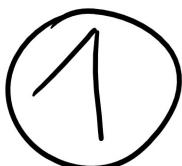


Intelligente Autonome Systeme iAS



Einführung

1. VE

Motivation

Märkte für iAS

Markt	Aktuell	Kurzfristig (5-10 J.)	Langfristig (10-20 J.)
Logistik ("Transport" von irgendwelchen Firmen) z.B. Pakete liefern, später auch Personen	Spurungebun- dene Fahrertrans- portsysteme Manipulation (z.T. auch spur- gebunden)	Autonome mobile Ro- boter (z.B. Batterie greifen, ohne zu hinfahren und einstecken) Autonome Einzeldrohnen	Kooperative mobile Roboter (z.B. in der Pflege, beim Arzt -> Emotionen zeigen etc.) „Drohenschwärme“
Flugzeuge	Ferngesteuerte Drohnen		
Automobil	Fahrerassistenz- Systeme (Lvl. 3)	Unfallvermeidendes Auto- mobil	Bedarfsweise autono- mes Fahrzeug (Lvl. 5) ↳ eigenständige Ein-/ Abschaltung (Autonomie)
Hausgeräte	Staubsauger, Rasenmäher etc.	Interaktive Dienst- leistungssroboter	Autonome Dienst- leistungssroboter
Raumfahrt	Ferngesteuerte Roboterfahrzeu- ge (wenig Auto- nomiefunktionen)	Autonome Roboter- fahrzeuge	„Roboter schwärme“

Stufen autonomes Fahren SAE J3016 ✓

Folgerungen:

- Prognostizierter Wachstumsmarkt
- hoher Personalbedarf

Zielsetzung:

- Programmierkonzepte (3 SWS)
- Laborübungen (1 SWS C107)

Überblick:

- Hardwareaufbau (Sensoren etc.)
- Softwarearchitektur
- Steuerungskonzepte
- Umweltmodellierung
- Kooperation („Schwarmintelligenz“ etc.)

2

Hardwareaufbau

2.1 Basiskomponenten

Fahrzeuge

Vorlesung -> a) Ebene Untergrund: Räder

- Literatur { b) Gelände: Laufmaschinen
c) Rohre (z.B. Kanalisation): Kettenantriebe
d) Fassaden (z.B. Fensterputzen): Saugnäpfe
e) Katastrophenszenarien: alle HW-Komponenten

2. VE

Flugzeuge

Klassischer Triebwerksantrieb mit Flügeln, Multicopter

Computergesteuerte unbemannte Luftfahrzeuge (Drohnen)

~> Fokus auf Multicopter

2.2 Multisensorik

Erfassung der Umwelt (nicht ein universeller Sensor: mehrere Umweltinflüsse und Messtechniken
→ Multisensoren)

Odometrie

Radposition, Geschwindigkeit und Weg (über Zeit) ermitteln

↳ Ungenauigkeiten („Radschlupf“, Reifendurchmesser, Reifenabnutzung etc.)

Inertialgeber (Gyrokop verbaut)

Lageheute nicht mehr so weit verbreitet

Beschleunigung ermitteln,
Verbesserung Odometriedaten,

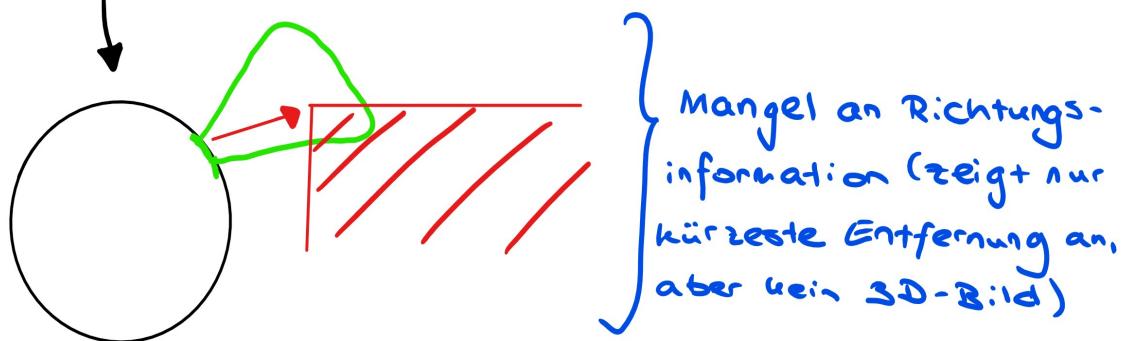
Taktile Sensoren (Nachbildung menschl. Tastsinns)

