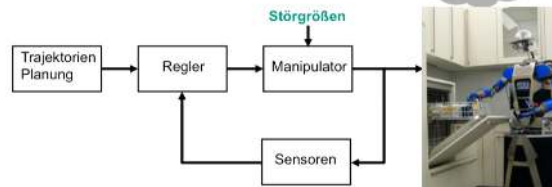
Adaptive Regler: Lageabhängige & zeitveränderliche Systemteile als Parameterschwankungen



Regler

**Regelungs-
konzepte für
Manipulatoren**

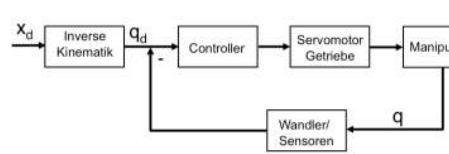
Blockbild einer Manipulator-Regelung



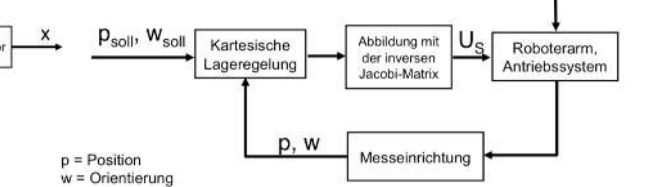
- Q : $n \times 1$ Vektor der allgemeinen Stellkräfte und -momente
- $M(q)$: $n \times n$ Trägheitsmatrix
- n : $n \times 1$ Vektor mit Zentrifugal- und Corioliskomponenten
- $g(q)$: $n \times 1$ Vektor mit Gravitationskomponenten
- R : $n \times n$ Diagonalmatrix zur Beschreibung der Reibungskräfte
- q : $n \times 1$ Winkellagen des Manipulators

Gravitations, Zentrifugal, Coriolis und Reibungskräfte: $Q = M(q)\ddot{q} + n(\dot{q}, q) + g(q) + R\dot{q}$

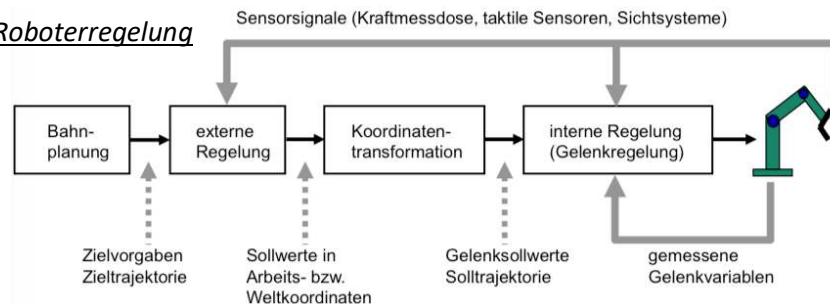
Regelung im Gelenkwinkelraum



Regelung im kartesischen Raum



Roboterregelung



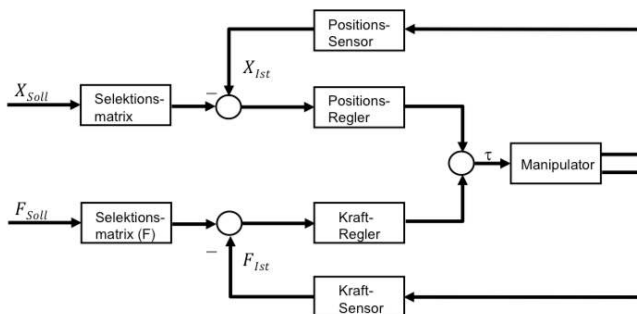
Exakte Systemmodellierung: Setzt a priori die exakte Kenntnis des Dynamikmodells und der Umgebung des Roboters voraus

Kraft/Positionsregelung: Ausführung von Aufgaben, die Interaktionskräfte berücksichtigen

Problem: Position & Kräfte eng verbunden → Roboter in Kontakt mit Umgebung →

Positionsändern auch eine Kraftänderung und umgekehrt

Lösung: Aus Beschreibung der Aufgabe resultieren natürliche Randbedingungen



Programmierung eines Reglers: Schnell, Robust, Inuitiv, Einfache Wartung, Wiederverwendbar

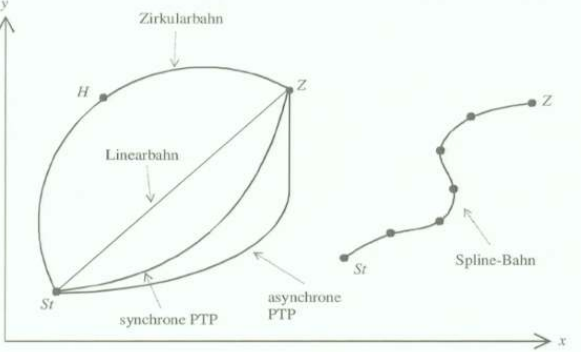
Impedanz-Regelung: Dyna. Beziehung zw. Kraft und Position $f(t) = d \cdot x(t) + b \cdot x'(t) + m \cdot x''(t)$

Bewegung des Roboters: Zustandsänderung über die Zeit relativ zu einem festen Koordinatensystem unter Einschränkung durch Zwangsbedingungen, Gütekriterien sowie Neben- und Randbedingungen

Steuerung: asynchro (Achsen unabhängig) vs. synchron (achseninterpoliert über Leitachse)

Bahnsteuerung

Kartesischer Raum	Gelenkwinkelraum
+ Bahn einfacher formulieren	+ Ansteuerung der Gelenke ist einfacher
+ Interpolation einfacher	+ Trajektorie eindeutig/berücksichtigt Gelenkwinkelgrenzen
- Inverse Kinematik für alle Trajektoriepunkte lösen	- Interpolation für mehrere Gelenke
- Trajektorie nicht immer ausführbar	- Formulierung der Trajektorie umständlicher

