

# **Entwicklung einer semantischen Missionssteuerung für autonome Inspektionsroboter**

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik  
des Karlsruher Instituts für Technologie (KIT)

**genehmigte**

**Dissertation**

von

**Marco Ziegenmeyer**

aus Hildesheim

Tag der mündlichen Prüfung: 05.07.2011

Erster Gutachter: Prof. Dr.–Ing. Rüdiger Dillmann

Zweiter Gutachter: Prof. Dr.–Ing. Rudi Studer



Für meinen Vater



## Vorwort und Danksagung

Mein Interesse an der Künstlichen Intelligenz und insbesondere am Maschinellen Lernen wurde durch eine Vortragsreihe zu Neuronalen Netzen am Max-Planck-Institut für Strömungsforschung (heute Max-Planck-Institut für Dynamik und Selbstorganisation) in Göttingen geweckt. Nach meinem Wechsel nach Karlsruhe machte ich mich auf die Suche nach einem interessanten Hiwi-Job in diesem Bereich und wurde ziemlich schnell fündig. Die Abteilung Interaktive Diagnose- und Servicesysteme am FZI Forschungszentrum Informatik hatte eine Stelle im Bereich der automatischen intelligenten Datenauswertung zu vergeben. Mit Hilfe von Neuronalen Netzen wurden verschiedene Defekte in den Ultraschalldaten von Ölpipelines erkannt. Unter der Anleitung meines Betreuers Dr. J. Marius Zöllner, seines Kollegen Dr. Robert Suna sowie des damaligen Abteilungsleiters Dr. Karsten Berns befasste ich mich näher mit diesem sehr interessanten und spannenden Forschungsfeld. Nach einer Weile bekam ich den Auftrag, bei der Untersuchung eines neuartigen Ansatzes zur Klassifikation der Pipelinedaten mit Hilfe von Support-Vektor-Maschinen sowie bei der Implementierung eines entsprechenden Systems mitzuwirken. Diese Arbeiten führten schließlich zu meiner Diplomarbeit, in der ich mich näher mit der Adaption von Support-Vektor-Maschinen für reale Diagnoseanwendungen befasste.

Direkt im Anschluss an meine Diplomarbeit trat ich als wissenschaftlicher Mitarbeiter in die Fußstapfen meines Betreuers Dr. J. Marius Zöllner, der nach dem Ruf von Dr. Karsten Berns an die Universität Kaiserslautern in der Zwischenzeit die Abteilungsleitung übernommen hatte. Bereits in der Anfangszeit meiner Tätigkeit als wissenschaftlicher Mitarbeiter reifte die Idee, dass ein System, welches sich nicht nur auf die a posteriori Datenauswertung beschränken würde, sondern stattdessen Defekte noch während der Inspektion aktiv untersuchen könnte, einen erheblichen Mehrwert bieten würde. Da es sich bei den in der Pipelineinspektion eingesetzten so genannten „Molchen“ um rein passive Systeme handelt, die mit dem Öl durch die Pipeline gepumpt werden, kamen diese jedoch für die Entwicklung und prototypische Erprobung eines solchen Systems nicht in Frage. Bei der Suche nach geeigneten Alternativen geriet zunächst der mehrsegmentige Inspektionsroboter KAIRO II ins Visier. Dieser war zum damaligen Zeitpunkt jedoch noch im Aufbau begriffen und so fiel die Wahl schließlich auf die sechsbeinige Laufmaschine LAURON IV, welche bereits über ein grundlegendes Verhaltensrepertoire sowie umfangreiche Fähigkeiten zur lokalen Navigation verfügte.

Rückblickend waren die Jahre am FZI Forschungszentrum Informatik eine äußerst interessante und spannende Zeit, in der ich viele Dinge gelernt habe und mich auch persönlich weiterentwickeln konnte. Ich möchte mich an dieser Stelle recht herzlich bei all denjenigen be-

danken, die mich während dieser Zeit begleitet und unterstützt haben. Mein besonderer Dank gilt Herrn Prof. Dr. Rüdiger Dillmann für die Schaffung des sehr interessanten wissenschaftlichen Arbeitsumfelds sowie für die Ermöglichung und Betreuung meiner Promotion. Darüber hinaus danke ich Herrn Prof. Dr. Rudi Studer für die Übernahme des Koreferats sowie Herrn Prof. Dr. Peter Schmitt, Herrn Prof. Dr. Ralf Reussner, Herrn Prof. Dr. Sebastian Abeck und Herrn Prof. Dr. Bernhard Beckert für ihre Mitwirkung in der Prüfungskommission.

Recht herzlich möchte ich mich auch bei all meinen Kolleginnen und Kollegen für ihre Unterstützung bei den vielen kleinen und großen Dingen des täglichen Projektgeschäfts und des Wissenschaftsbetriebs bedanken. Insbesondere bei Klaus Uhl für die gemeinsame Erarbeitung der grundlegenden Ideen für die Steuerungsarchitektur und die Wissensbasis, bei Arne Rönna für die tatkräftige Unterstützung rund um die sechsbeinige Laufmaschine LAURON IV sowie bei Jan Oberländer für das eingehende Lektorat der Ausarbeitung. Auch für die tatkräftige Zuarbeit meiner Studentinnen und Studenten Sabine Saylor, Lars Pfozter, Benjamin Betram, Stephan Kluge, Ronald Mayer-Hermann und Mark Engelmann im Rahmen ihrer Studien- und Diplomarbeiten sowie Hiwi-Tätigkeiten sage ich herzlichen Dank. Georg Heppner danke ich für seinen unermüdlichen Einsatz bei den Experimenten und bei den Reparaturen von LAURON IV.

Ganz besonders möchte ich mich jedoch bei meiner Familie bedanken, die mir in all den Jahren Rückhalt gegeben hat und auch in schwierigen Zeiten immer für mich da war. Meinem Bruder André danke ich darüber hinaus für die Durchsicht des Manuskripts. Ein ganz großes und liebevolles Dankeschön geht an Claudia, die mir tapfer den Rücken freigehalten und auf vieles verzichtet hat.

*Marco Ziegenmeyer*

Achern, im November 2011

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Motivation und Zielsetzung . . . . .	1
1.2	Einordnung und wissenschaftlicher Beitrag . . . . .	3
1.3	Aufbau der Arbeit . . . . .	4
<b>2</b>	<b>Stand der Forschung</b>	<b>7</b>
2.1	Inspektion mit autonomen Robotersystemen . . . . .	7
2.1.1	Autonome Inspektionsroboter . . . . .	8
2.1.2	Steuerungsarchitekturen . . . . .	16
2.1.3	Sensordatenauswertung . . . . .	23
2.2	Wissensrepräsentation . . . . .	25
2.2.1	Ontologien . . . . .	26
2.2.2	Semantische Technologien . . . . .	31
2.2.3	Semantisches Wissen in der Robotik . . . . .	37
2.3	Planungsverfahren und Ausführungssysteme . . . . .	41
2.3.1	Hierarchische Aufgabennetzwerke . . . . .	41
2.3.2	Flexible Programme . . . . .	45
2.3.3	Nebenläufige Hierarchische Pläne . . . . .	47
2.4	Situationsbewertung und Entscheidungsfindung . . . . .	51
2.4.1	Bayes'sche Netzwerke . . . . .	51
2.4.2	Markow'sche Entscheidungsprozesse . . . . .	53
2.5	Zusammenfassung und Fazit . . . . .	59
<b>3</b>	<b>Entwurf der Steuerungsarchitektur</b>	<b>63</b>
3.1	Anforderungen . . . . .	63
3.2	Architektur der semantischen Missionssteuerung . . . . .	64
3.2.1	Struktureller Aufbau . . . . .	64
3.2.2	Aufgaben der einzelnen Komponenten . . . . .	65
3.2.3	Zusammenwirken der Komponenten . . . . .	69
3.3	Einordnung der Architektur . . . . .	71
3.4	Zusammenfassung . . . . .	73

<b>4</b>	<b>Entwurf der Wissensbasis</b>	<b>75</b>
4.1	Anforderungen . . . . .	75
4.1.1	Kompetenzfragen . . . . .	76
4.2	Struktureller Aufbau der Wissensbasis . . . . .	77
4.3	Modelliertes Wissen . . . . .	79
4.3.1	Basisontologie . . . . .	79
4.3.2	Kernontologie . . . . .	79
4.3.3	Domänenontologie . . . . .	87
4.4	Einordnung der Wissensbasis . . . . .	88
4.5	Zusammenfassung . . . . .	88
<b>5</b>	<b>Erzeugung und Ausführung von Inspektionsplänen</b>	<b>89</b>
5.1	Anforderungen . . . . .	89
5.1.1	Kriterien zur Charakterisierung von Planungsverfahren . . . . .	89
5.1.2	Bestimmung der Anforderungen . . . . .	91
5.2	Flexible Hierarchische Pläne . . . . .	92
5.3	Planerzeugung . . . . .	98
5.3.1	Übersicht über die Komponenten des Planers . . . . .	98
5.3.2	Initialisierung . . . . .	99
5.3.3	Vorverarbeitung . . . . .	100
5.3.4	Koordination . . . . .	105
5.3.5	Serialisierung . . . . .	112
5.4	Planausführung . . . . .	112
5.4.1	Umgang mit Fehlern und Ausnahmesituationen . . . . .	113
5.5	Einordnung des Planungsansatzes . . . . .	113
5.6	Synergetische Vernetzung von Planung und Inferenz . . . . .	115
5.7	Zusammenfassung . . . . .	116
<b>6</b>	<b>Aktive Untersuchung interessierender Entitäten</b>	<b>119</b>
6.1	Anforderungen . . . . .	119
6.2	Struktureller Ablauf der aktiven Untersuchung . . . . .	120
6.3	Verfahren zur aktiven Untersuchung . . . . .	120
6.3.1	Fusion der Datenauswertungsergebnisse . . . . .	122
6.3.2	Bewertung der fusionierten Datenauswertungsergebnisse . . . . .	124
6.3.3	Auswahl und Priorisierung von Inspektionsaufgaben . . . . .	126
6.4	Iterative Adaption der bestehenden Inspektionspläne . . . . .	126
6.5	Kontinuierliche Anpassung der Regelbasis . . . . .	127
6.6	Zusammenfassung . . . . .	128



<b>7</b>	<b>Umsetzung der semantischen Missionssteuerung auf LAURON IV</b>	<b>129</b>
7.1	Die sechsbeinige Laufmaschine LAURON IV . . . . .	130
7.1.1	Systemüberblick . . . . .	130
7.1.2	Verhaltensbasierte Steuerung . . . . .	131
7.1.3	Lokalisation und Umweltmodellierung . . . . .	134
7.1.4	Navigation . . . . .	135
7.2	Globale Navigation . . . . .	136
7.2.1	Erreichung vorgegebener Zielpunkte und Regionen . . . . .	136
7.2.2	Systematisches Absuchen von Regionen . . . . .	142
7.3	Detektion und Klassifikation von Abfall . . . . .	143
7.3.1	Detektion auffälliger Bildbereiche . . . . .	145
7.3.2	Klassifikation anhand von Farbtexturmerkmalen . . . . .	152
7.4	Umsetzung der Kernkomponenten . . . . .	159
7.4.1	Steuerungsarchitektur . . . . .	160
7.4.2	Wissensbasis . . . . .	168
7.4.3	Erzeugung und Ausführung von Inspektionsplänen . . . . .	173
7.4.4	Aktive Untersuchung interessierender Entitäten . . . . .	175
7.5	Zusammenfassung . . . . .	178
<b>8</b>	<b>Experimentelle Evaluierung</b>	<b>181</b>
8.1	Evaluationskriterien . . . . .	181
8.2	Simulationsmodus . . . . .	184
8.3	Evaluierung der Kernkomponenten . . . . .	185
8.3.1	Steuerungsarchitektur . . . . .	186
8.3.2	Wissensbasis . . . . .	186
8.3.3	Erzeugung und Ausführung von Inspektionsplänen . . . . .	187
8.3.4	Aktive Untersuchung interessierender Entitäten . . . . .	189
8.4	Evaluierung der erweiterten Fähigkeiten von LAURON IV . . . . .	190
8.4.1	Globale Navigation . . . . .	191
8.4.2	Detektion und Klassifikation von Abfall . . . . .	196
8.5	Evaluierung des Gesamtsystems . . . . .	201
8.5.1	Aktive Inspektion von Regionen . . . . .	201
8.6	Portierung auf den mehrsegmentigen Inspektionsroboter KAIRO II . . . . .	208
8.7	Zusammenfassung und Fazit . . . . .	212
<b>9</b>	<b>Zusammenfassung</b>	<b>219</b>
9.1	Wissenschaftliche Erkenntnisse der Arbeit . . . . .	222
9.2	Ausblick . . . . .	223

<b>A</b>	<b>Algorithmen für die nebenläufige hierarchische Koordination</b>	<b>225</b>
A.1	Muss Erreichen . . . . .	225
A.2	Muss Überschreiben . . . . .	226
A.3	Muss Zurücksetzen . . . . .	227
A.4	Kann Erreichen . . . . .	228
A.5	Kann Überschreiben . . . . .	229
A.6	Kann Zurücksetzen . . . . .	230
<b>B</b>	<b>Bilddaten zur Detektion und Klassifikation von Abfall</b>	<b>231</b>
B.1	Lerndaten . . . . .	231
B.1.1	Abfall . . . . .	231
B.1.2	Kein Abfall . . . . .	244
B.2	Validierungsdaten . . . . .	250
B.2.1	Abfall . . . . .	251
B.2.2	Kein Abfall . . . . .	254
<b>C</b>	<b>Ergebnisse des Trainings der SVM-Klassifikatoren</b>	<b>257</b>
C.1	Klassifikation Abfall / Kein Abfall . . . . .	257
C.2	Klassifikation der Abfallart . . . . .	259
C.3	Absicherung der Abfallart . . . . .	260
C.3.1	Abfallart Flasche . . . . .	260
C.3.2	Abfallart Papier . . . . .	261
C.3.3	Abfallart Plastiktüte . . . . .	262
C.3.4	Abfallart Tetrapak . . . . .	264
C.3.5	Abfallart Dose . . . . .	265
<b>D</b>	<b>Abbildungsverzeichnis</b>	<b>267</b>
<b>E</b>	<b>Tabellenverzeichnis</b>	<b>273</b>
<b>F</b>	<b>Algorithmenverzeichnis</b>	<b>275</b>
<b>G</b>	<b>Literaturverzeichnis</b>	<b>277</b>

# 1. Einleitung

## 1.1. Motivation und Zielsetzung

Die Inspektion von komplexen technischen Anlagen, wie zum Beispiel Abwasserkanälen, Pipelines, Kernkraftwerken, Hochspannungsleitungen oder Dämmen, ist eine sehr anspruchsvolle Aufgabe. In vielen Bereichen werden hierfür gegenwärtig teleoperierte Inspektionssysteme eingesetzt, die von erfahrenen menschlichen Operatoren fernbedient werden (siehe etwa [80]). Dies ist jedoch häufig mit einem sehr hohen Aufwand verbunden. Im Bereich der Kanalinspektion werden beispielsweise überwiegend kabelgebundene TV-Inspektionssysteme eingesetzt [14]. Aufgrund der Traktion des Schleppkabels besitzen diese eine maximale Reichweite von etwa 100 bis 200 Metern. Bei der Inspektion von größeren Rohrleitungsnetzen ist somit ein häufiges Umsetzen des Inspektionssystems notwendig. Ein autonomer Inspektionsroboter bietet hier viele Vorteile. Neben der größeren Reichweite kann ein solches System zudem in weniger befahrenen Seitenstraßen eingesetzt werden und selbständig zum vorgegebenen Einsatzort fahren, wodurch das Ausmaß der Verkehrsbeeinträchtigungen reduziert wird. Wie eine in [68] vorgestellte Untersuchung zeigt, besteht bei der herkömmlichen Befahrung mit TV-Inspektionssystemen darüber hinaus eine erhebliche Abhängigkeit der Inspektionsergebnisse vom so genannten Fernaugenführer. Im Rahmen der Untersuchung wurden 307 Kanalhaltungen innerhalb eines Zeitraums von weniger als drei Jahren wiederholt inspiziert. Bei der Gegenüberstellung der Protokollierungen der jeweils doppelt erfassten Kanalhaltungen wurde festgestellt, dass die Kanalhaltungen in über 50 % der Fälle bei beiden Inspektionen nicht in die gleiche Zustandsklasse eingestuft wurden. Selbst wenn Kanalhaltungen in die gleiche Zustandsklasse eingestuft wurden, wiesen die Zustandsprotokollierungen zum Teil erhebliche Unterschiede auf. Sogar beim Vergleich der Protokollierungen von Inspektionen durch ein und dieselbe Person zeigten sich zum Teil deutliche Abweichungen. Geht man davon aus, dass jährlich etwa 5 % des deutschen Kanalnetzes inspiziert werden, so entspricht dies einer Gesamtlänge von etwa 26.000 km. Nimmt man für die Kosten zur Inspektion und zur erforderlichen vorherigen Reinigung wie in [68] angegeben etwa 1.600 €/km bis 4.000 €/km an, so liegen die Gesamtinspektionskosten zwischen 42 und 104 Millionen Euro. Schätzt man den Anteil der fehlerhaften Inspektionen gemäß der oben genannten Untersuchung weiterhin mit 50 % ab, so ergeben sich jährliche Fehlinvestitionen in Höhe von 21 bis 52 Millionen Euro. Durch eine verlässliche und robuste automatische Datenerfassung und -auswertung an Bord eines autonomen Inspektionsroboters könnte somit die Objektivität, Vergleichbarkeit und Qualität der Inspektionsergebnisse

deutlich verbessert werden, was wiederum zu erheblichen Kosteneinsparungen führen würde.

Ziel dieser Arbeit ist die Entwicklung einer semantischen Missionssteuerung für autonome Inspektionsroboter. Es wird untersucht, welche Anforderungen eine Missionssteuerung erfüllen muss, um den in der teleoperierten Inspektion vom menschlichen Operator eingenommenen Platz zu füllen. Für die Auswertung und Beurteilung der vielfach komplexen Sensordaten wird von menschlichen Datenauswertungsexperten in der Regel auf umfangreiche Fachkenntnisse und Erfahrungen zurückgegriffen. So sind etwa für die Auswertung der Ultraschalldaten, die während der Inspektion von Ölpipelines mit so genannten Molchen aufgezeichnet werden, mehrwöchige Schulungen des Datenauswertungspersonals notwendig, um allein die elementaren Kenntnisse zu vermitteln. Darüber hinaus sollte ein autonomer Inspektionsroboter *interessierende Entitäten* (siehe Definition 1.1) aktiv untersuchen.

**Definition 1.1.** *Interessierende Entitäten*<sup>1</sup> (engl. *Entity Of Interest* (EOI)) sind der Gegenstand der Inspektion. Es kann sich hierbei um Defekte, Schäden, Anomalien oder sonstige Auffälligkeiten handeln, die gefunden und untersucht werden sollen.

Sind die Ergebnisse der Datenauswertung unsicher, so sollten weitere Untersuchungen durchgeführt werden, bis eine ausreichende Konfidenz erreicht wird. Zusätzliche Inspektionsaktionen können zum Beispiel in der Aktivierung und Verwendung weiterer Sensorik (welche etwa aus Gründen der Energiesparsamkeit standardmäßig ausgeschaltet ist), der Aufnahme weiterer Daten aus einer anderen Perspektive oder der Anwendung weiterer Datenauswertungsverfahren bestehen. Die Wahl der konkreten Inspektionsaktionen sollte dabei unter Berücksichtigung der jeweils vorliegenden Inspektionssituation erfolgen. In dieser Arbeit wird daher ein wissensbasierter Ansatz untersucht, der dem autonomen Inspektionssystem das menschliche Expertenwissen in Form eines *semantischen Inspektionsmodells* (siehe Definition 1.2) zur Verfügung stellt.

**Definition 1.2.** Ein *semantisches Inspektionsmodell* ist eine ontologiebasierte Beschreibung von Fach-, Erfahrungs- und Hintergrundwissen, welche ein autonomes Robotersystem in die Lage versetzt, interessierende Entitäten in den Sensordaten zu erkennen, zu beurteilen und aktiv zu untersuchen.

Die wesentlichen Thesen, deren Validierung sich diese Arbeit widmet, lauten wie folgt:

- Die Inspektion von komplexen Umgebungen ist mit Servicerobotern autonom plan- und durchführbar.
- Das für die Inspektion von komplexen Umgebungen notwendige Experten- und Hintergrundwissen kann einem autonomen System mit Hilfe eines semantischen Inspektionsmodells zur Verfügung gestellt werden.

---

<sup>1</sup>Entität [lat.-mlat.] die; -, -en: 1. Dasein im Unterschied zum Wesen eines Dinges (Philos.). 2. [gegebene] Größe. – Duden Fremdwörterbuch, 1990.

- Die Verwendung eines semantischen Inspektionsmodells führt zu einem autonomeren und intelligenteren Systemverhalten, erhöht die Transparenz für den Benutzer und führt zu einer verbesserten Wartbarkeit, Erweiterbarkeit sowie Flexibilität des Systems.

## 1.2. Einordnung und wissenschaftlicher Beitrag

Im Rahmen dieser Arbeit wird untersucht, welche Verfahren geeignet sind, um den in Abbildung 1.1 dargestellten Regelkreis bestehend aus Inspektionsplanung, Planausführung, Inspektionsdatenauswertung, Bewertung der Datenauswertungsergebnisse, Entscheidungsfindung und Neuplanung an Bord des Roboters zu schließen: Für eine Menge vom Benutzer vorgegebener Inspektionsaufgaben wird ein geeigneter Plan zur Lösung der gestellten Aufgaben erzeugt (*Inspektionsplanung*). Dieser wird anschließend autonom vom System ausgeführt (*Planausführung*). Die während der Planausführung gewonnenen Inspektionsdaten werden direkt an Bord ausgewertet (*Inspektionsdatenauswertung*). Anschließend werden die Ergebnisse der Datenauswertung bewertet (*Bewertung der Ergebnisse*) und es wird autonom eine Entscheidung getroffen (*Entscheidungsfindung*), ob und wenn ja wie die gefundenen interessierenden Entitäten weiter untersucht werden. Die hieraus resultierenden neuen Inspektionsaufgaben werden priorisiert und anschließend durch Neuplanung (*Inspektionsplanung*) in den bestehenden Missionsplan integriert. Im Mittelpunkt dieser Arbeit steht jedoch die Untersuchung von Verfahren, mit deren Hilfe dem System das für eine aktive Untersuchung interessierender Entitäten erforderliche menschliche Expertenwissen bestehend aus Fachkenntnissen, Erfahrungen und Hintergrundwissen zur Verfügung gestellt werden kann (*Semantisches Inspektionsmodell*).

Die vorliegende Arbeit gliedert sich in die in Abbildung 1.2 dargestellten vier Teilbereiche:

1. Entwurf einer Steuerungsarchitektur zur Realisierung des in Abbildung 1.1 dargestellten Regelkreises.
2. Entwurf einer Wissensbasis zur expliziten Repräsentation des für die Durchführung von autonomen Inspektionsmissionen notwendigen Experten- und Hintergrundwissens.
3. Erzeugung und Ausführung von Inspektionsplänen zur Durchführung von autonomen Inspektionsmissionen.
4. Aktive Untersuchung interessierender Entitäten durch Bewertung der Ergebnisse der Inspektionsdatenauswertung, Auswahl und Priorisierung von zusätzlichen Inspektionsaufgaben und Berücksichtigung dieser Aufgaben mittels Adaption der bestehenden Inspektionspläne.

Der wissenschaftliche Beitrag dieser Arbeit umfasst die folgenden Punkte:

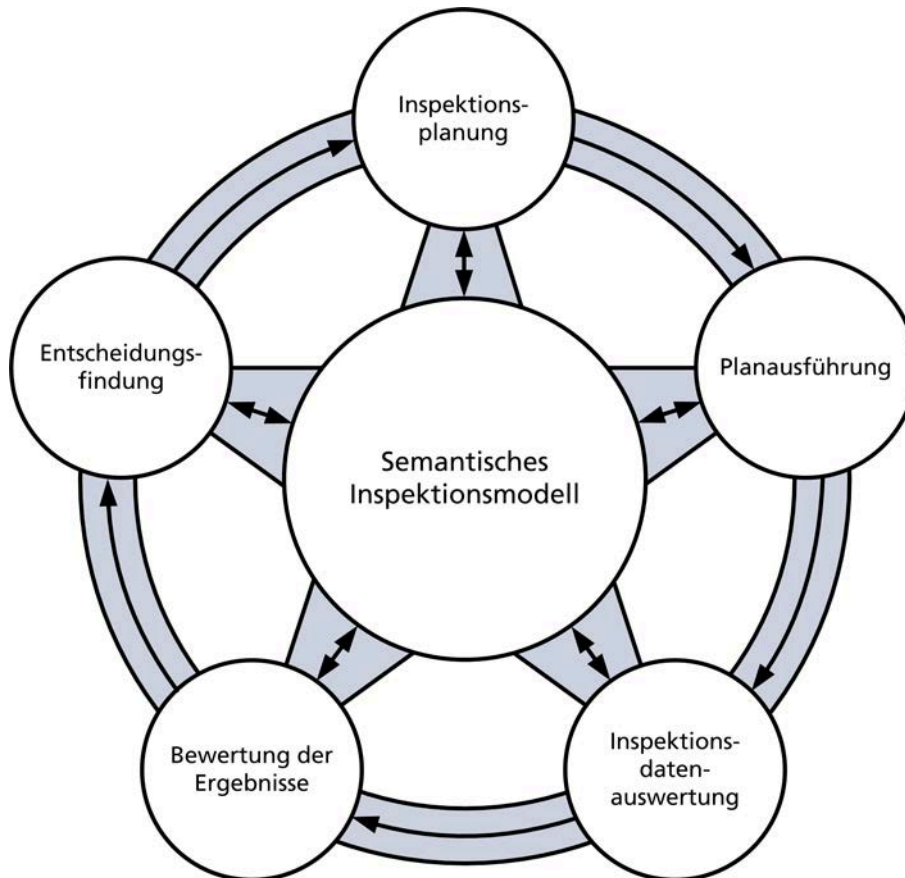


Abb. 1.1.: Regelkreis zur Inspektion von komplexen Umgebungen mit autonomen Servicerobotern.

- Konzeption einer modularen, auf Beschreibungslogiken basierenden Wissensbasis für die Inspektion von komplexen Umgebungen mit autonomen Servicerobotern.
- Entwicklung eines domänenkonfigurierbaren Planungsverfahrens auf Basis von Hierarchischen Aufgabennetzwerken, welches das in der Wissensbasis gespeicherte Problemlösungswissen zur Lösung von Inspektionsaufgaben nutzt.
- Entwicklung eines probabilistischen Verfahrens auf Basis von Bayes'schen Netzwerken zur Fusion der Datenauswertungsergebnisse sowie einer semantischen Regelbasis für die Auswahl und Priorisierung von Inspektionsaufgaben zur weiteren Untersuchung von interessierenden Entitäten.

### 1.3. Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in insgesamt neun Kapitel. Nach der Motivation der Problemstellung, der Formulierung der Zielsetzung sowie der Einordnung der Arbeit in diesem Kapitel, beschäftigt sich Kapitel 2 eingehend mit dem Stand der Forschung. Zunächst wird auf die Inspektion mit autonomen Robotersystemen eingegangen. Dabei werden insbesondere

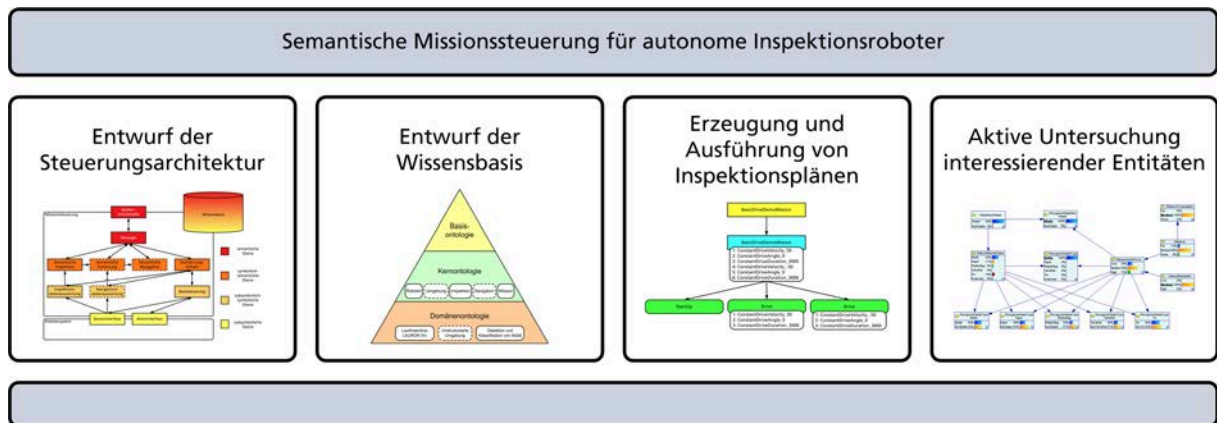


Abb. 1.2.: Übersicht über die vier Teilbereiche dieser Arbeit.

die verwendeten Steuerungsarchitekturen und die Verfahren zur Sensordatenauswertung näher untersucht. Daran anschließend werden grundlegende Verfahren und Techniken zur ontologiebasierten Wissensrepräsentation vorgestellt. Dies umfasst unter anderem verschiedene semantische Technologien aus dem Bereich des Semantic Web sowie eine Übersicht über die Verwendung von semantischem Wissen in der Robotik. Im Anschluss wird ein Überblick über aktuelle Planungsverfahren und Ausführungssysteme mit Schwerpunkt auf Hierarchischen Aufgabennetzwerken und verwandten Planrepräsentationen gegeben. Abschließend wird auf Verfahren zur Situationsbewertung und Entscheidungsfindung eingegangen. Dies beinhaltet im Wesentlichen Bayes'sche Netzwerke sowie Markow'sche Entscheidungsprozesse.

In Kapitel 3 wird der Entwurf der Steuerungsarchitektur der semantischen Missionssteuerung erläutert. Dazu wird zunächst auf die diesbezüglichen Anforderungen eingegangen. Daran anschließend wird der strukturelle Aufbau der Steuerungsarchitektur vorgestellt und das Zusammenwirken der Komponenten beschrieben. Abschließend wird eine Einordnung des Entwurfs vorgenommen.

In Kapitel 4 wird der Entwurf der Wissensbasis beschrieben. Auch hier wird zunächst auf die entsprechenden Anforderungen eingegangen, bevor der strukturelle Aufbau der Wissensbasis erläutert wird. Den Kern des Kapitels bildet die Beschreibung des modellierten Wissens in den einzelnen Ontologien der Wissensbasis.

In Kapitel 5 wird auf die Erzeugung und Ausführung von Inspektionsplänen eingegangen. Anhand verschiedener Kriterien zur Charakterisierung von Planungsverfahren werden zunächst die Anforderungen diskutiert. Anschließend werden die im Rahmen dieser Arbeit für die Repräsentation von Inspektionsplänen entworfenen *Flexiblen Hierarchischen Pläne* vorgestellt. Daran anknüpfend werden die Verfahren zur Erzeugung und Ausführung von Inspektionsplänen präsentiert. Abschließend wird eine Einordnung des gewählten Ansatzes vorgenommen und auf potentielle Synergien zwischen Planung und Inferenz eingegangen.

Kapitel 6 widmet sich der aktiven Untersuchung von interessierenden Entitäten. Zuerst wer-

den auch hier die entsprechenden Anforderungen erläutert. Daran anschließend wird der strukturelle Ablauf der aktiven Untersuchung vorgestellt und es wird näher auf die einzelnen Verfahren zur aktiven Untersuchung eingegangen. Dies umfasst insbesondere die Fusion und Bewertung der Datenauswertungsergebnisse, die Auswahl und Priorisierung von Inspektionsaufgaben sowie die Adaption der bestehenden Inspektionspläne. Abschließend werden Möglichkeiten zur kontinuierlichen Anpassung der verwendeten Regelbasis erläutert.

In Kapitel 7 wird die Umsetzung des entwickelten Konzepts der semantischen Missionssteuerung für die sechsbeinige Laufmaschine LAURON IV beschrieben. Als konkretes Inspektionsszenario wird die Detektion und Klassifikation von Abfall in unstrukturiertem Gelände gewählt. Zunächst wird ein kurzer Überblick über die Eigenschaften und Fähigkeiten des aktuellen LAURON IV-Systems gegeben. Daran anschließend werden die im Rahmen dieser Arbeit entwickelten Verfahren zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall vorgestellt, welche eine wesentliche Grundlage für die Durchführung von Inspektionsmissionen im Rahmen des gewählten Szenarios bilden. Nachfolgend wird die Umsetzung der vier Kernkomponenten der semantischen Missionssteuerung beschrieben, welche aus der Steuerungsarchitektur, der Wissensbasis, der Erzeugung und Ausführung von Inspektionsplänen sowie der aktiven Untersuchung von interessierenden Entitäten bestehen.

In Kapitel 8 wird die experimentelle Evaluierung der semantischen Missionsteuerung erläutert. Hierzu werden zunächst geeignete Evaluationskriterien definiert und die Eigenschaften des zur Evaluierung verwendeten Simulationsmodus der semantischen Missionssteuerung vorgestellt. Daran anknüpfend wird auf die Evaluierung der Kernkomponenten der semantischen Missionssteuerung hinsichtlich ihrer Funktionsfähigkeit und ihres Verhaltens in relevanten Inspektionssituationen eingegangen. Anschließend wird die Evaluierung der im Rahmen dieser Arbeit realisierten erweiterten Fähigkeiten von LAURON IV erläutert. Dies umfasst insbesondere die Fähigkeiten zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall. Nachfolgend wird die Evaluierung des Gesamtsystems unter kontrollierten Laborbedingungen sowie in unstrukturiertem Gelände beschrieben. Die Übertragbarkeit der semantischen Missionssteuerung auf andere Roboterplattformen und Inspektionsdomänen wird am Beispiel des mehrsegmentigen Inspektionsroboters KAIRO II veranschaulicht. Die erzielten Ergebnisse werden abschließend einer eingehenden Diskussion und Bewertung unterzogen.

Kapitel 9 fasst die wissenschaftlichen Erkenntnisse dieser Arbeit noch einmal zusammen und diskutiert die gegenüber dem Stand der Forschung erzielten Verbesserungen. Abschließend wird ein Ausblick auf weitere mögliche Arbeiten in diesem Bereich gegeben.



## 2. Stand der Forschung

In diesem Kapitel wird der Stand der Forschung in den für diese Arbeit relevanten Bereichen vorgestellt. Ausgehend von dem in Abbildung 1.1 dargestellten Regelkreis der autonomen Inspektion werden vier verschiedene Themenfelder behandelt.

Das erste Themenfeld widmet sich der Inspektion von komplexen Umgebungen mit autonomen Robotersystemen an sich und enthält einen Überblick über aktuell existierende Systeme. Daran anknüpfend werden verschiedene Arten von Steuerungsarchitekturen betrachtet, da eine geeignete Steuerungsarchitektur eine wesentliche Grundlage für die Umsetzung des Regelkreises bildet. Des Weiteren werden die zur Sensordatenauswertung verwendeten Verfahren und ihre Einbettung in die Steuerungsarchitekturen analysiert, womit die Komponente *Inspektionsdatenauswertung* des Regelkreises adressiert wird.

Das zweite Themenfeld beschäftigt sich mit grundlegenden Verfahren und Techniken zur ontologiebasierten Wissensrepräsentation und bezieht sich auf die zentrale Komponente *Semantisches Inspektionsmodell* des Regelkreises. In diesem Zusammenhang werden insbesondere semantische Technologien aus dem Bereich des Semantic Web vorgestellt. Außerdem wird die Verwendung von semantischem Wissen in der Robotik näher untersucht.

Das dritte Themenfeld adressiert die Erzeugung und Ausführung von Inspektionsplänen und setzt sich somit mit den Komponenten *Inspektionsplanung* und *Planausführung* des Regelkreises auseinander. An dieser Stelle wird auf aktuelle Planungsverfahren und Ausführungssysteme mit Schwerpunkt auf Hierarchischen Aufgabennetzwerken eingegangen.

Das vierte Themenfeld widmet sich abschließend grundlegenden Verfahren zur Situationsbewertung und Entscheidungsfindung und wendet sich somit den Komponenten *Bewertung der Ergebnisse* und *Entscheidungsfindung* des Regelkreises zu. Hier werden im Wesentlichen Bayes'sche Netzwerke und Markow'sche Entscheidungsprozesse sowie darauf basierende Ansätze vorgestellt.

### 2.1. Inspektion mit autonomen Robotersystemen

Die Inspektion von komplexen Umgebungen mit autonomen Robotersystemen ist ein Themenfeld, das in den letzten Jahren verstärkte Aufmerksamkeit erfahren hat und Gegenstand vieler aktueller Forschungsarbeiten ist. Aufgrund der technischen Fortschritte und der Verfügbarkeit von immer leistungsfähigeren Prozessoren ist die Realisierung von geeigneten Servicerobotern für dieses anspruchsvolle Aufgabengebiet in greifbare Nähe gerückt. Das herausragendste Bei-

spiel dafür, was in diesem Bereich möglich ist, bilden sicherlich die beiden Mars Rover *Spirit* und *Opportunity*, welche die in sie gesetzten Erwartungen um ein Vielfaches übertroffen haben.

### 2.1.1. Autonome Inspektionsroboter

Im Folgenden werden einige für diese Arbeit relevante autonome Inspektionsroboter vorgestellt. Zunächst wird dabei exemplarisch auf unbemannte autonome Helikopter zur Inspektion von Hochspannungsleitungen sowie mobile Roboter für die Inspektion und Manipulation im Offshore-Bereich eingegangen. Anschließend werden verschiedene mehrsegmentige Roboter zur Inspektion von Abwasserkanälen sowie die bereits angesprochenen Mars Rover und ihre Fähigkeiten betrachtet.

#### Unbemannte autonome Helikopter zur Inspektion von Hochspannungsleitungen

In [104] wird ein auf unbemannten autonomen Helikoptern (engl. *Unmanned Autonomous Helicopters* (UAHs)) basierendes Robotersystem zur Inspektion von Hochspannungsleitungen vorgestellt. Bisher wurden die notwendigen Inspektionen entweder manuell oder mit herkömmlichen Helikoptern durchgeführt. Die manuelle Inspektion ist jedoch sehr arbeitsintensiv und ineffizient, da die Inspektoren dem Verlauf der Hochspannungsleitungen häufig zu Fuß folgen müssen. Die Kosten für die Inspektion mit Helikoptern sind hingegen sehr hoch und die Flugsicherheit kann nicht immer vollständig gewährleistet werden, wie einige gemeldete schwere Unfälle zeigen. Die Vorteile der Inspektion mit einem unbemannten autonomen Helikopter liegen somit auf der Hand: eine höhere Effizienz, niedrigere Kosten und eine verbesserte Sicherheit. Das in [104] vorgestellte Robotersystem besteht aus einem Flug- und einem Inspektionssystem. Das Flugsystem besteht aus dem unbemannten Helikopter und einer Flugsteuerung. Der Helikopter verfügt über eine Nutzlast von 18 kg inklusive Treibstoff und eine Reichweite von 20 km. Er ist mit GPS, Beschleunigungssensoren, Gyroskop, Kompass, Höhenmesser und Hall-Sensor ausgestattet. Das Flugsystem kann in drei verschiedenen Flugmodi betrieben werden. Im manuellen Modus werden die Servoaktuatoren direkt über eine Fernbedienung gesteuert. Dieser Modus wird in Notfallsituationen und zur Unterstützung beim Starten und Landen verwendet. Im semiautonomen Flugmodus werden von der Bodenstation Fluginstruktionen (wie zum Beispiel auf der Stelle schweben, vorwärts/rückwärts fliegen, steigen, nach links/rechts drehen) an den Helikopter gesendet. Im automatischen Flugmodus fliegt der Helikopter eine vorgegebene Route, die zuvor von der Flugsteuerungssoftware berechnet wurde, autonom ab. Das Inspektionssystem verwendet eine Videokamera (10 Megapixel, bis zu zwanzigfacher optischer Zoom) und eine Infrarotkamera zur Inspektion der einzelnen Bauelemente und Komponenten. Die Videokamera erzeugt sowohl ein Video als auch hochaufgelöste Einzelbilder, während die Infrarotkamera ein Video und Bilder mit thermalen Informationen liefert. Die zwei analogen Videos werden an Bord des Helikopters von einem Videoserver komprimiert und zusammen

mit den aufgenommenen Bildern und den thermalen Informationen mit Hilfe eines speziellen drahtlosen Kommunikationsmoduls an die Bodenstation übertragen und dort angezeigt. Die beiden genannten Inspektionsmittel sind laut den Autoren von [104] zur Bestimmung von einigen gängigen Defekten ausreichend. Das vorgestellte Robotersystem bietet somit einen guten Ausgangspunkt für die Anwendung von unbemannten autonomen Helikoptern in der Energiewirtschaft.



Abb. 2.1.: Der für das Robotersystem zur Inspektion von Hochspannungsleitungen verwendete unbemannte Helikopter [104].

### **Mobile Roboter für die Inspektion und Manipulation im Offshore-Bereich**

In [10] wird das Potential des Einsatzes von mobilen Servicerobotern in Offshore-Produktionsumgebungen für Öl und Gas analysiert. Die möglichen Anwendungen von mobilen Service Robotern reichen von einfachen visuellen Inspektionen bis hin zu physischen Eingriffen in die Verarbeitungsanlagen. Diese umfassen zum Beispiel die Entnahme von Proben, das Drehen von Ventilen, das Bereinigen kleinerer Verstopfungen sowie die Bedienung von Kontrollpulten. In [10] werden die benötigten Hardware- und Softwarekomponenten sowie die benötigten Fähigkeiten eines solchen mobilen Inspektions- und Manipulationsroboters für den Offshore-Bereich präsentiert. Der vorgestellte Prototyp eines mobilen Offshore-Inspektionsroboters MIMROex ist mit einem Roboterarm ausgestattet, an dem eine Kamera zur visuellen Inspektion befestigt ist. Darüber hinaus verfügt er über verschiedene Anwendungssensoren wie Mikrofone und Gas- und Feuersensoren. Er ist in der Lage, sowohl teleoperierte als auch autonome Inspektionen von industriellen Verarbeitungsanlagen durchzuführen. Im automatischen Modus führt der Roboter autonom vorprogrammierte Inspektionsaufgaben durch. Die Ergebnisse der Inspektionsaufgaben werden in einer Datenbank gespeichert und können jederzeit vom verantwortlichen Operator im Kontrollraum begutachtet werden. Die Evaluierung des ersten autonomen mobilen Roboters, der jemals in einer Offshore-Umgebung eingesetzt wurde, hat die prinzipielle Anwendbarkeit der mobilen Robotik in Offshore-Umgebungen gezeigt. Der Einsatz von mobilen

Robotern kann die Anzahl der zum Betrieb von Produktionsanlagen notwendigen menschlichen Interventionen verringern und so die Effizienz steigern. Darüber hinaus kann er zur Verbesserung der Sicherheits- und Arbeitsbedingungen beitragen.



Abb. 2.2.: Der Offshore-Inspektionsroboter MIMROex [10].

### **Mehrsegmentige Roboter zur Inspektion von Abwasserkanälen**

**Kairo** Der mehrsegmentige Roboter KAIRO (*Karlsruher Inspektionsroboter*) [95] wurde für die autonome Inspektion von Kanalsystemen mit Rohrdurchmessern zwischen 300 und 600 Millimetern entwickelt. Durch die Fähigkeit einzelne Segmente anzuheben und abzuknicken ist der Roboter in der Lage, auch Abzweigungen zu folgen, die nicht sohlengleich angebracht sind. Darüber hinaus können Hindernisse und Stufen überwunden werden, indem Teile des Roboters gezielt angehoben oder abgesenkt werden.

KAIRO besteht aus fünf Antriebssegmenten und vier Knickelementen. Die Antriebssegmente verfügen über je zwei Antriebsräder und stellen den Nutzraum für Akkumulatoren, Steuerungsrechner, Mikrokontroller und Sensoren zur Verfügung. Die Knickelemente verbinden jeweils zwei Antriebssegmente miteinander und bestehen aus drei voneinander unabhängigen, aktiv angetriebenen Gelenken. Um die zum Anheben von Segmenten notwendigen hohen Kräfte erzeugen zu können, werden so genannte *Harmonic Drive* Getriebe eingesetzt. In den Kopfsegmenten befinden sich Lagesensoren, die es erlauben, die Lage des Roboters im Kanal zu kontrollieren. Darüber hinaus verfügt eines der Endsegmente über einen Kamerakopf, der sowohl horizontal als auch vertikal geschwenkt werden kann.

Für den mehrsegmentigen Roboter KAIRO wurde eine hierarchische Steuerungsarchitektur entwickelt. Die Regelung der Gelenkstellungen und Radgeschwindigkeiten ist auf verschiedene C167-Mikrokontroller verteilt. Sämtliche Koordinations- und Planungsaufgaben werden hingegen auf einem zentralen PC/104-System durchgeführt. Die Kommunikation zwischen den einzelnen Komponenten erfolgt mit Hilfe eines CAN-Busses. Basierend auf der Sensordatenauswertung und der Bewegungssteuerung wird ein Zustandsautomat eingesetzt, der als Missionssteuerung das Gesamtsystem durch ein Kanalsystem navigiert und Inspektionsaufgaben durchführt (siehe [95]).

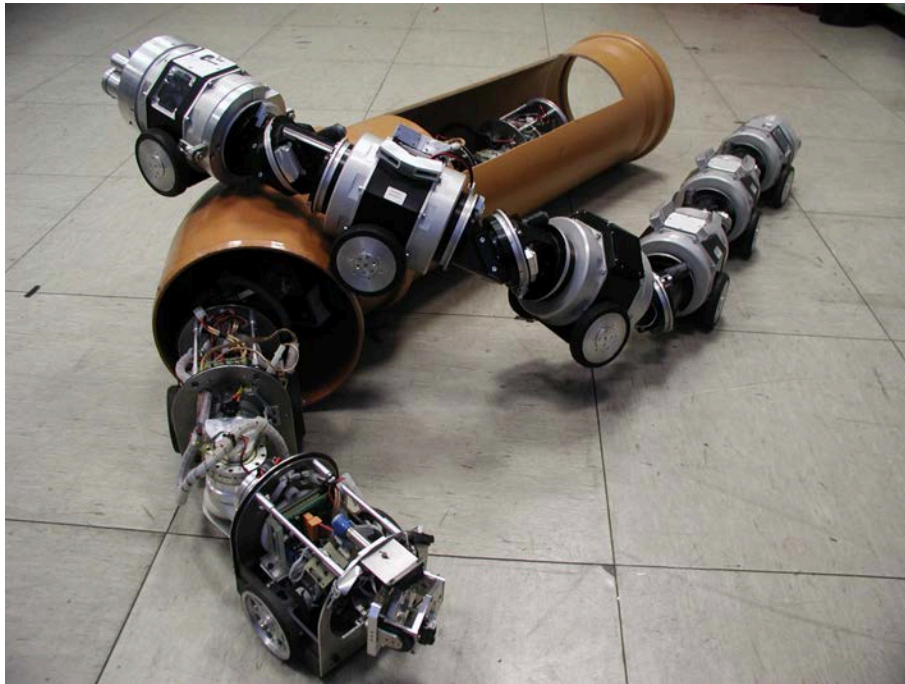


Abb. 2.3.: Die mehrsegmentigen Kanalinspektionsroboter KAIRO (unten) und MAKRO (oben).

**Makro** Basierend auf den mit KAIRO gemachten Erfahrungen wurde das Nachfolgemodell MAKRO (*Mehrgliedriger, autonomer Kanalroboter*) entwickelt (siehe [95] und [97]), das eine vollständige Kapselung gegen Flüssigkeiten besitzt. Die Anzahl der Antriebssegmente wurde auf sechs erhöht und es wurde eine andere Anordnung der Gelenke in den Knickelementen gewählt, die es ermöglicht Kabel durch die Gelenke hindurch zu führen.

MAKRO verfügt ähnlich wie KAIRO über sieben Mikrokontroller und ein PC/104-System, die über CAN-Bus miteinander kommunizieren. Die Steuerungsarchitektur von MAKRO besteht aus vier verschiedenen Abstraktionsebenen und ist speziell auf die autonome Steuerung des Kanalinspektionsroboters ausgelegt (siehe [97]). Die oberste Ebene wird dabei vom so genannten *Planer* gebildet, der Informationen über das Kanalnetz aus einer internen Datenbank empfängt und diese in ein symbolisches Weltmodell parst. Parallel hierzu werden die Ziele der Mission von einem menschlichen Operator definiert und anschließend zusammen mit dem

symbolischen Weltmodell an einen KI-Planer übergeben. Dieser erzeugt einen entsprechenden Plan und gibt die Ergebnisse an die nächste Ebene, den so genannten *Action Controller*, weiter. Die einzelnen Aktionen werden vom *Action Controller* wiederum in Jobs zerlegt. Schlägt ein Job fehl, so kann sich die Aktion in einem undefinierten Zustand befinden. Aus diesem Grund verfügt der *Action Controller* für jeden Job über eine spezifische Wiederherstellungsaktion. Die darunter liegende Ebene, die so genannte *Base Machine*, enthält alle Module, die für die Bewegungssteuerung benötigt werden und stellt verschiedene abstrakte Sensordaten, wie etwa die Detektion von gefährlichen Positionen oder von Hindernissen, zur Verfügung. Sie führt zu jedem Zeitpunkt genau einen Job aus, der jedoch verschiedene Module (*Verhalten*) simultan verwenden kann. Die Schnittstelle zur darunter liegenden Ebene, dem so genannten *Low-Level Controller*, ist wertebasiert. Der *Low-Level Controller* empfängt von der *Base Machine* Werte für alle Gelenke und Geschwindigkeiten für alle Motoren und sendet alle Sensorwerte, aktuellen Geschwindigkeiten und zurückgelegten Entfernungen an diese zurück.

Die Steuerungsarchitektur wurde mit Hilfe des Softwarerahmenwerks MCA2 [100] realisiert. Die Software läuft dabei auf RT-Linux<sup>2</sup>. Die Ebene des *Action Controllers* wurde mit Hilfe der synchronen Programmiersprache *sE* [22] entwickelt, welche die Erstellung hierarchischer Zustandsautomaten unterstützt. Als KI-Planer kommt das so genannte *FF-Planungssystem* [51] zum Einsatz, bei dem es sich um einen vorwärtsverkettenden heuristischen Zustandsraumplaner handelt.

**MakroPlus** Die Entwicklung des Nachfolgesystems MAKROPLUS erfolgte unter der Maßgabe, dass der Roboter auch in Abwasserkanälen eingesetzt werden kann, die sich in Betrieb befinden. Den Anforderungen dieses Inspektionsszenarios wurde unter anderem durch Entwicklungen in den Bereichen der internen Sensoren und der Basissteuerung Rechnung getragen [12]. Darüber hinaus wurden für MAKROPLUS zwei Anwendungsmodule entwickelt: Eins zur exakten Vermessung von Kanalhaltungen und Schächten und eins für die Analyse von Abwasser an Bord des Roboters [97]. Da MAKROPLUS jedoch keine Sensoren zur internen Zustandserkennung besitzt, können keine komplexen bzw. adaptiven Manöver durchgeführt werden und die Bewegungsplanung wurde in einem offenen Regelkreis rein gesteuert realisiert.



Abb. 2.4.: Der mehrsegmentige Kanalinspektionsroboter MAKROPLUS.

---

<sup>2</sup>RT-Linux: <http://www.rtlinuxfree.com/>

**Kairo II** Die Komponenten des nochmals weiterentwickelten Roboters KAIRO II entsprechen im Wesentlichen denen von MAKROPLUS. Die Elektromechanik des Roboters wurde jedoch so erweitert, dass sie den Anforderungen einer geregelten Inspektionsfahrt genügt (siehe [12]). So wird etwa der Bodenkontakt der Roboters durch die Messung des Motorstroms der Antriebsmotoren ermittelt. Darüber hinaus wurde die interne Sensorik zur absoluten Messung der Winkelpositionen in den Knickelementen erweitert und es wurden Dehnungsmessstreifen zur Detektion von mechanischen Zuständen in die Knickelemente integriert. Des Weiteren wurden die Antriebssegmente um einen zusätzlichen Bewegungsfreiheitsgrad *Flipper* erweitert, wodurch Kontaktpunkte des Roboters mit dem Untergrund gezielt gesucht und angefahren werden können. Basierend auf diesen elektromechanischen Verbesserungen wurde eine adaptive Steuerung des mehrsegmentigen Inspektionsroboters entwickelt, mit deren Hilfe es möglich ist, die zentralen Aufgabenklassen der Inspektion von schwer zugänglichen Bereichen regelungs- und steuerungstechnisch abzudecken [12].



Abb. 2.5.: Der mehrsegmentige Kanalinspektionsroboter KAIRO II.

### Mars Rover

Die zur Planetenerkundung eingesetzten Rover (siehe Abbildung 2.6) erfordern einen hohen Autonomiegrad, um in anspruchsvollem und unbekanntem Gelände navigieren, Zielobjekte untersuchen und wissenschaftlich interessierende Ereignisse detektieren zu können [8]. Durch die geringe Bandbreite und die hohen Latenzen der Kommunikationskanäle ist eine direkte, teileoperierte Steuerung der Rover nicht möglich. Die Signallaufzeiten vom Mars zur Erde und

zurück betragen zwischen 8 und 42 Minuten. Darüber hinaus ist die Kommunikation nur einige Male im Laufe eines Marstages möglich. Aus diesem Grund ist es erforderlich, die Operationen der Rover für einen ganzen Marstag im Voraus zu planen und zu übertragen. Anschließend muss die Ausführung ohne menschliche Überwachung oder Bestätigung abgewartet werden.



Abb. 2.6.: Modelle der aktuellen Mars Rover [8]. Mitte: Sojourner Rover der NASA Mars Pathfinder Mission, 1997. Links: Mars Exploration Rover (MER), 2004. Rechts: Mars Science Laboratory (MSL), 2011.

Die Erkundung der Oberfläche fremder Planeten birgt neben den eingeschränkten Kommunikationsmöglichkeiten auch viele Herausforderungen im Bereich der Mobilität und der Sensorwahrnehmung. So muss ein Rover das Gelände unter stark variierenden Beleuchtungsverhältnissen wahrnehmen, mit nicht vollständig charakterisiertem Gelände interagieren und Unsicherheiten in der Sensorwahrnehmung sowie Systemfehler berücksichtigen. Darüber hinaus sind die Rechenkapazitäten der Rover aufgrund der hohen Strahlung und der großen Temperaturschwankungen extrem begrenzt (siehe Tabelle 2.1).

**Sojourner** Der im Juli 1997 als Teil der NASA Pathfinder Mission eingesetzte *Sojourner* Rover ist das erste Raumfahrzeug, welches autonom auf einem anderen Planeten gefahren ist. Die autonomen Fähigkeiten des Rovers bestanden aus der Navigation im Gelände, der Behandlung



	<b>Sojourner</b>	<b>MER</b>	<b>MSL</b>
<b>CPU</b>	0,1 MHz Intel 80C85	20 MHz RAD6000	200 MHz RAD750 PowerPC
<b>Arbeitsspeicher</b>	512 KB	128 MB	256 MB
<b>Flash-Speicher</b>	176 KB	256 MB	512 MB

Tab. 2.1.: Die technischen Daten der Mars Rover [8].

von Ausnahmefällen sowie der Verwaltung von Ressourcen. Sojourner konnte autonom durch flaches, aber felsiges Marsgelände zu von der Erde aus vorgegebenen Positionen navigieren. Zur Erkennung von Hindernissen verwendete er Stereokameras sowie fünf infrarote Streifenlaser. Es wurde keine dauerhafte Geländekarte erstellt und daher handelte es sich um ein rein reaktives System.

**Mars Exploration Rover (MER)** Verglichen mit dem *Sojourner* Rover verfügen die im Januar 2004 gelandeten *Mars Exploration Rover (MER) Spirit* und *Opportunity* über eine deutlich umfangreichere Ausstattung mit autonomen Fähigkeiten. Während der 90 Marstage dauernden primären Mission umfassten diese initial die sichere Navigation im Gelände unter Berücksichtigung geometrischer Gefahren, die visuelle Schätzung der Roboterpose sowie die Bestimmung der absoluten Orientierung. Durch Aktualisierungen der Software während der erweiterten Mission wurden diese später um einen globalen Pfadplaner zur verbesserten Navigation und neue Fähigkeiten zum Annähern und Platzieren von Instrumenten auf einem Zielobjekt sowie zur Erkennung von wissenschaftlich interessierenden Ereignissen ergänzt.

**Mars Science Laboratory (MSL)** Im Herbst 2011 plant die NASA den *Mars Science Laboratory (MSL) Rover* zu starten. Die primäre Mission wird zwei Jahre dauern und der Erkundung der Marsoberfläche dienen. Der MSL Rover wird in der Lage sein, Gesteinsproben zu sammeln und zu analysieren. Er wird auf den autonomen Fähigkeiten der MER Rover aufbauen, um denselben Grad an Performanz sicherzustellen [8].

**Mars Sample Return (MSR)** In naher Zukunft wird im Rahmen der *Mars Sample Return (MSR) Mission*, einem gemeinsamen Projekt der NASA und der ESA, ein leichter Rover zum Einsatz kommen. Dieser wird vielfältige Gesteinsproben sammeln und sie zu einer Raumfähre bringen, welche zur Erde zurückkehren wird. Der Rover wird über ein ungekanntes Maß an Autonomie verfügen, da die Lebenszeit einer Rakete auf der Marsoberfläche begrenzt ist und der Wunsch besteht, Gesteinsproben aus entfernten Kraterwänden zu entnehmen. Der Rover muss daher in kurzer Zeit zu vielen, bis zu fünf Kilometer von der Landestelle entfernten Orten fahren können. Darüber hinaus muss der Rover trotz seines geringen Gewichts in der Lage sein, Kernbohrungen in hartem Gestein von potentiell instabilen Positionen aus durchzuführen.

### 2.1.2. Steuerungsarchitekturen

Softwaresysteme für Roboter sind üblicherweise mit einer hohen Komplexität behaftet. Eine der wesentlichen Ursachen hierfür ist laut [63], dass Roboter häufig asynchron und in Echtzeit mit einer unsicheren und in der Regel dynamischen Umwelt interagieren müssen. Des Weiteren müssen viele Robotersysteme mit variierenden Antwortzeiten reagieren können, die von Regelzeiten im Millisekundenbereich bis hin zu Minuten oder Stunden für komplexere Aufgaben reichen.

Eine saubere, gut durchdachte Architektur zusammen mit geeigneten Werkzeugen kann laut [63] dabei helfen, diese hohe Komplexität zu beherrschen. Darüber hinaus kann sie signifikante Vorteile bei der Spezifikation, der Umsetzung und der Validierung von Robotersystemen bieten.

Zur Erfüllung der genannten Anforderungen beinhalten Steuerungsarchitekturen für Roboter in der Regel Fähigkeiten, um in Echtzeit zu agieren, Sensoren und Aktuatoren anzusteuern, Nebenläufigkeiten zu unterstützen, Ausnahmesituationen zu erkennen und darauf zu reagieren, mit Unsicherheiten umzugehen und symbolische Planung auf höheren Ebenen mit numerischer Steuerung auf niedrigeren Ebenen zu verzahnen.

Zu den weit verbreiteten Eigenschaften von Steuerungsarchitekturen für Roboter gehört laut [63] zudem ein modularer Aufbau, der das System in kleinere, größtenteils voneinander unabhängige Teile strukturiert, um auf diese Weise die Gesamtkomplexität des Systems zu verringern und die Gesamtzuverlässigkeit zu erhöhen. Häufig erfolgt diese Zerlegung des Gesamtsystems in einer hierarchischen Art und Weise, indem modulare Komponenten auf anderen modularen Komponenten aufbauen. Die Abstraktion kann dabei entlang unterschiedlicher Dimensionen erfolgen. In einigen Steuerungsarchitekturen wird eine zeitliche Dimension gewählt, so dass jede Ebene in der Hierarchie mit einer eigenen Frequenz arbeitet, die in der Regel von unten nach oben abnimmt. In anderen Steuerungsarchitekturen wird der Abstraktionsgrad von Aufgaben zur Strukturierung genutzt, indem Aufgaben auf den verschiedenen Ebenen durch Unteraufgaben auf den darunter liegenden Ebenen erfüllt werden. In manchen Fällen ist auch eine Zerlegung anhand einer räumlichen Abstraktion sinnvoll, wie etwa bei einer Unterteilung in globale und lokale Navigation.

### Allgemeiner Überblick

In Anlehnung an [63] wird im Folgenden ein kurzer Abriss über die verschiedenen Arten von Steuerungsarchitekturen und ihre wichtigsten Vertreter gegeben. Dabei wird auf deliberative sowie auf reaktive und verhaltenbasierte Architekturen eingegangen. Den Schwerpunkt bilden jedoch die in jüngerer Zeit meist verwendeten mehrschichtigen Architekturen. Einen Überblick über die wichtigsten Repräsentanten der einzelnen Architekturarten bietet auch Abbildung 2.7.

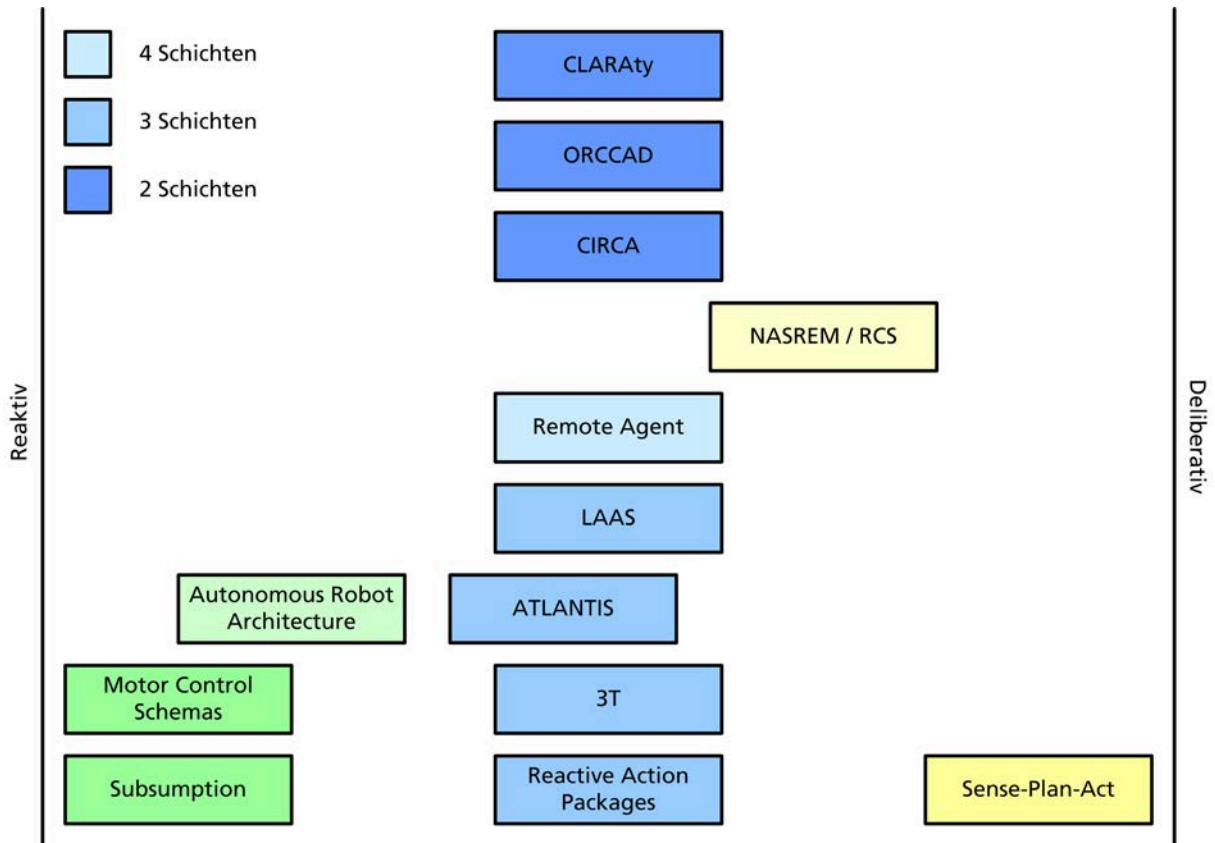


Abb. 2.7.: Übersicht über verschiedene Vertreter der unterschiedlichen Arten von Steuerungsarchitekturen. Links: Reaktive und verhaltensbasierte Steuerungsarchitekturen. Mitte: Mehrschichtige Steuerungsarchitekturen. Rechts: Deliberative Steuerungsarchitekturen.

**Deliberative Architekturen** Die Entwicklung von Roboterarchitekturen begann in den späten 1960er Jahren mit dem Roboter Shakey [75]. Die Architektur von Shakey bestand aus drei funktionalen Elementen: Wahrnehmung, Planung und Ausführung [76]. Dieser Ansatz wurde als *Sense-Plan-Act* (SPA) Paradigma bekannt. Die wesentlichen Merkmale der Architektur bestanden darin, dass die Informationen der Wahrnehmung in ein Weltmodell flossen, welches anschließend von einem Planer zur Planerstellung verwendet wurde. Dieser Plan wurde schließlich ohne erneute Verwendung der Sensoren ausgeführt.

**Reaktive und verhaltensbasierte Architekturen** In den frühen 1980er Jahren wurden die Probleme des SPA-Paradigmas offenkundig. So benötigt die Planung für Domänen der realen Welt in der Regel eine signifikante Zeit, während der der Roboter quasi blockiert ist. Darüber hinaus ist die Ausführung von Plänen ohne Einbeziehung der Wahrnehmung in einer dynamischen Welt gefährlich.

Es entstanden neue Ansätze für Steuerungsarchitekturen, wie etwa die reaktive Planung. Hier wurden die Pläne schnell erzeugt und basierten direkter auf den Sensorinformationen anstatt auf internen Modellen. Die einflussreichste Arbeit in diesem Bereich ist der *Subsumption*-

Ansatz von Brooks [21]. Eine Subsumption-Architektur besteht aus Ebenen von interagierenden endlichen Zustandsautomaten, welche Sensoren und Aktuatoren direkt miteinander verbinden. Die Zustandsautomaten wurden *Verhalten* genannt – es entstand der Begriff der verhaltensbasierten Robotik [6]. Da mehrere Verhalten zur gleichen Zeit aktiv sein können, verfügt der Subsumption-Ansatz über einen Schlichtungsmechanismus, welcher es Verhalten auf höheren Ebenen erlaubt, die Signale von Verhalten auf niedrigeren Ebenen zu überschreiben. Durch mehrere, miteinander interagierende Ebenen von Verhalten ist es so möglich, komplexere Roboterverhalten zu erzeugen.

Ein bekanntes Beispiel für verhaltensbasierte Architekturen sind die so genannten *motor-control schemas* von Arkin [5]. In diesem biologisch motivierten Ansatz werden Motor- und Wahrnehmungsschemata dynamisch miteinander verknüpft. Die Motorschemata generieren dabei Antwortvektoren basierend auf den Ausgaben der Wahrnehmungsschemata. Diese werden anschließend mit einem Potentialfeldern ähnelnden Ansatz verknüpft. Zur Erreichung komplexerer Ziele fügte die *Autonomous Robot Architecture* (AuRA) [7] dem reaktiven Schema einen Navigationsplaner und einen Plansequenzer basierend auf *Finite-State Acceptors* (FSAs) hinzu.

Roboter mit dem SPA-Paradigma entsprechenden Architekturen waren meist langsam und schwerfällig, Roboter mit Subsumption-Architekturen hingegen schnell und reaktiv. Jedoch stießen auch verhaltensbasierte Roboter schnell an die Grenzen ihrer Fähigkeiten. So erwies es sich als sehr schwierig, Verhalten zur Erreichung langfristiger Ziele zu erzeugen und das Gesamtrroboterverhalten zu optimieren. Es stellte sich heraus, dass Roboter sowohl über die Planungsfähigkeiten der frühen Architekturen als auch über die Reaktivität der verhaltensbasierten Architekturen verfügen sollten. Hieraus ergaben sich hybride Ansätze in Form von mehrschichtigen Steuerungsarchitekturen.

**Mehrschichtige Architekturen** Einer der ersten Schritte in Richtung der Integration von Reaktivität und Deliberation war das so genannte *Reactive Action Packages* (RAPs) System von Robert J. Firby. In seiner Doktorarbeit [32] waren die ersten Grundzüge einer integrierten Drei-Schichten-Architektur zu erkennen. Die mittlere Ebene war Gegenstand der Doktorarbeit und wurde vom RAPs-System gebildet.

Zeitgleich und unabhängig entwickelte Bonasso eine Architektur [16], deren unterste Schicht aus Roboterverhalten bestand, welche mit Hilfe der Sprache *Rex* programmiert wurden. Diese so genannten Rex-Maschinen gewährleisteten eine konsistente Semantik zwischen den internen Zuständen des Agenten und denen der realen Welt. Die mittlere Schicht wurde von einem bedingten Sequenzer gebildet, welcher in der Sprache *Gapps* umgesetzt war und kontinuierlich die Rex-Fähigkeiten aktivierte und deaktivierte, bis die Aufgabe des Roboters vollständig erfüllt war. Eine Synthetisierung des Sequenzers durch traditionellere Planungstechniken bot sich an. Diese Arbeiten gipfelten schließlich in der so genannten 3T-Architektur [15], welche nach ihren drei Schichten (engl. *tier*) von interagierenden Steuerungsprozessen benannt ist: Planung,

Sequenzierung und Echtzeitsteuerung.

Nachfolgend wurden viele 3T ähnelnde Architekturen entwickelt. Ein Beispiel hierfür ist die ATLANTIS-Architektur [37], welche einen großen Teil der Kontrolle der Sequenzierungsebene überlässt. In ATLANTIS muss die deliberative Ebene explizit von der Sequenzierungsebene aufgerufen werden.

Ein weiteres Beispiel ist die *Intelligent Control Architecture* von Saridis [88]. Die Architektur beginnt mit den in einem Roboter verfügbaren Servosystemen und augmentiert diese, um die Ausführungsalgorithmen der nächsten Schicht zu integrieren. Die nächste Ebene besteht aus einer Menge von Koordinationsroutinen für jedes Subsystem. Diese werden mit Hilfe von *Petri Net Transducers* (PNTs) umgesetzt, welche eine Art Schedulingmechanismus bilden und von einem mit der organisatorischen Ebene verbundenen Dispatcher aktiviert werden. Die organisatorische Ebene besteht aus einem Planer, der als Neuronales Netzwerk in Form einer *Boltzmann-Maschine* implementiert ist.

Die *LAAS Architecture for Autonomous Systems* (LAAS) [2] ist eine Drei-Schichten-Architektur, welche Softwarewerkzeuge zur Unterstützung der Entwicklung und Programmierung für jede Schicht beinhaltet. Die unterste, funktionale Ebene besteht aus einem Netzwerk von Modulen, welche dynamisch parametrisierte Steuerungs- und Wahrnehmungsalgorithmen enthalten. Die Module werden mit Hilfe der *Generator Of Modules* (GenoM) Sprache geschrieben, die standardisierte Templates erzeugt, welche die Interaktion der Module untereinander ermöglichen. Im Gegensatz zu den meisten Drei-Schichten-Architekturen ist die Ausführungsebene relativ einfach gehalten. Sie ist rein reaktiv und führt keine Aufgabenzerlegung durch, sondern dient vielmehr als Brücke, indem sie Aufgabensequenzen von der obersten Ebene empfängt, Aufgaben auswählt, parametrisiert und an die funktionale Ebene weitergibt. Die Ausführung ist in der *Kheops*-Sprache geschrieben, welche automatisch Entscheidungsnetzwerke erzeugt, die formal verifiziert werden können. Die Entscheidungsebene besteht aus einem Planer und einem Supervisor. Als Planer kommt der *Indexed Time Table (IxTeT) Temporal Planner* [39] zum Einsatz. Der Supervisor ist mit Hilfe des *Procedural Reasoning Systems* (PRS) [53] umgesetzt und ähnelt der Ausführungsebene anderer Drei-Schichten-Architekturen. Er zerlegt Aufgaben, wählt alternative Methoden zur Erfüllung von Aufgaben aus und überwacht die Ausführung. Durch die Kombination von Planer und Supervisor in einer Ebene wird eine engere Verzahnung dieser beiden Komponenten erreicht, wodurch eine höhere Flexibilität erzielt wird, wann und wie neu geplant wird.

Die so genannte *Remote Agent* Architektur [70] ist eine Architektur für die autonome Steuerung von Raumfahrzeugen. Sie besteht aus vier Ebenen: einer verhaltensbasierten Steuerungsebene, einer Ausführungsebene, einem Planer/Scheduler und einer *Mode Identification and Recovery* (MIR) Ebene, welche Fehlererkennung und -behebung kombiniert. Die Steuerungsebene wird durch ein traditionelles Echtzeitsteuerungssystem für Raumfahrzeuge gebildet. Die Ausfüh-

rungsebene bildet den Kern der Architektur. Sie ist für die Verfeinerung, Auswahl und Überwachung von Aufgaben, die Behebung von Fehlerzuständen sowie die Ressourcenverwaltung zuständig. Der Planer/Scheduler ist als Stapelverarbeitungsprozess realisiert, welcher als Eingabe Ziele, einen initialen (projizierten) Zustand sowie die aktuell geschedulten Aktivitäten erhält und einen Plan erzeugt, der flexible Bereiche von Start- und Endzeiten für Aufgaben beinhaltet. Der Plan enthält außerdem eine spezielle Aufgabe, die den Planer erneut aufruft, um das nächste Plansegment zu generieren. Ein wesentlicher Bestandteil der *Remote Agent* Architektur ist das so genannte Konfigurationsmanagement. Es ist dafür zuständig, Hardware zur Unterstützung von Aufgaben zu konfigurieren und zu überwachen, dass die Hardware in bekannten, stabilen Zuständen verbleibt. Das Konfigurationsmanagement ist zwischen der Ausführungs- und der MIR-Ebene aufgeteilt. Auf der Ausführungsebene kommen hierfür reaktive Verfahren zum Einsatz. Auf der MIR-Ebene hingegen werden deklarative Modelle des Raumfahrzeugs sowie deliberative Algorithmen verwendet, um zu bestimmen, wie die Hardware zur Behandlung von erkannten Fehlern rekonfiguriert werden muss.

In der Literatur finden sich weitere Vertreter mehrschichtiger Architekturen. Ein bekanntes Beispiel ist das vom National Bureau of Standards (NBS) für die NASA entwickelte *NASA/NBS Standard Reference Model* (NASREM) [4], welches später in *Real-time Control System* (RCS) [3] umbenannt wurde. Hierbei handelt es sich um ein frühes Referenzsystem für den Bereich der Telerobotik, welches vielschichtig aufgebaut ist. Jede Schicht verfügt hierbei über den gleichen generischen Aufbau, arbeitet jedoch mit zunehmend langsameren Frequenzen, je weiter sie sich von der Servoebene hin zur deliberativen Ebene bewegt. Im Gegensatz zu anderen Dreischichten-Architekturen verfügt NASREM über ein globales Weltmodell. Darüber hinaus enthält es alle Daten- und Steuerpfade, welche in anderen Architekturen wie 3T vorkommen. Bei NASREM handelt es sich jedoch um ein Referenzmodell und nicht um eine Implementierung. Die nachfolgenden Implementierungen von NASREM folgten im Wesentlichen dem traditionellen SPA-Paradigma und wurden hauptsächlich in der Telerobotik und nicht im Bereich der autonomen Roboter eingesetzt.

Neben den dreischichtigen Architekturen wurden auch verschiedene zweischichtige Ansätze untersucht. Die *Cooperative Intelligent Real-Time Control Architecture* (CIRCA) [71] ist eine Zwei-Schichten-Architektur, die sich mit der Garantierung von verlässlichem Verhalten beschäftigt. CIRCA besteht aus einem *Real-Time System* (RTS) und einem *Artificial Intelligence System* (AIS). Das RTS führt einen zyklischen Zeitplan von so genannten *Test Action Pairs* (TAPs) aus, die ein garantiertes Worst-Case-Verhalten in Bezug auf die Wahrnehmung der Umgebung und ein konditionales Antwortverhalten besitzen. Das AIS hat die Aufgabe, einen Zeitplan zu erstellen, der das Auftreten katastrophaler Fehler verhindert, während das System in harter Echtzeit läuft. Hierfür plant das AIS mit Hilfe eines Zustandsübergangsgraphen, welcher Zustandsübergänge für Aktionen, exogene Ereignisse und das Verstreichen von Zeit enthält.

Das AIS überprüft für jeden Plan (Menge von TAPs), ob dieser tatsächlich geschedult werden kann. Ist dies nicht der Fall, wird der Plan durch das Löschen von Tasks (basierend auf Zielprioritäten) oder durch die Änderung der Parametrisierung von Verhalten abgeändert. Dies wird solange durchgeführt, bis ein Plan erfolgreich geschedult werden kann. In diesem Fall wird der neue Plan in einer atomaren Operation an das RTS übergeben.

Ähnlich wie CIRCA ist ORCCAD [17] eine Zwei-Schichten-Architektur, die sich mit garantierter Verlässlichkeit beschäftigt. Diese Garantie wird durch formale Verifikationstechniken erzielt. Roboteraufgaben (Verhalten auf niedriger Ebene) und Robotermethoden (Aktionen auf höherer Ebene) werden in höheren Programmiersprachen definiert und anschließend zur logischen Verifikation in die *Esterel*-Programmiersprache und zur zeitlichen Verifikation in die *Timed-Argus*-Sprache übersetzt. Die Verifikation erfolgt im Hinblick auf Antwort- und Sicherheitseigenschaften sowie die Vermeidung von Ressourcenengpässen.

### **Coupled Layered Architecture for Robotic Autonomy (CLARAty)**

Die gemeinsam vom Jet Propulsion Laboratory, dem NASA Ames Research Center und der Carnegie Mellon University entwickelte *Coupled Layered Architecture for Robotic Autonomy* (CLARAty) [74] ist eine domänenspezifische Robotik-Softwarearchitektur, die gemäß den folgenden vier Hauptzielen entworfen wurde:

1. Reduzierung der Notwendigkeit zur Entwicklung einer angepassten Robotik-Infrastruktur für jedes neue Forschungsvorhaben.
2. Vereinfachung der Integration neuer Technologien in bestehende Systeme.
3. Enge Kopplung von deklarativen und prozeduralen Algorithmen.
4. Betrieb einer Vielzahl von heterogenen Rovern mit unterschiedlichen physikalischen Eigenschaften und Hardwarearchitekturen.

CLARAty besteht aus zwei unterschiedlichen Schichten: einer funktionalen Schicht und einer Entscheidungsschicht. Die funktionale Schicht definiert verschiedene Abstraktionen des Systems und adaptiert die abstrakten Komponenten an reale oder simulierte Geräte. Sie bietet ein Rahmenwerk zur Entwicklung von Algorithmen für Autonomie auf niedriger und mittlerer Ebene. Die Entscheidungsschicht stellt die Autonomie des Systems auf höchster Ebene zur Verfügung, die Schlussfolgerungen bezüglich der globalen Ressourcen und Randbedingungen der Mission anstellt. Sie greift auf Informationen der funktionalen Schicht auf verschiedenen Granularitätsebenen zu. Die Entscheidungsschicht verwendet ein deklaratives Modell, die funktionale Schicht ein prozedurales Modell.

Ein Unterschied zwischen CLARAty und herkömmlichen aus funktionaler Schicht, Ausführungsschicht und Planungsschicht bestehenden Drei-Schichten-Architekturen liegt in der expliziten Unterscheidung zwischen Granularitätsschichten und Intelligenzschichten. In herkömmlichen Architekturen sind sowohl Granularität als auch Intelligenz entlang einer gemeinsamen Achse angeordnet. Mit steigender Abstraktion des Systems nimmt die Intelligenz zu. Dies trifft für CLARAty nicht zu, da hier Granularität und Intelligenz entlang zweier verschiedener Achsen angeordnet sind. Dies erlaubt intelligentes Verhalten auf niedriger Systemebene bei gleichzeitiger Beibehaltung der verschiedenen Abstraktionsebenen.

**Funktionale Schicht** Die funktionale Schicht enthält eine Vielzahl generischer Pakete: digitale und analoge Ein- und Ausgabe, Bewegungssteuerung und -koordination, Lokomotion, Manipulation, Bildverarbeitung, Navigation, Kartierung, Geländebewertung, Pfadplanung, wissenschaftliche Analyse, Schätzverfahren, Simulation und Systemverhalten. Sie stellt die Systemfähigkeiten mit niedrigem und mittlerem Autonomiegrad zur Verfügung. Steuerungsalgorithmen wie visuelle Navigation, sensorbasierte Manipulation oder visuelle Zielverfolgung, die eine vordefinierte Sequenz von Operationen verwenden, werden oft in der funktionalen Schicht implementiert. In manchen Fällen ist es jedoch möglich, diese Operationssequenzen zu generieren, indem sie als Aktivitäten modelliert werden und die Entscheidungsschicht Instanzierungen dieser Aktivitäten basierend auf geeigneten Missionszielen und Randbedingungen einplant.

Die funktionale Schicht hat vier Haupteigenschaften. Erstens stellt sie eine Zerlegung in unterschiedliche Systemebenen mit verschiedenen Abstraktionsgraden bereit. Zweitens separiert sie die algorithmischen Fähigkeiten von den Systemfähigkeiten. Drittens trennt sie die Verhaltensdefinition und die Interaktion des Systems von der Implementierung. Und viertens stellt sie flexible Laufzeitmodelle zur Verfügung.

**Entscheidungsschicht** Die Entscheidungsschicht übernimmt die Planung, zeitliche Ablaufkoordination und Ausführung von Aktivitätsplänen. Sie enthält allgemeine Planer, Ausführungseinheiten, Scheduler, Aktivitätsdatenbanken sowie rover- und planerspezifische Heuristiken. Ziel der generischen Entscheidungsschicht ist eine einheitliche Repräsentation von Aktivitäten und Schnittstellen. Die gegenwärtig am JPL verwendete Instantiierung der Entscheidungsschicht ist das *Closed-Loop Execution and Recovery* (CLEaR) System [30]. Es besteht aus dem *Continuous Activity Scheduling, Planning, Execution and Replanning* (CASPER) reparaturbasierten Planer [59] und der *Task Description Language* (TDL) Ausführungssprache [96]. CLEaR bietet einen eng gekoppelten Ansatz, um ziel- und ereignisgesteuertes Verhalten zu erzeugen, dessen Kern in der Fähigkeit zur schnellen und kontinuierlichen Neuplanung basierend auf häufigen Zustands- und Ressourcenaktualisierungen von der Ausführungsüberwachung besteht. Dies ermöglicht es dem Planer, schnell auf viele Ausnahmesituationen zu reagieren. In CLEaR verfügen sowohl die Planungs- als auch die Ausführungskomponenten über Fähigkei-



ten zur Behandlung von Ressourcenkonflikten und Ausnahmesituationen, wobei anhand von Heuristiken darüber entschieden wird, welche Komponenten in einer bestimmten Situation involviert werden.

Die Entscheidungsschicht kommuniziert mit der funktionalen Schicht unter Verwendung eines Client-Server Modells. Die Entscheidungsschicht fragt die funktionale Schicht bezüglich der Verfügbarkeit von Ressourcen an, um den Ressourcenverbrauch einer gegebenen Operation zu bestimmen. Darüber hinaus sendet sie Befehle auf verschiedenen Granularitätsebenen an die funktionale Schicht. Die Entscheidungsschicht kann gekapselte Fähigkeiten der funktionalen Schicht mit Kommandos auf relativ hoher Ebene verwenden oder auf Fähigkeiten auf niedrigerer Ebene zugreifen und diese in einer nicht von der funktionalen Schicht zur Verfügung gestellten Art und Weise kombinieren. Der Status von Ressourcen, Zustandsbedingungen und Aktivitätsausführungen kann asynchron oder synchron mit von der Entscheidungsschicht festgelegten Raten von der funktionalen Schicht an die Entscheidungsschicht berichtet werden.

### 2.1.3. Sensordatenauswertung

Im Folgenden wird näher auf die im Bereich der autonomen Inspektionsroboter verwendeten Verfahren zur Sensordatenauswertung eingegangen. Dabei wird insbesondere das am Jet Propulsion Laboratory entwickelte OASIS-System vorgestellt, das an Bord der Mars Rover Verwendung findet.

#### **Onboard Autonomous Science Investigation System (OASIS)**

Das *Onboard Autonomous Science Investigation System* (OASIS) [25] wurde am Jet Propulsion Laboratory entwickelt, um wissenschaftliche Daten, die von *in situ* Raumfahrzeugen wie planetaren Landefähren und Rovern gewonnen werden, zu evaluieren und automatisch mit intelligenten Aktionen darauf zu reagieren. Ziel ist es, die dem Rover zur Verfügung stehenden Ressourcen effizient zu nutzen und die Qualität der wissenschaftlichen Daten, die zurück zur Erde gesendet werden, zu maximieren. OASIS analysiert geologische Daten, die von Rovern gesammelt werden, direkt an Bord. Diese Analyse dient der Identifikation von interessierenden Geländemerkmale sowie von günstigen Gelegenheiten, weitere interessierende wissenschaftliche Daten zu sammeln. Eine Komponente zur Planung und Ablaufkoordination versetzt den Rover in die Lage, die identifizierten Gelegenheiten zu nutzen, indem die Befehlsfolge so aktualisiert wird, dass sie zusätzliche Messungen umfasst. OASIS arbeitet in einer geschlossenen Regelschleife mit der Steuerungssoftware des Rovers zusammen und verfügt über die Möglichkeit, autonom die folgenden Schritte durchzuführen: Analyse von Grauwertbildern zur Detektion von Felsen, Extraktion der Felseigenschaften, Identifikation von interessierenden Felsen,

Anweisung des Rovers zur Aufnahme weiterer Bilder der identifizierten Ziele und anschließende Fortsetzung der ursprünglichen Mission.

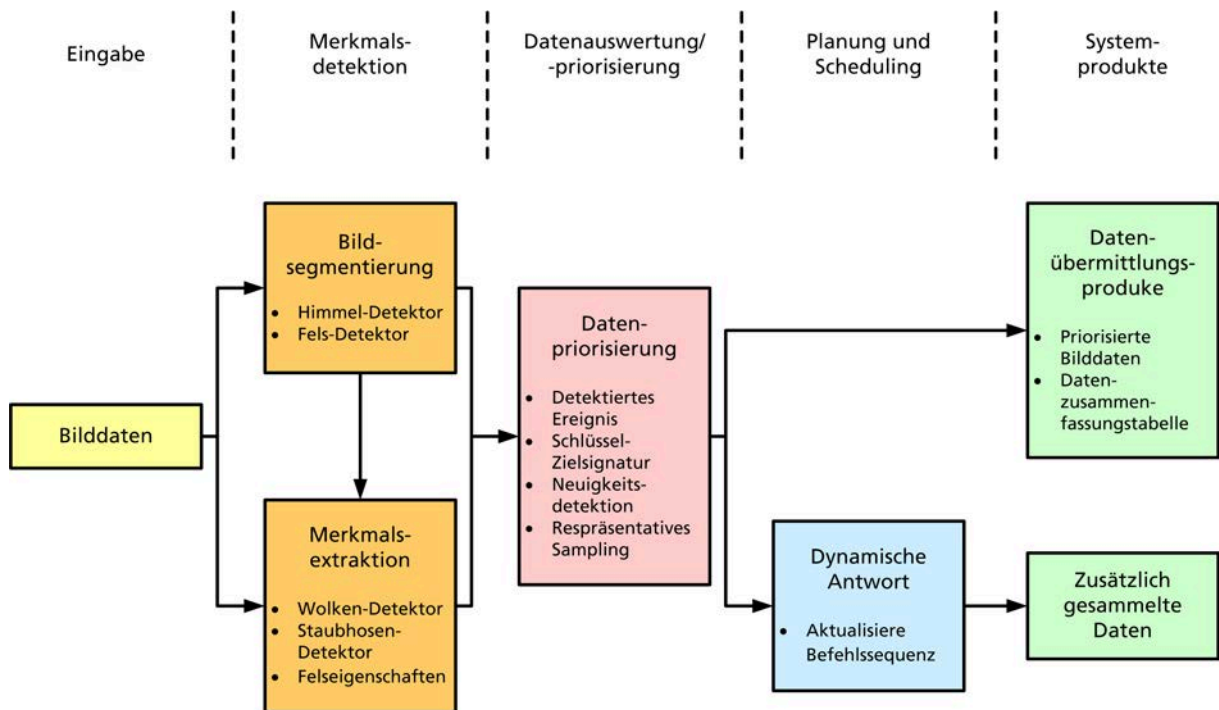


Abb. 2.8.: Überblick über das OASIS-System und den Datenfluss zwischen den einzelnen Komponenten [25].

Das OASIS-System besteht aus den Komponenten *Merkmalsdetektion*, *Datenauswertung und -priorisierung* sowie *Planung und Ablaufkoordination* (siehe Abbildung 2.8). Die *Merkmalsdetektion* ermöglicht die Extraktion von interessierenden Merkmalen aus den gesammelten Bildern des umgebenden Gebiets. Sie lokalisiert Felsen in den Bildern und extrahiert Felseigenschaften wie Form, Textur und Rückstrahlungsvermögen (Albedo). Außerdem enthält sie Module zur Analyse der Atmosphäre. Dies umfasst die Erkennung von Wolken und von Staubhosen. Die *Datenauswertung und -priorisierung* analysiert die extrahierten Merkmale, um den wissenschaftlichen Wert der Daten zu beurteilen und neue wissenschaftliche Ziele zu generieren. Die *Planung und Ablaufkoordination* erlaubt die dynamische Anpassung der gegenwärtigen Befehlsfolge (des Plans), um neue wissenschaftliche Ziele zu berücksichtigen. Hierzu wird ein kontinuierlicher Planungsansatz verwendet, der den Plan iterativ anpasst, sobald sich neue Ziele ergeben. Ziele zur Nutzung von unvorhergesehenen Gelegenheiten können in den Plan eingefügt werden, solange die Ressourcenbedingungen und die operationellen Bedingungen eingehalten und alle höher priorisierten Ziele erreicht werden können.

Die Fähigkeiten zur Planung und Ausführung werden in OASIS durch das *Closed-Loop Execution and Recovery (CLEaR)* System [30] bereitgestellt, welches den *Continuous Activity Scheduling, Planning, Execution and Replanning (CASPER)* reparaturbasierten Planer [59] und die *Task Description Language (TDL)* Ausführungssprache [96] integriert (siehe oben). OASIS

erweitert CLEaR um die Auswertung von wissenschaftlichen Daten, so dass die Architektur auch von sich ergebenden günstigen wissenschaftlichen Gelegenheiten getrieben werden kann.

Im Gegensatz zu vielen Drei-Schichten-Architekturen (siehe Abschnitt 2.1.2), die üblicherweise im Stapelverarbeitungsbetrieb planen und für die Planung in der Größenordnung von Minuten bis Stunden benötigen, verwendet das integrierte OASIS-System einen kontinuierlichen Planungsansatz, mit dem Pläne innerhalb von Sekunden aktualisiert und repariert werden. Dies ermöglicht die Verwendung von Planungstechniken mit einer feingranulareren Zeitskala zur Nachverfolgung des Fortschritts der Planausführung, wodurch mögliche Probleme in zukünftigen Teilen der Pläne schnell erkannt und entsprechend behandelt werden. Da erwartet wird, dass sich kleinere Teile des Plans häufig ändern, wird eine leichtgewichtige Planbearbeitung verwendet, die Aktivitäten erst wenige Sekunden vor dem geplanten Startzeitpunkt an die Ausführung übergibt. Hierdurch unterscheidet sich dieser Ansatz vom gewöhnlichen Batch-Ansatz, bei dem der vollständige Plan an die Ausführungskomponente zur Ausführung übergeben wird.

Wissenschaftliche Alarme können Reaktionen des Planungs- und Ausführungssystems auf verschiedenen Ebenen zur Folge haben. Die grundlegendste Reaktion besteht in einer Anpassung des Plans, so dass der Rover an seiner gegenwärtigen Position anhält und die mit einem Flag markierten Daten zur weiteren Analyse zurück zur Erde sendet. Die nächste Reaktionsebene besteht im Sammeln weiterer Daten von der aktuellen Position aus, bevor die Daten zurück zur Erde gesendet werden. Ein weiterer Schritt besteht in einer Pfadänderung des Rovers, so dass sich dieser dem interessierenden Objekt nähert, bevor weitere Messungen durchgeführt werden. Ein wissenschaftlicher Alarm wird als optionales Planungsziel mit einer bestimmten Priorität repräsentiert. Die Priorität wird dabei vom Datenauswertungssystem an Bord bestimmt. Da es sich um optionale Ziele handelt, ist ihre Erreichung nicht zwingend erforderlich. Sie kann jedoch die Optimierungspunktzahl des Plans erhöhen.

Zur Evaluierung des OASIS-Systems mit Rover-Hardware, ist die Komponente zur Planung, Ablaufkoordination und Ausführung in die *Coupled Layered Architecture for Robotic Autonomy* (CLARAty) [74] integriert (siehe oben).

## 2.2. Wissensrepräsentation

*Wissensrepräsentation und Schlussfolgerung bilden ein Gebiet der Künstlichen Intelligenz (KI), welches sich damit beschäftigt, wie Wissen symbolisch repräsentiert werden und in einer automatisierten Art und Weise von Schlussfolgerungsprogrammen gehandhabt werden kann. Etwas informeller ausgedrückt handelt es sich um den Teil der KI, der sich mit dem Denken an sich und damit beschäftigt, wie das Denken zu intelligentem Verhalten beiträgt [20].*

Im Folgenden werden die für diese Arbeit relevanten Verfahren zur Wissensrepräsentation vorgestellt. Im Fokus stehen dabei insbesondere Ontologien und semantische Technologien aus

dem Bereich des Semantic Web. Nachdem die Grundlagen dieser beiden Themenfelder erläutert wurden, wird anschließend auf die Verwendung von semantischem Wissen in der Robotik eingegangen und exemplarisch ein semantisches Rahmenwerk zur Verbesserung des Situationsbewusstseins vorgestellt.

### 2.2.1. Ontologien

In diesem Abschnitt wird die Repräsentation von Wissen mit Hilfe von so genannten *Ontologien* betrachtet. Dazu werden zunächst einige entsprechende Definitionen vorgestellt. Daran anknüpfend werden verschiedene Kriterien, die beim Entwurf von Ontologien zu beachten sind, erläutert. Abschließend wird kurz auf die verschiedenen Rollen und Vorteile von Ontologien im Hinblick auf informationsverarbeitende Systeme eingegangen.

#### Definitionen

Nach Gruber [43] ist eine *Ontologie* eine explizite Spezifikation einer Konzeptualisierung. Eine *Konzeptualisierung* wiederum ist eine abstrakte, vereinfachte Sicht auf die Welt, die für einen bestimmten Zweck repräsentiert wird. Sie ist definiert durch die Objekte, Konzepte und anderen Entitäten, von denen angenommen wird, dass sie in einem bestimmten Bereich, der von Interesse ist, existieren, und durch die Beziehungen, die zwischen diesen gelten (siehe [38]). Jede Wissensbasis, jedes wissensbasierte System und jeder Agent auf Wissensebene ist implizit oder explizit an eine solche Konzeptualisierung gebunden.

Ursprünglich stammt der Begriff der Ontologie aus der Philosophie, in der er eine systematische Beschreibung der Existenz bezeichnet. Im Kontext der Künstlichen Intelligenz kann eine Ontologie durch eine Menge von begrifflichen Termen definiert werden. In einer solchen Ontologie verbinden Definitionen Namen von Entitäten des Gegenstandsbereichs mit vom Menschen lesbaren Texten, die beschreiben, was diese Namen bedeuten, und formalen Axiomen, welche die Interpretation und den wohldefinierten Gebrauch dieser Terme einschränken. Formal betrachtet ist eine Ontologie die Aufstellung einer logischen Theorie.

Pragmatisch definiert eine gemeinsame Ontologie das Vokabular, mit dem Anfragen und Zusicherungen zwischen Agenten ausgetauscht werden. Ein Agent *bindet* sich an eine Ontologie, wenn seine beobachtbaren Handlungen mit den Definitionen der Ontologie konsistent sind. *Ontologische Bindungen* sind somit Übereinkünfte, das gemeinsame Vokabular in einer kohärenten und konsistenten Art und Weise zu verwenden. Die Bindung an eine gemeinsame Ontologie garantiert die Konsistenz, jedoch nicht die Vollständigkeit in Bezug auf Anfragen und Zusicherungen, welche das in der Ontologie definierte Vokabular nutzen.

Eine Ontologie dient einem anderen Zweck als eine Wissensbasis. Eine gemeinsame Ontologie definiert das Vokabular, das benötigt wird, um über eine Domäne zu sprechen, wohingegen

eine Wissensbasis das Wissen beinhalten kann, das benötigt wird, um Probleme zu lösen oder beliebige Anfragen in Bezug auf eine Domäne zu beantworten.

Die Definition von Gruber wird in [44] von Guarino aufgegriffen und weiter präzisiert. Nach Guarino kann eine Ontologie im philosophischen Sinne als eine spezielles System von Kategorien betrachtet werden, das eine bestimmte Sicht auf die Welt begründet. Eine Ontologie im Sinne der Künstlichen Intelligenz ist hingegen ein technisches Artefakt, welches aus einem spezifischen Vokabular zur Beschreibung einer bestimmten Realität und einer Menge von expliziten Annahmen bezüglich der intendierten Bedeutung der Worte des Vokabulars besteht. Die Menge der expliziten Annahmen hat üblicherweise die Form einer logischen Theorie erster Ordnung, in der Vokabularworte als Namen von unären oder binären Prädikaten auftreten, welche Konzepte bzw. Relationen genannt werden. Zur Unterscheidung der beiden Lesarten wird der Begriff Konzeptualisierung eingeführt, der die philosophische Lesart benennt. Zwei Ontologien können sich also im verwendeten Vokabular unterscheiden, während sie eine gemeinsame Konzeptualisierung teilen. Dies führt Guarino zu der folgenden Definition einer Ontologie:

*Eine Ontologie ist eine logische Theorie, welche die intendierte Bedeutung eines formalen Vokabulars begründet, das heißt seine ontologische Bindung an eine bestimmte Konzeptualisierung der Welt. Die intendierten Modelle einer logischen Sprache, welche ein solches Vokabular verwendet, werden durch seine ontologische Bindung eingeschränkt. Eine Ontologie reflektiert diese Bindung (und die zugrunde liegende Konzeptualisierung) indirekt, indem sie die intendierten Modelle approximiert [44].*

Die Beziehungen zwischen einem Vokabular, einer Konzeptualisierung, einer ontologischen Bindung und einer Ontologie sind in Abbildung 2.9 dargestellt. Eine Ontologie ist sprachabhängig während eine Konzeptualisierung sprachunabhängig ist. Bei der Verwendung in der Künstlichen Intelligenz fallen diese Begriffe de facto zusammen. Eine klare Trennung zwischen den beiden Aspekten ist jedoch notwendig, um Themen wie die gemeinsame Nutzung, die Fusion und Übersetzung von Ontologien adressieren zu können.

Auch wenn zwei Systeme dasselbe Vokabular verwenden, gibt es keine Garantie, dass sie sich bezüglich einer bestimmten Information verständigen können. Eine notwendige Bedingung, um diese Verständigung ermöglichen zu können, besteht darin, dass sich die intendierten Modelle der ursprünglichen Konzeptualisierung überlappen. Werden diese zwei Mengen der intendierten Modelle von zwei unterschiedlichen Ontologien approximiert, so kann es vorkommen, dass sich die Ontologien überlappen, die intendierten Modelle hingegen jedoch nicht. Aus diesem Grund erscheint es zweckmäßiger, sich auf eine gemeinsame Top-Level-Ontologie zu verständigen, als sich auf Vereinbarungen basierend auf der Schnittmenge verschiedener Ontologien zu stützen.

Guarino unterscheidet verschiedene Arten von Ontologien basierend auf der Ebene ihrer Allgemeingültigkeit:

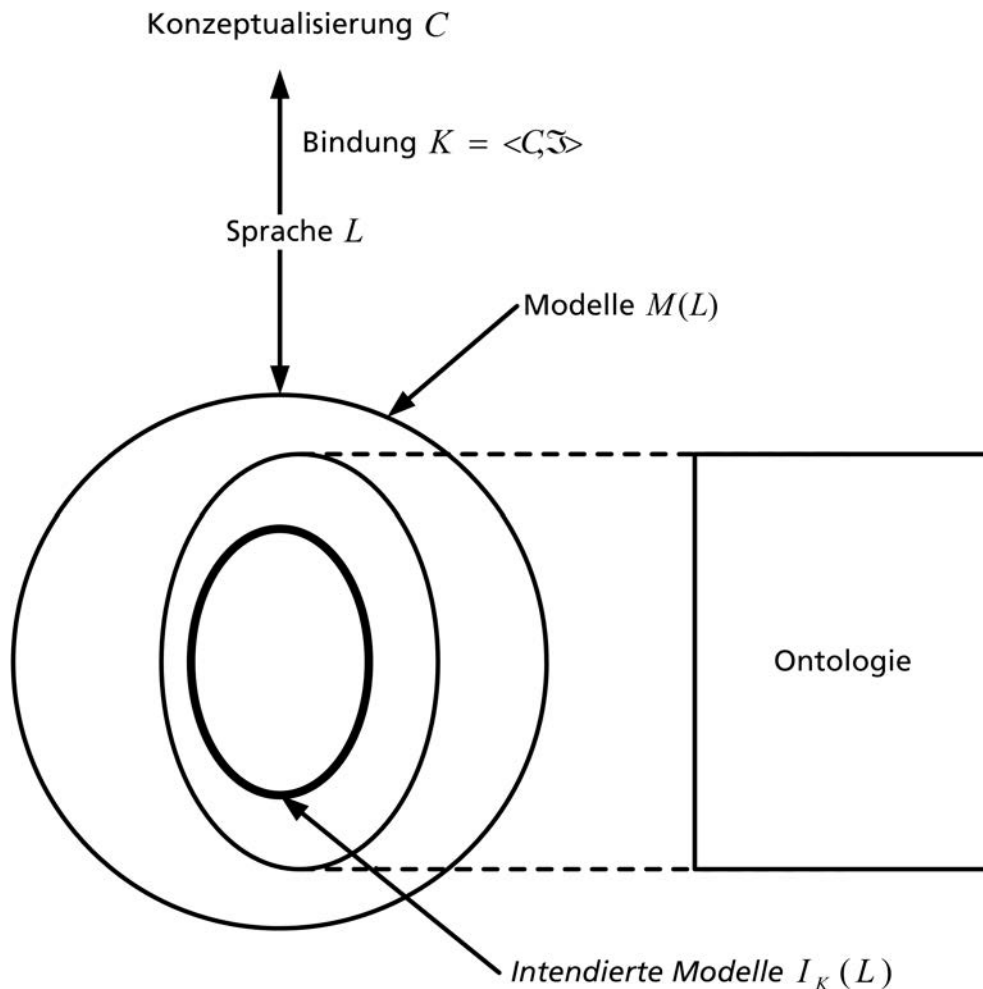


Abb. 2.9.: Die intendierten Modelle einer logischen Sprache reflektieren ihre Bindung an eine Konzeptualisierung. Eine Ontologie reflektiert indirekt diese Bindung (und die zugrunde liegende Konzeptualisierung) durch die Approximation der Menge der intendierten Modelle [44].

