Erwerb und Koordination von Verhalten für visuomotorische Systeme in realen Umgebungen mittels Reinforcement Lernen

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)
vorgelegt der
Fakultät für Informatik und Automatisierung
der
Technischen Universität Ilmenau

von
Dimitrij Surmeli

Ilmenau
2002

Danksagung

Allein hätte ich das vorliegende Heft und die darin beschriebene Arbeit nie fertiggebracht. Deshalb gebührt vielen Dank, Menschen und darüber hinaus.

Ohne meinen Betreuer, Prof. Dr.-Ing. Horst-Michael Gross, wäre ich nicht wieder nach Ilmenau gekommen, aber besonders dankbar bin ich für die Ideen und richtungweisenden Fragen und Hinweise, ohne die ich immer noch in einem Wust interessanter Probleme verloren wäre. Seine Geduld, besonders auch beim Warten darauf, daß sich ein Knoten in meinem Kopf löse und seine Fähigkeit, den strategischen Überblick zu behalten und trotzdem auch Parameter zu diskutieren, waren notwendige Voraussetzungen. Am Ende beförderte auch das Warten für Nichtraucher auf die raren 'Chefdates' eigenverantwortliche Lösungsansätze. Unterschiede in unseren Arbeitsstilen wurden mehr als wettgemacht durch das Kollektiv, das er zusammenstellen und in freundschaftlich-kollegialer Atmosphäre zusammenhalten konnte.

Von meinen Kollegen möchte ich mich besonders bei Dr. Volker Stephan und (noch) Dipl.-Inf. Andrea Heinze bedanken für die stets offene Tür und fortgesetzte offene Diskussionen und Hinweise, die vieles anregten und sehr vieles sehr viel einfacher machten, und für mehrmalige Erfahrungen mit biologischer Wetware. Dipl.-Inf. Carsten Schauer verhalf mir zu einer neuen Sicht auf grafische Darstellungen.

Dr. Hans-Joachim Böhme und Dr. Klaus Debes sorgten neben Fachdiskussionen für den notwendigem studentischen und finanziellen Nachschub, ohne den sehr wenig funktioniert hätte, während Dipl.-Ing. Heike Gross und Dipl.-Ing. Sabine Schulz mit nahezu magischen Reaktionszeiten existentielle Rechnerprobleme lösten. Bei der guten Seele des Fachgebiets, Ute Schütz, möchte ich mich herzlich bedanken für verschiedenste Nettigkeiten.

Meine Eltern, Dr. Dimitrij Surmeli und Dr. Ilse Surmeli sowie meine Freundin und Partnerin Wiebke Rohde erst an dieser Stelle bedenken zu können, ist ungerecht und schmerzhaft. Ohne ihre unbegrenzte Geduld und Unterstützung wäre ich ein Blatt im Wind. Und Willi: die Stelle an meinen hinteren Teilen verträgt jetzt ein wenig Pflege in unserem Garten.

Besonders bedacht werden soll hier Luca: ohne dich hätte das alles nicht stattgefunden!

Zusammenfassung

In der vorliegenden Arbeit werden Algorithmen des Reinforcement Lernens benutzt, um Verhalten zu erlernen und zu koordinieren. Der Schwerpunkt liegt dabei auf einer Umsetzung einer adaptiven Koordination, dem W-Lernen, mit einem Miniroboter Khepera in einer realen, kontrollierten Umwelt.

Nach der Einführung des konzeptionelle Rahmens, der Aufgaben für autonome intelligente mobile Systeme, und der grundlegenden Begriffe Verhalten, Agent und Multiagentensystem besteht der Kern der Arbeit in Untersuchungen zum lernenden Erwerb von Verhalten und Koordination. Die Möglichkeit einer Umsetzung von Reinforcement Lernen und einer adaptiven Koordination auf dem realen Roboter wird demonstriert. Die Ergebnisse der Koordination durch W-Lernen werden denen eines mit derselben Gesamtaufgabe betrauten monolithischen Agenten gegenübergestellt. Konzeptionelle Nachteile des W-Lernen werden diskutiert und in den Ergebnissen gezeigt. Erweiterungen zum W-Lernen werden untersucht, erbringen aber nicht die erhoffte Leistungssteigerung. In einem ausführlichen Ausblick werden weitere Untersuchungen detailliert angeregt.

Inhaltsverzeichnis

1 Einführung **1**

1.1 Zielstellung 2

1.2 Konzeptioneller Rahmen 2

1.3 Abgrenzung 3

1.4 Intelligenz in der Robotik 4

1.4.1 Symbolbasierte Künstliche Intelligenz 6

1.4.2 Probabilistische Robotik 6

1.4.3 Verhaltensbasierte Systeme 7

1.4.4 Sensomotorisches Paradigma 9

1.5 Reale Umwelt und Simulation 9

1.6 Übersicht 10

1.7 Zusammenfassung 11

2 Grundlagen für Verhalten und deren Koordination **13**

2.1 Verhalten 13

2.2 Agenten 14

2.2.1 Eigenschaften von Agenten 15

2.2.2 Agenten und Verhalten 16

2.2.3 Animat 17

2.3 Erzielung von Verhalten 19

2.3.1 Verhalten ohne Lernen 19

2.3.2 Lernender Verhaltenserwerb 19

2.3.3 Anforderung an selbstlernende Agenten 20

2.3.4 Erwerb von Verhalten mittels Reinforcement Lernen 21

2.4 Verhaltenskoordination 22

2.4.1 Multiagentensysteme 24

2.4.2 Vor- und Nachteile monolithischer bzw. verteilter Realisierungen . . 25

2.4.3 Organisation von MAS 27
2.4.4 Kategorisierung von MAS 30
2.5 Distributed Artificial Intelligence 32
2.6 Zusammenfassung 36

3 Szenarien für Verhaltensexperimente 37
3.1 Merkmale von Szenarien 37
3.2 Eigenschaften der Objekte 38
3.3 Qualitative Anforderungen an Animats 39
3.4 Auswahl möglicher Aufgaben 40
3.5 Benchmarks und Performanzmaße 41
3.6 Zusammenfassung 43

4 Verhaltenserwerb mit Reinforcement Lernen 45
4.1 Einordnung und Übersicht 46
4.2 Grundlagen 47
4.2.1 Markow-Eigenschaft 47
4.2.2 Wertfunktion 48
4.2.3 Dynamische Programmierung 52
4.3 RL mit Zustandsaktionswerten 53
4.3.1 Diskontierte Zustandsaktionswerte 53
4.3.1.1 Eligibility Traces 54
4.3.1.2 *Q-Learning* und *Sarsa* 55
4.3.2 Nichtdiskontierte Zustandsaktionswerte: Average Reward 56
4.3.3 Residuale Gradienten 57
4.3.4 RL mit Policysuche 58
4.4 Gradientenbasierte Policysuche 59
4.5 Das Exploration-Exploitations-Dilemma 61
4.6 Probleme beim Reinforcement Lernen 64
4.6.1 Teilweise Beobachtbarkeit 64
4.6.2 Komplexität des RL 65
4.7 Hierarchien von Teilaufgaben 67
4.8 Makroaktionen 68
4.9 Neuronale Umsetzung von RL-Agenten 69
4.9.1 Partitionierung des Eingangsraumes mit lokaler Funktionsapproximation 70

4.9.2 Direkte, globale Funktionsapproximation 72
4.10 Zusammenfassung 72

5 Adaptive Verhaltenskoordination 75
5.1 Grundlagen für MAS 75
5.1.1 Multiagent MDP 76
5.1.2 Vollständig kompetitive MMDP 78
5.1.3 Nichtkooperative MMDP 78
5.1.4 Vollständig kooperative MMDP 79
5.1.5 Alternative Ansätze 80
5.2 Vorstellung von W-Lernen 81
5.2.1 Bedeutung der W-Werte 82
5.2.2 Auswertung der W-Werte 85
5.2.3 Wertung von WL 86
5.3 Alternative Multiagentenansätze 87
5.3.1 Psi (Ψ) 87
5.3.1.1 Bedürfnisse 88
5.3.1.2 Verhaltenserwerb 89
5.3.1.3 Verhaltensmodi 89
5.3.1.4 Wertung 90
5.3.2 Democratic Integration 90
5.3.3 Dynamische Policyverschmelzung 91
5.3.3.1 Simple Koordinierung 92
5.3.3.2 Lookahead Suche 93
5.4 Zusammenfassung 93

6 Festlegungen zu Aufgaben und Experimenten 95
6.1 Szenario 95
6.1.1 Höhere Anforderungen an das Animat 96
6.1.2 Umsetzung des Szenarios 97
6.2 Struktur der Agenten 97
6.2.1 Clusterer 98
6.2.2 Reinforcement Lernalgorithmus 98
6.3 Koordination der Agenten 99
6.3.1 Gegenüberstellung von Monolith und Koordination 99

6.3.2 Gleichzeitiges Lernen der Agenten und des W-Lernens 101
6.3.3 Exploration im W-Lernen 101
6.3.4 Modifikationen am Lernalgorithmus 102
6.3.5 Zustandsabhängige Priorisierung der Agenten 103
6.4 Verhaltensexperimente 105
6.5 Zusammenfassung 106

7 Experimentelle Umsetzung 107
7.1 Szenario 107
7.1.1 Szenario mit dynamischen Objekten 108
7.1.2 Steuerung der dynamischen Objekte 109
7.2 Das Experimentalsystem 110
7.2.1 Die Effektoren 111
7.2.2 Die Infrarotsensoren 111
7.2.3 Die omnidirektionale Farbkamera 112
7.2.3.1 Bildvorverarbeitung 113
7.2.3.2 Farbverarbeitung 114
7.2.3.3 Auswahl des Farbraumes und der Farben 115
7.3 Umsetzung des Szenarios als Simulation 118
7.4 Umsetzung der Einzelagenten 120
7.5 Durchführung der Experimente 123
7.6 Voruntersuchungen 125
7.6.1 Auswahl der Agentenkonfiguration 126
7.6.2 Notwendigkeit eines Vortrainings und Wahl des Trainingsmodus . . 127
7.6.3 Vergleich der rezeptiven Felder mit und ohne farbspezifische Vorverarbeitung 129
7.6.4 Das Szenario als MDP 131
7.6.4.1 Realisierung mit RL 132
7.6.4.2 Abschätzung des Lernaufwandes 132
7.6.4.3 Experimente und Ergebnisse 133
7.7 Zusammenfassung 134

8 Ergebnisse der Experimente 135
8.1 Performanz der Agenten und des W-Lernens 136
8.1.1 Verhalten der Experten im simulierten Szenario 136
8.1.2 Verhalten der Koordinationen im simulierten Szenario 139

8.1.3 Verhalten der Experten im realen Labyrinth 141
8.1.4 Verhalten der Koordinationen im realen Labyrinth 142
8.1.5 Analyse des Verhaltens 145
8.2 Szenario mit dynamischen Objekten 147
8.2.1 Ergebnisse der Experten mit dynamischen Objekten im simulierten Szenario 147
8.2.2 Ergebnisse der Koordinationen mit dynamischen Objekten im simulierten Szenario 149
8.2.3 Ergebnisse der Experten mit dynamischen Objekten im realen Labyrinth 151
8.2.4 Ergebnisse der Koordinationen mit dynamischen Objekten im realen Labyrinth 153
8.3 Erweiterungen des W-Lernens 154
8.3.1 Gleichzeitiges Anlernen der Expertenagenten 155
8.3.2 Exploration im W-Lernen 157
8.3.3 Zustandsabhängige Priorisierung der Agenten 159
8.3.4 Direktes Lernen aus dem Gesamtreward 161
8.4 Zusammenfassung und Wertung 162

9 Weiterführende Untersuchungen 165
9.1 Erweiterungen am Animat 165
9.1.1 Zeitlich variable Ziele: Bedürfnisse 165
9.1.2 Koordination mit dynamischen Bedürfnissen 169
9.1.3 Biologisch motivierte Agenten 171
9.2 Optimierung der Agenten 171
9.2.1 Clusterer 172
9.2.2 RL-Algorithmus 172
9.3 Weiterentwicklung der Koordination 173
9.4 Emergente Modularisierung 174

10 Zusammenfassung 175

Lokalisation des Roboters 177
A.1 Grundsätzlicher Ablauf 177
A.2 Bestimmung von Position und Orientierung des Roboters 178
A.2.1 Positionsbestimmung 179
A.2.2 Orientierungsbestimmung 180

A.3 Bestimmung der Position von Hindernissen im Labyrinth 181

Abkürzungen **183**

Literaturverzeichnis **183**

Glossar 197

Index 199

Kapitel 1

Einführung

Viele sind begeistert von denkenden Maschinen wie IBM's *Deep Blue*, Sony's *Aibo* oder dem 'Terminator', und doch sind noch keine praxistauglichen Lösungen, wie intelligente Haushaltsroboter, verfügbar.

Ein Grundanliegen der Neuroinformatik und der Künstlichen Intelligenz allgemein ist die Herausstellung von Prinzipien und Methoden von **Intelligenz**. Motiviert wird dieses Anliegen sowohl durch wissenschaftliche Neugier, aber auch durch gewachsene und wachsende Komplexität der praktischen Aufgabenstellungen. Komplexere Aufgabenstellungen ergeben sich unter anderem aus dem Bestreben, traditionelle Einschränkungen für Steuerungen oder Verhaltensgenerierung aufzuheben. Die Systeme sollen auftretende Probleme selber lösen. Das von der Neuroinformatik eingesetzte Vorgehen ist geprägt durch die Idee, solche prinzipiellen Methoden durch ein **Nachempfinden natürlicher Vorbilder**, also zunächst Tieren und dann auch Menschen, zu finden. Brooks drückt dies im Begleitmaterial zur Ausstellung 'Computer.Gehirn' [Brooks, 2002] wie folgt aus:

> Um Vorstellungen von Intelligenz tatsächlich überprüfen zu können, ist es wichtig, vollständige Agenten zu bauen, welche mit realen Sensoren in einer dynamischen Umwelt operieren. Die vollständige Repräsentation der tatsächlichen realen Umgebung ist für die interne Modellbildung nicht notwendig, um den Agenten kompetent handeln zu lassen, zumal eine derartige Modellbildung unrealistisch ist. Viele Aktionen eines Agenten sind eindeutig separierbar – eine kohärente Intelligenz kann durch das Zusammenfügen von in ihrer Umwelt interagierenden Teilkomponenten entstehen.

Es ist also sinnvoll, als Agenten, die mit einer **realen Umwelt** interagieren, Roboter zu bauen, um Intelligenzleistungen erforschen zu können. Werbos [Werbos, 1995] umreißt die zur Nachbildung intelligenten Verhaltens essentiell notwendigen Teile als

1. ein emotionales System, das den Wert beobachteter Objekte und Variablen für den Organismus abschätzt,
2. ein Erwartungs- oder Prädiktionssubsystem, das selbst für so initiale Leistungen wie klassische Konditionierung notwendig ist,
3. ein Aktions- oder Motorsystem, das die Aktionen berechnet.

WERBOS verlangt weiterhin, daß das entstehende System ingenieurtechnisch für eine optimale Steuerung schwieriger Probleme mit Lernen sinnvoll sein muß.

Für ein intelligentes Verhalten (siehe Abschnitt 2.3.2) und zur Erzielung von Autonomie der Roboter ist eine **Lernfähigkeit** dieser einzelnen Bestandteile notwendig. Mit ihrem klassischen Ziel, die Performanz eines lernenden Systems über dessen Lebenszeit zu optimieren [Mataric, 2001], bieten sich dafür Methoden des Maschinenlernens an.

1.1 Zielstellung

In der vorliegenden Arbeit sollen Algorithmen des Maschinenlernens herangezogen werden, um mit einem realen Roboter in realen, unbekannten Umwelten verschiedene adaptive Verhalten zu erzeugen und sie zu koordinieren. Die Verhalten sind (noch) nicht als intelligent, aber zumindest als zielgerichtet zu charakterisieren. Die Umwelt, in der sie sich bewegen, ist eine kontrollierte Laborumgebung, aber doch real.

Die Algorithmen setzen die WERBOS'schen Anforderungen zwar um, jedoch wurden sie zumeist bisher erst in Simulationen realisiert. Der Grund liegt darin, daß der lernende Erwerb von Verhalten auf realen Robotern wegen der Verrauschtheit, Stochastik und Dynamik der realen Welt kompliziert ist [Connell and Mahadevan, 1993a]. Die Algorithmen zur Erzeugung von Verhalten entstammen dem in [Connell and Mahadevan, 1993a, Brooks and Mataric, 1993, Werbos, 1995] aus einer Reihe von Gründen favorisierten *Reinforcement Lernen*. Schwerpunkt der Arbeit ist aber die Koordinierung der so entstandenen Verhalten mit einem lernenden Multiagentensystem, genauer dem **W-Lernen**. Die Koordination muß die von BROOKS angeführte Separierbarkeit des Verhaltens und die später diskutierten Vorzüge einer verteilten Problemlösung ergänzen [Brooks and Maes, 1990]. Damit reiht sich die Arbeit in die (noch) nicht häufigen Untersuchungen ein, die die Idee einer realen Erprobung der Vielzahl der vorgeschlagenen Ansätze tatsächlich bis zu einer Umsetzung auf realen mobilen Robotern verfolgen. Allerdings widmet sie sich dabei weniger der Entwicklung neuer Algorithmen, sondern dem *Vergleich des W-Lernens als lernender Koordination mit einem monolithischen Einzelagenten* beim Einsatz zur Steuerung eines zunächst einfachen, aber immer handlungsfähigen Gesamtsystems mit Sensorik, Motivation, Verhaltenslernen und lernender Verhaltenskoordination bis hin zu Aktionen in realen Umwelten.

1.2 Konzeptioneller Rahmen

Forscher wie die oben zitierten sind sich einig, daß es kompliziert und ziemlich schwierig sei [Mataric, 2001], Roboter real einzusetzen und die sie steuernden Computer lernen zu lassen. Die Gründe liegen in sensorischen und effektorischen Unsicherheiten und nur teilweiser Beobachtbarkeit und Nichtstationarität der Welt. Warum also sollte beides auf einmal unternommen werden? Die Antwort liegt in den Einsichten hinsichtlich wesentlicher Ziele der Künstlichen Intelligenz und der Neuroinformatik, die so und unter Umständen nur so erlangt werden können. Der Versuch, die Intelligenz von Tieren und Menschen durch Imitation bzw. Reimplementation zu verstehen, macht das Zusammenspiel so verschiedener und an sich komplexer Teilgebiete wie Sensoraufnahme, Aktions-

planung, Aktionsausführung und zielgerichtete Veränderung bzw. Adaptation der entsprechenden Algorithmen notwendig. Jedes dieser Teilgebiete hält für sich genommen einige Herausforderungen bereit, und doch ergibt erst ihr Zusammenwirken den gesuchten Sinn. Erst ihr Zusammenspiel entscheidet tatsächlich über Adäquatheit oder Optimalität, über die Schnittstellen und Randbedingungen der Teilgebiete. Die eigentliche Prüfung aber ist die Anwendung des Gesamtsystems in der Realität – all die Verfahren und Algorithmen in der Umwelt zu prüfen, in der sich auch die Vorbilder, Tiere und Menschen, beweisen.

Insgesamt ergibt das eine sehr anspruchsvolle Zielvorstellung für autonome intelligente Systeme: eingesetzt in einer dynamischen Umwelt und ausgestattet mit Zielen, entscheiden sie selber über Aufgabenaufteilung, notwendige Sensorik und Aktuatorik, sowie die Verarbeitung zwischen Sensorik und Aktuatorik. Das läßt sich im Sinne eines Top-Down-Vorgehens wie folgt umreißen:

- Finden einer optimalen Aufgabenzerlegung für die Gesamtaufgabe
- Finden eines minimalen Satzes von Merkmalen für die Zustände
- Finden eines minimalen Satzes von Zuständen für jede Teilaufgabe
- Finden einer optimalen Verteilung der Zustände im Zustandsraum
- Finden einer optimalen Policy für jede Teilaufgabe
- Finden einer optimalen Koordination der Teilpolicies

Aus diesem Gesamtkatalog zu lösender Aufgaben beleuchtet die vorliegende Arbeit nur einen kleinen Ausschnitt, jedoch bilden diese Aufgaben für sie den notwendigen Rahmen. Es werden Methoden zur Erzeugung von Verhalten und deren adaptiver Koordinierung untersucht. Die notwendige Aufgabenzerlegung, wie auch die relevanten Eingangsinformationen, werden noch vorgegeben. Allerdings erfolgt auch die Verteilung der Zustände im Zustandsraum adaptiv. Um vollständige Autonomie zu erreichen und dem Gesamtsystem zu ermöglichen, eine ihm optimal entsprechende Informationsverarbeitung selber zu bilden, müssen in späteren Arbeiten auch die Aufgabenzerlegung und eine adaptive, selektive Aufmerksamkeit der Sensorik betrachtet werden.

1.3 Abgrenzung

Die zentralen Begriffe dieser Arbeit, **Agent** und **Multiagentensystem**, drängen Assoziationen zu verschiedenen Forschungsfeldern auf, von denen viele nicht näher angesprochen werden können und die hier sehr knapp ausgegrenzt werden sollen. Auf die relevanten Konzepte wird an den entsprechenden Stellen später ausführlich eingegangen.

Der Begriff des Agenten beschreibt allgemein Systeme, die tätig sind ('agens'=handeln), um bestimmte Aufträge zu erfüllen und Ziele zu erreichen. Die hier relevanten Agenten steuern einen Roboter hinsichtlich gegebener Ziele in der Umwelt und werden in Kapitel 2 genauer umrissen. Sie schließen jedoch weder Suchbots im WWW noch Börsenmakler,

virtuelle oder kognitive Aktoren ein. Aus informatischer Sicht und aufgrund ihrer Implementierung steht ihnen das Konzept eines Objekts aus der Objekt-Orientierten Programmierung näher, von dem sich ein Agent aber durch seine Kontrolle über die Ausführung seiner Aktionen unterscheidet. Ein Agent bestimmt selber, wann er was unternimmt, während ein Objekt ausführen muß, was andere Objekte, oder letztendlich der Designer, ihm abverlangt.

Die Beispiele zu Multiagentensystemen sind wesentlich weitläufiger, da diese (noch) nicht rigide definiert sind und es sich um ein Feld handelt, das einige Tradition hat, aber in jüngster Vergangenheit wieder stark wachsendes Interesse mit entsprechenden Umbrüchen und Neuerungen verzeichnet. Ein Multiagentensystem (MAS) umfaßt natürlich mehrere Agenten, die wegen Ressourcenbegrenzungen interagieren müssen, aufgrund sich ergänzender Fähigkeiten interagieren sollten, oder funktionale Teile eines Ganzen bilden. In diesem Sinne könnten auch die in dieser Arbeit betrachteten Agenten als Multiagentensystem aufgefaßt werden, da alle Funktionen zwischen Sensorik und Aktuatorik als untrennbare Einheit betrachtet werden müssen [Jennings et al., 1998].
Eine Objektbibliothek, aber auch ein neuronales Netz, die Organe eines Tiers oder ein Vogelschwarm können als Multiagentensystem betrachtet werden. Diese Vergleiche sind schwieriger abzugrenzen als die für Agenten. Die hier betrachteten Multiagentensysteme sind weder eine generische Softwareschicht zur Abbildung bestimmter Modelle auf eine verteilte oder zentrale Hardware [Schumacher, 2001], noch werden sie als Suchalgorithmen zur Constraint Satisfaction [Yokoo, 2001] betrachtet, obgleich sie als solche aufgefaßt werden können. Auch Kollektive mehrerer physischer Agenten mit problemunabhängigen Interaktionsmechanismen ('Kenetik' [Ferber, 2001]) sind nicht Gegenstand dieser Arbeit, obgleich der Begriff Multiagentensystem hierfür sehr häufig benutzt wird. In dieser Arbeit werden Koordinationen von Agenten als Experten für eine Teilaufgabe untersucht, die zugunsten des Überlebens des Gesamtsystems kooperieren, aber um den Zugriff auf dessen Aktuatorik konkurrieren. Kapitel 2 vergleicht die entsprechenden Koordinationen mit den verbreiteten, relevanten Ansätzen der Distributed Artificial Intelligence (DAI) (Verteilte Künstliche Intelligenz (VKI)).

1.4 Intelligenz in der Robotik

Die intelligente Problemlösung in der realen Umwelt ist ein Anwendungsgebiet für verschiedene Forschungsfelder: zum einen die Erforschung zumindest von Prinzipien der Funktion menschlicher Intelligenz, und zum anderen die Konstruktion technischer Systeme, die sich in realen Umgebungen zielgerichtet verhalten sollen.

Beide Ziele werden seit langem mit unterschiedlichen Ansätzen verfolgt, ersteres insbesondere durch die symbolbasierte Künstliche Intelligenz, letzteres insbesondere durch Regelungs- und Steuerungstechnik. Ohne auf die Erfolge und Grenzen dieser Herangehensweisen detailliert eingehen zu wollen, ist doch evident, daß keine von beiden bislang bei der Erzeugung technischer Systeme erfolgreich war, die mit der Welt auf eine Art und Weise interagieren könnten, die der von Tieren oder gar Menschen nahe kommt [Digney and Gupta, 1994].

Zu einem tatsächlichen Verständnis der Intelligenzleistungen von Tieren oder Men-

schen scheint es unumgänglich, das klassische Vorgehen, ein komplexes Problem, wie z.B. eine Verhaltenserzeugung, in funktionale Bestandteile zu zerlegen, aufzugeben. An seine Stelle tritt ein synthetisierender Ansatz: es muß ein reales, sich so realistisch wie beherrschbar verhaltendes System geschaffen werden, das alle Aspekte des Überlebens und der Zielverfolgung umsetzt (vgl. Abschnitt 1.4.3). Die verschiedenen Informationsverarbeitungsvorgänge, die offensichtlich zu Intelligenzleistungen gehören oder führen, können nicht wirklich losgelöst voneinander und von ihrer Umwelt verstanden werden [Connell and Mahadevan, 1993c]. Ein wirkliches Verstehen kann sich erst aus der eigenen realen Umsetzung ergeben, wie ein wirkliches Verstehen von Autofahren sich nicht aus der Beobachtung vom Beifahrersitz aus, sondern erst hinter dem Lenkrad erschließt [Millman, 1980].

Kuipers postuliert in [Kuipers et al., 1993] eine Hierarchie der Verfahren für den Rahmen einer technischen Nachbildung tierischer oder menschlicher Leistungsfähigkeit hinsichtlich der Erfüllung bestimmter Aufgaben, bis hin zu kognitiven Leistungen bzw. Intelligenz:

1. sensomotorisches Verhalten
2. Topologie
3. Geometrie

Jede der Stufen verfügt über eigene Methoden zur Beschreibung der Umwelt und Problemlösung.

Die Stufe des sensomotorischen Verhaltens bildet Unterschiede in der Sensorik direkt auf Unterschiede im Verhalten ab. Sie erreicht robuste Performanz unter Überwindung der Effekte von Unsicherheiten in der Sensorik und Aktuatorik. In der topologischen Stufe werden die von der Verhaltensstufe unterschiedenen Orte und Pfade zugrunde gelegt und dazwischen Beziehungen definiert, die ein Netzwerk ergeben. Diese Ebene dient der Führung einer Exploration in neuen Umwelten, aktiver Exploration zur Unterscheidung zweideutiger Zustände und zur Pfadplanung. Auf der Stufe der Geometrie wird die topologische Information um metrische Eigenschaften wie Entfernung, Richtung oder Form erweitert.

Zur Steuerung autonomer mobiler Roboter als realer Umsetzung einer solchen technischen Nachbildung intelligenten Verhaltens bieten sich drei Zweige der Robotik an. Dabei handelt es sich um die symbolbasierte Künstliche Intelligenz (sKI), die noch junge probabilistische Robotik (PR) und verhaltensbasierte Systeme. Alle drei setzen die eben genannten Aufgaben mit unterschiedlichen Ausgangspunkten bzw. unterschiedlichem Herangehen um.

Maes [Maes, 1993] gibt einen Vergleich zwischen symbol- und wissensbasierter Intelligenz und verhaltensbasierten Systemen. Eine sehr ähnliche Abgrenzung trifft Schumacher in [Schumacher, 2001]. Diese beiden Zweige der KI gehen im gleichen Rahmen unterschiedliche Probleme an. Während der Schwerpunkt der sKI auf Planung liegt und von menschlicher Intelligenz ausgeht, erzeugen verhaltensbasierte Systeme immer ein Verhalten in realen Umwelten ausgehend von einfachen Reflexen. Wegen des gleichen Rahmens

der Erzeugung intelligenten Verhaltens werden die beiden Zweige oft als Konkurrenten verstanden, obgleich nicht nur MAES und KUIPERS sie für komplementär halten.

Die Probabilistische Robotik verfolgt nicht die Festlegung eines eindeutigen Zustandes oder einer eindeutigen Aktion, sondern bezieht die Unsicherheiten der Sensoren und Aktuatoren explizit in Wahrnehmung und Aktionsplanung ein. Statt bester Vertreter werden immer Wahrscheinlichkeitsverteilungen betrachtet. Zur Erzeugung von Verhalten kann sie sich der Methoden beider Zweige der KI bedienen.

Als zentraler Punkt der sKI könnte die Repräsentation bezeichnet werden, der der Probabilistischen Robotik liegt in der Beachtung sensorischer und aktuatorischer Unsicherheiten auf der Basis von Designerwissen, während der Fokus der verhaltensbasierten Systeme die adaptive Interaktion mit der Umwelt ist.

1.4.1 Symbolbasierte Künstliche Intelligenz

Die symbolbasierte KI (sKI) wird hier kurz charakterisiert, da sie die Grundlage für viele Realisierungen von Multiagentensystemen darstellt. Auf sie bezieht sich BROOKS in dem zu Anfang des Kapitels erwähnten Zitat.

Auf der Basis einer symbolischen Repräsentation wird meist ohne direkte Interaktion mit ihrem Problemfeld versucht, spezielle und schwierige Probleme sequentiell und ohne Zeitbeschränkungen optimal zu lösen. Als hervorragendes Beispiel sei IBM's 'Deep Blue' genannt, der Schachweltmeister Garri Kasparow schlug.
Die sKI bedienen sich feststehender, funktional orientierter Strukturen, die ihnen weite perspektivische Planungen, Optimalität der Pläne und die Möglichkeit einräumen, diese Pläne auch zu erklären. Basierend auf einer zentralen Repräsentation aus der Perzeption werden top-down nacheinander die Kommunikation, die Inferenz und eine Aktion ausgeführt. Die entsprechenden Module können so generell wie möglich ausgelegt werden und setzen ein möglichst vollständiges internes Modell voraus. Andererseits schließt das nichtmodellierte Umweltzustände oder Umweltänderungen aus und eine Adaptation ist nicht vorgesehen. Auf dieser generischen Grundlage jedoch sind sie in der Lage, mit minimalen internen Veränderungen auf Änderungen der Rahmenbedingungen, wie veränderte Aufgabenstellungen oder Rewardverteilungen, Sensoren oder Effektoren zu reagieren, soweit das interne Modell dafür beschreibungsfähig genug ist.

Die Systeme der sKI verfügen mit dem Modell über ein allgemein gültiges Werkzeug, das nicht von den konkreten Aufgaben oder Sensoren abhängt. Damit lassen sich auf einem solchen Modell beliebige Manipulationen auf Aufgaben und Algorithmen anwenden.

Für ihre Symbolmanipulation nimmt die sKI Voraussetzungen hinsichtlich der Rationalität und Planbarkeit über die Umwelt an, die einer mobilen Umsetzung in realen Umwelten widersprechen, sodaß die Lösungen der sKI für ein Verhalten auf einem realen mobilen Roboter schlecht implementierbar sind.

1.4.2 Probabilistische Robotik

THRUN diskutiert in [Thrun, 2000] eine Herangehensweise an die Steuerung von Robotern, die mit einer prinzipiellen Annahme bricht, welche sKI und verhaltensbasierte Systeme

teilen; daß es nämlich nur genau eine beste Interpretation des Umwelt- und des Roboterzustandes geben soll, die zu einer einzigen adäquaten Aktion führt. Während offensichtlich der Roboter nur eine Aktion ausführen kann, ist die Beschränkung auf eine beste Zustandsinterpretation nicht tatsächlich notwendig und nach THRUN sogar der Leistungsfähigkeit des Roboters mit wachsender Aufgabenkomplexität abträglich. Die Planung in der probabilistische Robotik basiert ähnlich der sKI auf einem zentralen, verfügbaren Umweltmodell, das allerdings mit starkem Bezug auf die eigenen Sensoren und Effektoren geschätzt oder gelernt werden kann, was wiederum den verhaltensbasierten Systemen ähnelt. Ein erfolgreiches Beispiel für diesen Ansatz ist der Roboter *Rhino*, der Besucher durch das Smithsonian Institute in Washington, DC, USA, führte.

Der Ansatz der *probabilistischen Robotik* bezieht die Unsicherheiten, die den Roboter bezüglich des Zustandes der Umwelt, aber auch seiner eigenen Sensoren und Aktuatoren hat, explizit in seine Methoden der Perzeption und der Aktionsgenerierung ein. Realisiert wird dies, indem statt einer Abschätzung des besten Kandidaten eine ganze Wahrscheinlichkeitsverteilung mitgeführt, ständig aktualisiert und verarbeitet wird.

Insbesondere bei der Aufbereitung der sensoriellen Information zeigt sich eine grosse Stärke der probabilistischen Robotik. Die dem Ansatz zugrundeliegende Theorie stützt sich auf die Annahme der MARKOW-Bedingung (siehe Abschnitt 4.2.1). Jedoch kann ein Teil der Probleme durch die gleichzeitige Verfolgung mehrerer alternativer Hypothesen erleichtert werden. Mit der Verwendung von Wahrscheinlichkeiten als gemeinsamer Darstellung und Grundlage der Hypothesen können verschiedene Arten von Sensoren integriert werden und so Mehrdeutigkeiten und Ausfälle besser behandelt werden.

Bei der Aktionsauswahl können mit den verschiedenen Hypothesen dann auch mehrere parallele, alternative Möglichkeiten von Entwicklungen betrachtet werden. Hierbei strebt die probabilistische Robotik ein entscheidungstheoretisches Optimum an, das gegenwärtig meist mit Werkzeugen aus dem Arsenal der sKI ermittelt wird, aber neuerdings auch mit subsymbolischen Methoden [Rodgríguez et al., 2000].

Die Robustheit und der Grad der Autonomie der entwickelten Systeme werden durch einen immensen Berechnungsaufwand für die verschiedenen Wahrscheinlichkeitsdichten und die notwendigen Schätzungen in einer kontinuierlichen Umwelt erkauft. Natürlich wird dieser Aufwand durch den Fortschritt der rechentechnischen Hardware relativiert. Wie in der sKI, bestimmt auch in der probabilistischen Robotik die Repräsentation über Robustheit und Effizienz der Algorithmen. Obgleich die Interaktion des Roboters mit seiner Umwelt eine essentielle Notwendigkeit darstellt, wird bisher Lernen mit probabilistischer Robotik kaum betrachtet. Gleiches gilt für ihre Anwendung in Multirobot- bzw. Multiagentenproblemen.

1.4.3 Verhaltensbasierte Systeme

Eine alternative, zunächst naiv erscheinende, Herangehensweise setzt das Nachempfinden tierischer Intelligenz mit der oben geforderten Betrachtung der Gesamtheit aller beteiligten Prozesse direkt um. Es werden immer vollständige, zunächst einfache, Systeme gebaut, die sich in der realen Welt bewegen müssen. Die Umwelt wird direkt statt eines Modells benutzt. In eng begrenzter Zeit nehmen sie Informationen aus der Umwelt auf und erzeugen eine Reaktion, die auf die Umwelt zurückwirkt. Das aufgenommene Wissen kann für

ein sKI-Modell unzureichend oder unvollständig sein, umfaßt aber alle nötigen Aspekte ihrer Umwelt, um Verhalten zu erzeugen. Die Verhalten können mehrere unterschiedliche, womöglich veränderliche, Ziele erfüllen. Ziel ist eine Aktion in einer begrenzten Zeit, damit das System überhaupt sinnvoll ist, ansonsten wird es zerstört oder die Umwelt verändert sich so signifikant, daß die gefundene Reaktion des Systems sinnlos wird.

Dies soll den Ansatz der 'verhaltensbasierte Systeme' (behavior-based systems (BBS) [Brooks, 1991a]) kurz umreißen. Ein Beispiel ist der immer weiter vervollständigte und mit einem wachsenden Umfang an Fähigkeiten ausgestattete Roboter 'Cog' am Massachussetts Institute of Technology.

Für BBS ist nach Brooks Intelligenz ein Nebenprodukt der Interaktion des Agenten mit der Umwelt [Jennings et al., 1998]. Diese Interaktion basiert auf Modulen, die Brooks 'behaviors' (Verhalten) nennt und von denen mehrere unterschiedliche unter der Zielsetzung des Überlebens des Organismus parallel arbeiten. Sie brauchen dazu nur adäquat, aber nicht optimal sein. Das Herangehen vollzieht mit einer inkrementellen Entwicklung der künstlichen Wesen die biologische Evolution in gewisser Weise nach. Beginnend mit einfachstem Verhalten muß immer eine Funktion des Wesens in der realen Umwelt gewährleistet sein [Brooks, 1991b]. Kompliziertere Verhalten benutzen und steuern einfache Verhalten in der hierarchischen *Subsumption*-Architektur [Brooks, 1986], die in Anschnitt 2.4.3 genauer betrachtet wird.

Diese Sichtweise bedeutet einen radikalen Bruch mit der funktionalen Unterteilung der Informationsverarbeitung in Sensorik, Planung, Aktionsgenerierung und Aktuatorik. Jedes der Verhalten umfaßt Sensorik und Aktuatorik und dazwischen eine minimale Abbildung.

Eine Repräsentation, ein Modell, eine Inferenz oder Planung werden nicht benötigt, da das System direkt mit der Umwelt interagieren und entsprechende Ressourcen besser verwenden kann. Statt vollständiger, genereller Pläne lösen BBS ihre zunächst einfachen Probleme mit minimaler, unter Umständen direkter [Braitenberg, 1993, Seth, 1998] Abbildung von Sensoren auf Aktuatoren, jeweils spezifisch und iterativ mit der Versuch-und-Irrtum-Methode. BBS entwickeln nur genau diejenige Kompetenz, die für ihre Verhalten notwendig ist, denn ein System braucht nur so gut zu sein, wie es die Umwelt und die Aufgabe verlangen. Eine Erklärung des 'Warum' ist meist ausgeschlossen, da nur eine notwendige, aber keine dem Beobachter verständliche Abbildung zwischen Sensorik und Aktuatorik produziert wird. Die entstehenden Strukturen sind aufgabenorientiert, parallel und fehlertolerant. Sie erreichen robuste, sehr schnelle, aber feststehende Reaktionen auch in wechselnden Umweltbedingungen.

Eine Umweltrepräsentation ist bei den BBS implizit und wurde nur unter den Randbedingungen der realen Zugänglichkeit der Umwelt und der aktuellen Aufgabe und Konfiguration des Systems erlernt. So ist eine zentrale Repräsentation einem BBS nicht dienlich, eine Abbildung des sensorischen Perzepts auf ein Symbol überflüssig. Jedoch startet das Lernen der sensomotorischen Abbildung wieder vom Nullpunkt, wenn sich eine der Randbedingungen, nämlich Aufgabenstellung, Sensorik oder Aktuatorik auch nur ein wenig ändert. Weitere Probleme ergeben sich für BBS bei allen Aufgaben, die sich nicht allein aus dem augenglicklichen Zustand des Systems lösen lassen. Dazu zählen solche, die interne Repräsentationen, Symbolmanipulationen, abwägende Planungen oder die Erläuterung von Verhalten erfordern.

Brooks stellt polemisch die Idee der Repräsentation allgemein in Frage. Allerdings erscheint nach jüngeren Forschungen eine Auffassung über sKI und verhaltensbasierte Systeme angebrachter, in der die beiden Zweige der KI sich ergänzen. Solche Ansätze werden in dieser Arbeit nicht betrachtet, da zunächst einfache Aufgaben verwendet werden.

1.4.4 Sensomotorisches Paradigma

Der Grundgedanke der in dieser Arbeit verfolgten Erzeugung von Verhalten wird im 'sensomotorischen Paradigma' [Möller, 1996] ausgedrückt. Es besagt, daß zwischen Sensorik und Aktuatorik eine minimale Abbildung stattfinden soll. Eine Abbildung der Sensorik auf ein symbolisches oder subsymbolisches Perzept und die Verwendung des Perzepts in einer zweiten oder in mehreren Abbildungen zur Erzeugung einer Aktion mag dem Designer entgegenkommen, behindert aber das System. Eine solche Darstellungsart und funktionale Sichtweise entspricht zwar dem Verständnis des Designers, nicht jedoch den Anforderungen des Systems.

Entsprechend bilden die eben diskutierten verhaltensbasierten Systeme (BBS) den grundsätzlichen Ansatzpunkt zur Erzeugung von Verhalten in dieser Arbeit. Das Paradigma erweitert den Ansatz der BBS aber hinsichtlich der Autonomie der Agenten beim Erwerb der Verhalten und der Koordination von verschiedenen Verhalten.

Weitergeführt bedeutet das Sensomotorische Paradigma eine wahre Autonomie des Systems: es muß nicht nur physisch manifest und selber agierend sein, sondern das System muß selber entscheiden können, welcher seiner Sensoren, welche Aktionen und Aktuatoren für welche Aufgaben notwendig sind. Es muß auch eine zweckmäßige Unterteilung einer komplexen Gesamtaufgabe in beherrschbare Teilaufgaben selber vornehmen können. Daraus ergibt sich der in Abschnitt 1.2 eingeführte konzeptionelle Rahmen der Arbeit. Über diese Arbeit hinaus ist das Paradigma weiterhin Grundlage der Untersuchungen an dem Fachgebiet, in das die Arbeit eingebettet war. Hier ist es insbesondere mit dem Konzept der Antizipativen Wahrnehmung verknüpft, die auch verhaltensbasierten Systemen eine flexiblere Planung ermöglichen (siehe [Stephan, 2002]).

1.5 Reale Umwelt und Simulation

Verhaltensbasierte Systeme und die Umsetzung des sensomotorischen Paradigmas schliessen essentiell eine reale Umsetzung der entwickelten Algorithmen ein.

Die Methoden, Algorithmen und Ergebnisse aus Simulationen sind nur sehr schwer [Brooks and Mataric, 1993, Kuipers et al., 1993], auf die Realität übertragbar. Simulationen sind dazu verurteilt, erfolgreich zu sein, da der Designer bewußt, aber auch unbewußt dafür Sorge tragen wird. Sie können nicht ausreichend realitätsnah gemacht werden, was zum Beispiel auf den Determinismus der Zustandsübergänge in Simulationen zutreffen kann. Darüberhinaus entstehen in einer Simulation, aus deren serieller Ausführung und mathematischen Exaktheit, Probleme, die in der Praxis einfach nicht in Erscheinung treten, wie perfekte Gleichzeitigkeit oder perfekte Symmetrie.

Die reale Umwelt bringt aber auch einige Eigenheiten mit sich, die besondere Anforderungen an Konzepte und Algorithmen, die oben genannte Probleme lösen können, stellen

bzw. einige Algorithmen komplett disqualifizieren. Diese Anforderungen werden hier zusammen aufgeführt, da bei den einzelnen Algorithmen auf einige eingegangen wird, der gesamte Katalog aber nicht gelöst werden kann:

- nie völlige Wiederholbarkeit
- nie vollständige Beschreibung
- Unzuverlässigkeit, Drift und Rauschen der Sensoren und Effektoren
- nie strenger Determinismus der Aktionen, d.h. der Effekt von Aktionen kann sich radikal unterscheiden
- Dynamik
- Nichtstationarität
- Reaktion auf unvorhergesehene Situationen in angemessener Zeit bzw. Echtzeit notwendig.

Ein realer Roboter eröffnet die Möglichkeit, alle Stufen der Realität von der sterilen Simulation über Fahrten im Labor bis hin zum tatsächlichen realen Einsatz zu erproben. Allen Problemen der realen Umwelt zum Trotz ist erst eine reale Umsetzung aller ineinandergreifender Teilprozesse ein wirklicher Test für die Prinzipien, die die Neuroinformatik und das Maschinenlernen über relevante Informationsverarbeitung gefunden zu haben glauben. Sie stellt somit die notwendige Schlußfolgerung aus dem sensomotorischen Paradigma dar.

Trotzdem wird auch in dieser Arbeit auf Simulationen zurückgegriffen, um Ideen der Algorithmen zu vergleichen, die lange Experimentierzeiten voraussetzen, deren Aussagen aber so prinzipiell sind, daß die Untersuchungen nicht in der realen Umwelt ausgeführt werden müssen oder sollten. Erfahrungen in der realen Umwelt sind meist so teuer bzw. zeitaufwendig, daß Vorbetrachtungen in simulierten Umwelten notwendig werden. Vergleiche der Ergebnisse aus simulierter und realer Umwelt werden wegen der Nichtvergleichbarkeit gar nicht oder nur qualitativ gezogen.

1.6 Übersicht

In Kapitel 2 werden nach der Festlegung auf das sensomotorische Paradigma die Terminologie und wesentlichen Eigenschaften von Verhalten, Agenten und Multiagentensystemen eingeführt. Kapitel 3 widmet sich einer Charakterisierung von verbreiteten Umwelten, in denen das System sich beweisen muß. Eine relativ ausführliche Darstellung der Algorithmen zum lernenden Erwerb von Verhalten in solchen Umwelten schließt sich in Kapitel 4 an und wird in Kapitel 5 um eine Darstellung des Lernens in Multiagentensystemen ergänzt. In Kapitel 6 werden Festlegungen zum Szenario, den Agenten und Koordinationen getroffen sowie die Untersuchungen motiviert, deren reale Umsetzung und Rahmenbedingungen in Kapitel 7 dargestellt werden. In Kapitel 8 schließlich werden

die Ergebnisse der Untersuchungen vorgestellt, und in Kapitel 9 weiterführende Untersuchungen im Sinne eines Ausblicks hinsichtlich intelligenten Verhaltens eines Tiers in realen Umwelten motiviert. Kapitel 10 schließt die Arbeit mit einer Übersicht über die realisierten Untersuchungen und Ergebnisse ab. Auf den darauffolgenden Seiten schließen sich ein Abkürzungsverzeichnis und ein Glossar an, die das Lesen der Arbeit erleichtern sollen.

1.7 Zusammenfassung

In diesem Kapitel wurde die Zielstellung dieser Arbeit eingeführt und in einen größeren konzeptionellen Rahmen eingebettet. Sie besteht in einer Untersuchung der Koordination verschiedener Verhalten mit W-Lernen auf realen Robotern. Die Verhalten werden dabei mit Algorithmen des Maschinenlernen erlernt und umfassen vollständige Systeme von Sensoren bis Aktuatoren für einfache Probleme in zunächst unbekannten Umwelten.

Die zentralen Begriffe der Arbeit, Agent und Multiagentensystem, wurden von anderen bekannten Anwendungen derselben Begriffe abgegrenzt. Zur Erzeugung eines adaptiven und einigermaßen komplexen Verhaltens wurde der Ansatz der verhaltensbasierten Systeme [Brooks, 1991b] aufgegriffen und zwei wesentlich unterschiedlichen Ansätzen gegenübergestellt. Die Erläuterungen zu Grundlagen und Terminologie von Verhalten, Agenten und Multiagentensystemen werden im nächsten Kapitel fortgeführt.

Kapitel 2

Grundlagen für Verhalten und deren Koordination

Ziel dieses Kapitels ist es, nach der in der Einleitung gegebenen Motivation eines verhaltensbasierten Ansatzes die Terminologie für Verhalten, Agenten, Verhaltenskoordinationen und Multiagentensysteme einzuführen. Neben einer Begriffsdefinition werden Merkmale und verschiedene Anforderungen an diese Strukturen dargestellt und die Vorzüge eines lernenden Erwerbs von Verhalten diskutiert.

2.1 Verhalten

Die Motivation für eine zielgerichtete Interaktion eines Tieres mit seiner Umwelt ist die Befriedigung von Bedürfnissen. Dabei muß sich das Tier nicht über diesen Fakt bewußt sein, oder Schmerz oder Hunger verspüren. Es muß nur schnell, robust und verläßlich genug ein Verhalten erzeugen, das sein weiteres Überleben sichert.

Die Erzeugung solchen Verhaltens verfolgt der Ansatz der verhaltensbasierten Systeme [Brooks, 1991a, Brooks, 1991b] (siehe Abschnitt 1.4.3). Er hat seine Vorzüge bei der Steuerung von mobilen Systemen, die sich mit einer realen Umwelt auseinandersetzen müssen, unter Beweis stellen können. Mit ihm verbindet sich ein Wechsel von einer funktionalen Zerlegung der notwendigen Verarbeitung zu einer Zerlegung in Verhalten, die mit einer möglichst direkten Kopplung von Sensoren und Effektoren eine Aktion des Systems erzeugt. In einer realen Umwelt ähneln verhaltensbasierte Systeme bedingten Reflexen. Diese minimale Realisierung der Kopplung zwischen Sensoren und Effektoren ist die grundlegendste Art der Interaktion mit einer Umwelt, die die Gesamtheit der notwendigen Verarbeitung betrachtet. Ein solcher Reflex muß Ausgangspunkt sein, wenn man Intelligenz als aktive Auseinandersetzung mit der natürlichen Umwelt durch Nachempfinden verstehen will, wie er auch Ausgangspunkt der Konditionierung [Klein, 1991] und der darauf aufbauenden Erforschung der Intelligenz in der Psychologie ist.

Im Feld der verhaltensbasierten Systeme wird Verhalten als eine Regularität in der Dynamik der Interaktion zwischen Tier und Umwelt angesehen [Dorigo and Colombetti, 1997], [Mataric, 1995]. Anders ausgedrückt handelt es sich um eine Steuerstruktur der Motorik zur Erreichung eines Ziels, die beobachtbare Muster von Aktivität aus der Interaktion zwi-

schen Roboter und Umgebung [Mataric, 2001] erzeugt. Biologisch entsteht das Verhalten aus der Situiertheit des Systems, speziell seiner Körperhaftigkeit. Zu seiner Erzeugung über vorgegebene Reflexe hinaus wird das Reinforcement Lernen (Kapitel 4) bevorzugt, da Lernen hierbei direkt in der entsprechenden Umwelt ohne besonderes Vorwissen oder Lehrer, mit den zu befriedigenden Bedürfnissen als Lernkriterium möglich ist.

Verhalten können in Basisverhalten und komplexe Verhalten unterschieden werden [Mataric, 1995]. Basisverhalten umfassen eine Minimalauswahl von Aktionssequenzen zur Erreichung eines Zieles. Komplexe Verhalten können aus einer Verkettung von Basisverhalten entstehen, die jeweils ihre Ziele erreichen, oder aus einer Folge einzelner Aktionen, die jeweils mehreren Zielen gleichzeitig dienen [Dorigo and Colombetti, 1997].

Verhalten sind nicht auf rein reaktive Reflexe begrenzt [Mataric, 1996, Mataric, 2001], sondern können auch interne Zustände speichern und verschiedene Repräsentationen aufweisen. Die verschiedenen Verhalten stützen sich jedoch nicht auf eine zentrale Repräsentation wie in der symbolischen KI, sondern auf dezentrale Repräsentationen mit ähnlichen Zeitskalen. Damit verfügen Verhalten über die Möglichkeit, ihre bisherigen Erfahrungen auszunutzen bzw. die Zukunft zu prädizieren und trotzdem noch die Echtzeitreaktion, Robustheit und einfache Erweiterbarkeit der reaktiven Systeme zu erhalten.

Die Umsetzung dieser Möglichkeiten führt zu Verhalten unterschiedlicher kognitiver Komplexität, die sich von gering zu hoch wie folgt charakterisieren lassen:

1. **reaktiv:** Das Verhalten schließt eine passive Wahrnehmung und zeitnahe Reaktion auf Änderungen der Umwelt ein.

2. **pro-aktiv:** Über reaktives Verhalten hinaus entwickelt das System eine eigene Initiative zur Erreichung von Zielen, und nimmt Gelegenheiten zur Erfüllung anderer Ziele wahr.

3. **sozial:** Über pro-aktives Verhalten hinaus geht das System Interaktionen mit anderen Agenten zur Erfüllung eigener oder altruistischer Zwecke ein.

Die in dieser Arbeit betrachteten Verhalten sind zunächst auf reaktive und pro-aktive beschränkt. Soziale Effekte entstehen aus der zu untersuchenden Koordination, aber nicht aus den Verhalten selber, da diese unabhängig voneinander sein sollen. Soziale Verhalten sind zentraler Forschungsgegenstand in Multiroboteruntersuchungen, jedoch nicht für mehrere Verhalten in einem gemeinsamen Körper.

2.2 Agenten

Verhalten werden in physischen oder virtuellen Systemen realisiert, die als Agenten aufgefaßt werden sollen. Dabei werden nicht die aus den Wirtschafts- oder Sozialwissenschaften bekannten Agenten verstanden, sondern ein computergestütztes System, das in eine Umgebung eingebettet ist und autonom variable Aktionen ausführen kann, um Ziele zu erreichen [Jennings et al., 1998]. Ein solcher Agent bezieht Informationen aus und über seine Umwelt, verändert diese und verändert sich selber, besteht in seiner Umwelt fort und hat darin einen eingeschränkten Wahrnehmungs- und Aktionsbereich [Klügl, 2000].

2.2.1 Eigenschaften von Agenten

Im Kontext verhaltensbasierter Systeme besteht weitgehend Übereinstimmung darüber, daß ein Agent durch folgende essentielle Eigenschaften definiert wird:

Autonomie: Der Agent braucht keine Intervention, Anleitung, Kommunikation oder Instruktion. Er generiert Aktionen aus sich selbst heraus und bestimmt selber seine eigenen Zustände und Aktionen. Autonomie wird unterschieden in

1. operationelle Autonomie: Dieser grundlegende Aspekt beschreibt das Agieren ohne menschliche Intervention als Reaktion auf die Umwelt (z.B. starre Reflexe)
2. Verhaltensautonomie: Dieser fortgeschrittene Aspekt beinhaltet das selbstbestimmte Formen und Anpassen der eigenen Verhaltensprinzipien.

Situiertheit: Der Agent überwacht seine eigene Interaktion mit der Umwelt. Er nimmt sie wahr, agiert entsprechend den eigenen Zielen (Reaktivität) und bestimmt so seine nächsten Wahrnehmungen mit.

Körperhaftigkeit (Embodiment): (eine Art von Situiertheit, [Mataric, 2001]) Der Agent muß ein physisches System in der realen Welt sein, mit einem von der Umgebung abgegrenzten Körper, physisch oder virtuell, mit Sensoren und Effektoren zur Interaktion mit der Umwelt und mit den Einschränkungen durch den Körper.

Flexibilität: Der Agent setzt seine Aktionen variabel zur Erreichung der Ziele mit sinnvollem, kohärentem Verhalten [Jennings et al., 1998] ein.

Ferber stattet Agenten in [Ferber, 2001], neben [Weiß, 1999] einem der ersten Lehrtexte zu Agenten und Multiagentensystemen, mit ähnlichen und weiteren Eigenschaften aus:

Ein Agent ist eine physische oder virtuelle Entität

1. die selbständig in einer Umwelt agieren kann
2. die direkt mit anderen Agenten kommunizieren kann
3. die durch eine Menge von Absichten angetrieben wird (in Form von individuellen Zielen, Befriedigungs- und Überlebensfunktionen, die der Agent versucht zu optimieren)
4. die eigene Ressourcen besitzt
5. die fähig ist, ihre Umwelt wahrzunehmen (allerdings nur in einem bestimmten Ausmaß)
6. die nur eine partielle Repräsentation ihrer Umwelt besitzt
7. die bestimmte Fähigkeiten besitzt und Dienste offerieren kann
8. die sich ggf. selbst reproduzieren kann
9. deren Verhalten darauf ausgerichtet ist, ihre Ziele unter Berücksichtigung der ihr zur Verfügung stehenden Ressourcen und Fähigkeiten zu befriedigen und die dabei auf ihre Wahrnehmung, ihre internen Modelle und ihre Kommunikation mit anderen Agenten (oder den Menschen) angewiesen ist.

Mit diesen Eigenschaften können Agenten individuell beschrieben werden. Entscheidend für den Einsatz eines Agenten ist das von ihm erzeugte Verhalten, sodaß vor einer Interpretation der genannten allgemeinen Eigenschaften und Anforderungen zunächst das Verhältnis zwischen Agenten und Verhalten betrachtet werden soll.

2.2.2 Agenten und Verhalten

Jeder Agent kann in bestimmtem Maße schnell und flexibel reagieren, was als seine Reaktivität bezeichnet wird. Diese Reaktion ist wegen ihrer Geschwindigkeit selten optimal, geplant oder erklärbar. Diese Attribute kennzeichnen die Deliberativität des Agenten. Beide Eigenschaften stehen im Widerstreit, es besteht also ein Reaktivitäts/Deliberativitäts-Dilemma. Jeder Agent muß dieses Dilemma lösen und damit entscheiden, in welchem Verhältnis vom Designer vorgegebene, 'eingebaute' Informationen und zur Laufzeit angestellte Berechnungen zueinander stehen. Diese Balance bestimmt die kognitive Leitungsfähigkeit und, in entgegengesetzter Abhängigkeit, die Zeitnähe der Reaktionen eines Agenten. Unter den Agenten, die in verschiedenen Teilgebieten der Künstlichen Intelligenz entwickelt werden, läßt sich aus eigener Erfahrung und [Klügl, 2000, Ferber, 2001, Mataric, 2001] analog zu der Hierarchie der Verhalten im vorigen Abschnitt 2.1 folgende Hierarchie der kognitiven Verhaltensleistungen zusammenstellen.

1. **reaktiv:** (subsymbolisch, tropistisch, Reflex)[1] Ein Verhalten wird durch Verschaltung oder durch einen Kontroller fest vorgegeben (Situations-Aktions-Regeln, Braitenberg-Vehikel, unbedingte Reflexe). Der Agent verfügt nicht über ein Umweltmodell, sondern bildet direkt von Sensoren auf Effektoren ab. MATARIC verdeutlicht das mit dem Motto: 'Denke nicht, reagiere'.

2. **subkognitiv:** Auch dieser Agent verfügt nicht über ein Umweltmodell, verwendet jedoch Symbole, wenn auch einfachste (Zustandsindizes). Er zeigt kein pro-aktives oder planendes Handeln.

3. **operand:** (Verhalten, evtl. mit internen Zuständen) Der Agent reagiert unmittelbar auf Eingangsinformationen, wobei die Reaktion z.B. durch Pawlow'sche klassische Konditionierung (Hebb-Lernregel) auf der Grundlage reaktiven Verhaltens erlernt wurde. Er speichert sein Wissen über die Dynamik der Umwelt und über die Konsequenzen eigener Aktionen. Er ist leistungsfähiger in teilweise beobachtbaren Umwelten, da er über bessere Informationen zur Erzeugung von Situations-Aktions-Regeln verfügt. Sein Motto: 'denke wie du handelst'.

4. **deliberativ:** (Planung, ziel-basiert) Der Agent wägt die Ergebnisse und Nützlichkeit eigenen Handelns ab und speichert Umweltzustände bzw. ein internes Umweltmodell. Er verfügt über eine explizite Repräsentation seiner Ziele. Sein Motto: 'Denke gut nach, dann handle'.

   (a) **klassische Planer:** Sie stellen zunächst einen vollständigen Plan auf, den sie dann mechanisch abarbeiten.

[1] Die in Klammern angeführten Begriffe werden in der Literatur als alternative Bezeichnungen derselben Art von Agenten verwendet.

(b) **reaktive Planer:** Sie beachten ungeplante Veränderungen der Umwelt während der Ausführung des Planes und planen neu. Sie können ein erlerntes Verhalten auf weitere Ziele generalisieren.

5. **hybride:** Solche Agenten versuchen, reaktives und deliberatives Verhalten zu kombinieren, um eine Echtzeitreaktion mit Rationalität und Optimalität zu erreichen. Damit setzen sie die in Abschnitt 1.4 geforderte gegenseitige Ergänzung der beiden KI-Ansätze um. Ihr Motto: 'Denke und handle unabhängig voneinander, parallel'.

6. **kognitiv:** (Nützlichkeits-basiert, hysteretisch) Ein derartiger Agent verfügt zusätzlich über komplexe Situationsbeschreibungen und kann zwischen verschiedenen Wegen zu einem Ziel bzw. zwischen verschiedenen Zielen abwägen, wie bei ethologischen, psychologischen oder kognitionswissenschaftlichen Modellen.

Ein Agent bringt die Komponenten intelligenten Verhaltens zu einer Einheit zusammen, die nicht ohne jede einzelne dieser Komponenten funktionieren kann: Sensorik, Planung, Aktionsgenerierung, Aktuatorik. Damit ist ein Agent die grundlegende Steuerstruktur, die zur Umsetzung des sensomotorischen Paradigmas aus Abschnitt 1.4.4 betrachtet werden muß. Der Schwerpunkt in den folgenden Untersuchungen liegt auf den ersten drei, modellfreien Arten, die mit den Werkzeugen der Neuroinformatik generiert werden, obgleich damit auch Agenten mit Modellen erzeugt werden könnten. In dieser Arbeit werden allerdings nur die genannten Agenten betrachtet, da der Schwerpunkt auf der Untersuchung ihrer Koordination liegt. Im Sinne von [Ferber, 2001] sind RL-Agenten reaktive Agenten, da sie keine expliziten Absichten und keine gegenseitigen Vorstellungen von einander haben. Komplexere Agenten würden die Einschätzung der Koordination erschweren. Sie sollen sich auch nicht selber fortpflanzen. Ihre Modellierungs-, Kommunikations- und Dienstfähigkeiten sind stark eingeschränkt, da soziale Effekte der Agenten selber, wie oben beschrieben, nicht untersucht werden.

2.2.3 Animat

In dieser Arbeit soll ein mobiles autonomes System, wie ein Roboter oder ein Tier in einer einengenden Sichtweise, als ein *Animat* bezeichnet werden. Diese Wortschöpfung geht auf die Konferenzen 'From Animals to Animats' zurück [Cliff et al., 1994, Maes et al., 1996, Pfeifer et al., 1998, Meyer et al., 2000] und setzt sich aus 'animal' (Tier) und 'artefakt' (künstliches Wesen) zusammen. In diesem Sinne ist Animat eher als zusammenfassende Kategorie zu verstehen, die viele Eigenschaften mit einem Verhalten bzw. einem Agenten teilt. Der Begriff wird eingeführt, um ein reales oder simuliertes, natürliches oder technisches Gesamtsystem mit einem oder mehreren Agenten zu bezeichnen, das in einer Umwelt Verhalten zeigt. Mit Agenten ausgestattet, die reales Verhalten in einem körperhaften System erzeugen, wird es häufig mit einem Multiagentensystem verwechselt, ist jedoch nicht auf diese beschränkt, sondern wird als zusammenfassender Begriff für die erwähnten Gesamtsysteme verwendet.

Die folgenden idealen Eigenschaften eines Animats sind vorrangig Pfeifer entnommen [Pfeifer, 1996]. Sie untersetzen Brooks' Anforderungen an verhaltensbasierte Systeme (siehe Abschnitt 1.4.3) näher:

Vollständigkeit: Es müssen immer komplette Systeme und nicht einzelne Eigenschaften (wie Gedächtnis, Planen, Aktionsauswahl) betrachtet werden. Das Animat muß alle Voraussetzungen haben, sich in der realen Welt verhalten zu können, d.h. in der realen Welt bestimmte Aufgaben ohne menschliche Intervention ausführen können.

Selbstsuffizienz: Das Animat muß sich für einige Zeit selber versorgen können.

Ökologische Nische: Es gibt keine Universalität: Die Prinzipien für die Vollständigkeit sind nur in Bezug auf eine ökologische Nische sinnvoll. Daraus ergeben sich auch Randbedingungen für das Design des Animats (Sensoren, Aktuatoren etc.).

Parallele, lose gekoppelte Prozesse (Anti-Homunculus): Intelligenz bzw. Kognition entsteht aus einer großen Anzahl lose gekoppelter Prozesse, die asynchron und ohne oder mit sehr wenigen zentralen Ressourcen laufen. Kohärentes Verhalten kann ohne zentrale Aktionsauswahl erzielt werden.

Wert: Das Animat muß über ein Wertsystem und über eigenüberwachtes Lernen verfügen. Es muß selber wissen, was für es gut ist. Dies betrifft den expliziten Aspekt des Wertsystems. Der implizite Aspekt ist durch den Designer vorgegeben: welche Sensoren, Aktuatoren, Algorithmen etc.

Interaktion durch sensomotorische Koordination: Die Interaktion mit der Umwelt ist als sensomotorische Koordination aufzufassen. Diese umfaßt Sensoren, Aktuatoren und das ganze Animat. Entsprechend sind Perzeption, Klassifikation und Speicherung als sensomotorische Koordination und nicht als individuelle Module zu betrachten. Durch sensomotorische Koordination sind entstehende Konzepte automatisch gebunden (Symbolgrounding). Wiedererkennen eines Objektes wird zum Abspielen der sensomotorischen Koordination. Dieser Automatismus ist jedoch nicht zwangsläufig, sodaß es eine erfolgversprechende verhaltensbasierte KI nur mit sensomotorischen neuronalen Netzen mit Rekurrenz und internen Zuständen geben können soll.

Ökologische Balance: Es muß eine Übereinstimmung der Komplexitäten für Sensoren, neuronales Substrat (eingesetzte Verarbeitung für sensomotorische Abbildung) und Aktuatoren geben.

Billig: Zum ersten buchstäblich billiges Design: der Designer bzw. das Animat muß sich um seine Kinematik etc. kümmern, die Komponenten dürfen nicht zu gut sein, die Interaktion mit der realen Welt muß beachtet werden müssen (und nicht schon gelöst sein). Zum zweiten sollte nur eingebaut werden, was tatsächlich gebraucht wird. Zum dritten sollen hier die Besonderheiten der ökologischen Nische ausgenutzt werden.

Diese Zusammenstellung beschreibt maximale Anforderungen an ein Animat, das mit dem Ziel entworfen wird, in der natürlichen Umwelt real zu überleben. Von diesen Anforderungen werden in den ausgewählten methodischen Untersuchungen dieser Arbeit nicht alle umgesetzt. Dies betrifft insbesondere jene mit Bezug auf eine natürliche Umwelt, wie Selbstsuffizienz, ökologische Nische, ökologische Balance und Billigkeit. Die betrachteten Animats bewegen sich in realer, aber noch nicht in natürlicher Umwelt.

2.3 Erzielung von Verhalten

2.3.1 Verhalten ohne Lernen

In der weitaus überwiegenden Mehrzahl der in der Literatur beschriebenen Agenten oder Robotersteuerungen wird deren Aktionsauswahl durch feste Regeln und Modelle realisiert. Damit ist die operationelle Autonomie der Roboter gegeben, aber festgeschrieben. Das Animat kann nicht über die Anpassung seines Verhaltens bestimmen.

Zur Umsetzung können Algorithmen der Fuzzy Logik, der formalen Logik, der symbolischen KI sowie der Regelungs- und Automatisierungstechnik eingesetzt werden. Verhalten ohne Lernen setzt jedoch ein vollständiges Wissen über die Einsatzumgebung voraus, was dem Grundanliegen in dieser Arbeit widerspricht.

Aufgrund der fehlenden Adaptivität kann nur begrenzt auf Rauscheinflüsse und Nichtlinearitäten in der Einsatzumgebung eingegangen werden [Sofge and White, 1992]. Für nichtlineare, unbekannte Umwelten sind kaum theoretische Erkenntnisse und konstruktive Algorithmen für das Design von Reglern verfügbar [Narendra and Mukhopadhyay, 1994]. Bei einem *off-line* Design der Kontroller kann auch die Optimalität der Eingriffe im Einsatz nur begrenzt betrachtet werden[Wang et al., 1995]. Um die von der Regelungstechnik erzielten Erfolge auch adaptiven Agenten zugänglich zu machen, können die Regler als Verhalten betrachtet werden, die z.B. von adaptiven Verhalten benutzt werden können. Eine andere Möglichkeit ihrer Integration ist ihr Einsatz als Agent in einer der später betrachteten Koordinationen. Für einen solchen Einsatz müssen aus den Reglern jedoch die zur Koordination notwendigen Grössen, z.B. Bewertungen von Zustands-Aktionspaaren, gewonnen werden. Die Untersuchung solcher Möglichkeiten liegt jedoch ausserhalb dieser Arbeit.

2.3.2 Lernender Verhaltenserwerb

In einer unbekannten Umwelt ist Lernen unumgänglich, in bekannten Umwelten sinnvoll. Um die Designer von der zumindest schwierigen Aufgabe zu befreien, alle Eigenheiten der Umwelt zu berücksichtigen, sollte der Agent selbständig lernen. Alle durch den Agenten anzutreffenden Situationen vorherzusehen ist meist eine nicht zu bewältigende Aufgabe. Lernen macht den Agenten anpassungsfähig an sich verändernde Umwelten. Darüberhinaus ist Lernen nützlich oder notwendig, wenn nötiges Wissen zu teuer, nicht vorhanden, nicht zu extrahieren oder nicht zu kodieren/übertragen ist.

Auf diese Weise vergrößert Lernen den Bereich der Umweltbedingungen, unter denen ein gegebenes System einsetzbar ist und verringert die Menge an Information, die einem System mitgegeben werden muß, um sich in einer gegebenen Umwelt erfolgreich zu verhalten [Brooks and Mataric, 1993].

Lernen ist Voraussetzung für die oben erwähnte Verhaltensautonomie und Teil wie auch notwendige Voraussetzung von Intelligenz. Hier wird es als eine relativ permanente Änderung des Verhaltens im Ergebnis von Erfahrungen angesehen. Eine mehr ergebnisorientierte Definition beschreibt Lernen als Verbesserung der Performanz eines Agenten über die Zeit [Kim, 1994].

Das Lernproblem für einen Agenten kann in drei verschiedene Teilprobleme unterteilt werden, die der eingesetzte Lernalgorithmus lösen muß.

temporal: Welche Aktion zu welchem Zeitpunkt war verantwortlich für den gegenwärtigen Erfolg?

sensorisch-strukturell: Welche Merkmale der Umwelt sagen Erfolg voraus?

aktuatorisch: Welche Aktionsparameter erzeugten den Erfolg?

2.3.3 Anforderung an selbstlernende Agenten

Ein Lernsystem für ein Animat, das in realen Umwelten Intelligenz unter Beweis stellen soll, sollte möglichst viele der Anforderungen aus der folgenden Zusammenstellung erfüllen. Diese Anforderungen ergeben sich einerseits aus den oben eingeführten Eigenschaften eines Agenten, besonders seiner Autonomie, aber auch seiner Situiertheit in der realen, stochastischen, dynamischen Umwelt.

- **Lernen ohne und mit Lehrer:** Das Animat sollte ohne direkte Anleitung, Vorgaben oder Intervention von einem Lehrer oder Designer, mit den verfügbaren Informationen aus der Umwelt sinnvolles Verhalten erlernen können. Ist jedoch Wissen über solches Verhalten verfügbar, sollte es möglichst durch denselben Algorithmus aufgenommen werden können.

- **Lernen aus einmaliger Präsentation:** Manche Ereignisse, die für das Animat relevant sind, treten nur sehr selten auf. Trotzdem sollte der Lernalgorithmus sie nicht ignorieren oder mit einer unangemessenen Generalisierung aus häufig auftretenden Ereignissen reagieren.

- **Lernen wichtiger Ausnahmen bei einmaliger Präsentation und statistisches Lernen des Regelfalls:** Neben relevanten Ausnahmen muß natürlich auch ein Verhalten für den Regelfall erzeugt werden. Es sollten auch Verschiebungen der Statistik zwischen Ausnahme und Regel betrachtet werden können.

- **lebenslanges Lernen:** Eine natürliche, aber auch eine reale Umwelt ändern sich ständig, und das Animat muß sein Verhalten laufend anpassen können.

- **schnelleres Lernen, als Umwelt sich ändert:** Die ständige Anpassung des Verhaltens des Animats sollte schneller erfolgen können, als die relevanten Teile der Umwelt sich ändern.

- **sinnvolle Abwägung zwischen Neulernen und Vergessen:** Dieses sogenannte 'Stabilitäts/Plastizitäts-Dilemma' [Carpenter and Grossberg, 1991] beschreibt die Balance zwischen notwendigem Neulernen bisher nicht betrachteter oder geänderter Umwelt, und Vergessen von bisher gelerntem. Vergessen ist bei begrenzten Ressourcen notwendig, muß aber sinnvoll begrenzt werden, damit das Animat noch sinnvolles Verhalten reproduzieren kann.

- **Generalisierung von bekannten auf unbekannte Eingangsmuster:** Aufgrund der beschränkten Ressourcen eines Animats können nicht alle Ereignisse protokolliert werden. Daher sollten Eingangsmuster geeignet zusammengefaßt werden. Beim Auftreten neuer Ereignisse ermöglicht eine Generalisierung eine evtl. nicht optimale, aber zunächst nicht komplett falsche Reaktion.

- **ungleichmäßige Unterteilung des Zustandsraumes und deren adaptive Verfeinerung:** Eine Generalisierung kann durch Unterteilung des Zustandsraumes und Zuordnung geeigneter Reaktionen erzielt werden. Eine aufgabengerechte, adaptive Unterteilung aus der Interaktion mit der Umwelt ist dabei einer festen, vorgegebenen, gleichmäßigen vorzuziehen.

- **implizites oder explizites Planen:** Um von einem reaktiven, passiven Verhalten zu einem pro-aktiven Verhalten zu gelangen, sollte eine implizite Planung durch Bewertung einer ganzen Folge von Aktionen, oder ein explizites Abwägen der Konsequenzen und Bewertungen eigener Aktionen möglich sein.

- **Lernen der Beachtung eigener physischer Gegegebenheiten:** Ein Algorithmus, der allgemein den lernenden Verhaltenserwerb modellieren soll, sollte keine bestimmten Sensoren oder ein gegebenes Modell der eigenen Kinematik voraussetzen, sondern solche Voraussetzungen der Umweltinteraktion auch erlernen.

Nicht alle dieser wünschenswerten Eigenschaften können erfüllt werden. Das im nächsten Abschnitt eingeführte und in Kapitel 4 ausführlich dargestellte Reinforcement Lernen, das die Grundlage der weiteren Arbeiten bilden wird, realisiert jedoch ein Lernen ohne oder mit Lehrer, eine reaktive Planung, eine Beachtung eigener Kinematik und kann das temporale und aktuatorische Lernproblem lösen.

2.3.4 Erwerb von Verhalten mittels Reinforcement Lernen

Roboter, Maschinen allgemein werden eigentlich gebaut, um Aufgaben zu erfüllen. Allerdings muß ihnen dazu bisher auch noch der Weg zur Erfüllung der Aufgaben mitgegeben werden. Reinforcement Lernen eröffnet die Möglichkeit, der Maschine für Menschen gut faßbar mitzuteilen, was die Aufgaben sind, und sie den Weg zu deren Erfüllung selber finden zu lassen. Der Designer muß weder dem System seine eigenen, unter Umständen unangebrachten Vorstellungen über mögliche Umweltzustände und entsprechende Reaktionen aufzwingen, noch sich darum sorgen, alle möglichen Situationen vorherzusehen oder in einem Trainingsset statistisch ausgewogen zu präsentieren. Solches Umweltwissen kann dynamisch veränderlich oder kostspielig zu erlangen oder einfach nicht vorhanden sein kann. Das System erwirbt das notwendige Wissen und ihm entsprechende Abbildungen selber [Connell and Mahadevan, 1993c].

Seine Bedürfnisse und Motivationen beim Überleben können dem Animat, wie seinem biologischen Vorbild, zur Bewertung der Umweltzustandes und der Kosten von Aktionen dienen. Algorithmen aus dem Reinforcement Lernen (RL) sollen eingesetzt werden, um durch eine eigenständige Exploration im Aktionsraum unter der Maßgabe der Optimierung solcher Bewertungen und Kosten eine optimale Steuerung zu gewährleisten.

Diese Algorithmen können z.B. mittels neuronaler Netze ein implizites Modell der Umwelt, der eigenen Kinematik und der Aktionsstrategie lernen. Sie approximieren inkrementell die 'Dynamische Programmierung', die ihrerseits die *einzige* exakte und effiziente Methode zur Findung einer optimalen Aktionsstrategie über die Zeit in einer verrauschten, nichtlinearen Umgebung ist [Werbos, 1992b]. Ohne detaillierte Vorgaben kann einem RL-Agenten mittels der Aktions- oder Zustandsbewertungen das Ziel bzw. die zu lösende Aufgabe mitgeteilt werden. Der Agent kann eigene Abbildungen zwischen Eingängen (Zuständen) und Führungsgrößen (Aktionen) finden, seine Lösungen ständig verbessern und sich so veränderten Bedingungen anpassen.

Nach [Werbos, 1992a] hat nur das Reinforcement Lernen das Potential, die Fähigkeiten des menschlichen Hirns nachzubilden, online zu lernen, robust gegen Störungen (Rauschen, kleine, evtl. statistische Schwankungen) mehrere parallele Inputs auszuwerten und darauf aufbauend, Aktuatoren zielgerichtet und darüberhinaus planend einzusetzen.

Die Algorithmen des RL ermitteln jedoch mehr als nur die beste Aktion für jeden Zustand: eine Aktionsbewertung für jede Aktion in jedem Zustand. Aufbauend auf diesen Aktionsbewertungen können Agenten mit verschiedenen Teilaufgaben koordiniert werden.

2.4 Verhaltenskoordination

Verhaltensbasierte Systeme gehen davon aus, daß zur Erfüllung verschiedener Bedürfnisse unterschiedliche, spezialisierte Verhalten existieren oder erlernt wurden. Da diese Verhalten jeweils nur Teile des gesamten Aufgabenumfanges des Animats lösen, besteht eines der grundlegenden Probleme für verhaltensbasierte Systeme darin zu bestimmen, wann welches Verhalten aktiv werden soll [Brooks and Maes, 1990]. Die einzelnen Verhalten nehmen dabei eine größere Zeitspanne zur Ausführung in Anspruch als primitive Aktionen. Sie interagieren mit anderen Verhalten über die Umwelt anstatt intern im Animat. Eine serielle Ausführung der Verhalten kann nur eine Form der Koordination der Verhalten darstellen, sodaß ein eigener Mechanismus für die Verhaltenkoordination notwendig wird.

Das Problem der Entscheidung für ein Verhalten aus mehreren relativ einfachen Einzelverhalten entstammt den biologischen Wissenschaften und wird dort von der Ethologie, der Verhaltensbiologie, behandelt. Dort ist es als 'action selection problem' (ASP) bekannt. Der Name wurde von Tyrrell verbreitet und von Humphrys strikt von dem Problem der Auswahl einer Aktion in einem bestimmten Zustand abgegrenzt. Diese Namensgebung ist allerdings irreführend. Liu et al. verwenden 'behavior selection problem' [Liu and Wu, 2001], was angebrachter erscheint.

Ausgehend von Tyrrell's Arbeiten wurden auf Grundlage der SAB-Konferenzen [Cliff et al., 1994, Maes et al., 1996, Pfeifer et al., 1998, Meyer et al., 2000] aus verschiedenen Beiträgen Anforderungen an den Koordinationsalgorithmus eines Animat zusammengestellt:

reelwertig: Die Sensoren arbeiten mit der kontuinierlichen, reelwertigen Realität

Sensibilität: Die Steuerung benutzt alle verfügbaren Informationen.

Behandelt alle Unterprobleme: Es soll nur einen Mechanismus geben, der für alle Aktionen zutrifft.

Persistenz: Das Animat führt ein konsumierendes Verhalten fort, auch wenn das entsprechende Bedürfnis schon geringer ist als andere. Eine zusammenhängende Aktionsfolge wird auch zu Ende geführt.

Priorisierung: Die Rangfolge der Verhalten entspricht den aktuellen Bedürfnissen.

Konsum: Das Animat zieht konsumierendes Verhalten einem appetitiven vor. Es stillt lieber ein Bedürfnis als der Stillung eines anderen näher zu kommen.

Notwendige Unterbrechungen: Eine Aktionsfolge muß durch eine wichtigere unterbrochen werden können.

Opportunismus: Konsumierendes Verhalten, das nicht zu der gerade ausgeführten Aktionsfolge gehört, kann ausgeführt werden, wenn es unerwartet sinnvoll wird.

einfache Erweiterbarkeit: Neue Verhalten können ohne große Störungen bestehender Verhaltenskoordination integriert werden.

Adaptivität: Die Auswahl einer Aktion in einem bestimmten Zustand kann durch Lernen geändert werden.

reale Aktion: Die Steuerung erzeugt direkt nutzbare Ausgabesignale.

Trigger: Das Animat agiert nicht ohne Umweltstimuli aus interner Motivation, d.h. es versucht nicht zu essen, wenn es hungrig, aber kein Futter vorhanden ist.

Diese Funktionen müssen erfüllt werden, um das Verhalten eines Tieres in einer realen oder natürlichen Umwelt modellieren zu können. Entsprechend muß sich eine Koordination mehrerer Verhalten daran messen lassen, wie sie diese Forderungen erfüllt.

Um die entstehenden, komplexen Verhalten erzeugen zu können, muß ein Koordinationsalgorithmus für Animats folgende **Anforderungen an die Verhaltenskombination** [Dorigo and Colombetti, 1997] zwischen den einzelnen Verhalten realisieren können:

Sequenz: Die erfolgreiche Ausführung eines Verhaltens ist Voraussetzung für ein anderes.

unabhängige Summe: Aktionen, die unabhängig auf unterschiedlichen Effektoren erfolgen, werden parallel ausgeführt.

Kombination: Die Verhalten unterstützen sich gegenseitig; es entsteht eine Kooperation zwischen ihnen.

Unterdrückung: Gegensätzliche Verhalten schliessen sich aus, sodaß zwischen ihnen ein Wettkampf bzw. eine Konkurrenz entsteht.

Verhandlung: Durch Verhandlung zwischen den Verhalten entstehen Zwischenformen oder Kombinationen von Kooperation und Konkurrenz.

Hierarchie: Mehrere Verhalten können zu einer festen Folge kombiniert werden.

TYRRELL vergleicht in [Tyrrell, 1993] sieben solcher Koordinierungsmechanismen ('action selection mechanism', ASM) anhand einer simulierten Umgebung und entwickelt den besten weiter zu einer eigenen Architektur. Alle untersuchten Mechanismen sind jedoch sehr stark biologisch motiviert und auf Robotern mit komplexen Aufgaben schwer umsetzbar.

Durch TYRRELL motiviert, entwickelte HUMPHRYS *W-Lernen* [Humphrys, 1997], einen ASM, der die Informationen nutzt, die Agenten des Reinforcement Lernens sowieso erlernen. Es realisiert adaptiv die genannten Aktionskoordinationen für jeden Zustand des Gesamtsystems, ausgenommen die der parallelen Aktionen in der unabhängigen Summe, und erfüllt die genannten Anforderungen an den Koordinationsmechanismus eines Animats. Die Erfüllung von 'Trigger' und 'Opportunismus' setzt die Existenz zeitlich veränderlicher, beeinflußbarer Bedürfnisse des Animats und deren Verwendung in seiner Zustandsbeschreibung voraus. W-Lernen als Mechanismus zur adaptiven Verhaltensauswahl bildet den zentralen Untersuchungsgegenstand dieser Arbeit und wird in Abschnitt 5.2 detailliert eingeführt.

