1 Einführung

1.1 Asimovsche Robotergesetze - ethische Richtlinien

1. Ein Roboter darf keinem Menschen Schaden zufügen, oder ihn durch Untätigkeit zu Schaden kommen lassen.
2. Ein Roboter muss den Befehlen eines Menschen folgen. Es sei denn Widerspruch zu 1
3. Ein Roboter muss seine Existenz schützen. Es sei denn Widerspruch zu 1 und 2

1.2 Definitionen und Historik

- Roboter: Technisches System, das mit der Umwelt auf Basis von Sensoren, Aktoren und Informationsverarbeitung zur Erfüllung von Aufgaben interagiert. Mechanische Vorrichtung, die autonom komplexe Aufgaben verrichtet. Verkörperung des Agentenzyklus (sense, plan, act)
- 1960 erster Industrieroboter „Unimate“
- 1970 erste mobile Platform
- 1980 PUMA-Roboter
- Getriebe: Drehbewegung und Kräfte übertragen und umwandeln, Bindeglied zwischen Gelenk und Motor

1.3 Generationen von Robotersystemen

1. programmierbare Manipulatoren (feste Haltepunkte)
2. adaptive Roboter (roboterorientierte Programmierung, mehr Sensoren)
3. autonome Roboter (aufgabenorientierte Programmierung)
4. humanoide Roboter (Lernfähigkeit)

2 Teilsysteme

Koordinatentransformationen

- Zylinderkoordinaten → Kartesische Koordinaten $(r, \alpha, z) \rightarrow (x, y, z)$

$$x = r\cos(\alpha) \quad y = r\sin(\alpha) \quad z = z \tag{1}$$

- Kartesische Koordinaten → Zylinderkoordinaten $(x, y, z) \rightarrow (r, \alpha, z)$

$$r = \sqrt{x^2 + y^2} \quad \alpha = \arctan\left(\frac{y}{x}\right) \quad z = z \tag{2}$$

- Kugelkoordinaten → Kartesische Koordinaten $(r, \alpha, \beta \rightarrow (x, y, z)$

$$x = r\sin(\alpha)\cos(\beta \quad y = r\sin(\alpha)\sin(\beta \quad z = r\cos(\alpha) \tag{3}$$

- Kartesische Koordinaten → Kugelkoordinaten $(x, y, z) \rightarrow (r, \alpha, \beta$

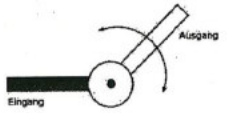
$$r = \sqrt{x^2 + y^2 + z^2} \quad \beta = \arctan\left(\frac{y}{x}\right) \quad \alpha = \arctan\left(\frac{\sqrt{x^2 + y^2}}{z}\right) \tag{4}$$

2.1 Arbeitsraum

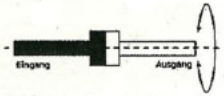
- Punkte im 3D Raum, die von Roboterhand angefahren werden können (3 Bewegungsfreiheitsgrade, also mindestens 3 Gelenke notwendig)
- Grundform des Arbeitsraums: Arbeitsraum ohne Begrenzung der Gelenkwinkel und gegenseitige Behinderung der Arme
- Form meist Quader (LLL z.B. Portalroboter), Hohlzylinder (LVL, RRLT z.B. Scara), Hohlkugel (TRR, VVRTRT z.B. Puma)
- z.B. Parallel Robot:

2.2 Gelenktypen

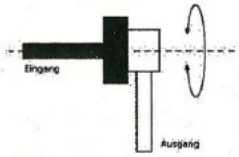
- Rotationsgelenk(R): Drehachse bildet rechten Winkel zu Achsen der Glieder, z.B. Ellbogengelenk



- Torsionsgelenk(T): Drehachse parallel zu Achsen der Glieder z.B. Unterarmdrehung



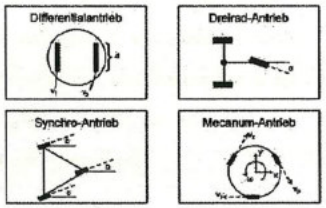
- Revolvergelenk(V): Eingangsglied parallel zu Drehachse, Ausgangsglied im rechten Winkel z.B. Schultergelenk (Arm nach vorne)



- Lineargelenk(L): auch Translationsgelenk, Schubgelenk, prismatisches Gelenk, bewirkt gleitende oder fortschreitende Bewegung entlang der Achse

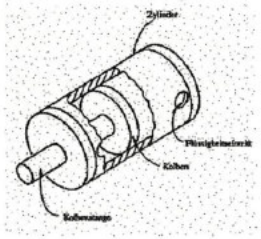
2.3 Antriebe

2.3.1 Radkonfigurationen:



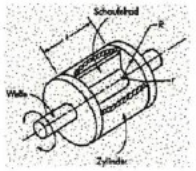
- Differentialantrieb: Kurve, Drehen auf Stelle, einfache Mechanik, Radregelung in Echtzeit
- Dreirad-Antrieb: Kurve, mechanisch einfach, eingeschränkte Manövrierfähigkeit
- Synchro-Antrieb: einfache Regelung, mechanisch komplex
- Mecanum-Antrieb: z.B. ARMAR III, uneingeschränkt in x,y und Gierwinkel beweglich, komplexe Mechanik und Regelung

2.3.2 Fluidischer Antrieb - Linearantrieb



- $f(t)$ Fließgeschwindigkeit des Mediums (Volumen/Zeit)
- A Grundfläche des Kolbens
- $P(t)$ Druck des Mediums
- Geschwindigkeit des Kolbens $v(t) = f(t)/A$
- Kraft des Kolbens $F(t) = P(t)A$
- Hände von ARMAR-III sind fluidisch angetrieben

2.3.3 Fluidischer Antrieb - Schaufelrad



- h Breite des Schaufelrades
- r innerer Radius des Schaufelrades
- R äußerer Radius des Schaufelrades
- Winkelgeschwindigkeit der Welle: $W(t) = 2f(t)/((R^2 - r^2)h)$
- Drehmoment der Welle $T(t) = 0.5P(t)h(R - r)(R + r)$

2.3.4 Muskelartiger Antrieb - Pneumatischer Antrieb

- z.B: am Roboter Airbug
- Stellenergie: Komprimierte Luft bewegt Kolben, kein Getriebe
- Vorteile: billig, einfacher Aufbau, schnelle Reaktionszeit, auch in ungünstigen Umgebungen brauchbar
- Nachteile: laut, geringe Tragkraft, schlechte Regelbarkeit (Luft kompressibel)
- Einsatz: kleinere Roboter mit schnellen Arbeitszyklen und wenig Kraft z.B. Palettierung kleiner Werkstücke

2.3.5 Muskelartiger Antrieb - Hydraulischer Antrieb

- Stellenergie: Öldruckpumpe und steuerbare Ventile
- Vorteile: sehr große Kräfte, mittlere Geschwindigkeit, hohe Energiedichte, hohe Dynamik
- Nachteile: laut, umfangreiche Peripherie (Kompressor, Schläuche), Ölverlust führt zu Verunreinigung (leckage), feuergefährlich, schlechte Reaktionszeiten, Positionier- und Wiederholgenauigkeiten (Ölviskosität), niedriger Wirkungsgrad
- Einsatz: große Roboter mit hohen Traglasten und/oder großem Arbeitsbereich, z.B. Schweißen

2.3.6 Elektrischer Antrieb

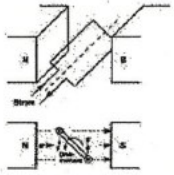
- Stellenergie: Schritt- oder Servomotoren
- Vorteile: kompakt, sehr gute Dynamik, gute Regelbarkeit, großer Leistungs- und Drehzahlbereich
- Nachteile: wenig Kraft, langsam, Umssetzungsgetriebe notwendig, Wärmeerzeugung
- Einsatz: kleinere Roboter für Präzisionsarbeiten z.B. Leiterplattenbestückung, Standard für Roboter aller Leistungsklassen

2.3.7 Elektrohydraulische Antriebe

- Stellenergie: Hydromotor oder Hydraulikzylinder und Servoventil
- Vorteile: lösen hochdynamischer und präziser (μm Bereich) Antriebsaufgaben
- Nachteile: aufwendige Installation und Instandhaltung
- Einsatz: fast nur bei sehr großen Kräften z.B. Flugzeugreinigung

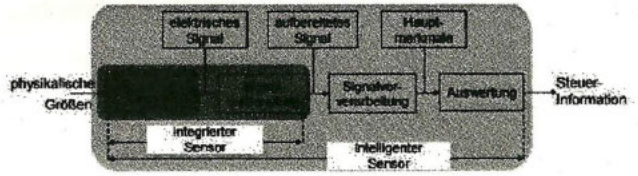
2.3.8 Funktionsweise Elektromotor

- Umwandlung elektrische in mechanische Energie
- stromdurchflossener Leiter wird im Magnetfeld abgelenkt
- Kraft ist proportional zum Strom und zur Magnetfeldstärke
- für Drehbewegung muss Polarität gewechselt werden
- Polaritätswechsel mechanisch (oder Wechselstrom)
- Eisenkerne bündeln das Magnetfeld



2.4 Sensoren

- Umwandlung physikalischer Größen und deren Änderung in geeignete elektronische Signale
- Ziel: Erfassung der Umwelt in nicht fest definierten und sich verändernden Umgebungen



- Elementarsensor: Aufnahme einer Messgröße und Abbildung auf Signal

- Integrierter Sensor: zusätzliche Signalaufbereitung (Verstärkung, Filterung ...)
- Intelligenter Sensor: zusätzliche rechnergesteuerte Auswertung, Ausgang: verarbeitete Größe z.B. Mustererkenner
- Sensortypen: mechanische (z.B. Kraftsensor), Temperatur, chemisch, optisch, akustisch, magnetisch, Gas ...
- Anforderungen: Genauigkeit, Präzision, Kalibrierung, Kosten ...→ Sensor wird nach Aufgabenstellung und Ort der Integration ausgewählt
- Probleme: Signalverarbeitung, Wahl der Sensorik (liefert nur partielle Informationen), Fusion der Messwerte in Multisensorsystemen, Abstraktionsstufen des Umweltmodells
- Interne Sensoren:
  - Kein Kontakt zur Umwelt
  - Bestimmung von Lage und Position durch Erfassung des inneren Zustands (Neigung, Orientierung, Beschleunigung)
  - z.B. Geschwindigkeitssensoren, Positionssensoren, Strom, Spannung, Temperatur
  - Aufgaben: Stellung, Geschwindigkeit, Kräfte und Momente der Gelenke
- Externe Sensoren:
  - Information aus Umwelt
  - Bestimmung von Lage und Position in Bezug auf Umwelt, Beschaffenheit der Umwelt, Kommandos
  - z.B. Taktile Sensoren, Abstandssensoren
  - Aufgaben: Entfernungen, Lage und Kontur von Objekten, Pixelbilder der Umwelt (CCD-Kamera)
- Aktive Sensoren: Simulation der Umwelt durch Eintrag von Energie, Messen und Auswertung z.B. Ultraschall, Laser-Entfernungsmesser
- Passive Sensoren: im Umwelt vorhandene Signale werden gemessen und ausgewertet z.B. Tastsensoren, Kameras, Mikrophone

3 Mathematische Grundlagen

- Basiskoordinatensystem (BKS): 3 dimensionales Koordinatensystem, durch orthogonale Einheitsvektoren $\vec{e}_{X,B}, \vec{e}_{Y,B}, \vec{e}_{Z,B}$ definiert (rechtsdrehend oder linksdrehend)
- Lokale Koordinatensysteme: z.B. Objektkoordinatensystem (OKS), Effektorkoordinatensystem (EKS = TCP), Sensorkoordinatensystem (SKS)
- Ort: Ortsvektor $\vec{u}$ von Ursprung BKS zu Ursprung OKS

