Deliberative Ebene

- Weltmodell (cartographer)

- Planer mit 3 Schichten: mission planer, navigator, pilot (vgl. NHC-Architektur)

- Überwachung und Kontrolle des “Gesundheitszustands” (homeostatic control)

Reaktive Ebene

- Sensor-Vorverarbeitung (perception) mit Schnittstelle zum Weltmodell

- Verhalten (Motorschemata), die Aktoren kontrollieren

- Manager zur Kontrolle der Verhalten, besitzt Schnittstelle zur deliberativen Ebene

(Planer, homeostatische Kontrolle)

Deliberatives (hierarchisches) Paradigma

- erfordert Plangenerierung auf einem globalen Weltmodell

- Probleme: closed-world assumption (Weltmodell muss alle Infos bereitstellen)

und frame problem (wird schnell umfangreich)

- daher nur anwendbar, wenn Weltmodell klar definierbar und nicht zu komplex

Reaktives Paradigma

- erlaubt sofortige Reaktion auf vorliegende Sensoreingaben

- legt kein Weltmodell zu Grunde (open-world assumption)

- Problem: nur für einfache Aufgaben anwendbar

Hybrides Paradigma

- kombiniert reaktives Verhalten auf unterer Ebene mit Planung auf höherer Ebene

1.Verhalten entspricht einem Regler,wobei die Regelstrecke die Umwelt ist

2.Ballistisch weil vorhersagbar / determiniertwie “freier Fall”

entspricht Ablaufsteuerungen mitZuständen (states) und Transitionen,

die durch Sensorwerte ausgelöst warden

3.Roboter soll “heim“ zu einer Lichtquelle fahren (Homing)

außer der genauen eigenen Position (XR, YR) braucht der Roboter auch einen Kompass, um im richtigen Winkel q auf das Ziel (XG, YG) zufahren zu können

Roboter glaubt immer wieder in einer anderen Zelle zu sein und reagiert entsprechend hektisch

mögliche Lösung: genauere Sensoren oder Umschalten auf anderes Homing-Verfahren in Zielnähe

4.weicht Hindernissen aus, um Kollisionen zu vermeiden

wird ein Hindernis links erkannt

dreht sich der Roboter mit Geschwindigkeit w nach rechts (-w) (Drehung ist beendet sobald L=0)

wird ein Hindernis rechts erkannt

dreht sich der Roboter mit Geschwindigkeit w nach links (+w) (Drehung ist beendet sobald R=0)

wird kein Hindernis erkannt,

fährt der Roboter mit Geschwindigkeit v in die aktuelle Richtung

Problem mit einfachem Avoid-Verhalten kann durch entgegengesetzte Reaktionen in ähnlichen Situationen

oszillieren und sich dadurch festfahren (Canyoning)

Oszillieren und Festfahren wird durch konsistente Drehung in eine Richtung

Roboter dreht sich evtl. sehr weit, fährt anschließend fast in Gegenrichtung,

wenn Timer läuft, Drehung in genau eine Richtung

immer noch Signal auf Sensor? Sensoren und Timer inaktiv?

ist T geeignet gewählt, dreht der Roboter weit

genug, um Oszillationen zu vermeiden

Oszillierendes Verhalten (Thrashing)

kann nicht nur durch gegensätzliche Aktionen innerhalb eines Verhaltens ausgelöst

werden, sondern auch durch mehrere gegensätzliche Verhalten

Gegenmaßnahmen- Zyklusdetektor- Randomisierung- neues Verhalten, speziell für diese Situation

Fluchtverhalten, wenn Roboter gegen ein Hindernis stößt, d.h. er “schreckt“ nach hinten zurück und dreht sich dann weg (ballistisches Verhalten, siehe 2-22); oft Auslösung durch mechanische Bumper

Avoid vermeidet dagegen eine Kollision, indem es kurz vorher abdreht;

Voraussetzung: Näherungssensoren, die schon in einigem Abstand vom Hindernis ansprechen (Infrarot oder Ultraschall), in der Regel Servo-Verhalten

- Roboter soll eine Begrenzung (confinement) nicht verlassen, z.B. Infrarotstrahl (virtuelle Wand für Staubsauger), Linie am Boden, Draht im Boden (Rasenmäher).

