Steuerung eines mobilen Roboters mit evolvierten Merkmalsdetektoren

Dissertation
der Fakultät für Informatik
der Eberhard-Karls-Universität zu Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. Marc Ebner
aus Tübingen

Tübingen
1999

Tag der mündlichen Qualifikation: 19. Mai 1999
Dekan: Prof. Dr. Klaus-Jörn Lange
1. Berichterstatter: Prof. Dr. Andreas Zell
2. Berichterstatter: Prof. Dr. Wolfgang Straßer

Kurzfassung

In der vorliegenden Arbeit wird aufgezeigt, wie durch den Einsatz evolutionärer Algorithmen unterschiedliche Probleme der Robotik gelöst werden können. Die natürliche Evolution hat eine Vielzahl von Lebewesen hervorgebracht, die sich problemlos in ihrer Umgebung zurechtfinden. Vergleichbare Leistungen technischer Systeme konnten bisher in der Robotik noch nicht realisiert werden. Daher ist die Hoffnung groß, durch den Einsatz evolutionärer Algorithmen in der Robotik, der sogenannten evolutionären Robotik, größere Fortschritte zu erzielen. Speziell geht es in der vorliegenden Arbeit um die Evolution von Merkmalsdetektoren, die in unterschiedlichen Bereichen, wie z.B. zur Selbstlokalisation, zur Steuerung und zur Berechnung des optischen Flusses, eingesetzt werden.

Zur Evolution der Merkmalsdetektoren wurde genetisches Programmieren eingesetzt. Beim genetischen Programmieren handelt es sich um eine Klasse evolutionärer Algorithmen, mit denen Programme evolviert werden können. D.h. als Ergebnis der simulierten Evolution entsteht ein Programm, das das gegebene Problem löst. Die Fitneß des Programmes beschreibt, wie gut das Programm das Problem löst. Durch genetisches Programmieren ist es theoretisch möglich, die Programmierung zu automatisieren.

Im Vergleich zu anderen Klassen evolutionärer Algorithmen, wie z.B. den genetischen Algorithmen, existieren für genetisches Programmieren relativ wenig theoretische Erkenntnisse. Zur Erweiterung des Wissens über genetisches Programmieren wurde daher der Suchraum von genetischem Programmieren analysiert. Es wurde gezeigt, daß sich der Suchraum von genetischem Programmieren ganz wesentlich vom Suchraum der genetischen Algorithmen unterscheidet. Es wurde eine Analogie zur natürlichen Evolution gezogen, die aufzeigt, welche Eigenschaften der Suchraum aufweisen sollte, damit mit Hilfe von genetischem Programmieren bessere Ergebnisse erzielt werden können.

Danach wurde gezeigt, wie ein evolutionärer Algorithmus zur Selbstlokalisation eingesetzt werden kann. Mit genetischem Programmieren wurde eine Funktion evolviert, die aus den aktuellen Sensorinformationen die Position des Roboters berechnet. Diese Funktion stellt eine interne Repräsentation der Umgebung des Roboters dar. Dies ist vermutlich das erste Mal, daß ein evolutionärer Algorithmus im Bereich der Selbstlokalisation eines mobilen Roboters eingesetzt wurde.

In einem zweimonatigen Experiment mit einem mobilen Service-Roboter wurde demonstriert, daß genetisches Programmieren zur automatischen Programmierung mobiler Roboter verwendet werden kann. Evolviert wurde eine Kontrollarchitektur, die einen Service-Roboter anhand von Informationen, die er über seine Sonar-Sensoren erhält, steuert. Frühere Experimente anderer Gruppen wurden lediglich in der Simulation oder mit Miniatur-Robotern (die mit Infrarot-Sensoren ausgestattet waren) durchgeführt. Bevor das Experiment an der Universität Tübingen durchgeführt wurde, war nicht klar, ob sich die in der Simulation gewonnenen Ergebnisse auch mit einem größeren Roboter reproduzieren lassen.

Nachdem ein abstandsbasierter Merkmalsdetektor zur Steuerung eines mobilen Roboters evolviert wurde, wurde genetisches Programmieren eingesetzt, um visuelle Merkmale aus Bildern zu extrahieren. Evolutionäre Algorithmen wurden relativ selten in der Bildverarbeitung eingesetzt. Seit kurzem hat sich dieser Bereich zu einem aktiven Forschungsgebiet entwickelt.

Evolviert wurde ein Kantendetektor, ein Operator zur Extraktion markanter Punkte und ein Operator, der sich zur Berechnung des optischen Flusses eignet. Bei der Evolution eines Kantendetektors wurde die Frage untersucht, ob die Evolution eventuell benötigte Operatoren aus einfachen Operationen selbst zusammensetzen kann. Dies wurde anhand der Evolution einer Approximation des Canny-Kantendetektors gezeigt, bei der der Evolution nur relativ einfache elementare Operationen als Bausteine zur Verfügung standen. Anschließend wurde ein Operator zur Extraktion markanter Punkte evolviert. Welche Punkte in einem Bild als markant betrachtet werden, hängt immer von der jeweiligen Umgebung und dem Algorithmus ab, der diese anschließend verarbeitet. Denn die Bildverarbeitung erfolgt nicht nur zum Selbstzweck, sondern dient immer einer ganz bestimmten Aufgabe, wie z.B. der Steuerung eines mobilen Roboters. In der vorliegenden Arbeit wird gezeigt, wie mit Hilfe von genetischem Programmieren aufgabenspezifische Operatoren für die Bildverarbeitung evolviert werden können. Dies wurde anhand der Evolution eines Operators zur Berechnung des optischen Flusses demonstriert.

Wesentlicher Bestandteil der Arbeit ist ein Verfahren, das zur visuellen Steuerung eines mobilen Roboters entwickelt wurde. Das Verfahren wurde stark durch das visuelle System des Menschen inspiriert. Damit wurden die Eigenschaften eines biologischen Systems zur Realisierung eines technischen Systems übertragen. Da das visuelle System des Menschen ein Produkt der natürlichen Evolution ist, kann man erwarten, daß es in vielerlei Hinsicht optimal arbeitet. Zur Steuerung des mobilen Roboters wird für markante Punkte im Bild der durch die translatorische Bewegung induzierte optische Fluß berechnet. Durch eine Kompensation der Eigenbewegung, die auf dem Reafferenzprinzip basiert, wird die Verarbeitung visueller Informationen für beliebige Bewegungen der Kamera ermöglicht. Der optische Fluß wird in den komplex-logarithmischen Raum transformiert. Auch die Transformation in den komplex-logarithmischen Raum ist durch das biologische Vorbild inspiriert. Die Information der linken und rechten periphären Bereiche wird eingesetzt, um den Roboter in der Mitte eines Ganges zu halten. Am Beispiel der Verarbeitung visueller Informationen wurde gezeigt, daß sich durch eine genaue Analyse des biologischen Systems wichtige Erkenntnisse auch auf ein technisches System übertragen lassen. Dies ist vor allem interessant, da es immer noch viele Bereiche gibt, in denen das biologische System technischen Realisierungen deutlich überlegen ist.

Danksagung

Die vorliegende Arbeit wurde vom 01.10.1996 bis 30.09.1998 durch ein Stipendium nach dem Landesgraduiertenförderungsgesetz an den Autor gefördert. Für diese Unterstützung danke ich der Universität Tübingen von ganzem Herzen. Sie ermöglichte mir, meine Ideen zu entwickeln und schließlich umzusetzen. Herrn Prof. Zell möchte ich dafür danken, daß er meine Arbeit wissenschaftlich betreute, mich anspornte und durch seine konstruktiven Beiträge die Publikationen ganz wesentlich verbesserte. Herrn Prof. Straßer danke ich ebenfalls für seine Unterstützung.

Zur Evolution der Merkmalsdetektoren wurde genetisches Programmieren eingesetzt. Die hier beschriebenen Experimente wurden alle mit dem lil-gp Programming System von Zongker und Punch [333] durchgeführt. Dabei handelt es sich um eine ausgezeichnete Experimentierumgebung, die an der Michigan State University entstanden ist. Den Autoren dieses Systems danke ich an dieser Stelle dafür, daß sie ihr System veröffentlichten.

Für die Bildverarbeitung wurde Vista eingesetzt. Vista wurde von Pope und Lowe entwickelt [239]. Vista kann leicht um eigene Routinen erweitert werden und ermöglicht so den Aufbau einer einheitlichen Programmbibliothek für die unterschiedlichsten Anwendungen im Bereich der Bildverarbeitung. Auch den Autoren von Vista danke ich ganz herzlich.

Der VisionServer (das Programm, das die digitalisierten Bilder Anwendungsprogrammen des mobilen Roboters zugänglich macht) wurde gemeinsam mit Stefan Feyrer, Oliver Schimmel und Ralf Möller nach Vorgaben von Herrn Prof. Zell erstellt. Der PanTiltServer (das Programm zur Ansteuerung der Schwenk-Neige-Einheit des mobilen Roboters) wurde von Stefan Feyrer und Oliver Schimmel geschrieben. Bug-Reports zu vom Autor erstellter Software kamen von Stefan Feyrer und Oliver Schimmel. Die Ausgabe der Benutzeroberflächen wurde unter X mit Motif realisiert [227, 226, 240, 229, 91, 125, 88, 228].

Schließlich möchte ich den ehemaligen und jetzigen Mitarbeitern von Herrn Prof. Zell danken. Vor allem dafür, daß sie auf meine Experimente Rücksicht nahmen, die in der Regel im Gang des Lehrstuhls für Rechnerarchitektur durchgeführt wurden und diesen dadurch belegten. Unter dem zweimonatigen Experiment zur Evolution einer Kontrollarchitektur hatten besonders Markus Cyprian, Roland Leichte und Alexander Mojaev zu leiden.

Zur Evolution des Umgebungsmodells, der simulierten Kontrollarchitekturen und der visuellen Merkmalsdetektoren wurden eine Reihe von PCs eingesetzt. Dafür, daß ich einen Großteil der PCs im Arbeitsbereich Rechnerarchitektur nutzen durfte, danke ich Herrn Prof. Zell. Außerdem danke ich Igor Fischer, Dirk Hönicke, Jutta Huhse, Fred Rapp und Alexander Mojaev, die mir gestatteten, ihre Rechner zu nutzen. Michael Plagge danke ich dafür, daß auch während der Vorbereitungen zum Robocup Experimente durchgeführt werden konnten. Oliver Schimmel danke ich ebenfalls für die Nutzung seines Rechners.

Jörn Ihlenburg und Tobias Frech danke ich für die Unterstützung bei einigen Experimenten zur Bewegungsverfolgung.

William B. Langdon vom Centrum voor Wiskunde en Informatica, Amsterdam, danke ich für hilfreiche Kommentare zu einer frühen Version des Artikels über die Evolution einer Kontrollarchitektur für den mobilen Roboter [86].

Ganz besonderer Dank aber gilt meinen Eltern für ihre Unterstützung.

Inhaltsverzeichnis

1 Einleitung und Motivation **1**
- 1.1 Natürliche und künstliche Evolution . . . 1
- 1.2 Evolutionäre Algorithmen . . . 2
- 1.3 Evolutionäre Robotik . . . 2
- 1.4 Gang der Arbeit . . . 2

2 Genetisches Programmieren **5**
- 2.1 Genetische Algorithmen und das Repräsentationsproblem . . . 5
- 2.2 Baumbasiertes genetisches Programmieren . . . 8
  - 2.2.1 Arbeitsweise von genetischem Programmieren . . . 9
  - 2.2.2 Funktionsweise der genetischen Operationen . . . 11
- 2.3 Genetisches Programmieren mit linearem Genotyp variabler Länge . . . 12
- 2.4 Evolution der Struktur . . . 13
- 2.5 Die Größe des Suchraumes von genetischem Programmieren . . . 15
- 2.6 Der Einfluß von Introns auf die Beschaffenheit des Suchraumes . . . 17
- 2.7 Analogie zwischen genetischem Programmieren und der natürlichen Evolution 20
- 2.8 Experimentierumgebungen für genetisches Programmieren . . . 22
- 2.9 Zusammenfassung . . . 22

3 Evolution abstandsbasierter Merkmalsdetektoren **24**
- 3.1 Lokalisation eines mobilen Roboters . . . 24
- 3.2 Evolution einer Abbildung zur Lokalisation eines mobilen Roboters . . . 26
  - 3.2.1 Terminal-Symbole . . . 26
  - 3.2.2 Elementare Funktionen . . . 27
- 3.3 Experimente . . . 27
- 3.4 Verwandte Arbeiten . . . 29
- 3.5 Zusammenfassung . . . 31

4 Evolution einer Steuerung für einen mobilen Roboter **32**
- 4.1 Verhaltensbasierte Kontrollarchitekturen . . . 34
- 4.2 Evolution verhaltensbasierter Kontrollarchitekturen . . . 35
  - 4.2.1 Terminal-Symbole . . . 36
  - 4.2.2 Elementare Funktionen . . . 37
  - 4.2.3 Berechnung der Fitneß . . . 37
- 4.3 Experimente in der Simulation . . . 38
  - 4.3.1 Die Experimentierumgebung . . . 38

4.3.2 Selbstentwickelte Individuen . . . 39
4.3.3 Simulationsergebnisse in der modellierten Umgebung des Service-Roboters 41
4.4 Experiment mit dem mobilen Service-Roboter . . . 43
4.5 Zufällige Suche nach geeigneten Individuen . . . 45
4.6 Verwandte Arbeiten . . . 45
4.7 Zusammenfassung . . . 48

5 Evolution visueller Merkmalsdetektoren 49
5.1 Existierende Operatoren zur Extraktion von markanten Punkten . . . 49
5.1.1 Kantendetektoren . . . 50
5.1.2 Orientierte Kanten . . . 51
5.1.3 Generische Nachbarschaftsoperatoren . . . 51
5.1.4 Eckenmerkmale und gekrümmte Kanten . . . 52
5.1.5 Moravec-Interest-Operator . . . 53
5.1.6 SUSAN-Merkmalsoperator . . . 54
5.1.7 Extraktion von markanten Punkten mit Gabor-Filtern . . . 55
5.1.8 Detektion von Symmetrien . . . 56
5.1.9 Statistische Methoden . . . 57
5.1.10 Geometrische Merkmale . . . 58
5.1.11 Invariante Merkmale . . . 58
5.2 Evolution von Kantendetektoren . . . 59
5.2.1 Repräsentation . . . 59
5.2.2 Experimente . . . 62
5.2.3 Zusammenfassung . . . 64
5.3 Evolution von Operatoren zur Extraktion markanter Punkte . . . 68
5.3.1 Repräsentation . . . 68
5.3.2 Experimente . . . 71
5.3.3 Zusammenfassung . . . 75
5.4 Evolution von Operatoren zur Berechnung des optischen Flusses . . . 77
5.4.1 Qualitätskriterien für Operatoren zur Extraktion von markanten Punkten . . . 77
5.4.2 Repräsentation . . . 82
5.4.3 Fitneßfunktion . . . 84
5.4.4 Experimente . . . 85
5.4.5 Vergleich der evolvierten Operatoren mit existierenden Operatoren . . . 92
5.4.6 Zusammenfassung . . . 93
5.5 Verwandte Arbeiten . . . 96
5.5.1 Adaptive Kantendetektion . . . 96
5.5.2 Visuelle Merkmalsdetektion mit neuronalen Netzen . . . 96
5.5.3 Evolutionäre Algorithmen in der Bildverarbeitung . . . 97
5.5.4 Adaptive Lösung des Korrespondenzproblems . . . 99
5.5.5 Adaptive Auswahl geeigneter Bildmerkmale . . . 99
5.5.6 Unterschiede zu den existierenden Arbeiten . . . 100
5.6 Zusammenfassung . . . 100

6 Visuelle Steuerung eines mobilen Roboters **102**
6.1 Motivation . . . 103
6.2 Das visuelle System . . . 104
6.2.1 Das Auge . . . 104
6.2.2 Die Retina . . . 105
6.2.3 Der Kern des seitlichen Kniehöckers . . . 107
6.2.4 Der visuelle Kortex (die Sehrinde) . . . 108
6.3 Die komplex-logarithmische Abbildung . . . 112
6.4 Das Reafferenzprinzip . . . 114
6.5 Visuelle Steuerung eines mobilen Roboters unter Verwendung des Reafferenzprinzips und der komplex-logarithmischen Abbildung . . . 117
6.5.1 Merkmalsextraktion . . . 117
6.5.2 Kompensation des rotatorischen Anteils der Eigenbewegung . . . 118
6.5.3 Der optische Fluß im komplex-logarithmischen Raum . . . 121
6.5.4 Steuerung des Roboters . . . 122
6.6 Experimente . . . 124
6.7 Verwandte Arbeiten . . . 127
6.7.1 Visuelle Steuerung mobiler Roboter . . . 127
6.7.2 Steuerung eines mobilen Roboters in einem Korridor . . . 129
6.7.3 Unterschiede zu den existierenden Arbeiten . . . 131
6.8 Zusammenfassung . . . 131

7 Zusammenfassung **133**

A Koza-Tableaus **136**
A.1 Evolution einer Abbildung zur Lokalisation eines mobilen Roboters . . . 137
A.2 Evolution einer Steuerung für einen mobilen Roboter . . . 138
A.3 Evolution von Kantendetektoren . . . 139
A.4 Evolution von Operatoren zur Extraktion markanter Punkte . . . 140
A.5 Evolution von Operatoren zur Berechnung des optischen Flusses . . . 141

B Existierende Operatoren zur Berechnung des optischen Flusses **142**

Literaturverzeichnis **147**

Kapitel 1

Einleitung und Motivation

Die Natur hat eine Vielzahl von Spezies produziert, die optimal an ihre Umgebung angepaßt sind. Darwin [64] hat auf seinen Reisen die unterschiedlichsten Arten vorgefunden und entwickelte eine Theorie über den Mechanismus, der diese Artenvielfalt hervorgebracht hat: die natürliche Auswahl. Die Arbeiten Darwins führten zu einer Reformierung der Biologie, weg von einer bloßen Katalogisierung unveränderlicher Arten und hin zu einer theoretischen Grundlage. Analysiert man die grundlegenden Eigenschaften der Evolution, so kann das Verfahren mit einem Computer simuliert werden. Durch Simulation solch komplexer Systeme, kann untersucht werden, welchen fundamentalen Gesetzmäßigkeiten die Systeme unterliegen [10, 154]. Ferner können durch eine Modellierung natürlicher Systeme eventuell Fragen der theoretischen Biologie beantwortet werden [294, 26]. Dieser Ansatz wird im Englischen als *Artificial Life* bezeichnet [173, 318].

1.1 Natürliche und künstliche Evolution

Maynard Smith [195] gibt eine sehr gute Einführung in die Theorie der natürlichen Evolution. Banzhaf et al. [16] führten die von Darwin und Maynard Smith genannten minimalen Bedingungen auf, damit Evolution stattfinden kann:

- Reproduktion der Individuen,
- Variation der Individuen, die einen Einfluß auf die Überlebenswahrscheinlichkeit hat,
- Vererbung in der Reproduktion, d.h. die Nachkommen haben ähnliche Eigenschaften wie ihre Eltern,
- begrenzte Ressourcen, die einen Wettbewerb um die Ressourcen auslösen.

Als Darwin sein Werk über die natürliche Selektion verfaßte, war noch nicht klar, wie der biologische Mechanismus der Vererbung arbeitet. Die Funktionsweise wurde erst sehr viel später entdeckt [67]. Urspünglich wurde vermutet, daß die einzelnen Eigenschaften über die Generationen hinweg herausgemittelt würden. Mit der Entdeckung, daß einzelne Faktoren der Eltern an ihre Kinder vererbt werden, war klar, daß es sich um einen diskreten Mechanismus handelt.

Faßt man die Reproduktion als die Erzeugung einer Kopie des ursprünglichen Individuums auf, so ist damit automatisch auch die Vererbung der Eigenschaften gegeben. Daher sind die

minimal erforderlichen genetischen Operatoren die Reproduktion, die Variation und die Selektion. Reproduktion kann sowohl sexuell als auch asexuell stattfinden. Als Operatoren, die die Individuen variieren, werden meist die Crossover-Operation und die Mutations-Operation eingesetzt.

1.2 Evolutionäre Algorithmen

In der Regel werden evolutionäre Algorithmen [98, 97, 99, 126, 109, 202, 247, 165, 167, 16] zur Lösung eines Optimierungsproblems verwendet. Die Qualität der Individuen wird durch eine Fitneßfunktion definiert. Bei einigen Verfahren wird die Fitneß explizit berechnet, bei anderen ist sie nur implizit gegeben. Letzteres wird als *Open-Ended Evolution*, also als Evolution mit offenem Ausgang bezeichnet [242, 279].

Wird ein evolutionärer Algorithmus eingesetzt, um das Optimum einer Funktion zu suchen, dann muß zunächst eine geeignete Kodierung des Problems definiert werden. Jedes Individuum stellt einen Punkt im Suchraum dar. Der Anwender muß nun eine Fitneßfunktion definieren, die für alle möglichen Individuen die Fitneß berechnet. Gute Individuen werden verstärkt selektiert und erzeugen so mehr Nachkommen als weniger gute Individuen. Durch die Veränderung der Individuen, wie z.B. Crossover oder Mutation, werden neue Punkte des Suchraumes untersucht. Mit der Crossover-Operation können zwei oder mehrere günstige Eigenschaften miteinander kombiniert werden. Um das gleiche nur mit der Mutation zu erreichen, wären unter Umständen sehr viele Einzelmutationen erforderlich [195, 154].

1.3 Evolutionäre Robotik

Die natürliche Evolution ist ein Verfahren, das nachweislich intelligentes Verhalten erzeugt [242]. Da in der Robotik bisher nichts Vergleichbares durch manuelle Programmierung produziert wurde, ist die Hoffnung groß, durch den Einsatz evolutionärer Algorithmen größere Fortschritte zu erzielen. Interessant ist in diesem Zusammenhang z.B. die Evolution von intelligentem Verhalten [266], die Evolution von Lernfähigkeit [46], die Evolution des Auges [217] und die Evolution des visuellen Systems [186, 130]. Der Einsatz evolutionärer Algorithmen in der Robotik wird als *Evolutionary Robotics* bezeichnet [124, 219, 194, 111, 112, 137]. Einige Gedankenexperimente zu diesem Gebiet wurden bereits sehr früh von Braitenberg [28] durchgeführt.

Von besonderem Interesse ist genetisches Programmieren [165, 167, 16], eine Klasse evolutionärer Algorithmen, die zur automatischen Programmierung eingesetzt werden kann. In der vorliegenden Arbeit wurde eine Vielzahl von Experimenten mit genetischem Programmieren durchgeführt. Im besonderen geht es um die Evolution von Merkmalsdetektoren zur Steuerung eines mobilen Roboters. Siehe [197, 148, 177, 267, 9] für eine Einführung in autonome mobile Roboter. In der vorliegenden Arbeit wird untersucht, wie mit genetischem Programmieren Merkmalsdetektoren evolviert werden können, die zur Steuerung eines mobilen Roboters genutzt werden können.

1.4 Gang der Arbeit

Die Arbeit gliedert sich wie folgt. Zunächst wird eine kurze Einführung in verschiedene evolutionäre Algorithmen gegeben. Dabei werden zunächst die Funktionsweise genetischer Algo-

rithmen erklärt, das Repräsentationsproblem [165] diskutiert und begründet, warum in der vorliegenden Arbeit genetisches Programmieren eingesetzt wird. Danach werden verschiedene Arten von genetischem Programmieren beschrieben. Als erstes wird Koza's baumbasiertes genetisches Programmieren [165, 167], danach Banzhaf et al.'s [16] genetisches Programmieren, das mit einem linearen Genotyp variabler Länge arbeitet und schließlich Lohmann's *Structure Evolution* vorgestellt. Die in dieser Arbeit beschriebenen Experimente wurden mit Koza's baumbasiertem genetischem Programmieren durchgeführt. Der Einfluß von Introns auf den Suchraum von genetischem Programmieren wird analysiert. Ferner wird eine Analogie zwischen genetischem Programmieren und der natürlichen Evolution gezogen.

Als erstes wird durch genetisches Programmieren ein abstandsbasierter Merkmalsdetektor evolviert, der die aktuelle Position eines mobilen Roboters berechnet. Evolviert wird ein Modell der Umgebung, wie sie über Sonar-Sensoren oder mit einem Laser-Scanner wahrgenommen wird. Die evolvierte Funktion berechnet für die gegebenen Abstandswerte der Sensoren die Position des Roboters. Ein Merkmalsdetektor, der die Position eines mobilen Roboters bestimmt, kann bei hinreichender Genauigkeit zur Relokalisation während der Steuerung eines Roboters eingesetzt werden.

Nachdem zunächst lediglich ein Merkmalsdetektor evolviert wurde, der die Position des Roboters ermittelt, diesen aber nicht steuert, wird im folgenden Kapitel ein vollständiges Programm, eine Kontrollarchitektur, evolviert. Aufgabe dieser Kontrollarchitektur ist es, den mobilen Roboter einen Gang entlang zu steuern. Die evolvierte Kontrollarchitektur bildet die gegebenen Abstandswerte direkt auf Steuerkommandos für den Roboter ab. So werden implizit für die Umgebung des Roboters relevante Merkmalsdetektoren und die zugehörigen Steuerkommandos evolviert. Die Experimente wurden zunächst in der Simulation durchgeführt und schließlich mit einem mobilen Service-Roboter reproduziert.

Danach werden visuelle Merkmalsdetektoren evolviert, die zur visuellen Steuerung eines mobilen Roboters genutzt werden sollen. Zunächst wird durch genetisches Programmieren ein Kantendetektor evolviert, der den Canny-Kantendetektor [141] approximiert. Danach wird ein Operator zur Extraktion markanter Punkte evolviert. Der evolvierte Operator approximiert den Moravec-Operator [205, 206]. Sowohl bei der Evolution des Kantendetektors als auch bei der Evolution des Operators zur Extraktion markanter Punkte wird bei der Berechnung der Fitneß eines Individuums ein bereits existierender Operator eingesetzt. Bei der Evolution eines Kantendetektors wird der Canny-Kantendetektor und bei der Evolution eines Operators zur Extraktion markanter Punkte wird der Moravec-Operator zur Berechnung der Fitneß herangezogen.

Ziel ist es jedoch, einen Operator zu evolvieren, der optimal an Aufgabe und durch die Umgebung des Roboters gegebenen Randbedingungen angepaßt ist. Es soll ein Operator evolviert werden, mit dessen Hilfe der optische Fluß für markante Punkte des Bildes berechnet werden kann. Daher werden eine Reihe von Kriterien definiert, die ein Operator, der zur Lösung dieses Problems eingesetzt wird, erfüllen sollte. Dabei wird kein bereits existierender Operator zur Berechnung der Fitneß herangezogen. Die Bewertung der Individuen erfolgt lediglich aufgrund der Qualitätskriterien und dem Vergleich mit anderen Individuen derselben Generation. So wurde ein Operator evolviert, der optimal bzw. annähernd optimal an die Aufgabe und die gegebenen Randbedingungen angepaßt ist.

Nachdem ein visueller Merkmalsdetektor evolviert wurde, wird ein Algorithmus zur Steuerung eines mobilen Roboters entwickelt. Der Algorithmus berechnet für die extrahierten Merkmale den optischen Fluß und steuert so den mobilen Roboter. Das Verfahren ist stark biologisch motiviert. Als Vorbild dient das visuelle System des Menschen. Daher werden

zunächst die wesentlichen Eigenschaften des visuellen Systems kurz beschrieben. In Analogie zum visuellen System des Menschen werden die Eigenschaften der komplex-logarithmischen Abbildung [272, 271, 140, 141] genutzt. Da die komplex-logarithmische Abbildung jedoch nur für einen translatierenden Beobachter definiert ist, wird die bekannte Eigenbewegung des Roboters genutzt, um die rotatorische Bewegung des Roboters zu kompensieren. Der Mechanismus, der dies realisiert, ist bei Tieren und beim Menschen als Reafferenzprinzip [310] bekannt.

Mit Hilfe der komplex-logarithmischen Abbildung und des Reafferenzprinzips wird ein System zur Steuerung eines mobilen Roboters entwickelt. Als erstes werden markante Punkte aus einer Bildsequenz extrahiert, für die der optische Fluß berechnet wird. Dann wird der optische Fluß in den komplex-logarithmischen Raum transformiert. Der optische Fluß im komplex-logarithmischen Raum ist ein Maß für den Abstand der Punkte zur Kamera [140]. Durch den Vergleich des optischen Flusses der linken und rechten periphären Bereiche wird der Roboter in der Mitte eines Korridors gehalten. Die translatorische Geschwindigkeit des Roboters wird durch den Betrag des wahrgenommenen optischen Flusses geregelt. Das Verfahren zur Steuerung des Roboters, basierend auf visuell extrahierter Merkmale, wird unter Verwendung des Moravec-Operators anhand mehrerer Experimente demonstriert.

Die Arbeit schließt mit einer Zusammenfassung, in der die wichtigsten Ergebnisse aufgeführt sind.

Kapitel 2

Genetisches Programmieren

Mit evolutionären Algorithmen kann die natürliche Evolution mit dem Computer simuliert werden. Die künstliche Evolution arbeitet dabei auf einer Population von Individuen. Jedes der Individuen ist eine mehr oder weniger gute Lösung des Problems. Durch Anwendung genetischer Operatoren wie Reproduktion, Variation und Selektion werden Individuen erzeugt, die optimal an ihre Umgebung angepaßt sind.

2.1 Genetische Algorithmen und das Repräsentationsproblem

Genetische Algorithmen [126, 109, 202] arbeiten in der Regel mit dem in Abbildung 2.1 dargestellten Algorithmus. Die einzelnen Individuen werden als Zeichenkette fester Länge kodiert. Die Zeichenkette soll in Analogie zur natürlichen Evolution ein Chromosom darstellen, das aus mehreren Genen besteht. Die möglichen Werte, die ein Gen an einer bestimmten Position des Chromosoms annehmen kann, werden als Allele bezeichnet. Das genetische Material eines Individuums wird als Genotyp bezeichnet und das daraus entstehende Individuum als Phänotyp.

Bei genetischen Algorithmen werden die Individuen meist als Bitstring kodiert. Wird z.B. das Optimum einer Funktion gesucht, die von den n Parametern x_i ($i \in \{1, ..., n\}$) abhängt, dann werden in jedem Individuum die n Parameter kodiert. Dazu wird zunächst der Suchraum diskretisiert. In Abbildung 2.2 ist ein Beispiel für ein Individuum angegeben. Das Individuum besteht aus drei Parametern, die jeweils mit 8 Bit kodiert werden. Der Suchraum umfaßt in diesem Fall genau 2^{24} Punkte. Je nach erforderlicher Genauigkeit werden für jeden Parameter b_i Bits zur Kodierung des Wertes verwendet. Das Individuum setzt sich aus den einzelnen Bits der Parameter zusammen und hat die Länge $l = \sum_i b_i$. Der Suchraum hat 2^l Punkte.

Als erstes werden die Individuen der Population zufällig initialisiert, d.h. alle Bits der Individuen werden mit Hilfe eines Pseudo-Zufallszahlen-Generators mit einem zufälligen Wert initialisiert. Anschließend wird für jedes Individuum die Fitneß berechnet. Dazu werden die in dem Individuum vorhandenen Parameter dekodiert und in die zu optimierende Funktion eingesetzt. Wird das Maximum der Funktion gesucht, so kann dieser Wert direkt als Fitneß des Individuums verwendet werden. Bei einer Minimierung muß der Wert je nach verwendeter Art der Selektion evtl. noch transformiert werden.

Nachdem die Fitneß aller Individuen der Population bestimmt wurde, wird eine neue Generation von Individuen erzeugt. Dazu werden mit einem Selektionsverfahren zwei Individuen ausgewählt. Einige Standardverfahren zur Selektion sind die fitneßproportionale

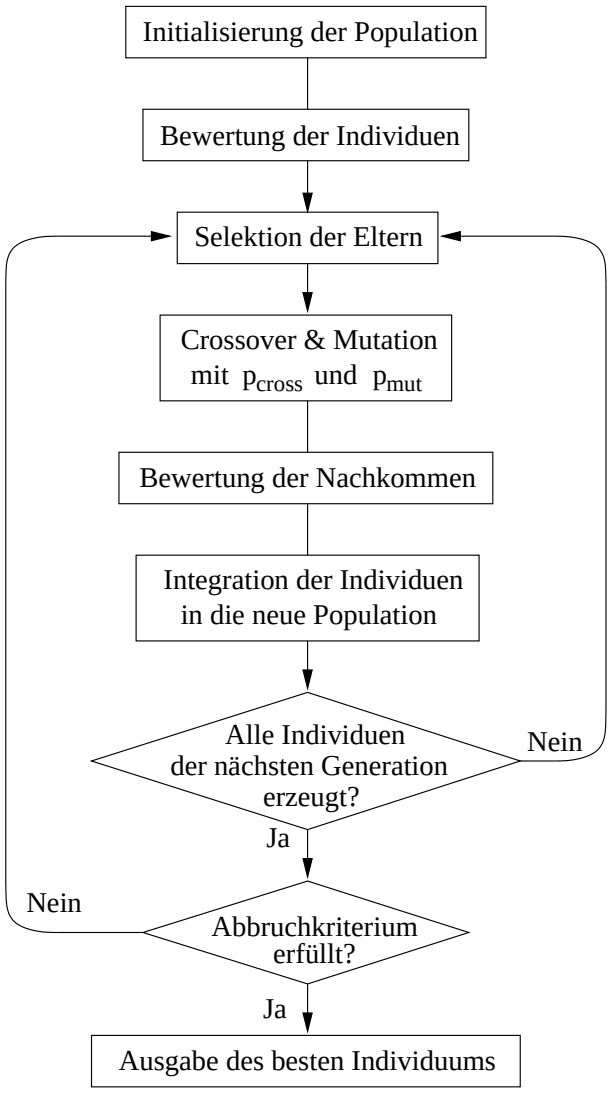


Abbildung 2.1: Flußdiagramm eines typischen genetischen Algorithmus (siehe [126, 109, 202]). Eine Population von Individuen wird zufällig initialisiert. Danach wird für jedes Individuum die Fitneß berechnet. Anschließend werden zwei Eltern selektiert, auf die mit der Wahrscheinlichkeit p_{cross} die Crossover-Operation angewandt wird. Die einzelnen Bits werden dann mit der Wahrscheinlichkeit p_{mut} mutiert. Die neuen Individuen werden bewertet und in die nächste Population integriert. Dies wird solange wiederholt, bis die neue Generation gefüllt ist. Falls das Abbruchkriterium nicht erfüllt ist, werden weitere Generationen erzeugt. Ist das Abbruchkriterium erfüllt, dann ist das beste gefundene Individuum das Ergebnis des Laufes.

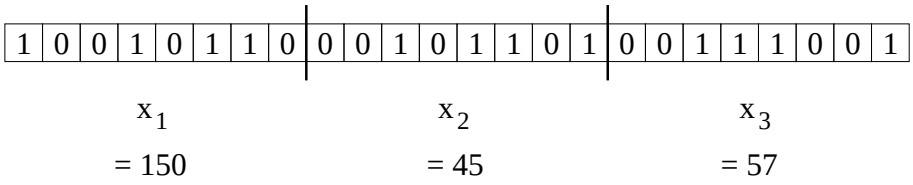


Abbildung 2.2: Beispiel für die Kodierung eines Individuums, das aus drei Werten besteht. Für jeden Wert wurden hier jeweils 8 Bit verwendet. Die dezimale Dekodierung des Individuums ist ebenfalls angegeben.

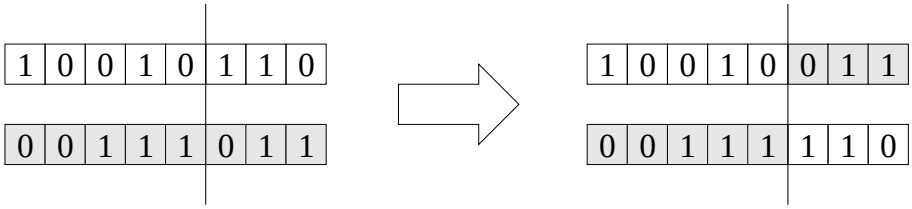


Abbildung 2.3: Crossover-Operation eines genetischen Algorithmus am Beispiel eines binären Genotyps. Zunächst wird eine Position (beim Ein-Punkt-Crossover) innerhalb des Bitstrings zufällig ausgewählt. Durch den Austausch der Teilbäume werden zwei Nachkommen erzeugt.

Selektion und die Tournament-Selektion. Bei der fitneßproportionalen Selektion werden die Individuen gemäß der Wahrscheinlichkeiten $p_i = \frac{f_i}{\sum_j f_j}$ selektiert. Dabei ist p_i die Selektionswahrscheinlichkeit von Individuum i und f_i gibt die Fitneß des Individuums an. Bei der Tournament-Selektion wird eine kleine Zahl von Individuen aus der Population zufällig ausgewählt und das beste dieser Individuen wird als Eltern-Individuum selektiert.

So werden zwei Eltern selektiert und mit der Wahrscheinlichkeit p_{cross} eine Crossover-Operation durchgeführt. Ein Beispiel für eine Crossover-Operation ist in Abbildung 2.3 zu sehen. Die beiden Individuen werden aneinandergereiht, und es wird eine zufällige Position ausgewählt (beim Ein-Punkt-Crossover), ab der die Bitwerte vertauscht werden. Durch die Crossover-Operation sollen einzelne geeignete Bausteine der Eltern-Individuen zusammengefügt werden. Neben dem Ein-Punkt-Crossover gibt es noch weitere Möglichkeiten, ein Crossover durchzuführen. Die verschiedenen Möglichkeiten zur Variation der Nachkommen sind ausführlich in den Lehrbüchern über genetische Algorithmen beschrieben [126, 109, 202, 268].

Danach werden die Nachkommen der Eltern mutiert. Wurden keine Nachkommen erzeugt, werden die Eltern-Individuen mutiert. Dazu werden die einzelnen Bits mit der Wahrschein-

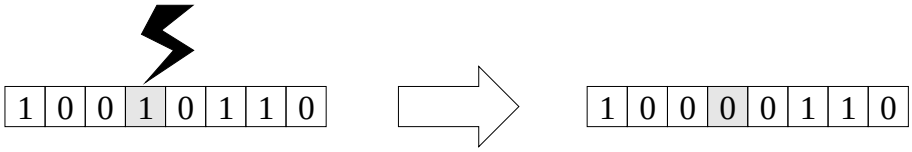


Abbildung 2.4: Mutations-Operation eines genetischen Algorithmus am Beispiel eines binären Genotyps. Ein Bit wird auf einen zufälligen Wert gesetzt.

lichkeit p_{mut} auf einen zufälligen Wert gesetzt. Die Mutations-Operation ist in Abbildung 2.4 dargestellt. Die Mutations-Operation wird eingesetzt, damit der gesamte Suchraum für die Individuen erreichbar bleibt. Wird lediglich die Crossover-Operation und Selektion eingesetzt, so könnte passieren, daß genetisches Material verlorengeht und dadurch Teile des Suchraumes nicht mehr erreicht werden können. Nach der Mutation wird die Fitneß der Individuen berechnet. Anschließend werden die Individuen in die neue Generation eingefügt. Dies wird solange wiederholt, bis die neue Generation gefüllt ist.

Es werden solange neue Generationen erzeugt, bis ein Abbruchkriterium erfüllt ist. Als Abbruchkriterium kann z.B. das Erreichen einer bestimmten Generation, die Erfüllung von Qualitätskriterien der Lösung oder die Konvergenz der Population eingesetzt werden. Nach Abbruch des Algorithmus wird das beste Individuum als gefundene Lösung des Problems ausgegeben. Dabei muß es sich jedoch nicht um das globale Optimum handeln. Oft kommt es vor, daß nur eine annähernd optimale Lösung gefunden wird.

Von Koza [165] wird das Repräsentationsproblem bei genetischen Algorithmen ausführlich diskutiert. Bei genetischen Algorithmen wird mit Individuen fester Länge gearbeitet. Der Suchraum ist von vornherein stark eingeschränkt. Meist werden genetische Algorithmen eingesetzt, um nach optimalen Parametern der Fitneßfunktion zu suchen. Zur Evolution von Programmen bietet sich daher eine andere Repräsentation an. Computerprogramme besitzen oft eine hierarchische Struktur und bestehen aus einer Vielzahl von Unterroutinen. Die Größe des Programmes, das eine Lösung des Problems darstellt, kann im voraus nicht exakt definiert werden. Daher sollte es möglich sein, Programme variabler Länge zu evolvieren. Bei der Compilierung von Programmen wird intern ein sogenannter Parse-Baum [2] erzeugt. Von Koza wurde daher eine Baumstruktur zur Evolution von Programmen eingesetzt. Koza prägte den Begriff genetisches Programmieren und zeigte, daß baumbasiertes genetisches Programmieren zur Lösung der unterschiedlichsten Probleme eingesetzt werden kann [165, 167].

2.2 Baumbasiertes genetisches Programmieren

Koza verwendete die Programmiersprache LISP, um Programme zu evolvieren [165, 167]. Ein LISP-Programm kann als Baum dargestellt werden. Diese Struktur ist äquivalent zu den Parse-Bäumen, die von Compilern anderer Programmiersprachen erzeugt werden. Programm und Daten haben die gleiche Form und können daher gleichzeitig evolviert werden. Die evolvierten Programme können direkt ausgewertet werden. Es findet also keine Dekodierung statt. In diesem Fall ist das Programm der Genotyp und das Verhalten des Programmes der Phänotyp [16]. Dies muß aber nicht immer so sein. Gruau [116] setzte genetisches Programmieren zur Evolution von neuronalen Netzen ein (siehe [258, 196, 5, 4, 164, 216, 328] für eine Einführung in neuronale Netze). Evolviert wurde eine Grammatik, in Form eines Baumes, aus der sich schließlich der Verbindungsgraph eines neuronalen Netzes entwickelt.

Beim genetischen Programmieren werden die inneren Knoten des Baumes als elementare Funktionen und die äußeren Knoten als Terminal-Symbole bezeichnet. Beim baumbasierten genetischem Programmieren wird oft ein symbolischer Ausdruck für eine Funktion evolviert. Viele Probleme können auf die Evolution eines symbolischen Ausdrucks zurückgeführt werden. Wird ein symbolischer Ausdruck evolviert, werden üblicherweise die arithmetischen Funktionen Addition (`+`), Subtraktion (`-`), Multiplikation (`*`) und Division (`/`) als elementare Funktionen verwendet. Als Terminal-Symbole werden die Parameter der Funktion und einige Konstanten verwendet.

Bei den hier angegebenen Beispielen für die Arbeitsweise der genetischen Operatoren wird angenommen, daß ein symbolischer Ausdruck für eine Funktion evolviert wird. Als elementare Funktionen werden die oben angegebenen arithmetischen Funktionen verwendet. Als Terminal-Symbole werden das Argument `X` sowie die Konstanten `1`, `2`, `3`, `4` und `5` verwendet. Da die Division durch Null nicht definiert ist, muß dies in geeigneter Weise abgefangen werden. Daher wird beim genetischen Programmieren meist bei Division durch Null der Wert Eins zurückgeliefert. Es wäre aber genauso möglich, daß in diesem Fall ein symbolischer Wert zurückgeliefert wird, der die Division durch Null anzeigt. Dann müßten die anderen Funktionen entsprechend angepaßt werden, um den Fehler entsprechend weiterzugeben.

2.2.1 Arbeitsweise von genetischem Programmieren

Die Arbeitsweise von genetischem Programmieren ist in Abbildung 2.5 graphisch dargestellt. Als erstes wird die Population zufällig initialisiert. Von Koza wurden drei unterschiedliche Verfahren zur Erzeugung eines Individuums beschrieben. Beim ersten Verfahren wird ein Baum erzeugt, dessen äußeren Knoten alle die gewünschte Tiefe besitzen. Dazu werden, beginnend bei der Wurzel des Baumes, solange rekursiv Knoten aus der Menge der elementaren Funktionen ausgewählt, bis die gewünschte Tiefe des Baumes erreicht ist. Ist die gewünschte Tiefe erreicht, werden die Knoten nur noch aus den Terminal-Symbolen ausgewählt. Beim zweiten Verfahren werden, ausgehend von der Wurzel des Baumes, zufällig Knoten sowohl aus der Menge der elementaren Funktionen als auch aus der Menge der Terminal-Symbole ausgewählt. Ist eine gewünschte Tiefe erreicht, werden nur noch Knoten aus der Menge der Terminal-Symbole ausgewählt. Während beim ersten Verfahren die äußeren Knoten alle die gleiche Tiefe besitzen, ist dies beim zweiten Verfahren nicht der Fall. Die durchschnittliche gewünschte Tiefe der Individuen unterscheidet sich je nach Zahl der elementaren Funktionen und Zahl der verwendeten Terminal-Symbole. Das dritte Verfahren ist eine Mischung der beiden ersten Verfahren. Dabei wird mit einer Wahrscheinlichkeit von 50% das erste Verfahren und mit einer Wahrscheinlichkeit von 50% das zweite Verfahren verwendet. Das dritte Verfahren wird daher 50:50 Regel genannt. Meist wird zur Erzeugung der Individuen ein Bereich für die gewünschte Tiefe der Individuen angegeben. In diesem Fall werden die Individuen gleichmäßig über den Bereich verteilt und die 50:50 Regel auf jede Tiefe innerhalb des Bereiches einzeln angewandt. Während des Laufes wird in der Regel die maximale Tiefe der Bäume und die Zahl der Knoten je Individuum beschränkt. Daher ist der Suchraum letztendlich doch nur endlich groß. Oft ist die maximale Tiefe der Bäume auf 17 beschränkt und es werden höchstens 1000 Knoten pro Individuum zugelassen.

Nachdem die Individuen der ersten Population erzeugt wurden, wird die Fitneß der Individuen berechnet. Anschließend wird eine neue Generation von Individuen erzeugt. Dazu wird als erstes ein genetischer Operator ausgewählt. Oft werden als genetische Operatoren Crossover, Mutation und Reproduktion verwendet. Die Operatoren werden mit den Wahrscheinlichkeiten p_{cross}, p_{mut} und p_{rep} selektiert (wobei $p_{\text{cross}} + p_{\text{mut}} + p_{\text{rep}} = 1$). Neben diesen drei Operatoren existieren noch weitere, die ebenfalls eingesetzt werden können [165, 16]. Da in der vorliegenden Arbeit Crossover, Mutation und Reproduktion verwendet wurden, sind sie hier kurz beschrieben. Je nach Operator werden entweder ein oder mehrere Eltern-Individuen aus der Eltern-Population ausgewählt. Hierzu kann wieder eines der üblichen Verfahren, wie z.B. fitneßproportionale Selektion oder die Tournament-Selektion verwendet werden. Koza definierte außerdem noch die verstärkte fitneßproportionale Selektion. Dabei wird die Population als erstes in zwei Gruppen eingeteilt. In die erste Gruppe kommen die Individuen, die

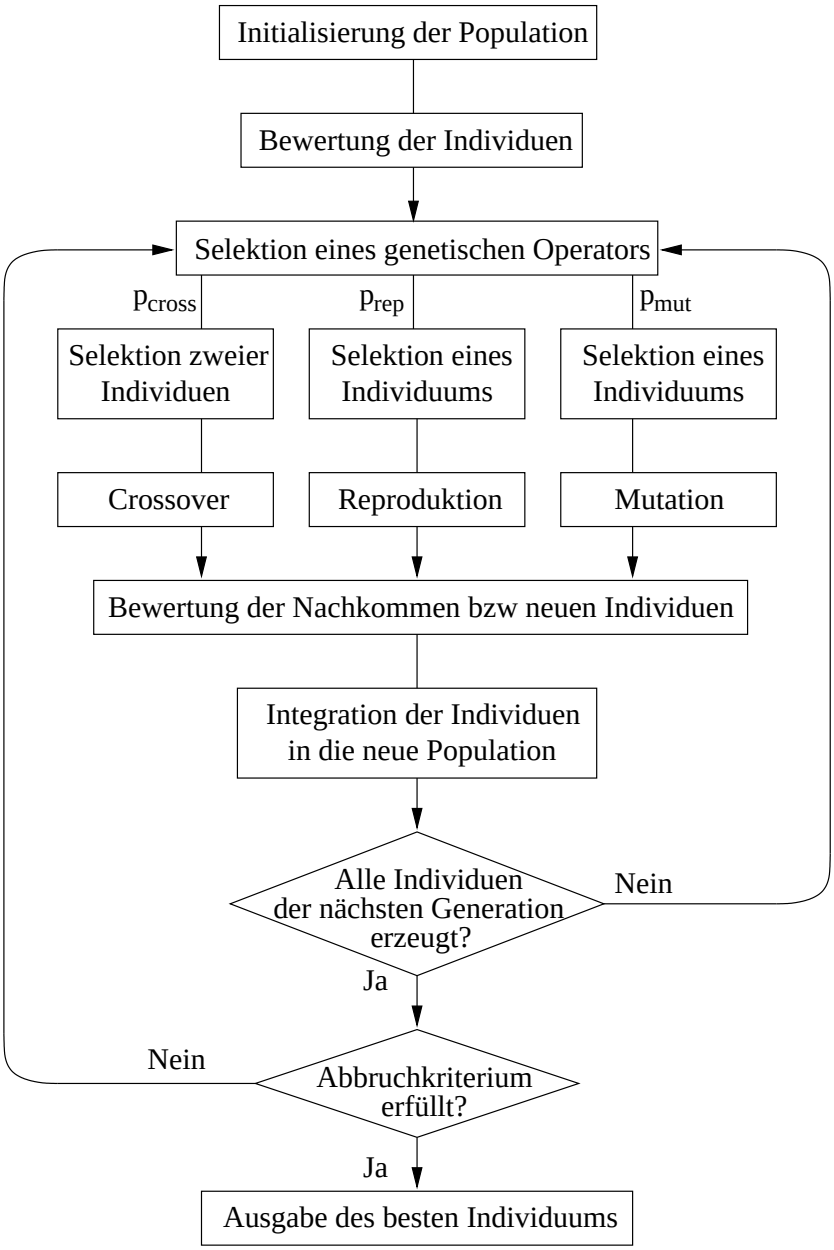


Abbildung 2.5: Flußdiagramm von genetischem Programmieren (nach Koza [165, 167] siehe auch Banzhaf et al. [16]). Eine Population von Individuen wird zufällig initialisiert. Danach wird für jedes Individuum die Fitneß berechnet. Anschließend wird ein genetischer Operator (z.B. Crossover, Reproduktion oder Mutation, jeweils mit den Wahrscheinlichkeiten p_{cross}, p_{rep} bzw. p_{mut}) zufällig ausgewählt. Die daraus resultierenden Individuen werden bewertet und in die neue Population integriert. Dies wird solange wiederholt, bis die neue Generation gefüllt ist. Falls das Abbruchkriterium nicht erfüllt ist, werden weitere Generationen erzeugt. Ist das Abbruchkriterium erfüllt, dann ist das beste gefundene Individuum das Ergebnis des Laufes.

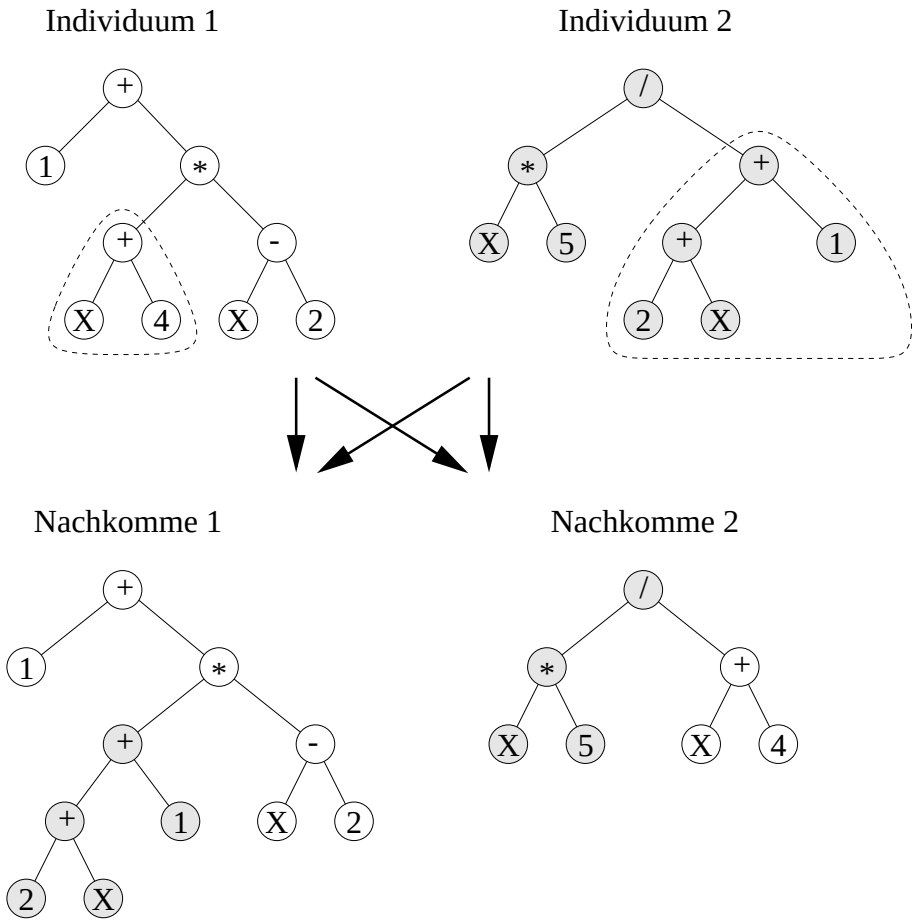


Abbildung 2.6: Crossover-Operation für baumbasiertes genetisches Programmieren. In jedem der beiden selektierten Individuen wird ein Teilbaum ausgewählt. Durch den Austausch der Teilbäume werden zwei Nachkommen erzeugt.

einen bestimmten Prozentsatz der Summe aller Fitneßwerte ausmachen. Die Individuen werden in der Reihenfolge ihrer Fitneß in diese Gruppe integriert. Der Prozentsatz c wird aus der Populationsgröße berechnet $c = \frac{320}{n}$ wobei n die Zahl der Individuen angibt. Individuen der ersten Gruppe werden in 80% der Fälle, Individuen der zweiten in 20% der Fälle selektiert.

2.2.2 Funktionsweise der genetischen Operationen

Ein Beispiel für die Funktionsweise der Crossover-Operation ist in Abbildung 2.6 dargestellt. Zwei Eltern werden aus der Population ausgewählt. Dann wird in jedem der beiden Individuen ein Teilbaum zufällig ausgewählt, und die beiden Teilbäume ausgetauscht. Daher werden durch eine Crossover-Operation zwei Nachkommen produziert. Durch die Crossover-Operation sollen, so hofft man, Unterroutinen, die einen positiven Einfluß auf die Fitneß des Individuums haben, zusammengefügt werden. Experimentelle Untersuchungen zeigten jedoch, daß der Großteil der durchgeführten Crossover-Operationen die Fitneß der Nachkommen im Vergleich zu ihren Eltern erheblich verschlechtert und nur ein geringer Teil zu einer Verbesserung führt [16]. Die Funktionsweise der Mutation ist in Abbildung 2.7 dargestellt.

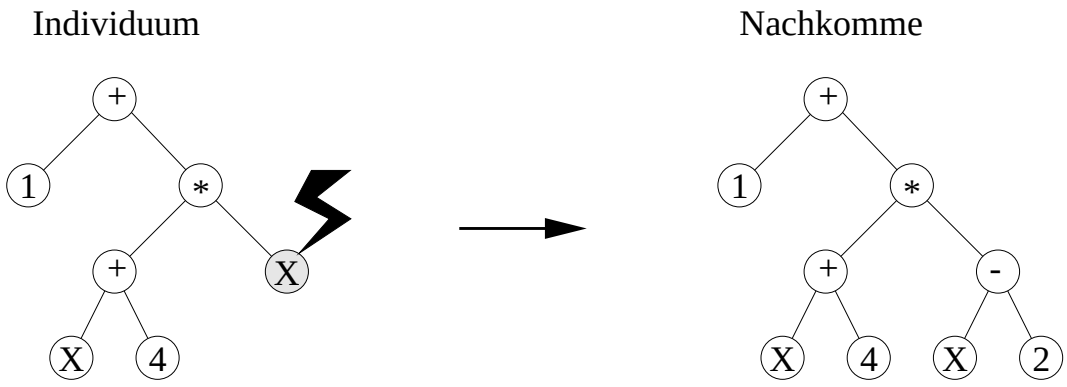


Abbildung 2.7: Mutations-Operation für baumbasiertes genetisches Programmieren. Im selektierten Individuum wird ein Teilbaum ausgewählt und dann neu generiert.

Bei der Mutation eines Individuums wird ein Teilbaum zufällig ausgewählt und durch einen neu erzeugten Teilbaum ersetzt. Das resultierende Individuum wird anschließend in die neue Population integriert. Bei der Reproduktions-Operation wird das ausgewählte Individuum unverändert in die neue Population kopiert.

Je nach Art der genetischen Operation entstehen entweder ein oder mehrere Nachkommen. Nachdem die Fitneß der Nachkommen berechnet wurde, werden sie in die neue Population integriert. Es werden solange neue Nachkommen erzeugt, bis die neue Population gefüllt ist. Danach wird das Abbruchkriterium überprüft. Falls das Abbruchkriterium nicht erfüllt ist, wird eine neue Generation von Individuen erzeugt. Ist das Abbruchkriterium erfüllt, wird das beste Individuum als Ergebnis des Laufes ausgegeben.

2.3 Genetisches Programmieren mit linearem Genotyp variabler Länge

Neben dem von Koza geprägten baumbasierten genetischen Programmieren gibt es noch weitere Varianten des automatischen Programmierens. Banzhaf et al. [16] fassen den Begriff genetisches Programmieren allgemeiner. Unter genetischem Programmieren verstehen sie die Generierung von Computer-Programmen mit Hilfe evolutionärer Algorithmen. Dazu zählt die Evolution von LISP-Programmen genauso wie die Evolution von linearem Maschinencode oder die Evolution von neuronalen Netzen. Durch das automatische Programmieren mit Hilfe evolutionärer Algorithmen könnte es möglich werden, daß sich Computer "selbst" programmieren. Das Problem der Programmierung wird in diesem Fall auf die Definition einer geeigneten Fitneßfunktion zurückgeführt.

Von Nordin und Banzhaf [220, 221, 222, 223, 224], Olmer et al. [231] und Banzhaf et al. [17] wurde genetisches Programmieren erfolgreich eingesetzt, um linearen Maschinencode für einen Miniatur-Roboter, den Khepera, zu evolvieren. Ein Individuum besteht dabei aus einer Liste von elementaren Befehlen. Die einzelnen Befehle arbeiten auf Registern. Vor dem Start des Programmes werden die Parameter in die Eingaberegister geladen. Die Befehle werden dann der Reihe nach abgearbeitet. Meist wird ohne Sprungbefehle gearbeitet, da durch die Verwendung von Sprungbefehlen Endlosschleifen entstehen könnten. Nachdem das Programm vollständig abgearbeitet wurde, wird das Ergebnis der Ausgaberegister ausgelesen.

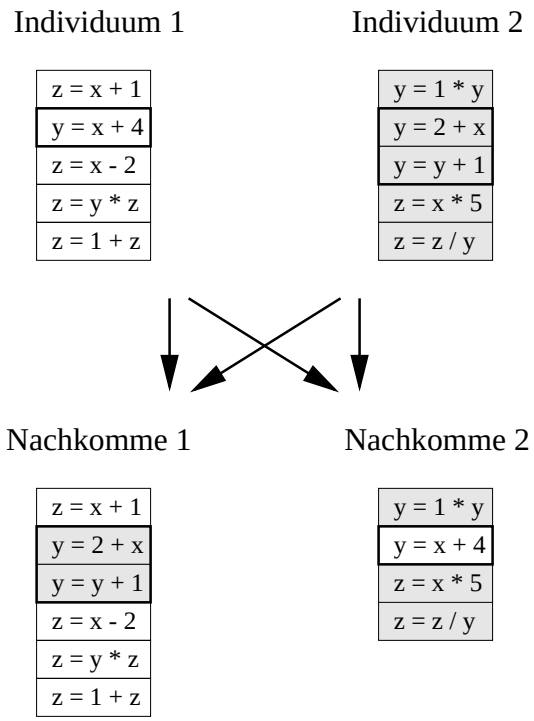


Abbildung 2.8: Crossover-Operation für genetisches Programmieren mit linearem Genotyp. In jedem der beiden selektierten Individuen wird ein Teilstück ausgewählt. Durch den Austausch der Teilstücke werden zwei Nachkommen erzeugt. Als Beispiel für die Crossover-Operation dient wieder die Evolution eines symbolischen Ausdrucks für eine Funktion. `x`, `y` und `z` seien Register, die dem evolvierten Programm zur Verfügung stehen. Der Parameter der Funktion wird im `x`-Register übergeben. Das Ergebnis der Berechnungen wird aus dem `z`-Register gelesen.

Wie beim baumbasierten genetischen Programmieren haben auch hier die Individuen eine variable Länge. Beim Crossover werden Teilstücke aus den Programmen herausgeschnitten und zwischen den Individuen ausgetauscht. Ein Beispiel für eine Crossover-Operation, die auf linearen Programmen operiert, ist in Abbildung 2.8 dargestellt. Bei der Mutation werden entweder die Argumente eines Befehls oder der Opcode des Befehls verändert. Bei der Evolution von Maschinencode wird meist auf den Divisionsbefehl verzichtet, da durch diesen Befehl bei der Division durch Null ein Fehler erzeugt werden würde, und dieser abgefangen werden müßte. Bei einer Interpretation der Programme können solche Ausnahmen vorher abgefangen werden. Werden die Befehle interpretiert, dann geht natürlich der Geschwindigkeitsvorteil, der bei der Evolution von Maschinencode existiert, verloren. Alternativ könnte man auch Macrobefehle einführen, die aus mehreren Maschinenbefehlen bestehen, und die genetischen Operatoren nur auf die Macrobefehle anwenden.

2.4 Evolution der Struktur

Eine weitere Variante von genetischem Programmieren ist die sogenannte *Structure Evolution*. *Structure Evolution* ist eine Variante der Evolutionsstrategie [247] und wurde von Lohmann [187, 184] entwickelt. Mit *Structure Evolution* können Strukturen und deren Parameter opti-

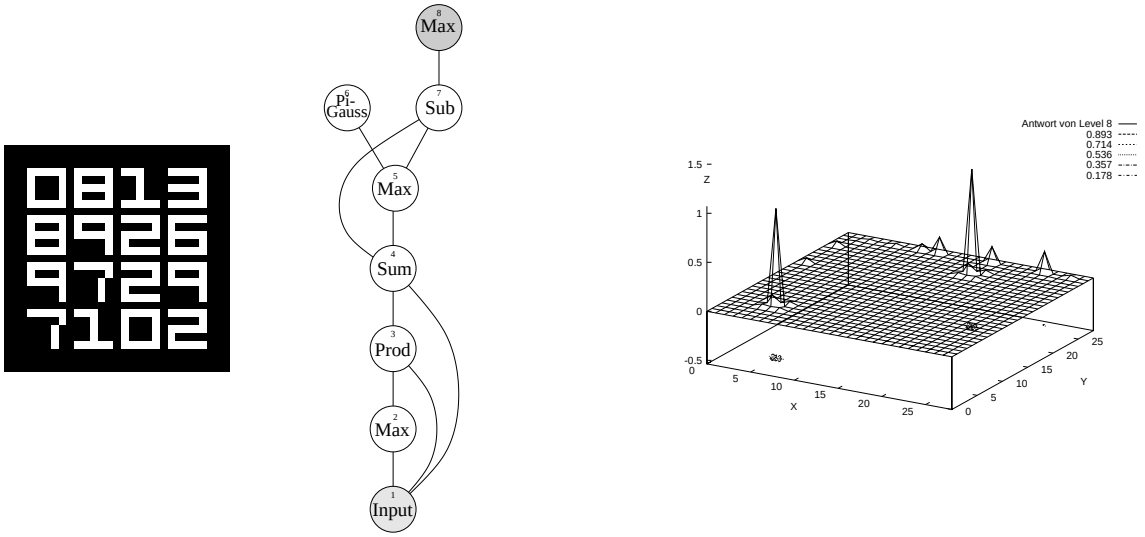


Abbildung 2.9: Evolvierter Detektor für die Ziffer 0. Als Eingabe wird eine Matrix aus 16 Ziffern verwendet. Der letzte Knoten des Individuums liefert die Ausgabe (Knoten 8). Die Ausgabe ist auf der rechten Seite dargestellt.

miert werden. Lohmann weist auf die Wichtigkeit der Strukturevolution hin. Wird ein System evolviert, das eine gewünschte Ausgabe erzeugen soll, dann wird in der Regel ein Fehlermaß berechnet, das von der gewünschten Ausgabe und der tatsächlichen Ausgabe abhängt. Die gewünschte Ausgabe des Systems ist lediglich eine Funktion der Eingabe. Die tatsächliche Ausgabe des Systems hängt aber von der Eingabe, der Struktur des Systems und deren Parametern ab. Daher sind alle drei Parameter, die Eingabe, die Struktur und die internen Parameter zu variieren, um das Optimum des Fehlermaßes zu finden. Bei *Structure Evolution* handelt es sich um eine geschachtelte Evolutionsstrategie. Dabei wird die Struktur in der äußeren und die Parameter in der inneren Schleife evolviert. Zur Selektion der Individuen der inneren und der äußeren Schleife können unterschiedliche Kriterien eingesetzt werden. Durch die unterschiedlichen Selektionskriterien soll die Fähigkeit zur Generalisierung erhöht werden. Bei der Mutation der Struktur ist die starke Kausalität zu beachten, d.h. kleine Veränderungen der Struktur sollen zu einer kleinen Veränderung des Fehlermaßes führen. Lohmann [185, 186, 187, 184] evolvierte mit *Structure Evolution* die Struktur und Parameter von Filtern, um aus einem Bild die Zahl der darin vorhandenen Objekte zu berechnen.

Ebner [75] setzte in ersten Experimenten *Structure Evolution* ein, um Detektoren für einfache, klar definierte Zeichen zu evolvieren. Die Detektoren waren dabei aus unterschiedlichen Filteroperationen zusammengesetzt. Evolviert wurde die Struktur der Individuen und die Parameter der Filterkerne. Als elementare Operationen wurden Addition, Subtraktion, Multiplikation, Addition und Multiplikation von Gaußfunktionen sowie Maximum- und Minimum-Funktionen verwendet.

Ein Individuum bestand aus einer Aneinanderreihung von Filteroperationen. Es wurde ein linearer Genotyp evolviert. Als Eingabe für die Operationen konnten die Bilder verwendet werden, die von einer vorangegangenen Operation erzeugt wurden. Es existierten unäre und binäre elementare Operationen. Das Resultat eines Laufes, bei dem ein Detektor für die Ziffer 0 evolviert wurde, ist in Abbildung 2.9 dargestellt. Als Eingabe diente eine Matrix mit

16 zufällig ausgewählten Zeichen. Als Zeichen wurden die Ziffern 0 bis 9 verwendet. Der letzte Knoten des Individuums liefert die Ausgabe (hier ist es Knoten 8). Die Ausgabe des Detektors ist auf der rechten Seite graphisch dargestellt. Wie in Abbildung 2.9 zu sehen ist, liefert das evolvierte Individuum an den Positionen, an denen sich die Ziffer 0 befindet, die gewünschte Antwort.

Weitere Experimente zur Evolution visueller Merkmalsdetektoren wurden mit baumbasiertem genetischem Programmieren durchgeführt. Baumbasiertes genetisches Programmieren wurde eingesetzt, da hierfür bereits sehr gute Experimentierumgebungen existieren. Da die Evolution von Merkmalsdetektoren in der Regel sehr rechenzeitintensiv ist, wurden in den folgenden Experimenten zur Evolution visueller Merkmalsdetektoren keine einzelnen Filtermatrizen mehr evolviert. Stattdessen wurden höhere Operationen, wie z.B. Gauß- und Gabor-Filter, in die Menge der elementaren Funktionen aufgenommen. Dadurch konnte die Evolution auf einer höheren Ebene ansetzen.

2.5 Die Größe des Suchraumes von genetischem Programmieren

Einer der wesentlichsten Unterschiede zwischen genetischen Algorithmen und genetischem Programmieren ist die Größe und die Beschaffenheit des Suchraumes. Der Suchraum von genetischem Programmieren wurde von Ebner [80] näher untersucht. Der Suchraum, der bei Experimenten mit genetischen Algorithmen durchsucht wird, ist verhältnismäßig klein. Ist l die Länge des Bitstrings (unter der Annahme, daß eine binäre Kodierung gewählt wird), dann enthält der Suchraum 2^l verschiedene Individuen.

Im folgenden wird zunächst die Größe des Suchraumes von baumbasiertem genetischem Programmieren abgeschätzt, und dann die Beschaffenheit des Suchraumes untersucht. Für die Abschätzung der Größe des Suchraumes werden lediglich Binärbäume betrachtet. Damit werden die möglichen elementaren Funktionen auf Funktionen mit nur zwei Argumenten beschränkt. Die Zahl der elementaren Funktionen sei F, die Zahl der Terminal-Symbole sei T. Es sei $B(d)$ die Zahl der Bäume, die mit der maximalen Tiefe d darstellbar sind. Ein Baum, der nur aus einem einzigen Knoten besteht, hat die Tiefe 0. Dann ergibt sich die Zahl der Bäume, die mit der maximalen Tiefe $d+1$ darstellbar sind, aus einer frei wählbaren Funktion für die Wurzel des Baumes und allen möglichen Teilbäumen der Tiefe d. Zu dieser Zahl muß dann noch der Baum, der lediglich aus einem Terminal-Symbol besteht, hinzugefügt werden. Daher ist die Zahl der Bäume mit der maximalen Tiefe d durch die folgende Rekursionsgleichung gegeben:

$$B(0) = T \tag{2.1}$$

$$B(d) = FB^2(d-1) + T \tag{2.2}$$

Die Zahl der Bäume mit der maximalen Tiefe d kann auch wie folgt beschrieben werden.

$$B(d) = \sum_{i=0}^{2^d-1} b_{d,i} F^i T^{i+1} \tag{2.3}$$

Dabei gibt $b_{d,i}$ die Zahl der Bäume an, die mit i elementaren Funktionen und maximaler Tiefe d dargestellt werden können. Die Zahl der unterschiedlichen Strukturen erhält man für

Max. Tiefe d	0	1	2	3	4	5	...	12
$F=1, T=1$	1	2	5	26	677	458330	...	$4.27{\cdot}10^{724}$
$F=2, T=2$	2	10	202	81610	$1.33{\cdot}10^{10}$	$3.55{\cdot}10^{20}$	...	$4.35{\cdot}10^{2668}$
$F=3, T=2$	2	14	590	$1.04{\cdot}10^{6}$	$3.27{\cdot}10^{12}$	$3.21{\cdot}10^{25}$	...	$2.80{\cdot}10^{3325}$
$F=4, T=2$	2	18	1298	$6.74{\cdot}10^{6}$	$1.82{\cdot}10^{14}$	$1.32{\cdot}10^{29}$	...	$7.96{\cdot}10^{3803}$

Tabelle 2.1: Zahl der unterschiedlichen Bäume bei maximaler Baumtiefe d.