- Orientierung: Rotationsmatrix zur Abbildung der Einheitsvektoren des OKS auf die Einheitsvektoren des BKS
- Lage:
  - Ortsvektor und Rotationsmatrix des OKS bezogen auf das BKS
  - kann durch 6-Tupel $\vec{v} = (x, y, z, \alpha, \beta\gamma)$ beschrieben werden
  - x,y,z: Koordinaten des Ursprungs des OKS bezogen auf Ursprung des BKS
  - $\alpha, \beta\gamma$: Drehwinkel bezogen auf Drehachsen

3.1 Bewegungsfreiheitsgrad

- Englisch: degrees of freedom (DOF)
- Freiheitsgrad f eines Objektes: Anzahl möglicher unabhängiger Bewegungen in Bezug auf BKS, in 3D $f = 6$ (3 Rotationen, 3 Translationen)
- Freiheitsgrad F eines Roboters mit $F_R \leq 3$ für Rotationsgelenke und $F_T = 1$ für Translationsgelenke, bei n Gelenken (meist $n \geq 6$): $F = \sum_i^n (F_{R_i} + F_{T_i})$
- $F \geq f \Rightarrow$ um $f = 6$ für Effektor zu erreichen sind mindestens $F = 6$ Bewegungsachsen notwendig

3.2 Rotationen und Translationen

3.2.1 3 × 3 Matrizen

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \quad R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

- Drehachse = normierter Eigenvektor von R zum Eigenwert 1
- Rotationsmatrix um Einheitsvektor $\vec{v}$:

$$R_{\vec{v},\alpha} = \begin{pmatrix} \cos(\alpha) + v_1^2(1-\cos(\alpha)) & v_1v_2(1-\cos(\alpha)) - v_3\sin(\alpha) & v_1v_3(1-\cos(\alpha)) + v_2\sin(\alpha) \\ v_2v_1(1-\cos(\alpha)) + v_3\sin(\alpha) & \cos(\alpha) + v_2^2(1-\cos(\alpha)) & v_2v_3(1-\cos(\alpha)) - v_1\sin(\alpha) \\ v_3v_1(1-\cos(\alpha)) - v_2\sin(\alpha) & v_3v_2(1-\cos(\alpha)) + v_1\sin(\alpha) & \cos(\alpha) + v_3^2(1-\cos(\alpha)) \end{pmatrix}$$

- Spur davon eignet sich um Rotationswinkel zu bekannter Rotationsachse zu bestimmen $Sp(R) = 3\cos(\alpha) + (v_1^2 + v_2^2 + v_3^2)(1 - \cos(\alpha))$
- in 2D:

$$R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{6}$$

Euler Winkel:

- Drehung um jeweils veränderte Achsen

- oft: α um z-Achse, βum y'-Achse und γ um z"-Achse $R_{z}(\alpha)R_{y'}(\beta R_{z''}(\gamma)$
- Nachmultiplikation (von links nach rechts): Drehung um momentanes Koordinatensystem (definiert durch linksstehendes Matrixprodukt, für das erste BKS)

Roll, Pitch, Yaw:

- Drehung um unveränderte Achsen
- oft: α um x-Achse, βum y-Achse und γ um z-Achse $R_{z}(\gamma)R_{y}(\beta R_{x}(\alpha)$
- Vormultiplikation (von rechts nach links): Drehung um Ursprungskoordinatensystem (= BKS)

3.2.2 Homogene Transformationen

- Zusammenfassung von Rotation und Translation in homogene 4×4 Matrizen (3×3 Matrizen in 2D)
- Homogene Koordinaten (x, y, z, w) ($n+1$-dim Raum), in Robotik $w = 1$
- enthält: Rotation, Translation, Skalierung und perspektivische Transformationen (alle Basisoperationen)

$$T = \left(\begin{array}{c|c} \mathbf{R_{3\times3}} & \mathbf{p_{3\times1}} \\ \hline \mathbf{f_{1\times3}} & \mathbf{1\times1} \end{array}\right)$$

- f = Perspektivische Transformation, hier nicht relevant, also $f = (0, 0, 0)$
- Basisrotationsmatrizen: oben links Rotationsmatrix, rechts unten Eins und sonst Null
- Wert rechts unten gibt globale Skalierung
- Abbildung Ortsvektor OKS ins BKS $p_{BKS} = Tp_{OKS}$ mit $T = \begin{pmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{u} \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Invertierung: ($R^{-1} = R^T$)

$$T^{-1} = \begin{pmatrix} n_x & n_y & n_z & -n^T\vec{u} \\ o_x & o_y & o_z & -o^T\vec{u} \\ a_x & a_y & a_z & -a^T\vec{u} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

- Verkettete Lagebeschreibung: ${}^AH_B$ Lage des Koordinatensystems B relativ zu A, es gilt: ${}^AH_B = {}^AH_C \cdot {}^CH_B$ (Matrix bezieht sich auf Stellung, die durch Matrix links gegeben ist)
- Nachteile homogene Darstellung: Hohe Redundanz (Orthogonalität), 12 nichttriviale Kenngrößen, obwohl nur 6 notwendig , Interpolation schwierig $\Rightarrow$ besser Quaternionen

3.2.3 Quaternionen

- Quaternion = hyperkomplexe Zahl $q = a + b \cdot i + c \cdot j + d \cdot k = (a, b, c, d)^T = (a, \vec{u})^T$ mit Realteil a und Imaginärteil $\vec{u} = (b, c, d)^T$

$$i^2 = j^2 = k^2 = ijk = -1; \quad ij = -ji = k; \quad jk = -kj = i; \quad ki = -ik = j; \quad (8)$$

- Rechenregeln $q = (a_q, \vec{u}_q), r = (a_r, \vec{u}_r)$
- Addition: $q + r = (a_q + a_r, \vec{u}_q + \vec{u}_r)$
- Punktprodukt (Skalarprodukt): $q \cdot r = a_q a_r + b_q b_r + c_q c_r + d_q d_r$
- Quaternionen-Multiplikation: $qr = (a_q + ib_q + jc_q + kd_q)(a_r + ib_r + jc_r + kd_r)$
- Konjungierter Quaternion: $q^* = (a_q, -\vec{u}_r)$
- Norm: $|q| = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$
- Multiplikatives inverses Element: $q^{-1} = q^*/|q|^2$
- Vektor $\vec{p}$ als Quaternion $q = (0, \vec{p})^T$
- Skalar s: $q = (s, \vec{0})^T$
- Einheitsquaternion: $|q| = 1$
- Rotation um θ mit Rotationsachse $\vec{u}$: $q = (\cos(\theta/2), \vec{u}\sin(\theta/2))$
- Punkt v wird rotiert durch $v' = qvq^+ = qvq^{-1}$
- Hintereinanderschalten: $f(v) = qvq^{-1}, h(v) = rvr^{-1} \Rightarrow f \circ h$ entspricht Rotation mit $p = qr$
- Vorteile: intuitiv, kompakt, numerisch stabil
- Nachteil: nur Rotation, keine Translation

3.2.4 Duale Quaternionen

- 4 reellen Werte werden durch Dualzahlen ersetzt $\Rightarrow$ Orientierung und Lage beschreibbar
- Duale Zahl: $d = p + \varepsilon s$ mit Primärteil p, Sekundärteil s und $\varepsilon^2 = 0$
- Addition: $d_1 + d_2 = p_1 + p_2 + \varepsilon(s_1 + s_2)$
- Multiplikation: $d_1 \cdot d_2 = p_1 \cdot p_2 + \varepsilon \cdot (p_1 \cdot s_2 + p_2 \cdot s_1)$
- Bei einer dualen Zahl beschreibt p die Drehung (Winkelwert $\theta/2$) und s die Verschiebungsgröße d, die restlichen drei Dualzahlen beschreiben eine beliebige gerichtete, normierte Gerade im Raum, bzgl. der die Rotation und Translation erfolgen
- Vorteile: zur Stellungsbeschreibung geeignet, alle benötigten Transformationen erlaubt, geringe Redundanz (8 Kenndaten), geringe Anzahl an Einzeloperationen der Rechenoperationen
- Nachteile: Stellungsbeschreibung schwierig, komplexe Verarbeitungsvorschriften

4 Modelltypen

4.1 Geometrische Modellierung

- Geometrie: mathematische Beschreibung der Form von Körpern
- Einsatzbereiche:
  - Graphische Darstellung von Körpern
  - Ausgangspunkt der Abstandsmessung und Kollisionserkennung
  - Grundlage zur Berechnung der Bewegungen von Körpern
  - Grundlage zur Ermittlung der wirkenden Kräfte und Momente
- 2D, 2,5D (dritte Dimension nur Attribut der 2D Objekte z.B. Geländehöhe als Funktion der Lage) oder 3D Modelle
- Kanten- bzw. Drahtmodelle: Körper werden durch Polygonzüge (Kanten) dargestellt
  - Vorteile: nützlich zur schnellen Visualisierung, wenige einfache Daten
  - Nachteile: Mehrdeutigkeiten, hoher Eingabeaufwand, keine Kollisionsberechnung, kein Schnitt
- Flächen- bzw. Oberflächenmodelle z.B. Blockwelt (Körper werden durch einhüllende Quader dargestellt, 1.Schritt bei Kollisionsvermeidung), Speicherung von Kanten und Oberflächen (Dreiecke, Bezier)
  - Vorteil: effiziente Verfahren, entspricht dem Vorgehen während der Modellierung, schnelle Kollisions- und Abstandsberechnung
  - Nachteile: hoher Eingabeaufwand, aufwendige Darstellung, Problem bei Schnittoperationen, Inkosistenzen möglich
- Volumenmodell, Körper werden genau (in Animation) dargestellt, für genaue Kollisionserkennung nötig

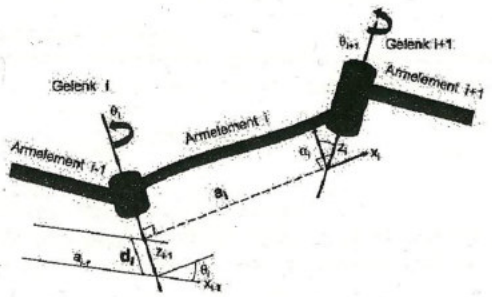
4.2 Kinematische Modellierung

- Kinematik: Lehre der geometrischen und analytischen Beschreibung der Bewegungszustände mechanischer Systeme
- Kinematische Kette:
  - mehrere Körper, die durch Gelenke kinematisch verbunden sind z.B. Roboterarm
  - offen oder geschlossen
  - Elemente: Roboter Basis, Armelemente (Glieder), Gelenke, Endeffektor
  - Armelement: starrer Körper, mit nächstem durch Schub- oder Rotationsgelenk (nur 1 Freiheitsgrad, rotatorisch oder translatorisch) verbunden
  - Kinematische Parameter: Gelenk- und Armelementparameter

- beschrieben durch Lage jedes einzelnen Armelementes bezogen auf ein Referenzsystem
- jedes Armelement hat OKS, Ursprung im Armelement, das entsprechendes Gelenk bewegt, Transformationsmatrix zwischen jedem OKS und BKS notwendig (6 Parameter pro Armelement, 3 rot., 3 trans.)