- geringe Überschreitung der Grenzlinie mit Rädern erlaubt (ein Sensor vorne reicht).

- Verhalten analog zu AVOID (vom “Hindernis” wegdrehen)

- Ziel vollständiges Abdecken einer begrenzten Fläche

- deterministische Strategien erfordern genaue Kenntnis der Position des Roboters, was in der Praxis evtl. schwierig ist

- zufällige Strategien einfach implementierbar, aber langwierig, gerade Linien mit zufälliger Drehung an der Begrenzung

Problem Roboter findet Tür nur zufällig und bleibt hauptsächlich im vorderen Raum (insbesondere, wenn sich dort seine Docking-Station zum Laden der Batterien befindet)

Lösung anstatt sich immer nur zufällig von der Wand wegzudrehen, folgt der Roboter eine Zeit lang der Wand und findet so eher durch Türen in andere Räume

um komplexe Aufgaben (Tasks) zu lösen,verschiedene parallel agierende primitive Verhalten kombinieren als zusammengesetztes abstraktes Gesamtverhalten, Wechselwirkung zwischen den Elementarverhalten lässt Gesamtverhalten mehr als die Summe seiner Einzelverhalten warden

der verhaltensbasierte Programmierung von Robotern nach dem S-A Paradigma begründet hat inspiriert durch das Verhalten bei Tieren, das eher eine horizontale Struktur aufweist und keine vertikale wie Sense-Plan-Act

(1) Module werden in Kompetenzebenen angeordnet.

(2) Module höherer Ebenen können die Ausgaben der nächst tieferen Ebene überschreiben

(3) interne Zustände werden vermieden, d.h. kein explizites Weltmodell

(4) eine Aufgabe (Task) wird durch ktivierung der jeweiligen Schichten implementiert

S: Suppressor-Knoten, d.h. höherpriores erhalten (oben) unterdrückt niederpriores (links) und

überschreibt dessen Ausgabe mit er eigenen

I: Inhibitor-Knoten, d.h. höherpriores erhalten verhindert Ausgabe des niederprioren und setzt sie auf Null

Grundverhalten auf der tiefsten Ebene der “Evolution”:

- wenn kein Hindernis vorhanden ist, bleibt der Roboter einfach stehen

- Weglaufen (RUN AWAY) bei Hindernissen durch Drehen und Fahren

- bei Hindernis direkt vor dem Roboter (COLLIDE) Vorwärtsfahrt anhalten und nur wegdrehen

WANDER berechnet alle n Sekunden eine zufällige Richtung (Vektor heading),

AVOID addiert heading mit dem repulsiven Kraftvektor force zu modified heading

Suppressor S ersetzt die Ausgabe von RUNAWAY durch modified heading.

Roboter fährt zufällig herum und weicht dabei Hindernissen aus,

bei Kollisionen hält er nach wie vor die Vorwärtsfahrt an und dreht nur

STAYIN MIDDLE berechnet aus corridor die Richtung zur Mitte heading to middle, falls corridor nicht vorliegt (z.B. Sonarfehler, Rechenzeit) wird heading to middle mit Hilfe des Vektors distance, direction traveled ermittelt unterdrückt WANDER mit S-Knoten Roboter fährt in der Mitte des Korridors, weicht dabei Hindernissen aus und stoppt bei Kollisionen seine Vorwärtsfahrt

Primitive Verhalten können auch durch Potentialfelder beschrieben werden, die (virtuelle) Kräfte erzeugen, dargestellt durch Vektoren (vgl. z.B. elektrische Felder) Roboter berechnet repulsive

