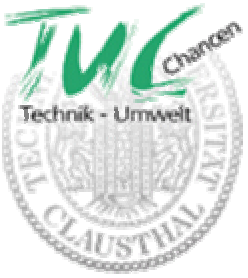


Technische Universität Clausthal Institut für Informatik

Wintersemester 2001/02

Ausarbeitung des Seminarvortrages

„Technische und physikalische sowie weitere Eigenschaften großer fussballspielender Roboter, der Middle-Size-League des RoboCups“

in der Vortragsreihe

„CI-Methoden in der Robotik“

bei Herrn Dr. Reuter

Robert Hartmann
(Matrikelnummer: 302300; Studiengang: Informatik)

Inhaltsverzeichnis:

Vorwort:

Im RoboCup geht es darum, in einem Fußballmatch mehrere autonome Roboter in einem Team gegen ein anderes Team von Robotern antreten zu lassen.

Middle-Size-League — F2000:

Teilnehmer der *Small-Size-League* und der *Middle-Size-League* müssen sich dazu bis zu fünf echte Roboter anschaffen oder bauen.

Abgesehen von der *Legged-Robot-Leagu*e, die sich noch in einem sehr frühen Teststadium befindet, stellt diese Klasse derzeit die mit den komplexesten Systemanforderungen dar. Im Gegensatz zur Simulatorliga müssen hier reale Roboter in einer realen Welt beherrscht werden, und im Vergleich zur *Small-Size-League* sind die einzelnen Roboter wirklich autonom und nur auf ihre lokalen Sensoren angewiesen.
Die Roboter der *Middle-Size-League* müssen zudem ihre komplette Steuerung "on-board" haben.

Da es verschiedene Ansichten darüber gibt, welche speziellen Eigenschaften und Fähigkeiten ein Roboter-Spieler haben soll, gibt es Forschungsgruppen, welche auf Standard-Modelle wie dem Pioneer1 aufbauen, und wiederum andere Forschungsgruppen, die ihre eigenen Roboter entwickeln.
Daher ist es ersichtlich, dass sich RoboCup-Roboter in vielen Bereichen unterscheiden:
z.B. Computerhardware, Sensorik, Mechanik, Kontrollsoftware, Taktik, Kosten und Design.

Im Gegensatz zu meiner ursprünglichen Annahme, dass *Middle-Size-League* Roboter vollkommen autonom - d.h. komplett ohne externen Coach oder Trainer - spielen würden, scheint es gängige Praxis zu sein, einen solchen externen Coach einzusetzen, der allerdings über keine eigenen „Sinnesorgane“ verfügt, sondern auf die von den Kameras der Spieler übermittelten Bilder angewiesen ist.

Eigenschaften:

Definition:
Eigenschaften sind zum Wesen einer Person oder Sache gehörende beschreibende Merkmale.

Die Eigenschaften eines Roboters lassen sich in drei große Bereiche aufspalten, zum einen in den physikalischen-technischen Hardware-Bereich zum anderen in den Software-Bereich, und in den Bereich „Sonstiges“ (z.B. taktisches Können, Kosten).

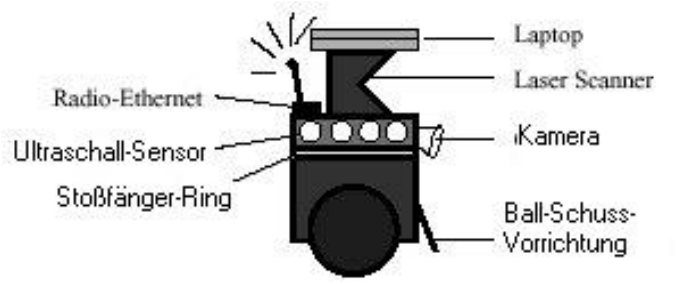
1. Technische und physikalische Hardware-Eigenschaften

Definition:

- Hardware: sämtliche technischen Teile einer elektronischen Datenverarbeitungsanlage (DVA).

1.1 Allgemeines

Im Allgemeinen besitzen die Roboter der *Middle-Size-League* zunächst ein dreistöckiges Fahrgestell: Meist aus Aluminium, nur die Rückseite besitzt eine geschlossene Wand, beide Seiten und die Front sind zumeist offen oder mit einer dünnen umschlagbaren Schicht bedeckt. So kann man leicht den charakteristischen Hardware-Aufbau erkennen:



Als unterste Ebene ist eindeutig das Fahrgestell zu sehen; in den beiden darüber liegenden Ebenen sind die einzelnen Hardware-Komponenten des „Rechners“ untergebracht und fest montiert.

Abbildung 1: Schema eines Spielers

Die Roboter der *Middle-Size-League* haben eine Grundfläche von 2000cm², was einem Durchmesser von etwa 50cm entspricht. Die gesamte Sensordatenverarbeitung, Planung und Robotersteuerung läuft lokal auf den Roboterrechnern ab.

Figure 2 - The 5dpo team posing near the yellow goal

Abbildung 2: Größenvergleich, Beispiel: Roboter-Team 5dpo

Als Sensoren kommen vor allem Kameras, Laser-Entfernungsmesser sowie taktile „Fühler“ und Ultraschallsensoren zum Einsatz.
Um die Bildverarbeitung etwas einfacher zu gestalten, haben alle relevanten Objekte definierte Farben.

Im Gegensatz zur *Small-Size-League* sind globale Kameras nicht zugelassen, d.h. den Robotern steht a-priori nur die durch die lokalen Sensoren gelieferte, relativ eingeschränkte Sicht auf die Szene zur Verfügung. Allerdings können diese Informationen ausgetauscht und zu einer „globalen“ Weltsicht fusioniert werden. In Anbetracht der Unsicherheiten bei der Objekterkennung, insbesondere aber bei der Bestimmung der eigenen Position, ist dies jedoch keineswegs trivial und stellt eine der großen Herausforderungen im Bereich der verteilten Sensordatenverarbeitung dar.

Objekt	**Farbe**
Ball	Rot
Tore	Blau bzw. Gelb
Spielfeld	Grün
Linien	Weiß
Bande	Weiß mit schwarzer Schrift
Roboter	Schwarz mit Markierungen in Zyan bzw. Magenta

Tabelle 1: Farben in der *Middle-Size-League*

Entscheidend ist dabei das Kamera-System, dass in drei verschiedenen Varianten vorhanden ist:

- omnidirektional,
- binokular
- und feste oder schwenkbare Einzelkamera-Systeme.

1.2 Kamera und Sensoren

Omnidirektionale Systeme bestehen aus einer nach oben gerichteten Kamera, über der ein Hohlspiegel montiert ist, sodass eine 360° Sicht ermöglicht wird.

Dagegen ist das Sichtfeld einer Einzelkamera ziemlich klein, um verlässliche Informationen zu erlangen, da so unter Anderem störende Effekte von Linsen minimiert werden. Dies kann aber kompensiert werden, indem die Kamera schwenkbar montiert wird. Bei Systemen mit mehreren Kameras kann auch ohne Bewegung ein großes Sichtfeld erzielt werden, es werden allerdings auch mehrere Framegrabber, das sind Analog-Digital-Wandler, und genauere Kalibrierung notwendig, um bei Überlappungen der Sichtbereiche Inkonsistenzen vorzubeugen.

