KARLSRUHER INSTITUT FÜR TECHNOLOGIE

Robotik I - Einführung in die Robotik WS2016/2017

Manuel Lang

8. April 2017

INHALTSVERZEICHNIS

1 Teilsysteme

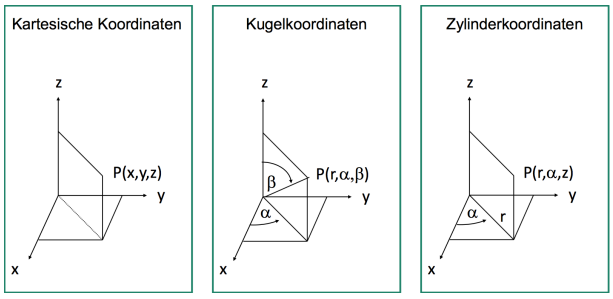
1.1 Mechanische Komponenten

1.1.1 Gelenktypen

- Rotationsgelenk (R)
  - Die Drehachse bildet einen rechten Winkel mit den Achsen der beiden angeschlossenen Glieder.
  - Beispiel: Ellbogengelenk
- Torsionsgelenk (T)
  - Die Drehachse des Torsionsgelenks verläuft parallel zu den Achsen der beiden Glieder.
  - Beispiel: Unterarmdehnung
- Revolvergelenk (V)
  - Das Eingangsgelenk verläuft parallel zur Drehachse, das Ausgangsglied steht im rechten Winkel zur Drehachse.
  - Beispiel: Schultergelenk (Arm nach vorne)
- Lineargelenk
  - Lineare Gelenke bewirken eine geleitende oder fortschreitende Bewegung entlang der Achse.
  - auch Translationsgelenk, Schubgelenk oder prismatisches Gelenk

1.1.2 Arbeitsraum

Der **Arbeitsraum** besteht aus denjenigen Punkten im 3D Raum, die von der Roboterhand angefahren werden können. Hierzu sind drei Freiheitsgrade in der Bewegung, also mindestens drei Gelenke erforderlich.
Die **Grundform des Arbeitsraums** ist der Arbeitsraum, der sich ergeben würde, wenn man die gegenseitige Behinderung der Arme des Roboters und die Begrenzung der Gelenkwinkel nicht berücksichtigt.



1.1.3 Radkonfiguration

- Differentialantrieb
  - Eigenschaften
    - Geradeaus- und Kurvenfahrten
    - Drehen auf der Stelle
    - Vorwärts- und Rückwährtsfahrten identisch
  - Vorteile
    - einfache Mechanik
  - Nachteile
    - Radregelung in Echtzeit
- Synchro-Drive
  - Eigenschaften
    - Geradeaus- und Kurvenfahrten
    - Vorwärts- und Rückwärtsfahrten identisch
    - Plattform dreht nicht mit
  - Vorteile
    - Einfache Regelung
    - Geradeausfahrt mechanisch garantiert
  - Nachteile
    - Mechanische Komplexität
- Dreirad-Antrieb
  - Eigenschaften
    - Geradeaus- und Kurvenfahrten
    - Vorwärts- und Rückwärtsfahrten unterschiedlich
  - Vorteile
    - Einfache Mechanik
  - Nachteile
    - Eingeschränkte Manövrierfähigkeit
- Mecanum-Antrieb
  - Eigenschaften / Vorteile
    - Uneingeschränkte Beweglichkeit in Richtungen x, y und ω
  - Nachteile
    - Mechanische Komplexität
    - Aufwendige Regelung

1.2 Antriebe

1.2.1 Fluidische Antriebe

- Linearantrieb
  - Kolbengeschwindigkeit $v(t) = f(t)/A$ mit $f(t)$: Fließgeschwindigkeit des Mediums (Volumen pro Zeit) und A: Grundfläche des Kolbens
  - Kolbenkraft $F(t) = P(t) * A$ mit $P(t)$: Durck des Mediums und A: Grundfläche des Kolbens
- Schaufelrad
  - Winkelgeschwindigkeit des Kolbens $W(t) = 2 * f(t)/((R^2 - r^2) * h)$ mit $f(t)$: Fließgeschwindigkeit des Mediums, h: Höhe des Schaufelrades, r: innerer Radius des Schaufelrades und R: äußerer Radius des Schaufelrades
  - Drehmoment des Kolbens $T(t) = 0.5 * P(t) * h * (R - r) * (R + r)$ mit $P(t)$: Druck des Mediums

1.2.2 Muskelartige Antriebe

- Pneumatischer Antrieb
  - Stellenergie: Komprimierte Luft bewegt Kolben, kein Getriebe
  - Vorteile: billig, einfacher Aufbau, schnelle Reaktionszeit, auch in ungünstigen Umgebungen brauchbar
  - Nachteile: laut, keine Steuerung der Geschwindigkeit bei der Bewegung, nur Punkt-zu-Punkt-Betrieb, schlechte Positioniergenauigkeit, da Luft kompressibel ist
  - Einsatz: kleinere Roboter mit schnellen Arbeitszyklen und wenig Kraft, beispielsweise zur Palettierung kleinerer Werkstücke
- Hydraulischer Antrieb
  - Stellenergie: Öldruckpumpe und steuerbare Ventile
  - Vorteile: sehr große Kräfte, mittlere Geschwindigkeit
  - Nachteile: laut, zusätzlicher Platz für Hydraulik, Ölverlust führt zu Verunreinigungen, Ölviskosität erlaubt keine guten Reaktionszeiten und keine hohen Positionier- und Wiederholgenauigkeiten
  - Einsatz: große Roboter, beispielsweise zum Schweißen

1.2.3 Elektrische Antriebe

- Stellenergie: Schritt- oder Servomotoren
- Vorteile: wenig Platzbedarf, kompakt, ruhig, gute Regelbarkeit der Drehzahl und des Drehmoments, daher auch Abfahren von Flächen oder gekrümmten Bahnen präzise möglich

- Nachteile: wenig Kraft, keine hohen Geschwindigkeiten
- Einsatz: kleinere Roboter für Präzisionsarbeiten, beispielsweise zur Leiterplattenbestückung

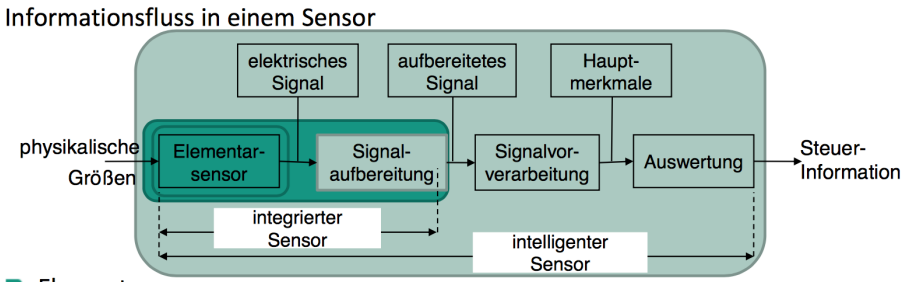
1.3 Getriebe

zur Übertragung und Umwandlung von Drehbewegungen und Kräften

- Stirnradgetriebe
  - Untersetzung $n = N_1 / N_2$
  - Winkelgeschwindigkeit $W_2 = n * W_1$
  - Drehmoment $T_2 = T_1 / n$
- Schrauben- und Spindelgetriebe
  - Lineargeschwindigkeit $v(t) = p * W(t)$ mit p: Steigerungskonstante (Ganghöhe) - Entfernung, welche die Schraube bei einer Umdrehung zurücklegt und $W(t)$: Winkelgeschwindigkeit
  - Kraft $F = \frac{2*T*\pi*d_m - \mu*p*sec\beta}{d_m*p+\mu*\pi*d_m*sec\beta}$ mit T: Drehmoment, d_m: Gewindewinkel, sec: mittlere Durchmesser der Schraube und μ: Reibungskoeffizient
- Harmonic Drive
  - Gutes Übersetzungsverhältnis
  - Sehr genaue Bewegung
  - Hohe Positioniergenauigkeit
  - Das Getriebe für Leuchtbauroboter

1.4 Sensoren

System zur Umwandlung physikalischer Größen und deren Änderung in geeignete elektronische Signale. Einsatz in Systemen, in denen der Zeitverlauf der Störgröße unbekannt ist (Regelung).



- Elementarsensor
  - Aufnahme einer Messgröße und Abbildung auf Signal
- Integrierter Sensor
  - zusätzliche Signalaufbereitung: Verstärkung, Filterung, Linearisierung, Normierung
- Intelligenter Sensor
  - integrierter Sensor mit rechnergesteuerter Auswertung, Ausgang: verarbeitete Größe, Bsp.: Mustererkenner

1.4.1 Anforderung an die Sensorik

- Genauigkeit
- Präzision
- Betriebsbereich
- Antwortgeschwindigkeit
- Kalibrierung
- Zuverlässigkeit
- Kosten
- Installationsaufwand

Wahl eines Sensors ausgehend von Aufgabenstellung und Integrationsort

1.4.2 Problemstellungen

- Ziel
  - Erfassung der Umwelt in nicht fest definierten oder sich verändernden Umgebungen
- Probleme

  - Signalverarbeitung
  - Sensorik liefert nur partielle Information: Wahl der Sensorik
  - Verwendung mehrerer Sensortypen in Mulstisensorsystemen: Fusion der Messwerte
  - Modellierung: Abstraktionsstufen des Umweltmodells

1.4.3 Klassifizierung

- Interne Sensoren
  - Kein Kontakt zur Umwelt
  - Bestimmung von Lage und Position durch Neigung, Orientierung, Drehrichtung, Beschleunigung, Lenkwinkel
- Externe Sensoren
  - Information aus Umwelt
  - Bestimmung von Position und Orientierung in Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos
- Aktive Sensoren
  - Simulation der Umwelt durch Eintrag von Energie, Messen und Auswerten der Antwort
- Passive Sensoren
  - Umwelt vorhandene Signale werden gemessen und ausgewertet

1.4.4 Aufgaben

- Interne Sensoren
  - Stellung der Gelenke
  - Geschwindigkeit, mit der sich Gelenke bewegen
  - Kräfte und Momente, die auf die Gelenke einwirken
- Externe Sensoren
  - Entfernungen
  - Lage von Positioniermarken und Objekten
  - Kontur von Objekten
  - Pixelbilder der Umwelt (CCD-Kamera)

1.4.5 Beispiele

- Interne Sensoren
  - Enkoder (inkrementell u. absolut)
  - Tachogenerator
  - Strom, Spannung, Temperatur, Feuchtigkeit
  - Kräfte
  - Neigungsmesser
  - Orientierungsmesser
  - Beschleunigungsmesser
  - Inertialsystem
- Externe Sensoren
  - Aktive Sensoren
    - Ultraschall
    - Infrarot
    - Laser-Entfernungsmesser
    - Lichtschnittverfahren
  - Passive Sensoren
    - Tastsensoren
    - Photodetektoren
    - Kameras
    - Mikrophone

2 Mathematische Grundlagen

2.1 Definitionen

- Kinematik analysiert die Geometrie eines Manipulators. Das essentielle Konzept ist die Position.
- Statik behandelt Kräfte und Momente, die sich auf den 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.

2.2 KINEMATIK

2.2.1 TERMINOLOGIE

- Gelenke (z. B. Ellbogen)
- Armelement/Glied (z. B. Oberarm)
- Endeffektor (Hand, Greifer, Schweißpistole)
- Kinematische Kette ist ein Satz an Gliedern die durch Gelenke verbunden sind.
- Kinematische Ketten können durch einen Graph repräsentiert werden. Kanten repräsentieren Glieder, Knoten repräsentieren Gelenke.

2.2.2 FREIHEITSGRADE

Freiheitsgrade (weniger formale Definition) ist eine Anzahl unabhängiger Parameter, die zur kompletten Spezifikation der Position eines Mechanismus/Objektes benötigt werden.
Beispiele:

- Ein Punkt auf einer Ebene hat 2 DoF (degrees of freedom)
- Ein Punkt im 3D Raum hat 3 DoF
- Ein Starrkörper im 2D Raum, z.B. auf einer Ebene hat 3 DoF
- Ein Starrkörper im 3D Raum hat 6 DoF

2.2.3 STARRKÖRPERBEWEGUNG

Starrkörperbewegungen werden durch zwei Eigenschaften charakterisiert

- Die Diestanz zweier beliebiger Punkte ist konstant.
- Die Orientierung des Körpers bleibt erhalten. (Ein rechtsdrehendes Koordinatensystem bleibt rechtsdrehend)

2.2.4 SO(3) UND SE(3)

Die folgenden beiden Gruppen sind in der Robotik von besonderem Interesse

- SO(3) - die Spezielle Orthogonal Gruppe, die Rotationen repräsentiert
  - Elemente aus SO(3) werden als reale 3x3 Matrizen beschrieben und erfüllen $R^T R = I$ mit $det(R) = 1$
- SE(3) - Spezielle Euklidsche Gruppe, die Starrkörperbewegungen repräsentiert
  - Elemente aus SE(3) sind von der Form (p, R), mit $p \in R^3$ und $R \in SO(3)$

2.3 Affine Geometrie

- Wir benutzen affine Geometrie um räumliche Transformationen zu beschreiben.
- Diese Transformationen bestehen aus Verknüpfungen von
  - Rotationen
  - Translationen
- Unterschiedliche Möglichkeiten zur mathematischen Beschreibung räumlicher Transformationen
  - Rotationsmatrizen und Translationsvektoren
  - Homogene Matrizen
  - Quaternionen
  - ...

2.4 Konventionen

Hier werden für Symbole folgende Konventionen vorwendet:

- Skalare: kleingeschriebene lateinische Buchstaben, Bsp: $s, t \in R$
- Vektoren: fette kleingeschriebene lateinische Buchstaben, Bsp: **a,b,c** $\in R^3$
- Matrizen: großgeschriebene lateinische Buchstaben, Bsp: $A \in R^{3x3}$
- Lineare Abbildungen (Transformationen): großgeschriebene griechische Buchstaben, Bsp: $\phi(\cdot) : R^3 \implies R^3$

2.5 Euklidscher Raum

Der euklidsche Raum ist der Vektorraum R^3 mit dem standard Skalarprodukt (auch: inneres Produkt). Beispiel: Ein Punkt c, der auf einer Linie zwischen den zwei Punkten a und b liegt, kann folgendermaßen repräsentiert werden: $c = t * a + (1 - t) * b$, $t \in (0,1) \in R$, a,b,c$\in R^3$ Ein Punkt a im euklidischen Raum wird durch Koordinaten repräsentiert, die sich auf ein Koordinatensystem e_x, e_y, e_z beziehen. $a = a_x * e_x + a_y * e_y + a_z * e_z = (a_x, a_y, a_z)^T$.
Konventionen:

- Wir benutzen orthonormale Koordinatensystem. Dis Basisvektoren e_x, e_y, e_z sind zu einander sekrecht (orthoganale) stehende Einheitsvektoren.
- Wir benutzen rechtsdrehende Koordinatensysteme.

2.6 Lineare Abbildungen, Endomorphismen

Lineare Abbildungen (Transformationen), die den euklidischen Raum auf sich selbst abbilden, nennt man Endomorphismen: $\phi(\cdot) : R^3 \implies R^3$. Endomorphismen können durch quadratische Matrizen repräsentiert werden: $\phi(a) = A * a$. A beschreibt einen Basiswechsel zwischen den originalen Basisvektoren e_x, e_y, e_z und den neuen Basisvektoren e_x', e_y',e_z': $A = (e'_x e'_y e'_z) * (e_x e_y e_z)^{-1}$

2.7 Isomorphismen

Bijektive (umkehrbare) Endomorphismen nennt man Isomorphismen. Diese können spezielle und interessante Eigenschaften besitzen:

- Sie können Winkel erhalten. (Beispiel: Skalierung und Rotation)
- Sie können Längen erhalten. (Beispiel: Rotation)
- Sie können Händigkeit erhalten. (Beispiel: Rotation. Rechtshändiges Koordinatensysten bleibt erhalten, usw.)

Eine spezielle Art von Isomorphismen, die alle genannten Kriterien erfüllt, ist die die Rotationsgruppe (oder spezielle orthogonale Gruppe) SO(3).

2.8 Die Rotationsgruppe SO(3)

- SO(3) beinhaltet alle möglichen Rotationen um willkürlich durch den Ursprung gelegte Achsen.
- SO(3) ist nicht abelsch (nicht kommutativ)
- Mit Hilfe von SO(3) kann eine Objektpose (z.B. Position und Orientierung) im Raum, so wie Transformationen zwischen zwei Robotergelenkachsen als Verkettung einer Translation und Rotation repräsentiert werden: $\phi(\cdot) : R^3 \implies R^3, \phi(x) = t + R * x$
- Die Abbildung $\phi(\cdot)$ ist nicht linear! Sie wird affin genannt.

2.9 Rotationen

2.9.1 Rotationen in 2D

- Eine Rotation in der xy-Ebene um (0,0) ist eine lineare Transformation.
- Eine Rotation von α um (0,0) transformiert den
  - Vektor $(1,0)^T$ zu $(cos\alpha, sin\alpha)^T$
  - Vektor $(0,1)^T$ zu $(-sin\alpha, cos\alpha)^T$
- Rotationsmatrix $R_\alpha(x) = (cos\alpha - sin\alpha)(sin\alpha cos\alpha) * x$

- Eine Rotation um einen Punkt c ungleich (0,0) ist keine lineare Transformation. Sie transformiert (0,0) in einen anderen Punkt als (0,0).
- Vorgehen bei der Rotation um ein beliebiges Rotationszentrum c:
  - Wir verschieben die Ebene um -c, sodass das Rotationszentrum in (0,0) liegt.
  - Dann führen wir eine Rotation um (0,0) durch.
  - Danach schieben wir die Ebene um +c zurück.
  - $R_{c,\alpha}(x) = R_\alpha(x-c)+c = R_\alpha(x)+(-R_\alpha(c)+c)$
  - $R_{c,\alpha}$ ist eine nichtlineare Transformation. Sie unterscheidet sich von R_α nur durch das Addieren einer Konstante.
  - Transformationen (wie $R_{c,\alpha}$) der Form $T(x) = A(x)+b$ werden affine Transformationen (oder auch affine Abbildungen) genannt.

2.9.2 Rotation in 3D

- Eine 2D Rotation in der xy-Ebene ist in 3D eine Rotation um die z-Achse.
- Eine Rotation von Punkten um z hängt nicht von ihren z-Werten ab. Punkte auf der z-Achse werden durch solch eine Rotation nicht betroffen.
- Die Rotationsmatrix um die z-Achse besitzt eine einfache Form:
  - Die zu xy zugehörige Teilmatrix ist identisch mit dem 2D Fall und die Einträge zum Einfluss von y auf x sowie y und umgekehrt sind 0.
  - ...

2.9.3 Rotationen um Koordinatenachsen

- Die Rotationen um die Achsen e_x, e_y, e_z des Bezugssystems sind besondere Rotationen. Sie können durch die Rotationsmatrizen R_x, R_y und R_z dargestellt werden.

2.9.4 Verkettung von Rotationen

Die Verkettung von Rotationen ist äquivalent zum Matrixprodukt. Wichtig: Es gibt zwei Möglichkeiten die Verkettung zu interpretieren (rechts nach links oder links anch rechts).

2.9.5 Probleme mit Rotationsmatrizen

Rotationsmatrizen haben mehrere Nachteile:

- Redundanz: Neun Werte für eine Rotationsmatrix
- Im Bereich des maschinellen Lernens: Wenn die Einträge einer Rotationsmatrix unabhängig voneinander präzidiert werden, ist es wahrscheinlich, dass die resultierende Matrix keine gültige Rotationsmatrix ist.

2.10 Eulerwinkel

- Es ist möglich, jede Rotation durch drei Rotationen um drei Koordinatenachsen darzustellen.
- Die Achsen können willkürlich gewählt werden, aber aus historischen Gründen wird oft die sogenannte Euler-Konvention z,x',z'‘’ verwendet.
- Die Winkel α, β, γ sind Eulerwinkel.

2.10.1 Euler-Konvention z, x', z'‘’

Euler-Winkel

- Drehung um α zm die z-Achse des BKS R_z
- Drehung um β um die neue x-Achse x' R'_x
- Drehung um γ um die neue z-Achse z'‘’ $R'_z{}''$

2.10.2 Roll, Pitch und Yaw

- Eine andere Konvention ist die Euler-Konvention x,y,z.
- Diese spezielle Eulerwinkel werden Roll, Pitch und Yaw genannt.
- Abfolge der Rotationen
  1. x-Achse des BKS um α (Roll)
  2. y-Achse des BKS um β (Pitch)
  3. z-Achse des BKS um γ (Yaw)

2.10.3 Beurteilung

- Vorteile der Eulerwinkel
  - Kompakter als Rotationsmatrizen
  - Aussagekräftiger als Rotationsmatrizen
- Nachteile der Eulerwinkel:
  - Nicht eindeutig:
    - Beispiel: in der Euler z, x', z'‘’ Konvention beschreiben die Eulerwinkel (45°,30°,-45°) und (0°,30°,-0°) die gleiche Rotation! Dieses Problem wird Gimbal Lock (karkdanische Blockade) genannt.
  - Nicht kontinuierlich:
    - Eulerwinkel einer kontinuierlichen Rotation sind nicht kontinuierlich.

    - Kleine Änderungen in der Orientierung können zu großen Änderungen der Eulerwinkel führen.
    - Konsequenz: Eine stetige Interpolation zwischen zwei Eulerwinkeln ist nicht möglich.

2.11 Affine Transformationen

- Der affine Raum ist eine Erweiterung des euklidischen Raums.
- Er beinhaltet Punkte und Vektoren, die in erweiterten (oder homogenen) Koordinaten ausgedrückt werden.
- Affine Transformationen könne so definitiert werden, dass lineare Transformationen im euklidischen Raum (wie Rotation, Skalierung und Scheren um den Ursprung) mit Translationen kombiniert und in homogenen Koordinaten ausgedrückt werden können.

Vorteile:

- Es wird ermöglicht, Rotationen um beliebige Achsen im affinen Raum zu formulieren.
- Rotationen und Translationen können in einer einzelnen Matrix kombiniert werden. Das heißt, Rotationen und Translationen können einheitlich abgehandelt werden.

2.12 Quaternionen

- Probleme mit Rotationsmatrizen:
    - Redundant
    - Schwierige Interpolation
- Probleme mit Euler Winkeln:
    - Numerisch instabil
- Quaternionen
    - Quaternionen sind eine Erweiterung der Komplex Zahlen

2.12.1 Definition

Die Menge der Quaternionen H ist definiert durch $H = C + Cj$ mit $j^2 = -1$ und $i \cdot j = -j \cdot i$. Ein Element $q \in H$ hat die Form $q = (a,u)^T = a + u_1 i + u_2 j + u_3 k$. Koeffizient a wird als Realteil bezeichnet. $u = (u_1, u_2, u_3)^T$ als Imaginärteil.