4.2.1 Denavit-Hartenberg-Konvention

- Systematische Beschreibung der Beziehungen zwischen benachbarten Gelenken ⇒ Reduktion von 6 auf 4 Parametern
- DH-Parameter: Armelementlänge a, Verwindung α, Gelenkabstand d (variabel bei Schubgelenk), Gelenkwinkel θ (variabel bei Rotationsgelenk)



- z_{i-1}-Achse entlang Bewegungsachse des i-ten Armelementes $i \in Basis, 1, \ldots, n$
- Kürzester Abstand (Normale) zwischen Gelenkachsen = Armelementlänge a_i
- x_i-Achse: Verlängerung a_i nach Gelenk $i+1$
- Fußpunkt von a_i mit Achse g_{i+1} ist Ursprung von i-tem Koordinatensystem
- Armelementverwindung α_i: Winkel zwischen z_{i-1}-Achse (in Richtung g_i) und z_i Achse
- Gelenkabstand d_i: Translation entlang z_{i-1} Achse um x_{i-1} mit x_i Achse zu schneiden
- θ_i: Winkel um den Gelenk i gedreht werden muss, damit x_{i-1} und x_i parallel sind
- y_i-Achse durch Rechte-Hand-Regel ergänzen
- Transformation OKS_{i-1} zu OKS_i

$$A_{i-1,i} = R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) =$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos\alpha_i & \sin\theta_i \cdot \sin\alpha_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & -\cos\theta_i \cdot \sin\alpha_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Transformation OKS_i zu OKS_{i-1}: $A_{i,i-1} = A_{i-1,i}^{-1}$
- Verkettung ergibt Lage des Endeffektors bezüglich BKS (Translationsvektor von Gesamtmatrix), der Translationsvektor dieser homogenen Matrix ist das f für die inverse Kinematik

4.3 Dynamische Modellierung

- Dynamik: Untersuchung der Bewegung von Körpern als Folge der auf sie wirkenden Kräfte und Momente

5 Vorwärts-Kinematik

- Kinematisches Problem: Von Gelenkwinkeln (Roboterkoordinaten, Konfigurationsraum) in kartesische Koordinaten z.B. Lage des Endeffektors (TCP) lokalisieren
- zur Erreichbarkeitsanalyse, (Selbst)kollisionserkennung
- Manipulator skizzieren, beschriften, $a_i, \alpha_i, d_i, \theta_i$ bestimmen, Transformationsmatrizen bestimmen und verknüpfen
- Stellung TCP bzgl. BKS:

$$S_{Basis,Greifer}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \ldots \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$$

Gelenkwinkel sind vorgegeben ⇒ Gleichung lösen

6 Repräsentation der Erreichbarkeit

- Erreichbarkeit: Es existiert Gelenkwinkelkonfiguration, sodass TCP an diesem Ort ist
- Arbeitsraum durch 6D Voxelgitter approximieren ⇒ in Simulation Gelenkwinkel abtasten ⇒ Lage des TCP über Vorwärtskinematik ⇒ setze entsprechenden Voxel auf erreichbar ⇒ in Binädatei speichern
- zusätzlich können Qualitätsinformationen gespeichert werden z.B. Manipulierbarkeit
- kann zur Griffselektion genutzt werden

7 Inverse Kinematik

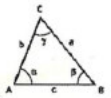
- Inverses kinematisches Problem: Von kartesischen Koordinaten in Gelenkwinkelraum z.B. Positionierung des TCP

- Lage des TCP:

$$T_{TCP} = \begin{pmatrix} N_x & O_x & A_x & P_x \\ N_y & O_y & A_y & P_y \\ N_z & O_z & A_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

$$T_{TCP} = S_{Basis/Greifer}(\theta) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \ldots \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$$

- Gleichung nach θ auflösen $\Rightarrow$ Nichtlineares Problem $\Rightarrow$ kein allgemeines Lösungsverfahren
- Unerreichbare Stellung: Außerhalb des Armradius des Roboters
- Unzulässige Stellung: Prinzipiell erreichbar, durch Winkelbeschränkungen oder (Selbst)kollisionen jedoch nicht einnehmbar $\Rightarrow$ sollten bei Trajektorienplanung vermieden werden
- Eindeutigkeit:
  - In der Ebene gibt es für Roboter mit $n \geq 3$ Bewegungsfreiheitsgrade mehrere Möglichkeiten eine Effektorstellung zu erreichen (bei 3D für $n \geq 6$)
  - Reduktionsstellungen lassen sich auch mit $n < 6$ Bewegungsfreiheitsgraden erreichen
- geometrische Lösung: Nutze geometrische Beziehungen (Trigonometrische Funktionen, Sinus-/Kosinussätze, ...) um die Gelenkwinkel aus T_{TCP} zu bestimmen, schnell, aber nur für bestimmten Robotertyp möglich
- Kosinussatz: $c^2 = a^2 + b^2 - 2ab\cos(\gamma)$



- Polynomialisierung: Transzendente Gleichungen in Polynomgleichungen umwandeln durch Substitution $u = \tan\theta/2$

$$\cos\theta = \frac{1-u^2}{1+u^2} \quad \sin\theta = \frac{2u}{1+u^2} \tag{10}$$

- analytische Lösung: Durch Koeffizientenvergleich der beiden homogenen Matrizen, 12 nichttriviale Gleichungen, Kenntnisse der Transformation ausnutzen

7.1 numerisch

- allgemeines Verfahren, aber hoher Aufwand und lange Zeitdauer
- Berechne in jeder Iteration die Abweichungen Δx und $\Delta\theta$ zur Sollposition
- Vorwärtskinematik als Funktion: $x(t) = f(\theta(t))$ ($x(t)$ = Beschreibungsvektor der TCP Lage)

- Ableitung nach der Zeit: $\dot{x}(t) = J(\theta)\dot{\theta}(t)$, $\dot{x}$ = Geschwindigkeit im kartesischen Raum, $\dot{\theta}$ = Gelenkgeschwindigkeit, J = Jacobi Matrix = äußere Ableitung von f $J_{ij} = \frac{\partial f_i}{\partial \theta_j}$

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial \theta_1} & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{pmatrix} \tag{11}$$

- Übergang zum Differenzenquotienten (Approximation, Linearisierung): $\Delta x \approx J(\theta)\Delta\theta$
- Umkehrung: $\Delta\theta \approx J^{-1}(\theta)\Delta x$
- Falls Inverse nicht existiert (J ist singulär), Pseudoinverse $J^{\pm} = (J^T J)^{-1} J^T$, Inverse existiert wenn Rang$(J) = m$, Singularitäten sollten vermieden werden
- Inverse einer 2×2 Matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{12}$$

- Levenberg-Marquardt Minimierung: Lösung mithilfe der Methode der kleinsten Quadrate, hilft Singularitätenproblem zu lösen

8 Grundlegende Dynamikgleichung

- dynamisches Modell: Zusammenhang von Kräften, Momenten und Bewegungen (Lage, Geschwindigkeit, Beschleunigung), welche in einem mechanischen Mehrkörpersystem auftreten
- Zweck: Analyse der Dynamik, Synthese mechanischer Strukturen, Modellierung elastischer Strukturen, Reglerentwurf
- Anwendungen: Systemauslegung (reine Lageregelung), Trajektorienoptimierung (Bahnregelung) und modellbasierte Regelung (regelt Kräfte aus, die während Handhabung auftreten) (Modellierung in Phasen getrennt)
- Probleme: Zeitaufwand, Fehler und Inkosistenzen wahrscheinlich, Wiederverwendbarkeit schwierig
- Generalisierte Koordinaten: reduzierter Satz unabhängiger Koordinaten, der den Systemszustand vollständig beschreibt und Zwangsbedingungen berücksichtigt
- Bewegungsgleichung:

$$Q = M(q) \cdot \ddot{q} + n(\dot{q}, q) + g(q) + R \cdot \dot{q} \tag{13}$$

  - Q: $n \times 1$ Vektor der Stellkräfte und -momente (Generalisierte Kräfte)
  - M: $n \times n$ Massenträgheitsmatrix
  - n: $n \times 1$ Vektor mit Zentrifugal- und Corioliskomponenten
  - g: $n \times 1$ Vektor mit Gravitationskomponenten

- R: $n \times n$ Diagonalmatrix zur Beschreibung der Reibungskräfte
- q: $n \times 1$ Winkellagen des Manipulators

- Direktes dynamisches Problem: Gegeben äußere Kräfte und Momente und Anfangszustand $Q(t), q(t_0), \dot{q}(t_0), \ddot{q}(t_0)$, gesucht Bewegungsänderung $q(t), \dot{q}(t), \ddot{q}(t)$
- Inverses dynamisches Problem: Gegeben gewünschte Bewegungsparameter $q(t), \dot{q}(t), \ddot{q}(t)$, gesucht erforderliche Stellkräfte und -momente $Q(t)$

8.1 Bewegungsgleichungen nach Lagrange

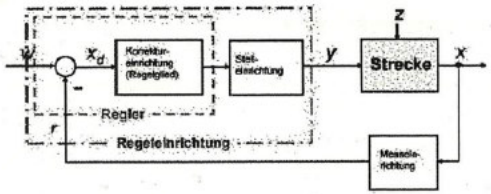
- analytische Methode, Arbeits- oder Energiebetrachtungen
- Ziel: ermittle für jedes Robotergelenk eine Bewegungsgleichung
- Bewegungsgleichung: $Q_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i}$
- mit Lagrange-Funktion: $L = E_{kin} - E_{pot}$ und q_k generalisiert Koordinaten (Zwangsbedingungen berücksichtigt)
- Rotationsenergie $= \frac{1}{2} J \omega^2$
- Trägheitsmoment eines dünnen Stabes: $J = \frac{1}{12} m l^2$
- Additionstheoreme: $\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y)$, $\cos(x \pm y) = \cos(x)\cos(y) \mp \sin(x)\sin(y)$
- Am Ende beide Bewegungsgleichungen in Matrizen schreiben und die generalisierten Koordinaten und ihre Ableitungen als Vektor hinter Matrix
- Vorteile: Einfaches Aufstellen der Gleichungen, geschlossenes Modell, analytisch auswertbar
- Nachteile: Berechnung umfangreich $O(n^3)$ (n = Anzahl der Gelenke), nur Antriebsmomente werden berechnet, keine Modellierung der Reibung

9 Regelung

9.1 Funktionsweise

- Lehre von der selbsttätigen, gezielten Beeinflussung dynamischer Prozesse während des Prozessablaufs
- Systemkenntnis ist unvollständig (Einwirkung von Störungen)
- allgemeingültige systemunabhängige Methoden
- Aufgabe: Der Ausgangsgröße eines dynamischen Systems soll mithilfe einer Stellgröße ein Sollverhalten, gegen den Einfluss einer unvollständig bekannten Stellgröße, aufgeprägt werden
- Lösungsprinzip: Strecke (dynamischer Prozess) laufend beobachten $\Rightarrow$ Stellgröße anpassen, damit trotz Störung Sollverlauf entsteht (Regelung)

- Blockdiagramm:



- w Führungsgröße z.B. Sollkurs, Gelenkwinkelvorgabe
- x_d Regeldifferenz
- Regler = Gehirn beim Menschen, PID-Regler, P,I,D,Kaskaden, Kennlinien, ...
- Stelleinrichtung z.B. Hand/Lenkrad beim Autofahren, Motor (an Pfeil vorher, welche Größe vorgegeben wird von Regler) ...
- Strecke z.B. Lenkrad/Auto beim Autofahren, Roboterarm, ...
- y Stellgröße z.B. Drehmomentvorgabe an den Motor
- z Störgröße
- x Regelgröße z.B. Kurs, Gelenkwinkel, ...
- r Rückführgröße z.B. Istkurs, gemessener Winkel, ...