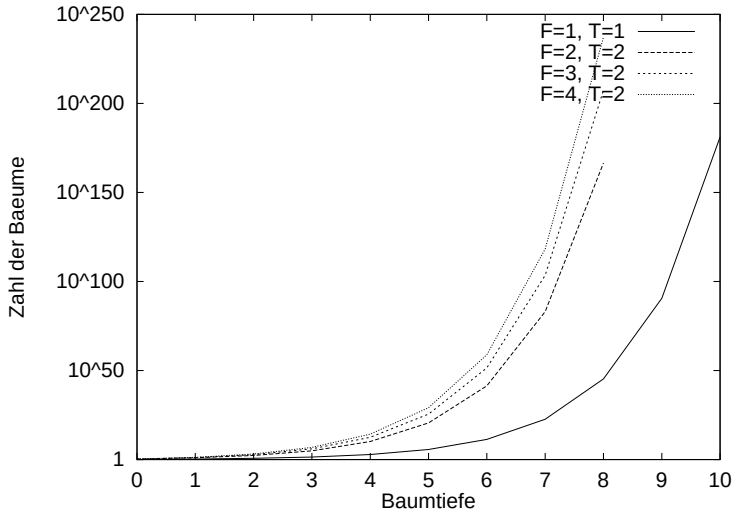


Abbildung 2.10: Zahl der unterschiedlichen Bäume bei maximaler Baumtiefe d. Die Grafik hat eine logarithmische Skala.

$F=1, T=1$. Ein Gefühl für das Wachstum der Funktion $B(d)$ erhält man, indem man den letzten Term, die Zahl der vollständigen Bäume der Tiefe d, betrachtet. Es gibt $F^{2^d-1}T^{2^d}$ unterschiedliche Funktionen, die die Struktur eines vollständigen Binärbaumes haben.

In Tabelle 2.1 sind die Zahl der unterschiedlichen Bäume, die sich für unterschiedliche Zahl der elementaren Funktionen und der Terminal-Symbole ergeben, angegeben. Als Beispiel soll hier wieder die Suche nach einem symbolischen Ausdruck dienen. Nimmt man an, daß Addition (`+`), Subtraktion (`-`), Multiplikation (`*`) und Division (`/`) als elementare Funktionen und die Variable `X` sowie eine Konstante als Terminal-Symbole eingesetzt werden, dann ist $F=4$ und $T=2$. Man sieht, daß der Suchraum schon bei geringen Tiefen und kleiner Zahl der elementaren Funktionen und Terminal-Symbolen sehr schnell sehr groß wird. Abbildung 2.10 zeigt die Zahl der Bäume in Abhängigkeit von der maximalen Tiefe.

Meist ist nicht nur die Tiefe der Bäume, sondern auch die maximale Zahl der Knoten beschränkt. Die Zahl der Bäume, die mit genau n inneren Knoten darstellbar sind, ist durch die folgende Rekursionsgleichung gegeben.

$$B(0) = T \tag{2.4}$$

$$B(n) = \sum_{i=0}^{n-1} FB(n-1-i)B(i) \tag{2.5}$$

Max. Anzahl innerer Knoten n	0	1	2	3	4	5	...	1000
$F = 1, T = 1$	1	2	4	9	23	65	...	$2.73 \cdot 10^{597}$
$F = 2, T = 2$	2	10	74	714	7882	93898	...	$5.01 \cdot 10^{1199}$
$F = 3, T = 2$	2	14	158	2318	38606	691790	...	$6.05 \cdot 10^{1375}$
$F = 4, T = 2$	2	18	274	5394	120082	$2.87 \cdot 10^{6}$	...	$5.20 \cdot 10^{1500}$

Tabelle 2.2: Zahl der unterschiedlichen Bäume mit maximal n inneren Knoten.

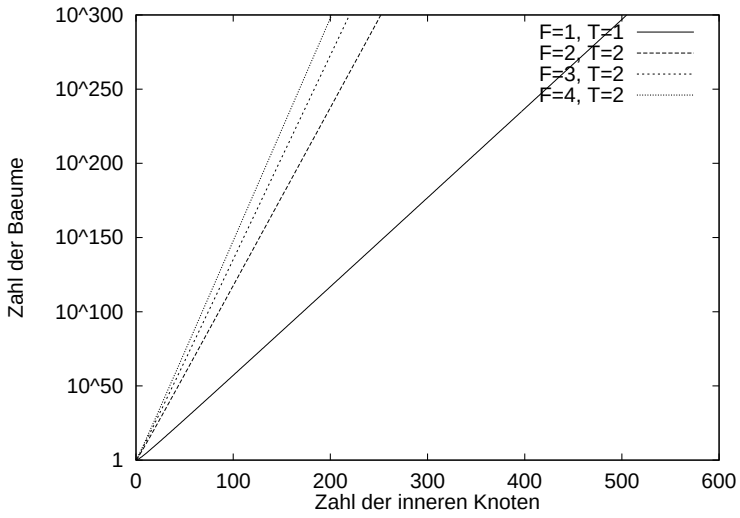


Abbildung 2.11: Zahl der unterschiedlichen Bäume mit maximal n inneren Knoten. Die Grafik hat eine logarithmische Skala.

Ein Baum mit n inneren Knoten kann aus allen möglichen Kombinationen zweier Teilbäume zusammengesetzt werden, deren Zahl der inneren Knoten zusammen $n-1$ beträgt. Die Wurzel des Baumes kann eine beliebige elementare Funktion sein. Es gilt $B(n) = b_{n,n} F^n T^{n+1}$. Die Zahl der unterschiedlichen Binärbäume mit n Knoten ist $\frac{1}{n+1}\binom{2n}{n}$ [57]. Daher gilt $b_{n,n} = \frac{1}{n+1}\binom{2n}{n}$. In Abbildung 2.11 sind die Zahl der Bäume mit maximal n inneren Knoten für unterschiedliche Zahl der elementaren Funktionen und Terminal-Symbole dargestellt. Tabelle 2.2 zeigt die Zahl der Bäume für maximal 1000 Knoten.

2.6 Der Einfluß von Introns auf die Beschaffenheit des Suchraumes

Im Vergleich zu den genetischen Algorithmen ist der Suchraum beim genetischen Programmieren völlig anders beschaffen. Bei genetischen Algorithmen wird meist mit einer Kodierung gearbeitet, bei der jedes Individuum einem Punkt im Suchraum entspricht. Es gibt dann nur eine einzige Lösung. Beim genetischen Programmieren gibt es dagegen meist mehrere Lösungen. Die unterschiedlichen Lösungen werden durch Programmcode verursacht, der nicht zum Verhalten des Individuums beiträgt. In Analogie zur natürlichen Evolution werden diese Programmstücke Introns genannt. Banzhaf et al. [16] geben die folgende Definition

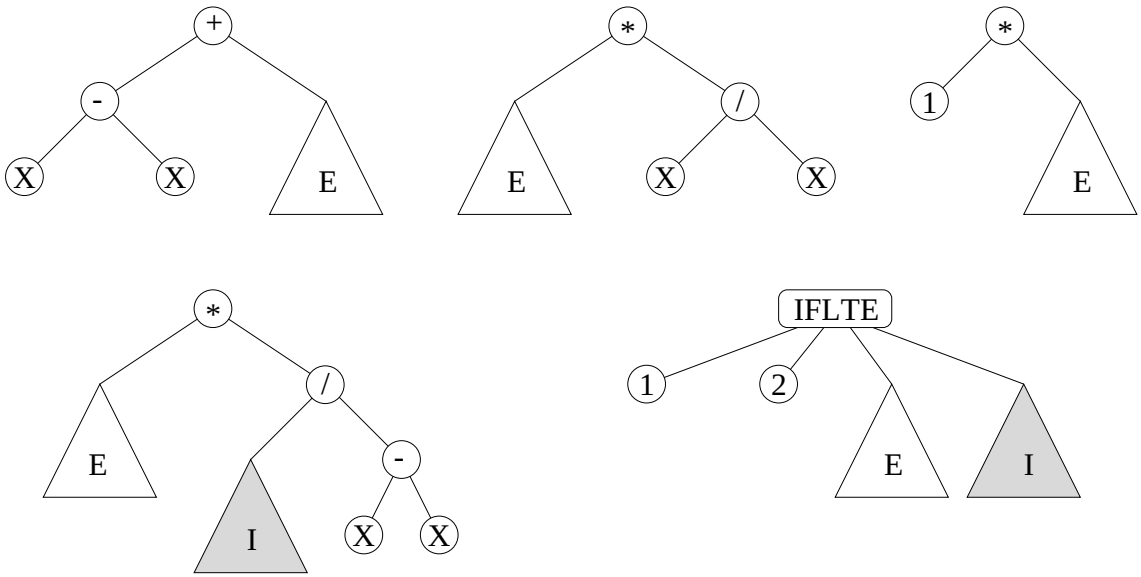


Abbildung 2.12: Fünf Beispiele für Introns. Nur die mit E bezeichneten Teilbäume tragen zum Verhalten des Individuums bei. Die dargestellten Bäume sind äquivalent zu den mit E bezeichneten Teilbäumen. Die mit I bezeichneten Teilbäume können beliebigen Programmcode enthalten. Im letzten Teilbaum wurde die elementare Funktion `IFLTE` (*IF Less Than or Equal to*) verwendet. Die Funktion wertet das dritte Argument aus, falls das erste Argument kleiner als oder gleich groß wie das zweite Argument ist, sonst wird das vierte Argument ausgewertet.

für Introns. Introns sind die Teile des Genotyps, die durch die Evolution von Individuen mit unterschiedlich großen Strukturen entstehen und die keinen Einfluß auf das Überleben der Individuen haben. In Abbildung 2.12 sind eine Reihe von Introns dargestellt.

Experimentelle Untersuchungen zeigten, daß durch die Anwendung genetischer Operationen, wie z.B. Mutation, Crossover und Selektion, der Anteil der Introns mit der Zeit wächst [16, 171, 172]. Zwar tragen die Introns nicht zum Verhalten eines Individuums bei, jedoch haben sie einen Einfluß auf die Lebensdauer der Nachkommen. Wird eine Crossover-Operation durchgeführt, dann wird ein sehr gutes Individuum aufgeteilt und wieder zusammengefügt. Man hofft, daß dadurch unterschiedliche Programmteile in geeigneter Weise miteinander kombiniert werden. In der Regel wird durch solch eine Operation ein Individuum zerrissen, und es entsteht unbrauchbarer Programmcode. Enthält ein Individuum jedoch ein Programmstück, das nicht ausgeführt wird, dann kann es passieren, daß die Crossover-Operation innerhalb dieses Programmstückes durchgeführt wird. Damit erhöht sich die Wahrscheinlichkeit, daß ein Nachkomme des Individuums die gleiche Fitneß wie das Eltern-Individuum hat, mit der Größe dieses Programmstückes. In Abbildung 2.12 sind zwei Bäume dargestellt, die Programmcode enthalten (graue Teilbäume), der durch beliebigen Code ausgetauscht werden kann, ohne daß dadurch die Fitneß des Individuums beeinflußt wird. Meist stagniert die Evolution, wenn der Anteil der Introns an den Individuen stark ansteigt [16]. Die Individuen "blähen" sich auf, da dies die einzige Möglichkeit ist, ihre effektive Fitneß (die den negativen Einfluß der genetischen Operatoren mitberücksichtigt) zu erhöhen. Daher könnte der Intron-Anteil der Individuen als Indikator herangezogen werden, daß ein lokales Optimum erreicht wurde.

Hierbei gibt es noch einen Unterschied zwischen baumbasiertem genetischem Program-

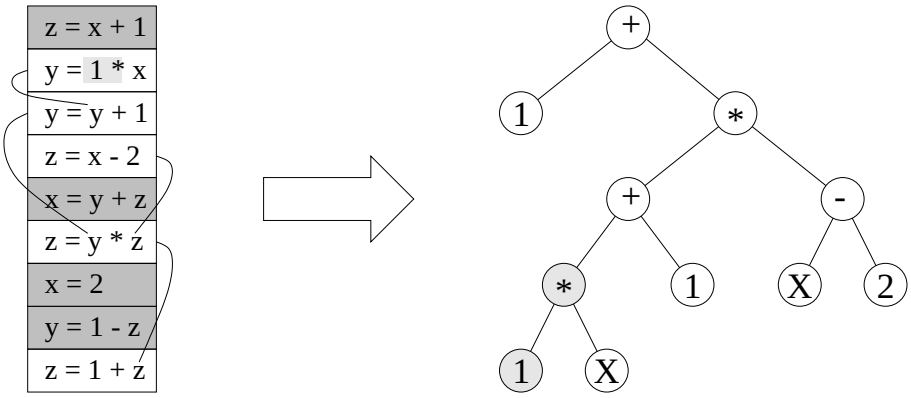


Abbildung 2.13: Beim genetischen Programmieren mit linearem Genotyp gibt es noch weitere Möglichkeiten Programmcode zu erzeugen, der nicht zum Verhalten des Individuums beiträgt. Als Beispiel dient hier die Suche nach einem symbolischen Ausdruck für eine Funktion. Rechts ist der Ausdruck als Baum dargestellt. Der Baum enthält ein Programmfragment, das nicht zum Verhalten des Individuums beiträgt (Hellgrau dargestellt). Auf der linken Seite ist Maschinencode zu sehen, der dieselbe Funktionalität besitzt. Neben dem hellgrauen Codefragment tragen die dunkelgrau unterlegten Befehle ebenfalls nicht zum Verhalten des Individuums bei.

Maximale Baumtiefe	0	1	2	3
$F = 2, T = 2$, Elementare Funktionen: `+-`	1	1	13	3173
$F = 3, T = 2$, Elementare Funktionen: `+-*`	1	3	44	27586
$F = 3, T = 2$, Elementare Funktionen: `+-/`	1	2	40	28648
$F = 4, T = 2$, Elementare Funktionen: `+-*/`	1	4	108	185709

Tabelle 2.3: Zahl der Funktionen mit $f(x) \approx x$.

mieren und genetischem Programmieren mit linearem Genotyp. Nimmt man an, daß nach einem symbolischen Ausdruck für eine Funktion gesucht wird, dann kann der lineare Genotyp neben dem Programmcode, der Teil des Baumes ist, noch zusätzlichen Code enthalten. Im Compilerbau wird dieser Code passiver Code oder, falls durch Sprungbefehle der Code umgangen wird, als unerreichbarer Code bezeichnet [2]. Ein Beispiel für ein Individuum mit passivem Code ist das in Abbildung 2.13 dargestellte Individuum. Auf der rechten Seite ist das Individuum als Baum dargestellt. In dem symbolischen Ausdruck ist ein Programmfragment enthalten, das nicht zum Verhalten des Individuums beiträgt (Hellgrau dargestellt). Links ist das gleiche Individuum als linearer Genotyp dargestellt. Es enthält neben dem hellgrau unterlegten Teil des Programmes noch weitere Codefragmente, die nicht zum Verhalten des Individuums beitragen. Die dunkelgrau unterlegten Befehle tragen ebenfalls nicht zum Verhalten des Individuums bei. Dort werden lediglich Variablen verändert, die nicht mehr benötigt werden oder die vor deren eigentlichen Verwendung erneut berechnet werden.

Aufgrund der Introns entspricht beim genetischen Programmieren der gesuchten Lösung meist nicht nur ein einziges Individuum sondern eine Vielzahl. Daher könnte je nach Art der elementaren Funktionen und verwendeten Terminal-Symbole eine Lösung verhältnismäßig oft im Suchraum vorkommen. Dies wurde am Beispiel der Suche nach einem symbolischen Ausdruck für eine Funktion untersucht. Als elementare Funktionen wurden Addition (`+`),

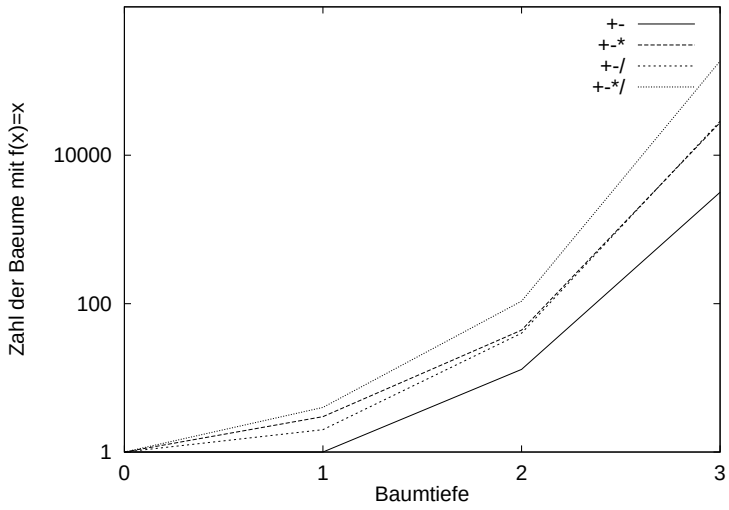


Abbildung 2.14: Zahl der Funktionen mit $f(x) \approx x$ in Abhängigkeit von der maximalen Tiefe d. Die Grafik hat eine logarithmische Skala.

Subtraktion (`-`), Multiplikation (`*`) und Division (`/`) verwendet. Die Division liefert in diesem Fall 1 zurück, falls der Betrag des Divisors kleiner als 10^{-10} ist. Als Terminal-Symbole wurden die Variable `X` und die Konstante `1` eingesetzt. Untersucht wurde, wieviele Bäume die Funktion $f(x) \approx x$ darstellen. Es gelte $f(x) \approx x$, falls $|f(x) - x| < 10^{-10}$ für alle $x \in \{\pi, e, 1, 2, 3, 7, 12\}$. Da die Zahl der Bäume mit der Tiefe sehr schnell wächst, konnte der Test nur für kleine Bäume durchgeführt werden. In Tabelle 2.3 sind die Zahl der Bäume, die die Identität darstellen, für unterschiedliche Tiefen und unterschiedliche Zahl der elementaren Funktionen aufgeführt. Dies ist in Abbildung 2.14 noch einmal graphisch dargestellt.

Die hier durchgeführte Untersuchung des Suchraumes soll dem Leser verdeutlichen, wie schwierig es ist, eine Lösung in diesem Suchraum zu finden. Die Untersuchung zeigte jedoch auch, daß die Wahrscheinlichkeit, eine Lösung in diesem Suchraum zu finden, aufgrund von Introns größer ist, als es auf den ersten Blick scheint. Sowohl die Zahl der unterschiedlichen Bäume als auch die Zahl der Funktionen, die die Identität darstellen, wachsen sehr stark mit wachsender Tiefe (siehe Abbildung 2.10 und Abbildung 2.14). Abschließend wird noch eine Analogie zwischen genetischem Programmieren und der natürlichen Evolution gezogen. Denn auch für den natürlichen Suchraum ist es nicht so unwahrscheinlich, eine Lösung zu finden, wie es zunächst scheint.

2.7 Analogie zwischen genetischem Programmieren und der natürlichen Evolution

Auch hier läßt sich wieder eine Analogie zur natürlichen Evolution ziehen. Die Erbinformation eines Individuums ist in der DNA gespeichert [195]. Die DNA besteht aus zwei Strängen, die die Form einer Doppelhelix besitzen. Jeder der Stränge besteht aus einer Zucker-Phosphat-Zucker-Phosphat Kette, an die vier unterschiedliche Basen angefügt sind. Als Basen stehen Adenin, Thymin, Guanin und Cytosin zur Verfügung. Ein Strang kann aus einer beliebigen

Sequenz dieser Basen bestehen. Zu jeder Sequenz eines Stranges gehört jedoch eine ganz bestimmte komplementäre Sequenz des anderen Stranges. Es können nur die Basenpaare Adenin-Thymin und Guanin-Cytosin gebildet werden. Je drei der Basen kodieren eine von 20 möglichen Aminosäuren. In einem zweistufigen Verfahren, genannt Transkription und Translation wird aus einem Teilstück der DNA ein Protein erzeugt. Ein Gen ist ein Teilstück der DNA. Der Anfang und das Ende der Sequenzen werden durch spezielle Start- bzw. Stop-Triplets markiert. Bei der Transkription wird ein sogenanntes mRNA Molekül erzeugt, das komplementär zu einem der DNA Stränge ist. RNA enthält anstatt der Base Thymin die Base Uracil. Aus diesem mRNA Molekül wird schließlich mit Hilfe von Transfer RNA das entsprechende Protein synthetisiert. Dieser zweite Schritt, bei dem die entsprechenden Aminosäuren aufgereiht werden, wird Translation genannt. Ein Enzym ist ein Proteinmolekül, das eine bestimmte Reaktion katalysiert. Die chemischen Reaktionen, die den Stoffwechsel eines Individuums ausmachen, sind auf diese Enzyme angewiesen. Damit ein bestimmtes Enzym erzeugt wird, muß ein entsprechendes Gen im Organismus vorhanden sein.

Durch die DNA-, RNA- und Protein-Sequenzen werden Sequenzräume aufgespannt. Kauffman [154] und Schuster [270] haben den Raum der DNA-, RNA-, bzw. Protein-Sequenzen näher untersucht. Kauffman stellte fest, daß eine begrenzte Zahl von Enzymen den Raum der Katalysationsaufgaben vollständig abdecken könnte. Daher muß die Kodierung der Funktionen in hohem Maße redundant sein. Sequenzen, die jede der häufigeren Formen kodieren, finden sich in geringem Abstand zu jeder zufällig ausgewählten Sequenz [270]. Es gibt wenig Formen, die häufig im Suchraum vorkommen und viele, die selten vorkommen. Außerdem sind unterschiedliche Sequenzen, die eine bestimmte Form kodieren, zufällig im Sequenzraum verteilt. Von einer Sequenz führt oft ein langer Pfad über neutrale Mutationen zu anderen Sequenzen, die dieselbe Form kodieren. Die Zahl der unterschiedlichen Formen und chemischen Reaktionen ist deutlich kleiner als die Zahl der möglichen Sequenzen. Daraus folgt wiederum, daß die Evolution von Leben nicht so unwahrscheinlich ist, wie es der unvorstellbar große Raum der Sequenzen vermuten läßt [154].

Wie in der Natur ist auch beim genetischen Programmieren die Kodierung redundant. Die obige Analyse zeigte (Tabelle 2.3 und Abbildung 2.14), daß es meist eine Vielzahl von Individuen gibt, die die gesuchte Lösung des Problems darstellen. Daher kann es vorkommen, daß das Erreichen des Optimums nicht so unwahrscheinlich ist, wie es die Größe des Suchraumes vermuten läßt. Zusätzlich kann es von Vorteil sein, die Repräsentation geeignet zu wählen. Dies wird durch Untersuchungen deutlich, bei denen der Einfluß der Repräsentation auf das Verhalten bzw. den Ausgang des Laufes analysiert wurde [250, 110]. In Analogie zum natürlichen Suchraum könnte es von Vorteil sein, eine Repräsentation zur verwenden, bei der Individuen, die häufig vorkommende Verhalten kodieren, innerhalb eines kleinen Radius von jedem zufällig ausgewählten Individuum liegen. Außerdem sollten unterschiedliche Individuen, die ein und dasselbe Verhalten kodieren, zufällig im Suchraum verteilt sein. Für ein gegebenes Individuum sollten möglichst weit entfernte Individuen über neutrale Mutationen erreichbar sein. Banzhaf [15] stellte einen Ansatz vor, der in diese Richtung geht. Er entwickelte eine Variante von genetischem Programmieren, die sich sehr stark an die natürliche Evolution anlehnt. Er verwendete eine Repräsentation, die einen hohen Grad an Redundanz und neutrale Mutationen zuläßt. Banzhaf evolvierte Bitstrings, aus denen, in Analogie zur natürlichen Evolution, in einem Prozeß aus Transkription und Translation der Phänotyp erzeugt wird. Ungültige Zeichenketten werden korrigiert, so daß nur semantisch korrekte Zeichenketten entstehen. In einer von Keller und Banzhaf [155] durchgeführten Untersuchung zeigte sich, daß der Einsatz dieser Variante vorteilhaft sein kann.

Nachdem der Suchraum von genetischem Programmieren analysiert wurde, werden nun einige Experimentierumgebungen für genetisches Programmieren beschrieben bevor im nächsten Kapitel die ersten Experimente beschrieben werden, die mit genetischem Programmieren durchgeführt wurden.

2.8 Experimentierumgebungen für genetisches Programmieren

Es existieren eine Reihe von Experimentierumgebungen für genetisches Programmieren. Deakin und Yates [66] geben eine Übersicht. Koza veröffentlichte in seinem Buch [165] den Quellcode seiner Experimentierumgebung zusammen mit drei Anwendungsbeispielen. Da Koza LISP-Programme evolviert, ist sein System in LISP geschrieben. Die meisten seiner Experimente wurden auf Rechnern durchgeführt, die mit speziellen LISP-Prozessoren bzw. Erweiterungskarten ausgestattet waren. Prinzipiell kann genetisches Programmieren jedoch in einer beliebigen Programmiersprache implementiert werden. Die neueren Experimentierumgebungen sind in der Regel in der Programmiersprache C [156, 183] implementiert. Da die meisten Experimente sehr viel Rechenzeit in Anspruch nehmen, bieten diese Systeme gegenüber einer Implementierung in LISP einen deutlichen Geschwindigkeitsvorteil.

Erste Experimente wurden vom Autor mit dem Geppetto Genetic Programming System Version 2.1, das von Dave Glowacki [108] entwickelt wurde, durchgeführt. Geppetto ist in C geschrieben und kann leicht für eigene Experimente angepaßt werden. Einige Beispielanwendungen für genetisches Programmieren, wie z.B. die Evolution eines symbolischen Ausdrucks für eine Funktion, die Evolution eines Multiplexers oder die Evolution eines Programmes zur Verfolgung eines diskretisierten Pfades, sind in dem Programmpaket enthalten.

Die hier beschriebenen Experimente wurden alle mit dem lil-gp Programming System Version 1.1, das von Zongker und Punch [333] entwickelt wurde, durchgeführt. lil-gp ist ebenfalls in C geschrieben. Im Gegensatz zu Geppetto handelt es sich bei lil-gp jedoch um ein wesentlich komfortableres System. Das System unterstützt POSIX und Solaris Threads [158], mehrere Populationen und die sogenannten automatisch definierten Funktionen [166, 167]. Es besteht die Möglichkeit, die Experimente nach jeder Generation zwischenzuspeichern und zu einem späteren Zeitpunkt fortzusetzen. Dies ist vor allem bei sehr langen Experimenten einer der größten Vorteile des Programmes. Zudem werden während des Laufes umfangreiche Statistiken berechnet, die bei einer späteren Analyse hilfreich sind.

2.9 Zusammenfassung

Es wurde eine kurze Einführung in genetische Algorithmen gegeben und das Repräsentationsproblem diskutiert. Anschließend wurde die Funktionsweise von genetischem Programmieren beschrieben. Dabei wurde auf drei Varianten, baumbasiertes genetisches Programmieren, genetisches Programmieren mit linearem Genotyp und die Evolution von Strukturen näher eingegangen. Genetisches Programmieren unterscheidet sich deutlich von genetischen Algorithmen. Bei den genetischen Algorithmen besitzen die Individuen meist eine fest vorgegebene Größe und eine klar definierte Struktur. Im Gegensatz dazu lassen sich mit genetischem Programmieren sowohl die Struktur der Individuen als auch deren Größe evolvieren. Bei genetischen Algorithmen entspricht in der Regel genau ein Individuum der gesuchten Lösung. Bei genetischem Programmieren existieren meist mehrere äquivalente Lösungen. Aufgrund von

Introns existieren für jeden Phänotyp oft mehrere Genotypen die diesen kodieren. Durch eine Analyse des Suchraumes von genetischem Programmieren wurde gezeigt, daß das Erreichen des Optimums nicht ganz so unwahrscheinlich ist, wie es die Größe des Suchraumes vermuten läßt. Durch eine Analogie zum natürlichen Suchraum wurden wesentliche Eigenschaften aufgezeigt, die auch bei genetischem Programmieren hilfreich sein könnten. Schließlich wurden verschiedene Experimentierumgebungen für genetisches Programmieren diskutiert.

Kapitel 3

Evolution abstandsbasierter Merkmalsdetektoren

Im folgenden wird genetisches Programmieren eingesetzt, um sonar- bzw. abstandsbasierte Merkmalsdetektoren für die Lokalisation eines mobilen Roboters zu evolvieren (siehe Ebner [81]). Ein mobiler Roboter ist in der Regel mit einer Reihe von Sensoren ausgestattet, über die der Roboter die Umgebung wahrnehmen kann. Mit Hilfe dieser Sensoren kann ein Modell der Umgebung erstellt werden. Solch ein Modell kann sehr hilfreich bei der Navigation sein. Mitunter kommt es vor, daß ein Roboter seine Position innerhalb der Umgebung verloren hat. Dies kann z.B. auftreten, wenn sich Odometriefehler aufsummieren oder der Roboter durch äußere Einwirkung bewegt wurde. Dann wird es erforderlich, daß der Roboter seine Position neu bestimmt. In diesem Zusammenhang wird der Einsatz von genetischem Programmieren zur Lokalisation eines mobilen Roboters untersucht. Mit genetischem Programmieren wird eine Funktion evolviert, die Sensordaten auf eine Position des Roboters (x- und y-Koordinate) abbildet. Diese Funktion stellt ein Modell der Umgebung des Roboters dar.

3.1 Lokalisation eines mobilen Roboters

In den hier beschriebenen Experimenten wird angenommen, daß der Roboter seine Umgebung mit einem Ring von Sonar-Sensoren oder einem Laser-Scanner wahrnimmt. Die Umgebung kann mit einem zweidimensionalen Plan beschrieben werden. Solch ein Plan wird oft von Simulatoren verwendet, um Abstandswerte für die Sensoren zu berechnen. Daher existiert

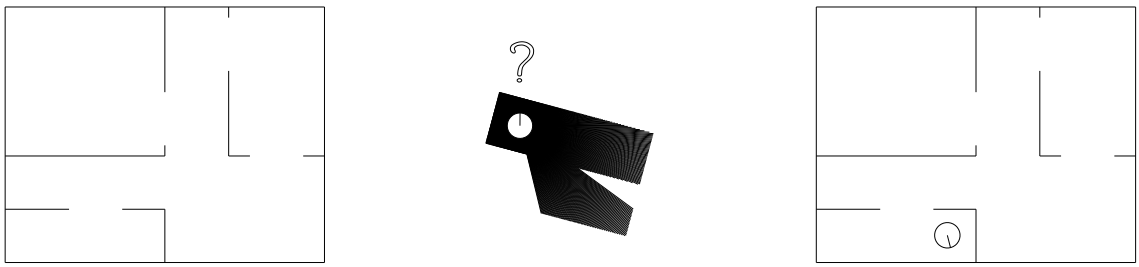


Abbildung 3.1: Wo in der Karte befindet sich der mobile Roboter? Die aktuellen Abstandswerte der Sensoren sind in der Mitte zu sehen. Die Antwort ist auf der rechten Seite dargestellt.

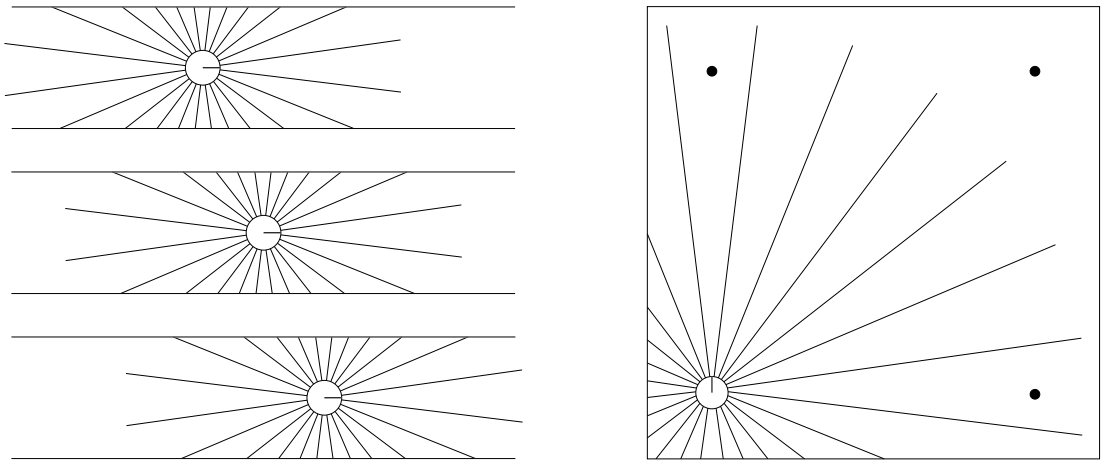


Abbildung 3.2: Zwei Beispiele für mehrdeutige Sensorinformationen. Bewegt sich der Roboter entlang eines Korridors, dann erhält er für alle Positionen entlang einer Linie, die parallel zum Korridor verläuft, die gleichen Sensorwerte [39] (links). Befindet sich ein Roboter in einer Ecke in einem großen Raum, dann ist es ebenfalls nicht möglich, ihn genau zu lokalisieren (rechts). Neben der tatsächlichen Position sind noch drei weitere Positionen möglich, die die gleichen Sensorwerte liefern.

eine Abbildung M, die die Position des Roboters auf die zugehörigen Sensorwerte S abbildet.

$$M : P \xrightarrow{\text{Welt}} S \tag{3.1}$$

Um die Position eines Roboters zu bestimmen, versucht man, diese Abbildung umzukehren. Es wird nach einer Abbildung M^{-1} gesucht, die für gegebene Sensorwerte S die zugehörige Position P bestimmt (Abbildung 3.1).

$$M^{-1} : S \xrightarrow{\text{Modell}} P \tag{3.2}$$

Natürlich ist es oft nicht möglich, eine Umkehrabbildung zu finden. Dies wird an den in Abbildung 3.2 gezeigten Beispielen deutlich. Ohne zusätzliche Informationen über die Umgebung kann das Problem nicht gelöst werden. Diese Information wird in der Regel durch Türen, Fenster oder Objekte, die im Raum verteilt sind, wie z.B. ein Tisch und einige Stühle, geliefert. Aufgrund der zusätzlichen Objekte kann eine Mehrdeutigkeit vermieden werden.

Traditionelle Methoden zur Lokalisation eines Roboters, die eine Korrespondenz zwischen den Sensorwerten und einer Karte herstellen, arbeiten oft nur bis zu einer begrenzten Genauigkeit. Hier wird versucht, eine inverse Abbildung für die Lokalisation eines Roboters zu evolvieren. Falls es möglich ist, solch eine Abbildung zu finden, kann der Roboter für beliebige Sensorwerte genau lokalisiert werden. Die Lokalisation ist in diesem Fall nicht quantisiert.

Bei einem Einsatz mit dem realen Roboter könnte die Assoziation zwischen Sensorwerten und den zugehörigen Positionen zunächst mit Hilfe der Odometrie des Roboters hergestellt werden. So wird ein internes Modell der Umgebung aufgebaut. Summieren sich mit der Zeit die Odometriefehler oder fällt die Odometrie ganz aus, so könnte das Modell eingesetzt werden, um die Fehler in der Odometrie zu korrigieren.

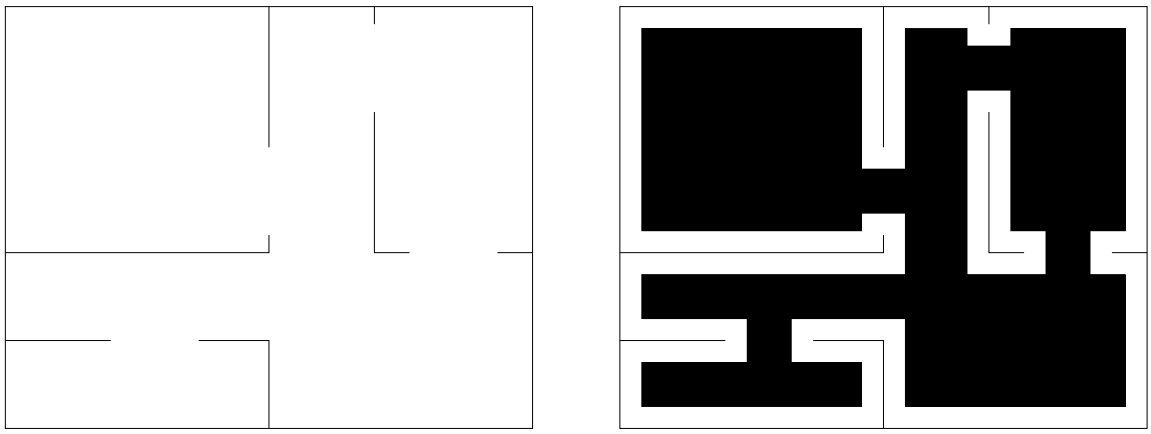


Abbildung 3.3: Umgebung, die für die Experimente eingesetzt wurde (links). Die schwarzen Bereiche sind mögliche Positionen des Roboters (rechts).

3.2 Evolution einer Abbildung zur Lokalisation eines mobilen Roboters

Die inverse Abbildung wird mit Hilfe von genetischem Programmieren [165, 167, 16] evolviert. Koza [165] hat bereits gezeigt, daß sich genetisches Programmieren sehr gut eignet, um einen symbolischen Ausdruck für eine Funktion zu evolvieren. Die Funktion wird dabei lediglich durch eine Reihe von Koordinaten und zugehörige Funktionswerte beschrieben. Um genetisches Programmieren für die Lokalisation eines mobilen Roboters zu verwenden, muß zunächst eine für das Problem geeignete Repräsentation definiert werden.

Das Problem, eine Funktion zu finden, die die Sensorwerte auf die zugehörigen Positionen des Roboters abbildet, ist aufgrund der großen Zahl der Sensorwerte äußerst schwierig. Daher werden die rohen Sensordaten zunächst vorverarbeitet. Diese so transformierten Daten werden dann als Eingabewerte für die evolvierte Funktion verwendet. Die Vorverarbeitung sollte eine Reihe von Kriterien erfüllen. Ihre Hauptaufgabe ist die Reduzierung der Eingabevariablen und die damit verbundene Verkleinerung des Suchraumes. Relevante Merkmale der Umgebung sollten hervorgehoben werden. Gleichzeitig sollte aber auch keine wertvolle Information verloren gehen. Ferner sollten die resultierenden Eingabevariablen sich mit einer kleinen Änderung der Position des Roboters auch nur geringfügig ändern, d.h. die Funktion sollte möglichst glatt sein. Da nur die Position des Roboters (x- und y-Koordinaten) aus den Sensorwerten bestimmt werden soll, sollten die transformierten Variablen unabhängig von der Orientierung des Roboters sein.

3.2.1 Terminal-Symbole

In den hier beschriebenen Experimenten wurden die Momente der Abstandsverteilung berechnet [31]. Die folgenden Terminal-Symbole wurden daher verwendet. Dabei gibt Abstand(i) den Wert an, der durch den i-ten Sensor gemessen wird und n ist die Zahl der gemessenen Abstandswerte. Die Differenz benachbarter Werte sei durch ΔAbstand(i) gegeben.

- Momente $\mathtt{M}_j = \frac{1}{n}\sum_{i=1}^{n}(\text{Abstand}(i))^j$ (j=$\{$1,2,3,4$\}$),

- Zentrale Momente $\texttt{CM}_j = \frac{1}{n}\sum_{i=1}^{n}(\text{Abstand}(i) - \texttt{M}_1)^j$ (j=$\{2,3,4,5\}$),
- Momente der Differenzen $\texttt{DM}_j = \frac{1}{n}\sum_{i=1}^{n}(\Delta\text{Abstand}(i))^j$ (j = $\{1,2,3,4\}$)
- sowie die Konstanten `RAND` (mit Bereich [0,1)), `RAND10` (mit Bereich [0,10)) und `RAND100` (mit Bereich [0,100)).

3.2.2 Elementare Funktionen

Die folgenden elementaren Funktionen wurden verwendet.

- Arithmetische Funktionen: `+`, `-`, `*`, `/`. Die Funktionen haben jeweils zwei Argumente und liefern das Ergebnis zurück. Die Division `/` liefert den Wert Eins zurück, falls der Betrag des Divisors kleiner als 10^{-10} ist.
- Trigonometrische Funktionen: `COS`, `SIN`, `TAN`, `ASIN`, `ACOS`, `ATAN`, `ATAN2`. Die Funktionen `COS`, `SIN`, `TAN` haben ein Argument und liefern das Ergebnis zurück. `ASIN`, `ACOS` und `ATAN` sind wie folgt definiert

  $$\texttt{ASIN}(x) = \begin{cases} \frac{\pi}{2} & x > 1 \\ -\frac{\pi}{2} & x < -1, \\ \text{asin}(x) & \text{sonst} \end{cases} \qquad \texttt{ACOS}(x) = \begin{cases} 0 & x > 1 \\ \pi & x < -1, \\ \text{acos}(x) & \text{sonst} \end{cases}$$

  $$\texttt{TAN}(x) = \begin{cases} 0 & |\cos(x)| < 10^{-10} \\ \frac{\sin(x)}{\cos(x)} & \text{sonst} \end{cases}.$$

  `ATAN2` verwendet die `atan2` Unix-Funktion zur Berechnung des Arcustangens aus den beiden Argumenten x und y.
- Der bedingte Befehl `IFLTE`. `IFLTE` ist eine Funktion mit vier Argumenten. Ist das erste Argument kleiner als oder gleich groß wie das zweite Argument, wird das dritte Argument zurückgeliefert, sonst wird das vierte Argument zurückgeliefert.

Sind die Sensoren dicht genug plaziert, dann sind die Momente der Abstandsverteilung annähernd unabhängig von der Orientierung des Roboters. In den hier beschriebenen Experimenten wurden 720 Sensoren eingesetzt. Diese Zahl entspricht der Anzahl der Werte, die ein kommerzieller Laser-Scanner zurückliefert.

3.3 Experimente

Für die Experimente wurde die Umgebung verwendet, die in Abbildung 3.3 zu sehen ist. Der dargestellte Grundriß ist ein Teil eines hypothetischen Bürogebäudes und hat die Größe $7,5\text{m} \times 6\text{m}$. Aufgabe ist es nun, eine Funktion zu evolvieren, die diese Umgebung beschreibt. Dazu wurden 1000 Fitneßtests (Koordinaten des Roboters und zugehörige Sensorwerte) eingesetzt. Für jeden Fitneßtest wurde eine Position des Roboters zufällig aus dem schwarz markierten Bereich der Karte ausgewählt (Abbildung 3.3). Es wurde ein Individuum für die x-Koordinate und ein Individuum für die y-Koordinate evolviert. Das Fehlermaß des Individuums, das die x-Koordinate berechnet, wird wie folgt berechnet.

$$\text{Fehler} = \frac{1}{n}\sum_{i=1}^{n}\left(x_{i,\text{tatsächlich}} - x_{i,\text{geschätzt}}\right)^2 \tag{3.3}$$

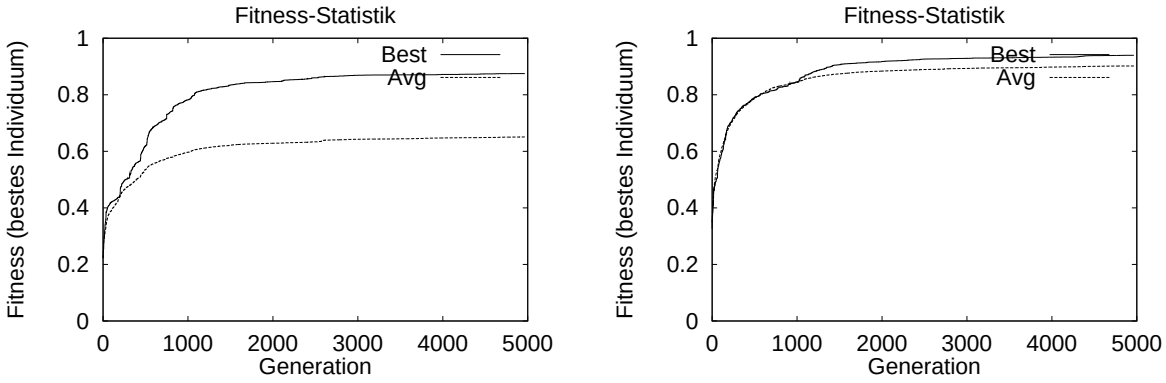


Abbildung 3.4: Fitneß-Statistiken der durchgeführten Experimente. Links sind die Fitneß-Statistiken für die x-Koordinate zu sehen, rechts die für die y-Koordinate. Die Fitneß wurde jeweils über 5 Experimente gemittelt. Die Fitneß des besten Individuums ist ebenfalls zu sehen.

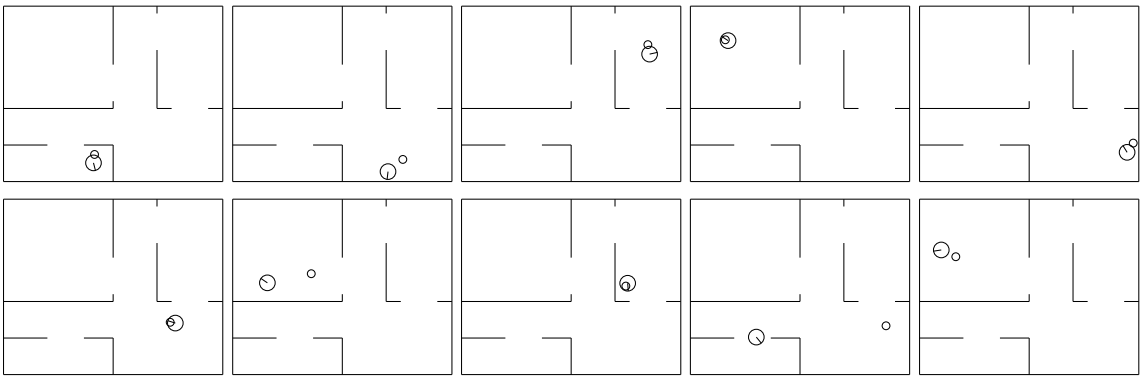


Abbildung 3.5: Die evolvierten Individuen wurden mit 10 zusätzlichen Fitneßtests getestet. Die durch die Individuen berechnete Position ist mit einem kleinen Kreis markiert.

dabei ist (für den i-ten Test) $x_{i,\text{tatsächlich}}$ die x-Koordinate, an der sich der Roboter befindet und $x_{i,\text{geschätzt}}$ ist die x-Koordinate, die vom evolvierten Individuum berechnet wurde. Die Zahl der Fitneßtests ist n. Die berechnete Position wird auf einen Bereich begrenzt, der doppelt so groß ist wie der Grundriß. Fällt die berechnete Position des Roboters außerhalb dieses Bereiches, so wird die Position auf den Rand des Bereiches gesetzt. Das Fehlermaß der y-Koordinate wird analog zur x-Koordinate berechnet. Die Fitneß des Individuums wird nach Fitneß $= \frac{1}{1+\text{Fehler}}$ berechnet.

Für jede Koordinate wurden fünf Experimente mit unterschiedlicher Initialisierung der Anfangspopulation durchgeführt. Es wurde eine Populationsgröße von 1000 Individuen verwendet. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 6 gefüllt. Die maximale Zahl der Knoten eines Individuums wurde auf 1000 und die Tiefe des Individuums auf 17 begrenzt. Die Wahrscheinlichkeiten für Crossover, Reproduktion und Mutation wurden auf 85%, 10% bzw. 5% gesetzt. Die Individuen wurden mit Tournament-Selektion ausgewählt. Die Läufe wurden jeweils nach 5000 Generationen abgebrochen. In Abbildung 3.4 sind die Fitneß-Statistiken für die beiden Koordinaten dargestellt. Die Fitneß wurde jeweils über die 5 Experimente gemittelt. Die Fitneß des besten Individuums ist ebenfalls zu sehen. Das Problem, eine inverse Funktion für die y-Koordinate zu

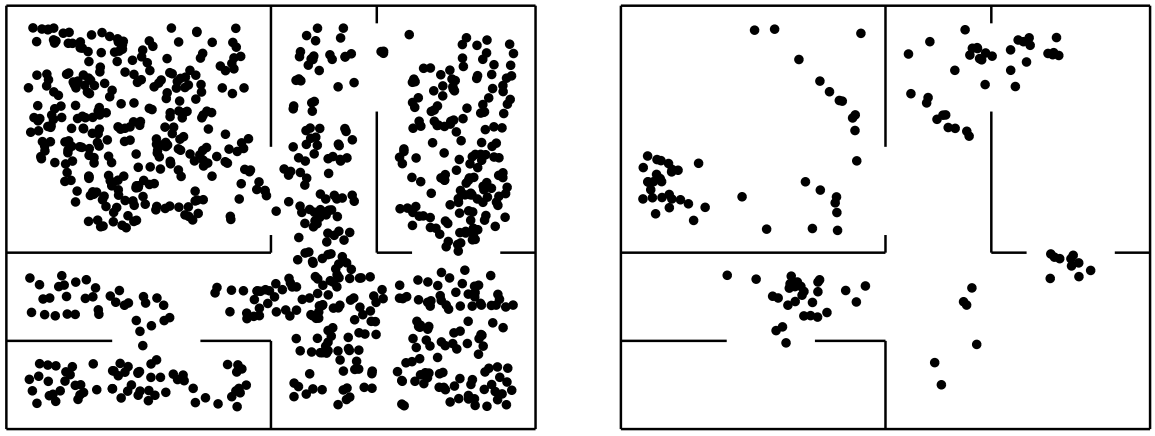


Abbildung 3.6: Mit 1000 weiteren Fitneßtests wurde die Genauigkeit der evolvierten Individuen getestet. Positionen, für die die berechnete Position der Individuen sich weniger als 1m von der tatsächlichen Position unterscheidet, sind links zu sehen, die anderen sind in der rechten Karte markiert.

evolvieren, scheint leichter zu sein, als die Funktion für die x-Koordinate zu evolvieren. Die beiden besten Individuen wurden schließlich zusammengefaßt und mit 10 neuen Fitneßtests ausgewertet. Die Ergebnisse dieses Tests sind in Abbildung 3.5 zu sehen. Die Position, die von den evolvierten Individuen berechnet wurde, ist jeweils durch einen kleinen Kreis markiert. Die geschätzte Position des Roboters ist für viele Fiteneßtests sehr genau, aber sie weicht bei einigen wenigen Fitneßtests auch sehr stark von der tatächlichen Position des Roboters ab. Mit weiteren 1000 Fitneßtests wurde die Genauigkeit der evolvierten Individuen getestet. Das Ergebnis dieses Tests ist in Abbildung 3.6 zu sehen. Positionen, für die die berechnete Position des Roboters weniger als 1m von der tatsächlichen Position abweicht, sind links zu sehen, die anderen sind im Bild rechts markiert.

3.4 Verwandte Arbeiten

Talluri und Aggarwal [291] geben eine ausführliche Übersicht über traditionelle Methoden zur Selbstlokalisation eines mobilen Roboters. Sie teilen die Methoden in vier Kategorien ein: (1) Methoden, die auf Landmarken basieren, (2) Methoden, die auf Pfadintegration bzw. der Odomerie basieren, (3) Methoden, die ein Referenzmuster zur Lokalisation verwenden und (4) Methoden, die ein vorgegebenes Modell der Umgebung verwenden, das mit den Sensorwerten des Roboters verglichen wird. In der Regel wird solch ein Modell, eine Karte der Umgebung, mit Hilfe von Ultraschall-Sensoren, einem Laser-Scanner oder einer Reihe von Bildern der Umgebung erstellt. Thrun et al. [297] geben eine Übersicht über verschiedene Verfahren zum Aufbau einer Karte der Umgebung. Tsuji und Li [304] repräsentieren die visuellen Informationen, die entlang eines Pfades wahrgenommen werden, als Panorama. Je nach Art der Sensoren ist es auch möglich, eine dreidimensionale Karte der Umgebung aufzubauen [316].

Talluri und Aggarwal [289, 290, 292] setzten visuelle Informationen ein, um einen mobilen Roboter in einem größeren Terrain zu lokalisieren. Dazu wurden Bildmerkmale verwendet, um in einer vorgegebenen Karte der Umgebung nach einer Position zu suchen, für die die

wahrgenommenen Merkmale den theoretisch vorhandenen Merkmalen am besten entsprechen. Burgard et al. [39] verwendeten eine zweidimensionale Gitterkarte, um die Position eines mit Sonar-Sensoren ausgestatteten mobilen Roboters zu bestimmen. Jeder Punkt der Karte ist ein Maß für die Wahrscheinlichkeit, daß sich der Roboter an dieser Position befindet. Yamauchi und Beer [325] setzten die Odometrie des Roboters ein, um eine topologische Karte der Umgebung aufzubauen. Kurz [169] erzeugte mit Hilfe einer selbstorganisierenden Karte eine topologische Karte der Umgebung. Der Roboter nahm seine Umgebung über Ultraschall-Sensoren wahr. Von Wichert [311, 312] und von Wichert und Tolle [313] entwickelten eine selbstorganisierende visuelle Repräsentation der Umgebung. Die topologische Karte der Umgebung wurde mit Hilfe von Rundumsichten aufgebaut. Die Bilder wurden mit einer selbstorganisierenden Karte in verschiedene Merkmalsbereiche eingeteilt. Für die Bereiche wurden verschiedene geometrische Momente berechnet, die schließlich zur Lokalisation des Roboters herangezogen wurden. Crowley et al. [61] berechneten aus den Abstandsinformationen eines Laser-Scanners die Hauptkomponenten. Als erstes erzeugten sie mit Hilfe von Trainingsdaten eine Tabelle, die Eigenwerte auf mögliche Positionen des Roboters abbildet. Die aktuellen Abstandsinformationen wurden in den Raum der Eigenwerte projiziert und als Index der Tabelle verwendet. In der Regel wurden mehrere mögliche Positionen gefunden. Ein Kalman Filter wurde eingesetzt, um während der Bewegung des Roboters ungültige Positionen herauszufiltern. Beetz et al. [21] setzten eine von Burgard et al. [40, 104] entwickelte aktive Methode zur Lokalisation eines mobilen Roboters ein. Um sich zu relokalisieren bewegte sich der Roboter zu einer Position, die zur Lokalisation am geeignetsten ist. Dabei wurden die durch die Bewegung entstehenden Kosten mitberücksichtigt. Gutmann et al. [117] verglichen zwei unterschiedliche Lokalisationsmethoden, einen wahrscheinlichkeitsbasierten Gitterkarten-Ansatz und einen Ansatz, der Abstandsinformationen mit einer vorgegebenen Karte vergleicht.

Balakrishnan und Honavar [11] verfolgten einen biologisch motivierten Ansatz zur Lokalisation eines mobilen Roboters indem sie den Hippocampus modellierten. Der simulierte Roboter erstellte ein neuronales Modell der Umgebung. Neue Sensorwerte wurden zusammen mit Odometrie-Informationen gespeichert. Falls später ähnliche Sensorwerte auftraten, wurde die zugehörige Positionsinformation abgefragt und zurückgeliefert. Diese Information wurde verwendet, um Fehler in der Lokalisation des Roboters zu korrigieren. Yamada [324] evolvierte Verhalten für einen mobilen Roboter, mit denen unterschiedliche Umgebungen erkannt werden konnten.

Der hier vorgestellte Ansatz unterscheidet sich völlig von den traditionellen Methoden zur Roboterlokalisation. Es wurde versucht, eine Funktion zu evolvieren, die Roboter-Positionen aufgrund von Sensorwerten berechnet. Diese Funktion stellt ein kompaktes Modell der Umgebung dar. Mit Hilfe der Evolution wurde dieses Modell an die Datensätze angepaßt, die über die Umgebung bekannt sind. Nordin et al. [222, 224, 225] evolvierten ein Modell der Umwelt, um einen mobilen Roboter zu steuern. Für ihre Experimente setzten sie genetisches Programmieren mit einem linearen Genotyp ein. Das Modell von Nordin und Banzhaf war eine Funktion, die die zu erwartende Fitneß von Steuerkommandos unter Verwendung der aktuellen Sensorinformationen abschätzte. Ein Planungsprozeß durchsuchte den Raum der möglichen Steuerkommandos und berechnete für jedes Kommando die zu erwartende Fitneß. Das Kommando mit der höchsten erwarteten Fitneß wurde schließlich ausgeführt. Danach wurde das Kommando und die tatsächliche Fitneß des Kommandos gespeichert. Die Liste mit den gespeicherten Kommandos und den zugehörigen Fitneßwerten wurde von einem Lernprozeß verwendet, um die Funktion zu evolvieren, die die zu erwartenden Fitneßwerte vorhersagt.

3.5 Zusammenfassung

Die oben beschriebenen Experimente zeigen, daß es möglich ist, durch genetisches Programmieren ein Modell der Umgebung zu evolvieren. Dieses Modell kann zur Lokalisation eines mobilen Roboters eingesetzt werden. Evolviert wurde eine Abbildung, die Sensorwerte auf Positionen des Roboters abbildet. Um den Suchraum zu verkleinern, wurden die rohen Sensordaten vorverarbeitet. Dazu wurden die Momente der Abstandsverteilung berechnet. Die Momente sollen die relevanten Merkmale der Umgebung hervorheben. Gleichzeitig soll aber keine wesentliche Information durch die Transformation der Eingabewerte verloren gehen. Die Momente wurden zusammen mit Zufallsvariablen als Terminal-Symbole verwendet. Als elementare Funktionen wurden arithmetische und trigonometrische Funktionen sowie ein bedingter Befehl eingesetzt.

Die Genauigkeit des Verfahrens könnte möglicherweise durch den Einsatz größerer Populationen erhöht werden. Ferner könnte die berechnete Position des Roboters während der Bewegung des Roboters gefiltert werden, um ihn so genauer zu lokalisieren. Außerdem könnte eine andere Art der Vorverarbeitung für die Lokalisation eines mobilen Roboters besser geeignet sein. Die Art der Vorverarbeitung könnte ebenfalls evolviert werden.

Kapitel 4

Evolution einer Steuerung für einen mobilen Roboter

Im folgenden soll untersucht werden, ob genetisches Programmieren eingesetzt werden kann, eine verhaltensbasierte Kontrollarchitektur für einen mobilen Roboter, einen Real World Interface B21 [244, 243] zu evolvieren (siehe Ebner und Zell [83, 86] und Ebner [77]). In Abbildung 4.4 ist der Roboter in der Umgebung zu sehen, die zur Evolution der Kontrollarchitektur verwendet wurde. Die Programmierung von Robotern ist in der Regel ein sehr aufwendiger und langwieriger Prozeß. Zur Programmierung von Manipulatoren wird häufig die zu programmierende Sequenz von Hand einprogrammiert und später einfach wieder abgespielt [197]. Die Programmierung von Robotern, die dynamisch auf ihre Umgebung reagieren müssen, ist dagegen deutlich schwieriger. Dies trifft ganz besonders auf mobile Roboter zu, die in zuvor unbekannter Umgebung eingesetzt werden sollen. Durch unvorhergesehene Wechselwirkungen zwischen dem Roboter und seiner Umgebung kann sich ein Programm völlig anders verhalten, als es vom Programmierer geplant war. In der Regel sind oft mehrere Zyklen zwischen Entwicklung und Erprobung erforderlich. Mit Hilfe der Darwinschen Evolution könnte sich dieser Prozeß automatisieren lassen. Braitenberg führte hierzu eine Reihe von Gedankenexperimenten durch [28]. Inzwischen hat sich der Einsatz evolutionärer Algorithmen in der Robotik zu einem aktiven Forschungsbereich, der evolutionären Robotik, entwickelt.

Fragen der evolutionären Robotik werden ausführlich von Harvey et al. [124] diskutiert. Matarić und Cliff [194] und Meyer et al. [199] geben einen ausführlichen Überblick über das Gebiet der evolutionären Robotik. Es ist möglich, Experimente lediglich in der Simulation durchzuführen, die Evolution vollständig auf einem echten Roboter ablaufen zu lassen oder eine Mischung der beiden Verfahren einzusetzen [219]. Wird mit echten Robotern gearbeitet, dann wird in der Regel ein Individuum nach dem anderen auf ein und demselben Roboter ausgewertet. Wie wichtig es ist, echte Roboter einzusetzen, anstatt sie zu simulieren, wird von Brooks [34] betont. Ein realer Roboter muß in der Lage sein, verrauschte Sensorinformationen zu verarbeiten. Ultraschall-Sensoren werden oft reflektiert, was dazu führt, daß ein längerer Abstand zum nächsten Hindernis gemessen wird, als tatsächlich vorhanden ist. Außerdem kann keine genaue Steuerung garantiert werden. Falls sowohl mit Simulation als auch mit einem realen Roboter gearbeitet wird, stellt sich das Problem, die Kluft zwischen der Simulation und dem realen Roboter zu schließen [143]. Manchmal wird eine sehr exakte Simulation angestrebt, wie z.B. bei Miglino et al. [200], so daß die in der Simulation evolvierten Individuen sofort oder nur nach einer kurzen Fortsetzung der Evolution mit dem realen Roboter das

gleiche Verhalten in der Realität zeigen. In anderen Fällen wird dagegen eine bewußt minimale Simulation durchgeführt, da argumentiert wird, daß nur das simuliert werden muß, was in der Realität auch auftreten kann [142]. Dadurch kann mit größeren Populationen gearbeitet werden oder man kann das Experiment für eine größere Zahl von Generationen durchführen. Simulationen eignen sich besonders für die Suche nach einer geeigneten Repräsentation und für die Suche nach geeigneten Parametern für die Evolution. Denn in der Simulation können die Experimente im Zeitraffer durchgeführt werden. Ein weiterer Vorteil der Simulation ist die Möglichkeit, ein Experiment unter exakt den gleichen Bedingungen zu wiederholen.

Neuronale Netze wurden bereits vielfach in der Robotik eingesetzt [230, 296, 36]. Auch im Bereich der evolutionären Robotik wurden meist neuronale Netze als Kontrollarchitektur eingesetzt. Nolfi [218], Floreano et al. [96, 95, 93, 92], sowie Mondada und Floreano [204] verwendeten einen genetischen Algorithmus, um nach geeigneten Gewichten für ein neuronales Netz mit gegebener Topologie zu suchen. Salomon [259] und Biró und Ziemke [25] evolvierten die Gewichte eines neuronalen Netzes mit einer Evolutionsstrategie. Floreano und Mondada [94] evolvierten plastische neuronale Netze. Dabei spezifizierte der Genotyp der Individuen den Mechanismus, mit dessen Hilfe die Gewichte während der Ausführung eines Individuums angepaßt wurden. Prinzipiell können neuronale Netze natürlich auch mit einem Lernverfahren trainiert werden, falls ein geeignetes bestärkendes Signal vorhanden ist [201, 332, 331, 273]. Ballard [12] gibt eine Einführung in die verschiedensten Lernverfahren, von neuronalen Netzen bis hin zu genetischen Lernverfahren. Meeden [198] vergleichte experimentelle Ergebnisse, die mit komplementärem, bestärkendem Backpropagation erreicht wurden, mit Ergebnissen, die mit einem evolutionären Algorithmus erreicht wurden, um geeignete Gewichte für ein neuronales Netz eines mobilen Roboters zu finden. In den durchgeführten Experimenten zeigten beide Verfahren sowohl Stärken als auch Schwächen, so daß eine Kombination beider Verfahren vorteilhaft sein könnte.

Wird eine vorgegebene Netz-Topologie verwendet, so stellt sich immer noch die Frage, welche Netz-Topologie die für das Problem geeignetste ist. Zur Evolution der Netz-Topologie können ebenfalls evolutionäre Algorithmen eingesetzt werden [7, 191]. Harvey et al. [122, 121, 123, 124], Cliff et al. [51] und Calabretta et al. [41] evolvierten auch die Netz-Topologie. Eine optimale Anordnung der Sensoren in Verbindung mit einer Steuerung eines mobilen Roboters wurde von Cliff et al. [51], Harvey et al. [123] und Huber et al. [136] evolviert. Sims [278] und Kikuchi und Hara [157] evolvierten außer den Algorithmen zur Steuerung des Roboters auch dessen Morphologie.

Dorigo et al. [70, 71], Colombetti et al. [53, 52], Donnart und Meyer [69], Lopez und Smith [188] setzen sogenannte *Classifier Systems* [126] ein, um einen mobilen Roboter zu steuern. Cliff und Bullock [50] untersuchten den Einfluß der Sensormorphologie auf das Verhalten eines Agenten ebenfalls im Zusammenhang mit einem *Classifier System*. Doch auf diese Arbeiten soll hier nicht näher eingegangen werden.

Im Gegensatz zu den oben genannten Arbeiten wird hier genetisches Programmieren eingesetzt, um eine verhaltensbasierte Kontrollarchitektur für einen mobilen Roboter zu evolvieren. Mit Hilfe von genetischem Programmieren haben die Roboter selbst die Möglichkeit, den Raum der Programme nach einer geeigneten Kontrollarchitektur zu durchsuchen. Genetisches Programmieren bietet den Vorteil, daß elementare Funktionen prinzipiell auf jeder Abstraktionsebene definiert werden können. Es lassen sich einfache Befehle in Maschinensprache bis hin zu komplexeren Verhaltensmustern als elementare Funktionen definieren. Im folgenden wird zuerst ein Überblick über verhaltensbasierte Kontrollarchitekturen gegeben und dann die Repräsentation beschrieben, die zur Evolution einer Kontrollarchitektur ein-

Abbildung 4.1: Struktur traditioneller Kontrollarchitekturen in der Robotik (nach Brooks [33]).

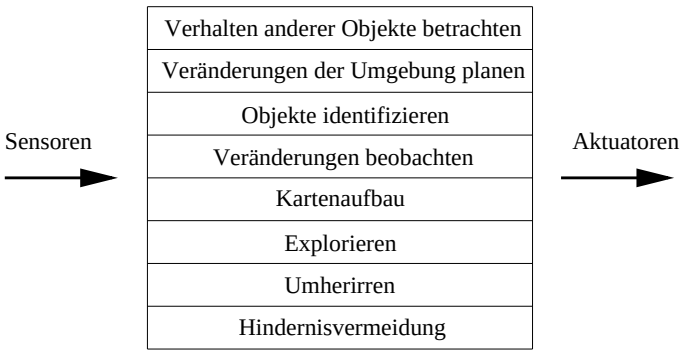


Abbildung 4.2: Struktur verhaltensbasierter Kontrollarchitekturen in der Robotik (nach Brooks [33]).

gesetzt wurde. Danach werden die in der Simulation durchgeführten Experimente, gefolgt von dem Experiment mit dem echten Roboter beschrieben. Schließlich werden am Ende des Kapitels verwandte Arbeiten diskutiert.

4.1 Verhaltensbasierte Kontrollarchitekturen

In der traditionellen Robotik wurde meist das Kontrollsystem eines Roboters, der eine bestimmte Aufgabe zu erfüllen hatte, in einzelne Module aufgeteilt, wie es in Abbildung 4.1 dargestellt ist. Der Roboter nimmt über Sensoren seine Umgebung wahr. Mit Hilfe der Sensordaten wird ein Modell der Umgebung erstellt. Anhand dieses Modells wird ein Plan erstellt. Aus dem Plan ergibt sich eine Handlung und die erforderlichen Signale, um die Aktuatoren anzusteuern.

Im Gegensatz zur traditionellen Robotik wurde von Brooks [33] eine Aufteilung des Kontrollsystems in einzelne Verhalten vorgeschlagen (Abbildung 4.2). Elementare Verhalten können z.B. die Exploration der Umgebung, die Fahrt durch einen Korridor oder eine Hindernisvermeidung sein. Jedes einzelne Verhalten erhält als Eingabe alle benötigten Sensorwerte und kann prinzipiell die Aktuatoren ansteuern. Zur Ansteuerung der Aktuatoren muß selbstverständlich ein geeigneter Mechanismus gewählt werden, der die eventuell widersprüchlichen Signale in geeigneter Weise miteinander kombiniert. Bei der Subsumption-Architektur ist dies über feste Prioritäten geregelt. Die Signale, die von einem Modul der Subsumption-Architektur zu einem anderen gesendet werden, können entweder über einen definierten Zeitraum hinweg unterdrückt oder durch ein anderes Signal überschrieben werden.

Durch diesen Mechanismus kann ein komplexes System, wie es in Abbildung 4.3 dargestellt ist, aufgebaut werden. Dabei sind in den unteren Schichten einfache Verhalten realisiert,

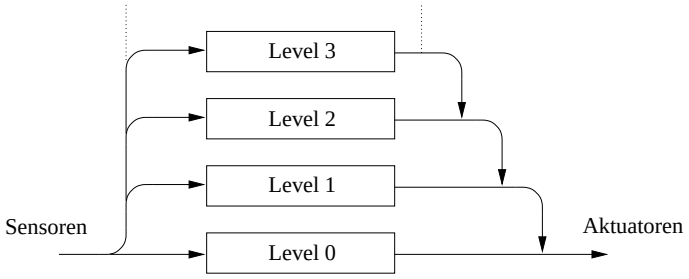


Abbildung 4.3: Brooks' Subsumption-Architektur. Die höheren Schichten subsumieren die Funktionalität der darunter liegenden Schichten, falls diese die Kontrolle übernehmen möchten (nach Brooks [33]).

während in den höheren Schichten komplexere Verhalten realisiert sind. Die höheren Schichten sind in der Lage, die Funktionalität der daruntergelegenen Schichten zu subsumieren. Dadurch ist es möglich, durch das Hinzufügen neuer Module, bzw. Verhaltensweisen ein immer komplexeres System aufzubauen. Ein weiterer Vorteil dieses Systems ist, daß es bereits nach der Fertigstellung der ersten Schicht eingesetzt werden kann. Es bleibt auch weiterhin, natürlich in gewissen Grenzen, funktionstüchtig, wenn einmal eine Schicht ausfallen sollte.