2.12.2 Rechenregeln

- Gegeben zwei Quaternionen $q, r : q = (a, u)^T, r = (b, v)^T$
- Addition: $q + r = (a + b, u + v)^T$
- Skalarprodukt: $< q|r >= a \cdot b + < v|u >= a \cdot b + v_1 * u_1 + v_2 * u_2 + v_3 * u_3$
- Multiplikation: $q \cdot r = (a + u_1 i + u_2 j + u_3 k) \cdot (b + v_1 i + v_2 j + v_3 k)$
- Konjugiertes Quaternion: $q^* = (a, -u)^T$
- Norm: $|q| = \sqrt{q * q^*} = \sqrt{q^* * q} = \sqrt{a^2 + u_1^2 + u_2^2 + u_3^2}$
- Inverse: $q^{-1} = \frac{q^*}{|q|^2}$

2.12.3 Rotationen

- Beschreibung eines Vektor p als Quaternion q: $p = (x, y, z)^T => q = (0, p)^T$
- Skalar s als Quaternion q: $q = (s, 0)^T$
- Einheitsquaternion S^3:
  - Es existiert eine Einbettung von SO(3) $\in R^3$ nach H
  - Rotationen sind definiert durch Einheitsquaternionen in S^3
  - Bilden eine Gruppe $S^3 = g \in H||q||^2 = 1$
- Sei eine Rotation beschrieben durch
  - Eine Drehachse a mit ||a|| = 1
  - und einen Drehwinkel ϕ
- dann existiert hierfür eine Repräsentation als Quaternion $q = (cos\frac{\phi}{2}, asin\frac{\phi}{2})$
- Ein Punkt v wird mit einem Quaternion q rotiert durch $v' = qvq^{-1}$
- Da q ein Einheitsquaternion ist, gilt $q^{-1} = q^*$ und somit folglich: $v' = qvq^*$
- Konkatenation zweier Rotation eines Vektors v mit zwei Quaternionen q und r: $q = (cos\frac{\phi_q}{2}, u_q sin\frac{\phi_q}{2}), r = (cos\frac{\phi_r}{2}, u_r sin\frac{\phi_r}{2})$
- Rotation mit einem Quaternion: $f(v) = qvq^*, h(v) = rvr^*$
- dann entspricht $f \cdot h$ gerade der Rotation mit dem Quaternion $p = q \cdot r$

2.12.4 Beispiel

Beispiel: Rotation des Punkts $p = (1, 0, 9)^T$ um die Drehachse $a = (1, 0, 0)^T$ mit Winkel $\phi = 90$ř

1. Darstellung von p als Quaternion v: $v = 0 + 1i + 0j + 9k$
2. Rotationsquaternion q: $q = cos\frac{\phi}{2} + 1i \cdot sin\frac{\phi}{2} + 0j + 0k$

3. Konjugiertes Quaternion $q^* = cos\frac{\phi}{2} - 1i \cdot sin\frac{\phi}{2} - 0j - 0k$
4. Rotation von v um q: $v_r = qvq^* = 0 + 1i - 9j + 0k$
5. Darstellung als Punkt $p_r = (1, -9, 0)^T$

2.12.5 Bewertung

- Vorteile:
  - Kompakte Darstellung: 4 Werte im Vergleich zu 9 bei Rotationsmatrizen
  - Anschaulich (angelehnt an Axis-Winkel Repräsentation)
  - Konkatenation möglich, ähnlich wie bei Rotationsmatrix
  - Kann für Berechnung der Inversen Kinematik verwendet werden
  - Kein Gimbal Lock
  - Repräsentation ist stetig (keine Sprünge)
- Nachteile
  - Nur Beschreibung von Rotation, keine Translation

2.12.6 Interpolation

- Ziel: Kontinuierliche Rotation zwischen zwei Orientierungen
- Probleme:
  - Euler-Winkel sind nicht kontinuierlich
  - Rotationsmatrizen haben viele Freiheitsgrade
- Interpolation von Quaternionen mittels SLERP (Spherical Linear Interploation)
  - $Slerp(q_1, q_2, t) = q_1 * (q^{-1} * q_2)^t$
  - Potenzieren von Quaternionen nicht behandelt

3 Kinematik

Robotermodellierung

- Geometrische Modellierung
  - Geometrie: Mathematische Beschreibung der Form von Körpern
- Kinematische Modellierung
  - Kinematik: Lehre der geometrischen und analytischen Beschreibung der Bewegungszustände mechanischer Systeme
- Dynamische Modellierung
  - Dynamik: Untersuchung der Bewegung von Körpern als Folge der auf sie wirkenden Kräfte und Momente

3.1 Kinematisches Modell

Definition: Das kinematische Modell eines Roboters beschreibt die Zusammenhänge zwischen dem Raum der Gelenkwinkel (Roboterkoordinaten, Konfigurationsraum) und dem Raum der Lage des Endeffektors in Weltkoordinaten (Arbeitsraum, kartesischer Raum).
Einsatzbereiche:

- Bestimmung des Zusammenhangs zwischen Gelenkwerten und Stellungen des Roboters
- Erreichbarkeitsanalyse
- Relation zwischen Körpern des Roboters (Selbstkollision)
- Relation zur Umgebung (Kollisionserkennung)

Arten der Kinematik:

- Diskretes kinematisches Modell (Vorwärtskinematik): Bestimmung der Lage des Endeffektors aus den Gelenkwinkelstellungen des Roboters.
- Inverses kinematisches Modell (Rückwärtskinematik): Bestimmung der Gelenkwinkelstellungen zu einer gewünschten Lage des Endeffektors.

3.1.1 Kinematische Kette

- Elemente einer kinematischen Kette
  - Endeffektor
  - Armelemente (Glieder, Segmente)
  - Gelenke
  - Roboterarm Basis
- Definition: Eine kinematische Kette wird von mehreren Körpern gebildet, die durch Gelenke kinematisch verbunden sind (z. B. Roboterarm).
- Typen: Offene vs geschlossene kinematische Kette
- Konventionen
  - Jedes Armelement entspricht einem starren Körper
  - Jedes Armelement ist mit dem nächsten durch ein Gelenk verbunden.
  - Bei Schub- und Rotationsgelenken: Jedes Gelenk hat nur einen Bewegungsfreiheitsgrad (Rotation oder Translation)
- Kinematische Parameter
  - Gelenkparameter, z.B. Rotationsgelenk: Rotationsachse oder Schubgelenk: Translationsrichtung

- Spezifikation der Lage der Gelenke zueinander
  - Feste Transformationen zwischen zwei Gelenken
  - Definiert die lokalen Koordinatensysteme der Gelenke
  - Transformation von Gelenk $i-1$ zu Gelenk i durch Transformationsmatrix ${}^{i-1}T_i$

  Für jedes Glied muss eine Transformation bestimmt werden (3 Rotationsparameter + 3 Translationsparameter) -> 6 Parameter pro Glied der kinematischen Kette

3.1.2 DENAVIT-HARTENBERG KONVENTION

- Ziel: Reduktion der Parameter zur Beschreibung eines Armelementes
- Eigenschaften
  - Systematische Beschreibung der Beziehungen (Translationen und Rotationen) zwischen benachbarten Gelenken
  - Reduktion der Anzahl der Parameter von 6 auf 4
- Jedes Koordinatensystem wird auf Basis der folgenden drei Regeln bestimmt:
  1. Die z_{i-1}-Achse liegt entlang der Bewegungsachse des i-ten Gelenks.
  2. Die x_i-Achse verläuft entlang der gemeinsamen Normalen (common normal) von z_{i-1} und z_i. Dabei zeigt sie weg von z_{i-1}.
  3. Die y_i-Achse vervollständigt das Koordinatensystem entsprechend der Rechte-Hand-Regel.

Bestimmung der DH Parameter:

1. Skizze des Manipulators
2. Identifiziere und nummeriere die Gelenke (1-n)
3. Zeichne die Achsen z_{i-1} für jedes Gelenk i
4. Bestimme die Parameter a_i zwischen z_{i-1} und z_i
5. Zeichne die x_i-Achsen
6. Bestimme die Parameter α_i (Verwindung um die

   $$x_i$$

   -Achsen)
7. Bestimme die Parameter d_i (Gelenkabstand)
8. Bestimme die Winkel θ_i um z_{i-1}-Achsen
9. Gelenk-Transformation-Matrizen $A_{i-1,i}$ - verknüpfe sie

3.1.3 Diskretes Kinematisches Problem

- Aus den DH-Parametern und den Gelenkwinkeln soll die Stellung des Endeffektors (Tool Center Point: TCP) ermittelt werden.
- Die Stellung des Endeffektors (TCP) in Bezug auf das BKS ist gegeben durch; $S_{Basis,Greifer}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot ... \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$
- Gelenkwinkel $\theta_1, ...\theta_n$ sind vorgegeben -> Stellung des TCP ergibt sich aus obiger Gleichung durch Einsetzen der Gelenkwinkelwerte

3.1.4 Beispiele

// TODO //

3.1.5 Jacobi-Matrizen

- Vorwärtskinematik: Gelenkwinkelstellung -> End-Effektor-Pose
- Wie sehen verwandte Beziehungen aus?
  - Gelenkwinkelgeschwindigkeiten -> End-Effektor-Geschwindigkeit
  - Drehmomente in Gelenkwinkeln -> End-Effektor-Kräfte und -Momente
- Ansatz: Vorwärtskinematik ableiten (-> Jacobi-Matrix)
- Problem: Vorwärtskinematik ist matrixwertig -> Jacobi-Matrix nicht definiert
- Lösung: Vektorwertige Repräsentation wählen (z.B. Roll-Pitch-Yaw)

End-Effektor-Geschwindigkeiten

- Annahme: Die kinematische Kette bewege sich einlang einer Trajektorie $\theta : \mathbb{R} -> \mathbb{R}^n$
- Dann gilt für die End-Effektor Pose $x(t) \in \mathbb{R}$ zum Zeitpunkt $t : x(t) = f(\theta(t))$
- Die End-Effektor-Geschwindigkeit hängt linear von den Gelenkgeschwindigkeiten ab (Kettenregel): $\dot{x}(t) = \frac{\partial f(\theta(t))}{\partial t} = J_f(\theta(t)) \cdot \dot{\theta}(t)$
- Die Jacobi-Matrix setzt kartesische End-Effektor-Geschwindigkeiten in Relation zu Gelenkwinkelgeschwindigkeiten
- Die folgenden Probleme können nun mit dieser Beziehung gelöst werden
  1. Gegeben eine kartesische End-Effektor-Geschwindgkeit, welche Gelenkwinkelgeschwindigkeiten sind notwendig, um diese zu realisieren?
  2. Gegeben die Gelenkwinkelgeschwindigkeiten, welche kartesische End-Effektor-Geschwindigkeit wird damit realisiert?

Kräfte und Momente am End-Effektor

- Annahme: Die kinematische Kette bewege sich entlang einer Trajektorie $\theta : \mathbb{R} -> \mathbb{R}^n$
- Die geleistete Arbeit muss unabhängig vom Bezugssystem konstant bleiben (Reibung vernachlässigt) $\int_{t_1}^{t_2} \dot{\theta}(t)^T \cdot \tau(t) dt = W = \int_{t_1}^{t_2} \dot{x}(t)^T \cdot F(t) dt$
- Die Beziehung muss für jedes Zeitintervall $[t_1, t_2]$ gelten, daher: $\dot{\theta}(t)^T \cdot \tau(t) = \dot{x}(t)^T \cdot F(t)$
- Bekannte Beziehung zwischen End-Effektor-Geschwindigkeit und Jacobi-Matrix: $\dot{\theta}(t)^T \cdot \tau(t) = \dot{\theta}(t)^T \cdot J_f^T(\theta(t)) \cdot F(t)$
- Da $\dot{\theta}(t)$ beliebig ist, folgt: $\tau(t) = J_f^T(\theta(t)) \cdot F(t)$
- Die Jacobi-Matrix setzt Kräfte und Momente am End-Effektor in Relation zu Drehmomenten in den Gelenken
- Die folgenden Probleme können mit dieser Beziehung gelöst werden:
  1. Gegeben eine Kraft am End-Effektor, welche Drehmomente müssen in den Gelenken wirken, um dieser Kraft zu widerstehen?
  2. Gegeben die Drehmomente in den Gelenken, welche Kräfte und Momente wirken dadurch am (fixierten) End-Effektor?

Berechnung der Jacobi-Matrix

- Jede Spalte der Jacobi-Matrix korrespondiert zu einem Gelenk θ_i der kinematischen Kette
- Ansatz: Die numerische Berechnung der Jacobi-Matrix verläuft spaltenweise
- 1. Fall: Translationsgelenk
  - Annahme: Das j-te Gelenk führe eine Translation in Richtung des Einheitsvektors $v_j \in$ durch.
  - Dann gilt: $\frac{\partial f(\theta)}{\partial \theta_j} = //TODO//$

Zusammenfassung: Jacobi-Matrix

- J_f beschreibt die Beziehung zwischen
  - Geschwindigkeit der Gelenkwinkel (n-dimensional) und Geschwindigkeit des Endeffektors (6-dimensional)
  - Drehmomente in Gelenken (n-dimensional) und Kräfte und Momente am Endeffektor (6-dimensional)
- Jacobi-Matrix ist nur gültig für eine bestimmte Gelenkwinkelkonfiguration

3.1.6 Singularitäten und Manipulierbarkeit

Singularitäten:

- Eine kinematische Kette ist in einer singulären Konfiguration, wenn die zugehörige Jacobi-Matrix nicht vollen Rang hat
  - Zwei oder mehr Spalten von J_f sind linear abhängig
- Die Jacobi-Matrix ist nicht invertierbar
  - Bestimmte Bewegungen sind unmöglich
- In der Umgebung von Singularitäten können große Geschwindigkeiten nötig werden, um eine End-Effektor-Geschwindigkeit zu halten.

Manipulierbarkeit:

- Ein Maß für die Bewegungsfreiheit „manipulability “ des End-Effektors
- Manipulierbarkeits-Ellipsoid
  - Nutze $f(\theta)$ um Einheitskreis der Gelenkwinkel-Geschwindigkeiten in den Raum der Endeffektor-Geschwindigkeiten abzubilden
  - Resultat: Manipulierbarkeits-Ellipsoid (manipulability ellipsoid)
  - Abhängig von der Gelenkwinkelkonfiguration
- Analyse
  - Kreis: Bewegungen des Endeffektors in alle Richtungen uneingeschränkt möglich
  - Degenerierte Fälle (Linie): Endeffektor-Bewegung ist eingeschränkt

Eigenwertanalyse:

- Konstruiere $A(\theta) = J(\theta)J(\theta)^T \in \mathbb{R}^{nxn}$
- $A(\theta)$ ist dann:
  - Quadratisch
  - Symmetrisch
  - positiv definit
  - invertierbar
- Eigenwerte λ und Eigenvektoren v von A: $Av = \lambda v$, $(\lambda I - A)v = 0$
- Singulärwerte $\sigma_i = \sqrt{\lambda_i}$

Berechnung:

- Maße für die Manipulierbarkeit
  - Kleinster Singulärwert $\mu_1(\theta) = \sigma_{min}(A(\theta))$

  - Inverse Kondition $\mu_2(\theta) = \frac{\sigma_{min}(A(\theta))}{\sigma_{max}(A(\theta))}$
  - Determinate $\mu_3(\theta) = detA(\theta)$
- Einsatz:
  - Analyse von Gelenkwinkelkonfigurationen
  - Vermeidung von Singularitäten

3.1.7 Repräsentation der Erreichbarkeit

- Erreichbarer Teil des Arbeitsraums für den Roboter in $\mathbb{R}^6$
- Approximation durch 6-dimensionale Gitter
- Eintrag in jeder Gitterzelle:
  - Erreichbarkeit (Reachability): Binär: Existiert mind. eine Gelenkwinkelkonfiguration, so dass der TCP innerhalb der 6D-Gitterzelle liegt.
  - Manipulierbarkeit (Manipulability): Maximaler Manipulierbarkeitswert einer Gitterzelle, z.B. $\mu_1(\theta)$
- Erstellung
  - Offline-Prozess in Simulation
  - Taste alle Gelenkwinkel ab
    * in x Schritten (z. B. x = 5°)
    * Bestimme die Lage des TCP über Vorwärtskinematik
    * Bestimme Gitterzelle und setze den Eintrag
- Anwendung
  - Vorberechnete Erreichbarkeitsinformationen
  - Schnell Entscheidung, ob eine Pose mit dem Endeffektor erreichbar ist. Aufwand: O(1)
  - Kann zur Griffselektion genutzt werden

3.2 Geometrisches Modell

3.2.1 Einsatzbereiche

- Graphische Darstellung von Körpern
- Ausgangspunkt der Abstandsmessung und Kollisionserkennung
- Grundlage zur Berechnung der Bewegungen
- Grundlage zur Ermittlung der wirkenden Kräfte und Momente

3.2.2 Klassifizierung

- Klassifizierung nach Raum
  - 2D Modelle
  - 2,5D Modelle
  - 3D Modelle
- Klassifizierung nach Grundprimitiven
  - Kanten- bzw. Drahtmodelle
  - Flächen- bzw. Oberflächenmodelle
  - Volumenmodelle

3.2.3 Beispiele

//TODO//

4 Inverse Kinematik

4.1 Inverses kinematisches Problem

Bestimmung der Gelenkwinkelstellungen zu einer gewünschten Lage des Endeffektors.
Vorgehensweise

- Kinematisches Modell: $P_{TCP} = T_{BKS,TCP}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot ... \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$
- Gegeben: P_{TCP}
- Gesucht: θ
- Ansatz: Gleichung nach θ auflösen (nichtlineares Problem)

Eindeutigkeit

- In der Ebene gibt es für Systene mit $n \geq 3$ Bewegungsfreiheitsgraden mehrere Möglichkeiten eine vorgegebene Endeffektorstellung zu erreichen.
- In SE(3) gilt dies für alle Roboter mit $n \geq 6$ Bewegungsfreiheitsgraden.
- Reduktionsstellungen sind solche, zu deren Erreichen $n \leq 5$ Bewegungsfreiheitsgrade ausreichen würden.

4.2 Geschlossene Methoden

4.2.1 Geometrische Methoden

Vorgehen:

- Nutze geometrische Beziehungen, um die Gelenkwinkel θ aus der T_{TCP} zu bestimmen.
- Das kinematische Modell wird dabei nicht direkt verwendet.
- Anwendung von:
  - Trigonometrischen Funktionen
  - Sinus- / Kosinussätzen

Polynomialisierung:

- Transzendente Gleichungen sind in der Regel schwer zu lösen, da die Variable θ gewöhnlich in der Form $cos\theta$ bzw. $sin\theta$ auftritt.
- Werkzeug: Substitution $u = tan(\frac{\theta}{2})$ unter Verwendung von:
  - $cos\theta = \frac{1-u^2}{1+u^2}$
  - $sin\theta = \frac{2u}{1+u^2}$

4.2.2 Algebraische Methoden

- Gleichsetzen von TCP Pose P_{TCP} und Transformation $T_{BKS,TCP}$ aus dem kinematischen Modell: $P_{TCP} = T_{BKS,TCP}(\theta)$
- Koeffizientenvergleich der beiden Matrizen $a_{ij} = b_{ij}$
- 16 Gleichungen bei homogenen Matrizen in 3D (4 trivial: 0=0,1=1) -> 12 nicht-triviale Gleichungen

Lösungsalgorithmus

- Problem: Oft können nicht alle Gelenkwinkel aus den 12 Gleichungen bestimmt werden.
- Ansatz: Kenntnis der Transformationen erhöht Chance, die Gleichungen zu lösen.
- Gegeben: Die Transformationsmatrizen $A_{0,1} \cdot A_{1,2} \cdot ... \cdot A_{n-1,n}$ und P_{TCP}
- Gesucht: Die Gelenkwinkel θ_1 bis θ_n

Vorgehensweise

- Algorithmus zur algebraischen Lösung
  - $P_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot A_{2,3}(\theta_3) \cdot A_{3,4}(\theta_4) \cdot A_{4,5}(\theta_5) \cdot A_{5,6}(\theta_6)$

- Vorgehensweise:
  1. Invertiere $A_{0,1}(\theta_1)$ und multipliziere beide Seiten der Gleichung mit $A_{0,1}^{-1}$
  2. Versuche aus dem neu entstehenden Gleichungssystem eine Gleichung zu finden, die nur eine Unbekannte enthält und löse diese Gleichung nach der Unbekannten.
  3. Versuche eine Gleichung im Gleichungssystem zu finden, die durch die Substitution der im letzten Schritt gefundenen Lösung nach einer Unbekannten lösbar ist.
  4. Falls keine Lösungen mehr gefunden werden können, so muss eine weitere Matrix $(A_{1,2}(\theta_2))$ invertiert werden.
  5. Widerhole die Schritte 1-4 bis alle Gelenkwinkel ermittelt sind.

4.3 Numerische Methoden

- Jacobi-Matrix (bereits gezeigt)
- Linearisierung
  - TCP-Pose über Vorwärtskinematik: $P_{TCP,t} = f(\theta_t)$
  - Jacobi-Matrix liefert Bewegungstangenten in der aktuellen Stellung θ_t: $J_f(\theta_t) = \frac{\partial f(\theta_t)}{\partial \theta_t}$
  - Annahme: Modell gültig für kleine $\Delta\theta$
    * Lineare Approximation der Bewegung
    * Approximationsfehler ϵ existiert
- Umkehrung
  - Bisher erreicht: Lokale, lineare Annäherung an die Vorwärtskinematik $\Delta x = f(\Delta\theta) \approx J_f(\theta) \cdot \Delta\theta$
  - Gesucht: Lösung für das inverse Problem $\Delta\theta \approx F(\Delta x) = J_f^{-1}(\theta) \cdot \Delta x$
  - Invertierung ist möglich, wenn:
    * $J_f(\theta)$ ist quadratisch
    * $J_f(\theta)$ hat vollen Rang
- Pseudoinverse
  - Ansatz: Pseudoinverse - Verallgemeinerung der inversen Matrix auf singuläre und nicht-quadratische Matrizen $A \in \mathbb{R}^{mxn}$.
  - Moore-Penrose Pseudoinverse $A^{\#} = A^T(AA^T)^{-1}$
  - Es gelten:
    * $(A^{\#})^{\#} = A$
    * $(A^T)^{\#} = (A^{\#})^T$

    * $(\lambda A)^{\#} = \lambda^{-1} A^{\#}$, für ein $\lambda \not\emptyset$
  - Problem: Berechne die im Sinne der Summe der Fehlerquadrate (least squares) bestmögliche Lösung eines Systems von linearen Gleichungen

- Zusammenfassung
  1. Vorwärtskinematik als Funktion: $x(t) = f(\theta(t))$
  2. Ableitung nach der Zeit: $\frac{\partial x(t)}{\partial t} = \dot{x}(t) = J_f(\theta)\dot{\theta}(t)$
  3. Übergang zum Differenzenquotienten: $\Delta x \approx J_f(\theta)\Delta\theta$
  4. Umkehrung: $\Delta\theta \approx J_f^{\#}(\theta)\Delta x$

- Iteratives Vorgehen
  - Gegeben: Sollpose des TCP $P_{TCP,soll}$
  - Gesucht: Gelenkwinkel θ, der $P_{TCP,soll}$ realisiert
  - Iterativer Ansatz beginnend bei Initialkonfiguration θ_0 und $P_{TCP,0}$
    1. Berechne $P_{TCP,t}$ in Iteration t aus Gelenkwinkelstellungen θ_t
    2. Berechne Fehler Δx aus $P_{TCP,soll}$ und berechneter $P_{TCP,t}$
    3. Benutze approximiertes inverses kinematisches Modell F, um Gelenkwinkelfehler $\Delta\theta$ zu berechnen
    4. Berechne $\theta_{t+1} = \theta_t + \Delta\theta$
    5. Fahre mit Iteration $t+1$ fort

- Singularitäten
  - Pseudoinverse ist in der Nähe von Singularitäten instabil
  - Umgang mit Singularitäten (nicht immer möglich)
  - Damped least squares (auch Levenberg-Marquardt Minimierung)
    * Die Pseudoinverse $J_f^{\#}(\theta)$ löst die Gleichung $J_f(\theta)\Delta\theta = \Delta x$ optimal nach $\Delta\theta$.
    * Optimal bezieht sich auf die Summe der Fehlerrate $\min_{\Delta\theta} ||J_f(\theta)\Delta\theta - \Delta x||_2^2$
    * Ansatz: Minimiere stattdessen $\min_{\Delta\theta} ||J_f(\theta)\Delta\theta - \Delta x||_2^2 + \lambda^2||\Delta\theta||_2^2$ mit einer Dämpfungskonstante $\lambda > 0$.
    * Die zugehörige Gleichung ist $(J^T J + \lambda^2 E)\Delta\theta = J^T \Delta x$
    * Daraus ergibt sich $\Delta\theta = (J^T J + \lambda^2 E)^{-1} J^T \Delta x = J^T (JJ^T + \lambda^2 E)^{-1} \Delta x$
    * Die Dämpfungskonstante $\lambda > 0$ muss anwendungsspezifisch gewählt werden
      · Groß genug für numerische Stabilität in Umgebungen von Singularitäten
      · Klein genug für schnelle Konvergenzrate

- Stabilitätsbetrachtung

- Beide Ansätze (Pseudoinverse und Damped Least Squares) können durch Singularitäten instabil werden
- Eine Analyse der Stabilität ist über eine Singulärwertzerlegung möglich
- Singulärwertzerlegung: Eine Matrix $J \in \mathbb{R}^{mxn}$ wird dargestellt durch zwei orthogonale Matrizen $U \in \mathbb{R}^{mxm}$ und $V \in \mathbb{R}^{nxn}$ und eine Diagonalmatrix $D \in \mathbb{R}^{mxn}$, in der Form: $J = UDV^T$
- OBdA: Singulärwerte σ_i auf der Diagonalen von D seien sortiert; $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$.
- Die Singulärwertzerlegung von J existiert immer und sie erlaubt die folgende Darstellung von J: $J = \sum_{i=1}^{m} \sigma_i u_i v_i^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$
- wobei u_i und v_i die Spanteln von U und V sind, sowie $r = rangJ$.
- Für die Pseudoinverse $J^\#$ gilt (durch die Orthogonalität von U und V): $J^\# = VD^\#U^T = \sum_{i=1}^{r} \sigma_i^{-1} v_i u_i^T$
- Für die innere (zu invertierende) Matrix gilt: $JJ^T + \lambda^2 E = (UDV)^T(VD^TU^T) + \lambda^2 E = U(DD^T + \lambda^2 E)U^T$
- $DD^T + \lambda^2 E$ ist eine nicht-singuläre Diagonalmatrix mit den Diagonaleinträgen $\sigma_i^2 + \lambda^2$. Daher ist $(DD^T + \lambda^2 E)^{-1}$ eine Diagonalmatrix mit den Diagonaleinträgen $(\sigma_i^2 + \lambda^2)^{-1}$.
- Es folgt: $J^T(JJ^T + \lambda^2 E)^{-1} = VD^T(DD^t + \lambda^2 E)^{-1}U^T = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T$
- Pseudoinverse: $J^\# = \sum_{i=1}^{r} \frac{1}{\sigma_i} v_i u_i^T$
- Damped least squares: $J^T(JJ^T + \lambda^2 E)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T$
- Die Invertierung von J hat in beiden Fällen eine ähnliche Forum.
- Die Pseudoinverse wird instabil, wenn $\sigma_i -> 0$ (Singularität)
- Für große σ_i (verglichen mit λ) verhält sich damped least squares wie die Pseudoinverse
- Für $\sigma_i -> 0$ verhält sich dumped least squares wohldefiniert.