Das Animat zeigt also Verhalten, die aus verschiedenen Bedürfnissen entstehen. Die Agenten, die ein solches Verhalten zur Bedürfnisbefriedigung umsetzen, müssen koordiniert werden, damit das Animat sinnvoll seine Bedürfnisse stillt. Die Realisierung einer Verhaltenskoordination aber muß als eine von mehreren möglichen Umsetzungen eines sogenannten Multiagentensystems angesehen werden. Daher werden diese im folgenden allgemein charakterisiert. Für eine Einordnung der Verhaltenskoordination in den Gesamtrahmen der Multiagentensysteme werden auch andere mögliche Realisierungen dargestellt.

2.4.1 Multiagentensysteme

In einem Multiagentensystem (MAS) teilen mehrere Agenten eine Umwelt oder einen bestimmten Teil davon, sie interagieren miteinander bei der Verfolgung eigener oder gemeinsamer Interessen. Dabei kann jeder Agent seine spezifische Sicht auf die Umwelt haben. Die Agenten werden nicht zentral gesteuert, das Verhalten des Gesamtsystems entsteht aus der Interaktion der Einzelverhalten und sollte Charakteristika aufweisen, die in keinem der Einzelverhalten enthalten sind, sondern erst aus der Interaktion der Agenten erwachsen (emergentes Verhalten).

Folgende Eigenschaften kennzeichnen ein MAS [Klügl, 2000, Jennings et al., 1998]:

- Jeder Agent besitzt nur unvollständige Informationen oder begrenzte Problemlösungsfähigkeiten.
- Jeder Agent verwaltet seine Daten lokal, die Datenhaltung ist also dezentral.
- Jeder Agent führt seine Berechnung lokal, parallel und asynchron aus.
- Im Idealfall fehlt eine zentrale Kontrolle.

Insofern muß das Animat als ein Multiagentensystem angesehen werden: es existieren mindestens zwei Experten, die gemeinsam an einem Ziel arbeiten, bei dem sie für einen Teil

verantwortlich sind. Einerseits tragen sie durch ihre Mitarbeit förderlich bei, andererseits aber tun sie das wegen ihrer Teilverantwortung nicht immer oder mit wechselndem Erfolg. Ihr eigener Erfolg und Beitrag zum Gesamterfolg hängt zumeist vom Zustand der Umwelt oder des Gesamtsystems in der Umwelt ab. Bei dem Problem der Verhaltenskoordination für ein Animat konkurrieren die Experten jedoch gleichzeitig auch um den Zugriff auf die nur einmalig vorhandene Ressource 'Aktuatorik' des Animats.

Vor einer Betrachtung verschiedener Umsetzungsmöglichkeiten eines MAS bezüglich dieser Randbedingungen muß diese verteilte Erfüllung einer Gesamtaufgabe der Realisierung als ein einizges grosses, komplexes Verhalten gegenübergestellt werden. Auch ein Animat kann ein solches komplexes Verhalten als einen monolithischen Agenten erlernen.

2.4.2 Vor- und Nachteile monolithischer bzw. verteilter Realisierungen

Da eine Koordinierung von Agenten mit erheblichem Aufwand hinsichtlich der Aufgabendekomposition, der Aktionssynthese, aber auch hinsichtlich der Berechnungen für die einzelnen Agenten verbunden ist, erhebt sich zunächst die Frage nach den Vorteilen einer Realisierung als MAS gegenüber einer Realisierung als ein einziger, monolithischer Agent.

Ein solcher Monolith muß folgende Funktionen erfüllen, um alle Teilaufgaben zu lösen:

- Lösung der Gesamtaufgabe in nur einer einzigen Struktur
- Verkettung aller notwendiger sensorischer Eingangsquellen
- Ansteuerung der gesamten Aktuatorik
- Erwerb einer einzigen Abbildung vom gesamten sensorischen Input auf die Aktuatorik
- Zustandsraum wächst unter Umständen exponentiell mit der Anzahl der Aufgaben/Agenten

Diese Funktionen ergeben für eine monolithische Lösung gravierende Probleme hinsichtlich der Trainingszeit. Wegen der notwendigen grossen, komplexen Abbildung zwischen Sensorik und Aktuatorik müssen viele Erfahrungen mit der Umwelt erlebt werden. Die Grösse des abzubildenden Zustandsraumes führt zu Einschränkungen hinsichtlich seiner Repräsentation, so ist eine schnell zu berechnende Tabelle der verschiedenen Zustände nur für kleine Probleme sinnvoll.
Bei dem Training werden die verschiedenen Bewertungsquellen für die unterschiedlichen Aufgaben überlagert. Diese Überlagerung macht den Entwurf der Bewertungen zur Erreichung des gewünschten Verhaltens kompliziert. Es kann bei einer unglücklichen Wahl zu einer Unterrepräsentation bestimmter Aufgaben kommen, die schlecht oder gar nicht mehr lernbar werden. Teilaufgaben, für die Bewertungen vergeben werden, die sich um Grössenordnungen von denen der anderen Aufgaben unterscheiden, könnten komplett ignoriert werden. Dies kann bei einer Ermittlung der Gesamtbewertung als Summe der Teilbewertungen auftreten.
Besondere Probleme jedoch entstehen bei Veränderungen in der Zusammenstellung der

Teilaufgaben. Für jede Änderung der betrachteten Teilaufgaben muß die gesamte Abbildung von der Sensorik auf die Aktuatorik neu gelernt werden. Bei der Hinzunahme neuer Teilaufgaben kann der Zustandsraum exponentiell wachsen. Dies verursacht Adaptationszeiten, die einen sinnvollen Einsatz von Monolithen stark einschränken.

Bei einer Realisierung derselben Gesamtaufgabe als Multiagentensystem können viele dieser Probleme umgangen werden. Die Anwendung von MAS wird allgemein durch folgende Vorzüge [Kim, 1994, Lesser, 1999] motiviert:

leichte Lösung inhärent verteilter Probleme: Wenn zum Beispiel Daten oder Steuerungen physisch verteilt sind, bietet sich eine verteilte Lösung an (z.B. Pumpstationen einer Ölleitung, ein Schwarm von Erkundungsrobotern)

robusteres Verhalten: Durch die spezifische Bearbeitung des Teilproblems kann das Verhalten stabiler gestaltet werden, und durch eine Redundanz zwischen den Agenten das Gesamtverhalten fehlertoleranter gegenüber Ausfällen einzelner Agenten.

Skalierbarkeit durch leichte Erweiterbarkeit: Neue Teilaufgaben können dem Gesamtsystem leicht hinzugefügt werden.

leichteres Retraining in veränderlichen Umwelten: Da meist nur Subagenten angepaßt werden müssen, kann sich das Gesamtsystem leichter an veränderliche Umwelten anpassen.

Parallelisierbarkeit durch unabhängige Agenten: Da die Agenten unabhängig voneinander agieren, können ihre Operationen parallelisiert werden.

Verwendbarkeit der Agenten in unterschiedlichen MAS: Bei entsprechender Modularisierung und Systementwurf können die Agenten in verschiedenen Systemen verwendet werden.

einfachere Zustandsrepräsentation pro Agent: Meist braucht jeder Agent nur einen Teil des gesamten Zustandsraumes, sodaß er auch mit einer einfacheren Repräsentation funktioniert.

leichter zu entwerfende Bewertungsfunktion: Die Teilaufgaben sind meist leichter zu formulieren und zu bewerten, da nicht alle Interaktionen zwischen Umweltmerkmalen und Teilaufgaben betrachtet werden müssen.

Ausnutzung von Synergieeffekten: Durch Ausnutzung unterschiedlicher Eingangsinformationen, verfügbarer Aktionen oder qualitativ unterschiedlicher Verhalten können Agenten sich gegenseitig unterstützen.

Beschleunigung durch simultanes Berechnen: Durch die Aufteilung der Aufgaben können unabhängige Berechnungen parallel ausgeführt werden, was zu einer Beschleunigung führt.

weniger Kommunikation: Bei inhärent verteilten Problemen brauchen sensorische Informationen und Steuerkommandos nicht mehr mit dem zentralen Kontroller kommuniziert werden, da die Verarbeitung im Agent näher an Informationsquelle erfolgt. Allerdings kann die Koordination eine Kommunikation erfordern.

verbesserte Reaktivität: Durch höhere Spezifität der verteilten Steuerungen und entfallenden Kommunikationsbedarf können die lokalen Agenten schneller reagieren, da Wahrnehmung, Verarbeitung und Aktionsgenerierung am gleichen Ort erfolgen.

einfacherer Systementwurf durch Modularität: Durch die Zerlegung in Teilprobleme lassen sich die einzelnen Agenten leichter entwerfen, die Zuordnung von Sensorik, Aktuatorik und notwendigen Verarbeitungsstrukturen ist einfacher.

Für eine Realisierung als MAS müssen allgemein folgende Punkte [Jennings et al., 1998] gelöst werden, die im Vergleich zum Monolithen zusätzliche Aufgaben umfassen:

- Zerlegung des Gesamtproblems in Teilaufgaben
- Festlegung der Art der Synthese der Gesamtlösung aus Teillösungen
- Festlegung der Spezifik der Agenten (Sensorischer Input, Motorik, Reward)
- Implementation von Kommunikation, -sprache, -protokoll der Agenten
- Sicherstellung der Kohärenz der Aktionen zwischen Agenten
- Festlegungen über Art und Erwerb des Wissens der Agenten über andere Agenten
- Koordination der Agenten
- Balance zwischen Aktion und Kommunikation der Agenten
- Bewertung des Gesamtverhaltens

Die zusätzlichen Aufgaben betreffen vor allem die Aufteilung der Gesamtaufgabe auf einzelne Agenten, die Kommunikation der Agenten und ihre Koordination beim Finden auszuführender Aktionen. Für ein Animat ergibt sich die Dekomposition aus der Separierbarkeit der Verhalten und der Zweckmäßigkeit einer Separierung. In den Untersuchungen in der vorliegenden Arbeit werden die Problemdekomposition, die Festlegung der Agenten und die Bewertung des Gesamtverhaltens durch den Designer ausgeführt. Die Agenten haben kein Wissen übereinander, kommunizieren nicht miteinander und die Balance zwischen Aktion und Kommunikation liegt auf der Aktion. Die Lösungssynthese wird durch die Koordination der Agenten übernommen.

2.4.3 Organisation von MAS

Nach einer Zerlegung der Gesamtaufgabe in Teilaufgaben, die meist gegeben oder vom Designer übernommen wird, ist die Festlegung einer Organisation für das Multiagentensystem eine der ersten Aufgaben. MAS können nach einer Vielzahl unterschiedlicher Organisationsformen konzipiert werden, von denen hier drei grundlegende vorgestellt werden sollen [Ferber, 2001]: die modular horizontale Organisation, das Blackboard und die Subsumption-Architektur als eine modular vertikale Organisation.

1. **Modular horizontale Organisation:** Diese Architektur weist einzelne, sequentiell angeordnete Module für Wahrnehmung, Kommunikation, Modellierung der Umwelt und anderer Agenten, Verwaltung der Verpflichtungen, Wissen über eigene Fähigkeiten, Zielmanagement und Entscheidungsfindung und Aktionsplanung auf. Alle Verbindungen zwischen ihnen sind fest und der Informationsfluß vom Designer vorgegeben. Informationen werden aus der Umwelt aufgenommen, abstrahiert, durch Modelle verarbeitet, in der Entscheidungsfindung mit Zielen in Beziehung gesetzt, ein Plan wird generiert und entsprechende Aktionen ausgeführt.

   Eine solche Architektur widerspricht mit ihrer grundlegenden funktionalen Modularisierung der Idee von Verhalten mit vollständiger, minimaler Abbildung zwischen Sensorik und Aktuatorik und kann für ein verhaltensbasiertes Animat nicht eingesetzt werden.

2. **Blackboard:** Blackboards bestehen aus Wissensquellen/Experten, einer gemeinsamen Datenstruktur (dem eigentlichen Blackboard) und einer Kontrolleinheit, die selber als ein zweites Blackboard realisiert ist. Die Wissensquellen oder Spezialisten beobachten das Blackboard, auf dem zunächst das Problem und initiale Daten erscheinen. Die Spezialisten veröffentlichen ihre Beiträge auf dem BlackBoard, wenn ihre Voraussetzungen erfüllt sind. Die Kontrolleinheit aktiviert Spezialisten und löst Konflikte zwischen ihnen. Der daraus entstehenden Vorteil ist die hohe Flexibilität der Module, der Nachteil die Ineffizienz aus dem hohen Kontrollaufwand. Blackboardsysteme sind durch folgende Eigenschaften [Huhns and Stephens, 1999] gekennzeichnet:

   Unabhängigkeit der Expertise: Jeder Spezialist ist unabhängig von der aktuellen Gruppe in anderen Gruppen arbeitsfähig.

   Vielfalt der Problemlösungstechniken: Interne Repräsentationen und Entscheidungsmechanismen der Experten sind nur ihnen selber zugänglich.

   Flexible Repräsentation der Blackboardinformationen: Das Blackboard hat keinerlei Einschränkungen bezüglich der darauf platzierten Information.

   Gemeinsame Interaktionssprache: Die Spezialisten verstehen die informationen, die von anderen Spezialisten auf dem Blackboard hinterlegt wurden.

   Ereignisbasierte Aktivierung: Die Spezialisten werden durch Ereignisse auf dem Blackboard und in der Umwelt aktiviert. Der Spezialist informiert das Blackboard, über welche neuen Ereignisse, Änderungen oder Löschungen er informiert werden will.

   Kontrollbedürfnis: Die Kontrolleinheit steuert die Lösung des Gesamtproblems. Es schätzt die Beiträge der verschiedenen Spezialisten anhand der Qualität (der Konfidenz) und der 'Realisierungskosten' jedes angebotenen Beitrags ein und aktiviert die jeweils sinnvollsten.

   Inkrementelle Problemlösung: Die Spezialisten tragen durch Verfeinerung, Widerspruch oder ganz neue Lösungslinien zur Gesamtlösung bei.

   Ein Blackboard ist eine sehr leistungsfähige, aber auch aufwendige Organisationsform für ein Multiagentensystem. Es erlaubt eine sehr flexible und auch adaptive

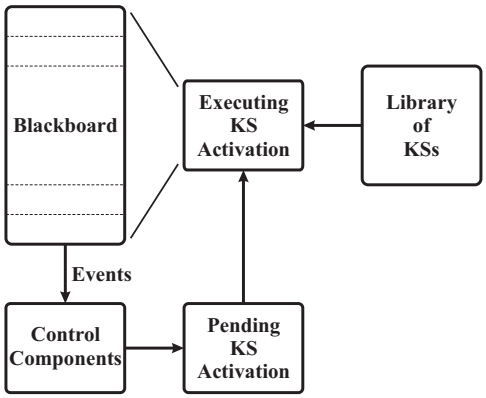

Abbildung 2.1: ***Struktur eines Blackboards.*** *Blackboard-Systeme bestehen nach* CORKILL *[Corkill, 1991] dem Blackboard, einer Kontrollkomponente sowie den Spezialisten oder 'Knowledge Sources' (KS). Das Blackboard ist die globale Datenbasis des Gesamtsystems. Es wird als Datenstruktur zur Repräsentation des Problemlösungszustandes, als Kommunikationsmedium sowie als temporärer Datenspeicher benutzt. Die Knowledge Sources sind die eigentlichen Spezialisten des Systems. Sie werden von der Kontrollkomponente benachrichtigt, sobald auf dem Blackboard Bedingungen für ihre Aktivität erfüllt werden.*

Koordination der Agenten. Der Aufwand und das Wissen, das der Designer beim Entwurf eines Blackboards investieren muß, scheinen für das gegebene Problem der Koordination einzelner Verhalten eines Tieres nicht angebracht und ethologisch schlecht vertretbar. Nach angemessener Reduktion der Komponenten des Blackboards könnte das in dieser Arbeit favorisierte W-Lernen als ein sehr einfaches Blackboard angesehen werden.

Subsumption: Ein Beispiel für eine vertikale Modularisierung ist BROOKS' *Subsumption*-Architektur [Brooks, 1986]. Diese Architektur ist auch die erste, die auf den in Abschnitt 2.1 eingeführten Verhalten basierte. Jedes Verhalten bildet in der Architektur eine Schicht, die wie alle anderen Schichten sowohl mit der Sensorik als auch der Aktuatorik verbunden ist. Jede der Schichten ist für ein bestimmtes Verhalten als Teil des Gesamtverhaltens verantwortlich. Alle Verhalten arbeiten parallel, aber nur eines kann die Aktuatorik kontrollieren. Für den Fall, daß widersprüchliche Verhalten auf die Aktuatorik zugreifen wollen, muß der Designer feste Dominanzbeziehungen festlegen. So entsteht eine Hierarchie von Verhalten, bei der übergeordnete Verhalten die Ausgaben untergeordneter Verhalten unterdrücken. Das Gesamtverhalten entsteht dann selbständig aus allen Verhalten, sein zielgerichtetes Design setzt die Vorausschau aller Situationen und Interaktionen voraus, aus der die hierarchischen Verhältnisse unter den Verhalten abgeleitet und kodiert werden müssen.

Ein Vorteil der Subsumption-Architektur besteht darin, daß es immer ein Verhalten gibt, das die Steuerung des Animats übernimmt, ein anderer besteht in der Reaktivität des Systems. Obwohl die Hierarchie eine naheliegende und wichtige Koordination von Verhalten ist, muß sie als eine Einschränkung der möglichen und nötigen Koordinierungen bezeichnet werden. Auch aus dieser Motivation heraus entwickelte HUMPHRYS das *W-Lernen*, das von gleichberechtigten Agenten ausgeht, die ihre gegenseitigen Beziehungen, darunter auch eine Hierarchie, für jeden Zustand in der

Umwelt adaptiv ermitteln. Dadurch brauchen nicht alle möglichen Umweltsituationen vorhergesehen werden wie bei einer Subsumption-Architektur. Eine Anpassung an veränderliche Umwelten und ein Einfügen neuer Agenten in die Architektur sind wesentlich einfacher.

2.4.4 Kategorisierung von MAS

Die Koordinierung der verschiedenen Agenten eines MAS kann als 'managing of dependencies', als Verwaltung von Abhängigkeiten [Schumacher, 2001], aufgefaßt werden. Dabei sollten Aspekte der Koordinierung zwischen den Agenten streng organisatorisch getrennt werden von Aspekten der Koordinierung innerhalb eines Agenten. Dies schließt eine gegenseitige Beeinflußung der beiden Koordinationsarten nicht aus, sie sind im Gegenteil abhängig voneinander. Die Koordination innerhalb eines Agenten wird als subjektive Koordination bezeichnet und beinhaltet die Art des Beitrags eines Agenten zum gemeinsamen Handeln: explizite oder implizite Koordinierung durch Planung, Verhandlung oder durch eine vorgegebene Organisation. Die Koordination zwischen den Agenten wird objektive Koordination genannt und umfaßt Kommunikation und Interaktion der Agenten sowie die Generierung und Destruktion von Agenten.

Aus der Vielzahl verschiedener Möglichkeiten für das Zusammenspiel objektiver und subjektiver Koordination ergiben sich mehrere Kriterien, nach denen Multiagentensysteme unterschieden werden können. In der Literatur ist eine grosse Anzahl verschiedener MAS zu finden, unter denen jedoch die wenigsten adaptiv sind. Die überwiegende Mehrzahl behandelt das Multiroboterproblem, in dem mehrere physische Roboter oder Animats koordiniert werden müssen.

Die Vorstellung von MAS soll mit Möglichkeiten der Kategorisierung der Vielzahl der Ansätze abgeschlossen werden:

1. Unterscheidung nach Übereinstimmung zwischen Agenten und Animats

   - jedes Animat beherbergt genau einen Agenten, die Umwelt weist mehrere Animats auf (Mehrkörpersystem)
   - ein Animat beherbergt mehrere Agenten (Einzelkörpersystem)

   Der wichtigste Unterschied zwischen MAS in einem und mehreren Körpern liegt in der Existenz einer theoretischen Grundlage zur Koordinierung mehrerer physischer Agenten (siehe Abschnitt 5.1.1). Aktionen in Multiroboterszenarien bzw. Mehrkörpersystemen *können*, aber müssen im Gegensatz zu Einzelkörperssystemen nicht koordiniert werden. Sollten die Agenten lernfähig sein, kann das Szenario für einzelne von mehreren Robotern instationär und nur teilweise beobachtbar werden. In Mehrkörpersystemen können die Rollen der Roboter dynamisch festgelegt werden, in Einzelkörpersystemen sind sie eher festgeschrieben.

2. Unterscheidung nach Prioritäten

   - hierarchisch organisierte Agenten
   - gleichberechtigte Agenten

Auf die Unterscheidung zwischen hierarchisch und gleichberechtigt organisierten Agenten wurde bei der Subsumption-Architektur eingegangen. Eine adaptive Gleichberechtigung kann eine Hierarchie einschliessen und ist daher vorzuziehen.

3. Unterscheidung nach gegenseitigem Verhältnis [Sen, 1997]

   - kompetitiv: jeder Agent erhält seinen eigenen Reward,
   - kooperativ: alle Agenten erhalten denselben Reward
   - sowohl kompetitiv als kooperativ möglich (je nach Umweltzustand o.ä.)

   Auch hier ist wegen der grösseren Flexibilität im Interesse der Gesamtaufgabe, eine adaptive Festlegung zwischen Kooperation und Konkurrenz vorzuziehen.

4. Unterscheidung nach Auflösung inkonsistenter Teillösungen [Durfee et al., 1989]

   Verhandlung: beinhaltet eine Aufteilung der Aufgaben an unabhängige Problemlöser von oben nach unten in einer Hierarchie

   funktional: bedeutet einen Austausch von Lösungsvorschlägen von unten nach oben

   Multi-Agenten Planung: beginnt mit Informationsaustausch zur Erstellung eines Kooperationsplans, dann Verteilung und Abarbeitung des Plans.

   Lokales Wissen: Lokale Problemlösung bezieht Betrachung über Aktionen und Beliefs anderer Agenten mit ein; Koordinierung ist Teil der Teillösungen und keine separate Struktur mehr

5. Unterscheidung nach Kontrollstrategie: Auswahl der resultierenden Aktion mittels

   - zentraler Kontrollstrategie
     - Koordinierung wird von zentraler Struktur ausgeführt
     - Agenten haben kein Wissen über die gegenseitigen Ziele
     - Funktion bzw. Rollen der Agenten werden von Zentralstruktur bestimmt (oder a priori)
     - nur eine zentrale Struktur hat Wissen über die Agenten
     - Agenten liefern Aktionsvorschläge – zentrale Struktur realisiert die Auswahl der resultierenden Aktion
   - verteilter Kontrollstrategie
     - Koordinierung ist Teil der Agentenfunktionen
     - Agenten haben Wissen über gegenseitige Existenz
     - Agenten verhandeln über resultierende Aktion
     - Rollenzuordnung durch [Hauskrecht et al., 1998a]
       * Konventionen, die eine koordinierte Aktion erzwingen
       * Kommunikation zwischen den Agenten vor der Aktionsauswahl
       * Lernen über wiederholte Interaktionen

Eine zentral koordinierte Architektur ist störanfälliger und weniger fehlertolerant als eine dezentral koordinierte, bei der jedoch Kommunikation notwendig wird, damit die Agenten über die Lösung des Ressourcenkonflikts verhandeln können. KIM stellte in [Kim, 1994] einen Vergleich (Tab. 2.1) zwischen zentralen und dezentralen MAS zusammen.

Kriterium	Zentral	Dezentral
Komplexität	wächst kombinatorisch mit Anzahl der Agenten	konstant bei unabhängigen Agenten, wächst bei kommunizierenden oder gegenseitig modellierenden Agenten
Reaktionszeit	wächst mit Anzahl der Agenten, Kommunikation mit Zentrale ist Flaschenhals	Agenten können eigene Entscheidungen treffen, daher konstant. Wächst bei Kommunikation der Agenten
Robustheit	Ausfall der Zentrale stoppt das ganze System	andere Agenten können weitermachen und u.U. die Funktionen ausgefallener übernehmen
Effizienz	Zentrale überblickt einzelne Agenten nur in einfachen Systemen, kaum lokale Flexibilität;	Individuelle Agenten können sich an lokale Gegebenheiten anpassen
	Zentrale kann Effekte einzelner Agenten auf Gesamtsystem abschätzen	Einzelagenten können Gesamtsystem stillegen oder müssen kommunizieren oder einander modellieren

Tabelle 2.1: *Vergleich zentraler mit dezentralen MAS*

W-Lernen ist als eine Koordination zu charakterisieren, bei der mehrere Agenten in einem Animat koordiniert werden, die sowohl gleichberechtigte als auch hierarchisch organisierte Beziehungen unter den Agenten lernen kann. Mit ihr können die Agenten sowohl kooperieren als auch konkurrieren. Eine zentrale Struktur ermittelt die resultierende Aktion aus einer top-down Verhandlung. Damit ermöglicht sie flexible Beziehungen unter den Agenten, verursacht durch die zentrale Koordinierung Kommunikationskosten und einen Schwachpunkt beim Ausfall der Koordination. W-Lernen erlaubt eine einfache Integration neuer Agenten, aber eine steigende Anzahl von Agenten vermehrt den Berechnungsaufwand der Koordination erheblich, womit die Reaktionszeit steigt.

2.5 Distributed Artificial Intelligence

Die Vorstellung von Multiagentensystemen soll durch die Erwähnung der entsprechenden Konzepte der Künstlichen Intelligenz abgerundet und vervollständigt werden.

Der Begriff der Multiagentensysteme selbst ist allgemein eigentlich weniger durch ethologische, sondern sehr stark durch Inhalte aus der Verteilten Künstlichen Intelligenz geprägt. Obwohl nur wenige lernende Ansätze verfolgt werden und die meisten Algorithmen von Mehrkörpersystemen ausgehen, sollen wichtige Begriffe des Gebietes kurz umrissen werden. Einige Konzepte der Multiagentensysteme, die in den vorangegangenen Abschnitten diskutiert wurden, ordnen sich dadurch in eine Reihe alterantiver Ansätze ein (z.B. hinsichtlich der Kommunikation oder Interaktion zwischen Agenten). Es lassen sich sicher einige Ideen und Techniken auf ethologische Verhaltenskoordination übertragen, jedoch erschweren wesentlich unterschiedliche Voraussetzungen und Randbedingung die Identifikation und Übertragung.

Die Koordinierung mehrerer Agenten zur Erfüllung einer gemeinsamen Aufgabe oder beim Zugriff auf gemeinsame Ressourcen wird in letzter Zeit verstärkt und durch das WWW gefördert, in dem Zweig der Künstlichen Intelligenz bearbeitet, der 'Distributed Artificial Intelligence' (DAI, Verteilte Künstliche Intelligenz, VKI) genannt wurde.

Bis vor wenigen Jahren wurde die Verteilte Künstliche Intelligenz weiter unterschieden in 'Distributed Problem Solving' (DPS) und Multiagentensysteme (MAS). MAS sind durch mehrere Agenten charakterisiert, die lose im selben Zustandsraum mit den selben Zielen interagieren, von denen manche die Fähigkeiten der Einzelagenten übersteigen. Bei verteilten Problemlösern (DPS) organisieren sich mehrere Agenten in eine Sequenz, um gemeinsam eine Aufgabe zu lösen. Bei solchen Problemlösern ist die Koordination integraler Teil des Systems.

Eine klare Trennung zwischen diesen Subgebieten der DAI scheint zweifelhaft, da die Grenzen zwischen Multiagentensystemen und verteilten Problemlösern fliessend sind. Ein Beispiel ist der Fall von Agenten, die Subagenten ausbilden können. Diese Subagenten können mehreren Primäragenten dienen, die ihrerseits die entsprechenden Kompetenzen verlieren. Daher erscheint die Organisation dann sequentiell wie in verteilten Problemlösern, obgleich mehrere Ziele gleichzeitig verfolgt werden. Die Entwicklungen in der DAI in den letzten Jahren verwischten die Grenzen zwischen den Teilgebieten weiter, sodaß das gesamte Gebiet in letzter Zeit als Multiagentensysteme bezeichnet wird.

Die in der DAI-Literatur behandelten MAS-Ansätze beziehen sich zum überwiegenden Teil auf Systeme aus mehreren komplett autonomen Agenten oder aus sequentiell organisierte Experten mit eingeschränkter Expertise, deren Verkettung erst die Gesamtaufgabe löst. Auch in den Multiagentensystemen für Roboter konzentrieren sich die Forschungen auf diese Probleme, nur werden sie hier 'modulare' Robotik für Multirobotersysteme oder 'zellulare' Robotik genannt [Ferber, 2001]. Bei den Systemen der zellularen Robotik werden Experten koordiniert, die Teile eines Gesamtsystems steuern, wie die Gelenke eines Roboterarms, zur Erfüllung einer Gesamtaufgabe, wie einer Greifbewegung, die ausserhalb des Aktionsradius eines einzelnen Gelenks liegt.

Grundelement der MAS in der DAI sind wiederum Agenten, die den in Abschnitt 2.2 gegebenen Definitionen entsprechen, aber in der DAI traditionell eher kognitiver Natur sind, also eher Verkörperungen der symbolischen KI darstellen. Einer der prominentesten Agenten ist 'BDI' (Belief, Desire, Intention)[Rao and Georgeff, 1995]: Hier wird versucht, menschliche Intuition über menschliche Entscheidungsprozesse ('Straßenpsychologie') zu modellieren. Der Agent hat einen 'mentalen Zustand' aus Beliefs (Annahmen, entspricht sensorischer Information), Desires (Wünschen, entspricht Verhaltensalternativen) und In-

tentions (Absichten, entspricht einem Plan). Durch wiederholte Aktualisierung der Beliefs, versucht er festzustellen, welche Desires verfügbar sind, und filtert die Desires, um neue Intentions und entsprechende Aktionen festzulegen.

Kommunikation und Interaktion Im Gegensatz zu reaktiven und verhaltensbasierten Agenten verfügen DAI-Agenten zur abgestimmten Planung nahezu immer über eine explizite Kommunikation untereinander und viel Forschungsarbeit wurde in den Entwurf entsprechender Sprachen und Protokolle gesteckt.

Die einfachste gemeinsame Sprache zwischen Agenten ist dabei die indirekte Kommunikation über eine Veränderung der Umwelt (*Stigmergy*), wie sie auch verhaltensbasierte Systeme benutzen. Die Agenten verändern die Umwelt dabei entweder zu eigenen Zwecken, oder um einen anderen Agenten zu einem bestimmten Verhalten zu bewegen. Die Kommunikation zwischen Agenten kann verschiedene direkte Formen annehmen, die wiederum unterschiedliche Anforderungen stellen [Schumacher, 2001]:

- **individuell:** Einzelne Agenten kommunizieren miteinander. Dazu müssen sie den Kommunikationsakt synchronisieren und sich gegenseitig identifizieren.
- **Gruppe:** Ein Agent kommuniziert mit einer Gruppe. Die Kommunikation muß synchronisiert werden.
- **Broadcast:** Ein Agent kommuniziert mit allen anderen. Die Kommunikation muß synchronisiert werden.
- **generativ:** Ein Agent generiert eine Nachricht auf einer speziellen Struktur, wie z.B. einem Blackboard [Corkill, 1991], die dort autonom existiert und von anderen Agenten wahrgenommen werden kann. Synchronisierung und Identifikation sind nicht notwendig.

Die Kommunikation dient natürlich der Organisation der Interaktion der Agenten untereinander. [Ferber, 2001] gibt eine Übersicht über mögliche Interaktionsarten, die sich nach den folgenden drei Kriterien unterscheiden lassen:

Verträglichkeit der Ziele: Agenten können gegensätzliche oder inkompatible Ziele verfolgen, oder gegenseitig neutrale oder kooperative, die dann als kompatibel bezeichnet werden.

Zugänglichkeit der Ressourcen: Alle Ressourcen sind begrenzt, und greift mehr als ein Agent auf dieselbe Ressource zu, entsteht ein Konflikt, der gelöst werden muß.

Fertigkeit der Agenten: Die einzelnen Agenten können hinsichtlich ihrer Aufgaben mehr oder weniger kompetent sein, also über ausreichende oder unzureichende Fertigkeiten verfügen, allein die ihnen übertragene Aufgabe zu erfüllen.

Die verschiedenen Kombinationen dieser Kriterien ergeben eine Reihe typischer Interaktionskategorien, die in Tabelle 2.2 aufgeführt und darauffolgend erläutert werden.

Die in der Spalte 'Interaktion' von Tabelle 2.2 aufgeführten möglichen Interaktionsarten können wie folgt kurz charakterisiert werden:

Interaktion	Ziel-verträglichkeit	Ressourcen-zugänglichkeit	Fertigkeiten
Unabhängigkeit	kompatibel	ausreichend	ausreichend
einfache Kooperation	kompatibel	ausreichend	unzureichend
Blockade	kompatibel	unzureichend	ausreichend
koordinierte Kooperation	kompatibel	unzureichend	unzureichend
rein individueller Wettbewerb	inkompatibel	ausreichend	ausreichend
rein kollektiver Wettbewerb	inkompatibel	ausreichend	unzureichend
individuelle Ressourcenkonflikte	inkompatibel	unzureichend	ausreichend
kollektive Ressourcenkonflikte	inkompatibel	unzureichend	unzureichend

Tabelle 2.2: ***Interaktionen zwischen Agenten.*** *Einteilung der Interaktionsarten nach* FERBER, *Tabelle in Anlehnung an [Ferber, 2001, Tabelle 2.1, S.90], Erläuterung der Interaktionen siehe Text*

Unabhängigkeit: Problemloses Nebeneinander der Agenten ohne Konflikte und ohne gegenseitige Unterstützung. Einfachste Kooperationsform.

einfache Kooperation: Zusammenarbeit durch Addition der Fertigkeiten ohne Koordination der Agenten. Kooperation steigert die Überlebensfähigkeit, verbessert die Performanz und löst Konflikte [Ferber, 2001].

Blockade: Agenten behindern sich, benötigen sich aber nicht.

koordinierte Kooperation: Synergieeffekte zwischen Agenten können durch Kooperation ausgenutzt werden. Komplexeste Kooperationsform.

rein individueller Wettbewerb: Einfacher Wettkampf, den der beste Agent gewinnt.

rein kollektiver Wettbewerb: Gruppenbildung, um mangelnde individuelle Fertigkeiten auszugleichen, und Wettkampf der Gruppen.

individuelle Ressourcenkonflikte: Der Konflikt resultiert aus beanspruchter Dominanz über eine Ressource

kollektive Ressourcenkonflikte: Verschiedene Gruppen kämpfen um Dominanz über eine Ressource.

Von den in der Tabelle 2.2 erwähnten Interaktionsarten werden einige relevante in Kapitel 5 im Kontext der lernenden MAS aufgegriffen.

Der zentrale Konflikt für diese Arbeit ist der konkurrierende Zugriff mehrer Agenten auf die einmalig vorhandene Ressource der Robotereffektoren. Der Konflikt soll bei Maximierung der Bewertungen für jeden einzelnen Agenten und die Gesamtaufgabe gelöst werden. Das Problem der Koordinierung mehrerer Experten in einem Körper ist also durch Ziele wechselnder Kompatibilität, einen Ressourcenkonflikt und vollkommen ausreichende Fertigkeiten der Agenten gekennzeichnet. Die Kategorisierung dieses Problems wechselt, entsprechend Tabelle 2.2, zwischen 'Blockade' und 'individuelle Ressourcenkonflikte'. Die Voraussetzung, diese selbstinteressierten Experten dazu zu bewegen, eine Koordination

einzugehen, ist eine Art globalen Reward bzw. eine Gesamtperformance, die sie über ihr eigenes Wohlergehen stellen müssen [Mataric, 2001].

2.6 Zusammenfassung

In diesem Kapitel wurden die Begriffe des Verhaltens, des Agenten und des Multiagentensystems eingeführt. Es wurden verschiedene Realisierungsmöglichkeiten und Anforderungen an Agenten ausgeführt. Ausführlich wurde auf die Vorzüge verhaltensbasierter Agenten und die Notwendigkeit eines Lernens beim Verhaltenserwerb eingegangen. Für die Umsetzung eines lernenden Verhaltenserwerbs wurde das Reinforcement Lernen eingeführt.

Die Ausführung einer komplexen Aufgabe durch Koordinierung mehrerer Agenten und die Notwendigkeit einer solchen Koordinierung beim Zugriff mehrerer Agenten auf eine begrenzte Ressource wurde in den Kontext der Multiagentensysteme gestellt und auch hier Eigenschaften und Anforderungen aufgeführt. Gleichzeitig wurde festgestellt, daß die Kombination beider Koordinierungsaufgaben wenig oder gar nicht behandelt wurde, wie auch lernende MAS, und mit *W-Lernen* eine ethologisch motivierte Alternative eingeführt.

Abschliessend wurden Multiagentensysteme selbst und ihre Eigenschaften in das Feld der Verteilten Künstlichen Intelligenz eingeordnet.

Kapitel 3

Szenarien für Verhaltensexperimente

Für Untersuchungen, die bestimmte Eigenschaften von Algorithmen zeigen oder widerlegen sollen, muß eine Aufgabe, oder, im Falle von verhaltenerzeugenden Algorithmen, ein Szenario vorhanden sein, das diese Charakteristiken zeigt.

Dieses Kapitel soll einige Gedanken zu Szenarien vermitteln, die in Untersuchungen benutzt werden, in deren Mittelpunkt die Erzeugung bestimmter Verhalten steht. Dieses Thema wird in der einschlägigen Literatur bislang sehr wenig beleuchtet, obwohl Szenarien unverzichtbar sind. Ihre Eigenschaften bestimmen darüber, ob ein Algorithmus seine Stärken und Vorzüge überhaupt demonstrieren kann.

3.1 Merkmale von Szenarien

Als ein Szenario soll in dieser Arbeit die Umwelt verstanden werden, in der ein Animat oder ein Agent Aufgaben erfüllen soll. Sie umfaßt den gesamten Bereich, in dem das Animat sich bewegt, mit äusseren Grenzen, Objekten und den Gesetzen, nach denen die Aktionen des Animats die Umwelt verändern.

Szenarien können nach verschiedenen Gesichtspunkten charakterisiert werden. Einer davon betrifft ihre Steuerung. Umwelten, insbesondere natürlich Simulationen, können ausschließlich durch die Interaktion mit dem Animat bestimmt sein. Je realer und natürlicher das Szenario ist, umso unabhängiger ist das Szenario vom Animat – die Umwelt verfügt über eine Eigendynamik. Sie verändert sich, ohne vom Animat Aktionen aufgenommen zu haben. Trotzdem muß das Animat unter Umständen auf solche Änderungen reagieren.

Hinsichtlich der Dynamik ihrer Zustandsänderungen sind die Szenarien in deterministische und stochastische zu unterscheiden. Deterministische erzeugen bei vollkommener Abhängigkeit vom Animat in derselben Situation bei Ausführung derselben Aktion immer dieselbe Folgesituation. Stochastische Szenarien dagegen können in derselben Situation auf dieselbe Aktion mit unterschiedlichen Folgesituationen reagieren.

Bleibt die Verteilung der Folgezustände eines stochastischen Szenario über die Zeit immer gleich, spricht man von einem stationären Szenario. Verändert sich die Verteilung dieser Folgezustände jedoch, handelt es sich um ein instationäres Szenario. Animats, soweit

sie in der Lage sind, überhaupt mit instationären stochastischen Szenarien umzugehen, stehen vor dem Problem, Gesetzmäßigkeiten in der Veränderung der Folgezustandsverteilungen zu finden. Diese können komplett zufällig sein oder Gesetzmäßigkeiten folgen, wie sie in Abschnitt 3.2 auch für die einzelnen Objekte, die die Umwelt ausmachen, diskutiert werden.

Darüberhinaus können Szenarien die Ereignisse, die für ein Animat relevant sind, mit bestimmten zeitlichen Verteilungsdichten erzeugen. Eine gefährliche Zone, wie ein Abgrund oder ausbrechender Vulkan, die negative Rewards abgibt, kann immer vorhanden sein oder nur zeitweise. Solche und ähnliche Veränderungen der Umwelt können unabhängig von den Aktionen des Animats geschehen, oder durch das Animat hervorgerufen werden.

Solche Veränderungen können als eine Interaktivität des Szenarios betrachtet werden. Eine Interaktivität stellt höhere kognitive Anforderungen an das Animat als normale, passive Änderungen der Umwelt, wie sie z.B. aus seiner Fortbewegung entstehen.

Sind im Szenario Werkzeuge vorhanden, so kann sich durch deren Benutzung die Effektivität der Aktionen des Animats ändern. Mit Werkzeugen kann sich das Animat leistungsfähiger verhalten, bzw. bestimmte Verhalten überhaupt erst realisieren.

Der Zustand der Umwelt wird durch physikalische Größen beschrieben, aus denen das Animat durch unterschiedliche Perzeptionsfunktionen für sich verwertbare Größen erzeugt. Die Umwelt kann vollständige und unvollständige, lokale oder globale Sensoriken zur Verfügung stellen. Diese Eigenschaft charakterisiert die Zugänglichkeit der Umwelt.

Littman charakterisiert in [Littman, 1994] die verschiedenen Szenarien nach den minimalen Fähigkeiten, die ein Agent haben muß, um optimale Performanz zu erzielen. Dies betrifft einerseits die minimal notwendige Anzahl von Bits, die gespeichert werden müssen, und andererseits die minimale Anzahl der Schritte bis zu einer Erlangung eines Zieles. Aufgaben, die für Algorithmen des Reinforcement Lernens entworfen werden, müssen der Markow-Eigenschaft (siehe Abschnitt 4.2.1) genügen, wenn die Algorithmen optimales Verhalten erzeugen sollen. Dies bedeutet, daß die Umwelten vollständig durch das Animat gesteuert sind, stochastisch sein können, aber stationär sein müssen. Die Verfügbarkeit globaler oder lokaler Sensorik entscheidet darüber, ob ein Algorithmus Ziele verfolgen kann, die außerhalb seiner aktuellen lokalen Sensorik liegen. Ohne globale Sensorik ist das nur möglich, wenn der Algorithmus ein Modell generiert, wie zum Beispiel eine globale Karte, die ihm eine Handlungsplanung ermöglicht.

3.2 Eigenschaften der Objekte

Als Objekte im Szenario sollen alle darin vorkommenden Bestandteile aufgefaßt werden. Ein Szenario besteht entsprechend aus mehreren solcher Objekte, von denen jedes unterschiedliche Eigenschaften aufweisen kann, bis hin zu solchen, die ein Objekt eher als Subjekt erscheinen lassen. Dies ist der Fall, wenn ein Objekt eigene Aktivitäten entwickelt, Farbe, Größe oder Position wechselt. Trotzdem sollen sie im Rahmen dieser Arbeit als Objekte bezeichnet werden, während das Animat als das einzige Subjekt im Szenario angesehen werden soll.

Über die oben für das Szenario als Ganzes eingeführten Merkmale, die auch für die einzelnen Objekte zutreffen, hinaus kann die Dynamik eines Objektes, ob als Eigendynamik oder als Reaktion auf eine Aktion des Animats, mit den Mitteln der Regelungstechnik als sogenanntes Proportional-, Integral-, Differentialglied, Totzeit oder beliebige Kombination dieser Eigenschaften beschrieben werden. Insbesondere können Objekte räumlich stationär oder beweglich sein. Handelt es sich um bewegliche Objekte, so kann ihre Bewegung von der Interaktion mit dem Animat abhängen oder nicht, sie können vom Animat unbeeindruckt ihre Trajektorien verfolgen oder das Animat verfolgen oder vor ihm fliehen.

Die Objekte in Reinforcementaufgaben, wie sie in Abschnitt 4.2 definiert werden, können Belohnungen oder Bestrafungen (im folgenden als Reward bezeichnet) an das Animat abgeben. Sie können sich dem Animat gegenüber neutral, oder direkt bzw. indirekt belohnend oder bestrafend verhalten. Dabei soll unter indirektem Reward die Eigenschaft eines Objektes verstanden werden, Rewardabgabe von anderen Objekten zu ermöglichen, also als Auslöser oder Voraussetzung zu wirken, oder die Effektivität der Handlungen des Animats zu verändern, wie Werkzeuge es tun.

Die Abgabe dieses Rewards weist wieder dieselben Eigenschaften auf wie die Objekte; sie kann zeitlich konstant oder veränderlich sein, hier wird an das Beispiel des Vulkans oben erinnert. Die Veränderung der Abgabe des Rewards kann wiederum interaktiv von den Aktionen des Animats abhängen, wie eine Futterquelle, die nachwachsen kann und deren Wachstumsrate von der Frequentierung durch das Animat abhängt (ähnlich Volterra-Lottka-Funktionen zur Beschreibung der Beziehungen zwischen Wild- und Jägerpopulationen).

3.3 Qualitative Anforderungen an Animats

Die oben beschriebenen Eigenschaften der Objekte und des Szenario als Ganzem stellen unterschiedliche Anforderungen an die Fähigkeiten des Animats. Diese können unterschieden werden in Aufgaben, die von einem Animat allein gelöst werden können, und solche, die nur in Zusammenarbeit mehrerer Animats bewältigt werden können.

Enthält das Szenario zum Beispiel ein Objekt (Wild), das aktiv vor Animats flieht, dann kann ein Animat allein es fangen, wenn das Animat schneller ist als das Wild, oder die kognitive Leistungsfähigkeit ausreicht, seine geringere Geschwindigkeit durch Planung auszugleichen und das Wild in eine Falle zu locken und zu fangen. Dazu jedoch muß das Animat sowohl die Reaktionen des Wildes lernen und vorhersagen können, als auch die geographischen Gegebenheiten der Umwelt global oder lokal so gut kennen, das Wild z.B. in eine Sackgasse zu dirigieren. Das gleiche Problem eines schnelleren Wildes kann aber auch durch gezielte Koordination mehrerer Animats gelöst werden.

Weisen die Objekte oder die Rewardabgabe eine Dynamik auf, so entsteht für das Animat ein Zeitdruck, unter dem es seine Aufgaben zu erfüllen hat. Dieser verschärft sich, sollte die Dynamik der Objekte oder des Rewards interaktiv von den Aktionen des Animats abhängen.

Die Art der dem Animat zugänglichen Information kann die von ihm erreichbare Verhaltenskomplexität einschränken. Es kann zwischen lokalen und globalen Informationen unterschieden werden. Als lokale Informationen werden solche bezeichnet, die aus einer

begrenzten lokalen Umgebung des Animats gewonnen werden können, also ein begrenztes lokales Sichtfeld. Globale Informationen bezeichnen solche, die eine globale Übersicht über das gesamte Szenario voraussetzen, vergleichbar einem Global Positioning System (GPS).

Stellt das Szenario nur lokale Informationen zur Verfügung, so kann das Animat reflexartige und reaktive Aufgaben erfüllen, also Aufgaben, die eine angeborene oder erlernte Reaktion auf eine bestimmte Umweltsituation erfordern, im Sinne eines Stimulus-Response-Reflexes. Kennzeichnen die lokalen Informationen die Situationen eineindeutig, sodaß ein bestimmtes sensorisches Muster nur genau einen Zustand kennzeichnet, so kann das Animat mit einer impliziten Planung durch eine Folge solcher Reflexe auch Aufgaben lösen, deren sensorische Voraussetzungen zunächst nicht erfüllt sind, wie eine Navigation durch ein Labyrinth. Eine solche eineindeutige Kennzeichnung erfüllt die schon oben erwähnte Markow-Eigenschaft. Mit dieser Eigenschaft erfüllt eine lokal gewonnene Information die Funktion einer globalen Information.

Zur Erfüllung von Aufgaben, die eine Planung im Sinne einer aktiven Abschätzung der Konsequenzen seiner Aktionen erfordern, braucht das Animat globale sensorische Informationen. Alternativ kann es sich solche Informationen aus lokalen Informationen und einer Karte bzw. einem ähnlich globalen Modell der Umwelt erzeugen. Wird das Animat zum Beispiel von einem beweglichen Objekt verfolgt, so sollte es Fallen wie Sackgassen vermeiden. Je nach kognitiven Fähigkeiten des Jägers und der Differenz zwischen den Geschwindigkeiten sollte es teilweise unvorhersagbares Verhalten erzeugen, um zu verhindern, daß es selber in eine Falle dirigiert wird. Beide Verhalten verlangen über reaktives Verhalten hinaus eine Planung.

Weitere mögliche Aufgaben für ein Animat schließen eine Identifikation bzw. Diskriminierung von Objekten ein, oder eine Generalisierung im Sinne einer Anwendung bekannter Verhalten auf neue Ziele sowie eine Dekomposition oder Subzielgenerierung.

3.4 Auswahl möglicher Aufgaben

Allgemein können Aufgaben in Vermeidungs-, Erreichungs- und Erhaltungsaufgaben unterteilt werden. Bei einer Vermeidungsaufgabe wird das Animat für bestimmte Aktionen im Szenario bestraft, wie zum Beispiel das Betreten gefährlicher Gebiete oder die Annäherung an einen Feind. Dasselbe Verhalten wird auch durch Belohnung aller anderen Aktionen erreicht. Bei einer Erreichungsaufgabe wird das Animat für bestimmte Aktionen im Szenario belohnt, wie das Erreichen einer Futterstelle. Während eine Vermeidungsaufgabe meist keinen Endzustand hat, sondern zeitlich fortbesteht und daher die Einhaltung eines dynamischen Gleichgewichts erfordert, sind Erreichungsaufgaben meist mit einem Endzustand verbunden. Bei einer Erhaltungsaufgabe erhält das Animat für jede Aktion einen Reward, der die Kosten von Aktionen oder Präferenzen für bestimmte Aktionen beinhalten kann. Dadurch sollen innere Zustände des Animats oder Verhalten erhalten werden.

Die folgende Liste enthält eine Reihe häufiger Navigationsaufgaben für Roboter.

Exploration: Grundlegend notwendige Erkundung der Umwelt.

Aktionsbestrafung: Jede Aktion verursacht Kosten, die dem Animat zum Beispiel als

Energie entzogen werden. Dies stellt eine Erhaltungsaufgabe dar, die das Animat zu ökonomischen Aktionen bewegen soll.

Aktionsbelohnung: Das Animat erhält unterschiedliche Belohnungen für jede Aktion, die zum Beispiel eine schnelle, gerade Fahrweise erzeugen sollen.

Hindernisvermeidung: Eine Kollision mit Hindernissen soll vermieden werden. Eine Vermeidungsaufgabe mit statischen Objekten, die mit lokalen oder globalen sensorischen Informationen und angemessenen Aktionen reaktiv lösbar sein sollte.

Wandverfolgung: Es soll einer Wand gefolgt werden, ohne mit dieser zu kollidieren, woraus eine Kombination einer Vermeidungs- mit einer Erreichungsaufgabe entsteht.

Flucht: Eine Kollision mit dynamischen Objekten, die das Animat verfolgen können, soll verhindert werden. Flucht ist eine Vermeidungsaufgabe, die unter Umständen eine Planung erfordert.

Zielanfahrt: Es soll eine bestimmte Stelle im Szenario erreicht werden. Unter Zielanfahrt soll eine Erreichungsaufgabe mit statischen Objekten verstanden werden.

Verfolgung: Es soll ein bewegliches Objekt im Szenario erreicht werden, daß sich unter Umständen aktiv vom Animat wegbewegt. Dies stellt eine Erreichungsaufgabe mit dynamischen Objekten dar.

Objektmanipulation: Das Animat muß Objekte in der Umwelt identifizieren und bewegen. Dies ist eine Erreichungsaufgabe.

Aus diesen Aufgaben lassen sich komplexere Aufgaben generieren, wie anwachsender Energiebedarf in einem Labyrinth, bei dem das Animat sein Energiebedürfnis stillen will und dazu ein Ziel (Futterstelle) anfahren muß. Dazu muß das Animat entweder die Futterstelle sehen, um sie anzufahren und gegebenenfalls Hindernissen auszuweichen. Alternativ muß es wissen, wo es sich befindet, wo die Futterstelle zu finden ist und wie es die Distanz dazwischen im gegebenen zeitlichen Rahmen überwindet.

3.5 Benchmarks und Performanzmaße

Benchmarks wurden für einen Vergleich verschiedener Algorithmen eingeführt, die zur Lösung gleicher Probleme geschaffen wurden. Um einen exakten, numerischen Vergleich der Leistungsfähigkeit der Algorithmen zu ermöglichen, wurden die Aufgaben, die Randbedingungen des Trainings und der Funktion der Algorithmen soweit wie möglich vorgeschrieben.

Einige häufig verwendete Benchmarks sollen im folgenden aufgeführt werden. Zunächst wird eine Liste mit Benchmarks für Einzelagenten angegeben, gefolgt von einer Liste von Aufgaben für mehrere Agenten. Dabei werden die Szenarien selber und die darin verfügbare Sensorik nur kurz vorgestellt, jedoch für eine Erläuterung der kognitiven Herausforderungen auf Literaturstellen verwiesen, die jeweils eines von vielen Beispielen der Umsetzung der jeweiligen Aufgabe sind:

Rasterwelt [Todd et al., 1994, Wiering and Schmidhuber, 1996]: In einer meist zweidimensionalen Umwelt, die in Zellen eingeteilt ist, sollen Hindernisse vermieden und/oder Ziele erreicht werden.

Stabbalance [Der and Herrmann, 1995]: Ein meist in einer Dimension beweglicher Wagen kann durch Kräfte in dieser Dimension ausgelenkt werden, wobei ein drehbar auf dem Wagen angebrachter Stab in der Senkrechten balanciert werden soll.

Bergwagen [Singh and Sutton, 1996]: Ein Fahrzeug soll aus einem Tal herausgebracht werden, hat aber nicht genügend Motorkraft, direkt herauszufahren. Der Agent muß genügend Impuls aufbauen, um das Fahrzeug herauszuführen.

Lorenzattraktor [Der and Herrmann, 1994]: Auf das System von Differentialgleichungen, das den Lorenzattraktor zur Wetterbeschreibung erzeugt, soll derart eingewirkt werden, daß das System eine bestimmte Trajektorie nicht verläßt.

Die folgende Liste enthält hingegen Aufgaben, die von mehreren Agenten zu lösen sind:

Savanne [Tyrrell, 1993]: In einer Simulation einer afrikanischen Ebene existieren sauberes und giftiges Wasser, Schattenstellen, Verstecke, Gefahrenzonen, unterschiedliche Arten von Futter, Gräben, neutrale Tiere, Jäger, Wild und Partner. Die Aufgabe ist zu überleben und sich fortzupflanzen.

Hausroboter [Humphrys, 1997]: In einem Haus sollen dessen Wände vermieden und verfolgt werden, Fremde angesehen und der Familie ausgewichen werden, sowie gesäubert und mögliche Feuer gelöscht werden.

Predator/Prey (Pursuit) [Ono and Ikeda, 1996]: Mehrere Agenten (meist vier) sollen in einer Gridwelt ein Wild fangen, indem sie es von allen Seiten blockieren.

DVMT [Ferber, 2001]: (Distributed Vehicle Monitoring Testbed) Mehrere geographisch verteilte Agenten beobachten Fahrzeugbewegungen auf ihrer lokalen Straße, um Fahrzeuge zu identifizieren, zu verfolgen und eine Karte von Fahrzeugbewegungen zu erstellen. Sie kommunizieren dazu mit den anderen Agenten, um höhere Sicherheit für die eigene Karte zu erlangen und Beobachtungsredundanzen zu vermeiden. Aus den lokalen Informationen soll eine global konsistente Strassenverkehrssituation erstellt werden.

Die kognitive Komplexität dieser Szenarien wird natürlich wesentlich durch die den Agenten zur Verfügung gestellten sensorischen Informationen mitbestimmt. Die Liste soll jedoch nur eine Auswahl oft eingesetzter Problemstellungen bieten, die in vielen Variationen wiederkehren.

Die Rangfolge der Algorithmen für ein Benchmarkproblem wird anhand der Leistung nach ebenfalls festgelegten Performanzmaßen ermittelt. Bei Szenarien für Algorithmen des Reinforcement Lernens soll das Performanzmaß über die Adäquatheit oder Optimalität des Verhaltens Auskunft geben. Entsprechend basieren die verwendeten Maße oft auf dem erzielten Reward oder auf einem Vergleich mit einem bekannten, optimalen Verhalten. Neben den Anforderungen an Steuerungsmechanismen von Animats (siehe Abschnitt 2.4) wurden in der Literatur dazu verschiedene Performanzmaße verwendet:

- Summe der einzelnen Rewards (über bestimmten Zeitraum)
- Zeit des erfolgreichen Überlebens
- Anzahl der korrekten Aktionen / Anzahl aller Aktionen
- Konvergenzgeschwindigkeit zum Endverhalten
- Geschwindigkeit der Annäherung an Optimalverhalten
- Erfüllung der optimalen Performanz der Einzelaufgaben.

Unter diesen Maßen ist kein allgemein anerkanntes oder universell verwendbares vertreten. Für die Berechnung einer prozentualen Erfüllung der Gesamtaufgabe oder der Teilaufgaben, wie auch für die Berechnung der Annäherungsgeschwindigkeit an die Optimalstrategie, muß es entsprechende Maßstäbe geben. Dies ist jedoch gerade im Falle der komplexen Aufgaben, die meist durch Animats mit mehreren Experten zu lösen sind, nicht der Fall. Daher wird als Bewertungskriterium die Summe der Rewards eingesetzt.

3.6 Zusammenfassung

Dieses Kapitel sollte vor allem der Verdeutlichung verschiedener Möglichkeiten bei der Gestaltung von Szenarien dienen und zur Betrachtung der daraus folgenden Anforderungen an das Animat anregen. Die Auswahl der vorgestellten Beispiele für Szenarien war deutlich geprägt durch solche Umwelten, wie sie zur Untersuchung von Algorithmen des Reinforcement Lernen häufig eingesetzt werden, das im folgenden Kapitel detailliert erläutert wird.

Kapitel 4

Verhaltenserwerb mit Reinforcement Lernen

Dieses Kapitel ist der Vorstellung von Reinforcement Lernen als der zentralen Methode zum Erwerb von Verhalten gewidmet. Die Vorteile bzw. die Notwendigkeit eines lernenden Erwerbs solchen Wissens wurde vor allem in Kapitel 2 motiviert. Aus der Unvorhersehbarkeit aller durch das Animat anzutreffenden Situationen ergibt sich zum Teil auch die Einschränkung auf Reinforcement Lernen. Eine verbreitete alternative Möglichkeit des Wissenerwerbs, wäre überwachtes Lernen, zum Beispiel mit Backpropagation oder rekurrenten neuronalen Netzen. Einerseits jedoch widerspricht die Annahme notwendiger, bekannter, optimaler Verhalten der vorne vorausgesetzten Unbekanntheit der Umwelt, andererseits kann solches eventuell bestehendes Wissen auch leicht mit den Methoden des Reinforcement Lernens vermittelt werden.

Wegen seiner Leistungsfähigkeit zur Verhaltensgenerierung widmet sich dieses Kapitel einer ausführlichen Darstellung des Reinforcement Lernens und seiner Umsetzung mit neuroinformatischen Mitteln. Die Darstellung hat einen deutlichen Übersichtscharakter, sodaß bei weitem nicht alle vorgestellten Algorithmen in den in Kapitel 8 vorgestellten Untersuchungen auch betrachtet oder gar herangezogen werden. Um eine kompakte Darstellung ihrer Prinzipien zu erleichtern, nimmt erst Abschnitt 4.10 eine kritische Wertung hinsichtlich der Untersuchungen dieser Arbeit vor.

Andere mögliche Ansätze zum lernenden Erwerb solchen Wissens schliessen vor allem genetische (siehe z.B. die Arbeiten von Nolfi, Floreano) oder evolutionäre Algorithmen (siehe Rechenberg, Schwefel), der adaptiven Regelung [Irwin et al., 1995] und dynamische Systeme ein, die teilweise biologisch motiviert sind (siehe Grossberg, Schmajuk, Balkenius, Steinhage). Sie werden in Kapitel 9 aufgegriffen. Darüber hinaus finden verstärkt Algorithmen, die auf Weiterentwicklungen der *Classifier Systems* zu Learning bzw. Adaptive Classifier Systems [Stolzmann, 2000] beruhen, Verwendung, wie auch solche aus der Fuzzy Logik. Ausführliche Darstellungen zu diesen Ansätzen sind in den genannten Literaturquellen bzw. bei den genannten Autoren zu finden, wenn deren Kreativität so viele Algorithmen hervorbrachte, daß ein einzelner Literaturverweis ihr nicht gerecht würde.

4.1 Einordnung und Übersicht

Das Reinforcement Lernen (RL) oder Verstärkungslernen hat seine Ursprünge in der Modellierung klassischer und instrumentaler Konditionierung [Klein, 1991, Barto, 1992], die die Grundlagen für das Lernen von Verhalten bei Tieren darstellen. Diese Algorithmen modellieren, wie ein Tier lernt, Zusammenhänge zwischen positiven und negativen Rückwirkungen aus der Umwelt und der zielgerichteten Befriedigung seiner eigenen Interessen oder Bedürfnisse herzustellen. Diese Rückwirkungen funktionieren als Bewertung und verstärkende Faktoren für die Aktionen, die zu eben diesen Rückwirkungen geführt haben, was den Namen Verstärkungs- oder Reinforcement Lernen erklärt.

Reinforcement Lernen kann aus konnektionistischer Sicht als eine Lernmethode bezeichnet werden, die zwischen überwachten (supervised) und unüberwachten (unsupervised, datengetriebenen) Algorithmen anzusiedeln ist. Da eine erzeugte Aktion nicht wie bei überwachten Lernverfahren mit einem vorgegebenen Verhalten verglichen, sondern durch das meist skalare Bewertungssignal bewertet wird, erhält der Agent anstelle eines Gradienten (Betrag und Richtung des Fehlers für jede Outputkomponente bezüglich des bestmöglichen Output) nur eine skalare Information.

Aufgrund dieser wesentlich unspezifischeren Trainingsinformationen konvergieren RL-Verfahren typischerweise langsamer als überwachte Lernverfahren (wie z.B. Error-Backpropagation), sind jedoch aufgrund des nicht benötigten Teachers universeller einsetzbar und aus methodischer Sicht durch die notwendige Interaktion der Systeme mit ihrer Umwelt und der angestrebten Verhaltensorganisation im Kontext realer Anwendungen wesentlich plausibler und leistungsfähiger als überwachte Lernverfahren (neural adaptive control [Zbikowski et al., 1994]). Ein solcher Vergleich ist zulässig, da RL-Verfahren durchaus zur Lösung von überwachten Problemen [Sutton, 1988] herangezogen werden können, indem der Reward aus der Abweichung zwischen bestmöglichem und tatsächlichem Output ermittelt wird.

Die in Abb. 4.1 skizzierte Taxonomie enthält eine Übersicht über die im folgenden erläuterten Zusammenhänge und relevante sowie eingeführte Algorithmen des RL.

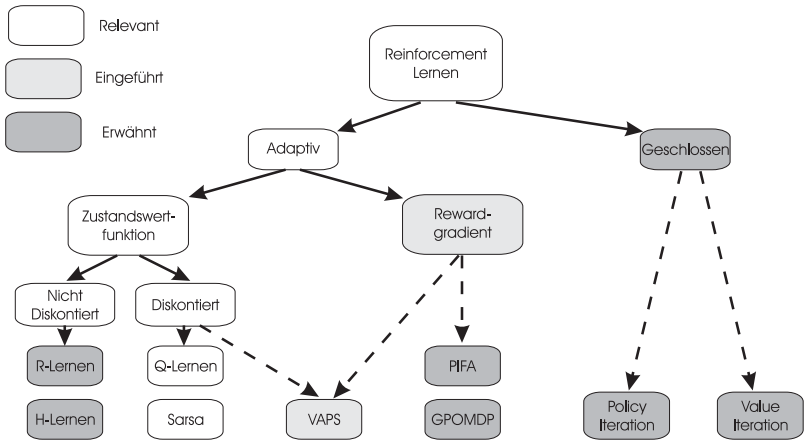

Abbildung 4.1: *Taxonomie über einige Verfahren des Reinforcement Lernens. weiss hinterlegt = Algorithmus ist für Arbeit relevant und wird diskutiert, hellgrau = Algorithmus wird erläutert, dunkelgrau = Algorithmus wird mit Literaturverweis erwähnt.*

4.2 Grundlagen

Im Reinforcement Lernen [Sutton and Barto, 1998] wird ein autonomer Agent in eine anfänglich unbekannte, beliebige dynamische Umgebung (*Environment*) eingesetzt, deren Zustand (*state* $s(t)$) er durch eine Beobachtung (*observation* $o(s,t)$) wahrnimmt. Er lernt durch umfassendes Austesten (Versuch-und-Irrtum, *Trial-and-Error*), genau solche Aktionen (*Action* $a(t)$) auszuführen, die über einen längeren Zeitraum eine akkumulierte Bewertung (*Return* R) maximieren. Dies beschreibt die sogenannte RL-Aufgabe, bei der eine Folge von Aktionen und Zustandsübergängen über eine bestimmte Anzahl von Schritten durch den dabei akkumulierten Return bewertet wird.

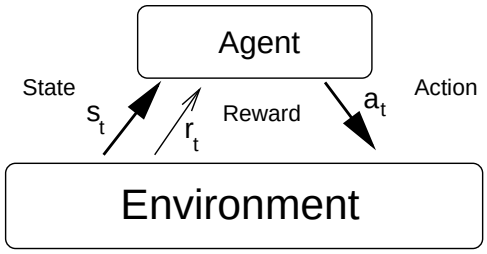

Abbildung 4.2: ***Beziehungen zwischen Umgebung und Agent***. *In einem Ausgangszustand der Umwelt s_t führt der Agent eine Aktion a_t aus, die die Umwelt in den Folgezustand $s_{t+1} = s'$ versetzt. Der Agent erhält dafür die Bewertung r_{t+1}*

Die Grundlage des Returns ist eine einzelne Bewertung (*Reward-Signal* $r(t+1)$), die der Agent nach der Aktionsausführung aus der Umwelt (siehe Abbildung 4.2) bzw. von einem Designer erhält oder aber selber aus einer entsprechend designten, 'inneren' Evaluierung ermittelt. Der Return ist das Ergebnis der Akkumulation des Rewards über eine bestimmte Anzahl von Schritten.
Charakteristisch für das RL ist, daß die Bewertung einer Aktion nicht unmittelbar erfolgen muß, sondern verzögert sein kann ('delayed reward'). Die Bewertung ist ein Maß der Verbesserung/Verschlechterung des Systemzustandes bezüglich der ausgeführten Aktion und kann sowohl eine Bewertung der Aktion selber, des resultierenden Umweltzustandes oder des Zustandes des Agenten, aber auch entsprechender Zustandsänderungen enthalten. Dies macht die Bewertung zu dem Instrument, mit dem der Designer dem Agenten mitteilt, **was** gelöst werden soll, jedoch nicht **wie** (anders als z.B. im überwachten Lernen). Dies muß der Agent selber finden und kann dabei durchaus zu besseren Lösungen als im überwachten Fall gelangen. Er muß durch *Exploration* (siehe Abschnitt 4.5) eine optimale Strategie (*Policy* $\pi(o,a)$) entwickeln. Die Policy ist eine stochastische Regel, mit der der Agent abhängig vom aktuellen Zustand eine Aktion auswählt. Optimal ist diejenige Policy, die jedem Zustand eine geeignete Aktion derart zuordnet, daß bei ihrer Ausführung der Return über einen bestimmten Zeitraum maximiert wird.

4.2.1 Markow-Eigenschaft

Der Ausgangspunkt der Entscheidung über die als nächstes auszuführende Aktion ist der gegenwärtig wahrgenommene Zustand der Umwelt. Dabei setzen RL-Algorithmen voraus, daß der aktuelle sensorische Eindruck des Agenten insofern vollständig ist, als das alle zur

Entscheidungsfindung und Vorhersage des Rewards notwendigen Informationen enthalten sind. Diese Eigenschaft wird als **Markow**-Eigenschaft bezeichnet. Sie besagt, daß der Zustand der Umwelt s' zum Zeitpunkt $t+1$ ausschließlich vom Zustand der Umwelt s zum Zeitpunkt t und der zum Zeitpunkt t ausgeführten Aktion abhängt, nicht jedoch von vorhergehenden Zuständen und Aktionen. Dies wird durch die *1-Schritt-Dynamik* des Systems beschrieben (Gl. 4.1): Die Wahrscheinlichkeit, daß die Ausführung der Aktion a im Zustand s zum Folgezustand s' ($P^a_{ss'}$), hängt allein vom gegenwärtigen Zustand s_t und der darin ausgeführten Aktion a_t ab. Der Erwartungswert des dabei erfahrenen Rewards $r^a_{ss'}$ ist vollständig durch diesen Zustandsübergang bestimmt.

$$\begin{aligned} P^a_{ss'} &= Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \\ r^a_{ss'} &= E\{r_{t+1} | s_{t+1} = s', s_t = s, a_t = a\} \end{aligned} \tag{4.1}$$

Implizit setzt diese Betrachtungsweise neben einer Diskretisierung der Zustands- und Aktionsräume auch eine Diskretisierung der kontinuierlichen Dynamik in einzelne Zeitpunkte voraus. Eine RL-Aufgabe, die die MARKOW-Eigenschaft erfüllt, wird als **Markov Decision Process** (MDP) bezeichnet.

Kann die Ausführung der Aktion a im Zustand s die Umwelt nur in genau einen Folgezustand s' versetzen, wird der dazugehörige MDP als *deterministisch* bezeichnet. Dies ist jedoch keine notwendige Voraussetzung für RL; es kann auch *stochastische* MDP bearbeiten, bei denen die Ausführung der Aktion a im Zustand s die Umwelt in unterschiedliche Folgezustände $s'_{1...n}$ versetzen kann.

4.2.2 Wertfunktion

Die Entscheidung über die in einem Zustand auszuführende Aktion wird durch die Policy $\pi(s,a)$ getroffen. Diese Policy steht im Wechselspiel mit der Zustandswertfunktion $V^\pi(s) \ : \ S \rightarrow R$. V liefert ein Gütemaß dafür, wie vorteilhaft es für den Agenten ist, sich in einem bestimmten Zustand aufzuhalten. Der Wert (*Value*, $V^\pi(s)$) eines Zustandes ergibt sich als Erwartungswert des Returns R_t (Gl. 4.2) und gibt daher Auskunft über die zu erwartenden akkumulierten Rewards. Der Erwartungswert hängt von den nachfolgend ausgeführten Aktionen und somit von der Policy π ab. Die Values der aus einem aktuellen Zustand erreichbaren Folgezustände werden ihrerseits aber zur Ermittlung der, entsprechend der Policy nächsten auszuführenden, Aktion herangezogen, woraus die Abhängigkeit der Policy von der Value-Funktion ersichtlich wird.
Alternativ kann eine Zustandsaktionswertfunktion $Q^\pi(s,a) \ : \ (S,A) \rightarrow R$ betrachtet werden, die prinzipiell die gleichen Eigenschaften hat und die gleiche Funktion erfüllt, aber nicht dem gesamten Zustand einen Wert hinsichtlich zukünftiger Bewertungen zuordnet, sondern jeder in diesem Zustand ausführbaren Aktion eine 'Nützlichkeit'. Es können mehrere Definitionen für den Wert bzw. die Nützlichkeit betrachtet werden:

$$V^\pi(s) = E^\pi\{R_t|s_t = s\} \quad \text{bzw.} Q^\pi(s,a) = E^\pi\{R_t|s_t = s, a_t = a\} \tag{4.2}$$

$$= E^\pi\left\{\frac{1}{h}\sum_{k=0}^{h-1} r_{t+k+1}\right\} \quad \text{bzw.} \quad = E^\pi\left\{\frac{1}{h}\sum_{k=0}^{h-1} r_{t+k+1}\right\} \tag{4.3}$$

$$= E^\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \quad \text{bzw.} \quad = E^\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \tag{4.4}$$

$$0 \leq \gamma \leq 1$$

oder

$$= E^\pi\left\{\lim_{k\to\infty}\frac{1}{k}\sum_{i=0}^{k} r_{t+i+1}\right\} \text{bzw.} \quad = E^\pi\left\{\lim_{k\to\infty}\frac{1}{k}\sum_{i=0}^{k} r_{t+i+1}\right\} \tag{4.5}$$

Gl. 4.3 beschreibt das einfachste Modell optimalen Verhaltens in Abhängigkeit von der Definition des Returns, in dem über eine feste Anzahl von Zeitschritten h der erwartete Return maximiert werden soll.

Gleichung 4.4 bewertet eine Policy mit *unbegrenztem Zeithorizont* und beinhaltet durch den *Diskontierungsfaktor* γ eine höhere Gewichtung kürzlich erfahrener Bewertungen gegenüber länger zurückliegenden. Dieser Faktor kann als Zinsrate oder aber auch als Wahrscheinlichkeit interpretiert werden, daß der Agent einen weiteren Schritt erlebt. Einer etwas anderen Interpretation folgend, kann γ auch als Kosten verstanden werden, die durch die Ausführung eines weiteren Schrittes entstehen und daher den zu erwartenden Reward verringern. Ein festes γ würde demzufolge feste Kosten pro Schritt modellieren. Es wäre auch denkbar, stattdessen die pro Schritt tatsächlich aufgewendeten Kosten zu verwenden. Diese Möglichkeit findet jedoch kaum Einsatz.
Mathematisch begrenzt der Faktor die Summe und ist daher für die Konvergenz notwendig. Dies ist das in den meisten RL-Algorithmen verwendete und daher am besten untersuchte Modell. Entsprechende Algorithmen werden in Abschnitt 4.3.1 vorgestellt. Eine Betrachtung dieses Modells für $\lim_{\gamma\to 1}$ ergibt andererseits eine Policy, wie sie mit Gl. 4.5[1] erzielt und mit den Algorithmen in Abschnitt 4.3.2 umgesetzt würde.

Optimalitätsmetriken Zum Vergleich der Leistungsfähigkeit der Policies wurden verschieden mächtige Optimalitätsmetriken eingeführt [Mahadevan, 1996b], die mit den oben genannten Wertfunktionen korrespondieren. In Abbildung 4.3 sind die Beziehungen der Metriken dargestellt, die im folgenden kurz umrissen werden.

- **Total reward Optimality**, bei dem die Rewards aufsummiert werden und entsprechend schnell alle Grenzen sprengen kann,
- **Discounted Optimality**, bei der zukünftiger Reward durch einem Discountfaktor $\gamma < 1$ abgewertet wird. Hier entstehen Konvergenzprobleme, wenn γ nahe 1 gewählt wird. Weiterhin ist zu beobachten, daß die optimale Policy sich mit dem Variieren

[1] Die Zeitabhängigkeit von $x = x(t)$ wird im folgenden aus Gründen der Übersichtlichkeit fallengelassen.

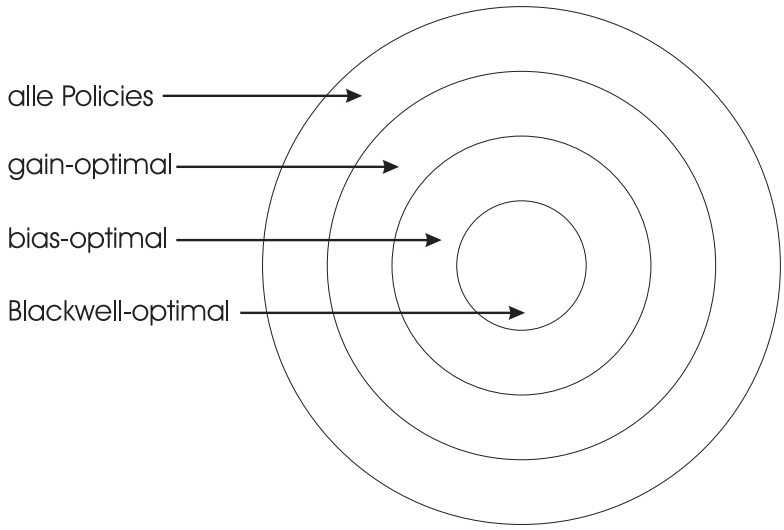

Abbildung 4.3: *Übersicht der Optimalitätsmetriken*

des Discountfaktors ändert. Die in Abschnitt 4.3.1 eingeführten Algorithmen ermitteln die zugehörigen Policies.

- **Gain Optimality**, bei der der erwartete Reward *pro Schritt, average reward (durchschnittlicher Reward)* ρ maximiert wird. Dieses Kriterium kann nicht zwischen Policies unterscheiden, die denselben average reward besitzen, aber sehr unterschiedliche 'Startkosten' bei den initialen transienten Zuständen aufweisen. Da hierbei über möglicherweise unendlich viele Zeitschritte gemittelt wird, gehen Unterschiede der Policies bei individuellen Zustandsübergängen verloren. Betrachtet man z.B. einen MDP, der Perioden enthält, also rekurrente Zustände, in die das System nach einer bestimmten Anzahl (der Periode) von Zustandsübergängen wieder zurückkehrt, so verringert sich mit fortschreitender Zeit der Einfluß von (nichtrekurrenten) Zustandsübergängen außerhalb der Periode, und seien sie mit noch so extremen Rewards verbunden. Diese Metrik ignoriert also, zu welchem Zeitpunkt der Aktionsfolge besonders große Rewards erzielt wurden, aber auch, ob die Aktionsfolge einen Attraktor oder *terminalen* bzw. absorbierenden Zustand, z.B. einen definierten Zielzustand, erreicht hat. Die in Abschnitt 4.3.2 vorgestellten Algorithmen ermitteln entsprechende Policies.

- **Bias Optimality**, bei dem die *durchschnittlich angepaßte (average adjusted)* Summe der Rewards durch Verwendung eines Bias betrachtet wird. Sie unterscheidet Policies mit extremen, aber vereinzelt auftretenden Rewards von Policies mit gleichen durchschnittlichen Rewards.

Darüberhinaus existieren weitere Metriken, die es z.B. erlauben, Policies mit gleichen durchschnittlichen und gleichen vereinzelten Rewards anhand der benötigten Anzahl von Aktionen zu unterscheiden [Mahadevan, 1996a] (*n-discount optimal, Blackwell optimal, T-optimal*). Da dem Autor hierzu keine Algorithmen bekannt sind und das Lernziel der Mehrzahl der RL-Algorithmen eine Policy entsprechend Gl. 4.4 ist, wird auf eine Darstellung des komplizierten mathematischen Apparates verzichtet.

Policies, die aus Gl 4.5 enstehen, sind gain optimal, während bei solchen aus Gl. 4.4 der Faktor γ entscheidet, ob sie bias optimal sind. An dieser Stelle eröffnet sich ein Widerspruch, der nur anhand der konkreten Aufgabenstellung gelöst werden kann: kleine γ berücksichtigen wegen des kleinen Planungshorizontes nur die unmittelbar nächsten Rewards. Die Policy würde also einen kurzen Weg mit mittlerem Return einem längeren Weg mit hohem Return vorziehen. Mit steigendem γ würde jedoch diese Policy umschlagen, könnte dafür jedoch wesentlich höhere Trainingszeiten benötigen, als eine Policy entsprechend Gl. 4.5 ([Mahadevan, 1996a]).

Optimale Wertfunktionen Die Ermittlung einer optimalen Policy steht in engem Zusammenhang mit der Ermittlung einer optimalen Zustandswertfunktion bzw. Aktionswertfunktion. Es kann nur genau eine optimale Zustandswertfunktion $V^*(s)$ geben, für die $V^*(s) \geq V^\pi(s)$ gilt. Sie wird möglicherweise durch mehrere optimale Policies erreicht. Bei Bewertung eines MDP für einen Zeithorizont $h > 1$ müssen die über den nächsten Zeitschritt hinausgehenden Konsequenzen einer Aktion berücksichtigt werden. Es müssen also für jeden Startzustand $|A|^h$ Policies geprüft werden: alle ausführbaren Aktionen in jedem der Schritte im Zeithorizont. Da infinite, diskontierte MDP stationäre Policies[2] erzeugen, kann die Suche auf alle möglichen Zustände und somit $|A|^{|S|}$ Policies[3] beschränkt werden. Der optimale Wert eines Zustandes $V^*(s)$ unter Gl 4.4 ergibt sich (Gl. 4.6) als Summe aus dem erwarteten unmittelbaren Reward $r^a_{ss'}$ und dem γ-diskontierten erwarteten Wert des Folgezustandes $V^*(s')$ nach Ausführung der besten verfügbaren Aktion $\max_{a \in A}$. Dabei wird der Erwartungswert über alle Folgezustände s' betrachtet, die mit der Wahrscheinlichkeit des Zustandsübergangs $P^a_{ss'}$ gewichtet werden.

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P^a_{ss'} \cdot [r^a_{ss'} + \gamma V^*(s')] \tag{4.6}$$

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P^a_{ss'} \cdot [r^a_{ss'} + \gamma V^*(s')] \tag{4.7}$$

Entsprechend kann mit Gl. 4.7 eine optimale Policy π^* bestimmt werden: jede Policy, die auf der Grundlage der optimalen Wertfunktion V^* diejenige Aktion $\arg\max_{a \in A}$ auswählt, die die Summe aus Reward und Wert des Folgezustandes (Return $R = r^a_{ss'} + \gamma V^*(s')$) maximiert, ist eine optimale Policy. Die Policy der Wahl der Aktion, die maximale Werte ergibt, wird als 'greedy' (gierige) Policy bezeichnet.
Es können mehrere π^* existieren, da es in einem Zustand mehrere Aktionen mit gleichem Reward und Folgezustandswert geben kann. Gl. 4.6 und Gl. 4.7 werden als Bellman Optimalitätsgleichungen [Bellman, 1957] bezeichnet, da sie das Bellman'sche Optimalitätsprinzip umsetzen. Nach diesem Prinzip ist diejenige Policy optimal, die unabhängig vom initialen Zustand und der initialen Aktion die danach folgenden Aktionen einer optimalen Policy realisieren.

[2]Für MDP mit finitem Zeithorizont kann die Policy vom Startpunkt abhängen, bei nichtdiskontierten MDP kann keine Konvergenz auf eine Policy bewiesen werden. Es kann also jeweils nicht von einer stationären Policy ausgegangen werden.