Sensor		Wiederholzeit τ_{cg}	Datenrate f_g
Kamera-Halbbild	PAL	20 ms	50 s^{-1}
	NTSC	16.67 ms	60 s^{-1}
Kamera-Vollbild	PAL	40 ms	25 s^{-1}
	NTSC	33.33 ms	30 s^{-1}
Roboter-Odometrie	Pioneer 1	100 ms	10 s^{-1}
Ultraschallsensoren – beliebiger Sonar	Pioneer 1	100/2 ms	20 s^{-1}
– kompletter Satz von 7 Sonaren		100∗7/2 ms	2.86 s^{-1}

Tabelle 2: Technische Daten der Sensoren in der *Middle-Size-League*

Im RoboCup kommen unterschiedliche Kamera- und Sensorentypen vor.
Als Beispiel sei hier der Laserscanner SICK-LMS200 genannt – dieser kommt beim CS Freiburg auf den Pioneer1-Robotern neben der Sony EVI-D30 (pan-tilt) Kamera auf dem Torwart, einem Pioneer DX-2, zum Einsatz.

SICK-LMS 200

Abbildung 3: SICK-LMS 200

Spezifikationen

- **Reichweite** max.150 m
- **Reichweite (ohne zusätzliche Reflektoren)** bis zu 30 m
- **Reichweite mit minimalem Reflexionsvermögen 1.8%** 4 m
- **Auflösung** 10 mm
- **Statistische Fehler**
  - +/- 15mm Reichweite 1 bis 8 m
  - (mm-Auflösung) Reflexionsvermögen: 10% - 10,000%
  - +/- 4 cm Reichweite 8 to 20 m
  - (mm-Auflösung) Reflexionsvermögen: 30% - 10,000%
- **Winkelbereich** 100° Scann: 0.25°/0.5°/1°, 180° Scann: 0.5°/1° wird konfiguriert durch Software
- **Reaktionszeit** 52/26/13 ms anhängig vom Winkelbereich (180° Scann in 25 ms)
- **Interface** Serial RS-422 or RS-232 umstellbar
- **Echtzeit-Messdaten-Transfer** als Binärdaten
- **Datentransferrate** 9.6/19.2/38.4/500 Kbaud
- **Ausgänge** 3 x PNP, Typ. 24 V DC, zugeordnet zu 3 zu überwachenden Gebieten
- **Betriebsspannung** 24 V DC +/-15%
- **Laser Protection Class 1** (Augen sicher)

Tabelle 3: SICK-LMS 200

EVI-D30

Effektive Pixel:	768 × 492 (NTSC) 751 × 585 (PAL)
Zoom:	12-fach
Blickwinkel:	4,4 ~ 48,8 Grad
Belichtungszeit:	1/60 ~ 1/10000 (NTSC) 1/50 ~ 1/10000 (PAL)
Blende:	1.8 ~ geschlossen
Verstärkung:	-3 dB ~ +18 dB
Schwenkung:	± 100 Grad bei maximal 80 Grad/Sekunde
Neigung:	± 25 Grad bei maximal 50 Grad/Sekunde

Tabelle 4 / Abbildung 4: EVI-D30

1.3 Lokalisierung

In der Middle-Size-League ist die Lokalisierung ein kritisches Problem, da globale Positionierungssensoren verboten sind. Die Lokalisierungsmethodik der Roboter muss aufgrund der folgenden Charakteristiken der Umgebung gewählt werden:

- Die Geometrie der Wände und der Markierungslinien ist bekannt.
- Die Umgebung ist hochdynamisch.
- Die Aufgabe (Lokalisation) muss kontinuierlich über einen langen Zeitraum erfolgen.
- Die Umgebung kann nicht verändert werden.
- Kollisionen zwischen Robotern sind möglich.

Zusammen ergibt dies ein schwieriges Szenario für die Methoden der Lokalisierung.
Ein geeignetes Verfahren zur Positionsbestimmung ist der Abgleich der Informationen der Sensoren mit einem Modell der Umgebung, beispielsweise der Vergleich der erkannten Markierungslinien mit einer gespeicherten Abbildung des Spielfeldes. Diese Idee setzt allerdings Kenntnisse über die Umwelt voraus, die in gewissen Bereichen jedoch gegeben sind. Eine Methode dieser Art ist die Lokalisierung basierend auf der Hough-Transformation, die für segmentierbare Umgebungen geeignet ist und bei RoboCup erfolgreich eingesetzt wurde. Die Eigenschaft der Hough-Transformation ist die Erkennung von Linien aufgrund einer gegebenen Menge von Pixeln auf einer Ebene. Außerdem werden relative und absolute Distanzen und Winkel der Linien sichtbar und damit aktuelle Position und Drehwinkel des Roboters.
Die Hough-Transformation bildet Punkte eines kartesischen Koordinatensystems auf den so genannten Hough-Raum ab:

$$(x; y) \rightarrow (q ; p)$$

Der Bildraum der Abbildung ist ebenfalls ein zweidimensionaler Raum, dessen Vektoren allerdings keine Pixel, sondern Geraden repräsentieren. Die Komponenten eines Vektors im Hough-Raum werden folgendermaßen interpretiert:
q ist der Winkel der Geraden zur Horizontalen, p die Entfernung vom Ursprung. Damit ist jede Gerade im zweidimensionalen Raum eindeutig bestimmt. Nun kann jedem einzelnen

Pixel natürlich keine eindeutige Gerade zugeordnet werden. Deswegen wird jedes Pixel auf eine bestimmte Menge an Geraden abgebildet, und zwar auf die, die durch diesen Punkt verlaufen könnten. Dem Pixel (x; y) wird nun eine Funktion

$$p(q) = x * \cos q + y * \sin q$$

zugeordnet, die unter der Bedingung $0 \leq q < p$ folgendermaßen interpretiert werden kann:
Alle Punkte auf dieser Kurve im Hough-Raum stehen für "potentielle" Geraden, in denen dieses Pixel liegen könnte bzw. die durch den Punkt (x; y) verlaufen. Werden nun mehrere Pixel, die auf einer solchen Geraden liegen auf den Hough-Raum abgebildet, so schneiden sich alle resultierenden Kurven in genau dem Punkt, der diese Gerade repräsentiert.
Definiert man nun auf dem Hough-Raum ein Raster und ermittelt für jede Rasterzelle die Anzahl der Kurven, die diese schneiden, so erhält man ein zweidimensionales Feld in dem die Geraden des Urbildes durch hohe Zahlen repräsentiert werden.

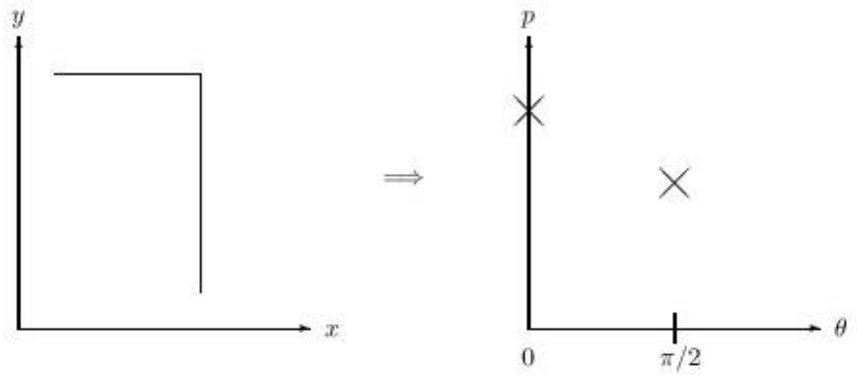


Abbildung 5: zwei Geraden (links) werden in den Hough-Raum (rechts) abgebildet

Die Hough-Transformation hat eine Reihe vorteilhafter Eigenschaften:
Lokale Maxima von HT geben die Gerade an, die am ehesten durch mehrere Pixel verläuft, auch wenn diese nicht exakt darauf liegen.
Das Verfahren ist robust gegen Störungen des Bildes durch verteilte Pixel die auf keiner Geraden liegen oder fehlende Bereiche.

