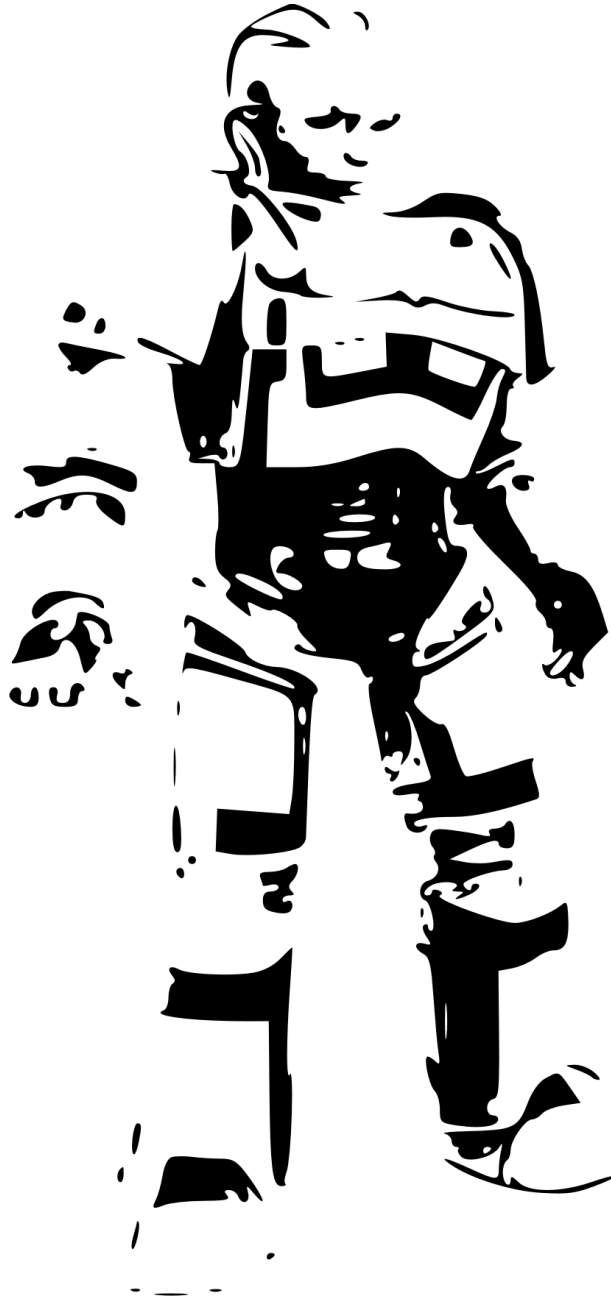


---

# YET ANOTHER KOGSYS SUMMARY

---



Adam L. Urban

[adam.urban@gmail.com](mailto:adam.urban@gmail.com)

L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> – Build vom 19. August 2014

*Revision: 5*



## Einleitung

### Anmerkung zur Zusammenfassung

Dies ist der Versuch einer Zusammenfassung der Vorlesung KOGNITIVE SYSTEME aus dem Sommersemester 2006 an der Universität Karlsruhe (TH). Sie erhebt weder Anspruch auf Vollständigkeit, noch auf Korrektheit. Dieses Dokument hält sich sehr stark an den Folien von Prof. R. Dillmann und Prof. A. Waibel.

Bei Fehlern würde ich mich über eine eMail an [adam.urban@gmail.com](mailto:adam.urban@gmail.com) freuen.

### Links

Meine Homepage:

[adam.urban.de.vu](http://adam.urban.de/vu)

Offizielle KogSys Seite:

<http://www.iain.ira.uka.de/Teaching/VorlesungKogSys/>



*für die Community*

*"Gebildet ist, wer weiß,  
wo er findet, was er nicht weiß."*

*G. Simmel*

## Inhaltsverzeichnis

# 1 Einführung in die digitale Signalverarbeitung

## 1.1 Grundlegendes

### Erfassen/Messen von Signalen:

- Signal als Funktion: Akustik  $f(t)$ ; Bilder  $f(x, y)$ ; Energie als Funktion der Zeit oder des Raumes; Energie: Lautstärke, Helligkeit, Grauwertintensität; Farbe, Stereo, Bildsequenzen
- Eigenschaften: Hinlänglich glatt;  $0 \leq f(x, y) < \infty$  (wertbeschränkt)

### Abtastung und Sampling:

- Man messe das Signal  $f(x)$  an verschiedenen Punkten  $x$
- Punkte  $x$ , in diskreten Abständen, an meist äquidistanten Stellen eines Abtastrasters
- Akustik:  $G = f(0), f(\Delta), f(2\Delta), \dots, f((N-1)\Delta)$
- Bild:

$$G = \begin{bmatrix} f(0, 0) & \dots & f(0, N-1) \\ \vdots & \ddots & \vdots \\ f(N-1, 0) & \dots & f(N-1, N-1) \end{bmatrix}$$

- Meist:  $N = 2^m$
- Bild: Rechtwinklig, schiefwinklig, sechseckige Raster
- Wie oft? Wie großes Raster?  $\rightarrow$  Sampling Theorem

### Quantisierung:

- Im Computer müssen Messwerte quantisiert werden. Die Anzahl der Quantisierungsstufen bestimmt Auflösung.
- Feinere Auflösung: Bessere Qualität, mehr Speicher
- Dynamische Abtastung: Starke Übergänge: Fein rastern - grob quantisieren; schwache Übergänge: Grob rastern, fein quantisieren

### Digitalisierung von Signalen:

- CD, Video (DV, DVD), Digitaler Rundfunk (DAB), ISDN
- Vorteile: Qualität (Bits sind Bits, verlustfreie Übertragung); Kompression; mehrfacher Nutzen von Kommunikationskanälen (Time Division Multiple Access)

### Diracfunktion:

- Definition:

$$\int_{-\infty}^{+\infty} f(x) \delta(x - x_0) dx = f(x_0) \quad \text{und} \quad \int_{-\infty}^{+\infty} \delta(x - x_0) dx = \int_{x_0^-}^{x_0^+} \delta(x - x_0) dx = 1$$

- Hat Fläche 1, an einer beliebig kleinen Umgebung
- $A\delta(x - x_0)$  Impuls mit Stärke  $A$  an der Stelle  $x = x_0$

## 1.2 Faltung

In der Mathematik und besonders in der Funktionalanalysis beschreibt die Faltung einen mathematischen Operator, welcher für zwei Funktionen  $f$  und  $g$  eine dritte Funktion liefert. Diese gibt eine Art "Überlappung" zwischen  $f$  und einer gespiegelten und verschobenen Version von  $g$  an. Definition:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(t - \tau)g(\tau)d\tau \quad (\text{kontinuierlich})$$

$$(f * g)[i] = \sum_{j=-\infty}^{\infty} f[i - j]g[j] \quad (\text{diskret})$$

Bedeutung:

Eine anschauliche Deutung der Faltung ist die Gewichtung einer Funktion mit einer anderen. Der Funktionswert der Gewichtungsfunktion an einer Stelle  $t$  gibt an, wie stark der um  $t$  zurückliegende Wert der gewichteten Funktion in den Wert der Ergebnisfunktion eingeht.

Faltung der Zeitfunktion:

$$F(f_1(t) * f_2(t)) = F_1(\omega) \cdot F_2(\omega)$$

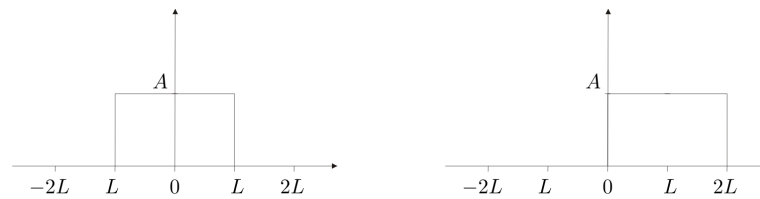
Multiplikation der Zeitfunktionen:

$$F(f_1(t) \cdot f_2(t)) = F_1(\omega) * F_2(\omega)$$

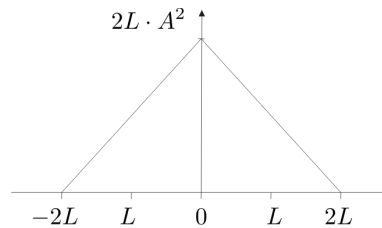
Faltung zweier Funktionen entspricht dem Filtern eines Signals.

Beispiel mit den folgenden zwei Funktionen:

$$f(x) = \begin{cases} A & \text{für } x \in [-L; L] \\ 0 & \text{sonst} \end{cases} \quad \text{und} \quad g(x) = \begin{cases} A & \text{für } x \in [0; 2L] \\ 0 & \text{sonst} \end{cases}$$



Die Faltung der beiden Funktionen sieht folgendermaßen aus:



## 1.3 Fouriertransformation

### Idee der Fouriertransformation

- Zerlegung eines Signals in eine Summe von komplexen Sinus- und Cosinusfunktionen.
- Unterschiedliche Frequenzen.
- Darstellung: Amplituden und Phasen von Frequenz

### Fouriertransformation

- Fouriertransformation:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$



- Inverse Fouriertransformation:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{+i\omega t} d\omega$$

- Betrag des Fourierspektrums:

$$|F(\omega)| = \sqrt{\Re(F(\omega))^2 + \Im(F(\omega))^2}$$

- Phase des Spektrums:

$$\phi(F(\omega)) = \arctan \frac{\Im(F(\omega))}{\Re(F(\omega))}$$

### Eigenschaften

- Linearität:

$$F(c_1 f_1 + c_2 f_2) = c_1 F(f_1) + c_2 F(f_2)$$

- Differentiation:

$$F(f^{(n)}) = (i\omega)^n F(f)$$

- Verschiebung:

$$\begin{aligned} F(f(t-T)) &= e^{-i\omega T} F(f) && \text{(Zeit)} \\ F(e^{-i\omega_0 t} f(t)) &= F(\omega - \omega_0) && \text{(Frequenz)} \end{aligned}$$

### Zusammenhänge

Signal	$\Rightarrow$	Transformierte
Transformierte	$\Leftarrow$	Signal
diskret	$\Leftrightarrow$	periodisch
reell	$\Leftrightarrow$	gerade
imaginär	$\Leftrightarrow$	ungerade

### Typische Fouriertransformationen

- Sinusfunktion:

$$f(x) = a \sin(2\pi\alpha x) \quad \Leftrightarrow \quad F(\omega) = \frac{a}{2} i\delta(\omega + \alpha) - \frac{a}{2} i\delta(\omega - \alpha)$$

- Cosinusfunktion:

$$f(x) = a \cos(2\pi\alpha x) \quad \Leftrightarrow \quad F(\omega) = \frac{a}{2} \delta(\omega + \alpha) + \frac{a}{2} \delta(\omega - \alpha)$$

(Sinus- und cosinusförmige Signale werden im Frequenzbereich als zwei Impulse wiedergegeben.)

- Impulszug:

$$f(x) = \sum_{\nu=-\infty}^{\infty} \delta(x - \nu T) \quad \Leftrightarrow \quad F(\omega) = \frac{1}{T} \sum_{\nu=-\infty}^{\infty} \delta(\omega - \frac{\nu}{T})$$

### Anmerkungen

- Periodisch unendlich ausgedehntes Signal wird im Fourierbereich als endliche Bandbreite dargestellt.
- Endliches Signal kann zu unendlichem Spektrum führen. Approximation nötig!
- Je stärker die Übergänge im Signal, desto höher die Frequenz der Komponenten, um so breiter das Spektrum.

## 1.4 Fourierreihen

### Fourierreihen und -transformationen

Problem: Beide kontinuierlich, schwer digital darzustellen

- Diskrete Zeit (*discrete time*) Fouriertransformation:  
Eingabe: diskret, aperiodisch  
Ergebnis: periodisch, kontinuierlich
- Diskrete Fouriertransformation (DFT, FFT):  
Eingabe: periodisch, diskret  
Ergebnis: diskret, periodisch  
Spezialfall der Z-Transformation. Periodische Eingabe? Man schneidet ein Fenster aus und tut so, als ob es so periodisch unendlich fortgesetzt wird.

### Fourierreihenzerlegung

- Periodisches Signal
- Signal:

$$c_k = \frac{1}{2L} \int_{-L}^{+L} f(x) e^{-i\pi \frac{k}{L} x} dx \quad \text{wobei} \quad f(x) = \sum_{k=-\infty}^{+\infty} c_k e^{i\pi \frac{k}{L} x}$$

- Komplexe Schreibweise:

$$e^{ix} = \cos(x) + i \sin(x)$$

- Parameter: Amplitude, Phase, Frequenz (Periode)

### Fourierreihe für Rechteckfunktion

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left( a_n \cos\left(\frac{2\pi}{T}nx\right) + b_n \sin\left(\frac{2\pi}{T}nx\right) \right)$$

Beispiel für Fourierreihenentwicklung:

Für  $f(x) = \text{sign}(\sin(x))$  gilt:

$$\begin{aligned} a_n &= \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \cdot \cos(nx) dx = 0 \\ b_n &= \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \cdot \sin(nx) dx = \frac{2}{n\pi} (1 - (-1)^n) \end{aligned}$$

und somit

$$f(x) = \frac{4}{\pi} \left( \sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \dots \right)$$

## 1.5 Aliasing

In der Signalverarbeitung treten Alias-Effekte beim Digitalisieren analoger Signale auf.

Damit das Ursprungssignal korrekt wiederhergestellt werden kann, dürfen im abgetasteten Signal nur Frequenzanteile vorkommen, die weniger als halb so groß wie die Abtastfrequenz sind. Wird dieses Abtasttheorem verletzt, werden Frequenzanteile, die größer sind als die halbe Abtastfrequenz als niedrigere Frequenzen interpretiert. Die hohen Frequenzen geben sich sozusagen als jemand anderes aus, daher die Bezeichnung Alias. Die halbe Abtastfrequenz wird als Nyquist-Frequenz bezeichnet.

Falls es nicht zu vermeiden ist, dass hohe Frequenzen im Eingangssignal vorhanden sind, wird das Eingangssignal zur Unterdrückung von Alias-Effekten durch einen Tiefpass gefiltert (Anti-Aliasing-Filter), wobei die aktive Filterwirkung dieses Abschneidens der hohen Frequenzen eindeutiger mit Hözensperre, Höhenfilter, High Cut und Treble Cut beschrieben wird.

**Abtast/Sampling Theorem**

$$\Delta x \leq \frac{1}{2\omega}$$

- Die Samplingfrequenz muss mindestens zweimal so gross sein als die höchste im Signal vorkommende Frequenz  $\omega$  (*Cutoff-frequency*).
- Um Aliasing zu vermeiden muss das Abtasttheorem eingehalten werden.
- Ist das Abtasttheorem eingehalten, kann ein Signal durch inverse Fouriertransformation vollständig rekonstruiert werden.

**Abtasten in der Praxis**

- Abtasten nur in endlichem Intervall möglich.
- Spektrum lokal (in kleinem Intervall interessant)
  - Multiplikation des Signals mit Fensterfunktion
  - Faltung des Fensterspektrums mit Signalspektrum
  - Ungenauigkeiten, genaue Rekonstruktion nicht mehr möglich
- Ausnahme: Signal ist periodisch und bandbegrenzt

**Behebung von Aliasing**

Möglichkeiten zur Behebung von Aliasing:

- Samplefrequenz erhöhen. Nachteil: erhöhtes Datenaufkommen
- Bandbegrenzen des Originalsignals durch Tiefpassfilter (Anti-Aliasing Filter). Nachteil: Informationsverlust

**1.6 Korrelation**

- Eindimensionale Kreuzkorrelation:

$$R_{f,g}(m) = \sum_i f(i)g(i-m)$$

- Zweidimensionale Kreuzkorrelation:

$$R_{f,g}(m) = \sum_i \sum_j f(i,j)g(i-m, j-n)$$

- Autokorrelation = Kreuzkorrelation mit sich selbst:

$$\begin{aligned} A(m) &= \sum_i f(i)f(i-m) \\ A(m,n) &= \sum_i \sum_j f(i,j)f(i-m, j-n) \end{aligned}$$

### 1.7 Schablonenanpassung (*Template Matching*)

- Eine einfache Form der Klassifikation.
- Ziel: Ein Muster zu erkennen das einem abgespeicherten Beispiel ähnlich ist.
- Maß der Übereinstimmung ist der Absolutbetrag zwischen Muster und Schablone (zentriert an  $(m, n)$ ):

$$M_{f,g}(m, n) = \sum_i \sum_j |f(i, j)g(i - m, j - n)|$$

- Abstand  $E(m, n)$  ist definiert als Quadrat der Kreuzkorrelation:

$$E_{f,g}(m, n) = \left( \sum_i \sum_j f(i, j)g(i - m, j - n) \right)^2$$

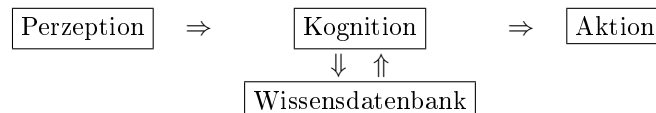
## 2 Kognition: Sensorik

### 2.1 Kognitionsbegriff

#### Elemente des Kognitionsbegriffs...

Wahrnehmung (Perzeption); Repräsentation, Modelle, Gedächtnis; Lernen, Verstehen; Denken, Problemlösen; Sprachgebrauch; Selbstreflexion; Sensomotorische Steuerung (Bewusstsein; Emotionen, Gefühle; Motivationen)

#### Elemente eines kognitiven Systems



**Perzeption** Aufnahme der Umwelt durch Sensoren (Taster, Kameras, Mikrofone, ...)

**Wissensdatenbank** Repräsentation gelernter Zusammenhänge und Reaktionen darauf

**Kognition** Auf der Wissensdatenbank arbeitende Algorithmen: Deduktion, Induktion, Lernen

**Aktion** Ausführende Komponente (Textausgabe, Robotersystem, ...)

### 2.2 Sensorik

**Sensor** System, das eine physikalische Größe und deren Änderung in geeignete (in unserem Fall elektrische) Signale umwandelt.

**Ziel** Erfassung der Umwelt (oft in nicht fest definierten Umgebungen)

#### Probleme

- Signalverarbeitung
- Sensorik liefert nur partielle Information: Wahl der Sensorik
- Sensorwerte sind vertauscht → statistische Verarbeitung
- Verwendung mehrerer Sensortypen in Multisensorsystemen: Fusion der der Meßwerte
- Modellierung: Abstraktionsstufen des Umweltmodells

#### Sensortypen

- Mechanische Sensoren (z.B. Kraftmesser)
- Temperatursensoren
- Chemische Sensoren (z.B. H<sup>+</sup>-Ionen Sensor)
- Akustische Sensoren
- Magnetische Sensoren
- Gassensoren (z.B. Lambda-Sonde)
- u.a.

#### 2.2.1 Sensor: Struktur

**Elementarsensor** Aufnahme einer Messgröße und Abbildung auf Signal.

**Integrierter Sensor** zusätzliche Signalaufbereitung: Verstärkung, Filterung, Linearisierung, Normierung

**Intelligenter Sensor** integrierter Sensor mit rechnergesteuerter Auswertung. Ausgang: verarbeitete Größe

## 2.3 Sensoren kognitiver Systeme

### 2.3.1 Sensortechnologie

Klassifikation von Sensoren nach ihrer Funktion:

**Interner Sensor** Erfassung innerer Zustände. Bsp.: Positionssensoren, Geschwindigkeitssensoren

**Externer Sensor** Information über Zustand der Umwelt. Bsp.: Taktile Sensoren, Abstandssensoren, Akustische Sensoren

### 2.3.2 Interne Sensorik

**Einfacher Positionssensor: Potentiometer** Lineare Position oder Drehposition  
Messung der abgefallenen Spannung  $U$  oder des Widerstands  $R$ , dann

$$U = (R/R_{ges}) \cdot U_{ges}$$

mit  $R_{ges}$  als Gesamtwiderstand und  $U_{ges}$  als Gesamtspannung  
 $U$  proportional zur Position  $P$ .  $P = const \cdot U$

**Positionssensoren: Opt. Codierer** Information über Position von Gelenken oder über die relative Position eines Fahrzeugs.  
Prinzip:

- Lichtstrahl wird auf Photodetektor ausgerichtet
- auf einer Scheibe angebrachtes codiertes Muster unterbricht periodisch den Strahl
- die Unterbrechungen werden gemessen und evtl. der Code festgestellt

Bauarten: inkrementelle Codierung, absolute Codierung

**Geschwindigkeitssensor: Tachogenerator** Gleichstromgenerator mit Ausgangsspannung  $V_0$ , Winkelgeschwindigkeit  $\omega_s$ :

$$V_0 = K_t \cdot \omega_s$$

Arbeitsprinzip: elektromagnetische Induktion (Faradays Gesetz) (*Analoges Ausgangssignal benötigt A/D-Wandler zur Weiterverarbeitung*)

**Beschleunigungssensoren: Silizium-Beschl.sensor** Messprinzip: seismische Masse  
Mikrosystemtechnik: Ätzen einer trägen Masse und deren Aufhängungen in Silizium.  
Auslenkung ändert mechanische Spannungen, welche den piezoresistiven Widerstandswert ändern. Widerstandswert messtechnisch erfassbar. Ursprüngliches Einsatzgebiet: Airbagauslösung im Kfz-Bereich. In der Robotik oft: Auslenkung zur Erdanziehung oder Lage im Raum.