Programmieren durch Schlüsselpunkte	<p><u>Teach-In:</u> Anfahren markanter Punkte der Bahn mit manueller Steuerung. Generierung des Programms durch Definition von Stützpunkten.</p> <p><u>Playback:</u> Roboter kann durch Bediener bewegt werden (Zero Force Control Mode) und einzelne Schritte gespeichert werden.</p> <table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Schnell für komplexe Bahnen</td><td>Schwere Roboter schwer zu bewegen</td></tr> <tr> <td>Intuitiv</td><td>Wenig Platz in engen Fertigungszellen für Bediener</td></tr> <tr> <td></td><td>Schlechte Korrekturmöglichkeiten</td></tr> <tr> <td></td><td>Optimierung & Kontrolle durch Interpolation schwer</td></tr> </tbody> </table>	Pro	Contra	Schnell für komplexe Bahnen	Schwere Roboter schwer zu bewegen	Intuitiv	Wenig Platz in engen Fertigungszellen für Bediener		Schlechte Korrekturmöglichkeiten		Optimierung & Kontrolle durch Interpolation schwer
Pro	Contra										
Schnell für komplexe Bahnen	Schwere Roboter schwer zu bewegen										
Intuitiv	Wenig Platz in engen Fertigungszellen für Bediener										
	Schlechte Korrekturmöglichkeiten										
	Optimierung & Kontrolle durch Interpolation schwer										
Interpolationsarten	 <p><u>Punkt zu Punkt Bewegung</u> Vorteil: Berechnung der Gelenkwinkeltrajektorie einfach und keine Singularitäten</p> <p><u>PTP mit Rampenprofil</u> Direkter Anstieg mit konstanter Steigung und genau so auch bei Abbremsen.</p> <p><u>PTP mit Sinoidenprofil</u></p> <ul style="list-style-type: none"> - Weichere Bewegung durch Verwendung einer sinuiden Zeitfunktion - Roboter wird weniger beansprucht <p><u>Synchrone PTP-Bahnen</u></p> <ul style="list-style-type: none"> - Alle Gelenke beginnen und beenden ihre Bewegung gemeinsame <p><u>Asynchrone PTP-Bahnen</u></p> <ul style="list-style-type: none"> - Jedes Gelenk wird sofort mit der maximalen Beschleunigung angesteuert - Jede Gelenkbewegung endet unabhängig von den anderen <p><u>Vollsynchrone PTP Bahn</u></p> <ul style="list-style-type: none"> - Zusätzlich Beschleunigungs- und Bremszeit mit beachten <p><u>Linear Interpolation (CP Interpolation)</u> Die Robotersteuerung interpoliert die Bahn zwischen je zwei Trajektorien.</p> <p><u>Zirkularinterpolation:</u> Bewegung des Endeffektors findet auf einem Kreis statt</p> <p><u>Segmentweise Bahninterpolation:</u> kubische Splines zwischen Stützpunkten</p>										
Approximierte Bahnsteuerung	<p><u>Bahninterpolation:</u> Die ausgeführte Bahn verläuft durch alle Stützpunkte der Trajektorie</p> <p><u>Bahnapproximation:</u> Kontrollpunkte beeinflussen den Bahnverlauf und werden approximiert</p> <p><u>Bézierkurven:</u> Stützpunkte werden nicht durchlaufen sonder beeinflussen nur die Bahn</p> $P(t) = \sum_{i=0}^n B_{i,n}(t) P_i \quad 0 \leq t \leq 1 \quad B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$ <p><u>De-Casteljau-Algorithmus:</u> Effiziente Näherungsdarstellung von Bézierkurven mit Polygonzug</p> <p>Gegeben: n Kontrollpunkte P_0, \dots, P_{n-1}</p> <p>Start: $P_i^0 = P_i$</p> <p>Iteration k: $P_i^{k+1} = (1-t_0)P_i^k + t_0P_{i+1}^k$</p>										

Bewegungs- planung	<p>Erzeugen einer kollisionsfreien Trajektorie unter Berücksichtigung verschiedener Ziele und Einschränkungen.</p> <p><u>Problemstellung</u> Gegeben: Konfigurationsraum C, Startkonfiguration q_{Start} und Zielkonfiguration q_{Ziel} Gesucht: Stetige Trajektorie $\tau: [0,1] \rightarrow C$ mit $\tau(0) = q_{\text{Start}}$ und $\tau(1) = q_{\text{Ziel}}$ (Beachtung Gütekriterien, Neben- & Randbedingungen, Zwangsbedingungen)</p> <p><u>Konfiguration</u>: Eine Konfiguration q aus C beschreibt den Zustand eines Roboters (Lage & Orientierung im euklidischen Raum oder Gelenkwinkelvektoren)</p> <p><u>Konfigurationsraum C</u>: Raum aller möglichen Konfigurationen eines Roboters R.</p> <p><u>Arbeitsraumhindernis H</u>: Raum der von Objekten im Arbeitsraum eingenommen wird.</p> <p><u>Konfigurationsraumhindernis</u>: Menge aller Punkte des Konfigurationsraums C, welche zu einer Kollision mit dem Hindernis H führen.</p> <p><u>Hindernisraum</u>: C_{obs} ist die Menge aller Konfigurationsraumhindernisse</p> <p><u>Freiraum</u>: Menge aller Punkte aus C, welche nicht im Hindernisraum C_{obs} liegen $C_{\text{free}} = \{q \in C \mid q \notin C_{\text{obs}}\} = C \setminus C_{\text{obs}}$ (Zur Bestimmung werden approximative Verfahren verwendet)</p> <p><u>Umweltmodellierung</u>: Exakt bspw. durch constructed solid geometry in Form von algebraischen Beschreibungen oder approximiert durch Näherung (Boxen, ..)</p> <p><u>Randbedingungen</u>: Globale Randbedingungen (Limitieren Konfigurationsraum), lokale Randbedingungen schränken Übergang zwischen Konfigurationen ein</p> <p><u>Komplexität</u>: Allgemeine Planungsaufgaben sind PSAPCE complete.</p> <p><u>Vollständiger Algorithmus</u>: Findet mindestens eine Lösung oder erkennt nach endlicher Zeit, dass keine Lösung existiert.</p> <p><u>Randomisiert Algorithmus</u>: Verwenden Zufallsgrößen um den Ablauf zu steuern, wobei oft heuristische Annahmen genutzt werden, um die Berechnung zu beschleunigen.</p> <p><u>Auflösungsvollständiger Algorithmus</u>: Approximativer Algorithmus der für eine diskrete Problemstellung vollständig ist, wird er auflösungsvollständig genannt.</p> <p><u>Probabilistischer-vollständiger Algorithmus</u>: Findet mindestens eine Lösung falls sie existiert, d.h. Wahrscheinlichkeit, dass eine Lösung gefunden wird, konvergiert mit fortlaufender Zeit gegen eins (kann nicht ermitteln ob keine Lösung existiert).</p>											
Konstruktion von Wegenetzen	<p><u>Voronoi-Diagramme</u> Zerlegung eines Raum in Regionen basierend auf vorgegebenen Punkten. Region ist definiert als Menge aller Punkte deren Abstand zum Zentrum geringer ist als zu allen anderen Zentren. Alle Punkte auf der Grenzen zwischen zwei Regionen haben den gleichen Abstand zum eigenen und benachbarten Zentrum.</p> <ul style="list-style-type: none"> - Teile Punktemenge P in zwei etwa gleich große Teilmengen P_1 und P_2 - Rekursive Unterteilung von zwei bzw. drei Punkten: - Verschmelze Diagramme durch verbinden der nächsten Nachbarn entlang Trennungslinie <table border="1" data-bbox="354 1624 1489 1771"> <thead> <tr> <th>Vorteile</th><th>Nachteile</th></tr> </thead> <tbody> <tr> <td>Maximale Abstand zu den Hindernissen</td><td>In der Regel ist der Weg nicht der kürzeste</td></tr> <tr> <td>Mit Abstandssensoren leicht prüfbar ob der richtige Weg abgefahren wurde</td><td>Bei wenig Hindernissen werden nur wenige Wege generiert</td></tr> </tbody> </table> <p><u>Sichtgraphen</u> Verbinde jedes Paar von Eckpunkten auf dem Rand von C_{free} durch ein gerade Liniensegment, wenn das Segment kein Hindernis schneidet. Verbinden dann Start und Ziel über diese Linien.</p> <table border="1" data-bbox="354 1906 1489 2085"> <thead> <tr> <th>Vorteile</th><th>Nachteile</th></tr> </thead> <tbody> <tr> <td>Methode exakt wenn nur 2 translatorische Freiheitsgrade existieren & alles als konvexe Polygone dargestellt werden kann</td><td rowspan="2">Nicht zwingend kollisionsfrei, da Hinderniskanten auch Wegesegmente sein können <i>Mögliche Problemlösung</i>: Hindernisse um Roboterform erweitern.</td></tr> <tr> <td>Wenn Weg gefunden dann der kürzeste</td></tr> </tbody> </table>	Vorteile	Nachteile	Maximale Abstand zu den Hindernissen	In der Regel ist der Weg nicht der kürzeste	Mit Abstandssensoren leicht prüfbar ob der richtige Weg abgefahren wurde	Bei wenig Hindernissen werden nur wenige Wege generiert	Vorteile	Nachteile	Methode exakt wenn nur 2 translatorische Freiheitsgrade existieren & alles als konvexe Polygone dargestellt werden kann	Nicht zwingend kollisionsfrei, da Hinderniskanten auch Wegesegmente sein können <i>Mögliche Problemlösung</i> : Hindernisse um Roboterform erweitern.	Wenn Weg gefunden dann der kürzeste
Vorteile	Nachteile											
Maximale Abstand zu den Hindernissen	In der Regel ist der Weg nicht der kürzeste											
Mit Abstandssensoren leicht prüfbar ob der richtige Weg abgefahren wurde	Bei wenig Hindernissen werden nur wenige Wege generiert											
Vorteile	Nachteile											
Methode exakt wenn nur 2 translatorische Freiheitsgrade existieren & alles als konvexe Polygone dargestellt werden kann	Nicht zwingend kollisionsfrei, da Hinderniskanten auch Wegesegmente sein können <i>Mögliche Problemlösung</i> : Hindernisse um Roboterform erweitern.											
Wenn Weg gefunden dann der kürzeste												