[3]$|A|$ Anzahl der Aktionen, $|S|$ Anzahl der Zustände

4.2.3 Dynamische Programmierung

Eine optimale Zustandswertfunktion bzw. Aktionswertfunktion kann mit Methoden der *Dynamischen Programmierung* (DP) gefunden werden. Es handelt sich nicht um Lernmethoden, sondern Suchmethoden, die anstelle ganzer Aktionssequenzen nur einzelne Aktionen (was die Suchkomplexität begrenzt) betrachten. Diese Methoden finden kaum Anwendung, da sie ein perfektes Modell des Prozesses voraussetzen und sehr berechnungsintensiv sind. Sie haben aber große Bedeutung für das Verständnis der RL-Algorithmen, da diese die Methoden der DP inkrementell und ohne perfektes Modell approximieren.
Eine Policy wird nach der von ihr erreichten Wertfunktion beurteilt, die nach Gl. 4.8 (*policy evaluation*) berechnet wird. Der neue Zustandswert $V^{\pi}_{k+1}(s)$ ergibt sich aus $\pi(s,a)$ (Wahrscheinlichkeit von Aktion a in Zustand s), dem Reward und dem alten Folgezustandswert $V^{\pi}_{k}(s')$.

$$V^{\pi}_{k+1}(s) = \sum_{a \in A} \pi(s,a) \cdot \sum_{s' \in S} P^{a}_{ss'} \cdot [r^{a}_{ss'} + \gamma V^{\pi}_{k}(s')] \tag{4.8}$$

$$\pi_{k+1}(s) = \arg\max_{a \in A} \sum_{s' \in S} P^{a}_{ss'} \cdot [r^{a}_{ss'} + \gamma V^{\pi_k}_{k}(s')] \tag{4.9}$$

Ist ein perfektes Modell der Umwelt verfügbar, läuft diese Berechnung auf die Lösung eines Gleichungssystems mit $|S|$ Gleichungen hinaus. Da dies sehr viele sein können, wird die Lösung zumeist iterativ berechnet, indem zunächst eine beliebige Anfangswertfunktion angenommen wird, und mit Gl. 4.8 solange ein neuer Wert (und dann eine neue Wertfunktion) für alle Zustände berechnet wird, bis sich die Werte nicht mehr ändern.

Policy Improvement (Gl. 4.9) besteht darin, auf eine bestehende Wertfunktion eine 'greedy' Policy (Wahl der besten Aktion) anzuwenden: Gl. 4.7 wird als Änderungsvorschrift für die vorhandene Wertfunktion umgesetzt. Dieses Vorgehen garantiert mindestens eine gleichwertige oder sogar verbesserte Policy. Wird die Policy π_{k+1} verwendet, um eine neue Wertfunktion zu berechnen (policy evaluation), und auf der Grundlage dieser Wertfunktion wiederum eine neue, bezüglich der gerade berechneten Wertfunktion 'greedy' Policy π_{k+2} ermittelt (policy improvement), so erhält man das Verfahren der *Policy Iteration*. Wird die darin enthaltene policy evaluation nicht bis zur Konvergenz für jede Policy iteriert, sondern jeder Zustandswert nur genau einmal aktualisiert, erhält man das Verfahren der *Value Iteration*. Hierzu wird Gl. 4.6 als Änderungsvorschrift interpretiert, wobei mit der vorhandenen Wertfunktion $V_k(s)$ die Gl. 4.10 entsteht.

$$V_{k+1}(s) = \max_{a \in A} \sum_{s' \in S} P^{a}_{ss'} \cdot [r^{a}_{ss'} + \gamma V_{k}(s')] \tag{4.10}$$

Diese Methoden der Dynamischen Programmierung gehen von diskreten Zuständen und Zeitschritten sowie perfekten Modellen und perfekter Zustandsinformation von der Umwelt aus; Eigenschaften, die für reale Prozesse in den seltensten Fällen erfüllt sind. Besonders die Anzahl diskreter Zustände, die schon bei relativ kleinen Längen des Zustandsvektors zu sehr großen Suchräumen und damit zu einer sehr schlechten Skalierbarkeit der DP-Methoden führt, verhindert ihre Anwendung.

4.3 RL mit Zustandsaktionswerten

Die in der jüngeren Vergangenheit gebräuchlichsten RL-Methoden verfolgten den Ansatz, für jede Aktion in jedem Zustand einen Wert zu lernen und später genau jene Aktion mit dem größten Wert auszuführen (vgl. Gl. 4.10). Dabei wird unterschieden zwischen Algorithmen mit diskontierten Zustandsaktionswerten ($0 \leq \gamma < 1$, Abschnitt 4.3.1) und solchen ohne Diskontierung ($\gamma = 1$, Abschnitt 4.3.2).

4.3.1 Diskontierte Zustandsaktionswerte

Reinforcement Lernen wird als approximierte Dynamische Programmierung bezeichnet und soll Ansätze zur Lösung [Mahadevan and Kaelbling, 1996] der in Abschnitt 4.2.3 erwähnten Probleme bieten. Value Iteration wird benutzt, um den Wert eines Zustands nach der Interaktion mit der Umwelt aus dem erhaltenen Reward und dem geschätzten Wert des Folgezustandes zu ermitteln.

Entsprechende Algorithmen heißen *Temporal Difference Methoden* (TD) und gehen auf [Sutton, 1988] zurück. Hierbei ergibt sich die Änderung eines Zustandswertes aus der Differenz zwischen dem alten Wert und der Summe aus erfahrenem Reward und dem Folgezustandswert (vgl. Gl.4.11). Dies ist die 'zeitliche Differenz' zwischen dem bisher erwarteten und dem real erfahrenen Return unter der Annahme, daß letzterer eine bessere Näherung des tatsächlichen Wertes sei, da er den realen Reward enthält.

Ursprünglich wurde ein *Adaptive Heuristic Critic* (AHCbenutzt, um die Zustandswertfunktion zu berechnen und so dem *Policy net* oder Adaptive Search Element (ASE) die Aktionsauswahl zu ermöglichen. Die beiden Komponenten werden mit der Policy Iteration abwechselnd oder auch parallel gelernt. Aus einer 'greedy' Policy ergibt sich die neue Zustandswertfunktion mit Gl. 4.11.

$$V_{k+1}(s) = V_k(s) + \alpha(r + \gamma V_k(s') - V_k(s));\ 0 < \alpha \leq 1 \tag{4.11}$$

Verläßt der Agent einen Zustand s, so ergibt sich der neue Wert dieses Zustands aus einem Bruchteil α (Lernrate) von $r + \gamma V_k(s')$, wo r das erhaltene Reward und $V_k(s')$ der geschätzte Wert des erreichten Folgezustandes ist, und einem Anteil $1 - \alpha$ aus dem alten Wert $V_k(s)$.

Ausgehend von einem Folgezustand s'_E, der ein Endzustand und dessen Wert gegeben und damit wahr ist, ergibt sich aus dem Term $V_k(s'_E)$ in Gl. 4.11 eine Annäherung des Wertes von dessen unmittelbarem Vorgängerzustand $V_{k+1}(s)$ an seinen wahren Wert, und in der Folge auch für dessen Vorgänger. Das Wirken der Vorschrift kann man sich als Ausbreitung der wahren Zustandswerte auf je einen Vorgängerzustand pro Durchlauf vorstellen.

Der AHC lernt also die Zustandswertfunktion als eine Kombination von unmittelbar nach Aktionsausführung erhaltener und nach Abschluß der Aktionsfolge erfahrener Rewards. Damit können auch stark verzögert eintretende Konsequenzen einer Aktion (der Reward wird z.B. erst nach Durchlaufen der gesamten Folge vom Endzustand geliefert, wie z.B. einer Fahrt in einem Labyrinth) schon im aktuellen Zeitschritt von der Policy berücksichtigt werden. Das bedeutet aber auch, daß zur Ermittlung der wahren Zustands-

werte umso höhere Trainingszeiten benötigt werden, je länger die durchlaufene Aktionsfolge ist und je kleiner die Werte der Lernrate α gewählt wurden. Eine höhere Trainingszeit bei langen Aktionsfolgen ergibt sich aus dem geringen Horizont von nur einem Schritt, nämlich nur dem Vorgängerzustand, pro Aktion.
Die Ausbreitung der Zustandswerte kann durch Verwendung von *Eligibility Traces* auf Teile oder auch die gesamte Folge bereits erlebter Zustände erweitert werden, wodurch die erfahrenen Informationen effizienter genutzt werden.

4.3.1.1 Eligibility Traces

Das Verfahren wurde daher um so genannte *Eligibility Traces* erweitert, oder richtiger, das obige Verfahren stellt die Realisierung eines TD(λ)-Verfahrens (Gl. 4.12) mit $\lambda = 0$ dar.

$$V_{k+1}(s) = V_k(s) + \alpha(r + \gamma V_k(s') - V_k(s)) \cdot l(s) \tag{4.12}$$

Im Unterschied zu Gl. 4.11 werden in Gl. 4.12 die Werte aller Zuständegeändert, deren Traces $l(s_u) > 0$ sind, anstatt nur die des unmittelbar vorhergehenden. Dadurch kann der Wert eines Zustandes mit Änderungen von Zustandswerten aktualisiert werden, die nach dem unmittelbar folgenden Zustand erreicht wurden. Dies beschleunigt die Ausbreitung der wahren Zustandswerte und damit die Lerngeschwindigkeit des Algorithmus.
Die Eligibility Traces verkörpern, wie oft (Häufigkeit) oder vor wievielen Zeitschritten (Aktualität) ein Zustand eingenommen wurde. Insofern bilden die Traces die Vorgeschichte des aktuellen Zustandes ab und ermöglichen daher auch den Einsatz von RL-Methoden in Problemen, die die MARKOW-Eigenschaft verletzen. Eligibility Traces können auf zwei Arten aktualisiert werden: als *akkumulierende* (Gl. 4.13) oder als *ersetzende* (Gl. 4.14).

$$l(s,t+1) = \begin{cases} \gamma\lambda l(s,t) + 1 & \text{falls } s = \text{ aktueller Zustand} \\ \gamma\lambda l(s,t) & \text{sonst} \end{cases} \tag{4.13}$$

$$l(s,t+1) = \begin{cases} 1 & \text{falls } s = \text{ aktueller Zustand} \\ \gamma\lambda l(s,t) & \text{sonst} \end{cases} \tag{4.14}$$

$$\text{mit} \quad 0 \leq \lambda \leq 1; \; 0 \leq \gamma < 1$$

Akkumulierende Traces (Gl. 4.13) werden bei jedem Durchlauf durch den betreffenden Zustand s erhöht, und ersetzende Traces (Gl. 4.14) bei jedem Durchlauf durch den Zustand zurückgesetzt. Die Traces für alle andere Zustände verringern sich bei beiden Arten. Während akkumulierende Traces sowohl die Häufigkeit, als auch die Aktualität eines Zustandes aufzeichnen, leisten die ersetzenden dies nur für die Aktualität. SINGH et al. [Singh and Sutton, 1996] untersuchen die Einflüsse der beiden Methoden auf die resultierenden Zustandswerte theoretisch und stellen fest, daß den ersetzenden Traces der Vorzug zu geben sei, da sie im statistischen Sinne korrektere Ergebnisse aus dem verfügbaren Reward lieferten. Darüberhinaus konvergiert die Zustandswertfunktion für nahezu alle λ schneller, robuster und auf bessere Ergebnisse. So kann TD(λ) mit akkumulierenden Traces beim Lernen instabil werden, da die Traces ohne Grenzen wachsen können, oder sogar schlechtere Ergebnisse verursachen als TD(0) [Baird, 1995].

4.3.1.2 *Q-Learning* und *Sarsa*

Anstelle der zwei Komponenten AHC und ASE kann TD(λ) auch direkt benutzt werden, indem nicht mehr die Werte für gesamte Zustände gelernt werden, sondern Werte für Zustands-Aktionspaare, wie in den Gleichungen auf der rechten Seite der Wertfunktionen (z.B. Gl. 4.4) angedeutet. Indem die Aktion mit dem maximalen Zustands-Aktions- oder Q-Wert gewählt wird, können die Funktionen von Aktionswahlmodul und Zustandswertfunktion vereinigt werden. So erhält man den wahrscheinlich am häufigsten angewendeten RL-Algorithmus, *Q-Learning* nach WATKINS [Watkins, 1989]. Das Lernen der Q-Werte erfolgt nach Gl. 4.15, die zwangsläufig starke Ähnlichkeiten zu Gl. 4.11 aufweisen muß, da beide ein TD-Verfahren realisieren. Gl. 4.15 beschreibt den Originalalgorithmus, der ein TD(0)-Verfahren realisiert.

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(r + \gamma \max_{a' \in A} Q_k(s',a') - Q_k(s,a)) \tag{4.15}$$

Beim Q-Lernen wird eine optimale Wertfunktion $Q^*(s,a)$ gelernt, indem der Q-Wert für ein Zustands-Aktionspaar $Q_{k+1}(s,a)$ mit dem Reward r, das bei der Aktion zum Folgezustand erfahren wurde, und dem geschätzten maximalen Wert aller Zustands-Aktionspaare im Folgezustand $\max_{a' \in A} Q_k(s',a')$ aktualisiert wird. Die ausgeführte Policy legt fest, welche Zustands-Aktionspaare tatsächlich eingenommen und gelernt werden. Würde ausschließlich der 'greedy'-Policy gefolgt, dann bestimmt die Initialisierung der Q-Werte die Aktionswahl. Aus pragmatischer Sicht könnte der Agent, statt der tatsächlich besten Aktion, immer nur diejenige ausführen, die zufällig als erste einen besseren Wert als die initialisierten erreichte. Somit würden die anderen Zustands-Aktionspaare nie erfahren und damit die Konvergenzbedingung, daß alle Zustands-Aktionspaare getestet werden, nicht erfüllt. Daher muß die Policy eine Explorationskomponente (siehe Abschnitt 4.5) enthalten, die unabhängig von der besten eine andere Aktion ausführen läßt. Deshalb wird mitunter eine Aktion ausgewählt, die nicht der Policy entspricht, weshalb SUTTON *Q-Learning* als 'Off-policy'-Verfahren [Sutton, 1997] bezeichnet. Zur Aktualisierung der Zustandsaktionswertfunktion wird eine andere Aktion benutzt, als im nächsten Schritt dann wirklich ausgeführt wurde, nämlich die hypothetisch beste.

Das entsprechende 'On-policy'-Verfahren wird *Sarsa* [Rummery and Niranjan, 1994] genannt, da es zur Verbesserung nur das tatsächlich erlebte Quintupel (s,a,r,s',a') verwendet. Diese Quintupel gab dem Verfahren auch seinen Namen (Gl. 4.16).

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(r + \gamma Q_k(s',a') - Q_k(s,a)) \tag{4.16}$$

Diese Vorschrift erzeugt so eine bessere Abschätzung für den Return des Zustands s ($Q^\pi(s,a)$ statt $\max_a Q(s,a)$), zumal letzteres wahrscheinlich zu einer zu hohen Schätzung des Wertes führt. Weiterhin sollte sie auch zu einer Beschleunigung des Lernens der Wertfunktion führen, wenn Eligibility Traces verwendet werden, da beim *Q-Learning* im Falle der Ausführung einer explorativen, also non-policy Aktion aus Konvergenzgründen die Trace abgebrochen ($\lambda = 0$ gesetzt) werden muß. Beim *Q-Learning* dürfen der durch Ausführung einer non-policy Aktion erfahrene Reward und der Folgewert nicht zur Verbesserung zurückliegender Werte benutzt werden, abgesehen vom unmittelbaren Zustands-Aktionspaar, weil TD ansonsten eine falsche Summe erzielen würde. Im Gegensatz dazu gestattet *Sarsa* die gleichzeitige Verwendung der Eligibility Traces und einer beliebigen Explorationsstrategie unter effektiver Nutzung der dabei erhaltenen Information. Gl. 4.17

wird nicht nur nach direkter Nutzung des Zustands-Aktionspaares eingesetzt, sondern nach *jedem* Zustands-Aktionspaar, wobei $l(s_u, a_u)$ mitunter 0 sein kann.

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(r + \gamma Q_k(s',a') - Q_k(s,a)) \cdot l(s,a) \tag{4.17}$$

In [Wiering and Schmidhuber, 1998] wird 'On-policy' Lernen unter Nutzung von 'lazy learning' erweitert, bei dem eine Verbesserung der Zustandsaktionswertfunktion erst dann ausgeführt wird, wenn der Wert des betreffenden Zustands-Aktionspaares tatsächlich zur Aktionswahl benötigt wird. Mit diesem Vorgehen wird eine Komplexität erreicht, die linear von der Anzahl der Aktionen abhängt. Die resultierende Beschleunigung ergab den Namen *Fast Online Q(λ)*. Damit sollte der Ansatz ein schnelleres Lernen erlauben als eine konventionelle Implementation von Q(λ) oder *Sarsa*(λ).

4.3.2 Nichtdiskontierte Zustandsaktionswerte: Average Reward

Ein Verfahren, das eine an das *Q-Learning* angelehnte Variante des Lernens mit nicht diskontierten Rewards ($\gamma = 1$) darstellt, ist *R-Lernen* [Schwartz, 1993, Mahadevan, 1996a]. Dieser Algorithmus erzeugt daher gain-optimale Policies.

Beim *R-Lernen* stellt der Zustandsaktionswert $R^\pi(s,a)$ den durchschnittlichen angepaßten Wert einer Aktion a im Zustand s entsprechend der Policy π dar. Die R-Werte werden durch die Lernregel in Gl. 4.18 aktualisiert:

$$R_{k+1}(s,a) = R_k(s,a) + \alpha(r - \rho_k + \max_{a'} R_k(s',a') - R_k(s,a)) \tag{4.18}$$

Diese Lernregel unterscheidet sich von der des *Q-Learning* nur dadurch, daß der durchschnittliche Reward ρ_k vom aktuellen Reward r subtrahiert und der Wert des Folgezustandes $R(s',a')$ nicht diskontiert wird. Der durchschnittliche Reward ρ_k wird nach Gl. 4.19 aus dem erfahrenen Reward und der Differenz der Werte der beiden aufeinanderfolgenden Zustände geschätzt:

$$\rho_{k+1} = \rho_k + \beta_k(r - \rho_k + \max_{a'} Q_k(s',a') - \max_a Q_k(s,a)) \tag{4.19}$$

Q-Learning diskontiert zukünftige Rewards und zieht Aktionen, die kurzfristige und möglicherweise mittelmäßige Rewards erzielen, solchen vor, die langfristig hohe Rewards erreichen. *Q-Learning* maximiert den diskontierten kummulativen Return, R-Lernen dagegen den *durchschnittlichen* Return. R-Lernen würde zum Beispiel eine Policy wählen, die anfangs keinen, später aber hohe Rewards erzielt und somit die durchschnittlichen Returns bis dahin *pro Schritt* maximiert. *R-Lernen* ist gain-optimal und setzt damit eine Wertefunktion und ein Optimalitätskriterium entsprechend Gl. 4.5 um. Neben dem durchschnittlichen Reward ρ_k unterscheidet es sich durch $\gamma = 1$ von *Q-Learning*, für das in diesem Fall keine Konvergenzgarantien mehr gelten. Allerdings wurde auch für R-Lernen noch keine Konvergenz bewiesen.

Eine modellbasierte Weiterentwicklung ist *H-Lernen* [Tadepalli and Ok, 1998]. Der Algorithmus schätzt die Parameter des Aktionsmodells und die optimale Policy simultan. Unter Verwendung des Aktionsmodells kann der Algorithmus eine bias-optimale Policy

erzeugen. Ein weiterer Vorteil des Verfahrens ist, daß es unabhängig von der Explorationsstrategie immer die optimale Policy lernt. Die Exploration beeinflußt nur die Geschwindigkeit, in der die optimale Policy gelernt wird, nicht jedoch deren Optimalität.

Beim *H-Lernen* wird die Schätzung des durchschnittlichen Rewards ρ_{k+1} (Gl. 4.20) durch Einbeziehung der angepaßten Rewardsummen, die in $h(s)$ gespeichert werden, geändert. Diese Biasabweichungen beinhalten die Summen über die erfahrenen Rewards abzüglich des durchschnittlichen Rewards (Gl. 4.22), wodurch die für einen bias-optimalen Algorithmus typischen verschachtelten Gleichungen entstehen.

$$\rho_{k+1} = \rho_k + \beta_k(r(s,a) - \rho_k - h(s) + h(s')) \tag{4.20}$$

$$\beta_{k+1} = \frac{\beta_k}{\beta_k + 1} \tag{4.21}$$

$$h(s) = \max_a \{r(s,a) + \sum_{s' \in S} P^a_{ss'} h(s')\} - \rho_k \tag{4.22}$$

4.3.3 Residuale Gradienten

Die bisher diskutierten Algorithmen, insbesondere *Q-Learning* und *Sarsa*, gehen in ihrer ursprünglichen Formulierung von diskreten Zuständen aus, für die die Werte in Form einer Tabelle gespeichert werden können. Dabei maximieren sie die Zustandswerte aus Gl. 4.4 direkt und mit bewiesener Konvergenz unter der Randbedingung eines MDP und fallender Lernraten. Aus diesem Grunde nennt BAIRD [Baird, 1999] sie *direkte Algorithmen*.

Bei sehr großen Zustandsräumen jedoch ist eine Speicherung in einer Tabelle nicht praktikabel und eine Implementation mit Funktionsapproximatoren wie Multilayer Perzeptrons (MLP) [Rumelhart and McClelland, 1986] bietet sich an. Hierbei werden einer mehrschichtigen Struktur aus einfachen Verarbeitungsknoten Paare von Eingangs- und gewünschten Ausgangsvektoren präsentiert. Die Verbindungen zwischen den Knoten, die Gewichte, werden durch Gradientenabstieg über der Differenz zwischen gewünschtem und erzieltem Ausgangsvektor so angepaßt, das letztere erzeugt werden. Die Informationen über Zustände und Zustandswerte sind bei diesen Approximatoren strukturell also in den Gewichten enthalten.

Wird ein solches überwachtes neuronales Netz eingesetzt, muß dessen Gewichtsvektor $\mathbf{w}$ durch ein direktes Training die Zustandswerte lernen. Die entsprechende Gewichtsänderung $\triangle_d \mathbf{w}$ ist in Gl. 4.23 beschrieben. Sie ergibt sich direkt als Produkt aus dem TD(0)-Fehler und der Ableitung des Zustandswertes nach den Gewichten.

$$\triangle_d \mathbf{w} = \alpha(r + \gamma V(s') - V(s)) \frac{\partial V(s)}{\partial \mathbf{w}} \tag{4.23}$$

Für solche Funktionsapproximatoren allerdings kann die Konvergenz der zustandswertbasierten Algorithmen nicht mehr bewiesen werden und es existieren darüberhinaus Beispiele für eine Divergenz auf relativ simplen MDP's [Baxter and Bartlett, 2001]. Trotzdem wurden Algorithmen wie *Q-Learning* oft erfolgreich mit Funktionsapproximatoren implementiert und eingesetzt [Lin, 1992, Humphrys, 1997]. Die originale Literatur zu $TD(\lambda)$ [Sutton, 1989] geht ebenfalls von einer solchen Umsetzung aus.

Um auch in diesem Fall die Konvergenz zu garantieren, muß die Änderungsvorschrift für die Gewichte des Funktionsapproximators geändert werden. Ziel des Funktionsapproximators muß sein, den Fehler der Bellman-Gleichung nach Gl. 4.24 zu minimieren, der sich aus dem Erwartungswert des Quadrates der Abweichung zwischen Return (Summe aus erfahrenem Reward und diskontiertem Wert des Folgezustandes) und dem bisherigen Zustandswert über alle N zu betrachtenden Zustände ergibt.

$$e = \frac{1}{N} \sum_s E\left\{(r + \gamma V(s') - V(s))^2\right\} \tag{4.24}$$

Ist dieses Ziel erreicht, repräsentiert die im Funktionsapproximator gelernte Policy auch eine optimale Policy und erfüllt somit auch das Ziel der Rewardmaximierung nach Gl.4.4. Gl.4.23 stützt sich auf nur eine Probe des gesuchten Gradienten, wo zwei notwendig wären (siehe [Baird, 1999] zu 'unbiased estimators'). Weiterhin geht die Schätzung für den Wert des Folgezustandes $V(s')$ ein, der natürlich gleichfalls vom zu ändernden $\mathbf{w}$ abhängt, dessen partielle Ableitung jedoch vernachlässigt wurde.
Aus Gl. 4.24 ergibt sich der 'residuale Gradient' $\triangle_{rg}\mathbf{w}$ nach Gl.4.25 für die Gewichte des Funktionsapproximators. Erst er gewährleistet ein korrektes Lernen der gesamten Rewardfunktion mit garantierter Konvergenz.

$$\triangle_{rg}\mathbf{w} = -\alpha\left[r + \gamma V(s_1') - V(s)\right]\left[\gamma\frac{\partial V(s_2')}{\partial \mathbf{w}} - \frac{\partial V(s)}{\partial \mathbf{w}}\right] \tag{4.25}$$

Dieser Gradient schätzt mit zwei Proben (den beiden unabhängigen Folgezuständen s_1' und s_2' des Zustandes s) und einer korrekten partiellen Ableitung den exakten Gradienten ab. Algorithmen, die diesen Gradienten benutzen, nennt BAIRD *residuale Gradientenalgorithmen.* Diese Algorithmen weisen zwar garantierte Konvergenz auch mit Funktionsapproximatoren auf, lernen jedoch sehr viel langsamer als direkte Algorithmen. Daher strebt BAIRD durch eine gewichtete Linearkombination der Gewichtsänderungen aus den Gl.4.23 und Gl.4.25 maximale Konvergenzgeschwindigkeit bei garantierter Konvergenz an:

$$\triangle_{r}\mathbf{w} = -\alpha\left[r + \gamma V(s_1') - V(s)\right]\left[\phi\gamma\frac{\partial V(s_2')}{\partial \mathbf{w}} - \frac{\partial V(s)}{\partial \mathbf{w}}\right] \tag{4.26}$$

Dabei soll ϕ so gewählt werden, daß der Winkel zwischen dem Vektor der resultierenden Gewichtsänderung $\triangle_r\mathbf{w}$ und dem der Gewichtsänderung $\triangle_{rg}\mathbf{w}$ spitz bleibt, was eine notwendige Bedingung für die Konvergenz ist. Entsprechende Algorithmen nennt BAIRD *residuale Algorithmen.*

4.3.4 RL mit Policysuche

Alternative Ansätze des RL entwickeln den Ansatz der *Policy Iteration* weiter und gelangen zu einer gradientenbasierten Formulierung. Ausgangspunkt für diese im Verhältnis zu den zustandswertbasierten Algorithmen bisher wenig verbreiteten Methoden ist *REINFORCE* [Williams, 1992]. Hier werden nun nicht mehr die Werte der Aktionen in verschiedenen Zuständen über mehrere Versuche gelernt, und dann diejenige Aktion mit größtem Wert ausgeführt, sondern die Wahrscheinlichkeiten der Ausführung der Aktionen in den

einzelnen Zuständen direkt gelernt. Da hier nicht mehr der Wert der Aktion in einem Zustand rekursiv aus dem Wert des Folgezustandes ermittelt werden kann, womit diesem Algorithmus Erfahrungen aus vergangenen Durchläufen nicht mehr zur Verfügung stehen, müssen immer die gesamten durchlaufenen Sequenzen zum Lernen herangezogen werden.

Ziel ist, wie oben, die Maximierung des kumulierten, diskontierten Returns. WILLIAMS [Williams, 1992] und PESHKIN [Meuleau et al., 2001] implementierten REINFORCE mit Tabellen, während BAIRD feststellte, daß sich aus den Sequenzen Gradienten berechnen ließen, die den residualen Gradienten sehr ähnlich wären. Entsprechend kombinierte er beide Varianten, woraus *VAPS* (Value and Policy Search) entstand. Mit diesem Algorithmus können beliebige Methoden der Zustandswertberechnung mit solchen der Policysuche gewichtet kombiniert und auf Funktionsapproximatoren implementiert werden (siehe Tab. 4.1), wobei ihre Konvergenz gesichert ist.

$\triangle \mathbf{w} = -\alpha \left[\frac{\partial e(s)}{\partial \mathbf{w}} + e(s)T\right]$ $\triangle T = \frac{\partial}{\partial \mathbf{w}} \ln(P(a\|s))$	$e_{Sarsa}(s) = \frac{1}{2}E^2 \{r + \gamma Q(s',a') - Q(s,a)\}$
	$e_{Q\text{-}Learning}(s) = \frac{1}{2}E^2 \{r + \gamma \max'_a Q(s',a') - Q(s,a)\}$
	$e_{Value-Iteration}(s) = \frac{1}{2}[\max_a E\{r + \gamma V(s')\} - V(s)]^2$
	$e_{Sarsa-policy}(s) = (1-\beta)e_{Sarsa}(s) + \beta(b - \gamma^t r)$

Tabelle 4.1: *Der allgemeine VAPS Algorithmus (links) und verschiedene Realisierungsformen davon (rechts). Dieser Algorithmus beinhaltet wertebasierte und policysuchende Verfahren - und deren Kombination. Dazu kann die Konvergenz für jeden Fall garantiert werden. T ist eine Trace über die Wahrscheinlichkeiten der Aktionsausführung in verschiedenen Zuständen, also die zu lernende Policy. Der zu minimierende Fehler e kann mit verschiedenen Ansätzen ermittelt werden.*

4.4 Gradientenbasierte Policysuche

Jüngste Arbeiten im RL konzentrieren sich auf die direkte Suche einer Policy unter Benutzung des Rewardgradienten (direct gradient descent), ähnlich den eben in Abschnitt 4.3.4 vorgestellten Verfahren oder sogar direkt aus ihnen hervorgegangen. Ihr großer Vorteil liegt in einer Behandlung des essentiellen Problems nur teilweise beobachtbarer Zustandsräume (Partially Observable Markov Decision Processes (POMDP)). Dazu setzen sie einerseits einen oder mehrere Speicher für erlebte Zustandübergänge und Rewards ein, die den in Abschnitt 4.3.1.1 diskutierten Traces ähneln. Durch diese Betrachtung des gesamten durchlaufenen Trials können sowohl sie als auch $Sarsa(\lambda)$ Informationen über die wahren Zustandswerte extrahieren. Andererseits benutzt die Suche nach Policygradienten

(PG) darüberhinaus die Information des Rewardgradienten und einen meist finiten Graphen. In diesem endlichen Graphen wird der unter Umständen infinite POMDP abgebildet und solche Trajektorien besucht, die ein Mindestmaß an Realisierungswahrscheinlichkeit aufweisen. Mit Hilfe des so gewonnenen stochastischen Gradienten werden garantiert optimale Trajektorien gefunden, allerdings ist die Optimalität durch die Begrenzung der durchsuchten Trajektorien lokal: der Algorithmus konvergiert garantiert zu einer lokal optimalen Trajektorie für einen POMDP.

Beispiele für entsprechende Algorithmen sollen wegen der Komplexität und Andersartigkeit ihrer Grundlagen hier nur aufgezählt, aber nicht erläutert werden. Sie schließen unter anderen *Gradient POMDP* (GPOMDP, [Baxter and Bartlett, 2001]), *Policy Iteration with Function Approximation* (PIFA, [Sutton et al., 2000]), *Finite State Controlers*, (FSC, [Meuleau et al., 1999]) und *Action Transition Policy Gradient* (ATPG, [Grudic and Ungar, 2000]) ein.

Die PG ermitteln eine Policy $\pi(s, a, \theta)$, die die Wahrscheinlichkeit wiedergibt, daß die Aktion a im Zustand s ausgeführt wird. Der Parametervektor θ repräsentiert alle modifizierbaren Parameter des gewählten Funktionsapproximators (z.B. die Gewichte eines Neuronalen Netzes). Dann wird die Schätzung $\widehat{\partial p}/\partial\theta$ des Performance-Gradienten $\partial p/\partial\theta$ gelernt, der dafür benutzt wird, die Policy in eine Richtung zu bewegen, die den Reward erhöht. Also lernt der Agent, indem er erst diesen Gradienten $\widehat{\partial p}/\partial\theta$ der momentanen Policy schätzt und dann die Parameter θ wie folgt verändert:

$$\theta_{t+1} = \theta_t + \alpha \frac{\widehat{\partial p}}{\partial \theta}; \alpha \in [0 \ldots 1) \tag{4.27}$$

Für die Attraktivität dieser Algorithmen sprechen mehrere Gründe:

- Die Formulierung des PG benutzt Funktionsapproximatoren (Neuronales Netz, Entscheidungsbaum etc.) für die Policy des Agenten, was das Trainingszeitproblem des RL für große bzw. kontinuierliche Zustandsräume löst.
- Der Berechnungsaufwand für die Gradienten der Rewardfunktion ist proportional zur Anzahl der Parameter, die die Policy beschreiben und damit deutlich geringer als das exponentiale Wachstum der Berechnungskomplexität, das für bisherige Algorithmen wie *Q-Learning* oder *Sarsa* vorauszusetzen ist. Das macht diese Formulierung auch interessant für hochdimensionale Probleme.
- Die PG-Algorithmen sind nachweislich konvergent für Funktionsapproximatoren und POMDP. Für alle bisher bekannten PG-Algorithmen wurde theoretisch nachgewiesen, daß sie zu einer lokal optimalen Policy konvergieren. Auch wenn bestimmte Algorithmen, wie z.B. VAPS, nicht garantieren können, daß ein lokales Optimum erreicht wird, so garantieren sie jedoch die Konvergenz zu einer (möglicherweise suboptimalen) Policy.

Alternativ zu TD- und Policysuch-Verfahren werden in *Globalized Dual Heuristic Programming* (GDHP, [Werbos, 1992a]) und *GREMLIN* [Caironi, 1997] mit Hilfe eines geeigneten Prozeßmodells auch zusätzlich die Ableitungen der Zustands- bzw. Aktionswertfunktion nach den Komponenten des Zustandsvektors gelernt. Dadurch wird der Informationsmangel von RL- gegenüber überwachten Verfahren gemindert und die Problematik

der Aktionsexploration (siehe unten) umgangen. Diese Verfahren stützen sich ebenfalls auf Funktionsapproximatoren wie Perzeptrons und sind daher in kontinuierlichen Zustands- und Aktionsräumen einsetzbar. Durch die Modelle ist ihnen auch die Benutzung des exakten Gradienten, die direkte Ermittlung der optimalen Werte der Aktionsparameter und eine Planung möglich.

4.5 Das Exploration-Exploitations-Dilemma

Neben der notwendigen und wünschenswerten Exploration besteht das Ziel des Agenten darin, die Aufgabe zu erfüllen, also eigentlich seinen aktuellen Wissensstand zu nutzen (Exploitation) und die beste Aktion auszuführen. Dieser Widerspruch wird als das 'Explorations/Exploitations-Dilemma' bezeichnet. Die beste Balance hängt unter anderem von der Länge der Aufgabe ab, also z.B. der Anzahl von Aktionen bis zu einem Endzustand, weil sich leicht suboptimale Entscheidungen wenig auf den Return kurzer Aktionsfolgen auswirken, jedoch stark auf längere. Abbildung 4.4 gibt einen Überblick über die im folgenden erläuterten Explorationsstrategien.

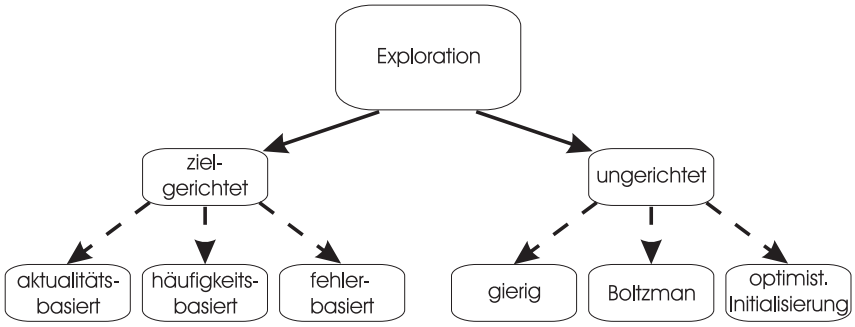

Abbildung 4.4: *Taxonomie der Explorationsverfahren im RL.*

Würde ein Agent ausschließlich der 'greedy'-Policy folgen, und immer die beste Aktion wählen, dann hinge die Rangfolge der Aktion von der Initialisierung der Werte ab. Sind zum Beispiel alle Werte gleichgroß und kleiner als die kleinsten erreichbaren tatsächlichen Werte initialisiert, dann wird die als erste gewählte Aktion immer ausgeführt werden, da ihr Wert bei der ersten Aktualisierung größer wird als alle anderen. Sind sie nicht gleichgroß initialisiert, hängt die Aktionswahl von den Größen der Wertänderung und der Initialisierung ab. Es kann aber keine Konvergenz zu den wahren Werten garantiert werden, weil nicht garantiert werden kann, daß alle Aktionen getestet werden. Sind dagegen alle Werte größer als die größten erreichbaren initialisiert, werden alle Aktionen mindestens einmal getestet, da die Aktualisierung sie mindestens auf die maximal erreichbare Größe verringert. Dieser Fall wird 'Optimistische Initialisierung' genannt und unten diskutiert.

Eine einfache Explorationsstrategie wäre, in einem kleinen Prozentsatz der Fälle eine zufällig gewählte Aktion ausführen zu lassen, in der Mehrzahl jedoch die beste Aktion (*semi-uniforme Exploration* oder *ϵ-greedy Exploration*, worin ϵ derjenige Prozentsatz der Fälle ist, indem eine zufällige Aktion ausgeführt wird). Dies allerdings bedeutet, daß die Aktionen unabhängig von der Höhe ihres Wertes ausgewählt würden und sich der Agent bei vollständig gelernter Wertfunktion mitunter (wissentlich) suboptimal verhält.

Eine weitere, einfache Explorationsstrategie besteht in der Initialisierung der Q-Werte etwas über dem maximal von ihnen erreichbaren Wert in Kombination mit der 'greedy'-Strategie (*optimistische Initialisierung*, siehe auch Abschnitt über Berechnungskomplexität weiter unten) [Kaelbling et al., 1996]. Das hätte zur Folge, daß sich der Wert suboptimaler Aktionen schnell verringern würde und so alle Aktionen zumindest einmal getestet würden. Für die Verfahren der asynchronen Dynamischen Programmierung, bei denen immer nur ein Teil des gesamten Zustandsraumes aktualisiert wird, ist diese Initialisierung Voraussetzung für die erfolgreiche Konvergenz [Weiß, 1999]. Aber auch diese Strategie ist nicht steuerbar.

Eine weitaus verbreitetere Alternative realisiert die naheliegende Abnahme der Zufälligkeit der Aktionsauswahl mit fortschreitendem Lernen und Beachtung des erwarteten Rewards: *Boltzman Exploration* (Gl. 4.28). Die Wahrscheinlichkeit der Auswahl einer Aktion a im Zustand s ergibt sich aus der Exponentialfuntion des Quotienten aus dem Q-Wert der Aktion und der Boltzman-Temperatur T, normiert mit der entsprechenden Summe für alle Aktionen.

$$P(a) = \frac{\exp(Q(s,a)/T)}{\exp(\sum_{a \in A} Q(s,a)/T)} \tag{4.28}$$

Durch graduelle Verringerung der Boltzmann-Temperatur T kann die Wahrscheinlichkeit $P(a)$ der Wahl der besten Aktion erhöht werden, während bei hohen T alle Aktionen nahezu die gleiche Auswahlchance haben.

Geht man jedoch davon aus, daß es zur Erfüllung der Aufgabe nicht notwendig und daher nicht immer sinnvoll ist, den gesamten dem Agenten zur Verfügung stehende Eingangsraum zu erkunden, erhebt sich die Frage nach einer zielgerichteten Exploration. Anders ausgedrückt: welche Aktion brächte den höchsten Informationsgewinn und stellt damit die effektivste Exploration dar? [Thrun, 1992] stellt sowohl eine Taxonomie (siehe Abb. 4.4) als auch eine Übersicht über Explorationsstrategien auf. Darin argumentiert Thrun, daß zielgerichtete Exploration effektiver als die bisher diskutierten, ungerichteten, seien.

Zielgerichtete Exploration stützt sich meist auf die Häufigkeit oder Aktualität der Auswahl einer Aktion bzw. eines Zustandes, einen erwarteten oder zuletzt erfahrenen hohen Schätzungsfehler oder eine Kombination der genannten. So kann jeder Zustand mit einem Zähler der Häufigkeit seines Auftretens versehen sein und eine Aktion ausgeführt werden, die den Agenten in einen selten aufgetretenen Zustand bringt (*häufigkeitsbasierte Exploration*). Der Effekt besteht in einer Erkundung schlecht explorierter Zustände und damit in einer höheren Gewißheit über deren Wert. Andererseits kann für jeden Zustand aber auch die Zeit mitgeführt werden, die seit seinem letzten Auftreten vergangen ist (*aktualitätsbasierte Exploration*). Dem entsprechend werden die am längsten zurückliegenden Zustände bevorzugt erkundet. Da akkumulierende Traces (vgl. Abschnitt 4.3.1.1) diese Information entahlten, könnten sie für diese Exploration herangezogen werden.

Die Annahme, daß große Fehler bei der Schätzung eines Wertes gleichbedeutend sind mit einer ungenügenden Exploration dieser Zustände, führt zur *fehlerbasierten Exploration*. Solche Fehler werden beim Lernen der Werte bestimmt. Darüber hinaus läßt sich auch die Entwicklung des Fehlers bei mehreren Durchläufen durch den Zustand auswerten. Ob dies z.B. seinen Anstieg oder seine Varianz betrifft, hängt von der Schätzvorschrift ab.

Eine Varianz kann nur bestimmt werden, wenn der Agent den gleichen Zustand mehrmals durchlaufen muß, bevor die Schätzung des Wertes verbessert wird. Letztlich können wiederum Methoden der DP angewandt werden, um auch die Exploration zu planen.

Aus jeder der verschiedenen Explorationsstrategien läßt sich ein Maß D_{explor} bestimmen, das gegen ein ähnliches Maß $D_{exploit}$ für die Exploitation abgewogen werden muß, welches sich aus den Zustands-Aktionswerten ergibt. Diese Maße dienen ausschließlich der Charakterisierung der Notwendigkeit einer Exploration. Ihre Berechnung richtet sich nach dem verwendeten Explorationsprinzip und den Zustands-Aktionswerten, D_{explor} könnte sich zum Beispiel aus dem TD-Fehler eines Zustands-Aktionspaares ergeben.

Eine statische, lineare Kombination wie $D = (1-\beta)D_{explor} + \beta D_{exploit}$ kann benutzt werden, diejenige Aktion mit maximalem D zu finden, oder eine allgemeine Entscheidung zwischen Exploration und Exploitation herbeizuführen. Meist ist ein konstantes β jedoch der Aufgabe nicht angemessen oder beide $D_{explor|exploit}$ halten sich die Waage. Daher wird zur dynamischen Bestimmung von β die jeweilige 'Motivation' $d(s,a)$ eingeführt (Gl. 4.29), die für jedes Zustands-Aktionspaar zusätzlich gespeichert wird.

$$\begin{aligned} d_{explor} &= E\{D_{explor}|s,a\} \\ d_{exploit} &= E\{-D_{exploit}|s,a\} \end{aligned} \tag{4.29}$$

d_{explor} gibt den erwarteten Explorationsnutzen an, $d_{exploit}$ die erwarteten Kosten. Alternative Möglichkeiten der Aktionswahl zwischen Exploration und Exploitation (Gl. 4.30) bestehen in der *Auswahl* (Gl.4.31) oder dem *dynamischen Schalten* (Gl.4.32).

$$a = \arg\max_{a \in A} D(s,a) \tag{4.30}$$

$$D(s,a) = \begin{cases} 0 & \text{wenn } d_{explor} > d_{exploit} \\ 1 & \text{sonst} \end{cases} \tag{4.31}$$

$$\begin{aligned} D &= (1-\beta_k)f(d_{explor}) - \beta_k f(d_{exploit}) \\ &\text{wo} \quad f(z) \text{ posit.,monot. z.B. } f(z) = \exp(z) \\ &\text{und} \quad \beta_{k+1} = \frac{1 - c_{explor} - c_{exploit}}{1 + \exp(-cD)} + c_{explor} \\ &\text{wo} \quad \beta_0 = 0,5;\ c > 0, const;\ c_{explor}, c_{exploit} \geq 0, c_{explor} + c_{exploit} \leq 1 \end{aligned} \tag{4.32}$$

Die Parameter c_{explor} und $c_{exploit}$ sind Konstanten, die die Auswahlminima für Exploration und Exploitation festlegen: $\beta \in (c_{explor}, 1 - c_{exploit})$, während c eine konstante Verstärkung darstellt.

Die erwähnten Methoden stützen sich auf diskrete Zustände und können daher nicht in kontinuierlichen Räumen verwendet werden. Aus diesem Grund benutzt [Thrun, 1992] ein BackPropagation-Netz, um eine *Competence Map* zu lernen. Eingänge sind hierfür Zustand und Aktion, als Teacherwerte werden die Prädiktionsfehler verwendet. Exploration und Exploitation werden mittels statischer Linearkombination gegeneinander abgewogen. Durch Einstellung eines Normierungsfaktors für die Teacherwerte wird ein variabler Mix der oben beschriebenen Methoden erreicht.

Die Auswahl einer der Aufgabe angemessenen Explorationsstrategie bestimmt in hohem Maße die zum Training einer ausreichend konvergierten Wertfunktion notwendigen Zeit. Sie kann, wie oben beschrieben, unter Umständen exponentiell von der Komplexität der Aufgabe, d.h. der Größe und Auflösung des Eingangs- und Aktionsraumes abhängen.

4.6 Probleme beim Reinforcement Lernen

4.6.1 Teilweise Beobachtbarkeit

Eine entscheidende Frage bei der Anwendung von RL-Algorithmen auf der Grundlage von MDP ist, welche Informationen der Agent über den Zustand der Umwelt s_t erhält. Intuitiv wird der Zustand der Umwelt zum Zeitpunkt t durch Größen, wie z.B die Position eines mobilen Roboters, beschrieben. Beschreibungen realer Systeme jedoch können selten vollständig sein, gerade hinsichtlich der Dynamik eines Systems. Sollten die dem Agenten zur Verfügung gestellten Informationen, also die Observation $o(t) = f(s(t))$, die für den Agenten relevanten Informationen nicht vollständig widergeben können, ist die MARKOW-Eigenschaft nicht erfüllt. Ein simples Beispiel aus einer Simulation ohne Dynamik ist in Abb. 4.5 gegeben. Der Agent soll von S nach G gelangen. Ihm steht die Belegung der Zellen links, rechts, oben und unten an seiner aktuellen Position als Sensorik zur Verfügung. Er kann 1 Schritt nach Norden, Süden, Osten oder Westen gehen. Die Situationen 1 und 3 sind nicht unterscheidbar, verlangen aber unterschiedliche Aktionen.

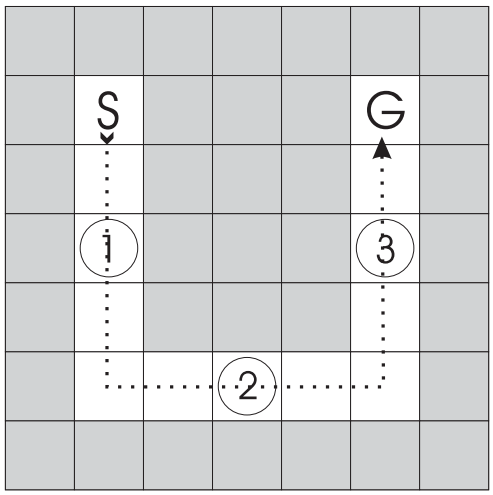

Abbildung 4.5: ***Einfaches Beispiel für einen POMDP.*** *Ein Agent soll von S nach G gelangen. Mit den sensorischen Informationen über die Belegung der unmittelbar benachbarten Zellen sind 1 und 3 nicht unterscheidbar.*

Hier hängt die Wahrscheinlichkeitsverteilung der Folgezustände nach Ausführung einer bestimmten Aktion in einem bestimmten Zustand zum Beispiel von den vorher durchlaufenen Zuständen ab (z.B. Systeme mit Trägheiten wie Ansprech- und Abklingzeiten, bei denen eine Pfadunabhängigkeit nicht mehr gegeben ist). Prozesse, deren Zustände die MARKOW-Eigenschaft nur teilweise erfüllen, werden als *Partially Observable Markov Decision Processes* (POMDP) bezeichnet, da hier nicht direkt beobachtbare oder *hidden states* bzw. *perceptual aliasing* [Whitehead and Ballard, 1990] vorliegen. Da jedoch die in den Zuständen enthaltenen Informationen u.U. nur wenig von der MARKOW-Eigenschaft abweichen, ist es dennoch sinnvoll, zur Approximation und zum Verständnis des Verhaltens des Agenten von der Annahme der MARKOW-Eigenschaft auszugehen. Andererseits ist die Behandlung des Problems der POMDP Gegenstand intensiver aktueller Forschung [Bacchus et al., 1996, Pendrith and McGarity, 1997, Loch and Singh, 1998].

Zur Behandlung des Problems POMDP existieren unterschiedliche Lösungsansätze:

- ignorieren und auf einen 'gutmütigen' POMDP in den interessanten Regionen hoffen
- Benutzung von impliziten Zeitinformationen, wie Traces (TD($\lambda > 0$))[Sutton, 1989] oder der Aktivierungsreihenfolge von Teil-MDP's für Hierarchien im Zustandsraum, durch die der Zustandsraum gröber aufgelöst wird, um einen Gesamt-MDP zu erreichen) (*HQ* [Wiering and Schmidhuber, 1996], siehe Abschnitt 4.7)
- Hinzunehmen expliziter, fester Zeitfenster für Aktionen oder Observationen aus zurückliegenden Zeitschritten (Time Delay Neural Nets, USM [McCallum, 1997])
- Verwendung variabler Zeitfenster zur Reproduktion von Zuständen des Netzes als Eingangsinformation in folgenden Zeitschritten (Rekurrente Neuronale Netze, Utile Distinction Memory [McCallum, 1997])
- PG-Algorithmen (GPOMDP [Baxter and Bartlett, 2000], VAPS [Baird, 1999], PI-FA [Sutton et al., 2000], siehe Abschnitt 4.3.4)
- hierarchische Zerlegung (*MaxQ* [Dietterich, 2000], siehe Abschnitt 4.7)
- Zusammenfassung von Elementaraktionen zu Optionen mit Start- und Endbedingungen (*Makro-Options* [McGovern and Sutton, 1998], siehe Abschnitt 4.8)
- Hierarchie von Auflösungen des Zustandsraumes, sodaß min. eine Auflösung der Markow-Eigenschaft entspricht und eine Reihenfolge von feiner auflösenden Teil-MDP steuert (*FeudalQ* [Dayan and Hinton, 1993])
- Benutzung eines Umweltmodells ([Jennings et al., 1998])

Der Einsatz dieser Algorithmen zum Erwerb von Verhalten wird wichtig, wenn die kontrollierte Laborumgebung verlassen wird und das System sich in nicht nur realen, sondern natürlichen Umgebungen bewegen muß. Für die in dieser Arbeit angestrebten prinzipiellen Untersuchungen der Koordination verschiedener Verhalten wurde diese Problematik zunächst ausgeklammert, sie wird jedoch in Abschnitt 9.2 wieder aufgegriffen.

4.6.2 Komplexität des RL

Eines der herausragendsten Probleme bei der Anwendung von RL-Algorithmen besteht in den extrem langen Trainingszeiten der Agenten, selbst bei einfachen Aufgabenstellungen. Für das Beispiel des *Q-Learning*s sind diese auf die Berechnungskomplexität des Algorithmus zurückzuführen. Wie mehrfach erwähnt, hängt diese exponentiell von der 'Größe' des Eingangsraumes [Koenig and Simmons, 1993], d.h. seiner Dimensionalität und Auflösung, von der Anzahl der in den Zuständen ausführbaren Aktionen, aber auch von der Tiefe der Aufgabe ab. Die Tiefe ergibt sich aus der mittleren Anzahl auszuführender Aktionen entsprechend der optimalen Policy. Diese Komplexität wird dann maximal, wenn die Aufgabenstellung Zielzustände vorsieht, ein Reward erst bei deren Erreichen geliefert wird (verzögertes Reward) und die Zustandswerte mit 0 initialisiert wurden (tabula rasa RL). Dann benötigt ein RL-Algorithmus in einer Markow-Kette bis zu $O(2^n)$ Aktionen

(n =Anzahl der Zustände, 2=Anzahl möglicher Aktionen pro Zustand), bevor der Zielzustand einmal erreicht wird. Wie in [Koenig and Simmons, 1993] gezeigt wird, erreicht man schon dann eine polynomiale Komplexität, wenn der Agent für jede Aktion ein Reward erhält (*immediate Reward*) oder die Wertfunktion *optimistisch initialisiert* wird. In beiden Fällen werden alle Aktionen mindestens einmal getestet und die Werte suboptimaler Aktionen verringern sich schnell.

Vergleichbare Komplexitätsreduktionen werden durch angemessene Explorationsstrategien (siehe oben, [Thrun, 1992]: $O(n^2)$) oder Traces erzielt. Bezüglich des Zeitpunktes einer Adaptation der Wertfunktion wird in [Salustowicz and Schmidhuber, 1998] stark für *online* Lernen argumentiert, bei dem die Wertfunktion nach jeder Aktion geändert wird. Gegenüber *offline* Lernen, bei dem die Änderung erst nach Erreichen eines Zielzustandes ausgeführt wird, steht durch die geänderten Zustandswerte somit jede Erfahrung sofort für folgende Durchläufe zur Verfügung. Dies bewahrt den Agenten davor, negative Erfahrungen unnötig zu wiederholen, muß aber mit einer geeigneten Explorationsstrategie verbunden sein, um ihm bessere Erfahrungen nicht vorzuenthalten. ASADA [Minato and Asada, 1998] verweist als weitere Möglichkeit der Beschleunigung auf *Learning from easy missions* (LEM), bei dem der Eingangsraum zunächst sehr eng um den Zielzustand begrenzt ist und per Hand bei steigender Erfolgsquote des Agenten erweitert wird. Dadurch findet der Agent zunächst relativ schnell den Zielzustand, sodaß sich die wahren Werte der Zustände leicht ausbreiten können. So wird virtuell der Zielzustand in einen Zielraum erweitert, wenn nun anschließend der Eingangsraum erweitert wird.

Der durch ein immediate Reward erzielte Effekt kann erheblich verstärkt werden, wenn der Reward durch einen externen Beobachter partiell gesteuert wird (*learning with external critic* (LEC), *teaching* , supervised RL [Lin, 1992]). Dies kann mehrere Formen annehmen: von einem zusätzlichen Reward für eine bekanntermaßen optimale Aktion (LEC) bis hin zum Vorschreiben einer optimalen Aktion durch den Beobachter (teaching). Beim teaching vertauschen sich die Rollen des Agenten und des Beobachters: der externe Experte wählt die Aktion aus, während der RL-Algorithmus die resultierenden Returns beobachtet. Der RL-Algorithmus wird später selbständig agieren und, aufbauend auf den Erfahrungen aus den Versuchen mit der Führung des Experten, weiter lernen. Diese Realisierungen setzen jedoch voraus, daß der Experte oder Beobachter die internen Repräsentationen des Agenten ausreichend versteht, um mit dem Agenten zu interagieren. Diese Bedingung wird in *advise* [Maclin and Shavlik, 1996] fallengelassen. Die 'Ratschläge' des Experten können quasi-linguistisch formuliert sein. Sie ähneln in gewisser Weise Fuzzy-Regeln. Diese müssen noch in die interne Repräsentation des Agenten übersetzt werden, der daraufhin den Wert der 'Ratschläge' prüft. LIN [Lin, 1992] schlägt *experience replay* vor, um die vom Agenten durchlebten Zustands-Aktionsfolgen zu speichern und so nicht nur einmal, sondern mehrfach zu benutzen, da sie für individuelle Zustände meist rar und daher mit Kosten verbunden sind. Insbesondere in Verbindung mit TD(λ) mit $\lambda > 0$ (also Traces) werden mit diesem Verfahren, vor allem für Aufgaben mit verzögerten Rewards, erhebliche Beschleunigungen erzielt [Wengerek, 1996]. Die gespeicherten 'Erfahrungen' (das Quadrupel s, a, r, s') werden, sofern die ausgeführten Aktionen noch immer der aktuellen Policy entsprechen, dem Agenten nochmals präsentiert.

Beim Erwerb der Verhalten in dieser Arbeit wird die Verwendung von Traces ($\lambda > 0$) untersucht, um einen RL-Algorithmus für alle Agenten festzulegen. Es soll prinzipiell online gelernt werden, wobei gute Erfahrungen mit der Boltzman-Exploration bestehen.

4.7 Hierarchien von Teilaufgaben

Die Komplexität der RL ensteht durch die Größe des zu durchsuchenden Zustands-Aktionsraumes $S \otimes A$. Diese Komplexität läßt sich demzufolge durch eine Verringerung der Zustände reduzieren, wie sie durch den Einsatz von Vektorquantisierern erreicht wird (siehe Abschnitt 4.9.1), oder auch durch eine Zerlegung des Gesamtproblems in mehrere Teilprobleme, die sequentiell abgearbeitet werden. Durch die in der Sequentialität implizit enthaltenen Zeitinformationen können mit diesem Ansatz darüberhinaus auch Probleme der unvollständigen Beobachtbarkeit des Gesamtproblems gelöst werden. Aus der Vielzahl der relevanten Algorithmen seien hier drei herausgegriffen.

Zur Lösung von POMDP stellen SCHMIDHUBER et al. [Wiering and Schmidhuber, 1996] eine hierarchische Erweiterung des *Q-Learnings* vor. HQ-Lernen basiert auf einer geordneten Sequenz von Subagenten. Jeder Subagent lernt eine Markov'sche Teilaufgabe der möglicherweise Nichtmarkov'schen Gesamtaufgabe zu identifizieren und zu lösen. Zur Identifikation der Teilaufgaben ist kein externer Teacher notwendig.

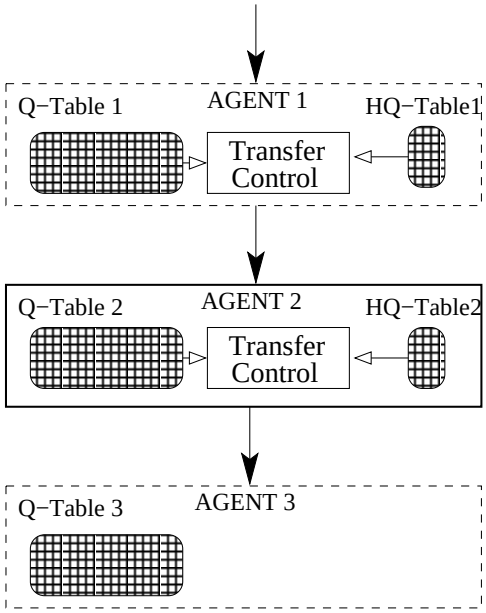

Abbildung 4.6: *Subagentenstruktur beim HQ-Lernen*

Die Grundarchitektur besteht aus einer vorzugebenden Anzahl sequentiell verbundener Subagenten. Jeder Agent verfügt über eine Q-Tabelle, eine HQ-Tabelle und eine Transfer Control Unit, abgesehen vom letzten Agenten, der nur eine Q-Tabelle besitzt. Die Q-Tabelle enthält dabei normale Q-Werte, während die HQ-Tabelle ähnliche Werte zur Schätzung des Wertes einer Observation als Teilziel für diesen Agenten enthält. Die Transfer Unit übergibt die Kontrolle über die Aktionen des Animats an den nächsten Agenten, wenn das Teilziel erreicht ist.

Die HQ-Tabelle nimmt die geschätzten Subgoal Values $HQ(o)$ auf. Für jede mögliche Observation o existiert ein HQ-Eintrag, und der Agent muß den Wert der Observation als Teilziel ermitteln. Die Anzahl und Abfolge der Subagenten ist festgelegt, jeder Subagent muß das von ihm zu erreichende Teilziel und die dazugehörige Policy lernen.

Im Gegensatz zum HQ-Lernen zerlegt bei *Hierarchical Learning* [Lin, 1993] der Designer die Gesamtaufgabe in günstige Teilaufgaben. Grundfähigkeiten ('elementary skills') müssen initial vortrainiert oder erlernt werden. Das Erlernen der Koordination mit Aktivierung und Abbruch der 'elementary skills' geschieht durch *Q-Learning* der Teilaufgaben.

Dietterich stellt *MAXQ* in [Dietterich, 2000] als einen Ansatz zum Erlernen einer hierarchischen, rekursiv optimalen Policy vor. Die Struktur der Hierarchie und sinnvolle Teilaufgaben werden durch den Designer vorgegeben. Die MAXQ Methode stellt einen theoretischen, verallgemeinernden Rahmen für die Dekomposition der Aufgaben sowie der Wertefunktion zur Verfügung.

Hierarchische Abhängigkeiten zwischen Verhalten stellen eine wichtige, aber eingeschränkte Art der Beziehung dar. Auch die Methoden zum Finden einer geeigneten Zerlegung sind erst in der Entwicklung begriffen. In dieser Arbeit soll eine Koordination betrachtet werden, die neben der hierarchischen auch andere Beziehungen zwischen den Verhalten zuläßt und adaptiv erwerben kann.

4.8 Makroaktionen

Die von den im letzten Abschnitt erwähnten Methoden gefundenen Policies können aus einer anderen Sichtweise auch als abstrakte Aktionen aufgefaßt werden: als feststehende Abfolgen von primitiven Aktionen, die zusammen eine Makroaktion ergeben. Weiterentwickelte Hierarchien von Makroaktionen werden von Parr in [Parr, 1998] und Hauskrecht in [Hauskrecht et al., 1998b] sowie McGovern in [McGovern et al., 1998, McGovern, 2002] betrachtet.

Solche Makroaktionen sind auch als 'Optionen' [Precup et al., 1998] oder 'Macros' [McGovern et al., 1997, Randlo, 1999] bekannt. Sie reduzieren die Lernaufgabe für den Agenten in zweifacher Hinsicht: zum einen werden Zustände, die Teil einer Makroaktion sind und für die die Policy folglich feststeht, aus dem Zustandsraum entfernt, was diesen verkleinert. Zum anderen wird eine Anpassung an veränderliche Umwelten erleichtert, indem wenige Makroaktionen, nicht aber die gesamte Policy neu gelernt werden müssen.

Dazu wurde eine Voraussetzung des bisher diskutierten RL aufgegeben: die konstante Dauer von Aktionen. Da Aktionssequenzen unterschiedlich lang sein können, mußte mit den Semi-MDP (SMDP) (Abb. 4.7) eine neue theoretische Grundlage für verallgemeinerte Wertfunktionen gefunden werden, die zeitlich abstrakte Aktionen berücksichtigt.

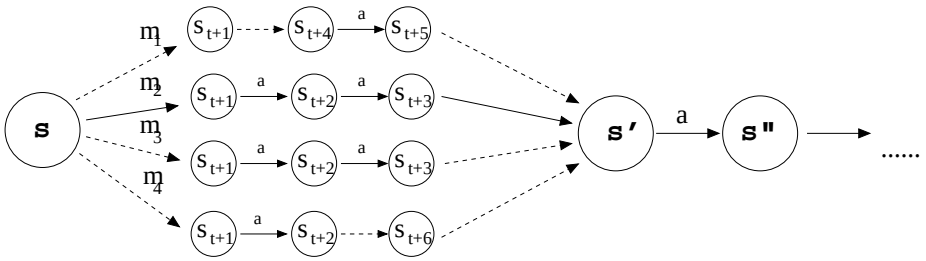

Abbildung 4.7: ***Prinzip eines Semi-MDP.*** *In einem Zustand s existieren mehrere Makroaktionen m, die über verschiedene Zustands-Aktionsfolgen in den Endzustand s' führen.*

Eine Makroaktion ist definiert durch eine eigene Policy, eine Abbruchbedingung und die Zustände, in denen sie verfügbar ist. Gl. 4.33 gibt die entsprechende Wertfunktion an.

$$V^{\pi}(s) = \overline{r}(s,m) + \sum_{s',N} P(s',N|s,m)\gamma^N V^{\pi}(s') \tag{4.33}$$

wobei N die Anzahl der Aktionen des Agenten bezeichnet, die bei der Ausführung von Makroaktion m im Zustand s bis zum Folgezustand s' ausgeführt werden. Der Zustandsübergang wird mit der Wahrscheinlichkeitsfunktion $P(s',N|s,m)$ beschrieben und $\overline{r}(s,m)$ stellt den erwarteten Reward für das Ausführen der Makroaktion m im Zustand s dar. Für den Fall einschrittiger Makroaktionen (d.h. $N = 1$) und der Annahme eines deterministischen Zustandsüberganges entspricht die Wertfunktion der des MDP.

Die Makroaktionen können bestehende konventionelle Regler oder feste Strategien sein, aber auch Verhalten, wie sie mit den Methoden des RL erlernt werden. Sie sind Hierarchien von Teilaufgaben ähnlich, da sie für bestimmte Teile des Zustandsraumes das Verhalten des Agenten festlegen. Allerdings bilden sie sich beim Lernen des Verhaltens heraus, während für Hierarchien zuerst die Teilaufgaben erzeugt und dann gelernt werden. Beide Varianten beschleunigen eine Verhaltensumstellung bei Änderungen der Umwelt, da nur relevante Teile des Verhaltens geändert werden müssen. Wie für Hierarchien gilt auch für Makroaktionen, daß in dieser Arbeit auch andere als hierarchische Beziehungen zwischen Teilverhalten betrachtet werden sollen und daher beide zunächst nicht eingesetzt werden.

4.9 Neuronale Umsetzung von RL-Agenten

Neuronale Netze bieten sich aufgrund ihrer Eignung zur universellen Funktionsapproximation und Generalisierung in hochdimensionalen, kontinuierlichen Räumen als Basis zur Umsetzung von RL-Agenten an. Sowohl überwachte Lernverfahren alleine als auch unüberwachte Lernverfahren kombiniert mit überwachten Lernverfahren werden zur Adaptation herangezogen, darüber hinaus aber auch lebenslang lernfähig gestaltet, um den Agenten veränderlichen Aufgaben und Umwelten anzupassen.

Sollen die Eingangsräume nicht mehr diskretisiert werden, sondern in ihrer natürlichen Kontinuität benutzt werden, wird die Anzahl der Zustände unendlich und eine Implementierung als Tabelle, mit Zustands-Aktionspaaren und zugehörigen Werten, entsprechend den ursprünglichen Algorithmenformulierungen unmöglich. Es ist daher naheliegend, zum Erlernen der Wertfunktion und ggf. der Policy Funktionsapproximatoren mit der nötigen Generalisierung einzusetzen, die wegen der unbekannten Charakteristik der Funktion eine möglichst uneingeschränkte Menge von Funktionen lernen können, d.h. neuronale Netze. Zur Umsetzung von RL-Verfahren mit neuronalen Systemen eignen sich prinzipiell alle Architekturen, die zum überwachten Lernen eingesetzt werden können. Der notwendige Fehler für die Ausgangsschicht entsteht z.B. mit $e = r + \gamma Q_k(s',a') - Q_k(s,a)$ aus der in Abschnitt 4.3.1 Temporalen Differenz. Prinzipiell lassen sich zwei Ansätze unterscheiden, um die notwendigen Funktionen zu lernen:

- Partitionierung des Eingangsraumes mit lokaler Funktionsapproximation
- direkte, globale Funktionsapproximation.

4.9.1 Partitionierung des Eingangsraumes mit lokaler Funktionsapproximation

Für eine lokale Funktionsapproximation können Algorithmen aus dem unüberwachten Lernen benutzt werden, um ähnliche Zustände zu Gruppen (Clustern) zusammenzufassen und für diese die Wertfunktion, ähnlich einer Tabellenimplementation, zu lernen. Unüberwachte Lernverfahren können Wertfunktionen prinzipiell nur in Kombinationen mit überwachten Lernverfahren erlernen. Sie selber bieten keine Möglichkeit, zur Approximation den o.g. Fehler zwischen vorhergesagtem und eingetretenem Wert auszuwerten. Oft verwendete Algorithmen zur Clusterung [Ripley, 1996] sind adaptive Vektorquantisierer, wie die *Self-Organizing Feature Map* (SOFM, [Kohonen, 1997]), das *Neural Gas* (NG, [Martinetz and Schulten, 1991]), das *Growing Neural Gas* (GNG, [Fritzke, 1995]) bzw. *Dynamische Zellstrukturen* (DCS, [Bruske, 1998]) oder in jüngster Zeit *Support Vector Machines* [Dietterich and Wang, 2002]. Damit ist die Anzahl der zu lernenden Cluster wesentlich geringer als die Anzahl der Vektoren im Eingangsraum (hier Zustände). Gleichzeitig ermöglichen diese Clusterer eine Generalisierung von erlebten auf noch nicht erfahrene Zustände.

SOFM und NG bilden mittels der Kooperation benachbarter Clusterzentren die Wahrscheinlichkeitsverteilung der Vektoren im Eingaberaum derart ab, daß beide aufgrund ihres Auflösungsfaktors als optimale Vektorquantisierer bezeichnet werden können. Bei dieser Abbildung führen aber die festgelegte Dimension und Nachbarschaft des SOFM-Gitters zu topologischen Defekten, wenn die den Daten innewohnende, *intrinsische* (topologische) Dimension nicht der festgelegten entspricht. Dieser Nachteil kann beim NG durch die fehlende Gitterstruktur vermieden werden. Die intrinsische Dimension wird im allgemeinen nicht mit der Anzahl der Eingangsmerkmale übereinstimmen, da diese selten unabhängig voneinander sind und besonders bei unvollständig beobachtbaren Systemen mehrdeutige Zuordnungen verursachen. Bei offenen Datenströmen, wie sie eine lebenslange Lernfähigkeit impliziert, kann sich die intrinsische Dimension mit der Zeit ändern und ist so nicht vorhersagbar.

Das aus dem NG entwickelte GNG, sowie die sehr ähnlichen DCS werden mit Hebb*'s Wettbewerbslernen* [Martinetz and Schulten, 1994] so adaptiert, daß sie die Topologie der Unterräume, aus denen die Eingabevektoren stammen, perfekt, bzw. bei endlichen Datenmengen optimal, abbilden. Damit zählen sie zu den *perfekt topologieerhaltenden Karten* [Bruske, 1998]. Diese Güte entsteht aus dem *inkrementellen*, fehlerbasierten Einfügen neuer Clusterzentren in den Graphen online während des Lernens, und aus der Definition zulässiger Kanten. Der verwendete Fehler kann sich aus der Vektorquantisierung, aber auch aus der folgenden Approximation, oder einem Mix beider ergeben.

Zum überwachten Lernen können bei diesen Netzen Radiale Basisfunktionen verwendet werden, so daß auch eine lokale Approximation mit Interpolation möglich wird, die bei ähnlichen Erweiterungen der SOFM (LVQ, [Huang et al., 1989]) oder des NG nicht möglich ist ([Bruske, 1998]). Als eine effektive Konfiguration der DCS erwies sich die Kombination von einem Aufbau der topologischen Karte mittels des Quantisierungsfehlers, mit einem Einfügen neuer Clusterzentren zwischen Zentren mit hohem Approximationsfehler, ergänzt durch Einfügen im Falle unzureichender Aktivierung des dem Eingansmuster ähnlichsten Zentrums [Bruske et al., 1997]. So wird mit Hilfe einer variablen, nützlichen Auflösung ein gleichmäßiger Approximationsfehler erzielt, aber auch die notwendige Ab-

deckung des Eingangsraums garantiert.
Eine ungünstige Clusterung ist gekennzeichnet durch Zustände, die radikal unterschiedliche Aktionen verlangen, rein sensorisch aber in dasselbe Cluster fallen und so eine Oszillation des Wertes bzw. der auszuführenden Aktion bewirken. Dies kann insbesondere dann eintreten, wenn der Raum starr gleichmäßig geclustert wird. Alle oben angeführten Algorithmen sind in der Lage, eine ungleichmäßige Clusterung (Auflösung) zu produzieren, die eine notwendige, aber noch nicht hinreichende Bedingung für eine günstige Clusterung sind. Sie setzt eine auf der Interaktion mit der Umwelt beruhende Adaptation der Cluster voraus und führt dazu, daß nur vom Agenten real angetroffene Zustände geclustert werden.
Natürlich entsteht dann das Problem wandernder Clusterzentren und sich damit ändernder Wertfunktionen, wo der Vorteil der genannten, inkrementellen Verfahren mit dynamischer Netzstruktur (GNG, DCS) ersichtlich wird. Sie sind nicht an eine begrenzte Zentrenanzahl und -struktur gebunden, sondern verfolgen das Ziel gleichmäßig kleiner Approximationsfehler. Ein von der Wahrscheinlichkeitsdichte der Inputs unabhängig ermittelter Fehler entschärft hierbei das Problem nichtstationärer Wahrscheinlichkeitsverteilungen und damit verbundener lokaler Minima durch die Betrachtung des globalen Fehlers. Bei echt überlagerten Clustern aber, wie sie bei teilweiser Beobachtbarkeit eines Systems auftreten, verursacht der entstehende Fehler eine anhaltende, jedoch verfehlte Einfügung neuer Zentren. In solchen Fällen kann der Eingangsraum um eine Repräsentation zurückliegender Aktionen erweitert werden, im einfachsten Fall z.B. den Vektor der zuletzt ausgeführten Aktion. Allerdings hat dies u.U. eine exponentielle Erhöhung der Anzahl der Clusterzentren zur Folge.
Diese Ansätze zur zumindest teilweisen Lösung des 'Stabilitäts/Plastizitäts-Dilemmas' und die Unabhängigkeit der Lernparameter von der Zeit bilden notwendige Voraussetzungen für den Einsatz dieser Netze in lebenslang lernfähigen Systemen.
Mit den die Clusterzentren repräsentierenden Neuronen wird auch ein intuitives Verständnis der Eligibility Traces (siehe Abschnitt 4.3.1.1) erleichtert, indem jedes Neuron ein Traceattribut erhält. Stellt das Neuron das dem aktuellen Input entsprechende Clusterzentrum dar, wird die Trace aktiv gesetzt, ansonsten klingt sie passiv ab.

Wird aus dem Eingangsstrom zunächst der Zustand ermittelt, und erst darauffolgend die entsprechenden Aktionswerte bestimmt, dann kann die folgende Aktualisierung des Zustands- oder Zustands-Aktionswertes gezielt und kontrolliert über die Nachbarschaftsbeziehungen des Clusterers auf ähnliche Zustände ausgedehnt werden [Gross et al., 1996]. Diese *SoftMax* genannte Ausweitung des Lernens von Zustands-Aktionswerten stützt sich auf die Annahme, daß topologisch ähnliche Zustände auch zu ähnlichen Aktionen in der optimalen Policy führen sollten. Im Interesse eines effizienten Einsatzes der Ressourcen eines solchen Clusterers können solche Knoten zusammengelegt und damit eingespart werden, die in ähnlichen Zuständen zu gleichen Aktionen führen. Mit Hilfe des in der folgenden Approximationsschicht ermittelten Fehlers können diese Knoten so platziert werden, daß der Q-Fehler minimiert wird. Dies hat eine Reduzierung des Zustandsraumes für den RL-Agenten zur Folge.

Weiterhin kann der Zustandsraum der Knoten durch den erlebten Reward erweitert und die Knoten selbst mit Traceattributen versehen werden, um Probleme der unvollständigen Beobachtbarkeit zu erleichtern.

4.9.2 Direkte, globale Funktionsapproximation

Die Realisierung von RL mit globalen Funktionsapproximatoren wurde schon im Zusammenhang mit residualen Algorithmen (siehe Abschnitt 4.3.3) sowie mit Policysuchverfahren (siehe Anschnitt 4.4) diskutiert.

Neben überwachten Methoden zur lokalen Funktionsapproximation, wie dem CMAC von ALBUS (siehe dazu [Houk et al., 1997]) und den Radialen Basisfunktionen (RBF) [Anderson, 1993], kommen auch sehr oft Funktionsapproximatoren mit globaler Aktivierungsfunktion zum Einsatz. So zeigte SUTTON in [Sutton, 1989], wie ein TD(λ)-Verfahren mit Multi-Layer Perzeptrons realisiert werden kann. Zu beachten ist dabei, daß die Eligibility Traces nicht für Zustände bzw. Zustands-Aktionspaare, sondern für jedes Gewicht im Netz berechnet werden müssen.
Es kann pro Aktion ein Netz benutzt werden, damit die für eine Aktion erforderlichen Änderungen der Gewichte nicht die Werte anderer Aktionen beeinflussen. Dies impliziert natürlich eine fehlende Generalisierungsfähgkeit hinsichtlich ähnlicher Aktionen und erfordert einen diskreten Aktionsraum.

4.10 Zusammenfassung

Dieses Kapitel konzentrierte sich auf eine breite Darstellung der verschiedenen Möglichkeiten zum Erlernen von Verhalten mit Reinforcement Lernen. Diese Methode wurde anderen, eingangs des Kapitels erwähnten, aus verschiedenen Gründen vorgezogen. Dort wurden neben genetischen Algorithmen auch Anticipative Classifier Systems genannt.
Derem Hauptaugenmerk von liegt auf einer Umweltmodellierung, nicht aber einer Verhaltensgenerierung. Existierende Algorithmen zum Lernen von Verhalten stützen sich auf dieselben Prinzipen wie RL. Genetische Algorithmen behandeln dieselben Probleme wie RL, sind ihr aber meist unterlegen, wenn Zustandsinformation vorhanden ist und ermöglichen kein Online Lernen während der Interaktion mit der Umwelt. Sind keine sensorischen Informationen über den Zustand vorhanden, sind GA dem RL völlig überlegen, da es ohne Zustandsinformation nicht auskommt, während GA aus einer zustandsübergreifenden Verhaltensbewertung adaptiert werden können.

Unter den in diesem Kapitel vorgestellten Grundlagen und gebräuchlichsten Algorithmen wurden besonders ausführlich die Ansätze mit diskontierten Zustandswerten betrachtet, da sie in der Literatur am weitesten verbreitet sind und mit ihnen auch positive Erfahrungen vorliegen. Grundlegende Ansätze wie Policy Iteration oder Value Iteration können wegen des benötigten Modells nicht verwendet werden, während für Average Reward Ansätze negative eigene Erfahrungen und auch keine Konvergenzgarantien vorliegen. Mit gradientenbasierten Verfahren schließlich liegen weder eigene Erfahrungen vor, noch sind aus der Literatur reale Anwendungen bekannt. Ein wichtiger Grund gegen ihren Einsatz ist auch, daß für solche Verfahren keine Ansätze zu ihrer adaptiven Koordination bekannt sind, und sie vermutlich (siehe Abschnitt 5.1.5) dazu auch noch grundlegende Forschung benötigen. Entsprechend sollte der Schwerpunkt der Untersuchungen auf *Q-Learning* und *Sarsa* liegen, zumal diese auch gradientenbasierte Erweiterungen zulassen.

Die wichtigsten Probleme des RL wurden erläutert und einige Ansätze zu ihrer Lösung. Dabei wurden hierarchische Zerlegungen der Gesamtaufgabe und das Zusammenfassen

von Aktionen zu Makroaktionen erläutert, sowie die Hierarchie als eingeschränkte Koordination verschiedener Verhalten.

Weiterhin wurden Möglichkeiten zur neuroinformatischen Umsetzung der RL-Algorithmen, insbesondere mit Clusterern und einer Approximationschicht, mit einigen Vorzügen und Problemen erläutert. Dieser Ansatz sollte den Ausgangspunkt für die Implementierung der Agenten in dieser Arbeit sein.

Kapitel 5

Adaptive Verhaltenskoordination

In Anlehnung an die Vorstellung des Reinforcement Lernens in Kapitel 4 werden in diesem Kapitel die Grundlagen von Lernen in Multiagentensystemen (MAS) eingeführt und der zentrale Algorithmus der Arbeit, das *W-Lernen*, vorgestellt. Aus einer überwältigenden Fülle von Ansätzen zu MAS werden dann einige wenige mit dem *W-Lernen* verglichen.

Lernen wird allgemein als notwendige Voraussetzung für Intelligenz angesehen. Weiss listet in [Weiß, 1999] eine Vielzahl von möglichen Arten und Ansatzpunkten für Lernen auf. Dieses Kapitel beschäftigt sich mit Lernen in MAS unter der Zielsetzung, koordinierte Aktionen zur Maximierung des Gesamtreturns für das Animat zu erreichen. Solche Arten von Lernen unterteilt Weiss in zentralisierte und dezentrale (vgl. 2.4.4). Der Schwerpunkt hier wird auf dezentralisiertem Lernen liegen, da die Agenten ihre Koordination selbst lernen sollen.

Nach den Recherchen des Autors sind die theoretischen Grundlagen für Lernen in MAS im Entstehen begriffen und ergeben noch kein geschlossenes Gebilde. Da es insbesondere zu MAS in einem einzigen Körper kaum Grundlagen gibt, kann der zentrale Algorithmus der Arbeit, das *W-Lernen*, auch nicht als eine Variante präsentiert werden wie *Sarsa* im vorigen Kapitel, die aus allgemeineren Prinzipien abgeleitet wurde. Die vorhandenen Grundlagen werden verbal umrissen, auf eine mathematische Darstellung aber verzichtet.

5.1 Grundlagen für MAS

Vermutlich aufgrund ihrer Übereinstimmung mit der grundlegenden menschlichen Problemlösungsstrategie, komplexe Probleme in einfachere Teilprobleme zu zerlegen und die gefundenen Teillösungen zu einer Gesamtlösung zu kombinieren, haben MAS eine lange Tradition auch in den Computerwissenschaften allgemein und in der Künstlichen Intelligenz im besonderen. Entsprechend gibt es hierzu formelle Betrachtungen, die allerdings einen ganz anderen Kontext annehmen als die Untersuchungen in dieser Arbeit, als aktuelles Beispiel sei auf [Ferber, 2001] verwiesen. Der dort behandelte Gegenstand der Kenetik geht von einem Entstehen der Intelligenz aus der Organisation vieler Individuen aus, ähnlich der Funktion des Hirns auf der Grundlage von Neuronen, während der Schwerpunkt der hier behandelten Methoden auf einer Koordination intelligenter Agenten liegt. Der Unterschied ähnelt dem zwischen Soziologie und Psychologie.

Die theoretischen Grundlagen für Lernen in MAS im Kontext des Reinforcement Lernen sind gerade im Entstehen begriffen und greifen auf Konzepte aus der Spieltheorie zurück. Der Beginn für eine umfassende Behandlung wird in [Bowling and Veloso, 2000] gelegt. Im Gegensatz dazu gibt es umfangreiche Literatur für nichtlernende MAS, die auf Grundlagen in der Ökonomie, Sozialwissenschaften, Multiagentenplanung der symbolischen KI etc. zurückgreift. Grundlegende Eigenschaften wurden bereits in Abschnitt 2.5 umrissen.

Handelt es sich bei den Agenten jeweils um lernfähige Systeme, die ihre Policy ändern, so steht der einzelne Agent einer nichtstationären Umwelt gegenüber, denn die Zustandsübergänge, die von den Aktionen der anderen Agenten verursacht werden, verändern sich, während diese lernen und ihre Policy ändern. Dies motiviert den Einsatz der Spieltheorie, die die Agenten als Spieler ansieht und entsprechende Konzepte entwickelt hat [Bowling and Veloso, 2000, Hu and Wellman, 1998].

Für jeden Markow-Prozeß (siehe Abschnitt 4.2.1) mit mehreren Agenten (Multiagent MDP) gibt es mindestens eine Kombination der Einzelpolicies, die einen maximalen Gesamtreturn ergeben. Eine solche Kombination ist in der Spieltheorie als *Nash*-Equilibrium bekannt. Ein solches Equilibrium ist als eine Kombination von Einzelpolicies definiert, bei der eine Änderung einer Einzelpolicy keine Verbesserung des Gesamtreturns ergibt.