Zum Beispiel das ART Team verwendet unter anderem diese Methode der Lokalisierung. Davor müssen allerdings aus den Rohdaten der Kamera oder der Sensoren die Linien extrahiert und entzerrt werden.

1.4 Motoren (Antrieb/Schussvorrichtung):

Die Roboter besitzen DC und auch Steppermotoren, die direkt von einem 8-bit Risc Mikrocontroller angesteuert werden. Der DC-Motor wird einzig und alleine für die Schussvorrichtung und die schwenkbare Kamera benützt. Die Steppermotoren steuern die einzelnen Räder an.

Es gibt unterschiedliche Ansätze für den Antrieb, zum einen sei der des CS-Freiburg genannt. Der CS-Freiburg baut seine Feldspieler auf den Robotern der Pioneer1 und den Torwart auf dem Pioneer DX auf. Beiden Pioneers ist der Antriebs Aufhängung fast identisch, (wobei der DX breitere Reifen besitzt).

Die Pioneers haben drei oder vier Räder, wobei zwei an der starren nicht lenkbaren Vorderachse befestigt sind, gesteuert wird der Pioneer über die Hinterachse; eine gewisse inverse Ähnlichkeit zur Lenkachse eines normalen Pkws ist nicht von der Hand zuweisen.

Abbildung 6: Pioneer-Roboter

Ein ganz anderer Ansatz ist der des amerikanischen Teams CMU Hammerheats:
Ihre Roboter haben genau wie die des CS-Freiburgs eine feste Achse, jedoch ist es hier die hintere und Vorderachse ist um 360° drehbar, d.h. es ist den Robotern des amerikanischen Teams möglich direkt, sozusagen aus dem Stand heraus, eine ganz andere Richtung einzuschlagen ohne großes Wendemanöver machen zu müssen. Das mag Vorteile in gewissen Situationen, wie zum Beispiel bei Bedrängnis durch einen gegnerischen Spieler haben. Jedoch ist die Steuerung dann wesentlich schwieriger

Noch ganz anders hat es das Golem Team aus Italien gelöst:
Ihre Roboter sind nicht kastenförmig sondern haben besitzen einen zylindrischen wenn nicht gar fassartigen Körper. Alle Räder sind fix montiert, der Roboter besitzt keine

Steuerachse. Die nicht rollenden Seiten aller Räder zeigen zum Zentrum, wobei drei Räder angetrieben werden. Wenn der Roboter geradeaus fahren will, ist mindestens ein Rad immer als Widerstand anzusehen. Einen besonderen Vorteil hat diese scheinbare sinnlose Konstruktion, sie ist verhältnismäßig einfach zu steuern, man braucht nur folgende Gleichungen zu implementieren, und schon hat man die Steuerung nach X und Y bzw. geradeaus.

Sei nun r der Rad-Radius, R die Distanz zwischen einem Rad und dem Zentrum des Roboters, und w_1, w_2, w_3 sei die Winkelgeschwindigkeit der Räder 1 bis 3:

$$v_x = \frac{1}{\sqrt{3}} \cdot r \cdot w_3 - \frac{1}{\sqrt{3}} \cdot r \cdot w_2$$

Nun ergibt sich:

$$v_y = -\frac{1}{3} \cdot r \cdot w_3 + \frac{2}{3} \cdot r \cdot w_1 - \frac{1}{3} \cdot r \cdot w_2$$

$$v_{vor} = \frac{r}{3 \cdot R} \cdot w_3 + \frac{r}{3 \cdot R} \cdot w_1 + \frac{r}{3 \cdot R} \cdot w_2$$

Erstaunlicherweise soll für den Fall, dass $w_3 = -w_2$ und $w_1 = 0$ ist, $v_x = 1{,}15 \cdot r \cdot w_3$ größer sein als die Geschwindigkeit, die von Robotern mit ähnlichem Antrieb wie dem des CS-Freiburgs erreicht werden können.

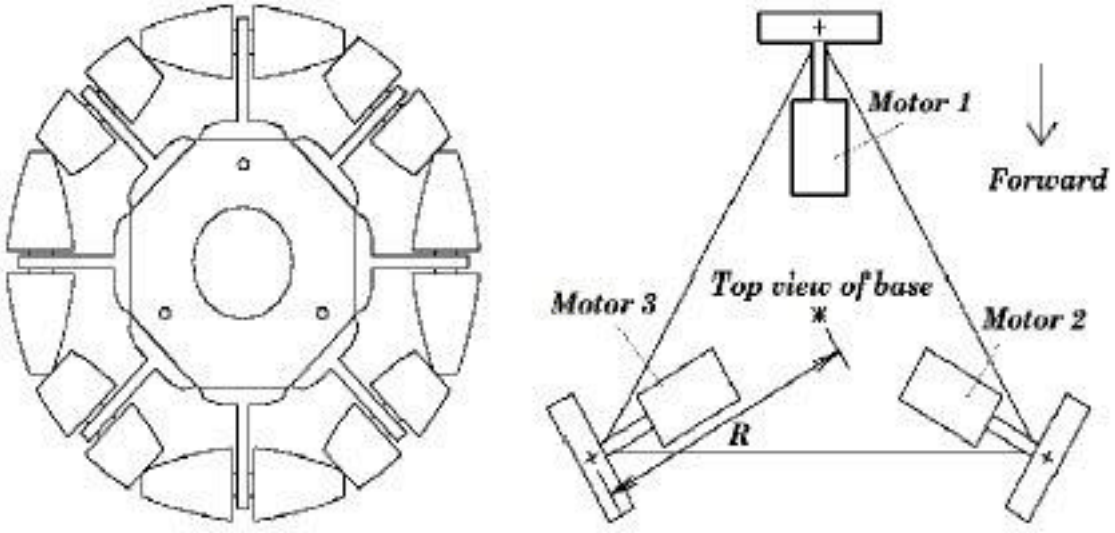


Abbildung 7: Golem-Roboter Antrieb

Geschwindigkeit:[1]

Roboter der Middle-Size-League besitzen eine Geschwindigkeit von ca. 0,5 bis 1,5 Meter pro Sekunde. Bernhard Nebel vom CS-Freiburg: "... In Paris 1998 lagen wir noch bei maximal 80 Zentimeter pro Sekunde, jetzt fahren wir etwa doppelt so schnell."

[1] aus "Roboterfußball ist Grundlagenforschung; Hans-Arthur Marsiske im Gespräch mit Bernhard Nebel, Professor für Grundlagen der Künstlichen Intelligenz an der Universität Freiburg" 01.08.2001

1.5 Koordination

Eine zentralisierte Koordination der Middle-Size Roboter ist wohl kaum zu realisieren: Einerseits wegen der Möglichkeit des Versagens der Kommunikation, andererseits wäre es nur schwer möglich eine zuverlässige globale Beschreibung der Umwelt zu konstruieren, die Basis für alle Entscheidungen wäre. Deswegen werden autonome Roboter eingesetzt, von denen eine eigenständige Teamkoordination abverlangt wird.