Konstruktion von Wegenetzen	<p><u>Zellzerlegung</u></p> <ol style="list-style-type: none"> 1. Zerlege C_{free} in Zellen, so dass ein Weg zwischen zwei Konfigurationen innerhalb einer Zelle leicht zu finden ist 2. Stelle die Nachbarschaft (Adjazenz) in einem Graphen dar 3. Suche den optimalen Weg von q_{start} nach q_{ziel} in dem Graphen <p><i>Exakte Zerlegung</i>(Line Sweep) Zellen überlappen nicht & Vereinigungsmenge ist C_{free}</p> <p><i>Approximative Zerlegung:</i></p> <ol style="list-style-type: none"> 1. Zerlege den Freiraum C_{free} in Zellen vordefinierter Form (z.B. rechteckig) 2. Wenn Zelle nicht vollständig in C_{free} liegt, verringere Größe & zerlege die Zelle weiter 3. Wende diesen Schritt bis zu Minimalgröße der Zellen an <table border="1" data-bbox="352 645 1485 719"> <thead> <tr> <th>Vorteile</th><th>Nachteile</th></tr> </thead> <tbody> <tr> <td>Einfache Zerlegung → einfache Wegsuche</td><td>Freiraum i.A. nur annähernd beschrieben</td></tr> </tbody> </table>	Vorteile	Nachteile	Einfache Zerlegung → einfache Wegsuche	Freiraum i.A. nur annähernd beschrieben		
Vorteile	Nachteile						
Einfache Zerlegung → einfache Wegsuche	Freiraum i.A. nur annähernd beschrieben						
Suche im Wegenetz	<p><u>Baumsuche</u></p> <ul style="list-style-type: none"> - Konfigurationsraum wird als Quadtree dargestellt (rekursive Unterteilung in Kacheln) - Kacheln sind entweder frei oder Hindernisse - Freie Kacheln vom Start zum Ziel verbinden (Kollisionsfrei durch freie Kacheln) <p><u>A*-Algorithmus</u></p> <p>Algorithmus zur Routenplanung für den kürzesten/besten Pfad von Start zu Ziel mit Optimalität in Bezug auf Pfadkosten. (Bestensuche ⇔ findet optimalen Pfad)</p> <p>Kostenfunktion: $f(x) = g(x) + h(x)$ ($g(x)$ Kosten von Start zu Knoten x) ($h(x)$ geschätzte Kosten von Knoten x nach Zielknoten)</p> <p>Open Set O: noch zu besuchende Knoten Closed Set C: Bereits besuchte Knoten</p> <p>Solange $O \neq \emptyset$</p> <p style="padding-left: 20px;">Bestimme den zu erweiternden Knoten Finde $v_i \in O$ mit minimalem $f(v_i) = g(v_i) + h(v_i)$ Wenn $v_i = v_{ziel}$</p> <p style="padding-left: 20px;">Lösung gefunden: Traversiere Vorgänger von v_i bis v_{start} erreicht ist O.remove(v_i) C.add(v_i)</p> <p style="padding-left: 20px;">Update für alle Nachfolger v_j von v_i durchführen</p> <ul style="list-style-type: none"> • Wenn $v_j \in C$, dann überspringe v_j • Wenn $v_j \notin O$, dann O.add(v_j) • Wenn $g(v_i) + cost(v_i, v_j) < g(v_j)$ <ul style="list-style-type: none"> • $g(v_j) = g(v_i) + cost(v_i, v_j)$ • $h(v_j) = heuristic(v_j, v_{Ziel})$ • $pred(v_j) = v_i$ <table border="1" data-bbox="352 1693 1485 1809"> <thead> <tr> <th>Vorteile</th><th>Nachteile</th></tr> </thead> <tbody> <tr> <td>Findet optimale Lösung wenn Heuristik zulässig (d.h. Kosten nicht überschätzt)</td><td></td></tr> <tr> <td>Optimal effizient für jede zulässige Heuristik h</td><td></td></tr> </tbody> </table>	Vorteile	Nachteile	Findet optimale Lösung wenn Heuristik zulässig (d.h. Kosten nicht überschätzt)		Optimal effizient für jede zulässige Heuristik h	
Vorteile	Nachteile						
Findet optimale Lösung wenn Heuristik zulässig (d.h. Kosten nicht überschätzt)							
Optimal effizient für jede zulässige Heuristik h							
Potentialfelder	<p><u>Definition:</u> Ein Potentialfeld U ist eine Skalarfunktion über dem Freiraum. Die Kraft in einem Punkt q des Potentialfeldes ist der negative Gradient im Punkt: $F(q) = -\nabla U(q)$</p> <p><i>Abstoßendes Potential:</i> Hindernisse erzeugen ein abstoßendes Potential. In großen Abständen zu Hindernissen soll der Roboter nicht beeinflusst werden.</p> $U_{ab}(q) = \frac{1}{2} v \left(\frac{1}{p(q, q_{obs})} - \frac{1}{p_0} \right)^2 \text{ für } p(q, q_{obs}) \leq p_0 \text{ \& 0 sonst mit } p(q, q_{obs}) \leq p_0$						