Einzelpolicies, die gemeinsam die optimale Teampolicy ergeben, können auf verschiedene Art und Weise erzeugt werden [Hauskrecht et al., 1998a]

- **zufällige Auswahl mit Lernen:** Jeder Agent wählt im aktuellen Zustand unter seinen optimalen Aktionen über mehrere Versuche aus, bis die beste Kombination gefunden wurde. Die Zeit, diese zu finden, kann exponentiell von der Anzahl der Agenten abhängen.

- **Konventionen:** Eine unmittelbare Koordination ergibt sich aus einer vorbestimmten Ordnung der Agenten und Aktionen. Konventionen sind meist nicht flexibel bezüglich der Eingliederung neuer Agenten oder dem Auftreten neuer Aufgaben in der Umwelt.

- **Kommunikation:** Agenten tauschen sich z.B. über die von ihnen bevorzugte Aktion aus, bzw. stellen sich auf die Aktionen anderer Agenten ein. Kommunikation kostet Zeit und ist unsicher.

Die Anwendung der Spieltheorie im RL steht jedoch einigen Problemen gegenüber, die die Unsicherheiten der realen Umwelt, die Begrenzung der Wahrnehmung der Agenten und die Kenntnis der Rewardfunktionen anderer Agenten betreffen. Die Spieltheorie beschäftigte sich meist mit der Ermittlung eines Endwertes für ein Spiel und setzte dazu Annahmen voraus, die von den genannten Problemen verletzt werden.

5.1.1 Multiagent MDP

Die Basis für eine formelle Beschreibung von Multiagentensystemen ist wiederum das Konzept eines Markow'schen Entscheidungsprozesses (Markow Decision Process, *MDP*). Ein Multiagent MDP, *MMDP* oder *stochastisches Spiel* entsteht aus einer Umwelt, die als

ein MDP beschrieben wird, und mehreren Agenten, sowie aus einem von allen Agenten kombinierten Aktionsraum und einer Rewardfunktion, die für jeden Agenten unterschiedlich aussehen kann [Littman, 1994].

Die Zustandsbeschreibung für den MMDP setzt sich aus der für den MDP und weiteren Informationen zusammen. Welche weiteren Informationen notwendig sind, hängt von der Art des MMDP ab. Handelt es sich um ein Szenario mit mehreren Agenten, die unterschiedliche oder dasselbe gemeinsame Ziel verfolgen, so müssen die Informationen vollständig über den Zustand der Agenten Auskunft geben. Bei diesem Fall handelt es sich um ein Mehrkörpersystem, in dem die Koordinierung Teil des Algorithmus der einzelnen Agenten ist und nicht von einer zentralen Entscheidungsinstanz, einem Arbiter ausgeführt wird.

Entsprechend kann die Koordinierung als Teil des Aktionsspektrums der Agenten aufgefaßt werden. Sie können sich entscheiden, an einer Koordinierung teilzunehmen oder nicht [Boutilier, 1999]. Sollten sie teilnehmen, so kann die folgende Aktion ein Kommunikationsakt sein, um sich mit den anderen Agenten abzustimmen [Durfee et al., 1989]. Protokoll und Inhalt der Kommunikation können dabei sehr variabel sein und hängen vom konkret eingesetzten MAS ab. Insgesamt lassen sich folgende Ansätze unterscheiden:

- Die Agenten nehmen sich gegenseitig nicht wahr, ignorieren sich also.
- Die Agenten kommunizieren hinsichtlich ihrer Sensorik, der erfahrenen Zustandsbewertung, auszuführender Aktionen, auf den zeitlichen Ebenen des Schritts, der Episode oder ganzer Policies [Tan, 1993]. Sie können auch in einer Auktion um die Chance der Aktionsausführung bieten [Baum, 1998].
- Agenten können andere Agenten hinsichtlich deren Policies oder deren Wissens über andere Agenten modellieren.

In dem Fall, daß es sich um nur ein Einzelkörpersystem mit mehreren Agenten handelt, muß die Zustandsbeschreibung um Informationen über die Aktivierung der Teilziele oder homöostatischen Zustände des Animats erweitert werden, um einem MMDP zu entsprechen [Whitehead et al., 1993]. Dieser Fall muß vermutlich als Kooperation selbstinteressierter fixer Agenten, also eine Mischung aus nichtkooperativem und vollständig kooperativem MMDP, wie sie unten definiert sind, klassifiziert werden.

Im folgenden wird ein kurzer Überblick über die Theorie zu stochastischen Spielen gegeben. Sie bieten für nichtkooperative Spiele einige wenige lernende Algorithmen, während kooperative Spiele zwar gelöst werden können, jedoch nur durch nicht lernende Algorithmen. Für die in dieser Arbeit interessierende Mischung beider Probleme sind keine direkten Lösungen bekannt.

Stochastische Spiele [Hu and Wellman, 1998, Bowling and Veloso, 2000] erweitern sowohl das Konzept des MDP für einzelne Agenten auf mehrere Agenten, die die Zustandsübergänge und die Rewards beeinflussen, als auch das der Matrixspiele von einem auf mehrere Zustände. Dadurch bilden sie die Grundlage einer theoretischen Behandlung von lernenden MAS. Sie heben die Forderung nach einer stationären Umwelt im RL auf und lassen Umwelten zu, die nicht nur von anderen Agenten beeinflußt werden, sondern diese Agenten können auch lernen und ihre Verhalten ändern.

Ausgangspunkt der folgenden Darstellung sind vollständig kompetitive MMDP, da sie Startpunkt der theoretischen Analyse [Littman, 1994] waren und Lösungen existieren.

Besonders interessant sind jedoch die vollständig kooperativen MMDP, weil deren Zielsetzung und damit die Probleme sehr den hier untersuchten ähnlich sind. Der Unterschied besteht darin, daß in dieser Arbeit mehrere Agenten eine Aktion für ein gemeinsames Animat als bestmöglichen Kompromiß finden sollen, während bei kooperativen MMDP jeder Agent für sein Animat diejenige Aktion als Komponente einer Gesamtaktion des Teams finden soll, die den Gesamtreturn maximiert. Weiterhin erhält in einem kooperativen MMDP jeder Agent denselben Reward, während die Untersuchungen dieser Arbeit von einem unterschiedlichen Reward pro Agent ausgehen.

5.1.2 Vollständig kompetitive MMDP

Szenarien, in denen nur eine festgelegte Menge Reward zur Verfügung steht, die unter mehreren Agenten aufgeteilt werden soll, werden als *zero sum game*, als Nullsummenspiel, bezeichnet. Die Agenten haben jeweils genau entgegengesetzte Interessen, des einen Agenten Gewinn entspricht des anderen Agenten Verlust. Hierfür existiert genau ein Nash-Equilibrium (vgl. Abschnitt 5.1) und es kann mit linearer Programmierung berechnet [Bowling and Veloso, 2000] werden. Das heißt, in Multiagenten-Szenarien, in denen die Agenten in striktem Wettkampf miteinander stehen, können relativ einfach die optimalen Policies für die Agenten berechnet werden. Beispiele für entsprechende Algorithmen sind *Minimax-Q* [Littman, 1994] oder *Ficticious Play* (FP, [Bowling and Veloso, 2000]).

5.1.3 Nichtkooperative MMDP

In diesen Szenarien verfolgen die Agenten zwar noch ausschließlich eigene Interessen, ihr Gewinn bedeutet jedoch nicht mehr einen Verlust in gleicher Höhe für einen oder mehrere andere Agenten. Allerdings hängt der Reward für jeden Agenten immer noch vom Reward aller anderen Agenten ab. In der Spieltheorie wird ein solcher MMDP als *Genral sum game*, als Spiel mit beliebiger Endsumme bezeichnet, da die Gesamtperformanz beliebige Werte annehmen kann. Nullsummenspiele sind eine Teilmenge hiervon. Hu et al. entwickeln *Minimax-Q* [Littman, 1994] auf der Grundlage von Q-Lernen für nichtkooperative MMDP weiter und nennen den entstandenen Algorithmus *Nash-Q*. Er ist einer der ersten Algorithmen, die andere als kompetitive Spiele lernen können. Die Agenten verfügen nicht über vollständige Informationen bezüglich der Rewardfunktionen der anderen Agenten und deren Zustandsübergangswahrscheinlichkeiten, sonst wäre das Gleichungssystem mit nichtlinearer Programmierung lösbar. Ein Agent kann jedoch Aktionen und Rewards der anderen Agenten beobachten und lernt neben seiner eigenen Q-Funktion weitere für die anderen Agenten. Er kann daraus eine gemischte Nash-Strategie entwickeln, die seinen Return unter Berücksichtigung der anderen Agenten maximiert, egal wie diese sich verhalten. *Nash-Q* konvergiert beweisbar zu einem Nash-Equilibrium. Sollte es davon jedoch mehrere im Spiel geben, muß der Algorithmus um eine Schätzung der Aktionswahl der anderen Agenten erweitert werden. Die Schwierigkeit besteht darin, daß das Equilibrium nicht nur gefunden werden muß, sondern daß auch alle beteiligten Agenten dasselbe Equilibrium anstreben müssen, sonst ist die Lösung suboptimal und instabil.

5.1.4 Vollständig kooperative MMDP

In einem vollständig kooperativen MMDP stehen mehrere Animats, jedes mit einem Agenten, vor dem Problem, eine gemeinsame Aufgabe lösen zu müssen. Sie erhalten dabei alle denselben Reward und müssen jeweils diejenige Aktion finden, die den gemeinsamen Return in einer der Markow-Bedingung gehorchenden Umwelt maximiert. Dies bedeutet, dieses Team von Agenten agiert anstelle eines einzigen Agenten, der im weiteren Verlauf der Arbeit als Monolith bezeichnet werden soll. Auch diese Szenarien fallen unter *General sum games*. *Team-Q*[Littman, 2001] konvergiert und löst diesen Fall garantiert und ohne Modell der anderen Agenten, während *Coordinated Reinforcement Learning* (CRL [Guestrin et al., 2002]), *FP* und *Joint Action Learners* (JAL, [Claus and Boutilier, 1998]) zumindest einige kooperative MMDP lösen können. *Team-Q* konvergiert zwar garantiert, allerdings ist nicht gesichert, daß alle Agenten zu demselben Equilibrium konvergieren, was die Gesamtperformanz beliebig schlecht machen kann.

Boutilier et al. betrachten in [Claus and Boutilier, 1998] die Anwendung normaler Q-Lerner, die die Existenz der anderen Agenten ignorieren und stellen fest, daß sich die Leistungsfähigkeit eines Teams von Agenten, die unabhängig voneinander mit normaler Exploration trainiert wurden (independent learners), nicht signifikant von der eines Meta-Agenten (*JAL*) unterscheidet, der explizit die Werte der möglichen Aktionskombinationen der Agenten lernt. Letzterer kann jedoch mit den von den Autoren vorgeschlagenen Explorationsstrategien sein besseres Potential zur Geltung bringen und gezielt ein optimales Equilibrium erreichen.

JAL und *FP* stützen sich auf eine Modellierung der Aktionswahl der anderen Agenten, da das Spiel vom Verhalten aller beteiligten Agenten abhängt und die Aktionswahl sich ihrerseits während des Lernens ändern kann. Sie sichern so die Wahl des gleichen Equilibriums. Dies veranlaßt viele Autoren, für jeden Agenten ein vollständiges Modell der anderen Agenten zu verlangen, das deren Erkenntnisstand über andere Agenten einschließt und so unendlich rekursiv wird [Gmytrasiewicz and Durfee, 2000, Hu and Wellman, 2001]. Hingegen benutzt *CRL* eine Kommunikation zwischen den Agenten über ihre lokalen Policies zur Abstimmung.

Minimax-Q, *Nash-Q* und *Team-Q* haben ohne Modelle der anderen Agenten keine Möglichkeit, alle Agenten zu einem einzigen Equilibrium konvergieren zu lassen.

Die optimale Policy π^* des Monolithen wird in kooperativen MMDP durch eine Kombination der Einzelpolicies der Agenten ersetzt. Nimmt man an, daß sich der Aktionsraum des Agententeams aus dem Produkt der Aktionen der Einzelagenten ergibt, so ergibt sich die Policy des Teams auch aus dem Produkt der Einzelpolicies, eine sogenannte *faktorisierte* Policy. Diese Teampolicy kann dabei nur einen Teil derjenigen Policies erzeugen, die der Monolith erzeugen könnte [Peshkin, 2001]. Dies sind genau diejenigen Policies, die keine zustandsabhängige Koordination erfordern. Ergibt sich also die Teampolicy im aktuellen Zustand aus den unabhängigen Einzelpolicies der Agenten ohne explizite Koordination, dann umfaßt sie nur einen Teil der dem Monolithen zugänglichen Policies.

Zur Verdeutlichung des eingeschränkten Policyraumes von Teampolicies sei ein Beispiel aus [Peshkin, 2001] zitiert. Das Problem sei die Zusammenstellung eines Mahlzeitenplanes. Ein Agent kümmere sich um die flüssigen Bestandteile $a_1 \in \{\text{Wodka,Milch}\}$ und der andere um die festen $a_2 \in \{\text{Pickles,Müsli}\}$. Ein Beispiel für eine optimale Policy wäre, in

90% der Fälle **a** ={Milch, Müsli} und in 10% der Fälle **a** ={Wodka, Pickles} zu wählen. Dies aber erfordert eine Koordination zwischen beiden Agenten, um die beiden anderen, offensichtlich suboptimalen Kombinationen auszuschließen. Eine faktorisierte Policy kann das nicht ohne Koordination leisten, ein Monolith schon.

Daraus ergibt sich auch zwingend, daß eine Teampolicy meist suboptimal im Vergleich zur Policy eines Monolithen ist. MATARIC formuliert entsprechend in [Mataric, 1995], daß individuelle Agenten ihre Optimalität zugunsten kollektiver Effizienz aufgeben müßten. Ein Kollektiv von Agenten kann durch Kooperation dem Gesamtsystem emergente Fähigkeiten verleihen, die über die Fähigkeiten der Einzelagenten bzw. deren einfache Kombination hinausgehen. Es ist anzunehmen, daß sich eine Superadditivität der Leistungen von Agenten nur dann ergibt, wenn sich die Gesamtaufgabe nicht additiv, sondern multiplikativ aus den Teilaufgaben ergibt, wenn also alle Teilaufgaben erfüllt sein *müssen*, um die Gesamtaufgabe zu erfüllen, wie bei einer seriellen Folge von Teilaufgaben.

5.1.5 Alternative Ansätze

Die in Abschnitt 4.7 vorgestellten Hierarchien von Teilaufgaben sind einer der am häufigsten gewählten Zugänge zu Multiagentensystemen. Die dort angeführten Ansätze, durch hierarchische Zerlegung einen POMDP in eine Serie von MDP zu überführen, können als Unterstützung der Vermutung bezüglich der Superadditivität gewertet werden: die geordnete Ausführung aller Teilaufgaben löst ein größeres, nur schlecht lösbares Problem.

Hierarchien verwandt sind die Makroaktionen, die in Abschnitt 4.8 eingeführt wurden. SUTTON et al. benutzen in [Sutton et al., 1999] die in dem Abschnitt vorgestellte Theorie der Semi-Markowprozesse zur Koordination mehrerer feststehender Agenten. *Termination Improvement* unterbricht die Ausführung der Policy eines Agenten, die hier als Makroaktion aufgefaßt wird, immer dann, wenn ein Zustand erreicht ist, von dem aus eine andere Policy bzw. Makroaktion einen besseren Return verspricht. Es ist allerdings leicht vorzustellen, daß ein solches Vorgehen zu permanentem Umschalten zwischen zwei Makroaktionen führt und nicht zu einer Erhöhung des Gesamtreturns beiträgt.

Es konnte gezeigt werden, daß Nash-Equilibria lokale Optima im Rewardgradientengebirge sind, und daß Gradientenaufstieg (siehe Abschnitt 4.4) in diesem Gebirge solche Optima findet [Meuleau et al., 2001, Peshkin et al., 2000, Pendrith and McGarity, 1997]. Andererseits sind nicht alle lokalen Minima, die der Gradientenaufstieg findet, auch Nash-Equilibria [Meuleau et al., 2001]. Das heißt, die Lösungen, die in einem vollständig kooperativen MMDP mit Agenten gefunden werden können, die einen Algorithmus des Gradientenaufstiegs einsetzen, sind nicht garantiert optimal, also nicht Lösungen des Kooperationsproblems. Es kann aber auch nicht garantiert werden, das solche Lösungen lokal optimal oder stabil sind. Das wiederum heißt, daß der Gradientenaufstieg zwar zu einer Lösung konvergieren kann, diese aber bereits durch kleine Störungen, wie Rauschen beim Lernen, verlassen wird und das sogar zu Verbesserungen führen kann. Aber die gefundenen Lösungen können auch die gesuchten optimalen, stabilen Koordinationen repräsentieren.

5.2 Vorstellung von W-Lernen

Aus den vorangegangenen Ausführungen geht hervor, daß es für das Problem mehrerer Experten, die sowohl in Wettkampf um die Ausführung einer Aktion durch die nur einmal vorhandenen Effektoren stehen, aber diese Aktion auch in Kooperation zur Erzielung eines maximalen Gesamtreturns für das gesamte Animat auswählen müssen, in der noch sehr jungen Theorie der lernenden MAS kaum geeignete Ansätze gibt.

W-Lernen (WL, [Humphrys, 1997]) benutzt die durch RL erlernten Zustands-Aktionswerte zum Training bzw. zur Ermittlung von W-Werten für jeden Zustand, den das Animat erlebt. Ein W-Wert enthält das Gewicht, mit dem jeder Experte seine Aktion in dem Zustand vorschlägt. Jeder Experte schlägt für jeden Zustand eine Aktion vor (z.B. die beste), jedoch sind bestimmte Zustände für ihn wichtiger als andere, und der 'richtige' Gewinner oder *'Leader'* für einen Zustand muß gefunden werden. Einer der Expertenagenten wird durch Vergleich dieser W-Werte ausgewählt, dessen Aktion das Animat dann ausführt (siehe Abb. 5.1). Die Varianten zur Ermittlung der W-Werte, die im folgenden vorgestellt werden, dienen zur Maximierung des Gesamtreturns für das Animat. Während des entsprechenden Lernprozesses ist es möglich, daß ein bestimmter Zustand nacheinander von unterschiedlichen Experten gewonnen wird, bis der beste gefunden ist. Die Experten haben dabei keine Kenntnis über den Gesamtreturn und sind auch hinsichtlich ihres Zustandsraumes und ihres Aktionsraumes völlig selbstständig, jeder Experte kann seinen eigenen haben. Sie haben auch keine Kenntnis darüber, an einem Multiagentensystem teilzunehmen, und kommunizieren daher nicht oder modellieren einander nicht.

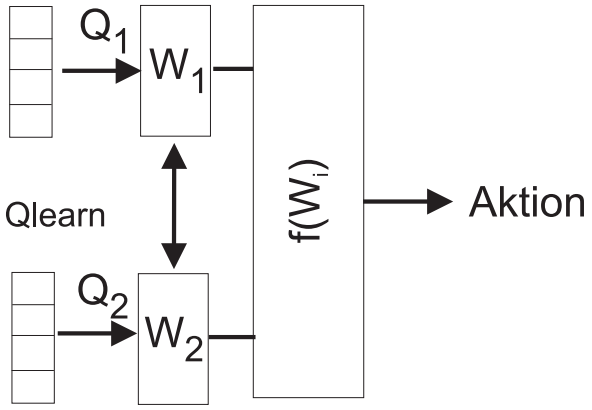

Abbildung 5.1: ***Prinzipdarstellung des W-Lernen.*** *Mehrere Expertenagenten benutzen in einer gegebenen Situation ihre Q-Werte, um auf verschiedene Arten W-Werte zu berechnen, die ihrerseits zur Auswahl eines Agenten für die aktuelle Aktionsausführung dienen.*

Durch die Adaptivität des Lernens wird die Notwendigkeit umgangen, alle Situationen vorauszusehen und entsprechende Agentenprioritäten festzulegen. Durch die Selbstorganisation der Experten wird der Brook'sche Ansatz erweitert: die Agenten müssen nicht mehr in einer starren Hierarchie organisiert sein, sondern ermitteln jeweils den zutreffendsten unter gleichberechtigten Experten. Dabei können sie sich unter anderem auch hierarchisch organisieren. Durch die einfache und dynamische Berechnung der W-Werte ist das Gesamtsystem auch leicht erweiterbar. Fallen Experten aus, sind immer andere vorhanden, die an der Ausführung einer Aktion interessiert sind.

Ein Schritt im W-Lernen läuft bei allen Varianten wie folgt ab:

1. Ermittlung agentenspezifischer Zustände aus der Situationsbeschreibung
2. Ermittlung der vorgeschlagenen Aktionen und deren Q-Werte
3. Bildung situationsspezifischer W-Werte (Berechnung aus Q-Werten)
4. Bestimmung des Agenten mit maximalem W-Wert (des Leaders)
5. der Leader legt die aktuelle Aktion fest
6. der Leader lernt nicht, die W-Werte aller anderen Agenten werden angepaßt (vgl. Gl. 5.2)
7. Ausführung der Aktion
8. Verteilung der agentenspezifischen Rewards

Die Anzahl der vorhandenen W-Werte hängt von der Anzahl der Zustände und der Agenten ab: jeder Agent i bildet in jedem seiner eigenen Zustände einen W-Wert für jeden der anderen Agenten. Dieser beinhaltet den Verlust, den ein Agent j als Leader dem Agenten i zugefügt hat. Dabei wird möglicherweise über mehrere Zustände von Agent j gemittelt.
Tabelle 5.1 verdeutlicht das anhand eines Beispieles mit 3 Agenten mit je 2, 4 und 3 Zuständen. Die Anzahl der Aktionen ist unerheblich, da jeder Agent genau eine vorschlägt.

Agent	Anzahl Zustände (Bsp)	resultierende Anzahl W-Werte
i	2	$2 \cdot (4+3) = 14$
j	4	$4 \cdot (2+3) = 20$
k	3	$3 \cdot (2+4) = 18$
gesamt	9	52

Tabelle 5.1: *Beispiel für die Anzahl von W-Werten mit drei Agenten.*

5.2.1 Bedeutung der W-Werte

Jeder Experte kann in jedem Zustand einen W-Wert ermitteln, jedoch ist nicht jeder Zustand für jeden Experten von gleicher Bedeutung. Manche Zustände sind wichtiger für den Experten als andere. Dies soll mit den W-Werten ermittelt werden. Insofern entsprechen die W-Werte der Stärke der Bedürfnisse (vgl. HULL's *Drive* [Klein, 1991]). Sie müssen jedoch nicht per Hand codiert werden, sondern können aus den Q-Werten ermittelt werden, die aussagen, wie wichtig ein Zustand für einen Agenten ist. Die Q-Werte legen für einen Agenten als Experten damit nicht nur fest, was getan werden soll, sondern auch, wie wichtig das ist.

Die W-Werte der verschiedenen Experten können mit unterschiedlichen Methoden aus den Q-Werten berechnet oder gelernt werden. HUMPHRYS unterscheidet zunächst *statische* von *dynamischen* W-Werten.
Die **statischen Werte** $W_i(s)$ werden dabei aus den Q-Werten jedes Experten ermittelt:

1. **Wichtigkeit:**
   $W_i(s) = \max_{a \in A} Q_i(s,a) - \min_{a \in A} Q_i(s,a)$
   Die Wichtigkeit kann als Differenz zwischen dem größten und dem kleinsten Q-Wert des Agenten i im Zustand s ermittelt werden. Sie wird klein, wenn alle Q-Werte sehr ähnlich sind, dem Experten die auszuführende Aktion also gleichgültig ist.

2. **Relevanz:**
   $W_i(s) = \sqrt{\frac{1}{|A|} \sum_{a \in A} (Q_i(s,a) - \overline{Q_i(s)})^2}$
   Die Relevanz einer Aktion kann aus der Abweichung ihres Q-Wertes vom mittleren Q-Wert ermittelt werden. Je größer die Abweichung, umso wichtiger die Aktion. Dafür wird über die Anzahl der verfügbaren Aktionen $|A|$ der mittlere Q-Wert $\overline{Q_i(s)}$ berechnet.

3. **Boltzman Normierung:**
   $W_i(s) = \frac{\exp \frac{Q_i(s,a_i)}{T}}{\sum_a \exp \frac{Q_i(s,a)}{T}}$
   Die absoluten Beträge der Q-Werte verschiedener Agenten können sehr unterschiedlich sein, deshalb wäre eine Möglichkeit, den Q-Wert der vorgeschlagenen Aktion auf $[0 \ldots 1]$ zu skalieren und die Agenten so vergleichbar zu machen.

In der Bestimmung der W-Werte nach ihrer Wichtigkeit oder Relevanz sagt der W-Wert aus, wie wichtig die Aktion dem Agenten im Vergleich zu den anderen verfügbaren Aktionen ist. Bei der Boltzman-Normierung werden alle Unterschiede durch Skalierung von der absoluten Größe der Q-Werte unabhängig gemacht. Allerdings gehen damit Unterschiede zwischen den Agenten verloren, die ihnen durch die Rewards, die in den Q-Werten enthalten sind, vom Designer verliehen wurden. Die statischen W-Werte eignen sich wegen ihrer Ermittlung ohne Betrachtung der Interaktion mit anderen Experten, also der Folgen von deren Aktionsausführung, schlecht für eine Koordination.

Die **dynamischen Werte** betrachten dagegen Interaktionen mit anderen Agenten und sind daher für eine Koordination interessanter. In der zentralen Variante, dem *W-Learning*, lernt ein Experte die Folgen der Aktionsausführung eines anderen Experten. Er beobachtet, in welchen Folgezustand die Aktion des anderen Experten ihn brachte, und ermittelt den ihm dadurch entstandenen Verlust gegenüber der Ausführung seiner eigenen vorgeschlagenen Aktion a_{sugg}. Der Verlust für einen Agenten i ergibt sich aus der Differenz des erwarteten Returns D_i zum erlebten Return F_i (siehe Gl. 5.1).

$$W_i \leftarrow D_i - F_i \tag{5.1}$$

$$W_i(s) \leftarrow Q_i(s, a_{sugg}) - (r_i + \gamma * \max_{a' \in A} Q_i(s', a')) \tag{5.2}$$

Bei Q-Lernern als Agenten ergibt sich der Verlust (Gl. 5.2) aus der Differenz zwischen dem erwarteten Return $Q_i(s, a_{sugg})$ und dem erfahrenen Return. Dieser ergibt sich als Summe aus dem erfahrenen Reward r_i und dem Wert des Folgezustandes s' für den Experten i,

also dem Q-Wert der besten Aktion für i. Das Lernen erfolgt hier für die Verlierer des Wettkampfes, da beim Lernen der W-Werte für die Gewinner oder *Leader* deren W-Werte grundlos kleiner würden und diese unmotiviert ihre Führung aufgeben würden. Ließe man die Leader ihre W-Werte dagegen alternativ erhöhen, würden ihre W-Werte schrankenlos wachsen und kein anderer Agent die Chance erhalten, Leader zu werden.
WL wählt beim Wettkampf denjenigen Experten aus, der den anderen Experten den kleinsten Verlust verursacht. Deshalb nennt HUMPHRYS sie *'Minimize Worst Unhappiness' (mWU)* mit gelernten W-Werten.

Haben die Experten jedoch alle den gleichen Aktionsraum, so brauchen die W-Werte nicht gelernt werden, da bereits Q-Werte mit den gesuchten Informationen vorhanden sind (Gl. 5.3). Diese Methode wird 'Negotiated WL' (neg) genannt. Der W-Wert für Agent i wird aus der Differenz zwischen dem Q-Wert seiner vorgeschlagenen Aktion a_{sugg} und *seinem* Q-Wert $Q_i(s, a_k)$ für die vom Agenten k vorgeschlagene Aktion a_k ermittelt.

$$W_i(s) = Q_i(s, a_{sugg}) - Q_i(s, a_k) \tag{5.3}$$

Diese Methode weist folgende Vorteile auf:

- alle notwendigen Werte sind schon bekannt (Q-Werte sind bereits gelernt)
- kein Lernen der W-Werte mehr erforderlich
- leichtere Auswertung der W-Werte
- immer beliebige Agenten(-zahlen) ohne Lernen koordinierbar

Die alternative, 'kollektive' Methode ermöglicht Synergien zwischen leicht suboptimalen, aber andere Agenten unterstützenden anderen Aktionen zu ermöglichen, indem alle Agenten einbezogen werden(siehe Gl.5.4). Sie verhindern, daß ein Experte mit sehr großen Q-Werten und entsprechenden Differenzen den Wettbewerb dominiert, indem er sich gegen die vereinten anderen Agenten durchsetzt. Allerdings kann das auch zu risikoscheuem Verhalten führen, da die Mehrheit der Agenten einen einzelnen großen Reward für einen Experten verhindert. Wurden dem Experten vom Designer gezielt große Rewards zugeordnet, wirken die kollektiven Methoden seiner beabsichtigten Dominanz entgegen.

$$W_i(s) = \sum_{k=1}^{N} (Q_k(s, a_{sugg}) - Q_k(s, a_i)) \tag{5.4}$$

Der W-Wert für den Agenten i ergibt sich hier als Summe der Verluste aller N Agenten, die sie durch die Aktion von Agent i erleiden.

In den bisher vorgestellten Varianten wurde davon ausgegangen, daß die Agenten ihre W-Werte aus den Q-Werten ihrer vorgeschlagenen Aktionen $Q_i(s, a_{sugg})$ ermittelten, die sie dazu vorschlugen. Eine absolute Aussage aber erhält man erst bei Betrachtung aller Q-Werte des betreffenden Agenten. Diese Methode wird *'pures' Minimize Worst Unhappiness* genannt (siehe Gl. 5.5).

$$W_i(s) = \max_{a \in A} (Q_i(s, a) - Q_i(s, a_k)) \tag{5.5}$$

Hierbei ermittelt der Agent seinen W-Wert als den maximalen Verlust über alle seine Aktionen, die eine Ausführung der Aktion a_k ihm verursachen würde.

Sowohl kollektive als auch pure Methoden ermöglichen es, eine Aktion auszuwählen, die keiner der Experten vorgeschlagen hat, die aber der Gesamtheit aller Experten bzw. dem Animat zugute kommt.

5.2.2 Auswertung der W-Werte

Die Auswertung der W-Werte durch ihren Vergleich erfolgt in einer Matrix, die für jeden Zustand spezifisch erstellt wird und Auskunft darüber gibt, wieviel Verlust jeder Agent jedem anderen Agenten zufügt. Die Aktion des Agenten k, der den maximalen Verlust erleidet, wird ausgeführt (Gl. 5.6).

$$W_k(s) = \max_{i \in 1 \ldots N} W_i(s) \tag{5.6}$$

Agent k, dessen vorgeschlagene Aktion a_k den größten Wert W_k erzielt, wird als 'Leader' bezeichnet und a_k ausgeführt. In der folgenden Matrix bezeichnet der erste Index den Verursacher und der zweite den Empfänger des Verlustes, W_{21} heißt also, das die Ausführung der Aktion, die Agent 2 vorschlägt, dem Agenten 1 den Verlust W_{21} zufügt. Agent 2 verursacht dem Agenten 1 einen Verlust von 3.

$$\begin{bmatrix} 0 & W_{21} & \ldots & W_{N1} \\ W_{12} & 0 & \ldots & W_{N2} \\ \ldots & \ldots & \ldots & \ldots \\ W_{1N} & \ldots & W_{N,N-1} & 0 \end{bmatrix} \quad \text{z.B.} \begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 9 \\ -6 & 0 & 0 \end{bmatrix}$$

Rechts ist ein Beispiel angegeben, das [Humphrys, 1997, S. 56] entnommen wurde und im folgenden exemplarisch ausgewertet wird. Zur Auswertung wird ein zufälliger Prozeß benutzt anstelle der Auswahl des Maximums über alle W-Werte. HUMPHRYS argumentiert, dies diene zur Beschleunigung des Lernens. Der Prozeß terminiert garantiert nach maximal $N+1$ Prüfungen bei N Experten, was die Reaktivität des Animats sichert. Die Matrix wird folgendermassen durchsucht[1]:

1. fange in zufälliger Spalte j an
   z.B. (Spalte 2)
2. suche Zeile m mit $W_{m,j} = \max_i(W_{i,j})$
   (Zeile 1 : 3)
3. speichere W^{max}, m =Zeile(W^{max})
   ($W^{max} = 3$, $m = 1$)
4. gehe zu Spalte $j' = m$
   (Spalte 1)

[1] $m = i^{max}$ bezeichnet den Zeilenindex des Agenten, dem maximaler Verlust zugefügt wird. Die Werte in Klammern beziehen sich auf das Beispiel rechts oben.

5. wenn $W^{max} < \max_i(W_{i,j'})$, weiter bei 3.
   (nein, da $3 > 0$).

6. bis alle $W_{i,j'} < W^{max}$
   (Aktion des Agent 1 wird ausgeführt)

Das Ergebnis hängt unter Umständen vom zufälligen Startpunkt ab, es müssen jedoch nicht alle Verluste auch wirklich erfahren werden. Die Verlierer müssen ihre W-Werte erhöhen, um beim nächsten Besuch desselben Zustandes selber Leader zu werden. Unter Verwendung der 'puren' Varianten kann auch eine globale Auswertung stattfinden, die aber nur sinnvoll ist, wenn alle W-Werte auch öfter erfahren werden, was den Trainingsaufwand sehr erhöht.

Die Tabelle 5.2 führt die wichtigsten Varianten von WL in ihrer 'vorgeschlagenen' und ihrer 'kollektiven' Formulierung auf. Zum einfacheren Verständnis haben alle Agenten den gleichen Aktionsraum, also müssen keine W-Werte gelernt werden. Die Auswertung der Matrix ist mit enthalten (Ermittlung der Minima bzw. Maxima im ersten Term).

Min. Worst Unhappiness	Minimiere größten Verlust
vorgeschlagen	$\min_{a_k \in A} \max_{i \in 1 \ldots N} (Q_i(s, a_{sugg}) - Q_i(s, a_k))$
kollektiv	$\min_{a_k \in A} \sum_{i=1}^{N} (Q_i(s, a_{sugg}) - Q_i(s, a_k))$
Maximize Best Happiness	Maximiere größten Gewinn
vorgeschlagen	$\max_{a_k \in A} \max_{i \in 1 \ldots N} Q_i(s, a_k)$
kollektiv	$\max_{a_k \in A} \sum_{i=1}^{N} Q_i(s, a_k)$
Maximize Best Profit	Maximiere größten Profit
vorgeschlagen	$\max_{a_k \in A} \left[Q_k(s, a_k) - \sum_{i, i \neq k} (Q_i(s, a_{sugg}) - Q_i(s, a_k)) \right]$
kollektiv	$\max_{a_k \in A} \left[\sum_i Q_i(s, a_k) - \sum_i (Q_i(s, a_{sugg}) - Q_i(s, a_k)) \right]$

Tabelle 5.2: *Varianten von WL in Formulierungen mit vorgeschlagenen Aktionen oder kollektiv über alle Agenten. Die Minimums-/Maximumsauswahl unter allen Aktionen legt die auszuführende Aktion fest. 'Worst Unhappines' wurde im Text erläutert.*

Verbal ausgedrückt realisiert 'Worst Unhappiness' eine Suche nach dem geringsten verursachten Verlust, 'Best Happiness' nach dem größten individuellen Gewinn und 'Best Profit' nach größtem Gewinn und kleinstem Verlust. Sie ist dabei wegen der Summe der Verluste als eine Variante zwischen vorgeschlagenen und kollektiven Methoden anzusehen.

Intuitiv scheint 'Best Happiness' die naheliegendste Variante, jedoch verfolgt sie nur den Gesamtreturn für genau einen Agenten auf Kosten der anderen. Meistens und besonders im Falle eines Gesamtreturns aus der Summe aller Teilreturns, schadet dies dem Animat insgesamt. Mit 'Worst Unhappiness' dagegen kann ein Agent i einen anderen Agenten j überzeugen, einen kleinen Verlust hinzunehmen, um katastrophale Verluste für i zu vermeiden. Dies ermöglicht dem Animat, opportunistisch Möglichkeiten für die Experten wahrzunehmen und den Gesamtreturn zu maximieren.

5.2.3 Wertung von WL

Mit W-Lernen steht eine Methode der Koordination zur Verfügung, die nach Bedarf Kooperation oder Konkurrenz realisieren kann, wobei sich die teilnehmenden Agenten selber

organisieren. Dieses Lernen erfordert dabei kaum Aufwand, da der Großteil der benötigten Werte aus den bereits austrainierten Q-Lernern verfügbar ist. Die Anzahl der Agenten ist dabei leicht und auch dynamisch änderbar. Hierarchien und Opportunismus (siehe Abschnitt 2.4) können realisiert werden und Agenten können Verluste für andere Agenten vermeiden, aber auch deren höhere Kompetenz zu eigenen Zwecken benutzen.

Das Gesamtverhalten des Animats hängt einerseits von den im Training verteilten Rewards und den daraus resultierenden Q-Werten der Agenten bzw. deren Relation unter den Agenten ab, und andererseits von der gewählten Variante zur Berechnung der W-Werte. Diese Varianten sollten es ermöglichen, jeweils unterschiedliche Aspekte der Koordinierung hervorzuheben.

Vor dem Hintergrund einer Anwendung von W-Lernen in realen Szenarien erscheint das individuelle Vortraining der Agenten und die Vernachlässigung des Gesamtrewards zur Bewertung der Effizienz der Koordination wenig effektiv, sodaß zumindest Ansätze eines gleichzeitigen Trainings von Agenten und ihrer Koordination betrachtet werden sollten.

5.3 Alternative Multiagentenansätze

Humphrys diskutiert schon in [Humphrys, 1997] einige verwandte Ansätze mit ähnlicher Zielsetzung, von denen einige hier nur kurz erwähnt werden sollen.

Lin [Lin, 1993] koordiniert eine Anzahl Q-Lerner durch einen übergeordneten Q-Lerner, der darauf trainiert wird, welcher Agent in welcher Situation einzusetzen ist. Ein solcher Lerner steht jedoch den Problemen des RL gleich zweifach gegenüber und muß für jeden neuen oder wegfallenden Agenten neu trainiert werden.

In Maes' *Spreading Activation Network* werden die Funktionen der einzelnen Experten mit ihrer Koordination vermischt, was zu Problemen führt, zwischen der Aktivierung eines Ziels durch mehrere Bedürfnisse und einer Aktivierung durch mehrere Wege von einem Bedürfnis zu unterscheiden.

Digney's *Nested Q-Learning* [Digney, 1996, Digney, 1998] erweitert *Q-Learning* dahingehend, daß nützliche Teilaufgaben automatisch gebildet und abgespalten werden, wobei eine Teilaufgabe nach zwei Kriterien gebildet wird: nach hohem Rewardgradienten und häufigem Auftreten. Solche Teilaufgaben können von mehreren anderen Aufgaben benutzt werden, die unter Umständen selber dafür sorgen müssen, daß diese Teilaufgaben tatsächlich aktiviert werden. Allerdings haben eigene Tests ergeben, daß sich beim *Nested QL* häufig nicht auflösbare Schleifen wiederkehrender Aufrufe zwischen Teilaufgaben ergeben und nützliche Teilaufgaben einfacher als mit Digney durch eine geeignete Zustandsclusterung unter Beachtung des erfahrenen Rewards realisiert werden können.

5.3.1 Psi (Ψ)

Mit Ψ (*Psi*) [Dörner, 1999, Dörner, 2001] stellen Dörner et al. ein System vor, das in der Lage ist, sowohl einzelne, zielgerichtete Verhalten zur Befriedigung verschiedener Bedürfnisse zu erlernen, als auch diese verschiedenen Verhalten geeignet zu planen und zu koordinieren. Dabei interagiert das Animat mit simulierten, relativ komplexen

Umwelten und zeigt psychologisch relevante Phänomene, ähnlich den vorne erwähnten *BDI*-Systemen (Belief-Desire-Intention, siehe Abschnitt 2.5), sodaß sich zwei Fragen aufdrängen: Warum haben seine Entwickler Ψ noch nicht auf realen Robotern implementiert, und warum haben Robotersteurer noch nicht auf Ψ oder zumindest Teilaspekte und Ideen daraus zurückgegriffen?

Die erste der beiden Fragen läßt sich zunächst pragmatisch einfach beantworten: Ψ's Entwickler sind als Psychologen zuallererst an einer Erklärung menschlichen Handelns interessiert und hierbei mit einer konkreten Realisierung als in sich geschlossene Architektur eines Animats weit über eine rein mathematische Formulierung hinaus gegangen. Damit befinden sie sich vermutlich an vorderer Front im Grenzgebiet zwischen Psychologie und Computational Neuroscience von psychologischer Seite aus. Die Beantwortung der zweiten Frage jedoch dürfte angesichts der von Ψ erbrachten Leistungen schwerer fallen. Zwar setzen die Entwickler in ihrer Simulation einfachste sensorische Vorverarbeitung, keinerlei Rauschen und eine der Markow-Bedingung genügende Umwelt voraus, die simpel genug ist, um ein nahezu vollständiges Protokoll aller Interaktionen mitzuführen. Jedoch sollte es nicht prinzipiell unmöglich sein, für einen Einsatz in der realen Umwelt aus dem Arsenal des Maschinenlernen und der Neuroinformatik geeignete Vorverarbeitungen zu finden, um die für die generelle Funktion von Ψ notwendigen Voraussetzungen zu schaffen. Sollte dies jedoch tatsächlich in den Details nicht realisierbar sein, so können immer noch Kernaspekte des Modells in realen Robotern eingesetzt werden.

Daher soll der prinzipielle AlgorithmusΨ im Überblick vorgestellt werden. Implementationsdetails sind in [Dörner et al., 1997, Dörner, 2001] zu finden.

5.3.1.1 Bedürfnisse

Die zu befriedigenden Bedürfnisse werden Ψ vom Designer vorgegeben und gliedern sich in zwei Kategorien: die *existentiellen* Bedürfnisse Hunger, Durst und Schmerzvermeidung, die wegen ständigen Energie- und Wasserverbrauchs Ψ's Selbsterhaltung dienen, sowie die Einsammlung eines Nukleotid genannten Rohstoffes, und die *kognitiven* Bedürfnisse der Steigerung von Bestimmtheit, Kompetenz und Affiliation.

Während die Affiliation, das Bedürfnis nach Interaktion mit anderen Individuen, in den Bereich der Soziologie führt und hier nicht weiter betrachtet werden soll, stellen die anderen beiden in ihrer expliziten Formulierung als Bedürfnisse zumindest eine in der Robotik bislang vernachlässigte Konsequenz, wenn nicht Neuheit dar. Zwar werden Kompetenz und Bestimmtheit in vielen Ansätzen berechnet und verwendet, jedoch nach Einsicht des Autors nicht explizit als Bedürfnisse dargestellt. Ihre explizite und getrennte Verwendung in Ψ ergibt sich aus dem besonderen Interesse von Ψ's Entwicklern an den sich aus ihnen ergebenden psychologischen Einsichten. Angesichts der resultierenden Flexibilität der Verhaltenssteuerung sollte jedoch der Einsatz solcher kognitiven Bedürfnisse auch im Maschinenlernen betrachtet werden.

Die existentiellen Bedürfnisse des Animats werden in Übereinstimmung mit einfachen Modellen über homöostatische Regelung als Abweichung eines Ist-Wertes von einem Soll-Wert angesehen. Eine Abweichung wird als ein Motiv angesehen, wenn sie eine Schwelle überschreitet. Diese Schwelle ist für eine Interaktion zwischen den unterschiedlichen Motiven und zur Vermeidung einer Überlastung des Animats variabel gestaltet.

5.3.1.2 Verhaltenserwerb

Führt eine Aktion zu einer Verminderung einer Abweichung, so wird das gelernt und diese Kausalkette später zur Bedürfnisbefriedigung herangezogen: Ψ wird bei Auftreten eines Motivs sein Gedächtnis, d.h. alle bisher erlebten Situationen und die dabei ausgeführten Aktionen, durchsuchen, um eine zu finden, in der das Motiv verringert wurde. Findet es eine, versucht es, die Situation herbeizuführen und die Aktion erneut auszuführen. Treten gleichzeitig mehrere Motive auf, so wird zunächst dasjenige bearbeitet, bei dem das Produkt aus Bedürfnis und erwarteter Befriedigung, also aus Bedürfnis und Kompetenz, am größten ist. Dies stellt das Grundprinzip von Ψ als Multiagentensystem dar und realisiert offensichtlich eine Priorisierung der verschiedenen Motive, indem dem stärksten Motiv, das auch realisiert werden kann, der Vorrang eingeräumt wird. Auch die notwendige Persistenz wird realisiert. Dabei wird eine zur Befriedigung des aktiven Bedürfnisses ermittelte Handlungskette trotz eventuell widersprüchlicher Nebeneffekte auf die anderen Bedürfnisse ausgeführt. Implementiert ist dies durch eine aktive Unterdrückung der anderen Motive durch das aktive Motiv. Da die Unterdrückung auch betraglich endlich ist, können bisher inaktive Motive Vorrang gewinnen und sowohl Opportunismus als auch notwendige Unterbrechungen ebenfalls umgesetzt werden. Nach Abarbeitung der Alternativpläne kann allerdings auch der ursprüngliche Plan weitergeführt werden, da das ursprünglich aktive Motiv noch nicht befriedigt ist und nun wieder aktiviert wird.

Der Wettkampf zwischen den Motiven wird jedoch auch von der Dringlichkeit beeinflußt; einem Zeitrahmen, der zur Befriedigung eines Motivs einzuhalten ist. Den Wettkampf gewinnt das Motiv mit dem größten Produkt aus Bedürfnis, Kompetenz und Dringlichkeit. Die Dringlichkeit und die so realisierte explizite Einbeziehung der Zeit stellen nach Ansicht des Autors auch im Maschinenlernen eine bisher selten erreichte Fähigkeit dar.

5.3.1.3 Verhaltensmodi

Zur Realisierung eines Verhaltens, also zur Verringerung eines Bedürfnisses, verfügt Ψ über drei Verhaltensmodi: Agieren, Planen und Exploration. Findet es, wie oben erwähnt, in seinem Gedächtnis eine bereits früher ausgeführte Handlungskette zur Befriedigung des aktiven Bedürfnisses, wird es diese ausführen. Sollte eine solche nicht existieren, plant Ψ eine Handlungskette: aus bekannten Teilketten wird eine erfolgversprechende Handlungskette zusammengestellt. Dazu formt Ψ sowohl aus sensorischen als auch aus motorischen Primitiven jeweils sogenannte Schemata, die Hierarchien beliebiger Tiefe erlauben. Die Generierung von Makroaktionen, die eine ähnliche Funktion erfüllen, ist aktueller Gegenstand der Forschung im Reinforcement Lernen und geeignet, sowohl POMDP als auch Semi-Markow-Prozesse (siehe Abschnitt 4.8) zu behandeln.

Sollte keine erfolgversprechende Kette komponierbar sein, beschafft sich Ψ durch Exploration das fehlende Wissen. Die Entscheidung zwischen Agieren, Planen und Exploration wird durch Bestimmtheit und Kompetenz moduliert.

5.3.1.4 Wertung

Ψ realisiert klassische und operande Konditionierung bis hin zur Planung mit variabler Breiten- und Tiefensuche, mit selektiver Aufmerksamkeit und sensorischen wie motorischen, gelernten Hierarchien und erfüllt die notwendigen Anforderungen an ein Multiagentensystem. Allerdings verfügt es nicht über unabhängige Agenten oder Experten im Sinne eines Multiagentensystems, sondern müßte im Sinne des RL als monolithischer Agent mit internem Modell und Makroaktionen eingestuft werden.

Ψ ist also ein Animat, welches innere Zustände hat, die zeitabhängig sind. Damit wird die Handlungsauswahl durch vorangegangene Aktionen beeinflußt, was die in der realen Welt auftretenden Probleme wie POMDP lösen oder erleichtern kann. Bei der Auswahl seiner Handlungen und seiner Planung stellt Ψ keinerlei Optimalitätsbetrachtungen an, die andererseits jedoch Zentralpunkt der Algorithmen des Reinforcement Lernen sind. Deren Einsatz in der Realität jedoch ist insbesondere durch POMDP eingeschränkt. Hier drängt sich die Frage nach einer gegenseitigen Ergänzung von RL und Ψ auf. Mit seinen Bedürfnissen würde Ψ das RL auch wieder zurück an seine ursprünglichen Wurzeln führen, der Modellierung von Konditionierung, die letztendlich der Bedürfnisbefriedigung dient.

5.3.2 Democratic Integration

Triesch et al. stellen in [Triesch and von der Malsburg, 2001] die Methode der *Democratic Integration* vor. Bei diesem Ansatz liegt zwar die Zielsetzung in der Fusionierung verschiedener sensorischer Modalitäten und nicht der Koordinierung verschiedener Agenten, jedoch werden auch hier mehrere Experten eingesetzt, um zu einem Gesamtergebnis zu kommen, an dem sich wiederum alle Experten orientieren.

In einem Szenario, bei dem unter veränderlichen Umweltbedingungen sich bewegende Gesichter verfolgt werden sollen, werden aus einem Kamerabild verschiedene Merkmale extrahiert. Jedes dieser Merkmale ist zu einer Verfolgung von Gesichtern geeignet, versagt aber unter bestimmten Umweltbedingungen. Daher wird jedem Merkmal eine Verläßlichkeit $v_i(t)$ zugeordnet. Jedes Merkmal berechnet eine Aktivitätskarte $A_i(\mathbf{x}, t)$, in der es die normierte Stärke seiner Reaktion auf das aktuelle Eingangsmuster $\mathbf{x}$ und damit seine Schätzung über die aktuelle Position des Gesichts verzeichnet. Diese Aktivitätskarten stellen die gemeinsame Sprache des MAS dar. Sie werden gewichtet überlagert zu einer Resultatskarte $K(\mathbf{x}, t)$.

$$K(\mathbf{x}, t) = \sum_{i=1}^{N} v_i(t) A(\mathbf{x}, t) \tag{5.7}$$

Aus dieser Resultatskarte wird die neue geschätzte Position des Gesichtes als Maximum der gewichtete Summe aller Merkmale ermittelt.

Im Anschluß wird für jedes Merkmal eine Qualität berechnet, die angibt, wie sehr die von dem Merkmal geschätzte Position von der von allen Merkmalen ermittelten abweicht. Für die Berechnung dieser Qualität werden verschiedene Maße vorgeschlagen.

Die Qualität eines Merkmals entscheidet mittels Verknüpfung über eine Leaky Integrator Gleichung über die Verläßlichkeit des Merkmals im nächsten Zyklus.

Interpretiert in der Terminologie des W-Lernens, können die Verläßlichkeiten $v_i(t)$ als Wichtungen angesehen werden, wie sie sich auch für die am W-Lernen beteiligten Agenten aus deren Q-Werten ergeben. Demzufolge verfolgt *Democratic Integration* die Methode des *Maximize Collective Happiness* (siehe Abschnitt 5.2.1) zur Ermittlung einer resultierenden Position. Die Experten schlagen nicht jeder eine geschätzte Position vor, sondern übermitteln der zentralen Entscheidungsinstanz (Arbiter) ihr gesamtes, übersetztes Schätzergebnis, wie es bei den Agenten im W-Lernen der Vektor aller Q-Werte in einem Zustand wäre. Der Arbiter kann ein Resultat ermitteln, das von keinem der Experten vorgeschlagen worden wäre.

Über das Konzept der Wichtung der Agenten im W-Lernen hinaus werden bei *Democratic Integration* die Wichtungen der einzelnen Experten jedoch auch anhand des Resultats adaptiert. Humphrys ermittelt die Wichtungen der Agenten in aufwendigen Evolutionsläufen anhand des erzielten Gesamtrewards. Diese Wichtungen sind fix für die Agenten in allen Zuständen, was der eigentlichen Zielsetzung von *Democratic Integration* zuwiderlaufen würde. Die Idee einer adaptiven und auch zustandsabhängigen Wichtung der Agenten eines modifizierten W-Lernen wird in Abschnitt 6.3 aufgegriffen.

5.3.3 Dynamische Policyverschmelzung

Whitehead et al. stellen in [Whitehead et al., 1993] ein dem W-Lernen ähnliches Herangehen an die Koordination mehrerer Experten vor. In der Simulation einer Gridwelt steht das Animat vor der Aufgabe, mehrere Ziele anzufahren. Diese Ziele nehmen binäre Zustände ein, sie können aktiviert sein oder nicht. Ihre Aktivierung unterliegt einer Zufallsfunktion nach der Bernoulliverteilung[2]. Ist ein Ziel jedoch einmal aktiviert, bleibt es aktiv, bis das Animat es konsumiert hat. Die Ziele stellen also alle Erreichungsziele dar, die auch gleiche kognitive Komplexität aufweisen. Durch diese Aktivierung ist die Umwelt dynamisch veränderlich und damit die Aufgabe für das Animat komplexer als in einer stationären Umwelt. Andererseits sollten die einzelnen Agenten sehr ähnliche Policies enthalten, die je nach zu erreichendem Ziel im Zustandsraum verschobene und skalierte Kopien einer einzigen Policy sein sollten. Diese Policy sollte ein Q-Gebirge mit einem einzelnen Gipfel darstellen, dessen Position je nach Position des Ziels verschoben ist, und dessen Höhe sich aus dem dem Ziel zugeordneten Reward ergibt.

Analog zum Herangehen in dieser Arbeit wird der Reward als Vektor aufgefaßt, der je einen Reward pro Ziel enthält. Der Gesamtreward ergibt sich aus der Summe der Einzelrewards.

Mit der Repräsentation der verwendeten Gridwelt, erweitert um die aktiven Ziele, erhält das Animat ausreichend Information, um die Aufgaben für die einzelnen Agenten, aber auch die Koordination als einen MDP zu lösen.

Bei den Agenten handelt es sich um Q-Agenten, die alle parallel und gleichzeitig zu ihrer Koordination trainiert werden. Ähnlich dem W-Lernen existiert mit dem Arbiter eine Entscheidungsinstanz, die unterschiedliche Koordinierungsstrategien verfolgen kann. Vor der Erläuterung der verwendeten Koordinierungsarten wird kurz der Ablauf des gesamten Algorithmus dargestellt:

[2]Ist das Ziel inaktiv, hat es eine festgelegte Wahrscheinlichkeit $P(z)$, aktiviert zu werden. Ergibt ein Zufallsgenerator eine Zahl $0 \leq x \leq P(z)$, wird das Ziel aktiviert, sonst nicht.

1. Wahrnehmung des Umweltzustandes und der Zielaktivierungen durch den Arbiter
2. Verteilung des Umweltzustands (hier identisch für alle Agenten) und des Rewards (unterschiedlich für jeden Agenten) durch den Arbiter
3. Berechnungen je eines Aktionsvorschlages durch jeden Agenten
4. Koordination und Berechnung der auszuführenden Aktion durch den Arbiter
5. Ausführung der Aktion

WHITEHEAD et al. untersuchen zwei Arten von Koordinierungsstrategien, die sich im wesentlichen in ihrem Planungshorizont unterscheiden und im folgenden erläutert werden.

5.3.3.1 Simple Koordinierung

Simple Koordinierungsstrategien betrachten nur die nächste auszuführende Aktion und setzen daher kein Umweltmodell voraus. Sie benötigen zur Berechnung dieser Aktion maximal eine Zeit, die polynomial von der Anzahl der aktiven Ziele abhängt. WHITEHEAD et al. untersuchen zwei Strategien dieser Art. Bei der ersten, die sie *Nearest Neighbor* nennen, setzt sich der Agent mit dem höchsten Q-Wert im aktuellen Zustand durch. Die Entsprechung dieser Strategie im W-Lernen ist direkt *Maximize Best Happiness* (Tabelle 5.2, 5.Zeile), das das erfolgversprechendste Ziel verfolgt.

Da alle Ziele so festgelegt wurden, daß sie den gleichen Reward verteilen, wird sich das Animat auf das nächstgelegene Ziel zubewegen, und diese Policy weiterverfolgen, bis dieses Ziel erreicht ist. Das Ziel wird deaktiviert und das Animat verfolgt das nächste Ziel und fährt so alle nacheinander an. Dies zeigt eine klare Priorisierung der Einzelziele und Persistenz bei deren Erreichung. Dies sollte auch gelten, wenn die Ziele nicht mehr gleiche Zielrewards erzielen. Dann hängt es von der Höhe der unterschiedlichen Zielrewards und dem jeweils benutzten Discountfaktor γ ab, wie weit ein bestimmtes Ziel andere 'überstrahlt' und das Animat anzieht. Sollten für alle Ziele gleiche γ's benutzt werden, so entscheiden die zugeordneten Zielrewards über die Priorisierung der Ziele. Allerdings ist die vom Animat erzielte Gesamtpolicy suboptimal, da es die Ziele einzeln betrachtet.

Diese Eigenschaft wird mit der zweiten simplen Strategie verbessert, die *Greatest Mass* genannt wird. Hier werden die Q-Werte aller Agenten für alle verfügbaren Aktionen pro Aktion summiert und die Aktion mit der größten Summe ausgeführt. In der Terminologie des W-Lernens entspricht dies genau dem *Maximize Best Happiness* mit kollektiver Methode. Diese Strategie versetzt das Animat in die Lage, ein einzelnes, nahes Ziel zugunsten einer dichten, aber weiter entfernten Ansammlung mehrerer Ziele zu ignorieren. Damit kommt es der optimalen Strategie für die Gesamtaufgabe näher. Der Ort der Entscheidung zwischen einer entfernten Gruppe von Zielen und einem nahen Einzelziel, und damit die Priorisierung, sollte von der Höhe der Einzelrewards, und den entsprechenden γ's abhängen. Diese Entscheidung ist bei dieser Strategie binär, sodaß das Animat die Eigenschaft des Opportunismus nicht tatsächlich aufweist. Man könnte sie aber als emergent betrachten, sollte sich das Animat für das Einzelziel entscheiden und dann die Gruppe anstreben. Sie hängt damit jedoch von der aktuellen Situation ab und kann daher nicht als Eigenschaft der Strategie angesehen werden. *Nearest Neighbor* erzielt als verbessertes Verfahren erwartungsgemäß eine bessere Performanz als *Greatest Mass*.

5.3.3.2 Lookahead Suche

Fortgeschrittene Koordinierungsstrategien auf der Grundlage einer *Suche* setzen dagegen ein Modell der Zustandsübergänge und der damit verbundenen Rewards voraus. Aus den Q-Werten der einzelnen Agenten versucht der Arbiter damit, ganze Aktionssequenzen zu erstellen und so der Wertfunktion für die Gesamtaufgabe näher zu kommen als die einfachen Strategien. Dieses Vorgehen behandelt also den Nachteil des MAS gegenüber einem Monolithen, keine Sequenzen von Aktionen und deren Wert zu betrachten. Da es unmöglich ist, alle vorstellbaren Sequenzen zu testen, wird angenommen, daß die Suche wie folgt eingeschränkt wird: Für alle verfügbaren Aktionen im aktuellen Zustand wird der Wert der Sequenz ermittelt, der sich ergibt, wenn diese Aktion ausgeführt würde, und danach einer der simplen Strategien oben gefolgt würde. Diese Strategien haben keine Entsprechung im W-Lernen, da sie ein Modell voraussetzen.

Sie erzielen höhere Gesamtreturns als jede der simplen Strategien, da sie per Suche mit dem Modell zumindest einige der möglichen Sequenzen anhand der Gesamtaufgabe evaluieren und ansonsten genauso gut sind wie die simplen Strategien. Die Strategien weisen Priorisierung und Persistenz auf, benötigen zur Erzielung besserer Performanz jedoch ein Modell.

Auch WHITEHEAD et al. verweisen jedoch darauf, daß die Verwendung eines Modells nur in weitgehend deterministischen Umwelten sinnvoll und der Erwerb des Modells mit hohen Kosten und großem zeitlichen Aufwand verbunden ist. Dieser könnte jedoch durch die verbesserte Performanz wettgemacht werden, wenn die Lebenszeit des Animats lang genug und das Lernen des Modells schnell genug sind, die Änderungen der Umwelt zu verfolgen.

5.4 Zusammenfassung

In diesem Kapitel wurden kurz die Grundlagen zur dezentral lernenden Koordination von Multiagentensystemen vorgestellt. Ausgehend von der Definition eines Multiagenten-MDP aus einer Umgebung, die als MDP beschrieben werden kann, sowie mehreren Agenten und den Aktivierungen der verschiedenen Ziele, wurden stochastische Spiele als sich entwickelnde theoretische Grundlage für eine lernende Koordination dargestellt. Sie unterscheidet prinzipiell drei Arten von Koordination hinsichtlich der Aufteilung des Rewards unter den Agenten, kompetitive, nichtkooperative und kooperative. Algorithmen, die unter verschiedenen Randbedingungen diese Probleme lösen können, wurden vorgestellt.

Im Kontext der im vorigen Kapitel diskutierten Methoden des Reinforcement Lernens wurde W-Lernen als zentrales Untersuchungsobjekt dieser Arbeit eingehend vorgestellt. In diesem Zusammenhang wurden einige alternative Koordinierungsansätze aus der Literatur diskutiert. W-Lernen stützt sich auf Agenten, deren Verhalten mit dem sehr leistungsfähigen Reinforcement Lernen erlernt werden kann. Es nutzt Informationen, die diese Agenten schon haben, verursacht den Agenten also keinen weiteren Lernaufwand, da auch die Koordination ohne Eingriffe oder Änderungen für die Agenten selbst funktioniert. W-Lernen ermöglicht es, für jeden Zustand adaptiv über die Auswahl des zu aktivierenden Agenten zu entscheiden. Dabei kann es bis auf parallele Aktionen die notwendigen

Aktionskombinationen (siehe Abschnitt 2.4) realisieren und darauf aufbauend, die meisten Anforderungen an einen Koordinationsmechanismus erfüllen. Alternative Ansätze können nur einen Teil der Koordinierungsarten erzeugen, die dem W-Lernen zugänglich sind, wie Hierarchien oder 'Maximize Best Happiness'. W-Lernen ist einer der sehr wenigen bekannten Ansätze zur lernenden Koordination mehrerer Experten in einem Animat, die zugunsten des Gesamtreturns kooperieren sollen, dabei aber um den Zugriff auf die Effektoren konkurrieren. Unter diesen erscheint er als der leistungsfähigste.

Aufschlußreich wäre ein Vergleich mit Weiterentwicklungen von Ψ hin zu unabhängigeren Experten für die einzelnen Bedürfnisse, oder Weiterentwicklungen von W-Lernen mit zeitlich veränderlichen Bedürfnissen. Als prinzipiell sehr leistungsfähig aufgrund seiner Flexibilität erscheint auch das Prinzip der Blackboards (siehe Abschnitt 2.4.3), das bisher noch nicht im Kontext des Reinforcement Lernens eingesetzt wurde.

Damit sind die Voraussetzungen geschaffen, um im nächsten Kapitel die in der Arbeit zu untersuchenden Fragestellungen hinsichtlich der Agenten, des Monolithen, der Koordination und des Szenarios darzulegen.

Kapitel 6

Festlegungen zu Aufgaben und Experimenten

In diesem Kapitel werden Überlegungen über notwendige Untersuchungen angestellt und Aufgaben festgelegt, auf deren Grundlage die Experimente durchgeführt wurden, deren Ergebnisse in den folgenden Kapiteln dargestellt werden.

Dies betrifft Festlegungen zur Gestaltung und Umsetzung des Szenarios, sowie zu den einzusetzenden Algorithmen für die Agenten. Auch ein Plan für die Durchführung der Experimente zur Prüfung von W-Lernen in einer realen Umgebung ist Teil dieses Kapitels. Es werden für Szenario, Agenten und Koordination grundlegende Experimente definiert und für Szenario und W-lernen einige Änderungen und Erweiterungen vorgesehen.

6.1 Szenario

Das Szenario dient vor allem dazu, die Leistungsfähigkeit der Koordination zu testen. Daher soll weniger Wert auf qualitativ unterschiedliche Aufgaben gelegt werden, die verschiedene kognitive Fähigkeiten der Agenten bewiesen hätten, sondern vor allem auf Aufgaben, in denen das Animat bzw. sein Koordinationsalgorithmus widerstreitendes Verhalten erfordernde Aufgaben miteinander in Einklang bringen muß.

Da von Aufgaben bzw. Objekten (vgl. Abschnitt 3.4) mit positivem Reward grundsätzlich gleiche Anforderungen an den Agenten gestellt werden, jedoch mit anderen Schwerpunkten, als von Objekten mit negativem Reward, sollen beide Arten im Szenario enthalten sein. Für Objekte mit positivem Reward, die eine Erreichungsaufgabe verkörpern, ist ein Verhalten zu erwarten, das aus einer eindeutigen Festlegung jeweils bester Aktionen in den verschiedenen Zuständen entsteht. Es ist ein Ziel gegeben, das der Agent erreichen soll. Daher gibt es in jedem Zustand mindestens eine Aktion, die den Agenten diesem Ziel auf schnellstem Wege näher bringt. Dies ist der Grundansatz der meisten Algorithmen des Reinforcement Lernens.

Für Objekte mit negativem Reward, die eine Vermeidungsaufgabe verkörpern, sind andere Verhalten optimal, für sie muß in jedem Zustand mindestens eine Aktion vermieden werden, die den Agenten dem Objekt näher bringen würde. Unter den anderen Aktionen

gibt es zwar mindestens eine, die den Agenten am besten von dem Objekt wegführt, jedoch hat der Agent meist wenig Veranlassung, diese anderen Aktionen vorzuziehen, die ebenfalls, wenn auch knapper, das Objekt vermeiden.

6.1.1 Höhere Anforderungen an das Animat

Die in Abschnitt 3.3 diskutierten unterschiedlichen kognitiven Anforderungen, die statische und dynamische Objekte stellen, werden zur Gestaltung des Szenarios nun detaillierter betrachtet.

Das Ziel, genau eine Aktion zu vermeiden, gilt jedoch nur für räumlich statische negative Objekte, also solche, die ortsfest sind. Während der oben erläuterte Grundsatz der besten zielführenden Aktion bei belohnenden beweglichen Objekten gültig bleibt, ändert sich bei bestrafenden beweglichen Objekten das zu erreichende optimale Verhalten des Agenten deutlich. Einige der Aktionen, die bei statischen Objekten ungefähr gleich bewertet wurden, also nicht auf das bestrafende Objekt führten, werden nun im nächsten oder einem der folgenden Schritte bestraft, weil das dynamische Objekt das Animat weiter verfolgt und erreicht. Abhängig von der Dynamik des Objektes ist zu erwarten, daß sich die Auswahl der günstigsten Aktion bis auf wenige oder eine einzige einschränken kann.

Ein Jäger, der das Animat aktiv verfolgt, stellt es vor qualitativ neue Anforderungen. Um zu verhindern, in Sackgassen gefangen zu werden, sollte es rudimentäre Ansätze von Planung, also der Vorhersage der Folgen eigener Aktionen und dem Abwägen von deren Bewertung vornehmen können, ohne sich dabei auf eine gleichbleibende Umgebung zu verlassen, wie es beim Ausweichen vor einem statischen 'Gefahrengebiet' der Fall war.

Wird der Jäger reflexartig oder reaktiv gesteuert, hätte das Animat auch die Möglichkeit, die Aktionen des Jägers vorherzusehen und für eine eigene Aktionsplanung auszunutzen.

Darüber hinaus wäre es vorstellbar, daß das Animat seine eigene Vorhersagbarkeit als Schwachpunkt identifiziert und in bestimmten Situationen eine zufällige Policy als am erfolgversprechendsten findet, die Bewegungen auf den Jäger zu vermeidet und ansonsten nicht vorhersagbar ist.

Dem Animat sollen im zu entwerfenden Szenario aber weder die globalen Informationen noch die kognitiven Mechanismen für solche Leistungen verliehen werden, es soll reaktiv bzw. implizit planend handeln. Im Interesse einer grundlegenden Untersuchung der Koordination mit W-Lernen ist eine solche Beschränkung zunächst angebracht. So soll das Animat prinzipiell dem Jäger ausweichen können, wie es gelernt hat, das Gefahrengebiet zu vermeiden. Der Jäger soll mit einfachsten Verfolgungsstrategien ausgerüstet sein, um Leistungsunterschiede zwischen den unterschiedlich trainierten Agenten in der Auseinandersetzung mit dem Jäger zu beobachten. Auch veränderte Koordinationsleistungen werden so in Verhalten und Performanzmaßen besser deutlich.

In dem zu entwerfenden Experimentalszenario sollen daher sowohl belohnende und bestrafende, als auch statische und dynamische Objekte eingesetzt werden. Da eine Abwägung verschiedener Quellen derselben Art von Reward zunächst nicht im Interesse der Arbeit liegt, soll es in jedem Szenario nur eine Quelle einer Art von Reward geben. Entsprechend soll nur entweder die statische oder die dynamische Quelle einer Rewardart eingesetzt werden.

6.1.2 Umsetzung des Szenarios

Die Zielsetzung, die Leistungsfähigkeit einer bestimmten Art von Koordination von Expertenagenten auf einem realen System zu untersuchen, setzt natürlich auch die Durchführung der Untersuchungen auf einem solchen realen System voraus. Da es sich, soweit bekannt, um erste Untersuchungen dieser Art mit realen Robotern handelt, ist es angebracht, zunächst die besser zu kontrollierende Umwelt einer Laborumgebung der vollen Realität, wie dem natürlichen Einsatzfeld eines Roboters, voranzustellen.

Für die Experimente sind kontrollierte und somit wiederholbare Randbedingungen hinsichtlich der Aufgaben, sensorischen Verhältnisse und Einsetzpunkte für den Roboter festzulegen, die in allen Experimenten gelten sollen.

Parallel zur Durchführung der Experimente im realen Szenario soll jedoch auch eine Simulation geschaffen werden, die mit dem realen Labyrinth und dem realen Roboter in Größenverhältnissen, Fahrweise des Roboters, Szenario, Bild- und Farbverarbeitung sowie Rewardbestimmung weitgehend übereinstimmt. Sie dient der Beantwortung prinzipieller methodischer Fragen, zur schnellen Testung von Ideen bzw. zur Ausführung umfangreicher Testreihen, die auf dem realen System zuviel Zeit in Anspruch nehmen würden, aber auch zur Überprüfung und Validierung der eingesetzten Software.

Dies widerspricht jedoch nicht der Konzentration auf reale Roboter in realen Umwelten. Die entstehenden Vergleiche zwischen Leistungsmessungen auf dem realen und dem simulierten System können nur bedingt die Simulation validieren, und eine solche Validierung ist nicht Hauptinteresse der Untersuchungen. Auf eine ausführliche Diskussion der Kriterien und Prüfung von Übereinstimmung zwischen Original und Simulation wird hier verzichtet, da die simulierte Umgebung nur zu qualitativen Abschätzungen herangezogen werden soll, jedoch keinerlei quantitative Verflechtungen zwischen Simulation und Realität vorgesehen sind, wie z.B. ein Vortraining in der Simulation. Die qualitative Vergleichbarkeit ist gegeben, wenn die oben genannten wesentlichen Eigenschaften der Umwelt, wie auch die für die Agenten eingesetzten Softwaremodule identisch sind.

Eine solche Simulation ist in ähnlich angelegten Untersuchungen in der verfügbaren Literatur meist die einzige Testumgebung. In dieser Arbeit wurde sie lediglich als Orientierung für Experimente und Testbett bei der Entwicklung der verschiedenen Agentenstrukturen eingesetzt, als ein Versuchsstadium für die reale Anwendung.

6.2 Struktur der Agenten

In Fortführung bisheriger erfolgreicher Einsätze und zur Erleichterung der Einsicht und des Verständnisses des Verhaltens der Agenten, soll für alle Agenten ein zweistufiger Aufbau festgelegt werden, wie er in Abschnitt 4.9 diskutiert wurde. Er besteht aus einem Clusterer bzw. Vektorquantisierer und einer darauf aufbauenden Schicht, die mit den Methoden des überwachten Lernens ein RL-Verfahren realisiert. Mit gradientenbasierten Verfahren liegen noch keinerlei eigene Erfahrungen oder Implementationen vor, während am Fachgebiet mit wertefunktionsbasierten RL-Verfahren bislang gute experimentelle Erfolge erzielt werden konnten [Gross et al., 1996, Surmeli and Gross, 2000, Surmeli, 2000, Stephan, 2002].

6.2.1 Clusterer

In einer ersten Verarbeitungsstufe erzeugt der Clusterer aus dem kontinuierlichen, reellwertigen Eingabestrom einen diskreten Zustand, der in der zweiten Stufe den Einsatz eines auf diskreten Zuständen basierenden Reinforcementlernverfahrens ermöglicht. Zur Ermittlung eines solchen diskreten Zustandes soll der Einsatz eines der in Abschnitt 4.9.1 diskutierten Verfahren untersucht werden.

Neben einer Diskretisierung des Eingaberaumes, einer Generalisierung über ähnliche Eingangsmuster und einer Datenreduktion ermöglichen diese Strukturen die Inspektion der Gewichte des den Zustand repräsentierenden Neurons und somit eine Protokollierung der sensorischen Wahrnehmung des Agenten. Mit der von den RL-Verfahren realisierten direkten sensomotorischen Abbildung wird zu jedem Zustand auch genau eine auszuführende Aktion ermittelt, die gemeinsam mit dem Zustand eine Plausibilitätsprüfung des Verhaltens des Agenten möglich macht. Ein weiterer Vorteil eines Clusterers besteht in der Reduktion von Redundanzen, die dadurch entstehen, daß die meisten Objekte im Szenario von visuellen Sensoren auf mehr als ein Element des Merkmalsvektors abgebildet werden. So treten häufig miteinander korrelierte Aktivierungen in benachbarten Merkmalen auf.

Aus der in Abschnitt 4.9.1 diskutierten Vielfalt an Clusterern ist einer auszuwählen, der dann für die verschiedenen Agenten zu verwenden ist. Ein inkrementeller Clusterer erzeugt genau diejenige Anzahl an Zuständen, die für eine ausreichende Repräsentation der Umwelt für den Agenten notwendig ist. Obgleich im Interesse einer effektiven Nutzung der begrenzten Ressourcen eines mobilen Roboters ein solcher Clusterer vorzuziehen wäre, stehen dazu doch noch notwendige Untersuchungen hinsichtlich des Zusammenspiels zwischen einer solchen variablen Zustandsrepräsentation und der von festen Zuständen ausgehenden Ermittlung von Zustands-Aktions-Werten aus. Diese waren nicht Gegenstand dieser Arbeit. Insofern sind zwar adaptive Clusterer notwendig, um die Umwelt überhaupt in beherrschbaren Größen darzustellen, jedoch sollte solchen mit fester Knotenanzahl noch der Vorzug gegeben werden.

6.2.2 Reinforcement Lernalgorithmus

Da das zu untersuchende W-Lernen sich auf Zustands-Aktions-Werte stützt, ist die Auswahl möglicher Algorithmen auf wertefunktionsbasierte, insbesondere also *Q-Learning* und *Sarsa* eingeschränkt. Mögliche alternative wertefunktionsbasierte Ansätze arbeiten zum Beispiel ohne Diskontierung und damit mit durchschnittlichem Reward (z.B. R- oder H-Lernen, siehe Abschnitt 4.3.2), was in einem Szenario mit wenigen oder keinen Endzuständen angebracht erscheint. Ein solches Szenario sollte eingesetzt werden, wenn die Modellierung des Verhaltens eines Tieres den großen Rahmen der Untersuchungen bildet. Dieses Verhalten kann als endlose Aufgabe mit wenigen sehr wichtigen Zuständen im Sinne von Zielzuständen aufgefaßt werden. Die nichtdiskontierten Ansätze konnten aber in separaten eigenen Untersuchungen nicht zu einem schlüssigen Ergebnis ihrer Funktion oder gar einem Vergleich mit den genannten Algorithmen geführt werden.

6.3 Koordination der Agenten

Die Koordination mehrerer Expertenagenten anstelle einer Lösung derselben Aufgabe mit einem einzigen monolithischen Agenten wird durch die geringere Komplexität der zu erlernenden Abbildung und die besser festzulegenden Rewardfunktionen und Sensoriken motiviert.

Insbesondere das W-Lernen baut auf den Erfolgen des Reinforcement Lernens in unbekannten Umgebungen auf und nutzt die damit ermittelten Zustands-Aktions-Werte der Experten für eine Vielzahl unterschiedlicher Methoden der Koordinierung. Dabei bleibt es wie die Algorithmen des RL modellfrei und benötigt auch keinerlei Symbole oder Pläne.

Seine Gleichungen bilden selbst eine Art Reinforcement Lernen, allerdings über den Agenten, die den Zustandsraum entsprechend der ihnen zugeordneten Rewards untereinander aufteilen. So kontrolliert nicht nur derjenige Agent den Roboter, für den der aktuelle Zustand am wichtigsten ist und führt folglich seine Expertenaufgabe aus, sondern Agenten können auch feststellen, daß andere Agenten ihre Aufgabe besser erfüllen können als sie selbst, oder daß andere Agenten notwendig sind, bevor sie ihre Aufgabe ausführen können.

Im Sinne des 'Wohls' des Animats als Ganzes können Agenten auch kleinere Verluste in Kauf nehmen, um große Verluste anderer Agenten zu vermeiden. Dies kann als eine Art altruistisches Verhalten angesehen werden.

Eine solche Art selbstorganisierender Bildung einer Heterarchie von Experten, die die Teilaufgaben jeweils besser und schneller erlernen als ein Monolith, läßt vermuten, daß das Ensemble die Gesamtaufgabe auch besser lösen kann als der Monolith, wenn sich die Bewertung der Gesamtaufgabe aus der Überlagerung der Teilaufgaben ergibt.

Entsprechend sollte W-Lernen getestet und, da die Aufgaben überschaubar und in kontrollierter, aber realer Umgebung ablaufen, einem Monolithen gegenübergestellt werden.

6.3.1 Gegenüberstellung von Monolith und Koordination

Aus den Optimalitätsgarantien für die RL-Algorithmen auf MDPs ergibt sich ein Widerspruch zu der oben formulierten Erwartung einer besseren Leistungsfähigkeit des W-Lernen gegenüber dem Monolithen.

Wenn sich jedoch der zu vergleichende Monolith und die durch W-Lernen koordinierten Experten der gleichen Sensorik bedienen, müssen auch eventuell dadurch entstehende Probleme gleich sein, sodaß Leistungsunterschiede nicht auf die Sensorik oder die Lernalgorithmen zurückgeführt werden können. Mit wachsender Komplexität und Anzahl der Aufgaben sollte der Monolith wegen seiner schlechteren Skalierungseigenschaften an die Grenzen seiner Vorteile gegenüber der Koordination stoßen. Dies ist ein weiterer Grund, neben statischen auch die komplexeren dynamischen Quellen im Szenario einzusetzen. Unter Berücksichtigung der Optimalität der gefundenen Policy sollte ein Monolith einer Koordination bezüglich einer einfachen Gesamtaufgabe also zunächst noch überlegen sein, die Experten jedoch bessere Leistungsfähigkeit hinsichtlich der Teilaufgaben aufweisen. Die Vorteile des Monolithen sollten bei komplexeren Aufgaben jedoch schwinden.

Allgemeiner ausgedrückt, stellt dies den Unterschied zwischen der optimalen Erfüllung

der Gesamtaufgabe zum Nachteil der Leistung bei den Teilaufgaben und der optimalen Erfüllung der Teilaufgaben zum Nachteil der Gesamtaufgabe dar. Ein simples Beispiel soll diesen unerwarteten Unterschied deutlicher machen: eine Teilaufgabe bestehe darin, Kuchen zu besorgen, und wird entsprechend am besten durch einen Besuch beim qualifizierten Bäcker gelöst. Eine andere Teilaufgabe bestehe darin, Wein dazuzuholen, und wird am besten durch einen Besuch beim Weinhändler des Vertrauens gelöst. Besteht die Gesamtaufgabe aber darin, beides zu erstehen, dann ist es möglicherweise effizienter, einen Supermarkt aufzusuchen, als die Einzelhändler der Reihe nach zu besuchen. Eine mögliche Lösung eines Monolithen, der vor der Gesamtaufgabe 'Kuchen und Wein' steht, könnte im Gang zum Supermarkt bestehen, während die Koordination diese Lösung nicht betrachtet, weil sie keiner der beiden Experten 'Kuchen holen' und 'Wein holen' vorschlägt. Die Koordination würde eine gute sequentielle Abarbeitung der Teilaufgaben, also eine Route zum Bäcker, zum Weinhändler und zurück finden, die insgesamt jedoch schlechter zu bewerten sein kann als die vom Monolithen gefundene Lösung.

Woraus aber kann ein solcher Leistungsnachteil für die Koordination entstehen? Kann die höhere Lerngeschwindigkeit, bessere Kompetenz und der geringere Ressourceneinsatz der Experten nicht doch genutzt werden?

Eine Koordination, die sich ausschließlich auf die durch die Experten gesammelten Erfahrungen stützt, muß aus folgenden Gründen einem Monolithen hinsichtlich der Gesamtaufgabe unterlegen sein.

- Zunächst ist augenscheinlich, daß die Koordination die Kosten des Übergangs von der Erfüllung einer Teilaufgabe zur nächsten gar nicht betrachtet (im Beispiel der Weg zum Weinhandel, zwischen Weinhändler und Bäcker und zurück im Vergleich mit dem Weg einmal zum Supermarkt und zurück).

- Weiterhin erfährt die Koordination den Reward für die Gesamtaufgabe nicht, anhand dessen der Monolith trainiert wird, obgleich sie daran gemessen wird. Natürlich sind die eben erwähnten Kosten auch Teil dieses Gesamtrewards, das gilt aber auch für Synergien zwischen den Teilaufgaben und nicht zuletzt für die Wichtung der einzelnen Teilaufgaben im Gesamtreward.

- Durch Exploration des gesamten Zustandsaktionsraumes kann der Monolith anhand des Gesamtrewards eine Balance zwischen Kosten und Rewards der einzelnen Teilaufgaben finden, ohne je die Teilrewards einzeln erlebt zu haben.

- Der Monolith kann die Eingaberäume der in der Koordination getrennten Agenten als Landmarken für jede der Teilaufgaben benutzen und hieraus Vorteil ziehen.

- Er betrachtet so auch die Optimalität einer Handlungskette, während die Koordination auf eine Betrachtung des jeweils nächsten Schrittes beschränkt ist. Sie ähnelt hierin einem reaktiven, schwachen RL-Verfahren mit $\gamma = 0$. Beide haben keine Möglichkeit zur impliziten Planung. Für die Koordination bezieht sich das natürlich ausschließlich auf die Gesamtaufgabe, während die einzelnen Q-Agenten über die implizite Planung verfügen.

- Wie in Abschnitt 5.1.4 erläutert wurde, stehen dem Monolithen auch Policies zur Verfügung, die eine Koordination gar nicht betrachten kann (siehe Milch-Wodka-Beispiel).

Dieser gedankliche Vergleich zwischen Monolith und W-Lernen als Koordination motiviert im folgenden einige Erweiterungen am W-Lernen, die dessen Leistungsfähigkeit der des Monolithen näher bringen und die Vorteile der Einzelagenten bewahren könnten.