5 Dynamik

5.1 Dynamisches Modell

- Definition: Das dynamische Modell beschreibt den Zusammenhang von Kräften, Momenten und Bewegungen, welche in einem mechanischen Mehrkörpersystem auftreten.
- Zweck:
  - Analyse der Dynamik
  - Synthese mechanischer Strukturen

  - Modellierung elastischer Strukturen
  - Reglerentwurf

- Allgemeines Modell:
  - Roboter besteht aus n Partikeln mit Masse m_i und Position r_i
  - Newtons zweites Gesetz: $F_i = m_i \cdot \ddot{r}_i$
  - Partikel können sich wegen Verbindungen und Gelenken nicht unabhängig voneinander bewegen
    * Einführung von Einschränkungen (constraints) der Forum $g_j(r_1, ..., r_k) = 0$
    * Einschränkungen dieser Art nennt man auch holonome Einschränkungen
    * Einschränkungen wirken auf den Roboter durch Ausübung von Einschränkungskräften (constraint forces)
  - Parameter im allgemeinen Modell: $3n + k$ (da $r_i \in \mathbb{R}^3$)
  - Tatsächliche Freiheitsgrade $3n - k$
  - Ziel: Minimaler Parametersatz, der das System vollständig beschreibt
  - -> Generalisierte Koordinaten

- Generalisierte Koordinaten
  - Definition: Minimaler Satz an voneinander unabhängigen Koordinaten, der den aktuellen Systemzustand vollständig beschreibt
  - Generalisierte Koordinaten: $q_1, ...q_m$ mit $m = 3n - k$
  - Gesucht: Funktionen für Position der Massepunkte: $r_i = f_i(q_1, ..., q_m)$ mit $i = 1, ..., n$
  - die gleichzeit die Einschränkungen (constrainsts) einhalten: $g_j(r_1, ..., r_k) = 0$ mit $j = 1, ..., k$

- Bewegungsgleichung
  - Im dynamischen Modell werden die Beziehungen zwischen Kräften/Momenten und den Lagen, Geschwindigkeiten und Beschleunigungen der Armelemente dargestellt.
  - $\tau = M(q) \cdot \ddot{q} + c(\dot{q}, q) + g(q)$ mit:
    * τ: $nx1$ Vektor der generalisierten Kräfte
    * $M(q)$: nxn Massenträgheitsmatrix
    * $c(\dot{q}, q)$: $nx1$ Vektor der Zentripetal- und Coriliskomponenten
    * $g(q)$: $nx1$ Vektor der Gravitationskomponenten
    * $q, \dot{q}, \dot{d}otq$: $nx1$ Vektor der generalisierten Koordinaten (Position, Geschwindigkeit, Beschleunigung)
  - Reibung kann als zusätzlicher Term einfließen, wird aber häufig vernachlässigt

- Direktes dynamisches Problem
  - Aus äußeren Kräften und Momenten sowie Anfangszustand wird unter Verwendung des dynamischen Modells die sich ergebenden Bewegungsänderungen berechnet.
  - $\tau = M(q) \cdot \ddot{q} + c(\dot{q}, q) + g(q$
  - Differentialgleichung nach $q(t), \dot{q}(t), \ddot{q}(t)$ lösen
- Inverses dynamisches Problem
  - Aus den gewünschten Bewegungsparametern sollen, unter Verwendung des dynamischen Modells, die dazu erforderlichen Stellkräfte und -momente ermittelt werden.
  - $\tau = M(q) \cdot \ddot{q} + c(\dot{q}, q) + g(q$
  - Rechten Teil der Gleichung berechnen

5.2 Modellierung der Dynamik

- Lagrange: Analytische Methode
  - Arbeits- oder Energiebetrachtungen
  - Formales Ableiten ergibt die Bewegungsgleichungen
- Newton-Euler: Synthetische Methode
  - Basiert auf linearem Impuls und Drehimpuls (Drall)
  - Isoliertes Betrachten der Armelemente

5.2.1 Methode nach Lagrange

- Langrange-Funktion: $L(q, \dot{q}) = E_{kin}(q, \dot{q}) - E_{pot}(q)$
- $\tau_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i}$ mit q_i: i-te Komponente der generalisierten Koordinaten und τ_i: i-te Komponente der generaliserten Kräfte
- Ziel: Ermittle für jedes Gelenk i eines Roboters die Bewegungsgleichung $\tau_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i}$
- Vorgehen:
  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 Ableitungen
- Zusammenfassung

- Zur Ermittlung der Bewegungsgleichungen müssen nur die kinetische und potentielle Energie aufgestellt werden
- Die Bewegungsgleichungen folgen dann formal durch differenzieren
- Einfaches Aufstellen der Gleichungen
- Geschlossenes Modell
- Analytisches auswertbar
- Berechnung sehr umfangreich $O(n^3)$, n = Anzahl der Gelenke
- Nur Antriebsmomente werden berechnet

5.2.2 Methoden nach Newton-Euler

- Grundprinzip:
  - Betrachtung des Massezentrums eines einzelnen Armelementes
    - Kraft = Impuls abgeleitet nach Zeit (Zweites Newtonsches Gesetz) $F_i = \frac{d}{dt}(m_i v_{s,i}) = m_i \dot{v}_{s,i}$
    - Drehmoment = Drehimpuls abgeleitet nach Zeit $N_i = \frac{d}{dt}(I_i \omega_{s,i}) = I_i \dot{\omega}_{s,i}$
  - Kräfte und Momente, die auf ein Armelement wirken, lassen sich aus Geschwindigkeit und Gelenkwinkelgeschwindigkeit berechnen.
- Verkettung
  - Die Beschleunigungen $\dot{v}_{s,i}$ und $\dot{\omega}_{s,i}$ 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 F_i und das Drehmoment N_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
- Vorwärtsgleichungen
  - Aus dem kinematischen Robotermodell und dem Systemzustand lassen sich Winkelgeschwindigkeiten, translatorische Geschwindigkeiten und Beschleunigungen des Gelenkes und Armelementes i im Basiskoordinatensystem bestimmen.
  - Bestimmt werden:
    - Winkelgeschwindigkeit ω_i
    - Winkelbeschleunigung $\dot{\omega}_i$

        * Geschwindigkeit v_i und Beschleunigung $\dot{v}_i$ der Basen der Koordinatensysteme
        * Geschwindigkeiten $v_{s,i}$ und Beschleunigungen $\dot{v}_{s,i}$ der Massenmittelpunkte der Armelemente
    - Die Geschwindigkeiten und Beschleunigungen für Armelement $i+1$ lassen sich rekursiv unter Berücksichtigung der Kinematik aus den Geschwindigkeiten und Beschleunigungen des Armelementes i berechnen.
    - Die Funktionen G_i lassen sich aus den Gleichungen für Kraft und Drehmoment ableiten

- Rückwärtsgleichungen
    - Beginnend vom Greifer des Roboters lassen sich die dynamischen Größen des Systems rekursiv bis zur Basis berechnen. Dabei werden die im Vorwärtsschritt berechneten Größen verwendet.
    - Folgende dynamische Größen werden bestimmt:
        * Auftretende Kraft F_i im Schwerpunkt von Armelement i
        * Auftretender Drehimpuls N_i im Schwerpunkt von Armelement i
        * Kraftvektor f_i ausgeübt von Armelement $i-1$ auf Armelement i
        * Momentvektor n_i ausgeübt von Armelement $i-1$ auf Armelement i
        * Skalares Drehmoment τ_i am i-ten Gelenk
    - Verwendung des Kraft-Momenten-Satzes: $F_i = m_i \cdot \dot{v}_i = f_i - f_{i+1}$
    - und des Drehimpulssatzes $N_i = I_i \cdot \dot{\omega}_i = n_i - n_{i+1} + (-s_i - p_i) x F_i - p_i x f_{i+1}$
    - Rekursive Berechnung der dynamischen Größen des Armelementes i

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

6 Regelung von Robotersystemen

6.1 Einführung

Regelungstechnik

- Regelungstechnik: Lehre von der selbsttätigen, gezielten Beeinflussung dynamischer Prozesse während des Prozessablaufs

- Regelungstechnische Grundsituation: Forderung nach selbsttätiger, gezielter Beeinflussung bei unvollständiger Systemkenntnis, insbesondere bei Einwirkung von Störungen
- Methoden der Regelungstechnik sind allgemeingültig, d. h. unabhängig von der speziellen Natur der Systeme
- Aufgabe: Der Ausgangsgröße eines dynamisches Systems soll mittels der Stellgröße ein Sollverhalten, d. h. ein gewünschtes Verhalten aufgeprägt werden, und zwar gegen den Einfluss einer Störgröße, dir nur unvollständig bekannt ist.
- Prinzip der Lösung: Die Strecke ist laufend zu beobachten und mit der so gewonnenen Information ist die Stellgröße derart zu verändern, dass trotz der Strörgrößeneinwirkung die Ausgangsgröße an den gewünschten Verlauf (Sollverlauf) angeglichen wird. Eine Anordnung, die dies bewirkt, heißt Regelung.

Wirkungsweise der Regelung

- Die Führungsgröße wird entsprechend eingeregelt, d.h. die Regelgröße folgt der Führungsgröße
- Die Regelung ist ein Wirkungskreislauf: Regelkreis
- Dabei entscheidend: Umkehr der Wirkungsrichtung im Soll-Istwert-Vergleich

Definition: Regelung
Unter einer Regelung versteht man eine Anordnung, durch welche bei unvollständig Bekannter Stecke, insbesondere unvollständiger Kenntnis der Störgröße, die Regelungsgröße, 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:

- Strukturbild einer Regelung
  - Aus den physikalischen Gesetzen ermittelt man Gleichungen zwischen zeitveränderlichen Größen des Systems
  - Die zeitveränderlichen Größen und ihre Gleichungen werden durch geeignete Symbole veranschaulicht
  - Ein Block des Strukturbildes ordnet jedem Zeitverlauf der Eingangsgröße eindeutig ein Zeitverlauf der Ausgangsgröße zu, und wirkt somit als Übertragungsglied

6.2 Grundlagen der Regelung

6.2.1 Grundlegende Regelkreise

Geschwindigkeitsregelung:

- Im Gelenkraum: Kontinuierliche Vorgabe von Gelenkgeschwindigkeiten
- Proportionale Reglung mit Faktor K_p: $\dot{\theta}_r(t) = K_p \cdot (\theta_v(t) - \theta(t))$

- Nachteil: Wenn $\theta_d = 0$ bewegt sich das Gelenk nicht

Vorsteuerung

- $\dot{\theta}_r(t) = K_p \cdot (\theta_v(t) - \theta(t)) + \dot{\theta}_d(t)$

6.2.2 Laplace-Transformation

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

- Rechenvereinfachung:
  - Differential- und Integralausdrücke werden zu algebraischen Ausdrücken ersetzt
- Gleichungslösung im Frequenzbereich statt im Zeitbereich
- Integral muss konvergieren - erfüllt für lineare $f(t)$
- Ableitungsfunktion $\mathscr{L} = \int_0^\infty e^{-st}\frac{df}{dt}dt = e^{-st}f(t)|_0^\infty - \int_0^\infty -s \cdot e^{-st}f(t)dt$
- Assumption: $\lim_{t\to\infty} e^{-st}f(t) \to 0$: $\mathscr{L}[\dot{f}(t)] = s\int_0^\infty e^{-st}f(t)dt - f(0) = s \cdot f(s) - f(0)$
- Impulsfunktion (Dirac-Impuls) $\delta(t)$: $\mathscr{L}[\delta(t)] = \int_0^\infty \delta(t) \cdot e^{-st}dt = 1$
- Sprungfunktion $\sigma(t)$: $\int_0^\infty \sigma(t) \cdot e^{-st}dt = \int_0^\infty \cdot e^{-st}dt = -\frac{1}{s} \cdot e^{-st}|_0^\infty$ mit $s = \delta + j\omega \implies e^{-st} = e^{-(\delta+j\omega)t} = e^{-\delta t} \cdot e^{-j\omega t} = e^{-\delta t} \cdot (cos\omega t - jsin\omega t)$
- Regeln:
  - Linearitätssatz $L\{\alpha f_1(t) + \beta f_2(t)\} = \alpha f_1(s) + \beta f_2(s)$
  - Faltungssatz $L\{f_1(t) * f_2(t)\} = f_1(s) * f_2(s)$
  - Grenzwertsatz $f_1(t=0) = \lim_{s\to\infty} s * f(s)$
  - Differentiationssatz $L\{\frac{d}{dt}f(t)\} = sF(s)$
  - Integrationssatz $L\{\int f(t)dt\} = \frac{1}{s}F(s)$
  - Verschiebung $L\{f(t-\tau)\} = e^{-\tau s}F(s)$
  - $L\{e^{\alpha t}\} = \frac{1}{s-\alpha}$, $L\{t^n\} = \frac{n!}{s^{n+1}}$ mit $n = 1, 2, ...$
  - $L\{sin(\alpha t)\} = \frac{\alpha}{s^2+\alpha^2}$, $L\{cos(\alpha t)\} = \frac{s}{s^2+\alpha^2}$

6.2.3 Übertragungsglieder

- Häufig Blöcke von folgendem Typ:
  - Lineares zeitinvariantes Übertragungsglied (LZI-Glied)
  - Im komplexen s-Bereich: $Y(s) = G(s) * U(s)$
  - Im Zeitbereich: $y(t) = g(t) * u(t) = \int_0^t g(t-\tau) * u(\tau) * d\tau$

Elementare Übertragungsglieder

- P-Glied (Proportionalglied): $y(t) = K * u(t)$
- I-Glied (Integrierglied): $y(t) = K * \int_0^t u(\tau) d\tau$
- D-Glied (Differenzierglied): $y(t) = K * \dot{u}(t - T_t)$
- T_t-Glied (Totzeit-Glied): $y(t) = K * u$
- S-Glied (Summenglied): $y(t) = \pm u_1(t) \pm u_2(t)$
- KL-Glied (Kennlinienglied): $y(t) = K * F(u(t))$
- M-Glied (Multiplizierglied): $y(t) = K * u_1(t) u_2(t)$

PID-Regelung

- Proportional-Integral-Derivative Controller: $\tau = K_p \theta_d + K_i \int \theta_d(t) dt + K_d \dot{\theta} e$

PID-Regler (und Unterklassen)

- sehr verbreitet, da für nahezu alle Prozesstypen geeignet, robust und mit geringem Aufwand realisierbar
- Charakterische Gleichung: $u(t) = K_p(e(t) + \frac{1}{T_N} \int_0^1 e(\tau) d\tau + T_V \frac{d}{dt} e(t))$

6.2.4 Stabilität einer Regelung

Drehmomentregelung

- Dynamisches Robotermodell wird in die Regelung miteinbezogen
- Dynamische Gleichung für 1DoF-Arm: $\tau = M\ddot{\theta} + mgrcos(\theta)$
- Vereinfachte Annahme: Viskose Reibung (Reibkoeffizient b): $\tau = M\ddot{\theta} + mgrcos(\theta) + b\dot{\theta}$
- PID-Regler: $\tau = K_p \theta_d + K_i \int \theta_d(t) dt + K_d \dot{\theta}_e$

Stabilität

- Ziel: Stabiles System, d.h. Regelabweichung geht mit der Zeit
  - 1. Annahme: $K_i = 0; g = 0$ -> Roboter bewegt sich in horizontaler Ebene
  - Gleichungen des PID-Reglers einsetzen (θ_d: Sollwert „desired value “) $\rightarrow M\ddot{\theta} + b\dot{\theta} = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta})$
  - 2. Annahme: Festpunktregler, d.h. Zielposition halten bei konstantem $\theta_d(\dot{\theta}_d = \ddot{\theta}_d = 0 -> \theta_d = \theta_v - \theta; \dot{\theta}_d = -\dot{\theta}; \ddot{\theta} = -\ddot{\theta})$: $\ddot{\theta}_d + \frac{(b+K_d)\dot{\theta}_d}{M} \frac{K_p}{M} \theta_e = 0 \rightarrow \ddot{\theta}_e + 2\zeta\omega_n \dot{\theta}_d + \omega_n^2 \theta_d = 0$

- Lösung des DGL mit Laplace-Transformation
  $a_n\theta_d^{(n)} + a_{n-1}^{(n-1)} + ... + a_2\ddot{\theta}_d + a_1\dot{\theta}_d + a_0\theta_d = 0$
  $\rightarrow a_n s^n + a_{n-1}s^{n-1} + ... + a_2 s^2 + a_1^s + a_0 = 0$
  $\ddot{\theta}_d + 2\zeta\omega_n\dot{\theta}_d + \omega_n^2\theta_e = 0$
  $\rightarrow s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$

- 3 mögliche Lösungstypen
  - $\zeta > 1$: aperiodische Lösung: Zielwert wird ohne zu schwingen (langsam) über die Exponentialfunktion erreicht
  - $\zeta = 1$: aperiodischer Grenzfall: Der Zielwert wird schnell erreicht, und das System überschwingt gerade nicht
  - $\zeta < 1$: gedämpfte Schwingung: Das System überschwingt.

6.2.5 Testfunktionen

- Impulsfunktion
- Sprungfunktion
- Anstiegsfunktion
- Harmonische Funktion
- Wird die Ausgangsgröße auf die Eingangsgröße bezigen, so entsteht die normierte Sprungantwort $h(t)$ (Übertragsfunktion der Regelstrecke).

6.3 Regler

- Zustandsregler
  - Verbessertes Regelverhalten: Regelabweichung und idealerweise alle Zustandsgrößen der Regelstrecke stehen zur Verfügung
  - Regeltechnische Behandlung: Mehrgrößensysteme, nichtlineare und zeitvariable Übertragungssysteme
- Kaskadenregler
  - Manipulator = Mehrgrößensystem: Unabhängige lineare Einzelkreise der einzelnen Gelenke
- Adaptive Regler
  - Lageabhängige und somit zeitveränderliche Systemteile werden als Parameterschwankungen aufgefasst (Beispiel: Überschallflugzeuge mit Reglern je nach aktueller Mach-Zahl)

6.4 Regelungskonzepte für Manipulatoren

Unter dem Begriff Regelung von Manipulatoren fällt nicht nur die klassische Positionsregelung, sondern auch die Einbeziehung weiterer Umwelteinflüsse. Besondere Stellung nimmt dabei die Regelung von Kräften und Drehmomenten ein.

Ausgangspunkt: Dynamikmodell

- Bei Bewegungen wirken aufgrund der Masseträgheit des Manipulators Gravitations-, Zentrifugal-, Coriolis und Reibungskräfte/momente auf die Gelenke.
  $Q = M(q)\ddot{q} + n(\dot{q}, q) + g(q) + R\dot{q}$

Regelung im Gelenkwinkelraum

- Koordinatentransformation (Soll-Trajektorie im Gelenkwinkelraum)
- Aus den Gelenkwinkel-Sollwerten und den gemessenen Gelenkwinkeln werden Stellgrößen für die Gelenkantriebe generiert

Regelung im kartesischen Raum

- Höhere Komplexität des Regelungsalgorithmus
- Direkte, gezielte Beeinflussung der einzelnen Raumkoordinaten
- Exakte Systemmodellierung setzt a priori die exakte Kenntnis des Dynamikmodells und der Umgebung des Roboters voraus
- Kraft-/Positionsregelung zur Ausführung von Aufgaben, die Interaktionskräfte berücksichtigen müssen (hybride Kraft-/Positionsregelung, Impedanz Regelung)

Kraft-Positionsregelung

- Grundlegendes Problem:
  - Positionen und Kräfte sind eng miteinander verknüpft
  - Steht der Roboter in Kontakt zur Umgebung, so bedeutet jede Positionsänderung auch eine Kraftänderung und umgekehrt
- Allgemeine Methode zur Lösung des Problems:
  - Aus der Beschreibung der auszuführenden Aufgabe resultieren natürliche Randbedingungen. Künstliche Randbedingungen werden zusätzlich eingeführt, um den Bewegungsablauf vollständig zu beschreiben.