Die Roboter nutzen Kommunikation via Funk zum Austausch und Abgleich von Informationen über die Spielsituation, das heißt über Positionen von Spielern und Ball, sowie für die Unterstützung einer dynamischen Rollenverteilung innerhalb des Teams. Die Besonderheiten hierbei sind zum einen die Heterogenität der Roboter in Bezug auf Hardware und Software: Das Kommunikationsprotokoll muss sehr flexibel sein. Zum anderen sind die Spieler auch beim Versagen der Kommunikation einsatzfähig, die Kommunikation unterstützt die Koordination und steigert dadurch die Teamperformance.

Ein weiteres Feature des Teams ist die dynamische Rollenverteilung. Die Roboter sind also nicht individuell für eine bestimmte Funktion in der Mannschaft programmiert, sondern können während des Spiels je nach Situation unterschiedliche Aufgaben übernehmen, z.B. Verteidigung oder Angriff. Die Verteilung der Rollen erfolgt über den Austausch von individuell berechneten Utility Values, dass sind Werte, die pro Roboter und für jede Rolle unterschiedlich aufgrund der aktuellen Spielsituation berechnet werden. Durch Vergleich der eigenen Utility Values mit denen der anderen Roboter können Rollen eindeutig vergeben werden.

1.6 Zusätzliche Hardware-Ausstattung

Manche Roboter verfügen über einen integrierten Kompass zur sofortigen Bestimmung der eigenen Ausrichtung, zusätzliche Infrarot- und Sonarsensoren, die vor allem der Positionsbestimmung von anderen Spielern dienen. Außerdem sind alle Roboter mit Funkgeräten mit hoher Bandbreite und einem Standard-PC ausgestattet. Zumeist handelt es sich bei diesem Standard-PC um einen Laptop mit einem Prozessor mit mindestens den positiven Eigenschaften eines Intel Pentium 200 MMX bis zum P III mit 450 MHz, 64

MB-Ram scheint den Anforderungen zu genügen (jedenfalls habe ich kein Team gefunden, dass ausdrücklich in seiner Teambeschreibung auf mehr Arbeitsspeicher hinweißt) Das Laptop besitzt eine 2 bis 10 Mbit Radio LAN mit einer Frequenz von ca. 2,4 GHz .

Zur Führung und zum Schuss des Balles verfügen sie auch über spezielle, pneumatische Mechanik. Pneumatische Mechanik sind mechanisch funktionierende Teile, die sich durch Luftdruck an Formen anpassen können.

2. Software-Eigenschaften:

Definition:

- Software… sind im weitesten Sinne grundsätzlich Programme für eine Datenverarbeitungsanlage (DVA).
- System-Software umfasst alle Befehlsfolgen, die für den Betrieb der Anlage nötig sind und ihre Leistungsfähigkeit bestimmen
- Anwendungs-Software enthält Programme, mit deren Hilfe die eigentlichen Aufgaben der DVA gelöst werden

Die in der *Middle-Size-League* benutzte Team-Software – als Anwendungs-Software – setzt sich architektonisch gesehen aus drei verschiedenen Bausteinen zusammen.

- Zum einen die so genannte "Supervisor Application" (SUP),
- zum anderen die "Hardware Abstraction Layer" (HAL)
- sowie "Decision and Cooperation agent" (DEC).

2.1 Supervisor Application (SUP)

Die Software-Applikation SUP läuft auf einer Maschine außerhalb des Spielfeldes, sie ist verantwortlich für das Team-coaching, also für Veränderung in Taktik und Spielerverhalten, weiterhin informiert SUP die spielenden Roboter vorhandene Kommunikationskanäle nutzend über nützliche Gesamtsituationsbewertungen.

2.2 Hardware Abstraction Layer (HAL)

HAL - auf jedem Roboter on-board installiert - ist der Software-Baustein, der dort jede Aktion und Wahrnehmung des Roboters kontrolliert. Er analysiert die Daten auf "high-level information" - also auf Informationen von hoher Nützlichkeit -, welche von den einzelnen "Wahrnehmungsorganen", den Sensoren, des Roboters gesammelt wurden.
HAL sendet diese gefilterten Informationen zum nächsten Baustein, der DEC Anwendung.

2.3 Decision and Cooperation agent (DEC)

Die Softwareanwendung DEC ist der Baustein, welcher dem Roboter es ermöglicht, aufgrund gesammelter Daten eigene Entschlüsse zu ziehen und quasi selbständig zu agieren und zu reagieren. Weiterhin ermöglicht dieser Baustein in Kombination mit den Kommunikationskanälen, dass gesammelte Daten entweder über den SUP oder auf direktem Wege an andere Roboter mitgeteilt werden, um die Team-Koordination zu verbessern.

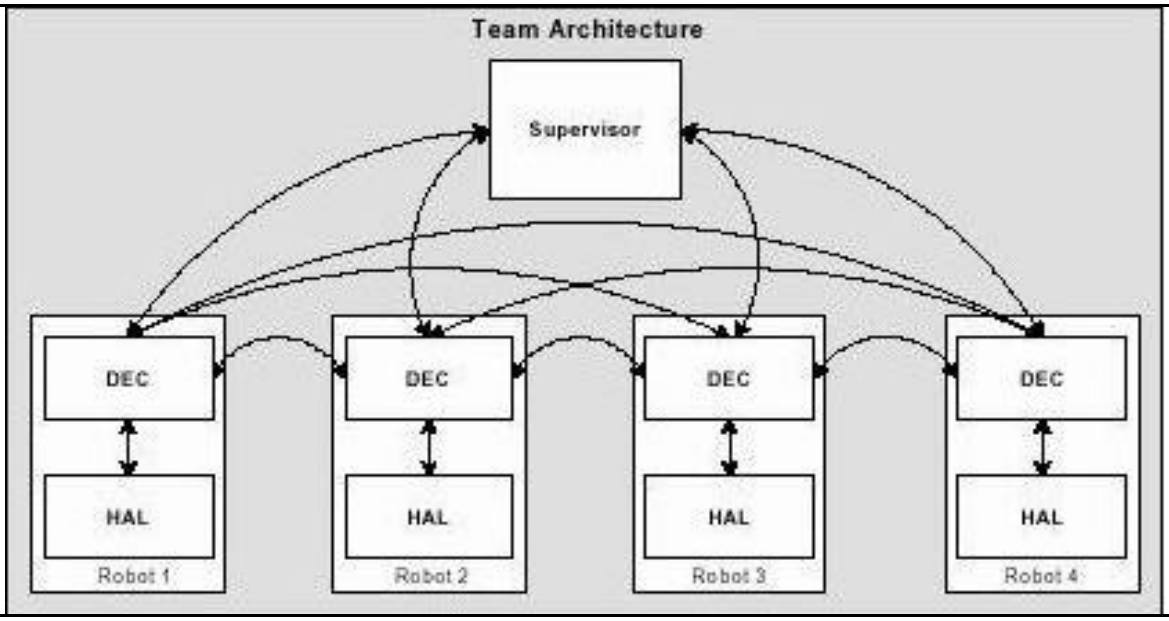


Abbildung 8: Team Architecture

2.4 Programmentwurf

Nicht nur die Software (DEC, HAL und SUP) haben gewisse Eigenschaften, sondern auch der Programmentwurf:
Die untere Graphik zeigt die hierarchische Gliederung des Programmentwurfs für den spielenden Roboter - in den einzelnen verschiedenen Abstraktionsebenen.

1. Die Systemspezifikationen, also das was den Roboter ausmacht – sozusagen die Wesensbeschreibung – werden in einem Pflichtenheft zusammen getragen.
2. Nun muss das Komplett-System in funktionale Bereiche aufgeteilt werden. Beim Roboter bieten sich die drei großen Bereiche der Sensordatenverarbeitung, der Planung und der Robotersteuerung an.
    a. Die Sensordatenverarbeitung beinhaltet:
        i. Die Selbstlokalisation, d.h. der Softwarebereich, der es dem Roboter ermöglicht, seine Position zuerkennen.
        ii. In der Farberkennung werden die Farben bekannten Gegenständen zugeordnet.