**Top-Level-Ontologie** Top-Level-Ontologien beschreiben sehr allgemeine Konzepte, welche unabhängig von einem bestimmten Problem oder einer bestimmten Domäne sind.

**Domänen- und Aufgabenontologien** Domänen- und Aufgabenontologien beschreiben das zu einer generischen Domäne bzw. einer generischen Aufgabe oder Aktivität gehörende Vokabular, indem sie in der Top-Level-Ontologie eingeführte Terme spezialisieren.

**Anwendungsontologien** Anwendungsontologien beschreiben Konzepte, die sowohl von einer speziellen Domäne als auch von einer speziellen Aufgabe abhängig sind und oft Spezialisierungen beider zugehöriger Ontologien sind. Diese Konzepte entsprechen oft Rollen, die von Domänenentitäten während der Ausführung von bestimmten Aktivitäten eingenommen werden.

## Entwurfskriterien

Wenn etwas in einer Ontologie repräsentiert werden soll, sind Entwurfsentscheidungen zu treffen. Um den Entwurf zu leiten und zu evaluieren, werden objektive Kriterien benötigt, welche auf dem Zweck des resultierenden Artefakts basieren. In [43] werden daher von Gruber die folgenden Entwurfskriterien vorgeschlagen:

**Klarheit** Eine Ontologie sollte effektiv die beabsichtigte Bedeutung von definierten Termen kommunizieren. Definitionen sollten daher objektiv und unabhängig von sozialen oder rechenbezogenen Kontexten sein. Wenn eine Definition in logischen Axiomen ausgedrückt werden kann, sollte dies geschehen. Wo möglich sollte eine vollständige Definition (ein Prädikat definiert durch notwendige und hinreichende Bedingungen) einer partiellen Definition (definiert nur durch notwendige oder hinreichende Bedingungen) vorgezogen werden. Alle Definitionen sollten natürlichsprachlich dokumentiert werden.

**Kohärenz** Eine Ontologie sollte kohärent sein. Als Minimalanforderung sollten die definierten Axiome logisch konsistent sein. Kohärenz sollte darüber hinaus auch für die informell definierten Konzepte gelten, welche in der natürlichsprachlichen Dokumentation oder in Beispielen beschrieben werden. Wenn ein Satz aus den Axiomen abgeleitet werden kann, der einer informellen Definition oder einem Beispiel widerspricht, so ist die Ontologie inkohärent.

**Erweiterbarkeit** Der Entwurf einer Ontologie sollte die Verwendung des gemeinsamen Vokabulars vorhersehen. Eine Ontologie sollte eine konzeptuelle Grundlage für einen Bereich von vorhergesehenen Aufgaben bereitstellen und die Repräsentation sollte so beschaffen sein, dass die Ontologie gleichbleibend erweitert und spezialisiert werden kann. Dies bedeutet, dass neue Terme für spezielle Verwendungen basierend auf dem bestehenden Vokabular definiert werden können, ohne dass bisherige Definitionen geändert werden müssen.

**Minimale Beeinflussung durch die Kodierung** Auf der Wissensebene sollte die Konzeptualisierung ohne Abhängigkeit von der Repräsentation auf einer bestimmten Symbol-ebene spezifiziert werden. Eine Beeinflussung durch die Kodierung entsteht, wenn Entscheidungen bezüglich der Repräsentation allein basierend auf der Annehmlichkeit der Notation und der Implementierung getroffen werden. Eine solche Beeinflussung der Kodierung sollte minimiert werden, da das Wissen teilende Agenten mit unterschiedlichen Systemen und Arten von Repräsentationen implementiert werden können.

**Minimale ontologische Bindung** Eine Ontologie sollte die minimale ontologische Bindung erfordern, welche hinreichend ist, um die beabsichtigten Wissensaustauschaktivitäten zu unterstützen. Eine Ontologie sollte so wenig Annahmen wie möglich über die modellierte

Welt treffen, um den an die Ontologie gebundenen Parteien die Freiheit zu geben, die Ontologie nach Bedarf zu spezialisieren und zu instantiieren. Da die ontologische Bindung auf der konsistenten Verwendung des Vokabulars beruht, kann die ontologische Bindung durch die Spezifikation der schwächsten Theorie (welche die meisten Modelle erlaubt) und die Definition nur der Terme, welche für die Kommunikation des mit der Theorie konsistenten Wissens essentiell sind, minimiert werden.

Gruber kommt zu dem Ergebnis, dass die Evaluation der Entwurfsentscheidungen anhand der genannten Kriterien vom verfügbaren Wissen und den für die Domäne antizipierten Anwendungen abhängig ist.

### **Rollen und Vorteile von Ontologien**

Jedes (symbolische) informationsverarbeitende System besitzt nach Guarino [44] seine eigene Ontologie, da es den Symbolen, welche entsprechend einer bestimmten Sichtweise auf die Welt verwendet werden, eine Bedeutung beimisst. Ein informationsverarbeitendes System besteht dabei aus den folgenden drei Komponenten: Anwendungsprogrammen, Informationsquellen wie Datenbanken und/oder Wissensbasen und Bedienschnittstellen. Die Einflüsse von Ontologien auf informationsverarbeitende Systeme lassen sich entlang zweier orthogonaler Dimensionen einordnen: einer zeitlichen Dimension und einer strukturellen Dimension.

Die Rollen und Vorteile von Ontologien zur Entwicklungszeit und zur Laufzeit lassen sich nach Guarino wie folgt beschreiben:

**Entwicklungszeit** Die Verwendung von Ontologien zur Entwicklungszeit erlaubt eine höhere Ebene der Wiederverwendung, nämlich die Wiederverwendung von Wissen statt von Software. Darüber hinaus erlaubt sie die Wiederverwendung und gemeinsame Nutzung von Anwendungsdomänenwissen unter Nutzung eines gemeinsamen Vokabulars auf heterogenen Softwareplattformen. Weiterhin ermöglicht sie es dem Entwickler, sich auf die Struktur der Domäne und die zu lösende Aufgabe zu konzentrieren und bewahrt ihn somit vor zu viel Beschäftigung mit Implementierungsdetails. Des Weiteren bietet sie ein mächtiges Werkzeug, welches die Qualität des Analyseprozesses erhöht und auch für das Reengineering nützlich ist.

**Laufzeit** Die Verwendung von Ontologien zur Laufzeit ermöglicht die Kommunikation zwischen Softwareagenten mit Hilfe von Nachrichten, welche Ausdrücke enthalten, die mit Bezug auf eine Ontologie formuliert sind.

Die Rollen und Vorteile von Ontologien in Bezug auf die einzelnen Komponenten eines informationsverarbeitenden Systems lassen sich nach Guarino wie folgt darstellen:



**Datenbank** Eine Ontologie kann mit einem Datenbankschema verglichen werden. Sie kann somit eine wichtige Rolle in der Anforderungsanalyse und in der konzeptuellen Modellierung spielen. Das konzeptuelle Modell kann als Ontologie repräsentiert und von dieser auf konkrete Zielplattformen abgebildet werden. Ein weiteres Anwendungsbeispiel für Ontologien ist die Informationsintegration. So können etwa heterogene konzeptuelle Schemata auf eine gemeinsame Top-Level-Ontologie abgebildet werden.

**Bedienschnittstelle** Eine wichtige Rolle einer Ontologie innerhalb einer Bedienschnittstelle besteht darin, dass sie das Stellen von Anfragen an sich selbst und das Durchsuchen durch den Benutzer erlaubt. Auf diese Weise ist sich der Benutzer der Existenz der Ontologie bewusst und kann sich ein besseres Verständnis des verwendeten Vokabulars aneignen. Hierdurch wird es ihm möglich, Anfragen mit der gewünschten Spezifität zu stellen. Eine weitere nützliche Aufgabe, die eine Ontologie in Bezug auf die Bedienschnittstelle wahrnehmen kann, ist die Abtrennung des Vokabulars: Der Benutzer kann frei seine eigenen natürlichsprachlichen Ausdrücke einführen, die dann auf das Vokabular der Ontologie abgebildet werden.

**Anwendungsprogramm** In Bezug auf Anwendungsprogramme können Ontologien die explizite Repräsentation des gesamten Domänenwissens übernehmen, das implizit in das Anwendungsprogramm kodiert ist. Dies resultiert in großen Vorteilen bezüglich der Einfachheit der Wartung, der Erweiterbarkeit und der Flexibilität. Zumindest die ontologische Bindung des Anwendungsprogramms sollte explizit gemacht werden. Auf diese Weise können Ontologien helfen, die Transparenz von Anwendungsprogrammen zu erhöhen.

### 2.2.2. Semantische Technologien

Im Folgenden werden verschiedene für diese Arbeit relevante Grundlagentechnologien aus dem Bereich des *Semantic Web* vorgestellt, die üblicherweise als *Semantische Technologien* bezeichnet werden. Diese umfassen Wissensrepräsentationssprachen für Ontologien sowie Methoden und Werkzeuge zur Erstellung, Wartung und Anwendung von Ontologien, welche inzwischen nicht nur im Bereich des World Wide Web, sondern auch in vielen anderen Bereichen der Informatik wie Wissensmanagement, Kognitive Systeme, Ambiente Intelligenz, Softwaretechnik, Maschinelles Lernen etc. zur Anwendung kommen [50].

Das Schichtenmodell des Semantic Web ist in Abbildung 2.10 dargestellt. Die für diese Arbeit relevanten Komponenten werden nachfolgend kurz erläutert.

**Uniform Resource Identifier (URI)** Ein *Uniform Resource Identifier* (URI) ist nach [50] eine Zeichenfolge, die auf einfache und erweiterbare Weise erzeugt wird und abstrakte oder

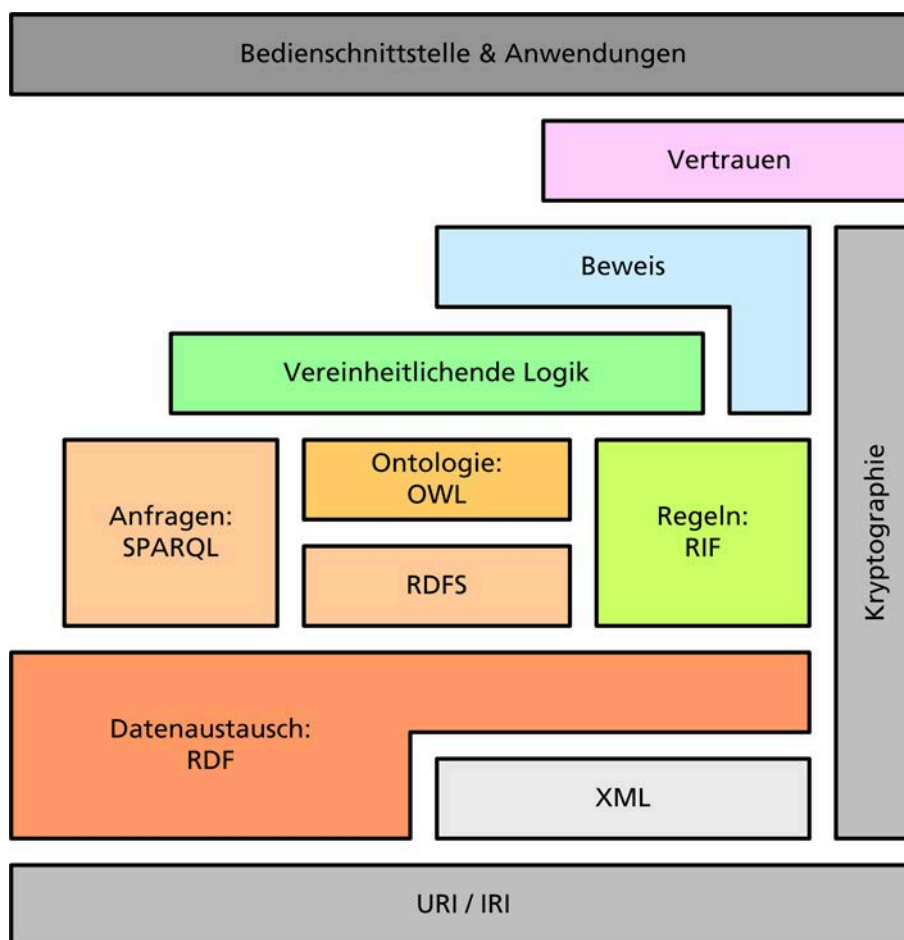


Abb. 2.10.: Das Schichtenmodell des Semantic Web [106].

auch physikalische Ressourcen bezeichnet. URIs werden insbesondere im World Wide Web verwendet, um Webseiten oder andere Dateien zu bezeichnen. Darüber hinaus können URIs jedoch auch unabhängig vom World Wide Web als genereller Mechanismus zur Erzeugung von eindeutigen Bezeichnern verwendet werden.

**eXtensible Markup Language (XML)** Die *eXtensible Markup Language* (XML) ist nach [50] eine Meta-Sprache zur Erstellung von Markup-Sprachen. XML legt die logische Struktur von Dokumenten fest und bietet eine einfache und sehr universell einsetzbare Möglichkeit Daten zu speichern. Selbst komplexe Datenmodelle können in XML serialisiert werden. XML selbst verwendet als Datenmodell eine Baumstruktur. Jedes XML-Tag entspricht dabei einem Baumknoten mit einem Namen im Datenmodell und jedes verschachtelte Tag einem Kindknoten.

**Resource Description Framework (RDF)** Das *Resource Description Framework* (RDF) ist nach [50] eine formale Sprache für die Beschreibung von allgemeinen Beziehungen zwischen Ressourcen. Mit Hilfe von RDF soll es Anwendungen ermöglicht werden, Daten im

World Wide Web auszutauschen, ohne dass ihre ursprüngliche Bedeutung dabei verloren geht. Aus diesem Grund wird RDF oft als grundlegendes Darstellungsformat für die Entwicklung des Semantic Web betrachtet.

Ein RDF-Dokument beschreibt einen gerichteten Graphen, wobei sowohl Knoten als auch Kanten mit eindeutigen Bezeichnern beschriftet sind. Jeder RDF-Graph kann vollständig durch die Angabe seiner Kanten beschrieben werden, wobei diese immer einem so genannten RDF-Tripel bestehend aus Subjekt, Prädikat und Objekt entsprechen. Diese drei Bestandteile werden zumeist durch URIs beschrieben. Nur das Objekt kann gegebenenfalls auch ein RDF-Literal sein. Zur Serialisierung von RDF stehen sowohl die (inoffizielle) RDF-Syntax *Turtle* als auch eine XML-basierte Schreibweise zur Verfügung.

**RDF Schema (RDFS)** *RDF Schema* (RDFS) ist ein spezielles RDF-Vokabular, das es ermöglicht, so genanntes *terminologisches Wissen* oder auch *Schemawissen* über die in einem Vokabular verwendeten Begriffe zu spezifizieren [50]. RDFS stellt universelle Ausdrucksmittel bereit, welche es erlauben, innerhalb eines RDFS-Dokumentes Aussagen über die semantischen Beziehungen der Termini eines beliebigen benutzerdefinierten Vokabulars zu machen. Es handelt sich somit bei RDFS um eine Wissensrepräsentations- und Ontologiesprache. RDFS wird mitunter auch als Beschreibungssprache für so genannte *leichtgewichtige* (engl. *lightweight*) Ontologien bezeichnet. Die Modellierungsfähigkeit von RDFS unterliegt jedoch einigen grundsätzlichen Einschränkungen. Die gravierendste besteht darin, dass es in RDFS nicht möglich ist auszudrücken, dass etwas nicht gilt.

**Web Ontology Language (OWL)** Die *Web Ontology Language* (OWL) ist eine Ontologiesprache, welche über umfangreiche Ausdrucksmittel als Teil der Sprache verfügt [50]. Mit Hilfe von zusammengesetzten Konzepten können komplexe Strukturen beschrieben, konjunktiv oder disjunktiv verknüpft und auch negiert werden. Bei der Entwicklung von OWL wurde besonderes Augenmerk auf das Gleichgewicht zwischen Ausdruckstärke und effizientem Schlussfolgern gelegt. Je nach Anforderungen stehen drei verschiedene Teilsprachen von OWL zu Verfügung: *OWL Full*, *OWL DL* und *OWL Lite*. Dabei handelt es sich bei OWL Lite um eine echte Teilsprache von OWL DL, welche wiederum eine echte Teilsprache von OWL Full ist. Für OWL wurden zwei verschiedene Syntaxen entwickelt. Die *OWL-RDF-Syntax* basiert auf RDF und wird für den Datenaustausch verwendet. Die *abstrakte OWL-Syntax* ist im Allgemeinen für den Menschen leichter zu lesen, ist jedoch nur für OWL DL verfügbar.

In OWL Full ist die uneingeschränkte Benutzung aller OWL-Sprachelemente gemeinsam mit den RDF(S)-Sprachelementen erlaubt, solange das Resultat gültiges RDF ist. Ohne gewisse Einschränkungen der durch RDF(S) gebotenen Freiheitsgrade ist das Berechnen von Inferenzen in OWL Full jedoch unentscheidbar. Die hieraus resultierenden Schwierigkeiten haben zur Definition von OWL DL und OWL Lite geführt. OWL Full wird von aktuellen Inferenzmaschinen

oft nur bedingt und meist überhaupt nicht unterstützt.

OWL DL wurde so definiert, dass es entscheidbar ist. Daher dürfen in OWL DL manche Sprachelemente von OWL Full nur eingeschränkt verwendet werden. Historisch kann OWL DL auf so genannte *semantische Netzwerke* zurückgeführt werden [50]. Diese dienten zur Modellierung von einfachen Zusammenhängen zwischen Individuen und Klassen mit Hilfe von Rollen, waren jedoch zu Beginn in ihrer Bedeutung nur vage festgelegt. Dies erforderte eine Formalisierung ihrer Semantik, woraus schließlich die so genannten *Beschreibungslogiken* entstanden. Bei OWL DL handelt es sich tatsächlich im Wesentlichen um eine Beschreibungslogik, welche wiederum als Fragment der Prädikatenlogik erster Stufe aufgefasst werden kann. Die Beschreibungslogik, die OWL DL abdeckt, wird mit  $\mathcal{SHOIN}(\mathcal{D})$  bezeichnet. OWL DL ist aufgrund ihrer logischen Eigenschaften die wichtigste Teilsprache. Sie wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.

Bei OWL Lite handelt es sich um ein einfach zu implementierendes Sprachfragment, welches die wichtigsten Sprachelemente enthält. Es ist jedoch in der Praxis nur von eingeschränkter Bedeutung.

**SPARQL Protocol and RDF Query Language (SPARQL)** Bei der *SPARQL Protocol and RDF Query Language (SPARQL)* handelt es sich um einen Standard für die Abfrage von in RDF spezifizierten Informationen sowie für die Darstellung der Resultate [50]. SPARQL basiert im Wesentlichen auf einfachen RDF-Anfragen in Form von Graph-Mustern, enthält jedoch auch erweiterte Funktionen für die Konstruktion komplexerer Anfragemuster, für die Verwendung zusätzlicher Filterbedingungen und für die Formatierung der Ausgabe. Die SPARQL-Spezifikation enthält zudem die Definition einer formalen Semantik. Der Kern der SPARQL-Semantik besteht in der so genannten *SPARQL-Algebra*, welche ein System aus klar definierten Rechenoperationen darstellt, mit denen das Ergebnis einer Anfrage ausgerechnet werden kann. In dieser Hinsicht besteht eine gewisse Ähnlichkeit zur Anfragesprache SQL für relationale Datenbanken, welche auf der so genannten *relationalen Algebra* beruht.

**Konjunktive Anfragen** Für OWL gibt es derzeit keine Standard-Anfragesprache. So genannte konjunktive Anfragen werden jedoch als grundlegender Formalismus für OWL DL betrachtet [50]. Konjunktive Anfragen bestehen aus Konjunktionen von Bedingungen, wobei es sich bei den einzelnen Bedingungen um einfache beschreibungslogische Formeln ohne logische Operatoren oder die Negation solcher Bedingungen handelt.

**Formale Semantik und automatisiertes Schlussfolgern** Der Begriff Semantik (von griechisch  $\sigma\eta\mu\alpha\nu\tau\iota\kappa\omicron\varsigma$  „zum Zeichen gehörend“) wird in verschiedenen Kontexten (Linguistik, Programmiersprachen usw.) sehr unterschiedlich verwendet [50]. Am treffendsten lässt

er sich wohl mit dem Wort „Bedeutung“ umschreiben. Die logische Dimension des Semantik-Begriffs wird häufig mit dem Begriff *formale Semantik* bezeichnet. Eine Logik  $L = (\mathcal{S}, \models)$  setzt sich aus einer Menge von Sätzen  $\mathcal{S}$  und einer Schlussfolgerungsrelation  $\models \subseteq 2^{\mathcal{S}} \times \mathcal{S}$  zusammen. Für eine konkrete Logik gibt es eine Vielzahl von Möglichkeiten, die Schlussfolgerungsrelation zu definieren. Eine hierfür häufig verwendete Methode wird als *modelltheoretische Semantik* bezeichnet. Die grundlegende Idee besteht hierbei darin, die syntaktischen Aussagen einer Logik (also die Sätze) zu so genannten *Interpretationen* ins Verhältnis zu setzen. In der formalen Logik werden bestimmte mathematische Strukturen als Interpretationen gewählt. Nach der Festlegung der Interpretationen werden anschließend Kriterien definiert, anhand derer entschieden werden kann, ob eine konkrete Interpretation  $I$  einen konkreten Satz  $s \in \mathcal{S}$  erfüllt, d. h. ein *Modell* von  $s$  ist ( $I \models s$ ). Basierend auf dieser „Modellrelation“ wird dann die eigentliche Schlussfolgerungsrelation anhand des folgenden Kriteriums definiert [50]: Ein Satz  $s \in \mathcal{S}$  folgt aus einer Menge von Sätzen  $S \subseteq \mathcal{S}$  ( $S \models s$ ) genau dann, wenn jede Interpretation  $I$ , die jeden Satz  $s'$  aus  $S$  erfüllt ( $I \models s'$  für alle  $s' \in S$ ), auch Modell von  $s$  ist ( $I \models s$ ).

Für RDF(S) kann eine modelltheoretische Semantik angegeben werden, welche mathematisch präzise definiert, wann ein RDF(S)-Graph aus einer Menge anderer RDF(S)-Graphen folgt [50]. Eine solche modelltheoretische Semantik eignet sich zwar sehr gut, um das gewünschte Verhalten einer Logik bezüglich ihrer Schlussfolgerungen zu charakterisieren, für eine direkte algorithmische Verwendung eignet sie sich jedoch nicht. Zur Entscheidung, ob ein Graph aus einer Menge von Graphen folgt, müssten im Prinzip alle RDF(S)-Interpretationen betrachtet werden. Dies ist jedoch nicht möglich, da es immer unendlich viele Interpretationen gibt und selbst eine einzige Interpretation unendlich viele Elemente enthalten kann. Hierbei handelt es sich um ein grundsätzliches Problem, das für viele Logiken zutrifft. Aus diesem Grund wurden Verfahren entwickelt, welche es erlauben, die Gültigkeit von Schlussfolgerungen syntaktisch zu entscheiden. Ein solches syntaktisches Verfahren besteht etwa in der Angabe von Deduktions- oder Ableitungsregeln. Die Gesamtheit der für eine Logik gegebenen Ableitungsregeln wird als *Deduktionskalkül* bezeichnet. Für RDF(S) lässt sich ein solches Kalkül angeben, welches die RDF(S)-Folgerung syntaktisch beschreibt.

Auch für OWL kann eine modelltheoretische Semantik angegeben werden. Dies kann entweder durch Rückführung auf die Prädikatenlogik erster Stufe erfolgen oder alternativ als so genannte *extensionale Semantik* [50]. Es kann gezeigt werden, dass beide Semantikdefinitionen äquivalent sind. Für das automatisierte Schlussfolgern mit OWL-Ontologien verwenden die meisten hochperformanten OWL-Beweiser das so genannte Tableauverfahren mit Blocking [50]. Der einzige zurzeit bekannte alternative Ansatz, der zu ähnlich effizienten Implementierungen führt, besteht in der Umwandlung der OWL-Wissensbasen in disjunktives Datalog und dem anschließenden Aufruf eines Datalog-Beweisers. Die entsprechenden Algorithmen wurden

im KAON2-System<sup>3</sup> umgesetzt. Dieses Verfahren eignet sich insbesondere für Wissensbasen mit großen ABoxen (enthalten das assertionale Instanzwissen), da die sehr aufwändigen Teilalgorithmen zur Übersetzung in disjunktives Datalog im Wesentlichen nur auf der TBox (enthält das terminologische Schemawissen) operieren.

**Semantic Web Rule Language (SWRL)** Die *Semantic Web Rule Language* (SWRL)<sup>4</sup> basiert auf einer Kombination der *OWL DL* und *OWL Lite* Untersprachen der *Web Ontology Language* mit den *Unary/Binary Datalog RuleML* Untersprachen der *Rule Markup Language*<sup>5</sup>. Sie erweitert die Menge der OWL-Axiome um Horn-artige Regeln und ermöglicht so die Kombination von Horn-artigen Regeln mit einer OWL-Wissensbasis. Die Regeln besitzen die Form einer Implikation zwischen einer Vorbedingung und einer Konsequenz. Die intendierte Bedeutung dieser Regeln lässt sich wie folgt beschreiben: Wann immer die Bedingungen in der Vorbedingung gültig sind, müssen auch die in der Konsequenz spezifizierten Bedingungen gültig sein.

Sowohl die Vorbedingung als auch die Konsequenz bestehen aus null oder mehr Atomen. Eine leere Vorbedingung wird als trivialerweise wahr angenommen (d. h., dass sie von jeder Interpretation erfüllt werden muss), so dass die Konsequenz ebenfalls von jeder Interpretation erfüllt werden muss. Eine leere Konsequenz wird als trivialerweise falsch angenommen (d. h., dass sie von keiner Interpretation erfüllt wird), so dass auch die Vorbedingung von keiner Interpretation erfüllt werden darf. Mehrere Atome werden als Konjunktion behandelt.

Atome in diesen Regeln können von der folgenden Form sein:  $C(x)$ ,  $P(x,y)$ ,  $\text{sameAs}(x,y)$  oder  $\text{differentFrom}(x,y)$ , wobei  $C$  eine OWL-Beschreibung und  $P$  eine OWL-Property bezeichnen und  $x, y$  entweder Variablen, OWL-Individuen oder OWL-Datenwerte sind.

Für SWRL existiert eine XML-Syntax, die auf RuleML und der OWL-XML-Präsentationsyntax basiert. Weiterhin existiert eine konkrete RDF-Syntax, die auf der OWL-RDF/XML-Austauschsyntax basiert.

Die modelltheoretische Semantik von SWRL ist eine direkte Erweiterung der modelltheoretischen Semantik von OWL. Die grundlegende Idee besteht in der Definition von *Bindungen*. Diese bilden eine Erweiterung der OWL-Interpretationen, indem sie Variablen auf Elemente der Domäne abbilden. Eine Regel wird von einer Interpretation genau dann erfüllt, wenn jede Bindung, welche die Vorbedingung erfüllt, auch die Konsequenz erfüllt. Die semantischen Bedingungen, die sich auf Axiome und Ontologien beziehen, bleiben unverändert. Dies bedeutet, dass eine Interpretation eine Ontologie genau dann erfüllt, wenn sie jedes Axiom (inklusive Regeln) und jeden Fakt in der Ontologie erfüllt.

---

<sup>3</sup>KAON2: <http://kaon2.semanticweb.org>

<sup>4</sup>SWRL: <http://www.w3.org/Submission/SWRL/>

<sup>5</sup>RuleML: <http://ruleml.org/>

### 2.2.3. Semantisches Wissen in der Robotik

Den Autoren von [49] zufolge ist eine steigende Tendenz zu erkennen, semantisches Wissen in der Robotik zu verwenden. Diese Tendenz ist in unterschiedlichen Formen in den verschiedenen Bereichen der Robotik erkennbar:

- Aktuelle Arbeiten im Bereich Kartierung und Lokalisierung versuchen, während der Kartenerstellung semantisch bedeutsame Strukturen aus den Sensordaten zu extrahieren oder semantisches Wissen im Prozess der Kartenerstellung zu verwenden, oder beides.
- Ein ähnlicher Trend kennzeichnet den Ansatz des kognitiven Computersehens für das Szenenverständnis.
- Aktuelle Anstrengungen im Bereich der Mensch-Roboter-Interaktion versuchen, den Roboter mit einem gewissen Verständnis der menschlichen Bedeutung von Wörtern, Gesten und Ausdrücken auszustatten.
- In verteilten Systemen wird ontologisches Wissen verstärkt eingesetzt, um die automatische Rekonfigurierbarkeit in den Bereichen flexible Automatisierung und ubiquitäre Robotik zu ermöglichen.
- Ontologisches Wissen wird aktuell außerdem verwendet, um die Interoperabilität von Robotik-Komponenten zu verbessern, die für unterschiedliche Systeme entwickelt wurden.

Die Problemstellung, Roboter mit der Fähigkeit auszustatten, sich semantisches Wissen anzueignen und zu nutzen, lässt sich laut [49] im Wesentlichen in drei Aspekte gliedern:

1. Die Erlangung von semantischem Wissen aus den Sensordaten an Bord eines mobilen Roboters (siehe etwa [69], [29], [79], [73] und [87]).
2. Die Verwendung von semantischem Wissen zur Verbesserung von Planungs- und Steuerungsaspekten im Roboter selbst (siehe etwa [18], [33], [98], [31] und [24]).
3. Die Verwendung von semantischem Wissen in der Robotik motiviert durch bestimmte Aufgaben, wie zum Beispiel die Mensch-Roboter-Interaktion (siehe [52]).

Im nachfolgenden Abschnitt wird näher auf einen für diese Arbeit besonders relevanten Ansatz eingegangen, der ein semantisches wissensbasiertes Rahmenwerk zur Verbesserung des Situationsbewusstseins von autonomen Unterwasserfahrzeugen beschreibt.

### **Semantisches Rahmenwerk zur Verbesserung des Situationsbewusstseins**

In [66] wird ein auf einem semantischen Weltmodell basierendes Rahmenwerk für die hierarchische, verteilte Repräsentation von Wissen in autonomen Unterwasserfahrzeugen (engl. *Autonomous Underwater Vehicle* (AUV)) vorgestellt. Ziel des Rahmenwerks ist ein leistungsfähigeres und ganzheitlicheres System, welches die semantische Interoperabilität zwischen allen Informationsquellen umfasst. Durch ein solches System werden die Interoperabilität, der unabhängige Betrieb und das Situationsbewusstsein der eingebetteten serviceorientierten Agenten verbessert, was sich wiederum auf die Flexibilität der Missionen, die Robustheit und den Autonomiegrad auswirkt. Das Rahmenwerk basiert auf der Idee, dass heterogene Daten sehr unterschiedlichen Typs von verschiedenen Schichten bearbeitet werden müssen, bis sie schließlich in einem geeigneten Format und an der richtigen Stelle den für die Entscheidungsfindung zuständigen Agenten auf den höheren Ebenen zur Verfügung gestellt werden. In [66] wird gezeigt, wie Schritt für Schritt mit Hilfe von semantischen Technologien von den realen Sensordaten abstrahiert werden kann.

Die menschliche Fähigkeit, mit hochdynamischen und komplexen Umgebungen umzugehen und sie zu verstehen, wird als *Situationsbewusstsein* bezeichnet. Das Situationsbewusstsein lässt sich nach [66] in drei verschiedene Ebenen unterteilen: die Wahrnehmung der Umgebung, das Verständnis der Situation und die Projektion des zukünftigen Zustands. Nach John R. Boyd [19] erfolgt die Entscheidungsfindung in Zyklen aus Beobachten, Orientieren, Entscheiden und Handeln (engl. *observe-orient-decide-act* (OODA)). Die Beobachtungskomponente entspricht dabei der Wahrnehmungsebene des Situationsbewusstseins. Die Orientierungskomponente umfasst das bisher erworbene Wissen und Verständnis der Situation. Die Entscheidungskomponente repräsentiert die Ebenen des Verstehens und der Projektion. Diese Phase bietet die zentrale Möglichkeit zur Adaption, bevor der Zyklus mit dem finalen Handlungsschritt abgeschlossen wird.

In aktuellen Systemen übernimmt in der Regel der menschliche Operator die Entscheidungsfindungsphase. Sofern eine breitbandige Kommunikationsverbindung existiert, verbleibt der Operator auch während der Ausführung in der Kontrollschleife. Ein Beispiel hierfür bilden etwa die ferngesteuerten Unterwasserfahrzeuge (engl. *Remote Operated Underwater Vehicles* (ROV)). Ist die Kommunikationsverbindung hingegen schlecht, unzuverlässig oder nicht möglich, versucht der Operator basierend auf Erfahrungswissen alle möglichen Verhalten zum Umgang mit Ausführungsalternativen zu integrieren. Unerwartete und unvorhersehbare Situationen können jedoch zu einem Abbruch der Mission und sogar zum Verlust des Unterwasserfahrzeugs führen. Beispiele für eine solche Architektur bilden die AUVs. Um einen autonomen Entscheidungsfindungszyklus zu erreichen, sind nach [66] zwei zusätzliche Komponenten notwendig: Ein Statusmonitor und ein Missionsplanadapter. Der Statusmonitor meldet jegliche während der Planausführung detektierten Abweichungen. Können diese Abweichungen nicht von der



Missionsausführung geeignet behandelt werden, wird der Missionsplanadapter aufgerufen, der einen neuen modifizierten Plan erzeugt, welcher mit dem aktualisierten Wissen über die Welt in Einklang steht (siehe Abbildung 2.11).

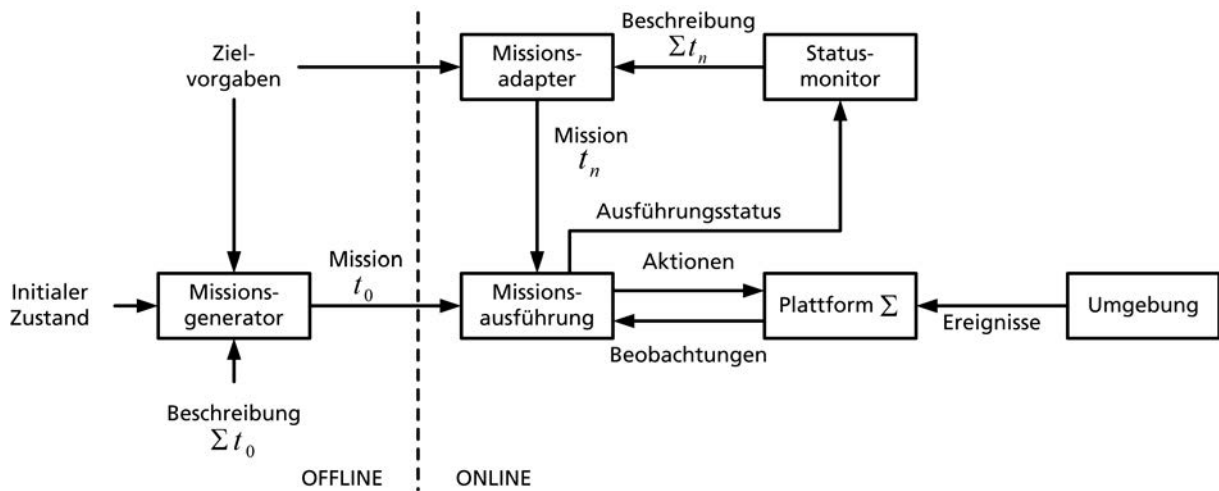


Abb. 2.11.: OODA-Entscheidungsfindungszyklus für unbemannte Systeme [66].

Die Bemühungen zur Steigerung der Operabilität von AUVs haben sich in den vergangenen Jahren auf die Erhöhung der Überlebensfähigkeit der AUVs durch Reduzierung von Anfälligkeiten und Schwachstellen konzentriert. Der Fokus lag hierbei vor allem auf der Anpassung der geplanten Fahrzeug-Trajektorien. Wird das Fahrzeug jedoch mit unvorhergesehenen Ereignissen, wie zum Beispiel unerwarteten Hardwarefehlern oder ungeplanten Interaktionen mit der Umgebung, konfrontiert, so sollte sich der Fokus der Mission darauf richten, alternative Kombinationen der verbleibenden Ressourcen zu nutzen. Die autonome, eingebettete Fähigkeit zur Wiederherstellung ist einer der zentralen Faktoren zur Steigerung der Lebensdauer von AUVs. Nach [66] kann diese Fähigkeit durch Adaption der Missionspläne erreicht werden. Eine wesentliche Voraussetzung hierfür ist die Verfügbarkeit von genauen Informationen zur Erkennung von Fehlern und ihren Ursachen.

Das Ziel des Situationsbewusstseins von autonomen Fahrzeugen besteht darin, dem Fahrzeug ein Bild des großen Ganzen (engl. *big picture*) zu vermitteln. Dieses Bild setzt sich dabei aus den in früheren Missionen gemachten Erfahrungen (Orientierung) und den während der Mission mit Hilfe der Sensoren gewonnenen Informationen (Beobachtung) zusammen. Nach [66] können die TBox und die ABox einer Wissensbasis der Orientierungs- bzw. der Beobachtungskomponente zugeordnet werden. Eine solche Wissensrepräsentation ermöglicht darüber hinaus den Entscheidungsfindungsprozess der serviceorientierten Agenten sowie die Extraktion bzw. Inferenz von neuem Wissen aus den beobachteten Daten. In [66] wird eine Menge von Ontologien vorgestellt, die zur Repräsentation von Wissen für das Situationsbewusstsein von autonomen Fahrzeugen entwickelt wurde. Die Repräsentation der Ontologien erfolgt dabei mit Hilfe von *OWL*.

Das in [66] vorgestellte System implementiert die vier Phasen des OODA-Entscheidungsfindungszyklus (siehe Abbildung 2.12). Der *Statusmonitor* meldet Veränderungen in der Umgebung und am internen Zustand der Plattform an das Weltmodell. Das *Weltmodell* speichert das ontologiebasierte Wissen, welches durch die Erfahrung von Experten (Orientierung) und die Beobachtung von Ereignissen, die vom Statusmonitor empfangen werden, zur Verfügung gestellt wird. Der *Missionsplaner* erzeugt und adaptiert Missionspläne basierend auf den Benachrichtigungen und Fähigkeiten, die von der Missionsausführung und dem Weltmodell gemeldet werden. Die *Missionsausführung* ist basierend auf den vom Missionsplaner empfangenen Aktionen für die Ausführung der Missionskommandos in der funktionalen Schicht verantwortlich. Um die Unabhängigkeit zwischen der Architektur und der funktionalen Schicht des Fahrzeugs zu gewährleisten, wurde eine Abstraktionsebenenschnittstelle (engl. *Abstract Layer Interface* (ALI)) entwickelt.

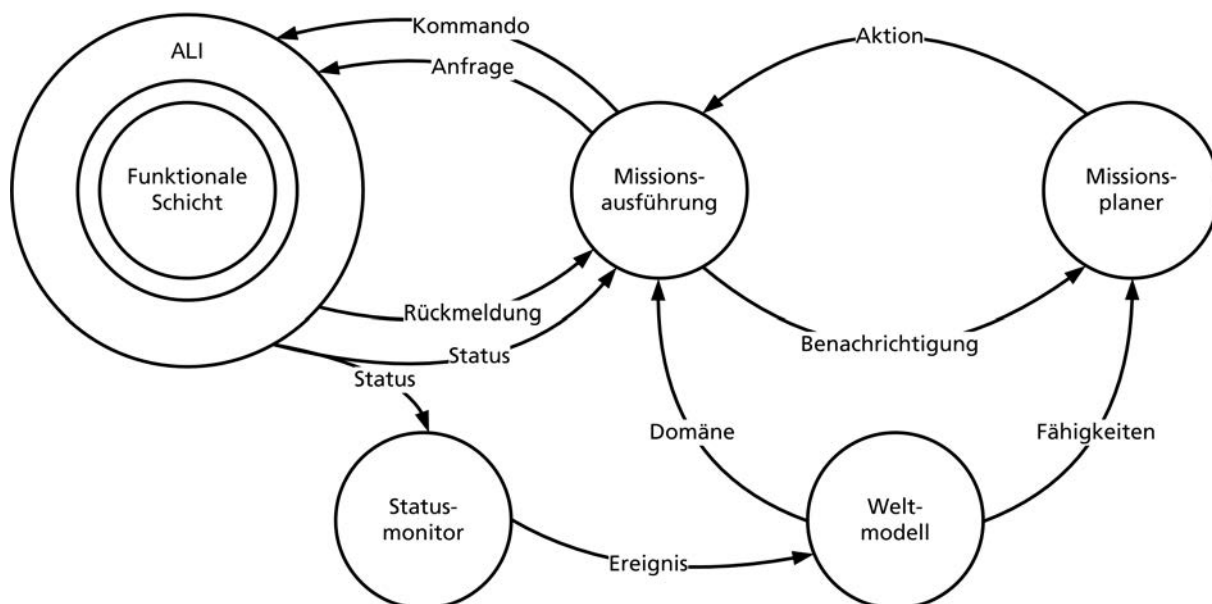


Abb. 2.12.: Realisierung des OODA-Entscheidungsfindungszyklus für unbemannte Systeme [66].

Das in [66] vorgestellte semantische wissensbasierte Rahmenwerk zur Verbesserung des Situationsbewusstseins von autonomen Unterwasserfahrzeugen ist in Abbildung 2.13 dargestellt. Jeder serviceorientierte Agent verfügt über eine eigene Anwendungsontologie. Diese repräsentiert das Bewusstsein des Agenten für die Situation, indem sie spezifische Konzepte, die für die Expertise oder den Service des Agenten von Bedeutung sind, beinhaltet. Die jeweiligen Umsetzungen des Rahmenwerks für die Agenten *Statusmonitor* und *Missionsplaner* sowie die zugehörigen Anwendungsontologien werden ausführlich in [66] beschrieben.

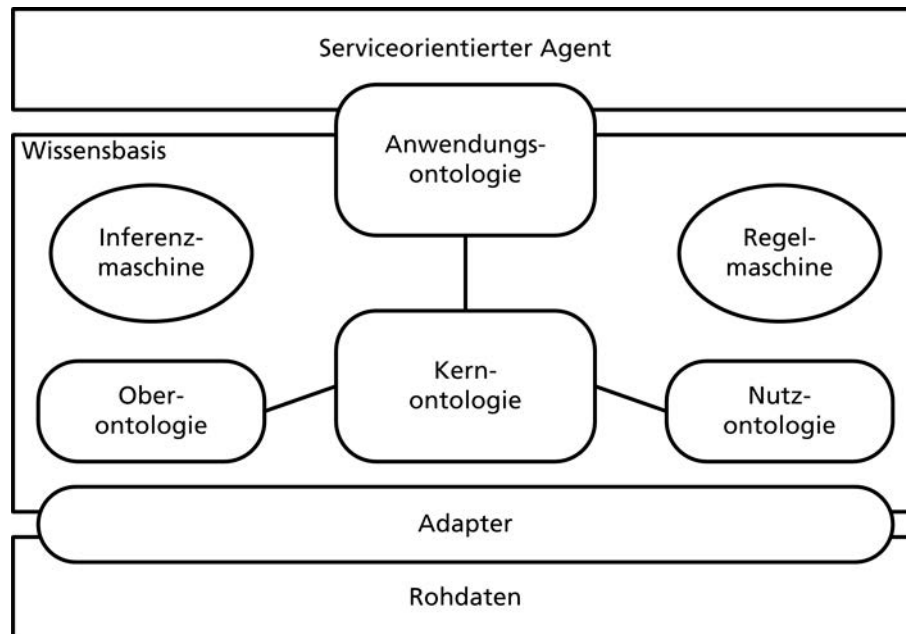


Abb. 2.13.: Konzeptrepräsentation (Kern- und Anwendungsontologie), Instanzenerzeugung (Adapter) und -behandlung (Inferenz und Entscheidungsfindungsagent) für das Situationsbewusstsein von autonomen Fahrzeugen [66].

## 2.3. Planungsverfahren und Ausführungssysteme

*Planen ist die schlussfolgernde Seite des Handelns. Es ist ein abstrakter, expliziter Überlegungsprozess, der Aktionen durch Antizipieren der erwarteten Resultate auswählt und organisiert. Diese Überlegung zielt darauf ab, bestimmte vorgegebene Ziele so gut wie möglich zu erreichen. Automatisierte Planung ist ein Bereich der Künstlichen Intelligenz, der diesen Überlegungsprozess unter Gesichtspunkten der Berechenbarkeit untersucht [40].*

In diesem Abschnitt werden verschiedene für diese Arbeit relevante Planungsverfahren und Ausführungssysteme erläutert. Der Fokus liegt dabei auf Hierarchischen Aufgabennetzwerken. Nachdem auf die Grundlagen des Planens mit Hierarchischen Aufgabennetzwerken eingegangen wurde, werden anschließend verschiedene Planungsverfahren und Ausführungssysteme in diesem Bereich vorgestellt. Dies umfasst insbesondere *Flexible Programme* (FPs) und *Concurrent Hierarchical Plans* (CHiPs).

### 2.3.1. Hierarchische Aufgabennetzwerke

Die Planung mit Hierarchischen Aufgabennetzwerken (engl. *Hierarchical Task Networks* (HTNs)) [40] ähnelt der klassischen Planung dahingehend, dass jeder Zustand der Welt durch eine Menge von Axiomen beschrieben wird und jede Aktion einem deterministischen Zustandsübergang entspricht. HTN-Planer unterscheiden sich jedoch von der klassischen Planung dar-

in, wofür und wie sie planen. Für einen HTN-Planer besteht die Zielsetzung nicht darin, eine Menge von Zielen zu erreichen, sondern eine Menge von Aufgaben zu erfüllen. Die Eingabe für das Planungssystem beinhaltet eine Menge von Operatoren, die denen der klassischen Planung ähneln, sowie eine Menge von Methoden, die eine Anleitung dafür bieten, wie bestimmte Aufgaben in eine bestimmte Menge von Unteraufgaben zerlegt werden können. Während der Planung werden nichtelementare Aufgaben rekursiv in immer kleinere Unteraufgaben zerlegt, bis elementare Aufgaben erreicht werden, welche direkt mit Hilfe von Planungsoperatoren erfüllt werden können. Laut [40] wurden HTN-Planungsverfahren häufiger für praktische Anwendungen verwendet als jedes andere Planungsverfahren. Dies liegt unter anderem daran, dass HTN-Methoden eine zweckmäßige Möglichkeit zur Verfügung stellen, Vorgehensweisen zur Problemlösung zu formulieren, die der Art und Weise entsprechen, wie menschliche Domänenexperten ein Planungsproblem lösen würden.

**Definition 2.1.** Ein *Aufgabennetzwerk* ist ein Paar  $w = (U, C)$ , wobei  $U$  eine Menge von Aufgabenknoten und  $C$  eine Menge von Bedingungen ist, wie sie im nachfolgenden Text beschrieben werden. Jeder Aufgabenknoten  $u \in U$  enthält eine Aufgabe  $t_u$ .  $w$  ist *elementar*, wenn alle seine Aufgaben  $\{t_u | u \in U\}$  elementar sind; andernfalls ist  $w$  *nichtelementar*.

Jede Bedingung in  $C$  spezifiziert eine Anforderung, die von jedem Plan, der eine Lösung für das Planungsproblem darstellt, erfüllt werden muss. Zur Beschreibung der Bedingungen ist die folgende Notation hilfreich: Sei  $\pi$  eine Lösung für  $w$ ,  $U' \subseteq U$  eine Menge von Aufgabenknoten in  $w$  und  $A$  die Menge aller Aktionen  $a_i \in \pi$ , so dass im Zerlegungsbaum von  $\pi$  gilt, dass  $a_i$  ein Nachfolger eines Knotens in  $U'$  ist. Dann beschreibt  $\text{first}(U', \pi)$  die Aktion  $a_i \in A$ , die als erstes auftritt, d. h.  $i \leq j$  für alle  $a_j \in A$ , und  $\text{last}(U', \pi)$  beschreibt die Aktion  $a_k \in A$ , die als letztes auftritt, d. h.  $k \geq j$  für alle  $a_j \in A$ .

In [40] werden die folgenden Arten von Bedingungen betrachtet:

- Eine *Vorrang-Bedingung* ist ein Ausdruck der Form  $u \prec v$ , wobei  $u$  und  $v$  Aufgabenknoten sind. Der Ausdruck besagt, dass in jeder Lösung  $\pi$  des Planungsproblems  $\mathcal{P}$  die Aktion  $\text{last}(\{u\}, \pi)$  der Aktion  $\text{first}(\{v\}, \pi)$  vorangehen muss.
- Eine *Vorher-Bedingung* ist eine Bedingung der Form  $\text{before}(U', l)$ , wobei  $U' \subseteq U$  eine Menge von Aufgabenknoten und  $l$  ein Literal ist. Sie besagt, dass in jeder Lösung  $\pi$  des Planungsproblems  $\mathcal{P}$  das Literal  $l$  in dem Zustand, der direkt vor  $\text{first}(U', \pi)$  auftritt, wahr sein muss.
- Eine *Nachher-Bedingung* hat die Form  $\text{after}(U', l)$ . Sie gleicht der Vorher-Bedingung, außer dass sie besagt, dass  $l$  in dem Zustand wahr sein muss, der direkt nach  $\text{last}(U', \pi)$  auftritt.

- Eine *Zwischen-Bedingung* hat die Form  $\text{between}(U', U'', l)$ . Sie besagt, dass das Literal  $l$  im Zustand direkt nach  $\text{last}(U', \pi)$ , im Zustand direkt vor  $\text{first}(U'', \pi)$  und in allen Zuständen zwischen diesen beiden Zuständen wahr sein muss.

**Definition 2.2.** Eine *HTN-Planungsdomäne* ist ein Paar

$$\mathcal{D} = (O, M) \quad [2.1]$$

und ein *HTN-Planungsproblem* ist ein 4-Tupel

$$\mathcal{P} = (s_0, w, O, M), \quad [2.2]$$

wobei  $s_0$  den initialen Zustand,  $w$  das initiale Aufgabennetzwerk,  $O$  eine Menge von Operatoren und  $M$  eine Menge von HTN-Methoden bezeichnet.

**Definition 2.3.** Eine *HTN-Methode* ist ein 4-Tupel

$$m = (\text{name}(m), \text{task}(m), \text{subtasks}(m), \text{constr}(m)), \quad [2.3]$$

wobei die einzelnen Elemente wie folgt beschrieben sind:

- $\text{name}(m)$  ist ein Ausdruck der Form  $n(x_1, \dots, x_k)$ , wobei  $n$  ein eindeutiges Methodensymbol ist (d. h. zwei Methoden in einer Planungsdomäne können niemals das gleiche Methodensymbol besitzen) und  $x_1, \dots, x_n$  alle Variablensymbole sind, die irgendwo in  $m$  auftreten.
- $\text{task}(m)$  ist eine nichtelementare Aufgabe.
- $(\text{subtasks}(m), \text{constr}(m))$  ist ein Aufgabennetzwerk.

Sei  $w = (U, C)$  ein Aufgabennetzwerk,  $u \in U$  ein Aufgabenknoten,  $t_u$  die zugehörige Aufgabe,  $m$  eine Instanz einer Methode in  $M$  und  $\text{task}(m) = t_u$ . Dann *zerlegt*  $m$  den Aufgabenknoten  $u$  in  $\text{subtasks}(m)$  und erzeugt das Aufgabennetzwerk

$$\delta(w, u, m) = ((U - \{u\}) \cup \text{subtasks}(m), C' \cup \text{constr}(m)), \quad [2.4]$$

wobei  $C'$  die wie folgt modifizierte Version von  $C$  ist:

- Jede Vorrang-Bedingung, die  $u$  enthält, wird durch eine Vorrang-Bedingung ersetzt, welche die Knoten von  $\text{subtasks}(m)$  enthält. Gilt beispielsweise  $\text{subtasks}(m) = \{u_1, u_2\}$ , dann wird die Bedingung  $u \prec v$  durch die Bedingungen  $u_i \prec v$  und  $u_2 \prec v$  ersetzt.

- Für jede Vorher-, Nachher- und Zwischen-Bedingung, in der es eine Menge von Aufgabenknoten  $U'$  gibt, die  $u$  enthält, wird  $U'$  mit  $(U' - \{u\}) \cup \text{subtasks}(m)$  ersetzt. Gilt beispielsweise  $\text{subtasks}(m) = \{u_1, u_2\}$ , dann wird die Bedingung  $\text{before}(\{u, v\}, l)$  mit der Bedingung  $\text{before}(\{u_1, u_2, v\}, l)$  ersetzt.

**Definition 2.4.** Wenn  $w = (U, C)$  elementar ist, dann ist ein Plan  $\pi = \langle a_1, a_2, \dots, a_k \rangle$  eine *Lösung* für  $\mathcal{P}$ , wenn es eine Grundinstanz  $(U', C')$  von  $(U, C)$  und eine totale Ordnung  $\langle u_1, u_2, \dots, u_k \rangle$  der Knoten in  $U'$  gibt, so dass alle der folgenden Bedingungen erfüllt sind:

- Die Aktionen in  $\pi$  sind diejenigen Aktionen, die von den Knoten  $u_1, u_2, \dots, u_k$  genannt werden, d. h.  $\text{name}(a_i) = t_{u_i}$  für  $i = 1, \dots, k$ .
- Der Plan  $\pi$  ist im Zustand  $s_0$  ausführbar.
- Die totale Ordnung  $\langle u_1, u_2, \dots, u_k \rangle$  erfüllt die Vorrang-Bedingungen in  $C'$ , d. h.  $C'$  enthält keine Vorrang-Bedingung  $u_i \prec u_j$ , so dass  $j \leq i$ .
- Für jede Bedingung  $\text{before}(U', l)$  in  $C'$  gilt, dass  $l$  in dem Zustand  $s_{i-1}$  wahr ist, der unmittelbar der Aktion  $a_i$  vorangeht, wobei  $a_i$  die vom ersten Knoten in  $U'$  genannte Aktion ist (d. h. von dem Knoten  $u_i \in U'$ , der in der totalen Ordnung  $\langle u_1, u_2, \dots, u_k \rangle$  zuerst kommt).
- Für jede Bedingung  $\text{after}(U', l)$  gilt, dass  $l$  in dem Zustand  $s_j$  wahr ist, der von der Aktion  $a_j$  erzeugt wird, wobei  $a_j$  die vom letzten Knoten in  $U'$  genannte Aktion ist (d. h. von dem Knoten  $u_j \in U'$ , der in der totalen Ordnung  $\langle u_1, u_2, \dots, u_k \rangle$  als letztes kommt).
- Für jede Bedingung  $\text{between}(U', U'', l)$  in  $C'$  gilt, dass  $l$  in jedem Zustand wahr ist, der zwischen  $a_i$  und  $a_j$  kommt, wobei  $a_i$  die vom letzten Knoten in  $U'$  genannte Aktion und  $a_j$  die vom ersten Knoten in  $U''$  genannte Aktion ist.

Ist  $w = (U, C)$  nichtelementar (d. h. wenigstens eine Aufgabe in  $U$  ist nichtelementar), dann ist  $\pi$  eine *Lösung* für  $\mathcal{P}$ , wenn es eine Folge von Aufgabenzerlegungen gibt, die auf  $w$  angewendet werden kann, um ein elementares Aufgabennetzwerk  $w'$  zu erzeugen, so dass  $\pi$  eine Lösung für  $w'$  ist. In diesem Fall ist der Zerlegungsbaum für  $\pi$  die Baumstruktur, die diesen Aufgabenzerlegungen entspricht.

**Vorteile und Nachteile** Verglichen mit klassischen Planern besteht der wesentliche Vorteil von HTN-Planern in der differenzierten Wissensrepräsentation und den Schlussfolgerungsfähigkeiten. HTN-Planer können eine Vielzahl von nichtklassischen Planungsproblemen repräsentieren und lösen [40]. Mit einer gut gewählten Menge von Hierarchischen Aufgabennetzwerken zu ihrer Lenkung können sie klassische Planungsprobleme um Größenordnungen schneller

lösen als klassische und neoklassische Planer. Der wesentliche Nachteil von HTN-Planern besteht darin, dass der Domänenautor nicht nur eine Menge von Planungsoperatoren, sondern auch eine Menge von Methoden spezifizieren muss.

**HTN-Planungssysteme** Eine Übersicht über verschiedene HTN-Planer findet sich zum Beispiel in [58]. Dort werden die wichtigsten Eigenschaften der domänenunabhängigen Planungssysteme *Nonlin*<sup>6</sup>, *O-Plan*<sup>7</sup>, *SIPE-2*<sup>8</sup> und *SHOP2*<sup>9</sup> vorgestellt. Darüber hinaus werden die HTN-Planer anhand der in [72] genannten Charakteristiken miteinander verglichen.

### 2.3.2. Flexible Programme

In [60] und [61] werden so genannte *Flexible Programme* zur hierarchischen Repräsentation von Handlungswissen für einen Serviceroboter vorgestellt. Flexible Programme werden ähnlich zu TDL (siehe Abschnitt 2.1.2) als Hierarchische Aufgabennetzwerke dargestellt. Eine Handlung wird dabei in Form einer Baumstruktur aus Knoten repräsentiert. Die Knoten beinhalten entweder weitere Nachfolgeknoten (auch *Kinder* genannt) oder atomare Handlungen. Eine Gesamthandlung setzt sich somit aus den atomaren Handlungen und der Struktur des Baumes zusammen. Die Wurzel des Baumes bildet die oberste Ebene der Handlung und die Gesamthandlung ist entlang einer Tiefensuche von links nach rechts innerhalb des Baumes definiert. Ein Flexibles Programm wird somit vollständig durch die im Baum beinhalteten Knoten beschrieben. Formal wird ein Knoten durch ein 8-Tupel beschrieben:

$$\mathcal{P} = (Id, C_{pre}, C_{post}, C_{rt}, R, S, P, A) \quad [2.5]$$

Dabei ist *Id* ein Bezeichner, der die jeweilige Handlung eindeutig identifiziert.  $C_{pre}$ ,  $C_{post}$  und  $C_{rt}$  bezeichnen die Vor-, Nach- und Laufzeitbedingungen (Währendbedingungen), die für den jeweiligen Knoten gelten müssen. *R* (engl. *Rating*) bezeichnet ein kontinuierliches Bewertungsmaß, das zur Auswahl zwischen alternativen Knoten verwendet werden kann. *S* (engl. *Success*) stellt ein Bewertungsmaß für den Erfolg der Ausführung eines Knotens dar. *P* (engl. *Prospect*) bezeichnet den Kindprospekt eines Knotens und gibt die Liste und Reihenfolge der möglichen Kindknoten an, aus denen zur Laufzeit die geeignetsten ausgewählt werden. Die Aktion *A* gibt die auszuführende Aktion einer atomaren Handlung an. Per Definition ist bei einem Knoten entweder der Prospekt oder die Aktion leer. Knoten mit Aktionen bilden somit die Blattknoten des Baumes, wohingegen Knoten mit Prospekt die inneren Knoten des Baumes repräsentieren.

Der Prospekt eines Knotens beinhaltet alle möglichen Konfigurationen von Kindknoten. Er

<sup>6</sup>Nonlin: <http://www.aiai.ed.ac.uk/project/nonlin/>

<sup>7</sup>O-Plan: <http://www.aiai.ed.ac.uk/oplan/>

<sup>8</sup>SIPE-2: <http://www.ai.sri.com/~sipe/>

<sup>9</sup>SHOP2: <http://www.cs.umd.edu/projects/shop/>

ist aus einer Menge von  $n > 0$  so genannten *Sitzen* aufgebaut, für die jeweils  $m > 0$  Kandidaten existieren. Die Sitze eines Knotens können somit als Teilhandlungen aufgefasst werden, für die jeweils mindestens ein Kandidat existiert. Existieren für einen Sitz mehrere Kandidaten, so ist zur Laufzeit diejenige Teilhandlung auszuwählen, welche für die gegebene Situation am besten geeignet ist. Dies erfolgt durch Überprüfung der Vorbedingungen und die Berechnung der Bewertungsmaße  $R$  der einzelnen Kandidaten. Benachbarte Sitze können zudem in  $p \leq n$  parallelen Gruppen angeordnet werden. Auf diese Weise können Teilhandlungen, die konfliktfrei nebenläufig ausgeführt werden können, parallel angeordnet werden. Darüber hinaus können Teilhandlungen als asynchron markiert werden. Asynchrone Teilhandlungen werden bei der Verarbeitung des Handlungsbaumes ebenfalls betreten, ihre Fertigstellung wird jedoch nicht abgewartet. Da auf jeder Ebene nur die direkten Kindknoten bekannt sind, können die resultierenden Bäume eine beliebige Tiefe besitzen. Durch rekursive Verwendung desselben Knotens kann ein Baum mit unendlicher Tiefe erzeugt werden.

Indem jeder Knoten eine eindeutige Handlung beschreibt und die dafür benötigten Teilhandlungen definiert, wird das Handlungswissen in Form eines Baumes rekursiv aufgebaut. Ein Flexibles Programm beschreibt somit alle möglichen Handlungsstränge, die bei der Ausführung verfolgt werden können. Umgekehrt beschreibt auch jeder Knoten immer ein gültiges Flexibles Programm. Abhängigkeiten zwischen den einzelnen Teilprogrammen entstehen zum einen durch die Art der Verknüpfung und zum anderen durch die Erzeugung und Verwendung von so genannten *Merkmalen*. In [60] wird als Beispiel die Verknüpfung einer sensorischen Aktion (zum Beispiel *Tisch lokalisieren*) und einer ausführenden Aktion (zum Beispiel *fahre zu Tisch*) über das Merkmal der Tischposition angegeben. Diese Merkmale werden jedoch nicht direkt zwischen den einzelnen Knoten ausgetauscht, sondern mit Hilfe eines lokalen Datenspeichers, welcher Teil des verwendeten Umweltmodells ist (siehe [60, Abschnitt 4.3.1]). Die Überprüfung der Vor-, Nach- und Währendbedingungen erfolgt ebenfalls anhand des Umweltmodells. Zur Formulierung der Vor-, Nach- und Währendbedingungen wird eine funktionsfreie Untermenge der Prädikatenlogik erster Stufe verwendet (siehe [60, Abschnitt 4.3.2]).

Die Ausführung der Flexiblen Programme erfolgt in einer verteilten, hybriden, ereignisbasierten Architektur (siehe [60] und [62]). Die Flexiblen Programme werden dabei mit Hilfe von zwei Schichten verarbeitet. Die so genannte *Baumverarbeitung* (der Kern) operiert auf dem Handlungsbaum und bestimmt die jeweilige Handlungsfolge entsprechend der Handlungsmodellierung mit sequentiellen und parallelen Teilhandlungen. Die auf diese Weise erzeugte Handlungsfolge wird an die so genannte *Kommunikationsschicht* (die Hülle) übergeben, welche diese zur Ausführung an die entsprechenden Komponenten übermittelt. Weiterhin ist die Kommunikationsschicht auch für das Empfangen der Antworten von den Komponenten und ihre Weitergabe an den Kern zuständig. Die einzelnen Knoten der Flexiblen Programme können während der Verarbeitung die Zustände *virtuell*, *instantiiert* oder *abgeschlossen* einnehmen.



Der Zustand *virtuell* bezeichnet dabei unbesuchte Knoten, deren Prospekt noch nicht ausgewertet wurde. Knoten, deren Prospekt ausgewertet wurde und für deren Sitze entsprechende Kandidaten ausgewählt wurden, nehmen den Zustand *instantiiert* ein. Der Zustand *abgeschlossen* kennzeichnet Knoten, die vollständig abgearbeitet sind. Die Verarbeitung des Baumes wird mit jedem Ereignis  $e$  angestoßen, das die Kommunikationsschicht von den ausführenden Komponenten empfängt. Die Bestimmung der als nächstes auszuführenden Blätter erfolgt in zwei Schritten:

- Als erstes wird ausgehend von dem Blatt, das  $e$  ausgelöst hat und damit von  $e$  als abgeschlossen bestätigt wird, im Baum aufwärts nach der letzten nicht abgeschlossenen Teilhandlung gesucht. Dieser Prozess wird als *Auftauchen* bezeichnet. Dabei werden alle Teilbäume, deren Kinder abgeschlossen sind, ebenfalls als abgeschlossen gekennzeichnet. Das Auftauchen wird beendet, sobald eine Ebene im Baum erreicht wird, deren Teilhandlungen nicht alle als abgeschlossen markiert sind. Das Auftauchen terminiert ebenfalls, wenn die Wurzel des Baumes erreicht wird und somit kein weiteres Auftauchen mehr möglich ist. In diesem Fall ist die Gesamthandlung abgeschlossen.
- Im zweiten Schritt werden ausgehend von der während des Auftauchens gefundenen Ebene im Baum abwärts alle Elementaraktionen gesucht, die zum jeweiligen Zeitpunkt ausführbar sind. Dieser Prozess wird als *Abtauchen* bezeichnet. Dabei werden gegebenenfalls Knoten instantiiert und es wird unter den verschiedenen Kandidaten ausgewählt. Die auszuführenden Blätter werden zusammengefasst an die Hülle übergeben.

### 2.3.3. Nebenläufige Hierarchische Pläne

Abstraktion kann bei der Lösung von umfangreichen Planungs- und Schedulingproblemen ein nützliches Werkzeug sein. Durch die Abstraktion weniger kritischer Details bei der Betrachtung eines umfangreichen Problems, kann ein Agent eine Gesamtlösung für das Problem einfacher finden. Steht der Rahmen der Gesamtlösung, kann der Agent zusätzliche Details in die Lösung einarbeiten. Ist es möglich, Abhängigkeiten bereits auf abstrakten Ebenen vollständig aufzulösen, so können ein oder mehrere Agenten Teile der Gesamtlösung unabhängig voneinander in vollem Detaillierungsgrad ausgestalten (sogar parallel). Es ist jedoch nicht offensichtlich, wie umfangreiche und komplexe Planungsprobleme am geeignetsten abstrahiert werden können, um die genannten effizienten Verbesserungen zu ermöglichen. Werden bei der Abstraktion die falschen Details ignoriert, so kann dies dazu führen, dass Pläne erstellt werden, die nicht funktionieren, beziehungsweise dass kostenintensives Backtracking oder Neuplanen erforderlich wird, sobald übersehene Abhängigkeiten zu Tage treten.

In [28] wird eine auf so genannten *Nebenläufigen Hierarchischen Plänen* basierende Strategie vorgeschlagen, welche die Vorteile und die Risiken der Abstraktion für umfangreiche

Einzelagenten- und Multiagenten-Planungsprobleme ausbalanciert. Die wesentliche Idee der Strategie besteht darin, jeden abstrakten Operator in der Planhierarchie mit Zusammenfassungsinformationen über alle seine potentiellen Anforderungen und Effekte unter allen möglichen Verfeinerungen zu annotieren. Weil alle möglicherweise relevanten Bedingungen und Effekte modelliert werden, kann sichergestellt werden, dass der Agent, der Schlussfolgerungen mit den abstrakten Operatoren anstellt, keine wichtigen Details übersieht. Da die Zusammenfassungsinformationen Details darüber abstrahieren, unter welchen Verfeinerungsentscheidungen Bedingungen und Effekte zutage treten, sowie Informationen über den relativen Zeitablauf abstrahieren, wann welche Bedingungen benötigt und welche Effekte erzielt werden, resultieren sie sogar oft in einer exponentiellen Informationsreduktion verglichen mit flachen Repräsentationen.

Ein Nebenläufiger Hierarchischer Plan (engl. *Concurrent Hierarchical Plan (CHiP)*)  $p$  ist ein Tupel  $(pre, in, post, usage, type, subplans, order)$ .  $pre(p)$ ,  $in(p)$  und  $post(p)$  sind Mengen von Literalen ( $v$  oder  $\neg v$  für eine aussagenlogische Variable  $v$ ), welche die für einen Plan definierten Vor-, Während- und Nachbedingungen repräsentieren. Währendbedingungen ( $in(p)$ ) wirken sich auf den Zustand direkt nach dem Startzeitpunkt ( $t_s(p)$ ) von  $p$  aus (bzw. sichern eine Bedingung für diesen zu) und müssen während der Dauer von  $p$  gültig sein. Vorbedingungen ( $pre(p)$ ) müssen zum Startzeitpunkt gültig sein und Nachbedingungen ( $post(p)$ ) werden zum Endzeitpunkt ( $t_f(p)$ ) von  $p$  zugesichert. Der Ressourcenverbrauch  $usage(p, res)$  gibt die Menge der Ressource  $res$  an, die von  $p$  verwendet wird. Eine metrische Ressource  $res$  ist ein Tupel  $(min\_value, max\_value, type)$ . Die Minimal- und Maximalwerte ( $min\_value$  und  $max\_value$ ) können ganzzahlig oder reellwertig sein und repräsentieren die Grenzen der Kapazität bzw. der verfügbaren Menge. Der Typ der Ressource ( $type$ ) gibt an, ob die Ressource verbrauchbar (*consumable*) oder unverbrauchbar (*non-consumable*) ist. Der Ressourcenverbrauch ist instantan zum Startzeitpunkt und wird, wenn es sich um eine unverbrauchbare Ressource handelt, instantan zum Endzeitpunkt rückgängig gemacht. Der Typ ( $type(p)$ ) eines Plans kann die Werte *primitive*, *and* oder *or* annehmen. Ein *and*-Plan ist ein nichtelementarer Plan, der durch Ausführung aller seiner Unterpläne fertiggestellt wird. Ein *or*-Plan ist ein nichtelementarer Plan, der durch Ausführung genau eines Unterplans fertiggestellt wird.  $subplans(p)$  ist eine Menge von Unterplänen. Für einen elementaren Plan ist diese Menge leer.  $order$  ist eine konsistente Menge von zeitlichen Relationen für Paare von Unterplänen, welche nur für *and*-Pläne definiert ist. Pläne, für die keine gegenseitigen Ordnungen angegeben sind, werden als potentiell nebenläufig ausführbar angenommen.

Die Zerlegung eines CHiPs erfolgt auf die gleiche Art und Weise wie die eines HTNs. Ein *and*-Plan ist ein Aufgabennetzwerk und ein *or*-Plan ist ein zusätzliches Konstrukt, welches die Menge aller Methoden zur Erreichung derselben nichtelementaren Aufgabe repräsentiert. Ein Netzwerk von Aufgaben korrespondiert zu den Unterplänen eines Plans. HTN-Planer, welche

abstrakte Bedingungen verwenden um die Suche zu leiten, basieren in der Regel auf einer benutzerdefinierten Untermenge von Bedingungen, die lediglich helfen kann, potentielle Konflikte zu erkennen. Im Gegensatz dazu können Zusammenfassungsinformationen verwendet werden, um abstrakte Lösungen zu finden, die garantiert erfolgreich sind, unabhängig davon, wie sie verfeinert werden, da die Informationen alle potentiellen Bedingungen der zugrunde liegenden Zerlegung beschreiben. Somit können Festlegungen für bestimmte Planmöglichkeiten basierend auf den Zusammenfassungsinformationen getroffen werden, ohne befürchten zu müssen, dass tiefer liegende Details diese Festlegungen zum Scheitern verurteilen.

Die Zusammenfassungsinformation für einen Plan  $p$  ist ein Tupel  $(pre_{sum}, in_{sum}, post_{sum}, usage_{sum}, consistent)$ , das aus Mengen von zusammengefassten Bedingungen, dem zusammengefassten Ressourcenverbrauch und einem *consistent*-Flag besteht, welches angibt, ob der Plan intern konsistent ausgeführt werden kann.  $pre_{sum}(p)$  und  $post_{sum}(p)$  sind zusammengefasste Vor- und Nachbedingungen, welche den externen Vor- und Nachbedingungen von  $p$  entsprechen. Die zusammengefassten Währendbedingungen  $in_{sum}(p)$  enthalten alle Bedingungen, die während einer Ausführung von  $p$  gelten müssen, damit diese erfolgreich ist. Eine Bedingung  $c$  in einer dieser Mengen ist ein Tupel  $(l, existence, timing)$ .  $l(c)$  ist das Literal der Bedingung  $c$ . Die Existenz von  $c$  ( $existence(c)$ ) kann die Werte *must* oder *may* annehmen. Gilt  $existence(c) = must$ , dann ist  $c$  eine so genannte *must*-Bedingung und  $l$  muss für jede erfolgreiche Planausführung gelten.  $c$  ist eine so genannte *may*-Bedingung, wenn  $l(c)$  für irgendeine erfolgreiche Ausführung gelten muss. Das Zeitverhalten  $timing$  einer Zusammenfassungsbedingung  $c$  kann die Werte *always*, *sometimes*, *first* oder *last* annehmen.  $timing(c)$  nimmt den Wert *always* für  $c \in in_{sum}$  an, wenn  $l(c)$  eine Währendbedingung ist, die während aller möglichen Ausführungen von  $p$  gelten muss. Anderenfalls bedeutet  $timing(c) = sometimes$ , dass  $l(c)$  zumindest für einen Zeitpunkt während einer Ausführung von  $p$  gelten muss. Für  $c \in pre_{sum}$  nimmt  $timing$  den Wert *first* an, wenn  $l(c)$  zu Beginn der Ausführung von  $p$  gilt, und sonst den Wert *sometimes*. Für  $c \in post_{sum}$  nimmt  $timing$  den Wert *last* an, wenn  $l(c)$  am Ende einer erfolgreichen Ausführung von  $p$  zugesichert wird, und andernfalls den Wert *sometimes*. Der zusammengefasste Ressourcenverbrauch  $usage_{sum}$  ist ein Tupel  $(local\_min, local\_max, persist)$ , wobei der lokale Ressourcenverbrauch ( $local\_min$  und  $local\_max$ ) innerhalb der Ausführung der Aufgabe auftritt und der persistente Ressourcenverbrauch ( $persist$ ) den fortdauernden Verbrauch einer verbrauchbaren Ressource nach der Beendigung der Aufgabe repräsentiert.

Die Zusammenfassungsinformationen eines Plans sind rekursiv definiert, indem sie auf den Zusammenfassungsinformationen der direkten Unterpläne des Plans statt auf der kompletten Zerlegung basieren. Letzteres würde bedeuten, dass die Berechnung der Zusammenfassungsinformationen genauso aufwändig wie die Lösung des Planungsproblems wäre. Einer der wesentliche Zwecke der Zusammenfassungsinformationen ist es jedoch, die Berechnungen des Planungsproblems zu reduzieren.

Der Algorithmus zur Berechnung der zusammengefassten Bedingungen für einen Plan  $p$  erhält als Eingabe die zusammengefassten Bedingungen der direkten Unterpläne von  $p$  und die für  $p$  definierten Bedingungen. Die Vor-, Während- und Nachbedingungen von  $p$  werden *must first*, *must always* und *must last* Zusammenfassungsbedingungen. Der Algorithmus behält die Existenz- und Zeitverhaltensangaben der Zusammenfassungsbedingungen von Unterplänen für den Elternplan in Abhängigkeit davon bei, ob die Bedingungen von Geschwisterplänen erreicht (*achieve*), überschrieben (*clobber*) oder rückgängig gemacht (*undo*) werden (siehe Abbildung 2.14), ob es sich um eine *and*- oder *or*-Zerlegung handelt, ob der Unterplan als erster oder als letzter angeordnet ist, und ob allen Unterplänen die gleiche Bedingung gemeinsam ist. Die *first*-, *always*- und *last*-Bedingungen von Unterplänen können *sometimes*-Bedingungen im Elternplan werden.

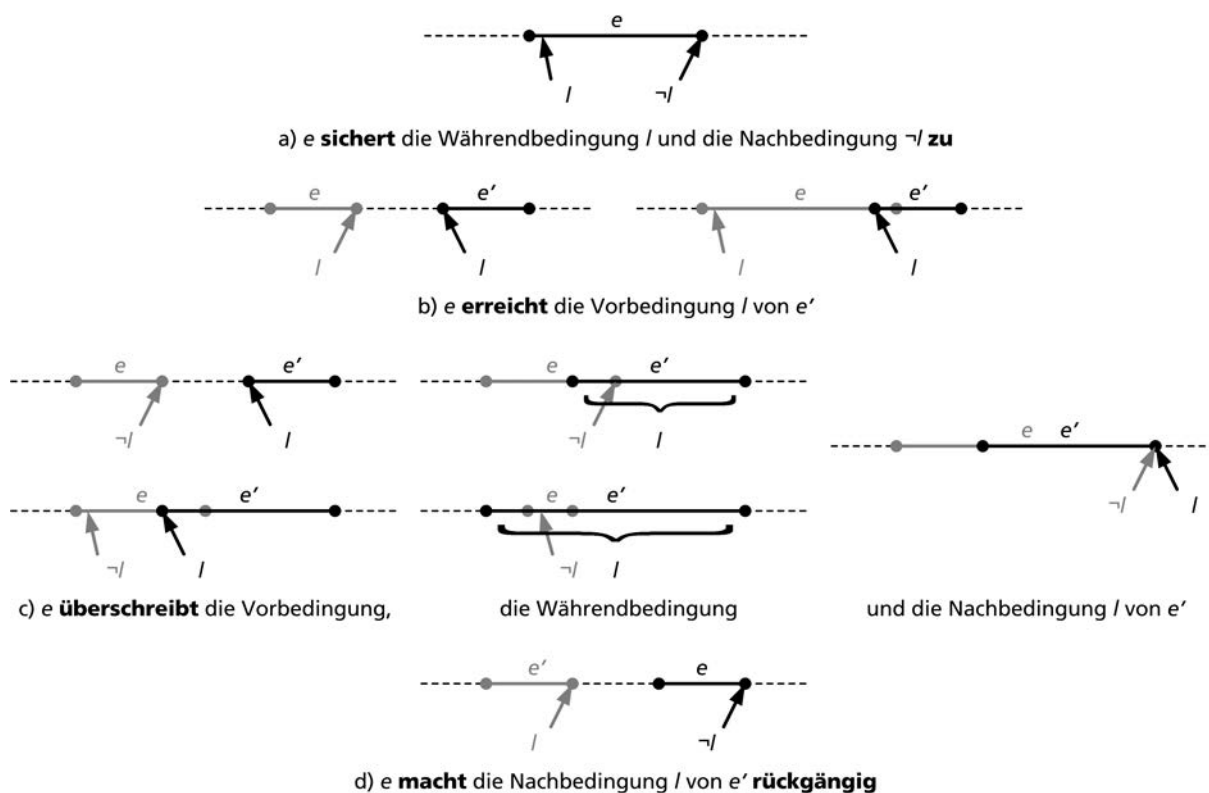


Abb. 2.14.: Intervall-Interaktionen zwischen Planschritten [28].

Der Algorithmus zur Bestimmung des zusammengefassten Ressourcenverbrauchs für eine abstrakte Aufgabe erhält die zusammengefassten Ressourcenverbräuche ihrer Unteraufgaben, berücksichtigt alle zulässigen Ordnungen der Unteraufgaben und alle mögliche Ressourcenverbräuche für alle Unterintervalle innerhalb des Intervalls der abstrakten Aufgabe, um mehrere Verbrauchsprofile zu generieren. Diese Profile werden mit Algorithmen zur Berechnung von parallelen, sequentiellen und disjunktiven Verbräuchen zum zusammengefassten Ressourcenverbrauch der Elternaufgabe kombiniert.

Der Elternplan wird als *consistent* berechnet, solange alle Unterpläne konsistent sind, kein Unterplan möglicherweise die Zusammenfassungsbedingungen eines anderen überschreibt und der zusammengefasste Ressourcenverbrauch nicht die Kapazitätsgrenzen der Ressourcen verletzt.

In [28] wird ein auf Zusammenfassungsinformationen beruhender Koordinationsalgorithmus vorgeschlagen, der nach Wegen sucht, die Zerlegung und Ordnung der kollektiven Aktionen des Agenten bzw. der Agenten zu beschränken, um Konflikte unter Maximierung der Nützlichkeit der individuellen Agenten oder der globalen Nützlichkeit der Gruppe aufzulösen. Die Suche beginnt mit den Plänen der Agenten auf oberster Ebene. Eine Lösung ist dadurch gegeben, dass es keine möglichen Konflikte zwischen den Plänen der Agenten gibt. Der Algorithmus versucht, eine Lösung auf dieser obersten Ebene zu finden und expandiert dann die Hierarchie tiefer und tiefer, bis eine optimale Lösung gefunden wird oder der Suchraum erschöpft ist.

Neben dem Finden von konfliktfreien oder koordinierten Plänen auf abstrakten Ebenen können Zusammenfassungsinformation auch beim Leiten der Suche mit Hilfe einer Ranking-Funktion wertvoll sein. Eine Strategie besteht zum Beispiel darin, zuerst die Pläne, die in die meisten Bedrohungen einbezogen sind, zu expandieren (engl. *expand on most threats first* (EMTF)). Diese Heuristik eignet sich insbesondere für die Bestimmung der Reihenfolge von zu expandierenden *and*-Unterplänen. Für die Reihenfolge der Auswahl von *or*-Unterplänen bietet es sich hingegen an, zunächst die Pläne, die in die wenigsten Bedrohungen einbezogen sind, auszuwählen (engl. *choose fewest threats first* (CFTF)).

## 2.4. Situationsbewertung und Entscheidungsfindung

In diesem Abschnitt wird auf verschiedene für diese Arbeit relevante Ansätze zur Situationsbewertung und Entscheidungsfindung eingegangen. Im Fokus stehen dabei im Wesentlichen probabilistische Verfahren. Nachfolgend wird daher insbesondere ein kurzer Überblick über zwei der zentralen Formalismen in diesem Bereich gegeben: Bayes'sche Netzwerke und Markow'sche Entscheidungsprozesse. Daran anknüpfend wird exemplarisch ein Ansatz zur Entscheidungsfindung für einen realen, hochgradig multimodalen Serviceroboter vorgestellt.

### 2.4.1. Bayes'sche Netzwerke

Ein Bayes'sches Netzwerk repräsentiert die gemeinsame Wahrscheinlichkeitsverteilung einer Menge von Zufallsvariablen (siehe [67]). Jede Zufallsvariable wird dabei durch einen Knoten im Bayes'schen Netzwerk repräsentiert. Für jede Zufallsvariable werden zwei Arten von Informationen spezifiziert. Erstens repräsentieren die Kanten im Netzwerk die Zusicherung, dass ei-

ne Zufallsvariable *bedingt unabhängig*<sup>10</sup> von ihren Nichtnachkommen<sup>11</sup> gegeben ihre direkten Vorgänger im Netzwerk ist. Zweitens wird für jede Zufallsvariable eine Tabelle mit bedingten Wahrscheinlichkeiten angegeben, welche die Wahrscheinlichkeitsverteilung für diese Zufallsvariable gegeben ihre direkten Vorgänger beschreibt. Die gemeinsame Wahrscheinlichkeit für jede beliebige Zuweisung von Werten  $(y_1, \dots, y_n)$  zum Tupel der Zufallsvariablen  $(Y_1, \dots, Y_n)$  kann mit Hilfe der folgenden Formel berechnet werden:

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Vorgänger}(Y_i)) \quad [2.6]$$

Dabei wird mit  $\text{Vorgänger}(Y_i)$  die Menge der direkten Vorgänger von  $Y_i$  im Netzwerk bezeichnet. Die Werte von  $P(y_i | \text{Vorgänger}(Y_i))$  entsprechen den in der zum Knoten  $Y_i$  gehörigen Tabelle mit bedingten Wahrscheinlichkeiten angegebenen Werten. Die Menge der lokalen Tabellen mit bedingten Wahrscheinlichkeiten für alle Zufallsvariablen zusammen mit der durch den azyklischen, gerichteten Graphen des Netzwerks beschriebenen Menge von Annahmen der bedingten Unabhängigkeit beschreiben somit die vollständige gemeinsame Wahrscheinlichkeitsverteilung für das Netzwerk.

Ein Bayes'sches Netzwerk kann dazu verwendet werden, die Wahrscheinlichkeitsverteilung für jede Untermenge von Zufallsvariablen des Netzwerks gegeben die Werte oder Wahrscheinlichkeitsverteilungen für jede Untermenge der verbleibenden Zufallsvariablen zu bestimmen. Im Allgemeinen ist die exakte Inferenz von Wahrscheinlichkeiten für ein beliebiges Bayes'sches Netzwerk jedoch NP-hart. Es existieren dennoch eine Vielzahl von verschiedenen Methoden für die probabilistische Inferenz in Bayes'schen Netzwerken. Dies umfasst sowohl exakte Verfahren als auch approximative Verfahren, welche im Gegenzug für mehr Effizienz einen gewissen Grad an Genauigkeit opfern. Ein Beispiel hierfür sind etwa die so genannten Monte Carlo Verfahren, die approximative Lösungen durch zufälliges Sampling der Verteilungen der unbeobachteten Zufallsvariablen bestimmen. Eine weitergehende Diskussion der unterschiedlichen Inferenzmethoden für Bayes'sche Netzwerke bieten zum Beispiel [86] oder [13].

So genannte *Entscheidungsnetzwerke* (engl. *decision network* oder auch *influence diagram*) kombinieren Bayes'sche Netzwerke mit zusätzlichen Knotentypen für Aktionen und Nützlichkeiten (siehe [86]). In seiner allgemeinsten Form repräsentiert ein Entscheidungsnetzwerk Informationen über den aktuellen Zustand des Agenten, seine möglichen Aktionen, den aus der Aktion des Agenten resultierenden Zustand sowie die Nützlichkeit dieses Zustands. Ein Entscheidungsnetzwerk enthält somit drei Arten von Knoten:

**Zufallsknoten** repräsentieren Zufallsvariablen wie in normalen Bayes'schen Netzwerken. Je-

<sup>10</sup>Seien  $X, Y$  und  $Z$  drei diskretwertige Zufallsvariablen. Dann ist  $X$  *bedingt unabhängig* von  $Y$  gegeben  $Z$ , wenn die  $X$  zugrunde liegende Wahrscheinlichkeitsverteilung unabhängig vom Wert von  $Y$  gegeben den Wert von  $Z$  ist:  $(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$ .

<sup>11</sup> $X$  ist ein *Nachkomme* von  $Y$ , wenn ein gerichteter Pfad von  $Y$  nach  $X$  existiert.

dem Knoten ist eine Tabelle mit bedingten Wahrscheinlichkeiten zugeordnet, die mit dem Zustand der Vorgängerknoten indiziert wird. In Entscheidungsnetzwerken können die Vorgänger sowohl andere Zufallsknoten als auch Entscheidungsknoten umfassen.

**Entscheidungsknoten** repräsentieren Punkte, an denen der Entscheidungsträger die Wahl zwischen verschiedenen Aktionen hat. Im einfachsten Fall besitzt ein Entscheidungsnetzwerk nur genau einen Entscheidungsknoten.

**Nützlichkeitsknoten** repräsentieren die Nützlichkeitsfunktion des Agenten. Die Vorgänger von Nützlichkeitsknoten beinhalten alle Variablen, welche das Ergebnis der Aktion beschreiben und direkten Einfluss auf die Nützlichkeit haben. Jedem Nützlichkeitsknoten ist eine Beschreibung der Nützlichkeit in Form einer Funktion der Attribute der Vorgängerknoten zugeordnet.

Die Auswahl von Aktionen erfolgt durch Evaluierung des Entscheidungsnetzwerks für alle möglichen Werte des Entscheidungsknotens. Sobald der Entscheidungsknoten gesetzt ist, verhält er sich genau wie eine Zufallsvariable, die als Evidenzvariable gesetzt wurde. Der Algorithmus zur Evaluierung von Entscheidungsnetzwerken lässt sich somit wie folgt angeben:

1. Setze die Evidenzvariablen für den aktuellen Zustand.
2. Für jeden möglichen Wert des Entscheidungsknotens verfare wie folgt:
  - Setze den Entscheidungsknoten auf diesen Wert.
  - Berechne die a posteriori Wahrscheinlichkeiten für die Vorgängerknoten des Nützlichkeitsknotens mit Hilfe eines Standardverfahrens der probabilistischen Inferenz.
  - Berechne die resultierende Nützlichkeit für die Aktion.
3. Gebe die Aktion mit der höchsten Nützlichkeit zurück.

### **2.4.2. Markow'sche Entscheidungsprozesse**

Im Folgenden wird zunächst die formale Definition von Markow'schen Entscheidungsprozessen erläutert und es werden zwei der grundlegenden Verfahren zur Bestimmung von Lösungen für Markow'schen Entscheidungsprozesse umrissen. Darauf aufbauend wird im zweiten Teil dieses Abschnitts ein Ansatz vorgestellt, der Markow'sche Entscheidungsprozesse zur Entscheidungsfindung für einen realen, hochgradig multimodalen Serviceroboter einsetzt.

#### **Formale Definition und Verfahren zur Bestimmung optimaler Strategien**

Ein Markow'scher Entscheidungsprozess (engl. *Markov decision process* (MDP)) ist die Spezifikation eines sequentiellen Entscheidungsproblems für eine vollständig beobachtbare Umgebung mit einem Markow'schen Transitionsmodell und additiven Belohnungen. Formal ist

ein Markow'scher Entscheidungsprozess durch die folgenden drei Komponenten definiert: den initialen Zustand  $s_0$ , das Transitionsmodell  $T(s, a, s')$  und die Belohnungsfunktion  $R(s)$ . Dabei gilt für das Transitionsmodell die so genannte Markow-Annahme. Dies bedeutet, dass die Wahrscheinlichkeit einen Zustand zu erreichen nur von seinem direkten Vorgängerzustand und nicht von der gesamten Historie der Vorgängerzustände abhängig ist.

Eine Lösung für einen Markow'schen Entscheidungsprozess wird als Strategie  $\pi$  (engl. *policy*) bezeichnet und gibt für jeden Zustand, den ein Agent erreichen kann, an, was er tun soll. Jedes Mal, wenn eine Strategie ausgehend vom initialen Zustand ausgeführt wird, ergibt sich aufgrund des nichtdeterministischen Charakters der Umgebung eine andere Folge von Zuständen. Die Qualität einer Strategie wird daher an der erwarteten Nützlichkeit der möglichen Zustandsfolgen bemessen, die von der Strategie erzeugt werden. Eine optimale Strategie, im Folgenden als  $\pi^*$  bezeichnet, ist somit eine Strategie, welche die höchste erwartete Nützlichkeit erzielt. Die Nützlichkeit einer Zustandsfolge ist dabei wie folgt definiert:

$$U_h([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots \quad [2.7]$$

Für eine optimale Strategie  $\pi^*$  gilt somit:

$$\pi^* = \arg \max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right] \quad [2.8]$$

In [86] werden zwei Verfahren für das Finden von optimalen Strategien vorgestellt: das so genannte *Werte-Iterationsverfahren* (engl. *value iteration*) und das so genannte *Strategie-Iterationsverfahren* (engl. *policy iteration*). Beide werden im Folgenden kurz erläutert.

**Werte-Iterationsverfahren** Die zentrale Idee des Werte-Iterationsverfahrens besteht darin, die Nützlichkeit jedes einzelnen Zustands zu berechnen und die Nützlichkeiten der Zustände anschließend zur Auswahl der optimalen Aktion in jedem Zustand zu verwenden. Die Nützlichkeit von Zuständen lässt sich dabei unter Bezugnahme auf die Nützlichkeit von Zustandsfolgen definieren. Da die Zustandsfolge ihrerseits von der ausgeführten Strategie abhängig ist, lässt sich zunächst die Nützlichkeit eines Zustands  $U^\pi(s)$  bezüglich einer Strategie  $\pi$  wie folgt definieren:

$$U^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s \right] \quad [2.9]$$

Die wahre Nützlichkeit eines Zustands, im Folgenden mit  $U(s)$  bezeichnet, ist somit einfach  $U^{\pi^*}(s)$ , d. h. die erwartete Summe der diskontierten Belohnungen, wenn der Agent eine optimale Strategie ausführt. Die Nützlichkeitsfunktion  $U(s)$  erlaubt es dem Agenten, die Aktion



auszuwählen, welche die erwartete Nützlichkeit des nachfolgenden Zustands maximiert:

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') U(s') \quad [2.10]$$

Es besteht somit ein direkter Zusammenhang zwischen der Nützlichkeit eines Zustands und den Nützlichkeiten seiner Nachbarn: Die Nützlichkeit eines Zustands ergibt sich aus der direkten Belohnung für diesen Zustand plus der erwarteten diskontierten Nützlichkeit des nächsten Zustands unter der Annahme, dass der Agent die optimale Aktion wählt. Die Nützlichkeit eines Zustands ist somit durch die so genannte *Bellmann-Gleichung* gegeben:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s') \quad [2.11]$$

Die Bellmann-Gleichung bildet die Grundlage zur Lösung von Markow'schen Entscheidungsprozessen. Für  $n$  mögliche Zustände gibt es  $n$  Bellmann-Gleichungen, eine für jeden Zustand. Die  $n$  Gleichungen enthalten  $n$  Unbekannte, nämlich die Nützlichkeiten der Zustände. Aufgrund des Maximums-Operators sind diese Gleichungen jedoch nichtlinear, so dass sich das Gleichungssystem nicht einfach mit Verfahren der linearen Algebra lösen lässt. Eine Lösung für dieses Problem besteht in einem iterativen Ansatz mit Hilfe der so genannten *Bellman-Aktualisierung*:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s') \quad [2.12]$$

Dabei wird von beliebigen initialen Werten für die Nützlichkeiten ausgegangen. Es lässt sich zeigen (siehe [86]), dass sich bei unendlicher Anwendung der Aktualisierung garantiert ein Gleichgewicht einstellt, die finalen Nützlichkeitswerte eine eindeutige Lösung der Bellmann-Gleichungen bilden und die zugehörige Strategie (siehe Gleichung 2.10) somit optimal ist.

**Strategie-Iterationsverfahren** Es kann gezeigt werden (siehe [86]), dass es möglich ist, eine optimale Strategie auch mit einer ungenauen Schätzung der Nützlichkeitsfunktion zu bestimmen. Wenn eine Aktion im Vergleich zu den anderen deutlich besser ist, muss die exakte Größe der Nützlichkeiten der beteiligten Zustände nicht genau bekannt sein. Diese Einsicht führt zu einem weiteren Verfahren zur Lösung von Markow'schen Entscheidungsprozessen, dem so genannten Strategie-Iterationsverfahren. Es beginnt mit einer beliebigen initialen Strategie  $\pi_i$  und führt alternierend die beiden folgenden Schritte aus:

**Evaluierung der Strategie** Gegeben eine Strategie  $\pi_i$ , berechne die Nützlichkeit  $U_i = U^{\pi_i}$  für jeden Zustand unter der Voraussetzung, dass  $\pi_i$  ausgeführt wird.

**Verbesserung der Strategie** Berechne eine neue Strategie  $\pi_{i+1}$  mit maximaler erwarteter Nützlichkeit unter Verwendung einer auf  $U_i$  basierenden Vorausschau von einem Schritt

(wie in Gleichung 2.10).

Der Algorithmus terminiert, sobald die Verbesserung der Strategie keine Veränderung der Nützlichkeiten mehr bewirkt. Da die Nützlichkeitsfunktion  $U_i$  einen Fixpunkt der Bellmann-Aktualisierung darstellt, ist sie somit eine Lösung der Bellmann-Gleichungen und  $\pi_i$  eine optimale Strategie. Für einen endlichen Zustandsraum existiert darüber hinaus nur eine endliche Anzahl von Strategien und da jede Iteration eine bessere Strategie erzeugt, ist die Terminierung des Strategie-Iterationsverfahrens garantiert. Der Schritt zur Verbesserung der Strategie lässt sich einfach berechnen. Die Evaluierung der Strategie lässt sich mittels der Gleichung

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s') \quad [2.13]$$

durchführen, da die Aktion in jedem Zustand durch die Strategie festgelegt ist. Es handelt sich somit um eine vereinfachte Version der Bellmann-Gleichung 2.12. Da der max-Operator herausgefallen ist, sind diese Gleichungen linear und können daher mit Hilfe von Standardverfahren der linearen Algebra in  $O(n^3)$  gelöst werden. Dies kann für große Zustandsräume trotzdem prohibitiv sein. Es ist jedoch glücklicherweise nicht erforderlich, eine exakte Evaluierung der Strategie durchzuführen (siehe [86]). Stattdessen kann eine gewisse Anzahl von vereinfachten Werte-Iterationen durchgeführt werden. Die vereinfachte Bellmann-Aktualisierung lautet wie folgt

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s') \quad [2.14]$$

und wird  $k$ -Mal zur Schätzung der nächsten Nützlichkeitsfunktion ausgeführt. Der resultierende Algorithmus wird als *modifiziertes Strategie-Iterationsverfahren* bezeichnet und ist oftmals sehr viel effizienter als das herkömmliche Strategie-Iterationsverfahren oder das Werte-Iterationsverfahren (siehe [86]).

Die bisherige Beschreibung von Markow'schen Entscheidungsprozessen erfolgte unter der Annahme, dass die Umgebung vollständig beobachtbar ist und der Agent somit immer weiß, in welchem Zustand er sich befindet. In Kombination mit der Markow-Annahme für das Transitionsmodell bedeutet dies, dass die optimale Strategie nur vom aktuellen Zustand abhängt. Ist die Umgebung hingegen nur teilweise beobachtbar, so weiß der Agent nicht notwendigerweise, in welchem Zustand er sich befindet. Er kann somit nicht einfach die Aktion  $\pi(s)$  ausführen. Darüber hinaus hängen die Nützlichkeitsfunktion eines Zustands  $s$  und die optimale Aktion in  $s$  nicht nur vom Zustand  $s$  ab, sondern davon, wie sicher der Agent es weiß, wenn er in  $s$  ist.

Ein so genannter *partiell beobachtbarer Markow'scher Entscheidungsprozess* (engl. *partially observable MDP* (POMDP)) besitzt die gleichen Elemente wie ein Markow'scher Entscheidungsprozess – ein Transitionsmodell  $T(s, a, s')$  und eine Belohnungsfunktion  $R(s)$ . Zusätzlich verfügt er jedoch über ein Beobachtungsmodell  $O(s, o)$ , das die Wahrscheinlichkeit angibt, im Zustand  $s$  die Beobachtung  $o$  zu machen.

Ein zentrales Konzept zur Beschreibung und Identifikation von Lösungen für POMDPs ist der so genannte *vermutete Zustand*  $b$  (engl. *belief state*). Er bezeichnet die Menge der tatsächlichen Zustände, in denen sich der Agent möglicherweise befindet. Es handelt sich somit um eine Wahrscheinlichkeitsverteilung über alle möglichen Zustände. Die einem tatsächlichen Zustand  $s$  von einem vermuteten Zustand  $b$  zugeordnete Wahrscheinlichkeit wird im Folgenden mit  $b(s)$  bezeichnet. Der Agent kann den aktuellen vermuteten Zustand als bedingte Wahrscheinlichkeitsverteilung über die tatsächlichen Zustände gegeben die bisherige Folge von Beobachtungen und Aktionen berechnen. Bezeichnet  $b(s)$  den bisherigen vermuteten Zustand, führt der Agent die Aktion  $a$  aus und macht die Beobachtung  $o$ , dann lässt sich der neue vermutete Zustand rekursiv mit Hilfe der folgenden Filter-Gleichung berechnen:

$$b'(s') = \alpha O(s', o) \sum_s T(s, a, s') b(s) \quad [2.15]$$

Dabei bezeichnet  $\alpha$  eine Normalisierungskonstante, die dafür sorgt, dass sich die vermuteten Zustände  $b(s)$  zu 1 summieren. Die wesentliche Erkenntnis zum Verständnis von POMDPs liegt darin begründet, dass die optimale Aktion nur vom aktuellen vermuteten Zustand des Agenten abhängt und nicht von dem tatsächlichen Zustand, in dem er sich befindet. Die optimale Strategie kann somit als Abbildung  $\pi^*(b)$  von vermuteten Zuständen auf Aktionen beschrieben werden. Der Entscheidungszyklus eines POMDP-Agenten umfasst die folgenden Schritte:

1. Gegeben den aktuellen vermuteten Zustand  $b$ , führe die Aktion  $a = \pi^*(b)$  aus.
2. Empfange die Beobachtung  $o$ .
3. Berechne den neuen aktuellen vermuteten Zustand mit Hilfe von Gleichung 2.15 und fahre mit Schritt 1 fort.

In [86] wird gezeigt, dass sich die Lösung eines POMDPs auf einem physikalischen Zustandsraum auf die Lösung eines MDPs auf dem korrespondierenden Raum der vermuteten Zustände reduzieren lässt. Der entsprechende MDP besitzt jedoch einen kontinuierlichen (und üblicherweise sehr hochdimensionalen) Zustandsraum. Aus diesem Grund lässt sich keines der oben vorgestellten Lösungsverfahren direkt auf solche MDPs anwenden. Es lassen sich jedoch spezielle Versionen des Werte-Iterationsverfahrens und des Strategie-Iterationsverfahrens entwickeln, die sich auf MDPs mit kontinuierlichen Zustandsräumen anwenden lassen.

### Erzeugung von POMDP-Modellen anhand von Hintergrundwissen

Wie im vorhergehenden Abschnitt erläutert wurde, bilden POMDPs ein Rahmenwerk für die Entscheidungsfindung, für das Algorithmen zur Berechnung von näherungsweise optimalen Entscheidungen existieren. Die Berechnung der Entscheidungsstrategien benötigt jedoch Modelle zur Beschreibung der stochastischen Eigenschaften der Umgebung, der Unsicherheiten

von Messungen und der Missionsziele. Die Erzeugung dieser Modelle für multimodale Serviceroboter und komplexe Missionsszenarien ist ein Feld aktueller Forschung. In [92] wird ein System umrissen, das POMDPs zur Entscheidungsfindung für einen realen, hochgradig multimodalen Serviceroboter einsetzt. In diesem Zusammenhang wird ein zweischichtiger Ansatz zur Erzeugung der Beobachtungs-, der Transitions- und der Belohnungsmodelle anhand von einfachen, abstrakten und wiederverwendbaren Beschreibungen der Eigenschaften des Szenarios und der Umgebung vorgestellt.