Hybride Kraft-/Positionsregelung

- Reihe Kraft- oder Positionsregelung für jede kartesische Bewegungsrichtung des Arms
- Achsen Regelung: Enschieben einer Box

  - kartesische Achsen (relativ zu Kraft-Momenten-Sensor im TCP)
  - Achsen sind unabhängig geregelt
  - Jeder Achsen-Regler ist ein PID-Regler
  - Bei dieser Anwendung ist der i-Anteil besonders relevant
- Probeleme in der Praxis: Reibung muss berücksichtigt werden, beim Regeln auf 0 kann bei hoher Reibung sein, dass sich nicht bewegt wird.
- Achsenüberlagerte Regelung: Spiralsuche
  - kartesische Achsen (relativ zu Kraft-Momenten-Sensor im TCP)
  - Hier reicht reiner hybrider Kraft-Positionsregler nicht mehr aus, da auf zwei Achsen, nämlich X-Y, sowohl Kraft- als auch Position geregelt werden müssen
- Offene Fragestellung: Programmierung Reglerprogramm
  - Schnell
  - Intuitiv
  - Robust
  - Einfache Wartung
  - Wiederverwendbar

Impedanz-Regelung

- Regelt die dynamische Beziehung zwischen Kraft und Position im Kontaktfall
- Idee:
  - Die Interaktion zwischen einem Roboter und der Umwelt verhält sich wie ein Feder-Dämpfer-Masse System
  - Kraft f und Bewegung können über die Feder-Dämpfer-Masse-Gleichung direkt in Zusammenhang gebracht werden.
- Die Impedanz kann über Steifigkeit, Dämpfung und Trägheit beeinflusst werden

7 Bahnsteuerung

7.1 Grundlagen der Bahnsteuerung

Bewegungen eines Roboters werden aufgefasst als

- Zustandsänderungen
  - über der Zeit
  - relativ zu einem stationären Koordinatensystem
- mit Einschränkungen durch

  - Zwangsbedingungen
  - Gütekriterien
  - Neben- und Randbedingungen

Problem:

- Gegeben
  - S_{Start}: Zustand zum Startzeitpunkt
  - S_{Ziel}: Zustand zum Zielzeitpunkt
- Gesucht S_i: Zwischenzustände (Stützpunkte), damit die Trajektorie glatt und stetig wird.
- Zustände können dargestellt werden im
  - Gelenkwinkelraum (Konfigurationsraum): $\mathbb{R}^n$
  - Kartesischen Raum (Arbeitsraum): $\mathbb{R}^3$,SO(3)
- Bahnsteuerung im Gelenkwinkelraum ist näher an der Ansteuerung der Teilsysteme des Roboters (Gelenke, Sensorik)
- Bahnsteuerung im kartesischen Raum ist näher an der zu lösenden Aufgabe (bei Steuerung im kartesischen Raum ist das Lösen der inversen Kinematik nötig)

Bahnsteuerung im Gelenkwinkelraum

- Bahnsteuerung als Funktion der Gelenkwinkelzustände
  - Verlauf der punktweise in Gelenkwinkel spezifizierten Bahn muss im kartesischen Raum nicht notwendigerweise definiert sein
- Abfahren dieser punktweise spezifizierten Trajektorien
  - Asynchron: Steuerung der Achsen unabhängig voneinander (Anwendung: Punktschweißen, Handhabungsaufgaben)
  - Synchron: achsinterpolierte Steuerung (Bewegung aller Achsen beginnt und endet zum gleichen Zeitpunkt, Anwendung: Bahnschweißen, Lackieren, Montieren)

Kartesischer Raum

- Angabe der Trajektorie erfolgt als Funktion der Zustände des Roboters (z. B. mit Beschreibungsvektor des TCP(Position, Geschwindigkeit, Beschleunigung))
- Endeffektor folgt in Lage und Orientierung einer definierten Bahn
- Bahntypen: lineare Bahnen, Polynombahnen, Splines

Teach-In

- Anfahren markanter Punkte der Bahn mit manueller Steuerung
  - Teach Box, Teach Panel, weitere: Spacemouse, Teach-Kugel
- Funktionalitäten einer Teach Box:
  - Einzelbewegung der Gelenke
  - Bewegung des Effektors in 6 Freiheitsgraden
  - Speichern/Löschen von Anfahrpunkten
  - Eingabe von Geschwindigkeiten
  - Eingabe von Befehlen zur Bedienung des Greifers
  - Starten/Stoppen ganzer Programme
- Speichern der Gelenkwerte
- Ergänzung der gespeicherten Werte um Parameter wie Geschwindigkeit, Beschleunigung usw.
- Anwendung: Fertigungsindustrie (Punktschweißen, Nieten), Handhabungsaufgaben (Pakete vom Fließband nehmen)

Playback

- Roboter im Zero-Force-Control Modus (Roboter kann durch den Bediener bewegt werden)
- Abfahren der gewünschten Bahn
- Speichern der Gelenkwerte (automatisch mit Abtastfrequenz oder manuell durch Tastendruck)
- Anwendung: mathematisch schwer beschreibbare Bewegungsabläufe, Integration der handwerklichen Erfahrung, typische Einsatzbereiche sind Lackieren oder Kleben
- Vorteile: schnell für komplexe Bahnen, intuitiv
- Nachteile: schwere Roboter schwierig zu bedienen, wenig Platz in engen Fertigungszellen für Bediener, schlechte Korrekturmöglichkeiten, Optimierung und Kontrolle durch Interpolationsmethoden schwierig

7.3 Interpolationsarten

7.3.1 Punkt-zu-Punkt (PTP)

- Roboter führt Punkt-zu-Punkt-Bewegung aus
- Vorteile:
  - Die Berechnung der Gelenkwinkeltrajektorie ist einfach
  - Keine Probleme mit Singularitäten
- Ablauf der Steuerung
  - Fahrzeit t_e
  - Beschleunigungszeit t_b
  - Beginn der Bremszeit t_v
    $s(t_e) = s_e = |q_z - s_{st}|$
    $\dot{s}(t_e) = v(t_e) = 0$

Interpolation für PTP mit Rampenprofil

- Einfache Art zur Berechnung des Bahnparameter $s(t)$
- Sprungförmige Auschaltung der Beschleunigung (ruckartig)
- Kann zu Eigenschwingungen von mechanischen Teilen führen

Interpolation für PTP mit Sinoidenprofil

- Weichere Bewegung durch Verwendung einer sinusförmigen Zeitfunktion
- Längere Beschleunigungs- und Bremsphase als beim Rampenprofil
- Roboter wird weniger beansprucht
- Bestimmung der Kurvenparameter für die Phase
  - Beschleunigung
  - Gleichförmige Bewegung
  - Bremsvorgang

Synchrone PTP-Bahnen

- Vorgehen bei synchronen PTP-Bahnen
  - Bestimme für jedes Gelenk i die PTP-Parameter $s_{e,i}$, $v_{m,i}$, $b_{m,i}$ und Fahrtzeit $t_{e,i}$
  - Bestimme $t_e = t_{e,max} = max(t_{e,i})$
  - Setze $t_{e,i} = t_e$ für alle Gelenke
  - Bestimme die neuen maximalen Geschwindigkeiten für alle Gelenke

- Alle Gelenke beginnen und beenden ihre Bewegung gemeinsam

Asynchrone PTP-Bahnen

- Jedes Gelenk wird sofort mit der maximalen Beschleunigung angesteuert.
- Jede Gelenkbewegung endet unabhängig von den anderen.

Vollsynchrone PTP-Bahnen

- Zusätzliche Berücksichtigung der Beschleunigungs- und Bremszeit
- Bessere Annhäherung der Start- und Zielpunkte im kartesischen Raum
- Bestimmung Leitachse mit t_e und $t_b \rightarrow t_v = t_e - t_b$
- Bestimmung der Geschwindigkeit und Beschleunigung der anderen Achse
- Nachteil: Beschleunigung jeder Achse wird vorgegeben

7.3.2 Linear- und Zirkularinterpolation

Linearinterpolation:

- Orientierung in Eulerwinkel
- Berechnung von $s_\omega(t)$ mit Rampen- oder Sinoidenprofil
- Angleich der Fahrzeiten t_{ep} (Position) und $t_{e\omega}$ (Orientierung)
- Analog zur Anpassung der Geschwindigkeiten bei synchronen PTP

CP (Continuous Path)

7.3.3 Splineinterpolation

- Die Endbedingungen der Teiltrajektorie $j-1$ (Richtung, Geschwindigkeit, Beschleunigkeit) und die Anfangsbedingungen der Teiltrajektorie werden aneinander angeglichen
- Teiltrajektorien werden separat beschrieben

7.4 Approximierte Bahnsteuerung

Bauninterpolation: Die ausgeführte Baun verläuft durch alle Stützpunkte der Trajektorie
Baunapproximation: Die Kontrollpunkte beeinflussen den Baunverlauf und werden approximiert.

PTP und CP mit Überschleifen

- Geschwindigkeitsüberschleifen

  - Beginn, wenn die Geschwindigkeit einen festgelegten Minimalwert unterschreitet.
  - Nachteil: Abhängig vom Geschwindigkeitsprofil.

- Positionsüberschleifen
  - Beginn, wenn der TCP in die Überschleifkugel eintritt
  - Außerhalb der Überschleifkugel wird Bahn exakt eingehalten
  - Vorteil: Gut kontrollierbar

Bézierkurven

- Im Unterschied zu kubischen Splines verlaufen Bézierkurven nicht durch alle Stützpunkte, sondern nur von ihnen beeinflusst.
- Basisfunktion:
  - $P(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i$
  - $B_{i,n}(t) = \overset{n}{i} t^i (1-t)^{n-i}$

  $B_{i,n}(t)$: i-tes Bernsteilpolynom n-ten Grads
- Berechnung beliebiger Zwischenstellungen

De-Casteljau-Algorithmus

- Annäherung an die Bézierkurve: Effiziente Berechnung
  Näherungsdarstellung von Bézierkurven durch einen Polygonzug
- Iterative Berechnung: Kann auch für große n effizient berechnet werden

8 Bewegungsplanung

8.1 Motivation

Erzeugen einer kollisionsfreien Trajektorie unter Berücksichtigung verschiedener Ziele und Einschränkungen

8.2 Grundlagen der Bewegungsplanung

8.2.1 Problemstellung

- Gegeben
  - Konfigurationsraum C
  - Startkonfiguration $q_{start} \in C$
  - Zielkonfiguration $q_{ziel} \in C$

- Gesucht
  - Stetige Trajektorie $\tau : [0,1] \implies C$ mit
    - $\tau(0) = q_{start}$
    - $\tau(1) = q_{ziel}$
  - Unter Berücksichtigung von
    - Gütekriterien
    - Neben- und Randbedingungen
    - Zwangsbedingungen

8.2.2 Definitionen

- Konfiguration: Eine Konfiguration $q \in C$ beschreibt den Zustand eines Roboters
  - als Lage und Orientierung im euklidischen Raum oder
  - als Gelenkwinkelvektor im Gelenkwinkelraum,
- Konfigurationsraum: Der Konfigurationsraum C eines Roboters R ist der Raum aller möglicher Konfigurationen von R.
- Arbeitsraumhindernis: Ein Arbeitsraumhindernis H ist der Raum, welcher von einem Objekt im Arbeitsraum eingenommen wird.
- Konfigurationsraumhindernis: Ein Konfigurationsraumhindernis C_H ist die Menge aller Punkte des Konfigurationsraumes C, welche zu einer Kollision mit dem Hindernis H führen.
- Hindernisraum: Der Hindernisraum C_{obs} ist die Menge aller Konfigurationshindernisse.
- Freiraum
  - Der Freiraum ist die Menge aller Punkte aus C, welche nicht im Hindernisraum C_{obs} liegen
  - Aufwand für die Berechnung des Freiraums: $O(m^n)$
  - Verwendung approximativer Verfahren zur vereinfachten Repräsentation von C_{free}
- Umweltmodellierung
  - Exakt: Beispielsweise über CSG (constructed-solid-geometry), in Form einer algebraischen Beschreibung
  - Approximiert: Die Umwelt wird durch Näherungen beschrieben (Kuben, verallgemeinerte Zylinder, Polyeder,...)
- Planungsverfahren

  - Vollständig: liefert immer eine korrekte Lösung, kann ermitteln, ob keine Lösung existiert
  - Probabilistisch vollständig: Falls eine Lösung existiert geht die Wahrscheinlichkeit, dass eine Lösung gefunden wird, bei fortschreitender Zeit gegen 1, existiert keine Lösung, terminiert das Verfahren nicht

8.2.3 Begriffsbildung

- Pfadplanung
  - Starres Objekt (z. B. mobiler Roboter, autonomes Fahrzeug)
  - 2D Problem (Position: x,y)
  - 3D Problem (Position: x,y; Rotation α) -> Piano Mover's Problem
- Bewegungsplanung
  - Mehrkörpersystem (z. B. Roboterarme, Systeme mit mehreren Robotern)
  - Hochdimensionale Problemstellungen
- Randbedingungen (auch Zwangsbedingungen)
  - Globale Randbedingungen: Limitieren den gültigen Konfigurationsraum (z. B. aufrechte Position des Endeffektors, maximale Motorströme, etc.)
  - Lokale Randbedingungen: Schränken die Übergänge zwischen Konfigurationen ein (z. B. nicht-holonome Fahrzeuge, max. Geschwindigkeit/Beschleunigung)
- Komplexität: Allgemeine Planungsaufgaben sind PSPACE-vollständig
- Vollständiger Algorithmus: Ein vollständiger Algorithmus findet für spezielle Planungsprobleme mindestens eine Lösung oder erkennt in endlicher Zeit, dass keine Lösung existiert.
- Randomisierter Algorithmus: Randomisierte Algorithmen verwenden Zufallsgrößen, um den Ablauf zu steuern, wobei oft heuristische Annahmen genutzt werden, um die Berechnung zu beschleunigen.
- Auflösungsvollständiger Algorithmus: Ist ein approximativer Algorithmus für eine diskretisierte Problemstellung vollständig, gehört er zur Klase der auflösungsvollständigen Algorithmen
- Probabilistisch-vollständiger Algorithmus: Ein probabilistisch-vollständiger Algorithmus findet mindestens eine Lösung falls sie existiert. D. h. die Wahrscheinlichkeit, dass eine Lösung gefunden wird, konvergiert mit fortlaufender Zeit gegen eins. Allerdings kann mit probabilistisch-vollständigen Algorithmen nicht ermittelt werden, ob keine Lösung existiert.

8.2.4 Problemklassen

- Klasse a)
  - Bekannt: vollständiges Umweltmodell, vollständige Neben-, Rand- und Zwangsbedingungen
  - Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
- Klasse b)
  - Bekannt: unvollständiges Umweltmodell, unvollständige Neben-, Rand- und Zwangsbedingungen
  - Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
  - Problem: Kollision mit unbekannten Objekten
- Klasse c)
  - Bekannt: zeitvariantes Umweltmodell (bewegliche Hindernisse)
  - Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
  - Problem: Hindernisse in Ort und Zeit variant
- Klasse d)
  - Bekannt: kein Umweltmodell
  - Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
  - Problem: Kartographieren
- Klasse e)
  - Bekannt: zeitvariantes Umweltmodell
  - Gesucht: Trajektorie zu einem beweglichen Ziel (Rendezvous-Problem)
  - Problem: Zielzustand in Ort und Zeit beweglich

8.3 Pfadplanung für mobile Roboter

- Gegeben
  - 2D Weltmodell (z. B. Straßenkarte)
  - Start- und Zielposition q_{start} und q_{ziel}
- Gesucht: Günstigste Verbindung von q_{start} nach q_{ziel}
- Ansatz:
  - Konstuiere ein Netz W von Wegen in C_{free}
  - Bilde q_{start} und q_{ziel} auf die nächsten Knoten q'_{start} und q'_{ziel} in W ab
  - Suche in W einen Weg von q'_{start} nach q'_{ziel}

  - Finde einen Weg zwischen q_{start} und q'_{start} sowie zwischen q'_{ziel} und q_{ziel}
- Konstruktion des Wegenetzes W: Voronoi-Diagramme, Sichtgraphen, Zellzerlegung
- Suche in W: Euklidischer Abstand, Potentialfelder, Baumsuche, A*

8.3.1 Voronoi-Diagramme

- Visualisiert die Zerlegung eines Raumes in Regionen basierend auf vorgegebenen Punkten.
- Eine Region ist definiert als die Menge aller Punkte, deren Abstand zum Zentrum geringer ist als zu allen anderen Zentren.
- Alle Punkte an der Grenze zwischen zwei Regionen besitzen den gleichen Abstand zum eigenen und zum benachbarten Zentrum.

Vorteile:

- Maximaler Abstand zu Hindernissen
- Ein Roboter kann mit Hilfe von Abstandssensoren leicht prüfen ob der richtige Weg abgefahren wird

Nachteile:

- In der Regel ist der Weg nicht der kürzeste.
- Bei wenigen Hindernissen werden nur wenige Wege generiert.

8.3.2 Sichtgraphen

Verbinde jedes Paar von Eckpunkten auf dem Rand von C_{free} durch ein gerades Liniensegment, wenn das Segment kein Hindernis schneidet und verbinde q_{start} und q_{ziel} analog.

Vor-und Nachteile:

- Vorteile:
  - Wenn ein Weg gefunden ist, ist es auch der kürzeste Weg.
  - Methode ist exakt, wenn nur zwei translatorische Freiheitsgrade existieren und sowohl Roboter als auch Hindernisse durch konvexe Polygone dargestellt werden können.
- Nachteile:
  - Wege sind nicht zwingend kollisionsfrei, da Hinderniskanten auch Wegsemente sein können. Abhilfe durch Erweiterung der Hindernisse.
- Methode auch im $\mathbb{R}^3$ anwendbar, jedoch sind die gefunden Wege i. A. keine kürzesten Wege mehr.

8.3.3 Zellzerlegung

- Vorgehen:
  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
- Es gibt zwei Zerlegungsarten:
  - Exakte Zerlegung
  - Approximative Zerlegung

8.3.4 Potentialfelder

- Der Roboter bewegt sich unter dem Einfluss von Kräften, welche ein Potentialfeld auf ihn ausübt
- Definition:
  - Ein Potentialfeld U ist eine Skalarfunktion über dem Freiraum $U : C_{free} \rightarrow \mathbb{R}$
  - Die Kraft in einem Punkt q des Potentialfeldes ist der negative Gradient in diesem Punkt $F(q) = -\nabla U(q)$
- Abstoßendes Potenzial
  - Hindernisse erzeugen ein abstoßendes Potential
  - In großen Abstand zu Hindernissen ($> p_0$) soll der Roboter nicht beeinflusst werden
- Anziehendes Potential: Es soll möglichst nur ein Minimum in q_{ziel} geben
- Lineare Funktion der Distanz zum Ziel: $F_{ab}(q) = -\nabla U_{ab}(q) = -\zeta \frac{q-q_{ziel}}{||q-q_{ziel}||}$, wird für kleine Distanzen sehr groß
- Quadratische Funktion der Distanz: $F_{an}(q) = -\nabla U_{an}(q) = -\zeta(q - q_{ziel})$
- Oft wird die Kombination auss linearer und quadratischer Funktion verwendet
  - Lineare Funktion, wenn weit vom Ziel entfernt
  - Quadratische Funktion, wenn nah am Ziel
- Lokale Minima: 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 solch einem lokalen Minimum steckenbleiben.
- Maßnahmen:
  - U_{an} und U_{ab} so definieren, dass U kein lokales Minimum hat, außer in q_{ziel}

Baumsuche:

- Anwendungsfall:
  - Mobiler Roboter
  - 2D-Arbeits- und Konfigurationsraum
- Darstellung des Konfigurationsraums als Quadtree
  - Rekursive Unterteilung des Konfigurationsraums in Kacheln
  - Kacheln sind entweder frei oder ein Hindernis
- Bewegungsplanung:
  - Kacheln finden, in denen sich Start- bzw. Zielkonfiguration befinden
  - Benachbarte freie Kacheln des Baums vom Start zum Ziel verbinden
  - Kollisionsfreie Routenplanung durch freie Kacheln

8.3.5 A*

- Kürzester Pfad von Start nach Ziel
- A* ist einer der beliebtesten Algorithmen zur Routenplanung
- Kostenfunktion ist $f(x) = g(x) + h(x)$
  - $g(x)$ entspricht Kosten von Start-Knoten nach Knoten x
  - $h(x)$ entspricht geschätzten Kosten von Knoten x nach Ziel-Knoten
- Unterteile Knoten in
  - Open Set: Enthält zu Beginn alle Knoten außer dem Start-Knoten
  - Current Set: Enthält zu Beginn nur den Start-Knoten
  - Closed Set: Enthält bereits untersuchte Knoten, zu welchen der kürzeste Weg bekannt ist, zu Beginn leer.
- Knoten aus dem Current Set mit gerinstem $f(x)$ wird als nächsten untersucht.
- Wenn ein Knoten x abschließend untersucht wurde, dann werden die Nachfolgeknoten in das Open Set eingefügt und x in das Closed Set aufgenommen.
- Algorithmus terminiert, wenn der Ziel-Knoten abschließend untersucht worden ist.
- Wenn Open Set leer ist, terminiert der Algorithmus ohne Lösung.

Eigenschaften:

- Findet eine optimale Lösung
- Heuristik h darf die minimalen Kosten, das Ziel zu erreichen, nicht überschätzen.

- A* ist auch optimal effizient für jede (zulässige) Heuristik
- Wenn $h(x) = 0 \forall x$: Dijksta's algorithm, d. h. $f = g$
  - Greedy Algorithmus: geachtet die Kosten nicht
  - Ignoriert Schätzung der Distanz zum Zielknoten
  - Besucht mehr Knoten als notwendig

8.4 Bewegungsplanung für Manipulatoren

- Pfadplanung
  - Starres Objekt (z. B. mobiler Roboter, autonomes Fahrzeug)
  - 2D Problem (Position: x,y)
  - 3D Problem (Position: x,y; Rotation α) -> Piano Mover's Problem
- Bewegungsplanung
  - Mehrkörpersystem (z. B. Roboterarme, Systeme mit mehreren Robotern)
  - Hochdimensionale Problemstellungen
- Randbedingungen (Zweckbedingungen)
  - Globale Randbedingungen: Limitieren den gültigen Konfigurationsraum (z.B. aufrechte Position des Endeffektors, maximale Motorströme, etc.)
  - Lokale Randbedingungen: Schränken die Übergänge zwischen Konfigurationen ein (z. B. holonome Fahrzeuge, max. Geschwindigkeit/Beschleunigung)

8.4.1 Probabilistic Roadmaps (PRM)

- PRM basiert auf Approximation des Freiraumes durch den Graphen (Roadmap) -> Effizienter als die Erzeugung einer expliziten Repräsentation des Freiraumes
- PRM ist probabilistisch: Zufallsgesteuerte Erzeugung durch das Sampling
- Algorithmus:
  1. Vorverarbeitung: Erzeugung einer Kollisionsfreien Straßenkarte (Graph)
  2. Anfrage: Verbinde q_{start} und q_{ziel} mit dem Graphen und suche einen Weg von q_{start} nach q_{ziel} durch den Graphen
- Vorverarbeitung:
  - Zufällige Erzeugung von kollisionsfreien Stichproben (Sampling)
  - Lokale Planung: Stichproben werden über kollisionsfreie Pfade miteinander verbunden
- Anfrage

  - Verbinde q_{start} und q_{ziel} mit dem Wegenetz
  - Suche im Graphen (z.B. mit A*)
- Konstruktion des Graphen
  - N: Anzahl der Knoten im Graphen
  - R: PRM, Graph
  - Algorithmus:
    * Erzeugung von N zufälligen Konfiguration in C_{free}
    * Einfügen der erzeugten Konfiguration als Knoten in R
    * Für jeden Knoten $v_i \in R$
      · Finde die k nächsten Nachbarn von v_i aus R: $N(v_i)$
      · Für jeden Knoten $v \in N(v_i)$: Wenn es einen (neuen) kollisionsfreien Pfad von v nach v_i gibt, dann füge die Kante(v, vi) in R ein
  - Ergebnis: R
- Eigenschaften
  - Einmalige Konstruktion des Graphen: Anfragen können effizient bearbeitet werden
  - Randomisierter Ansatz zur Konstruktion: Exponentieller Anstieg der Laufzeit mit der Dimension des Konfigurationsraums wird vermieden
  - Verwfahren hängt start vom verwendeten Sampling ab
    * Problem: Schmale Passagen zwischen Hindernissen
    * Lösungsansatz: Sampling in der Nähe von Hindernissen erhöhen
- Unterschiedliche Sampling-Strategien
  - Zufällig: Konfiguration wird zufällig generiert und auf Kollision geprüft
  - Grid: Konfigurationen werden mit diskreter Auflösung erzeugt, Auflösung einzelner Zellen wird hierarchisch bestimmt
  - Halton: Halton-Menge: Menge von Punkten, die ein Bereich besser abdeckt als Grid, basiert auf dem mathematischen Konzept der Diskrepanz
  - Zellenbasiert: Sampling in Zellen mit kleiner werdenden Ausmaßen, Zellgröße wird mit jeder Iteration verkleinert (z.B. auf 1/8)

8.4.2 Rapidly-exploring Random Trees (RRT)

- Algorithmus zur Einmalanfrage
  - Im Gegensatz zu PRMs
  - Keine Vorverarbeitung nötig

  - Keine Probleme mit sich verändernden Umgebungen

- Probabilistisch vollständiger, randomisierter Algorithmus
  - Keine Garantie, dass eine Lösung innerhalb eines Zeitlimits gefunden wird
  - Wenn eine Lösung existiert, wird sie gefunden (Laufzeit geht gegen Unendlich)
  - Terminiert nicht, wenn keine Lösung existiert
- Effizient für hochdimensionale Problemstellungen
- Erweiterung der klassischen RRT für spezifische Problemstellungen z.B. enge Durchgänge

Bidirektionale RRTs

- Es werden zwei Bäume aufgebaut
  - T_1 ausgehend von q_{start}
  - T_2 ausgehend von q_{ziel}
- Zufällig gewählte Punkte q_s erweitern beide Bäume über $q_{nn,1}$ und $q_{nn,2}$
- Eine Lösung ist gefunden, wenn beide Bäume mit q_s verbunden wurden

Nachbearbeitung

- Lösungen können durch Nachbearbeitung verbessert werden
  - Zufällige Wahl zweier Knoten im Lösungsweg
  - Falls die Verbindung kollisionsfrei ist, verbinde beide Knoten und lösche den dazwischenliegenden Knoten aus dem Lösungspfad
  - Erzeugt glattere Trajektorien

8.4.3 Erweiterungen

Contrained RRT:

- Bei der Bewegungsplanung müssen evtl. Nebenbedingungen (Constraints) erfüllt werden, z.B. gleichbleibende Orientierung des Endeffektors oder (statische) Stabilität eines zweibeinigen Roboters
- Problem: Nebenbedingungen können niederdimensionale Gebilde im Konfigurationsraum darstellen, z.B. die Menge aller Konfigurationen q, die eine Nebenbedingung erfüllen, bilden eine Ebene im dreidimensionalen Konfigurationsraum, sampling-basierte Ansätze können diese Nebenbedingungen prinzipiell nicht erfüllen.
- Lösungsansätze: Randomized Gradient Descent (RGD) oder First Order Retraction (FR)

- Idee: Projiziere eine Stichprobe q_s auf eine Konfiguration q'_s, die die Nebenbedingungen erfüllt
- Beispiel: eine Nebenbedingung NB bilde eine 2d Mannigfaltikeit
- Problem: Wie wird die Projektion durchgeführt? -> Randomized Gradient Descent oder First Order Retraction
- Randomized Gradient Descrent
  - Toleranzwert für Nebenbedingung α
  - Zufällige Bestimmung von n Nachbarn von q_s (in Hyperkugel mit Radius d_{max})
  - Falls die Distanz eines Nachbarn zu C_{NB} kleiner als die Distanz von q_s zu C_{NB}, ersetze q_s mit diesem Nachbarn
  - Wiederhole bis maximale Iterantionszahl erreicht oder die Distanz von q_s zu C_{NB} kleiner ist als α
- First Order Retraction
  - Toleranzwert für die Nebenbedingung: α
  - Jacobi-Matrix J liefert Richtungsinformationen
  - Berechnung wie bei Bestimmung der inversen Kintematik: $q'_s = q_s - J(q_s)^{\#}\Delta x_s$

RRT*

- Problem: RRTs finden Trajektorien, die üblicherweise nicht optimal sind
- RRT* optimiert den Suchbaum iterativ während der Suche: Mit ausreichender Zeit wird der optimale Pfad zwischen q_{start} und q_{ziel} gefunden => asymptotische Optimalität
- Optimierung des Suchbaums aufgeteilt in zwei Schritte
  - Ermittle zu jedem Knoten die Kosten
  - Rewiring des Suchbaums beim Hinzufügen neuer Knoten: Die Verbindungen des Suchbaumes werden in einer lokalen Umgebung des neuen Knoten optimiert.
- Nachteil:
  - Uni-direktionaler Ansatz
  - Längere Laufzeiten

Enge Passagen:

- Klassische RRTs bestimmen neue Punkte q_s durch gleichverteilte Zufallswahl im Konfigurationsraum C
- Ergebnis gleichverteilter Zufallswahl:
  - Viele eher uninteressante Stichproben

  - Wenige interessante Stichproben

- Klassische RRTs können viel Zeit benötigen, bis eine Lösung für einen Durchgang durch eine enge Passage gefunden wurde
- Hauptidee der Verfahren: Sampling ist deutlich günstiger als Kollisionsprüfung von Pfaden im Baum
- Dynamic Domain RRT
  - Problem: RRTs erkennen enge Passagen nicht und können nicht zielgerichtet sampeln
  - Ideal: Nur in sichtbarer Voronoi Region eines Knoten sampeln
  - Aber: Berechnung sichtbarer Voronoi Regionen ist aufwendig
  - Stattdessen: Approximation sichtbarer Voronoi Regionen durch Kugeln mit Radius r (Dynamic Domain)
  - Dynamic Domain RRT beschränkt in der Nähe von Hindernissen die Sampling Domäne eines Knotens auf dessen Dynamic Domain (DD)
    * Initial wird der DD-Radius r jedes Knotens auf ∞ gesetzt. Sampling findet in gesamter Voronoi Region des Knotens statt.
    * Wenn während des RRT-Erweiterungsschritts keine Verbindung zu einem Knoten hergestellt werden kann, wird dessen DD-Radius auf einen festgelegten Wert R reduziert. Knoten dieser Art werden Grenzknoten genannt, da sie an der Grenze von C_{free} und C_{obs} liegen.
  - Sampling: Ein Sample q_s wird verworfen, falls q_s außerhalb des DD-Radius seines nächsten Nachbarn liegt.
  - Bei engen Passagen werden häufige Kollisionsprüfungen vermieden und keine Expansionsversuche zu weit entfernten und unerreichbaren Knoten unternommen.

- Bridge Sampling
  - Idee: Wähle zielgerichtet Punkte in engen Passagen für die nächste Stichprobe
  - Vorgehen
    1. Wähle gleichverteilt einen zufälligen Punkt $q_1 \in C_{obs}$
    2. Wähle nach einer geeigneten Wahrscheinlichkeitsverteilung einen zweiten Punkt $q_2 \in C_{obs}$ in der Nähe von q_1
    3. Wenn der Mittelpunkt q_s zwischen q_1 und q_2 in C_{free} liegt, dann verwende ihn als neue Stichprobe für den RRT (oder die PRM)
    4. Wiederhole
  - Bridge Sampling erhöht die Stichprobendichte in interessanten Bereich des Konfigurationsraumes C (Interessant sind die Bereiche in der Nähe von Hindernissen, besonders in engen Passagen)

  - Bridge Sampling kann für RRTs und PRMs verwendet werden
  - Das zentrale Element des Verfahren, der Bridge Test, ist auch in hochdimensionalen Räumen effizient berechenbar

9 Greifplanung

9.1 Einleitung

- Die menschliche Hand
  - 27 Knochen
  - insgesamt 27 Bewegungsfreiheitsgrade.
    * 3 DoF flexion/extension pro Finger
    * 1 DoF abduction/adduction pro Finger
    * 5 DoF Daumen (3 DoF flexion/extension, 2 DoF abduction/adduction)
    * 6 DoF für die Handwurzel (Handfläche)
- Modellierung:
  - kinematisches Modell
  - flächenbasiertes Geometriemodell

Cutosky Griffaxanomie

- Griffaxanomie
  - Benchmark für die Evaluation von Roboterhänden
  - Vereinfachung der Griffsynthese
  - Grundlagen für das Design von Roboterhänden
  - Einsatz bei der autonomen Greifplanung
- Cutkosky Griffaxanomie
  - 16 einzelne Griffarten
  - Hierarchiebaum: Griffarten werden zu Gruppen zusammengefasst
  - Erste Ebene: Unterscheidung in Kraft- und Präzisionsgriffe

Greianalyse und Greifsynthese:

- Griff: Eine Menge von Kontaktpunkten auf der Oberfläche eines Objekts, die potentielle Bewegungen des Objekts untr dem Einfluss externer Kräfte einschränken/kompensieren
- Greifanalyse
  - Gegeben: Objekt und eine Menge von Kontaktpunkten

  - Gesucht: Aussagen zur Stabilität des Griffs unter Berücksichtigung von Nebenbedingungen

- Greifsynthese
  - Gegeben: Objekt und eine Menge von Nebenbedingungen
  - Gesucht: Eine Menge von Kontaktpunkten

Fingerspitzengriff-Modell

- Das Fingerspitzengriff-Modell vereinfacht die Algorithmen zur Synthese möglicher Griffe eines Objektes, da nur eine geeignete Anordnung der Kontaktpunkte auf der Oberfläche der zu greifenden Objektes bestimmt werden muss.

- Ein gravierender Nachteil liegt in der Nichtbeachtung fundamentaler Nebenbedingungen des Greifvorgangs, wie z.B. der Kollisionsfreiheit und Zugänglichkeit eines Griffes (Anfahr-/Abrückbewegung).

Fingerspitzenkontakte mit der Objektoberfläche
Man unterscheidet verschiedene Fingerspitzenkontakte zur Objektoberfläche mit folgenden Annahmen:

- Punktkontakt ohne Reibung: Eine an einem Punktkontakt ohne Reibung auf eine Fläche eines Objektes angreifende Kraft wirkt ausschließlich normal zur Fläche

- Starrer Punktkontakt mit Reibung: Eine an einem starren Punktkontakt mit Reibung auf eine Fläche eines Objektes angreifende Kraft wirkt sowohl normal als auch tangential zur Fläche. Die beiden Kräfte sind über das Coulombsche Reibungsgesetz miteinander verknüpft

- Nicht starrer Punktkontakt mit Reibung (Soft-Kontakt): Eine an einem nicht starren Punktkontakt mit Reibung auf eine Fläche eines Objektes angreifende Kraft wirkt sowohl normal als auch tangential. Zusätzlich wirken auch axiale Momente. Es gilt ebenfalls das Coulombsche Reibungsgesetz

Wrenchvektor: Die in einem Kontaktpunkt p_i wirkenden Kräfte f_i und Momente τ_i mit $i \in \{x, y, z\}$ können zu einem Vektor zusammen gefasst werden. Solch ein Vektor wird im folgenden als Wrenchvektor q bezeichnet (wrench: engl.: drehen, winden).

- In Abhänigkeit vom Typ des i-ten Kontaktpunktes folgen Wrenchvektoren, welche die am Kontaktpunkt wirkenden normalen (n) und tangentialen (t) Kräfte und die am Kontaktpunkt wirkenden axialen Momente (θ) beschreiben

Greifmatrix: Die Wrenchvektoren können für einen räumlichen Griff als Spaltenvektor einer 6x3m Matrix G dargestellt werden:
$G = [{}^1w_n, {}^1w_t, {}^1w_\theta, ..., {}^mw_n, {}^mw_t, {}^mw_\theta] \in \mathbb{R}^{6x3m}$ mit m: Anzahl der Kontaktpunkte
Die Matrix G repräsentiert die geometrischen und physikalischen Eigenschaften eines Fingerspitzengriffes und wird im folgenden als Greifmatrix bezeichnet. Für die Skalare erhält man den Vektor: $\vec{c} = ({}^1c_n, {}^1c_t, {}^1c_\theta, ..., {}^mc_n, {}^mc_t, {}^mc_\theta) \in \mathbb{R}^{3m}$

Gleichgewichtsgriff: Ein durch eine Greifmatrix G spzifizierter Griff, auf den eine externe Kraft und ein externes Moment $e = (f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)^T \in \mathbb{R}^6$ ausgeübt werden, wird als Gleichgewichtsgriff bezeichnet, falls:

1. $\forall i \in 1, ..., m$: ${}^i c_n \geq 0$, ${}^i \mu_t * {}^i c_n \geq |{}^i c_t|$, ${}^i \mu_\theta * {}^i c_n \geq |{}^i c_\theta|$
2. $\exists c \in \mathbb{R}^{3m} : G.c + e = 0$

Kraftgeschlossene Griffe

- Während der Transferbewegung und der Ausführung einer Greif- operation ist ein gegriffenes Objekt verschiedenen externen Kräften und Momenten ausgesetzt.
- Die Stabilität eines Griffes erfordert, dass das gegriffene Objekt im Kräftegleichgewicht bleibt. Dies bedeutet, dass die Kräfte und Momente, die durch die Greiferfinger auf das gegriffene Objekt ausgeübt werden, sämtliche externen Kräfte und Momente kompensieren müssen.
- Sind die externen Kräfte wie z.B. Störkräfte im voraus nicht bekannt, bietet sich das kraftgeschlossene Greifen zur Erreichung eines stabilen Griffes an.
- Ein durch eine Greifmatrix G spezifizierter Griff ist kraftgeschlossen, falls:
  - $\forall e = (f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)^T \in \mathbb{R}^6$

Anzahl benötigter Kontaktpunkte

- Für die minimale Anzahl der benötigten Kontaktpunkte eines kraftgeschlossenen Fingerspitzengriffes gilt folgendes:
  - Kraftgeschlossenheit basierend auf Punktkontakten ohne Reibung: Vorausgesetzt, dass das zu greifende Objekt keine Rotationssymmetrie besitzt, benötigt ein planarer, kraftgeschlossener Griff mindestens 4 Kontaktpunkte. Werden beliebige 3D-Objekte betrachtet, so werden höchstens 12 Kontaktpunkte benötigt. Wird die Klasse der zu greifenden Objekte auf Polyeder eingeschränkt, so gilt eine generelle obere Grenze von 7 Kontaktpunkten.
  - Kraftgeschlossenheit basierend auf Punktkontakten mit Reibung: Jedes planare Objekt kann durch einen auf 3 Kontaktpunkten basierenden Fingerspitzengriff kraftgeschlossen gegriffen werden. Für den räumlichen Fall gilt eine untere Grenze von 4 Kontaktpunkten.

Formgeschlossene Griffe

- Ein formgeschlossener Griff unterliegt stärkeren Einschränkungen als ein kraftgeschlossener Griff, da für jeden Kontaktpunkt ausschließlich die Nichtdurchdringungseigenschaften co-linear zum korrespondierenden externen Oberflächennormalvektor berücksichtigt werden.

- Somit ist die Formgeschlossenheit eines Griffes nur von der Position der Kontaktpunkte auf der Oberfläche des zu greifenden Objektes und den korrespondierenden externen Oberflächennormalvektoren abhängig.
- Es werden weder Normal- oder Tangentialkräfte noch Drehmomente, die u.a. aufgrund von Reibung auftreten könnten, berücksichtigt.
- Die zu den Kontaktpunkten korrespondierenden externen Oberflächen-Normalenvektoren spezifizieren die Kontaktgeometrie des Figerspritzengriffs. Diese kann durch folgende modifizierte Greifmatrix t'$G \in \mathbb{R}^{3xm}$ ausgedrückt werden.
t'$G = [{}^1 w_n, {}^2 w_n, ..., {}^m w_n]$
- Für einen formgeschlossenen, planaren Griff sind mindestens 4 Kontaktpunkte erforderlich. Bei beliebigen, 3D Objekten erhöht sich die Anzahl auf wenigstens 7 Kontaktpunkte.

Kraft- und Formschlüssige Griffe

- Kraftschluss: Die Kinematik der Hand kann aktiv Kräfte erzeugen, um einer externen Störung zu wiederstehen
- Formschluss: Die Kontakte an sich verhindern, dass sich das Objekt bewegen kann
- Formschluss ist mit weniger Kontaktpunkten möglich und wird deshalb bei Präzisionsgriffen verwendet; erfordert jedoch eine Regelung der intern auftretenden Kräfte bei einem Griff.
- Die Analyse von formschlüssigen Griffen erfolgt basierend auf die Geometrie.

Stabile Griffe

- Bisher wurde vorausgesetzt, dass die Kontaktpunkte eines Griffes durch starre Greiferfinger hervorgerufen werden. In der Realität ist dies jedoch nicht der Fall und oft nicht wünschenswert.
- Zur Modellierung von Fingerkräften, die durch ihre Nachgiebigkeit kleine Änderungen von der Nominallage des gegriffenen Objektes kompensieren können, führt man eine Potentialfunktion V ein: $V : \mathbb{R}^6 \rightarrow \mathbb{R}$
- Die Potentialfunktion V spezifiziert die im Griff gespeicherte potentielle Energie in Abhängigkeit von Lage und Orientierung des gegriffenen Objektes.

Klassifikation von Greifplanungssystemen

1. Typ des verwendeten Greifers (Zweifinger-, Dreifinger-, Mehrfingergreifer,...)
2. Typ der zugrunde liegenden Greifplanungsalgorithmen. Geometrisch-basiert (Berücksichtigung von CAD-Daten), physikalisch basiert (auftretende Kräfte und Momente),...

3. Typ der zu manipulierenden Szenen. Deterministisch, d.h. Lage und Orientierung aller Objekte in der Szene bekannt oder nicht deterministisch

4. Einsatz von Sensorik. Keine Sensorik, taktile Sensorik, visuelle Sensorik,...

Suchraum

- Soll ein zulässiger Griff geplant werden, so beträgt die Dimension des Suchraums für den physikalischen als auch für den geometrischen Ansatz $6 + n$ (6 Parameter für die Position und Orientierung der Hand im Raum, n Anzahl der Konfigurationsparameter der Greiferfinger)
- Bei einem Parallelbackengreifer hat der Suchraum die Dimension 7 (1 für den Greifer + 6)
- Bei der ARMAR-III Hand hat der Suchraum die Dimension 14 (8 für die Hand + 6)
- Bei der menschlichen Hand hat der Suchraum die Dimension 28 (22 für die Hand + 6)

Objektklassen für das Greifen

- Bekannte Objekte (known objects)
  - Bekannte Objektgeometrie (d.h. wir haben ein komplettes Objektmodell)
  - Ansatz: Verwende einen Greifplaner, der mit bekannten Objektgeometrien arbeitet
  - Domäne für klassische Greifplanung
  - Schwierig
- Bekannte Objektklasse (familiar objects)
  - Konkrete Objektgeometrie ist nicht bekannt (z.B. „Objekt ist vom Typ Flasche“)
  - Ansatz: Übertrage Wissen von bekannten Klassenelementen auf das zu greifende Objekt
  - Schwieriger
- Unbekannte Objekte (unknown objects)
  - Weder Objektgeometrie, noch Objektklasse ist bekannt
  - Probleme (u.a.)
    - Verarbeiten von (unvollständigen) Sensordaten (Stereo Vision, RGB-D, Laserscans, haptische Daten)
    - Segmentierung des Objektes vom Hintergrund
    - Erstellen eines (teilweisen) Objektmodells
  - Ansätze (u.a.)
    - Sensorfusion

        * Objekt verschieben
    - Am schwierigsten!

Aktuelle Algorithmen zur Griffsynthese

- Vorraussetzungen
    - Definition der Handkinematik
    - Kontaktmodell (meist: Punktkontakt mit Reibung)
    - Objektmodell (meist: vollständig bekannt)
- Algorithmen zur Greifsynthese bei bekannten Objekten
    - Siehe Kapitel 9 - Folie 46

Griffsynthese durch Vorwärtsplanung

- Ansatz
    - Planung in Simulation
    - Bestimmung von Anfahrtspunkt und Anfahrtsrichtung
    - Hand nähert sich dem Objekt, bis ein Kontakt detektiert wird
    - Finger schließen sich um das Objekt, bis Kontakt hergestellt ist
    - Evaluation der Kontakte zwischen Hand und Objekt
- Vorteile
    - Vorwärtsplanung ist ähnlich zur Ausführung eines Griffs auf einem realen Roboter
    - Griffe, die erfolgreich in der Simulation evaluiert wurden, können mit hoher Wahrscheinlichkeit auch mit einem realen Roboter durchgeführt werden
- Algorithmus
    1. Lade Hand- und Objektmodell in eine Simulationsumgebung
    2. Erzeuge Griffkandidaten
        - Bestimme Näherungsrichtung der Hand zum Objekt
        - Bestimme Orientierung der Hand
        - Bestimme Handkonfiguration (beginnend mit der geöffneten Hand)
    3. Evaluation der Griffkandidaten
        - Bewege die Hand entlang der Näherungsrichtung bis zu Kontakt mit dem Objekt
        - Schließe die Hand bis Kontakt mit dem Objekt
        - Bestimmt die Kontaktpunkte
        - Bestimme die Griffqualität (Krafschluss-Metrik)

Griffqualität (Kraftschluss-Metrik)

- Prinzip: Wie gut kann ein Griff externen Kräften widerstehen?
- Ansatz
  - Bestimme Kontaktpunkte und Kontaktnormalen zwischen Hand und Objekt
  - Bestimme den Reibungskegel an jedem Kontaktpunkt (Öffnungswinkel des Kegels hängt von den Reibunskoeffizienten ab)
  - Berechne den Grasp Wrench Space als konvexe Hülle über alle Reibungskegel
  - Die minimale Distanz vom Zentrum zum Rand des GWS ist ein Maß für die Stabilität des Griffs

Grasp Wrench Space

- Grasp Wrench Space (GWS): Konvexe Hülle über die Vereinigung aller Kontakt-Wrenches
- Qualitätsmaß
  - Kraftschluss (force closure): GWS enthält Ursprung
  - Volumen (V): Volumen des GWS
  - Epsilon (ϵ): größte einschließende Kugel, bzw. kleinste Distanz ϵ vom Ursprung zum Rand des GWS

Zufallsbasierte Vorwärts-Greifplanung

- Ablauf
  1. Randomisierte Erzeugung von Greifhypothesen (Position und Orientierung der Hand, Konfiguration der Finger)
  2. Kontaktermittlung
  3. Evaluation der Hypothesen (Kraftschluss, Kollision, Robustheit)
- Handmodell: Grasp Center Point (GCP): Definiert das Zentrum g sowie die Anfahrtsrichtung a für einen Grifftyp
- Erzeugung von Greifhypothesen: Bestimmung der Anfahrtsrichtung: Zufällige Auswahl eines Oberflächenpunktes p, Ermittlung der Oberflächennormalen n
- Bestimmung der Greifhypothese
  - Positionierung der Hand (Position der Hand: g liegt auf Halbgerade, welche durch p und n definiert wird, Ausrichtung der Handorientierung, so dass a und n kollinear sind. Der freie Parameter (Orientierung um a) wird zufällig gewählt)
  - Kontaktermittlung: Bewegung der Hand entlang a zum Objekt, Schließen der Finger, Ermittlung von n Kontaktpunkten $k_1, ...k_n$
- Analyse der Greifhypothese

  - Kontaktpunkte $k_1, ..., k_n$ werden analysiert (Kraftschluss, Griffqualität)
  - Valide Griffe werden gespeichert
  - Greifhypothese mit unzureichenden Eigenschaften werden verworfen

Griffsynthese auf Teilobjekten

- Frage: Wie können gute Griffkandidaten erzeugt werden?
- Ansätze für verschiedene Unterteilungsverfahren
  - Formprimitive (shapes primitives): Manuelle Unterteilung in Primitive (z.B. Boxe, Zylinder, Kugeln, Kegel, etc.)
  - Box decomposition: Automatische Unterteilung (nur Boxen)
  - Superquadriken: Automatische Unterteilung
  - Mediale-Achse-Transformation: Nur Kugeln
  - Oberflächennormalen

Greifplanung mit Formprimiviten

- Objekte werden durch einfache Formprimitive (shape primivites) dargestellt
- Für jedes Formprimitiv sind unterschiedliche Greifstrategien vordefiniert (inklusiv Startpunkt und Anrückrichtung)
- Vorwärtssimulation des Greifprozesses:
  - Der Startpunkt definiert die initiale Greiferposition
  - Basierend auf der Anrückrichtung wird der Greifer zum Objekt bewegt bis ein Kontakt ermittelt wird
  - Kontaktpunkte werden durch Schließen des Greifers bestimmt
  - Evaluation der Kontakte über den GWS Ansatz
- Greifstrategien
  - Box
  - Kugel
  - Zylinder
  - Kegel
- Parameter
  - Anzahl paralleler Unterteilungen (Boxen, Seiten von Zylinder und Kegel)
  - Anzahl Kreisunterteilungen (Kugel, Seiten von Zylinder und Kegel)
  - Anzahl Handorientierungen (um die Anfahrtsrichtung)
  - Spiegelung der Griffe (bei seitensymmetrischen Primitiven)

- Die Parameter werden automatisch aus den Objektdimensionen bestimmt.

Greifen bekannter Objekte: Ein Box-basierter Ansatz

- Approximation der Objektgeometrie durch Boxen (box decomposition)
- Griffhypothesen für Boxen erzeugen
- Evaluation der Griffhypothesen in GraspIt!
- Haushaltsobjekte mit bekannter Geometrie
  - Aus der KIT object models database
  - Objektrepräsentation: 3D Punktwolen, Meshes
- Approximation der Objektgeometrie durch Box Decomposition
  - Effizienter Mimimum Volume Bounding Box (MVBB) Algorithmus
  - Basierend auf Punktwolken

Erzeugen von Griffhypothesen: Von Boxen zu Griffen

- Jede Box hat sechs Seitenflächen, welche jeweils vier Griffhypothesen implizieren
  - Griffpunkt: Mittelpunkt der Seitenfläche
  - Griffrichtung: Entlang der Normale der Seitenfläche
  - Handorientierung: Vier Möglichkeiten, orientiert an den Kanten der Fläche
- Über die Größen der Seitenflächen können unmögliche Griffe direkt ausgeschlossen werden (z.B. Seitenfläche ist größer als die Handöffnung)
- Griffhypothese für blockierte oder verdeckte Seitenflächen können durch einfache geometrische Berechnungen verworfen werden

Ergebnisse:

- Exakte Form ist nicht notwendig, um Griffe zu erzeugen
- Objekte können durch vereinfachte Form (Boxen) besser verarbeitet werden. Hierzu können Punktwolken der Oberflächen verwendet werden.
- Einfache Parametrisierung der Algorithmen
- Evaluation: Der box-decomposition Ansatz erzeugt Griffe, welche mit einem humanoiden Roboter erfolgreich ausgeführt werden können.

