

# Grundlagen der Robotik

**Zusammenfassung**

Fabian Damken

9. November 2021



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

---

# Inhaltsverzeichnis

---

<b>1. Einleitung</b>	<b>11</b>
1.1. Was ist ein Roboter?	11
1.2. Was ist KI?	11
1.3. Was ist Robotik?	11
1.4. Sense – Plan – Act	11
1.5. Geschichte der Robotik	12
1.5.1. Historische Entwicklung	12
1.5.2. Die drei Gebote der Robotik	12
1.5.3. Autonome Fahrzeuge	12
1.5.4. Entwicklungstrend	12
1.6. Herausforderungen	12
1.6.1. Humanoide Bewegung	12
1.6.2. Roboter für menschliche Mobilität	12
1.6.3. Roboter-Avatare	12
1.6.4. Die Robotik an sich	12
<b>2. Räumliche Darstellungen und Transformationen</b>	<b>13</b>
2.1. Mathematische Grundlagen und Notation	13
2.1.1. Vektoren	13
2.1.2. Matrizen	13
2.2. Klassische Transformationsbeziehungen	13
2.3. Rotation eines Koordinatensystems	14
2.3.1. Rotationsmatrizen	14
2.3.2. Verkettete Rotationen	14
2.3.3. Winkelparameter	15
2.4. Homogene Transformationen	15
<b>3. Roboterkinematik</b>	<b>16</b>
3.1. Vorwärtskinematik	16
3.1.1. Kinematische Ketten	16
3.1.2. Kinematische Modellbildung	16
3.1.3. Denavit-Hartenberg (DH) Konventionen	16
3.2. Rückwärtskinematik (Inverse Kinematik)	19
3.2.1. Numerische Berechnung	19
3.2.2. Analytische Lösung	20
3.3. Genauigkeit des kinematischen Modells	20
<b>4. Geschwindigkeit, Jacobi-Matrix und statische Kräfte</b>	<b>21</b>
4.1. Vektor der Winkelgeschwindigkeiten	21

4.2.	Jacobi-Matrix eines Manipulators . . . . .	21
4.2.1.	Verkettung von Rotationen . . . . .	21
4.2.2.	Zusammenfassung . . . . .	22
4.3.	Inverses Jacobi-Modell . . . . .	22
4.4.	Kinematische Singularitäten . . . . .	23
4.4.1.	Vermeidung . . . . .	23
4.4.2.	Umgang mit unvermeidbaren Singularitäten . . . . .	23
4.5.	Nicht-holonome Kinematik mehrrädriger Fahrzeuge . . . . .	23
4.5.1.	Differentialantrieb . . . . .	23
4.5.2.	Allgemeines Vorwärtskinematikproblem für Fahrzeuge . . . . .	23
4.5.3.	Inverses Kinematikproblem . . . . .	23
4.5.4.	Omnidirektionale Dreirad-Kinematik . . . . .	23
4.5.5.	Weitere Antriebsarten von Fahrzeugen . . . . .	23
4.6.	Statische Kräfte bei Manipulatoren . . . . .	23
<b>5.</b>	<b>Roboterdynamik</b>	<b>24</b>
5.1.	Massenverteilung eines Starrkörpers . . . . .	24
5.1.1.	Transformation von Trägheitstensoren . . . . .	24
5.2.	Newton-Euler Formulierung der Roboterdynamik . . . . .	25
5.2.1.	Iterative Berechnung von INV DYN . . . . .	25
5.2.2.	Bemerkungen und DIR DYN . . . . .	26
5.3.	Lagrangesche Formulierung der Roboterdynamik . . . . .	26
5.4.	Numerische Aspekte . . . . .	27
5.4.1.	Modularität . . . . .	28
5.4.2.	Simulation . . . . .	28
5.5.	Rekursive Verfahren zur Berechnung der Vorwärtsdynamik . . . . .	28
5.5.1.	Verfahren mit expliziter Berechnung der Massenmatrix . . . . .	28
5.5.2.	Verfahren ohne explizite Berechnung der Massenmatrix . . . . .	28
5.5.3.	Multibody Systems Library MBSlib . . . . .	28
5.6.	Geschlossene kinematische Ketten . . . . .	28
5.7.	Berücksichtigung von Nichtstarrkörpereffekten . . . . .	28
5.7.1.	Reibung . . . . .	28
5.7.2.	Elastizität . . . . .	29
5.8.	Spezielle Dynamikmodelle für zweibeinige, humanoide Roboter und deren Stabilitätsregelung . . . . .	32
5.8.1.	Zero-Moment-Point (ZMP) . . . . .	33
5.8.2.	Center of Pressure (CoP) . . . . .	33
5.8.3.	Inverses Pendel und Feder-Masse-Modell . . . . .	33
5.8.4.	Globale Stabilitätsbegriffe . . . . .	34
5.8.5.	Ausblicke . . . . .	34
5.9.	Spezielle Dynamikmodelle für zweibeinige, nicht-humanoide Roboter und deren Stabilitätsregelung . . . . .	34
5.9.1.	Hüpfende Roboter mit Teleskop-Beinen . . . . .	34
5.9.2.	Passive Dynamic Walkers . . . . .	35
<b>6.</b>	<b>Antriebssysteme</b>	<b>36</b>
6.1.	Gebräuchliche Antriebssysteme . . . . .	36
6.1.1.	Hydraulische Antriebe . . . . .	36

6.1.2. Pneumatische Antriebe . . . . .	36
6.1.3. Elektrische Antriebe . . . . .	37
6.2. DC-Bürsten-Motoren . . . . .	37
6.3. Getriebe . . . . .	38
6.3.1. Gewinde . . . . .	38
6.3.2. Riemen-/Seilzug-Getriebe . . . . .	39
6.3.3. Rädergetriebe . . . . .	39
6.4. Alternative und elastische Antriebskonzepte . . . . .	41
6.4.1. Beine . . . . .	41
6.4.2. Neue Materialien . . . . .	41
6.4.3. Compliant Robot Actuation . . . . .	41
6.4.4. Elastische Antriebskonzepte . . . . .	41
6.4.5. Vom Muskel-Skelett-Apparat inspirierte Roboter . . . . .	42
<b>7. Sensoren</b>	<b>43</b>
7.1. Interne Sensoren . . . . .	43
7.1.1. Positionssensoren . . . . .	43
7.1.2. Geschwindigkeitssensoren . . . . .	44
7.1.3. Beschleunigungssensoren (Silizium-Beschleunigungssensor) . . . . .	44
7.1.4. Inertial Navigation System (INS) . . . . .	45
7.1.5. Kraft-Momenten-Sensoren . . . . .	46
7.2. Externe und intelligente Sensoren . . . . .	46
7.2.1. Abstandssensoren . . . . .	47
7.2.2. Visuelle Sensoren . . . . .	49
7.2.3. 3D-Sensoren und Perzeption . . . . .	57
<b>8. Regelung</b>	<b>63</b>
8.1. Lineare Regelung . . . . .	63
8.1.1. Begriffe . . . . .	63
8.1.2. Lineare Systemdynamik und Feder-Masse-System . . . . .	65
8.1.3. PD-Regelung linearer Systeme 2. Ordnung . . . . .	66
8.1.4. Partitionierung des Regelgesetzes durch Feedback-Linearisierung . . . . .	68
8.1.5. Sollwerttrajektorien-Folgeregelung . . . . .	69
8.1.6. PID-Regelung linearer Systeme . . . . .	69
8.1.7. Kaskadenregelung . . . . .	71
8.1.8. Stabilität als Sprungantwortverhalten und PID-Parameter . . . . .	71
8.1.9. Digitale Implementierung eines PID-Reglers und Diskretisierung . . . . .	72
8.2. Nichtlineare Regelung . . . . .	72
8.2.1. Systemlinearisierung . . . . .	73
8.2.2. Modellbasierte Manipulatorregelung . . . . .	73
8.2.3. Adaptive Manipulatorregelung . . . . .	74
8.2.4. Bahnregelung in Weltkoordinaten . . . . .	74
8.3. Kraft-/Momenten-Regelung . . . . .	74
8.4. Bahn-/Kraft-Regelung . . . . .	75
8.4.1. Hybride Regelung . . . . .	75
8.4.2. Parallele Regelung . . . . .	75
8.4.3. Diskussion . . . . .	76

8.5. Nachgiebigkeitsregelung (Compliant Control)	76
8.5.1. Verallgemeinerte Betrachtung nach Hogan	77
8.5.2. Impedanzregelung	77
8.5.3. Admittanzregelung	78
8.5.4. Aktiv-passive Konzepte für Impedanz/Admittanz	78
8.6. Bildgeführte Regelung	79
8.6.1. Bildbasiert	79
8.7. Multimodale Regelung physikalischer Interaktionen	80
8.8. Regelung und Steuerung bei Mensch und Tier	80
8.8.1. Propriozeption	80
8.8.2. Sensoren	80
8.8.3. Zentrales Nervensystem	80
8.8.4. Neurale Integration	80
8.8.5. Informationsverarbeitung	80
8.8.6. Sonstiges	80
8.9. Elementare Roboterbewegungen	80
8.9.1. Schwierigkeiten bei kartesischer Bahnvorgabe	81
8.9.2. Programmierung einer Bahn als Folge elementarer Bewegungen	81
8.9.3. Elementare Bewegungsarten für fahrende Roboter	81
<b>9. Bahnplanung</b>	<b>82</b>
9.1. Bahnplanungsarten	84
9.2. Topologische Wegplanung	84
9.3. Bahnplanung im Arbeitsraum vs. Bahnplanung im Konfigurationsraum	84
9.4. Geometrische Bahnplanung	85
9.4.1. Metrische Darstellung	85
9.4.2. Roadmap-Verfahren	85
9.4.3. Exakte Zellzerlegung	87
9.4.4. Approximative Zellzerlegung	87
9.4.5. Potentialfeld-Methoden	89
9.4.6. Komplexität der geometrischen Bahnplanung	91
9.4.7. Stichprobenverfahren	91
9.4.8. Rapidly Exploring Random Trees (RRTs)	94
9.4.9. Beispiel: MINERVA	94
9.5. Kinematische und dynamische Trajektorienplanung	96
9.5.1. Allgemeine Formulierung	96
<b>10. Navigation mobiler Roboter</b>	<b>98</b>
10.1. Lokalisierung und Positionierung	98
10.1.1. Nichtlineare Ausgleichsrechnung	99
10.2. Selbstlokalisierung und Navigation	100
10.2.1. Metrische und Topologische Beschreibung des Aufenthaltsortes	100
10.2.2. Messungenauigkeiten/-unsicherheiten	101
10.2.3. Lokalisierung mit einer Hypothese	101
10.2.4. Lokalisierung mit mehreren Hypothesen	104
10.2.5. Simultaneous Localization and Mapping (SLAM)	109

<b>11. Middleware und Simulation</b>	<b>113</b>
11.1. Szenarien, Eigenschaften und Herausforderungen . . . . .	113
11.2. Middleware . . . . .	114
11.2.1. Nachrichtenbasierte Kommunikation . . . . .	114
11.2.2. Laufzeit-Effizienz . . . . .	114
11.3. Sicherstellung von Korrektheit und Zuverlässigkeit . . . . .	115
11.3.1. Simulation . . . . .	115
11.3.2. Automatisierte Testabläufe . . . . .	116
11.3.3. Monitoring . . . . .	116
11.3.4. Visuelles Debuggen . . . . .	116
11.3.5. Offline Analyse . . . . .	116
11.3.6. Kooperierendes Verhalten . . . . .	117
<b>12. Steuerung autonomer Roboter</b>	<b>118</b>
12.1. Steuerungsarchitekturen . . . . .	118
12.1.1. Hierarchisches Steuerungsparadigma . . . . .	118
12.1.2. Reaktives Steuerungsparadigma . . . . .	119
12.1.3. Hybrid deliberativ-reaktives Steuerungsparadigma . . . . .	119
12.1.4. Beispiel: Subsumption Architecture . . . . .	120
12.2. Programmierung von Verhalten . . . . .	122
12.2.1. XABSL . . . . .	122
12.2.2. FlexBE . . . . .	123
<b>A. Quaternionen</b>	<b>124</b>
A.1. Einleitung . . . . .	124
A.2. Rechenregeln . . . . .	124
A.3. Umrechnung: Quaternionen zu Rotationsmatrizen und zurück . . . . .	124
A.4. Verkettung von Drehungen . . . . .	124
A.5. Repräsentation der Koordinaten eines Punktes bei Rotation . . . . .	124
A.6. Vergleich mit anderen Darstellungsarten . . . . .	124
<b>B. Zusammenhang zwischen Rotationsmatrix, Drehvektor und Drehwinkel</b>	<b>125</b>
B.1. Umwandlung von Drehvektor und Drehwinkel zu Rotationsmatrix . . . . .	125
B.2. Umwandlung von Rotationsmatrix zu Drehvektor und Drehwinkel . . . . .	125
<b>C. Notationen</b>	<b>126</b>

---

# Abbildungsverzeichnis

---

6.1. Stromkreis eines DC-Bürsten-Motors mit dem Widerstand der Spule $R_a$ , der Induktivität der Spule $L_a$ , dem Ankerstrom $i_a$ , der angelegten Spannung $u_a$ und der induzierten Spannung $u_{ind}$ .	38
6.2. Planetengetriebe mit Sonnenrad (gelb), Planetenrädern (blau), Planetenradträger (grün) sowie Hohlräder (rot). Das rechte Bild zeigt die relative Verschiebung von Sonnenrad und Planetenradträger, nachdem letzterer um $45^\circ$ rotiert wurde. Quelle: Wikipedia	40
7.1. Lineares (links) und rotatorisches (rechts) Potentiometer mit Widerstandselement (blau) und Kontakt (orange).	44
7.2. Typische Struktur eines Sensorsystems, wobei die blau hinterlegten Schritte (größtenteils) analog und die orange hinterlegten Schritte (größtenteils) digital ablaufen.	46
7.3. Klassische Hierarchie der Bildverarbeitung. Dabei findet alles bis zur Bildverarbeitung numerisch und alles ab der Bildbeschreibung symbolisch statt.	50
7.4. Triangulation der Distanz $Z$ eines Objekts $X$ durch zwei Kameras $O$ und $O'$ .	59
8.1. Blockschaltbild einer normalen Steuerung (Feedforward Control).	63
8.2. Blockschaltbild einer Regelung (Feedback Control).	64
8.3. Die dynamischen Größen eines Regelkreises.	64
8.4. Blockschaltbild einer PD-Regelung mit den Soll-Werten $x_d, \dot{x}_d$ .	67
8.5. Blockschaltbild eines PID-Reglers mit den Sollwerten $x_d, \dot{x}_d$ .	70
8.6. Blockschaltbild eines PID-Reglers mit Anti-Windup-Teil.	70
8.7. Blockschaltbild einer allgemeinen Kaskadenregelung.	71
8.8. Blockschaltbild eines verallgemeinerten hybriden Bahn-/Kraft-Reglers mit den Sollwerten ${}^0T_{n,d}$ , ${}^0\dot{T}_{n,d}$ , ${}^0\ddot{T}_{n,d}$ , $n_d$ und $f_d$ .	76
9.1. Hierarchische Auftragsbearbeitung und Planung.	83
9.2. Sichtbarkeitsgraph mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen. Zur Vereinfachung wurden die Verbindungen zum Konfigurationsraumrand weggelassen.	86
9.3. Zellzerlegung mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.	88
9.4. Trapez-Zerlegung mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.	88
9.5. Approximative Zellzerlegung im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.	89
9.6. Quadtree des vierten Quadranten der approximativen Zerlegung aus Abbildung 9.5.	90
9.7. Probabilistic Roadmap, Schritt 1: Zufällige Stichproben von Konfigurationen.	92
9.8. Probabilistic Roadmap, Schritt 2: Entfernung der „verbotenen“ Stichproben.	92
9.9. Probabilistic Roadmap, Schritt 3: Verbinden jeder gültigen Stichprobe mit den $k = 2$ nächsten Nachbar.	93
9.10. Probabilistic Roadmap, Schritt 4: Entfernen der „verbotenen“ Verbindungen.	93

---

9.11. Probabilistic Roadmap, Ergebnis: Der resultierende (nicht-planare) Graph ist die Probabilistic Roadmap (PRM) . . . . .	94
12.1. Aufbau eines hierarchischen Steuerungsparadigma eines autonomen Roboters. . . . .	119
12.2. Aufbau eines reaktiven Steuerungsparadigmas eines autonomen Roboters. . . . .	119



---

# Tabellenverzeichnis

---

7.1. Beispielhafte Werte der Illumination $i$ . . . . .	51
7.2. Beispielhafte Werte der Reflexion $r$ . . . . .	51
9.1. Arten der Bahnplanung, wobei die Verfahren absteigend von globalen und grob granularen zu lokalen und fein granularen Verfahren sortiert sind. . . . .	84
12.1. Basisfunktionalitäten von autonomen Robotern. . . . .	118
C.1. Notationen . . . . .	127



---

## Liste der Algorithmen

---

1. Rapidly Exploring Random Trees . . . . . 95

---

# 1. Einleitung

---

---

## 1.1. Was ist ein Roboter?

---

---

## 1.2. Was ist KI?

---

---

## 1.3. Was ist Robotik?

---

---

## 1.4. Sense – Plan – Act

---

### Act

Kinematik

Dynamik

Steuerung

### Sense

Sensoren

### Plan

Lokalisierung, Kartographie, Navigation, Bahnplanung

---

## **1.5. Geschichte der Robotik**

---

### **1.5.1. Historische Entwicklung**

---

### **1.5.2. Die drei Gebote der Robotik**

---

### **1.5.3. Autonome Fahrzeuge**

---

### **1.5.4. Entwicklungstrend**

---

## **1.6. Herausforderungen**

---

### **1.6.1. Humanoide Bewegung**

---

### **1.6.2. Roboter für menschliche Mobilität**

---

### **1.6.3. Roboter-Avatare**

---

**Beine**

---

**Katastrophenbewältigung und -hilfe**

---

**Objekt-Vorlagen**

---

**Greifen und Manipulation**

---

### **1.6.4. Die Robotik an sich**

---

---

## 2. Räumliche Darstellungen und Transformationen

---

---

### 2.1. Mathematische Grundlagen und Notation

---

---

#### 2.1.1. Vektoren

---

- Vektor:  $\mathbf{p}$  (fett)
- Nullvektor:  $\mathbf{0}$
- Transponierter Vektor:  $\mathbf{p}^T$
- Euklidische Norm eines Vektor:  $\|\mathbf{p}\|$
- Standard-Skalarprodukt:  $\mathbf{p} \circ \mathbf{r}$
- Orthogonale Vektoren:  $\mathbf{p} \perp \mathbf{r}$
- Winkel zwischen  $\mathbf{p}, \mathbf{r}$ :  $\phi$
- Kreuzprodukt:  $\mathbf{p} \times \mathbf{r}$
- Euklidische Einheitsvektoren:  $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$
- Positionsvektor:  $\mathbf{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$

---

#### 2.1.2. Matrizen

---

- Matrix:  $\mathbf{R}$  (fett)
- Transponierte Matrix:  $\mathbf{R}^T$
- Einheitsmatrix:  $\mathbf{E}$
- Nullmatrix:  $\mathbf{O}$
- Inverse Matrix:  $\mathbf{R}^{-1}$

---

### 2.2. Klassische Transformationsbeziehungen

---

Darstellung von  $\cdot$  bzgl. einem Koordinatensystem  $S_a$ :  $^a\cdot$ .  
Klassische Transformationsbeziehung:

$$^a\mathbf{p} = ^a\mathbf{r}_b + ^a\mathbf{R}_b\ ^b\mathbf{p}$$

mit

- ${}^a p$ : Koordinaten von Punkt  $P$  bzgl.  $S_a$ ,
- ${}^b p$ : Koordinaten von Punkt  $P$  bzgl.  $S_b$ ,
- ${}^a r_b$ : Translationsvektor,
- ${}^a R_b$ : Rotationsmatrix zwischen  $S_a$  und  $S_b$ .

## 2.3. Rotation eines Koordinatensystems

### 2.3.1. Rotationsmatrizen

Eigenschaften einer Rotationsmatrix  $R$ :

- Orthonormalität:  $RR^T = E$
- Orthonormale Spalten (d. h. Einheitsvektoren und paarweise orthogonal).

#### Elementare Rotationsmatrizen

- Rotation um  $x$ -Achse um  $\theta_x$ :

$$R(x; \theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{bmatrix}$$

- Rotation um  $y$ -Achse um  $\theta_y$ :

$$R(y; \theta_y) = \begin{bmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{bmatrix}$$

- Rotation um  $z$ -Achse um  $\theta_z$ :

$$R(z; \theta_z) = \begin{bmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

### 2.3.2. Verkettete Rotationen

Im Allgemeinen können mehrere Rotationen direkt hintereinander ausgeführt werden, wobei dafür die Rotationsmatrizen multipliziert werden. Dafür gibt es zwei mögliche Interpretationsmöglichkeiten:

$${}^a R_b = \underbrace{R_n \cdot R_{n-1} \cdots R_2 \cdot R_1}_{\text{Nachmultiplikation}} \xrightarrow{\text{Vormultiplikation}}$$

- *Vormultiplikation*  
Die Rotationen finden in der Reihenfolge  $1, 2, \dots, n$  statt und drehen das (momentane)  $S_b$ -System um die festen  $S_a$ -Achsen, das Bezugssystem ist immer  $S_a$ .
- *Nachmultiplikation*  
Die Rotationen finden in der Reihenfolge  $n, n-1, \dots, 2, 1$  statt und drehen das (momentane)  $S_b$ -System um die momentanen  $S_b$ -Achsen, das Bezugssystem ist immer das momentane  $S_b$ .

Prinzipiell liefern beide Interpretationen jedoch die gleiche Endmatrix (da Matrixmultiplikation assoziativ ist).

---

### 2.3.3. Winkelparameter

---

Eine allgemeine Rotation um alle Achsen benötigt nur 3 Parameter (es existieren nur drei Freiheitsgrade). Dabei können die verketteten Rotationen in sechs verschiedenen Reihenfolgen um drei bzw. zwei unterschiedliche Achsen, also insgesamt auf zwölf Arten geschehen. Geläufige Winkelkonventionen sind z. B. RPY- oder Euler-Winkel.

---

#### Kardan-Winkel

---

Bei *Kardan-Winkeln* erfolgen die drei verketteten Rotationen um die Winkel  $\psi$  (Yaw),  $\theta$  (Pitch),  $\phi$  (Roll) in dieser Reihenfolge um die Achsen des festen Bezugssystems  $S_a$ , d. h. durch Vormultiplikation.

Am häufigsten werden *X-Y-Z-Winkel* (auch *RPY-Winkel*) verwendet:

$$\begin{aligned} {}^a\mathbf{R}_b(\psi, \theta, \phi) &= \mathbf{R}(z; \phi) \cdot (\mathbf{R}(y; \theta) \cdot \mathbf{R}(x; \psi)) \\ &= \begin{bmatrix} \cos \phi \cos \theta & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \sin \phi \cos \theta & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi \\ -\sin \theta & \cos \theta \sin \psi & \cos \theta \cos \psi \end{bmatrix} \end{aligned}$$

Die inverse Umwandlung (von einer Rotationsmatrix zu RPY-Winkeln) kann dann wie folgt erfolgen:

1. Berechnung von  $\theta$  aus  $R_{31}$ .
2. Berechnung von  $\psi$  aus  $R_{23}$  oder  $R_{33}$ .
3. Berechnung von  $\phi$  aus einem der anderen Matrixelemente.

Diese Umwandlung ist nicht immer eindeutig!

---

#### Euler-Winkel

---

Bei *Euler-Winkeln* erfolgen die drei verketteten Rotationen um die Winkel  $\alpha$ ,  $\beta$ ,  $\gamma$  in dieser Reihenfolge um die Achsen des momentanen Bezugssystems  $S_b$ , d. h. durch Nachmultiplikation.

Am häufigsten werden dabei *Z-Y-Z-Winkel* verwendet:

$${}^a\mathbf{R}_b = (\mathbf{R}(z; \alpha) \cdot \mathbf{R}(y; \beta)) \cdot \mathbf{R}(z; \gamma)$$

Andere Möglichkeiten sind z. B. *X-Y-Z-Winkel*

$${}^a\mathbf{R}_b = (\mathbf{R}(z; \alpha) \cdot \mathbf{R}(y; \beta)) \cdot \mathbf{R}(x; \gamma)$$

---

## 2.4. Homogene Transformationen

---

Die klassische Transformationsbeziehung

$${}^a\mathbf{p} = {}^a\mathbf{r}_b + {}^a\mathbf{R}_b {}^b\mathbf{p}$$

lässt sich durch *homogene Transformationen* auch durch eine einzigen Matrixmultiplikation darstellen:

$${}^a\hat{\mathbf{p}} = \begin{bmatrix} {}^a\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^a\mathbf{R}_b & {}^a\mathbf{r}_b \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^b\mathbf{p} \\ 1 \end{bmatrix} = {}^a\mathbf{T}_b \cdot {}^b\hat{\mathbf{p}}$$

Somit lautet die inverse einer homogenen Transformationsmatrix:

$$({}^a\mathbf{T}_b)^{-1} = \begin{bmatrix} ({}^a\mathbf{R}_b)^T & -({}^a\mathbf{R}_b)^T \cdot {}^a\mathbf{r}_b \\ \mathbf{0}^T & 1 \end{bmatrix}$$

---

## 3. Roboterkinematik

---

---

### 3.1. Vorwärtskinematik

---

Die Vorwärtskinematik DIR KIN berechnet die Endeffektor-Position/-Orientierung aus den Gelenkstellungen.

---

#### 3.1.1. Kinematische Ketten

---

Der Roboter wird dabei als *offene kinematische Kette* betrachtet, d. h. die

- Schub- (mit Ausfahrweite  $d_i$ ) und
- Drehgelenke (mit Rotation  $\theta_i$ )

als auch die Glieder werden als starr-, massen- sowie schleifenlos betrachtet.

Dabei hat jedes Gelenk einen Freiheitsgrad (Degree of Freedom, DOF). Ein Manipulator mit  $n$  Gelenken hat somit  $n$  DOF, wobei

1. die Gelenke von 1 bis  $n$  nummeriert werden und
2. das Gelenk  $i$  die Glieder  $i - 1$  und  $i$  verbindet.

Es gibt also  $n + 1$  Glieder wobei Glied 0 die Basis ist und Glied  $n$  den Endeffektor trägt.

---

#### 3.1.2. Kinematische Modellbildung

---

Mit jedem Glied  $i$  wird am Gelenk  $i + 1$  ein gliedfestes Koordinatensystem  $S_i$  befestigt, wobei zusätzlich das Basiskoordinatensystem  $S_0$  und das Endeffektorkoordinatensystem  $S_n$  definiert wird. Ein Manipulator wird also durch  $n + 1$  Koordinatensysteme beschrieben.

Ziel ist es nun, eine kinematische Modellstruktur

$${}^0T_n = {}^0T_1 \cdot {}^1T_2 \cdots {}^{n-2}T_{n-1} \cdot {}^{n-1}T_n = \prod_{i=1}^n {}^{i-1}T_i$$

wobei die homogene Transformationsmatrix  ${}^0T_n = {}^0T_n(\mathbf{q}, \mathbf{f})$  eine Funktion der verallgemeinerten Gelenkvariablen  $\mathbf{q}$  (mit  $q_i = \theta_i$  für ein Drehgelenk und  $q_i = d_i$  für ein Schubgelenk) und der geometrischen Parameter  $\mathbf{f}$  ist.

---

#### 3.1.3. Denavit-Hartenberg (DH) Konventionen

---

Die Denavit-Hartenberg Konventionen (DH-Konventionen) ist ein internationaler, formaler Standard zur Festlegung der Koordinatensysteme. Einige der Basiseigenschaften von per DH-Konvention festgelegten Koordinatensystemen sind:



- Die Koordinatensysteme liegen in den jeweiligen Bewegungsachsen.
- Die  $z_{i-1}$ -Achse liegt entlang der Bewegungsachse des  $i$ -ten Gelenks.
- Die  $x_i$ -Achse steht senkrecht zur  $z_{i-1}$ -Achse und zeigt von ihr weg.
- Die  $x_i$ -Achse und die  $z_{i-1}$ -Achse haben einen Schnittpunkt.

---

### Pseudo-Algorithmus zur Festlegung der Koordinatensysteme

---

Voraussetzung: Der Manipulator befindet sich in allen  $n$  Gelenken in Nullstellung (z. B. ausgestreckt nach oben).

Legt der folgende Algorithmus in einzelnen Schritten nicht eindeutig sein, so ist zuerst die Einhaltung der DH-Eigenschaften zu gewährleisten und ansonsten die Wahl einer Lösung mit möglichst geringer Komplexität der Transformationen zu bevorzugen.

#### Schritt 1

**Schritt 1a** Nummerierung der Glieder von 0 (Basis) bis  $n$ .

**Schritt 1b (für  $i = 0, \dots, n-1$ )** Festlegung der  $z_i$ -Achsen als koinzident mit der Bewegungsachse des  $(i+1)$ -ten Gelenks:

- Bei Schubgelenken in Richtung weg von Gelenk  $i+1$ .
- Bei Drehgelenken als Rotationsachse in Richtung positiver Drehwinkel (wird festgelegt).

**Schritt 2** Festlegung von  $S_0$  mit Ursprung auf der  $z_0$ -Achse, sodass sich ein Rechtskoordinatensystem ergibt (oft werden  $x_0$  und  $y_0$  parallel zum Welt-Koordinatensystem gewählt).

**Schritt 3 (für  $i = 1, \dots, n-1$ )** Festlegung des Ursprungs von  $S_i$ :

- Falls  $z_{i-1}$  und  $z_i$  sich schneiden:  
Schnittpunkt wird der Ursprung.
- Falls  $z_{i-1}$  und  $z_i$  parallel sind:  
Festlegung auf der  $z_i$ -Achse am Gelenk  $i+1$ .
- Sonst ( $z_{i-1}$  und  $z_i$  windschief):  
Bilde eine gemeinsame Normale zu  $z_i$  und  $z_{i-1}$ , Schnittpunkt der Normalen mit  $z_i$  wird der Ursprung.

#### Kurzschreibweise

$$S_i = \begin{cases} z_{i-1} \cap z_i & \text{falls } |z_{i-1} \cap z_i| = 1 \\ z_i \cap \text{Gelenk}_{i+1} & \text{falls } z_{i-1} \parallel z_i \\ z_i \cap (\perp(z_i, z_{i-1})) & \text{sonst} \end{cases}$$

#### Schritt 4 (für $i = 1, \dots, n-1$ ) Festlegung der $x_i$ -Achse:

- Falls  $z_{i-1}$  und  $z_i$  sich schneiden:  
 $x_i = z_{i-1} \times z_i$  oder  $x_i = z_i \times z_{i-1}$  (unter Berücksichtigung der DH-Eigenschaften!).
- Sonst ( $z_{i-1}$  und  $z_i$  parallel oder windschief):  
 $x_i$ -Achse in Richtung der gemeinsamen Normalen von  $z_{i-1}$  und  $z_i$ , sodass sich  $x_i$  und  $z_{i-1}$  schneiden.

##### Kurzschreibweise

$$x_i = \begin{cases} x_i \in \{ z_{i-1} \times z_i, z_i \times z_{i-1} \} & \text{falls } |z_{i-1} \cap z_i| = 1 \\ \perp(z_{i-1}, z_i) & \text{sonst} \end{cases}$$

#### Schritt 5 (für $i = 1, \dots, n-1$ ) Festlegung der $y_i$ -Achse, sodass $x_i, y_i, z_i$ ein Rechtskoordinatensystem bilden (d.h. $z_i = x_i \times y_i$ , Rechte-Hand-Regel!).

##### Kurzschreibweise

$$z_i \stackrel{!}{=} x_i \times y_i$$

#### Schritt 6 Festlegung des Endeffektor-Koordinatensystems $S_n$ : Der Ursprung wird meistens in den Tool-Center-Point (TCP) gelegt.

- Liegen keine besonderen Bedingungen vor:  
Erstellung einer möglichst einfachen Transformation (häufig eine reine Translation).
- Ist der Endeffektor ein einfacher Greifer:  
Meist eine solche Festlegung, dass Yaw-, Pitch- und Roll-Winkel verwendet werden können.

#### Schritt 7 (für $i = 1, \dots, n$ ) Erstellen einer Tabelle von Gliedparametern $\theta_i, d_i, a_i, \alpha_i$ :

$\theta_i$  Winkel zwischen  $x_{i-1}$  und  $x_i$ , gemessen um  $z_{i-1}$ . Variabel, falls  $i$  ein Drehgelenk ist.

$d_i$  Entfernung vom  $S_{i-1}$ -Ursprung entlang  $z_{i-1}$  zum Schnittpunkt mit  $x_i$ . Variablen, falls  $i$  ein Schubgelenk ist.

$a_i$  Entfernung vom Schnittpunkt von  $z_{i-1}$  und  $x_i$  entlang  $x_i$  zum  $S_i$ -Ursprung. Je nach Orientierung von  $x_i$  kann  $a_i$  auch negativ sein.

$\alpha_i$  Winkel zwischen  $z_{i-1}$  und  $z_i$ , gemessen um  $x_i$ .

##### Kurzschreibweise

$$\begin{aligned} \theta_i &:= x_{i-1} \angle_{z_{i-1}} x_i \\ d_i &:= |S_{i-1} \rightarrow_{z_{i-1}} (z_{i-1} \cap x_i)| \\ a_i &:= |(z_{i-1} \cap x_i) \rightarrow_{x_i} S_i| \\ \alpha_i &:= z_{i-1} \angle_{x_i} z_i \end{aligned}$$

**Schritt 8** (für  $i = 1, \dots, n$ ) Bildung der homogenen Transformationsmatrizen:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Schritt 9** Berechnung von

$${}^0\mathbf{T}_n = {}^0\mathbf{T}_1 \cdot {}^1\mathbf{T}_2 \cdots {}^{n-2}\mathbf{T}_{n-1} \cdot {}^{n-1}\mathbf{T}_n = \prod_{i=1}^n {}^{i-1}\mathbf{T}_i$$

---

## 3.2. Rückwärtskinematik (Inverse Kinematik)

---

Die Rückwärtskinematik INV KIN berechnet die benötigten Gelenkstellungen für eine Endeffektor-Position/-Orientierung.

Gegeben einer Endeffektor-Position/-Orientierung  ${}^0\tilde{\mathbf{T}}_E$  ist ein Vektor  $\mathbf{q} \in \mathbb{R}^n$  gesucht, sodass

$${}^0\mathbf{T}_E(\mathbf{q}) = {}^0\tilde{\mathbf{T}}_E$$

gilt. Dies liefert ein System von 6 unabhängigen, nichtlinearen, gekoppelten Gleichungen.

Allgemein gilt: Wird eine RAN beliebig vorgegeben, so existiert

- für  $n < 6$  Gelenkvariablen i. A. keine Lösung (es dürfen maximal  $n$  Werte vorgegeben werden),
- für  $n = 6$  Gelenkvariablen i. A. genau eine Lösung (dies ist der Fall für viele 6-DOF Industrieroboter),
- für  $n > 6$  Gelenkvariablen i. A. unendliche viele Lösungen.

Definition: Der *erreichbare Arbeitsraum*  $\mathcal{A} \subseteq \mathbb{R}^3$  ist der Raum, der von dem Manipulator in mindestens einer Stellung erreicht werden kann.

Definition: Der *vollmanipulierbare Arbeitsraum* die ist Teilmenge von  $\mathcal{A}$ , die von dem Manipulator mit allen Orientierungen erreicht werden kann.

---

### 3.2.1. Numerische Berechnung

---

Es kann bspw. das klassische Newton-Verfahren zur Lösung nichtlinearer Gleichungssysteme eingesetzt werden (sofern DIR KIN differenzierbar ist).

**Nachteile:**

- Kein iteratives Verfahren kann alle Lösungen garantiert berechnen. Dies wird jedoch oftmals benötigt.
- Iterative Berechnungsverfahren sind wesentlich langsamer als explizite, geschlossene Verfahren. Außerdem sind sie nicht mit den Echtzeitanforderungen vereinbar.
- Das Konvergenzverhalten ist oftmals problematisch.

---

### 3.2.2. Analytische Lösung

---

- **Vorteil:** Es können alle Lösungen in garantierter Berechnungszeit berechnet werden.
- **Nachteil:** Aufwendig zu bestimmen.
- Dennoch sollten, wenn möglich, explizite Lösungsformeln zumindest für einen Teil gefunden werden.

---

### Algebraische Ermittlung

---

Die Rückwärtslösung wird rein durch umformen von Gleichungen gefunden.

---

### Geometrische Lösung

---

Die Rückwärtslösung wird durch geometrische (bspw. trigonometrische) Betrachtung des Roboters ermittelt, wobei das Problem in mehrere Teilprobleme zerlegt werden kann.

---

### Algorithmische Ermittlung

---

- **Vorteil:** Falls eine Lösungsformel bestimmt werden kann schneller als iterative Verfahren.
- **Nachteil:** Eine algorithmische Lösung garantiert nicht das finden einer Lösung, auch wenn eine solche existiert.

---

## 3.3. Genauigkeit des kinematischen Modells

---

Es wird zwischen mehreren, unterschiedlichen „Genauigkeiten“ unterschieden:

- *Positionsgenauigkeit*  
Die Differenz zwischen tatsächlicher und vorgegebener RAN.
- *Wiederholungsgenauigkeit*  
Die Variation der Positionsgenauigkeit bei mehrfacher Wiederholung.
- *Auflösungsgenauigkeit*  
Die kleinstmögliche Distanz, über die der TCP garantiert bewegt werden kann.

Für die meisten Roboter sind Wiederholungsgenauigkeit und Auflösungsgenauigkeit viel besser als die Positionsgenauigkeit (z. B. durch Unsicherheiten in den geometrischen Parametern und die Nichtberücksichtigung von Reibung, Spiel, temperaturabhängigen Ausdehnungen, ...).

---

## 4. Geschwindigkeit, Jacobi-Matrix und statische Kräfte

---

Bisher wurde ausschließlich die Positionen von Gelenken, Gliedern und dem Endeffektor untersucht. In diesem Kapitel werden die Geschwindigkeiten und Beschleunigungen untersucht.

---

### 4.1. Vektor der Winkelgeschwindigkeiten

---

Sei  $R(q(t))$  eine Rotationsmatrix in impliziter Abhängigkeit von der Zeit. Dann gilt

$$\frac{d}{dt}(R(q(t))) = B(\omega(t)) \cdot R(q(t))$$

mit der schiefsymmetrischen Matrix

$$B(\omega) := \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

wobei  $\omega = [\omega_x \ \omega_y \ \omega_z]^T$  den Vektor der Winkelgeschwindigkeiten der Rotation darstellt.

Über die Beziehung

$$B(\omega(q(t), \dot{q}(t))) = \left( \sum_{i=1}^n \frac{\partial R(q(t))}{\partial q_i(t)} \cdot \dot{q}_i(t) \right) \cdot R(q(t))^T$$

kann damit die Winkelgeschwindigkeit der Rotation des Endeffektor bestimmt werden (durch Ablesen der Elemente von  $B$ ).

---

### 4.2. Jacobi-Matrix eines Manipulators

---

**Achtung:** Die Jacobi-Matrix eines Manipulators darf nicht mit der Jacobi-Matrix einer Funktion mehrerer Veränderlicher verwechselt werden!