<p>Potentialfelder</p>	<p><i>Anziehendes Potential:</i> Es soll möglichst nur ein Minimum q_{Ziel} geben.</p> <p><i>Distanz zum Ziel (linear):</i> $U_{an}(q) = k \cdot q - q_{Ziel}$</p> <p><i>Distanz zum Ziel (quadratisch):</i> $U_{an}(q) = k \cdot \frac{1}{2} q - q_{Ziel} ^2$</p> <p><i>Zielstellung:</i> q_{Ziel} anziehendes Potent U_{an} <i>Hindernisraum:</i> C_{obs} abstoßendes Potent. U_{ab}</p> <p><i>Kräftefeld:</i> $F(q) = F_{an}(q) + F_{ab}(q)$</p> <p><i>Lokale Minima:</i> Durch Summation von U_{an} und U_{ab} kann U lokale Minima besitzen. Wenn der Roboter sich in Richtung des negativen Gradienten des Potentialfeldes bewegt, kann er in einem solchen lokalen Minimum „steckenbleiben“ => sicherstellen, dass q_{Ziel} das einzige lokale Minimum ist.</p>
<p>Bewegungs- planung</p>	<p><u><i>Probabilistic Roadmaps (PRM)</i></u></p> <p><i>Vorverarbeitung:</i> Erzeuge kollisionsfreien Graphen durch Wahl zufälliger Punkte (sampling) die dann mit kollisionsfreien Pfaden verbunden werden</p> <p><i>Algorithmus (Lokale Planung)</i></p> <p>N: Anzahl der Knoten im Graphen R: PRM, Graph</p> <p>Erzeugen von N zufälligen Konfigurationen in C_{free}</p> <p>Einfügen der erzeugten Konfigurationen als Knoten in R</p> <p>Für jeden Knoten v_i in R</p> <p> Finde die k nächsten Nachbarn von v_i aus R: $N(v_i)$</p> <p> Für jeden Knoten v aus $N(v_i)$</p> <p> Wenn neuer kollisionsfreier Pfad von v nach v_i, dann füge die Kante (v, v_i) in R ein</p> <p>Ergebnis: R</p> <p>Zufällig: Konfiguration wird zufällig generiert und auf Kollisionen geprüft</p> <p>Grid: Konfigurationen mit diskreter Auflösung → Auflösung einzelner Zellen hierarchisch</p> <p>Halton: Menge von Punkten die Bereich besser abdeckt als Grid</p> <p>Zellenbasiert: Sampling in Zellen mit kleiner werdenden Maß, Zellgröße jeder Iteration kleiner</p> <p><i>Anfrage:</i> Verbinde q_{start} und q_{Ziel} mit dem Graphen Suche Weg zwischen q_{start} & q_{Ziel} (z.B. A*)</p> <p><i>Eigenschaften</i></p> <ul style="list-style-type: none"> - Einmalige Konstruktion des Graphen (multi-query mehrere Anfragen gleichzeitig) - Randomisierter Ansatz zur Konstruktion (probabilistisch) - Verfahren hängt stark vom verwendeten Sampling ab - Nicht vollständig, da der Graph C_{free} nur approximiert <p><u><i>Dynamic Roadmaps (DRM)</i></u></p> <p><i>Vorverarbeitung</i></p> <ul style="list-style-type: none"> - Approximation des Konfigurationsraum durch Roadmap (selbstkollisionsfrei, zufällige Punkte) - Approximation des Arbeitsraums durch Voxel (Würfel) - Abbildung ϕ_{WC} von Voxel → Roadmap (Knoten, Kanten) überprüfe alle Kollisionen zwischen Knoten/Kanten und allen Voxeln <p><i>Anfrage</i></p> <ul style="list-style-type: none"> - Ermittle Voxel mit Hindernis - Anpassen der Roadmap (Lösche alle zugehörigen Knoten und Kanten aus Roadmap) - Planen in der angepassten Roadmap (Verbinde q_{start} & q_{Ziel} mit Graphen und suche Weg) <p><i>Eigenschaften</i></p> <ul style="list-style-type: none"> - sucht kürzesten Weg → Bahn nah an Objekten → Voxel mit Sicherheitsabstand löschen - Kanten die durch Voxel nahe an Objekten gehen höheres Gewicht zuordnen

Bewegungs- planung

Rapidly-exploring Random Trees (RRT)

Initialisierung: Erzeuge leeren Baum T und füge q_{Start} in T ein & C_{Obs} Form unbekannt

Iteration

- Erzeuge zufälligen Punkt q_s
- bestimme den nächsten Nachbarn q_{nn} in T
- Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein (Schrittweite d , Prüfe jeden Teilpfad auf Kollision mit C_{Obs} und stoppen wenn Kollision erkannt wurde)
- Gehe zum Anfang

Prüfe in jedem k -ten Schritt, ob q_{Ziel} mit T verbunden werden kann.

Kollisionsprüfung

Im Arbeitsraum ausführen (jeder Punkt beschreibt Konfiguration. Continuous Collision Detection (langsam) vs. Sampling basiert (schnell aber nicht exakt mit Sampling Distanz))

Eigenschaften

- RRTs probabilistisch vollständig, d.h. sie breiten sich gleichmäßig im Konfigurationsraum aus
- Wahrscheinlichkeit, dass Knoten von T erweitert proportional zur Größe der Voronoi Region

Nachbearbeitung: wähle zufällig zwei Knoten im Lösungsweg \rightarrow Verbindung frei \rightarrow verbinde

Bidirektionale RRTs

Zwei Bäume (von q_{Start} bzw q_{Ziel}), zufälliger Punkt erweitert beide Punkte (Nächst Nachbar $T1$ und nächster Nachbar $T2$), Lösung gefunden wenn beide Bäume bei q_s

Constrained RRT:

Nebenbedingungen (Orientierung, etc.) \rightarrow Random Gradient Descent/First Order Retraction

RRT*:

- Trajektorie i.d.R. nicht optimal \rightarrow iterative Optimierung des Suchbaums
- **Nachteil:** Uni-direktionaler Ansatz & längere Laufzeiten
- **Vorteil:** Findet bessere Pfadkosten, niedrigere Varianz

Dynamic Domain RRT

Beschränkt in der Nähe von Hindernissen die Sampling Domäne eines Knotens auf dessen Dynamic Domain (DD = Approximation sichtbarer Voronoi-Regionen durch Radius r).

Bridge Sampling

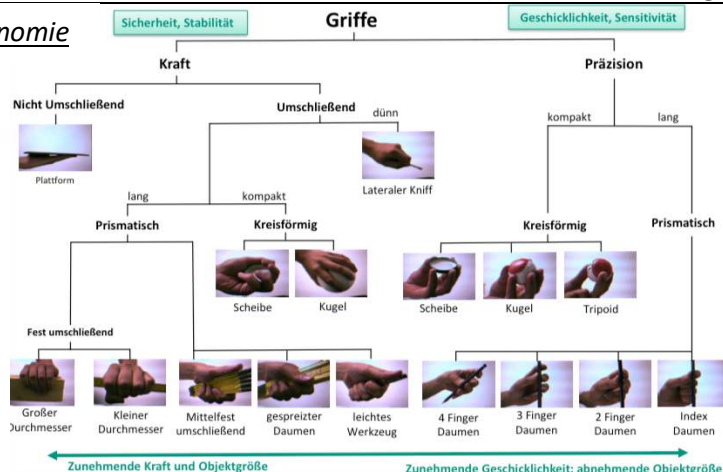
Wähle zielgerichtet Punkte in engen Passagen für nächste Stichprobe

1. Wähle gleichverteilt einen zufälligen Punkt q_1 aus C_{Obs}
2. Wähle nach geeigneter Wahrscheinlichkeitsverteilung q_2 aus C_{Obs} in der Nähe von q_1
3. Wenn Mittelpunkt q_s zwischen q_1 und q_2 in C_{free} , dann wähle diesen als neue Stichprobe
4. Wiederhole

\rightarrow Erhöht die Stichprobendichte in interessanten Bereichen (für RRT und PRM geeignet)

Griffplanung

Cutkosky Grifftaxonomy



Griffplanung

Menschliche Hand: 27 Knochen, 27 DoF

Griff: Menge von Kontaktpunkten auf der Oberfläche eines Objekts, die potentielle Bewegungen des Objekts unter dem Einfluss externer Kräfte einschränken/kompensieren

Greifanalyse: Objekt & Griff → Stabilität des Griffs unter Nebenbedingungen?

Greifsynthese: Objekt und Menge von Nebenbedingungen → Menge von Kontaktpunkten?