6.3.2 Gleichzeitiges Lernen der Agenten und des W-Lernens

Vor dem Hintergrund einer Anwendung der Koordinierung mehrerer Expertenagenten in realen Umwelten mit den dafür bestehenden Begrenzungen hinsichtlich verfügbarer Trainingszeit und einzuschränkender Kosten der Erfahrungen mit der Umwelt erscheint ein separates Anlernen der einzelnen Agenten zumindest verschwenderisch. Daher müssen Methoden gefunden werden, wie die Experten effizient und möglichst zusammen mit ihrer Koordination trainiert werden können, sofern es sich nicht um anderweitig zur Verfügung gestellte und so schon austrainierte Experten handelt (z.B. [Sutton et al., 1999]). Ausgangspunkt ist die Messung der Leistungsfähigkeit von Agenten und Koordination, wenn einfach Agenten und Koordination ohne besondere Vorkehrungen gleichzeitig trainiert werden. Die Schritte, in denen ein Agent nur beobachten kann, welche Auswirkungen die Aktion eines anderen Agenten haben, sind für diesen gleichbedeutend mit einer Exploration und sollten ähnliche Effekte bewirken wie ein offline Vortraining aus einem feststehenden Datensatz, bei dem ein Agent auch keine Kontrolle über die ausgeführte Aktion hat. Dies gilt wiederum besonders für Agenten mit Traces (d.h. $\lambda > 0$), da deren Länge, bzw. die Korrektheit des erlernten Q-Wertes, von der Wahl der Aktion abhängt.

WOOLDRIDGE untersucht in [Jennings et al., 1998] anhand des Systems *Bazaar* die Auswirkungen eines solchen simultanen Trainings: wenn alle Agenten lernen, wird der Gesamtreward fast optimal und die Expertise der Agenten ähnlich hoch; wenn kein Agent lernt, sind die Expertisen ähnlich hoch, aber der Gesamtreward niedrig; wenn ein Agent lernt, erhöht er seine Expertise auf Kosten der Expertise der anderen und des Gesamtrewards. Entsprechend scheint der Ansatz, alle Experten gleichzeitig lernen zu lassen, erfolgversprechend und effizient.

6.3.3 Exploration im W-Lernen

Da auch das W-Lernen in den Varianten, in denen die W-Werte tatsächlich gelernt werden müssen und nicht einfach berechnet werden können, eine Art des Reinforcement Lernens darstellt, sollen Explorationsvarianten experimentell untersucht werden. So erhalten die Agenten die Möglichkeit, tatsächlich zu erfahren, welche Auswirkungen die Festlegung der auszuführenden Aktion durch verschiedene Leader auf die eigenen Rewards in den unterschiedlichen Zuständen haben, und die Kompetenz der Leader für die eigene Aufgabe abzuschätzen. Sie können durch die Exploration auch Synergien erfahren, die erst bei Betrachtung mehrerer Teilaufgaben entstehen und die Exploration zunächst suboptimal erscheinender Aktionen voraussetzen.

Ausgehend vom Training der Einzelagenten, sollten zunächst zwei Arten der Exploration betrachtet werden. Eine betrifft die Initialisierung der W-Werte mit sehr geringen Werten, analog einer optimistischen Initialisierung im Q-Lernen. Da die Leader im WL nicht lernen, sondern nur die anderen Agenten, können diese so ihre Verluste anpassen und die Wahrscheinlichkeit der Exploration mehrerer Leader in einem Zustand wächst.

Bei der anderen Variante der Exploration wird auch im W-Lernen eine Boltzman-Auswahl unter den W-Werten getroffen, die mit abklingender Temperatur in die normale Maximumauswahl übergeht. Letzte geht auch auf eine Anregung in [Humphrys, 1997] zurück und verhindert besonders die Auswirkungen von Rewards, die an den extremen Enden der entsprechenden Verteilung liegen und so falsche W-Werte erzeugen.

6.3.4 Modifikationen am Lernalgorithmus

Die Leistungsfähigkeit der verschiedenen Formen der Koordinierung werden mittels des Gesamtrewards miteinander verglichen. Diesen Gesamtreward verwendet HUMPHRYS auch, um nach erfolgtem Q-Lernen die besten Wichtungen der einzelnen Agenten zu bestimmen. Werden alle Teilaufgaben zusammengenommen von einem Monolithen ausgeführt, der z.B. mit Q-Lernen trainiert wird, so ist es wiederum der Gesamtreward, der zum Training herangezogen wird. Andererseits erfahren die verschiedenen Formen des W-Lernens nie, woran sie gemessen werden. Darum erscheint es naheliegend, den Gesamtreward als Reward auch für die Koordination heranzuziehen. Während die einzelnen Agenten ihr Verhalten aus den Einzelrewards erlernen, wird die Koordination anhand des Gesamtrewards trainiert. Nach KIM [Kim, 1994] ist die Betrachtung der Konsequenzen der Aktionen einzelner Agenten auf das Kollektiv entscheidend für dessen Leistungsfähigkeit. Diese Möglichkeit eröffnet sich dem W-Lernen erst nach Einbeziehung des Gesamtrewards.

Steht ein solches Rewardsignal aber zur Verfügung, dann drängt sich nahezu automatisch der Gedanke auf, es zu einem Reinforcement Lernen wie für Einzelagenten, z.B. Q-Lernen, zu nutzen, nur auf einer übergeordneten Ebene: der Koordinierungsagent würde lernen, in welcher Situation welcher Agent aktiviert werden soll, um maximalen Gesamtreturn zu erzielen. Da die Experten im WL jedoch austrainiert sind und somit immer gleiche Aktionen vorschlagen, soll hier der Koordinationsagent direkt die Werte der einzelnen Aktionen lernen. Ein solches Verfahren, allerdings ohne die Konsequenz, den Gesamtreward direkt zu nutzen, diskutiert auch HUMPHRYS in [Humphrys, 1997] bezüglich der Arbeiten von LIN. Da der Koordination aber keine Ressourcen zur sensorischen Verarbeitung und Zustandsermittlung zur Verfügung stehen, muß sie dafür auf die Experten zurückgreifen.

Die Agenten können nun zur Korrektur des Gesamtrewards herangezogen werden. Eine Korrektur des Gesamtrewards ergäbe sich aus der Betrachtung der Expertise der Agenten, d.h. der Einzelstrategien, die sonst unbeachtet bleiben würden. Je nach bevorzugter Variante des W-Lernen können sie als Gewinn g oder Verlust v in Gl. 6.1 einfließen.

Aufgrund der Betrachtung des Wertes des Folgezustandes $V(s')$ des Gesamtsystems wird hiermit auch die dem W-Lernen fehlende Betrachtung der Handlungskette aufgehoben und eine Suche ähnlich der in Abschnitt 5.3.3.2 realisiert. In Anlehnung an die Vorbilder aus dem RL der Expertenagenten (siehe Kapitel 4) wird die Methode in dieser Arbeit als *WQ* bezeichnet. r_G bezeichnet hierbei den Gesamtreward und s den durch die Experten ermittelten Zustand des Gesamtsystems.

$$WQ(s,a) \leftarrow (r_G + \gamma V(s')) + (g - v) \tag{6.1}$$

In Gl. 6.1 beinhaltet der erste Term auf der rechten Seite ein normales 1-Schrittlernen mit *Sarsa*, der zweite die Differenz aus Verlust und Gewinn, wie sie jeweils mit den

verschiedenen Varianten des WL berechnet werden können. So wird einerseits ein RL-Algorithmus realisiert, der dem des Monolithen gleicht, und andererseits die Expertise der Agenten in Betracht gezogen. In Experimenten ist zu untersuchen, ob diese Art der Nutzung des Gesamtrewards einige Leistungsnachteile der Koordination wettmachen kann.

Über diese Arbeiten hinaus kann der Gesamtreward jedoch auch zur Ermittlung angemessener Wichtungen der Agenten im W-Lernen herangezogen werden. In der ursprünglichen Formulierung des WL ergeben sich die Wichtungen der Agenten mittelbar aus den Q-Werten, die die Experten bei ihren individuellen Trainings erlernt haben. Es erfordert wiederum einen erheblichen Aufwand seitens des Designers oder aber den Einsatz weiterer adaptiver Methoden, wie z.B. genetischer Algorithmen [Humphrys, 1997], um angemessene Wichtungen für die einzelnen Agenten zu finden. Für diese adaptiven Methoden wird wiederum der Gesamtreward herangezogen. Dieser kann jedoch auch direkt beim W-Lernen zur Adaptation angemessener Wichtungen der Agenten benutzt werden.

6.3.5 Zustandsabhängige Priorisierung der Agenten

Zwar realisiert WL in seiner ursprünglichen Formulierung von Humphrys schon eine Wichtung der Experten, um ihnen unterschiedliche Prioritäten zu verleihen. Diese Wichtung wird nach dem W-Lernen in einem separaten genetischen Evolutionsprozeß ermittelt, kann also nicht zu einer Koordinierung der Experten hinsichtlich einer optimalen Erfüllung der Gesamtaufgabe beitragen. Für die Koordinierung sind die Wichtungen der Agenten fest und ergeben sich aus der Höhe der Q-Werte. Diese Wichtungen gelten für jeden Agenten jeweils über den gesamten Zustandsraum.

Eine solche generelle Wichtung der Agenten erscheint jedoch nicht optimal. Um in wichtigen Situationen das gewünschte Verhalten zu erzeugen, kann es vorkommen, daß die verteilten Rewards und damit die für die Ermittlung der W-Werte zugrundeliegenden Q-Werte eines Agenten so hoch gesetzt wurden, daß ein Agent auch Zustände gewinnt, die für einen anderen Agenten wichtiger wären. Erhält ein Agent Rewards im Bereich $[0 \ldots 1]$, so ist eine Differenz und damit ein W-Wert von 0.5 sehr erheblich. Für einen Agenten, der Rewards im Bereich $[0 \ldots 100]$ erhält, ist ein solcher W-Wert nahezu unerheblich und kann aus dem Rauschen entstehen. Der letztere Agent würde den Wettbewerb nahezu immer dominieren, aber nicht immer berechtigterweise. So kann die Rewardstruktur der einzelnen Agenten bewirken, daß in einem bestimmten Zustand ein Agent zwar im Vergleich zu einem anderen Agenten einen großen Verlust erleidet, dieser Verlust aber für den Agenten selbst eigentlich relativ geringfügig ist. Da sich der andere Agent dort nicht durchsetzen kann, wo sein Verhalten wichtig wäre, verringert sich der erzielbare Gesamtreward.

Diese generelle Wichtung macht es den Agenten zwar möglich, sich in Zuständen durchzusetzen, die besonders wichtig für sie sind, doch sie ermöglicht keine flexible Wichtung der Agenten im Zustandsraum zugunsten eines optimalen Gesamtrewards. Gl. 6.2 soll die geplante Modifikation beschreiben: die durch das W-Lernen ermittelten W-Werte werden durch die Wichtungen der Agenten modifiziert. Der W-Wert des Agenten i im Zustand s ergibt sich aus dem originalen W-Wert $W_i^o(s)$ und der Wichtung $w_i(s)$.

$$W_i(s) = w_i(s) \cdot W_i^o(s) \tag{6.2}$$

Bei der Zuweisung der Rewards an die Teilaufgaben sowie des Gesamtrewards steht der Designer vor der Aufgabe, die Teilrewards und den Gesamtreward so auszubalancieren, daß das gewünschte Verhalten entsteht. In der bisherigen Praxis (z.B. [Humphrys, 1997]) ist das ein komplizierter, zeitaufwendiger Prozeß. Während es meist einfach ist, die einzelnen Teilrewards zuzuweisen, sind die möglichen Interaktionen der Teilrewards und Konsequenzen für das Gesamtverhalten schlecht abzuschätzen.
Durch adaptive Wichtung der Agenten können vom Designer ungünstig gewählte Rewards für die Teilaufgaben korrigiert werden, bzw. die Rewardstruktur der Agenten für die Koordinierung im Hinblick auf die Maximierung des Gesamtrewards angepaßt werden. Durch eine flexible Zuordnung könnten die Agentenwichtungen den unterschiedlichen Anforderungen im Zustandsraum besser angepaßt werden und einer einzigen Wichtung für die Agenten hinsichtlich des erzielten Gesamtrewards überlegen sein. Die Abbildung 6.1 soll das Prinzip veranschaulichen.

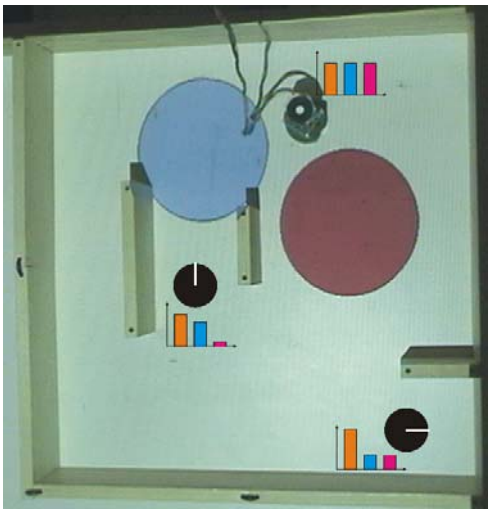

Abbildung 6.1: ***Prinzip der zustandsabhängigen Priorisierung*** *Der hellgraue/blaue Kreis sei eine Futterquelle, der dunkelgraue/rote eine Gefahrenzone. Für jeden Zustand in der Umwelt gibt es eine optimale Kombination der Wichtungen der Agenten (vgl. Balkendiagramme in der Szene). Der linke Balken soll die Wichtung eines Kollisionsagenten symbolisieren, der mittlere Hunger und der rechte Flucht. Rechts unten sollte die Koordination dem Kollisionsagenten folgen, in der Mitte sollte Hunger die Richtung bestimmen und der Kollisionsagent die Wände vermeiden, oben sind alle drei gleichberechtigt.*

Eine Möglichkeit der zustandsabhängigen Priorisierung ergibt sich aus der Relevanz der Aktionen in einem bestimmten Zustand, wie sie Humphrys für die Ermittlung statischer W-Werte (vgl. Abschnitt 5.2.1) betrachtet. Die Relevanz einer bestimmten Aktion in einem Zustand ergibt sich aus der Relation zu den anderen verfügbaren Aktionen. Diese Wichtung beantwortet die Frage: Wie wichtig ist die Aktion für den einzelnen Agenten?

Q-Werte werden in die Ermittlung der W-Werte eingebracht, nachdem sie unter den anderen Q-Werten desselben Agenten in dieser Situation gewichtet wurden. Die Relevanz einer Aktion ergibt sich aus der Abweichung ihres Q-Wertes vom Durchschnitt aller Q-Werte in diesem Zustand.

$$w_i(s) = \sqrt{\frac{1}{|A|} \sum_{a \in A} (Q_i(s,a) - \overline{Q_i(s)})^2} \tag{6.3}$$

Mit Gl. 6.3 wird eine zustandsabhängige Wichtung der Agenten erreicht, indem ermittelt wird, wie wichtig einem Agenten seine Aktion im Vergleich zu den anderen verfügbaren Aktionen ist.

Dies verhindert, daß Agenten mit sehr hohen Reward- und Q-Levels Zustände gewinnen, in denen zwar ihr Verlust im Vergleich zu anderen Agenten hoch wäre, aber dies nur aufgrund unproportional hoher Q-Level. Eigentlich ist der Zustand im Vergleich zu anderen Zuständen nicht wichtig für den Agenten.

Eine weitere Möglichkeit der Ermittlung von Agentenwichtungen ergibt sich aus der Verwendung des Gesamtrewards. Aus der Differenz des in einem bestimmten Zustand s erhaltenen Gesamtrewards r_G in aufeinanderfolgenden Durchläufen mit unterschiedlichen Leadern, ergibt sich für den Leader eine Änderung seiner Wichtung: erzielte er mehr Gesamtreward als der Leader im letzten Durchlauf, wird seine Wichtung in diesem Zustand erhöht (siehe Gl. 6.4). Die hierfür notwendige Exploration verschiedener Leader kann mit einer der in Abschnitt 6.3.3 diskutierten Methoden erreicht werden. So könnte mit Hilfe des Gesamtrewards der optimale Leader für eine gegebene Wichtung der Teilaufgaben im Gesamtreward ermittelt werden. Diese Untersuchung beantwortet die Frage: Wie wichtig ist der Agent für den Gesamtreward in dem jeweiligen Zustand?

$$w(i,s) \leftarrow \beta * (r_G(t,s) - r_G(t-1,s)) \tag{6.4}$$

6.4 Verhaltensexperimente

Wie in Abschnitt 6.1 erwähnt, soll das Szenario aufgrund der zu erwartenden unterschiedlichen optimalen Policies sowohl Objekte, die positiven Reward verteilen, als auch solche mit negativem Reward enthalten. Weiterhin sollen sich die Objekte statisch oder dynamisch verhalten.

Zur Überprüfung der oben aufgestellten Erwartungen hinsichtlich der Leistungen der Agenten, des Monolithen und des W-Lernens ist das Szenario mit den Teilaufgaben Kollisionsvermeidung, Vermeidung von Gebieten negativen Rewards und Anfahrt von Gebieten positiven Rewards stufenweise umzusetzen, um darin jeweils Agenten einzeln zu trainieren und ihre Leistungsfähigkeit zu ermitteln.

Das Szenario soll zunächst nur mit statischen Objekten realisiert und die Agenten jeweils nur mit dem ihrer Teilaufgabe zugehörigen Teil des Eingaberaumes einzeln trainiert werden. Daraus ergeben sich Agenten, die aufgrund der Beschränkung der verfügbaren Eingangsinformation nur jeweils ein bestimmtes Verhalten, wie Kollisionsvermeidung oder Zielanfahrt, realisieren können. Da jedoch alle Agenten im selben Szenario trainiert werden, können durch Protokollierung aller Rewards ihre individuellen Kompetenzen im Vergleich zu den anderen Agenten und einem sich komplett zufällig bewegenden Agenten festgestellt werden.

Die Agenten sollen entsprechend ihre Kompetenz bezüglich negativen oder positiven Rewards erst einzeln unter Beweis stellen, bevor sie mit W-Lernen koordiniert und dessen Performanz mit der eines Monolithen verglichen wird. In der folgenden Ausbaustufe werden die Agenten anhand des Szenarios mit dynamischen Objekten trainiert, indem die statischen Objekte sich zum Roboter hin (negativer Reward) oder von ihm weg (positiver Reward) bewegen. Auch hier werden zunächst die Einzelagenten betrachtet, und danach ihre Koordination. Die Leistungen der mit dynamischen Objekten trainierten Agenten sind mit denen der Agenten zu vergleichen, die nur mit statischen Objekten trainiert wurden, aber mit dynamischen getestet. Dies soll die erweiterte Kompetenz der dynamisch trainierten Agenten zeigen.

6.5 Zusammenfassung

In diesem Kapitel wurden anhand gedanklicher Betrachtungen die Bestandteile des Szenarios mit negativen und positiven Rewards als statische und dynamische Objekte motiviert, das sowohl real als auch in einer Simulation umgesetzt wird.

Weiterhin wurde die Konfiguration eines Agenten auf eine Kombination aus einem Clusterer und einem wertefunktionsbasierten RL-Algorithmus festgelegt. Zur Koordination der verschiedenen Teilaufgaben soll einerseits ein Monolith, der genau wie ein Einzelagent aufgebaut ist, eingesetzt werden, und andererseits W-Lernen. Die von diesen beiden Koordinationen erbrachten Leistungen sollen gegenübergestellt werden. Mögliche Nachteile des WL gegenüber dem Monolithen sollen durch die vorgeschlagenen Erweiterungen des WL aufgehoben werden.

Abschließend wurde eine stufenweise Durchführung für die auszuführenden Experimente entworfen, deren konkrete technische Umsetzung Gegenstand des nächsten Kapitels ist.

Kapitel 7

Experimentelle Umsetzung

Nachdem im vorangegangenen Kapitel theoretische Überlegungen zum Szenario, den Agenten und der Koordination angestellt wurden, behandelt dieses die konkrete Ausgestaltung des Szenarios mit den Teilaufgaben für die Expertenagenten, sowie dessen Umsetzung im realen Labyrinth und in der Simulation. Es werden neben den Details zur Sensorik und Aktuatorik des Roboters mit entsprechenden Vorverarbeitungen auch die Ermittlung der Leistungen der verschiedenen Agenten und die verwendeten Maße erläutert.

Weiterhin werden Voruntersuchungen vorgestellt, die für Festlegungen zur Durchführung der eigentlichen Leistungsuntersuchungen dienten. Sie betreffen die Konfiguration der einzelnen Agenten und den Modus für das Training der Agenten.

7.1 Szenario

Im entworfenen Szenario werden sowohl Objekte oder Quellen negativen als auch positiven Rewards eingesetzt, die sich in verschiedenen Ausbaustufen des Szenarios räumlich statisch, aber auch dynamisch verhalten sollen. Während für statische Rewardquellen viele mögliche Alternativen eingesetzt werden (Beispiele sind in den Literaturverweisen in Abschnitt 3.5 enthalten), ist die praktische Umsetzung von dynamischen Quellen erheblich seltener und schwieriger.

Die folgende Aufzählung gibt eine Übersicht über die verschiedenen Objekte bzw. Rewardquellen im Szenario.

- **Hindernisse:** Statische Hindernisse wie Außenwände und Hindernisse (Wände) im Labyrinth sollen vermieden werden. Im Freiraum dazwischen soll eine schnelle, gerade Fahrt angestrebt werden. Dies stellt eine Kombination einer Vermeidungs- mit einer Erhaltungsaufgabe dar, die in der Literatur vielfach Basis weiterführender Aufgaben ist.

- **Futterquelle:** Als Erreichungsaufgabe existiert eine statische Quelle positiven Rewards, die als Futterstelle oder Tränke angesehen werden kann.

- **Wild:** Als weitere Erreichungsaufgabe existiert eine dynamische Quelle positiven Rewards, die sich vom Roboter wegbewegt. Dabei ist sie langsamer als der Roboter,

wenn der sich so schnell wie möglich bewegt. Mit dieser vereinfachenden Randbedingung benötigt der Roboter keine komplizierte Planung, die nötig wäre, um z.B. ein schnelleres Wild in einer Sackgasse zu fangen.

- **Gefahrenzone:** Als Vermeidungsaufgabe existiert eine statische Quelle negativen Rewards, die als gefährliches Gebiet angesehen werden kann.
- **Jäger:** Als Vermeidungsaufgabe mit dynamischer Quelle negativen Rewards wird ein Jäger vorgesehen, der bestrebt ist, den Roboter zu erreichen. Auch der Jäger bewegt sich langsamer als der Roboter, wenn dieser mit maximaler verfügbarer Geschwindigkeit gefahren wird. So benötigt der den Roboter steuernde Agent keine Planung, die ihm ermöglichen würde, Sackgassen zu vermeiden oder gezielt unvorhersehbares Verhalten zu entwickeln.

Dieses Szenario wurde sowohl als reales System mit einem mobilen Miniroboter Khepera in einem speziell angefertigten Labyrinth (siehe Abschnitt 7.2), als auch in einer entsprechenden Simulation (Abschnitt 7.3) umgesetzt.

7.1.1 Szenario mit dynamischen Objekten

Für eine Realisierung statischer Objekte existiert eine Fülle von Möglichkeiten, die weitgehend vom verwendeten Roboter und dessen Sensorik abhängt. Lichquellen, unterschiedlich elektrisch leitfähige Objekte oder visuelle Texturen kommen in der Literatur zum Einsatz.

Naheliegend und in ähnlichen Untersuchungen oft verwendet wird eine Umsetzung verschiedener beweglicher Subjekte mit mehreren Robotern. Sie würde allerdings leicht zur Verwirrung und Verwechslung der hier vorgestellten Multiagentenuntersuchungen mit Multiroboteruntersuchungen führen. Aus diesem und anderen, praktischen Gründen wurde eine Lösung mit einem Projektor (Beamer) geschaffen, der die oben aufgeführten Objekte als farbige Muster in ein Experimentierfeld projiziert. Eine Umsetzung mit dem Beamer in einem realen Labyrinth ist weit flexibler hinsichtlich der Anzahl, Form, Farbe und Dynamik der Objekte sowie kostengünstiger als ein Einsatz weiterer Roboter. Auch konnten statische und dynamische Objekte auf die gleiche Art und Weise realisiert werden. Eine solche Lösung war möglich, da der einzusetzende Roboter Khepera über eine Farbkamera verfügte und daher jedem Objekt eine Farbe zugeordnet werden kann.

Um eine aktive Flucht bzw. Verfolgung des Roboters zu ermöglichen, wurde der Aufbau um eine Kamera ergänzt, die der Lokalisierung des Roboters dient. Da z.B. der Jäger den Roboter finden soll, muß er Informationen über dessen Aufenthaltsort relativ zur eigenen Position erhalten. Die Kamera wurde in unmittelbarer Nachbarschaft zum Beamer angebracht und erzeugt so ein Überblicksbild (siehe Abb. 7.1). Zur Lokalisation des Roboters mußten Algorithmen eingesetzt werden, die den Roboter anhand eines Musters lokalisieren. Die üblicherweise verwendeten, einfachen Methoden mittels Farblokalisation mußten verworfen werden, da der Roboter auf die vom Beamer projizierten Farbflecken fährt und eine ihm zugeordnete Farbe sich durch das Aufficht stark ändert. Dadurch würde seine Lokalisierung sehr erschwert oder unmöglich gemacht. Die eingesetzten Algorithmen werden im Anhang 10 dargestellt und können auch zur Protokollierung der Fahrspur des Roboters (Abb. 7.1) oder zur Bereitstellung der notwendigen sensorischen Informationen für eine MDP-Darstellung des Szenarios (siehe Abschnitt 7.6.4) benutzt werden.

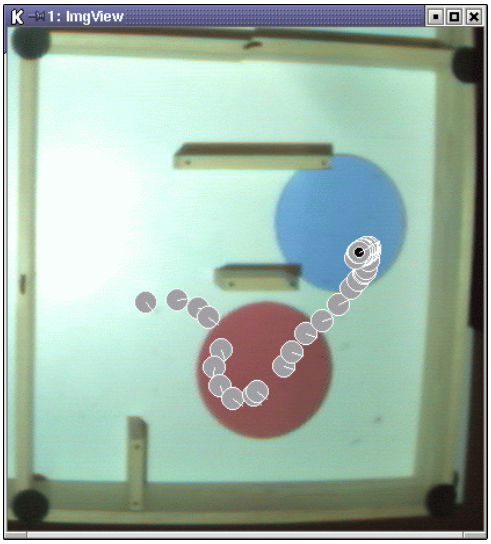

Abbildung 7.1: ***Beispiel der Nutzung der Positions- und Orientierungsbestimmung zur Fahrspuraufzeichnung***. *Der Roboter wurde in das reale Labyrinth eingesetzt und seine Positions- und Orientierungsdaten bis zum Abbruch des Versuches ermittelt.*

7.1.2 Steuerung der dynamischen Objekte

Zur Steuerung der dynamischen Objekte wurde in jedem Schritt jeweils deren Mittelpunkt mit der ermittelten Position des Roboters verglichen. Weiterhin wurden die Positionen der Hindernisse im Labyrinth ermittelt.

Um die Bewegung des *Jägers* zu implementieren, wurde der direkte, gerade Weg zur Roboterposition berechnet. An Hindernissen auf diesem Weg wurde die Positionsänderung in ihre Bestandteile, je eine X- und eine Y-Komponente zerlegt. Diese Komponenten wurden einzeln geprüft, ob sie zu einer Kollision mit einem blockierenden Hindernis führen.

Es wurde eine resultierende Bewegung ausgeführt, die sich aus der oder den nicht blockierten Komponenten ergab. Für die Flucht des *Wildes* wurden die Positionsänderungen mit genau denselben Berechnungen ermittelt, aber im Vorzeichen umgekehrt.

Die dynamischen Quellen bewegen sich jeweils direkt vom Roboter weg bzw. auf ihn zu. Sie haben dabei eine konstante Geschwindigkeit, die nicht von der Entfernung zum Roboter abhängt. Diese Bewegung wurde nicht mit Rauschen beaufschlagt, ist also komplett deterministisch. In weiterführenden Untersuchungen könnten diese Steuerungen komplexer gestaltet werden, um die kognitiven Anforderungen an die Agenten zu erhöhen, die den Roboter steuern und auf die Bewegungen der Objekte reagieren oder diese antizipieren müssen. Die Bewegungen auf direkter Linie könnten verrauscht werden, teilweise zufällig sein (für das Wild z.B. zufällig, aber immer weg vom Roboter), die Bewegungsgeschwindigkeit kann vom Abstand zum Roboter abhängen, es können einfache reaktive Agenten eingesetzt werden (Braitenbergvehikel, Homeostasis) oder lernende Agenten, wie sie auch den Roboter steuern.

7.2 Das Experimentalsystem

Der eingesetzte experimentelle Aufbau bestand aus dem Miniroboter Khepera und einem quadratischen flachen Kasten der Maße $60*60*10cm$ mit Holzwänden und beschichtetem, diffus reflektierendem Boden, mit dem das Labyrinth aufgebaut werden konnte. Dazu konnten weitere Holzstücke verschiedener Länge und Höhe als flexible Hindernisse in das Labyrinth eingebracht werden.

Über dem Labyrinth war ein Projektor angebracht, mit dem in das gesamte Labyrinth sowohl Farbflecken als auch Hintergrundfarbe in frei wählbarer Anzahl, Form und Farbe projiziert werden können. Auf diese Weise können beliebig viele Bereiche mit unterschiedlicher Bedeutung für die Agenten erzeugt werden und diese auch dynamisch, also beweglich oder anderweitig veränderlich, gestaltet werden.

Zur Steuerung der dynamischen Objekte wurde neben dem Projektor eine Kamera angebracht, mit deren Hilfe sowohl der Roboter als auch die Außenkanten und Wände des Labyrinths detektiert werden können. Damit können die farblich abgesetzten beweglichen Objekte in Richtung des Roboters oder von ihm weg gesteuert werden, ohne die Begrenzungen des Labyrinths zu verlassen oder Wände zu überspringen. Abbildung 7.2 links veranschaulicht den Aufbau, Abb. 7.2 rechts zeigt eine Draufsicht mit Khepera.

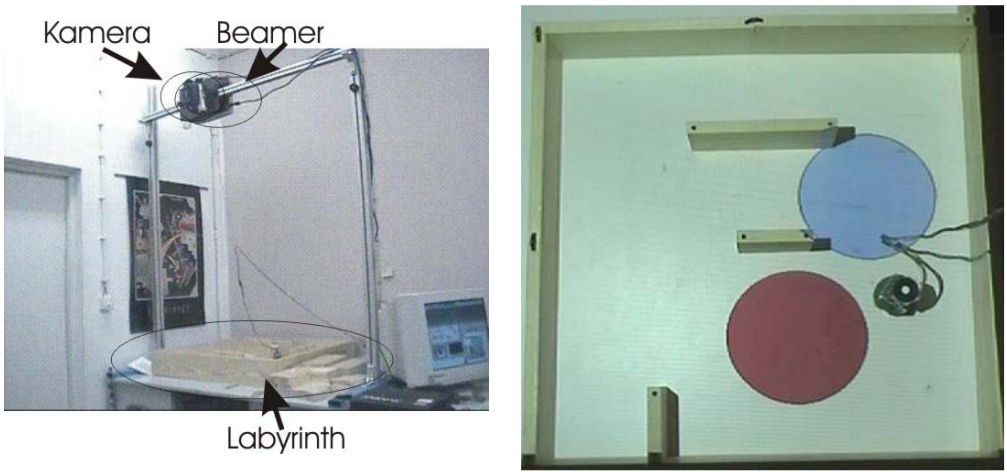

Abbildung 7.2: ***Eingesetzter Experimentalaufbau.*** *Links eine Gesamtansicht mit Labyrinth (unten im Bild) und Träger mit Beamer zur Projektion der Objekte sowie Kamera zur Roboterlokalisierung, rechts eine Draufsicht der Kamera auf das Labyrinth mit Miniroboter Khepera, Hindernissen und projizierten Objekten (hellgrau/rot: Futterquelle, dunkelgrau/blau: Gefahrenzone).*

Die als Rewardquellen eingesetzten Objekte wurden vom Beamer als Kreise bestimmten Durchmessers und festgelegter Farbe in das Labyrinth projiziert. Der Durchmesser sollte möglichst gering gehalten werden, um auch mit mehreren Kreisen unterschiedlicher Bedeutung ein möglichst kleines Labyrinth erzeugen zu können, das beherrschbare Trainingszeiten ermöglicht.

Sowohl die Form als Kreis als auch der minimale Durchmesser der Kreise ergaben sich aus der Charakteristik der Kamera des Khepera. Diese und die des des verwendeten

Beamers bestimmten die Festlegung der für die Objekte bzw. Rewardquellen eingesetzten Farben. Die Festlegung der projizierten Farben wird in Abschnitt 7.2.3.3 erläutert.

Der Miniroboter Khepera wurde in einem Aufbau aus dem Grundmodul und einem zusätzlichen Kameramodul benutzt, wie sie in Abb. 7.3 links abgebildet sind.

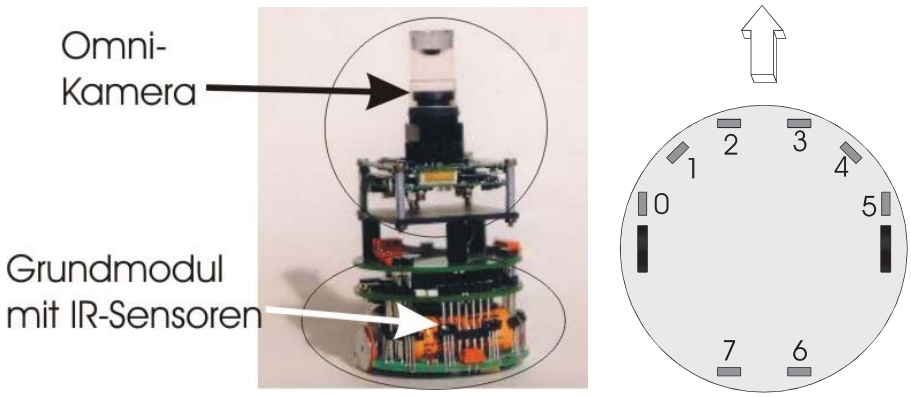

Abbildung 7.3: *Khepera mit Omnikamera (links) und Anordnung der Infrarotsensoren (rechts).*

7.2.1 Die Effektoren

Für alle Experimente wurden ausschließlich die Fahrbewegungen des Miniroboters Khepera als Aktuatorik verwendet. Eine Bewegung wurde dabei als zweidimensionaler Vektor aus dem Betrag der Fahrgeschwindigkeit $|v| \in [-1, 3, 7]$ Einheiten und dem Lenkwinkel $\phi \in [-30°, 0°, 30°]$ beschrieben. Die Einheiten für die Geschwindigkeit beziehen sich direkt auf die Ansteuerung der Räder und verkörpern ca. $8\frac{mm}{s}$.

Damit standen den Agenten jeweils 9 Fahrkommandos als Aktionen zur Verfügung. Das Fahrkommando wurde umgesetzt, indem zunächst eine Drehung auf der Stelle mit vorgeschriebenen Radgeschwindigkeiten und einer Dauer ausgeführt wurde, die sich nach dem gewünschten Lenkwinkel richtete, und anschließend eine Vorwärtsbewegung mit der gewünschten Fahrgeschwindigkeit für eine festgelegte Zeit absolviert wurde.

Hervorzuheben ist, daß den Agenten immer alle 9 Aktionen zur Verfügung standen, jedoch immer eine Bewegung ausgeführt werden mußte, also die Aktion 'Stehenbleiben' nie verfügbar war. Diese Aktion wurde nach ersten Tests ausgeschlossen, da sie sowohl für einen Erreichungsagenten in seinem Zielgebiet als auch für einen Vermeidungsagenten ausserhalb seines Gefahrengebietes schnell Verhalten produzierte, das der Grundidee eines *mobilen* Systems zuwider gelaufen wäre.

7.2.2 Die Infrarotsensoren

Die verwendeten sensorischen Informationen richteten sich zwangsläufig nach der verfügbaren sensorischen Ausstattung des Miniroboters Khepera. Diese schließt acht Infrarotsensoren, sowie eine omnidirektionale Farbkamera ein. In Abbildung 7.3 links befinden sie sich auf der untersten Ebene des Grundmoduls, ihre Anordnung zeigt Abb. 7.3 rechts.

Die Infrarotsensoren des Khepera gehören zu dessen Grundausstattung und können in zwei Modi betrieben werden. Im passiven Modus empfangen sie infrarote Strahlung und können zur Detektion und Entfernungsmessung von Lichtquellen eingesetzt werden. Dieser Modus fand keine Anwendung.

Im aktiven Modus senden sie auch infrarote Strahlung aus und können somit, über die Reflektion der Strahlung und ihre Messung, für die Entfernungsmessung zu den reflektierenden Objekten verwendet werden. Sie sind symmetrisch zur Vorwärtsrichtung des Robors angebracht, jedoch mit deutlicher Vorzugsrichtung nach vorne, sodaß sie in Fahrtrichtung sehr viel höhere seitliche Auflösung haben als an den Seiten oder nach hinten. Ihre Messungen liefern nur im Abstand von ca. 2 . . . 5cm Ergebnisse, die allerdings sehr verrauscht sind. Diese Infrarotsensoren wurden zur Kollisionsdetektion und Entfernungsmessung zu den hölzernen Hindernissen zur Kollisionsvermeidung eingesetzt.

7.2.3 Die omnidirektionale Farbkamera

Weiterhin verfügt der Khepera über eine Farbkamera, die mit Hilfe eines Parabolspiegels, auf den sie von unten gerichtet ist, wie eine Kamera mit Rundumsicht, eine Omni(visions)kamera funktioniert. Aufgrund ihrer nahezu universellen Einsetzbarkeit auf Robotern unterschiedlichster Größen und Fabrikate werden Informationen, die aus einer Bild- und Farbverarbeitung gewonnen werden, in vielen Robotikanwendungen eingesetzt.

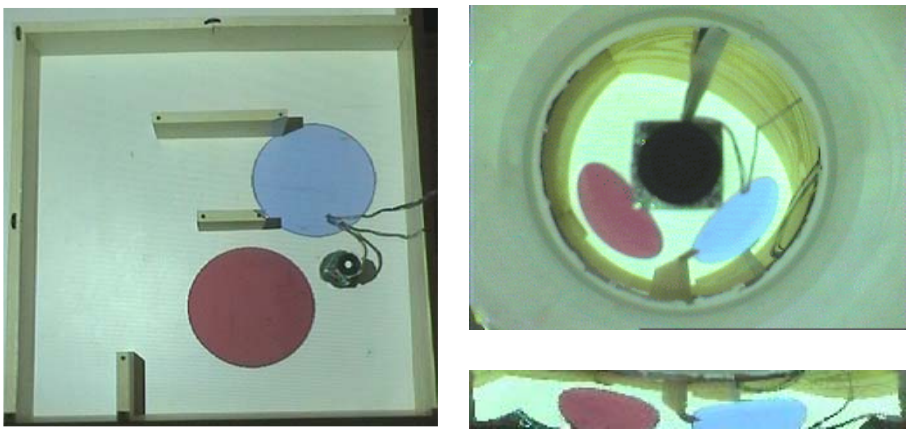

Abbildung 7.4: *Draufsicht auf eine Szene mit Khepera (links), zugehörigem Omni- (rechts oben) und Panoramabild (rechts unten)*

Ein wichtiger Grund für die Bevorzugung visueller Informationen liegt in der Augendominanz beim Menschen und in der sehr regen und äußerst umfangreichen Forschung zur Bild- und Farbverarbeitung. Dies demonstriert ein unmittelbares Forschungsinteresse, das sich auch in der Robotik zunehmend durchsetzt, da aus dem visuellen Datenstrom Entfernungs-, Bewegungs-, Farb- und Texturinformationen gewonnen werden können.

Abbildung 7.4 veranschaulicht eine Szene beim Fahren des Roboters mit einer Aufnahme aus der Vogelperspektive, die dem Roboter so allerdings nicht zur Verfügung stand, sondern nur zur Orientierung des Betrachters dienen soll, sowie dem Omnibild (Abb. 7.4 rechts oben), das der Roboter in dieser Szene aufnahm, und dem daraus entstandenen Panoramabild (Abb. 7.4 rechts unten).

7.2.3.1 Bildvorverarbeitung

Das Omnibild wird zunächst konzentrisch ausgeschnitten, der äußere Radius ist durch den Rand des Parabolspiegels und der innere Rand durch den Khepera selber gegeben. Die resultierende Aufnahme wurde durch eine Polar-Transformation in das Panoramabild überführt. Eigentlich ist diese Überführung in ein Panoramabild nicht zwingend notwendig, die folgende Bildbearbeitung könnte ebenso gut auf dem ausgeschnittenen Omnibild erfolgen. Jedoch ist das Panoramabild und die darauf basierenden Ergebnisse der folgenden Bearbeitung sehr viel einfacher für den Beobachter verständlich und enthalten nachwievor dieselbe Information. Damit sind auch die Ergebnisse der weiteren Bearbeitung besser nachvollziehbar und nicht zuletzt visuell überprüfbar. Weiterhin beschleunigt die entstandene rechteckige Form die folgende Bearbeitung auch. Zu beachten ist jedoch immer, daß die seitlichen Ränder des Panoramabildes eigentlich zu einem Ring schließbar sind und somit der linke und der rechte Rand des Bildes Informationen enthalten, die **hinter** dem Roboter liegen, während die Informationen in der Mitte des Bildes **vor** dem Roboter zu finden sind.

Zur Merkmalsextraktion wurden mehrere Wichtungsfunktionen in Form zweidimensionaler Gaussfunktionen so im Panoramabild plaziert, daß sie jeweils grob den Abstand und die Richtung der enthaltenen Information erhalten (siehe Abb. 7.5).

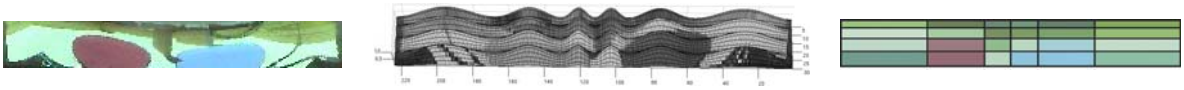

Abbildung 7.5: ***Merkmalsextraktion mit gezielt plazierten Gaussfunktionen***. *Im entstandenen Panoramabild (links) wurden 2D-Gaussfunktionen so plaziert (Mitte), daß rezeptive Felder entstanden (rechts), die neben gemittelten Farb- auch Abstands- und Richtungsinformationen enthielten. Die Unterteilungen sollen die gewählten Abstände nur veranschaulichen.*

So ergeben sich rezeptive Felder, die ähnlich den in [Connell and Mahadevan, 1993b] verwendeten sind. Enthalten die rezeptiven Felder zum Beispiel eine bestimmte Farbe, die für einen Agenten handlungsrelevant ist, lassen sich aus Abstand und Richtung einfach sinnvolle Aktionen ableiten. Allerdings werden diese Aktionen dem Agenten weder mitgeteilt noch explizit berechnet, sodaß er sie selbst ermitteln muß. Dabei wurden die Abstände im Panoramabild in horizontaler Richtung so gewählt, daß sie sowohl symmetrisch zur Vorwärtsrichtung des Roboters sind, als auch eine höhere Auflösung nach vorne als an den Seiten oder nach hinten ergeben. Dies korrespondiert zum gewählten Aktionsspektrum und sollte für den Roboter bevorzugt Bewegungen nach vorne ergeben. In vertikaler Richtung wurden die Abstände zwischen den entstehenden Entfernungsbändern im Panoramabild so gewählt, daß sie in oberen Bildabschnitten am geringsten sind. Diese Teile des Panoramabildes enthalten Informationen über Teile der Umwelt, die am weitesten vom Roboter entfernt liegen. Dagegen wurden am unteren Bildrand, mit Bildinhalten sehr nah am Roboter, die Abstände größer gewählt. Diese Einteilung wirkt dem Effekt des Parabolspiegels der Omnikamera entgegen, sehr weit entferntes nur mit sehr geringer Auflösung widerzugeben. Da am unteren und den äußeren Bildrändern größere Bildteile in ein rezeptives Feld eingehen als am oberen oder den mittleren, alle rezeptiven Felder aber gleiche Bedeutung für die Agenten haben, verstärkt sich so der Einfluß weit entfernter Bildinhalte und solcher, die in Vorwärtsrichtung liegen.

Mit diesen Wichtungsfunktionen wird eine mittlere Farbe für das jeweilige rezeptive Feld ermittelt, indem alle Pixel des Bildes, gewichtet mit dem Wert der Gaussfunktion an der Pixelposition, summiert werden. Dadurch enthalten die rezeptiven Felder eine mittlere Farbe aus der lokalen Umgebung des Mittelpunktes der Gaussfunktion. Durch die Einbeziehung des Gesamtbildes wandern bei den Bewegungen des Roboters Farbflecken, wie sie für die Agenten durch den Beamer in das Szenario projiziert werden, nicht abrupt durch die rezeptiven Felder.

Die rezeptiven Felder bewirken so eine Glättung der Bewegung der Farbaktivierungen durch die rezeptiven Felder und verringern Probleme, die aus Diskrepanzen zwischen der Auflösung der sensorischen Information und jener der verfügbaren Aktionen entstehen.

Die rezeptiven Felder enthalten auf diese Weise Merkmale, die mit weit weniger Daten als das Ausgangsbild dessen Farb-, Abstands- und Richtungsinformationen kodieren.

7.2.3.2 Farbverarbeitung

Die durch räumliche Mittelung entstandenen mittleren Farben für die rezeptiven Felder werden nun mit der HSI-Transformation [Gonzalez and Woods, 1993] in einen Farbraum gebracht, der im Ergebnis einer Untersuchung mehrerer möglicher Farbräume aufgrund verschiedener Vorzüge gewählt wurde (siehe Abschnitt 7.2.3.3) und der eine weitere Reduktion der Datenmenge auf nur einen der drei Kanäle, den Hue-Kanal, gestattet.

In diesem Kanal, der den Farbton repräsentiert, wurden wiederum eindimensionale Gaussfunktionen zur Merkmalsextraktion für die einzelnen Agenten eingesetzt. Sie dienen der Ermittlung einer Aktivierung für jedes Objekt, aus der der jeweilige Reward abgeleitet wird. Die Fitwerte der jeweiligen Gaussfunktion in jedem der rezeptiven Felder bilden andererseits auch zumindest einen Teil des Inputvektors des entsprechenden Agenten. Damit wird dem Agenten genau der für ihn relevante Teil des Farbraumes zur Verfügung gestellt. Die Information über diese Auswahl wurde ihm als dem Experten für das mit der Farbe verbundene Verhalten vom Designer vermittelt und braucht nicht mehr gelernt werden. Anstelle solcher Festlegungen hätte auch ein überwachtes Lernverfahren zum Einsatz kommen können, das lokale Kernel an die entsprechenden Stellen des H-Kanals setzt. Durchführbar wäre eine solche Verfahrensweise zum Beispiel mit einem überwachten Neural Gas oder einem Netz mit Radialen BasisFunktionen (RBF), dem Trainingspaare aus projizierten Farben und einer Farbcodierung (z.B. ein Index) präsentiert werden. Da aber die notwendigen Daten zur Verfügung standen, und der Fokus der Arbeit auf den Verhalten und Koordinationen liegen sollte, wurde auf diesen Trainingsprozeß und die entsprechenden Berechnungen zur Vorverarbeitung verzichtet und mit den Daten eine feststehende Farbinterpretation benutzt.

Eine solche selektive Gaussfunktion existiert für die Farben jedes der in das Szenario projizierten Objekte sowie der Hindernisse und des Untergrundes. Ihr Mittelpunkt stimmt mit dem zu erkennenden Farbwert im Hue-Kanal überein und ihre Varianz wurde zur Trennung der unterschiedlichen Farben bei gleichzeitiger ausreichender Erkennung der Zielfarbe mit $\sigma = 0.03$ für die Beamerfarben und $\sigma = 0.05$ für die Farben des Holzes und des Untergrundes im Labyrinth festgelegt. Abbildung 7.6 zeigt die aus der Bild- und Farbverarbeitung entstandenen Merkmale für die Farben des Holzes und der Objekte.

So entstehen für jeden Agenten Merkmale, die die relevante Farbinformation mit

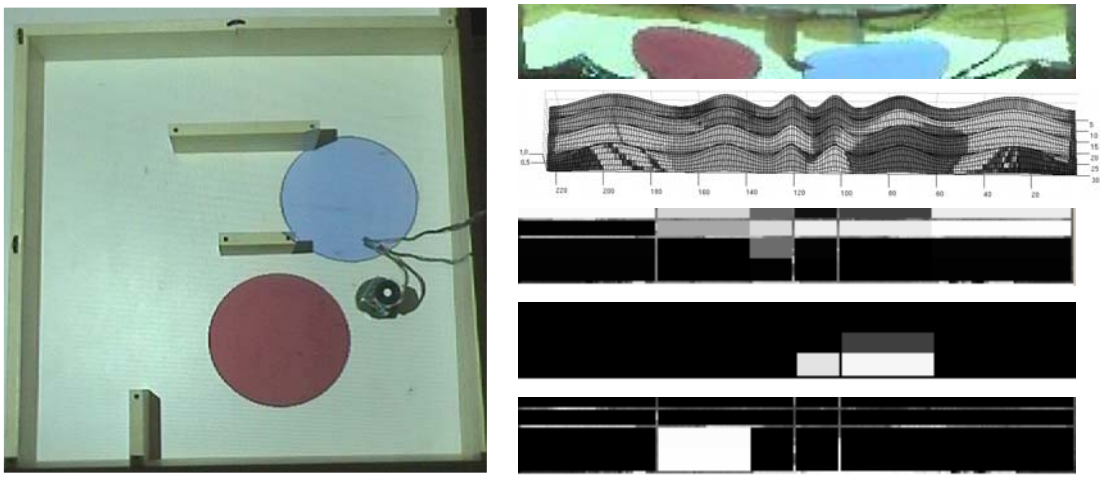

Abbildung 7.6: ***Gesamte Bild- und Farbverarbeitung.*** *In der Szene links entstehen aus dem Panoramabild (rechts oben) mit Verarbeitung durch die rezeptiven Felder (2. rechts) visuelle Merkmale, die (rechts, von unten) auf den dunkelgrauen (roten) Farbfleck (Gefahrenzone), den hellgrauen (blauen) Farbfleck (Futterquelle) und das Holz (Hindernisse) reagieren. Im Vergleich zwischen Gefahrenzone und Futterquelle wird auch die Erhaltung der Richtungsinformation, und zwischen Tränke und Holz der Erhalt der Abstandsinformation deutlich.*

Richtungs- und Abstandsinformationen enthalten. Einem monolithischen Agenten, der mehrere durch Farbobjekte verkörperte Aufgaben erfüllt, werden entsprechend viele Merkmalsvektoren bereitgestellt. Jedoch kann der entstehende größere Merkmalsraum beim Lernen den Vorteil der einzeln präsentierten relevanten Farben mehr als wett machen, und für die Einzelagenten können Informationen bezüglich der anderen Farben für ihre Pfadplanung wichtig sein. Die Ergebnisse einer entsprechenden Untersuchung werden nach Einführung der Agenten, Testmethoden und Leistungsmaße in Abschnitt 7.6.3 vorgestellt.

7.2.3.3 Auswahl des Farbraumes und der Farben

Zur Merkmalsextraktion für die Agenten, deren Eingangssignale auf unterschiedlichen Farben beruhen, mußte ein Farbraum gefunden werden, der die folgenden Anforderungen möglichst gut erfüllt:

- gute Differenzierbarkeit zwischen den Farben des beleuchteten und unbeleuchteten Holzes der Wände des Labyrinths einerseits und des Untergrundes andererseits
- gute Ausnutzung des Dynamikbereiches der Dimensionen, in denen sich die Farbinformation konzentriert.
- möglichst hohe Konzentration der Farbinformation in wenigen Dimensionen

Dazu wurde zunächst die Kamera des Khepera modifiziert, indem der eingebaute automatische Farbausgleich ausgeschaltet und die von der Kamera aufgenommene Farbtemperatur auf $4600K$ festgelegt wurde. Dies war notwendig, da der automatische Weißabgleich versucht, über mehrere Bilder ein neutrales Graubild herzustellen. Dazu werden die Empfindlichkeiten der Kamerasensoren für die verschiedenen Farbanteile geändert. Füllt eine

Farbe die Bilder über mehrere Aufnahmen großteilig aus, z.B. wenn sich der Roboter auf einem projizierten Objekt aufhält, so ändern sich die Anteile der Farbkanäle abhängig vom Bildinhalt, wodurch eine gleichbleibende Farbwahrnehmung nicht gewährleistet ist. Im Ergebnis änderte sich durch den Farbabgleich die von der Roboterkamera aufgenommene Farbe, auch wenn sich der Roboter nicht bewegt hatte und die Projektion unverändert war.

Um einen Vergleich verschiedener Farbräume zu erhalten, wurde der Roboterkamera das gesamte verfügbare Spektrum der Beamerfarben präsentiert und die von ihr aufgenommenen Farben in verschiedene Farbräume transformiert.

Der Khepera mit seiner Omnikamera wurde unbeweglich in einem Farbkreis positioniert, der einen Großteil des Kamerabildes ausfüllte, um Rauscheffekte durch Mittelung gering zu halten. In diesen Farbkreis wurden mit dem Beamer in zufälliger Reihenfolge Farben projiziert, die sich aus einer ebenfalls zufälligen Zusammenstellung der 256^3 möglichen Werte des RGB-Farbraumes des Beamers ergaben. Das Kamerabild wurde auf den relevanten Ausschnitt beschnitten und die nun einheitliche Farbe des Restbildes durch Mittelung berechnet. Das resultierende, aufgenommene RGB-Triplet wurde den verschiedenen Farbraumtransformationen unterzogen und die hieraus entstandenen Daten jeweils einem Neuralen Gas (siehe Abschnitt 4.9) zur optimalen Clusterung übergeben.

Im Neuralen Gas ist die Konzentration der Farbinformation sowie die Ausnutzung des Dynamikbereiches unmittelbar ablesbar. Untersucht wurden die Farbräume RGB, der physiologische Farbraum nach Pomierski [Pomierski, 1996], LAB und schließlich HSI [Gonzalez and Woods, 1993], da diese in Robotikexperimenten oft eingesetzt werden (Abb. 7.7).

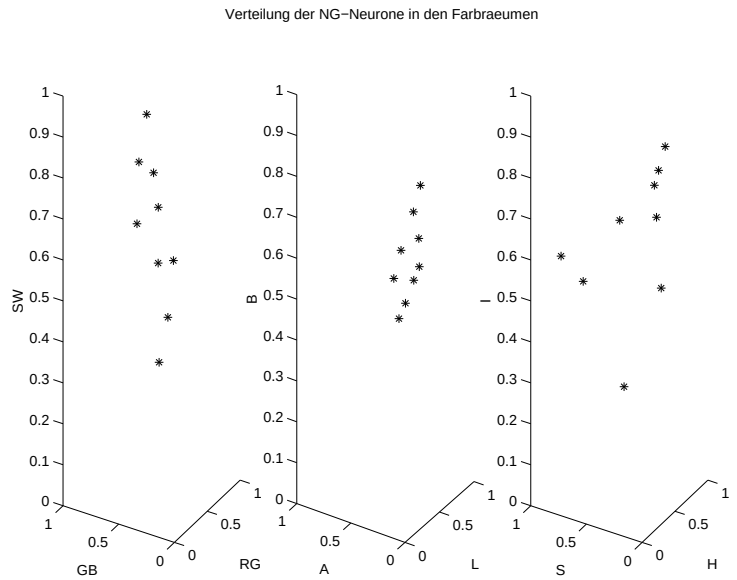

Abbildung 7.7: ***Vergleich der Farbräume.*** *Die Farbeindrücke der Roboterkamera aus dem gesamten Spektrum der Beamerfarben wurden in den physiologischen Farbraum nach* Pomierski *(links), LAB (Mitte) und HSI (rechts) transformiert und durch NG geclustert. Dargestellt ist die resultierende Lage der Gewichtsvektoren der Knoten des NG. HSI wurde letztlich zur Farbverarbeitung gewählt (siehe Text zur Begründung).*

Im Ergebnis dieser Untersuchung wurde der HSI-Farbraum gewählt, da er folgende Vorteile aufwies:

- größter Abstand zwischen Untergrund und Holz,
- beste absolute Ausnutzung des Dynamikbereiches (Differenz zwischen Maximum und Minimum, in Dimension Hue),
- beste statistische Ausnutzung (Varianz der Einträge in der Dimension Hue).
- einfache Unterscheidung von Farben durch Schwellen in der Dimension Hue
- relativ robust gegenüber Beleuchtungsvariationen

Dieser Farbraum ist an das menschliche Farbempfinden angelehnt, was auch die Überprüfung der Ergebnisse der Farbverarbeitung anhand des sichtbaren Szenarios, und die Interpretation des Verhaltens der Agenten erleichtert. In diesem Farbraum werden die Farbtöne auf einen Kreis abgebildet, der bei $H = 0.0$ für Rot beginnt und bei $H = 1.0$, ebenfalls für Rot, endet, bzw. eigentlich neu beginnt.

Anhand der aufgenommenen, verfügbaren RGB-HSI-Paare konnten nun Farben für die einzelnen umzusetzenden Objekte des Szenarios festgelegt werden, die einerseits möglichst weit auseinander liegen und sich andererseits nicht mit den Farben für das Holz der Wände und des Untergrundes überdecken.

In einer weiteren Untersuchung wurden die H-Werte und die Varianzen der Farben für beleuchtetes und unbeleuchtetes Holz sowie den Untergrund ermittelt und festgelegt, daß der Beamer aufgrund der höheren Erkennungssicherheit in den Teilen der Projektion, die nicht von Farbflecken eingenommen werden, weisses Licht projizieren soll. Die Festlegung für dieVerteilung der Farben in der Hue-Dimension wurde in Abb. 7.8 visualisiert.

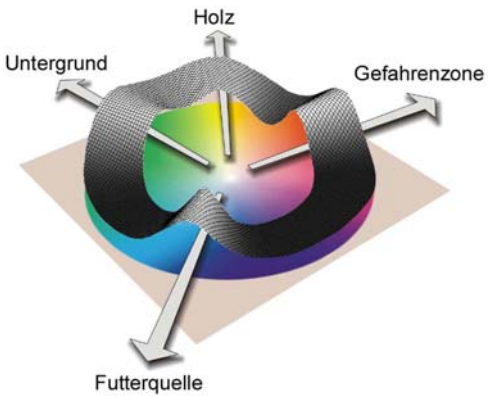

Abbildung 7.8: ***Festlegung der Objektfarben.*** *Darstellung der Lage der Farben entsprechend Tab. 7.1 für die wichtigsten zu unterscheidenden Objekte Holz (Hindernisse), Untergrund des Labyrinths, Futterquelle (positiver Reward) und Gefahrenzone (negativer Reward) im toroidalen Hue-Kanal des HSI-Farbraumes.*

Die nicht von Holz und Untergrund eingenommenen Teile des Hue-Kanals wurden unter den Farben für die Objekte aufgeteilt. Die RGB-Triplets, die den so enstandenen

Mittelwerten der Objektfarben entsprechen und vom Beamer projiziert werden sollen, wurden unter impliziter Berücksichtigung der Farbtransferfunktionen des Beamers und der Kamera direkt aus den aufgezeichneten zusammengehörigen RGB- und HSI-Triplets ermittelt. Die Festlegungen sind in Tab. 7.1 aufgeführt.

Region	H	S	I	R	G	B
Gefahrenzone	0.95	0.7	0.5	220	40	100
Jäger	0.75	0.5	0.7	255	5	160
Futterquelle	0.55	0.5	0.7	15	150	250
Wild	0.1	0.5	0.5	200	90	20

Tabelle 7.1: ***HSI-Triplets und deren RGB-Entsprechungen für die Objekte****. Die Farben für die verschiedenen Objekte im Szenario wurden unter Berücksichtigung der Beamer- und Kameraverfälschung festgelegt.*

Die Übereinstimmung der beabsichtigten mit den vom Khepera tatsächlich erkannten H-Werten wurde in einer weiteren Überprüfung gesichert. Die Ergebnisse sind in Tab. 7.2 dargestellt. Für diese Untersuchung wurde der Khepera wie vorher unbeweglich in einem grossen Farbkreis positioniert und die oben aufgeführten Farben vom Beamer wiederholt projiziert. Aus dem geeignet beschnittenen Bild wurde wiederum die mittlere Farbe berechnet und über mehrere Versuche gemittelt.

Region	μ (H)	σ (H)
Gefahrenzone	0.9676	0.0020
Jäger	0.7497	0.0017
Futterquelle	0.5683	0.0028
Wild	0.1057	0.0113
Holz (beleuchtet)	0.2351	0.0005
Holz (unbeleuchtet)	0.2416	0.0034
Boden	0.4063	0.0025

Tabelle 7.2: ***Mittelwert μ und Standardabweichung des H-Wertes σ der durch Khepera erkannten Farben.*** *Die festgelegten, projizierten Farben wurden sicher erkannt.*

Da der gleichzeitige Einsatz der statischen und dynamischen Quellen gleichen Rewards (*Futterquelle* und *Wild* bzw. *Gefahrenzone* und *Jäger*) nicht untersucht wurde, zeigt Abb. 7.8 nur die beiden Farben für die *Futterquelle* und die *Gefahrenzone*.

7.3 Umsetzung des Szenarios als Simulation

Zur Realisierung zeitintensiver Trainingsläufe sowie als Testbett für methodische Fragen und Software wurde eine Simulation des Szenarios geschaffen, die den Khepera mit seiner Fortbewegung und seiner Sensorik, sowie die Umwelt modelliert. Ausgangspunkt war ein vorhandener Simulator für den Khepera, der um die Bild- und Farbverarbeitung mit der Omnikamera erweitert wurde.

Wie im realen Labyrinth dreht sich der Roboter zuerst um den angegebenen Winkel, um danach mit der gewünschten Geschwindigkeit zu fahren. Würde diese Bewegung ihn in eine Kollision mit einem Hindernis bringen, dann wird die Bewegung nicht ausgeführt. Nach Ausführung der Bewegung werden die Meßwerte der verschiedenen Sensoren berechnet.

In der Simulation werden die Hindernisse als Vierecke aus senkrecht stehenden Wänden variabler Höhe und Farbe dargestellt, die jeweils parallel zu den Außenwänden verlaufen. Weiterhin kann der Boden mit beliebig großen Farbflecken, allerdings nur viereckigen, versehen werden. Die Angaben über Außenwände, Hindernisse und Farbflecken werden in einer einfach zu editierenden Datei gehalten, die die Arbeit mit verschiedenen Umwelten ermöglicht. Abb. 7.9 zeigt den simulierten Roboter in seiner Umwelt mit den Ergebnissen der sensorischen Vorverarbeitung in einer ähnlichen Situation wie der reale in Abb. 7.6.

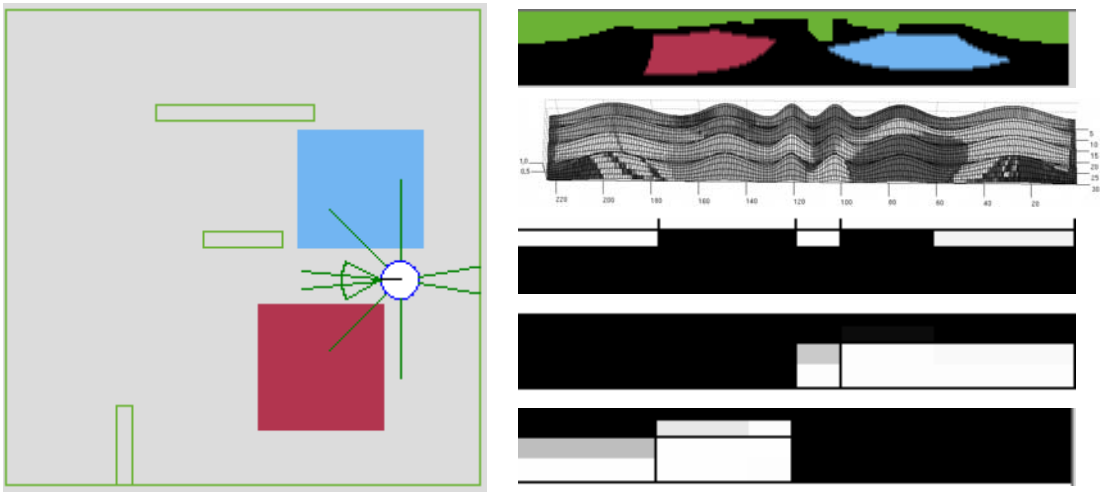

Abbildung 7.9: *Der simulierte Roboter*[2] *in ähnlicher Situation wie der reale in Abb. 7.6. Die rezeptiven Felder (rechts unten) reagieren (von unten) auf den dunkelgrauen (roten) Farbfleck (Gefahrenzone), den hellgrauen (blauen) Farbfleck (Futterquelle) und das Holz (Hindernisse).*

Durch die senkrechten Wände können die Meßwerte der Infrarotsensoren relativ einfach zu allen Hindernissen ermittelt werden.

In der Simulation können Kameras mit beliebiger Sensorenanzahl (Anzahl der CCD-Sensoren) in rechteckiger Anordnung, mit variablen Öffnungswinkeln in X- und Y-Richtung sowie einstellbarer Höhe und Neigungswinkel zum Boden verwendet werden. Dies ermöglicht auch eine Simulation der Omnikamera des realen Khepera.

Mit dem Verfahren des Raycasting wird ermittelt, wo ein Strahl, der aus jedem der simulierten Sensoren der Kamera fällt, auftreffen würde. Aus der Farbe der Wand oder des Bodens an der Auftreffstelle ergibt sich der Farbeindruck für diesen Sensor.

Daraus ergibt sich ein Panoramabild (siehe Abb. 7.9 rechts oben), das dem aus dem Omnibild des realen Khepera gewonnenen ähnelt. Mit denselben Methoden wie für das reale Bild, werden aus dem Panoramabild über die Einteilung in ungleichmäßige rezeptive Felder mit zweidimensionalen Gaussfunktionen lokale mittlere Farben ermittelt. Mit farblichen Gaussfunktionen mit denselben Parametern wie für das reale System wird dann ein Vektor von Aktivierungen der agentenspezifischen rezeptiven Felder erzeugt.

In der Simulation wird kein Rauschen in den Sensoren nachempfunden, desgleichen ignoriert das Raycasting Spiegelungen und Schattenwürfe sowie Licht außerhalb der angegebenen Außenwände.

Bis auf diese Unterschiede liefert die Simulation dieselben Merkmale wie der reale Khepera, sodaß dieselben Agenten wie für den realen Roboter zum Einsatz kamen.

7.4 Umsetzung der Einzelagenten

Alle Agenten bestanden aus einem Clusterer mit nachgeschalteter Approximationsschicht zum Lernen der Zustandsaktionswerte. Die Agenten *Bump*, *Hunger*, *Flucht* wurden umgesetzt. Für die Agenten mit dynamischem Jäger oder Wild wurden dieselben Strukturen und Eingangsinformationen benutzt, wie für die Agenten mit entsprechenden statischen Objekten. Nur ihre Trainingsszenarien und optimalen Policies unterscheiden sich.

Bump-Agent Die Aufgabe des *Bump*-Agenten (Abb.7.10) bestand darin, eine Wandvermeidung zu realisieren und im Freiraum, möglichst schnell und gerade zu fahren.

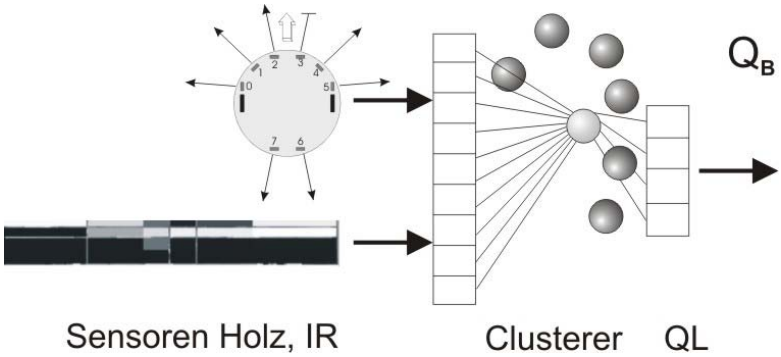

Abbildung 7.10: *Schematische Darstellung des* Bump-*Agenten mit Aktivierung der rezeptiven Felder für die Hindernisse in der Situation in Abb. 7.6 und den symbolisch dargestellten Werten der IR-Sensoren, die ein Eingangsmuster bilden. Das Muster wird durch den Clusterer einem Zustand zugeordnet, der dafür Zustands-Aktionswerte lernt.*

Dazu erhielt er die Werte der acht IR-Sensoren und die Aktivierung der rezeptiven Felder mit Farbspezialisierung auf Holz. Ein Eingangsvektor bestand insgesamt aus 32 Elementen (8 IR und $6 \cdot 4$ visuell). Das Reward-Signal für*Bump* ergab sich sowohl aus den Werten der IR-Sensoren (nah an den Hindernissen bzw. Kollision) als auch der untersten Reihe der rezeptiven Felder, die Hindernisse in unmittelbarer Nähe des Roboters wahrnehmen. Hinzu kam ein Reward-Signal, das aus dem Bewegungskommando abgeleitet war. In Gl. 7.1 steht $|v|$ für die Fahrgeschwindigkeit des Khepera und ϕ für den Lenkwinkel in Grad. Die Distanz d wurde aus den IR- und visuellen Informationen ermittelt.

$$r(Bump) = \begin{cases} -8.0, & \text{wenn } d < d_{min} \\ |v| * 0.25 - |(0.025 * \phi)| & sonst \end{cases} \tag{7.1}$$

Hunger-Agent Die Aufgabe des *Hunger*-Agenten (Abb. 7.11) bestand darin, den blauen Farbfleck (*Futterquelle*) zu finden und darauf zu bleiben. Da die Aktion 'Stehenbleiben' ausgeschlossen war, mußte der Agent andere geeignete Strategien finden. Er erhielt die Aktivierung der rezeptiven Felder mit Farbspezialisierung auf blau, sodaß der Eingangsvektor aus 24 Elementen bestand. Sein Reward-Signal (Gl. 7.2) war die Summe der Aktivierung der untersten Reihe der rezeptiven Felder, also dann am höchsten, wenn der Roboter in der Mitte des blauen Farbflecks stand, und 0.0, wenn er den blauen Fleck verließ.

$$r(Hunger) = \sum_{i=1}^{6} RF_{1i}(blau) \tag{7.2}$$

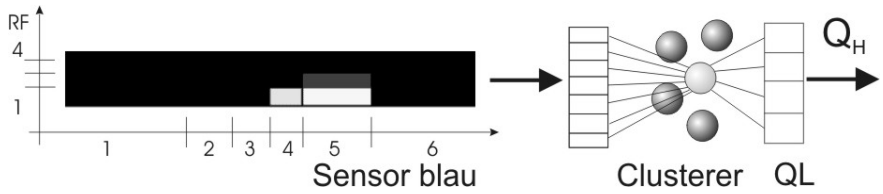

Abbildung 7.11: *Schematische Darstellung des* Hunger-*Agenten mit Aktivierung der rezeptiven Felder für die Futterquelle in der Situation in Abb. 7.6*

Der *Hunger*-Agent wurde auch mit einem dynamischen Objekt, dem dynamischen *Wild*, getestet. Da die Anforderungen an den Agenten grob betrachtet gleich bleiben und qualitativ keine erheblichen Änderungen des Verhaltens im Vergleich zur statischen *Futterquelle* erwarten ließen, wurden die Koordinationen nicht mit einem dynamischen *Wild* getestet. Es wurden keine Szenarien untersucht, in denen sowohl *Futterquelle* als auch *Wild* eingesetzt wurden. Der trainierte *Hunger*-Agent wurde als Experte in den verschiedenen WL-Varianten benutzt.

Flucht-Agent Die Aufgabe des *Flucht*-Agenten (Abb. 7.12) bestand darin, den roten Fleck (*Gefahrenzone*) zu vermeiden. Dazu erhielt er die Aktivierung der rezeptiven Felder mit Farbspezialisierung auf rot (24 Elemente) und der Reward wurde aus der negativen Summe der Aktivierungen der untersten rezeptiven Felder bestimmt (Gl. 7.3).

$$r(Flucht) = -\sum_{i=1}^{6} RF_{1i}(rot) \tag{7.3}$$

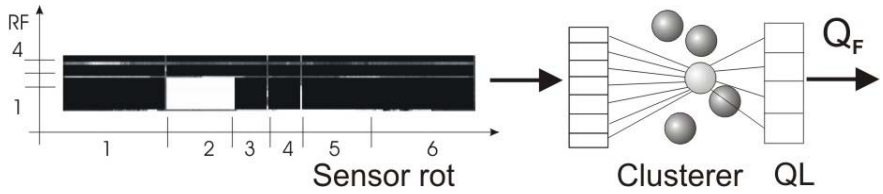

Abbildung 7.12: *Schematische Darstellung des Flucht-Agenten mit Aktivierung der rezeptiven Felder für die Gefahrenzone in der Situation in Abb. 7.6*

Auch der *Flucht*-Agent wurde darüberhinaus mit einem dynamischen Objekt, dem dynamischen *Jäger*, angelernt. Dabei ändern sich die Anforderungen an den Agenten drastisch, wie in Abschnitt 6.1.1 erläutert wurde. Die Policy muß von einer zu vermeidenden Aktion und mehreren ungefähr gleich guten anderen Aktionen auf mehrere zu vermeidende Aktionen umgestellt werden. Insgesamt wurde der Agent jeweils mit der statischen *Gefahrenzone* oder dem dynamischen *Jäger* trainiert und einzeln getestet. Wie bei *Hunger*, wurden für *Flucht* keine Szenarien untersucht, in denen sowohl *Gefahrenzone* als auch der *Jäger* eingesetzt wurden. Jedoch diente auch der trainierte *Flucht*-Agent als Experte in den verschiedenen Varianten des W-Lernens.

Monolith Die Aufgabe des Monolithen (Abb. 7.13) setzt sich aus den Teilaufgaben der Experten zusammen: Wände und rote Flecken vermeiden, blaue Flecken suchen, ansonsten möglichst schnell und gerade fahren. Dazu erhielt er, jeweils als einzelne Vektoren, die zu einem langen Eingangsvektor (80 Elemente) verkettet wurden, die Aktivierung der IR-Sensoren, sowie die Aktivierung der rezeptiven Felder für Holz, blau und rot. Mit allen, sich nun ergebenden Verbindungen wächst der Eingangsraum für den Monolithen stark an. Sein Reward-Signal wurde als Summe der Rewards der Experten berechnet.

$$r(Monolith) = r(Bump) + r(Hunger) + r(Flucht) \tag{7.4}$$

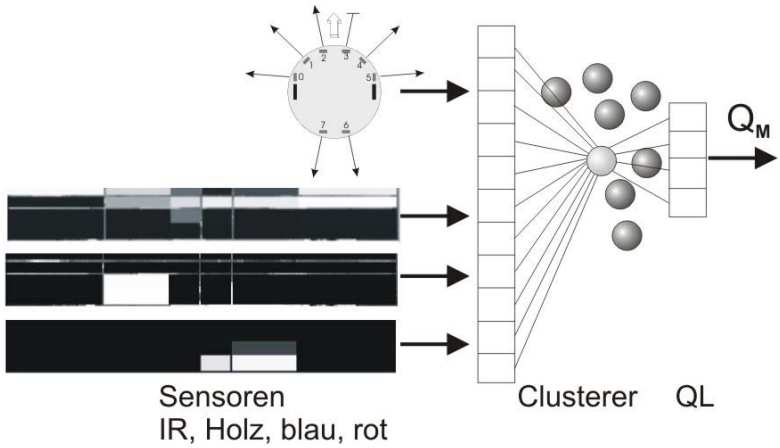

Abbildung 7.13: *Schematische Darstellung des Monolithen mit den Aktivierungen der rezeptiven Felder für Holz, rot und blau sowie den Messungen der IR-Sensoren in einem Vektor.*

Der Monolith wurde im Szenario mit allen statischen Objekten und auch mit der Kombination aus dynamischem *Jäger* und statischer *Futterquelle* untersucht.

W-Lernen Das W-Lernen (WL, Abb. 7.14) wurde mit identischen, völlig austrainierten Agenten für *Bump*, *Hunger* und *Flucht* ausgeführt, wie sie auch einzeln untersucht wurden. Dabei wurden verschiedene Varianten des W-Lernens eingesetzt. Für die wichtigste Variante 'Minimize Worst Happiness' wurden verschiedene Modi untersucht: bei W-Learning mußten die W-Werte gelernt werden, bei 'Negotiated' konnten sie für jede Situation einfach berechnet werden und stützten sich auf die von den Agenten vorgeschlagenen Aktionen, während beim 'Puren' WL alle verfügbaren Q-Werte betrachtet wurden.

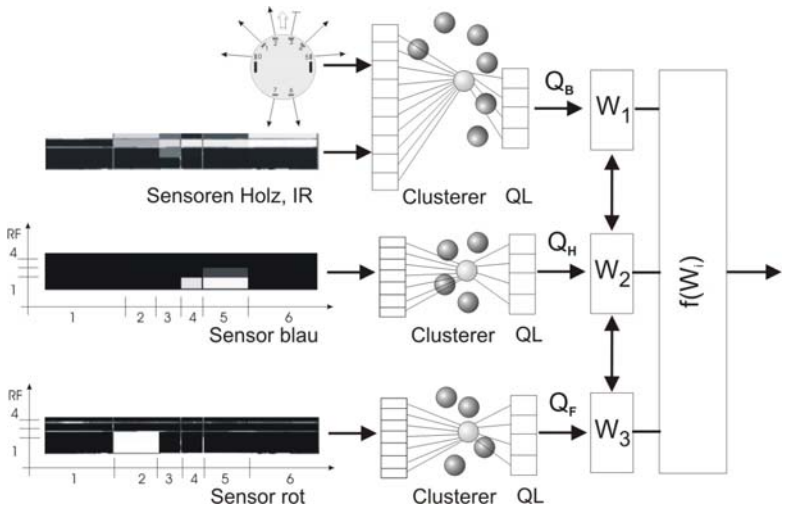

Abbildung 7.14: ***Schematische Darstellung des W-Lernen***. *Aus den Q-Werten der zu den oben dargestellten Agenten identischen Experten werden W-Werte ermittelt und ausgewertet (ganz rechts).*

Die verschiedenen Varianten des WL wurden, wie der Monolith, im Szenario mit allen statischen Objekten und einige ausgewählte Varianten im Szenario mit dynamischem *Jäger* und statischer *Futterquelle* untersucht.

7.5 Durchführung der Experimente

Die Art der Durchführung der Experimente lehnt sich in Ermangelung einer allgemein anerkannten Methodik im RL an die Durchführung von Benchmarktests im überwachten Lernen an. Dort wird anhand eines Trainingsdatensatzes das Netz trainiert und zu bestimmten Punkten während des Trainings mit einem Validierungsdatensatz die erreichte Leistung bestimmt. Es existieren verschiedene Kriterien, wann das Training abgebrochen werden soll, und unterschiedliche Praktiken hinsichtlich der Frage, ob der Validierungsdatensatz Teil des Trainingsdatensatzes sein darf bzw. ob während der Validierung gelernt werden darf.

Streng sind jedoch die Vorschriften bezüglich des Testdatensatzes, mit dem nach Abschluß des Trainings die vom Netz erreichte Leistung ermittelt wird. Erst anhand dieser Performanz wird das Netz bzw. der Algorithmus mit anderen verglichen. Der Testdatensatz darf sich überhaupt nicht mit Trainings- oder Validierungsdatensatz überschneiden, und jegliches Lernen des Algorithmus während des Tests muß unterbunden werden.

Da beim RL die Gefahr des Übertrainings, zumindest mit den hier verwendeten Algorithmen, nicht besteht, kann auf die Validierung während des Trainings verzichtet werden. Zur Messung der Performanz der verschiedenen Agenten bei möglichst reproduzierbaren Ausgangsbedingungen im Sinne eines Testdatensatzes wurden in dem Labyrinth Startpunkte festgelegt, an denen der Roboter per Hand eingesetzt wurde (siehe Abb. 7.15). Dieser Parcour mußte von jedem Agenten mehrmals absolviert werden. Von jedem der Punkte aus mußte jeder der in Abschnitt 7.4 vorgestellten Agenten seine Leistungsfähigkeit bewei-

sen, indem er den Roboter bis zur nächsten Kollision oder über die maximale Schrittanzahl steuerte. Wurden Agenten mit dynamischen Objekten untersucht, so wurden die Objekte nach jeder Kollision sowohl im Training als auch beim Test auf ihre ursprünglichen Positionen zurückgesetzt. Es wurde in einer Untersuchung jeweils maximal eine dynamische Quelle betrachtet. In dieser Untersuchung wurde die entsprechende statische Quelle nicht projiziert. In einer gegebenen Untersuchung gab es also immer genau eine projizierte Quelle positiven und eine Quelle negativen Rewards.

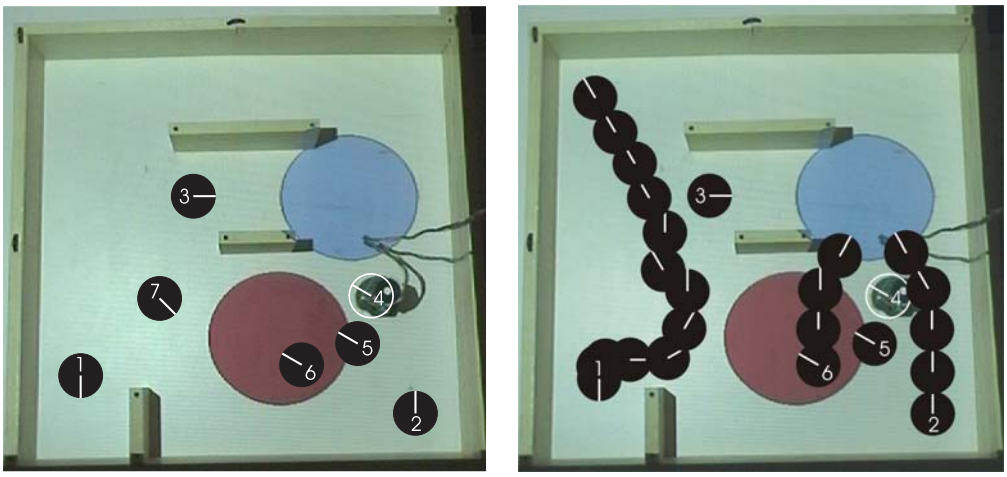

Abbildung 7.15: ***Labyrinth mit festgelegten Einsetzpunkten und Orientierungen für den Roboter.*** *Links: Der hellgraue/rote Fleck stellt die positive Quelle (Futterquelle bzw. Wild) und der dunkelgraue/blaue die negative Quelle (Gefahrenzone oder Jäger) dar. Die schwarzen Kreise kennzeichnen Einsetzpunkte für den Roboter in der durch den weißen Strich markierten Orientierung. Die Zahlen korrespondieren mit den im Text erläuterten Punkten. Rechts sind die Fahrspuren einiger möglicher Verhalten zu sehen, wie im Text erläutert.*

Die Punkte wurden mit der zugehörigen Orientierung des Roboters physisch im Labyrinth gekennzeichnet. Sie wurden so ausgewählt, daß sie jeweils anschaulich die Erfüllung einer Aufgabe durch einen bestimmten Agenten oder aber die gleichzeitige Erfüllung mehrerer Teilaufgaben durch die Koordination der Agenten demonstrieren können. Folgende Fragen motivierten die Auswahl der in Abbildung 7.15 links verwendeten Einsetzpunkte (siehe Abb. 7.15 rechts mit möglichen Anfängen für erfolgreiche Fahrspuren):

1. Sackgasse. Wird der Roboter in die Sackgasse manövriert oder gelingt es dem Agenten, diese unvorteilhafte Situation zu vermeiden? Vorteilhaft wäre ein Umdrehen und Ausnutzung der langen Geraden zum schnellen geraden Fahren.