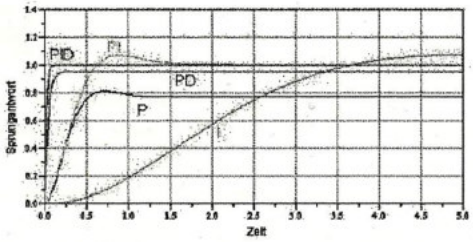
Laplacetransformation:

- Rechenvereinfachung: Differential- und Integralausdrücke werden zu algebraischen Ausdrücken
- Zeitbereich t $\rightarrow$ Frequenzbereich s
- Integral über Zeit $\rightarrow \frac{1}{s}$
- pro zeitliche Ableitung $\cdot s$
- Sprungfunktion: $\frac{1}{s}$ im Frequenzbereich

PID-Regler:

- verbreitet, für nahezu alle Prozesstypen geeignet, robust, mit geringem Aufwand realisierbar
- Proportionalglied (günstiges Regelverhalten) - Integrierglied (stationäre Genauigkeit) - Differenzierglied (schnelle Ausregelung)
- charakteristische Gleichung: $u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t)$
- Laplacetransformierte: $u(s) = K_P e(s) + K_I \frac{1}{s} e(s) + K_D s e(s)$

- Übertragungsfunktion Regler = Ausgang/Eingang = $\frac{u(s)}{e(s)} = G_r(s) = K_P + K_I\frac{1}{s} + K_D s$
- Übertragungsfunktion Strecke: Lagrangebewegungsgleichung berechnen, $Q = T =$ Eingang, Laplace Transformation, G_s =Ausgang/T
- Gesamtübertragungsfunktion: $G = \frac{G_s \cdot G_r}{1 G_s \cdot G_r}$
- Ausgangsfunktion in Zeitbereich zurück transformieren, falls Regler dauerhaft oszilliert ist er falsch getuned
- Sprungantwort:



9.2 Kraft-Positionsregelung

- für exakte Systemmodellierung benötigt man exakte Kenntnisse des Dynamikmodells und der Umgebung des Roboters
- Kraft-Positionsregelung: zur Ausführung von Aufgaben, die Interaktionskräfte berücksichtigen müssen
- Positionen und Kräfte sind eng verbunden, jede Positionsänderung bedeutet Kraftänderung und umgekehrt
- Hybride Kraft-/Positionsregelung:
  - für jede kartesische Bewegungsrichtung wird zwischen reiner Kraft- und reiner Positionsregelung gewählt
  - in Praxis muss Reibung berücksichtigt werden
  - Programmierung ist: schnell, intuitiv, robust, einfach zu warten und wiederverwendbar
  - z.B. Einschieben einer Box
  - reicht z.B: bei Spiralsuche nicht aus ⇒ Kraft- und Positionsregelung auf jeder Achse notwendig

9.3 Impedanzregelung

- regelt dynamische Beziehung zwischen Kraft und Position im Kontaktfall
- Interaktion zwischen Roboter und Umwelt verhält sich wie Feder-Dämpfer-Masse-System
- Zusammenhang Kraft und Bewegung (Feder-Dämpfer-Masse-Gleichung):

$$f(t) = d \cdot x(t) + b \cdot \dot{x}(t) + m \cdot \ddot{x}(t) \tag{14}$$

- Steifigkeit d, Dämpfung b und Trägheit m
- Über Laplace-Transformation folgt Impedanz

$$F(s) = (d + b \cdot s + m \cdot s^2) \cdot X(s) \tag{15}$$

10 Bahnsteuerung

- Trajektorie: Bewegungen eines Roboters sind Zustandsänderungen über der Zeit relativ zu einem stationären Koordinatensystem mit Einschränkungen durch Zwangsbedingungen, Gütekriterien, Neben- und Randbedingungen
- Problem: Gegeben Start- (aus Sensoren und direkter Kinematik) und Zielzustand, gesucht Zwischenzustände (Trajektorie soll glatt und stetig sein)
- Bahnsteuerung im Gelenkwinkelraum (PTP)
  - n-dimensionaler Konfigurationsraum
  - näher an Ansteuerung des Roboters, berücksichtigt Gelenkwinkelgrenzen
  - Bahn ist im kartesischen Raum nicht zwingend definiert (umständliche Formulierung der Trajektorie)
  - asynchrones Abfahren der Trajektorie: Steuerung der Achsen unabhängig voneinander z.B. beim Punktschweißen
  - synchrones Abfahren: achsinterpolierte Steuerung (gibt Leitachse, Bewegung aller Achsen beginnt und endet gleichzeitig) z.B. Bahnschweißen
  - Interpolation der Gelenkwinkel (schwierig, da mehrere) nach vorheriger inverser Kinematik
- Bahnsteuerung im kartesischen Raum (continous path CP)
  - 3D/6D Arbeitsraum
  - näher an lösender Aufgabe (Bahn einfach zu formulieren), aber inverse Kinematik nötig
  - Zustand des Roboters = Beschreibungsvektor des TCP
  - Endeffektor folgt definierter Bahn
  - Gegeben: Anfangs- und Zielpunkt (Orientierung + Position), maximale Geschwindigkeit und Beschleunigung und Bahntyp

  - Interpolation in Weltkoordinaten (aus Zwischenpunkten, einfach) und dann inverse Kinematik
  - Gelenkwinkelgrenzen nicht berücksichtigt (nicht jede Trajektorie wirklich ausführbar)
  - Interpolation: Bahn verläuft durch alle Stützpunkte der Trajektorie
  - Interpolationsarten: linear (Verschleiß durch ruckartige Bewegungen), zirkular (gut, da die meisten Werkstücke aus Geraden und Kurven bestehen) oder segmentweise z.B. mit Splines (Bedingungen der Teiltrajektorien werden aneinander angepasst)

- Programmierung
  - Teach-In:
    * manuelles Anfahren markanter Punkte der Bahn
    * speichern der Gelenkwerte und ergänzen von Geschwindigkeit und Beschleunigung
    * Teach Box/Teach Panel (Bewegung der Gelenke, Speichern/Löschen von Anfahrpunkten, Eingabe von Geschwindigkeiten und Befehlen zur Bedienung des Greifers)
    * Maus/Joystick (nur 2 Freiheitsgrade, schlechte Positioniergenauigkeit)
    * Spacemouse/Teach Kugel (6 Freiheitsgrade, hohe Positioniergenauigkeit)
    * Einsatz z.B. in der Fertigungsindustrie
  - Playback:
    * Bediener bewegt Roboter (im Zero-Force-Control Modus) entlang Bahn, Gelenkwerte werden (automatisch oder manuell) abgespeichert
    * Vorteile: keine mathematische Beschreibung notwendig, handwerkliche Erfahrung kann integriert werden, schnell, intuitiv
    * Nachteile: schwere Roboter schwierig zu bewegen, kein Platz in engen Fertigungszellen (Sicherheitsrisiko), suboptimale Bahn, schlechte Korrekturmöglichkeiten, hoher Speicherbedarf (bei hoher Abtastrate)
    * z.B. Lackieren, Kleben

10.1 Punkt zu Punkt Steuerung (PTP)

- Vorteile: einfache Berechnung der Gelenkwinkeltrajektorie, keine Singularitäten
- mit Rampenprofil: einfach, ruckartige Aufschaltung der Beschleunigung (kann zu Eigenschwingungen führen)
- mit Sinoidenprofil: weichere Bewegung durch sinusförmige Zeitfunktion, Beschleunigungs- und Bremsphase länger, aber Roboter wird weniger beansprucht
- synchrone PTP: Leitachse bestimmen (Achse mit maximaler Fahrzeit bei maximaler Beschleunigung), Bewegungen der anderen Achsen so anpassen, dass sie selbe Zeit benötigen (Bahn gleichmäßiger)
- asynchrone PTP: jedes Gelenk wird mit maximaler Beschleunigung angesteuert, Gelenkbewegungen enden unabhängig voneinander

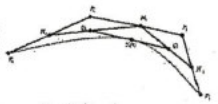
- vollsynchrone PTP: Beschleunigungsphasen werden an Leitachse angepasst, alle Achsen haben selbe Gesamtzeit und selbe Beschleunigungszeit wie langsamste Achse, Vorteil: bessere Annäherung der Start- und Zielpunkte im kartesischen Raum, Nachteil: Beschleunigung jeder Achse wird vorgegeben

10.2 Approximation mit Bezier-Kurven

- Bahnapproximation: Kontrollpunkte beeinflussen Bahnverlauf und werden approximiert (kaum Ruck, hohe Geschwindigkeit möglich)
- Geschwindigkeitsüberschleifen: wenn Geschwindigkeit Minimalwert unterschreitet wird begonnen die Parameter der alten Teiltrajektorie in die Parameter der neuen zu überführen (Stützpunkt wird in der Regel nicht erreicht), Nachteil: Abhängig von Geschwindigkeitsprofil
- Positionsüberschleifen: um Stützpunkt wird Überschleifkugel gezogen, man beginnt mit überschleifen wenn TCP in Kugel eintritt, außerhalb der Kugel wird die Bahn exakt eingehalten, Vorteil: Gut kontrollierbar
- Bezierkurven bilden weiche Verfahrbahnen aus Bernsteinpolynomen
- Basisfunktion:

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i \quad 0 \le t \le 1 \quad \text{mit} \quad B_{i,n}(t) = \binom{n}{i} t^i (a-n)^{n-i} \qquad (16)$$

- De-Casteljau-Algorithmus: Bezierkurven werden durch Polygonzug angenähert, iterativ, immer im Teilungsverhältnis teilen und verbinden, z.B. für Teilungsverhältnis $\frac{1}{2}$ und 2 Iterationen



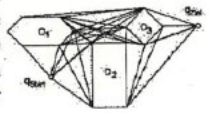
11 Bahnplanung

- unterschieden nach Zustandsräumen: Gelenkwinkelraum, euklidischer Raum, Sensorzustandsraum, Objektzustandsraum
- unterschieden nach Art des Roboters: mobile Roboter, Laufmaschinen und anthropomorphe Systeme, Manipulatoren, Greif- und Montageplanung
- Problemklasse a) vollständiges Umweltmodell, sowie vollständige Neben-Rand- und Zwangsbedingungen bekannt, Kollisionsfreie Bahn vom Start- zum Zielzustand gesucht
- Problemklasse b) unvollständiges Umweltmodell, sowie unvollständige Neben-Rand- und Zwangsbedingungen bekannt (unbekannte Objekte), Kollisionsfreie Bahn vom Start- zum Zielzustand gesucht
- Problemklasse c) zeitvariantes Umweltmodell bekannt (Hindernisse in Ort und Zeit variant), Kollisionsfreie Bahn vom Start- zum Zielzustand gesucht

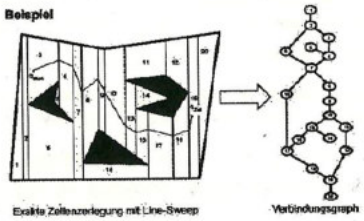
- Problemklasse d) kein Umweltmodell bekannt, Kollisionsfreie Bahn vom Start- zum Zielzustand gesucht (Kartografieren notwendig!)
- Problemklasse e) zeitvariantes Umweltmodell bekannt, Bahn zu einem beweglichen Ziel gesucht (Rendezvous-Problem)
- Konfiguration q: beschreibt Zustand eines Roboters A im euklidischen Raum durch seine Lage und Orientierung bzw. im Gelenkwinkelraum durch die Werte der Gelenke
- Konfigurationsraum C: Raum aller möglicher Konfigurationen von A
- Weg für A von der Konfiguration q_{Start} nach q_{Ziel}: stetige Abbildung $\tau : [0,1] \rightarrow C$ mit $\tau(0) = q_{start}$ und $\tau(1) = q_{Ziel}$
- Arbeitsraumhindernis H: Raum, welcher von einem Objekt im Arbeitsraum eingenommen wird
- Konfigurationsraumhindernis C_H: Menge aller Punkte des C-Raumes, welche zu einer Kollision mit H führen
- Hindernisraum: Menge aller Konfigurationsraumhindernisse $C_{obst} = \cup C_{Hi}$
- Freiraum: Menge aller Punkte aus C, welche nicht im Hindernisraum liegen $C_{free} = C\ C_{obst}$, Berechnung des Freiraums sehr aufwendig ⇒ vereinfachte Repräsentation des Freiraumes durch approximative Verfahren