Punktekontakt ohne Reibung: Angreifende Kraft wirkt ausschließlich normal zur Fläche

Starrer Punktekontakt mit Reibung: Angreifende Kraft wirkt normal & tangential (verbunden durch Coulombsches Reibungsgesetz)

Nicht starrer Punktkontakt mit Reibung: Angreifende Kraft wirkt sowohl normal als auch tangential. Zusätzlich axiale Momente (verbunden durch Coulombsches Reibungsgesetz)

Wrenchvektor: Kontaktpunkte p_i wirkende Kräfte f_i und Momente τ_i zu einem Vektor

- Planarer Griff: $w = (f_x, f_y, f_z)^T$

- Räumlicher Griff: $w = (f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)^T$

Greifmatrix: $G = [{}^1w_n, {}^1w_t, {}^1w_\theta, \dots, {}^m w_n, {}^m w_t, {}^m w_\theta]$ normale, tangentiale, axiale Momente θ

Gleichgewichtsgriff: Summe aller Kräfte und Momente die auf das gegriffene Objekt wirken null

Kraftgeschlossener Griff: Während Transferbewegung und Ausführung bleibt Gleichgewicht

Ohne Reibung: Rotationssymmet. Objekt → 4 Kontaktpunkte, Beliebig → 12, Polyeder → 7

Mit Reibung: Planares Objekt → 3 Kontaktpunkte, räumlicher Fall → 4 Kontaktpunkte

Formgeschlossener Griff: Unterliegt stärkeren Einschränkungen als ein kraftgeschlossener Griff, da für jeden Kontaktpunkt ausschließlich die Nichtdurchdringungseigenschaften co-linear zum korrespondierenden externen Oberflächen-Normalenvektor berücksichtigt werden.

Kraftschluss: Kinematik kann aktiv Kräfte erzeugen, um einer externen Störung zu widerstehen

Formschluss: Die Kontakte an sich verhindern, dass sich das Objekt bewegen kann

Greifplanungssysteme

Kriterium 1: Typ des verwendeten Greifers

Kriterium 2: Typ der zugrunde liegenden Greifplanungsalgorithmen. Geometrisch basiert

Kriterium 3: Typ der zu greifenden Objekte

Kriterium 4: Typ der zu manipulierenden Szenen. Deterministisch oder nicht deterministisch

Kriterium 5: Einsatz von Sensorik (keine, taktil, visuell)

Objektklassen für das Greifen

Bekannte Objekte: Geometrie bekannt, Schwierig

Bekannte Objektklasse: konkrete Geometrie unbekannt (z.B. Typ „Flasche“), Schwieriger

Unbekannte Objekte: Geometrie & Klasse unbekannt, Am Schwierigsten

Vorwärtsplanung

Vorteil: ähnlich zur realen Ausführung, Griffe die mit hoher Wahrscheinlichkeit funktionieren

Algorithmus

1. Lade Hand- & Objektmodell in Simulationsumgebung

2. Erzeuge Griffkandidaten

3. Evaluation der Griffkandidaten (Griffqualität mittels Kraftschluss Metrik)

Kraftschluss-Metrik: Wie gut kann Griff externen Kräften widerstehen (Grasp Wrench Space)

Zufallsbasierte Vorwärts-Griffplanung

1. Randomisierte Erzeugung von Greifhypothesen

2. Kontaktermittlung

3. Evaluation von Hypothesen

Teilobjekte: Formprimitiven (Greifstrategien), Box-Dekomposition, Superquadriken, Medial-Achse Transformation, Oberflächennormalen

Griffplanung	<p><u>Griffhypothesen von Boxen</u>: Griffpunkt (Mittelpunkt Seitenfläche), Griffrichtung (Entlang der Normalen Seitenfläche), Handorientierung (Vier Möglichkeiten)</p> <p><u>Griffplanung mit medialen Achsen</u></p> <ol style="list-style-type: none"> 1. Abtasten der Objektoberfläche 2. Berechnen der medialen Achse 3. Analyse der Querschnitte der medialen Achse (Minim Spanning Tree, Clustern, Konvexe) 4. Erzeuge Griffhypothese 5. Evaluere Griffstabilität
Bildrepräsentation	<p>Ein Bild ist ein 2D-Gitter von diskreten Punkten (Pixel). Bildkoordinaten sind definiert durch u (horizontal) und v (vertikal). Der Ursprung ist oben links und die Einheit ist Pixel.</p> <p><u>Auflösung</u>: Auflösung ist das kleinste erkennbare Detail in einem Bild.</p> <p><u>Monochrombild</u>: diskrete Funktion $\text{Img}: [0 \dots n - 1] \times [0 \dots m - 1] \rightarrow [0, q]$ (i.d.R. $[0, 255]$)</p> <p><u>RGB Farbraum</u>: Additive Farbmischung, i.d.R. 1 Byte/Farbe $\text{Img}: [0 \dots n - 1] \times [0 \dots m - 1] \rightarrow [0 \dots R] \times [0 \dots G] \times [0 \dots B]$</p> <p><u>HSV Farbraum</u>: Hue, Saturation, Intensity/Value → Farbe getrennt von Helligkeit und Sättigung → unempfindlich gegen Beleuchtungsänderung</p> $H = \begin{cases} \theta & \text{falls } B < G \\ 360 - \theta & \text{sonst} \end{cases} \quad \theta = \arccos \frac{2R - G - B}{2\sqrt{(R - G)^2 + (R - B)(G - B)}}$ $S = 1 - \frac{3}{R + G + B} \min(R, G, B) \quad I = \frac{1}{3}(R + G + B) \quad V = \max(R, G, B)$ <p><u>Repräsentation Graustufenbild</u>: zeilenweise linear von oben links nach unten rechts abgelegt</p> <p><u>Repräsentation Farbbild</u>: ablegen wie bei Graustufenbild aber 3 Einträge pro Pixel</p>
Kamera	<p>BLATT 7 mal anschauen was wichtig ist</p>
Filteroperation	<p><u>Additiver linearer Filter</u>: $f(x + y) = f(x) + f(y)$</p> <p><u>Homogener linearer Filter</u>: $f(ax) = a f(x)$</p> <p><u>Filter Ränder</u>: Spiegeln(Mirror/Reflect), Wiederholen (Clamp/Replicate)</p> <p><u>Tiefpassfilter</u>: Gättung, Rauschelimination (Median, Mittelwert, Gauß)</p> <p><u>Hochpassfilter</u>: Kantendetektion (Prewitt, Sobel, Laplace)</p> <p><u>Kombiniert</u>: Laplacian of Gaussian</p> <p><u>Medianfilter</u>: Rauschunterdrücken → Größe (bsp. 3x3) → Sortiere Werte → Median (Max/Min)</p> <p><u>Mittelwertfilter</u>: Rauschunterdrücken → $g(x, y) = \sum_{s=-1}^1 \sum_{t=-1}^1 f(x - s, y - t) \cdot w(i, j)$</p> <p><u>Gauß-Filter</u>: Rausch.un./Glätten → 2D-Gauß $F_{\text{Gauß}} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \sigma$ größer mehr Glättung</p> <p><u>Prewitt-X & Y Filter</u>: $p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad p_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad M \approx \sqrt{p_x^2 + p_y^2} \text{ Formel}$</p> <p><u>Sobel-X & Y Filter</u>: $s_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad s_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \quad M \approx \sqrt{s_x^2 + s_y^2} \text{ Formel?}$</p> <p><u>Laplace</u>: $\nabla^2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \rightarrow$ Kanten dünner als bei Prewitt oder Sobel Formel?</p> <p><u>Laplacian of Gaussian</u>: $\Delta F(x, y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix} \rightarrow$ Gauss glätten → Laplace</p>