    Das Sensordatenverarbeitungs-Modul besitzt sendende Schnittstellen zum Planungs-Modul und zum Robotersteuerungs-Modul.
    b. Im Planungs-Modul, muss die künstliche Intelligenz, z.B. die Taktik, Kombinatorik usw., verankert sein. Das Planungs-Modul besitzt empfangende Schnittstellen zum Sensordatenverarbeitungs-Modul und sendende zum Robotersteuerungs-Modul.
    c. Das Robotersteuerungs-Modul besitzt empfangende Schnittstellen zum Sensordatenverarbeitungs-Modul und zum Planungs-Modul, es besitzt sendende Schnittstellen zur Hardware des Roboters, es ist direkt „funktional-verantwortlich“ für die Bewegungen der Maschine.
3. Im high-level Implementierungsschritt werden die Module mit hohen Programmiersprachen sowohl objektorientiert /modular als auch funktional strukturiert.
4. Im nächsten Schritt, werden die Bibliotheken eingebunden, als da wären, die Bildfolgen-Bibliothek, die Bildverarbeitungs-Bibliothek (Halcon) und die Basis-Bibliotheken (z.B. Standard C / C++)

5. Hier als letzter Schritt generiert ein passender Compiler aus dem Quellcode ein Maschinenprogramm im Maschinencode, welches direkt von der Maschine ausgeführt werden kann.

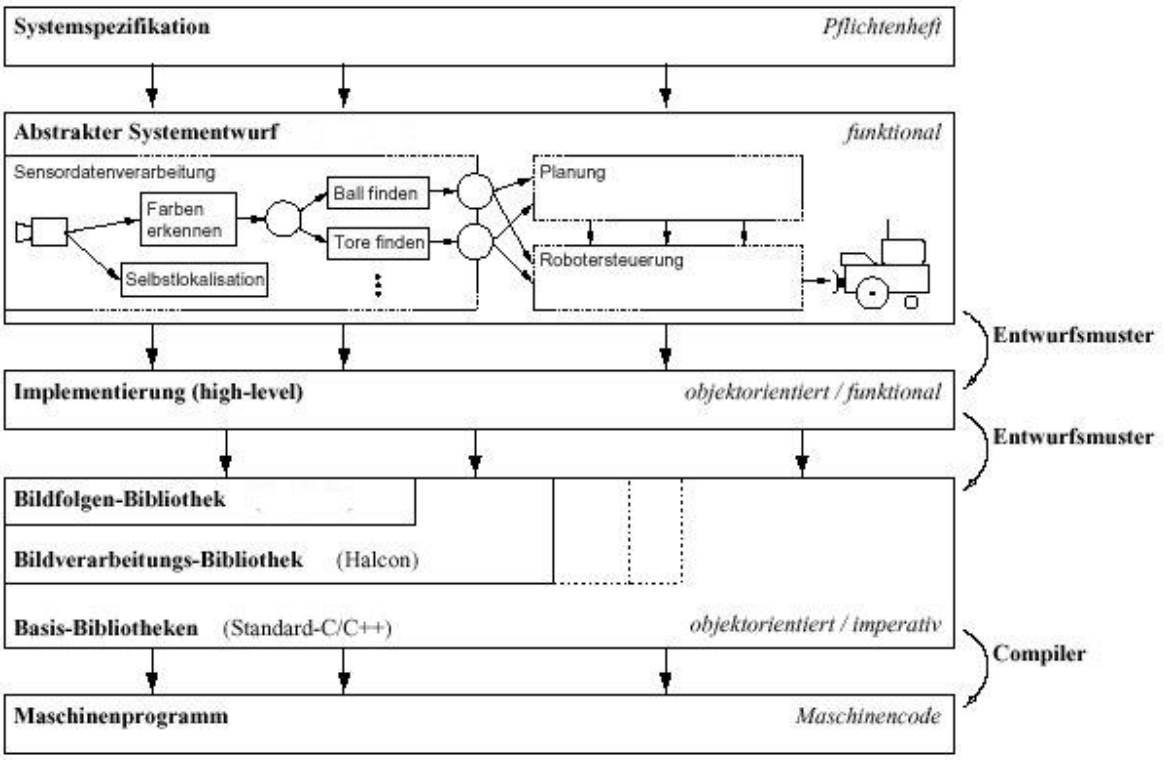


Abbildung 9: Hierarchische Gliederung (in verschiedene Abstraktionsebenen) des Programmentwurfs für den spielenden Roboter

3. Weitere Eigenschaften:

3.1 Kosten:[2]

Bernhard Nebel vom CS-Freiburg: "...gegenwärtig macht es kaum Sinn beliebig viel Geld in ein Team zu stecken. Mit 200.000 Mark sind die Kosten für den Aufbau einer Robotermannschaft ganz gut beziffert."
Ein Pioneer 1 Roboter (benutzt von verschiedenen Teams) kostet ca. 2500 US $.

[2] aus "Roboterfußball ist Grundlagenforschung; Hans-Arthur Marsiske im Gespräch mit Bernhard Nebel, Professor für Grundlagen der Künstlichen Intelligenz an der Universität Freiburg" 01.08.2001

3.2 Gewicht:

Ein Pioneer 1 (ohne zusätzlich aufmontiertes) wiegt ca. 12 kg

3.3 Design:

Um zu zeigen, dass es keinen einheitlich designten Roboter beim RoboCup gibt, hier ein paar Abbildungen verschiedener Typen:

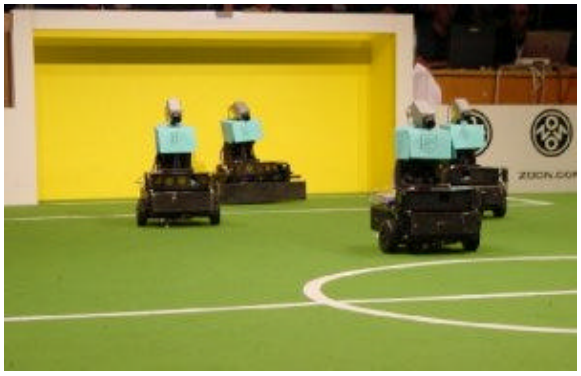


Abbildung 10: Team: CS-Freiburg

Abbildung 11: ein weiters Team

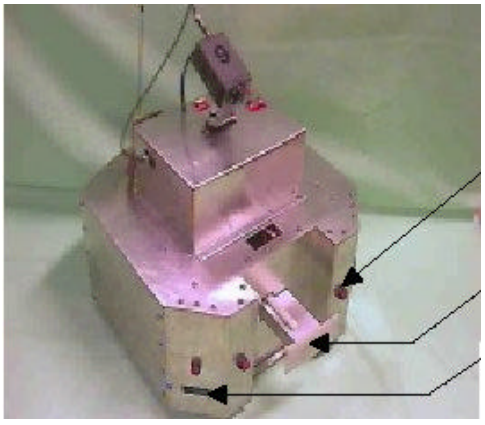


Abbildung 12: Team NP-Lion (voll gepanzert)