### Lagesensor

- Messung der Orientierung zum Erdmagnetfeld und des Kipp- und Neigungswinkels.
- Orientierung: magnetischer Kompass
- Kipp- und Neigungswinkel: Inklinometer
- Funktionsprinzip der Wasserwaage
- Nachteil: träge
- Alternativ: Gyroskop zur Messung der Winkelbeschleunigung (mechanischer Kreisel oder Faseroptisch)

### 2.3.3 Externe Sensorik

**Taktile Sensoren** Umweltinformation durch direktes Berühren von Objekten (physikalischer Kontakt zwischen Roboter und Werkstück)

Einsatz: Ermittlung

- geometrischer Größen: Lage, Orientierung, Form
- physikalischer Größen: Kräfte, Momenten, Druck

Hierzu dienen: tastende Sensoren, gleitende Sensoren, Kraft-Momenten-Sensoren

Beispiele:

**künstliche Haut:** Array-Anordnungen

**Dehnmeßstreifen:** Dehnmeßstreifen verformt sich bei Krafteinwirkung. Veränderung des ohmschen Widerstands

**Drucksensoren:** Verkürzung von oder Entstehung neuer Strompfade bei Belastung

**Gleitsensoren:** Information über Oberflächenbeschaffenheit und geom. Struktur

**Kraft-Momenten-Sensoren** Erfassung der Kräfte und Drehmomente zwischen Effektor und Handhabungsobjekt

**Abstandssensoren** Messung von Abstand zwischen Sensor und Gegenstand (Typen: optische, akustische, Radar)

## 2.4 Beispiele

### Sensorik bei Robotergelenken

- Gelenkwinkelstellung bei Roboterarmen / Laufmaschinen-Beinen
- Überwachung der Positionsgenauigkeit bei Greifern (Karlsruher Hand, Barrett-Hand)  
früher: mechanische Anschläge  
heute: Bildverarbeitung, Achsencoder

### Sensorik bei autonomen Systemen

- Positionsbestimmung und Navigation in dynamischer Umgebung. Einsatz von: Laserscanner, Ultraschallsensoren, Bildverarbeitung
- Bsp.: autonome Systeme Mortimer, Viper (IPR), HelpMate (TRC)

### Sensorik bei Laufmaschinen

Sensorik zu Erfassung von: Hindernissen, Orientierung, Untergrundkontakt, Schwerpunkt

## 3 Klassifikation

### 3.1 Supervised – Unsupervised Training

**Supervised Training** Die zu erkennende Klasse ist für jede Auswahl in den Trainingsdaten bekannt. Benötigt a priori Wissen von nützlichen Eigenschaften und Kenntnis / Bezeichnung von jedem Trainingsmerkmal (Kosten!).

**Unsupervised Training** Die Klasse ist nicht bekannt und die Struktur muss automatisch herausgefunden werden.

### 3.2 Parametrisch – Nicht-parametrisch

Parametrisch:

- grundlegende Aufteilungswahrscheinlichkeit annehmen
- Parameter der Aufteilung abschätzen
- Beispiel: "Gauss Klassifikation"

Nicht-parametrisch:

- keine Aufteilung annehmen
- Fehlerwahrscheinlichkeit oder Fehlerkriterium direkt aus den Trainingsdaten berechnen
- Beispiele: Parzen Fenster,  $k$ -nächster Nachbar, Perzeptron

#### Parzen Fenster

Es werden keine Annahmen über die Verteilung gemacht, stattdessen wird  $p(x)$  direkt aus den Daten geschätzt.

- wähle ein Fenster mit dem Volumen  $V$
- zähle die Anzahl Samples, die in das Fenster fallen
- $p(x) \approx \frac{k/n}{V}$ ,  $k$  = Anzahl,  $n$  = Anzahl Samples

Probleme:

- Volumen zu groß  
⇒ Auflösung geht verloren
- Volumen zu klein  
⇒ unbeständig, schlechte Abschätzung

Setze

$$V_n = \frac{1}{\sqrt{n}}$$

#### $k$ -nächster Nachbar

Volumen als Funktion der Daten. Verwende die  $k$  nächsten Nachbarn für die Abschätzung. Setze

$$k = \sqrt{n}$$

Um Sample  $x$  zu klassifizieren:

- finde  $k$  nächste Nachbarn von  $x$
- bestimme die am häufigsten vorkommende Klasse in diesen  $k$  Samples
- ordne  $x$  dieser Klasse zu



Probleme: Für eine endliche Anzahl von Samples  $n$  sollte  $k$  möglichst

- groß sein für eine gute Abschätzung
- klein sein, um zu garantieren, dass alle  $k$  Nachbarn nah beieinander sind

Trainingsdatenbanken müssen groß sein.

### 3.3 Bayes Entscheidungstheorie

Bayes Regel:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \quad \text{wobei} \quad p(x) = \sum_j p(x|\omega_j)P(\omega_j)$$

A priori Wahrscheinlichkeit:

$$P(\omega_j)$$

A posteriori Wahrscheinlichkeit:

$$P(\omega_j|x)$$

Klassenbedingte Wahrscheinlichkeitsdichte:

$$p(x|\omega_j)$$

### 3.4 Zwei Klassen Fall

$$P(error|x) = \begin{cases} P(\omega_1|x) & \text{wenn wir uns für } \omega_2 \text{ entscheiden} \\ P(\omega_2|x) & \text{sonst} \end{cases}$$

Der Fehler ist minimiert, wenn wir uns entscheiden für:

- $\omega_1$  wenn  $P(\omega_1|x) > P(\omega_2|x)$   
 $\omega_2$  sonst
- $\omega_1$  wenn  $p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2)$   
 $\omega_2$  sonst

Mehr Klassen:

- $\omega_i$  wenn  $P(\omega_i|x) > P(\omega_j|x)$  für alle  $i \neq j$

### 3.5 Klassifizierende Diskriminanzfunktionen

$$g_i(x) \quad , \quad i = 1, \dots, c$$

Ordne  $x$  der Klasse  $\omega_i$  zu, wenn  $g_i(x) > g_j(x)$  für alle  $j \neq i$

$$\begin{aligned} g_i(x) &= P(\omega_i|x) \\ &= \frac{p(x|\omega_i)P(\omega_i)}{\sum_{j=1}^c p(x|\omega_j)P(\omega_j)} \\ g_i(x) &= p(x|\omega_i)P(\omega_i) \\ g_i(x) &= \log(p(x|\omega_i)) + \log(P(\omega_i)) \end{aligned}$$

### 3.6 Gauss Klassifizierer

Eindimensionale Normaldichte:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \sim N(\mu, \sigma^2)$$

Mehrdimensionale Dichte:

$$p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^t \Sigma^{-1} (\vec{x}-\vec{\mu})} \sim N(\vec{\mu}, \Sigma)$$

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^t \sum_i^{-1} (x - \mu_i) - \frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_i| + \log P(\omega_i)$$

Für jede Klasse  $i$  muss folgendes aus den Trainingsdaten berechnet werden:

- Kovarianz-Matrix  $\Sigma_i$
- Mittelwertsvektor  $\vec{\mu}_i$

#### Bestimmung der Parameter

- MLE, Maximum Likelihood Estimation
- Für den mehrdimensionalen Fall:

$$\vec{\mu} = \frac{1}{N} \sum_{k=1}^N \vec{x}_k$$

$$\Sigma = \frac{1}{N} \sum_{k=1}^N (\vec{x}_k - \vec{\mu})(\vec{x}_k - \vec{\mu})^T$$

### 3.7 Probleme beim Klassifikationsentwurf

Eigenschaften:

- Welche und wie viele Eigenschaften sollten gewählt werden?
- Beliebige Eigenschaften?
- Je mehr desto besser?
- Wenn zusätzliche Eigenschaften nicht nützlich sind, sollen sie dann automatisch ignoriert werden?

#### Der Unsegen der Dimensionalität

- Allgemein gilt: das Hinzufügen von Eigenschaften verschlechtert die Performance!
- Grund: Trainingsdaten vs. Anzahl der Parameter; beschränkte Trainingsdaten
- Lösung: Eigenschaften sorgfältig wählen; Dimension verringern; Principle Component Analysis

### 3.8 Risiko

- Es kann zu Entscheidungsverweigerungen kommen in mehrdeutigen Fällen ( $\rightarrow$  Bandbreite)
- Bewerte die Kosten für jede Entscheidung (etwas Kostenaufwendiger als anders)

$\Omega = \{\omega_1, \dots, \omega_s\}$   $s$  Zustände der Eigenschaften

$A = \{\alpha_1, \dots, \alpha_a\}$  mögliche Aktionen

**Verlustfunktion**  $\lambda(\alpha_i|\omega_j)$ : Verlust der Aktion  $\alpha_i$  beim gegebenen Zustand  $\omega_j$

$$P(\omega_j | \vec{x}) = \frac{P(\vec{x} | \omega_j)P(\omega_j)}{P(\vec{x})}$$

Angenommener Verlust von Aktion  $\alpha_i$ :

$$R(\alpha_i | \vec{x}) = \sum_{j=1}^s \lambda(\alpha_i|\omega_j)P(\omega_j | \vec{x}) \quad (\text{bedingtes Risiko})$$

Minimierung des angenommenen Verlusts indem man die Aktion  $\alpha_i$  wählt, die das bedingte Risiko minimiert.

#### Zwei Kategorien Fall

$$\lambda(\alpha_i|\omega_j) \triangleq \lambda_{ij}$$

$$\begin{aligned} R(\alpha_1 | \vec{x}) &= \lambda_{11}P(\omega_1 | \vec{x}) + \lambda_{12}P(\omega_2 | \vec{x}) \\ R(\alpha_2 | \vec{x}) &= \lambda_{21}P(\omega_1 | \vec{x}) + \lambda_{22}P(\omega_2 | \vec{x}) \end{aligned}$$

- Wähle  $\omega_1$ , wenn  $R(\alpha_1 | \vec{x}) < R(\alpha_2 | \vec{x})$
- Wähle  $\omega_1$ , wenn  $(\lambda_{21} - \lambda_{11})P(\omega_1 | \vec{x}) > (\lambda_{12} - \lambda_{22})P(\omega_2 | \vec{x})$
- Wähle  $\omega_1$ , wenn  $(\lambda_{21} - \lambda_{11})P(\vec{x} | \omega_1)P(\omega_1) > (\lambda_{12} - \lambda_{22})P(\vec{x} | \omega_2)P(\omega_2)$
- Wähle  $\omega_1$ , wenn  $\frac{p(\vec{x}|\omega_1)}{p(\vec{x}|\omega_2)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}} \cdot \frac{P(\omega_2)}{P(\omega_1)}$

## 4 Maschinelles lernen

### 4.1 Unsupervised Learning

- Datensammlung und Bezeichnung kostenaufwendig und zeitverschwenderisch.
- Die Charaktermerkmale der Muster können sich ändern im Verlauf der Zeit.
- Vielleicht hat man keine Einsicht in die Struktur der Daten.

⇒ Die Klassen sind nicht bekannt.

### 4.2 Mischdichten

- Beispiele kommen von  $c$  Klassen
- A priori Wahrscheinlichkeit  $P(\omega_j)$
- Annahme: Die Formen der Klassenbedingten PDF (probability density function – Wahrscheinlichkeitsdichtefunktion)  $P(X|\omega_j, \theta_j)$  sind bekannt
- Unbekannter Parametervektor  $\theta_1, \dots, \theta_c$

Problem: Haarige Mathematik

Vereinfachung / Approximation

Betrachte nur Mittel → Isodaten

- Wähle initiale  $\mu_1, \dots, \mu_c$
- Klassifiziere  $n$  Muster zu dem Mittel, das am nächsten liegt.
- Berechne die Mittel neu aus den Mustern der Klasse
- Mittel hat sich geändert? Gehe zu Schritt 2, sonst: stopp

Isodaten, Probleme:

- Die Wahl der initialen Mittel  $\mu$ .
- Wissen über die Anzahl der Klassen.
- Annahme über die Verteilung.
- Was bedeutet "nah"?

### 4.3 Clustering

- Ähnlichkeit
- Kriteriumsfunktion
- Muster in der selben Klasse sollten die Kriteriumsfunktion noch extremer machen, dies erfasst die Cluserqualität.
- Beispiel: Summe des Fehlerkriteriums:

$$J = \sum_{i=1}^c \sum_{\max} ||x - m_i||^2$$

## 4.4 Hierarchisches Clustering

- $c$  muss nicht bestimmt werden
- Mittel müssen nicht erraten werden

- (1) Initialisiere  $c := n$
- (2) Finde die nächsten Paare von eindeutigen Clustern  $X_i$  und  $X_j$
- (3) Vereinige sie und dekrementiere  $c$
- (4) Wenn  $c < C_{stop}$ : stoppen, sonst: gehe zu Schritt (2)

## 5 Neuronale Netze

### 5.1 Wieso neuronale Netze?

- Massiver Parallelismus
- Massive Randbedingungsangewandtheit für "krank"-definierte Eingaben
- Einfache Recheneinheiten
- Viele Prozesseinheiten, viele Verbindungen
- Einheitlichkeit ( $\rightarrow$  Sensorverschmelzung)
- Nichtlineare Klassifizierer / Abbilder ( $\rightarrow$  gute Performance)
- Lernfähig / anpassbar

### 5.2 Entscheidungsfunktion $g(x)$

$$\begin{aligned} g(\vec{x}) > 0 &\Rightarrow \text{Klasse A} \\ g(\vec{x}) < 0 &\Rightarrow \text{nicht Klasse A} \\ g(\vec{x}) = 0 &\Rightarrow \text{keine Entscheidung} \end{aligned}$$

$$g(\vec{x}) = \sum_{i=1}^n w_i x_i + w_0 = \vec{w}^T \vec{x} + w_0$$

$\vec{x} = (x_1, \dots, x_n)^T$  : Eigenschaftsvektor,  $\vec{w} = (w_1, \dots, w_n)^T$  : Gewichtsvektor,  $w_0$  : Schwellenwert

### 5.3 Lineare Diskriminantenfunktion

Hyperebene  $H$ :  $g(\vec{x}) = \sum_{i=1}^n w_i x_i + w_0 = \vec{w}^T \vec{x} + w_0 = 0$

$$\Rightarrow \vec{x} = q \frac{\vec{w}}{\|\vec{w}\|} + r \frac{\vec{w}}{\|\vec{w}\|} + \vec{x}_p$$

Vektor  $q \frac{\vec{w}}{\|\vec{w}\|}$  entspricht:  $g\left(q \frac{\vec{w}}{\|\vec{w}\|}\right) = 0 = q \|\vec{w}\| + w_0$

$$\Rightarrow q = -\frac{w_0}{\|\vec{w}\|}$$

Und mit  $g(\vec{x}) = \vec{w}^T q \frac{\vec{w}}{\|\vec{w}\|} + \vec{w}^T r \frac{\vec{w}}{\|\vec{w}\|} + \vec{w}^T \vec{x}_p + w_0 = -w_0 + r \|\vec{w}\| + w_0$  erhalten wir

$$r = \frac{g(\vec{x})}{\|\vec{w}\|}$$

## 5.4 Fisher-lineare Diskriminante

- Dimensionsreduktion
- Projiziert eine Menge von mehrdimensionalen Punkten auf eine Line  $y = \vec{w} \vec{x}$
- Die Fisher Diskriminante ist eine Funktion, die folgendes Kriterium maximiert

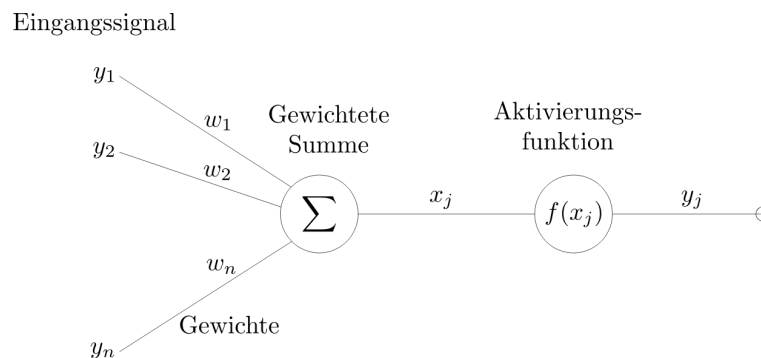
$$g(x) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1 + \tilde{s}_2}$$

wobei  $\tilde{m}_i = \frac{1}{n} \sum_{y \in Y_i} y$  : Mittel für projizierte Muster,  $\tilde{s}_i^2 = \sum_{y \in Y_i} (y - \tilde{m}_i)^2$  : Streuung für projizierte Muster

- Fisher's lineare Diskriminante:

$$\begin{aligned} \vec{w} &= s_w^{-1}(\vec{m}_1 - \vec{m}_2) \\ s_w &= s_1 + s_2 \\ s_i &= \sum_{x \in X_i} (\vec{x} - \vec{m}_i)(\vec{x} - \vec{m}_i)^T \end{aligned}$$

## 5.5 Das Perzeptron



### Lineare Diskriminantenfunktionen

- Keine Annahme über die Verteilung (Nicht-parametrisch)
- Lineare Entscheidungsflächen
- Start durch überwachtes Training (Klassen der Trainingsdaten gegeben)
- Diskriminantenfunktion:

$$g(x) = w_0 + \sum_{i=1}^n w_i x_i = \sum_{i=0}^n w_i x_i, \quad x_0 = 1$$

- $g(x)$  ergibt die Distanz von der Entscheidungsfläche
- Zwei Kategorien Fall:

$$\begin{aligned} g_1(x) > 0 &\Rightarrow \text{Klasse 1} \\ g_1(x) < 0 &\Rightarrow \text{Klasse 2} \end{aligned}$$

**Perzeptron**

$$g(x) = \sum_{i=0}^n w_i x_i \quad x_0 = 1 \quad \rightarrow \quad \text{finde } w$$

Alle Vektoren  $x_i$  sind korrekt benannt

$$\begin{aligned} \vec{w} \vec{x}_i &> 0 && \text{wenn } x_i \text{ zu } \omega_1 \text{ gehört} \\ \vec{w} \vec{x}_i &< 0 && \text{wenn } x_i \text{ zu } \omega_2 \text{ gehört} \end{aligned}$$

Oder setze alle Muster, die zu  $\omega_2$  gehören auf ihr negatives ( $-\vec{x}_i$ ). Dann sind alle Vektoren korrekt klassifiziert, wenn  $\vec{w} \vec{x}_i > 0$  für alle  $i$ .

**Kriteriumsfunktion vom Perzeptron**

$$J_P(\vec{w}) = \sum_{x \in X} (-\vec{w} \vec{x})$$

$X$  ist die Menge der falsch klassifizierten Merkmale. Da  $\vec{w} \vec{x}$  die Negation für falsch klassifizierte Merkmale ist, ist  $J_P(\vec{w})$  positiv. Sobald  $J_P = 0$  gilt, ist ein Lösungsvektor gefunden.

$J_P$  ist proportional zu der Summe der Distanzen der falsch klassifizierten Mustern zur Entscheidungsgrenze.

$$\nabla J_P = \sum_{x \in X} (-\vec{x}) \quad \vec{w}_{k+1} = \vec{w}_k + \zeta_k \sum_{x \in X} \vec{x}$$

**Lernen des Perzeptron**

Fragen:

- Wie soll man die Lernrate setzen?
- Wie soll man die initialen Gewichte setzen?

Probleme:

- Untrennbare Daten
- Trennbare Daten, aber welche Entscheidungsfläche
- Nicht linear trennbar

**Variationen**

Entspannungsprozedur:

$$J_q(\vec{w}) = \sum_{x \in X} (\vec{w}^t \vec{y})^2$$

Der Gradient ist kontinuierlich  $\rightarrow$  gleichmäßige Fläche. Manchmal geht es gegen Null!

$$J_q(\vec{w}) = \frac{1}{2} \sum_{x \in X} \frac{(\vec{x}^t \vec{w} - b)^2}{\|\vec{x}\|^2} \text{ begrenzt durch } b$$

**5.6 Generalisierung**

- Leistung auf Trainingsdaten interessiert uns nicht, sondern ungesehene real Welt...
- Wie gut funktioniert mein System in unvorhergesehenen Situationen?

**Generalisierungsfähigkeit**

Fähigkeit, das aus den Trainingsdaten Erlernte auf neue Daten (Testdaten) anzuwenden.