---

#### 4.2.1. Verkettung von Rotationen

---

Für verkettete Rotationen

$${}^0R_n = {}^0R_1 \cdot {}^1R_2 \cdots {}^{n-2}R_{n-1} \cdot {}^{n-1}R_n$$

gilt

$$\frac{d}{dt}({}^0R_n) = B({}^0\omega_n) \cdot {}^0R_n$$

mit dem Winkelgeschwindigkeitsvektor

$${}^0\omega_n = {}^0\omega_1 + {}^0R_1 \cdot {}^1\omega_2 + \cdots + {}^0R_{n-1} \cdot {}^{n-1}\omega_n$$

---

### 4.2.2. Zusammenfassung

---

Das folgende gilt nur, wenn die Koordinatensysteme entsprechend der DH-Konvention platziert sind!

Für die Jacobi-Matrix  ${}^0J$  gilt:

$$\begin{bmatrix} {}^0v(t) \\ {}^0\omega(t) \end{bmatrix} = {}^0J_n(q(t)) \cdot \dot{q}(t)$$

mit der Lineargeschwindigkeit  ${}^0v(t)$  und Winkelgeschwindigkeit  ${}^0\omega(t)$  des Endeffektors in Bezug auf das Basiskoordinatensystem.

Die  $i$ -te Spalte  ${}^0J_{n,i}$  der Manipulator-Jacobi-Matrix berechnet sich wie folgt. Ist Gelenk  $i$  ein

- Drehgelenk:

$${}^0J_{n,i} = \begin{bmatrix} {}^0e_{z_{i-1}} \times ({}^0r_n - {}^0r_{i-1}) \\ {}^0e_{z_{i-1}} \end{bmatrix}$$

- Schubgelenk:

$${}^0J_{n,i} = \begin{bmatrix} {}^0e_{z_{i-1}} \\ \mathbf{0} \end{bmatrix}$$

Somit kann die Jacobi-Matrix direkt aus DIR KIN berechnet werden.

Anmerkung: Oft kann es (insbesondere bei der linearen Geschwindigkeit eines Drehgelenks) einfacher sein,  ${}^0r_n$  direkt nach den Gelenkvariablen abzuleiten.

---

### 4.3. Inverses Jacobi-Modell

---

Das inverse Jacobi-Modell (INV KIN) berechnet die benötigten Gelenkgeschwindigkeiten für eine Endeffektor-Geschwindigkeit.

Gegeben einer Linear- und Winkelgeschwindigkeit  ${}^0v_n, {}^0\omega_n$  ist ein Vektor  $\dot{q}$  gesucht, sodass

$$\begin{bmatrix} {}^0v_n \\ {}^0\omega_n \end{bmatrix} = {}^0J_n \cdot \dot{q}$$

gilt. Dies liefert ein System von 6 unabhängigen, nichtlinearen, gekoppelten Gleichungen.

Allgemein gilt: Werden Geschwindigkeiten  ${}^0v_n, {}^0\omega_n$  beliebig vorgegeben, so existiert

- für  $n < 6$  Gelenkvariablen i. A. keine Lösung (es dürfen maximal  $n$  Werte vorgegeben werden),
- für  $n = 6$  Gelenkvariablen i. A. genau eine Lösung (dies ist der Fall für viele 6-DOF Industrieroboter),
- für  $n > 6$  Gelenkvariablen i. A. unendliche viele Lösungen.

Für solche „redundanten“ Manipulatoren können die üblichen Freiheitsgrade z. B. durch Hinzunahme von kinematischen Zwangsbedingungen oder durch die Maximierung eines Gütekriteriums, z. B. des Manipulierbarkeitsmaßes nach Yoshikawa

$$\mu(q) = \sqrt{\det({}^0J_n \cdot {}^0J_n^T)}$$

festgelegt werden.

---

## 4.4. Kinematische Singularitäten

---

Sei

$$m := \max_{q \in \mathcal{A}} \text{rank}({}^0J_n(q))$$

der maximale Rang der Jacobi-Matrix. Dann heißt jede Konfiguration  $q_s$  mit

$$\text{rank}({}^0J_n(q_s)) < m$$

*kinematische Singularität.*

Bei einer solchen Singularität ist die Jacobi-Matrix singulär und es müssten unendliche hohe Geschwindigkeiten erreicht werden (ebenso nehmen die Geschwindigkeit bei der Annäherung an eine solche Singularität i. A. zu).

Zur Berechnung der kinematischen Singularitäten (z. B. um diese bei der Bahnplanung zu vermeiden), ist folgendes vorgehen möglich: Zunächst wird die Jacobi-Matrix auf eine  $(m \times m)$ -Untermatrix  ${}^0\hat{J}_n(q)$  mit  $m = \text{rank}({}^0\hat{J}_n(q))$  eingeschränkt. Dann ist eine Konfiguration  $q_s$  genau dann singulär, wenn

$$\det {}^0\hat{J}_n(q_s) = 0$$

gilt, d. h. die singulären Konfigurationen können durch Nullsetzen der Determinante der eingeschränkten Jacobi-Matrix gefunden werden.

---

### 4.4.1. Vermeidung

---

---

### 4.4.2. Umgang mit unvermeidbaren Singularitäten

---

---

## 4.5. Nicht-holonome Kinematik mehrrädriger Fahrzeuge

---

---

### 4.5.1. Differentialantrieb

---

---

### 4.5.2. Allgemeines Vorwärtskinematikproblem für Fahrzeuge

---

---

### 4.5.3. Inverses Kinematikproblem

---

---

### 4.5.4. Omnidirektionale Dreirad-Kinematik

---

---

### 4.5.5. Weitere Antriebsarten von Fahrzeugen

---

---

## 4.6. Statische Kräfte bei Manipulatoren

---

## 5. Roboterdynamik

Bisher wurden keine (statischen) Kräfte, sondern nur Positionen und Geschwindigkeiten betrachtet. Die inverse Dynamik/Kinetik INV DYN berechnet für gegebene Kräfte (z. B. die Gravitation) die Kräfte/Momente, die die Gelenke aufbringen müssen.

Seien  $\tau_i$  die verallgemeinerten Kräfte/Drehmomente wobei

- $\tau_i = n_i$  für ein Drehgelenk (Drehmoment) und
- $\tau_i = f_i$  für ein Schubgelenk (Kraft) gilt.

### 5.1. Massenverteilung eines Starrkörpers

Die Massenverteilung eines Starrkörpers wird durch den symmetrischen Trägheitstensor

$${}^a\mathbf{I} \in \mathbb{R}^{3 \times 3} = \begin{bmatrix} {}^aI_{xx} & -{}^aI_{xy} & -{}^aI_{xz} \\ -{}^aI_{xy} & {}^aI_{yy} & -{}^aI_{yz} \\ -{}^aI_{xz} & -{}^aI_{yz} & {}^aI_{zz} \end{bmatrix}$$

bezüglich eines körperfesten Koordinatensystems  $S_a$  beschrieben. Die Elemente der Hauptdiagonale heißen dabei *Massenträgheitsmomente* und die restlichen *Massenträgheitsprodukte*, die sich wie folgt berechnen (mit dem Volumen  $\mathcal{V}$ , einem differentiellen Volumenelement  $dv$  und der Materialdichte  $\rho(x, y, z)$ ):

$$\begin{aligned} {}^aI_{xx} &= \iiint_{\mathcal{V}} (y^2 + z^2) \rho(x, y, z) dv & {}^aI_{xy} &= \iiint_{\mathcal{V}} xy \rho(x, y, z) dv \\ {}^aI_{yy} &= \iiint_{\mathcal{V}} (x^2 + z^2) \rho(x, y, z) dv & {}^aI_{xz} &= \iiint_{\mathcal{V}} xz \rho(x, y, z) dv \\ {}^aI_{zz} &= \iiint_{\mathcal{V}} (x^2 + y^2) \rho(x, y, z) dv & {}^aI_{yz} &= \iiint_{\mathcal{V}} yz \rho(x, y, z) dv \end{aligned}$$

Sind  $x_a$ ,  $y_a$  und  $z_a$  die Hauptträgheitsachsen, so verschwinden die die Massenträgheitsprodukte. Dies ist zum Beispiel der Fall, wenn die Achsen paarweise Symmetrieebenen des Körpers bilden (bilden zwei Achsen eine Symmetrieebene, so verschwinden die Trägheitsprodukte der dritten Achse).

#### 5.1.1. Transformation von Trägheitstensoren

**(Reine) Rotation des Bezugssystems** Ist das System  $S_b$  um  ${}^a\mathbf{R}_b$  zu  $S_a$  rotiert, so gilt:

$${}^b\mathbf{I} = {}^b\mathbf{R}_a \cdot {}^a\mathbf{I} \cdot {}^a\mathbf{R}_b$$

**Translation des Bezugssystems (Satz von Steiner)** Liegt  $S_c$  im *Schwerpunkt* des Körpers und das System  $S_b$  ist um  ${}^c\mathbf{r}_b$  zu  $S_c$  verschoben, aber nicht rotiert, so gilt:

$${}^b\mathbf{I} = {}^c\mathbf{I} + m \cdot ({}^c\mathbf{r}_b^T \cdot {}^c\mathbf{r}_b \cdot \mathbf{E} - {}^c\mathbf{r}_b \cdot {}^c\mathbf{r}_b^T)$$



**Zusammenfügen von Körpern** Werden die Trägheitstensoren  $I_1$  und  $I_2$  von zwei Körpern, die verbunden werden, im gleichen Koordinatensystem dargestellt, so ergibt sich der Gesamtträgheitstensor des zusammengeführten Körpers:

$$I = I_1 + I_2$$

## 5.2. Newton-Euler Formulierung der Roboterdynamik

Seien für jedes Roboterglied  $i$  die folgenden Daten gegeben:

- Der Koordinatenvektor des Schwerpunkts
  - bezüglich  $S_i$  als  ${}^i r_{c_i}$ ,
  - bezüglich  $S_0$  als  ${}^0 r_{c_i}$  und
  - bezüglich  $S_{i-1}$  als  ${}^{i-1} r_{c_i}$ .
- Die Gesamtmasse  $m_i$ .
- Der Trägheitstensor  ${}^{c_i} I_i$  bezüglich  $S_{c_i}$ .

Dabei sei  $S_{c_i}$  das Schwerpunktkoordinatensystem von Glied  $i$  mit der gleichen Orientierung wie  $S_i$ .

Weitere Notationen:

- $\omega_i := {}^i R_0 \cdot {}^0 \omega_i$   
Winkelgeschwindigkeit des  $i$ -ten Gelenks, gemessen in  $S_0$  und dargestellt in  $S_i$ .
- $\dot{\omega}_i := {}^i R_0 \cdot {}^0 \dot{\omega}_i$   
Winkelbeschleunigung des  $i$ -ten Gelenks, gemessen in  $S_0$  und dargestellt in  $S_i$ .
- $v_i := {}^i R_0 \cdot {}^0 v_i$   
Linearer Geschwindigkeitsvektor des  $i$ -ten Glieds an Gelenk  $i+1$ , gemessen in  $S_0$  und dargestellt in  $S_i$ .
- $\dot{v}_i := {}^i R_0 \cdot {}^0 \dot{v}_i$   
Linearer Beschleunigungsvektor des  $i$ -ten Glieds an Gelenk  $i+1$ , gemessen in  $S_0$  und dargestellt in  $S_i$ .

### 5.2.1. Iterative Berechnung von INV DYN

Für die Basis werden die Werte  $\omega_0$ ,  $\dot{\omega}_0$ ,  $v_0$  und  $\dot{v}_0$  vorgegeben (z. B.  $\dot{v}_0 = g$  mit dem Gravitationsvektor  $g$ ). Des weiteren werden die Werte  $f_{n+1}$  und  $n_{n+1}$  für den Endeffektor vorgegeben (eine am Endeffektor sorgt dabei nicht für die Änderung dieser Werte, sondern für eine Änderung der Masse  $m_n$ ).

Sei außerdem

$$\rho_i := \begin{cases} 1 & \text{falls } i \text{ Drehgelenk} \\ 0 & \text{falls } i \text{ Schubgelenk} \end{cases}$$

**Schritt NE.1 (für  $n = 1, \dots, n$ )** Berechnung der Linear- und Winkelbeschleunigungen (von innen nach außen).

$$\boldsymbol{\omega}_i = {}^i\mathbf{R}_{i-1} \left( \boldsymbol{\omega}_{i-1} + \rho_i \left[ {}^{i-1}\mathbf{e}_{z_{i-1}} \dot{q}_i \right] \right)$$

$$\dot{\boldsymbol{\omega}}_i = {}^i\mathbf{R}_{i-1} \left( \dot{\boldsymbol{\omega}}_{i-1} + \rho_i \left[ {}^{i-1}\mathbf{e}_{z_{i-1}} \ddot{q}_i + \boldsymbol{\omega}_{i-1} \times \left( {}^{i-1}\mathbf{e}_{z_{i-1}} \dot{q}_i \right) \right] \right)$$

$$\mathbf{v}_i = {}^i\mathbf{R}_{i-1} \mathbf{v}_{i-1} + \boldsymbol{\omega}_i \times \left( {}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{r}_i \right) + (1 - \rho_i) \left[ {}^i\mathbf{e}_{z_{i-1}} \dot{q}_i \right]$$

$$\dot{\mathbf{v}}_i = {}^i\mathbf{R}_{i-1} \dot{\mathbf{v}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \left( {}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{r}_i \right) + \boldsymbol{\omega}_i \times \left( \boldsymbol{\omega}_i \times \left( {}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{r}_i \right) \right) + (1 - \rho_i) \left[ 2\boldsymbol{\omega}_i \times \left( {}^i\mathbf{e}_{z_{i-1}} \dot{q}_i \right) + {}^i\mathbf{e}_{z_{i-1}} \ddot{q}_i \right]$$

$$\dot{\mathbf{v}}_{c_i} = \dot{\mathbf{v}}_i + (\dot{\boldsymbol{\omega}}_i \times {}^i\mathbf{r}_{c_i}) + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times {}^i\mathbf{r}_{c_i})$$

**Schritt NE.2 (für  $i = 1, \dots, n$ )** Berechnung der am Schwerpunkt wirkenden Kräfte  $\mathbf{F}_i$  und Drehmomente  $\mathbf{N}_i$ . Dieser Schritt kann auch in Schritt NE.1 integriert werden.

$$\mathbf{F}_i = \begin{cases} m_i \dot{\mathbf{v}}_{c_i} - m_i \mathbf{g}_i & \text{falls der Einfluss der Gravitation im Schritt NE.2 berücksichtigt wird} \\ m_i \dot{\mathbf{v}}_{c_i} & \text{falls der Einfluss der Gravitation im Anfangswert } \dot{\mathbf{v}}_0 = -\mathbf{g} \text{ berücksichtigt wird} \end{cases}$$

$$\mathbf{N}_i = {}^{c_i}\mathbf{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times ({}^{c_i}\mathbf{I}_i \boldsymbol{\omega}_i)$$

**Schritt NE.3 (für  $i = n, \dots, 1$ )** Berechnung der Kräfte und Drehmomente  $\tau_i$  am Gelenk jedes Roboterlinks (von außen nach innen).

$$\mathbf{f}_i = {}^i\mathbf{R}_{i+1} \mathbf{f}_{i+1} + \mathbf{F}_i$$

$$\mathbf{n}_i = {}^i\mathbf{R}_{i+1} \mathbf{n}_{i+1} + ({}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{r}_i + {}^i\mathbf{r}_{c_i}) \times \mathbf{F}_i + ({}^i\mathbf{R}_{i-1} {}^{i-1}\mathbf{r}_i) \times ({}^i\mathbf{R}_{i+1} \mathbf{f}_{i+1}) + \mathbf{N}_i$$

$$\tau_i = \begin{cases} ({}^i\mathbf{e}_{z_{i-1}})^T \mathbf{n}_i & \text{falls } i \text{ Drehgelenk} \\ ({}^i\mathbf{e}_{z_{i-1}})^T \mathbf{f}_i & \text{falls } i \text{ Schubgelenk} \end{cases}$$

### 5.2.2. Bemerkungen und DIR DYN

Die inverse Dynamik INV DYN kann immer in der Form

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

mit der symmetrischen, positiv definiten Massenmatrix  $\mathbf{q} \in \mathbb{R}^{n \times n}$ , dem Vektor der Zentrifugal- und Coriolis-anteile  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  und dem Vektor der Gravitationsanteile  $\mathbf{G}(\mathbf{q})$ .

Die direkte Dynamik DIR DYN kann in Abhängigkeit von den Antriebsmomenten und -kräften  $\boldsymbol{\tau}$  formuliert werden

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1} (\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}))$$

wobei die inverse der Massenmatrix in der Praxis nie direkt ausgerechnet, sondern stattdessen das lineare Gleichungssystem  $\mathbf{M} \ddot{\mathbf{q}} = \dots$  gelöst wird.

## 5.3. Lagrangesche Formulierung der Roboterdynamik

Die Lagrangesche Formulierung der Dynamik liefert am Ende die gleichen Bewegungsgleichungen, funktioniert aber grundlegend anders (Energie-basiert).

**Kinetische Energie** Die kinetische Energie des  $i$ -ten Gliedes lautet:

$$K_i(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \dot{\mathbf{q}}^T \left( m_i {}^0\mathbf{J}_{c_i,v}^T {}^0\mathbf{J}_{c_i,v} + {}^0\mathbf{J}_{i,\omega}^T {}^{c_i}\mathbf{I}_i {}^0\mathbf{J}_{i,\omega} \right) \dot{\mathbf{q}}$$

Damit lautet die gesamte kinetische Energie des Manipulators:

$$\begin{aligned} K(\mathbf{q}, \dot{\mathbf{q}}) &= \sum_{i=1}^n K_i(\mathbf{q}, \dot{\mathbf{q}}) \\ &= \frac{1}{2} \dot{\mathbf{q}}^T \left( \sum_{i=1}^n \underbrace{\left( m_i {}^0\mathbf{J}_{c_i,v}^T {}^0\mathbf{J}_{c_i,v} + {}^0\mathbf{J}_{i,\omega}^T {}^{c_i}\mathbf{I}_i {}^0\mathbf{J}_{i,\omega} \right)}_{\mathbf{M}(\mathbf{q})} \right) \dot{\mathbf{q}} \\ &= \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}} \end{aligned}$$

Wobei  $\mathbf{M}(\mathbf{q})$  Massenmatrix darstellt.

**Potentielle Energie** Die potentielle Energie des  $i$ -ten Gliedes lautet

$$P_i = -m_i \mathbf{g}^T {}^0\mathbf{r}_{c_i} + P_{\text{Ref},i}$$

mit dem Gravitationsvektor  $\mathbf{g}$  und einer Konstante  $P_{\text{Ref},i}$ , die im späteren Verlauf der Ableitungen wieder verschwindet.

Damit lautet die gesamte potentielle Energie des Manipulators:

$$P(\mathbf{q}) = \sum_{i=1}^n P_i$$

**Lagrangefunktion und Lagrange-Gleichungen** Die Lagrangefunktion ist definiert als:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = K(\mathbf{q}, \dot{\mathbf{q}}) - P(\mathbf{q})$$

Nach dem Prinzip der kleinsten Wirkung (Hamiltonsches Prinzip) bewegt der Manipulator sich dann entsprechend der  $n$  Bewegungsgleichungen

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i, \quad i = 1, \dots, n$$

welche, werden sie symbolisch ausgerechnet, das inverse Dynamikmodell von der Form

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

liefert.

## 5.4. Numerische Aspekte

Die Newton-Euler-Formulierung hat, wenn sie numerisch implementiert wird, deutlich weniger Rechenoperationen als die Lagrangesche Formulierung. Wird die inverse Dynamik jedoch symbolisch „von Hand“ ausgerechnet, so ist die Lagrangesche Formulierung geeigneter.

---

#### 5.4.1. Modularität

---

#### 5.4.2. Simulation

---

### 5.5. Rekursive Verfahren zur Berechnung der Vorwärtsdynamik

---

#### 5.5.1. Verfahren mit expliziter Berechnung der Massenmatrix

---

Verfahren 1: Berechnung von  $M$  durch wiederholte Auswertung des Newton-Euler-Verfahrens

---

Verfahren 2: Ausnutzen der Symmetrie von  $M$

---

Verfahren 3: Aggregation von Teilmanipulatoren (CRBA)

---

Vergleich der Verfahren

---

#### 5.5.2. Verfahren ohne explizite Berechnung der Massenmatrix

---

#### 5.5.3. Multibody Systems Library MBSlib

---

Beispiele

---

### 5.6. Geschlossene kinematische Ketten

---

### 5.7. Berücksichtigung von Nichtstarrkörpereffekten

---

Bisher wurde, selbst bei der Dynamik, viele Vereinfachungen eingeführt, insbesondere wurden Reibung, Elastizitäten und Loser (Gelenkspiel) nicht berücksichtigt. In diesem Abschnitt sollen diese Effekte (vereinfacht) berücksichtigt werden.

---

#### 5.7.1. Reibung

---

Berühren sich zwei Oberflächen, so sind die typischerweise nicht glatt, sodass auch tangentiale Kräfte wirken (die *Reibung*). In der Regel können die Reibungskräfte und -momente, die in den Gelenken wirken, nicht vernachlässigt werden.

Zur Berücksichtigung von Reibung wird den Bewegungsdifferentialgleichungen INV DYN ein Vektor  $F$  für Reibungs- und Dämpfungskräfte und -momente hinzugefügt:

$$\tau = M(q) \ddot{q} + C(q, \dot{q}) + G(q) + F(q, \dot{q})$$

---

#### Reibungsmodell

---

Ein häufiges, einfaches Reibungsmodell (des  $i$ -ten Gelenks) ist

$$F_i(q, \dot{q}) = \underbrace{\gamma_{C,i} \cdot \text{sign } \dot{q}_i}_{\text{Coulombsche Reibung}} + \underbrace{\gamma_{v,i} \cdot \dot{q}_i}_{\text{Viskose Gleitreibung}}$$

---

mit  $\gamma_{C,i}, \gamma_{v,i} = \text{const.}$

Sobald die Zugkraft  $Z$  einen bestimmten Grenzwert überschreitet, wird die Haftung überwunden und die Oberflächen gleiten übereinander. Diese *Coulombsche Reibung* wird mit  $\gamma_{C,i}$  beschrieben, welches nach Charles Augustine proportional zur Normalkraft  $N_i$  ist (mit dem *Reibungskoeffizient*  $\mu_{R,i}$ ):

$$\gamma_{C,i} = \mu_{R,i} \cdot N_i$$

Der Grenzwert der Haftung  $\gamma_{s,i}$  ist mit dem *Haftungskoeffizienten*  $\mu_{H,i}$  ebenfalls proportional zur Normalkraft:

$$\gamma_{s,i} = \mu_{H,i} \cdot N_i$$

Dabei ist der Haftreibungskoeffizient typischerweise größer als der Reibungskoeffizient, was zu einem ruckartigen Gleiten (Losbrechen) bzw. Festsetzen führt. Das resultierende, klassische Reibungsmodell lautet:

$$F_i(\mathbf{q}, \dot{\mathbf{q}}) = \begin{cases} \gamma_{C,i} \cdot \text{sign } \dot{q}_i + \gamma_{v,i} \cdot \dot{q}_i & \text{falls } \dot{q}_i \neq 0 \\ Z & \text{falls } \dot{q}_i = 0 \text{ und } |Z| \leq \gamma_{s,i} \\ \gamma_{s,i} \cdot \text{sign } Z & \text{sonst} \end{cases}$$

Die *viskose Gleitreibung* wird häufig proportional zur Geschwindigkeit modelliert und berücksichtigt hydromechanische Effekte (z. B. Schmiermittel).

**Verfeinerungen** Das bisherige Modell modelliert einen sprunghaften Übergang zwischen Haft- und Gleitreibung. Dies entspricht jedoch nur annähernd der Realität, wobei Reibung mit zunehmender Geschwindigkeit zunächst abnimmt. Dieser Übergang kann mit der Stribeck-Reibung beschrieben werden (dabei ist  $\dot{q}_{\text{stribeck}}$  die Geschwindigkeit, bei der die minimale Reibung auftritt):

$$F_i(\mathbf{q}, \dot{\mathbf{q}}) = \left( \gamma_{C,i} + (\gamma_{s,i} - \gamma_{C,i}) \exp \left\{ - \left| \frac{\dot{q}_i}{\dot{q}_{\text{stribeck},i}} \right| \right\} \right) \cdot \text{sign } \dot{q}_i + \gamma_{v,i} \cdot \dot{q}_i$$

## Anmerkungen

- Reibungsmodelle sind immer mit Unsicherheiten behaftet (Last-, Temperatur-, Winkelabhängigkeiten, ...).
- Es gibt zahlreiche unterschiedliche Ansätze zur Modellierung von Reibung.
- Im Newton-Euler-Verfahren werden Reibungskräfte im letzten Schritt von NE.3 eingeführt:

$$\tau_{i,\text{neu}} = \tau_i + F_i(\mathbf{q}, \dot{\mathbf{q}})$$

## 5.7.2. Elastizität

In der Industrie und der klassischen Robotik werden möglichst starre kinematische Ketten zur genauen und schnellen Manipulation schwerer Objekte konstruiert. In Leichtbaurobotern oder in der Mensch-Roboter-Interaktion können elastische Gelenke und Glieder jedoch von Vorteil sein. Zur exakten Positionsregelung ist hier die Betrachtung der Elastizitäten nötig.

---

## Grundlagen

---

Ein elastischer Körper verformt sich dabei und Krafteinwirkung, speichert diese Kraft als potentielle Energie und gibt die bei Rückfederung wieder frei. Zur Charakterisierung wird die sogenannte *Federsteifigkeit* eingeführt (lineare und rotatorisch):

$$c = \frac{\Delta f}{\Delta s} \qquad c_\varphi = \frac{\Delta n}{\Delta \varphi}$$

Dabei beschreibt

- $\Delta f$ ,  $\Delta n$  die Änderung der Kraft/des Drehmoments und
- $\Delta s$ ,  $\Delta \varphi$  die Änderung der Auslenkung/des Drehwinkels.

Werden Kraft/Moment und Auslenkung/Drehung gegeneinander aufgetragen, so ergibt sich die *Federkennlinie*. Dabei werden drei Typen an Kennlinien unterschieden:

- Progressive Kennlinie: Die Federsteifigkeit nimmt mit der Zeit zu.
- Degressive Kennlinie: Die Federsteifigkeit nimmt mit der Zeit ab.
- Lineare/Hookesche Kennlinie: Die Federsteifigkeit bleibt konstant (d. h.  $c = \text{const}$ ).

---

## Elastizitäten in der Robotik

---

In der Robotik können zwei Typen von Elastizitäten auftreten:

- Elastische Glieder  
Die Deformation ist abhängig von der Gelenkwinkelstellung  $q_i$ , der Gliedlänge  $r$  und der Temperatur  $T$ .
- Elastische Gelenke  
Führt zu einem Unterschied in der Gelenkposition: antriebsseitig  $\theta$ , abtriebsseitig  $\theta_{\text{El}}$ .

In der Regel werden diese Fälle getrennt betrachtet.

Im Ersatzmodell wird jedes Gelenk mit zwei Gelenkvariablen modelliert:

$\theta_i$  Die Position des starren Gelenkantriebs vor der Elastizität (antriebsseitig).

$q_i = \theta_{\text{El},i}$  Die tatsächliche Gelenkposition (abtriebsseitig).

Dadurch verdoppelt sich die Anzahl an Bewegungsgleichungen und es werden  $2n$  Gelenkvariablen zur Beschreibung der Bewegung benötigt.

Die Dynamikgleichungen lassen sich am besten durch eine Modifikation des Lagrange-Formalismus herleiten. Dabei wird die potentielle Energie

$$P(\mathbf{q}, \boldsymbol{\theta}) = P_G(\mathbf{q}) + P_{\text{El}}(\mathbf{q}, \boldsymbol{\theta})$$

durch einen Term  $P_{\text{El}}$  der in Gelenkverformungen gespeicherten Energie ergänzt ( $P_G(\mathbf{q})$  ist das Gravitationspotential, welches offensichtlich nur von den abtriebsseitigen Stellungen abhängt). Die potentielle Energie der Verformungen ist gegeben durch:

$$P_{\text{El}}(\mathbf{q}, \boldsymbol{\theta}) = \frac{1}{2} (\boldsymbol{\theta} - \mathbf{q})^T \mathbf{E}_S (\boldsymbol{\theta} - \mathbf{q})$$

Mit der diagonalen Steifigkeitsmatrix  $E_S$ . Die kinetische Energie  $K_R(q, \theta, \dot{q}, \dot{\theta})$  ist nun abhängig von den an- und abtriebsseitigen Positionen und Geschwindigkeiten. Mit den zusammengefassten verallgemeinerten Gelenkvariablen

$$p = \begin{bmatrix} q \\ \theta \end{bmatrix}$$

hat die kinetische Energie die Form

$$K_R(p, \dot{p}) = \frac{1}{2} \dot{p}^T M_R(p) \dot{p}$$

mit der erweiterten Trägheitsmatrix:

$$M_R(p) = \begin{bmatrix} M(q) & S(q, \theta) \\ S^T(q, \theta) & J_M(\theta) \end{bmatrix}$$

Die Teilmatrizen haben jeweils die Dimension  $(n \times n)$ , wobei

- die Matrix  $M$  der Massenmatrix des starren Roboters in Abhängigkeit von den abtriebsseitigen Position entspricht.
- $J_M$  ist die Trägheitsmatrix der Motoren und theoretisch abhängig von der Motorposition. Meistens ist die Massenverteilung des Rotors jedoch symmetrisch und der Schwerpunkt liegt auf der Drehachse, wodurch die Matrix diagonal und konstant wird.
- Die Matrix  $S$  ist die Trägheitsverkopplung zwischen Rotoren und den Gliedern. Sie entsprechen der kinetischen Energie der Rotoren um andere Achsen als der Motorachse. Meistens wird die kinetische Energie jedoch hauptsächlich durch die Drehgeschwindigkeit um die Motorachse verursacht, wodurch  $S = O$  gilt, d. h. die Matrix verschwindet.

Dadurch lässt sich die kinetische Energie schreiben als:

$$\begin{aligned} K_R &= K_A + K_M \\ \text{mit} \quad K_A(q, \dot{q}) &= \frac{1}{2} \dot{q}^T M(q) \dot{q} \\ K_M(\dot{\theta}) &= \frac{1}{2} \dot{\theta}^T J_M \dot{\theta} \end{aligned}$$

Wobei  $K_A$  die kinetische Energie in den Armgliedern und  $K_M$  die kinetische Energie in den rotierenden Motoren darstellt. Durch Lösen der Lagrange-Gleichungen ergibt sich ein gekoppeltes Differentialgleichungssystem:

$$\begin{aligned} \tau_{El} &= M(q) \ddot{q} + C(q, \dot{q}) + G(q) \\ \tau_m &= J_M \cdot \ddot{\theta} + \tau_{El} \\ \tau_{El} &= E_S \cdot (\theta - q) \end{aligned}$$

---

### Berechnung von INV DYN bei Drehgelenkelastizitäten

---

Zur Berechnung von INV DYN werden die folgenden Schritte ausgeführt:

1. Berechnung der Drehmomente  $\tau_{El}$  in den Gelenken mit dem starren Modell (z. B. mit Newton-Euler).
2. Berechnung von  $\ddot{\theta}$  durch zweifaches differenzieren und umstellen nach  $\ddot{\theta}$  von ??:

$$\ddot{\theta} = E_S^{-1} \cdot \ddot{\tau}_{El} + \ddot{q}$$

3. Berechnung der Motordrehmomente:

$$\tau_m = J_M \cdot \ddot{\theta} + \tau_{El}$$

---

## Elastizitäten in der Biologie

---

### Menschlicher Bewegungsapparat

Gelenkmodelle

Skelettmuskulatur

Muskel-Sehnen-Komplex

### Muskelaktivierungsdynamik

Muskelmodell

Hebelarme

### Weichteilmodelle

### Dynamikmodell

Dynamiksimulation

### Software und Daten

### Einschränkungen

---

## Steuerung und Regelung bei Mensch und Tier

---

### Reafferenzprinzip

---

## 5.8. Spezielle Dynamikmodelle für zweibeinige, humanoide Roboter und deren Stabilitätsregelung

---

Heutige humanoide Roboter bestehen aus starren kinematischen Ketten mit steifen Drehgelenken (meistens sechs bis sieben Drehgelenke pro Bein), sind Kaskadengeregelte.

Dabei beruht die posturale Stabilität auf dem *Zero Moment Point*. Der Begriff von *statisch stabilen Laufen* bezeichnet dabei, dass die Laufbewegung jederzeit gestoppt werden kann, ohne dass der Roboter umfällt. *Dynamische Stabilität* ist hingegen gegeben, wenn der Roboter zwar nicht statisch stabil ist, aber trotzdem nicht umfällt.



---

### 5.8.1. Zero-Moment-Point (ZMP)

---

Es wird ein stabiler (rutschfester) Stand auf einem Bein mit starrer Sole in einem Weltkoordinatensystem so, dass der Roboter in  $x$ -Richtung läuft und  $z$  nach oben zeigt, angenommen. Daraus resultieren Bodenreaktionskräfte und -momente. Diese sind zusammengesetzt aus normalen Kräfte

$$\mathbf{f}_{N_i} = [f_{N_i,z}]$$

und tangentialen Kräften

$$\mathbf{f}_{t_i} = \begin{bmatrix} f_{t_i,x} \\ f_{t_i,y} \\ 0 \end{bmatrix}$$

Der *Zero-Moment-Point* ist nun derjenige Punkt  $P$ , in dem die resultierende normale Kraft  $\mathbf{F}_N = \sum_i \mathbf{f}_{N_i}$  wirkt:

$$\mathbf{r}_{\text{ZMP}} := \frac{\sum_i \mathbf{r}_{N_i} f_{N_i,z}}{\sum_i f_{N_i,z}}$$

Dabei ist  $\mathbf{r}_{N_i}$  der Angriffspunkt der Kraft  $\mathbf{f}_{N_i}$ . Durch die tangentialen Kräfte kann  $\mathbf{N}_t$ , der Vektor der Drehmomente am Punkt  $P$ , bestimmt werden:

$$\mathbf{N}_t = \sum_i \mathbf{d}_i \times \mathbf{f}_{t_i} = \begin{bmatrix} * \\ * \\ 0 \end{bmatrix} \times \begin{bmatrix} * \\ * \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ * \end{bmatrix}$$

Somit wirkt am Zero-Moment-Point zwar ein Moment, dieses führt aber nicht zum Kippen.

Der Roboter kann nur dann kippen, wenn sich der ZMP am Rand der konvexen Hülle der Bodenkontaktpunkte liegt. Daher muss der ZMP strikt innerhalb der konvexen Hülle gehalten werden.

---

### 5.8.2. Center of Pressure (CoP)

---

Ein alternatives Stabilitätsmaß ist das *Center of Pressure* (CoP). Dieser ist die resultierende Ersatzkraft aus allen Kräften, die auf die Sole des Fußes wirken und den Bodenreaktionskräften entgegen wirken.

Achtung: ZMP und CoP beschreiben nicht das gleiche! Dies ist nur der Fall bei einem dynamisch ausbalancierten Laufroboter.

---

### 5.8.3. Inverses Pendel und Feder-Masse-Modell

---

Für humanoide Roboter (mit oft mehr als 30 Gelenken) ist das aufstellen des inversen Dynamikmodells häufig zu komplex und damit die Berechnung nicht echtzeitfähig. Daher werden vereinfachte Dynamikmodelle eingesetzt, z. B. ein inverses Pendel.

Die Energien eines seitwärts beschleunigten, inversen Pendels mit Länge  $l$  und Auslenkung  $\theta$  ergibt sich die (um die Nullauslenkung, d. h. aufrechte Haltung) linearisierte Bewegungsgleichung

$$l\ddot{\theta} - g\theta = -\ddot{x}$$

Dieses Modell liefert nur gute Ergebnisse für langsames Gehen. Alternativ kann ein Feder-Masse-Modell mit

- Masse  $m$ ,
- Ruhelänge  $l_0$ ,

- Federkonstante  $k$  und
- momentanem Zustand  $(r, \varphi)$  in Polarkoordinaten

verwendet werden. Die Bewegungsgleichungen lauten dann:

$$\begin{aligned} m\ddot{r} &= k(l_0 - r) + mr\dot{\varphi}^2 - mg \sin \varphi \\ r\ddot{\varphi} &= -2\dot{r}\dot{\varphi} - g \cos \varphi \end{aligned}$$

---

#### 5.8.4. Globale Stabilitätsbegriffe

---

---

#### 5.8.5. Ausblicke

---

---

##### Capture Steps

---

- Ein *Capture Point* ist ein Punkt, auf dem der Roboter beim Betreten vollständig zum Stillstand kommen kann. Ist dieser Punkt erreichbar, so wird er auch *Capture Step* genannt.
- Capture Steps sind eine effektive Methode, um (harte) externe Stöße abzufangen.

---

##### Unebenes Terrain

---

Auf unebenem Terrain gelten die vorherigen Annahmen für den ZMP nicht mehr, weshalb andere Ansätze notwendig sind.

---

### 5.9. Spezielle Dynamikmodelle für zweibeinige, nicht-humanoide Roboter und deren Stabilitätsregelung

---

In diesem Abschnitt werden z. B. hüpfende Roboter betrachtet.

---

#### 5.9.1. Hüpfende Roboter mit Teleskop-Beinen

---

Bei einem hüpfendem Roboter wird die absorbierte Energie beim Bodenkontakt durch Kompression zwischen den Sprüngen transferiert.

---

##### Ein Modell dynamischer Stabilität

---

Das Steuerungsproblem für Hüpfroboter kann in drei unabhängige, zeitabhängige Teilprobleme zerlegt werden:

1. Hüpfhöhe
2. Vorwärtsgeschwindigkeit
3. Haltung

---

Diese Probleme können mit je einer Differentialgleichung modelliert werden, wobei die drei Steuerungen durch einen diskreten Zustandsautomaten für den Hüpfzyklus synchronisiert werden.

Dieser hat fünf diskrete Zustände und Übergänge mit jeweils eigener, kontinuierlicher Systemdynamik (hybride diskret-kontinuierliche Systemdynamik). Dieses „Umschalten“ von kontinuierlichen Systemdynamikmodell für die verschiedenen Phasen einer Gangart ist ein heute gängiges Vorgehen bei laufenden Robotern. Sensoren messen dabei die Zeitpunkte

- des Bodenkontakts,
- wenn das Bein nahe voller Ausdehnung oder
- wenn das Bein nahe voller Kompression ist.

Diese Steuerung kann ebenfalls auf mehrbeinige, hüpfende Roboter übertragen werden.

---

### **5.9.2. Passive Dynamic Walkers**

---

Passive Laufmaschinen haben keine Antriebe, kein Regelungssystem und sind rein durch die Gravitationskraft angetrieben (schiefe Ebene). Diese Art der Fortbewegung ist nur begrenzt auf zweibeinige Roboter übertragbar, da weder Bergauf-Gehen, Kurven-Gehen oder Manipulation von Lasten möglich ist.

---

## 6. Antriebssysteme

---

---

### 6.1. Gebräuchliche Antriebssysteme

---

---

#### 6.1.1. Hydraulische Antriebe

---

Hydraulische Antriebe nutzen Öl und Druckveränderungen dieses Öls in einer Kammer zur Bewegung.

- **Vorteile:**

- Können hohe Lasten tragen.
- Gutes Verhältnis von Leistung zu Gewicht.
- Durch die Inkompressibilität von Öl kann der Antrieb gut in einer Stellung fixiert werden.
- Der Antrieb schmiert und kühlt sich selbstständig.
- Schnelle Reaktionszeit.
- Sicher in Umgebungen mit brennbaren Gasen (da keine Funken entstehen).
- Flüssige Bewegungen sind auch bei langsamer Geschwindigkeit möglich.