LD nehmen Berührungen wahr

Bumper, Anwesenheitserfassung, Notabschaltung

Sonar

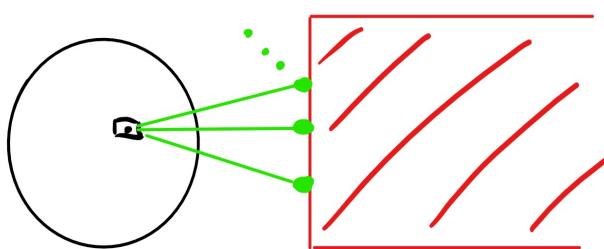
Entfernung mit Ultraschall messen (z.B. Parkassistent),
Kegelförmige Signalausbreitung, Öffnungswinkel 5° - 30° ,
Rin abdeckung:



Probleme: Öffnungsbereich, falsche Quellen: z.B.
Absorption (Schaumstoff), Ablenkung (Metall),
Fremdschall etc.

Laserscanner

Entfernung messen mit Laserlicht, Öffnungswinkel vernachlässigbar, Reflektorenerkennung



Probleme:

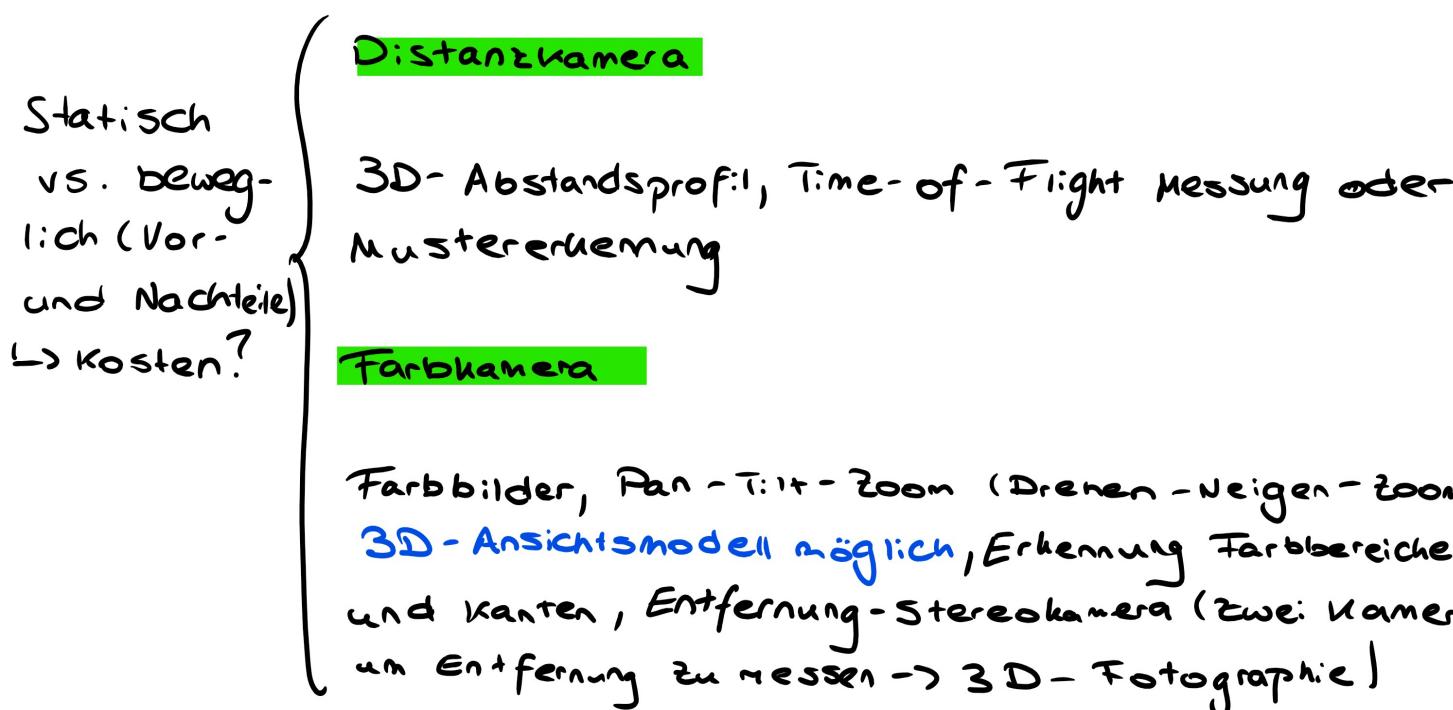
Lichtverhältnisse un-
geeignet, Spiegel lenkt
Strahl ab (Roboter
krallt dagegen), Laser-
absorption