Segmentierung	<p>Aufteilung eines Bildes in aussagekräftige Segmente.</p> <p><u>Schwelwertfilterung</u>: $Img'(u, v) = \begin{cases} 255 & \text{falls } Img(u, v) > T \\ 0 & \text{sonst} \end{cases}$ T ist der Schwellwert</p> <p><u>HSV-Schwelwert</u>: $Img'(u, v) = \begin{cases} 255 & \text{falls } \begin{matrix} H_{\max} \geq Img_H(u, v) \geq H_{\min} \\ S_{\max} \geq Img_S(u, v) \geq S_{\min} \\ V_{\max} \geq Img_V(u, v) \geq V_{\min} \end{matrix} \\ 0 & \text{sonst} \end{cases}$</p>																				
Morphologische Operatoren	<p><u>Dilatation</u>: Vergrößert Pixel zu größeren Bereichen. Algorithmus?</p> <p><u>Erosion</u>: Entfernen einzelner Pixel und schwach zusammenhängender Pixelgruppen Algo?</p> <p><u>Öffnen</u>: Erosion danach Dilatation → entferne dünne Stege/kleine außenliegende Objekte</p> <p><u>Schließen</u>: Dilatation danach Erosion → Überbrückung kl. Distanzen & Schließen innerer Löcher</p>																				
Canny-Kantendetektor	<ul style="list-style-type: none"> - Geringe Fehlerrate: alle Kanten finden und so nahe wie möglich an realer Kante - Kantenpunkte gut detektiert: Distanz zw. Kantenpunkte und Zentrum realer Kanten minimal - Eindeutigkeit: Detektor soll einen Punkt, nicht mehrere zu realem Kantenpunkt liefern <p><u>Algorithmus</u></p> <ol style="list-style-type: none"> 1. Rauschunterdrückung mittels Gauß Filter 2. Berechnung der Gradienten in horizontaler und vertikaler Richtung (Prewitt/Sobel) 3. Non-Maximum Supression (Kantenausdünnung) → Gradient muss lokales Maximum sein 4. Hysterese-Schwelwertverfahren (Kanten rekursive verfolgen mit low/high Schwellwert) 																				
Tiefenkameras	<p><u>Passive Sensoren</u>: Empfangen Energie, welche aus der Umwelt abgestrahlt werden</p> <p><u>Aktive Sensoren</u>: Erzeugen und senden Energie aus, die von der Umgebung reflektiert wird</p> <p><u>Stereo Vision</u>: Disparitäten für nähere Objekte haben geringe Varianz als weiter entfernte.</p> <table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Einstellbare Brennweiten/Baseline</td><td>Mindestens zwei Kameras</td></tr> <tr> <td>Keine explizite Lichtquelle erforderlich</td><td>Korrespondenzprobleme bei homogenen Flächen</td></tr> </tbody> </table> <p><u>Time of Flight</u>: Infrarot Impuls wird emittiert und dann die Zeit berechnet bis dieser zurück ist</p> <table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Keine explizite Tiefenberechnung notwendig</td><td>Niedrige Auflösung</td></tr> <tr> <td>Hohe Bildrate</td><td>Interferenz ändert Lichtquellen/Reflektion</td></tr> <tr> <td>Robust gegenüber der meisten Oberflächen</td><td></td></tr> </tbody> </table> <p><u>Structured Light</u></p> <table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Preiswert</td><td>Eingeschränkte Reichweite</td></tr> <tr> <td>Keine Korrespondenzprobleme auf homogenen Flächen</td><td>Erfordert bestimmte Lichtverhältnisse (nur für Innenraum geeignet)</td></tr> </tbody> </table>	Pro	Contra	Einstellbare Brennweiten/Baseline	Mindestens zwei Kameras	Keine explizite Lichtquelle erforderlich	Korrespondenzprobleme bei homogenen Flächen	Pro	Contra	Keine explizite Tiefenberechnung notwendig	Niedrige Auflösung	Hohe Bildrate	Interferenz ändert Lichtquellen/Reflektion	Robust gegenüber der meisten Oberflächen		Pro	Contra	Preiswert	Eingeschränkte Reichweite	Keine Korrespondenzprobleme auf homogenen Flächen	Erfordert bestimmte Lichtverhältnisse (nur für Innenraum geeignet)
Pro	Contra																				
Einstellbare Brennweiten/Baseline	Mindestens zwei Kameras																				
Keine explizite Lichtquelle erforderlich	Korrespondenzprobleme bei homogenen Flächen																				
Pro	Contra																				
Keine explizite Tiefenberechnung notwendig	Niedrige Auflösung																				
Hohe Bildrate	Interferenz ändert Lichtquellen/Reflektion																				
Robust gegenüber der meisten Oberflächen																					
Pro	Contra																				
Preiswert	Eingeschränkte Reichweite																				
Keine Korrespondenzprobleme auf homogenen Flächen	Erfordert bestimmte Lichtverhältnisse (nur für Innenraum geeignet)																				
Visual Servoing	<p>Beschreibt Verfahren bei denen visuelle Eingabedaten genutzt werden, um die Bewegung eines Roboters zu steuern.</p> <p><u>Kamera-in-Hand</u>: Kamera an Manipulator, Bewegungen beeinflussen Pose der Kamera</p> <p><u>Externes Kamera System</u>: Externes System wird zur Beobachtung genutzt</p> <p><u>Positionsbasiertes Virtual Servoing</u>: Aktuelle Pose des Zielobjekts wird aus Bild extrahiert</p> <p><u>Bildbasiertes Virtual Servoing</u>: Bewegung ergibt aus gewünschter Position der Bildmerkmale</p>																				
Punktwolken	<p>Diskrete Menge von 3D-Punkten mit einem festen Koordinaten System.</p> <p><u>Pixel</u>: Bild Elemente im 2D (Höhe, Tiefe) <u>Voxel</u>: Volumen Pixel 3D (Breite, Höhe, Tiefe)</p> <p><u>Punktwolke</u>: $P = \{(X, C) X \in \mathbb{R}^3, C \in [0 \dots 255]^3\}$ X = (x, y, z) Ortsinformation, C = (r, g, b) Farbe</p> <p><u>Iterative Closest Point</u>: Algorithmus für die Registrierung zweier Menge A, B (a-priori unbekannte Zuordnung)</p> <table border="1"> <thead> <tr> <th>Pro</th><th>Contra</th></tr> </thead> <tbody> <tr> <td>Anwendbar für Punkte, Normalen & andere</td><td>Nicht geeignet für symmetrische Objekte</td></tr> <tr> <td>Basiert auf einfachen mathem. Operationen</td><td>Konvergenz in lokalem Minimum möglich</td></tr> </tbody> </table>	Pro	Contra	Anwendbar für Punkte, Normalen & andere	Nicht geeignet für symmetrische Objekte	Basiert auf einfachen mathem. Operationen	Konvergenz in lokalem Minimum möglich														
Pro	Contra																				
Anwendbar für Punkte, Normalen & andere	Nicht geeignet für symmetrische Objekte																				
Basiert auf einfachen mathem. Operationen	Konvergenz in lokalem Minimum möglich																				