- **Nachteile:**

- Teuer.
- Die Dichtungen müssen gut gewartet werden, sonst entstehen Lecks.
- Stark limitierte Geschwindigkeiten.
- Benötigen eine Rückführleitung.
- Können aufgrund der hohen Leitungsdrücke und -flüsse schlecht klein gebaut werden (Miniaturisierung).
- Großer Platzbedarf für externe Leistungsversorgung.
- Das Regel- und Positionierverhalten ist Temperaturabhängig.

Daher gibt es nur wenige mobile hydraulisch aktuierte Roboter (z. B. Atlas und BigDog von Boston Dynamics).

---

#### 6.1.2. Pneumatische Antriebe

---

Pneumatische Antriebe nutzen Pressluft und Veränderungen des Drucks zur Bewegung.

- **Vorteile:**

- Preiswert.
- Pressluft ist in der Industrie überall verfügbar.

- Hohe Geschwindigkeiten sind möglich.
- Keine Verunreinigung durch auslaufende Flüssigkeiten (wie Öl).
- Reine Rückführleitungen nötig.
- Aufgrund der Kompressibilität von Luft gibt der Antrieb auf natürliche Weise nach: vorteilhaft für Interaktionen mit Menschen.

- **Nachteile:**

- Die Kompressibilität der Luft beschränkt die Regel- und Positionierbarkeit (der Roboter kann nicht gut in einer Position fixiert werden).
- Die entweichende Luft erzeugt Lärm.
- Zusätzliches Trocknen/Filtern der Luft kann erforderlich sein.
- Das hochheben von Lasten und Druckabfälle in der Leitung erschweren die Regelung.
- Ungeeignet für mobile Anwendung, da ein Kompressor benötigt wird.

---

### 6.1.3. Elektrische Antriebe

---

- **Vorteile:**

- Schnelligkeit und Genauigkeit.
- Es sind ausgefeilte Regelungsmethoden anwendbar.
- Preiswert.
- Neue Motortypen werden schnell entwickelt, bzw. können schnell entwickelt werden.

- **Nachteile:**

- Typischerweise haben die Motoren hohe Geschwindigkeiten bei niedrigem Drehmoment, d. h. es sind Getriebe notwendig.
- Das Getriebeispiel beschränkt die erreichbare Genauigkeit.
- Elektrische Lichtbögen (Funken, Blitze) sind problematisch bei entzündlicher Atmosphäre (z. B. innerhalb von entzündlichen Gasen).
- Im Dauereinsatz ist eine Überhitzung möglich.
- Es sind Bremsen notwendig, um eine Position dauerhaft zu fixieren.

---

## 6.2. DC-Bürsten-Motoren

---

Durch das anlegen einer Spannung an den *Bürsten*, die den *Kollektor* berühren, wird eine Spannung induziert. Durch die Rotorwicklung wird ein magnetisches Felds erzeugt, was das magnetische Fels des Permanentmagneten verzerrt. Das hieraus resultierende Drehmoment führt zur Bewegung des Kollektors. Durch das Reiben der Bürsten an dem Kollektor entstehen Funken, die in einer entzündlichen Umgebung zu Bränden führen können.

Abbildung 6.1 zeigt den elektrischen Stromkreis eines DC-Bürsten-Motors mit allen relevanten elektrischen Größen. Mit der Drehzahlkonstante  $k_v$  sowie der Drehmomentkonstante  $k_t$  und den mechanischen

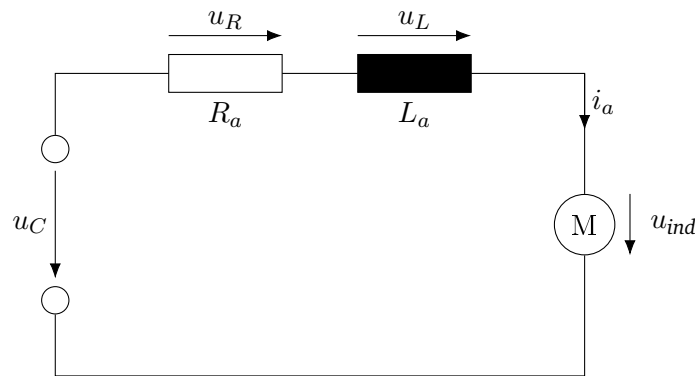


Abbildung 6.1.: Stromkreis eines DC-Bürsten-Motors mit dem Widerstand der Spule  $R_a$ , der Induktivität der Spule  $L_a$ , dem Ankerstrom  $i_a$ , der angelegten Spannung  $u_a$  und der induzierten Spannung  $u_{ind}$ .

Größen des Motordrehmoments  $\tau_r$  und der Motorgeschwindigkeit  $\dot{\theta}_r$  wird der Stromkreis durch folgende (Differential-) Gleichungen beschrieben:

$$u_C = R_a i_a + L_a \frac{di_a}{dt} + u_{ind} \quad (\text{Ankerstromkreis})$$

$$u_{ind} = \dot{\theta}_r k_v \quad (\text{Induzierte Spannung})$$

$$\tau_r = i_a k_t \quad (\text{Motordrehmoment})$$

## 6.3. Getriebe

Direkte Elektrische Antriebe produzieren hohe Geschwindigkeiten bei geringen Drehmomenten. Daher ist ein *Getriebe* zur Übersetzung notwendig, wenn hohe Drehmomente gewünscht sind (dies ist meistens der Fall, bspw. wenn schwere Lasten gehoben werden sollen). Zu den gebräuchlichsten Getriebe-Bauarten zählen Gewinde, Riemen-, Ketten oder Seilzug-Getriebe sowie Rädergetriebe. Diese werden in den folgenden Abschnitten detaillierter behandelt.

Die *Getriebeübersetzung* bezeichnet im Allgemeinen das Verhältnis von Eingangs- zu Austrittsgeschwindigkeit aus dem Getriebe. Dieses ist dabei gleich dem Verhältnis von Ausgangs- zu Eingangsdrehmoment:

$$\text{Getriebeübersetzung} := \frac{\text{Eingangsgeschwindigkeit}}{\text{Ausgangsgeschwindigkeit}} = \frac{\text{Ausgangsdrehmoment}}{\text{Eingangsdrehmoment}}$$

Bei elektrischen Antrieben werden vor allem *Getriebeübersetzungen* verwendet, die die Ausgangsdrehgeschwindigkeit reduzieren und das Ausgangsdrehmoment erhöhen.

### 6.3.1. Gewinde

Ein *Gewinde* funktioniert wie eine „inverse Schraube“, d. h. eine Schraube mit Gewinde wird durch eine Platte bewegt, die sich daraufhin linear bewegt. Dies führt zu einer großen Geschwindigkeitsreduktion und damit zu einer starken Momenterhöhung. Durch die hohe Steifigkeit des Systems sind außerdem hohe Traglasten möglich.

---

### 6.3.2. Riemen-/Seilzug-Getriebe

---

Bei einem *Riemen-* oder auch *Seilzug-Gewinde* bestehen aus zwei unterschiedlich großen Rädern (z. B. Zahnrädern) mit den Radien  $r_1$  und  $r_2$ , die über einen Riemen verbunden sind (dabei ist  $r_1$  der Eingang, d. h. dieses Rad wird von dem Motor gedreht.). Problematisch sind hier die auftretenden Elastizitäten im Riemen. Zur Vermeidung dieser muss der Riemen durchgehend gespannt bleiben.

Die Getriebeübersetzung gleich dem Quotient der beiden Radien:

$$\text{Getriebeübersetzung} = \frac{r_2}{r_1}$$

---

### 6.3.3. Rädergetriebe

---

Bei Zahnrädern treten einige elementare Probleme auf:

- (Verdreh-) Spiel (*Backlash*) durch ungenaue Verzahnung (d. h. die Räder können ein bisschen hin und her gewackelt werden).
- Reibung zwischen den Rädern.
- Durch eine enge Verzahnung kann das Spiel verringert werden, dies führt aber zu einer höheren Reibung.

Gängige Typen von Zahnradgetrieben sind

- Stirnradgetriebe,
- Planetengetriebe und
- Harmonic Drive Getriebe.

Diese werden in den kommenden Abschnitten näher betrachtet.

---

#### Stirnradgetriebe

---

Ein *Stirnradgetriebe* besteht aus ein oder mehreren Paarungen von Zahnrädern, wobei das erste Zahnrad direkt auf der Motorwelle montiert ist. Diese Getriebe sind preisgünstig und eignen sich für kleine Drehmomente.

---

#### Planetengetriebe

---

Ein *Planetengetriebe* besteht aus einem zentralen *Sonnenrad*, welches von mehreren *Planetenrädern*, die auf einem Planetenradträger montiert sind, umkreist wird. Um diese liegt wiederum das *Hohlrad*, welches mit allen Planetenrädern verzahnt wird. Das Getriebe hat drei Wellen (Sonnenrad, Planetenradträger, Hohlrad), wodurch sechs Übersetzungsmöglichkeiten existieren. Die üblichsten sind:

1. Sonnenrad angetrieben, Hohlrad festgehalten, Planetenradträger als (langsamere) Abtriebswelle
2. Sonnenrad angetrieben, Planetenradträger festgehalten, Hohlrad als (langsamere) Abtriebswelle (rückwärts)
3. Hohlrad angetrieben, Sonnenrad festgehalten, Planetenrad als (langsamere) Abtriebswelle
4. Sonnenrad und Hohlrad mit gleicher Drehzahl angetrieben, Planetenradträger als (gleich schnelle) Abtriebswelle

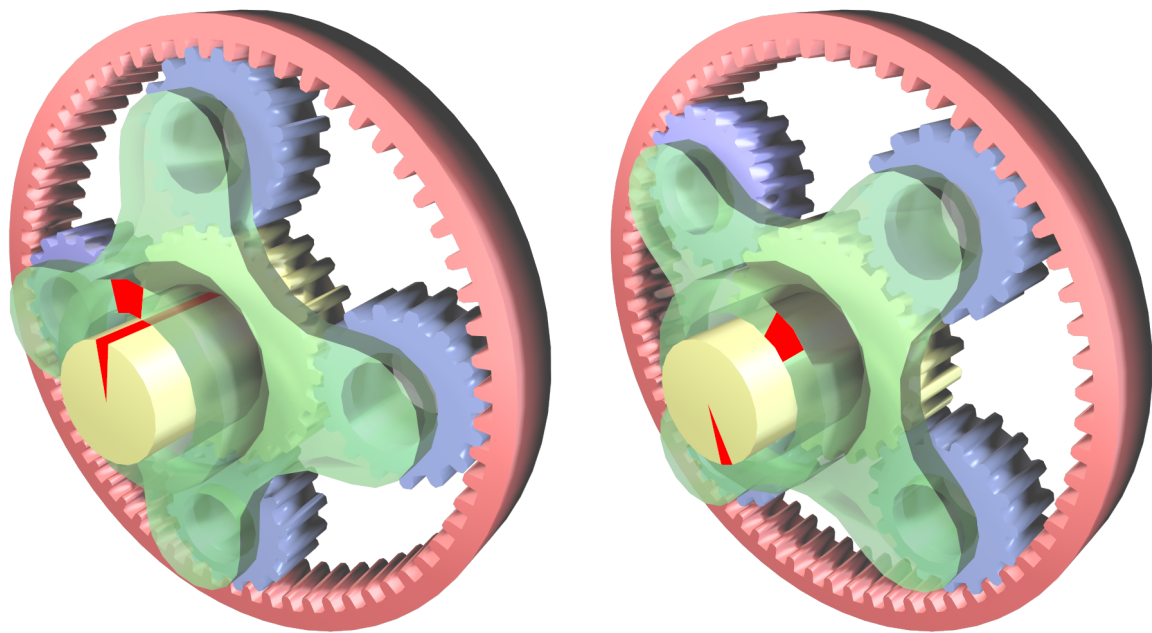


Abbildung 6.2.: Planetengetriebe mit Sonnenrad (gelb), Planetenrädern (blau), Planetenradträger (grün) sowie Hohlrad (rot). Das rechte Bild zeigt die relative Verschiebung von Sonnenrad und Planetenradträger, nachdem letzterer um  $45^\circ$  rotiert wurde. Quelle: Wikipedia

---

## Harmonic Drive Getriebe

---

Ein *Harmonic Drive Getriebe* besteht aus:

- *Elliptischer Wave Generator*  
Elliptische Stahlscheibe mit zentrischer Nabe sowie aufgezogenem elliptisch Verformbaren Kugellager.
- *Flexspline*  
Zylindrische, verformbare Stahlbüchse mit Außenverzahnung.
- *Circular Spline*  
Zylindrischer Ring mit Innenverzahnung.

Dabei wird der elliptische Wave Generator angetrieben und verformt über das Kugellager den Flexspline. Dieser befindet sich gegenüber der großen Ellipsenachse mit dem Circular Spline im Eingriff. Drehen des Wave Generators führt zu einer Verlagerung der großen Ellipsenachse und damit zu einer Verlagerung des Zahneingriffsbereichs von Flexspline und Circular Spline. Eine halbe Umdrehung des Wave Generators führt dann zu einer Relativbewegung zwischen Flexspline und Circular Spline um einen Zahn.

- **Vorteile:**
  - Spielfreiheit (d. h. es existiert kein Spiel).
  - Sehr genaue Positionier- und Wiederholgenauigkeit
  - Kompakte Bauweise.
  - Hohe Drehmomentkapazität.



- Hoher Wirkungsgrad.
- Hohe Torsionssteifigkeit (d. h. das Getriebe ist sehr resistent gegenüber Verformungen).
- Hohe Zuverlässigkeit und lange Lebensdauer.

- **Nachteile:**

- Teuer.
- War bislang nicht beliebig klein realisierbar, neue Entwicklungen ermöglichen aber auch kleine Varianten.

---

## 6.4. Alternative und elastische Antriebskonzepte

---

### 6.4.1. Beine

---

### 6.4.2. Neue Materialien

---

### 6.4.3. Compliant Robot Actuation

---

### 6.4.4. Elastische Antriebskonzepte

---

Die Grundidee ist die Kombination elektrischer Antriebe mit mechanischer (potentiell verstellbarer) Elastizität.

- Seriell-Elastische Antriebe (SEA):

- **Stärken:**

- \* Energiespeicherung reduziert die benötigte Leistungsspitze des Motors und den Energieverbrauch.
- \* Die Entkopplung von Antrieb und Abtrieb liefert Stoßkompensation.
- \* Sicherheitspotential.

- **Schwächen:**

- \* Erhöhte Masse des Antriebssystems.
- \* Erhöhte Komplexität der mechanischen Konstruktion.
- \* Erhöhte Komplexität der Regelung.

- Parallel-Elastische Antriebe (PAE):

- **Stärken:**

- \* Energiespeicherung reduziert die benötigte Leistungsspitze des Motors und den Energieverbrauch, allerdings potentiell auch das Motordrehmoment bei SPitzen.

- **Schwächen:**

- \* Erhöhte Masse des Antriebssystems.
- \* Erhöhte Komplexität der mechanischen Konstruktion.
- \* Erhöhte Komplexität der Regelung.

---

---

**Variable Stiffness Actuator**

---

**Variable Impedance Actuators**

---

**6.4.5. Vom Muskel-Skelett-Apparat inspirierte Roboter**

---

---

## 7. Sensoren

---

Ein *Sensor* ist ein technisches System, welches eine physikalische (analoge) Größe (z. B. Position) und gegebenenfalls deren Änderung (z. B. Geschwindigkeit) in geeignete (unsicherheitsbehaftete) elektronische (digitale) Signale transformiert.

Ein *internen Sensor* („proprioceptive sensor“) erfasst dabei die Roboter-internen Zustände, wobei ein *externer Sensor* („exteroceptive sensor“) Informationen über den Zustand der Umwelt sammelt.

---

### 7.1. Interne Sensoren

---

Interne Sensoren können grob in die Kategorien

- Positionssensoren
- Geschwindigkeitssensoren
- Beschleunigungssensoren
- Inertial Navigation System (INS)
- Kraft-Momenten-Sensoren

unterteilt werden.

---

#### 7.1.1. Positionssensoren

---

---

##### Potentiometer

---

Ein *Potentiometer* besteht aus einem *Widerstandselement* mit darüber verlaufender Spannung und einem gleitendem Kontakt (*Wischer*, „wiper“), der sich über das Widerstandselement bewegt. Dabei kann sich entweder der Wischer oder das Widerstandselement bewegen. Abbildung 7.1 zeigt ein lineares und ein rotatorisches Potentiometer.

Die Spannung im Kontakt ist dann abhängig von der Position auf dem Widerstandselement und damit proportional zur Gelenkposition, womit die Position gemessen werden kann. Die Genauigkeit liegt (unter der Voraussetzung einer stabilen Spannungsquelle) bei ca. 0.5 %.

---

##### Optische Codierer

---

Auf einer Scheibe wird ein kodierte Muster angebracht, welches den Strahl einer Lichtquelle periodisch unterbricht, wobei diese Quelle auf einen Photodetektor ausgerichtet ist. Durch Messung der Unterbrechung kann die Position gemessen werden. Dabei werden zwei Arten von optischen Kodierern unterschieden:

- Inkrementelle Codierer
- Absolute Codierer

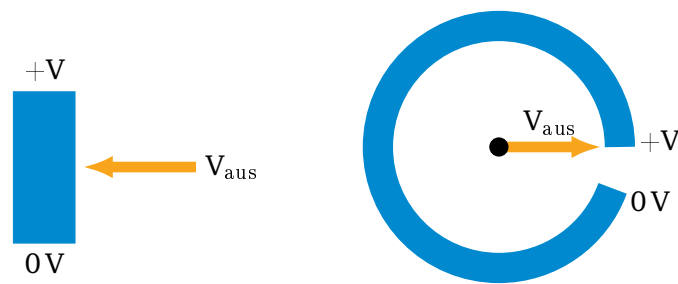


Abbildung 7.1.: Lineares (links) und rotatorisches (rechts) Potentiometer mit Widerstandselement (blau) und Kontakt (orange).

**Inkrementell** Die Empfangseinheit zählt die Inkremente (die durch die periodischen Unterbrechungen erzeugt werden) und zählt diese (daher der Name *inkrementeller Codierer*). Daraus kann anschließend die Position berechnet werden. Ein großer Nachteil dieser Methode ist, dass die Startposition unbekannt ist, d. h. der Zähler muss zunächst kalibriert werden.

Durch mehrspurige Codierer können durch die Verschiebung der Inkremente mehr Daten gemessen werden:

- 1-spuriger Codierer: Winkel
- 2-spuriger Codierer: Winkel und Richtung
- 3-spuriger Codierer: Winkel, Richtung, Anfang/Ende

**Absolut** Im Gegensatz zum inkrementellen wird beim *absoluten Codierer* jeder Achsenposition ein individuelles Wortmuster zugeteilt, welches anschließend gemessen wird. Dadurch ist ein direktes Ablesen der Gelenkposition möglich und somit keine Kalibrierung notwendig. Allerdings ist die Konstruktion sehr aufwendig.

Mit dem Gray-Code unterscheidet sich der Code an jeder Stelle nur um ein Bit, wodurch große Fehler vermieden werden.

---

## Resolver

---

Resolver sind robust, störungssicher und haben eine hohe Lebensdauer, weshalb sie oft bei Industrierobotern eingesetzt werden.

---

### 7.1.2. Geschwindigkeitssensoren

---

Geschwindigkeitssensoren lassen sich bspw. durch Differentiation und Filterung aus mit hoher Frequenz abgetasteten Positionswerten erstellen (d. h. die Geschwindigkeit wird indirekt gemessen).

---

### 7.1.3. Beschleunigungssensoren (Silizium-Beschleunigungssensor)

---

In einem *Silizium-Beschleunigungssensor* befindet sich eine träge Masse in einem „Siliziumbad“, wobei Auslenkungen der Aufhängung die mechanischen Spannungen ändern, wodurch der piezoresistive Widerstandswert geändert wird. Dieser Wert ist messtechnisch erfassbar, wodurch Beschleunigungen erkannt werden können. Allerdings ist eine Erfassung von Richtung und Betrag der Gravitation nötig, um diese herausrechnen zu

---

können. Durch die Verwendung von drei 1D-Sensoren lässt sich ein 3D-Beschleunigungssensor konstruieren (indem diese orthogonal zueinander angebracht werden).

---

#### 7.1.4. Inertial Navigation System (INS)

---

Die Aufgabe eines *Inertial Navigation Systems* (INS) ist die Bestimmung der Orientierung relativ zu einem Inertialsystem. Dazu lassen sich bspw. folgende Sensoren einsetzen:

- Kompass
- Lagesensor
  - Hierbei wird eine elektrisch leitfähige Flüssigkeit in einem versiegeltem Glas mit drei Elektroden platziert.
  - Eine Lageänderung verursacht eine ungleiche Flüssigkeitsverteilung und damit ungleiche Widerstände relativ zum Winkel.
  - Daraus kann anschließend die Position berechnet werden.
- Gyroskop (mechanisch oder mikromechanisch)

---

#### Mechanisches Gyroskop

---

Ein *Gyroskop* ist ein schnell rotierender, semi-kardanisch aufgehängter Kreisel. Durch die Erdrotation wird ein Drehmoment in Richtung des Medians verursacht, wodurch eine Richtungsanzeige möglich wird.

Angewendet werden Gyroskope an vielen Stellen:

- Linienflugzeug (ca. 12 Gyroskope)
- Raumstation MIR (11 Gyroskope zur Orientierungshaltung zur Sonne)
- Häufig werden 3D-Beschleunigungssensoren für die lineare Beschleunigung und Gyroskope für die 3D-Winkelgeschwindigkeiten verwendet. Die Integration über die Geschwindigkeiten ergibt die zurückgelegte Bewegung (*Odometrie*). Durch Filtern der Messwerte können Fehler und Drifts kompensiert werden.
- Ein normales Flugzeug-Gyroskop hat pro Betriebsstunde einen Drift von ca. 1.85 km, „High-End“-Gyroskope kommen auf einen Drift von weniger als 0.1 % der zurückgelegten Distanz.

---

#### Mikromechanische Gyroskope

---

Bei mikromechanischen Gyroskopen wird die Rotation durch schwingende, mechanische Elemente registriert. Sie werden z. B. als „Stimmgabel“ realisiert:

- Zwei Zinken mit unterschiedlichen, festen Schwingfrequenzen.
- Bei einer Rotation verursacht die Corioliskraft eine differentielle, sinusförmige Kraft orthogonal zur Hauptschwingung in jedem Zinken.
- Durch die differentielle Verbiegungen der Zinken oder die Torsionsschwingung am Stamm der Stimmgabel kann diese Kraft detektiert werden.

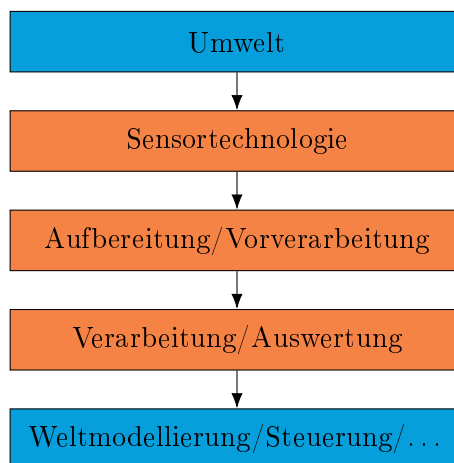


Abbildung 7.2.: Typische Struktur eines Sensorsystems, wobei die blau hinterlegten Schritte (größtenteils) analog und die orange hinterlegten Schritte (größtenteils) digital ablaufen.

- Die Anregung der Resonanzfrequenz der Zinken kann elektrostatisch, elektromagnetisch oder piezoresistiv erfolgen
- Die durch die Corioliskraft verursachten Schwingungen können kapazitiv, piezoresistiv oder piezoelektrisch detektiert werden.
- Eine optische Detektion ist auch möglich, ist i. A. allerdings sehr teuer.

---

### 7.1.5. Kraft-Momenten-Sensoren

---

Zeil von *Kraft-Momenten-Sensoren* ist die Messung der Kräfte und Momente zwischen Effektor und Objekt. Oft wird hierfür eine sogenannte *Kraftmessdose* verwendet, die eine Kombination aus mehreren Messapparaturen darstellt. Typische Apparaturen sind hierbei:

- Dehnungsmessstreifen (DMS)
- Piezokristalle
- Optische Effekte

Die Dose wird dabei zwischen Effektor und Roboterhand und -fuß angebracht.

Oft haben Kraft-Momenten-Sensoren eine Speichenradform, wobei auf den Speichen/Stegen Dehnungsmessstreifen angebracht sind. Wirkt eine Kraft auf die Dose, so ändert sich die Länge der Stege. Auf den Dehnungsmessstreifen ist eine dünne Metallfolie in Matrixform angebracht, die bei Änderung der Länge ihren Widerstand ändert. Durch Messung dieser Änderung kann die Krafteinwirkung gemessen werden.

---

## 7.2. Externe und intelligente Sensoren

---

Abbildung 7.2 zeigt die typische Struktur eines (externen) Sensorsystems.

Externe Sensoren können grob in die Kategorien

- Taktile Sensoren

- 
- Näherungssensoren
  - Abstandssensoren
  - Positionssensoren
  - Visuelle Sensoren

unterteilt werden.

---

### 7.2.1. Abstandssensoren

---

*Abstandssensoren* messen den Abstand zwischen dem Sensor und einem Gegenstand. Diese Sensoren sind geeignet zur Erfassung von geometrischen Umweltinformation. Außerdem haben sie eine größere Reichweite als Näherungssensoren.

Es werden drei grundlegende Typen von Abstandssensoren unterschieden:

- Optisch (Nutzung von sichtbaren, elektromagnetischen Wellen)
- Radar (elektromagnetische Wellen mit sehr kurzen Längen)
- Akustisch (mechanische Wellen)

---

#### Akustische Abstandssensoren

---

*Schall* ist ein an Materie gebundener Energie- und Impulstransport, wobei eine *Schallwelle* die wellenförmige Ausbreitung der periodischen Anregung eines Übertragungsmediums beschreibt. Der hörbare Schall liegt dabei zwischen 16 Hz und 16 kHz, wobei alles darunter als *Infraschall* und alles darüber als *Ultraschall* bezeichnet wird.

In den 1980er und 1990er sind Ultraschallsensoren die wichtigsten Sensoren zur Erdfassung der Umwelt. Seit Ende der 1990 werden jedoch vermehrt Laserscanner kombiniert mit Farbkameras eingesetzt. Im Nahbereich sind Ultraschallsensoren jedoch auch noch heute nützlich (Kollisionsdetektion, bzw.-vermeidung), sowie in Bereichen wo weder Laserscanner noch Kameras einsetzbar sind (z. B. in Gebäuden mit Rauch).

**Ultraschall** Die Messung von Abständen mittels Ultraschall heißt *sonar* („sound navigation and ranging“). In der Natur wird diese Art der Navigation häufig eingesetzt, z. B. von Delfinen und Fledermäusen.

Zur Erzeugung von Ultraschall wird eine schwingende Folie eingesetzt, die im Sende-Modus durch die elektrostatische Kraft, ausgelöst von einer geladenen Aluplatte, vibriert. Im Empfangs-Modus bewegt der Klangdruck die Folie, wodurch sich die Kapazität des Kondensators ändern (elektrostatisches Mikrofon).

Im Normalbetrieb sendet ein Sensor 1 ms bis 1.2 ms langes „Zwitschern“ pro 200 ms. Dieses Zwitschern wird aufgrund der unterschiedlichen Materialbeschaffenheit und deren unterschiedlichen Reflexionseigenschaften in unterschiedlichen Frequenzen gesendet.

#### Messsituationen

- Im Idealzustand ist die Achse der Schallkeule orthogonal zu einem flachen Objekt. In diesem Fall wird ein genauer Abstand gemessen.
- Befindet sich ein kleines Objekt vor der Wand, so wird der Abstand weiterhin genau gemessen, die Querposition ist jedoch ungenau.

- Steht der Sender rotiert zur Wand, so ist der gemessene Abstand vom Rand der Schallkeule zu kurz (Unterschätzung). Die Unsicherheit des Abstands ist eine Funktion des Rotationswinkels und des Keulenöffnungswinkels.
- Ist die Rotation größer als der Keulenöffnungswinkel, so ist die Wand unsicherbar.
- Eine Kante ist ebenfalls unsichtbar, solange die Ecke nicht ein wenig abgerundet ist.
- Mehrfachreflexionen (wie in einer Ecke) bewirkt eine Starke Überschätzung des Abstands.

**Abhilfe der Schwierigkeiten** Bei nur einem Empfänger darf die Umweltszene nur ein Objekt innerhalb der Schallkeule enthalten der reflektierende Objektteil muss orthogonal zur Strahlrichtung stehen.

Gebräuchliche Abhilfen sind die Verwendung von mehreren Ultraschallsensoren und algorithmische Verfahren wie Peilverfahren: Die Position wird aus den Laufzeitunterschieden, z. B. mit Triangulationsverfahren (ähnlich wie bei GPS), berechnet.

**Natur** Fledermäuse haben sehr unterschiedliche Ultraschalle (enge Schallkeulen, weite Schallkeulen, Zwitschern mit fester/veränderlicher Frequenz, ...). Der aktuelle Stand der Ultraschallerzeugung ist vergleichbar mit dem Ultraschall von Fledermäusen, allerdings liegen die technischen Fähigkeiten zur Verarbeitung noch weiter hinter jenen von Fledermäusen.

---

## Optische Abstandssensoren

---

Der Aufbau von optischen Abstandssensoren besteht aus einem Emitter (LED, Laserdioden) und einem Empfänger (Fototransistor). Optische Sensoren liefern sehr genaue Entfernungsmessungen und sind stabil gegen viele Fremdeinflüsse.

Es gibt verschiedene Messverfahren:

- Ermittlung der Flugzeit (Laufzeit)
- aktive Triangulation
- Interferometrie (Phasenverschiebung)
- Stereoskopie

**Laufzeitermittlung** Bei der *Laufzeitermittlung* wird die Hin- und Rücklaufzeit eines Impulses (z. B. Lichtimpuls) gemessen. Wurde zwischen Aussenden und Empfangen die Zeit  $\Delta t$  gemessen, so berechnet sich die Entfernung  $d$  wie folgt:

$$d = \frac{c \cdot \Delta t}{2}$$

Es muss die Hälfte genommen werden, da der Lichtstrahl zum Objekt hin und zurück verläuft.

Typische Probleme sind die Absorption des Strahls vom Objekt, Wegreflexion (z. B. bei einem Spiegel) sowie Mehrfachreflexion (wodurch der Weg verlängert wird).



---

**2D-Laserscanner** In einem *Laserscanner* wird ein gepulster Laserstrahl, abgelenkt durch einen beweglichen Spiegel, ausgesandt. Durch diesen Drehspiegel wird die Umgebung fächerförmig abgetastet. Trifft der Impuls auf ein Objekt, so wird er reflektiert und die Entfernung kann mittels Laufzeitermittlung berechnet werden. Durch die Abfolge der empfangenen Impulse kann die Entfernung, Lage und Kontur eines Objekts berechnet werden.

Bei der *Kartografierung* werden alle gescannten Szenen (gescannt mit einem mobilen System) zu einer Karte zusammengefasst.

---

### 7.2.2. Visuelle Sensoren

---

- Visuelle Wahrnehmung („visual perception“)
  - Aufnahme und Verarbeitung von visuellen Reizen
  - Auge und Gehirn extrahieren relevante Informationen
  - Erkennen von Elementen und deren Interpretation und Abgleich mit Erinnerungen
- Maschinelles Sehen („machine vision“)
  - Digitale Bildverarbeitung und Mustererkennung
  - Eingesetzt z. B. bei Prüfverfahren in der industriellen Fertigung
- Bildverstehen („computer vision“)
  - Extraktion komplexer Interpretationen und Schlüsse aus 2D-Abbildungen realer 3D-Szenen
  - Viele langfristig orientierte Grundlagenforschungsfragen
- Robotersehen („robot vision“)
  - Maschinelles Sehen und Bildverstehen unter den Echtzeitbedingungen eines Roboters
  - Begrenzte Rechenleistung und schnelle Reaktionszeiten

Voraussetzungen für maschinelles Sehen sind:

1. „Technisches Auge“, d. h. ein Kamerachip mit Optik
2. „Intelligente“, bildverarbeitende Algorithmen
3. Ausreichende Strukturierung (Material, Merkmale der relevanten Objekte, ...) der Szene
4. Ausreichende Beleuchtung (Helligkeit, Lampen-Lichtspektrum, Reflexionen, ...) der Szene

Je nach Aufgabe ist eine Abstimmung dieser vier Voraussetzungen nötig, d. h. es gibt keine allgemeine Lösung. Die klassische Hierarchie der Bildverarbeitungsoperationen ist in Abbildung 7.3 aufgezeigt.

---

### Bilderzeugung

---

Die *Kameramatrix* beschreibt die Projektionsabbildung von 3D-Punkten durch eine Lochkamera auf 2D-Punkte der Bildebene. Dadurch können spezifische Verzerrungseffekte der Kameralinse berücksichtigt werden.

Zur Bilderzeugung gibt es mehrere grundlegende Methoden von Bildwandlersystemen (Kamerachip):

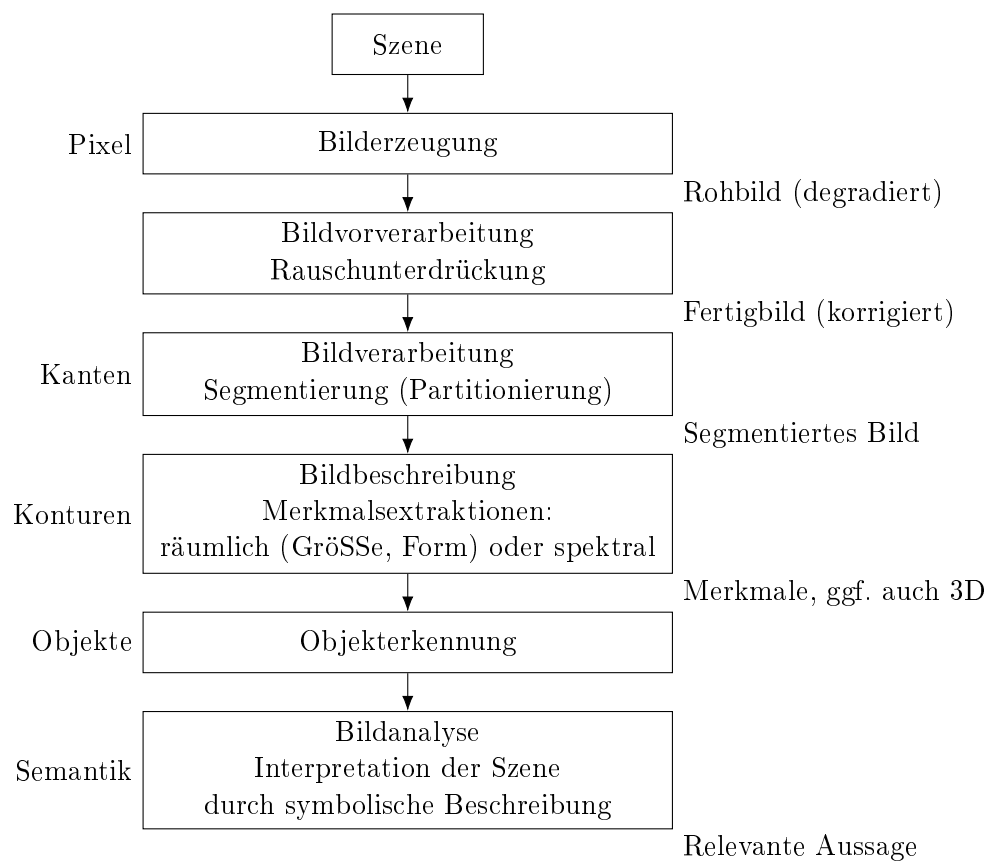


Abbildung 7.3.: Klassische Hierarchie der Bildverarbeitung. Dabei findet alles bis zur Bildverarbeitung numerisch und alles ab der Bildbeschreibung symbolisch statt.

Ort	Illumination $i$
Sonne	10 000 cd
Wolken	1000 cd
Arbeitsplatz	100 cd

Tabelle 7.1.: Beispielhafte Werte der Illumination  $i$ .

Material	Reflexion $r$
Schwarzer Samt	0.01
Edelstahl	0.65
Weisse Wand	0.80
Spiegel	0.90
Schnee	0.93

Tabelle 7.2.: Beispielhafte Werte der Reflexion  $r$ .

- Binäre fotoelektrische Zellen  
Der Chip besteht aus einzelnen Fotodioden, wobei die Bildpunkte entweder 1 oder 0 wahrnehmen (hell oder dunkel).
- Zeilen von Fotodioden (Linearkamera)  
Die fotoelektrischen Zellen sind entlang einer Geraden angeordnet.
- Felder von Fotoelementen (Matrixkamera)  
Die Zellen sind zweidimensional Angeordnet.
- CCD-Fernsehkameras  
Fernsehnorm, vorgeschaltete Kameraoptik, ...

## Grundlagen

Die *Lichtintensitätsfunktion*  $f^*$ , die für einen Pixel  $(x, y)$  die Lichtintensität  $[f^*] = \text{cd}$  angibt, kann in ein Produkt

$$f^*(x, y) = i(x, y) \cdot r(x, y)$$

aufgespalten werden mit der Illuminationsfunktion  $i(x, y)$  und der Reflexionsfunktion  $0 < r(x, y) < 1$ . Dabei hat die Illuminationsfunktion die Einheit  $[i] = \text{cd}$  (Candela) und die Reflexion ist einheitenlos. Beispielhafte Werte für die Illumination  $i$  und die Reflexion  $r$  sind in den Tabellen 7.1 und 7.2 gegeben.

Die Absorptionsfunktion

$$a(x, y) := 1 - r(x, y)$$

beschreibt, wie viel Licht an einem Pixel  $(x, y)$  absorbiert wird/wurde.

Zusammengefasst ergibt sich die *Bildmatrix*  $f = (f_{i,j})$

$$f = \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M-1,0} & f_{M-1,1} & \cdots & f_{M-1,N-1} \end{bmatrix}$$

mit den Elementen als Werten der Lichtintensitätsfunktion. Typische Formate  $M \times N$  sind

- PAL:  $N = 768, M = 576$  (576i50)
- NTSC:  $N = 640, M = 480$  (480i60)
- HDTV:  $N = 1280, M = 720$  (720p50, WXGA)
- HDTV:  $N = 1920, M = 1080$  (1080i50, Full HD)

Bei einer monochromen Kamera wird pro Bild eine Matrix verwendet, bei einer polychromen Kamera drei Matrizen. Für diese drei Matrizen können unterschiedliche Farbmodelle verwendet werden, z. B.:

- RGB (Red, Green, Blue)  
Speziell für Monitore relevant, da diese meistens die drei Farben Rot, Grün und Blau anzeigen können.
- HSI/HSV (Hue, Saturation, Intensity/Value)  
Speziell für die Bildverarbeitung, da Segmentierung leicht ist.