11.1 Planungsverfahren für mobile Roboter

- Umweltmodellierung kann exakt (algebraisch beschrieben) oder approximiert (durch Polyeder etc. genähert) sein
- Vollständige Verfahren: finden immer korrekte Lösung, können ermitteln ob keine Lösung existiert
- Probabilistisch vollständige Verfahren: Falls eine Lösung existiert konvergiert die Wahrscheinlichkeit, dass eine Lösung gefunden wird, bei fortschreitender Zeit gegen 1. Existiert keine Lösung, kann dies nicht ermittelt werden.
- Voronoi-Diagramm: visualisiert die Zerlegung eines Raumes in Regionen basierend auf vorgegebenen Punkten/Zentren, Region = Menge aller Punkte, deren Abstand zum Zentrum geringer ist, als zu allen anderen Zentren, alle Punkte auf der Grenze zwischen zwei Regionen besitzen den gleichen Abstand zum eigenen und zum benachbarten Zentrum.
- Sichtgraphen: verbinde jedes Paar von Eckpunkten auf dem Rand durch gerades Liniensegment, falls die Linie kein Hindernis schneidet, Erweiterung der Hindernisse (Kreis, Rechteck), um Kollisionen mit Hinderniskanten zu vermeiden, in 2D findet man kürzeste Wege in 3D nur beliebigen Weg



- Zellzerlegung: Freiraum in Zellen zerlegen und Relationen in Graph darstellen, exakt (Menge aller Zellen = Freiraum) oder approximiert (möglichst gut durch Zellen vordefinierter Form beschreiben)

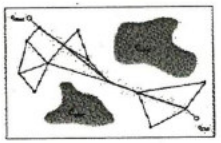


- Potentialfelder: Roboter bewegt sich unter Einfluss von Kräften, die Potentialfeld auf ihn ausüben, Hindernisse haben abstoßendes Potential, Ziel anziehendes (globales Minimum), Achtung: mögliches feststecken in lokalen Minima

11.2 Planungsverfahren für Manipulatoren

11.2.1 Probabilistic Roadmaps PRM

- Erzeugung einer kollisionsfreien Straßenkarte (zufällige Erzeugung von kollisionsfreien Stichproben (Sampling), verbinden durch kollisionsfreie Pfade) → Graph
- Verbinde Start und Ziel mit dem Wegenetz (Suche im Graphen)
- effizient, da Freiraum nur approximiert wird

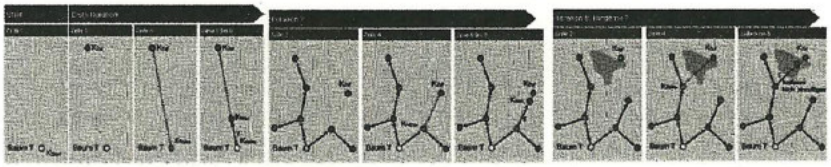


11.2.2 Rapidly-exploring Random Trees (RRT)

- probabilistisch vollständig, effizient für hochdimensionale Problemstellungen

```
Data: K_Start, n, ε
Result: T
BUILD_RRT(K_Start, n, ε):
  T.init(K_Start)
  for i ← 1 to n do
    K_Zuf ← RANDOM_CONFIGURATION()
    K_Nahe ← NEAREST_VERTEX(K_Zuf, T)
    K_Neu ← EXTEND(K_Nahe, K_Zuf, ε)
    if VALID(K_Neu) then
      T.add_vertex(K_Neu)
      T.add_edge(K_Nahe, K_Neu)
    end
  end
  return T
```

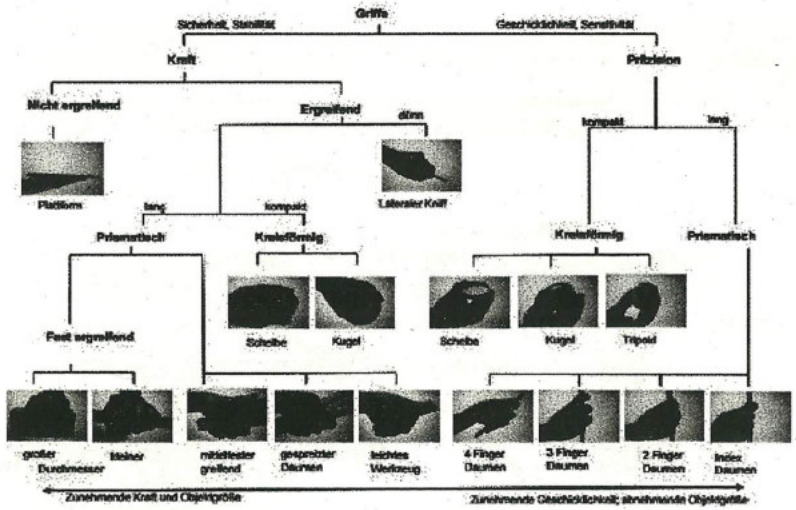
- Zeile 1: Neuer Baum
- Zeile 3: zufällige Konfiguration
- Zeile 4: Bestimmung des nächsten Knoten
- Zeile 5 - 8: Erzeugen einer neuen Konfiguration, falls gültig hinzufügen von Knoten und Kante



- Aufbau zweier Bäume von Start und Ziel, ein Baum wird um K_{neu} erweitert, dann versucht K_{neu} von zweitem Baum zu erreichen, falls nicht möglich Rolle der Bäume vertauschen
- Wahrscheinlichkeit, dass Konten erweitert wird, ist proportional zur entsprechenden Voronoi-Region, RRT expandiert in Richtung Knoten mit größter Voronoi-Region (Voronoi-Bias)
- Glättungs-Algorithmus: zufällige Wahl zweier Knoten im Lösungsweg, falls die Verbindung kollisionsfrei ist, verbinden der Knoten und Löschen der Knoten dazwischen

12 Greifplanung

- menschliche Hand: 22 Bewegungsfreiheitsgrade
- Modellierung: kinematisches Modell (Gelenke, Armelemente...), flächenbasiertes Geometriemodell (Oberflächen)
- Schritte: Greifklassenselektion, Berechnung einer kollisionsfreien Anrückbewegung, Bestimmung von Fingerhindernissen, Berechnung von zugänglichen Griffen, Berechnung von erreichbaren Griffen, Griffselektion
- Cutkosky Grifftaxanomie: prinzipiell 16 Griffarten



12.1 Bewegungstypen

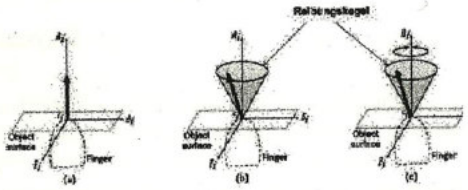
- Bei allen: Kollisionen zwischen Greifer, Objekten, Roboter vermeiden
- Greifen bzw. Loslassen eines Objektes: Auswahl eines sicheren Griffes (Bestimmung geometrischer Relation der Greiffinger zum gegriffenen Objekt)
- Anrückbewegung bzw. Abrückbewegung des Greifers: Planung der Bewegung, um die Greiffinger zu positionieren und orientieren
- Anrück- bzw. Abrückbewegung des Greifers mit gegriffenem Objekt: Planung der Beweggung des Greifers mit gegriffenem Objekt
- Verbinden des gegriffenen Objektes mit anderen Objekten: sensorüberwachte und/ oder -geführte Bewegung
- Transferbewegung des Greifers: hlhere Ausführungsgeschwindigkeit, geringere Genauigkeitsanforderungen

12.2 Nebenbedingungen des Greifvorgangs

- interne: Gültigkeit (Überlappung der Greifmerkmale des Objektes und der Greiferfinger), Kollisionsfreiheit (zwischen Greifer und Objekt), Zugänglichkeit (kollisionsfrei erreichbar)
- externe: kollisionsfreie An/Abrückbewegung, Roboterkinematik (Griff im Arbeitsraum, Trajektorien möglich), Stabilität des Griffes (relative Lage/Orientierung des Objektes zu Greifer bleibt gleich), Stabilität der Szene, Aufgabenabhängigkeit (bei

Pick-and-Place eventuell umgreifen, eventuell Ausübung von Kräften und Momenten nötig)

- Griff: Eine Menge von Kontaktpunkten auf der Oberfläche eines Objekts, die potentielle Bewegungen des Objekts unter dem Einfluss externe 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
- Wichtige Faktoren für die Generierung von Griffen bzw. Griffhypothesen: Handkinematik, Griffrepräsentation, Objektwissen, Greifsynthese (analytisch, datengetrieben), verfügbare Merkmale (2D, 2,5D, 3D, visuelle, haptische, ...), Aufgabe
- Fingerspitzengriff-Modell: bestimmt nur Anordnung der Kontaktpunkte auf der Oberfläche des Objektes, Nebenbedingungen werden nicht berücksichtigt (einfacher, aber auch Nachteile)
- Punktkontakt ohne Reibung: nur normale Kraft
- starrer Punktkontakt mit Reibung: normale und tangentiale Kraft, verknüpft durch Coulombsches Reibungsgesetz
- nicht starrer Punktkontakt mit Reibung (Soft-Kontakt): normale und tangentiale Kraft + axiale Momente + Coulombsches Reibungsgesetz
- Kontaktmodelle:



a) Kontakt ohne Reibung (existiert nicht in der Robotik!)
b) Kontakt mit Reibung
c) Soft-Kontakt

- Reibungskegel kann auch durch einbeschriebene Polyeder approximiert werden
- Wrenchvektor $\vec{w}$: fasst Kräfte f_i und Momente τ_i in Kontaktpunkt zusammen $\vec{w} = (f_x, f_y, f_z)^T$ für planaren Griff und $\vec{w} = (f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)^T$ für einen räumlichen Griff, Index n, t, θ für normale und tangentiale Kräfte und axiale Momente, c für korrespondierende Skalare

- Greifmatrix: repräsentiert geometrische und physikalisch Eigenschaften eines Fingerspitzengriffes mit m Kontaktpunkten $G = [{}^1\vec{w}_n, {}^1\vec{w}_t, {}^1\vec{w}_\theta, \ldots, {}^m\vec{w}_n, {}^m\vec{w}_t, {}^m\vec{w}_\theta] \in \mathcal{R}^{6\times 3m}$
- Gleichgewichtsgriff: Summe aller Kräfte und Momente, die auf das gegriffene Objekt wirken, ist Null

- Kraftgeschlossene Griffe: sinnvoll um unbekannte externe Kräfte zu kompensieren, Kinematik der Hand erzeugt aktiv Kräfte um externer Störung zu widerstehen, ohne Reibung mindestens 4 Kontaktpunkte (planarer Griff) mit Reibung 3, häufig für Präzisionsgriffe (da wenige Kontaktpunkte), hängt von Kinematik des Roboters ab

- Formgeschlossene Griffe: stärkere Einschränkungen als bei kraftgeschlossenem Griff, Kontakte an sich verhindern, dass Objekt sich bewegen kann (Nichtdurchdringungseigenschaften), Kräfte werden nicht berücksichtigt, nur Position der Kontaktpunkte und Oberflächen-Normalvektor, mindestens 4 Kontaktpunkte für planaren Griff und mindestens 7 für 3D Objekte, häufig für Kraftgriffe verwendet, Analyse auf Basis von Geometrie