3.4 Taktisches Geschick

Fußballspieler menschlicher Art müssen sich schulen und trainieren, um immer besser zu werden, um ihre Techniken und Taktiken zu schulen.
Gleiches gilt auch für die Roboter der Middle-Size-League. Die menschlichen Betreuer des CS-Freiburg scheinen es begriffen zu haben, dass es nicht darauf ankommt, wer die bessere Hardware hat, sondern wer das Beste aus den Robotern herausholt.
Was nützt es wenn ein super schneller Roboter den Ball hat, aber 5 Minuten braucht, um zu „begreifen“ dass er nur noch aufs Tor zustürmen müsste.
Roboter mit „Torschusspanik“ sieht man häufiger auf Turnieren.
Damit solchen Situationen vorgegriffen werden kann besteht die Möglichkeit, von vornherein eine Fülle von möglichen Taktiken im Code hart zu implementieren, sozusagen eine Taktik-Wissensbasis zu erstellen. Für den Anfang mag solch eine Wissensbasis schon mal ein guter Start sein, doch wie im richtigen Fußball, gibt es Situationen, an die der „menschliche Coach“ (der Programmierer) nicht gedacht hat.
Abhilfe schafft hier ein lernfähiges System, welches durch neuronale Netze wohl am ehesten realisiert werden kann. Auch ist es von Vorteil, wenn man eine gewisse Wahrscheinlichkeit für kommende Ereignisse zurate ziehen kann, dies wird unter anderem mit der Fuzzy-Logik realisiert.
Eine große Wissensbank (= z.B. Ergebnisse von Spielauswertungen aller Ligen) mit vorgeschriebenen Reaktionen bei Standard-Situationen - z.B. Ecke, Freistoß oder „Ball aus den Sensoren verloren“ - inklusive einer abschätzenden und lernenden Einheit ermöglicht es, dem Roboter (siehe CS-Freiburg) in ganz manierlicher Art Fußball zu spielen.

3.5 Ausblick auf die Human-League:

Als Ausblick auf die Human League, die in einigen Jahren wohl soweit ausgereift sein könnte, dass sie gegen echte Menschen spielen könnten, mag an dieser Stelle ein Bild reichen.

Abbildung 13:
mögl. Human-League Roboter

Anhang: Funktionen und Methoden:

A.1 Bildverarbeitungs-library: HALCON

In HALCON ist jedes Bild mit einer oder mehreren Bildmatrizen assoziiert. Eine Matrix ist kein direkter Bestandteil der Bildobjekte und kann somit auch von mehreren Bildern gemeinsam verwendet werden. Der Vorteil dieser Vorgehensweise liegt u.a. im geringeren Speicherplatzbedarf und in der hohen Effizienz von Operatoren, die neue Bildobjekte erzeugen, ohne die Bildmatrix zu verändern (z.B. Zugriff auf einzelne Kanäle, Modifikation des Definitionsbereichs, etc.). Der funktionale Charakter der HALCON-Operatoren — die Operatoren haben keine Seiteneffekte, d.h. die Eingangsdaten bleiben i.d.R. unverändert — garantiert dabei, dass einzelne Matrizen nicht durch andere Operationen unbeabsichtigt modifiziert werden.

A.2 Die Hough-Transformation

Die Hough-Transformation ist eine Methode zur Mustererkennung. Sie kann eingesetzt werden, wenn es darum geht, Abbildungen eines vorher bekannten Mustergegenstands in einem Bild (Eingangsbild) zu finden. Die Abbildung eines Mustergegenstands im Eingangsbild kann durch Parameter beschrieben werden. Die Parameter drücken aus, ob der abgebildete Gegenstand beispielsweise in gedrehter oder gekippter Stellung abgebildet ist, wo er sich befindet und wie groß die Abbildung ist.
Mit Hilfe der Hough-Transformation wird das Eingangsbild in einen Parameterraum abgebildet (transformiert), wobei folgendes gilt:

- Der Parameterraum hat genauso viele Dimensionen (z.B. x,y), wie der Mustergegenstand Freiheitsgrade hat.
- Jeder Punkt [z.B. (x;y)] im Parameterraum repräsentiert eine Position des Mustergegenstands.

Der Wert jedes Punktes im Parameterraum gibt Auskunft darüber, mit welcher Wahrscheinlichkeit sich der Gegenstand an der jeweiligen Position befindet.

Anstatt *Parameterraum* wird auch der anschauliche Begriff *Ähnlichkeitsfeld* verwendet. Das Ähnlichkeitsfeld gibt Auskunft darüber, wie ähnlich das Eingangsbild zur Abbildung eines vorher bekannten Mustergegenstands ist, der sich an einer bestimmten Position befindet.

Die Berechnung eines Ähnlichkeitsfeldes erfolgt durch Vergleich von Abbildungen des Mustergegenstands mit dem Eingangsbild. Für jede mögliche Position des Mustergegenstands existiert eine solche Abbildung. Mit Hilfe der Kreuzkorrelation wird die Ähnlichkeit zwischen der Abbildung des Mustergegenstands und dem Eingangsbild bestimmt. Bei diesem Mustervergleich wird mit Hilfe der Kreuzkorrelation jeweils für jede mögliche Position des Mustergegenstands die Ähnlichkeit zwischen der Abbildung des Mustergegenstands und dem Eingangsbild bestimmt. Der Mustervergleich mit Hilfe der Korrelation ist zwar leistungsfähig, erfordert aber sehr viel Rechenleistung.

Die Hough-Transformation ist eine Methode, um den Rechenaufwand für den Mustervergleich zu senken. Bei der Verarbeitung von Bildfolgen beschränkt sich der erforderliche Rechenaufwand dank der Methode der Hough-Transformation auf die Transformation von Bilddifferenzen.

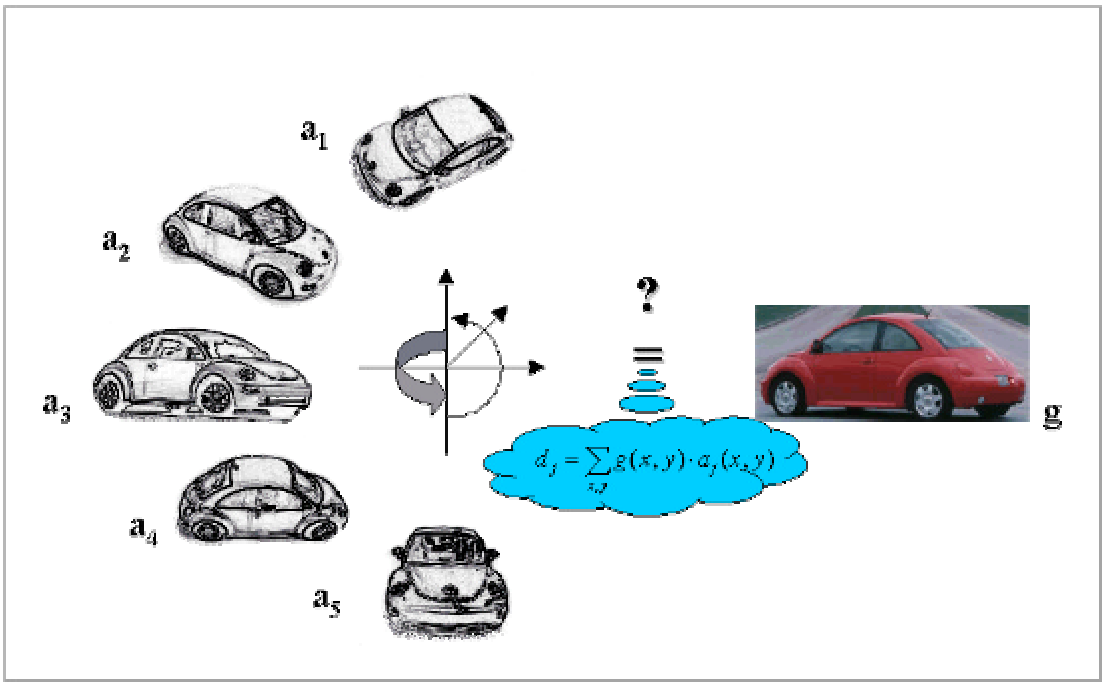


Abbildung 14: Vergleich von Musterbildern a_i des Mustergegenstands mit dem Eingangsbild g

Durch geeignete Auswertung des Parameterraums können Abbildungen von Mustergegenständen im Eingangsbild erkannt werden. Problematisch ist dabei, dass das Ähnlichkeitsfeld je nach Anzahl der Parameter speicherintensiv ist und dass die Suche nach Extremwerten im Ähnlichkeitsfeld aufwändig ist.