Das RGB-Farbmodell mit 8 Bit je Farbe hat damit  $256 \cdot 256 \cdot 256 = 16\,777\,216$  mögliche Farben. Eine alternative zum RGB-Modell ist das HSI/HSV-Modell, welches aus den Komponenten

- Hue (Farbwellenlänge)
- Saturation (Sättigung, Reinheit der Farbe)
- Intensity/Value (Intensität, Helligkeit)

besteht. Das RGB-Modell mit den Anteilen  $R$ ,  $G$  und  $B$  lässt sich wie folgt in das HSI/HSV-Modell umrechnen:

$$\cos H = \frac{2R - G - B}{2\sqrt{(R - G)^2 + (R - B)(G - B)}}$$

$$S = 1 - \frac{3}{R + G + B} \min\{R, G, B\}$$

$$V = I = \frac{1}{3}(R + G + B)$$

Ein Vorteil des HSI/HSV-Modells ist, dass die Farbmanipulation sehr ähnlich zur menschlichen Wahrnehmung ist.

---

## Bildvorverarbeitung

---

Es gibt zwei Klassen von *Bildvorverarbeitungsverfahren*:

### 1. Ortsbereichsverfahren

- Die Manipulation findet direkt auf der Pixelebene statt.
- Die allgemeine Berechnungsformel lautet  $g(x, y) = h(f(x, y))$ , wobei  $g$  die neuen und  $f$  die alten Bildintensitäten darstellt.  $h$  ist ein Operator, der die Intensität transformiert (bspw. durch Nachbarschaftsbeziehungen).
- Aufgrund der geringen Rechenzeit vor allem für Roboter relevant (aufgrund der Echtzeitanforderungen).

### 2. Frequenzbereichsverfahren

- Es wird zunächst eine Bildtransformation (diskrete Fouriertransformation, Wavelets, ...) in den Frequenzraum durchgeführt.
- Anschließend wird ein Verfahren im Frequenzraum angewandt und das Bild anschließend zurück transformiert.
- Wird bspw. bei der Offline-Auswertung von Satellitenbildern eingesetzt.
- Aufgrund der hohen Rechenzeit werden diese Verfahren in der Robotik selten eingesetzt.

**Nachbarschaftsbildverarbeitung** Mit der  $n \times n$ -Nachbarschaft eines Pixels  $(x, y)$  wird die Matrix (bzw. die Pixel)

$$\begin{bmatrix} (x - \lfloor n/2 \rfloor, y - \lfloor n/2 \rfloor) & \cdots & (x - \lfloor n/2 \rfloor, y) & \cdots & (x - \lfloor n/2 \rfloor, y + \lfloor n/2 \rfloor) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ (x, y - \lfloor n/2 \rfloor) & \cdots & (x, y) & \cdots & (x, y + \lfloor n/2 \rfloor) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ (x + \lfloor n/2 \rfloor, y - \lfloor n/2 \rfloor) & \cdots & (x + \lfloor n/2 \rfloor, y) & \cdots & (x + \lfloor n/2 \rfloor, y + \lfloor n/2 \rfloor) \end{bmatrix}$$

bezeichnet, wobei die  $y$ -Achse von links nach rechts und die  $x$ -Achse von oben nach unten wächst. Dies ergibt beispielhaft für die  $3 \times 3$ -Nachbarschaft von  $(x, y)$ :

$$\begin{bmatrix} (x - 1, y - 1) & (x - 1, y) & (x - 1, y + 1) \\ (x, y - 1) & (x, y) & (x, y + 1) \\ (x + 1, y - 1) & (x + 1, y) & (x + 1, y + 1) \end{bmatrix}$$

In der *Nachbarschaftsbildverarbeitung* wird nun ein solcher Bildausschnitt bezüglich eines Pixels  $(x, y)$  betrachtet und z. B. mittels *Nachbarschaftsdurchschnittsbildung* eine neue Intensität des Pixels berechnet. Der folgende Teil beschränkt sich auf  $3 \times 3$ -Nachbarschaften, theoretisch sind alle Verfahren aber auch mit größeren Nachbarschaften möglich.

Bei der Nachbarschaftsdurchschnittsbildung wird jeder Pixels eines Bildausschnitts

$$\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}$$

mit einem Gewicht  $\omega_1, \dots, \omega_9$  (wobei  $\sum_{k=1}^9 \omega_k = 1$  gelten muss) multipliziert, was sogenannte *Gewichtsmaske* darstellt. Diese wird häufig auch durch eine Matrix

$$\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \\ \omega_4 & \omega_5 & \omega_6 \\ \omega_7 & \omega_8 & \omega_9 \end{bmatrix}$$

dargestellt. Die neue Intensität  $g_5$  des zentralen Pixels wird dann durch

$$g_5 = \sum_{k=1}^9 \omega_k f_k$$

oder Allgemein

$$g = \sum_{k=1}^{n^2} \omega_k f_k$$

berechnet.

Beispiele für Nachbarschaftsfilter:

- Mittelwertbildung:

$$\omega = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

- Detektion isolierter Ausreißer (z. B. bei konstant weißem Hintergrund):

$$\omega = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Ein prinzipielles Problem bei Nachbarschaftsdurchschnittsbildung ist, dass scharfe Bilddetails (z. B. Ecken und Kanten) unscharf werden. Eine Abhilfe ist, einen *Median-Filter* zu verwenden. Dabei berechnet sich die neue Intensität  $g$  eines Pixels durch den Medianwert der Intensitäten der Nachbarschaft, d. h. ist Intensitäten werden sortiert und es wird der mittlere Wert (der Median) als neue Intensität genutzt.

---

## Bildverarbeitung

---

Die zentralen Bildverarbeitungsschritte sind *Kantendetektion* (als Voraussetzung für viele Objekterkennungs-algorithmen) und *Segmentierung* (die Zerlegung des Bildes in Komponenten oder Objekte). Dabei werden die zentralen Prinzipien der Unstetigkeiten (zur Kanten- und Konturerkennung) sowie Ähnlichkeit (Schwellwertverfahren, Bereichswachstum) verwendet.

**Kantendetektion** Bei der *Kantendetektion* wird verwendet, dass starke, lokale Änderungen der Intensitätsfunktion häufig Projektionen von Kanten der Szene sind. Allerdings entsprechen nicht alle Änderungen der Intensität von tatsächlichen Objektbegrenzungen und nicht alle Begrenzungen führen zu einer starken Änderung der Intensität. Das allgemeine Vorgehen bei der Kantendetektion besteht aus zwei Schritten:

1. Extraktion von Kandidaten für Kantenelemente
2. Kombination der Elemente zu längeren Strukturen

Die allgemeine Idee ist, die Größe der ersten Ableitung zur Detektion von Kantenelementen und die Größe der zweiten Ableitung zur Interpretation der Lage der Pixel (negative zweite Ableitung → Pixel liegt auf der dunklen, positive zweite Ableitung → Pixel liegt auf der hellen Seite der Kante).

Der Gradient einer kontinuierlichen Funktion, differenzierbaren Funktion  $f(x, y)$  ist

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Problem:  $f(x, y)$ ,  $x$  und  $y$  stammen aus einem diskreten (ganzzahligen) Wertebereich. Daher muss der Gradient durch den Differenzenquotienten angenähert werden. Dabei können entweder die Vorwärtsdifferenzen

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h} \approx f(x+1, y) - f(x, y)$$

oder die zentrale Differenzen

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x-h, y)}{2h} \approx \frac{1}{2}(f(x+1, y) - f(x-1, y))$$

verwendet werden (analog für  $\partial f / \partial y$ ). Aufgrund des Mittlungseffekts eignen sich die zentralen Differenzen hier mehr.

**Prewitt-Operator** Der *Prewitt-Operator* ist Zusammengesetzt auf zwei Filtern: Das Prewitt-X Filter und das Prewitt-Y Filter. Ersteres findet Kanten über die  $x$ -Achse, letzterer über die  $y$ -Achse. Diskretisiert ergeben sich folgende Gewichtsmasken:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (\text{Prewitt-X Filter})$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{Prewitt-Y Filter})$$

Daraus ergeben sich die gefilterten Bilder  $p_x$  (Prewitt-X Filter) und  $p_y$  (Prewitt-Y Filter).

Zusammengesetzt ergibt sich der Prewitt-Operator (bzw. das Graustufenbild)

$$M \approx \sqrt{p_x^2 + p_y^2}$$

**Andere Operatoren** Andere Operatoren zur Kantendetektion sind bspw. der Sobel-Operator und das Roberts-Filter. Zusammengefasst lauten diese:

- Prewitt-Operator:

$$M \approx \sqrt{p_x^2 + p_y^2} \quad p_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad p_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- Sobel-Operator:

$$M \approx \sqrt{p_x^2 + p_y^2} \quad p_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad p_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Roberts-Filter:

$$R = |r_x| + |r_y| \quad r_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad r_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

## Konturerkennung

**Lokale Analyse** Es wird die lokale Nachbarschaft eines Pixels  $(x, y)$  untersucht. Außerdem werden regelbasierte Verbindungen von Kandidaten für Kantenelemente durchgeführt:

- Wenn  $|\nabla f(x, y) - \nabla f(\tilde{x}, \tilde{y})| \leq T$ ,  $T > 0$  (die Gradienten sind Betragsmäßig ähnlich) und  $|\alpha(x, y) - \alpha(\tilde{x}, \tilde{y})| < A$ ,  $A > 0$  (die Winkel der Gradientenvektoren sind ähnlich), dann:
- Verbinde  $(x, y)$  mit  $(\tilde{x}, \tilde{y})$  und kennzeichne diese Verbindung (z. B. durch einen bestimmten Grauwert/eine bestimmte Farbe oder ein Symbol).

Dabei ist

$$\alpha(x, y) := \arctan \frac{\partial f / \partial y}{\partial f / \partial x}$$

der Winkel des Gradientenvektors.

**Globale Analyse und Hough-Transformation** Die globale Analyse geschieht durch eine sogenannte *Hough-Transformation*. Dabei werden mehrere Punkte verbunden, sofern diese auf einer vorgegebenen Kurve (z. B. eine Gerade oder eine Parabel) liegen. Im folgende wird Beispielhaft die Hough-Transformation bzgl. einer Geraden betrachtet, prinzipiell kann sie aber für jede durch eine Gleichung

$$g(x, y, c_1, c_2, c_3) = 0$$

gegebene Kurve durchgeführt werden (z. B. für einen Kreis  $(x - c_1)^2 + (y - c_2)^2 - c_3^2 = 0$  mit Radius  $c_3$ ).

Liegen Punkte  $(x_i, y_i)$  auf einer Geraden, so erfüllen diese die Gleichung  $y_i = ax_i + b$  mit den Parametern  $a, b$ . Diese Gleichung kann mit

$$y_i = ax_i + b \quad \Longleftrightarrow \quad b = -x_i a + y_i$$

zu einer Geradengleichung im Parameterraum  $(a, b)$  umgeformt werden. Bei der Hough-Transformation wird nun jeder Punkt  $(x_i, y_i)$  einer Geraden in der  $(a, b)$ -Ebene zugeordnet. Liegen zwei Punkte  $(x_i, y_i)$  und  $(x_j, y_j)$  auf einer Geraden, so schneiden sich die Hough-Transformierten in einem Punkt. Dieser Schnittpunkt  $(a^*, b^*)$  entspricht den Parameter der Geraden, auf der die zwei Punkte liegen, d. h. es gilt:

$$y_i = a^* x_i + b^* \quad \text{und} \quad y_j = a^* x_j + b^*$$

Schneiden sich drei Geraden in einem Punkt, so liegen alle entsprechenden Punkte auf einer Geraden.

In der Praxis ist die Darstellung  $y = ax + b$  der Geraden problematisch, da vertikale Geraden nicht dargestellt werden. Es wird daher eher die Parametrisierung  $x \cos \theta + y \sin \theta = \rho$  verwendet, da hier das Problem nicht auftritt.

**Segmentierung** Ziel der *Segmentierung* ist, das Bild  $B$  in flächige *Zusammenhangskomponenten*  $S_1, \dots, S_n$  zu zerlegen, wobei diese paarweise disjunkt sind und  $S_1 \cup \dots \cup S_n = B$  gilt. Dazu müssen die Grenzen von Objektflächen gefunden werden, die für die weitere Interpretation von Bedeutung sind.

**Schwelldwertabfrage** Dieses Verfahren wird häufig verwendet und ist insbesondere im HSI/HSV-Farbraum sehr nützlich. Bezogen auf die Intensität  $f(x, y)$  werden zwei Schwellwerte  $\tau_1, \tau_2$  festgelegt, wodurch sich über

$$g(x, y) := \begin{cases} 0 & \text{falls } f(x, y) < \tau_1 \\ 1 & \text{falls } \tau_1 < f(x, y) < \tau_2 \\ 2 & \text{falls } \tau_2 < f(x, y) \end{cases}$$

ein trinäres Schwellwertbild ergibt.

Dieses Verfahren wirft allerdings Probleme auf, da die Intensitätsfunktion stark von der aktuellen Beleuchtung abhängig ist, welche von Bild zu Bild variieren kann. Eine automatische Adaption der Schwellwerte erfordert eine robuste Minimum-Suche, wobei die Minima abhängig sind von Beleuchtung, Lager der Objekte, ....

## Bereichswachstum

- Es wird mit einem oder mehreren geeigneten „Saat“-Punkten („seed points“) gestartet und die direkten Nachbarpixel überprüft.
- Ein untersuchter Pixel wird zu dem Bereich des Startpixels hinzugefügt, wenn dieser dem Startpixel ähnlich ist (z. B. gleicher Grauwert, Farbe, Textur, ...).
- Unterscheidet sich der Pixel stark, so wird ein neuer Bereich eröffnet.



- Anschließend wird das Bild rekursiv weiter untersucht.
- Weitere Operationen können z. B. das Aufteilen und Verbindungen von Bereichen sein.

---

## Merkmalsextraktion

---

Bisher wurden Objekte bildhaft durch diskrete Bildfunktionen beschrieben. Die *Merkmalsextraktion* beschäftigt sich nun mit der Charakterisierung von Objekten durch einen Vektor von Merkmalen sowie eventuell Beziehungen zwischen Objekten (Relationstupel) wie Nachbarschaft und Größe. Ausgangspunkt ist dabei das durch Segmentierung und Bildvorverarbeitung entstandene Binärbild  $g(x, y)$ . Ziel ist die Extraktion der Merkmale, die ein Segment beschreiben.

Merkmale können z. B. sein:

- $m_1$ : Objektumfang  $m_1 = \int_{\text{Kontur}} g(x, y) \, ds \approx \sum_{\text{Kontur}} g(x, y)$
- $m_2$ : Objektfläche  $m_2 = \iint g(x, y) \, dx \, dy \approx \sum_x \sum_y g(x, y)$
- $m_3$ : Anzahl der „Löcher“
- $m_4$ : Farbe
- ...

Solche Merkmale sollten möglichst (weitgehend)

- invariant gegen Skalierung, Rotation sowie Translation,
- robust (gegen Rauschen), signifikant, diskriminierend (unterscheidbar) sowie
- schnell berechenbar sein.

---

## Objektklassifikation

---

- Klassische Ansätze zur Objektklassifikation:
  - Bildvergleich im Merkmalsraum
  - Template Matching
- In den letzten Jahren wurden große Fortschritte durch Deep Learning erzielt.
- Deep Learning Systeme scheitern allerdings noch an einem echten Verstehen der Objekte.

---

## Bildverstehen

---

---

### 7.2.3. 3D-Sensoren und Perzeption

---

Bei einem einzelnen RGB-Pixel eines 2D-Farbbildes fließen viele Faktoren ein:

- Lichtquelle (Anzahl, Helligkeit, Farbspektrum)
- Objekte (Farbe, physikalische Eigenschaften wie Transparenz und Reflexion)
- Szene (Schattenwurf)

- Projektion (Perspektivische Verzerrung)
- Sensorqualität

Ein Mensch ist extrem gut darin, aus einem einzelnen Bild Tiefeninformationen zu extrahieren, wobei dies im Bereich Computer Vision noch immer eine große Herausforderung ist.

Werden zwei Farbbilder (leicht versetzt) verwendet, so wird dies als *Stereo Vision* (oder in der Biologie auch „binokulares Sehen“) bezeichnet. Dies erlaubt eine passive Tiefenschätzung. Andere Spezies wie Fledermäuse haben sogar eine aktive Tiefenschätzung (Sonar).

Distanzinformationen sind extrem wichtig:

- 3D-Navigation
- Interaktion
- Kollisionsvermeidung
- Objekterkennung
- Gestenerkennung
- Virtual/Augmented Reality
- ...

---

## Bildbasierte 3D-Sensoren

---

**Stereo Vision** *Stereo Vision* basiert auf der Triangulation von Bildpunkten durch die Verwendung von zwei Bildern.

Basierend auf Abbildung 7.4 kann die Distanz  $Z$  wie folgt berechnet werden:

$$\frac{Z - f}{B - (x - x')} = \frac{Z}{B} \quad \Longleftrightarrow \quad BZ - Bf = BZ - Z(x - x') \quad \Longleftrightarrow \quad Z = \frac{Bf}{x - x'}$$

Diese Art der Berechnung wirft das Problem auf, dass detektiert werden muss, welche Pixel das selbe Objekt abbilden.

- **Vorteile:**
  - Nur zwei Kameras nötig.
  - Entspricht dem menschlichen Vorbild (binokulares Sehen).
  - Farbinformationen sind für jeden Punkt verfügbar.
  - Hohe Bilddraten möglich.
- **Nachteile:**
  - Benötigt komplizierte Algorithmen, um die Pixel zuzuordnen.
  - Erfordert viel CPU Leistung.
  - Angewiesen auf texturierte Oberflächen (zur Pixelzuordnung).
  - Sensitiv gegenüber Hintergrundbeleuchtung.

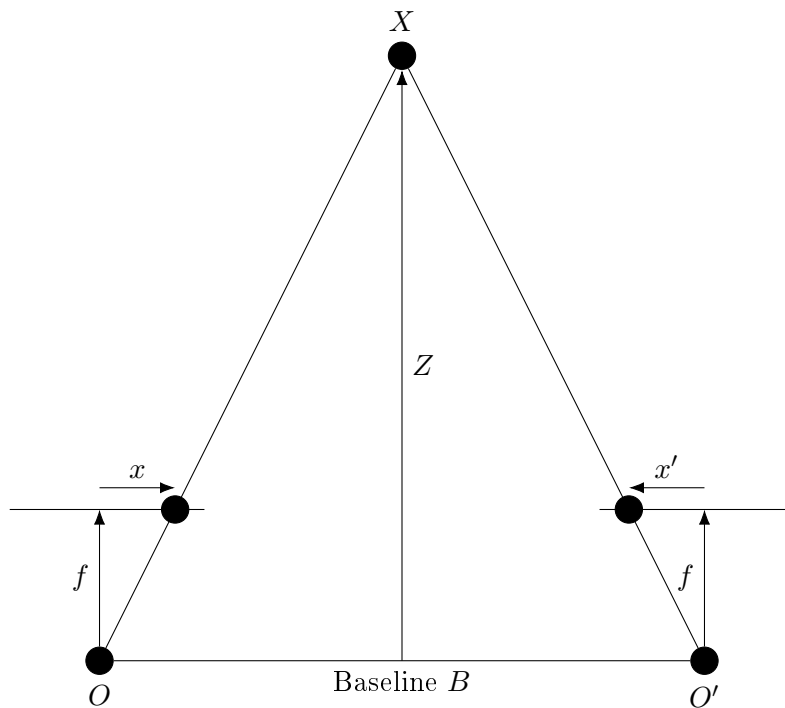


Abbildung 7.4.: Triangulation der Distanz  $Z$  eines Objekts  $X$  durch zwei Kameras  $O$  und  $O'$ .

## Structured Light Kamera

### Stereo Vision + Structured Light

**Time-of-Flight (ToF) Kamera** Eine *Time-of-Flight* (ToF) Kamera sendet Lichtimpulse (meist Infrarot) und misst die Reflexionen. Hierzu gibt es drei grundlegende Techniken:

- Direct Time-of-Flight Imagers/Flash Lidar
  - Direkte Zeitmessung der Lichtstrecke
  - Benötigt starke Signalfanken (z. B. Laserdioden)
- Carrier Wave Modulation (Phasenvergleichsverfahren)
  - Es wird die Phasenverschiebung des reflektierten Lichtsignals gemessen.
- Ranged Gated Imagers („Shutter“)
  - Er Shutter öffnet sich in der gleichen Frequenz, in der die Lichtimpulse ausgesendet werden.
  - Ein reflektierter Impuls wird teilweise vom Shutter blockiert, was einen Rückschluss auf die zurückgelegte Lichtstrecke zulässt.
- Vorteile:
  - Hohe Bildrate (bis zu 160 FPS üblich), daher geeignet für Echtzeitanwendungen
  - Sehr wenig CPU Leistung nötig.
  - Musterunabhängig, d. h. es werden keine strukturierten/texturierten Oberflächen benötigt.

---

- **Nachteile:**

- Geringe Auflösung, es sind nur große Pixel realisierbar.
- Sensitiv gegenüber Hintergrundbeleuchtung.
- Interferenzen zwischen mehreren ToF Kameras.
- Mehrfachreflexionen durch großflächige Bestrahlung von Oberflächen.

---

### **Laserbasierte 3D-Sensoren**

---

Es ist auch möglich, Tiefeninformationen ohne Kamerabilder zu generieren. Ein Beispiel hierfür sind Laserscanner. Diese können aus mehreren Lasern (Mehrfach-Lasersysteme) zusammengesetzt sein oder dreh-/schwenkbar sein (Mono-Lasersystem). Es ist ebenfalls möglich, Laserscanner und Kamerabilder zu kombinieren, um ein detaillierteres Weltbild zu erhalten.

---

### **Datenstrukturen und -Repräsentation**

---

Eine klassische, unkomprimierte Punktwolke („point cloud“) benötigt sehr viel Speicherplatz, d. h. es werden effektiver Datenrepräsentationen/-strukturen benötigt:

- Reduktion des Speicherbedarfs.
- Reduktion des Rechenaufwands bei der Verarbeitung.
- Effiziente Aggregation von 3D-Punkten.
- Effizienter Zugriff auf einzelne Punkte (z. B. Nachbarschaftsinformationen).

**KD-Baum** Ein *KD-Baum* ist eine verallgemeinerte Version eines Binärbaums (KD steht für  $k$ -dimensional). Die geometrische Idee ist eine rekursive Aufteilung der Daten mit Hyperebenen.

- **Vorteile:**

- Anwendung der „Branch-and-Bound“ Methode: Die Lage des Suchpunktes relativ zum aktuell betrachteten Knoten des Baums verrät, in welchem Unterteilbaum ein potentiell besserer (näherer) Knoten liegt.
- Es gibt Algorithmen, die  $\mathcal{O}(\log n)$  garantieren.
- Einfach erweiterbar auf  $k$ -nearest Search.

- **Nachteile:**

- Das Einfügen und Entfernen von Punkten ist kompliziert (ggf. Rebalancierung des Baums notwendig).
- Keine Datenkompression und Aggregation.

---

**Octree** Ein *Octree* unterteilt den Raum rekursiv-hierarchisch in *Oktanten*. Er ist dabei ein gewurzelter Baum, indem jeder Knoten entweder acht oder keine Nachfolger hat.

- **Vorteile:**

- Leicht zu Implementieren.
- Geringer Speicherplatzbedarf bei dünnbesetzten Punktwolken.
- Mehrere Punkte können in einem Oktanten zusammengefasst werden.
- Einfaches Einfügen und Entfernen von Punkten.
- Effizienter Zugriff, da die Baumtiefe beschränkt ist.
- Raycasting-Algorithmen effizient umsetzbar.

- **Nachteile:**

- Ineffiziente Nachbarschaftssuche.
- Kubische Datenrepräsentation vs. Längliche Umweltgeometrien.
- Auflösung beschränkt, da die Umwelt in ein festes Voxelgitter eingeteilt wird.

### **Truncated Signed Distance Fields (TSDF)**

- Truncated Signed Distance Fields (TSDFs) werden als Voxelgitter gespeichert.
- Jede Zelle speichert dabei den Abstand zur nächsten Oberfläche.
- Zellen mit einem Wert nahe Null repräsentieren die *Isolinie* der Oberfläche.
- Zellen fern von Oberflächen werden „gelöscht“ (truncated).
- **Vorteile:**
  - Oberflächen können annähernd beliebig genau erfasst werden.
  - Mesh-Rekonstruktion ist „inklusive“.
- **Nachteile:**
  - Voxelgitter benötigen enorm viel Speicherplatz (Lösung: Hashed TSDF).
  - Updates sind rechenintensiv (häufig werden diese auf der GPU berechnet).

---

### **Point-Cloud Processing**

---

*Point Cloud Library* (PCL) eine umfangreiche Bibliothek zur Verarbeitung von 3D-Daten und ist die meist genutzte Bibliothek zur 3D-Datenverarbeitung (analog zu OpenCV in der 2D-Bildverarbeitung).

- Datenaufbesserung: Statistical Outlier Removal
  - Viele Verfahren benötigen dichte Punktwolken.
  - Daher ist eine Entfernung von Ausreißern notwendig.
  - Analyse des mittleren Abstands zu den  $k$ -nächsten Nachbarn: Wird der Abstand zu groß, wird der Punkt entfernt.

- Datenaufbesserung: Glättung (Smoothing)
  - Oftmals sind 3D-Daten verrauscht, in Gebäuden gibt es jedoch oftmals glatte Oberflächen.
  - Polynomial Reconstruction: Lokale Approximation mit Polynomen  $n$ -ter Ordnung approximiert.
  - Resampling: Das Polynom wird verwendet, um den Punkt auf das ermittelte Polygon zu verschieben.
  - Durch dieses Verfahren können auch Oberflächen rekonstruiert werden.
- Informationsextraktion: Surface Normal Estimation
  - Oberflächenkonstruktion ist häufig auf Oberflächennormale angewiesen.
  - Eine Schätzung ist bspw. mittels Principal Component Analysis (PCA) möglich.
  - Durch eine Singulärwertzerlegung werden die Eigenvektoren der Punkte gefunden.
  - Der kleinste Eigenvektor entspricht dann der geschätzten Oberflächennormale.
- Informationsextraktion: Plane Segmentation

---

## Point-Cloud Registration

---

Iterative Closest Point (ICP):

- Bestimmt den Versatz zweier Teilmengen von Punktwolken.
- Ansatz: Ist die korrekte Übereinstimmung der Punkte bekannt, so lässt sich die relative Transformation bestimmen.
- Dabei wird angenommen, dass die jeweils zueinander nächsten Punkte zueinander gehören (korrpondieren).
- Dann wird eine Transformation gesucht, die den quadratischen Fehler minimiert.
- Anschließend wird die Eingangspunktwolke basierend auf der Transformation verschoben.
- Dies wird so lange wiederholt, bis der Fehler konvergiert.

---

## 8. Regelung

---

### 8.1. Lineare Regelung

---

Bei einer „normalen“ Steuerung (Abbildung 8.1) findet keine Rückkopplung der tatsächlich durchgeführten Aktion statt, während bei einer Regelung (Abbildung 8.2) die Aktion gemessen wird und in das nächste Steuerungssignal eingerechnet wird.

#### 8.1.1. Begriffe

---

Eine Steuerung wird auch als Open-Loop oder Feedforward Control bezeichnet, eine Regelung als Closed-Loop oder Feedback Control.

Im folgenden wird folgende Notation verwendet:

- Sollwert:  $x_d(t)$
- Regelgröße:  $x(t)$
- Messgröße:  $\tilde{x}(t)$
- Regelfehler:  $e(t)$
- Stellgröße:  $u(t)$
- Störung:  $s(t)$

Abbildung 8.3 zeigt einen normalen Regelkreis mit den entsprechenden Größen.

Des weiteren seien folgende Begriffe definiert:

- *Regelungstechnik*: Lehre der
  - mathematischen Beschreibung und
  - gezielten Beeinflussung
  - dynamischer Prozesse

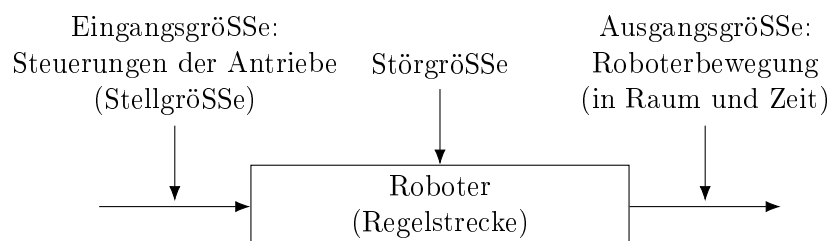


Abbildung 8.1.: Blockschaltbild einer normalen Steuerung (Feedforward Control).

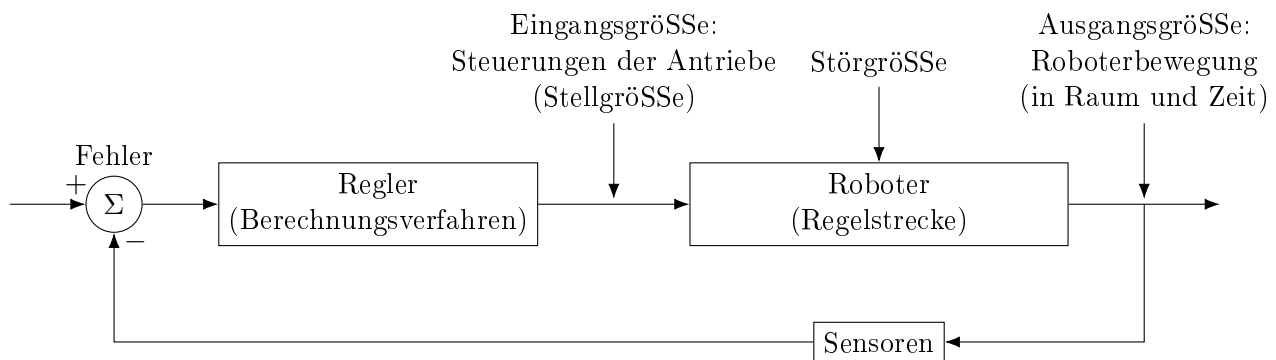


Abbildung 8.2.: Blockschaltbild einer Regelung (Feedback Control).

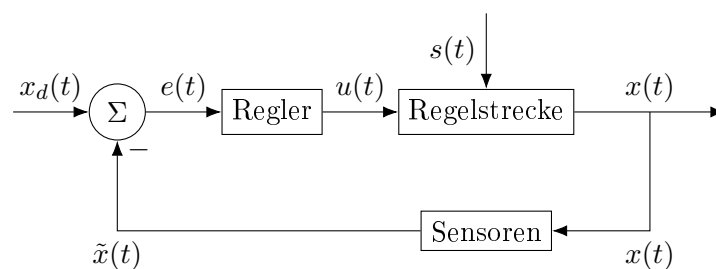


Abbildung 8.3.: Die dynamischen Größen eines Regelkreises.

- in Technik, Biologie, Ökologie, Soziologie, ...
- *Aufgabe der Regelung*
  - Konstant halten oder gezieltes Ändern
  - von einer oder mehreren Regel(ungs)größen
  - unter Einwirkung von Störgrößen auf einen Prozess.
- *Führungsgröße (Soll-Wert)*
  - Wenn konstant  $\rightarrow$  *Festwertregelung*
  - Wenn veränderlich (zeit- und/oder zustandsabhängig)  $\rightarrow$  *Folgeregelung*
- *Regeln*
  - Herstellen und Bewahren einer wünschenswerten Situation.
- *Steuerung*
  - Nachstellen einer *Stellgröße* nach gegebenen Kennlinien (Open-Loop Control, Feedforward Control).
- *Regelung*
  - Nachstellen aufgrund laufender Messungen der Regelgrößen (Closed-Loop Control, Feedback Control).
- *Vermaschte Regelung*



- Mindestens zwei sich wechselseitig beeinflussende Regelkreise.
- *Kaskadenregelung*
  - (Hierarchisch) überlagerte Regelkreise.
- *Stabilitätsproblem*
  - Eine sprungartige Verstellung des Sollwerts führt zu einem verzögertem Einschwingvorgang der Regelgröße und einer verzögerten Ankunft des Korrektursignals.
  - Eine kleine Regelabweichung soll ausreichen, um die Stellgröße zu steuern (über Kreisverstärkung in einem geschlossenem Regelkreis).
  - Zeitverzug und Signalverstärkung führen zu Instabilität (d. h. Schwingung um den Sollwert).

---

### 8.1.2. Lineare Systemdynamik und Feder-Masse-System

---

Im folgenden wird ein Feder-Masse-System mit der Masse  $m > 0$ , der Federsteifigkeit  $k > 0$  und dem Reibungskoeffizienten  $b > 0$  sowie einer Antriebs- oder Bremskraft  $f(t)$  betrachtet. Der Wert  $x$  beschreibt dabei die lineare Verschiebung der Masse bzgl. der Ruhelage um  $x_{\text{stat}}(t) = 0$ . Die Bewegungsgleichung lautet

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = f(t)$$

und ist eine lineare, inhomogene, gewöhnliche Differentialgleichung zweiter Ordnung mit konstanten Koeffizienten.

---

#### Untersuchung des Bewegungsverhaltens

---

Mit einer gegebenen Anfangsposition  $x(0)$  und einer gegebenen Anfangsgeschwindigkeit  $\dot{x}(0)$  sowie einem Antriebskraftverlauf  $f(t)$  kann der zeitliche Bewegungsverlauf der Masse aus der Bewegungsgleichung bestimmt werden.

**Erwartetes Bewegungsverhalten** Wird angenommen, dass  $f(t) = 0$  gilt sowie  $x_{\text{stat}}(t) = 0$  die Ruhelage darstellt, so lassen sich folgende Bewegungsmuster erkennen:

- Schwache Feder ( $k$  klein), große Reibungskraft ( $b$  groß): Die Masse wird aus einer Anfangsauslenkung  $x(0) \neq x_{\text{stat}}$  langsam in den stationären Zustand zurückkehren.
- Starke Feder ( $k$  groß), schwache Reibungskraft ( $b$  klein): Die Masse wird aus einer Anfangsauslenkung  $x(0) \neq x_{\text{stat}}$  mehrmals schnell hin und her schwingen (oszillieren), bis sie in den stationären Zustand zurückkehrt.

Unter der Annahme  $f(t) = 0$  kann die Differentialgleichung mit dem Lösungsansatz  $x(t) = ce^{\lambda t}$  explizit gelöst werden, wodurch sich das charakteristische Polynom

$$\lambda^2 + \frac{b}{m}\lambda + \frac{k}{m} = 0$$

mit den Nullstellen (den *Polen*)

$$\lambda_{1,2} = -\frac{b}{2m} \pm \frac{\sqrt{b^2 - 4mk}}{2m}$$

ergibt. Diese Pole bestimmen das Lösungsverhalten und damit das Bewegungsverhalten der Masse.

---

**Überkritisch Gedämpftes Verhalten**  $\lambda_{1,2}$  sind einfache, reelle Nullstellen, wenn

$$b^2 - 4mk > 0 \quad \Longleftrightarrow \quad b^2 > 4mk$$

wodurch auch  $\lambda_{1,2} < 0$  gilt. Dieser Fall wird *überkritisch gedämpft* genannt.

Die Lösung ist in diesem Fall gegeben durch

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$$

wobei die Konstanten  $c_1, c_2$  durch die Anfangsbedingungen festgelegt werden.

**Kritisch Gedämpftes Verhalten**  $\lambda$  ist eine doppelte, reelle Nullstelle, wenn

$$b^2 - 4mk = 0 \quad \Longleftrightarrow \quad b^2 = 4mk$$

wodurch auch  $\lambda < 0$  gilt. Dieser Fall wird *kritisch gedämpft* genannt.

Die Lösung ist in diesem Fall gegeben durch

$$x(t) = c_1 e^{\lambda t} + c_2 t e^{\lambda t}$$

wobei die Konstanten  $c_1, c_2$  durch die Anfangsbedingungen festgelegt werden.

**Unterkritisch Gedämpftes Verhalten**  $\lambda_{1,2}$  sind einfache, komplexe Nullstellen (mit  $\lambda_1 = \bar{\lambda}_2$ ), wenn

$$b^2 - 4mk < 0 \quad \Longleftrightarrow \quad b^2 < 4mk$$

wodurch mit  $\lambda_{1,2} =: \lambda_r \pm i\lambda_i$  auch  $\lambda_r < 0$  gilt. Dieser Fall wird *unterkritisch gedämpft* genannt.

Die Lösung ist in diesem Fall gegeben durch

$$c_1 e^{\lambda_r t} \cos(\lambda_i t) + c_2 e^{\lambda_r t} \sin(\lambda_i t)$$

wobei die Konstanten  $c_1, c_2$  durch die Anfangsbedingungen festgelegt werden.

**Gewünschtes Bewegungsverhalten** Wünschenswert ist oftmals die kritische Dämpfung, da

- das System dabei aus einer Anfangsauslenkung  $x(0) \neq x_{\text{stat}}$  schnellstmöglich in die stationäre Lage zurückkehrt und
- keine Oszillationen oder Überschwingungen auftreten.

In der Regelungstechnik wird üblicherweise nicht im Zeitbereich gearbeitet, sondern die Laplace-Transformation zur Definition von Übertragungsfunktionen und der Untersuchung der Pole eingesetzt. Dies wird hier aber nicht weiter behandelt...

---

### 8.1.3. PD-Regelung linearer Systeme 2. Ordnung

---

Die Aufgabe der Regelung ist nun, ein gewünschtes Bewegungsverhalten durch geeignete Wahl der Steuerung zu erreichen.

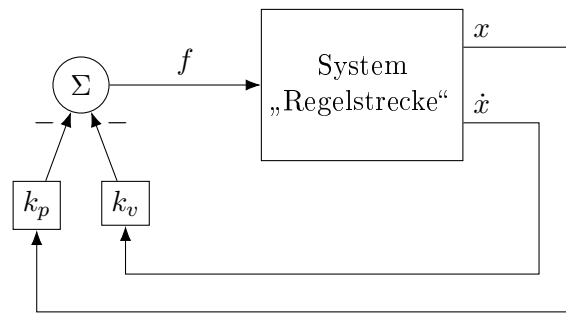


Abbildung 8.4.: Blockschaltbild einer PD-Regelung mit den Soll-Werten  $x_d, \dot{x}_d$ .

### Beispiel: Feder-Masse-System

Mit Hilfe von Sensoren zur Messung der Position  $x$  sowie der Geschwindigkeit  $\dot{x}$  und der Verwendung einer Antriebskraft  $f$  (Steuerung) und einem geeigneten Regelgesetz zur Bestimmung von  $f$  soll das Bewegungsverhalten so modifiziert werden, dass eine kritische Dämpfung eintritt. Der Ansatz für die erzeugte Kraft ist dabei die Funktion

$$f(t) = -k_v \dot{x}(t) - k_p x(t)$$

mit den Regelparametern/Verstärkungsfaktoren  $k_v$  (Differentialanteil) und  $k_p$  (Proportionalanteil). Abbildung 8.4 zeigt das Blockschaltbild für eine solche Regelung.

Wird das Regelgesetz auf die Bewegungsgleichung angewandt, so ergibt sich:

$$\begin{aligned} m\ddot{x}(t) + b\dot{x}(t) + kx(t) &= f(t) \\ \iff m\ddot{x}(t) + b\dot{x}(t) + kx(t) &= -k_v \dot{x}(t) - k_p x(t) \\ \iff m\ddot{x}(t) + \underbrace{(b + k_v)}_{\hat{b}} \dot{x}(t) + \underbrace{(k + k_p)}_{\hat{k}} x(t) &= 0 \end{aligned}$$

Die Pole des geregelten Systems entsprechen den Nullstellen des charakteristischen Polynoms:

$$\lambda_{1,2} = -\frac{\hat{b}}{2m} \pm \frac{\sqrt{\hat{b}^2 - 4m\hat{k}}}{2m}$$

Durch geeignete Wahl der Regelparameter  $k_v, k_p$  kann also das gewünschte Bewegungsverhalten der kritischen Dämpfung erreicht werden.