Verhaltensbasierte Kontrollarchitekturen haben sich als sehr geeignet zur Steuerung mobiler Roboter erwiesen. Arkin [9] gibt eine detailierte Einführung in die verhaltensbasierte Robotik. Den verhaltensbasierten Architekturen ist gemein, daß einzelne Verhalten durch einen geeigneten Mechanismus miteinander kombiniert werden. Meist wird eine explizite Repräsentation von Wissen vermieden und es existiert eine enge Kopplung von Wahrnehmung und Handlung. Kontrollarchitekturen, die eine enge Kopplung zwischen Wahrnehmung und Handlung besitzen, ohne daß eine abstrakte Repräsentation der Information erstellt wird oder Informationen gespeichert werden, werden als reaktive Kontrollarchitektur bezeichnet. Kontrollarchitekturen können nach der Art der Kodierung der Verhalten, z.B. diskret oder kontinuierlich, und nach der Art des Mechanismus, der zur Koordination der einzelnen Verhalten eingesetzt wird, z.B. kompetitiv oder kooperativ, unterschieden werden. Ein Beispiel für eine Architektur für eine diskrete Kodierung der Verhalten in Verbindung mit einer kompetitiven Koordination ist die Subsumption-Architektur [33]. In der von Arkin entwickelten Architektur, die aus sogenannten Bewegungsschemata zusammengesetzt ist, werden die Verhalten kontinuierlich kodiert und durch Vektoraddition zusammengefaßt [8, 9].

4.2 Evolution verhaltensbasierter Kontrollarchitekturen

Koza [165] zeigte, daß genetisches Programmieren prinzipiell zur Evolution einer Subsumption-Architektur eingesetzt werden kann. Dazu analysierte er einen von Mataric programmierten Roboter, der ein Folge-der-Wand-Verhalten realisierte. Die Experimente von Koza wurden lediglich in der Simulation durchgeführt. Wie schon der Roboter von Mataric, so konnte auch Koza's simulierter Roboter sich nur in diskreten Schritten bewegen. Auf die Arbeit von Koza wird in Abschnitt 4.6 genauer eingegangen. In Generation 57 des Experimentes fand Koza ein Individuum, bestehend aus 145 Knoten, das das vorgegebene Problem löste. Er verwendete eine Populationsgröße von 1000 Individuen.

Hier sollte jedoch versucht werden, eine Kontrollarchitektur vollständig auf einem realen

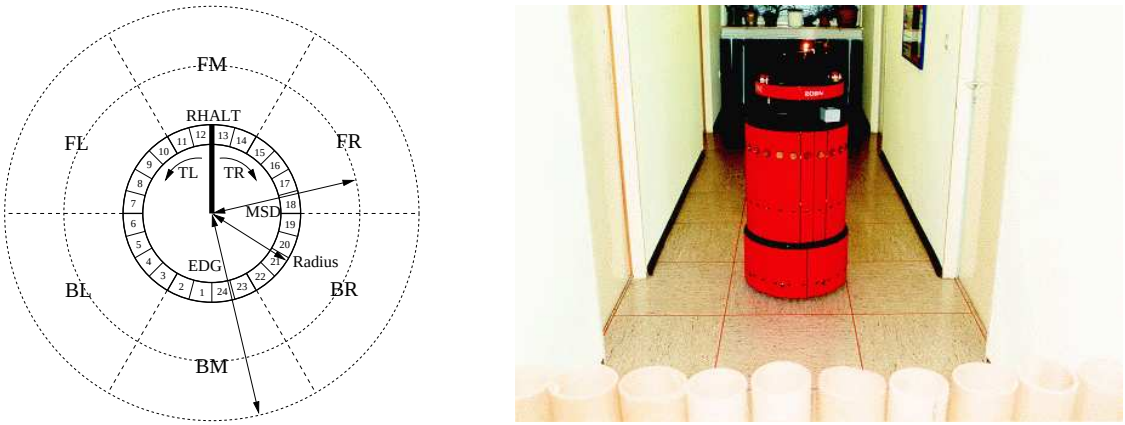


Abbildung 4.4: Der im Experiment zur Evolution einer Kontrollarchitektur eingesetzte Service-Roboter, ein Real World Interface B21 [244, 243], ist auf der rechten Seite zu sehen. Der Roboter konnte sich lediglich innerhalb des abgesperrten Bereiches bewegen. Auf der linken Seite ist der Roboter schematisch dargestellt (Aufsicht). Die 24 Ultraschall-Sensoren werden zu 6 virtuellen Sensoren kombiniert: `FL` (vorne links), `FM` (Mitte vorne), `FR` (vorne rechts), `BL` (hinten links), `BM` (Mitte hinten), `BR` (hinten rechts). Der Roboter bewegt sich mit konstanter Geschwindigkeit vorwärts. Er kann mit dem Terminal-Symbol `TL` mit konstanter Geschwindigkeit nach links gedreht bzw. mit dem Terminal-Symbol `TR` mit konstanter Geschwindigkeit nach rechts gedreht werden. Das Terminal-Symbol `RHALT` stoppt die Drehung des Roboters. Zwei Terminal-Symbole stehen als Konstanten zur Verfügung: `EDG`, der gewünschte Abstand zur Wand, und `MSD`, der kleinste, noch sichere Abstand zur Wand.

mobilen Service-Roboter zu evolvieren. Daher mußte die Populationsgröße auf 75 Individuen und die Zahl der Generationen auf 50 limitiert werden. Ein erster Test mit der von Koza verwendeten Repräsentation führte zu keinen Ergebnissen. Daher wurde versucht, den Suchraum zu verkleinern. Dies wird am einfachsten durch die Reduktion der Zahl der verwendeten Sonar-Sensoren erreicht. Ferner wurde eine für einen mobilen Service-Roboter angemessene, kontinuierliche Steuerung verwendet.

4.2.1 Terminal-Symbole

Die 24 Ultraschall-Sensoren des Roboters werden zu 6 virtuellen Sensoren zusammengefaßt. Die 6 virtuellen Sensoren `FL` (vorne links), `FM` (Mitte vorne), `FR` (vorne rechts), `BL` (hinten links), `BM` (Mitte hinten), `BR` (hinten rechts) sind in Abbildung 4.4 graphisch dargestellt. Jeder der 6 Terminal-Symbole gibt als Wert jeweils das Minimum der 4 realen Sensoren zurück. Die Verwendung des Minimums erhöht die Robustheit des virtuellen Sensors. Denn aufgrund von Reflektionen kann es vorkommen, daß ein längerer Abstand gemessen wird, als tatsächlich vorliegt. Neben den 6 virtuellen Sensoren wird ein weiteres Terminal-Symbol `SS` eingesetzt, das das Minimum aller 24 realen Sensoren als Rückgabewert liefert. Ferner stehen zwei Terminal-Symbole zur Verfügung, die einen konstanten Wert liefern: `EDG`, der gewünschte Abstand zur Wand beträgt 50cm, und `MSD`, die kleinste, noch sichere Entfernung zur Wand beträgt 70cm.

Der Roboter bewegt sich mit einer konstanten Geschwindigkeit von $10\frac{\text{cm}}{\text{s}}$ vorwärts. Die Drehgeschwindigkeit des Roboters kann über drei Terminal-Symbole gesteuert werden. `TL` dreht den Roboter mit der konstanten Geschwindigkeit $-40\frac{\text{°}}{\text{s}}$ nach links, `TR` dreht den Roboter

mit der konstanten Geschwindigkeit $+40\frac{°}{s}$ nach rechts und `RHALT` stoppt die Drehung des Roboters. `TL` liefert als Rückgabewert den Durchschnitt der Abstände, die von den beiden realen Sensoren 11 und 12 gemessen werden. `RHALT` liefert den Durchschnitt der Abstände der Sensoren 12 und 13 und `TR` liefert den Durchschnitt der Abstände der Sensoren 13 und 14 zurück.

4.2.2 Elementare Funktionen

Als elementare Funktionen wird der bedingte Befehl `IFLTE` und die Funktion `PROGN2` eingesetzt. `IFLTE` ist eine Funktion mit vier Argumenten. Falls das erste Argument einen Wert hat, der kleiner als oder gleich groß ist wie der Wert des zweiten Argumentes, wird das dritte Argument ausgewertet, sonst wird das vierte Argument ausgewertet. Die Funktion `PROGN2` hat zwei Argumente. Zuerst wird das erste Argument ausgewertet, danach wird das zweite Argument ausgewertet. Der Wert des zweiten Argumentes wird zurückgegeben.

4.2.3 Berechnung der Fitneß

Zur Berechnung der Fitneß eines Individuums wurde es für eine begrenzte Zeit getestet. Der Test wurde entweder mit Ablauf der zur Verfügung stehenden Zeit abgebrochen oder falls das Individuum zu nahe an eine Wand kam. Dazu wurden sowohl die Ultraschall-Sensoren als auch die taktilen Sensoren ausgewertet. Der Test wurde abgebrochen, falls von einem Ultraschall-Sensor eine Distanz gemessen wurde, die kleiner als 40cm war. Bei den Experimenten mit dem Service-Roboter konnte es vorkommen, daß aufgrund von Reflektionen ein Hindernis nicht erkannt wurde, und es somit zu einer Kollision kam. In diesem Fall stellte der Roboter über die taktilen Sensoren fest, daß eine Kollision stattgefunden hatte, und der Test wurde ebenfalls abgebrochen. In der Simulation ist es möglich, jedes Individuum unter den gleichen Anfangsbedingungen auszuwerten. Bei einem realen Roboter ist dies natürlich aufgrund der begrenzten Genauigkeit der Repositionierungsalgorithmen nicht möglich. Um allen Individuen eine faire Chance zu geben, wurden die Individuen repositioniert um einen weiteren Test durchzuführen. Dazu wurde der Roboter in eine Richtung entgegengesetzt zu eventuell vorhandenen Hindernissen gedreht. Die Richtung wurde analog zur Potentialfeldmethode [215, 9], die zur Steuerung mobiler Roboter eingesetzt wird, mit einem logarithmischen Potential bestimmt. Anschließend wurde der Roboter 15cm vorwärts bewegt.

Nachdem der Test abgebrochen wurde, wurde die Fitneß des Individuums berechnet. Hierbei wurde versucht, die Zeit bis zum Abbrechen des Tests zu maximieren und dabei möglichst wenig Rotationen auszuführen. Würde man versuchen, lediglich die Zeit bis zum Abbrechen des Tests zu maximieren, dann hätte ein Individuum, das sich ständig im Kreis dreht, eine maximale Fitneß. Ein solches Individuum ist jedoch völlig ungeeignet, um einen mobilen Roboter durch einen Gang zu steuern. Zur Optimierung mehrerer unabhängiger Kriterien eignet sich die Pareto-Optimierung [102]. In diesem Fall erhält man als Ergebnis eine Reihe von optimalen Individuen. Diese Menge muß anschließend manuell ausgewertet werden. In den hier durchgeführten Experimenten wurde eine Fitneßfunktion eingesetzt, die in ähnlicher Form bereits in der evolutionären Robotik eingesetzt wurde.

Zur Berechnung des Fehlers eines Tests wurde die Zeit bis zum Abbruch des Tests gemessen, und ein Maß für die während des Tests durchgeführten Rotationen berechnet. Um die Notation im folgenden zu vereinfachen, werden die folgenden Variablen verwendet. Es sei t die Zeit, bis der Test abgebrochen wurde und T die gesamte Zeit, die dem Individuum für

den Test zur Verfügung stand. Die normalisierte Zeit sei $D = \frac{t}{T}$. Es sei r_s die Summe aller Rotationen, die vom Roboter während des Tests gemacht wurden und r_u sei die Summe aller absoluten Rotationen. Die Drehgeschwindigkeit des Roboters sei ω. Dann ist $R_s = \frac{|r_s|}{\omega t}$ ein Maß für die Zahl der nicht ausgeglichenen Rotationen.

Unter Verwendung dieser Variablen wird der Fehler f_i für den i-ten Test berechnet. In der Regel wurden mehrere Tests für ein Individuum durchgeführt und die Ergebnisse dieser Tests zu einer Fitneß des Individuums kombiniert. Hierzu wird häufig einfach der Mittelwert berechnet: Fehler $= \frac{1}{n} \sum_{i=1}^{n} f_i$. Dabei gibt n die Zahl der einzelnen Tests, die für jedes Individuum durchgeführt werden, an. Über Fitneß $= \frac{1}{1+\text{Fehler}}$ kann dann die Fitneß eines Individuums berechnet werden. Weitere Verfahren zur Kombination der einzelnen Tests zur Bewertung eines Individuums sind z.B. die Berechnung des Medians oder des Minimums. Das Minimum der einzelnen Tests wurde z.B. von Harvey et al. [124, 123], Cliff et al. [51] und von Reynolds [252] eingesetzt.

4.3 Experimente in der Simulation

Mit der oben beschriebenen Repräsentation wurde eine Vielzahl von Experimenten in der Simulation durchgeführt. Schließlich wurde ein Experiment mit dem Service-Roboter wiederholt. Für die Experimente wurden die folgenden Parameter für die Evolution verwendet. Es wurde eine Populationsgröße von 75 Individuen gewählt. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 6 gefüllt. Die maximale Zahl der Knoten eines Individuums wurde auf 1000 und die Tiefe des Individuums auf 17 begrenzt. Es wurde Tournament-Selektion mit der Crossover-Wahrscheinlichkeit $p_{\text{cross}} = 0.85$, der Reproduktions-Wahrscheinlichkeit $p_{\text{rep}} = 0.1$ und der Mutations-Wahrscheinlichkeit $p_{\text{mut}} = 0.05$ verwendet. Die Experimente wurden jeweils nach 50 Generationen abgebrochen. Bevor jedoch die Ergebnisse der Experimente präsentiert werden, werden zunächst die Experimentierumgebung und die mit selbstentwickelten Algorithmen erreichten Resultate beschrieben.

4.3.1 Die Experimentierumgebung

Um Experimente in der Simulation durchzuführen, wurde ein Simulator für einen mobilen Roboter entwickelt. Der Simulator läuft als separater Prozeß und empfängt von der Kontrollarchitektur die Befehle zur Steuerung des mobilen Roboters. Die Kommunikation erfolgt über *Shared-Memory*. Auf diesem Weg kann die Kontrollarchitektur die Sensorwerte auslesen. Der Zugriff wird mit Hilfe von Semaphoren [293, 277] geregelt. Da zwei separate Prozesse eingesetzt werden, kann es zu Verzögerungen bei der Ausführung der Befehle kommen. Je nachdem, wie lange diese Verzögerung dauert, hat sich der Roboter in diesem Zeitraum unterschiedlich weit bewegt. Ähnliche Verzögerungen können auch bei dem echten Service-Roboter auftreten, da dieser über eine verteilte Client-Server-Architektur gesteuert wird [245, 246, 329]. Die Simulation erfolgt im Zeitraffer. Dadurch können wesentlich mehr Experimente durchgeführt werden, als mit einem Simulator, der in Echtzeit arbeitet.

Die Umgebung des Roboters ist intern als Menge von Linien gespeichert. Hierzu wird die von Vista [239] bereitgestellte Datenstruktur VEdges genutzt. Der simulierte Roboter besitzt 24 Sonar-Sensoren und 16 taktile Sensoren. Für jeden Sonar-Sensor wird die Abstandsinformation aus dem nächsten Schnittpunkt des Sonar-Strahls mit der Umgebung berechnet. In

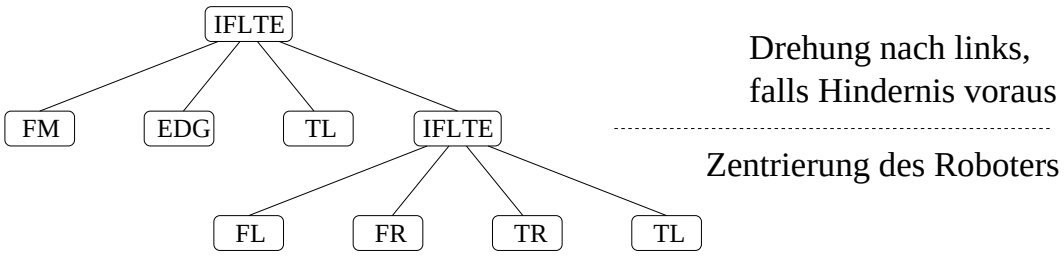


Abbildung 4.5: Kontrollarchitektur, die den mobilen Roboter in der Mitte eines Korridors zentriert. Die Kontrollarchitektur wurde manuell um den Teil erweitert, der feststellt, ob sich vor dem Roboter ein Hindernis befindet.

der Computergrafik wird dies als *ray tracing* bezeichnet [100]. Um die Werte der taktilen Sensoren zu berechnen, wird der Roboter als Kreis modelliert. Es liegt eine Kollision vor, falls der Roboter eine der Umgebungslinien berührt. Die in der Simulation eingesetzten Umgebungen wurden mit dem Programm XFIG [286] erstellt. Der zurückgelegte Weg des Roboters kann entweder als XFIG-Datei oder als Postscript-Datei [1] abgespeichert werden.

4.3.2 Selbstentwickelte Individuen

Um sicher zu gehen, daß sich im Suchraum auch geeignete Individuen befinden, die das Problem lösen, wurden einige Individuen manuell erstellt. Es werden drei Individuen vorgestellt. Eines der hier besprochenen Individuen entstand während einer der unten beschriebenen Experimente und wurde lediglich manuell ergänzt. Es hat eine verblüffend einfache Struktur und zeigt dennoch ein scheinbar komplexes Verhalten (siehe auch Braitenberg [28] für eine Reihe von Individuen, die mit einer sehr einfachen neuronalen Kontrollarchitektur realisiert wurden und dennoch ein scheinbar komplexes Verhalten zeigen). Dieses Individuum wird als erstes besprochen. Im Vergleich zu diesem Individuum hatten die ersten manuell erstellten Individuen eine wesentlich komplexere Struktur. Das Individuum verwendet lediglich die Sensoren vorne links und vorne rechts, um den Roboter in der Mitte eines Korridors zu halten. Das erweiterte Individuum ist in Abbildung 4.5 dargestellt. Der obere Teil des Individuums, der prüft, ob sich vor dem Roboter ein Hindernis befindet, wurde manuell ergänzt. Falls sich ein Hindernis vor dem Roboter befindet (falls der Abstand, der durch den Sensor `FM` ermittelt wird, kleiner als die gewünschte Entfernung zur Wand ist), dreht sich der Roboter nach links. Sonst werden die beiden Sensoren, die sich vorne links und vorne rechts befinden, miteinander verglichen. Falls der Wert, der vom Sensor vorne links ermittelt wird, kleiner ist als der Wert, der vom Sensor vorne rechts ermittelt wird, dreht sich der Roboter nach rechts, im anderen Fall dreht sich der Roboter nach links. Dieses Individuum zeigt, daß erfolgreiche Kontrollarchitekturen bereits in einer Tiefe von 2 gefunden werden können. Dies setzt natürlich eine Umgebung voraus, in der diese Strategie angewendet werden kann. Das Verhalten des Individuums ist für zwei verschiedene Umgebungen in Abbildung 4.8 zu sehen. Die Strategie schlägt in der zweiten Umgebung fehl, weil das dünne, spitze Hindernis den Sonar-Sensoren entgeht.

Das zweite Individuum zeigt ein Folge-der-Wand-Verhalten. Das Individuum hat ebenfalls die Tiefe 2 und ist in Abbildung 4.6 dargestellt. Das Individuum prüft zuerst, ob sich direkt vor dem Roboter ein Hindernis befindet. Befindet sich vor dem Roboter ein Hindernis, so

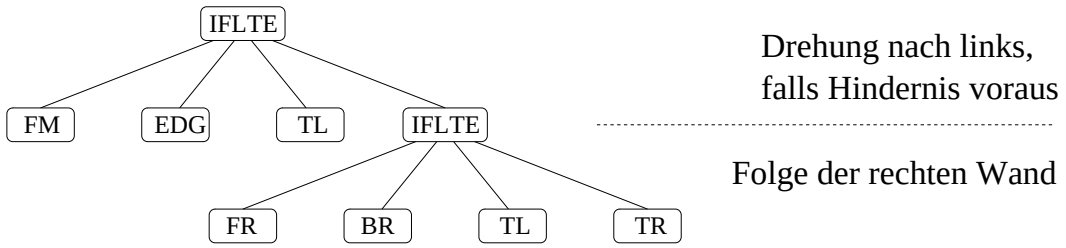


Abbildung 4.6: Individuum mit einem Folge-der-Wand-Verhalten. Das Individuum wurde manuell erstellt.

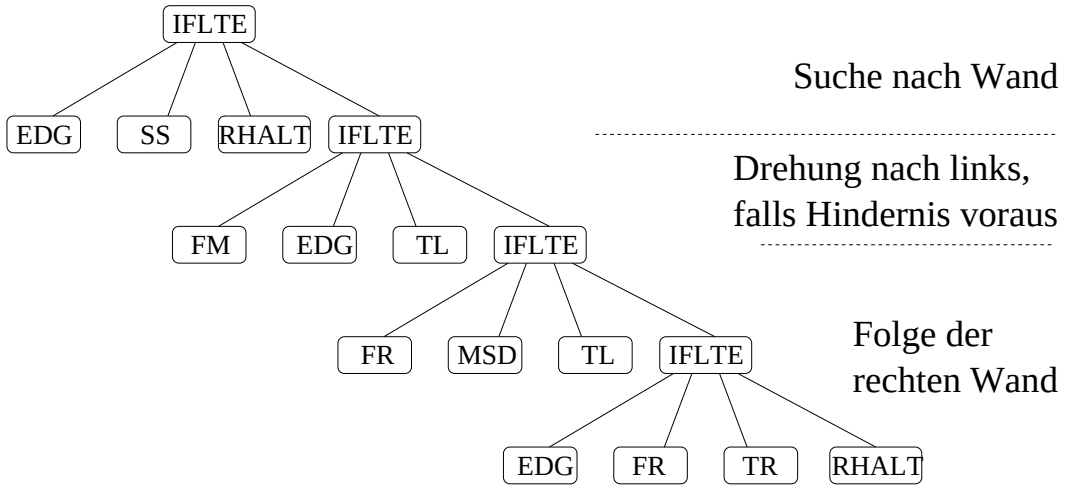


Abbildung 4.7: Weitere Variante eines Individuums mit einem Folge-der-Wand-Verhalten. Das Individuum wurde manuell erstellt.

wird der Roboter nach links gedreht. Andernfalls wird der Sensor vorne rechts und der Sensor hinten rechts dazu eingesetzt, der rechten Wand zu folgen. Der Roboter wird nach links gedreht, falls der Sensor vorne rechts einen kleineren Wert zurückliefert als der Sensor hinten rechts. Im anderen Fall wird der Roboter nach rechts gedreht. Dadurch bewegt sich der Roboter mit konstantem Abstand entlang der Wand, in dem er sich abwechselnd nach rechts und links dreht. Der Pfad des Individuums ist in Abbildung 4.8 zu sehen. Das Individuum funktioniert sehr gut in der größeren Umgebung, stößt aber leider in der kleineren Umgebung gegen eine Türkante. Das Individuum folgt zwar der Wand, erkennt jedoch die Kante an der Tür zu spät, so daß der Lauf abgebrochen wird.

Das dritte Individuum zeigt ebenfalls ein Folge-der-Wand-Verhalten. Das Individuum mit der Tiefe 4 ist in Abbildung 4.7 dargestellt. Das Individuum ist aus drei einzelnen, einander übergeordneten Verhaltensmustern aufgebaut. Zuerst prüft das Individuum, ob sich in der Nähe des Roboters ein Hindernis befindet. Ist dies nicht der Fall, dann bewegt sich der Roboter gerade aus. Sonst wird der Sensor `FM` abgefragt und mit dem gewünschten Abstand zur Wand verglichen. Ist der Abstand kleiner, so dreht sich der Roboter nach links. Andernfalls führt das Individuum ein Folge-der-Wand-Verhalten aus. Dazu wird der Sensor vorne rechts abgefragt. Falls dieser Wert kleiner als der kleinste noch sichere Abstand zur Wand ist, wird der Roboter nach links gedreht. Dadurch wird der Abstand zur Wand auf

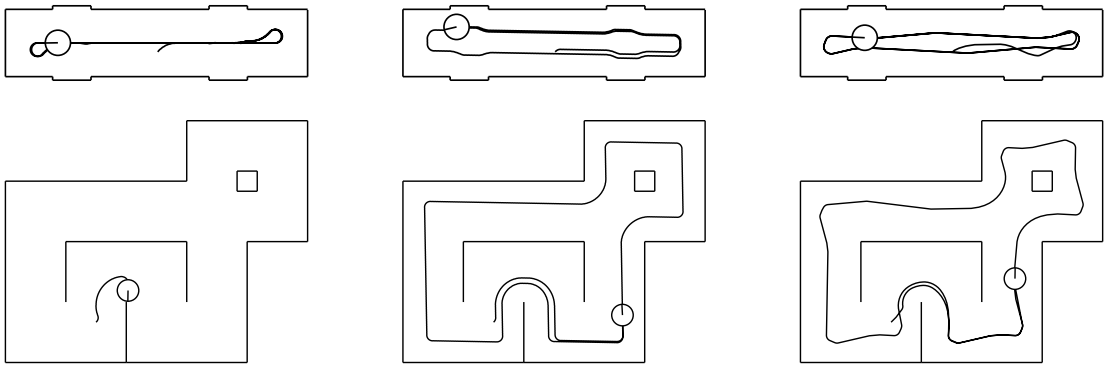


Abbildung 4.8: Das Verhalten von drei verschiedenen Individuen in zwei unterschiedlichen Umgebungen. Der Pfad auf der linken Seite wurde mit dem Individuum aus Abbildung 4.5, der Pfad in der Mitte mit dem Individuum aus Abbildung 4.6 und der Pfad auf der rechten Seite mit dem Individuum aus Abbildung 4.7 erzeugt.

der rechten Seite vergrößert. Ist der Abstand jedoch größer, wird überprüft, ob der Abstand vorne rechts größer als der gewünschte Abstand zur Wand ist. In diesem Fall wird der Roboter nach rechts gedreht, ansonsten wird die Rotation des Roboters gestoppt. Der Pfad dieses Individuums ist in Abbildung 4.8 zu sehen. Das Individuum bewegt sich erfolgreich in beiden Umgebungen.

4.3.3 Simulationsergebnisse in der modellierten Umgebung des Service-Roboters

In einer weiteren Reihe von Experimenten wurde ein Modell der Umgebung eingesetzt, wie sie für das Experiment mit dem mobilen Service-Roboter zur Verfügung stand. Die Umgebung des echten Roboters ist in Abbildung 4.4 zu sehen, das Modell dieser Umgebung wurde bereits in Abbildung 4.8 gezeigt. Zur Berechnung des Fehlers wurde $1 - D(1 - \sqrt{R_s})$ verwendet. Eine ähnliche Funktion wurde bereits von Floreano und Mondada [92, 204, 93] eingesetzt, um Gewichte eines neuronalen Netzes zur Steuerung eines Miniatur-Roboters zu evolvieren. Es wurde mit einer, zwei und drei Auswertungen je Individuum experimentiert. Je öfter ein Individuum getestet wird, desto robuster ist der daraus berechnete Fitneßwert. Daher sollten die Zahl der Tests möglichst groß sein. Da jede einzelne Auswertung aber die zur Berechnung der Fitneß erforderliche Zeit verlängert, sollte versucht werden, die Zahl der notwendigen Tests möglichst klein zu halten.

Es wurden jeweils 20 Experimente mit einem, zwei und drei Tests durchgeführt. Nur zwei erfolgreiche Individuen entstanden, als lediglich ein einzelner Test verwendet wurde. Bei zwei Tests je Individuum entstanden 6 erfolgreiche Individuen, bei drei Tests entstanden 8 erfolgreiche Individuen. Die erfolgreichen Individuen der Experimente sind in den Abbildungen 4.9, 4.10 und 4.11 zu sehen. Für jedes Experiment wurde für jede Generation die Fitneß des besten Individuums gespeichert. Neben den Individuen ist jeweils der Durchschnitt über alle Experimente und die Fitneß des Laufes zu sehen, der das beste Individuum produzierte.

Während des Experimentes mit drei Tests erreichte ein äußerst kleines Individuum die beste Fitneß: `(IFLTE FR FL TL TR)`.

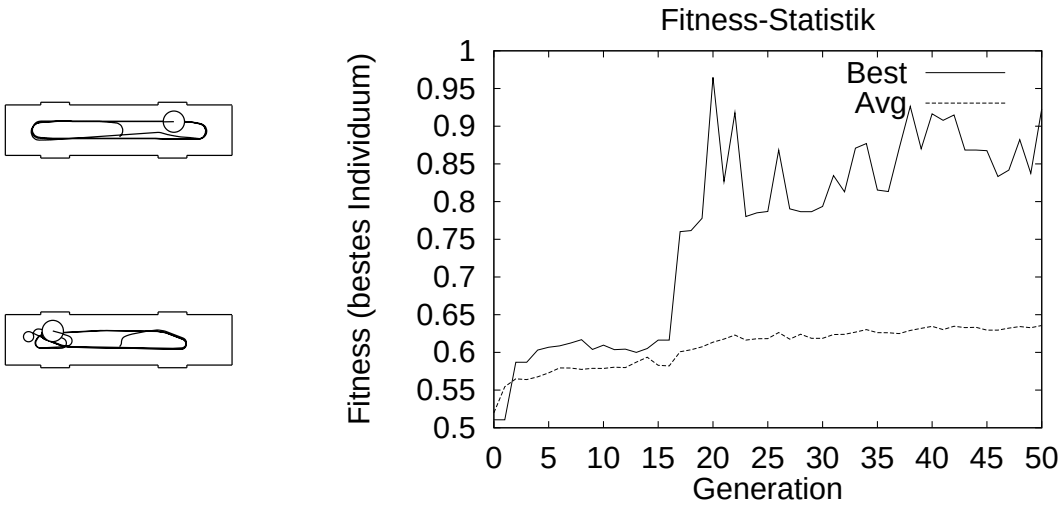


Abbildung 4.9: Mit nur einem Fitneßtest je Individuum entstanden lediglich 2 erfolgreiche Individuen. Für jedes Experiment wurde für jede Generation die Fitneß des besten Individuums gespeichert. Auf der rechten Seite ist der Mittelwert dieser Fitneß über alle Experimente und die Kurve für das Experiment zu sehen, das das Individuum mit der besten Fitneß produzierte.

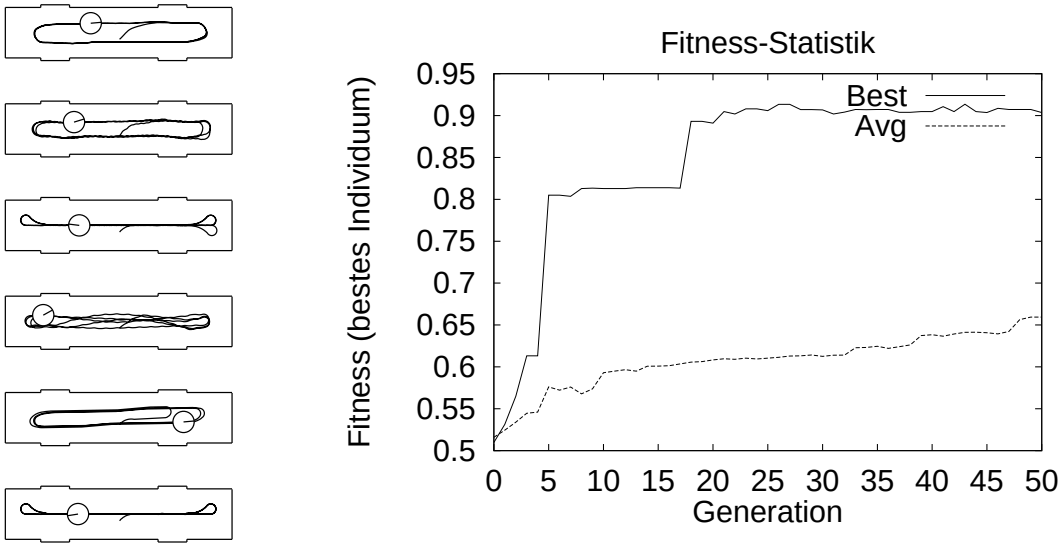


Abbildung 4.10: Mit 2 Fitneßtests je Individuum entstanden 6 erfolgreiche Individuen.

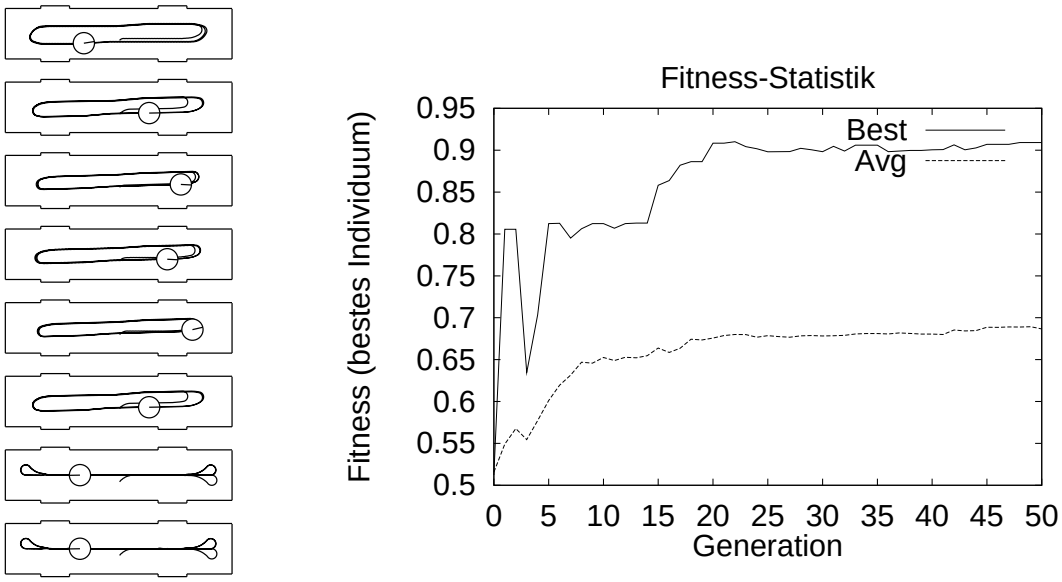


Abbildung 4.11: Mit 3 Fitneßtests je Individuum entstanden 8 erfolgreiche Individuen.

4.4 Experiment mit dem mobilen Service-Roboter

Eines der Experimente, die in der Simulation durchgeführt wurden, wurde schließlich auch mit dem mobilen Service-Roboter durchgeführt. Um die Dauer des Experimentes möglichst klein zu halten, wurden lediglich zwei Tests je Individuum verwendet. Das Experiment wurde über einen Zeitraum von 2 Monaten durchgeführt. Um die Individuen auszuwerten wurden 197.8 Stunden benötigt. Als Umgebung stand ein abgesperrter Bereich eines Korridors zu Verfügung. Aufgrund der langen Dauer des Experimentes mußten die Batterien selbstverständlich von Zeit zu Zeit gewechselt werden. Anfangs wurden die Batterien gewechselt, nachdem eine Generation abgeschlossen war, später wurden die Batterien auch innerhalb einer Generation gewechselt. Dazu wurde die Evolution jeweils kurzzeitig angehalten.

Ein Individuum mit 457 Knoten erreichte die beste Fitneß in der 45. Generation. Das Individuum wurde schließlich 10 Mal unter den gleichen Bedingungen getestet. Dabei absolvierte es 8 der 10 Tests erfolgreich. Zweimal wurde der Lauf vorzeitig abgebrochen, weil der Roboter der Wand zu nahe kam. Die Wand wurde dabei nicht berührt. Die Ergebnisse dieser Experimente sind in Abbildung 4.12 zu sehen. Der Pfad des Roboters wurde mit Hilfe der Odometrie des Roboters aufgezeichnet. Zusätzlich wurde in einem weiteren Experiment der Pfad sowohl über die Odometrie als auch mit einer Fotokamera aufgenommen. Dazu wurde eine kleine Lampe am Roboter angebracht. Das Foto mit dem zurückgelegten Weg ist ebenfalls in Abbildung 4.12 zu sehen. Es wurde mit Langzeitbelichtung aufgenommen.

Die Fitneß-Statistik des Experimentes, das mit dem Service-Roboter durchgeführt wurde, ist in Abbildung 4.13 zu sehen. Die Individuen der ersten Generation bewegten sich entweder im Kreis oder stießen sehr bald gegen eine Wand. Im Laufe der Evolution verbesserten sich die Individuen. Daher wurde wesentlich mehr Zeit für die Auswertung der Individuen der späteren Generationen benötigt, als am Anfang der Evolution. Die durchschnittliche Zeit, die zur Auswertung der Individuen benötigt wurde, stieg von 12,9s in der ersten Generation auf 196,7s in der 40. Generation an. Die durchschnittliche Zeit, die zur Auswertung der

Abbildung 4.12: Mit der besten evolvierten Kontrollarchitektur wurden 10 Tests durchgeführt. Die Ergebnisse dieser Tests sind auf der linken Seite zu sehen. Das Individuum absolvierte 8 der 10 Tests erfolgreich. Der Pfad wurde jeweils mit Hilfe der Odometrie des Roboters aufgezeichnet. Zusätzlich wurde in einem weiteren Test eine kleine Lampe an dem Roboter angebracht. Das Foto mit Langzeitbelichtung auf der rechten Seite wurde zur gleichen Zeit wie der Pfad oben rechts aufgenommen.

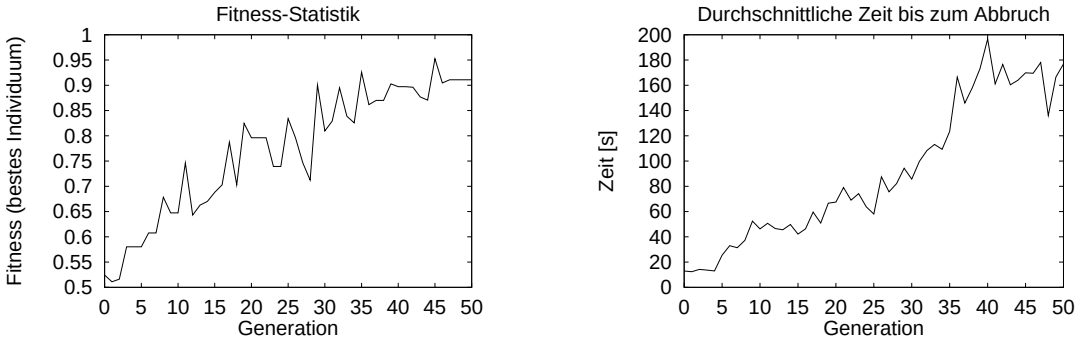


Abbildung 4.13: Fitneß-Statistik für das Experiment, das mit dem Service-Roboter durchgeführt wurde. Auf der rechten Seite ist die durchschnittliche Zeit bis zum Abbruch der Auswertung eines Individuums gezeigt.

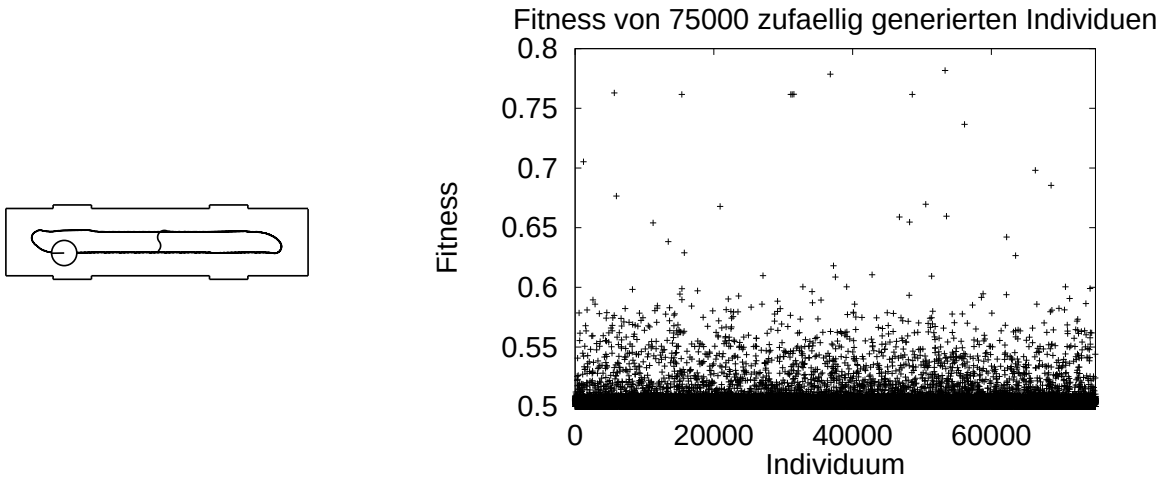


Abbildung 4.14: Fitneß von 75000 zufällig generierten Individuen. Das beste Individuum, das bei dieser zufälligen Suche im Raum der möglichen Kontrollarchitekturen gefunden wurde, ist links dargestellt.

Individuen benötigt wurde, ist ebenfalls in Abbildung 4.13 zu sehen.

4.5 Zufällige Suche nach geeigneten Individuen

In 50 Generationen werden bei einer Populationsgröße von 75 Individuen insgesamt 3750 Individuen ausgewertet. Um die Ergebnisse der Evolution mit einer zufälligen Suche zu vergleichen, wurden auch 3750 zufällig erzeugte Individuen ausgewertet. Die Individuen wurden mit der ansteigenden 50:50 Regel generiert. Es wurden Individuen von einer minimalen Tiefe von 2 bis zu einer maximalen Tiefe von 6 generiert. Die Fitneß eines Individuums wurde aus dem Mittelwert von 3 Tests berechnet. Zwischen jeder Auswertung wurde jedes Individuum repositioniert, indem es vom nächsten Hindernis abgestoßen wurde. Bei dieser zufälligen Suche im Raum der möglichen Kontrollarchitekturen wurde kein erfolgreiches Individuum gefunden. Die Suche wurde mit unterschiedlichen Initialisierungen des Zufallszahlengenerators fortgesetzt. So wurden insgesamt 75000 Individuen zufällig generiert. Das Verhalten des besten Individuums, das bei dieser Suche gefunden wurde, ist in Abbildung 4.14 zu sehen. Die Fitneßwerte der 75000 Individuen sind ebenfalls in Abbildung 4.14 gezeigt. Die zufällige Suche zeigte, daß auch auf diese Weise erfolgreiche Individuen gefunden werden können. Genetisches Programmieren produzierte jedoch Individuen mit höherer Fitneß.

4.6 Verwandte Arbeiten

Im folgenden werden verwandte Arbeiten diskutiert, die mit Hilfe von genetischem Programmieren Kontrollarchitekturen für simulierte oder reale Roboter evolvierten.

Koza [167] evolvierte mit Hilfe von Computer-Simulationen ein Programm, um einen Roboter zu steuern, der ein 8×8 großes Feld fegen sollte. Dabei konnten einige der Felder durch ein Hindernis belegt sein. Der Roboter konnte nur von einem Feld zum nächsten

"springen". Das relativ einfache und abstrakte Problem hatte die Aufgabe, zu demonstrieren, daß sogenannte automatisch definierte Funktionen [167] nützlich sein können.

In einem realistischeren Szenario evolvierte Koza [165] eine Subsumption-Kontrollarchitektur [33], die ein Folge-der-Wand-Verhalten für einen mobilen Roboter realisiert. Die Experimente wurden wieder nur in der Simulation und mit diskreten Bewegungen durchgeführt. In seiner Repräsentation verwendete Koza den bedingten Befehl `IFLTE` und die Verbindungsfunktion `PROGN2` als elementare Funktionen. Der Befehl `IFLTE` ist eine Funktion mit vier Argumenten. Falls das erste Argument kleiner als oder gleich groß ist wie das zweite Argument, dann wird das dritte Argument ausgewertet, andernfalls wird das vierte Argument ausgewertet. `PROGN2` ist eine Funktion mit zwei Argumenten. Die Argumente werden der Reihe nach ausgeführt, und der Wert des zweiten Argumentes wird zurückgegeben. Der Roboter konnte seine Umgebung mit Hilfe von 12 Sonar-Sensoren wahrnehmen. Daher bestand die Menge der Terminal-Symbole aus `S00` ... `S11`, die den jeweiligen Wert des Sensors zurückgaben und `SS`, das das Minimum aller Sonar-Sensoren zurückgab. Ferner wurden die Konstanten `MSD`, der kleinste noch sichere Abstand zur Wand, und `EDG`, der gewünschte Abstand zur Wand, verwendet. Der Roboter wurde über die Terminal-Symbole `(MF)`, `(MB)`, `(TL)` und `(TR)` gesteuert. Dabei bewegte `(MF)` den Roboter 30cm vorwärts, `(MB)` bewegte den Roboter 40cm rückwärts, `(TL)` drehte den Roboter um 30 Grad nach links und `(TR)` drehte den Roboter um 30 Grad nach rechts. Die Individuen wurden jeweils mit der gleichen Anfangsposition ausgewertet. Dies ist nur in der Simulation möglich und kann mit einem echten Roboter, der nur über eine begrenzte Positionierungsgenauigkeit verfügt, nicht reproduziert werden. Aus der Zahl der Kacheln entlang der Wand, über die sich der Roboter hinwegbewegte, wurde die Fitneß berechnet. Von Ross et al. [255] wurde vorgeschlagen, die Aufgabe, ein Folge-der-Wand-Verhalten zu evolvieren, als sogenanntes *Benchmark*-Problem zur Erzeugung von Verhaltensmustern in der Robotik zu verwenden. Außerdem untersuchten Ross et al. den Suchraum und stellten fest, daß die gefundenen Individuen den bedingten Befehl und die verbindende Funktion häufig, die Symbole, um den Roboter zu steuern, selten und nur einen Teil der Sensoren verwendeten.

Reynolds [253] evolvierte ein Verhalten zur Hindernisvermeidung für einen simulierten Roboter. Die Umgebung wurde mittels eines Sensors wahrgenommen, der in eine beliebige Richtung relativ zur Orientierung des Roboters gerichtet werden konnte. Der Sensor wurde über die Funktion `look-for-obstacle` gesteuert, deren Argument die Richtung des Sensors angab. Der Rückgabewert war ein Maß für den Abstand zum Hindernis. Bei jedem Schritt bewegte sich der Roboter um die Hälfte seiner Körperlänge vorwärts. Die Auswertung des Individuums wurde abgebrochen, sobald der Roboter ein Hindernis berührte. Zur Steuerung des Roboters konnte die Funktion `turn` verwendet werden. Das Argument der Funktion gab den Winkel an, um den sich der Roboter dreht. Als weitere Funktionen wurden die bedingte Funktion `iflte`, die Betragsfunktion `abs`, und die arithmetischen Funktionen `+`, `-`, `*` und `%` (sichere Division) verwendet. Ferner standen die Konstanten `0`, `0.01`, `0.1`, `0.5` und `2` zur Verfügung. In weiteren Experimenten setzte Reynolds Rauschen ein, um robuste Lösungen zu erhalten [251, 252]. Zudem untersuchte Reynolds die Auswirkungen der gewählten Repräsentation auf die Schwierigkeit des Problems [250]. Reynolds experimentierte mit einer Repräsentation, bei der der Sensor dynamisch ausgerichtet werden konnte, und mit einer Repräsentation, bei der die Sensoren fest am Roboter angebracht waren. Die Repräsentation mit fester Sensorik vereinfachte das Problem deutlich.

Nordin und Banzhaf [220, 221, 222, 223, 224] setzten als erste genetisches Programmieren ein, um eine Kontrollarchitektur für einen Miniatur-Roboter, den Khepera, zu evolvieren. Die

Umgebung des Roboters wurde über 8 Infrarot-Sensoren wahrgenommen. Nordin und Banzhaf verwendeten einen linearen Genotyp, der aus einer Reihe von Maschinencode-Befehlen bestand. Die genetischen Operatoren manipulierten daher direkt die Maschinencode-Sequenzen. Nordin und Banzhaf verwendeten die elementaren Funktionen `ADD`, `SUB`, `MUL`, `SHL`, `SHR`, `XOR`, `OR` und `AND`. Dabei sind `SHL` und `SHR` zwei Verschiebeoperationen. Die anderen Funktionen führen die üblichen arithmetischen und logischen Operationen aus. Neben diesen Funktionen wurden außerdem ganze Zahlen aus dem Bereich 0-8192 verwendet. Nordin und Banzhaf evolvierten so erfolgreich ein Verhalten zur Hindernisvermeidung. Olmer et al. [231] evolvierten mit der gleichen Repräsentation Module für einzelne Verhalten wie z.B. Geradeausfahrt oder Hindernisvermeidung. Danach wurde auf einer höheren Stufe ein Selektionsmechanismus evolviert, der zwischen den einzelnen Verhaltensstrategien auswählt. Eine Übersicht über die Arbeiten wird von Banzhaf et al. [17] gegeben. Nordin et al. [225] evolvierten auch ein Folge-der-Wand Verhalten für den Khepera. Dabei verwendeten sie die elementaren Funktionen `ADD`, `SUB`, `MUL` und die beiden Sprungbefehle `BG` (*Branch on Greater*) und `BLE` (*Branch on Less or Equal*). Es stellte sich heraus, daß zur Evolution des Folge-der-Wand-Verhaltens die Sprungbefehle vorhanden sein mußten.

Wilson et al. [320] evolvierten hierarchische Verhalten für einen mobilen Roboter. Zunächst wurde eine Population zufälliger Individuen erzeugt. Jedes Individuum bestand aus einer Sequenz elementarer Verhalten. Dabei verwendeten Wilson et al. die folgende Menge elementarer Verhalten: `MoveForward`, `MoveBackward`, `TurnLeft` und `TurnRight`. Die einzelnen Individuen wurden zuerst in der Simulation getestet. Das beste Individuum dieser Stufe wurde verwendet, um zusätzliche Individuen zu züchten, die dann schließlich auf dem echten Roboter getestet wurden. Aus den Sequenzen der Individuen mit der höchsten Fitneß wurden Programmblöcke gebildet, die dann in der nächsten Iteration des Verfahrens als elementare Bausteine für neue Sequenzen zur Verfügung standen. Sensorinformationen wurden nicht verarbeitet.

Lee et al. [175] evolvierten eine verhaltensbasierte Kontrollarchitektur in der Simulation. Die evolvierten Architekturen wurden anschließend auf einem Khepera-Roboter getestet. Die von Lee et al. evolvierte Kontrollarchitektur bestand aus elementaren Verhaltensmustern und einem Selektionsmechanismus, der zwischen den Verhaltensmustern das jeweils geeignete auswählt. Der Roboter hatte die Aufgabe, eine Box auf ein Licht hinzuschieben. Als elementare Funktionen verwendeten Lee et al. die logischen Funktionen `AND`, `OR`, `NOT`, `XOR` sowie eine Vergleichsoperation `>=`. Als Terminal-Symbole wurden die Infrarot- und Licht-Sensoren des Khepera sowie einige Konstanten definiert.

Der hier vorgestellte Ansatz hebt sich von den zuvor genannten dadurch ab, daß eine hierarchische, verhaltensbasierte Kontrollarchitektur für einen mobilen Service-Roboter evolviert wird. Koza und Reynolds verwendeten in ihren Experimenten lediglich Computer-Simulationen. Banzhaf et al. verwendeten einen Miniatur-Roboter und eine andere Form von genetischem Programmieren. Sie evolvierten lineare Sequenzen aus Maschinencode. Wilson et al. verwendeten keinen bedingten Befehl. Bei den hier durchgeführten Experimenten wurden erstmals Sonar-Sensoren eingesetzt, um eine Kontrollarchitektur für einen Service-Roboter zu evolvieren. Ein Service-Roboter unterscheidet sich in vieler Hinsicht von einem Miniatur-Roboter. Experimente mit einem Miniatur-Roboter können auch dann fortgesetzt werden, wenn der Roboter gegen eine Wand fährt. Der Roboter kommt entweder zum Stillstand oder er rutscht einfach an der Wand entlang. Ein Service-Roboter kann dagegen große Kräfte erzeugen und somit großen Schaden verursachen, wenn er gegen ein Hindernis fährt. Daher müssen spezielle Sicherheitsmaßnahmen bei der Durchführung der Experimente getroffen

werden.

4.7 Zusammenfassung

Mit genetischem Programmieren wurde eine verhaltensbasierte Kontrollarchitektur für einen Service-Roboter evolviert. Der Roboter nahm seine Umgebung über Ultraschall-Sensoren wahr. Er bewegte sich mit konstanter Geschwindigkeit vorwärts. Der Kontrollalgorithmus mußte die Drehgeschwindigkeit des Roboters steuern, um Hindernisse zu vermeiden.

Zunächst wurde mit Hilfe der Simulation nach einer geeigneten Repräsentation gesucht. Die Simulation erfolgte im Zeitraffer, dadurch konnten relativ viele Experimente durchgeführt werden. Nach den Experimenten in der Simulation wurde eines der Experimente mit dem Service-Roboter wiederholt. Das Experiment mit dem Service-Roboter wurde über einen Zeitraum von zwei Monaten durchgeführt. Das Experiment zeigte, daß es möglich ist, eine Kontrollarchitektur vollständig auf dem Service-Roboter zu evolvieren.

Kapitel 5

Evolution visueller Merkmalsdetektoren

Evolutionäre Algorithmen eignen sich auch zur Generierung visueller Merkmalsdetektoren. So kann ein für ein ganz bestimmtes Problem optimaler Merkmalsdetektor erzeugt werden. Die Evolution visueller Merkmalsdetektoren soll hier an drei Beispielen gezeigt werden. Zunächst wird ein Kantendetektor evolviert, der den Canny-Operator [141] approximiert. Dann wird ein Operator zur Extraktion markanter Punkte evolviert. Hierbei wird zunächst versucht, die Ausgabe eines bereits existierenden Operators, des Moravec-Interest-Operators [205, 206], zu approximieren. Danach wird ein Operator evolviert, der sich zur Berechnung des optischen Flusses eignet. Während bei den ersten Experimenten versucht wird, die Ausgabe eines existierenden Operators zu approximieren, wird im dritten Beispiel gezeigt, wie ein Operator evolviert werden kann, für den nur noch die gewünschten Eigenschaften vorgegeben werden. Als Problemstellung wird die Berechnung des optischen Flusses herangezogen. Bei allen drei Beispielen wurde genetisches Programmieren eingesetzt, da mit genetischem Programmieren hierarchische Strukturen evolviert werden können. Solche Strukturen werden oft benötigt, um Probleme der Bildverarbeitung zu lösen. Zur Bildverarbeitung wurde Vista [239] eingesetzt, da es leicht um eigene Routinen erweitert werden kann. Eine Übersicht über gängige Operatoren der Bildverarbeitung wird durch die verschiedenen Lehrbücher zu diesem Thema gegeben [193, 128, 113, 241, 141, 190].

5.1 Existierende Operatoren zur Extraktion von markanten Punkten

Es existieren bereits eine Vielzahl unterschiedlicher Operatoren zur Extraktion markanter Punkte. Im folgenden werden einige der bekannteren Operatoren kurz besprochen. Dabei bezeichnet $I(x, y)$ die Intensität des Bildpunktes mit den Koordinaten x und y. Am Ende dieses Kapitels wird für die Operatoren, die sich zur Berechnung des optischen Flusses eignen, dasselbe Qualitätsmaß, das auch zur Bewertung der evolvierten Individuen eingesetzt wurde, berechnet. Dadurch wird ein Vergleich der Qualität der evolvierten Operatoren mit der Qualität der existierenden Operatoren möglich.

5.1.1 Kantendetektoren

Kanten lassen sich in einem Bild durch Berechnung der Stärke des Gradienten G berechnen [193, 128, 20, 113, 241, 141, 190].

$$G = |\nabla I| = \sqrt{I_x^2 + I_y^2} \tag{5.1}$$

Dabei sei $I_x = \frac{\partial}{\partial x} I$ die partielle Ableitung in x-Richtung und $I_y = \frac{\partial}{\partial y} I$ die partielle Ableitung in y-Richtung. Die Richtung des Gradienten ist durch

$$\tan\theta = \frac{I_x}{I_y} \tag{5.2}$$

gegeben. Je nach Art der Approximation des Gradienten ergeben sich unterschiedliche Kantendetektoren, wie z.B. der Sobel-Operator, der durch die Matrizen

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \text{und} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5.3}$$

definiert ist, mit denen jeweils vertikale und horizontale Kanten detektiert werden. Kanten lassen sich entweder durch Lokalisation von Maxima der ersten oder durch Lokalisation von Nulldurchgängen der zweiten Ableitung extrahieren. Die zweite Ableitung in Richtung des Gradienten ist [141, 190]:

$$\frac{\partial}{\partial\theta} G = G_\theta = \nabla G \frac{\nabla I}{|\nabla I|} = \frac{I_x^2 I_{xx} + 2 I_x I_y I_{xy} + I_y^2 I_{yy}}{I_x^2 + I_y^2} \tag{5.4}$$

wobei

$$I_x^2 I_{xx} + 2 I_x I_y I_{xy} + I_y^2 I_{yy} = [I_x, I_y] \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} = [I_x, I_y] H_I \begin{bmatrix} I_x \\ I_y \end{bmatrix} \tag{5.5}$$

Die Matrix H_I wird als Hesse-Matrix bezeichnet. Ein weiterer Operator zur Extraktion von Kanten ist der Laplace-Operator.

$$\nabla^2 I = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) I \tag{5.6}$$

Da die zweite Ableitung sehr stark vom Rauschen abhängt, das im Bild vorhanden ist, wird in der Praxis erst ein Gaußscher Filter auf das Bild angewandt und anschließend die Kanten extrahiert. Diese Kombination von Laplace-Operator und Gaußscher Glättung wird Laplace-Gauß-Operator genannt [241]. Der Laplace-Gauß-Operator kann als Differenz zweier Gauß-Operatoren mit unterschiedlicher Breite approximiert werden [193, 190].

Ein Operator zur Kantenextraktion, der optimal in Bezug auf Signal-Rauschverhältnis und Lokalisationsgenauigkeit ist, wurde von Canny entwickelt [141]. Als erstes wird das Bild mit einem Gauß-Operator geglättet und anschließend die Stärke des Gradienten und die Gradientenrichtung bestimmt. Danach werden Punkte, die kein lokales Maximum entlang der Gradientenrichtung sind, unterdrückt. Zum Schluß werden zwei unterschiedliche Schwellwerte eingesetzt, um Lücken in Kantensegmenten zu schließen.

5.1.2 Orientierte Kanten

Mit Gabor-Filtern lassen sich Kanten mit bestimmter Orientierung extrahieren [65, 241, 170, 192, 190]. Ein Gabor-Filter mit der Richtung θ, der Ausdehnung σ_x, σ_y und der Frequenz ω ist wie folgt definiert.

$$\Psi(x,y,\theta) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x'^2}{\sigma_x}+\frac{y'^2}{\sigma_y})+2\pi i\omega x'} \tag{5.7}$$

$$x' = x\cos\theta + y\sin\theta \tag{5.8}$$

$$y' = -x\sin\theta + y\cos\theta \tag{5.9}$$

Der Filter setzt sich aus einer Gaußschen Glättung und einem Phasenfaktor zusammen. Die Funktion Ψ ist komplex, daher erhält man als Antwort des Operators ebenfalls eine komplexe Funktion. Orientierte Kanten können durch die Berechnung des Betrages der Antwort extrahiert werden.

$$M(x,y,\theta) = |\int \Psi(x',y',\theta) I(x-x',y-y') dx'dy'| \tag{5.10}$$

Der Gabor-Filter wurde von Granlund [114] als allgemeingültiger Operator für die Bildverarbeitung zur Strukturanalyse vorgeschlagen. Er ergibt sich als Filter, der die Unschärferelation der Bildverarbeitung, d.h. die Relation zwischen den dualen Räumen des Ortes und der Ortsfrequenz, optimal erfüllt [241].

5.1.3 Generische Nachbarschaftsoperatoren

Koenderink und van Doorn [160, 161] entwickelten generische Nachbarschaftsoperatoren für die Bildverarbeitung. Sie fordern, daß durch eine Verringerung der Auflösung keine falschen Details eingeführt werden, und daß die Operatoren bei einer Änderung der Auflösung ähnlich zu sich selbst bleiben sollten. Dies erreichen sie durch die Forderung, daß die Operatoren die Diffusionsgleichung

$$\Delta I(x,y,s) = \frac{\partial I(x,y,s)}{\partial s} \tag{5.11}$$

erfüllen. Dabei ist das Bild I durch die Koordinaten x und y und die Auflösung bzw. Skalierung s definiert. Koenderink und van Doorn separieren den Operator in eine Fensterfunktion (eine Gaußsche Glättung), die von der Skalierung abhängt und einen Anteil, der bei einer Skalierung seine Form nicht verändert. Unter der Annahme, daß die Operatoren die Diffusionsgleichung erfüllen, erhalten sie im zweidimensionalen Fall die folgenden Bildoperatoren.

$$C_{pq}(x,y,s) = \left(\frac{1}{\sqrt{4s}}\right)^{p+q} \frac{H_p(\frac{x}{\sqrt{4s}}) H_q(\frac{x}{\sqrt{4s}}) e^{-\frac{x^2+y^2}{4s}}}{8\pi s\sqrt{2^{p+q}p!q!\pi}} \tag{5.12}$$

Dabei ist $H_n(\xi)$ das Hermitesche Polynom und p, q geben die Ordnung in x- bzw. y-Richtung an. Die n-te Ableitung in θ-Richtung hat dann die Form.

$$W_n(r,\theta,s) = C_{n0}(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta, s) \tag{5.13}$$

In Radialkoordinaten sind die Operatoren durch die folgende Gleichung definiert,

$$P_{km}(r,\theta,s) = \left(\frac{1}{\sqrt{4s}}\right)^{m+2k} \frac{1}{\sqrt{\pi}} \Lambda_k^m(\frac{r^2}{4s}) e^{\pm im\theta} \frac{e^{-\frac{r^2}{8s}}}{8\pi s} \tag{5.14}$$

wobei

$$\Lambda_k^a(t) = \left[\Gamma(a+1)\binom{k+a}{k}\right]^{-\frac{1}{2}} e^{-\frac{t}{2}} t^{-\frac{a}{2}} L_k^a(t) \tag{5.15}$$

und $\Gamma(x)$ die Gammafunktion und $L_k^a(t)$ das zugehörige Laguerresche Polynom ist. Die Ordnung in radialer Richtung bzw. in Richtung der Winkelkoordinate ist durch k bzw. m gegeben. Auf diese Weise definieren Koenderink und van Doorn eine vollständige und orthonormale Familie von Nachbarschaftsoperatoren mit unterschiedlichen Symmetrien, wie z.B. Translations-, Rotations-, und Skalierungssymmetrie, mit der es möglich wird, die Strukturen rezeptiver Felder zu klassifizieren.

5.1.4 Eckenmerkmale und gekrümmte Kanten

Eine weitere Methode, markante Punkte zu extrahieren, ist die Extraktion von Eckenmerkmalen. Shah und Jain [274] entwickelten einen Ansatz zur Extraktion bewegter Eckenmerkmale. Sie zeigten, daß die drei Merkmalsdetektoren, der Detektor von Zuniga und Haralick, der Detektor von Kitchen und Rosenfeld und der Detektor von Dreschler und Nagel äquivalent sind. Diese Detektoren betrachten die Änderung der Gradientenrichtung θ.

$$\nabla\theta = \frac{1}{I_x^2 + I_y^2} \begin{bmatrix} I_{xx}I_y - I_x I_{xy} \\ I_{yx}I_y - I_x I_{yy} \end{bmatrix} \tag{5.16}$$

Projiziert man die Änderung der Gradientenrichtung auf einen Vektor orthogonal zur Kantenrichtung $(-I_y, I_x)$, so erhält man ein Maß, das sich zur Extraktion von Ecken aus einem Kantenbild eignet. Dies ist der sogenannte Zuniga-Haralick-Eckendetektor [274].

$$M = \left| \frac{-I_y^2 I_{xx} + 2I_x I_y I_{xy} - I_x^2 I_{yy}}{(I_x^2 + I_y^2)^{1.5}} \right| \tag{5.17}$$

Anschließend können Punkte, bei denen es sich nicht um ein lokales Maximum handelt, unterdrückt werden. Beim Kitchen-Rosenfeld-Eckendetektor wird das Maß mit der Größe des Gradienten multipliziert. Daher werden Ecken aus einem Grauwertbild nur dann extrahiert, wenn an dieser Stelle auch eine Kante vorhanden ist. Bewegte Eckenmerkmale werden von Shah und Jain [274] extrahiert, indem sie zunächst die Ecken für ein einzelnes Bild extrahieren und diese dann mit Bildänderungen verknüpfen:

$$M_{\text{bewegt}} = M \cdot \frac{\partial}{\partial t} I \tag{5.18}$$

Daher wird ein Bildpunkt dann extrahiert, wenn es sich um einen Eckpunkt handelt und eine Veränderung an diesem Bildpunkt stattgefunden hat.

Die Determinante der Hesse-Matrix kann ebenfalls zur Extraktion gekrümmter Kanten eingesetzt werden [274, 190].

$$M = \text{Det}(H_I) = I_{xx}I_{yy} - I_{xy}^2 \tag{5.19}$$

Die Determinante entspricht dem Zähler der Gaußschen Krümmung K [31, 141].

$$K = \frac{I_{xx} I_{yy} - I_{xy}^2}{(1 + I_x^2 + I_y^2)^2} \tag{5.20}$$

Der Operator extrahiert die Punkte auf beiden Seiten der Ecke, d.h. innen und außen. Daher suchen Dreschler und Nagel jeweils korrespondierende Punkte maximaler und minimaler Gaußscher Krümmung und lokalisieren den Eckpunkt an der Stelle des Nulldurchgangs der Gaußschen Krümmung. Außer den hier beschriebenen Methoden können auch neuronale Netze zur Extraktion von Eckenmerkmalen eingesetzt werden. Büker und Hartmann [37] setzten neuronal detektierte Eckenmerkmale zur Objekterkennung und Fovealisierung ein.

5.1.5 Moravec-Interest-Operator

Moravec [205, 206, 141] entwickelte einen Operator zur Extraktion markanter Punkte, basierend auf der Differenz benachbarter Punkte in vier Richtungen: horizontal, vertikal und in den beiden Diagonalen. Als erstes wird die Summe der quadrierten Differenzen benachbarter Pixel für die vier Richtungen berechnet und dann das Minimum der vier Richtungen zurückgeliefert.

$$\begin{aligned} M(x,y) = \min\{ & \\ & \sum_{x-2 \le x' < x+2} \sum_{y-2 \le y' < y+2} \big(I(x',y') - I(x'+1,y')\big)^2, \\ & \sum_{x-2 \le x' < x+2} \sum_{y-2 \le y' < y+2} \big(I(x',y') - I(x',y'+1)\big)^2, \\ & \sum_{x-2 \le x' < x+2} \sum_{y-2 \le y' < y+2} \big(I(x'+1,y') - I(x',y'+1)\big)^2, \\ & \sum_{x-2 \le x' < x+2} \sum_{y-2 \le y' < y+2} \big(I(x',y') - I(x'+1,y'+1)\big)^2 \} \end{aligned} \tag{5.21}$$

Danach werden Punkte, die kein lokales Maximum sind, unterdrückt und schließlich wird ein Schwellwert angewandt. Durch die Verwendung der Minimum-Operation werden Punkte extrahiert, die eine hohe Differenz benachbarter Pixelwerte in allen vier Richtungen aufweisen. Daher werden keine horizontalen, vertikalen oder diagonalen Kanten extrahiert. Allerdings spricht der Operator auf Kanten an, die sich zwischen diesen vier Richtungen befinden. Der Operator ist sehr einfach zu implementieren, schnell zu berechnen und liefert vor allem in strukturierter Umgebung, wie z.B. einer Büroumgebung, gute Ergebnisse.

Harris und Stephens [119] gingen auf die Probleme des Moravec-Operators, wie z.B. seine Anisotropie und die Verwendung eines rechteckigen Fensters, ein und verallgemeinerten den Operator. Betrachtet man beliebige Verschiebungen (dx, dy), dann ist der Moravec-Operator wie folgt definiert.

$$M(x,y) = \min_{dx,dy} E(x,y,dx,dy) \tag{5.22}$$

Dabei ist

$$E(x,y,dx,dy) = \sum_{u,v} w_{u,v} \left(I(x+u, y+v) - I(x+u+dx, y+v+dy)\right)^2 \tag{5.23}$$

und $w_{u,v}$ definiert die Ausmaße des Fensters. Mit einer Taylor-Entwicklung, wobei die Terme höherer Ordnung vernachlässigt werden, ergibt sich

$$E(x,y,dx,dy) \approx \sum_{u,v} w_{u,v} \left(dxI_x(x+u,y+v) + dyI_y(x+u,y+v)\right)^2 \tag{5.24}$$

$$= \sum_{u,v} w_{u,v} \left([dx,dy] \begin{bmatrix} I_x^2(x+u,y+v) & I_xI_y(x+u,y+v) \\ I_xI_y(x+u,y+v) & I_y^2(x+u,y+v) \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \right) \tag{5.25}$$

Verwendet man als Fenster eine Gaußfunktion, so ergibt sich als Verallgemeinerung des Moravec-Operators der folgende Ausdruck.

$$M(x,y) =$$

$$= \min_{dx,dy} \int e^{-\frac{u^2+v^2}{2\sigma^2}} \left([dx,dy] \begin{bmatrix} I_x^2(x+u,y+v) & I_xI_y(x+u,y+v) \\ I_xI_y(x+u,y+v) & I_y^2(x+u,y+v) \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \right) dudv \tag{5.26}$$

$$= \min_{dx,dy} \left([dx,dy] \begin{bmatrix} (I_x^2)*G & (I_xI_y)*G \\ (I_xI_y)*G & (I_y^2)*G \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \right) \tag{5.27}$$

$$= \min_{dx,dy} \left([dx,dy] M_A \begin{bmatrix} dx \\ dy \end{bmatrix} \right) \tag{5.28}$$

Dabei ist $I*G$ die Faltung des Bildes I mit einem Gauß-Operator. Die Matrix M_A beschreibt die Form der Autokorrelationsfunktion im Ursprung. Die Eigenwerte dieser Matrix können ebenfalls zur Merkmalsdetektion eingesetzt werden. Falls beide Eigenwerte groß sind, hat die Autokorrelationsfunktion an dieser Stelle ein lokales Maximum, und es liegt ein Eckpunkt vor. In der Literatur lassen sich mehrere Versionen dieses Merkmalsdetektors finden: $M = Det(M_A) - \alpha(Spur(M_A))^2$ [119] wobei α eine Konstante ist oder $M = \frac{Det(M_A)}{(Spur(M_A))^2}$ [118] oder $M = \frac{Det(M_A)}{(Spur(M_A))}$ [56].