2. Ecke. Kann der Agent den Roboter aus der Ecke herausbringen und eine Wandverfolgung realisieren, die die Quelle negativen Rewards (*Gefahrenzone* oder *Jäger*) vermeidet und zur Quelle positiven Rewards (*Futterquelle* oder *Wild*) führt, oder führt die Vermeidung der negativen Quelle zur Wandkollision bzw. die Kollisionsvermeidung auf die negative Quelle?

3. Korridor. Wird der Agent den Roboter durch die schwierige Passage manövrieren und auf die positive Quelle gelangen oder sie gleich vermeiden und die lange Ge-

rade vorziehen? Diese Situation muß von der im ersten Einsetzpunkt unterschieden werden, um maximalen Return zu erzielen.

4. Kann der Agent sofort auf die positive Quelle fahren und die negative Quelle und die Wand vor ihm vermeiden?

5. Gelingt es dem Agenten, den Roboter auf kürzestem Weg von der negativen Quelle herunter bzw. auf die positive Quelle zu bringen?

6. Wird der Agent den Roboter schnellstmöglich von der negativen Quelle herunter oder auf die positive Quelle bringen?

7. Kann der Agent die lange Gerade zwischen linker oberer Ecke und unterer Wand zur Fahrt nutzen und dabei der negativen Quelle ausweichen?

Beim Einsatz einer Koordinierung verschiedener Expertenagenten für die unterschiedlichen Teilaufgaben müssen diese sich an den Punkten 2 und 3 (*Bump*, *Hunger*) unterstützen oder an Punkten 2 (*Flucht*, *Bump*) und 6 (*Flucht*, *Hunger*) einigen oder aber einem Agenten die Führung überlassen: Punkt 1 (*Bump*) , 4 (*Hunger*), 5 (*Hunger*) bzw. nur selten eingreifen: Punkt 7 (*Bump*, *Flucht*).

Während der Experimente wurden für jeden Agenten alle Rewards für die Teilaufgaben und der Gesamtreward protokolliert. Das heißt, wenn zum Beispiel *Bump* untersucht wurde, so wurden auch die Rewards protokolliert, die die Agenten *Hunger* und *Flucht*, bei gleichem Verhalten erzielt hätten, sowie der Gesamtreward (Gl. 7.4).

Da der Vergleich Agenten mit sehr unterschiedlichen Zielen betraf, wurde die Leistung jedes Agenten von jedem Einsetzpunkt aus als Durchschnitt für jeden dieser Rewards pro Schritt ermittelt, der dann über alle Einsetzpunkte gemittelt wurde. Dies soll die Leistungsfähigkeit für alle Teilaufgaben und die Gesamtaufgabe des Agenten im gesamten Szenario charakterisieren und ihn so mit den anderen Agenten vergleichbar machen. Der Roboter wurde eingesetzt und der Agent steuerte ihn bis zur nächsten Kollision oder maximal 51 Schritte. Dies wird im folgenden als 'Trial' bezeichnet.

Das eben eingeführte Maß wurde 'Trialreward' genannt und, als Durchschnitt über mehrere Experimente, in diesem und im folgenden Kapitel als Leistungsmaß angegeben. Das zweite Maß betrifft die durch den Agenten erreichte Schrittzahl für einen Trial, wiederum gemittelt über alle Einsetzpunkte und über mehrere Experimente. Die Varianz beider Maße wurde in Klammern nach den Mittelwerten angegeben und kann als Maß für die Zuverlässigkeit des entsprechenden Verhaltens angesehen werden.

7.6 Voruntersuchungen

In den folgenden Abschnitten werden Untersuchungen vorgestellt, die den eigentlichen Leistungsuntersuchungen in Kapitel 8 vorausgingen und Detailfragen der Realisierung klären sollten. Dies betrifft einen umfangreichen Test über geeigneteste Konfiguration aus Clusterer und RL-Algorithmus, die für die Agenten in den weiteren Experimenten eingesetzt werden sollte (Abschnitt 7.6.1). Weiterhin wurde der Trainingsmodus der Agenten

festgelegt, nach einem Vergleich zwischen Agenten, die ausschließlich im realen Labyrinth trainiert wurden, mit solchen, die ausschließlich aus einer Datei trainiert wurden (Vortraining, siehe Abschnitt 7.6.2). Die Datei wurde hierfür zuvor durch einen Khepera aufgezeichnet, dessen Aktionsauswahl komplett zufällig war. Abschließende Voruntersuchungen betrafen die Vorteile, die die Agenten durch die Zuordnung der relevanten Farben erlangten (Abschnitt 7.6.3). Eine Fragestellung, die nur im simulierten Szenario betrachtet wurde, betrifft die Behandlung des Szenarios als MDP mit entsprechend optimaler Navigation durch einen RL-Agenten und wird in Abschnitt 7.6.4 behandelt.

7.6.1 Auswahl der Agentenkonfiguration

Für die Einzelagenten als Experten mußte eine gute Agentenkonfiguration gewählt werden. Dabei wurde die Breite des Spektrums eingeschränkt auf wertefunktionsbasierte RL-Verfahren (siehe Abschnitt 4.2.2). Weiterhin erleichtert die Kombination mit einem Vektorquantisierer die Interpretation des erzielten Verhaltens.

Daher wurden Kombinationen verschiedener Vektorquantisierer (Neural Gas (NG), Growing Neural Gas (GNG), Growing Resonant Adaptive Neural Gas (GRANG)) mit den zwei häufigst eingesetzten RL-Verfahren, *Q-Learning* (Q) und *Sarsa* (S), jeweils mit und ohne Traces ($\lambda = 0$ bzw. $\lambda = 0.8$) untersucht. Die verschiedenen Konfigurationen wurden jeweils als Monolith trainiert und getestet. Als Entwicklungsszenario kam ein Vorläufer des in Abschnitt 7.5 vorgestellten mit 10 Einsetzpunkten (siehe Abb. 7.16) zum Einsatz.

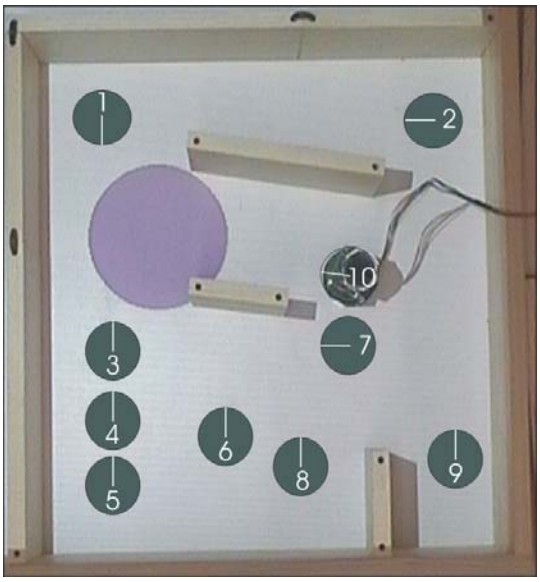

Abbildung 7.16: *Das Entwicklungsszenario mit 10 Einsetzpunkten und der Futterquelle.*

Da das Hauptinteresse im Finden der geeignetesten Agentenkonfiguration für die zentralen Koordinierungsexperimente bestand und bereits Erfahrungen hinsichtlich der Parameter vorhanden waren, wurde auf eine systematische Untersuchung der optimalen Parameter für die einzelnen Konfigurationen verzichtet und Erfahrungswerte verwendet. Die Trainingsparameter wurden für alle Agenten gleich gehalten, soweit sie relevant waren.

Agent	Trialreward (σ)	Schritte (σ)
NGS	1.5 (3.2)	18.3 (6.1)
NGS(0.8)	**1.8** (2.1)	**18.2** (4.6)
NGQ	1.3 (4.6)	18.9 (12.8)
NGQ(0.8)	-0.04 (1.8)	15.7 (7.7)
GNGQ	-0.3 (3.2)	14.4 (5.9)
GNGQ(0.8)	-2.7 (2.1)	12.4 (6.1)
GRANGQ	-0.5 (1.8)	15.4 (4.8)
Zufall	-0.8 (5.4)	11.8 (9.3)

Tabelle 7.3: ***Ergebnisse verschiedener Agentenkonfigurationen im realen Labyrinth.*** *Zur Festlegung einer einheitlichen Konfiguration für die im weiteren einzusetzenden Agenten wurden mehrere Kombinationen unterschiedlicher Clusterer ('NG'=Neural Gas, 'GNG'=Growing Neural Gas, 'GRANG'=Growing Resonant Adaptive Neural Gas) und der beiden RL-Verfahren Q-Lernen ('Q') und* Sarsa *('S') jeweils mit und ohne Traces anhand eines Monolithen mit visuellen und IR-Eingang getestet. 'Trialreward' bezeichnet den erreichten durchschnittlichen Reward pro Schritt, 'Schritte' die Anzahl der pro Einsetzpunkt erreichten Schritte. Aufgrund der insgesamt besten Leistung wurde NGSarsa(0.8) der Vorzug in den folgenden Experimenten gegeben.*

Eine ähnliche Untersuchung wurde bereits in einem Benchmarktest auf dem weit verbreiteten Stabbalanceproblem durchgeführt [Surmeli and Gross, 2000]. Wie dort, zeigte auch hier die Kombination eines einfachen Neural Gas mit dem normalen QL($\lambda = 0$) überraschend gute Ergebnisse. Anders als dort erzielten jedoch auch die Kombinationen des NG mit Sarsa($\lambda = 0$) und Sarsa($\lambda = 0.8$) gute Resultate. Beide übertreffen NGQ hinsichtlich des erreichten Trialrewards und erreichen hinsichtlich der Schrittzahl eine ähnliche Anzahl, jedoch erheblich zuverlässiger. Aufgrund dieser Ergebnisse fiel die Entscheidung für alle Agenten zugunsten der Kombination NG mit Sarsa($\lambda = 0.8$) aus. Sarsa($\lambda = 0.8$) verdrängt in letzter Zeit auch international das QL als Vergleichsmaßstab für neu entwickelte Lernverfahren (z.B. [Peshkin et al., 2000]).

7.6.2 Notwendigkeit eines Vortrainings und Wahl des Trainingsmodus

Ein Zitat aus [Connell and Mahadevan, 1993b] soll die Haltung vieler Autoren zur Frage des Trainingsmodus realer Roboter widergeben:

> To get real robots to work in the real world, they must be trained (at least partially) in the real world.

Roboter, die in einer realen Umwelt funktionieren sollen, müssen auch in ihr trainiert werden. Deshalb wurden Pläne verworfen, Agenten in der Simulation vorzutrainieren.

Andererseits nimmt das Training von RL-Agenten in realer Umwelt lange Zeit in Anspruch, da jede Aktion in jedem Zustand mehrfach ausgeführt werden muß und ihre reale

Ausführung eine Zeit im Sekundenmaßstab erfordert. Als eine Art Kompromiß wurden alle unvorverarbeiteten Meßdaten der IR-Sensoren und der Kamera des Khepera sowie die Aktionen während mehrerer Fahrten aufgezeichnet, die jede 1-2 Stunden andauerten. Während dieser Fahrten wurde die Aktionsauswahl durch eine sehr hohe, konstante Boltzman-Temperatur weitestgehend zufällig gestaltet, um eine neutrale Ausgangsbasis für das Training von Agenten mit verschiedensten Zielen zu gewährleisten. Da die Fahrt im voll ausgebauten Szenario stattfand, sind alle Objekte bzw. Rewardquellen und eine statistisch repräsentative Auswahl aller anzutreffenden Situationen im Datensatz enthalten. Entsprechend können sowohl Clusterer als auch RL-Algorithmen angelernt werden.

Anhand des so entstandenen Datensatzes konnten im folgenden mehrere Agenten in kurzer Zeit trainiert werden. Darüber hinaus ergab sich der Effekt, daß zu vergleichende Agenten tatsächlich anhand genau derselben Trainingsdaten angelernt wurden, was ihre Vergleichbarkeit erhöht und die Forderung nach einer einheitlichen Trainingsdatei, wie in Abschnitt 7.5 diskutiert, realisiert.

Während die Agenten aus dem Datensatz vortrainiert werden, haben sie keine Möglichkeit mehr, selber die Aktionsauswahl zu beeinflussen. Ein solches Vorgehen ähnelt einem überwachten RL, bei dem ein Trainer die auszuführende Aktion vorgibt und der Agent nur noch deren Wert lernen und später bei seiner eigenen Aktionswahl berücksichtigen kann. Die Agenten müssen hier die Aktion bearbeiten, die von dem zufälligen Agenten vorgegeben wurde. Dies sollte vor allem Agenten benachteiligen, die mit Traces ganze Zustandsaktionssequenzen betrachten, also solche mit $\lambda > 0$. Dies trifft insbesondere auf Q-Agenten zu, denn diese setzen zur Benutzung von Traces voraus, daß jeweils die beste Aktion ausgewählt wird. Bei zufälliger Aktionswahl ist das jedoch sehr unwahrscheinlich. Deshalb wurden in den Experimenten Agenten untersucht, die nur aus dem Datensatz ('Off' in Tabelle 7.4), oder nur im realen Labyrinth ('On' in Tabelle 7.4) trainiert wurden, ergänzt um solche, in denen die Agenten mit dem Datensatz vortrainiert wurden und dann im Labyrinth mit einer verbliebenen Zufälligkeit der Aktionsauswahl nachlernen konnten ('Off+On' in Tabelle 7.4).

Agent	Trialreward (σ)	Schritte (σ)
NGS On	-1.0 (0.9)	21.9 (7.2)
NGS Off	-1.1 (1.0)	23.9 (6.1)
NGS Off+On	-1.2 (0.6)	22.9 (3.6)
NGS(0.8) On	-1.2 (0.8)	16.3 (3.3)
NGS(0.8) Off	-3.3 (1.1)	11.2 (3.5)
NGS(0.8) Off+On	**-1.0** (1.1)	**23.7** (4.2)
Zufall	-2.2 (1.0)	14.4 (4.5)

Tabelle 7.4: ***Vergleich der Trainingsmodi***: *Vergleich der Ergebnisse des* Bump-*Agenten mit visuellem und IR-Eingang bezüglich des Trainings nur aus dem Datensatz ('Off'), nur im realen Labyrinth('On'), bzw. mit Vortraining aus dem Datensatz und Nachlernen im realen Labyrinth ('Off+On'). Da Training aus dem Datensatz verschiedene Agentenkonfigurationen unterschiedlich beeinflußt, wurden mehrere Kombinationen für Agenten mit Neuralem Gas ('NG') als Clusterer und* Sarsa *('S') als RL-Algorithmus, jeweils mit und ohne Trace ($\lambda = 0.0$ und $\lambda = 0.8$) getestet.* NGSarsa*(0.8) mit gemischtem Training ('Off+On') wurde der Vorzug gegeben.*

Für diese Untersuchung wurde wieder das Entwicklungsszenario (siehe Abb. 7.16) mit zehn Einsetzpunkten verwendet, den jeder Agent mindestens fünf Mal durchlaufen mußte. Zum Einsatz kam diesmal der *Bump*-Agent mit acht IR-Sensoren und sechs rezeptiven Feldern in drei Entfernungsbändern in der festgelegten Kombination aus einem Neural Gas und einem *Sarsa* mit Boltzman Exploration.

Eine konsistente Interpretation der Ergebnisse dieser Untersuchung fällt überraschend schwer und ergibt auch für die beiden Maße unterschiedliche Resultate. Weder ist der erwartete Effekt für Agenten mit $\lambda > 0$ eindeutig herauszustellen, noch zeigt sich ein klarer Vorteil für einen der Trainingsmodi.

Aufgrund der Ergebnisse sowohl bei der erzielten Anzahl der Schritte des gemischten Trainings und des erzielten durchschnittlichen Rewards pro Schritt gab diese Untersuchung den Ausschlag, für die weiteren Experimente aus dem Datensatz vorzutrainieren und dann im realen Labyrinth nachzulernen. Zum Vortraining wurde ein Datensatz festgelegt, der 300 Trials mit ca. 6000 Schritten enthält. Wegen der zufälligen Aktionswahl wird eine große Zahl verschiedener Situationen präsentiert, was vor allem für die Strukturierung des Clusterers wichtig ist. Aber auch der RL-Algorithmus profitiert vom Testen vieler unterschiedlicher Aktionen in verschiedenen Situationen. Allerdings ist dieses Training wegen der aufgezwungenen Aktionswahl relativ uneffektiv für ihn. Für das Nachtraining, das vorrangig der Konsolidierung der Aktionswerte, besonders für die Algorithmen mit $\lambda > 0$ dient, wurde die Trainingsdauer durch eine geeignet hohe und für alle Agenten gleiche Boltzman-Temperatur festgelegt, die pro Schritt abfällt. Eine Festlegung der Trainingsdauer mit Trials würde die unterschiedlichen Längen der Trials für die verschiedenen Agenten und das Anwachsen dieser Länge während des Trainings vernachlässigen.

Mit der Aufzeichnung aller unverarbeiteten sensorischen Messungen eines sich komplett zufällig bewegenden Khepera in eine Datei ist ein einheitlicher Trainingsdatensatz geschaffen worden. Dieser wurde zum Vortraining aller Agenten genutzt. Da ein RL-Algorithmus aber kritisch von den Erfahrungen abhängt, die er bei Ausnutzung der selbst gelernten Policy macht, muß der Agent auch selbst den Roboter kontrollieren. Dies wurde mit dem anschließenden Nachlernen im realen Labyrinth umgesetzt.

7.6.3 Vergleich der rezeptiven Felder mit und ohne farbspezifische Vorverarbeitung

Die Verwendung der in Abschnitt 7.2.3.2 diskutierten farbsensitiven Gaussfunktionen im H-Kanal des HSI-Farbraumes war motiviert durch eine möglichst aufgabenangepaßte Bild- und Farbverarbeitung, die dem Agenten einen unkomplizierten, robusten Erwerb des entsprechenden Verhaltens ermöglicht.

Jedoch ist dies im gegebenen konkreten Fall nicht zwingend notwendig. Vorstellbar wäre, den Agenten einfach die ermittelten Werte des H-Kanals ohne weitere Verarbeitung durch die Farbgaussfunktionen zur Verfügung zu stellen.

Alle Agenten erhielten in diesem Falle genau dieselben Informationen: die gemittelten Farbtöne der rezeptiven Felder. Für die Expertenagenten sind zwei, gegensätzlich wirkende Effekte vorstellbar. Zum einen muß jeder der Experten zunächst die für seinen Reward relevanten Farbwerte finden. Zum anderen steht jedem Experten durch die Zuordnung der

Farben vom Designer nur Informationen über genau eine, nämlich die für ihn relevante, in der Umwelt vorkommende Farbe zur Verfügung. Damit verliert er jedoch die Farbinformation in den Teilen des Szenarios, in denen diese Farbe nicht auftritt. Deshalb kann er diese Teile nicht unterscheiden und z.B. einen Weg zu den relevanten Farbflecken nicht finden. Er kann die anderen Farben nicht mehr als Landmarken verwenden. Daher ist die Aufgabe für die Experten mit Farbvorverarbeitung zwar einfacher, aber auch schlechter beobachtbar, unter Umständen entsteht sogar ein POMDP.

Für den Monolithen wäre die Größe des Eingangsraumes nunmehr die gleiche wie für jeden einzelnen der Experten, jedoch müßte er die verschiedenen Bedeutungen der Farben im H-Kanal selbst ermitteln. Dies sollte das Training komplizierter gestalten und verlängern, da der Monolith mehrere Quellen desselben, positiven oder negativen Rewards unterscheiden muß, die jedoch zu einem einzigen Gesamtrewardsignal überlagert sind.

Daher wurde anhand der Experten und des Monolithen untersucht, welchen Effekt die vom Designer gratis zur Verfügung gestellte Information über die Bedeutung der verschiedenen Farben hat. Es wurden Agenten verglichen, die bis auf die zur Verfügung gestellte Eingangsinformation völlig identisch waren. Der eine Agent erhielt direkt die ungefilterten Werte des H-Kanals in den rezeptiven Feldern als Eingangsinformation, der andere die Fitwerte der entsprechenden Gaussfunktion auf dem H-Kanal.

Agent	Trialreward (σ)	Schritte (σ)
Monolith mit FarbGauss	3.2 (0.4)	46.1 (3.1)
Monolith ohne FarbGauss	-0.2 (0.2)	43.5 (4.3)
Bump mit	-0.6 (0.7)	41.8 (4.1)
Bump ohne	-0.9 (0.7)	42.3 (2.7)
Durst mit	-1.2 (1.1)	26.2 (4.1)
Durst ohne	-0.2 (0.6)	26.7 (10.1)
Flucht mit	-1.0 (0.9)	19.6 (4.2)
Flucht ohne	-1.1 (0.2)	10.7 (4.0)

Tabelle 7.5: ***Ergebnisse mit und ohne farbspezifische Vorverarbeitung im realen Labyrinth.*** *Der Effekt eines Einsatzes farbspezifischer Gaussfunktionen ('mit FarbGauss') wurde im Vergleich zur Verwendung des ungefilterten Wertes des H-Kanals, also des Farbtons im HSI-Farbraum ('ohne') ermittelt. Hierfür wurde ein Monolith verwendet, da für diesen mit seinen Teilaufgaben Bump, Hunger und Flucht die Aussagekraft der Farbinformation besonders relevant ist. Der Vorteil bei Verwendung der Farbcodierung für den Monolithen ist ersichtlich.*

Dieser Test wurde mit dem Parcour aus Abschnitt 7.5 realisiert. Die Trainingsparameter wurden für alle Agenten gleich denen in den Tests in Kapitel 8 gehalten. Alle Agenten wurden an demselben Datensatz vortrainiert und dann im realen Labyrinth nachgelernt. Bei allen Agenten handelt es sich um NG als Clusterer und $Sarsa(0.8)$ als RL-Verfahren.

Der Leistungsunterschied zwischen dem Monolithen mit Farbcodierung und dem mit den unvorverarbeiteten H-Werten fällt klar zugunsten des ersteren aus (Tab. 7.5). Wie erwartet, führt der Einsatz der Farbcodierung zu einer leichter zu erlernenden Abbildung. Für den Monolithen trifft auch die oben diskutierte POMDP-Problematik nicht zu, da er die Fitwerte aller Farbgaussfunktionen in den rezeptiven Feldern als Eingangsinformation erhält. Im Gegenteil, dies erleichtert seine Aufgabe in einer Hinsicht, da die verschieden-

farbigen Objekte nun in verschiedenen rezeptiven Feldern und damit unterschiedlichen Elementen des Eingangsvektors (verschiedenen Dimensionen des Eingaberaumes) abgebildet waren.

Aufgrund der Clusterung entsteht ihm auch kein größerer Aufwand, drei Sätze der rezeptiven Felder mit unterschiedlichen Farbcodierungen auswerten zu müssen, während der Monolith mit den unvorverarbeiteten H-Werten nur einen Satz der rezeptiven Felder verarbeiten muß.

Der Effekt für die Expertenagenten ist weniger eindeutig, hier zeigt sich der Unterschied zwischen optimalen Policies für Vermeidungs- und Erreichungsaufgaben. Während *Bump* und *Flucht* Vorteil aus der Vorverarbeitung durch die Farbgausse ziehen, die ihnen genau mitteilt, welche Gebiete und folglich, welche Aktionen sie jeweils vermeiden sollten, schneidet *Hunger* mit den Farbgaussen schlechter ab. Dies wird dadurch erklärt, daß ihm die anderen Farben als Orientierung für seine implizite Planung eines Weges zu seinem Zielgebiet fehlen. Eine solche Planung ist für *Bump* und *Flucht* zumindest bei statischen Quellen negativen Rewards nicht notwendig.

7.6.4 Das Szenario als MDP

Für einen systematischen Vergleich von Agenten im entworfenen Szenario wäre ein Maßstab dienlich, der eine Referenz für tatsächlich mögliche Leistungen im gegebenen Szenario darstellt, um die Auswirkungen möglicher unzureichender sensorischer Beschreibung oder der Clusterung abzuschätzen. Ein solcher Maßstab aber ist theoretisch vorgezeichnet durch einen Agenten, der mit einem Algorithmus des RL einen Markow'schen Entscheidungsprozeß, z.B. einer Rasterwelt, löst. Hierfür existieren Beweise zur Optimalität der gefundenen Policies.

Vorstellbare Ansätze zur Erzeugung eines solchen Leistungsstandards sind mit verschiedenen Problemen verbunden. Eine diskutierte Alternative bestand in der Anwendung bekannter klassischer Planungsverfahren wie des Dijkstra-Algorithmus. Dieser basiert auf Graphensuchverfahren, die topologische Informationen nur in den Kantengewichten enthalten können. Es ist jedoch kompliziert, die Geometrie der speziellen Bewegungen des Khepera auf die Nachbarschaften von Knoten in Graphen abzubilden. Weiterhin müßten für eine dem Szenario angemessene Planung die verschiedenen Rewards für die Futterquelle, den Jäger und die Wände ausgemessen und in die Kantengewichte integriert werden. Besonders schwierig dürfte sich das für Rewards erweisen, die für die Fahrweise des Khepera vergeben werden.

Ähnliche Probleme treten auf, wenn die direkte Euklidische Verbindung zwischen Einsetz- und Zielpunkten betrachtet wird. Auch hier müssen die vergebenen Rewards eingearbeitet werden, die Berechnung wird aber auch durch die Hindernisse komplizierter. Generell ist festzustellen, daß die Einbeziehung der Informationen aus dem gegebenen Szenario, die notwendig sind, um vergleichbare Leistungen zu berechnen, zu einer immensen Komplexität und sehr hohem Aufwand führen. Bezieht man sie jedoch nicht ein, erhält man zwar Leistungsmaße, diese sind jedoch nicht mehr vergleichbar mit den Randbedingungen, unter denen die Agenten ihre Leistungen in dem Szenario erzielen.

7.6.4.1 Realisierung mit RL

Um also die Einflüsse von POMDPs aus einer unzureichenden sensorischen Beschreibung sowie aus der Zusammenfassung und Generalisierung von Zuständen durch einen Clusterer auszuschließen, wurden die Experimente in der Simulation mit einer ursprünglichen Implementation des eingesetzten RL-Verfahrens *Sarsa* ausgeführt. Eine solche Implementation bezieht sich auf feste Zustände der RL-Aufgabe, wie sie zum Beispiel durch eine festgelegte Rasterung der möglichen Wertebereiche der Dimensionen des Eingaberaumes erzeugt werden können (Rasterwelt). Mit einer solchen festen Rasterung ergibt sich der Zustand als Index in einer Tabelle, in der alle möglichen Kombinationen der Unterteilungen der Eingabedimensionen verzeichnet sind.

Ein Eingaberaum, der das in den Abschnitten 7.1 und 7.5 eingeführte Szenario als MDP beschreiben sollte, besteht aus der Position und Orientierung des Khepera, wenn im Szenario ausschließlich statische Objekte vorhanden sind, und zusätzlich aus Entfernung und Richtung des dynamischen Objektes, wenn ein solches vorhanden war. Da diese Angaben auch zur Steuerung der dynamischen Objekte notwendig waren, wird ihre Ermittlung zusammen in Anhang 10 erläutert.

Die Position wurde in einem Raster von 25 Einteilungen sowohl in X- als auch in Y-Richtung angegeben. Für ein Szenario mit 60 cm Seitenlänge bedeutet das eine Auflösung von $2,4$ cm. Eine gröbere Rasterung wäre bei einem Roboterdurchmesser von ca. 5 cm nicht mehr hinreichend gewesen, mehr jedoch hätten eine größere Lernaufgabe für den Agenten bedeutet. Die Orientierung wurde in 36 Richtungen, also in Schritten von 10° eingeteilt. Diese Auflösung sollte bei den vorgegebenen Lenkwinkeln von 30° ausreichen. Aus diesen Angaben entsteht die globale Position und Orientierung des Roboters nach Art eines Global Positioning Systems (GPS). Diese Information sollte für das Erlernen der Aufgaben mit statischen Objekten genügen, also für die Hindernisvermeidung, sowie für räumlich invariante Farbflecken unterschiedlicher Bedeutung. In Experimenten mit räumlich veränderlichen Objekten wie einem dynamischen *Jäger*, der den Roboter verfolgt, oder einem vor dem Roboter fliehenden *Wild*, benötigt der Agent zusätzlich Informationen über den Zustand dieser Objekte, wie deren Abstand und Richtung.

7.6.4.2 Abschätzung des Lernaufwandes

An dieser Stelle soll jedoch eine Überschlagsrechnung die Komplexität des Reinforcement Lernen greifbarer machen. Bei einer Implementation als Tabelle hat der Agent mit den Eingangsinformationen X- und Y-Position (je 25 mögliche Werte), Orientierung (36 Werte), Entfernung (4 Werte) und Richtung (6 Werte) $25 * 25 * 36 * 6 * 4 = 540.000$ unterschiedliche Zustände. Bei 9 möglichen Aktionen (3 Geschwindigkeiten in 3 Richtungen) ergibt das 4.860.000 Zustands-Aktions-Paare, die jedes mehrfach durchlaufen werden müssen, um einen Q-Wert zu ermitteln. Bei ca. 10 Durchläufen für jedes Paar müssen also 50 Millionen Lernschritte veranschlagt werden. Ein PC mit einem 1GHz-Prozessor ist mit dieser Simulation über 30 Tage beschäftigt (ca. 15 Zyklen pro Sekunde, mit grafischer Ausgabe, nicht als eigenes Programm), eine Umsetzung im realen Labyrinth ist schlicht unmöglich. Bei Annahme einer linearen Beschleunigung von nur 30 (der Roboter benötigt zwischen 1 und 2 Sekunden für einen Schritt im realen Labyrinth) zwischen Simulation und realem Labyrinth hieße das, der Roboter müßte monatelang ununterbrochen fahren.

7.6.4.3 Experimente und Ergebnisse

Die Experimente wurden zunächst nur in der Simulation ausgeführt, um Hinweise zu notwendigen Auflösungen und Lernzeiten zu erhalten. Dabei wurden in dem Szenario entsprechend Abb. 7.9, das auch für die Experimente in Kapitel 8 verwendet wurde, nur statische Objekte eingesetzt. Untersucht wurde ein Monolith mit *Sarsa*(0.8) als RL-Algorithmus.

Agent	Trialreward (σ)	Schritte (σ)
halbe Auflösung X,Y,ϕ 10 Step/SAP	-9.1 (0.3)	20.3 (4.6)
halbe Auflösung X,Y 10 Step/SAP	-3.3 (0.2)	10.5 (2.5)
halbe Auflösung ϕ 10 Step/SAP	-3.5 (0.3)	28.9 (2.3)
volle Auflösung X,Y,ϕ, 1 Step/SAP	-0.2 (0.9)	35.9 (4.6)
volle Auflösung X,Y,ϕ, 10 Step/SAP	-1.1 (0.6)	30.5 (3.6)
volle Auflösung X,Y,ϕ, 100 Step/SAP	-0.9 (0.6)	38.2 (4.9)
visuell	2.8 (0.5)	51.0 (0.0)
Zufall	-4.0 (3.8)	14.3 (4.6)

Tabelle 7.6: ***Ergebnisse der GPS-Agenten in der Simulation.*** *Die sensorischen Informationen bestanden aus einer globalen Position des Khepera im Labyrinth und seiner Orientierung. Aufgrund der Größe des Zustandsraumes und der entsprechend langen Lernzeiten wurden verschiedene Auflösungen der Position und der Orientierung sowie unterschiedliche Trainingsdauern (Step/SAP) untersucht. 'SAP' = State-Action-Pair, Zustands-Aktionspaar. 'visuell' gibt die Leistungen des Monolithen mit visueller Sensorik an, 'Zufall' die eines sich zufällig verhaltenden Agenten.*

Die Untersuchung hinsichtlich der notwendigen Auflösung ergibt das zu erwartende Ergebnis, daß eine höhere Auflösung bessere Leistungen zur Folge hat. Dabei ist der Einfluß einer Reduzierung der Auflösung der Position größer als der für die Orientierung.

Unerwartet ist jedoch der Effekt, das die Agenten mit steigender Lerndauer schlechtere statt bessere Ergebnisse erzielen. Ähnlich schlecht interpretierbar ist auch die Schrittzahl. Es ist jedoch leicht zu sehen, daß der Monolith mit GPS-Sensorik keine Vorteile im Vergleich zu Monolithen mit der lokalen visuellen Sensorik erreicht (siehe auch Abschnitt 8.1.2 auf Seite 139 und Tabelle 8.2) .

Stellvertretend für die untersuchten Agenten wird das Verhalten des Monolithen mit voller Sensorik und voller Trainingszeit beschrieben und in Abb. 7.17 dargestellt. Allgemein zeichnet er sich durch eine wenig zielstrebige und inkonsequente Fahrweise aus. Von Punkt 1 erreicht er eine hohe Schrittzahl, aber wenig Reward, da er ein wenig in die Sackgasse hineinfährt und dort abwechselnd nach links und rechts dreht. Von Punkt 2 fährt er an der *Gefahrenzone* vorbei auf die *Futterquelle*, mit einigen unnötigen Drehbewegungen, und verläßt die *Futterquelle* nach einigen Schritten, um mit der rechten Seitenwand zu kollidieren. Von Punkt 3 verfährt er ähnlich wie von Punkt 1 und verpaßt dadurch die Belohnung aus der *Futterquelle*. Von den Punkten 4 und 5 fährt er wieder mit einigen unnötigen Drehbewegungen auf die *Futterquelle* und verläßt diese nach einige Schritten, um zu kollidieren. Von den Punkten 6 und 7 kollidiert er nach einigen Drehbewegungen mit der nächsten Wand.

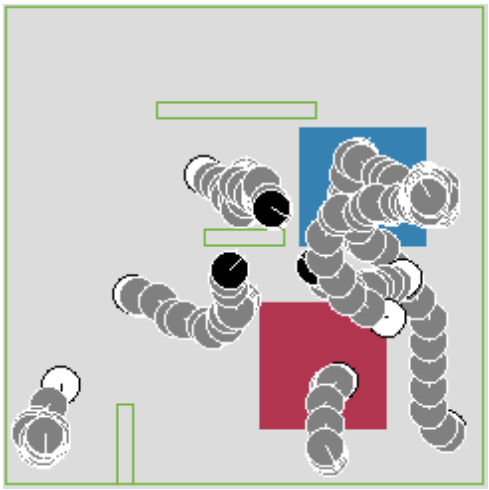

Abbildung 7.17: ***Verhalten des Monolithen mit GPS-Sensorik.*** *Der hellgraue/rote Fleck stellt die positive Quelle (Futterquelle bzw. Wild) und der dunkelgraue/blaue die negative Quelle (Gefahrenzone oder Jäger) dar. Weiße Kreise markieren einen Einsetzpunkt, schwarze den Endpunkt des Trials, graue die Schritte dazwischen. Fehlt der schwarze Punkt, wurde der Trial nach 51 Schritten abgebrochen.*

Die Untersuchungen mit dieser Umsetzung des Szenarios als MDP ergaben nicht den beabsichtigten Vergleichsmaßstab für das Szenario. Aufgrund dieser Tatsache und des enormen zeitlichen Aufwands wurde letztlich auf eine Untersuchung im realen Labyrinth verzichtet.

7.7 Zusammenfassung

In diesem Kapitel wurden die konkrete technische Umsetzung des Szenarios mit einem mobilen Miniroboter Khepera und verschiedenen Objekten vorgestellt. Im Szenario wird es Hindernisse, eine statische *Gefahrenzone*, einen dynamischen *Jäger*, eine statische *Futterquelle* und ein dynamisches *Wild* geben. Diese Aufgaben sollen von verschiedenen Expertenagenten gelöst werden, die mit IR und visuellen Informationen ausgestattet werden. Die Vorverarbeitung der visuellen Informationen wurde erläutert und die Agenten vorgestellt. Weiterhin wurden Festlegungen zur Durchführung und zur Auswertung der Experimente getroffen.

Darüberhinaus wurde im Ergebnis von Voruntersuchungen die Fahrweise des Roboters als schrittweise festgelegt und der Trainingsmodus als zweigeteilt, zunächst wird aus demselben Datensatz für alle Agenten offline vortrainiert und im Anschluß daran im realen Labyrinth nachtrainiert. Das Neurale Gas wurde als Clusterer und *Sarsa(*0.8*)* als RL-Algorithmus festgehalten.

In gesonderten Untersuchungen wurde der Einfluß der hier angewendeten Farbcodierung im Vergleich zu einer unvorverarbeiteten Präsentation der Farben sowie die Ermittlung eines Leistungsstandards für das Szenario als MDP untersucht. Der Vorteil der eingesetzten Farbcodierung konnte gezeigt werden, während die Ermittlung eines Leistungsstandards fehlschlug.

Kapitel 8

Ergebnisse der Experimente

Entsprechend den in den vorangegangenen Kapiteln vorgestellten Plänen zur Ermittlung der Leistungsfähigkeit der verschiedenen Agenten und ihrer Koordinierung werden nun die Ergebnisse der zugehörigen Experimente vorgestellt. Neben der reinen numerischen Dokumentation der Verhaltensleistungen der Agenten wird versucht, eine verbale Beschreibung des typischen Verhaltens der Agenten an wichtigen Stellen zu geben. Es existieren ausführliche Aufzeichnungen über die Schritte der Agenten für die Testläufe mit dem jeweiligen Experten, Zustand und Aktion, jedoch würde diese Dokumentation die Arbeit sprengen. Die verbale Darstellung wurde gewählt, da sowohl wegen der auftretenden, meist sehr spezifischen Probleme als auch wegen der Fülle der untersuchten Agenten, eine detaillierte Analyse des Verhaltens der Agenten in diesem Rahmen sehr problematisch ist. Besonders für die Untersuchungen im realen Labyrinth muß MATARIC bestätigt werden. Sie argumentiert in [Mataric, 1996], daß die Analyse komplexen Verhaltens noch ungelöst und in der Gegenwart von Unsicherheiten, wie die reale Umwelt sie aufweist, unlösbar ist.

Die Ergebnisse umfassen die ermittelten Leistungen der Expertenagenten und der Koordinationen, sowohl in der Simulation als auch im realen Labyrinth, jeweils mit statischen und dann mit dynamischen Objekten. Im Anschluß werden in der simulierten Umgebung mit statischen Objekten einige Erweiterungen zum W-Lernen untersucht.

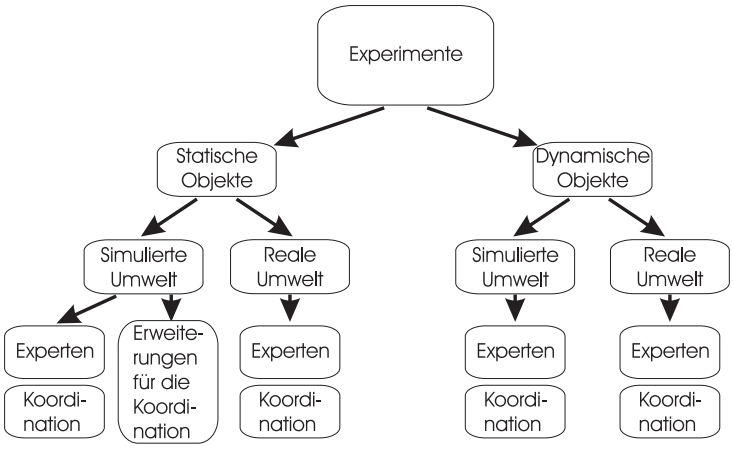

Abbildung 8.1: ***Übersicht über alle Experimente*** *mit Experten und Multiagentensystemen in simulierter und realer Umwelt, entweder mit statischen oder dynamischen Objekten.*

Abbildung 8.1 vermittelt eine Übersicht über alle dokumentierten Ergebnisse. Die Ergebnisse werden jeweils numerisch in einer Tabelle und verbal für die Einzelagenten, den Monolithen und ausgewählte Varianten des W-Lernens dargestellt. Weiter werden für einige Experimente die erzielten Fahrspuren des Roboters dargestellt und kommentiert. Abgeleitete Übersichten, in denen die jeweils diskutierten Experimente durch graue Hinterlegung gekennzeichnet wurden, sind den einzelnen Abschnitten beigefügt.

8.1 Performanz der Agenten und des W-Lernens

In den ersten Untersuchungen werden die durch die Expertenagenten, durch den Monolithen und das W-Lernen als Koordination erreichten Leistungen gegenübergestellt. Dazu wird zunächst ein Szenario mit 3 statischen Aufgaben benutzt, einer Hindernisvermeidung mit schneller gerader Fahrt im Freiraum (Agent *Bump*), einem projizierten Farbfleck (blauer bzw. dunkelgrauer Kreis) mit positivem Reward (Objekt *Futterquelle*, Agent *Hunger*) und einem Farbfleck mit negativem Reward (roter bzw. hellgrauer Kreis, Objekt *Gefahrenzone*, Agent *Flucht*). Der Roboter wurde jeweils an der zu prüfenden Position in der festgelegten Orientierung eingesetzt (vgl. die Festlegung zur Durchführung der Experimente in Abschnitt 7.5). Danach konnte der Agent den Roboter bis zur nächsten Kollision oder maximal 51 Schritte steuern. Diese Zahl als Abbruchkriterium für Untersuchungen besonders von kollisionsvermeidenden Agenten ergab sich als Kompromiß zwischen deutlicher Demonstration einer Kollisionsvermeidung und vertretbaren Zeitaufwänden für die Experimente. Diese Anzahl reicht auch bequem aus, den rechteckigen Grundaufbau auf der Diagonalen zu durchqueren.

Die Vergleichsgrundlage und untere Leistungsgrenze für die Expertenagenten bildeten die Leistungen eines sich komplett zufällig verhaltenden Agenten. Für jeden Expertenagenten wurden jedoch auch seine Leistungen hinsichtlich der Aufgaben der anderen Experten aufgeführt, obgleich der betreffende Experte nicht über die Informationen verfügte, diese gezielt zu erfüllen. Damit wird eine Abschätzung der spezifischen Kompetenz der Experten und der Abhängigkeit der Aufgaben untereinander möglich.

Vergleichsgrundlage für die Leistungen der verschiedenen Formen des W-Lernens und des Monolithen waren die Ergebnisse einer zufälligen Koordination, bei der in jedem Zustand einer der Expertenagenten zufällig ausgewählt wurde. Abschnitt 8.1.5 enthält eine zusammenfassende Interpretation der Ergebnisse der Experimente, sodaß die Interpretationen zu den vielen einzelnen Experimenten recht knapp gefaßt wurden.

8.1.1 Verhalten der Experten im simulierten Szenario

Zunächst wird auf die Leistungen der Experten unter den kontrollierten Bedingungen einer simulierten Umwelt, die nur über statische Objekte verfügt, eingegangen. Bei allen Agenten kam ein NG als Vektorquantisierer und *Sarsa*(0.8) als RL-Verfahren zum Einsatz. Die Agenten wurden mit Erfahrungswerten parametrisiert und von zufälligen Einsetzpunkten aus trainiert (kein Vortraining). Nach dem Training wurden alle Agenten anhand der 7 Einsetzpunkte getestet. Von jedem Einsetzpunkt aus wurde die Anzahl der Schritte bis zur Kollision und für den Kompetenzvergleich die Rewards für alle Teilaufgaben bzw. Objekte ermittelt und für 10 Durchläufe über alle Einsetzpunkte gemittelt. Dagegen geben

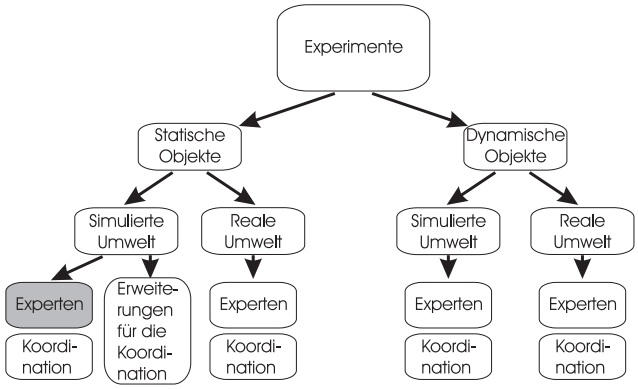

Abbildung 8.2: *In diesem Abschnitt werden die Leistungen der Expertenagenten in simulierter Umgebung mit statischen Objekten untersucht.*

die Fahrspuren das Verhalten des Agenten genau für einen Durchlauf wider. Verschiedene Durchläufe weisen das gleiche prinzipielle Verhalten, aber kleine Unterschiede in Rewards und Schritten auf.

Tabelle 8.1 gibt einen Überblick über die Leistungen aller Agenten für alle Aufgaben. Dabei wurden die Leistungen der Agenten für ihre eigentliche, spezielle Aufgabe hervorgehoben. Die Leistungen eines zufällig handelnden Agenten und des Monolithen wurden zum Vergleich mit aufgeführt. Da die spezielle Aufgabe für den Agenten *Bump* zweigeteilt mit zwei unterschiedlichen relevanten Leistungsmaßen ist, werden beide Leistungsmaße, die erreichte Schrittzahl von einem Einsetzpunkt (in einem Trial) und der Reward für schnelles und gerades Fahren, aufgeführt. Die erste Spalte enthält die durchschnittliche Anzahl gefahrener Schritte bis zur Kollision und die weiteren Spalten jeweils den durchschnittlichen Reward pro Schritt für die unterschiedlichen Aufgaben Fahren (Agent *Bump*), Objekt *Gefahrenzone* (Agent *Flucht*) und Objekt *Futterquelle* (Agent *Hunger*).

Agent	Schritte (σ)	Fahren (σ)	*Gefahrenzone*(σ)	*Futterquelle* (σ)
Bump	**43.85** (0.0)	**-0.36** (0.45)	-0.34 (0.25)	0.23 (0.21)
Flucht	17.27 (2.88)	-0.83 (0.30)	**-0.06** (0.10)	0.23 (0.29)
Hunger	23.50 (6.99)	-0.61 (0.27)	-0.16 (0.12)	**0.33** (0.28)
Monolith	43.28 (1.96)	-0.53 (0.50)	-0.04 (0.06)	0.43 (0.41)
Zufall	24.85 (0.0)	-0.66 (0.34)	-0.33 (0.22)	0.04 (0.06)

Tabelle 8.1: ***Ergebnisse der Expertenagenten im simulierten Szenario.*** *Die erreichten Leistungen belegen die spezifische Kompetenz der Expertenagenten im Vergleich untereinander.*

Die positiven Ergebnisse für die Experten in der Simulation, also mit kontrollierten Randbedingungen, unterstützen auch die Tauglichkeit der gewählten Sensorik und Agentenstruktur. Wie die Zahlen in Tabelle 8.1 belegen, erzielt jeder Expertenagent im Vergleich zu den anderen Experten für seine Teilaufgabe eine deutliche Kompetenz. Allerdings übertrifft der zufällig handelnde Agent die Agenten *Flucht* und *Hunger* bezüglich der erreichten Schrittzahl. Dies ist nicht verwunderlich, da diese diesbezüglich weder Eingangsinformationen noch Reward erhielten. Die Verhalten der einzelnen Experten werden im folgenden verbal erläutert.

Die Erzielung der größten Schrittlänge mit höchster Zuverlässigkeit, sowie bester Rewards für die Teilaufgabe, schnell und geradeaus zu fahren, belegen die Kompetenz des *Bump*-Agenten (vgl. Abb. 8.3, 2. v. links). Dabei zeigte die Inspektion der Fahrtrajektorien neben längeren schnellen Fahrten (vgl. von Punkten 2,3 und 7) und einem Verlassen von Wänden und Hindernissen jedoch auch vorsichtiges, sicheres Vor- und Zurückfahren oder Kreise (vgl. Punkte 1 und 4). Von den Punkten 5 und 6 aus beginnt er mit langen, schnellen Schritten, geht im Freiraum jedoch zu kleinem Hin- und Herfahren über.So erreicht er schnell Freiraum, wobei er Reward für Fahren sammelt, und vermeidet dann durch kleinstmögliche Bewegungen sehr gut Kollisionen. Die vergebenen Rewards waren zwar geeignet, ein Drehen in kleinsten Kreisen zu reduzieren, das der Agent nach Abschaffung der Aktion 'Stillstehen' produzierte, aber nicht, um in allen Fällen ein gerades, schnelles Fahrverhalten zu erzeugen, das auch das Risiko naher Wände eingehen würde.

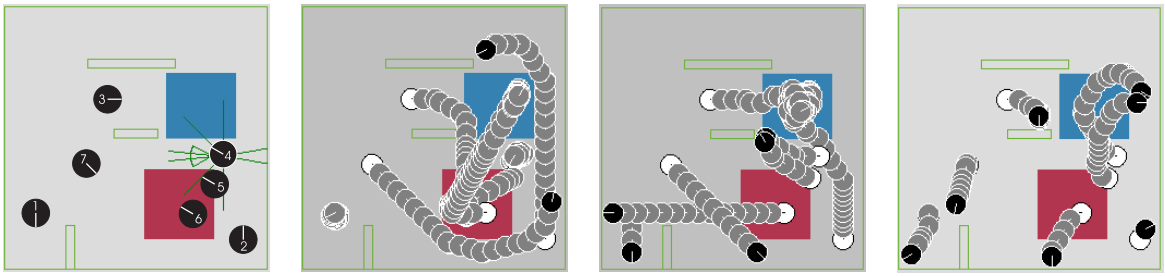

Abbildung 8.3: ***Darstellung der Fahrspuren für die drei Experten.*** *Die Fahrspuren der drei Expertenagenten (2. v. links* Bump*, 3. v. links* Hunger *und rechts* Flucht*) sind für alle Startpunkte eingezeichnet. Der Startpunkt ist durch einen weißen Kreis gekennzeichnet, der Endpunkt durch einen schwarzen. Fehlt der schwarze Endpunkt, wurde der Trial nach 51 Schritten abgebrochen.*

Hunger (vgl. Abb. 8.3, 3. v. links) entwickelt für den Fall, daß keine *Futterquelle* zu sehen ist, die Strategie, geradeaus zu fahren (Punkte 1,6,7). Sie führt ihn zu schneller Kollision von Punkt 1, jedoch auch zu Erfolgen von den Punkten 2 und 3, die die Basis seiner guten Rewards für seine Aufgabe sind. Von den Punkten 4 und 5 dreht er zwar auf die *Futterquelle*, kollidiert jedoch schnell mit der unteren, kürzeren Wand der Passage. Von 6 und 7 aus erzielt *Hunger* durch die Geraden höhere Schrittzahlen, gelangt jedoch nicht auf die *Futterquelle*. Insgesamt fährt er zielgerichtet auf die *Futterquelle*, sofern sie nicht zu weit entfernt ist, und fährt sonst geradeaus.

Im simulierten Szenario überzeugt *Flucht* (vgl. Abb. 8.3 rechts) durch gute, also geringe negative Rewards in seiner Aufgabe, erzeugt jedoch sehr schnell Kollisionen von nahezu allen Punkten (vgl. Abb. 8.3 rechts), die seine Rewards für die Teilaufgabe Fahren und seine Schrittzahlen sinken lassen. Die guten Werte für seine eigentliche und die Aufgabe *Futterquelle* entstehen durch die Punkte 4 und 5, von denen *Flucht* zunächst auf die *Futterquelle* fährt, um sie wieder zu verlassen und mit einer Wand zu kollidieren. Von Punkt 6 aus verläßt er die *Gefahrenzone* auf sehr kurzem Weg. Auch *Flucht* beweist eine gute Kompetenz bezüglich seiner Aufgabe und in anderen Zuständen folgerichtig wenig gerichtetes Verhalten.

Damit sind die beabsichtigten Interessenkonflikte, z.B. mit *Hunger* (Punkte 2, 3 und mit Einschränkungen 6) sowie *Bump* (1, 2) eingetreten.

8.1.2 Verhalten der Koordinationen im simulierten Szenario

Dieselbe simulierte, statische Umwelt diente zum Test des Monolithen und der verschiedenen Varianten von W-Lernen, das die soeben vorgestellten Expertenagenten koordiniert.

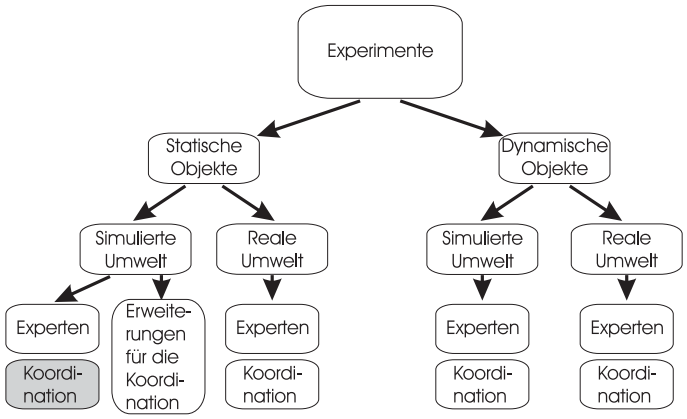

Abbildung 8.4: *In diesem Abschnitt werden Koordinationen in simulierter Umgebung untersucht.*

Die gleiche Testmethode mit 7 Einsetzpunkten wie für die Expertenagenten kam für die Varianten des W-Lernen und den Monolithen zum Einsatz. Da ihre Aufgabe jedoch in der Erfüllung der Gesamtaufgabe bestand, für die ein Gesamtreward berechnet und dem Monolithen zur Verfügung gestellt wurde, wird in Tabelle 8.2 wegen einer besseren Übersichtlichkeit nur noch der durchschnittliche Reward und die Schrittzahl aufgeführt.

Das zusammenfassende Fazit aus den Zahlen in Tabelle 8.2 lautet: Der Monolith bestätigt seine vermutete Überlegenheit gegenüber allen Koordinationen durch WL. Er erreicht, wie *Bump*, die höchste Schrittzahl und sehr viel besseren Gesamtreward als diese. Die Ergebnisse der Varianten des W-Lernen, mit Ausnahme der Varianten mit gelernten W-Werten, ähneln sich sehr. Wie für die Experten, folgt auch für den Monolithen und W-Lernen mit gelernten W-Werten eine verbale Beschreibung ihres Verhaltens.

Agent	Trialreward (σ)	Schritte (σ)
Monolith	0.92 (0.54)	43.28 (1.96)
WL(gelernte W-Werte)	-0.12 (0.08)	19.85 (0.20)
WL(negotiated)	-0.83 (0.48)	28.20 (3.74)
WL(Minimize Worst Unhappiness)	-1.05 (0.00)	33.85 (0.00)
WL(Minimize Collective Unhappiness)	-1.46 (0.00)	21.14 (0.00)
WL(Maximize Best Happiness)	-1.73 (0.00)	19.28 (0.00)
WL(Maximize Collective Happiness)	-1.46 (0.00)	21.14 (0.00)
WL(Maximize Best Profit)	-0.94 (0.78)	20.87 (6.58)
WL(Maximize Collective Profit)	-1.46 (0.00)	21.14 (0.00)
WL(zufällig)	-1.42 (0.80)	21.74 (5.68)

Tabelle 8.2: ***Ergebnisse des Monolithen und des W-Lernens im simulierten Szenario.*** *Der Monolith ist allen Koordinationen überlegen. 'zufällig'=zufällige Auswahl eines Experten.*

Abbildung 8.5 stellt das Verhalten des Monolithen und des WL mit gelernten W-Werten (beste Koordination) und des WL mit Maximize Best Happiness (schlechteste Koordination) von allen Startpunkten mit einer Kodierung für die einzelnen Leader dar.

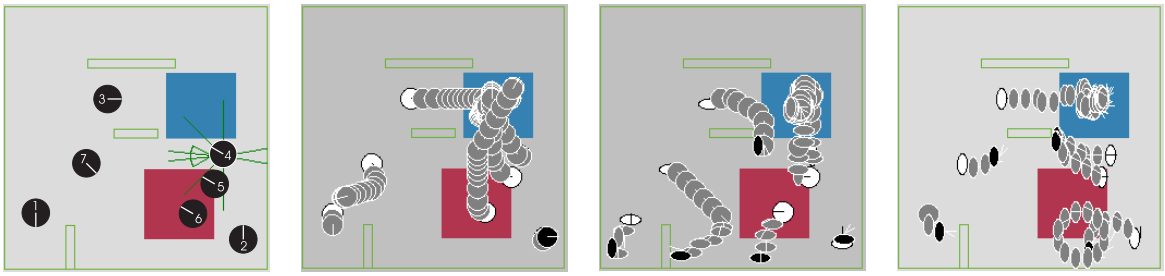

Abbildung 8.5: ***Fahrspuren ausgewählter Koordinationen****: 2. v. links der Monolith, 3. v. links WL mit lernenden W-Werten, rechts Maximize Best Happiness WL. Mit Kreis gezeichnete Schritte wurden durch* Bump*, hohe Ovale durch* Hunger *und flache Ovale durch* Flucht *veranlaßt.*

In Punkt 1 vermeidet der Monolith (Abb. 8.5, 2. v. links) zwar eine Kollision, fährt aber mit kleinsten Bewegungen vor- und zurück, anstelle umzudrehen und lange Geraden zu fahren. Von Punkt 2 fährt er nicht auf die *Futterquelle*, sondern dreht scharf nach rechts in die Wand. Von Punkt 3 jedoch fährt er direkt auf die *Futterquelle* und bleibt dort durch vor- und zurückfahren. Ein ähnliches Verhalten zeigt er von den Punkten 4, 5 und 6: er dreht scharf auf die *Futterquelle* und bleibt dort durch vor- und zurückfahren. Von Punkt 7 aus fährt er einen kleinen Kreis auf die Sackgasse zu und bleibt dort mit vor- und zurückfahren. Sein Gesamterfolg basiert auf der gelungenen Kombination von Flucht und Hunger (Punkte 3-6) und sonst eine gute Wandvermeidung (Punkte 1, 7).

Stellvertretend für die WL-Varianten wird ihre beste, 'Minimize Worst Unhappiness' mit *gelernten W-Werten* (Abb. 8.5, 3. v. links) dokumentiert. Von Punkt 1 aus führt *Flucht*, wechselt sich mit *Hunger* ab und beide steuern den Roboter in die Wand. Obgleich die Situation für *Flucht* und *Hunger* nicht unmittelbar relevant war, ausgedrückt in einem niedrigen Niveau ihrer Q-Werte, übernahmen sie die Kontrolle, weil eine oder zwei Aktionen Differenzen zu den anderen aufwiesen, die die vom zu erwartenden Leader *Bump* überstiegen. Von Punkt 2 aus steuert *Flucht* den Roboter direkt auf die rechte Wand, da die Q-Werte von *Bump* zwar höheres Niveau, aber geringere Differenzen aufweisen. Aus Punkt 3 übernimmt *Bump* berechtigt die Führung, behält sie, weil *Hunger* dieselben Aktionen vorschlägt, und steuert den Roboter an die *Futterquelle*, wo *Hunger* übernimmt, aber eine Kollision verursacht. Aus Punkt 4 führt zunächst *Flucht* den Roboter auf die *Futterquelle*, was *Hunger* wenig Verlust erzeugt, obgleich er andere Aktionen vorschlug. Dann übernimmt *Bump*, trotzdem *Hunger* größere Q-Wertdifferenzen aufweist. Da die W-Werte gelernt sind, muß es sich um einen Effekt der Generalisierung durch den Clusterer handeln, denn *Hunger* erfährt nicht den von seinen Q-Werten prädizierten Verlust. *Bump* hält den Roboter auf der *Futterquelle*. Von Punkt 5 gewinnt *Hunger*, dann bringt *Flucht* den Roboter von der *Gefahrenzone* auf die *Futterquelle*, wo *Bump* ihn wieder vor- und zurückfahren läßt, wie auch *Hunger* es tun würde. Von Punkt 6 gewinnt *Flucht* und steuert den Roboter konkurrenzlos in die untere Wand. Aus Punkt 7 führt *Bump*, bis *Flucht* den Roboter von der *Gefahrenzone* abdreht und in die Wand der Sackgasse setzt. Die Koordination erzielt sowohl gutes (Pkte 3, 4, 5), angemessenes (Pkt 6) und schlechtes Verhalten (Pkte 1, 7), aber nie eine gelungene Kombination wie der Monolith.

8.1.3 Verhalten der Experten im realen Labyrinth

Für die Untersuchungen der Experten im realen Labyrinth wurden dieselben Agenten verwendet wie für die in der simulierten Umgebung. Dies betrifft das NG als Clusterer und *Sarsa*(0.8) als RL-Algorithmus, aber auch die Vorverarbeitung der Eingangsinformationen und die Rewardermittlung. Die folgenden verbalen Beschreibungen des Verhaltens der Agenten und auch der Koordinationen beziehen sich auf Beobachtungen über mehrere Durchläufe, dagegen können die abgebildeten Fahrspuren nur die Ergebnisse eines bestimmten Durchlaufes widergeben. Verschiedene Durchläufe erzeugten abweichendes Verhalten und entsprechende Fahrspuren, deren Gründe unten diskutiert werden.

Zunächst wurden die Expertenagenten in dem in Abschnitt 7.5 vorgestellten Szenario mit 7 Einsetzpunkten trainiert und getestet. Das Training war, wie in Abschnitt 7.6.2 festgelegt, zweigeteilt: erst wurden alle Agenten anhand derselben Datei vortrainiert und danach im realen Szenario nachtrainiert.

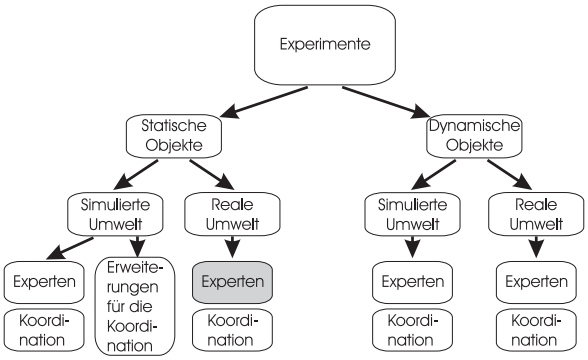

Abbildung 8.6: *In diesem Abschnitt werden Experten im realen, statischen Labyrinth untersucht.*

Agent	Schritte (σ)	Fahren (σ)	*Gefahrenzone*(σ)	*Futterquelle* (σ)
Bump	**41.08** (4.39)	**-0.05** (0.02)	-0.13 (0.11)	0.06 (0.08)
Flucht	24.55 (6.51)	-0.34 (0.27)	**-0.02** (0.04)	0.14 (0.20)
Hunger	25.58 (4.52)	-0.07 (0.08)	-0.06 (0.09)	**0.29** (0.22)
Zufall	14.28 (4.59)	-2.47 (1.93)	-0.53 (0.51)	0.06 (0.07)

Tabelle 8.3: ***Ergebnisse der Expertenagenten im realen Labyrinth.*** *Für jeden Agenten werden alle erreichten Teilleistungen (Fahren, Gefahrenzone, Futterquelle) einzeln aufgeführt. Die hervorgehobenen Zahlen belegen die Kompetenz des Agenten für seine jeweilige Teilaufgabe.*

Zusammenfassend lautet das Ergebnis aus Tabelle 8.3, daß die Expertenagenten ihre jeweilige Kompetenz unter Beweis stellen konnten. Dies durch die Zahlen belegt und im folgenden verbal diskutiert. Allerdings zeigen die weniger stark ausgeprägten Unterschiede zwischen den Experten auch deutlich die Probleme einer realen Umsetzung. So konnten jeweils nur ein sehr knappes spezifisches Training realisiert werden. Auch 10 Durchläufe pro Agent stellten eine erhebliche, kaum wiederholbare Belastung dar.

Bump gelingt die Kollisionsvermeidung gut, wie vor allem die Zahlen für die Anzahl der gefahrenen Schritte über alle Einsetzpunkte beweisen. Ein Beispiel wird in Abb. 8.7,

2. v. links demonstriert, in der *Bump* durch Drehung nach links und Rückwärtsfahren erfolgreich die Sackgasse von Punkt 1 aus vermeidet. Allerdings erreicht er seine Kollisionsvermeidung durch eine vorsichtige Fahrweise, sodaß die Rewards für *Bump* zwar besser als die der anderen Agenten sind, aber weniger deutlich.

Hunger (Abb.8.7, 3. v. links) ist bestrebt, so schnell wie möglich auf die *Futterquelle* zu gelangen und erzielt dadurch auch bessere Rewardwerte für schnelles und gerades Fahren als *Bump*. Auch verweilt der Agent nicht lange auf der Gefahrenzone, sondern fährt zielstrebig auf die *Futterquelle*. Dort angekommen, bewegt er sich vor und zurück, da ihm ein Stillstehen versagt ist. Der von ihm erzielte Reward in seiner Aufgabe liegt so deutlich über dem aller anderen Experten.

Flucht (Abb.8.7 rechts) ist zwar eindeutig der Experte, wenn es darum geht, die Gefahrenzone zu verlassen, verhält sich jedoch dann in anderen Teilen des Szenarios relativ ziellos und erreicht nur geringe Rewards hinsichtlich des Fahrens. Andererseits läßt ihn das auch auf der *Futterquelle* verweilen, sodaß er hier relativ hohe Rewards erzielt, obgleich er dorthin nur auf der Flucht von der Gefahrenzone gelangt und die *Futterquelle* auch wieder verläßt. Von den Punkten 5 und 6 aus verläßt er in wenigen Schritten die *Gefahrenzone*, und kollidiert mit der rechten Wand.

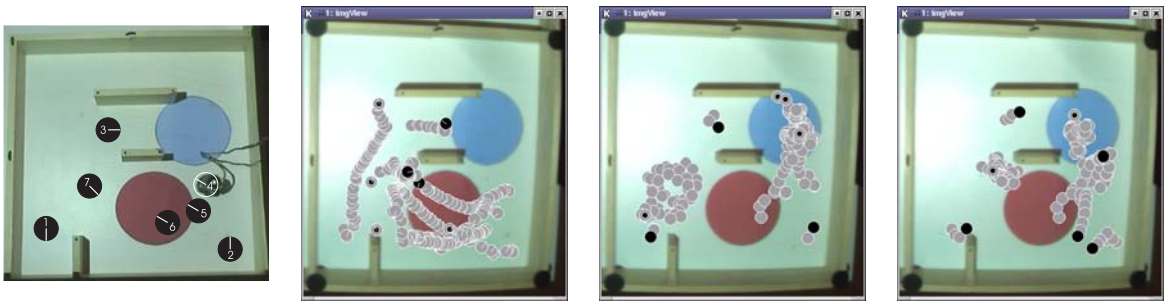

Abbildung 8.7: ***Darstellung der Fahrspuren für die drei Experten.*** *Die Fahrspuren der drei Expertenagenten (2. v. links* Bump*, 3. v. links* Hunger *und rechts* Flucht*) ist für alle Startpunkte eingezeichnet. Die Startpunkte sind meist durch die folgenden Spuren verdeckt. Trials, die nach 51 Schritten endeten, sind durch kleine schwarze Kreise markiert, während große schwarze Kreise eine Kollision andeuten. Teilweise fehlen die Orientierungen des Roboters, da diese bei Aufzeichnung der Fahrspuren noch nicht ermittelt werden konnten.*

8.1.4 Verhalten der Koordinationen im realen Labyrinth

Auch im realen Labyrinth wurden die verschiedenen Varianten der Koordination mit W-Lernen auf der Basis der eben vorgestellten Expertenagenten untersucht. Wiederum konnten nur 10 Durchläufe pro Koordination durchgeführt werden.

Wie in der simulierten Umgebung bestätigt der Monolith mit den Ergebnissen in Tabelle 8.4 seine vermutete Überlegenheit gegenüber den Varianten des WL als Koordination. Von nahezu allen Einsetzpunkten aus verfolgt er Pfade, die dem Beobachter, in Abwesenheit gesichert optimaler Verhaltensweisen, als sehr sinnvoll erscheinen. Die aufgeführten Varianzen belegen jedoch, daß er diese Verhalten nicht immer zeigt. Die Varianten des

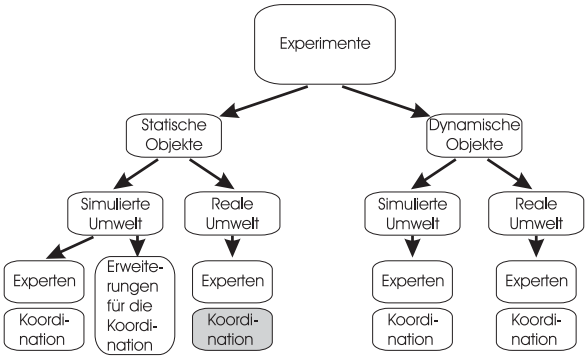

Abbildung 8.8: *In diesem Abschnitt werden Koordinationen im realen Labyrinth mit statischen Objekten diskutiert.*

W-Lernens weisen nicht mehr einigermaßen einheitliche Leistungen auf, und in der realen Umgebung ist die ungelernte, 'pure' Variante von 'Minimize Worst Unhappiness' den anderen überlegen.

Agent	Trialreward (σ)	Schritte (σ)
Monolith	3.16 (0.37)	46.05 (3.11)
WL(gelernte W-Werte)	0.69 (0.37)	17.85 (3.980)
WL(negotiated)	-0.19 (0.26)	20.55 (4.24)
WL(Minimize Worst Unhappiness)	1.45 (0.49)	31.40 (3.13)
WL(Minimize Collective Unhappiness)	0.38 (0.28)	20.75 (3.65)
WL(Maximize Best Happiness)	0.98 (0.32)	23.98 (3.20)
WL(Maximize Collective Happiness)	0.13 (0.43)	18.27 (3.99)
WL(Maximize Best Profit)	0.99 (0.52)	23.60 (5.14)
WL(Maximize Collective Profit)	0.44 (0.41)	18.58 (3.22)
WL(zufällig)	0.11 (0.41)	14.42 (4.49)

Tabelle 8.4: ***Ergebnisse des Monolithen und verschiedener Varianten des W-Lernens im realen Labyrinth.*** *Wiederum ist der Monolith den Koordinationen überlegen. Deren Vergleich untereinander ist sehr viel inhomogener als in der Simulation. 'zufällig'=zufällige Auswahl eines trainierten Experten.*

Eine Verhaltensanalyse ähnlich der für W-Lernen in Abschnitt 8.1.2 muß allerdings auf weiterführende Experimente außerhalb dieser Arbeit verschoben werden. Sie muß die vorhandenen Protokolle der Experimente beim Test hinsichtlich der W- und Q-Werte ergänzen, da für jeden Schritt jedes Trials in jedem Durchlauf der Leader, dessen Zustand und die gewählte Aktion festgehalten wurden. Diese Protokolle zeigen vor allem einen erheblichen Zeitbedarf hinsichtlich der Auswertung verschiedener Koordinationen, aber auch möglicher Fehlerquellen. Diese betreffen Auswirkungen von Ungenauigkeiten beim Einsetzen des Roboters per Hand, von Beleuchtungsschwankungen, die zu einer veränderten Farbwahrnehmung und daher zu anderen Zuständen für die Agenten führen, was wiederum ein verändertes Koordinierungsergebnis hervorbringt. Trotz eines Einzeichnens der vorgesehenen Orientierung des Roboters im Labyrinth durch Markierungen an dessen vorderem Teil und den Rädern kann der Roboter zum Beispiel dennoch leicht

gedreht eingesetzt worden sein, oder seine Räder können unterschiedlich rutschen und dadurch andere Folgezustände erzeugen, was wieder eine andere Koordinierung zur Folge hat.

Von Punkt 1 aus nimmt der Monolith die Bestrafung für Rückwärtsfahren in Kauf, um die Sackgasse zu vermeiden, und dreht den Roboter dann mit Linksdrehungen um, kann im Anschluß einige lange Schritte unternehmen, bis er an die obere Wand gelangt. Von Punkt 2 aus fährt der Agent den Roboter mit vorsichtigen Links- und Rechtsdrehungen zunächst aus der Ecke und auf die *Gefahrenzone* zu, an dessen Rand wählt er jedoch lange Schritte, um auf die *Futterquelle* zu fahren. So erreicht er sie jedes Mal, zwar nicht in einer minimalen Schrittzahl und mit vergebenen Rewards für lange Schritte direkt aus der Ecke und einigen unnötigen Bestrafungen am Rand der *Gefahrenzone*, aber sehr sicher. Die Taktiken des initialen Drehens wendet er auch von Punkt 3 aus an und gerät in die Nähe der oberen, langen Wand, kann diese jedoch mehrfach knapp vermeiden und so mit einigen zusätzlichen Schritten durch die Passage auf die *Futterquelle* fahren. Im Gegensatz dazu fährt er von Punkt 4 direkt auf die *Futterquelle* und bleibt darauf. Gleiches gilt von Punkt 5, wo er in etwas mehr als minimaler Schrittzahl von der *Gefahrenzone* herunter, aber im nächsten Schritt auf die *Futterquelle* fährt. Auch den konfliktträchtigen Punkt 6 löst er zugunsten der *Futterquelle*. Er braucht einige Schritte mehr als minimal, um die *Gefahrenzone* zu verlassen, gelangt dadurch jedoch schnellstmöglich auf die *Futterquelle*.

Abbildung 8.9 stellt das Verhalten des Monolithen, des besten, 'puren' und des schlechtesten WL (Negotiated Minimize Worst Unhappiness) von allen Startpunkten dar.

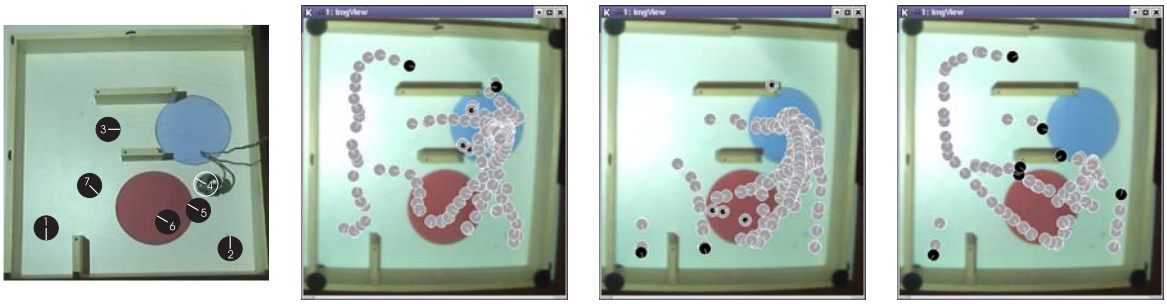

Abbildung 8.9: ***Darstellung der Fahrspuren ausgewählter Koordinationen****: 2. v. links der Monolith, 3. v. links 'pures' WL (*mWU*), rechts Negotiated WL.*

Ein Verhalten jedoch, das keiner der Expertenagenten und auch keine der Koordinationen zeigen konnte, ist in Abb. 8.9 2. v. links veranschaulicht: von Punkt 7 aus kann der Monolith sich an den Hindernissen und der *Gefahrenzone* so orientieren, daß er die Bestrafung durch die *Gefahrenzone* in Kauf nimmt und an den Hindernissen vorbei auf die *Futterquelle* fährt. Dies zeigt sowohl die Ausnutzung aller zur Verfügung stehenden Informationen, als auch die implizite Planung der gesamten Sequenz, wie sie die Koordination nicht leisten kann. Ihr fehlt die Vorschau auf Werte folgender Zustände, da deren Leader nicht bekannt sind.

Stellvertretend für die Varianten der Koordination durch W-Lernen soll das Verhalten von purem 'Minimize Worst Unhappiness' (*mWU*, Abb. 8.9 3. v. links), als der besten Koordination dokumentiert werden. Bei dieser Variante wird der W-Wert als Verlust eines Agenten berechnet, jedoch nicht aus den vorgeschlagenen Aktionen, sondern aus allen, und

es wird aus diesen Verlusten in jedem Zustand das absolute Maximum ausgewählt. Daher gibt es in den verschiedenen Zuständen keine Leader, sondern nur ausgeführte Aktionen.

Wie alle anderen Varianten, fährt auch *mWU* in Punkt 1 den Roboter in 3 Schritten gegen die untere Wand. Offenbar erzeugt der von *Bump* für das Rückwärtsfahren, Umdrehen mit anschließenden Geraden aufgebaute Reward keine ausreichenden Verluste, damit sich die entsprechende Aktion durchsetzen kann. Anstelle dessen überwiegt die Taktik der langen geraden Schritte in Erwartung anschließender Rewards von der *Futterquelle*, wie sie *Hunger* bevorzugt und in dieser Situation ungerechtfertigt einsetzt. Jedoch ist diese Strategie in einigen Fällen von Punkt 2 aus erfolgreich, von wo *mWU* in einigen Läufen mit einigen zusätzlichen Schritten die *Futterquelle* erreichen kann. Dies gilt auch für Punkt 3, wo *mWU*, wie die anderen Koordinationen auch, zunächst wieder lange Schritte macht, aber dann manchmal die Kollision der anderen WL-Varianten verhindern und auf die *Futterquelle* gelangen kann. Von Punkt 4 aus erreicht der Roboter zwar sicher die *Futterquelle*, braucht wegen der häufigen Links- und Rechtsdrehungen jedoch viel mehr Schritte dazu. Auch von 5 und 6 aus erreicht der Roboter schließlich die *Futterquelle*, jedoch nach leistungsmindernden Hin- und Herdrehungen mit mittlerer Geschwindigkeit anstelle langer Schritte auf die *Futterquelle*. Von Punkt 7 aus fährt er lange Schritte und umfährt auch die *Gefahrenzone*, kollidiert jedoch mit der unteren Wand des Labyrinths.

Zusammenfassend ist festzustellen, daß alle 'Kollektiven' Methoden schlechter abschneiden als die zugehörigen 'Best' Methoden, in denen die Agenten jeweils eine Aktion vorschlagen. Die besten Ergebnisse zeigten die Varianten, die sich nahezu ausschließlich auf den *Hunger* als Leader verließen (*Best Happiness*, *Best Profit)*. Nur sehr selten und dann meist zu spät kamen *Bump* oder *Flucht* zum Zuge. Dies wirkt sich besonders in den Punkten 1, 3, 6 und 7 leistungsmindernd aus.

Die WL-Varianten ähneln sich in ihrem beobachtbaren Verhalten sehr, bis auf einige wenige Situationen, die aber für die weitere Aktionsfolge entscheidend sind und so die Leistungsunterschiede besonders von den Punkten 4-6 aus begründen.

8.1.5 Analyse des Verhaltens

Wie schon von Humphrys hervorgehoben, ist die Höhe der Q-Werte entscheidend für die Koordinierung. So kann es leicht vorkommen, daß ein Agent mit hohem Q-Wertniveau und entsprechend hohen Verlusten, sollte er nicht zum Zuge kommen, das gesamte Verhalten dominiert.

In dem vorgelegten Vergleich nahezu aller in [Humphrys, 1997] beschriebenen Methoden des WL läßt sich feststellen, daß die kollektiven Methoden schlechter abschneiden als diejenigen, bei denen die Agenten aus ihrer isolierten Sicht eine Aktion vorschlagen. Dies bestätigt die Intuition in [Humphrys, 1997], S. 153, wo er mutmaßt, daß die Unterordnung einzelner Experten unter die gegensätzliche Meinung des Restkollektivs dem Gesamtverhalten abträglich sei. Dies widerlegt auch die Vermutung, die kollektiven Methoden könnten behilflich sein, den Gesamtreturn zu maximieren, indem nicht nur der Erfolg des einzelnen Experten betrachtet wird, sondern der für alle Agenten, wie es der Gesamtreturn auch tut. Ein Vergleich der verschiedenen WL-Varianten mit Ergebnissen aus der Literatur konnte nicht durchgeführt werden, da die meisten in dieser Arbeit erstmals implementiert und untersucht wurden.

Weiterhin bestätigt sich zumindest im realen Labyrinth seine Mutmaßung, pures 'Minimze Worst Unhappiness' sei eine erfolgversprechende Variante, wenn alle Agenten das gleiche Aktionsspektrum aufweisen.

Generell muß festgestellt werden, daß ein Monolith in beiden Umsetzungen bessere Leistungen erzielte als W-Lernen. Dies ist besonders auf die unflexible Wichtung der Agenten und die Vernachlässigung des Gesamtrewards als Leistungskriterium für die Koordination zurückzuführen. Dies kommt im Widerspruch zwischen den Punkten 1 und 3 gut zum Ausdruck: gewinnt zum Beispiel *Bump* den Wettkampf in Punkt 1, kann es den Roboter aus der Sackgasse herausführen. *Bump* gewinnt dann jedoch auch in Punkt 3 und wird den Roboter zur Wandvermeidung rückwärts aus der Passage heraus oder nach links in die lange Wand drehen. Umgekehrt verhielte sich *Hunger*: es führt den Roboter durch die Passage auf die *Futterquelle*, wird ihn jedoch von Punkt 1 aus gegen die Wand setzen.

In der realen Umsetzung des Szenarios machte sich die Empfindlichkeit des RL gegenüber Rauschen und die Inflexibilität des gelernten Verhaltens stark bemerkbar: schon kleinste Änderungen beim Einsetzen des Roboters oder Schwankungen der Beleuchtung durch den Beamer oder die Hintergrundbeleuchtung oder auch der Räder des Khepera bewirkten ein Erkennen eines anderen Zustandes mit entsprechend unterschiedlichen Aktionen. Solches Rauschen aber ist unvermeidbar. Schon wenn der Roboter bewegungslos im Farbfleck steht, und die ihn umgebende Farbe aufnimmt, ändert sich der H-Wert relativ stark trotz der Mittelung und konstanter Beleuchtungsverhältnisse. Die Änderung einer einzigen Zustandserkennung jedoch beeinflußt die gesamte folgende Aktionsfolge.

Die Ursache dieser starken Auswirkungen unvermeidbaren Rauschens des visuellen Eingabestroms liegt in der Anwendung eines Clusterers, der seine Zustände allein nach der Ähnlichkeit der präsentierten Eingabemuster im sensorischen Zustandsraum plazierte, ohne Berücksichtigung der Konsequenzen für die folgende Aktionsgenerierung. So wurden Situationen auf den gleichen Zustand abgebildet, die sich zwar sensorisch sehr ähnlich sind, aber optimalerweise unterschiedliche Aktionen nach sich ziehen sollten, wie Punkt 1 im Vergleich zu Punkt 3, aber auch 2. Durch eine zweckgemäßere Identifikation des Zustandes könnte das Verhalten in diesem zielgerichteter ausfallen und auch die Koordinierung fiele eindeutiger aus, da sowohl Zustände und folglich auch Agenten genauer zuzuordnen wären. Eine solche Beeinflussung der sensorischen Vorverarbeitung durch die anschließende Aktionsgenerierung führt das in Kapitel 1 diskutierte verhaltensbasierte, sensomotorische Paradigma logisch weiter von der funktionalen Unterteilung hin zu einer aktiven Situationsinterpretation aus der Sicht der Aktionen, die mit ihren zugehörigen Bewertungen dann optimales Verhalten ermöglichen.