Neben der Bedingung  $\hat{b}^2 = 2m\hat{k}$  muss (da diese noch nicht ausreichend ist für eine kritische Dämpfung, da  $k_v$  und  $k_p$  positiv oder negativ gewählt werden können) unter anderem  $\hat{b} > 0$  sichergestellt werden, da Regelfehler sonst die Instabilität verstärken statt dämpfen würde.

- Auf diese Weise kann das System auf einen konstanten Sollwert  $x_d$  geregelt werden (Festpunktregelung).
- Analog dazu kann die Regelung auch auf eine gegebene Sollwerttrajektorie  $x_d(t)$  erweitert werden.
- Ebenfalls kann das System auf mehrdimensionale und nichtlineare Systeme erweitert werden.

**Gewünschte Polplatzierung** Damit das geregelte System stabil ist, müssen die Pole des Systems in der linken Hälfte der komplexen Ebene liegen (d. h. der Realteil muss negativ sein).

---

### 8.1.4. Partitionierung des Regelgesetzes durch Feedback-Linearisierung

---

In diesem Abschnitt wird wieder das Feder-Masse-System

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = f(t)$$

mit der Regelkraft  $f(t)$  betrachtet.

Bei der Partitionierung des Regelgesetzes wird dieser in zwei Teile aufgebrochen:

1. Den („inneren“) *modellbasierten Anteil*, der die Kenntnis über die Systemdynamik ausnutzt und
2. den („äußeren“) *Servo-Anteil*, welcher die Servomesswerte zurück koppelt zur Anpassung des dynamischen Systemverhaltens.

Durch (1) wird die Systemdynamik transformiert, sodass der Entwurf von (2) sehr viel einfacher wird.

Für den modellbasierten Regleranteil wird der Ansatz

$$f = \alpha \hat{f} + \beta$$

betrachtet, wobei  $\alpha$  und  $\beta$  Funktionen sind, die möglichst so zu wählen sind, dass die Systemdynamik für den Servo-Anteil  $\hat{f}$  in einem System mit Einheitsmasse und ohne Feder und Dämpfer resultiert, d. h. es soll

$$\ddot{x} = \hat{f}$$

gelten. Dieses Vorgehen wird *Feedback-Linearisierung* genannt. Für das Feder-Masse-Dämpfer-System ergibt sich über die Beziehung

$$\underbrace{m}_{\alpha:=} \ddot{x} + \underbrace{b\dot{x} + kx}_{\beta:=} = \alpha \hat{f} + \beta$$

der modellbasierte Anteil des Reglers:

$$f = m\hat{f} + b\dot{x} + kx$$

Der Servo-Anteil wird nun als PD-Regler für das sehr einfache System  $\ddot{x} = \hat{f}$  bestimmt:

$$\hat{f} = -k_p x - k_v \dot{x}$$

Durch Anwendung des Regelgesetzes ergibt sich folgende Bedingung für eine kritische Dämpfung:

$$k_v = 2\sqrt{k_p}$$

Durch dieses Vorgehen können viele komplizierte Systemdynamiken linearisiert werden.

**Beispiel** Sei die nichtlineare Systemdynamik mit Regelkraft  $f$  gegeben:

$$m\ddot{x} + kx^2 = f$$

Es wird wieder der Ansatz

$$f = \alpha \hat{f} + \beta$$

betrachtet, wobei der Servo-Anteil  $\hat{f}$  für das einfache System  $\ddot{x} = \hat{f}$  eingesetzt werden soll. Ein Einsetzen dieser Gleichung ergibt:

$$m\ddot{x} + kx^2 = \alpha \hat{f} + \beta \quad \implies \quad \alpha = m, \beta = kx^2$$

Damit lautet das modellbasierte Regelgesetz

$$f = m\hat{f} + kx^2$$

und für den Servo-Anteil kann wieder ein einfach PD-Regler

$$\hat{f} = -k_p x - k_v \dot{x}$$

mit der Bedingung  $k_v = 2\sqrt{k_p}$  für eine „kritische Dämpfung“ verwendet werden.

---

### 8.1.5. Sollwerttrajektorien-Folgeregelung

---

Im Gegensatz für Festwertregelung soll bei der Sollwerttrajektorien-Folgeregelung eine Trajektorie  $x_d(t)$ ,  $\dot{x}_d(t)$ ,  $\ddot{x}_d(t)$  verfolgt werden. Sei die Systemdynamik wie in 8.1.4 auf das System

$$\ddot{x} = \hat{f}$$

transformiert. Mit dem Ansatz

$$\hat{f} = \ddot{x}_d + k_v \underbrace{(\dot{x}_d - \dot{x})}_{\dot{e}:=} + k_p \underbrace{(x_d - x)}_{e:=}$$

für das Regelgesetz lautet die Systemdynamik:

$$\ddot{x} = \ddot{x}_d + k_v(\dot{x}_d - \dot{x}) + k_p(x_d - x) \quad \Longleftrightarrow \quad 0 = \ddot{e} + k_v\dot{e} + k_p e$$

was einer linearen, homogenen Differentialgleichung im *Fehlerraum* entspricht. Dieses System kann nun wieder als Feder-Masse-Dämpfer-System betrachtet werden. Somit ist z. B. eine kritische Dämpfung durch die

$$k_v = 2\sqrt{k_p}$$

gegeben, sodass der Anfangsfehler  $e(0) \neq 0$  möglichst schnell abklingt.

---

### 8.1.6. PID-Regelung linearer Systeme

---

Mit den Sollwerten  $x_d, \dot{x}_d$  für Position und Geschwindigkeit lautet das PID-Regelgesetz

$$f = k_v \underbrace{(\dot{x}_d - \dot{x})}_{\dot{e}} + k_p \underbrace{(x_d - x)}_e + k_i \int \underbrace{(x_d - x)}_e dt$$

wobei  $e := x_d - x$  und  $\dot{e} := \dot{x}_d - \dot{x}$  den Positions- und Geschwindigkeitsfehler darstellen. Das Blockschaltbild der PID-Regelung ist in Abbildung 8.5 gegeben. Die Wirkung der drei Terme (Differential, Proportional und Integral) haben folgende intuitive Effekte:

- Der Differentialteil „schaut in die Zukunft“ und führt zu einer vorausschauenden Regelung,
- der Proportionalteil „schaut auf die Gegenwart“ und führt zu einer akuten Regelung und
- der Integralteil „schaut in die Vergangenheit“ und korrigiert (regelt) Fehler raus, die in der Vergangenheit gemacht wurden.

---

### Windup-Effekt

---

Üblicherweise ist die maximale Stellleistung endlich, d. h. jede Stellgröße  $f$  ist auf einen maximalen Wertebereich  $[-f_{\max}, f_{\max}]$  begrenzt.

Problem: Der I-Anteil des Reglers integriert auch nach Erreichen der Sättigung der Stellgröße die Regelabweichung weiter auf, obwohl der Reglerausgang keinen Einfluss mehr auf die Regelstrecke hat. Beim Erreichen der Zielvorgabe kann  $|f|$  dadurch deutlich größer als  $f_{\max}$  sein, wodurch dieser Überschuss zunächst wieder abgebaut werden muss. Dieser Effekt wird als *Windup-Effekt* bezeichnet und verschlechtert das Regelverhalten (Überschwingen).

Zur „Schadensbegrenzung“ müssen Anti-Windup-Maßnahmen eingesetzt werden, bspw. die Begrenzung des I-Anteils bei Auftreten der Sättigung. Dazu wird die Differenz  $\tilde{f} - f$  zwischen Stellgröße  $f$  und begrenzter Stellgröße  $\tilde{f}$  über ein P-Glied mit Verstärkung  $k_{aw}$  zurück gekoppelt. Diese Rückkopplung hat keinen Einfluss, wenn  $f \in [-f_{\max}, f_{\max}]$  gilt. Abbildung 8.6 zeigt das Blockschaltbild einer solchen Anti-Windup-Strategie.

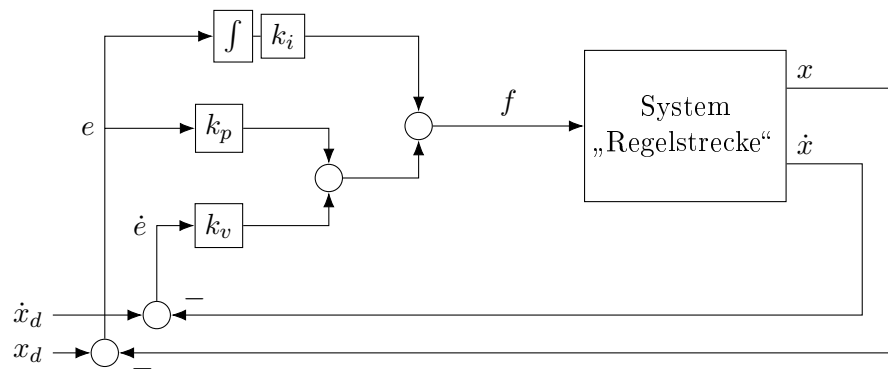


Abbildung 8.5.: Blockschaltbild eines PID-Reglers mit den Sollwerten  $x_d, \dot{x}_d$ .

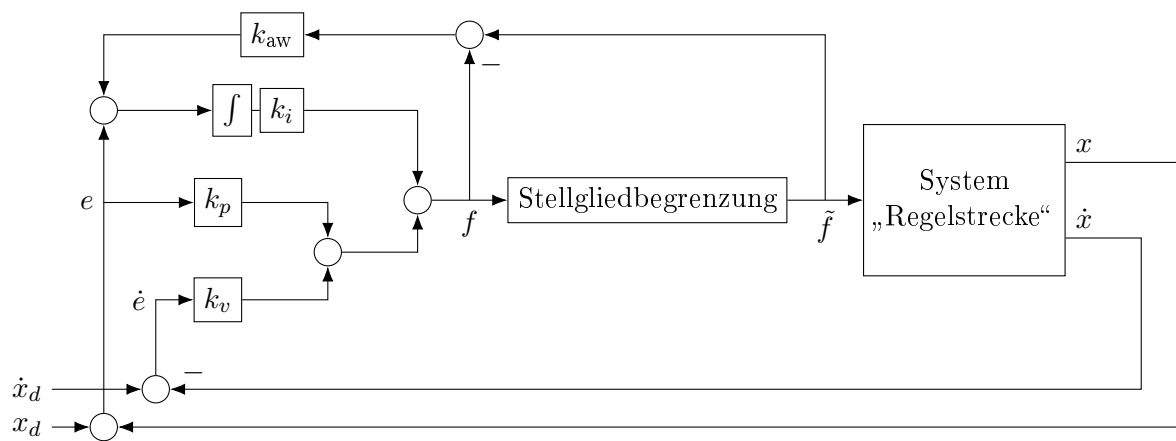


Abbildung 8.6.: Blockschaltbild eines PID-Reglers mit Anti-Windup-Teil.

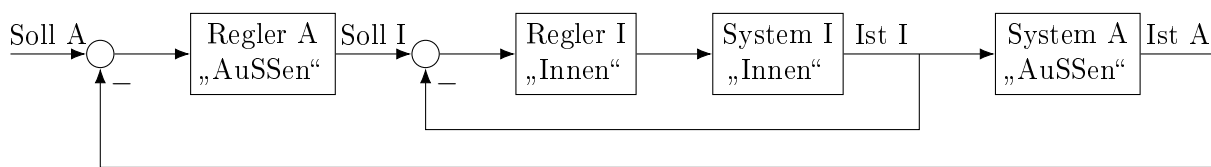


Abbildung 8.7.: Blockschaltbild einer allgemeinen Kaskadenregelung.

### 8.1.7. Kaskadenregelung

Bei einer *Kaskadenregelung* werden mehrere Regler hintereinander geschaltet (siehe Abbildung 8.7), wobei die Führungsgröße des „inneren“ Reglers von einem „äußeren“ Regler eingestellt wird. Dadurch können bspw. verschiedene Taktraten umgesetzt werden (z. B. eine geringere Taktrate zur Berechnung der Gelenkstellungen und eine hohe Taktrate zur individuellen PID-Gelenkregelung).

### 8.1.8. Stabilität als Sprungantwortverhalten und PID-Parameter

Wird der Sollwert  $x_d(t)$  als Sprung/Stufe gegeben, so hängt die Reaktion des Reglers (bzw. der rückgekopelten Werte) von dem Dämpfungsgrad des Systems ab. Möglich sind dabei folgende Verhaltensweisen:

1. Hunting: Oszillation um den Sollwert (ohne diesen zu erreichen).
2. Ringing: Überschwingen, aber konvergieren gegen den Sollwert.
3. Overshoot (underdamped): Unterkritische Dämpfung, Überschießen. Die Kontrollparameter sind zu hoch, es besteht Gefahr für Beschädigungen bei Greif- und Transportaufgaben.
4. Overdamped: Überkritische Dämpfung, der Sollwert wird nur langsam erreicht (ohne Überschießen).
5. Critically Damped: Kritische Dämpfung, der Sollwert wird bei minimaler Verzögerung ohne Überschießen erreicht. Dies ist das ideale Verhalten.

Schwierigkeiten bei der Regelung:

- Einstellung der Regelparameter und Regelstruktur.
- Nichtlineare Roboterdynamik
- Veränderung der Parameter im Laufe der Zeit (z. B. durch Veränderung der Nutzlast oder durch Verschleiß).

### Einstellung der PID-Parameter

Für die Einstellung der Parameter, bzw. deren Analyse und Auswertung, sind folgende Metriken relevant:

- *Ausregelzeit*  
Die Zeitspanne, bis der Sollwerttoleranzbereich dauerhaft erreicht wird.
- *Anregelzeit*  
Die Zeitspanne von Sprungeingabe bis zum erstmaligen Erreichen des Sollwerttoleranzbereichs.
- *Überschwingweite*  
Prozentualer Wert des Überschwingens des Endwertes bei der ersten Oszillation.

Zur Einstellung der PID-Parameter kommen zwei grundlegende Verfahren in Frage:

1. Bei einem bekanntem mathematischen Modell der Prozessdynamik (Transferfunktion) können analytische Methoden angewendet werden, um die Parameter zu bestimmen (und damit die transienten und stationären Spezifikationen zu erfüllen).
2. Ist die Systemdynamik nicht ausreichend bekannt, so müssen die Parameter experimentell eingestellt werden, z. B. nach dem verfahren nach Ziegler-Nichols:
  - Der Regelkreis wird mit einem einfachen P-Regler durch Vergrößerung von  $k_p$  an die Stabilitätsgrenze (der Wert, bis zu dem ungedämpfte Schwingungen auftreten)  $k_p = K$  gebracht.
  - Sei  $P$  die Schwingungsdauer, dann können die restlichen Parameter wie folgt bestimmt werden (Heuristik!):
    - Für P-Regler:  $k_p = 0.5K$
    - Für PI-Regler:  $k_p = 0.45K$   $k_i = 1.2/P$
    - Für PID-Regler:  $k_p = 0.6K$   $k_i = 2.0/P$   $k_v = P/8.0$

### 8.1.9. Digitale Implementierung eines PID-Reglers und Diskretisierung

Wird ein PID-Regler digital implementiert, so müssen die I- und D-Terme diskret approximiert werden. Für ein Zeitintervall der Länge 1 heißt das:

- P-Term:  $e_i = q_d - q$  zur Zeit  $i$
- I-Term:  $\sum_{i=0}^{i=\text{jetzt}} e_i$
- D-Term:  $e_i - e_{i-1}$

Diese Diskretisierung kann das Systemverhalten stark beeinflussen! Ist die Abtastrate  $F_{\text{abast}}$  nicht hoch genug, so können Aliasing-Effekte auftreten (d. h. bestimmte Muster werden nicht erkannt, die Frequenz erscheint nach der Abtastung geringer als sie tatsächlich ist). Nach dem Nyquist-Shannon-Abtasttheorem muss die Abtastrate zur Vermeidung von Alias-Effekten größer als das doppelte der Analogen Frequenz  $F_{\text{analog}}$  sein:

$$F_{\text{abast}} > 2F_{\text{analog}}$$

In der Praxis ist eine typische Wahl für die Abtastfrequenz  $1/5$  bis  $1/10$  der Anregelzeit.

## 8.2. Nichtlineare Regelung

Im Gegensatz zu dem Feder-Masse-System ist die Roboterdynamik i. A. nichtlinear, mehrdimensional und in allen Gelenken verkoppelt. Die allgemeine Form einer Systemdynamik erster Ordnung<sup>1</sup> lautet:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_m(t) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}(t), \mathbf{u}(t)) \\ \vdots \\ f_m(\mathbf{x}(t), \mathbf{u}(t)) \end{bmatrix} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$$

mit dem Zustandsvektor  $\mathbf{x}(t) = [x_1(t) \ \cdots \ x_m(t)]^T$  und den Steuerungsgrößen  $\mathbf{u}(t) = [u_1(t) \ \cdots \ u_l(t)]^T$ .

<sup>1</sup>alle Dynamiken lassen sich auf ein solches System zurückführen



---

### 8.2.1. Systemlinearisierung

---

Eine lineare Systemdynamik hat die Form  $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$  mit zwei konstanten Matrizen  $\mathbf{A} \in \mathbb{R}^{m \times m}$  und  $\mathbf{B} \in \mathbb{R}^{m \times l}$ . Ein nichtlineares System  $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$  kann lokal um einen Arbeitspunkt (bspw. die Ruhelage)  $(\mathbf{x}_0, \mathbf{u}_0)$  durch die Taylorentwicklung

$$\dot{\mathbf{x}}_0 + \Delta \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \mathbf{J}_x(\mathbf{x}_0, \mathbf{u}_0) \cdot \Delta \mathbf{x}(t) + \mathbf{J}_u(\mathbf{x}_0, \mathbf{u}_0) \cdot \Delta \mathbf{u}(t) + \boldsymbol{\xi}(t)$$

linearisiert werden, indem das Restglied  $\boldsymbol{\xi}(t)$  entfernt wird. Dadurch ergibt sich folgendes linearisiertes System:

$$\Delta \dot{\mathbf{x}}(t) = \mathbf{J}_x(\mathbf{x}_0, \mathbf{u}_0) \cdot \Delta \mathbf{x}(t) + \mathbf{J}_u(\mathbf{x}_0, \mathbf{u}_0) \cdot \Delta \mathbf{u}(t)$$

mit  $\Delta \mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_0$  und  $\Delta \mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_0$  sowie den Jacobi-Matrizen  $\mathbf{J}_x(\mathbf{x}_0, \mathbf{u}_0)$  und  $\mathbf{J}_u(\mathbf{x}_0, \mathbf{u}_0)$  ausgewertet am Arbeitspunkt  $(\mathbf{x}_0, \mathbf{u}_0)$ .

Achtung: Dieses linearisierte Systems gilt nur approximativ um den Arbeitspunkt und kann nicht global verwendet werden!

---

### 8.2.2. Modellbasierte Manipulatorregelung

---

Zur Systemlinearisierung gibt es drei Grundlegende Ansätze:

1. Lokale Linearisierung um einen Arbeitspunkt  
Problem: Ein Roboter bewegt sich ständig über unterschiedliche Bereiche und keine Linearisierung kann überall gültig sein.
2. Verwendung vieler Linearisierungen und Verschiebung des Arbeitspunktes mit der Bewegung des Roboters  
Problem: Wenig praktikabel, da hoher Rechenaufwand.
3. Verwendung der inversen Roboterdynamik

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

und Partitionierung der Reglerstruktur:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\hat{\boldsymbol{\tau}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

$$\hat{\boldsymbol{\tau}} = \ddot{\mathbf{q}}_d + \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e}$$

$$\mathbf{e} = \mathbf{q}_d - \mathbf{q}$$

Dadurch werden die  $n$  Gleichungen entkoppelt und das Regelgesetz „linearisiert“ (entkoppelt).

- Diese für allgemeine Regelstrecken anwendbare, modellbasierte Regelungsmethode („feedback linearization“) wird auch als „inverse dynamics control“ bezeichnet und ist eng verwandt mit „computed torque control“.
- Die Realisierung setzt voraus, dass die Gelenkregelung zentral (d.h. nicht wie bei einer Kaskadenregelung) geschieht. Bei den meisten Industrierobotern ist dies nicht möglich, bzw. technisch nicht vorgesehen.
- Der der Praxis muss die Roboterdynamik in Echtzeit berechnet werden, weshalb die Gleichungen entweder „von Hand“ optimiert werden müssen oder sehr leistungsfähige Mikroprozessoren notwendig sind.

- Die Parameter der Dynamik müssen ausreichend bekannt sein (gute Reibungsmodelle sind i. A. schwierig, Trägheit und Massen, Unsicherheiten, ...).
- Vorteil: Es sind hohe Genauigkeiten und Geschwindigkeiten in der Bewegung erreichbar.

---

### 8.2.3. Adaptive Manipulatorregelung

---

Bei der adaptiven Regelung wieder die modellbasierte Regelung mit Dynamikmodell kombiniert mit einem Adaptionsverfahren, um die Modellparameter durch Vergleich von berechneter und gemessener Zustände anzupassen (bspw. mittels Gaussian Process Regression).

---

### 8.2.4. Bahnregelung in Weltkoordinaten

---

Die Bahnregelung kann direkt (naiv) durch Nutzung der (inversen) Kinematik und der (inversen) Jacobi-Formulieren in Weltkoordinaten genutzt werden. Besser ist allerdings eine direkte Messung der Position in Weltkoordinaten als die (aufwendige und Fehlerbehaftete) Umrechnung.

---

## 8.3. Kraft-/Momenten-Regelung

---

Manipulationsaufgaben erfordern den gezielten Umgang mit dem physikalischen Kontakt des Roboters mit der Umwelt. Bei einer reinen Positionsregelung können kleine Variationen der Positionen extrem große Kontaktkräfte hervorrufen. Dadurch können Schäden hervorgerufen werden, bspw. beim hantieren mit Glas oder generell bei eingeschränkten Aufgaben. Somit ist ebenfalls eine Messung sowie Regelung der Kontaktkräfte und -momente nötig.

Im folgenden werden nur Systeme betrachtet, bei denen beide Objekte (d. h. der Manipulator und das manipulierte Objekt) eine hohe Steifigkeit besitzen (d. h. keine passiven, strukturellen Nachgiebigkeiten) und nur geringe Störungen (z. B. durch Reibung) auftreten.

Die Modellierung und Unterteilung der Manipulationsaufgabe wird in Teilaufgaben unterteilt. Dabei wird jede Teilaufgabe mit einer Anzahl an „natürlichen Beschränkungen“ verbunden und für jede Teilaufgabe eine *verallgemeinerte Fläche definiert*. Dabei gelten die Positions-/Geschwindigkeits-Beschränkungen entlang der Flächennormalen und die Kraft-/Momenten-Beschränkungen entlang der Flächentangenten. Dies ermöglicht eine Partitionierung der Freiheitsgrade in zwei „orthogonale“ Mengen, die nach verschiedenen Methoden geregelt werden.

Die *Zusammensetzungsstrategie* ist eine geplante Abfolge künstlicher Beschränkungen zur Durchführung der Manipulationsaufgabe in einer bestimmten Weise. Dazu werden Strategien zur Erkennung von Änderungen der Kontaktsituation und für unerwartete Situationen benötigt.

### Praxis

- Industrielle Manipulatoren sind extrem steif in den Gelenken und Gliedern konstruiert, damit hohe Positionsgenauigkeiten bei hohen Geschwindigkeiten erreicht werden können.
- Durch eine Kraft-/Momenten-Regelung kann (mit relativ hohem Regelungs- und Sensoraufwand) die Nachgiebigkeit am Endeffektor aktiv geregelt werden.
- Alternativ kann durch die Konstruktion eines elastischen Gelenkantriebs (z. B. über Feder- oder Dämpferelemente) eine passive Nachgiebigkeit realisiert werden.

- Eine solche passive Nachgiebigkeit hat jedoch häufig den Nachteil, dass der Manipulator in der Regel nur noch für sehr spezielle Anwendungen einsetzbar ist und keine hohen Positionsgenauigkeiten mehr möglich sind (der Endeffektor „wackelt“ zu sehr hin und her).
- In der Praxis treten Kraft-/Momenten-Regelungsprobleme jedoch meistens nur in teilweise beschränkten Manipulationsaufgaben auf.  
Daher sind hybride Bahn- und Kraft-Regelungsverfahren notwendig.
- Anforderungen:
  - Bahn-/Positions-Regelung entlang der Richtungen der natürlichen Kraftbeschränkungen.
  - Kraft-/Momenten-Regelung entlang der Richtungen von natürlicher Positionsbeschränkungen.
  - Außerdem wird eine Methode zur (beliebigen) Kombination dieser Modi entlang orthogonaler kartesischer Richtungen eines beliebigen Beschränkungs-Koordinatensystems gebraucht.
- Konsequenz: Es sind kartesische Bahnregelungsverfahren notwendig (siehe Abschnitt 8.2.4).

---

## 8.4. Bahn-/Kraft-Regelung

---

### 8.4.1. Hybride Regelung

---

Beispiel: Ein dreiachsiger, kartesischer Manipulator muss eine Fensterscheibe putzen, wobei  $y_w$ -Achse die Fensternormale darstellt. Dann ist eine Kraftregelung in  $y_w$ -Richtung und eine Positionsregelung in  $x_w$ - und  $z_w$ -Richtung nötig. Die Lösung ist hier eine Positionsregelung für zwei und eine Kraftregelung für eines der Gelenke (bei einer entsprechenden Roboterstruktur mit drei Schubgelenken).

Abbildung 8.8 zeigt das Blockschaltbild einer verallgemeinerten hybriden Regelstruktur, wobei  $S$  und  $S'$  mit  $S + S' = E$  ( $E$  ist die Einheitsmatrix) die jeweiligen Komponenten ein-/ausblenden, die den jeweiligen kartesischen Richtungen des Koordinatensystems der Beschränkungen entsprechen. Im oben genannten Beispiel lauten diese Matrizen wie folgt:

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad S' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

---

### 8.4.2. Parallele Regelung

---

Bei der *parallelen Regelung* wird die Interaktionskraft, bzw. die Position, direkt geregelt, wobei der Kraftregler in der Regel als dominierend ausgelegt wird.

- **Stärken:**
  - Kraft- und Positionsregelung sowie Sollwerttrajektorie-Folgeregelung können unabhängig voneinander implementiert werden.
  - Robust gegenüber Unsicherheiten im Umgebungsmodell.
- **Schwächen:**
  - Es wird eine Erfassung der Drehmomente benötigt.
  - Es gibt (noch) keinen industriellen Standard.
  - Die Impedanz ist nicht wählbar.

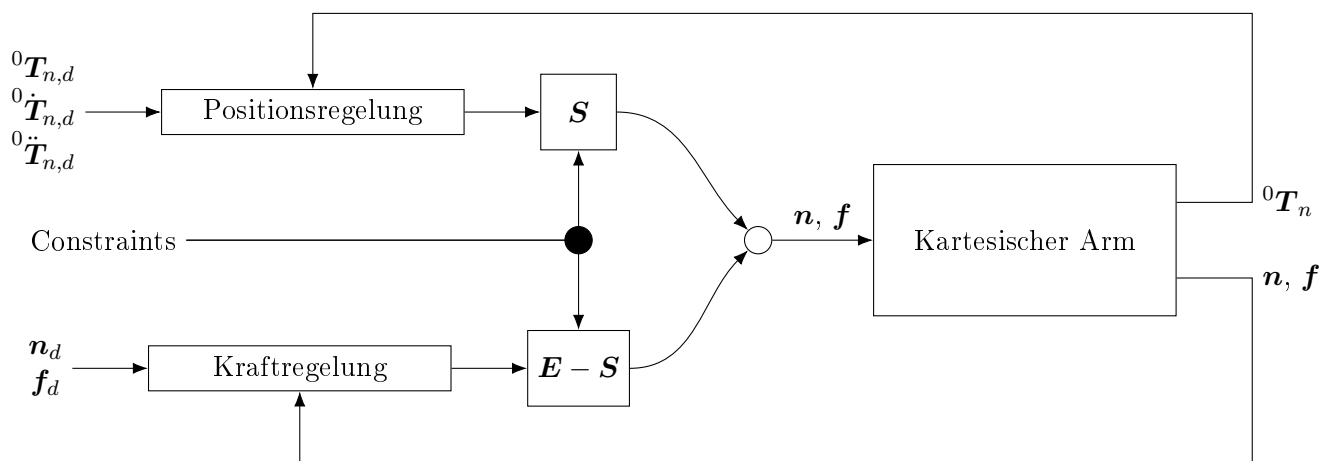


Abbildung 8.8.: Blockschaltbild eines verallgemeinerten hybriden Bahn-/Kraft-Reglers mit den Sollwerten  ${}^0T_{n,d}$ ,  ${}^0\dot{T}_{n,d}$ ,  ${}^0\ddot{T}_{n,d}$ ,  $n_d$  und  $f_d$ .

### 8.4.3. Diskussion

Der Aufgabenraum wird aufgeteilt in separate Positions-/Orientierungs- sowie Kraft-/Momenten-geregelte Teilräume.

- Entlang einer gegebenen Richtung kann dabei nur entweder Position oder Kraft geregelt werden, aber nicht beides gleichzeitig.
- Die gewünschte Bewegung und die gewünschte Kontaktkraft müssen konsistent zu den Umgebungs- und Kontaktbeschränkungen formuliert werden (d. h. die direkte Kraftregelung benötigt ein gutes explizites Modell der Aufgabe).
- Die hybride Positions-/Kraft-Regelung berücksichtigt keine dynamischen Kopplungen und Interaktionen zwischen dem Manipulator und der Umgebung, sofern diese nicht vorab genau spezifiziert werden (können). Für diese Fälle ist die hybride Regelung dementsprechend keine gute Regelung.

## 8.5. Nachgiebigkeitsregelung (Compliant Control)

Die sichere und reaktive Interaktion des Roboters mit der Umgebung bei direktem physikalischem Kontakt ermöglicht viele wichtige Anwendungen (z. B. physikalische Mensch-Roboter-Interaktion) und feinfühligke Kontaktaufgabe. Durch die in Abschnitt 8.4.3 beschriebenen Nachteile der direkten hybriden Bahn-/Kraft-Regelung ist dies mit einer reinen solchen nicht möglich.

Eine Idee zur Lösung ist die *Impedanz-Regelung*:

- Die Interaktion im Kontakt mit der Umgebung verursacht Abweichungen der Endeffektor-Bewegung von der Soll-Bewegung.
- Diese Abweichung wird zu Kontaktkräften und -momenten in eine dynamische Soll-Beziehung zwischen Position/Orientierung und Kraft/Moment in Form von *mechanischer Impedanz* bzw. *Admittanz* mit anpassbaren Parametern gesetzt.
- Dadurch wird eine indirekte Kraft-/Momenten-Regelung ermöglicht.

---

### 8.5.1. Verallgemeinerte Betrachtung nach Hogan

---

In der verallgemeinerten Betrachtung nach Hogan wird ein Roboter als physikalisches System in einer physikalischen Umgebung angesehen. Entlang jedes Freiheitsgrades der Interaktion zwischen dem System und seiner Umgebung werden je zwei *Leistungsvariablen* betrachtet:

- *Effort* („Anstrengung“): Kraft oder Moment
- *Flow* („Fluss“): Lineare oder rotatorische Geschwindigkeit

Aus Sicht der Umgebung gibt es damit nur zwei Typen von physikalischen Systemen:

- *Admittanzen*, welche Effort (Kraft, Moment) aufnehmen und Flow (Geschwindigkeit) ausgeben.
- *Impedanzen*, welche Flow (Geschwindigkeit) aufnehmen und Effort (Kraft, Moment) ausgeben.

Bei der dynamischen Interaktion zweier physikalischer Systeme (Manipulator und seine Umgebung) müssen sich beide physikalisch entlang jedes Freiheitsgrades ergänzen, d. h. falls der eine sich als Impedanz verhält, muss der andere sich als Admittanz verhalten und umgekehrt. Auf ein Objekt (mit Trägheit und kinematischen Beschränkungen) kann jedoch nicht immer eine Kraft (oder ein Moment) ausgeübt werden, es muss sich jedoch auch nicht immer bewegen. In solchen Fällen können die beschreibenden Gleichungen nicht in Impedanz-Form dargestellt werden, eine Admittanz-Form ist jedoch immer möglich.

Damit ein Manipulator, der mechanisch an eine solche Umgebung gekoppelt ist, physikalisch kompatibel mit der Admittanz der Umgebung ist, sollte dieser ein Impedanz-Verhalten annehmen.

---

### 8.5.2. Impedanzregelung

---

Bei der *Impedanzregelung* wird eine aktiv nachgiebige Bewegung erzeugt (ein Roboter mit Kraftregelung agiert, ein Roboter mit Impedanzregelung reagiert). Das Grundprinzip (bzw. das grundlegende Impedanzgesetz) lautet:

Erzeugte Kontaktkraft

$$= (\text{aktive Federkonstante}) \cdot (\text{Regeldifferenz in Position und Orientierung}) \\ + (\text{aktive Dämpfungskonstante}) \cdot (\text{Regeldifferenz in linearer und Winkel-Geschwindigkeit})$$

Ein weiterer Vorteil der Impedanzregelung ist, dass ein schneller und definierter Wechsel zwischen kontaktfreiem und kontaktbehaftetem Zustand möglich ist (z. B. zum Abfangen von Stößen), sofern die Sensorik hinreichend schnell arbeitet.

#### Prinzip:

- Indirekte Regelung der Interaktionskraft durch eine direkte Regelung des Interaktionsverhaltens als Impedanz mit (nahezu) beliebigen Dämpfungs- und Steifigkeitskoeffizienten.
- Ermöglicht die Regelung von Steifigkeit und Dämpfung.
- Durch die Parameterwahl kann eine gute Positionsfolge auf eine gute Kraftfolge abgestimmt werden.
- Kann bei niedrigen Impedanzen angewandt werden.
- **Vorteile:**
  - Trajektorienfolge der Impedanz (nicht der Position oder der Kraft).

- Bei dem Wegfall der inneren Rückführung auch ohne Gelenkdrehmomentmessung als open-loop implementierbar.

- **Nachteile:**

- Benötigt Rückdrehbarkeit der Antriebe.
- Es gibt (noch) keinen industriellen Standard.

---

### 8.5.3. Admittanzregelung

---

Für eine closed-loop Impedanzregelung ist eine Kraft-/Drehmomentmessung notwendig. Ideal ist eine direkte Messung mit spezifischer Sensorik in jedem Gelenk, was allerdings sehr kostenaufwendig und noch nicht standardmäßig verfügbar ist. Alternativ können die Gelenkdrehmomente über den Stromfluss geschätzt werden. Dies benötigt allerdings eine hinreichend genaue Stromflussmessung und ist noch weniger entwickelt und verfügbar.

Alternativ können, zur Simulation, die Kräfte/Drehmomente gelenk-extern im Hand- oder Fußgelenk gemessen werden. Ein nachgiebiges Verhalten im kartesischen Raum ist simulierbar mit einer *Admittanzregelung*.

**Prinzip:**

- Indirekte Regelung der Interaktionskraft durch direkte Regelung des Interaktionsverhaltens als Admittanz mit (nahezu) beliebigen Dämpfungs- und Steifigkeitskoeffizienten.
- Ermöglicht die Regelung der Nachgiebigkeit.
- Durch die Parameterwahl kann eine gute Positionsfolge auf eine gute Kraftfolge abgestimmt werden.
- Kann bei hohen Impedanzen angewandt werden.
- Es existieren viele Varianten von Admittanzregelungen.
- **Vorteile:**
  - Robust gegenüber Modellfehlern und Reibung.
  - Keine Rückdrehbarkeit der Gelenkantriebe benötigt.
  - (Nahezu) Standard.
  - Es werden nur Kraft-/Moment-Sensoren am Endeffektor (und nicht in jedem Gelenk) benötigt.
- **Nachteile:**
  - Keine Impedanz-, nur Admittanz-Regelung.

---

### Implementierung

---

---

#### 8.5.4. Aktiv-passive Konzepte für Impedanz/Admittanz

---

- Passive (bzw. mechanische) Nachgiebigkeit
  - Durch geeignete mechanische Elemente (z. B. Federn), ohne externe Energiezuführung.
  - Viele unterschiedliche, aktuelle Entwicklungen bei VSA und deren Regelung.

- Aktive (bzw. virtuelle) Nachgiebigkeit durch Regelung
  - Hybride Kraft-/Positionsregelung.
  - Parallele Kraft-/Positionsregelung.
  - Impedanzregelung.
  - Admittanzregelung.
- Kombination von passiver und aktiver Nachgiebigkeit
  - Aktuelles Forschungsgebiet.
  - Unterschiedliche Ansätze.
  - Bisher wenig Vergleichbarkeit.

---

## 8.6. Bildgeführte Regelung

---

Die Genauigkeit der „klassischen“ Regelung ist unter anderem beschränkt durch die Genauigkeit der Roboterkinematik bzw. des kinematischen Modells. Abhilfe kann hier durch Messung der zu regelnder Größe durch externe Sensoren gewonnen werden.

Bei der bildgeführten Regelung gibt es dabei zwei grundsätzlich verschiedene Strukturen:

1. Positionsbasierte, kamerageführte Regelung
  - Extraktion von Informationen aus Kamerabildern zur Rekonstruktion der aktuellen RAN eines Objekts.
  - Zusammen mit dem Sollwert für die RAN: Berechnung der kartesischen Regelabweichung.
  - Die RAN wird dabei bspw. durch Bildverarbeitung gewonnen (maschinelles Sehen).
2. Bildbasiert, kamerageführte Regelung
  - Direkte Berechnung der Regelabweichung durch Vergleich der aktuellen Werte eines Merkmalsvektors anhand des 2D-Kamerabildes und deren Sollwerten (ohne 3D Rekonstruktion).
  - Der Regler soll den Roboter so steuern, dass der aktuelle Bildmerkmalsvektor mit den gewünschten Werten übereinstimmt.

---

### 8.6.1. Bildbasiert

---

Idee:

- Das 2D-Kamerabild  $B(q)$  ist abhängig von der Konfiguration  $q$ .
- Aus dem Kamerabild  $B$  werden dann:
  1. Merkmale  $m(B)$  extrahiert und
  2. Merkmale Positionen zugeordnet.

**Aufgabe:** Es soll eine Konfiguration  $q_{\text{Zielposition}}$  gefunden werden, sodass

$$F(q_{\text{Zielposition}}) = m(B(q_{\text{Zielposition}})) - m(B(q_{\text{ist}})) = 0$$

gilt, bzw. sodass

$$\left\| m\left(B(q_{\text{Zielposition}})\right) - m\left(B(q_{\text{ist}})\right) \right\|_2^2$$

minimal wird.

**Ansatz:** Das Gauß-Newton-Verfahrens für nichtlineare Ausgleichsprobleme motiviert ein Iterations- und Bewegungsverfahren:

$$q_{\text{neu}} = q_{\text{ist}} + \Delta q, \quad \Delta q = J(q_{\text{ist}})^{-1} \cdot F(q_{\text{ist}})$$

mit der Jacobi-Matrix  $J = \frac{\partial m}{\partial q}$  des Merkmalsvektors.

---

## 8.7. Multimodale Regelung physikalischer Interaktionen

---

Bei der multimodalen Interaktionsregelung werden z. B. visuelle und Kontaktkraftregelung kombiniert (z. B. visuelle Impedanz-Regelung und Posenschätzung mittels Vision und Kontaktkräften). Es ist auch möglich, weitere Modalitäten zu Kombinieren, z. B. natürliche Sprache, künstliche Haut, ....