Greifplanung mit Superquadriken

- Superquadrik
  - Parametrisierbare Funktionen definieren die Form des geometrischen Objektes

  - Eine an den Koordinatenachsen ausgerichtete Superquadrik im Ursprung des Koordinatensystems wird durch fünf Parameter mit folgender Gleichung beschrieben (...)

- Objektdarstellung über Superquadriken
  - Objektoberfläche wird als Punktwolke dargestellt
  - Ermittle Superquadrik deren Oberfläche die Punktwolke am besten darstellt
  - Decomposition Tree
    * Verfeinere die Approximation schrittweise: Suche Superquadrik mit größtem Fehler zu Punktwolke, teile die zugehörige Punktwolke, erstelle zwei Superquadriken
    * Speichere Resultate für jeden Approximationsschritt im decomposition tree

- Greifplanung
  - Für jede Superquadrik werden Greifhypothesen erzeugt: Gleichverteilte Anfahrtspunkte und Anfahrtsrichtungen relative zur Oberfläche der entsprechenden Superquadrik
  - Evaluation der Greifhypothesen mittels GWS Ansätzen: Detailliertes Objektmodell (3D mesh)
  - Für alle Stufen des decomposition trees: Greifplanung auf groben sowie feinen Strukturen des Objektes

Motivation

- Stand der Forschung: Algorithmen zur Griffplanung in Simulationsumgebungen
  - Griffhypothesen werden auf Stabilität untersucht
  - Effizienz hängt von den Heuristiken zur Erzeugung von Griffhypothesen ab

- Ziel: Verbesserung der Effizienz von Griffplanungs-Algorithmen, indem nur „geometrisch sinnvolle“ Griffe untersucht werden müssen

- Ansatz: Verwende lokale Symmetrien der Objektgeometrie
  - Die Repräsentation des Objektes ist wichtig
    * Dreiecksnetz (Mesh): Zu niedriges Abstraktionsniveau
    * Mediale Achse als Objektrepräsentation

Griffplanung mit medialen Achsen

- Mediale Achse
  - Objektform wird approximiert über enthaltene Kugeln mit maximalem Durchmesser
  - Enthaltene Kugeln müssen die Objekthülle an zwei oder mehr Punkten berühren

- Die mediale Achse ist die Vereinigung der Mittelpunkte aller enthaltenen Kugeln
- Die mediale Achse beschreibt das topologische Skelett des Objekts
- Vorteile:
  - Gute Approximation der Objektgeometrie
  - Details bleiben erhalten
  - Gute Beschreibung der Symmetrien
- Algorithmus
  1. Abtasten der Objektoberfläche
  2. Berechnen der medialen Achse
  3. Analyse der Querschnitte der medialen Achse
     - Minimum Spanning Tree (MST)
     - Clustern
     - Konvexe Hülle
  4. Erzeuge Griffhypothesen
  5. Evaluiere Griffstabilität
- Zusammenfassung:
  - Die mediale Achse enthält Informationen über Struktur und Geometrie des Objektes
  - Generierung möglicher Griffe mit Heuristiken
    * Einbeziehung von Objektsymmetrie
    * Mögliche Griffe sind geometrisch sinnvolle: Hoher prozentualer Anteil von stabilen Griffen, resultierende Griffe wirken natürlich
  - Genaue Approximation der Objektgeometrie: Mögliche Griffe werden nicht durch eine schlecht approximierte Objektgeometrie beeinflusst
  - Erweiterbares Konzept: Erweiterbarer Satz von Heuristiken um mögliche Griff zu generieren

10 Umweltmodellierung

10.1 Motivation: Adaptive Roboteraufgaben

Bisher in VL: un- oder marginal flexible Roboteraufgaben

- Feste Trajektorie programmiert:
  - Positionsregelung

  - Eventuell lokale Kraftregelung
  - Benötigte Modelle für Programmierung und Regelung: Kinematisches Modell, dynamisches Modell
  - Benötigte Sensoren: Gelenkencoder, (eventuell) Kraft-Momenten-Sensor
- Bahn absolut fest oder nur lokal-adaptiv, repetitiv ausgeführt

Adaptive Roboteraufgaben:

- Keine feste Trajektorie programmier
  - Trajektorie muss flexibel vom Roboter bestimmt werden
  - Benötigte Modelle für Programmierung und Regelung:
    * Kinematisches Modell
    * Dynamisches Modell
    * + Umweltmodelle
    * + Aufgaben/Planungsmodelle
  - Benötigte Sensoren:
    * Gelenkencoder
    * (eventuell) Kraft-Momenten-Sensor
    * + visuelle und eventuell taktile Sensoren
- Bahn völlig frei, flexibel, nicht-repetitiv ausgeführt

Anforderungen für Adaption (Zusätzliche Anforderungen gegenüber starren Aufgaben)

1. Visuelle und (eventuell) taktile Sensorik zur Umwelterfassung (Rob3 im SoSe)
2. Datenrepräsentation der Umgebung/Umwelt (dieses Kapitel)
3. Planungsmethoden zur Aufgaben/Bewegungsberechnung (Kapitel 9,10,12)

Übersicht Modelle (Verschiedene Arten von Modellen in der Robotik)

1. Modell der Kinematik (Manipulator, mobile Plattform)
2. Modell der Dynamik (Manipulator, mobile Plattform)
3. Modelle der Sensoren
4. Umweltmodell (Umgebung: Einzelobjekte und Szenen)
5. Aufgabenmodell (Bewegungen und abstrakte Aufgaben)

10.2 Objektmodelle

10.2.1 Geometrische Beschreibung

Anwendungsgebiete für das geometrische Modell

- Umweltwahrnehmung (Live)
  - Objektklassifizierung
  - Objektlokalisierung
  - Bewegungserkennung/klassifikation des Menschen
- Bewegungsplanung (Live, offline)
  - Pfadplanung (mobile Systeme)
  - Bewegungsplanungen (Manipulatoren)
  - Greifplanung und Planung von Inhandmanipulationsaufgaben
- Abstrakte Aufgabenplanung (Live, offline)
- Dynamische Effektsimulation und -prädiktion (Live,offline)
- Arbeitsraum/Anlagenplanung (Offline)

Volumenmodelle

- Punktwolken
- Approximative Oberflächenmodelle (B-Rep)
  - Dreiecksflächen (Meshes)
  - Viereckflächen
  - Bezierflächen
- Approximative Zellenbelegung
  - Voxel
  - Octree
- Analytisch-parametrische Modelle
  - Constructive Solide Geometry (CSG)

Punktwolken (Point Clouds) sind das einfachste Objektmodell

- Modell ist eine beliebige Menge aus Raumpunkten
- Daten direkt aus aktuellen Tiefensensoren (3D-Sensoren)
- Objekte nur implizit, indirekt modelliert

- Mehrdeutigkeiten
- Hochgradig approximativ
- Datenlücken durch Modellform
- Grundlage für weitergehende Objektmodelle (z.B. Erkennung/Lokalisierung von Objekten im Raum)

Approximative Oberflächen

- Bildung einer großen Fläche aus einem Netz (Mesh) von einfachen Einzelflächen, z.B. Dreiecke, Vierecke
- Vorteile:
  - Definition sehr einfach
  - einfache Algorithmen
- Nachteile:
  - hoher Speicherbedarf
  - hoher Rechenaufwand

Dreiecksflächen

- Approximation von Freiformflächen: Die einfachste Fläche ist die Dreiecksfläche.
- Definition: Gegeben seien 3 Punkte im Raum P_1, P_2, P_3. Damit hat die Fläche folgende Gleichung: $F(u,v) = u \cdot P_1 + v \cdot P_2 + (1-u-v) \cdot P_3$ mit $0 \leq u, v, u+, v \leq 1$

Bilineare Viereckflächen

- Definition: Gegeben sind 4 Punkte im Raum P_1, P_2, P_3, P_4. Damit wird die Fläche definiert durch: $F(u,v) = (1-u)(1-v) \cdot P_1 + (1-u)v \cdot P_2 + u(1-v) \cdot P_3 + uv \cdot P_4$ mit $0 \leq u \leq 1$, $0 \leq v \leq 1$
- Vorteil: Flächenelemente können gekrümmt sein (weniger Gitterpunkte bei gleich guter Approximation)
- Nachteil: Rechnen mit gekrümmten Flächen ist aufwendig

Bezierflächen (Erweiterung des Ansatzes der Bezierkurven)

- Definition: Gegeben ist ein Gitter von Führungspunkten P_{ij}
- Damit ist die Fläche beschreiben durch $F(u,v) = \sum_{i=0}^{N} \sum_{j=0} MP_{ij} \cdot B_{i,N}(u) \cdot B_{j,M}(v)$

Kantenmodell (reines Kantenmodell ohne weitere Informationen)

- Nur die Kanten werden gespeichert, d.h. Punkte und Verbindungen (Gerade, Polygonzug, Bezierkurve)
- Vorteile:
  - einfache Daten
  - wenige Daten
- Nachteile:
  - Mehrdeutigkeiten
  - hoher Eingabeaufwand
  - keine Kollisionsberechnung
  - kein Schnitt

Flächenmodell

- Speicherung von Kanten & Oberflächen (Dreiecke,Bezier)
- Vorteile:
  - effiziente Verfahren
  - entspricht dem Vorgehen während der Modellierung
  - schnelle Kollisions- und Abstandsberechnung
- Nachteile:
  - hoher Eingabeaufwand
  - Darstellung aufwendig
  - Problem bei Schnittoperationen
  - Inkosistenzen möglich

Boundary Representation

- Organisationsform der geometrischen Flächenmodelle
  - Hierarchische Darstellung eines Objektes durch begrenzende Elemente, i.d.R. Kanten oder Flächen.
  - Elemente eines Quaders im Flächenmodell
  - Elemente: Quader, 6 Flächen, 12 Kanten, 8 Ecken
- Vorteile: Aus topologischer Struktur Informationen über
  - Welche Flächen gehören zum Objekt?
  - Welche Kanten gehören zur Fläche?
  - Zu welchem Objekt gehört eine Fläche?
  - Zu welchem Objekt gehört eine Kante?

  - Welche Flächen stoßen aneinander?
  - → kantenbasierte Objekterkennung

Approximative Zellenbelegung

- Objekte werden aus disjunkten Elementarzellen aufgebaut. Verwendung finden einfache geometrische Objekte z.B. Tetraeder, Quader,...
- Benutzt in der Strukturanalyse mit Finite-Elemente-Methoden (FEM).

Voxeldarstellung (äquidistante Raumunterteilung in 3D)

- Speicherungsmöglichkeiten (0: nicht-belegt, 1: belegt, 2: weiß-nicht (unbekannt), Wahrscheinlichkeit belegt 0.0 bis 1.0)
- Vorteile:
  - Einfache Darstellung
  - Berechnungen homogen, parallel auf GPU
  - Raytracing u.ä. einfach parallelisierbar
- Nachteile:
  - Festes Gitter, gleicher Detaillierungsgrad sowohl bei großflächigen als auch bei feiner aufgelösten Strukturen
  - Kartengröße/Präzision durch Speicher begrenzt

Octree (effizientere Topologie als Voxel)

- Der Raum wird in mehrere Zellen unterteilt
- Zelle komplett vom Objekt belegt → als belegt markieren
- Wenn die Zelle nur teilweise belegt ist, dann wird auf diese Zelle das Verfahren rekursiv angewendet. Ansonsten ist die Zelle leer.
- Die Rekursion terminiert bei einer vorbestimmten minimalen Zellgröße.
- Teilbelegte kleinste Zellen werden als belegt markiert.

Analytisch-parametrische Modelle (Beispiel analytisch gegebene Fläche)

- Beispiel: 3D-Kugel
- $r = ||x - p||$
- Exaktes Modellierungsverfahren
- Vorteile:
  - Geschlossene Darstellung (wenig Speicherbedarf)

  - Analytische Darstellung erlaubt einfache Rechenverfahren (z.B. Schnitt von Ebenen/Kugeln -> schnelle Kollisionsberechnung)
- Nachteil: Wenige Flächen sind analytisch darstellbar

Parametrische Volumenmodelle

- Grundkörper und topologische Operationen auf diesen (Schnitt, Vereinigung,...) werden abgespeichert.
- Vorteile:
  - eindeutige Objektbeschreibung
  - geringer Eingabeaufwand
  - Ergebnis von Operationen sind korrekte Objekte
- Nachteile:
  - hoher Implementierungsaufwand
  - Einbindung von Freiformflächen schwierig
- Objekte sind bereits vorhanden und können durch Angabe von Parameter angepasst werden (Varianten).
- Konsistenzprüfungen sind notwendig!

Constructive Solid Geometry (CSG)

- Es gibt eine Menge von einfachen Grundkörpern, die parametriert werden können und auf denen verschiedene Operationen definiert sind.
- Speicherung erfolgt als Binär-Baum mit Knoten als Operation und linker bzw. rechter Teilbaum als Teil-CSG oder Grundkörper

Anwendungsgebiete für das geometrische Modell

- Punktwolke:
  - Lokalisation, Klassifikation
  - Kartierung (movile Systeme)
- Mesh:
  - Bewegungsplanung (v.a. Manipulatoren)
  - Dynamische Simulation (komplexere Starrkörper)
- Voxel/Octtree
  - Bewegungsplanung (v.a. mobile Systeme)
  - Dynamische Simulation elastische Materialien (FEM)
- CSG:
  - CAD/CAE/CAM
  - Dynamische Simulation (einfache Starrkörper)

10.2.2 Zusätzliche Eigenschaften

Eigenschaften des Objektes mit Geometriebezug

- Masse
- Oberflächeneigenschaften (z.B. Reigung)
- Temperatur
- Steifigkeit
- Greifpunkte
- Verbindungspunkte zur Montage
- Ablagepunkte bei Paletten
- Stellung bezüglich eines Referenzobjektes

10.3 Szenenmodelle

Entity-Relationship-Modelle (ER) //TODO

Semantische Netze (Konzept: Objekte und Beziehungen)

- Semantisches Netz = gerichteter Graph
- Knoten = Objektklasse oder Einzelobjekt
- gerichtete und benannte Kanten = Beziehungen
- nur zweistellige Beziehungen
- mehrstellige Beziehungen => als eigenes Objekt
- keine Attribute => Attribute als Wertobjekt

Frame Modell nach Minsky

- Frame = Schablone (nicht mit Koordinaten-Frames verwechseln)
- Erfassung der:
  - Eigenschaften von Objekten
  - Einordnung in Hierarchie von Objektklassen
- Leichte Implementierung mit objektorientierten Sprachen
- Frame beschreibt ein Objekt/Objektklasse und enthält eine Menge von Slots
- Slots enthalten Attribute und Verweise auf andere Frames

- Instanzen, Vererbung

Implicit Shape Modell (ISM) - Moderne Szenenbeschreibung, robust bei Varianzen

- Szene besteht aus relativen Transformationsrelationen zwischen Objekten
- ISM ist Variation der Generalisierten Hough Transformation (Maschinensehen)
- Daraus folgt, dass Instanzen der Relationen in buckets abstimmen (voten)
- Baumartige Topologie der relativen Relationen

Objekt Constellation Modell (OCM): probabilistisches Modell

- Probabilistische Szenenrelationen
- Basiert auf Gauss-Mixturen (GMMs) und probabilistischer Graph-Inferenz

11 BILDVERARBEITUNG

Motivation

- Das Sehvermögen ermöglicht die Wahrnehmung der Umwelt
- Nutzbarkeit in einem technischen Systen
  - Visuellen Informationen müssen aufgenommen werden
    - gute Qualität
    - digitales Format
  - Relevante Informationen müssen aus den Daten extrahiert werden
- Bilderfassung: Hardware
- Bildverarbeitung: überwiegend Software

Bildrepräsentation

- Bilder müssen im Computer/Roboter repräsentiert werden
- Ein Bild ist ein 2D Gitter von diskreten Punkten (Pixel)
- Bildkoordinaten (hier):
  - u (horizontal)
  - v (vertikal)
  - Urpsrung ist oben links
  - Einheiten: Pixel
- Die Farbe eines Bildpunktes kann auf unterschiedliche Weise repräsentiert werden

- Graustufenbilder: Für jeden Pixel wird ein Helligkeitswert abgelegt (Normalerweise ein Byte pro Pixel, i.A. Werte in [0,255])
- Monochrombild: Diskrete Funktion
- Farbbild
  - Verschiedene Farbmodelle für unterschiedliche Anwendungen
  - Klassifikation nach erreichbarem Farbraum

Koordinatensysteme

- Hauptachse: Gerade durch das Projektionszentrum, rechtwinklig zur Bildebene
- Hauptpunkt: Schnitt der Hauptachse mit der Bildebene
- Bildkoordinaten: 2D Koordinaten eines Punktes im Bild. Einheit: Pixel
- Kamerakoordinatensystem: 3D Koordinaten eines Punktes relativ zur Kamera. Der Ursprung liegt im Projektionszentrum, die x- und y-Achse ist parallel zur u- bzw. v-Achse in der Bildebene. Einheit: mm
- Weltkoordinatensystem: 3D Basiskoordinatensystem in der Welt. Einheit: mm

Kameraparameter

- Parameter, die ds Kameramodell vollständig beschreiben
- Man unterscheidet zwischen intrinsischen und extrinsischen Kameraparametern
  - Intrinsische Parameter sind unabhängig von der Auswahl des Weltkoordinatensystems. Sie bleiben konstant bei Veränderung des Aufbaus.
  - Extrinsische Parameter modellieren die Transformation vom Weltkoordinatensystem in das Kamerakoordinatensystem. Sie sind bei einer neuen Lage der Kamera neu zu bestimmen.
- Bisher:
  - Im Lochkameramodell nur ein intrinsischer Kameraparameter (f)
  - Kein Weltkoordinatensystem berücksichtigt, deshalb keine extrinisches Parameter
  - Annahmen: Hauptpunkt im Ursprung des Bildkoordinatensystem; Pixel sind exakt quadratisch; Keine Linzenverzerrung

Filteroperationen

- Filter in der Bildbearbeitung (auch spartial filters, spatial masks, kernels, windows)
- Ein Filter besteht aus einer Nachbarschaft und einer vordefinierten Operation

- Ein Filter wird auf alle Pixel de Bildes angewendet: Berechnung eines neuen Pixelwertes durch Anwendung der Filteroperation unter Berücksichtigung der Nachbarschaftspixel
- Tiefpassfilter: Glättung, Rauschelimination (Mitterlwertfilter, Gauß-Filter)
- Hochpassfilter: Kantendetektion (Prewitt, Sobel, Laplace, Roberts)
- Kombinierte Operatoren (Laplacian of Gaussian)

Mittelwertfilter (Rauschunterdrückung)

- Beispiel: Durchschnitt aus einem Pixel und seine 8 Nachbarn; Größe beliebig wählbar
- 3x3 Mittelwertfilter: Filtermaske (3x3 Matrix mit je 1/9 als Wert)

Gauß-Filter (Rauschunterdrückung, Glättung)

- Definiert durch zweidimensionale Gauß-Funktion
- Approximation von $f(x)$ durch einen 3x3-Filter $1/16(121;242;121)$
- Die Stärke der Glättung ist ausschließlich durch den Parameter σ bestimmt: Je größer σ, umso stärker die Glättung.
- Die Größe nxn der Filtermaske beeinflusst die Güte der Approximation des Filters.

Prewitt-Filter

- Prewitt-X approximiert durch $(-101;-101;-101)$
- Prewitt-Y approximiert durch $(-1-1-1;000;111)$
- Eigenschaften: Gute Ergebnisse bei Detektion von vertikalen bzw. horizontalen Kanten
- Komvination der Prewitt-Filter zur Bestimmung des Gradientenbetrags $M \approx \sqrt{P_x^2 + P_y^2}$
- Danach: Schwellwertfilterung

Segmentierung

- Segmentierung ist die Aufteilung eines Bildes in aussagekräftige Segmente
- Jedes Pixel wird mindestens einem Segment zugeordnet
- Identifikation von interessanten Bildregionen für die Analyse, Erkennung unf Klassifikation
- Mögliche Verfahren:
  - Schwellwertfilterung
  - Clustering
  - Kantenextraktion

  - Region-Growing

Schwellwertfilterung

- Schwellwertfilterung zur Konvertierung eines Grauwertbildes in ein binäres Bild
- Intensität vin jedem Pixel (u, v) wird mit einem vordefinierten Schwellwert T abgeglichen
- Oft können Objekte über ihre Farbe segmentiert werden (menschliche Haut, einfarbige Objekte)
- Problem: Wechselnde Lichtbedingungen, Reflexionen, Schattenwürfe

Morphologische Operatoren

- Morphologische Operatoren werden oft für die Nachbearbeitung binärer Bilder verwendet (z.B. Ergebnis einer Farbsegmentierung)
- Gängige morphologische Operatoren:
  - Dilatation: vergrößert Pixel zu größeren Bereichen
  - Erosion: entfernt vereinzelte Pixel und schwach zusammenhängende Pixelgruppen
- Der Effekt eines morphologischen Operators hängt von der Größe und der Form der berücksichtigten Pixelnachbarschaft ab

Canny-Kantendetektor

- Der Canny-Kantendetektor ist weit verbreitet und bezüglich der Leistung im Vergleich zu anderen Kantendetektoren im Allgemeinen überlegen.
- Ziel war es den optimalen Kantendetektor zu finden:
  - Gute Detektion
  - Gute Lokalisierung
  - Minimale Antwort
- Canny-Kantendetektor berechnet binäre Antwort
- Subpixelgenauigkeit durch Erweiterung möglich
- Algorithmus besteht aus mehreren Schritten
- Grundsätze:
  1. Geringe Fehlerrate: Alle Kanten sollen gefunden werden und detektierte Kanten sollten so nah wie möglich an den realen Kanten sein.
  2. Kantenpunkte sollen gut detektiert werden: Distanz zwischen detektierten Kantenpunkten und dem Zentrum der realen Kanten soll minimal sein.