Kraft mit Hilfe seiner Sensoren und bewegt sich in Richtung des Kraftvektors vom Hindernis weg

Eigenschaften des Komplexere Verhalten

- keine Kenntnis des globalen Feldes erforderlich, Roboter wertet in jedem Schritt nur seine Sensoren lokal aus

- sanfter Übergang von einem Verhalten zum anderen

komplexere Aufgaben erfordern teilweise, dass Sequenzen primitiver Verhalten nacheinander ausgeführt werden müssen, d.h. nicht alle primitiven Verhalten sind immer gleichzeitig aktiv

Nachteil: gleichzeitig mitlaufende Verhalten (z.B. Avoid) nicht darstellbar, da klassisches Automatenmodell keine Nebenläufigkeit zulässt; entsprechende Erweiterungen aber bekannt (siehe z.B. Vorlesung Echtzeitsysteme)

- Roboter soll eine Begrenzung (confinement) nicht verlassen, z.B. Infrarotstrahl Linie am Boden, Draht im Boden (Rasenmäher).

- geringe Überschreitung der Grenzlinie mit Rädern erlaubt

- verhindert, dass Roboter über Kanten (Cliffs) fährt und abstürzt (Stufen, Tisch etc.)

- Sensoren, die nach unten gerichtet sind, um Kante zu erkennen (z.B. IR).

Anordnung so, dass Kante erkannt wird, bevor Räder über die Kante fahren

-

Kollisionssensoren:

sprechensprechen erst an, wenn der Roboter bereits mit einem Hindernis kollidiert ist

zu sensible Einstellung: Ansprechen beim Anfahren oder Bremsen durch Schwingungen der Schürze (“false positive”)

je nach Anbringung der Schürze: zu niedrige oder zu hohe, vorspringende Hindernisse unerkannt (“false negative”)

Motorstrommessung:

bei blockierten Rädern steigt der Motorstrom an, was zur Erkennung von Kollisionen verwendet werden kann

Probleme

- auch beim Anfahren zunächst hoher Strom, der dann wieder abnimmt

- je nach Boden und Haftung der Räder schwankender Strom hoher Strom muss mindestens

eine Zeit td vorliegen, sonst Gefahr von „false positives”

Vorteil: einfach zu implementieren

Kollisionsvermeidungssensoren:

Sprechen bereits in gewisser Entfernung zum Hindernis an,

so dass der Roboter dem Hindernis noch ausweichen kann, ohne es zu berühren

Infrarot-Näherungssensoren

Vorsicht

dunkle und schräg gestellte Objekte werden oft nicht erkannt, da nicht genügend Reflektion (false negative)

Dioden haben nur begrenzte Erkennungsbereiche (Distanz und Breite); außerhalb liegende Objekte führen zu false negatives Intensität sagt wegen unterschiedlicher Reflektionsverhalten der Objekte nur bedingt etwas über Entfernung aus

=> nicht als Distanzsensor geeignet

Infrarot-Entfernungssensoren:

Vorsicht

starkes Umgebungslicht kann bei allen IR-Sensoren zum “Erblinden” der Detektoren führen

false positives bei IR-Sensoren dagegen sehr selten

Ultraschall-Distanzsensoren:

während des Sendens und kurz danach wird die Empfindlichkeit des Empfängers

(threshold) auf das Minimum gesetzt (d.h. geringe Empfindlichkeit), damit das Senden den Empfänger nicht stört (blanking interval)

- Laufzeitmessung (time of flight, TOF) eines reflektierten

Laserstrahls (meist IR-Laser)

- gesendeter und empfangener Strahl sind koaxial

- Empfänger misst die Roundtrip-Zeit (hin und zurück)

- Mechanik mit beweglichem Spiegel für 2D- oder 3D Messungen

wesentlich höhere Genauigkeit als Ultraschall gute Reflektions- und Absorptionseigenschaften (nur Glas/Spiegel problematisch).