RANSAC	Random Sample Consensus: Iterative Methode zur Schätzung von Modellparametern aus Datenpunkte (nicht deterministisch). Schätzung von Linien in 2D und Ebenen in 3D.			
	Algorithmus			
	1. Wähle zufällig minimale Anzahl an Punkten, die nötig sind um Modellparameter zu berechnen (2 für 2D Linie Line Fitting, 3 für 3D Ebene)			
	2. Schätze ein Modell aus gewähltem Datensatz			
	3. Bewertung der Modellschätzung Inlier & Outlier (Berechne Teilmenge der Datenpunkte deren Abstand zum Modell kleiner ist als ein vordefinierter Schwellwert)			
	4. Wiederhole 1 bis 3 bis das Modell mit den meisten Inlinern gefunden wurde			
	Pro		Contra	
Simpel, Allgemein und einfach implementieren		Nicht deterministisch		
Robuste Modellschätzung wenn wenig Outlier		Viele Parameter		
Vielseitig Anwendbar		Genauigkeit vs. Laufzeit		
		nur wenn Outlier/Inlier Verhältnis gut		
SLAM	Simultaneous Localization and Mapping. Beschreibt die gleichzeitige Schätzung der Roboterpose und der Karte der Umgebung. Dafür werden Szenefeatures als Orientierungspunkte verwendet. Wenn sich der Roboter bewegt kann geprüft werden inwiefern sich diese bewegt haben			
Programmierung	Klassische Programmierung: erfordert Experten (Teach Panel, Textuelle Programmierung)			
	Interaktive Programmierung: Demonstration/Aufzeichnung → Segmentierung/Interpretation → Ausführung → Abstraktion & Simulation			
	On-line		Off-line	
	direkt		textuell	graphisch
	Teach-In Punkte anfahren & speichern	Play-Back Bahn abfahren & speichern	Hybride Verfahren	Explizite Programmierung
	Sensor-unterstützt Werkstück-kontur erfassen	Master-Slave Einsatz kleiner kinematischer Modelle	PdV (Interaktive Verfahren)	Bewegungs-orientiert
			Implizite Programmierung	Aufgaben-orientiert
	On-Line (direkt): Die Programmierung erfolgt direkt am Roboter (an der Robotersteuerung)			
	Off-Line (indirekt): Programmierung erfolgt ohne den Roboter mit Hilfe textueller graphischer, interaktiver methode			
	Explizit/Roboterorientiert Programmierung (imperativ): Bewegung und Greifbefehle direkt in Programmiersprache eingebunden			
Implizite/Aufgabenorientiert Programmierung (deklarativ): Aufgabe die der Roboter durchführen soll wird beschrieben bspw. in Form von Zuständen				
Direkte Programmierung	Klassisch: Jedes Gelenk fährt bis zum Anschlag → nur wenig Anfahrpunkte			
	Pro		Contra	
	Schnell bei einfachen Trajektorien		Hoher Aufwand bei komplexen Trajektorien	
	Sofort anwendbar		Nur mit und am Roboter möglich	
	Geringe Fehleranfälligkeit		Spezifisch für einen Robotertyp	
	keine Programmierkenntnisse notwendig		Verletzungsgefahr hoch durch Roboter	
	Kein Modell der Umwelt erforderlich		Keine Adaption an neue Gegebenheiten	
	Playback Bewertung: manche Roboter schwer zu bewegen, Sicherheitsrisiko, Korrektur schwer			
Master-Slave-Bewertung: teuer da zwei Roboter notwendig, aber auch für schwerste Roboter				
Sensorgestützte-Bewertung: Fehler/Verdeckung in der Bahn, Keine Erfahrung des Bedieners				

Textuelle Programmierverfahren	Pro	Contra
	Programmierung unabhängig von Roboter	Benötigt Programmierkenntnisse
	Strukturierte, übersichtliche Logik	
	Erstellung komplexer Programme	
	<u>Verbindungsprogrammierte Steuerung:</u> Hardwaresteuerung, Änderung = Hardware ändern <u>Speicherprogrammierte Steuerung:</u> Steuerungs-/Regelablauf programmiert (sehr flexibel) <u>Computerized Numerical Control:</u> Geometrische- & Technologische Beschreibungen	
Hybride Verfahren	Pro	Contra
	weniger Programmkenntnis als textuell	Sensorielle Erfassung ungenau
	Einfache Programme & leichter Fehlererkennung	Leistungsfähige Hardware notwendig
	Schnelles Erstellen komplexer Programme	Komplexe Modell notwendig
Graphische Roboterprogrammierung	<u>Virtuelles Teach-In:</u> Roboter in 3D Visualisierung manipulieren → benötigt exaktes Modell	
	Pro	Contra
	Realer Roboter wird nicht benötigt	Leistungsfähige Hardware notwendig
	Benötigt wenig Programmierkenntnisse	Komplexe Modelle für Simulation notwendig
	Leichtere Fehlererkennung	Roboter & Umwelt muss modelliert werden
	Schnelles Erstellen komplexer Programme	
	<u>Harel Statecharts:</u> Graphischer Formalismus zum Entwurf komplexer Systeme	
	<ul style="list-style-type: none">- Hierarchie: Zustände die von anderen umgeben sind, sind Überzustände- Interlevel Transition: Transitionen können zwischen Hierarchien stattfinden- Gestrichelte Linie: Markiert eine parallele Ausführung der Zustände- Zustandsphasen: entry beim Betreten, throughout während Ausführung & vor Verlassen exit- Erweiterung transitionsbasierter Datenfluss: beliebige Daten zwischen Zuständen übergeben- Erweiterung mehrere Host PCs: Lastverteilung, Robustheit & Fehlertoleranz- Erweiterung dynamische Struktur: Austausch von Unterzuständen zur Laufzeit	
	<u>STRIPS (Stanford Research Institute Problem Solver):</u> Funktionsfreie Sprache erster Ordnung	
	<ul style="list-style-type: none">- Konstantensymbole: A, B, C, ... (Namen der Blöcke)- Variablensymbole: u, v, x₁, x₂- Prädikate: handempty, ontable(bottle), on(bottle, table), in(water, cup), at(robot, fridge), ...- Operatorensymbole: pickup, putdown, stack, unstack, grasp, move, ...- Zustand der Welt: $A \wedge on(A, B)$ (keine Variablen, negative Literale, Funktionen)- Ziel: teilweise spezifizierter Zustand (alle Literale des Ziels müssen im Weltzustand sein)- Aktion: Tripel bestehend aus Deklaration, Vorbedingungen & Effekten<ul style="list-style-type: none">- Deklaration: Name und Parameterlist (bspw. grasp(hand, object, location)- Vorbedingungen: V_A Literale die wahr sein müssen (bspw. clear(object) \wedge handempty)- Effekte E_A: Auswirkung der Aktion auf den Weltzustand (Liste pos. & neg. Literale)- Ausführbarkeit: Aktion ausführbar wenn Vorbedingen im Zustand erfüllt sind	
Symbolische Planung	Pro	Contra
	Einfache Beschreibungssprache	Viele Restriktionen bei der Modellierung (geschlossene Welt, nur pos. Literale in Zuständen, Keine Quantoren, ...)
	Einfache Planung	
	Lösbarkeitsbeweis einfach	
<u>Suche im Zustandsraum</u>		
<ul style="list-style-type: none">- Tiefensuche: Findet meistens nicht den kürzesten Plan- Breitensuche: Findet immer den kürzesten Plan aber Speicheraufwendig- Heuristik basierte Suche: Güte ausschlaggebend (keine allgemeingültige Heursitik)- FastForward-Planner: Gierige Vorwärtssuche, Breitensuche um lokalen Minima entkommen- Planungsgraphen: Analyse der Prädikate und Aktionen um Heuristiken aufzustellen		