3. Eindeutigkeit: Der Detektor soll nur ein Punkt, nicht mehrere Kantenpunkte, zu einem realen Kantenpunkt liefern.

- Algorithmus
  1. Rauschunterdrückung: Gauß-Filter
  2. Berechnung der Gradienten in horinzontaler und in vertikaler Richtung (Prewitt/Sobel)
  3. Non-Maximum Surpression
     - Gradient muss lokales Maximum sein
     - Betrachtung der zwei direkten Nachbarn entlang der Gradientenrichtung
     - Überprüfung erfolgt gemäßt dem jeweiligen Quadranten
  4. Hysterese-Schwellwertverfahren
     a) Verwendung von zwei Schwellwerten: low/high
     b) Liegt der Betrag des Gradienten für ein Pixel über dem Schwellwert high, so wird es auf den Fall als Teil einer Kante akzeptiert
     c) Liegt der Betrag des Gradienten für ein Pixel unter dem Schwellwert low, so wird es als Teil einer Kante abgelehnt
     d) Ausgehend von den akzeptierten Pixeln werden Kanten (rekursiv) verfolgt:
        - Überprüfung der acht direkten Nachbarn
        - Betrag Gradient muss über Schwellwert low liegen

Visual Servoing

- Der Ausruck Visual Servoing beschreibt Verfahren, bei denen visuelle Eingabedaten genutzt werden, um die Bewegung eines Roboters zu steuern.
- Motivation
  - Modellfehler (z.B. Kinematik)
  - Fehler bei der Ausführung (z.B. fehlerhafte Positionierungen der Robotergelenke)
  - Überwachung der Szene (z.B. Reaktionen auf Kollisionen)
- Systemaufbau
  - Kamera-in-Hand (eye-in-hand)
    * Kamera ist an dem Manipulator angebracht
    * Bewegungen des Manipulators beeinflussen die Pose der Kamera
  - Externes (festes) Kamera System (eye-to-hand)
    * Externes Kamerasystem wird zur Beobachtung der Bewegung genutzt
- Verfahren

  - Positionsbasiertes Visual Servoing
    * Position-based visual servoing (PBVS)
    * Einfache Regler
    * 3D Rekonstruktion aufwendig
  - Bildbasiertes Visual Servoing
    * Image-based visual servoing (IBVS)
    * Merkmalsextraktion einfach
    * Komplexere Regelungsvorschriften
  - Hybride Verfahren (z.B. 2,5D visual servo)

Positionsbasiertes Visual Servoing

- Zielpose X_g ist vorgegeben
- Ablauf der Regelschleife
  - 3D Lage Schätzung: Aktuelle Pose X_c des Zielobjektes (bzw. Hand) wird aus Bildmerkmalen extrahiert
  - Regelvorgabe (kartesisch): Differenz $\Delta X = X_g - X_c$
  - Ziel erreicht: Distanz ΔX Kleiner als Schwellwert

Bildbasiertes Visual Servoing

- Ansatz: Die Bewegung des Manipulators ergibt sich aus der aktuellen und gewünschten Position von Bildmerkmalen
- Bildmerkmale: Bildverarbeitungsmethoden zur Extraktion von Bildmerkmalen
- Regelung: Geschwindigkeitsvorgaben werden direkt aus der aktuellen und gewünschten Stellung der Bildmerkmale erzeugt
- Bildmerkmal: Ein Bildmerkmal $s = (u, v)^T$ ist die Projektion eines 3D Punktes $P = (x, y, z)^T$ im Kamerabild.
- Fehlerfunktion
  - Interaction Matrix/Image Jacobian: Die interaction matrix L beschreibt die Beziehung zwischen der Bewegung eines Bildmerkmals $s = (u, v)^T$ und des entsprechenden 3D Punktes $P = (x, y, z)^T$
  - Aus den Projektionsvorschriften (Lochkameramodell) ergibt sich L.
- Invertierung der Interaction Matrix
  - Distanz z wird geschätzt
  - Mehrere Merkmale werden beobachtet (mind. 3)

  - Annahme: Kamera-In-Hand System - Bewegung der 3D Punkt korrespondiert mit Bewegung der Kamera (Geschwindigkeit der Kamera v_c)
  - Daraus folgt $\dot{e} = Lv_c$
  - Mit $e = -\lambda e$ ergibt sich $v_c = -\lambda L^+ e$
  - Hierbei ist L^+ die Pseudoinverse von L

Point Clouds

- Eine Punktwolke ist eine diskrete Menge von 3D-Punkten mit einem festen Koordinatensystem.
- Punktwolke $P = \{(X, C) | X \in \mathbb{R}^3, C \in [0...255]^3\}$
  - $X = (x, y, z)$ Ortsinformationen
  - $C = (r, g, b)$ Farbinformationen
  - Es können weitere (Sensor-)Informationen abgespeichert werden
- Zwei verschiedene Arten der Repräsentation
  - Organisierte Punktwolke (2D Array Format, Größe muss vorab bekannt sein)
  - Unorganisierte Punktwolke (Vector Format)

Normalenschätzung in 3D Punktwolken
Ziel:

- Zusätzliche Oberflächeninformationen durch Einbeziehung von lokalen Nachbarpunkten
- Grundlage für weiterführende Algorithmen
  - Segmentierung
  - Deskriptoren
  - Objekterkennung
  - Oberflächenmodellierung
- PCA-basierter Ansatz
- Erstelle die Kovarianzmatrix C der k-Punktnachbarschaft für jeden Punkt p: $C = \frac{1}{k} \sum_{1}^{k} (p_o - \bar{p}) \cdot (p_i - \bar{p})^T$
- Bestimme die Eigenwerte und Eigenvektoren von C
  - Hauptkomponentenanalyse (PCA)
  - Eigenvektor zu kleinstem Eigenwert korrespondiert mit der Normalen

Registrierung von Punktwolken

- Registrierung: Zusammenführend von Punktwolken, welche das gleiche Objekt aus unterscheidlichen Ansichten beschreibt
- Überführung in ein übergeordnetes Koordinatensystem (z.B. Weltkoordinatensystem)
- Extrinsische Kalibrierung der Kameras erforderlich

Iterative Closest Point

- Iterative Closest Point (ICP) ist ein gängier Algorithmus für die Registrierung zweier Mengen A, B mit a priori unbekannter Zuordnung
- Beispiel: Registrierung zweier 3D Punktwolken
  - Für jede Iteration k gilt:
    - Für jeden Punkt a_i aus A suche Punkt b_i aus B, der a_i am nächsten liegt
    - Berechne eine Transformation T_k so dass D_k minimal wird, z.B. mit $D_k = \sum_i ||a_i - T_k \cdot b_i||^2$
  - Wende Transformation T_k auf alle Punkte aus B an
  - Abbruchkriterien:
    - Schwellwert für $D_{k-1} - D_k$
    - Maximale Anzahl an Iterationen erreicht
- Minimiert die Distanz zwischen zwei Punktwolken
- Sehr gut geeignet für die Rekonstruktion in 2D und 3D
- Vorteile
  - Algorithmus für Punkte, Normalenvektoren und andere Darstellungsformen anwendbar
  - Nur einfache mathematische Operationen notwendig
  - Schnelles Registrierungsergebnis
- Nachteile
  - Symmetrische Objekte können nicht ohne weiteres registriert werden
  - Konvergenz in ein lokales Minimum möglich
  - Überlappung der Punktwolken erforderlich

RANSAC (Random Sample Consensus)

- RANSAC ist eine iterative Methode zur Schätzung von Modellparametern aus Datenpunkten
- RANSAC ist ein nicht-deterministischer Algorithmus

- Robust gegenüber Ausreißern und fehlenden Datenpunkten
- Anwendung in der Bildverarbeitung
  - Schätzung von Linien in 2D Bildern
  - Schätzung von Ebenen und anderen Primitiven in 3D Punktwolken
- Der RANSAC-Algorithmus
  1. Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
     - 2 Punkte für Linien in 2D
     - 3 Punkte für Ebenen in 3D
  2. Schätze ein Modell aus dem ausgewählten Datensatz
  3. Bewertung der Modellschätzung: Berechne die Teilmenge der Datenpunkte (Inliers), deren Abstand zum Modell kleiner ist als ein vordefinierter Schwellwert
  4. Wiederhole 1-3 bis das Modell mit den meisten Inliers gefunden wird
- Die voraussichtliche Anzahl an Iterationen k damit der Algorithmus mit der Wahrscheinlichkeit P erfolgreich terminiert kann wie folgt berechnet werden:
  - Wahrscheinlichkeit für einen Inlier: $P(inlier) = w$ (w = Anzahl der Inlier/Gesamtzahl der Samples)
  - Wahrscheinlichkeit ein Inlier-Modell zu ziehen: $P(subsetwithnooutlier) = w^n$
  - Wahrscheinlichkeit ein Outlier-Modell zu ziehen: $P(subsetwithoutliers) = 1 - w^n$
  - Wahrscheinlichkeit in k Iterationen ein Outlier-Modell zu ziehen: $P(ksubsetswithoutliers) = (1 - w^n)^k$
  - Wahrscheinlichkeit für einen erfolglosen Durchlauf: $P(fail) = (1 - w^n)^k$
  - Wahrscheinlichkeit für einen erfolgreichen Durchlauf: $P(success) = 1 - (1 - w^n)^k$
  - Vorraussichtliche Anzahl an Iterationen: $\implies k = \frac{log(1-P(success))}{log(1-w^n)}$
- RANSAC benötigt mehr Iterationen im Fall von vielen Outliern
- Vorraussichtliche Anzahl an Iterationen: $\implies k = \frac{log(1-P(success))}{log(1-w^n)}$
- Minimale Sample Anzahl n muss vorher definiert sein!
- $P(success)$ muss hoch angesetzt werden (d.h. $\geq 95\%$)
- Vorteile:
  - Simpel, allgemein und einfach zu implementieren
  - Robuste Modellschätzung für Daten mit wenigen Ausreißen
  - Vielseitig anwendbar

- Nachteile:
  - Nicht-deterministisch
  - Viele Parameter
  - Trade-off zwischen Genauigkeit und Laufzeit (benötigt viele Iterationen)
  - Nicht anwendbar wenn das Verhältnis Inliers/Outliers zu klein ist

Simultaneous Localization and Mapping (SLAM)

- Das SLAM Problem: Wie kann ein Roboter in einer unbekannten Umgebung navigieren und dabei eine Karte von seiner Umgebung erstellen und auch aktualisieren?
- SLAM bedeutet die Schätzung der aktuellen Roboterpose als auch der Karte der Umgebung zur gleichen Zeit.
- SLAM wird benötigt
  - um voll autonome Roboter zu bauen
  - um in einer unbekannten Umgebung zu überleben
  - um zu wissen wo sich der Roboter gerade befindet
  - um die Navigation ohne externe Positionsbestimmung (z.B. GPS) zu ermöglichen.
- Es entsteht ein Henne-Ei-Problem. Wir brauchen eine Karte, um den Roboter zu lokalisieren (aber SLAM hat kein Vorwissen über die Umgebung) und eine genaue Schätzung der Position, um eine Karte zu erstellen
- Begriffe:
  - Lokalisierung: Schätzen der Roboterposition bei gegebener Karte
  - Katierung: Aus einer Menge von Positionen einer Karte ableiten
  - SLAM: Gleichzeitige Lokalisierung und Kartierung
- Lösung: Position von Kamera und Roboter während einer Bewegung verfolgen
- Deshalb braucht der Roboter Features, die er zuordnen und verfolgen kann
  - SLAM wählt Szenenfeatures als Orientierungspunkte
  - Visuelle Features: Punkte, Ecken, Kanten, Texturen, Oberflächen
  - Hinweis: Features müssen unterscheidbar und dekriptiv sein (invariant gegenüber Standpunktwechsel)
- Wie funktioniert SLAM?
  - Interne Repräsentation für:
    * Position der Orientierungspunkte
    * Parameter der kamera

- In jedem Frame:
  - Vorhersage, wie viel sich der Roboter bewegt hat
  - Neue Orientierungspunkte aufnehmen
  - Interne Repräsentation aktualisieren (Messungenauigkeiten)

12 Roboterprogrammierung

12.1 Klassische Roboterprogrammierverfahren (Übersicht)

Klassifizierung der Roboterprogrammierverfahren - Kriterien:

1. Programmierort
   - on-line/direkt: Die Programmierung erfolgt direkt am Roboter (an der Robotersteuerung)
     - Einstellen des Roboters
       - Ältestes Programmierverfahren
       - Der Bewegungsbereich jedes Gelenks wird durch Stopper eingeschränkt
       - Die Bewegungen erfolgen für jedes Gelenk einzeln bis zum Anschlag
       - Zuordnung zwischen Anfahrpunkten zu Stopper kann mit Codiermatrizen erfolgen
       - Sog.: Bang-Bang-Robot
       - Nachteil: Sehr kleine Menge von Anfahrpunkten
     - Teach-In Programmierung
       - Anfahren markanter Punkte der Bahn mit manueller Steuerung (Teach Box, Teach Panel)
       - Funktionalität einer Teach Box:
         - Einzelbewegung der Gelenke
         - Bewegung des Effektors in 6 Freiheitsgraden
         - Speichern/Löschen von Anfahrpunkten
         - Eingabe von Geschwindigkeiten
         - Eingabe von Befehlen zur Bedienung des Greifers
         - Starten/Stoppen ganzer Programme
       - Weitere Eingabegeräte: Maus, Joystick, Spacemouse (Teach-Kugel)
       - Vorgehensweise beim Teach-In
         - Anfahren markanter Punkte der Bahn
         - Bahn = Folge von Zwischenpunkten

        - Speichern der Gelenkwerte
        - nachträgliche Ergänzung der gespeicherten Werte um Parameter wie Geschwindigkeit, Beschleunigung usw.
        - Anwendung in der Fertigungsindustrie (Punktschweißen, Nieten) und Handhabungsaufgaben (Pakete vom Fließband nehmen)
- Play-Back Programmierung (manuelle Programmierung)
    * Einstellung des Roboters auf Zero-Force-Control (Roboter kann durch den Bediener bewegt werden)
    * Auch kinästhetisches Lernen genannt
    * Abfahren der gewünschten Bahn
    * Speichern der Gelenkwerte:
        - automatisch (definierte Abtastfrequenz) oder
        - manuell (durch Tastendruck)
    * Anwendung:
        - mathematisch schwer beschreibbare Bewegungsabläufe
        - Integrierung der handwerklichen Erfahrung des Bedieners
    * Nachteile:
        - schwere Roboter schwierig zu bewegen
        - direkter Kontakt mit Roboter -> Sicherheitsrisiko
        - hoher Speicherbedarf (bei hoher Abtastrate)
        - schlechte Korrekturmöglichkeiten
- Master-Slave Programmierung (mit Sonderfall Teleoperation)
    * Bediener führt einen kleinen, leicht bewegbaren Master-Roboter (entspricht einem kinematischen Modell des Slave-Roboters)
    * Bewegung wird auf den Slave-Roboter übertragen
    * Bewegungen werden synchron ausgeführt
    * Slave-Roboter wirkt als Kraftverstärker
    * Anwendung: Handhabung großer Lasten bzw. großer Roboter
    * Vor-und Nachteile: teuer, da zwei Roboter benötigt werden; Möglichkeit, auch schwerste Roboter zu programmieren
- Sensorunterstützte Programmierung
    * Manuell
        - Bediener führt Programmiergriffel (Leuchtstift, Laserstift) entlang der abzufahrenden Bahn

            - Erfassung der Bewegung durch externe Sensoren (z.B. Kameras, Laserscanner)
            - Berechnung der inversen Kinematik
            - Abspeichern der Bahn als Folge der Gelenkwinkel
        * Automatisch
            - Vorgabe des Start- und Zielpunktes
            - Sensorische Ertastung der Sollkontur (z.B. über Kraft-Momenten-Sensor)
        * Anwendung: Schleifen, Entgraten von Werkstücken
    - off-line/indirekt: Die Programmierung erfolgt ohne den Roboter mit Hilfe textueller, graphischer, interaktiver Methoden.

2. Art der Programmierung
    - Direkte Programmierung
        - Vorteile: schnell bei einfachen Trajektorien; sofort anwendbar; geringe Fehleranfälligkeit; Bediener benötigt keine Programmierkenntnise; kein Modell der Umwelt erforderlich
        - Nachteile: hoher Aufwand bei komplexen Trajektorien; nur mit und am Roboter möglich; spezifisch für einen Robotertyp; Verletzungsgefahr durch Roboter; keine Adaption an neue Gegebenheiten
    - Textuelle Verfahren
        - Programmierung erfolgt mittels erweiterter, höherer Programmiersprachen wie PASRO, VAL, V+ (Unimation/Stäubli), RAPID (ABB), KRL (KUKA),... -> Robotersteuerung
        - Vorteile:
            * Programmierung kann unabhängig vom Roboter erfolgen
            * strukturierte, übersichtliche Programmierlogik
            * Erstellung komplexer Programme (Einbezug von Wissensbasis, Weltmodell, Auswertung von Sensoren)
        - Nachteile: Bediener benötigt Programmierkenntnise; keine/schlechte Korrekturmöglichkeiten
        - DIN 66024
            * Befehlscodierung nach DIN 66025
            * Programm = Menge numerierter Sätze
            * Sprachen: APT (Automatically Programmed Tools), EXAPT (Extended Subset of APT)
        - SPS

        * VPS: Verbindungprogrammierte Steuerung (historisch): Steuerung erfolgt über Hardware; Programmänderung = Hardwareänderung
        * SPS: Speicherprogrammierbare Steuerung: Steuerungs- und Regelungsablauf werden programmiert; große Flexibilität
    - CNC: Computerized Numerical Control
        * Steuerung von Werkzeugmaschinen: geometrische Beschreibung des Werkstücks bzw. der Bearbeitungsflächen + Kurven (Bahnkurve des Werkzeugs); technologische Beschreibung (z.B. Vorschubgeschwindigkeit oder Spindeldrehzahl)
        * Steuerungsarten: Punktsteuerung (Bohrung); Achsenparallele Steuerung (Fräsen); Bahnsteuerung z.B. Strecke oder Kreis (Brennschneidenmaschinen)
        * Sprachen: APT, EXAPT
- Graphische Verfahren
    - Virtuelles Teach-In
        * Manipulation von Roboter und Umgebung in 3D Visualisierung
        * Abspeichern der Bewegungen
        * Benötigt exaktes 3D Modell von Roboter und Umgebung
        * Vorteile
            · Programmierung kann unabhängig vom Roboter erfolgen
            · Programmierer benötigt weniger Programmierkenntnisse
            · einfache Programmierung, leichte Fehlererkennung
            · schnelles Erstellen komplexer Programme (rapid porototyping)
        * Nachteile
            · Leistungsfähige Hardware für Visualisierung und Simulation
            · Komplexe (Rechen-)Modelle für eine realitätsnahe Simulation benötigt
            · Roboter und Umwelt müssen modelliert werden!
    - Graphische Modellierungsformalismen
        * Endliche Zustandsautomaten
        * Petri-Netze
        * Statecharts
- Gemischte Verfahren
    - Graphische Programmierung basierend auf sensorieller Erfassung der Benutzervorführung -> Simulation der Roboterprogramme

  - Vorteile
    * Programmierer benötigt weniger Programmierkenntnisse
    * einfache Programmierung, leichte Fehlererkennung
    * schnelles Erstellen komplexer Programme (rapid prototyping)
  - Nachteile
    * sensorielle Benutererfassung noch zu ungenau
    * Leistungsfähige Hardware für Signalanalyse, Modellierung,...
    * Komplexxe Modelle benötigt
- Abstraktionsgrad der Programmierung
  - Implizite Programmierung: Die Aufgabe, die der Roboter durchführen soll, wird beschrieben, z.B. in Form von Zuständen.
  - Explizite Programmierung: Bewegungen und Greiferbefehle sind direkt in eine Programmiersprache eingebunden.

12.2 Graphisches Programmierverfahren: Statecharts

- Warum Statecharts zur Programmierung von Robotern?
- Lernen von Aktionen aus Beobachtung schwierig
  - Perzeption
  - Embodiment
  - Ausführungsunsicherheiten
- Textuelle Aktionsprogrammierung schwierig
  - Systemkomplexität
  - Roboterfähigkeiten stark zustandsbehaftet
  - Fähigkeiten bestehen zumeist aus Subfähigkeiten
  - Textuelle Programmierung unübersichtlich
- Graphische Programmierung von Roboteraktionen: Statecharts

Harel Statechart Formalismus

- Graphischer Formalismus zum Entwurf komplexer Systeme
- Wichtigste Features:
  - Hierarchisch
  - Interlevel Transitionen
  - Orthogonalität

  - Zustandsaktionsphasen: Entry, Exit, Throughout

- Limitation: Keine Datenflussspezifikation
  - Datenfluss in der Robotik ist notwendig; Ergebnisse von Zuständen werden in Folgezuständen benötigt
    * Beispiel: Objektlokalisierung wird für Visual Servoing oder inverse Kinematik benötigt
  - Wiederverwendbarkeit erfordert eine Adaption von Parametern
    * Kontrollparameter
    * Kinematikparameter
    * Objektparameter
    * ...

Erweiterung des Harel Statechart Formalismus am H^2T

- Datenflussspezifikation: Transitionsbasierter Datenfluss
- Keine Inter-level Transition, da sie die Widerverwendbarkeit verhindern
- Erfolgs- und Misserfolgszustände in jedem Zustand
- Dynamische Struktur
- Verteilung über mehrere Host-PCs
- Verbindung von graphischer Kontroll- und Datenflussspezifikation mit C++ Benutzer-Code

Statechart Erweiterungen

- Transitionsbasierter Datenfluss
  - Beliebige Datentypen (grundlegende: int, float, string, ... + komplexe: Position, 6D-Pose, Matrizen, Listen, ...)
  - 3 Parameter Container pro Zustand (Eingabe, Lokale Parameter, Ausgabe)
  - Parameterabbildung pro Transition (Abbildung von Quell-Parameter auf Eingabe-Parameter des Zielzustandes)
  - Spezifizierter Datenfluss → Keine Seiteneffekte durch globale Variablen
- Verteilung über mehrere Host-PCs
  - Transparente Verteilung von Statecharts (Netzwerk Middleware Zero Ice)
  - Kind-Zustände können Zeiger auf entfernt liegende Zustände sein (grüner Zustand)
  - Vorteile:

        * Lastverteilung
        * Erhöht Robustheit und Fehlertoleranz
        * Näher an eingesetzter Hardware (Sensorik, Aktorik)
- Dynamische Struktur
    - Austausch von Unterzuständen zur Laufzeit
    - Transparente Verbindungen zu beliebigem entfernten Zustand
    - Unterzustand durch Parameterabbildung spezifiziert
    - Einsatz: Ausführung von generierten Plänen

12.3 Symbolische Planung

- Plan in der Robotik: Sequenz von parametrisierten Aktionen zum Erreichen eines definierten Ziels
- Symbolisch: Repräsentationen des Weltzustandes durch Boolesche Prädikaten statt kontinuierliche Werten
- Planung: Finden einer Aktionssequenz zur Lösung eines Zielzustandes
- Klassisches Planungsbeispiel: Blockwelt
- Herausforderung: Wie kann man das Planungsproblem so formulieren, dass eine Lösung einfach/schnell zu finden ist und die Existenz einer Lösung bewiesen/wiederlegt werden kann?
- Klassischer Lösungsansatz: Wissensmodellierung (Beschreibssprache der Umwelt und Aktionen (STRIPS,ADL,PDDL,...); Suche: Suchalgorithmus zum Finden der gültigen Aktionsseuqenz)
- Probleme: Suchraum sehr groß (Branching factor); Abbildung der realen Welt auf Symbole (Symbole Grounding Problem)

STRIPS (Standford Research Institute Problem Solver)

- Bekannteste und eine der ältesten Sprachen zur Beschreibung von Planungsdomänen. Wurde in 1971 von Fikes und Nilson zur Steuerung des Roboters „Shakey“ entwickelt
- Sehr einfach, daher auch eingeschränkt
- Action Description Language (ADL) und Planning Domain Definition Language (PDDL) sind von STRIPS abgeleitet
- Bestandteile von STRIPS (auch Planungsdomäne genannt): Zustände, Aktionen, Ziele
- Symbole

  - Funktionsfreie Sprache erster Ordnung (Prädikatenlogik erster Stufe): Endliche Anzahl an Prädikaten und Konstanten, keine Funktionssymbol, endliche Anzahl an Operatorensymbolen

- Zustände
  - Konjunktionen positiver aussagenlogischer Literale aus Prädikaten und Konstantensymbole
  - Einschränkungen: Endliche Anzahl an Literalen, keine Variablen, keine negativen Literale, keine Funktionen
  - Geschlossene Welt (Closed World Assumption): Jeder Zustand muss vollständig bekannt sein und beschrieben werden können, nicht vorkommende Literale gelten als negative (negierte) Literale

- Ziele
  - Ein Ziel ist ein teilweise spezifizierter Zustand: Konjunktion von positiven Literalen
  - Erfüllung von Zielen: Alle Literale des Zieles müssen im Weltzustand vorkommen

- Aktionen: Tripel bestehend aus (Deklaration, Vorbedingungen, Effekte)
  - Deklaration: Name und Parameterliste (Beispiel: Greife(h,o,l), eindeutiger Name, Parameterliste muss alle Variablen enthalten, die in den Vorbedingungen und Effekten benutzt werden)
  - Vorbedingungen (Preconditions): Konjunktionen von Literalen, die wahr sein müssen, um einen Operator aufwenden zu können (Negative Literale erlaubt, nur Variablen in Literalen)
  - Effekte (Effects): Auswirkungen der Aktion auf den Weltzustand, Liste von positiven oder negativen Literalen; auch bekannt als Hinzufüge-Listen (add-List) oder Lösch-Listen (Delete-List) von positiven Literalen

- Ausführbarkeit
  - Eine Aktion A ist ausführbar in allen Zuständen, die die Vorbedingung erfüllen
  - Ergebnis der Ausführung: Das Ergebnis der Ausführung der Aktion auf einem Zustand erhält man durch Entfernen aller negativen Literale des Effekts aus dem Zustand und Hinzufügen aller positiven Literale des Effekts zum Zustand

- Vorteile
  - Einfache Beschreibungssprache
  - Einfache Planung
  - Lösbarkeitsbeweis einfach

- Nachteile

  - Viele Restriktionen beim Modellieren (Geschlossene Welt, nur positive Literale in Zuständen (unbekanntes Wissen nicht modellierbar), keine Quantoren und Disjunktionen in Zielen, Vorbedingungen, Effekte, keine Typisierung (nur mit Prädikaten umsetzbar), nur boolesche Prädikate)

- ADL und PDDL sind mächtiger, erlauben bessere Modellierung, sind aber auch komplexer

- Suche im Zustandsraum
  - Suche: Anwenden von ausführbaren Aktionen
  - Suche ist einfach umzusetzen
    * Keine Funktionssymbole (endlicher Zustandsraum, Standard-Algorithmus zur Baumsuche möglich)
    * Transformation der Vorbedingung (Effekte bijektiv -> Suche auch rückwärts möglich)
  - Problem: Suchraum ist sehr groß (Branching factor: Anzahl möglicher Verzweigungen pro Zustand, Aktion mit n Parametern und m Konstanten in der Domäne hat m^n Aktionskandidaten. Prüfung auf Ausführbarkeit -> Alle ausführbaren Aktionen werden weiter verfolgt)
  - Suchalgorithmen:
    * Tiefensuche findet meistens nicht den kürzesten Plan
    * Breitensuche findet immer den kürzesten Plan, hoher Speicherbedarf -> oft nicht anwendbar
    * Heuristik basierte Suche: Güte der Heuristik ist ausschlaggebend, bisher keine allgemeingültige Heuristik
    * FastForward-Planer: gierige Vorwärtssuche mit Heuristik, Breitensuche um aus lokalen Minima zu entkommen, findet nicht immer den kürzesten Plan, aber in viele Fällen einen kurzen
    * Planungsgraphen: Analyse der Prädikate und Aktionen um eine Heuristik aufzustellen

13 Programmieren durch Vormachen

- Hauptziel von Programmieren durch Vormachen
  - Intuitive Roboterprogrammierung ohne Expertenwissen
  - Lernen von Aufgaben durch Generalisierung auf Basis mehrerer Observationen

- PdV impliziert Lernen und Generalisierung und ist deshalb kein Playback Verfahren.