---

## 8.8. Regelung und Steuerung bei Mensch und Tier

---

---

### 8.8.1. Propriozeption

---

---

### 8.8.2. Sensoren

---

---

### 8.8.3. Zentrales Nervensystem

---

---

### 8.8.4. Neurale Integration

---

---

### 8.8.5. Informationsverarbeitung

---

---

### Reflexe

---

---

### 8.8.6. Sonstiges

---

---

## 8.9. Elementare Roboterbewegungen

---

Definition: Eine *elementare Bewegungsart* ist eine direkt programmierbare, „kleinste“ Bewegungsmöglichkeit zwischen einer Anfangsposition und einer Endposition. Diese werden (bei einem Industrieroboter) meistens standardmäßig bereitgestellt. Die *kürzeste Fahrzeit*  $t_f$  ist dabei i. A. nicht vom Benutzer wählbar, sondern ergibt sich aus der Bewegungsart.

Elementare Bewegungsarten für Industrieroboter sind z. B.:

- Lineare Interpolation in Gelenkkoordinaten
  - Bei der linearen Interpolation werden Start- und Endgelenkstellung einfach durch eine Gerade verbunden (mit der unabhängigen Variable  $0 \leq s \leq 1$ ):

$$q(0) + s \cdot (q(t_f) - q(0))$$



- Problem: Die lineare Interpolation benötigt unendlich hohe Ableitungen der Anfangsbeschleunigung zur Erlangung der der Steigung entsprechenden, konstanten Geschwindigkeit. Eine mögliche Lösung stellt das Anfahren und Abbremsen mit einer Parabel dar.

- Lineare Interpolation in Weltkoordinaten

$${}^0\mathbf{r}_E(0) + s \cdot \left( {}^0\mathbf{r}_E(t_f) - {}^0\mathbf{r}_E(0) \right), \quad 0 \leq s \leq 1$$

- Kreisbogen-Interpolation
- Spline-Interpolation

---

### 8.9.1. Schwierigkeiten bei kartesischer Bahnvorgabe

---

1. Unerreichbarkeit von Zwischenpunkten.  
Auch wenn Anfangs- und Endpunkt innerhalb des Arbeitsbereichs sind, kann die Verbindungsstrecke Punkte außerhalb von diesem enthalten.
2. Hohe Gelenkgeschwindigkeiten in der Nähe von kinematischen Singularitäten.  
Aufgrund von beschränkten Geschwindigkeiten kann es zu Bahnabweichungen kommen.
3. Anfangs- und/oder Endposition können mit verschiedenen Gelenkpositionen erreicht werden.

Daher wird die Bahnplanung bevorzugt in Gelenkkoordinaten durchgeführt. Diese haben allerdings, bei mehr als drei Gelenken, eine höhere Dimension als der Planungsraum.

---

### 8.9.2. Programmierung einer Bahn als Folge elementarer Bewegungen

---

Es werden Bahnpunkte (Positionen) und dazugehörige Attribute (Orientierung, Geschwindigkeit, ...) definiert.

„Überschleifen“ mit „fly-by“-Zonen:

- Idee: Die einzelnen Positionen werden nur näherungsweise und nicht exakt angefahren.
- Dadurch muss der Roboterarm bei Bahninterpolation nicht vollständig abgebremst werden und die reale Bahngeschwindigkeit kann erhöht werden.

---

### 8.9.3. Elementare Bewegungsarten für fahrende Roboter

---

Für die drei Freiheitsgrade eines punktförmigen Fahrzeuges (Position und Orientierung) sind ähnliche Elementbewegungen wie für Manipulationen implementierbar (lineare, kubische, Kreisbogen Interpolation).

---

## 9. Bahnplanung

---

Abbildung 9.1 zeigt den typischen hierarchischen Aufbau der Auftragsbearbeitung und Planung, wobei sich dieses Kapitel auf die topologische, geometrische, kinematische und kinetische Planung bezieht.

Die *Bahnplanung* ist eine wichtige Mindestanforderung an „autonome“, mobile Roboter und beinhaltet die Planung und Umsetzung eigener Bewegungen. Benötigt werden dazu

- Die Start- und Ziel-RAN,
- eine geeignete, interne Repräsentation der „Welt“, d. h. der Umwelt und des Roboters sowie
- Algorithmen, um sinnvolle Schlussfolgerungen/Planungen zu ermöglichen.

Die interne Repräsentation des Raumes (als Teil der Umwelt) beinhaltet dabei die Darstellung des frei befahrbaren Raumes und das Erkennen von Bereichen/Position und Objekten in der Umgebung. Diese Repräsentationen können in zwei Klassen eingeteilt werden:

- Topologisch (abstrakte Wege)
- Metrisch (geometrische Bahnen und Trajektorien)

Für die folgenden Betrachtungen seien zunächst einige Begriffe definiert:

- *Weg*  
Benannte Verbindung zwischen zwei Punkten, wobei die konkrete Bewegung entlang des Weges nicht berücksichtigt wird.
- *Bahn/Pfad*  
Kontinuierliche Punktfolge mit Position und Orientierung, bspw. bewegen sich Planeten auf einer Bahn.
- *Trajektorie*  
Bahn mit zeitlichen Verlaufsinformationen, d. h. Positions-, Geschwindigkeits und ggf. Beschleunigungswerte. Bahnen/Trajektorien gibt es dabei im  $n$ -dimensionalen Gelenkraum (Konfigurationsraum) sowie im 3-dimensionalen Weltkoordinatensystem (Arbeitsraum).
- *Bahn-/Trajektorien-Planungsproblem*  
Die Bestimmung einer Bahn/Trajektorie von Anfangs-RAN zu End-RAN (in der Regel im Konfigurationsraum), sodass der Roboter nicht mit Hindernissen kollidiert und die geplante Bewegung möglichst konsistent mit den kinematischen und kinetischen Beschränkungen des Roboters ist.

Meistens sind viele, alternative Bahnen von Start zum Ziel möglich. Die Auswahl einer optimalen Bahn kann zum Beispiel durch die Minimierung von Kosten (z. B. Verfahrzeit, Bahnlänge, Energieaufwand, ...) unter der Berücksichtigung von Beschränkungen (Kollisionsvermeidung, Sicherheit, Möglichkeit guter Sensordatenaufnahme, ...) bestimmt werden. Ideal sind hierzu „Anytime“-Algorithmen, in der Realität erfordert die (optimale) Bahnplanung jedoch hohe Rechenzeiten. Daher werden oft Heuristiken verwendet, um die Echtzeitanforderungen einzuhalten.

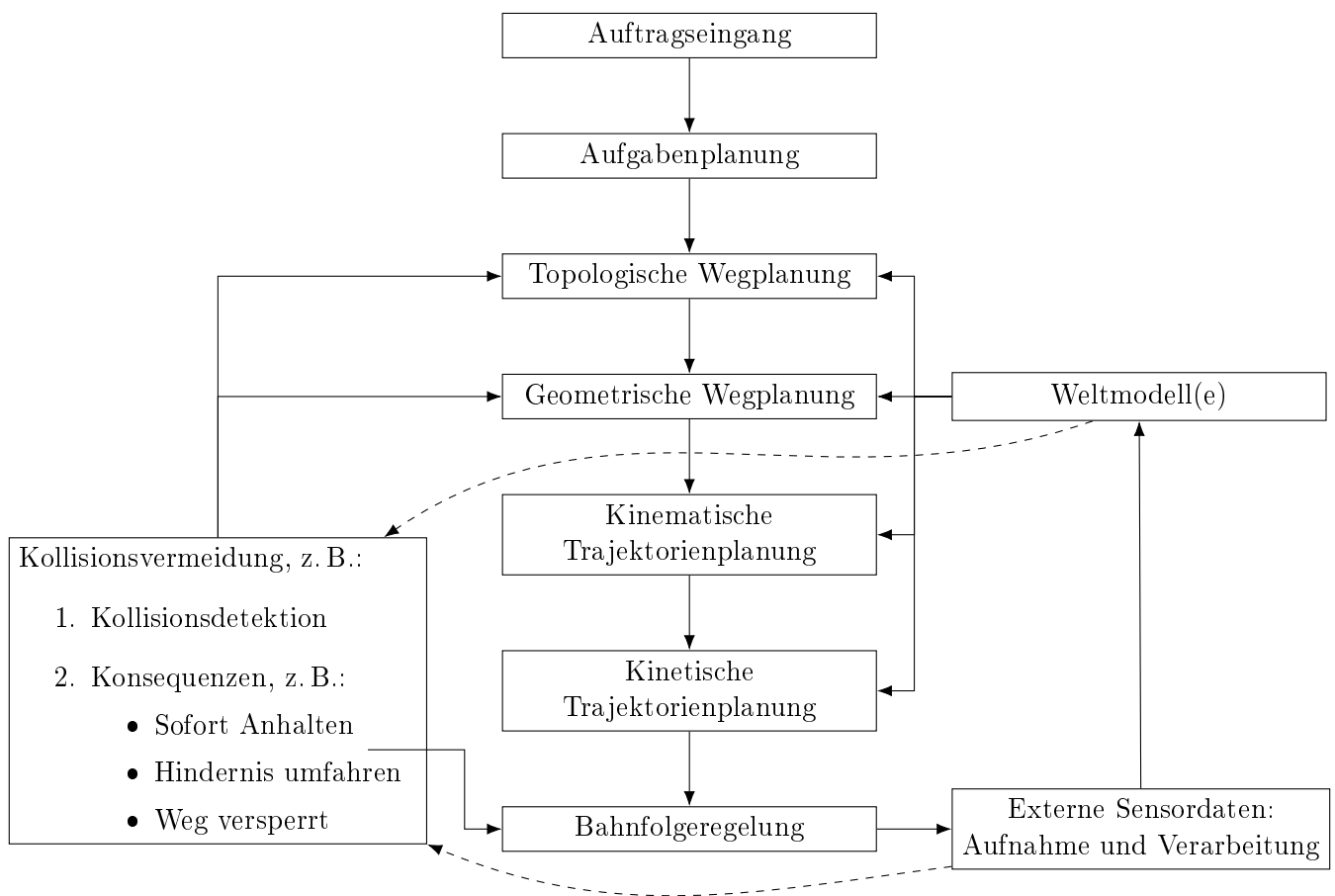


Abbildung 9.1.: Hierarchische Auftragsbearbeitung und Planung.

Art/Ebene der Bahnplanung	Umweltmodell	Robotermodell
Topologische Wegplanung	Global (2D): Graph	Punkt
Geometrische Bahnplanung	Global (2D, 3D): Zellzerlegung, Voronoi-Diagramm, Potentialfeld	Punkt, ggf. mit Ausdehnung
Kinematische Trajektorienplanung	Lokal (3D, 4D): Nichtlineare Optimierung	Kinematisches Robotermodell
Kinetische Trajektorienplanung	Lokal (4D): Dynamische Optimierung	Kinetisches Robotermodell

Tabelle 9.1.: Arten der Bahnplanung, wobei die Verfahren absteigend von globalen und grob granularen zu lokalen und fein granularen Verfahren sortiert sind.

## 9.1. Bahnplanungsarten

Tabelle 9.1 zeigt die unterschiedlichen Arten der Bahnplanung auf. Dabei ist zu beachten, dass viele der im Folgenden betrachteten Verfahren ein (nahezu) perfektes Weltmodell voraussetzen. In der Praxis sind jedoch zusätzlich die Unsicherheiten und Eigenschaften der Sensorik sowie die Echtzeitanforderungen eine große Rolle.

## 9.2. Topologische Wegplanung

Die *topologischen Wegplanung* basiert auf einer Abstraktion der Umgebung in Form diskreter Orte mit verbindenden Kanten. Oftmals wird dies als kantengewichteter (ggf. gerichteter) Graph dargestellt.

Zur Wegplanung wird ein Startknoten  $K_1$  und ein Zielknoten  $K_{\text{Ziel}}$  angegeben und der günstigste Weg vom Start- zum Zielknoten bestimmt (z. B. mit dem Dijkstra- oder A\*-Algorithmus). Eine größeres Kantengewicht bedeutet z. B., dass

- die Passage schwieriger zu passieren ist,
- die Durchfahrt mehr Zeit benötigt (z. B. da sie länger ist oder hoch frequentiert) oder
- die Kante häufig versperrt ist.

Es ist möglich, die Kantengewichte adaptiv aus den Erfahrungswerten des Roboters anzupassen.

## 9.3. Bahnplanung im Arbeitsraum vs. Bahnplanung im Konfigurationsraum

Der Arbeitsraum eines Roboters ist eine Teilmenge des 2- oder 3-dimensionalen kartesischen Raums, wobei Hindernisse durch einfache Hüllen (Kugel, Ellipsoide, Polygone, ...) approximiert werden. Dabei werden Hindernisse oftmals virtuell vergrößert, sodass die Ausdehnung des Roboters auf einen Punkt reduziert werden kann (dies erleichtert die Bahnplanung, kann jedoch einige Bahnen anscheinend unmöglich machen, auch wenn diese möglich wären).

- **Vorteil:** Der kartesische Arbeitsraum ist intuitiv verständlich.
- **Nachteile:**

- Unerreichbare Zwischenpunkte der geplanten Bahn sind schwerer zu detektieren (die gesamte Bahn muss im erreichbaren Arbeitsraum liegen).
- In der Nähe kinematischer Singularitäten werden hohe Geschwindigkeiten geplant, was bei Gelenkbeschränkungen zu Bahnabweichungen führt.
- Möglicherweise sind Start- oder Ziel-RAN mit unterschiedlichen Gelenkstellungen erreichbar.

Aufgrund dieser vielen Nachteile wird Bahnplanung und -interpolation im Konfigurationsraum (Gelenkraum) präferiert. Dies führt bei mehr als drei DOF jedoch zu einem hochdimensionalen Planungsraum.

Definition: Der *Konfigurationsraum* bezeichnet die Abbildung des (erreichbaren) Arbeitsraum in den Raum aller möglichen Gelenkkonfigurationen unter Berücksichtigung der

- Arbeitsraumsituation (Hindernisse, Begrenzungen, ...),
- räumlicher Ausdehnung des Roboters und
- Kinematikmodell des Roboters.

Für mobile Roboter sind häufig Teilmengen des Arbeits- und Konfigurationsraumes identisch (z. B. in der 2-dimensionalen Ebene).

---

## 9.4. Geometrische Bahnplanung

---

### 9.4.1. Metrische Darstellung

---

Die geometrische Bahnplanung basiert auf einer metrischen Darstellung des Raumes. Dies sind diskrete, räumliche Zerlegung des Raumes selbst anstelle der Objekte im Raum. Dieser metrische Raum ist dabei aus elementaren geometrischen 2- oder 3-dimensionalen Körpern (Punkte, Polygone, Kreise; Würfel, Ellipsoide) aufgebaut.

Eigenschaften:

- Menge von elementaren Grundkörpern zur Beschreibung der Objekte verwendet.
- Menge von Zusammensetzungs- und Verformungsoperatoren zur Manipulation von Objekten.

Probleme:

- Stabilität: Kleine Variationen in der Eingabe können zu einer drastischen Veränderung der Darstellung führen. Daher müssen Methoden zur Dämpfung von Schwankungen eingesetzt werden.
- Eindeutigkeit: Verschiedene Umgebungen können die gleiche geometrische Darstellung haben.
- Darstellungsmöglichkeiten: Besondere Eigenschaften der Umwelt sind mglw. nur mangelhaft darstellbar.

---

### 9.4.2. Roadmap-Verfahren

---

Bei *Roadmap-Verfahren* werden Verbindungen zwischen Freiräumen im Konfigurationsraum dargestellt (durch ein Netzwerk von eindimensionalen Kurven), was eine „Straßenkarte“  $R$  produziert. Die Bahnplanung erfordert dann die

- Identifikation der Anfangs- und Endkonfiguration mit Knoten in  $R$  sowie das

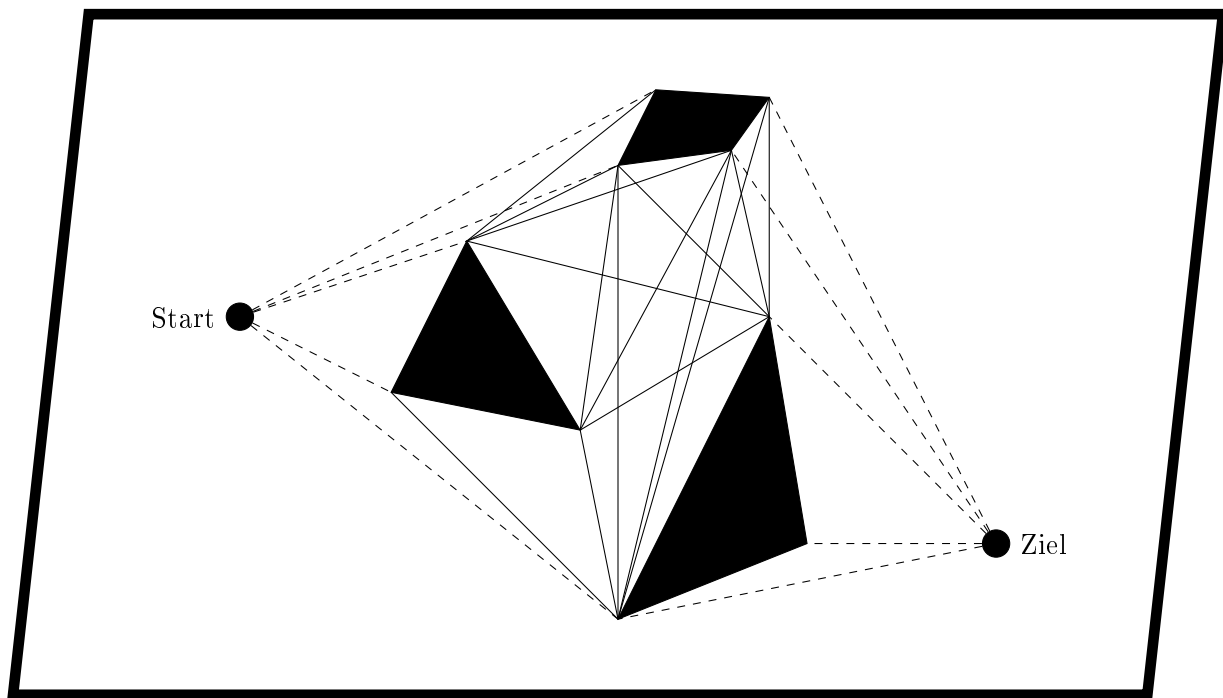


Abbildung 9.2.: Sichtbarkeitsgraph mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen. Zur Vereinfachung wurden die Verbindungen zum Konfigurationsraumrand weggelassen.

- Suchen eines geeigneten Verbindungspfades in  $R$ .

Es gibt verschiedene Verfahren zur Erzeugung von  $R$ :

- Sichtbarkeitsgraphen
- Tangentengraphen
- Voronoi-Diagramme
- Sonstige: Freeway Nets, Silhouettes

---

### Sichtbarkeitsgraph

---

**Ansatz:** Der Sichtbarkeitsgraph wird erzeugt aus der Verbindung zwischen allen Eckpunkten der Hindernisse sowie von Start- und Endkonfiguration, wobei die Verbindungen im freien Konfigurationsraum liegen müssen.

**Bahnplanung:** Bestimmung des kürzesten Verbindungsweges vom Start- zum Zielknoten.

Abbildung 9.2 zeigt einen beispielhaften Sichtbarkeitsgraph im Konfigurationsraum.

---

### Tangentengraph

---

Der Sichtbarkeitsgraph enthält sehr viel mehr Informationen, als nötig sind (für je zwei Hindernisse sind nur die kotangente Verbindungen zwischen vier Eckpunkten relevant). Bei einem *Tangentengraph* werden nur diese Verbindungen betrachtet, d. h. der Sichtbarkeitsgraph wird auf diese reduziert.

---

## Voronoi-Diagramme

---

Eine Schwierigkeit bei Sichtbarkeitsgraphen ist, dass resultierende Bahnen meistens nahe an Hindernissen vorbei gehen, diese sogar berühren können. Daher sind die durch Sichtbarkeitsgraphen generierten Bahnen nur „semi-frei“. Einen alternativen Ansatz verfolgen *Voronoi-Diagramme*, bei denen die Bahnen so weit wie möglich von den nächsten Hindernissen entfernt liegen. Diese werden konstruiert, indem die Wege entlang der Punkte gelegt werden, die den gleichen Abstand zu den nächsten Hindernissen haben (wobei der Rand des Arbeitsbereichs als Hindernis zählt). Dadurch haben die Bahnen immer den maximalen Abstand zu den festen Hindernissen.

**Bahnplanung:** Für die Bahnplanung bewegt sich der Roboter zunächst orthogonal zum Voronoi-Diagramm auf dieses zu und bewegt sich anschließend entlang dem Diagramm zum „Absprungpunkt“ zum Ziel. Dieser liegt so, dass der Roboter wieder orthogonal zum Diagramm auf das Ziel zubewegt.

**Nachteile:**

- Die berechneten Bahnen sind im Vergleich zum Sichtbarkeitsgraphen sehr lang.
- Eckige Bahnen, die nur mit Korrekturen in einer kontinuierlichen Bewegung umsetzbar sind.

---

### 9.4.3. Exakte Zellzerlegung

---

**Ansatz:** Zerlegung des freien Konfigurationsraums in eine Menge an nicht überlappenden, konvexen Gebieten (Zellen), deren Vereinigung genau den freien Konfigurationsraum ergibt. Über die benachbarten Zellen wird dann der Verbindungsgraph (Freiraumgraph) konstruiert. Abbildung 9.3 zeigt eine solche Zellzerlegung.

**Bahnplanung:** Zur Bahnplanung wird ein *Kanal* von der Zelle mit Start zur Zelle mit Ziel konstruiert. Aus diesem Kanal wird anschließend eine Bahn konstruiert, bspw. durch Verbinden der Mittelpunkte der Ränder von benachbarten Zellen.

---

### Trapez-Zerlegung

---

Der Aufwand einer solchen konvexen, optimalen Zellzerlegung ist mindestens polynomial in der Anzahl der Polygonecken der Zerlegung. Eine einfachere Variante stellt die *Trapez-Zerlegung* dar.

Dabei wird eine vertikale Linie (bzw. Hyper-Ebene) von links nach rechts durch den Konfigurationsraum geschoben. Schneidet sich die Linie mit einem Eckpunkt, so wird eine Gerade vom Schnittpunkt zum Rand, bzw. bis zum nächsten Hindernis erzeugt. Dies führt zu einer Zerlegung in trapezförmige oder dreieckige Zellen.

Die Bahnplanung funktioniert dann wie bei einer optimalen Zellzerlegung.

---

### 9.4.4. Approximative Zellzerlegung

---

Ein Vorteil der Zellzerlegung liegt darin, dass lokale Änderungen der Umwelt nur lokale Änderungen der Repräsentation (im Unterschied zum Roadmap-Verfahren). Bei der exakten Zellzerlegung ergibt die Vereinigung der Zellen exakt den Freiraum. Eine approximative Zellzerlegung

- verwendet nur Zellen fester, vorgegebener Größe(n),
- die Vereinigung der Zellen ist eine Teilmenge des Freiraums und
- der Rand einer Zelle hat meistens keine physikalische Bedeutung.

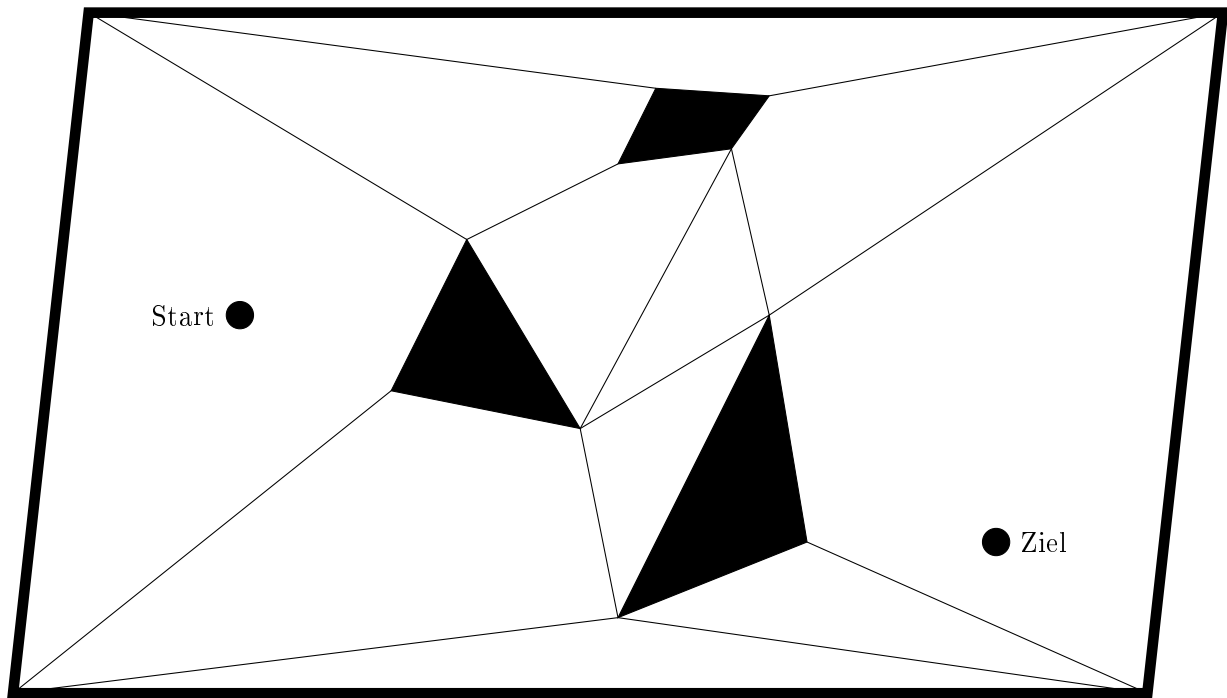


Abbildung 9.3.: Zellzerlegung mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.

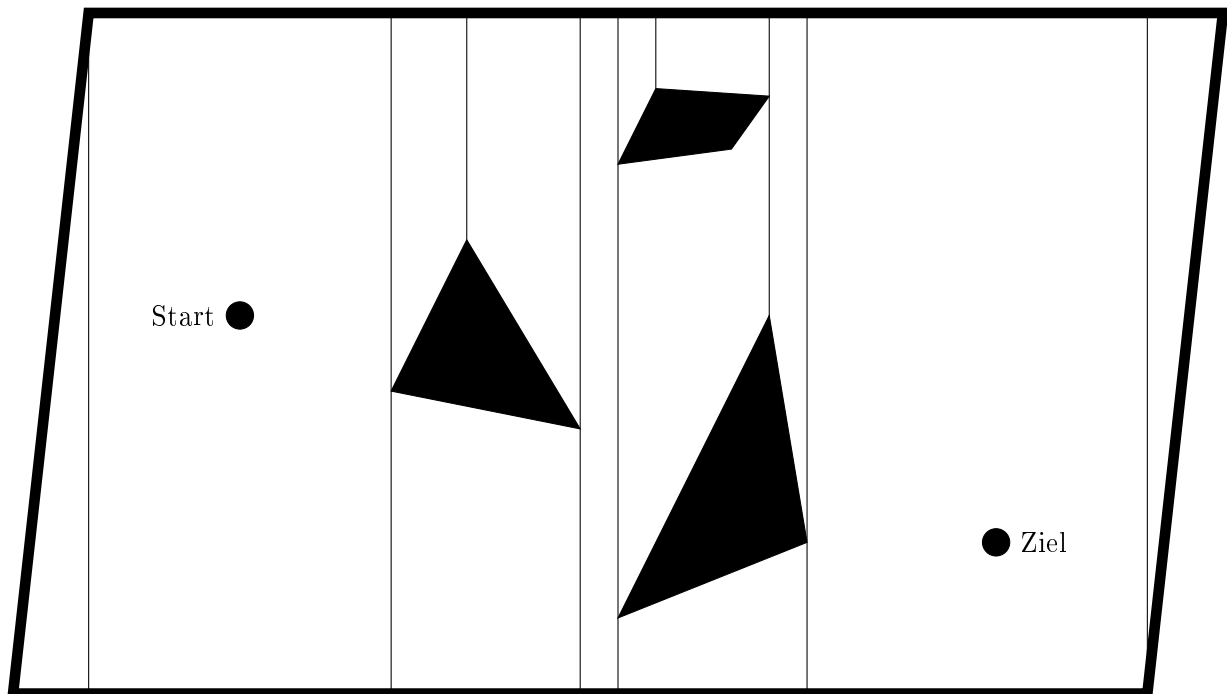


Abbildung 9.4.: Trapez-Zerlegung mit Start- und Zielknoten im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.



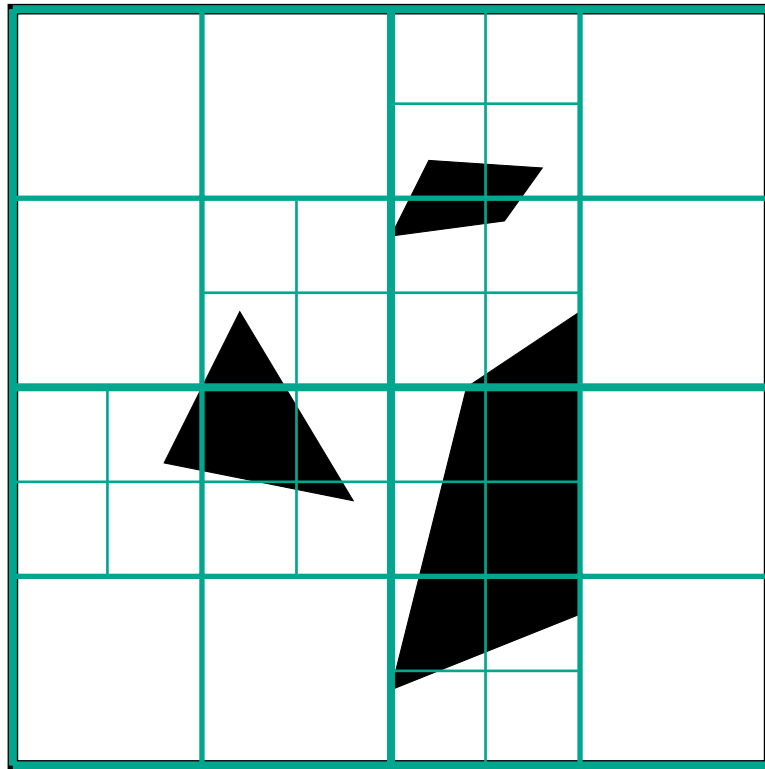





Abbildung 9.5.: Approximative Zellzerlegung im beschränkten Konfigurationsraum, wobei die schwarzen Flächen Hindernisse darstellen.

Wie in Abbildung 9.5 wird der Raum rekursiv in immer vier gleich Große Zellen zerlegt. Daraus kann ein Quadtree erstellt werden, dessen Knoten die folgenden Werte haben können:

- Nur Freiraum: 
- Nur Hindernis: 
- Gemischt: 

Der Raum wird solange weiter zerlegt, bis die maximale Auflösung erreicht wurde. Die Knoten werden dabei in der Reihenfolge

1	2
3	4

von links nach rechts gelistet. Abbildung 9.6 zeigt den Quadtree des vierten Quadranten der approximativen Zerlegung aus Abbildung 9.5.

Das Verfahren lässt sich analog auf einen 3- oder  $n$ -dimensionalen Raum anwenden.

#### 9.4.5. Potentialfeld-Methoden

Zerlegungs-Verfahren beruhen auf der Suche in einem diskreten (bzw. diskretisierten) Raum. Eine alternative sind Heuristiken zur (lokalen) Suche in einem kontinuierlichem Raum.

Bei *Potentialfeld-Methoden* werden Objekte als geladene, punktförmige Teilchen betrachtet, wobei das

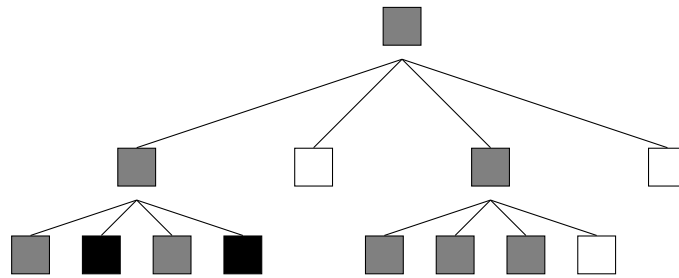


Abbildung 9.6.: Quadtree des vierten Quadranten der approximativen Zerlegung aus Abbildung 9.5.

- Ziel  $q_{\text{Ziel}}$  ein anziehendes Potential  $U_{\text{Ziel}}$  hat und
- Hindernisse ein abstoßendes Potential  $U_{\text{Hindernis}}$  haben.

Daraus ergibt für jede Konfiguration  $q$  ein Potential  $U$ :

$$U(q) = U_{\text{Ziel}}(q) + \sum U_{\text{Hindernis}}(q)$$

Auf den Roboter wirkt dann eine fiktive Anziehungskraft als Gradient des Potentialfelds:

$$F(q) = -\nabla U(q) = - \begin{bmatrix} \partial U / \partial q_1 \\ \vdots \\ \partial U / \partial q_n \end{bmatrix}$$

Dies entspricht dem Verfahren des steilsten Abstiegs (Steepest Descent) zur Minimierung des Gesamtpotentials.

**Berechnung** Es sind viele verschiedene Definitionen der Potentiale möglich, z. B.:

$$U_{\text{Ziel}}(q) := \alpha \cdot \text{dist}(q, \text{Ziel})^2$$

$$U_{\text{Hindernis}}(q) := \frac{\beta}{\beta_0 + \text{dist}(q, \text{Hindernis})}$$

mit geeigneten zu wählenden Parametern  $\alpha, \beta, \beta_0 > 0$ , wobei  $\beta_0$  die Potentialhöhe der Hindernisse festlegt ( $\beta_0 = 0$  entspricht einem unendlichen Potential, d. h. einer Barriere).  $\text{dist}(q, \text{Hindernis})$  entspricht dem minimalen Abstand des Punktes zum Hindernis, wobei „Null“ bedeutet, dass der Punkt im Hindernis liegt.

Für  $U_{\text{Hindernis}}(q)$  gibt es unterschiedliche Berechnungsvarianten, z. B.:

- nur zum nächsten Hindernis oder
- summiert über alle Hindernisse (in der Nähe).

## Diskussion

- **Vorteile:**
  - Einfache Algorithmen.
  - Glatte Bahnformen.

- Online einsetzbar, d. h. die Planung mit aktuellen Sensordaten ist in Echtzeit mit der Regelung koppelbar.

- **Nachteil:**

- Die Gefahr, in lokalen Minima hängen zu bleiben, ist groß.

Abhilfen bzgl. der lokalen Minima sind z. B.:

- Konstruktion eines Potentials ohne lokales Minima, z. B. über Strömungsmodelle (hohe Rechenzeiten).
- Erhöhung des Abstoßungspotentials in einer Region, in der ein lokales Minima erkannt wurde (dieses als Hindernis markieren).
- Kopplung der lokalen Potentialfeld-Methode mit einem globalen Verfahren:
  - Randomisierte Bahnplaner (entkommen aus den Minima durch stochastische Suche).
  - Verwendung von globalen Zellzerlegungs-Verfahren zur Generierung einer Ausgangsbahn.

---

#### 9.4.6. Komplexität der geometrischen Bahnplanung

---

Eine obere Schranke der Komplexität der (globalen) Bahnplanung im Konfigurationsraum wurde von Schwartz und Sharir im Jahre 1983 gefunden:

- Die Komplexität des Problems wächst (vermutlich immer) exponentiell in der Anzahl der Roboterfreiheitsgrade.
- Wenn es schnellere Algorithmen geben sollte, so benötigen diese mindestens polynomiale Zeit bzgl.  $n$ .

Anmerkungen:

- In Spezialfällen mit besonderen Geometrien kann die Komplexität niedriger sein.
- Im Allgemeinen wächst die Komplexität weiter an, wenn sich die Hindernisse bewegen und/oder Unsicherheiten berücksichtigt werden.
- Alle gängigen Bahnplanungsverfahren sind „unvollständig“ bzw. beruhen auf Heuristiken oder Diskretisierungen.

---

#### 9.4.7. Stichprobenverfahren

---

Ein Stichprobenverfahren zur Erstellung einer *Probabilistic Roadmap* (PRM) besteht aus folgenden Schritten:

1. Generieren von zufälligen Stichproben von Konfigurationen (Abbildung 9.7).
2. Entfernen von Stichproben im verbotenen Bereich (Abbildung 9.8).
3. Verbinden jeder Stichproben mit den  $k$  nächsten (noch nicht verbundenen) Nachbarn (Abbildung 9.9).
4. Entfernen von allen Verbindungen, die durch eine verbotene Region gehen (Abbildung 9.10).

Der daraus entstehende (mglw. nicht-planare) Graph ist eine „probabilistische Straßenkarte“ (Probabilistic Roadmap, PRM) (Abbildung 9.11). Die Bahnplanung kann dann unter Verwendung von Graphenalgorithmen (z. B. A\*) stattfinden.

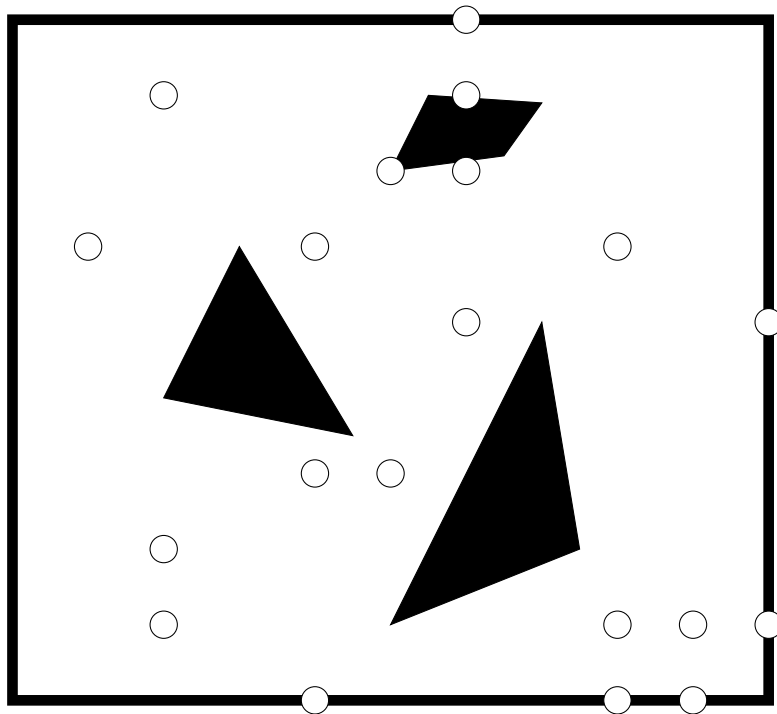


Abbildung 9.7.: Probabilistic Roadmap, Schritt 1: Zufällige Stichproben von Konfigurationen.