Da sehr gute Verfahren zur Strategieberechnung für nichtfaktorisierte POMDPs existieren, werden verschiedene Unterräume  $S_i$ , die unterschiedliche Modalitäten repräsentieren, in einen einzigen vereinigten Zustandsraum  $S$  abgebildet:

$$s = (s_0, s_1, \dots, s_n) \in S = S_0 \times S_1 \times \dots \times S_n \quad [2.16]$$

Die erste Ebene des genannten Ansatzes erzeugt Modellmatrizen anhand von parametrisierbaren Funktionen. Ein zentrales Element hierbei sind Tabellen, welche die temporären Ergebnisse der auf diesen Funktionen basierenden Berechnungen speichern. Alle Tabellen verfügen unabhängig davon, ob sie für das Beobachtungsmodell, das Belohnungsmodell oder für individuelle Aktionen innerhalb eines Transitionsmodells bestimmt sind, über einen einheitlichen Aufbau:

$$T : \{1, \dots, k\} \times \{1, \dots, m\} \times \{1, \dots, n\} \mapsto \mathbb{R} \quad [2.17]$$

Dabei bezeichnen  $k$  und  $m$  die Anzahlen der Zeilen und Spalten in den zugehörigen Modellmatrizen und  $n$  gibt die Anzahl der den einzelnen Modalitäten entsprechenden Unterräume an. Nachdem die finalen Ergebnisse für alle Tabellen berechnet wurden, lässt sich das Modell durch Produktbildung über alle Unterräume und anschließende Normalisierung berechnen:

$$M : \{1, \dots, k\} \times \{1, \dots, m\} \mapsto \mathbb{R}, \quad M(i, j) = \prod_{l=1}^n T(i, j, l) \quad [2.18]$$

Die Tabellen selbst werden Schritt für Schritt durch die Anwendung von Regeln berechnet. Regeln können einzelne Einträge, Reihen, Spalten oder ganze Tabellen verändern, während sie unterschiedliche Arten von arithmetischen Operationen auf diese anwenden. Hierdurch wird es beispielsweise für das Transitionsmodell möglich, Regeln für alle Transitionen von einem bestimmten Zustand aus, für alle Transitionen in einen bestimmten Zustand, für alle Transitionen einer bestimmten Aktion oder für das ganze Modell anzuwenden.

Die Erzeugung von Modellen mit einer größeren Anzahl von Zuständen ist nur durch die Anwendung der oben genannten Funktionen und Regeln möglich. Ihre manuelle Verwendung ist jedoch sehr aufwändig, da für jedes Szenario neue Regeln erstellt werden müssen. Darüber hinaus enthalten sie kaum semantisch fundiertes und wiederverwendbares Wissen über die Welt.

Aus diesem Grund wird in [92] eine zweite, abstraktere Ebene der Weltmodellierung verwendet. Anstelle von arithmetischen Regeln werden Wissensdomänen als wesentliche konzeptuelle Struktur verwendet. Wissen aus dieser konzeptuellen Struktur wird für die Transitions- und die Belohnungsmodelle verwendet. Die Beobachtungsmodelle werden hingegen anhand einer spezifischen Untersuchung der Messungengenauigkeiten der individuellen Modalitäten abgeleitet:

$$O(i, j) = \prod_{l=1}^n (1 - \langle i, j \rangle_l), \quad [2.19]$$

wobei  $\langle \dots \rangle_l$  eine geeignete Distanzmetrik für die Einzelmodalität  $l$  bezeichnet.

Für die Erzeugung der Transitions- und Belohnungsmodelle werden verschiedene Wissensdomänen verwendet. Für das in [92] gewählte Anwendungsszenario eines anthropomorphen Serviceroboters, der Bedienungstätigkeiten ausübt, sind die folgenden Domänen angegeben:

1. Allgemeine beschreibende Attribute für den Roboter, Menschen und die Umgebung.
2. Modalitätsspezifisches Zustandswissen.
3. Aktionstypen-spezifisches Wissen.
4. Wissen über die roboterunabhängige Dynamik der Umgebung.
5. Allgemeines Interaktionswissen.

Beispiele für die erste Wissensdomäne sind etwa *Roboter::explorativ* oder *Roboter::effizient*, welche die zustandsspezifischen Regeln des Belohnungsmodells beeinflussen. Die Wissens-elemente der zweiten Ebene besitzen ebenfalls eine Beschreibung in Form von Regeln, welche wiederum die Regeln der ersten Ebene, die bei Verwendung des Wissens erzeugt werden, sowie die entsprechende Parametrisierung enthält. Das in diesen Regeln adressierte Wissen ähnelt den Regeln der ersten Ebene, verfügt jedoch über keine arithmetischen Parameter.

Der beschriebene Ansatz wurde zur Erstellung eines Modells für ein realistisches Missions-szenario verwendet, welches anschließend in der in [93] vorgestellten Steuerungsarchitektur verwendet wurde, um das Verhalten eines anthropomorphen Serviceroboters in dem modellierten Szenario zu steuern. Im Rahmen des gewählten Bedienungsszenarios wurden 28 Zustände und 11 Aktionen unterschieden. Dies führte zu einem Belohnungsmodell mit 308 Einträgen, einem Beobachtungsmodell mit 784 Wahrscheinlichkeiten und einem Transitionsmodell mit 8624 Wahrscheinlichkeiten. Die manuelle Erzeugung eines solchen Modells wäre nicht praktikabel.

## 2.5. Zusammenfassung und Fazit

In diesem Kapitel wurde der Stand der Forschung in den für die vorliegende Arbeit relevanten Bereichen vorgestellt. Im ersten Teil wurde zunächst ein Überblick über autonome Servicerob-

boter im Bereich der Inspektion von komplexen Umgebungen präsentiert. Nach der Vorstellung der aktuell existierenden Systeme, welche insbesondere verschiedene mehrsegmentige Roboter zur Inspektion von Kanalsystemen sowie verschiedene Rover zur Erkundung von Planeten umfasste, wurde auf die in diesem Bereich verwendeten Steuerungsarchitekturen sowie auf die verwendeten Verfahren zur Sensordatenauswertung eingegangen.

Im zweiten Teil des Kapitels wurden grundlegende Verfahren und Techniken zur ontologiebasierten Wissensrepräsentation vorgestellt. Nach der Erläuterung der Definition des Begriffs der Ontologie wurde näher auf geeignete Entwurfskriterien bei der Erstellung von Ontologien sowie auf die verschiedenen Vorteile der Verwendung von Ontologien eingegangen. Daran anknüpfend wurden semantische Technologien aus dem Bereich des Semantic Web erläutert und es wurde ein Überblick über die Verwendung von semantischem Wissen in der Robotik präsentiert. In diesem Zusammenhang wurde insbesondere auf ein semantisches wissensbasiertes Rahmenwerk zur Verbesserung des Situationsbewusstseins von autonomen Unterwasserfahrzeugen eingegangen.

Im dritten Teil des Kapitels wurden für diese Arbeit relevante Planungsverfahren und Ausführungssysteme vorgestellt. Der Schwerpunkt lag hierbei auf Hierarchischen Aufgabennetzwerken und verwandten Repräsentationsformen. Nach der Vorstellung der grundlegenden Definitionen in Bezug auf Hierarchische Aufgabennetze wurde näher auf Flexible Programme (FPs) eingegangen. Diese wurden zur hierarchischen Repräsentation von Handlungswissen für Serviceroboter entwickelt und werden ähnlich zu TDL als Hierarchische Aufgabennetze dargestellt. Schließlich wurden Nebenläufige Hierarchische Pläne in Form von Concurrent Hierarchical Plans (CHiPs) vorgestellt. Diese ermöglichen es, anhand von zusammengefassten Planinformationen bereits auf abstrakten Ebenen mögliche Konflikte zwischen den einzelnen Unterplänen zu erkennen und diese mit geeigneten Operationen aufzulösen.

Im vierten Teil des Kapitels wurde auf grundlegende Verfahren und Techniken zur Situationsbewertung und Entscheidungsfindung eingegangen. Im Fokus standen dabei insbesondere probabilistische Ansätze. Es wurden zwei der zentralen Formalismen in diesem Bereich vorgestellt: Bayes'sche Netzwerke und Markow'sche Entscheidungsprozesse. Für partiell beobachtbare Markow'sche Entscheidungsprozesse (POMDPs) wurde außerdem ein vielversprechender Ansatz zur Erzeugung der zur Beschreibung der stochastischen Eigenschaften der Umgebung, der Unsicherheiten von Messungen und der Missionsziele benötigten Modelle anhand von Hintergrundwissen erläutert.

Zusammenfassend ist festzustellen, dass sich die bestehenden Ansätze zur Entwicklung von autonomen Inspektionsrobotern im Wesentlichen in zwei Bereiche gliedern. Diese umfassen zum einen die Hardwareentwicklung und die Bewegungssteuerung und zum anderen die Entwicklung geeigneter Sensorsysteme, die automatische Platzierung der Sensoren und die Sensordatenauswertung. Es existieren hingegen nur sehr wenige integrierte Ansätze. Insbesonde-

re findet die Bewertung der Datenauswertungsergebnisse an Bord des Roboters bisher kaum Beachtung. Diese bildet jedoch eine wesentliche Voraussetzung für die aktive Untersuchung interessierender Entitäten und das Treffen situationsabhängiger Entscheidungen.

Im Hinblick auf den in Kapitel 1 definierten Regelkreis der autonomen Inspektion (siehe Abbildung 1.1) ist festzuhalten, dass gegenwärtig kein Ansatz bekannt ist, der diesen vollständig abbildet und das benötigte menschliche Expertenwissen bestehend aus Fachwissen, Erfahrungen und Hintergrundwissen mit Hilfe von semantischen Technologien einbindet. Bei Betrachtung der in diesem Kapitel vorgestellten Themenfelder ist jedoch zu erkennen, dass in den einzelnen Bereichen bereits vielversprechende Ansätze existieren und somit die Voraussetzungen für die Entwicklung einer semantischen Missionssteuerung für autonome Inspektionsroboter gegeben sind.





### **3. Entwurf der Steuerungsarchitektur**

Ein autonomer Inspektionsroboter soll selbständig vorgegebene Inspektionsmissionen planen und durchführen. Insbesondere sollen gefundene interessierende Entitäten aktiv und datengetrieben untersucht werden, bis eine ausreichende Konfidenz der Datenauswertungsergebnisse erreicht ist. Sind die Ergebnisse der Datenauswertung unsicher, sollen in Abhängigkeit von der jeweiligen Situation zusätzliche Aktionen zur Untersuchung durchgeführt werden. Dies kann etwa die Verwendung zusätzlicher Sensoren, die Anwendung alternativer Datenauswertungsverfahren oder die Aufnahme weiterer Daten aus unterschiedlichen Perspektiven umfassen. All dies muss zudem in einer dynamischen, nichtdeterministischen Umgebung und unter Berücksichtigung der vorhandenen Ressourcen des Systems erfolgen. Die Komplexität eines solchen Systems ist daher als hoch einzuschätzen. Eine klar strukturierte und speziell auf die Anforderungen eines autonomen Inspektionssystems zugeschnittene Steuerungsarchitektur bildet somit eine wesentliche Voraussetzung, um die hohe Systemkomplexität beherrschen zu können.

In diesem Kapitel wird zunächst auf die Anforderungen an die Steuerungsarchitektur eines autonomen Inspektionsroboters eingegangen. Anschließend wird das im Rahmen dieser Arbeit entwickelte Konzept für die Architektur der semantischen Missionssteuerung vorgestellt. Hierzu wird zunächst ein Überblick über den Aufbau und die einzelnen Komponenten der Architektur gegeben. Daran anknüpfend wird die Interaktion der einzelnen Komponenten näher erläutert. Abschließend wird eine Einordnung der Architektur im Vergleich zu bereits bestehenden Ansätzen vorgenommen.

#### **3.1. Anforderungen**

Die Steuerungsarchitektur eines autonomen Inspektionsroboters sollte den Regelkreis bestehend aus Inspektionsplanung, Planausführung, Inspektionsdatenauswertung, Bewertung der Datenauswertungsergebnisse, Entscheidungsfindung und Neuplanung (siehe Abbildung 1.1) geeignet abbilden. Darüber hinaus sollte sie mit möglichst geringem Aufwand für unterschiedliche Inspektionsroboter und Inspektionsdomänen umgesetzt werden können. Dies erfordert einen klar strukturierten, modularen und erweiterbaren Aufbau, welcher die einzelnen Komponenten klar voneinander abgrenzt und ihr Zusammenwirken eindeutig festlegt. Dies vereinfacht neben der Umsetzung auch die spätere Evaluierung des Systems. Des Weiteren sollte die Architektur über einen hierarchischen Aufbau verfügen, welcher einen schrittweisen Übergang von der reaktiven bzw. verhaltensbasierten Basissteuerung des Roboters hin zu den eher deliberativen

bzw. wissensbasierten Ebenen der Planung und Entscheidungsfindung ermöglicht. Hierdurch wird sowohl ein schnelle Reaktion des Systems auf Ausnahmesituationen als auch das Verfolgen längerfristiger Ziele ermöglicht. Ein hierarchischer Aufbau der Architektur ist neben dem Kontrollfluss auch für den Datenfluss wichtig. Auch hier sollte eine schrittweise Überführung der Daten von der hardwarenahen, subsymbolischen Ebene hin zu symbolischen bzw. semantischen Ebenen möglich sein. Dies ist vor allem für das so genannte *Symbol Grounding* [45], welches alle auf der semantischen Ebene definierten Konzepte und Relationen in den Sensor- und Aktuatordaten verankert, von entscheidender Bedeutung.

Darüber hinaus ist eine enge Verzahnung der Planung und Ausführung wichtig, da die Untersuchung interessierender Entitäten meist in einer iterativen Art und Weise erfolgt, welche häufiges Neuplanen erfordert. Zudem sollte die Architektur die nebenläufige Ausführung verschiedener Aufgaben unterstützen. So sollte der Inspektionsroboter etwa in der Lage sein, sich während des Absuchens einer Region gleichzeitig fortzubewegen und die Daten der Inspektionssensoren auf potentielle interessierende Entitäten hin zu untersuchen oder mehrere Sensoren gleichzeitig einzusetzen. Aufgrund des dynamischen, nichtdeterministischen Charakters der Umgebung sollte die Architektur zudem die Behandlung von Fehlern und Ausnahmen auf unterschiedlichen Ebenen erlauben. Des Weiteren ist die Verwaltung der Ressourcen des Systems von großer Wichtigkeit und sollte daher klar in der Architektur adressiert werden.

Zentrales Element der Architektur sollte jedoch eine Wissensbasis sein, welche ein geeignetes Weltmodell enthält sowie das zur Durchführung von autonomen Inspektionsmissionen notwendige Hintergrund- und Expertenwissen beinhaltet und dem System zur Verfügung stellt. Nur mit Hilfe einer solchen Wissensbasis lässt sich die semantische Interoperabilität der einzelnen Komponenten der Steuerungsarchitektur sicherstellen. Auf die genauen Anforderungen an die Wissensbasis wird in Kapitel 4 detaillierter eingegangen.

## **3.2. Architektur der semantischen Missionssteuerung**

Nachdem im vorhergehenden Abschnitt auf die Anforderungen an die Steuerungsarchitektur eines autonomen Inspektionsroboters eingegangen wurde, wird nun das im Rahmen dieser Arbeit entwickelte Konzept für die Architektur der semantischen Missionssteuerung vorgestellt. Hierzu wird zunächst auf den allgemeinen Aufbau der Architektur eingegangen. Anschließend werden die Komponenten und ihre Interaktion beschrieben.

### **3.2.1. Struktureller Aufbau**

Für den Aufbau der Missionssteuerung (siehe Abbildung 3.1) wurde den Anforderungen entsprechend ein modularer, hierarchischer Ansatz gewählt, welcher die Komponenten der Missionssteuerung auf vier unterschiedlichen Ebenen anordnet [114]. Als Abstraktionskriterium für

den hierarchischen Aufbau wurde hierbei die Art der verarbeiteten Daten gewählt, da diesen bei der aktiven, datengetriebenen Inspektion eine besondere Bedeutung zukommt. Auf der untersten, der subsymbolischen Ebene befinden sich *Sensor-* und *Aktorinterface* zum Robotersystem. Auf der darüber liegenden subsymbolisch-symbolischen Ebene befinden sich die *Inspektionsdaten-* und die *Navigationsdatenauswertung* sowie die *Basissteuerung*. Die symbolisch-semantische Ebene umfasst die *Semantische Inspektion*, die *Semantische Kartierung*, die *Semantische Navigation* sowie die *Ausführungseinheit*. Die oberste, semantische Ebene enthält den *Manager* und die *Bedienschnittstelle*. Zentrales Element der Missionssteuerung ist jedoch die *Wissensbasis*, welche das für die Durchführung von autonomen Inspektionsmissionen benötigte Wissen enthält.

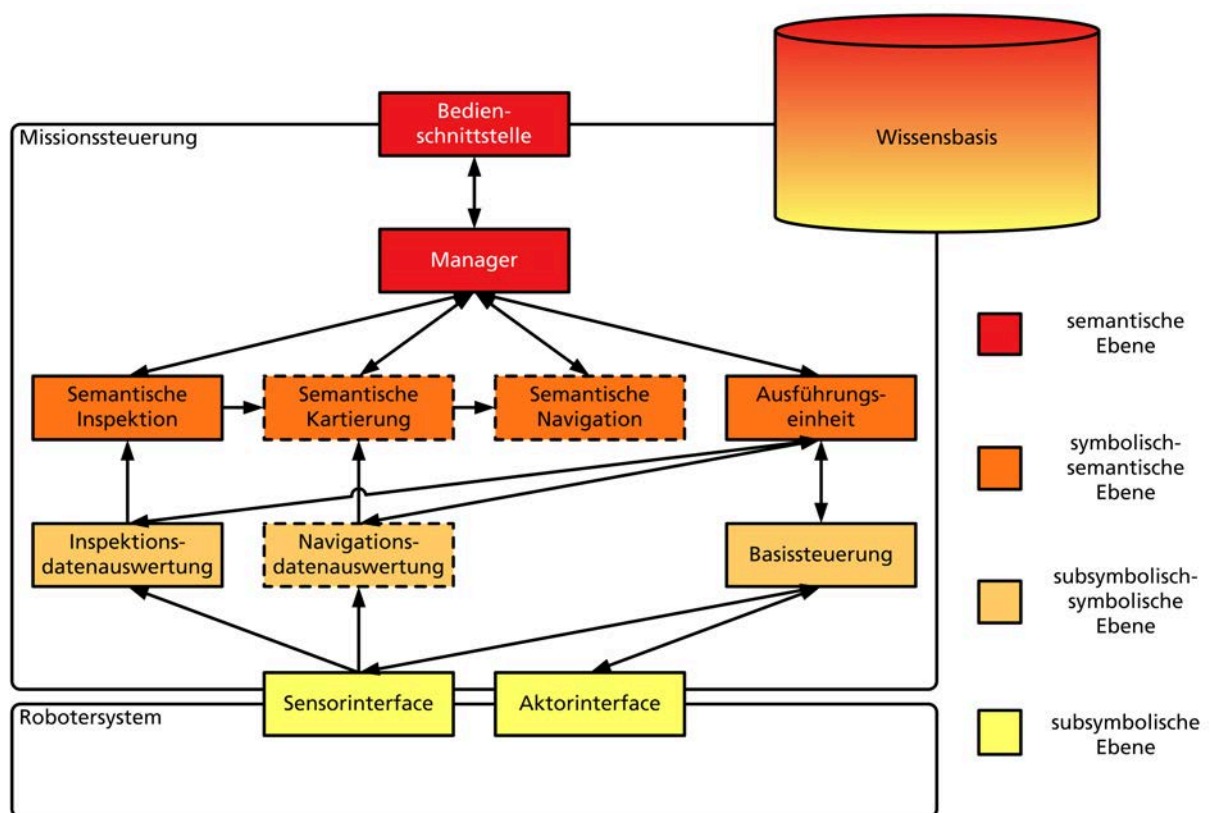


Abb. 3.1.: Die Architektur der semantischen Missionssteuerung. Die gestrichelt umrandeten Komponenten sind nicht Gegenstand dieser Arbeit, sondern werden im Rahmen einer parallel laufenden Arbeit entwickelt (siehe [101]).

### 3.2.2. Aufgaben der einzelnen Komponenten

Im Folgenden werden die Aufgaben der einzelnen Komponenten beschrieben.

**Inspektionsdatenauswertung** Die *Inspektionsdatenauswertung* (siehe Abbildung 3.2) verarbeitet die Daten der Inspektionssensoren und analysiert diese im Hinblick auf potentielle in-

interessierende Entitäten. Hierzu werden die Sensordaten zunächst geeignet vorverarbeitet. Wird eine potentielle interessierende Entität detektiert, so wird die zugehörige Region in den Sensordaten segmentiert. Für die segmentierten Regionen werden anschließend Merkmale berechnet, anhand derer eine Klassifikation der potentiellen interessierenden Entitäten durchgeführt wird. Abschließend findet eine Zuordnung zu gegebenenfalls bereits gefundenen interessierenden Entitäten statt. Die *Inspektionsdatenauswertung* ist zustandslos und berücksichtigt jeweils nur die aktuelle Sensormessung.

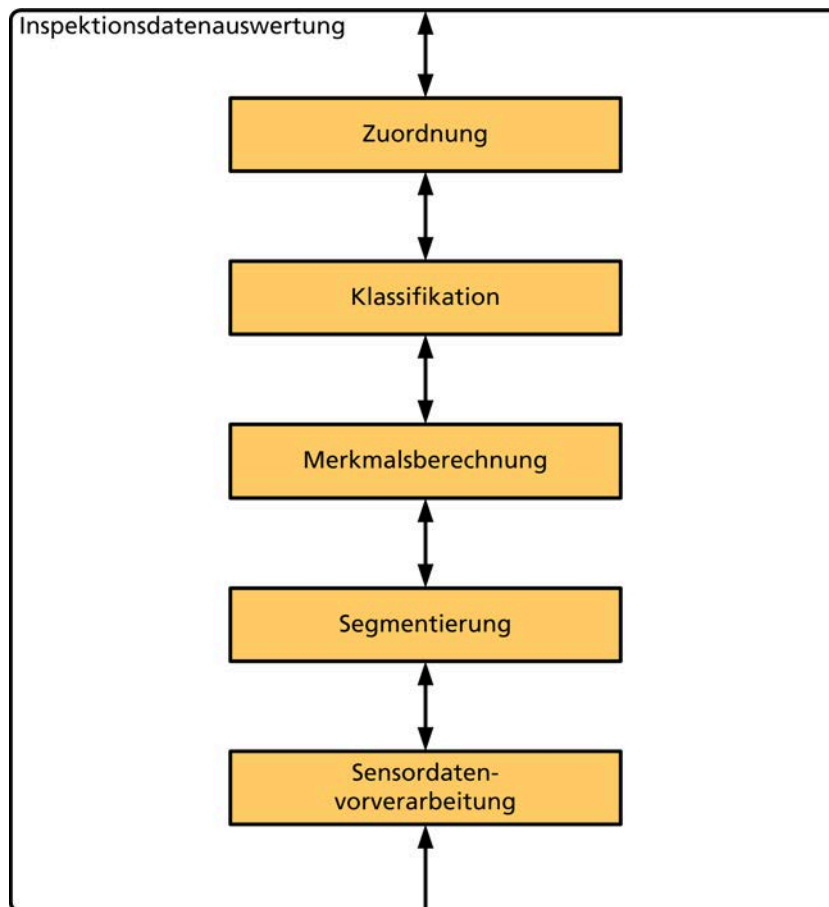


Abb. 3.2.: Die Unterkomponenten der *Inspektionsdatenauswertung*.

**Navigationdatenauswertung** Die *Navigationdatenauswertung* verarbeitet die Daten der Navigationssensoren. Sie lokalisiert und klassifiziert Regionen in den Sensordaten und bestimmt die Parameter der Regionen. Wie die *Inspektionsdatenauswertung* ist auch die *Navigationdatenauswertung* zustandslos und berücksichtigt jeweils nur die aktuelle Sensormessung.

**Semantische Inspektion** Die *Semantische Inspektion* (siehe Abbildung 3.3) führt eine temporale Fusion der einzelnen Ergebnisse der *Inspektionsdatenauswertung* durch und bewertet diese. In Abhängigkeit von der Bewertung der fusionierten Ergebnisse und basierend auf dem

semantischen Inspektionsmodell generiert sie Inspektionsaufgaben zur weiteren Untersuchung der gefundenen interessierenden Entitäten und schlägt diese dem *Manager* vor.

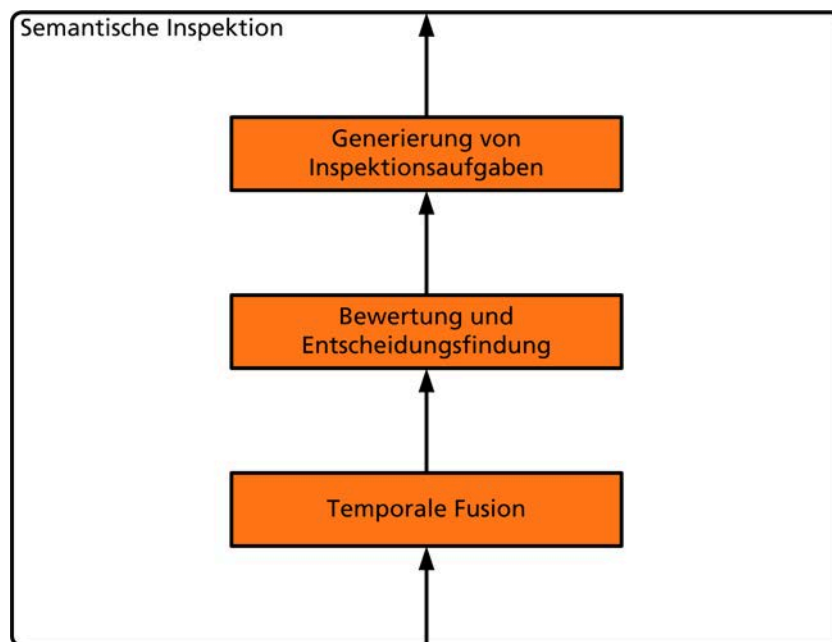


Abb. 3.3.: Die Unterkomponenten der *Semantischen Inspektion*.

**Semantische Kartierung** Die *Semantische Kartierung* führt eine temporale Fusion der einzelnen Ergebnisse der *Navigationsdatenauswertung* durch und berechnet bzw. aktualisiert die semantische Regionenkarte der Umgebung. Darüber hinaus registriert sie die von der *Semantischen Inspektion* gefundenen interessierenden Entitäten in der semantischen Regionenkarte.

**Semantische Navigation** Die *Semantische Navigation* bewertet anhand der semantischen Regionenkarte die aktuelle Situation in Bezug auf die Erfüllung der Missionsaufgaben. In Abhängigkeit von dieser Bewertung und basierend auf dem semantischen Navigationsmodell generiert sie Navigationsaufgaben und schlägt diese dem *Manager* vor.

**Manager** Der *Manager* ist die oberste Kontroll- und Entscheidungskomponente. Er erzeugt für die vom Benutzer über die *Bedienschnittstelle* vorgegebenen Missionsaufgaben entsprechende Pläne (siehe Abschnitt 5.3), gibt diese an die *Ausführungseinheit* weiter und überwacht und koordiniert ihre Ausführung. Des Weiteren nimmt er die von der *Semantischen Inspektion* und der *Semantischen Navigation* vorgeschlagenen Aufgaben entgegen und integriert diese mittels iterativer Neuplanung in die bestehenden Pläne. Darüber hinaus ist er für die Behandlung von Ausnahme- und Fehlersituationen zuständig, sofern diese nicht von der *Ausführungseinheit* aufgelöst werden können. Dies erfolgt üblicherweise durch Neuplanung.

**Ausführungseinheit** Die *Ausführungseinheit* empfängt Pläne vom *Manager*. Sie zerlegt die Pläne in die jeweiligen elementaren Aufgaben und wählt dabei in Abhängigkeit von der aktuellen Situation zwischen alternativen Methoden zur Erfüllung der Aufgaben aus. Darüber hinaus prüft sie die Ausführbarkeit der Aufgaben anhand der Vorbedingungen, der temporalen Bedingungen zwischen den einzelnen Aufgaben (zum Beispiel serielle oder nebenläufige Ausführung) sowie anhand der Ressourcenanforderungen. Die zum jeweiligen Zeitpunkt ausführbaren Aktionen gibt sie an die *Basissteuerung*, die *Inspektionsdatenauswertung* und die *Navigationsdatenauswertung* weiter und überwacht ihre Ausführung. Die *Ausführungseinheit* ist zudem für die Behandlung von Fehler- und Ausnahmesituationen zuständig. Kann sie eine solche Situation nicht auflösen, meldet sie dies dem *Manager*.

**Basissteuerung** Die *Basissteuerung* nimmt einzelne elementare Aufgaben oder eine Menge von nebenläufig ausführbaren elementaren Aufgaben von der *Ausführungseinheit* entgegen. Für diese werden entsprechende subsymbolische Kommandos an das *Sensor-* und das *Aktorinterface* der Roboterplattform weitergegeben, deren Ausführung überwacht wird.

**Sensorinterface** Das *Sensorinterface* stellt eine Schnittstelle zur Sensorik der Roboterplattform zur Verfügung und erlaubt deren Ansteuerung auf subsymbolischer Ebene.

**Aktorinterface** Das *Aktorinterface* stellt eine Schnittstelle zur Aktorik der Roboterplattform zur Verfügung und erlaubt deren Ansteuerung auf subsymbolischer Ebene.

**Bedienschnittstelle** Die *Bedienschnittstelle* dient der Interaktion des Systems mit dem Benutzer. Über die Bedienschnittstelle kann der Benutzer das System in Betrieb nehmen und entsprechende Missionsaufgaben vorgeben. Außerdem kann er bei Bedarf das laufende System überwachen und nach Abschluss der Inspektion die entsprechenden Ergebnisse abrufen.

**Wissensbasis** Die *Wissensbasis* enthält das Weltmodell, welches den aktuellen Zustand der realen Welt abbildet. Darüber hinaus stellt die Wissensbasis dem System das für die Durchführung von autonomen Inspektionsmissionen benötigte Hintergrund- und Expertenwissen in Form des semantischen Inspektionsmodells zur Verfügung. Der Zugriff auf die Wissensbasis erfolgt überwiegend von den Komponenten der symbolisch-semantischen und der semantischen Ebene aus. Neben den semantischen Informationen enthält die Wissensbasis jedoch auch Informationen auf symbolischer und subsymbolischer Ebene. Eine detaillierte Beschreibung der Wissensbasis findet sich in Kapitel 4.

Die *Navigationsdatenauswertung*, die *Semantische Kartierung* und die *Semantische Navigation* sind nicht Gegenstand dieser Arbeit, sondern werden im Rahmen einer parallel laufenden

Arbeit entwickelt (siehe [101]). Sie werden jedoch der Vollständigkeit halber mit aufgeführt, um das Gesamtkonzept zu verdeutlichen.

### 3.2.3. Zusammenwirken der Komponenten

Das Zusammenwirken der Komponenten wird im Folgenden anhand von zwei Abläufen verdeutlicht. Der erste Ablauf beschreibt allgemein die Vorgabe von Missionsaufgaben durch den Benutzer, die Erzeugung von entsprechenden Plänen zur Erfüllung der Missionsaufgaben und die Planausführung. Der zweite Ablauf bezieht die Auswertung der Inspektionsdaten, ihre Beurteilung und die automatische Erzeugung von neuen Inspektionsaufgaben zur aktiven Untersuchung interessierender Entitäten mit ein.

#### Allgemeiner Ablauf

Die im Rahmen einer Inspektionsmission zu erfüllenden Aufgaben, die so genannten *Missionsaufgaben*, werden dem System vom Benutzer über die Bedienschnittstelle vorgegeben. Die Missionsaufgaben werden von der Bedienschnittstelle an den Manager weitergegeben. Dieser erstellt ein entsprechendes Planungsproblem und ruft den Inspektionsplaner auf (siehe Abschnitt 5.3). Der generierte Plan wird anschließend an die Ausführungseinheit übergeben, welche den Plan in einzelne bzw. Mengen von parallel ausführbaren elementaren Aufgaben verfeinert, die an die Basissteuerung, die Inspektions- und die Navigationsdatenauswertung weitergegeben werden. Diese beinhalten jeweils so genannte *Ausführungskomponenten*, welche die einzelnen Fähigkeiten des Systems abbilden und über einen generischen Aufbau verfügen. Die Ausführungskomponenten sind für die Ausführung der elementaren Aufgaben zuständig und übermitteln der Ausführungseinheit den Status der jeweiligen Ausführung. Wird der Ausführungseinheit signalisiert, dass eine Aufgabe nicht wie gewünscht ausgeführt werden konnte, kann diese entsprechende Maßnahmen zur Behandlung der Fehlersituation ergreifen (siehe Abschnitt 5.4.1). Gelingt dies nicht, kann die Ausführungseinheit dies ihrerseits dem Manager signalisieren, welcher versuchen kann, das Problem durch Neuplanung zu beheben. Zur Veranschaulichung des geschilderten Ablaufs ist dieser in Form eines UML-Sequenzdiagramms in Abbildung 3.4 dargestellt.

#### Aktive Untersuchung interessierender Entitäten

Für die aktive Untersuchung interessierender Entitäten erweitert sich der im vorigen Abschnitt beschriebene allgemeine Ablauf wie folgt: Neben der Statusübermittlung an die Ausführungseinheit senden die Ausführungskomponenten der Inspektionsdatenauswertung die Ergebnisse der einzelnen Datenauswertungsschritte an die Semantische Inspektion. Diese fusioniert die Ergebnisse fortlaufend und bewertet sie. Anhand des semantischen Inspektionsmodells wird für

### 3. Entwurf der Steuerungsarchitektur

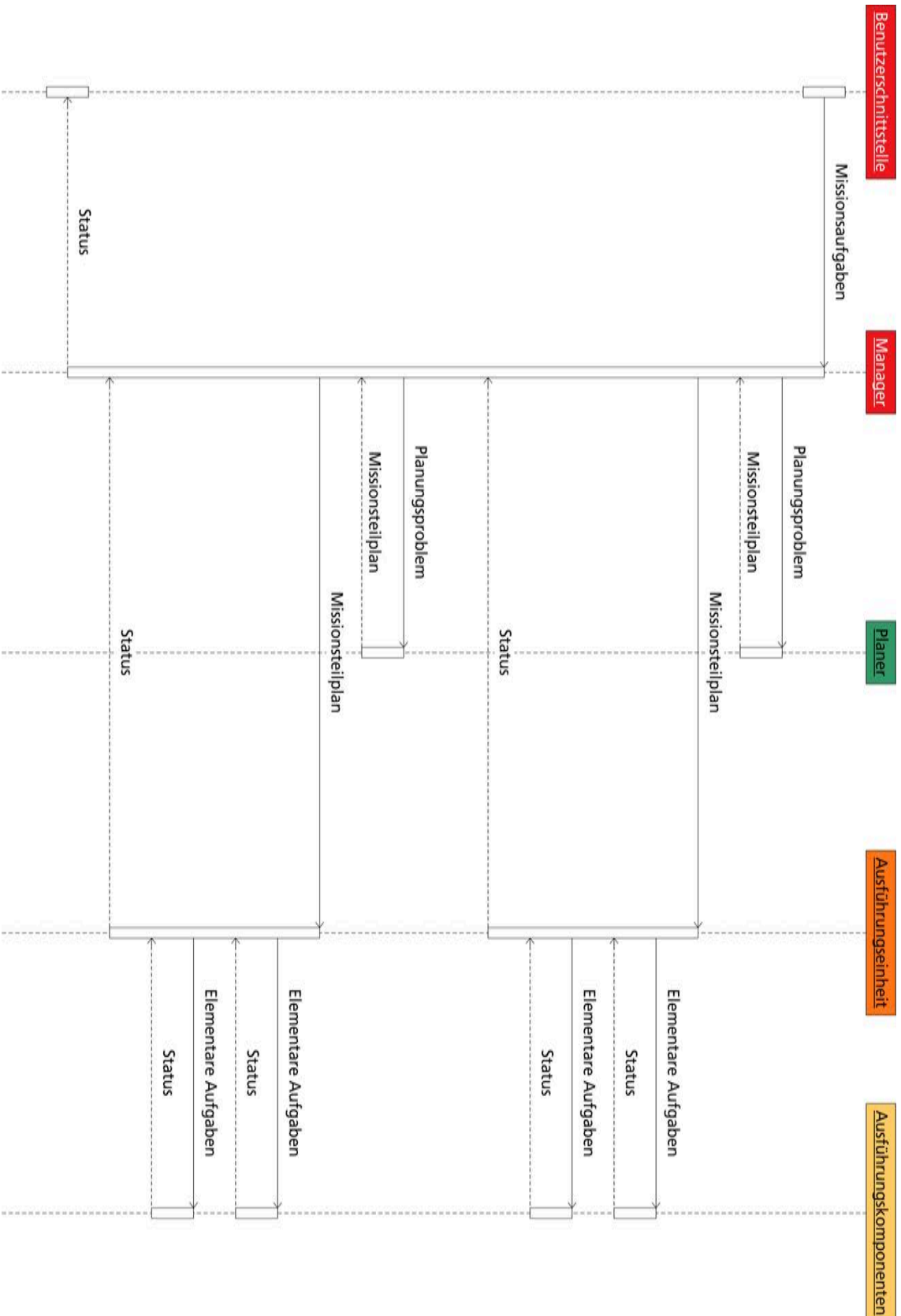


Abb. 3.4.: UML-Sequenzdiagramm zur Veranschaulichung des allgemeinen Ablaufs bei der Planerzeugung und -ausführung.



jede untersuchte interessierende Entität entschieden, ob das Ergebnis der Inspektionsdatenauswertung eine ausreichende Konfidenz aufweist oder ob und wenn ja wie die interessierende Entität weiter untersucht werden soll. Entsprechende neue Inspektionsaufgaben werden mit Prioritäten versehen und dem Manager vorgeschlagen. Dieser berücksichtigt die vorgeschlagenen Inspektionsaufgaben bei der Festlegung der im nächsten Teil der Mission durchzuführenden Aufgaben in Abhängigkeit von den zur Verfügung stehenden Ressourcen. Anschließend beginnt ein neuer Zyklus des Regelkreises bestehend aus Inspektionsplanung, Planausführung, Inspektionsdatenauswertung, Bewertung der Datenauswertungsergebnisse, Entscheidungsfindung und Neuplanung. Zur Veranschaulichung ist auch dieser Ablauf in Form eines UML-Sequenzdiagramms in Abbildung 3.5 dargestellt. Dabei wurde aus Gründen der Übersichtlichkeit auf eine Darstellung der die Navigation betreffenden Aspekte verzichtet.

### **3.3. Einordnung der Architektur**

Bei der im Rahmen dieser Arbeit entworfenen Architektur der semantischen Missionssteuerung handelt es sich um eine modular aufgebaute Mehrschichten-Architektur. Ihr Aufbau folgt jedoch nicht durchgängig dem klassischen Drei-Schichten-Modell bestehend aus Planung, Sequenzierung und Echtzeitsteuerung. Vielmehr lässt sich ihr Aufbau in zwei Schichten mit jeweils zwei Unterebenen gliedern. Die untere Schicht ist funktional und umfasst die subsymbolische und die subsymbolisch-symbolische Ebene. Die obere Schicht besteht aus der symbolisch-semantischen und der semantischen Ebene. Sie ähnelt der Entscheidungsschicht der LAAS-Architektur, welche einen Planer und einen Supervisor enthält. Der Aspekt der Planung wird in der semantischen Missionssteuerung im Wesentlichen vom Manager abgebildet. Die Funktionalität des Supervisors wird hauptsächlich von der Ausführungseinheit übernommen, welche Aufgaben zerlegt, alternative Methoden zur Erfüllung von Aufgaben auswählt und die Ausführung überwacht. Darüber hinaus nimmt der Manager die Funktion eines Koordinators ein, der Planung und Ausführung miteinander verzahnt und für die Erfüllung der Missionsaufgaben verantwortlich ist.

Neben den beschriebenen Ähnlichkeiten zur LAAS-Architektur ist auch eine gewisse Affinität zur CLARATy-Architektur der NASA vorhanden. Diese besteht ebenfalls aus zwei Schichten: einer funktionalen Schicht und einer Entscheidungsschicht, welche ähnlich wie in der LAAS-Architektur Planungs- und Ausführungsfähigkeiten kombiniert, um durch kontinuierliche Neuplanung auf unvorhergesehene Situationen reagieren zu können. Die Aspekte der aktiven, datengetriebenen Untersuchung von interessierenden Entitäten ähneln in gewisser Hinsicht dem OASIS-System der NASA, welches das CLEaR-System (eine Instantiierung der CLARATy-Entscheidungsebene) um die Auswertung von wissenschaftlichen Daten erweitert und es so ermöglicht, dass das System von sich ergebenden günstigen wissenschaftlichen Gelegenheiten getrieben wird. Die Wissensbasis als zentrales Element der semantischen Missions-

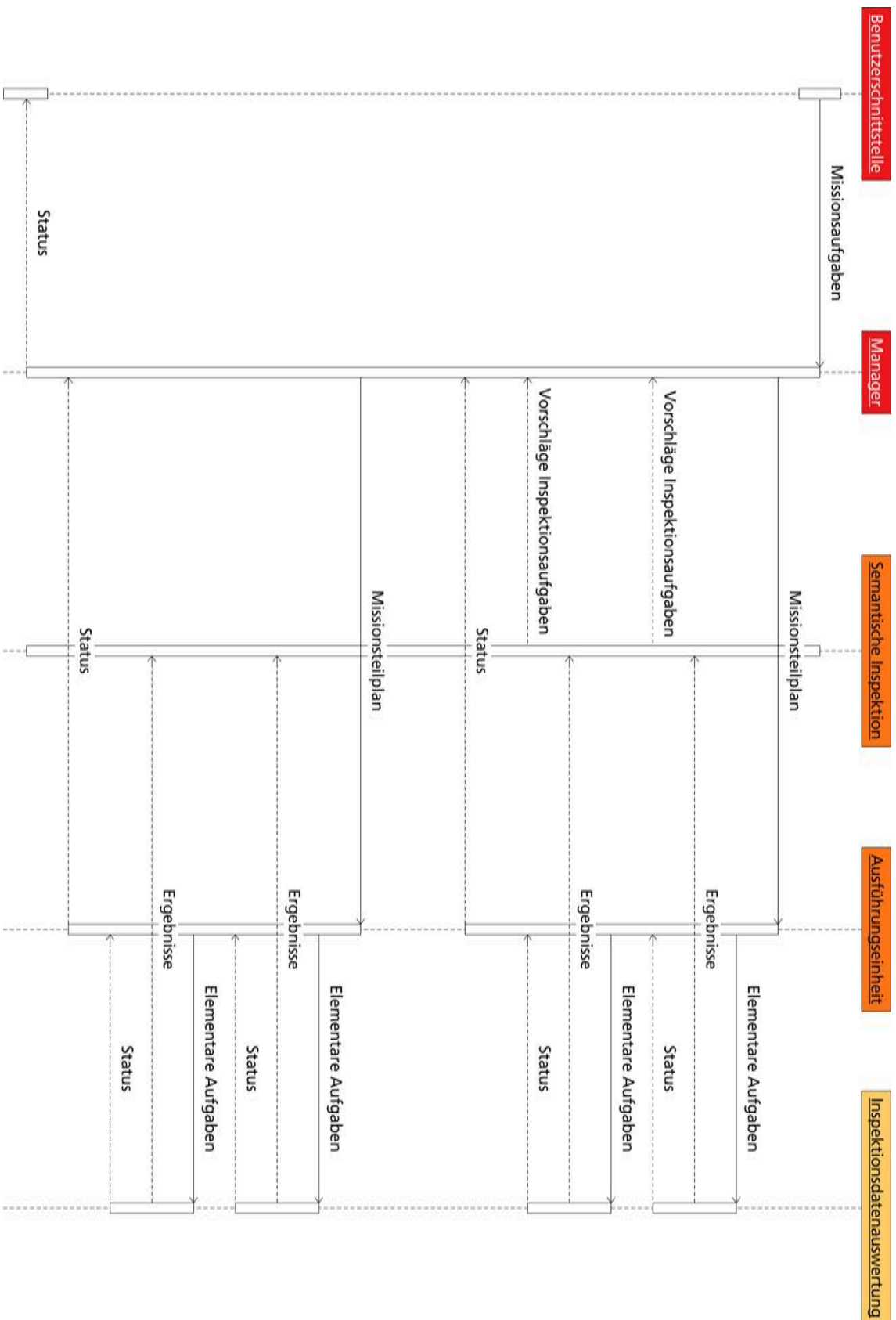


Abb. 3.5.: UML-Sequenzdiagramm zur Veranschaulichung des Ablaufs bei der aktiven Untersuchung interessierender Entitäten.

steuerung weist eine gewisse Ähnlichkeit zum globalen Weltmodell der NASREM-Architektur bzw. des RCS-Systems auf.

Der zentrale Unterschied zu den genannten Architekturen besteht jedoch in der durchgängig auf semantische Interoperabilität ausgelegten Struktur der semantischen Missionssteuerung. Sowohl der Sensor- als auch der Kontrolldatenfluss sind vollständig darauf ausgerichtet, die in der Wissensbasis definierten Konzepte in den Sensordaten und in den Aktuatorbefehlen zu verankern. Die Bindung der einzelnen Komponenten an die zentrale Wissensbasis stellt zudem sicher, dass das gemeinsame Vokabular in einer kohärenten und konsistenten Art und Weise verwendet wird, und trägt auf diese Weise zu einer Erhöhung der Transparenz des Systems bei. Durch die explizite Repräsentation des Wissens ergeben sich darüber hinaus große Vorteile hinsichtlich der Einfachheit der Wartung, der Erweiterbarkeit sowie der Flexibilität.

### **3.4. Zusammenfassung**

In diesem Kapitel wurden zunächst die Anforderungen an eine Steuerungsarchitektur für autonome Inspektionsroboter beschrieben. Anschließend wurde die im Rahmen dieser Arbeit entwickelte Architektur der semantischen Missionssteuerung vorgestellt. Die Architektur besitzt einen modularen, hierarchischen Aufbau und gliedert sich in Abhängigkeit von der Art der verarbeiteten Daten in vier Ebenen. Die einzelnen Ebenen lassen sich wiederum zwei Schichten zuordnen: einer funktionalen Schicht und einer Planungs-, Ausführungs- und Entscheidungsschicht. Nachfolgend wurden die einzelnen Komponenten der Architektur vorgestellt und ihre Aufgaben erläutert. Darüber hinaus wurde anhand von zwei Abläufen das Zusammenwirken der verschiedenen Komponenten verdeutlicht. Abschließend wurde die entworfene Architektur im Hinblick auf bestehende Ansätze eingeordnet.



## 4. Entwurf der Wissensbasis

Bei der Durchführung von Inspektionen und der Auswertung der entsprechenden Daten wird von menschlichen Experten auf umfangreiches Fachwissen sowie Hintergrundwissen und gemachte Erfahrungen zurückgegriffen. Für die autonome Durchführung von Inspektionsmissionen mit Servicerobotern muss dieses Wissen dem System in einer maschinell verarbeitbaren und nutzbaren Form zur Verfügung gestellt werden. Hierfür sind eine geeignete Form der Wissensrepräsentation sowie ein geeigneter Inferenzmechanismus auszuwählen. Ein sauber definiertes Vokabular mit einer klar definierten Bedeutung vereinfacht darüber hinaus die Kommunikation und Interaktion mit dem System, da sowohl der Anwender als auch der Roboter die „gleiche Sprache sprechen“.

In diesem Kapitel werden zunächst die Anforderungen an eine Wissensbasis für autonome Inspektionsroboter beschrieben. Hierfür werden unter anderem exemplarische Kompetenzfragen definiert, welche mit Hilfe der Wissensbasis beantwortet werden können sollen. Daran anschließend wird das im Rahmen dieser Arbeit entwickelte Konzept der Wissensbasis vorgestellt. Dazu wird zunächst ein Überblick über den strukturellen Aufbau der Wissensbasis gegeben. Anschließend wird näher auf das in den einzelnen Ontologien modellierte Wissen eingegangen.

### 4.1. Anforderungen

Die Wissensbasis eines autonomen Inspektionsroboters sollte das für die Durchführung von autonomen Inspektionsmissionen benötigte Fach- und Expertenwissen enthalten. Dieses gliedert sich in die folgenden Teilbereiche:

- Modellierung der Eigenschaften und Fähigkeiten des Inspektionsroboters.
- Modellierung der Umgebung, in welcher der Inspektionsroboter eingesetzt werden soll.
- Modellierung des für das Auffinden, Untersuchen und Beurteilen von interessierenden Entitäten notwendigen Wissens.
- Modellierung des für das Navigieren in der Umgebung notwendigen Wissens.
- Modellierung des für die Vorgabe von Inspektionsaufgaben sowie die Erzeugung und Ausführung von Missionsplänen notwendigen Wissens.

Allgemeine grundlegende Konzepte und Relationen sowie die zentralen Konzepte und Relationen der einzelnen Wissensbereiche sollten dabei zur Modellierung verschiedener Inspektionsroboter und verschiedener Inspektionsdomänen genutzt werden können. Dies erfordert einen klar strukturierten, modularen und erweiterbaren Aufbau der Wissensbasis. Neben der Modellierung von Konzepten, Relationen und Fakten sollte auch Regelwissen für die Auswertung und Bewertung der Inspektionsdaten sowie die Entscheidungsfindung modelliert werden können. Aufgrund der begrenzten Rechenkapazitäten an Bord von Inspektionsrobotern sollte die Wissensbasis insgesamt möglichst schlank und kompakt aufgebaut sein und nur die für die Durchführung von autonomen Inspektionsmissionen unbedingt notwendigen Konzepte und Relationen umfassen.

### 4.1.1. Kompetenzfragen

Zur Verdeutlichung der Anforderungen an die Wissensbasis eines autonomen Inspektionsroboters werden im Folgenden einige exemplarische Kompetenzfragen definiert, welche mit Hilfe der Wissensbasis beantwortet werden können sollen. Kompetenzfragen begleiten die Erstellung von Ontologien über alle Phasen hinweg und lassen sich nach [94] in die folgenden Kategorien einteilen:

**Strukturfragen** Fragen nach einfachen strukturellen Zusammenhängen zwischen Konzepten.

**Existenzfragen** Fragen zur Überprüfung der Modellierung einzelner Konzepte auf Vollständigkeit.

**Definitionsfragen** Fragen der Form „Was ist ...?“, die direkt von der Ontologie beantwortet werden können.

**Instanzfragen** Fragen zur Ermittlung und Beschreibung einzelner Instanzen.

**Komplexere Mischfragen** Fragen nach syntaktischen und semantischen Eigenschaften, die sich aus den obigen Fragetypen zusammensetzen.

Für die oben genannten Wissensbereiche werden im Folgenden beispielhafte Kompetenzfragen aus den einzelnen Kategorien angegeben:

#### Eigenschaften und Fähigkeiten des Inspektionsroboters

- Welche Arten von Robotern gibt es?
- Gibt es Informationen über die aktuell verbleibenden Ressourcen eines Roboters?
- Was ist ein Inspektionsroboter?

- Über welche Sensoren verfügt ein bestimmter Roboter?
- Wie viele Sensordaten eines bestimmten Typs können noch aufgezeichnet werden?

#### **Auffinden, Untersuchen und Beurteilen von interessierenden Entitäten**

- Welche Klassen von interessierenden Entitäten gibt es?
- Enthält eine Klasse von interessierenden Entitäten Informationen über charakteristische Merkmale?
- Was ist eine interessierende Entität?
- Welche interessierenden Entitäten wurden während der Inspektion gefunden?
- Wie sollte eine bestimmte interessierende Entität weiter untersucht werden?

#### **Erzeugung und Ausführung von Missionsplänen**

- Welche Arten von Aufgaben gibt es?
- Enthält eine Aufgabe Informationen darüber, unter welchen Bedingungen sie erfüllt werden kann?
- Was ist ein Planungsproblem?
- Welche Aufgaben kann ein bestimmter Inspektionsroboter direkt ausführen?
- Wie hoch ist der Ressourcenverbrauch zur Erfüllung einer bestimmten Aufgabe?

### **4.2. Struktureller Aufbau der Wissensbasis**

Gemäß den Anforderungen wurde für den Aufbau der Wissensbasis eine modulare, hierarchische Struktur gewählt [112]. Die Wissensbasis gliedert sich in drei Abstraktionsebenen und besitzt in Anlehnung an [94] einen pyramidenförmigen Aufbau (siehe Abbildung 4.1). Die oberste Ebene wird von der so genannten *Basisontologie* gebildet. Auf der mittleren Ebene befindet sich die so genannte *Kernontologie* und die unterste Ebene besteht aus der so genannten *Domänenontologie*. Im Folgenden werden die einzelnen Ebenen kurz erläutert.

**Basisontologie** Die *Basisontologie* enthält allgemeine, grundlegende Konzepte und Relationen, welche in den darunter liegenden Ebenen zur Definition komplexerer Konzepte und Relationen genutzt werden. Sie gliedert sich in kleinere, thematisch zusammengehörige Unterontologien, die in Abschnitt 4.3.1 detaillierter erläutert werden.

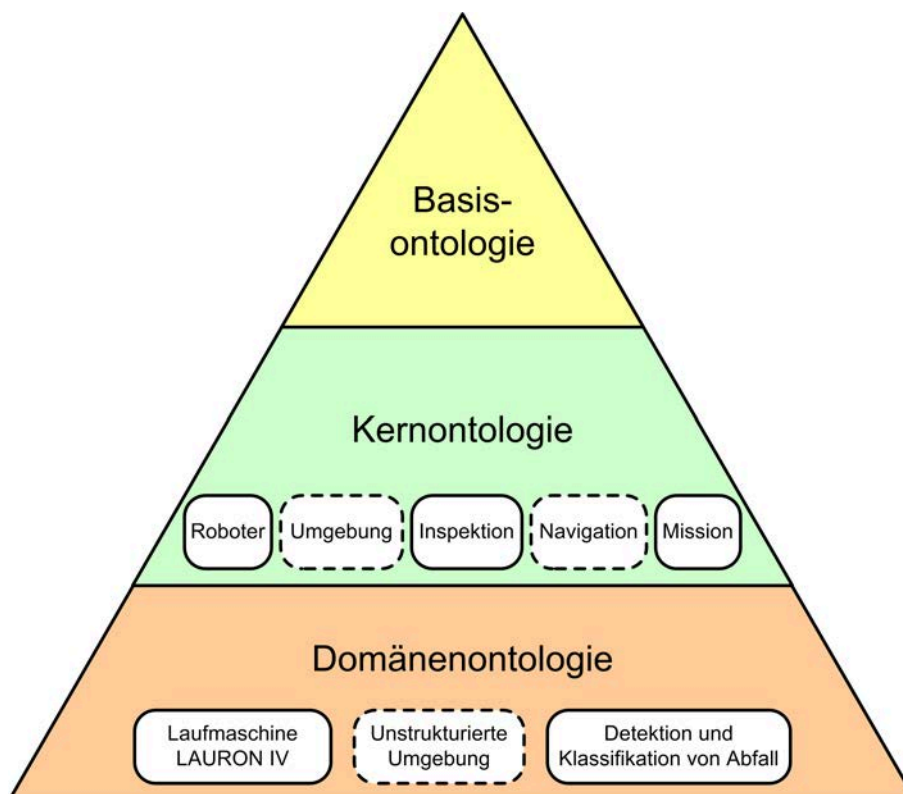


Abb. 4.1.: Der strukturelle Aufbau der Wissensbasis. Die gestrichelt umrandeten Unterontologien sind nicht Gegenstand dieser Arbeit, sondern werden im Rahmen einer parallel durchgeführten Arbeit entwickelt (siehe [101]).

**Kernontologie** Die *Kernontologie* dient der Modellierung der zentralen Konzepte und Relationen aller Wissensgebiete, die für die Durchführung von autonomen Inspektionsmissionen relevant sind. Hierfür gliedert sich die Kernontologie in die Unterontologien *Roboter*, *Umgebung*, *Inspektion*, *Navigation* und *Mission*. Das in den einzelnen Unterontologien modellierte Wissen wird in Abschnitt 4.3.2 vorgestellt. Es wird in der Domänenontologie verwendet, um das benötigte Wissen für die jeweilige applikationsspezifische Inspektionsdomäne zu definieren.

**Domänenontologie** Die *Domänenontologie* enthält das für den jeweiligen Inspektionsroboter, die jeweilige Inspektionsumgebung und die jeweiligen aufzufindenden interessierenden Entitäten spezifische Wissen. In Abbildung 4.1 sind hierfür exemplarisch die Unterontologien *Laufmaschine LAURON IV*, *Unstrukturierte Umgebung* und *Detektion und Klassifikation von Abfall* eingezeichnet, welche das entsprechende Wissen für das im Rahmen dieser Arbeit betrachtete Inspektionsszenario modellieren. Die Beschreibung des in den einzelnen Unterontologien modellierten Wissens erfolgt im Wesentlichen in Kapitel 7.

Die einzelnen Unterontologien auf den verschiedenen Ebenen sind zum Teil ebenfalls hierarchisch angeordnet, da die Konzepte und Relationen einiger Unterontologien in anderen Un-



terontologien zur Definition aggregierter Konzepte und Relationen verwendet werden. Die sich hierdurch ergebenden Abhängigkeiten werden durch eine hierarchische Anordnung aufgelöst, auf deren Darstellung in Abbildung 4.1 aus Gründen der Übersichtlichkeit verzichtet wurde. In Abschnitt 4.3 wird jedoch an den entsprechenden Stellen darauf hingewiesen.

Die Unterontologien *Umgebung*, *Navigation* und *Unstrukturierte Umgebung* sind nicht Gegenstand dieser Arbeit, sondern werden im Rahmen einer parallel laufenden Arbeit entwickelt (siehe [101]). Sie werden jedoch der Vollständigkeit halber mit aufgeführt, um das Gesamtkonzept zu verdeutlichen.

### 4.3. Modelliertes Wissen

Im Folgenden wird das in den einzelnen Ontologien modellierte Wissen anhand von ausgewählten Konzepten und Relationen näher erläutert.

#### 4.3.1. Basisontologie

Die *Basisontologie* enthält allgemeine grundlegende Konzepte und Relationen, die in der Kern- und der Domänenontologie verwendet werden, um komplexere Konzepte und Relationen zu definieren. Sie gliedert sich in kleinere Unterontologien, welche jeweils ein bestimmtes Themenfeld abdecken. So existieren etwa Unterontologien für die Beschreibung von Hardwarekomponenten, wie zum Beispiel Sensoren (siehe Abbildung 4.2), und für die Verwendung von Elementen der Prädikatenlogik (siehe Abbildung 4.3). Weitere Unterontologien dienen der Verwendung von Parametern (siehe Abbildung 4.8) und Verknüpfungen (siehe Abbildung 4.9) sowie der Beschreibung von Ressourcen (siehe Abbildung 4.4).

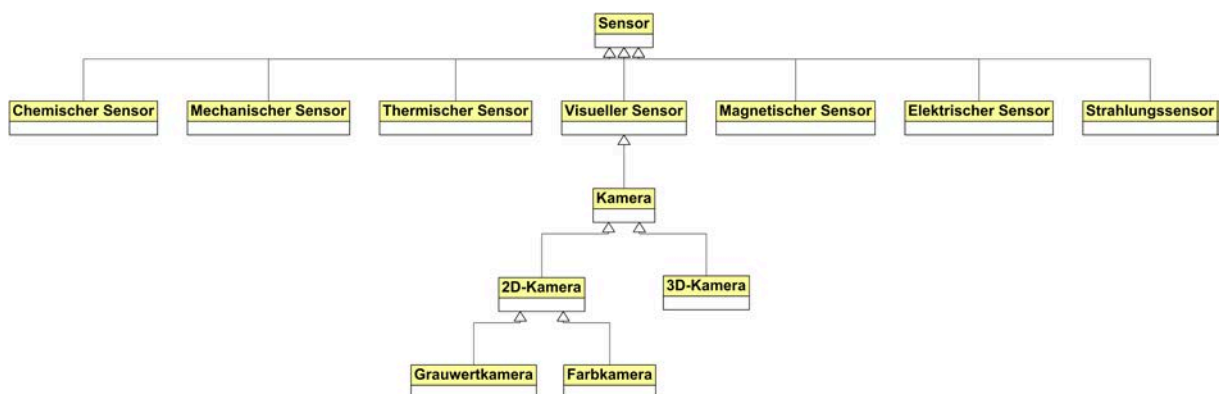


Abb. 4.2.: UML-Strukturdiagramm eines Ausschnitts der Sensortaxonomie.

#### 4.3.2. Kernontologie

Die *Kernontologie* enthält zentrale Konzepte und Relationen aus verschiedenen Wissensbereichen, welche für die Durchführung von autonomen Inspektionsmissionen benötigt werden. Sie

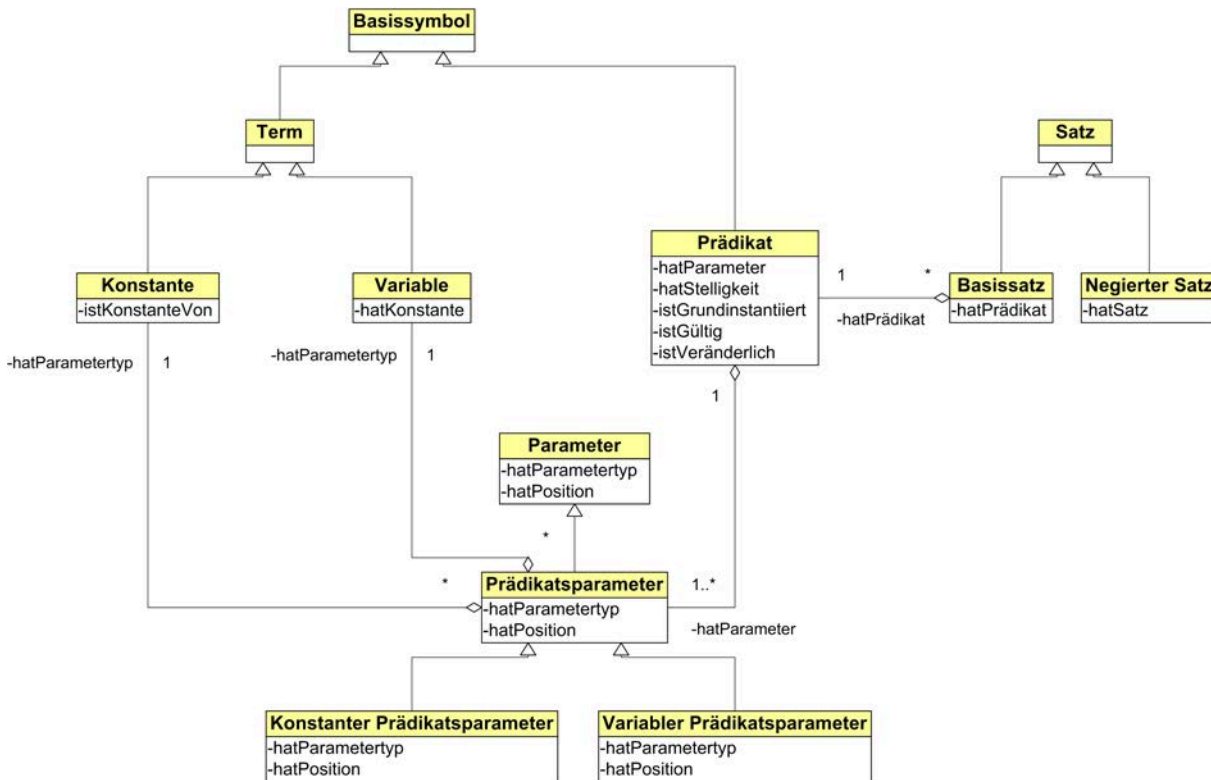


Abb. 4.3.: UML-Strukturdiagramm der Konzepte für Elemente der Prädikatenlogik.

besteht im Wesentlichen aus den Unterontologien *Roboter*, *Umgebung*, *Inspektion*, *Navigation* und *Mission*, die im Folgenden detaillierter beschrieben werden.

## Roboter

Die Kernontologie *Roboter* enthält Konzepte und Relationen zur Beschreibung der Eigenschaften und Fähigkeiten eines Roboters (siehe auch [65]). Dies umfasst unter anderem den mechanischen Aufbau des Roboters sowie die verfügbaren Sensoren und Aktoren. Darüber hinaus werden die benötigten Aspekte der Rechnerarchitektur des Roboters sowie die ihm zur Verfügung stehenden Ressourcen, insbesondere die für autonome Roboter wichtige Energieversor-

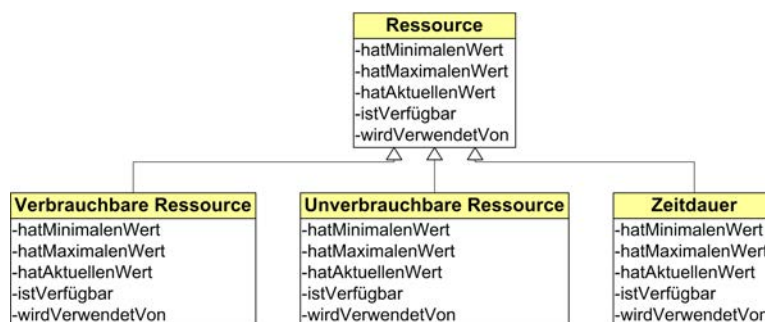


Abb. 4.4.: UML-Strukturdiagramm der Konzepte für Ressourcen.

gung, modelliert. Neben den technischen Daten und Eigenschaften des Roboters sind jedoch vor allem die Fähigkeiten der reaktiven bzw. verhaltensbasierten Basissteuerung von zentraler Bedeutung, da von dieser in der Regel ein elementares Verhaltensrepertoire für die Lokomotion, Perzeption, Lokalisierung und Navigation zur Verfügung gestellt wird, welches die Grundlage für komplexere Fähigkeiten und Verhalten bildet.

## Umgebung

Die Kernontologie *Umgebung* enthält die zur Beschreibung der Umgebung des Roboters benötigten Konzepte und Relationen. Diese erlauben es, die für den Roboter relevanten Eigenschaften der Umgebung geeignet zu repräsentieren und für die Lokomotion, Lokalisation, Navigation und Inspektion nutzbar zu machen. Entsprechend den Anforderungen und der Verwendung können die zu repräsentierenden Eigenschaften dabei auf geometrischer, topologischer oder semantischer Ebene beschrieben werden. Die Repräsentation auf semantischer Ebene erfolgt dabei mit Hilfe von so genannten *semantischen Karten* (siehe [101]).

## Inspektion

Die Kernontologie *Inspektion* enthält das zur Beschreibung von Inspektionsdaten und interessierenden Entitäten benötigte Wissen. Wie in Kapitel 3 beschrieben, folgt die Erkennung interessierender Entitäten in der Regel einem generischen Ablauf bestehend aus Sensoraufnahme, Sensordatenvorverarbeitung, Merkmalsberechnung, Klassifikation, Zuordnung, temporaler Fusion sowie Bewertung und Entscheidungsfindung. Die Ergebnisse der einzelnen Verarbeitungsschritte werden durch entsprechende Konzepte und Relationen beschrieben, die vom allgemeinen Konzept *Datentyp* abgeleitet sind (siehe Abbildung 4.5). Am Anfang der Verarbeitungskette steht das in der Kernontologie *Roboter* definierte Konzept der *Sensoraufnahme*. Es charakterisiert sowohl das Ergebnis der Sensoraufnahme als auch das Ergebnis der Sensordatenvorverarbeitung. Das Ergebnis der Segmentierung wird durch das Konzept *Interessierende Region* (engl. *Region of Interest (ROI)*) beschrieben. Es enthält neben einem eindeutigen Bezeichner und einem Zeitstempel einen Verweis auf die entsprechenden Sensoraufnahmen, die zur Segmentierung verwendet wurden. Darüber hinaus wird für jede interessierende Region die zugehörige Position in Weltkoordinaten gespeichert. Für die einzelnen interessierenden Regionen werden anschließend Merkmalsvektoren berechnet, die durch das Konzept *Merkmalsvektor* beschrieben werden. Neben einem eindeutigen Bezeichner und einem Zeitstempel enthalten diese hierfür einen entsprechenden Verweis auf die zugehörige interessierende Region. Die Merkmalsvektoren ihrerseits dienen wiederum als Eingabegröße für die Klassifikation. Das Ergebnis der Klassifikation wird durch das Konzept *Klassifikationsergebnis* beschrieben. Dieses enthält neben einem eindeutigen Bezeichner und einem Zeitstempel einen Verweis auf den entsprechenden Merkmalsvektor sowie eine Liste von Hypothesen, welche durch das gleichnamige

Konzept beschrieben werden. Das Konzept *Hypothese* wird durch die Wahrscheinlichkeit für eine bestimmte Klasse charakterisiert. Nach der Klassifikation erfolgt die Zuordnung zu einer gegebenenfalls bereits untersuchten interessierenden Entität bzw. die Erzeugung einer neuen interessierenden Entität. Interessierende Entitäten werden ebenfalls durch ein entsprechendes Konzept beschrieben. Wie die Ergebnisse der vorangegangenen Verarbeitungsschritte beinhaltet auch das Konzept *Interessierende Entität* einen eindeutigen Bezeichner und einen Zeitstempel. Des Weiteren enthält es einen Verweis auf das entsprechende Klassifikationsergebnis. Die Zuordnung zu einer bereits untersuchten interessierenden Entität wird mit Hilfe einer entsprechenden Vorgängerrelation modelliert. Jede interessierende Entität verfügt darüber hinaus über eine Liste von Hypothesen, welche die Wahrscheinlichkeiten für die Zugehörigkeit zu einer Entitätsklasse angeben und das Ergebnis der temporalen Fusion bilden. Interessierende Entitäten werden zudem durch die Konfidenz der fusionierten Datenauswertungsergebnisse und den Inspektionsstatus beschrieben, welcher angibt, ob die Untersuchung fortgesetzt werden soll. Die gegebenenfalls zur weiteren Untersuchung der interessierenden Entität ausgewählte Inspektionsaufgabe sowie deren Priorität sind ebenfalls entsprechend modelliert. Die Inspektionsaufgabe wird dabei durch das in der Kernontologie *Mission* beschriebene Konzept *Aufgabe* charakterisiert (siehe unten). Das Konzept der *Entitätsklasse* beschreibt die Klassen von interessierenden Entitäten, die während der Inspektion gefunden und erkannt werden sollen. Zur Beschreibung der charakteristischen Eigenschaften der jeweiligen Klasse steht die Entitätsklasse in Relation zum Konzept *Merkmal*, welches die entsprechenden Merkmale beschreibt. Darüber hinaus können für eine Entitätsklasse *Ursachen* angegeben und Auftretensorte spezifiziert werden, an denen interessierende Entitäten dieser Klasse üblicherweise vorkommen. Des Weiteren kann jede Entitätsklasse mit einer Priorität versehen werden und es können Verwechslungsmöglichkeiten mit anderen Entitätsklassen angegeben werden.

Wie in Kapitel 7 beschrieben, kann die Bestimmung der Entitätsklasse einer interessierenden Entität auf zwei unterschiedliche Arten erfolgen. Zum einen kann ein subsymbolischer Klassifikator verwendet werden, welcher direkt die wahrscheinlichste Entitätsklasse als Ergebnis liefert. Zum anderen kann auch eine wissensbasierte Klassifikation durchgeführt werden. Hierfür werden spezielle Klassifikatoren verwendet, die zunächst die beobachtbaren Merkmale einer interessierenden Entität einzeln bestimmen. Basierend auf der Menge dieser beobachtbaren Merkmale kann anschließend eine regelbasierte Bestimmung der möglichen Zugehörigkeit zu den einzelnen Entitätsklassen erfolgen. Für diese zweite Möglichkeit enthält das Konzept *Interessierende Entität* jeweils eine Liste der beobachtbaren Merkmale sowie eine Liste von Entitätsklassen, zu denen die interessierende Entität möglicherweise gehört.

Zur Repräsentation der Ergebnisse der unterschiedlichen Ausprägungen der einzelnen Verarbeitungsschritte werden die einzelnen Konzepte in der Domänenontologie entsprechend spezialisiert.

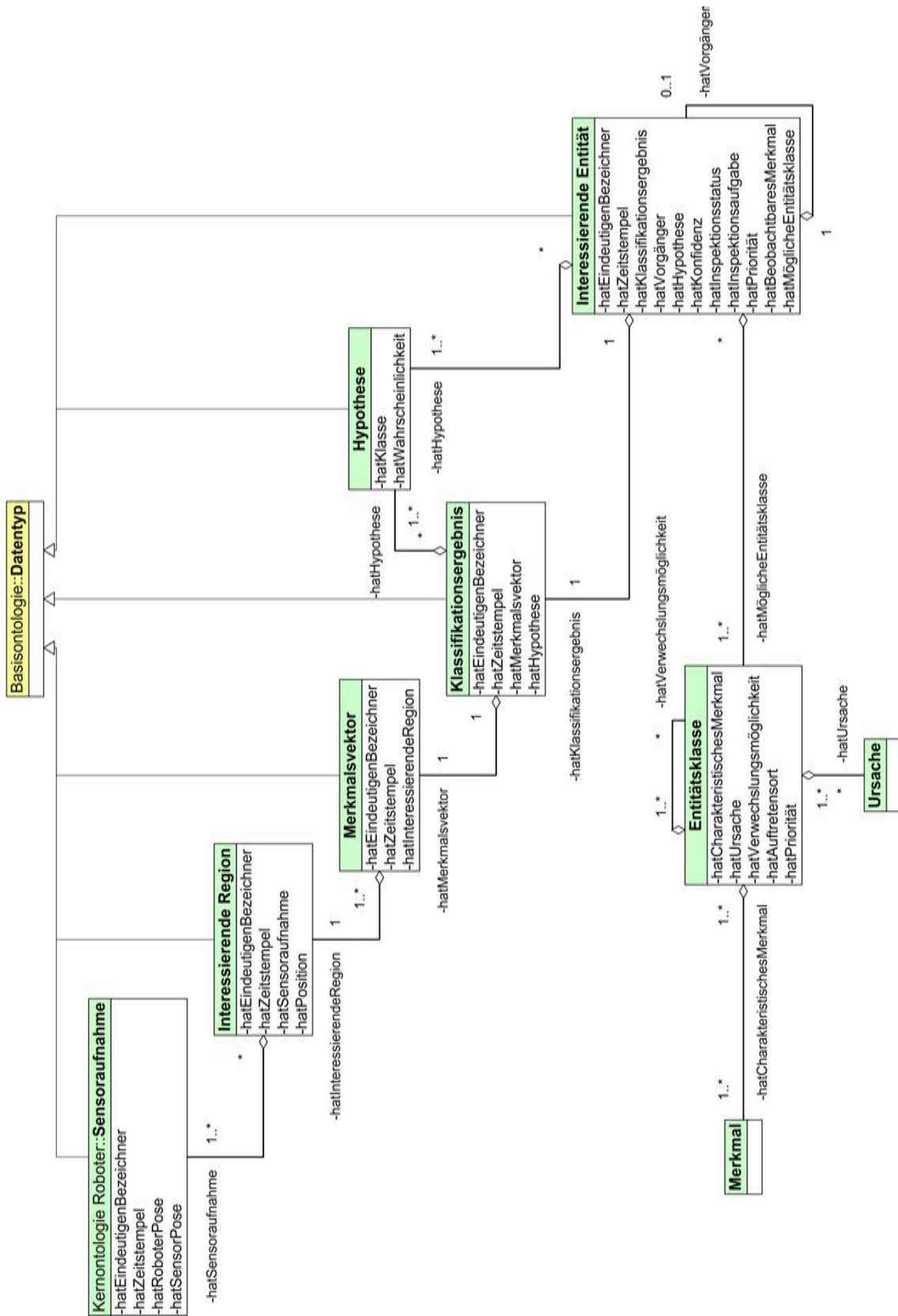


Abb. 4.5.: UML-Strukturdiagramm der wichtigsten Konzepte und Relationen der Kernontologie *Inspektion*.

### Navigation

Die Kernontologie *Navigation* enthält das für die Lokalisation und Navigation benötigte Wissen. Die Ergebnisse der einzelnen Verarbeitungsschritte der Navigationsdatenauswertung werden dabei ähnlich wie bei der Inspektionsdatenauswertung durch geeignete Konzepte und Relationen beschrieben. Darüber hinaus werden geeignete Konzepte und Relationen zur Repräsentation von Pfaden und Routen definiert. Hierfür wird auf entsprechende Konzepte und Relationen aus der Kernontologie *Umgebung* zurückgegriffen.

### Mission

Die Kernontologie *Mission* enthält das zur Vorgabe von Inspektionsaufgaben sowie zur Erzeugung und Ausführung von Missionsplänen benötigte Wissen. Da für die Erzeugung und Ausführung von Plänen ein auf Hierarchischen Aufgabennetzwerken (engl. *Hierarchical Task Networks* (HTNs)) beruhendes Planungsverfahren zum Einsatz kommt (siehe Kapitel 5), umfasst die Kernontologie *Mission* die für das Planen mit HTNs wesentlichen Konzepte und Relationen. Die Modellierung orientiert sich dabei an den in Abschnitt 2.3.1 angegebenen Definitionen. Das modellierte Wissen ist dadurch weitgehend vom konkret verwendeten Planer unabhängig und kann somit prinzipiell auch von anderen HTN-Planern genutzt werden.

Das zentrale Konzept der Kernontologie *Mission* ist das so genannte *Planungsproblem* (siehe Abbildung 4.6). Es enthält eine Beschreibung der Planungsdomäne, des aktuellen Zustands der Welt, des auszuführenden Aufgabennetzwerks sowie von Planungsparametern. Mit Hilfe des Konzepts *Planungsparameter* können eine *Planungsstrategie* vorgegeben sowie entsprechende *Ressourcenbewertungen* für die gewählte Planungsstrategie angegeben werden. Das Konzept der *Planungsdomäne* enthält eine Liste der in der jeweiligen Domäne zur Verfügung stehenden Aufgaben und Zerlegungsmethoden. Der aktuelle Zustand der Welt wird mit Hilfe des Konzepts *Zustand* beschrieben, welches durch eine Liste von Prädikaten und die zur Verfügung stehenden Ressourcen charakterisiert ist. Das Konzept *Aufgabennetzwerk* enthält eine Liste von Missionsaufgaben, für die ein Plan erzeugt werden soll. *Missionsaufgaben* enthalten einen Verweis auf die jeweilige zu erfüllende *Aufgabe* und geben eine Priorität für diese an. Des Weiteren können *Parametersubstitutionen* für die zu erfüllende Aufgabe spezifiziert werden und es kann angegeben werden, ob es sich um eine optionale Aufgabe handelt. In Hierarchischen Aufgabennetzwerken wird zwischen zwei Arten von Aufgaben unterschieden: elementaren Aufgaben und nichtelementaren Aufgaben. Beide werden durch entsprechende Konzepte modelliert. Das Konzept *Elementare Aufgabe* wird durch Vor-, Während- und Nachbedingungen sowie durch Ressourcenbedingungen charakterisiert (siehe Abbildung 4.7). Darüber hinaus können elementare Aufgaben parametrisiert werden (siehe Abbildungen 4.8 und 4.9) und besitzen eine Verfügbarkeit. Das Konzept *Nichtelementare Aufgabe* verfügt über Verweise auf Zerlegungsmethoden, die zur Verfeinerung verwendet werden können. Nichtelementare Aufgaben besitzen ebenfalls

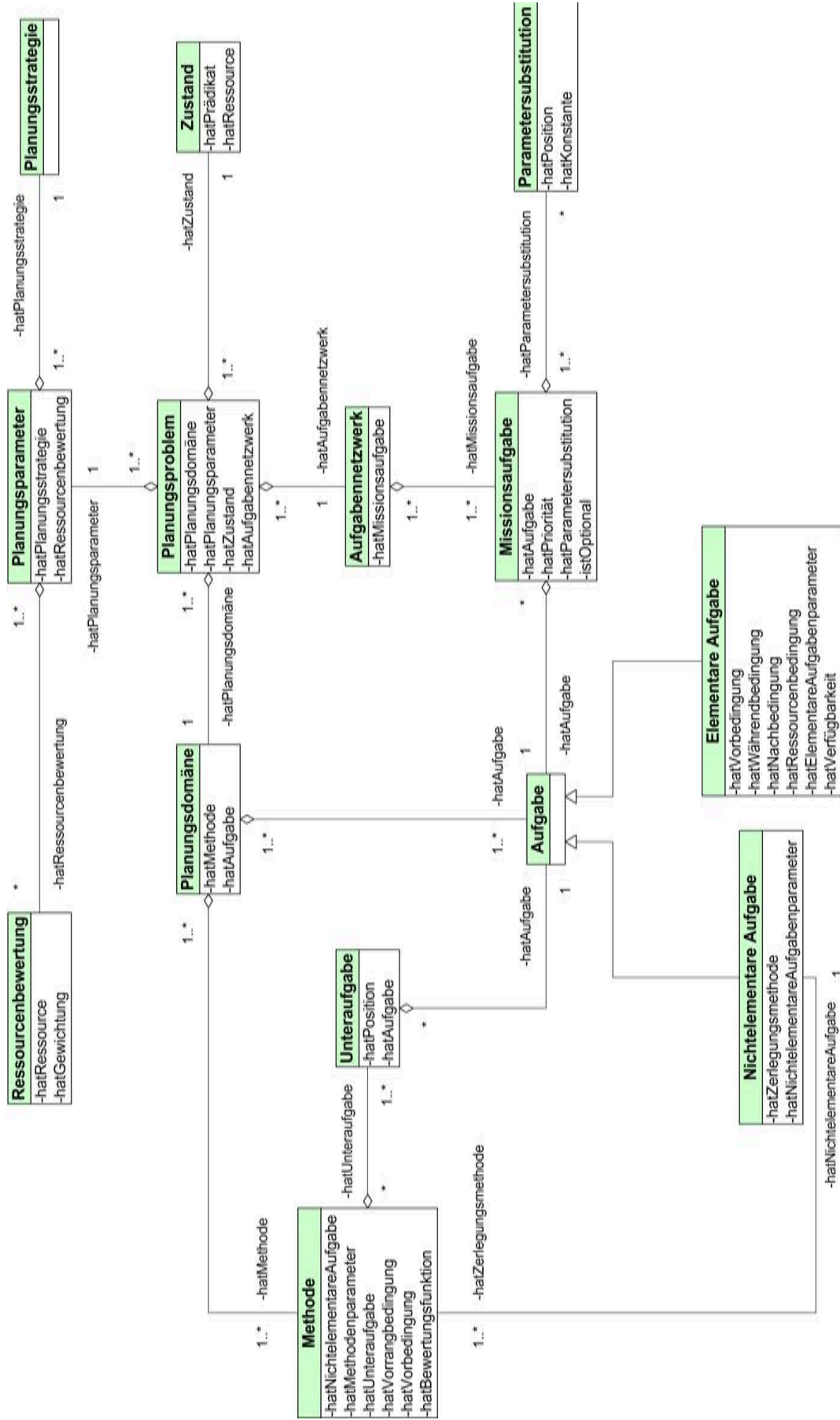


Abb. 4.6.: UML-Strukturdiagramm der wichtigsten Konzepte und Relationen der Kernontologie *Mission*.

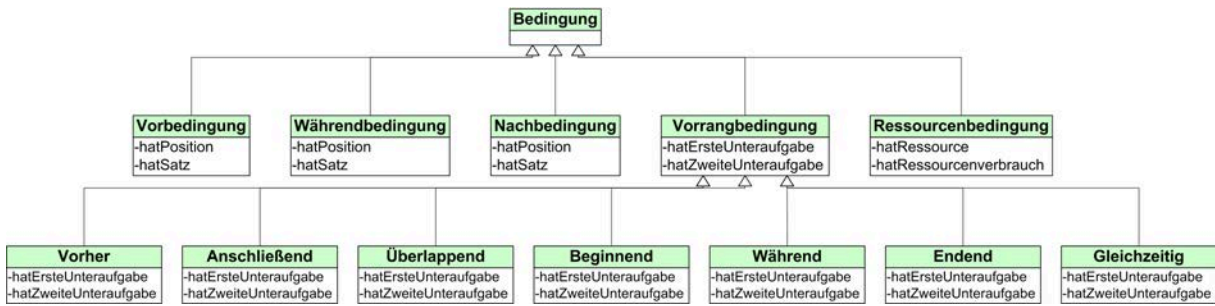


Abb. 4.7.: UML-Strukturdiagramm der Konzepte für Bedingungen.

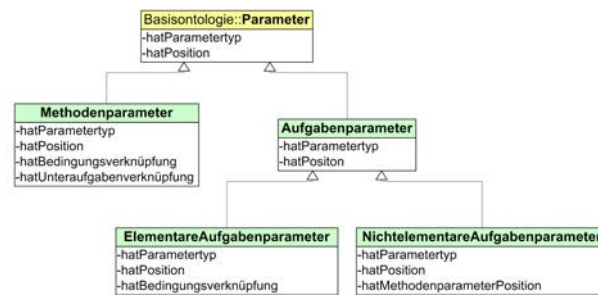


Abb. 4.8.: UML-Strukturdiagramm der Konzepte für Parameter.

Parameter, welche an die entsprechenden Zerlegungsmethoden weitergegeben werden. Zerlegungsmethoden werden durch das Konzept *Methode* modelliert. Eine Methode verfügt über Vorbedingungen, welche die Anwendbarkeit der Methode spezifizieren, sowie über eine Menge von *Unteraufgaben*. Dabei können jeweils für Paare von Unteraufgaben Vorrang-Bedingungen definiert werden. Weiterhin besitzen Methoden einen Verweis auf die nichtelementare Aufgabe, welche mit Hilfe der Methode zerlegt werden kann, sowie entsprechende Parameter. Jede Methode verfügt zudem über eine Bewertungsfunktion, mit deren Hilfe zwischen verschiedenen anwendbaren Methoden zur Zerlegung einer nichtelementaren Aufgabe ausgewählt werden kann. Auf die Bedeutung und Verwendung der einzelnen Konzepte wird in Kapitel 5 detaillierter eingegangen.

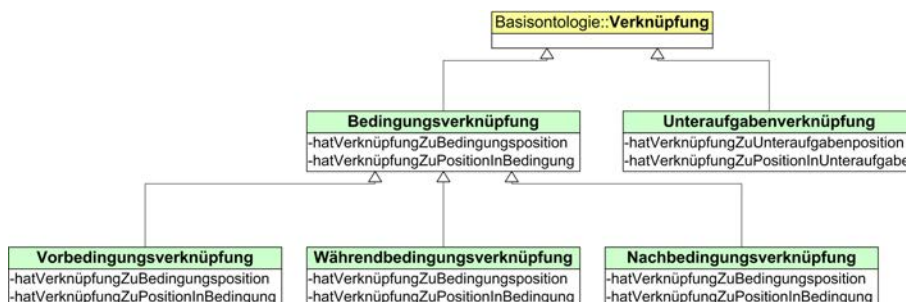


Abb. 4.9.: UML-Strukturdiagramm der Konzepte für Verknüpfungen.



### 4.3.3. Domänenontologie

Die *Domänenontologie* enthält das spezifische Wissen über den konkret verwendeten Inspektionsroboter (siehe [65]), die konkret zu inspizierende Umgebung sowie die konkreten Entitätsklassen, die gefunden und untersucht werden sollen. Sie gliedert sich hierfür in entsprechende Unterontologien. Für das im Rahmen dieser Arbeit gewählte Inspektionsszenario (siehe Kapitel 7) sind dies die Unterontologien *Laufmaschine LAURON IV*, *Unstrukturierte Umgebung* und *Detektion und Klassifikation von Abfall*. In der Domänenontologie werden überwiegend Instanzen der in der Basis- und in der Kernontologie definierten Konzepte modelliert und mit den entsprechenden Relationen zueinander in Beziehung gesetzt. Darüber hinaus enthält die Domänenontologie einen Großteil der Regelbasis des semantischen Inspektionsmodells. Eine detaillierte Beschreibung der modellierten Instanzen und Regeln für das gewählte Inspektionsszenario erfolgt in Kapitel 7. Als Beispiele für die in der Unterontologie *Detektion und Klassifikation von Abfall* definierten Konzepte sind in Abbildung 4.10 die Merkmalsklassen zur Charakterisierung von Abfall und in Abbildung 4.11 die Abfall-Entitätsklassen dargestellt.

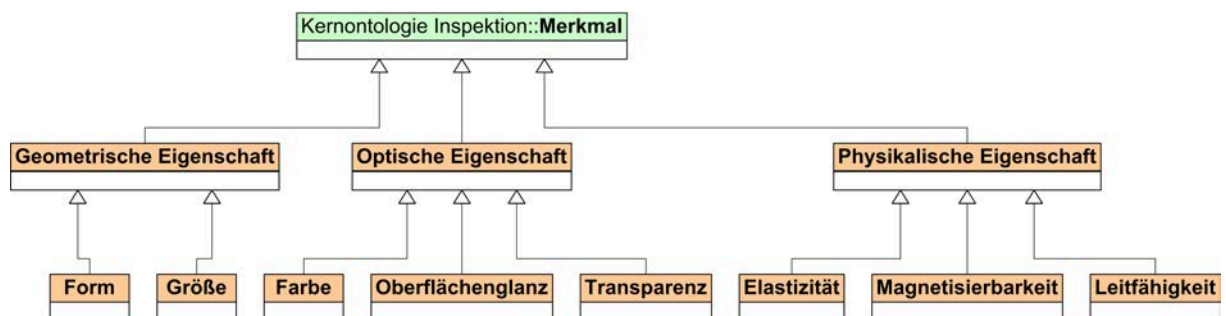


Abb. 4.10.: UML-Strukturdiagramm der Merkmalsklassen zur Charakterisierung von Abfall.

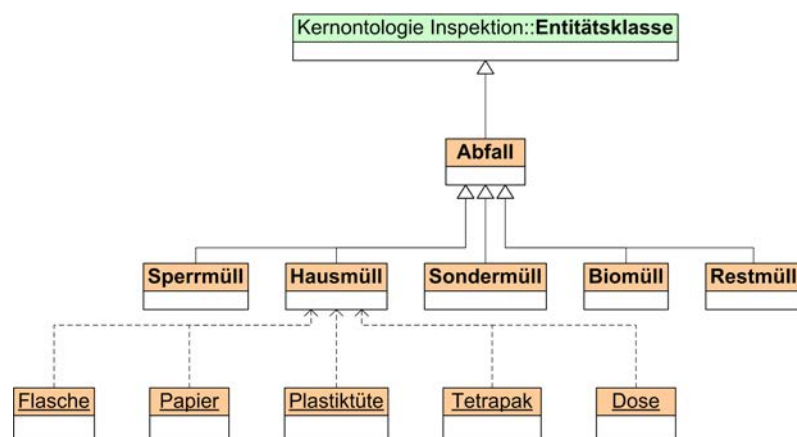


Abb. 4.11.: UML-Strukturdiagramm der Abfall-Entitätsklassen.

### 4.4. Einordnung der Wissensbasis

Im Rahmen dieser Arbeit wurde in Anlehnung an [94] eine pyramidenförmige Struktur der Wissensbasis gewählt, da sie die Taxonomie der Konzepte und Relationen (die so genannte *terminologische Box* (TBox) bzw. *relationale Box* (RBox)) geeignet abbildet. In der Basisontologie auf der obersten Ebene werden die grundlegenden und allgemeingültigen Konzepte und Relationen definiert. Diese werden nachfolgend in der Kern- und Domänenontologie immer weiter spezialisiert und zur Definition immer komplexerer Konzepte und Relationen verwendet. Die konkreten Instanzen der Konzepte (die Individuen der so genannten *assertionalen Box* (ABox)) werden schließlich überwiegend in der Domänenontologie definiert. Die pyramidenförmige Struktur symbolisiert darüber hinaus die Anzahl der Konzepte, Relationen und Instanzen, welche von oben nach unten zunimmt (siehe Tabelle 7.4).

Die Basis- und die Kernontologie sind unabhängig vom konkret verwendeten Inspektionsroboter sowie der konkret betrachteten Inspektionsdomäne und sind daher wiederverwendbar. Da sie die zentralen Konzepte und Relationen für die Inspektion mit autonomen Robotersystemen enthalten, stellen sie zudem sicher, dass die grundlegenden Funktionsweisen der in Kapitel 3 vorgestellten Steuerungsarchitektur auch bei der Übertragung auf andere Inspektionsroboter und andere Inspektionsdomänen gewährleistet sind. Darüber hinaus tragen sie wesentlich dazu bei, dass die für eine solche Übertragung benötigten Aufwände relativ gering sind (siehe Abschnitt 8.6).

### 4.5. Zusammenfassung

In diesem Kapitel wurden zunächst die Anforderungen an eine Wissensbasis für autonome Inspektionsroboter beschrieben. Zur Verdeutlichung wurden exemplarische Kompetenzfragen für die einzelnen Wissensbereiche definiert, welche die Wissensbasis beantworten können soll. Anschließend wurde der strukturelle Aufbau der Wissensbasis erläutert. Es wurde ein modularer, hierarchischer Aufbau in Pyramidenform gewählt, der sich in drei Ebenen gliedert. Auf der obersten Ebene befindet sich die so genannte *Basisontologie*, welche allgemeine und grundlegende Konzepte und Relationen definiert. Auf der mittleren Ebene befindet sich die so genannte *Kernontologie*, welche die zentralen Konzepte und Relationen der Wissensbereiche *Roboter*, *Umgebung*, *Inspektion*, *Navigation* und *Mission* beschreibt. Die unterste Ebene wird von der *Domänenontologie* gebildet, welche das Wissen über den konkreten Inspektionsroboter, die konkret zu inspizierende Umgebung und die konkret aufzufindenden interessierenden Entitäten modelliert. Abschließend wurde das in den einzelnen Ontologien modellierte Wissen genauer beschrieben und eine Einordnung des gewählten Ansatzes vorgenommen.

## 5. Erzeugung und Ausführung von Inspektionsplänen

Ein autonomer Inspektionsroboter soll für eine Menge vom Benutzer vorgegebener Missionsaufgaben autonom einen geeigneten Inspektionsplan erstellen und diesen ausführen. Dabei müssen sowohl temporale Randbedingungen als auch die dem System zur Verfügung stehenden Ressourcen beachtet werden. Außerdem muss der Roboter in einer dynamischen und nichtdeterministischen Umgebung agieren und soll gefundene interessierende Entitäten mit den zur Verfügung stehenden Mitteln aktiv untersuchen, bis eine ausreichende Konfidenz der Datenauswertungsergebnisse vorliegt. Hierfür müssen die bestehenden Inspektionspläne kontinuierlich um situativ generierte Inspektionsaufgaben erweitert und entsprechend angepasst werden. Zur Erstellung und Ausführung von Inspektionsplänen ist somit ein geeignetes Planungsverfahren erforderlich, welches den genannten komplexen Randbedingungen in einer angemessenen Weise gerecht wird.

In diesem Kapitel werden zunächst die Anforderungen an ein Planungsverfahren zur Erzeugung von Plänen für die Durchführung von autonomen Inspektionsmissionen beschrieben. Anschließend werden die im Rahmen dieser Arbeit zur Repräsentation von Inspektionsplänen entwickelten so genannten *Flexiblen Hierarchischen Pläne* (FHiP) vorgestellt. Darauf aufbauend wird das gewählte Planungsverfahren erläutert. Des Weiteren wird die Ausführung der generierten Pläne sowie die Behandlung von Fehler- und Ausnahmezuständen beschrieben. Abschließend wird eine Einordnung des Planungsverfahrens vorgenommen und auf potentielle Synergien zwischen Planung und Inferenz eingegangen.

### 5.1. Anforderungen

In diesem Abschnitt wird auf die Anforderungen an ein Planungsverfahren für autonome Inspektionsroboter eingegangen. Hierfür werden zunächst geeignete Kriterien zur Charakterisierung von Planungsverfahren vorgestellt. Anschließend werden mit Hilfe dieser Kriterien die Anforderungen definiert.

#### 5.1.1. Kriterien zur Charakterisierung von Planungsverfahren

In [72] werden typische Charakteristiken von Anwendungsdomänen, in denen Planung und Scheduling eingesetzt werden, genannt. Im Folgenden werden die einzelnen Charakteristiken kurz erläutert. Alle genannten Kriterien werden von der klassischen Planung ausgeschlossen (vergleiche etwa [40]).

**Dynamische Welt** Der Zustand der Welt kann sich dynamisch, d. h. durch Ereignisse und ohne Ausführung von Aktionen, verändern. Ereignisse entsprechen der internen Dynamik des Systems. Sie sollten bei der Planung berücksichtigt werden, können jedoch nicht vom Agenten kontrolliert oder ausgelöst werden.

**Mehrere Agenten** In der Welt agieren neben dem betrachteten Agenten weitere Agenten. Diese können sich sowohl kooperativ als auch unkooperativ verhalten. Ihr Verhalten sollte bei der Planung berücksichtigt werden.

**Unvollständige und unsichere Informationen** Die Informationen, die der Agent über den Zustand der Welt besitzt, sind mit Unsicherheiten behaftet und nicht vollständig.

**Externe Informationsquellen** Um Informationen über den Zustand der Welt zu erhalten, besteht die Notwendigkeit externe Informationsquellen (Sensoren, Datenbanken, Menschen) zu befragen.

**Zeitliche Dauer** Aktionen besitzen eine Ausführungsdauer und sind nicht instantan abgeschlossen.

**Zeitliche Beschränkungen** Die Ausführung von Aktionen kann zeitlichen Beschränkungen unterliegen.

**Überlappende Aktionen** Aktionen können sich überlappen und gleichzeitig ausgeführt werden.

**Asynchrone Aktionen** Aktionen können asynchron ausgeführt werden.

**Numerische Berechnungen** Bei der Planung können numerische Berechnungen durchgeführt werden, die Ressourcen, Wahrscheinlichkeiten oder geometrische und räumliche Beziehungen beinhalten.

**Ressourcen** Bei der Planung sind Ressourcen zu berücksichtigen.

**Inferenzregeln und abgeleitete Effekte** Es können Inferenzregeln spezifiziert werden, um Prädikate/Fakten zu schlussfolgern, die von keiner dem Planer zur Verfügung stehenden Aktion beeinflusst werden.

**Beschränkung von Zustandstrajektorien** Es können Bedingungen an die Trajektorien der Zustände definiert werden.

**Planmetriken** Es können Metriken definiert werden, die bei der Erzeugung und Auswahl von Plänen berücksichtigt werden.

**Weiche Ziele** Es können Ziele definiert werden, die nicht zwingend erfüllt werden müssen. Für den Fall, dass diese Ziele nicht erreicht werden, sind Bestrafungskosten definiert. Diese können entsprechend in den Planmetriken berücksichtigt werden.

**Präferenzen bezüglich Zustandstrajektorien** Für die Trajektorien der Zustände können Präferenzen definiert werden. Für den Fall, dass diese Präferenzen nicht eingehalten werden, sind Bestrafungskosten definiert. Diese können entsprechend in den Planmetriken berücksichtigt werden.

### 5.1.2. Bestimmung der Anforderungen

Basierend auf den oben genannten Kriterien werden im Folgenden die Anforderungen an die Erzeugung und Ausführung von Plänen für die autonome Durchführung von Inspektionsmissionen bestimmt.

Ein autonomer Inspektionsroboter muss in einer sich dynamisch verändernden Welt agieren können. Im Rahmen dieser Arbeit wird jedoch davon ausgegangen, dass es sich um einen einzelnen Inspektionsroboter handelt und sich somit in der zu inspizierenden Umgebung nur ein Agent befindet. Zur Erfassung des Zustands der Welt ist der Inspektionsroboter auf die Wahrnehmung seiner Umwelt mit Hilfe von Sensoren und somit auf externe Informationsquellen angewiesen. Da jegliche Sensormessungen mit Fehlern und Ungenauigkeiten behaftet sind, verfügt er somit nur über unvollständige und unsichere Informationen über die Welt. Aufgrund der Tatsache, dass ein autonomer Inspektionsroboter nur über begrenzte Ressourcen verfügt, müssen diese bei der Planung explizit berücksichtigt werden. Hierfür sind numerische Berechnungen bezüglich des Ressourcenverbrauchs notwendig. Ebenso ist zu berücksichtigen, dass die Ausführung von Aktionen unterschiedliche Zeitdauern in Anspruch nehmen kann. Die einzelnen während einer Inspektionsmission auszuführenden Aktionen unterliegen zudem zeitlichen Beschränkungen in Form von Ordnungsrelationen und sollten aus Gründen der Effizienz und Ressourcensparsamkeit nebenläufig und bei Bedarf asynchron ausgeführt werden können. Da für eine Menge von vorgegebenen Missionsaufgaben in der Regel eine Vielzahl von möglichen Lösungsplänen existieren, sollten die generierten Pläne anhand von geeigneten Planmetriken bewertet werden können, um bezogen auf die jeweilige Situation möglichst optimale Lösungen zu finden. Die einzelnen Inspektionsaufgaben sollten außerdem mit Prioritäten versehen werden können, welche die Wichtigkeit der entsprechenden Aufgaben angeben. Darüber hinaus sollte zwischen zwingend auszuführenden Aktionen (die zum Beispiel der Erhaltung der Funktionsfähigkeit des Systems dienen) und solchen, die wünschenswert sind, unterschieden werden. Diese so genannten weichen Ziele sollten entsprechend gekennzeichnet und bei der Planung und Ausführung je nach Verfügbarkeit der entsprechenden Ressourcen berücksichtigt werden. Da bei der autonomen Inspektion von komplexen Umgebungen primär das Ausführen bestimmter Inspektionsaufgaben im Vordergrund steht, werden zunächst keine Beschränkungen

gen oder Präferenzen bezüglich der zu durchlaufenden Zustandstrajektorien gefordert. Entsprechende Bedingungen und Präferenzen sollten vielmehr in Form von Ordnungsbedingungen für Inspektionsaufgaben angegeben werden. Aufgrund des im Rahmen dieser Arbeit gewählten wissensbasierten Ansatzes bietet es sich darüber hinaus an, die Wissensbasis und die dazugehörigen Inferenzmechanismen zu nutzen und Planung und Inferenz synergetisch zu vernetzen (zum Beispiel zur Bestimmung von abgeleiteten Effekten mit Hilfe von Inferenzregeln). Auf die sich aus der Vernetzung von Planung und Inferenz ergebenden potentiellen Synergien wird in Abschnitt 5.6 detaillierter eingegangen.

Der Übersichtlichkeit halber sind die Anforderungen nochmals in Tabelle 5.1 zusammengefasst.

Charakteristik	Anforderung
Dynamische Welt	✓
Mehrere Agenten	×
Unvollständige und unsichere Informationen	✓
Externe Informationsquellen	✓
Zeitliche Dauer	✓
Zeitliche Beschränkungen	✓
Überlappende Aktionen	✓
Asynchrone Aktionen	✓
Numerische Berechnungen	✓
Ressourcen	✓
Inferenzregeln und abgeleitete Effekte	✓
Beschränkung von Zustandstrajektorien	×
Planmetriken	✓
Weiche Ziele	✓
Präferenzen bezüglich Zustandstrajektorien	×

Tab. 5.1.: Anforderungen an den Planungsansatz.

## 5.2. Flexible Hierarchische Pläne

Zur Repräsentation von Inspektionsplänen wurden im Rahmen dieser Arbeit so genannte *Flexible Hierarchische Pläne* (FHiPs) entwickelt (siehe Abbildung 5.2), die sich am Planen mit Hierarchischen Aufgabennetzwerken (siehe Abschnitt 2.3.1) orientieren. Flexible Hierarchische Pläne basieren auf *Flexiblen Programmen* (FPs) (siehe Abschnitt 2.3.2) und *Concurrent Hierarchical Plans* (CHiPs) (siehe Abschnitt 2.3.3), kombinieren verschiedene Elemente der beiden Ansätze und erweitern sie um zusätzliche Komponenten. Formal ist ein Flexibler Hierarchischer Plan als 16-Tupel definiert und besteht aus den folgenden Elementen:

**ID** Eindeutige Identifikationsnummer des Flexiblen Hierarchischen Plans.

**Typ** Typ des Flexiblen Hierarchischen Plans. Dieser kann einen der Werte *Elementar*, *Oder* oder *Und* annehmen. Ein *elementarer* FHiP entspricht einer elementaren Aufgabe, ein *Oder*-FHiP einer nichtelementaren Aufgabe und ein *Und*-FHiP einer Methode des Planens mit Hierarchischen Aufgabennetzwerken.