- Drei-Phasen-Modell

  - Zerlegung in Teilprobleme
  - Lösung in getrennten Modulen
  - Austausch von Modulen möglich
  - Kette: Perzeption -> Kognition -> Aktion

Drei Gründe für PdV:

- Mächtiger Mechanismus zur Komplexitätsredukation des Suchraums während des Lernens (im Gegensatz zum Ausprobieren aller Möglichkeiten).
  - Gute Observationen können als Startpunkt gewählt werden
  - Schlechte Observationen können direkt aus dem Suchraum eliminiert werden
- Implizierter Mechanismus zum Trainieren durch Maschinen, der das explizite und mühsame händische Programmieren durch Menschen reduziert oder sogar ganz eliminiert.
- Hilft beim Verständnis der Kopplung zwischen Perzeption und Aktion dem lernen relvenater Beziehungen zwischen diesen

Hauptherausforderungen in PdV

- Wer soll imitiert werden?
  - Lehrerauswahl: wähle einen Lehrer von dessen Verhalten der Imitierende am meisten profitiert.
  - Problem wird bei den meisten Methoden durch Auswahl eines dedizierten Lehrers vermieden.
- Wann soll imitiert werden?
  - Der Imitierende muss die Bewegung segmentieren und Start und Ende des gezeigten Verhaltens ermitteln.
  - Der Imitierende muss entscheiden, ob es angemessen ist das beobachtete Verhalten im aktuellen Kontext auszuführen und wie oft es wiederholt werden muss.
  - Problem wird bei den meisten Methoden durch explizites Kennzeichnen von Start und Ende der Demonstration vermieden.
- Was soll limitiert werden?
  - Wie findet man heraus welche Aspekte der Demonstrationen von Interesse sind? Bei Aktionen sind es Bewegungen (Trajektorien) des Vorführenden. In anderen Situationen werdend as Ergebnis und die Effekte von Aktionen als wichtig erachtet.
  - Manche beobachtbaren Eigeschaften sind irrelevant und können problemlos ignoriert werden. Ist es zum Beispiel wichtig, dass der Vorführende immer von Norden her zum Tisch läuft?

  - Bei kontinuierlichen Regelungsaufgaben entspricht dise Aufgabe der Bestimmung des Merkmalraums für das Lernen, sowie der Constraints und der Kostenfunktion.
  - Bei diskreten Regelungsaufgaben, die z.B. mit Reinforcement Learning und symbolischem Reasoning behandelt werden, entspricht diese Aufgabe der Definition des Zustands- und Aktionsraums und wie dem automatischen Lernen von Vor- und Nachbedingungen (Constraints).

- Wie soll imitiert werden?
  - Bestimme, wie der Roboter das gelernte Verhalten tatsächlich ausführen wird, um die Metrik zu maximieren, die beim „Was soll imitiert werden?“ Problem gefunden wurde. Auf Grund unterschiedlicher Kinematik kann z.B. ein Roboter menschliche Bewegungen nicht identisch reproduzieren. Darf z.B. ein Roboter mit Rädern ein Objekt anfahren, wenn der Lehrer es mit dem Fuß angestoßen hat, oder muss er stattdessen einen Arm einsetzen?
  - Der Roboter muss eine Abbildung von Perzeption zu einer Sequenz eigener Bewegungen lernen. Deshalb bestimmen das Embodiment und die Körpereinschränkungen wie beobachtete Aktionen imitiert werden können (Korrespondenzproblem).

Zwei unterschiedliche Arten wie bestimmt werden kann, ob der Zustand des Lehrers (Mensch) und des Imitators (Roboter) korrelieren

- Perzeptive Äquivalenz: Auf Grund von Unterschieden zwischen menschlicher und robotischer Sensorfähigkeiten kann die gleiche Szene sehr unterschiedlich aussehen. Ein Mensch erkennt beispielsweise andere Menschen und Gesten anhand von Farbe und Intensität, während ein Roboter Tiefeninformationen dazu verwendet.
- Physikalische Äquivalenz: Auf Grund unterschiedlicher Embodiements führen Menschen und Roboter z.B. unterschiedliche Aktionen aus, um den gleichen physikalischen Effekt zu erzielen.

13.0.1 Perzeption

Aufnahme der Demonstration

- Beim Lernen: Übertragung der aufgenommenen Aktion vom Lehrer auf den Roboter
- Zwei Schritte:
  - Aktionen des Lehrers aufnehmen
  - Entsprechende Aktion des Roboters bestimmen (Direkte Abbildung oder Abbildungsvorschrift notwendig)
- Beide Schritte können direkt oder indirekt durchgeführt werden
- Es gibt also vier Typen von Lernen durch Vormachen:

  - Teleoperationen
  - Shadowing
  - Sensoren am Lehrer
  - Externe Beobachtung

Schnittstellen zum Vormachen

- Erste Zeile: Aufgenommene Daten können direkt vom Roboter genutzt werden
- Zweite Zeile: Benötigte Daten sind nur indirekt verfügbar und müssen aus den aufgenommenen Daten abgeleitet werden.
- Linke Spalte: Die vorgemachten Daten entsprechen dem Körper des Roboters und können direkt benutzt werden
- Rechte Spalte: Die vorgemachten Daten entsprechen nicht dem Körper des Roboters, weshalb eine Abbildung auf den Roboter notwendig ist
- Teleoperation
  - Lehrer steuert den Roboter direkt mittels Joysticks oder anderen Remote-Steuerungsgeräten
  - Die Sensoren des Roboters (z.B. Position, Kraft) nehmen die Demonstration auf
  - Embodiments des Lehrers und Schülers sind die selben (der Roboter selbst) -> keine Nachverarbeitung nötigt (kein Korrespondenzproblem)
  - Aber: Benutzung des Steuerungsgeräts benötigt oft Training, da oft 6+ Freiheitsgrade gesteuert werden
- Teleoperation vs Kinästhetisches Lernen
  - bei kinästhetischem Lernen wird der Roboter direkt durch physikalische Interaktion bewegt (setzt Compliance (aktiv/passiv) beim Roboter voraus)
  - Aufnahme der Demonstration wie bei Teleoperation
  - Mensch braucht für gewöhnlich mehr DoF als der Roboter, um eine Aktion zu demonstrieren (zweihändige Aktionen nur schwer möglich)
  - Mensch muss sich im Arbeitsbereich des Roboters befinden (Sicherheitsrisiko, nicht von Entfernungen möglich)
- Shadowing
  - Lehrer und Schüler haben gleiches Embodiment
  - kein direkter Zugriff auf die Daten des Lehrers
  - Nachahmung der Aktionen des Lehrers (durch Beobachtung) und Aufnahme der Aktion mit eigenen Sensoren
  - Daher ist die Aufnahme indirekt

  - Aufnahme direkt im Embodiment des Schülers

- Sensoren auf dem Lehrer
  - Bewegung des Lehrers werden direkt aufgenommen und der Roboter hat direkten Zugriff -> Explizite und direkte Aufnahme
  - hohe Genauigkeit der Aufnahmen
  - Mensch kann sich frei bewegen
  - Abbildung vom Menschen auf Roboter nötig (Korrespondenzproblem)

- Externe Beobachtungen
  - Beobachtung durch Sensoren des Roboters (Kameras)
  - Komplexe Algorithmen nötig zur Bildverarbeitung
  - Niedrige Präzision
  - Haptik des Lehrers kann nicht aufgenommen werden
  - Abbildung vom Menschen auf Roborer nötig (Korrespondenzproblem)

Erfassung der menschlichen Bewegung

- Die Erfassung der menschlichen Bewegung ist Ausgangspunkt für die Programmierung durch Vormachen

- Marker-basierte Verfahren:
  - Marker werden an vorher festgelegten anatomischen Landmarken des menschlichen Körpers angebracht
  - Marker sind entweder aktiv (z.B. Codierung mit LED) oder passiv (z.B. reflektierend)
  - Mensch muss zur Erfassung vorbereitet (vermarkert) werden

- Marker-lose Verfahren:
  - Direkte Rekonstruktion der menschlichen Pose aus Kamera-Daten (RGB-/Tiefendaten)
  - Keine Vermarkerung des Menschen erforderlich

Marker-basierte optisch-passive Bewegungserfassung

- Lokalisierung von infrarot-reflektierenden passiven Markern mit Hilfe eines Mehr-Kamera-Systems

- Weit verbreitete Lösung, verschiedene kommerzielle Anbieter

- Vorteile: hohe räumliche Genauigkeit, hohe zeitliche Auflösung

- Nachteile: teuer und hoher technischer Aufwand, Aufwand für Nachbearbeitung durch Verdeckung und Labeling der Marker, Anforderungen an Aufnahmebereich

Marker-basierte optisch-aktive Bewegungserfassung

- Marker übertragen eine codierte Kennung, z.B. mir Infrarot-LED
- Vor-/Nachteile ähnlich zu optisch-passiven Systemen, aber vereinfachte Handhabung des Marker Labelings, Dafür: Batterien/Kabel zur Stromversorgung jedes Markers notwendig

IMU-basierte Bewegungserfassung

- Anbringung von inertialen Messenheiten (IMUs) an anatomisch definierten Punkten des menschlichen Probandes
- Vorteile: Geringe Anforderungen an Aufnahme-Umgebung, genaue Bestimmung der menschlichen Pose
- Nachteile: Exakte Platzierung der IMUs erforderlich, keine exakte Positionsbestimmung des Menschen im global Koordinatensystem möglich

Mechanische Bewegungserfassung

- Direkte Messung von Gelenkwinkeln (z.B. Potentiometer)
- Vorteile: Geringe Anforderungen an Aufnahme-Umgebung
- Nachteile: Einschränkung der menschlichen Bewegung durch Exoskelett, hoher technischer Aufwand, keine Positionsbestimmung des Menschen im globalen Koordinatensystem möglich

Magnetische/akustische Bewegungserfassung

- Lokalisierung von Markern mit elektromagnetischen oder akustischen (Ultraschall) Prinzipien
- Vorteile: Günstiger als marker-basierte optische Lösungen, geringe Probleme mit Verdeckungen
- Nachteile: Stark begrenzte Reichweite, Empfindlichkeit ggü. Störungen, teilweise hohe Ungenauigkeiten

Master Motor Map (MMM) Framework

- Framework zur vereinheitlichenden Darstellung menschlicher Ganzkörperbewegung
- Erfasste Bewegungenn unterschiedlicher Menschen und Erfassungstechniken werden auf Referenzmodell des menschlichen Körpers mit 104 Bewegungsfreiheitsgraden abgebildet

Marker-lose optische Bewegungserfassung

- Rekonstruktion der menschlichen Pose aus RGB- und/oder Tiefen-Daten

- Vorteile: Sehr günstig, geringer Hardware-Aufwand, geringe Anforderungen an Aufnahme-Umgebung
- Nachteile: Komplexe Algorithmen und hohe Fehlerrate, niedrige zeitliche Auflösung, vor allem bei Online-Einsatz

Skeleton Tracking

- Tiefenkameras liefern Tiefenbilder und Skelettdaten mit 30 fps
- Skelettdaten werden aus jedem Einzelbild berechnet
- Genauigkeit zwischen RGB-Tracking und Marker-basiert

Segmentation & Tracking

- Tracking von Lehrer und Objekten durch Farb- und Tiefenregionen
- Optischer Fluss (Bewegungsmuster) wird benutzt, um Bewegungen zwischen 2 Frames zu tracken
- Segmentmittelpunkte pro Frame stellen die Trajektorie dar

Interfaces for Demonstration

- Demonstration für Teleoperation und Shadowing
- Imitation für Sensoren auf dem Lehrer und externe Beobachtung

Mensch-Roboter Interaktion für PdV

- Weitere Aufnahmekanäle für PdV: multimodale Interaktion
- Interaktion während des Lernvorgangs mit dem Menschen
  - Blickrichtung des Lehrers
  - Zeige-Gesten des Lehrers
  - Sprachliche Informationen: Ergänzende Informationen (vokale Deixis: Hier, Dort, Jetzt,...; Hervorhebung von einzelnen Schritten)
- Roboter fragt bei Unklarheit: Repräsentation mit Konfidenzmaß, Roboter fragt gezielt nach Details
- Mensch weist den Roboter gezielt auf schlecht ausgeführte Aspekte der Aktionen hin

13.0.2 KOGNITION

Lernen einer Fähigkeit (PdV)

- Eine Fähigkeit kann auf zwei Ebenen repräsentiert werden
  - Trajektorie: Eine nicht lineare Zuordnung zwischen Sensor- und Motorinformation stellt eine Repräsentation auf niedriger Ebene dar
    - Verwendet Methoden zur Dimensionalreduktion, die das aufgenommene Signal in einen latenten Bewegungsraum mit niedrigerer Dimension projiziert
    - Diese Methoden setzen lokale lineare Transformationen oder globale nichtlineare Verfahren ein
    - Die Kodierung kann spezifisch zu einer zyklischen (periodischen) Bewegung, zu einer diskreten Bewegung oder zu einer Kombination aus beiden sein
  - Symbolisch: Eine Zerlegung in eine Folge von Aktionen (action perception units) stellt eine Repräsentation auf hoher Ebene dar
    - Ein üblicher Ansatz segmentiert und kodiert die Aufgabe anhand von Folgen vordefinierter Aktionen, die symbolisch beschrieben sind.
    - Das Kodieren und Reproduzieren der Aktionsfolgen kann mit klassischen Machine-Learning-Verfahren wie z. B. HMMs erfolgen.
    - Vorteil: Komplexe Fertigkeiten können effizient über einen interaktiven Prozess gelernt werden
    - Nachteil: Die Verfahren benötigen sehr viel Vorwissen, um die wichtigen Kennzeichen zur Segmentierung erkennen zu können

Roboterprogrammierung durch Vormachen (symbolisch)

- Der Roboter muss die geeignetste Aktion auf Basis des aktuellen Weltzustands bestimmen oder abschätzen
- Ziel: Eine Zuordnung zwischen Weltzustand und Aktion (Strategie zur Aktionswahl)
- Um eine Strategie zu lernen zeigt ein Demonstrator dem Roboter Beispiele für mögliche Aktionen im aktuellen Weltzustand (Demonstrator: Lehrer, Roboter: Schüler)
- Der Weltzustand kann nicht direkt bestimmt werden, sondern nur über die Beobachtungen des Roboters (Sensorik)
- Die gelernte Strategie erlaubt dem Roboter, Aktionen auf Basis seiner Beobachtungne auszuwählen

Vorverarbeitung: Segmentierung von Demonstrationen

- Demonstrationen bestehen meist aus Aktionssequenzen
- Verständnis von Demonstration einfacher, wenn diese in kleinere Segmente oder sogar bekannte Aktionen zerlegt sind

- Aber:
  - Welche/Wie viele Aktionen vorkommen ist unbekannt
  - Start und Ende von Aktionen ist unbekannt
  - Aktionen gehen nahtlos ineinander
- Hauptsächlich zwei Arten von Segmentierung
  - Bewegungssegmentierung
    - Untersuchung der Bewegungstrajektorien nach Mustern und Änderungen in der Bewegungsart
    - Algorithmen: HMM, PCA, Clustering, Template matching, Klassifikation
  - Semantische Segmentierung (Aufgabensegmentierung)
    - Beobachtung des Lehrers und der Umgebung
    - Extraktion von semantischen Merkmalen (Zustände, Objekt-Relationen, Regeln)
- Objektkontaktrelationen zur Repräsentation des Weltzustandes
- Keyframes werden bei Änderungen der Kontaktrelation eingefügt
- Jedes Segment repräsentiert eine Manipulationsaktion (Teig umrühren enthält z.B. greifen, rühren, abstellen, ...)
- Aber: Segmente haben noch keine Label
- Objektrelationen ermöglichen die Extraktion von Vorbedingungen und Effekten von Aktionen für symbolische Planer
  - Zustand am Segmentanfang: LeftHand berührt nichts
  - Zustant am Segmentende: LeftHand beführt RoteTasse
  - Vorbedingung: empty(LeftHand)
  - Nachbedingung: in(RedCup,LeftHand), !empty(LeftHand)

Hierarchische Aufgabensegmentierung unter Berücksichtigung von Bewegung und relevanten Objekten

- Semantische Segmentierung basierend auf Objekt-Kontaktrelationen
- Bewegungssegmentierung basierend auf Charakteristiken von Trajektorien (Bewegungsdynamik)

Aufnahme der Demonstrationen

- Marker basiertes Motion-Capture von Proband, Objekten, Umgebung
- Transformation zu 6D Posen Trajektorien: 3D Modelle mit virtuellen Markern

- Master Motor Map (MMM) Datenformat

Bewegungscharakteristik Heuristik

- Heuristik basierend auf dem Beschleunigungsprofil: Messen der Länge der Beschleunigungskurve
  $s_{l,d}(t_c) = \int_{t_c-\frac{w}{2}}^{t_c-1} \sqrt{1 + a'_d(t)^2 dt(\frac{\hat{U}_l}{\hat{U}_r})^2}$
  $s_{r,d}(t_c) = \int_{t_c}^{t_c+\frac{w}{2}-1} \sqrt{1 + a'_d(t)^2 dt(\frac{\hat{U}_r}{\hat{U}_l})^2}$
- Iterative Suche des besten Keyframe Kandidaten mittels gleitendem Fenster
  - Vergleiche der Segmente links und rechts des Keyframe Kandidaten mit der Auswertfunktion s
  - Keyframe Qualität: $q_d = \frac{s_{l,d}}{s_{r,d}}$ für $s_{l,d} > s_{r,d}$ und $\frac{s_{r,d}}{s_{l,d}}$ für $s_{l,d} \leq s_{r,d}$
- Rekursive Unterteilung bis die Segmentgröße oder Qualität zu klein/niedrig

Evaluation der Segmentierung

- Neue Metrik für die Segmentierungsergebnisse: Bestrafung für fehlende und zusätzliche Keyframes $e = (m + f) * p + \sum_i \min_j (k_{r,i} - k_{r,j})^2$

Demonstrationsverarbeitung
2 Möglichkeiten Akionen zu verarbeiten:

- Batch Lernen: Die Aktion wird gelernt wenn alle Aktionen/Wiederholungen aufgenommen werden
- Inkrementelles Lernen: Die Aktionsrepräsentation wird nach jeder Aktion/Wiederholung gerlernt/aktualisiert

Aktionsrepräsentation auf Trajektorienebene

- Lernen von Aktionen und Repräsentationen geschieht oft zusammen: Repräsentation ist stark an der Lernverfahren gekoppelt
- Populäre Verfahren
  - Hidden Markov Models (HMM)
  - Dynamic Movement Primitives (DMP)
  - Gaussian Mixture Models (GMM)
- Verfeinerung von gelernten Trajektorien: Reinforcement Learning

Dynamic Movement Primitives (DMP)

- Gegeben

  - Demonstration für jeden Zeitpunkt Position y_D und Geschwindigkeit v_D
  - Ausführung: Anfangsposition y_0, Anfangsgeschwindigkeit v_0, Zielposition y_z, Zeitdauer t
- Gesucht: Eine Trajektorie von y_0 bis zu y_z ähnlich zu Demonstration y_D
- Lösung: Feder-Masse-Dämpfer System mit dem Perturbationsterm $\tau \ddot{y} = K(g-y) - D\dot{y} + f_\theta(g-y_0)$ und $\tau \dot{x} = -ax$
- $f\theta(x)$ wird durch die Beobachtung von y_D gelernt.
- Hyperparameter für die Anpassung
  - τ: Zeitlicher Faktor -> Anpassung der Geschwindigkeit
  - g: Ziel -> Anpassung der Zielposition
  - y_0: Skalierungsfaktor -> Anpassung der Anfangsposition
- Kanonisches System: $\tau \dot{x} = -ax$: Das Transformationssystem ist hiermit nicht unmittelbar von der Zeit abhängig

13.0.3 Aktion

Trajektorienausführung

- Generierter Trajektorie $Y = (y_t)_{t=1}^T$ folgen (z.B. PD Regler: $u_t = k_p(y_t - y) - k_d v$)
- Direktes Konstrollsignal (speziell für DMP): $u_t = \ddot{y} = \frac{1}{\tau}(K(g-y) - D\dot{y} + f_\theta(x_t)(g-y_0))$

Online Anpassung bei der Ausführung

- Online Anpassung ist möglich durch den Kopplungsterm in DMP
- Online Anpassung ist schwierig für GMM und HMM, weil beide nicht dynamisch sind
- DMP Online Anpassung $\tau \ddot{y} = K(g-y) - D\dot{y} + f_\theta(x)(g-y_0) + C$

Ausführung von komplexen Aufgaben

- Repräsentationen von komplexen Aufgaben mit Aktionssquenzen
- Ausführung von Aktionssequenzen in dynamischen Umgebungen
  - Aktionen einzeln programmiert oder gelernt
  - Aktionssequenzen hängen vom Weltzustand ab -> Einsatz von symbolischen Planern möglich
- Zustand der Umgebung
  - Selbstlokalisierung (z.B. mit Laserscanner)
  - Objektlokalisierung (z.B. Stereo-RGB, RGBD)

  - Symbolisches Umgebungswissen (Landmarken, Zustand der Umgebung)
  - Abbildung von kontinuierlichem Wissen auf symbolisches Wissen z.B. 6D Objektpose -> auf(Objekt,Tisch)
- Initialer Weltzustand
  - Klassische Planer benötigen vollständigen Weltzustand
  - Problem: Roboter weiß zunächst nicht, wo die Objekte sind
  - Lösung: Hypothesen generieren anhand von Roboterwissen aus
    * Erfahrung gewonnen aus früheren Ausführungen
    * Allgemeines Objektwissen z.B. Data-Mining, Semantische Netzt
- Fehlerbehandlung
  - Erkennen von Problemen
    * Fehlerhafter Weltzustand beim Planen (bei der Ausführung entdeckt)
    * Fehlerhafte Ausführung
  - Lösungen
    * Neu planen auf dem aktuellen Weltzustand
    * Wenn nicht möglich: Suche nach Alternativen
      · Gemeinsame Affordanzen: Soda nicht verfügbar -> Soda ist trinkbar -> Orangensaft
      · Ähnliche Funktion aus visuellen Merkmalen (Form der Objekte)
      · etc.

Zusammenfassung:

- PdV ermöglicht es, Fähigkeiten zu transferieren
- PdV kann den Lernprozess beschleunigen (im Vergleich zu Reinforcement Learning bzw. Versuch-und-Irrtum Methoden)
- PdV liefert Vorschläge für die optimale Lösung für eine Aufgabe zu ermitteln
- Beim Imitationslernen können Positiv- und Negativbeispiele verwendet werden
- Der Roboter muss über eine Bibliothek an generischen Fähigkeiten (skills) verfügen und in der Lage sein diese an den Kontext anzupassen
- Imitation ist
  - Nicht geeignet wenn eine gute Lösung des Lehrers keine gute Lösung des Nachahmers ist
  - Nichr geeignet wenn die Demonstrationen des Lehrers stark verrauscht sind