Das Training eines RL-Agenten erweist sich im realen Labyrinth als quälend ineffizient, da alle Aktionen in allen Zuständen probiert werden müssen und sich so die zielführenden Sequenzen, also erfolgreiche Folgen von Aktionen und Zuständen, nur sehr langsam herausbilden. Die erheblichen Trainingszeiten könnten verkürzt oder effektiver gestaltet werden, wenn die Charakteristik der Ausbreitung des Rewards von seinen Quellen mit den RL-Verfahren systematisch ausgenutzt würde, indem eine entsprechende Exploration entwickelt wird. Für weiterführende Arbeiten betrifft dies sowohl die Verwendung der in Abschnitt 4.6.2 erwähnten Methoden des 'Learning from easy missions', bei dem in nur langsam wachsender Entfernung von der Rewardquelle exploriert wird, und die Weiterentwicklung der zielgerichteten Explorationen aus Abschnitt 4.5.

8.2 Szenario mit dynamischen Objekten

Im nächsten Schritt wurde das Szenario um dynamische Objekte, wie den *Jäger*, weiter vervollständigt. Der dynamische *Jäger* unterscheidet sich von der statischen *Gefahrenzone* nur dadurch, daß der Khepera aktiv verfolgt wird. In ähnlicher Weise flieht das dynamische *Wild* vor dem Khepera. Dieser anscheinend kleine Schritt hatte einige gravierende Auswirkungen. Die Konsequenzen technischer Natur für den Experimentalaufbau wurden in Abschnitt 7.1.2 erläutert, die interessanteren hinsichtlich der Anforderungen an das Animat und den Agenten dagegen im Abschnitt 6.1.1. Das Szenario wurde um dynamische Objekte erweitert und die Agenten unverändert belassen bzgl. ihrer Eingangsdaten und Rewards. Sie erhalten also keine Form von Zeitinformation als jene, die implizit in den Traces steckt. Da aber die Objekte auf die Aktionen des Agenten reagieren, würde ihm zum Beispiel ein Mitführen der letzten eigenen Aktionen die Vorhersage der folgenden Zustände ermöglichen. Der Agent lernt zwar kein explizites Bewegungsmodell für die Objekte, es steckt jedoch implizit in der Wahrscheinlichkeitsverteilung der Folgezustände, die in die Q-Werte eingeht. Die Folgezustände jedoch ändern sich abhängig von den Aktionen des Agenten, womit der Agent zumindest während des Lernens mit einem POMDP konfrontiert ist.

Da mit den Traces die gesamte durchlaufene Aktionssequenz adaptiert wird, enthalten diese am Ende der Sequenz Informationen über die Vorgeschichte zu einer Aktion, zum Beispiel die vorangegangenen Aktionen. So kann das Problem des POMDP zumindest erleichtert, wenn auch nicht systematisch gelöst werden.

8.2.1 Ergebnisse der Experten mit dynamischen Objekten im simulierten Szenario

Die gleichen Experten wie oben werden nun mit dynamischen Objekten trainiert und getestet. Die dynamischen Objekte waren so konfiguriert, daß sie mit einer gleichbleibenden Geschwindigkeit, die ungefähr der minimalen Vorwärtsgeschwindigkeit des Roboters entsprach, dem Roboter ausweichen oder ihm folgen konnten. Da dynamische Wände nicht betrachtet wurden, beschränkten sich die Experimente auf die Agenten *Flucht* und *Hunger*. Die dynamischen Objekte starteten von den gleichen Positionen wie vorher die statischen und wurden nach jeder Kollision des Roboters wieder dorthin zurückgesetzt.

Im simulierten Szenario wurden zwei Agenten mit je einem dynamischen Objekt trainiert, ohne zusätzliche Informationen über dessen Bewegung zu erhalten. Zum Vergleich wurden dann jeweils die mit den statischen Objekten trainierten Agenten im Szenario mit den dynamischen Objekten getestet.

Prinzipiell stehen die Agenten vor ähnlichen Aufgaben. Mit ihren lokalen Informationen müssen die Agenten auf die Objekte reagieren, nur hängt der Erfolg beim dynamischen Objekt stärker davon ab, daß der Agent die höchste verfügbare Geschwindigkeit wählt. Das Training für die dynamischen Agenten ist jedoch deshalb schwieriger, weil die Objekte auf die Aktionen des Roboters reagieren, und so die positiven Erfahrungen für beide Agenten noch spärlicher sind. Erfolgreiche Aktionssequenzen sind entsprechend natürlich sehr viel seltener, da sie mehrere Versuche mit Ketten erfolgreicher Aktionen benötigen.

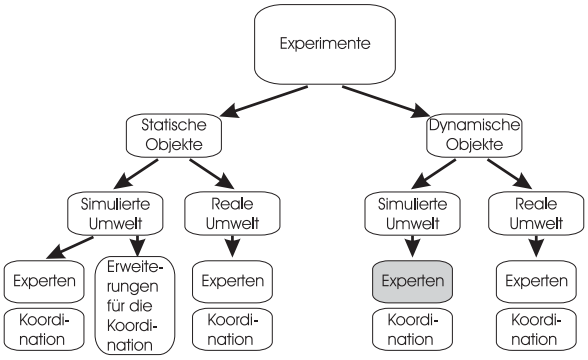

Abbildung 8.10: *In diesem Abschnitt werden Experten in der simulierten Umwelt mit dynamischen Objekten untersucht.*

Die entsprechenden Ergebnisse sind in Tabelle 8.5 aufgeführt. Der Agent, der mit der Flucht vor dem dynamischen *Jäger* betraut ist, löst seine Aufgabe sehr viel weniger überzeugend, als der, der das dynamische *Wild* erreichen soll. Dies zeigt die höhere Schwierigkeit, die die Dynamik des *Jägers* erzeugt. Der mit der statischen *Gefahrenzone* trainierte Agent entwickelte erfolgreichere Fluchtstrategien, als der mit dem dynamischen *Jäger* trainierte. Hingegen zeigt der mit dem dynamischen *Wild* trainierte Agent wie erwartet ein erfolgreicheres Verhalten, als der mit der statischen *Futterquelle* trainierte. Zum Vergleich der spezifischen Kompetenz der Agenten wurden auch die Rewards für die jeweils andere Aufgabe mit aufgeführt.

Agent	Schritte (σ)	*Jäger* (σ)	*Wild* (σ)
Stat *Flucht*	7.42 (0.0)	**-0.11** (0.13)	0.13 (0.20)
Dyn *Flucht*	11.85 (0.0)	**-0.18** (0.23)	0.18 (0.29)
Stat *Hunger*	19.2 (0.0)	-0.24 (0.20)	**0.22** (0.31)
Dyn *Hunger*	49.1 (0.0)	-0.13 (0.27)	**0.46** (0.42)

Tabelle 8.5: ***Ergebnisse der Expertenagenten in der Simulation mit dynamischen Objekten.*** *Die Agenten beweisen nur zum Teil ihre spezifische Kompetenz für ihre Aufgaben. 'Dyn' bezeichnet den mit einem dynamischen Objekt trainierten Agenten, 'Stat' den mit dem entsprechenden statischen Objekt trainierten.*

Das Verhalten der verschiedenen Agenten wird in Abb. 8.11 für alle Einsetzpunkte in der simulierten Welt dargestellt. Links die Fahrspuren der Agenten, die mit einem *Jäger* getestet wurden, rechts die mit einem *Wild*. Die Bewegung der Objekte wurde zur Erhaltung der Übersichtlichkeit nicht mit angedeutet. Die obere Reihe enthält die Fahrspuren der statisch trainierten Agenten, die untere die Spuren der mit dynamischen Objekten trainierten Experten.

Der dynamische *Hunger*-Agent entwickelt spezielle Taktiken, wie sich rechtzeitig umzudrehen, wenn er über den Mittelpunkt des *Wildes* hinaus gefahren ist und das *Wild* folglich seine Bewegungsrichtung ändert, oder über deren Mittelpunkt hin- und herzufahren, sodaß das *Wild* in jedem Schritt die Bewegungsrichtung ändert und das Animat stets auf dem *Wild* bleibt. Der dynamische Agent folgt dem *Wild* auch ohne Drehung, während der statische versucht, Drehbewegungen zu starten, sobald er auf dem *Wild* ist,

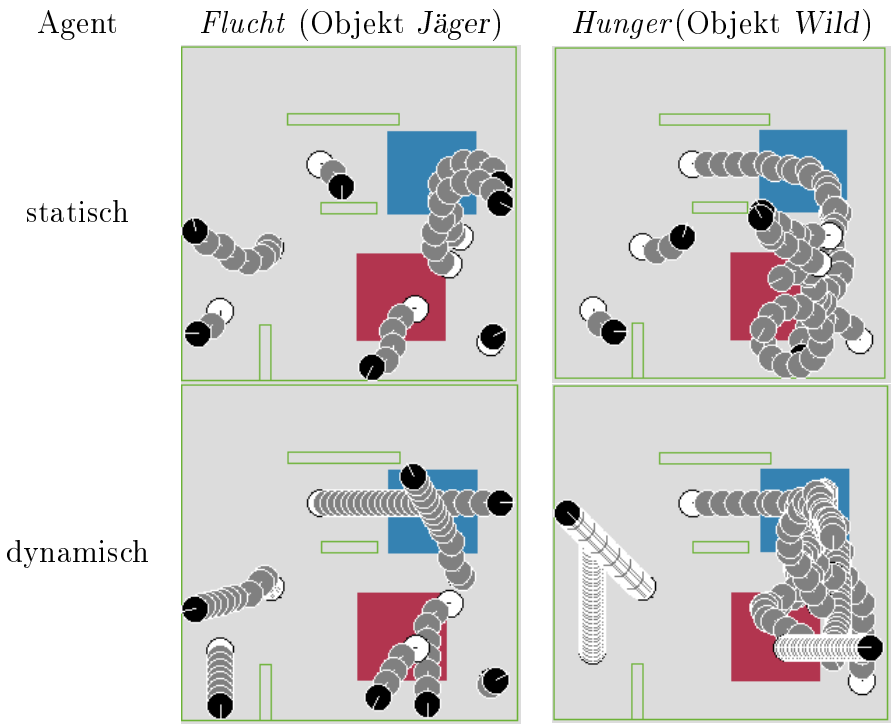

Abbildung 8.11: ***Gegenüberstellung des Verhaltens dynamischer und statischer Agenten in dynamischer Umwelt***. *Die obere Reihe enthält die Fahrspuren der statisch trainierten Agenten in der simulierten Umwelt mit jeweils einem dynamischen Objekt. Die untere Reihe zeigt die Fahrspuren der mit diesen dynamischen Objekten trainierten Agenten. Jäger: dunkelgrau/rot, Wild: hellgrau/blau*

um dessen Rand zu umrunden. Nur bewegt sich das *Wild* unter ihm weg und er muß zurückdrehen, um ihm zu folgen.

Es gelingt den simulierten Agenten auch, das *Wild* in die Ecke zu drängen und dort zu fixieren, indem er nicht soweit auf das Objekt fährt, daß es beginnen würde, in die Gegenrichtung auszuweichen. Dazu benutzt er einen Bewegungszyklus, der aus geraden Schritten vorwärts und rückwärts besteht.

Der dynamische *Flucht*-Agent entwickelt erwartungsgemäß sehr viel energischere Strategien, wenn er auf dem *Jäger* gestartet wird, während der statische Agent mit Drehbewegungen beginnt, sobald er vom *Jäger* heruntergefahren ist, und auch nicht so zielstrebig herunter fährt. Darüber hinaus entwickelt der dynamische Agent fortgeschrittene Taktiken, wie zum Beispiel, sich zwischen den Wänden der Passage von Punkt 3 durch minimale Vor- und Zurückbewegungen zu verstecken, da der *Jäger* die Wand dorthin nicht durchbrechen kann.

8.2.2 Ergebnisse der Koordinationen mit dynamischen Objekten im simulierten Szenario

In diesen Experimenten wurden nicht mehr alle Varianten der Koordinierung mit W-Lernen getestet, sondern nur die zwei, die sich in den Experimenten mit statischen Objekten bewährt hatten, aber auch das zentrale Prinzip 'Minimize Worst Happiness' des

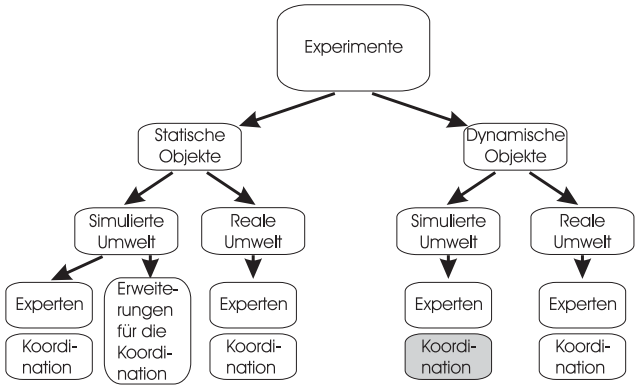

Abbildung 8.12: *In diesem Abschnitt werden Koordinationen in der simulierten Umwelt mit dynamischen Objekten untersucht.*

W-Lernens umsetzen, einmal mit gelernten W-Werten (*wlWL*), das andere Mal negotiated (*negot WL*). Wegen der veränderten Anforderungen an den Agenten, die ein dynamischer *Jäger* gegenüber einer statischen *Gefahrenzone* stellt, wurden die Koordinierungsversuche nur mit einem dynamischen *Jäger* ausgeführt. Von einem dynamischen *Wild* waren keine prinzipiellen Änderungen des Verhaltens und daher auch der Koordination zu erwarten, obgleich der Agent anschauliches Verhalten erzeugte.

Tabelle 8.6 belegt einmal mehr die Überlegenheit des Monolithen über W-Lernen und den Vorteil, den die Koordinierung mit gelernten W-Werten aus der tatsächlichen Erfahrung mit der Umwelt zieht.

Agent	Trialreward (σ)	Schritte (σ)
Mono Dyn *Flucht*	-0.01 (0.02)	37.57 (0.00)
wlWL Dyn *Flucht*	-0.89 (0.68)	22.45 (2.86)
negot WL Dyn *Flucht*	-1.99 (0.12)	16.37 (0.29)

Tabelle 8.6: ***Ergebnisse der Koordinationen in der Simulation mit dynamischer* Flucht.** *Bei diesen Versuchen wurden die oben diskutierten Expertenagenten in ausgewählten Koordinationsvarianten eingesetzt, während für* Bump *und* Hunger *dieselben, statischen Agenten verwendet wurden.*

Der Monolith (Abb.8.13, 2. v. links) flüchtet hier von Punkt 1 in die Sackgasse und von Punkt 2 rückwärts vor dem *Jäger*. Von Punkt 3 aus fährt er durch die Passage bis zum Rand der *Futterquelle* und bleibt dort, weil der *Jäger* wegen der Wand nicht zu ihm gelangen kann. Dies gilt nicht von Punkt 4, wo er dem Jäger entgegen, über ihn hinweg in den Freiraum und die kurze Wand der Passage fährt. Beginnend von Punkt 5 und 6 fährt der Monolith zuerst auf die *Futterquelle* und bleibt dort, obgleich der *Jäger* ihn erreicht. Von Punkt 7 aus fährt der Monolith in Richtung der unteren Wand, um dann entschlossen auf die *Futterquelle* zu fahren und dort zu bleiben, trotz sich unter ihn schiebenden Jägers. Die maximale Triallänge von 51 reichte offenbar nicht, ihn noch zur Flucht vor dem Abbruch zu bewegen.

Das Verhalten von gelerntem WL (Abb. 8.13, 3. v. links) mit dem Experten für dynamische *Jäger* spiegelt dessen gestiegene Wichtigkeit wider. Von Punkt 1 aus übernimmt

erst *Hunger*, dann jedoch *Flucht* die Kontrolle über den Roboter und flüchtet vor dem *Jäger* in die Sackgasse und kollidiert mit deren Wand. Von Punkt 2 aus kontrolliert *Flucht* das Verhalten sofort und dreht vor dem sich nähernden *Jäger* in die rechte Wand des Szenarios. Von Punkt 3 aus beginnt zunächst *Bump*, zwischen den Wänden zu navigieren, dann fährt *Hunger* den Roboter bis auf die *Futterquelle* und hält ihn dort, obwohl der *Jäger* sich nähert. Von Punkt 4 aus steuert *Flucht* den Roboter entschlossen weg vom *Jäger* auf die *Futterquelle*, wo *Hunger* ihn hält, obgleich der *Jäger* den Roboter erreicht. *Hunger* bleibt weiter der führende Agent. Ganz anders von Punkt 5, wo *Flucht* sofort dem Jäger entgegen dreht, über ihn und in die untere Wand fährt. Die gleiche Policy verfolgt *Flucht* von Punkt 6 aus. Auch aus Punkt 7 dreht er den Roboter vor dem nahenden *Jäger* in die Sackgasse und in die Wand.

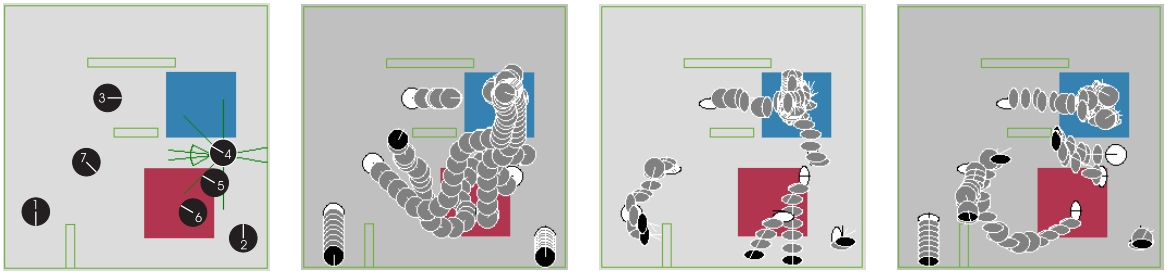

Abbildung 8.13: ***Verhalten der Koordinationen mit dynamischem Jäger in der Simulation.*** *2. v. links zeigt das Verhalten des Monolithen, 3. v. links das des WL mit gelernten W-Werten und rechts das des Negotiated WL zum Vergleich. Mit Vollkreis gezeichnete Schritte wurden durch* Bump, *hohe Ovale durch* Hunger *und flache Ovale durch* Flucht *veranlaßt.*

Die unerwartet cleveren Taktiken der Agenten spiegeln das dem Reinforcement Lernen innewohnende Potential zur Verhaltenserzeugung wider, sofern genügend Trainingszeit in einer gut beschriebenen, unverrauschten Umgebung verfügbar ist. Dieses Potential beschreibt jedoch eher die kognitive Leistungsfähigkeit des RL für einen einzelnen Agenten. Ob die Agenten diese Fähigkeiten in der Koordination entfalten und persistent verfolgen oder bei entsprechender Gelegenheit opportunistisch zum Einsatz bringen können, hängt von der Koordination ab.

8.2.3 Ergebnisse der Experten mit dynamischen Objekten im realen Labyrinth

Auch im realen Labyrinth wurden die Agenten und einige Koordinationen mit dynamischen Objekten getestet. Dabei wurde das Trainingsregime aus der statischen Umwelt für eine Vergleichbarkeit beibehalten. Mit den dynamischen Objekten, die auf die Bewegungen des Roboters reagierten, wirkte sich die notwendige Begrenzung der Trainingszeit stärker aus als in der Simulation.

Zunächst wurden zwei Expertenagenten nach demselben zweistufigen Verfahren mit dynamischen Objekten antrainiert. Wegen der Reaktion der Objekte auf die Aktionen des Roboters und dadurch sehr viel höhere Variabilität der anzutreffenden Situationen war das Vortraining aus der Datei weniger effektiv. Bei den Objekten handelte es sich einerseits um ein dynamisches *Wild* und andererseits um einen dynamischen *Jäger*.

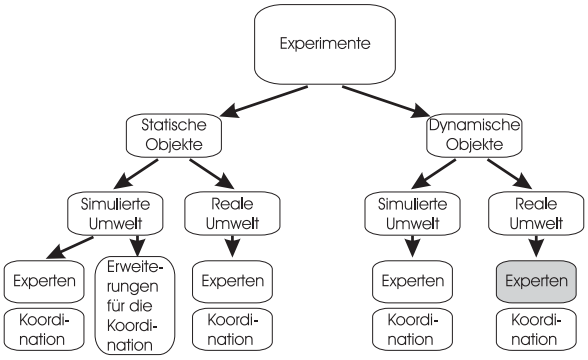

Abbildung 8.14: *In diesem Abschnitt werden Expertenagenten im realen Labyrinth mit dynamischen Objekten untersucht.*

Wie Tabelle 8.7 zeigt, sind die Ergebnisse für die Agenten in der realen Umwelt ähnlich mehrdeutig und entsprechen den Erfahrungen aus der Simulation.

Agent	Schritte (σ)	*Jäger* (σ)	*Wild* (σ)
Stat *Flucht*	13.35 (2.12)	**-0.07** (0.08)	0.06 (0.09)
Dyn *Flucht*	12.52 (1.63)	**-0.07** (0.11)	0.05 (0.06)
Stat *Hunger*	16.31 (1.87)	-0.05 (0.06)	**0.22** (0.15)
Dyn *Hunger*	13.01 (2.59)	-0.09 (0.12)	**0.11** (0.12)

Tabelle 8.7: ***Ergebnisse der Expertenagenten im realen Labyrinth mit dynamischen Objekten.*** *In einem realen Labyrinth mit jeweils einem dynamischen Objekt wurden sowohl ein mit dem dynamischen Objekt trainierter Experte ('Dyn') als auch ein mit dem entsprechenden statischen Objekt trainierter Experte ('Stat') getestet.*

Auf das Ausweichen des *Wildes* und entsprechend weniger Trainingserlebnisse bei gleichbleibender Trainingsschrittzahl ist zurückzuführen, daß der dynamische *Hunger*-Agent von Punkt 2 aus sehr selten ein Erfolgserlebnis verbuchen kann und es daher sehr viel seltener schafft als der statische Agent, das fliehende *Wild* zu erreichen. Der statische Agent fährt mit gewohnt langen Schritten darauf zu und braucht nur 2-3 Schritte mehr, um sie zu erreichen. Von Punkt 4 jedoch hat der dynamische Agent viel bessere Trainingschancen, die er auch genutzt hat. Beide Agenten fahren auf das fliehende *Wild*, jedoch benötigt der dynamische Agent diesmal 2 Schritte weniger als der statische Agent. Von den Punkten 5 und 6 schafft es wieder nur der statische Agent, wiederum mit etwas mehr Schritten, das fliehende *Wild* zu erreichen.

Ein ähnlicher Effekt tritt bei *Flucht* auf: während für den statisch trainierten Agenten ein energischer Schritt den Erfolg bedeutet, von der *Gefahrenzone* herunter zu fahren, tritt ein solcher Erfolg für den dynamisch trainierten Agenten erst nach 2-3 energischen Schritten ein. Andererseits braucht der statische Agent nicht unbedingt diese energischen Schritte und benutzt sie daher auch nicht konsequent. Entsprechend ähneln sich die Rewards, jedoch nicht die Policies: während der statische Agent nahezu immer die *Gefahrenzone* verlassen kann, aber dann langsamer flieht und wieder vom *Jäger* eingeholt wird, flüchtet der dynamische Agent seltener erfolgreich.

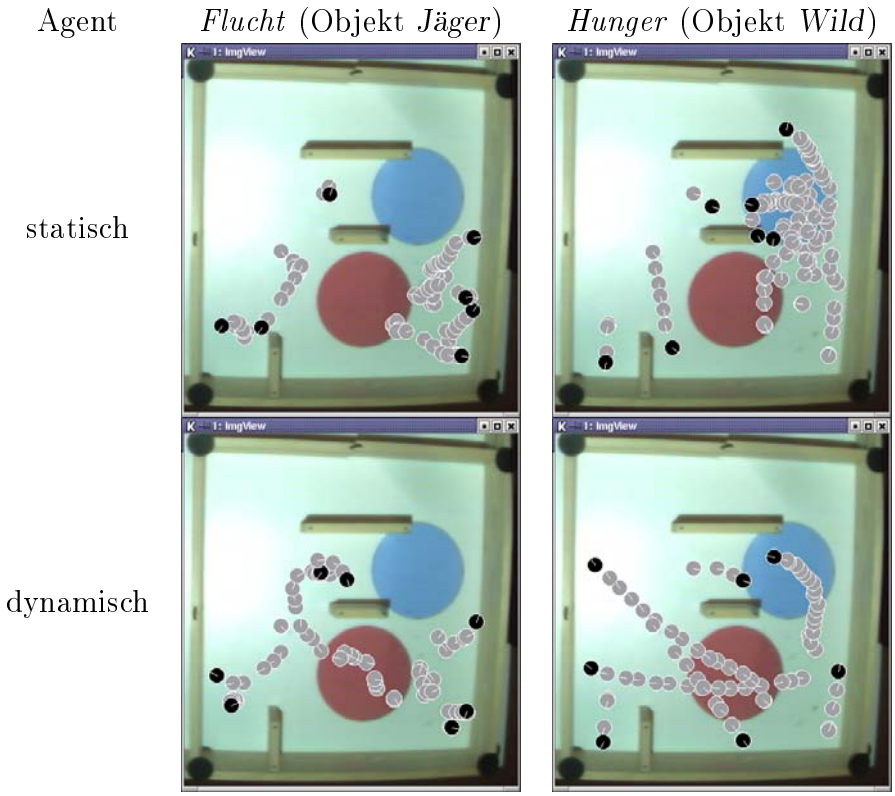

Abbildung 8.15: ***Gegenüberstellung des Verhaltens dynamischer und statischer Agenten im dynamischen realen Labyrinth***. *Die obere Reihe enthält die Fahrspuren der statisch trainierten Agenten im realen Labyrinth mit jeweils einem dynamischen Objekt: links* Flucht *vor Jäger, rechts* Hunger *nachWild. Die untere Reihe enthält die Fahrspuren der mit diesen dynamischen Objekten trainierten Agenten.*

8.2.4 Ergebnisse der Koordinationen mit dynamischen Objekten im realen Labyrinth

Wie bei den Untersuchungen in der simulierten Umwelt wurden auch im realen Labyrinth nur dieselben, ausgewählten Koordinationen mit dem dynamischen *Jäger* untersucht.

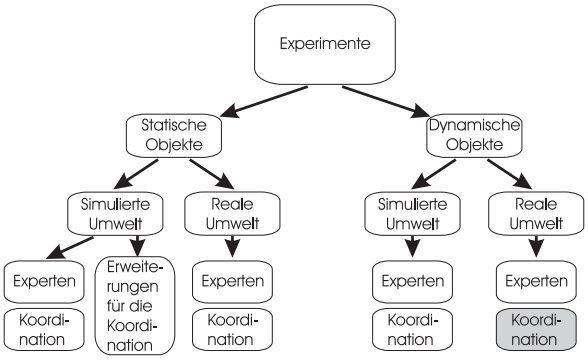

Abbildung 8.16: *In diesem Abschnitt werden Koordinationen im realen Labyrinth mit dynamischen Objekten untersucht.*

Tabelle 8.8 zeigt, daß sich die Leistungsunterschiede zwischen Monolith und Koordinationen nun nahezu verlieren. Nur bezüglich der Anzahl kollisionsfreier Schritte zeigt sich der Monolith noch überlegen.

Agent	Trialreward (σ)	Schritte (σ)
Mono Dyn *Flucht*	0.25 (0.26)	35.46 (4.96)
Negot WL Dyn *Flucht*	0.24 (0.21)	22.83 (3.73)
wl WL Dyn *Flucht*	0.25 (0.35)	16.51 (2.48)

Tabelle 8.8: ***Ergebnisse der Koordinationen mit dynamischen Agenten.*** *Mit dynamischem* Flucht-*Agenten und ansonsten den statischen Agenten im realen Labyrinth sind hier die Leistungen der Koordinationen ausgeglichen.*

Bei dieser Untersuchung gelang es nur dem Monolithen (Abb. 8.17, 2. v. links), von Punkt 2 aus auf die *Futterquelle* zu fahren. Dort allerdings wurde er vom *Jäger* eingeholt, versuchte, in den schmaler werdenden Streifen der *Futterquelle* zu bleiben und verursachte schließlich eine Kollision. Andererseits konnte nur Negotiated WL (Abb. 8.17 rechts) den Roboter von Punkt 3 auf die *Futterquelle* führen.

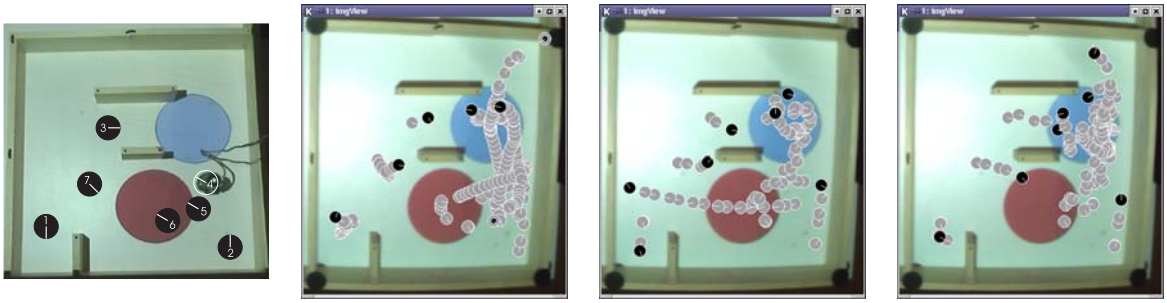

Abbildung 8.17: ***Verhalten der Koordinationen mit dynamischem Jäger im realen Labyrinth.*** *2. v. links das Verhalten des Monolithen, 3. v. links das des WL mit gelernten W-Werten und rechts das des Negotiated WL zum Vergleich.*

Die Leistungen der Koordinationen glichen sich jedoch an den Punkten 4-6 aus, von denen alle Koordinationen die *Futterquelle* erreichten, dafür jedoch jeweils unterschiedlich lange brauchten. Es lassen sich jedoch keine verallgemeinernden Aussagen über die verschiedenen Koordinationsvarianten treffen, da sie sich jeweils sehr unterschiedlich von den verschiedenen Punkten aus verhielten.

8.3 Erweiterungen des W-Lernens

Um die Nachteile der Koordination mit W-Lernen gegenüber einem Monolithen auszugleichen (Abschnitt 6.3.1), ohne die Vorteile eines MAS (Abschnitt 2.4.2) zu verlieren, wurden Modifikationen am W-Lernen untersucht, die in Abschnitt 6.3 dargestellt wurden. Alle Experimente wurden ausschließlich in der simulierten Umwelt ausgeführt, da deren Ergebnisse den Aufwand der realen Umsetzung als ungerechtfertigt erscheinen ließen.

In diesen Untersuchungen werden zuerst die Effekte eines gleichzeitigen Lernens der W-Werte der Koordination und der Q-Werte der Expertenagenten sowie einer besseren Exploration beim W-Lernen betrachtet. Die Arten der Exploration sind wegen der Ähnlichkeit des W-Lernens zu RL-Verfahren an die Explorationsverfahren im RL angelehnt. Weiterhin werden verschiedene Arten der Wichtung der Agenten bei der Ermittlung der W-Werte untersucht und der Gesamtreward zum Lernen der W-Werte herangezogen.

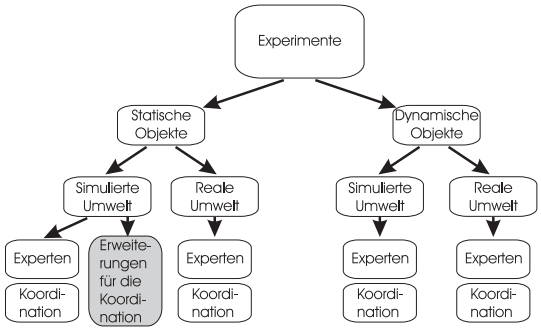

Abbildung 8.18: *In diesem Abschnitt werden Erweiterungen zur Koordination im simulierten Szenario mit statischen Objekten diskutiert.*

Die Untersuchungen werden mit der Variante 'Minimize Worst Unhappiness' mit lernenden W-Werten (siehe Abschnitt 5.2.1, Gl. 5.2) durchgeführt, da mit ihr ein Lernen durch Austesten der unterschiedlichen Leader möglich ist und sie auch eine zentrale Rolle unter den von HUMPHRYS diskutierten Methoden einnimmt. Bei dieser Variante schlagen alle Experten eine Aktion vor, und derjenige Agent, der den größten Verlust (den maximalen W-Wert) aufweist, wird der Leader, der Gewinner im aktuellen Zustand. Die anderen Experten passen ihre W-Werte an, indem sie beobachten, in welchen Folgezustand die Aktion des Leaders sie bringt. Die Differenz aus einerseits dem Q-Wert der vorgeschlagenen Aktion, und andererseits der Summe aus erfahrenem Reward und dem Wert des Folgezustands für den Experten wird zur Aktualisierung des W-Wertes benutzt.

8.3.1 Gleichzeitiges Anlernen der Expertenagenten

Im simulierten Szenario mit statischen Objekten wurden zunächst ohne besondere Vorkehrungen die Agenten und deren Koordinierung gleichzeitig trainiert.

Die Ergebnisse in Tabelle 8.9 sind mit denen in Tabelle 8.1, auch hinsichtlich eines zufällig handelnden Agenten zu vergleichen.

Agent	Schritte (σ)	Fahren (σ)	*Gefahrenzone*(σ)	*Futterquelle* (σ)
Bump	**23.14** (0.0)	**-0.67** (0.27)	-0.24 (0.18)	0.11 (0.17)
Flucht	37.14 (0.0)	-0.65 (0.32)	**-0.32** (0.32)	0.12 (0.18)
Hunger	31.00 (0.0)	-0.78 (0.25)	-0.23 (0.27)	**0.07** (0.11)

Tabelle 8.9: ***Ergebnisse der Expertenagenten nach gleichzeitigem Training mit der Koordination im simulierten Szenario.*** *Gleichzeitig zum Lernen der W-Werte wurden die einzelnen Agenten anhand ihrer jeweiligen Teilrewards trainiert.*

Die Kompetenz der Expertenagenten im Vergleich untereinander ist nicht ausgeprägt, was prinzipiell gegen die Vorgehensweise spricht. Zum großen Teil sollte das an den erheblich geringeren Trainingsmöglichkeiten liegen, die insbesondere verhindern, daß ein Expertenagent den Roboter über mehrere ganze Sequenzen kontrolliert. Dabei ist für den einzelnen Agenten das Training dem Vortraining aus einer Datei ähnlich: meist stimmt die ausgeführte Aktion nicht mit der durch den Agenten vorgeschlagenen überein. Würde eine solche Führungsmöglichkeit jedoch eingeräumt, unterschiede sich das Training kaum mehr vom Einzeltraining. Abbildung 8.19 stellt die Fahrspuren für die drei Expertenagenten dar.

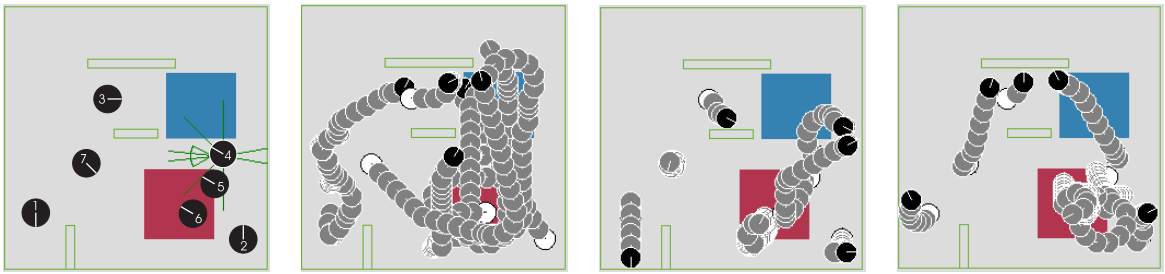

Abbildung 8.19: ***Fahrspuren der simultan antrainierten Agenten.*** *Die Fahrspuren der gleichzeitig mit dem W-Lernen antrainierten Agenten (2. v. links* Bump*, 3. v. links* Hunger*, rechts* Flucht*) sind von allen Einsetzpunkten aus dargestellt.*

Allerdings kann die Koordination ausführlich die Auswirkungen der verschiedenen Kombinationen der Agenten in einer Sequenz beobachten, was den gelernten W-Werten zugute kommen sollte. Solche unterschiedlichen Leader resultieren aus dem gleichzeitigen, aber ungleichmäßigen Training der Experten, auf das die Koordination sich stützt. Aus diesem Grunde wurden zwei Varianten des WL mit gelernten W-Werten untersucht. Zum einen die normale Variante, in der jeder Agent einen W-Wert für jeden seiner eigenen Zustände gegenüber jedem der anderen Experten aufbaut (agentenspezifischer Zustandsraum, *AZR*). Zum anderen wurde wegen der besseren Unterscheidung der verschiedenen Kombinationen von Agentenzuständen auch die Variante untersucht, in der jeder Agent einen W-Wert *für jeden Zustand des Gesamtsystems* gegenüber den anderen Experten aufbaut (voller Zustandsraum, *VZR*). Hierbei ergibt sich der Zustand des Gesamtsystems aus allen möglichen Kombinationen der Expertenzustände. Agent *AZR* baut also genau einen W-Wert für den Zustand des betreffenden Agenten auf, wo *VZR* mehrere W-Werte für die verschiedenen Kombinationen von Zuständen der anderen Agenten unterscheidet. Der W-Wert von *AZR* nimmt alle diese verschiedenen Werte in einem auf.

Koordination	Trialreward (σ)	Schritte(σ)
WL(AZR)	-2.24 (0.53)	31.04 (4.53)
WL(VZR)	-2.22 (0.90)	14.62 (3.65)

Tabelle 8.10: ***Ergebnisse des W-Lernen mit gleichzeitigem Training der Expertenagenten in der Simulation.*** *Hierfür wurden die Variante des Lernens der W-Werte gewählt. Es wurde sowohl mit agentenspezifischen Zustandsraum (*AZR*) als auch mit vollem Zustandsraum (*VZR*) experimentiert.*

Die Ergebnisse der Koordinationen in Tabelle 8.10 mit gelernten W-Werten im standard W-Lernen sind wesentlich schlechter als jene mit einzeln trainierten Expertenagenten. Dies liegt natürlich vor allem an den sehr schlechten Agenten. Auch der erhöhte Ressourceneinsatz für den vollen Zustandsraum kann daher keine Vorteile erreichen.

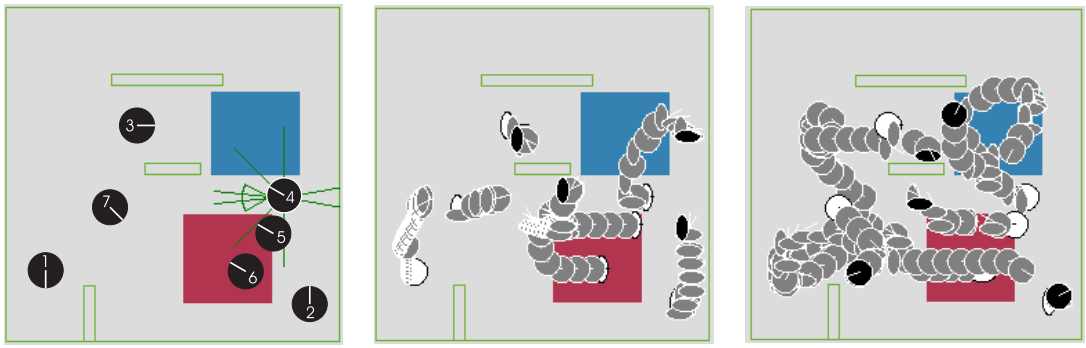

Abbildung 8.20: ***Verhalten der Koordination mit gelernten W-Werten nach gleichzeitigem Training der Experten.*** *Gleichzeitig mit ihrer Koordination mit gelernten W-Werten wurden die Expertenagenten trainiert. In der Mitte WL mit agentenspezifischem Zustandsräumen, rechts mit vollem Zustandsraum. Mit Vollkreis gezeichnete Schritte wurden durch* Bump*, hohe Ovale durch* Hunger *und flache Ovale durch* Flucht *veranlaßt.*

Von Punkt 1 aus steuert zunächst *Hunger* für *wlT* (Abb. 8.20 Mitte) den Roboter und läßt ihn rückwärts fahren. Von Punkt 2 aus dreht *Flucht* den Roboter schnell in die rechte Wand. Von Punkt 3 aus führt *Hunger* nahezu alle Schritte, mit einer Unterbrechung durch *Bump*, und steuert den Roboter in die untere Wand der Passage. Von Punkt 4 aus fährt *Bump* zunächst weg von der *Gefahrenzone* in Richtung der *Futterquelle*, und *Bump* stimmt mit *Flucht* überein, den Roboter in die rechte Wand zu fahren. Von Punkt 5 und 6 aus führt zunächst *Bump* den Roboter auf die *Gefahrenzone*, dann *Flucht* in die kurze Wand der Passage. Ähnliche Wechsel zwischen *Bump* und *Flucht* bestimmen das Verhalten von Punkt 6, ohne den Roboter von der *Gefahrenzone* weg zu bringen. Von Punkt 7 aus dreht *Hunger* den Roboter in Richtung der kurzen Wand der Passage und läßt ihn dort vorfahren, während *Bump* ihn bis zur maximalen Schrittanzahl vor einer Kollision bewahren kann. WL mit vollem Zustandsraum (Abb. 8.20 rechts) kann die Erwartungen nicht erfüllen, da es aufgrund sehr häufiger Wechsel keine Policy wirklich verfolgen kann und insgesamt schlechter abschneidet als WL mit agentenspezifischem Zustandsraum.

Insgesamt ist leider festzustellen, daß sich kein Experte an den jeweils relevanten Situationen durchsetzen konnte und auch keine erfolgreiche Koordinierung gefunden wurde.

8.3.2 Exploration im W-Lernen

Beim Lernen der W-Werte in direkter Interaktion mit der Umwelt, also bei den Varianten mit gelernten W-Werten, können die Agenten zu Anfang zufällig Rewards aus obersten Regionen der Rewardverteilung erleben und sehr hohe, aber ungerechtfertigte W-Werte aufbauen. Schon in Abschnitt 4.5 wurde die Wichtigkeit einer Exploration im RL erläutert, und Lernen von W-Werten stellt auch eine Art des RL dar.

Eine optimistische Initialisierung und eine Boltzman-Auswahl der aktionsentscheidenden W-Werte sorgt für eine Exploration der verschiedenen Gewinner im Wettstreit der Experten um die Aktionsausführung in einem bestimmten Zustand. Die unterlegenen Agenten können so erfahren, welche Konsequenzen die verschiedenen Gewinner tatsächlich für sie haben, und die verschiedenen Gewinner werden tatsächlich getestet.

Sie können so lernen, Synergien auszunutzen, die sie mit einer strikten Auswahl des Gewinners nie erfahren würden. Sollte es Korrelationen der verschiedenen Rewardquellen in der Umwelt geben, würden diese hiermit auch erfahren werden. Allerdings beschränken sich im gegenwärtigen Szenario solche Korrelationen auf die selten vorkommenden Überdeckungen zwischen einem dynamischen und einem statischen Objekt. Eine solche Überdeckung liegt zum Beispiel vor, wenn der Roboter auf die *Futterquelle* fährt und vom dynamischen *Jäger* eingeholt wird, der sich über das Futter schiebt.

Eine optimistische Initialisierung bedeutet eine Initialisierung der W-Werte mit sehr geringen Verlusten, sodaß jeder Experte mindestens einmal Leader wird, da alle Agenten außer dem Leader ihre W-Werte aktualisieren. Eine Boltzman-Auswahl unter den W-Werten erzeugt ein Abweichen vom größten W-Wert umso häufiger, je geringer die Unterschiede zwischen den W-Werten und je höher die Boltzman-Temperatur ist.

Die beiden Explorationsarten wurden im simulierten Szenario mit der WL-Variante mit lernenden W-Werten trainiert und getestet. Die in Tabelle 8.11 dargestellten Ergebnisse zeigen allerdings keine Vorteile für die explorierenden Modifikationen des W-Lernens.

Agent	Trialreward (σ)	Schritte(σ)
wlWL OptInit	-0.68 (0.61)	33.40 (4.04)
wlWL Boltz AZR	-1.63 (0.02)	28.55 (0.21)
wlWL Boltz VZR	-0.06 (0.11)	28.25 (2.19)

Tabelle 8.11: ***Ergebnisse des W-Lernen mit Exploration im simulierten Labyrinth.*** *Die W-Werte wurden unter Verwendung einer Exploration, der optimistischen Initialisierung 'OptInit' oder der Boltzman-Exploration 'Boltz' gelernt. Wiederum wurden agentenspezifischer und voller Zustandsraum für die W-Werte (VZR) unterschieden*

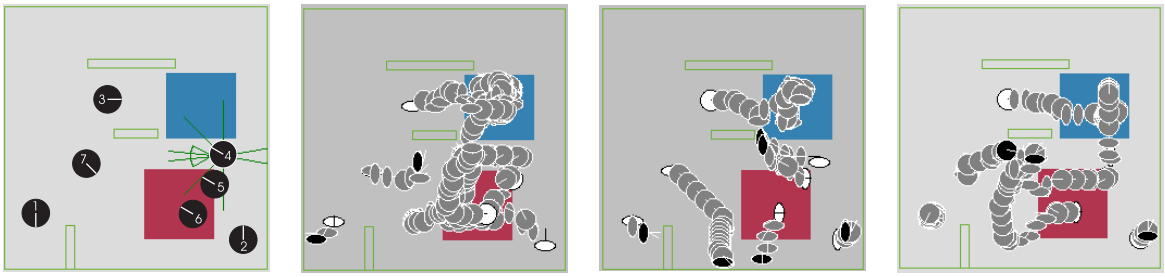

Abbildung 8.21: ***Verschiedene Arten der Exploration im WL.*** *Die W-Werte im W wurden mit verschiedenen Arten der Exploration gelernt: 2. v. links mit einer optimistischen Initialisierung, 3. v. links mit der Boltzman-Exploration und agentenspezifischem Zustandsraum, rechts mit Boltzman-Exploration und vollem Zustandsraum. Mit Vollkreis gezeichnete Schritte wurden durch* Bump, *hohe Ovale durch* Hunger *und flache Ovale durch* Flucht *veranlaßt.*

Bei den W-Werten, die mit einer Exploration gelernt wurden, entsteht im Gegensatz zu den simultan gelernten Expertenagenten der Trend, daß ein Agent größere Teilstücke oder die gesamte Sequenz übernimmt. Dies gilt für WL mit gelernten W-Werten und Boltzman-Exploration mit vollem Zustandsraum (Abb. 8.21 rechts) von Punkt 1, wo *Bump* den Roboter praktisch auf der Stelle hält. Von Punkt 2 streiten sich *Bump* und *Flucht* und fahren den Roboter nach wenigen Schritten in die rechte Wand. Erfolgreich ist diese abwechselnde Strategie jedoch von Punkt 3, wo *Hunger* startet und *Bump* den Roboter auf das *Wild* fährt und dort sicher hält. Die gleiche Koordinierungstaktik ist von Punkt 4 nahezu optimal erfolgreich, nur startet diesmal *Flucht* und *Bump* hält den Roboter lange allein auf der *Futterquelle*. Von Punkt 5 allerdings fährt *Bump* den Roboter erst durch die *Gefahrenzone*, bevor er durch *Hunger* umkehrt und mit der kurzen Wand der Passage kollidiert. Von Punkt 6 aus wechseln sich *Flucht* und *Bump* ab, und fahren den Roboter zwar von der *Gefahrenzone* herunter, halten ihn jedoch sehr nah daran. Von Punkt 7 übernimmt *Bump* die Kontrolle und läßt den Roboter im Freiraum drehen bis zur Kollision mit der kurzen Wand der Passage.

Obwohl die Exploration mit der Boltzman-Auswahl bessere Ergebnisse erreicht als die mit einer optimistischen Initialisierung, kann nur WL mit vollem Zustandsraum die Leistung des WL mit gelernten W-Werten ohne Exploration erzielen oder verbessern.

8.3.3 Zustandsabhängige Priorisierung der Agenten

Die Analyse des Verhaltens der Koordinationen zeigte, daß eine einzige Wichtung für einen Agenten in der Koordinierung zu unflexibel sein könnte, um den maximalen Gesamtreturn zu erzielen. Daher werden zwei Arten der zustandsabhängigen Wichtung der Agenten untersucht. Die Wichtungen der Agenten ergeben sich aus der absoluten Größe ihrer Q-Werte, also letztlich aus den Größen der verteilten Rewards.

Bei der ersten ergeben sich die Wichtungen der Agenten aus der Relevanz der von ihnen vorgeschlagenen Aktion für den Agenten. Diese Methode greift die von Humphrys als 'statische W-Werte' eingeführten Berechnungen auf und ermittelt nicht den W-Wert, sondern seine Wichtigkeit für den Agenten im aktuellen Zustand. Hier wurde die Methode der Abweichung vom durchschnittlichen Q-Wert angewendet [Humphrys, 1997].

Da alle benötigten Werte statisch sind und zum Start der Koordinierung vorliegen, entsprechend kein Lernen nötig und alle Agenten dieselbe Auswahl an Aktionen haben, wurde hier 'Negotiated' W-Lernen eingesetzt, bei dem die W-Werte nicht gelernt werden, sondern für alle Experten aus den Q-Werten berechnet werden.

Auch diese Erweiterung des W-Lernen zeigt in Tabelle 8.12 nicht die erhoffte Leistungssteigerung für die Koordination. Im Gegenteil, die Leistungen sind erheblich schlechter.

Agent	Trialreward (σ)	Schritte (σ)
negot WL	-0.85 (0.03)	36.91 (0.44)

Tabelle 8.12: ***Ergebnis des W-Lernens mit Relevanzwichtung der einzelnen Agenten im simulierten Labyrinth.*** *Die Relevanz einer Aktion und damit die statische Wichtung für einen Experten ergibt sich aus der Abweichung des Q-Wertes vom Durchschnitt seiner Q-Werte.*

Von Punkt 1 aus ist diese Agentenwichtung (Abb. 8.22, 2. v. links) mäßig erfolgreich: *Bump* bestimmt den überwiegenden Anteil der Schritte und fährt kleine Kreise. Seltene Unterbrechungen durch *Flucht* führen zur Kollision. Von Punkt 2 aus jedoch übernimmt *Flucht* die Führung, und wechselt sich mit *Bump* ab, den Roboter zwar von der *Gefahrenzone* wegzuhalten, aber schnell zu kollidieren. Von Punkt 3 aus ist die Koordination wieder sehr sinnfällig und erfolgreich, *Hunger* und *Bump* steuern jeweils 4-5 Schritte den Roboter geradeaus durch die Passage und auf die *Futterquelle*, wo sie ihn halten. Von Punkt 4 aus kommt *Hunger* zum Zuge und fährt den Roboter auf die *Futterquelle*. Von Punkt 5 aus beginnt *Flucht*, aber dann geht die Führung an *Hunger*, der den Roboter am Rand der *Futterquelle* führt und mit der unteren Wand der Passage kollidiert. Von Punkten 6 und 7 schließlich kontrollieren *Bump* und *Flucht* abwechselnd die Bewegungen und halten den Roboter auf der *Gefahrenzone*. Insgesamt hat *Bump* offensichtlich die stärkste Varianz in den Q-Werten seiner Zustände und dominiert so den Wettbewerb.

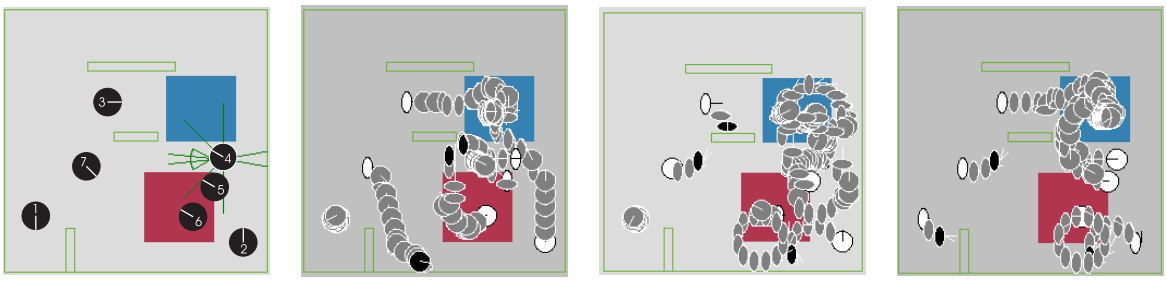

Abbildung 8.22: ***Verhalten von WL mit verschiedenen Agentenwichtungen.*** *Verschiedene Arten der Wichtung der Experten wurden verglichen: bei 2. v. links aus der Relevanz der vorgeschlagenen Aktion berechnet, bei 3. v. links aus dem Gesamtreward mit optimistischer Initialisierung, rechts auch aus dem Gesamtreward mit der Boltzman-Exploration gelernt.*

Bei der zweiten, adaptiven Wichtung der Agenten wird der Gesamtreward genutzt, um die Wichtungen von Agenten so zu verändern, daß der Gesamtreturn maximiert wird. Für jeden Zustand, den das Gesamtsystem einnimmt, wird die Differenz zwischen dem Gesamtreward in diesem Versuch und beim letzten Versuch berechnet und, mit einer Lernrate beaufschlagt, als Änderung der Wichtung des Leaders verwendet (siehe Gl 6.4 in Abschnitt 6.3.5). Die Experimente wurden wieder mit gelernten W-Werten durchgeführt, die zum Erfahren verschiedener Leader und deren Auswirkungen auf den Gesamtreward mit den beiden schon in Abschnitt 8.3.2 untersuchten Explorationen trainiert wurden.

Die Ergebnisse in Tabelle 8.13 belegen, daß auch diese Modifikation am W-Lernen nicht die vorgestellten Leistungssteigerungen erbringt. Zwar sind die Leistungen besser als die der statischen Relevanzwichtung, aber doch schlechter als die des normalen W-Lernen.

Agent	Trialreward (σ)	Schritte (σ)
wlWL Opt Init	-2.43 (0.54)	27.47 (3.62)
wlWL Boltz	-0.68 (0.37)	28.55 (3.17)

Tabelle 8.13: ***Ergebnisse des W-Lernen mit adaptiver Wichtung der einzelnen Agenten im simulierten Labyrinth.*** *Im WL mit gelernten W-Werten wurden die Wichtungen der Agenten aus dem Gesamtreward unter Verwendung zweier Explorationen gelernt. Dazu wurden wieder die optimistische Initialisierung 'OptInit' und die Boltzman-Auswahl 'Boltz' verwendet.*

Bei den Agentenwichtungen, die aus dem Gesamtreward gelernt werden, fällt die Mehrzahl der Schritte an *Hunger* (Abb. 8.22, 3. v. links und rechts). Entsprechend fährt *Hunger* den Roboter von Punkt 1 aus nach wenigen Schritten an die Wand, von Punkt 2 aus in eine Drehbewegung nach links über die *Gefahrenzone* an die untere Wand des Szenarios, von Punkt 3 aus jedoch mit einigen Schritten der Unterstützung durch *Bump* direkt auf die *Futterquelle*, wie auch von Punkt 4. Von Punkt 5 aus kehrt sich das Aktivierungsmuster um, nun beherrscht *Bump* die meisten Schritte und *Hunger* übernimmt den Anfang, um den Roboter auf die *Futterquelle* zu führen, wo *Bump* ihn hält. Von Punkt 6 aus regiert *Hunger* allein und fährt den Roboter in einem großen Kreis über die *Gefahrenzone* an die untere Wand des Szenarios und wieder über die *Gefahrenzone*. Auch von Punkt 7 fährt *Hunger* den Roboter durch eine große Linksdrehung an die untere Wand der Passage

8.3.4 Direktes Lernen aus dem Gesamtreward

Die Methoden des W-Lernens können danach unterschieden werden, ob sie die Auswahl eines Agenten, oder aber direkt die Auswahl einer kollektiv optimalen Aktion zum Ergebnis haben. Die ausgewählten Agenten würden dann die von ihnen bevorzugte Aktion ausführen. Da es sich um austrainierte Q-Lerner handelt, wählen sie jeweils ihre beste Aktion, die sich auch nicht mehr ändert. Damit sucht W-Lernen insgesamt nach einer optimalen Aktion. Unter Verwendung des Gesamtrewards kann diese mit den in Kapitel 4 erläuterten Methoden direkt gelernt werden. Die hier getestete Methode wurde in Abschnitt 6.3.4, Gl. 6.1 entwickelt. Mit Hilfe des Gesamtrewards und des aus den Agentenzuständen ermittelten Gesamtzustandes wurde durch ein dem *Sarsa* ähnlichen Algorithmus ein Zustands-Aktionswert für das Gesamtsystem gelernt und der resultierende Agent wie alle anderen getestet.

Diese Erweiterung des W-Lernens erzeugt, wie die Zahlen in Tabelle 8.14 und Abb. 8.23 zeigen, keine Leistungssteigerung des W-Lernens, aber auch keine Verschlechterung. Weitere Untersuchungen sollten hier bezüglich der Ausnutzung der Varianten des W-Lernens erfolgen, die als Gewinn oder Verlust in Gl. 6.1 einfließen können. In den hier vorgestellten Untersuchungen wurden sie beide im Sinne einer grundlegenden Untersuchung als 0.0 angenommen worden.

Agent	Trialreward (σ)	Schritte (σ)
WQ	-1.33 (0.00)	22.57 (0.00)

Tabelle 8.14: *Ergebnis des W-Lernen direkt als Reinforcement Lernen im simulierten Labyrinth.*

Diese Erweiterung schafft es, von Punkt 1 aus eine große Anzahl von Schritten zu fahren. Allerdings führt sie den Roboter von den anderen Punkten aus in das sichere Verhalten der kleinen Bewegungen, welches unter Umständen auf der *Futterquelle* zu einigem Reward, auf der *Gefahrenzone* aber zu Strafen führt. Der erhoffte zielgerichtete Einsatz der Experten zur Maximierung des Gesamtreturns konnte so nicht erreicht werden.

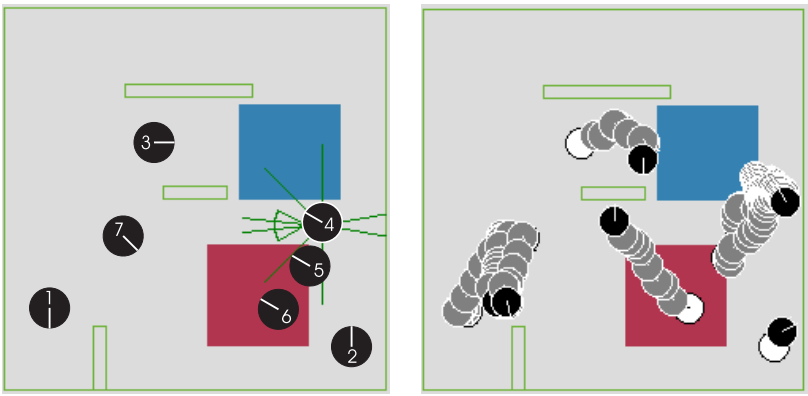

Abbildung 8.23: *Die* ***Koordination nach direktem Lernen aus dem Gesamtreward*** *kann ebenfalls kein besseres Verhalten erzeugen.*

8.4 Zusammenfassung und Wertung

In diesem Kapitel wurden die Ergebnisse eines lernenden Erwerbs von Verhalten mit Reinforcement Lernen in einem aus Hindernissen sowie aus Quellen positiven und negativen Rewards bestehenden Szenario dokumentiert. Die Quellen wurden in einer zweiten Stufe des Ausbaus dynamisch gestaltet und die entsprechenden Leistungen der Expertenagenten und der Koordinationen ebenfalls festgehalten.

Die Experten konnten die von ihnen erwartete Kompetenz erwerben und beweisen und die Möglichkeit einer Koordinierung der Experten durch W-Lernen auf einem realen Roboter wurde demonstriert, wobei einige Varianten, insbesondere 'Minimize Worst Unhappiness', gute Ergebnisse erzielten. Damit konnten die eingesetzten Eingangsinformationen mit ihrer Vorverarbeitung und die gewählten Rewardfunktionen als sinnvoll bestätigt werden. Die Wahl der Agentenarchitektur unterstützte sehr die Interpretation und Auswertung der Verhalten.
Weiterhin erwies sich auch die Zweiteilung des Trainings der Agenten im realen Labyrinth als effektiv. Der Erfolg der Koordinationen hängt großteilig von den Agenten ab, sodaß bessere Agenten auch bessere Leistungen der Koordinationen ergeben. Im einheitlichen Rahmen von W-Lernen konnte einer Fülle unterschiedlicher Ansätze auf der Grundlage derselben Experten untersucht werden.
Mit komplexer werdenden Aufgaben durch dynamische Objekte verschlechtern sich die Leistungen sowohl des Monolithen, aber auch der Koordinationen, obwohl letztere einen geringeren Leistungsabfall zeigen. Die mit den dynamischen Objekten trainierten Experten zeigten jedoch teilweise sehr interessantes Verhalten. Zum einen bestätigt das das generelle Interesse an dynamischen Objekten und rechtfertigt den zu ihrer Erzeugung und Steuerung betriebenen Aufwand. Zum anderen bestätigt es auch die Unterschiede zwischen Erreichungs- und Vermeidungsaufgaben bei komplexer werdenden Aufgaben. Es unterstreicht auch die Leistungsfähigkeit von Reinforcement Lernen beim Erwerb emergenter Verhalten. Es bestätigt auch den schrittweisen Ausbau des Szenarios, da solche Verhalten nicht vorherzusehen waren.
Ein gleichzeitiges Training der Experten mit ihrer Koordination sowohl hinsichtlich der Leistung der Experten als auch der Koordination ist nicht ohne Leistungseinbußen oder

weitere Vorkehrungen möglich. Daher sollte auch diese Problematik in weiterführenden Arbeiten eingehend betrachtet werden.

Die in Kapitel 6 geäußerten Vermutungen hinsichtlich einer Überlegenheit des Monolithen gegenüber der Koordination, hier durch WL, konnten bestätigt werden. Als Ursachen wurden vor allem die unflexible Wichtung der Experten und die fehlende implizite Planung im WL herausgestellt. Letztere stammt aus der Auswahl eines Agenten genau für den nächsten Schritt, aber ohne Betrachtung der Optimalität für folgende Aktionen. Der Agent selber verfügt über implizite Planung und wählt eine ihm dienende Aktion, die jedoch dem Gesamtsystem nicht optimal dienen muß. Entsprechende Vorschläge zur Verbesserung des WL konnten die genannten Probleme jedoch nicht lösen. Entsprechend sind eingehende Untersuchungen notwendig, die vermutlich das Szenario systematisch von einfachsten Aufgaben beginnend aufbauen und dabei einen Monolithen und eine Koordination parallel betrachten.

Weitere, generellere Überlegungen betreffen die Unterteilung der Gesamtaufgabe in Teilaufgaben durch den Designer.
Erfolgt diese nach inhaltlichen Gesichtspunkten des Designers, also zum Beispiel die Gesamtaufgabe 'Überleben' in die inhaltlichen Teilaufgaben 'Essen', 'Trinken', 'Reproduzieren' und 'Feindvermeiden', so ist hier das Prinzip der Koordination des Verhaltens nicht dem Prinzip der Unterteilung der Gesamtaufgabe angemessen. Ein hartes Umschalten zwischen den einzelnen Verhalten, je nach Situation, befriedigt weder die einzelnen Agenten, die nach ihrem Training davon ausgehen, ihre Policy jeweils realisieren zu können, noch erfüllt es die Gesamtaufgabe optimal. Die Agenten können ihre Policy nicht umsetzen, da gleichzeitig auch alle anderen Aufgaben erfüllt werden sollen.

Wurde die Gesamtaufgabe jedoch nach Kriterien des Eingaberaumes unterteilt, sodaß sich für einen gegebenen Zustand im Zustandsraum des Gesamtsystems immer genau ein Agent findet, der wesentlich besser als alle anderen befähigt ist, das System zu leiten, wie es zum Beispiel durch eine der in Abschnitt 4.7 diskutierten hierarchischen Unterteilungen geschieht, so tritt dieser Mangel nicht auf: die Dekomposition der Aufgabe ist der Rekomposition des Verhaltens genau angemessen. Daraus folgt eine möglichst eigenständige Dekomposition der Gesamtaufgabe in Teilaufgaben, entsprechend dem Zustandsaktionsraum und der Rekomposition der Teilaufgaben. Entsprechende Ansätze finden sich neben der erwähnten hierarchischen Unterteilung, die jedoch wie *Subsumption* sogenannte Heterarchien ausschließt und so nur einen Teil möglicher Lösungen betrachtet, zum Beispiel in den Arbeiten von Digney [Digney, 1996, Digney, 1998]. Sie gehen von zwei Kriterien zur Herausbildung von Teilaufgaben aus, die den Eingaberaum unterteilen: einerseits häufiger Aufenthalt in bestimmten Unterräumen des Eingaberaumes, andererseits hohe Rewardgradienten, die einem Unterraum besondere Bedeutung verleihen. Alternative Kriterien wären die Minimale Beschreibungslänge (Minimal Description Length, MDL) für eine Aufgabenzerlegung oder die in Abschnitt 4.8 erwähnten Arbeiten zur Herausbildung von Makroaktionen.

Eine sinnvolle Aufgabenzerlegung ist jedoch nur eine der notwendigen Funktionen bei der Nachbildung tierischer Intelligenz durch verhaltensbasierte Synthese. Weitere notwendige Weiterentwicklungen betreffen die Realitätstauglichkeit der eingesetzten Agenten hinsichtlich eines schnellen Lernens und der Behandlung des POMDP-Problems. Wichtig auf dem Weg zum Verständnis eines Tiers ist jedoch die Betrachtung von dessen Handlungs-

antrieben, wie sie in Abschnitt 2.1 diskutiert wurden. Sie könnten eine Weiterentwicklung der Koordination verschiedener Experten oder Kompetenzen des Animats zu einer nicht nur räumlich variablen Koordination, wie sie in Abschnitt 8.3.3 angedacht wurde, sondern auch zu einer zeitlich variablen, also dynamischen Koordination erlauben. Diesen Fragestellungen widmet sich das nächste Kapitel.

Kapitel 9

Weiterführende Untersuchungen

Die Zielsetzung, Roboter in realen Umwelten selbstständig intelligentes Verhalten finden zu lassen, macht über die in Kapitel 8 vorgestellten Untersuchungen hinaus weitere Untersuchungen in zwei Hauptrichtungen nötig. Diese Untersuchungsrichtungen sind mit unterschiedlichen Zielsetzungen bezüglich des Einsatzes von Multiagentensystemen gekoppelt. Die in Kapitel 8 deutlich gewordene Notwendigkeit einer Weiterentwicklung des W-Lernens bzw. die Betrachtung anderer Koordinationsansätze sollte diese weiteren Ziele neben dem Leistungsvergleich zu einem Monolithen verfolgen.

Zum einen handelt es sich dabei um die Fortführung der Entwicklung der einzelnen Agenten und deren Koordination für eine reale Anwendung in einer natürlichen Umgebung. Ausgehend von dieser Zielsetzung müssen vor allem Anforderungen einer komplett unkontrollierten Umwelt betrachtet werden, sowie die Optimalität der eingesetzten Teilalgorithmen stärker beachtet werden.

Unter dem anderen Gesichtspunkt einer Erforschung der Prinzipien tierischer und menschlicher Entscheidungsfindung und Verhaltenserzeugung sollte zunächst die verwendete Umwelt erweitert werden und zeitlich variable Handlungsantriebe, wie sie dem Verhalten von Tieren zugrundeliegen, eingeführt werden. Solche Handlungsantriebe oder Bedürfnisse können zusammen mit den bislang betrachteten Ansätzen aus dem RL eingesetzt werden, aber auch mit alternativen, biologisch motivierten Ansätzen des Reinforcement Lernens.

9.1 Erweiterungen am Animat

9.1.1 Zeitlich variable Ziele: Bedürfnisse

Mit der Einführung von Bedürfnissen für das Animat können realistischere Szenarien erzeugt werden, die komplexeres Verhalten erfordern, das sich einem intelligent zu nennenden Verhalten weiter nähert.

Mit solchen Bedürfnissen kann auch eine Brücke geschlagen werden vom bisher betrachteten Reinforcement Lernen, wie es im Maschinenlernen verwendet und auch in technischen Applikationen eingesetzt wird, hin zu ethologisch oder psychologisch motivierten Ansätzen wie dem von Dörner [Dörner, 1999] (siehe Abschnitt 5.3.1).

Setzt man grundlegende biologische Mechanismen voraus, so können Bedürfnisse als Basis der homöostatischen Regelung des Überlebens von Tieren und damit als Ausgangspunkt von Handlungsantrieben angesehen werden. Verallgemeinernd kann man davon ausgehen, daß die Lebenserhaltungssysteme jeweils Optimalpunkte aufweisen, die einen erstrebenswerten Zustand für das Animat darstellen, z.B. einen ausreichenden Blutzuckerspiegel oder ein angenehmes Wärmegefühl. Weicht der aktuelle Wert von diesem Optimalpunkt ab, entsteht ein Mangel und folglich ein Bedürfnis, diesen Mangel auszugleichen, z.B. zu fressen, um den Blutzuckerspiegel zu heben. Neben einem Optimalpunkt weisen solche Bedürfnisse auch eine Sättigung am oberen und unteren Extrem auf. Entsprechend sollte es dem Animat keinen Reward eintragen, zu fressen, wenn es schon satt ist. Gleichung 9.1 ist als Entwurf für die Berechnung eines solchen Bedürfnisses B aus einem Antrieb T zu verstehen.

$$\begin{aligned} B &= f(T_{opt} - T) \\ \Delta T &= \alpha * (dazu - weg) \\ dazu &= \beta * (Max_T - T) * (zufuhr + MinZufuhr) \\ weg &= \delta * (T - Min_T) * (abfuhr + MinAbfuhr) \end{aligned} \tag{9.1}$$

Ein Bedürfnis entsteht aus der Abweichung eines Antriebes T von seinem Optimum T_{opt}. Die Größen $zufuhr$ und $abfuhr$ beinhalten die äußeren Einflüsse; z.B. nimmt das Tier auf einer Futterquelle pro Zeiteinheit eine Anzahl Einheiten Futter mit spezifischem Nährwert auf, die insgesamt $zufuhr$ ergeben. Davon werden $dazu$ Einheiten wirksam, die von der Differenz zwischen aktuellem Stand T des Antriebes (hier: Energie) und seinem Maximum Max_T abhängen. Damit werden weniger Einheiten aufgenommen, je näher der Stand dem Maximum kommt. β dient der Modellierung einer Aufnahmerate (hier: Effizienz der Verdauung) und Normierung. Die Rate kann vom aufgenommenen Futter oder dem Zustand des Tieres (Schläfrigkeit, Durst etc.) abhängen. Die Größe $MinZufuhr$ beinhaltet eine Zufuhr, die unabhängig von den Aktionen des Animats ist. Der Sinn einer solchen Größe wird deutlicher an der negativen Entsprechung $MinAbfuhr$, die im Beispiel die permanente Abfuhr von Energie durch Lebensprozesse des Animats modelliert (Denken, Verdauung selber o.ä.). Die anderen Größen, die in $abfuhr$ eingehen, sind analog den Größen für $zufuhr$ zu interpretieren und modellieren einen Energieverlust durch Zusammenstoß mit einem Feind, oder durch die Ausführung von Aktionen etc. Ähnliche Definitionen für Bedürfnisse finden sich auch bei Grossberg [Carpenter and Grossberg, 1991], Dörner [Dörner, 1999] und anderen.

Die bisher implementierten RL-Algorithmen mit einer Ableitung des Rewards und des aktuellen Zustandes allein aus den sensorischen Eingangsgrößen können als fortwährend hungrig interpretiert werden. Ein Ziel wird immer verfolgt, ohne Rücksicht darauf, wie oft es schon erfolgreich realisiert wurde.

Um solch unnatürliches Verhalten zu vermeiden, sollte einerseits der Reward für die Lernalgorithmen also nicht der aktuelle Stand T eines Antriebes sein, sondern die Änderung $(dazu - weg)$, die in dem Antrieb durch die Aktion ausgelöst wurde. Frißt das Animat, wenn es großen Hunger hat, erhält es aus einer aufgenommenen Einheit Futter viel Reward. Frißt es ohne Hunger, erhält es aus derselben Einheit Futter nur wenig oder keinen Reward. Andererseits muß dem Lernalgorithmus ermöglicht werden, die Ursache

der unterschiedlichen Rewards zu finden. Dazu liegt es nahe, den Stand der Antriebe in die Merkmale aufzunehmen, die den Zustand des Tieres definieren, also den Zustandsraum des Lernalgorithmus um die Bedürfnisse zu erweitern.

Es soll dem Tier gutgehen, wenn alle seine Bedürfnisse im Optimum liegen. Also lassen sich die Stände aller Antriebe gut zu einem Gesamtreward zusammenfassen, der beim Lernen einer optimalen Koordination der Verhalten verwendet werden kann.

Eine Verwendung solcher Bedürfnisse ist also mit den bisher verwendeten RL-Algorithmen und einigen sinnvollen Definitionen des Zustandsraumes und der Rewardfunktionen möglich. Aus der zeitlichen Varianz der Bedürfnisse ist eine Erhöhung der beobachtbaren Komplexität des entstehenden Verhaltens zu erwarten. Daher wurde für die grundsätzlichen Untersuchungen der Koordination mit W-Lernen in Kapitel 8 zunächst von den üblichen RL-Algorithmen mit unveränderlichen Bedürfnissen ausgegangen. Das von WL mit dynamischen Bedürfnissen zu erwartende Verhalten sollte für 'Best Happiness' ähnliche dem in Abschnitt 5.3.3.1 diskutierten eine Aneinanderreihung der Ziele erzeugen.

In der folgenden Auflistung wird eine Anzahl möglicher existentieller Antriebe mit den Funktionen gezeigt, die sich mit ihnen demonstrieren lassen sollten. Mit diesen Antrieben sollte ein realistischeres Szenario mit einem Animat gestaltet werden können, das in der Natur überleben will. Alle folgenden Bedürfnisse sollten aus der grundsätzlichen Gleichung 9.1 umgesetzt werden können.

- **Hunger/Energie:** Energie wird immer verbraucht (auch im Ruhezustand). Aktionen verbrauchen mehr Energie, schnellere Aktionen noch mehr und Begegnungen mit Feinden am meisten. Damit ist ein primärer Handlungsantrieb (Hunger: Energie nachladen) und Antrieb zu effizientem Handeln gegeben. Es kann verschiedene Quellen von Energie geben, die bezüglich Kosten und Reward gegeneinander abgewogen werden müssen, z.B. eine statische Quelle mit wenig Reward und eine bewegliche Quelle mit mehr Reward, aber auch größerem Aufwand, sie zu fangen.

- **Durst:** Das Animat muß auch Flüssigkeit aufnehmen. Dieser Antrieb muß gegen den ersten abgewogen werden. Sind die beiden Bedürfnisse mit sehr unterschiedlichen Zeitkonstanten (α in Gl. 9.1) implementiert, so kann dies zu interessanten kognitiven Anforderungen führen. Steigt der Durst sehr viel schneller an als der Hunger, muß das Animat seinen Weg zum Futter mit Tränken planen, um zwischendurch nicht zu verdursten, oder einen Vorrat anlegen und mitführen. Im Sinne einer Rewardsubstitution (siehe folgende Liste) hat Wasser auch einen, aber sehr geringen Futterwert und das Futter auch ein wenig Flüssigkeit.

- **Schmerz:** Das Animat hat das Bedürfnis, seine körperliche Integrität zu erhalten. Begegnungen mit Jägern, Hindernissen oder Gefahrenzonen verursachen Schmerz, solche mit ebenfalls möglichen unterstützenden Gebieten lindern ihn. Neben optimaler Wegplanung demonstrieren solche Objekte, wie die Risikofreudigkeit des Animats von seiner Befriedigung über alle Bedürfnisse abhängt: der Energiezustand hat die Bevorzugung bzw. Unterdrückung bestimmter Aktionen zur Folge.

- **Schlaf:** Es gibt einen 'Standby' oder Schlafzustand, in dem weniger Energie verbraucht wird, wegen eingeschränkter Sensorik aber auch Feinde später wahrgenom-

men werden. Es gibt ein 'Schlafbedürfnis' derart, daß sich z.B. die maximal erreichbare Geschwindigkeit umgekehrt abhängig zum Schlafbedürfnis B_s verringert (z.B. $|v_{max}| = 1 - \frac{1}{1+\exp(-B_s)}$), aber beim Schlaf nach einem T_1-Verhalten regeneriert wird.

In den Untersuchungen in Kapitel 8 wurden die kognitiven Anforderungen an das Animat bewußt gering gehalten (siehe Kapitel 6), um die erzeugten Verhalten und deren Koordination sicher erzeugen und bewerten zu können. Im Interesse der Erzeugung intelligenten Verhaltens sollte diese Einschränkung jedoch schließlich fallengelassen werden und das Animat könnte mit folgenden kognitiven Bedürfnissen ausgestattet werden:

- **Neugier:** Das Animat hat ein Interesse an aktiver Exploration. Dazu sollte ein Erkenntnisgewinn durch Exploration Rewardqualitäten aufweisen, also direkt meßbaren Gewinn erzeugen. Damit muß das Animat den Konflikt zwischen Exploration und Exploitation, der Ausnutzung des bisher gelernten, selber lösen. Bisher geschieht dies durch Festlegung des Designers bezüglich der Exploration der Agenten und Koordination. Ansätze hierzu sind z.B. in Abschnitt 4.5 zur Exploration beschrieben.

- **Planung:** Durch vorausschauende Testung verschiedener Handlungsketten anhand eines Modells kann das Animat die Konsequenzen seiner Aktionen erfolgreicher vorhersehen. Das Modell kann z.B. aus erlernten Zustandsübergangswahrscheinlichkeiten bestehen oder einfach aus einem Gedächtnis über erlebte Zustandsübergänge.

- **Kompetenz:** Die Planung, Verhaltensgenerierung und Koordination profitieren vom Wissen darüber, wie gut das Animat seine Umwelt gezielt aktiv beeinflussen kann. Daraus entsteht dieses Bedürfnis. Es kann z.B als mitlaufender Mittelwert über den Erfolg (Differenz zwischen erwartetem und eingetretenem Reward) der Aktionen implementiert werden. Bei mangelnder Kompetenz könnte sich die Explorations- und Risikofreudigkeit des Animats mindern.

- **Werkzeuggebrauch:** Der Gebrauch von Werkzeugen erleichtert die Erfüllung bestimmter Aufgaben oder macht diese erst möglich. Entsprechend demonstriert die Fähigkeit, Werkzeuggebrauch zu erlernen, die Eigenschaft des Algorithmus, Hierarchien bilden zu können [Spier and McFarland, 1998].

- **Kosten-Reward-Balance:** Jeder Agent und jede Aktion verursacht Kosten: sowohl dem Gesamtsystem, als auch Gewinnausfälle bei anderen Agenten. Neben den erzielten Rewards sollte jeder Agent auch die von ihm verursachten Kosten lernen, um sie für unterschiedliche Verhalten, die das gleiche Ziel oder Bedürfnis verfolgen, gegeneinander abwägen zu können.

- **Rewardsubstitution:** Bedürfnisse können aus verschiedenen Quellen unterschiedlich gut bzw. effektiv befriedigt werden. Das Animat muß nach erhaltenem Reward, Dringlichkeit, Erfolgsaussicht und Kosten eine Quelle auswählen. Hier besteht die Möglichkeit der Validierung des Ansatzes anhand der Ergebnisse vorhandener Tierexperimente.

Diese existentiellen und kognitiven Bedürfnisse sollten sich durch Erweiterungen des in Kapitel 7 definierten Experimentalaufbaus umsetzen lassen, um in weiterführenden Arbeiten realistische, interessante Szenarien und Verhalten zu erzeugen.

9.1.2 Koordination mit dynamischen Bedürfnissen

Mit dynamischen Bedürfnissen rückt das Animat in die Nähe biologisch motivierter Modellierung, und für seine Koordination sollten die wichtigen ethologischen Funktionen aus Abschnitt 2.4 als Leistungsmaßstab herangezogen werden. Die folgende Liste enthält die relevanten Funktionen und ihre Erfüllung durch RL-Agenten in Szenarien mit den bisherigen, zeitlich unveränderlichen 'Bedürfnissen'.

- **Priorisierung:** Eine Handlungsauswahl unter vorrangiger Behandlung eines aktuell aktiven Handlungsantriebes ist nur sinnvoll mit Bedürfnissen, die aktiviert werden können. Sofern man die Zielerfüllung beim normalen Reinforcement Lernen als Priorisierung ansehen möchte, sind diese Bedürfnisse immer aktiviert und es werden immer Handlungen zu ihrer Erfüllung ausgeführt. Die Agenten haben nicht die Freiheit, über diese Erfüllung zu entscheiden und daher kann diese Funktion erst mit dynamischen Bedürfnissen untersucht werden.

- **Persistenz:** Bezüglich statischer Objekte weisen alle vom RL gelernten Strategien mit Traces ($\gamma > 0$) eine Persistenz auf. Auch wenn die Objekte nicht unmittelbar wahrgenommen werden können, werden aufgrund der impliziten Planung des RL Aktionen ausgeführt, die zu ihrer Erlangung führen. Da sie jedoch nicht über ein Gedächtnis verfügen, können bei dynamischen Objekten (siehe Abschnitt 6.1.1) einmal unterbrochene Strategien auch nicht gezielt wieder aufgenommen werden.

- **Opportunismus:** Mit statischen Objekten wird immer ein Opportunismus realisiert, indem ein Pfad gewählt wird, der maximalen Gesamtreward erzeugt. Mit dynamischen Objekten würde wohl eine sich bietende Gelegenheit genutzt oder einer auftauchenden Gefahr ausgewichen werden, wegen fehlender Persistenz kann der ursprüngliche Plan jedoch nicht wieder aufgenommen werden.

- **Sequenz:** Notwendige Voraussetzung zur Erfüllung dieser Funktion ist ein Gedächtnis darüber, ob die Umwelt durch die auslösende Handlung in jenen Zustand versetzt wurde, die eine Erfüllung des eigentlichen Zieles möglich macht. Fehlt ein solches Gedächtnis, kann auch die für RL notwendige Austestung aller möglichen Aktionssequenzen nicht zur Etablierung einer Sequenz führen, da es sich ohne Gedächtnis für den Agenten um einen POMDP handelt. Für einen Teil der Zustände muß gelten, daß nach Ausführung der Triggerhandlung andere Aktionen ausgeführt werden sollten als vorher. Diese Funktion kann durch Algorithmen des RL erfüllt werden, die Hierarchien von Makroaktionen bilden ([Parr, 1998, Precup et al., 2000, McGovern, 2002]) und in Abschnitt 4.8 eingeführt wurden.