**Name** Name des Individuums der entsprechenden elementaren Aufgabe, nichtelementaren Aufgabe oder Methode in der Wissensbasis.

**Klassenname** Name der Klasse des Individuums der entsprechenden elementaren Aufgabe, nichtelementaren Aufgabe oder Methode in der Wissensbasis.

**Priorität** Priorität des Flexiblen Hierarchischen Plans. Wird für Aufgaben des initialen Aufgabennetzwerks (die so genannten *Missionsaufgaben*, siehe Abbildung 4.6) spezifiziert und während der hierarchischen Zerlegung im entsprechenden Teilplan nach unten propagiert. Aufgaben mit hohen Prioritäten werden in der Ausführungsreihenfolge möglichst früh angeordnet.

**Optional** Gibt an, ob der Flexible Hierarchische Plan optional ist. Wird für Aufgaben des initialen Aufgabennetzwerks spezifiziert und während der hierarchischen Zerlegung im entsprechenden Teilplan nach unten propagiert. Optionale FHiPs können im Rahmen der hierarchischen Koordination (siehe Abschnitt 5.3.4) gelöscht werden, falls sich andernfalls keine konsistente Lösung für das Planungsproblem finden lässt. Müssen optionale FHiPs gelöscht werden, so werden nach Möglichkeit zunächst die FHiPs mit den niedrigsten Prioritäten entfernt.

**Parameter** Liste von Parametern. Diese entsprechen entweder Variablen oder Konstanten (siehe Abschnitt 4.3.1). Für die Berechnung der zusammengefassten Planinformationen (siehe Abschnitt 5.3.3) ist es jedoch wichtig, dass bei der hierarchischen Zerlegung alle Variablen durch Konstanten substituiert werden.

**Vorbedingungen** Bedingungen, die erfüllt sein müssen, damit die entsprechende elementare Aufgabe ausgeführt bzw. die entsprechende Zerlegungsmethode angewendet werden kann. Vorbedingungen sind nur für FHiPs der Typen *Elementar* und *Und* definiert.

**Währendbedingungen** Bedingungen, die im offenen Intervall zwischen Start- und Endzeitpunkt während der Ausführung der entsprechenden elementaren Aufgabe zugesichert werden. Währendbedingungen sind nur für *elementare* FHiPs definiert.

**Nachbedingungen** Bedingungen, die zum Endzeitpunkt der Ausführung der entsprechenden elementaren Aufgabe zugesichert werden. Nachbedingungen sind nur für *elementare* FHiPs definiert.

**Ressourcenverbrauch** Ressourcenmengen, die von der entsprechenden elementaren Aufgabe benötigt werden. Der Ressourcenverbrauch ist nur für *elementare* FHiPs definiert.

**Unterpläne** Unterpläne des Flexiblen Hierarchischen Plans. Bei *Und*-FHiPs entsprechen diese den Unteraufgaben der entsprechende Methode, bei *Oder*-FHiPs den Zerlegungsmethoden der entsprechenden nichtelementaren Aufgabe. Nur FHiPs der Typen *Oder* und *Und* verfügen über Unterpläne.

**Ordnungsrelationen** Ordnungsrelationen für Paare von Unterplänen der entsprechenden Zerlegungsmethode (siehe Abbildung 5.1). Ordnungsrelationen sind nur für *Und*-FHiPs definiert.

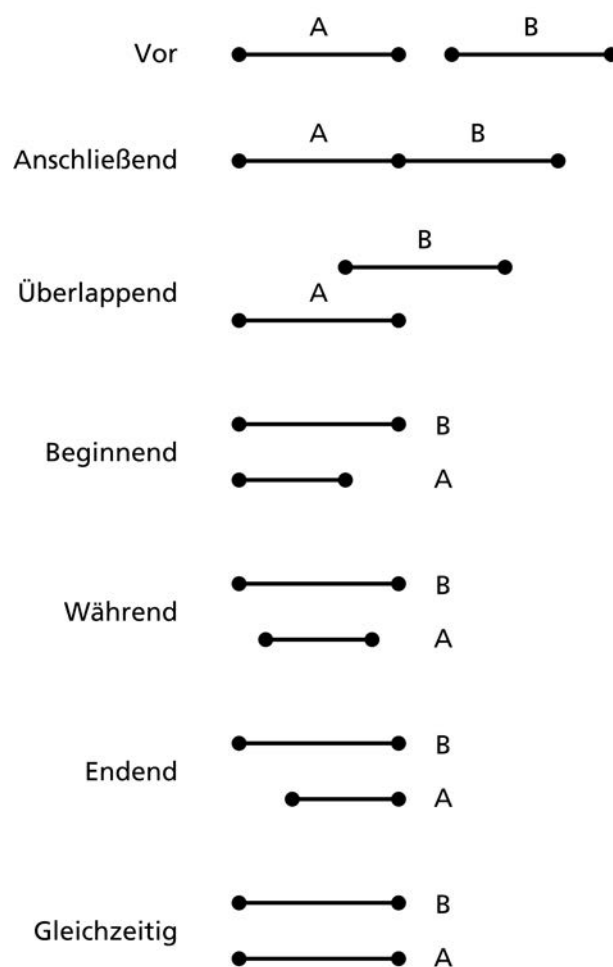


Abb. 5.1.: Übersicht über die verschiedenen Arten von zeitlichen Ordnungsbedingungen für Paare von Unterplänen.

**Zusammengefasste Planinformationen** Zusammengefasste Informationen über die Vor-, Während- und Nachbedingungen sowie den Ressourcenverbrauch (siehe Abschnitt 5.3.3) des Flexiblen Hierarchischen Plans und seiner Unterpläne.



**Verfügbarkeit** Verfügbarkeit der entsprechenden elementaren Aufgabe. Schlägt die Ausführung einer elementaren Aufgabe fehl (siehe Abschnitt 5.4), wird die Verfügbarkeit um einen bestimmten Wert verringert und es wird versucht die Aufgabe erneut auszuführen. Sinkt die Verfügbarkeit unter einen festgelegten Schwellenwert, schlägt die Ausführung endgültig fehl und die elementare Aufgabe wird bei der Planung nicht mehr berücksichtigt. Die Verfügbarkeit ist nur für *elementare* FHiPs definiert.

**Bewertungsfunktion** Bewertungsfunktion zur Auswahl zwischen den Zerlegungsmethoden einer nichtelementaren Aufgabe. Die Bewertungsfunktion ist nur für *Oder*-FHiPs definiert.

In Tabelle 5.2 wird ein Überblick über die Herkunft der einzelnen Komponenten gegeben. Sind die Komponenten Teil der entsprechenden Planrepräsentation, so ist dies mit ✓ gekennzeichnet. Sind die Komponenten nur indirekt Teil der entsprechenden Planrepräsentation, so ist dies mit ○ angegeben.

Komponente	FPS	CHiPs	Erweiterung
ID	✓		
Typ	○	✓	
Name			✓
Klassenname			✓
Priorität			✓
Optional			✓
Parameter	○		✓
Vorbedingungen	✓	✓	
Währendbedingungen	✓	✓	
Nachbedingungen	✓	✓	
Ressourcenverbrauch		✓	
Unterpläne	✓	✓	
Ordnungsrelationen	○	✓	
Zusammengefasste Planinformationen		✓	
Verfügbarkeit			✓
Bewertungsfunktion	✓		

Tab. 5.2.: Überblick über die Herkunft der einzelnen Komponenten der Flexiblen Hierarchischen Pläne.

**Vergleich mit Flexiblen Programmen** Flexible Programme (siehe Abschnitt 2.3.2) verfügen indirekt über einen Typ, da bei den inneren Knoten des entsprechenden Baumes die *Aktion* und bei den Blattknoten der *Prospekt* leer ist. Des Weiteren können bei Flexiblen Programmen für Aktionen zwar Parameter angegeben werden, diese können jedoch nicht wie beim Planen mit Hierarchischen Aufgabennetzwerken üblich von oben durch den Baum nach unten (in die entsprechenden Unteraufgaben bzw. Bedingungen) propagiert werden. Es handelt sich vielmehr

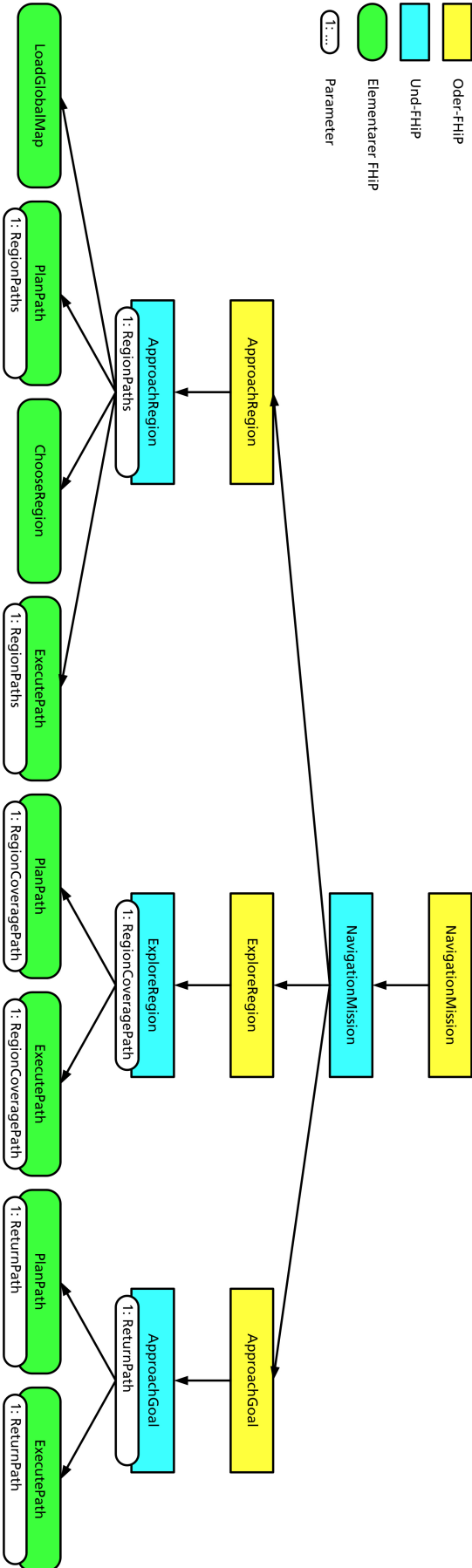


Abb. 5.2.: Beispiel für einen Flexiblen Hierarchischen Plan.

um für die Ausführung der jeweiligen Aktion erforderliche lokale Parameter, die zwischen den einzelnen Flexiblen Programmen über ein Umweltmodell ausgetauscht werden können. Ordnungsrelationen können für Flexible Programme nur eingeschränkt angegeben werden, indem Sitze als parallel ausführbar markiert werden. Andernfalls werden die entsprechenden Unterpläne seriell direkt nacheinander ausgeführt. Eine überlappende Ausführung wie bei Concurrent Hierarchical Plans ist nicht möglich.

**Vergleich mit Concurrent Hierarchical Plans** Die Angabe des Ressourcenverbrauchs erfolgt analog zu Concurrent Hierarchical Plans. Hierfür wird das entsprechende Modell zur Beschreibung metrischer Ressourcen verwendet, in welchem eine Ressource durch ein Tupel  $(wert_{min}, wert_{max}, typ)$  charakterisiert wird. Hierbei beschreiben  $wert_{min}$  und  $wert_{max}$  die Grenzen der Kapazität bzw. der verfügbaren Menge der Ressource. Der Typ  $typ$  der Ressource kann die Werte *verbrauchbar* und *unverbrauchbar* annehmen. Darüber hinaus können Zeitdauern ebenfalls in Form eines speziellen Ressourcentyps spezifiziert werden (siehe Abschnitt 5.3.3). Der Ressourcenverbrauch ist eine Abbildung von den metrischen Ressourcen auf die Menge, die verbraucht wird. Die Angabe von Ordnungsrelationen für Paare von Unterplänen wurde ebenfalls von den Concurrent Hierarchical Plans übernommen.

Ein wesentlicher Unterschied zwischen Concurrent Hierarchical Plans und Flexiblen Hierarchischen Plänen besteht in der jeweils zugrunde liegenden Repräsentationsform für Planungsprobleme. Während Concurrent Hierarchical Plans auf einer *mengentheoretischen Repräsentation* beruhen, wird für Flexible Hierarchische Pläne die *klassische Repräsentation* verwendet (siehe [40]), in welcher Zustände durch grundinstantiierte Atome einer funktionsfreien Sprache erster Ordnung beschrieben werden. Bei CHiPs werden die Vor-, Während- und Nachbedingungen entsprechend durch Mengen von Literalen ( $v$  oder  $\neg v$  für ein aussagenlogisches Symbol  $v$ ) angegeben. Bei Flexiblen Hierarchischen Plänen werden die Vor-, Während- und Nachbedingungen hingegen durch Mengen von Prädikaten und negierten Prädikaten beschrieben. Entsprechend verfügen auch elementare und nichtelementare Flexible Hierarchische Pläne selbst über Argumente in Form von Variablen bzw. Konstanten, hier als Parameter bezeichnet, die mit Hilfe von Verknüpfungen (siehe Abschnitt 4.3.1) an die jeweiligen Bedingungen und Unterpläne weitergegeben werden. Wichtig ist jedoch, dass alle in den Prädikaten der Bedingungen vorkommenden Variablen im Zuge der hierarchischen Verfeinerung durch entsprechende Konstanten ersetzt werden (die Prädikate also grundinstantiiert werden), da andernfalls die in Abschnitt 5.3.3 angegebenen Verfahren zur Bestimmung von zusammengefassten Planinformationen nicht anwendbar sind.

**Erweiterungen** Die in dieser Arbeit entwickelten Erweiterungen gegenüber Flexiblen Programmen und Concurrent Hierarchical Plans sind neben den bereits genannten Parametern die Elemente *Name*, *Klassenname*, *Priorität*, *Optional* und *Verfügbarkeit*. *Name* und *Klassenname*

beschreiben dabei das Individuum und die Klasse des Individuums der zugehörigen elementaren Aufgabe, nichtelementaren Aufgabe bzw. Methode in der Wissensbasis und dienen in erster Linie einer verbesserten Nachvollziehbarkeit der Planerzeugung und -ausführung. Die Angabe von *Prioritäten* für Aufgaben des initialen Aufgabennetzwerks und ihre *Optional-Kennzeichnung* erlauben den Umgang mit unterschiedlich gewichteten Aufgaben und optionalen Aufgaben. Diese beiden Erweiterungen sind vom OASIS-System (siehe Abschnitt 2.1.3) inspiriert und erlauben es, umfassendere Planungsstrategien zu realisieren. Die Angabe der *Verfügbarkeit* von elementaren Aufgaben zielt auf ein robusteres Verhalten der Planausführung ab, indem fehlgeschlagene Aktionen mit einer individuell definierten maximalen Anzahl von Wiederholungen erneut ausgeführt werden können.

### 5.3. Planerzeugung

In diesem Abschnitt wird zunächst ein Überblick über die Komponenten des Planers gegeben. Anschließend werden die einzelnen Schritte der Planerzeugung detailliert beschrieben.

#### 5.3.1. Übersicht über die Komponenten des Planers

Der im Rahmen dieser Arbeit entwickelte Planungsansatz besteht aus den folgenden vier Hauptkomponenten:

**Initialisierung** Auslesen des Planungsproblems aus der Wissensbasis und Konvertierung der Aufgaben und Methoden in Flexible Hierarchische Pläne.

**Vorverarbeitung** Rekursive Bestimmung der zusammengefassten Planinformationen für die Flexiblen Hierarchischen Pläne des initialen Aufgabennetzwerks.

**Koordination** Hierarchische Koordination der Flexiblen Hierarchischen Pläne des initialen Aufgabennetzwerks mit Hilfe der zusammengefassten Planinformationen.

**Serialisierung** Serialisierung der koordinierten Flexiblen Hierarchischen Pläne in eine XML-Repräsentation.

Diese vier Hauptkomponenten gliedern sich wie in Abbildung 5.3 dargestellt in den Gesamtprozess der semantischen Missionssteuerung (siehe auch Abschnitt 3.2.3) ein. Der *Manager* erstellt für die vom Benutzer vorgegebenen Missionsaufgaben sowie für die von der *Semantischen Inspektion* und der *Semantischen Navigation* vorgeschlagenen Inspektions- bzw. Navigationsaufgaben ein entsprechendes Planungsproblem in der Wissensbasis. Anschließend ruft er den *Planer* auf. Dieser liest zunächst das Planungsproblem aus der Wissensbasis aus und initialisiert für die Aufgaben und Methoden der Planungsdomäne entsprechende Flexible Hierarchische

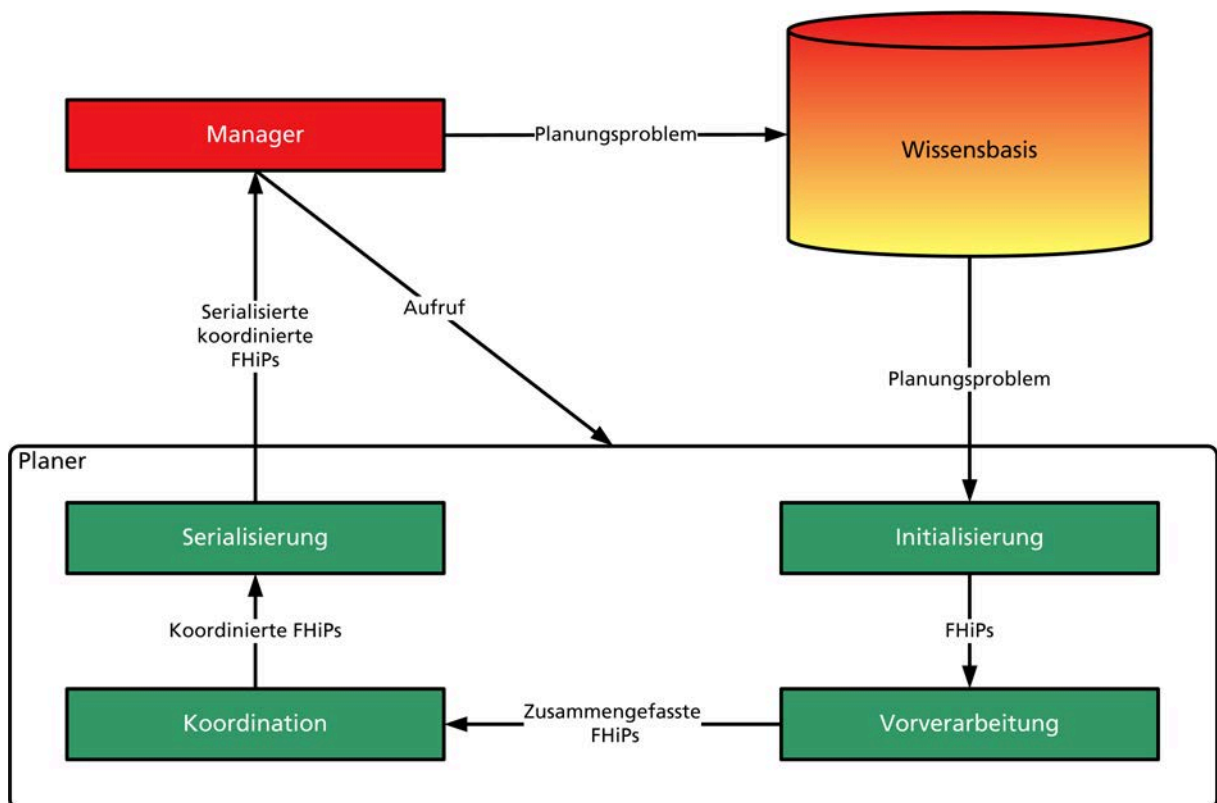


Abb. 5.3.: Übersicht über die Komponenten des Planers.

Pläne (*Initialisierung*). Anschließend werden für die im initialen Aufgabennetzwerk angegebenen Aufgaben rekursiv zusammengefasste Planinformationen berechnet (*Vorverarbeitung*). Im nächsten Schritt werden die mit zusammengefassten Planinformationen angereicherten Pläne hierarchisch koordiniert (*Koordination*). Abschließend werden die koordinierten Pläne in eine XML-Repräsentation serialisiert, die an den *Manager* zurückgegeben wird, welcher sie schließlich zur Ausführung an die *Ausführungseinheit* weitergibt.

### 5.3.2. Initialisierung

Bei der Initialisierung wird das vom *Manager* angegebene Planungsproblem aus der Wissensbasis ausgelesen. Dabei werden für die elementaren und nichtelementaren Aufgaben sowie für die Methoden der Planungsdomäne entsprechende Flexible Hierarchische Pläne der Typen *Elementar*, *Oder* bzw. *Und* erzeugt. Tabelle 5.3 gibt einen Überblick darüber, welche Komponenten der Flexiblen Hierarchischen Pläne beim Auslesen mit welchen in der Wissensbasis für elementare und nichtelementare Aufgaben sowie für Methoden angegebenen Informationen initialisiert werden. Die Informationen, die direkt aus der Ontologie ausgelesen werden, sind in der Tabelle mit ✓ gekennzeichnet.

Komponente	El. Aufgabe	Nichtel. Aufgabe	Methode
ID			
Typ	Elementar	Oder	Und
Name	✓	✓	✓
Klassenname	✓	✓	✓
Priorität	✓	✓	✓
Optional	✓	✓	✓
Parameter	✓	✓	✓
Vorbedingungen	✓		✓
Währendbedingungen	✓		
Nachbedingungen	✓		
Ressourcenverbrauch	✓		
Unterpläne		✓	✓
Ordnungsrelationen			✓
Zusammengefasste Planinformationen			
Verfügbarkeit	✓		
Bewertungsfunktion			✓

Tab. 5.3.: Flexible Hierarchische Pläne und ihre Beziehung zu elementaren und nichtelementaren Aufgaben sowie Methoden des Planens mit Hierarchischen Aufgabennetzwerken.

### 5.3.3. Vorverarbeitung

Die Vorverarbeitung gliedert sich in zwei Teilschritte. Im ersten Schritt werden die den Aufgaben des initialen Aufgabennetzwerks entsprechenden Flexiblen Hierarchischen Pläne solange hierarchisch zerlegt, bis in allen Unterzweigen nur noch nicht weiter expandierbare elementare FHiPs vorhanden sind. Dabei werden für die Oder-FHiPs sämtliche Zerlegungsmethoden beibehalten. Im zweiten Schritt werden ausgehend von den elementaren FHiPs rekursiv die zusammengefassten Planinformationen für die einzelnen Flexiblen Hierarchischen Pläne bestimmt. Die zusammengefassten Planinformationen bestehen aus zusammengefassten Bedingungen und zusammengefassten Ressourcenverbräuchen, deren Gewinnung in den nachfolgenden Abschnitten erläutert wird. Die in diesen Abschnitten beschriebenen Verfahren zur Berechnung von zusammengefassten Planinformationen stammen aus [27] und wurden entsprechend für Flexible Hierarchische Pläne angepasst.

#### Bestimmung von zusammengefassten Bedingungen

Die zusammengefassten Informationen für einen Plan  $p$  werden mit  $p_{sum}$  bezeichnet und bilden ein Tupel  $(V_{sum}, W_{sum}, N_{sum})$ , welches aus Mengen von zusammengefassten Bedingungen besteht. Die zusammengefassten [Vor-, Nach-] Bedingungen von  $p$ ,  $[V_{sum}, N_{sum}]$ , enthalten die externen [Vor-, Nach-] Bedingungen von  $p$ . Die zusammengefassten Während-Bedingungen von  $p$ ,  $W_{sum}$ , enthalten alle Bedingungen, die während irgendeiner Ausführung von  $p$  gelten müssen, damit diese erfolgreich ist. Eine Bedingung  $c$  in einer dieser Mengen ist ein Tupel

$(l, E, Z)$ , wobei  $l(c)$  das Literal der Bedingung,  $E(c)$  die Existenz und  $Z(c)$  das Zeitverhalten der Bedingung bezeichnen. Die Existenz  $E(c)$  kann die Werte *muss* oder *kann* annehmen. Bei einer *muss*-Bedingung muss  $l(c)$  für jede erfolgreiche Ausführung von  $p$  gelten. Bei einer *kann*-Bedingung muss  $l(c)$  mindestens für eine erfolgreiche Ausführung gelten. Das Zeitverhalten  $Z(c)$  kann die Werte *immer*, *manchmal*, *zuerst* oder *zuletzt* annehmen. Für  $c \in W_{sum}$  gilt  $Z(c) = immer$ , wenn  $l(c)$  eine Während-Bedingung ist, die während der gesamten Ausführung von  $p$  gelten muss, andernfalls gilt  $Z(c) = manchmal$ . Letzteres bedeutet, dass  $l(c)$  an mindestens einem Punkt der Ausführung gelten muss. Für  $c \in V_{sum}$  gilt  $Z(c) = zuerst$ , wenn  $l(c)$  zu Beginn der Ausführung von  $p$  gelten muss, andernfalls gilt  $Z(c) = manchmal$ . Für  $c \in N_{sum}$  gilt  $Z(c) = zuletzt$ , wenn  $l(c)$  am Ende einer erfolgreichen Ausführung von  $p$  zugesichert wird, andernfalls gilt  $Z(c) = manchmal$ .

Das Verfahren zur Bestimmung von zusammengefassten Bedingungen für einen Plan  $p$  ist rekursiv. Zunächst müssen für jeden Unterplan von  $p$  die zusammengefassten Bedingungen bestimmt werden. Anschließend werden mit Hilfe der im Folgenden beschriebenen Verfahren die zusammengefassten Bedingungen für  $p$  gegeben die zusammengefassten Bedingungen der Unterpläne von  $p$  und gegeben die Bedingungen von  $p$  selbst bestimmt.

### Zusammengefasste Bedingungen für elementare und nichtelementare Pläne

- Für jedes Literal  $l$  in  $V(p)$ ,  $W(p)$  und  $N(p)$  wird eine Bedingung  $c$  mit dem Literal  $l$  zur entsprechenden Menge der zusammengefassten Bedingungen von Plan  $p$  hinzugefügt. Für die Existenz gilt  $E(c) = muss$  und das Zeitverhalten  $Z(c)$  nimmt in Abhängigkeit davon, ob es sich bei  $l$  um eine Vor-, Während- oder Nach-Bedingung handelt, den entsprechenden Wert *zuerst*, *immer* oder *zuletzt* an.

### Zusammengefasste [Vor-, Nach-] Bedingungen für einen *Und*-Plan

- Bei einem *Und*-Plan wird für jede zusammengefasste [Vor-, Nach-] Bedingung  $c'$  der Unterpläne von  $p$ , die nicht von einem anderen Unterplan von  $p$  [erreicht werden muss, zurückgesetzt werden muss] oder überschrieben werden muss, eine entsprechende Bedingung  $c$  zu den zusammengefassten [Vor-, Nach-] Bedingungen von  $p$  hinzugefügt und  $l(c) = l(c')$  gesetzt.
- Wenn  $l(c)$  eine [Vor-, Nach-] Bedingung von  $p$  oder das Literal einer zusammengefassten [Vor-, Nach-] Bedingung mit Existenz *muss* eines Unterplans von  $p$  ist, die von keinen anderen Unterplänen [erreicht werden kann, zurückgesetzt werden kann] oder überschrieben werden kann, dann gilt  $E(c) = muss$ . Andernfalls gilt  $E(c) = kann$ .
- Wenn  $l(c)$  eine [Vor-, Nach-] Bedingung von  $p$  oder das Literal einer zusammengefassten [Vor-, Nach-] Bedingung mit Zeitverhalten [zuerst, zuletzt] eines Unterplans von  $p$  ist, der

zeitlich am [niedrigsten, höchsten] geordnet ist (d. h. dass es keine anderen Unterpläne gibt, die entsprechend der Ordnungsrelationen [vorher beginnen, danach enden]), dann gilt  $Z(c) = [zuerst, zuletzt]$ . Andernfalls gilt  $Z(c) = manchmal$ .

### Zusammengefasste [Vor-, Nach-] Bedingungen für einen *Oder*-Plan

- Bei einem *Oder*-Plan wird für jede zusammengefasste [Vor-, Nach-] Bedingung  $c'$  der Unterpläne von  $p$  eine entsprechende Bedingung  $c$  zu den zusammengefassten [Vor-, Nach-] Bedingungen von  $p$  hinzugefügt und  $l(c) = l(c')$  gesetzt.
- Wenn  $l(c)$  eine [Vor-, Nach-] Bedingung von  $p$  oder das Literal einer zusammengefassten [Vor-, Nach-] Bedingung mit Existenz *muss* von allen Unterplänen von  $p$  ist, dann gilt  $E(c) = muss$ . Andernfalls gilt  $E(c) = kann$ .
- Wenn  $l(c)$  eine [Vor-, Nach-] Bedingung von  $p$  oder das Literal einer zusammengefassten [Vor-, Nach-] Bedingung mit Zeitverhalten  $[zuerst, zuletzt]$  eines Unterplans von  $p$  ist, dann gilt  $Z(c) = [zuerst, zuletzt]$ . Andernfalls gilt  $Z(c) = manchmal$ .

### Zusammengefasste Während-Bedingungen für einen *Und*-Plan

- Sei  $C$  definiert als die Menge aller zusammengefassten Während-Bedingungen der Unterpläne von  $p$  vereinigt mit der Menge der zusammengefassten Vor-Bedingungen der Unterpläne, die nicht immer das Zeitverhalten *zuerst* in einem zeitlich am niedrigsten geordneten Unterplan besitzen, und der Menge der zusammengefassten Nach-Bedingungen der Unterpläne, die nicht immer das Zeitverhalten *zuletzt* in einem zeitlich am höchsten geordneten Unterplan besitzen. Für jedes  $c' \in C$  wird eine Bedingung  $c$  zu den zusammengefassten Während-Bedingungen von  $p$  hinzugefügt und  $l(c) = l(c')$  gesetzt.
- Wenn  $l(c)$  eine Während-Bedingung von  $p$  oder ein Literal einer zusammengefassten Bedingung  $c' \in C$ , wie oben definiert, mit Existenz *muss* ist und nie eine Bedingung mit Zeitverhalten *zuerst* oder *zuletzt* ist, dann gilt  $E(c) = muss$ . Andernfalls gilt  $E(c) = kann$ .
- Wenn  $l(c)$  eine Während-Bedingung von  $p$  oder ein Literal in den zusammengefassten Während-Bedingungen mit Zeitverhalten *immer* in den Unterplänen von  $p$  ist, dessen Intervalle das Intervall der Ausführung von  $p$  *abdecken müssen*, dann gilt  $Z(c) = immer$ . Andernfalls gilt  $Z(c) = manchmal$ .

### Zusammengefasste Während-Bedingungen für einen *Oder*-Plan

- Bei einem *Oder*-Plan wird für jede zusammengefasste Während-Bedingung  $c'$  der Unterpläne von  $p$  eine entsprechende Bedingung  $c$  zu den zusammengefassten Während-Bedingungen von  $p$  hinzugefügt und  $l(c) = l(c')$  gesetzt.



- Wenn  $l(c)$  eine Während-Bedingung von  $p$  oder eine zusammengefasste Während-Bedingung mit Existenz *muss* von allen Unterplänen von  $p$  ist, dann gilt  $E(c) = muss$ . Andernfalls gilt  $E(c) = kann$ .
- Wenn  $l(c)$  eine Während-Bedingung von  $p$  oder eine zusammengefasste Während-Bedingung mit Zeitverhalten *immer* von allen Unterplänen von  $p$  ist, dann gilt  $Z(c) = immer$ . Andernfalls gilt  $Z(c) = manchmal$ .

### Bestimmung von zusammengefassten Ressourcenverbräuchen

Der zusammengefasste Ressourcenverbrauch besteht aus Intervallen für die möglicherweise verbrauchten Mengen an Ressourcen während und nach der Ausführung einer abstrakten Aufgabe und wird als Tupel  $(lokal_{min}, lokal_{max}, dauerhaft)$  repräsentiert. Dabei findet der lokale Ressourcenverbrauch während der Ausführung der Aufgabe statt und der dauerhafte Ressourcenverbrauch repräsentiert den Verbrauch, der nach der Beendigung der Aufgabe für verbrauchbare Ressourcen anhält.

Das Verfahren zur Bestimmung von zusammengefassten Ressourcenverbräuchen geht nach dem gleichen Ansatz wie die Bestimmung von zusammengefassten Bedingungen vor. Beginnend mit den Blättern findet der Algorithmus elementare Aufgaben, die eine konstante Menge einer Ressource verbrauchen. Der zusammengefasste Ressourcenverbrauch einer Aufgabe, die  $x$  Einheiten einer Ressource verwendet, beträgt über die Dauer der Aufgabe  $([x, x], [x, x], [0, 0])$  bzw.  $([x, x], [x, x], [x, x])$  für unverbrauchbare bzw. verbrauchbare Ressourcen.

Beim Aufsteigen im Und/Oder-Baum des Plans erreicht der Algorithmus entweder einen *Und*- oder einen *Oder*-Zweig. Für einen *Oder*-Zweig ergibt sich der kombinierte zusammengefasste Ressourcenverbrauch aus der folgenden *Oder*-Berechnung:

$$\begin{aligned} & ([ \min_{k \in Kinder} (ug(lokal_{min}(k))) \quad , \quad \max_{k \in Kinder} (og(lokal_{min}(k))) ], \\ & [ \min_{k \in Kinder} (ug(lokal_{max}(k))) \quad , \quad \max_{k \in Kinder} (og(lokal_{max}(k))) ], \\ & [ \min_{k \in Kinder} (ug(dauerhaft(k))) \quad , \quad \max_{k \in Kinder} (og(dauerhaft(k))) ], \end{aligned} \quad [5.1]$$

wobei  $ug$  und  $og$  die untere bzw. obere Grenze des Intervalls extrahieren. Mit *Kinder* werden die Kinder des Zweiges bezeichnet, wobei ihre jeweilige Dauer auf die Länge des am längsten dauernden Kindes verlängert wird. Durch diese Verlängerung der Dauer verändert sich der zusammengefasste Ressourcenverbrauch eines Kindes, da das Verbrauchsprofil des Kindes während der Verlängerung einen Ressourcenverbrauch von Null aufweist.

Die Berechnung des zusammengefassten Ressourcenverbrauchs für einen *Und*-Zweig ist aufgrund der zeitlichen Anordnungsmöglichkeiten von lose beschränkten Unteraufgaben etwas komplizierter. Im einfachsten Fall, in dem alle Unteraufgaben sequentiell direkt nacheinander

ausgeführt werden, werden die Verbrauchsprofile aneinander angefügt und der sich ergebende zusammengefasste Ressourcenverbrauch wird mit Hilfe der *Seriell-Und*-Berechnung bestimmt:

$$\begin{aligned}
 & ([ \min_{k \in \text{Kinder}} (ug(\text{lokal}_{\min}(k)) + \sum_{ug}^{vor}(k)), \min_{k \in \text{Kinder}} (og(\text{lokal}_{\min}(k)) + \sum_{og}^{vor}(k)) ], \\
 & [ \max_{k \in \text{Kinder}} (ug(\text{lokal}_{\max}(k)) + \sum_{ug}^{vor}(k)), \max_{k \in \text{Kinder}} (og(\text{lokal}_{\max}(k)) + \sum_{og}^{vor}(k)) ], \\
 & [ \sum_{k \in \text{Kinder}} (ug(\text{dauerhaft}(k))) \quad , \quad \sum_{k \in \text{Kinder}} (og(\text{dauerhaft}(k))) ]), \quad [5.2]
 \end{aligned}$$

wobei  $\sum_{ug}^{vor}(k)$  und  $\sum_{og}^{vor}(k)$  die untere bzw. obere Grenze der akkumulierten dauerhaften Ressourcenverbräuche der Kinder bezeichnen, die vor  $k$  ausgeführt werden.

Im Fall, dass alle Unteraufgaben parallel ausgeführt werden und eine identische Dauer besitzen, addieren sich die Verbrauchsprofile, und der resultierende zusammengefasste Ressourcenverbrauch des Zweiges wird mit Hilfe der *Parallel-Und*-Berechnung bestimmt:

$$\begin{aligned}
 & ([ \sum_{k \in \text{Kinder}} (ug(\text{lokal}_{\min}(k))) \quad , \quad \max_{k \in \text{Kinder}} (og(\text{lokal}_{\min}(k)) + \sum_{og}^{nicht}(k)) ], \\
 & [ \min_{k \in \text{Kinder}} (ug(\text{lokal}_{\max}(k)) + \sum_{ug}^{nicht}(k)), \sum_{k \in \text{Kinder}} (og(\text{lokal}_{\max}(k))) ], \\
 & [ \sum_{k \in \text{Kinder}} (ug(\text{dauerhaft}(k))) \quad , \quad \sum_{k \in \text{Kinder}} (og(\text{dauerhaft}(k))) ]), \quad [5.3]
 \end{aligned}$$

wobei  $\sum_{og}^{nicht}(k)$  und  $\sum_{ug}^{nicht}(k)$  die Summe der oberen Grenzen von  $\text{lokal}_{\min}$  bzw. die Summe der unteren Grenzen von  $\text{lokal}_{\max}$  für alle Kinder mit Ausnahme von  $k$  bezeichnen.

Zur Behandlung von *Und*-Plänen mit losen zeitlichen Ordnungen, werden alle zulässigen Ordnungen von Endzeitpunkten der Kindaufgaben betrachtet. Gegeben eine Ordnung, werden die folgenden Schritte durchgeführt:

1. Bestimmung von Unterintervallen anhand der Endzeitpunkte der Kinder.
2. Berechnung von zusammengefassten Ressourcenverbräuchen für jede Kindaufgabe-/Unterintervall-Kombination.
3. Kombination der zusammengefassten Ressourcenverbräuche von parallelen Unterintervallen mit Hilfe der *Parallel-Und*-Berechnung.
4. Verkettung der Unterintervalle mit Hilfe der *Seriell-Und*-Berechnung.

Abschließend wird der zusammengefasste Ressourcenverbrauch des *Und*-Plans berechnet, indem die zusammengefassten Ressourcenverbräuche aller möglichen zeitlichen Ordnungen mit Hilfe einer *Oder*-Berechnung kombiniert werden.

Das oben beschriebene Verfahren zur Bestimmung von zusammengefassten Ressourcenverbräuchen wurde um die Sonderbehandlung der Ressource *Zeitdauer* erweitert. Die Berechnung des zusammengefassten Ressourcenverbrauchs für die Ressource *Zeitdauer* erfolgt entsprechend der folgenden Regeln:

- Für einen *Oder-Zweig* ist die minimale *Zeitdauer* des Elternplans gleich der kürzesten minimalen *Zeitdauer* eines Unterplans. Die maximale *Zeitdauer* des Elternplans entspricht der längsten maximalen *Zeitdauer* eines Unterplans.
- Für einen seriellen *Und-Zweig*, bei dem alle Unteraufgaben sequentiell direkt nacheinander ausgeführt werden, beträgt die minimale *Zeitdauer* des Elternplans die Summe der minimalen *Zeitdauern* der Unterpläne. Die maximale *Zeitdauer* ist entsprechend gleich der Summe der maximalen *Zeitdauern* der Unterpläne.
- Für einen parallelen *Und-Zweig*, bei dem alle Unteraufgaben parallel ausgeführt werden, werden jeweils die längsten minimalen *Zeitdauern* und die längsten maximalen *Zeitdauern* der Unterpläne als minimale bzw. maximale *Zeitdauer* des Elternplans verwendet.

Zur Behandlung von teilgeordneten *Und-Plänen* wird analog zu den oben genannten Schritten für die Berechnung von zusammengefassten Ressourcenverbräuchen für metrische Ressourcen verfahren.

#### 5.3.4. Koordination

Nach der Gewinnung von zusammengefassten Planinformationen werden diese im nächsten Schritt, der Koordination, dazu verwendet, mögliche Konflikte zwischen den Plänen für die einzelnen Aufgaben des initialen Aufgabennetzwerks zu erkennen und diese mit geeigneten Verfahren aufzulösen. Im Folgenden werden zunächst entsprechende Verfahren zur Identifikation von so genannten *Bedrohungen* auf abstrakten Ebenen vorgestellt. Darauf basierend wird ein Algorithmus zur nebenläufigen hierarchischen Koordination vorgestellt, welcher die erkannten potentiellen Konflikte mit geeigneten Operatoren auflöst. Das gleiche Verfahren kann darüber hinaus auch dazu verwendet werden, mögliche Konflikte in den jeweiligen einzelnen Plänen für die Aufgaben des initialen Aufgabennetzwerks zu erkennen und diese so zu verfeinern, dass die Konflikte aufgelöst werden. Abschließend werden verschiedene mögliche Strategien zur Auflösung von potentiellen Konflikten diskutiert.

Die in diesem Abschnitt vorgestellten Verfahren stammen aus [27] und [28] und wurden entsprechend für Flexible Hierarchische Pläne angepasst.

## Identifikation von Bedrohungen auf abstrakten Ebenen

Nach [27] besteht für eine Menge von Plänen  $P$  mit den Ordnungsbedingungen *ordnungen* eine *Bedrohung* zwischen einem abstrakten Plan  $p \in P$  und einer zusammengefassten Bedingung  $c'$  eines anderen Plans  $p' \in P$  genau dann, wenn  $p$   $c'$  *überschreiben könnte* (engl. *may-clobber*). Die Bedrohung kann nicht aufgelöst werden, wenn  $p$   $c'$  *überschreiben muss* (engl. *must-clobber*) und  $\text{muss}(c')$  gilt. Existiert kein Plan, der  $c'$  überschreiben könnte, dann wird jede Bedingung, die von  $c'$  zusammengefasst wird, erfüllt, unabhängig davon wie die Pläne von  $P$  zerlegt und ausgeführt werden. Gilt jedoch, dass  $c'$  überschrieben werden muss, dann werden alle Bedingungen, die von  $c'$  zusammengefasst werden, für jede Ausführung von  $P$  überschrieben. Dies bedeutet, dass der mögliche Erfolg der Pläne eines Agenten davon abhängt, ob ihre Bedingungen Bedrohungen unterliegen und ob diese Bedrohungen aufgelöst werden können.

Wenn Agenten feststellen könnten, dass ihre Pläne unter bestimmten zeitlichen Ordnungsbedingungen auf jede Weise zerlegt werden können (*KannAufJedeWeise*, engl. *CanAnyWay*) oder dass es unter diesen Bedingungen keinen Weg gibt, sie erfolgreich zu zerlegen (*–KönnteAufMancheWeise*, engl. *–MightSomeWay*), dann könnten sie Koordinationsentscheidungen auf abstrakten Ebenen treffen, ohne eine potentiell kostenintensive Suche für Planzusammenführungen auf niedrigeren Ebenen beginnen zu müssen. Die vorstehend genannten Relationen für Pläne mit Ordnungsbedingungen basieren auf potentiellen Bedrohungen zwischen den Plänen. Existieren keine Bedrohungen, dann ist die *KannAufJedeWeise*-Eigenschaft erfüllt. Gibt es keine unauflösbaren Bedrohungen, dann ist die *KönnteAufMancheWeise*-Eigenschaft erfüllt. Über das potentielle Treffen von Koordinationsentscheidungen auf abstrakten Ebenen hinaus, kann sich die Planung/Koordination der Agenten außerdem auf die Auflösung der Bedrohungen fokussieren und es kann vermieden werden, sich mit Details von Plänen zu beschäftigen, für die keine Konflikte bestehen.

Der Algorithmus zur Bestimmung von *KannAufJedeWeise* (siehe Algorithmus 5.1) für zusammengefasste Bedingungen ist einfach, da nur das Vorhandensein von Bedrohungen geprüft werden muss. Der Algorithmus zur Bestimmung von *KönnteAufMancheWeise* ist komplizierter, da es nicht ausreichend ist, nur auf das Vorhandensein einer nicht auflösbaren Bedrohung zu prüfen (siehe [28]). Um zu bestimmen, ob *KönnteAufMancheWeise* falsch ist, muss ein Agent eine erschöpfende Suche über eine exponentielle Anzahl von Ablaufplänen durchführen, um zu prüfen, ob nicht alle Konflikte aufgelöst werden können. Anstelle der exponentiellen Suche wird in [28] ein einfacher Algorithmus verwendet (siehe Algorithmus 5.2), der nur auf *muss überschreiben* Beziehungen prüft. Der *KannAufJedeWeise*-Algorithmus ist *zuverlässig* und *vollständig*. Der *KönnteAufMancheWeise*-Algorithmus ist hingegen *vollständig*, aber *nicht zuverlässig*. Dies bedeutet, dass die Bestimmung von *–KönnteAufMancheWeise* *zuverlässig*, aber *nicht vollständig* ist. Davon unbeschadet werden beide Algorithmen im nächsten Abschnitt verwendet, um einen *zuverlässigen* und *vollständigen* Planungs-/Koordinationsalgorithmus zu definieren.

---

**Algorithmus 5.1** KannAufJedeWeise (siehe [28])

---

**Eingabe:** ordnungen,  $P_{sum}$ **Ausgabe:** wahr oder falsch

```

for all  $p_{sum} \in P_{sum}$  do
  if  $\neg$ konsistent( $p_{sum}$ ) then
    return falsch
  end if
  for all  $p'_{sum} \in P_{sum}$  do
    for all zusammengefasste Bedingung  $c$  von  $p_{sum}$  do
      if ( $p'$  kann überschreiben  $c$ ) und ( $c$  ist kann oder muss) then
        return falsch
      end if
    end for
  end for
end for
for all  $res \in$  ressource do
  if  $\neg$ KannAufJedeWeise(ordnungen,  $P_{sum}$ ,  $res$ ) then
    return falsch
  end if
end for
return wahr

```

---



---

**Algorithmus 5.2** KönnteAufMancheWeise (siehe [28])

---

**Eingabe:** ordnungen,  $P_{sum}$ **Ausgabe:** wahr oder falsch

```

for all  $p_{sum} \in P_{sum}$  do
  for all  $p'_{sum} \in P_{sum}$  do
    for all zusammengefasste Bedingung  $c$  von  $p_{sum}$  do
      if ( $p'$  muss überschreiben  $c$ ) und ( $c$  ist muss) then
        return falsch
      end if
    end for
  end for
end for
for all  $res \in$  ressource do
  if  $\neg$ KönnteAufMancheWeise(ordnungen,  $P_{sum}$ ,  $res$ ) then
    return falsch
  end if
end for
return wahr

```

---

$KannAufJedeWeise(ordnungen, P_{sum}, res)$  und  $KönnteAufMancheWeise(ordnungen, P_{sum}, res)$  können anhand der Werte der zusammengefassten Ressourcenverbräuche, die aus Aufrufen der *Seriell-Und*- und *Parallel-Und*-Berechnungen (siehe Abschnitt 5.3.3) resultieren, bestimmt werden.  $KannAufJedeWeise(ordnungen, P_{sum}, res)$  ist wahr genau dann, wenn es keine potentiellen Bedrohungen gibt. Eine Bedrohung wird von den genannten Berechnungen festgestellt, wenn ein Intervall  $i$  gefunden wird, für das gilt

$$\begin{aligned} ug(local_{min}(i)) &< min(res) && \vee \\ ug(dauerhaft(i)) &< min(res) && \vee \\ og(local_{max}(i)) &> max(res) && \vee \\ og(dauerhaft(i)) &> max(res) && [5.4] \end{aligned}$$

$KönnteAufMancheWeise(ordnungen, P_{sum}, res)$  ist wahr genau dann, wenn es eine mögliche Ausführung mit keinen potentiellen Bedrohungen gibt. Eine solche Ausführung wird von der *Seriell-Und*-Berechnung festgestellt, wenn sie einen zusammengefassten Ressourcenverbrauch zurückliefert, für den gilt

$$\begin{aligned} ug(local_{min}(i)) &\geq min(res) && \wedge \\ ug(dauerhaft(i)) &\geq min(res) && \wedge \\ og(local_{max}(i)) &\leq max(res) && \wedge \\ og(dauerhaft(i)) &\leq max(res) && [5.5] \end{aligned}$$

### Nebenläufige hierarchische Verfeinerung

Im vorhergehenden Abschnitt wurden Algorithmen vorgestellt, mit denen für eine Menge von Plänen festgestellt werden kann, ob sie unter einer Teilmenge von Ordnungsbedingungen definitiv konfliktfrei ist ( $KannAufJedeWeise$ ) oder über unauflösbare Konflikte verfügt ( $\neg KönnteAufMancheWeise$ ). In diesem Abschnitt werden diese Verfahren gemäß [27] in einen zuverlässigen und vollständigen Algorithmus integriert, der für eine Gruppe von Agenten einen konsistenten koordinierten Plan sucht. Anschließend wird beschrieben, wie dieser Koordinationsalgorithmus zu einem nebenläufigen hierarchischen Einzelagenten-Planungsalgorithmus modifiziert werden kann.

Der Koordinationsalgorithmus (siehe Algorithmus 5.3) beginnt die Suche mit einer Menge, die für jeden Agenten einen Plan auf oberster Ebene enthält und den koordinierten Plan repräsentiert. Zunächst wird versucht, auf dieser Ebene eine Lösung zu finden, dann wird die Hierarchie solange tiefer und tiefer expandiert, bis die optimale Lösung gefunden wird oder der Suchraum erschöpft ist. Ein Zustand der Suche ist ein teilweise ausgearbeiteter Plan, der als Menge von *Und*-Plänen (einer für jeden Agenten) zusammen mit einer Menge von zeitlichen

**Algorithmus 5.3** Nebenläufiger hierarchischer Koordinationsalgorithmus (siehe [27])**Eingabe:** Menge von Plänen auf oberster Ebene (pläne), initialer Zustand (initialer\_zustand)**Ausgabe:** Menge von Lösungen (lösungen)warteschlange = {(pläne,  $\emptyset$ ,  $\emptyset$ )}lösungen =  $\emptyset$ **loop****if** warteschlange ==  $\emptyset$  **then****return** lösungen**end if**

(pläne, ordnungen, blockiert) = Pop(warteschlange)

**if** *KannAufJedeWeise*(initialer\_zustand, pläne, ordnungen, blockiert) **then**

lösung = (pläne, ordnungen, blockiert)

lösungen = lösungen  $\cup$  {lösung}**for all**  $l_1$  und  $l_2$  in lösungen **do****if** Dominiert( $l_1$ ,  $l_2$ ) **then**lösungen = lösungen  $\setminus$  { $l_2$ }**end if****end for****end if****if** *KönnteAufMancheWeise*(initialer\_zustand, pläne, ordnungen, blockiert) **then**operator = WähleOperator({*Expandiere*, *Selektiere*, *Blockiere*, *Beschränke*})

WendeOperatorAn(operator, pläne, ordnungen, blockiert, warteschlange)

**end if****end loop****return** lösungen

Ordnungsbedingungen und einer Menge von blockierten Plänen repräsentiert wird. Die Unterpläne der *Und*-Pläne sind die Blätter der teilweise expandierten Planhierarchien der einzelnen Agenten. Die Menge der zeitlichen Ordnungsbedingungen enthält Synchronisationsbedingungen, die während der Suche zusätzlich zu den Ordnungsbedingungen, die von den Planhierarchien der einzelnen Agenten vorgegeben sind, hinzugefügt werden. Die Menge der blockierten Unterpläne beinhaltet die während der Suche entfernten Unterpläne.

Die Suche kann mit Hilfe verschiedener Operatoren gelenkt werden. Diese Operatoren expandieren nichtelementare Pläne (*Expandiere*), blockieren *Oder*-Unterpläne (*Blockiere*) und fügen zeitliche Ordnungsbedingungen für Paare von Plänen hinzu (*Beschränke*). Ein Unterplan eines *Oder*-Plans wird nur zum Suchzustand hinzugefügt, wenn alle anderen Unterpläne blockiert sind. Die Blockierung eines *Oder*-Unterplans kann ein effektives Mittel sein, um einen Konflikt aufzulösen, in den die anderen *Oder*-Unterpläne nicht involviert sind. Dies kann zu am wenigsten einschränkenden (engl. *least commitment*) abstrakten Lösungen führen, bei denen die Agenten flexibel zwischen mehreren verbleibenden anwendbaren Unterplänen auswählen können. Darüber hinaus kann auch ein anderer Ansatz verfolgt werden, indem Unterpläne gezielt ausgewählt werden (*Selektiere*), wodurch alle anderen effektiv blockiert werden. Auf diese

Weise können Wahlmöglichkeiten, die eine höhere Präferenz besitzen oder wahrscheinlicher zur Auflösung eines Konflikts führen, untersucht werden.

Im Pseudocode des Algorithmus 5.3 enthält die *warteschlange* die expandierten Suchzustände. Ist die *KannAufJedeWeise*-Relation für einen Suchzustand gültig, so wird mit Hilfe der *Dominiert*-Funktion entschieden, ob die bisher gefundenen Lösungen besser für jeden Agenten sind als die durch den aktuellen Suchzustand repräsentierte Lösung. Die durch den aktuellen Suchzustand repräsentierte Lösung wird beibehalten, wenn sie nicht von den bisher gefundenen Lösungen dominiert wird. Ist die *KönnteAufMancheWeise*-Relation für einen Suchzustand nicht gültig, kann der durch den aktuellen Suchzustand repräsentierte Suchraum verworfen werden. Andernfalls werden die oben genannten Operatoren angewendet, um den Suchzustand zu expandieren.

Die Zuverlässigkeit und Vollständigkeit des Koordinationsalgorithmus hängt von der Zuverlässigkeit und Vollständigkeit der Identifikation von Lösungen sowie der vollständigen Exploration des Suchraums ab. Jeder Suchzustand wird mit Hilfe des *KannAufJedeWeise*-Algorithmus daraufhin überprüft, ob er eine Lösung ist. Obwohl der Algorithmus zur Bestimmung von  $\neg$ *KönnteAufMancheWeise* nur für eine total geordnete Menge von Plänen vollständig ist, wird er verwendet, um ungültige Zweige des Suchraums zu verwerfen. Aus diesem Grund ist es ausreichend, dass er zuverlässig ist.

Die Suche des Koordinationsalgorithmus setzt voraus, dass die hierarchischen Pläne der einzelnen Agenten intern konsistent sind, d. h. dass sie die *KannAufJedeWeise*-Eigenschaft besitzen. Während der Ausführung irgendeiner Aufgabe eines Planes irgendeines Agenten kann somit kein Konflikt auftreten, der nicht von einer Aufgabe eines anderen Agenten verursacht wird. Der Koordinationsalgorithmus kann einfach zu einem nebenläufigen hierarchischen Einzelagenten-Planungsalgorithmus modifiziert werden, indem diese Annahme fallen gelassen wird. Damit dies möglich ist, muss der *KannAufJedeWeise*-Algorithmus erweitert werden, um sicherzustellen, dass jeder Plan in *pläne* intern konsistent ist. Diese Information wird während der Berechnung der zusammengefassten Bedingungen gewonnen. Wenn irgendeine zusammengefasste Bedingung überschrieben werden kann, dann ist die Elternaufgabe intern inkonsistent und wird entsprechend markiert. Während der Planung werden intern nicht konsistente Aufgaben solange zerlegt, bis die Konflikte auf niedrigerer Ebene aufgelöst werden können. Wird eine Bedingung gefunden, die überschrieben werden muss, dann ist die Elternaufgabe inkonsistent und  $\neg$ *KönnteAufMancheWeise* ist wahr. Hierdurch wird der Planer zur Auswahl eines anderen *Oder*-Unterplans veranlasst, um den inkonsistenten Zweig zu vermeiden. Ist kein solcher *Oder*-Unterplan verfügbar, erkennt der Planer zuverlässig, dass keine Lösung für das Problem existiert. Mit dieser Modifikation kann der Algorithmus zur hierarchischen Verfeinerungsplanung für einen einzelnen Agenten verwendet werden.



## Planungsstrategien

Neben dem Auffinden von konfliktfreien oder koordinierten Plänen auf abstrakten Ebenen, eignen sich zusammengefasste Planinformationen nach [28] auch zur Lenkung der Suche, indem Zweige im Suchraum, die zu inkonsistenten oder suboptimalen koordinierten Lösungen führen, vermieden werden können. Auf abstrakter Ebene können inkonsistente Pläne durch eine Überprüfung, ob *KönnteAufMancheWeise* falsch ist, verworfen werden. In Bezug auf die Anzahl der expandierten Suchzustände schneidet die Anwendung dieses Verfahrens mindestens genauso gut ab wie seine Nichtverwendung. Das Beschneiden des Suchraums auf abstrakter Ebene führt jedoch zu einer deutlichen Verringerung des Suchaufwands.

Eine weitere Strategie besteht darin, zuerst diejenigen Pläne zu expandieren, die in die meisten Konflikte involviert sind. Für die Vollständigkeit des Koordinationsalgorithmus ist es unerheblich, in welcher Reihenfolge die Suchzustände expandiert werden, solange sie alle irgendwann expandiert werden, wenn die entsprechenden Suchzweige nicht verworfen werden können. Die Anwendung dieser EMTF-Heuristik (engl. *expand most threats first* (EMTF)) zielt darauf ab, die Suche durch die Hierarchie zu denjenigen Unterplänen zu leiten, die Konflikte verursachen, um diese möglichst schnell auflösen zu können. Diese Heuristik ähnelt einer am meisten beschränkten Variablen-Heuristik in *Constraint Satisfaction* Problemen.

Eine weitere Heuristik, die parallel zur EMTF-Heuristik angewendet werden kann, ist die Auswahl von Suchzuständen mit den wenigsten Konflikten. Diese Heuristik wird als CFTF (engl. *choose fewest threats first* (CFTF)) bezeichnet. Hierbei ordnet die Suche die Suchzustände in der Warteschlange aufsteigend nach der Anzahl der Konflikte an, was einer am wenigsten beschränkten Variablen-Heuristik im *Constraint Satisfaction* Ansatz entspricht. Durch die Auflösung der Bedrohungen von Suchzuständen mit den wenigsten Konflikten, wird das schnellere Auffinden von Lösungen angestrebt.

Die EMTF-Heuristik eignet sich somit zur Sortierung von zu expandierenden *Und*-Unterplänen, während sich die CFTF-Heuristik zur Sortierung von auszuwählenden *Oder*-Unterplänen anbietet.

Neben den vorstehend genannten Heuristiken zum Finden von optimalen Lösungen werden in [27] auch Kosten für abstrakte Lösungen verwendet, um Zweige des Suchraums zu verwerfen, deren minimale Kosten größer als die maximalen Kosten der aktuell besten Lösung sind. Dies ist die Aufgabe der *Dominiert*-Funktion in Algorithmus 5.3. Hierfür wird vorausgesetzt, dass sich die verwendete Kosten-Nutzen-Information über Hierarchien von Aktionen zerlegen lässt und die Kosten einer abstrakten Aktion eine Funktion ihrer Zerlegungen sind. Ein Beispiel hierfür ist die Minimierung der Dauer der koordinierten Pläne. Hierfür können die zusammengefassten Planinformationen die minimalen und maximalen Dauern für die möglichen Zerlegungen wie in Abschnitt 5.3.3 beschrieben nachverfolgen.

### 5.3.5. Serialisierung

Bei der Serialisierung werden die koordinierten Flexiblen Hierarchischen Pläne in eine XML-Repräsentation überführt, die von der semantischen Missionssteuerung zur Verarbeitung und Ausführung der Pläne verwendet wird. Die entwickelte XML-Repräsentation wird in Abschnitt 7.4.3 vorgestellt.

### 5.4. Planausführung

Die serialisierten koordinierten Flexiblen Hierarchischen Pläne werden vom *Manager* an die *Ausführungseinheit* übergeben (siehe Abbildung 3.4). Die Ausführungseinheit deserialisiert die koordinierten Flexiblen Hierarchischen Pläne und erzeugt eine entsprechende objektorientierte Repräsentation für die weitere Verarbeitung. Anschließend werden die Unterpläne der einzelnen FHiPs gemäß den angegebenen Ordnungsbedingungen (teil-)geordnet. Die Abarbeitung der geordneten koordinierten Pläne erfolgt gemäß einer Tiefensuchstrategie ähnlich wie bei Flexiblen Programmen (siehe Abschnitt 2.3.2). Die geordneten koordinierten Pläne entsprechen dabei einem Baum, dessen innere Knoten von *Oder-* und *Und-FHiPs* und dessen Blattknoten von *elementaren FHiPs* gebildet werden. Der Wurzelknoten des Baumes besteht aus einem speziellen *Und-FHiP*, dessen Kinder die koordinierten Flexiblen Hierarchischen Pläne auf oberster Ebene sind. Ausgehend von diesem Wurzelknoten wird im Baum rekursiv absteigend nach den jeweils nächsten ausführbaren *elementaren FHiPs* gesucht. Bei den nichtelementaren Aufgaben (*Oder-Fhips*) werden hierbei die zusammengefassten Vorbedingungen sowie die zusammengefassten Ressourcenverbräuche der nach der Koordination verbliebenen Zerlegungsmethoden (*Und-FHiPs*) geprüft. Unter den anwendbaren Zerlegungsmethoden wird diejenige mit der besten Bewertung ausgewählt. Bei den elementaren Aufgaben (*elementare FHiPs*) wird die Ausführbarkeit anhand der spezifizierten Vorbedingungen und des angegebenen Ressourcenverbrauchs geprüft. Die ausführbaren *elementaren FHiPs* werden von der *Ausführungseinheit* an die *Ausführungskomponenten* zur Abarbeitung übergeben. Der Erfolg der Ausführung wird von der *Ausführungseinheit* überwacht. Sobald entsprechende Statusmeldungen in der *Ausführungseinheit* eintreffen, werden die erfolgreich ausgeführten *elementaren FHiPs* als abgeschlossen markiert. Von diesen ausgehend wird anschließend rekursiv im Baum aufsteigend nach den nächsten noch nicht als abgeschlossen markierten *nichtelementaren FHiPs* gesucht (entspricht dem *Auftauchen*-Prozess bei Flexiblen Programmen). Diese bilden den Ausgangspunkt für eine erneute rekursiv im Baum absteigende Suche nach den nächsten ausführbaren *elementaren FHiPs* (entspricht dem *Abtauchen*-Prozess bei Flexiblen Programmen). Die beiden Schritte werden im Wechsel solange durchgeführt, bis die koordinierten Flexiblen Hierarchischen Pläne auf oberster Ebene abgeschlossen sind.

### 5.4.1. Umgang mit Fehlern und Ausnahmesituationen

Treten während der Ausführung von elementaren Aufgaben Fehler auf, wird dies der *Ausführungseinheit* von der jeweiligen *Ausführungskomponente* entsprechend signalisiert. Die Ausführungseinheit verringert daraufhin die Verfügbarkeit der elementaren Aufgabe. Liegt die Verfügbarkeit über einem definierten Schwellenwert, so wird erneut versucht, die fehlgeschlagene elementare Aufgabe auszuführen. Wird der genannte Schwellenwert unterschritten, ist die elementare Aufgabe endgültig fehlgeschlagen und der entsprechende Unterplan, welcher die Aufgabe enthält, wird als fehlgeschlagen markiert. Stehen weitere Methoden zur Erreichung von übergeordneten nichtelementaren Aufgaben zur Verfügung, werden diese entsprechend ihrer Bewertung ausgewählt und zur Verfeinerung der jeweiligen nichtelementaren Aufgabe verwendet. Ist keine der anwendbaren Zerlegungsmethoden erfolgreich, wird der gesamte koordinierte Plan als fehlgeschlagen markiert und die Ausführung abgebrochen. Dies wird dem *Manager* von der *Ausführungseinheit* entsprechend signalisiert. Dieser kann daraufhin versuchen, das Problem durch Neuplanung zu lösen.

## 5.5. Einordnung des Planungsansatzes

Im Folgenden wird der vorgestellte Planungsansatz anhand der in Abschnitt 5.1 beschriebenen Anforderungen bewertet. Dazu werden in Tabelle 5.4 die eingangs definierten Kriterien verwendet, um Soll und Ist bezüglich der Anforderungen gegenüberzustellen. Erfüllte Anforderungen sind dabei mit ✓, nur teilweise bzw. indirekt erfüllte Anforderungen mit ○ gekennzeichnet.

Charakteristik	Soll	Ist
Dynamische Welt	✓	○
Mehrere Agenten	×	×
Unvollständige und unsichere Informationen	✓	○
Externe Informationsquellen	✓	○
Zeitliche Dauer	✓	✓
Zeitliche Beschränkungen	✓	✓
Überlappende Aktionen	✓	✓
Asynchrone Aktionen	✓	×
Numerische Berechnungen	✓	○
Ressourcen	✓	✓
Inferenzregeln und abgeleitete Effekte	✓	○
Beschränkung von Zustandstrajektorien	×	×
Planmetriken	✓	✓
Weiche Ziele	✓	✓
Präferenzen bezüglich Zustandstrajektorien	×	×

Tab. 5.4.: Erfüllung der Anforderungen an den Planungsansatz.

Mit Hilfe der gewählten Planrepräsentation in Form von Flexiblen Hierarchischen Plänen

können viele der genannten Anforderungen direkt abgedeckt werden. Dies trifft etwa auf die Berücksichtigung von Ausführungsdauern von Aktionen, auf zeitliche Beschränkungen in Form von Ordnungsrelationen und auf nebenläufige, sich überlappende Aktionen zu. Darüber hinaus wird der Ressourcenverbrauch von Aktionen berücksichtigt und es können entsprechende numerische Berechnungen durchgeführt werden. Numerische Berechnungen allgemeiner Art werden hingegen nicht unterstützt. Mit Hilfe der zusammengefassten Planinformationen können außerdem vielfältige Planmetriken bezogen auf den Ressourcenverbrauch und die Ausführungsdauer von Plänen definiert werden, welche mit Hilfe der *Dominiert*-Funktion im Koordinations- und Verfeinerungsalgorithmus Eingang finden. Weiterhin können für die Aufgaben des initialen Aufgabennetzwerks Prioritäten definiert werden und es kann angegeben werden, ob es sich um optionale Aufgaben und somit um weiche Ziele handelt. Asynchrone Aktionen können hingegen mit dem gewählten Planungsansatz nicht berücksichtigt werden, da dies bei der Berechnung der zusammengefassten Planinformationen für Zeitdauern (siehe Abschnitt 5.3.3) ohne zusätzliche Annahmen zu undefinierten Ergebnissen führen würde.

Einige der Anforderungen werden nicht vom Planungsansatz direkt abgedeckt, sondern durch die iterative Verzahnung von Planung und Ausführung sowie die Verbindung mit der Wissensbasis. Dies trifft insbesondere auf die Verwendung externer Informationsquellen zu. Die einzige Informationsquelle, welche dem Planer zur Verfügung steht, ist die Wissensbasis. Diese wird jedoch von den Komponenten der semantischen Missionssteuerung laufend mit entsprechenden Informationen über den aktuellen Zustand der Welt versorgt. Mit dynamischen Veränderungen der Welt und unsicheren oder fehlerhaften Informationen über den Zustand der Welt kann das Gesamtsystem dahingehend umgehen, dass erst zur Laufzeit unter den verbliebenen Methoden zur Erfüllung von Aufgaben ausgewählt wird. Dabei wird sowohl die Anwendbarkeit jeder Methode als auch die Ausführbarkeit jeder Aktion anhand des Weltzustandes zur Laufzeit erneut überprüft. Existiert keine mögliche Ausführung des Plans oder schlägt die Ausführung wiederholt fehl, so wird darauf mit entsprechender Neuplanung reagiert.

Unvollständige Informationen über den Weltzustand können unter Umständen mit Hilfe von Inferenzregeln ergänzt werden (siehe Abschnitt 5.6). Dabei bleibt dieser Schritt jedoch wie bei der Bestimmung von abgeleiteten Effekten vor dem Planer verborgen, da er bereits in der Wissensbasis stattfindet und der Planer nur das Endergebnis in Form des jeweiligen Planungsproblems erhält.

**Vorteile des gewählten Planungsansatzes** Der gewählte Planungsansatz verfügt über eine Reihe von Eigenschaften, welche ihn gegenüber anderen HTN-Planungsansätzen auszeichnen. So basiert er etwa auf einem formalen Modell für hierarchische Pläne mit zeitlicher Ausdehnung (siehe [28]). Darüber hinaus beinhaltet er Algorithmen zur Berechnung von zusammengefassten Bedingungen sowie zusammengefassten Ressourcenverbräuchen und zur Verwendung dieser zusammengefassten Informationen zur Bestimmung von potentiellen und

definitiven Interaktionen zwischen abstrakten Aufgaben. Für diese kann bewiesen werden (siehe [28]), dass sie alle Bedingungen und Effekte, die mit einem abstrakten Operator verbunden sind, korrekt abbilden, indem sie modale Informationen darüber enthalten, ob Bedingungen gültig sein müssen oder gültig sein können und ob sie während der gesamten Operation oder nur zeitweilig gelten. Aus diesem Grund müssen Bedingungen und Ressourcenverbräuche nur für elementare Aufgaben angegeben werden, wodurch die Erstellung von Planungsdomänen deutlich vereinfacht wird. Basierend auf den zusammengefassten Planinformationen werden zuverlässige und vollständige Algorithmen zur hierarchischen Koordination und Verfeinerung verwendet, welche es ermöglichen, Konflikte zwischen Teilplänen bereits auf abstrakten Ebenen zu erkennen und aufzulösen. Da die resultierenden koordinierten Pläne garantiert auf jede mögliche Weise verfeinert werden können, wird zudem eine flexible Planausführung ermöglicht, welche die Notwendigkeit zur Neuplanung reduziert. Darüber hinaus können die zusammengefassten Planinformationen auch mit Hilfe der vorgestellten Planungstrategien (siehe Abschnitt 5.3.4) zur Lenkung der Suche und zur Einschränkung des Suchraums verwendet werden, wodurch bis zu doppelt exponentielle Beschleunigungen der Verfeinerungsplanung erzielt werden können (siehe [28]).

## 5.6. Synergetische Vernetzung von Planung und Inferenz

Neben den im vorhergehenden Abschnitt genannten Vorteilen des gewählten Planungsansatzes verspricht auch die Kombination von Planung und Inferenz Synergiepotential. In [41] wird ein Überblick über erste Arbeiten in diesem Bereich gegeben. Im Fokus steht dabei die Kombination von Planungstechniken mit ausdrucksstarker Wissensrepräsentation in Form von Beschreibungslogiken, um Schlussfolgerungen bezüglich Aufgaben, Plänen und Zielen zu ziehen. Insbesondere werden Beispiele dafür genannt, wie semantische Beschreibungen von Aktionen, Plänen und Aufgaben während der Planerzeugung, der Planerkennung oder Planevaluierung genutzt werden können, um schwierigere und anspruchsvollere Aufgaben zu lösen als mit konventionellen Planungstechniken möglich ist.

In [46], [47] und [48] wird ein Ansatz beschrieben, der eine auf Beschreibungslogiken basierende Inferenzmaschine (Pellet<sup>12</sup>) mit einem HTN-Planer (JSHOP2<sup>13</sup>) kombiniert. Hierzu werden die für die HTN-Planung wesentlichen Konzepte *Planungsdomäne*, *Planungsproblem*, *Methode* und *Operator* in OWL DL modelliert. Die Methoden und Operatoren werden dabei durch Strings charakterisiert, welche eine entsprechende Beschreibung in der Syntax von JSHOP2 beinhalten. Nach Vorgabe eines Ziels in Form einer Methode durch den Benutzer, werden automatisch eine entsprechende Planungsdomäne und ein entsprechendes Planungsproblem erzeugt, welche nur die zur Lösung des Planungsproblems notwendigen Informationen

---

<sup>12</sup>Pellet: <http://clarkparsia.com/pellet/>

<sup>13</sup>JSHOP2: <http://www.cs.umd.edu/projects/shop/>

enthalten. Der initiale Zustand wird dabei anhand des Weltmodells bestimmt. Methoden und Operatoren enthalten hierfür spezielle Anfragestrings, welche die für die Methoden und Operatoren relevanten Zustandsfakten spezifizieren. Außerdem kann vom Benutzer eine zusätzliche Anfrage gestellt werden (zum Beispiel mit SPARQL), um das Planungsproblem weiter einzuschränken. Hierdurch wird die Größe des Planungsproblems reduziert und somit der Planungsprozess beschleunigt. Die mit Hilfe von JSHOP2 erzeugten Pläne werden anschließend ebenfalls in der Wissensbasis gespeichert, um den Neuplanungsbedarf im Falle eines Fehlers während der Planausführung zu minimieren.

Um den Planungsprozess im Rahmen dieser Arbeit so schlank und effizient wie möglich zu gestalten, sind für die verschiedenen Inspektions- und Navigationsaufgaben bereits entsprechende Planungsprobleme in der Wissensbasis vorkonfiguriert. Diese beinhalten insbesondere entsprechende Planungsdomänen und Zustände, welche die benötigten Aufgaben und Zerlegungsmethoden sowie die zur Beschreibung des Weltzustands benötigten Prädikate und Ressourcen benennen. Wird der Planer vom Manager aufgerufen, so werden in der Initialisierungsphase des Planers (siehe Abschnitt 5.3.2) nur die im entsprechenden Planungsproblem spezifizierten Aufgaben und Methoden sowie Prädikate und Ressourcen eingelesen. Auf diese Weise werden vom Planer nur die wirklich benötigten Informationen bei der Lösung des Planungsproblems berücksichtigt, obwohl die Wissensbasis an sich deutlich mehr Aufgaben und Zerlegungsmethoden sowie Prädikate und Ressourcen umfassen kann.

Die Steigerung der Effizienz durch Einschränkung der Planungsprobleme bildet jedoch sicherlich nur einen Ausgangspunkt für weitere Arbeiten in diesem Bereich. Interessante Themen umfassen etwa die Vervollständigung der Beschreibung des Weltzustands durch Schlussfolgerungen, die Lösung von komplexeren Planungsproblemen durch Ausnutzung der Inferenz sowie die Plausibilitätsprüfung während der Ausführung.

### 5.7. Zusammenfassung

In diesem Kapitel wurde auf das entwickelte Konzept zur Erstellung und Ausführung von Inspektionsplänen eingegangen. Hierfür wurden zunächst die Anforderungen an einen entsprechenden Planungsansatz mit Hilfe von geeigneten Kriterien charakterisiert. Anschließend wurde auf die im Rahmen dieser Arbeit entwickelte Repräsentation von Inspektionsplänen, so genannte *Flexible Hierarchische Pläne* (FHiPs), eingegangen. Diese kombinieren Elemente von *Flexiblen Programmen* (FPs) [60] und *Concurrent Hierarchical Plans* (CHiPs) [28] und erweitern sie um zusätzliche Komponenten. Darauf aufbauend wurde die Planerzeugung, bestehend aus den vier Teilschritten *Initialisierung*, *Vorverarbeitung*, *Koordination* und *Serialisierung*, erläutert. Mit Hilfe von so genannten *zusammengefassten Planinformationen* [28] können potentielle Konflikte zwischen Teilplänen bereits auf abstrakten Ebenen erkannt werden und mit einem nebenläufigen hierarchischen Koordinations- und Verfeinerungsalgorithmus [28] aufge-

löst werden. Im Anschluss wurden die Besonderheiten der Ausführung der generierten Pläne vorgestellt. Dabei wurde insbesondere auf den Umgang mit Fehler- und Ausnahmesituationen eingegangen. Abschließend wurde der gewählte Planungsansatz im Hinblick auf die eingangs definierten Anforderungen eingeordnet und es wurden die potentiellen Synergien einer Vernetzung von Planung und Inferenz diskutiert.





## 6. Aktive Untersuchung interessierender Entitäten

Um eine möglichst hohe Qualität der Inspektionsergebnisse zu erzielen, sollte ein autonomer Inspektionsroboter gefundene interessierende Entitäten aktiv und datengetrieben untersuchen. Sind die Ergebnisse der Datenauswertung unsicher, sollten weitere Untersuchungen durchgeführt werden, bis eine ausreichende Konfidenz erreicht ist. Dies kann etwa die Durchführung zusätzlicher Messungen aus anderen Perspektiven, den Einsatz spezieller Sensorik oder die Verwendung weiterer Datenauswertungsverfahren umfassen. Die Untersuchung sollte dabei unter Berücksichtigung der zur Verfügung stehenden Ressourcen so effizient wie möglich durchgeführt werden, um eine möglichst hohe Reichweite des Systems zu erzielen.

In diesem Kapitel wird auf die Verfahren zu aktiven Untersuchung von interessierenden Entitäten eingegangen. Dazu werden zunächst die Anforderungen an die Untersuchung von interessierenden Entitäten definiert. Anschließend wird ein Überblick über den strukturellen Ablauf der aktiven Untersuchung gegeben. Daran anknüpfend werden die Verfahren zur Fusion und Bewertung der Datenauswertungsergebnisse sowie zur Auswahl und Priorisierung von Inspektionsaufgaben für die weitere Untersuchung vorgestellt. Ferner wird die Adaption der bestehenden Inspektionspläne durch iterative Neuplanung beschrieben. Abschließend wird auf Möglichkeiten zur kontinuierlichen Anpassung der Regelbasis zur Auswahl von Inspektionsaufgaben eingegangen.

### 6.1. Anforderungen

Die während der Inspektion mit Hilfe der Sensoren gewonnenen Daten sind mit Unsicherheiten behaftet. Aus diesem Grund sind die Ergebnisse der Datenauswertung ebenfalls unsicher und sollten in einer geeigneten Form repräsentiert werden (siehe Abschnitt 4.3.2). Für die Fusion der Datenauswertungsergebnisse sollte nach Möglichkeit ein probabilistisches Verfahren zum Einsatz kommen, welches es erlaubt, auf mathematisch fundierte Weise zu berechnen, wie groß die Wahrscheinlichkeiten für das tatsächliche Vorliegen der einzelnen Entitätsklassen sind. Basierend auf diesen berechneten Wahrscheinlichkeiten muss anschließend entschieden werden, ob und wenn ja wie die gefundenen interessierenden Entitäten weiter untersucht werden. Diese Entscheidung sollte situationsabhängig und unter Nutzung des zur Verfügung stehenden Erfahrungs- und Expertenwissens getroffen werden. Darüber hinaus sind die dem System zur Verfügung stehenden Ressourcen zu berücksichtigen. Es muss somit ein Kompromiss zwischen dem Nutzen der weiteren Untersuchung und den Kosten für die Durchführung der Untersuchung

gefunden werden, um eine möglichst optimale Balance zwischen der Qualität der Inspektionsergebnisse und der Reichweite des autonomen Systems zu erreichen.

### 6.2. Struktureller Ablauf der aktiven Untersuchung

Der strukturelle Ablauf der aktiven Untersuchung von interessierenden Entitäten ist in Abbildung 6.1 dargestellt. Während der Inspektion wird zunächst nach Auffälligkeiten in den Sensordaten gesucht. Wird eine solche Auffälligkeit gefunden, wird der entsprechende Bereich in den Sensordaten segmentiert. Die so gewonnene interessierende Region wird anschließend mit Hilfe eines merkmalsbasierten Klassifikationsansatzes daraufhin überprüft, ob es sich um eine interessierende Entität oder um eine irrelevante Auffälligkeit handelt. Handelt es sich um eine interessierende Entität, wird ebenfalls mit Hilfe eines merkmalsbasierten Klassifikationsansatzes die Entitätsklasse bestimmt. Im Anschluss wird die Konfidenz des Ergebnisses überprüft. Ist diese ausreichend, wird das Ergebnis akzeptiert und die Inspektion fortgesetzt. Ist dies nicht der Fall, wird eine geeignete Inspektionsaufgabe zu weiteren Untersuchung der interessierenden Entität ausgewählt. Steht eine solche zur Verfügung, wird der bestehende Inspektionsplan mittels Neuplanung entsprechend adaptiert und die gewählte Inspektionsaufgabe wird ausgeführt. Andernfalls wird die Untersuchung der interessierenden Entität abgebrochen. Nach der Ausführung der gewählten Inspektionsaufgabe werden die Ergebnisse der Datenauswertung fusioniert und anschließend erneut hinsichtlich ihrer Konfidenz bewertet.

### 6.3. Verfahren zur aktiven Untersuchung

Die aktive Untersuchung von interessierenden Entitäten gliedert sich in die folgenden drei Schritte:

**Fusion der Datenauswertungsergebnisse** Bestimmung der Wahrscheinlichkeiten für das tatsächliche Vorliegen der einzelnen Entitätsklassen anhand der bisher verfügbaren Datenauswertungsergebnisse (Berechnung der Hypothesen der interessierenden Entität, siehe Abschnitt 4.3.2).

**Bewertung der fusionierten Datenauswertungsergebnisse** Bestimmung der Konfidenz der fusionierten Datenauswertungsergebnisse und Entscheidung darüber, ob die interessierende Entität weiter untersucht werden soll (Berechnung der Konfidenz und Inferenz des Inspektionsstatus der interessierenden Entität, siehe Abschnitt 4.3.2).

**Auswahl und Priorisierung von Inspektionsaufgaben** Auswahl der am besten geeigneten Inspektionsaufgabe zur weiteren Untersuchung der interessierenden Entität sowie Festlegung der zugehörigen Priorität (Inferenz der Inspektionsaufgabe und der Priorität der interessierenden Entität, siehe Abschnitt 4.3.2).

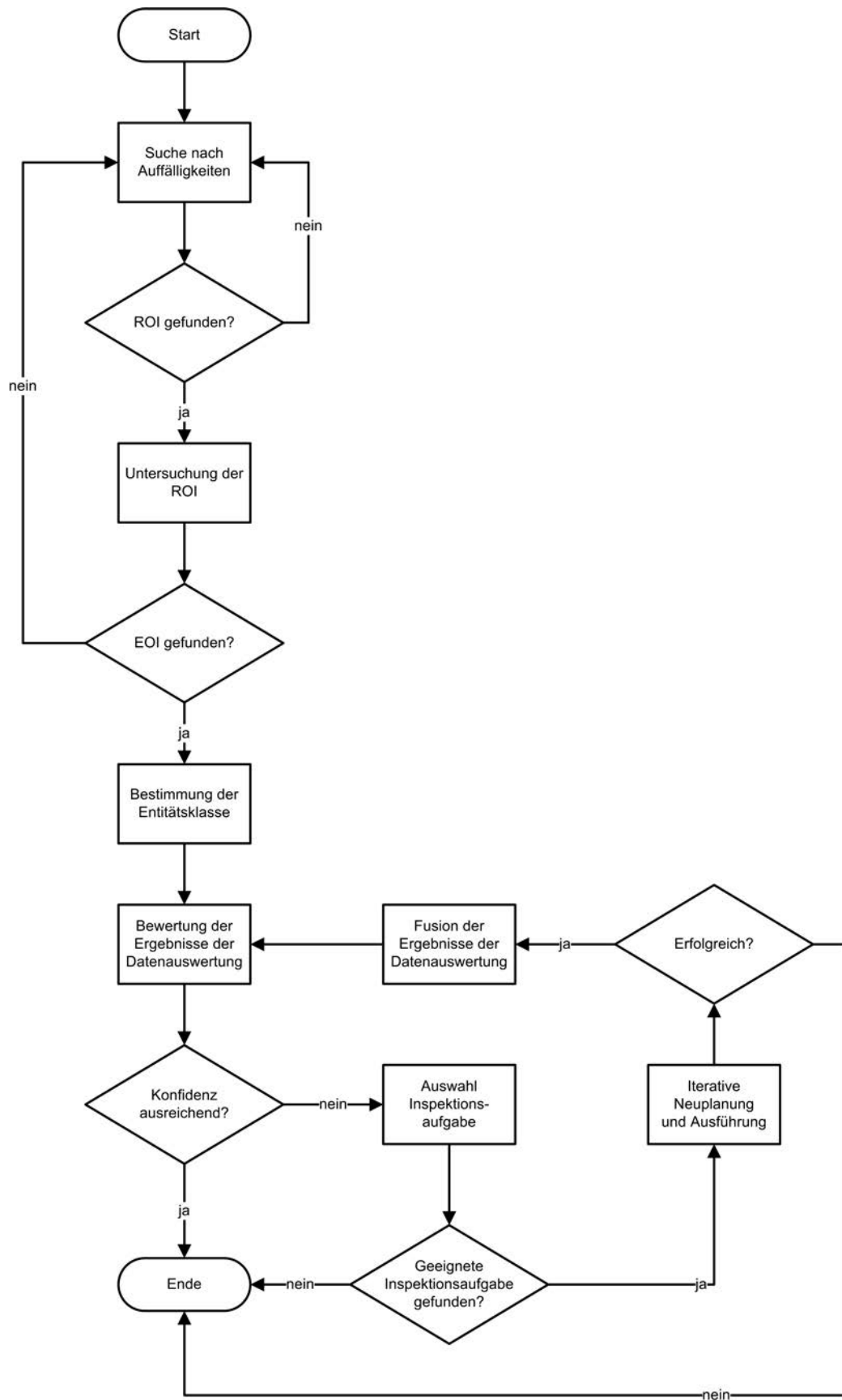


Abb. 6.1.: Ablauf der aktiven Untersuchung interessierender Entitäten.

Im Folgenden werden die in den genannten Schritten jeweils verwendeten Verfahren näher erläutert.

### 6.3.1. Fusion der Datenauswertungsergebnisse

Die Fusion der einzelnen Datenauswertungsergebnisse wird mit Hilfe von Bayes'schen Netzwerken durchgeführt [113]. Die Struktur der verwendeten Bayes'schen Netzwerke folgt dabei dem in Abbildung 6.2 angegebenen generischen Aufbau. Die Netzwerke enthalten im Wesentlichen drei Arten von Knoten:

1. Einen Knoten für die gesuchte tatsächliche Entitätsklasse. Diese Zufallsvariable wird im Folgenden mit  $X$  bezeichnet. Die möglichen Werte, welche die Zufallsvariable annehmen kann (also die Entitätsklassen), werden mit  $x_i, i = 1, \dots, m_x$  bezeichnet.
2. Jeweils einen Knoten für jede zur Bestimmung der Entitätsklasse verfügbare Inspektionsaufgabe. Diese Zufallsvariablen werden im Folgenden mit  $Y_i, i = 1, \dots, n_y$  bezeichnet. Die möglichen Werte, welche diese Zufallsvariablen annehmen können, werden mit  $y_{ij}, i = 1, \dots, n_y, j = 1, \dots, m_{y_i}$  bezeichnet.
3. Jeweils einen Knoten für jeden Einflussfaktor, der sich auf das Ergebnis einer Inspektionsaufgabe auswirkt. Diese Knoten werden im Folgenden mit  $Z_{ij}, i = 1, \dots, n_y, j = 1, \dots, n_{z_i}$  bezeichnet. Die möglichen Werte, welche diese Zufallsvariablen annehmen können, werden mit  $z_{ijk}, i = 1, \dots, n_y, j = 1, \dots, n_{z_i}, k = 1, \dots, m_{z_{ij}}$  bezeichnet.

Neben der Struktur der Bayes'schen Netzwerke sind vor allem die Tabellen mit den bedingten Wahrscheinlichkeiten (engl. *conditional probability table* (CPT)) wesentlich. Für die drei Arten von Knoten werden diese wie folgt gefüllt:

1. Der Knoten für die Entitätsklasse verfügt über keine Elternknoten. Daher wird für jede mögliche Entitätsklasse die a-priori-Wahrscheinlichkeit dieser Entitätsklasse eingetragen. Diese entspricht der statistischen Auftretenswahrscheinlichkeit der entsprechenden Entitätsklasse.
2. Die Elternknoten der Knoten für die Inspektionsaufgaben bestehen aus der tatsächlichen Entitätsklasse sowie aus den sich auf das Ergebnis der Inspektionsaufgabe auswirkenden Einflussfaktoren. Für jede mögliche Kombination der Werte der Elternknoten werden die bedingten Wahrscheinlichkeiten für die Ergebnisse der Inspektionsaufgabe gegeben diese Werte eingetragen. Im einfachsten Fall, in dem das Ergebnis der Inspektionsaufgabe direkt die Entitätsklasse angibt und durch keine Einflussfaktoren beeinflusst wird, entspricht die Tabelle der bedingten Wahrscheinlichkeiten der so genannten *Konfusionsmatrix*.

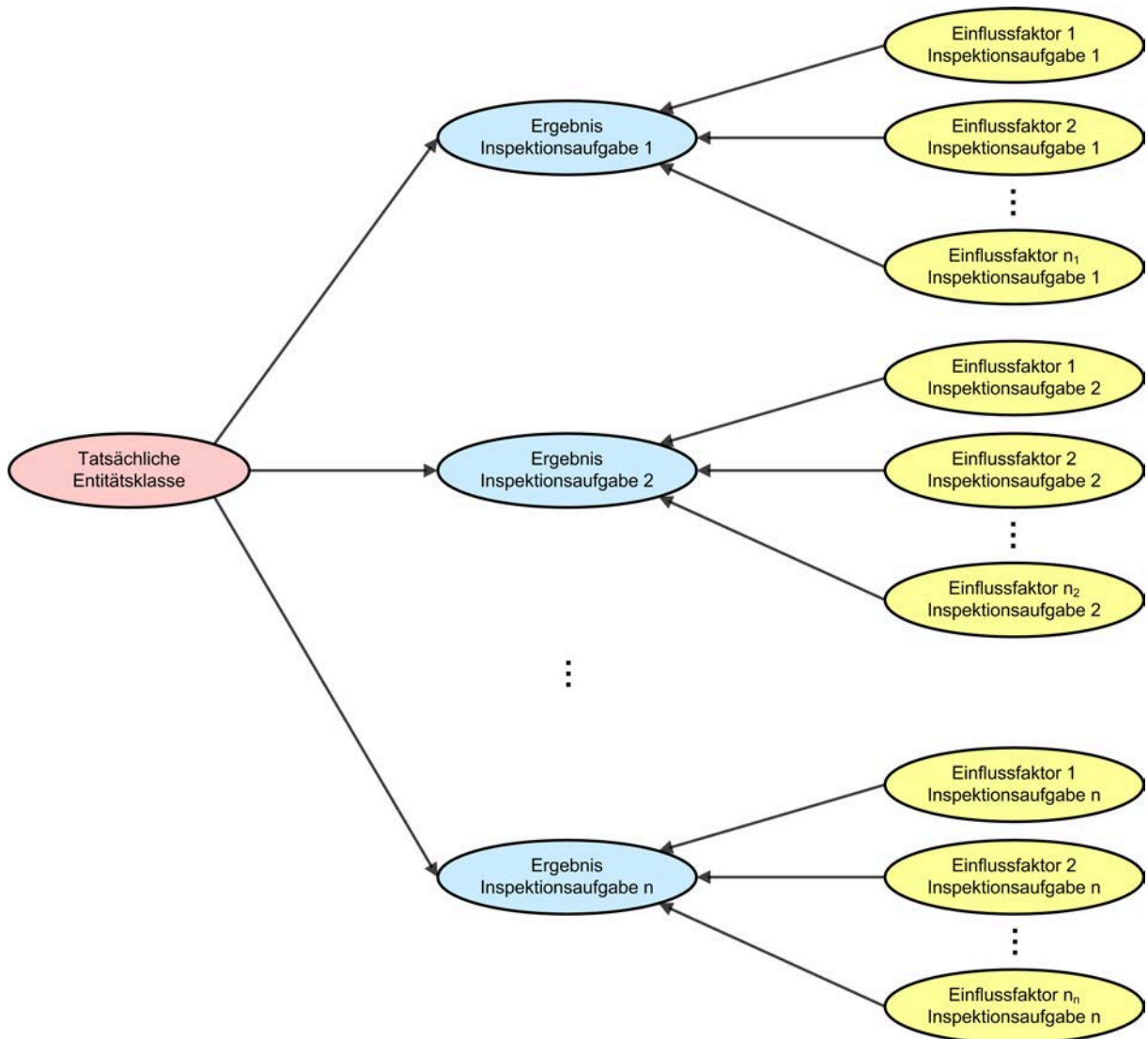


Abb. 6.2.: Generisches Bayes'sches Netzwerk zur Bestimmung der tatsächlichen Entitätsklasse anhand der zur Verfügung stehenden Inspektionsaufgaben und ihrer jeweiligen Einflussfaktoren.

- Die Knoten für die Einflussfaktoren verfügen über keine Elternknoten. Es werden daher die a-priori-Wahrscheinlichkeiten für die möglichen Werte der Einflussfaktoren eingetragen.

Zur Fusion der Datenauswertungsergebnisse kann nun mit den bekannten Inferenzalgorithmen für Bayes'sche Netzwerke (siehe [86]) die folgende bedingte Wahrscheinlichkeit berechnet werden:

$$P\left(X|Y_{a_1} = y_{a_1c_1}, \dots, Y_{a_{n_a}} = y_{a_{n_a}c_{n_a}}, Z_{a_1b_1} = z_{a_1b_1d_1}, \dots, Z_{a_{n_a}b_{n_b}} = z_{a_{n_a}b_{n_b}d_{n_b}}\right), \quad [6.1]$$

wobei die bisher durchgeführten Inspektionsaufgaben mit den Indizes  $a_k$  und die Einflussfaktoren mit bekannten Werten für diese Inspektionsaufgaben mit den Indizes  $a_k b_l$  bezeichnet

werden. Dabei gilt:

$$a_k \in \{1, \dots, n_y\}, \quad k = 1, \dots, n_a, \quad a_k \neq a_l \quad \forall k \neq l \quad [6.2]$$

$$b_k \in \{1, \dots, n_{z_i}\}, \quad k = 1, \dots, n_b, \quad b_k \neq b_l \quad \forall k \neq l \quad [6.3]$$

$$c_k \in \{1, \dots, m_{y_i}\}, \quad k = 1, \dots, n_a \quad [6.4]$$

$$d_k \in \{1, \dots, m_{z_{ij}}\}, \quad k = 1, \dots, n_b \quad [6.5]$$

Mit Hilfe eines solchen Bayes'schen Netzwerks kann also sukzessive die Wahrscheinlichkeit für die tatsächliche Entitätsklasse gegeben die Ergebnisse der bisher durchgeführten Inspektionsaufgaben und gegeben die bekannten Werte der Einflussfaktoren bestimmt werden. Wurde noch keine Inspektionsaufgabe durchgeführt, entsprechen die Wahrscheinlichkeiten für die tatsächliche Entitätsklasse den a-priori-Wahrscheinlichkeiten.

**Erweiterungsmöglichkeiten** Das in Abbildung 6.2 gezeigte generische Bayes'sche Netzwerk lässt sich bei Bedarf wie folgt erweitern:

1. Ist die Auftretenswahrscheinlichkeit für die tatsächliche Entitätsklasse von bestimmten Faktoren, wie zum Beispiel dem Auftretensort, abhängig, können diese Faktoren dem Bayes'schen Netzwerk in Form von Elternknoten der Entitätsklasse hinzugefügt werden.
2. Wird ein Einflussfaktor wiederum von bestimmten anderen Faktoren beeinflusst, so können diese dem Bayes'schen Netzwerk ebenfalls in Form von Elternknoten des entsprechenden Einflussfaktors hinzugefügt werden.

In beiden Fällen sind für die hinzugefügten Elternknoten die a-priori-Wahrscheinlichkeiten der jeweiligen Werte einzutragen. Darüber hinaus sind für die tatsächliche Entitätsklasse bzw. den jeweiligen Einflussfaktor die bedingten Wahrscheinlichkeiten gegeben die Werte der Elternknoten zu spezifizieren. Für die Berechnung der in Gleichung 6.1 angegebenen bedingten Wahrscheinlichkeit sind außerdem entsprechenden Bedingungen für die Zufallsvariablen der neu hinzugefügten Elternknoten zu ergänzen.

### 6.3.2. Bewertung der fusionierten Datenauswertungsergebnisse

Nachdem die Ergebnisse der einzelnen Inspektionsaufgaben mit Hilfe von Bayes'schen Netzwerken fusioniert worden sind, muss entschieden werden, ob die jeweilige interessierende Entität weiter untersucht werden soll oder ob das fusionierte Ergebnis eine ausreichende Konfidenz aufweist. Das fusionierte Ergebnis liegt dabei in Form von Wahrscheinlichkeiten für die einzelnen Entitätsklassen vor (siehe Abschnitt 6.3.1). Für das Konfidenzmaß wird die Entropie dieser Wahrscheinlichkeitsverteilung verwendet, da sie sich als Maß der Homogenität sehr gut dazu eignet, die in Abbildung 6.3 beispielhaft dargestellten Wahrscheinlichkeitsverteilungen klar

voneinander abzugrenzen. Ziel der aktiven Untersuchung ist es, ein möglichst klar ausgeprägtes Votum für eine einzelne Entitätsklasse zu erzielen. Sei  $X$  die Menge der Entitätsklassen. Dann

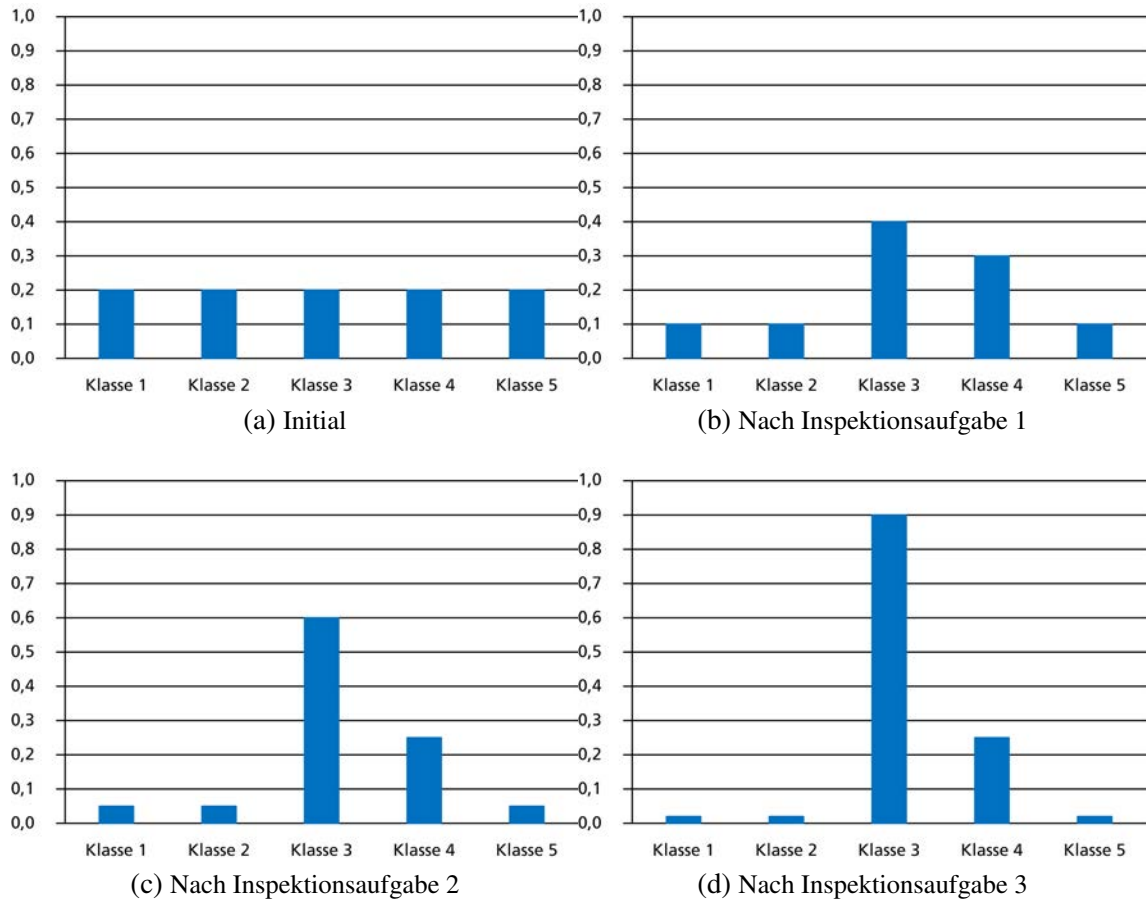


Abb. 6.3.: Beispielhafte Entwicklung der Wahrscheinlichkeiten für die tatsächlichen Entitätsklassen während der aktiven Untersuchung.

ist die Entropie  $E$  wie folgt definiert:

$$E = - \sum_{x \in X} p(x) \log_2 p(x), \quad [6.6]$$

wobei  $p(x)$  die Wahrscheinlichkeit für das Vorliegen der Entitätsklasse  $x \in X$  bezeichnet. Die Entropie nimmt ihren maximalen Wert  $E_{max}$  an, wenn alle Entitätsklassen gleich wahrscheinlich sind:

$$E_{max} = - \sum_{x \in X} \frac{1}{|X|} \log_2 \frac{1}{|X|} \quad [6.7]$$

$$= -\log_2 \frac{1}{|X|} \quad [6.8]$$

Darauf aufbauend wird die Konfidenz  $K$  wie folgt definiert:

$$K = \frac{E_{max} - E}{E_{max}} \quad [6.9]$$

Bei einer Gleichverteilung nimmt die Konfidenz den Wert 0 an. Geht die Wahrscheinlichkeit für das Vorliegen einer bestimmten Entitätsklasse gegen 1, so geht die Konfidenz ebenfalls gegen 1. Zur Beurteilung der auf diese Weise berechneten Konfidenz  $K$  wird ein regelbasiertes Verfahren verwendet: Liegt die Konfidenz über einem anhand von Regeln festgelegten Schwellenwert, wird das Untersuchungsergebnis akzeptiert. Andernfalls wird die Untersuchung fortgesetzt.

### 6.3.3. Auswahl und Priorisierung von Inspektionsaufgaben

Die Auswahl von Inspektionsaufgaben zur weiteren Untersuchung von interessierenden Entitäten erfolgt anhand der Regelbasis des semantischen Inspektionsmodells. Zur Erstellung der entsprechenden Regeln wird ein auf Entscheidungsbäumen basierendes Verfahren verwendet. Die Blattknoten des Entscheidungsbaums enthalten dabei die jeweils als nächstes auszuführende Inspektionsaufgabe. Die inneren Knoten enthalten so genannte Attributtests, mit deren Hilfe die jeweilige Inspektionssituation auf bestimmte Eigenschaften hin überprüft werden kann. Die Inspektionssituation wird dabei im Wesentlichen durch die Ergebnisse der einzelnen Datenauswertungsschritte (siehe Abschnitt 4.3.2) sowie durch Aussagen über den Zustand der Welt in Form von Prädikaten charakterisiert (siehe Abschnitt 4.3.1). Des Weiteren können auch die aktuell zur Verfügung stehenden Ressourcen mit einbezogen werden. Die Erstellung des Entscheidungsbaums kann entweder manuell von einem Experten vorgenommen werden oder mit Hilfe entsprechender Verfahren des Maschinellen Lernens (siehe etwa [67]) erfolgen. Letzteres erfordert zwar eine geeignete Lerndatenmenge, bietet jedoch den Vorteil, dass der Baum entsprechend des Informationsgewinns der einzelnen Attributtests aufgebaut wird. Der erstellte Entscheidungsbaum wird anschließend in eine Menge von Regeln konvertiert, indem für jeden möglichen Pfad durch den Baum eine entsprechende Regel erstellt wird. Die Vorbedingungen der Regeln entsprechen somit den Attributtests des zugehörigen Pfades im Entscheidungsbaum und die Konsequenz der als nächstes auszuführenden Inspektionsaufgabe.