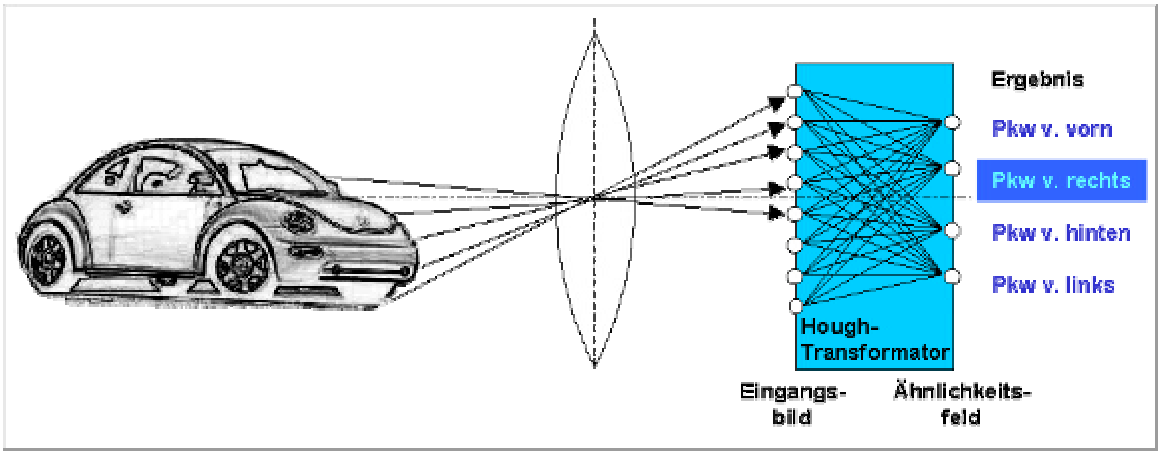


Abbildung 15: Mustererkennung mit der Hough-Transformation

A.3 Rekursiver Kalman-Filter als Geschwindigkeitsschätzer

Eine Erweiterung der einfachen linearen Vorhersageschätzer beruht auf *Kalmans* Theorie der optimalen, linearen und rekursiven Schätzer. Die grundlegenden Gleichungen der Theorie sind:

$$\vec{h}(t+1) = \Phi(t+1,t)\vec{h}(t) + \overrightarrow{s_{sys}}(t)$$

$$\vec{m}(t) = D(t)\vec{h}(t) + \overrightarrow{s_{mess}}$$

mit

$\vec{m}(t)$	Vektor der Observablen zum Zeitraum t
$\vec{h}(t)$	Vektor der System-Zustandsvariablen, Zeitschritt t
D(t)	Messmatrix
$\Phi(t+1,t)$	Systemmatrix
s_{sys}	normalverteiltes Systemrauchen, mit Erwartungswert 0
s_{mess}	normalverteilter Messfehler, mit Erwartungswert 0

Für die Vorhersage der Position von Tracerteilchen im Folgebild sei folgendes Modell angenommen, in welches die Geschwindigkeit und die Beschleunigung eingehen:

$$\begin{pmatrix} \vec{x}(t+1) \\ \triangle\vec{x}(t+1) \\ \triangle\vec{v}(t+1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \vec{x}(t) \\ \triangle\vec{x}(t) \\ \triangle\vec{v}(t) \end{pmatrix} + \overrightarrow{s_{sys}}$$

$$\begin{pmatrix} \vec{x}(t) \\ \triangle\vec{x}(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \vec{x}(t) \\ \triangle\vec{x}(t) \\ \triangle\vec{v}(t) \end{pmatrix} + \overrightarrow{s_{\exp}}$$

wobei

$\vec{x}(t), \triangle\vec{x}(t)$, sowie $\triangle\vec{v}(t)$ die erwartete, wahre Position, Geschwindigkeit und Beschleunigung zum Zeitpunkt t bedeuten

$x(t), \triangle x(t)$ die gemessenen Positionen bzw. Geschwindigkeiten zur Zeit t sind.

Ein auf dieser Theorie beruhender Extrapolationsfilter ist zur Zeit in der Entwicklung und soll in Zukunft die Positionsvorhersage in turbulenten Strömungen erleichtern.

A.4 Markov Lokalisation / Bayes-Filter

Lokalisation stellt eine fundamentale Voraussetzung für zuverlässige Navigation und Operation mobiler autonomer Roboter dar. Das Markov-Verfahren zur Lokalisation verwaltet die Schätzung der aktuellen Roboterposition als Wahrscheinlichkeitsdichte über den möglichen Koordinaten des Weltmodells, welches meist durch Wahrscheinlichkeitsraster approximiert wird.

Mit $x = (u, v, q)$ sei eine dreidimensionale Positionsangabe bezeichnet, welche aus den kartesischen Koordinaten und der Orientierung des Roboters gebildet ist. Weiterhin ist mit x_t die tatsächliche Position zum Zeitpunkt t gemeint. Die Grundidee eines Bayes-Filters besteht darin, eine a posteriori Wahrscheinlichkeitsdichte zu schätzen, welche durch alle früheren Messungen d_i bedingt ist, mit

$$Bel(x_t) = (x_t \mid d_{0..t})$$

bezeichnet wird und sich über den gesamten Zustandsraum erstreckt. Beispielsweise wird eine bekannte initiale Position durch eine Dirac-Verteilung, deren Maximum auf dieser

Position liegt, abgebildet, ohne dass ein Vorwissen über eine uniforme Verteilung des gesamten Zustandsraums benötigt wird. Dabei steht $d_{0..t}$ für die Eingabedaten vom Zeitpunkt 0 bis t, welche in zwei Typen unterschieden werden: a (action) und Umgebungsmessungen o (observation). Ohne Einschränkung kann eine alternierende Abfolge festgelegt werden, wo raus sich ergibt:

$$Bel(x_t) = (x_t \mid o_t, a_{t-1}, o_{t-1}, a_{t-2}, ..., a_0, o_0) \tag{1}$$

Unter Anwendung der Bayes'schen Formeln ergibt sich aus (1):

$$Bel(x_t) = \frac{p(o_t \mid x_t, a_{t-1}, ..., o_0) p(x_t \mid a_{t-1}, ..., o_0)}{p(o_t \mid a_{t-1}, ..., o_0)} = \frac{p(o_t \mid x_t, a_{t-1}, ..., o_0)\, p(x_t \mid a_{t-1}, ..., o_0)}{p(o_t \mid a_{t-1}, ..., d_{0..t-1})} \tag{2}$$