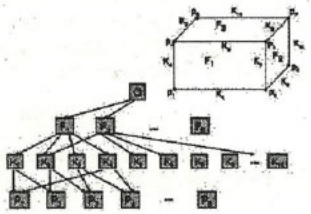
13 Umweltmodellierung

- nötig für adaptive Roboteraufgaben (neben Sensorik und Planungsmethoden)
- unterteilbar in Szenenmodelle und Objektmodelle
- Objektmodelle unterteilen sich in Geometrische Modelle und zusätzliche Eigenschaften
- zusätzliche Eigenschaften: Masse, Oberflächeneigenschaften, Temperatur, Steifigkeit, Greifpunkte, Füllmenge, ... (stark aufgabenabhängig)

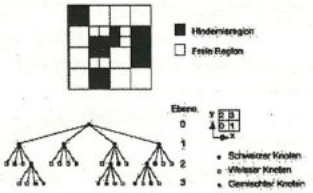
13.1 Geometrische Modelle

- Anwendungsgebiete: Umweltwahrnehmung (Objektklassifizierung, -lokalisierung, Bewegungserkennung/-klassifikation), Bewegungsplanung, Greifplanung, ...
- Punktwolken: beliebige Menge an Raumpunkten, Daten aus Sensorik, hochgradig approximativ (einfach, aber Datenlücken), z.B. zur Lokalisation, Klassifikation, Kartierung

- Approximative Oberflächenmodelle: Aufbau aus Netz (Mesh) von einfachen Einzelflächen, einfache Definition/Algorithmen, aber hoher Speicherbedarf und Rechenaufwand z.B. Bewegungsplanung, dynamische Simulation
  - Dreiecksflächen: einfachste Fläche
  - Bilineare Viereckflächen: Flächenlemente können gekrümmt sein, aber Rechnen mit gekrümmten Flächen aufwendig
  - Bezierflächen: Erweiterung der Bezierkurven, Fläche aus Bernsteinpolynomen
- Boundary Representation: Hierarische Darstellung durch begrenzende Elemente Flächen F, Kanten K, Ecken P, ..., Vorteil: klar welche Kanten/Ecken zwischen welchen Flächen sind etc.

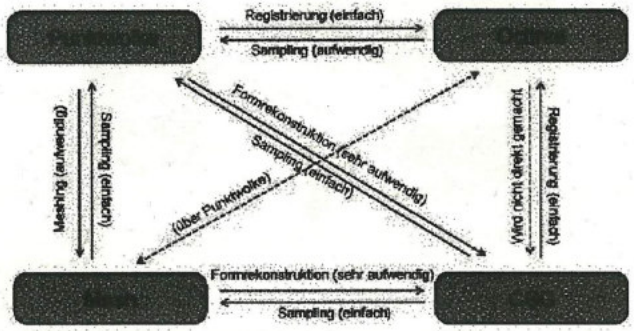


- Approximative Zellenbelegung: Aufbau aus disjunkten Elementarzellen z.B. Bewegungsplanung, Dynamische Simulation elastischer Materialien
  - Voxeldarstellung: belegt/ nicht belegt, einfache Darstellung und Berechnung, aber oft nicht präzise genug
  - Octree/Quadtree: Raum wird in mehrere (8 bzw. 4) Zellen zerteilt, komplett belegte Zellen werden markiert, teilbelegte Zellen bis Mindestgröße weiter zerlegt, teilbelegte kleinste Zelle wird als belegt markiert, dann Baum anlegen, kollisionsfreier Pfad geht über freie Kacheln



- Analytisch-parametrische Modelle: Fläche wird analytisch dargestellt, geschlossene Darstellung (wenig Speicherbedarf), einfache Berechnungen, aber nur für wenige Flächen möglich
- Parametrische Volumenmodelle: Grundkörper (Anpassung durch Parameter) + topologische Operationen (Schnitt, Vereinigung, ...), Objekte können mit geringem Eingabeaufwand eindeutig beschrieben werden, Implementierungsaufwand ist hoch, Freiformflächen schwierig

- Constructive Solid Geometry (CSG): Menge einfach Grundkörper (Quader, Prisma,Zylinder, Kegel, Ellipsoid, Rotationskörper), können parametrisiert werden (Länge, Breite, Radius, ...), verschiedene Operationen definiert (Vereinigung, Schnitt, Differenz, Sweep (Grundelement z.B: Fläche wird verschoben, durchdrungener Raum = neues Objekt)), Speicherung als Binär-Baum (Knoten als Operation, Teilbäume = Grundkörper) z.B. dynamische Simulation,
- Umwandlungen zwischen Modelltypen



14 Interaktive Programmierung

- klassische Programmierung ist aufwendig und erfordert Expertenwissen, besonders schwierig bei komplexer Umgebung oder flexiblen Aufgaben ⇒ Interaktive Programmierung besser geeignet
- Demonstration & Aufzeichnung ⇒ Segmentierung & Interpretation ⇒ Abstraktion & Simulation ⇒ Ausführung
- Bediener gibt Sollwerte für Regelung vor

14.1 Klassifizierung der Roboterprogrammierverfahren

14.1.1 Programmierort

- on-line (auch direkt oder prozessnah): Programmierung direkt am Roboter
- off-line (auch indirekt oder prozessfern): Programmierung ohne Roboter, textuell, graphisch oder interaktiv
- Hybride Verfahren und Interaktive Verfahren: zwischen on-line und off-line

14.1.2 Art der Programmierung:

- Direkte Programmierung:

- Einstellen des Roboters: ältestes Programmierverfahren, Bewegungsbereich der Gelenke durch Stopper eingeschränkt, Gelenke werden einzeln bis zum Anschlag bewegt
- Teach-In Programmierung
- Play-Back Programmierung
- Master-Slave Programmierung (Sonderfall Teleoperation)
  * Bediener führt kleinen Master-Roboter (= kinematisches Modell des Slave-Roboters)
  * Bewegung wird auf Slave-Roboter synchron übertragen, Slave = Kraftverstärker
  * Anwendung: Handhabung großer Lasten/Roboter
  * Nachteil: teuer, 2 Roboter nötig
  * Vorteil: Möglichkeit schwere Roboter zu programmieren
- sensorunterstützte Programmierung:
  * manuell: Programmierer führt Programmiergriffel entlang der abzufahrenden Bahn, Erfassung durch Sensoren, inverse Kinematik, Gelenkwinkel abspeichern
  * automatisch: Vorgabe des Star- und Zielpunktes, sensorische Erfassung der Sollkontur
  * z.B. zum schleifen, entgraten von Werkstücken
- Vorteile direkte Programmierung: schnell bei einfachen Trajektorien, sofort anwendbar, geringe Fehleranfälligkeit, keine Programmierkenntnisse notwendig, kein Umweltmodell nötig
- Nachteile: hoher Aufwand bei komplexen Trajektorien, nur am Roboter möglich, spezifisch für Robotertyp, Verletzungsgefahr

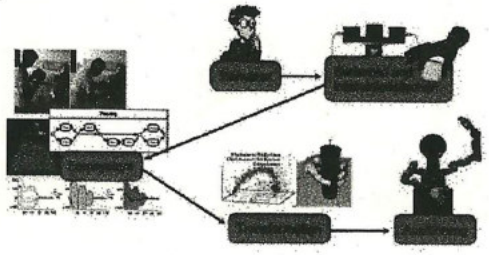
- indirekte Programmierung
  - Textuelle Verfahren: Programmierung mittels höherer Programmiersprachen z.B. APT (Automatically Programmed Tools) oder EXAPT (Extended Subset of APT) ⇒ Robotersteuerprogramm
    * Vorteile: Programmierung ist unabhängig von Roboter, strukturierte, übersichtliche Programmierlogik, komplexe Programme
    * Nachteile: Programmierkenntnisse nötig, schlechte Korrekturmöglichkeiten
    * früher: Programmänderung = Hardwareänderung, heute flexibler, da Steuerungs- und Regelungsablauf programmiert werden
  - Graphische Verfahren:
    * textuelle Beschreibung + graphische Darstellung des Weltmodells ⇒ Simulation der Roboterprogramme
    * Aufgabenorientiert oder Bewegungsorientiert
    * Vorteile: Programmierung unabhängig von Roboter, weniger Programmierkenntnisse nötig, leichte Fehlererkennung, einfaches Erstellen komplexer Programme

    * Nachteile: Leistungsfähige Hardware für Visualisierung und Simulation, Roboter und Umwelt müssen modelliert werden (komplex)

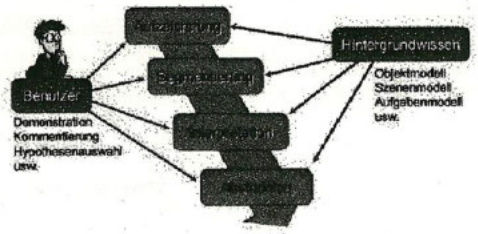
- Gemischte Verfahren:
  - Graphische Programmierung basierend auf sensorielle Erfassung der Benutzervorführung
  - Vorteile: wenig Programmierkenntnisse nötig, einfaches Erstellen komplexer Programme, leichte Fehlererkennung
  - Nachteile: sensorielle Benutzererfassung ungenau, leistungsfähige Hardware für Signalanalyse etc., komplexe Modelle nötig

14.1.3 Abstraktionsgrad der Programmierung:

- explizite oder roboterorientierte Programmierung: Bewegungen und Greiferbefehle sind direkt in eine Programmiersprache eingebunden, Wie ist es zu tun?
- Implizite oder aufgabenorientierte Programmierung: Aufgabe wird beschrieben, Was ist zu tun?
- Phasen: Modellierung der Umwelt, Spezifikation der Aufgaben, Erzeugung der Roboterprogramme, (Überprüfung, Simulation), Ausführung
- Grundidee der Interaktiven Programmierung: Mensch ist Domänenexperte (Manipulation), explizite Demonstration, sensorielle Erfassung, Erzeugung der internen Repräsentation des Roboterprogramms, Abbildung auf das Robotersystem, Ausführung
- Komponenten:



  - Interaktionsformen: physische, graphische, ikonische Demonstration, Kommentierung
  - Sensoren: Kameras, taktile Sensoren, Trackingsystem, Datenhandschuh
  - Lernsystem - Wissensrepräsentation: manipulatorabhängig (Aktionssequenzen, Gelenkwinkel-, Kraft und Momenttrajektorien), manipulatorunabhängig (Elementaroperatoren mit Start-, End- und Fehlerkriterien, Effekte in der Umwelt sind manipulatorunabhängig)



- Transformation von Demonstrationsumgebung auf Ausführungsumgebung: Unterschiede z.B. in Szene (Objektanzahl,-typen,-anordnung), werden durch Planungsmethoden (Griff, Bewegung) berücksichtigt

15 Greifen auf ARMAR

- Objekt erkennen und lokalisieren: mithilfe CAD Objektmodell, visuell durch Farbe oder Textur, auch Hand lokalisieren vor Griffplanung, Vision System
- Bahnplanung für Arm, Griff Auswahl und Ausführung: Arbeitsraum des Roboters bekannt, In Objektdatenbank (z.B. KIT Object Model Database) sind alltägliche Objekte (von z.B. Stereo Kameras) + Griff-Set gespeichert
- Griffplanung: Offline
- Griff: Grasping center point GCP (wo TCP auf Objekt soll), Approach Vector (Bahn auf der sich Hand Objekt nähert), Handgelenksposition (Visuell durch Kugeln am Handgelenk erkannt) und ursprüngliche Finger Konfiguration z.B. Tripod 3 Finger Griff zum Tür öffnen (5 Finger schwer rauszuziehen)
- Bekannte Objekte: Domäne der Greifplanung, komplettes Objektmodell vorhanden, schwer aber am einfachsten, aus Forschungssicht langweilig, in Industrie schwer da 100%ige Sicherheit nötig
- Ähnliche Objekte: Objektklasse bekannt (z.B. Flaschen), Ansatz: Das Wissen über das Greifen bekannter Objekte für das Greifen neuer Objekte benutzen, schwieriger als das Greifen bekannter Objekte, aktuelles Forschungsthema, vielversprechende Ansätze, Schwierigkeit: Ähnlichkeit erkennen
- Unbekannte Objekte: kein Wissen über das Objekt vorhanden, Sensordaten (von Stereo vison, Laser scan, haptic data, ...) unvollständig, Unterscheidung Objekt/Hintergrund, (unvollständiges) Objekt Modell daraus bilden, z.B. mit Multi Sensor Fusion, schieben des Objektes, ..., größtes Problem bei Greifplanung
- Grasping by parts:
  - Idee: Objekt in einfache Abschnitte unterteilen und Griffe für diese Abschnitte planen (für ähnliche Objekte)