Die Priorisierung der ausgewählten Inspektionsaufgabe erfolgt mit Hilfe geeigneter Heuristiken, die ebenfalls in Form von Regeln repräsentiert werden. So können etwa für die einzelnen Entitätsklassen Prioritäten angegeben werden, welche die Relevanz der jeweiligen Entitätsklasse für die aktuelle Inspektionsmission charakterisieren.

## 6.4. Iterative Adaption der bestehenden Inspektionspläne

Durch das in Abschnitt 3.2.3 dargestellte Zusammenwirken der einzelnen Komponenten der Steuerungsarchitektur ergibt sich ein iterativer Inspektionsprozess. In Abhängigkeit von den Er-



gebnissen der *Inspektionsdatenauswertung* und der *Navigationsdatenauswertung* werden dem *Manager* von der *Semantischen Inspektion* und der *Semantischen Navigation* weitere Missionsaufgaben vorgeschlagen. Diese werden vom *Manager* bei der Festlegung der im nächsten Teil der Mission durchzuführenden Aufgaben berücksichtigt, indem ein entsprechendes Planungsproblem erzeugt und an den Planer übergeben wird. Unter Berücksichtigung der zur Verfügung stehenden Ressourcen und der Prioritäten der einzelnen Missionsaufgaben wird vom Planer anschließend ein entsprechender Teilmissionsplan zur Erfüllung der ausgewählten Missionsaufgaben generiert. Mit der Ausführung dieses Plans beginnt ein neuer Zyklus des Regelkreises der autonomen Inspektion (siehe Abbildung 1.1). Mit Hilfe des vorgestellten iterativen Verfahrens kann sehr flexibel auf die jeweiligen Inspektionssituationen reagiert werden und es können in Abhängigkeit von den zur Verfügung stehenden Ressourcen bedarfsgerecht Inspektionsaufgaben zur weiteren Untersuchung von gefundenen interessierenden Entitäten in die bestehenden Inspektionspläne integriert werden.

## 6.5. Kontinuierliche Anpassung der Regelbasis

Die Auswahl der Inspektionsaufgaben zur weiteren Untersuchung von interessierenden Entitäten erfolgt wie in Abschnitt 6.3.3 beschrieben anhand der Regelbasis des semantischen Inspektionsmodells. Diese Regelbasis wird initial anhand des Fach- und Erfahrungswissens von menschlichen Experten erstellt. In einem nächsten Schritt ist jedoch die schritthaltende Adaption der Regelbasis an die im Laufe der autonomen Inspektionsmissionen gesammelten Erfahrungen wünschenswert. Hierfür sind verschiedene alternative Vorgehensweisen denkbar. Idealerweise wird diese Anpassung jedoch direkt vom System selbst vorgenommen. Da im Rahmen dieser Arbeit, wie in Abschnitt 6.3.3 erläutert, ein auf Entscheidungsbäumen basierender Ansatz zur Erstellung der Regelbasis genutzt wird, bietet es sich an, zur Adaption der Regelbasis ein inkrementelles Verfahren zum Lernen von Entscheidungsbäumen zu verwenden. Hierfür ist jedoch eine geeignete Lerndatenmenge erforderlich, welche durch entsprechende Attribute charakterisierten Inspektionssituationen die jeweils am besten geeignete Inspektionsaufgabe zuordnet.

Zur Gewinnung einer solchen Lerndatenmenge bietet sich die im Folgenden beschriebene Vorgehensweise an. Zunächst werden die im Rahmen der autonomen Inspektionsmissionen auftretenden Inspektionssituationen und die jeweils gewählten Inspektionsaufgaben protokolliert. Anschließend wird überprüft, welche der Inspektionsaufgaben nach ihrer Durchführung zu einer Verbesserung der Konfidenz der Datenauswertungsergebnisse geführt haben. Für diese wird jeweils ein entsprechendes Lernbeispiel erzeugt und der Lerndatenmenge hinzugefügt. Die Adaption der Regelbasis durch inkrementelles Lernen des korrespondierenden Entscheidungsbaums anhand der Lernbeispiele kann entweder schritthaltend während der Durchführung der Inspektionsmissionen erfolgen oder nach Beendigung der Inspektionsmissionen in einem

Schritt gesammelt durchgeführt werden. Letzteres ermöglicht auch ein semiautonomes Verfahren, bei dem die neuen Lernbeispiele vor der Aktualisierung der Regelbasis nochmals von einem menschlichen Experten überprüft werden.

### **6.6. Zusammenfassung**

In diesem Kapitel wurden zunächst die Anforderungen an die aktive Untersuchung von interessierenden Entitäten diskutiert. Anschließend wurde ein Überblick über den strukturellen Ablauf der aktiven Untersuchung gegeben, bevor die einzelnen Verfahren näher erläutert wurden. Zur Fusion der Datenauswertungsergebnisse wird ein generisches, auf Bayes'schen Netzwerken basierendes Verfahren eingesetzt, das ausgehend von den Ergebnissen der durchgeführten Inspektionsaufgaben und den Werten ihrer jeweiligen Einflussfaktoren die Wahrscheinlichkeiten dafür berechnet, dass die untersuchte interessierende Entität tatsächlich zu einer bestimmten Entitätsklasse gehört. Die Konfidenz der fusionierten Datenauswertungsergebnisse wird anschließend mittels eines entropiebasierten Konfidenzmaßes berechnet und mit einem anhand von Regeln festgelegten Schwellenwert verglichen. Liegt die Konfidenz über dem Schwellenwert, wird das Ergebnis akzeptiert. Andernfalls wird anhand der Regelbasis des semantischen Inspektionsmodells die nächste auszuführende Inspektionsaufgabe bestimmt. Zur Erstellung der Regelbasis kommt ein auf Entscheidungsbäumen basierendes Verfahren zum Einsatz. Nach der mit Hilfe von regelbasierten Heuristiken erfolgenden Priorisierung der ausgewählten Inspektionsaufgabe wird diese mittels iterativer Neuplanung in die bestehenden Inspektionspläne integriert. Abschließend wurden Möglichkeiten zur kontinuierlichen Anpassung der Regelbasis erörtert.

## 7. Umsetzung der semantischen Missionssteuerung auf LAURON IV

Das in den vorangegangenen Kapiteln vorgestellte Konzept der semantischen Missionssteuerung für autonome Inspektionsroboter wurde exemplarisch für die sechsbeinige Laufmaschine LAURON IV umgesetzt (siehe Abbildung 7.1). Als konkretes Inspektionsszenario wurde die Detektion und Klassifikation von Abfall in unstrukturiertem Gelände, wie zum Beispiel Wiesen, Wäldern und Böschungen von vielbefahrenen Straßen, Flüssen und Kanälen, gewählt.

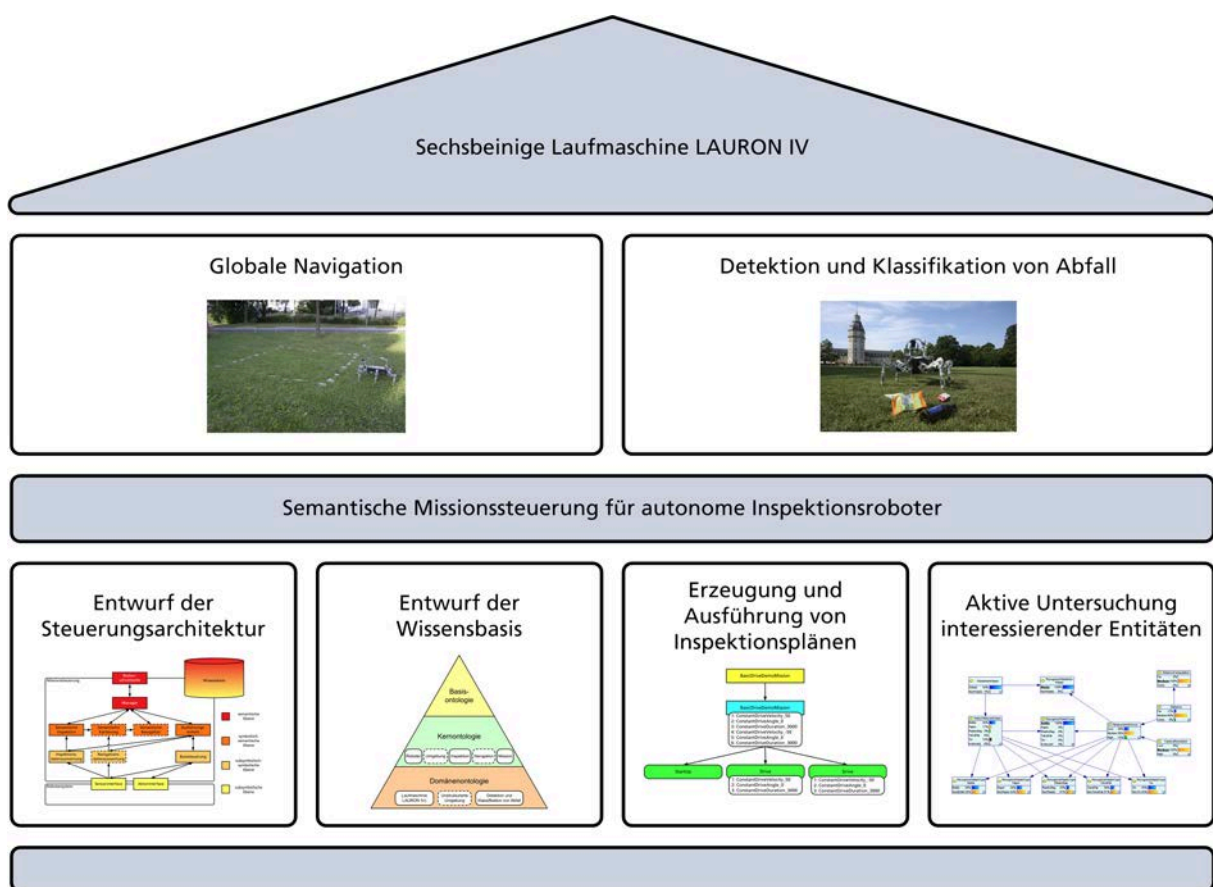


Abb. 7.1.: Übersicht über die im Rahmen dieser Arbeit entwickelten Komponenten.

In diesem Kapitel wird zunächst die sechsbeinige Laufmaschine LAURON IV vorgestellt. Dies umfasst einen kurzen Systemüberblick sowie eine Beschreibung der grundlegenden Eigenschaften und Fähigkeiten der Roboterplattform. Ausgehend von der verhaltensbasierten Basissteuerung werden die Fähigkeiten zur Lokalisation und Umweltmodellierung sowie zur lokalen, sensorgestützten Navigation beschrieben. Daran anknüpfend wird auf die im Rahmen dieser Arbeit entwickelten Fähigkeiten zur globalen Navigation eingegangen. Diese ermögli-

chen es, anhand einer grob aufgelösten globalen Karte Pfade zum Erreichen von vorgegebenen Zielpunkten und Regionen sowie zum systematischen Absuchen von Regionen zu planen und auszuführen. Des Weiteren wird auf die entwickelten Fähigkeiten zur bildbasierten Detektion und Klassifikation von Abfall in unstrukturiertem Gelände eingegangen. Diese umfassen ein aufmerksamkeitsbasiertes Verfahren zur Detektion auffälliger Bildbereiche sowie ein auf Farbtexturmerkmalen basierendes Verfahren zur Klassifikation der segmentierten Bildbereiche. Anschließend wird die Umsetzung der Kernkomponenten der semantischen Missionssteuerung erläutert. Zunächst werden die Realisierung der Steuerungsarchitektur und der Wissensbasis beschrieben. Dabei wird insbesondere auf die Modellierung der Fähigkeiten von LAURON IV und auf die Modellierung des Inspektionwissens für das gewählte Inspektionsszenario eingegangen. Schließlich wird die Realisierung des Planers zur Erzeugung und Ausführung von Inspektionsplänen beschrieben sowie die Umsetzung der aktiven Untersuchung von Abfallobjekten mit LAURON IV erläutert.

### 7.1. Die sechsbeinige Laufmaschine LAURON IV

Die an der Stabheuschrecke orientierte sechsbeinige Laufmaschine LAURON wurde am Forschungszentrum Informatik (FZI) entwickelt, um statisch stabiles Laufen in unstrukturiertem Gelände zu untersuchen [111]. Anfang der neunziger Jahre wurden in der Abteilung Interaktive Diagnose- und Servicesysteme (IDS) am FZI erste grundlegende theoretische und modellbildende Untersuchungen zum sechsbeinigen Laufen durchgeführt. Im Jahr 1994 wurde dann der erste lauffähige Roboter LAURON (**L**aufender **R**oboter **n**euronal gesteuert) als Ergebnis dieser Untersuchungen auf der CeBIT in Hannover der Öffentlichkeit vorgestellt. Die Forschung konzentrierte sich in den ersten Jahren primär auf das Laufen in unwegsamen und schwierigen Umgebungen. Über die Jahre und Robotergenerationen hinweg wurden neben der eigentlichen Robotersteuerungssoftware auch die Mechanik und die Sensorik stetig weiterentwickelt. Die aktuelle Laufmaschine, LAURON IV, wurde im Jahr 2005 fertiggestellt (siehe Abb. 7.2).

#### 7.1.1. Systemüberblick

Die Konstruktion der Laufmaschine LAURON orientiert sich am biologischen Vorbild der Stabheuschrecke. Wie dieses Vorbild besitzt sie sechs Beine an einem länglichen Zentralkörper, in dem die notwendige Steuerungselektronik untergebracht ist. Jedes der sechs identischen Beine besitzt einen federgedämpften Fuß und kann mit Hilfe von drei Gelenken bewegt werden. Zusätzlich kann die Blickrichtung des Kopfes durch zwei unabhängige Achsen (Schwenken und Neigen) verändert werden, so dass LAURON insgesamt über 20 Freiheitsgrade verfügt.

LAURON wurde mit zahlreichen Sensorsystemen ausgestattet. In jedem Fuß befinden sich 3D-Kraftsensoren und Federkraft-Messsysteme, die zusammen mit einer Motorstrommessung

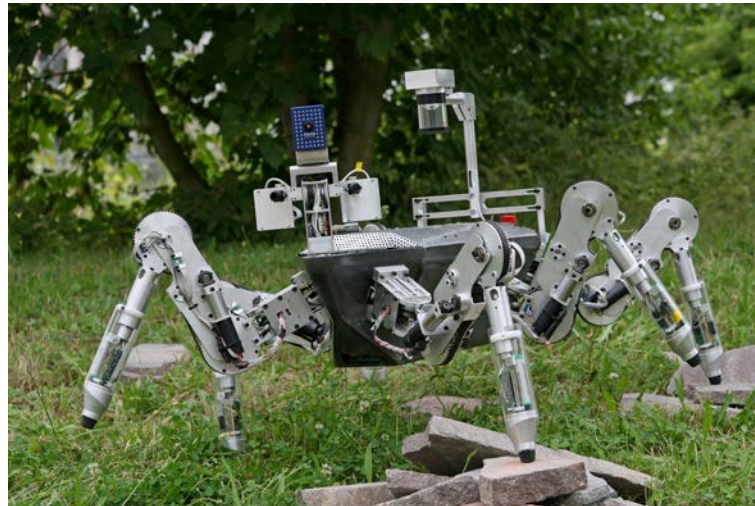


Abb. 7.2.: Die sechsbeinige Laufmaschine LAURON IV.

genutzt werden, um Kollisionen und den Kontakt mit dem Boden zu erkennen. Zur Bestimmung der Orientierung und der Position verfügt LAURON über ein Inertiales Navigationssystem (INS) und einen GPS-Sensor. Zwei Kamerasysteme, ein Stereokamerasystem auf dem Kopf und eine 360°-Kamera auf dem Rücken, liefern Informationen über die Umgebung des Roboters. Eine kleine und leichte Time-of-Flight Kamera (SwissRanger SR-3000), die sich auf dem Kopf befindet, ergänzt diese Informationen um detaillierte Tiefendaten. Die Gelenkwinkel der Beine werden durch hochpräzise optische Encoder erfasst. Zudem verfügt jeder Motor über einen hochauflösenden Encoder, der zusätzliche Informationen über die Gelenkwinkel liefert.

Die Bewegungen der sechs Beine und des Kopfes werden mit Hilfe von so genannten *Universal Controller Modules* (UCoMs) geregelt [81]. Jedes dieser sieben UCoMs verfügt unter anderem über einen eigenen DSP und FPGA. Alle UCoMs sind über einen CAN-Bus miteinander und mit dem Steuerrechner (PC/104-System) verbunden. Neben der PWM-basierten Motoransteuerung werden die UCoMs eingesetzt, um zahlreiche Sensoren (zum Beispiel 3D-Kraftsensoren, optische Gelenkencoder) auszulesen. Auf den UCoMs sorgt ein Geschwindigkeits-Kaskaden-Regler dafür, dass die von der verhaltensbasierten Steuerung erzeugten Beintrajektorien auch korrekt abgefahren werden. Ein Überblick über die gesamte Hardwarearchitektur kann Abb. 7.3 entnommen werden. Alle im Folgenden vorgestellten Komponenten der Steuerungssoftware wurden mit Hilfe des Softwarerahmenwerks MCA2 [100] realisiert.

### 7.1.2. Verhaltensbasierte Steuerung

Die auf LAURON IV eingesetzte verhaltensbasierte Steuerung erzeugt und überwacht sämtliche Bewegungstrajektorien der Laufmaschine. Sie setzt sich aus einer großen Anzahl einfacher Basisverhalten zusammen [57], die durch Interaktion und Fusion in der Lage sind, auch äußerst komplexe Aufgaben, wie das Überwinden von Hindernissen, zu bewältigen.

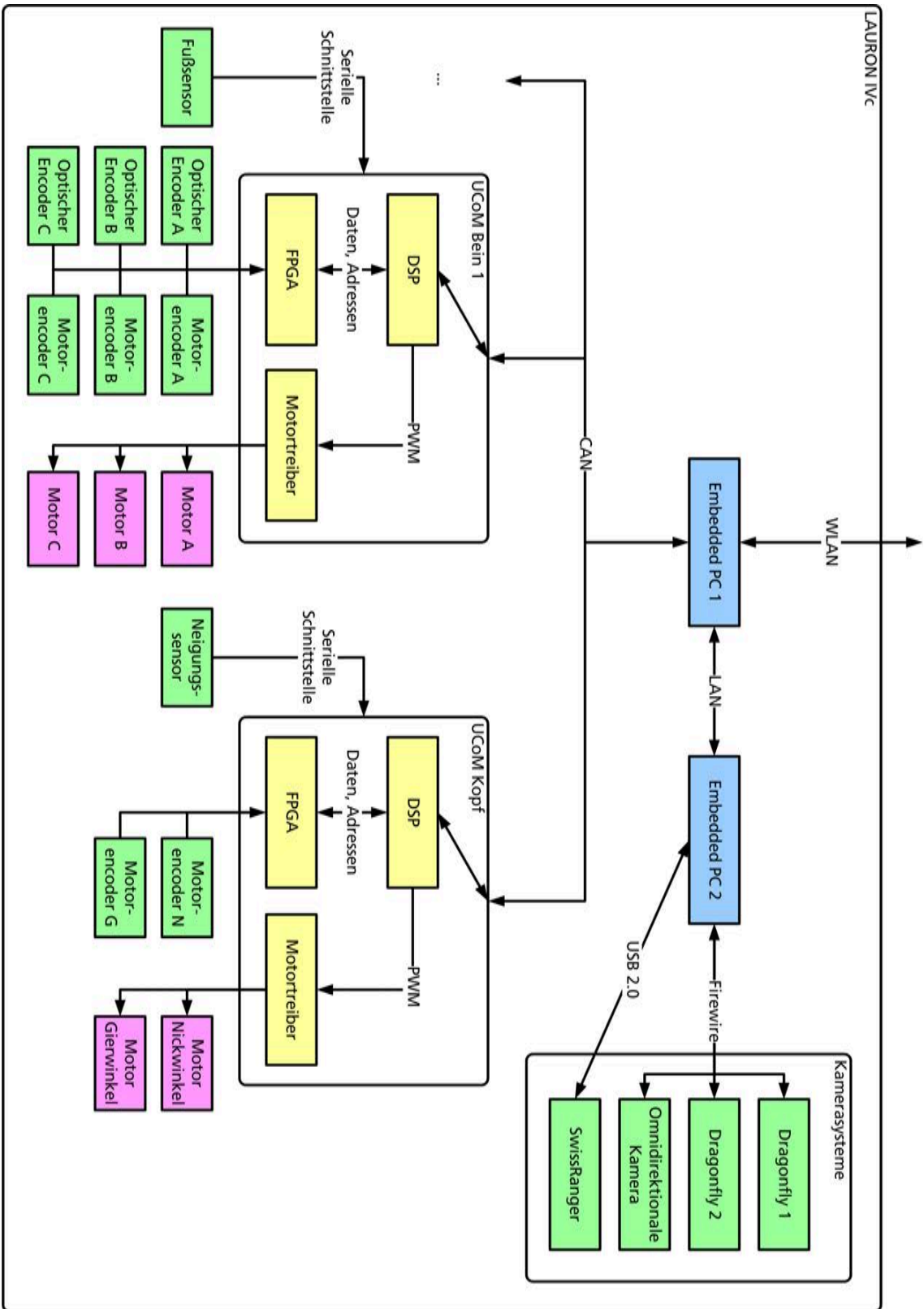


Abb. 7.3.: Die Hardwarearchitektur von LAURON IV.

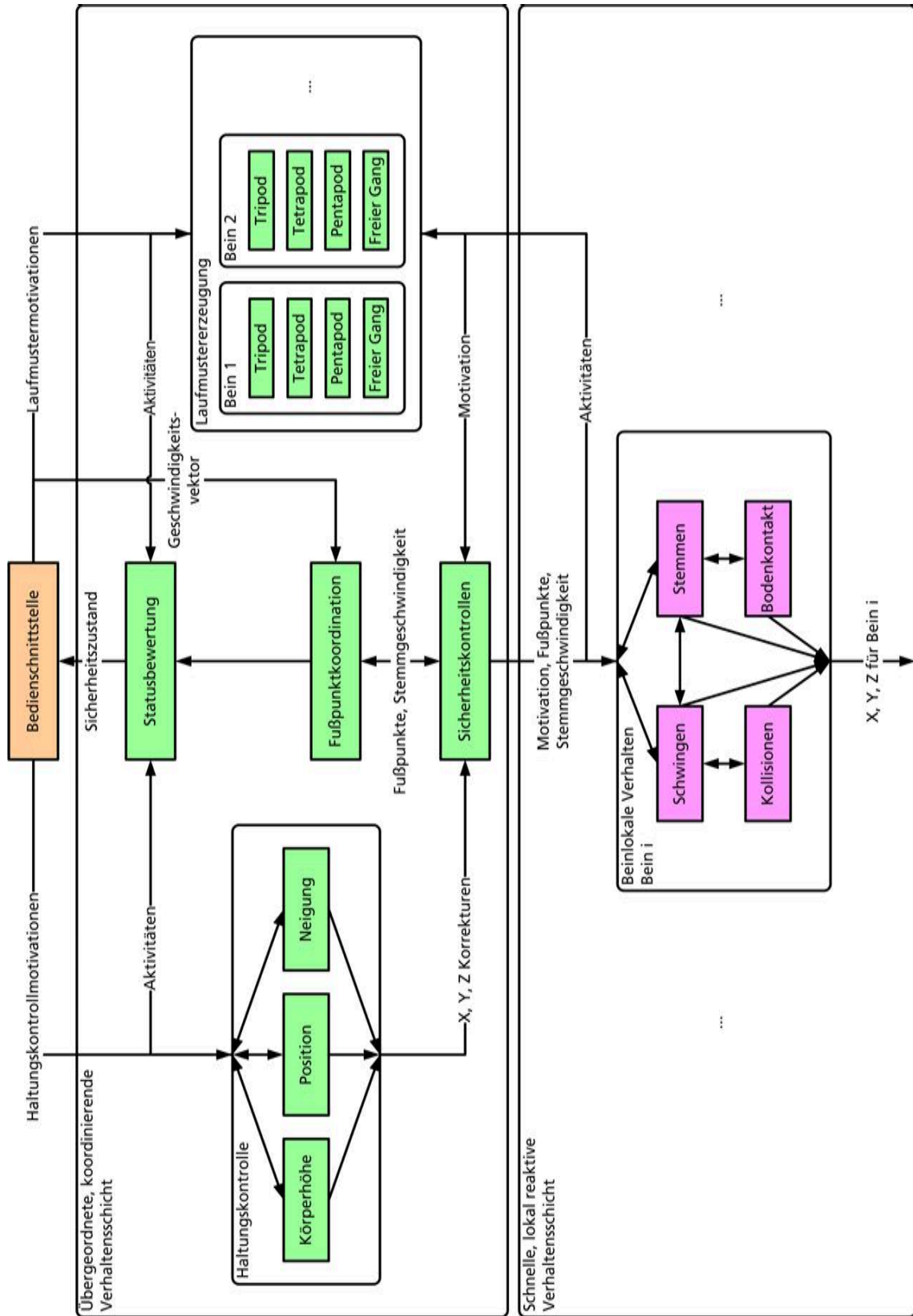


Abb. 7.4.: Architektur der verhaltensbasierten Steuerung der Laufmaschine LAURON IV.

Die grundlegenden Schwing-Stemm-Trajektorien der Beine werden von zwei beinlokalen Verhalten, dem Schwing- und dem Stemmverhalten, generiert. Jedes Bein verfügt über eigene beinlokale Verhalten. Zu diesen beinlokalen Verhalten gehören außerdem ein Kollisions- und ein Bodenkontaktverhalten. Zusammen sind diese vier Verhalten in der Lage, zuverlässig Trajektorien zum Laufen in komplexen und unstrukturierten Gebieten zu generieren. Koordiniert werden die einzelnen Gruppen der beinlokalen Verhalten durch die verschiedenen Laufmusterverhalten. Eigenständige Verhalten für den Tripod, Tetrapod, Pentapod und den freien Gang gewährleisten so stabile und sichere Laufmuster. Die Stabilität des Zentralkörpers wird von drei unabhängigen Haltungskontrollverhalten überwacht und kontrolliert. Das Höhenverhalten passt die Gesamtkörperhöhe dem Gelände an und stellt so sicher, dass immer ausreichend viel Bodenfreiheit vorhanden ist. Mit Hilfe des Neigungssensors kontrolliert das Neigungsverhalten die Orientierung des Zentralkörpers im Raum. Das Positionsverhalten verlagert durch die Verschiebung des Zentralkörpers den Schwerpunkt des Roboters so, dass LAURON IV stets stabil steht.

Ein übergeordnetes Statusverhalten nutzt verschiedene Verhaltensaktivitäten und ausgewählte Sensorinformationen, um einen Robotergesamtstatus zu erzeugen. Dieser Status wird wiederum verwendet, um verschiedene Schlüsselparameter (zum Beispiel die Schwinghöhe der Beine) automatisch anzupassen. LAURON IV ist so in der Lage, sich auch schwierigen Situationen autonom anzupassen. Weitere Details zur verhaltensbasierten Steuerung finden sich in [57].

### 7.1.3. Lokalisation und Umweltmodellierung

Die Lokalisation einer Laufmaschine ist eine sehr anspruchsvolle Aufgabe. Eine rein gelenkwinkelbasierte Odometrie ist aufgrund der hohen Anzahl an Freiheitsgraden und den stets vorhandenen mechanischen Ungenauigkeiten eine sehr fehleranfällige Lokalisationsmethode. Deshalb wird auf LAURON eine Kombination aus verschiedenen Lokalisationsmethoden eingesetzt. Zum einen werden die Informationen der Odometrie mit den Daten des GPS-Sensors und des Inertialen Navigationssystems fusioniert [35]. Zum anderen werden die Tiefendaten der Time-of-Flight-Kamera dazu verwendet, zusätzlich die Position des Roboters zu bestimmen [83]. Durch diese Kombination ist es möglich, die Laufmaschine LAURON ausreichend genau zu lokalisieren.

Das eingesetzte Umweltmodell basiert auf den Daten der Tiefenbildkamera. Diese zeichnet sich besonders durch ihre kleine und leichte Bauform, ihre hohen Datenraten und ihren verhältnismäßig niedrigen Stromverbrauch aus. Zudem liegen die erreichbaren Genauigkeiten im Bereich von einigen wenigen Millimetern. Allerdings erzeugen diese Kameras in gewissen Situationen fehlerhafte Daten, so dass vor einer weiteren Verwendung der Daten zusätzliche Vorverarbeitungsschritte notwendig sind.



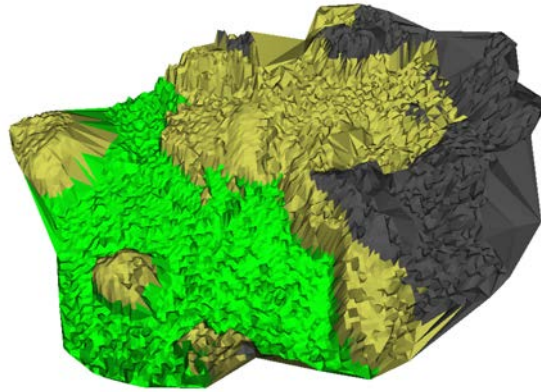


Abb. 7.5.: Nach der Filterung (zum Beispiel Intensitätsfilter) werden die Tiefendaten in das richtige Koordinatensystem transformiert. Anschließend können sie in das Umweltmodell übernommen werden. Bei diesem Modell handelt es sich um eine gridbasierte 2,5D-Höhenkarte, welche für jede Zelle neben der Höhe auch einen Zuverlässigkeitswert speichert.

Das Umweltmodell (siehe Abbildung 7.5) wurde so konzipiert, dass es in der Lage ist, die großen Datenmengen (380.000 Messpunkte/Sekunde) des SwissRangers in Echtzeit in das Modell einzutragen. Weitere Informationen zum entwickelten Umweltmodell werden in [83] beschrieben.

#### 7.1.4. Navigation

Die Navigation unterteilt sich in eine globale und eine lokale Planung. Die Aufgabe der globalen Navigationsplanung (siehe Abschnitt 7.2) besteht darin, anhand einer grob aufgelösten Umgebungskarte Pfade zum Erreichen von Zielpunkten und Regionen sowie zum systematischen Absuchen von Regionen zu planen. Für die Planung von Pfaden zum Erreichen von Zielpunkten und Regionen werden als probabilistisches Bahnplanungsverfahren so genannte *Rapidly-Exploring Random Trees* (RRTs) verwendet. Mit Hilfe der RRTs ist es möglich, Pfade zu mehreren Zielen gleichzeitig zu berechnen sowie zusätzliche Parameter, wie zum Beispiel den Energieverbrauch, bei der Planung zu berücksichtigen. Zum systematischen Absuchen von Regionen wird ein auf einem numerischen Potentialfeldverfahren basierender Ansatz verwendet. Die von der globalen Navigationsplanung erzeugten Pfade berücksichtigen keine dynamischen Hindernisse und dienen daher lediglich zur Generierung von Zwischenpunkten. Die Navigation zwischen diesen Zwischenpunkten wird von der lokalen Navigationsplanung übernommen, welche mit Hilfe eines lokalen, hoch aufgelösten Umweltmodells und einer Potentialfeldmethode [34] geeignete Pfade bestimmt.

Mit Hilfe der Kombination aus globaler und lokaler Navigationsplanung kann sich LAURON IV auch in unbekanntem oder nur teilweise bekannten Gebieten sicherer bewegen und dabei sowohl im Voraus bekannte als auch dynamische Hindernisse berücksichtigen.

## 7.2. Globale Navigation

In diesem Abschnitt wird die globale Navigationsplanung näher vorgestellt, da sie speziell für die Durchführung von autonomen Inspektionsmissionen mit der sechsbeinigen Laufmaschine LAURON IV entwickelt wurde (siehe [78]). Im Folgenden werden zunächst die Anforderungen an die globale Navigationsplanung beschrieben. Anschließend werden die entwickelten Pfadplanungsverfahren für das Erreichen von vorgegebenen Zielpunkten und Regionen sowie zum systematischen Absuchen von Regionen erläutert.

Die wesentliche Aufgabe der globalen Navigationsplanung besteht darin, dem Roboter zu ermöglichen, anhand einer globalen Karte auch über die Sensorreichweite hinaus navigieren (vergl. etwa [82]) und dabei bereits bekannte Hindernisse berücksichtigen zu können. Zu Beginn einer Inspektionsmission werden vom Benutzer zu inspizierende Objekte und Regionen in eine globale Karte eingetragen. Anschließend sind von der globalen Navigation sowohl zum Erreichen der Objekte und Regionen als auch zum systematischen Absuchen der jeweiligen Regionen entsprechende Pfade zu planen.

Für eine effiziente Durchführung der Inspektionsmissionen ist es darüber hinaus wichtig, eine geeignete Reihenfolge zu bestimmen, in der die angegebenen Objekte und Regionen inspiziert werden. Die Entscheidung für eine bestimmte Inspektionsroute hängt jedoch nicht alleine von den Entfernungen zu den zu inspizierenden Objekten und Regionen ab, sondern wird auch von den für die Inspektion vorgegebenen sonstigen Randbedingungen sowie von der Untersuchung der während der Inspektion gefundenen interessierenden Entitäten beeinflusst. Als Grundlage für die Entscheidungsfindung ist es daher wichtig, dass Pfade zu mehreren Objekten und Regionen gleichzeitig geplant und mit entsprechenden Pfadkosten versehen werden können.

### 7.2.1. Erreichung vorgegebener Zielpunkte und Regionen

Zur Planung von Pfaden zu vorgegebenen Zielpunkten und Regionen wird ein auf Rapidly-Exploring Random Trees (RRT) basierendes Verfahren verwendet. Das grundlegende Verfahren zur Erzeugung von RRTs [64] lässt sich durch die im Folgenden beschriebenen Modifikationen so anpassen, dass in einem Planungsdurchlauf gleichzeitig Pfade zu allen Zielobjekten und Zielregionen geplant werden können. Darüber hinaus verfügen RRTs über den Vorteil, dass roboterspezifische Parameter und Eigenschaften in die Planung mit einfließen können, da die Planung nicht im geometrischen Raum, sondern im Konfigurationsraum stattfindet. Dadurch ist es möglich, Eigenschaften wie zum Beispiel den Energieverbrauch, der zum Ablaufen eines Pfades benötigt wird, bereits bei der Planung zu berücksichtigen. Ein weiterer Vorteil von RRTs besteht darin, dass das Verfahren durch die zufällige Erzeugung von Konfigurationen sehr robust gegenüber lokalen Minima ist.

Im Folgenden wird der zielgerichtete Aufbau eines Rapidly-Exploring Random Trees be-

schrieben. Die Hauptfunktion *ErzeugeRRT* (siehe Algorithmus 7.1) erhält eine Startkonfiguration  $x_{initial}$  als Eingabeparameter und fügt diese als initialen Knoten in einen Graphen  $G$  ein. Der Graph  $G$  ist in Form einer Baumstruktur realisiert, die sich sukzessive aus explorierten Konfigurationen aufbaut. Dazu wird in jedem Schritt mit der *WähleKonfiguration*-Funktion (siehe Algorithmus 7.2) eine neue Konfiguration erzeugt. Dabei wird der RRT vorzugsweise in Richtung der Ziele ausgebreitet, indem mit einer Wahrscheinlichkeit  $p$  eines der Ziele als zu explorierende Konfiguration zurückgegeben wird. Andernfalls wird eine zufällige Konfiguration mit Hilfe der Funktion *ErzeugeZufälligeKonfiguration* erzeugt. Dieser probabilistische Anteil ermöglicht es dem RRT aus lokalen Minima zu entkommen. Die so erzeugte Konfiguration wird anschließend mit Hilfe der Funktion *ErweitereRRT* (siehe Algorithmus 7.3) in den Graphen eingefügt. Mit Hilfe der *PrüfeObZielErreicht*-Funktion wird in jeder Iteration geprüft, ob es sich bei der neu hinzugefügten Konfiguration um eine Zielkonfiguration handelt. Ist dies der Fall, so wird diese aus der Liste der Ziele entfernt. Das Einfügen neuer Konfigurationen wird so lange wiederholt, bis alle Ziele erreicht sind oder eine definierte Anzahl von Iterationen erreicht wurde.

---

**Algorithmus 7.1** ErzeugeRRT
 

---

**Eingabe:**  $x_{initial}, ziele$

**Ausgabe:**  $G$

```

G.Initialisiere( $x_{initial}$ )
 $i \leftarrow 1$ 
while  $i \leq (K \cdot (\#ziele + 1) / 2) \wedge \#ziele \neq 0$  do
   $x_{zufall} \leftarrow$  WähleKonfiguration( $ziele$ )
   $x_{neu} \leftarrow$  ErweitereRRT( $G, x_{zufall}$ )
  if  $x_{neu} \neq NULL$  then
    PrüfeObZielErreicht( $x_{neu}, ziele$ )
  end if
   $i \leftarrow i + 1$ 
end while
return  $G$ 

```

---



---

**Algorithmus 7.2** WähleKonfiguration
 

---

**Eingabe:**  $ziele$

**Ausgabe:**  $x_{zufall}$

```

 $z \leftarrow$  Zufallszahl(0, 1)
if  $z < p$  then
   $index \leftarrow$  Zufallszahl(0,  $\#ziele - 1$ )
  return  $ziele[index]$ 
else
   $x_{zufall} \leftarrow$  ErzeugeZufälligeKonfiguration()
  return  $x_{zufall}$ 
end if

```

---

**Algorithmus 7.3** *ErweitereRRT*

---

**Eingabe:**  $G, x$ **Ausgabe:**  $x_{neu}$  $x_{nachbar} \leftarrow \text{NächsterNachbar}(x, G)$ **if**  $\text{NeuerZustand}(x, x_{nachbar}, x_{neu}, u_{neu})$  **then** $G.\text{FügeKnotenHinzu}(x_{neu})$  $G.\text{FügeKanteHinzu}(x_{nachbar}, x_{neu}, u_{neu})$ **return**  $x_{neu}$ **end if****return**  $NULL$ 

---

Die Funktion *ErweitereRRT* erhält als Eingangsgrößen den Graphen  $G$  und die ausgewählte neue Konfiguration  $x$ . Im ersten Schritt wird mit Hilfe der Funktion *NächsterNachbar* die Konfiguration  $x_{nachbar}$  im Graphen  $G$  gesucht, die am nächsten zu  $x$  liegt. Diese Bestimmung erfolgt effizient mit Hilfe von KD-Trees [107]. Anschließend wird mit Hilfe der Funktion *NeuerZustand* ausgehend von  $x_{nachbar}$  eine Bewegung in Richtung von  $x$  erzeugt. Befindet sich  $x$  zu weit von  $x_{nachbar}$  entfernt, wird die Ausbreitung in Richtung von  $x$  nach Erreichen eines Schwellenwerts  $\varepsilon$  angehalten.  $x_{neu}$  entspricht dann der Konfiguration, die von  $x_{nachbar}$  um  $\varepsilon$  in Richtung von  $x$  erweitert wurde. Zur Erzeugung der neuen Konfiguration wird eine Übergangsfunktion  $x_{neu} = f(x_{nachbar}, u)$  verwendet, wobei der Vektor  $u$  aus einer Menge  $U$  von Eingabegrößen gewählt wird. Mit Hilfe dieser Funktion ist es möglich, die Übergänge zwischen Konfigurationen durch spezielle Robotereigenschaften  $u$  einzuschränken. Mit der Angabe von  $u$  lassen sich zum Beispiel Abhängigkeiten zwischen Lenkwinkel und Position eines nichtholonomen Roboters beschreiben. Handelt es sich wie bei LAURON um einen holonomen Roboter, so können hingegen für alle Zustände die identischen Eigenschaften  $u$  angenommen werden. Als Ergebnis liefert die Funktion *NeuerZustand* eine neue Konfiguration  $x_{neu}$  und einen neuen Vektor  $u_{neu}$  zurück. Während der Exploration überprüft die Funktion *NeuerZustand* die Konfigurationen außerdem auf mögliche Kollisionen. Wenn aufgrund von Hindernissen keine Konfiguration in Richtung von  $x$  gefunden werden kann, gibt die Funktion *NeuerZustand* den Wert *falsch* zurück und es wird keine Konfiguration eingefügt. Gibt es hingegen keine Kollision, werden in der Funktion *ErweitereRRT* der Knoten  $x_{neu}$  sowie eine Kante von  $x_{nachbar}$  nach  $x_{neu}$  zum Graphen  $G$  hinzugefügt. Beim Einfügen der Kante werden die Robotereigenschaften  $u_{neu}$  ebenfalls einbezogen. Auf diese Weise kann sichergestellt werden, dass der resultierende Pfad auch tatsächlich vom Roboter ausgeführt werden kann.

Die Angabe von Zielregionen erfolgt mit Hilfe von Polygonen. Für die Pfadplanung zu Regionen wird jeweils der dem Roboter am nächsten gelegene Eckpunkt der entsprechenden Polygone in die Liste der Ziele eingefügt. Eine Region gilt als erreicht, sobald sich der Roboter innerhalb des Polygons befindet. Dies wird entsprechend in der Funktion *PrüfeObZielErreicht* überprüft. Befindet sich die zu überprüfende Konfiguration innerhalb des Polygons einer Regi-

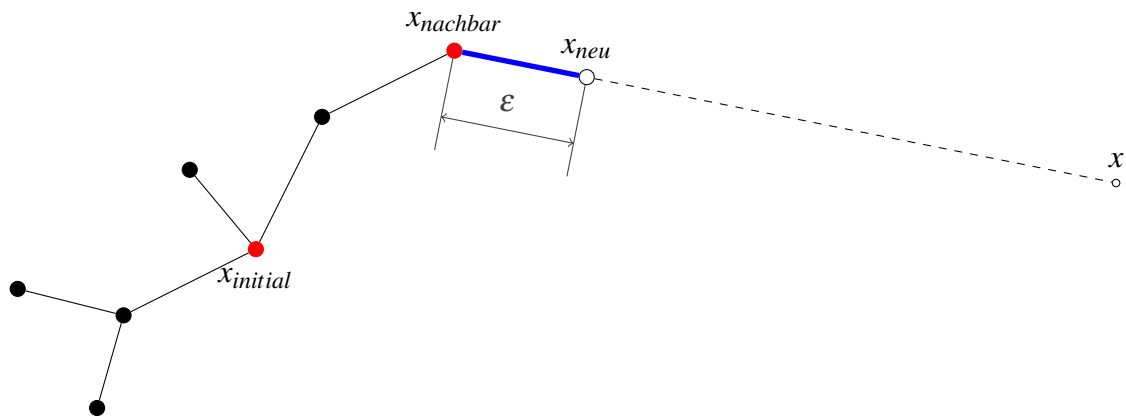


Abb. 7.6.: Ausführung eines Explorationsschrittes mit der Funktion *NeuerZustand* zum iterativen Aufbau eines RRTs [64].

on, wird der entsprechende Eckpunkt der Region aus der Liste der Ziele entfernt.

Nachdem der RRT vollständig aufgebaut ist, erfolgt die Extraktion der Pfade durch rekursives Ablaufen des Baumes von den Zielknoten zum Startknoten, welcher dem Wurzelknoten des Baumes entspricht. Hierzu enthält jeder Knoten des Baumes, mit Ausnahme des Wurzelknotens, einen Zeiger auf den jeweiligen Vorgängerknoten.

### Pfadglättung

Das probabilistische Verfahren zum Aufbau der RRTs erzeugt oft zackenförmige Pfade (siehe Abbildung 7.7). Darüber hinaus sind die Pfade teilweise weiter von den Hindernissen entfernt als notwendig. Aus diesem Grund wird ein zusätzliches Verfahren zur Glättung und Optimierung der Pfade eingesetzt, welches auf der *Minimal Deviation Methode* [105] beruht. Die Grundidee dieser Methode besteht darin, Konfigurationen aus dem Pfad zu entfernen, falls sich die seitliche Abweichung des durch die Entfernung der Konfiguration geglätteten Pfades vom ursprünglichen Pfad innerhalb einer Schranke  $d_s$  befindet. Auf diese Weise werden nur Konfigurationen gelöscht, durch die sich der geglättete Pfad nicht zu sehr vom ursprünglichen Pfad entfernt. Der geglättete Pfad wird somit innerhalb eines Bereiches von  $d_s$  um den ursprünglichen Pfad verschoben. Die Pfadglättung erfolgt entsprechend dem folgenden Algorithmus (siehe [105]):

1. Für alle noch nicht als gelöscht markierten Konfigurationen  $x_k$  wird die seitliche Abweichung  $x_k.d$  zwischen dem geglätteten Pfad  $P'$  und dem ursprünglichen Pfad  $P$  unter der Annahme berechnet, dass  $x_k$  entfernt wird.
2. Auswahl der Konfiguration  $x_k$  mit der kleinsten seitlichen Abweichung  $x_k.d$ :
  - Ist die Abweichung  $x_k.d$  kleiner als die maximale Abweichung  $d_s$ , wird  $x_k$  als gelöscht markiert und mit Schritt 1 fortgefahren.

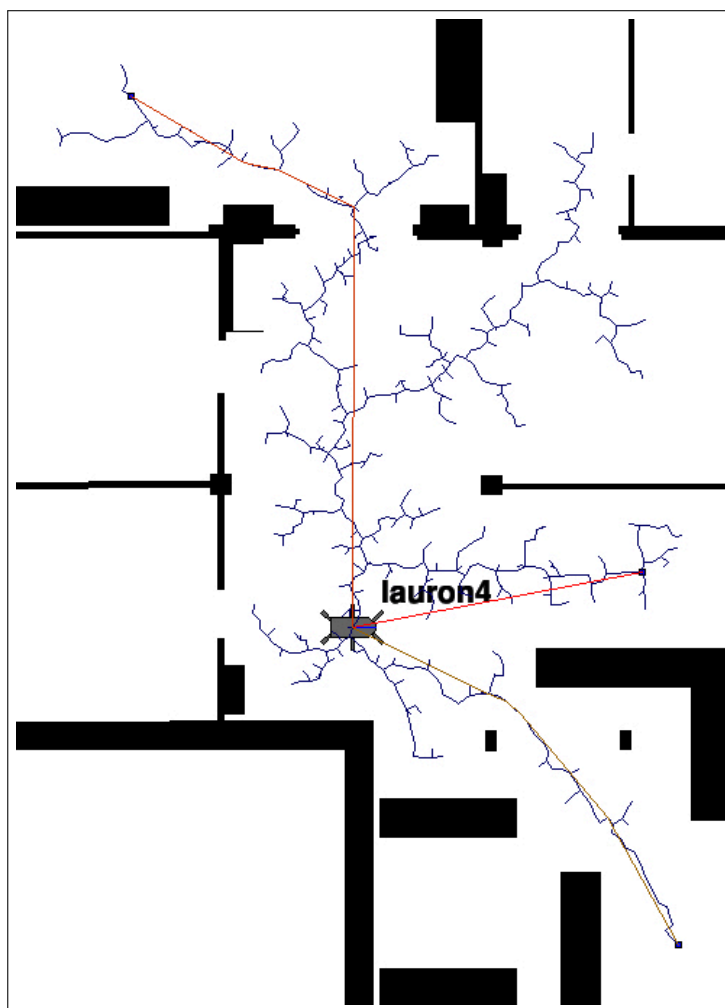


Abb. 7.7.: Zielgerichteter RRT zur Planung von Pfaden zu drei verschiedenen Zielobjekten.

- Andernfalls können keine weiteren Konfigurationen entfernt werden und es wird der Pfad  $P' = \{x_i | x_i \text{ ist nicht als gelöscht markiert}\}$  zurückgegeben.

Die seitliche Abweichung einer Konfiguration  $x_i.d$  kann mit Hilfe der drei Fehlerfunktionen  $K_1$ ,  $K_2$  oder  $K_3$  (siehe Abbildung 7.8) berechnet werden:

**Fehlerfunktion  $K_1$**   $x_i.d$  ist die maximale euklidische Abweichung des geglätteten Pfadsegments  $P_{i,t} = \{x_{min}, x_{max}\}$  vom ursprünglichen Pfadsegment  $P_{i,o} = \{x_{min}, \dots, x_i, \dots, x_{max}\}$ . Die Konfigurationen  $x_{min}$  und  $x_{max}$  entsprechen dabei der linken und rechten Nachbarkonfiguration von  $x_i$ , die noch nicht als gelöscht markiert sind. Das geglättete Pfadsegment  $P_{i,t}$  enthält nur den nächsten linken und rechten Nachbar  $x_{min}$  und  $x_{max}$ . Das ursprüngliche Pfadsegment  $P_{i,o}$  hingegen enthält die Konfigurationen  $x_{min}$  und  $x_{max}$  sowie alle dazwischen liegenden Konfigurationen.

**Fehlerfunktion  $K_2$**   $x_i.d$  ist das quadratische Mittel der Abweichungen aller Konfigurationen des ursprünglichen Pfads  $P_{i,o}$  vom geglätteten Pfad  $P_{i,t}$  zwischen den Konfigurationen  $x_{min}$

und  $x_{max}$ :

$$x_i.d = \sqrt{\frac{1}{n} \sum_{i=1}^n t_i^2}, \quad [7.1]$$

wobei mit  $t_i$  jeweils die kürzesten Entfernungen zwischen den Konfigurationen  $x_i \in P_{i,o}$  und dem Pfadsegment  $P_{i,t}$  bezeichnet werden.

**Fehlerfunktion  $K_3$**   $x_i.d$  ist die Fläche zwischen dem geglätteten Pfadsegment  $P_{i,t}$  und dem entsprechenden Segment  $P_{i,o}$  des ursprünglichen Pfades.

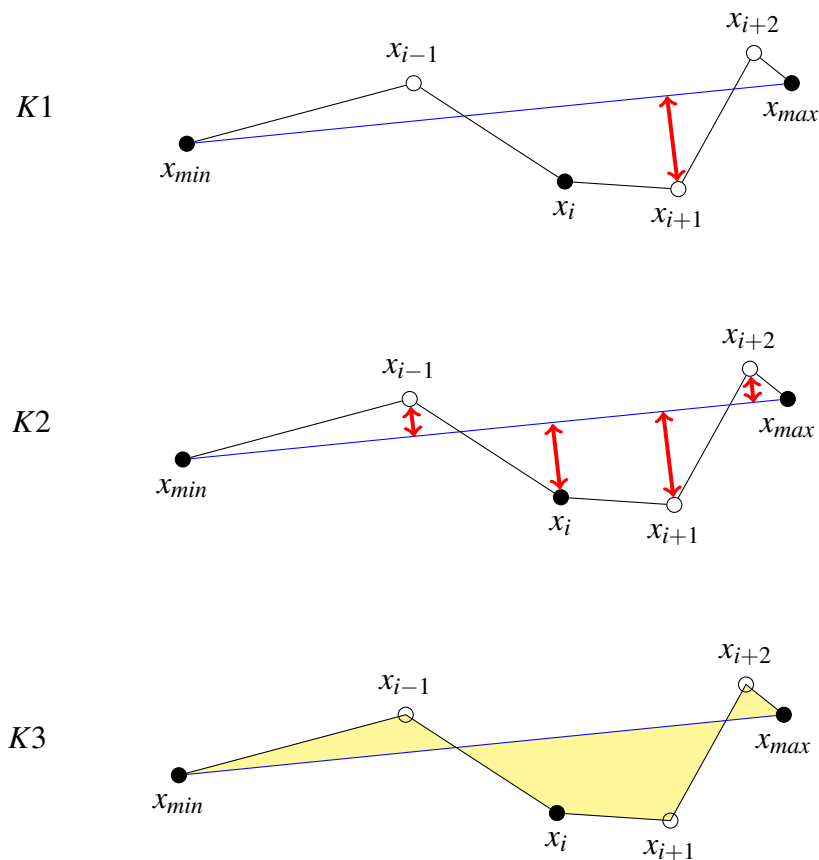


Abb. 7.8.: Fehlerfunktionen der *Minimal Deviation* Methode zur Pfadglättung [105].

Das in [105] vorgestellte *Minimal Deviation* Verfahren berücksichtigt bei der Glättung von Pfaden keine Hindernisse. Dies kann bei großen  $d_s$  dazu führen, dass ein Pfadsegment zu nahe an einem Hindernis verläuft oder sogar das Hindernis schneidet. Deshalb muss vor dem Löschen einer Konfiguration für das jeweils entstehende Pfadsegment geprüft werden, ob es kollisionsfrei ist. Hierzu wird eine virtuelle Box um den Roboter gelegt, welche seiner maximalen Ausdehnung plus einer Sicherheitsmarge entspricht. Diese Box wird entlang des Pfadsegments verschoben und im globalen Umweltmodell auf Kollisionen überprüft. Tritt eine Kollision auf, kann die untersuchte Konfiguration nicht gelöscht werden und wird entsprechend markiert.

### 7.2.2. Systematisches Absuchen von Regionen

Neben dem Erreichen von Zielobjekten und Regionen ist das systematische Absuchen von Regionen ein wesentlicher Bestandteil der globalen Navigationsplanung. Hierfür wird das so genannte *Complete Coverage* Verfahren [108] verwendet. Die Funktionsweise des *Complete Coverage* Verfahrens basiert auf dem Einsatz eines numerischen Potentialfelds. Hierfür wird eine in Zellen eingeteilte Karte der Umwelt benötigt. In diesem Fall wird ein Belegtheitsgitter mit Kosten für jede Zelle verwendet, das aus der globalen Karte erzeugt wird. Welche Kosten in den Zellen gespeichert werden, hängt von der verwendeten Kostenfunktion ab. Es stehen zwei unterschiedliche Kostenfunktionen zur Verfügung. Die *Distance Transform* Kostenfunktion ist ein sehr einfaches Prinzip zum Errechnen der Zellkosten. Darauf aufbauend können die Zellkosten auch mit der komplexeren *Path Transform* Kostenfunktion bestimmt werden. Beide Verfahren werden im Folgenden kurz erläutert.

#### **Distance Transform Kostenfunktion**

Die wesentliche Idee besteht darin, eine Distanz-Wellenfront von der Zielzelle ausgehend durch alle freien Zellen der Umgebung zu propagieren. Die Berechnung der *Distance Transform* Kosten  $DT(c)$  für jede Zelle  $c$  erfolgt rekursiv:

$$DT(c) = \begin{cases} 0 & \text{falls } c = \text{Zielzelle} \\ \min_{x \in NZ(c)} (DT(x)) + d & \text{sonst} \end{cases} \quad [7.2]$$

Die Kosten  $DT(c)$  für die aktuelle Zelle  $c$  werden aus den Kosten der Nachbarzelle  $x$  mit den geringsten Kosten  $DT(x)$  und den Kosten  $d$  berechnet. Dazu wird das Minimum über die Menge  $NZ(c)$  aller direkten Nachbarzellen gebildet. Die Kosten  $d$  entsprechen den Kosten, um von einer Zelle zur nächsten zu gelangen, und werden üblicherweise auf 1 gesetzt. Die *Distance Transform* Kosten haben den Nachteil, dass der resultierende Pfad meist relativ kurze gerade Abschnitte und sehr viele Kurven enthält. Durch das häufige Drehen des Roboters verlangsamt sich die Ausführung des Absuchens der Region [108].

#### **Path Transform Kostenfunktion**

Das Ziel der *Path Transform* Kostenfunktion besteht darin, Pfade mit möglichst langen Geraden und wenigen Kurven zu erzeugen. Hierzu werden die Abstände der Zellen zu Hindernissen berücksichtigt. Neben den *Distance Transform* Kosten werden so genannte *Obstacle Transform* Kosten  $OT(c)$  für jede Zelle  $c$  berechnet. Diese Kosten geben an, wie weit sich eine Zelle vom nächsten Hindernis entfernt befindet, wobei die Entfernung als Anzahl von Zellen gemessen wird. Zur Berechnung der *Obstacle Transform* Kosten werden die Kosten aller Zellen, die durch



Hindernisse belegt sind, auf 0 gesetzt. Ausgehend von den Hinderniszellen werden die Kosten für alle Zellen rekursiv aus den Kosten der Nachbarzellen berechnet. Dazu werden die Kosten der Nachbarzelle mit den geringsten Kosten um eins erhöht und der aktuellen Zelle zugewiesen. Die eigentlichen *Path Transform* Kosten  $PT(c)$  werden wie folgt berechnet:

$$PT(c) = \min_{p \in P} \left( length(p) + \sum_{c_i \in p} (\alpha \cdot obstacle(c_i)) \right), \quad [7.3]$$

wobei  $P$  die Menge aller möglichen Pfade von einer Zelle  $c$  zum Ziel bezeichnet. Die Funktion  $length(p)$  gibt die Länge eines Pfades  $p \in P$  zum Ziel an und die Funktion  $obstacle(c)$  beschreibt die Hinderniskosten in Abhängigkeit davon, wie weit sich die Zelle  $c$  vom nächsten Hindernis entfernt befindet. Sie wird mit Hilfe der *Obstacle Transform* Kosten  $OT(c)$  bestimmt, wobei große Entfernungen und somit große  $OT(c)$  auf kleine  $obstacle(c)$  abgebildet werden und umgekehrt. Somit erhalten Zellen direkt neben Hindernissen die größten Kosten. Mit Hilfe des Gewichtungsfaktors  $\alpha \geq 0$  wird der Einfluss der Hinderniskosten  $obstacle(c)$  auf die Pfadkosten festgelegt.

### Das *Complete Coverage* Verfahren

Nach der Bestimmung der Kosten für die einzelnen Zellen erfolgt die Planung eines Pfades zum vollständigen Absuchen von Regionen mit Hilfe des *Complete Coverage* Verfahrens (siehe Algorithmus 7.4). Ausgehend von der Startzelle wird dabei rekursiv nach unbesuchten Nachbarzellen mit den jeweils höchsten Kosten gesucht. Die entsprechende Nachbarzelle wird als besucht markiert und als neue Ausgangszelle für die weitere Suche verwendet. Wird kein unbesuchter Nachbar mehr gefunden, obwohl noch unbesuchte Zellen existieren, so wird der bisher erzeugte Pfad rückwärts nach noch nicht besuchten Zellen abgesucht. Auf diese Weise wird sichergestellt, dass der Algorithmus immer terminiert, sobald alle Zellen, die der Roboter erreichen kann, abgearbeitet sind. Der Algorithmus kann sowohl mit den *Distance Transform* Kosten  $DT(c)$  als auch mit den *Path Transform* Kosten  $PT(c)$  ausgeführt werden. Bei der Verwendung der *Path Transform* Kosten enthalten die generierten Pfade jedoch längere gerade Strecken und weniger Kurven als bei den *Distance Transform* Kosten [108]. Ein Beispiel für einen mit Hilfe der *Path Transform* Kostenfunktion generierten Pfad zeigt Abbildung 7.9.

## 7.3. Detektion und Klassifikation von Abfall

In diesem Abschnitt wird näher auf die Fähigkeiten zur bildbasierten Detektion und Klassifikation von Abfall eingegangen, da diese speziell für die Durchführung von autonomen Inspektionsmissionen mit LAURON IV entwickelt wurden (siehe [11]) und eine wesentliche Grundlage für das gewählte Inspektionsszenario bilden. Im Folgenden werden zunächst die Anforderungen

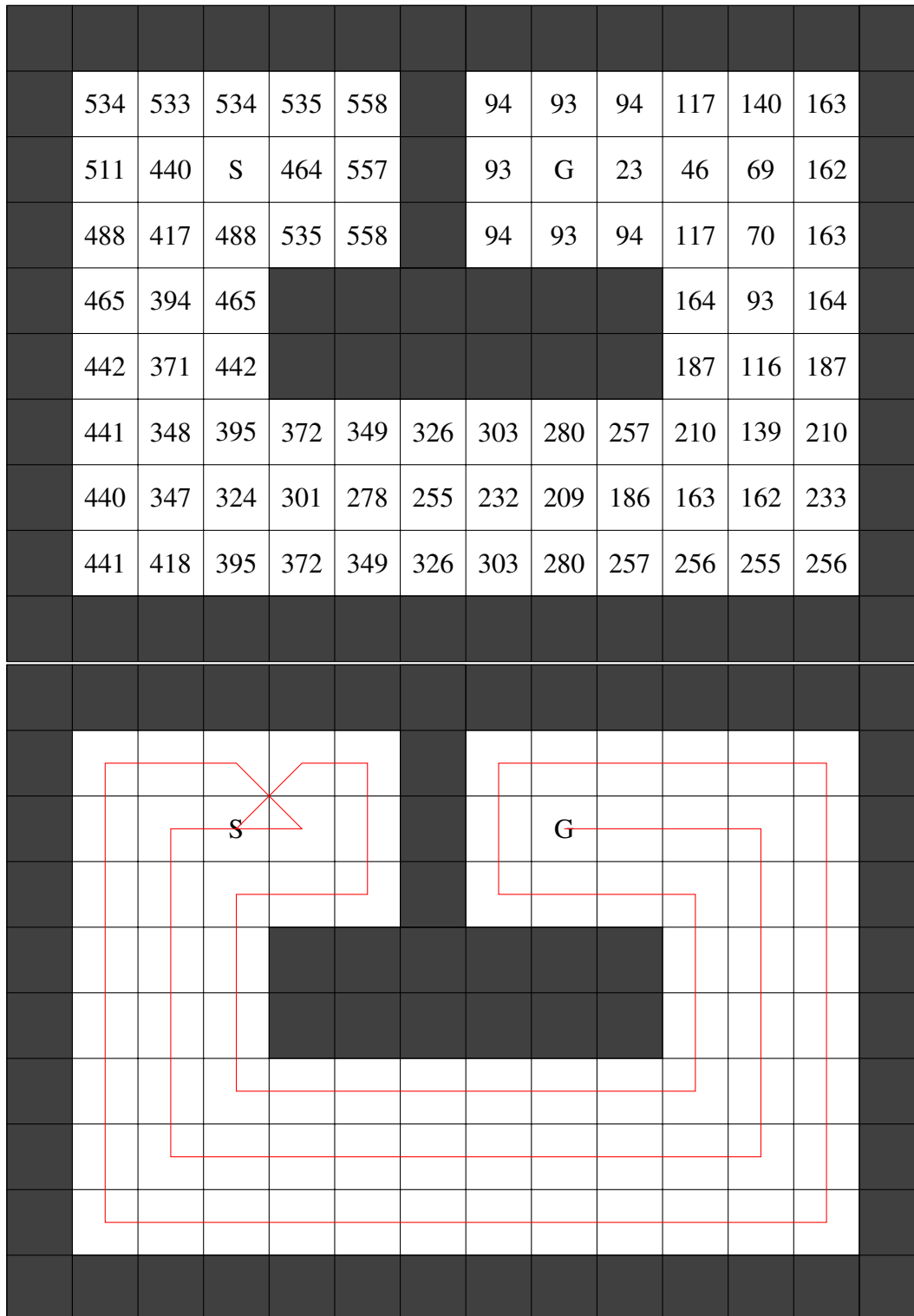


Abb. 7.9.: Beispiel zur Pfadgenerierung mit *Path Transform Kosten* [108].

**Algorithmus 7.4** Vollständiges Absuchen von Regionen [108]

---

```

Eingabe: start_zelle
           aktuelle_zelle = start_zelle
while  $PT(aktuelle\_zelle) > 0$  do
    Markiere aktuelle_zelle als besucht
    Finde unbesuchte nachbar_zelle mit höchsten PT-Kosten
    if keine nachbar_zelle gefunden then
      Setze aktuelle_zelle auf zuvor_besuchte_zelle
    else
      Setze aktuelle_zelle auf nachbar_zelle
    end if
end while

```

---

an die bildbasierte Detektion und Klassifikation von Abfall beschrieben, bevor anschließend die einzelnen Verfahren genauer vorgestellt und erläutert werden.

Die wesentliche Aufgabe der Fähigkeiten zur Detektion und Klassifikation von Abfall besteht darin, dem Roboter zu ermöglichen mit Hilfe seiner Kameras (siehe Abschnitt 7.1) Abfallobjekte in seiner direkten Umgebung zu erkennen und die Art der gefundenen Abfallobjekte zu bestimmen. Hierzu wird zunächst ein Verfahren zur Lokalisierung und Segmentierung interessierender Regionen in den Bilddaten benötigt. Die so gewonnenen Bildbereiche sind anschließend mit Hilfe eines geeigneten Klassifikationsansatzes auf das tatsächliche Vorliegen von Abfall zu prüfen. Darüber hinaus werden Verfahren zur Klassifikation sowie zur Absicherung der Abfallart benötigt.

### 7.3.1. Detektion auffälliger Bildbereiche

Die Detektion und Segmentierung interessierender Bildregionen wird mit Hilfe eines aufmerksamkeitsbasierten Verfahrens (siehe [56], [54] und [55]) durchgeführt, dessen Funktionsweise an die Verarbeitung optischer Reize bei Säugetieren und beim Menschen angelehnt ist. Hierzu wird das Eingangsbild zunächst in eine Menge von topographischen Merkmalskarten (engl. *feature map*) zerlegt. Innerhalb der einzelnen Merkmalskarten konkurrieren verschiedene Regionen um die Aufmerksamkeit, so dass nur solche Regionen erhalten bleiben, die sich lokal deutlich von ihrer Umgebung abheben. Alle Merkmalskarten werden dann zu einer einzigen Aufmerksamkeitskarte (engl. *saliency map*) fusioniert, welche lokal hervorstechende Regionen in der gesamten Szene topographisch kodiert. Zur Berechnung der Aufmerksamkeitskarten wird die Bibliothek iNVT<sup>14</sup> eingesetzt.

---

<sup>14</sup>iLab Neuromorphic Vision C++ Toolkit: <http://ilab.usc.edu/toolkit/>

## Extraktion visueller Merkmale

Zunächst werden charakteristische Merkmale (Farbe, Intensität und Orientierung) der untersten Ebene der visuellen Wahrnehmung mit Hilfe von linearen Filterverfahren auf verschiedenen Skalen aus dem ursprünglichen Farbbild extrahiert. Die verschiedenen Skalen werden dabei mit Hilfe von Gauß-Pyramiden erzeugt, die aus stufenweisen Tiefpassfilterungen und Unterabtastungen bestehen. Es wird eine Tiefe von 9 Skalen verwendet, wobei die horizontalen und vertikalen Bildreduktionsfaktoren von 1:1 (Ebene 0, das ursprüngliche Bild) bis zu 1:256 (Ebene 8) in aufeinanderfolgenden Zweierpotenzen reichen. In Anlehnung an die visuellen rezeptiven Felder werden die einzelnen Merkmale in einer so genannten *center-surround*-Struktur berechnet. Die Verwendung dieses biologischen Paradigmas führt dazu, dass das System sensitiv auf lokale Kontraste in den einzelnen Merkmalen reagiert und nicht auf die Amplituden in den zugehörigen Merkmalskarten. Die *center-surround*-Operationen werden im verwendeten Modell für ein gegebenes Merkmal durch die Differenzen zwischen einer fein- und einer grob aufgelösten Skala realisiert: Das Zentrum des rezeptiven Felds entspricht einem Pixel auf der Ebene  $c \in \{2, 3, 4\}$  in der Pyramide und die Umgebung dem entsprechenden Pixel auf der Ebene  $s = c + \delta$  mit  $\delta \in \{3, 4\}$ . Es werden somit für jedes Merkmal sechs Merkmalskarten berechnet. Die skalenübergreifende Differenz zwischen zwei Karten, im Folgenden mit  $\ominus$  bezeichnet, wird durch Interpolation auf die feiner aufgelöste Skala und durch punktweise Subtraktion gewonnen.

Werden die Rot-, Grün- und Blaukanäle des Eingangsbildes mit  $r$ ,  $g$  bzw.  $b$  bezeichnet, so ergibt sich das Intensitätsbild wie folgt:

$$I = (r + g + b)/3 \quad [7.4]$$

Das Intensitätsbild  $I$  wird verwendet, um eine Gauß-Pyramide  $I(\sigma)$  zu erzeugen, wobei  $\sigma \in [0..8]$  die Skala bezeichnet. Die  $r$ ,  $g$  und  $b$  Kanäle werden mit  $I$  normalisiert, um den Farbton von der Intensität zu entkoppeln. Da Farbtonvariationen jedoch bei einer sehr geringen Helligkeit nicht mehr wahrgenommen werden können (und somit auch nicht hervorstehten können), wird die Normalisierung nur auf Bildregionen angewendet, in denen die Intensität  $I$  größer als 1/10 ihres Maximums bezogen auf das gesamte Bild ist. Neben dem Intensitätsbild werden die vier Farbkanäle Rot, Grün, Blau und Gelb erzeugt:

$$R = r - (g + b)/2 \quad [7.5]$$

$$G = g - (r + b)/2 \quad [7.6]$$

$$B = b - (r + g)/2 \quad [7.7]$$

$$Y = (r + g)/2 - |r - g|/2 - b, \quad [7.8]$$

wobei negative Werte auf Null gesetzt werden. Aus diesen vier Farbkanälen werden entsprechende Gauß-Pyramiden  $R(\sigma)$ ,  $G(\sigma)$ ,  $B(\sigma)$  und  $Y(\sigma)$  erzeugt. Darauf aufbauend werden die Merkmalskarten anhand der *center-surround*-Differenzen (siehe oben) zwischen der feinaufgelösten Zentralskala  $c$  und der grobaufgelösten Umgebungsskala  $s$  berechnet. Für den Intensitätskontrast ergibt sich eine Menge von sechs Karten:

$$\mathcal{I}(c, s) = |I(c) \ominus I(s)| \quad [7.9]$$

Eine zweite Menge von Merkmalskarten wird auf ähnliche Art und Weise für die Farbkanäle erzeugt. Dabei werden in Anlehnung an die Gegenfarbentheorie die Merkmalskarten  $\mathcal{RG}(c, s)$  und  $\mathcal{BY}(c, s)$  berechnet, welche jeweils die Farbkontraste Rot/Grün und Grün/Rot sowie Blau/Gelb und Gelb/Blau repräsentieren:

$$\mathcal{RG}(c, s) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \quad [7.10]$$

$$\mathcal{BY}(c, s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \quad [7.11]$$

Informationen über die lokale Orientierung werden aus dem Intensitätsbild  $I$  mit Hilfe von orientierten Gabor-Pyramiden  $O(\sigma, \theta)$  gewonnen, wobei  $\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$  die bevorzugte Orientierung angibt [42]. Die Merkmalskarten  $\mathcal{O}(c, s, \theta)$  für die Orientierung kodieren als Gruppe den lokalen Orientierungskontrast zwischen der Zentral- und der Umgebungsskala:

$$\mathcal{O}(c, s, \theta) = |O(c, \theta) \ominus O(s, \theta)| \quad [7.12]$$

### Erzeugung der Aufmerksamkeitskarte

Jede der Merkmalskarten wird zunächst auf einen festen Wertebereich (zwischen 0 und 1) normiert, um merkmalsabhängige Amplitudendifferenzen aufgrund der unterschiedlichen Verfahren zur Merkmalsextraktion zu vermeiden. Anschließend wird jede Merkmalskarte iterativ mit einem großen 2D *Difference-of-Gaussians*-Filter gefaltet:

$$\mathcal{DoG}(x, y) = \frac{c_{ex}^2}{2\pi\sigma_{ex}^2} e^{-(x^2+y^2)/(2\sigma_{ex}^2)} - \frac{c_{inh}^2}{2\pi\sigma_{inh}^2} e^{-(x^2+y^2)/(2\sigma_{inh}^2)} \quad [7.13]$$

Auf das Ergebnis wird jeweils das Originalbild addiert und negative Werte werden nach jeder Iteration auf Null gesetzt. Der  $\mathcal{DoG}$ -Filter führt an jedem Ort zu einer starken Anregung (Excitation), der von einer breiten Hemmung (Inhibition) der benachbarten Orte entgegen gewirkt wird. Konkret werden  $\sigma_{ex} = 2\%$  und  $\sigma_{inh} = 25\%$  der Breite der Eingangsbildes sowie  $c_{ex} = 0,5$  und  $c_{inh} = 1,5$  verwendet. In jeder Iteration des Normalisierungsprozesses ist eine gegebene

Merkmalskarte  $\mathcal{M}$  somit Gegenstand der folgenden Transformation:

$$\mathcal{M} \leftarrow |\mathcal{M} + \mathcal{M} * \mathcal{DoG} - C_{inh}|_{\geq 0}, \quad [7.14]$$

wobei  $\mathcal{DoG}$  den beschriebenen *Difference-of-Gaussians*-Filter bezeichnet,  $||_{\geq 0}$  negative Werte verwirft und  $C_{inh}$  ein konstanter hemmender Term ist.  $C_{inh}$  führt dazu, dass Regionen, in denen Anregung und Hemmung fast gleich stark sind, leicht unterdrückt werden. Solche Regionen entsprechen üblicherweise gleichförmigen Texturen, die nicht als auffällig angesehen werden. Für jede Merkmalskarte werden zehn Iterationen des in Gleichung 7.14 beschriebenen Prozesses durchgeführt. Auf diese Weise werden Merkmalskarten mit anfänglich vielen Peaks ähnlicher Amplitude unterdrückt, wohingegen Merkmalskarten mit wenigen ausgeprägten Peaks verstärkt werden. Das Ergebnis dieser Normalisierung wird im Folgenden mit  $\mathcal{N}(\mathcal{M})$  bezeichnet.

Nach der Normalisierung werden die Merkmalskarten für Intensität, Farbe und Orientierung skalenübergreifend zu drei separaten Auffälligkeitskarten (engl. *conspicuity maps*) addiert: einer für Intensität  $\bar{\mathcal{I}}$ , einer für Farbe  $\bar{\mathcal{C}}$  und einer für Orientierung  $\bar{\mathcal{O}}$ . Die skalenübergreifende Addition zweier Karten, im Folgenden mit  $\oplus$  bezeichnet, wird durch Reduktion jeder Karte auf die Skala 4 und punktweise Addition gewonnen. Die drei Auffälligkeitskarten berechnen sich somit wie folgt:

$$\bar{\mathcal{I}} = \bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} \mathcal{N}(\mathcal{I}(c, s)) \quad [7.15]$$

$$\bar{\mathcal{C}} = \bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} [\mathcal{N}(\mathcal{RG}(c, s)) + \mathcal{N}(\mathcal{BY}(c, s))] \quad [7.16]$$

$$\bar{\mathcal{O}} = \sum_{\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}} \mathcal{N} \left( \bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} \mathcal{N}(\mathcal{O}(c, s, \theta)) \right) \quad [7.17]$$

Jede dieser Auffälligkeitskarten durchläuft anschließend weitere zehn Iterationen der in Gleichung 7.14 beschriebenen Normalisierung. Die Motivation für diese Vorgehensweise besteht darin, dass ähnliche Merkmale stark in Bezug auf die Aufmerksamkeit konkurrieren, wohingegen unterschiedliche Modalitäten unabhängig zur Aufmerksamkeitskarte beitragen. Darüber hinaus wird auf diese Weise sichergestellt, dass nur wenige stark ausgeprägte Peaks innerhalb der einzelnen Merkmalstypen existieren.

Abschließend werden die drei normalisierten Auffälligkeitskarten linear zur Eingabe  $\mathcal{S}$  für die Aufmerksamkeitskarte  $\mathcal{SM}$  aufaddiert, welche sich auf Skala 4 (entspricht einem Reduktionsfaktor 1:16 verglichen mit dem ursprünglichen Bild) befindet:

$$\mathcal{S} = \frac{1}{3} (\mathcal{N}(\bar{\mathcal{I}}) + \mathcal{N}(\bar{\mathcal{C}}) + \mathcal{N}(\bar{\mathcal{O}})) \quad [7.18]$$

Das Maximum der Aufmerksamkeitskarte  $\mathcal{SM}$  definiert zu jedem Zeitpunkt die auffälligste

Bildposition, auf die sich der Aufmerksamkeitfokus als nächstes richten sollte. Die Aufmerksamkeitskarte  $SM$  wird als zweischichtiges Neuronales Netzwerk modelliert, das aus schlecht isolierten *integrate-and-fire* Neuronen besteht. Jedes Neuron besteht aus einer schlecht isolierten Kapazität, welche von der vom synaptischen Eingang gelieferten Ladung aufgeladen wird, und einem Spannungsschwellenwert. Ist der Schwellenwert erreicht, feuert das Neuron und die Kapazität wird vollständig entladen. Die Aufmerksamkeitskarte  $SM$  dient als Eingabe für ein weiteres Neuronales Netzwerk, das nach dem *winner-take-all* Prinzip (WTA) funktioniert und sicherstellt, dass jeweils nur die aktivste Region übrig bleibt und alle anderen Regionen unterdrückt werden. Dieses befindet sich wie die Aufmerksamkeitskarte  $SM$  ebenfalls auf Skala 4. Die Neuronen der Aufmerksamkeitskarte erhalten ihre anregende Eingabe aus  $S$  und sind voneinander unabhängig. Die Kapazitäten der Neuronen, die sich an auffälligen Bildpositionen befinden, werden dementsprechend schneller aufgeladen. Jedes  $SM$ -Neuron regt das zugehörige WTA-Neuron an. Alle WTA-Neuronen entwickeln sich ebenfalls unabhängig voneinander, bis eines (der „Gewinner“) zuerst den Schwellenwert erreicht und feuert. Hierdurch werden drei simultane Mechanismen ausgelöst:

1. Der Aufmerksamkeitfokus richtet sich auf das Gewinner-Neuron.
2. Die globale Inhibition des WTA-Netzwerks wird aktiviert und unterdrückt alle WTA-Neuronen (setzt sie zurück).
3. Eine lokale Inhibition in der Aufmerksamkeitskarte wird vorübergehend aktiviert. Die gehemmte Region entspricht der Größe und der neuen Position des Aufmerksamkeitfokus. Dies führt nicht nur dazu, dass die nächstauffälligere Region daraufhin der Gewinner wird, sondern sorgt auch dafür, dass der Aufmerksamkeitfokus nicht direkt zur vorhergehenden Position zurückkehrt.

Der Aufmerksamkeitfokus entspricht dabei einer einfachen Kreisschreibe, deren Radius fest auf  $1/6$  der kürzeren Bildseite eingestellt ist. Zusammenfassend sind die einzelnen Schritte zur Bestimmung auffälliger Bildbereiche nochmals in Abbildung 7.10 dargestellt.

## Segmentierung

Das bisher vorgestellte Verfahren zur Erzeugung der Aufmerksamkeitskarte eignet sich zur Bestimmung der auffälligsten Bildposition, es macht jedoch keinerlei Aussagen über die Ausdehnung des Objekts bzw. des Objektteils, auf den sich der Aufmerksamkeitfokus richtet. In [103] wird eine Methode zur Schätzung dieser Regionen vorgestellt, welche auf den bereits berechneten Karten und auffälligen Bildpositionen beruht, indem rückführende Verbindungen in der Berechnungshierarchie verwendet werden. Hierfür wird zunächst die Auffälligkeitskarte

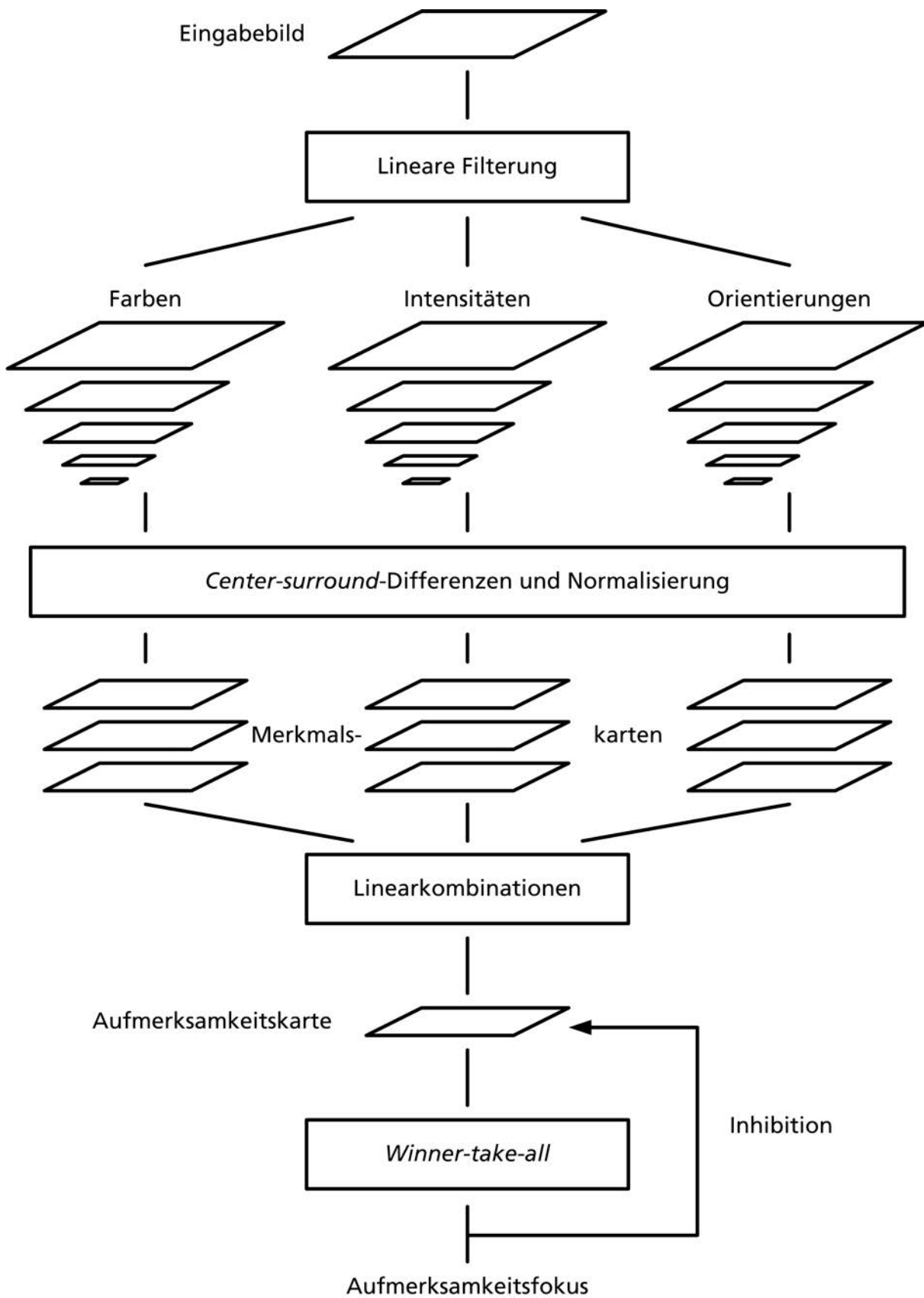


Abb. 7.10.: Verfahren zur Bestimmung auffälliger Bildbereiche [56].



bestimmt, die am meisten zur Aktivität an der auffälligsten Bildposition  $(x_w, y_w)$  beiträgt:

$$k_w = \arg \max_{k \in \{I, C, O\}} \mathcal{C}_k(x_w, y_w), \quad [7.19]$$

wobei  $\mathcal{C}_k$  wie folgt definiert ist:

$$\mathcal{C}_I = \mathcal{N}(\bar{I}), \quad \mathcal{C}_C = \mathcal{N}(\bar{C}), \quad \mathcal{C}_O = \mathcal{N}(\bar{O}) \quad [7.20]$$

Anschließend werden die Merkmalskarten betrachtet, aus denen die Auffälligkeitskarte  $\mathcal{C}_{k_w}$  berechnet wird, und es wird diejenige Merkmalskarte bestimmt, die am meisten zur Aktivität an der „Gewinner“-Bildposition beiträgt:

$$(l_w, c_w, s_w) = \arg \max_{l \in L_{k_w}, c \in \{2, 3, 4\}, s \in \{c+3, c+4\}} \mathcal{F}_{l, c, s}(x_w, y_w), \quad [7.21]$$

wobei  $L_{k_w}$  und  $\mathcal{F}_{l, c, s}$  wie folgt definiert sind:

$$L_I = \{I\}, \quad L_C = \{RG, BY\}, \quad L_O = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\} \quad [7.22]$$

$$\mathcal{F}_{I, c, s} = \mathcal{N}(\mathcal{I}(c, s)) \quad [7.23]$$

$$\mathcal{F}_{RG, c, s} = \mathcal{N}(\mathcal{RG}(c, s)) \quad [7.24]$$

$$\mathcal{F}_{BY, c, s} = \mathcal{N}(\mathcal{BY}(c, s)) \quad [7.25]$$

$$\mathcal{F}_{\theta, c, s} = \mathcal{N}(\mathcal{O}(c, s, \theta)) \quad [7.26]$$

Die „Gewinner“-Merkmalskarte wird anschließend um die Bildposition  $(x_w, y_w)$  herum segmentiert. Hierfür wird eine binarisierte Version der Karte ( $\mathcal{B}$ ) verwendet, welche mit Hilfe des folgenden Schwellenwertverfahrens erzeugt wird:

$$\mathcal{B}(x, y) = \begin{cases} 1 & \text{wenn } \mathcal{F}_{l_w, c_w, s_w}(x, y) \geq 0,1 \cdot \mathcal{F}_{l_w, c_w, s_w}(x_w, y_w) \\ 0 & \text{sonst} \end{cases} \quad [7.27]$$

Die 4er-Nachbarschaft der aktiven Pixel in  $\mathcal{B}$  dient als Schablone zur Schätzung der Bildregion des Objekts, auf das sich der Aufmerksamkeitsfokus richtet:

$$\hat{\mathcal{F}}_w = \text{label}(\mathcal{B}, (x_w, y_w)), \quad [7.28]$$

wobei die *label*-Funktion aus [85] verwendet wird. Die auf diese Weise segmentierte Merkmalskarte  $\hat{\mathcal{F}}_w$  wird darüber hinaus anstelle des kreisförmigen Aufmerksamkeitsfokus zur Inhibition der Aufmerksamkeitskarte verwendet.

### 7.3.2. Klassifikation anhand von Farbtexturmerkmalen

Für die Klassifikation der mit dem oben beschriebenen Verfahren segmentierten Bildbereiche werden Farbtexturmerkmale verwendet (siehe [11]). Unter Farbtexturmerkmalen sind dabei Texturmerkmale zu verstehen, die nicht nur die Intensitätsverteilung, sondern auch die Farbverteilung in einem bestimmten Bildbereich berücksichtigen. Eine Verallgemeinerung der klassischen Texturmerkmale (siehe Tabelle 7.1) auf mehrere Farbkanäle besteht in der getrennten Anwendung der Merkmalsextraktion auf jedem der Kanäle. Dabei entstehen mehrere Merkmalsvektoren, die anschließend zu einem Gesamtvektor konkateniert werden. Da bei Außenaufnahmen die Lichtintensität sehr stark variiert, wird vor der Berechnung der Farbtexturmerkmale zunächst eine Konvertierung des RGB-Bildes in den HSL-Farbraum durchgeführt (siehe [1, S. 305]). Die Komponenten des HSL-Farbraums stehen für den Farbwert H (engl. *Hue*), die Farbsättigung S (engl. *Saturation*) und die Helligkeit L (engl. *Lightness*).

#### Flächenbasierte Merkmale

Klassische Texturmerkmale der Zweipunktstatistik sind die von Haralick [84] definierten Merkmale der Grauwertübergangsmatrix  $P(i, j)$  (siehe Tabelle 7.1). Die Summe der Einträge von  $P$

Merkmalsname	Berechnungsformel
Energie	$\sum_i \sum_j P^2(i, j)$
Entropie	$-\sum_i \sum_j P(i, j) \log P(i, j)$
Trägheitsmoment	$\sum_i \sum_j (i - j)^2 P(i, j)$
Homogenität	$\sum_i \sum_j \frac{P(i, j)}{1 +  i - j }$
Korrelation	$\sum_i \sum_j \frac{(i - \mu_x)(j - \mu_y)P(i, j)}{\sigma_x \sigma_y}$

Tab. 7.1.: Texturmerkmale nach Haralick [99].

ist dabei auf 1 normiert. Grauwertübergangsmatrizen beschreiben die Verbundwahrscheinlichkeit, mit der zwei Grauwerte an zwei Bildpositionen vorkommen, die sich um den Vektor  $\mathbf{d}$  versetzt zueinander befinden. Der Vektor  $\mathbf{d}$  wird meistens fest gewählt, wobei eine entsprechende Vorzugsrichtung zu finden ist. Alternativ wird über alle Grauwertübergangsmatrizen gemittelt, deren Richtungsvektor  $\mathbf{d}$  einen vorgegebenen Längenbereich  $d_{min} \leq |\mathbf{d}| \leq d_{max}$  einhält. Dadurch entsteht automatisch ein rotationsinvarianter Texturdeskriptor. Eine höhere Toleranz  $d_{max} - d_{min}$  sorgt für eine höhere Anzahl an Histogrammeinträgen und somit für einen stabileren Deskriptor. Allerdings nimmt die Rechenzeit mit jeder betrachteten Pixelkonstellation zu. Farbverbundhistogramme (engl. *Color Cooccurrence Histogram* (CCH)) sind die Erweiterung

der Grauwertübergangsmatrix auf mehrere Farbkanäle und werden in [26] zur Objekterkennung eingesetzt.

**Farbverbundhistogramme** Die Berechnung des Verbundhistogramms erfolgt, indem zunächst alle Verschiebungsvektoren  $\mathbf{d}$  mit Hilfe eines *Bresenham*-Algorithmus [1] ermittelt werden (siehe [11]). Für jede Pixelposition  $\mathbf{p}$  der zu klassifizierenden Region wird bestimmt, ob  $\mathbf{p} + \mathbf{d}$  ebenfalls in der Region liegt, und gegebenenfalls wird die Zelle im Verbundhistogramm  $P_0$  um eins erhöht. Das Verbundhistogramm des vollständigen Kreises ergibt sich wie folgt:

$$P = \frac{P_0 + P_0^T}{\sum_{i,j} P_0(i,j)} \quad [7.29]$$

Das Verfahren wird sowohl für den Farbwert- als auch für den Sättigungskanal durchgeführt. Für die jeweiligen Verbundhistogramme werden anschließend einige der Texturmerkmale nach Haralick berechnet. Die in Tabelle 7.1 angegebenen Merkmale sind teilweise redundant. Beispielsweise messen Energie und Entropie, wie gleichmäßig die Farbwertpaare verteilt sind. Homogenität und Trägheitsmoment beschreiben hingegen, wie häufig sehr unterschiedliche Farbpaarungen im betrachteten Abstand vorkommen. Es werden daher nur Energie und Trägheitsmoment verwendet. Da die Korrelation für den Farbwertkanal problematisch ist (auf einem zyklischen Merkmal kann kein sinnvoller Mittelwert berechnet werden), wird sie durch ein Normalverteilungsmaß ersetzt (siehe [11]).

Die meisten Einträge der Verbundmatrix sind üblicherweise auf die Hauptdiagonale konzentriert, da durch Farbflächen zumeist Pixelpaare der gleichen Farbe den Toleranzbereich einhalten. Zu den Histogrammzellen abseits der Diagonalen tragen die Pixelpaare bei, deren Positionen auf den beiden Seiten einer Farbkante im Bild liegen. Falls die zu klassifizierende Region aus flächigen Bereichen zusammengesetzt ist, die sich im Farbton stark unterscheiden und durch Kanten mit hohem Farb- oder Sättigungskontrast getrennt sind, dann sollte die Verbundmatrix Histogrammzeilen enthalten, die mindestens ein Nebenmaximum abseits der Diagonalen aufweisen. Histogrammzeilen, deren Diagonalfarbe nur in der Nähe von ähnlichen Farben vorkommt, sollten demgegenüber als annähernd normalverteilt modelliert werden. Für den Sättigungskanal werden für jede Zeile  $Z(i)$  der Mittelwert  $\mu$  und die Varianz  $\sigma^2$  bestimmt. Aus der Normalverteilung mit diesen Parametern wird das Modellhistogramm  $M(i)$  gebildet:

$$M(i) = \frac{1}{\sqrt{2\pi}} \frac{N\Delta_x}{\sigma} \exp - \left( \frac{\Delta(i + \frac{1}{2}, \mu)}{2\sigma} \right)^2, \quad [7.30]$$

wobei  $N$  die Anzahl der Histogrammzellen in einer Zeile und  $\Delta_x$  die Breite einer Histogrammzelle bezeichnen. Für das Sättigungshistogramm gilt  $\Delta_x = 1/N$  und als Abstand wird  $\Delta(i, j) = ||i - j||$  verwendet. Für das Farbwertshistogramm wird  $\Delta_x = 360/N$  gewählt. Da sich für die-

sen Fall kein sinnvoller Mittelwert  $\mu$  definieren lässt, wird dieser durch den Modus des Histogramms ersetzt. Die Varianz wird mit dem durchschnittlichen quadratischen Abstand zum Modus ersetzt. Als Abstand wird der kleinere Winkel zwischen den Farbwerten verwendet:

$$a = ||j - i|| \quad [7.31]$$

$$\Delta(i, j) = \begin{cases} a & a \leq 180 \\ 360 - a & \text{sonst} \end{cases} \quad [7.32]$$

Die Verteilungen  $Z$  und  $M$  werden wie folgt verglichen:

$$v = \sum_{i=1}^N \frac{||M(i) - Z(i)||}{M(i) + Z(i) + 1} \quad [7.33]$$

Als Normalverteilungsmaß wird schließlich  $1/v$  verwendet.

Neben Energie, Trägheitsmoment, Normalverteilungsmaß und Maximum  $\max_{i,j} P(i, j)$  wird außerdem die Energie des Standardhistogramms mit  $N$  Zellen als Merkmal der Einpunktstatistik hinzugenommen. Das Standardhistogramm kann aus der Verbundmatrix durch Aufsummieren der Matrixzeilen berechnet werden. Somit ergibt sich die Energie des Standardhistogramms wie folgt:

$$E_1 = \sum_i \left( \sum_j P(i, j) \right)^2 \quad [7.34]$$

Der vollständige Merkmalsvektor aus den Farbverbundhistogrammen setzt sich somit aus den Kennzahlen Energie, Trägheitsmoment, Normalverteilungsmaß, Maximum und Energie des Standardhistogramms jeweils für den Farbwert- und den Sättigungskanal zusammen.

**Lokale Binäre Muster** In [77] wird ein auf so genannten *Lokalen Binären Mustern* (engl. *Local Binary Patterns* (LBP)) basierender Ansatz zur Klassifikation von Texturen vorgestellt. Zur Charakterisierung der Struktur einer lokalen Bildtextur wird der folgende Texturoperator definiert:

$$LBP_{P;R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p, \quad [7.35]$$

wobei  $g_c$  den Grauwert des zentralen Pixels einer lokalen Nachbarschaft bezeichnet,  $g_p$  ( $p = 0, \dots, P-1$ ) den Grauwerten von  $P$  Punkten mit gleichmäßigem Abstand auf einem Kreis mit dem Radius  $R$  ( $R > 0$ ) entspricht und  $s(x)$  wie folgt definiert ist:

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad [7.36]$$

Der Name *Lokale Binäre Muster* reflektiert die Funktionalität des Operators, da dieser die Grauwerte einer lokalen Nachbarschaft, die mit dem Grauwert des zentralen Pixels verglichen werden, in ein binäres Muster abbildet. Der  $LBP_{P,R}$ -Operator ist per Definition invariant gegenüber monotonen Transformationen der Grauwertskala.

Der  $LBP_{P,R}$ -Operator liefert  $2^P$  verschiedene Ausgabewerte, welche den  $2^P$  verschiedenen binären Mustern entsprechen, die von den  $P$  Pixeln der Nachbarschaftsmenge gebildet werden können. Die Rotation eines binären Musters um den zentralen Pixel der lokalen Nachbarschaft führt allerdings in der Regel zu einem unterschiedlichen  $LBP_{P,R}$ -Wert. Es lässt sich jedoch wie folgt ein rotationsinvarianter Texturoperator definieren:

$$LBP_{P,R}^{ri} = \min \{ ROR(LBP_{P,R}, i) \mid i = 0, 1, \dots, P-1 \}, \quad [7.37]$$

wobei  $ROR(x, i)$  einen zyklischen Bitshift der aus  $P$  Bits bestehenden Zahl  $x$  um  $i$  Bits nach rechts bezeichnet. Der in Gleichung 7.37 definierte Operator entspricht einer Verschiebung der Nachbarschaftsmenge im Uhrzeigersinn, so dass eine maximale Anzahl der führenden Bits (engl. *most significant bits*), beginnend mit  $g_{P-1}$ , den Wert 0 annimmt.

Als Merkmale für die zu klassifizierende Region werden im Rahmen dieser Arbeit die Operatoren  $LBP_{8;1}^{ri}$  und  $LBP_{12;1,5}^{ri}$  verwendet, welche aus der  $3 \times 3$ - und der  $5 \times 5$ -Nachbarschaft eines Pixels berechnet werden. Die kreisförmige Nachbarschaftsmenge wird dabei jeweils durch Rastern angenähert (siehe Abbildung 7.11). Der  $LBP_{12;1,5}^{ri}$ -Operator wird auf jeden Bildpunkt der zu klassifizierenden Region angewendet, der mindestens zwei Pixel Abstand zur Bounding Box der Region hat. Analog wird der  $LBP_{8;1}^{ri}$ -Operator auf alle Bildpunkte angewendet, die mindestens einen Pixel Abstand zur Bounding Box der Region haben. Die Ergebnisse werden jeweils in einem entsprechenden Histogramm gespeichert. Das Histogramm für den  $LBP_{8;1}^{ri}$ -Operator enthält 36 verschiedene Histogrammklassen und das Histogramm für den  $LBP_{12;1,5}^{ri}$ -Operator 352 Histogrammklassen. Dies entspricht jeweils der Anzahl der unterschiedlichen rotationsinvarianten binären Muster mit 8 bzw. 12 Nachbarschaftspunkten. Die Histogramme werden anschließend auf 1 normiert und bilden konkateniert einen weiteren Teil des Gesamtmerkmalsvektors (siehe Abbildung 7.12).

### Lokal invariante Merkmale

Neben den auf der ganzen Fläche der zu klassifizierenden Region berechneten globalen Texturmerkmalen aus Farbverbundhistogrammen und Lokalen Binären Mustern, werden als dritte Komponente lokale Bildmerkmale verwendet. Hierfür werden zunächst mit Hilfe eines Detektors interessierende Bildpunkte (auch Schlüsselpunkte genannt, engl. *keypoints*) in der Region bestimmt. Anschließend wird die Nachbarschaft jedes interessierenden Bildpunkts mit Hilfe eines Deskriptors als Merkmalsvektor repräsentiert. Als Detektor kommt im Rahmen dieser Ar-

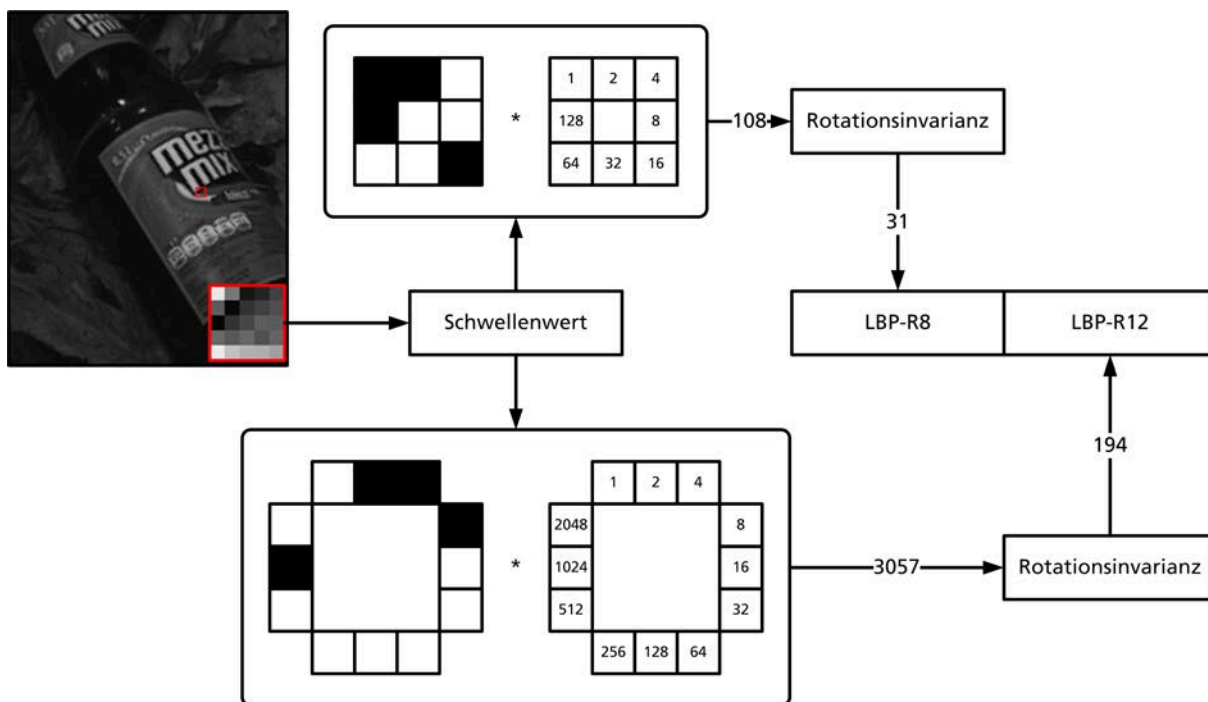


Abb. 7.11.: Lokale Binäre Muster [11].

beit der in [9] vorgestellte *Fast-Hessian*-Detektor zum Einsatz, als Deskriptor wird der ebenfalls in [9] vorgestellte *SURF*-Deskriptor (engl. *Speeded-Up Robust Features* (SURF)) verwendet.

Mit dem beschriebenen Verfahren werden in jeder Region unterschiedlich viele interessierende Bildpunkte gefunden und somit stehen jeweils auch unterschiedlich viele Deskriptoren zur Charakterisierung der Region zur Verfügung. Für die im Rahmen dieser Arbeit gewählte Klassifikation mittels Support-Vektor-Maschinen (engl. *Support Vector Machines* (SVMs)) mit RBF-Kernel (siehe unten) wird jedoch ein Merkmalsvektor fester Länge benötigt. Aus diesem Grund wird ein in [109] beschriebener Ansatz zur Abbildung von Deskriptormengen auf eine feste Anzahl von Merkmalen mit Hilfe eines globalen *Texturvokabulars* verwendet. Hierzu werden die Deskriptoren einer speziellen Trainingsdatenmenge geclustert und jede Region wird anschließend als Histogramm repräsentiert, indem jeder lokale Deskriptor mit einem Nächste-Nachbar-Verfahren einem der *Texturworte* zugeordnet und der entsprechende Histogrammeintrag erhöht wird. Im Rahmen dieser Arbeit werden, wie in [109] vorgeschlagen, mit einem *k-means*-Verfahren für jede Entitätsklasse separat jeweils 10 Texturworte (Cluster) bestimmt. Die Texturworte der einzelnen Entitätsklassen werden anschließend zu einem globalen Texturvokabular konkateniert. Die relativen Häufigkeiten der einzelnen Texturworte für eine interessierende Region werden an den bestehenden Merkmalsvektor angehängt (siehe Abbildung 7.12).