aber

relativ groß, schwer und teuer, daher eher nur für größere Roboter geeignet; neuere Modelle bereits kleiner, leichter, günstige

Gyroskop

Richtungssensor, der die Orientierung in Bezug auf einen festen Rahmen ermittelt;

damit Orientierung eines mobilen Roboters im Weltkoordinatensystem ermittelbar  
Beschleunigungssensor

misst externe Kräfte, die auf ein System einwirken.b basiert meist auf einem Feder-Masse-

Dämpfersystem, Position einer Referenzmasse Messen

Gyroskope und Accelerometer können zu einer sog. IMU (Inertial Measurement Unit)

kombiniert werden, die eine Trägheitsnavigation erlaubt, d.h. ausgehend von bekannter Ausgangspose die aktuelle Pose (Position und Orientierung im 3D Raum) ermittelt; teilweise kombiniert mit Kompass; allerdings sehr fehleranfällig und vor allem bei längerem Einsatz ungenau

Standard-Rad

2 Freiheitsgrade:

- Rotation um die Radachse (Antrieb)

- Rotation um Kontaktpunkt (Lenkung)

Castor-Rad

3 Freiheitsgrade:

- Rotation um die Radachse (Antrieb)

- Rotation um Kontaktpunkt (“Lenkung”)

- Rotation um die Castor-Achse

Allseiten-Rad (Swedish Wheel)

3 Freiheitsgrade:

- Rotation um die Radachse (Antrieb)

- Rotation um Kontaktpunkt (Lenkung)

- Rotation um die Roller-Achse

sphärisches Rad

(Kugel-Rad)

3 Freiheitsgrade

technisch schwierig

realisierbar

im Gegensatz zum Standard-Rad kann ein Roboter mit

Castor-Rädern auch zur Seite fahren,

d.h. mit Castor-Rädern sind holonome Antriebe möglich

(siehe Räder an Servierwägen

oder Bürostühlen)

Motion Control

es können aber prinzipiell alle Zielposen (x, y, q) entlang geeigneter Bahnen (x, y)

von einer beliebigen Startpose aus angefahren werden

Problem (motion control):welche Bahn bzw. Trajektorie soll gefahren werden und wie findet man

Strategien:

- reine Steuerung (open loop control)

- Regelung (feedback control)

Open Loop Control (Steuerung)

meist Zusammensetzung der Bahn aus Geraden und Kreisbahnen,d.h. die einzelnen Segmente werden mit konstanten Geschwindigkeiten der Räder gefahren

Feedback Control (Regelung)

aktuelle Abweichung des Roboters von der Zielpose wird als Regelabweichung e

interpretiert, hier zur Vereinfachung e = (x, y, q), d.h. gleich der aktuellen Pose des Roboters

Dead-Reckoning (Koppeln)

theoretisch kann mit kinematischen Gleichungen die aktuelle Pose von einer beliebigen

Ausgangspose aus exakt berechnet werden, wenn man die Geschwindigkeiten v1(t) und v2(t)

kennt, wobei v1(t) bzw. v2(t) mittels Wegstreckenmessung (Odometrie) z.B. durch

Radenkoder und gefahrener Zeit ermittelbar sind (vgl. Industrieroboter);

es wäre dann immer genau bekannt,

in welcher Pose der Roboter sich gerade befindet (Selbstlokalisation)

da durch den Prozessor des Roboters die Auswertung der Radenkoder in diskreten

Zeitabständen erfolgt, ist eine Diskretisierung erforderlich

Odometrie immer ungenau, z.B. durch

- Unebenheiten im Boden

- Durchdrehen der Räder

- Asymmetrien des Antriebs

(z.B. nicht exakt gleich große Räder)

Q-Learning [Watkins 89]

populäres Reinforcement-Learning-Verfahren für mobile Roboter

Ziel ：der Roboter soll ein Kontrollverfahren, z.B. ein primitives reaktives Verhalten lernen

gegeben

- Menge von diskreten Zuständen s ∈ S (entsprechen hier den Sensorwerten)