**Drei Gründe für schlechte Generalisierung**

1. Overfitting / Overtraining (zu lange trainiert)
2. Zu viele Parameter (weights) bzw. zu wenig Trainingsmaterial
3. falsche Netzwerkstruktur

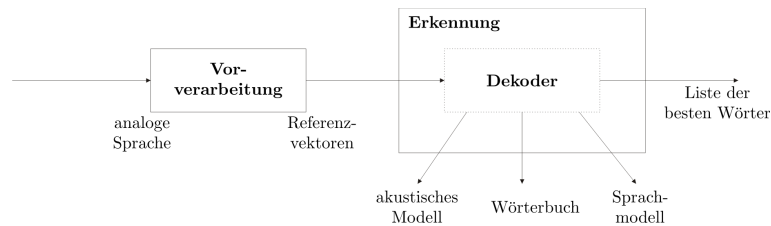
**Methoden zur Verbesserung der Generalisierung**

1. Training am Besten Punkt abbrechen
  - Aufteilung der Daten in: Trainings-, Crossvalidation- und Test-Menge
  - Wichtig: Testdaten von Trainingsdaten trennen! Wiederholtes Testen ist leichtes Trainieren (Tuning) → Crossvalidation Set
2. Reduzierung der Komplexität des Netzwerkes durch Regularisierung: Weight Decay, Weight Elimination, Optimal Brain Damage, Optimal Brain Surgeon
3. Schrittweises Vergrößern eines zu kleinen Netzes (konstruktiv): Cascade Correlation, Meiosis Netzwerke, ASO (Automativ Structure Optimization)



## 6 Spracherkennung

### 6.1 Spracherkennungssystem



### 6.2 Erkennung

**gegeben:** akustische Daten  $A = a_1, a_2, \dots, a_k$

**Ziel:** Wortsequenz  $W = w_1, w_2, \dots, w_n$  finden, so dass  $P(W | A)$  maximiert wird.

Wiederholung: **Bayes Regel**

$$P(W | A) = \frac{P(A | W) \cdot P(W)}{P(A)}$$

wobei  $P(A | W)$  : akustisches Modell (HMM),  $P(W)$  : Sprachmodell,  $P(A)$  : Konstante für einen kompletten Satz

### 6.3 Hidden Markov Modelle (HMM)

Ein Hidden Markov Modell ist ein stochastisches Modell, das sich durch zwei Zufallsprozesse beschreiben lässt.

Der erste Zufallsprozess entspricht dabei einer Markow-Kette, die durch Zustände und Übergangswahrscheinlichkeiten gekennzeichnet ist. Die Zustände der Kette sind von außen jedoch nicht direkt sichtbar (sie sind verborgen). Stattdessen erzeugt ein zweiter Zufallsprozess zu jedem Zeitpunkt beobachtbare Ausgangssymbole gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung. Die Aufgabe besteht häufig darin, aus der Sequenz der Ausgabesymbole auf die Sequenz der verborgenen Zustände zu schließen.

#### Elemente

Menge an Zuständen:	$S = \{S_0, S_1, \dots, S_N\}$
Übergangswahrscheinlichkeiten:	$P(q_t = S_i   q_{t-1} = S_j) = a_{ij}$
Ausgabewahrscheinlichkeitsverteilungen: (bei Zustand $j$ für Symbol $k$ )	$P(y_t = O_k   q_t = S_j) = b_j(k)$

#### HMM Probleme und Lösungen

**Evaluation:** *Problem:* Bei einem gegebenen Modell soll die Wahrscheinlichkeit einer speziellen Ausgabesequenz bestimmt werden.

*Lösung:* **Forward-Algorithmus** und **Viterbi-Algorithmus**

**Dekodierung:** *Problem:* Es soll die wahrscheinlichste Sequenz der Zustände bestimmt werden, die eine vorgegebene Ausgabesequenz erzeugt hat.

*Lösung:* **Viterbi-Algorithmus**

**Training:** *Problem:* Gegeben ist nur die Ausgabesequenz. Es sollen die Parameter des HMM bestimmt werden, die am wahrscheinlichsten die Ausgabesequenz erzeugen.

*Lösung:* **Forward-Backward-Algorithmus**

## 6.4 Evaluation

Wahrscheinlichkeit einer Ausgabesequenz  $O = O_1 O_2 \dots O_T$  bei einem gegebenen Hidden Markov Modell  $\lambda$  ist

$$\begin{aligned} P(O | \lambda) &= \sum_{\forall Q} P(O, Q | \lambda) \\ &= \sum_{\forall q_0, \dots, q_T} a_{q_0 q_1} b_{q_1}(O_1) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \end{aligned}$$

wobei  $Q = q_0 q_1 \dots q_T$  eine Folge von Zuständen ist

**Nicht praktisch, da die Zahl der Wege in  $O(N^T)$  liegt.** ( $N$ : Anzahl der Zustände im Modell,  $T$ : Anzahl der Ausgabesequenzen)

### Der Forward Algorithmus

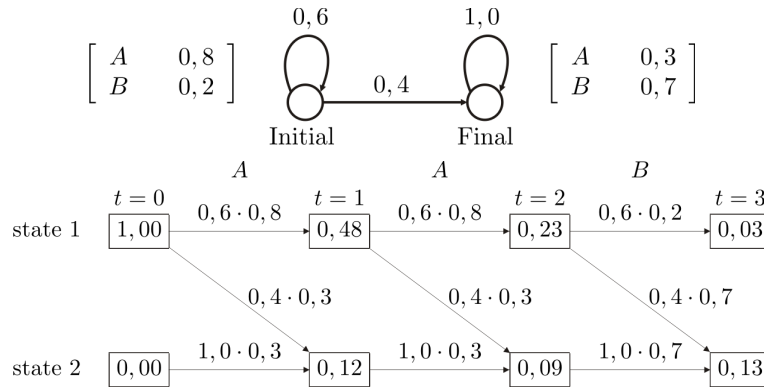
$$\alpha_t(j) = P(O_1 O_2 \dots O_t, q_t = S_j | \lambda)$$

rekursive Berechnung von  $\alpha$ :

$$\begin{aligned} \alpha_0(j) &= \begin{cases} 1 & \text{wenn } j \text{ Startzustand} \\ 0 & \text{sonst} \end{cases} \\ \alpha_t(j) &= \left( \sum_{i=0}^N \alpha_{t-1}(i) a_{ij} \right) b_j(O_t) \quad t > 0 \end{aligned}$$

( $P(O | \lambda) = \alpha_T(S_N)$ , Berechnung liegt in  $O(N^2 T)$ )

### Forward Trellis



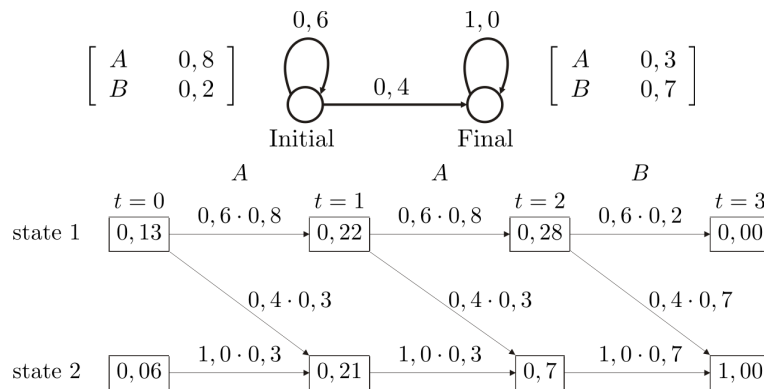
### Der Backward Algorithmus

$$\beta_t(i) = P(O_{t+1} O_{t+2} \dots O_T, q_t = S_i | \lambda)$$

rekursive Berechnung von  $\beta$ :

$$\begin{aligned} \beta_0(i) &= \begin{cases} 1 & \text{wenn } i \text{ Endzustand} \\ 0 & \text{sonst} \end{cases} \\ \beta_t(i) &= \sum_{j=0}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad t < T \end{aligned}$$

( $P(O | \lambda) = \beta_0(S_0) = \alpha_T(S_N)$ , Berechnung liegt in  $O(N^2 T)$ )

**Backward Trellis****6.5 Dekodierung****Der Viterbi Algorithmus**

- Finde die Zustandsfolge  $Q$ , die  $P(O, Q | \lambda)$  maximiert.
- Verläuft ähnlich wie der Forward Algorithmus, jedoch wird das Maximum anstatt der Summe berechnet.

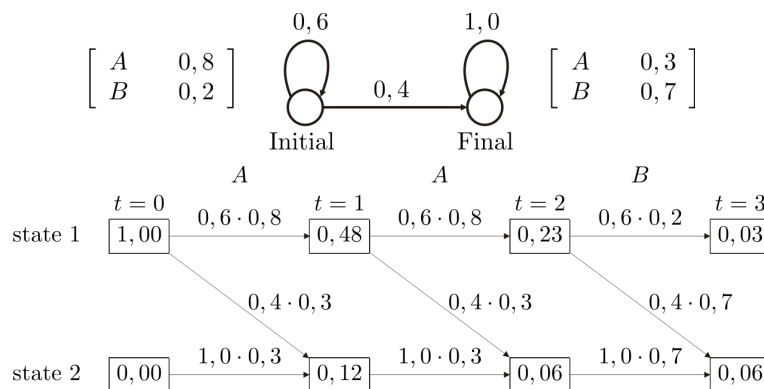
$$VP_t(i) = \max_{q_0, \dots, q_{t-1}} P(O_1 O_2 \dots O_t, q_t = i | \lambda)$$

rekursive Berechnung:

$$VP_t(j) = \max_{i=0, \dots, N} VP_{t-1}(i) a_{ij} b_j(O_t) \quad t > 0$$

$$P(O, Q | \lambda) = VP_T(S_N)$$

Speicher jedes Maximum für die Ablaufverfolgung am Ende.

**Viterbi Trellis****6.6 Training**

- Trainiere Parameter vom Hidden Markov Modell
  - $\lambda$  einstellen, um  $P(O | \lambda)$  zu maximieren
  - kein effizienter Algorithmus für das globale Optimum
  - ein effizienter iterativer Algorithmus findet das lokale Optimum
- Viterbi-Training
  - berechne den Viterbi-Weg mittels aktuellem Modell

- bewerte die Parameter neu, indem die Bezeichnung benutzt werden, die der Viterbi Algorithmus bestimmt hat
- Baum-Welch (Forward-Backward)
  - berechne Wahrscheinlichkeiten mittels aktuellem Modell
  - Filtere  $\lambda \rightarrow \lambda$  basierend auf den berechneten Werten
  - benutze  $\alpha$  und  $\beta$  vom Forward-Backward

### Der Forward-Backward Algorithmus

Wahrscheinlichkeit beim Übergang von  $S_i$  nach  $S_j$  zur Zeit  $t$  bei gegebenem  $O$ :

$$\begin{aligned}\xi_t(i, j) &= P(q_t = S_i, q_{t+1} = S_j \mid \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}\end{aligned}$$

### Baum-Welch Neubewertung

$$\begin{aligned}\bar{a}_{ij} &= \frac{\text{angenommene Anzahl an Übergängen von } S_i \text{ nach } S_j}{\text{angenommene Anzahl an Übergängen von } S_i} \\ &= \frac{\sum_{t=1}^T \xi_t(i, j)}{\sum_{t=1}^T \sum_{j=0}^N \xi_t(i, j)} \\ \bar{b}_j(k) &= \frac{\text{angenommene Dauer im Zustand } j \text{ mit Symbol } k}{\text{angenommene Dauer im Zustand } j} \\ &= \frac{\sum_{t: O_t=k} \sum_{i=0}^N \xi_t(i, j)}{\sum_{t=1}^T \sum_{i=0}^N \xi_t(i, j)}\end{aligned}$$

### Konvergenz des Forward-Backward Algorithmus

1. initialisiere  $\lambda = (A, B)$
2. berechne  $\alpha, \beta$  und  $\xi$
3. bewerte  $\bar{\lambda} = (\bar{A}, \bar{B})$  aus  $\xi$
4. ersetze  $\lambda$  durch  $\bar{\lambda}$
5. wenn nicht konvergiert, gehe zu 2

Es kann gezeigt werden, dass  $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$  gilt, wenn nicht  $\lambda = \bar{\lambda}$  gilt.

## 6.7 Sprachmodelle

### Basierend auf die Grammatik

- Bestimme Grammatik für mögliche Satzmuster
- Vorteile: Lange Historie / Kontext; es wird keine große Textdatenbank benötigt
- Problem: Grammatik zu schreiben ist sehr aufwendig; unflexibel: nur erstellte Muster können erkannt werden

**N-Gram**

- nächstes Wort wird anhand der Historie bestimmt
- die Historie ist approximiert durch die letzten 2 oder 3 (allgemein  $n$ ) Wörter
- alles vor Wort  $w_{i-n+1}$  wird in eine Äquivalenzklasse plaziert
- schliesslich ist die Wahrscheinlichkeit für das nächste Wort gegeben durch
  - Trigram:  $P(w_i | w_{i-1}, w_{i-2})$
  - Bigram:  $P(w_i | w_{i-1})$
  - Unigram:  $P(w_i)$
- Vorteile:
  - trainierbar auf großen Textdatenbanken
  - "milde" Vorhersage
  - kann direkt kombiniert werden mit einem Akustikmodell
- Problem: benötigt eine große Textdatenbank für jedes Gebiet

**Objektive Bewertung der Qualität von Sprachmodellen**

Genau ein Sprachmodell ist besser als eine Alternative, wenn die Wahrscheinlichkeit  $\hat{P}(w_1, w_2, \dots, w_n)$  mit dem es den Großteil eines Tests  $W$  erzeugen würde, größer ist. Aber:

$$\hat{P}(w_1, w_2, \dots, w_n) = \prod_{i=1}^n Q(w_i | \Psi(w_1, \dots, w_{i-1}))$$

ein gutes Qualitätsmaß ist das LOGPROB:

$$\hat{H}(W) = \frac{1}{n} \sum_{i=1}^n \log_2 Q(w_i | \Psi(w_1, \dots, w_{i-1}))$$

Wenn Wörter einheitlich durch Zufall aus einem Vokabular der Größe  $V$  von "Sprachmechanismen" erzeugt wurden, dann gilt

$$Q(w_i | \Psi(w_1, \dots, w_{i-1})) = \frac{1}{V}$$

und

$$2^{\hat{H}(W)} = 2^{\log V} = V$$

Also definieren wir die Perplexität eines Sprachmodells folgendermaßen:

$$PP(W) = 2^{\hat{H}(W)}$$

und interpretieren sie als "Abzweigungsfaktor" der Sprache, wenn  $\Psi$  gegeben ist.

**Erfasste Erkennungspersormance**

Wortfehlerrate

$$WER = \frac{\#Ins + \#Del + \#Sub}{N}$$

## 7 Maschinensehen

### 7.1 Bildgenerierung

- Heute werden oftmals Digitalkameras verwendet (meist CCD, aber auch CMOS)
- Anschluss erfolgt über: Firewire (IEEE1394), USB, Camera Link, ...
- Kameras liefern direkt digitalisierte Bilddaten
  - Format je nach Kamera und Modus unterschiedlich
  - Bei S/W-Kameras meist 8bit Graustufen
  - Bei Farbkameras entweder als bayer-Pattern oder meist bereits konvertiert als RGB24, YUV422, etc.
- Kameras unterscheiden sich durch:
  - Bildqualität (Qualität CCD-Chip, aber auch grundlegend: Auswahl Linse/Objektiv)
  - Graustufen- oder Farbkamera
  - Auflösung
  - Bei Farbkameras: Welche Farbkodierungen sind verfügbar? (8bit, 16bit, 24bit)
  - Wichtig je nach Anwendung: Welche maximale Framerate ist bei welchem Bildformat noch verfügbar? (z.B. 15/30/60/120/200 Hz)

### 7.2 Bildrepräsentation

**Monochrombild:** Diskrete Funktion

$$\begin{aligned} \text{Img} : [0 \dots n-1] \times [0 \dots m-1] &\rightarrow [0 \dots q] \\ (u, v) &\mapsto \text{Img}(u, v) \end{aligned}$$

Üblich:  $q = 255$ ;  $n = 640$ ,  $m = 480$  (VGA) oder  $n = 768$ ,  $m = 576$  (PAL)

**Farbbild:**

- Viele verschiedene Farbmodelle für unterschiedliche Anwendungen
- Klassifikation nach erreichbarem Farbraum

**Beispiele:**

- S/W, Grauwertstufen
- RGB-Modell: speziell für Monitore (Phosphor-Kristalle), sehr üblich

$$\text{Img}(u, v) \in \mathbb{R}^3 = (r, g, b)^T$$

- HSI (Hue, Saturation, Intensity): speziell für Farbsegmentierung
- CIE: physikalisch (Wellenlänge)
- CMYK- Modell: Farbdrucker (subtraktive Farbmischung)
- YIQ: Fernsehmodell

**RGB-Modell**

$$\begin{aligned} \text{Img} : [0 \dots n - 1] \times [0 \dots m - 1] &\rightarrow [0 \dots R] \times [0 \dots G] \times [0 \dots B] \\ (u, v) &\mapsto \text{Img}(u, v) = (r, g, b) \end{aligned}$$

- additive Farbmischung
- drei Farbwerte: Rot, Grün, Blau  
oft:  $256 \times 256 \times 256$  Nuancen  
( $R = G = B = 255$ , 8Bit, "RGB24") = 16,8 Mio. Farben
- oft verwendet von Kamera-Treibern

**HSI- /HSV-Modell**

- Hue (Farbnuancen), Saturation (Sättigung), Intensity/Value (Helligkeit)
- trennt Helligkeit vom Farbwert  $\Rightarrow$  unempfindlich gegen Beleuchtungsänderungen
- Umrechnung von RGB nach HSI (falls  $R = G = B$ , dann ist  $H$  undefiniert; falls  $R = G = B = 0$ , dann ist  $S$  undefiniert)

$$\begin{aligned} c &= \arccos \frac{2R - G - B}{2\sqrt{(R - G)^2 + (R - B)(G - B)}} \\ H &= \begin{cases} c & \text{falls } B < G \\ 360^\circ - c & \text{sonst} \end{cases} \\ S &= 1 - \frac{3}{R + G + B} \min(R, G, B) \\ I &= \frac{1}{3}(R + G + B) \end{aligned}$$

**Hinterlegung**

- Hinterlegung eines 8bit Graustufen-Bildes im Speicher
  - Pixel werden zeilenweise, von oben links nach unten rechts, linear abgelegt (Achtung: z.B. bei Bitmaps von unten links nach oben rechts)
  - Graustufen-Kodierung: ein Byte pro pixel; 0 schwarz, 255 weiß, dazwischen Graustufen
- Hinterlegung eines RGB24 Farbbildes im Speicher
  - Pixel werden zeilenweise, wie beim Graustufen-Bild, abgelegt
  - Farbkodierung: drei Bytes pro Pixel; für jeden Kanal gilt: 0 minimale, 255 maximale Intensität, dazwischen Nuancen

**Grauwert-Transformation**

Transformation von RGB24 nach 8bit Graustufen:

- Eine Möglichkeit:  $g = (R + G + B)/3$ , aber: menschliches Auge ist am empfindlichsten gegenüber der Farbe Grün.
- Üblicherweise wird deshalb verwendet:

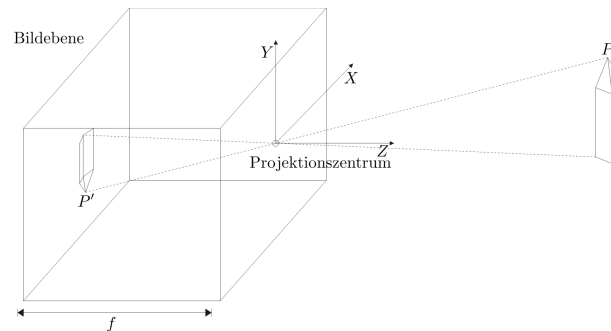
$$g = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

### Bayer-Pattern

Sehr hochwertige Kameras, wie z.B. zum Filmen verwendet, besitzen drei Chips pro Pixel. Die meisten Farbkameras haben einen Chip pro pixel, der gegenüber der Farbe Rot, Grün oder Blau empfindlich ist. Bei "Ein-Chip-Kameras" wird meist das Bayer-Pattern verwendet:

- Um nach RGB24 zu konvertieren, muss interpoliert werden.
- Die Empfindlichkeit einer Ein-Chip-Kamera ist um den Faktor 3 niedriger als die einer reinen Graustufen-Kamera.

### Lochkameramodell



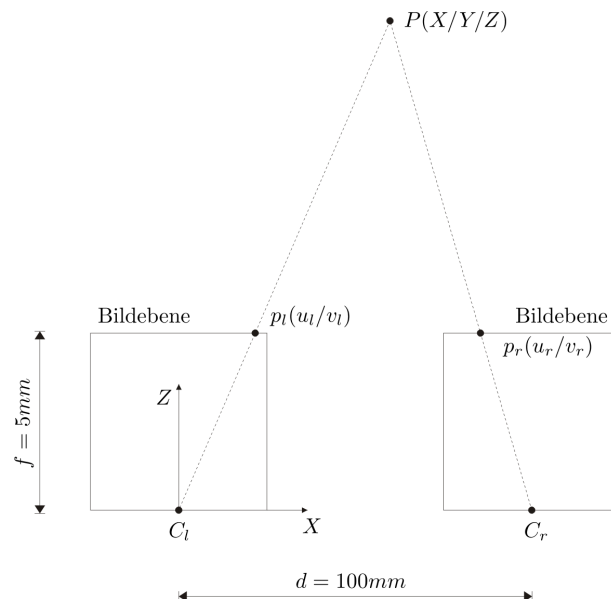
Projektion eines Szenenpunktes  $P = (X, Y, Z)$  auf einen Bildpunkt  $p = (u, v, w)$  mit Brennweite  $f$ :

$$\frac{-u}{f} = \frac{X}{Z} \quad , \quad \frac{-v}{f} = \frac{Y}{Z} \quad , \quad w = -f \quad \Rightarrow \quad X = -\frac{uZ}{f} \quad , \quad -\frac{vZ}{f}$$

$$p = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} u \\ v \\ -f \end{pmatrix} = -\frac{f}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = -\frac{f}{Z} P$$

Bei der Projektion geht die Z-Komponente verloren!