  - 1. Schritt: Hand und Objekt Modelle in Simulator laden
  - 2. Schritt: Griff Kandidaten generieren (Kontaktpunkt mit Objekt, Näherungsvektor, Handorientierung, Form der Hand), Suchraum durch Heuristik einschränken, Decomposition der Objekte (Box, Mediale Achse, Grundformen ...)
  - 3. Schritt: Hand Griffpunkt entlang Näherungsvektor nähren bis sie mit Objekt kollidiert ⇒ Finger schließen bis Kollision ⇒ Kontakte auswerten (Ist Griff kraftgeschlossen?)
  - Griffstabilität: Kontaktpunkte und Normalenvektoren berechnen, Reibungskegel für jeden Kontaktpunkt berechnen, Grasp Wrench Space (GWS, 6D) ist konvexe Hülle über alle Reibungskegel, minimale Distanz zwischen GWS Zentrum und GWS Fläche gibt Auskunft über Stabilität
  - Simulierte Griffe nah an Realität

15.1 Box Decomposition

- Objekte werden in Boxen zerlegt, basierend auf (willkürlich dichter) Punktwolke
- Annahme: Genaue Form des Objekts ist unwichtig zur Griffgenerierung
- Griffgenerierung/Experimente mit Simulator Grasplt! Griff abhängig von Aufgabe z.B. Objekt übergeben
- Minimum Volume Bounding Box (MVBB) Algorithmus berechnet Box aus Punktwolke, Komplexität $O(n \log n + n/\varepsilon^3)$ (ε Maß für schlechtesten Fall)
- Gesucht: Kriterium um ursprüngliche MVBB in kleinere Boxen (die Form besser beschreiben) zu unterteilen
- Ansatz: Iterativ Punktwolke splitten an Ebenen parallel zu ursprünglicher MVBB und 2 neue Boxen fitten, Schnitt ist am besten wenn die neuen Boxen das geringste Volumen haben
- Durch Projektionen Problem auf 2D reduzieren
- jede Box hat 6 Flächen, die man greifen kann (am Mittelpunkt, entlang Normale) je 4 Griffe (4 Handorientierungen)
- Oberflächen, die größer sind als Handöffnung können nicht gegriffen werden, verdeckte Oberflächen auch nicht
- Vergleich: Griffe aus sphärischen Koordinaten und Box Decomposition
- Ergebnis: Box decomposition gibt nur wenige der sphärischen Griffe, aber es sind gute darunter, Griffe aus diesem Verfahren können in Realität verwendet werden
- KA-Hand = 5 Finger

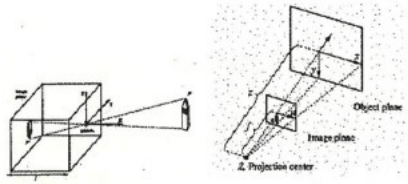
15.2 Mediale Achse

- Ziel: Nur noch geometrisch sinnvolle Griffe testen ⇒ Lokale Symmetrie der Objekte erkennen ⇒ Mediale Achse
- Form durch eingeschriebene Kugeln mit maximalem Durchmesser nähern, mindestens 2 Kontaktpunkte zwischen Objekt und Kugel
- Mediale Achse: Linie durch Kugelmittelpunkte
- repräsentiert topologisches Skelett des Objektes
- Vorteile: Objektgeometrie gut genähert, Details bleiben erhalten, Symmetrie erkennbar, gut geeignet für Greifplanung
- Algorithmus: Objekt Oberfläche ⇒ Mediale Achse ⇒ Achsen in Abschnitte unterteilen (Minimum Spanning Tree MST, Clustern, konvexe Hülle) ⇒ Griffkandiaten generieren ⇒ Griffstabilität testen
- MST: Jeder Punkt wird mit nächsten Nachbarn verbunden ⇒ Kanten, die länger als d_cut sind werden entfernt ⇒ man erhält Cluster
- Für jedes Cluster: Volumen der konvexen Hülle bestimmen ⇒ Punkte auf Rand markieren ⇒ Verzweigungspunkte markieren ⇒ Cluster Schwerpunkt markieren
- Unterteilen der Strukturen in Ringe (Abstand von Schwerpunkt zu nächstem Punkt ist groß, Viele Punkte auf Rand), Sterne (exakt ein Verzweigungspunkt), ...
- Handrollwinkel an Symmetrieachsen/-flächen ausrichten
- Griff an Zacken der Sterne, Verzweigungspunkten, Symmetrieachsen, Rand von offenen Objekten ...
- Mediale Achse ist effizient, auch bei großen Objekten
- Griffe meist stabil und „natürlich"
- Probleme: komplexe Objekte enthalten zu viele Strukturen, zu viele Parameter, ...
- Mediale Kugeln sind besser (nutzt Radius der Kugeln und Winkel zu Berührungspunkten, Kugeln auf Symmetrieachsen/-ebenen besonders interessant)

16 Bildverarbeitung

- sehen wichtig um Umwelt wahrzunehmen
- Aufgabe: Visuelle Informationen müssen (in guter Qualität und digital) aufgenommen werden, relevante Informationen müssen extrahiert werden
- Bilderfassung: Hardware
- Bildverarbeitung: überwiegend Software (bei intelligenten Kameras auch Hardware)

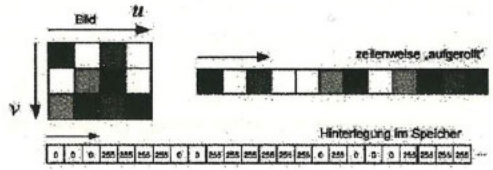
- Lochkamera: normal und Positivlage (Rest nur Strahlensatz)



16.1 Bildrepräsentation

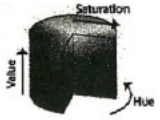
- Bild = 2D Pixel Gitter
- Koordinaten: u (horizontal), v (vertikal), Ursprung oben links
- Monochrombild: Diskrete Funktion $Img : [0 \dots n-1] \times [0 \dots m-1] \rightarrow [0 \dots q]$
- Graustufenbild:
  - Helligkeitswert für jeden Pixel, 1 Byte $[0, 255]$, 0 schwarz, 255 weiß
  - Pixel werden zeilenweise, linear im Speicher abgelegt, von oben links nach unten rechts (Bei Bitmap von unten links nah oben rechts)

- Farbbild: verschiedene Farbmodelle nach Anwendung/erreichbarem Farbraum
- RGB-Modell:
  - für Monitore
  - Additive Farbmischung
  - jedes Pixel → 3 Bytes (rot, grün, blau)
  - Byte = 8 Bit → 256 Abstufungen für jeden Farbwert, insgesamt 16,8 Mio. Farben
  - 0 minimale, 255 maximale Intensität



- HSI/HSV:
  - geeignet für Farbsegmentierung
  - Hue (Farbnuance), Saturation (Sättigung), Intensity/Value (Helligkeit)
  - Trennung von Helligkeit und Farbwert ⇒ unempfindlich gegen Beleuchtungsänderungen

$$H = \begin{cases} \theta, & \text{falls } B < G \\ 360 - \theta, & \text{sonst} \end{cases} \quad (17)$$



$$\theta = \arccos\left(\frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right) \quad (18)$$

$$S = 1 - \frac{3}{R+G+B}\min(R,G,B) \quad I = \frac{1}{3}(R+G+B) \quad (19)$$

  - Änderung der Lichtintensität(z.B. Licht heller drehen): R,G,B verändern sich um selben Faktor ⇒ H und S bleiben gleich, I ändert sich
  - Farbsprektrum des Lichts ändert sich (z.B. Wolke schiebt sich vor Sonne) : alle Werte verändern sich, aber RGB mehr als H und S
- CMYK-Modell: Farbdrucker, ...

16.2 Filteroperationen

- Das Filter in der Bildverarbeitung (spatial filters, spatial masks, kernels, windows)
- Filtern: Pixelwert im Ergebnisbild = Filtermatrix über Pixel im Originalbild, aufeinanderliegende Werte des Filters und des Bildes multiplizieren und aufsummieren
- Lineare Filter: z.B. Mittelwertfilter, Gauß-Filter, Prewitt
- Tiefpassfilter: Glättung, Rauschelimination z.B. Mittelwertfilter, Gauß-Filter
- Hochpassfilter: Kantendetektion z.B. Prewitt, Laplace

- Kombinierte Operatoren: z.B. Laplacian of Gaussian
- Mittelwertfilter: Größe beliebig wählbar, Beispiel 3×3 Mittelwertfilter, alle 9 Werte durch 9 dividieren und addieren, am Rand Nuller hinzufügen oder Rand auf Null setzen
- Prewitt:
  - Gradient wird genähert, großer Intensitätsunterschied an Kanten

  - Prewitt-X Filter: $P_x = \frac{\partial g(x,y)}{\partial x} \approx \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$, gute Detektion vertikaler Kanten
  - Prewitt-Y Filter: $P_y = \frac{\partial g(x,y)}{\partial y} \approx \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, gute Detektion horizontaler Kanten
  - Kombination, Gradientenbetrag $M \approx \sqrt{P_x^2 + P_y^2}$, beste Kantendetektion

16.3 Segmentierung

- Segmentierung bei Trajektorien ≠ Segmentierung bei Bildverarbeitung
- Aufteilung eines Bildes in aussagekräftige Segmente
- jedes Pixel wird mindestens einem Segment zugeordnet
- Identifikation interessanter Bildregionen für Analyse, Erkennung und Klassifikation
- Verfahren: Schwellwertfilterung, Clustering, Kantenextraktion, Region-Growing
- Schwellwertfilterung: Graubild → binäres Bild, Intensität jedes Pixel wird mit Schwellwert abgeglichen, 255 falls größer, 0 falls kleiner, bei Farbbildern Intervallschranken im HSY-Farbraum
- Probleme bei Schwellwertfilterung: wechselnde Lichtbedingungen, Reflexionen, Schattenwürfe

16.4 Morphologische Operatoren

- für Nachbearbeitung binärisierter Bilder
- Dilatation: vergrößert Pixel zu größeren Bereichen
- Erosion: entfernt vereinzelte Pixel und schwach zusammenhängende Pixelgruppen
- Effekt hängt von berücksichtigter Pixelnachbarschaft ab z.B. 4-er, 8-er, ...