5.1.6 SUSAN-Merkmalsoperator

Smith [280] entwickelte einen Operator zur Extraktion von Eckenmerkmalen, der nicht auf der Ableitung der Bildintensitäten basiert. Stattdessen wird der betrachtete Bildausschnitt in zwei Regionen unterteilt. Die erste Region enthält den Punkt im Zentrum des Ausschnittes sowie alle Punkte, die eine ähnliche Helligkeit wie der Punkt im Zentrum besitzen. Alle anderen Punkte gehören zur zweiten Region. Ist die Fläche der ersten Region kleiner als die Hälfte der Fläche des Bildausschnittes, dann handelt es sich um eine konvexe Kante. Ist die Fläche der ersten Region minimal, dann hat man den Eckpunkt gefunden. Abbildung 5.1 stellt einige Beispiele für mögliche Segmentierungen dar. Die erste Region wird *Univalue Segment Assimilating Nucleus* (USAN), d.h. uniforme Region, zu der der Mittelpunkt des Ausschnittes gehört, genannt. Da nach der kleinsten Fläche dieser Region gesucht wird, wird der Operator von Smith als SUSAN-Operator *Smallest Univalue Segment Assimilating Nucleus* bezeichnet.

Zum Vergleich, ob ein Punkt eine ähnliche Helligkeit wie der Punkt im Zentrum des Operators besitzt, wird die folgende Funktion eingesetzt.

$$c(p,p_0) = e^{-\left(\frac{I(p)-I(p_0)}{t}\right)^6} \tag{5.29}$$

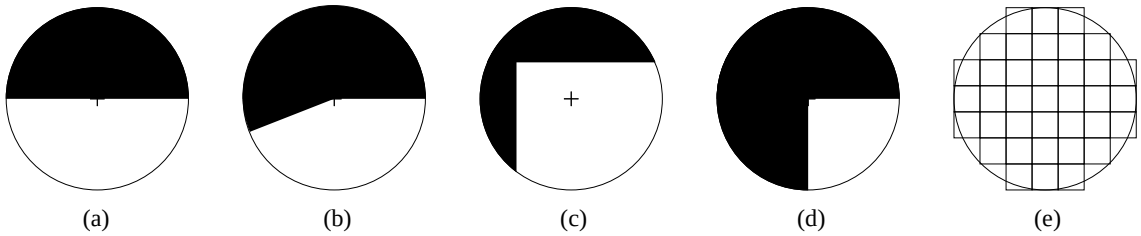


Abbildung 5.1: Gezeigt sind vier unterschiedliche Bildausschnitte, die in zwei Regionen eingeteilt wurden. Die Region, die das Zentrum des Bildausschnittes, sowie Punkte mit ähnlicher Helligkeit enthält, ist weiß gezeichnet. Eine konvexe Kante wird detektiert, wenn die weiße Region kleiner als die Hälfte der Gesamtfläche ist (siehe (a) und (b)). Ein Eckpunkt wird lokalisiert, wenn die weiße Fläche minimal ist (siehe (c) und (d)). (e) zeigt die diskrete Form des SUSAN-Operators, die sich in Experimenten von Smith als sehr geeignet erwiesen hat (nach Smith [280]).

Dabei ist p_0 der Punkt im Zentrum des Operators und p ein beliebiger Punkt innerhalb des Bildausschnittes. Über den Parameter t kann die Sensitivität des Vergleichs eingestellt werden. Je größer t ist, desto stärker dürfen die Intensitätswerte voneinander abweichen und werden trotzdem als ähnlich betrachtet. In den von Smith durchgeführten Experimenten wurden normalerweise $t = 25$ und $t = 7$ für Bilder mit geringem Kontrast verwendet. Um die Pixel der USAN-Region zu zählen, wird einfach über alle Bildpunkte summiert.

$$n = \sum_{p} c(p, p_0) \tag{5.30}$$

Extrahiert werden die Bildpunkte, für die $n < g$ gilt, und für die $M = g - n$ ein lokales Maximum ist. Als geometrischer Schwellwert g für die Größe der USAN-Region wird meist 18.5, d.h. die Hälfte der Pixel im Bildausschnitt verwendet. Über diesen Schwellwert kann die Art der extrahierten Merkmale reguliert werden. Geht man von Kantenmerkmalen aus, dann sind die extrahierten Ecken umso schärfer, je kleiner der Schwellwert ist.

5.1.7 Extraktion von markanten Punkten mit Gabor-Filtern

Manjunath et al. [192] entwickelten ein Verfahren zur Extraktion markanter Punkte, basierend auf Gabor-Filtern. Sie verwenden die folgende Definition des Gabor-Filters:

$$\Psi(x, y, \theta) = e^{-(x'^2 + y'^2) + i\pi x'} \tag{5.31}$$

$$x' = x\cos\theta + y\sin\theta \tag{5.32}$$

$$y' = -x\sin\theta + y\cos\theta \tag{5.33}$$

wobei θ die Orientierung des Filters angibt. Zur Extraktion der Bildmerkmale werden Gabor-Filter unterschiedlicher Auflösung eingesetzt:

$$W_j(x, y, \theta_k) = \int I(x', y')\Psi(\alpha^{-j}(x - x'), \alpha^{-j}(y - y'), \theta_k)dx'dy' \tag{5.34}$$

Über die Parameter $j \in \mathbb{N}$ und α wird der Gabor-Filter skaliert. In den von Manjunath et al. durchgeführten Experimenten wurde $\alpha = \sqrt{2}$ gesetzt. Die Richtung wird über $\theta_k = \frac{k\pi}{N}$

(wobei $k \in \{0, ..., N-1\}$) diskretisiert. Dabei gibt N die Zahl der verwendeten Richtungen an. Extrahiert werden die Punkte, für die das folgende Maß ein lokales Maximum ist.

$$M(x, y, \theta_k) = |W_i(x, y, \theta_k) - \alpha^{2(i-j)} W_j(x, y, \theta_k)| \tag{5.35}$$

Manjunath et al. verwendeten $i = 0$ und $j = 6$ bzw. $i = 2$ und $j = 5$ als Parameter zur Merkmalsextraktion. Extrahiert werden Änderungen der Kantenrichtung sowie Anfangs- und Endsegmente von Kanten.

Eine Variante dieses Detektors wurde von Zheng und Chellappa [330] zur automatischen Merkmalsextraktion und -verfolgung eingesetzt. Als markante Punkte extrahieren sie die Punkte, für die das folgende Maß ein lokales Maximum annimmt.

$$M(x, y) = \max_k |W_i(x, y, \theta_k) - \alpha^{2(i-j)} W_j(x, y, \theta_k)| \tag{5.36}$$

Extrahiert werden demnach die Punkte, an denen die lokale Krümmung ein lokales Maximum ist.

5.1.8 Detektion von Symmetrien

Reisfeld et al. [249] entwickelten einen Merkmalsdetektor, um symmetrische Punkte zu extrahieren. Dazu werden für jeden Punkt $p = (x, y)$ einander gegenüberliegende Punkte mit gleichem Abstand betrachtet und die Stärke und die Richtung der Gradienten miteinander verglichen. Die Symmetrie eines Punktes p in Richtung θ und einer Umgebung σ wird als

$$S_\sigma(p, \theta) = \sum_{(i,j) \in \Gamma(p,\theta)} D_\sigma(i, j) P(i, j) r_i r_j \tag{5.37}$$

definiert, wobei

$$\Gamma(p, \theta) = \left\{ (i, j) | \frac{p_i + p_j}{2} = p, \frac{\theta_i + \theta_j}{2} = \theta \right\} \tag{5.38}$$

$$D_\sigma(i, j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||p_i - p_j||}{2\sigma}} \tag{5.39}$$

$$P(i, j) = (1 - \cos(\theta_i + \theta_j - 2\alpha_{ij}))(1 - \cos(\theta_i - \theta_j)) \tag{5.40}$$

$$r_k = \log(1 + ||\nabla p_k||) \tag{5.41}$$

und θ_i bzw. θ_j die Richtung des Gradienten am Punkt p_i bzw. p_j angibt. Die Umgebung wird durch $D_\sigma(i, j)$ über den Parameter σ definiert. Mit $P(i, j)$ wird die Richtung der Gradienten verglichen. Dabei ist α_{ij} der Winkel zwischen der Geraden, die die Punkte p_i und p_j verbindet und der Gerade in x-Richtung. Das Maß $P(i, j)$ ist maximal, wenn die Summe der Gradientenrichtung relativ zur Verbindungsgeraden gleich π ist. Als Maß für die Stärke der Gradienten wird r_i bzw. r_j verwendet, die beide möglichst groß sein sollten.

Reisfeld et al. teilen die Richtungen in n Bereiche ein ($i \in \{1, ..., n\}$) und diskretisieren so das Symmetriemaß.

$$S_{n,\sigma}(p, i) = \int_{\theta \in \mathrm{R(i)}} S_\sigma(p, \theta) d\theta \tag{5.42}$$

$$R(i) = \bigcup_{k=0,1} \left[k\pi + \frac{(i-1)\pi}{n} - \frac{\pi}{2n}, k\pi + \frac{i\pi}{n} - \frac{\pi}{2n} \right) \tag{5.43}$$

Dadurch wird z.B. für $n = 1$ ein isotroper Symmetrie-Operator $S_{1,\sigma}(p, 1)$ definiert.

$$S_{1,\sigma}(p, 1) = \int_0^{2\pi} S_\sigma(p, \theta) d\theta \tag{5.44}$$

Für zwei diskrete Richtungen werden mit $S_{2,\sigma}(p, 1)$ Punkte detektiert, die eine Symmetrie in x-Richtung, und mit $S_{2,\sigma}(p, 2)$ Punkte, die eine Symmetrie in y-Richtung aufweisen, detektiert. Ein Maß für die Symmetrie in mehreren der diskreten Richtungen ist durch

$$CS_{n,\sigma}(p) = \prod_{i=1}^{n} (1 + S_{n,\sigma}(p, i)) \tag{5.45}$$

gegeben. Das Maß $CS_{8,\sigma}(p)$ wurde von Reisfeld et al. zur Detektion von Gesichtern eingesetzt. Mit $S_{8,\sigma}(p, 1)$ wurden Augen-, Mund- und Nasenregionen detektiert.

Zabrodsky et al. [326] definierten ein Maß für die Symmetrie einer Form, die durch eine Punktliste repräsentiert ist. Als Symmetrie definierten sie den Abstand zwischen den Punkten der Originalform und einer symmetrisierten Form. Mit diesem Maß lassen sich die Formen im Bezug auf unterschiedliche Symmetrieklassen, wie Spiegel- oder Rotationssymmetrie, hin untersuchen. Auch Zabrodsky et al. setzten das Symmetriemaß zur Lokalisation von Gesichtern in Bildern ein. Kalinke und von Seelen [152] setzten ein Hopfield-Netz ein, um vertikale Symmetrieachsen zu detektieren.

5.1.9 Statistische Methoden

Statistische Methoden werden oft zur Texturanalyse eingesetzt [141, 29]. Sie können ebenfalls zur Merkmalsextraktion herangezogen werden. Für eine gegebene Textur lassen sich Eigenschaften wie Entropie, Energie, Kontrast und Homogenität definieren. Zur Berechnung der Eigenschaften wird als erstes eine Wahrscheinlichkeitsverteilung P benachbarter Pixel, die sogenannte *co-occurrence* Matrix, berechnet. Dazu werden die in der Textur auftretenden Grauwerte diskretisiert. Die möglichen Grauwerte seien g_i mit $i \in \{1, ..., n\}$. Dann gibt der Eintrag $P[g_i, g_j]$ die relative Häufigkeit der Grauwerte g_i und g_j im Abstand $\mathbf{d}$ an. Als Abstandsvektor wird dabei z.B. $\mathbf{d} = (1, 1)$ oder $\mathbf{d} = (0, 1)$ verwendet. Die Eigenschaften Entropie, Energie, Kontrast und Homogenität berechnen sich aus der Matrix P wie folgt.

$$\text{Entropie} = -\sum_i \sum_j P[g_i, g_j] \log P[g_i, g_j] \tag{5.46}$$

$$\text{Energie} = \sum_i \sum_j P^2[g_i, g_j] \tag{5.47}$$

$$\text{Kontrast} = \sum_i \sum_j (g_i - g_j)^2 P[g_i, g_j] \tag{5.48}$$

$$\text{Homogenität} = \sum_i \sum_j \frac{P[g_i, g_j]}{1 + |g_i - g_j|} \tag{5.49}$$

Man beachte, daß es sich hierbei um einen Operator mit einer Vorzugsrichtung handelt, da für die Wahrscheinlichkeitsverteilung benachbarter Pixel die Richtung $\mathbf{d}$ verwendet wird. Nach Jain et al. [141] wird daher oft die Matrix für mehrere Richtungen berechnet und die Richtung zur Berechnung des jeweiligen Maßes verwendet, die es maximiert. Die Matrix kann auf eine

symmetrische Form durch Addition der transponierten Matrix gebracht werden. Dadurch wird sie unabhänig von der Reihenfolge der betrachteten Grauwerte [29]. Die Entropie kann auch wie folgt definiert werden [113].

$$\text{Entropie} = -\sum_i P(g_i) \log P(g_i) \tag{5.50}$$

Dabei gibt $P(g_i)$ die Wahrscheinlichkeit an, daß der Grauwert g_i im untersuchten Bildausschnitt vorkommt. Kalinke und von Seelen [151] setzten die Entropie zur Realisierung einer Aufmerksamkeitssteuerung ein. Durch die Berechnung der lokalen Entropie segmentierten sie Bilder in Bereiche, mit hohem bzw. niedrigem Informationsgehalt.

5.1.10 Geometrische Merkmale

Linien und Kreise können aus einem Bild mit Hilfe der Hough-Transformation [20, 113, 29, 141, 174, 190] extrahiert werden. Die Hough-Transformation basiert auf einer Transformation in einen diskretisierten Parameterraum, der die geometrischen Merkmale beschreibt. Das gesuchte Merkmal sei durch eine Funktion $g(x, y, \bar{a}) = 0$ beschrieben. Die Bildpunkte werden mit x und y bezeichnet und die Parameter des Merkmals seien $\bar{a}$. Dann ist die Hough-Transformation durch

$$H(\bar{a}) = \int\int e(x, y)\delta(g(x, y, \bar{a}))dxdy \tag{5.51}$$

definiert [174]. Dabei ist $\delta(x)$ die Delta-Funktion und $e(x, y)$ eine Funktion, die die Positionen von Teilmerkmalen im Bild definiert, also z.B. ein Kantenbild mit $e(x, y) = 1$ für die Bildpunkte, an denen sich eine Kante befindet. Bei der Hough-Transformation werden für jeden Punkt des Originalbildes die Punkte im Parameterraum markiert, die diesen Punkt erzeugt haben könnten. Die Maxima im Parameterraum stellen die Parameter der Merkmale im Originalbild dar.

5.1.11 Invariante Merkmale

Schulz-Mirbach [269] entwickelten eine Methode, um Merkmale invariant gegenüber Rotationen und Translationen zu machen. Es wird einfach über alle möglichen Transformationen gemittelt.

5.2 Evolution von Kantendetektoren

Beim Einsatz von evolutionären Algorithmen in der Bildverarbeitung stellt sich die Frage, ob es möglich ist, daß die Evolution eventuell benötigte Operatoren aus elementaren Operatoren selbst generieren kann. Dies wird am Beispiel der Evolution von Kantendetektion untersucht (siehe Ebner [76]). Hierbei wird wiederum genetisches Programmieren eingesetzt. Es wird versucht, einen Detektor zu evolvieren, dessen Ausgabe den Canny-Kantendetektor [141] approximiert. Als elementare Funktionen werden nur einfache, elementare Operatoren definiert. Ein Gauß-Operator oder ein Operator, der nicht-lokale Maxima unterdrückt, wird nicht bereitgestellt. Es werden mehrere Experimente mit unterschiedlichen elementaren Funktionen durchgeführt. Schließlich wird der beste evolvierte Kantendetektor eingehend beschrieben. Ferner werden zwei Detektoren näher untersucht, die sich darauf spezialisierten, vertikale Kanten zu detektieren.

5.2.1 Repräsentation

Im folgenden wird die Repräsentation beschrieben, die zur Evolution der Kantendetektoren eingesetzt wurde. Es werden zunächst die verwendeten Terminal-Symbole, die eingesetzten elementaren Funktionen und die Fitneßfunktion beschrieben.

Terminal-Symbole

Als Terminal-Symbol `Image` wurde das Eingabebild mit dem Wertebereich [0,1] verwendet. In einem Teil der Experimente wurden zudem Zufallszahlen $\mathcal{R}$ aus dem Bereich $[-1.0, 1.0)$ eingesetzt. Dabei beschreibt eine Zufallszahl das Bild, bei dem alle Pixel den Wert der Zufallszahl haben.

Weitere Terminal-Symbole wären ebenfalls denkbar. So könnten z.B. die Farbbänder Rot, Grün und Blau für den RGB-Farbraum oder die Bänder Farbe, Sättigung und Intensität für den HSI-Farbraum [113, 241, 141, 29, 190] eingesetzt werden. Die Evolution hätte in diesem Falle die Möglichkeit, die Farbräume bzw. Bänder zu selektieren, in denen die Merkmale besonders gut extrahiert werden können.

Elementare Funktionen

Die elementaren Funktionen operieren jeweils auf einem Bild bzw. auf zwei Bildern und erzeugen ein neues Bild. Es sei I das Eingabebild für unäre Funktionen und I_1 und I_2 die Eingabebilder für binäre Funktionen. Das Bild, das durch die Anwendung der elementaren Funktion entsteht sei I_R. Bildkoordinaten werden mit x und y bezeichnet. Die folgenden elementaren Funktionen wurden zur Evolution der Kantendetektoren verwendet.
Unäre Funktionen:

- Negation (`Neg`) negiert alle Pixelwerte.
  $I_R(x, y) = -I(x, y)$
- Betrag (`Abs`) berechnet für jedes Pixel den Betrag.
  $I_R(x, y) = |I(x, y)|$
- Shift nach links (`ShiftL`) verschiebt das Bild I um ein Pixel nach links.
  $I_R(x, y) = I(x + 1, y)$

- Shift nach rechts (`ShiftR`) verschiebt das Bild I um ein Pixel nach rechts.
  $I_R(x, y) = I(x - 1, y)$
- Shift nach oben (`ShiftU`) verschiebt das Bild I um ein Pixel nach oben.
  $I_R(x, y) = I(x, y + 1)$
- Shift nach unten (`ShiftD`) verschiebt das Bild I um ein Pixel nach unten.
  $I_R(x, y) = I(x, y - 1)$
- Schwellwert (`UThr`) setzt alle Pixel, die größer sind als Null auf Eins und sonst auf Null.
  $$I_R(x, y) = \begin{cases} 1 & I(x, y) > 0 \\ 0 & \text{sonst} \end{cases}$$

Binäre Funktionen:

- Schwellwert (`Thr`) setzt alle Pixel in I_1, die einen Wert größer als der korrespondierende Wert in I_2 haben, auf Eins und sonst auf Null.
  $$I_R(x, y) = \begin{cases} 1 & I_1(x, y) > I_2(x, y) \\ 0 & \text{sonst} \end{cases}$$
- Addition (`+`) addiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x, y) = I_1(x, y) + I_2(x, y)$
- Subtraktion (`-`) subtrahiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x, y) = I_1(x, y) - I_2(x, y)$
- Multiplikation (`*`) multipliziert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x, y) = I_1(x, y) \cdot I_2(x, y)$
- Division (`/`) teilt jeden Pixelwert des Bildes I_1 durch den korrespondierenden Wert in I_2. Bei Division durch Null wird das Pixel auf den Wert Eins gesetzt.
  $$I_R(x, y) = \begin{cases} 1 & \text{falls} \quad I_2(x, y) = 0 \\ I_1(x, y)/I_2(x, y) & \text{sonst} \end{cases}$$
- Minimum (`Min`) berechnet das Minimum der korrespondierenden Pixel von I_1 und I_2.
  $I_R(x, y) = \min\{I_1(x, y), I_2(x, y)\}$
- Maximum (`Max`) berechnet das Maximum der korrespondierenden Pixel von I_1 und I_2.
  $I_R(x, y) = \max\{I_1(x, y), I_2(x, y)\}$

Mit den so definierten elementaren Funktionen lassen sich einfache Kantendetektoren aufbauen. Solch ein einfacher Kantenoperator, basierend auf den Sobel-Masken [113], ist in Abbildung 5.2 gezeigt.

Fitneßfunktion

Das Ziel des Experimentes war es, die Ausgabe des Canny-Kantendetektors zu approximieren. Daher wurde zur Fehlerberechnung die quadrierte Differenz zwischen der gewünschten und der tatsächlichen Ausgabe herangezogen. Zudem wurde ein Term definiert, der für uniforme Ausgaben der Individuen einen sehr großen Fehler erzeugt. Der Term wurde eingeführt, da sonst ein Bild, bei dem alle Pixel auf Null gesetzt sind (oder auf einen kleinen Wert), einen

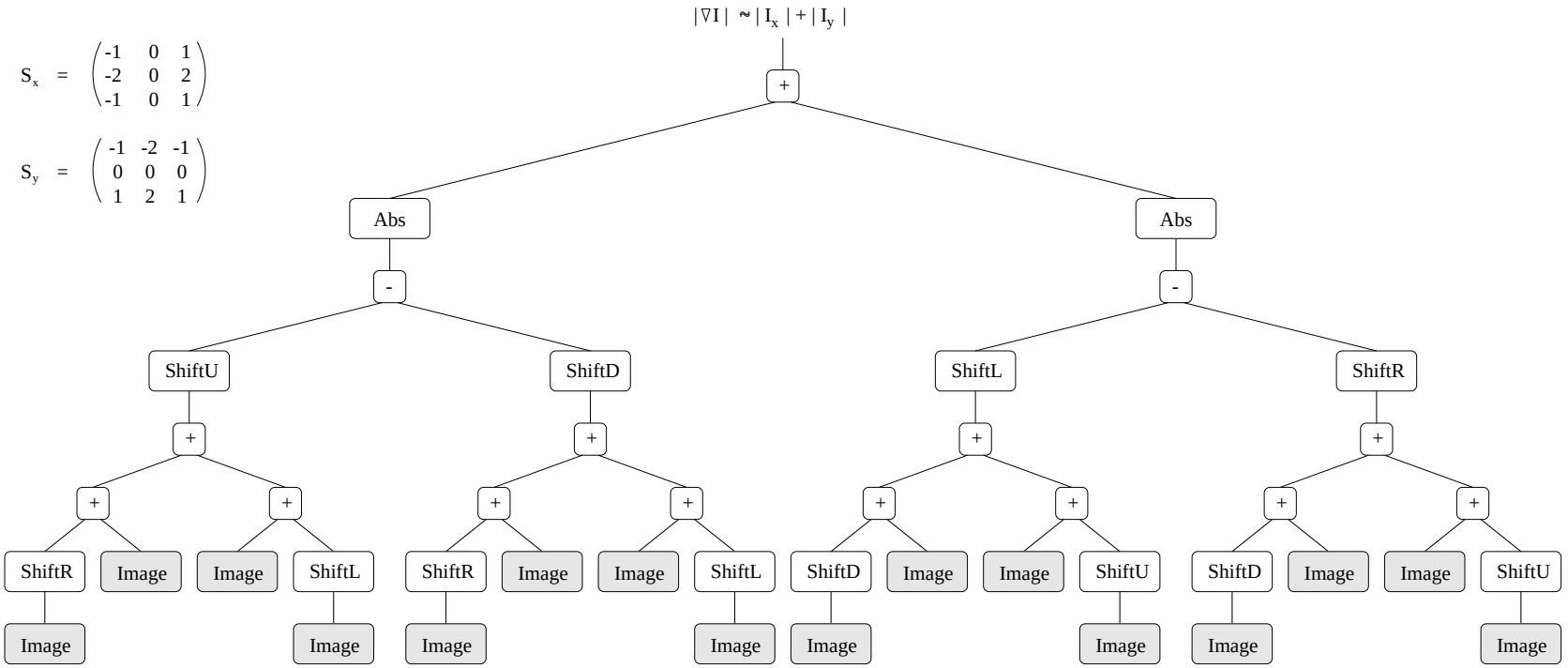


Abbildung 5.2: Ein Kantendetektor, basierend auf den Sobel-Masken.

sehr kleinen Fehler besitzen würde. Um die Kantendetektoren zu evolvieren, wurden fünf verschiedene Fitneßtests verwendet. Der Fehler e wurde wie folgt berechnet.

$$e = \sum_{i=1}^{5} \left(U(\mathrm{Ind}_i) + \frac{1}{n} \sum_{p \in I_i} (\mathrm{Ind}_i(p) - \mathrm{Edges}_i(p))^2 \right) \tag{5.52}$$

Dabei sind die fünf Eingabebilder als $\{I_1, \ldots, I_5\}$ gegeben. Es sei p ein Punkt des Bildes und die Zahl der Punkte sei n. Die Ausgabe des Kantendetektors angewandt auf das Bild I_i sei Edges_i. Die Ausgabe des Individuums angewandt auf das Bild I_i sei Ind_i. Für uniforme Bilder ergibt der Term $U(\mathrm{Ind}_i)$ einen sehr hohen Wert, für nicht-uniforme Bilder ist er Null. Die Fitneß der Individuen wird schließlich durch Fitneß $= \frac{1}{1+e}$ berechnet.

5.2.2 Experimente

In den hier beschriebenen Experimenten wurde der Canny-Kantendetektor eingesetzt, um Kanten aus den Eingabebildern zu extrahieren. Als Fitneßtests wurden fünf unterschiedliche Bilder ausgewählt, die unter alltäglichen Bedingungen aufgenommen wurden. Jedes der Bilder hat 128×128 Pixel. Die Ausgabe des Canny-Kantendetektors wurde binarisiert, so daß alle Kantenpixel auf den Wert Eins gesetzt waren und alle anderen auf den Wert Null.

Als Menge von Terminal-Symbolen und elementaren Funktionen wurde mit verschiedenen Teilmengen der oben beschriebenen Funktionen gearbeitet. Es wurde nicht erwartet, daß die Evolution einen vollständigen Canny-Kantendetektor produziert, da weder ein Gauß-Operator noch eine nicht-lokale Maximaunterdrückung als elementare Funktion bereitgestellt wurde. Die folgende Basis-Menge von elementaren Funktionen wurde in allen Experimenten eingesetzt.

$$\mathrm{BASE} = \{\texttt{Neg}, \texttt{Abs}, \texttt{ShiftL}, \texttt{ShiftR}, \texttt{ShiftU}, \texttt{ShiftD}, \texttt{-}, \texttt{/}, \texttt{*}, \texttt{+}, \texttt{Max}, \texttt{Min}, \texttt{Image}\} \tag{5.53}$$

Hierbei ist das Eingabebild durch das Terminal-Symbol `Image` gegeben. Als Ergänzung dieser Basis-Menge wurde die folgende Menge elementarer Funktionen eingesetzt.

$$\mathrm{EXT} = \{\texttt{UThr}, \texttt{Thr}, \mathcal{R}\} \tag{5.54}$$

Mit allen Teilmengen aus EXT und der Basis-Menge BASE wurden 8 verschiedene Experimente durchgeführt. Jedes Experiment wurde zweimal durchgeführt. Dabei wurde jedes Mal eine Populationsgröße von 4000 Individuen verwendet. Für den ersten Durchlauf wurde die Zahl der Knoten auf 1000 begrenzt und die maximal zulässige Tiefe der Individuen auf 17 gesetzt. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 6 gefüllt. Es wurde fitneßproportionale Selektion mit der Crossover-Wahrscheinlichkeit $p_{\mathrm{cross}} = 0.9$ und der Reproduktions-Wahrscheinlichkeit $p_{\mathrm{rep}} = 0.1$ eingesetzt. Für den zweiten Durchlauf wurde weder die Zahl der Knoten noch die Tiefe der Individuen limitiert. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 10 gefüllt. Außerdem wurde verstärkte fitneßproportionale Selektion mit der Crossover-Wahrscheinlichkeit $p_{\mathrm{cross}} = 0.85$, der Reproduktions-Wahrscheinlichkeit $p_{\mathrm{rep}} = 0.1$ und der Mutations-Wahrscheinlichkeit $p_{\mathrm{mut}} = 0.05$ verwendet. Im zweiten Durchlauf verbesserte sich die Fitneß annähernd stetig, während sie sich im ersten Durchlauf oft sprunghaft veränderte. Die Experimente wurden jeweils nach 25 Generationen abgebrochen. Die evolvierten Kantendetektoren wurden mit 5 weiteren Bildern getestet. Das Individuum, das die beste Fitneß

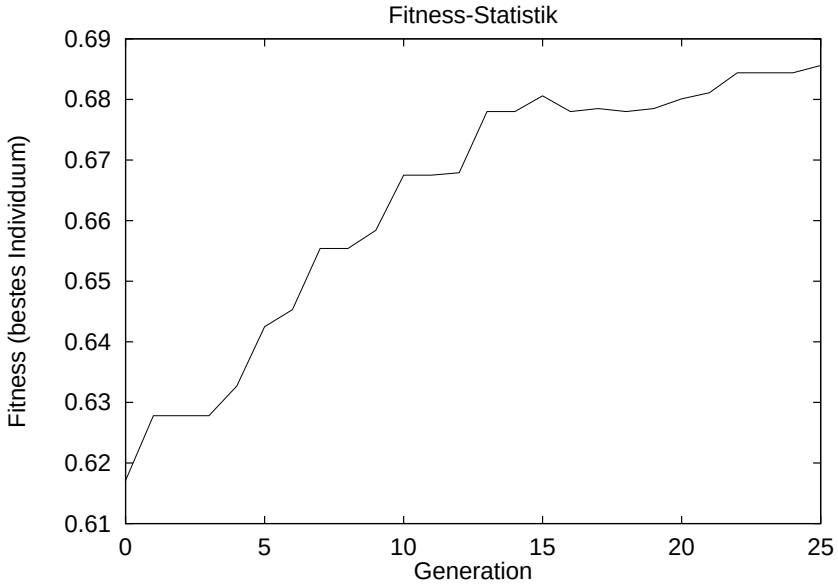


Abbildung 5.3: Fitneß-Statistik des Experimentes, das das beste Individuum produzierte.

erreichte, wurde mit den elementaren Funktionen BASE $\cup$ { `UThr` } während des zweiten Durchlaufs evolviert. Die Fitneß-Statistik dieses Experimentes ist in Abbildung 5.3 dargestellt.

Eine vereinfachte Form des Individuums wird in Abbildung 5.4 gezeigt. Um das Individuum zu vereinfachen, wurden einige Zweige herausgetrennt, wie es z.B. bei $\min\{\max\{a, b\}, b\} = b$ möglich ist. Die Antwort des evolvierten Kantendetektors auf die Eingabebilder, die für die Fitneßberechnungen herangezogen wurden, ist in Abbildung 5.5 gezeigt. Zum Vergleich ist auch die Antwort des Canny-Kantendetektors angegeben. Ferner wurde die Antwort des evolvierten Detektors mit einem Schwellwert binarisiert. Das Ergebnis dieser Binarisierung ist ebenfalls in Abbildung 5.5 zu sehen. In Abbildung 5.6 sind die Ergebnisse für die Testbilder zu sehen. Wiederum wird die Antwort des Canny-Kantendetektors, die Antwort des evolvierten Individuums und die binarisierte Antwort des Individuums gezeigt.

Die erste Generation des Laufes, der das beste Individuum produzierte, wurde genauer untersucht. Das beste Individuum aus der ersten Generation ist nicht in der Lage, Kanten zu detektieren. Die Ausgabe dieses zufällig erzeugten Individuums ist in Abbildung 5.7 zu sehen.

Neben dem oben beschriebenen Kantendetektor wurden im ersten Durchlauf der Experimente zwei interessante Detektoren evolviert. Sie entstanden in den Experimenten, bei denen die elementaren Funktionen BASE $\cup \{\text{UThr}, \mathcal{R}\}$ und BASE $\cup \{\text{UThr}, \text{Thr}, \mathcal{R}\}$ eingesetzt wurden. Beide Individuen extrahieren vertikale Kanten. Sie subtrahieren für jedes Pixel das Pixel auf der rechten Seite vom Pixel auf der linken Seite. Danach wird das Minimum einer kleinen Konstante und des Betrages der Differenz berechnet. Dadurch werden große Pixelunterschiede auf einen kleinen Wert gesetzt. Die Struktur der beiden Individuen ist in Abbildung 5.8 zu sehen. Die Antwort des ersten der beiden Individuen ist in Abbildung 5.9 zu sehen.

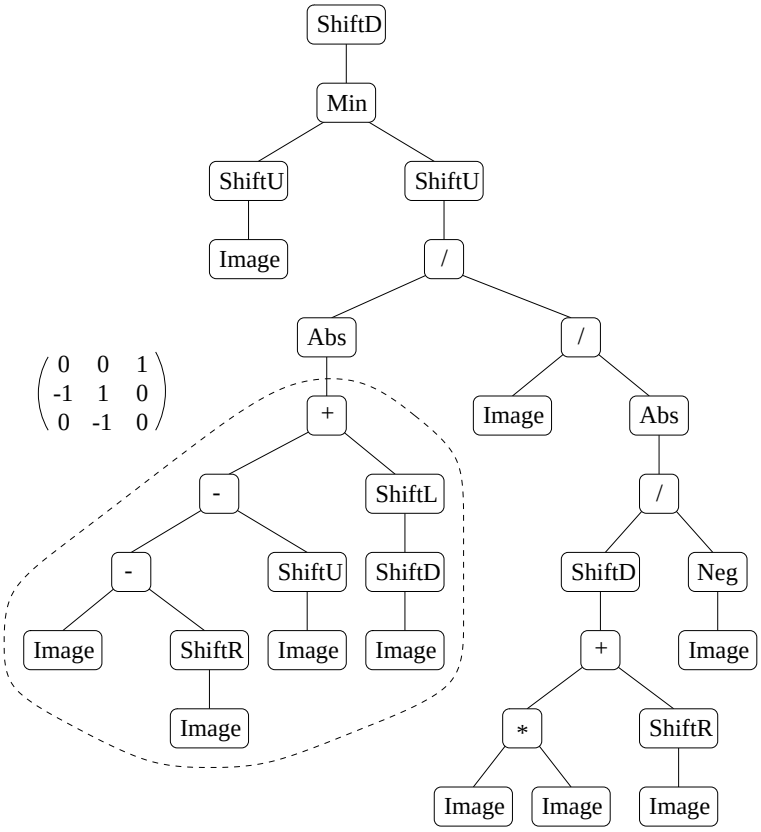


Abbildung 5.4: Der evolvierte Kantendetektor (manuell vereinfacht). Der markierte Teilbaum des Individuums kann als Maske aufgefaßt werden, die sich zur Kantendetektion eignet. Der rechte Teilbaum des Individuums ist ein vom Bild abhängiger Skalierungsfaktor.

5.2.3 Zusammenfassung

Es wurde ein Kantendetektor evolviert, der den Canny-Kantenoperator approximiert. Als Menge elementarer Funktionen wurden lediglich einfache Operationen, wie z.B. Verschiebung des Bildes oder arithmetische Operationen eingesetzt. Die Evolution hat die Funktion eines komplexen Operators mit einfachen elementaren Funktionen approximiert. Dieses Ergebnis ist interessant, da es beim Einsatz von evolutionären Algorithmen zur Lösung beliebiger Probleme der Bildverarbeitung passieren kann, daß eventuell wichtige elementare Funktionen vom Anwender nicht definiert wurden.

Eingabebilder:

Canny-Kantendetektor:

Bester evolvierter Kantendetektor:

Bester evolvierter Kantendetektor (binarisiert):

Abbildung 5.5: Ergebnisse, die mit dem evolvierten Kantendetektor erreicht wurden. Die erste Reihe zeigt die Eingabebilder. In der zweiten Reihe ist jeweils die Ausgabe des Canny-Kantendetektors zu sehen. In der dritten Reihe ist die Antwort des besten evolvierten Kantendetektors zu sehen. Die letzte Reihe zeigt die binarisierten Kanten.

Eingabebilder:

Canny-Kantendetektor:

Bester evolvierter Kantendetektor:

Bester evolvierter Kantendetektor (binarisiert):

Abbildung 5.6: Test des evolvierten Kantendetektors auf neuen Bildern. Die erste Reihe zeigt die Eingabebilder. In der zweiten Reihe ist jeweils die Ausgabe des Canny-Kantendetektors zu sehen. In der dritten Reihe ist die Antwort des besten evolvierten Kantendetektors zu sehen. Die letzte Reihe zeigt die binarisierten Kanten.

Abbildung 5.7: Antwort des besten Individuums, aus der ersten Generation, die schließlich das beste Individuum aller Experimente produzierte. Dieses Individuum ist nicht in der Lage, Kanten zu detektieren.

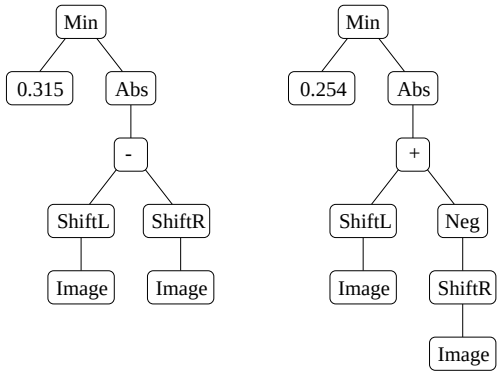


Abbildung 5.8: Evolvierte Detektoren, die vertikale Kanten detektieren. Beide Detektoren wurden manuell vereinfacht.

Detektor für vertikale Kanten:

Detektor für vertikale Kanten (binarisiert):

Abbildung 5.9: Die Bilder in der ersten Reihe zeigen die Ausgabe des ersten der beiden Detektoren, die vertikale Kanten detektieren. In der zweiten Reihe wurde die Antwort des Detektors binarisiert.

5.3 Evolution von Operatoren zur Extraktion markanter Punkte

Operatoren zur Extraktion markanter Punkte werden in vielen Bereichen der Bildverarbeitung, z.B. in der Photogrammetrie und in der Robotik eingesetzt. Die extrahierten Punkte können z.B. zur Berechnung des optischen Flusses oder zur visuellen Lokalisation eines mobilen Roboters eingesetzt werden. Für die markanten Punkte eines Bildes lassen sich sehr leicht die korrespondierenden Punkte in weiteren Bildern der gleichen Szene finden [133, 315, 330, 323]. Auf diese Weise kann aus einer Bildsequenz oder aus zwei Bildern, die mit einer Stereo-Kamera aufgenommen wurden, ein dreidimensionales Modell der Umgebung erstellt werden [305, 301, 302, 27]. In der Photogrammetrie wird ein Punkt als markant definiert, wenn er eindeutig und präzise lokalisiert werden kann [118]. Was ein markanter Punkt einer Umgebung ist, hängt von der Art der Umgebung und dem Algorithmus ab, der die extrahierten Informationen schließlich weiterverarbeitet. Daher bietet sich ein adaptiver Ansatz an, der je nach Umgebung und Algorithmus optimale Merkmale extrahiert.

Im folgenden wird mit Hilfe von genetischem Programmieren versucht, einen bereits existierenden Operator, den sogenannten Moravec-Interest-Operator zu evolvieren (siehe Ebner [79]). Während hier noch ein bereits bekannter Operator vorgegeben wird, wird im nächsten Unterkapitel versucht, einen neuen Operator zu evolvieren, der optimal eine Reihe von Kriterien erfüllt.

5.3.1 Repräsentation

Der in Abschnitt 5.1.5 definierte Ausdruck für den Moravec-Operator, der auf einzelnen Pixeln arbeitet, kann in einen Ausdruck umgeformt werden, der auf ganzen Bildern arbeitet. Diese Form eignet sich besonders für eine parallele Implementierung. Die Struktur des Operators ist in Abbildung 5.10 zu sehen. Zunächst wird das Eingabebild für jede der vier Richtungen verschoben. Danach wird die quadrierte Differenz gebildet. Schließlich werden einzelne Werte in einem 4×4 großen Bereich zusammengefaßt und das Minimum für die vier Richtungen berechnet. Aufgrund der Struktur des Operators werden die Terminal-Symbole und die elementaren Funktionen wie folgt definiert.

Terminal-Symbole

Als Terminal-Symbol wird lediglich das Eingabebild `Image` verwendet. Die Intensitätswerte sind auf den Bereich [0,1] skaliert.

Elementare Funktionen

Das Eingabebild einer elementaren Funktion wird im folgenden mit I oder I_i, wobei $i \in \{1, \ldots, 4\}$, bezeichnet. Das Ausgabebild sei I_R.
Unäre Funktionen:

- Negation (`Neg`) negiert alle Pixelwerte.
  $I_R(x, y) = -I(x, y)$
- Betrag (`Abs`) berechnet für jeden Pixel den Betrag.
  $I_R(x, y) = |I(x, y)|$

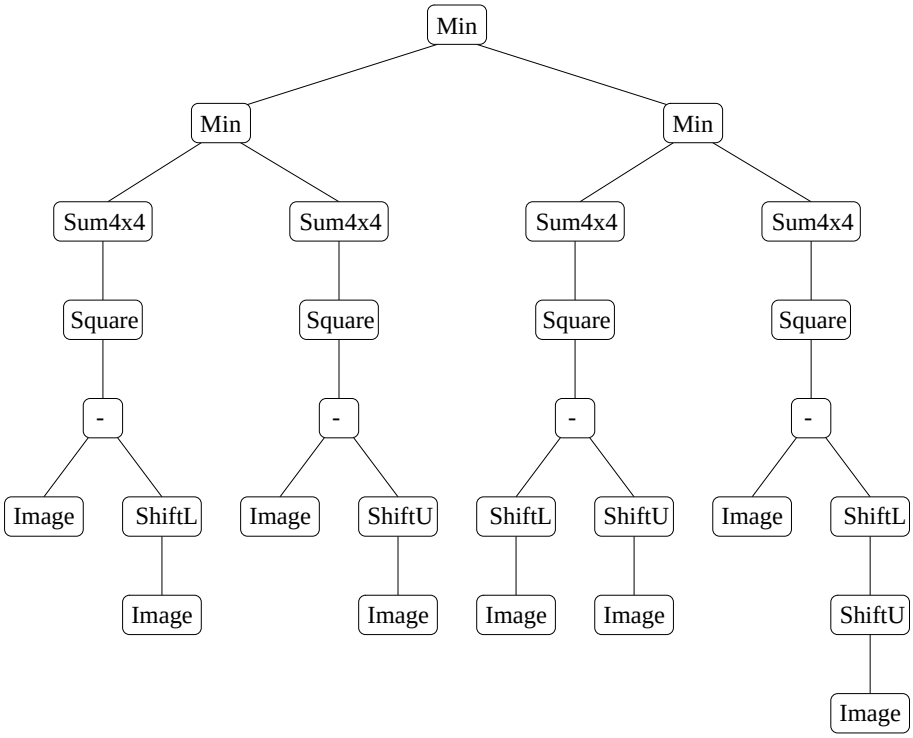


Abbildung 5.10: Struktur des Moravec-Operators.

- Quadrat (`Square`) quadriert alle Pixelwerte.
  $I_R(x, y) = I(x, y) \cdot I(x, y)$
- Shift nach links (`ShiftL`) verschiebt das Bild I um ein Pixel nach links.
  $I_R(x, y) = I(x + 1, y)$
- Shift nach rechts (`ShiftR`) verschiebt das Bild I um ein Pixel nach rechts.
  $I_R(x, y) = I(x - 1, y)$
- Shift nach oben (`ShiftU`) verschiebt das Bild I um ein Pixel nach oben.
  $I_R(x, y) = I(x, y + 1)$
- Shift nach unten (`ShiftD`) verschiebt das Bild I um ein Pixel nach unten.
  $I_R(x, y) = I(x, y - 1)$
- Durchschnitt (`Avg4x4`) berechnet den durchschnittlichen Wert der Pixel in einem 4×4 großen Bereich.
  $I_R(x, y) = \frac{1}{16} \sum_{-1 \leq i,j \leq 2} I(x + i, y + j)$
- Summe (`Sum4x4`) summiert die Pixelwerte in einem 4×4 großen Bereich.
  $I_R(x, y) = \sum_{-1 \leq i,j \leq 2} I(x + i, y + j)$

Binäre Funktionen:

- Addition (`+`) addiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x, y) = I_1(x, y) + I_2(x, y)$

- Subtraktion (`-`) subtrahiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x,y) = I_1(x,y) - I_2(x,y)$
- Multiplikation (`*`) multipliziert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x,y) = I_1(x,y) \cdot I_2(x,y)$
- Division (`/`) teilt jeden Pixelwert des Bildes I_1 durch den korrespondierenden Wert in I_2. Bei Division durch Null wird das Pixel auf den Wert Eins gesetzt.
  $$I_R(x,y) = \begin{cases} 1 & \text{falls} \quad I_2(x,y) = 0 \\ I_1(x,y)/I_2(x,y) & \text{sonst} \end{cases}$$
- Minimum (`Min`) berechnet das Minimum der korrespondierenden Pixel von I_1 und I_2.
  $I_R(x,y) = \min\{I_1(x,y), I_2(x,y)\}$
- Maximum (`Max`) berechnet das Maximum der korrespondierenden Pixel von I_1 und I_2.
  $I_R(x,y) = \max\{I_1(x,y), I_2(x,y)\}$

Funktionen mit N Argumenten ($N \in \{3,4\}$)

- Addition (`AddN`) addiert korrespondierende Pixelwerte von I_i.
  $I_R(x,y) = \sum_{i=1}^{i=N} I_i(x,y)$
- Multiplikation (`PiN`) multipliziert korrespondierende Pixelwerte von I_i.
  $I_R(x,y) = \prod_{i=1}^{i=N} I_i(x,y)$
- Minimum (`MinN`) berechnet das Minimum der korrespondierenden Pixel von I_i.
  $I_R(x,y) = \min\{I_i(x,y) | i \in \{1,\ldots,N\}\}$
- Maximum (`MaxN`) berechnet das Maximum der korrespondierenden Pixel von I_i.
  $I_R(x,y) = \max\{I_i(x,y) | i \in \{1,\ldots,N\}\}$

Fitneßfunktion

Wie bei der Evolution der Kantendetektoren wurde hier wieder der Fehler aus der quadrierten Differenz zwischen gewünschter Ausgabe und der tatsächlichen Ausgabe berechnet. Als gesuchter Operator zur Extraktion markanter Punkte wurde der Moravec-Operator eingesetzt. Ferner erhielten Individuen, die ein uniformes Bild ausgeben, durch einen zusätzlich eingeführten Term einen großen Fehler. Jedes Individuum wurde mit 5 verschiedenen Eingabebildern der Größe 128×128 getestet. Der Fehler e der Individuen wurde wie folgt berechnet.

$$e = \sum_{i=1}^{5} \left(U(\mathrm{Ind}_i) + \frac{1}{n} \sum_{p \in I_i} \left(\mathrm{Ind}_i(p) - \mathrm{Moravec}_i(p)\right)^2 \right) \tag{5.55}$$

Dabei sind die fünf Eingabebilder als $\{I_1, \ldots, I_5\}$ gegeben. Es sei p ein Punkt des Bildes und die Zahl der Punkte sei n. Die Ausgabe des Moravec-Operators, angewandt auf das Bild I_i, sei $\mathrm{Moravec}_i$. Die Ausgabe des Individuums, angewandt auf das Bild I_i, sei Ind_i. Für uniforme Bilder ergibt der Term $U(\mathrm{Ind}_i)$ einen sehr hohen Wert, für nicht-uniforme Bilder ist er Null. Aus dem Fehlermaß e wird die Fitneß der Individuen durch $\mathrm{Fitneß} = \frac{1}{1+e}$ berechnet.

1	BASE
2	BASE $\cup$ { `Avg4x4` }
3	BASE $\cup$ { `Sum4x4` }
4	BASE $\cup$ { `Sum4x4`, `Pi3`, `Add3`, `Max3`, `Min3` }
5	BASE $\cup$ { `Sum4x4`, `Pi3`, `Add3`, `Max3`, `Min3`, `Pi4`, `Add4`, `Max4`, `Min4` }

Tabelle 5.1: Mengen elementarer Funktionen, die bei den einzelnen Experimenten eingesetzt wurden.

5.3.2 Experimente

Fünf Experimente wurden durchgeführt, um einen Operator zur Extraktion markanter Punkte zu evolvieren. Bei den Experimenten wurde jeweils mit einer anderen Menge elementarer Funktionen gearbeitet. Für die Experimente wurde eine Populationsgröße von 4000 Individuen eingesetzt. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 6 gefüllt. Die maximale Zahl der Knoten eines Individuums wurde auf 1000 und die Tiefe des Individuums auf 17 begrenzt. Die Crossover-Wahrscheinlichkeit wurde auf 85%, die Reproduktions-Wahrscheinlichkeit wurde auf 10% und die Mutations-Wahrscheinlichkeit wurde auf 5% gesetzt. Zur Selektion der Individuen wurden die verstärkte fitneßproportionaler Selektion eingesetzt. Die Läufe wurden jeweils nach 50 Generationen abgebrochen.

Die folgende Menge elementarer Funktionen wurde bei allen Experimenten eingesetzt.

$$\text{BASE} = \{\texttt{Neg}, \texttt{Abs}, \texttt{Square}, \texttt{ShiftL}, \texttt{ShiftR}, \texttt{ShiftU}, \texttt{ShiftD}, \texttt{-}, \texttt{/}, \texttt{*}, \texttt{+}, \texttt{Max}, \texttt{Min}\} \quad (5.56)$$

Im ersten Experiment wurde lediglich diese Basis-Menge eingesetzt. In den folgenden Experimenten wurde die Basis-Menge durch weitere elementare Funktionen ergänzt. Die elementaren Funktionen, die bei den einzelnen Experimenten verwendet wurden, sind in Tabelle 5.1 zusammengefaßt.

Die Basis-Menge reicht aus, um den Moravec-Operator zu evolvieren. Eine elementare Funktion, `Sum4x4`, scheint zu fehlen. Diese Funktion kann jedoch aus den Verschiebungsoperationen und der Additionsoperation aufgebaut werden. Da die `Sum4x4`-Operation im Moravec-Operator viermal eingesetzt wird, erschwert das Fehlen dieser Operation die Suche nach einem korrekten Operator. Im zweiten Experiment wurde die Basis-Menge und die elementare Funktion `Avg4x4` eingesetzt. Jetzt hätte die Evolution die Möglichkeit, die benötigte Funktion Sum4x4 durch Evolution einer Konstanten zu erzeugen (z.B. $\texttt{Sum4x4} = 16 \cdot \texttt{Avg4x4}$ wobei $16 = \texttt{Square}(\texttt{Square}(\frac{\texttt{Image}}{\texttt{Image}} + \frac{\texttt{Image}}{\texttt{Image}}))$). Im dritten Experiment wurde die Basis-Menge und die Funktion `Sum4x4` verwendet. Im vierten und fünften Experiment wurden die Zahl der möglichen Argumente der Funktionen `Min`, `Max`, `+`, `*` auf 3 bzw. auf 3 und 4 erweitert.

Das beste Individuum aller Experimente wurde mit den elementaren Funktionen BASE $\cup$ { `Sum4x4`, `Pi3`, `Add3`, `Max3`, `Min3`, `Pi4`, `Add4`, `Max4`, `Min4` } (Experiment 5) evolviert. In Abbildung 5.11 ist das Individuum manuell vereinfacht dargestellt. Dabei wurden Vereinfachungen, wie z.B. $a - 0 = a$ oder $\max\{a, a\} = a$, vorgenommen. Das beste evolvierte Individuum ist nur eine Approximation des Moravec-Operators. Die Hälfte des Individuums entspricht dem Moravec-Operator genau. Die Fitneß des Individuums ist besonders gut für vier der fünf Eingabebilder. Für sie ist das Fehlermaß kleiner als 0.0005. Da der Moravec-Operator eine relativ einfache Struktur hat, ist es überraschend, daß kein 100% korrekter Moravec-Operator evolviert wurde. Dies könnte an der Struktur des Operators an der Spitze

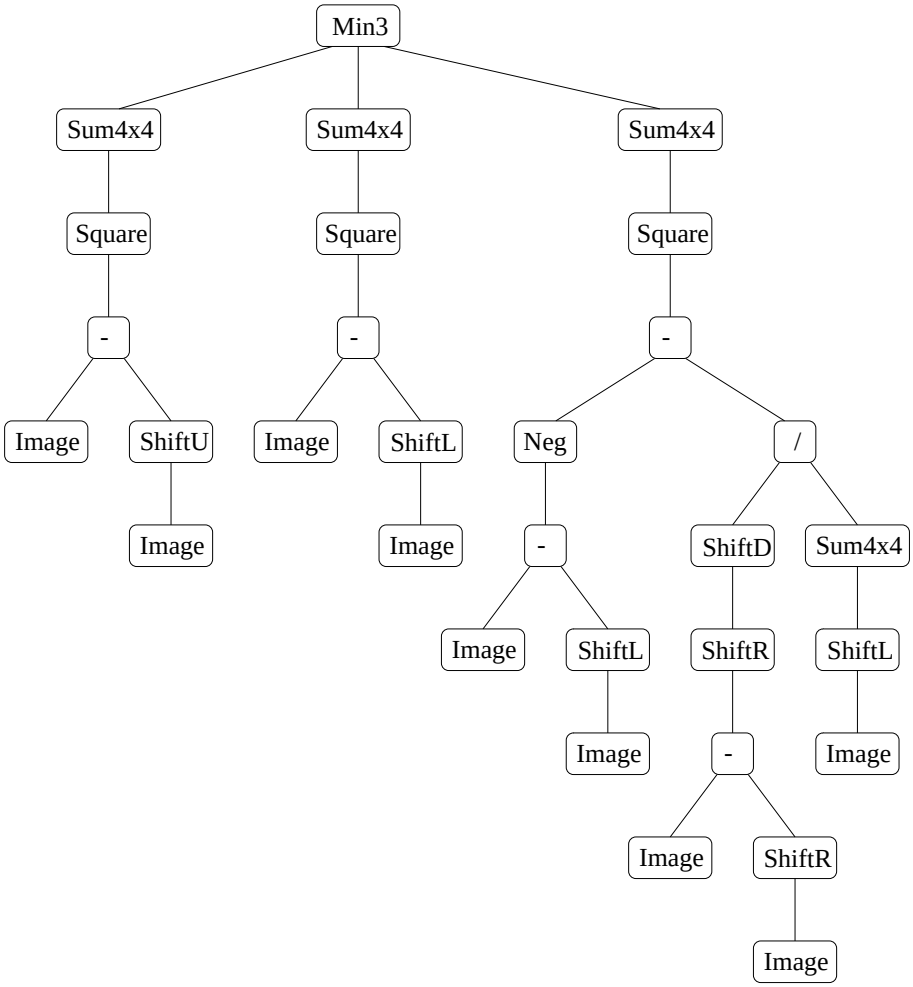


Abbildung 5.11: Bester evolvierter Operator zur Extraktion markanter Punkte (der Operator wurde manuell vereinfacht). Der Operator wurde mit der Menge BASE ∪ { `Sum4x4`, `Pi3`, `Add3`, `Max3`, `Min3`, `Pi4`, `Add4`, `Max4`, `Min4` } (Experiment 5) evolviert.

des Baumes liegen. Die Struktur, bestehend aus drei Minimum-Operatoren, könnte schwierig zu finden sein. Das beste Individuum der ersten Generation des Laufes, der schließlich das oben gezeigte Individuum produzierte, wurde ebenfalls untersucht. Es hebt lediglich vertikale Kanten hervor.

Abbildung 5.12 zeigt die fünf Eingabebilder, die zur Evolution des Operators verwendet wurden. Die markierten Punkte wurden mit dem Moravec-Operator extrahiert. Darunter ist jeweils die Ausgabe des Moravec-Operators gezeigt. In der dritten Reihe ist jeweils die Ausgabe des evolvierten Operators zu sehen. In der vierten Reihe sind nochmals die Eingabebilder zu sehen, wobei die markanten Punkte mit dem evolvierten Operator extrahiert wurden. Dazu wurde eine nicht-lokale Maxima-Unterdrückung durchgeführt und Punkte, die über einem Schwellwert liegen, wurden als markante Punkte extrahiert.

Der evolvierte Operator wurde auf fünf neuen Bildern getestet. Das Ergebnis dieses Tests ist in Abbildung 5.13 zu sehen. In der ersten Reihe sind die fünf Testbilder zu sehen. Die

Eingabebilder (markante Punkte extrahiert mit dem Moravec-Operator):

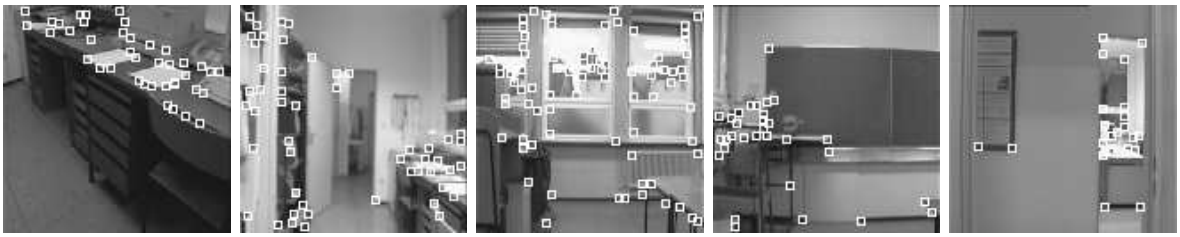
Antwort des Moravec-Operators:

Bester evolvierter Operator zur Extraktion markanter Punkte:

Merkmale, die mit dem evolvierten Operator extrahiert wurden:

Abbildung 5.12: Die erste Reihe zeigt die Eingabebilder. Die markanten Punkte wurden mit dem Moravec-Interest-Operator extrahiert. In der zweiten Reihe ist die Antwort des Moravec-Operators zu sehen. In der dritten Reihe ist die Antwort des besten evolvierten Individuums zu sehen. Die markanten Punkte, die dieses Individuum extrahiert, sind in der letzten Zeile in den Eingabebildern markiert.

Eingabebilder (markante Punkte extrahiert mit dem Moravec-Operator):



Antwort des Moravec-Operators:

Bester evolvierter Operator zur Extraktion markanter Punkte:

Merkmale, die mit dem evolvierten Operator extrahiert wurden.

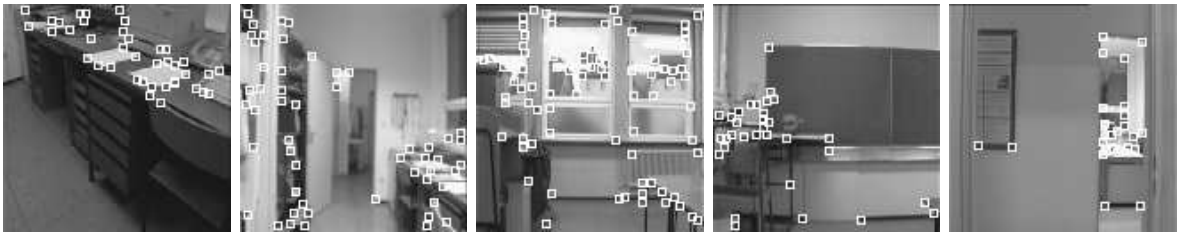


Abbildung 5.13: Fünf Bilder wurden eingesetzt, um den evolvierten Operator zu testen. Die erste Reihe zeigt die Eingabebilder. Die markanten Punkte wurden mit dem Moravec-Operator extrahiert. In der zweiten Reihe ist jeweils die Antwort des Moravec-Operators zu sehen. In der dritten Zeile ist die Antwort des besten evolvierten Operators zu sehen. Die von diesem Operator extrahierten Merkmale sind in der letzten Reihe in den Eingabebildern markiert.

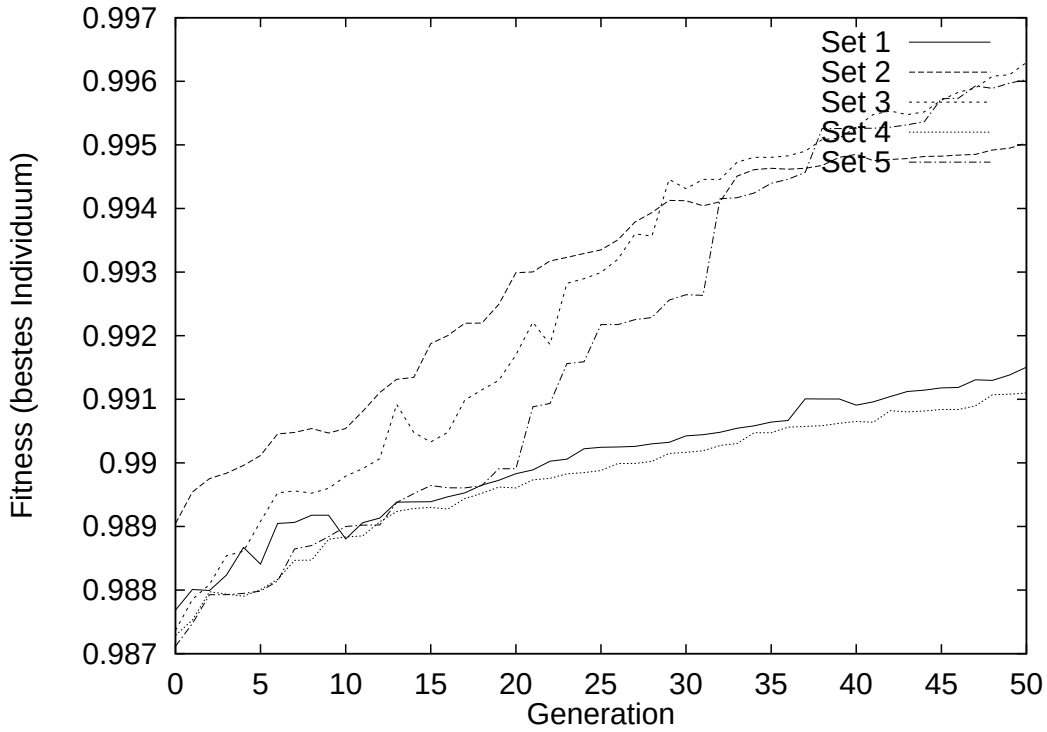


Abbildung 5.14: Fitneß-Statistiken aller Experimente. Die Fitneß wurde für jedes Experiment über drei Läufe mit unterschiedlicher Initialisierung gemittelt.

markanten Punkte wurden mit dem Moravec-Operator extrahiert. In der Zeile darunter ist jeweils die Ausgabe des Moravec-Operators dargestellt. In der dritten Zeile ist die Ausgabe des besten evolvierten Operators zu sehen. In der vierten Zeile sind schließlich nochmals die Testbilder zu sehen. Die markanten Punkte wurden mit dem evolvierten Operator extrahiert.

Die Fitneß-Statistiken sind in Abbildung 5.14 zusammengefaßt. Dargestellt ist für jede Generation die Fitneß des besten Individuums, gemittelt über drei verschiedene Läufe. Im dritten Experiment mit der Menge BASE $\cup$ { Sum4x4 } wurde die beste durchschittliche Fitneß erreicht. Für die Experimente 1 und 4 ist die durchschnittliche Fitneß besonders niedrig. Dies läßt sich dadurch erklären, daß beim Experiment 1 ein wesentlicher Operator, eine lokale Summation, nicht bereit gestellt wurde und beim Experiment 4 der Suchraum durch zusätzliche Funktionen vergrößert wurde.

5.3.3 Zusammenfassung

Es wurde ein Operator zur Extraktion markanter Punkte evolviert. Aufgabe war es, die Ausgabe eines bereits existierenden Operators, des Moravec-Interest-Operators, zu approximieren. Unter Verwendung einer Populationsgöße von 4000 Individuen wurden in 50 Generationen kein 100% korrekter Operator evolviert. Dennoch ist der Fehler zwischen evolviertem Operator und dem vorgegebenen Operator sehr gering. Die Schwierigkeit des Problems könnte in der Struktur des Operators an der Spitze des Baumes begründet sein.

Nachdem hier noch ein bereits existierender Operator vorgegeben wurde, wird im nächsten

Abbildung 5.15: Antwort des besten Operators aus der ersten Generation des Laufes, der das beste Individuum produzierte.

Unterkapitel versucht, einen neuen Operator zu evolvieren. Dabei werden lediglich die gewünschten Eigenschaften des Operators vorgegeben.

5.4 Evolution von Operatoren zur Berechnung des optischen Flusses

In der Bildverarbeitung werden in der Regel eine Reihe von Operatoren eingesetzt, um ein bestimmtes Problem zu lösen. Welcher Operator geeignet ist, hängt von der Aufgabe und von den Randbedingungen der Umgebung ab, in der er eingesetzt wird. Hier soll nun ein Operator evolviert werden, der optimal an die Aufgabe und die gegebenen Randbedingungen angepaßt ist. Zur Evolution der Operatoren wird wieder genetisches Programmieren eingesetzt. Aufgabe soll sein, einen Operator zu evolvieren, der markante Punkte extrahiert, für die der optische Fluß berechnet wird (siehe Ebner und Zell [87]). Dazu werden eine Reihe von Qualitätskriterien definiert, die der Operator erfüllen sollte. Im folgenden wird gezeigt, wie ein Operator evolviert werden kann, der auf die gegebenen Kriterien hin optimiert ist.

Im Zuge immer schneller werdender Rechner könnte es in Zukunft möglich werden, Operatoren zu evolvieren, während sie eingesetzt werden. Dann könnte ein künstliches, visuelles System eines mobilen Roboters sich z.B. an gegebene Lichtverhältnisse anpassen. Genau wie beim visuellen System des Menschen der Durchmesser der Pupille sich an die gegebenen Lichtverhältnisse anpaßt, könnten in einem künstlichen System Operatoren evolviert werden, die für die Lichtverhältnisse optimal arbeiten.

Mobile Roboter könnten aber auch lernen, nur relevante Merkmale zu extrahieren. Für eine Lokalisation eines Roboters sind hauptsächlich stationäre Merkmale relevant. Merkmale, die zyklische Bewegungen ausführen aber dennoch einen bestimmten Punkt lokalisieren oder blinkende Merkmale, können ebenfalls zur Lokalisation eingesetzt werden. Merkmale, die durch nicht wiederkehrende Bewegungen verursacht wurden, sind nicht hilfreich und sollten daher gar nicht erst extrahiert werden. Sie könnten von anderen Robotern oder Personen verursacht worden sein, die sich in der gleichen Umgebung aufhalten.

Mit einem adaptiven Algorithmus zur Extraktion markanter Punkte könnte man auch während der Laufzeit optimale Merkmale extrahieren. Aufgrund der dafür erforderlichen Rechenleistung sind dem Verfahren derzeit jedoch noch Grenzen gesetzt. Die hier beschriebenen Ergebnisse wurden off line evolviert.

5.4.1 Qualitätskriterien für Operatoren zur Extraktion von markanten Punkten

Die vom Operator extrahierten markanten Punkte werden eingesetzt, um den optischen Fluß zu berechnen. Der optische Fluß wird für einzelne Punkte berechnet, indem Korrespondenzen zwischen dem vorangegangenen und dem aktuellen Bild hergestellt werden. Dabei wird angenommen, daß der optische Fluß größer als ein Pixel ist. Dies ist entweder bei sehr schnellen Kamerabewegungen der Fall oder falls die Bilder mit einer großen zeitlichen Verzögerung aufgenommen werden. Im konkreten Einsatz kann dies, durch einen Algorithmus bedingt sein, der viel Zeit für die Verarbeitung der visuellen Informationen benötigt. In beiden Fällen kann die Berechnung des optischen Flusses durch die Extraktion markanter Punkte vereinfacht werden. Ein Punkt, der sich an einer Kante befindet, kann im folgenden Bild, sofern nur ein kleiner Ausschnitt betrachtet wird, nicht exakt lokalisiert werden. Dies wird als Apertur-Problem bezeichnet (Abbildung 5.16). Ist jedoch ausreichend Struktur in dem betrachteten Bereich vorhanden und die Bewegung hinreichend klein, so kann der Punkt genau lokalisiert werden.

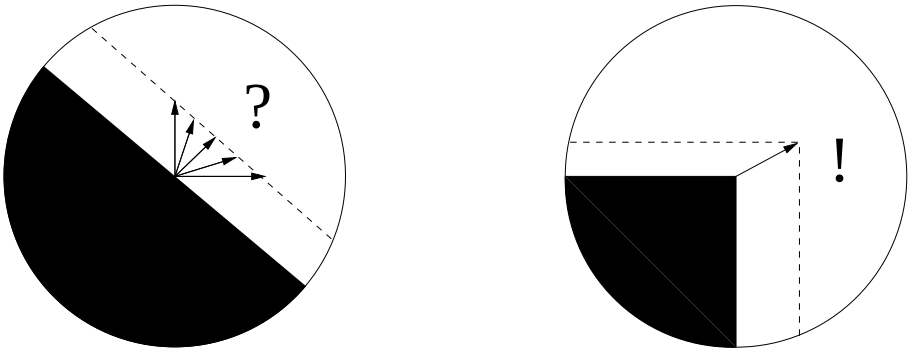


Abbildung 5.16: Ein Punkt, der sich auf einer Kante befindet, kann im Folgebild, sofern nur ein kleiner Ausschnitt sichtbar ist, nicht exakt lokalisiert werden (links). Eine Vielzahl von Flußvektoren sind denkbar. Dies wird als Apertur-Problem bezeichnet. Rechts ist unter der Annahme einer kleinen Bewegung des betrachteten Objektes die Zuordnung eindeutig (Siehe [193, 128, 29, 141, 190]).

Hier wird der optische Fluß lediglich für die markanten Punkte berechnet. Um korrespondierende Punkte zu finden, werden die Intensitätswerte in einem kleinen Bereich um den Punkt miteinander verglichen. Ziel ist es, nur solche Punkte zu extrahieren, die im folgenden Bild exakt lokalisiert werden können. Daher werden eine Reihe von Kriterien definiert, die ein geeigneter Operator zur Extraktion markanter Punkte erfüllen sollte. Mit genetischem Programmieren soll versucht werden, einen Operator zu finden, der die Kriterien optimal erfüllt. Die Kriterien werden zunächst nur qualitativ beschrieben. Sie werden später formalisiert. Um den Operator zu evolvieren, wurden die folgenden Kriterien verwendet.

- Die Zahl der Punkte, für die ein Flußvektor berechnet wird, sollte groß sein. Wird für jedes Bild nur ein einziger Punkt extrahiert, dann kann sehr leicht eine Korrespondenz zwischen den Punkten hergestellt werden. Selbstverständlich soll der Operator soviele Punkte extrahieren wie möglich.

- Die Qualität der Korrespondenzen sollte gut sein. Zum Vergleich zweier Punkte wird ein Fehlermaß berechnet. Der Operator sollte die Punkte extrahieren, für die das Fehlermaß sehr klein ist und die anderen nicht berücksichtigen.