Beispiel mit 2 Kameras:



### Mattscheibenmodell

Einziger Unterschied Mattscheibenmodell  $\Leftrightarrow$  Lochkameramodell:



- Projektionszentrum  $C$  liegt hinter der Bildebene
- dadurch: keine Spiegelung (Minuszeichen entfallen)

### Linsensysteme

Moderne Kameras benutzen Linsen.

Vorteil:

- mehr Lichteinfall

Nachteile:

- nur Teile der Szene können gleichzeitig fokussiert werden
- Linsenkrümmung führt zu Bildverzerrungen

Bild eines Objekts in Tiefe  $Z$  wird in Abstand von der Linse  $Z'$  gebildet mit:

$$\frac{1}{Z} + \frac{1}{Z'} = \frac{1}{f} \quad f : \text{Brennweite}$$

- Bei Einstellung der Bildebene auf  $Z'_0$  wird ein gewisser Objektbereich um  $Z_0$  "ausreichend scharf" abgebildet.
- Beim Menschen: Änderung der Linsenbrennweite.
- Bei Kameras: Verschieben der Linse gegenüber Bildebene
- Vereinfachung wegen  $Z \gg Z'$ :  
 $\Rightarrow$  Perspektivgleichungen aus dem Lochkameramodell können weiterverwendet werden!

$$\frac{1}{Z} + \frac{1}{Z'} \approx \frac{1}{Z'} \quad \Rightarrow \quad \frac{1}{Z'} \approx \frac{1}{f}$$

## 7.3 Bildverarbeitung

### Konzepte

- Homogene Punktoperationen
- Histogrammauswertung
- Filterung
- Geometrische Operatoren

$\rightarrow$  Unterdrückung von Störungen, "Verschönern" von Bildern, Verformen von Bildern

### Homogene Punktoperatoren

Anwendung:

$$Img'(u, v) = f(Img(u, v))$$

Unabhängig von der Position bzw. den Nachbarn des Pixels. Implementierung der Funktion  $f$  oftmals als Look-Up-Table (Hardware).

## Affine Punktoperatoren

Affine Punktoperatoren:

$$\begin{aligned} f : [0 \dots q] &\rightarrow [0 \dots q] \\ x &\mapsto ax + b \end{aligned}$$

Parameter  $a$  und  $b$  legen die Funktion fest. Anwendungen:

- Kontrasterhöhung:  $b = 0, a > 0$   
Kontrastverminderung:  $b = 0, a < 0$
- Helligkeitserhöhung:  $b > 0, a = 1$   
Helligkeitsverminderung:  $b < 0, a = 1$
- Invertierung:  $b = q, a = -1$
- Kombinationen

Spreizung:

- Intensitäten im Bild werden linear auf gesamten Bereich ausgedehnt
- Abbildungsvorschrift:

$$Img'(x, y) = q * \frac{Img(x, y) - min}{max - min}$$

- Als affine Punktoperation:

$$Img'(x, y) = \frac{q}{max - min} * Img(x, y) - \frac{q * min}{max - min}$$

## Nicht-Affine Punktoperationen

Beliebige Abbildungsfunktion

$$f : [0 \dots q] \rightarrow [0 \dots q]$$

Anwendung:

- Ausgleich von Sensor-Nichtlinearitäten
- Gewichtung
- Binarisierung

## Histogramme

Histogrammfunktion: gibt die Häufigkeit eines Selektionsmerkmals an. Normalerweise: Grauwert

$$H_{Img}(x) = \#(u, v) : Img(u, v) = x, x \in [0 \dots q]$$

Histogrammdehnung:

- Verbesserung der Spreizung
- Anstatt  $min$  und  $max$  werden Quantile verwendet
- Akkumuliertes Histogramm berechnen:

$$H_a(x) := \sum_{k=0}^x H(k)$$

- $H_q(p_{min})$  und  $H_q(p_{max})$  (z.B.  $p_{min} = 0.1$  und  $p_{max} = 0.9$ ) bestimmen mit:

$$H_q(p) := \inf\{x \in \{0, \dots, q\} : H_a(x) \geq p * H_a(q)\}$$

- Falls  $H_q(p_{min}) = H_q(p_{max})$ : Bild ist homogen
- Sonst:

$$a := \frac{q}{H_q(p_{max}) - H_q(p_{min})}$$

$$b := -\frac{q * H_q(p_{min})}{H_q(p_{max}) - H_q(p_{min})}$$

$$Img' \leftarrow \text{AffinePunktoperation}(Img, a, b)$$

Histogrammausgleich:

- Histogrammausgleich: bessere Anpassung an das Sehvermögen des Menschen; kein Informationsgewinn  
 $H_n(0) := H_{Img}(0)$   
 for  $x := 1$  to  $q$  do  
 $H_n(x) := H_n(x-1) + H_{Img}(x)$   
 endfor
- Automatische Kontrast- und Helligkeitsapassung durch Histogrammanalyse  
 $H_n(0) := H_{Img}(0)$   
 for  $x := 1$  to  $q$  do  
 $H_n(x) := H_n(x) \cdot \frac{q}{width \cdot height}$   
 endfor
- Bei bereits berechnetem Histogramm  $H_{Img}(x)$  kann der Histogrammausgleich durch den folgenden Algorithmus berechnet werden:  
 $H_n(0) := H_{Img}(0)$   
 for  $y := 0$  to  $height - 1$  do  
 for  $x := 0$  to  $width - 1$  do  
 $Img'(u, v) := H_n(Img(u, v))$   
 endfor  
 endfor

## Heute

- Bildanalyse durch Frequenzanalyse
- Filter in Frequenzbereich
- Filter in Ortsbereich

## 7.4 Bildanalyse durch Frequenzanalyse

- (Grauwert-) Bilder lassen sich signaltheoretisch als Summe verschiedenfrequenter Signale betrachten.
- Niedrige Frequenzen: Schwache Grauwertübergänge
- Hohe Frequenzen: Scharfe Grauwertübergänge
- Nützlich z.B. zum Finden gerader Linien

→ Fourier-Analyse!

## 2-Dim Fouriertransformation

Kontinuierliches 2-dimensionales Signal:

$$F(u, v) = \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} f(x, y) e^{-2i\pi(ux+vy)} dx dy$$

Diskretes 2-dimensionales Signal:

$$F(u, v) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f[x, y] e^{-2i\pi(ux+vy)T}$$

## Fouriertransformation in der Bildverarbeitung

DFT bei Bild mit ImgSize  $[0 \leq x \leq M][0 \leq y \leq N]$ :

$$F(u, v) = \sum_{y=0}^{N-1} \left( \sum_{x=0}^{M-1} f[x, y] e^{-2i\pi \frac{ux}{M}} \right) \cdot e^{-2i\pi \frac{vy}{N}}$$

DFT bei quadratischem Bild mit ImgSize  $[0 \leq x \leq N][0 \leq y \leq N]$ :

$$\sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f[x, y] e^{-\frac{2i\pi(ux+vy)}{N}}$$

Anschaulich: Durchführung der 1D-DFT auf jeder Zeile und Speicherung der Daten in Matrix (innere Klammer). Durchführung der 1D-DFT auf jeder Spalte der ermittelten Matrix.

- Gewichtete Summation aller Bildpunkte
- Zerlegung des Bildes in Sinus- und Cosinusfunktionen
- Je weiter ein Punkt im Spektrum vom Bildmittelpunkt entfernt ist, desto höher ist seine darstellende Frequenz  $u$  bzw.  $v$ .
- D.h. im Bildinneren tiefe Frequenzen, in äußeren Bereichen hohe Frequenzen
- Anwendung:
  - Bildanalyse (z.B. Muster-, Geschwindigkeitserkennung)
  - Bildfilterung (z.B. Tiefpass)
  - Bildkompression (z.B. in jpeg Format)

## Fouriertransformation

- Die Variablen  $u$  und  $v$  heissen Frequenzvariablen
- $F(u, v)$  ist komplexe Funktion
- $F(u, v)$  ist darstellbar als 2 Bilder
  - in Realteil und Imaginärteil

$$F(u, v) = R(u, v) + I(u, v)$$

- oder Betrag (auch Spektrum genannt, oft logarithmisch dargestellt) und Phase

$$F(u, v) = |F(u, v)| \cdot e^{i\varphi(u, v)}$$

- Quadrat des Spektrums heisst spektrale Dichte.

## 7.5 Bildbearbeitung

- Durch Filter im Ortsbereich oder Transferfunktionen im Frequenzbereich.
- Ortsbereich:
  - Manipulation von Grauwerten
  - anschaulich
  - häufig: Punktoperationen, Glättung
- Frequenzbereich:
  - Manipulation der Frequenzanteile
  - keine unmittelbare bildliche Vorstellung
  - häufig: starke Glättung, frequenzselektive Filter

### Bildbearbeitung im Ortsbereich

Faltung zweier Funktionen 1D kontinuierlich:

$$h(x) = f(x) * g(x) = \int_{a=-\infty}^{\infty} f(a)g(x-a)da$$

Faltung zweier Funktionen 1D zeitdiskret:

$$h[x] = f[x] * g[x] = \sum_{a=-\infty}^{\infty} f[a]g[x-a]$$

Faltung zweier Funktionen 2D kontinuierlich:

$$h(x, y) = f(x, y) * g(x, y) = \int_{a=-\infty}^{\infty} \int_{b=-\infty}^{\infty} f(a, b)g(x-a, y-b)dadb$$

Faltung zweier Funktionen 2D zeitdiskret:

$$h[x, y] = f[x, y] * g[x, y] = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f[a, b]g[x-a, y-b]$$

### Faltung

- Bildmatrizen werden in den relevanten Randbereichen mit Nullen gefüllt.
- Der neue Bildwert ist eine gewichtete Summe der Pixel die unter der gespiegelten Matrix liegen.
- Als Gewichte dienen die Matrizenwerte.
- Bildfilterung ist die Faltung eines Bildes mit einer Filtermatrix bzw. Maske.
- Die Transferfunktion  $H(u, v)$  ist die Fouriertransformierte der Filterfunktion  $h(x, y)$ .

### Filteroperationen

- Glättungsoperatoren (Rauschelimination): Mittelwertfilter, Gauß
- Kantendetektoren: Prewitt, Sobel, Roberts, Laplace
- kombinierte Filter: Laplacian of Gauß

## 7.6 Filter

### Gauß

Gaußfilter mit Radius  $\sigma$  definiert durch die Gaußfunktion

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

### Mittelwertfilter

Ziel: Störunterdrückung

Beispiel: Durchschnitt aus 8-Umgebung und Punkt. Größe beliebig wählbar.

$$m'(x, y) = \sum_{j=-1}^1 \sum_{i=-1}^1 m(i, j) \cdot p(x - j, y - i)$$

### Anwendung der nachfolgenden Filter

am Beispiel des Sobel-X Filters:

$$\begin{array}{|c|c|c|c|c|} \hline f(x, y) & & & & \\ \hline a & b & c & d & e \\ \hline f & g & h & i & j \\ \hline k & l & m & n & o \\ \hline p & q & r & s & t \\ \hline u & v & w & x & y \\ \hline \end{array}
 \quad * \quad
 \begin{array}{|c|c|c|} \hline g(x, y) & & \\ \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|c|c|} \hline h(x, y) & & & & \\ \hline & & & & \\ \hline & g' & h' & i' & \\ \hline & l' & m' & n' & \\ \hline & q' & r' & s' & \\ \hline & & & & \\ \hline \end{array}$$

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

$$g' = -a + c - 2f + 2h - k + m$$

$$h' = (d + 2l + n) - (b + 2g + l)$$

$$l' = (e + 2j + o) - (c + 2h + m)$$

### Prewitt

#### Prewitt-X Filter

$$P_x = \frac{\partial g(x, y)}{\partial x}$$

approximiert durch

$$p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

Kantendetektion: vertikal gut, horizontal schlecht

#### Prewitt-Y Filter

$$P_y = \frac{\partial g(x, y)}{\partial y}$$

approximiert durch

$$p_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Kantendetektion: vertikal schlecht, horizontal gut

### **Prewitt-Operator**

- Kombination der Prewitt-Filter zur Bestimmung des Grauwertgradientenbetrages  $M$ :

$$M \approx \sqrt{P_x^2 + P_y^2}$$

- Danach: Schwellwertfilterung

### **Sobel**

#### **Sobel-X Filter**

$$S_x = \frac{\partial g(x, y)}{\partial x}$$

approximiert durch

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

Kantendetektion: vertikal gut, horizontal schlecht

#### **Sobel-Y Filter**

$$S_y = \frac{\partial g(x, y)}{\partial y}$$

approximiert durch

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

Kantendetektion: vertikal schlecht, horizontal gut

### **Sobel-Operator**

- Kombination der Sobel-Filter zur Bestimmung des Grauwertgradienten-Betrages  $M$ :

$$M \approx \sqrt{S_x^2 + S_y^2}$$

- Danach: Schwellwertfilterung

### **Roberts**

$$R(g(x, y)) = |R_x(g(x, y))| + |R_y(g(x, y))|$$

wobei

$$R_x = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad , \quad R_y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Kantendetektion: diagonal gut

## Laplace

Laplace-Operator:

$$\nabla^2 g(x, y) = \frac{\partial^2 g(x, y)}{\partial x^2} + \frac{\partial^2 g(x, y)}{\partial y^2}$$

wobei

$$\nabla^2 \approx \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Kantendetektion: Nulldurchgänge markieren Kanten, Subpixelgenauigkeit erreichbar  
 Näherung des Laplace-Operators:

$$\nabla^2 \approx \begin{pmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{pmatrix}$$

Kantendetektion: Stärkere Kanten, aber mehr Störkanten

## Laplacian of Gauß (LoG)

Der Laplace-Operator ist gegen Rauschen sehr empfindlich. Wesentlich bessere Ergebnisse erhält man, wenn man das Bild zunächst mit einem Gauß-Filter glättet und danach den Laplace-Operator anwendet.

$$LoG(g(x, y)) = \nabla^2(G(x, y) * g(x, y))$$

Approximation (Faltung mit Matrix):

$$\nabla^2 G(x, y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Kantendetektion: Stärkere Kanten, weniger Rauschen

## Canny-Kantendetektor

1. Rauschunterdrückung: Gauß-Filter

2. Kanten detektieren:

- Prewitt oder Sobel
- Zusammenrechnen mit  $B' = \sqrt{B_x^2 + B_y^2}$
- Gradientenrichtung bestimmen:  $\Phi(x, y) = \arctan\left(\frac{B_y(x, y)}{B_x(x, y)}\right)$
- Runden auf  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  oder  $135^\circ$

3. Non-Maximum Supression:

- Für jeden Pixel in Gradientenrichtung schauen, ob das Pixel davor oder dahinter einen höheren Wert hat. Falls ja, dann Pixel auf 0 setzen, falls nein, dann beibehalten.

4. Hysterese-Schwellwertverfahren:

- Verwende zwei Schwellwerte  $T_1$  und  $T_2$  mit  $T_1 \leq T_2$
- Markiere alle Pixel mit Werten größer  $T_2$  als Kantenpixel
- Setze alle Pixel mit Werten kleiner  $T_1$  auf 0
- Beginnend bei jedem Kantenpixel:
  - Verfolge alle angrenzenden Kanten, solange Wert  $\geq T_1$
  - Markiere alle dazugehörigen Pixel als Kanten
- Setze alle noch nicht als Kante markierten Pixel auf 0



## 7.7 2D Bildverarbeitung

### Sensorische Erfassung

Aufgaben der sensorischen Umwelterfassung:

- Wiedererkennung bekannter Sachverhalte: Objekte, Personen, Orte
- Erlernen neuer Sachverhalte
- Erkennung der eigenen Bewegung

Verfahren zur Lösung dieser Aufgaben:

- Sensorische Primitive, Segmentierung
- Annahmen, Einschränkungen
- Lernverfahren

### Segmentierung

Segmentierung ist die Aufteilung eines Bildes in aussagekräftige Bestandteile. Erlaubt:

- Aussagen über das Bild
- Reduktion der Datenmenge
- Verfolgung von Merkmalen über die Zeit / mehrere Sensoren

Beliebt sind: Kanten, Ecken, Textur, Farbe

### Schwellwertfilterung

Schwellwertfilterung zur Konvertierung eines Grauwertbildes in ein binäres Bild. Ziel: Trennung interessanter Objekte vom Hintergrund.

$$Img'(u, v) = \begin{cases} q & \text{falls } Img(u, v) \geq T \\ 0 & \text{sonst} \end{cases}$$

### Farbe

Oft können Objekte über ihre Farbe segmentiert werden:

- menschliche Hautfarbe
- einheitlich gefärbte Objekte

Problem:

- wechselnde Lichtbedingungen
- Reflexionen, Schattenwürfe

Verfahren:

- Histogrammbasiert (z.B. in RGB, HSV bzw. RG, HS)  
HS-Farbhistogramm:
  - Weglassen des  $I$ -Kanals ergibt 2D-Histogramm
  - Training eines Klassifikators auf dem Histogramm

- mit Hilfe der Mahalanobis-Distanz (z.B. in RGB):  
gegeben:  $x_i = (R, G, B)^T$  sind manuell positiv klassifizierte Pixel

$$C = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T \quad \text{Kovarianzmatrix}$$

$$p(x) = e^{-\frac{1}{2\sigma^2} x^T C^{-1} x} \quad \text{Berechnung der Farbwahrsch.}$$

- Klassifikation durch Verwendung von Neuronalen Netzen
- durch Intervallschranken im HSI-Farbraum:

$$f(H, S, V) = H \geq H_{\min} \wedge H \leq H_{\max} \wedge S \geq S_{\min} \wedge S \leq S_{\max} \wedge V \geq V_{\min} \wedge V \leq V_{\max}$$

### Morphologische Operatoren

- Wähle Strukturelement, z.B. Quadrat mit 3x3 Pixel
- Wandere mit Strukturelement über Bild und führe Operation durch
- **Dilatation** ist eine Operation auf Binärbildern. Alle Bereiche, die farbig (1 und nicht 0, da Binär) sind, werden ausgedehnt. Immer wenn unter dem mittleren Pixel des Strukturelements ein farbiges Pixel ist, werden alle Pixel unter dem Strukturelement als farbig markiert.
- **Erosion** ist die Komplementäroperation zu Dilatation. Nur falls alle Pixel unter dem Strukturelement farbig sind, wird das Element in der Mitte farbig gelassen und alle anderen auf 0 gesetzt. Falls nicht alle Pixel farbig sind, werden alle auf 0 gesetzt.
- Öffnen-Operation: Zuerst Erosion und anschließend Dilatation anwenden.
- Schließen-Operation: Zuerst Dilatation und anschließend Erosion anwenden.