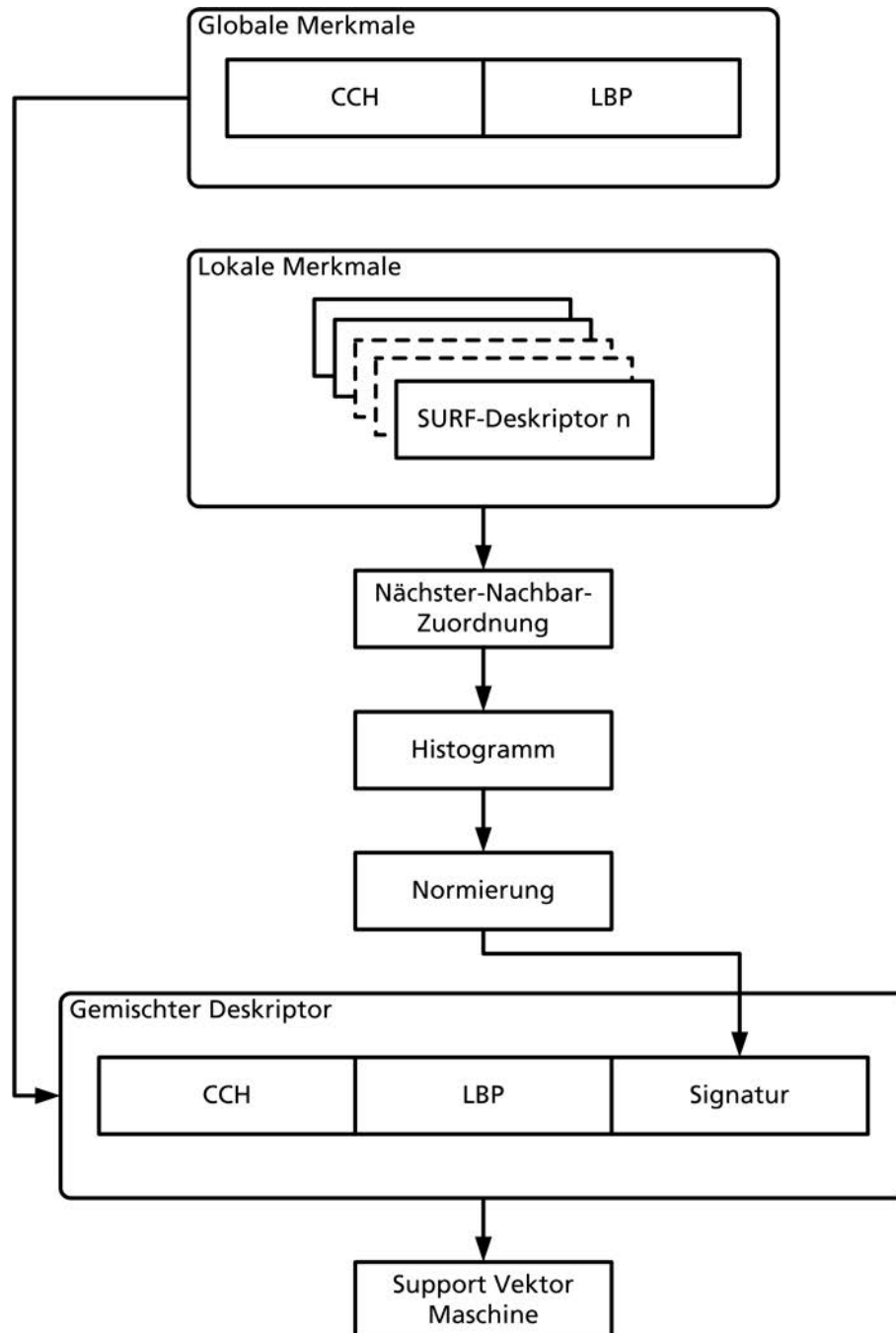


Abb. 7.12.: Überblick über die verwendeten Texturmerkmale.

## Klassifikation

Basierend auf den im vorherigen Abschnitt beschriebenen Farbtexturmerkmalen wird eine Klassifikation der segmentierten Bildbereiche mit Support-Vektor-Maschinen (siehe [102], [90] und [23]) durchgeführt. Diese gliedert sich in bis zu drei Stufen (siehe auch Abschnitt 7.4.4). Dabei erfolgt zunächst eine Klassifikation in die Klassen *Abfall* bzw. *Kein Abfall*. Ist die Klasse *Abfall* am wahrscheinlichsten, wird anschließend eine Klassifikation der Abfallart vorgenommen. Im Rahmen dieser Arbeit werden exemplarisch die fünf Abfallklassen *Flasche*, *Papier*, *Plastiktüte*, *Tetrapak* und *Dose* unterschieden. Weist das Klassifikationsergebnis eine geringe Konfidenz auf, kann darüber hinaus eine Absicherung der Abfallart mit Hilfe einer Ein-Klassen-SVM, der so genannten Träger-Schätzung [91], vorgenommen werden. Für alle drei Arten von Klassifikatoren wird die Bibliothek LIBSVM<sup>15</sup> eingesetzt und jeweils ein RBF-Kernel (engl. *Radial Basis Function* (RBF)) verwendet:

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\gamma \|\vec{x}_i - \vec{x}_j\|^2\right) \quad [7.38]$$

Für das Training der SVM-Klassifikatoren wurden unter Laborbedingungen Bilddaten von verschiedenen Objekten der einzelnen Klassen aufgezeichnet. Die Bilddaten wurden mit dem in Abschnitt 7.3.1 beschriebenen Verfahren automatisch segmentiert. Die auf diese Weise gewonnenen interessierenden Bildregionen wurden anschließend halbautomatisch mit der entsprechenden Klasseninformation annotiert. Eine Übersicht über die Anzahl der aufgezeichneten Objekte und der segmentierten interessierenden Bildbereiche in den einzelnen Kategorien bietet Tabelle 7.2. Pro Objekt wurden jeweils mindestens drei verschiedene Ansichten aufgenommen. Ein vollständiger Überblick über die aufgezeichneten Bilddaten befindet sich in Anhang B.

Klasse	Objekte	Interessierende Regionen
Flasche	12	99
Papier	13	106
Plastiktüte	8	88
Tetrapak	8	67
Dose	9	66
Abfall	50	426
Kein Abfall	25	318
<b>Insgesamt</b>	<b>75</b>	<b>744</b>

Tab. 7.2.: Übersicht über die Bilddaten zur Detektion und Klassifikation von Abfall.

Zur Erzielung einer möglichst guten Generalisierungsfähigkeit wurde für das Training der Klassifikatoren eine Kreuzvalidierung in Kombination mit einer gridbasierten Parametersuche

<sup>15</sup>LIBSVM: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>



durchgeführt. Die Aufteilung der segmentierten Bildbereiche in Lern- und Testdaten entsprechend eines vorgegebenen Prozentsatzes erfolgte dabei in drei verschiedenen Varianten: blind, nach Zugehörigkeit zu den einzelnen Objektansichten und nach Zugehörigkeit zu den jeweiligen Objekten. Bei den beiden letztgenannten Varianten wurden die zu einer Objektansicht bzw. einem Objekt gehörenden segmentierten Bildbereiche jeweils vollständig entweder der Lern- oder der Testdatenmenge zugeordnet. Somit dienen diese Varianten als Indikator dafür, wie gut von bekannten Objektansichten auf unbekannte Objektansichten bzw. von bekannten Objekten auf unbekannte Objekte generalisiert werden kann. Die Parameter der SVM-Klassifikatoren wurden für die jeweilige Aufteilung der Daten systematisch in einem definierten Wertebereich variiert. Die vollständigen Ergebnisse der Kombination aus Kreuzvalidierung und gridbasierter Parametersuche für die einzelnen Klassifikatoren befinden sich in Anhang C. Neben unterschiedlichen Prozentsätzen für die Aufteilung in Lern- und Testdaten wurden dabei auch gezielt die verschiedenen Merkmalsgruppen untersucht.

In Tabelle 7.3 sind die jeweils niedrigsten mittleren Fehlerraten für die einzelnen Klassifikatoren in Abhängigkeit von der Art der Kreuzvalidierung zusammenfassend dargestellt.

<b>Klassifikator</b>	<b>Objekt</b>	<b>Ansicht</b>	<b>Blind</b>
Klassifikation Abfall / Kein Abfall	0,110854	0,0784784	0,067391
Klassifikation der Abfallart	0,58647	0,498083	0,352083
Absicherung der Abfallart Flasche	0,0352655	0,0423802	0,0166667
Absicherung der Abfallart Papier	0,0404015	0,00735294	0,02
Absicherung der Abfallart Plastiktüte	0,0227273	0,0166667	0,0113636
Absicherung der Abfallart Tetrapak	0,0513393	0,0217391	0,0166667
Absicherung der Abfallart Dose	0,0430403	0,0584416	0,025

Tab. 7.3.: Niedrigste mittlere Fehlerraten der Klassifikatoren für die einzelnen Varianten der Kreuzvalidierung.

Das beste Ergebnis für die Klassifikation Abfall / Kein Abfall weist eine mittlere Fehlerrate von 7 % auf. Die niedrigste mittlere Fehlerrate für die Klassifikation der Abfallart liegt bei 35 %. Diese Werte werden jeweils mit der blinden Aufteilung der segmentierten Bildbereiche erzielt. Die mittleren Fehlerraten für die Kreuzvalidierung mit Aufteilung nach Zugehörigkeit zu den Objektansichten bzw. zu den Objekten liegen hingegen deutlich höher. Die besten Ergebnisse für die Absicherung der Abfallart bewegen sich über alle Klassen hinweg bei einer mittleren Fehlerrate im Bereich von 1 bis 2 %.

## 7.4. Umsetzung der Kernkomponenten

In diesem Abschnitt wird auf die Umsetzung der vier Kernkomponenten der semantischen Missionssteuerung eingegangen. Dies umfasst die Steuerungsarchitektur, die Wissensbasis, die

Erzeugung und Ausführung von Inspektionsplänen sowie die aktive Untersuchung interessierender Entitäten.

### 7.4.1. Steuerungsarchitektur

Die in Kapitel 3 vorgestellte Architektur der semantischen Missionssteuerung wurde mit Hilfe der *Modular Controller Architecture 2* (MCA2) [100] umgesetzt. Dabei handelt es sich um ein plattformunabhängiges, netzwerktransparentes und echtzeitfähiges Softwarerahmenwerk, welches auf C++ basiert.

Die semantische Missionssteuerung wurde als so genannter MCA-Part realisiert und die einzelnen Komponenten der Steuerungsarchitektur wurden in Form von entsprechenden MCA-Gruppen und MCA-Modulen umgesetzt (siehe Abbildung 7.13). Die MCA-Gruppen und MCA-Module sind dabei über so genannte Kanten miteinander verbunden, welche die Kommunikation zwischen den einzelnen Komponenten übernehmen. Für den Austausch größerer und variabler Datenmengen werden so genannte Blackboards verwendet, die dem Entwurfsmuster *Tafel* (siehe [36]) entsprechen.

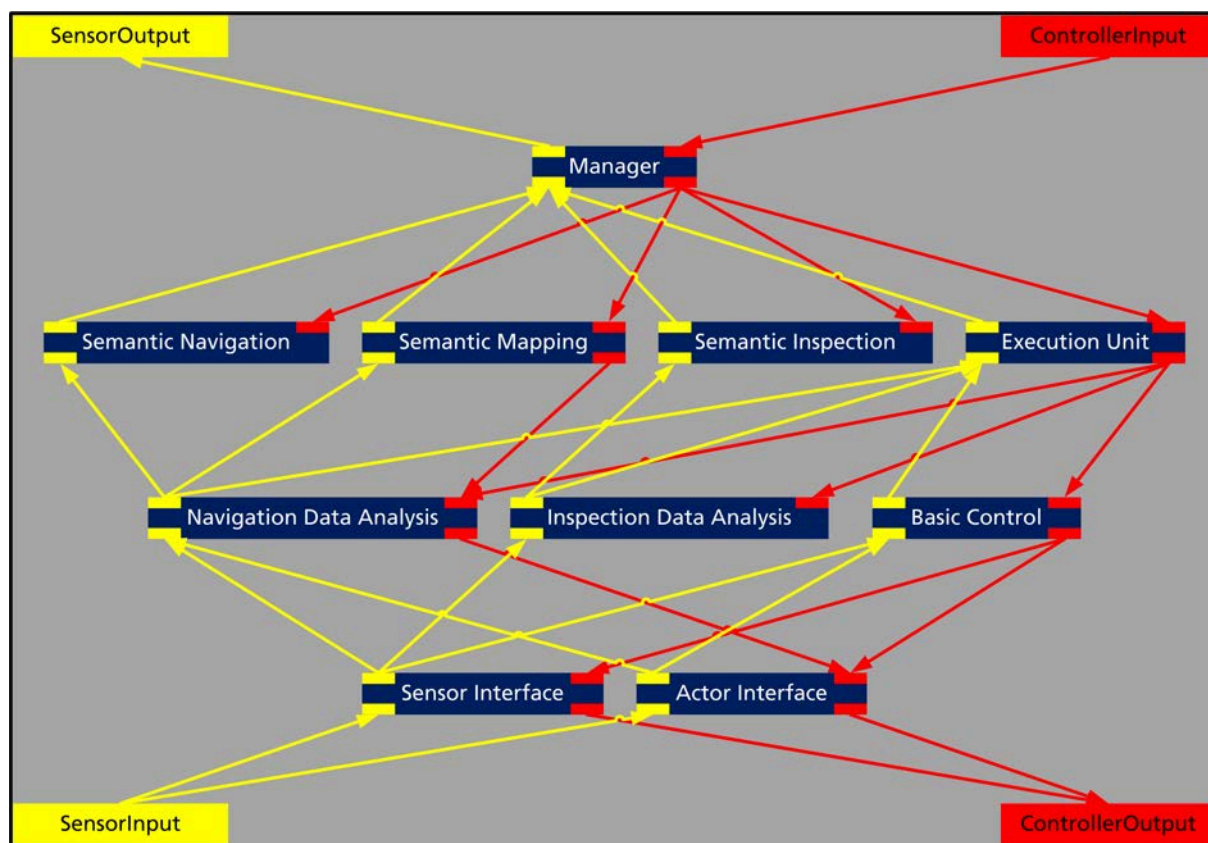


Abb. 7.13.: Überblick über die semantische Missionssteuerung von LAURON IV.

Die semantische Missionssteuerung setzt auf den bereits existierenden Lokomotions- und Navigationsfähigkeiten von LAURON IV auf, welche ebenfalls als MCA-Parts realisiert sind

(siehe Abbildung 7.14) und sich wie folgt auf die beiden Steuerungsrechner verteilen: Der so genannte Hardware Abstraction Layer (*LauronHal*), die verhaltensbasierte Steuerung (*BehaviourControl*) und die Lokalisation (*Localisation*) werden auf PC-1 ausgeführt. Der Sensor Abstraction Layer (*LauronSal*), die Navigation (*Navigation*) und die semantische Missionssteuerung (*MissionControl*) werden auf PC-2 ausgeführt.

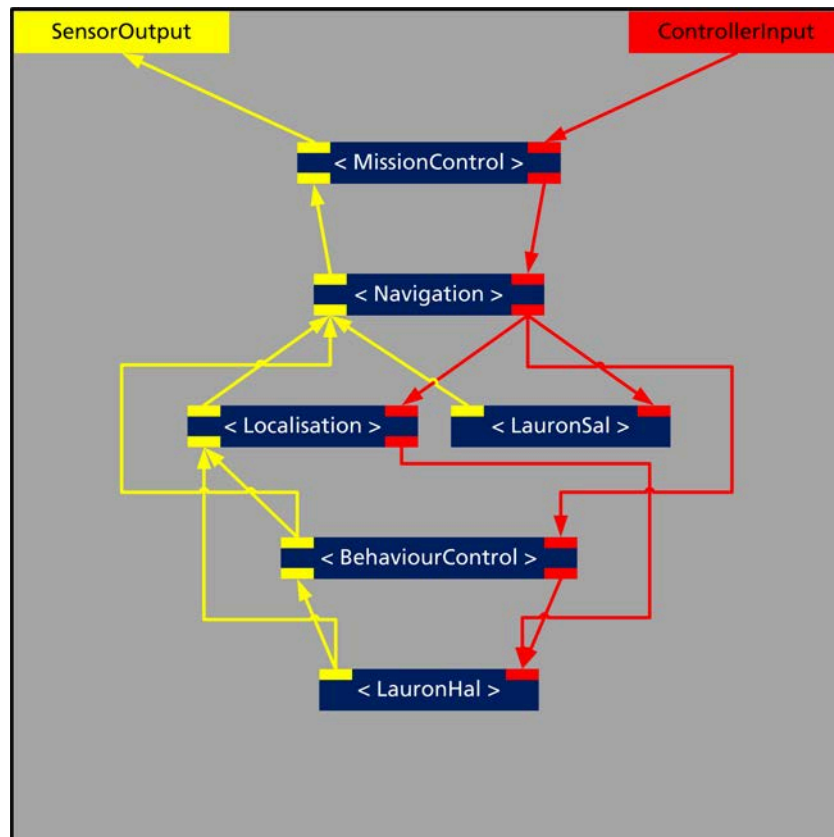


Abb. 7.14.: Überblick über die Gesamtarchitektur der Steuerung von LAURON IV.

### Ausführungseinheit und Ausführungskomponenten

Für die Ausführungskomponenten wurde ein spezielles MCA-Modul entwickelt, von dem die einzelnen Ausführungskomponenten erben. Es besitzt einen generischen Aufbau und übernimmt die Kommunikation mit der Ausführungseinheit, so dass sich die Implementierung der einzelnen Ausführungskomponenten auf die reine Funktionalität beschränkt. Der Ablauf innerhalb der Ausführungskomponenten wird mit endlichen Zustandsautomaten gesteuert. Neben den implementierten Fähigkeiten verfügen die Ausführungskomponenten auch über einen so genannten *interaktiven Modus*. In diesem werden für die einzelnen zu erfüllenden elementaren Aufgaben mit Hilfe der Bedienschnittstelle Fragen an den Benutzer gerichtet. Dies umfasst sowohl den Status der Ausführung (Erfolg oder Fehlschlag) als auch das Ergebnis der jeweiligen elementaren Aufgabe. Auf diese Weise war es möglich, das Gesamtsystem bereits in einem sehr

frühen Entwicklungsstadium in Betrieb zu nehmen und implementierte Fähigkeiten zusammen mit simulierten Fähigkeiten zu testen. Neben der Beantwortung der Fragen durch den Benutzer ist auch eine automatisierte Beantwortung möglich, so dass sich anhand von dedizierten Testfällen die Funktionsfähigkeit des Systems automatisch überprüfen lässt (siehe Abschnitt 8.2).

### **Bedienschnittstelle**

Die Bedienschnittstelle wurde mit Hilfe des MCA-Werkzeugs *mcgui* erstellt und umfasst sieben verschiedene Komponenten. Diese werden im Folgenden kurz zusammen mit ihren wichtigsten Funktionen vorgestellt.

**Hauptkomponente** Die *Hauptkomponente* (siehe Abbildung 7.15) dient der Vorgabe von Missionsaufgaben, dem Starten und Stoppen der Missionen sowie der Überwachung des Status der Missionen. Darüber hinaus kann die Übergabe der Kontrolle des Roboters vom menschlichen Operator an die semantische Missionssteuerung und umgekehrt vorgenommen werden. Des Weiteren können Prioritäten für die einzelnen Entitätsklassen vorgegeben und die jeweils aktuell ausgeführten elementaren Aufgaben überwacht werden. Wird die semantische Missionssteuerung im interaktiven Modus ausgeführt, werden die Fragen an den Benutzer zu den Ergebnissen der elementaren Aufgaben und die zugehörigen möglichen Antworten ebenfalls in dieser Komponente dargestellt.

**Lauron** Die Komponente LAURON (siehe Abbildung 7.16) dient dem Start und der Inbetriebnahme des Roboters. Darüber hinaus können der Sicherheitszustand der verhaltensbasierten Steuerung und die Spannung der Akkumulatoren überwacht werden. Zusätzlich können einige grundlegende Parameter der verhaltensbasierten Steuerung manuell angepasst werden.

**Navigation** Die Komponente *Navigation* (siehe Abbildung 7.17) erlaubt anhand einer grob aufgelösten globalen Karte die Vorgabe von zu inspizierenden Regionen sowie die Angabe eines Zielpunkts, an den der Roboter nach Abschluss der Inspektion zurückkehren soll. Zudem können die geplanten Pfade und das Ablaufen dieser Pfade überwacht werden. Eine weitere Aufgabe dieser Komponente besteht in der Visualisierung der Orte, an denen interessierende Entitäten gefunden wurden. So kann sich der menschliche Operator schnell einen Überblick über die Ergebnisse der Inspektion verschaffen.

**Flexible Hierarchische Pläne** Die Komponente *Flexible Hierarchische Pläne* (siehe Abbildung 7.18) dient der Überwachung der generierten Pläne. Hierzu wird die hierarchische Baumstruktur der Pläne dargestellt und die jeweils in der Ausführung befindlichen elementaren Aufgaben werden hervorgehoben. Darüber hinaus können detaillierte Informationen über

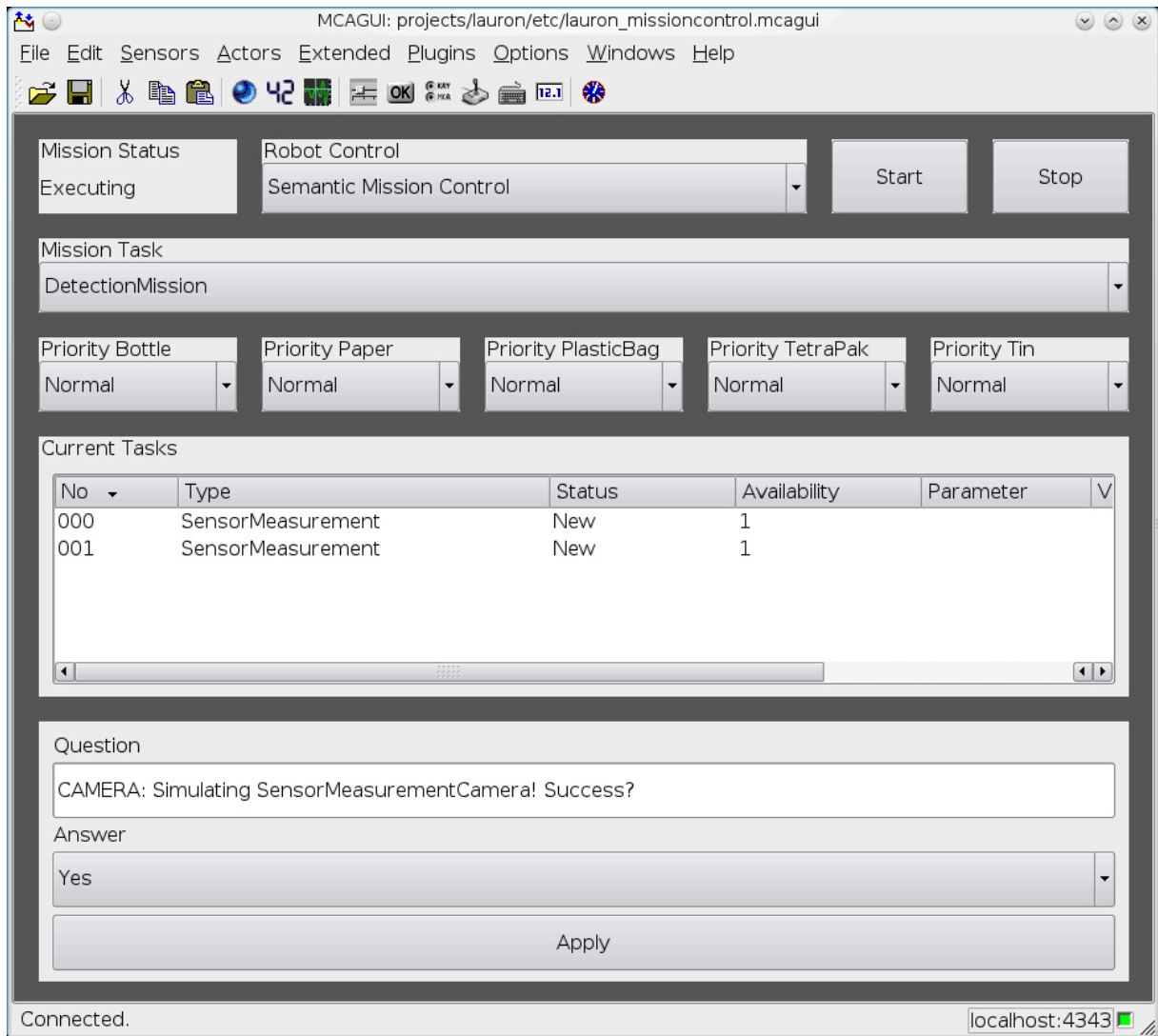


Abb. 7.15.: Die Bedienkomponente zur Vorgabe von Missionsaufgaben.

die einzelne Komponenten der Flexiblen Hierarchischen Pläne angezeigt werden. Dies umfasst insbesondere die Parameter und die zusammengefassten Vor-, Während- und Nachbedingungen sowie sie zusammengefassten Ressourcenverbräuche.

**Bilddaten** Die Komponente *Bilddaten* (siehe Abbildung 7.19) stellt die zuletzt mit den Kameras aufgenommenen Bilddaten dar. Darüber hinaus werden die in den Bilddaten segmentierten Bereiche sowie die zugehörigen Klassifikationsergebnisse visualisiert. Auf diese Weise kann sich der menschliche Operator schnell einen Überblick über die gerade aktuell verarbeiteten Bilddaten verschaffen.

**Inspektionsdaten** Die Komponente *Inspektionsdaten* (siehe Abbildung 7.20) bietet einen tabellarischen Überblick über die gesamten während der Inspektion verarbeiteten Daten. Für die in den einzelnen Verarbeitungsschritten der Inspektionsdatenauswertung anfallenden Daten

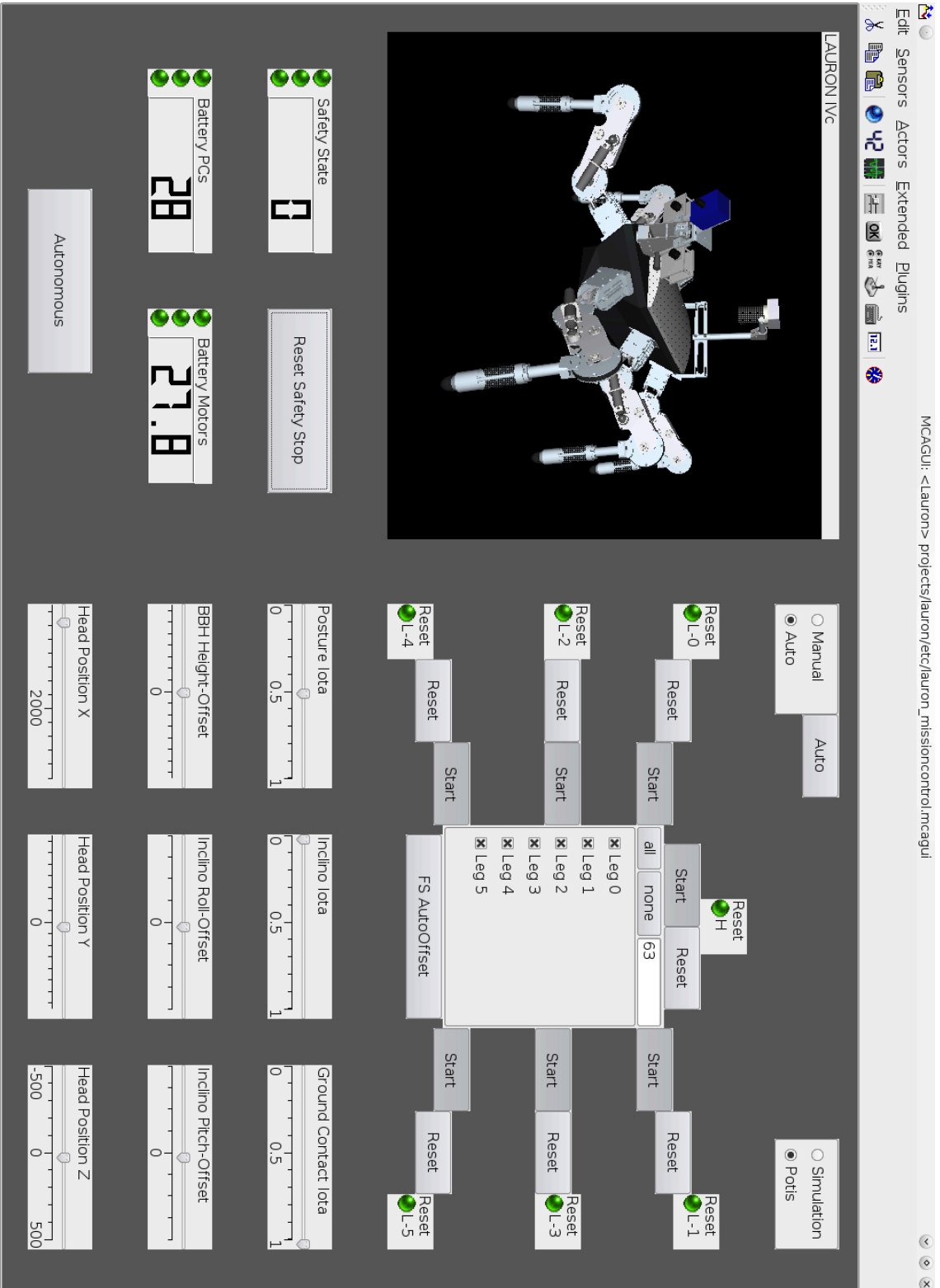


Abb. 7.16.: Die Bedienkomponente zur Inbetriebnahme und Überwachung von LAURON IV.

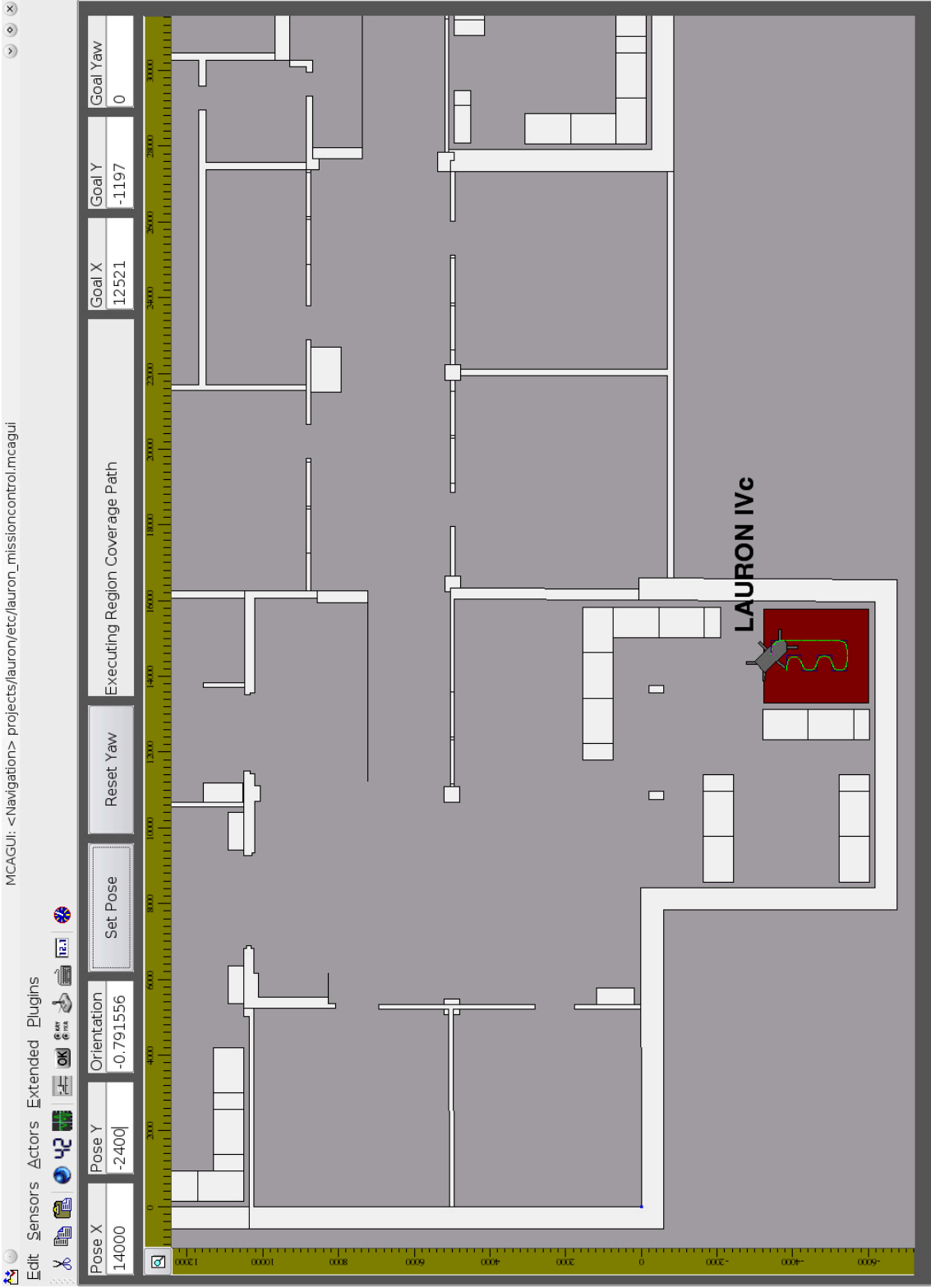


Abb. 7.17.: Die Bedienkomponente zur Vorgabe von zu inspizierenden Regionen sowie zur Anzeige von gefundenen interessierenden Entitäten.

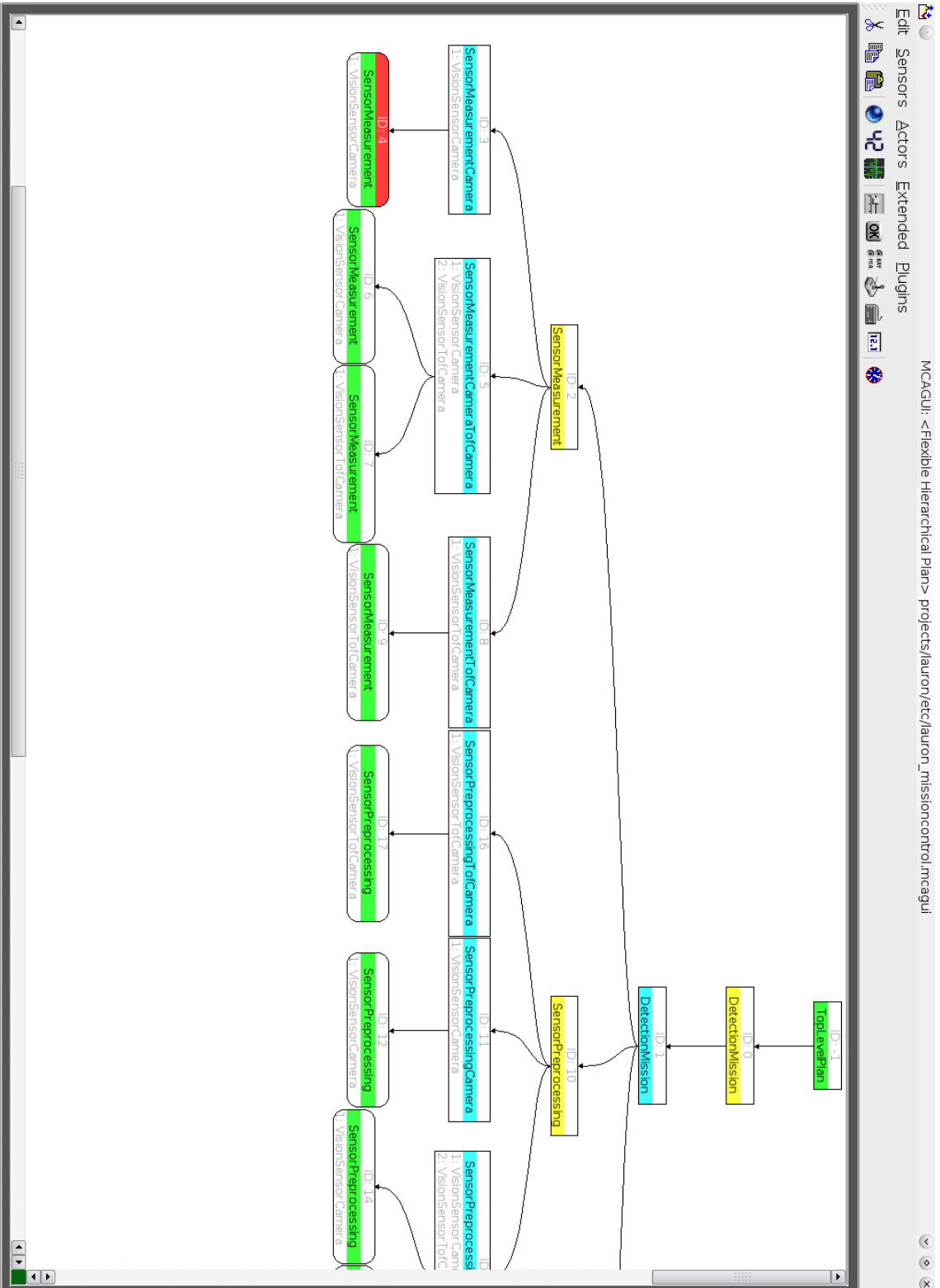


Abb. 7.18.: Die Bedienkomponente zur Überwachung der generierten Flexiblen Hierarchischen Pläne.



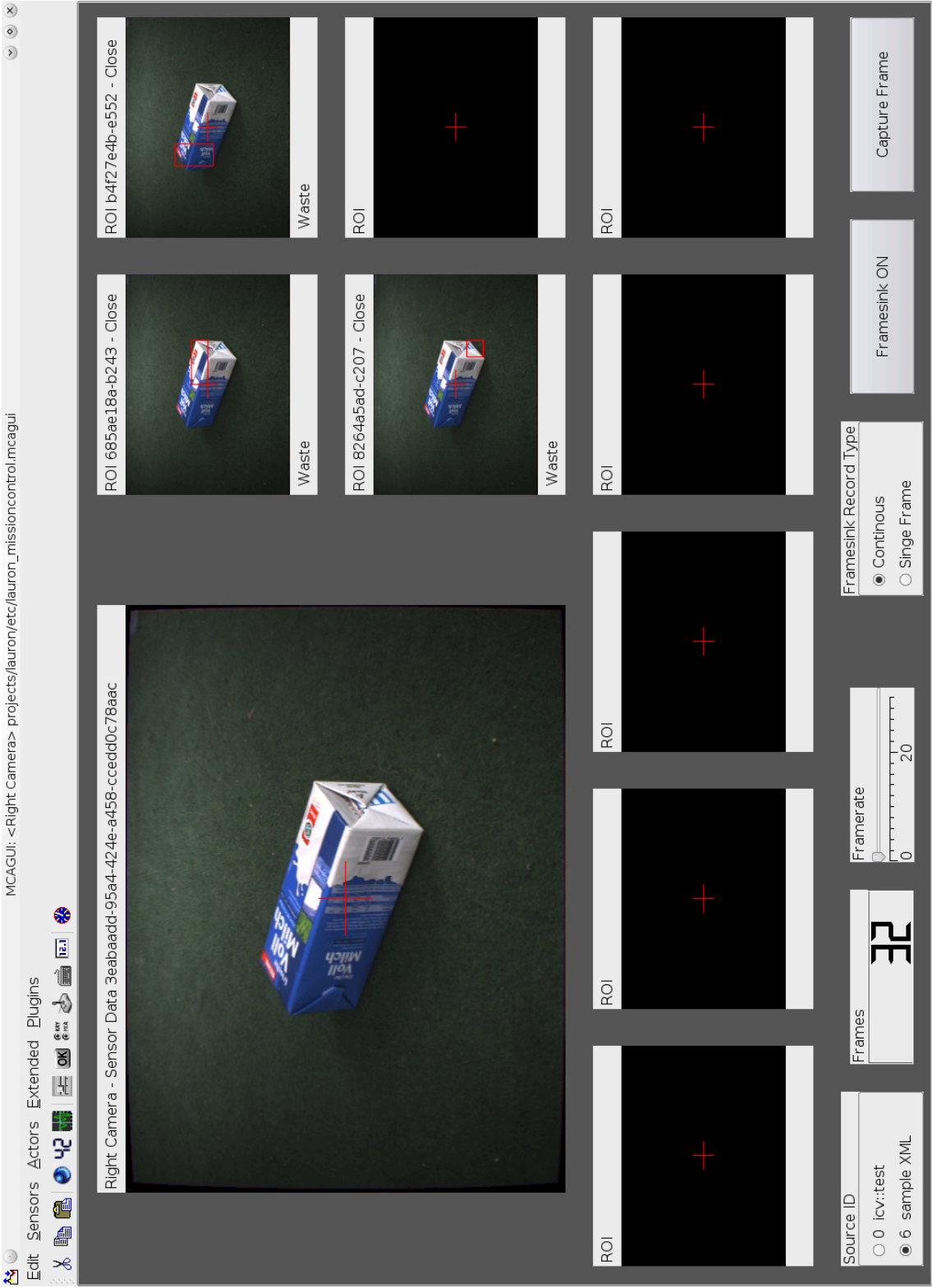


Abb. 7.19.: Die Bedienkomponente zur Überwachung der segmentierten Bildbereiche sowie der zugehörigen Klassifikationsergebnisse.

werden entsprechende Einträge mit detaillierten Informationen sowie dem jeweiligen Verarbeitungsstatus angezeigt. Für die gefundenen interessierenden Entitäten werden die Wahrscheinlichkeiten der Zugehörigkeit zu den einzelnen Entitätsklassen, der entsprechende Konfidenzwert sowie die als nächstes zur weiteren Untersuchung vorgeschlagene Inspektionsaufgabe nebst zugehöriger Priorität dargestellt. Die aktuell untersuchte interessierende Entität wird entsprechend hervorgehoben.

**Wissensbasis** Die Komponente *Wissensbasis* (siehe Abbildung 7.21) erlaubt es dem menschlichen Operator zur Laufzeit Einblick in die Wissensbasis zu nehmen und entsprechende Anfragen zu stellen. Dabei kann er sowohl von vordefinierten Anfragen Gebrauch machen als auch mit Hilfe der SPARQL-Anfragesprache (siehe Abschnitt 2.2.2) beliebige Anfragen direkt formulieren.

#### 7.4.2. Wissensbasis

Die in Kapitel 4 vorgestellte Wissensbasis wurde mit Hilfe von *Protégé*<sup>16</sup> in *OWL DL* (siehe Abschnitt 2.2.2) umgesetzt. Die einzelnen Unterontologien wurden dabei in Form von separaten, wiederverwendbaren Dateien realisiert, die mittels Import-Direktiven zu der in Abbildung 4.1 dargestellten Struktur der Wissensbasis verknüpft wurden (siehe Abbildung 7.22).

Die Gesamtanzahl der modellierten Klassen, Relationen und Instanzen ist in Abbildung 7.23 angegeben. Eine Übersicht über die Verteilung der modellierten Klassen, Relationen und Instanzen auf die verschiedenen Schichten der Wissensbasis befindet sich in Tabelle 7.4.

Ontologie	Klassen	Relationen	Individuen
Basisontologie	47	44	5
Kernontologie	84	95	13
Domänenontologie	212	29	1641
<b>Insgesamt</b>	<b>343</b>	<b>168</b>	<b>1659</b>

Tab. 7.4.: Übersicht über die Anzahl der Klassen, Relationen und Individuen in den einzelnen Ontologien.

Neben den Klassen, Relationen und Instanzen enthält die Wissensbasis eine Regelbasis (siehe Abbildung 7.27) in Form von 63 SWRL-Regeln (siehe Abschnitt 2.2.2). Diese wird zur Bewertung der Konfidenz der fusionierten Datenauswertungsergebnisse sowie zur situationsabhängigen Auswahl und Priorisierung von Inspektionsaufgaben verwendet (siehe Abschnitt 6.3). Darüber hinaus kann optional auch eine regelbasierte Klassifikation der gefundenen interessierenden Entitäten anhand von beobachtbaren Merkmalen erfolgen (siehe Abbildung 7.28). Der

<sup>16</sup>Protégé: <http://protege.stanford.edu/>

MCAGUI: <inspection> projects/auron/etc/auron\_missioncontrol.mcagui

Edit Sensors Actors Extended Plugins

Sensor Data

No	Time Stamp	UUID	Status	Type	X [m]	Y [m]
00017	2010-07-10 13:14:31	02900933-z80c-4620-9901-f13394033479	Closed	Camera	0.303089	-0.109816
00018	2010-07-10 13:16:52	0b36a682-fa15-45d3-b308-f55668ee93479	Closed	Camera	0.303089	-0.109816
00019	2010-07-10 13:19:05	ecb47abd-8da6-4818-8213-c6ef6cf19078	Closed	Camera	0.303089	-0.109816
00020	2010-07-10 13:31:54	234a9014-486d-41e6-b3d9-0942dd5b5aba	Closed	Camera	0.303089	-0.109816

Regions Of Interest

No	Time Stamp	UUID	Status	Type	Camera
00046	2010-07-10 13:32:07	29f57cc1-11da-4fe0-a186-4d3b024c474f	Closed	Camera	234a9014-486d-41e6-b3d9-c
00047	2010-07-10 13:32:07	a0f2207a-5a25-4244-a3b8-83a6c6ee8672	Closed	Camera	234a9014-486d-41e6-b3d9-c
00048	2010-07-10 13:32:07	95336440-6875-4ba4-aa40-e01e28a6fc9d	Open	Camera	234a9014-486d-41e6-b3d9-c

Feature Vectors

No	Time Stamp	UUID	Status	Type	Region Of Interest
00071	2010-07-10 13:32:33	29050c0c-0217-4321-0001-abb012e33017	Closed	CameraWasteNonWaste	z9137cc1-11da-4fe0-a
00072	2010-07-10 13:32:55	0f86f8b5-1368-41fe-8c8a-7f917005b3f1	Closed	CameraWasteNonWaste	a0f2207a-5a25-4244-e
00073	2010-07-10 13:33:16	f30dd6dd-e443-48a4-b822-25ae2bfc6b54	Open	CameraWasteNonWaste	95336440-6875-4ba4-
00074	2010-07-10 13:33:43	6a7d4179-1a83-443c-8e90-1be96abc4118	New	CameraWasteType	95336440-6875-4ba4-

Classification Results

No	Time Stamp	UUID	Status	Type	Feature Vector
00071	2010-07-10 13:32:30	1cc5001e-104e-497d-9540-f073c0000124	Closed	CameraWasteNonWaste	z9050c0c-0217-4321-0
00072	2010-07-10 13:32:58	547b93de-9ef1-40de-a4af-f4812a794ccf	Closed	CameraWasteNonWaste	0f86f8b5-1368-41fe-8
00073	2010-07-10 13:33:19	545004ee-6972-42cd-93c0-95391a272688	Open	CameraWasteNonWaste	f30dd6dd-e443-48a4-b
00074	2010-07-10 13:33:46	a64b19a3-e20f-4765-a05d-16703ba82c99	New	CameraWasteType	6a7d4179-1a83-443c-f

Entities Of Interest

No	Time Stamp	UUID	Status	Classification Result	Predecessor
00071	2010-07-10 13:32:13	e9c02370-e596-4b30-az93-63131c000e20	Closed	00c4040e-ef0c-4092-003c-3c7e7000130e	70be0000
00071	2010-07-10 13:32:37	022be942-0acc-444c-8f10-f618785532b6	Closed	1cc3501e-184e-497a-934d-f67232dd6f24	
00072	2010-07-10 13:32:59	21f063a0-488d-42c5-ad19-e6bddd17d0953	Closed	547b93de-9ef1-40de-a4af-f4812a794ccf	
00073	2010-07-10 13:33:20	4c48b82c-0158-497e-87ab-c0ac1e4a8ee4	Current	545004ee-6972-42cd-93c0-95391a272688	

Abb. 7.20.: Die Bedienkomponente zur Überwachung der Inspektionsdatenauswertung.

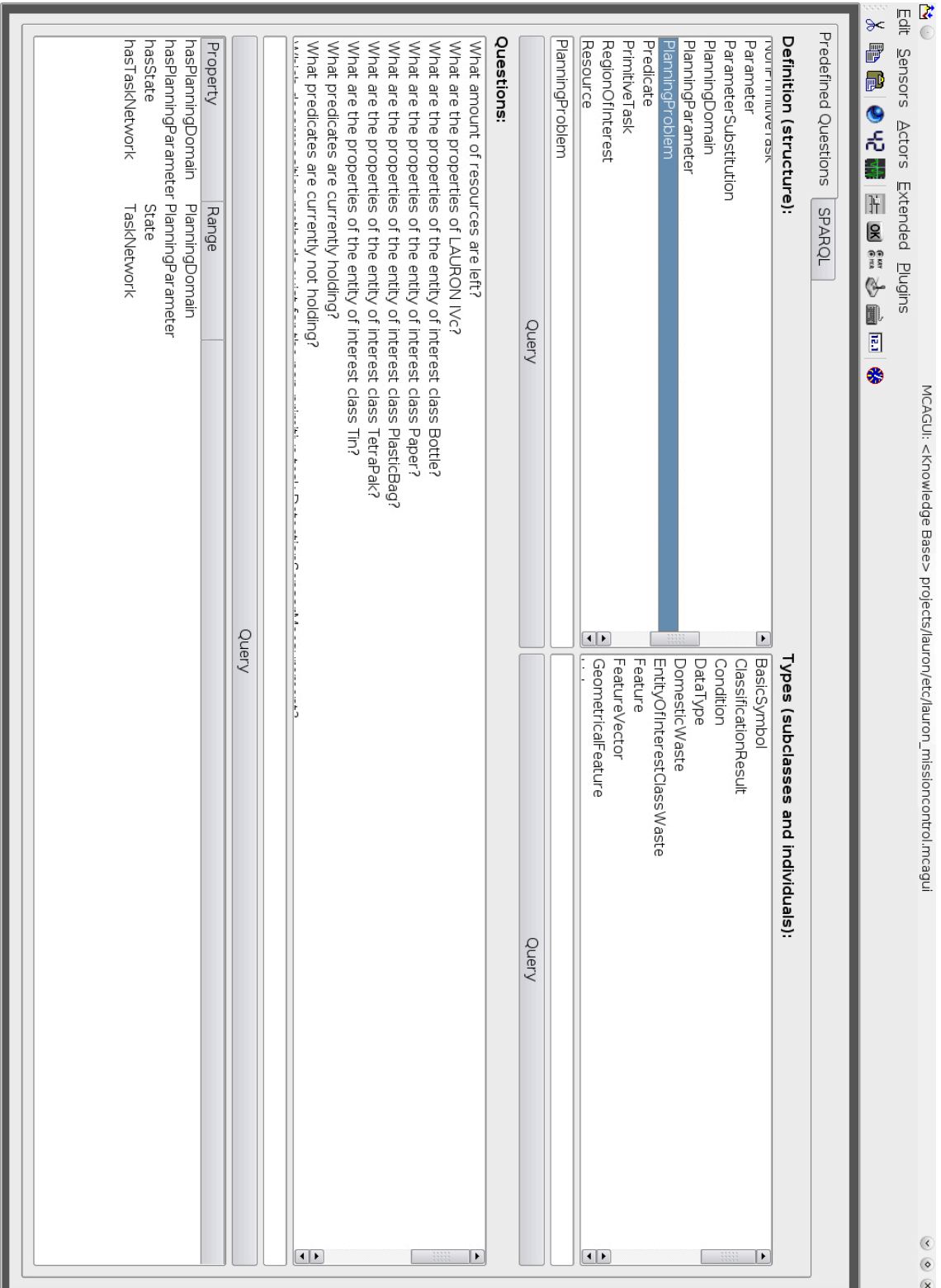


Abb. 7.21.: Die Bedienkomponente zum Stellen von Anfragen an die Wissensbasis.

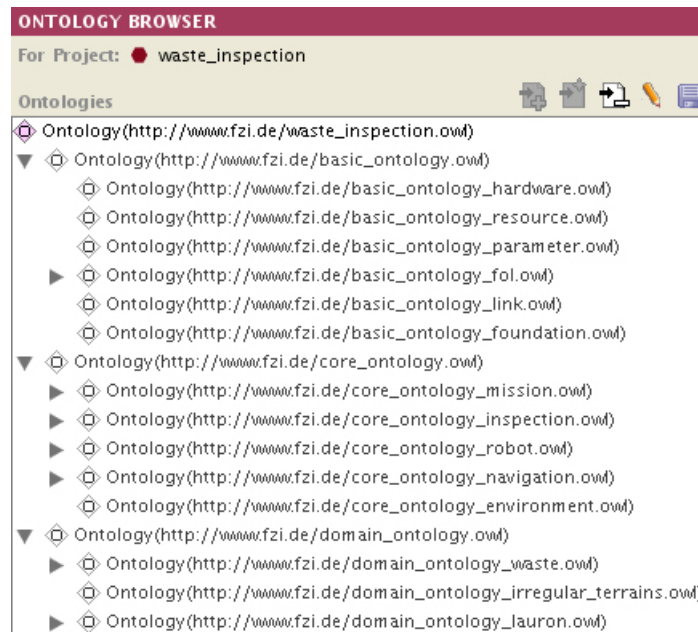


Abb. 7.22.: Der Aufbau der Wissensbasis in Form der verschiedenen in Protégé modellierten Ontologien.

Metrics	
Class count	343
Object property count	111
Data property count	57
Individual count	1659
DL expressivity	ALUHI(D)

Abb. 7.23.: Anzahl der modellierten Klassen, Relationen und Instanzen der Wissensbasis.

Zugriff auf die Wissensbasis erfolgt mit Hilfe von *KAON2* (siehe Abschnitt 2.2.2), welches über *JNI*<sup>17</sup> an die Steuerungsarchitektur angebunden ist.

### Eigenschaften und Fähigkeiten von LAURON IV

Die Roboterfähigkeiten sind in der Unterontologie *Laufmaschine LAURON IV* der Domänenontologie modelliert (siehe Abschnitt 4.3.3). Sie werden im Wesentlichen durch die vom Roboter ausführbaren elementaren und nichtelementaren Aufgaben charakterisiert, welche als Instanzen der entsprechenden Konzepte in der Kernontologie *Mission* (siehe Abschnitt 4.3.2) realisiert sind. Die elementaren, direkt vom Roboter ausführbaren Aufgaben lassen sich grob in die zwei Bereiche Navigation und Detektion gliedern. Der erste Bereich umfasst die das Planen und Ausführen von Pfaden in unstrukturiertem Gelände betreffenden Fähigkeiten (siehe Abbildung 7.24), wohingegen der zweite Bereich die zur Detektion und Klassifikation von Abfallobjekten benötigten Fähigkeiten beschreibt (siehe Abbildung 7.25). Komplexere Fähigkeiten, wie das systematische Absuchen von Regionen oder die Erkennung der Abfallart, sind als nichtelemen-

<sup>17</sup>Java Native Interface: <http://java.sun.com/javase/6/docs/technotes/guides/jni/>

tare Aufgaben modelliert, welche mit Hilfe von Zerlegungsmethoden auf die entsprechenden elementaren Aufgaben zurückgeführt werden. Neben den elementaren und nichtelementaren Aufgaben sind in der Unterontologie *Laufmaschine LAURON IV* darüber hinaus die Prädikate zur Beschreibung des Roboterzustands und die dem Roboter zur Verfügung stehenden Ressourcen modelliert.

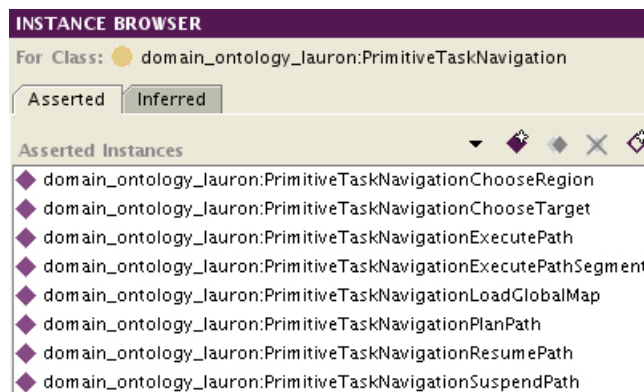


Abb. 7.24.: Elementare Aufgaben zur Beschreibung der Navigationsfähigkeiten.

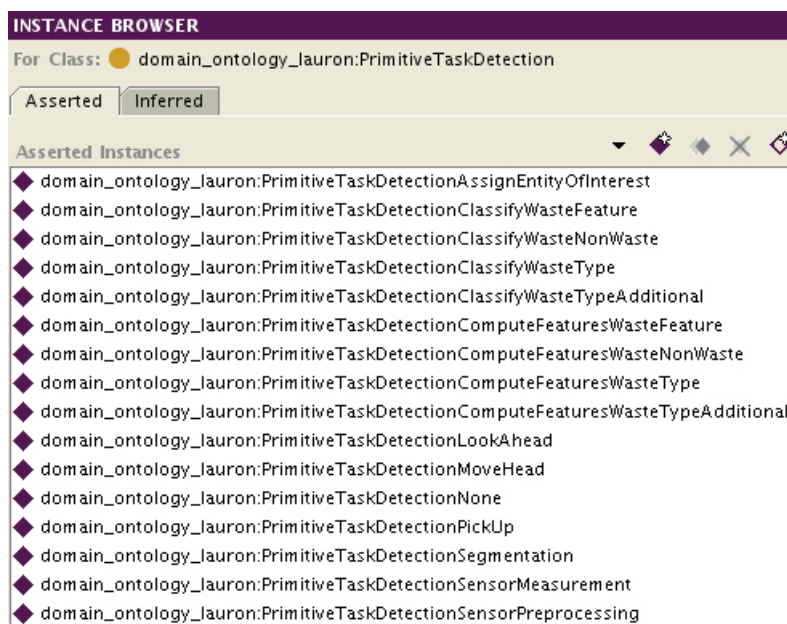


Abb. 7.25.: Elementare Aufgaben zur Beschreibung der Detektionsfähigkeiten.

## Inspektionswissen für die Erkennung von Abfall

Das Inspektionswissen ist in der Unterontologie *Detektion und Klassifikation von Abfall* der Domänenontologie modelliert (siehe Abschnitt 4.3.3). Es gliedert sich zum einen in die Beschreibung der verschiedenen Abfallarten und ihrer charakteristischen Eigenschaften und zum

anderen in Vorgehensweisen zur Beurteilung und Untersuchung von gefundenen interessierenden Entitäten in Form einer Regelbasis.

Als Entitätsklassen für Abfallobjekte werden im Rahmen dieser Arbeit exemplarisch die fünf Klassen *Flasche*, *Papier*, *Plastiktüte*, *Tetrapak* und *Dose* unterschieden. Die Merkmale zur Beschreibung der Abfallobjekte gliedern sich in die Kategorien *Geometrische Eigenschaft*, *Optische Eigenschaft* und *Physikalische Eigenschaft* (siehe Abbildung 7.26). Zu den geometrischen Eigenschaften zählen Form und Größe. Die optischen Eigenschaften werden durch Farbe, Oberflächenglanz und Transparenz beschrieben. Die physikalischen Eigenschaften umfassen die elektrische Leitfähigkeit, die Elastizität und die Magnetisierbarkeit.

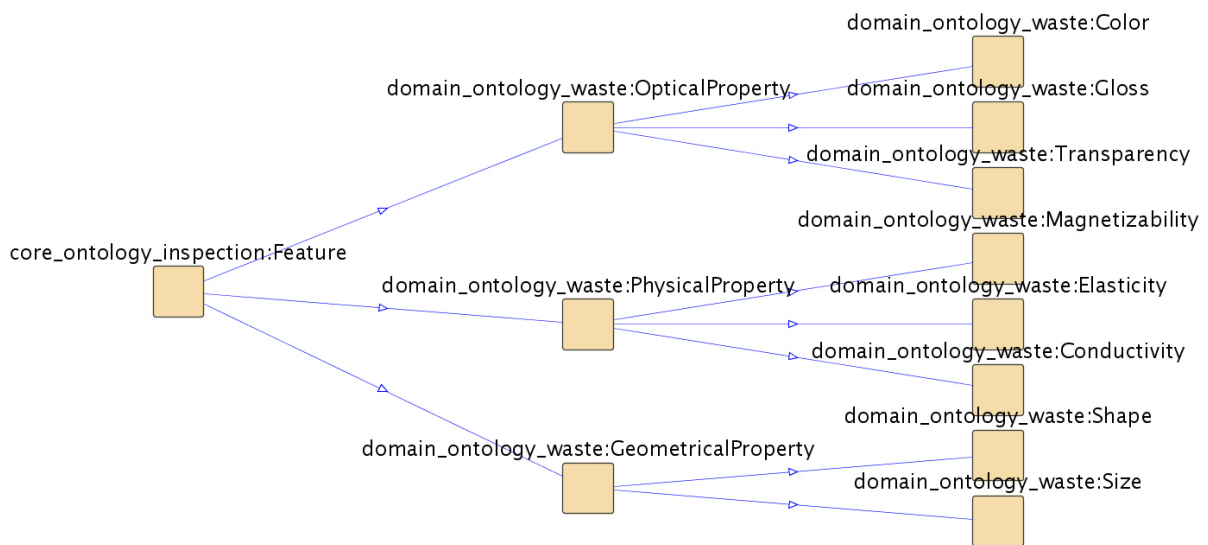


Abb. 7.26.: Die verschiedenen Merkmalsklassen zur Charakterisierung von Abfallobjekten.

Die Regelbasis (siehe Abbildung 7.27) wurde mit Hilfe von Protégé in der *Semantic Web Rule Language* (SWRL) (siehe Abschnitt 2.2.2) umgesetzt. Sie beinhaltet im Wesentlichen Regeln zur Beurteilung von gefundenen interessierenden Entitäten und zur situationsabhängigen Auswahl und Priorisierung von Inspektionsaufgaben für die weitere Untersuchung. Darüber hinaus ermöglicht sie es, optional eine regelbasierte Klassifikation der interessierenden Entitäten anhand von beobachtbaren Merkmalen durchzuführen (siehe Abbildung 7.28).

### 7.4.3. Erzeugung und Ausführung von Inspektionsplänen

Das in Kapitel 5 beschriebene Planungsverfahren wurde in Java implementiert (siehe [58]) und mittels JNI an die mit Hilfe des Softwarerahmenwerks MCA2 realisierte Steuerungsarchitektur angebunden. Hierdurch ist insbesondere ein schneller und direkter Zugriff auf die Wissensbasis mit Hilfe von KAON2 gewährleistet, welches ebenfalls in Java implementiert ist. Ein Beispiel für einen generierten Flexiblen Hierarchischen Plan in XML-Serialisierung ist in Abbildung





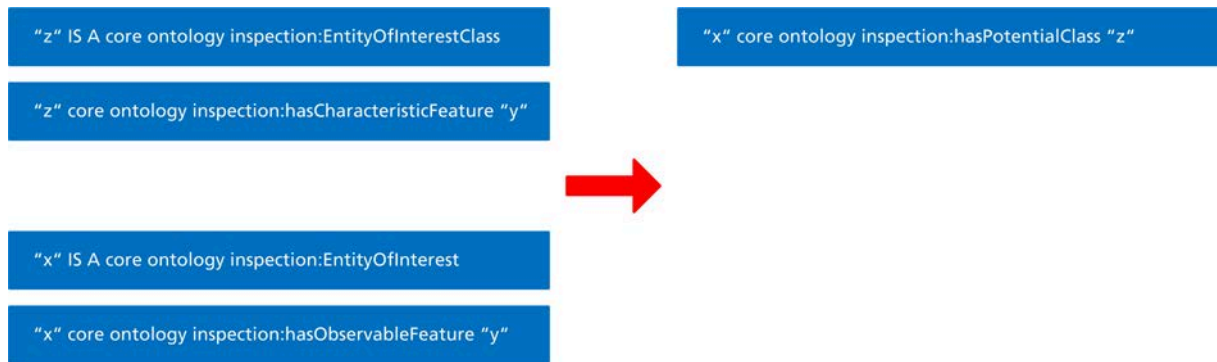


Abb. 7.28.: SWRL-Regel zur Bestimmung der potentiellen Klassenzugehörigkeit von interessierenden Entitäten anhand von beobachtbaren Merkmalen.

7.29 dargestellt. Die in Abschnitt 5.2 vorgestellten Elemente der Flexiblen Hierarchischen Pläne werden dabei jeweils durch entsprechende XML-Tags abgebildet.

#### 7.4.4. Aktive Untersuchung interessierender Entitäten

Die aktive Untersuchung von interessierenden Entitäten setzt sich wie in Kapitel 6 beschrieben aus den folgenden drei Schritten zusammen:

**Fusion der Datenauswertungsergebnisse** Bestimmung der Wahrscheinlichkeiten für das tatsächliche Vorliegen der einzelnen Entitätsklassen anhand der bisher verfügbaren Datenauswertungsergebnisse (siehe Abschnitt 6.3.1).

**Bewertung der fusionierten Datenauswertungsergebnisse** Bestimmung der Konfidenz der fusionierten Datenauswertungsergebnisse und Entscheidung darüber, ob die interessierende Entität weiter untersucht werden soll (siehe Abschnitt 6.3.2).

**Auswahl und Priorisierung von Inspektionsaufgaben** Auswahl der am besten geeigneten Inspektionsaufgabe zur weiteren Untersuchung der interessierenden Entität sowie Festlegung der zugehörigen Priorität (siehe Abschnitt 6.3.3).

In diesem Abschnitt wird auf die konkrete Umsetzung der einzelnen Schritte für die Detektion und Klassifikation von Abfall mit LAURON IV eingegangen.

Zur Fusion der Datenauswertungsergebnisse wurde das in Kapitel 6 beschriebene generische Bayes'sche Netzwerk mit Hilfe von *GeNIe*<sup>18</sup> für das gewählte Inspektionsszenario umgesetzt [89]. Das realisierte Bayes'sche Netzwerk besteht aus insgesamt dreizehn Knoten (siehe Abbildung 7.30). Der Knoten *WasteNonWaste* gibt die Wahrscheinlichkeit an, dass es sich tatsächlich um Abfall bzw. nicht um Abfall handelt, gegeben das Ergebnis der Klassifikation Abfall / Kein Abfall, welches durch den Knoten *RecognizeWasteNonWaste* repräsentiert wird. Der Knoten *EntityOfInterestClass* beschreibt die Wahrscheinlichkeit, dass es sich tatsächlich um die

<sup>18</sup>GeNIe: [http://genie.sis.pitt.edu/wiki/GeNIe\\_Documentation](http://genie.sis.pitt.edu/wiki/GeNIe_Documentation)

## 7. Umsetzung der semantischen Missionssteuerung auf LAURON IV

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <FHHP ID="1" Name="TopLevelPlan" Optional="false" Priority="PriorityNormal" RatingFunction="1.0" Type="And">
3   <Parameter />
4   <Condition Type="Pre" />
5   <Condition Type="In" />
6   <Condition Type="Post" />
7   <ResourceUsage />
8   <Subplans>
9     <FHHP ID="0" Name="NonPrimitiveTaskDetectionRecognizeWasteTypeIn" Optional="false" Priority="PriorityNormal" Type="Or">
10      <Parameter>
11        <ParameterItem Name="RegionOfInterestCamera_311fcb12-ed17-4d5c-9643-6ea2b6129e" Position="1" />
12      </Parameter>
13      <Condition Type="Pre" />
14      <Condition Type="In" />
15      <Condition Type="Post" />
16      <ResourceUsage />
17      <Subplans>
18        <FHHP ID="1" Name="MethodDetectionRecognizeWasteTypeIn" Optional="false" Priority="PriorityNormal" RatingFunction="1.0" Type="And">
19          <Parameter>
20            <ParameterItem Name="RegionOfInterestCamera_311fcb12-ed17-4d5c-9643-6ea2b6129e" Position="1" />
21            <ParameterItem Name="Tin" Position="2" />
22          </Parameter>
23          <Condition Type="Pre">
24            <ConditionItem Name="SegmentedROI" Negated="false">
25              <Parameter>
26                <ParameterItem Name="RegionOfInterestCamera_311fcb12-ed17-4d5c-9643-6ea2b6129e" Position="1" />
27              </Parameter>
28              </ConditionItem>
29            </Condition>
30            <Condition Type="In" />
31            <Condition Type="Post" />
32            <ResourceUsage />
33            <Subplans>
34              <FHHP Availability="1.0" ID="16" Name="PrimitiveTaskDetectionAssignEntityOfInterest" Optional="false" Priority="PriorityNormal" Type="Primitive">
35                <Parameter>
36                  <FHHP ID="2" Name="NonPrimitiveTaskDetectionComputeFeatureWasteTypeAdditional" Optional="false" Priority="PriorityNormal" Type="Or">
37                    <FHHP ID="9" Name="NonPrimitiveTaskDetectionClassifyWasteTypeAdditional" Optional="false" Priority="PriorityNormal" Type="Or">
38                      <Subplans>
39                        <OrderingConstraints>
40                          <TemporalRelation First="2" Second="9" Type="Meets" />
41                          <TemporalRelation First="16" Type="Meets" />
42                          <TemporalRelation First="2" Second="16" Type="Before" />
43                        </OrderingConstraints>
44                        <SummaryCondition Type="Pre">
45                          <SummaryCondition Type="In">
46                            <SummaryResourceUsage>
47                              </FHHP>
48                            </SummaryResourceUsage>
49                        </Subplans>
50                        <OrderingConstraints />
51                        <SummaryCondition Type="Pre">
52                          <SummaryCondition Type="In">
53                            <SummaryResourceUsage>
54                              </FHHP>
55                            </SummaryResourceUsage>
56                        </Subplans>
57                      </FHHP>
58                    </SummaryCondition Type="Post">
59                      <SummaryResourceUsage>
60                        </FHHP>
61                      </SummaryResourceUsage>
62                    </SummaryCondition Type="Post">
63                      </SummaryResourceUsage>
64                    </SummaryCondition Type="Post">
65                      </SummaryResourceUsage>
66                    </SummaryCondition Type="Post">
67                      </SummaryResourceUsage>
68                    </SummaryCondition Type="Post">
69                      </SummaryResourceUsage>
70                    </SummaryCondition Type="Post">
71                      </SummaryResourceUsage>
72                    </SummaryCondition Type="Post">
73                      </SummaryResourceUsage>
74                    </SummaryCondition Type="Post">
75                      </SummaryResourceUsage>
76                    </SummaryCondition Type="Post">
77                      </SummaryResourceUsage>
78                    </SummaryCondition Type="Post">
79                      </SummaryResourceUsage>
80                    </SummaryCondition Type="Post">
81                      </SummaryResourceUsage>
82                    </SummaryCondition Type="Post">
83                      </SummaryResourceUsage>
84                    </SummaryCondition Type="Post">
85                      </SummaryResourceUsage>
86                    </SummaryCondition Type="Post">
87                      </SummaryResourceUsage>
88                    </SummaryCondition Type="Post">
89                      </SummaryResourceUsage>
90                    </SummaryCondition Type="Post">
91                      </SummaryResourceUsage>
92                    </SummaryCondition Type="Post">
93                      </SummaryResourceUsage>
94                    </SummaryCondition Type="Post">
95                      </SummaryResourceUsage>
96                    </SummaryCondition Type="Post">
97                      </SummaryResourceUsage>
98                    </SummaryCondition Type="Post">
99                      </SummaryResourceUsage>
100                    </SummaryCondition Type="Post">
101                      </SummaryResourceUsage>
102                    </SummaryCondition Type="Post">
103                      </SummaryResourceUsage>
104                    </SummaryCondition Type="Post">
105                      </SummaryResourceUsage>
106                    </SummaryCondition Type="Post">
107                      </SummaryResourceUsage>
108                    </SummaryCondition Type="Post">
109                      </SummaryResourceUsage>
110                    </SummaryCondition Type="Post">
111                      </SummaryResourceUsage>
112                    </SummaryCondition Type="Post">
113                      </SummaryResourceUsage>
114                    </SummaryCondition Type="Post">
115                      </SummaryResourceUsage>
116                    </SummaryCondition Type="Post">
117                      </SummaryResourceUsage>
118                </SummaryCondition Type="Post">
119              </SummaryCondition Type="Post">
120            </SummaryCondition Type="Post">
121          </SummaryCondition Type="Post">
122        </FHHP ID="1">
123      </Subplans>
124    </FHHP ID="0">
125  </Subplans>
126 </FHHP ID="1">
127 </FHHP>

```

Abb. 7.29.: Fragment eines in XML serialisierten Flexiblen Hierarchischen Plans.

jeweilige Entitätsklasse handelt, gegeben die Ergebnisse der bisher durchgeführten Inspektionsaufgaben. Diese werden durch die Knoten *RecognizeWasteType*, *RecognizeWasteTypeBottle*, *RecognizeWasteTypePaper*, *RecognizeWasteTypePlasticBag*, *RecognizeWasteTypeTetraPak* und *RecognizeWasteTypeTin* repräsentiert. Dabei beschreibt der Knoten *RecognizeWasteType* das Ergebnis der Klassifikation der Abfallart und die anderen Knoten geben das Ergebnis der Absicherung der jeweiligen Abfallart an. Die Ergebnisse dieser Inspektionsaufgaben werden jeweils von der Messgenauigkeit (Knoten *MeasurementAccuracy*) beeinflusst, welche ihrerseits von der Kameraauflösung (Knoten *CameraResolution*) und der tatsächlichen Entfernung (Knoten *Distance*) abhängig ist. Da die tatsächliche Entfernung nicht direkt beobachtbar ist, wird sie anhand des Ergebnisses der Distanzberechnung (Knoten *DistanceComputation*) geschätzt. Für den Zugriff auf das Bayes'sche Netzwerk und die Berechnung der bedingten Wahrscheinlichkeiten wird die in C++ implementierte Bibliothek *SMILE*<sup>19</sup> verwendet.

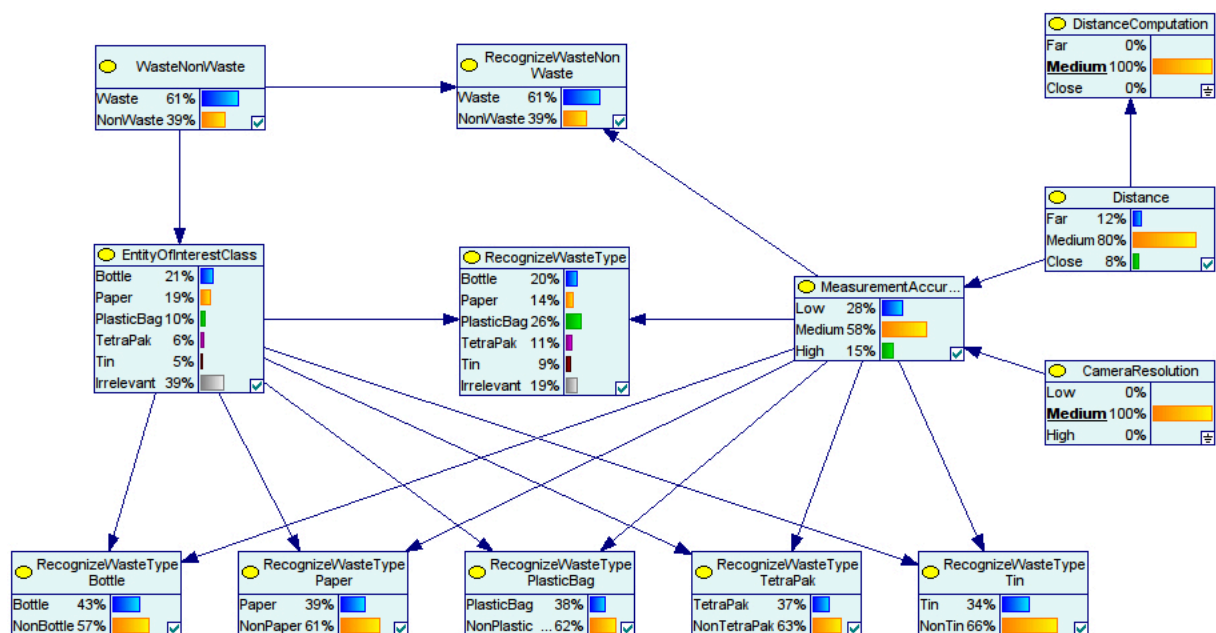


Abb. 7.30.: Bayes'sches Netzwerk zur temporalen Fusion der Datenauswertungsergebnisse.

Die Bewertung der fusionierten Datenauswertungsergebnisse erfolgt anhand der ermittelten Wahrscheinlichkeitsverteilung für die Entitätsklassen. Zur Bestimmung der Konfidenz wird ein entropiebasiertes Konfidenzmaß für diese Wahrscheinlichkeitsverteilung berechnet (siehe Gleichung 6.9). Liegt die Konfidenz über einem anhand von Regeln festgelegten Schwellenwert, wird das Untersuchungsergebnis akzeptiert. Ist dies nicht der Fall, wird mit Hilfe der Regelbasis des semantischen Inspektionsmodells die in Abhängigkeit von der jeweiligen Situation am besten geeignete Inspektionsaufgabe zur weiteren Untersuchung bestimmt und entsprechend priorisiert.

<sup>19</sup>SMILE: [http://genie.sis.pitt.edu/wiki/SMILE\\_Documentation](http://genie.sis.pitt.edu/wiki/SMILE_Documentation)

Die Auswahl der nächsten Inspektionsaufgabe zur weiteren Untersuchung einer interessierenden Entität geschieht im Wesentlichen wie folgt (siehe auch Abbildung 7.31): Zunächst wird für die jeweils interessierende Bildregion eine Klassifikation *Abfall / Kein Abfall* durchgeführt. Besitzt die Entitätsklasse *Abfall* die größte Wahrscheinlichkeit und liegt eine ausreichende Messgenauigkeit vor, wird anschließend eine Bestimmung der Abfallart durchgeführt. Liegt jedoch nur eine geringe Messgenauigkeit vor, so wird direkt nach der Klassifikation *Abfall / Kein Abfall* versucht, die Messgenauigkeit zu erhöhen. Dies kann zum Beispiel durch eine Annäherung an die interessierende Entität und eine erneute Sensormessung erfolgen. Nach der Bestimmung der Abfallart wird anhand der Konfidenz entschieden, ob das Klassifikationsergebnis akzeptiert wird oder ob eine weitere Untersuchung durchgeführt werden soll. Diese besteht in der Regel zunächst in der Absicherung der Abfallart mit Hilfe einer Ein-Klassen-SVM. Bestätigt diese zusätzliche Überprüfung die Abfallart, wird das Ergebnis akzeptiert. Weist sie die Abfallart hingegen zurück, wird ebenfalls versucht, die Messgenauigkeit zu erhöhen.

### 7.5. Zusammenfassung

In diesem Kapitel wurde zunächst ein kurzer Überblick über die Eigenschaften und Fähigkeiten der sechsbeinigen Laufmaschine LAURON IV gegeben, für die das im Rahmen dieser Arbeit entwickelte Konzept der semantischen Missionssteuerung exemplarisch umgesetzt wurde. Daran anknüpfend wurde auf die speziell im Rahmen dieser Arbeit entwickelten Verfahren zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall in unstrukturiertem Gelände eingegangen. Die Hauptaufgabe der globalen Navigation besteht in der Planung von Pfaden zu vorgegebenen Zielpunkten und Regionen sowie zum systematischen Absuchen von Regionen. Für die Pfadplanung zu Zielpunkten und Regionen wurde ein auf *Rapidly-Exploring Random Trees* (RRT) basierendes Verfahren verwendet. Das systematische Absuchen von Regionen wurde mit Hilfe eines numerischen Potentialfeldverfahrens realisiert. Die Detektion von Abfall erfolgt mittels eines aufmerksamkeitsbasierten Verfahrens, welches charakteristische Merkmale (Farbe, Intensität und Orientierung) aus dem ursprünglichen Farbbild extrahiert und zur Berechnung so genannter Aufmerksamkeitskarten (engl. *saliency map*) nutzt. Anschließend wird eine auf Farbtexturmerkmalen basierende Klassifikation der segmentierten Bildbereiche mit Support-Vektor-Maschinen durchgeführt. Als Texturmerkmale kommen dabei Farbverbundhistogramme (engl. *Color Cooccurrence Histogram* (CCH)), Lokale binäre Muster (engl. *Local Binary Patterns* (LBP)) sowie lokal invariante SURF-Deskriptoren (engl. *Speeded-Up Robust Features* (SURF)) zum Einsatz. Anschließend wurde die Umsetzung der vier Kernkomponenten der semantischen Missionssteuerung erläutert. Die Steuerungsarchitektur wurde mit Hilfe des Softwarerahmenwerks MCA2 realisiert. Die Wissensbasis wurde mit Protégé in OWL DL umgesetzt. In Bezug auf die Wissensbasis wurde insbesondere auf die modellierten Roboterfähigkeiten sowie auf das modellierte Inspektionswissen für die Detektion und Klassifikation von

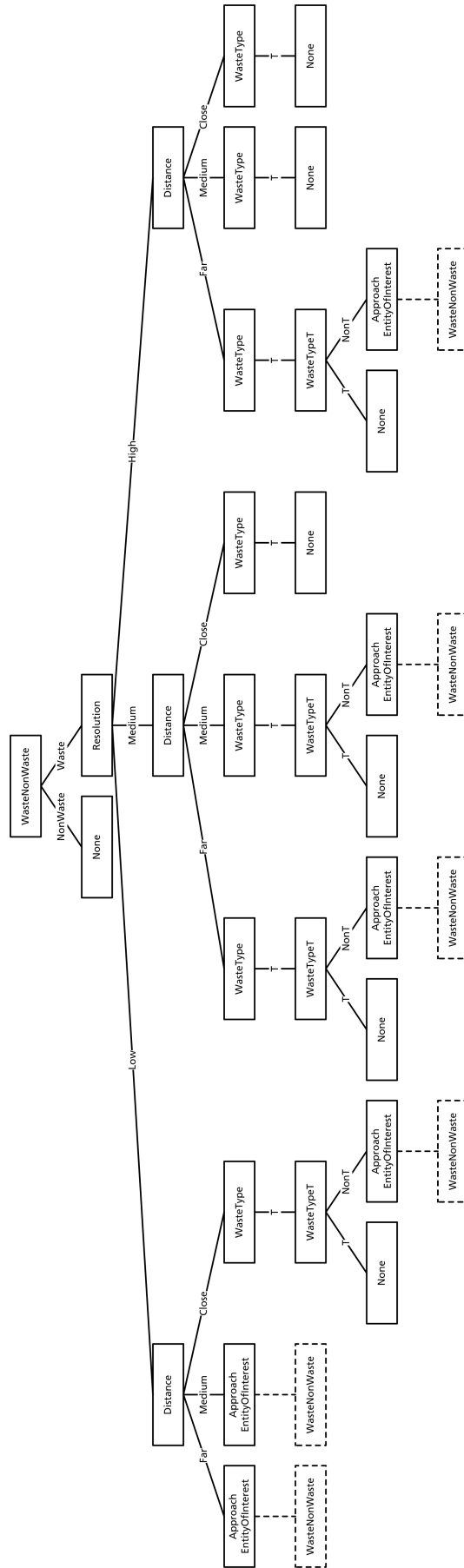


Abb. 7.31.: Entscheidungsbaum zur aktiven Untersuchung interessierender Entitäten.

Abfall eingegangen. Abschließend wurde die Umsetzung des Planers vorgestellt und die aktive Untersuchung von Abfallobjekten beschrieben.

## 8. Experimentelle Evaluierung

In diesem Kapitel wird die experimentelle Evaluierung des entwickelten Konzepts der semantischen Missionssteuerung für autonome Inspektionsroboter erläutert. Dazu wird zunächst auf die Evaluationskriterien eingegangen. Daran anschließend wird der so genannte Simulationsmodus vorgestellt, der eine wesentliche Grundlage für die Evaluierung der Kernkomponenten der semantischen Missionssteuerung bildet. Die eigentliche Evaluierung gliedert sich in vier Teilbereiche.

Im ersten Bereich werden die mit Hilfe des Simulationsmodus durchgeführten Untersuchungen zur Funktionsfähigkeit und zum Verhalten der semantischen Missionssteuerung in relevanten Inspektionssituationen vorgestellt. Im Fokus stehen dabei insbesondere die Funktionsfähigkeit des Regelkreises der autonomen Inspektion, die Konsistenz und Korrektheit des in der Wissensbasis spezifizierten Wissens sowie die korrekte Erzeugung und Ausführung von Inspektionsplänen. Weitere zentrale Punkte umfassen die korrekte Fusion und Bewertung der Datenauswertungsergebnisse sowie die korrekte situationsabhängige Auswahl und Priorisierung von Inspektionsaufgaben zur weiteren Untersuchung interessierender Entitäten.

Im zweiten Bereich wird die Evaluierung der erweiterten Fähigkeiten von LAURON IV vorgestellt. Dabei wird insbesondere auf die Evaluierung der entwickelten Verfahren zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall eingegangen, welche die wesentliche Grundlage für die tatsächliche Ausführung von Inspektionsmissionen mit LAURON IV im Rahmen des gewählten Szenarios bilden.

Daran anschließend wird im dritten Bereich die Evaluierung des Gesamtsystems beschrieben. Dies umfasst insbesondere das aktive Absuchen von Regionen mit LAURON IV. Hierzu wurden sowohl unter kontrollierten Laborbedingungen als auch in unstrukturiertem Gelände Experimente durchgeführt.

Im vierten Bereich wird am Beispiel der Portierung der semantischen Missionssteuerung auf den mehrsegmentigen Inspektionsroboter KAIRO II die Übertragbarkeit des im Rahmen dieser Arbeit entwickelten Konzepts auf andere Roboterplattformen demonstriert.

Abschließend erfolgt eine eingehende Diskussion und Bewertung der Ergebnisse.

### 8.1. Evaluationskriterien

Die Evaluierung der im Rahmen dieser Arbeit entwickelten semantischen Missionssteuerung für autonome Inspektionsroboter sollte in verschiedenen, aufeinander aufbauenden Phasen er-

folgen (siehe Abbildung 8.1).

In der ersten Phase sollten zunächst die in den vier Teilbereichen der Arbeit entwickelten Kernkomponenten im Hinblick auf ihre Funktionsfähigkeit und ihr Verhalten in relevanten Inspektionssituationen untersucht werden. Neben der Feststellung der prinzipiellen Eignung der entwickelten Konzepte und Verfahren für die Inspektion mit autonomen Servicerobotern sind für die einzelnen Kernkomponenten jeweils weitere spezifische Eigenschaften zu zeigen.

Für die Steuerungsarchitektur umfasst dies etwa die Überprüfung des Zusammenwirkens der einzelnen Komponenten und der prinzipiellen Abläufe innerhalb der Steuerungsarchitektur.

Für die Wissensbasis stehen die Konsistenz und die Korrektheit des spezifizierten Wissens im Vordergrund. So sind etwa die syntaktischen und semantischen Eigenschaften der modellierten Konzepte sowie ihre strukturellen Zusammenhänge geeignet zu validieren. Darüber hinaus ist die Modellierung der einzelnen Konzepte auf Vollständigkeit zu prüfen.

Für die Erzeugung und Ausführung von Inspektionsplänen ist die Validierung der Verfahren zur Bestimmung von zusammengefassten Planinformationen von besonderer Bedeutung, da sie die wesentliche Grundlage für die hierarchische Koordination und Verfeinerung bilden. Zur Evaluierung der Planerstellung und -ausführung sollten möglichst vielfältige Planungsprobleme zur Anwendung kommen. Darüber hinaus sollte der Umgang mit Ausführungsfehlern und Ausnahmesituationen gezielt überprüft werden.

Für die aktive Untersuchung interessierender Entitäten ist neben der korrekten Fusion und Bewertung der Datenauswertungsergebnisse insbesondere die situationsabhängige Auswahl und Priorisierung von Inspektionsaufgaben zur weiteren Untersuchung von interessierenden Entitäten wichtig. Zu diesem Zweck sollte eine ausreichende Anzahl von relevanten Inspektionssituationen systematisch erzeugt und zur Evaluierung der jeweiligen Verfahren verwendet werden.

In der zweiten Phase der Evaluierung sollten basierend auf den Kernkomponenten die im Rahmen dieser Arbeit entwickelten erweiterten Fähigkeiten von LAURON IV untersucht werden. Dies umfasst im Wesentlichen die Verfahren zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall. Neben der reinen Funktionsfähigkeit sind auch hier jeweils weitere Eigenschaften zu zeigen.

Die Verfahren zur Planung von Pfaden zu vorgegebenen Zielpunkten und Regionen sowie zum systematischen Absuchen von Regionen sollten anhand von standardisierten Benchmarks evaluiert werden. Für die auf RRTs basierende Pfadplanung zu vorgegebenen Zielpunkten und Regionen sind die Anzahl der benötigten Explorationsschritte sowie die Laufzeiten zum Aufbau der RRTs und zur anschließenden Pfadglättung von Interesse. Für die Pfadplanung zum systematischen Absuchen von Regionen bieten sich hingegen die Anzahl der beim Ablaufen des Pfades notwendigen Drehungen sowie die Anzahl und die Länge der mehrfach abzulaufenden Pfadsegmente als Evaluationskriterien an. Darüber hinaus sollte auch das Zusammenwirken der beiden Pfadplanungsbereiche intensiv untersucht werden. Hierfür sollten entsprechende Navi-



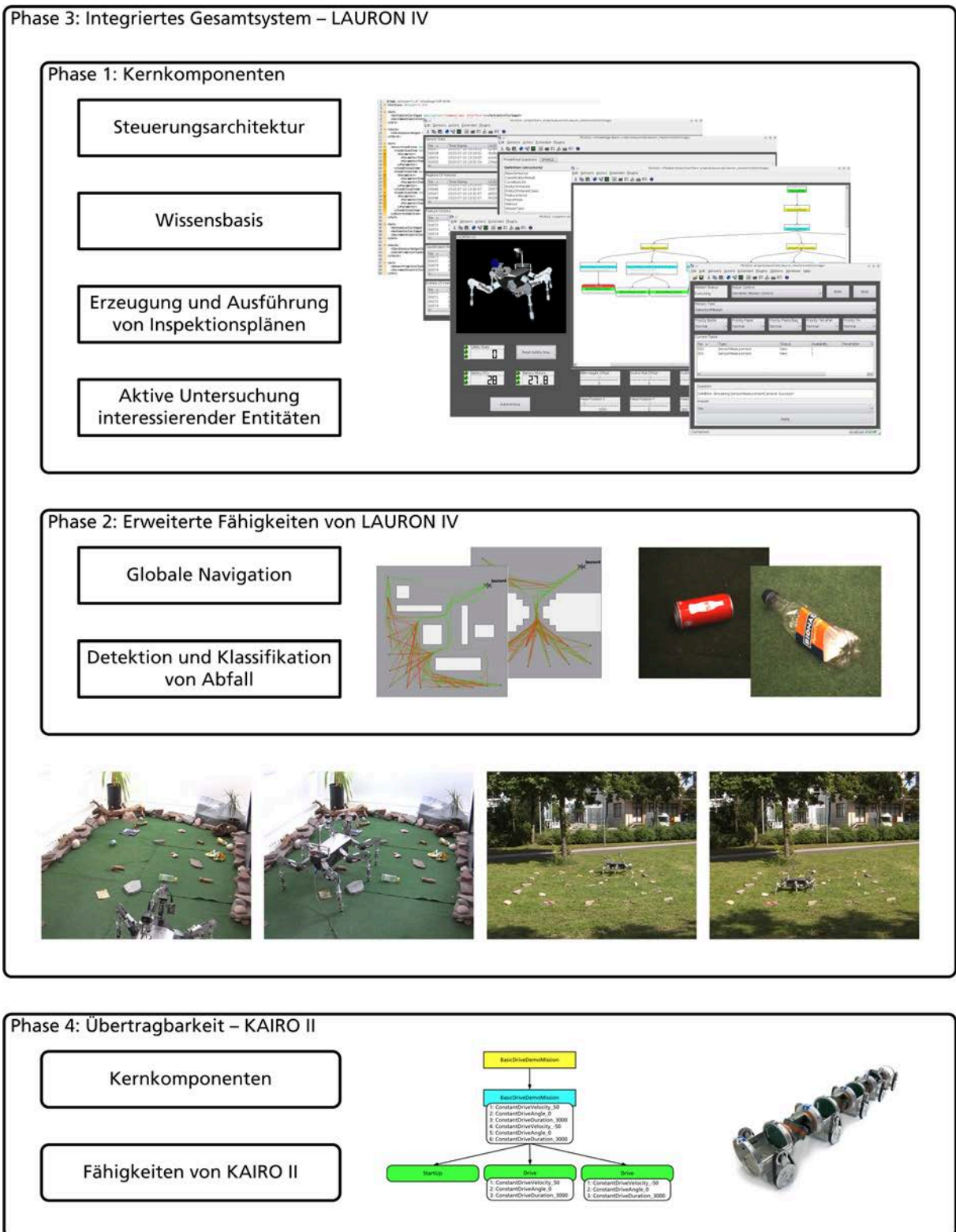


Abb. 8.1.: Die verschiedenen Phasen zur Evaluierung der semantischen Missionssteuerung.

gationsmissionen mit dem System absolviert werden.

Die Verfahren zur Detektion und Klassifikation von Abfall sollten mit Hilfe der in diesem Bereich gängigen Evaluationskriterien untersucht und bewertet werden. Für die Detektion von interessierenden Regionen ist insbesondere die Anzahl der nicht detektierten Abfallobjekte von Interesse. Die Klassifikation der interessierenden Regionen lässt sich nachfolgend anhand von Konfusionsmatrizen und verschiedenen darauf basierenden Kennzahlen beurteilen. Auch hier sollte anschließend das Zusammenwirken der beiden Bereiche mit Hilfe von geeigneten Detektionsmissionen analysiert werden.

In der dritten Phase sollte schließlich das integrierte Gesamtsystem, bestehend aus den Kernkomponenten der semantischen Missionssteuerung sowie den verschiedenen Fähigkeiten von LAURON IV, evaluiert werden. Hierzu sollte ein repräsentatives Inspektionsszenario gewählt werden, mit dessen Hilfe die aktive Untersuchung interessierender Entitäten und das Verhalten des Gesamtsystems in unterschiedlichen Situationen gezielt untersucht werden kann.

In der vierten Phase sollte abschließend die Übertragbarkeit des entwickelten Ansatzes auf andere Inspektionsroboter und andere Inspektionsdomänen analysiert werden, um zum einen die Allgemeingültigkeit des Ansatzes zu belegen und zum anderen die Vorteile der ontologiebasierten Vorgehensweise im Hinblick auf die Wiederverwendbarkeit zu demonstrieren. Als Evaluationskriterien bieten sich in diesem Zusammenhang die benötigten Zeitaufwände für die Übertragung der Kernkomponenten der semantischen Missionssteuerung sowie für die exemplarische Realisierung von grundlegenden Missionen an.

### 8.2. Simulationsmodus

Die semantische Missionssteuerung erlaubt es, sowohl mit dem realen als auch mit dem simulierten Roboter zu arbeiten. Darüber hinaus ist es möglich, sowohl die realen Sensoren als auch aufgezeichnete Sensordaten zu verwenden und reale sowie simulierte Fähigkeiten zu kombinieren. Diese Möglichkeiten werden im Folgenden kurz erläutert.

Zur Simulation der sechsbeinigen Laufmaschine LAURON IV wird die Hardwareabstraktionsebene (*LauronHal*, siehe Abbildung 7.14) in einem speziellem Modus gestartet, welcher es ermöglicht, die Bewegungen des Roboters und die entsprechenden Werte der Encoder und Fußsensoren nachzubilden. Die Art der verwendeten Sensordaten wird über die Konfiguration der Sensorabstraktionsebene (*LauronSal*) vorgegeben. Hier können sowohl die aktuell von den realen Sensoren gelieferten Daten verwendet als auch aufgezeichnete Daten eingespielt werden. Wie bereits in Abschnitt 7.4.1 beschrieben, können die einzelnen Ausführungskomponenten zudem in einem interaktiven Modus ausgeführt werden, der es erlaubt, den Status und die Ergebnisse der elementaren Aufgaben direkt vom Benutzer abzufragen. Der interaktive Modus kann für jede Ausführungskomponente separat aktiviert bzw. deaktiviert werden. Hierdurch wurde das flexible Testen von einzelnen Fähigkeiten und die frühe Inbetriebnahme des

gesamten Systems ermöglicht.

Darüber hinaus wurde für die Evaluierung des Gesamtsystems ein weiterer MCA2-Part realisiert, der sich mit der semantischen Missionssteuerung verbindet und anstelle des menschlichen Operators Kommandos erteilt, Fragen beantwortet und Daten überwacht. Mit Hilfe dieser Testautomatisierungskomponente können Testbibliotheken automatisch abgearbeitet werden. Die einzelnen Testfälle sind dabei als XML-Dateien realisiert (siehe Abbildung 8.2). Innerhalb der Testfälle kann auch die Wissensbasis (zum Beispiel der in Form von Prädikaten beschriebene Weltzustand) gezielt manipuliert werden. Auf diese Weise lassen sich mit geringem Aufwand viele unterschiedliche Inspektionssituationen generieren, in denen das Systemverhalten gezielt untersucht werden kann.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <TestCase Version="1.0">
3
4  <Set>
5    <SetControllerInput Description="Command User Interface">1</SetControllerInput>
6    <IncrementControllerInput Description="Change Indicator User Interface"/>
7  </Set>
8
9  <Check>
10   <CheckSensorOutput Description="Mission Status">1</CheckSensorOutput>
11 </Check>
12
13 <Set>
14   <AssertCondition Description="Sensor Resolution Medium">
15     <ConditionItem Name="SensorResolution" Negated="False">
16       <Parameter>
17         <ParameterItem Name="VisionSensorCamera" Position="1" />
18         <ParameterItem Name="ResolutionLow" Position="2" />
19       </Parameter>
20     </ConditionItem>
21     <ConditionItem Name="SensorResolution" Negated="True">
22       <Parameter>
23         <ParameterItem Name="VisionSensorCamera" Position="1" />
24         <ParameterItem Name="ResolutionMedium" Position="2" />
25       </Parameter>
26     </ConditionItem>
27     <ConditionItem Name="SensorResolution" Negated="True">
28       <Parameter>
29         <ParameterItem Name="VisionSensorCamera" Position="1" />
30         <ParameterItem Name="ResolutionHigh" Position="2" />
31       </Parameter>
32     </ConditionItem>
33   </AssertCondition>
34 </Set>
35
36 <Set>
37   <SetControllerInput Description="Planning Problem">2</SetControllerInput>
38   <SetControllerInput Description="Command User Interface">2</SetControllerInput>
39   <IncrementControllerInput Description="Change Indicator User Interface"/>
40 </Set>
41
42 <Check>
43   <CheckSensorOutputChanged Description="Change Indicator Question"/>
44   <CheckPrimitiveTaskQuery Description="NAVIGATION: Simulating LoadGlobalMap! Success?"/>
45 </Check>
46
47 <Set>
48   <AnswerPrimitiveTaskQuery Description="NAVIGATION: Simulating LoadGlobalMap! Success?">Yes</AnswerPrimitiveTaskQuery>
49   <IncrementControllerInput Description="Change Indicator Answer"/>
50 </Set>

```

Abb. 8.2.: Auszug aus einem Testfall in Form einer XML-Datei zur gezielten Überprüfung des Systemverhaltens in unterschiedlichen Inspektionssituationen.

### 8.3. Evaluierung der Kernkomponenten

In diesem Abschnitt wird die Evaluierung der Kernkomponenten in Bezug auf die Funktionsfähigkeit und das Verhalten in relevanten Inspektionssituationen erläutert. Sie erfolgt im Wesentli-

chen mit Hilfe des Simulationsmodus und umfasst die Steuerungsarchitektur, die Wissensbasis, die Erzeugung und Ausführung von Inspektionsplänen sowie die aktive Untersuchung interessierender Entitäten.

### 8.3.1. Steuerungsarchitektur

Die Funktionsfähigkeit der Steuerungsarchitektur und die korrekte Abbildung des Regelkreises der autonomen Inspektion wurden zunächst mit Hilfe des Simulationsmodus überprüft. Nach der Umsetzung der Architektur in entsprechende Gruppen und Module mit Hilfe von MCA2 (siehe Abschnitt 7.4.1), wurden die zentralen Komponenten der Steuerungsarchitektur realisiert: *Wissensbasis*, *Planer*, *Manager*, *Ausführungseinheit* und *Semantische Inspektion*. Für die einzelnen Ausführungskomponenten, zum Beispiel in der *Basissteuerung* und der *Inspektionsdatenauswertung*, wurde zunächst der interaktive Modus umgesetzt. Durch die Simulation der Ausführung der elementaren Aufgaben mit Hilfe von Fragen an den Benutzer wurde so bereits zu einem frühen Zeitpunkt die Überprüfung der einzelnen zentralen Komponenten und die Evaluierung des Zusammenwirkens der Komponenten in der Steuerungsarchitektur ermöglicht. Auf diese Weise konnte die Realisierung der in Abschnitt 3.2.3 beschriebenen Abläufe erfolgreich validiert und somit die prinzipielle Eignung der vorgeschlagenen Steuerungsarchitektur gezeigt werden.

Daran anschließend wurden die Ausführungskomponenten sukzessive um Fähigkeiten zur tatsächlichen Ausführung der elementaren Aufgaben erweitert. Auf diese Weise konnte das System stufenweise in Betrieb genommen und schritthaltend evaluiert werden. Darüber hinaus wurden verschiedene Testfälle entwickelt, welche eine automatisierte Überprüfung der Abläufe innerhalb der Steuerungsarchitektur erlauben.

### 8.3.2. Wissensbasis

Die Evaluierung der Wissensbasis gliedert sich in die zwei Teilbereiche Konsistenz und Korrektheit. Die Konsistenz der einzelnen Ontologien wird jeweils beim initialen Öffnen der Ontologien mit Hilfe des Reasoners KAON2 überprüft. Wird hierbei eine Inkonsistenz entdeckt, wird die Ausführung der semantischen Missionssteuerung mit einer entsprechenden Fehlermeldung beendet. Die Überprüfung der Korrektheit des spezifizierten Wissens erfolgte anhand von Kompetenzfragen (siehe Abschnitt 4.1.1). Für den Zugriff auf die Wissensbasis wurde eine spezielle Bedienkomponente entwickelt (siehe Abbildung 7.21). Mit ihrer Hilfe können sowohl vordefinierte Anfragen, wie zum Beispiel die Kompetenzfragen, als auch beliebige benutzerdefinierte SPARQL-Anfragen (siehe Abschnitt 2.2.2) an die Wissensbasis gerichtet werden. Die entsprechenden Antworten werden übersichtlich in Form einer Tabelle dargestellt und erlauben so eine einfache Überprüfung der Ergebnisse (siehe Abbildung 8.3). Insgesamt existieren 74 vordefinierte Anfragen, die sich in drei unterschiedliche Bereiche gliedern: Fragen zur Definition von

OWL-Klassen (siehe Abbildung 8.3, links oben), Fragen zu Unterklassen und Individuen von OWL-Klassen (siehe Abbildung 8.3, rechts oben) sowie komplexere Mischfragen (siehe Abbildung 8.3, Mitte). Die Antworten auf die Fragen zur Definition von OWL-Klassen bestehen aus einer Liste von OWL-Propertyts mit ihrem jeweiligen Wertebereich. Sie können somit zur Überprüfung der Modellierung einzelner Konzepte auf Vollständigkeit herangezogen werden und geben darüber hinaus auch über einfache strukturelle Zusammenhänge zwischen den einzelnen Konzepten Aufschluss. Die Fragen zu Unterklassen und Individuen liefern jeweils eine entsprechende Liste für eine gegebene OWL-Klasse zurück und dienen damit unter anderem der Ermittlung von Instanzen. Die komplexeren Mischfragen realisieren weitergehende Anfragen an die Wissensbasis, die sich zum Teil aus den bisher genannten Fragetypen zusammensetzen. Sie erlauben die Überprüfung von komplexeren syntaktischen und semantischen Eigenschaften und Zusammenhängen. Mit ihrer Hilfe kann das gesamte Potential des Reasoners KAON2 ausgeschöpft werden.