Zur Reduzierung der Berechnungskomplexität unterliegt allen Anwendungen des Markov-Filters die Annahme, dass zukünftige Messungen unabhängig von früheren sind (und umgekehrt), vorausgesetzt, die aktuelle Position des Roboters ist bekannt, d.h., dass die Umwelt statisch ist:

$$p(d_{t+1} \mid x_t, d_0, ..., d_t) = p(d_{t+1} \mid x_t) \forall t \tag{3}$$

Das bedeutet, dass der komplette Zustand der Welt mit der Position des Roboters beschrieben ist und ausreicht, um Daten vorauszuberechnen, und wird Markov-Bedingung genannt. Dies ist eine stark einschränkende Annahme, die sofort verletzt wird, sobald sich andere bewegende Objekte in der Szene befinden. Auch wenn in praktischen Ergebnissen eine gewisse Robustheit gegenüber geringer Dynamik festgestellt wurde, kann man Markov-Filter nicht sicher in dynamischen Umgebungen einsetzen.
Unter dieser Annahme vereinfacht sich Formel (2) zu:

$$Bel(x_t) = \frac{p(o_t \mid x_t) p(x_t \mid a_{t-1}, ..., o_0)}{p(o_t \mid a_{t-1}, ..., d_{0..t-1})}$$

Nun muss über x_{t-1} integriert werden und man erhält:

$$Bel(x_t) = \frac{p(o_t \mid x_t)}{p(o_t \mid a_{t-1},..., d_{0..t-1})} \int p(x_t \mid x_{t-1}, a_{t-1},..., o_0)\, p(x_{t-1} \mid a_{t-1},..., o_0)\, dx_{t-1}$$

Weiterhin impliziert die Markov-Bedingung, dass bei gegebenen x_{t-1} und a_{t-1} der Zustand x_t unabhängig von früheren Umgebungsmessungen $o_1,..., o_{t-1}$ und

Odometrie[3]-Sensordaten $a_0,..., a_{t-2}$ bis zum Zeitpunkt t-2 ist:

$$p(x_t \mid x_{t-1}, a_{t-1},..., o_0) = p(x_t \mid x_{t-1}, a_{t-1})$$

Letztendlich ergibt sich daraus die gesuchte rekursive Formel, welche Bayes-Filter genannt wird und worauf der Algorithmus der Markov-Lokalisation aufbaut:

$$\begin{aligned} Bel(x_t) &= \frac{p(o_t \mid x_t)}{p(o_t \mid a_{t-1},..., d_{0..t-1})} \int p(x_t \mid x_{t-1}, a_{t-1})\, Bel(x_{t-1})\, dx_{t-1} \\ &= h\, p(o_1 \mid x_1) \int p(x_t \mid x_{t-1}, a_{t-1})\, Bel(x_{t-1})\, dx_{t-1} \end{aligned} \qquad (4)$$

Hierbei bezeichnet h eine Normalisierungskonstante.
Formel (4) gleicht der grundlegenden Aktualisierungsvorschrift von Kalman-Filtern.

Aus (4) ist ersichtlich, dass die folgenden Modelle zur Realisierung der Markov-Lokalisation benötigt werden:

- die initiale Positionsschätzung $Bel(x_0)$,
- die Übergangswahrscheinlichkeiten $p(x_t \mid x_{t-1}, a_{t-1})$, welche als Odometriemodell bezeichnet werden,
- die Wahrscheinlichkeiten der Wahrnehmung einer Beobachtung der Umwelt an einer bestimmten Position $p(o_t \mid x_t)$, welche das Sensormodell der umweltmessenden Sensoren darstellen

[3] Odometrie: Die Winkelmesswerte werden über den CAN-Bus an den Steuerrechner gesendet, der die Encoder-Daten auswertet und eine Schätzung der gefahrenen Wegstrecke ermittelt.

Quellennachweis:

2001 Team Description - Middle Size League António Paulo Moreira_F2000.pdf (von: www.fe.up.pt/~robosoc/sci/)

ar.pdf (von: http://www.informatik.uni-ulm.de/ki/Edu/Proseminare/Intelligente.Systeme/SS01/#robocup)

robocup.pdf (von: ais.gmd.de/dfg-robocup)

thesis.ps.gz (von: www.uni-koblenz.de/~fruit/PAPERS)

klupsch.pdf (von : tumb1.biblio.tu-muenchen.de/publ/diss/in/2000)

studienarbeit.pdf (von: www-ti.informatik.uni-tuebingen.de/~diebold/papers)

robo.pdf (von: www-user.tu-chemnitz.de/~stj/lehre)

Factbook_Robocup.pdf (von: ais.gmd.de/de/docs)

pio1ref.pdf (von: www.informatik.fh-hamburg.de/~pioneer/Downloads)

http://www.activrobots.com/HISTORY/index.html#erratic

http://vini25.vkw.tu-dresden.de/vinije/leich/diplom/Seite1.html

http://www.robot-news.com/robocup_2001_images/avi/

http://www.coral.cs.cmu.edu/aaairobot/location/venue.avi

http://www.robosoft.fr/SHEET/02Local/1004SickLMS200/SickLMS200.html

http://www.neurocheck.com/products/news/product_news_g.html

http://klimt.iwr.uni-heidelberg.de/Projects/paper/doktor/node64.html

http://www.inf.tu-dresden.de/~mf14/mcl-report.ps.gz

http://www.fzi.de/mmr/german/projects/robot/details/Motors.html

http://www.smallrobots.de/ws2000/labyrinth/refmechanik/

http://www.cs-freiburg.de/

Benutzte Team-Descriptions:
http://www.informatik.uni-freiburg.de/~gutmann/midleague-2000/midleage-team-descriptions-2000.html

Team name	Team Address/Organisation	Team city	Team country
Golem Team	University of Padua	Padua	Italy
RMIT United	124 La Trobe Street	Melbourne, VIC	Australia
GMD Robots	Schloss Birlinghoven	Sankt Augustin	Germany
CS Freiburg	University of Freiburg	Freiburg	Germany
CoPS Stuttgart	Breitwiesenstr 20-22	Stuttgart	Germany
Munich Agilo RoboCuppers	Orleansstr. 34 Room O 241	Munich	Germany
The Ulm Sparrows	University of Ulm	Ulm	Germany
Sharif CE	Department of Computer Engineering, Sharif University of Tech. Azadi Ave	Tehran	Iran
ART2000	National team		Italy
WinKIT	RoboCup Project Factory for Dreams and Ideas Kanazawa Institute of Technology	Nonoichi, Ishikawa	Japan
NAIST2000	8916-5, Takayama-cho	Ikoma, Nara	Japan
EIGEN	University of Keio	Yokohama City	Japan
KIRC	Kawazu 680-4	Iizuka, Fukuoka	Japan
Trackies	Osaka University, Yamadaoka 2-1	Suita, Osaka	Japan
The Dutch Team	Lorenzweg 1	Delft	Netherlands
IsocRob	ISR/IST- Av, Rovisco Pais, 1	Lisboa	Portugal
Minho	DEI - Campus de Azurem	Guimaraes	Portugal
5DPO-F2000	FEUP-Rua dos Bragas	Porto	Portugal
NP Lions	Ngee Ann Polytech	Singa-pore	Singapore
Vanquish	Singapore Polytechnic, 500 Dover Road	Singa-pore	Singapore
RFC Uppsala	DoCS, Box 325	Uppsala	Sweden
CMU Hammerheads	Robotic Institute, Carnegie Mellon University	Pittburgh, Pennsyl-vania	USA
The Scarabs	3131 Olypic Blvd	Santa Monica, CA	USA
Paradise	Tehran, Evin	Tehran	Iran