### Bewegung

Einfacher Ansatz: Differenzbilder

- Subtraktion aufeinander folgender Bilder einer Video-Sequenz:

$$Img'_t(u, v) = |Img_t(u, v) - Img_{t-1}(u, v)|$$

- Anschließend kann auf  $Img'_t$  Schwellwertfilterung durchgeführt werden
- Regionen, in denen sich etwas bewegt, erscheinen weiß; ruhige Regionen erscheinen schwarz
- Bewegung in homogenen Regionen wird nicht erkannt (Kanten, Textur sind notwendig)
- Richtung der Bewegung wird nicht erkannt

Differenzbilder werden auch für Hintergrundsubtraktion verwendet, dann wird  $Img_{t-1}$  durch ein festes  $Img_0$  ersetzt. Weitere Ansätze: Optical Flow und Erweiterungen

### Region Growing

gegeben: Graustufen-Bild, gesucht: zusammenhängende Regionen

Algorithmus in Pseudocode:

1. wähle Saatpunkt  $p_0 = (u_0, v_0)$
2. initialisiere Region  $R = \{p_0\}$ , wähle Schwelle  $\varepsilon$
3. solange  $\exists p \in R, q \notin R$  mit  $\|p - q\| \leq 1$  und  $|Img(p_0) - Img(q)| \leq \varepsilon$  mache  $R = R \cup \{q\}$

**Kanten**

Vom Menschen konstruierte Umgebungen:

- gut strukturiert
- viele gerade Linien (Wände, Türen, Schränke)
- einfache Segmentierung / 3D-Rekonstruktion
- viele Informationen in einem einzigen Merkmal

Vorgehensweise:

Bildaufnahme  $\Rightarrow$  Filtern, Binarisieren  $\Rightarrow$  Pixel  $\rightarrow$  Kantensegmente

**Pixel  $\rightarrow$  Kanten****Iterative Endpoint Fit:**

gegeben: Punkte  $P$ , Linien  $L = \{\}$ , Distanzschwelle  $d$

- Finde  $x_1, x_2$  aus  $P$  mit  $\|x_1 - x_2\| = \max$ ;  
verbinde sie durch Linie  $l_0 = \{x_1, x_2\}$ ;  $L = L \cup \{l_0\}$
- Entferne  $x_1, x_2$  aus  $P$
- Für alle  $l \in L$ :
  - Finde  $x \in P$  mit  $\|l - x\| = \max$
  - Wenn  $\|l - x\| < d$ :
    - \* Ordne  $x$  als Mitgliedpunkt  $l$  zu
    - \* Entferne  $x$  aus  $P$
  - Sonst
    - \* Brich  $l$  in  $l_1 = \{x_1, x\}$  und  $l_2 = \{x, x_2\}$  auf
    - \* Andere Mitgliedspunkte von  $l$  wieder in  $P$
  - $P$  leer  $\Rightarrow$  Abbruch, sonst weiter
- Lösche Linien mit weniger als  $n$  Punkten

**Hough-Transformation:**

- Ziel: Erkennung gerader Linien im Bild
- Ansatz: Stelle Linie durch Normalenvektor (Länge, Winkel) in Polarkoordinaten dar (Sinus-Kosinus-Kurve)
- Kurven für kollineare Punkte schneiden sich in genau zwei Punkten

$$r = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

- Transformation in den Hough-Raum: Additives Eintragen aller Sinus-Kosinus-Kurven für alle Pixel in ein Histogramm
- Finden der Maxima bzw. Cluster von "Treffern" im Hough-Raum
- Brute-Force-Ansatz; für ein  $n \times n$ -Bild liegt die Laufzeit in  $O(n^3)$

Unterschied zur Regressionsanalyse: Die Bestimmung der Regressionsgerade ist das geeignetere Verfahren, wenn schon klar ist, welche Pixel eine Gerade bilden sollen; dann ist die Regressionsgerade die optimale Gerade im Sinne der Summe der Fehlerquadrate. Die Regressionsgerade kann jedoch keine Segmentierung vornehmen. Dahingegen lassen sich mit der Hough-Transformation in beliebigen Bildern geradlinige Strukturen berechnen; die Punkte dazu müssen jedoch verhältnismäßig exakt auf einer Linie liegen.

### Punktmerkmale

Kanten/Konturen/Farbe können nicht immer für die Segmentierung herangezogen werden. Texturierte Objekte lassen sich in der Regel nicht durch die bislang vorgestellten Verfahren segmentieren. Lösung: Verwendung von *lokalen* Punktmerkmalen (auch genannt: Textmerkmale):

- Harris Corner Detector
- Shi-Tomasi Features
- SIFT-Features
- Maximally Stable Extremal Regions

Punktmerkmal:  $(2n + 1) \times (2n + 1)$ -Pixel-Block um Pixel  $p$ . Fast immer basierend auf Grauwertbildern. Gewünschte Eigenschaft: Wiedererkennbarkeit  
 $\Rightarrow$  hoher Gradient in mehrerer Richtungen

#### Harris Corner Detector:

Sind die Eigenwerte der Matrix

$$A = \begin{pmatrix} \left( \frac{\partial \text{Img}(x,y)}{\partial x} \right)^2 & \frac{\partial \text{Img}(x,y)}{\partial x} \frac{\partial \text{Img}(x,y)}{\partial y} \\ \frac{\partial \text{Img}(x,y)}{\partial x} \frac{\partial \text{Img}(x,y)}{\partial y} & \left( \frac{\partial \text{Img}(x,y)}{\partial y} \right)^2 \end{pmatrix}$$

groß, dann verursacht eine kleine Bewegung in beliebiger Richtung eine große Grauwertänderung. Finden von Ecken durch Suche nach lokalen Maxima in:

$$R = \det(A) - k \cdot \text{trace}(A)^2 \quad , \quad k \approx 0,04$$

Häufiges Problem:

- Wiederfinden bzw. Zuordnung von Punktmerkmalen für:
  - Objekterkennung auf der Basis von Punktmerkmalen
  - Stereo-Sehen bzw. "Structure from Motion" (Korrespondenzproblem)
- Lösung des Korrespondenzproblems erfolgt für Punktmerkmale durch Korrelationsverfahren:
  - **Sum of Squared Differences** (SSD) wird minimal bei guter Übereinstimmung:

$$\sum_{i=-n}^n \sum_{j=-n}^n (\text{Img}_0(x-i, y-j) - \text{Img}_1(x-i, y-j))^2$$

- **Sum of Absolute Differences** (SAD) wird minimal bei guter Übereinstimmung:

$$\sum_{i=-n}^n \sum_{j=-n}^n |\text{Img}_0(x-i, y-j) - \text{Img}_1(x-i, y-j)|$$

- **(Zero Mean) Cross Correlation** wird maximal bei guter Übereinstimmung:

$$\sum_{i=-n}^n \sum_{j=-n}^n (\text{Img}_0(x-i, y-j) - \overline{\text{Img}_0})(\text{Img}_1(x-i, y-j) - \overline{\text{Img}_1})$$

wobei

$\overline{\text{Img}}$  : Durchschnitt

- **Zero Mean Normalized Cross Correlation** wird maximal bei guter Übereinstimmung:

$$\frac{\sum_{i=-n}^n \sum_{j=-n}^n \text{Img}_1(u_1 + i, v_1 + j) - \overline{\text{Img}_1}(u_1, v_1, n)) \cdot (\text{Img}_2(u_2 + i, v_2 + j) - \overline{\text{Img}_2}(u_2, v_2, n))}{\sqrt{\sum_{i=-n}^n \sum_{j=-n}^n (\text{Img}_1(u_1 + i, v_1 + j) - \overline{\text{Img}_1}(u_1, v_1, n))^2 \sum_{i=-n}^n \sum_{j=-n}^n (\text{Img}_2(u_2 + i, v_2 + j) - \overline{\text{Img}_2}(u_2, v_2, n))^2}}$$

wobei

$$\overline{\text{Img}}(u, v, n) = \frac{1}{(2n+1)^2} \sum_{i=-n}^n \sum_{j=-n}^n \text{Img}(u + i, v + j)$$

Für die Objekterkennung geschieht das Wiederfinden oftmals unter Verwendung:

- der Hauptkomponentenanalyse oder engl. *Principal Component Analysis* (PCA) und Nearest-Neighbor- bzw. *k*-Nearest-Neighbor-Klassifikator
- von Neuronalen Netzen
- oder Kombinationen (zuerst PCA zur Kompression, dann SVM (*Support Vector Machine*) zur Klassifikation)

## 7.8 Geometrische 2D-Transformationen

### Translation

Translation eines 2D-Vektors:

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 + x \\ y_0 + y \end{pmatrix}$$

### Rotation

- o.B.d.A. auf Einheitsvektor zurückführbar (Basistransformation)
- Konvention: Rechtsskordinatensystem
- Rotation von  $(x_0, y_0)$  um Winkel  $\beta$  mit Ergebnis  $(x, y)$ :

Aus Additionstheorem:

$$\begin{aligned} x &= \cos(\alpha + \beta) = \cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta) \\ y &= \sin(\alpha + \beta) = \sin(\beta) \cos(\alpha) + \cos(\beta) \sin(\alpha) \end{aligned}$$

und mit  $(x_0, y_0) = (\cos(\alpha), \sin(\alpha))$ :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

### Homogene Koordinaten

Homogene Koordinaten

$$h = (h_0, h_1, \dots, h_i, h_{i+1})$$

eines Punktes  $p$  im  $R^i$  mit

$$p = (p_0, \dots, p_i)$$

sind Zahlen, für die gilt:

$$p_k = \frac{h_k}{h_{i+1}} \quad \forall 0 \leq k \leq i$$

### Homogene 2D-Transformationen

Transformationen definiert durch Rotation  $R$  und Translation  $t$ .

$$\begin{pmatrix} x \\ y \end{pmatrix} = R \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + t = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Darstellung mit Hilfe homogener Koordinaten und einer geschlossenen Transformationsmatrix:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \left( \begin{array}{cc|c} R & t \\ \hline 0 & 0 & 1 \end{array} \right) \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \left( \begin{array}{cc|c} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ \hline 0 & 0 & 1 \end{array} \right) = A \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

### Partikel Filter und 2D-Tracking

Für Tracking-Applikationen wird oftmals das Kalmanfilter, das Partikel Filter oder Kombinationen aus beiden verwendet. Vorteile des Partikel Filters gegenüber dem Kalmanfilter:

- Es kann automatisch mehrere, parallel existierende Hypothesen halten  $\Rightarrow$  geringere Gefahr in lokalen Minima zu verharren.
- Es kann beliebige Wahrscheinlichkeitsdichten modellieren und dadurch nichtlineare Bewegungen auf natürliche Weise verfolgen.

Nachteil: Rechenaufwand ist proportional zur Anzahl der notwendigen Partikel, welche (bei konstanter Auflösung) mit der Dimension des Konfigurationsraums exponentiell steigt.

Im Kern des Partikel Filters befinden sich ein Modell mit Konfigurationsraum  $R^n$ .

*Eingabe:* Beobachtungen  $z$ ; hier: Bilder einer Videosequenz

*Ausgabe:* Schätzung der Konfiguration  $s \in R^n$ , die den aktuellen Beobachtungen  $z$  entspricht

*Zentrale Funktion:* Bewertungsfunktion  $P(z|s)$ , die die a-posteriori Wahrscheinlichkeit berechnet, dass  $s$  die zu  $z$  passende Konfiguration ist.

Das Partikel Filter modelliert die Wahrscheinlichkeitsdichtefunktion oder engl. *probability density function* (*pdf*) durch eine feste Anzahl von  $N$  Partikeln. Die Wahrscheinlichkeitsdichtefunktion beschreibt die a-posteriori Wahrscheinlichkeiten für den Konfigurationsraum. Jedes Partikel ist ein Paar  $(s_i, \pi_i)$ , wobei  $s_i$  eine Konfiguration ist und  $\pi_i$  die dazugehörige a-posteriori Wahrscheinlichkeit mit  $\sum_{i=1}^n \pi_i = 1$ . Die aktuelle Schätzung des Partikel Filters erfolgt über das gewichtete Mittel über alle Partikel:

$$\bar{s} = \sum_{i=1}^N \pi_i \cdot s_i$$

### Algorithmus

1. Initialisiere alle  $N$  Partikel (z.B. mit Gleichverteilung)
2. Verarbeite neue Beobachtungen (z.B. Vorverarbeitung neuer Bilder)
3. Ziehe  $N$  Partikel aus der letzten Generation, proportional zu ihrer Wahrscheinlichkeit  $\pi_i$ , und für jedes dieser Partikel:
  - Berechne neue Konfiguration auf Basis der alten Konfiguration durch Addition normalverteilten Rauschens und evtl. durch Hinzunahme eines dynamischen Modells.
  - Berechne für diese neue Konfiguration die neue a-posteriori Wahrscheinlichkeit mit Hilfe der Bewertungsfunktion  $P(z|s)$
4. Berechne aktuelle Schätzung  $\bar{s}$  über alle Partikel
5. Fahre fort mit Schritt 2

### Einfaches Beispiel für eine Bewertungsfunktion

- Anwendung:  
2D-Tracking einer dichten, in etwa quadratischen Fläche von in etwa fester Größe in einem binarisierten Bild
- Modell:  
Quadrat fester Größe mit Kantenlänge  $k$  mit Konfigurationsraum  $R^2$  (Koordinaten  $u, v$  im Bild)
- Bewertungsfunktion:

$$P(z | s) \propto e^{-\frac{1}{2\sigma^2}(k^2 - \sum_{m \in M} g_m)}$$

wobei  $M$  die Menge aller Pixel im binarisierten Bild  $z$  (Beobachtungen) im durch  $s$  (Konfiguration) definierten Quadrat beschreibt und  $g_m$  deren Intensität aus  $\{0, 1\}$ .

## 7.9 Geometrische 3D-Transformationen

Grundlage von Sensorik und Aktorik: Beschreibung von Objektposen (Position, Rotation)

- Pose des messenden Systems im Raum
- Pose mehrerer Sensoren zueinander
- Pose sensorisch erfasster Objekte relativ zum Sensor
- Pose von Aktoren (Greifer, Lötlampen etc.) im Raum und relativ zum manipulierten Objekt
- Gelenkwinkelstellungen im Roboterarm

Anforderungen:

- geschlossene Ausdrücke
- Invertierbarkeit
- Interpolation

Zwei Systeme haben sich durchgesetzt:

- Homogene Geometrie (für Translationen und Rotationen)
- Quaternionendarstellung (nur für Rotationen)

### Translation

Translation eines 3D-Vektors:

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 + x \\ y_0 + y \\ z_0 + z \end{pmatrix}$$

### Rotation

- o.B.d.A. auf Einheitsvektor zurückführbar (Basistransformation)
- Konvention: Rechtshandkoordinatensystem
- Rotation von  $(x_0, y_0, z_0)$  um Winkel  $\beta$  mit Ergebnis  $(x, y, z_0)$

Aus Additionstheorem:

$$\begin{aligned} x &= \cos(\alpha + \beta) = \cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta) \\ y &= \sin(\alpha + \beta) = \sin(\beta) \cos(\alpha) + \cos(\beta) \sin(\alpha) \end{aligned}$$

und mit  $(x_0, y_0) = (\cos(\alpha), \sin(\alpha))$ :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

$z_0$  invariant, da Rotation um  $z$ !

### Rotationsmatrix

Rotationsmatrix allgemein:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

Rotation um  $x$ :

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Rotation um  $y$ :

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

Rotation um  $z$ :

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

### Eigenschaften von Rotationsmatrizen:

- regulär, invertierbar, Determinante = 1
- jede beliebige Rotation im Raum kann durch drei Variablen beschrieben werden (Eulers Theorem)
- Einzelrotationen können als eine Matrix dargestellt werden:

$$R_r(\gamma)R_q(\beta)R_p(\alpha) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{mit} \quad p, q, r \in \{x, y, z\} = R_{pqr}(\alpha, \beta, \gamma) \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- damit reicht Angabe von  $\alpha, \beta, \gamma$  zur Beschreibung der Rotation
- das macht natürlich nur Sinn, wenn eine Konvention für die Zuordnung  $p, q, r$  zu den Achsen  $x, y, z$  definiert wurde

Zwei grundlegend unterschiedliche Rotationstypen:

- Rotation um mitgedrehte Achsen (Euler-Winkel)
- Rotation um raumfeste Achsen (Roll Pitch Yaw)

Konventionen zur Erstellung von Rotationsmatrizen:

- Standard-Beispiel für Euler-Winkel:  
Zuerst um die  $x$ -Achse, dann um die mitgedrehte  $y$ -Achse, dann um die (zweimal) mitgedrehte  $z$ -Achse

$$R_{X'Y'Z'}(\alpha, \beta, \gamma) = R_X(\alpha)R_Y(\beta)R_Z(\gamma)$$

- Standard-Beispiel für raumfeste Achsen:

$$R_{XYZ} = R_Z(\alpha)R_Y(\beta)R_X(\gamma)$$



**Homogene 3D-Transformation**

Transformation definiert durch Rotation  $R$  und Translation  $t$ :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + t = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Darstellung mit Hilfe homogener Koordinaten und einer geschlossenen Transformationsmatrix:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \left( \begin{array}{ccc|c} & & & \\ & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix} = \left( \begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix} = A \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix}$$

**Probleme mit Rotationsmatrizen:**

- hoch redundant
- rechenaufwendig
- Interpolation schwierig
- numerisch unrobust (Singularitäten, Rundungsfehler)

**Quaternionen**

Erweiterung der komplexen Zahlen ins vierdimensionale. Definition:

Ein Quaternion  $\mathbf{q}$  ist eine Zahl

$$\begin{aligned} \mathbf{q} &= (q_x, q_y, q_z, q_w)(i, j, k, 1)^T \\ &= (\mathbf{q}_v, q_w)(i, j, k, 1)^T \\ &= iq_x + jq_y + kq_z + q_w \end{aligned}$$

mit

$$\begin{aligned} i^2 &= j^2 = k^2 = -1 \\ ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned}$$

$q_w$  ist der Realteil,  $\mathbf{q}_v = (q_x, q_y, q_z)$  der Imaginärteil des Quaternions. Man schreibt einfach  $(q_x, q_y, q_z, q_w)$  oder  $(\mathbf{q}_v, q_w)$ .

Rechenregeln für Quaternionen:

Konjugation:

$$\bar{\mathbf{q}} = (-\mathbf{q}_v, q_w)$$

Norm:

$$N(\mathbf{q}) = \sqrt{\mathbf{q}\bar{\mathbf{q}}} = \sqrt{\bar{\mathbf{q}}\mathbf{q}} = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}$$

Quaternionen  $\mathbf{q}$  mit  $N(\mathbf{q}) = 1$  heißen Einheitsquaternionen.

Multiplikative Identität:

$$I = (0, 1)$$

Multiplikative Inverse:

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{N^2(\mathbf{q})}$$

Rotation mit Quaternionen:

- Einheitsquaternion  $\mathbf{q}$  ist definiert durch Rotationsachse  $u$  mit  $|u| = 1$  und Winkel  $\theta$ :

$$\mathbf{q} = \left( \cos \frac{\theta}{2}, u \sin \frac{\theta}{2} \right)$$

- Quaternion  $\mathbf{a}$  ist definiert durch zu rotierenden Vektor  $v$ :

$$\mathbf{a} = (v, 0)$$

- Das Produkt  $\mathbf{qa}\bar{\mathbf{q}}$  rotiert  $v$  um die Achse  $u$  mit dem Winkel  $\theta$ .

Interpolation zwischen zwei Quaternionen:

- Sphärische Lineare Interpolation (SLERP)
- Berechnet für  $t \in [0, 1]$  die kürzeste Verbindung auf der vierdimensionalen Einheitssphäre zwischen  $q$  und  $r$ .
- Analytisch:

$$SLERP(q, r, t) = q(rq^{-1})^t$$

- Numerisch:

$$SLERP(q, r, t) = q \frac{\sin((1-t)\theta)}{\sin(\theta)} + r \frac{\sin(t\theta)}{\sin(\theta)}$$

mit Winkel  $\theta$  zwischen  $r$  und  $q$ .

Quaternion  $\rightarrow$  Rotationsmatrix:

$$\mathbf{q} = (q_x, q_y, q_z, q_w) \Rightarrow M_q = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{pmatrix}$$

Rotationsmatrix  $\rightarrow$  Quaternion:

$$\begin{aligned} q_w &= \frac{1}{2} \sqrt{1 + \sum_{i=1}^3 m_{ii}} \\ q_x &= \frac{(m_{32} - m_{23})}{4q_w} \\ q_y &= \frac{(m_{13} - m_{31})}{4q_w} \\ q_z &= \frac{(m_{21} - m_{12})}{4q_w} \end{aligned}$$

### **Vor- und Nachteile**

Vorteile:

- Rotation direkt um gewünschte Drehachse
- Interpolation möglich
- weniger Rechenaufwand
- keine Redundanz  $\Rightarrow$  numerisch stabiler, weniger Gefahr für Singularitäten

Nachteil:

- nur Rotation berechenbar  $\Rightarrow$  Kombination mit Matrizen nötig  $\Rightarrow$  Rechenaufwand für Umwandlungen

### Realistischeres Kameramodell

Lochkameramodell vereinfacht die realen Verhältnisse stark. Deshalb werden in der Praxis Erweiterungen des Lochkameramodells verwendet. Zunächst einige Definitionen:

**Optische Achse:** Gerade durch das Projektionszentrum, senkrecht zur Bildebene

**Bildhauptpunkt**  $C(c_x, c_y)$ : Schnittpunkt der optischen Achse mit der Bildebene

Koordinatensysteme:

**Bildkoordinatensystem:** 2D-Koordinatensystem, Einheit [Pixel], Vereinbarung für die Vorlesung (gilt für die meisten Kamerasreiber): Ursprung in der linken oberen Ecke des Bildes,  $u$ -Achse zeigt nach rechts,  $v$ -Achse zeigt nach unten.

**Kamerakoordinatensystem:** 3D-Koordinatensystem, Einheit [mm], Ursprung liegt im Projektionszentrum, Achsen parallel zu den Achsen des Bildkoordinatensystems, d.h.  $x$ -Achse nach rechts,  $y$ -Achse nach unten und die  $z$ -Achse gemäß der Dreifingerregel für ein rechtshändiges Koordinatensystem nach vorne.

**Weltkoordinatensystem:** 3D-Koordinatensystem, Einheit [mm], Basiskoordinatensystem, das beliebig im Raum liegen kann.

Begriffe:

**Intrinsische Kameraparameter:** Brennweite, Bildhauptpunkt, Parameter für die Beschreibung radialer/tangentialer Linsenverzerrung; definieren die nicht (eindeutig) umkehrbare Abbildung vom Kamerakoordinatensystem in das Bildkoordinatensystem.