Radar

Entfernung messen durch elektromagnetische Wellen (2,5 GHz), Variation Frequenzen (Short Range, Long Range), Relativgeschwindigkeiten messbar

Probleme:

Reflexion, Brechung, Beugung ungeeignet (Radiowellen ähnlich wie Lichtwellen) → Richtung der Wellen veränderbar



GNSS (globales Navigationssatellitensystem)

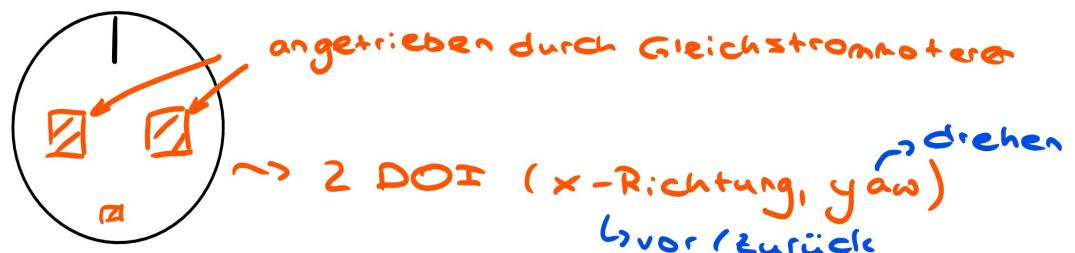
Satellitennavigation, Genauigkeit 2m / 2cm, Positionsbestimmung, Verwendung bestehender Satellitensysteme GLONASS (Russland), Galileo (Europäische Union) etc.

Veränderung der Umwelt, Manipulatoren (zur Veränderung der Umwelt) mit Freiheitsgraden (z.B. Roboterarm mit 6 Freiheitsgraden: x-, y-, z-Richtung + Rotation)