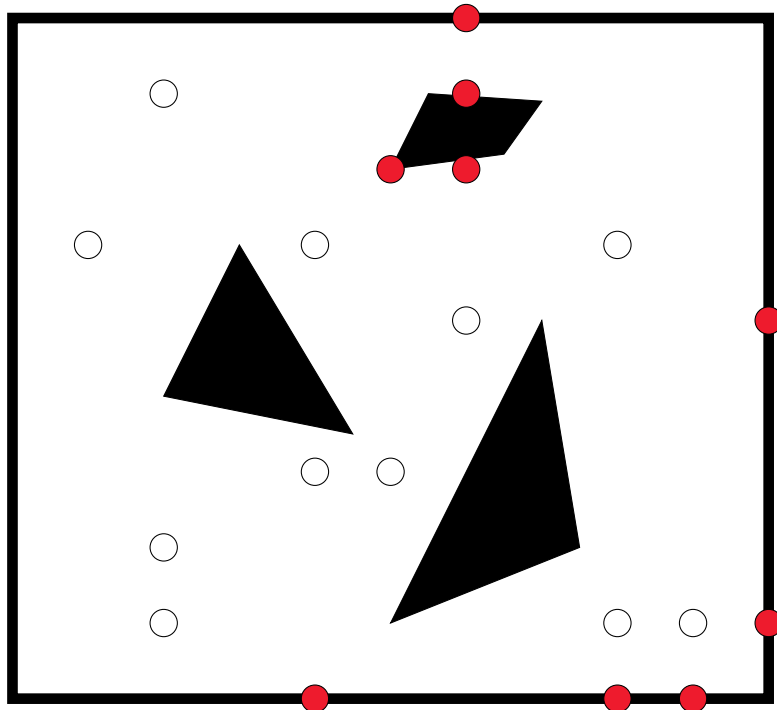


Abbildung 9.8.: Probabilistic Roadmap, Schritt 2: Entfernung der „verbotenen“ Stichproben.

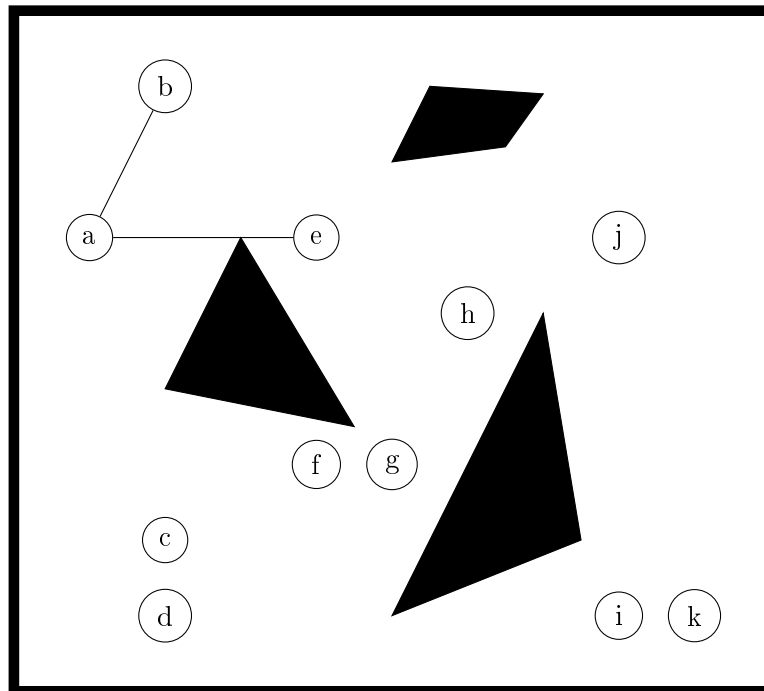


Abbildung 9.9.: Probabilistic Roadmap, Schritt 3: Verbinden jeder gültigen Stichprobe mit den  $k = 2$  nächsten Nachbar.

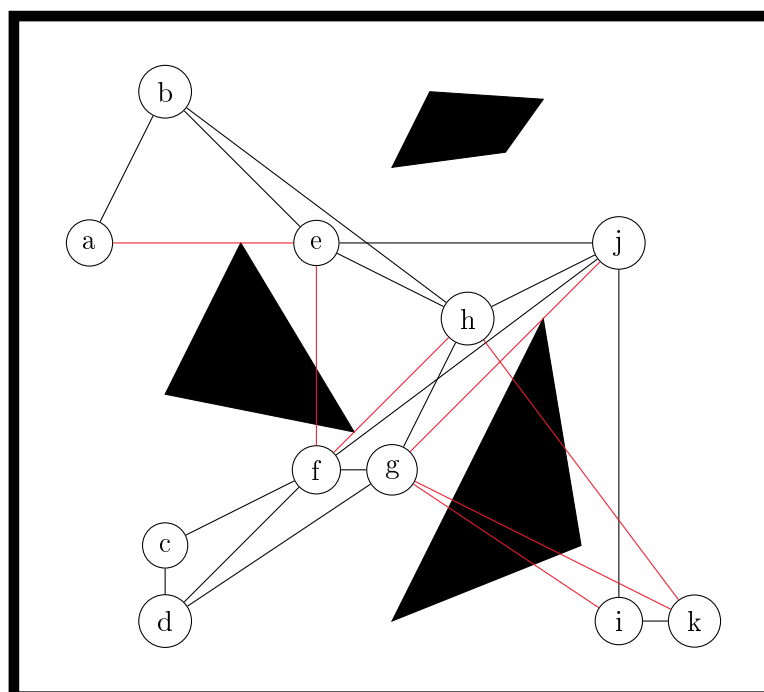


Abbildung 9.10.: Probabilistic Roadmap, Schritt 4: Entfernen der „verbotenen“ Verbindungen.

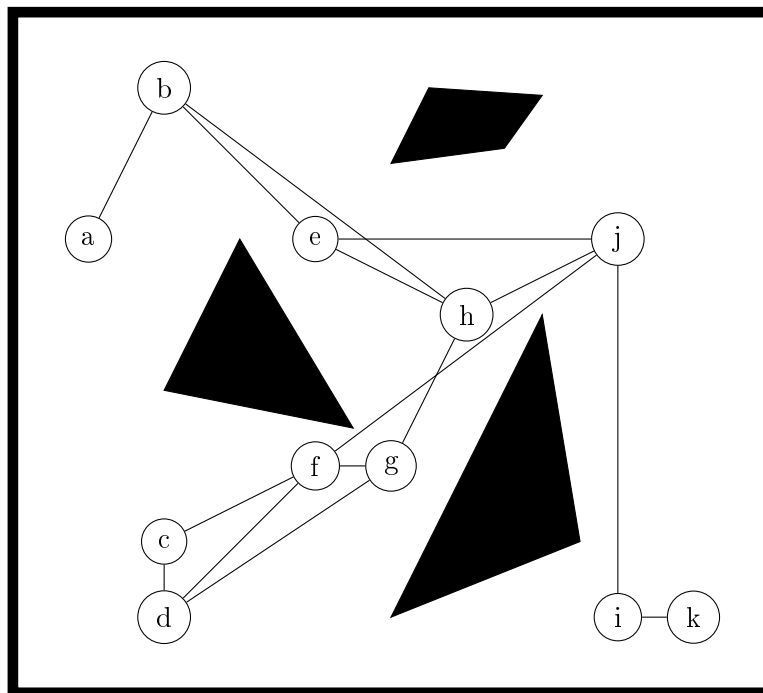


Abbildung 9.11.: Probabilistic Roadmap, Ergebnis: Der resultierende (nicht-planare) Graph ist die Probabilistic Roadmap (PRM)

**Allgemeines Vorgehen** Das allgemeine Vorgehen von Stichprobenverfahren ist erst die Diskretisierung durch Stichproben und anschließende Graphensuche. Dabei gibt es zwei Kategorien von Graphbasierten Verfahren:

- Informiertes (heuristisches) Suchen
- Uninformiertes (blindes) Suchen

Es ist wichtig, dass die Stichproben „gut“ gewählt werden, z. B.:

- Gleichförmige Stichprobenverteilung
- mehr Stichproben bei Punkten mit wenigen Nachbarn
- mehr Stichproben bei Hindernissen

Durch korrekte (zulässige) Heuristiken kann eine enorme Performanzverbesserung erzielt werden. In der Praxis ist die Auswahl solcher Heuristiken jedoch kompliziert...

---

#### 9.4.8. Rapidly Exploring Random Trees (RRTs)

---

Siehe Algorithmus 1.

---

#### 9.4.9. Beispiel: MINERVA

---

MINERVA ist ein Museumstourführungsroboter.

---

### Algorithmus 1 : Rapidly Exploring Random Trees

---

```
Data :  $x_{init}$ ,  $K$ ,  $\Delta t$ 
1 begin
2    $\mathcal{T}.init(x_{init})$  ;
3   for  $k = 1$  to  $K$  do
4      $x_{rand} \leftarrow \text{RANDOM\_STATE}()$  ;
5      $x_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(x_{rand}, \mathcal{T})$  ;
6      $u \leftarrow \text{SELECT\_INPUT}(x_{rand}, x_{near})$  ;
7      $x_{new} \leftarrow \text{NEW\_STATE}(x_{near}, u, \Delta t)$  ;
8      $\mathcal{T}.add\_vertex(x_{new})$  ;
9      $\mathcal{T}.add\_edge(x_{near}, x_{new}, u)$  ;
10  return  $\mathcal{T}$  ;
```

---

---

### Umweltmodell: Belegungskarte

---

- Die Umgebungskarte wird eingeteilt in ein Netz aus quadratischen Zellen, wobei freie Stellen hell und Hindernisse dunkel markiert werden.
- Diese *Belegungskarte* erlernt der Roboter selbstständig (durch Herumfahren).
- Die Verknüpfung der beiden Problemstellungen (Aufbau einer Karte und Bestimmung der Position/Orientierung) wird als *SLAM* (Self Localization and Mapping) bezeichnet (siehe Abschnitt 10.2.5).
- Die Bewegungskarte wird dabei in zwei Schritten aufgebaut:
  1. Erfassung von Laser-, Sonar und Odometriedaten.
  2. Fusion der Sensordaten zur Berechnung der Umgebungskarte und der Position/Orientierung des Roboters.
- Die Bahnplanung (mittels Kostenminimierung) erfolgt anschließend auf Basis der Belegungskarte, wobei diese während der Fahrt bei Bedarf neu berechnet wird.
- Schwierigkeiten:
  - Kürzeste-Wege-Planer sind prinzipiell einsetzbar, allerdings ist in weiten, offenen Umgebung das Versagen der externen Distanzsensoren möglich.
  - Eine Abhilfe ist, bei der Bahnplanung die für Sensoren günstigen Positionen zu berücksichtigen (z. B. nahe am Rand).

---

### „Küstennahe“ Bahnplanung

---

Bei der „küstennahen“ Bahnplanung wird versucht, den Roboter möglichst nah an Wänden (bzw. Allgemein dem Rand der Karte) zu fahren.

---

### Kollisionsvermeidung (MINERVA)

---

- Das Kollisionsvermeidungs-Modul steuert sowohl die momentane Bewegungsrichtung als auch die Geschwindigkeit.

- Dabei wird die Roboterkinematik und -dynamik berücksichtigt (Masseträgheiten, maximale und minimale Drehmomente der Radantriebe).
- $\mu$ DWA (Dynamic Window Algorithm):
  - Eingabe: Rohe Abstandsmesswerte (sowohl gemessene als auch simulierte) und Zielposition.
  - Ausgabe: Sollwert für die lineare und Drehwinkel-Geschwindigkeit.
  - Berücksichtigung von: harten Beschränkungen (der Roboter muss immer vor dem Aufprall auf ein Hindernis zum Stehen) kommen müssen sowie von weichen Beschränkungen (maximale/minimale Momente, Abstand zum Ziel verringern, Abstand zu Hindernissen groß halten (unterschiedlich Gewichtet)).

## 9.5. Kinematische und dynamische Trajektorienplanung

Bei der kinematischen und dynamischen (kinetischen) Trajektorienplanung werden zusätzlich zu den geometrischen Anforderungen auch die kinematischen bzw. kinetischen Anforderungen des Robotermodells berücksichtigt.

### 9.5.1. Allgemeine Formulierung

Es ist die Systemdynamik

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) = \begin{bmatrix} f_1(\mathbf{x}(t), \mathbf{u}(t), t) \\ \vdots \\ f_m(\mathbf{x}(t), \mathbf{u}(t), t) \end{bmatrix}$$

mit den Zustandsvariablen  $\mathbf{x}(t) = [x_1(t) \ \cdots \ x_m(t)]^T$  und den Steuergrößen  $\mathbf{u}(t) = [u_1(t) \ \cdots \ u_l(t)]^T$ , die bestimmten Steuerbeschränkungen

$$\mathbf{u}(t) \leq \mathbf{u}_{\max} \quad \Longleftrightarrow \quad \begin{array}{c} u_1(t) \leq u_{1,\max} \\ \vdots \\ u_l(t) \leq u_{l,\max} \end{array}$$

unterliegen. Für Anfangswerte  $\mathbf{x}(0)$  und Endwerte  $\mathbf{x}(t_f)$  unterliegt das gesamte System im Zeitraum  $0 \leq t \leq t_f$  außerdem bestimmten Zustandsbeschränkungen:

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \geq \mathbf{0} \quad \Longleftrightarrow \quad \begin{array}{c} g_1(\mathbf{x}(t), \mathbf{u}(t), t) \geq 0 \\ \vdots \\ g_n(\mathbf{x}(t), \mathbf{u}(t), t) \geq 0 \end{array}$$

Das *optimale Steuerungsproblem* (Optimal Control Problem) lässt sich nun durch ein Gütekriterium/eine Kostenfunktion wie folgt definieren

$$\min_{\mathbf{u}} J(\mathbf{x}, \mathbf{u}) = \Phi(\mathbf{x}(t_f), t_f) + \int_0^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t) dt$$

wobei die Kostenfunktion  $J(\mathbf{x}, \mathbf{u})$  in zwei Funktionen  $\Phi$  und  $L$  aufgespalten wird. Beispielhafte Gütekriterien sind:



- Zeitminimalität ( $t_f$  frei):

$$\Phi = t_f, \quad L = 0 \quad \Longrightarrow \quad \min_{\mathbf{u}} J_t(\mathbf{x}, \mathbf{u}) = t_f$$

- Energieminimalität ( $t_f$  vorgegeben):

$$\Phi = 0, \quad L = \sum_{i=1}^l u_i^2(t) \quad \Longrightarrow \quad \min_{\mathbf{u}} J_e(\mathbf{x}, \mathbf{u}) = \int_0^{t_f} \sum_{i=1}^l u_i^2(t) \, dt$$

- Kombiniertes Kriterium ( $t_f$  frei):

$$\min_{\mathbf{u}} J(\mathbf{x}, \mathbf{u}) = \rho_1 \cdot J_t(\mathbf{x}, \mathbf{u}) + \rho_2 \cdot J_e(\mathbf{x}, \mathbf{u}), \quad \rho_1, \rho_2 > 0$$

Siehe auch Vorlesung „Optimierung statischer und dynamischer Systeme“.

---

# 10. Navigation mobiler Roboter

---

*Navigation* ist die Fähigkeit, den Kurs (die Fahrt) eines mobilen Roboters so zu bestimmen, dass die Umgebung (Land, Wasser, Luft) durchquert und das Ziel erreicht ist. Die Aufgabe ist dementsprechend die Bestimmung sowie die Umsetzung einer Bahn (bzw. Trajektorie) vom Start  $S$  zum Ziel  $Z$ . Die Navigation besteht dabei aus folgenden Teilaufgaben:

- „Wo bin ich?“ → dieses Kapitel.
- „Wohin bewege ich mich gerade?“ (Bahnumsetzung, Echtzeit-Steuerung und -Regelung, Kollisionsvermeidung)
- „Wie soll ich zum Ziel gelangen?“ (Bahnplanung) → Kapitel 9.
- „Wo komme ich her?“ (Umweltmodelle, Karten) → Kapitel 9 und Abschnitt 10.2.5.

---

## 10.1. Lokalisierung und Positionierung

---

Die Positionsbestimmung kann relativ, absolut oder stochastisch erfolgen.

### Relative Verfahren:

#### 1. Odometrie

- Die Position wird durch „Aufsummieren“ des zurückgelegten Weges von der Startposition anhand von Messwerten berechnet.
- Vorteil: Rein intern-basiert, d. h. es sind keine externen Sensoren nötig.
- Nachteil: Ohne Korrekturen wächst der Positionsfehler sehr schnell.

#### 2. Inertial-Navigation

- Die Beschleunigungs- und Rotationsraten werden mit einem Gyroskop gemessen, die aktuelle Position wird durch zweifache Integration über die Messwerte bestimmt (ähnlich zur Odometrie).
- Vorteil: Rein intern-basiert, d. h. es sind keine externen Sensoren nötig.
- Nachteil: Ohne Korrekturen wächst der Drift-Fehler sehr schnell.

Werden 1 und 2 kombiniert, so ergibt sich eine *Koppelnavigation*.

### Absolute Verfahren:

#### 3. Aktive „Leuchtfeuer“

- Die Position wird aus Winkel- oder Abstandsmessung zu mindestens drei „Leuchtfeuern“ (optisch/Funk) mit bekannten Positionen berechnet (GPS-artig).
- Dabei entstehen mehr Messdaten als unbekannte Parameter  $p$  (nichtlineares Ausgleichsproblem, siehe Abschnitt 10.1.1).

#### 4. Erkennung künstlicher Landmarken

- Die Position wird durch die Erkennung von künstlichen „Grenzpfeilen“ (Formen, Flächen, Farben) an bekannten Positionen berechnet.

#### 5. Erkennung natürlicher Landmarken

- Die Position wird durch Ausnutzen besonderer Eigenschaften der Umwelt berechnet (wie die Erkennung mit künstlichen Landmarken, nur dass die Umwelt nicht modifiziert werden, aber bekannt sein muss).

#### 6. Modellvergleich

- Die aus Sensordaten aufgebaute Karte (Umweltmodell) wird mit einer Referenzkarte verglichen und daraus die Position bestimmt.

### Stochastische Verfahren:

#### 7. Markov-Lokalisierung

- Die Position wird als Aufenthaltswahrscheinlichkeit auf einem bekannten Umweltmodell modelliert, basierend auf den aktuellen Sensordaten.

#### 8. Monte-Carlo-Lokalisierung

- Die Position wird als Dichte von Partikeln betrachtet.
- Es werden Beobachtungs- und Bewegungsmodelle verwendet;
  - Das Beobachtungsmodell modelliert die Wahrscheinlichkeit der Aufnahme von Sensordaten an bestimmten Positionen.
  - Das Bewegungsmodell modelliert die Wahrscheinlichkeit von Bewegungen, die zu einer bestimmten, relativen RAN führen.

---

### 10.1.1. Nichtlineare Ausgleichsrechnung

---

Bei der *nichtlinearen Ausgleichsregelung* (auch als *Methode der kleinsten Quadrate*, *nonlinear least squares* bezeichnet) ist ein nichtlineares Minimierungsproblem

$$\min_{\mathbf{p} \in \mathbb{R}^{n_p}} \varphi_2(\mathbf{p}), \quad \varphi_2 : \mathbb{R}^{n_p} \rightarrow \mathbb{R}$$

mit dem zu minimierenden Gütekriterium (Kostenfunktion)

$$\varphi_2(\mathbf{p}) := \frac{1}{2} \sum_{i=1}^{n_r} \omega_i (r_i(\mathbf{p}))^2$$

mit den reellen Gewichten  $\omega_i \in \mathbb{R}^+$  und einer Fehlerfunktion  $r_i : \mathbb{R}^{n_p} \rightarrow \mathbb{R}$ .

**Beispiel** In diesem Beispiel sind mehrere Randpunkte  $(x_i, y_i)$ ,  $i = 1, \dots, n_r$  eines Balls gegeben und es soll der Mittelpunkt  $(x_K, y_K)$  sowie der Radius  $R_K$  des Balls gefunden werden, d. h. Parameter  $\mathbf{p} = [x_K \ y_K \ R_K] \in \mathbb{R}^3$  ( $n_p = 3$ ), für die für alle  $(x_i, y_i)$  gilt:

$$(x_i - x_K)^2 + (y_i - y_K)^2 = R_K^2 \quad (\text{Kreisgleichung})$$

---

Da die Messwerte unsicherheitsbehaftet sind, ist eine exakte Bestimmung der Parameter nicht möglich, weshalb eine nichtlineare Ausgleichsregelung mit der Fehlerfunktion

$$r_i(x_K, y_K, R_K) = \sqrt{(x_i - x_K)^2 + (y_i - y_K)^2} - R_K \quad (10.1)$$

verwendet wird.

Im Gegensatz zu einer alternativen Fehlerfunktion

$$r_i(x_K, y_K, R_K) = \sqrt{(x_i - x_K)^2 + (y_i - y_K)^2} - R_K^2 \quad (10.2)$$

ist Funktion 10.1 numerisch besser, da die Ableitung von 10.2 im Grenzfall nicht differenzierbar ist.

---

## 10.2. Selbstlokalisierung und Navigation

---

Das Ziel der *Selbstlokalisierung* ist die Bestimmung des Aufenthaltsortes eines Roboters relativ zu einer Ausgangsposition oder absolut bezogen auf ein Weltkoordinatensystem. Für den Rest des Kapitels wird dabei angenommen, dass eine absolute Positionsbestimmung mit GPS oder ähnlichem nicht verfügbar ist (z. B. innerhalb von Gebäuden oder Tunneln). In der Praxis ist eine Kombination der im folgenden untersuchten Methoden mit GPS natürlich möglich und sinnvoll.

Das allgemeine Vorgehen besteht aus zwei Schritten:

1. Messen der (Eigen-) Bewegung des Roboters (Änderung des Aufenthaltsortes) sowie der Umgebung (mit externen Sensoren), was Hinweise auf die absolute Position liefert.
2. Verarbeitung der Messdaten zu einer (deterministischen oder probabilistischen) Vermutung (Hypothese) über den Aufenthaltsort.

Dazu werden geeignete interne und externe Sensoren, Modelle der Sensoren und der Bewegung, geeignete Darstellung der Vermutung(en) über den Aufenthaltsort sowie geeignete Verfahren zur Sensordatenfusion benötigt.

Die unterschiedlichen Verfahren können nach einigen Eigenschaften klassifiziert werden:

1. Verwendung einer Karte (bekannt oder nicht bekannt, metrisch oder topologisch)
2. Art und Anzahl der Hypothesen über den Aufenthaltsort/die RAN des Roboters.
3. Jeweilige Problemstellung (relative/absolute Bestimmung/Verfolgung der RAN).
4. Deterministisch oder nicht deterministisch (probabilistisch).
5. Verwendete/benötigte Sensoren.

---

### 10.2.1. Metrische und Topologische Beschreibung des Aufenthaltsortes

---

**Metrische Beschreibung** Bei der metrischen Beschreibung wird die RAN Allgemein durch einen Vektor  $x \in \mathbb{R}^n$  beschrieben, wobei  $n$  die Dimension der RAN ist. Bei einem Fahrzeug in der Ebene ist diese  $n = 3$  (Position und Winkel), bei einem Körper im Raum ist diese  $n = 6$  (Position und Orientierung). Prinzipiell sind aber auch höhere Dimensionen denkbar, wenn der Roboterzustand genauer modelliert wird.

---

**Topologische Beschreibung** Bei der topologischen Beschreibung wird das Umgebungsmodell als Graph modelliert und der mobile Roboter befindet sich in einem der Knoten.

---

### 10.2.2. Messungenauigkeiten/-unsicherheiten

---

Bei sowohl der metrischen als auch der topologischen Beschreibung tritt das Problem auf, dass keine genauen Daten der Roboterbewegung verfügbar sind:

- Ungenauigkeiten der Sensoren
- Spiel und Elastizitäten (z. B. in den Antrieben)
- nicht im Kinematikmodell erfasstes, reales Roboterverhalten (z. B. Rutschen, Kippen, Wackeln)

Eine Lösungsmöglichkeit ist die Modellierung der RAN/des aktuellen Knotens durch eine Wahrscheinlichkeitsverteilung über den verwendeten Raum/den verwendeten Graphen. Dabei bestimmt die Art der Modellierung der Wahrscheinlichkeitsverteilung wesentlich das verwendete Verfahren zur Selbstlokalisierung, bzw. Sensordatenfusion.

---

### 10.2.3. Lokalisierung mit einer Hypothese

---

Bei der Lokalisierung mit einer Hypothese wird zu jedem Zeitpunkt mit genau einer Schätzung der RAN gearbeitet.

---

### Koppelnavigation

---

Bei der *Koppelnavigation* wird die Roboterbewegung anhand interner Sensoren gemessen und durch Auswertung der Bewegungsgleichungen die Änderung von Position und Orientierung nach jeder erfassten Bewegung berechnet. Dabei erfolgt keine Messung der Umgebung, d. h. es findet nur eine relative Bestimmung der RAN statt und Messfehler können weder erkannt noch korrigiert werden.

Dies ist problematisch, da der Fehler der RAN-Schätzung so beliebig ansteigen kann. Daher ist eine Koppelnavigation nur für kurzzeitige Schätzungen geeignet.

---

### (Erweitertes) Kalman-Filter

---

Das *erweiterte Kalman-Filter* (EKF) erlaubt die Fusion der Messdaten von mehreren Sensoren (auch mit verschiedenen Messraten). Dabei wird die RAN-Schätzung als normalverteilte Zufallsvariable modelliert. Das Ergebnis ist eine statistisch optimale Schätzung des Systemzustands<sup>1</sup>.

Sowohl das Robotersystem als auch die Messungen werden durch Modelle beschrieben. Das Mess- und Systemrauschen wird sowie Schätzung des Systemzustands werden dabei als ein Vektor normalverteilter Zufallsvariablen modelliert.

Ausgehend von einer Startschätzung wird dann die RAN des Roboters verfolgt. Während der Laufzeit werden zwei Verarbeitungsschritte immer wieder ausgeführt:

- *Time-Update* (Vorhersage)  
Auf Basis des momentanen Zustands wird, unter Verwendung von Systemfunktionen (Bewegungsmodell), der Folgezustand geschätzt (a-priori Schätzung).

---

<sup>1</sup>Das Kalman-Filter ist ein Bayes'scher Minimum-Varianz-Schätzer für lineare stochastische Systeme in Zustandsraumdarstellung

- *Measurement-Update* (Korrektur)

Die a-priori Schätzung wird mit Hilfe der neusten Messungen externer Sensoren korrigiert.

**Diskretisierung der Bewegungsdynamik** Das allgemeine Roboterbewegungsmodell ist eine nichtlineare und zeitkontinuierliche Differentialgleichung

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t))$$

mit dem Systemzustand  $\mathbf{x} \in \mathbb{R}^n$  und der Systemsteuerung  $\mathbf{u} \in \mathbb{R}^l$ .

Mit  $\mathbf{x}_k := \mathbf{x}(t_k)$ ,  $\mathbf{u}_k := \mathbf{u}(t_k)$ , einer „Schrittweite“  $h_k := t_{k+1} - t_k$  und der Approximation

$$\dot{\mathbf{x}}(t_{k+1}) \approx \frac{1}{h_k}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k)$$

kann daraus ein zeitdiskretes, dynamisches Modell ermittelt werden:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) &:= \mathbf{x}_k + h_k \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k)\end{aligned}$$

Im folgenden wird immer ein zeitdiskrete Modell vorausgesetzt. In der Praxis orientieren sich die Zeitschritte  $t_k$ , bzw.  $h_k$ , an den Taktraten der (schnellsten) Sensordaten.

**Probabilistisches Systemmodell** Mit dem Systemrauschen  $\mathbf{w} \in \mathbb{R}^n$ , ein Vektor von mittelwertfreien, normalverteilten Zufallsvariablen mit Kovarianzmatrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$ , wird das Systemmodell wie folgt erweitert:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$$

Durch dieses Rauschen kann der Systemzustand nicht mehr deterministisch angegeben werden, da er von der Zufallsvariablen  $\mathbf{w}$  abhängt.

Jedoch kann der Systemzustand nun als ein Vektor normalverteilter, unabhängiger Zufallsvariablen mit

- Mittelwert  $\hat{\mathbf{x}} \in \mathbb{R}^n$  und
- Kovarianzmatrix  $\mathbf{P} \in \mathbb{R}^{n \times n}$

geschätzt/betrachtet werden.

**Probabilistisches Messmodell** Mit dem Messrauschen  $\mathbf{v} \in \mathbb{R}^m$ , ein Vektor von mittelwertfreien, normalverteilten Zufallsvariablen mit Kovarianzmatrix  $\mathbf{R} \in \mathbb{R}^{m \times m}$ , lautet das Messmodell wie folgt:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k$$

Dabei ist  $\mathbf{x} \in \mathbb{R}^n$  der Systemzustand,  $\mathbf{z} \in \mathbb{R}^m$  der Messwert und  $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  das Modell der Messung.

**Time-Update** Beim Time-Update wird, basierend auf

- der Schätzung des Systemzustands  $\hat{\mathbf{x}}_{k-1}$  und der Kovarianzmatrix  $\mathbf{P}_{k-1}$  zum Zeitpunkt  $k - 1$  sowie
- der Steuereingabe  $\hat{\mathbf{u}}_{k-1}$  zum Zeitpunkt  $k - 1$ ,

eine a-priori Schätzung  $\hat{\mathbf{x}}_k^-$ ,  $\mathbf{P}_k^-$  des Systemzustands zum Zeitpunkt  $k$  berechnet, d. h. die Parameter des Priors  $p(\mathbf{x}_k) = \mathcal{N}(\hat{\mathbf{x}}_k^-, \mathbf{P}_k^-)$ .

Mit der gegebenen Kovarianzmatrix  $\mathbf{Q}$  des Systemrauschens sowie der Jacobi-Matrix

$$\mathbf{A}_k := \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_k^-, \mathbf{u}_k)}{\partial \mathbf{x}}$$

der Systemfunktion ausgewertet an der a-priori Schätzung werden  $\hat{\mathbf{x}}_k^-$  und  $\mathbf{P}_k^-$  wie folgt berechnet:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \\ \mathbf{P}_k^- &= \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}\end{aligned}$$

Dabei wird die Schätzung ungenauer, wenn die Werte auf der Hauptdiagonalen vom  $\mathbf{P}$  ansteigen (d. h. wenn die Varianz steigt).

**Measurement-Update** Beim Measurement-Update wird, basierend auf

- der a-priori Schätzung  $\hat{\mathbf{x}}_k^-$ ,  $\mathbf{P}_k^-$  des Systemzustands und der Kovarianzmatrix zum Zeitpunkt  $k$  und
- den Messungen  $z_k$  der externen Sensoren zum Zeitpunkt  $k$ ,

eine a-posteriori Schätzung  $\hat{\mathbf{x}}_k$ ,  $\mathbf{P}_k$  des Systemzustands und der Kovarianzmatrix zum Zeitpunkt  $k$  berechnet, d. h. die Parameter des Posteriors  $p(\mathbf{x}_k | z_k, \dots, z_1) = \mathcal{N}(\hat{\mathbf{x}}_k, \mathbf{P}_k)$ .

Mit der gegebenen Kovarianzmatrix  $\mathbf{R}$  des Messrauschens sowie der Jacobi-Matrix

$$\mathbf{H}_k := \frac{\partial \mathbf{h}(\hat{\mathbf{x}}_k^-)}{\partial \mathbf{x}}$$

der Messfunktion ausgewertet an der a-priori Schätzung werden  $\mathbf{K}_k$  (die *Kalmanverstärkung*),  $\hat{\mathbf{x}}_k$  und  $\mathbf{P}_k$  wie folgt berechnet:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k (z_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)) \\ \mathbf{P}_k &= (\mathbf{E} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-\end{aligned}$$

Wobei  $\mathbf{E}$  die Einheitsmatrix darstellt.

Die grundlegende Idee des Filter ist, zunächst die Kalmanverstärkung  $\mathbf{K}$  so zu berechnen, dass die Varianz der a-posteriori Schätzung minimal wird. Anschließend wird der Mittelwert der der (durch  $\mathbf{K}$  gewichteten) Abweichung der erwarteten Messung von der tatsächlichen Messung korrigiert.

- In der Regel weicht die tatsächliche Messung  $z$  von dem erwarteten Messwert  $\mathbf{h}$  ab (um das Residuum  $\mathbf{r}_k := z_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)$ ).
- Die Fehlerkovarianzmatrix  $\mathbf{P}$  beschreibt insgesamt die Zuverlässigkeit der Zustandsschätzung.
- Der Mahalanobis-Abstand  $d_k := \mathbf{r}_k^T \mathbf{P}_k^{-1} \mathbf{r}_k$  kann als Maß für die Vertrauenswürdigkeit der Messungen genutzt werden. Dadurch können potentielle Ausreißer und falsche Messwerte anhand besonders großer  $d_k \gg 0$  erkannt werden.
- Durch die Elimination von Ausreißern ist das Verfahren somit stabil gegenüber einzelnen Falschmessungen in Komponenten von  $z_k$ .

---

## Vor- und Nachteile des EKF

- **Vorteile:**
  - Das EKF ist ein etabliertes Verfahren, vielfach implementiert und es ist viel Know-How vorhanden.
  - Es ermöglicht eine Plausibilitätsprüfung der Messwerte, d. h. Ausreißer können erkannt und verworfen werden.
- **Nachteile:**
  - Mehrdeutigkeiten können nicht dargestellt werden (d. h. es kann immer nur genau eine Hypothese verfolgt werden).
  - Die Bestimmung der Anfangsposition erfordert eine absolute Positionsbestimmung, z. B. über Landkarten.
  - Die Normalverteilungen erlauben keine Abbildung von Hindernissen im Modell (z. B. können keine Nebenbedingungen der Form  $g(x) \geq 0$  zur Kollisionsvermeidung berücksichtigt werden).
  - Das zeitkontinuierliche Modell wird linearisiert, d. h. die Unsicherheiten müssen, relativ zu den Taktraten  $h_k$ , klein sein.

---

### 10.2.4. Lokalisierung mit mehreren Hypothesen

---

- Verfahren mit nur einer Hypothese haben vielfältige Nachteile:
  - Mehrdeutigkeiten können nicht modelliert werden.
  - Es ist eine eindeutige Zuordnung der Merkmale erforderlich.
  - Es ist eine Schätzung des Anfangszustandes nötig.
  - Es werden globale Merkmale benötigt, um den Zustand eines verirrtten Roboters wiederzufinden.
- Eine Abhilfe dieser Probleme ist die gleichzeitige Verfolgung mehrerer Hypothesen über den Systemzustand.

---

### Verwendung mehrerer Kalman-Filter („Multiple Hypothesis Tracking“)

---

- Bei der Verwendung mehrerer Kalman-Filter wird für jede Hypothese ein Kalman-Filter genutzt und die Time- und Measurement-Updates parallel auf allen Hypothesen durchgeführt.
- Die Plausibilität der einzelnen Hypothesen wird auf Basis der vorliegenden Messwerte bewertet.
- Sind Messwerte uneindeutig, so wird eine Hypothese in mehrere aufgespaltet.
- Problem: Die Anzahl der Hypothesen kann exponentiell anwachsen.
- Das Ziel ist somit das klein halten der Anzahl an Hypothesen. Dazu sind verschiedene Operationen nötig:
  - Bewerten von Hypothesen
  - Erzeugen neuer Hypothesen
  - Verwerfen von Hypothesen
  - Vereinigen von Hypothesen



Zunächst muss die Hypothesenmenge jedoch modelliert werden. Die  $i$ -te Hypothese ( $i \in \{1, \dots, N\}$ ) zum Zeitpunkt  $k$  wird dabei dargestellt durch ein Tripel

$$H_k^{(i)} := \left( \mathbf{x}_k^{(i)}, \mathbf{P}_k^{(i)}, p_k^{(i)} \right)$$

mit dem Erwartungswert  $\mathbf{x}_k^{(i)}$ , der Kovarianzmatrix  $\mathbf{P}_k^{(i)}$  und einer Gewichtung  $p_k^{(i)} \geq 0$  (die Wahrscheinlichkeit, dass eine Hypothese gilt). Jede Hypothese wird dabei durch eine Normalverteilung  $p_k^{(i)} = \mathcal{N}(\mathbf{x}_k^{(i)}, \mathbf{P}_k^{(i)})$  dargestellt.

Außerdem muss dafür gesorgt werden, dass die Wahrscheinlichkeiten  $p_k^{(i)}$  normiert sind, d. h. es muss

$$p_k^{(0)} + \sum_{i=1}^N p_k^{(i)} = 1$$

gelten, wobei  $p_k^{(i)}$  die Wahrscheinlichkeit ist, dass keine Hypothese gilt.

## Vorgehen

### 1. Time-Update

Für jede Hypothese wird ein normales Time-Update durchgeführt.

### 2. Measurement-Update

Die Umgebungsdaten werden gemessen und ein Measurement-Update auf die Hypothesen angewandt, zu denen die Umgebungsdaten passen.

### 3. Neubewerten der Wahrscheinlichkeiten der Hypothesen

- Die Wahrscheinlichkeit, dass der aktuelle Messwert  $z$  bei Gültigkeit der Hypothese  $i$  gemessen wird, sei  $P(z | H_k^{(i)})$ .
- Mit dem Normierungsfaktor  $c$  werden nun alle Hypothesen neu bewertet:

$$p_k^{(i_{\text{neu}})} = c p_k^{(i)} P(z | H_k^{(i)})$$

Eine Messung der Umgebung kann neue Hypothesen (sogenannte Kandidaten) über den Systemzustand produzieren. Diese Kandidaten werden unterteilt in

- solche, die in der Nähe einer bestehenden Hypothese liegen und dieser unterstützen → Anwenden eines Measurement-Updates auf diese Hypothese.
- solche, die keine bestehende Hypothese unterstützen → Hinzufügen einer neuen Hypothese zur Hypothesenmenge.

Dabei muss beachtet werden, dass die Menge an Hypothesen nicht beliebig anwächst, d. h. es müssen auch Hypothesen verworfen werden, z. B. wenn

- die Wahrscheinlichkeit  $p_k^{(i)}$  unter einer bestimmten Schwelle liegt:  $p_k^{(i)} < p_{\min}$ ,
- die Hypothese „weit genug“ außerhalb der Karte liegt (unter der Annahme, dass die Karte vollständig ist) oder
- die Hypothese innerhalb von Hindernissen liegt.

---

Wurde eine Hypothese entfernt, so müssen die Wahrscheinlichkeiten danach neu normiert werden! Neben dem Entfernen von Hypothesen können diese auch zusammengefasst werden. Dies wird durchgeführt, wenn die Hypothesen dicht beieinander liegen, d. h.  $\|\mathbf{x}_k^{(i)} - \mathbf{x}_k^{(j)}\|$  klein ist. Die Wahrscheinlichkeit der neuen Hypothese entspricht dann der Summe der Ausgangshypothesen.

### Vor- und Nachteile von Multiple Hypothesis Tracking

- **Vorteile:**
  - Es sind mehrere, konkurrierende Hypothesen möglich.
  - Der Rechenaufwand skaliert mit der Anzahl der Hypothesen.
  - Es ist eine aktive Lokalisierung möglich.
    - \* Aktive Lokalisierungsverfahren steuern den Roboter so, dass der Lokalisierungsfehler/die Kosten für die Bewegung eines schlecht lokalisierten Roboters in einen gefährlichen Bereich, minimiert werden.
    - \* Passive Lokalisierungsverfahren beobachten nur den Roboter, der auf andere Weise gesteuert wird, aber nicht so, dass eine Lokalisierung vereinfacht wird.
- **Nachteile:**
  - Die geometrischen Umgebungsdaten können nicht direkt mit in die Hypothesen einbezogen werden.
  - Es wird angenommen, dass Störungen normalverteilt und statistisch unabhängig sind.
  - Das zeitkontinuierliche Modell wird linearisiert, d. h. die Unsicherheiten müssen, relativ zu den Taktraten  $h_k$ , klein sein.