**Extrinsische Kameraparameter:** Definieren die Transformation vom Kamerakoordinatensystem in das Weltkoordinatensystem, im Allgemeinen durch eine Rotation  $R$  und eine Translation  $t$ .

Vereinfachungen des Lochkameramodells:

- Ursprung des Bildkoordinatensystems ist identisch mit dem Bildhauptpunkt
- Pixel werden als quadratisch angenommen
- keinerlei Modellierung der Linsenverzerrung
- es existiert kein Weltkoordinatensystem bzw. es ist identisch mit dem Kamerakoordinatensystem, d.h. es werden keine extrinsischen Kameraparameter modelliert

Brennweite:

- In der Praxis wird die Umrechnung von [mm] nach [Pixel] in den/die Parameter für Brennweite mit aufgenommen.
- Da Pixel nicht mehr als quadratisch sondern als rechteckig angenommen werden, gibt es deshalb für jede Richtung einen Parameter, also:  $f_x, f_y$ .
- Die Parameter  $f_x, f_y$  sind dann das Produkt aus der tatsächlichen Brennweite mit Einheit [mm] und dem jeweiligen Umrechnungsfaktor mit Einheit [Pixel/mm].
- Die Einheit für die Parameter  $f_x, f_y$  ist somit [Pixel].

Die Abbildung vom Kamerakoordinatensystem in das Bildkoordinatensystem, ausschließlich mit den intrinsischen Parametern, ist dann definiert durch:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{Z} \cdot \begin{pmatrix} f_x \cdot X \\ f_y \cdot Y \end{pmatrix}$$

oder als Matrixmultiplikation mit Kalibriermatrix  $K$  auf homogenen Koordinaten:

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = K \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Extrinsische Kamerakalibrierung:

- Ist definiert durch eine Koordinatentransformation bestehend aus Rotation und Translation.
- Koordinatentransformation vom Weltkoordinatensystem in das Kamerakoordinatensystem:

$$x_c = R x_w + t$$

Koordinatentransformation vom Kamerakoordinatensystem in das Weltkoordinatensystem:

$$x_w = R^T x_c - R^T t$$

- $3 \times 4$  Gesamt-Projektionsmatrix  $P$  (intrinsisch und extrinsisch) auf homogenen Koordinaten:

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = P \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad P = (K \ R \ | \ K \ t)$$

### Kamerakalibrierung

Die Kalibrierung einer Kamera bedeutet die Bestimmung ihrer Parameter bezüglich eines gewählten Kameramodells. Die Bestimmung der intrinsischen Parameter ist unabhängig vom Aufbau; solange Zoom und Fokus der Kamera gleich bleiben, verändern sich diese Parameter nicht. Die Bestimmung der extrinsischen Parameter ist abhängig von der Wahl des Weltkoordinatensystems und ändert sich je nach Aufbau.

Ist die Kamera kalibriert, dann liegt die Abbildungsfunktion  $f$  vor, die einen Punkt vom Weltkoordinatensystem eindeutig in das Bildkoordinatensystem abbildet:

$$f : R^3 \rightarrow R^2$$

$f$  ist definiert durch die Projektionsmatrix  $P$  und anschließender Transformation der homogenen Koordinaten durch Division durch  $w$ . Die Inverse Abbildung bildet einen Punkt im Bildkoordinatensystem auf eine Gerade im Weltkoordinatensystem ab, die durch das Projektionszentrum verläuft.

Verfahren zur Kamerakalibrierung:

- Direkte Lineare Transformation (DLT)
- Erweiterungen der DLT, welche Linsenverzerrung modellieren

gesucht:  $3 \times 4$ -Matrix, hat also 12 Unbekannte; Verfahren Testfeldkalibrierung:

- Bestimmung einer Menge von Punktkorrespondenzen: 3D-Punkt in einem gewählten Weltkoordinatensystem und 2D-Punkt im Bildkoordinatensystem
- 3D-Punkte sind durch Verwendung eines geeigneten Kalibrierobjekts oder -musters a-priori bekannt
- 2D-Punkte werden durch Methoden der Bildverarbeitung berechnet

benötigt: 6 bekannte Objektpunkte, da jede Punktkorrespondenz zwei Gleichungen liefert

Bedingung: 3D-Punkte dürfen nicht koplanar liegen, d.h. sie müssen einen dreidimensionalen Raum aufspannen

Möglichkeiten:

- Verwendung eines 2D-Musters, das in mindestens zwei verschiedenen Tiefen präsentiert wird.
- Verwendung eines geeigneten 3D-Kalibrierobjekts.

### Direkte Lineare Transformation

Ein Standard-Verfahren für die Berechnung der Projektionsmatrix  $P$  ist die Direkte Lineare Transformation (DLT)

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = P \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad P = (K \ R | \ K \ t) = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{pmatrix}$$

$$\begin{aligned} \Rightarrow u &= \frac{p_1 X + p_2 Y + p_3 Z + p_4}{p_9 X + p_{10} Y + p_{11} Z + p_{12}} \\ v &= \frac{p_5 X + p_6 Y + p_7 Z + p_8}{p_9 X + p_{10} Y + p_{11} Z + p_{12}} \end{aligned}$$

o.B.d.A. kann ein Parameter normiert werden. Üblicherweise wird  $p_{12} = 1$  gewählt.

$$\begin{aligned} p_1 X + p_2 Y + p_3 Z + p_4 &= u p_9 X + u p_{10} Y + u p_{11} Z + u \\ p_5 X + p_6 Y + p_7 Z + p_8 &= v p_9 X + v p_{10} Y + v p_{11} Z + v \end{aligned}$$

Formuliert als überbestimmtes LGS  $Ax = b$  mit  $n \geq 6$  Punktkorrespondenzen, das beispielsweise mit Hilfe der Normalengleichung gelöst werden kann:

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n \end{pmatrix} \quad x = \begin{pmatrix} p_1 \\ \vdots \\ p_{11} \end{pmatrix} \quad b = \begin{pmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{pmatrix}$$

## 7.10 3D Bildverarbeitung

### Stereokonstruktion

gegeben:

- zwei Kameras (durch ihre Zentren  $C$  und  $C'$ ) mit Projektionsmatrizen  $P$  und  $P'$
- zwei Abbilder  $x$  und  $x'$  des Punktes  $X$
- dann kann  $X$  rekonstruiert werden

Triangulation zwischen linker und rechter Kamera möglich durch Kenntnis der Kameraparameter. Eine Möglichkeit zur Berechnung von 3D-Punkten aus Bildpunkt-Korrespondenzen  $x, x'$ :

- Aufstellen der beiden Geraden  $g, g'$  der möglichen Punkte zu  $x, x'$  im Weltkoordinatensystem mit Hilfe der Projektionsmatrizen  $P, P'$ :

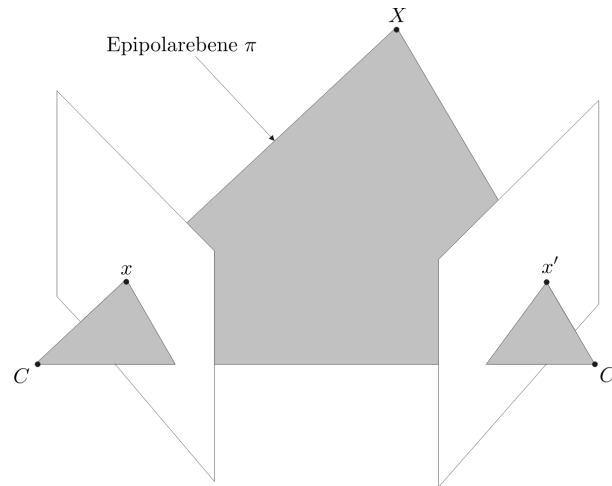
$$\begin{aligned} g : x &= a + r \cdot u \\ g' : x &= b + s \cdot v \end{aligned}$$

- Berechnung des optimalen "Schnittpunktes"  $S$  durch Lösung des überbestimmten LGS  $Ax = c$  mit:

$$A = \begin{pmatrix} u_1 & -v_1 \\ u_2 & -v_2 \\ u_3 & -v_3 \end{pmatrix}, \quad x = \begin{pmatrix} r' \\ s' \end{pmatrix}, \quad c = b - a \quad s = \frac{a + r' \cdot u + b + s' \cdot v}{2}$$

3D-Tracking von Kopf und Händen auf der Basis von Hautfarbe, Region-Growing und Stereokalibrierung.

## Epipolargeometrie



Zusammenhang zwischen zwei Kameras ist gegeben durch die Epipolargeometrie. Die Schnittpunkte  $e$  und  $e'$  der Geraden durch die Projektionszentren mit den Bildebenen nennt man Epipole.

**Epipolarebene  $\pi(X)$ :** Ebene, die durch  $C$ ,  $C'$  und Szenenpunkt  $X$  aufgespannt wird.

**Epipolarlinie  $l'(x)$ :** Schnittgerade von  $\pi(X)$  mit Bildebene.

Alle Punkte  $X$ , die auf  $x$  in Kamera 1 abgebildet werden, werden auf einen Punkt der Linie  $l'(x)$  in Kamera 2 abgebildet. Alle Epipolarlinien eines Kamerasystems schneiden sich in den Epipolen  $e$  und  $e'$ .

*Nutzen:* Einschränkung des Korrespondenzproblems von zwei Dimensionen auf eine Dimension, da nach entsprechenden Merkmalen nur noch entlang der Epipolarlinie gesucht werden muss:

- höhere Robustheit (weniger falsche Korrespondenzen)
- höhere Effizienz

## Fundamentalmatrix

Mathematische Beschreibung der Epipolargeometrie erfolgt durch die Fundamentalmatrix. Eigenschaften der Fundamentalmatrix  $F$ :

- $3 \times 3$ -Matrix
- Rang 2
- für alle Korrespondenzen  $x$ ,  $x'$  gilt:

$$x'^T F x = 0$$

$x$  und  $x'$  sind Bildpunkte in homogenen Koordinaten mit  $w = 1$

Mit der Fundamentalmatrix lassen sich die Epipolarlinien berechnen:

$$l = F^T x' \quad \text{und} \quad l' = F x$$

Für die Epipole gilt:

$$F e = 0 \quad \text{und} \quad F^T e' = 0$$

Hinweis:  $l$  (bzw.  $l'$ ) definieren eine 2D-Gerade wie folgt:

$l x = 0$  für alle Bildpunkte  $x$  (in homogenen Koordinaten mit  $w = 1$ ), die auf dieser Geraden liegen. Die Fundamentalmatrix lässt sich auf mehrere Arten berechnen:

- über Bildpunkt-Korrespondenzen in der linken und rechten Kamera

- bei bekannter intrinsischer und extrinsischer Kalibrierung der Kameras direkt über die Kalibriermatrizen  $K$ ,  $K'$  und der Essentialmatrix  $E$ , die durch die extrinsischen Parameter definiert ist

**Berechnung über Bildpunkt-Korrespondenzen:**

$$x'^T F x = 0 \quad , \quad x' = (x', y', z') \quad , \quad x = (x, y, z)$$

$$\Rightarrow \begin{aligned} & x' x f_{11} + x' y f_{12} + x' z f_{13} \\ & + y' x f_{21} + y' y f_{22} + y' z f_{23} \\ & + x f_{31} + y f_{32} + z f_{33} = 0 \end{aligned}$$

Für  $n \geq 7$  Korrespondenzen  $x, x'$ :

$$\underbrace{\begin{pmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{pmatrix}}_A \underbrace{\begin{pmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{33} \end{pmatrix}}_f = 0$$

$Af = 0$  lösen z.B. mit Singulärwertzerlegung (SVD)

**Berechnung über Essentialmatrix:**

Essentialmatrix lässt sich durch die extrinsischen Parameter berechnen:  
gegeben:

- Kamera 1 mit  $(I|0)$  als Transformation (Identität)
- Kamera 2 mit  $(R|t)$  als Transformation

Essentialmatrix  $E$  lässt sich berechnen zu:

$$E = [t]_x R = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}$$

Für die Epipole gilt:

$$e = KR^T t \quad \text{und} \quad e' = K' t$$

Hat man die Essentialmatrix (z.B. über die extrinsischen Parameter) berechnet und die intrinsischen Parameter, d.h. Kalibriermatrizen  $K, K'$ , so lässt sich die Fundamentalmatrix berechnen zu:

$$F = K'^{-T} E K^{-1}$$

Hat man umgekehrt die Fundamentalmatrix bestimmt (z.B. über Bildpunkt-Korrespondenzen) und die intrinsischen Parameter, d.h. die Kalibriermatrizen  $K, K'$ , so lässt sich die Essentialmatrix berechnen zu:

$$E = K'^T F K$$

**Stereo-Sehen**

Weitere Eigenschaften der Fundamentalmatrix:

- Mit ihr lassen sich die Eingabebilder rektifizieren.
  - Nach Rektifizierung verlaufen alle Epipolarlinien horizontal mit derselben  $v$ -Koordinate wie der Bildpunkt im anderen Kamerabild.
  - Nach Korrespondenzen muss nur noch horizontal (in eine Richtung) gesucht werden.
- Mit Hilfe der Essentialmatrix lassen sich die Projektionsmatrizen bis auf Skalierung genau rekonstruieren, mit Hilfe der Fundamentalmatrix bis auf Skalierung und Projektion genau.

Rektifizierte Bilder haben den Vorteil, dass sich optimierte Korrelations-Algorithmen für die Lösung des Korrespondenzproblems verwenden lassen.  $\Rightarrow$  Laufzeit unabhängig von der Fenstergröße  
Nachteile:

- Interpolation notwendig für die Berechnung der rektifizierten Bilder  $\Rightarrow$  Qualitätsverlust
- Bilder je nach Aufbau stark verzerrt

Nach Lösung des Korrespondenzproblems können

- Punktwolken berechnet werden durch Triangulation, wie zuvor erläutert
- Tiefenbilder erzeugt werden durch Eintrag der Disparitäten (Differenz der  $u$ -Koordinaten für gefundene Korrespondenzen in den rektifizierten Bildern) in ein Graustufenbild:  
 $\Rightarrow$  Je höher der Grauwert, desto näher befindet sich der entsprechende 3D-Punkt zur Kamera



## 8 Wissensrepräsentation

### 8.1 Einführung

#### Was ist Wissen?

Gespeicherte

- Beschreibungen, Modelle, Aktionsfolgen
- Vorschriften zur Reaktion auf Ereignisse
- Motorische, kognitive Fähigkeiten

konkreter:

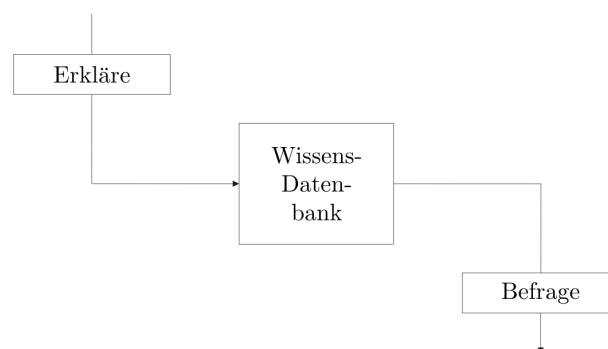
- Computerprogramme
- Regeln der Aussagenlogik
- Gewichte von Neuronalen Netzen, ...

### 8.2 Grundlagen



Eine Wissensdatenbank besteht aus Sätzen, die in einer Wissensrepräsentationssprache abgefasst sind. Jeder Satz stellt eine Annahme über die Welt dar:

$\alpha$ : "Ident1 IST-EIN Fahrrad."  
 $\beta$ : "Ident2 IST-EIN Mensch."  
 $\gamma$ : "Ident3 BESITZT Ident1." oder  
 $\alpha$ : " $r \in R^{20}$ ."

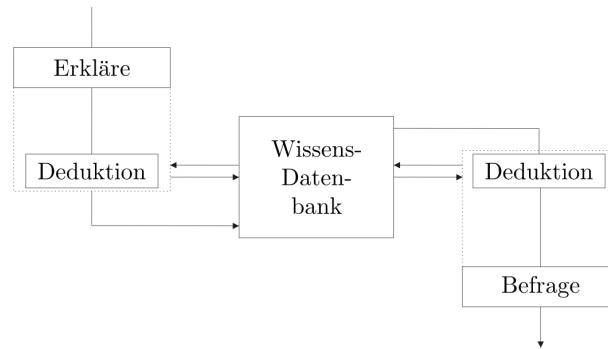


**Erkläre:** Füge der Datenbank Wissen hinzu.

**Befrage:** Frage Wissen aus Datenbank ab.

```

Erkläre(DB, "Ident1 BESITZT Ident2.")
      → OK
Befrage(DB, "Was BESITZT Ident2?")
      → Ident1
  
```



**Deduktion:** Leite neue Sätze von bekannten ab. Injektive Abbildung Deduktion:

$$DB \rightarrow DB$$

Minimale Deduktion: Identität.

### 8.3 Logik allgemein

Logik : (Symbolmenge, Modellmenge, Syntax, Semantik, Folgerungsop.  $\models$ )

#### Symbol-, Modellmenge

- Elementarwerte für Aussagen: wahr, falsch
- Symbolmenge  $S$  enthält alle Symbole, über die Aussagen gemacht werden können.
- Modell: Menge  $\{(s_i, w_i)\}$  von Symbolen  $s_i \in S$  mit zugeordneten Werten,  $w_i \in Def(s_i)$
- Modellmenge: Menge aller möglichen Modelle:

$$\forall i \forall w \in Def(s_i) \exists M : (s_i, w_i) \in M = (s_i, w)$$

#### Beispiel: Aussagenlogik

$$\begin{aligned} S &= \{a, b\} \\ M &= \{ \{(a, wahr), (b, wahr)\}, \\ &\quad \{(a, wahr), (b, falsch)\}, \\ &\quad \{(a, falsch), (b, wahr)\}, \\ &\quad \{(a, falsch), (b, falsch)\} \} \end{aligned}$$

Schreibweise: lateinische Buchstaben für Symbole, griechische Buchstaben für Sätze

#### Syntax

**Syntax:** Legt fest, welche Sätze wohlgeformt sind.

Nur wohlgeformte Sätze sind gültig!

$(a \vee b) \Rightarrow c$	wohlgeformt
$()a \vee \Rightarrow bc$	nicht wohlgeformt

### Semantik

**Semantik:** Bestimmt den Wahrheitswert eines Satzes in Bezug auf ein Modell.

$$\begin{aligned}
 \text{Satz } \alpha &= (a \wedge b) \Rightarrow c \\
 \text{Modell } M &= \{(a, \text{wahr}), (b, \text{wahr}), (c, \text{wahr})\} \\
 &\Rightarrow \alpha/M \text{ wahr, } M \text{ erfüllt } \alpha \\
 \text{Modell } M &= \{(a, \text{wahr}), (b, \text{wahr}), (c, \text{falsch})\} \\
 &\Rightarrow \alpha/M \text{ falsch}
 \end{aligned}$$

**Folgerung**  $\models$

$$\alpha \models \beta : \text{"}\beta \text{ folgt aus } \alpha\text{"}$$

$\alpha \models \beta$  genau dann, wenn für alle Modelle, in denen  $\alpha$  wahr ist,  $\beta$  ebenfalls wahr ist.

$$\alpha : x + y = 4 \quad \beta_1 : y = 4 - x \quad \beta_2 : \frac{x}{y} = 1 \quad \Rightarrow \quad \alpha \models \beta_1 \quad \alpha \not\models \beta_2$$

Zurück zur Wissensdatenbank  $WB$ :

Kann als  $\wedge$ -verknüpfte Sequenz von Sätzen aufgefasst werden:

$$WB : \left\{ \begin{array}{l} \alpha_1 : x + y = 4 \\ \alpha_2 : x/y = 1 \end{array} \right\}$$

$$\Rightarrow WB = \alpha_1 \wedge \alpha_2 \text{ wahr } \forall M : x = 2, y = 2$$

$$\Rightarrow M \text{ erfüllt } WB \forall M : x = 2, y = 2$$

$$WB \models \beta \quad \text{gdw} \quad \left( \bigwedge_{\alpha_i \in WB} \alpha_i \right) \Rightarrow \beta = \text{wahr}$$

### Modellprüfung

Algorithmus zur Überprüfung einer  $WB \models \alpha$ -Relation:

1. Finde Menge aller Modelle  $M = \{M_i\}$  über  $S$
2. Eliminiere alle  $M_i$  mit  $WB/M_i = \text{falsch}$
3.  $WB \models \alpha$  genau dann, wenn
 
$$\forall M_i \in M : \alpha/M_i = \text{wahr}$$

### Deduktion

Wenn ein Algorithmus  $i$  existiert, der den Satz  $\alpha$  aus der Wissensbasis  $WB$  ableiten kann, schreiben wir

$$WB \vdash_i \alpha$$

(" $\alpha$  wird durch  $i$  aus  $WB$  abgeleitet" oder " $i$  leitet  $\alpha$  aus  $WB$  ab")

Eigenschaften von Deduktionsalgorithmen:

- Algorithmus  $i$  "korrekt" genau dann, wenn er nur Sätze aus  $WB$  ableitet, die aus  $WB$  folgen:

$$\forall \alpha : WB \vdash_i \alpha \quad \Rightarrow \quad WB \models \alpha$$

- Algorithmus  $i$  "vollständig" genau dann, wenn er alle Sätze aus  $WB$  ableitet, die aus  $WB$  folgen:

$$\forall \alpha : WB \models \alpha \quad \Rightarrow \quad WB \vdash_i \alpha$$

### Zusammenfassung

Eine Logik:

- bestimmt den Wahrheitsgehalt von Sätzen in Bezug auf Modelle
- besteht aus Symbolmenge, Modellmenge, Syntax, Semantik, Folgerung

Eine Wissensbasis:

- besteht aus einer Menge von Sätzen
- passt zu einem Modell  $M$  (oder auch nicht)

Ein Deduktions-Algorithmus

- leitet Sätze aus einer Wissensbasis ab
- kann korrekt und / oder vollständig sein

## 8.4 Aussagenlogik

- Aussagenlogik als Beispiel einer sehr einfachen Logik
- sehr alt (antikes Griechenland)
- bildet die Basis, von der die allgemeine Logik abgeleitet wurde
- eignet sich sehr "natürlich" zur Illustration bestimmter Konzepte

### Syntax

$$\begin{aligned}
 \text{Satz} &\rightarrow \text{Atom} \mid \text{Komplex} \\
 \text{Atom} &\rightarrow \text{True} \mid \text{False} \mid \text{Symbol} \\
 \text{Symbol} &\rightarrow P \mid Q \mid R \dots \\
 \text{Komplex} &\rightarrow \neg \text{Satz} \mid (\text{Satz} \wedge \text{Satz}) \mid (\text{Satz} \vee \text{Satz}) \mid (\text{Satz} \Rightarrow \text{Satz}) \mid (\text{Satz} \Leftrightarrow \text{Satz})
 \end{aligned}$$

### Semantik

Wahrheitsgehalt im Modell  $M$ :

1.  $\text{True} = \text{wahr}$ ,  $\text{False} = \text{falsch} \forall M$
  2. Wahrheitsgehalt von Symbolen wird in  $M$  spezifiziert
  3. Wahrheitsgehalt aller anderen Sätze rekursiv:
    - (a)  $\forall$  Sätze  $\alpha \forall$  Modelle  $M$  :  $\neg \alpha$  falsch gdw  $\alpha \in M$  wahr
    - (b)  $\forall \alpha, \beta \forall M$  :  $\alpha \wedge \beta$  wahr gdw  $\alpha \in M$  wahr und  $\beta \in M$  wahr
    - (c) etc.
- (Alternative: Wahrheitstabelle)

**Muster**

Allgemeingültige Deduktionsregeln aus der klassischen Literatur:

- **Modus Ponens:**

Wenn  $\alpha \Rightarrow \beta$  und  $\alpha$  gegeben sind, kann  $\beta$  inferiert werden.