~ 2 DOF Gruppen im Labor (Greifer + Lift)
 ↳ Degrees of freedom

Antriebskonzepte Räder ~ Handout

* Differentialantrieb → simples System

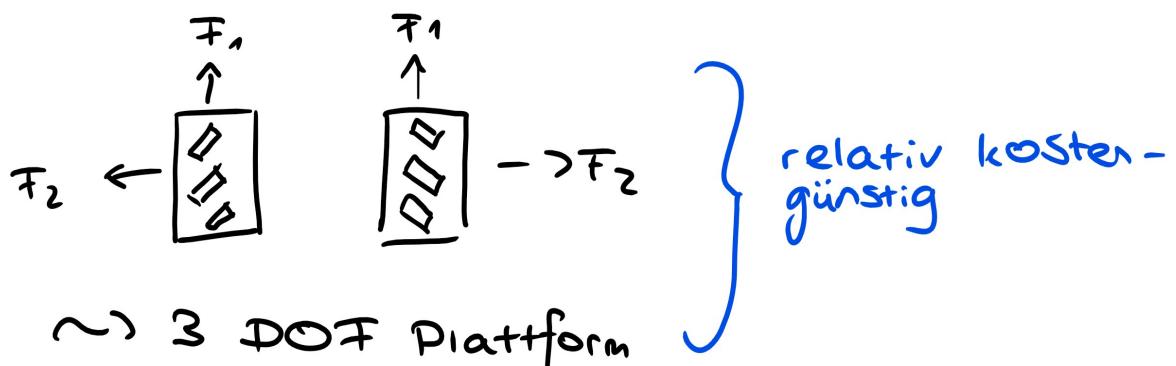


* Fahr- / Drehmodul

~ 2 DOF Rad ~ 3 DOF Plattform
 (Fahren, Drehen)
 ↳ vor/zurück
 ↳ komplexeres System, teuer!

→ (Fahren, Drehen, Schräg-
 fahren)
 ↳ vor/zurück

* Mecanum - Rad



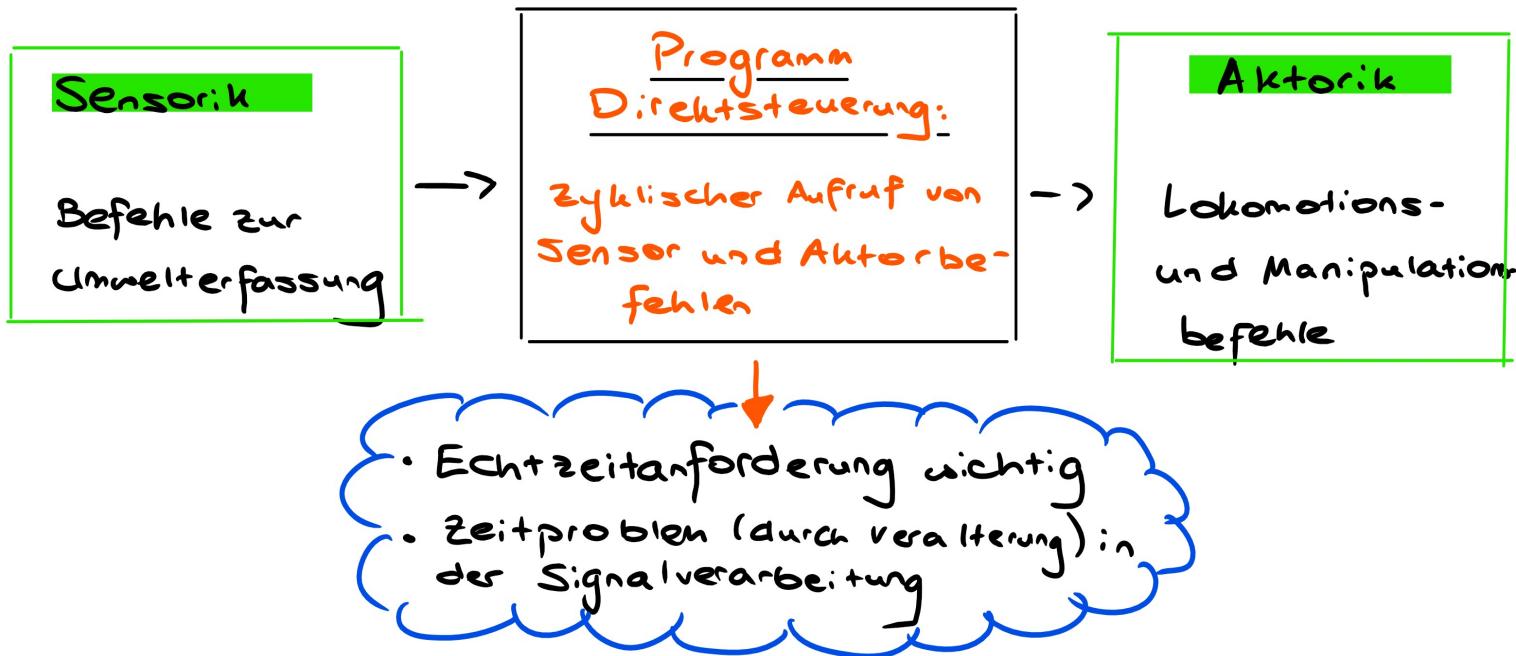
↪ Nachteil: ungeeignet für schlechten Boden (z.B. Bremsen), Schleift!