---

### Diskretisierte Wahrscheinlichkeitsverteilung

Die RAN eines Roboters wird durch einen  $n$ -dimensionalen Vektor beschrieben, wobei die genaue RAN nicht angegeben werden kann, sondern nur eine Wahrscheinlichkeitsverteilung über den metrischen Raum. Diese Verteilung kann sehr ungleichmäßig sein, speziell bei der Berücksichtigung von Hindernissen.

**Ansatz:** Der metrische Raum wird diskretisiert (z. B. durch eine Zellzerlegung im Konfigurationsraum) und es wird die Wahrscheinlichkeit

$$P(\mathbf{x}), \quad \mathbf{x} \in \{\mathbf{x}_1, \dots, \mathbf{x}_N\}, \quad P(\mathbf{x}) \in [0, 1]$$

für jede RAN in einem Feld (Position Probability Grid) gespeichert (zu Anfang einer Gleichverteilung folgend). Bewegt der Roboter sich oder treffen neue Sensordaten ein, so erfolgt eine Neuberechnung der Wahrscheinlichkeiten.

**Motion-Update** Eine Bewegung sei gegeben als Änderung  $\Delta$  des Systemzustands  $\mathbf{x}$ :

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta$$

Des Weiteren wird durch ein Weltmodell  $m$  eine Wahrscheinlichkeit  $P(\mathbf{x} | m)$  vorgegeben, dass  $\mathbf{x}$  eingenommen werden kann.

---

Bei einem solchen *Motion-Update* wird die Wahrscheinlichkeitsverteilung wie folgt neu berechnet:

$$P_{\text{neu}}(\mathbf{x}) = \begin{cases} P(\mathbf{x} - \Delta) \cdot P(\mathbf{x} | m) & \text{falls } (\mathbf{x} - \Delta) \text{ möglicher Ausgangsort} \\ 0 & \text{sonst} \end{cases}$$

Nach den Berechnungen muss  $p$  normiert werden. Durch dieses Aktualisieren der Wahrscheinlichkeitsverteilung werden Fehler in der Messung durch ein Glätten von  $p$  modelliert.

**Measurement-Update** Das Sensormodell und das Weltmodell  $m$  geben die Wahrscheinlichkeit

$$P(\mathbf{x} | \mathbf{z}, m)$$

an, dass  $z$  bei der RAN  $\mathbf{x}$  gemessen wird.

Dadurch wird die Berechnung der aktualisierten Wahrscheinlichkeit nach einer Messung  $z$  (*Measurement-Update*) möglich:

$$P_{\text{neu}}(\mathbf{x}) = P(z | \mathbf{x}, m) \cdot P(\mathbf{x})$$

Somit werden aus der anfänglichen Gleichverteilung nach und nach die unwahrscheinlichen Hypothesen eliminiert.

### Vor- und Nachteile des Position Probability Grids

- **Vorteile:**
  - Die Modellierung beliebig vieler Hypothesen verändert die Laufzeit nicht.
  - Es kann Wissen über die Umgebungsgeometrie einbezogen werden.
  - Es ist eine aktive Lokalisierung möglich.
- **Nachteile:**
  - Sehr rechenaufwändig.
  - Die Auflösung des Zustandsraums ist begrenzt.

---

### Monte-Carlo Lokalisierung

---

Bei der *Monte-Carlo Lokalisierung* wird die Wahrscheinlichkeitsverteilung approximiert:

- Die Darstellung erfolgt durch eine Menge von Partikeln (samples).
- Dabei beschreibt jeder Partikel eine mögliche RAN des Roboters.
- Ein solcher Partikel kann eine RAN präzise wiedergeben (es ist keine Diskretisierung notwendig).

Bei einer Bewegung des Roboters wird die Lage der Partikel verändert, bei einer Sensormessung die Wahrscheinlichkeit der Partikel. Vorausgesetzt werden stochastische Modelle der Bewegung und der Sensorik.

Die Wahrscheinlichkeitsverteilung wird durch  $N$  Partikel dargestellt, die jeweils durch ein Tupel aus Systemzustand  $\mathbf{x}_i$  und Wahrscheinlichkeit  $p_i \in [0, 1]$  beschrieben werden:

$$S_i := (\mathbf{x}_i, p_i)$$

Die Wahrscheinlichkeit über alle Partikel muss dabei normiert sein, d. h.  $\sum_{i=1}^N p_i = 1$ .

## Probabilistisches Bewegungsmodell Das probabilistische Bewegungsmodell

$$P(x | x', u)$$

beschreibt die Wahrscheinlichkeit eines Zustands  $x$  nach der Ausführung einer Bewegung  $u$  im Zustand  $x'$ .

**Bewegungen** Bewegt sich der Roboter, so wird eine neue Wahrscheinlichkeitsverteilung erzeugt:

1. Erzeugen einer neuen Verteilung
  - $N$ -maliges „Ziehen“ aus der alten Verteilung unter Berücksichtigung der Wahrscheinlichkeiten.
  - Dadurch werden Partikel mit hoher Wahrscheinlichkeit vervielfacht und Partikel mit niedriger Wahrscheinlichkeit ausgedünnt.
  - Danach wird die Wahrscheinlichkeit aller neuen Partikel auf  $1/N$  gesetzt.
2. Bewegen der Partikel
  - Vorausgesetzt wird, dass die ausgeführte Bewegung  $u$  bekannt ist.
  - Der in den Partikeln gespeicherte Systemzustand wird, entsprechend der Wahrscheinlichkeitsverteilung  $P(x | x', u)$ , zufällig geändert.

**Messungen** Der externe Sensorwert  $z$  hängt von dem Systemzustand des Roboters ab, wobei das Verhalten der Messung als Wahrscheinlichkeitsverteilung

$$P(z | x)$$

beschrieben wird. Für jeden Systemzustand ist damit die Wahrscheinlichkeit jeder möglichen Messung gegeben.

Das Ziel ist nun, die Wahrscheinlichkeiten der Partikel durch gemessene Sensordaten neu zu bewerten. Dazu werden alle Partikel neu gewichtet:

$$p_{i_{\text{neu}}} = p_i P(z | x_i)$$

Anschließend müssen die Wahrscheinlichkeiten wieder normiert werden.

## Vor- und Nachteile der Monte-Carlo Lokalisierung

- **Vorteile:**
  - Es ist fast jede Wahrscheinlichkeitsverteilung annäherbar.
  - Einfach zu implementieren.
  - Die Anzahl der Partikel kann zur Laufzeit angepasst werden (Genauigkeit  $\leftrightarrow$  Rechenleistung).
  - Es kann Wissen über die Umgebungsgeometrie einbezogen werden.
  - Es ist eine aktive Lokalisierung möglich.
  - Effizienter als Position Probability Grid.
- **Nachteile:**
  - Für die Robustheit des Verfahrens wird eine Mindestanzahl an Partikeln benötigt.

---

### 10.2.5. Simultaneous Localization and Mapping (SLAM)

---

Bisher wurde immer davon ausgegangen, dass ein Umweltmodell gegeben ist. Wird ein Roboter jedoch in einer unbekannten Umgebung eingesetzt, so müssen Selbstlokalisierung auf Aufbau der Karte simultan gelöst werden. Verfügbar sind dabei:

- Odometrie sowie Steuerungsdaten
- Daten externer Sensoren

Zu ermitteln sind:

- Karte der Umgebung
- Trajektorie des Roboters

SLAM ist dabei ein klassisches Henne-Ei-Problem:

- Für die Lokalisierung ist eine Karte erforderlich,
- für die Kartierung ist Lokalisierung erforderlich.

Beide Probleme allein sind dabei vergleichsweise einfach lösbar, die Kombination ist jedoch kompliziert:

- Im Allgemeinen sind nur Sensordaten relativ zum Roboter verfügbar und keine absoluten Daten.
- Ein Fehler in der RAN-Schätzung erzeugt Fehler bei der Kartenerstellung.
- Fehler können erst beim Schließen von Schleifen („Loop Closing“) erkannt und korrigiert werden.
- Die Assoziation von Daten kann schwer sein (die Zuordnung von Sensordaten).

Dabei gibt es zwei grundlegende Kartentypen:

- *Merkmals-basierte Karten*
  - Es werden Merkmale aus den Sensordaten extrahiert und eine Menge von Merkmalen bildet die Karte.
  - Die Karte kann nur so gut sein wie die gelieferten Merkmale.
- *Gitter-basierte Karten*
  - Der Zustandsraum wird Diskretisiert und in Gitter aufgeteilt.
  - Pro Gitterzelle wird eine Schätzung der Belegungswahrscheinlichkeit durchgeführt.
  - Eine „dichte“ Karte ist Vorteilhaft bei der Bahnplanung.
  - Sehr hoher Rechenbedarf für eine hohe Anzahl an Freiheitsgraden, daher fast nur für 2D-Karten eingesetzt.

---

## EKF SLAM

---

Es wird ein erweitertes Kalman-Filter eingesetzt:

- Erster erfolgreicher SLAM-Algorithmus.
- Der geschätzte Zustand ist nicht nur die RAN des Roboters, sondern auch die Position der Landmarken.
- Der Zustandsvektor vergrößert sich mit der Anzahl der Landmarken.
- Die Anzahl der Elemente der beim Update zu invertierenden Matrix wächst quadratisch.
- Eine einzige Kantenhypothese.
- Datenassoziation problematisch.
- **Vorteile:**
  - Eignet sich auch gut für die Schätzung einer 6-DoF-RAN.
  - Die Korrelation zwischen Landmarken wird mit geschätzt.
- **Nachteile:**
  - Nicht robust bei fehlerhafter Datenassoziation, was zu einer Divergenz des Filters und zum Verlust von Karte und Lokalisierung führen kann.
  - Aufgrund der in EKF zugrundeliegenden Linearisierung müssen Unsicherheiten klein bleiben.
  - Der Rechenaufwand steigt quadratisch mit Anzahl der Landmarken.
  - Ausschließlich merkmalsbasiert und daher stark anfällig von Qualität und Anzahl der Merkmale.

---

## FastSLAM

---

Benutzung eines Rao-Blackwellize Particle Filters:

- Die RAN des Roboters wird mit Partikeln modelliert.
- Jeder Partikel besitzt eine eigene Kantenhypothese.
- Bei bekannter Robotertrajektorie sind die Merkmale stochastisch unabhängig voneinander.
  - Ausnutzung dieser Tatsache, Schätzung mehrerer Robotertrajektorien und dazugehörige Kanten.
  - Kantenmerkmale können unabhängig voneinander geschätzt werden.
- Es ist sowohl eine Merkmals- als auch Gitter-basierte Variante der Kartenrepräsentation möglich.
- **Vorteile:**
  - Robust durch Datenassoziation und Kartenschätzung pro Partikel.
  - Die Laufzeit und die Qualität sind über die Partikelanzahl skalierbar.
- **Nachteile:**
  - Je nach Kartenrepräsentation und Anzahl der Partikel hoher Speicher- und Rechenzeitbedarf.
  - Beim Schließen großer oder verschachtelter Schleifen treten Problem durch Partikelverarmung auf.
  - Skaliert schlecht mit der Erhöhung der geschätzten Freiheitsgrade der RAN.

---

## Graph-basiertes SLAM

---

SLAM-Problem als RAN-Graph-Optimierungsproblem:

- Aufeinanderfolgende Odometriedaten erzeugen Zwangsbedingungen zwischen mehreren RANs.
- Die Beobachtung von Landmarken erzeugen Zwangsbedingungen zwischen RANs und Landmarken.
- Optimierung über den gesamten Pfad des Roboters (hier sind eine Vielzahl unterschiedlicher Optimierungsverfahren und Implementierungsvarianten möglich).
- Bisher hauptsächlich offline angewandt, mittlerweile aber auch verstärkt im Einsatz unter Echtzeitbedingungen.

---

## Limitierungen

---

SLAM ist bereits für eine Vielzahl von Problemen gelöst, aber es gibt noch keine gleichzeitige Erfüllung aller Kriterien:

- Dreidimensionale Kartierung
- Integration semantischer Informationen
- Echtzeitfähigkeit
- Robustheit
- Skalierbarkeit
- Einsetzbarkeit in beliebigem Terrain
- Nicht-statische Karten

Aktuelle Forschungsrichtungen:

- Semantic Mapping  
Statt „nur“ geometrische Informationen zu kartieren werden zusätzlich semantische Informationen (z. B. Place Labeling (Flur, Büro, Schränke)) gespeichert. Dies ist notwendig für zielorientiertes Handeln des Roboters.
- Life-long Map Learning  
Es werden auch Veränderungen in der Umgebung des Roboters berücksichtigt.
- Visual SLAM  
Benutzung von Mono- oder Stereo-Kameras, auch zur 3D-Rekonstruktion der Umgebung.

---

## Visual SLAM

---

Feature-basierte Methoden (Detektion und Matching von eindeutigen Merkmalen):

- **Vorteile:**
  - Effizient (Speicher und Rechenleistung)
  - Robust gegen Beleuchtungsänderungen, Modellfehler und Initialisierungsfehler

---

- **Nachteile:**

- Sparse Geometrierekonstruktion
- Erfordert Features in der Umgebung

Direkte Methoden (Direktes Matching der Bildinformation):

- **Vorteile:**

- (Semi-) Dichte Geometrierekonstruktion
- Robuster bei wenigen Features durch Nutzung der gesamten Bildinformation

- **Nachteile:**

- Erfordert gute Initialisierung
- Weniger robust gegen Beleuchtungsänderungen und Modellfehler
- Rechenintensiv (aber Parallelisierbar auf einer GPU)

---

### **Scan-Matching**

---

Schnelle Registrierung des aktuellsten Scans mit bisher erstellter Karte.

- Approximation der räumlichen Gradienten der Karte.
- Interpolation zwischen den Gitterzellen, dadurch „Subpixel“-Genauigkeit.
- Gauss-Newton Verfahren um Maximum-Likelihood Anordnung der Scan-Endpunkte in der Karte zu ermitteln.
- Schätzen von Kovarianz der ermittelten Anordnung zur Bewertung der Qualität des Scan-Matchings.

---

### **3D SLAM – Aktuelle Weiterentwicklungen**

---

- Integration TSDF (Truncated Signed Distance Function) Backend  
Effiziente Echtzeit-Generierung von Gitternetzmodell.
- SLAM unter schwierigen Umgebungsbedingungen (z. B. Rauch) mit zusätzlichen Sensoren (z. B. Radar, Ultraschall).
- Semantisches SLAM
  - CNN-basierte Instanzsegmentierung
  - TSDF Mapping



---

# 11. Middleware und Simulation

---

## Voraussetzungen für intelligente Roboter

- Forschung
  - Seit circa 40 Jahre, sehr stark steigende Anzahl an Konferenzen und Publikationen pro Jahr.
  - Die Robotics and Automation Society ist die größte und am schnellsten wachsende Fachgesellschaft der IEEE.
- Technologie
  - Enorm steigende Rechenleistung und Speicherkapazitäten.
  - Stark steigende Leistungsfähigkeit von Sensortechnologien (z.B. LIDAR, Farb-Tiefbild-Kameras wie Kinect und Intel RealSense).
  - Kontinuierlich steigende Leistungsfähigkeit von Antriebstechnologien, insbesondere von elektrischen Antrieben.

---

## 11.1. Szenarien, Eigenschaften und Herausforderungen

---

Roboter können für die unterschiedlichsten Aufgaben eingesetzt werden. Ständig steigende Komplexität und sich schnell ändernde Hardware erschweren dabei die effiziente Programmierung für solche Einsätze. Wie kann eine „korrekte“ und „zuverlässige“ Funktionsweise sichergestellt werden?

In Robotern kommen werden viele verschiedene Algorithmen aus verschiedenen Bereichen integriert (Sensordaten Verarbeitung, Objekterkennung, Filterung, Modellierung, Lokalisierung, Kartografierung, Vorhersage, Verhaltensentscheidung, Bewegungssteuerung, ...). All diese Algorithmen müssen in ein Gesamtsystem integriert und auf dieses und den jeweiligen Roboter angepasst werden. Diese Integration erfordert einen hohen Aufwand.

Gewünschte Software Engineering Kriterien für die Hard-/Software-Komponenten sind dabei:

- Modularität (lose Kopplung zwischen den Komponenten, einfacher Austausch/einfache Änderung der Komponenten)
- Wiederverwendbarkeit (auf anderen Robotern, in anderen Szenarien)
- Flexibilität (schnelle Anpassung an sich ändernde Hardware, geänderte Aufgabenstellungen)
- Korrektheit
- Zuverlässigkeit

---

## 11.2. Middleware

---

Roboter-Middleware bildet eine Zwischenschicht zwischen der Hardware und dem Sense – Plan – Act Zyklus. Ein weltweit akzeptierter Standard für ein solches System ist ROS, das „Robot Operating System“ (wobei es eigentlich kein Betriebssystem ist, sondern eine Middleware, die auf einem Betriebssystem, z. B. einem Echtzeit-Linux, eingesetzt wird).

Die Middleware ist dabei der „Kleber“ zwischen den Komponenten und sorgt für

- Separation der Komponenten
- Standardisierte Mechanismen zur Kommunikation
- Interfaces definiert durch die ausgetauschten Daten.

Dabei gibt es unterschiedliche Ansätze:

- Kommunikations-Orientiert (RPC, Java RMI, Web Service)
- Applikation-Orientiert (CORBA, J2EE, .NET)
- Nachrichten-Orientiert (JSM, RoboFrame, ROS)

---

### 11.2.1. Nachrichtenbasierte Kommunikation

---

- Gängige Paradigmen
  - Point-to-Point
  - Fan-Out
  - Request-Response
  - Publish/Subscribe
- Synchron vs. Asynchron
- Lokal vs. Remote
- Zusätzliche Komponente: „Message Broker“  
Führt zu einem erhöhten Aufwand und zusätzlichem Overhead, aber einer erheblich einfacheren Struktur.

---

### 11.2.2. Laufzeit-Effizienz

---

Insbesondere bei mobilen Robotern ist die Laufzeit-Effizienz enorm wichtig (es werden oftmals tausende Nachrichten pro Sekunde ausgetauscht). Bei jeder Nachricht muss diese dabei serialisiert, in den Speicher kopiert und deserialisiert werden. Diese Übertragung benötigt (auch lokal) Zeit.

Bei einer einzelnen Nachricht in die Latenz gering, aber in der Summe signifikant. Bei einem Echtzeit-Betriebssystem ist dabei nicht unbedingt schnell, aber das Zeitverhalten ist definiert (d. h. die Dauer einer Operation ist exakt definiert).

---

## 11.3. Sicherstellung von Korrektheit und Zuverlässigkeit

---

- Aufgrund der Komplexität der Software ist eine formale Verifikation nicht praktikabel (zusätzlich gibt es unendlich viele, nichtlineare und stochastische Einflüsse).
- Alternative: Viel testen.
  - Korrektheit und Zuverlässigkeit werden anhand einer vordefinierten Menge an Situationen überprüft.
  - Tritt hierbei kein Fehler auf, so bedeutet das nicht nicht, dass das System „korrekt“ ist.
- Ansätze im Software Engineering:
  - Unit-Tests
  - Regression-Tests
  - Integration-Tests
    - \* Durchlaufen möglichst einfach und dennoch repräsentativer Szenarien (Roboter muss Ball finden, zum Ball laufen, ins Tor schießen, ...).
    - \* Ausführen auf dem Roboter ist dabei zeitintensiv (Vorbereitung) und ressourcenintensiv (Verschleiß).
    - \* Sofern es möglich ist, sollte die Evaluation in der Simulation stattfinden (Software-in-the-Loop).
  - Komponenten-Tests
    - \* Abdecken der „low-level“ Funktionalität.
    - \* Dies ist sinnvoll für einfache Komponenten mit geringer Anzahl an möglichen Eingaben.
    - \* Allerdings nicht sinnvoll für das „high-level“ Verhalten oder ungenaue und verrauschte Eingangsdaten.
    - \* Anwendbar z. B. für die Infrastruktur der Middleware (Weiterleitung von Nachrichten, Funktion unter Last, ...).

---

### 11.3.1. Simulation

---

Die Anforderungen an einen Simulator sind sehr unterschiedlich, z. B.:

- Testen von Laufbewegungen:
  - Ein genaues Dynamikmodell ist unabdingbar.
  - Hoher Rechenaufwand.
  - Nur eine kleine Anzahl an Robotern simulierbar.
- Testen von Teamverhalten:
  - Ein einfaches Kinematikmodell ist ausreichend.
  - Geringer Rechenaufwand.
  - Auch mehrere Roboter gleichzeitig simulierbar.

Ein guter Simulator ermöglicht dementsprechend eine Auswahl von Simulationsmodellen mit unterschiedlichen Abstraktionsebenen und Detaillierungsgraden.

---

### 11.3.2. Automatisierte Testabläufe

---

- Die manuelle Durchführung von Testszenarien ist zeitaufwendig und fehleranfällig.
- Die Tests können z. B. durch Skript automatisiert werden.
  - Ein Skript initialisiert das Szenario, positioniert alle Objekte (Roboter, Ball, Hindernisse) in der Szene.
  - Für jedes Szenario werden Abbruchbedingungen und -zeiten definiert.
  - Das passende Verhalten zum Szenario wird gestartet.
  - Die Ergebnisse werden automatisiert erfasst und ausgewertet.
- Durch Continuous Integration ist ein zeitnahes Feedback und schnell Eingrenzung von Fehler möglich. Außerdem gibt es eine kontinuierliche Statistik über die Entwicklung der Performanz.
- Dadurch wird eine einfache Reproduktion von Problemfällen möglich.
  - Nutzung von „ground-truth“ Daten zur Vereinfachung, z. B. Umgehung der Bildverarbeitung oder Lokalisierung durch die Simulation und nicht mit dem eigentlichen Verfahren.
  - Dennoch sind die Tests nicht vollständig deterministisch.
- Automatisierte Untersuchung auf Regressionen (aufgrund der Komplexität können Seiteneffekte durch Änderungen nicht ausgeschlossen werden).
- Das Beheben von Problem ist allerdings noch immer schwierig (die Ursachen für „falsches“ Verhalten sind nicht direkt offensichtlich, es sind viele Komponenten beteiligt).

---

### 11.3.3. Monitoring

---

- Eine Fehlersuche durch reines „Anschauen des Roboters“ ist aufgrund der Komplexität unmöglich.
- Daher muss „in den Roboter hinein“ geschaut werden.
- Dies geschieht bspw. mit einer (grafischen) Benutzerschnittstelle, um intrinsische Daten zu visualisieren.

---

### 11.3.4. Visuelles Debuggen

---

- Durch Visualisierung einiger Algorithmen können Fehler schnell erkannt werden.
- Beispielsweise kann die Trajektorie des Roboterschwerpunktes visualisiert werden.
- Es ist wünschenswert, dass jeder Algorithmus/jedes Modell passende Visualisierungen oder Parametrisierungen zur Verfügung stellt.

---

### 11.3.5. Offline Analyse

---

- Die Situation und die intrinsischen Daten eines Roboters ändern sich unter Umständen sehr schnell, sodass sie ggf. unmöglich sind online nachzuverfolgen.

- 
- Im Fehlerfall ist mglw. kein Monitoring aktiv, eine nachträgliche Analyse soll jedoch trotzdem möglich sein.
  - Daher sollten alle ausgetauschten Nachrichten aufgezeichnet werden.
    - Mit Nachrichten-basierter Kommunikation ist dies prinzipiell einfach möglich.
    - Die Aufzeichnung muss jedoch dezentral auf jedem Roboter erfolgen (die Bandbreite/Verfügbarkeit von externen Systemen ist limitiert; externe Kommunikation ist während eines Wettkampfs mglw. durch die Regeln untersagt).
    - Je nach Menge der Daten reicht die Schreibrate eventuell nicht aus.
  - Die aufgezeichneten Daten können anschließend abgespielt und analysiert werden.
    - Dabei kann die Situation Schritt für Schritt nachvollzogen werden.

---

#### **11.3.6. Kooperierendes Verhalten**

---

- Die Analyse eines einzelnen Roboters ist meistens ungenügend, es müssen alle Informationen aller Teammitglieder betrachtet werden. Dazu müssen die dezentral aufgenommenen Daten synchronisiert werden.
- Häufig ist die Spielsituation im Nachhinein unklar, weshalb externe Informationsquellen berücksichtigt werden sollten (z. B. ein externes Video des Spielfelds).
- Es ist ein synchrones Abspielen von Videos sowie aufgenommenen Daten zur Analyse nötig.

---

# 12. Steuerung autonomer Roboter

---

---

## 12.1. Steuerungsarchitekturen

---

Es existieren drei grundlegende Paradigmen der Steuerungsarchitektur von autonomen Robotern:

1. Hierarchisch
2. Reaktiv
3. Hybris Deliberativ-Reaktiv

Die Basisfunktionalitäten autonomer Roboter sind in Tabelle 12.1 abgebildet.

---

### 12.1.1. Hierarchisches Steuerungsparadigma

---

Die Steuerung wird hier hierarchisch zerlegt (deliberativ) und verfolgt damit den Ansatz „erst denken, dann handeln“, siehe Abbildung 12.1.

- Über dieses Paradigma wurden die ersten autonomen, mobilen Roboter erstellt.
- Kritik in den 80er Jahren:
  - Der Ausfall eines Moduls führt zu dem Ausfall des Gesamtsystems, bzw. zu drastischer Performanz-Reduktion.
  - Die Planung basiert allein auf dem Weltmodell, nicht direkt auf den Sensordaten. Dadurch werden bspw. Objekte übersehen, die zwar in den Sensordaten auftauchen, im Weltmodell aber nicht berücksichtigt worden sind.
  - Ein zu detailliertes Weltmodell führt zu langen Rechenzeiten und langsamen Reaktionszeiten.
  - Während das Weltmodell aufgebaut wird, ändert sich die Umweltsituation bereits.

Basisfunktionalität	Eingabe	Ausgabe
Sense	Sensordaten	Informationen (Sinne)
Plan/Think	Information (Sinne oder Erkenntnis)	Handlungsanweisungen
Act	Sinne oder Handlungsanweisungen	Steuerbefehle an Antriebe

Tabelle 12.1.: Basisfunktionalitäten von autonomen Robotern.

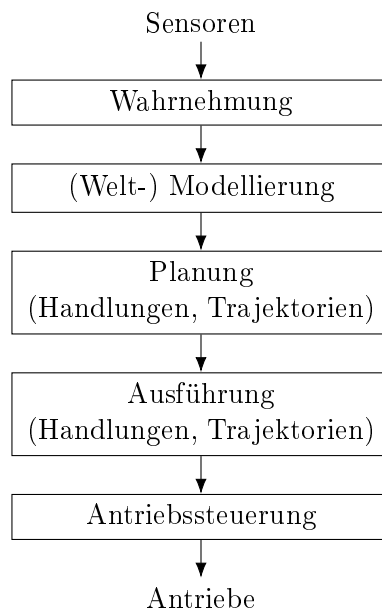


Abbildung 12.1.: Aufbau eines hierarchischen Steuerungsparadigma eines autonomen Roboters.

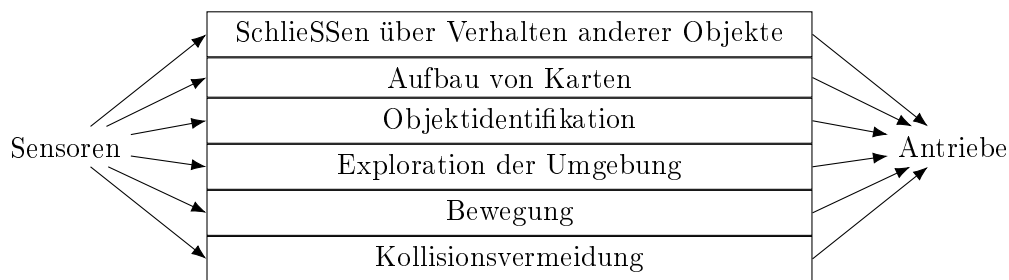


Abbildung 12.2.: Aufbau eines reaktiven Steuerungsparadigmas eines autonomen Roboters.

### 12.1.2. Reaktives Steuerungsparadigma

Das reaktive Paradigma wurde als „Antwort“ auf das hierarchische Paradigma, bzw. die Kritik an diesem, erstellt. Dabei greifen Sense und Act ineinander und der Plan-Schritt fällt weg („handeln ohne zu denken“), siehe Abbildung 12.2.

Vor- und Nachteil dieses Systems ist die Entfernung aller Planungskomponenten. Dadurch werden zwar schnelle Reaktionszeiten erreicht und das System ist robust gegen Modulausfälle, aber das Verhalten ist schwer vorherzusagen und nicht planbar.

### 12.1.3. Hybrid deliberativ-reaktives Steuerungsparadigma

Bei dem hybriden Paradigma

1. plant (deliberativ) der Roboter die beste Unterteilung einer Aufgabe in Teilaufgaben und
2. wählt geeignete (Elementar-) Verhalten zur Lösung der Teilaufgaben aus einer Menge möglicher Verhalten aus.

---

3. Dieser Verhalten arbeiten dann gemäß einem reaktiven Paradigma.

Dabei verschwimmen die effektiven Unterschiede zwischen deliberativer und reaktiver Steuerungsarchitektur, wenn die Durchlauf- und Verarbeitungszeit in der Taktfrequenz der wichtigsten Sensoren erfolgt.

---

#### 12.1.4. Beispiel: Subsumption Architecture

---

Anforderungen an die Steuerungsarchitektur:

1. Mehrere, unterschiedliche Aufgabenziele:
  - Z. B. einerseits die Zielpunkte erreichen, andererseits Hindernisse vermeiden.
  - Die Priorität des Ziels ist relativ und kontextabhängig.
  - Ziele mit einer höheren Priorität müssen erreicht werden, aber auch niedrigere Aufgaben müssen erfüllt werden.
2. Mehrere, unterschiedliche Sensoren:
  - Fehlerbehaftete Sensordaten.
  - Komplexe Beziehungen zwischen Rohdaten und interpretierten Daten.
  - Dateninkonsistenz bei konkurrierenden Sensoren (klein – groß).
  - Sensorausfälle.
3. Robustheit:
  - Die Aufgabenausführung sollte auch beim Ausfall von Sensoren oder anderer Hardware fortgeführt werden.
  - Drastische Umgebungsänderungen müssen bewältigt werden.
4. Erweiterbarkeit
  - Bzgl. Sensoren, Fähigkeiten, Rechenleistung, ...

Einige grundlegende Annahmen hinter der Subsumption Architecture sind:

- Der Aufbau eines komplexen Verhaltens setzt sich aus einfachen Verhaltensweisen zusammen.
- KI: Autonomie und Insektenfähigkeiten als Voraussetzung für intelligentes Verhalten.
- Keine künstliche Umgebung, keine externe Kalibrierung.

In der Subsumption Architecture wird die Robotersteuerung in mehrere, übereinanderliegende Schichten von Kompetenzebenen strukturiert, wobei keine ein explizites Weltmodell aufweist. Dabei benötigt jede Schicht Sensoren und Aktuatoren und jede Schicht beinhaltet die Fähigkeiten der niedrigeren Ebenen. Die Berechnung der Verhaltensweisen findet dann auf allen Ebenen parallel statt, wobei die anzuwendende Verhaltensweise (bzw. Ebene) abhängig von der kontextabhängigen Priorität ausgewählt wird.

Beispielhafte Kompetenzebenen:

0. Kollisionsvermeidung mit (stationären und beweglichen) Objekten
1. Ziellos Herumfahren (unter Vermeidung von Kollisionen)
2. Erforschen der Umwelt (durch Anfahren entfernter Orte)



- 
3. Erstellen einer Landkarte und Planen von Routen
  4. Erkennen von Änderungen der statischen Umwelt
  5. „Nachdenken“ über die Welt in Bezug auf identifizierbare Objekte und Aufgaben bzgl. dieser Objekte
  6. Pläne, die Umwelt in sinnvoller Weise zu verändern, formulieren und ausführen
  7. „Nachdenken“ über das Verhalten von Objekten in der Umwelt und Berücksichtigung in der Planung

- **Vorteile:**

- Schnelle, immer vorhandene Reaktion.
  - Im Wesentlichen die Welt selbst als Weltmodell.

- **Nachteile:**

- Kein explizites internes Weltmodell, daher im Wesentlichen und reaktives Verhalten.
  - Keine „Evolution“ von intelligentem Verhalten.

---

## Schicht 0

---

Kollisionsvermeidung.

- Erzeugung einer momentanen Umgebungskarte aus Ultraschallsensoren.
- Erzeugung einer „gefühlten“, abstoßenden Kraft durch Summierung simulierter, abstoßender Kräfte von Hindernissen.

Hindernisvermeidung durch zwei Taktiken:

- *Collide*: Das Hindernis ist direkt vor dem Roboter, der Roboter bleibt stehen.
- *Runaway*: Wegbewegung von Hindernissen gemäß der (simulierten) Abstoßungskraft.

---

## Schicht 1

---

Zielloses Herumfahren durch die Welt mit Heuristik zur vorausschauenden Kollisionsvermeidung.

- *Wander*: Zufällige Bewegungsrichtungen.
- *Avoid*: Ermittlung der tatsächlichen Bewegungsrichtung unter Berücksichtigung der simulierten Abstoßungskraft.

---

## Schicht 2

---

Exploration der Umwelt.

- Auswahl und
- Anfahren

interessanter Orte.

---

## 12.2. Programmierung von Verhalten

---

„Konventionelle“, wissenschaftliche Programmierung arbeitet nur mit Variablen, die in der Regel nur im Computer existieren. Ebenso ist die Ausführungszeit häufig unkritisch. Bei der Roboterprogrammierung wird jedoch direkt mit dreidimensionalen, physikalischen Objekten gearbeitet. Außerdem sind die Ausführungszeiten kritisch (harte und weiche Echtzeitbeschränkungen, in ereignisbezogenen Takten, ...). Außerdem sind die Daten i. A. stark unsicherheitsbehaftet.

Zu den Aufgaben von Roboterprogrammiersprachen gehören:

- Formulierung der durchzuführenden Aufgaben sowie die
- Steuerung des Roboters bei der Durchführung.

Einige solcher Programmiersprachen sind:

- COLBERT
- Task Description Language
- Behavior Language
- XABSL (basierend auf hierarchischen Zustandsautomaten)
- Petrinetze
- Reactive Plan Language
- FlexBE (basierend auf hierarchischen Zustandsautomaten)

---

### 12.2.1. XABSL

---

- Die Aufgaben sind eingebettet in entsprechendes Verhalten.
- Das Roboterverhalten wird durch einen getakteten, hierarchischen, endlichen Zustandsautomaten beschrieben.
- Die Agentenverhalten werden in Teilverhalten, Teilverhalten wiederum in Basisverhalten zerlegt, die physikalischen Aktionen entsprechen.
- Der dem Automaten entsprechende Graph kann visualisiert werden, was die Überwachung und Fehlersuche vereinfacht.
- Während der Ausführung des Zustandsautomaten ist der aktuelle Zustand definiert als die Teilmenge der aktivierten Optionen entlang dem gerichteten Graphen ausgehend von der Wurzeloption und den entsprechenden Zuständen.
- Der Baum/Pfad der aktiven Optionen ist der Aktivierungsbaum/Aktivierungspfad.
- Die Verhaltenssteuerung interagiert mit der umgebenden Software über eine Reihe von Symbolen:
  - Eingabesymbole sind z. B. Sensordaten, Weltmodell Daten oder Nachrichten von Teammitgliedern.
  - Ausgabesymbole sind z. B. Motorsteuerbefehle oder Nachrichten an andere Agenten.

---

### 12.2.2. FlexBE

---

XABSL unterstützt weder ROS, noch eine Interaktion mit dem Operator oder adaptive Level der Autonomie. Die Bibliothek FlexBE (basierend auch SMACH) fügt ebendiese Roboter-Operator Kollaboration hinzu.

---

#### Autonomy Level

---

Jedes Verhalten läuft mit einem expliziten Autonomie-Level, das während der Ausführung geändert werden kann. Die Ausgangswerte eines Zustands definieren dabei die erforderliche Autonomie:

- Autonomie hoch genug → autonome Ausführung.
- Autonomie zu niedrig → der Operator muss die Ausführung bestätigen.

Dadurch kann der Operator die Ausführung jederzeit steuern.

---

## **A. Quaternionen**

---

---

### **A.1. Einleitung**

---

---

### **A.2. Rechenregeln**

---

---

### **A.3. Umrechnung: Quaternionen zu Rotationsmatrizen und zurück**

---

---

### **A.4. Verkettung von Drehungen**

---

---

### **A.5. Repräsentation der Koordinaten eines Punktes bei Rotation**

---

---

### **A.6. Vergleich mit anderen Darstellungsarten**

---



---

## **B. Zusammenhang zwischen Rotationsmatrix, Drehvektor und Drehwinkel**

---

---

### **B.1. Umwandlung von Drehvektor und Drehwinkel zu Rotationsmatrix**

---

---

### **B.2. Umwandlung von Rotationsmatrix zu Drehvektor und Drehwinkel**

---

---

## C. Notationen

---

Typische Symbole und ihre Bedeutung:

- Kleine und kursive Buchstaben bezeichnen skalare Größen, z. B.  $t, x, f, g, h$ .
- Große, fettgedruckte und kursive Buchstaben bezeichnen Matrizen, z. B.  $A, B, R, T$ .
- Fettgedruckte Buchstaben bezeichnen Vektoren, z. B.  $x, q, F$ .

Tabelle C.1 zeigt typische Buchstaben- und Symbolzuordnungen und Notationen.

### Allgemein

$f, g, h$	(skalare) Funktionsnamen
$t$	Zeit
$t_a, t_s$	Startzeitpunkt
$t_e, t_f$	Endzeitpunkt
$x, y, z$	Koordinatenachsen
$i, j, k$	Laufindex
$l$	Länge
$m$	Masse
$\rho$	Dichte
$g$	Gravitationskonstante
$\delta$	kleiner Abstand
$\frac{df}{dx} = f'(x)$	(totale) Ableitung nach der unabhängigen Variablen $x$
$\frac{df}{dt} = \dot{x}(t)$	(totale) Ableitung nach der unabhängigen Variablen Zeit $t$

### Räumliche Darstellungen und Transformationen

$\mathbf{p}$	Koordinatenvektor eines Punktes
$\theta$	Winkel
$(\alpha, \beta, \gamma)$	Winkeltripel
$(\psi, \theta, \varphi)$	Winkeltripel
$\mathbf{E}$	Einheitsmatrix
$\mathbf{R}$	Rotationsmatrix
$\mathbf{r}$	Translationsvektor
$\mathbf{T}$	homogene Transformationsmatrix

### Manipulatorkinematik

$S$	Koordinatensystem
$n$	Anzahl der Freiheitsgrade
$\mathbf{q} = [q_1 \ \cdots \ q_n]^T$	Zustandsvektor der Gelenkvariablen

### Geschwindigkeit, Jacobi-Matrix und statische Kräfte

$\boldsymbol{\omega}$	Winkelgeschwindigkeitsvektor
$\mathbf{v}$	linearer Geschwindigkeitsvektor
$\mathbf{J}$	Jacobi-Matrix

### Manipulatorodynamik

$\mathbf{I}$	Trägheitstensor
$n, N$	(skalares) Drehmoment
$f, F$	(skalare) Kraft
$\tau, \boldsymbol{\tau}$	(skalare oder vektorielle) Kraft oder Drehmoment
$\mathbf{M}$	Massenmatrix
$\mathbf{C}$	Vektorfunktion der Zentrifugal- oder Coriolisanteile
$\mathbf{G}$	Vektorfunktion der Gravitationsanteile
$K$	kinetische Energie
$P$	potentielle Energie

**Tabelle C.1.: Notationen**