Mit der Einführung dynamischer, z.B. homöostatischer Bedürfnisse und damit interner Zustände in das Animat, muß sich die Koordination der verschiedenen Expertenagenten nach den Ständen der verschiedenen Antriebe richten. Ein Lernen der Rangfolge der Experten für einen gegebenen Zustand mit einer festen Wichtung der Experten im gesamten Zustandsraum oder auch einer zustandsabhängigen Wichtung kann den Anforderungen dynamischer Bedürfnisse nicht mehr entsprechen. Derselbe, allein aus den sensorischen Merkmalen abgeleitete Zustand kann mit unterschiedlichsten Bedürfniskonstellationen eingenommen werden, die dann verschiedene Aktionen erfordern.

Da der Stand der Antriebe von allen bisher erlebten Zustands-Aktionspaaren abhängig ist, erfüllt ein Zustand, der nur aus den sensorischen Merkmalen ermittelt wird, direkt die Definition für einen POMDP. Ein möglicher Ansatz ist die oben erwähnte Erweiterung des Zustandsraumes um den Stand der Antriebe. Dies löst das POMDP-Problem für den Experten, ermöglicht gleichzeitig aber auch wieder die Verwendung bekannter Koordinationsansätze wie W-Lernen, da der Zustand der Experten auch den jeweiligen Bedürfnisstand und damit die Konstellation der verschiedenen Bedürfnisse kodiert. Eine Koordination sollte den Stand aller Antriebe, bzw. die Überlagerung aller Abweichungen vom jeweiligen Optimum, als zu minimierende Größe verwenden. Im Sinne einer systematischen Fortführung der Untersuchungen zum W-Lernen könnte also ein W-Lernagent ohne größeren Aufwand mit dynamischen Bedürfnissen ausgestattet und untersucht werden.

Humphrys argumentiert in [Humphrys, 1997], Abschnitt 15.1, daß W-Lernen auch Priorisierung und Persistenz aufweist, ohne auf zeitbasierte Koordinierungsmechanismen zurückgreifen zu müssen. Dies gilt jedoch nur unter der Voraussetzung, daß die entsprechenden Antriebe Teil des Zustandsraumes werden und die Agenten entsprechend unterschiedliche Q-Werte für den gleichen externen Zustand ausbilden, je nach Höhe des Bedürfnisses. Dann würde der Agent priorisiert, dessen Bedürfnis hoch ist und der im gegenwärtigen externen Zustand gute Aussichten auf Befriedigung dieses Bedürfnisses hat. Da er zur Ausführung kommt und so seinem Ziel einen Schritt näher, sollte er im folgenden Zustand noch größere W-Werte erzeugen und so sein Ziel erreichen können, ohne unterbrochen zu werden.

So realisiert WL Priorisierung und Persistenz, auf deren Grundlage auch Opportunismus entstehen kann. Erscheint ein unerwarteter Feind oder bietet sich die Gelegenheit, ein anderes Bedürfnis zu befriedigen, so entscheidet wieder das Zusammenspiel aus Größe des Bedürfnisses und Aussicht auf dessen Befriedigung über den gewählten Agenten. Da das ursprüngliche Bedürfnis noch nicht befriedigt wurde, hängt es vom Umfang der Abweichung vom ursprünglichen Plan ab, ob dieser wieder aufgenommen wird. Sind die Aussichten auf Erfüllung des Bedürfnisses nicht mehr gegeben, wird der ursprüngliche Agent auch nicht mehr zum Zuge kommen, anderenfalls wird ihm nachgegangen. Entsprechend ist zu vermuten und in einem geeigneten Szenario mit dynamischen Bedürfnissen zu untersuchen, daß W-Lernen die genannten ethologischen Funktionen erfüllen kann. Dies würde es in die Lage versetzen, auch über technische Anwendungen hinaus ein Modell zur Koordination verschiedener Bedürfnisse und Experten zu bilden.

Für ein Animat mit dynamischen Bedürfnissen sollten über die in Abschnitt 6.3.1 aufgeführten Experimente hinaus folgende Verhalten gezeigt werden:

Exp. 1: **Persistenz:** Anfahrt der Futterstelle mit Passieren der Tränke, ohne zu trinken wegen fehlendem Durst bzw. Anfahrt der zeitweilig verdeckten Futterstelle.

Exp. 2: **Fluchtpersistenz:** Flucht vor dem Jäger wird fortgesetzt, obgleich der Jäger nicht mehr wahrnehmbar ist.

Exp. 3: **Opportunismus:** Anfahrt der Futterstelle mit Passieren der Tränke, mit Trinken wegen Durst.

Exp. 4: **Fluchtopportunismus:** Anfahrt der Futterstelle wird wegen Erscheinen des Jägers abgebrochen.

Exp. 5: **Kosten-Nutzen-Abwägung:** Anfahrt einer weit entfernten Futterstelle wird wegen Erscheinen des Wildes abgebrochen, Wild wird gefangen.

Exp. 6: **Sequenz:** Eine Teilaufgabe setzt zwingend die Ausführung einer anderen Teilaufgabe voraus, z.B. muß eine kollisionsträchtige Sackgasse befahren werden, bevor eine Futterstelle erscheint.

9.1.3 Biologisch motivierte Agenten

Ein Animat mit dynamischen Bedürfnissen und entsprechender Koordination kann dann auch im selben Szenario mit Ψ [Dörner, 1999] verglichen werden und im Anschluß seine Fähigkeiten zur Modellierung psychologischer Phänomene unter Beweis stellen.

Der Ausbau des Animats hin zu eher biologisch relevanten Algorithmen kann dann mit Ansätzen weitergeführt werden, die an die Ursprünge der RL-Algorithmen zurückgehen und Reinforcement Lernen auf operande bzw. instrumentelle Konditionierung [Klein, 1991] zurückführen. Diese Lernphänomene werden mit biologisch motivierten Modellen der Interaktion beteiligter Neuronengruppen implementiert und schließen zum Beispiel *READ* [Grossberg and Schmajuk, 1987], oder Modelle der Basalganglien ein [Brown et al., 1999], [Prescott et al., 1999] bzw. [Balkenius and Morén, 1998]. Ihre Leistungsfähigkeit bei der Steuerung realer Roboter wurde bisher wenig untersucht und sie wurden bisher auch nicht den gängigen RL-Verfahren gegenübergestellt, obwohl auch sie auf den schon erwähnten Konferenzen 'From animals to animats' [Pfeifer et al., 1998, Meyer et al., 2000] vertreten sind. Eine explizite, biologisch relevante Koordination ist allerdings nicht bekannt, sollte aber vermutlich ähnlich den in [Dörner, 1999] und [Blumberg et al., 1996] realisierten Koordinationen auf dem Produkt aus Bedürfnis und Erfolgschance auf dessen Erfüllung basieren und so integraler Bestandteil der Modelle sein.

Aus der größeren Nähe zum biologischen Vorbild ergeben sich mit diesen Modellen Möglichkeiten des direkten Vergleichs der Verhalten und resultierenden Leistungen mit psychophysiologischen Daten, z.B. aus Experimenten zur operanden oder instrumentalen Konditionierung. Die Untersuchungen müssen auch versuchen, Antwort auf die Frage zu finden, welche Verhalten biologisch motivierte Ansätze besser erzeugen oder gar erst ermöglichen, die Tiere zu ihren komplexen Verhaltensleistungen in der realen Umwelt befähigt.

9.2 Optimierung der Agenten

Hinsichtlich einer Anwendung von RL-Agenten und deren Koordination zur Lösung definierter Aufgaben in einer realen Umwelt müssen die Agenten und ihre Bestandteile optimiert werden.

9.2.1 Clusterer

Da in der realen Umwelt Erfahrungen teuer bzw. selten sind, sollten Agenten in der Lage sein, wichtige Situationen bereits beim einmaligen Auftreten zu erkennen und zu lernen. Dazu sind u.a. andere Clusterer für die Ermittlung des Zustandes notwendig.

Diese dürfen andererseits statistisch häufige Fälle nicht mißrepräsentieren, z.B. nicht als eine Generalisierung der Ausnahmen. Entsprechend müssen sie nicht nur ermitteln, welcher ihrer Repräsentanten einem gegebenen Eingangsmuster am besten entspricht, sondern sie müssen auch prüfen, ob es ihm auch gut genug entspricht.

Dies kann realisiert werden durch einen zweiten Vergleich des Eingangsmusters, das der Gewinnerknoten erwartet, mit dem tatsächlich anliegenden. Für diesen Vergleich sollte eine andere Metrik eingesetzt werden als für die Ermittlung des am besten entsprechenden Knoten. Der Vergleich der beiden Metriken (z.B. ihr Quotient) bildet die Grundlage der Entscheidung über die Relevanz des Knoten.

Ein solcher Vergleich zwischen Top-Down-Erwartung und Bottom-Up-Input wird in Systemen der Adaptive Resonance Theory (ART) [Carpenter and Grossberg, 1991] umgesetzt. Da solche Netze auch eine weitere Anforderung an Anwendungen in der realen Umwelt erfüllen, nämlich lebenslang lernfähig zu sein, sollten weiterführende Untersuchungen zu den Agenten solche Systeme einbeziehen.

9.2.2 RL-Algorithmus

Weitere Möglichkeiten der Optimierung der Expertenagenten bestehen auch in der zweiten Komponente des hier verwendeten Agentenaufbaus, dem RL-Algorithmus. Für reale Anwendungen tritt das Problem der POMDP sehr in den Vordergrund. Zu seiner Lösung sind mehrere konkrete Ansätze möglich, wie sie auch in Kapitel 4 vorgestellt wurden. Sie schließen die Einbeziehung relevanter zeitabhängiger Informationen in die Zustandsbeschreibung in verschiedenen Formen (Traces, vergangene Aktionen oder sensorische Merkmale), die Zerlegung in Markow'sche Teilprobleme mit fester zeitlicher Abfolge, Makroaktionen oder den Einsatz gradientenbasierter RL-Verfahren ein.

Im gegebenen Szenario sollte untersucht werden, welchen Effekt eine Hinzunahme des letzten Fahrkommandos oder eines Differenzbildes der letzten Fahrbewegungen auf die Leistung der Agenten hat. In diesen Erweiterungen steckt die Zeitinformation, um ansonsten gleiche Eingangsmuster voneinander zu unterscheiden. Andererseits erhöhen diese Lösungen die Komplexität des Eingangsraumes und benötigen daher längere Lernzeiten.

Algorithmen des RL, die sich auf das Erlernen des Rewardgradienten konzentrieren (siehe Abschnitt 4.4) wurden unter anderem mit der Zielrichtung entwickelt, POMDP zu lösen und zeigten in ersten Veröffentlichungen (z.B. [Baxter et al., 2001]) vielversprechende Ergebnisse. Zu ihrer Integration in eine Koordination muß allerdings die Eignung von Wahrscheinlichkeiten für die Ausführung von Aktionen zur Verwendung in Koordinationsansätzen wie WL untersucht werden. Entsprechend sind grundsätzliche Betrachtungen zur Entwicklung einer Koordination auf der Grundlage dieser Algorithmen notwendig.

Verschiedene Untersuchungen setzen Netze, die zum überwachten Training von Zeitreihen verwendet werden, auch für RL-Probleme [Yamauchi and Beer, 1994] ein. Mit inne-

ren Zuständen und Zeitinformation aus Rückkopplungen können POMDPs durch diese Präsenz der Sequenz bisheriger Zustandsaktionsfolgen aufgelöst werden.

Ein grundsätzlich anderer Ansatz zur Detektion und Behandlung von POMDP besteht in der Verwendung statistischer Tests wie des Kolmogorow-Smirnow-Tests über Baumstrukturen aus den bisherigen Zustands-Aktionspaaren und resultierenden Rewards [McCallum, 1997]. Die entsprechenden Ansätzen basieren jedoch stark auf Methoden der symbolischen KI, deren neuroinformatische Realisierung aber nicht unmöglich sein sollte.

9.3 Weiterentwicklung der Koordination

Die Leistungsfähigkeit einer Koordination sollte durch eine Ausnutzung synergistischer Effekte zwischen den Agenten erhöht werden können. Das heißt, die Koordination soll nicht nur darum bemüht sein, die Leistungsfähigkeit eines Monolithen zu erreichen, sondern darüber hinaus die speziellen Fähigkeiten der Agenten zu deren gegenseitiger Unterstützung einsetzen. Solche synergistischen Effekte können sich jedoch nur ergeben, wenn die Aktionen der verschiedenen Agenten nicht völlig unabhängig voneinander sind.

Ein prinzipiell erfolgreicher Ansatz für die Koordination ist die Maximierung des Produktes aus (Bedürfnis * Erfolgsaussicht * Dringlichkeit) [Spier and McFarland, 1998, Dörner, 1999], gekoppelt mit einer Exploration des gegenseitigen Einflusses der Agenten und einer Lernfähigkeit, um unvorhersehbare, synergistische Effekte zwischen den Agenten zu lernen.

Vorstellbar ist die Koordinierung als ein rekurrenter Prozeß ähnlich einem Blackboard, in dem die einzelnen Agenten ihre Aktionen nicht nur als einen dynamisch gewichteten Wert vorschlagen, sondern als Gaussfunktion, deren Position im Aktionsraum die vorgeschlagene Aktion verkörpert, deren Höhe dem Wert (Erfolgsaussicht) entspricht und deren Breite eine Umkehrfunktion der Konfidenz für diesen Vorschlag ausdrückt (je breiter, um so unsicherer). Darüber hinaus können sie auch solche Funktionen negativer Art propagieren, als Ausdruck für Aktionen, die sie nicht zulassen können. Der Prozeß soll in Abbildung 9.1 schematisch dargestellt werden.

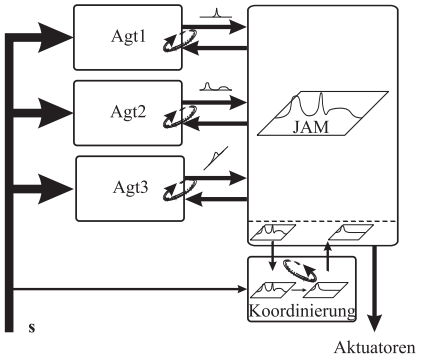

Abbildung 9.1: ***Rekurrenter Einigungsprozeß der Expertenagenten auf eine Aktion.*** *Agenten unterbreiten Aktionsvorschläge, die überlagert die 'Joint Action Map' (JAM) ergeben, und verarbeiten die Vorschläge anderer Agenten in mehreren Iterationen, bis ein Koordinierungsprozeß die auszuführende Aktion, abhängig vom Gesamtzustand 's' festlegt und allen mitteilt.*

Ihr Vorschlag erhält mit wachsender Dringlichkeit mehr Gewicht, womit auch eine zeitliche Komponente für die Aktionsplanung zu berücksichtigen ist.

Nachdem alle Agenten solche Vorschläge unterbreitet haben, können sie auch die resultierende Aktionshypothese wieder lesen und darauf reagieren (ähnlich Blackboardsystemen, siehe Abschnitt 2.4.3), woraus der genannte rekurrente Prozeß entsteht. Er wird durch zeitliche Randbedingungen oder eine Einigung der Agenten abgebrochen. Die auszuführende Aktion ergibt sich aus einem zustandsabhängigen Koordinierungsprozeß, der die aktuellen Bedürfnisse und Dringlichkeiten berücksichtigt, und wird den Agenten wieder mitgeteilt.

9.4 Emergente Modularisierung

Schließlich sollte das in Kapitel 1 eingeführte sensomotorische Paradigma hinsichtlich der Autonomie und eines minimalen Ressourceneinsatzes des Animats konsequent umgesetzt werden. Die Untersuchungen in dieser Arbeit und die Ergebnisse der vorgeschlagenen weiterführenden Untersuchungen sollten schrittweise zusammengeführt und ausgebaut werden. Dazu sollen im folgenden Anregungen gegeben werden, die sehr viel konzeptionellerer Natur als die bezüglich der dynamischen Bedürfnisse sind. Anstelle implementationsreifer Gleichungen umfassen sie eher besonders wichtig erscheinende Forschungsfelder.

Nach einer Optimierung der eingesetzten Agenten hinsichtlich schnellen, lebenslangen Lernens wichtiger Ausnahmen, aber auch statistisch häufiger, wichtiger Fälle sollte die resultierende optimale Policy mit einer minimalen Anzahl von Zuständen repräsentiert werden. Solche Zustände können mit inkrementellen Netzen erzeugt werden, die nach Bedarf wachsen oder schrumpfen. Für die Verteilung dieser Zustände im Zustandsraum sollte dies auch bedeuten, daß benachbarte Zustände nicht dieselbe Aktion veranlassen.

Für die Zustände sollte jeder Agent die für ihn besonders relevanten Merkmale der Umwelt (z.B. [Finton, 2002]) herausfinden können, nach denen bei einer Aktivierung des entsprechenden Bedürfnisses in der Umwelt mit einer selektiven Aufmerksamkeit auch gesucht werden kann.

Die Anzahl und Aufgabengebiete der Agenten sollten vom Algorithmus selber festgelegt werden (z.B. [Baum, 1998]), nach Optimalitätskriterien hinsichtlich optimaler Redundanz und gegenseitiger Unterstützung mit möglichst geringer Interaktion vom Designer. Unter dieser Voraussetzung sollte eine Superadditivität durch Ausnutzung synergistischer Effekte möglich sein. Damit sollte das System die Vorzüge eine Multiagentensystems ausnutzen, sich aber auch mit einem Monolithen messen und ihn übertreffen können. Ein resultierendes System würde ein Prinzip umsetzen, das 'emergente Modularisierung' genannt werden könnte.

Eine erfolgreiche Lösung dieser sicherlich umfangreichen Aufgaben bringt das System einer Umsetzung des in Kapitel 1 vorgestellten Gesamtkatalogs von Anforderungen an ein intelligentes autonomes System näher. Vermutlich schon vor der vollständigen Erfüllung des Katalogs könnten auch intelligente Haushaltsroboter realisiert werden.

Kapitel 10

Zusammenfassung

Dieses Kapitel faßt die Untersuchungen und Ergebnisse dieser Arbeit im Überblick zusammen, Details sollten den Zusammenfassungen der einzelnen kapitel entnommen werden.

Diese Arbeit hatte zum Ziel, auf einem realen Roboter Verhalten zu erzeugen und zu koordinieren, indem die Aktionsvorschläge mehrerer Agenten koordiniert werden. Dabei sollten die Verhalten durch Lernen in einer unbekannten Umwelt erworben werden. Der Schwerpunkt lag auf der adaptiven Koordination der Verhalten mit W-Lernen.

Der Grundansatz für die Verhalten lag in Brooks' verhaltensbasierten Systemen, die zunächst von alternativen Paradigmen zur Steuerung von Robotern, der symbolischen KI und der probabalistischen Robotik abgegrenzt wurden. Wesentliche Eigenschaften und Anforderungen an Verhalten und Agenten wurden herausgestellt und zu ihrem lernenden Erwerb der Einsatz von Reinforcement Lernen motiviert.

Zum Reinforcement Lernen wurde eine detaillierte Einführung und Übersicht über aktuelle Algorithmen vorgestellt. Die Probleme des Reinforcement Lernens wurden diskutiert und die zu untersuchenden Algorithmen *Q-Learning* und *Sarsa* festgelegt.

Das Problem der Koordinierung mehrerer Expertenagenten beim Zugriff auf die limitierte Ressource der Aktuatoren des Roboters, bei gleichzeitiger Maximierung des Gesamtrewards für das Animat, wurde als ein ethologisches Problem der Aktions- bzw. Verhaltensauswahl charakterisiert. Die Verhaltenskoordination wurde in den größeren Rahmen der Multiagentensysteme eingeordnet und Eigenschaften und Anforderungen an ein Multiagentensystem herausgestellt. Die formellen Grundlagen für Multiagentensysteme im Reinforcement Lernen wurden mit stochastischen Spielen bzw. Multiagent-MDP (MMDP) eingeführt und einige relevante Algorithmen vorgestellt. Die Lösung einer Aufgabe durch ein Multiagentensystem wurde der Lösung derselben Aufgabe durch einen monolithischen Agenten gegenübergestellt. Das oben genannte Koordinierungsproblem wurde als Mischform aus nichtkooperativem und vollständig kooperativem MMDP klassifiziert, dessen Lösung als stochastisches Spiel noch aussteht.

W-Lernen [Humphrys, 1997] behandelt genau dieses Koordinierungsproblem und stützt sich auf Agenten, die mit Reinforcement Lernen trainiert wurden. Unter den bekannten Algorithmen erscheint es als der flexibelste und leistungsfähigste. Es erlaubt die Herausbildung beliebiger Beziehungen zwischen den Experten einschließlich der Hierarchie und erfüllt dabei die überwiegende Mehrzahl der Anforderungen an ein Multiagentensystem.

Der Kern der Arbeit beschäftigte sich mit dem Entwurf und der Umsetzung eines Szenarios zur Überprüfung der Leistungsfähigkeit von W-Lernen sowohl in einem realen Labyrinth, als auch in einer simulierten Umgebung. Dabei wurden erstmals nahezu alle Varianten des W-Lernens getestet. Diese Arbeit stellt nach Wissen des Autors auch die erste Umsetzung des W-Lernen auf einem realen Roboter dar. Der zentrale Vergleich der Untersuchungen betraf den zwischen der Leistungsfähigkeit eines komplexen monolithischen Agenten, der alle Teilaufgaben geschlossen löst, und dem W-Lernen. Er bezog sich auf die Optimalität des Verhaltens des Monolithen als RL-Algorithmus in einem MDP, und der emergenten Herausbildung von Fähigkeiten für W-Lernen aus einer geeigneten Koordination. Diese sollte dazu Synergien zwischen den spezialisierten Verhalten der Experten ausnutzen.

Die Ergebnisse bestätigten die Überlegenheit eines mit Algorithmen des RL trainierten Monolithen gegenüber den verschiedenen Varianten der Koordinierung von Expertenagenten durch W-Lernen. Sie bestätigten auch verschiedene Vermutungen Humphrys' hinsichtlich der Leistungsfähigkeit des puren W-Lernen und der kollektiven Varianten. Die Gründe der Unterlegenheit des W-Lernens sind vor allem in einer inflexiblen Wichtung der Agenten und einer fehlenden impliziten Planung bezüglich des Gesamtreturns durch das W-Lernen zu suchen. Es kann jedoch festgestellt werden, daß Reinforcement Lernen und W-Lernen auf realen Robotern eingesetzt werden können.

Hingegen mußte festgestellt werden, daß die in Abschnitt 6.3 vorgeschlagenen Erweiterungen zum W-Lernen hinsichtlich dieser Nachteile nicht zum Erfolg führten. Neben detaillierten Untersuchungen in einem anderen Szenario sollten vor allem die Experten verbessert werden, wozu in Abschnitt 9.2 konkrete Vorschläge unterbreitet werden. Untersuchungen in einem weiteren Szenario sollten szenariospezifische Probleme herausstellen und der Vermutung nachgehen, daß sich eine Superadditivität der Fähigkeiten der Experten nur bei Teilaufgaben einstellen kann, die stark voneinander abhängig sind, also zum Beispiel in einer Sequenz ausgeführt werden müssen.

Weitere Vorschläge zur Erweiterung des Szenarios, um Methoden der Koordination von Expertenagenten weiterentwickeln zu können, wurden in Kapitel 9 vorgestellt. Sie betreffen vor allem die Erweiterung des Szenarios für eine weitere Annäherung an ein Tier, das in seiner Umwelt überleben muß. Der Schlüssel sind dynamische Bedürfnisse, die sich mit den bisher eingesetzten Mitteln des Reinforcement Lernens und des W-Lernens implementieren lassen sollten. Im Ergebnis der vorgeschlagenen Untersuchungen könnte ein Animat entwickelt werden, das die notwendigen Verhalten für die ihm gegebenen Ziele und sein Überleben selbst definiert, lernt und koordiniert. Die Ergebnisse und aufgetretenen Probleme jedoch bestätigten das Vorgehen, zunächst das W-Lernen mit statischen Bedürfnissen zu untersuchen, um die Grundlagen für eine Umsetzung einer Koordination in realen Umwelten zu schaffen.

Lokalisation des Roboters

Zur Steuerung der dynamischen Objekte, die vom Beamer projiziert wurden, aber auch zur Aufzeichnung der Fahrspuren des Roboters und zur Ermittlung der sensorischen Informationen für eine Darstellung des Szenarios als MDP mußte der reale Roboter in seinem Labyrinth lokalisiert werden. In diesem Anhang wird die dazu eingesetzte, recht einfache Methode erläutert.

A.1 Grundsätzlicher Ablauf

Nach jedem Schritt des Roboters mußte dessen Position und Orientierung ermittelt und den Beamerobjekten mitgeteilt werden. Um zu verhindern, daß diese bei ihrer Bewegung auf den Roboter zu oder von ihm weg die im Labyrinth aufgebauten Hindernisse ignorierten, mußten auch die Hindernisse gefunden und die Beamerobjekte entsprechend gesteuert werden.

Zur Ermittlung der Position und Orientierung des Roboters wurde eine weitere Farbkamera benutzt, von der nach jedem Schritt des Roboters ein Bild von oben auf das Labyrinth mit dem Roboter gewonnen wurde. Die Ecken des Labyrinths und der Roboter waren für die Positionsbestimmung mit einem einfachen schwarzen Muster markiert, nach denen im Bild gesucht wurde. An der Position des Roboters wurde dann unter Ausnutzung der Kenntnis fester Abbildungsverhältnisse die Orientierung des Roboters mit einer vorgegebenen Winkelauflösung ermittelt.

Die Hindernisse wurden anhand desselben Bildes unter Verwendung der vorher bestimmten Farbe des Holzes ebenfalls zu jedem Zeitschritt gesucht. Dies ermöglicht prinzipiell auch ein Verschieben der Wände, wobei auch die verschobenen Wände bei der Bewegung der dynamischen Beamerobjekte beachtet werden.

Um diese Bestimmungen unabhängig von der konkret eingesetzten Kamera und dem konkret verwendeten Beamer zu gestalten, wurden die Positionen des Roboters, der Beamerobjekte und der Wände in ein einheitliches, gemeinsames Koordinatensystem umgerechnet. Dieses Koordinatensystem kann zur Berechnung der Bewegung der Beamerobjekte eingesetzt werden. Die Steuerung der Objekte folgt dabei dem Vorbild der Robotersteuerung, d.h. zur Steuerung der Beamerobjekte wurden dieselben programmtechnischen Schnittstellen benutzt wie zur Steuerung des Roboters. Jeder Algorithmus, der den Roboter steuern kann, kann auch die Beamerobjekte steuern und umgekehrt.

A.2 Bestimmung von Position und Orientierung des Roboters

Die Ermittlung der Position des Roboters stützt sich auf ein einfaches, schwarzes, rotationssymmetrisches Muster, das sowohl bei Drehungen des Roboters als auch bei seinen Bewegungen unter wechselnden Licht- und Farbverhältnisse robust auffindbar ist. Es wurde ein schwarzer Kreis vom Durchmesser des Kheperas verwendet. Dieser Durchmesser ergab sich als größte mögliche Fläche, die das Sichtfeld der Kamera des Khepera nicht beeinträchtigte. Eine möglichst große Fläche war notwendig, um Verwechslungen des Musters mit anderen schwarzen Bildinhalten, wie Schatten der Wände, des Roboters selber oder der Steuerkabel zu vermeiden.

Zur Bestimmung der Orientierung des Roboters wurde das Muster um einen konzentrischen kleineren weißen Kreis, der die Erkennungssicherheit gegenüber schwarzen Schattenwürfen erhöht, und einen definierten Durchbruch erweitert (siehe Abb. A.1 rechts). Nach einigen Versuchen wurde der Öffnungswinkel des ausgeschnittenen Ringabschnitts auf 60° festgelegt.

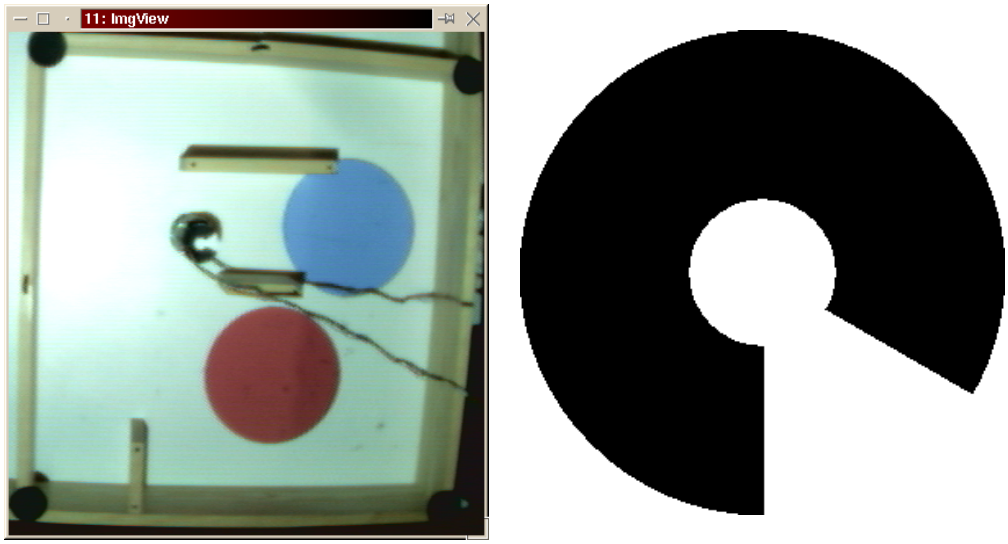

Abbildung A.1: *Erweiterter Experimentalaufbau mit markiertem Khepera und Ecken des Labyrinths und zur Orientierungsbestimmung eingesetztes Muster*

Zum Beginn einer Serie von Experimenten (Initialisierung des Systems) werden die Positionen der Ecken des Labyrinths bestimmt. Bis zum Neustart des Programmmoduls wird von einer festen Position der Ecken ausgegangen und diese nicht wieder bestimmt. Bei der Bestimmung der Position der Ecken wird weiterhin davon ausgegangen, daß die Bildgröße ungefähr mit dem zu durchsuchenden Bereich übereinstimmt, denn der Suchbereich für die Ecken umfaßt zur Beschleunigung des Verfahrens nur das betreffende Viertel des Bildes (das linke obere Viertel wird nach der linken oberen Ecke durchsucht usw.). Nachdem die Ecken gefunden sind, wird ein Suchbereich für den Roboter festgelegt, der als parallel zu den Achsen des Bildes angenommen wird. Um Fehler aus Verzerrungen zu vermeiden, wird der Suchbereich über den notwendigen hinaus erweitert. Abb. A.2 links zeigt eine Darstellung der Szene aus Abb. A.1 links mit eingezeichnetem Suchbereich,

wie sie auch zur visuellen Kontrolle der Lokalisation während der Experimente verwendet wurde. In dem gefundenen Bereich wird nach dem Muster für den Khepera gesucht.

Abbildung A.2: ***Lokalisation des Roboters.*** *Links ist eine Draufsicht auf die Szene mit Labyrinth und Roboter mit eingezeichnetem Suchbereich und gefundener Roboterposition dargestellt. Die Darstellung in der Mitte zeigt den originalen Bildausschnitt, in dem der Roboter gesucht wurde. Er kann maximal ungefähr so weit fahren, wie sein Durchmesser beträgt (5cm). Das Suchfenster ist nach allen Seiten etwas größer. Die Darstellung rechts enthält entsprechenden Bildausschnitt nach der Segmentierung.*

In jedem weiteren Schritt wird der Roboter nur noch in einem kleineren Suchbereich um seine Position aus dem vorigen Schritt herum gesucht. Der Suchbereich (Abb. A.2 Mitte) wurde etwas größer als die maximale Fahrstrecke des Roboters gewählt. Die zur Positionssuche verwendete Maske ist diejenige, die der im letzten Schritt ermittelten Orientierung entspricht (siehe Abschnitt A.2.2). An der gefundenen Position (siehe Abschnitt A.2.1) wird danach die neue Orientierung des Roboters ermittelt. Beide Teilschritte werden im folgenden detailliert erläutert. Bei jedem Beginn eines neuen Trials, also bei jedem Einsetzen des Roboters, wurde jedoch wieder der gesamte Suchbereich durchsucht.

A.2.1 Positionsbestimmung

Die Position der Ecken- und Robotermarkierung im Bild wurde in einem zweistufigen Prozeß ermittelt. Zunächst wurde der angegebene Bildausschnitt, in dem die Position gefunden werden sollte, hinsichtlich schwarzer Bildinhalte segmentiert. Das heißt, alle schwarzen Bildinhalte blieben weiterhin schwarz, alle anderen wurden weiß gefärbt. Dazu wurden für jeden Pixel des Bildausschnittes die Werte des Rot-, Grün- und Blaukanals mit einer Schwelle verglichen. Nach mehreren Versuchen wurde diese Schwelle auf einen Wert von 90 festgelegt. Jeder der Kanäle hat einen Wertebereich von $[0 \ldots 255]$. Zu einer zusätzlichen Erhöhung der Erkennungssicherheit wurde der Grauwert aus den drei Kanälen ermittelt und mit derselben Schwelle verglichen. Der Grauwert ergab sich aus $Grau = (16 \cdot Rot + 11 \cdot Gruen + 5 \cdot Blau)/32$. Erst wenn die logische UND-Kombination aller vier Vergleiche erfüllt war, wurde der Pixel als schwarz eingezeichnet. Der so segmentierte Bildausschnitt (siehe Abb. A.2 rechts) wurde in eine gleichgroße Matrix aus Boole'schen Werten übertragen, um einen schnellen Vergleich mit einer Maske zu ermöglichen, die eine Abbildung des in Abb. A.1 rechts dargestellten Musters enthält (siehe Abb. A.3 links).

Die zweite Stufe des Prozesses war die Bestimmung der Position der besten Übereinstimmung der Maske im segmentierten Bild. Dazu wurde per Hand eine Maske in der Größe der Abbildung der Markierung im Bild angelegt. Diese Größe ist von den Abbildungsverhältnissen, also der verwendeten Kamera mit den Abbildungseigenschaften des Objektivs und deren Abstand zur Markierung abhängig. Unter der Annahme feststehender Abbildungsverhältnisse wurde die Maskengröße einmal gemessen und fest eingestellt. Die Koordinaten der besten Übereinstimmung zwischen Bildausschnitt und Maske wurden durch Summierung der Übereinstimmungen (Hamming-Distanz) über alle Pixel der Maske (Äquivalenz zwischen Maske und Matrix aus dem segmentierten Bildausschnitt) ermittelt. Die Maske wurde hierfür pixelweise über die Matrix bewegt, sodaß zuerst die linken oberen Ecken von Maske und Matrix übereinstimmen und zuletzt die rechten unteren Ecken. Die Koordinate des Pixels mit der größten Summe entspricht der Position der Roboter- bzw. der Eckenmarkierungen. Diese wurde in das gemeinsame Koordinatensystem umgerechnet und eingetragen.

A.2.2 Orientierungsbestimmung

Nach der Bestimmung der Position des Roboters wurde an der gefundenen Position die Orientierung des Roboters ermittelt. Das dazu eingesetzte Verfahren ist dem zur Positionsbestimmung ähnlich und auch zweistufig.

In einem ersten Schritt wurde wieder der zu durchsuchende Bildausschnitt nach schwarzen Bildinhalten segmentiert, jedoch unterschied sich die Segmentierung etwas. Da nun die Fehler eher durch den Roboter selbst verursacht wurden, genauer gesagt durch die Farbe seiner Platinen und entsprechende Schatten, wurde der Roboter hell ummantelt. Entsprechend konnte die Segmentierung vereinfacht, die Kriterien aber auch verschärft werden. Es wurden wieder Vergleiche der drei Farbkanäle mit einer Schwelle durchgeführt, die diesmal bei 100 lag, jedoch der zusätzliche Vergleich mit dem Grauwert konnte entfallen. Das beschleunigte die Segmentierung.

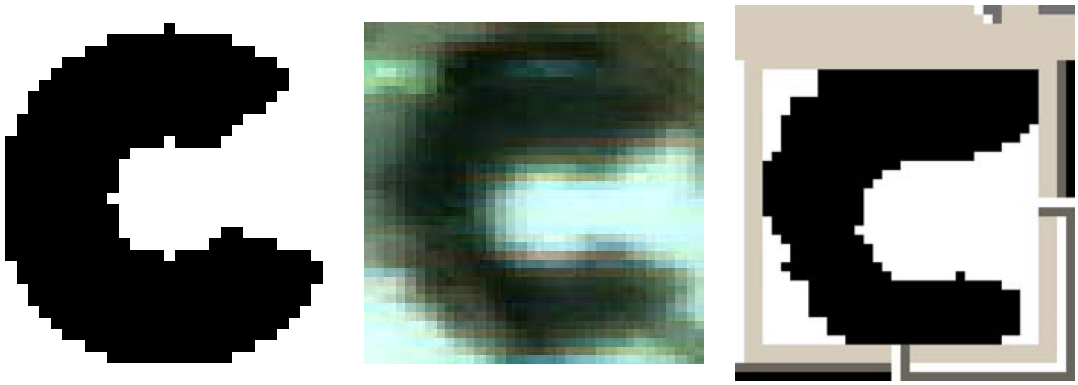

Abbildung A.3: ***Orientierung des Roboters.*** *Links ist eine Abbildung der Boole'schen Suchmaske dargestellt, in der Mitte der Bildausschnitt an der gefundenen Roboterposition, in dem die Orientierung gesucht wurde, rechts der segmentierte Bildausschnitt.*

Der Bildausschnitt, der nun ebenso groß ist wie die Maske (siehe Abb. A.3 links), wurde segmentiert und wieder in eine gleichgroße Boole'sche Matrix übertragen. Die Übereinstimmung der Matrix mit der Maske wurde wieder durch eine Hamming-Distanz

über die Äquivalenzen zwischen Matrix und Maske ermittelt. Anstelle einer pixelweisen Verschiebung wurde nun jedoch die Matrix mit einer Serie von Masken verglichen. Jede der Masken enthielt ein gedrehtes Abbild des originalen Musters. Es wurde eine Auflösung für die Orientierungsbestimmung festgelegt, die 10° betrug. Dies entspricht einem Drittel des Drehwinkels des Roboters und sollte ausreichend sein. Entsprechend entstanden 36 Masken, die jede ein um 10° gedrehtes Abbild des Musters enthalten. Die Maske mit der besten Übereinstimmung ergab die Orientierung des Roboters.

A.3 Bestimmung der Position von Hindernissen im Labyrinth

Zur Steuerung der dynamischen Objekte durch den Beamer sollte die Position der Hindernisse im Labyrinth ermittelt werden, damit die dynamischen Objekte sie beachten konnten. Diese Bestimmung wurde in jedem Schritt durchgeführt.

Hierzu wurde das gesamte Bild aus der Übersichtskamera (siehe Abb. A.1 links) nach holzfarbenen Bildinhalten durchsucht und diese in das gemeinsame Koordinatensystem für Hindernisse, Objekt- und Roboterposition eingetragen. Dazu wurde das Bild segmentiert, indem es zunächst einer HSI-Transformation [Gonzalez and Woods, 1993] unterzogen wurde. Danach wurden die Pixel, die eine Übereinstimmung mit dem in den Farbuntersuchungen (siehe Abschnitt 7.2.3.3) ermittelten Farbton (H-Wert) für das Holz aufwiesen, in das gemeinsame Koordinatensystem eingetragen. Die Übereinstimmung wurde durch die Aktivierung einer Gaussfunktion mit ihrer Mitte über dem Farbton und einer Standardabweichung $\sigma = 0.03$, ähnlich der für die Aktivierungen der rezeptiven Felder verwendeten (siehe Abschnitt 7.2.3.2), berechnet.

Einige Fehler beim Finden des Holzes machten jedoch eine Erweiterung zu dieser Bestimmung notwendig. Da solche Störungen durch den Roboter und seine Steuerkabel sowie die Objekte aus dem Beamer selbst verursacht wurden, die sich alle bewegen, konnten sie durch Mittelung über mehrere aufeinanderfolgende Bilder beseitigt werden. Die Positionen, an denen Holz gefunden worden war, wurden in eine Matrix mit der Größe des gemeinsam verwendeten Koordinatensystems eingetragen. Jedes der Elemente der Matrix enthielt eine laufende Summe der ermittelten Aktivierungen, aus denen nur die größten in das gemeinsame Koordinatensystem übertragen wurden. Durch die Bewegung der Fehlerquellen sinkt diese Summe für die entsprechenden Positionen schnell unter die Summe für die Positionen mit festen Holzobjekten. Die Positionen mit den größten Summen wurden durch Vergleich mit einer Schwelle ermittelt, die sich bei jedem Zeitschritt aus der Summe über die gesamte Matrix, insgesamt also aus einem laufenden Mittelwert, ergab.

Die Abbildungen A.4 zeigen eine Sequenz mit fahrendem Roboter, sich bewegendem *Jäger* und den ermittelten Positionen mit Holz. Links ist die Ausgangssituation dargestellt, in der das erkannte Holz als schwarze Punkte gekennzeichnet ist. In der Mitte die Situation beim zweiten Schritt des Roboters, der *Jäger* bewegt sich auf den Roboter zu. Die Störungen durch die Steuerkabel des Roboters und den *Jäger* sind zu erkennen. Nach 9 Schritten, bei der Kollision des Roboters mit der Wand, sind diese Störungen bereits zu großen Teilen beseitigt. Der *Jäger* wurde wegen der Kollision wieder auf seinen Ausgangspunkt zurückgesetzt, und der Roboter wich von der Wand zurück.

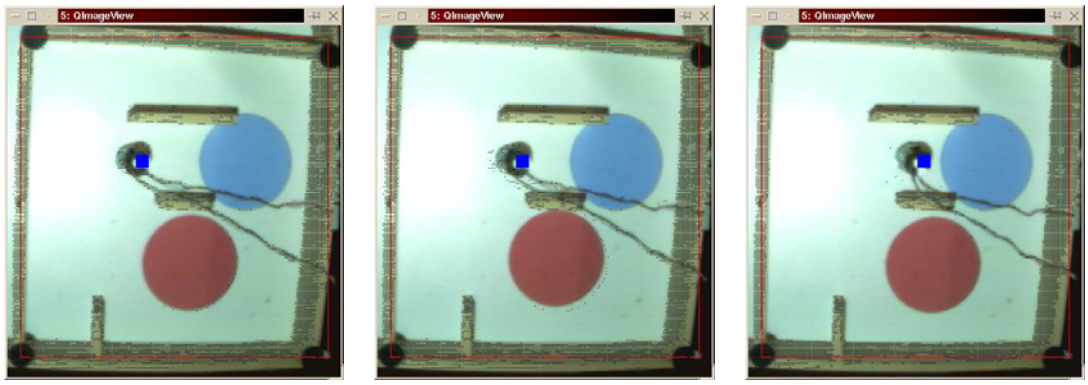

Abbildung A.4: ***Ermittlung der Position der Hindernisse.*** *Links ist die Ausgangssituation dargestellt, mit Suchbereich und Position des Khepera. Die Positionen, an denen Holz erkannt wurde, sind mit schwarzen Punkten markiert. In der Mitte ist die gleiche Aufnahme für den zweiten Schritt des Roboters gezeigt, die auch Störungen durch die Steuerkabel und den Jäger zeigt. Der Jäger bewegt sich auf den Roboter zu. Die Abbildung rechts zeigt die Szene nach 9 Schritten bei Kollision. Die in der Mitte sichtbaren Störungen wurden durch Mittelung beseitigt.*

Abkürzungen

AHC	Adaptive Heuristic Critic
ART	Adaptive Resonance Theory
ASE	Adaptive Search Element
BBS	behavior-based system
DAI	Distributed Artificial Intelligence
GNG	Growing Neural Gas
GPS	Global Positioning System
MAS	Multiagentensystem
MDP	Markov Decision Process
MLP	Multilayer Perzeptron
MMDP	Multiagent MDP
NG	Neural Gas
PG	Policygradient
POMDP	Partially Observable Markov Decision Process
PR	Probabilistische Robotik
RL	Reinforcement Lernen
SAP	State-Action-Pair, Zustands-Aktionspaar
sKI	symbolbasierte Künstliche Intelligenz
SMDP	Semi-MDP
TD	Temporal Difference
VKI	Verteilte Künstliche Intelligenz
WL	W-Lernen

Literaturverzeichnis

[Anderson, 1993] Anderson, C. (1993). Q-Learning with hidden-unit restarting. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *NIPS 5*, volume 5 of *Advances in Neural Information Processing Systems*, pages 81–88, San Mateo, CA. Morgan Kaufmann Publishers.

[Bacchus et al., 1996] Bacchus, F., Boutilier, C., and Grove, A. (1996). Rewarding behaviors. In *AAAI-96*, volume To appear.

[Baird, 1995] Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In Prieditis, A. and Russell, S., editors, *Proceedings of the Twelfth International Conference*, Machine Learning. Morgan Kaufman Publisher.

[Baird, 1999] Baird, L. C. (1999). *Reinforcement Learning through gradient descent*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[Balkenius and Morén, 1998] Balkenius, C. and Morén, J. (1998). Computational models of classical conditioning: A comparative study. In [Pfeifer et al., 1998].

[Barto, 1992] Barto, A. G. (1992). Reinforcement learning and adaptive critic methods. In [White and Sofge, 1992], chapter 12.

[Baum, 1998] Baum, E. B. (1998). Manifesto for an evolutionary economics of intelligence. In Bishop, C. M., editor, *Neural Networks and Machine Learning*, pages 285–344. Springer-Verlag.

[Baxter and Bartlett, 2000] Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning in pomdp's via direct gradient ascent. In *Proceedings of the 17th International Conference on Machine Learning*.

[Baxter and Bartlett, 2001] Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

[Baxter et al., 2001] Baxter, J., Bartlett, P. L., and Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

[Blumberg et al., 1996] Blumberg, B. M., Todd, P. M., and Maes, P. (1996). No bad dogs: Ethological lessons for learning in hamsterdam. In [Maes et al., 1996], pages 295–304.

[Boutilier, 1999] Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *IJCAI-99*.

[Bowling and Veloso, 2000] Bowling, M. and Veloso, M. M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical Report Technical report CMU-CS-00-165, Dept. of Computer Science, Carnegie Mellon University.

[Braitenberg, 1993] Braitenberg, V. (1993). *Vehikel: Experimente mit kybernetischen Wesen*. Rowohlt, Reinbek bei Hamburg.

[Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

[Brooks, 1991a] Brooks, R. A. (1991a). Intelligence without reason. Memo 1293, MIT AI Lab.

[Brooks, 1991b] Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159.

[Brooks, 2002] Brooks, R. A. (2002). Künstliche intelligenz und roboterentwicklung. In *Computer.Gehirn*, Paderborn. Schöningh.

[Brooks and Maes, 1990] Brooks, R. A. and Maes, P. (1990). Learning to coordinate behaviors. In *AAAI*, pages 796–802.

[Brooks and Mataric, 1993] Brooks, R. A. and Mataric, M. J. (1993). Real robots, real learning problems. In [Connell and Mahadevan, 1993c], pages 193–214.

[Brown et al., 1999] Brown, J., Bullock, D., and Grossberg, S. (1999). How the basal ganglia use parallel excitatory and inhibitory learning pathways to selectively respond to unexpected rewarding cues. Technical Report CAS/CNS-TR-99-011, Boston University, Department of Cognitive and Neural Systems, Boston, MA, USA.

[Bruske, 1998] Bruske, J. (1998). *Dynamische Zellstrukturen: Theorie und Anwendung eines KNN-Modells*. PhD thesis, Technische Fakultät der Christian-Albrechts-Universität zu Kiel.

[Bruske et al., 1997] Bruske, J., Ahrns, I., and Sommer, G. (1997). An integrated architecture for learning of reactive behaviors based on dynamic cell structures. *Robotics and Autonomous Systems*, 22:87–101.

[Caironi, 1997] Caironi, P. V. (1997). Gradient-based reinforcement learning: Learning combinations of control policies. Technical Report Technical Report 97.50, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy.

[Carpenter and Grossberg, 1991] Carpenter, G. A. and Grossberg, S. (1991). *Pattern Recognition by Self-organizing Neural Networks*. MIT Press.

[Claus and Boutilier, 1998] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of AAAI'98*.

[Cliff et al., 1994] Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., editors (1994). *From animals to animats 3: Proceedings of the Third Conference on Simulation of Adaptive Behavior*. MIT Press.

[Connell and Mahadevan, 1993a] Connell, J. H. and Mahadevan, S. (1993a). Introduction to robot learning. In [Connell and Mahadevan, 1993c], pages 1–18.

[Connell and Mahadevan, 1993b] Connell, J. H. and Mahadevan, S. (1993b). Rapid task learning for real robots. In [Connell and Mahadevan, 1993c], pages 105–140.

[Connell and Mahadevan, 1993c] Connell, J. H. and Mahadevan, S. (1993c). *Robot Learning*. Kluwer Academic Publishers.

[Corkill, 1991] Corkill, D. D. (1991). Blackboard systems. *AI Expert*, 6(9):40–47.

[Dayan and Hinton, 1993] Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *NIPS92*, volume 5 of *Advances in Neural Information Processing Systems*, pages 271–278, San Mateo, CA, USA. Morgan Kaufmann Publishers.

[Der and Herrmann, 1994] Der, R. and Herrmann, M. (1994). Q-learning chaos controller. In *Proc. WCNN'94, Orlando, FL*, pages 2472–2475.

[Der and Herrmann, 1995] Der, R. and Herrmann, M. (1995). Efficient Q-Learning by division of labour. In *Proc. International Conference on Artificial Neural Networks - ICANN95*, volume II, pages 129–134, Paris.

[Dietterich, 2000] Dietterich, T. G. (2000). State abstraction in maxq hierarchical reinforcement learning. In [Solla et al., 2000].

[Dietterich and Wang, 2002] Dietterich, T. G. and Wang, X. (2002). Batch value function approximation via support vectors. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *NIPS01*, volume 14 of *Advances in Neural Information Processing Systems*, Cambridge, MA, USA. The MIT Press.

[Digney, 1996] Digney, B. L. (1996). Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In [Maes et al., 1996], pages 363–372.

[Digney, 1998] Digney, B. L. (1998). Learning hierarchical control structures for multiple tasks and changing environments. In [Pfeifer et al., 1998], pages 321–330.

[Digney and Gupta, 1994] Digney, B. L. and Gupta, M. M. (1994). A distributed adaptive control system for a quadruped mobile robot. In [Cliff et al., 1994], pages 344–353.

[Dorigo and Colombetti, 1997] Dorigo, M. and Colombetti, M. (1997). *Robot Shaping: An experiment in Behavior Engineering*. MIT Press/Bradford Books, Cambridge, MA, USA.

[Dörner, 1999] Dörner, D. (1999). *Bauplan für eine Seele*. Rowohlt, Reinbek bei Hamburg.

[Dörner, 2001] Dörner, D. (2001). ψ Eine neuronale theorie der handlungregulation. Technical report, Institut für Theoretische Psychologie, Otto-Friedrich-Universität Bamberg.

[Dörner et al., 1997] Dörner, D., Hamm, A., and Hille, K. (1997). Emoregul. Technical report, Institut für Theoretische Psychologie, Otto-Friedrich-Universität Bamberg.

[Durfee et al., 1989] Durfee, E. H., Lesser, V. R., and Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering, KDE*, 1(1):63–83.

[Ferber, 2001] Ferber, J., editor (2001). *Multiagentensysteme*. Addison-Wesley.

[Finton, 2002] Finton, D. (2002). *Cognitive Economy and the Role of Representation in On-Line Learning*. PhD thesis, Dept. of Computer Science, University of Wisconsin.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D., and Leen, T., editors, *NIPS95*, volume 7 of *Advances in Neural Information Processing Systems (NIPS'95)*. MIT Press.

[Gmytrasiewicz and Durfee, 2000] Gmytrasiewicz, P. J. and Durfee, E. H. (2000). Rational coordination in multi-agent systems. *Autonomous Agents and Multi-Agent Systems Journal*, 3(4):319–350.

[Gonzalez and Woods, 1993] Gonzalez, R. C. and Woods, R. E. (1993). *Digital image processing*. Addison-Wesley, Reading, MA, USA.

[Gross et al., 1996] Gross, H.-M., Stephan, V., and Boehme, H.-J. (1996). Sensory-based robot navigation using self-organizing networks and Q-Learning. In *Proc. WCNN'96, World Congress on Neural Networks, September 1996, San Diego, U.S.A.*, pages 94–99. Lawrence Erlbaum Associates, Inc.

[Grossberg and Schmajuk, 1987] Grossberg, S. and Schmajuk, N. A. (1987). Neural dynamics of attentionally modulated pavlovian conditioning: conditioned reinforcement, inhibition, and opponent processing. *Psychobiology*, 15(3):195–240.

[Grudic and Ungar, 2000] Grudic, G. R. and Ungar, L. H. (2000). Localizing policy gradient estimates to action transitions. In *Seventeenth Int. Conf on Machine Learning (ICML2000)*, pages 343–350.

[Guestrin et al., 2002] Guestrin, C., Lagoudakis, M., and Parr, R. (2002). Coordinated reinforcement learning,. In *Proceedings of the 2002 AAAI Spring Symposium Series: Collaborative Learning Agents*.

[Hauskrecht et al., 1998a] Hauskrecht, M., Meuleau, N., Boutilier, C., Kaelbling, L. P., and Dean, T. (1998a). Hierarchical solution of markov decision processes using macro-actions. In *Submitted*.

[Hauskrecht et al., 1998b] Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T., and Boutilier, C. (1998b). Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligenc*, pages 220–229.

[Houk et al., 1997] Houk, J. C., Buckingham, J. T., and Barto, A. G. (1997). Models of the cerebellum and motor learning. *Behavioral and Brain Sciences.*

[Hu and Wellman, 1998] Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98).*

[Hu and Wellman, 2001] Hu, J. and Wellman, M. P. (2001). Learning about other agents in a dynamic multiagent system. *Cognitive Systems Research*, 2(1):67–79.

[Huang et al., 1989] Huang, T. S., Kohonen, T., and Schroeder, M. R., editors (1989). *Self-Organization and Associative Memory.* Number 8 in Springer Series in Information Sciences. Springer-Verlag, Berlin Heidelberg New York etc., 3 edition.

[Huhns and Stephens, 1999] Huhns, M. N. and Stephens, L. M. (1999). Multiagent systems and societies of agents. In [Weiß, 1999], chapter 2, pages 79–120.

[Humphrys, 1997] Humphrys, M. (1997). *Action Selection methods using Reinforcement Learning.* PhD thesis, University of Cambridge, Computer Laboratory.

[Irwin et al., 1995] Irwin, G., Warwick, K., and Hunt, K. (1995). *Neural Network applications in control.* Number 53 in IEE Control Engineering Series. The Institution of Electrical Engineers.

[Jennings et al., 1998] Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Int Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Kearns et al., 1999] Kearns, M. S., Solla, S. A., and Cohn, D. A., editors (1999). *NIPS98*, volume 11 of *Advances in Neural Information Processing Systems*, Cambridge, MA, USA. The MIT Press.

[Kim, 1994] Kim, S. H. (1994). *Learning and Coordination.* Kluwer Academic Publishers, Dordrecht, Boston, London.

[Klein, 1991] Klein, S. B. (1991). *Learning: Principles and applications.* McGraw-Hill, New York, second edition.

[Klügl, 2000] Klügl, F. (2000). *Aktivitätsbasierte Verhaltensmodellierung und ihre Unterstützung bei Multiagentensimulationen.* PhD thesis, Institute of Computer Science, University of Würzburg.

[Koenig and Simmons, 1993] Koenig, S. and Simmons, R. G. (1993). Complexity analysis of real-time reinforcement learning. In *Proceedings of the 11th National Conference on AI*, pages 99–105.

[Kohonen, 1997] Kohonen, T., editor (1997). *Self-Organizing Maps.* Springer Series in Information Sciences. Springer-Verlag, Berlin Heidelberg New York etc.

[Kuipers et al., 1993] Kuipers, B., Froom, R., Lee, W.-Y., and Pierce, D. (1993). The semantic hierarchy in robot learning. In [Connell and Mahadevan, 1993c], pages 141–170.

[Lesser, 1999] Lesser, V. R. (1999). Cooperative multiagent systems: A personal view of the state of the art. *Knowledge and Data Engineering*, 11(1):133–142.

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

[Lin, 1993] Lin, L.-J. (1993). Scaling up reinforcement learning for robot control. In *Proceedings if 10th Int Conf on Machine Leanring*.

[Littman, 1994] Littman, M. L. (1994). Memoryless policies: theoretical limitations and practical results. In [Cliff et al., 1994], pages 238–245.

[Littman, 2001] Littman, M. L. (2001). Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66.

[Liu and Wu, 2001] Liu, J. and Wu, J. (2001). *Multi-Agent robotic systems*. International Series on Computational Intelligence. CRC Press, Boca Raton.

[Loch and Singh, 1998] Loch, J. and Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML'98*, volume to appear of *International Conference on Machine Learning*.

[Maclin and Shavlik, 1996] Maclin, R. and Shavlik, J. W. (1996). Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3):251–281.

[Maes, 1993] Maes, P. (1993). Behavior-based artificial intelligence. In *Proc. 2nd Conf on Adaptive Behavior*. MIT Press.

[Maes et al., 1996] Maes, P., Mataric, M. M., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors (1996). *From animals to animats 4: Proceedings of the Fourth Conference on Simulation of Adaptive Behavior*. MIT Press.

[Mahadevan, 1996a] Mahadevan, S. (1996a). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(Special Issue on Reinforcement Learning):159–196.

[Mahadevan, 1996b] Mahadevan, S. (1996b). Optimality criteria in reinforcement learning. In *Proceedings of the AAAI Fall Symposium on Learning Complex Behaviors for Intelligent Adaptive Systems*, Cambridge. MITPress.

[Mahadevan and Kaelbling, 1996] Mahadevan, S. and Kaelbling, L. P. (1996). Report of the 1996 workshop on reinforcement learning. Technical report, NSF.

[Martinetz and Schulten, 1991] Martinetz, T. and Schulten, K. (1991). A "Neural Gas" network learns topologies. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 397–402. Elsevier.

[Martinetz and Schulten, 1994] Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7:505–522.

[Mataric, 1995] Mataric, M. J. (1995). Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331.

[Mataric, 1996] Mataric, M. J. (1996). Behavior-based control: Examples from navigation, learning and group behavior. Technical report, Volen Center for Complex Systems, Computer Science Dept, Brandeis University.

[Mataric, 2001] Mataric, M. J. (2001). Learning in behavior-based multi-robot systems: policies, models, and other agents. *Cognitive Systems Research*, 2(1):81–93.

[McCallum, 1997] McCallum, A. K. (1997). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester.

[McGovern, 2002] McGovern, A. (2002). *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. PhD thesis, Computer Science Dept., University of Massachusetts at Amherst.

[McGovern et al., 1998] McGovern, A., Precup, D., Ravindran, B., Singh, S., and Sutton, R. S. (1998). Hierarchical optimal control of mdps. In *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, pages 186–191.

[McGovern and Sutton, 1998] McGovern, A. and Sutton, R. S. (1998). Macro-actions in reinforcement learning: An empirical analysis. In *Proceedings of the 15th International Conference on Machine Learning*, pages 556–564. Morgan Kaufmann.

[McGovern et al., 1997] McGovern, A., Sutton, R. S., and Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *The Grace Hopper Celebration Of Women in Computing*, San Jose, CA.

[Meuleau et al., 2001] Meuleau, N., Peshkin, L., and Kim, K.-E. (2001). Exploration in gradient-based reinforcement learning. Technical report, MIT, AI Lab.

[Meuleau et al., 1999] Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 427–436. Morgan Kaufmann.

[Meyer et al., 2000] Meyer, J.-A., Berthoz, A., Floriano, D., Roitblat, H., and Wilson, S. W., editors (2000). *From animals to animats 6: Proceedings of the Sixth Conference on Simulation of Adaptive Behavior*, volume Proceedings. MIT Press.

[Millman, 1980] Millman, D. (1980). *Way of the peaceful warrior*. H.J.Kramer, Inc., Tiburon, CA, USA.

[Minato and Asada, 1998] Minato, T. and Asada, M. (1998). Skill acquisition and self-improvement for environmental change adaptation of mobile robot. In [Pfeifer et al., 1998], pages 360–365.

[Möller, 1996] Möller, R. (1996). *Wahrnehmung durch Vorhersage: Eine Konzeption der handlungsorientierten Wahrnehmung*. PhD thesis, TU Ilmenau, FG Neuroinformatik.

[Narendra and Mukhopadhyay, 1994] Narendra, K. S. and Mukhopadhyay, S. (1994). Adaptive control of nonlinear multivariable systems using neural ntworks. *Neural Networks*, 7(5):737–752.

[Ono and Ikeda, 1996] Ono, N. and Ikeda, K. F. O. (1996). Collective behavior by modular reinforcement-learning agents. In [Maes et al., 1996], pages 618–624.

[Parr, 1998] Parr, R. E. (1998). *Hierarchical Control and learning for Markov decision processes*. PhD thesis, University of California at Berkeley.

[Pendrith and McGarity, 1997] Pendrith, M. D. and McGarity, M. (1997). An analysis of non-markov automata games: Implications for reinforcement learning. Technical Report UNSW-CSE-TR-9702, School of Computer Science and Engineering, University of New South Wales.

[Peshkin, 2001] Peshkin, L. (2001). *Proposal: Architectures for Policy Search*. PhD thesis, Brown University.

[Peshkin et al., 2000] Peshkin, L., Kim, K.-E., Meuleau, N., and Kaelbling, L. P. (2000). Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 489–496. Morgan Kaufmann.

[Pfeifer, 1996] Pfeifer, R. (1996). Building fungus eaters: Design principles of autonomous agents. In [Maes et al., 1996], pages 3–12.

[Pfeifer et al., 1998] Pfeifer, R., Bloomberg, B., Meyer, J.-A., and Wilson, S. W., editors (1998). *From animals to animats 5: Proceedings of the Fifth Conference on Simulation of Adaptive Behavior*. MIT Press.

[Pomierski, 1996] Pomierski, T. (1996). *Neurophysiologisch motivierte Architektur zur Erzeugung stabiler Farb- und Texturrepräsentationen*. PhD thesis, Ilmenau Technical University, Dept. of Neuroinformatics.

[Precup et al., 1998] Precup, D., S.Sutton, R., and Singh, S. P. (1998). Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of ECML98*, European Conference on Machine Learning. Springer Verlag.

[Precup et al., 2000] Precup, D., Sutton, R. S., and Singh, S. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 759–766. Morgan Kaufmann.

[Prescott et al., 1999] Prescott, T. A., Redgrave, P., and Gurney, K. (1999). Layered control architectures in robots and vertebrates. *Adaptive Behavior*, 7(1):99–127.

[Randlo, 1999] Randlo, J. (1999). Learning macro-actions in reinforcement learning. In [Kearns et al., 1999].

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco.

[Ripley, 1996] Ripley, B. D. (1996). *Pattern Recognition and neural networks.* Cambridge University Press, Cambridge, UK.

[Rodgríguez et al., 2000] Rodgríguez, A., Parr, R., and Koller, D. (2000). Reinforcement learning using approximate belief states. In [Solla et al., 2000].

[Rumelhart and McClelland, 1986] Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel Distributed Processing: Eplorations in the microstructure of cognition.* The MIT Press.

[Rummery and Niranjan, 1994] Rummery, G. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, UK.

[Salustowicz and Schmidhuber, 1998] Salustowicz, R. and Schmidhuber, J. (1998). Learning to predict through probabilistic incremental program evolution and automatic task decomposition. Technical Report IDSIA-11-98, IDSIA, Switzerland.

[Schumacher, 2001] Schumacher, M., editor (2001). *Objective Coordination in Multi-Agent System Engineering.* Number 2039 in Lecture Notes in Artificial Intelligence. Springer.

[Schwartz, 1993] Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 298–305. Morgan Kaufmann.

[Sen, 1997] Sen, S. (1997). Multiagent systems: Milestones and new horizons. *Trends in Cognitive Science*, 1(9):334–339.

[Seth, 1998] Seth, A. K. (1998). Evolving action selection and selective attention without actions, attention or selection. In [Pfeifer et al., 1998], pages 139–146.

[Singh and Sutton, 1996] Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

[Sofge and White, 1992] Sofge, D. A. and White, D. A. (1992). Applied learning-optimal control for manufacturing. In [White and Sofge, 1992], chapter 9.

[Solla et al., 2000] Solla, S. A., Leen, T. K., and Müller, K.-R., editors (2000). *NIPS99*, volume 12 of *Advances in Neural Information Processing Systems*, Cambridge, MA, USA. The MIT Press.

[Spier and McFarland, 1998] Spier, E. and McFarland, D. (1998). Learning to do without cognition. In [Pfeifer et al., 1998], pages 38–47.

[Stephan, 2002] Stephan, V. (2002). *Visuomotorische Antizipation: eine handlungsorientierte Sicht auf die visuelle Wahrnehmung.* Number 696 in Fortschritt-Berichte, Reihe 10. VDI Verlag, Düsseldorf.

[Stolzmann, 2000] Stolzmann, W. (2000). An introduction to Anticipatory Classifier Systems. In Lonzi, Stolzmann, and Wilson, editors, *Learning Classifier Systems: From foundations to applications*. Springer, Heidelberg.

[Surmeli, 2000] Surmeli, D. (2000). Reinforcement Lernen in komplexen Umwelten: Basiskonzepte und vergleichende Untersuchungen. In Gross, H.-M. and Böhme, H.-J., editors, *SOAVE2000 - Selbstorganisation von adaptivem Verhalten*, volume To appear of *Fortschrittberichte VDI Informatik/Kommunikationstechnik*. VDI-Verlag.

[Surmeli and Gross, 2000] Surmeli, D. and Gross, H.-M. (2000). Benchmarking reinforcement learning based on neural function approximators. In *Proceedings of the Forth International Conference on Cognitive and Neural Systems*, page 22, Boston. Boston University Press.

[Sutton, 1988] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

[Sutton, 1989] Sutton, R. S. (1989). Implementation details of the TD(λ) procedure for the case of vector predictions and backpropagation. Technical Report Technical Note TN87-509.1, GTE Laboratories.

[Sutton, 1997] Sutton, R. S. (1997). On the significance of markov decision processes. In Gerstner, W., A. Germond, M., and Nicoud, J.-D., editors, *ICANN'97*, International Conference on Artificial Neural Networks, pages 273–282. Springer.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An introduction*. MIT Press: A Bradford Book, Cambridge, MA, USA.

[Sutton et al., 2000] Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In [Solla et al., 2000], pages 1057–1063.

[Sutton et al., 1999] Sutton, R. S., Singh, S., Precup, D., and Ravindran, B. (1999). Improved switching among temporally abstract actions. In [Kearns et al., 1999].

[Tadepalli and Ok, 1998] Tadepalli, P. and Ok, D. (1998). Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224.

[Tan, 1993] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337.

[Thrun, 2000] Thrun, S. (2000). Probabalistic algorithms in robotics. Technical Report CMU-CS-00-126, School of Computer Science, Carnegie Mellon University.

[Thrun, 1992] Thrun, S. B. (1992). The role of exploration in learning control. In [White and Sofge, 1992], chapter 14.

[Todd et al., 1994] Todd, P. M., W.Wilson, S., Somayaji, A. B., and Yanco, H. A. (1994). The blind breeding the blind: Adaptive behavior without looking. In [Cliff et al., 1994], pages 228–237.

[Triesch and von der Malsburg, 2001] Triesch, J. and von der Malsburg, C. (2001). Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074.

[Tyrrell, 1993] Tyrrell, T. (1993). *Computational mechanisms for action selection.* PhD thesis, Centre for Cognitive Science, University of Edinburgh.

[Wang et al., 1995] Wang, H., Liu, G. P., Harris, C. J., and Brown, M. (1995). *Advanced adaptive control.* Pergamon (Elsevier Science Ltd.).

[Watkins, 1989] Watkins, C. J. (1989). *Learning from delayed rewards.* PhD thesis, King's College, Cambridge, UK.

[Weiß, 1999] Weiß, G., editor (1999). *Multiagent systems: A modern approach to distributed artificial intelligence.* MIT Press.

[Wengerek, 1996] Wengerek, T. (1996). *Reinforcement-Lernen in der Robotik.* Number 119 in Dissertationen zur Künstlichen Intelligenz (DISKI). infix, Sankt Augustin.

[Werbos, 1992a] Werbos, P. J. (1992a). Approximate dynamic programming for real-time control and neural modeling. In [White and Sofge, 1992], chapter 13.

[Werbos, 1992b] Werbos, P. J. (1992b). Neurocontrol and supervised learning: An overview and evaluation. In [White and Sofge, 1992], chapter 3.

[Werbos, 1995] Werbos, P. J. (1995). Optimal neurocontrol:Practical benefits, new results and biological evidence. Unpublished, available from WWW.

[White and Sofge, 1992] White, D. A. and Sofge, D. A. (1992). *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches.* van Nostrand Reinhold.

[Whitehead et al., 1993] Whitehead, S., Karlsson, J., and Tennenberg, J. (1993). Learning multiple goal behavior via task decomposition and dynamic policy merging. In [Connell and Mahadevan, 1993c], pages 45–78.

[Whitehead and Ballard, 1990] Whitehead, S. D. and Ballard, D. H. (1990). Active perception and reinforcement learning. *Neural Computation*, 2:409–419.

[Wiering and Schmidhuber, 1996] Wiering, M. and Schmidhuber, J. (1996). HQ-learning: Discovering markovian subgoals for non-markovian reinforcement learning. Technical report, IDSIA.

[Wiering and Schmidhuber, 1998] Wiering, M. and Schmidhuber, J. (1998). Fast online Q(λ). *Machine Learning*, 33(1):105–116.

[Williams, 1992] Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[Yamauchi and Beer, 1994] Yamauchi, B. and Beer, R. (1994). Integrating reactive, sequential and learning behavior using dynamical neural networks. In [Cliff et al., 1994], pages 382–391.

[Yokoo, 2001] Yokoo, M. (2001). *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems.* Springer.

[Zbikowski et al., 1994] Zbikowski, R., Hunt, K., Dzielinski, A., Murray-Smith, R., and Gawthrop, P. (1994). A review of advances in neural adaptive control systems. Technical report, ESPRIT III Project 8039: NACT.

Glossar

Agent:
computergestütztes System, das in eine Umgebung eingebettet ist und autonom variable Aktionen ausführen kann, 14

Animat:
mobiles autonomes System, 17

Arbiter:
zentrale Entscheidungsinstanz, 90

Autonomie:
Agent braucht keine Intervention, Anleitung, Kommunikation oder Instruktion, 15

Bump:
Experte bzw. Agent, der Wandvermeidung und schnelle, gerade Fahrt realisieren soll, 120

Clusterer:
Vektorquantisierer, fasst ähnliche Eingangsmuster zu Gruppen zusammen, 70

Deliberativität:
optimales, geplantes und erklärbares Verhalten, 16

Explorations/Exploitations-Dilemma:
Balance zwischen Erkundung des Zustandsraumes und Ausnutzung vorhandenen Verhaltens, 61

Flexibilität:
Agent setzt seine Aktionen variabel zur Erreichung der Ziele ein, 15

Flucht:
Experte bzw. Agent, der die Gefahrenzone bzw. den Jäger meiden soll, 121

Futterquelle:
statische Quelle positiven Rewards, blauer Farbfleck im Szenario, 107

Gefahrenzone:
statische Quelle negativen Rewards, roter Farbfleck im Szenario, 108

Hunger:
Experte bzw. Agent, der die Futterquelle bzw. das Wild erreichen soll, 121

Jäger:
dynamische Quelle negativen Rewards, roter Farbfleck im Szenario, 108

Körperhaftigkeit:
Agent ist ein physisches System in der realen Welt, 15

Kenetik:
problemunabhängige Interaktionsmechanismen zwischen mehreren physischen Agenten, 75

Koordination:
Algorithmus zur Koordinierung mehrerer Verhalten, 22

Leader:
Gewinner des Wettkampfes um Aktionsausführung in einem Zustand, 81

Markov Decision Process:
Umwelt, die die Markow-Eigenschaft erfüllt, 48

Markow-Eigenschaft:
der Zustand der Umwelt und die ausgeführte Aktion bestimmen vollständig den Folgezustand und den Reward, 48

Monolith:
realisiert komplexe Aufgabe in einer einzigen Struktur, 25

Multiagent MDP:
MDP mit mehreren Agenten, 76

Multiagentensystem:
Mehrere Agenten teilen eine Umwelt und interagieren miteinander, 24

Nash-Equilibrium:
optimale Kombination der Einzelpolicies mehrerer Agenten, 78

Opportunismus:
Animat unterbricht Verhalten, um sich bietende Chance zu nutzen., 23

Persistenz:
Animat vollendet ein konsumierendes Verhalten, 23

Policy:
Regel zur Aktionsauswahl in einem Zustand, 47

Priorisierung:
Verhalten des Animat folgt Rangfolge der Bedürfnisse, 23

Reaktivität:
schnell und flexibel reagieren, 16

Reward:
Belohnung oder Bestrafung, Bewertung einer Aktion des Agenten oder eines Umweltzustandes, 39

Semi-MDP:
MDP mit Makroaktionen verschiedener Länge, 68

Sensomotorische Paradigma:
Zwischen Sensorik und Aktuatorik soll nur eine minimale Abbildung stattfinden., 9

Situiertheit:
Agent überwacht seine eigene Interaktion mit der Umwelt, 15

Stabilitäts/Plastizitäts-Dilemma:
Balance zwischen Neulernen und Vergessen, 20

Superadditivität:
die Fähigkeiten des Gesamtsystems übersteigen die einer einfachen Überlagerung der Agentenfähigkeiten aufgrund emergenten Verhaltens, 80

Szenario:
Umwelt, in der ein Animat oder Agent Aufgaben erfüllt, 37

Trial:
Abschnitt vom Einsetzen des Roboters an einem Punkt der Umgebung bis zum Abbruch durch Kollision oder Überschreitung der Maximalschrittzahl (51), 125

active ss, der durchschnittliche erreichte Reward pro Schritt, gemittelt über alle Einsetzpunkte und mehrere Realisierungen, 125

Verhalten:
Regularität in der Dynamik der Interaktion zwischen Animat und Umwelt, 13

Wild:
dynamische Quelle positiven Rewards, blauer Farbfleck im Szenario, 108

emergentes Verhalten:
Verhalten des Gesamtsystems weist Charakteristika auf, die erst aus der Interaktion der Agenten entstehen, 24

probabilistische Robotik:
Methoden zur Robotersteuerung, die die Unsicherheiten der Sensoren und Aktuatoren explizit in Wahrnehmung und Aktionsplanung einbezieht. Statt bester Vertreter werden immer Wahrscheinlichkeitsverteilungen betrachtet., 6

symbolbasierte KI:
wissensbasierte Künstliche Intelligenz mit zentraler symbolischer Repräsentation, erzeugt optimale Lösungen für schwere Probleme, aber ohne Reaktivität und Körperhaftigkeit, 6

verhaltensbasierte Systeme:
Nachempfinden tierischer Intelligenz mit vollständigen, zunächst einfachen, Systemen mit Reaktivität und Körperhaftigkeit, ohne Optimalität der Lösungen., 7

zustandsabhängige Priorisierung:
Wichtung der Experten im W-Lernen, die zustandsabhängig adaptiert wird, 104

Index

Ψ, 87
'greedy'-Policy, 61
überwachtes Lernen, 45, 70, 72

action selection mechanism, 24
action selection problem, 22
Adaptive Heuristic Critic, 53
Adaptive Resonance Theory, 172
Adaptive Search Element, 53
Agent, 3, 14, 47, 96
    in DAI, 33
Agentenkonfiguration, 126
Anforderungen an die Verhaltenskombination, 23
Anforderungen an Koordination, 22
Animat, 17, 96
Antrieb, 166
Arbiter, 90, 91
Autonomie, 15

Backpropagation, 45
Basalganglien, 171
BDI, 33, 87
Beamer, 108
Bedürfnis, 13, 24, 46, 88, 165, 166
behavior selection problem, 22
behavior-based systems, 7
Benchmarks, 41
Beobachtung, 47
Berechnungskomplexität, 65
Bildverarbeitung, 113
Blackboard, 27, 28, 34, 173
Boltzman-Exploration, 62, 102, 158
Bump, 120

Classifier Systems, 45
Clusterer, 70, 73, 97, 120, 171

Deliberativität, 16
dezentrale Kontrollstrategie, 31
Diskontierungsfaktor, 49
Distributed Artificial Intelligence, 4, 33
Distributed Problem Solving, 33
Dynamische Programmierung, 52

Eigenschaften eines MAS, 24
Einsetzpunkte, 124
Einzelkörpersystem, 30, 77
Einzelpolicy, 76
Eligibility Traces, 54, 71, 72
emergente Modularisierung, 174
emergentes Verhalten, 24
Erhaltungsaufgabe, 40, 107
Erreichungsaufgabe, 40, 95, 107
Erwerb von Verhalten, 45
evolutionäre Algorithmen, 45
existentielle Antriebe, 167
experience replay, 66
Experimente, 123
    Festlegungen, 105
Exploration, 47, 89, 101
Explorations/Exploitations-Dilemma, 61
Explorationsstrategie, 61, 66
    'greedy-Policy, 61
    Boltzman, 62
    optimistische Initialisierung, 62
    zielgerichtete, 62

faktorisierte Policy, 80
Farbraum, 115
Farbverarbeitung, 114, 129
Flexibilität, 15
Flucht, 121
Futterquelle, 107, 118

Gefahrenzone, 108, 118
Generalisierung, 21, 98
genetische Algorithmen, 45
Genral sum game, 78
Gesamtreward, 87, 91, 100–102, 125
Global Positioning Systems, 132
Growing Neural Gas, 70, 126

Handlungsantrieb, 166
Hierarchie, 23, 29, 65, 80, 86
Hierarchie der kognitiven Verhaltensleistungen, 16
Hierarchie der Verhalten, 14
homöostatische Regelung, 88, 166
HSI-Transformation, 114
Hunger, 121

immediate Reward, 65
Interaktion, 34
Interaktionsarten, 34
interne Zustände, 14

Jäger, 108, 118

Körperhaftigkeit, 14, 15
Kenetik, 4, 75
Khepera, 108
    Grundmodul, 111
    Infrarotsensoren, 111
    Omnikamera, 111, 112
kognitive Antriebe, 168
Kommunikation, 6, 34, 76
Kommunikationsarten, 34
Konditionierung, 13, 46, 89, 171
Koordination, 22, 65, 99, 100, 171, 173
    objektive, 30
    subjektive, 30

Leader, 81, 85, 101
Learning from easy missions, 66
learning with external critic, 66
Leistungsfähigkeit, 135
Lernen, 19, 75
Lernen in MAS, 75
Lernen von Verhalten, 46

Makroaktion, 68, 80
Markov Decision Process, 48
Markow, 7, 38, 54, 64, 76, 79
Markow-Eigenschaft, 48
Maschinenlernen, 2
Mehrkörpersystem, 30, 77
Merkmalsextraktion, 113, 114
MMDP
    nichtkooperative, 78
    vollständig kompetitive, 78
    vollständig kooperative, 79
Monolith, 25, 79, 99, 102, 122, 136
Multiagent MDP, 76
Multiagentensystem, 2–4, 24, 26
Multiagentensysteme, 75
Multilayer Perzeptrons, 57

Nachempfinden natürlicher Vorbilder, 1
Nash-Equilibrium, 76, 78
Navigationsaufgaben für Roboter, 40
Neural Gas, 70, 126
Neuroinformatik, 1, 10, 46, 69
neuronale Netze, 69
neuronale Netze, 22
neuronales Netz, 57
Nullsummenspiel, 78

Objekt, 38, 95
    dynamisch, 39, 96
    statisch, 39, 96
Omnibild, 113
Opportunismus, 23, 24, 86, 89, 169
Optimalitätsmetriken, 49
Optimalitätsprinzip, 51
optimistische Initialisierung, 62, 101, 158

Panoramabild, 113
Parcour, 123
Partially Observable Markov Decision Processes, 59
active sse, 42
Persistenz, 23, 89, 169
PG, 80
Policy, 47, 51, 56, 71
Policy Iteration, 52, 53, 58, 72
Policygradienten, 60
POMDP, 64, 80, 90, 130, 132, 170
Priorisierung, 23, 89, 169
probabilistische Robotik, 5, 6
Psi, 87

Q-Learning, 55, 98, 126

Raycasting, 120
reaktiv, 40, 96
Reaktivität, 15, 16, 85
Reaktivitäts/Deliberativitäts-Dilemma, 16
reale Umwelt, 2, 9, 97
realer Roboter, 2, 10, 97
Referenz, 131
Reflex, 13
Reinforcement Lernen, 2, 14, 21, 45, 95
rekurrente neuronale Netze, 45
Repräsentation
    in symbolischer KI, 6
    in BBS, 8
    in probabilistischer Robotik, 7
    in Verhalten, 14

Return, 47, 49
Reward, 39, 47
    durchschnittlicher, 56
rezeptive Felder, 113
RL-Algorithmus, 172
    direkter, 57
    residualer, 58
    residualer Gradienten-, 58
RL-Aufgabe, 47

Sarsa, 55, 98, 126
Semi-MDP, 68
Sensomotorische Paradigma, 9
sensomotorisches Paradigma, 10, 174
Sensorik
    globale, 38, 40
    lokale, 38, 40
Sequenz, 169
Simulation, 91, 97, 108, 118
Simulationen, 9
Situiertheit, 14, 15
sozial, 88
Spieltheorie, 76
Stabilitäts/Plastizitäts-Dilemma, 20, 71
State-Action-Pair, Zustands-Aktionspaar, 133
Steuerung der dynamischen Objekte, 109
stochastisches Spiel, 76
Strategie, 47
Subsumption, 8, 27, 29
Superadditivität, 80
symbolbasierte Künstliche Intelligenz, 4, 5
symbolbasierte KI, 6
Szenario, 37, 95, 105
    Implementation, 107
    Simulation, 118
    verwendetes, 105

Teampolicy, 76, 79
Temporal Difference, 69
Temporal Difference, 53
Testmethode, 123
Trainingsmodus, 127
Trial, 125
Trialreward, 125

Umgebung, 47
unüberwachtes Lernen, 70
unüberwachtes Lernen, 46
Unvorhersehbarkeit, 45

Value Iteration, 52, 72
Vektorquantisierer, 70, 126
Verhalten, 8, 13
Verhaltenkoordination, 22
verhaltensbasierte Systeme, 5, 7, 13, 22
Verhaltensbiologie, 22
Verhaltenskoordination, 24, 25
Vermeidungsaufgabe, 40, 95, 108
Verstärkungslernen, 46
verteilte Kontrollstrategie, 31
Verteilte Künstliche Intelligenz, 4, 33
Vorteile eines MAS, 26
Vortraining, 129

W-Lernen, 2, 24, 32, 75, 81, 95, 103, 122, 136, 170
W-Werte
    Auswertung, 85
    dynamische, 83
    statische, 82, 104
    Varianten, 86
Werkzeuge, 38
Wild, 108, 118
WQ, 102, 161

zentrale Kontrollstrategie, 31
zielgerichtete Exploration, 62
Zielvorstellung, 3
Zugänglichkeit der Umwelt, 38
Zustand, 47
zustandsabhängige Priorisierung, 104
    dynamisch, 105, 160
    statisch, 104, 159
Zustandsaktionswert, 48
    diskontiert, 53
Zustandsaktionswertfunktion, 48
Zustandswert
    nicht diskontiert, 56
Zustandswertfunktion, 48, 54