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- **Und-Elimination:**

$$\frac{\alpha \wedge \beta}{\alpha} \quad \frac{\alpha \wedge \beta}{\beta}$$

Aus einer Konjunktion kann jedes der Elemente inferiert werden.  
Korrektheitsbeweis aus Wahrheitstabelle!

**8.5 Resolution**

Basis für Deduktionsalgorithmen

**Einheits-Resolutionsregel** (aus Modus Ponens):

Seien  $p_1, \dots, p_k$  und  $q$  Literale mit  $q = \neg p_i$ . Dann gilt:

$$\frac{p_1 \vee \dots \vee p_k, q}{p_1 \vee \dots \vee p_{i-1} \vee p_{i+1} \vee \dots \vee p_k}$$

**Allgemeine Form** der Resolutionsregel:

$p_1, \dots, p_k$  und  $q_1, \dots, q_n$  Literale mit  $p_i = \neg q_j$ . Dann gilt:

$$\frac{p_1 \vee \dots \vee p_k, q_1 \vee \dots \vee q_n}{p_1 \vee \dots \vee p_{i-1} \vee p_{i+1} \vee \dots \vee p_k \vee q_1 \vee \dots \vee q_{j-1} \vee q_{j+1} \vee \dots \vee q_n}$$

Jeder vollständige Suchalgorithmus, kombiniert mit der Resolutionsregel,

- kann jede Schlussfolgerung ableiten, die aus jeder Wissensbasis der Aussagenlogik folgt.
- bildet die Basis für Familien von vollständigen Deduktionsalgorithmen.
- beweist  $WB \models \alpha$  durch Widerlegung von  $(WB \wedge \neg \alpha)$ .

Voraussetzung: Sätze müssen in konjunktiver Form vorliegen.

**Klausel:** Eine Disjunktion von Literalen

$$(l_1 \vee \dots \vee l_n)$$

**Konjunktive Form (KF):** Jede aussagenlogische Formel kann als Konjunktion von Klauseln ausgedrückt werden. Dann liegt sie in konjunktiver Form vor:

$$(l_{1,1} \vee \dots \vee l_{1,k_1}) \wedge \dots \wedge (l_{n,1} \vee \dots \vee l_{n,k_n})$$

**Resolutionsalgorithmus**

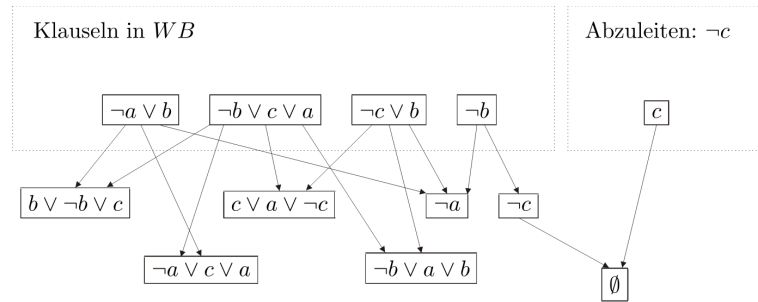
Zeige  $WB \models \alpha$  durch Widerlegung von  $(WB \wedge \neg \alpha)$

```

klauseln := Menge Klauseln in KF  $(WB \wedge \neg \alpha)$ 
neu := {}
loop:
  for all  $C_m, C_n$  in klauseln:
    res := resolution( $C_m, C_n$ )
    if  $\emptyset = res$ : return true
    neu := neu  $\cup$  res
  if neu  $\subseteq$  klauseln: return false
klauseln := klauseln  $\cup$  neu

```

Beispiel:



Der Resolutionsalgorithmus ist vollständig aber auch sehr aufwendig –  $O(n^2)$ . Realistische Vereinfachungen:

- Horn-Klauseln (Prolog), Vorwärts-/Rückwärts-Verkettung
- Davis-Putnam-Logemann-Loveland (DPLL)

## 8.6 Effiziente Logik-Algorithmen

**Horn-Klausel:** Disjunktion von Literalen, von denen höchstens eins positiv ist:

$$(\neg a \vee \neg b \vee c)$$

lässt sich schreiben als:

$$\underbrace{(a \wedge b)}_{\text{Koerper}} \Rightarrow \underbrace{c}_{\text{Kopf}}$$

Spezialtypen von Horn-Klauseln:

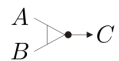
- keine negativen Literale: Fakt, Axiom
- genau ein positives Literal: Definition
- kein positives Literal: Integritätseinschränkung

Eigenschaften von Horn-Klausel-basierten Wissensbasen:

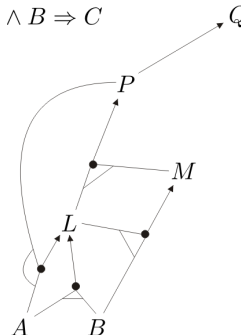
- einfach visualisierbar: Und-Oder-Graph
- Inferenz durch Vorwärts- und Rückwärtsverkettung, sehr natürliche und leicht verständliche Algorithmen
- Folgerungsentscheidung kann in linearer Zeit geschehen!

**Und-Oder-Graph:** einfache Visualisierung eines Horn-Klausel-Systems.

Legende:  $A \rightarrow B : A \Rightarrow B$



$P \Rightarrow Q$   
 $A \wedge P \Rightarrow L$   
 $L \wedge M \Rightarrow P$   
 $B \wedge L \Rightarrow M$   
 $A \wedge B \Rightarrow L$   
 $A \ B$



Vorwärtsverkettung:

- gehe von Fakten aus
- arbeite dich vorwärts durch den Baum
- Ergebnis: alle durchführbaren Folgerungsentscheidungen

Rückwärtsverkettung:

- gehe von Anfrage aus
- arbeite dich rückwärts durch den Baum
- stoppe bei bekannten Fakten (Ground Truth, Axiome)
- Ergebnis: Wahrheitsgehalt der Anfrage

**Davis-Putnam-Logemann-Loveland (DPLL):** Rekursiver Modellprüfungs-Algorithmus mit folgenden Verbesserungen:

- früher Abbruch: benutzt  $(A \vee B) \wedge (A \vee C) = true$ , sobald  $A = true$
- Einheitsklauseln (Klauseln mit nur einem Literal): Konjunktionen mit Einheitsklauseln sind nur wahr, wenn die Einheitsklauseln wahr sind  
 $\Rightarrow$  weitere Einschränkung des Suchraums
- reine Symbole: in  $(A \vee \neg B)$ ,  $(\neg B \vee \neg C)$ ,  $(C \vee A)$  sind  $A$  und  $\neg B$  rein und  $C$  unrein  
 Suche nach reinen Symbolen schränkt Raum möglicher Modelle stark ein!

Algorithmus:

```

funktion DPLL(k, s, m):
  # k Klauselmenge, s Symbolmenge, m Modell
  if alle Klauseln wahr in m: return true
  if eine Klausel falsch in m: return false
  P, wert := FINDE_EINHEITSKLAUSEL(k, s, m)
  if P nicht leer:
    return DPLL(k, s - P, SETZE_IN_MODELL(P, wert, m))
  P, wert := FINDE_REINES_SYMBOL(k, m)
  if P nicht leer:
    return DPLL(k, s - P, SETZE_IN_MODELL(P, wert, m))
  P := ERSTES(s); rest := REST(s)
  return DPLL(k, rest, SETZE_IN_MODELL(P, true, m)) or
         DPLL(k, rest, SETZE_IN_MODELL(P, false, m))

```

DPLL zur Überprüfung von Erfüllbarkeit:

```

function DPLL_ERFUELLBAR(s):
  # s : Satz der Aussagenlogik
  klauseln := Klauselmenge der KNF von s
  symbole := Menge der Symbole aus s
  return DPLL(klauseln, symbole, {})

```

## 8.7 Semantisches Planen

### Repräsentation von Plänen

Wie kann man Probleme so formulieren, dass

- ein Lösungsplan einfach zu erstellen ist?
- die Existenz einer Lösung bewiesen/widerlegt werden kann?

$\rightarrow$  Einschränkungsregeln für formelle Spezifikation von Zuständen, Zielen, Aktionen. Beispiele solcher Regelsysteme: STRIPS, ADL

**STRIPS**

- **ST**anford **R**esearch **I**nstitute **P**roblem **S**olver
- "Urvater" vieler Planungssysteme
- sehr einfach aufgebaut
- teilweise eingeschränkt

Repräsentation von Zuständen:

- Konjunktion positiver aussagenlogischer Literale

$$Blau \wedge Rund$$

- Literale erster Ordnung ( $L1$ )

$$IstTasse(T1) \wedge IstUntertasse(U1) \wedge StehtAuf(T1, U1)$$

- $L1$  müssen funktions- und variablenfrei sein!

**Closed-World Assumption**

- geschlossene Welt: was nicht im Zustand vorkommt, ist falsch!
- keine Negationen im Zustand und Vorbedingungen, aber in Nachbedingungen benötigt:

$$b \Rightarrow \begin{array}{l} a \text{ darf nicht im Weltmodell vorkommen} \\ b \text{ muss im Weltmodell vorkommen} \end{array}$$

- Negationen in Nachbedingungen:

$$\neg c \wedge d \Rightarrow \begin{array}{l} \text{entferne } c \text{ aus dem Weltmodell} \\ \text{füge } d \text{ dem Weltmodell hinzu} \end{array}$$

Repräsentation von Zielen:

- Ziel: teilweise spezifizierter Zustand
- angegeben als Konjunktion von positiven Literalen

$$StehtAuf(T1, U1) \wedge StehtAuf(U1, Tisch)$$

- ein Zustand  $S$  erfüllt ein Ziel  $Z$ , wenn er alle Literale in  $Z$  enthält

$$StehtAuf(T1, U1) \wedge StehtAuf(U1, Tisch) \wedge StehtAuf(Teller, Tisch)$$

erfüllt

$$StehtAuf(T1, U1) \wedge StehtAuf(U1, Tisch)$$

Aktionen werden angegeben durch

- Aktionsname
- Parameter
- Vorbedingungen
- Effekte

$$A = (N_A, P_A, V_A, E_A)$$



$$\begin{aligned}
& (StelleAuf, \\
& (Obj1, Obj2), \\
& IstUntersatz(Obj2) \wedge Auf(Obj1, Tisch) \wedge Auf(Obj2, Tisch), \\
& \neg Auf(Obj1, Tisch \wedge Auf(Obj1, Obj2))
\end{aligned}$$

Alternative Schreibweise:

$$\begin{aligned}
\textbf{Action} \quad & ( StelleAuf(Obj1, Obj2), \\
& \textbf{Vorbed:} \quad IstUntersatz(Obj2) \wedge \\
& \quad Auf(Obj1, Tisch) \wedge \\
& \quad Auf(Obj2, Tisch), \\
& \textbf{Effekt:} \quad \neg Auf(Obj1, Tisch) \wedge \\
& \quad Auf(Obj, Obj2) \quad )
\end{aligned}$$

Semantik:

**Anwendbarkeit:** Eine Aktion ist anwendbar auf allen Modellen  $M$ , die ein Modell für die Vorbedingung  $V_A$  sind.

**Ergebnis:** Das Ergebnis  $M'$  der Ausführung einer Aktion  $A$  auf einem Modell  $M$  erhält man durch

- entfernen aller negativen Literale des Effekts  $E_A$  aus  $M$
- hinzufügen aller positiven Literale aus  $E_A$  zu  $M$ .

Einschränkungen von STRIPS:

- Literale müssen funktionsfrei sein
  - endliche Grammatik
  - jede Aktion kann als endliche aussagenlogische Konjunktion dargestellt werden
  - Lösbarkeitsbeweis einfach
- Nur positive Literale  $l$ , "geschlossene Welt"
  - einfachere Planung
  - aber: Falsch-Sachverhalte nur implizit
- keine Quantisierung

$\Rightarrow$  STRIPS-Aussagen oft lang und unübersichtlich

## ADL

- **Action Description Language:** Weiterentwicklung von STRIPS
- "offene Welt": Was nicht im Zustand vorkommt, ist unbekannt.
- negative Literale und Disjunktionen erlaubt
- Effekt  $\neg P \wedge Q$ :
  - füge  $\neg P$  und  $Q$  hinzu
  - lösche  $P$  und  $\neg Q$
- Gleichheits-Prädikat "=" eingebaut
  - $StelleAuf(T1, T1)$  nicht mehr möglich

Typüberprüfung

- Typüberprüfung benötigt für Ausführbarkeit bestimmter Aktionen

- in STRIPS nur explizit (als Prädikat):  $IstUntersetzer(x)$ ,  $IstTasse(y)$
- häufig weggelassen (Schreibarbeit!), aber eigentlich nötig
- in ADL implizit möglich:

**Action** (  $StelleAuf(Obj1 : Untersatz, Obj2 : Manipulierbar)$ ,  
**Vorbed:**  $Auf(Obj1, Tisch) \wedge$   
 $Auf(Obj2, Tisch)$ ,  
**Effekt:**  $\neg Auf(Obj1, Tisch) \wedge$   
 $Auf(Obj, Obj2)$  )

Suche im Zustandsraum ist einfach:

- keine Funktionssymbole  
 $\Rightarrow$  endlicher Zustandsraum  
 $\Rightarrow$  Standard-Algorithmen zur Baumsuche (z.B. A\*)
- Transformation Vorbedingungen  $\Leftrightarrow$  Effekte bijektiv  
 $\Rightarrow$  Suche auch rückwärts möglich

Aber:

- Zustandsraum oft sehr groß
- naive Suche sehr ineffizient
- benötigt gute Heuristik oder Segmentierung in Teilprobleme

Heuristiken für Zustandsraumsuche

- versuche, die Suchtiefe einzuschränken
- Def. **Heuristikfunktion**:

$$h = H(M_0, M_1)$$

angenommene maximale Anzahl von Aktionen, um von  $M_0$  zu  $M_1$  zu kommen

- beschränke Suche auf Teilbaum der Tiefe  $h$
- wenn nicht gefunden: suche je eine Ebene tiefer
- Problem: finde möglichst gute Heuristikfunktionen

### Vorranggraph

Vor- und Nachbedingungen können zur Problemdekomposition verwendet werden. Parallelisierung möglich?  $\rightarrow$  Vorranggraph

Vorranggraph ermöglicht:

- Parallelisierung der Planung
- Parallelisierung der Ausführung
- bei dynamischen Effekten (Hindernisse, etc.)
  - Blockierung des momentanen Teilplans
  - zuerst Ausführung anderer Teilpläne
  - dann Überprüfung, ob blockierter Teilplan jetzt ausführbar (Hindernis jetzt weg?)
  - ggf. Neuplanung nur ab der letzten Gabelung notwendig

## 8.8 Umweltmodell

Das Umweltmodell eines kognitiven Systems bildet die reale Umwelt auf eine innere Repräsentation ab. Logisch-semantische Beziehungen genügen für ein System mit Aktorik nicht; zusätzlich:

- geometrisches Modell: Ausdehnung und Lage von Objekten
- topologisches Modell: (teil-geometrische) Beziehungen zwischen Objekten

**Geometrisches Modell:** Anwendungen:

- Bahnplanung feinkörnig
- aktives Messen
- Objekterkennung (Basisobjekte)
- fahren von  $x_s, y_s, z_s$  nach  $x_e, y_e, z_e$

**Topologisches Modell:** Anwendungen:

- Planung (Manipulationsplanung, mittlere Körnigkeit, Bewegungsplanung)
- "fahre von Raum1 nach Raum5"

Über topologische Modelle:

- Anordnung von Objekten und Umwelt relativ zueinander gespeichert
- abgeleitet aus geometrischen Modellen:  
Pfade z.B. aus Voronoi-Diagrammen, Quadtree, Potentialfeldern
- grobe Aktionsplanung aus topologischen Modellen
- kurzfristige Verwendung anderer Modelle bei unerwarteten Hindernissen (geometrischer Bodenplan, ad-hoc erzeugte Kantenmodelle aus Laserscan, Kamera)

**Semantisches Modell:** Anwendung: Planung auf Aufgabeneben → "fahre durch alle Büros"

Über semantische Modelle:

- besondere Bedeutung im Rahmen der Mensch-Maschine-Kommunikation
- belegte Fläche als Objekt klassifiziert (z.B. Schrank, Stuhl)
- eingeordnet in topologische Beschreibung
- Objekteigenschaften verwertbar: Objektzustand (Tür offen, Tasse leer); äußere Erscheinung des Objekts (Form, Farbe) als Diskriminator und Hilfe für die Sensorik; geometrisches Objektmodell für Greifplanung etc.

zugehörig zu semantischen Umweltbeschreibungen:

- Funktion eines Objektes
- Landmarken
- geometrische Objekte
- topologische Umweltdarstellungen
- Positionen

Umweltrepräsentation der Information:

- Pfade
- Freiraum
- Objekte
- gemischte Modelle

### Pfade

Normal: 2-dimensionaler Raum

- Polygonale Beschreibung der Objekte
- Pfad: lineare oder nichtlineare Verbindung zweier Punkte im Operationsraum
- Speicherung der Umwelt und Planungsinformation
- Repräsentation als ungerichteter Graph

Sichtbarkeitsgraphen:

- einfache Umweltdarstellung, partielle Modellierung
- kollisionsfreie Bewegung nur auf gespeicherten Pfaden
- Sichtbarkeitsgraphen zur automatischen Generierung von Pfaden

Erstellung eines Sichtbarkeitsgraphen:

```

for i = 1..#Obj
  for j = 1..#Objekte[i]
    for k = i..#Obj
      for l = 1..#Objekte[k]
        if (Sichtbarkeitstest(objekte[i][j], objekte[k], [l] == ok)
          neuer Pfad(i,j,k,l)

```

**Voronoi-Diagramme:** Fahrwege liegen auf maximalem Abstand zu den Hindernissen. Das Voronoi-Diagramm ist der duale Graph der Delaunay-Triangulation.