3

Softwarearchitekturen

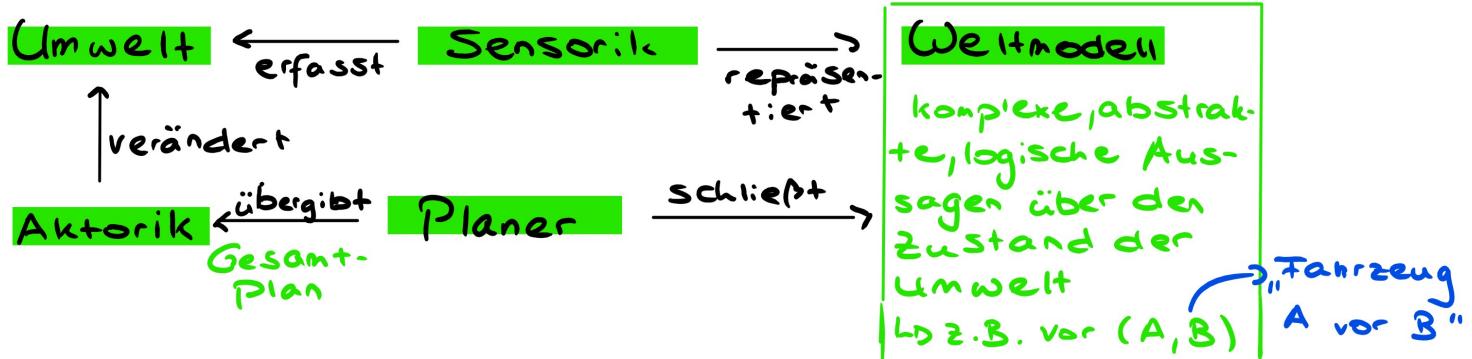
3.1 Direktsteuerung

Regelungstechnischer / Signalverarbeitungstechnischer Ansatz 2



3.2 Sense - Plan - Act

geht auf 60/70er Jahre zurück, Nilsson (erste Schritte)



Vorteile: wertfähig (Gedächtnis durch Planer), Plan \rightarrow Zielführung!
Nachteile: Komplexität des (Um-)Weltmodells ist gigantisch \rightarrow Widersprüchlichkeiten können entstehen, Planerstellung dauert z.T. lange, Planerstellung nicht flexibel (Plan sagt: Nur Bus um 11-12 Uhr fährt kurz aus)

3.3 Subsumptionsarchitekturen

gehen zurück auf Brooks, MIT, 1986

Frage: Warum kann ein Wurm mit seinem relativ kleinen Gehirn (z.B. Fadenwurm mit 302 Neuronen) in Echtzeit reagieren? (Und wir nicht?)