### 8.3.3. Erzeugung und Ausführung von Inspektionsplänen

Das in Kapitel 5 vorgestellte Planungsverfahren basiert im Wesentlichen auf der Bestimmung von so genannten *zusammengefassten Planinformationen*. Zur Überprüfung der korrekten Realisierung der Verfahren zur Bestimmung von zusammengefassten Planinformationen und zur nebenläufigen hierarchischen Koordination wurde die in [27] als Arbeitsbeispiel verwendete Planungsdomäne *Produktionsbetrieb* (engl. *manufacturing domain*) umgesetzt (siehe [58]). Anhand der in [27] zu den konkreten zusammengefassten Planinformationen gemachten Angaben konnte die Korrektheit der realisierten Verfahren erfolgreich bestätigt werden.

Darüber hinaus wurde zur manuellen Überprüfung der generierten Flexiblen Hierarchischen Pläne eine entsprechende Bedienkomponente realisiert, welche die Baumstruktur der Pläne geeignet visualisiert und detaillierte Informationen zu den einzelnen Knoten zur Verfügung stellt (siehe Abschnitt 7.4.1). Dies umfasst unter anderem die zusammengefassten Bedingungen und die zusammengefassten Ressourcenverbräuche. Zur Evaluierung wurden 19 verschiedene Planungsprobleme mit insgesamt 27 elementaren Aufgaben, 38 nichtelementaren Aufgaben sowie 59 Zerlegungsmethoden an den Planer übergeben. Für alle Planungsprobleme wurden erfolgreich korrekte Pläne erzeugt und ausgeführt.

Des Weiteren wurden im Simulationsmodus gezielt Ausführungsfehler und Ausnahmesituationen provoziert, um auf diese Weise das Verhalten von *Manager*, *Planer* und *Ausführungseinheit* zu untersuchen. An dieser Stelle hat sich der in Abschnitt 5.4.1 beschriebene Vorteil des Verfahrens bestätigt, dass durch die Beibehaltung aller konsistenten Planverfeinerungen der erforderliche Neuplanungsaufwand reduziert werden kann.

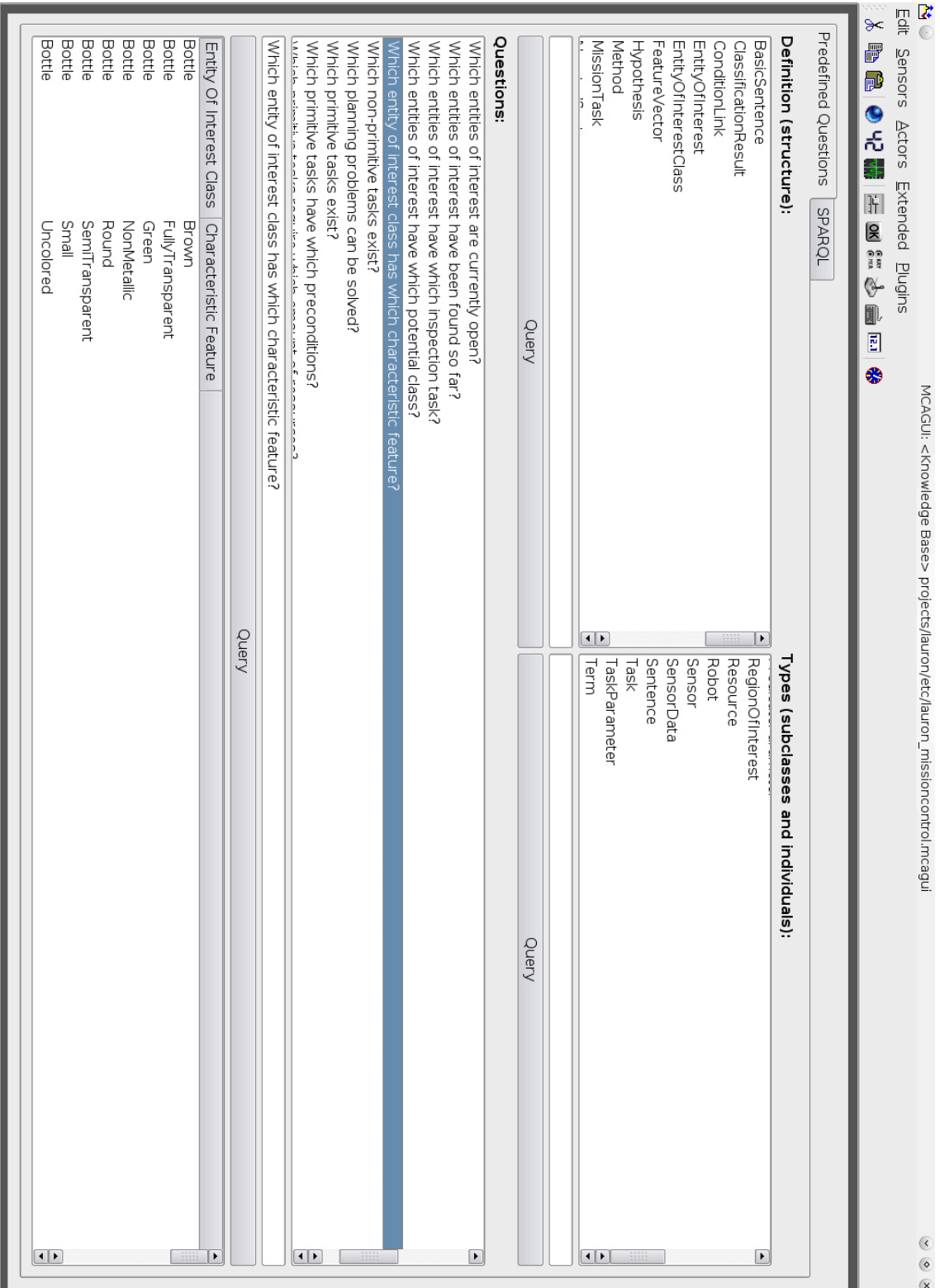


Abb. 8.3.: Vordefinierte Anfragen zur Überprüfung der Korrektheit des in der Wissensbasis spezifizierten Wissens.

### 8.3.4. Aktive Untersuchung interessierender Entitäten

Die Überprüfung der Korrektheit der aktiven Untersuchung interessierender Entitäten erfolgte im Wesentlichen anhand von definierten Testfällen. Mit ihrer Hilfe konnten unterschiedliche Inspektionssituationen systematisch erzeugt und der Ablauf der aktiven Untersuchung (siehe Abschnitt 6.2) überprüft werden. Insbesondere konnten mit Hilfe der Testfälle die Ergebnisse der einzelnen Inspektionsaufgaben und die Werte der entsprechenden Einflussfaktoren systematisch variiert werden. Auf diese Weise konnten die einzelnen Verfahren der aktiven Untersuchung gezielt untersucht und erfolgreich getestet werden. Dies umfasste insbesondere die Fusion der Datenauswertungsergebnisse mit Hilfe des Bayes'schen Netzwerks, die Bewertung der Datenauswertungsergebnisse anhand der Entropie-basierten Konfidenz sowie die Auswahl und Priorisierung von Inspektionsaufgaben mit Hilfe der Regelbasis des semantischen Inspektionsmodells. Insgesamt wurden 97 verschiedene Testfälle zur Validierung eingesetzt, die alle erfolgreich vom System durchlaufen wurden.

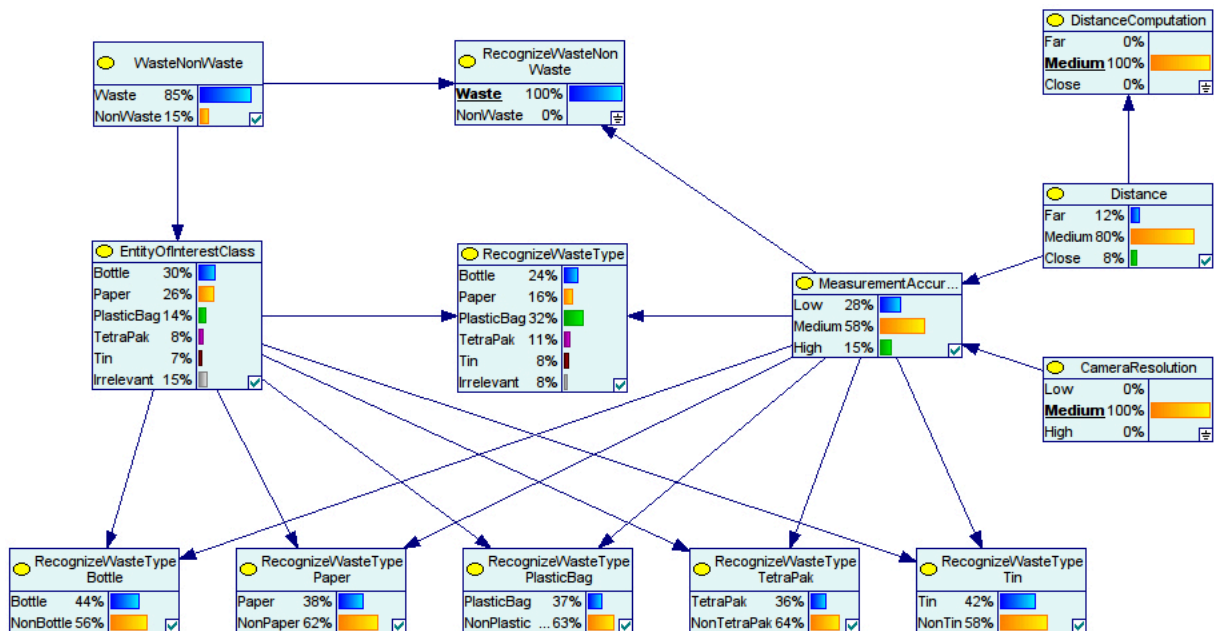


Abb. 8.4.: Bayes'sches Netzwerk nach der Durchführung der Klassifikation Abfall / Kein Abfall. Das Ergebnis *Abfall* wurde im Knoten *RecognizeWasteNonWaste* als Evidenz eingetragen.

Im Folgenden wird die Evaluierung der genannten Punkte anhand eines Beispiels nochmals verdeutlicht. Nach der Segmentierung einer interessierenden Region in den Bilddaten wird zunächst die Klassifikation Abfall / Kein Abfall durchgeführt. Das Ergebnis der Klassifikation, in diesem Fall *Abfall*, wird zusammen mit den Werten der indirekten Einflussfaktoren *Kameraauflösung* und *berechnete Distanz* in das Bayes'sche Netzwerk zur Fusion der Datenauswertungsergebnisse als Evidenz eingetragen. Anschließend wird die Wahrscheinlichkeitsverteilung für das tatsächliche Vorliegen der verschiedenen Entitätsklassen aktualisiert (siehe Abbildung 8.4). Die höchste Wahrscheinlichkeit entfällt mit 30 % auf die Abfallart *Flasche*, direkt gefolgt von

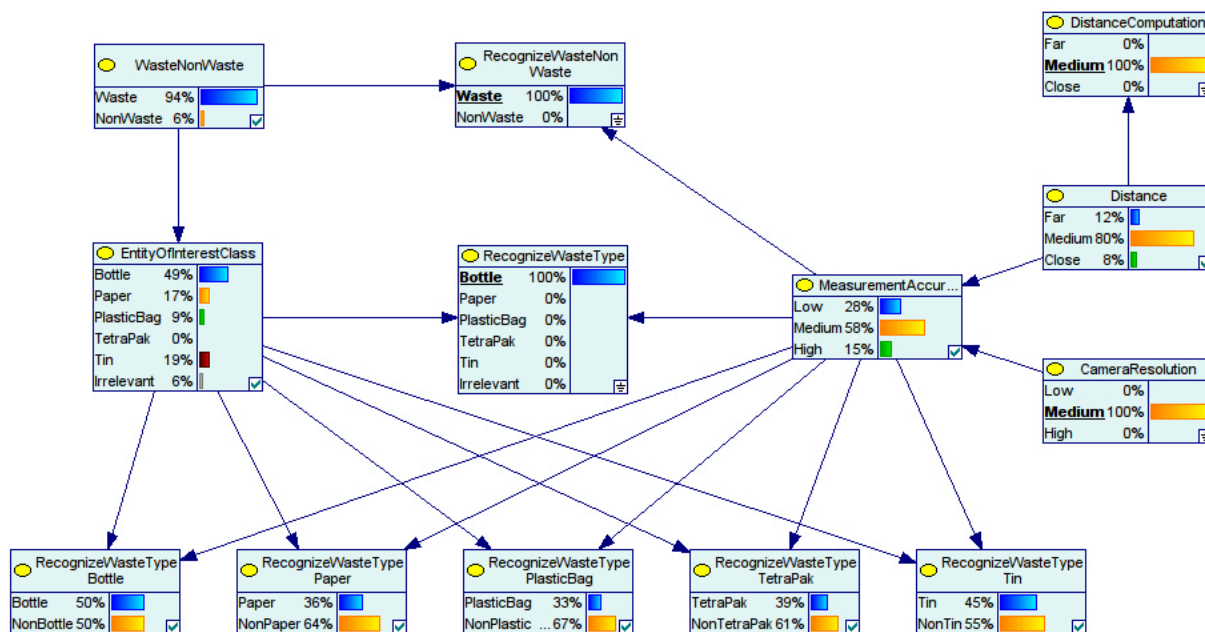


Abb. 8.5.: Bayes'sches Netzwerk nach der Durchführung der Klassifikation der Abfallart. Das Ergebnis *Flasche* wurde im Knoten *RecognizeWasteType* eingetragen.

der Abfallart *Papier* mit 26 %. Die berechnete Konfidenz für die Wahrscheinlichkeitsverteilung beträgt 7 % (siehe Abbildung 8.7). Im nächsten Schritt wird die Klassifikation der Abfallart durchgeführt. Diese liefert als Ergebnis die Klasse *Flasche*, was wiederum in das Bayes'sche Netzwerk als Evidenz eingetragen wird. Die Aktualisierung der Wahrscheinlichkeitsverteilung (siehe Abbildung 8.5) führt dazu, dass die Wahrscheinlichkeit für die Abfallart *Flasche* auf 49 % ansteigt und die Wahrscheinlichkeit für die Abfallart *Papier* auf 17 % zurückgeht. Die berechnete Konfidenz für die aktualisierte Wahrscheinlichkeitsverteilung beträgt 24 %. Anhand der Regelbasis des semantischen Inspektionsmodells wird festgestellt, dass die Konfidenz noch nicht ausreichend ist und es wird die Absicherung der Abfallart *Flasche* als nächste durchzuführende Inspektionsaufgabe bestimmt. Das Ergebnis der Absicherung ist ebenfalls die Klasse *Flasche*. Nach der Eintragung in das Bayes'sche Netzwerk und der Aktualisierung der Wahrscheinlichkeitsverteilung (siehe Abbildung 8.6) beträgt die Wahrscheinlichkeit für die Abfallart *Flasche* 66 %. Die Wahrscheinlichkeit für die Abfallart *Papier* sinkt auf 11 % und die berechnete Konfidenz für die Wahrscheinlichkeitsverteilung steigt auf 39 % an. Die Konfidenz wird anhand der Regelbasis für ausreichend befunden und als nächste Inspektionsaufgabe wird das Aufsammeln der interessierenden Entität ausgewählt.

### 8.4. Evaluierung der erweiterten Fähigkeiten von LAURON IV

Im Folgenden wird auf die Evaluierung der erweiterten Fähigkeiten von LAURON IV eingegangen. Dies umfasst insbesondere die im Rahmen dieser Arbeit entwickelten Fähigkeiten zur



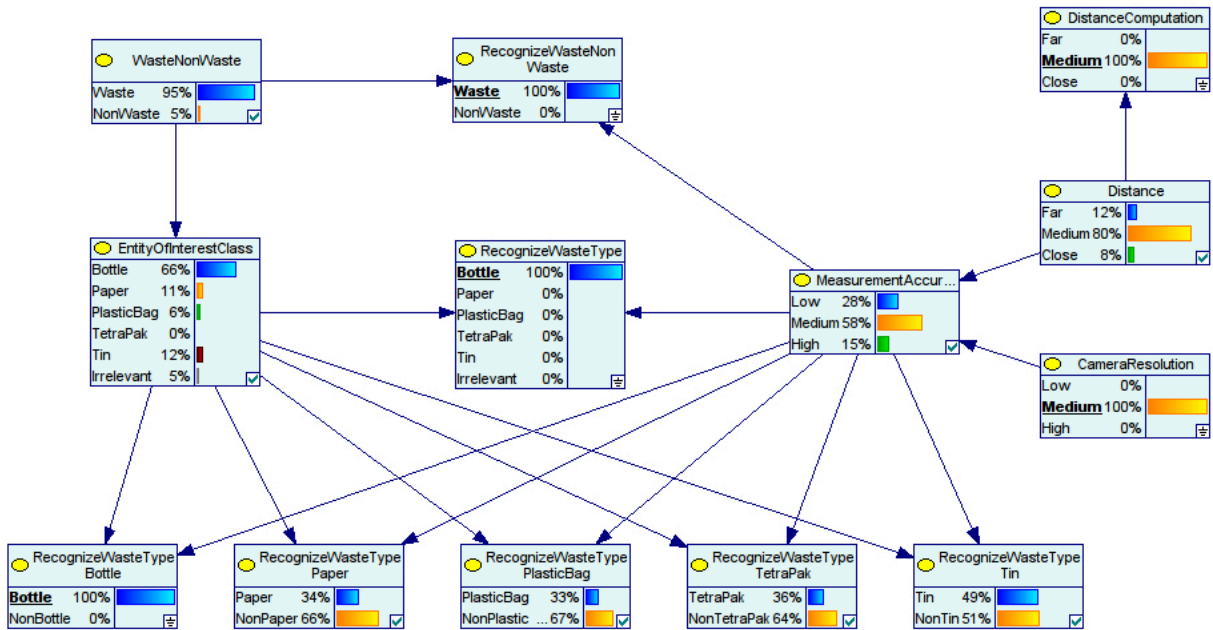


Abb. 8.6.: Bayes'sches Netzwerk nach der Durchführung der Absicherung der Abfallart *Flasche*. Das Ergebnis *Flasche* wurde im Knoten *RecognizeWasteTypeBottle* eingetragen.

Bottle [%]	Paper [%]	PlasticBag [%]	TetraPak [%]	Tin [%]	Irrelevant [%]	Confidence [%]
30	26	14	8	7	15	7
49	17	9	0	19	6	24
66	11	6	0	12	5	39

Abb. 8.7.: Wahrscheinlichkeitsverteilungen für das tatsächliche Vorliegen der einzelnen Abfallklassen mit den zugehörigen Konfidenzen.

globalen Navigation sowie zur Detektion und Klassifikation von Abfall, welche eine wesentliche Grundlage für das gewählte Inspektionsszenario bilden.

### 8.4.1. Globale Navigation

Die Evaluierung der Fähigkeiten zur globalen Navigation gliedert sich in drei Teilbereiche. In den ersten beiden Teilbereichen wird die Evaluierung der entwickelten Verfahren zur Erreichung vorgegebener Zielpunkte und Regionen sowie zum systematischen Abläufen von Regionen anhand von Standard-Benchmarks erläutert. Daran anschließend wird im dritten Teilbereich die Evaluierung der gesamten globalen Navigation mit LAURON IV vorgestellt. Hierfür wurden mit dem Gesamtsystem sowohl unter Laborbedingungen als auch in unstrukturiertem Gelände so genannte *Navigationsmissionen* durchgeführt.

#### Erreichung vorgegebener Zielpunkte und Regionen

Das auf *Rapidly-Exploring Random Trees* (RRT) basierende Verfahren zur Planung von Pfaden zu vorgegebenen Zielpunkten und Regionen (siehe Abschnitt 7.2.1) wurde mit Hilfe spezieller Benchmark-Umgebungen evaluiert, die häufig zur Bewertung von Bahnplanungsverfahren her-

angezogen werden (siehe auch [78]). Im Rahmen dieser Arbeit wurden die fünf in Abbildung 8.8 dargestellten Benchmark-Umgebungen verwendet. Als Evaluationskriterien zur Beurteilung des entwickelten Verfahrens wurden die Anzahl der benötigten Explorationsschritte sowie die benötigten Laufzeiten zum Aufbau des RRTs und zur anschließenden Pfadglättung gewählt. Die Evaluierung wurde auf einem System mit Intel® Core™2 Quad Q9450 Prozessor mit 2,66 GHz und 2 GB Arbeitsspeicher durchgeführt, wobei die Pfadplanung in einem einzelnen Thread ablief.

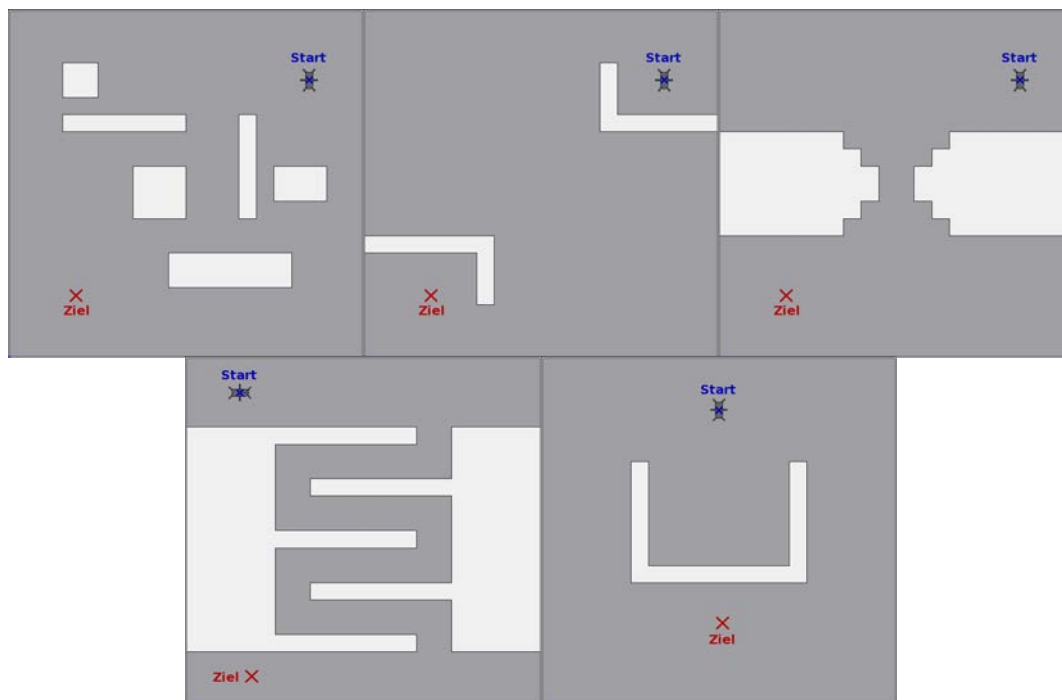


Abb. 8.8.: Benchmark-Umgebungen zur Evaluierung von Bahnplanungsverfahren. Von oben links nach unten rechts: *Simple*, *Star*, *Bottleneck*, *Detour* und *Trap*.

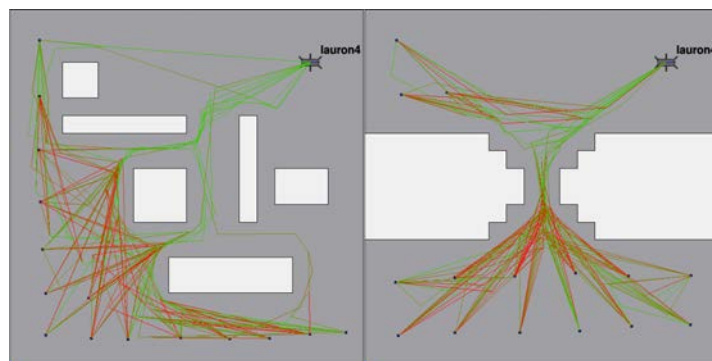
Im Rahmen der Evaluierung wurden zwei unterschiedliche Problemstellungen betrachtet. Bei der ersten Problemstellung waren Pfade von einem Startpunkt zu genau einem Zielpunkt zu planen (siehe Abbildung 8.8). Im Rahmen der zweiten Problemstellung waren von einem Startpunkt gleichzeitig Pfade zu fünfzehn gleichmäßig verteilten Zielpunkten zu planen (siehe Abbildung 8.9). Für die einzelnen Benchmark-Umgebungen wurden pro Problemstellung jeweils zehn Versuche durchgeführt.

Die Ergebnisse für die erste Problemstellung sind in Tabelle 8.1 zusammengefasst. Es sind jeweils der Mittelwert und die Standardabweichung für die zehn pro Benchmark-Umgebung durchgeführten Versuche angegeben. Bei Betrachtung der Ergebnisse ist gut zu erkennen, dass die Anzahl der Explorationsschritte von der Komplexität der Benchmark-Umgebungen abhängig ist. Für die einfachste Benchmark-Umgebung *Bottleneck* wurden im Mittel 425 Explorationsschritte benötigt, wohingegen für die schwierigste Benchmark-Umgebung *Detour* im Mittel

Benchmark	Explorationsschritte	RRT-Planung [s]	Pfadglättung [s]
Simple	599 ± 311	0,04 ± 0,02	0,10 ± 0,02
Star	3797 ± 895	0,23 ± 0,07	0,18 ± 0,02
Bottleneck	425 ± 221	0,03 ± 0,01	0,09 ± 0,02
Detour	27630 ± 2034	2,55 ± 0,50	0,37 ± 0,05
Trap	938 ± 95	0,07 ± 0,01	0,13 ± 0,02

Tab. 8.1.: Ergebnisse der RRT-Bahnplanung für einen Zielpunkt.

27630 Explorationsschritte notwendig waren. Die Ergebnisse bilden somit die intuitive Erwartungshaltung qualitativ sehr gut ab. Die Laufzeiten für die Erzeugung der zielgerichteten RRTs liegen fast immer deutlich unter einer Sekunde. Nur für die Benchmark-Umgebung *Detour* wurden im Mittel 2,55 Sekunden benötigt. Anhand der Benchmark-Umgebung *Trap* und der hierfür benötigten mittleren Anzahl von 938 Explorationsschritten wird zudem deutlich, dass das gewählte Verfahren nicht für lokale Minima anfällig ist. Die zum Teil relativ großen Standardabweichungen bei der Anzahl der benötigten Explorationsschritte ergeben sich aufgrund der probabilistischen Auswahl neuer Konfigurationen beim Aufbau der RRTs. Für die Pfadglättung ergibt sich ein direkter Zusammenhang zwischen der benötigten Laufzeit und der Länge der Pfade, die aus dem RRT extrahiert wurden.

Abb. 8.9.: Evaluierung der RRT-Bahnplanung für fünfzehn Zielpunkte anhand der Benchmark-Umgebungen *Simple* (links) und *Bottleneck* (rechts).

Die Ergebnisse für die zweite Problemstellung, bei der ausgehend von einem Zielpunkt mit nur einem RRT Pfade zu fünfzehn Zielpunkten gleichzeitig geplant wurden, sind in Tabelle 8.2 angegeben. Die Angaben zur Anzahl der Explorationsschritte und zur benötigten Zeit für die RRT-Planung entsprechen jeweils den über die zehn Versuche gebildeten Mittelwerten sowie den zugehörigen Standardabweichungen. Die benötigte Zeit für die Pfadglättung wurde hingegen jeweils über alle 150 Pfade gemittelt. Vergleicht man die angegebenen Werte mit denen der ersten Problemstellung, so lässt sich feststellen, dass die Anzahl der benötigten Explorationsschritte nicht um den Faktor 15, sondern im Mittel nur um den Faktor 1,8 höher liegt. Mit Hilfe von RRTs ist es somit sehr effizient möglich, Pfade zu mehreren Zielen gleichzeitig zu planen.

Benchmark	Explorationsschritte	RRT-Planung [s]	Pfadglättung [s]
Simple	1768 ± 394	0,15 ± 0,03	0,11 ± 0,01
Star	3460 ± 830	0,24 ± 0,06	0,14 ± 0,03
Bottleneck	1129 ± 80	0,11 ± 0,02	0,09 ± 0,02
Detour	13753 ± 2475	1,14 ± 0,23	0,25 ± 0,06
Trap	1784 ± 147	0,17 ± 0,03	0,16 ± 0,03

Tab. 8.2.: Ergebnisse der RRT-Bahnplanung für fünfzehn Zielpunkte.

Die jeweiligen Zeiten für die Pfadglättung entsprechen wie zu erwarten größenordnungsmäßig denen der ersten Problemstellung.

### Systematisches Absuchen von Regionen

Zur Evaluierung des Verfahrens zum systematischen Absuchen von Regionen wurden ebenfalls Benchmark-Umgebungen eingesetzt (siehe [78]). Die vier im Rahmen dieser Arbeit verwendeten Umgebungen sind in Abbildung 8.10 dargestellt. Das in Abschnitt 7.2.2 erläuterte *Complete Coverage* Verfahren wurde sowohl für die *Distance Transform* als auch für die *Path Transform* Kostenfunktion evaluiert. Die Bewertung der generierten Pfade zum Absuchen der Regionen erfolgte qualitativ. Als Evaluationskriterien wurden dabei die Anzahl der beim Ablaufen des Pfades notwendigen Drehungen des Roboters sowie die Anzahl und die Länge der mehrfach abzulaufenden Pfadsegmente herangezogen.

**Evaluierung der *Distance Transform* Kostenfunktion** Die mit der *Distance Transform* Kostenfunktion erzielten Ergebnisse des *Complete Coverage* Verfahrens sind in Abbildung 8.11 dargestellt. Insgesamt lässt sich feststellen, dass nur wenige kurze Pfadabschnitte doppelt abgelaufen werden müssen – in der Benchmark-Umgebung *Bottleneck* sogar gar keine. Die für die Benchmark-Umgebungen *Bottleneck* und *Fringe* generierten Pfade zeichnen sich zudem durch viele lange gerade Pfadsegmente aus und vermeiden so unnötige Drehungen des Roboters. Die Zahl der notwendigen Drehungen fällt für die Benchmark-Umgebungen *Cycle* und *Fringe* zwar höher aus, erscheint jedoch auch hier durchaus akzeptabel.

**Evaluierung der *Path Transform* Kostenfunktion** Die Ergebnisse des *Complete Coverage* Verfahrens mit der *Path Transform* Kostenfunktion sind in Abbildung 8.12 angegeben. Bezüglich der Anzahl und der Länge der doppelt abzulaufenden Pfadsegmente fällt die Bilanz hier sehr unterschiedlich aus. Obwohl für die Benchmark-Umgebungen *Fringe* und *Zigzag* keinerlei, und für die Benchmark-Umgebung *Cycle* nur sehr wenige kurze doppelte Pfadsegmente erzeugt wurden, enthält der für die Benchmark-Umgebung *Bottleneck* generierte Pfad ein sehr langes doppelt abzulaufendes Pfadsegment im oberen rechten Quadranten. Die Pfade für die Benchmark-Umgebungen *Bottleneck* und *Zigzag* fallen darüber hinaus durch ihre hohe Anzahl

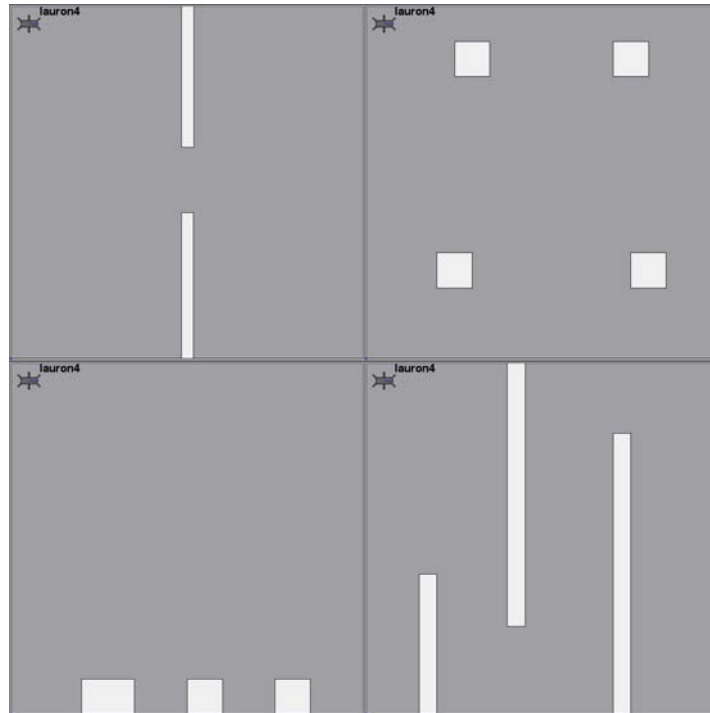


Abb. 8.10.: Benchmark-Umgebungen zur Evaluierung des *Complete Coverage* Verfahrens. Von oben links nach unten rechts: *Bottleneck*, *Cycle*, *Fringe* und *Zigzag*.

von Drehungen auf. Der für die Benchmark-Umgebung *Cycle* generierte Pfad erscheint im Vergleich zudem deutlich ungeordneter als der mit Hilfe der *Distance Transform* Kostenfunktion erzeugte Pfad. Einzig der Pfad für die Benchmark-Umgebung *Fringe* fällt durch die verhältnismäßig wenigen Drehungen positiv auf. Insgesamt lässt sich somit feststellen, dass die *Distance Transform* Kostenfunktion zum systematischen Absuchen von Regionen geeigneter erscheint.

### Durchführung von Navigationsmissionen

Zur Evaluierung der integrierten globalen Navigation wurden mit dem Gesamtsystem so genannte *Navigationsmissionen* durchgeführt. Ziel der Navigationsmissionen ist das Erreichen und systematische Ablaufen von vorgegebenen Regionen. Zu Beginn der Mission gibt der menschliche Operator der semantischen Missionssteuerung mit Hilfe der graphischen Bedienoberfläche eine abzulaufende Region sowie einen Zielpunkt anhand einer globalen Karte vor und übergibt anschließend die Kontrolle an die semantische Missionssteuerung. Die Aufgabe besteht nun zunächst darin, autonom einen Pfad zum Erreichen der vorgegebenen Region zu planen und auszuführen. Ist die Region erreicht, so ist ein geeigneter Pfad zum systematischen Ablaufen der Region zu planen und auszuführen. Nach dem vollständigen Ablaufen der Region ist wiederum ein Pfad zum vorgegebenen Zielpunkt zu planen und auszuführen.

Im Rahmen der Evaluierung wurden verschiedene Navigationsmissionen durchgeführt. Dies

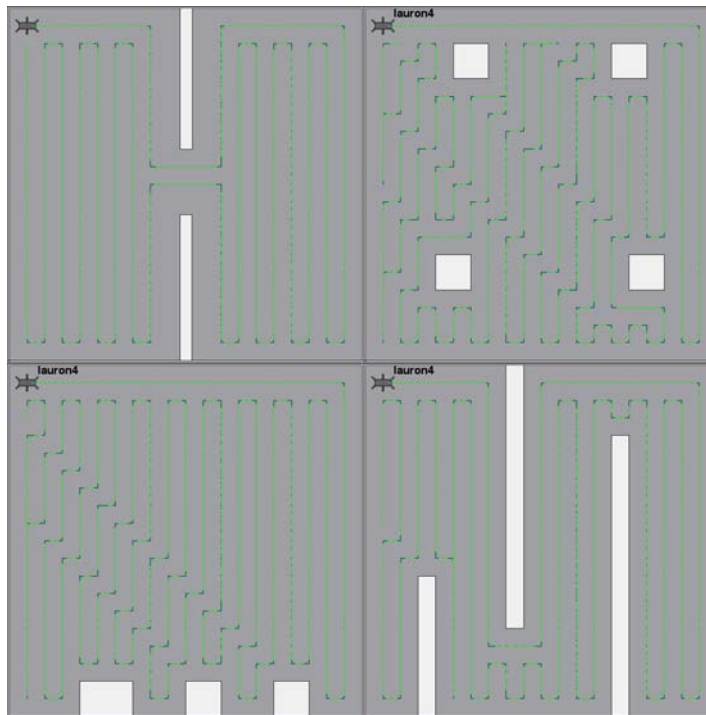


Abb. 8.11.: Evaluierung der *Distance Transform* Kostenfunktion anhand der Benchmark-Umgebungen.

erfolgte sowohl unter kontrollierten Laborbedingungen als auch in unstrukturiertem Gelände. Ein Beispiel für eine Navigationsmission in unstrukturiertem Gelände ist in Abbildung 8.13 abgebildet. Zur Markierung der systematisch abzulaufenden Region wurde diese mit einer Begrenzung aus Steinen umgeben. Der vorgegebene Zielpunkt wurde durch einen Steinhaufen markiert (oben rechts im Bild). Im Rahmen der einzelnen Navigationsmissionen konnten jeweils erfolgreich Pfade zum Erreichen der vorgegebenen Regionen, zum systematischen Absuchen der Regionen sowie zum Erreichen der Zielpunkte generiert werden.

#### 8.4.2. Detektion und Klassifikation von Abfall

Zur Evaluierung der Detektion und Klassifikation von Abfall wurden mit dem Gesamtsystem systematisch so genannte *Detektionsmissionen* durchgeführt. Ziel der Detektionsmissionen ist die Detektion und Klassifikation von interessierenden Entitäten. Dazu werden dem Roboter unter kontrollierten und nahezu identischen Randbedingungen verschiedene Objekte zur Erkennung vorgelegt. Anschließend wird der semantischen Missionssteuerung eine entsprechende Missionsaufgabe übergeben, woraufhin autonom Sensormessungen durchgeführt und die Verfahren zur Detektion auffälliger Bildbereiche sowie zur Klassifikation anhand von Texturmerkmalen angewendet werden. Im Folgenden werden die Ergebnisse der sowohl unter kontrollierten Laborbedingungen als auch in unstrukturiertem Gelände durchgeführten Detektionsmissionen mit LAURON IV vorgestellt.

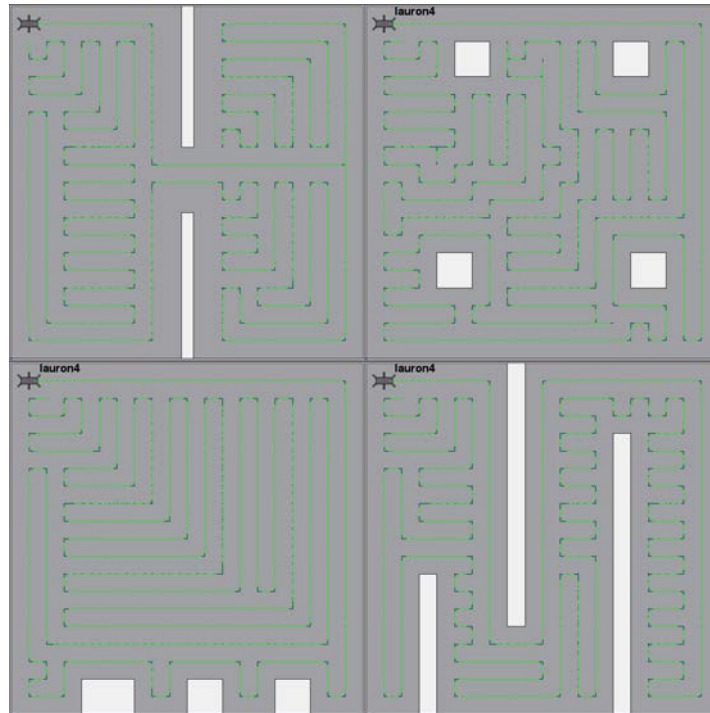


Abb. 8.12.: Evaluierung der *Path Transform* Kostenfunktion anhand der Benchmark-Umgebungen.

### Erprobung unter Laborbedingungen

Die Anzahl der Objekte in den einzelnen Klassen sowie die Anzahl der zugehörigen segmentierten Bildbereiche sind für die unter Laborbedingungen durchgeführten Experimente in Tabelle 8.3 aufgeführt. Eine vollständige Übersicht über die verwendeten Validierungsobjekte befindet sich in Anhang B.2. Bezüglich der Detektion der Objekte ist festzustellen, dass für jedes Objekt jeweils mindestens ein zugehöriger Bildbereich erfolgreich segmentiert wurde. Die segmentierten Bildbereiche erfassen jedoch in der Regel nicht das ganze Objekt, sondern oft nur Ausschnitte. Dafür werden jedoch häufig mehrere zu einem Objekt gehörende Bildbereiche segmentiert.



Abb. 8.13.: Beispiel für in unstrukturiertem Gelände durchgeführte Navigationsmissionen.

Klasse	Objekte	Interessierende Regionen
Flasche	13	41
Papier	12	36
Plastiktüte	6	19
Tetrapak	4	12
Dose	3	9
Abfall	38	117
Kein Abfall	24	76
<b>Insgesamt</b>	<b>62</b>	<b>193</b>

Tab. 8.3.: Bilddaten der unter Laborbedingungen durchgeführten Detektionsmissionen.

Insgesamt weist die Detektion somit unter kontrollierten Laborbedingungen eine sehr hohe Zuverlässigkeit auf. Die Ergebnisse der Klassifikation Abfall / Kein Abfall sind in Form einer Konfusionsmatrix in Tabelle 8.4 angegeben. Dabei bezeichnen die Spalten der Konfusionsmatrix jeweils die tatsächlichen Klassen und die Zeilen das Ergebnis der Klassifikation. Anhand

	Abfall	Kein Abfall
Abfall	99	18
Kein Abfall	18	58

Tab. 8.4.: Konfusionsmatrix für die Klassifikation Abfall / Kein Abfall im Rahmen der unter Laborbedingungen durchgeführten Detektionsmissionen.

der Konfusionsmatrix lassen sich die in Tabelle 8.5 beschriebenen Kennzahlen berechnen. Dabei bezeichnet *RP* (richtig positiv) die Anzahl der zu Abfallobjekten gehörenden Bildbereiche, die als *Abfall* klassifiziert wurden. *FN* (falsch negativ) gibt die Anzahl der zu Abfallobjekten gehörenden Bildbereiche an, die als *Kein Abfall* klassifiziert wurden. *RN* (richtig negativ) und *FP* (falsch positiv) sind analog definiert und bezeichnen die Anzahl der Bildbereiche, die nicht zu Abfallobjekten gehören und als *Kein Abfall* bzw. *Abfall* klassifiziert wurden. Die entsprechend berechneten Kennzahlen für die Klassifikation Abfall / Kein Abfall sind in Tabelle 8.6 aufgeführt. Die Korrekturklassifikationsrate für die Klassifikation Abfall / Kein Abfall liegt bei 81,3 %. Es fällt insbesondere die hohe Fehlalarmrate von 23,7 % auf. Bei genauerer Analyse lässt sich feststellen, dass viele segmentierte Bildbereiche der Kategorie *Pflanze* falsch positiv klassifiziert werden. Es sind jedoch auch andere Kategorien betroffen. Insgesamt deutet dies darauf hin, dass einige Kategorien der Klasse *Kein Abfall* nicht ausreichend in den Lerndaten abgedeckt sind.

Die Ergebnisse der Klassifikation der Abfallart sind als Konfusionsmatrix in Tabelle 8.7 angegeben. Die Korrekturklassifikationsrate beträgt lediglich 34,3 % und es kommt zu vielen Verwechslungen zwischen den einzelnen Abfallarten. Eine rein texturbasierte Unterscheidung verschiedener Abfallklassen erscheint somit schwierig. Dies ist jedoch in gewisser Hinsicht plau-



Kennzahl	Formel
Sensitivität	$\frac{RP}{RP+FN}$
Spezifität	$\frac{RN}{RN+FP}$
Fehlalarmrate	$\frac{FP}{FP+RN}$
Relevanz	$\frac{RP}{RP+FP}$
Segreganz	$\frac{RN}{RN+FN}$
Korrektklassifikationsrate	$\frac{RP+RN}{RP+FP+RN+FN}$
Fehlklassifikationsrate	$\frac{FP+FN}{RP+FP+RN+FN}$

Tab. 8.5.: Kriterien zur Evaluierung der Klassifikationsergebnisse Abfall / Kein Abfall.

Kennzahl	Wert [%]
Sensitivität	84,6
Spezifität	76,3
Fehlalarmrate	23,7
Relevanz	84,6
Segreganz	76,3
Korrektklassifikationsrate	81,3
Fehlklassifikationsrate	18,7

Tab. 8.6.: Kennzahlen für die Klassifikation Abfall / Kein Abfall im Rahmen der unter Laborbedingungen durchgeführten Detektionsmissionen.

sibel, da sich aufgrund der hohen Diversität allein anhand der Textur eines begrenzten Bildbereichs auch für den Menschen nur schwer bestimmen lässt, um welche Abfallart es sich handelt. Zur Verbesserung der Ergebnisse sind daher zusätzliche Sensormodalitäten hinzuzuziehen, die weiteren Aufschluss über die geometrischen, optischen und physikalischen Objekteigenschaften geben.

### Erprobung in unstrukturiertem Gelände

Neben den unter Laborbedingungen durchgeführten Detektionsmissionen wurden auch in unstrukturiertem Gelände entsprechende Versuche unternommen. Einige Beispiele hierfür sind in Abbildung 8.14 abgebildet. Zur Veranschaulichung sind die Ergebnisse der Klassifikation Abfall / Kein Abfall für zwei Detektionsmissionen exemplarisch in den Abbildungen 8.15 und 8.16 dargestellt. Auf der linken Seite der Abbildungen befindet sich die jeweilige Sensoraufnahme. Daneben und darunter befinden sich die segmentierten Bildbereiche sowie die jeweiligen Klassifikationsergebnisse.

	<b>Flasche</b>	<b>Papier</b>	<b>Plastiktüte</b>	<b>Tetrapak</b>	<b>Dose</b>
<b>Flasche</b>	14	5	2	0	6
<b>Papier</b>	2	8	2	5	0
<b>Plastiktüte</b>	11	11	8	5	0
<b>Tetrapak</b>	3	6	0	2	1
<b>Dose</b>	5	1	0	0	2

Tab. 8.7.: Konfusionsmatrix für die Klassifikation der Abfallart im Rahmen der unter Laborbedingungen durchgeführten Detektionsmissionen.

Einen Überblick über die Anzahl der untersuchten Objekte und der segmentierten Bildbereiche gibt Tabelle 8.8. Von den 29 Abfallobjekten wurden 6 Objekte nicht detektiert. Dabei

<b>Klasse</b>	<b>Objekte</b>	<b>Interessierende Regionen</b>
Flasche	9	10
Papier	11	19
Plastiktüte	3	7
Tetrapak	2	5
Dose	4	4
<b>Abfall</b>	<b>29</b>	<b>45</b>
<b>Kein Abfall</b>	<b>18</b>	<b>26</b>
<b>Insgesamt</b>	<b>47</b>	<b>71</b>

Tab. 8.8.: Bilddaten der in unstrukturiertem Gelände durchgeführten Detektionsmissionen.

handelte es sich in allen sechs Fällen um Szenen mit drei oder mehr verschiedenen Objekten, in denen jeweils ein Objekt nicht detektiert wurde. Ein entsprechendes Beispiel hierfür ist in Abbildung 8.16 zu sehen. Die Detektionsleistung fällt somit bei einer größeren Anzahl von Objekten etwas ab. Dies ist in Teilen auch damit zu erklären, dass in Abhängigkeit vom gewählten Neuronenspannungsschwellenwert nur eine gewisse Anzahl der salientesten Bildregionen detektiert wird. Die Ergebnisse der Klassifikation der verschiedenen Objekte in Abfall / Kein Abfall sind in Tabelle 8.9 als Konfusionsmatrix angegeben. Die anhand der Konfusionsmatrix

	<b>Abfall</b>	<b>Kein Abfall</b>
<b>Abfall</b>	42	1
<b>Kein Abfall</b>	3	25

Tab. 8.9.: Konfusionsmatrix für die Klassifikation Abfall / Kein Abfall im Rahmen der in unstrukturiertem Gelände durchgeführten Detektionsmissionen.

berechneten Kenngrößen sind in Tabelle 8.10 zusammengefasst. Die Korrektklassifikationsrate beträgt 94,4 % und fällt damit im Vergleich zu den unter Laborbedingungen durchgeführten Experimenten höher aus. Die Fehlalarmrate liegt mit 3,8 % deutlich unter dem im Labor ermittelten Wert. Die Ergebnisse der Klassifikation der Abfallart sind in Tabelle 8.11 ebenfalls



Abb. 8.14.: Beispiele für in unstrukturiertem Gelände durchgeführte Detektionsmissionen.

als Konfusionsmatrix angegeben. Die Korrektklassifikationsrate liegt mit 26,2 % unter dem im Labor ermittelten Wert. Die Zahl der Verwechslungen ist somit sehr hoch und bestätigt, dass sich eine rein texturbasierte Erkennung der Abfallart schwierig gestaltet. Um eine sicherere Klassifikation der Abfallart zu erzielen, ist es somit erforderlich, weitere Sensormodalitäten mit einzubeziehen.

## 8.5. Evaluierung des Gesamtsystems

Im Folgenden wird auf die Evaluierung des Gesamtsystems eingegangen. Hierzu wurden sowohl Experimente unter kontrollierten Laborbedingungen als auch in unstrukturiertem Gelände durchgeführt.

### 8.5.1. Aktive Inspektion von Regionen

Zur Evaluierung des integrierten Gesamtsystems mit allen Fähigkeiten wurden so genannte *Inspektionsmissionen* durchgeführt. Ziel der Inspektionsmissionen ist das systematische Absuchen von vorgegebenen Regionen. Zu Beginn der Mission gibt der menschliche Operator der semantischen Missionssteuerung mit Hilfe der graphischen Bedienoberfläche eine abzusuchende Region sowie einen Zielpunkt anhand einer globalen Karte vor und übergibt anschließend

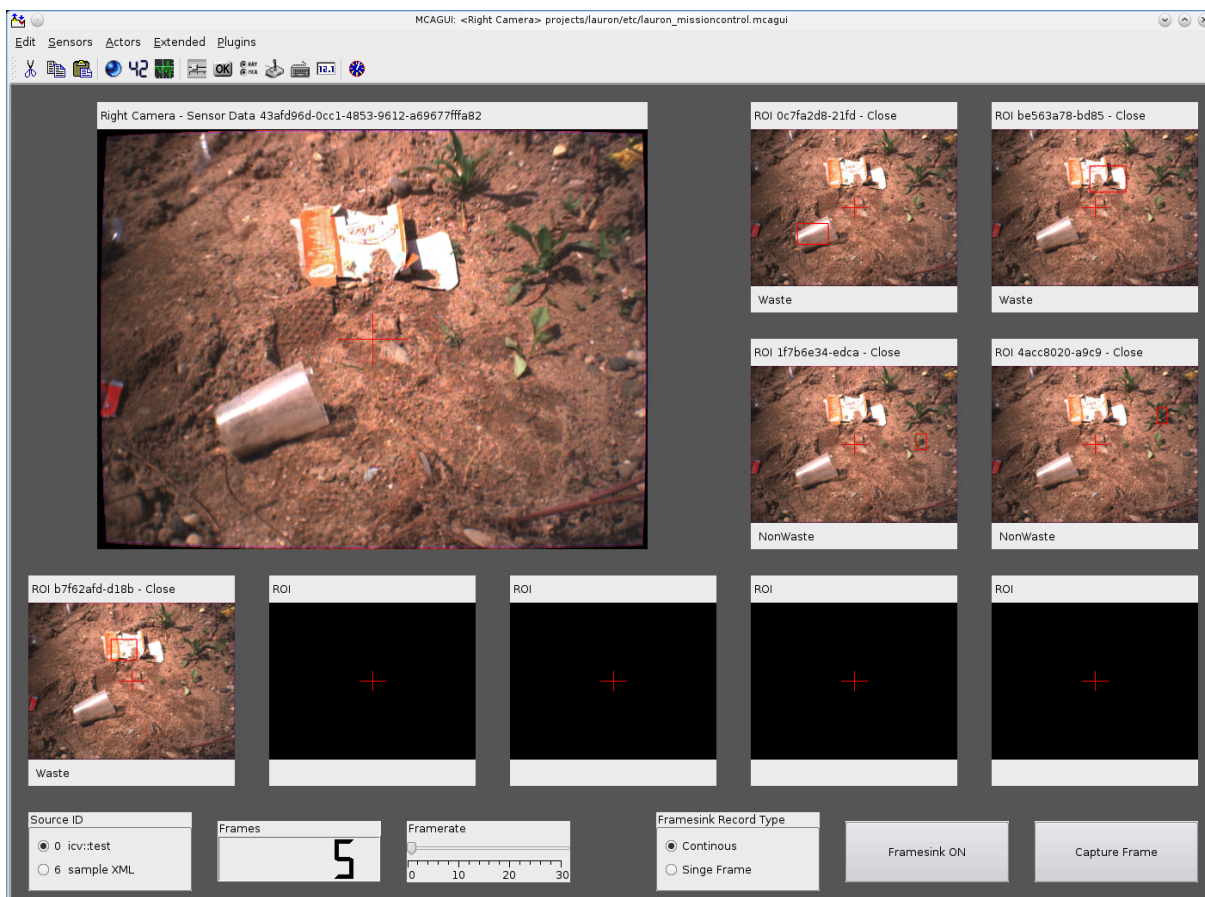


Abb. 8.15.: Ergebnis der Klassifikation Abfall / Kein Abfall einer Detektionsmission an einem Seeufer.

die Kontrolle an die semantische Missionssteuerung. Die Aufgabe besteht nun zunächst darin, einen Pfad zum Erreichen der vorgegebenen Region zu planen und auszuführen. Ist die Region erreicht, so ist ein geeigneter Pfad zum systematischen Absuchen der Region unter Einsatz der Inspektionssensoren zu planen und auszuführen. Werden beim Absuchen interessierende Entitäten gefunden, so ist die Suche zu unterbrechen und die gefundene interessierende Entität so lange aktiv zu untersuchen, bis eine ausreichende Konfidenz der Datenauswertungsergebnisse erreicht ist. Nach dem vollständigen Absuchen der Region ist ein Pfad zum vorgegebenen Zielpunkt zu planen und auszuführen.

### Erprobung unter Laborbedingungen

Unter Laborbedingungen wurden verschiedene Inspektionsmissionen durchgeführt. Zwei der dabei abgesuchten Regionen sind in Abbildung 8.17 dargestellt. Neben einer Begrenzung aus Steinen enthalten die Regionen sowohl verschiedene natürliche Objekte als auch diverse Abfallobjekte. Im Folgenden wird exemplarisch näher auf die Inspektion der linken Region in Abbildung 8.17 eingegangen. Der von der globalen Navigation generierte Pfad zum Absuchen dieser Region ist in Abbildung 8.18 abgebildet. Die autonome Ausführung des Pfades und das

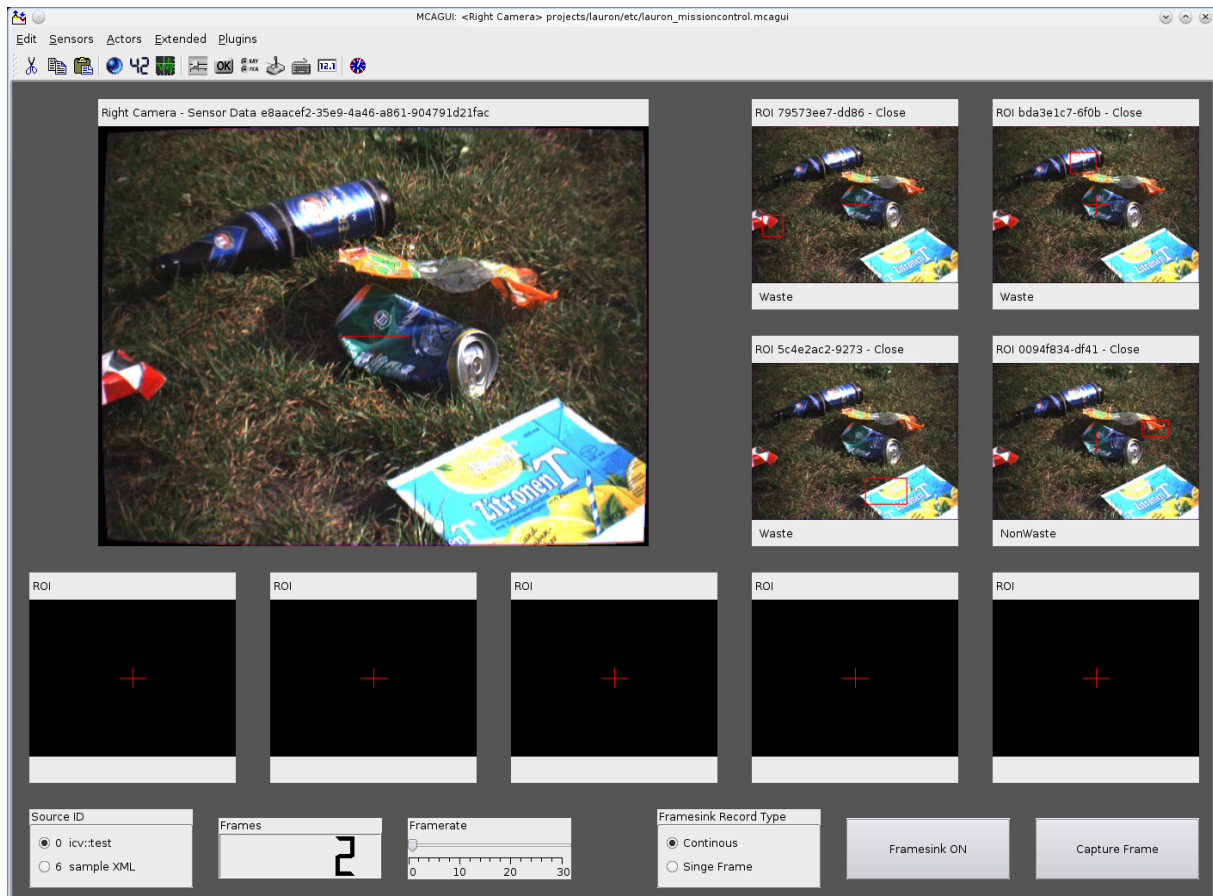


Abb. 8.16.: Ergebnis der Klassifikation Abfall / Kein Abfall einer Detektionsmission in einer Parkanlage.

gleichzeitige Absuchen der Region mit Hilfe des Kamerasystems werden in Abbildung 8.19 veranschaulicht. Die während der Inspektion gefundenen interessierenden Entitäten wurden entsprechend ihrer Entitätsklasse farblich kodiert in die Karte eingezeichnet (siehe Abbildung 8.20). Die quantitative Auswertung der Inspektionsergebnisse ist in den Tabellen 8.12 und 8.13 angegeben. Insgesamt wurden 47 interessierende Regionen gefunden und untersucht. Dabei wurden alle Abfallobjekte in den Sensordaten erfolgreich detektiert. Darüber hinaus wurden alle detektierten Abfallobjekte auch als solche identifiziert. Es traten somit keine falsch Negativen auf. Insgesamt liegen die Korrektklassifikationsrate mit 89,4 % und die Fehlalarmrate mit 12,5 % jeweils leicht über bzw. unter den in Abschnitt 8.4.2 ermittelten Werten. Auf eine nähere Betrachtung der Klassifikation der Abfallart wurde an dieser Stelle verzichtet, da bereits die unter Laborbedingungen durchgeführten Detektionsmissionen gezeigt haben, dass sich eine rein texturbasierte Unterscheidung der Abfallarten schwierig gestaltet.

Bei Betrachtung der Karte mit den gefundenen interessierenden Entitäten (siehe Abbildung 8.20) fällt auf, dass die Positionsgenauigkeit der Einträge eine relativ große Schwankungsbreite aufweist. Dies ist auf Ungenauigkeiten in der Odometrie des Roboters zurückzuführen. Insbesondere die Höhenwerte des Zentralkörpers von LAURON IV sind einem zeitlichen Drift unter-

Kennzahl	Wert [%]
Sensitivität	93,3
Spezifität	96,2
Fehlalarmrate	3,8
Relevanz	97,7
Segreganz	89,3
Korrektklassifikationsrate	94,4
Fehlklassifikationsrate	5,6

Tab. 8.10.: Kennzahlen für die Klassifikation Abfall / Kein Abfall im Rahmen der in unstrukturiertem Gelände durchgeführten Detektionsmissionen.

	Flasche	Papier	Plastiktüte	Tetrapak	Dose
Flasche	5	6	1	0	1
Papier	1	2	0	1	0
Plastiktüte	1	6	1	1	1
Tetrapak	2	3	4	2	1
Dose	1	1	0	0	1

Tab. 8.11.: Konfusionsmatrix für die Klassifikation der Abfallart im Rahmen der in unstrukturiertem Gelände durchgeführten Detektionsmissionen.

worfen, was sich zum Teil erheblich auf die Schätzung der Weltkoordinaten der interessierenden Entitäten anhand ihrer Positionen im Kamerabild auswirkt.

### Erprobung in unstrukturiertem Gelände

In unstrukturiertem Gelände wurden ebenfalls verschiedene Inspektionsmissionen ausgeführt. Zwei Beispiele hierfür sind in Abbildung 8.21 angegeben. Ähnlich wie bei den unter Laborbedingungen durchgeführten Inspektionsmissionen verfügen die zu inspizierenden Regionen über eine Begrenzung aus Steinen und enthalten sowohl natürliche Objekte als auch Abfall-



Abb. 8.17.: Beispiele für unter Laborbedingungen durchgeführte Inspektionsmissionen.

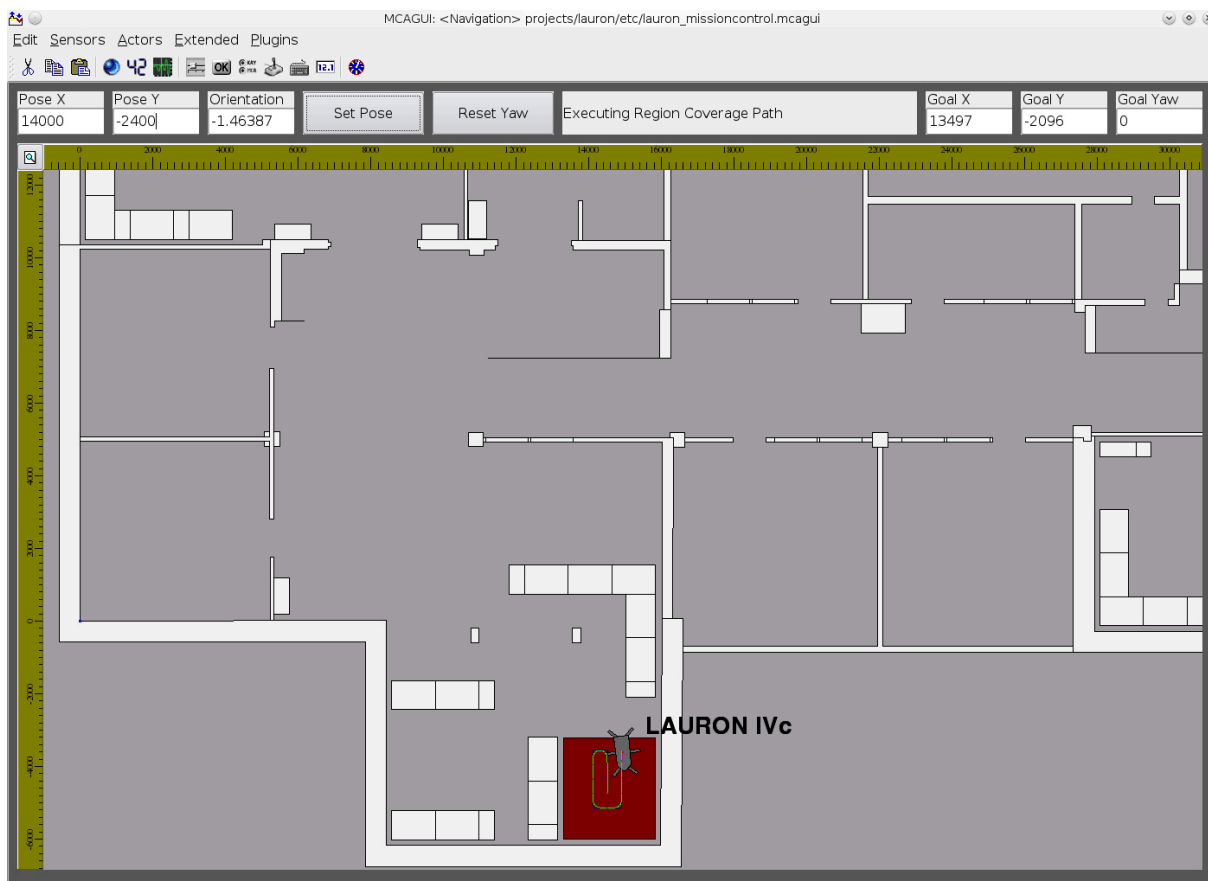


Abb. 8.18.: Pfad zum Absuchen einer Region im Rahmen einer unter Laborbedingungen durchgeführten Inspektionsmission.

	<b>Abfall</b>	<b>Kein Abfall</b>
<b>Abfall</b>	7	5
<b>Kein Abfall</b>	0	35

Tab. 8.12.: Konfusionsmatrix für die Klassifikation Abfall / Kein Abfall im Rahmen einer unter Laborbedingungen durchgeführten Inspektionsmission.

objekte unterschiedlicher Entitätsklassen. Im Folgenden wird exemplarisch auf die Inspektion der rechten Region in Abbildung 8.21 eingegangen. Der generierte Pfad zum Absuchen dieser Region ist in Abbildung 8.22 abgebildet. Die Durchführung der aktiven Inspektion der Region wird in Abbildung 8.23 veranschaulicht und die Ergebnisse der Klassifikation Abfall / Kein Abfall sind in Form einer Konfusionsmatrix in Tabelle 8.14 angegeben. Insgesamt wurden 61 interessierende Regionen detektiert. Lediglich ein Abfallobjekt wurde in den Sensordaten nicht detektiert. Dieses war jedoch nur teilweise im Kamerabild zu sehen. Die anhand der Konfusionsmatrix berechneten Kenngrößen sind in Tabelle 8.15 zusammengestellt. Die segmentierten Bildbereiche von Abfallobjekten wurden ausnahmslos der Klasse Abfall zugeordnet. Es sind somit keine falsch Negativen aufgetreten. Die Korrekturklassifikationsrate liegt jedoch mit 59,0 % deutlich unter dem in Abschnitt 8.4.2 ermittelten Wert. Es fällt insbesondere die sehr hohe

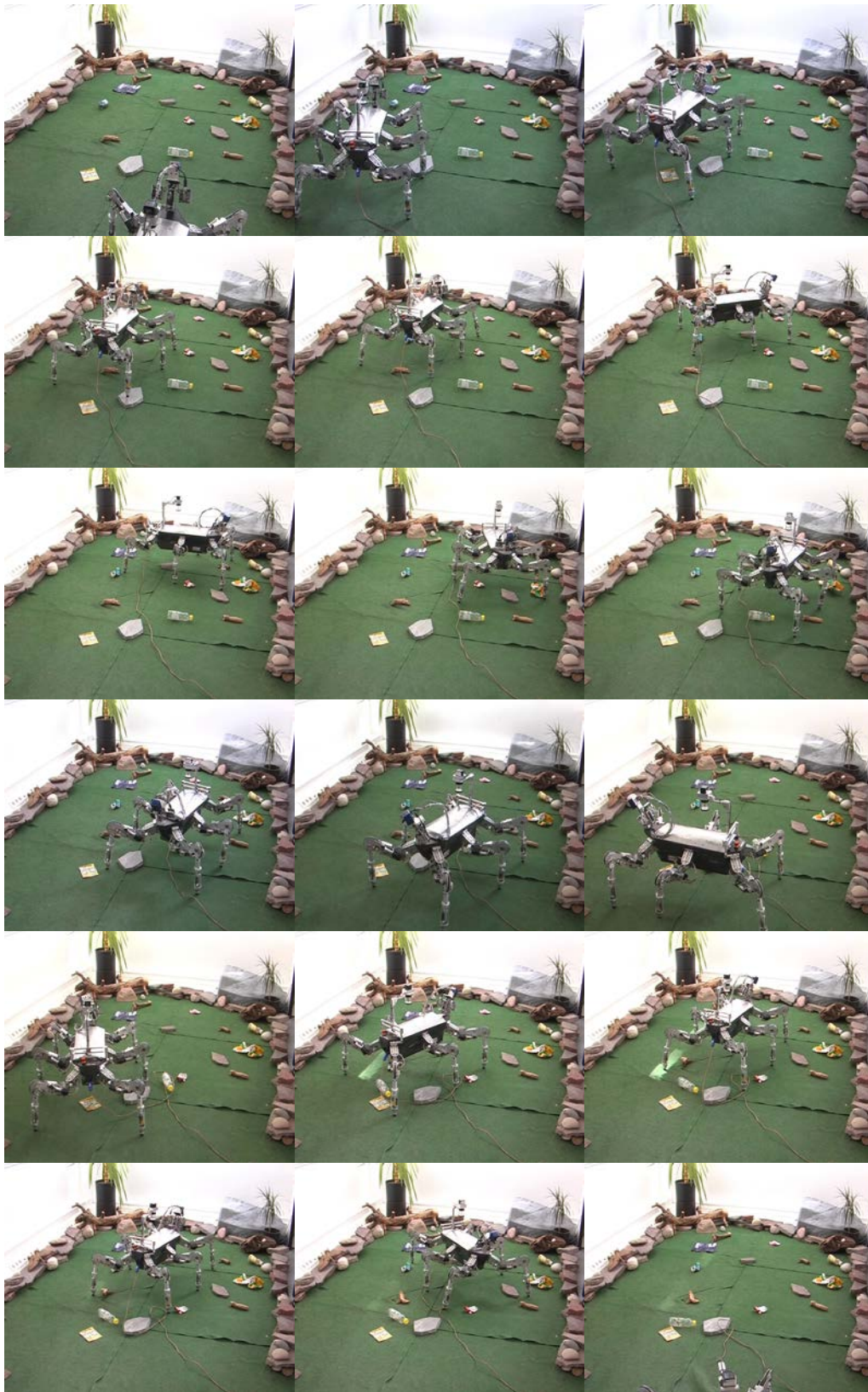


Abb. 8.19.: Bildsequenz zur Veranschaulichung einer unter Laborbedingungen durchgeführten Inspektionsmission.



<b>Kennzahl</b>	<b>Wert [%]</b>
Sensitivität	100,0
Spezifität	87,5
Fehlalarmrate	12,5
Relevanz	58,3
Segreganz	100,0
Korrektklassifikationsrate	89,4
Fehlklassifikationsrate	10,6

Tab. 8.13.: Kennzahlen für die Klassifikation Abfall / Kein Abfall im Rahmen einer unter Laborbedingungen durchgeführten Inspektionsmission.

	<b>Abfall</b>	<b>Kein Abfall</b>
<b>Abfall</b>	13	25
<b>Kein Abfall</b>	0	23

Tab. 8.14.: Konfusionsmatrix für die Klassifikation Abfall / Kein Abfall im Rahmen einer in einer Grünanlage durchgeführten Inspektionsmission.

Fehlalarmrate von 52,1 % auf. Bei genauerer Untersuchung zeigt sich, dass die meisten falsch Positiven bei segmentierten Bildbereichen auftreten, in denen Gras bzw. Wiese zu sehen ist. Dies erscheint plausibel, da in den Lerndaten keine expliziten Texturen für Gras bzw. Wiese enthalten sind. Darüber hinaus weisen diese Bildbereiche eine hohe Diversität an Texturen auf und werden vermutlich auch daher häufig mit Abfallobjekten verwechselt. Durch eine entsprechende Ergänzung der Lerndaten und erneutes Trainieren der Klassifikatoren sollte sich dieser Effekt jedoch deutlich reduzieren lassen. Auf die Betrachtung der Klassifikation der Abfallart wurde aus den oben genannten Gründen an dieser Stelle ebenfalls verzichtet.

Betrachtet man die Karte mit den gefundenen interessierenden Entitäten (siehe Abbildung 8.24), so fallen die Positionsungenauigkeiten noch etwas größer als bei den unter Laborbedingungen durchgeführten Experimenten aus. Dies ist sehr wahrscheinlich auf die in unstrukturiertem Gelände ebenfalls stärker anwachsenden Odometriefehler zurückzuführen.

<b>Kennzahl</b>	<b>Wert [%]</b>
Sensitivität	100,0
Spezifität	47,9
Fehlalarmrate	52,1
Relevanz	34,2
Segreganz	100,0
Korrektklassifikationsrate	59,0
Fehlklassifikationsrate	41,0

Tab. 8.15.: Kennzahlen für die Klassifikation Abfall / Kein Abfall im Rahmen einer in einer Grünanlage durchgeführten Inspektionsmission.

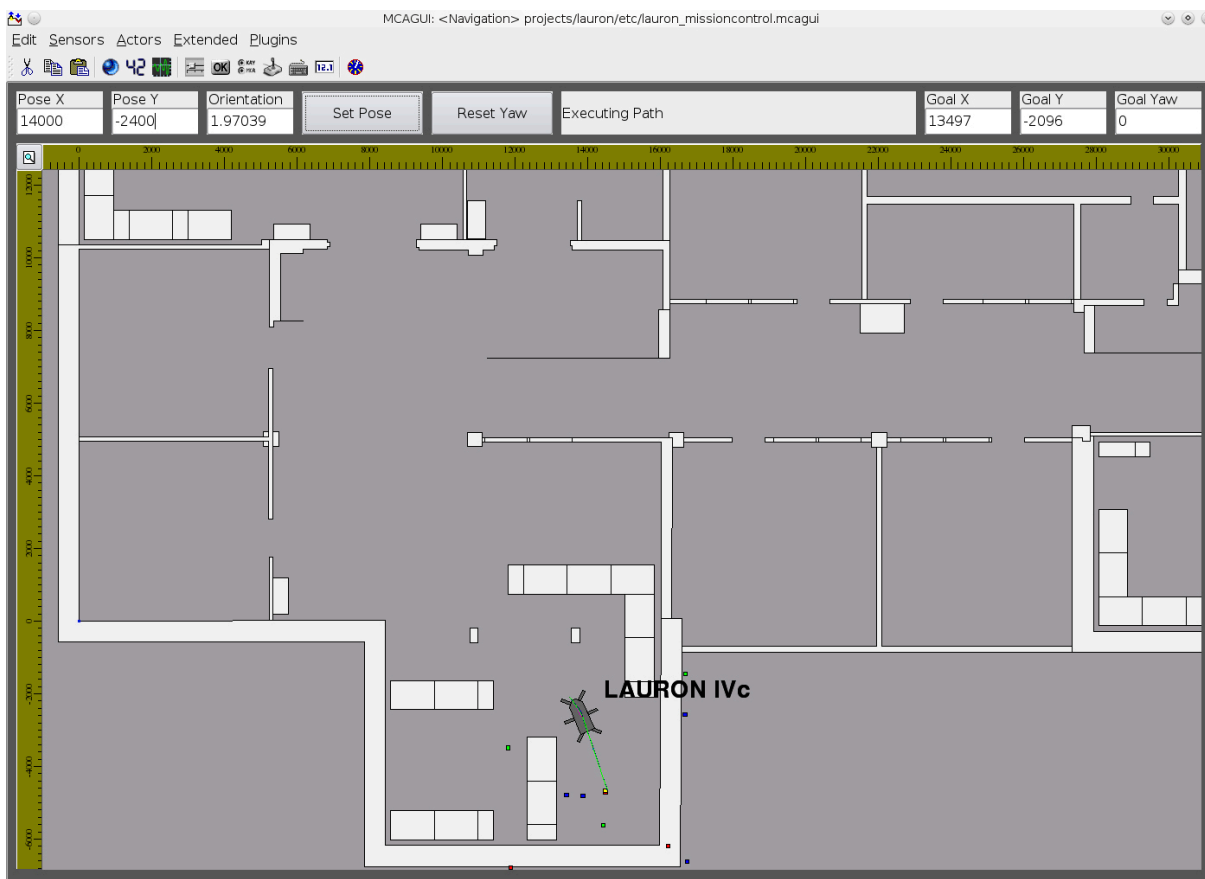


Abb. 8.20.: Interessierende Entitäten, die im Rahmen einer unter Laborbedingungen durchgeführten Inspektionsmission gefunden wurden. Die einzelnen Entitätsklassen sind farblich kodiert dargestellt: Flasche (rot), Papier (grün), Plastiktüte (blau), Tetrapak (cyan) und Dose (gelb).

### 8.6. Portierung auf den mehrsegmentigen Inspektionsroboter KAIRO II

Zur Evaluierung der Übertragbarkeit der semantischen Missionssteuerung auf andere Roboterplattformen und Inspektionsdomänen wurde diese auf den mehrsegmentigen Inspektionsroboter KAIRO II (siehe Abschnitt 2.1.1) portiert. Hierzu war es zunächst erforderlich, den strukturellen Aufbau der Steuerungsarchitektur und der Wissensbasis zu übertragen. Dabei konnten jeweils große Teile der bestehenden Komponenten unverändert bzw. mit geringem Änderungsaufwand übernommen werden. Bezogen auf die Steuerungsarchitektur waren dies die Bedienschnittstelle, der Manager, der Planer, die Ausführungseinheit sowie die Semantische Inspektion. Im Hinblick auf die Wissensbasis konnten die Basisontologien sowie die Kernontologien unverändert übernommen werden. Der benötigte Zeitaufwand, um die Strukturen sowie die genannten Komponenten zu übertragen, ist in Tabelle 8.16 aufgeführt.

Der nächste Schritt nach der strukturellen Übertragung und Anpassung der semantischen Missionssteuerung bestand in der exemplarischen Realisierung einer Mission, die einige grundlegende Fähigkeiten des Roboters demonstriert. Hierzu wurden die Komponenten *Basissteue-*



Abb. 8.21.: Beispiele für in unstrukturiertem Gelände durchgeführte Inspektionsmissionen.

Teilbereich	Zeitaufwand [min]
Steuerungsarchitektur	172
Wissensbasis	31
<b>Summe</b>	<b>203</b>

Tab. 8.16.: Zeitaufwand für die strukturelle Übertragung und Anpassung der semantischen Missionssteuerung auf KAIRO II.

*run* und *Aktorinterface* der Steuerungsarchitektur an die bestehende Steuerung von KAIRO II angepasst. Insbesondere wurde eine neue Ausführungskomponente *Adaptive Steuerung* realisiert. Zunächst wurden zwei elementare Aufgaben umgesetzt: eine Aufgabe zur Initialisierung des Roboters und eine parametrisierbare Aufgabe, die dem in [12] definierten Manöver *Freie Fahrt* entspricht. Darüber hinaus wurde in der Wissensbasis eine entsprechende Mission angelegt (siehe Abbildung 8.25). Die Missionaufgabe besteht hierbei darin, den Roboter zu initialisieren, ein Stück vorwärts und anschließend wieder ein Stück rückwärts zu fahren. Die zur Realisierung dieser Demomission benötigten Zeitaufwände sind in Tabelle 8.17 angegeben. Die benötigte Gesamtzeit bis zur ersten Fahrt des Roboters mit der semantischen Missionssteuerung in der Simulation betrug 232 Minuten.

Teilbereich	Zeitaufwand [min]
Ausführungskomponente <i>Adaptive Steuerung</i>	24
Domänenontologie KAIRO II	53
<b>Summe</b>	<b>77</b>

Tab. 8.17.: Zeitaufwand für die Erstellung der Demomission für KAIRO II.

In einem weiteren Schritt wurde das in [12] beschriebene Manöver *Positioniere Modul* in die semantische Missionssteuerung eingebunden. Dies umfasste zum einen die Erweiterung der entsprechenden Ausführungskomponente *Adaptive Steuerung* und zum anderen die Modellierung des Manövers in der Domänenontologie KAIRO II. Die hierfür jeweils benötigten Zeitaufwände

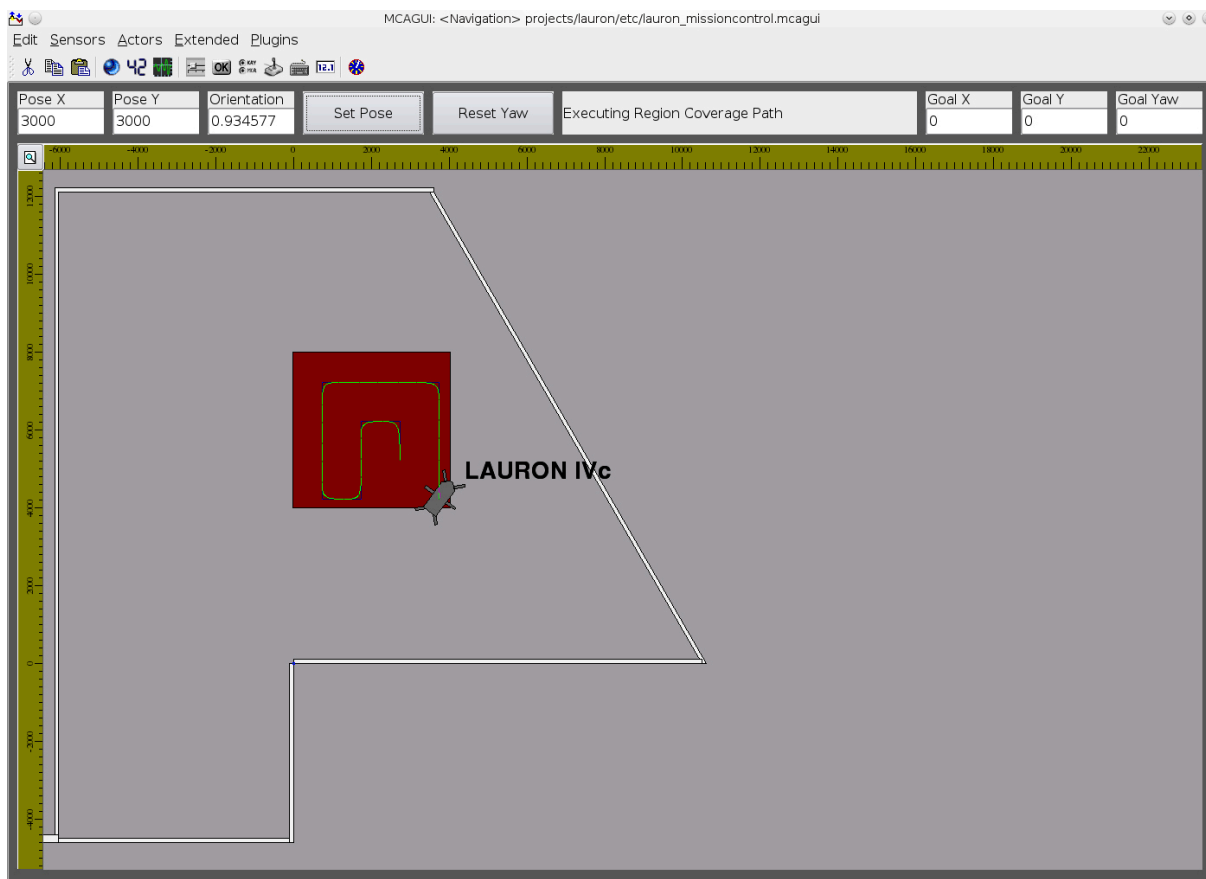


Abb. 8.22.: Pfad zum Absuchen einer Region im Rahmen einer in einer Grünanlage durchgeführten Inspektionsmission.

sind in Tabelle 8.18 angegeben. Daran anschließend wurde das Manöver mit Hilfe der Wissens-

Teilbereich	Zeitaufwand [min]
Ausführungskomponente <i>Adaptive Steuerung</i>	20
Domänenontologie KAIRO II	14
<b>Summe</b>	<b>34</b>

Tab. 8.18.: Zeitaufwand für die Realisierung des Manövers *Positioniere Modul* für KAIRO II.

basis in die bestehende Mission integriert. Die erweiterte Missionsaufgabe (siehe Abbildung 8.26) besteht nun darin, den Roboter zu initialisieren, ein Stück vorwärts zu fahren, ein Segment zu positionieren, es anschließend wieder in die Ausgangslage zurück zu bewegen und ein Stück rückwärts zu fahren. Die auf diese Weise erweiterte Missionsaufgabe ist somit ein Vertreter der in [12] definierten Aufgabenklasse *Inspektion*. Für die entsprechende Einbindung des Manövers *Positioniere Modul* in die bestehende Demomission wurden 22 Minuten benötigt.

Auf die beschriebene Art und Weise lässt sich das mit Hilfe der semantischen Missionssteuerung flexibel nutzbare Verhaltensrepertoire Schritt für Schritt ausbauen. Die angegebenen Zeiten vermitteln einen ersten Eindruck der hierfür notwendigen Aufwände und zeigen, dass



Abb. 8.23.: Bildsequenz zur Veranschaulichung einer in einer Grünanlage durchgeführten Inspektionsmission.

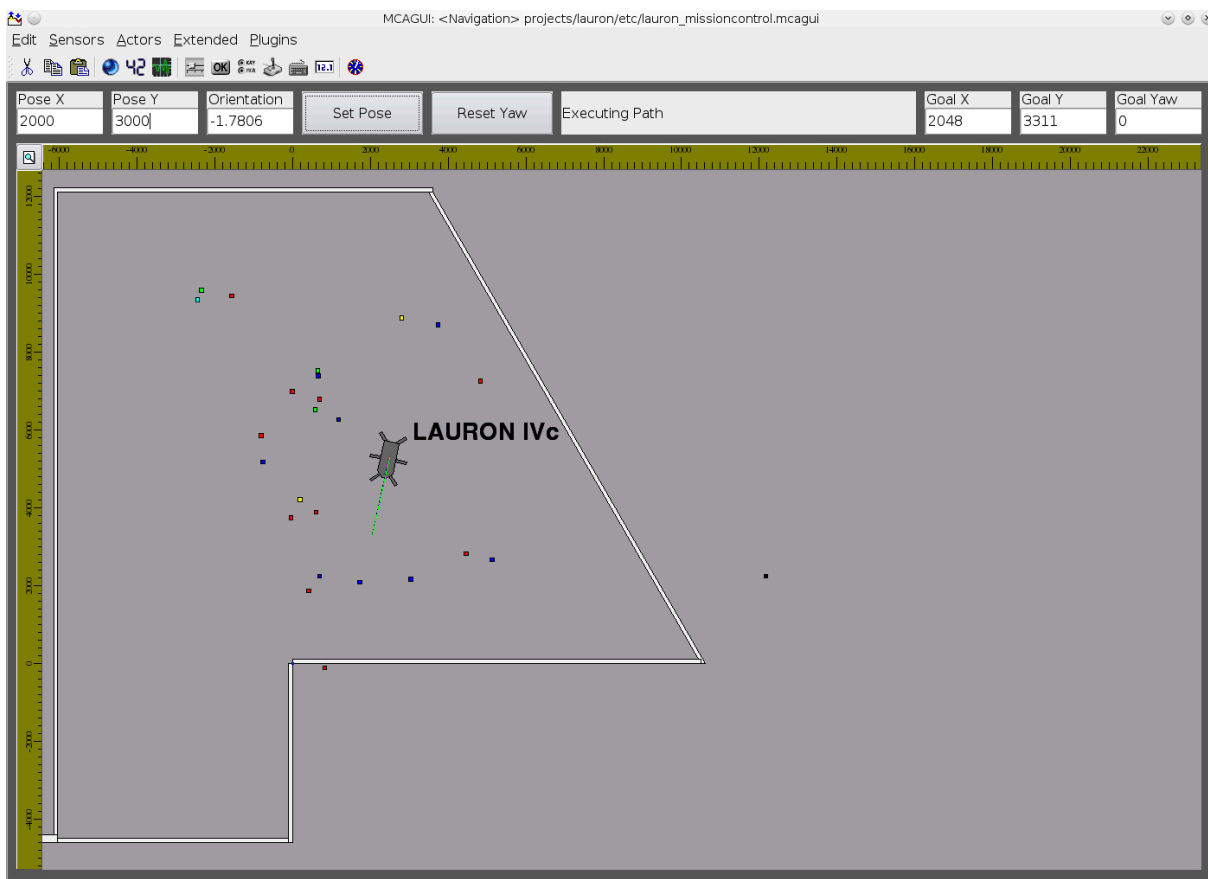


Abb. 8.24.: Interessierende Entitäten, die im Rahmen einer in einer Grünanlage durchgeführten Inspektionsmission gefunden wurden. Die einzelnen Entitätsklassen sind farblich kodiert dargestellt: Flasche (rot), Papier (grün), Plastiktüte (blau), Tetrapak (cyan) und Dose (gelb).

sich das im Rahmen dieser Arbeit entwickelte Konzept der semantischen Missionssteuerung mit verhältnismäßig geringem Aufwand auf andere Roboterplattformen übertragen lässt.

### 8.7. Zusammenfassung und Fazit

Im Mittelpunkt der Evaluierung standen zunächst die in den vier Teilbereichen dieser Arbeit entwickelten Konzepte: der Entwurf der Steuerungsarchitektur, der Entwurf der Wissensbasis, die Erzeugung und Ausführung von Inspektionsplänen sowie die aktive Untersuchung interessierender Entitäten. Die Evaluierung dieser Komponenten erfolgte im Wesentlichen mit Hilfe des Simulationsmodus der semantischen Missionssteuerung sowie einer speziell entwickelten Testautomatisierungskomponente, mit der unterschiedliche Inspektionssituationen systematisch erzeugt und abgeprüft werden können.

Die geeignete Abbildung des Regelkreises der autonomen Inspektion durch die Steuerungsarchitektur und die Realisierung der in Abschnitt 3.2.3 beschriebenen Abläufe konnten bereits zu einem frühen Zeitpunkt gezeigt werden, indem nach der Umsetzung der zentralen Komponenten zunächst der interaktive Modus für die einzelnen Ausführungskomponenten umgesetzt wurde.

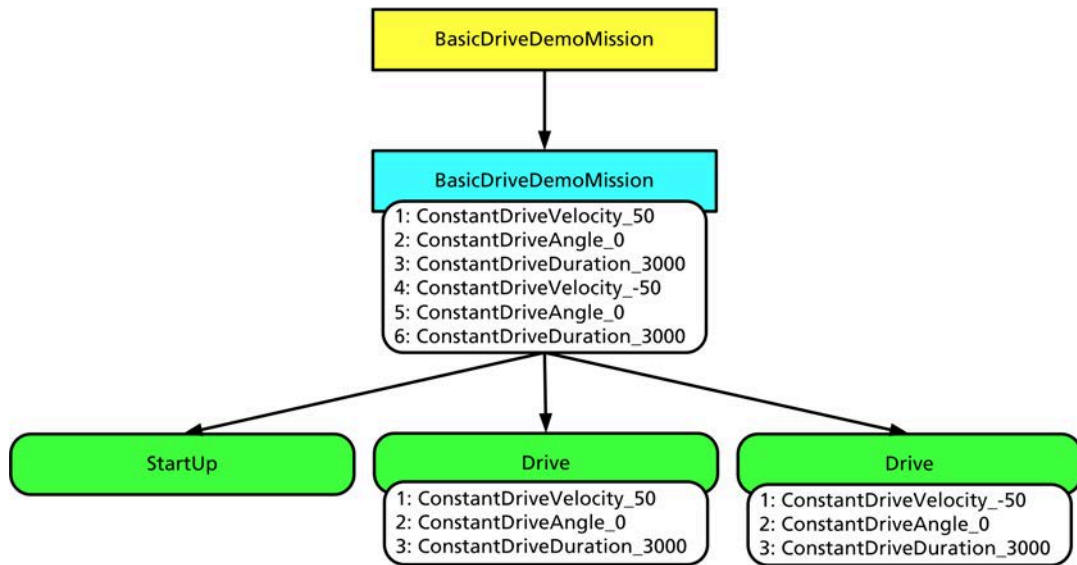


Abb. 8.25.: Demomission für KAIRO II in Form des generierten Flexiblen Hierarchischen Plans.

Die Fähigkeiten zur tatsächlichen Ausführung der jeweiligen elementaren Aufgaben wurden anschließend sukzessive ausgebaut. Begleitend wurden verschiedene Testfälle entwickelt, die schritt haltend eine automatisierte Überprüfung der Abläufe innerhalb der Steuerungsarchitektur ermöglichen.

Die Evaluierung der Wissensbasis gliederte sich in die zwei Teilbereiche Konsistenz und Korrektheit des spezifizierten Wissens. Die Konsistenz der Wissensbasis wird bei jedem Start der semantischen Missionssteuerung automatisch mit Hilfe des Reasoners KAON2 überprüft. Die Korrektheit der Wissensbasis wurde anhand von Kompetenzfragen (siehe Abschnitt 4.1.1) evaluiert. Hierzu wurde eine spezielle Bedienkomponente entwickelt, mit deren Hilfe sowohl die Kompetenzfragen als auch beliebige benutzerdefinierte SPARQL-Anfragen an die Wissensbasis gerichtet und die Ergebnisse überprüft werden können.

Die korrekte Erzeugung und Ausführung von Inspektionsplänen in Gestalt von Flexiblen Hierarchischen Plänen wurde in mehreren Schritten überprüft. Zunächst wurde das zur Bestimmung der zusammengefassten Planinformationen realisierte Verfahren anhand der für die Planungsdomäne Produktionsbetrieb in [27] gemachten Detailangaben erfolgreich verifiziert. Darüber hinaus wurde eine spezielle Bedienkomponente zur Visualisierung und Überwachung der generierten Pläne realisiert. Insgesamt wurden 19 verschiedene Planungsprobleme mit insgesamt 27 elementaren Aufgaben, 38 nichtelementaren Aufgaben sowie 59 Zerlegungsmethoden erfolgreich überprüft. Mit Hilfe des Simulationsmodus wurden zudem gezielt Ausführungsfehler und Ausnahmesituationen erzeugt. Die Reduzierung des Neuplanungsaufwands durch die Beibehaltung aller konsistenten Planverfeinerungen konnte auf diese Weise ebenfalls bestätigt werden.

Zur experimentellen Evaluierung der aktiven Untersuchung von interessierenden Entitäten

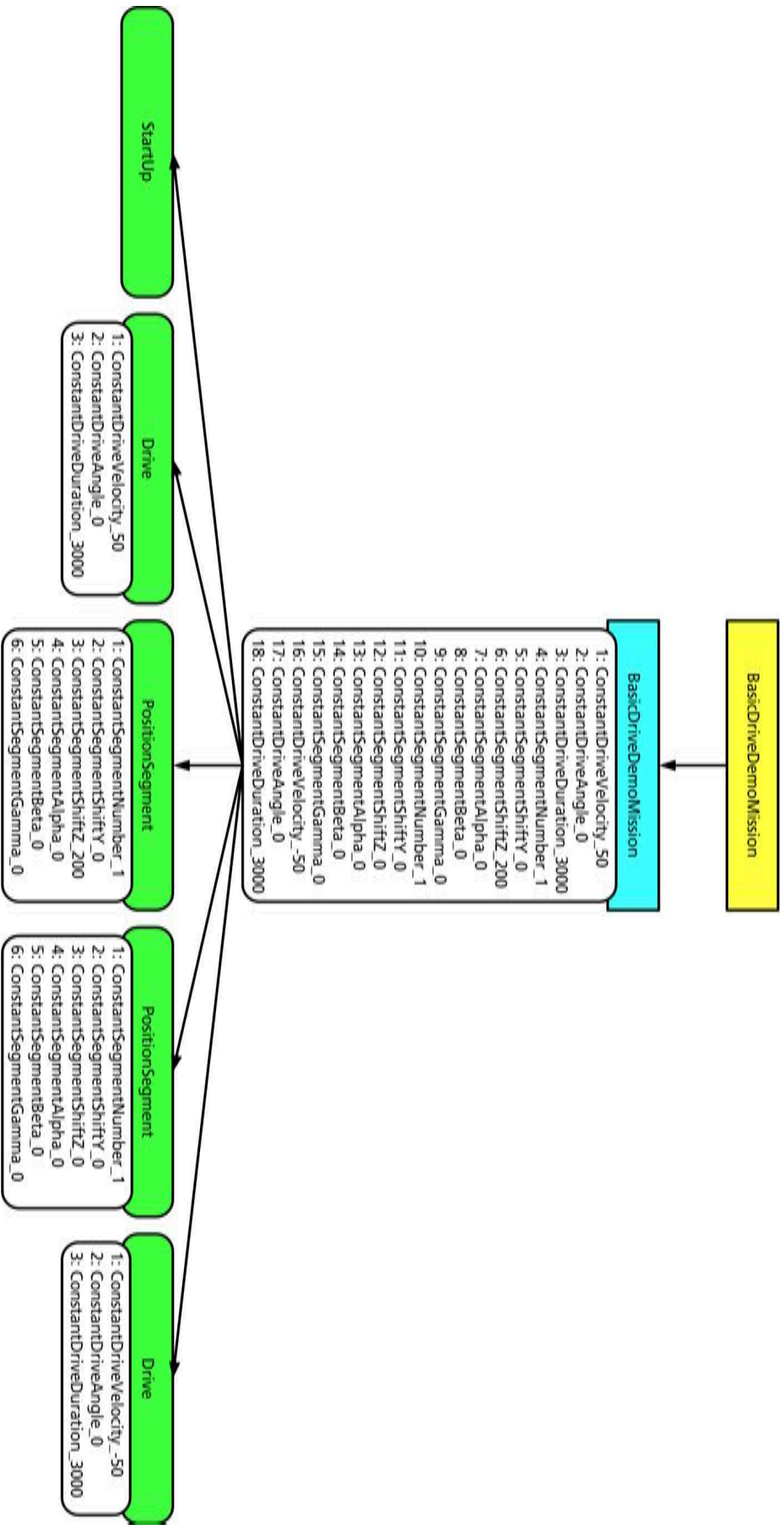


Abb. 8.26.: Erweiterte Demomission für KAIR0 II in Form des generierten Flexiblen Hierarchischen Plans.



wurde im Wesentlichen eine speziell entwickelte Testautomatisierungskomponente verwendet, mit deren Hilfe die Ergebnisse der Inspektionsaufgaben und die Werte der zugehörigen Einflussfaktoren systematisch variiert wurden. Auf diese Weise konnten die Fusion der Datenauswertungsergebnisse mit Hilfe des Bayes'schen Netzwerks, die Bewertung der fusionierten Ergebnisse anhand der Entropie-basierten Konfidenz sowie die situationsabhängige Auswahl und Priorisierung von Inspektionsaufgaben anhand der Regelbasis des semantischen Inspektionsmodells gezielt überprüft und erfolgreich validiert werden.

Im zweiten Teil der Evaluierung wurde auf die erweiterten Fähigkeiten von LAURON IV eingegangen, die eine wesentliche Grundlage für das im Rahmen dieser Arbeit gewählte Inspektionszenario bilden. Dies umfasste insbesondere die Fähigkeiten zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall.

Die Evaluierung der Fähigkeiten zur Erreichung vorgegebener Zielpunkte und Regionen sowie zum systematischen Ablaufen von Regionen erfolgte anhand von speziellen Benchmark-Umgebungen, die häufig zur Bewertung von Bahnplanungsverfahren herangezogen werden. Für die Planung von Pfaden zu Zielpunkten und Regionen mit Hilfe von RRTs wurden zwei unterschiedliche Problemstellungen betrachtet: Die Planung von Pfaden von einem Startpunkt zu genau einem Zielpunkt sowie die gleichzeitige Planung von Pfaden von einem Startpunkt zu fünfzehn gleichmäßig verteilten Zielpunkten. Dabei zeigte sich, dass die Anzahl der benötigten Explorationsschritte wie zu erwarten von der Komplexität der Benchmark-Umgebungen abhängig ist. Darüber hinaus wurde deutlich, dass das gewählte Verfahren nicht für lokale Minima anfällig ist und mit Hilfe von RRTs sehr effizient Pfade zu mehreren Zielen gleichzeitig geplant werden können. Die Laufzeiten für die Erzeugung der zielgerichteten RRTs lagen fast immer deutlich unter einer Sekunde. Das *Complete Coverage* Verfahren zur Planung von Pfaden zum Ablaufen von Regionen wurde anhand der Ergebnisse für die einzelnen Benchmark-Umgebungen qualitativ bewertet. Dabei zeigte sich, dass die *Distance Transform* Kostenfunktion in Bezug auf die beim Ablaufen der Pfade benötigten Drehungen des Roboters sowie die Anzahl und die Länge der doppelt abzulaufenden Pfadsegmente besser abschneidet als die *Path Transform* Kostenfunktion. Zur Evaluierung der gesamten globalen Navigation wurden mit LAURON IV sowohl unter Laborbedingungen als auch in unstrukturiertem Gelände so genannte Navigationsmissionen durchgeführt. Dabei wurden dem Roboter jeweils eine abzulaufende Region sowie ein Zielpunkt vorgegeben, zu dem der Roboter nach Ablaufen der Region zurückkehren sollte. Im Rahmen der einzelnen Navigationsmissionen konnte das nahtlose Zusammenwirken der verschiedenen Navigationsfähigkeiten erfolgreich demonstriert werden und es konnten jeweils geeignete Pfade zum Erreichen der vorgegebenen Region, zum systematischen Ablaufen der Region sowie zum Erreichen des vorgegebenen Zielpunkts generiert werden.

Die Qualität der entwickelten Verfahren zur Detektion und Klassifikation von Abfall wurde mit Hilfe von so genannten Detektionsmissionen untersucht. Dabei wurden dem Roboter unter

kontrollierten und nahezu identischen Randbedingungen verschiedene Objekte zur Erkennung vorgelegt. Unter Laborbedingungen wurde für jedes Objekt jeweils mindestens ein zugehöriger Bildbereich erfolgreich segmentiert. In unstrukturiertem Gelände wurden einige Abfallobjekte übersehen, wobei alle entsprechenden Szenen drei oder mehr verschiedene Abfallobjekte enthielten, von denen jeweils nur ein Objekt nicht detektiert wurde. Die Detektion weist somit insgesamt eine hohe Zuverlässigkeit auf. Die Korrektorklassifikationsraten für die Klassifikation Abfall / Kein Abfall liegen zwischen 81,3 % und 94,4 %. Das gewählte Verfahren erscheint somit prinzipiell geeignet. Die Korrektorklassifikationsraten für die Klassifikation der Abfallart liegen hingegen nur zwischen 26,2 % und 34,3 %. Eine rein texturbasierte Erkennung der Abfallart gestaltet sich somit schwierig. Zur erfolgreichen Bestimmung der Abfallart sind daher zusätzliche Sensormodalitäten hinzuzuziehen, die weiteren Aufschluss über die geometrischen, optischen und physikalischen Objekteigenschaften geben.