- Menge von diskreten Aktionen a ∈ A (entsprechen hier den Aktorwerten)

Gesucht ：möglichst leistungsfähiges Verhalten beh: S → A

VorNach

+ einfache Implementierung

+ akzeptables Lernverhalten für einfache Roboterverhalten

experimentell nachgewiesen

+ Kombination mit anderen Verfahren wie z.B. neuronalen Netzen möglich

- langsames Lernen, d.h. langsame Konvergenz

- durch Trial-and-Error kann der Roboter in für ihn gefährliche Situationen geraten

- bei komplexeren Aufgabenstellungen kombinatorische Explosion des

Eingaberaums und damit der Größe der Q-Tabelle

- Reward-Werte nicht immer leicht zu definieren und ggf. Ungenau

Markov Decision Process (MDP)

dt. Markow-Entscheidungsproblem

Modell für Agenten dessen Nutzen von Folgen von Entscheidungen abhängt.

Zustand s, Folgezustand s’, Aktion a

Fuzzifizierung

Zuordnung von linguistischen Variablenzu Führungsgröße w und Regelgröße xmittels Zugehörigkeitsfunktionen

Inferenzglied und Regelbasis unscharfes Schließen (Fuzzy Inferenz)mittels WENN-DANN-Regeln

Defuzzifizierungsglied Bilden einer scharfen Ausgangsgröße

(Regelgröße y) aus den unscharfen

Ausgangsgrößen der Regelbasis

BCU-Regelbasis wird vom Entwickler vorgegeben und implementiert das

gewünschte Verhalten bereits grob

OCU verfeinert das Wissen, indem sie die Abweichung vom optimalen Verhalten

ermittelt und die Konklusion der BCU-Regeln entsprechend

modifiziert, indem sie den angegebenen Wert von m aufaddiert

um unsichere Aktionen zu vermeiden, sind einige Regeln von der Modifikation

ausgeschlossen

BCU lernt damit in sicheren Grenzen das optimale Verhalten und passt es auch bei

sich ändernder Umwelt ständig neu an (Online-Lernen)

Problem: mit steigender Anzahl von Eingangsvariablen ‚Explosion‘ der Größe der

Regelbasis

mögliche Gegenmaßnahme: Zerlegung in kleinere Regelbasen, die in mehreren

Stufen angeordnet werden

auch Kombination mit scharfen (crisp) Regeln möglich, z.B. für Boolesche

Funktionen,

KKN

KNN sind parallele Rechensysteme,

da Signale unabhängig voneinander

durch gerichtete Kanäle wandern;

werden meistens aber auf seriellen Computern simuliert

künstliche Neuronen sind durch

gerichtete Kanten verbunden (vgl. Synapsen), um Signale zu übertragen

Eingangsebene erhält Signale aus der Umwelt per Sensoren,

Ausgangsebene schickt Signale in die Umwelt per Aktoren

Gewichtete Summe

Die Eingangssignale qi werden zunächst gewichtetaddiert. Die Gewichte wi beschreiben die

Stärke der exzitatorischen (wi > 0) bzw.inhibitorischen (wi < 0) Wirkung des betreffenden Eingangs

Die Gewichte werden i. Allg. in einerTrainingsphase gelernt (überwachtes Lernen)und bestimmen dann das Verhalten des neuronalen Netzes.

Aktivierungsfunktion F

bestimmt die innere Aktivität A(t) des Neurons aufgrund der Eingangserregung S(t)

und evtl. vorangehenden Aktivität A(t - 1).

Aktivierungsfunktion: Stufenfunktion in Abhängigkeit eines Schwellwertes θj

Ausgangsfunktion f

gibt an, wie stark die innere Aktivität A(t) nach außen als a(t) (vgl. Axon) weitergeleitet

wird (meist Schwellwertfunktionen oder Identität).