- Ein Schwellwert wird eingesetzt, um Korrespondenzen zwischen den Punkten herzustellen, für die das Fehlermaß hinreichend klein ist. Es wird nicht für jeden Punkt ein Flußvektor berechnet. Es sollte aber möglichst für jeden Punkt, der extrahiert wird, auch ein Flußvektor berechnet werden. Daher sollte der Anteil der Punkte, für die ein Flußvektor berechnet wird, möglichst groß sein. Sonst könnte ein Operator entstehen, der einfach alle Punkte extrahiert und die Auswahl der Punkte dem Algorithmus zur Herstellung der Korrespondenzen überläßt. In diesem Fall müssen sehr viele Punkte miteinander verglichen werden. Das soll natürlich vermieden werden. Denn die Aufgabe des Operators ist es ja gerade, den Suchraum stark einzuschränken.

- Es sollte eindeutig sein, welches der korrespondierende Punkt ist. Jeder Punkt des vorangegangenen Bildes wird mit allen Punkten des aktuellen Bildes verglichen. Daher sollte das Fehlermaß zwischen dem Punkt und dem tatsächlich korrespondierenden

Punkt deutlich kleiner sein, als das für die anderen Punkte. Es sollte also völlig klar sein, welches der korrespondierende Punkt ist.

- Der optische Fluß sollte, abgesehen von einigen Diskontinuitäten, relativ glatt sein. Daher sollten sich benachbarte Flußvektoren nur geringfügig unterscheiden.
- Die Dichte des Flußfeldes kann ebenfalls gesteuert werden. Der Operator sollte sich wirklich nur auf markante Punkte beschränken. Daher kann die Einführung einer maximalen Dichte der extrahierten Punkte sinnvoll sein. Sonst könnte z.B. ein einfacher Kantendetektor entstehen.

Nun sollen die einzelnen Kriterien formalisiert werden. Es sei $I(t)$ das Bild, das zum Zeitpunkt t aufgenommen wurde. Als erstes wird auf dieses Bild der Operator angewandt. Danach werden alle Punkte, die kein lokales Maximum sind, auf Null gesetzt. Extrahiert werden die Punkte, für die die Antwort des Operators größer als ein Schwellwert ϵ_1 ist. Die markanten Punkte des Bildes $I(t)$ seien $F(t)$. Es werden zwei Bilder $I(t_1)$ und $I(t_2)$ zum Zeitpunkt t_1 bzw. t_2 aufgenommen. Zwischen den markanten Punkten $F(t_1)$ und $F(t_2)$ werden korrespondierende Paare ermittelt. Für einen Punkt $(x_1, y_1) \in F(t_1)$ wird für jeden Punkt $(x_2, y_2) \in F(t_2)$, der sich innerhalb eines spezifizierten Abstandes befindet, das folgende Fehlermaß e berechnet.

$$e(x_1, y_1, x_2, y_2) = \sqrt{\frac{1}{wh} \sum_{-\frac{w}{2} \le i < \frac{w}{2}} \sum_{-\frac{h}{2} \le j < \frac{h}{2}} \left(\tilde{I}(x_1 + i, y_1 + j) - \tilde{I}(x_2 + i, y_2 + j) \right)^2} \tag{5.57}$$

Dabei ist $\tilde{I}(t)$ ein mit einem Gauß-Operator geglättetes Bild $I(t)$. Die Größe des Bereiches, der zum Vergleich der Pixelwerte herangezogen wird, ist durch die Breite w und Höhe h definiert. Der Punkt (x_2, y_2), für den das Fehlermaß am kleinsten ist, wird als korrespondierender Punkt ausgewählt. Ein weiterer Schwellwert wird eingesetzt, um schlecht korrespondierende Punkte zurückzuweisen. Eine Korrespondenz wird nur dann hergestellt, wenn das Fehlermaß kleiner als der Schwellwert ϵ_2 ist. Es sei $F_m(t)$ die Menge der Punkte, für die ein korrespondierender Punkt gefunden wurde. Die Zahl der extrahierten Punkte sei n_p, die Zahl der Punkte, für die ein korrespondierender Punkt gefunden wurde, sei n_m.

Die folgenden Kriterien wurden maximiert.

- Zahl der berechneten Flußvektoren (Abbildung 5.17):
  Die Zahl der berechneten Flußvektoren sollte möglichst groß sein.

$$m_1(t_1) = n_m \tag{5.58}$$

- Qualität der Korrespondenzen:
  Es sollten die Punkte extrahiert werden, für die das Fehlermaß klein ist.

$$m_2(t_1) = \frac{1}{n_m} \sum_{(x,y) \in F_m(t_1)} \frac{1}{1 + e_{\min}(x, y)} \tag{5.59}$$

  Wobei

$$e_{\min}(x_1, y_1) = \min_{(x,y) \in F(t_2)} e(x_1, y_1, x, y) \tag{5.60}$$

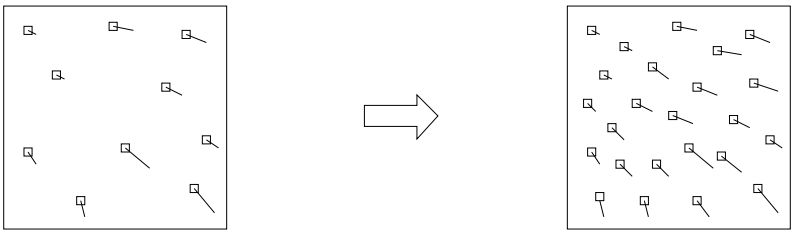


Abbildung 5.17: Die Zahl der berechneten Flußvektoren sollte möglichst groß sein.

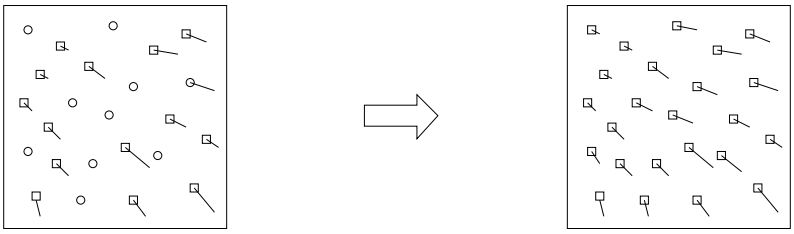


Abbildung 5.18: Der Prozentsatz der Punkte, für die ein Flußvektor berechnet werden kann, sollte möglichst groß sein.

das Minimum des Fehlermaßes e ist. Das Kriterium m_2 ist analog zu einem Qualitätsmaß von Pratt definiert, das zur Bewertung von Kantendetektoren eingesetzt wird [282, 285, 141].

- Prozentsatz der Punkte, für die ein Flußvektor berechnet werden kann (Abbildung 5.18): Es sollte möglichst für alle extrahierten Punkte ein Flußvektor berechnet werden.

$$m_3(t_1) = \frac{n_m}{n_p} \tag{5.61}$$

- Eindeutigkeit der Korrespondenzen:
  Es sollte eindeutig sein, zwischen welchen Punkten eine Korrespondenz hergestellt wird.

$$m_4(t_1) = \frac{1}{n_m} \sum_{(x,y)\in F_m(t_1)} \frac{e_{\text{next}}(x,y) - e_{\min}(x,y)}{e_{\max}(x,y) - e_{\min}(x,y)} \tag{5.62}$$

  Wobei

$$e_{\max}(x_1,y_1) = \max_{(x,y)\in F(t_2)} e(x_1,y_1,x,y) \tag{5.63}$$

  das Maximum von Fehlermaß e ist. Es sei (x_m, y_m) der Punkt, für den das Fehlermaß e minimal ist. Dann ist mit

$$e_{\text{next}}(x_1,y_1) = \min_{(x,y)\in F(t_2)\setminus(x_m,y_m)} e(x_1,y_1,x,y) \tag{5.64}$$

  der Fehler für den zweitbesten Punkt gegeben. Das Kriterium m_4 versucht demnach, den Abstand des Fehlers zwischen bestem und zweitbestem Punkt zu vergrößern. Es sollen nur die Punkte extrahiert werden, für die die Zuordnung eindeutig ist. Ein ähnliches

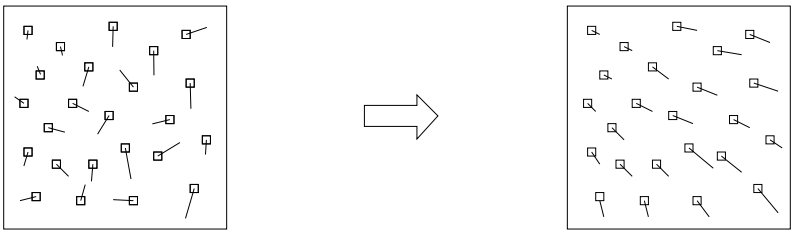


Abbildung 5.19: Das Flußfeld sollte abgesehen von einigen Diskontinuitäten, die nicht zu vermeiden sind, relativ glatt sein.

Maß wurde von Neven et al. [214] bei der Berechnung des optischen Flusses eingesetzt. Der optische Fluß wird nur an den Punkten berechnet, für die die Differenz zwischen bestem Korrelationswert und zweitbestem Korrelationswert groß genug ist. Lew et al. [178] setzten ebenfalls den Abstand zwischen bestem und zweitbestem Fehler zur adaptiven Auswahl geeigneter Merkmale ein, um korrespondierende Punkte in Stereobildern zu finden.

- Glattheit des Flußfeldes (Abbildung 5.19):
  Das Flußfeld sollte abgesehen von einigen Diskontinuitäten, die nicht zu vermeiden sind, relativ glatt sein.

$$m_5(t_1) = \frac{1}{n_p} \sum_{(x,y)\in F(t_1)} s(x,y) \tag{5.65}$$

  wobei s ein Maß für die Glattheit des Flußfeldes innerhalb eines kleinen Bereiches um den Punkt (x, y) ist. Es seien $F_{N(x,y)}$ die Punkte, die sich im Abstand ϵ_3 vom Punkt (x, y) befinden.

$$F_{N(x,y)}(t) = \{(x', y') \in F(t) | \sqrt{(x'-x)^2 + (y'-y)^2} < \epsilon_3\} \tag{5.66}$$

  Dann wird das Maß für die Glattheit wie folgt berechnet.

$$s(x,y) = \frac{1}{2|F_{N(x,y)}(t_1)|} \sum_{(x',y')\in F_{N(x,y)}(t_1)} 1 + \frac{\Delta x \Delta x' + \Delta y \Delta y'}{\sqrt{\Delta x^2 + \Delta y^2}\sqrt{\Delta x'^2 + \Delta y'^2}} \tag{5.67}$$

  wobei $(\Delta x, \Delta y)$ der für den Punkt (x, y) berechnete optische Fluß ist.

- Maximale Dichte der extrahierten Punkte (Abbildung 5.20):
  Die extrahierten Punkte sollten nicht zu dicht beieinander liegen.

$$m_6(t_1) = \frac{1}{n_p} \sum_{(x,y)\in F(t_1)} \min\left\{\frac{d_{\min}(x,y)}{d_{\text{des}}}, 1.0\right\} \tag{5.68}$$

  wobei $d_{\min}(x, y)$ der kleinste Abstand zwischen dem Punkt (x, y) und benachbarten Punkten ist. Die maximale Dichte wird durch den kleinsten gewünschten Abstand der Punkte d_{des} spezifiziert.

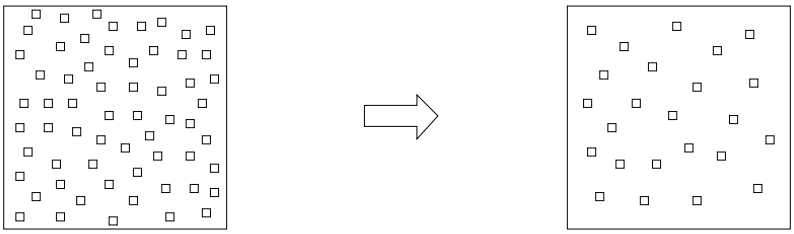


Abbildung 5.20: Die extrahierten Punkte sollten nicht zu dicht beieinander liegen.

5.4.2 Repräsentation

Um genetisches Programmieren zur Evolution eines Operators zur Extraktion markanter Punkte einzusetzen, müssen die Menge der Terminal-Symbole, die Menge der elementaren Funktionen und eine geeignete Fitneßfunktion definiert werden.

Terminal-Symbole

Als einziges Terminal-Symbol wurde das Eingabebild `Image` verwendet. Die Pixelwerte wurden auf den Bereich [0,1] skaliert.

Elementare Funktionen

Als elementare Funktionen wurden die folgenden unären und binären Funktionen verwendet. Dabei sei I_R das Bild, das bei der Anwendung der Funktion entsteht. Mit I ist das Eingabebild bei einer unären bzw. I_1 und I_2 bei binären Funktionen bezeichnet. Bildkoordinaten werden mit x und y bezeichnet.
Unäre Funktionen:

- Betrag (`Abs`) berechnet für jedes Pixel den Betrag.
  $I_R(x,y) = |I(x,y)|$
- Quadratwurzel (`Sqrt`) berechnet für jedes Pixel die Quadratwurzel.
  $I_R(x,y) = \sqrt{|I(x,y)|}$
- Quadrat (`Square`) quadriert alle Pixelwerte.
  $I_R(x,y) = I(x,y) \cdot I(x,y)$
- Gabor-Filter (`Gabor0`,...,`Gabor7`) faltet das Bild I mit einem Gabor-Filter.
  $I_R(x,y) = |\int \Psi(x',y',f,\theta_j) I(x-x', y-y') dx' dy'|$
  wobei $\Psi(x,y,f,\theta) = \exp(i(fx\cos\theta + fy\sin\theta) - \frac{f^2(x^2+y^2)}{2\sigma^2})$,
  $\sigma = \pi$, $f = \frac{\pi}{2}$, $\theta_j = \frac{\pi j}{8}$ und $j \in \{0,...,7\}$ (die Gabor-Filter sind wie in [170] definiert).
- Durchschnitt (`Avg3x3`) berechnet den durchschnittlichen Wert der Pixel in einem 3×3 großen Bereich.
  $I_R(x,y) = \frac{1}{9} \sum_{-1 \le i,j \le 2} I(x+i, y+j)$
- Median-Filter (`Median3x3`) berechnet den Median der Pixelwerte in einem 3×3 großen Bereich.
  $I_R(x,y) = \text{Median}\{I(x+i, y+j) | -1 \le i,j \le 1\}$

- Gauß-Operator (`Gauss`) glättet das Bild I.
  $I_R(x,y) = \int e^{\frac{x'^2+y'^2}{2\sigma^2}} I(x-x', x-y')dx'dy'$ wobei $\sigma = 1.0$.
- Ableitung des Gauß-Operators in x-Richtung (`GaussDx`) extrahiert Kanten in x-Richtung.
  $I_R(x,y) = \frac{1}{\sqrt{2\pi}\sigma^3} \int x e^{-\frac{1}{2\sigma^2}(x'^2+y'^2)} I(x-x', y-y')dx'dy'$ wobei $\sigma = 1.0$.
- Ableitung des Gauß-Operators in y-Richtung (`GaussDy`) extrahiert Kanten in y-Richtung.
  $I_R(x,y) = \frac{1}{\sqrt{2\pi}\sigma^3} \int y e^{-\frac{1}{2\sigma^2}(x'^2+y'^2)} I(x-x', y-y')dx'dy'$ wobei $\sigma = 1.0$.
- Shift nach links (`ShiftL`) verschiebt das Bild I um ein Pixel nach links.
  $I_R(x,y) = I(x+1, y)$
- Shift nach rechts (`ShiftR`) verschiebt das Bild I um ein Pixel nach rechts.
  $I_R(x,y) = I(x-1, y)$
- Shift nach oben (`ShiftU`) verschiebt das Bild I um ein Pixel nach oben.
  $I_R(x,y) = I(x, y+1)$
- Shift nach unten (`ShiftD`) verschiebt das Bild I um ein Pixel nach unten.
  $I_R(x,y) = I(x, y-1)$
- Nulldurchgänge (`ZeroCross`) berechnet die Nulldurchgänge im Bild I.
  $$I_R(x,y) = \begin{cases} \max\{I_x, I_y, 1\} & \text{falls} \quad I(x,y) \geq 0 \wedge (I(x,y+1) < 0 \vee I(x+1,y) < 0) \\ 0 & \text{falls} \quad I(x,y) \geq 0 \wedge (I(x,y+1) \geq 0 \wedge I(x+1,y) \geq 0) \\ \max\{-I_x, -I_y, 1\} & \text{falls} \quad I(x,y) < 0 \wedge (I(x,y+1) \geq 0 \vee I(x+1,y) \geq 0) \\ 0 & \text{falls} \quad I(x,y) < 0 \wedge (I(x,y+1) < 0 \wedge I(x+1,y) < 0) \end{cases}$$
  wobei $I_x = I(x,y) - I(x+1,y)$ und $I_y = I(x,y) - I(x,y+1)$.
- Erosion (`Erosion`) erodiert das Bild I.
  $I_R(x,y) = \min_{-1 \leq i,j \leq 1} I(x+i, y+j)$
- Dilation (`Dilation`) dilatiert das Bild I.
  $I_R(x,y) = \max_{-1 \leq i,j \leq 1} I(x+i, y+j)$

Binäre elementare Funktionen:

- Addition (`+`) addiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x,y) = I_1(x,y) + I_2(x,y)$
- Subtraktion (`-`) subtrahiert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x,y) = I_1(x,y) - I_2(x,y)$
- Multiplikation (`*`) multipliziert korrespondierende Pixel der Bilder I_1 und I_2.
  $I_R(x,y) = I_1(x,y) \cdot I_2(x,y)$
- Division (`/`) teilt jeden Pixelwert des Bildes I_1 durch den korrespondierenden Wert in I_2. Bei Division durch Null wird das Pixel auf den Wert Eins gesetzt.
  $$I_R(x,y) = \begin{cases} 1 & \text{falls} \quad I_2(x,y) = 0 \\ I_1(x,y)/I_2(x,y) & \text{sonst} \end{cases}$$

5.4.3 Fitneßfunktion

Die oben definierten Qualitätsmaße müssen in ein Maß für die Fitneß eines Individuums integriert werden. Die unterschiedlichen Qualitätsmaße gleichzeitig zu erfüllen, ist nicht einfach. Zur gleichzeitigen Optimierung unterschiedlicher, sich teilweise sogar widersprechender Maße wurden spezielle Verfahren die sogenannte *multi-objective Optimization*, entwickelt. Ein Überblick über die Verfahren wird von Fonseca und Fleming [102, 101] gegeben.

Um die Operatoren zur Extraktion markanter Punkte zu evolvieren, werden Bilder aus einer Bildsequenz verwendet. Für jedes Bild werden die Qualitätsmaße m_i ($i \in \{1, ..., 6\}$) berechnet. Werden insgesamt $n+1$ Bilder eingesetzt, so wird zunächst für jedes Qualitätsmaß der Mittelwert berechnet.

$$\bar{m}_i = \frac{1}{n} \sum_{j=1}^{n} m_i(t_j) \tag{5.69}$$

Danach werden die Qualitätskriterien normalisiert. So wird für jedes Kriterium k eine Selektionswahrscheinlichkeit $p_k(i)$ für das Individuum i berechnet.

$$p_k(i) = \frac{\bar{m}_k(i)}{\sum_j \bar{m}_k(j)} \tag{5.70}$$

Die Selektionswahrscheinlichkeiten können additiv oder multiplikativ kombiniert werden. Ein additiver Beitrag wurde mit dem von Schaffer und Grefenstette [265] bzw. Schaffer [264] entwickelten Verfahren realisiert. Dabei werden für jedes Kriterium unabhängig die Individuen selektiert. Die so selektierten Individuen werden dann wieder zu einer Population zusammengefügt. Dies entspricht einem additiven Betrag der einzelnen Selektionswahrscheinlichkeiten [102].

$$\text{Fitneß}(i) = \sum_k p_k(i) \tag{5.71}$$

Ein additiver Beitrag eines Gens zur Fitneß eines Individuums wurde auch von Kauffman [154] in seinem NK-Model verwendet, das den Einfluß der Kopplungsstärke zwischen den Genen untersucht. Als Resultat der Pareto-Optimierung erhält man in der Regel mehrere optimale Individuen, die auf einer Oberfläche verteilt sind. In den hier durchgeführten Experimenten stellt sich dann das praktische Problem, daß schließlich eines der Individuen aus dieser Pareto-optimalen Menge zur Extraktion der markanten Punkte ausgewählt werden muß.

Die Selektionswahrscheinlichkeiten können aber auch multiplikativ kombiniert werden. Dies entspricht einer Korrelation der Qualitätsmaße.

$$\text{Fitneß}(i) = \prod_k p_k(i) \tag{5.72}$$

In diesem Fall ist eine Normalisierung nicht notwendig. Zum Vergleich der Individuen wird die absolute Fitneß definiert.

$$\text{Fitneß}_a(i) = \prod_j \bar{m}_j(i) \tag{5.73}$$

In den hier beschriebenen Experimenten wurde die Selektionswahrscheinlichkeit eines Individuums multiplikativ berechnet. Experimente wurden auch mit einem additiven Beitrag durchgeführt, die jedoch keine so guten Ergebnisse lieferten.

Experiment	Elementare Funktionen
A	`Sqrt`, `Square`, `Gabor0`, ..., `Gabor7`, `Avg3`, `Median3`, `Gauss`, `GaussDx`, `GaussDy`, `+`, `-`, `*`, `/`
B	`Abs`, `ShiftL`, `ShiftR`, `ShiftU`, `ShiftD`, `ZeroCross`, `Erosion`, `Dilation`, `Gauss`, `GaussDx`, `GaussDy`, `+`, `-`, `*`, `/`

Tabelle 5.2: Übersicht über die bei den Experimenten verwendeten elementaren Funktionen zur Evolution eines Operators zur Berechnung des optischen Flusses.

5.4.4 Experimente

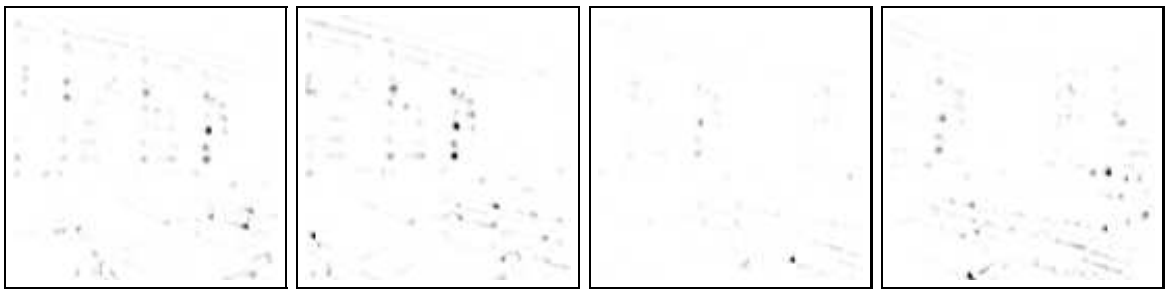
Bei den durchgeführten Experimenten wurde jeweils eine Sequenz, die aus vier Einzelbildern besteht, verwendet. Die einzelnen Bilder haben die Größe 128×128. Sie sind in Abbildung 5.21 zu sehen. Die Parameter wurden wie folgt gesetzt. Es wurde eine Populationsgröße von 500 Individuen verwendet. Die erste Generation wurde mit der ansteigenden 50:50 Regel mit Individuen der Tiefe 2 bis 6 gefüllt. Die Individuen wurden auf 1000 Knoten und auf eine maximale Tiefe von 17 limitiert. Es wurde Tournament-Selektion mit der Crossover-Wahrscheinlichkeit $p_{\text{cross}} = 0.85$, der Reproduktions-Wahrscheinlichkeit $p_{\text{rep}} = 0.1$ und der Mutations-Wahrscheinlichkeit $p_{\text{mut}} = 0.05$ verwendet. Die einzelnen Läufe wurden jeweils nach 50 Generationen abgebrochen. Bei den Experimenten wurden wieder unterschiedliche Teilmengen der elementaren Funktionen eingesetzt. Tabelle 5.2 gibt eine Übersicht über die verwendeten elementaren Funktionen. Der Schwellwert $\epsilon_1 = 0.02$ wurde zur Extraktion der markanten Punkte verwendet. Eine Korrespondenz wurde ab einem Wert $\epsilon_2 = 0.025$ hergestellt.

Bei Experiment A wurden die folgenden elementaren Funktionen eingesetzt: `Sqrt`, `Square`, `Gabor0`, ..., `Gabor7`, `Avg3`, `Median3`, `Gauss`, `GaussDx`, `GaussDy`, `+`, `-`, `*`, und `/`. Der Radius zur Berechnung der Glattheit des optischen Flusses wurde auf $\epsilon_3 = 8$ Pixel und der kleinste gewünschte Abstand der extrahierten Punkte wurde auf $d_{\text{des}} = 10$ gesetzt. Die Ergebnisse des Laufes sind in Abbildung 5.21 zu sehen. Die erste Reihe zeigt die Antwort des Operators. In der zweiten Reihe sind die vom Operator extrahierten Punkte zu sehen. Die dritte Reihe zeigt den optischen Fluß, der für die einzelnen Punkte berechnet wurde. Der Operator hebt die Punkte hervor, die sich besonders zur Berechnung des optischen Flusses eignen. Einige falsche Korrespondenzen sind auch vorhanden. Dies ergibt sich durch die Tatsache, daß mehrere Qualitätskriterien gleichzeitig erfüllt werden sollen, die sich auch widersprechen können. So sollen z.B. möglichst viele Punkte extrahiert werden und zugleich soll die Zuordnung der Punkte eindeutig sein.

Die Entwicklung der absoluten Fitneß über die Generationen ist in Abbildung 5.26 zu sehen. Die Entwicklung der einzelnen Qualitätskriterien ist ebenfalls in Abbildung 5.26 dargestellt. Das beste Individuum wurde auch auf einer neuen Bildsequenz getestet. Diese Bildsequenz ist in Abbildung 5.22 zu sehen. Die Ergebnisse, die bei dieser Bildsequenz erreicht wurden, sind ebenfalls in Abbildung 5.22 dargestellt. Der evolvierte Operator verwendet bis auf den Gauß-Operator alle vorhandenen elementaren Funktionen. Die Funktionen `/`, `GaussDy`, `Avg3`, `Gabor`, `Sqrt` und `Square` wurden mehrfach eingesetzt. Der beste Operator aus der ersten Generation des Laufes ist in Abbildung 5.23 gezeigt. Dieser Operator verwen-

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:

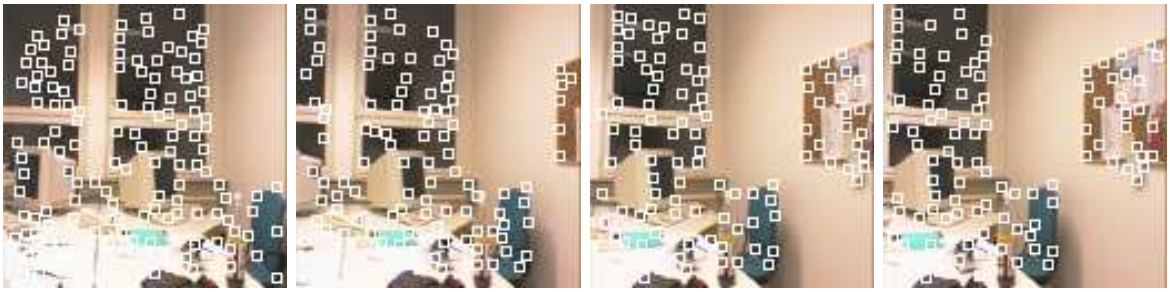
Der optische Fluß:

Abbildung 5.21: Bestes Individuum aus Generation 50 von Experiment A. In der ersten Reihe sind die Eingabebilder zu sehen. Die zweite Reihe zeigt die Antwort des evolvierten Operators. In der dritten Reihe sind die vom Operator extrahierten markanten Punkte zu sehen. Die vierte Reihe zeigt den optischen Fluß, der für die einzelnen Punkte zweier aufeinanderfolgender Bilder berechnet wurde. Punkte, für die kein Flußvektor berechnet werden konnte, sind durch einen Kreis markiert.

Eingabebilder:

Antwort des Operators:

Die vom Operator extrahierte Punkte:



Der optische Fluß:

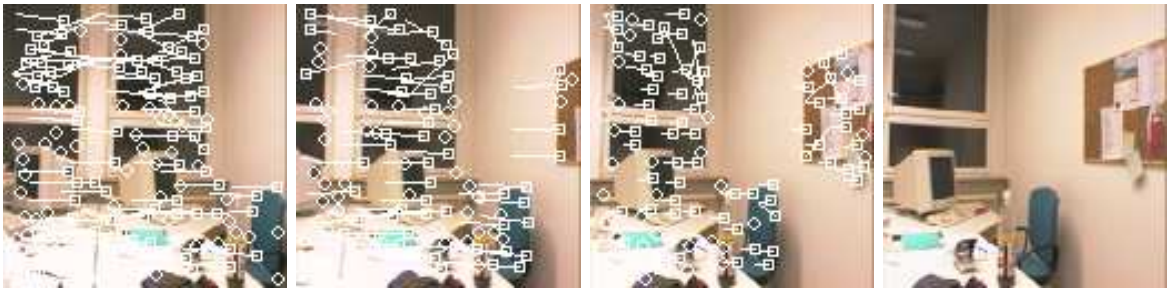
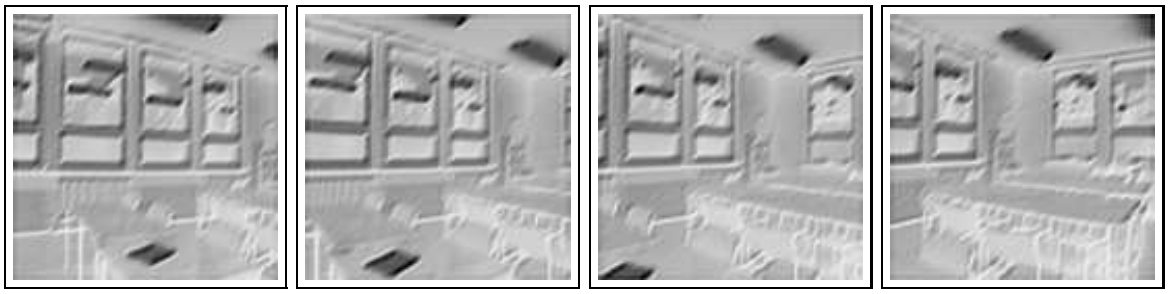


Abbildung 5.22: Ergebnisse des besten Individuums aus Generation 50 von Experiment A auf einer neuen Bildsequenz.

Eingabebilder:

Antwort des Operators:



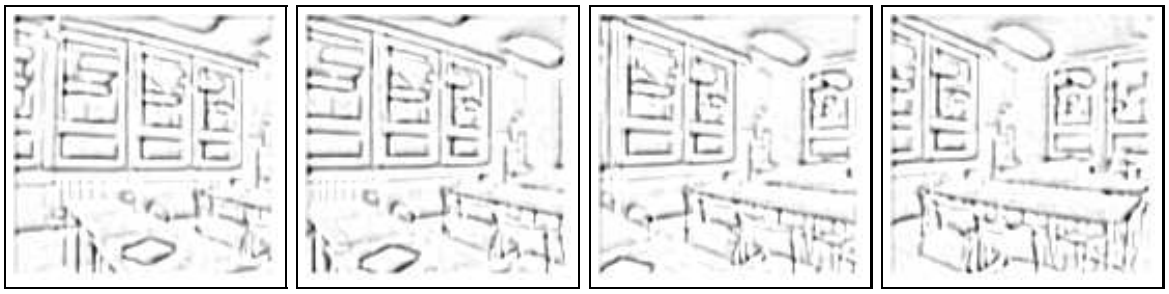
Vom Operator extrahierte Punkte:

Der optische Fluß:

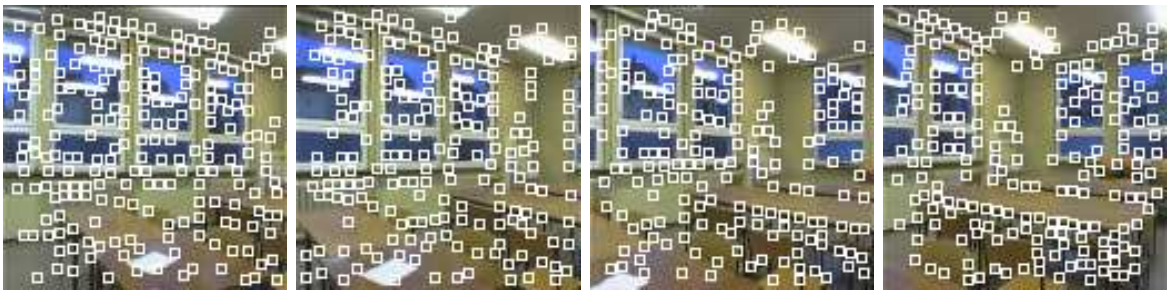
Abbildung 5.23: Bestes Individuum der ersten Generation von Experiment A.

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:



Der optische Fluß:

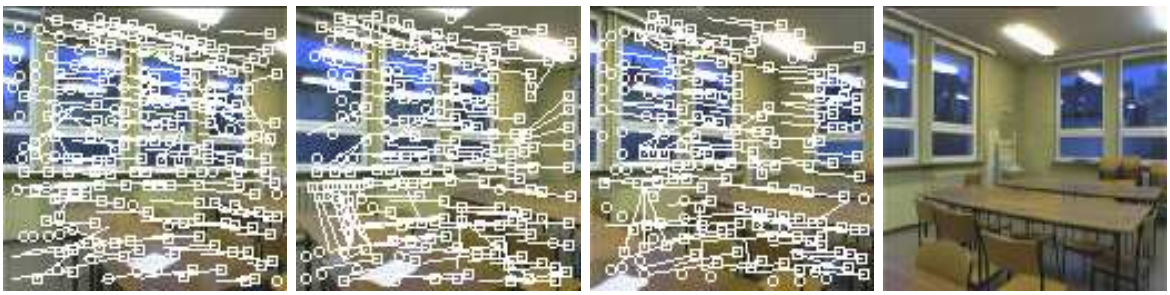
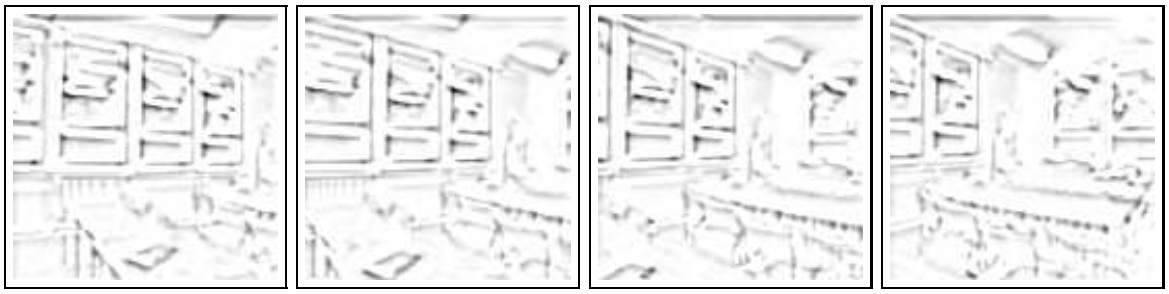


Abbildung 5.24: Bestes Individuum aus Generation 50 von Experiment B1.

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:

Der optische Fluß:

Abbildung 5.25: Bestes Individuum aus Generation 50 von Experiment B2.

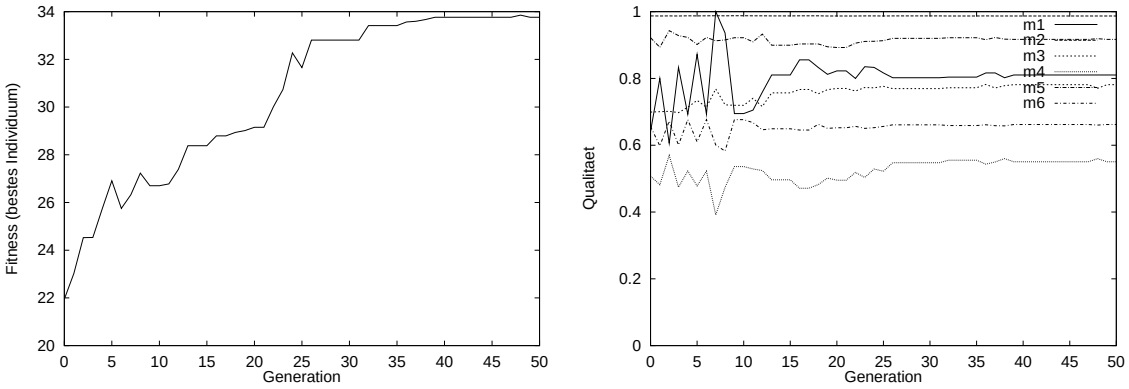


Abbildung 5.26: Fitneß-Statistik von Experiment A zur Evolution eines Operators zur Berechnung des optischen Flusses. Das linke Diagramm zeigt für jede Generation die beste absolute Fitneß. Im rechten Diagramm sind die einzelnen Qualitätskriterien, die zu dieser Fitneß gehören, aufgetragen. Das erste Qualitätsmaß (m_1) wurde auf den Bereich [0,1] normalisiert, um es in dasselbe Diagramm zu integrieren, das auch die anderen Kriterien zeigt.

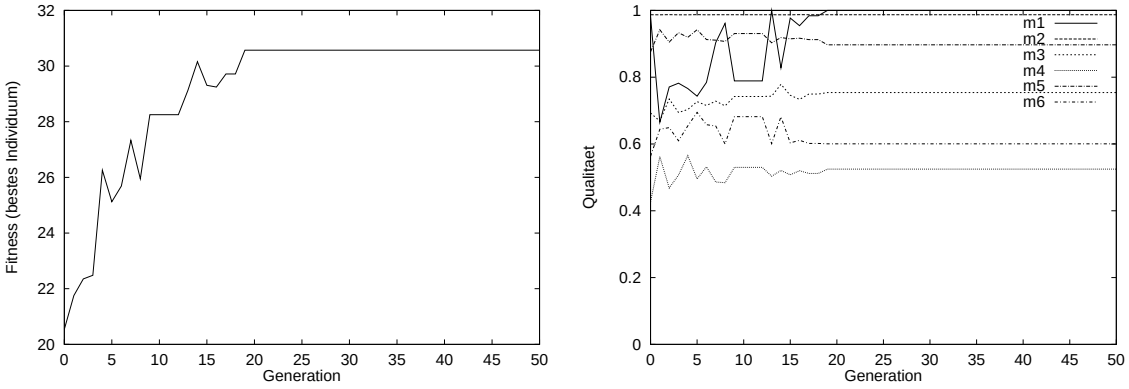


Abbildung 5.27: Fitneß-Statistik des Experimentes B1 zur Evolution eines Operators zur Berechnung des optischen Flusses.

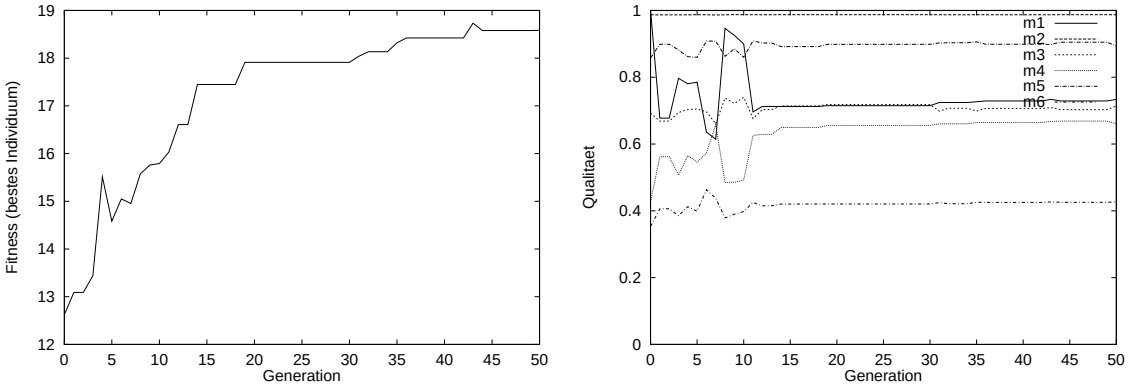


Abbildung 5.28: Fitneß-Statistik des Experimentes B2 zur Evolution eines Operators zur Berechnung des optischen Flusses.

Name des Operators	$\bar{m}_1$	$\bar{m}_2$	$\bar{m}_3$	$\bar{m}_4$	$\bar{m}_5$	$\bar{m}_6$	Fitneß$_a$	Zeit [s]
Kitchen-Rosenfeld	102.67	0.9867	0.5697	0.4845	0.8999	0.5843	14.70	1.077
Det(H_I)	109.67	0.9864	0.6625	0.4615	0.9121	0.6252	18.86	1.070
Moravec	64.67	0.9849	0.5455	0.6067	0.9320	0.7258	14.26	0.086
Det(M_A)/Spur(M_A)	62.00	0.9871	0.5753	0.7027	0.9767	0.7478	18.07	1.102
SUSAN	74.00	0.9862	0.5088	0.5517	0.9243	0.5732	10.85	0.261
Gabor-Filter	64.00	0.9865	0.5967	0.5803	0.9348	0.6891	14.08	37.281
Evolviert A	131.0	0.9871	0.7814	0.5504	0.9170	0.6620	33.77	7.533
Evolviert B1	145.3	0.9870	0.7540	0.5247	0.8970	0.6006	30.57	2.844
Evolviert B2	104.7	0.9871	0.7142	0.6608	0.8935	0.4266	18.58	5.245

Tabelle 5.3: Vergleich des evolvierten Operators mit existierenden Operatoren. Hierbei ist zu beachten, daß die Fitneß des evolvierten Operators des Experiments B2 eigentlich nicht mit der Fitneß der anderen Operatoren verglichen werden kann, da die Parameter ϵ_3 und d_{des} in diesem Experiment auf einen anderen Wert gesetzt wurden. Die existierenden Operatoren wurden außer durch die Wahl eines günstigen Skalierungsfaktors nicht optimiert. In der letzten Spalte ist die Zeit angegeben, die zur Anwendung des jeweiligen Operators benötigt wird. Zur Berechnung wurde ein Pentium PC mit 133 MHz eingesetzt.

det den `Gabor2`-Filter zweimal nacheinander auf das Eingabebild an. Er extrahiert Kanten, die in die Richtung $\frac{\pi}{4}$ orientiert sind.

Bei Experiment B wurden die folgenden elementaren Funktionen eingesetzt: `Abs`, `ShiftL`, `ShiftR`, `ShiftU`, `ShiftD`, `ZeroCross`, `Erosion`, `Dilation`, `Gauss`, `GaussDx`, `GaussDy`, `+`, `-`, `*` und `/`. Der Radius zur Berechnung der Glattheit des optischen Flusses wurde auf $\epsilon_3 = 8$ Pixel und der kleinste gewünschte Abstand der extrahierten Punkte wurde auf $d_{\text{des}} = 10$ gesetzt. Die Ergebnisse dieses Experimentes (Individuum B1) sind in Abbildung 5.24 zu sehen. Die Entwicklung der absoluten Fitneß über die Generationen ist in Abbildung 5.27 dargestellt. Der evolvierte Operator verwendet bis auf die Funktionen `Erosion` und `ShiftR` alle vorhandenen elementaren Funktionen.

In einem weiteren Experiment (B2) wurde der Radius zur Berechnung der Glattheit des optischen Flusses auf $\epsilon_3 = 16$ Pixel und der kleinste gewünschte Abstand der extrahierten Punkte auf $d_{\text{des}} = 16$ gesetzt. Die Ausgabe des dabei entstandenen Individuums (B2) ist in Abbildung 5.25 dargestellt. Wie man deutlich sieht, hat sich der Abstand zwischen den extrahierten Punkten gegenüber dem vorangegangenen Experiment vergrößert. Der evolvierte Operator verwendet bis auf die Funktionen `ShiftR`, `Erosion`, `Dilation` und `*` alle vorhandenen elementaren Funktionen. Der qualitative Eindruck, den man beim Betrachten der Antwort dieses Operators erhält, ist, daß er dem zuvor evolvierten sehr ähnlich ist. Die Entwicklung der absoluten Fitneß ist in Abbildung 5.28 zu sehen.

5.4.5 Vergleich der evolvierten Operatoren mit existierenden Operatoren

Die evolvierten Operatoren zur Extraktion markanter Punkte wurden mit bereits existierenden Operatoren verglichen. Tabelle 5.3 zeigt die Qualität der evolvierten Operatoren im Vergleich zum Kitchen-Rosenfeld-Eckendetektor [274], zur Determinante der Hesse-Matrix [274, 190], zum Moravec-Operator [205], zur Determinante der Autokorrelationsfunktion [118, 56], zum SUSAN-Merkmalsoperator [280] und zur Extraktion markanter Punkte mit Gabor-Filtern [330]. Die Funktionsweise dieser Operatoren wurde bereits ausführlich in Ab-

schnitt 5.1 beschrieben. Die beiden Operatoren mit den höchsten Fitneßwerten (die Determinante der Hesse-Matrix und die Determinante der Autokorrelationsfunktion) sind zum Vergleich in Abbildung 5.29 und 5.30 dargestellt. Die Ergebnisse, die mit den restlichen Operatoren erreicht wurden, sind in den Abbildungen B.1, B.2, B.3 und B.4 im Anhang dargestellt. Die Qualitätswerte der existierenden Operatoren konnten gegenüber den in [87] angegebenen Werten noch verbessert werden. Dies wurde durch Skalierung der Operatoren erreicht. Dadurch wurden mehr Punkte extrahiert, was zu einer verbesserten Fitneß führte. Durch einen Programmfehler bei der Implementierung des SUSAN-Operators, ergab sich zunächst ein besserer Fitneßwert. Durch die Korrektur ergibt sich der hier berichtete Wert. Die evolvierten Operatoren sind im Bezug auf einige der Qualitätsmaße (m_1, m_2 und m_3) deutlich besser als oder fast gleichwertig wie die existierenden Operatoren. Jedoch gibt es auch Qualitätsmaße, für die die existierenden Operatoren besser sind (m_4, m_5, m_6). Hierbei ist zu bedenken, daß die Selektion aufgrund der gesamten Fitneß des Individuums durchgeführt wird. Im Bezug auf die absolute Fitneß sind die evolvierten Operatoren deutlich besser als die existierenden Operatoren.

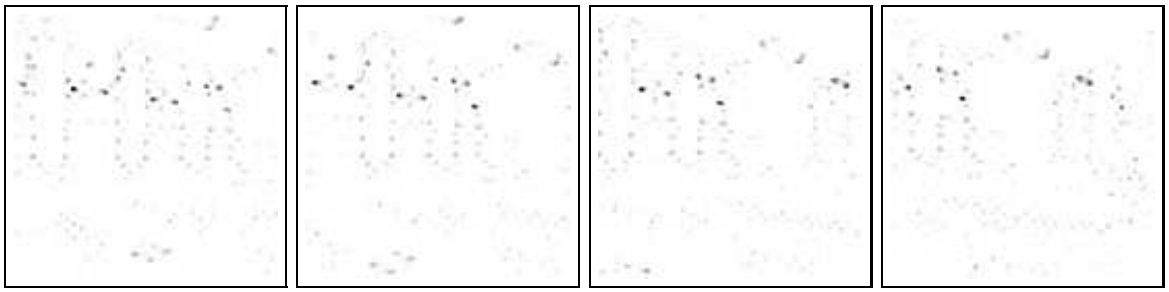
5.4.6 Zusammenfassung

Die durchgeführten Experimente zeigen, daß mit genetischem Programmieren ein Operator für eine gegebene Aufgabe optimiert werden kann. Um dies zu zeigen, wurde ein Operator zur Berechnung des optischen Flusses evolviert. Dafür wurden speziell für diese Aufgabe eine Reihe von Qualitätskriterien definiert: 1) Die Zahl der Flußvektoren sollte groß sein. 2) Die Qualität der Korrespondenzen sollte gut sein. 3) Das Verhältnis zwischen Zahl der Flußvektoren und Zahl der extrahierten Punkte sollte groß sein. 4) Die Zuordnung der Punkte sollte eindeutig sein. 5) Das Flußfeld sollte möglichst glatt sein. 6) Das Flußfeld sollte eine maximale Dichte besitzen. Der Operator sollte diese Kriterien gleichzeitig erfüllen. Mit den so definierten Kriterien wurden Operatoren evolviert, die Punkte aus einem Bild extrahieren, die sich zur Berechnung des optischen Flusses eignen.

Mit zunehmender Rechenleistung könnte es in Zukunft möglich sein, mit dem hier präsentierten Verfahren ein adaptives Bildverarbeitungssystem aufzubauen. Ein solches System würde sich selbständig an die gegebenen Randbedingungen der Umgebung anpassen. Genau wie sich der Durchmesser der Pupille an die aktuellen Lichtverhältnisse anpaßt, so könnten in einem adaptiven Bildverarbeitungssystemen kontinuierlich Operatoren evolviert werden, die optimal oder annähernd optimal arbeiten. Derzeit werden jedoch für die Evolution eines Operators, der sich zur Berechnung des optischen Flusses eignet, noch mehrere Tage Rechenzeit auf einem handelsüblichen PC mit Pentium Prozessor benötigt.

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:

Der optische Fluß:

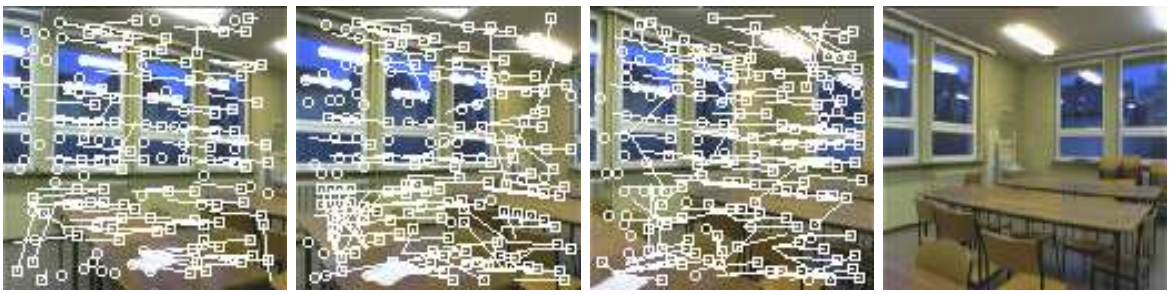
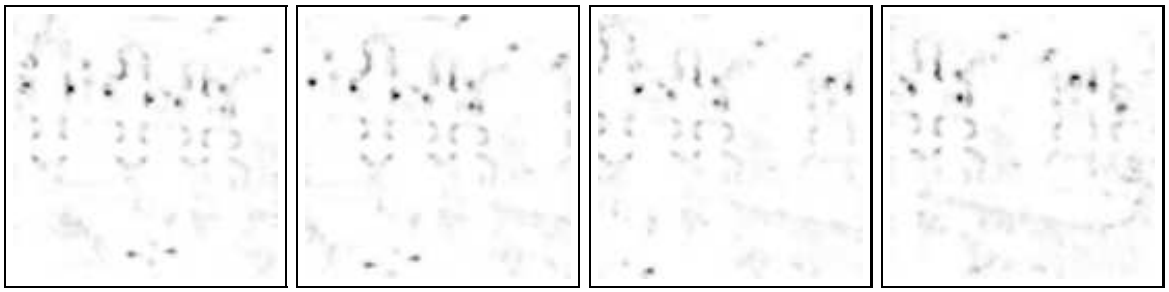


Abbildung 5.29: Determinante der Hesse-Matrix [274, 190].

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:

Der optische Fluß:

Abbildung 5.30: $\mathrm{Det}(M_A)/\mathrm{Spur}(M_A)$ [118, 56].

5.5 Verwandte Arbeiten

Im folgenden werden verwandte Arbeiten zur adaptiven Merkmalsextraktion aus Bildern besprochen.

5.5.1 Adaptive Kantendetektion

Von Jeong und Kim [145] wurden Arbeiten zur adaptiven Kantendetektion durchgeführt. Sie entwickelten eine Methode, um die Größe des Gauß-Operators, wie er in Kantendetektoren, wie z.B. dem Marr-Hildreth-Kantendetektor oder Canny's Kantendetektor eingesetzt wird, zu bestimmen. Die optimale Größe des Gaußschen Filters wurde für jedes Pixel einzeln, durch Minimierung einer Energie-Funktion, berechnet. Die Energie-Funktion wurde so definiert, daß der Filter in Bereichen mit gleichförmiger Intensität groß ist und in Bereichen, in denen sich die Intensitätswerte der Pixel sehr stark verändern, klein ist. Außerdem sollte sich die Größe des Filters nicht abrupt von einem Pixel zum nächsten verändern.

5.5.2 Visuelle Merkmalsdetektion mit neuronalen Netzen

In einer Reihe von Arbeiten wurden neuronale Netze zur Kantendetektion eingesetzt bzw. deren Eigenschaften analysiert. Linsker [181, 179, 180, 182] zeigte, daß sich die einzelnen Schichten eines vierschichtigen neuronalen Netzes unter Verwendung der Hebbschen Lernregel und weißem Rauschen als Eingangssignal wie folgt spezialisieren. Die zweite Schicht führt eine Glättung des Bildes durch. In der dritten Schicht entstehen sogenannte ON-OFF-Neurone und in der vierten Schicht entstehen Detektoren, die sich auf orientierte Kanten spezialisiert haben. Der Schwerpunkt der Arbeiten Linskers lag in der Selbstorganisation des Netzwerkes. Eine gewünschte Ausgabe wurde nicht vorgegeben. Barrow [19] modellierte die Signalverarbeitung der Retina und des seitlichen Kniehöckers jeweils als eine Faltung mit der Differenz zweier Gauß-Operatoren. Die Funktion des primären visuellen Kortex modellierte Barrow mit einem einschichtigen neuronalen Netz. Das Netz wurde mit digitalisierten Bildern und einer kompetitiven Hebbschen Lernregel trainiert. Nach dem Training hatten die Gewichte die Form von Masken, die sich zur Detektion orientierter Kanten eignen. Rubner und Schulten [257] zeigten, daß ein zweischichtiges neuronales Netz, bei dem die Ausgabeneuronen hierarchisch angeordnet und durch zusätzliche Gewichte miteinander verbunden sind, die Hauptkomponenten der Eingabedaten bestimmt. Die Gewichte zwischen der Eingabe- und der Ausgabeschicht wurden mit der Hebbschen Lernregel, die zwischen Neuronen der Ausgabeschicht mit einer anti-Hebbschen Lernregel trainiert. Das neuronale Netz könnte ein Modell der Selbstorganisation der Neuronen des visuellen Systems sein. Rubner und Schulten zeigten, daß das neuronale Netz bei biologisch plausiblen Eingabemustern Detektoren entwickelt, wie sie im visuellen System vorkommen. Joshi und Lee [149] erstellten ein relativ einfaches neuronales Modell der Retina. Dabei repräsentieren die Neuronen der ersten Schicht die Photorezeptoren, die Neuronen der zweiten Schicht die Bipolarzellen und die Neuronen der dritten Schicht die Ganglionzellen. Dieses Netz trainierten Joshi und Lee mit Backpropagation, um das Verhalten von ON-OFF-Neuronen zu reproduzieren. Die Gewichte des Netzes approximierten den Laplace-Gauß-Operator. Damit zeigten Joshi und Lee, daß mit Backpropagation Marr's Operator [193] erlernt werden kann [150].

Spreeuwers [282] setzte mehrschichtige neuronale Netze zur Wiederherstellung verrauschter oder unscharfer Bilder und zur Kantendetektion ein. Die Netzwerke wurden mit Back-

propagation trainiert. Zum Training der Filter für die Wiederherstellung verrauschter oder unscharfer Bilder wurden die Originalbilder durch Hinzufügen von Rauschen bzw. durch Glättung bearbeitet und als Eingabe des Netzwerkes verwendet. Spreeuwers entwickelte ein Maß für die Qualität eines Filters, das die Wahrscheinlichkeit eines Fehlers bei Anwendung des Filters und die dadurch entstehenden Kosten berücksichtigt. Ein solches Maß wurde zum Training von Kantendetektoren mit einem entsprechend angepaßtem Backpropagation-Algorithmus eingesetzt. Srinivasan et al. [285] trainierten ein zweischichtiges neuronales Netz mit synthetischen Daten. Die erste Schicht komprimierte die Daten des rezeptiven Feldes. In der zweiten Schicht wurde ein Kantenvektor aus den komprimierten Daten berechnet. Die erste Schicht wurde mit einer kompetitiven Hebbschen Lernregel trainiert. Dabei sollte die Varianz der Ausgabe über alle Eingabemuster maximiert werden. In der zweiten Schicht wurde Backpropagation zum Training der Gewichte eingesetzt. Die Qualität des trainierten neuronalen Netzes ist vergleichbar mit der des Canny-Kantendetektors. Chao und Dhawan [47] entwickelten ein Verfahren zur Kantenextraktion, basierend auf einem Hopfield-Netz. Das Netzwerk besaß für jedes Pixel genau ein Neuron. Die Neuronen waren in einer kleinen Umgebung miteinander verbunden. Die Gewichte des Netzes waren abhängig vom Kontrast und der Entfernung der Neurone die sie verbinden. Die Aktivitätsänderung der Neurone wurde durch eine dynamische Gleichung beschrieben, mit der die Energiefunktion des Netzes optimiert wurde. Nachdem das Netz konvergierte, waren Pixel, an denen eine Kante detektiert wurde, von den anderen getrennt.

Hussain und Kabuka [138] setzten ein neuronales Netz zur Zeichenerkennung ein. Das Netz basierte auf der translationsinvarianten Erkennung von Teilmustern des vollständigen Zeichens. Für jedes zu lernende Muster wurde ein dem Muster entsprechender Gewichtsvektor definiert. Kröner und Schulz-Mirbach [168] berechneten mit einem neuronalen Netz invariante Merkmale, die zur Bilderkennung genutzt wurden. Die Merkmale waren invariant gegenüber Translationen und Drehungen von 90 Grad. Die Invarianz wurde durch Mittelung über alle möglichen Transformationen erreicht [269]. Die Parameter des Netzes wurden durch Minimierung des Klassifikationsfehlers bestimmt. Lampinen und Oja [170] entwickelten ein System zur Mustererkennung, das tolerant gegenüber Verformungen ist. Zuerst wurden aus den Eingabebildern Merkmale mit Hilfe von Gabor-Filtern mit unterschiedlichen Auflösungen extrahiert. Danach wurden die Merkmalsvektoren mit einer zweistufigen selbstorganisierenden Karte klassifiziert. Schließlich wurden die transformierten Bilder mit Hilfe eines histogrammbasierten Ansatzes klassifiziert. Kohonen [162] zeigte, daß bei einer speziellen Form der selbstorganisierenden Karte, der sogenannten *adaptive-subspace self-organizing Map* invariante Merkmalsdetektoren entstehen können. Bei Verschiebung des Eingabemusters entstanden Gabor-ähnliche Filter. Eine Drehung des Eingabemusters bzw. eine Vergrößerung erzeugte Filter, die sensitiv für rotatorischen optischen Fluß sind, bzw. Filter die auf radialen optischen Fluß ansprechen.

5.5.3 Evolutionäre Algorithmen in der Bildverarbeitung

Bhattacharjya und Roysam [24] verwendeten eine Mischung aus Gradientenabstiegsverfahren und *Evolutionary Programming* für die modellbasierte Objekterkennung. Sie definierten eine Bewertungsfunktion, die gleichzeitig Probleme der niederen, mittleren und höheren Bildverarbeitung lösen sollte. So wurden selbst bei geringen Signal/Rausch-Verhältnissen durch Optimierung der Bewertungsfunktion die Pixelintensitäten ermittelt, die Bilder segmentiert, die Positionen der Objekte ermittelt und die Objekte klassifiziert.

Lohmann [185, 186, 187, 184] setzte *Structure Evolution* ein, um die Struktur und Parameter von Filtern zu evolvieren. *Structure Evolution* ist eine geschachtelte Evolutionsstrategie [247], bei der die Struktur in der äußeren und die Parameter in der inneren Schleife evolviert werden. Lohmann evolvierte einen Filter, um die in einem Bild vorhandenen Objekte zu zählen. Der evolvierte Filter berechnet die Eulerzahl des Bildes.

Bei einer Reihe von Arbeiten wurden genetische Algorithmen in der Bildverarbeitung eingesetzt. Rizki et al. [254] setzten *Evolutionary Programming* und genetische Algorithmen zur Zeichenerkennung ein. Die Zeichen wurden als erstes mit einer *Closing*-Operation mit verschiedenen Strukturelementen auf unterschiedlichen Auflösungsstufen vorverarbeitet. Anschließend wurden die transformierten Zeichen durch Anwendung einer morphologischen Operation mit evolvierten Strukturelementen bearbeitet. Die Zeichen wurden je nach Impulsantwort dieser Filteroperation klassifiziert. Roth und Levine [256] extrahierten mit einem genetischen Algorithmus aus digitalisierten Bildern geometrische Objekte. Die Objekte mußten durch eine implizite Gleichung der Form $f(\bar{x}, \bar{a}) = 0$ beschreibbar sein. Dabei ist $\bar{x}$ ein Objektpunkt und $\bar{a}$ ist der Parametervektor, der das Objekt beschreibt. Roth und Levine extrahierten so Kreise, Ellipsen, Flächen und Kugeln. Katz und Thrift [153] evolvierten lineare Filter um gesuchte Objekte aus Bildern herauszufiltern. Für jede extrahierte Region wurde mit Hilfe weiterer evolvierter Filter ein Merkmalsvektor berechnet. Dieser Merkmalsvektor wurde dann von einem konventionellen Klassifikator weiterverarbeitet. Bhandarkar et al. [23] setzten genetische Algorithmen zur Kantendetektion ein. Evolviert wurde dabei das gesamte resultierende Kantenbild. Zur Bewertung der Kantenbilder definierten Bhandarkar et al. eine Reihe von Kostenfaktoren, basierend auf den Differenzen der Grauwerte zwischen den Regionen, die die Kanten trennen, der Dicke, der Krümmung, der Fragmentierung und der Zahl der Kanten. Huang et al. [134] verwendeten eine Kombination aus Schätzer und genetischem Algorithmus, um ein Flußfeld in zusammengehörige Bewegungen und Ausreißer zu segmentieren und die dreidimensionale Bewegung der zugehörigen Objekte zu berechnen. Brooks et al. [32] kalibrierten mit Hilfe eines genetischen Algorithmus verrauschte Sensoren. Aufgabe war es, die Verschiebung in horizontaler und vertikaler Richtung sowie Drehung zu finden, die zwei verrauschte Grauwertbilder zur Deckung bringen.

Auch genetisches Programmieren wurde bereits in einigen Arbeiten eingesetzt. Tackett [288, 287] setzte genetisches Programmieren zur Klassifikation visueller Merkmale ein. Zunächst wurden mit einem Filter einzelne Regionen extrahiert, die durch das System weiterbearbeitet wurden. Für die Regionen wurden primitive Merkmale, wie z.B. die durchschnittliche Intensität der Bildpunkte und die Standardabweichung, berechnet. Momenten- und intensitätsbasierte Merkmale wurden für segmentierte Regionen berechnet. Evolviert wurde ein symbolischer Ausdruck, der die einzelnen Regionen klassifiziert, wobei die berechneten Merkmale als Terminal-Symbole verwendet wurden. Koza [167] setzte genetisches Programmieren zur Zeichendetektion ein und evolvierte einen Detektor, um die Zeichen "I" und "L" zu erkennen. Die Detektoren bewegten sich über das binäre Eingabemuster, wobei sie jeweils die Pixelwerte in einem 3×3 Pixel großen Bereich um die aktuelle Position herum abfragen konnten. Andre [6] kombinierte genetisches Programmieren mit genetischen Algorithmen um Zeichendetektoren zu evolvieren. Wie bei Koza [167] bewegten sich die Detektoren über das Eingabemuster. Zum Vergleich wurden jedoch zweidimensionale Muster eingesetzt, anstatt die Pixel einzeln abzufragen. Die Muster wurden mit einem genetischen Algorithmus evolviert, der Entscheidungsbaum mit genetischem Programmieren. Der Crossover-Operator des genetischen Algorithmus vertauschte einen rechteckigen Bereich zweier Muster. Johnson et al. [147] evolvierten mit genetischem Programmieren Ullman's visuelle Routinen [306]. Aufgabe

war es, in Bildern, die die Silhouette einer Person zeigt, die Position der rechten und linken Hand zu bestimmen. Als Eingabe wurden binarisierte Bilder einer Person verwendet. Die tatsächliche Position der Hand wurde für jedes Bild einzeln manuell spezifiziert. Poli [236, 237] setzte genetisches Programmieren zur Evolution von Filtern für die Bild-Segmentierung ein. Als Anwendung wählte er die Segmentierung des Gehirns und die Extraktion von Arterien in der medizinischen Bildverarbeitung. Die Bilder wurden zunächst mit verschiedenen Filtergrößen geglättet. Mit dieser Repäsentation wurde eine Funktion evolviert, die für jedes Pixel seine Zugehörigkeit zum Vordergrund bzw. Hintergrund berechnet. Daida et al. [63, 62] setzten genetisches Programmieren zur Detektion von Eiskämmen in Satellitenaufnahmen des arktischen Meeres ein. Aufgabe war es, einen Klassifikator zu evolvieren, der für jedes Pixel angibt, ob es sich um eine Stelle handelt, an der ein Grat im Eis vorhanden war oder nicht. Für diese Aufgabe existierte zuvor kein Algorithmus, der dieses Problem löste. Harris und Buxton [120] evolvierten mit genetischem Programmieren Kantendetektoren für eindimensionale Signale. Zur Evolution der Detektoren wurde eine Mischung aus synthetischen und echten Signalen eingesetzt. Optimiert wurde die Stärke der Impulsantwort, die Lokalisationsgenauigkeit und die Fähigkeit zur Rauschunterdrückung. Winkeler und Manjunath [321] setzten genetisches Programmieren zur Objektdetektion, speziell zur Detektion von Gesichtern, ein. Sie entwickelten zwei unterschiedliche Ansätze. Mit dem ersten sollten Bilder, die größennormierte Gesichter zeigen, von Bildern mit anderen Objekten unterschieden werden. Der zweite Ansatz arbeitet auf ganzen Bildern. Evolviert wurde hier eine Funktion, die für jeden Bildpunkt angibt, ob es sich um einem Punkt handelt, der zum Gesicht gehört. Poli und Cagnoni [238] evolvierten Algorithmen zur Visualisierung von Bildern in der medizinischen Bildverarbeitung. Evolviert wurde eine Funktion, die mehrere Bilder in ein Falschfarbenbild transformiert. Die Funktion berechnet für jeden Bildpunkt einen Index in eine Farbtabelle. Da keine gewünschte bzw. korrekte Ausgabe existierte (es ist lediglich eine subjektive Bewertung möglich), evolvierten Poli und Cagnoni die Algorithmen interaktiv. Der Anwender mußte entscheiden, welches Individuum die Tournament-Selektion gewinnt. Um das Verfahren zu beschleunigen, versuchten sie ein Modell des Anwenders (wiederum durch den Einsatz von genetischem Programmieren) zu evolvieren.

5.5.4 Adaptive Lösung des Korrespondenzproblems

Hopfield-Netze wurden vielfach eingesetzt, um korrespondierende Bildmerkmale zu finden. Nasrabadi und Choo [211] sowie Mousavi und Schalkoff [207] setzten Hopfield-Netze im Bereich der Stereobildverarbeitung ein. Von Thirumalai und Ahuja [295] und Chen et al. [48] wurden mit einem Hopfield-Netz Korrespondenzen zwischen Bildmerkmalen einer Bildsequenz hergestellt. Chen et al. [49] verwendeten ein neuronales Netz zur Analyse nicht-rigider Bewegungen. Reimann und Haken [248] definierten einen dynamischen selbstorganisierenden Prozeß zur Lösung des Korrespondenzproblems. Van Deemter und Mastebroek [308] setzten ein neuronales Netz zur Bestimmung korrespondierender Linienmerkmale ein. Die Experimente wurden jedoch nur auf simulierten Daten durchgeführt.

5.5.5 Adaptive Auswahl geeigneter Bildmerkmale

Eine Reihe von Arbeiten befaßten sich mit der adaptiven Auswahl geeigneter Merkmale. Shi und Tomasi [275, 276] argumentierten, daß geeignete Merkmale genau die Merkmale sind, für die das Programm, das die Korrespondenz herstellt, der sogenannte *Tracker*, am besten

funktioniert. In ihren Experimenten wurden als erstes Regionen extrahiert, die eine ausgeprägte Textur aufwiesen. Anschließend wurde für jedes Merkmal unter Verwendung eines affinen Modells der Bildbewegung ein Maß für die Verformung des Merkmals berechnet. Bei zu großen Verformungen sollte ein extrahiertes Merkmal nicht weiter verwendet werden. Lew et al. [178] entwickelten einen Ansatz zur adaptiven Auswahl geeigneter Merkmale um korrespondierende Punkte in Stereobildern zu finden. Als Merkmale wurden die Intensität eines Bildpunktes, der Gradient in horizontaler und vertikaler Richtung, die Stärke und Richtung des Gradienten, der Laplace-Gauß-Operator und die Krümmung eingesetzt. Der von Lew et al. entwickelte Algorithmus versucht, eine Teilmenge der Merkmale zu finden, die eine optimale Unterscheidung zwischen dem korrespondierenden Punkt und weiteren möglichen Punkten ermöglicht. Greiner und Isukapalli [115] bestimmten mit einem probabilistischen Gradientenaufstiegsverfahren eine Funktion, die aus einer Reihe von Merkmalen diejenigen auswählt, die sich zur Lokalisation eines Roboters eignen. Dabei wurde angenommen, daß dem Roboter alle Merkmale bekannt sind, die sich in der Umgebung des Roboters befinden können. Als Kriterien zur Auswahl geeigneter Merkmale wurden der vermutete Abstand des Roboters zu den Merkmalen und die Art der Merkmale verwendet. Die Umgebung nahm der Roboter über eine 360° Kamera wahr, aus der ein Streifen extrahiert wurde. Janabi-Sharifi und Wilson [144] definierten eine Reihe von Kriterien, um geeignete Merkmale zur Steuerung eines Manipulators zu extrahieren. Als Kriterien wurden unter anderem die Sichtbarkeit, die Auflösung, der Abstand der Merkmale, der Winkel der Merkmale zur Kamera und die Sensitivität der Merkmalspositionen auf Kamerabewegungen herangezogen. Dabei wurde angenommen, daß ein Modell der Umgebung und des zu bearbeitenden Teils existiert. Ferner wurde angenommen, daß der relative Pfad des Manipulators bekannt ist.