16.5 Registrierung von Punktwolken mit ICP

- Registrierung: Zusammenführen von Punktwolken, welche das gleiche Objekt aus unterschiedlichen Ansichten beschreibt
- Überführung in übergeordnetes Koordinatensystem z.B. Weltkoordinatensystem
- Extrinsische Kalibrierung der Kameras erforderlich (Wo ist Kamera in übergeordnetem Koordinatensystem)
- Iterative Closest Point (ICP): Gängiger Algorithmus (in 2D und 3D) für die Registrierung zweier Mengen, minimiert Distanz zwischen 2 Punktwolken
- 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 $D_K = \sum_i ||a_i - T_k \cdot b_i||^2$
  - D_K kombiniert Translation und Rotation
  - Update: Wende T_K auf alle Punkte aus B an
  - Abbruchkriterium: Iterationsanzahl, Schwellwert für $D_{K-1} - D_K$
- Vorteile ICP: anwendbar für verschiedene Darstellungsformen (Punkte, Normalenvektoren, ...), einfache mathematische Operationen, schnelles Ergebnis
- Nachteile ICP: symmetrische Objekte problematisch (z.B: Flasche welche Seite ist welche), Konvergenz in lokales Minimum möglich, Überlappung der Punktwolken erforderlich (damit klar ist,dass sie zum selben Objekt gehören)
- Optimierungsfunktion Quadrat (v = Eckpunkte des Quadrates, p = Punkte der Punktwolke): $F_Q(V) = \sum_{j=0}^{3}(argmin_i||v_j - p_i||)^2$
- Optimierungsfunktion Kreis: (Kreismittelpunkt p_m): $F_C(p_m) = \sum_{i=0}^{3}(||p_m - p_i|| - r)^2$
- Gradient der Optimierungsfunktion für den Kreis:

$$\nabla F_C(p_m) = \frac{1}{|P|}\begin{pmatrix} \sum_{i=0}^{3} 2(||p_m - p_i|| - r)\frac{p_m^T - p_i^T}{||p_m - p_i||} \cdot (1,0)^T \\ \sum_{i=0}^{3} 2(||p_m - p_i|| - r)\frac{p_m^T - p_i^T}{||p_m - p_i||} \cdot (0,1)^T \end{pmatrix} \quad (20)$$

- ICP mittels Gradientenabstieg:

```
Data: P, r, α, K, ε
Result: p_{m,K} = (2.5, 2)
k = 0;
p_{m,k} = (6, 0);
maxIterations = K;
errorThres = ε;
while k < maxIterations do
    p_{m,k+1} = p_{m,k} − α∇F_C(p_{m,k});
    if ||p_{m,k+1} − p_{m,k}|| < 1e − 6 then
        break;
    else
        p_{m,k} = p_{m,k+1};
        k + +;
    end
end
```

16.6 Random Sample Consensus/RANSAC

- iterative Methode zur Schätzung von Modellparametern aus Datenpunkten z.B: Schätzung von Linien in 2D, Schätzung von Ebenen und Primitiven in 3D
- 1.Schritt: Wähle zufällig die minimale Anzahl an Punkten aus, die nötig ist um die Modellparameter zu berechnen
- 2.Schritt: Schätzung des Modells aus dem ausgewählten Datensatz
- 3.Schritt: Bewertung der Modellschätzung, berechne die Teilmenge der Datenpunkte (Inliers), deren Abstand zum Modell kleiner ist als ein vordefinierter Schwellwert

- 4.Schritt: Wiederhole 1-3 bis das Modell mit den meisten Inliers gefunden wird
- Vorteile: allgemein, einfach, robust für wenige Ausreißer, vielseitig
- Nachteile: nicht-deterministisch, viele Parameter, viele Iterationen, Verhältnis Inliers/Outliers nicht immer groß genug

16.7 Normalenschätzung in 3D Punktwolken

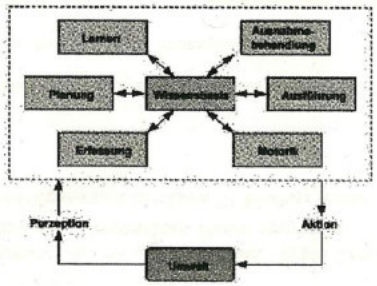
- Zusätzliche Oberflächeninformation durch Einbeziehung von lokalen Nachbarpunkten
- Grundlage für weiterführende Algorithmen z.B. Segmentierung oder Objekterkennung
- Hauptkomponentenanalyse PCA: Kovarianzmatrix $C = \frac{1}{k}\sum_1^k(p_i - \bar{p})(p_i - \bar{p})^T$, p_i k Nachbarpunkte, $\bar{p}$ Mittelpunkt aller k Nachbarn
- Eigenvektor zu kleinstem Eigenwert von C ist Normale (z.B. Normalen in gleicher Richtung gehören zum gleichen Element)

16.8 Simultaneous Localization and Mapping/ SLAM

- Problem: Navigieren in unbekannter Umgebung, dabei Karte der Umgebung erstellen und konstant aktualisieren
- nötig für: autonome Roboter, unbekannte Umgebungen, um den Roboter zu lokalisieren, navigieren ohne GPS oder ähnliches
- Huhn-Ei Problem: brauchen Karte um Roboter zu lokalisieren, und Roboterposition um Karte zu erstellen
- Lokalisieren: Roboterposition mit gegebener Karte abschätzen
- Mapping/Kartographieren: Karte aus gegebenen Orten erstellen
- SLAM: Gleichzeitig Karte erzeugen und Roboter lokalisieren
- Lösung: Bewegung von Kamera (Kamera Parameter) und Roboter (eindeutige Landmarken (unabhängig von Sichtrichtung) z.B. Ecken, Texturen, ...) verfolgen
- 1.Schritt: Abschätzen wie weit Roboter sich bewegt hat
- 2.Schritt: Neue Landmarken aufnehmen
- 3.Schritt: Interne Repräsentation erneuern (Messunsicherheiten berücksichtigen, schwer zu modellieren)
- Irgendwann findet man alte Landmarken wieder ⇒ Schleife geschlossen, Schätzung kann verbessert werden

17 Autonome Roboter

- autonom: funktionieren ohne führendes Element, oft energetisch autark, müssen Aufgabenklasse kompetent und sicher ausführen können, muss Ausnahmesituationen beherrschen, verstehen meist Schriftsprache, sprechen noch schwer
- humanoide Roboter sind autonom
- Industrieroboter: starr (nicht autonom, Umgebung muss bekannt sein)
- Serviceroboter: adaptiv (laufen, manipulieren)
- Personal Robot (noch in Entwicklung z.B. von Google): extrem adaptiv
- Einsatzgebiete Serviceroboter: Inspektionen (z.B. Starkstromleitungen, Abwasserkanäle), Drohnen (Wasser, Flug), Erntemaschinen, Pflege von Alten und Kranken, Diener im Haushalt
- Basisfunktionen: Orientierung in Bewegung, Umgebungserfassung, Umweltmodellierung, Ortung, Bahnplanung, Bewegungsführung, Sicherheit und Kollisionsschutz, Handhabung
- Teilsysteme: Handhabungsarm, Endeffektor, Mobile Plattformen, Antriebstechnik, Steuerung, Sensorik, Mensch-Maschine-Schnittstelle, Sicherheitssystem
- ARMAR III a (der in der Küche), ARMAR IIIb (der der nach macht), ARMAR IV (mit Beinen)
- Es muss zur Laufzeit Code ausgeführt werden, Interpreter, aufgabenorientiert programmieren
- Fähigkeiten eines autonomen Roboters



- Architektur: organisiert Zusammenspiel der Komponenten z.B. Planung, Regelung, Steuerung, Transformation, ...
- Hybride Architekturen: meist 3 Schichten Modell, Oben Planung, Mitte Sequentialisierung (Ausführung/Abspielen von Plänen), Unten Verhaltensebene, z.B. autonom fahrende Autos

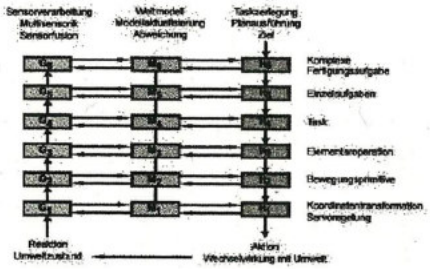
- Kognitive Architekturen: lernfähig, Vorbild: kognitive Prozesse des Menschen, Ziel: Hinzufügen neues Handlungswissens, Eigendynamik, Erweiterbarkeit, Generalisierung des Systems
- gibt symbolische, hybride, emergente kognitive Architekturen
- Symbolische kognitive Architekturen: basieren auf symbolischer Repräsentation, Vorschriften oft in Form von Regeln
- Emergente kognitive Architekturen: Basierend auf einfachen Grundelementen, komplexe Funktionen durch Verschaltung der Elemente realisieren, Repräsentationen sind subsymbolisch und verteilt, z.B. Neuronale Netze

17.1 Funktionsorientierte Architekturen

- generell: Sense → Plan → Act, unidirektionaler Informationsfluss, schlechte für dynamische Umgebungen und Interaktion, aber einfacher Aufbau, wenn eine Komponente ausfällt ⇒ Totalausfall (bei verhaltensorientiert nicht)

17.1.1 hierarchisch

- hat sich bei komplizierten Robotern durchgesetzt z.B. im All, meiste Industrieroboter so aufgebaut
- NASREM-Modell: Aufteilung in 4 bis 6 Ebenen mit je 3 Modulen, primitiv unten, komplex oben

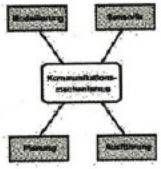


- H-Modul: Taskzerlegungs-, Planungs- und Ausführungsmodul
  - erhält eine Aufgabe aus der oberen Ebene
  - Aufgabe wir mithilfe M-Modul in Teilaufgaben zerlegt
  - M-Modul wird aktualisiert
  - Teilaufgaben werden einzeln an untere Ebene weitergereicht
  - Plandekomposition: z.B. Verbinde Teil A mit Teil B → Greife Teil B → Bewege Endeffektor zu Anrückframe von Teil B → bestimme Anrückframe → Gelenkvektor für Roboter (Folge von Stellmomenten)
  - Menge von Plan- und Ausführungseinheiten
- G-Modul: Sensorverarbeitungsmodul
  - Signale der Elementarsensoren aus der unteren Ebene werden z.B. zu Punkten oder Vektoren fusioniert
  - werden weiter oben zu Merkmalen und an M-Modul weitergereicht

- Filter-/Detektionsoperationen, berechnet Merkmale, Muster, Objekte, Ereignisse und Relationen mithilfe der Daten aus dem M-Modul

- M-Modul: Weltmodell- und Referenzdatenmodul
  - beantwortet Anfragen vom G- und H-Modul
  - berechnet Vorhersagen, bewertet Pläne
  - nimmt Modellaktualisierung von G-Modul, liefert Zustandsvorhersagen zurück, die das G-Modul mit beobachten Daten vergleicht
  - Gibt Sensorinformationen an Aktor(H-Modul) weiter

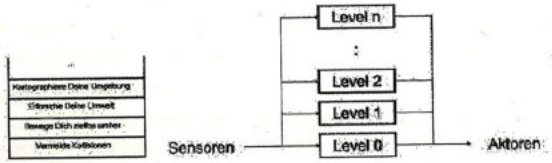
17.1.2 verteilt



- Menge spezialisierter Teilsysteme, hoher Grad an Parallelität
- Kommunikationsmechanismus als Wurzelknoten
- Synchronisation muss festgelegt sein, Ausnahmesituationen, Fehler

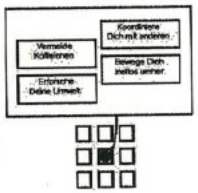
17.2 Verhaltensorientierte Architektur

17.2.1 Hierarchisch



- Steuerung durch Verhaltensmuster bzw. Reflexe
- Muster: Reaktion des Systems auf bestimmte Sensorstimuli (Umweltsituation)
- hierarchische Anordnung, übergeordnete Verhalten können darunter liegende hemmen

17.2.2 Verteilt



- Menge von unabhängigen Teilsystemen mit identischen Verhaltensmustern
- Koordination erfolgt über ein Verhaltensmuster
- z.B. Multi-Agenten-System: Selbstorganisation (biologisch motiviert), assoziative Speicherung von Information, Erkennung von bestimmten Mustern z.B. neuronale Netze