- ↳ Wurm hat verschiedene Verhaltensmuster. Je nach Situation, wird eines dieser Muster unabhängig von den anderen angewendet → Aktion
- ↳ Ein Verhaltensmuster weiß von anderen nichts (Ein- und Abschaltung) der Muster möglich
- ↳ Priorisierung in Verhaltensmustern: Nahrungssuche < Überleben → geschichtete Muster, abhängig von Szenario, dynamische Schichtung)

Idee: elementare Verhaltensmuster bilden die Grundbausteine für Intelligenz

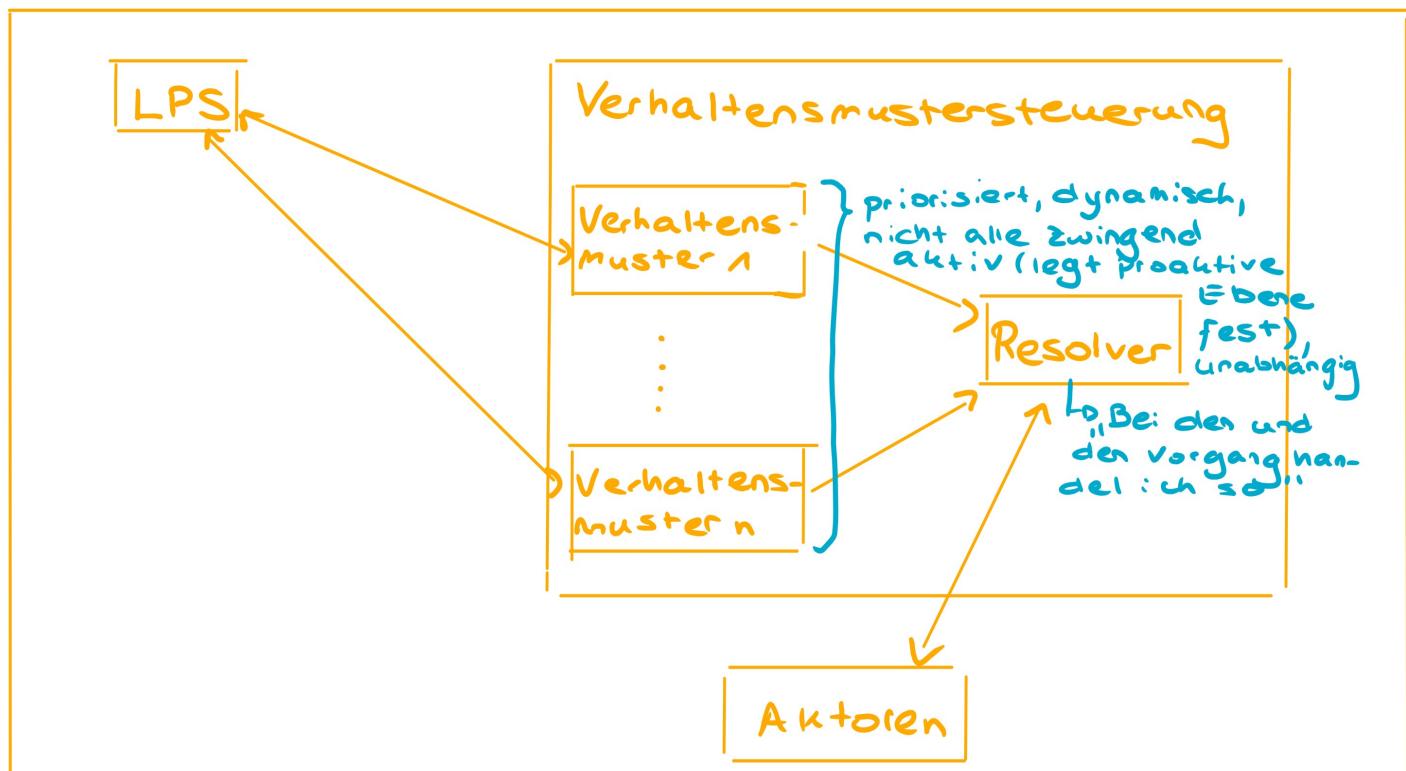
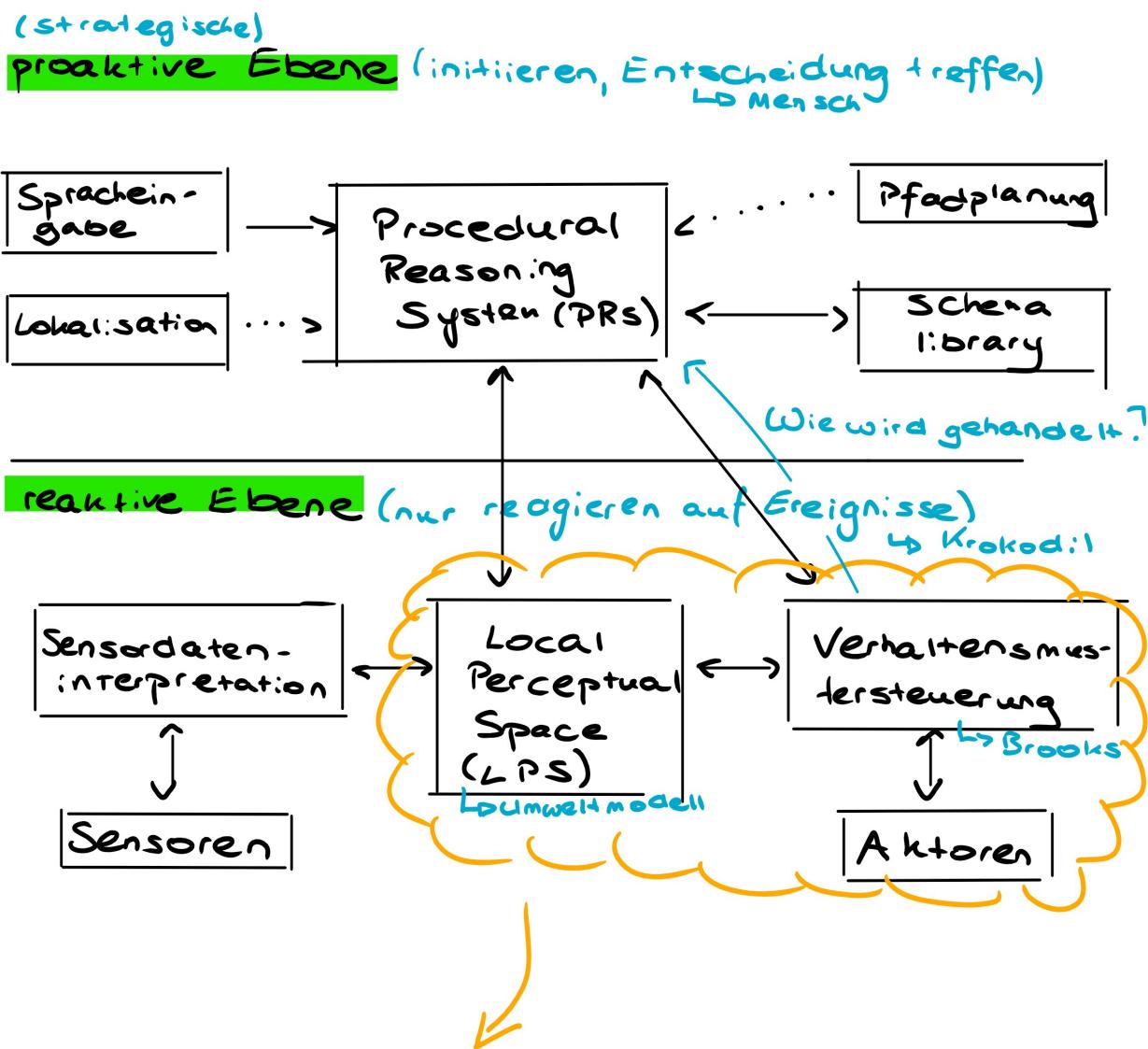
Beispiel:

- Erkundungsverhalten (wie ist die Lage?
wie verhalte ich mich?)
- Hindernisvermeidung
- Nahrungsanfrahme

Situation einschätzen, Gefahren erkennen

Verhaltensmuster (engl. behaviours) haben unterschiedliche Komplexität, Jedes Verhaltensmuster ist eigenständig funktionsfähig, Niedere Schichten unterdrücken die höheren!

entwickelt von Kondo Ido, SRI, 90er Jahre



- Die Verteilermaster berechnen gewünschte Aktorvorgaben, Resolver verknüpft die Wünsche zu konkreten Steuerbefehlen
- PRS verwendet Schemata von Aktivitäten aus Schema Library. Jedes Schema ist ein parametrierbarer endlicher Automat
- THI RobCon ist einfache Implementierung von Direktesteuerung und Saphira in Java

4

Direktesteuerung

Skriptsprache in THI RobCon basierend auf Java

Lokomotionsbefehle

	Translation (vor/zurück)	Rotation
Geschwindigkeit (nicht blockierend)	void speed (int v) v in $\frac{\text{mm}}{\text{s}}$ ↳ positive/negative Werte (vor/zurück)	void rotate (int av) av in $\frac{\text{deg}}{\text{s}}$
Position (Odometrie blockierend)	void move (int d) d in mm	void turn (int a) a in deg

Manipulationsbefehle (2 DOF)

```
void griOpen ()  
void griClose ()  
void liftUp ()  
void liftDown ()
```

nicht blockierend } nicht blockierend

nicht blockierend } void panTilt (int pan, int tilt)
↳ Drehen Kamera auf pan, Neigung Kamera auf tilt
↳ int pan, int tilt in deg

Sensorbefehle

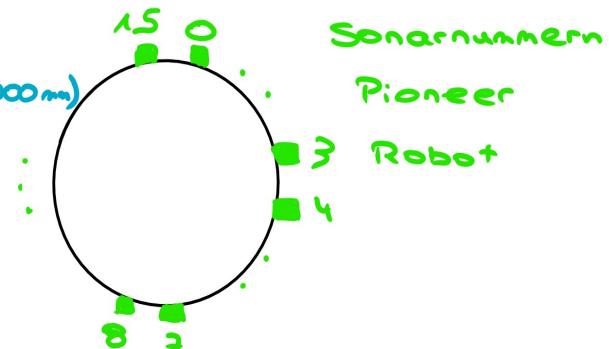
Distanz: int getSonarRange(int n)

- min: set minimale Hindernisdistanz

- int sonar n in mm

- Default 5.000 (erst ab 3.000m)

- Bezug Robotermittpunkt



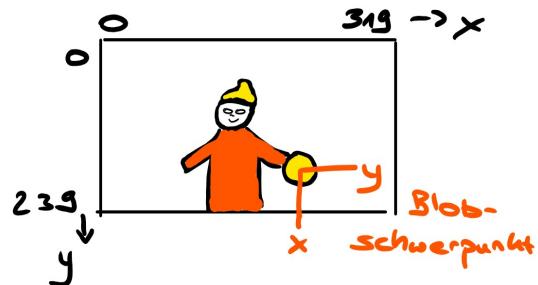
Bilder: int getBlobX(int ch)

ch Kanalnummer ($\textcolor{red}{c} \geq 1, \textcolor{yellow}{c} \leq 2$)

Offline konfiguriert

Blobberechnung über Region-Growing mit Schwerpunkt

Rückgabe: x - Koordinate Blobschwerpunkts



Kontrollstrukturen

void wait(int n)

↳ Warten von n Echtzeitzyklen zu je 100ms

void setPos(int x, int y, int th)

↳ Orientierung

↳ setzen der Roboterposition in Simulation