5.5.6 Unterschiede zu den existierenden Arbeiten

Die vorliegende Arbeit liefert einen Beitrag zum Einsatz evolutionärer Algorithmen in der Bildverarbeitung. Durch genetisches Programmieren ist es möglich, die Bildverarbeitung zu automatisieren. In diesem Fall wird die Aufgabenstellung auf die Definition einer geeigneten Fitneßfunktion zurückgeführt. Daher hebt sich der hier vorgestellte Ansatz zur adaptiven Merkmalsextraktion deutlich von den Arbeiten ab, die neuronale Netze zur Merkmalsextraktion einsetzten. Auch zu den Arbeiten, bei denen z.B. ein genetischer Algorithmus zur Optimierung eingesetzt wurde, existieren deutliche Unterschiede, da bei einem genetischen Algorithmus nur mit einer klar definierten Struktur gearbeitet wird. Während in einigen Arbeiten bereits genetisches Programmieren in der Bildverarbeitung eingesetzt wurde, lag hier der Schwerpunkt auf der adaptiven Extraktion von Bildmerkmalen, wie z.B. Kanten oder markante Punkte im Bild. Es wurde gezeigt, wie ein künstliches visuelles System aufgebaut werden kann, das je nach Umgebung und Algorithmus, der die Informationen anschließend verarbeitet, optimale bzw. annähernd optimale Merkmale extrahiert.

5.6 Zusammenfassung

Zur Extraktion von Bildmerkmalen existieren bereits eine Vielzahl von Operatoren. Die bekanntesten Operatoren wurden hier kurz beschrieben. In der Bildverarbeitung werden meist eine Reihe von bekannten Operatoren zur Lösung eines bestimmten Problems in geeigneter Weise miteinander kombiniert. Genetisches Programmieren bietet die Möglichkeit, diesen Prozeß zu automatisieren.

Mit genetischem Programmieren wurde ein adaptives Verfahren zur Merkmalsextraktion entwickelt. Als erstes wurde ein Kantendetektor evolviert, der den Canny-Kantendetektor approximiert. Danach wurde ein Operator zur Extraktion markanter Punkte evolviert, der den Moravec-Operator approximiert. In beiden Fällen wurde ein bereits existierender Operator zur Berechnung der Fitneß der Individuen eingesetzt. Schließlich wurde ein Operator evolviert, der zur Berechnung des optischen Flusses verwendet wurde. Um diesen Operator zu evolvieren, wurden eine Reihe von Qualitätskriterien definiert, die sich aus der Problemstellung ergaben.

Was ein markanter Punkt ist, hängt von der Umgebung und dem Verfahren ab, das die extrahierten Punkte anschließend verarbeitet. Mit zunehmender Rechenleistung könnte das hier vorgestellte Verfahren zur Evolution geeigneter Operatoren während eines Einsatzes verwendet werden. Genauso wie sich der Durchmesser der Pupille an die aktuellen Lichtverhältnisse anpaßt, so könnte das System die verwendeten Operatoren an die gegebenen Randbedingungen der Umgebung anpassen.

Kapitel 6

Visuelle Steuerung eines mobilen Roboters

Die extrahierten Merkmale können zur Steuerung eines mobilen Roboters eingesetzt werden. Um dies zu zeigen, wurde ein System zur Steuerung eines mobilen Roboters, das auf Eigenschaften des menschlichen visuellen Systems basiert, entwickelt. Erste Ergebnisse hierzu wurden in kompakter Form von Ebner und Zell [85] beschrieben. Eine ausführlichere Darstellung findet sich in Ebner und Zell [82]. Damit soll gezeigt werden, daß sich durch eine genaue Analyse des natürlichen visuellen Systems wichtige Erkenntnisse speziell in der Bildverarbeitung für mobile Roboter nutzen lassen. Es existieren bereits viele Beispiele, die gezeigt haben, daß biologische Systeme viele Anregungen zur Realisierung technischer Systeme liefern können [209]. Dies ist vor allem dadurch interessant, da das visuelle System des Menschen als Produkt der natürlichen Evolution viele Aufgaben optimal erfüllt. Wie stark der Mensch durch die Evolution geformt wurde, faßt Pinker [234] in eindrucksvoller Weise zusammen. Brooks [35] geht sogar soweit, einen humanoiden Roboter zu konstruieren. Bisher stehen zu biologischen Systemen vergleichbare Leistungen in den Bereichen Robotik und in der maschinellen Bildverarbeitung noch aus.

Der Roboter, mit dem das Verfahren realisiert wurde, ein Real World Interface B21 [244, 243], ist mit einer Kamera ausgestattet, über die er seine Umgebung wahrnimmt. Die Bilder der Kamera werden digitalisiert [30] und vom Algorithmus verarbeitet. Zunächst werden markante Punkte aus den Bildern extrahiert. Diese Punkte werden zur Berechnung des optischen Flusses, der durch die translatorische Bewegung der Kamera hervorgerufen wurde, eingesetzt. Dazu wird die bekannte Eigenbewegung des Roboters und der Kamera genutzt, um die rotatorische Bewegung der Kamera zu kompensieren. Eine Korrespondenz zwischen den Punkten des aktuellen Bildes und des vorangegangenen Bildes wird durch ein einfaches Fehlermaß hergestellt. Der optische Fluß wird anschließend in den komplex-logarithmischen Raum transformiert. Die Differenz zwischen dem optischen Fluß der linken und rechten periphären visuellen Bereiche wird zur Steuerung des Roboters verwendet. Die Vorwärtsgeschwindigkeit des Roboters wird innerhalb der minimalen und maximalen Geschwindigkeit so geregelt, daß der wahrgenommene optische Fluß konstant bleibt.

Im Unterkapitel 6.1 wird zunächst auf die Motivation für die Entwicklung des Systems eingegangen. Danach werden die wesentlichen Bestandteile des menschlichen visuellen Systems beschrieben. Ausgehend vom Auge und der Retina wird der Weg der visuellen Informationen über den seitlichen Kniehöcker bis zur Verarbeitung der Signale im visuellen Kortex

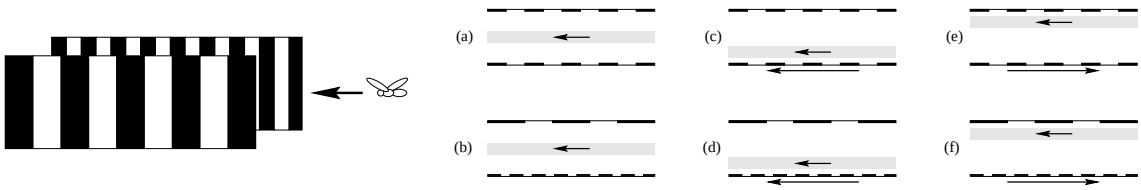


Abbildung 6.1: Flug einer Biene durch einen Korridor. In (a) und (b) bewegt sich die Biene genau in der Mitte des Korridors unabhändig vom Muster, das sich an der Wand befindet. In (c) und (d) wurde die linke Wand in Flugrichtung bewegt. Dadurch scheint die linke Wand weiter entfernt zu sein, wodurch sich die Flugbahn der Biene nach links verschiebt. In (e) und (f) wurde die linke Wand entgegengesetzt zur Flugrichtung bewegt. Dadurch scheint die linke Wand näher zu sein, als sie tatsächlich ist und die Biene fliegt etwas näher an der rechten Wand (nach Srinivasan [284, 283]).

dargestellt. Anschließend wird auf die komplex-logarithmische Abbildung und das Reafferenzprinzip eingegangen. Schließlich wird das komplette System, wie es mit dem mobilen Roboter realisiert wurde, beschrieben.

6.1 Motivation

Traditionelle Ansätze zur Steuerung eines mobilen Roboters versuchen vielfach, ein sehr genaues, dreidimensionales Modell der Umgebung zu konstruieren. Dieses Modell wird dann verwendet, um einen Pfad zu planen und dann den Roboter entlang der Punkte des Pfades zu bewegen. Dies entspricht der traditionellen Schleife, erst wahrnehmen, dann denken und schließlich handeln (*Sense-Think-Act*). In der Praxis haben sich reaktive, verhaltensbasierte Ansätze oft gegenüber den traditionellen Ansätzen als überlegen erwiesen.

Im besonderen ist es nicht erforderlich ein dreidimensionales Umweltmodell zu erstellen, um dieses dann auf eine einfache Zustandsgröße, wie z.B. die Orientierung innerhalb eines Korridors, zu reduzieren. Darauf hat Horswill [132] hingewiesen. Durch Ausnutzung verschiedener Randbedingungen ist es ihm gelungen, ein sehr erfolgreiches System zu konstruieren, das auf einer relativ einfachen visuellen Steuerung basiert. Aloimonos [3] prägte den Begriff der zielgerichteten Bildverarbeitung (*Purposive Vision*). Vor allem die aktive Bildverarbeitung dient nicht nur der Extraktion von Informationen, sondern dient auch einem ganz bestimmten Zweck, z.B. der Fortbewegung oder dem Greifen eines Objektes. Dieses Paradigma wird auch als *Animate Vision* bezeichnet [13]. Bei der zielgerichteten Bildverarbeitung werden lediglich die für die Aufgabe erforderlichen Daten berechnet und somit Rechenleistung eingespart. Dadurch kann schnell auf plötzliche Veränderungen reagiert werden.

Bekey [22] zeigte, daß eine biologisch inspirierte Steuerung erfolgreich sein kann, wo traditionelle Ansätze schwierig oder unmöglich zu realisieren sind. Sandini et al. [260] betonen die Wichtigkeit der Verarbeitung visueller Informationen während einer Bewegung. In diesem Zusammenhang kann man auch viel von der Funktionsweise des visuellen Systems von Insekten lernen [129, 131]. Ein Teil der Ansätze könnte auch auf das Gebiet der Bildverarbeitung mit mobilen Robotern übertragen werden [105].

Eine Reihe von Arbeiten, die sich mit der Steuerung mobiler Roboter befassen, wurden von Insekten, wie z.B. Bienen, inspiriert [55, 260, 262, 263, 261]. Bienen können aufgrund ihres geringen Augenabstandes nur sehr kleine Entfernungen durch Stereobildverarbeitung abschätzen. Sie verwenden die Bildbewegung zur Entfernungsabschätzung [176]. Die Ent-

fernung eines Objektes kann bei bekannter Eigenbewegung des Auges unter Verwendung der Richtung und der Winkelgeschwindigkeit des Objektes berechnet werden [283]. In einem Korridor versuchen Bienen, die retinale Geschwindigkeit der Bilder des linken und rechten Auges abzugleichen und bewegen sich dadurch genau in der Mitte des Korridors [284, 283]. Denn im Zentrum des Korridors ist die retinale Geschwindigkeit für beide Augen gleich groß. Bienen sind in der Lage, dieses Verhalten unabhängig vom Kontrast oder der Frequenz von Streifenmustern entlang des Korridors auszuführen (Abbildung 6.1).

Das visuelle System des Menschen unterscheidet sich deutlich vom visuellen System der Insekten. Eine Einführung in das visuelle System des Menschen wird von Tovée [303] gegeben. Die Augen sind nach vorne gerichtet und haben im Vergleich zu Insekten einen kleineren Sichtbereich [43]. Die Auflösung ist im Zentrum der Retina (Fovea) deutlich höher als an der Peripherie. Information über die Bewegung der Augen wird eingesetzt, um die Eigenbewegung der Augen zu kompensieren. Dieser Mechanismus (das Reafferenzprinzip [310]) wird eingesetzt, um die Umgebung als stationär wahrzunehmen, obwohl sich das Auge bewegt. Bewegt man das Auge passiv, indem man es mit seinem Finger anstößt, oder versucht man die Blickrichtung zu ändern, wenn das Auge mechanisch fixiert ist, dann scheint sich die Umwelt zu bewegen. Die Eigenbewegung des Kopfes wird durch den Vestibularapparat [43] ermittelt. Informationen des Vestibularapparates werden vermutlich eingesetzt, um die Qualität des wahrgenommenen Bildes zu verbessern. Dies läßt sich testen, indem man einmal die ausgestreckte Hand nach links und rechts bewegt und diese Wahrnehmung mit der einer ruhigen Hand bei bewegtem Kopf vergleicht [43]. Da das hier vorgestellte System stark durch das menschliche visuelle System inspiriert wurde, werden nun die wesentlichen Bestandteile des menschlichen Systems beschrieben.

6.2 Das visuelle System

Die im folgenden dargestellten Erkenntnisse über das visuelle System des Menschen stützen sich auf die Werke von Dowling [72], Zeki [327] und Tovée [303]. Eine kompakte Darstellung des visuellen Systems wird von Kleiner [159] und von Mallot [190] gegeben. Im folgenden werden die Hauptbestandteile des visuellen Systems in der Reihenfolge beschrieben, in der die visuellen Signale verarbeitet werden.

6.2.1 Das Auge

Aufgabe des Auges ist es, das Licht einzufangen und auf eine Schicht von Rezeptoren, die Retina, im Innern des Auges zu fokussieren. Die Struktur des menschlichen Auges ist in Abbildung 6.2 dargestellt. Das Auge besitzt 6 Muskeln, mit denen es bewegt werden kann. So können z.B. bewegte Signale mit dem Blick verfolgt werden. Da das Auge nur in einem relativ kleinen Bereich über ein hochauflösendes Sehvermögen verfügt, werden Augenbewegungen durchgeführt, mit denen der gesamte Sichtbereich untersucht wird. Diese schnellen, ballistischen Augenbewegungen werden Sakkaden genannt. Ein spezieller Mechanismus, genannt optokinetischer Nystagmus sorgt bei Bewegungen dafür, daß das Bild auf der Retina stabilisiert wird. Dabei wechseln sich die glatte Verfolgungsbewegung mit sakkadischen Rückstellbewegungen ab. Eine umfassende Darstellung der Physiologie von Augenbewegungen wird von Carpenter [43] gegeben. Die Cornea (Augenhornhaut) und die Linse fokussieren das einfallende Licht. An der Linse des Auges befinden sich Muskeln, über die die Brennweite

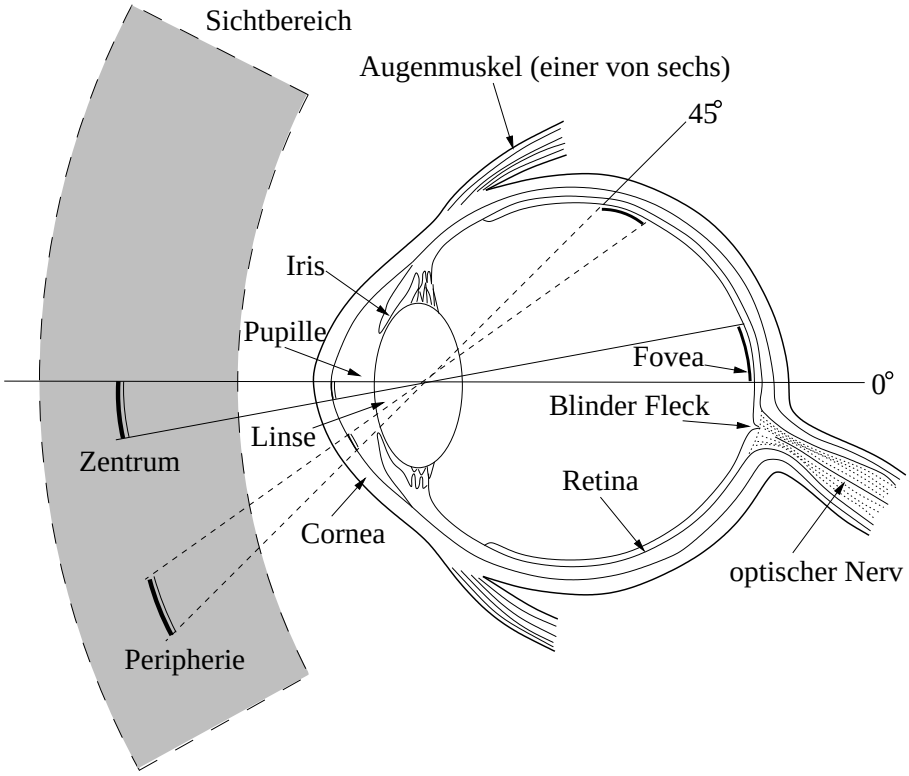


Abbildung 6.2: Das menschliche Auge im Querschnitt (nach Zeki [327], Tovée [303] und Mallot [190]).

der Linse reguliert werden kann. Die Menge des einfallendes Lichtes wird über den Durchmesser der Pupille durch einen Ring von Muskeln, genannt Iris reguliert. Das einfallende Licht erreicht schließlich die Retina. In der Retina befinden sich Rezeptoren, die auf Licht reagieren. Bereits in der Retina findet eine Informationsverarbeitung statt. Das Signal der Retina wird über den optischen Nerv an den seitlichen Kniehöcker weitergeleitet. An der Stelle, an der der optische Nerv aus dem Auge austritt, befinden sich keine Rezeptoren. Daher kann an diesem sogenannten blinden Punkt auch keine Information wahrgenommen werden. Das visuelle System füllt den blinden Punkt mit Information, so daß die Existenz des Punktes nicht bewußt wahrgenommen wird.

6.2.2 Die Retina

Die Retina besteht aus mehreren Schichten (Abbildung 6.3). In der ersten Schicht befinden sich die Fotorezeptoren. Sie befinden sich unter der Oberfläche. Daher muß das Licht zuerst durch die anderen Schichten hindurch, bevor es die Rezeptoren erreicht. Es gibt zwei Arten von Rezeptoren, stabförmige und zapfenförmige. Die Zapfen sind für das Farbsehen zuständig und sind in einem kleinen Bereich im Zentrum des Auges konzentriert. Es existieren drei Arten von Zapfen, die auf rotes, grünes bzw. blaues Licht, spezialisiert sind. Die Stäbchen sind für das Sehen bei geringen Intensitäten zuständig. In der zweiten Schicht befinden sich die Bipolarzellen, in der dritten die Ganglionzellen. Durch die Horizontal- und Amakrinzellen wird die visuelle Information auch in horizontaler Richtung weitergeleitet. Die Axone

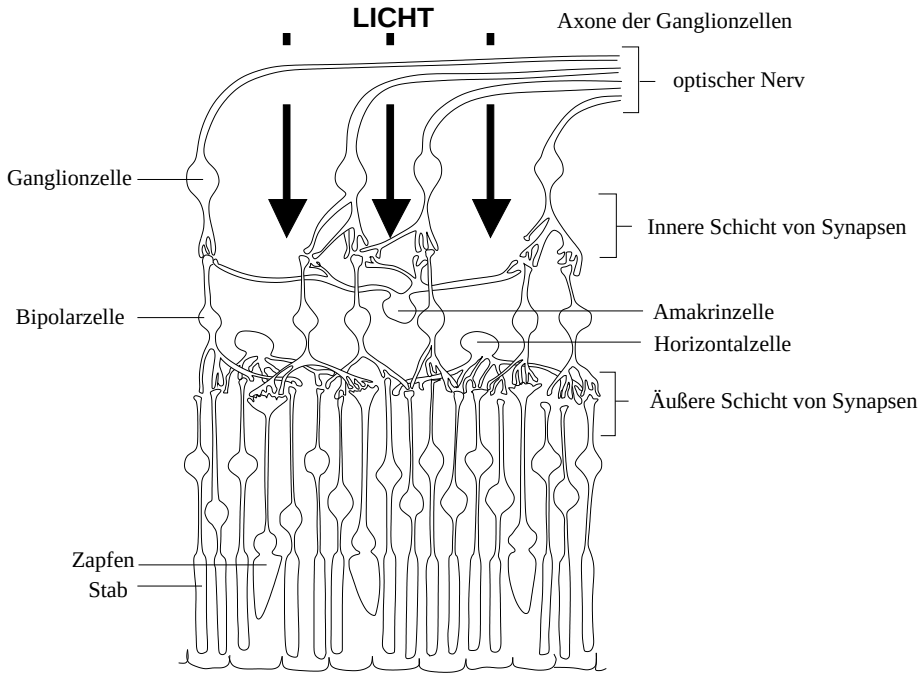


Abbildung 6.3: Struktur der Retina (nach Tovée [303]). Die Retina besteht aus mehreren Schichten. Das Licht trifft zuerst auf eine Schicht mit Ganglionzellen und eine Schicht mit Bipolarzellen, bevor es auf die Rezeptoren der Retina, die Stäbchen und die Zapfen, trifft.

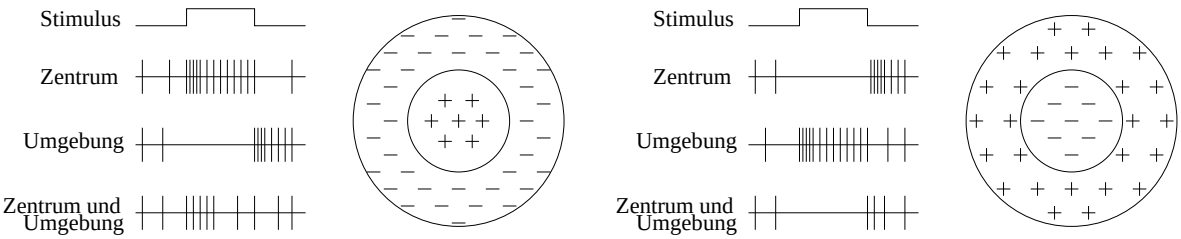


Abbildung 6.4: Arbeitsweise der *ON-Center* Zelle und *OFF-Center* Zelle (nach Dowling [72]). Die *ON-Center* Zelle wird durch Licht im Zentrum des rezeptiven Feldes aktiviert, während Licht im Randbereich des rezeptiven Feldes die Aktivität der Zelle hemmt. Die Zelle feuert solange der Stimulus präsentiert wird, wobei die Frequenz von der Intensität abhängt. Wird der Stimulus im Randbereich des rezeptiven Feldes der Zelle präsentiert, so feuert sie, nachdem der Stimulus ausgeschaltet wird. Eine Beleuchtung des gesamten rezeptiven Feldes aktiviert die Zelle kaum. Bei der *OFF-Center* Zelle ist der erregende und der hemmende Bereich vertauscht.

der Ganglionzellen bilden den optischen Nerv, über den die visuellen Signale den seitlichen Kniehöcker erreichen. In der Retina befinden sich mehr als 120 Millionen Stäbchen und 6 Millionen Zapfen, die auf das einfallende Licht reagieren, aber nur eine Million Ganglionzellen, die die Information aus dem Auge herausleiten. Das Verhältnis zwischen Rezeptoren der Retina und Ganglionzellen ist in der Fovea etwa eins zu eins, während es in der Peripherie mehrere hundert zu eins betragen kann.

Bereits in der Retina findet eine erste Verarbeitung der visuellen Informationen statt. Es gibt zwei Arten von Ganglionzellen. Die sogenannten M-Zellen reagieren auf Änderungen des Stimulus, während die P-Zellen für die gesamte Dauer des Stimulus aktiv sind. Die P-Zellen

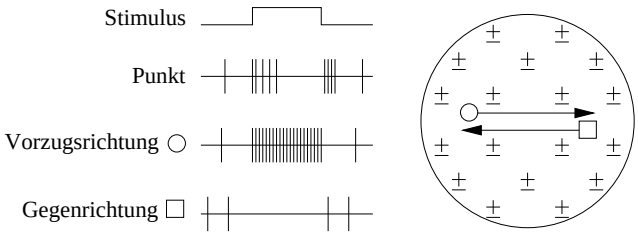


Abbildung 6.5: *ON-OFF* Zelle (nach Dowling [72]). Die *ON-OFF* Zelle reagiert, falls ein Lichtpunkt innerhalb des rezeptiven Feldes an oder ausgeschaltet wird (durch ± gekennzeichnet) oder falls sich ein Lichtpunkt in eine ganz bestimmte Richtung bewegt. Bewegt sich der Stimulus in der entgegengesetzten Richtung, dann wird die Aktivität der Zelle gehemmt.

sind auf bestimmte Wellenlängen spezialisiert und reagieren auf Signale mit hohem Kontrast. Sie besitzen eine sogenannte *Center-Surround*-Charakteristik. Der Sichtbereich, für den eine Zelle bei korrektem Stimulus aktiviert ist, wird rezeptives Feld der Zelle genannt. Es gibt zwei Arten von *Center-Surround*-Zellen (Abbildung 6.4). Die *ON-Center* Zellen reagieren auf Licht im Zentrum, während Licht im Randbereich des rezeptiven Feldes die Zellen hemmt. Bei den *OFF-Center* Zellen ist es genau umgekehrt. Daher sprechen die *ON-* bzw. *OFF-Center* Zellen auf starke Kontrastunterschiede zwischen Zentrum und Randbereich des rezeptiven Feldes an. In der Retina werden drei unterschiedliche Vergleichsmechanismen realisiert. Der erste vergleicht die Aktivität der roten und grünen Zapfen, der zweite vergleicht die Aktivität die blauen Zapfen mit der Summe der roten und grünen Zapfen, also mit Gelb und schließlich existiert noch ein achromatischer Vergleich.

Im Gegensatz zu den P-Zellen reagieren die M-Zellen auch auf Signale mit geringem Kontrast und unabhängig von der Wellenlänge. Dieser Zelltyp besitzt Axone, die ihre Signale sehr schnell weiterleiten. Diese Zellen werden als *ON-OFF* Zellen bezeichnet (Abbildung 6.5). Sie reagieren auf das An- und Ausschalten eines Lichtpunktes und auf Bewegungen in einer Vorzugsrichtung. Dabei ist die Aktivierung unabhängig davon, ob sich ein heller Punkt vor einem dunklen Hintergrund bewegt oder umgekehrt. Bewegt sich der Punkt in der entgegengesetzten Richtung, dann wird die Aktivität der Zelle gehemmt.

6.2.3 Der Kern des seitlichen Kniehöckers

Der Signalverlauf von der Retina bis zum primären visuellen Kortex ist in Abbildung 6.6 dargestellt. Die Ganglionzellen der Retina senden ihre Signale an den seitlichen Kniehöcker. Von dort aus gelangen die Signale in den primären visuellen Kortex. Interessant hierbei ist, daß die Retina in zwei Bereiche aufgeteilt ist, der eine deckt den linken Sichtbereich, der andere den rechten Sichtbereich ab. Auf dem Weg zum seitlichen Kniehöcker kreuzen sich die Signale, die den linken Sichtbereich des linken Auges abdecken, mit denen, die den rechten Sichtbereich des rechten Auges abdecken. Auf diese Weise gelangen alle Signale des linken Sichtbereiches in die rechte Gehirnhälfte und alle Signale des rechten Sichtbereiches in die linke Gehirnhälfte. Die Kreuzung von Signalen kann zur Steuerung eines Systems sehr hilfreich sein, wie wir bereits oben bei der Steuerung eines mobilen Roboters feststellen konnten. Dort wurde ein zentrierendes Verhalten des Roboters durch Vergleich von linker und rechter Abstandsinformation erreicht. Braitenberg [28] zeigte in einer Reihe von Gedankenexperimenten, daß vermeintlich komplexe Verhaltensmuster oft relativ einfach durch eine neuronale Kontrollarchitektur mit gekreuzten Signalen erreicht werden können.

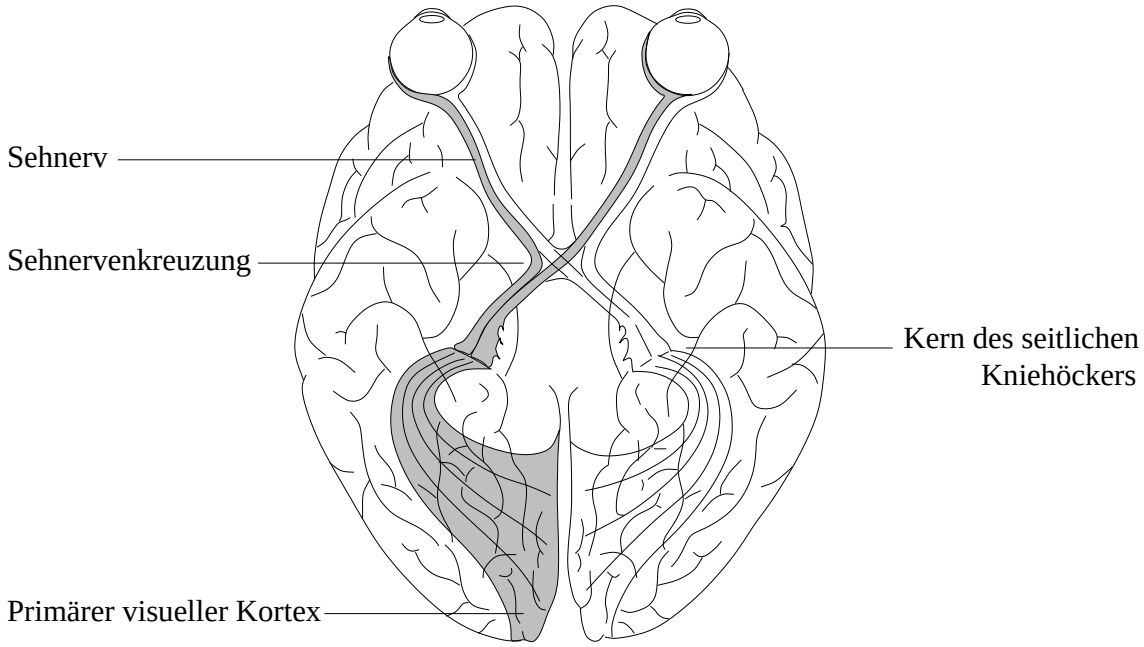


Abbildung 6.6: Der Weg der visuellen Signale von der Retina zum primären visuellen Kortex im Querschnitt von oben gesehen (nach Zeki [327]). Die Signale der Retina erreichen den seitlichen Kniehöcker und gelangen von dort zum primären visuellen Kortex. Signale des rechten Sichtbereiches werden in der linken Gehirnhälfte, die des linken Sichtbereiches in der rechten Gehirnhälfte verarbeitet.

Der seitliche Kniehöcker besitzt eine komplexe Struktur, die aus 6 Schichten besteht. Die Signale, die vom Auge derselben Gehirnhälfte erzeugt werden, terminieren in den Schichten 2, 3 und 5, während die Signale, die vom anderen Auge erzeugt werden, in den Schichten 1, 4 und 6 terminieren. Benachbarte Punkte der Retina projizieren jeweils auch auf benachbarte Punkte im seitlichen Kniehöcker. Der Grund für die Aufteilung des seitlichen Kniehöckers in unterschiedliche Schichten ist noch nicht geklärt. Es wird vermutet, daß dies mit einer funktionalen Trennung der Signale zusammenhängt. Die Zellen der vier oberen Schichten haben kleinere Zellkörper, die unteren zwei haben größere. Die vier oberen Schichten werden P Schichten und die unteren zwei werden M Schichten genannt. Die P Schichten sind für das Farbsehen zuständig. Die Signale der M-Zellen der Retina werden auf die M-Schichten des seitlichen Kniehöckers und die Signale der P-Zellen werden auf die P-Schichten abgebildet. Die Zellen des seitlichen Kniehöckers haben wie die Ganglionzellen der Retina eine *Center-Surround*-Charakteristik. Man nimmt an, daß die Funktion des seitlichen Kniehöckers ist, eine schärfere Version des retinalen Signals zu erzeugen. Im seitlichen Kniehöcker befinden sich auch die sogenannten *Double-Opponent* Zellen. Diese Zellen sprechen z.B. auf rotes Licht im Zentrum und grünes Licht im äußeren Bereich an, während grünes Licht im Zentrum bzw. rotes Licht im äußeren Bereich die Aktivität der Zelle hemmt.

6.2.4 Der visuelle Kortex (die Sehrinde)

Vom seitlichen Kniehöcker aus erreichen die Signale den primären visuellen Kortex, genannt V1. Ein kleiner Teil der Signale projiziert aber auch auf Bereiche außerhalb von V1. Bei

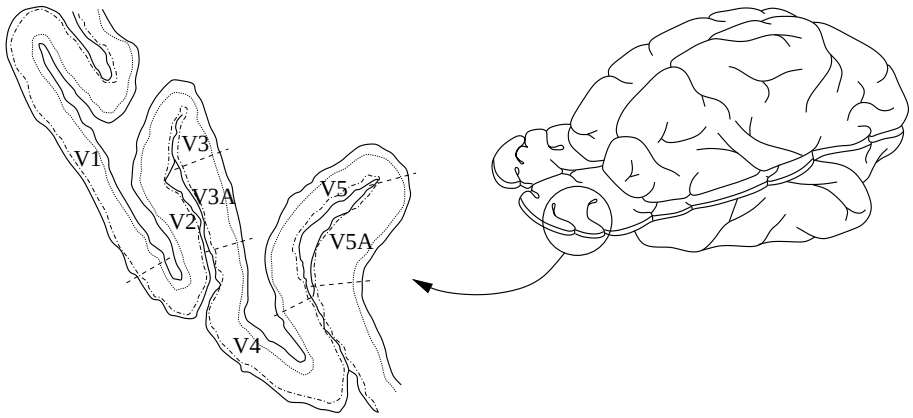


Abbildung 6.7: Der visuelle Kortex ist in mehrere Bereiche eingeteilt, die auf ganz bestimmte Aufgaben spezialisiert sind. Rechts ist ein horizontaler Schnitt durch das Gehirn eines Makakenaffen dargestellt. In der Ausschnittvergrößerung sind die Bereiche V1, V2, der Komplex V3, V4 und der Komplex V5 zu sehen. In den Bereichen V1 und V2 sind alle Teilaspekte der visuellen Informationsverarbeitung vorhanden. Sowohl in V1 als auch in V2 finden sich Zellen, die auf Bewegung, Orientierung, Wellenlänge des Lichtes und Tiefeninformation ansprechen. V3 ist für die Wahrnehmung von Formen, V4 für das Farbsehen und V5 ist für die Wahrnehmung von Bewegungen zuständig. Entsprechende Bereiche finden sich auch im menschlichen Gehirn, allerdings in unterschiedlicher Anordnung (nach Zeki [327]).

der Abbildung von der Retina auf V1 handelt es sich um eine topographische Abbildung. Die Struktur des visuellen Kortex des Makakenaffen ist in Abbildung 6.7 dargestellt. Der visuelle Kortex wird in unterschiedliche Bereiche eingeteilt, die jeweils eine spezielle Funktion besitzen. Entsprechende Bereiche finden sich auch im menschlichen Gehirn, allerdings in unterschiedlicher Anordnung. In den Bereichen V1 und V2 sind alle Teilaspekte der visuellen Informationsverarbeitung vorhanden. Sowohl in V1 als auch in V2 finden sich Zellen, die auf Bewegung, Orientierung, Wellenlänge des Lichtes und Tiefeninformation ansprechen. Die Zellen von V1 besitzen ein relativ kleines rezeptives Feld. Die Größe der rezeptiven Felder nimmt mit dem Abstand zur Fovea zu.

V1 ist in Blöcke eingeteilt. Die Struktur von V1 ist in Abbildung 6.8 dargestellt. Die P-Schicht des seitlichen Kniehöckers ist mit den Schichten 2 und 3 von V1 verbunden. Diese Schicht ist für die Farbe und die Form des Stimulus zuständig. Die Ausgabe der M-Schicht des seitlichen Kniehöckers erreicht V1 in der Schicht 4B. Hier wird Bewegung und Form des Stimulus verarbeitet. Untersucht man die Zellen von V1 in einer Richtung senkrecht zum Kortex, so sprechen sie auf Linien mit gleicher Orientierung eines Auges an. In einer Richtung parallel zur Oberfläche des Kortex sprechen die Zellen auf unterschiedliche Orientierungen an und die Blöcke des linken und rechten Auges wechseln sich ab. Daher werden diese Blöcke Okular-Dominanz-Säulen genannt. Es existieren auch Zellen, die auf orientierte Linien unabhängig von der Position innerhalb des rezeptiven Feldes ansprechen. Sie werden komplexe Zellen genannt. Außerhalb von V1 wurden Zellen entdeckt, die auf orientierte Linien ansprechen, die eine bestimmte Länge besitzen. Diese Zellen werden hyperkomplexe Zellen genannt. In den Okular-Dominanz-Säulen befinden sich kreisförmige Bereiche (in Abbildung 6.8 hellgrau). Die Zellen in diesen Bereichen sind nicht auf orientierte Linien spezialisiert. Etwa die Hälfte der Zellen sprechen auf Licht mit spezieller Wellenlänge an. Sie werden aufgrund der

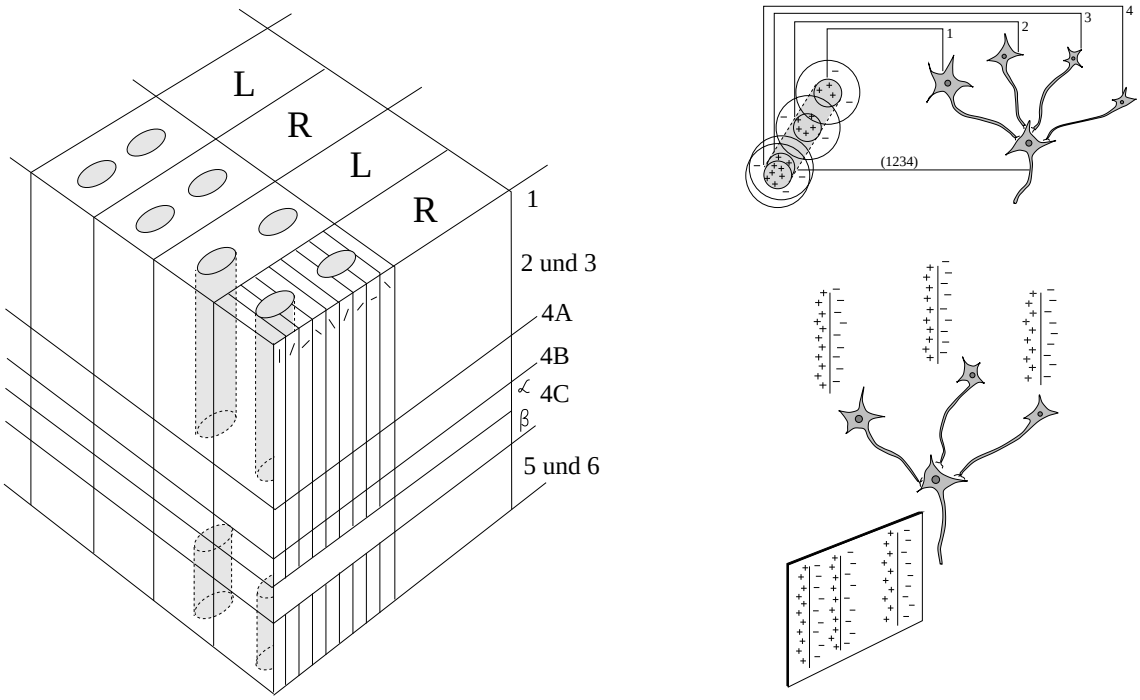


Abbildung 6.8: Struktur von V1. In V1 finden sich alle Aspekte der Verarbeitung visueller Informationen. V1 ist in Blöcke eingeteilt, deren Zellen jeweils hauptsächlich auf Signale des linken bzw. rechten Auges ansprechen. In jedem der Blöcke wird der entsprechende Bildbereich auf Linien mit unterschiedlichen Orientierungen untersucht. Die Säulen in der Mitte dieser Blöcke sind für das Farbsehen zuständig. Bewegte Signale werden in der Schicht 4B verarbeitet. Daher wird in jedem kleinen Bereich von V1 der zugehörige Bildbereich auf Orientierung, Bewegung, Farbe und Tiefenunterschiede hin analysiert. Rechts oben ist das Modell von Hubel und Wiesel für eine Zelle, wie sie in V1 vorkommt, die auf bestimmte Orientierungen reagiert, dargestellt. Rechts unten ist das Modell von Hubel und Wiesel für eine komplexe Zelle zu sehen (nach Zeki [327] und Tovée [303]).

Art des rezeptiven Feldes als *Double Opponent* Zellen bezeichnet. Die restlichen Zellen sprechen auf Licht aller möglichen Wellenlängen an, aber auch sie sind nicht auf orientierte Linien spezialisiert. Die Zellen der Schicht 4B sind für die Verarbeitung bewegter Signale zuständig. Sie sprechen auf orientierte Stimuli an, die sich in eine bestimmte Richtung bewegen. Daher sind in V1 alle Aspekte der Verarbeitung visueller Informationen repräsentiert. In jedem kleinen Bereich von V1 wird der zugehörige Bildbereich auf Orientierung, Bewegung, Farbe und Tiefenunterschiede hin untersucht.

Die Abbildung von V1 auf V2 ist ebenfalls topographisch. In V2 wurden ebenfalls Zellen gefunden, die auf bestimmte Orientierungen, Wellenlängen und bewegte Linien reagieren. Hier finden sich auch Zellen, die auf illusorische Konturen reagieren. V1 und V2 arbeiten als eine Art Verteiler, die die Signale an die unterschiedlichen visuellen Bereiche leiten. V1 sendet Signale an V2, V3, V4 und V5. V2 sendet Signale an V3, V4, V5 und V6. Im Vergleich zu den Zellen aus V1 haben die Zellen aus V3 ein deutlich größeres rezeptives Feld. Die Größe des rezeptiven Feldes der Zellen aus V2 liegt zwischen denen aus V1 und V3. Die rezeptiven Felder der Zellen von V4, V5 und V6 sind gewöhnlich größer als die der Bereiche V1, V2 und V3.

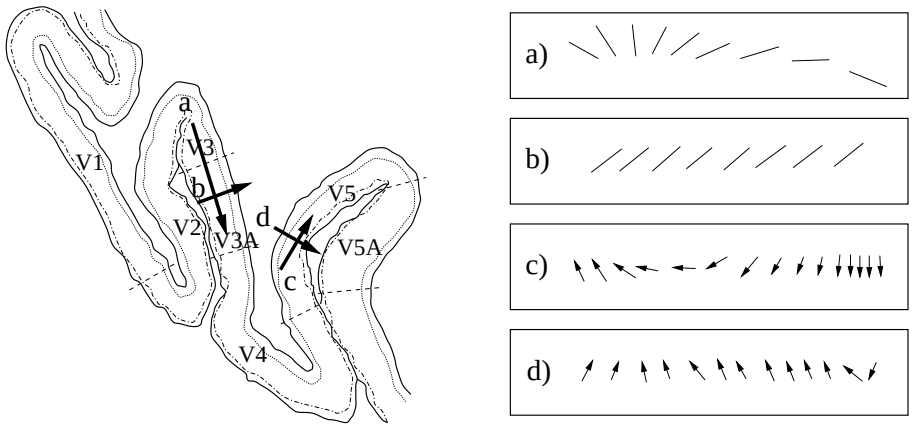


Abbildung 6.9: V3 ist für die Wahrnehmung von Formen zuständig. Die Zellen von V3 sprechen auf bestimmte Orientierungen an. Betrachtet man die Orientierungspräferenzen der Zellen in V3A, entlang einer Richtung parallel zum Kortex, so stellt man fest, daß sich die Orientierungen stetig verändern (a). Dagegen bleibt die Orientierungspräferenz der Zellen in einer Richtung rechtwinklig zum Kortex unverändert (b). V5 ist für die Wahrnehmung von Bewegungen zuständig. Die Zellen von V5 sprechen auf bewegte Signale an. Betrachtet man die Präferenzen der Zellen für die Bewegung in bestimmte Richtungen, entlang einer Richtung parallel zu V5, so ändern sich die Präferenzen für bestimmte Richtungen stetig, wobei auch manchmal abrupte Änderungen auftauchen (c). Betrachtet man dagegen die Präferenz der Zellen für bestimmte Bewegungen in einer Richtung im rechten Winkel zu V5, so ändern sich die Richtungen kaum (d). Allerdings treten auch hier ab und zu abrupte Sprünge auf (nach Zeki [327]).

Die meisten Zellen aus V3 sprechen auf orientierte Linien, unabhängig von der Farbe der Linien, an und sind für die Wahrnehmung von Formen zuständig (Abbildung 6.9). Häufig reagieren die Zellen nur, falls der Blick in eine bestimmte Richtung weist. Daher ist eine der Aufgaben von V3 vermutlich auch die Positionsbestimmung von Objekten im Raum.

Die Zellen von V4 dagegen sind für das Farbsehen zuständig. Die Zellen reagieren auf Licht mit bestimmter Wellenlänge. Einige der Zellen von V4 sprechen auch auf orientierte Linien an, doch die meisten dieser Zellen reagieren ebenfalls auf Licht mit spezieller Wellenlänge. Im Gegensatz zu Zellen aus V1, von denen einige ebenfalls auf Licht mit bestimmter Wellenlänge reagieren, reagieren die Zellen aus V4 wie es der bewußten Farbwahrnehmung entspricht. D.h eine Zelle aus V4 reagiert auf eine rote Fläche in einer komplexen Szene unabhängig von der Zusammensetzung des Lichtes, das von dieser Fläche reflektiert wird. Dieses Phänomen wird als Farbkonstanz bezeichnet.

Die Zellen von V5 sprechen hauptsächlich auf bewegte Signale an (Abbildung 6.9). Sie detektieren Lichtpunkte, unabhängig von ihrer Farbe, die sich in eine ganz bestimmte Richtung bewegen. Einige der Zellen sprechen auch auf bewegte Linien an. Ihre Eingabe erhalten sie von der Schicht 4B aus V1. Im Gegensatz zu den Zellen aus V1, deren bewegungssensitive Zellen die Bewegungen kleiner Teilbereiche eines Stimulus registriert, detektieren die Zellen von V5 die gesamte Bewegung des Stimulus. Von V5 aus erreichen visuelle Signale den Scheitellappen des Gehirns. Dort finden sich Neuronen, die auf Bewegungen reagieren, die durch die Eigenbewegung des Betrachters verursacht wurden, und Neuronen, die auf die Bewegung von Objekten im Raum reagieren.

Eine der Funktionen von V6 ist vermutlich die Repräsentation des Raumes. Manche der

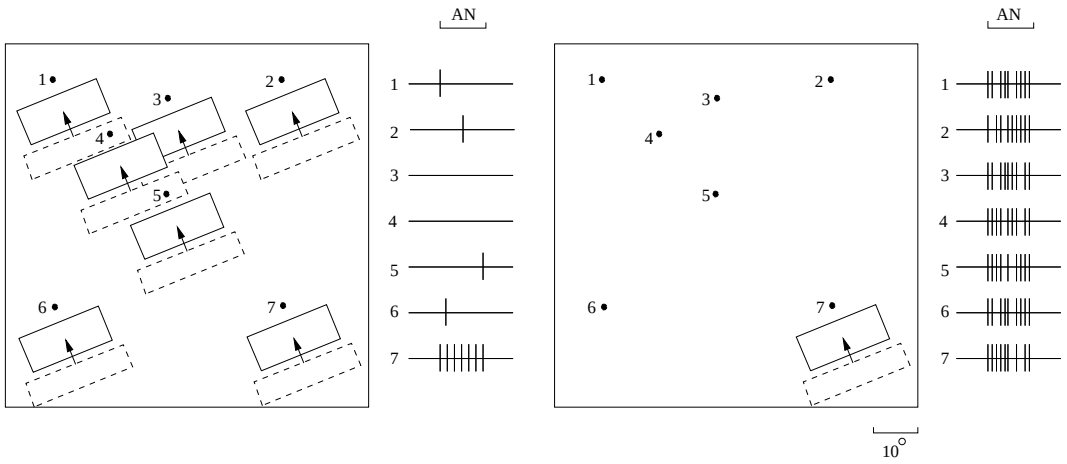


Abbildung 6.10: In V6 wurden Zellen nachgewiesen, deren rezeptives Feld unabhängig von der Blickrichtung an der gleichen Stelle bleibt. Die hier untersuchte Zelle spricht auf einen Balken an, der sich in Richtung der 11 Uhr Position bewegt, sofern der Blick auf den 7. Punkt gerichtet ist. Wird die Blickrichtung geändert, d.h. einer der Punkte 1 bis 6 fixiert, so würde man erwarten, daß das rezeptive Feld sich jeweils an den im Bild links dargestellten Positionen befinden würde. Denn das rezeptive Feld einer Zelle ist durch die Verbindungen zu den Rezeptoren der Retina gegeben, deren Orientierung sich bei einer Augenbewegung geändert hat. Die Zelle reagiert aber kaum, falls in den entsprechenden Sichtbereichen ein Stimulus präsentiert wird (links). Wird der Stimulus jedoch auch bei geänderter Blickrichtung wie zuvor bei 7 präsentiert, dann spricht die Zelle an (rechts). Die Orientierung des Auges wird also "herausgerechnet" (nach Zeki [327]).

Zellen aus V6 sprechen nur dann an, wenn der Blick in eine bestimmte Richtung weist. In V6 wurden auch Zellen gefunden, deren rezeptives Feld unabhängig von der Blickrichtung immer an der gleichen Position bleibt (Abbildung 6.10). Die Orientierung des Auges wird also "herausgerechnet". Einige dieser Zellen sind auf bestimmte Orientierungen des Stimulus spezialisiert.

Die Verarbeitung visueller Informationen findet aber nicht nur im hinteren Bereich des Gehirns statt, sondern auch in anderen Bereichen. In den vorderen Lappen werden z.B. visuelle Informationen, die zur Steuerung von Augenbewegungen genutzt werden, verarbeitet. Abschließend bleibt zu sagen, daß das visuelle System des Menschen aus vielen unterschiedlichen Bereichen besteht, die jeweils auf ganz bestimmte Aufgaben spezialisiert sind. Eine Verletzung einer oder mehrerer dieser spezialisierten Bereiche führt zu Symptomen, die in Zusammenhang mit der Spezialisierung stehen. So kann z.B. eine Verletzung von V4 die Farbwahrnehmung stören. Je nachdem, welche Bereiche intakt sind, können die anderen visuellen Wahrnehmungen weiterhin funktionieren. So ist es z.B. möglich, daß eine Person zwar keine Gesichter, trotzdem aber Gesichtsausdrücke erkennen kann.

6.3 Die komplex-logarithmische Abbildung

Schwartz [271, 272] zeigte, daß die Abbildung der Retina auf den primären visuellen Kortex durch eine komplex-logarithmische Abbildung beschrieben werden kann. Nach Schwartz entsteht die komplex-logarithmische Abbildung einerseits durch die Verteilung der Ganglionzellen auf der Retina und durch die Projektion vom seitlichen Kniehöcker auf den primären

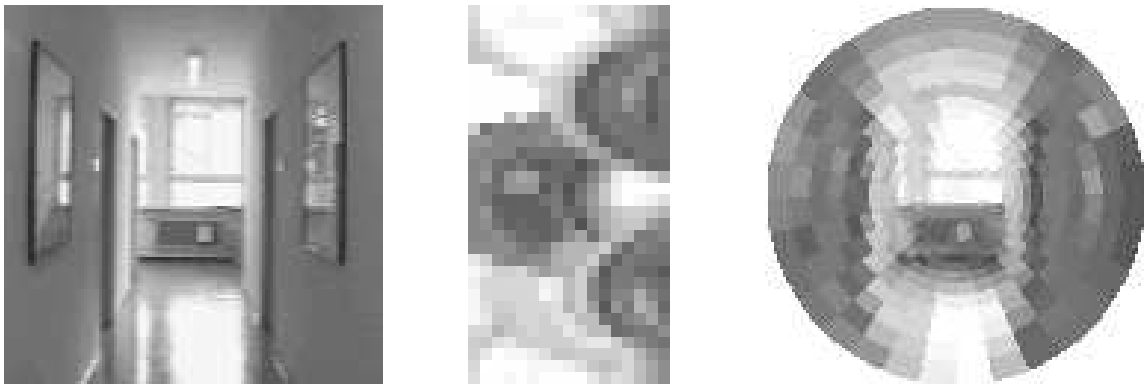


Abbildung 6.11: Links ist das Originalbild dargestellt. In der Mitte ist dasselbe Bild im komplex-logarithmischen Raum zu sehen. Das rechte Bild entstand durch eine Rücktransformation des mittleren Bildes in den ursprünglichen Raum.

visuellen Kortex.

Die komplex-logarithmische Abbildung bzw. die Transformation in Polarkoordinaten wurde bereits von anderen eingehend untersucht und in so unterschiedlichen Bereichen, wie z.B. Bewegungs-Stereo [140], Extraktion bewegter Objekte [139, 107, 210], Zeichenerkennung [232], Mustererkennung [314] und Objekterkennung und Zentrierung [233], eingesetzt. Die komplex-logarithmische Abbildung ist pseudo-invariant gegenüber Größenveränderungen, Rotationen und Skalierung der Projektion [272]. Wird ein Objekt fixiert, so führt eine Größenveränderung oder eine Rotation zur einer Verschiebung im Raum der komplex-logarithmischen Abbildung. Nach Cavanagh [44, 45] entsteht die Größen- und Positionsinvarianz im visuellen System durch eine Kombination aus komplex-logarithmischer Abbildung und Fouriertransformation.

Die komplex-logarithmische Abbildung wurde von Tistarelli und Sandini [298] eingehend analysiert. Die komplex-logarithmische Abbildung ist eine konforme Abbildung vom kartesischen in den komplex-logarithmischen Raum. Neben den bereits erwähnten Invarianzeigenschaften besitzt die komplex-logarithmische Abbildung weitere vorteilhafte Eigenschaften. Aufgrund der nicht-uniformen Abtastung wird eine Datenreduktion durchgeführt. Bewegungsgleichungen, die eine Beziehung zwischen der Eigenbewegung und dem optischen Fluß herstellen, werden vereinfacht, und die Zeit bis zu einem Zusammenstoß (*Time to Impact*) kann unter Verwendung der komplex-logarithmischen Abbildung einfach berechnet werden.

Die sogenannte komplex-logarithmische Abbildung in Richtung der Eigenbewegung hat einige besondere Eigenschaften. Bei der komplex-logarithmischen Abbildung wird die Transformation um den Ursprung, also in Richtung der optischen Achse vorgenommen. Im Gegensatz dazu wird bei der komplex-logarithmischen Abbildung in Richtung der Eigenbewegung die Transformation um das Zentrum der Expansion $(x_{\mathrm{FOE}}, y_{\mathrm{FOE}})$ vorgenommen. Die komplex-logarithmische Abbildung in Richtung der Eigenbewegung ist wie folgt definiert [141]:

$$r(x,y) = \log\sqrt{(x-x_{\mathrm{FOE}})^2+(y-y_{\mathrm{FOE}})^2} \tag{6.1}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{y-y_{\mathrm{FOE}}}{x-x_{\mathrm{FOE}}}\right) \tag{6.2}$$

Ein Bild, das mit dieser Abbildung transformiert wurde, ist in Abbildung 6.11 zu sehen.

Bewegt sich ein Betrachter vorwärts, so entsteht lediglich radialer optischer Fluß. Im

komplex-logarithmischen Raum entsteht daher lediglich optischer Fluß in horizontaler Richtung [139]. Jain et al. [140] zeigten, daß folgende Beziehung gilt. Dabei sei angenommen, daß die Brennweite $f = 1$ ist.

$$\frac{dr}{dZ} = -\frac{1}{Z} \qquad \text{and} \qquad \frac{d\theta}{dZ} = 0 \tag{6.3}$$

Daher ist der durch die Eigenbewegung induzierte optische Fluß im komplex-logarithmischen Raum ein direktes Maß für den Abstand der betrachteten Objekte.

Die komplex-logarithmische Abbildung ist lediglich für einen translatierenden Beobachter definiert. Daher muß ein Mechanismus existieren, der die rotatorische Bewegung bei einer technischen Anwendung herausrechnet. In biologischen Systemen existiert ein Mechanismus, der unter anderem dazu eingesetzt wird, die Umgebung als stationär wahrzunehmen. Dieser Mechanismus, genannt Reafferenzprinzip, wird im folgenden näher betrachtet.

6.4 Das Reafferenzprinzip

Die hier beschriebenen Ausführungen zum Reafferenzprinzip basieren auf der Darstellung von von Holst und Mittelstaedt [310]. Nach der klassischen Reflextheorie wird eine Bewegung durch einen Reflex ausgelöst. Wird eine Fliege z.B. in der Mitte eines senkrecht gestreiften Zylinders plaziert und dieser dann rotiert, so dreht sich die Fliege in der gleichen Richtung mit. Die Fliege versucht, die Bewegung der Umwelt zu kompensieren. Diese Reaktion wird als optomotorischer Reflex bezeichnet. Führt die Fliege eine aktive Bewegung aus, so wird nach der Reflextheorie der Reflex unterdrückt. Ohne die Unterdrückung dieses Reflexes könnte die Bewegung natürlich nicht ausgeführt werden.

Von Holst und Mittelstaedt betrachteten die Wechselwirkungen zwischen Zentralnervensystem und Peripherie und stellten statt des äußeren Reizes die aktive Bewegung in den Vordergrund. Ein das Zentralnervensystem erreichender Impuls, ausgelöst durch einen Reiz, wird als Afferenz bezeichnet. Ein Impuls, der aus dem Zentralnervensystem herauskommt, wird als Efferenz bezeichnet. Im Gegensatz zur klassischen Reflextheorie fragten von Holst und Mittelstaedt nicht nach der Beziehung zwischen der Afferenz und der Efferenz sondern betrachteten stattdessen zuerst die Efferenz. In Abbildung 6.12 ist das Reafferenzprinzip graphisch dargestellt. Die vom Zentralnervensystem ausgehende Efferenz erreicht den Effektor und verursacht wiederum eine Afferenz. Diese von der Efferenz verursachte Afferenz bezeichnen von Holst und Mittelstaedt als Reafferenz. Neben dem Efferenzsignal, das den Effektor erreicht, wird es außerdem als Kopie zwischengespeichert. Diese Efferenzkopie wird mit der entstandenen Afferenz verglichen. Als Exafferenz wird die Differenz zwischen Afferenz und Efferenzkopie bezeichnet. Demnach ist jede Änderung der Afferenz, die nicht Folge einer Efferenz ist, eine Exafferenz. Die Differenz entsteht also durch äußere Einwirkungen und kann zur Regelung des Systems bzw. zur Steuerung des Effektors eingesetzt werden.

Das Reafferenzprinzip wird nun anhand der Funktionsweise des menschlichen Auges näher besprochen. Wird das Auge aktiv bewegt, so wird eine Efferenz erzeugt, die den Augenmuskel erreicht. Eine Kopie dieses Signals wird kurz zwischengespeichert. Durch die Bewegung des Auges entsteht eine retinale Bildverschiebung, die als Afferenz zurückgeliefert wird. Dieses Signal wird mit der Efferenzkopie verglichen. Im Idealfall entspricht die erwartete Bildverschiebung der tatsächlichen Bildverschiebung. Dadurch ist es möglich, die Umwelt als stillstehend wahrzunehmen (Abbildung 6.13a). Bewegt sich ein Objekt im Blickfeld des Betrachters, so

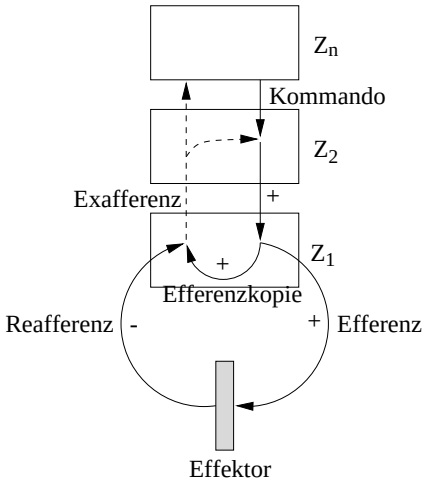


Abbildung 6.12: Eine aktive Bewegung (Kommando) eines Zentrums Z_n erzeugt im Zentrum Z_1 eine Efferenz, die den Effektor erreicht. Die Efferenz wird zugleich zwischengespeichert und mit der Reafferenz, die vom Effektor zurückgeliefert wird, verglichen. Die Differenz zwischen Reafferenz und Efferenzkopie wird als Exafferenz bezeichnet. Die Exafferenz kann zur Steuerung des Effektors eingesetzt werden (nach von Holst und Mittelstaedt [310]).

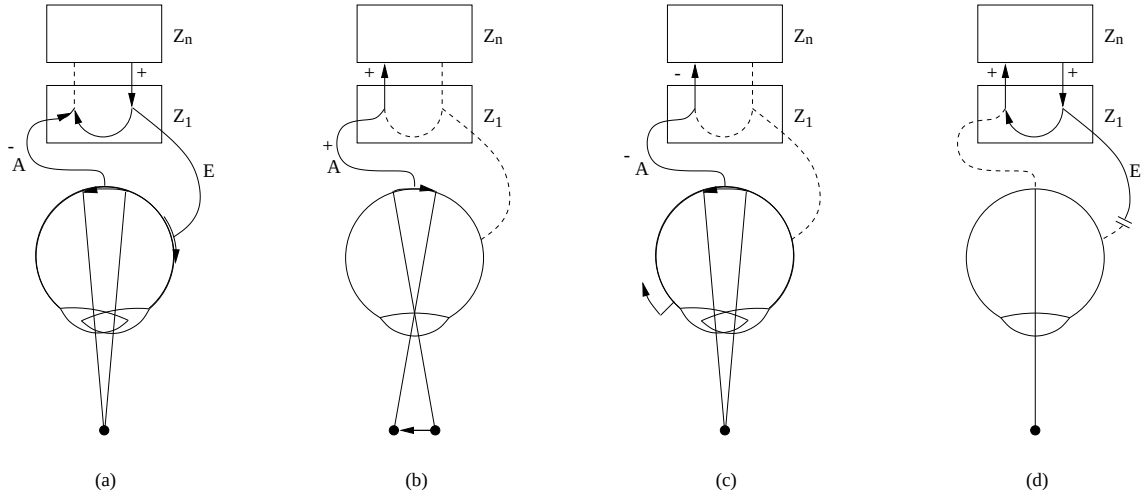


Abbildung 6.13: Reafferenzprinzip am Beispiel des Auges. Z_n bezeichnet ein höheres, Z_1 ein niederes optisches Zentrum. Die Efferenz ist mit E, die Afferenz mit A bezeichnet. (a) Das Signal aus Z_1 erreicht den Augenmuskel und dieser bewegt das Auge. Dadurch entsteht eine retinale Bildverschiebung (Afferenz A), die mit der Erwartung (Efferenzkopie) verglichen wird. Die Umgebung wird als stillstehend wahrgenommen. (b) Bewegt sich ein Objekt, so verändert sich das retinale Bild und die Bewegung wird als solche wahrgenommen. (c) Wird der Augapfel durch einen äußeren Einfluß bewegt, so scheint sich die Umgebung zu bewegen. Der Vergleich der Afferenz mit der Efferenzkopie ergibt einen starken Unterschied, wodurch die Bewegung der Umwelt signalisiert wird. (d) Ist der Augenmuskel betäubt, und man versucht, die Blickrichtung zu verändern, so scheint sich die Umgebung zu bewegen. In diesem Fall ist das retinale Signal gleich Null und die Efferenzkopie ungleich Null, was zum Bewegungseindruck führt (nach von Holst und Mittelstaedt [310]).

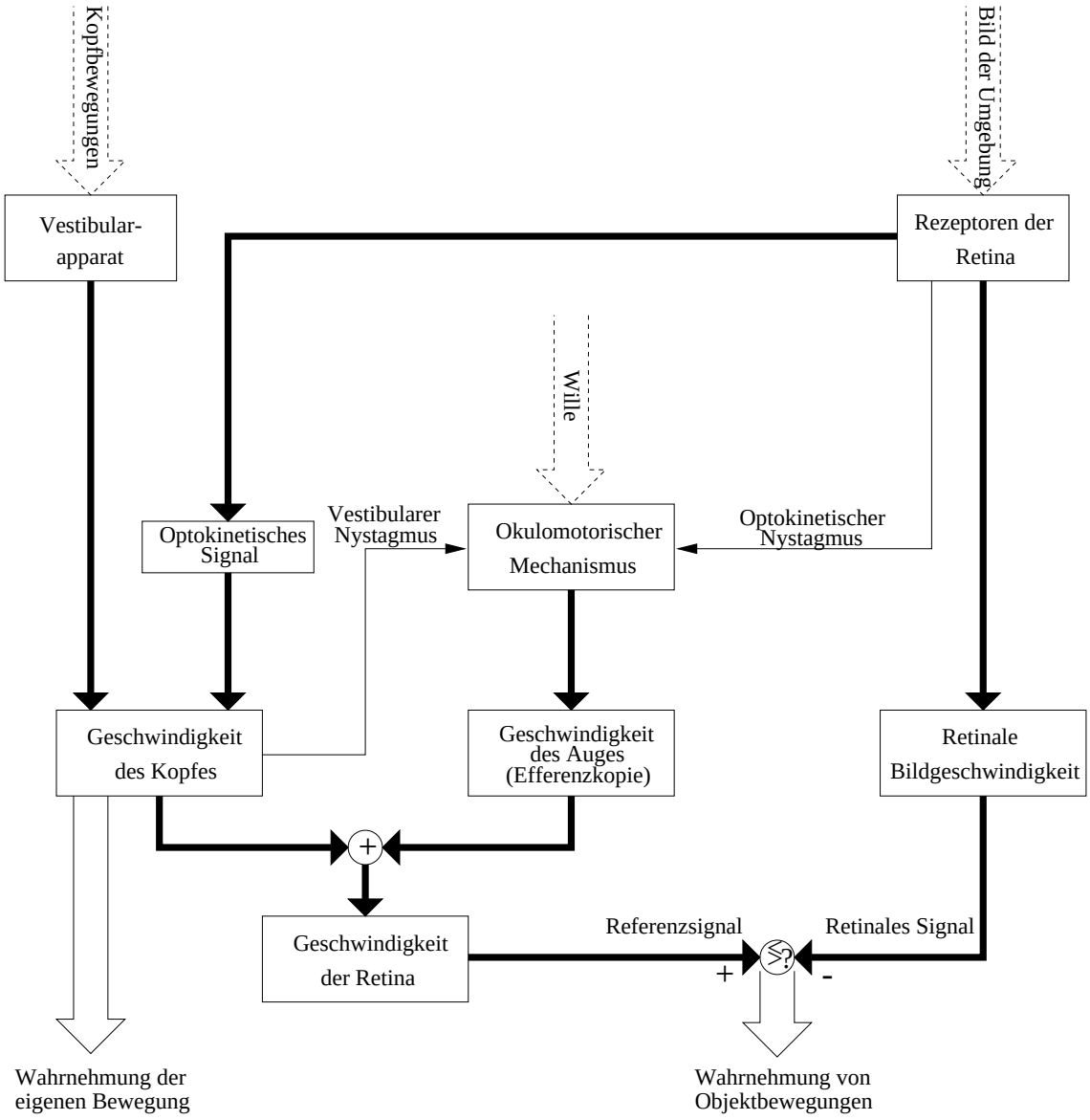


Abbildung 6.14: Wertheim's Modell zur Wahrnehmung von Objektbewegungen. Die retinale Bildgeschwindigkeit wird mit einem Referenzsignal (der Geschwindigkeit der Retina) verglichen. Die Geschwindigkeit der Retina berechnet sich aus der Summe der Augen- und Kopfbewegung. Die Bewegung des Kopfes wird sowohl durch Informationen des Vestibularapparates als auch aus visuellen Informationen ermittelt. Ein Objekt im Raum wird als bewegt wahrgenommen, wenn das Referenzsignal stark genug vom retinalen Signal abweicht (nach Wertheim [317]).

wird diese Bewegung über die retinale Bildverschiebung wahrgenommen (Abbildung 6.13b). Wird das Auge dagegen passiv bewegt, z.B. indem man vorsichtig mit dem Finger das Auge berührt, so scheint sich die Umgebung zu bewegen. In diesem Falle ist die Efferenzkopie gleich Null und der Vergleich der Afferenz mit der Efferenzkopie ergibt eine deutliche Differenz (Abbildung 6.13c). Versucht man, mit betäubtem Augenmuskel die Blickrichtung zu ändern, so scheint die Umgebung einen Sprung zu machen, der der Größe der beabsichtigten Blickbewegung entspricht. In diesem Fall ist die Afferenz gleich Null (das retinale Signal hat sich nicht verändert), während die Efferenzkopie ungleich Null ist (Abbildung 6.13d).

Wertheim [317] entwickelte ein umfangreiches Modell zur Wahrnehmung von Objektbewegungen (Abbildung 6.14). In seinem Modell tritt an die Stelle der extraretinalen Information (der Efferenzkopie der Augenbewegung) ein Referenzsignal. Dieses Referenzsignal beschreibt die geschätzte Geschwindigkeit der Retina. Die Geschwindigkeit der Retina wird aus Summe der Augen- und Kopfbewegung berechnet. Die Geschwindigkeit des Kopfes wird nach Wertheim zum Teil aus der Information des Vestibularapparates aber auch aus optischen Informationen ermittelt. Da der Vestibularapparat lediglich die Beschleunigung des Kopfes mißt, wird ein optischer Anteil des Signals benötigt, wenn sich der Kopf mit konstanter Geschwindigkeit bewegt. Ein Objekt im Raum wird als bewegt wahrgenommen, wenn das Referenzsignal stark genug vom retinalen Signal abweicht.

Das hier entwickelte Verfahren zur visuellen Steuerung eines mobilen Roboters nutzt sowohl die Eigenschaften des Reafferenzprinzips als auch die Eigenschaften der komplex-logarithmischen Abbildung. Analog zum visuellen System des Menschen, das die Information des Vestibularapparates zur Bestimmung der Eigenbewegung nutzt, werden hier die Statusinformationen des Roboters und der Kamera eingesetzt.

6.5 Visuelle Steuerung eines mobilen Roboters unter Verwendung des Reafferenzprinzips und der komplex-logarithmischen Abbildung

Das System zur Steuerung eines mobilen Roboters läßt sich in drei Teile gliedern. Zuerst werden für die Steuerung relevante Merkmale aus den Bildern extrahiert. Diese Punktmerkmale werden verwendet, um den optischen Fluß der Punkte zu berechnen. Der optische Fluß wird in den komplex-logarithmischen Raum transformiert. Schließlich wird die Differenz des optischen Flusses der linken und rechten periphären visuellen Bereiche eingesetzt, um den Roboter zu steuern. Als Eingabe erhält der Algorithmus das Bild $I(t_2)$, das zum Zeitpunkt t_2 aufgenommen wurde und Statusinformationen vom Roboter und von der Kamera. Aus diesen Statusinformationen läßt sich die Eigenbewegung der Kamera berechnen. Der Algorithmus arbeitet in jeder Iteration mit zwei Bildern. Das Bild $I(t_1)$ aus der vorangegangenen Iteration steht dem Algorithmus ebenfalls zur Verfügung.

6.5.1 Merkmalsextraktion

Als erstes werden markante Punkte aus dem Eingabebild extrahiert. Dies erleichtert die Berechnung des optischen Flusses. Aufgrund des Apperturproblems [141] ist ein dichtes Flußfeld nur sehr schwer zu berechnen. Viele Methoden zur Berechnung des optischen Flusses verwenden einen iterativen Ansatz, wie z.B. Horn und Schunck [127]. Meist wird noch eine zeitliche Gaußsche Glättung eingesetzt. Daher werden mehrere Bilder benötigt, was zu einer

zusätzlichen Verzögerung führt, da der optische Fluß nur für das mittlere Bild der Sequenz berechnet wird.