<div>Programmieren durch Vormachen</div>	<div><div><div><div><div>Generalisierung</div><div>■ Lernen von generalisierten Aktionsrepräsentationen</div><div>■ Lernen aufgabenspezifischer Constraints</div></div><div><div>Beobachtung</div><div>■ Beobachtung menschlicher Aktionen</div><div>■ Markerlos/Markerbasiert</div></div><div><div>Reproduktion</div><div>■ Erfolgsbewertung</div><div>■ Lernen aus Erfahrung</div><div>■ Modell Verfeinerung</div></div><div>Modelle</div></div></div><div><p><u>Definition:</u> Lernen aus Beobachtung des Menschen.</p><p><u>Phasen:</u> Perzeption → Kognition → Aktion</p><p><u>Vorteile:</u> Komplexitätsreduktion des Suchraums beim Lernen, Implizites Trainieren von Robotern (händisches Programmieren reduziert/eliminiert), hilft beim Verständnis von Kopplung von Perzeption und Aktion. Fähigkeiten transferieren, Beschleunigung des Lernprozesses, liefert Vorschläge für optimale Lösung.</p><p><u>Nachteile:</u> nicht geeignet wenn Lösung des Lehrers keine gute Lösung für Nachahmer ist oder Demonstration des Lehrers stark verrauscht ist</p><p><u>Herausforderungen:</u> Wer?, Wann? (Start/End/Kontext), Was? (Relevanz), Wie? (Reproduktion)</p></div></div>																	
<div>Perzeption</div>	<div><p>Teacher Execution → <i>Record Mapping</i> → Record Execution → <i>Embodiment Mapping</i> → Learner</p><p>Record Mapping: Aktionen des Lehrers Aufnehm</p><p>Embodiment Mapping: Aktionen auf den Roboter abbilden</p><table><tr><th colspan="2" rowspan="2"></th><th colspan="2">Embodiment Mapping</th></tr><tr><th>Direkt benutzt</th><th>Abgeleitet</th></tr><tr><th rowspan="2">Recording Mapping</th><th>Direkt benutzt</th><td>Teleoperation (Steuerung)</td><td>Sensors on Teacher</td></tr><tr><th>Abgeleitet</th><td>Shadowing (anderen Roboter beobachten)</td><td>External Observation (Mensch beobachten)</td></tr><tr><th colspan="2"></th><td>Demonstration</td><td>Imitation</td></tr></table><p><u>Teleoperation:</u> Direkt steuern (Joystick) → Sensoren zeichnen auf → keine Nachbearbeitung</p><p><u>Kinästhetisch:</u> durch physikalische Interaktion bewegen → Sicherheitsrisiko, nur vor Ort</p><p><u>Shadowing:</u> gleiches Embodiment, Nachahmung durch Aufnahme mit eigenen Sensoren</p><p><u>Sensors on Teacher:</u> Bewegungen direkt aufgenommen (sehr genau!) Abbildung nötig</p><p>Marker-basiert (aktiv LED & passiv) mit Erkennung durch Mehr-Kamera-Systeme → räumlich und zeitlich exakt aber teuer und hoher Aufwand/Anforderungen,</p><p><i>Codierte Kennung</i> → vereinfachte Handhabung des Marker Labelings</p><p><i>Inertiale Messeinheiten (IMUs)</i> → geringe Anforderung aber exakte Positionierung notwendig</p><p><i>Mechanische Erfassung</i> → geringe Anforderungen, Bewegungseinschränkung, techn. Komplex</p><p><i>Magn./Akust. Erfassung</i> → günstig & weniger Verdeckung vs. geringe Reichweite und ungenau</p><p><u>External Observation:</u> Marker-lose Verfahren (direkt) → Aufnahme mit RGB/Tiefenkamera günstig & geringe Umgebungsanf. vs. komplexe Algos, hoher Error, niedrige zeitl. Auflösung</p><p><i>Skeleton Tracking</i> → Tiefen- & Skelettdaten mit 30fps aus Einzelbildern berechnet</p><p><i>Segmentation & Tracking</i> → Tracking von Farb- & Tiefenregionen, Optischer Fluss zw. Frames</p></div>			Embodiment Mapping		Direkt benutzt	Abgeleitet	Recording Mapping	Direkt benutzt	Teleoperation (Steuerung)	Sensors on Teacher	Abgeleitet	Shadowing (anderen Roboter beobachten)	External Observation (Mensch beobachten)			Demonstration	Imitation
				Embodiment Mapping														
		Direkt benutzt	Abgeleitet															
Recording Mapping	Direkt benutzt	Teleoperation (Steuerung)	Sensors on Teacher															
	Abgeleitet	Shadowing (anderen Roboter beobachten)	External Observation (Mensch beobachten)															
		Demonstration	Imitation															
<div>Kognition</div>	<div><p>Beschreibt das Lernen und Repräsentieren von Fähigkeiten und Aufgaben.</p><p><u>Trajektorie:</u> nicht lineare Zuordnung zwischen Sensoren- und Motorinformationen stellt eine Repräsentation auf niedriger Eben dar. Zur Generalisierung von Bewegungen.</p><table><tr><th>Pro</th><th>Contra</th></tr><tr><td>Allgemeine Repräsentation von Bewegungen, die es erlaubt verschiedenste Arten von Signalen und Gesten zu kodieren.</td><td>Kann komplexe Fertigkeiten nicht reproduzieren</td></tr></table><p><u>Symbolisch:</u> Eine Zerlegung in eine Folge von Aktionen stellt eine Repräsentation auf hohe Ebene dar. Ein üblicher Ansatz segmentiert und kodiert die Aufgabe anhand Folgen vordefinierter Aktionen, die symbolisch beschrieben sind. Beschreibt eine sequentielle Organisation von vordefinierten Bewegungselementen.</p><table><tr><th>Pro</th><th>Contra</th></tr><tr><td>Komplexe Fertigkeiten können effizient über interaktiven Prozess gelernt werden</td><td>Benötigt viel Wissen, um die wichtigen Kennzeichnungen zur Segmentierung zu erkennen</td></tr><tr><td>Ermöglich hierarchisches Lernen</td><td>Benötigt vordefiniert Menge von einfachen Controllern zur Reproduktion</td></tr></table></div>	Pro	Contra	Allgemeine Repräsentation von Bewegungen, die es erlaubt verschiedenste Arten von Signalen und Gesten zu kodieren.	Kann komplexe Fertigkeiten nicht reproduzieren	Pro	Contra	Komplexe Fertigkeiten können effizient über interaktiven Prozess gelernt werden	Benötigt viel Wissen, um die wichtigen Kennzeichnungen zur Segmentierung zu erkennen	Ermöglich hierarchisches Lernen	Benötigt vordefiniert Menge von einfachen Controllern zur Reproduktion							
Pro	Contra																	
Allgemeine Repräsentation von Bewegungen, die es erlaubt verschiedenste Arten von Signalen und Gesten zu kodieren.	Kann komplexe Fertigkeiten nicht reproduzieren																	
Pro	Contra																	
Komplexe Fertigkeiten können effizient über interaktiven Prozess gelernt werden	Benötigt viel Wissen, um die wichtigen Kennzeichnungen zur Segmentierung zu erkennen																	
Ermöglich hierarchisches Lernen	Benötigt vordefiniert Menge von einfachen Controllern zur Reproduktion																	

Kognition	<u>Hierarchische Aufgabensegmentierung:</u> <u>Semantische Segmentierung:</u> Basiert auf Objekt-Kontaktrelation, Bewegungstrajektorie (LVL 1) <u>Bewegungssegmentierung:</u> Charakteristiken von Trajektorien, semantische Merkmale (LVL 2) <u>Batch Lernen:</u> Aktion wird gelernt, wenn alle Aktionen/Wiederholungen aufgenommen wurde <u>Inkrementelles Lernen:</u> Aktionsrepräsentation nach jeder Aktion/Wdh. gelernt/aktualisiert
Aktion	<u>Trajektorieausführung:</u> folgen mittels PD Regel oder direkten Kontrollsignal <u>Online Anpassung:</u> möglich durch Kopplungsterm in DMP, schwierig für GMM und HMM <u>Komplexe Aufgaben:</u> Aktionssequenzen in dynamischer Umgebung wobei Umgebung wahrgenommen werden muss (Hypothesen über Startzustand generieren) und Fehlerbehandlung (erkennen & lösen durch neu planen)

Rechte Hand Regel