Die Transferfunktion ist typischerweise nur eine Summation,

ihr Ergebnis ist der net input

net input: Summe aller eingehenden Signale xi gewichtet durch wij

Lernphase (Trainingsphase, offline)

nach Anlegen von Eingangsmustern werden die Gewichte so verändert, bis sich das gewünschte

Ausgangsverhalten einstellt (supervised learning) Lernphase endet, wenn das gewünschte Verhalten

nach geeigneten Gütekriterien erreicht ist

Auswertungsphase (operativ/online)

das neuronale Netz reagiert wie gelernt auf angelegte Eingangsmuster; es ist dabei im Allg. nicht nur in der Lage zu reproduzieren, sondern auch auf ähnliche, nicht explizit gelernte Muster richtig zu reagieren,

mehrere oder viele Roboter imgleichzeitigen Einsatz statt nur eineralle arbeiten zusammen, gemeinsaman einer Aufgabe

Eigenschaften der Schwarmrobotik

(1) dezentrale Steuerung

(2) keine Synchronisierung

(3) einfache (quasi-)identische Einheiten / (quasi-)homogen

(4) massenproduziert

3 Hauptvorteile

(1) Robustheit: redundantes System, kein single-point-of-failure, der Verlust eines

gewissen Teils des Schwarms hat keinen großen Einfluss auf die Effektivität

(2) Flexibilität: identische Teammitglieder, keine spezialisierte Hardware

(3) Skalierbarkeit: der gleiche Algorithmus ist anwendbar auf ganz verschiedene

Schwarmgrößen, die Effizienz pro Roboter sinkt nicht maßgeblich mit steigender

Schwarmgröße

superlinear heißt z.B. “doppelte CPU Anzahl bringt dreifache Leistung”

somit leistet auch jeder einzelne mehr, wenn das Team größer wird (!)

Computersystem: Cache-Effekte, z.B. mehr CPUs bedeutet kleinere Workpackages

und ab bestimmter Größe passt das Workpackage gerade in den Cache

Roboterschwarm: Teameffekt, z.B. gegenseitiges Helfen über Loch oder

komplexere Effekte - Wartezeiten, Schwarmgröße und Interaktionen

KI-Paradigamm : 1.SPA 2.SA 3.P-SA

Primitives Verhalten :

1.Servo 2.Ballistische Verhalten 3.Homing-Verhalten

4.Avoid-Verhalten 5.Oszillierends Verhalten 6.Escape-Verhalten

Wall-Follow : 1.Confinement 2.Cliff Avoidance 3.Area Coverage

Abstrakte Verhalten :

1.Subsumption Architecture 2.Motor-Schema Architektur 3.Sequenzen primitiver Verhalten

Sensor: 1.Kollisionssensoren 2.Kollisionsvermeidungssensoren 3.Homing-Sensoren

4.Positions und Navigationssensoren 5.Laserscanner

Kollisionsvermeidungssensoren:

1.Infrarot-Näherungssensoren 2.Infrarot-Entfernungssensoren

3.Ultraschall-Distanzsensoren 4. Laser range finder light detection and ranging(LIDAR)

Homing-Sensoren:1.IR-Beacon 2.Lichtsensoren

Positions und Navigationssensoren:1.Gyroskop 2.Rad-Encoder 3.Beschleunigungssensor

Roboter-Sehesystem:

1. CCD Kameras 2.CMOS Kameras 3.Grauwertbilder und Farbbilder

4.RGB(Red Green Blue) 5.HSV(Hue Saturation Intensity Value)

5.Robotersehen versus Computersehen 6.Algorethmen Robotersehen 7.Color Blob Tracking 8.Lichtstreifen 9.RGB-D Vision Microsoft Kinect

Motor:1.Gleichstrommotoren 2.Bürstenloser Gleichstrommotor 3.Schrittmotoren 4.Servos