Zur Evaluierung des integrierten Gesamtsystems wurden so genannte Inspektionsmissionen durchgeführt. Dabei wurden dem System über die graphische Bedienoberfläche zu inspizierende Regionen vorgegeben. Anschließend waren von der semantischen Missionssteuerung autonome Pfade zum Erreichen und zum systematischen Absuchen der Regionen zu planen und auszuführen. Beim Auffinden von interessierenden Entitäten war die Suche zu unterbrechen und die jeweilige Entität solange aktiv zu untersuchen, bis eine ausreichende Konfidenz der Datenauswertungsergebnisse erreicht wurde. Abschließend war ein Pfad zu einer ebenfalls vorgegebenen Zielposition zu planen und auszuführen. Entsprechende Inspektionsmissionen wurden sowohl unter Laborbedingungen als auch in unstrukturiertem Gelände durchgeführt. Dabei konnte die Funktionsfähigkeit des Gesamtsystems erfolgreich demonstriert werden. Die Ergebnisse der Detektion und Klassifikation von Abfall fielen dabei ähnlich wie im Rahmen der Detektionsmissionen aus. Lediglich die Klassifikation von Bildbereichen, die Gras bzw. Wiese enthielten, führte zu einer erhöhten Anzahl an falsch Positiven. Dieser Effekt sollte sich jedoch durch eine entsprechende Erweiterung der Lerndaten und erneutes Trainieren der Klassifikatoren deutlich reduzieren lassen. Die Positionsbestimmung der interessierenden Entitäten besitzt aufgrund von Ungenauigkeiten der Odometrie ebenfalls noch Optimierungspotential. Die im Rahmen dieser Arbeit entwickelten Kernkomponenten der semantischen Missionssteuerung funktionierten hingegen wie gewünscht und bieten somit ein leistungsfähiges und flexibles System für die aktive autonome Inspektion.

Die Übertragbarkeit der semantischen Missionssteuerung auf andere Roboterplattformen und Inspektionsdomänen wurde anhand der beispielhaften Portierung auf den mehrsegmentigen Inspektionsroboter KAIRO II untersucht. Hierzu wurde zunächst der strukturelle Aufbau der Steuerungsarchitektur und der Wissensbasis übertragen, wofür 203 Minuten benötigt wurden. Daran anschließend wurde exemplarisch eine einfache Mission zur Demonstration von einigen grundlegenden Fähigkeiten des Roboters realisiert. Diese umfasste zunächst eine elementare

Aufgabe zur Initialisierung des Roboters und eine elementare Aufgabe, welche dem Manöver *Freie Fahrt* (siehe [12]) entspricht. Der hierfür benötigte Zeitaufwand betrug 77 Minuten. Für die anschließende Erweiterung dieser Mission um das Manöver *Positioniere Modul* wurden 56 Minuten benötigt. Die angegebenen Zeiten vermitteln einen ersten Eindruck für die benötigten Aufwände zur Portierung und zeigen, dass sich die semantische Missionssteuerung mit verhältnismäßig geringem Aufwand auf andere Roboterplattformen übertragen lässt.

Insgesamt lässt sich somit feststellen, dass die Eignung und Funktionsfähigkeit der entwickelten semantischen Missionssteuerung für autonome Inspektionsroboter anhand der durchgeführten Experimente erfolgreich gezeigt werden konnte. Aufgrund des semantischen Ansatzes, der alle wichtigen Informationen in einer von allen Systemkomponenten genutzten Wissensbasis vereint, ist das System sehr flexibel einsetzbar und leicht erweiterbar. Ein menschlicher Operator kann durch Anpassung der Wissensbasis leicht das Verhalten des Systems in verschiedenen Inspektionssituationen konfigurieren und benötigt hierfür keinerlei Programmierkenntnisse. Aufgrund des gewählten Ansatzes für die Regelbasis ist auch eine autonome und schritthaltende Anpassung an die während der Inspektionen gesammelten Erfahrungen prinzipiell möglich. Wie am Beispiel des mehrsegmentigen Kanalroboters KAIRO II gezeigt wurde, ist die entwickelte semantische Missionssteuerung zudem mit verhältnismäßig geringem Aufwand auf andere Roboterplattformen und Inspektionsdomänen übertragbar.



## 9. Zusammenfassung

Die Inspektion von komplexen Umgebungen, wie zum Beispiel technischen Anlagen in Form von Abwasserkanälen, Pipelines, Kernkraftwerken, Hochspannungsleitungen oder Dämmen, mit autonomen Servicerobotern ist eine sehr anspruchsvolle Aufgabe. Bestehende Ansätze zur Entwicklung von autonomen Inspektionsrobotern lassen sich im Wesentlichen in zwei Bereiche gliedern. Diese umfassen zum einen die Hardwareentwicklung und die Bewegungssteuerung und zum anderen die Entwicklung geeigneter Sensorsysteme, die automatische Platzierung der Sensoren und die Sensordatenauswertung. Es existieren jedoch nur sehr wenige integrierte Ansätze. Insbesondere findet die Bewertung der Datenauswertungsergebnisse an Bord des Roboters bisher kaum Beachtung. Diese bildet jedoch eine wesentliche Voraussetzung für die aktive Untersuchung interessierender Entitäten und das Treffen situationsabhängiger Entscheidungen.

Ziel dieser Arbeit war die Entwicklung einer semantischen Missionssteuerung für autonome Inspektionsroboter, welche es ermöglicht, den Regelkreis bestehend aus Inspektionsplanung, Planausführung, Inspektionsdatenauswertung, Bewertung der Datenauswertungsergebnisse, Entscheidungsfindung und Neuplanung an Bord des Roboters zu schließen. Für die Auswertung und Beurteilung der vielfach komplexen Sensordaten wird von menschlichen Datenauswertungsexperten in der Regel auf umfangreiche Fachkenntnisse und Erfahrungen zurückgegriffen. In dieser Arbeit wurde daher ein wissensbasierter Ansatz untersucht, welcher dem autonomen Inspektionssystem das menschliche Expertenwissen in Form eines *semantischen Inspektionsmodells* zur Verfügung stellt.

Der wissenschaftliche Beitrag dieser Arbeit besteht in einem integrierten, durchgängig ontologiebasierten Ansatz, welcher zum einen die Erzeugung und Ausführung von Inspektionsplänen und zum anderen die Bewertung der Datenauswertungsergebnisse sowie die darauf basierende situationsabhängige Auswahl und Priorisierung von Inspektionsaufgaben umfasst.

Die im Rahmen dieser Arbeit betrachteten Fragestellungen und die erarbeiteten Lösungen gliedern sich in die im Folgenden kurz zusammengefassten vier Teilbereiche.

**Entwurf der Steuerungsarchitektur** Zur Realisierung des Regelkreises bestehend aus Inspektionsplanung, Planausführung, Inspektionsdatenauswertung, Bewertung der Datenauswertungsergebnisse, Entscheidungsfindung und Neuplanung bedarf es einer geeigneten Steuerungsarchitektur. Hier wurde ein modularer, hierarchischer Ansatz gewählt, der die einzelnen Komponenten der Missionssteuerung auf vier Ebenen anordnet. Die einzelnen Ebenen definieren sich anhand der Art der verarbeiteten Daten. Auf der untersten, der subsymbolischen Ebene befinden

sich *Sensor-* und *Aktorinterface* zum Robotersystem. Auf der darüber liegenden subsymbolisch-symbolischen Ebene befinden sich die *Inspektions-* und die *Navigationsdatenauswertung* sowie die *Basissteuerung*. Die symbolisch-semantische Ebene umfasst die *Semantische Inspektion*, die *Semantische Navigation*, die *Semantische Kartierung* sowie die *Ausführungseinheit*. Die oberste, semantische Ebene enthält den *Manager* und die *Bedienschnittstelle*. Zentrales Element der Missionssteuerung ist jedoch die *Wissensbasis*, welche das für die Durchführung von autonomen Inspektionsmissionen benötigte Wissen enthält.

**Entwurf der Wissensbasis** Zur expliziten Repräsentation des für die Durchführung von autonomen Inspektionsmissionen notwendigen Experten- und Hintergrundwissens bedarf es einer geeigneten Wissensbasis. Hier wurde eine modulare, auf Beschreibungslogiken basierende Wissensbasis entwickelt, welche in drei Abstraktionsebenen unterteilt ist: eine Basisontologie, eine Kernontologie und eine Domänenontologie. Die Basisontologie umfasst allgemeine, grundlegende Konzepte und Relationen. Die Kernontologie enthält die Unterontologien *Roboter*, *Umgebung*, *Inspektion*, *Navigation* und *Mission*, welche die zentralen Konzepte und Relationen des jeweiligen Themenfelds modellieren. Die Domänenontologie enthält die applikationsspezifischen Unterontologien für den konkreten Inspektionsroboter, die konkret zu inspizierende Umgebung sowie die konkret aufzufindenden interessierender Entitäten.

**Erzeugung und Ausführung von Inspektionsplänen** Für die Erzeugung und Ausführung von Inspektionsplänen werden geeignete Verfahren benötigt, die zur Durchführung von autonomen Inspektionsmissionen geeignet sind. Zur Repräsentation von Inspektionsplänen wurden so genannte *Flexible Hierarchische Pläne* (FHiP) entworfen, welche an Hierarchische Aufgabennetzwerke (engl. *Hierarchical Task Networks* (HTN)) angelehnt sind. Für die Inspektionsplanung wurde ein ontologiebasiertes HTN-Planungsverfahren entwickelt, welches eine Menge komplexer Inspektionsaufgaben in Abhängigkeit von der aktuellen Situation rekursiv in einfachere Unteraufgaben zerlegt, bis diese elementar gelöst werden können. Dabei werden potentielle Konflikte zwischen den einzelnen Teilplänen bereits auf abstrakten Ebenen mit Hilfe von zusammengefassten Planinformationen aufgelöst. Das für die Zerlegung notwendige Wissen ist in Form der verfügbaren Aufgaben und Zerlegungsmethoden in der Wissensbasis gespeichert. Während der Ausführung werden die Flexiblen Hierarchischen Pläne gemäß einer Tiefensuchstrategie abgearbeitet.

**Aktive Untersuchung interessierender Entitäten** Um eine aktive Untersuchung interessierender Entitäten zu ermöglichen, werden Methoden und Verfahren zur Fusion und Bewertung der Datenauswertungsergebnisse, zur Auswahl und Priorisierung von zusätzlichen Inspektionsaufgaben sowie zur Berücksichtigung dieser Aufgaben mittels Adaption der bestehenden

---

Inspektionspläne benötigt. Für die Fusion der Datenauswertungsergebnisse wurde ein generisches, auf Bayes'schen Netzwerken basierendes Verfahren entwickelt. Anhand eines Entropie-basierten Konfidenzmaßes und der Regelbasis des semantischen Inspektionsmodells wird anschließend entschieden, ob und wenn ja wie die gefundenen interessierenden Entitäten weiter untersucht werden. Die hieraus resultierenden neuen Inspektionsaufgaben werden mit Hilfe einer regelbasierten Heuristik priorisiert und durch iterative Neuplanung in den bestehenden Missionsplan integriert.

Die im Rahmen dieser Arbeit entwickelte semantische Missionssteuerung wurde für die sechsbeinige Laufmaschine LAURON IV umgesetzt. Als konkretes Inspektionsszenario wurde das Auffinden von Abfall in unstrukturiertem Gelände, wie zum Beispiel Wiesen, Wäldern und Böschungen von vielbefahrenen Straßen, Flüssen und Kanälen, untersucht. Für die Realisierung dieses Szenarios wurde LAURON IV mit weiteren Fähigkeiten ausgestattet. Dies umfasste insbesondere Fähigkeiten zur globalen Navigation sowie zur bildbasierte Detektion und Klassifikation von Abfall. Für die Erreichung von vorgegebenen Zielpunkten und Regionen wurde ein auf *Rapidly-Exploring Random Trees* (RRTs) basierendes Verfahren entwickelt, das auch Pfade zu mehreren Zielen gleichzeitig planen kann. Die Pfadplanung zum systematischen Absuchen von Regionen wurde mit Hilfe eines numerischen Potentialfeldverfahrens realisiert. Für die bildbasierte Erkennung von Abfall wurde ein aufmerksamkeitsbasierter Ansatz gewählt, bei dem zunächst so genannte Aufmerksamkeitskarten (engl. *saliency maps*) berechnet werden. Die auf diese Weise gewonnenen Aufmerksamkeitsbereiche werden anschließend mittels eines merkmalsbasierten Klassifikationsansatzes auf das Vorhandensein von Abfallobjekten überprüft. Als Merkmale werden Farbverbundhistogramme (engl. *Color Cooccurrence Histogram* (CCH)), Lokale Binäre Muster (engl. *Local Binary Patterns* (LBP)) sowie lokal invariante SURF-Deskriptoren (engl. *Speeded-Up Robust Features* (SURF)) verwendet. Die Klassifikation wird mittels Support-Vektor-Maschinen (SVM) durchgeführt.

Die Evaluierung der semantischen Missionssteuerung gliederte sich in vier Teilbereiche. Im ersten Bereich wurden die Funktionsfähigkeit und das Verhalten der Kernkomponenten der semantischen Missionssteuerung in relevanten Inspektionssituationen untersucht. Die Evaluierung erfolgte dabei im Wesentlichen mit Hilfe des Simulationsmodus der semantischen Missionssteuerung sowie einer speziell entwickelten Testautomatisierungskomponente, mit der systematisch verschiedene Inspektionssituationen erzeugt wurden. Zu den Evaluationskriterien gehörten unter anderem die Funktionsfähigkeit des Regelkreises der autonomen Inspektion, die Konsistenz und Korrektheit des in der Wissensbasis spezifizierten Wissens, die korrekte Fusion und Bewertung der Datenauswertungsergebnisse sowie die darauf basierende situationsabhängige Auswahl und Priorisierung neuer Inspektionsaufgaben. Im zweiten Bereich wurden die erweiterten Fähigkeiten von LAURON IV evaluiert, da diese eine wesentliche Grundlage für

das gewählte Inspektionsszenario bilden. Dies umfasste insbesondere die Fähigkeiten zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall. Die globale Navigation wurde mit Hilfe von verschiedenen Benchmark-Umgebungen untersucht, die häufig zur Bewertung von Bahnplanungsverfahren herangezogen werden. Für die Evaluierung der Detektion und Klassifikation von Abfall wurden entsprechende Validierungsdaten von verschiedenen Abfallobjekten und natürlichen Objekten eingesetzt. Daran anschließend wurde im dritten Bereich das integrierte Gesamtsystem mit allen Fähigkeiten sowohl unter kontrollierten Laborbedingungen als auch in unstrukturiertem Gelände erprobt und evaluiert. Hierzu wurden verschiedene Inspektionsmissionen durchgeführt, in denen vom menschlichen Operator vorgegebene Regionen autonom nach Abfall abgesucht wurden. Abschließend wurde im vierten Bereich am Beispiel des mehrsegmentigen Inspektionsroboters KAIRO II demonstriert, dass sich die semantische Missionssteuerung mit verhältnismäßig geringem Aufwand auch auf andere Roboterplattformen übertragen und für andere Inspektionsdomänen einsetzen lässt.

### 9.1. Wissenschaftliche Erkenntnisse der Arbeit

Anhand der durchgeführten Experimente konnten die Funktionsfähigkeit der semantischen Missionssteuerung und ihre Eignung für die autonome Durchführung von Inspektionsmissionen erfolgreich gezeigt werden. Die Erprobung des integrierten Gesamtsystems mit der sechsbeinigen Laufmaschine LAURON IV hat bestätigt, dass die Inspektion von komplexen Umgebungen mit Servicerobotern autonom plan- und durchführbar ist. Durch die Wahl eines semantischen Ansatzes mit einer zentralen Wissensbasis, auf die von allen Systemkomponenten aus zugegriffen wird, konnte außerdem gezeigt werden, wie das für die Inspektion von komplexen Umgebungen notwendige Experten- und Hintergrundwissen einem autonomen System mit Hilfe eines semantischen Inspektionsmodells zur Verfügung gestellt werden kann. Mit Hilfe der Regelbasis des semantischen Inspektionsmodells ist es dem System darüber hinaus möglich, die Konfidenz der Datenauswertungsergebnisse zu beurteilen und in Abhängigkeit von der jeweiligen Inspektionssituation zu entscheiden, wie gefundene interessierende Entitäten aktiv weiter untersucht werden. Dies führt zu einem autonomeren und intelligenteren Systemverhalten. Durch die Bündelung aller notwendigen Informationen in einer zentralen Wissensbasis sowie durch die Verwendung einer klar definierten Semantik wird darüber hinaus die Transparenz für den Benutzer erhöht. Er kann sich zu jeder Zeit einen Überblick über die verfügbaren Informationen sowie Regeln zur Beurteilung und Entscheidungsfindung verschaffen und so die vom System getroffenen Entscheidungen einfacher nachvollziehen. Da nur an einer zentralen Stelle Änderungen vorgenommen werden müssen, um das Verhalten des Systems anzupassen, wird auch die Wartbarkeit verbessert. Der gewählte modulare Aufbau der Wissensbasis und der Steuerungsarchitektur resultiert zudem in einer einfacheren Erweiterbarkeit und einer erhöhten Flexibilität des Systems, wie anhand der beispielhaften Portierung der semantischen Missionssteuerung auf



den mehrsegmentigen Inspektionsroboter KAIRO II gezeigt wurde.

## 9.2. Ausblick

Wie im Rahmen dieser Arbeit gezeigt wurde, bildet die entwickelte semantische Missionssteuerung ein tragfähiges Konzept für die autonome Inspektion mit Servicerobotern. Die Gesamtqualität eines solchen Systems wird jedoch auch maßgeblich durch die realisierten Fähigkeiten mitbestimmt. Die im Hinblick auf das gewählte Inspektionsszenario realisierten erweiterten Fähigkeiten von LAURON IV zur globalen Navigation sowie zur Detektion und Klassifikation von Abfall bilden hier sicherlich nur einen Anfang. Einige der in diesem Bereich möglichen Weiterentwicklungen wurden bereits in Kapitel 8 adressiert. Sie umfassen etwa den Ausbau der Fähigkeiten zur Bestimmung geometrischer, optischer und physikalischer Eigenschaften von detektierten Objekten, um eine sicherere Unterscheidung der verschiedenen Abfallarten zu gewährleisten. Darüber hinaus bildet die Verbesserung der Fähigkeiten zur Selbstlokalisierung und zur Positionsbestimmung von detektierten Objekten eine wesentliche Voraussetzung für die erfolgreiche Weiterentwicklung des prototypisch umgesetzten Systems hin zu einem Produktivsystem.

Richtet man den Blick auf die im Rahmen dieser Arbeit entwickelten Kernkomponenten der semantischen Missionssteuerung, so bieten sich auch hier sehr interessante Möglichkeiten, das entwickelte Konzept weiter auszubauen. An erster Stelle ist dabei sicherlich die volle Integration der Komponenten zur semantischen Navigation zu nennen, die im Rahmen einer parallel laufenden Arbeit entwickelt werden (siehe [101]). Auch die in Abschnitt 5.6 nur kurz angesprochene synergetische Vernetzung von Planung und Inferenz sollte Gegenstand weiterer Untersuchungen in diesem Bereich sein, da sie erhebliches Potential verspricht. Darüber hinaus wurde das autonome Lernen von neuem Wissen anhand der gemachten Erfahrungen des Systems in Abschnitt 6.5 nur im Hinblick auf die kontinuierliche Anpassung der Regelbasis thematisiert. Hier bieten sich jedoch eine Reihe weiterer Ansatzpunkte, wie zum Beispiel das selbständige Lernen neuer Objektkategorien. Einen weiteren Themenkomplex, der im Rahmen von zukünftigen Forschungsarbeiten untersucht werden sollte, bildet die explizite Repräsentation des internen Zustands des Roboters im Hinblick auf die autonome Fehlerdiagnose und -behebung, die Selbstbewertung des Systemverhaltens sowie die dynamische Selbstkonfiguration des Systems. Auch die situationsabhängige Optimierung der lokalen Parametrisierung von Aufgaben ist ein vielversprechendes Thema für weitere Forschungsarbeiten.

Die hier genannten Themen für zukünftige Arbeiten in diesem Bereich lassen sich sicherlich noch weiter fortsetzen. Festzuhalten bleibt, dass die Verwendung von semantischem Wissen in der Robotik gegenwärtig ein sehr reges Forschungsfeld ist und insbesondere für die aktive Inspektion von komplexen Umgebungen mit autonomen Servicerobotern ein enormes Potential bietet, welches im Rahmen dieser Arbeit sicherlich noch nicht voll ausgeschöpft wurde.



## A. Algorithmen für die nebenläufige hierarchische Koordination

In diesem Anhang werden die Algorithmen zur Bestimmung der *kann*- und *muss*-Beziehungen angegeben, welche für die in Abschnitt 5.3.4 beschriebene nebenläufige hierarchische Koordination benötigt werden. Eine ausführliche Erläuterung dieser Algorithmen findet sich in [28] und [27].

### A.1. Muss Erreichen

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen eingespart werden.

---

**Algorithmus A.1** Muss Erreichen (must-achieve)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , Ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow l(c) \wedge muss(c')$  then
      if  $c \in pre_{sum}(p) \wedge p'$  muss zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if  $(p'$  kann zusichern  $c'$  vor  $c'' \wedge p''$  kann zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow \neg l(c)) \vee (p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow l(c) \wedge muss(c''))$  then
            dazwischen_liegende_zusicherung = wahr
          end if
        end for
      end if
    end for
  end if
end for
return falsch
```

---

## A.2. Muss Überschreiben

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen eingespart werden.

---

### Algorithmus A.2 Muss Überschreiben (must-clobber)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow \neg l(c) \wedge muss(c')$  then
      if  $c \in in_{sum}(p) \wedge p'$  muss zusichern  $c'$  in  $c$  then
        return wahr
      end if
      if  $c \in post_{sum}(p) \wedge p'$  muss zusichern  $c'$  während  $c$  then
        return wahr
      end if
      if  $c \in pre_{sum}(p) \wedge p'$  muss zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if ( $p'$  kann zusichern  $c'$  vor  $c'' \wedge p''$  kann zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow l(c)$ )  $\vee$ 
            ( $p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow \neg l(c) \wedge muss(c'')$ ) then
              dazwischen_liegende_zusicherung = wahr
            end if
          end for
        end for
        if  $\neg$  dazwischen_liegende_zusicherung then
          return wahr
        end if
      end if
    end if
  end for
end for
return falsch
```

---

### A.3. Muss Zurücksetzen

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen eingespart werden.

---

**Algorithmus A.3** Muss Zurücksetzen (must-undo)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , Ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow \neg l(c) \wedge muss(c')$  then
      if  $c \in post_{sum}(p) \wedge p'$  muss zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if  $(p'$  kann zusichern  $c'$  vor  $c'' \wedge p''$  kann zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow l(c)) \vee (p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow \neg l(c) \wedge muss(c''))$  then
            dazwischen_liegende_zusicherung = wahr
          end if
        end for
      if  $\neg$  dazwischen_liegende_zusicherung then
        return wahr
      end if
    end if
  end for
end for
return falsch
```

---

#### A.4. Kann Erreichen

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen eingespart werden.

---

**Algorithmus A.4** Kann Erreichen (may-achieve)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , Ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow l(c)$  then
      if  $c \in pre_{sum}(p) \wedge p'$  kann zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if ( $p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow (l(c) \vee \neg l(c)) \wedge muss(c'')$ ) then
            dazwischen_liegende_zusicherung = wahr
          end if
        end for
      end if
    end if
  end if
end for
return falsch
```

---

## A.5. Kann Überschreiben

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen eingespart werden.

---

### Algorithmus A.5 Kann Überschreiben (may-clobber)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , Ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow \neg l(c)$  then
      if  $c \in in_{sum}(p) \wedge p'$  kann zusichern  $c'$  in  $c$  then
        return wahr
      end if
      if  $c \in post_{sum}(p) \wedge p'$  kann zusichern  $c'$  während  $c$  then
        return wahr
      end if
      if  $c \in pre_{sum}(p) \wedge p'$  kann zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if ( $p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow (l(c) \vee \neg l(c)) \wedge muss(c'')$ ) then
            dazwischen_liegende_zusicherung = wahr
          end if
        end for
        if  $\neg$  dazwischen_liegende_zusicherung then
          return wahr
        end if
      end if
    end if
  end if
end for
return falsch
```

---

## A.6. Kann Zurücksetzen

Der im Folgenden angegebene Algorithmus stammt aus [28]. Er wurde jedoch dahingehend modifiziert, dass über alle  $p \in P_{sum}$  iteriert wird, wodurch externe Schleifen ein gespart werden.

---

**Algorithmus A.6** Kann Zurücksetzen (may-undo)

---

**Eingabe:** zusammengefasste Bedingung  $c$  des Plans  $p$ ,  $P_{sum}$ , Ordnungen

**Ausgabe:** wahr oder falsch

```
for all  $p'_{sum} \in P_{sum}$  do
  for all  $c' \in (in_{sum}(p') \cup post_{sum}(p'))$  do
    if  $l(c') \Leftrightarrow \neg l(c)$  then
      if  $c \in post_{sum}(p) \wedge p'$  muss zusichern  $c'$  durch  $c$  then
        dazwischen_liegende_zusicherung = falsch
        for all  $c'' \in in_{sum}(p'') \cup post_{sum}(p''), p'' \in P_{sum}$  solange dazwischen_liegende_zusicherung = falsch do
          if  $p'$  muss zusichern  $c'$  vor  $c'' \wedge p''$  muss zusichern  $c''$  durch  $c \wedge l(c'') \Leftrightarrow (\neg l(c) \vee l(c)) \wedge muss(c'')$  then
            dazwischen_liegende_zusicherung = wahr
          end if
        end for
      end if
    end if
  end if
end for
return falsch
```

---



## B. Bilddaten zur Detektion und Klassifikation von Abfall

In diesem Anhang werden die unter Laborbedingungen aufgenommenen Bilddaten aufgeführt, die zum Trainieren der SVM-Klassifikatoren und zur Evaluierung verwendet wurden. Zur Gewinnung der Aufnahmen wurden LAURON IV unter nahezu identischen Randbedingungen verschiedene Objekte vorgelegt.

### B.1. Lerndaten

Dieser Abschnitt enthält die zum Trainieren der SVM-Klassifikatoren verwendeten Bilddaten (siehe Abschnitt 7.3.2). Die im Folgenden angegebenen Objektbezeichner setzen sich jeweils aus einem Klassennamen und einem Objektnamen zusammen, welche durch einen Doppelpunkt voneinander getrennt sind.

#### B.1.1. Abfall



Abb. B.1.: Lerndatenobjekt *Flasche:Aldi*.



Abb. B.2.: Lerndatenobjekt *Flasche:AlteRebe*.



Abb. B.3.: Lerndatenobjekt *Flasche:Bionade*.



Abb. B.4.: Lerndatenobjekt *Flasche:BlackForest*.



Abb. B.5.: Lerndatenobjekt *Flasche:Apfelsaftschorle*.



Abb. B.6.: Lerndatenobjekt *Flasche:ColaPET*.



Abb. B.7.: Lerndatenobjekt *Flasche:ColaZero*.



Abb. B.8.: Lerndatenobjekt *Flasche:KumpfOrangensaft*.



Abb. B.9.: Lerndatenobjekt *Flasche:KurparkMedium*.



Abb. B.10.: Lerndatenobjekt *Flasche:MezzoMix*.



Abb. B.11.: Lerndatenobjekt *Flasche:NesteaZitrone*.



Abb. B.12.: Lerndatenobjekt *Flasche:Zäpfle*.



Abb. B.13.: Lerndatenobjekt *Papier:Baguette*.



Abb. B.14.: Lerndatenobjekt *Papier:Balisto*.



Abb. B.15.: Lerndatenobjekt *Papier:Express*.



Abb. B.16.: Lerndatenobjekt *Papier:Gauloises*.



Abb. B.17.: Lerndatenobjekt *Papier:Goldring*.



Abb. B.18.: Lerndatenobjekt *Papier:L&M*.



Abb. B.19.: Lerndatenobjekt *Papier:Marlboro*.



Abb. B.20.: Lerndatenobjekt *Papier:MarlboroDuhani*.

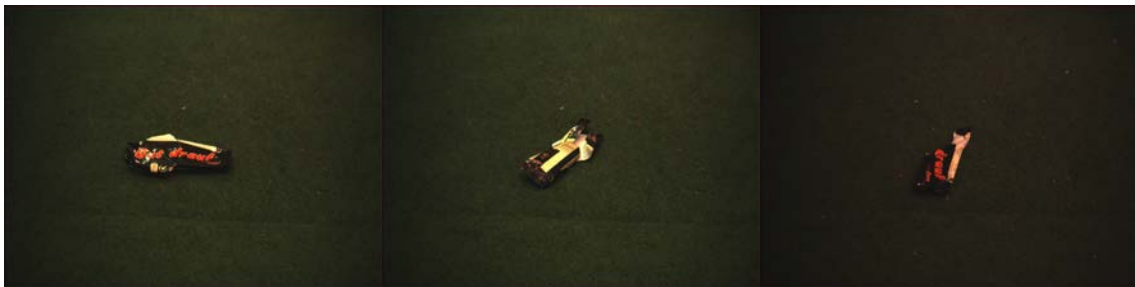


Abb. B.21.: Lerndatenobjekt *Papier:Mars*.



Abb. B.22.: Lerndatenobjekt *Papier:McDonalds*.



Abb. B.23.: Lerndatenobjekt *Papier:MrBaker*.



Abb. B.24.: Lerndatenobjekt *Papier:Neff*.



Abb. B.25.: Lerndatenobjekt *Papier:Visel*.



Abb. B.26.: Lerndatenobjekt *Plastiktüte:Aldi*.



Abb. B.27.: Lerndatenobjekt *Plastiktüte: Buchhandlung.*



Abb. B.28.: Lerndatenobjekt *Plastiktüte: Esprit Celebration.*



Abb. B.29.: Lerndatenobjekt *Plastiktüte: QuaxiFröschli.*



Abb. B.30.: Lerndatenobjekt *Plastiktüte: Rewe.*





Abb. B.31.: Lerndatenobjekt *Plastiktüte:SaftGoldbären.*



Abb. B.32.: Lerndatenobjekt *Plastiktüte:Thalia.*



Abb. B.33.: Lerndatenobjekt *Plastiktüte:Weiß.*



Abb. B.34.: Lerndatenobjekt *Tetrapak:Breisgau.*



Abb. B.35.: Lerndatenobjekt *Tetrapak:MangoMaracuja*.



Abb. B.36.: Lerndatenobjekt *Tetrapak:Milfina*.



Abb. B.37.: Lerndatenobjekt *Tetrapak:Milch*.



Abb. B.38.: Lerndatenobjekt *Tetrapak:OrangenT*.



Abb. B.39.: Lerndatenobjekt *Tetrapak:PfirsichT.*



Abb. B.40.: Lerndatenobjekt *Tetrapak:WildbeerenT.*



Abb. B.41.: Lerndatenobjekt *Tetrapak:ZitronenT.*



Abb. B.42.: Lerndatenobjekt *Dose:AfriCola.*



Abb. B.43.: Lerndatenobjekt *Dose:Cashews*.



Abb. B.44.: Lerndatenobjekt *Dose:Cola*.



Abb. B.45.: Lerndatenobjekt *Dose:ColaLight*.



Abb. B.46.: Lerndatenobjekt *Dose:Fanta*.



Abb. B.47.: Lerndatenobjekt *Dose:MezzoMix*.



Abb. B.48.: Lerndatenobjekt *Dose:Mildessa*.



Abb. B.49.: Lerndatenobjekt *Dose:Rotkohl*.



Abb. B.50.: Lerndatenobjekt *Dose:Pringles*.

## B.1.2. Kein Abfall



Abb. B.51.: Lerndatenobjekt *KeinAbfall:FlacherStein1.*



Abb. B.52.: Lerndatenobjekt *KeinAbfall:FlacherStein2.*



Abb. B.53.: Lerndatenobjekt *KeinAbfall:FlacherStein3.*



Abb. B.54.: Lerndatenobjekt *KeinAbfall:FlacherStein4.*



Abb. B.55.: Lerndatenobjekt *KeinAbfall:FlacherStein5*.



Abb. B.56.: Lerndatenobjekt *KeinAbfall:FlacherStein6*.



Abb. B.57.: Lerndatenobjekt *KeinAbfall:FlacherStein7*.



Abb. B.58.: Lerndatenobjekt *KeinAbfall:Holzstück1*.

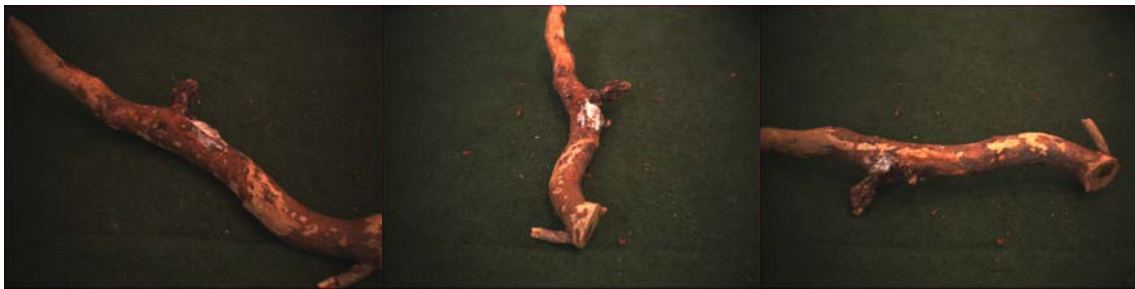


Abb. B.59.: Lerndatenobjekt *KeinAbfall:Holzstück2*.



Abb. B.60.: Lerndatenobjekt *KeinAbfall:Holzstück3*.





Abb. B.61.: Lerndatenobjekt *KeinAbfall:Holzstück4*.



Abb. B.62.: Lerndatenobjekt *KeinAbfall:Holzstück5*.

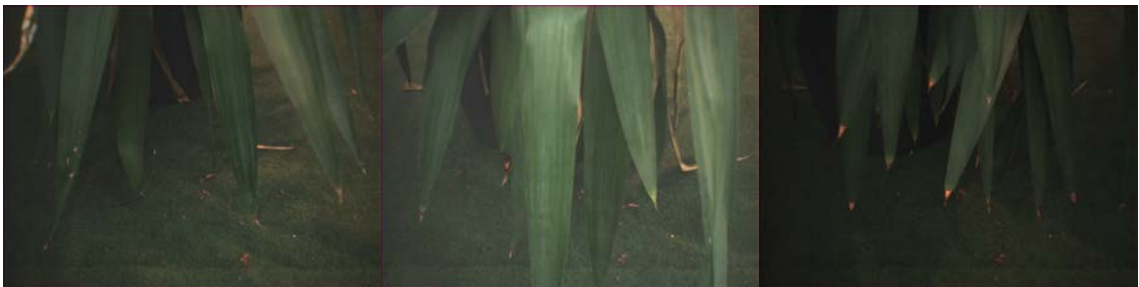


Abb. B.63.: Lerndatenobjekt *KeinAbfall:Pflanze1*.



Abb. B.64.: Lerndatenobjekt *KeinAbfall:Pflanze2*.



Abb. B.65.: Lerndatenobjekt *KeinAbfall:Felsen1.*



Abb. B.66.: Lerndatenobjekt *KeinAbfall:Felsen2.*



Abb. B.67.: Lerndatenobjekt *KeinAbfall:Felsen3.*



Abb. B.68.: Lerndatenobjekt *KeinAbfall:RunderStein1.*



Abb. B.69.: Lerndatenobjekt *KeinAbfall:RunderStein2.*



Abb. B.70.: Lerndatenobjekt *KeinAbfall:RunderStein3.*



Abb. B.71.: Lerndatenobjekt *KeinAbfall:RunderStein4.*



Abb. B.72.: Lerndatenobjekt *KeinAbfall:Steinfeld1.*



Abb. B.73.: Lerndatenobjekt *KeinAbfall:Steinfeld2*.



Abb. B.74.: Lerndatenobjekt *KeinAbfall:Stein&Holz1*.



Abb. B.75.: Lerndatenobjekt *KeinAbfall:Baumstamm1*.

## B.2. Validierungsdaten

Dieser Abschnitt enthält die zur Evaluierung der SVM-Klassifikatoren verwendeten Bilddaten (siehe Abschnitt 8.4.2).

## B.2.1. Abfall



Abb. B.76.: Validierungsdatenobjekte der Klasse *Flasche*.



Abb. B.77.: Validierungsdatenobjekte der Klasse *Papier*.



Abb. B.78.: Validierungsdatenobjekte der Klasse *Plastiktüte*.

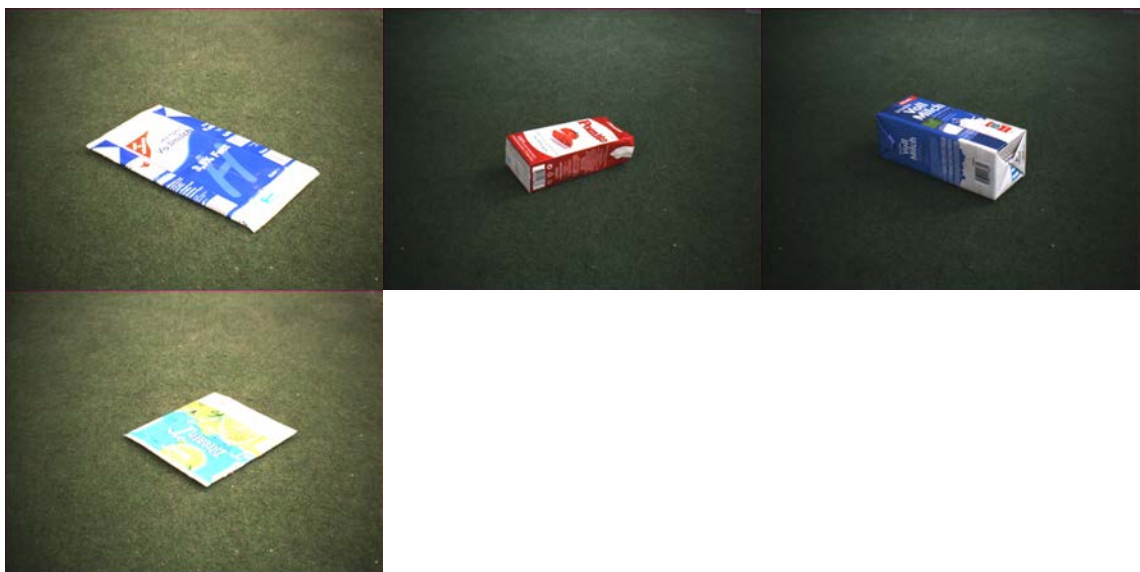


Abb. B.79.: Validierungsdatenobjekte der Klasse *Tetrapak*.

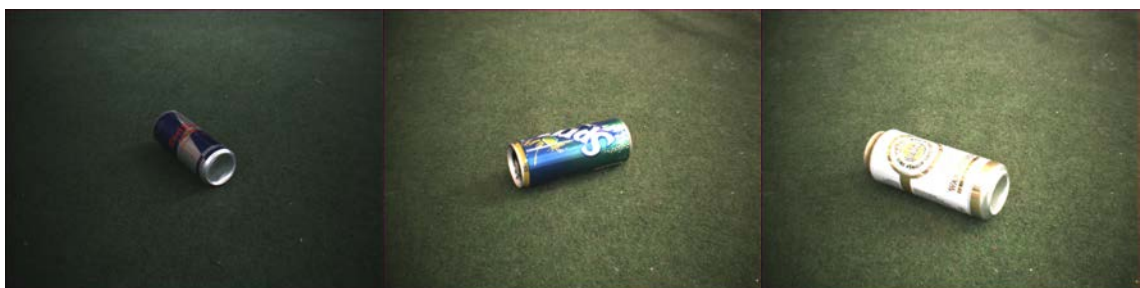


Abb. B.80.: Validierungsdatenobjekte der Klasse *Dose*.

## B.2.2. Kein Abfall



Abb. B.81.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Flacher Stein*.

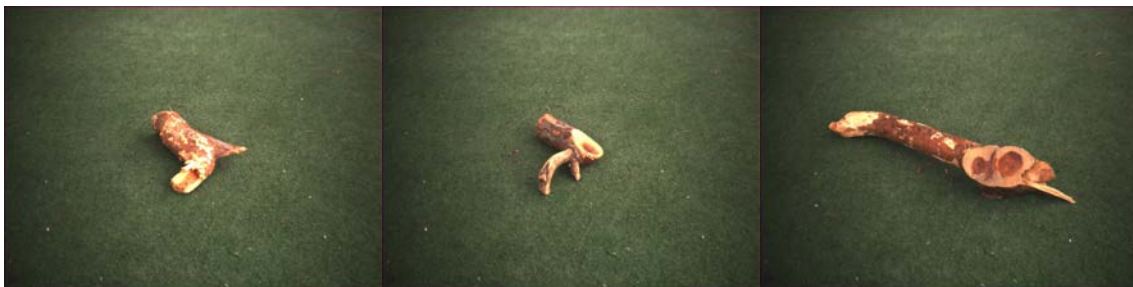


Abb. B.82.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Holzstück*.





Abb. B.83.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Pflanze*.



Abb. B.84.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Felsen*.



Abb. B.85.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Runder Stein*.



Abb. B.86.: Validierungsdatenobjekte der Klasse *Kein Abfall*, Kategorie *Steinfeld*.

## C. Ergebnisse des Trainings der SVM-Klassifikatoren

In diesem Anhang werden die Ergebnisse der Kombination aus Kreuzvalidierung und gridbasierter Parametersuche für das Training der einzelnen SVM-Klassifikatoren angegeben (siehe Abschnitt 7.3.2). Die mit dem in Abschnitt 7.3.1 beschriebenen Verfahren automatisch segmentierten Bildbereiche wurden für vorgegebene Prozentsätze jeweils in drei verschiedenen Varianten aufgeteilt: blind, nach Zugehörigkeit zu den einzelnen Objektansichten und nach Zugehörigkeit zu den jeweiligen Objekten. Dabei wurden in den beiden letztgenannten Fällen die zu einer Objektansicht bzw. zu einem Objekt gehörenden segmentierten Bildbereiche jeweils entweder vollständig der Lerndatenmenge oder der Testdatenmenge zugeordnet. Diese Varianten sind somit ein Indikator dafür, wie gut von bekannten Objektansichten auf unbekannte Objektansichten bzw. von bekannten Objekten auf unbekannte Objekte generalisiert werden kann. Für die jeweilige Aufteilung der Daten wurden die Parameter der SVM-Klassifikatoren in einem definierten Wertebereich systematisch variiert. Dies umfasste die Parameter  $\gamma$  und  $C$  für die zur Klassifikation *Abfall / Kein Abfall* bzw. zur Klassifikation der Abfallart eingesetzte  $C$ -SV L1 Klassifikation sowie die Parameter  $\gamma$  und  $\nu$  für die zur Absicherung der Abfallart eingesetzte  $\nu$ -SV L1 Träger-Schätzung (siehe [110]). Darüber hinaus wurden jeweils die in Abschnitt 7.3.2 vorgestellten Merkmalsgruppen untersucht und gegenübergestellt.

### C.1. Klassifikation Abfall / Kein Abfall

In diesem Abschnitt werden die Ergebnisse des Trainings der SVM-Klassifikatoren für die Bestimmung, ob es sich um Abfall handelt oder nicht, aufgeführt.

Kreuzvalidierung	Aufteilung [%]	Merkmale	C	$\gamma$	Fehler
Objekt	50	Alle	10000	0,016	0,150315
Objekt	50	Globale	10000	0,064	0,117047
Objekt	50	CCH	10000	0,002	0,187011
Objekt	50	LBP R8	10000	0,512	0,163914
Objekt	50	LBP R12	10000	1,024	0,158846
Objekt	75	Alle	100	1,024	0,129555
<i>wird fortgesetzt</i>					

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	<b>C</b>	<b><math>\gamma</math></b>	<b>Fehler</b>
Objekt	75	Globale	10000	0,064	0,110854
Objekt	75	CCH	10000	0,256	0,17902
Objekt	75	LBP R8	1000	1,024	0,131948
Objekt	75	LBP R12	1000	1,024	0,120049
Ansicht	50	Alle	10000	0,004	0,101253
Ansicht	50	Globale	10000	0,016	0,0918737
Ansicht	50	CCH	1000	4,096	0,168449
Ansicht	50	LBP R8	1	524,288	0,125644
Ansicht	50	LBP R12	1000	1,024	0,126166
Ansicht	75	Alle	10000	0,008	0,0837075
Ansicht	75	Globale	10000	0,064	0,0784784
Ansicht	75	CCH	1000	2,048	0,147304
Ansicht	75	LBP R8	10000	2,048	0,112874
Ansicht	75	LBP R12	100	16,384	0,0947224
Blind	50	Alle	10000	0,008	0,094086
Blind	50	Globale	10000	0,032	0,0860215
Blind	50	CCH	10	4,096	0,153898
Blind	50	LBP R8	1000	4,096	0,114919
Blind	50	LBP R12	100	16,384	0,108199
Blind	75	Alle	10000	0,256	0,0819462
Blind	75	Globale	10000	0,064	0,0683755
Blind	75	CCH	1000	1,024	0,143622
Blind	75	LBP R8	100	524,288	0,107222
Blind	75	LBP R12	1000	2,048	0,0872944
Blind	90	Alle	100	2,048	0,0701307
Blind	90	Globale	10000	0,064	0,067391
Blind	90	CCH	10	32,768	0,143048
Blind	90	LBP R8	100	262,144	0,111935
Blind	90	LBP R12	100	8,192	0,0877185

Tab. C.1.: Ergebnisse der Parametersuche für die Klassifikation Abfall / Kein Abfall.

## C.2. Klassifikation der Abfallart

In diesem Abschnitt werden die Ergebnisse des Trainings der SVM-Klassifikatoren für die Bestimmung, um welche Abfallart es sich handelt, angegeben.

Kreuzvalidierung	Aufteilung [%]	Merkmale	C	$\gamma$	Fehler
Objekt	50	Alle	10000	0,032	0,58647
Objekt	50	Globale	1000	1,024	0,625055
Objekt	50	CCH	100	1,024	0,668623
Objekt	50	LBP R8	100	4,096	0,658721
Objekt	50	LBP R12	100	32,768	0,680866
Objekt	75	Alle	1000	0,032	0,617167
Objekt	75	Globale	100	1,024	0,614495
Objekt	75	CCH	100	1,024	0,625425
Objekt	75	LBP R8	100	8,192	0,658035
Objekt	75	LBP R12	100	32,768	0,684048
Ansicht	50	Alle	10000	0,004	0,515716
Ansicht	50	Globale	1000	0,032	0,533165
Ansicht	50	CCH	1000	0,064	0,571705
Ansicht	50	LBP R8	1000	0,032	0,598728
Ansicht	50	LBP R12	1	131,072	0,624523
Ansicht	75	Alle	100	0,256	0,498083
Ansicht	75	Globale	10000	0,008	0,505166
Ansicht	75	CCH	10000	0,004	0,548956
Ansicht	75	LBP R8	10	131,072	0,60529
Ansicht	75	LBP R12	10	65,536	0,623713
Blind	50	Alle	10	4,096	0,431925
Blind	50	Globale	10000	0,128	0,463615
Blind	50	CCH	100	4,096	0,503521
Blind	50	LBP R8	10	131,072	0,503521
Blind	50	LBP R12	10	65,536	0,523474
Blind	75	Alle	100	1,024	0,375524
Blind	75	Globale	100	1,024	0,37548
Blind	75	CCH	100	2,048	0,403651
<i>wird fortgesetzt</i>					

Fortsetzung					
Kreuzvalidierung	Aufteilung [%]	Merkmale	C	$\gamma$	Fehler
Blind	75	LBP R8	100	131,072	0,410989
Blind	75	LBP R12	10	65,536	0,43667
Blind	90	Alle	1000	1,024	0,352083
Blind	90	Globale	10000	0,064	0,354167
Blind	90	CCH	1000	8,192	0,415476
Blind	90	LBP R8	10	131,072	0,386607
Blind	90	LBP R12	10	65,536	0,406845

Tab. C.2.: Ergebnisse der Parametersuche für die Klassifikation der Abfallart.

### C.3. Absicherung der Abfallart

In diesem Abschnitt werden die Ergebnisse des Trainings der SVM-Klassifikatoren für die Überprüfung, ob es sich tatsächlich um eine bestimmte Abfallart handelt, angegeben.

#### C.3.1. Abfallart Flasche

Kreuzvalidierung	Aufteilung [%]	Merkmale	$\nu$	$\gamma$	Fehler
Objekt	50	Alle	0,008	0,064	0,0660961
Objekt	50	Globale	0,004	0,064	0,0352655
Objekt	50	CCH	0,001	0,128	0,0400731
Objekt	50	LBP R8	0,008	2,048	0,0550601
Objekt	50	LBP R12	0,004	2,048	0,0748548
Objekt	75	Alle	0,016	0,008	0,101364
Objekt	75	Globale	0,016	0,008	0,0813636
Objekt	75	CCH	0,002	0,064	0,0813636
Objekt	75	LBP R8	0,001	2,048	0,0554545
Objekt	75	LBP R12	0,004	2,048	0,0737284
Ansicht	50	Alle	0,016	0,032	0,109255
Ansicht	50	Globale	0,004	0,064	0,0814132
Ansicht	50	CCH	0,008	1,024	0,0754608
Ansicht	50	LBP R8	0,002	0,002	0,0675115
Ansicht	50	LBP R12	0,032	1,024	0,1298
Ansicht	75	Alle	0,016	0,032	0,09887

*wird fortgesetzt*

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Ansicht	75	Globale	0,001	0,064	0,0765724
Ansicht	75	CCH	0,001	0,128	0,0692195
Ansicht	75	LBP R8	0,032	0,064	0,0423802
Ansicht	75	LBP R12	0,016	2,048	0,0577333
Blind	50	Alle	0,032	0,004	0,0757143
Blind	50	Globale	0,032	0,004	0,0506122
Blind	50	CCH	0,032	0,004	0,0506122
Blind	50	LBP R8	0,008	4,096	0,0705102
Blind	50	LBP R12	0,008	4,096	0,0806122
Blind	75	Alle	0,016	0,032	0,0405093
Blind	75	Globale	0,016	0,256	0,0196759
Blind	75	CCH	0,002	0,512	0,0196759
Blind	75	LBP R8	0,002	8,192	0,03125
Blind	75	LBP R12	0,001	8,192	0,0625
Blind	90	Alle	0,032	0,064	0,0166667
Blind	90	Globale	0,004	0,256	0,0277778
Blind	90	CCH	0,008	0,004	0,0166667
Blind	90	LBP R8	0,001	4,096	0,0333333
Blind	90	LBP R12	0,002	4,096	0,0277778

Tab. C.3.: Ergebnisse der Parametersuche für die Absicherung der Abfallart Flasche.

### C.3.2. Abfallart Papier

<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	50	Alle	0,032	0,004	0,0878494
Objekt	50	Globale	0,016	0,064	0,0919477
Objekt	50	CCH	0,008	0,064	0,083539
Objekt	50	LBP R8	0,032	0,064	0,0964194
Objekt	50	LBP R12	0,016	2,048	0,137474
Objekt	75	Alle	0,002	0,064	0,0667173
Objekt	75	Globale	0,002	0,128	0,0471491
Objekt	75	CCH	0,002	0,064	0,0404015
<i>wird fortgesetzt</i>					

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	75	LBP R8	0,064	0,032	0,0667173
Objekt	75	LBP R12	0,016	0,256	0,0987686
Ansicht	50	Alle	0,008	0,016	0,0892003
Ansicht	50	Globale	0,032	0,008	0,068106
Ansicht	50	CCH	0,032	0,004	0,0577778
Ansicht	50	LBP R8	0,004	2,048	0,0854547
Ansicht	50	LBP R12	0,032	0,512	0,104446
Ansicht	75	Alle	0,008	0,004	0,0362135
Ansicht	75	Globale	0,001	0,032	0,0362135
Ansicht	75	CCH	0,001	0,032	0,0362135
Ansicht	75	LBP R8	0,001	2,048	0,00735294
Ansicht	75	LBP R12	0,001	8,192	0,0519162
Blind	50	Alle	0,004	0,032	0,0849057
Blind	50	Globale	0,008	0,016	0,0707547
Blind	50	CCH	0,001	0,128	0,0471698
Blind	50	LBP R8	0,001	16,384	0,0613208
Blind	50	LBP R12	0,064	0,512	0,0660377
Blind	75	Alle	0,016	0,016	0,0364011
Blind	75	Globale	0,008	0,016	0,0274725
Blind	75	CCH	0,002	0,128	0,0274725
Blind	75	LBP R8	0,001	8,192	0,0473901
Blind	75	LBP R12	0,001	8,192	0,0480769
Blind	90	Alle	0,001	0,128	0,04625
Blind	90	Globale	0,004	0,008	0,02625
Blind	90	CCH	0,002	0,032	0,02625
Blind	90	LBP R8	0,001	2,048	0,02
Blind	90	LBP R12	0,001	8,192	0,02625

Tab. C.4.: Ergebnisse der Parametersuche für die Absicherung der Abfallart Papier.

### C.3.3. Abfallart Plastiktüte

<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	50	Alle	0,008	0,008	0,0590693
<i>wird fortgesetzt</i>					



<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	50	Globale	0,001	0,064	0,0642433
Objekt	50	CCH	0,032	0,016	0,0649061
Objekt	50	LBP R8	0,032	0,032	0,0550952
Objekt	50	LBP R12	0,032	0,064	0,0882766
Objekt	75	Alle	0,002	0,032	0,0624098
Objekt	75	Globale	0,032	0,002	0,0464165
Objekt	75	CCH	0,001	0,064	0,0231481
Objekt	75	LBP R8	0,001	0,256	0,0319865
Objekt	75	LBP R12	0,032	0,032	0,0227273
Ansicht	50	Alle	0,008	0,016	0,0344828
Ansicht	50	Globale	0,004	0,032	0,0344828
Ansicht	50	CCH	0,032	0,002	0,0301724
Ansicht	50	LBP R8	0,002	0,512	0,0603448
Ansicht	50	LBP R12	0,016	0,064	0,0474138
Ansicht	75	Alle	0,032	0,008	0,0255952
Ansicht	75	Globale	0,016	0,016	0,0255952
Ansicht	75	CCH	0,001	0,064	0,0357804
Ansicht	75	LBP R8	0,032	0,016	0,0166667
Ansicht	75	LBP R12	0,032	0,032	0,0434524
Blind	50	Alle	0,032	0,008	0,0909091
Blind	50	Globale	0,032	0,002	0,0568182
Blind	50	CCH	0,004	0,064	0,0795455
Blind	50	LBP R8	0,032	0,256	0,0625
Blind	50	LBP R12	0,032	0,064	0,0681818
Blind	75	Alle	0,004	0,032	0,0340909
Blind	75	Globale	0,008	0,008	0,0340909
Blind	75	CCH	0,002	0,016	0,0454545
Blind	75	LBP R8	0,016	0,128	0,0113636
Blind	75	LBP R12	0,004	1,024	0,0340909
Blind	90	Alle	0,004	0,016	0,04375
Blind	90	Globale	0,004	0,016	0,04375
Blind	90	CCH	0,008	0,008	0,0375
Blind	90	LBP R8	0,001	0,256	0,025
<i>wird fortgesetzt</i>					

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Blind	90	LBP R12	0,008	0,512	0,03125

Tab. C.5.: Ergebnisse der Parametersuche für die Absicherung der Abfallart Plastiktüte.

### C.3.4. Abfallart Tetrapak

<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	50	Alle	0,004	0,064	0,217023
Objekt	50	Globale	0,008	0,064	0,217023
Objekt	50	CCH	0,002	0,512	0,202094
Objekt	50	LBP R8	0,064	0,032	0,158645
Objekt	50	LBP R12	0,064	0,064	0,142825
Objekt	75	Alle	0,002	0,064	0,0989583
Objekt	75	Globale	0,001	0,128	0,0870536
Objekt	75	CCH	0,001	0,256	0,0751488
Objekt	75	LBP R8	0,032	0,064	0,0513393
Objekt	75	LBP R12	0,002	2,048	0,0669643
Ansicht	50	Alle	0,032	0,032	0,0883325
Ansicht	50	Globale	0,004	0,128	0,0826506
Ansicht	50	CCH	0,032	0,008	0,0826506
Ansicht	50	LBP R8	0,016	0,512	0,106457
Ansicht	50	LBP R12	0,016	2,048	0,106325
Ansicht	75	Alle	0,032	0,064	0,0434783
Ansicht	75	Globale	0,004	0,128	0,0326087
Ansicht	75	CCH	0,004	0,128	0,0326087
Ansicht	75	LBP R8	0,016	0,512	0,0326087
Ansicht	75	LBP R12	0,002	1,024	0,0217391
Blind	50	Alle	0,064	0,004	0,089795
Blind	50	Globale	0,004	0,128	0,0895722
Blind	50	CCH	0,064	0,002	0,0893494
Blind	50	LBP R8	0,032	0,512	0,0748663
Blind	50	LBP R12	0,032	0,256	0,097148
Blind	75	Alle	0,032	0,004	0,0444079
<i>wird fortgesetzt</i>					

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Blind	75	Globale	0,004	0,064	0,0444079
Blind	75	CCH	0,002	0,064	0,0444079
Blind	75	LBP R8	0,002	0,512	0,03125
Blind	75	LBP R12	0,032	0,128	0,0287829
Blind	90	Alle	0,002	0,128	0,0666667
Blind	90	Globale	0,001	0,128	0,0666667
Blind	90	CCH	0,001	0,128	0,0666667
Blind	90	LBP R8	0,032	1,024	0,0333333
Blind	90	LBP R12	0,002	0,256	0,0166667

Tab. C.6.: Ergebnisse der Parametersuche für die Absicherung der Abfallart Tetrapak.

### C.3.5. Abfallart Dose

<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Objekt	50	Alle	0,008	0,064	0,231014
Objekt	50	Globale	0,004	0,256	0,211277
Objekt	50	CCH	0,032	0,004	0,179763
Objekt	50	LBP R8	0,008	4,096	0,142822
Objekt	50	LBP R12	0,064	0,064	0,173993
Objekt	75	Alle	0,032	0,008	0,188645
Objekt	75	Globale	0,008	0,016	0,181319
Objekt	75	CCH	0,001	0,256	0,181319
Objekt	75	LBP R8	0,001	8,192	0,0430403
Objekt	75	LBP R12	0,004	0,256	0,0733173
Ansicht	50	Alle	0,064	0,002	0,0821705
Ansicht	50	Globale	0,001	0,064	0,0905452
Ansicht	50	CCH	0,064	0,002	0,0602657
Ansicht	50	LBP R8	0,001	16,384	0,0877778
Ansicht	50	LBP R12	0,032	0,064	0,0831297
Ansicht	75	Alle	0,064	0,002	0,0822511
Ansicht	75	Globale	0,001	0,064	0,0941558
Ansicht	75	CCH	0,064	0,002	0,0584416
<i>wird fortgesetzt</i>					

<i>Fortsetzung</i>					
<b>Kreuzvalidierung</b>	<b>Aufteilung [%]</b>	<b>Merkmale</b>	$\nu$	$\gamma$	<b>Fehler</b>
Ansicht	75	LBP R8	0,032	0,256	0,0692641
Ansicht	75	LBP R12	0,512	0,001	0,162338
Blind	50	Alle	0,032	0,256	0,106061
Blind	50	Globale	0,032	0,032	0,0984848
Blind	50	CCH	0,004	0,256	0,0909091
Blind	50	LBP R8	0,008	4,096	0,128788
Blind	50	LBP R12	0,016	2,048	0,0757576
Blind	75	Alle	0,032	0,004	0,0451389
Blind	75	Globale	0,032	0,004	0,0451389
Blind	75	CCH	0,008	0,064	0,0295139
Blind	75	LBP R8	0,001	8,192	0,104167
Blind	75	LBP R12	0,002	2,048	0,0416667
Blind	90	Alle	0,064	0,002	0,025
Blind	90	Globale	0,032	0,002	0,0416667
Blind	90	CCH	0,001	0,128	0,0333333
Blind	90	LBP R8	0,002	4,096	0,0666667
Blind	90	LBP R12	0,032	0,128	0,05

Tab. C.7.: Ergebnisse der Parametersuche für die Absicherung der Abfallart Dose.

## D. Abbildungsverzeichnis

1.1	Regelkreis der autonomen Inspektion . . . . .	4
1.2	Übersicht über die vier Teilbereiche dieser Arbeit . . . . .	5
2.1	Unbemannter Helikopter zur Inspektion von Hochspannungsleitungen . . . . .	9
2.2	Der Offshore-Inspektionsroboter MIMROex . . . . .	10
2.3	Die mehrsegmentigen Kanalinspektionsroboter KAIRO und MAKRO . . . . .	11
2.4	Der mehrsegmentige Kanalinspektionsroboter MAKROPLUS. . . . .	12
2.5	Der mehrsegmentige Kanalinspektionsroboter KAIRO II. . . . .	13
2.6	Modelle der aktuellen Mars Rover . . . . .	14
2.7	Übersicht Steuerungsarchitekturen . . . . .	17
2.8	Überblick über das OASIS-System . . . . .	24
2.9	Zusammenhang zwischen einer Konzeptualisierung und einer Ontologie . . . . .	28
2.10	Das Schichtenmodell des Semantic Web . . . . .	32
2.11	OODA-Entscheidungsfindungszyklus für unbemannte Systeme . . . . .	39
2.12	Realisierung des OODA-Entscheidungsfindungszyklus . . . . .	40
2.13	Konzeptrepräsentation, Instanzenerzeugung und -behandlung . . . . .	41
2.14	Intervall-Interaktionen zwischen Planschritten . . . . .	50
3.1	Architektur der semantischen Missionssteuerung . . . . .	65
3.2	Unterkomponenten der Inspektionsdatenauswertung . . . . .	66
3.3	Unterkomponenten der Semantischen Inspektion . . . . .	67
3.4	Sequenzdiagramm zur allgemeinen Planerzeugung und -ausführung . . . . .	70
3.5	Sequenzdiagramm zur aktiven Untersuchung von interessierenden Entitäten . . . . .	72
4.1	Struktureller Aufbau der Wissensbasis . . . . .	78
4.2	Strukturdiagramm Sensortaxonomie . . . . .	79
4.3	Strukturdiagramm Prädikatenlogik und Parameter . . . . .	80
4.4	Strukturdiagramm Ressourcen . . . . .	80
4.5	Strukturdiagramm Kernontologie Inspektion . . . . .	83
4.6	Strukturdiagramm Kernontologie Mission . . . . .	85
4.7	Strukturdiagramm Bedingungen . . . . .	86
4.8	Strukturdiagramm Parameter . . . . .	86

4.9	Strukturdiagramm Verknüpfungen . . . . .	86
4.10	Strukturdiagramm Merkmalsklassen zur Charakterisierung von Abfall . . . . .	87
4.11	Strukturdiagramm Abfall-Entitätsklassen . . . . .	87
5.1	Zeitliche Ordnungsbedingungen . . . . .	94
5.2	Beispiel für einen Flexiblen Hierarchischen Plan . . . . .	96
5.3	Übersicht über die Komponenten des Planers . . . . .	99
6.1	Ablauf der aktiven Untersuchung interessierender Entitäten . . . . .	121
6.2	Bayes'sches Netzwerk zur Bestimmung der tatsächlichen Entitätsklasse . . . . .	123
6.3	Entwicklung der Wahrscheinlichkeiten für die Entitätsklassen . . . . .	125
7.1	Übersicht über die im Rahmen dieser Arbeit entwickelten Komponenten . . . . .	129
7.2	Die sechsbeinige Laufmaschine LAURON IV . . . . .	131
7.3	Hardwarearchitektur von LAURON IV . . . . .	132
7.4	Architektur der verhaltensbasierten Steuerung von LAURON IV . . . . .	133
7.5	Ausschnitt des aus Tiefendaten gewonnenen Umweltmodells . . . . .	135
7.6	Ausführung eines RRT-Explorationsschrittes . . . . .	139
7.7	Pfadplanung mittels RRT . . . . .	140
7.8	Fehlerfunktionen der <i>Minimal Deviation</i> Methode . . . . .	141
7.9	Beispiel zur Pfadgenerierung mit <i>Path Transform</i> Kosten . . . . .	144
7.10	Verfahren zur Bestimmung auffälliger Bildbereiche . . . . .	150
7.11	Lokale Binäre Muster . . . . .	156
7.12	Überblick über die verwendeten Texturmerkmale . . . . .	157
7.13	Überblick über die semantische Missionssteuerung von LAURON IV . . . . .	160
7.14	Überblick über die Gesamtarchitektur der Steuerung von LAURON IV . . . . .	161
7.15	Bedienkomponente zur Vorgabe von Missionssaufgaben . . . . .	163
7.16	Bedienkomponente zur Inbetriebnahme von LAURON IV . . . . .	164
7.17	Bedienkomponente zur Vorgabe von zu inspizierenden Regionen . . . . .	165
7.18	Bedienkomponente zur Überwachung der generierten Pläne . . . . .	166
7.19	Bedienkomponente zur Überwachung der segmentierten Bildbereiche . . . . .	167
7.20	Bedienkomponente zur Überwachung der Inspektionsdatenauswertung . . . . .	169
7.21	Bedienkomponente zum Stellen von Anfragen an die Wissensbasis . . . . .	170
7.22	Protégé: Hierarchie der modellierten Ontologien . . . . .	171
7.23	Protégé: Anzahl der modellierten Klassen, Relationen und Instanzen . . . . .	171
7.24	Protégé: Elementare Aufgaben zur Navigation . . . . .	172
7.25	Protégé: Elementare Aufgaben zur Detektion . . . . .	172
7.26	Protégé: Merkmalsklassen zur Charakterisierung von Abfallobjekten. . . . .	173
7.27	Protégé: Regelbasis des semantischen Inspektionsmodells . . . . .	174

7.28	Protégé: Regel zur Bestimmung der potentiellen Klassenzugehörigkeit . . . . .	175
7.29	Flexibler Hierarchischer Plan in XML-Serialisierung . . . . .	176
7.30	Bayes'sches Netzwerk zur Fusion der Datenauswertungsergebnisse . . . . .	177
7.31	Entscheidungsbaum zur aktiven Untersuchung interessierender Entitäten . . . . .	179
8.1	Die verschiedenen Evaluierungsphasen . . . . .	183
8.2	Testfall in Form einer XML-Datei . . . . .	185
8.3	Vordefinierte Anfragen zur Überprüfung der Wissensbasis . . . . .	188
8.4	Bayes'sches Netzwerk nach Klassifikation Abfall / Kein Abfall . . . . .	189
8.5	Bayes'sches Netzwerk nach Klassifikation der Abfallart . . . . .	190
8.6	Bayes'sches Netzwerk nach Absicherung der Abfallart . . . . .	191
8.7	Wahrscheinlichkeitsverteilungen mit zugehörigen Konfidenzen . . . . .	191
8.8	Benchmark-Umgebungen zur Evaluierung von Bahnplanungsverfahren . . . . .	192
8.9	Evaluierung der RRT-Bahnplanung für fünfzehn Zielpunkte . . . . .	193
8.10	Benchmark-Umgebungen zur Evaluierung des <i>Complete Coverage</i> Verfahrens .	195
8.11	Evaluierung der <i>Distance Transform</i> Kostenfunktion . . . . .	196
8.12	Evaluierung der <i>Path Transform</i> Kostenfunktion . . . . .	197
8.13	Beispiele für Navigationsmissionen in unstrukturiertem Gelände . . . . .	197
8.14	Beispiele für Detektionsmissionen in unstrukturiertem Gelände . . . . .	201
8.15	Ergebnis der Klassifikation Abfall / Kein Abfall an einem Seeufer . . . . .	202
8.16	Ergebnis der Klassifikation Abfall / Kein Abfall in einer Parkanlage . . . . .	203
8.17	Beispiele für Inspektionsmissionen unter Laborbedingungen . . . . .	204
8.18	Pfad zum Absuchen einer Region im Labor . . . . .	205
8.19	Bildsequenz einer Inspektionsmission im Labor . . . . .	206
8.20	Gefundene interessierende Entitäten im Labor . . . . .	208
8.21	Beispiele für Inspektionsmissionen in unstrukturiertem Gelände . . . . .	209
8.22	Pfad zum Absuchen einer Region in einer Grünanlage . . . . .	210
8.23	Bildsequenz einer Inspektionsmission in einer Grünanlage . . . . .	211
8.24	Gefundene interessierende Entitäten in einer Grünanlage . . . . .	212
8.25	Demomission für KAIRO II . . . . .	213
8.26	Erweiterte Demomission für KAIRO II . . . . .	214
B.1	Lerndatenobjekt <i>Flasche:Aldi</i> . . . . .	231
B.2	Lerndatenobjekt <i>Flasche:AlteRebe</i> . . . . .	231
B.3	Lerndatenobjekt <i>Flasche:Bionade</i> . . . . .	232
B.4	Lerndatenobjekt <i>Flasche:BlackForest</i> . . . . .	232
B.5	Lerndatenobjekt <i>Flasche:Apfelsaftschorle</i> . . . . .	232
B.6	Lerndatenobjekt <i>Flasche:ColaPET</i> . . . . .	232

B.7	Lerndatenobjekt <i>Flasche:ColaZero</i>	233
B.8	Lerndatenobjekt <i>Flasche:KumpfOrangensaft</i>	233
B.9	Lerndatenobjekt <i>Flasche:KurparkMedium</i>	233
B.10	Lerndatenobjekt <i>Flasche:MezzoMix</i>	233
B.11	Lerndatenobjekt <i>Flasche:NesteaZitrone</i>	234
B.12	Lerndatenobjekt <i>Flasche:Zäpfle</i>	234
B.13	Lerndatenobjekt <i>Papier:Baguette</i>	234
B.14	Lerndatenobjekt <i>Papier:Balisto</i>	234
B.15	Lerndatenobjekt <i>Papier:Express</i>	235
B.16	Lerndatenobjekt <i>Papier:Gauloises</i>	235
B.17	Lerndatenobjekt <i>Papier:GoldringeFlach</i>	235
B.18	Lerndatenobjekt <i>Papier:L&amp;M</i>	235
B.19	Lerndatenobjekt <i>Papier:Marlboro</i>	236
B.20	Lerndatenobjekt <i>Papier:MarlboroDuhani</i>	236
B.21	Lerndatenobjekt <i>Papier:Mars</i>	236
B.22	Lerndatenobjekt <i>Papier:McDonalds</i>	236
B.23	Lerndatenobjekt <i>Papier:MrBaker</i>	237
B.24	Lerndatenobjekt <i>Papier:Neff</i>	237
B.25	Lerndatenobjekt <i>Papier:Visel</i>	237
B.26	Lerndatenobjekt <i>Plastiktüte:Aldi</i>	237
B.27	Lerndatenobjekt <i>Plastiktüte:Buchhandlung</i>	238
B.28	Lerndatenobjekt <i>Plastiktüte:EspritCelebration</i>	238
B.29	Lerndatenobjekt <i>Plastiktüte:QuaxiFröschli</i>	238
B.30	Lerndatenobjekt <i>Plastiktüte:Rewe</i>	238
B.31	Lerndatenobjekt <i>Plastiktüte:SaftGoldbären</i>	239
B.32	Lerndatenobjekt <i>Plastiktüte:Thalia</i>	239
B.33	Lerndatenobjekt <i>Plastiktüte:Weiß</i>	239
B.34	Lerndatenobjekt <i>Tetrapak:Breisgau</i>	239
B.35	Lerndatenobjekt <i>Tetrapak:MangoMaracuja</i>	240
B.36	Lerndatenobjekt <i>Tetrapak:Milfina</i>	240
B.37	Lerndatenobjekt <i>Tetrapak:Milch</i>	240
B.38	Lerndatenobjekt <i>Tetrapak:OrangenT</i>	240
B.39	Lerndatenobjekt <i>Tetrapak:PfirsichT</i>	241
B.40	Lerndatenobjekt <i>Tetrapak:WildbeerenT</i>	241
B.41	Lerndatenobjekt <i>Tetrapak:ZitronenT</i>	241
B.42	Lerndatenobjekt <i>Dose:AfriCola</i>	241
B.43	Lerndatenobjekt <i>Dose:Cashews</i>	242



B.44	Lerndatenobjekt <i>Dose:Cola</i> . . . . .	242
B.45	Lerndatenobjekt <i>Dose:ColaLight</i> . . . . .	242
B.46	Lerndatenobjekt <i>Dose:Fanta</i> . . . . .	242
B.47	Lerndatenobjekt <i>Dose:MezzoMix</i> . . . . .	243
B.48	Lerndatenobjekt <i>Dose:Mildessa</i> . . . . .	243
B.49	Lerndatenobjekt <i>Dose:Rotkohl</i> . . . . .	243
B.50	Lerndatenobjekt <i>Dose:Pringles</i> . . . . .	243
B.51	Lerndatenobjekt <i>KeinAbfall:FlacherStein1</i> . . . . .	244
B.52	Lerndatenobjekt <i>KeinAbfall:FlacherStein2</i> . . . . .	244
B.53	Lerndatenobjekt <i>KeinAbfall:FlacherStein3</i> . . . . .	244
B.54	Lerndatenobjekt <i>KeinAbfall:FlacherStein4</i> . . . . .	244
B.55	Lerndatenobjekt <i>KeinAbfall:FlacherStein5</i> . . . . .	245
B.56	Lerndatenobjekt <i>KeinAbfall:FlacherStein6</i> . . . . .	245
B.57	Lerndatenobjekt <i>KeinAbfall:FlacherStein7</i> . . . . .	245
B.58	Lerndatenobjekt <i>KeinAbfall:Holzstück1</i> . . . . .	246
B.59	Lerndatenobjekt <i>KeinAbfall:Holzstück2</i> . . . . .	246
B.60	Lerndatenobjekt <i>KeinAbfall:Holzstück3</i> . . . . .	246
B.61	Lerndatenobjekt <i>KeinAbfall:Holzstück4</i> . . . . .	247
B.62	Lerndatenobjekt <i>KeinAbfall:Holzstück5</i> . . . . .	247
B.63	Lerndatenobjekt <i>KeinAbfall:Pflanze1</i> . . . . .	247
B.64	Lerndatenobjekt <i>KeinAbfall:Pflanze2</i> . . . . .	247
B.65	Lerndatenobjekt <i>KeinAbfall:Felsen1</i> . . . . .	248
B.66	Lerndatenobjekt <i>KeinAbfall:Felsen2</i> . . . . .	248
B.67	Lerndatenobjekt <i>KeinAbfall:Felsen3</i> . . . . .	248
B.68	Lerndatenobjekt <i>KeinAbfall:RunderStein1</i> . . . . .	248
B.69	Lerndatenobjekt <i>KeinAbfall:RunderStein2</i> . . . . .	249
B.70	Lerndatenobjekt <i>KeinAbfall:RunderStein3</i> . . . . .	249
B.71	Lerndatenobjekt <i>KeinAbfall:RunderStein4</i> . . . . .	249
B.72	Lerndatenobjekt <i>KeinAbfall:Steinfeld1</i> . . . . .	249
B.73	Lerndatenobjekt <i>KeinAbfall:Steinfeld2</i> . . . . .	250
B.74	Lerndatenobjekt <i>KeinAbfall:Stein&amp;Holz1</i> . . . . .	250
B.75	Lerndatenobjekt <i>KeinAbfall:Baumstamm1</i> . . . . .	250
B.76	Validierungsdatenobjekte der Klasse <i>Flasche</i> . . . . .	251
B.77	Validierungsdatenobjekte der Klasse <i>Papier</i> . . . . .	252
B.78	Validierungsdatenobjekte der Klasse <i>Plastiktüte</i> . . . . .	253
B.79	Validierungsdatenobjekte der Klasse <i>Tetrapak</i> . . . . .	253
B.80	Validierungsdatenobjekte der Klasse <i>Dose</i> . . . . .	253

B.81	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Flacher Stein</i> . . . . .	254
B.82	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Holzstück</i> . . . . .	254
B.83	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Pflanze</i> . . . . .	255
B.84	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Felsen</i> . . . . .	255
B.85	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Runder Stein</i> . . . . .	255
B.86	Validierungsdatenobjekte der Klasse <i>Kein Abfall</i> , Kategorie <i>Steinfeld</i> . . . . .	256

## E. Tabellenverzeichnis

2.1	Technische Daten der Mars Rover . . . . .	15
5.1	Anforderungen an den Planungsansatz . . . . .	92
5.2	Flexible Hierarchische Pläne: Herkunft der Komponenten . . . . .	95
5.3	Flexible Hierarchische Pläne: Beziehung zu Aufgaben und Methoden . . . . .	100
5.4	Erfüllung der Anforderungen an den Planungsansatz . . . . .	113
7.1	Texturmerkmale nach Haralick . . . . .	152
7.2	Bilddaten zur Detektion und Klassifikation von Abfall . . . . .	158
7.3	Niedrigste mittlere Fehlerraten der Klassifikatoren . . . . .	159
7.4	Anzahl der Klassen, Relationen und Individuen in den einzelnen Ontologien . .	168
8.1	Ergebnisse der RRT-Bahnplanung für einen Zielpunkt . . . . .	193
8.2	Ergebnisse der RRT-Bahnplanung für fünfzehn Zielpunkte . . . . .	194
8.3	Bilddaten der Detektionsmissionen im Labor . . . . .	198
8.4	Konfusionsmatrix Abfall / Kein Abfall für Detektionsmissionen im Labor . . .	198
8.5	Evaluationskriterien Klassifikation Abfall / Kein Abfall . . . . .	199
8.6	Kennzahlen Abfall / Kein Abfall für Detektionsmissionen im Labor . . . . .	199
8.7	Konfusionsmatrix Abfallart für Detektionsmissionen im Labor . . . . .	200
8.8	Bilddaten der Detektionsmissionen in unstrukturiertem Gelände . . . . .	200
8.9	Konfusionsmatrix Abfall / Kein Abfall für Detektionsmissionen im Gelände . .	200
8.10	Kennzahlen Abfall / Kein Abfall für Detektionsmissionen im Gelände . . . . .	204
8.11	Konfusionsmatrix Abfallart für Detektionsmissionen im Gelände . . . . .	204
8.12	Konfusionsmatrix Abfall / Kein Abfall für Inspektionsmission im Labor . . . .	205
8.13	Kennzahlen Abfall / Kein Abfall für Inspektionsmission im Labor . . . . .	207
8.14	Konfusionsmatrix Abfall / Kein Abfall für Inspektionsmission in Grünanlage .	207
8.15	Kennzahlen Abfall / Kein Abfall für Inspektionsmission in Grünanlage . . . . .	207
8.16	Zeitaufwand strukturelle Übertragung und Anpassung auf KAIRO II . . . . .	209
8.17	Zeitaufwand Demomission für KAIRO II . . . . .	209
8.18	Zeitaufwand Manöver <i>Positioniere Modul</i> für KAIRO II . . . . .	210
C.1	Parametersuche Klassifikation Abfall / Kein Abfall . . . . .	258
C.2	Parametersuche Klassifikation Abfallart . . . . .	260

C.3	Parametersuche Absicherung Abfallart Flasche . . . . .	261
C.4	Parametersuche Absicherung Abfallart Papier . . . . .	262
C.5	Parametersuche Absicherung Abfallart Plastiktüte . . . . .	264
C.6	Parametersuche Absicherung Abfallart Tetrapak . . . . .	265
C.7	Parametersuche Absicherung Abfallart Dose . . . . .	266

## F. Algorithmenverzeichnis

5.1	KannAufJedeWeise . . . . .	107
5.2	KönnteAufMancheWeise . . . . .	107
5.3	Nebenläufiger hierarchischer Koordinationsalgorithmus . . . . .	109
7.1	Erzeuge RRT . . . . .	137
7.2	Wähle Konfiguration . . . . .	137
7.3	Erweitere RRT . . . . .	138
7.4	Vollständiges Absuchen von Regionen . . . . .	145
A.1	Muss Erreichen . . . . .	225
A.2	Muss Überschreiben . . . . .	226
A.3	Muss Zurücksetzen . . . . .	227
A.4	Kann Erreichen . . . . .	228
A.5	Kann Überschreiben . . . . .	229
A.6	Kann Zurücksetzen . . . . .	230



## G. Literaturverzeichnis

- [1] AGOSTON, M. K.: *Computer Graphics and Geometric Modeling. Implementation and Algorithms*. Springer, 2005
- [2] ALAMI, R. ; CHATILA, R. ; FLEURY, S. ; GHALLAB, M. ; INGRAND, F. : An Architecture for Autonomy. In: *International Journal of Robotics Research* 17 (1998), Nr. 4, S. 315–337
- [3] ALBUS, J. S.: RCS: A Reference Model Architecture for Intelligent Systems. In: *AAAI Technical Report SS-95-02* (1995)
- [4] ALBUS, J. S. ; MCCAIN, H. G. ; LUMIA, R. : NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). In: *NIST Technical Note 1235* (1986)
- [5] ARKIN, R. C.: Motor Schema-Based Mobile Robot Navigation. In: *International Journal of Robotics Research* 8 (1989), Nr. 4, S. 92–112
- [6] ARKIN, R. C.: *Behavior-Based Robotics*. MIT Press, 1998
- [7] ARKIN, R. C. ; BALCH, T. : AuRA: Principles and Practice in Review. In: *Journal of Experimental and Theoretical Artificial Intelligence* 9 (1997), S. 175–189
- [8] BAJRACHARYA, M. ; MAIMONE, M. ; HELMICK, D. : Autonomy for Mars Rovers: Past, Present, and Future. In: *IEEE Computer Magazine* 41 (2008), Nr. 12, S. 44–50
- [9] BAY, H. ; ESS, A. ; TUYTELAARS, T. ; GOO, L. V.: SURF: Speeded Up Robust Features. In: *Computer Vision and Image Understanding (CVIU)* 110 (2008), Nr. 3, S. 346–359
- [10] BENGEL, M. ; PFEIFFER, K. ; GRAF, B. ; BUBECK, A. ; VERL, A. : Mobile Robots for Offshore Inspection and Manipulation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, S. 3317–3322
- [11] BERTRAM, B. : *Bildbasierte Methoden zur Erkennung von Abfall in unstrukturiertem Gelände*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Diplomarbeit, 2009

- [12] BIRKENHOFER, C. : *Adaptive Steuerung eines mehrsegmentigen Inspektionsroboters*, Fakultät für Informatik, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Dissertation, 2009
- [13] BISHOP, C. M.: *Pattern Recognition and Machine Learning*. Springer, 2008
- [14] BÖLKE, K.-P. : *Kanalinspektion – Zustände erkennen und dokumentieren*. Springer, 2009
- [15] BONASSO, R. P. ; FIRBY, R. J. ; GAT, E. ; KORTENKAMP, D. ; MILLER, D. P. ; SLACK, M. G.: Experiences with an Architecture for Intelligent, Reactive Agents. In: *Journal of Experimental and Theoretical Artificial Intelligence* 9 (1997), Nr. 2/3, S. 237–256
- [16] BONASSO, R. P.: Integrating Reaction Plans and Layered Competences through Synchronous Control. In: *International Joint Conference on Artificial Intelligence (IJCAI)* Bd. 2, 1991, S. 1225–1231
- [17] BORRELLY, J.-J. ; COSTE-MANIERE, E. ; ESPIAU, B. ; KAPELLOS, K. ; PISSARD-GIBOLLET, R. ; SIMON, D. ; TURRO, N. : The ORCCAD Architecture. In: *International Journal of Robotics Research* 17 (1998), Nr. 4, S. 338–359
- [18] BOUGUERRA, A. ; KARLSSON, L. ; SAFFIOTTI, A. : Monitoring the execution of robot plans using semantic knowledge. In: *Robotics and Autonomous Systems* 56 (2008), S. 942–954
- [19] BOYD, J. R.: *A discourse on winning and losing*. Unpublished set of briefing slides available at Air University Library, Maxwell AFB, Alabama, 1987
- [20] BRACHMAN, R. J. ; LEVESQUE, H. J.: *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2004
- [21] BROOKS, R. A.: A Robust Layered Control System For A Mobile Robot. In: *IEEE Journal of Robotics and Automation* 2 (1986), Nr. 1, S. 14–23
- [22] BUDDE, R. ; POINGÉ, A. : Complex Reactive Control with Synchronous Models. In: *Languages, compilers, and tools for embedded systems: ACM SIGPLAN Workshop LCTES*, 2000
- [23] BURGESS, C. J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Mining and Knowledge Discovery* 2 (1998), Nr. 2, S. 121–167
- [24] CALISI, D. ; IOCCHI, L. ; NARDI, D. ; SCALZO, C. M. ; ZIPARO, V. A.: Context-based design of robotic systems. In: *Robotics and Autonomous Systems* 56 (2008), S. 992–1003



- [25] CASTANO, R. ; ESTLIN, T. ; ANDERSON, R. C. ; GAINES, D. M. ; CASTANO, A. ; BORNSTEIN, B. ; CHOUINARD, C. ; JUDD, M. : OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. In: *Journal of Field Robotics* 24 (2007), Nr. 5, S. 379–397
- [26] CHANG, P. ; KRUMM, J. : Object Recognition with Color Cooccurrence Histograms. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999
- [27] CLEMENT, B. J.: *Abstract Reasoning for Multiagent Coordination and Planning*, University of Michigan, Dissertation, 2002
- [28] CLEMENT, B. J. ; DURFEE, E. H. ; BARRETT, A. C.: Abstract Reasoning for Planning and Coordination. In: *Journal of Artificial Intelligence Research* 28 (2007), S. 453–515
- [29] D’ESTE, C. ; SAMMUT, C. : Learning and generalising semantic knowledge from object scenes. In: *Robotics and Autonomous Systems* 56 (2008), S. 891–900
- [30] ESTLIN, T. ; GAINES, D. ; CHOUINARD, C. ; FISHER, F. ; CASTANO, R. ; JUDD, M. ; ANDERSON, R. C. ; NESNAS, I. : Enabling Autonomous Rover Science Through Dynamic Planning and Scheduling. In: *IEEE Aerospace Conference*, 2005, S. 385–396
- [31] FERREIN, A. ; LAKEMEYER, G. : Logic-based robot control in highly dynamic domains. In: *Robotics and Autonomous Systems* 56 (2008), S. 980–991
- [32] FIRBY, R. J.: *Adaptive Execution in Complex Dynamic Worlds*, Yale University, Dissertation, 1989
- [33] GALINDO, C. ; FERNÁNDEZ-MADRIGAL, J.-A. ; GONZÁLEZ, J. ; SAFFIOTTI, A. : Robot task planning using semantic maps. In: *Robotics and Autonomous Systems* 56 (2008), S. 955–966
- [34] GASSMANN, B. : *Modellbasierte, sensorgestützte Navigation von Laufmaschinen im Gelände*, Fakultät für Informatik, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Dissertation, 2007
- [35] GASSMANN, B. ; ZACHARIAS, F. ; ZÖLLNER, J. M. ; DILLMANN, R. : Localization of Walking Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*, 2005
- [36] GAMMA, E. ; HELM, R. ; JOHNSON, R. E.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1994

- [37] GAT, E. : Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. In: *AAAI National Conference on Artificial Intelligence*, 1992, S. 809–815
- [38] GENESERETH, M. R. ; NILSSON, N. J.: *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987
- [39] GHALLAB, M. ; LARUELLE, H. : Representation and Control in IxTeT, a Temporal Planner. In: *International Conference on Artificial Intelligence Planning Systems (AIPS)*, 1994, S. 61–67
- [40] GHALLAB, M. ; NAU, D. ; TRAVERSO, P. : *Automated Planning: theory and practice*. Morgan Kaufmann, 2004
- [41] GIL, Y. : Description Logics and Planning. In: *AI Magazine* 26 (2005), Nr. 2, S. 73–84
- [42] GREENSPAN, H. ; BELONGIE, S. ; GOODMAN, R. ; PERONA, P. ; RAKSHIT, S. ; ANDERSON, C. H.: Overcomplete steerable pyramid filters and rotation invariance. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, S. 222–228
- [43] GRUBER, T. R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: GUARINO, N. (Hrsg.) ; POLI, R. (Hrsg.): *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Deventer, The Netherlands : Kluwer Academic Publishers, 1993
- [44] GUARINO, N. : Formal Ontology and Information Systems. In: GUARINO, N. (Hrsg.): *International Conference on Formal Ontologies in Information Systems*, IOS Press, 1998, S. 3–15
- [45] HARNAD, S. : The symbol grounding problem. In: *Physica D: Nonlinear Phenomena* 42 (1990), S. 335–346
- [46] HARTANTO, R. ; HERTZBERG, J. : Augmenting JSHOP2 Planning with OWL-DL. In: *Workshop Planen, Scheduling und Konfigurieren, Entwerfen*, 2007
- [47] HARTANTO, R. ; HERTZBERG, J. : Fusing DL Reasoning with HTN Planning. In: *Künstliche Intelligenz (KI)*, 2008
- [48] HARTANTO, R. ; HERTZBERG, J. : On the Benefit of Fusing DL-Reasoning with HTN-Planning. In: *Künstliche Intelligenz (KI)*, 2009
- [49] HERTZBERG, J. ; SAFFIOTTI, A. : Using semantic knowledge in robotics. In: *Robotics and Autonomous Systems* 56 (2008), S. 875–877

- [50] HITZLER, P. ; KRÖTZSCH, M. ; RUDOLPH, S. ; SURE, Y. : *Semantic Web Grundlagen*. Springer, 2008
- [51] HOFFMANN, J. ; NEBEL, B. : The FF Planning System: Fast Plan Generation Through Heuristic Search. In: *Journal of Artificial Intelligence Research* 14 (2001), S. 253–302
- [52] HOLZAPFEL, H. ; NEUBIG, D. ; WAIBEL, A. : A dialogue approach to learning object descriptions and semantic categories. In: *Robotics and Autonomous Systems* 56 (2008), S. 1004–1013
- [53] INGRAND, F. F. ; CHATILA, R. ; ALAMI, R. ; ROBERT, F. : PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*, 1996, S. 43–49
- [54] ITTI, L. ; KOCH, C. : A Comparison of Feature Combination Strategies for Saliency-Based Visual Attention Systems. In: *SPIE Conference on Human Vision and Electronic Imaging (HVEI)* Bd. 3644, 1999, S. 473–482
- [55] ITTI, L. ; KOCH, C. : A saliency-based search mechanism for overt and covert shifts of visual attention. In: *Vision Research* 40 (2000), S. 1489–1506
- [56] ITTI, L. ; KOCH, C. ; NIEBUR, E. : A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Nr. 11, S. 1254–1259
- [57] KERSCHER, T. ; RÖNNAU, A. ; ZIEGENMEYER, M. ; GASSMANN, B. ; ZÖLLNER, J. M. ; DILLMANN, R. : Behaviour-based control of the six-legged walking machine LAURON IVc. In: *International Conference on Climbing and Walking Robots (CLAWAR)*, 2008
- [58] KLUGE, S. : *Entwicklung eines ontologiebasierten HTN-Planers für autonome Inspektionsroboter*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Diplomarbeit, 2010
- [59] KNIGHT, R. ; RABIDEAU, G. ; CHIEN, S. ; ENGELHARDT, B. ; SHERWOOD, R. : CASPER: Space Exploration through Continuous Planning. In: *IEEE Intelligent Systems* 16 (2001), S. 70–75
- [60] KNOOP, S. : *Interaktive Erstellung und Ausführung von Handlungswissen für einen Serviceroboter*, Fakultät für Informatik, Institut für Technische Informatik (ITEC), Universität Karlsruhe (TH), Dissertation, 2007

- [61] KNOOP, S. ; SCHMIDT-ROHR, S. R. ; DILLMANN, R. : A Flexible Task Knowledge Representation for Service Robots. In: *International Conference on Intelligent Autonomous Systems (IAS)*, 2006
- [62] KNOOP, S. ; VACEK, S. ; ZÖLLNER, R. ; AU, C. ; DILLMANN, R. : A CORBA-Based Distributed Software Architecture for Control of Service Robots. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004
- [63] KORTENKAMP, D. ; SIMMONS, R. : Robotic Systems Architecture and Programming. In: SICILIANO, B. (Hrsg.) ; KHATIB, O. (Hrsg.): *Springer Handbook of Robotics*. Springer, 2008, Kapitel 8
- [64] LAVALLE, S. M. ; KUFFNER, J. J.: Rapidly-Exploring Random Trees: Progress and Prospects. In: *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2000
- [65] MAYER-HERMANN, R. : *Entwurf einer Ontologie zur semantischen Beschreibung der Eigenschaften und Fähigkeiten der sechsbeinigen Laufmaschine LAURON IVc*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Studienarbeit, 2010
- [66] MIGUELÁÑEZ, E. ; PATRÓN, P. ; BROWN, K. E. ; PETILLOT, Y. R. ; LANE, D. M.: Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles. In: *IEEE Transactions on Knowledge and Data Engineering* 23 (2011), S. 759–773
- [67] MITCHELL, T. M.: *Machine Learning*. The McGraw-Hill Companies, Inc., 1997
- [68] MÜLLER, K. ; FISCHER, B. ; LEHMANN, T. ; HUNGER, W. ; SCHÄFER, T. : Entwicklung von Bilderkennungsverfahren zur Qualitätssicherung bei der Zustandserfassung von Kanalisationen. In: *bi UmweltBau* 5 (2006), S. 64–71
- [69] MODAYIL, J. ; KUIPERS, B. : The initial development of object knowledge by a learning robot. In: *Robotics and Autonomous Systems* 56 (2008), S. 879–890
- [70] MUSCETTOLA, N. ; NAYAK, P. P. ; PELL, B. ; WILLIAMS, B. C.: Remote Agent: To Boldly Go Where No AI System Has Gone Before. In: *Journal of Artificial Intelligence* 103 (1998), S. 5–47
- [71] MUSLINER, D. J. ; DURFEE, E. H.: World Modeling for the Dynamic Construction of Real Time Control Plans. In: *Journal of Artificial Intelligence* 74 (1995), Nr. 1, S. 83–127
- [72] NAU, D. S.: Current Trends in Automated Planning. In: *AI Magazine* 28 (2007), Nr. 4, S. 43–58

- [73] NÜCHTER, A. ; HERTZBERG, J. : Towards semantic maps for mobile robots. In: *Robotics and Autonomous Systems* 56 (2008), S. 915–926
- [74] NESNAS, I. A. D. ; WRIGHT, A. ; BAJRACHARYA, M. ; SIMMONS, R. ; ESTLIN, T. ; KIM, W. S.: CLARATy: An architecture for reusable robotic software. In: *SPIE Aerosense Conference*, 2003
- [75] NILSSON, N. J.: A Mobile Automaton: An Application of AI Techniques. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 1969, S. 509–520
- [76] NILSSON, N. J.: *Principles of Artificial Intelligence*. Tioga Pub. Co., 1980
- [77] OJALA, T. ; PIETIKÄINEN, M. ; MÄENPÄÄ, T. : Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), Nr. 7, S. 971–987
- [78] PFOTZER, L. : *Entwicklung einer Navigationsplanung zur Inspektion von komplexen Umgebungen mit einem sechsbeinigen Laufroboter*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Diplomarbeit, 2009
- [79] POSNER, I. ; SCHROETER, D. ; NEWMAN, P. : Online generation of scene descriptions in urban environments. In: *Robotics and Autonomous Systems* 56 (2008), S. 901–914
- [80] POULIOT, N. ; LATULIPPE, P. ; MONTAMBAULT, S. ; TREMBLAY, S. : Reliable and Intuitive Teleoperation of LineScout: a Mobile Robot for Live Transmission Line Maintenance. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, S. 1703–1710
- [81] REGENSTEIN, K. ; KERSCHER, T. ; BIRKENHOFER, C. ; ASFOUR, T. ; ZÖLLNER, J. M. ; DILLMANN, R. : Universal Controller Module (UCoM) - component of a modular concept in robotic systems. In: *IEEE International Symposium on Industrial Electronics (ISIE)*, 2007
- [82] REKLEITIS, I. ; BEDWANI, J.-L. ; DUPUIS, E. : Over-the-horizon, autonomous navigation for planetary exploration. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, S. 2248–2255
- [83] RÖNNAU, A. ; KERSCHER, T. ; ZIEGENMEYER, M. ; ZÖLLNER, J. M. ; DILLMANN, R. : Adaptation of a six-legged walking robot to its local environment. In: *International Workshop on Robot Motion and Control (RoMoCo)*, 2009

- [84] ROBERT M. HARALICK, K. S. ; DINSTEIN, I. : Textural Features for Image Classification. In: *IEEE Transactions on Systems, Man, and Cybernetics SMC-3* (1973), Nr. 6, S. 610–621
- [85] ROSENFELD, A. ; PFALTZ, J. L.: Sequential Operations in Digital Picture Processing. In: *Journal of the Association for Computing Machinery* 13 (1966), Nr. 4, S. 471–494
- [86] RUSSELL, S. ; NORVIG, P. : *Artificial Intelligence: A Modern Approach*. 2. Auflage. Prentice Hall, 2003 (Series in Artificial Intelligence)
- [87] RUSU, R. B. ; MARTON, Z. C. ; BLODOW, N. ; DOLHA, M. ; BEETZ, M. : Towards 3D Point cloud based object maps for household environments. In: *Robotics and Autonomous Systems* 56 (2008), S. 927–941
- [88] SARIDIS, G. N.: Architectures for Intelligent Controls. In: GUPTA, M. M. (Hrsg.) ; SINHA, N. K. (Hrsg.): *Intelligent Control Systems: Theory and Applications*. IEEE Press, 1995, Kapitel 6
- [89] SAYLER, S. : *Entwurf eines Inspektionsplaners für autonome Serviceroboter*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Diplomarbeit, 2008
- [90] SCHÖLKOPF, B. : *Support Vector Learning*, Fachbereich Informatik, Technische Universität Berlin, Dissertation, 1997
- [91] SCHÖLKOPF, B. ; PLATT, J. C. ; SHAWE-TAYLOR, J. ; SMOLA, A. J. ; WILLIAMSON, R. C.: Estimating the Support of a High-Dimensional Distribution. In: *Microsoft Research Technical Report MSR-TR-99-87* (1999)
- [92] SCHMIDT-ROHR, S. R. ; JÄKEL, R. ; LÖSCH, M. ; DILLMANN, R. : Compiling POMDP models for a multimodal service robot from background knowledge. In: *European Robotics Symposium (EUROS)*, 2008
- [93] SCHMIDT-ROHR, S. R. ; KNOOP, S. ; LÖSCH, M. ; DILLMANN, R. : A probabilistic control architecture for robust autonomy of an anthropomorphic service robot. In: *International Conference on Cognitive Systems (CogSys)*, 2008
- [94] SCHÖNBEIN, R. : *Agenten- und ontologiebasierte Software-Architektur zur interaktiven Bildauswertung*, Fakultät für Informatik, Universität Karlsruhe (TH), Dissertation, 2006
- [95] SCHOLL, K.-U. : *Konzeption und Realisierung einer Steuerung für vielsegmentige, autonome Kanalroboter*, Fakultät für Informatik, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Dissertation, 2003

- [96] SIMMONS, R. ; APFELBAUM, D. : A task description language for robot control. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* Bd. 3, 1998, S. 1931–1937
- [97] STREICH, H. ; ADRIA, O. : Software approach for the autonomous inspection robot MAKRO. In: *IEEE International Conference on Robotics and Automation (ICRA)* Bd. 4, 2004, S. 3411–3416
- [98] STULP, F. ; BEETZ, M. : Combining declarative, procedural, and predictive knowledge to generate, execute, and optimize robot plans. In: *Robotics and Autonomous Systems* 56 (2008), S. 967–979
- [99] TUCERYAN, M. : Texture Analysis. In: CHEN, C. H. (Hrsg.) ; PAU, L. F. (Hrsg.) ; WANG, P. S. P. (Hrsg.): *The Handbook of Pattern Recognition and Computer Vision*. World Scientific Publishing Co., 1998, Kapitel 2
- [100] UHL, K. ; ZIEGENMEYER, M. : MCA2 - An Extensible Modular Framework for Robot Control Applications. In: *International Conference on Climbing and Walking Robots (CLAWAR)*, 2007
- [101] UHL, K. ; ZIEGENMEYER, M. ; GASSMANN, B. ; ZÖLLNER, J. M. ; DILLMANN, R. : Entwurf einer semantischen Missionssteuerung für autonome Serviceroboter. In: *Fachgespräche Autonome Mobile Systeme (AMS)*, 2007
- [102] VAPNIK, V. N.: *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998
- [103] WALTHER, D. : *Interactions of Visual Attention and Object Recognition: Computational Modeling, Algorithms, and Psychophysics*, California Institute of Technology, Dissertation, 2006
- [104] WANG, B. ; HAN, L. ; ZHANG, H. ; WANG, Q. ; LI, B. : A Flying Robotic System for Power Line Corridor Inspection. In: *IEEE International Conference on Robotics and Biomimetics*, 2009, S. 2468–2473
- [105] WARINGO, M. ; HENRICH, D. : Efficient Smoothing of Piecewise Linear Paths with Minimal Deviation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006
- [106] WORLD WIDE WEB CONSORTIUM (W3C): *Das Schichtenmodell des Semantic Web*. <http://www.w3.org/2007/03/layerCake.png>, Abruf: 30.10.2010
- [107] YERSHOVA, A. ; LAVALLE, S. M.: Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. In: *IEEE Transactions on Robotics* (2007), S. 151–157

- [108] ZELINSKY, A. ; JARVIS, R. ; BYRNE, J. C. ; YUTA, S. : Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. In: *International Conference on Advanced Robotics (ICAR)*, 1993, S. 533–538
- [109] ZHANG, J. ; MARSZALEK, M. ; LAZEBNIK, S. ; SCHMID, C. : Local features and kernels for classification of texture and object categories: A comprehensive study. In: *International Journal of Computer Vision* 73 (2007), Nr. 2, S. 213–238
- [110] ZIEGENMEYER, M. : *Optimierung und Anpassung der Support-Vektor-Klassifikation motiviert durch reale Diagnoseanwendungen*, FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH), Diplomarbeit, 2003
- [111] ZIEGENMEYER, M. ; RÖNNAU, A. ; KERSCHER, T. ; ZÖLLNER, J. M. ; DILLMANN, R. : Die sechsbeinige Laufmaschine LAURON IVc. In: *Fachgespräche Autonome Mobile Systeme (AMS)*, 2009
- [112] ZIEGENMEYER, M. ; UHL, K. ; SAYLER, S. ; GASSMANN, B. ; ZÖLLNER, J. M. ; DILLMANN, R. : Can Semantics Help Autonomous Service Robots in Inspecting Complex Environments? In: *International Conference on Climbing and Walking Robots (CLAWAR)*, 2008
- [113] ZIEGENMEYER, M. ; UHL, K. ; SAYLER, S. ; ZÖLLNER, J. M. ; DILLMANN, R. : A Semantic Approach for the Inspection of Complex Environments with Autonomous Service Robots. In: *IARP Workshop on Environmental Maintenance and Protection*, 2008
- [114] ZIEGENMEYER, M. ; UHL, K. ; ZÖLLNER, J. M. ; DILLMANN, R. : Autonomous Inspection of Complex Environments by Means of Semantic Techniques. In: *Workshop on Artificial Intelligence Applications in Environmental Protection (AIAEP)*, 2009