### Freiraum

- Projektion der realen Welt in 2D/3D-Grundrissdarstellung
- kollisionsfrei befahrbare Freiräume in geeignete Bereiche zerlegt
- vorhandene Objekte und Hindernisse nicht berücksichtigt

Freiraumgraph:

- Knoten: Freiraumbereiche
- Kanten: Verbindung der Gebiete

Vorteile:

- geringere Komplexität der Wegplanung
- reduzierter Zeit- und Rechenaufwand
- Fahrsicherheitsüberlegungen deutlich vereinfacht
- Verbesserung der Anpassungsfähigkeit der Algorithmen an verschiedene Umwelten

Darstellungsformen der Freiflächen:

- Kacheln (Quader)
  - Darstellung von besetzten Räumen durch orthogonale begrenzende Linien (Näherungen)
  - Verlängerung der Linien bis zum nächsten besetzten Raum
  - bildet Mosaik von besetzten und freien Kacheln/Quadern
  - freie Kacheln → Graph, durch den Bewegung möglich ist

Quadtree-/Octtree-Aufteilung:

- hierarchische Aufteilung in freie und belegte Zellen
  - Aufteilung teilbelegter Zellen in 4 (2D) bzw. 8 (3D) Unterzellen
  - Resultat: Baumstruktur
  - Pfadplanung: Aufstieg von Start- und Zielknoten bis zum ersten gemeinsamen Knoten; Liste besuchter Knoten: Pfad
- konvexe Polygone (Polyeder)

## Objekte

- Darstellung der Objekte der realen Umwelt (Türen, Wände, Hindernisse)
- 3-dim. Darstellung der Umgebung aus Sensorwahrnehmungen
- Projektion auf  $x$ - $y$ -Ebene für Navigation bodengebundener Roboter ausreichend

verschiedene Darstellungen:

- Kantenmodelle:  
Ermittlung von markanten Punkten. Verbinden durch geeignete Kanten auf Oberfläche des Objekts
- Oberflächenmodelle
  - Nachbildung der Objektoberflächen
  - Darstellung ebener Flächenelemente mit Polygonen
  - gekrümmte Flächenelemente:
    - \* mathematische Grundflächen (Zylinder, Kegel, Torusflächen)
    - \* Bezier-Flächen (Erweiterung des Ansatzes der Bezierkurven):  
gegeben ist ein Gitter von Führungspunkten  $P_{ij}$  mit  $0 \leq i \leq N$  und  $0 \leq j \leq M$ .  
Damit ist die Fläche beschrieben durch

$$F(u, v) = \sum_{i=0}^N \sum_{j=0}^M P_{ij} \cdot B_{i,N}(u) \cdot B_{j,M}(v)$$

mit

$$\begin{aligned} B_{i,N}(u) &= (1-u)B_{i,N-1}(u) + uB_{i+1,N-1}(u) \quad , \quad B_{i,0} = 1 \\ B_{j,M}(v) &= (1-v)B_{j,M-1}(v) + vB_{j+1,M-1}(v) \quad , \quad B_{j,0} = 1 \end{aligned}$$

Die  $B_{i,N}$  bzw.  $B_{j,M}$  heißen auch Bernsteinpolynome.

- \* näherungsweise durch Freiformflächen (Patches)
- Volumenmodelle:
  - Unterscheidung von Raumpunkten hinsichtlich ihrer Lage zum Objekt (innen-/außenliegend)
  - Repräsentationsmöglichkeiten: Begrenzungsflächenmethode, Grundkörperdarstellung, Zellenzerlegung (Octtree), Volumenapproximation, einhüllende Quader, Geradensegmente

## Modelle

Aufgliederung des benötigten Wissens:

- Ausführung von Aufgaben
  - semantisches Aufgabenmodell
  - Umweltmodell vorher, nachher
- Bewegung über Grund in 2D, Bewegung des Arms in 3D
  - statische Umweltkarte (geometrisch und topologisch)
  - dynamische Hinderniserfassung in 3D
  - Freiraum-, Hindernismodell
- Manipulation von Objekten
  - Objektpositionen
  - geometrische und topologische Objektmodelle

## 8.9 Geometrisches Planen

### Grundlagen der Bahnplanung

Bewegung eines Roboters:

- Zustandsänderungen über der Zeit (Trajektorie)
- relativ zu stationärem Koordinatensystem (kartesischer Raum, Gelenkwinkelraum)
- häufig Gütekriterien, Neben-, Randbedingungen

Bekannt:

- $S_{start}$ : Zustand zum Startzeitpunkt
- $S_{ziel}$ : Zustand zum Zielzeitpunkt

Gesucht:

- $S_I$ : Zwischenzustände (Stützpunkte)
- glatte, stetige Trajektorie

Bahnplanungsverfahren nach Zustandsraum:

- Gelenkwinkelzustandsraum (Konfigurationsraum)
- 3-dim. euklidischer Raum
- Sensorzustandsraum, Objektzustandsraum, ...

Bahnplanungsverfahren nach Art des Roboters:

- Bahnplanung für Manipulatoren
- Bahnplanung für mobile Roboter
- Bahnplanung für Laufmaschinen und antropomorphe Systeme
- Greif- und Montageplanung

**Bahnplanung im Gelenkwinkelraum**

- Trajektorie als Funktion der Gelenkwinkel
- Ausführung solcher Trajekturen durch
  - Steuerung der Achsen unabhängig voneinander (Punkt zu Punkt) oder
  - achsinterpolierte Steuerung (Bewegung aller Achsen beginnt und endet zum gleichen Zeitpunkt) erfolgen
- Bahnverlauf muss im kartesischen Raum nicht definiert sein
- Vorteile: einfach; keine Singularitäten

**Bahnplanung im kartesischen Raum**

- Trajektorie angegeben als Funktion der Endeffektorposition
- Funktionen z.B.: lineare Bahnen, Polynombahnen, Splines
- Vorteile:
  - Verlauf der Trajektorie explizit in 3D
  - einfach nachvollziehbar, visualisierbar
- Nachteile:
  - für jeden Punkt muss Gelenkwinkelrücktransformation berechnet werden
  - Trajektorie nicht immer ausführbar (Arbeitsraumbegrenzung, Singularitäten des Roboters)

**Bahnplanungs-Schema**

gegeben: Robotermodell (Geometrie, Kinematik), Umweltmodell

1. Berechnung des Konfigurationsraums  $K$
2. Berechnung des Hindernisraums  $H$
3. Berechnung des Freiraums  $F = K \setminus H$
4. Zerlegung des Freiraums in Unterräume
5. Bahnplanung im Unterraum
6. Integration der lokalen Lösungen in Gesamtlösung

**Konfiguration**

**Konfiguration  $k_R$ :** beschreibt den Zustand eines Roboters  $R$

- im euklidischen Raum durch Lage und Orientierung
- im Gelenkwinkelraum durch die Werte der Gelenke

**Konfigurationsraum  $K_R$ :** Raum aller möglichen Konfigurationen von  $R$

**Weg  $w$  von  $k_{start}$  bis  $k_{ziel}$ :** stetige Abbildung

$$w : [0, 1] \rightarrow K \quad \text{mit} \quad w(0) = k_{start}, w(1) = k_{ziel}$$

**Arbeitsraum-Hindernis  $h_{A,O}$ :** Raum, der von einem Objekt  $O$  im Arbeitsraum  $A$  eingenommen wird

**Konfigurationsraum-Hindernis**  $h_{K,O}$ : Raum, der von einem Objekt  $O$  im Konfigurationsraum  $K$  eingenommen wird

**Hindernisraum**  $H$ :

$$H = \bigcup_O h_{K,O}$$

**Freiraum**

- Freiraum  $F_R : F_R = K_R \setminus H$
- Aufwand für Freiraumberechnung:  $O(m^n)$  mit
  - $n$ : Anzahl der Freiheitsgrade des Roboters
  - $m$ : Anzahl der Hindernisse
- Deshalb oft approximative Verfahren zur Vereinfachung des Freiraums
  - Sichtgraphen
  - Quadtree, Octtree

## 8.10 Bahnplanung in 2D

Einfacher Algorithmus: "Strassenkarten"

gegeben: 2-dim. Weltmodell, Start und Ziel

gesucht: günstigste Verbindung von Start zu Ziel

Lösung:

1. konstruiere Netz von Wegen  $W$  in  $F_R$
2. bilde  $k_{start}$  und  $k_{ziel}$  auf  $W$  ab:  $(W(k_{start}), W(k_{ziel}))$
3. suche Weg  $w$ , der  $W(k_{start})$  mit  $W(k_{ziel})$  verbindet

### Wegkonstruktion

Wegkonstruktion mit

- Retraktionverfahren (z.B. Voronoi-Diagramm)
- Sichtgraphen
- Zellzerlegungsmethoden

Suche im Wegnetz mit z.B.

- A\*-Algorithmus (Baumsuche)
- euklidischer Abstand
- Potentialfeld

**Retraktion:** Sei  $X$  eine Menge und  $Y \subset X$ . Eine surjektive Abbildung

$$p : X \rightarrow Y$$

heisst Retraktion genau dann, wenn  $p$  stetig ist und  $p(y) = y$  für alle  $y \in Y$  gilt.

D.h. die Abbildung der Menge  $X$  auf ihre Teilmenge  $Y$ , wobei die Menge  $Y$  auf sich selbst abgebildet wird. Für die Bahnplanung gilt:

- $Y$  ist ein Netz von eindimensionalen Kurven (Wegenetz).
- Retraktionsmethoden unterscheiden sich in Wahl von  $p$ .



## Sichtgraphen

Konstruktion:

- Verbinde jedes Paar von Eckpunkten auf dem Rand von  $F_R$  durch gerades Liniensegment, wenn das Segment kein Hindernis schneidet.
- Verbinde  $k_{start}$  mit  $k_{ziel}$  analog dazu.

Anmerkungen:

1. Wege sind nur "halbfrei" (nicht kollisionsfrei), da Hinderniskanten auch Wegsegmente sein können.  
Abhilfe: Erweiterung der Hindernisse
2. Wenn ein Weg gefunden ist, ist es auch der kürzeste Weg.
3. Methode ist exakt, wenn Roboter nur 2 translatorische Freiheitsgrade hat und sowohl Roboter als auch Hindernisse durch Polygone dargestellt werden können.
4. Methoden auch im  $R^3$  anwendbar, jedoch sind die gefundenen Wege i.A. keine kürzesten Wege mehr.

## Kürzester Pfad im Graphen

vom Startknoten ausgehend:

- wähle Nachbarknoten  $N_k$  so, dass Evaluationsfunktion  $f(N_k)$  minimal
- suche von  $N_k$  ausgehend weiter
- wenn  $f(N_k)$  nicht mehr kleiner wird, mache weiter oben im Baum weiter
- Problem:  $f(N_k)$  darf nicht vom Teilbaum an  $N_k$  abhängen!  
(Würden wir den kennen, bräuchten wir nicht suchen.)
- Lösung: verwende Heuristik  $h(N_k)$
- beliebt für die Heuristikfunktion in 2D: euklidischer Abstand

$$h(N_k) = ||N_k - N_{ziel}||$$

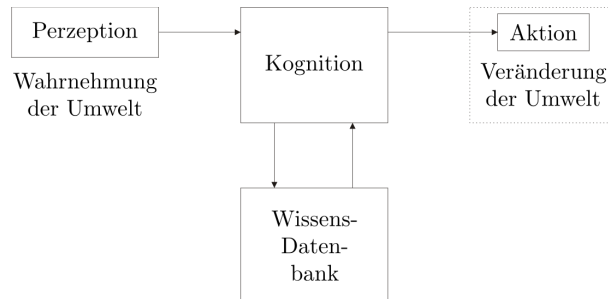
- rein heuristikbasierte Suche ist nicht optimal und führt nur unter Einschränkungen zum Ziel
- A\*-Algorithmus:  
mit  $g(N_k)$  Kosten zum Erreichen von  $N_k$
- beweisbar optimal

$$f(N_k) = g(N_k) + h(N_k)$$

## 9 Robotik

### 9.1 Einführung

Aufbau eines kognitiven Systems:



Aktion:

- Ausführung einer geplanten Handlung
- Ergebnis des Verarbeitungsschritts  $\text{Perzeption} \rightarrow \text{Kognition} \rightarrow \text{Aktion}$
- Computersystem mit aktorischer Komponente: "Roboter"
  - *robota [poln.]*: Fronarbeit
  - zuerst verwendet in dem futur. Theaterstück "Rossum's Universal Robots" von Karel Capek

Typen von Robotersystemen:

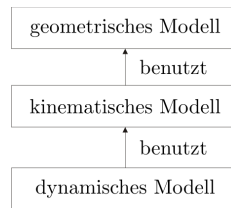
- Industrieroboter, automatische Fertigungssysteme, Roboterarme
  - stationär, hochspezialisiert, allgemein verfügbar
  - Anwendungen: Schweißen, Schneiden, Fräsen, Lackieren, Bestücken, ...
- Autonome mobile Roboter
  - mobil, auf Transportaufgaben ausgerichtet, noch recht selten
  - Anwendungen: Lagerhaltung, Waren-, Nachrichtentransport, Führungen, ...
- Humanoide Assistenzsysteme
  - mobil, allgemeinbefähigt, Gegenstand aktueller Forschung
  - Anwendungen: Fertigungs-, Haushaltsassistent, Pflege, Begleitung, Schutz, ...  
→ "Alleskönner"

Probleme der Robotik:

- Kinematik:
  - Lehre der Bewegung von Körpern
  - Wohin bewegt sich der Roboter, wenn er Gelenk  $a$  auf  $45^\circ$  einstellt?
  - Wie müssen die Gelenke eingestellt werden, damit sich der Roboter von  $A$  nach  $B$  bewegt?
  - Ist diese Bewegung möglich?
- Dynamik
  - Lehre von Geschwindigkeiten, Kräften, Momenten
  - Wieviel Kraft muss der Roboter aufwenden, um Gelenk  $a$  auf  $45^\circ$  einzustellen?
  - Welche Kräfte entstehen bei Bewegung  $A$  nach  $B$ ?
  - Geht dabei auch nichts kaputt?
- Konfigurations-, Hindernis-, Freiraum, Kollisionsvermeidung, ...

## 9.2 Robotermodellierung

Zum Lösen der Probleme der Robotik Kinematik, Dynamik, Bahnplanung stellt man verschiedene Modelle auf den Roboter auf:



### Geometrische Modellierung

#### Geometrisches Modell

- dient als Grundlage zur Berechnung der Bewegung
- wird benutzt zur Ermittlung der wirkenden Kräfte und Momente
- stellt den Körper graphisch dar
- ist Ausgangspunkt der Abstandsmessung und Kollisionserkennung

Komplexitätsstufen des geometrischen Modells:

1. Bounding-Box-Modell
  - die Körper werden durch einhüllende Quader dargestellt
  - wird in den ersten Schritten der Kollisionsvermeidung benutzt
2. Kantenmodell
  - die Körper werden durch Polygonzüge (Kanten) dargestellt
  - wird zur schnellen Visualisierung (Rechenleistung für Grafik) benutzt
3. Volumenmodell
  - die Körper werden genau dargestellt
  - wird für die Ermittlung der genauen Werte der Kollisionserkennung benutzt

### Kinematische Modellierung

**Kinematisches Modell** ermittelt mit Hilfe des geometrischen Modells die Stellung (Position und Orientierung) von Körpern im Raum.

Eine **kinematische Kette** wird von mehreren Körpern gebildet, die durch Gelenke kinematisch verbunden sind (z.B. Roboterarm)

Zweck des kinematischen Modells:

- Bestimmung des Zusammenhangs zwischen Gelenkwerten und Stellungen
- Erreichbarkeitsanalyse: Welche Punkte sind anfahrbar, ohne dass innere oder äußere Kollisionen auftreten?

Aufstellung des kinematischen Modells erfolgt iterativ. Jedes Armelement entspricht einem starren Körper. (Bsp.: Puma 260, 6-achsiger Roboter, Basis und 6 Armelemente)

Zu jedem Körper existiert ein Objekt-Koordinatensystem

Roboterbasis	$\text{OKS}_{\text{Basis}}$
Armelement 1	$\text{OKS}_{\text{Arm1}}$
$\vdots$	$\vdots$
Armelement 6	$\text{OKS}_{\text{Arm6}}$

Koordinatensysteme werden urch homogene Transformationen miteinander verknüpft.

### Wdh. homogene Koordinaten

$4 \times 4$ -Matrizen, die geschlossen Rotation und Translation ausdrücken.

$$\begin{aligned}
 p + Rot(\alpha, p') &= \begin{pmatrix} rot_{0,0}(\alpha) & rot_{0,1}(\alpha) & rot_{0,2}(\alpha) & 0 \\ rot_{1,0}(\alpha) & rot_{1,1}(\alpha) & rot_{1,2}(\alpha) & 0 \\ rot_{2,0}(\alpha) & rot_{2,1}(\alpha) & rot_{2,2}(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & x' \\ 0 & 1 & 0 & y' \\ 0 & 0 & 1 & z' \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \\
 &= \begin{pmatrix} rot_{0,0}(\alpha) & rot_{0,1}(\alpha) & rot_{0,2}(\alpha) & x' \\ rot_{1,0}(\alpha) & rot_{1,1}(\alpha) & rot_{1,2}(\alpha) & y' \\ rot_{2,0}(\alpha) & rot_{2,1}(\alpha) & rot_{2,2}(\alpha) & z' \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
 \end{aligned}$$

### Denavit-Hartenberg-Transformation

Die Transformation vom OKS des  $i$ -ten Armelements auf das OKS des  $(i-1)$ -ten Armelements erfolgt nach der **Denavit-Hartenberg (DH)-Konvention**:

- (a) die Koordinatensysteme liegen in den Bewegungsachsen
- (b) die  $z_{i-1}$ -Achse liegt entlang der Bewegungsachse des  $i$ -ten Armelements
- (c) die  $x_i$ -Achse steht senkrecht zur  $z_{i-1}$ -Achse und zeigt von ihr weg

( $i \in \{Basis, 1, \dots, n\}$ )

#### DH-Parameter

Transformation von OKS $_{i-1}$  auf OKS $_i$ :

1. Eine Rotation  $\theta_i$  um die  $z_{i-1}$ -Achse, damit die  $x_{i-1}$ -Achse parallel zu der  $x_i$ -Achse liegt (wegen Voraussetzung (c)).

$$R_{z_{i-1}}(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Eine Translation  $s_i$  entlang der  $z_{i-1}$ -Achse zu dem Punkt, wo sich  $z_{i-1}$  und  $x_i$  schneiden (wegen Voraussetzungen (b), (c)).

$$T_{z_{i-1}}(s_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & s_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Eine Translation  $a_i$  entlang der  $x_i$ -Achse um die Ursprünge in Deckung zu bringen (wegen Voraussetzung (c)).

$$T_{x_i}(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Eine Rotation  $\alpha_i$  um die  $x_i$ -Achse um die  $x$ - und  $y$ -Achse in Übereinstimmung zu bringen.

$$R_{x_i}(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In MATrizenschreibweise lautet die Denavit-Hartenberg-Transformation von Armelement  $i - 1$  auf  $i$ :

$$\begin{aligned}
 A_{i-1,i} &= R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(s_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) \\
 &= \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \cdot \cos(\alpha_i) & \sin(\theta_i) \cdot \sin(\alpha_i) & a_i \cdot \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \cdot \cos(\alpha_i) & -\cos(\theta_i) \cdot \sin(\alpha_i) & a_i \cdot \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & s_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

$\theta_i$  und  $s_i$  sind variable Größen während der Bewegung des Roboters, abhängig von dessen spezieller Geometrie und Maßen.  $\alpha_i$  und  $a_i$  dagegen sind sowohl bei Rotations-, als auch bei Schubgelenken invariante Größen und müssen für die spätere Berechnung der direkten Kinematik nur einmal für jedes einzelne Armelement bestimmt werden.

### Gelenktypen

Um einen Roboter zu bewegen, sind Gelenke erforderlich (üblicherweise elektrisch angetrieben). Gelenktypen:

- Rotationsgelenk: Drehachse in  $90^\circ$  zu den Achsen der angeschlossenen Glieder.
- Torsionsgelenk: Drehachse parallel zu den Achsen der angeschlossenen Glieder.
- Revolvergelenk: Eingangsglied parallel, Ausgangsglied rechtwinklig zur Drehachse.
- Lineargelenk: gleitende Bewegung entlang der Achse.

## 9.3 Kinematik

Zwei Probleme:

- Bei gegebenem Gelenkwinkel: Wo ist der Greifer?  
→ direktes kinematisches Problem
- Bei gegebener Greifer-Sollposition: Wie müssen die Gelenkwinkel eingestellt werden?  
→ inverses kinematisches Problem

### Direktes kinematisches Problem

Um die Stellung des Greifers in Bezug auf das BKS zu bestimmen, werden die DH-Matrizen in der Reihenfolge der Armelemente multipliziert.

Bei  $n$  Armelementen und  $n$ -dimensionalem Gelenkwinkelvektor  $\theta$ :

$$p_{TCP}(\theta) = \prod_{i=1}^n A_{i-1,i}(\theta_i)$$

### Inverses kinematisches Problem

Aus den DH-Parametern und der Stellung des Greifers sollen die Gelenkwinkel bestimmt werden. Die Gleichung

$$p_{TCP}(\theta) = \prod_{i=1}^n A_{i-1,i}(\theta_i)$$

muss daher nach  $\theta$  aufgelöst werden. Für 6 Gelenke sind dies 12 Gleichungen mit 6 Unbekannten. Die Gleichungssysteme sind überbestimmt und damit existiert keine eindeutige Lösung.