Der hier vorgestellte Ansatz arbeitet zu jedem Zeitpunkt auf zwei Bildern. Der optische Fluß wird lediglich für die extrahierten Punktmerkmale berechnet. Da nur markante Punkte extrahiert werden, kann eine Korrespondenz sehr einfach hergestellt werden. Zudem kann über die bekannte Eigenbewegung der Kamera der Suchbereich eingeschränkt werden. Bei den in diesem Kapitel beschriebenen Experimenten wurden die markanten Punkte mit dem Moravec-Operator [205, 206] extrahiert. Der Moravec-Operator wurde bereits in Abschnitt 5.1.5 ausführlich beschrieben. Der Moravec-Operator extrahiert Punkte, bei denen die Varianz der Intensität in horizontaler, vertikaler und diagonaler Richtung ein lokales Maximum annimmt. Der Operator ist selbst bei einfacher Implementierung sehr schnell zu berechnen. Dies wurde bereits in Abschnitt 5.4.5 gezeigt. Für den Einsatz wurde der Operator zusätzlich noch optimiert. Durch die Optimierung kann der Operator für ein 128×128 großes Bild in 0.0543 Sekunden auf einem Pentium PC mit 133MHz berechnet werden.

Im folgenden sei $F(t_2)$ die Menge der Punktmerkmale, die aus dem Bild $I(t_2)$ extrahiert wurden.

6.5.2 Kompensation des rotatorischen Anteils der Eigenbewegung

Die komplex-logarithmische Abbildung in Richtung der Eigenbewegung ist nur für einen Betrachter definiert, der eine translatorische Bewegung macht. Ein mobiler Roboter kann jedoch auch rotatorische Bewegungen ausführen. Im allgemeinen besteht die Bewegung der Kamera des Roboters aus einem translatorischen und einem rotatorischen Anteil. Aus diesem Grund wird in dem hier vorgestellten Ansatz der rotatorische Anteil der Bewegung kompensiert. Durch diese Kompensation wird lediglich der durch die translatorische Bewegung der Kamera induzierte optische Fluß berechnet. Der so entstehende optische Fluß wird dann im komplex-logarithmischen Raum ausgewertet.

Eine Kompensation der Eigenbewegung wurde von Murray und Basu [208] zur Extraktion bewegter Kanten, von Ebner zur Extraktion bewegter Objekte [74, 78] und zur Bewegungsverfolgung [84], von Feyrer und Zell [90] bei der Personenverfolgung und von Nair und Aggarwal [210] zur Detektion bewegter Hindernisse eingesetzt. Die Berechnung virtueller Ansichten eignet sich auch zur Berechnung des optischen Flusses [189] und zur Detektion von Hindernissen am Boden [189, 322]. Siehe auch Möller [203] für ein neuronales Modell zur Wahrnehmung durch Vorhersage.

Die homogene Transformation [59, 197], die die Eigenbewegung der Kamera beschreibt, kann aus den Statusinformationen des Roboters und der Kamera berechnet werden. Die Eigenbewegung der Kamera setzt sich aus der Bewegung des Roboters und der Bewegung des Manipulators, an dem die Kamera befestigt ist, zusammen. Während der Experimente, die im folgenden beschriebenen werden, befand sich die Kamera auf einer Schwenk-Neige-Einheit. Folgende Werte können über Statusinformationen abgefragt werden: die rotatorische und translatorische Geschwindigkeit des Roboters und die Position der Gelenke der Schwenk-Neige-Einheit. Die Geschwindigkeit der Gelenke der Schwenk-Neige-Einheit wird nicht gemessen. Es ist lediglich die gewünschte Geschwindigkeit, nicht aber die tatsächliche Geschwindigkeit abfragbar [68].

Es sei $\{C\}$ das Koordinatensystem der Kamera und $\{R\}$ das Koordinatensystem der Basis des Roboters. Die Koordinatensysteme sind in Abbildung 6.15 dargestellt. Ebenfalls zu sehen ist der Roboter, mit dem die hier beschriebenen Experimente durchgeführt wurden.

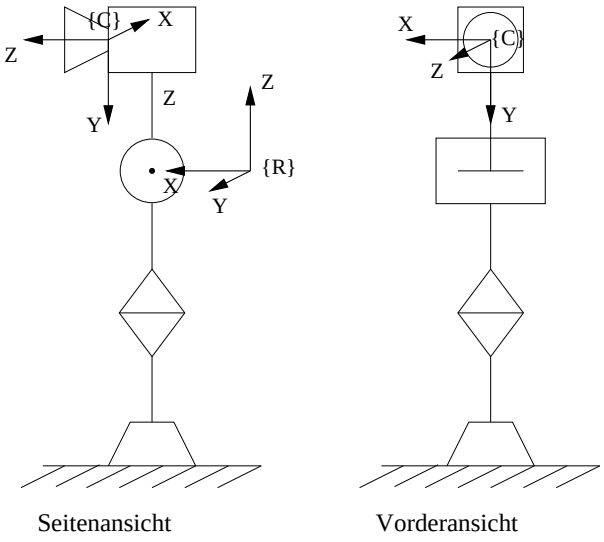


Abbildung 6.15: Der Real World Interface B21 Roboter [244, 243], mit dem die Experimente zur visuellen Steuerung durchgeführt wurden. Rechts ist die verwendete Schwenk-Neige-Einheit zusammen mit den verwendeten Koordinatensystemen für die Kamera $\{C\}$ und die Basis des Roboters $\{R\}$ dargestellt.

Die homogene Transformation (Abbildung 6.16), die die Bewegung der Kamera beschreibt, läßt sich in drei elementare Transformationen aufteilen. Die erste beschreibt die Position der Kamera relativ zur Basis des Roboters zum Zeitpunkt t_2. Die zweite beschreibt die Bewegung des Roboters. Die dritte beschreibt die Position der Kamera relativ zur Basis des Roboters zum Zeitpunkt t_1. Aus diesen drei Transformationen kann die homogene Transformation ${}^{C(t_2)}_{C(t_1)}\mathbf{T}$ berechnet werden. Sie transformiert Koordinaten im Koordinatensystem der Kamera $\{C(t_1)\}$ zum Zeitpunkt t_1 in das Koordinatensystem der Kamera $\{C(t_2)\}$ zum Zeitpunkt t_2.

$$ {}^{C(t_2)}_{C(t_1)}\mathbf{T} = {}^{C(t_2)}_{R(t_2)}\mathbf{T} \cdot {}^{R(t_2)}_{R(t_1)}\mathbf{T} \cdot {}^{R(t_1)}_{C(t_1)}\mathbf{T} \tag{6.4} $$

Dabei beschreibt ${}^{C(t_2)}_{R(t_2)}\mathbf{T}$ die Transformation vom Koordinatensystem des Roboters $\{R(t_2)\}$ in das Koordinatensystem der Kamera $\{C(t_2)\}$ zum Zeitpunkt t_2, ${}^{R(t_2)}_{R(t_1)}\mathbf{T}$ beschreibt die Bewegung des Roboters, und ${}^{R(t_1)}_{C(t_1)}\mathbf{T}$ beschreibt die Transformation vom Koordinatensystem der Kamera $\{C(t_1)\}$ in das Koordinatensystem des Roboters zum Zeitpunkt t_1. Die Bewegung des Roboters ${}^{R(t_2)}_{R(t_1)}\mathbf{T}$ wird aus den Statusinformationen des Roboters wie folgt berechnet.

$$ {}^{R(t_2)}_{R(t_1)}\mathbf{T} = \mathbf{R}_Z(-\omega(t_2 - t_1)) \cdot \mathbf{D}_X(-v(t_2 - t_1)) \tag{6.5} $$

Dabei ist $\mathbf{D}_X$ eine Matrix, die eine translatorische Bewegung entlang der X-Achse mit der Geschwindigkeit v definiert. $\mathbf{R}_Z$ beschreibt die rotatorische Bewegung des Roboters mit einer Geschwindigkeit ω um die Z-Achse. Die homogenen Transformationen ${}^{C(t_2)}_{R(t_2)}\mathbf{T}$ und ${}^{R(t_2)}_{R(t_1)}\mathbf{T}$ werden mit Standardverfahren aus der Manipulatorkinematik berechnet.

Im folgenden werden Punkte mit dreidimensionalen Koordinaten mit großen Buchstaben bezeichnet. Punkte in Bildkoordinaten werden mit kleinen Buchstaben bezeichnet. Es sei

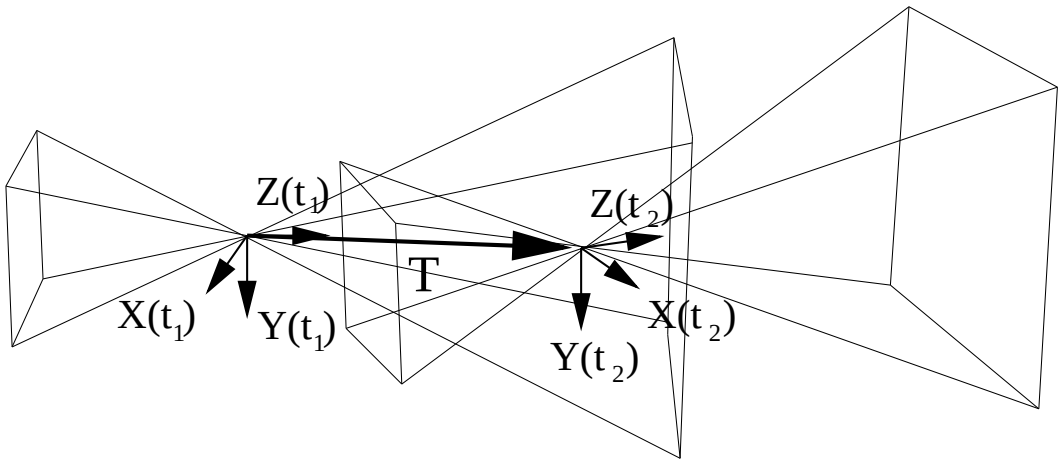


Abbildung 6.16: Bewegung der Kamera.

${}^{C(t_1)}\mathbf{P} = [X(t_1), Y(t_1), Z(t_1), 1]$ ein Punkt im Koordinatensystem der Kamera $\{C(t_1)\}$ zum Zeitpunkt t_1. Der gleiche Punkt hat dann die Koordinaten ${}^{C(t_2)}\mathbf{P} = [X(t_2), Y(t_2), Z(t_2), 1]$ im Koordinatensystem der Kamera zum Zeitpunkt t_2. Es gilt

$$ {}^{C(t_2)}\mathbf{P} = {}^{C(t_2)}_{C(t_1)}\mathbf{T} \cdot {}^{C(t_1)}\mathbf{P} \tag{6.6} $$

$$ \begin{bmatrix} X(t_2) \\ Y(t_2) \\ Z(t_2) \\ 1 \end{bmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X(t_1) \\ Y(t_1) \\ Z(t_1) \\ 1 \end{bmatrix} \tag{6.7} $$

Mit perspektivischer Projektion und der Brennweite f transformieren sich die Bildkoordinaten wie folgt [208]:

$$ x(t_2) = f\frac{X(t_2)}{Z(t_2)} \tag{6.8} $$

$$ = f\frac{r_{11}X(t_1) + r_{12}Y(t_1) + r_{13}Z(t_1) + t_x}{r_{31}X(t_1) + r_{32}Y(t_1) + r_{33}Z(t_1) + t_z} \tag{6.9} $$

$$ = f\frac{r_{11}x(t_1) + r_{12}y(t_1) + fr_{13} + f\frac{t_x}{Z(t_1)}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{t_z}{Z(t_1)}} \tag{6.10} $$

$$ y(t_2) = f\frac{Y(t_2)}{Z(t_2)} \tag{6.11} $$

$$ = f\frac{r_{21}X(t_1) + r_{22}Y(t_1) + r_{23}Z(t_1) + t_y}{r_{31}X(t_1) + r_{32}Y(t_1) + r_{33}Z(t_1) + t_z} \tag{6.12} $$

$$ = f\frac{r_{21}x(t_1) + r_{22}y(t_1) + fr_{23} + f\frac{t_y}{Z(t_1)}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33} + f\frac{t_z}{Z(t_1)}} \tag{6.13} $$

$$ \tag{6.14} $$

Für rotatorische Kamerabewegungen werden die einzelnen Punkte des Bildes wie folgt trans-

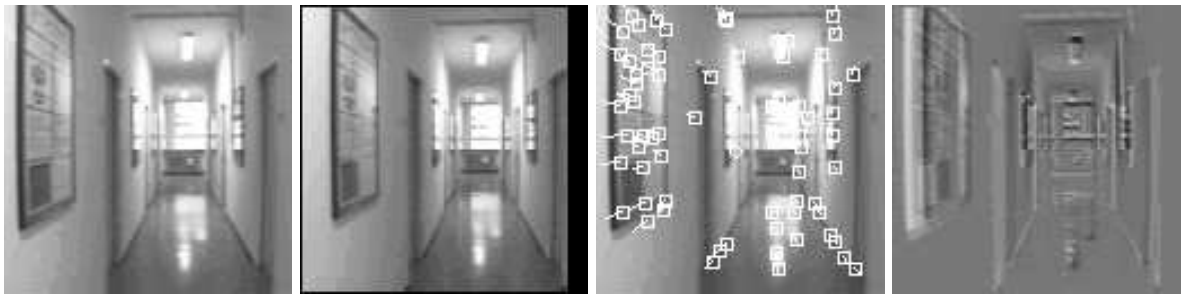


Abbildung 6.17: Kompensation der rotatorischen Eigenbewegung und der daraus berechnete optische Fluß. Von links nach rechts sind dargestellt: Bild $I(t_2)$, Bild $\tilde{I}(t_2)$ (berechnet aus $I(t_1)$), optischer Fluß zwischen den Bildern $I(t_2)$ und $\tilde{I}(t_2)$, Differenzbild berechnet aus $I(t_2)$ und $\tilde{I}(t_2)$.

formiert:

$$x(t_2) = f\frac{r_{11}x(t_1) + r_{12}y(t_1) + fr_{13}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33}} \tag{6.15}$$

$$y(t_2) = f\frac{r_{21}x(t_1) + r_{22}y(t_1) + fr_{23}}{r_{31}x(t_1) + r_{32}y(t_1) + fr_{33}} \tag{6.16}$$

Es sei $\tilde{I}(t_2)$ das Bild, das durch Anwendung der obigen Gleichung aus $I(t_1)$ berechnet wurde. Damit wird die Bewegung, die durch die rotatorische Bewegung der Kamera entstanden ist, herausgerechnet. Die Auswirkung dieser Transformation ist in Abbildung 6.17 zu sehen. Analog wird die zugehörige Position der Punktmerkmale $\tilde{F}(t_2)$ aus $F(t_1)$ berechnet.

Zur Kompensation der rotatorischen Bewegung der Kamera sind sehr exakte Statusinformationen erforderlich. Der hier eingesetzte Roboter wird über eine verteilte Client-Server-Architektur [245, 246] gesteuert. Daher wird eine Verzögerung von 150ms eingesetzt, nachdem die Geschwindigkeit des Roboters verändert wurde, um genaue Statusinformationen zu erhalten. Erst nach dieser Verzögerung wird dann ein neues Bild digitalisiert. Die Genauigkeit der Kompensation der rotatorischen Eigenbewegung ist in Abbildung 6.17 zu sehen.

6.5.3 Der optische Fluß im komplex-logarithmischen Raum

Zwischen den Punktmerkmalen $\tilde{F}(t_2)$ und $\tilde{F}(t_1)$ wird eine Korrespondenz hergestellt. Dazu wird ein einfaches Fehlermaß, basierend auf der quadrierten Differenz der Pixel innerhalb eines kleinen Bereiches um die Punktmerkmale eingesetzt. Das Fehlermaß wird mit der Entfernung des korrespondierenden Punktes skaliert. Dadurch wird bei zwei gleichen Merkmalen das nähergelegene bevorzugt. Da die rotatorische Bewegung herausgerechnet wurde, bewegt sich der optische Fluß ausgehend vom Zentrum der Expansion radial nach außen. Aus diesem Grund kann der Suchbereich, der für die Herstellung der Korrespondenz in Frage kommt, auf einen sehr kleinen Bereich eingeschränkt werden. Der so berechnete optische Fluß ist in Abbildung 6.17 zu sehen.

Der optische Fluß in Richtung der translatorischen Eigenbewegung wird nun in den komplex-logarithmischen Raum transformiert. Dazu werden die Koordinaten des Zentrums der Expansion benötigt. Die Koordinaten können aus der bekannten Eigenbewegung der

Kamera berechnet werden [139, 140, 141, 190].

$$\begin{bmatrix} x_{\mathrm{FOE}} \\ y_{\mathrm{FOE}} \end{bmatrix} = f \begin{bmatrix} t_x/t_z \\ t_y/t_z \end{bmatrix} \tag{6.17}$$

Die komplex-logarithmische Abbildung wurde bereits oben genau beschrieben. Für den praktischen Einsatz müssen aufgrund der begrenzten Auflösung der Bilder noch eine Reihe von Parametern definiert werden. Daher werden die Bilder wie folgt transformiert.

$$r = \frac{r_{\max}}{\log r_{\mathrm{src}}} \log \sqrt{(x - x_{\mathrm{FOE}})^2 + (y - y_{\mathrm{FOE}})^2} \tag{6.18}$$

$$\theta = \frac{\theta_{\max}}{2\pi} \left(\pi + \tan^{-1} \left(-\frac{x - x_{\mathrm{FOE}}}{y - y_{\mathrm{FOE}}} \right) \right) \tag{6.19}$$

Dabei bezeichnet r die radiale Koordinate und θ den Winkel im komplex-logarithmischen Raum. Der maximale Radius im Originalbild sei r_{src}. Der maximale Radius im komplex-logarithmischen Raum sei $r_{\max}$ und $\theta_{\max}$ sei die Zahl der Pixel in θ-Richtung. Aufgrund der beschränkten Auflösung wird die Transformation nur für den Bereich $r_{\min} \leq r \leq r_{\max}$ und $0 \leq \theta \leq \theta_{\max}$ mit $r_{\min} = \frac{r_{\max}}{\log r_{\mathrm{src}}} \log \frac{\theta_{\max}}{2\pi}$ vorgenommen.

Der optische Fluß wird für das Originalbild berechnet und dann in den komplex-logarithmischen Raum transformiert. Dadurch wird die größere Auflösung des urspünglichen Bildes genutzt. Prinzipiell wäre es denkbar, den optischen Fluß im komplex-logarithmischen Raum zu berechnen. Neben dem hier vorgestellten Ansatz wurde auch versucht, Merkmale im komplex-logarithmischen Raum zu extrahieren. Aufgrund der geringen Auflösung kann dort jedoch nur eine kleine Zahl von markanten Punkten extrahiert werden. Außerdem wurde versucht, ein dichtes Flußfeld für das Originalbild zu berechnen und dieses dann in den komplex-logarithmischen Raum zu transformieren sowie ein dichtes Flußfeld im komplex-logarithmischen Raum zu berechnen. Die Berechnung eines dichten Flußfeldes benötigt jedoch deutlich länger als ein merkmalsbasierter Ansatz. Die Berechnung des Flußfeldes im komplex-logarithmischen Raum setzt voraus, daß der Expansionspunkt exakt bestimmt werden kann. Die Berechnung des optischen Flusses für markante Punkte des Originalbildes und die anschließende Transformation in den komplex-logarithmischen Raum toleriert kleine Ungenauigkeiten bei der Bestimmung des Expansionspunktes. Aus diesen Gründen wurde der hier vorgestellte Ansatz gewählt.

6.5.4 Steuerung des Roboters

Die Transformation des optischen Flusses in den komplex-logarithmischen Raum erleichtert den Vergleich der Flußvektoren. Je näher sich der Betrachter an einem Objekt befindet, desto größer ist der optische Fluß. Der Vergleich der Flußvektoren der linken und rechten periphären visuellen Bereiche kann genutzt werden, um einen Roboter innerhalb eines Korridors zu zentrieren. Um den Roboter zu steuern, wird der optische Fluß der Bereiche, die durch $\theta_{\mathrm{lmin}} \leq \theta \leq \theta_{\mathrm{lmax}}$, $\theta_{\mathrm{rmin}} \leq \theta \leq \theta_{\mathrm{rmax}}$ und $r'_{\min} \leq r \leq r_{\max}$ definiert sind, verwendet. Die Bereiche sind in Abbildung 6.18 graphisch dargestellt. Für die unten beschriebenen Experimente wurden die Werte $\theta_{\mathrm{lmin}} = 45°$, $\theta_{\mathrm{lmax}} = 135°$, $\theta_{\mathrm{rmin}} = 225°$ und $\theta_{\mathrm{rmax}} = 315°$ gewählt. Dadurch wird der optische Fluß, der durch die Decke und den Fußboden verursacht wird, ausgeschlossen. Außerdem wird lediglich der optische Fluß der periphären Bereiche verwendet. Der optische Fluß in der Nähe des Expansionspunktes ist meist sehr klein und kann durch ungenaue Kompensation der rotatorischen Kamerabewegung gestört sein. Daher wurde in

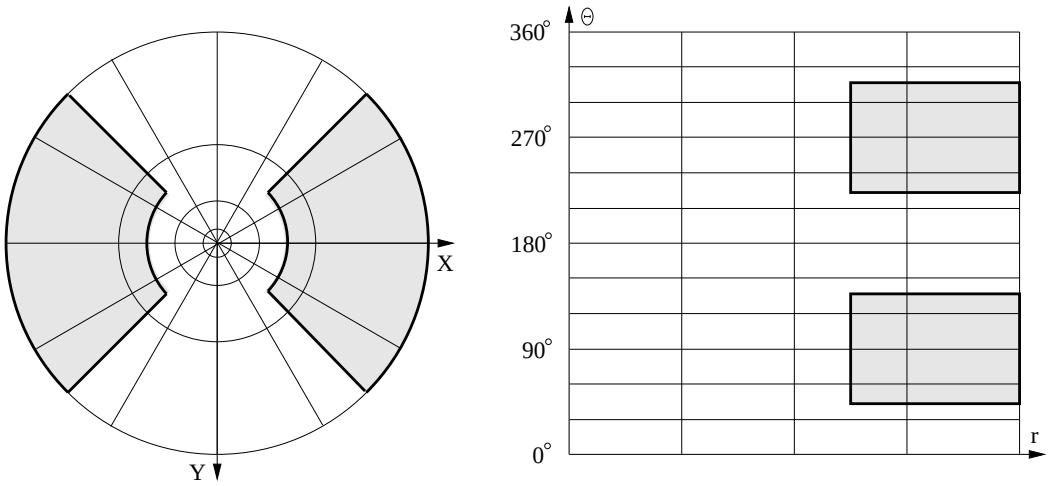


Abbildung 6.18: Bereiche des komplex-logarithmischen Raumes, die zur Steuerung des Roboters eingesetzt wurden.

den unten beschriebenen Experimenten ein minimaler Radius $r'_{\min} = 0.75(r_{\max} - r_{\min}) + r_{\min}$ gewählt. Im folgenden sei f_l und f_r der Median der Flußfelder in den soeben definierten Bereichen.

Zur Steuerung der rotatorischen Geschwindigkeit des Roboters wurde ein PID-Regler [59, 197] eingesetzt. Die Parameter des Reglers wurden experimentell bestimmt. In den unten beschriebenen Experimenten wurden sie auf $k_p = 5.0$, $k_i = 0.1$ und $k_d = 0.005$ gesetzt. Die Steuerung der Drehung des Roboters ist durch

$$\omega_d(t) = k_p e(t) + k_i \int e(\tau) d\tau + k_d \dot{e}(t) \tag{6.20}$$

gegeben. Dabei ist ω_d die gewünschte rotatorische Geschwindigkeit des Roboters und $e(t) = f_l - f_r$ gibt die Differenz zwischen dem optischen Fluß der linken und rechten periphären Bereiche des komplex-logarithmischen Raumes an.

Falls auf einer Seite einmal keine Punktmerkmale extrahiert werden können, dann wird die gewünschte Geschwindigkeit des Roboters so gesetzt, daß er sich in die Richtung der Seite dreht, für die Punktmerkmale extrahiert wurden. Dieses Verhaltensmuster ist durch die Arbeit von Sobey [281] inspiriert. Es ist sicherer, sich in die Richtung zu bewegen, für die der Abstand zum nächsten Hindernis bekannt ist. Bewegt sich der Roboter innerhalb eines Korridors und auf einer Seite fehlen plötzlich jegliche Merkmale, dann bewegt sich der Roboter in Richtung der gegenüberliegenden Wand. Sobald die Punktmerkmale dieser Wand beide Hälften des Sichtbereiches einnehmen, wird der Roboter wieder von dieser Wand abgestoßen.

Die translatorische Geschwindigkeit v_d des Roboters wird ebenfalls geregelt. Dabei wird versucht, den wahrgenommenen optischen Fluß konstant zu halten.

$$v_d = \begin{cases} v_{\max} & \text{falls } \frac{f_d}{f_a} v_{\max} > v_{\max}, \\ \frac{f_d}{f_a} v_{\max} & \text{falls } v_{\min} \leq \frac{f_d}{f_a} v_{\max} \leq v_{\max}, \\ v_{\min} & \text{falls } \frac{f_d}{f_a} v_{\max} < v_{\min} \end{cases} \tag{6.21}$$

Dabei definiert $v_{\min}$ und $v_{\max}$ die minimale und die maximale translatorische Geschwindigkeit des Roboters. Der gewünschte optische Fluß des Originalbildes wird durch f_d spezifiziert. Der tatsächliche optische Fluß f_a wird aus dem Maximum der beiden Werte f_l und f_r berechnet. Dazu wird das Maximum in den ursprünglichen Raum zurücktransformiert. Die Kontrollregel hätte genausogut im komplex-logarithmischen Raum formuliert werden können. Dann hätte der gewünschte Fluß im Originalbild in den komplex-logarithmischen Raum transformiert werden müssen.

Mit der so definierten Kontrollregel wird versucht, den optischen Fluß auf einem konstanten Wert (innerhalb zulässiger Grenzen) zu halten. Dadurch wird der Roboter abgebremst, wenn er sehr nahe an eine Wand kommt und wieder beschleunigt, wenn sich keine nahegelegenen Hindernisse in der Umgebung befinden. Eine ähnliche Kontrollregel (unter Verwendung einer sigmoiden Funktion), um den wahrgenommenen optischen Fluß auf einem konstanten Wert zu halten, wurde von Santos-Victor et al. [263] im Zusammenhang mit ihrem von Bienen inspirierten System realisiert.

Der oben definierte PID-Regler erzeugt in der Regel sehr gerade Bahnen mit wenig Rotationen des Roboters. Daher wurde eine weitere, wesentlich einfachere Steuerung definiert, um das Verfahren zu testen. Der Roboter bewegt sich dabei mit konstanter Geschwindigkeit vorwärts. Falls der optische Fluß auf der rechten Seite größer ist als auf der linken, wird der Roboter mit konstanter Geschwindigkeit nach links gedreht. Der Roboter wird nach rechts gedreht, wenn der optische Fluß auf der linken Seite größer ist als auf der rechten. Nur wenn der optische Fluß auf beiden Seiten gleich groß ist, wird die Rotation des Roboters gestoppt. In der Praxis tauchte dieser Fall jedoch fast nicht auf. Dadurch wird ein oszillierendes Verhalten erreicht, bei dem sich der Roboter ständig nach links oder rechts dreht. Da das hier beschriebene Verfahren auf der Kompensation der rotatorischen Eigenbewegung beruht, ist diese Strategie geeignet, um das Verfahren zu testen.

6.6 Experimente

Für die hier beschriebenen Experimente wurde ein Real World Interface B21 Roboter [244, 243] (Abbildung 6.15), der mit einer Directed Perception Schwenk-Neige-Einheit [68] ausgestattet ist, eingesetzt. Die Bilder wurden mit einer Auflösung von 128×128 digitalisiert. Die Transformation in den komplex-logarithmischen Raum wurde mit den Parametern $r_{\max} = 32$, $r_{\text{src}} = 64$ und $\theta_{\max} = 32$ vorgenommen. Die Experimente wurden in einem 1.41m breiten Korridor durchgeführt. An den Wänden hingen eine Reihe von Bildern. Der Roboter in der Mitte des Korridors ist in Abbildung 6.15 zu sehen.

Es wurden eine Reihe von Experimenten mit der einfachen Steuerung und dem PID-Regler durchgeführt. Für die Experimente mit der einfachen Steuerung wurde eine konstante translatorische Geschwindigkeit von $40\frac{\text{cm}}{\text{s}}$ verwendet. Für die Experimente mit dem PID-Regler wurde die minimale translatorische Geschwindigkeit $v_{\min}$ auf $20\frac{\text{cm}}{\text{s}}$ und die maximale translatorische Geschwindigkeit $v_{\max}$ auf $55\frac{\text{cm}}{\text{s}}$ gesetzt. Der gewünschte optische Fluß f_d wurde auf 3 Pixel gesetzt. Drei Läufe, die mit dem einfachen Regler durchgeführt wurden, sind in Abbildung 6.19 zu sehen. In Abbildung 6.20 sind die Läufe, die mit dem PID-Regler gemacht wurden, zu sehen. Der Pfad des Roboters wurde mit Hilfe der Odometrie aufgezeichnet. Die dargestellten Bilder sind nur ein Teil der Bilder, die während des Experimentes aufgenommen wurden.

Der Korridor ist auf einer Strecke von 3.20m um 0.35m breiter als im restlichen Teil des

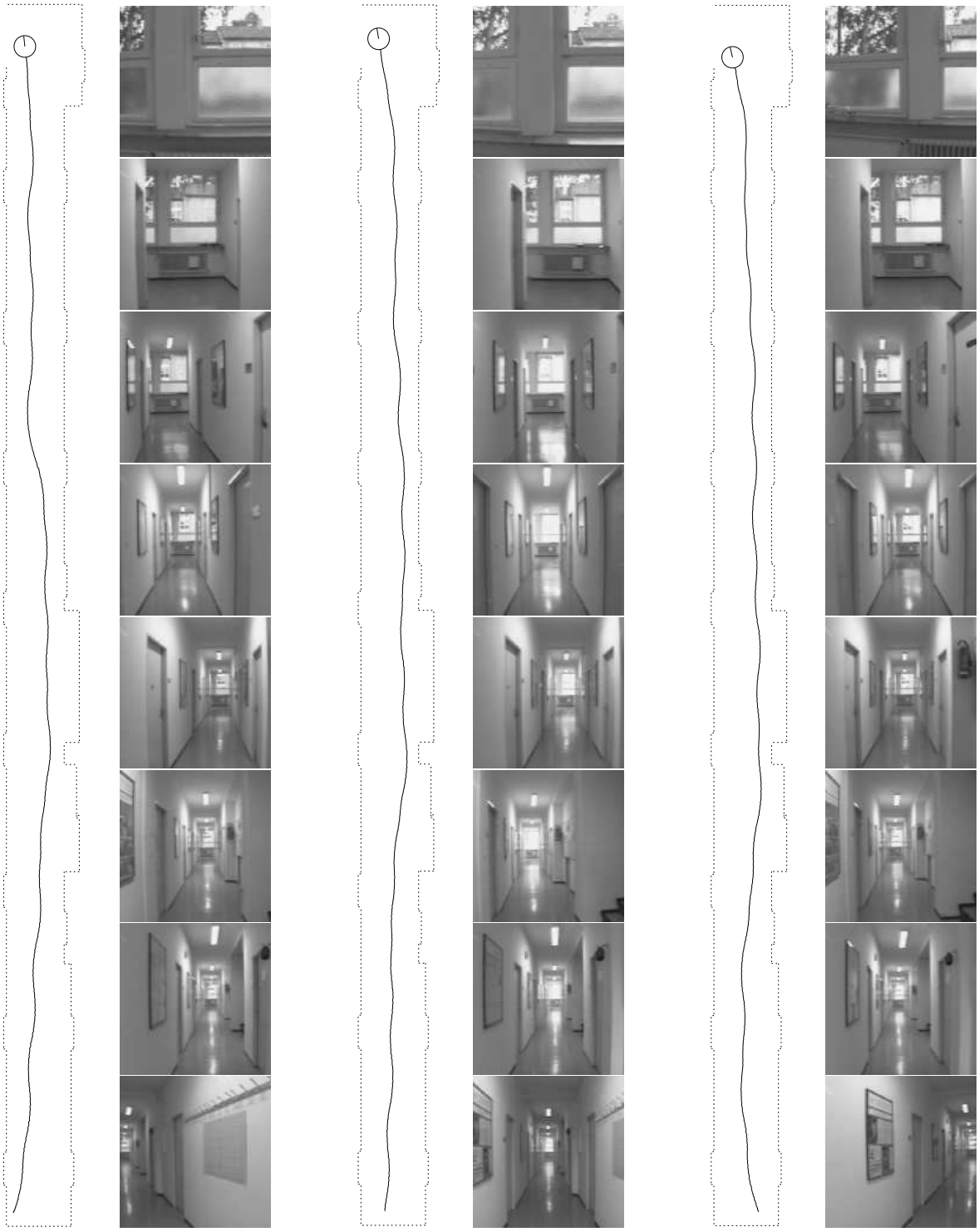


Abbildung 6.19: 3 Experimente, die mit der einfachen Steuerung durchgeführt wurden. Der Pfad des Roboters wurde mit Hilfe der Odometrie aufgezeichnet. Die Bilder entlang des jeweiligen Pfades wurden zur Steuerung des Roboters eingesetzt. Nur ein Teil der verwendeten Bilder ist hier gezeigt. Der Roboter bewegte sich in der Abbildung von unten nach oben. Zur besseren Visualisierung ist der Pfad jeweils in einer manuell erstellten Karte der Umgebung gezeigt.

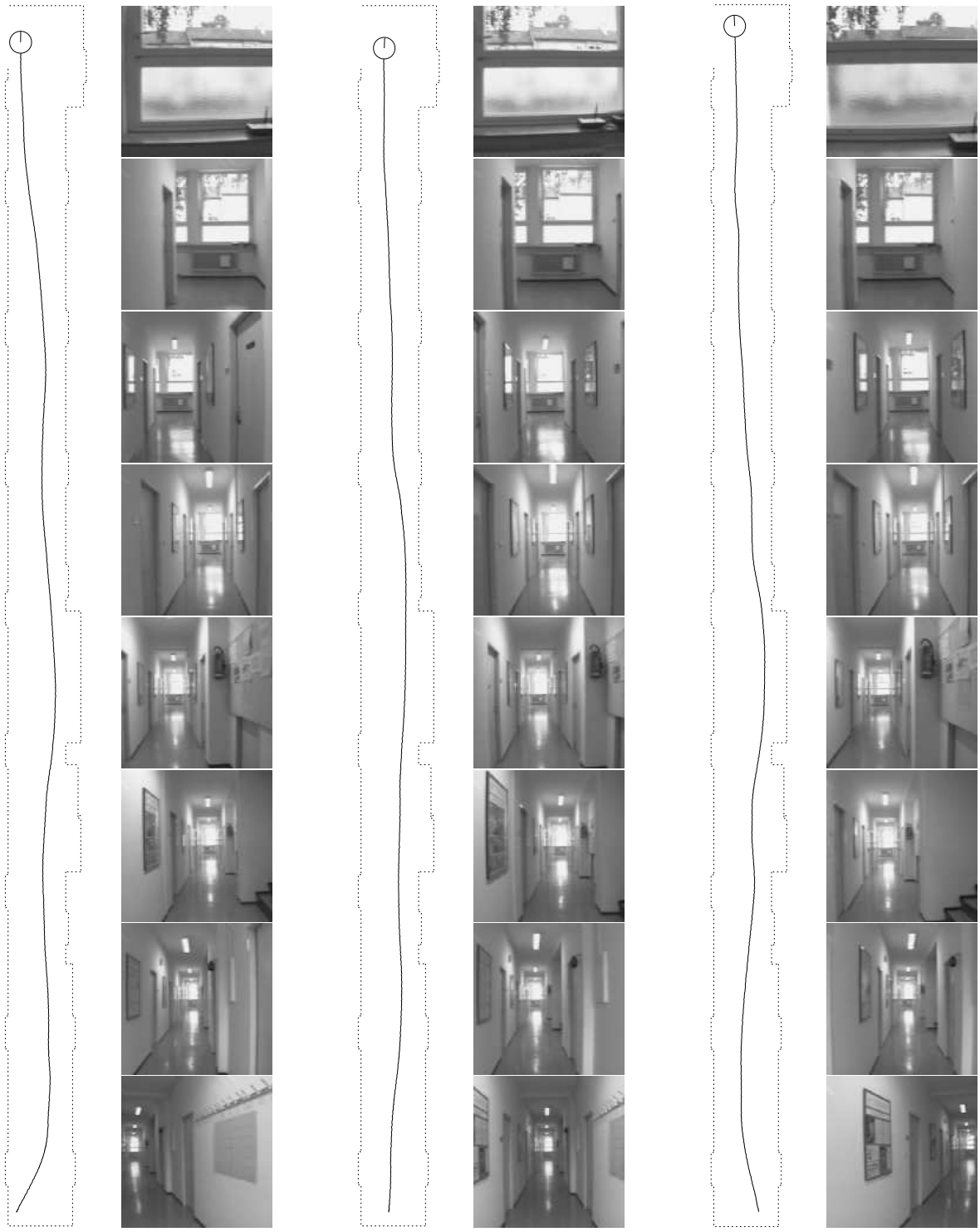


Abbildung 6.20: 3 Experimente, die mit dem PID-Regler durchgeführt wurden. Der Pfad des Roboters wurde mit Hilfe der Odometrie aufgezeichnet. Die Bilder entlang des jeweiligen Pfades wurden zur Steuerung des Roboters eingesetzt. Nur ein Teil der verwendeten Bilder ist hier gezeigt. Der Roboter bewegte sich in der Abbildung von unten nach oben. Zur besseren Visualisierung ist der Pfad jeweils in einer manuell erstellten Karte der Umgebung gezeigt.

Korridors. Kurz vor diesem Bereich befindet sich auf der rechten Seite das Treppenhaus. Wie man in den in Abbildung 6.20 gezeigten Bildern sieht, ist der Pfad des Roboters in diesen Bereichen nach rechts verschoben. Der Pfad, der mit dem PID-Regler aufgezeichnet wurde, ist deutlich glatter als der Pfad, der mit dem einfachen Regler aufgezeichnet wurde. Während des Laufes mit dem einfachen Regler (Startposition in der Mitte) wurden 126 Bilder aufgenommen (1,77 Bilder pro Sekunde). Während des Laufes mit dem PID-Regler (Startposition in der Mitte) wurden 125 Bilder (1,73 Bilder pro Sekunde) aufgenommen. Dabei legte der Roboter eine Strecke von 27,1m bzw. 26,9m zurück. Die zurückgelegte Strecke wurde jeweils mit Hilfe der Odometrie gemessen.

Die durchgeführten Experimente zeigen, daß sich das Verfahren zur visuellen Steuerung eines mobilen Roboters in einem Korridor eignet. Durch eine genaue Analyse des menschlichen visuellen Systems konnte ein System entwickelt werden, das stark durch das biologische System inspiriert wurde und die wesentlichen Eigenschaften des biologischen Vorbildes effektiv nutzt.

6.7 Verwandte Arbeiten

Im folgenden werden verwandte Arbeiten zur visuellen Steuerung mobiler Roboter diskutiert.

6.7.1 Visuelle Steuerung mobiler Roboter

Prinzipiell können aus der Bewegung des Beobachters Abstandsinformationen berechnet werden [319, 212]. Moravec [206] berechnete mit Hilfe einer horizontal bewegbaren Kamera die dreidimensionale Position markanter Punkte in der Umgebung eines Roboters. Die Bewegung der Kamera in horizontaler Richtung wurde durch das Verhalten von Echsen inspiriert. Durch diese dreidimensionale Karte der Umgebung bewegte sich der Roboter mit einer geometrischen Pfadplanung in diskreten Schritten von 1m.

Franceschini et al. [106, 105] entwickelten ein künstliches Facettenauge zur Steuerung eines mobilen Roboters, das durch das Auge der Fliege inspiriert wurde. Zur Steuerung des Roboters wurde zuerst eine translatorische Bewegung und danach die rotatorische Bewegung durchgeführt. Durch diese Art der Bewegung konnten sie die translatorische Komponente des optischen Flusses einfach berechnen.

Brady und Wang [27] bestimmten die dreidimensionale Struktur der Umgebung eines mobilen Roboters, indem sie Eckpunkte extrahierten. Den Abstand der Punkte berechneten sie entweder mit Hilfe einer Stereokamera oder aus der Bewegung des Roboters. Nach Brady und Wang ist es nicht unbedingt notwendig, den Abstand der Punkte zur Kamera explizit auszurechnen. Brauchbare Informationen können auch direkt aus dem optischen Fluß, der Disparität oder der Änderung der Disparität gewonnen werden. In der Tat zeigten Tomasi und Kanade [301, 302], siehe auch Poelman und Kanade [235] und Costeira und Kanade [58], daß man die Struktur einer Szene und die Bewegung der Kamera direkt aus der Bewegung markanter Punkte bestimmen kann, ohne für jeden Punkt die Abstandsinformationen explizit zu berechnen.

Ferrari et al. [89] entwickelten eine mehrschichtige Architektur zur visuellen Steuerung eines mobilen Roboters mit einer Stereokamera. Die Ausgaben der unterschiedlichen Schichten, wie z.B. die Bewegung in Richtung eines Objektes, die Bewegung entlang einer Richtung oder die Hindernisvermeidung, konnten entweder durch feste Priotitäten oder im Falle einer kontinuierlichen Ausgabe durch Multiplikation der Ausgabehistogramme geregelt werden. Fossa

et al. [103] entwickelten ein System zur visuellen Navigation, basierend auf drei Kameras. Zwei Kameras wurden zur Hindernisvermeidung und eine zur Selbstlokalisation eingesetzt.

Horswill [132] zeigte, daß zur visuellen Steuerung eines mobilen Roboters nicht unbedingt ein dreidimensionales Modell der Umgebung erstellt werden muß. Er entwickelte ein einfaches, günstiges und zugleich robustes System, um einen Roboter einen Korridor entlangzusteuern. Sowohl die rotatorische als auch die translatorische Geschwindigkeit wurde gesteuert. Dazu wurde lediglich die Orientierung des Roboters und ein Maß für den Abstand zur linken und rechten Wand aus den visuellen Informationen berechnet. Die Steuerung erfolgte in einer geschlossenen Schleife innerhalb des Korridors. Richtungsänderungen am Ende des Korridors wurden mit einer offenen Regelschleife durchgeführt.

Crespi et al. [60] entwickelten einen speicherbasierten Ansatz zur visuellen Navigation in einem Korridor. In einer Lernphase wurden Bilder kodiert, die mit der Orientierung und der Abweichung des Roboters vom Zentrum des Korridors versehen waren. Während der Steuerung des Roboters wurde das aktuelle Bild mit den gespeicherten Daten verglichen und eine Wahrscheinlichkeitsverteilung für die Orientierung und die seitliche Abweichung berechnet. Dabei wurden etwa drei bis vier Werte für die Orientierung und drei Werte für die seitliche Abweichung verwendet. Durch die berechnete Wahrscheinlichkeitsverteilung wurde die neue translatorische und rotatorische Geschwindigkeit des Roboters spezifiziert.

Sobey [281] entwickelte eine Zick-Zack-Bewegung für einen monokularen Roboter zur Hindernisvermeidung. Mit einer translatorischen Bewegung zur Orientierung wurde als erstes die Entfernung der Objekte auf beiden Seiten des Zielpunktes bestimmt. Alle weiteren Bewegungen wurden mit einem Potentialfeld-Ansatz in eine Richtung ausgeführt, für die sichere Abstandsinformationen vorlagen. Nach jeder translatorischen Bewegung wurde die Richtung um 20° bis 120° geändert, wodurch die Zick-Zack-Bewegung entstand. Abstandsinformationen wurden lediglich für einen horizontalen Streifen im Bild berechnet. Sobey verwendete einen Handlungs-Wahrnehmung-Planungs-Zyklus, bei dem jeweils 16 Bilder für jede translatorische Bewegung aufgenommen wurden.

Cord und Pallmer [56] berechneten Abstandsinformationen markanter Punkte aus einer translatorisch bewegten Kamera. Vogelgesang et al. [309] berechneten mit einem neuronalen Ansatz Abstandsinformationen aus radialen Flußfeldern. Allerdings mußte bei einer Richtungsänderung der Algorithmus neu aufgesetzt werden. Vogelgesang et al. wollten das Verfahren in Zukunft auch auf einem mobilen Roboter testen.

Jochem und Pomerleau [146] entwickelten ein sehr erfolgreiches System zur visuellen Steuerung eines Fahrzeuges, das in der Lage war, auch Manöver, wie z.B. Spurwechsel, durchzuführen. Das System basierte auf einem neuronalen Netz, das mit Verhaltensmustern eines menschlichen Fahrers trainiert wurde. Um das Netz zu trainieren und um Spurwechsel durchzuführen wurden virtuelle Ansichten berechnet. Baluja [14] setzte einen evolutionären Algorithmus ein, um die Gewichte eines neuronalen Netzes zu bestimmen, das ein Fahrzeug steuert.

Košecká [163] entwickelte einen Ansatz zur visuellen Navigation, bei dem die Umgebung als Graph repräsentiert ist. Der Roboter bewegte sich mit Hilfe visueller Informationen von einem Punkt des Graphen zu einem anderen. Dabei wurde angenommen, daß die dreidimensionalen Koordinaten von Landmarken, die zur visuellen Steuerung verwendet wurden, bekannt sind.

Huber und Bülthoff [135] modellierten das visuelle System der Fliege. In Experimenten in der Simulation und mit dem Miniatur-Roboter Khepera zeigten sie, daß die optomotorische Reaktion und das Fixationsverhalten einer Fliege mit einem einzigen Regler realisiert werden

kann.

6.7.2 Steuerung eines mobilen Roboters in einem Korridor

Die folgenden Arbeiten haben einen stärkeren Bezug zu dem hier entwickelten Ansatz zur Steuerung eines mobilen Roboters in einem Korridor. Daher werden sie ausführlicher diskutiert.

Coombs und Roberts [55] entwickelten einen Algorithmus zur Steuerung eines mobilen Roboters mit einem durch Bienen inspirierten Verhalten. Mit dem Algorithmus wurde der Roboter in der Mitte eines Korridors zentriert. Die beiden Kameras waren dabei links und rechts im Winkel von 60° zur Fahrtrichtung angebracht. Coombs und Roberts berechneten den optischen Fluß parallel zur Richtung des Gradienten für das gesamte Bild und setzten ein Histogrammverfahren ein, um den maximalen Fluß im Bild zu bestimmen. Die Differenz zwischen dem maximalen Fluß des linken Bildes und dem maximalen Fluß des rechten Bildes wurde zur Steuerung der rotatorischen Geschwindigkeit des Roboters verwendet. Coombs und Roberts gaben an, daß der Ansatz erfolgreich arbeitet, solange der Expansionspunkt nicht innerhalb des Bildes liegt. Damit dies nicht geschah, wurde die maximale Drehung des Roboters begrenzt. Die Orientierung der Kamera war auch bei einer Drehung des Roboters aufgrund der speziellen Architektur des Roboters stabilisiert. Damit das Verfahren auch bei einer nicht stabilisierten Kamera eingesetzt werden kann, schlugen Coombs und Roberts vor, den optischen Fluß in der Nähe des Expansionspunktes vom periphären optischen Fluß zu subtrahieren.

Coombs et al. [54] setzten zwei Kameras ein. Eine Kamera war mit einer Linse mit großem Öffnungswinkel ausgestattet. Mit dieser Kamera wurde der periphäre optische Fluß bestimmt, der zur Zentrierung des Roboters eingesetzt wurde. Die zweite Kamera besaß eine Linse mit kleinem Öffnungswinkel. Mit dieser Kamera wurde die Zeit bis zu einer Kollision aus der Divergenz des optischen Flusses berechnet. Bevor es zu einer Kollision kam, wurde der Roboter gedreht, so daß er in die umgekehrte Richtung fuhr. Die Blickrichtung der Kamera wurde während einer Drehung des Roboters stabilisiert, indem die Kamera in eine Richtung entgegengesetzt zur Drehung des Roboters gedreht wurde. Wich die Blickrichtung der Kameras zu stark von der Fahrtrichtung des Roboters ab, wurden die Kameras mit einer Sakkade wieder in Fahrtrichtung orientiert.

Santos-Victor et al. [262, 263] statteten einen mobilen Roboter mit zwei seitwärtsgerichteten Kameras aus und realisierten so einen durch Bienen inspirierten Zentrierungsreflex. Dieser Grundreflex wurde um einen Mechanismus erweitert, der die Steuerung auch dann ermöglichte, wenn auf einer Seite einmal kein optischer Fluß vorlag. In diesem Fall sollte der wahrgenommene Fluß konstant gehalten werden. Aufgrund der Anordnung der Kameras nahmen Santos-Victor et al. an, daß nur horizontaler Fluß entsteht, wodurch die Berechnung des optischen Flusses vereinfacht wurde. Die Differenz des durchschnittlichen Flusses der linken und rechten Kamera wurde zur Steuerung des Roboters eingesetzt. Santos-Victor stellten eine Reihe von Randbedingungen auf, die erfüllt sein müssen, um den Roboter auch während rotatorischer Bewegungen zu steuern. Die Randbedingungen können erfüllt werden, indem die minimale translatorische Geschwindigkeit, die maximale rotatorische Geschwindigkeit und die Anordnung der Kameras geeignet gewählt werden. Falls die Randbedingungen einmal nicht erfüllt sind, schlugen Santos-Victor et al. vor, die Steuerung in diesem Fall zu unterdrücken. Die rotatorische Geschwindigkeit steuerten Santos-Victor et al. mit einem PID-Regler. Die translatorische Geschwindigkeit wurde so gesteuert, daß der wahrgenomme-

ne optische Fluß auf einem konstanten Wert gehalten wurde. Ein Überblick über die Arbeiten wurde von Santos-Victor und Sandini [261] gegeben.

Neven und Schöner [213] entwickelten einen verhaltensbasierten Ansatz zur visuellen Navigation, bei dem die einzelnen Verhalten durch die stabilen Zustände eines dynamischen Systems definiert waren. Sie setzten zwei vorwärts gerichtete Kameras ein, eine auf der linken Seite und eine auf der rechten Seite, für die jeweils der optische Fluß berechnet wurde. Aus dem optischen Fluß wurde die Zeit bis zu einer Kollision berechnet, die wiederum zur Steuerung der translatorischen und rotatorischen Geschwindigkeit des Roboters verwendet wurde. Dabei wurde zuerst eine translatorische Bewegung und danach eine rotatorische Bewegung durchgeführt. Dadurch bewegte sich der Roboter auf stückweise geraden Abschnitten. Der optische Fluß in der Bildmitte wurde nicht zur Berechnung der Zeit bis zu einer Kollision verwendet, da der optische Fluß in der Bildmitte zu ungenau ist. Zusätzlich wurde der mittlere Fluß des vertikalen und horizontalen Meridians vom Flußfeld subtrahiert, um eventuell auftretende Rotationen während der translatorischen Bewegung des Roboters zu kompensieren. Falls keine brauchbare Information aus dem optischen Fluß extrahiert werden konnte, wurde der Roboter nur über die Odometrie gesteuert. Zur Berechnung des optischen Flusses wurde ein korrelationsbasierter Ansatz verwendet [38]. Neven et al. [214] erweiterten den Ansatz, indem sie auch zur Berechnung des optischen Flusses ein dynamisches System definierten.

Duchon [73] erweiterten das durch Bienen inspirierte Zentrierungsverhalten um einen Mechanismus, der eine zielgerichtete Bewegung erlaubt. Das Verfahren war durch die Funktion des Hippocampus inspiriert und verwendete externe Landmarken zur Orientierung.

Toepfer et al. [300, 299] und Baratoff et al. [18] setzten ebenfalls die komplex-logarithmische Abbildung zur Steuerung eines Roboters ein. Der Roboter nahm seine Umgebung über eine Kamera wahr, die am Roboter fixiert war. Die digitalisierten Bilder wurden in den komplex-logarithmischen Raum transformiert. Dort wurde mit einem korrelationsbasierten Ansatz [38] der optische Fluß berechnet. Der Fluß der linken und rechten periphären Bereiche wurde zur Steuerung des Roboters eingesetzt, während der Fluß im Zentrum des Bildes zur Detektion von bevorstehenden Kollisionen verwendet wurde. Die Richtung des Roboters wurde um einen festen Winkel von 9° korrigiert, wenn die Differenz zwischen dem optischen Fluß der linken und rechten periphären Bereiche größer als ein Schwellwert wurde. Die translatorische Geschwindigkeit wurde so geregelt, daß die Summe des optischen Flusses der linken und rechten periphären Bereiche konstant blieb. Die Zeit bis zu einer Kollision wurde aus dem optischen Fluß im Zentrum des Bildes berechnet. Zur Auswahl repräsentativer Werte, die aus dem Flußfeld berechnet wurden, experimentierten Toepfer et al. mit dem Median, dem Mittelwert, dem Maximum und der Vernachlässigung eines kleinen Prozentsatzes der Werte. War die Zeit bis zu einer Kollision kleiner als ein bestimmter Schwellwert, so wurde der Roboter um 135° gedreht. Eine Kompensation der rotatorischen Kamerabewegung fand nicht statt. Bisher führten Toepfer et al. lediglich Experimente in der Simulation durch. Sie planen jedoch, das Verfahren auch auf einen echten Roboter zu portieren. Aus den visuellen Informationen möchten sie zukünftig die Eigenbewegung der Kamera berechnen. Dadurch könnte die Kamera auch in andere Richtungen als die Fahrtrichtung orientiert werden, und die Zeit bis zu einer Kollision könnte während beliebiger Kamerabewegungen berechnet werden.

Cameron et al. [42] entwickelten ein selbstorganisierendes neuronales Netz zur visuellen Steuerung eines Roboters, der ein bewegliches Auge bzw. eine bewegliche Kamera besaß. Das neuronale Netz bestand aus mehreren Schichten. In der untersten Schicht sprachen die Neurone auf den optischen Fluß an. An jeder Position der simulierten Retina waren mehrere Detektoren vorhanden, die auf eine ganz bestimmte Richtung des optischen Flusses anspra-

chen. In der zweiten Schicht wurde der durch die Translation des Roboters entstandene optische Fluß extrahiert. Dazu wurde die Rotation der Kamera kompensiert. Dies geschah durch hemmende Signale, die durch die bekannte rotatorische Geschwindigkeit der Kamera erzeugt wurde. Aus dem durch die Translation induzierten Flußfeld wurde, nach einer Normalisierung der Flußvektoren in der dritten Schicht, in der vierten Schicht die Bewegungsrichtung des Roboters, d.h. der Expansionspunkt bestimmt. Aus der bekannten Richtung des Roboters und des aktuellen Flußfeldes wurde in einer weiteren Schicht für jeden Bildpunkt ein Maß für den Abstand des Punktes berechnet. In einer letzten Schicht wurden aus der bekannten Richtung und aus dem normalisierten Flußfeld bewegte Objekte im Sichtbereich des Roboters extrahiert. Die Gewichte des Netzes wurden durch spezielle Lernverfahren jeweils so trainiert, daß in den jeweiligen Schichten die relevante Information extrahiert wurde. So bestand z.B. die Schicht, die die Richtung des Roboters bestimmte, aus einer selbstorganisierenden Merkmalskarte. Zur Extraktion der bewegten Objekte wurde ein Lernverfahren eingesetzt, das Unterschiede zwischen dem optischen Fluß, der aufgrund der Bewegungsrichtung erwartet wurde und dem optischen Fluß, der tatsächlich vorhanden war, hervorhob. Cameron et al. demonstierten mit Experimenten, die in der Simulation durchgeführt wurden, daß mit ihrem Modell Hindernisvermeidung und Objektverfolgung realisiert werden kann.

6.7.3 Unterschiede zu den existierenden Arbeiten

Der hier entwickelte Ansatz arbeitet mit einer einzigen Kamera, bei der der Expansionspunkt sich im Zentrum des Sichtbereiches befindet. Die Daten werden zielgerichtet bestimmt [3], d.h. es werden lediglich die Daten berechnet, die für die Aufgabe notwendig sind. Die Steuerung erfolgt kontinuierlich in einer geschlossenen Schleife und realisiert damit die Verarbeitung visueller Informationen auch während der Bewegung [260]. Der hier vorgestellte Ansatz hebt sich deutlich von den Ansätzen ab, die von Duchon, Coombs et al., Santos-Victor et al., Neven et al. und Toepfer et al. entwickelt wurden. Das Verfahren ist stark durch das visuelle System des Menschen motiviert. Die Information über die Eigenbewegung der Kamera wird aus der Statusinformation des Roboters und der Kamera berechnet. Sie wird dazu verwendet, die rotatorische Bewegung des Roboters zu kompensieren. Da als Statusinformation die über Sensoren wahrgenommenen Geschwindigkeiten anstatt der gewünschten Geschwindigkeiten verwendet wird, besteht die Möglichkeit, daß ein separater Algorithmus die Blickrichtung der Kamera steuert. Die Kamera bleibt dadurch frei für weitere Aufgaben, wie z.B. zur Objekterkennung [307]. Dabei wird natürlich angenommen, daß die Kamera vorwärtsgerichtet bleibt, so daß weiterhin visuelle Informationen links und rechts vom Expansionspunkt wahrgenommen werden. Der Ansatz von Cameron et al. verwendet ebenfalls Informationen über die bekannte Bewegung der Augen bzw. der Kamera, jedoch verwenden Cameron et al. die komplex-logarithmische Abbildung nicht, die den Vergleich der Flußvektoren vereinfacht.

6.8 Zusammenfassung

Ein Ansatz zur visuellen Steuerung eines mobilen Roboters, inspiriert durch Eigenschaften des visuellen Systems des Menschen, wurde vorgestellt. Bilder werden über eine Kamera aufgenommen und digitalisiert. Aus den Bildern werden markante Punkte extrahiert und für die Punktmerkmale der optische Fluß berechnet. Die Statusinformationen des Roboters werden verwendet, um die Eigenbewegung der Kamera zu berechnen. Die bekannte Eigenbewegung

der Kamera wird dazu verwendet, die rotatorische Bewegung aus den Bildern herauszurechnen. Dadurch entsteht lediglich optischer Fluß, der durch die translatorische Bewegung der Kamera induziert wurde. Der optische Fluß wird in den komplex-logarithmischen Raum transformiert. Er ist ein direktes Maß für die Entfernung der Punktmerkmale. Der optische Fluß der linken und rechten periphären Bereiche wird verwendet, um den Roboter zu steuern. Experimentelle Ergebnisse wurden für die Steuerung eines mobilen Roboters in einem Korridor präsentiert.

Damit wurde gezeigt, wie visuelle Merkmale zur Steuerung eines mobilen Roboters eingesetzt werden können. Zur Durchführung der Experimente wurde der Moravec-Operator eingesetzt, da dieser sehr schnell zu berechnen ist. Prinzipiell könnte der Moravec-Operator durch einen der evolvierten Operatoren ersetzt werden. Aufgrund der hohen Komplexität der evolvierten Operatoren, die daher sehr lange zur Berechnung benötigen, war dies jedoch nicht möglich. Gezeigt wurde ferner, daß sich durch eine genaue Analyse des menschlichen visuellen Systems die wesentlichen Eigenschaften des biologischen Vorbildes in einer technischen Realisierung effektiv nutzen lassen.

Kapitel 7

Zusammenfassung

Die vorliegende Arbeit liefert einen Beitrag über die Beschaffenheit des Suchraumes von genetischem Programmieren, zur Selbstlokalisation mobiler Roboter, zur evolutionären Robotik und zum Einsatz von genetischem Programmieren in der Bildverarbeitung. Speziell wurde genetisches Programmieren zur Evolution von Merkmalsdetektoren eingesetzt. Ein weiterer Beitrag der Arbeit liegt in der Entwicklung eines Algorithmus zur visuellen Steuerung eines mobilen Roboters basierend auf Eigenschaften des visuellen Systems des Menschen.

Über genetisches Programmieren existieren noch relativ wenig theoretische Erkenntnisse. Als Beitrag zur Erweiterung des Wissens über genetisches Programmieren wurde der Suchraum von genetischem Programmieren untersucht. Im Vergleich zu genetischen Algorithmen ist der Suchraum von genetischem Programmieren anders beschaffen. Während bei genetischen Algorithmen meist jedem Phänotyp genau ein Genotyp entspricht, existieren bei genetischem Programmieren aufgrund von Introns für jeden Phänotyp mehrere Genotypen. Daher kann es je nach gewählter Kodierung vorkommen, daß das Erreichen des Optimums nicht so unwahrscheinlich ist, wie es die Größe des Suchraumes vermuten läßt. Es wurde ein Vergleich zum Suchraum der natürlichen Evolution aufgestellt. Eine Übertragung der Eigenschaften der natürlichen Evolution auf die künstliche Evolution könnte dazu beitragen, daß mit genetischem Programmieren größere Fortschritte erreicht werden. Im besonderen könnte es von Vorteil sein, eine Repräsentation einzusetzen, bei der Individuen, die häufig vorkommende Verhalten kodieren, innerhalb eines kleinen Radius von jedem zufällig ausgewählten Individuum liegen. Unterschiedliche Individuen, die ein und dasselbe Verhalten kodieren, sollten zufällig im Suchraum verteilt sein. Außerdem sollte für ein gegebenes Individuum möglichst weit entfernte Individuen über neutrale Mutationen erreichbar sein.

Genetisches Programmieren wurde zur Lokalisation eines mobilen Roboters in einer simulierten Umgebung eingesetzt. Der Roboter hat seine Umgebung durch einen Ring von Abstandssensoren, wie sie z.B. durch den Einsatz eines Laser-Scanners gegeben sind, wahrgenommen. Mit genetischem Programmieren wurde eine Funktion evolviert, die die gegebenen Abstandsinformationen auf die Position des Roboters abbildet. Zur Reduktion des Suchraumes wurden geometrische Momente aus der Abstandsverteilung berechnet und als Terminal-Symbole verwendet. Als elementare Funktionen wurden arithmetische, trigonometrische Funktionen und ein bedingter Befehl verwendet. Die evolvierte Funktion stellt ein internes Modell der Umgebung dar. Es ist vermutlich das erste Mal, daß ein evolutionärer Algorithmus zur Lokalisation eines mobilen Roboters eingesetzt wurde. In dem Experiment wurde lediglich eine Assoziation zwischen Umgebungsmerkmalen und der Position des Robo-

ters hergestellt, es erfolgte jedoch keine Steuerung des Roboters.

Anschließend wurde gezeigt, daß mit genetischem Programmieren auch ein Programm zur Steuerung eines mobilen Roboters evolviert werden kann. Evolviert wurde eine reaktive, verhaltensbasierte Kontrollarchitektur. Die aktuellen Sensorwerte wurden eingesetzt, um den Roboter entlang eines Ganges zur steuern. In der Simulation wurde nach einer geeigneten Repräsentation für das Problem gesucht, die es erlaubt, das Experiment auch mit dem echten Service-Roboter zu wiederholen. Die Simulation ermöglicht die Durchführung einer Vielzahl von Experimenten in kürzester Zeit. Aufgrund von unvorhersehbaren Wechselwirkungen zwischen dem Roboter und seiner Umgebung kann es jedoch vorkommen, daß sich eine evolvierte Kontrollarchitektur völlig anders verhält, als man es zunächst erwartet. In einem zweimonatigen Experiment wurde gezeigt, daß mit genetischem Programmieren eine Kontrollarchitektur auch für den echten Service-Roboter evolviert werden kann. Dabei nahm der Roboter seine Umgebung über sechs virtuelle Abstandssensoren wahr. Außer diesen Sensoren wurden zwei Konstanten und der Wert des Sensors, der den kürzesten Abstand anzeigt, als Terminal-Symbole verwendet. Der Roboter bewegte sich mit konstanter Geschwindigkeit vorwärts. Aufgabe der Kontrollarchitektur war es, die Rotation des Roboters so zu steuern, daß dieser nicht gegen eine Wand fährt. Als elementare Funktionen wurden ein bedingter Befehl und eine Verbindungsfunktion eingesetzt. Mit genetischem Programmieren wurde auf diese Weise eine reaktive Kontrollarchitektur zur Steuerung eines mobilen Roboters evolviert. Durch den Einsatz evolutionärer Algorithmen in der Robotik, der sogenannten evolutionären Robotik, könnte die Programmierung autonomer mobiler Roboter in Zukunft automatisiert werden.

In den folgenden Experimenten wurde genetisches Programmieren eingesetzt, um visuelle Merkmalsdetektoren zu evolvieren. Dabei wurde jeweils das Eingabebild als Terminal-Symbol und eine Reihe von Operatoren zur Bildverarbeitung als elementare Funktionen eingesetzt. Als erstes wurde ein Kantendetektor evolviert, der den Canny-Kantendetektor approximiert. Untersucht wurde dabei die Frage, ob es möglich ist, daß die Evolution einen sehr komplexen Operator aus einfachen Operationen aufbauen kann. Obwohl nur sehr einfache Funktionen zur Verfügung gestellt wurden, konnte dennoch eine Approximation des Canny-Kantendetektors evolviert werden. Danach wurde mit genetischem Programmieren ein Operator zur Extraktion markanter Punkte evolviert. Was ein markanter Punkt ist, hängt von der aktuellen Umgebung und von dem Algorithmus ab, der die extrahierten Punkte schließlich verarbeitet. In ersten Experimenten wurde zur Berechnung der Fitneß der Individuen ein bekannter Operator, der Moravec-Interest-Operator, eingesetzt. Danach wurde ein Operator zur Berechnung des optischen Flusses evolviert. Zur Berechnung der Fitneß der Individuen wurden eine Reihe von Kriterien definiert, die der evolvierte Operator erfüllen sollte. Um einen Operator zur Berechnung des optischen Flusses zu evolvieren, wurden die folgenden Kriterien verwendet: 1) Die Zahl der Flußvektoren sollte groß sein. 2) Die Qualität der Korrespondenzen sollte gut sein. 3) Das Verhältnis zwischen Zahl der Flußvektoren und Zahl der extrahierten Punkte sollte groß sein. 4) Die Zuordnung der Punkte sollte eindeutig sein. 5) Das Flußfeld sollte möglichst glatt sein. 6) Das Flußfeld sollte eine maximale Dichte besitzen. Der Einsatz evolutionärer Algorithmen und ganz besonders genetischen Programmierens ist interessant, da in der Bildverarbeitung oft mit einem Baukasten aus Standardoperatoren gearbeitet wird. Zur Lösung eines gegebenen Problems werden die vorhandenen Operatoren in geeigneter Weise kombiniert. Mit genetischem Programmieren besteht nun die Möglichkeit, diesen Prozeß zu automatisieren.

Wie die extrahierten Merkmale zur Steuerung eines mobilen Roboters genutzt werden

können, wurde im folgenden gezeigt. Es wurde ein Algorithmus zur visuellen Steuerung eines mobilen Roboters entwickelt, der durch die Bewegung visueller Merkmale in der Umgebung des Roboters den optischen Fluß berechnet. Der Algorithmus wurde durch das visuelle System des Menschen inspiriert, das als Produkt der natürlichen Evolution optimal an seine Umgebung angepaßt ist. Der hier entwickelte Algorithmus zur visuellen Steuerung setzt die komplex-logarithmische Abbildung ein, um den optischen Fluß der linken und rechten periphären Bereiche zu vergleichen. Die Differenz des optischen Flusses des linken und rechten Bereiches wird zur Zentrierung des Roboters in einem Korridor verwendet. Da die komplex-logarithmische Abbildung nur für einen translatorisch bewegten Betrachter definiert ist, wird die rotatorische Bewegung der Kamera kompensiert. Zur Kompensation der rotatorischen Bewegung wird die bekannte Eigenbewegung der Kamera genutzt, die aus den Statusinformationen des Roboters und der Schwenk-Neige-Einheit berechnet wird.

Dieser Algorithmus zur Steuerung eines technischen Systems wurde durch die folgenden Merkmale des visuellen Systems inspiriert: Die Abbildung der retinalen Signale auf den primären visuellen Kortex kann durch eine komplex-logarithmische Abbildung beschrieben werden. Dabei kreuzen sich die Sehnerven im Gehirn, so daß die Signale aus dem rechten Sichtbereich die linke Gehirnhälfte und die des linken Sichtbereiches die rechte Gehirnhälfte erreichen. Die Information über die Augen- und Kopfbewegung wird eingesetzt, um die Umgebung als stationär wahrzunehmen. Der Mechanismus, mit dem dies realisiert werden kann, wird in biologischen Systemen als Reafferenzprinzip bezeichnet. Bei dem hier entwickelten Algorithmus entsprechen den Kopfbewegungen die Bewegungen der Schwenk-Neige-Einheit und den Körperbewegungen die Bewegungen des Roboters. Die Transformation in dem komplex-logarithmischen Raum wird immer um den Expansionspunkt vorgenommen. Dies kann als Bewegung der Augen aufgefaßt werden, d.h. hier ist der Blick immer auf den Expansionspunkt gerichtet.

Somit wurde gezeigt, wie sich Erkenntnisse, die vom visuellen System des Menschen gewonnen wurden, auf ein künstliches visuelles System zur Steuerung eines mobilen Roboters übertragen lassen. Dies ist vor allem dadurch interessant, da Leistungen, die mit der des natürlichen visuellen Systems vergleichbar sind, weder in der Robotik noch in der maschinellen Bildverarbeitung realisiert wurden. Durch eine umfangreiche Analyse biologischer Systeme lassen sich wesentliche Erkenntnisse gewinnen, die in der Robotik oder in der Bildverarbeitung eingesetzt werden können.

Anhang A

Koza-Tableaus

Im folgenden sind für die in der vorliegenden Arbeit mit genetischem Programmieren durchgeführten Experimente die sogenannten Koza-Tableaus aufgeführt. Diese Tabellen wurden von Koza eingeführt [165] und geben in kompakter Form alle wichtigen Informationen über das Experiment an.

Als erstes wird das Ziel des Experimentes beschrieben. Danach werden die verwendeten Terminal-Symbole und die elementaren Funktionen genannt. Darauf folgt die Beschreibung der Fitneßtests und die Berechnung der Fitneß. Bei den Experimenten, bei denen das Minimum einer Funktion gesucht wird, ist stattdessen die Beschreibung einer Fehlerfunktion angegeben. In Kozas Arbeiten wird rohe Fitneß, standardisierte Fitneß und angepaßte Fitneß unterschieden. Da die Berechnung der Fitneß in den folgenden Tabellen nur mit Worten beschrieben ist, wird diese Unterscheidung hier nicht gemacht. Nach der Fitneß wird eine mögliche Verpackung (*Wrapper*) der Individuen beschrieben. Je nach Aufgabe kann es vorkommen, daß die Ausgabe des Individuums noch nachbearbeitet oder transformiert wird. Es kann auch vorkommen, daß ein Individuum in einer Schleife solange angewendet wird, bis ein Abbruchkriterium erfüllt ist. Dies ist in der Rubrik Verpackung eingetragen. Am Ende der Tabelle sind schließlich die wichtigsten Parameter des Experimentes aufgeführt, das ist die Größe der Population M und die Zahl der berechneten Generationen G.

A.1 Evolution einer Abbildung zur Lokalisation eines mobilen Roboters

Ziel:	Eine Abbildung zu finden, die aus den gegebenen Sensorwerten die Position des Roboters berechnet.
Terminal-Symbole :	Sensorik: Momente $\texttt{M}_j$ (j = {1,2,3,4}), Zentrale Momente $\texttt{CM}_j$ (j = {2,3,4,5}), Momente der Differenzen $\texttt{DM}_j$ (j = {1,2,3,4}) Konstanten: `RAND` (mit Bereich [0,1)), `RAND10` (mit Bereich [0,10)) und `RAND100` (mit Bereich [0,100)).

Elementare Funktionen:	Arithmetische Funktionen:
	`+`, `-`, `*`, `/`
	Trigonometrische Funktionen:
	`COS`, `SIN`, `TAN`,
	`ASIN`, `ACOS`, `ATAN`, `ATAN2`
	und der bedingte Befehl `IFLTE`.

Fitneßtests:	1000 Positionen mit zugehörigen Sensorwerten.
Fehlermaß:	Summe der quadrierten Differenzen zwischen tatsächlicher und berechneter Position des Roboters.
Verpackung:	Beschränkung der Position auf einen Bereich, der doppelt so groß ist wie der Grundriß der Umgebung.
Parameter:	$M = 1000$. $G = 5000$.

A.2 Evolution einer Steuerung für einen mobilen Roboter

Ziel:	Eine Kontrollarchitektur für einen mobilen Service-Roboter zu evolvieren. Der Roboter bewegt sich mit einer konstanten Geschwindigkeit vorwärts. Aufgabe der Kontrollarchitektur ist es, den Roboter zu steuern, so daß Hindernisse vermieden werden und sich der Roboter möglichst geradeaus bewegt.
Terminal-Symbole :	Sensorik: Die 24 Ultraschall-Sensoren des Roboters werden zu 6 virtuellen Sensoren zusammengefaßt: `FL` (vorne links), `FM` (Mitte vorne), `FR` (vorne rechts), `BL` (hinten links), `BM` (Mitte hinten), `BR` (hinten rechts), Minimum aller 24 realen Sensoren `SS`. Steuerung: Drehung nach links `TL`, Drehung nach rechts `TR`, Stop der Drehung `RHALT`. Konstanten: Gewünschter Abstand zur Wand `EDG`, kleinste, noch sichere Entfernung zur Wand `MSD`.
Elementare Funktionen:	Der bedingte Befehl `IFLTE` und die Funktion `PROGN2`, die zwei Befehle miteinander verbindet.
Fitneßtests:	1-3 Tests des Roboters in unterschiedlichen Situationen, d.h. mit unterschiedlicher Anfangsposition.
Fitneß:	Individuen, die nur wenig Drehungen ausführen und die zur Verfügung stehende Zeit ausnutzen, d.h. Hindernisse vermeiden, erhalten eine hohe Fitneß.
Verpackung:	Der Kontrollalgorithmus wird solange in einer Schleife ausgeführt, bis die zur Verfügung stehende Zeit abgelaufen ist.
Parameter:	$M = 75$. $G = 50$.

A.3 Evolution von Kantendetektoren

Ziel:	Einen Kantendetektor zu evolvieren, der die Ausgabe des Canny-Operators approximiert.
Terminal-Symbole :	Bild `Image` (Größe 128×128) mit Pixelwerten im Bereich $[0, 1]$.
elementare Funktionen:	Unäre Funktionen: `Neg`, `Abs`, `ShiftL`, `ShiftR`, `ShiftU`, `ShiftD`, `UThr`. Binäre Funktionen: `Thr`, `+`, `-`, `*`, `/`, `Min`, `Max`.
Fitneßtests:	5 Bilder aus der Umgebung eines mobilen Roboters.
Fehlermaß:	Summe der quadrierten Differenzen zwischen gewünschter und tatsächlicher Ausgabe des evolvierten Kantendetektors über alle Pixelwerte. Ein weiterer Term erzeugt eine schlechte Fitneß für Individuen, die eine homogene Ausgabe liefern.
Verpackung:	Keine.
Parameter:	$M = 4000$. $G = 25$.

A.4 Evolution von Operatoren zur Extraktion markanter Punkte

Ziel:	Einen Operator zur evolvieren, der die Ausgabe des Moravec-Operators approximiert und somit markante Punkte aus den Bildern extrahiert.
Terminal-Symbole :	Bild `Image` (Größe 128×128) mit Pixelwerten im Bereich $[0, 1]$.
elementare Funktionen:	Unäre Funktionen: `Neg`, `Abs`, `Square`, `Avg4x4`, `Sum4x4`, `ShiftL`, `ShiftR`, `ShiftU`, `ShiftD`. Binäre Funktionen: `+`, `-`, `*`, `/`, `Min`, `Max`. Funktionen mit N Argumenten ($N \in \{3, 4\}$): `SumN`, `PiN`, `MinN`, `MaxN`.
Fitneßtests:	5 Bilder aus der Umgebung eines mobilen Roboters.
Fehlermaß:	Summe der quadrierten Differenzen zwischen gewünschter und tatsächlicher Ausgabe des evolvierten Operators zur Extraktion markanter Punkte über alle Pixelwerte. Ein weiterer Term erzeugt eine schlechte Fitneß für Individuen, die eine homogene Ausgabe liefern.
Verpackung:	Keine.
Parameter:	$M = 4000$. $G = 50$.

A.5 Evolution von Operatoren zur Berechnung des optischen Flusses

Ziel:	Einen Operator zu evolvieren der Punkte extrahiert, die sich zur Berechnung des optischen Flusses eignen.
Terminal-Symbole :	Bild `Image` (Größe 128×128) mit Pixelwerten im Bereich $[0, 1]$.
elementare Funktionen:	Unäre Funktionen: `Abs`, `Sqrt`, `Square`, `Gabor0`, ..., `Gabor7`, `Avg3x3`, `Median3x3`,`Gauss`, `GaussDx`, `GaussDy`, `ShiftL`, `ShiftR`, `ShiftU`, `ShiftD`, `ZeroCross`, `Erosion`, `Dilation`. Binäre Funktionen: `+`, `-`, `*`, `/`.
Fitneßtests:	4 Bilder aus der Umgebung eines mobilen Roboters.
Fitneß:	Kombination aus 6 verschiedenen Qualitätskriterien: 1) Zahl der Flußvektoren, 2) Qualität der Korrespondenzen, 3) Verhältnis von Zahl der Flußvektoren zur Zahl der extrahierten Punkte, 4) Eindeutigkeit der Korrespondenzen, 5) Glattheit des Flußfeldes und 6) maximale Dichte des Flußfeldes.
Verpackung:	Punkte, die kein lokales Maximum sind, werden auf Null gesetzt. Extrahiert werden alle Punkte, die größer als ein Schwellwert sind.
Parameter:	$M = 500$. $G = 50$.

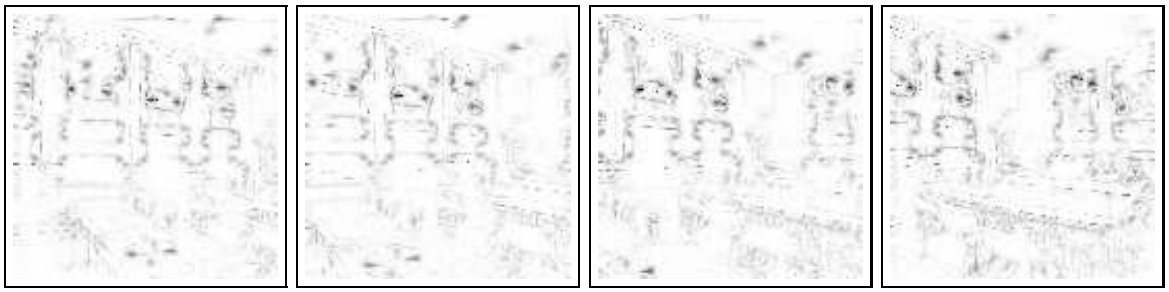
Anhang B

Existierende Operatoren zur Berechnung des optischen Flusses

Auf den folgenden Seiten sind die Ergebnisse zusammengefaßt, die mit existierenden Operatoren zur Extraktion markanter Punkte erreicht wurden. Der Reihe nach sind für jeden Operator die Eingabebilder, darunter die Antwort des Operators, gefolgt von den markanten Punkten, die von dem jeweiligen Operator extrahiert wurden, dargestellt. In der untersten Zeile ist jeweils der optische Fluß zu sehen, der mit Hilfe dieses Operators berechnet wurde. Diese Daten sind hier zusammengefaßt, um einen qualitativen Vergleich mit den evolvierten Operatoren zur Berechnung des optischen Flusses zu ermöglichen. Für einen quantitativen Vergleich kann die Fitneß der Operatoren herangezogen werden. Die Fitneßwerte der Operatoren wurden bereits in Abschnitt 5.4.5 verglichen.

Eingabebilder:

Antwort des Operators:



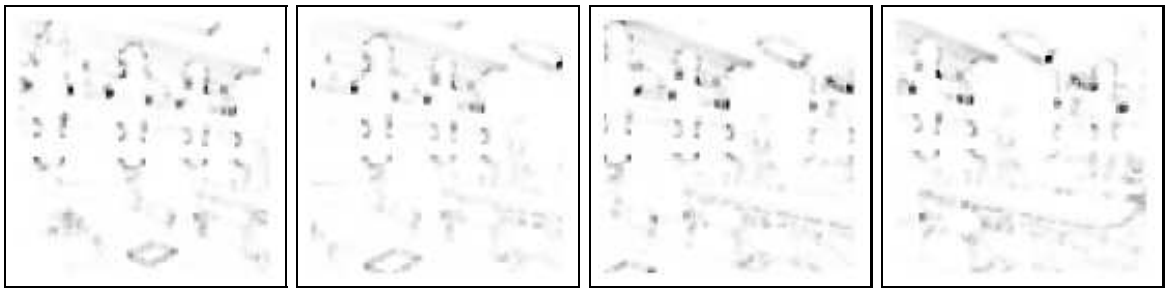
Vom Operator extrahierte Punkte:

Der optische Fluß:

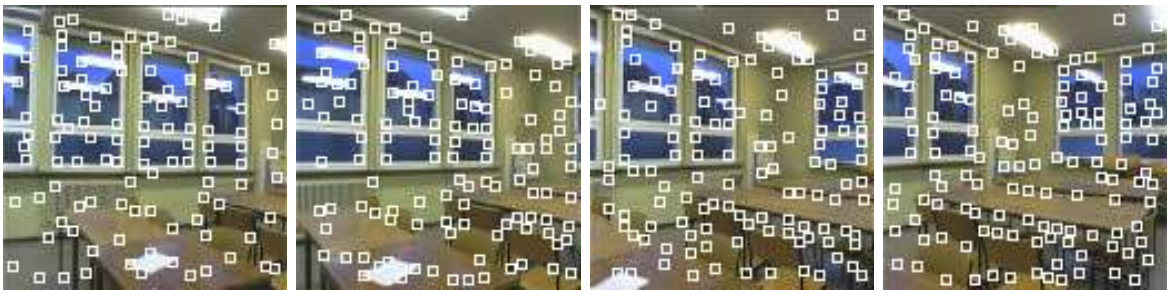
Abbildung B.1: Kitchen-Rosenfeld-Eckendetektor [274].

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:



Der optische Fluß:

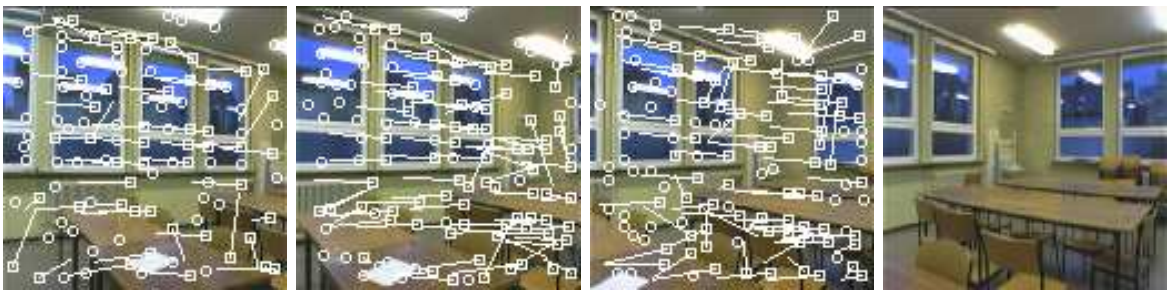
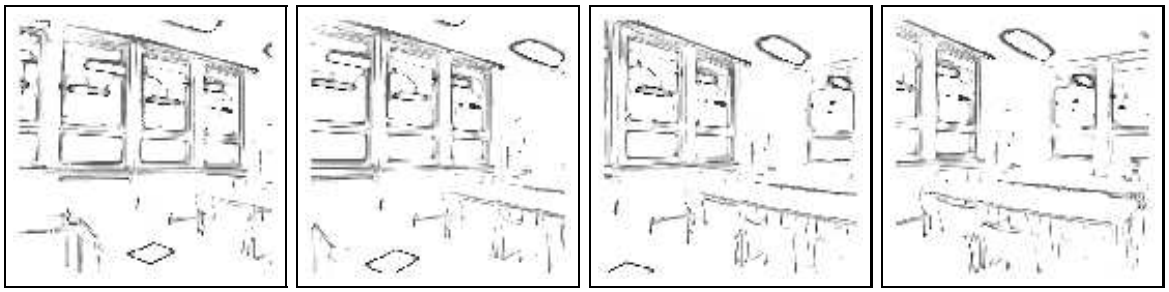


Abbildung B.2: Moravec-Operator [205, 206].

Eingabebilder:

Antwort des Operators:



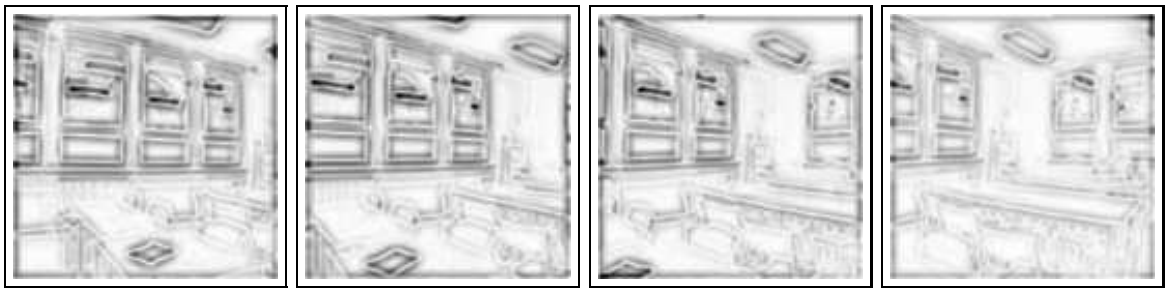
Vom Operator extrahierte Punkte:

Der optische Fluß:

Abbildung B.3: SUSAN-Merkmalsoperator [280].

Eingabebilder:

Antwort des Operators:



Vom Operator extrahierte Punkte:

Der optische Fluß:

Abbildung B.4: Extraktion markanter Punkte mit der Differenz von Gabor-Filtern [330].

Literaturverzeichnis

[1] Adobe Systems. *PostScript Language Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[2] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilerbau Teil 1 und Teil 2.* Addison-Wesley (Deutschland) GmbH, Bonn, 1988.

[3] Y. Aloimonos, editor. *Active Perception.* Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1993.

[4] J. A. Anderson, A. Pellionisz, and E. Rosenfeld, editors. *Neurocomputing 2: Directions for Research.* The MIT Press, Cambridge, Massachusetts, 1990.

[5] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* The MIT Press, Cambridge, Massachusetts, 1988.

[6] D. Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 477–494, Cambridge, Massachusetts, 1994. The MIT Press.

[7] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. on Neural Networks*, 5(1):54–65, January 1994.

[8] R. C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[9] R. C. Arkin. *Behavior-Based Robotics.* The MIT Press, Cambridge, Massachusetts, 1998.

[10] P. Bak. *How Nature Works: The Science of Self-Organized Criticality.* Copernicus An imprint of Springer-Verlag, New York, 1996.

[11] K. Balakrishnan and V. Honavar. Spatial learning for robot localization. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proc. of the Second Int. Conference on Genetic Programming, 1997*, pages 389–397, San Francisco, California, 1997. Morgan Kaufmann Publishers. also available as Technical Report TR #97-08, Artificial Intelligence Research Group, Department of Computer Science, Iowa State University.

[12] D. H. Ballard. *An Introduction to Natural Computation.* The MIT Press, Cambridge, Massachusetts, 1997.

[13] D. H. Ballard, R. C. Nelson, and B. Yamauchi. Animate vision. *Optics News*, 15(5):9,17–25, 1989.

[14] S. Baluja. Evolution of an artificial neural network based autonomous land vehicle controller. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):450–463, 1996.

[15] W. Banzhaf. Genotype-phenotype-mapping and neutral variation – a case study in genetic programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III. Proc. of the Int. Conference on Evolutionary Computation*, pages 322–332, Berlin, 1994. Springer-Verlag.

[16] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann Publishers, San Francisco, California, 1998.

[17] W. Banzhaf, P. Nordin, and M. Olmer. Generating adaptive behavior using function regression within genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proc. of the Second Int. Conference on Genetic Programming, 1997*, pages 35–43, San Francisco, California, 1997. Morgan Kaufmann Publishers.

[18] G. Baratoff, C. Toepfer, M. Wende, and H. Neumann. Real-time navigation and obstacle avoidance from optical flow on a space-variant map. In *Special Session on Biomimetic Robotics of ISIC/CIRA/ISAS '98 NIST, Gaithersburg, MD, USA*, September 1998.

[19] H. G. Barrow. Learning receptive fields. In *Proc. of the 1st Int. Conference on Neural Networks*, volume 4, pages 115–121. IEEE, 1987.

[20] H. Bässmann and P. W. Besslich. *Konturorientierte Verfahren in der digitalen Bildverarbeitung.* Springer-Verlag, Berlin, Heidelberg, 1989.

[21] M. Beetz, W. Burgard, D. Fox, and A. B. Cremers. Integrating active localization into high-level robot control systems. *Robotics and Autonomous Systems*, 23:205–220, 1998.

[22] G. A. Bekey. Biologically inspired control of autonomous robots. *Robotics and Autonomous Systems*, 18:21–31, 1996.

[23] S. M. Bhandarkar, Y. Zhang, and W. D. Potter. An edge detection technique using genetic algorithm-based optimization. *Pattern Recognition*, 27(9):1159–1180, 1994.

[24] A. K. Bhattacharjya and B. Roysam. Joint solution of low, intermediate, and high-level vision tasks by evolutionary optimization: Application to computer vision at low SNR. *IEEE Trans. on Neural Networks*, 5(1):83–95, January 1994.

[25] Z. Biró and T. Ziemke. Evolution of visually-guided approach behaviour in recurrent artificial neural network robot controllers. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proc. of the Fifth Int. Conference on Simulation of Adaptive Behavior*, pages 73–76. The MIT Press, 1998.

[26] E. W. Bonabeau. Why do we need artificial lifeΓ In C. G. Langton, editor, *Artificial Life: An Overview*, pages 303–325, Cambridge, Massachusetts, 1995. The MIT Press.

[27] M. Brady and H. Wang. Vision for mobile robots. *Phil. Trans. R. Soc. Lond. B*, 337:341–350, 1992.

[28] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* The MIT Press, Cambridge, Massachusetts, 1984.

[29] T. Bräunl, S. Feyrer, W. Rapf, and M. Reinhardt. *Parallel Bildverarbeitung.* Addison-Wesley, Bonn, 1995.

[30] J. Bray. *Matrox Meteor FrameGrabber Driver v1.4 (a port and partial rewrite of the FreeBSD Matrox Meteor FrameGrabber Driver, written by Jim Lowe and Mark Tinguely)*, 1996.

[31] I. N. Bronštein und K. A. Semendjajew. *Taschenbuch der Mathematik.* Verlag Harri Deutsch, Thun und Frankfurt/Main, 24. Auflage, 1989.

[32] R. R. Brooks, S. S. Iyengar, and J. Chen. Automatic correlation and calibration of noisy sensor readings using elite genetic algorithms. *Artificial Intelligence*, 84:339–354, 1996.

[33] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[34] R. A. Brooks. Artifical life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems: Proc. of the First European Conference on Artificial Life*, pages 3–10, Cambridge, MA, 1992. The MIT Press.

[35] R. A. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20:291–304, 1997.

[36] A. Browne, editor. *Neural Network Perspectives on Cognition and Adaptive Robotics.* Institute of Physics Publishing, Bristol, 1997.

[37] U. Büker und G. Hartmann. Eckenmerkmale für robuste Erkennung und Fovealisierung in einem Robot Vision System. In B. Jähne, P. Geißler, H. Haußecker und F. Hering (Hrsg.), *Mustererkennung 1996*, pages 590–597, Berlin, 1996. Springer-Verlag.

[38] H. Bülthoff, J. Little, and T. Poggio. A parallel algorithm for real-time computation of optical flow. *Nature*, 337(9):549–553, February 1989.

[39] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. of the 14th National Conference on Artificial Intelligence*, pages 896–901. AAAI Press/MIT Press, 1996.

[40] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proc. of the 15th Int. Joint Conference on Artificial Intelligence Volume 2*, pages 1346–1352, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

[41] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner. Emergence of functional modularity in robots. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proc. of the Fifth Int. Conference on Simulation of Adaptive Behavior*, pages 497–504. The MIT Press, 1998.

[42] S. Cameron, S. Grossberg, and F. H. Guenther. A self-organizing neural network architecture for navigation using optic flow. *Neural Computation*, 10:313–352, 1998.

[43] R. H. S. Carpenter. *Movements of the Eyes.* Pion Limited, London, second edition, 1988.

[44] P. Cavanagh. Size and position invariance in the visual system. *Perception*, 7:167–177, 1978.

[45] P. Cavanagh. Image transforms in the visual system. In P. C. Dogwell and T. Caelli, editors, *Figural Synthesis*, pages 185–218, Hillsdale, NJ, 1984. Erlbaum.

[46] D. J. Chalmers. The evolution of learning: An experiment in genetic connectionism. In D. S. Touretzky, editor, *Proc. of the 1990 Connectionist Models Summer School*, pages 81–90, 1990.

[47] C.-H. Chao and A. P. Dhawan. Edge detection using a hopfield neural network. *Optical Engineering*, 33(11):3739–3747, November 1994.

[48] T. Chen, W.-C. Lin, and C.-T. Chen. Artificial neural networks for 3-D motion analysis - part I: Rigid motion. *IEEE Trans. on Neural Networks*, 6(6):1386–1393, November 1995.

[49] T. Chen, W.-C. Lin, and C.-T. Chen. Artificial neural networks for 3-D motion analysis - part II: Nonrigid motion. *IEEE Trans. on Neural Networks*, 6(6):1394–1401, November 1995.

[50] D. Cliff and S. Bullock. Adding "foveal vision" to Wilson's animat. *Adaptive Behavior*, 2(1):49–72, 1993.

[51] D. Cliff, P. Husbands, and I. Harvey. Evolving visually guided robots. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 374–383. The MIT Press, 1993.

[52] M. Colombetti and M. Dorigo. Learning to control an autonomous robot by distributed genetic algorithms. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 305–312. The MIT Press, 1993.

[53] M. Colombetti, M. Dorigo, and G. Borghi. Behavior analysis and training – a methodology for behavior enginerring. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):365–380, 1996.

[54] D. Coombs, M. Herman, T.-H. Hong, and M. Nashman. Real-time obstacle avoidance using central flow divergence, and peripheral flow. *IEEE Trans. on Robotics and Automation*, 14(1):49–59, February 1998.

[55] D. Coombs and K. Roberts. "bee-bot": using peripheral optical flow to avoid obstacles. In D. Casasent, editor, *Intelligent Robots and Computer Vision XI, Proc. of the Society of Photo-Optical Instrumentation Engineers*, pages 714–721, 1992.

[56] T. Cord und D. Pallmer. Axiales Motion Stereo zur Abstandmessung für mobile Roboter. In P. Levi und T. Bräunl (Hrsg.), *Autonome Mobile Systeme 1994*, pages 89–94, Berlin, 1994. Springer-Verlag.

[57] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.

[58] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Int. Conference on Computer Vision*, pages 1071–1076. IEEE, 1995.

[59] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1989.

[60] B. Crespi, C. Furlanello, and L. Stringa. A memory-based approach to navigation. *Biological Cybernetics*, 69:385–393, 1993.

[61] J. L. Crowley, F. Wallner, and B. Schiele. Position estimation using principal components of range data. *Robotics and Autonomous Systems*, 23:267–276, 1998.

[62] J. M. Daida, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 279–284, Cambridge, Massachusetts, 1996. The MIT Press.

[63] J. M. Daida, J. D. Hommes, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from SAR images of arctic ice. In P. J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming Volume II*, pages 417–442, Cambridge, Massachusetts, 1996. The MIT Press.

[64] C. Darwin. *The Origin of Species*. Oxford University Press, Oxford, England, 1996.

[65] J. G. Daugman. Complete discrete 2-D gabor transforms by neural networks for image analysis and compression. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988.

[66] A. G. Deakin and D. F. Yates. GP tools available on the web: A first encounter. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, page 420, Cambridge, Massachusetts, 1996. The MIT Press.

[67] D. C. Dennett. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Allen Lane, The Penguin Press, Harmondsworth, Middlesex, England, 1995.

[68] Directed Perception. *Computer Controlled Pan-Tilt Unit Model PTU User's Manual Version 1.07b (Preliminary)*. Burlingame, California, 1994.

[69] J.-Y. Donnart and J.-A. Meyer. Learning reactive and planning rules in a motivationally autonomous animat. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):381–395, June 1996.

[70] M. Dorigo and M. Colombetti. *Robot Shaping: An Experiment in Behavior Engineering.* The MIT Press, Cambridge, Massachusetts, 1998.

[71] M. Dorigo and U. Schnepf. Genetics-based machine learning and behavior-based robotics: A new synthesis. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(1):141–154, 1993.

[72] J. E. Dowling. *The retina: an approachable part of the brain.* The Belknap Press of Harvard University Press, Cambridge, Massachusetts, 1987.

[73] A. P. Duchon. Maze navigation using optical flow. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals To Animats 4, Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 224–232, Cambridge, Massachusetts, 1996. The MIT Press.

[74] M. Ebner. *Robuste Extraktion sich bewegender Objekte aus einer mit dynamischer Kamera aufgenommenen Videosequenz, Diplomarbeit Nr. 1355.* Fakultät Informatik, IPVR, Universität Stuttgart, Mai 1996.

[75] M. Ebner. Evolution of hierarchical translation-invariant feature detectors with an application to character recognition. In E. Paulus und F. M. Wahl (Hrsg.), *Mustererkennung 1997*, pages 456–463, Berlin, 1997. Springer-Verlag.

[76] M. Ebner. On the evolution of edge detectors for robot vision using genetic programming. In H.-M. Groß (Hrgs.), *Workshop SOAVE '97 - Selbstorganisation von Adaptivem Verhalten, VDI Reihe 8 Nr. 663*, pages 127–134, Düsseldorf, 1997. VDI Verlag.

[77] M. Ebner. Evolution of a control architecture for a mobile robot. In M. Sipper, D. Mange, and A. Pérez-Uribe, editors, *Proc. of the Second Int. Conference on Evolvable Systems: From Biology to Hardware (ICES 98), Lausanne, Switzerland*, pages 303–310, Berlin, 1998. Springer-Verlag.

[78] M. Ebner. Extraction of moving objects with a moving mobile robot. In M. Á. Salichs and A. Halme, editors, *3rd IFAC Symposium on Intelligent Autonomous Vehicles, Volume II, Madrid, Spain*, pages 749–754. Elsevier Science, 1998.

[79] M. Ebner. On the evolution of interest operators using genetic programming. In R. Poli, W. B. Langdon, M. Schoenauer, T. Fogarty, and W. Banzhaf, editors, *Late Breaking Papers at EuroGP'98: the First European Workshop on Genetic Programming*, pages 6–10, Paris, France, 14-15 April 1998. The University of Birmingham, UK.

[80] M. Ebner. On the search space of genetic programming and its relation to nature's search space. *Proc. of the 1999 Congress on Evolutionary Computation, Washington, D.C., July 6-9*, 1999.

[81] M. Ebner. Evolving an environment model for robot localization. In R. Poli and P. Nordin and W. B. Langdon and T. C. Fogarty, editors, *Proc. of the Second European Workshop on Genetic Programming EuroGP'99, Göteborg, Sweden*, pages 184–192, Berlin, 1999. Springer-Verlag.

[82] M. Ebner and A. Zell. Centering behavior with a mobile robot using monocular foveated vision (submitted to Robotics and Autonomous Systems), 1997.

[83] M. Ebner and A. Zell. Evolution of a control architecture under real world constraints using genetic programming (unpublished), 1997.

[84] M. Ebner and A. Zell. Motion tracking with a fast moving active camera (unpublished), 1998.

[85] M. Ebner and A. Zell. Visual navigation using ego-motion information. In S. Posch und H. Ritter (Hrsg.), *Dynamische Perzeption: Workshop der GI-Fachgruppe 1.0.4 Bildverstehen in Kooperation mit dem SFB 360: Situierte Künstliche Kommunikatoren, Bielefeld, Juni 1998*, pages 55–62, Sankt Augustin, June 1998. Infix.

[86] M. Ebner and A. Zell. Evolving a behavior-based control architecture - from simulations to the real world. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proc. of the Genetic and Evolutionary Computation Conference, July 13-17, 1999, Orlando, Florida*, San Francisco, California, 1999. Morgan Kaufman Publishers.

[87] M. Ebner and A. Zell. Evolving a task specific image operator. In R. Poli and H.-M. Voigt and S. Cagnoni and D. Corne and G. D. Smith and T. C. Fogarty, editors, *Joint Proc. of the First European Workshops on Evolutionary Image Analysis, Signal Processing and Telecommunications EvoIASP'99 and EuroEcTel'99, Göteborg, Sweden*, pages 74–89, Berlin, 1999. Springer-Verlag.

[88] P. M. Ferguson, editor. *Motif Reference Manual for OSF/Motif Release 1.2.* O'Reilly & Associates, Inc., Sebastopol, California, second edition, April 1994.

[89] M. Ferrari, M. Fossa, E. Grosso, M. Magrassi, and G. Sandini. A practical implementation of a multilevel architecture for vision-based navigation. In *Proc. of the 5th Int. Conference Advanced Robotics, Pisa, Italy*, pages 1092–1098. IEEE, June 1991.

[90] S. Feyrer und A. Zell. Videobasierte Erkennung von Personen und hindernisvermeidendes Folgeverhalten mit einem mobilen Serviceroboter. In H. Wörn, R. Dillmann und D. Henrich (Hrsg.), *Autonome Mobile Systeme 1998*, pages 75–84, Berlin, 1998. Springer-Verlag.

[91] D. Flanagan, editor. *X Toolkit Intrinsics Reference Manual for Version 11 R4/R5.* O'Reilly & Associates, Inc., Sebastopol, California, January 1995.

[92] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proc. of the Third Int. Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 421–430. The MIT Press, 1994.

[93] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):396–407, 1996.

[94] D. Floreano and F. Mondada. Evolution of plastic neurocontrollers for situated agents. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From animals to animats 4: Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 402–410. The MIT Press, 1996.

[95] D. Floreano and S. Nolfi. God save the red queen! Competition in co-evolutionary robotics. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proc. of the Second Int. Conference on Genetic Programming, July 13-16, 1997*, pages 398–406, San Francisco, California, 1997. Morgan Kaufmann Publishers.

[96] D. Floreano, S. Nolfi, and F. Mondada. Competitive co-evolutionary robotics: From theory to practice. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proc. of the Fifth Int. Conference on Simulation of Adaptive Behavior*, pages 515–524. The MIT Press, 1998.

[97] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. on Neural Networks*, 5(1):3–14, January 1994.

[98] D. B. Fogel, editor. *Evolutionary Computation: The Fossil Record*. IEEE Press, Piscataway, NJ, 1998.

[99] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, Inc., New York, 1966.

[100] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice. Second Edition in C.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1996.

[101] C. M. Fonseca and P. J. Fleming. Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction. *First IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 414:45–52, 1995.

[102] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[103] M. Fossa, E. Grosso, F. Ferrari, M. Magrassi, G. Sandini, and M. Zapendouski. A visually guided mobile robot acting in indoor environments. In *Proc. of the IEEE Workshop on Applications of Computer Vision, Palm Springs, USA, 1992*, pages 308–316. IEEE, 1992.

[104] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[105] N. Franceschini, J. M. Pichon, and C. Blanes. From insect vision to robot vision. *Phil. Trans. R. Soc. Lond. B*, 337:283–294, 1992.

[106] N. Franceschini, J.-M. Pichon, and C. Blanes. Real time visuomotor control: from flies to robots. In *Proc. of the 5th Int. Conference Advanced Robotics, Pisa, Italy*, pages 931–935. IEEE, June 1991.

[107] J. Frazier and R. Nevatia. Detecting moving objects from a moving platform. In *Proc. of the DARPA Image Understanding Workshop, Pittsburgh, PA*, pages 348–355, 1990.

[108] D. Glowacki. *Geppetto Genetic Programming System*, 1993.

[109] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[110] D. E. Goldberg and U.-M. O'Reilly. Where does the good stuff go, and whyΓ How contextual semantics influences program structure in simple genetic programming. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming: Proc. of the First European Workshop, EuroGP'98, Paris, France, April 14-15*, pages 16–36, Berlin, 1998. Springer-Verlag.

[111] T. Gomi, editor. *Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER '97).* AAI Books, Ontario, Kanada, 1997.

[112] T. Gomi, editor. *Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER '98).* AAI Books, Ontario, Canada, 1998.

[113] R. C. Gonzales and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[114] G. H. Granlund. In search of a general picture processing operator. *Computer Graphics and Image Processing*, 8:155–173, 1978.

[115] R. Greiner and R. Isukapalli. Learning to select useful landmarks. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):437–449, June 1996.

[116] F. Gruau. Genetic synthesis of modular neural networks. In S. Forrest, editor, *Proc. of the Fifth Int. Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, July 17-21, 1993*, pages 318–325, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[117] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Int. Conference on Intelligent Robots and Systems, Victoria, B.C.*, October 1998.

[118] M. Hahn. *Digitale Bildverarbeitung Teil 1: Grundlagen, Vorlesungsmanuskript WS 90/91*, Februar 1991.

[119] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[120] C. Harris and B. Buxton. Evolving edge detectors with genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 309–314, Cambridge, Massachusetts, 1996. The MIT Press.

[121] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[122] I. Harvey. Artificial evolution and real robots. *Artificial Life and Robotics*, 1(1):35–38, 1997.

[123] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proc. of the Third Int. Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 392–401. The MIT Press, 1994.

[124] I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 364–373. The MIT Press, 1993.

[125] D. Heller and P. M. Ferguson, editors. *Motif Programming Manual for OSF/Motif Release 1.2.* O'Reilly & Associates, Inc., Sebastopol, California, second edition, February 1994.

[126] J. H. Holland. *Adaptation in natural and artifical systems: an introductory analysis with applications to biology, control, and artificial intelligence.* The MIT Press, Cambridge, Massachusetts, 1992.

[127] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[128] B. K. Paul Horn. *Robot Vision.* The MIT Press, Cambridge, Massachusetts, 1986.

[129] G. A. Horridge. A theory of insect vision: velocity parallax. *Proc. R. Soc. Lond. B*, 229:13–27, 1986.

[130] G. A. Horridge. The evolution of visual processing and the construction of seeing systems. *Proc. R. Soc. Lond. B*, 230:279–292, 1987.

[131] G. A. Horridge. What can engineers learn from insect visionΓ *Phil. Trans. R. Soc. Lond. B*, 337:271–282, 1992.

[132] I. Horswill. A simple, cheap, and robust visual navigation system. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 129–136. The MIT Press, 1993.

[133] H.-J. Lee and H.-C. Deng. Three-frame corner matching and moving object extraction in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 52:210–238, 1990.

[134] Y. Huang, K. Palaniappan, X. Zhuang, and J. E. Cavanaugh. Opticl flow field segmentation and motion estimation using a robust genetic partitioning algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(12):1177–1190, December 1995.

[135] S. A. Huber and H. H. Bülthoff. Simulation and robot implementation of visual orientation behaviors of flies. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proc. of the Fifth Int. Conference on Simulation of Adaptive Behavior*, pages 77–85. The MIT Press, 1994.

[136] S. A. Huber, H. A. Mallot, and H. H. Bülthoff. Evolution of the sensorimotor control in an autonomous agent. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From animals to animats 4: Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 449–457. The MIT Press, 1996. also available as Technical Report No. 035, Max-Planck-Institut für biologische Kybernetik, Spemannstraße 38, 72076 Tübingen.

[137] P. Husbands and J.-A. Meyer, editors, *Evolutionary Robotics: Proc. of the First European Workshop, EvoRobot'98, Paris, France, April 14-15.* Springer-Verlag, Berlin, 1998.

[138] B. Hussain and M. R. Kabuka. A novel feature recognition neural network and its application to character recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(1):98–106, January 1994.

[139] R. Jain. Segmenation of frame sequences obtained by a moving observer. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6(5):624–629, September 1984.

[140] R. Jain, S. L. Bartlett, and N. O'Brian. Motion stereo using ego-motion complex logarithmic mapping. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(3):356–369, 1987.

[141] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision.* McGraw-Hill, Inc., New York, 1995.

[142] N. Jakobi. The minimal simulation approach to evolutionary robotics. In T. Gomi, editor, *Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER '98)*, pages 133–190, Ontario, Canada, 1998. AAI Books.

[143] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proc. of the Third European Conference on Artificial Life, Granada, Spain, June 4-6, 1995*, pages 704–720, Berlin, 1995. Springer-Verlag.

[144] F. Janabi-Sharifi and W. J. Wilson. Automatic selection of image features for visual servoing. *IEEE Trans. on Robotics and Automation*, 13(6):890–903, December 1997.

[145] H. Jeong and C. I. Kim. Adaptive determination of filter scales for edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(5):579–585, May 1992.

[146] T. Jochem and D. Pomerleau. Life in the fast lane: The evolution of an adaptive vehicle control system. *AI Magazine*, 17(2):11–50, 1996.

[147] M. P. Johnson, P. Maes, and T. Darrell. Evolving visual routines. In R. A. Brooks and P. Maes, editors, *Artificial Life IV, Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, Cambridge, Massachusetts, 1994. The MIT Press.

[148] J. L. Jones and A. M. Flynn. *Mobile Robots - Inspiration to Implementation.* A K Peters, Wellesley, Massachusetss, 1993.

[149] A. Joshi and C.-H. Lee. On modelling the retina using neural networks. In *Proc. Int. Joint Conf. on Neural Networks 91, Singapore*, pages 2343–2348. IEEE, 1991.

[150] A. Joshi and C.-H. Lee. Backpropagation learns Marr's operator. *Biological Cybernetics*, 70:65–73, 1993.

[151] T. Kalinke und W. von Seelen. Entropie als Maß des lokalen Informationsgehalts in Bildern zur Realisierung einer Aufmerksamkeitssteuerung. In B. Jähne, P. Geißler, H. Haußecker und F. Hering (Hrsg.), *Mustererkennung 1996*, pages 627–634, Berlin, 1996. Springer-Verlag.

[152] T. Kalinke and W. von Seelen. A neural network for symmetry-based object detection and tracking. In B. Jähne, P. Geißler, H. Haußecker und F. Hering (Hrsg.), *Mustererkennung 1996*, pages 37–44. Springer-Verlag, 1996.

[153] A. J. Katz and P. R. Thrift. Generating image filters for target recognition by genetic learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):906–910, September 1994.

[154] S. A. Kauffman. *The Origins of Order. Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[155] R. E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 116–122, Cambridge, Massachusetts, 1996. The MIT Press.

[156] B. W. Kernighan and D. M. Ritchie. *Programmieren in C mit dem C-Reference Manual in deutscher Sprache.* Carl Hanser Verlag, München Wien, 1983.

[157] K. Kikuchi and F. Hara. Evolutionary design of morphology and intelligence in robotic system using genetic programming. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proc. of the Fifth Int. Conference on Simulation of Adaptive Behavior*, pages 540–545. The MIT Press, 1998.

[158] S. Kleiman, D. Shah, and B. Smaalders. *Programming with Threads.* SunSoft Press, A Prentice Hall Title, Upper Saddle River, New Jersey 07458, 1996.

[159] M. Kleiner. *Biologische visuelle Merkmalsdetektion.* Seminar Biologische Strategien für mobile Roboter, WS 1997/98. Eberhard-Karls-Universität Tübingen, 1997.

[160] J. J. Koenderink and A. J. van Doorn. Receptive field families. *Biological Cybernetics*, 63:291–297, 1990.

[161] J. J. Koenderink and A. J. van Doorn. Generic neighborhood operators. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(6):597–605, June 1992.

[162] T. Kohonen. Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biological Cybernetics*, 75:281–291, 1996.

[163] J. Košecká. Visually guided navigation. *Robotics and Autonomous Systems*, 21:37–50, 1997.

[164] B. Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Prentice-Hall, Inc., A Simon & Schuster Company, Englewood Cliffs, New Jersey, 1992.

[165] J. R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.

[166] J. R. Koza. Simultaneous discovery of reusable detectors and subroutines using genetic programming. In S. Forrest, editor, *Proc. of the Fifth Int. Conference on Genetic Algorithms*, pages 295–302. Morgan Kaufmann, 1993.

[167] J. R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.

[168] S. Kröner and H. Schulz-Mirbach. Fast adaptive calculation of invariant features. In G. Sagerer, S. Posch und F. Kummert (Hrsg.), *Mustererkennung 1995*, pages 23–35, Berlin, 1995. Springer-Verlag.

[169] A. Kurz. Constructing maps for mobile robot navigation based on ultrasonic range data. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(2):233–242, April 1996.

[170] J. Lampinen and E. Oja. Distortion tolerant pattern recognition based on self-organizing feature extraction. *IEEE Trans. on Neural Networks*, 6(3):539–547, May 1995.

[171] W. B. Langdon and R. Poli. Fitness causes bloat: Mutation. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming: Proc. of the First European Workshop, EuroGP'98, Paris, France, April 14-15*, pages 37–48, Berlin, 1998. Springer-Verlag.

[172] W. B. Langdon and R. Poli. Genetic programming bloat with dynamic fitness. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming: Proc. of the First European Workshop, EuroGP'98, Paris, France, April 14-15*, pages 97–112, Berlin, 1998. Springer-Verlag.

[173] C. G. Langton. *Artificial Life: An Overview.* The MIT Press, Cambridge, Massachusetts, 1995.

[174] J. Langwald und G. Hirzinger. Robuste Echtzeitverfolgung kreisförmiger Objektmerkmale mittels Hough-Transformation. In E. Paulus und F. M. Wahl (Hrsg.), *Musterer-kennung 1997*, pages 509–516, Berlin, 1997. Springer-Verlag.

[175] W.-P. Lee, J. Hallam, and H. H. Lund. Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *Proc. of the IEEE 4th Int. Conference on Evolutionary Computation*, pages 501–506. IEEE Press, 1997.

[176] M. Lehrer, M. V. Srinivasan, S. W. Zhang, and G. A. Horridge. Motion cues provide the bee's visual world with a third dimension. *Nature*, 332(24):356–357, March 1988.

[177] P. Levi und T. Bräunl (Hrsg.). *Autonome Mobile Systeme 1994.* Springer-Verlag, Berlin, 1994.

[178] M. S. Lew, T. S. Huang, and K. Wong. Learning and feature selection in stereo matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):869–881, September 1994.

[179] R. Linsker. From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proc. Natl. Acad. Sci. USA*, 83:8390–8394, November 1986.

[180] R. Linsker. From basic network principles to neural architecture: Emergence of orientation columns. *Proc. Natl. Acad. Sci. USA*, 83:8779–8783, November 1986.

[181] R. Linsker. From basic network principles to neural architecture: Emergence of spatial-opponent cells. *Proc. Natl. Acad. Sci. USA*, 83:7508–7512, October 1986.

[182] R. Linsker. Self-organization in a perceptual network. *Computer*, 21:105–117, 1988.

[183] S. B. Lippman. *C++ Primer.* Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1991.

[184] R. Lohmann. Structure evolution and incomplete induction. *Biological Cybernetics*, 69:319–326, 1993.

[185] R. Lohmann. Selforganization by evolution strategy in visual systems. In H.-M. Voigt, H. Mühlenbein, and H.-P. Schwefel, editors, *Evolution and Optimization '89*, pages 61–68. Akademie-Verlag, 1990.

[186] R. Lohmann. *Bionische Verfahren zur Entwicklung visueller Systeme.* Dissertation, Technische Universität Berlin, Fachbereich 10 Verfahrenstechnik und Energietechnik, 1991.

[187] R. Lohmann. Structure evolution in neural systems. In B. Souček and the IRIS Group, editors, *Dynamic, Genetic, and Chaotic Programming*, pages 395–411. John Wiley & Sons, Inc., 1992.

[188] L. R. Lopez and R. E. Smith. Evolving artificial insect brains for artificial compound eye robotics. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 425–430. The MIT Press, 1993.

[189] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1991.

[190] H. A. Mallot. *Sehen und die Verarbeitung visueller Information, Eine Einführung.* Vieweg, Braunschweig, 1998.

[191] V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Trans. on Neural Networks*, 5(1):39–53, January 1994.

[192] B. S. Manjunath, C. Shekhar, and R. Chellappa. A new approach to image feature detection with applications. *Pattern Recohnition*, 29(4):627–640, 1996.

[193] D. Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[194] M. Matarić and D. Cliff. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19:67–83, 1996.

[195] J. Maynard Smith. *The Theory of Evolution*. Cambridge University Press, Cambridge, England, 1993.

[196] J. L. McClelland, D. E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*. The MIT Press, Cambridge, Massachusetts, 1986.

[197] P. J. McKerrow. *Introduction to Robotics*. Addison-Wesley Publishing Company, Sydney, 1991.

[198] L. A. Meeden. An incremental approach to developing intelligent neural network controllers for robots. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):474–485, 1996.

[199] J.-A. Meyer, P. Husbands, and I. Harvey. Evolutionary robotics: a survey of applications and problems. In P. Husbands and J.-A. Meyer, editors, *Proc. of the First European Workshop on Evolutionary Robotics, Paris, April 1998*, pages 1–21, 1998.

[200] O. Miglino, H. H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2:417–434, 1996.

[201] J. del R. Millán. Rapid, safe, and incremental learning of navigation strategies. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):408–420, June 1996.

[202] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, Massachusetts, 1996.

[203] R. Möller. *Wahrnehmung durch Vorhersage. Eine Konzeption der handlungsorientierten Wahrnehmung*. Dissertation, Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau, Juni 1996.

[204] F. Mondada and D. Floreano. Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, 16:183–195, 1995.

[205] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. of the 5th Int. Joint Conference on Artificial Intelligence*, Vision–1: page 584, 1977.

[206] H. P. Moravec. *Ostacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Computer Science Department, Stanford University, No. STAN-CS-80-813 and AIM-340, September 1980.

[207] M. S. Mousavi and R. J. Schalkoff. ANN implementation of stereo vision using a multi-layer feedback architecture. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(8):1220–1237, 1994.

[208] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):449–459, May 1994.

[209] W. Nachtigall. *Bionik: Grundlagen und Beispiele für Ingenieure und Naturwissenschaftler*. Springer-Verlag, Berlin, 1998.

[210] D. Nair and J. K. Aggarwal. Moving obstacle detection from a navigating robot. *IEEE Trans. on Robotics and Automation*, 14(3):404–416, June 1998.

[211] N. M. Nasrabadi and C. Y. Choo. Hopfield network for stereo vision correspondence. *IEEE Trans. on Neural Networks*, 3(1):5–13, January 1992.

[212] R. Nevatia. Depth mesurement by motion stereo. *Computer Graphics and Image Processing*, 5:203–214, 1976.

[213] H. Neven and G. Schöner. Dynamics parametrically controlled by image correlations organize robot navigation. *Biological Cybernetics*, 75:293–307, 1996.

[214] H. Neven, A. Steinhage, M. Giese, and C. Bruckhoff. Dynamics for vision-based autonomous mobile robots. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals To Animats 4, Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 113–123, Cambridge, Massachusetts, 1996. The MIT Press.

[215] W. S. Newman and N. Hogan. High speed robot control and obstacle avoidance using dynamic potential functions. *IEEE Int. Conference on Robotics and Automation, Piscataway, NJ*, pages 14–24, 1987.

[216] A. Nigrin. *Neural Networks for Pattern Recognition*. The MIT Press, Cambridge, Massachusetts, 1993.

[217] D.-E. Nilsson and S. Pelger. A pessimistic estimate of the time required for an eye to evolve. *Proc. R. Soc. Lond. B*, 256:53–58, 1994.

[218] S. Nolfi. Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22:187–198, 1997.

[219] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: different approaches in evolutionary robotics. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 190–197. The MIT Press, 1994.

[220] P. Nordin and W. Banzhaf. Genetic programming controlling a miniature robot. In *Working Notes of the AAAI-95 Fall Symposium Series, Symposium on Genetic Programming, MIT, Cambridge, MA, 10-12 November 1995*, pages 61–67, 1995.

[221] P. Nordin and W. Banzhaf. A genetic programming system learning obstacle avoiding behavior and controlling a miniature robot in real time. Technical Report SysReport 4/95, Department of Computer Science, University of Dortmund, 44221 Dortmund, Germany, 1995.

[222] P. Nordin and W. Banzhaf. Real time evolution of behavior and a world model for a miniature robot using genetic programming. Technical Report SysReport 5/95, Department of Computer Science, University of Dortmund, 44221 Dortmund, Germany, 1995.

[223] P. Nordin and W. Banzhaf. An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behaviour*, 5(2):107–140, 1997.

[224] P. Nordin and W. Banzhaf. Real time control of a Khepera robot using genetic programming. *Cybernetics and Control*, 26(3):533–561, 1997.

[225] P. Nordin, W. Banzhaf, and M. Brameier. Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, 25:105–116, 1998.

[226] A. Nye, editor. *Xlib Reference Manual for Version 11 R4/R5.* O'Reilly & Associates, Inc., June 1992.

[227] A. Nye, editor. *Xlib Programming Manual for Version 11 R4/R5.* O'Reilly & Associates, Inc., July 1993.

[228] A. Nye, editor. *Programmer's Supplement for Release 6 of the X Window System.* O'Reilly & Associates, Inc., Sebastopol, California, September 1995.

[229] A. Nye and T. O'Reilly, editors. *X Toolkit Intrinsics Programming Manual for Version 11 R4/R5.* O'Reilly & Associates, Inc., Sebastopol, California, August 1994.

[230] M. Oberdorfer. *NEUROPA Ein paralleles System zum Einparken eines autonomen mobilen Roboters, Diplomarbeit Nr. 1274.* Universität Stuttgart, Fakultät Informatik, IPVR, 1995.

[231] M. Olmer, P. Nordin, and W. Banzhaf. Evolving real-time behavioral modules for a robot with GP. In M. Jamshidi, F. Pin, and P. Danchez, editors, *Robotics and Manufacturing, Proc. 6th Int. Symposium on Robotics And Manufacturing (ISRAM-96), Montpellier, France*, pages 675–680, New York, 1996. Asme Press.

[232] P.-W. Ong, R. S. Wallace, and E. L. Schwartz. Space-variant optical character recognition. In *11th Int. Conference on Pattern Recognition*, pages 504–507. IEEE, August 1992.

[233] R. A. Peters II and M. Bishay. Centering peripheral features in an indoor environment using a binocular log-polar 4 DOF camera head. *Robotics and Autonomous Systems*, 18:271–281, 1996.

[234] S. Pinker. *How the Mind Works.* W. W. Norton & Company, New York, 1997.

[235] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In J.-O. Eklundh, editor, *Lecture Notes in Computer Science: Third European Conference on Computer Vision*, volume 801, pages 97–108, May 1994.

[236] R. Poli. Genetic programming for feature detection and image segmentation. In T. C. Fogarty, editor, *Evolutionary Computing AISB Workshop, Brighton, U.K., April 1-2, 1996*, pages 110–125, Berlin, 1996. Springer-Verlag.

[237] R. Poli. Genetic programming for image analysis. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 363–368, Cambridge, Massachusetts, 1996. The MIT Press.

[238] R. Poli and S. Cagnoni. Genetic programming with user-driven selection: Experiments on the evolution of algorithms for image enhancement. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997, Proc. of the Second Annual Conference, July 13-16, 1997, Stanford University*, pages 269–277, San Francisco, California, 1996. Morgan Kaufmann Publishers.

[239] A. R. Pope and D. G. Lowe. Vista: A software environment for computer vision research. In *Proc. of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 768–772. IEEE, 1994.

[240] V. Quercia and T. O'Reilly. *X Window System User's Guide OSF/Motif 1.2 Edition.* O'Reilly & Associates, Inc., August 1993.

[241] B. Radig. *Verarbeiten und Verstehen von Bildern.* R. Oldenburg Verlag, München, Wien, 1993.

[242] T. S. Ray. An evolutionary approach to synthetic biology: Zen and the art of creating life. In C. G. Langton, editor, *Artificial Life: An Overview*, pages 179–209, Cambridge, Massachusetts, 1995. The MIT Press.

[243] Real World Interface, Inc. *B21 Robot System Manual (Preliminary Version 0.5).* Real World Interface, Inc., PO Box 375, 32 Fitzgerald Drive, Jaffrey, New Hampshire 03452 USA, 1996.

[244] Real World Interface, Inc. *B21 Users Guide.* Real World Interface, Inc., PO Box 375, 32 Fitzgerald Drive, Jaffrey, New Hampshire 03452 USA, 1996.

[245] Real World Interface, Inc. *System software and RAI-1.2.2 documentation.* Real World Interface, Inc., 32 Fitzgerald Drive, Jaffrey, New Hampshire 03452 USA, 1996.

[246] Real World Interface, Inc. *BeeSoft User's Guide and Software Reference.* Real World Interface, Inc., 32 Fitzgerald Drive, Jaffrey, New Hampshire 03452 USA, 1997.

[247] I. Rechenberg. *Evolutionsstrategie '94.* frommann-holzboog, Stuttgart, 1994.

[248] D. Reimann and H. Haken. Stereo vision by self-organization. *Biological Cybernetics*, 71:17–26, 1994.

[249] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Detection of interest points using symmetry. In *Proc. of the Int. Conference on Computer Vision, Osaka, Japan*, pages 62–65. IEEE, December 1990.

[250] C. W. Reynolds. The difficulty of roving eyes. In *Proc. of the First IEEE Conference on Evolutionary Computation*, pages 262–267. IEEE, 1994.

[251] C. W. Reynolds. Evolution of corridor following behavior in a noisy world. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proc. of the Third Int. Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 402–410. The MIT Press, 1994.

[252] C. W. Reynolds. Evolution of obstacle avoidance bahavior: Using noise to promote robust solutions. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 221–241, Cambridge, Massachusetts, 1994. The MIT Press.

[253] C. W. Reynolds. An evolved, vision-based model of obstacle avoidance behavior. In C. G. Langton, editor, *Proc. of the Workshop of Artificial Life III SFI Studies in the Sciences of Complexity*, pages 327–346. Addison-Wesley, 1994.

[254] M. M. Rizki, L. A. Tamburino, and M. A. Zmuda. Evolving multi-resolution feature-detectors. In D. B. Fogel and W. Atmar, editors, *Proc. of the Second American Conference on Evolutionary Programming*, pages 108–118. Evolutionary Programming Society, 1993.

[255] S. J. Ross, J. M. Daida, C. M. Doan, T. F. Bersano-Begey, and J. J. McClain. Variations in evolution of subsumption architectures using genetic programming: The wall following robot revisited. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proc. of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 191–199, Cambridge, Massachusetts, 1996. The MIT Press.

[256] G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):901–905, September 1994.

[257] J. Rubner and K. Schulten. Development of feature detectors by self-organization. *Biological Cybernetics*, 62:193–199, 1990.

[258] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations.* The MIT Press, Cambridge, Massachusetts, 1986.

[259] R. Salomon. Increasing adaptivity through evolution strategies. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From animals to animats 4: Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 411–420. The MIT Press, 1996.

[260] G. Sandini, F. Gandolfo, E. Grosso, and M. Tistarelli. Vision during action. In Y. Aloimonos, editor, *Active Perception*, pages 151–190, Hillsdale, New Jersey, 1993. Lawrence Erlbaum Associates, Publishers.

[261] J. Santos-Victor and G. Sandini. Embedded visual behaviors for navigation. *Robotics and Autonomous Systems*, 19:299–313, 1997.

[262] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo for robot navigation: Learning from bees. In *Proc. of Computer Vision and Pattern Recognition, New York*, pages 434–439, 1993.

[263] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo in autonomous navigation: From bees to robots. *Int. Journal of Computer Vision*, 14:159–177, 1995.

[264] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. of an Int. Conference on Genetic Algorithms and Their Applications*, pages 93–100, 1985.

[265] J. D. Schaffer and J. J. Grefenstette. Multi-objective learning via genetic algorithms. In *Proc. of the Ninth Int. Joint Conference on Artificial Intelligence, 18-23 August, Los Angeles, California*, pages 593–595, Los Altos, California, 1985. Morgan Kaufmann Publishers.

[266] A. B. Scheibel and J. W. Schopf, editors. *The Origin and Evolution of Intelligence.* Jones and Bartlett Publishers International, Sudbury, Massachusetts, 1997.

[267] G. Schmidt und F. Freyberger (Hrsg.). *Autonome Mobile Systeme 1996.* Springer-Verlag, Berlin, 1996.

[268] E. Schöneburg, F. Heinzmann, and S. Feddersen. *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley (Deutschland) GmbH, Bonn, 1994.

[269] H. Schulz-Mirbach. Invariant features for gray scale images. In G. Sagerer, S. Posch und F. Kummert (Hrsg.), *Mustererkennung 1995*, pages 1–14, Berlin, 1995. Springer-Verlag.

[270] P. Schuster. Extended molecular evolutionary biology: Artificial life bridging the gap between chemistry and biology. In C. G. Langton, editor, *Artificial Life: An Overview*, pages 39–60, Cambridge, Massachusetts, 1995. The MIT Press.

[271] E. L. Schwartz. Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25:181–194, 1977.

[272] E. L. Schwartz. Computational anatomy and functional architecture of striate cortex: A spatial mapping approach to perceptual coding. *Vision Research*, 20:645–669, 1980.

[273] T. Scutt. The five neuron trick: Using classical conditioning to learn how to seek light. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proc. of the Third Int. Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 364–370. The MIT Press, 1994.

[274] M. A. Shah and R. Jain. Detecting time-varying corners. *Computer Vision, Graphics, and Image Processing*, 28:345–355, 1984.

[275] J. Shi and C. Tomasi. Good features to track. Technical Report TR 93-1399, Department of Computer Science, Cornell University, Ithaca, NY 14853-7501, November 1993.

[276] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[277] A. Silberschatz and P. B. Galvin. *Operating System Concepts.* Addison-Wesley Publishing Company, Reading, Massachusetts, fourth edition, 1994.

[278] K. Sims. Evolving 3D morphology and behavior by competition. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 28–39. The MIT Press, 1994.

[279] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach.* Springer-Verlag, Berlin, 1997.

[280] S. Smith. A new class of corner finder. In *Proc. of the 3rd British Machine Vision Conference 1992*, pages 139–148, 1992.

[281] P. J. Sobey. Active navigation with a monocular robot. *Biological Cybernetics*, 71:433–440, 1994.

[282] L. Spreeuwers. *Image Filtering with Neural Networks, Applications and Performance Evaluation.* PhD thesis, University of Twente, Netherlands, 1992.

[283] M. V. Srinivasan. Distance perception in insects. *Current Directions in Psychological Science*, 1(1):22–26, February 1992.

[284] M. V. Srinivasan. How bees exploit optic flow: behavioural experiments and neural models. *Phil. Trans. R. Soc. Lond. B*, 337:253–259, 1992.

[285] V. Srinivasan, P. Bhatia, and S. H. Ong. Edge detection using a neural network. *Pattern Recognition*, 27(12):1653–1662, 1994.

[286] S. Sutanthavibul et al. *xfig - Facility for Interactive Generation of figures under X11.* University of Texas at Austin, 1997.

[287] W. A. Tackett. Genetic generation of "dendritic" trees for image classification. In *Proc. of the World Conference on Neural Networks, Portland, Oregon, July, 1993*, pages IV–646–IV–649. IEEE, 1993.

[288] W. A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proc. of the Fifth Int. Conference on Genetic Algorithms*, pages 303–309. Morgan Kaufmann, 1993.

[289] R. Talluri and J. K. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Trans. on Robotics and Automation*, 8(5):573–584, October 1992.

[290] R. Talluri and J. K. Aggarwal. Image/map correspondence for mobile robot self-location using computer graphics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6):597–601, June 1993.

[291] R. Talluri and J. K. Aggarwal. Position estimation techniques for an autonomous mobile robot – a review. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 4.4, pages 769–801. World Scientific Publishing Company, 1993.

[292] R. Talluri and J. K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Trans. on Robotics and Automation*, 12(1):63–77, February 1996.

[293] A. S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall International (UK) Limited, London, 1992.

[294] C. Taylor and D. Jefferson. Artificial life as a tool for biological inquiry. In C. G. Langton, editor, *Artificial Life: An Overview*, pages 1–13, Cambridge, Massachusetts, 1995. The MIT Press.

[295] S. Thirumalai and N. Ahuja. Parallel distributed detection of feature trajectories in multiple discontinuous motion image sequences. *IEEE Trans. on Neural Networks*, 7(3):594–603, 1996.

[296] S. Thrun. *Explanation-Based Neural Network Learning, A Lifelong Learning Approach.* Kluwer Academic Publishers, Boston, 1996.

[297] S. Thrun, A. Brücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 21–52, Menlo Park, California, 1998. AAAI Press/The MIT Press.

[298] M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):401–410, April 1993.

[299] C. Toepfer, M. Wende, G. Baratoff und H. Neumann. Ortsvariante Karte als effektives Datenformat zur Integration visueller Navigationsaufgaben. In P. Levi, R.-J. Ahlers, F. May und M. Schanz (Hrsg.), *Mustererkennung 1998*, September 1998.

[300] C. Toepfer, M. Wende, G. Baratoff, and H. Neumann. Robot navigation by combining central and peripheral optical flow detection on a space-variant map. In *Proc. 14th Int. Conf. on Pattern Recognition, Brisbane, Australia*, 17-20 August 1998.

[301] C. Tomasi and T. Kanade. Shape and motion without depth. In *Proc. of the Int. Conference on Computer Vision*, pages 91–95. IEEE, 1990.

[302] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *Proc. of the IEEE Workshop on Visual Motion, October 7-9, 1991, Nassau Inn, Priceton, New Jersey*, pages 21–28. IEEE Computer Society Press, 1991.

[303] M. J. Tovée. *An introduction to the visual system.* Cambridge University Press, Cambridge, 1996.

[304] S. Tsuji and S. Li. Memorizing and representing route scenes. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proc. of the Second Int. Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 225–232. The MIT Press, 1993.

[305] S. Ullman. *The Interpretation of Visual Motion.* The MIT Press, Cambridge, Massachusetts, 1979.

[306] S. Ullman. Visual routines. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 298–328, Los Altos, California, 1987. Morgan Kaufmann Publishers.

[307] S. Ullman. *High-level Vision, Object Recognition and Visual Cognition.* The MIT Press, Cambridge, Massachusetts, 1996.

[308] J. H. van Deemter and H. A. K. Mastebroek. A statistical correlation technique and a neural network for the motion correspondence problem. *Biological Cybernetics*, 70:329–344, 1994.

[309] J. Vogelgesang, A. Cozzi, and F. Wörgötter. A parallel algorithm for depth perception from radial optical flow fields. In C. von der Malsburg, J. C. Vorbrüggen, W. von Seelen, and B. Sendhoff, editors, *Artificial Neural Networks: 6th Int. Conference; Proc. / ICANN 96*, pages 721–725. Springer-Verlag, 1996.

[310] E. von Holst und H. Mittelstaedt. Das Reafferenzprinzip. (Wechselwirkung zwischen Zentralnervensystem und Peripherie). *Die Naturwissenschaften*, 37(20):464–476, 1950.

[311] G. von Wichert. VISMOB: Aufbau und Nutzung selbstorganisierender, bildbasierter Umweltrepräsentationen für mobile Roboter. In P. Levi, T. Bräunl und N. Oswald (Hrsg.), *Autonome Mobile Systeme 1997*, pages 84–94, Berlin, 1997. Springer-Verlag.

[312] G. von Wichert. Mobile robot localization using a self-organized visual environment representation. *Robotics and Autonomous Systems*, 25:185–194, 1998.

[313] G. von Wichert and H. Tolle. Towards constructing and using selforganizing visual environment representations for mobile robots. In M. Á. Salichs and A. Halme, editors, *3rd IFAC Symposium on Intelligent Autonomous Vehicles, March 25-27, 1998, Madrid, Spain*, pages 712–717, 1998.

[314] R. S. Wallace, P.-W. Ong, B. B. Bederson, and E. L. Schwartz. Space variant image processing. *Int. Journal of Computer Vision*, 13(1):71–90, 1994.

[315] H. Wang, M. Brady, and I. Page. A fast algorithm for computing optic flow and its implementation on a transputer array. In Zisserman, editor, *Proc. of the British Machine Vision Conference*, pages 175–180, Oxford, 1990. British Machine Vision Association.

[316] P. Weckesser and R. Dillmann. Modeling unknown environments with a mobile robot. *Robotics and Autonomous Systems*, 23:293–300, 1998.

[317] A. H. Wertheim. Motion perception during self-motion: The direct versus inferential controversy revisited. *Behavioral and Brain Sciences*, 17:293–355, 1994.

[318] C. Wilke, S. Altmeyer, and T. Martinetz, editors. *Third German Workshop on Artificial Life. Abstracting and Synthesizing the Principles of Living Systems.* Verlag Harri Deutsch, Frankfurt am Main, 1998.

[319] T. D. Williams. Depth from camera motion in a real world scene. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:511–516, 1980.

[320] M. S. Wilson, C. M. King, and J. E. Hunt. Evolving hierarchical robot behaviours. *Robotics and Autonomous Systems*, 22:215–230, 1997.

[321] J. F. Winkeler and B. S. Manjunath. Genetic programming for object detection. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997, Proc. of the Second Annual Conference, July 13-16, 1997, Stanford University*, pages 330–335, San Francisco, California, 1997. Morgan Kaufmann Publishers.

[322] M. Xie. Matching free stereovision for detecting obstacles on a ground plane. *Machine Vision and Applications*, 9:9–13, 1996.

[323] M. Xie. Automatic feature matching in uncalibrated stereo vision through the use of color. *Robotics and Autonomous Systems*, 21:355–364, 1997.

[324] S. Yamada. Learning behaviors for environment modeling by genetic algorithm. In P. Husbands and J.-A. Meyer, editors, *Proc. of the First European Workshop on Evolutionary Robotics, Paris, April 1998*, pages 179–191, 1998.

[325] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):496–505, June 1996.

[326] H. Zabrodsky, S. Peleg, and D. Avnir. Symmertry as a continuous feature. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(12):1154–1165, 1995.

[327] S. Zeki. *A Vision of the Brain.* Blackwell Science, Oxford, 1993.

[328] A. Zell. *Simulation Neuronaler Netze.* Addison-Wesley (Deutschland) GmbH, Bonn, 1994.

[329] A. Zell, S. Feyrer, M. Ebner, A. Mojaev, O. Schimmel und K. Langenbacher. Eingeladener Vortrag: Systemarchitektur der autonomen mobilen Roboter Robin und Colin der Universität Tübingen. In H.-M. Groß (Hrsg.), *Workshop SOAVE '97 - Selbstorganisation von Adaptivem Verhalten, VDI Reihe 8 Nr. 663*, pages 151–155, Düsseldorf, 1997. VDI Verlag.

[330] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *Int. Journal of Computer Vision*, 15:31–76, 1995.

[331] T. Ziemke. Towards adaptive behaviour system integration using connectionist infinite state automata. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From animals to animats 4: Proc. of the Fourth Int. Conference on Simulation of Adaptive Behavior*, pages 145–154. The MIT Press, 1996.

[332] T. Ziemke. Towards autonomous robot control via self-adapting recurrent networks. In C. von der Malsburg, J. C. Vorbrüggen, W. von Seelen, and B. Sendhoff, editors, *Artificial Neural Networks: 6th Int. Conference; Proc. / ICANN 96*, pages 611–616. Springer-Verlag, 1996.

[333] D. Zongker and B. Punch. *lil-gp 1.01 User's Manual (support and enhancements Bill Rand).* Michigan State University, March 1996.

Lebens- und Bildungsgang

Marc Ebner
Gertrud-Bäumer-Straße 2
72074 Tübingen

m.ebner.1@alumni.nyu.edu

Universität Tübingen, Wilhelm-Schickard-Institut für Informatik Doktorand	**Jun. 1996 - Mai 1999**
Universität Stuttgart Diplom in Informatik	**Okt. 1994 - Mai 1996**
New York University, Courant Institute of Mathematical Sciences, NY, U.S.A. Master of Science in Computer Science (Spezialisierung in Robotik)	**Sep. 1993 - Sep. 1994**
Universität Stuttgart Studium der Informatik mit Nebenfach Physik	**Okt. 1990 - Sep. 1993**
Wehrdienst	**Jun. 1989 - Aug. 1990**
Gymnasium Gerlingen Abitur	**Jan. 1986 - Apr. 1989**
Suffern Senior High School, NY, U.S.A.	**Aug. 1985 - Jan. 1986**
Gymnasium Gerlingen	**Aug. 1980 - Jul. 1985**
Breitwiesenschule Gerlingen	**Aug. 1976 - Jul. 1980**

Geboren am 17. November 1969 in Stuttgart.