Autonome Optimierung des Verhaltens von Fahrzeugsteuerungen auf der Basis von Verstärkungslernen

Vom Fachbereich Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von
Dipl.-Ing. Michael Krödel

1. Gutachter: Professor Dr.-Ing. Klaus-Dieter Kuhnert
2. Gutachter: Professor Dr. rer. nat. Volker Graefe
Vorsitzender: Professor Dr. Rainer Brück

Tag der mündlichen Prüfung: 30. Mai 2006

urn:nbn:de:hbz:467-2371

Danksagung

Mein ganz besonderer Dank geht an Herrn Professor Dr.-Ing. Klaus-Dieter Kuhnert für die Betreuung der vorliegenden Arbeit.

Durch seine Ideen und Anregungen sowie stetige Gesprächsbereitschaft war er stets ein wertvoller Ansprechpartner, der mich in allen Phasen des Dissertationsvorhabens in jeglicher Form unterstützt hat.

Zusätzlich möchte ich Herrn Jordan und Frau Thiemt erwähnen und ihnen für die freundschaftliche Unterstützung und Zusammenarbeit danken.

Ottobrunn, im Juli 2006

INHALT

Abstract

Steering an autonomous vehicle requires the permanent adaptation of behaviour in relationship to the various situations the vehicle is in. In the context of autonomous steering, behaviour is understood as the situation-specific issuance of actions like steering or acceleration. The study presented in the following deals with the concept and the implementation of a system which, based on experience over a period of time, autonomously learns to steer different vehicles and optimises its behaviour to various possible road courses. This shall be done a different way than researched in many other works before as described further below.

Key element is the fact that any behaviour is dependent on the situation to which a vehicle is exposed. If a vehicle is exposed to a real environment, situations are subject to permanent changes and therefore any true autonomous system will have to continuously adapt its behaviour.

Till now the visual control of systems for autonomous vehicle driving with learning components have been implemented in several ways. [Pommerlau 91] describes a short direct connection between the image processing and a soft computing learning method using a neural network. Prior to any usage, such neural network is being trained with sample images and expected actions. This approach provides good results but only as long as future images of the scene are similar to the trained images. This approach was being enhanced by a multiple neural network [Jochem et al 93], but could not completely solve the dependency problem of the taught training images. Further developments then included a GPS system [Jochem et al 95] to support orientation or enhanced the approach with object-oriented vision in order to distinguish between road following and obstacle detection [Baluja & Pommerlau 97], [Franke et al 98]. In all those variations, however, neural networks with their inherent dependency on training patterns are embedded. Also, as a major difference to the presented research, the established knowledge on vehicle driving is stored within the neural network but not explicitly available, e.g. for optimisation or for further learning processes.

A completely different approach is being followed by using explicit modelling, therefore trying to rebuild a model of the environment as well as the vehicle and to derive proper actions from it. The basic idea of such a model is to try to understand interaction between vehicle and environment and to predict consequences of any behaviour thus allowing vice-versa to determine a suitable behaviour in a given situation.

The major challenge of such approach is to find a suitable model which approximates the true vehicle behaviour and environment in the best way. Any difference between the model and either the real environment or the vehicle results in a difference between the calculated behaviour and an optimum behaviour. Any model also needs to be shaped in architecture and tuned with parameters. Usually there are no versatile models, so any change of e.g. ve-

hicle or environment requires a corresponding modification of parameters or architecture, if not both. In other words, any tuned model is valid only for a certain environment or vehicle and is more or less sensible to any change of these. [Dickmanns & Zapp 87] describes an early success with international attention of a vehicle system using a real-time vision system BVV2 [Kuhnert 86]. Further developments in this area (e.g. [Dickmanns 87], [Dickmanns et al 94]) are being pursued with significant progress, however always dependent on many parameters for the modelling process.

In consequence, some researchers focussed on enhancing a model approach with learning capabilities. [Ramachandran et al 05] utilizes reinforcement learning algorithms to select between different steering command candidates – each provided by a different sub-model. [Kuhnert & Dong 03] incorporate a neural network in order to learn the model parameters rather than doing the calibration by hand.

Other research work tried to avoid both modelling and neural networks and built systems that accumulate driving capabilities over time. [Bischoff & Graefe 04] describes a system which demonstrates driving capabilities of an indoor robot based on sensor information without interpreting their detailed meaning (e.g. object detection). Information of sensor data, issued actions and their near-future impact on the sensor data are being stored and provide a basis of action selection.

Also, some work has been focused on the usage of Reinforcement Learning as the basis for learning capabilities. Reinforcement Learning systems provide capabilities of self-optimising behaviour based on feedback they get from an environment. The behaviour of such systems is mainly based on a choice of situation-specific actions as well as a rating of their long-term success regarding the overall goal to be achieved. The situations, actions and their ratings span a state-space to be autonomously explored while coping with delayed feedback as well as coping with disturbed feedback.

Given the above aspects, it should also be noted that our research does not strive to compete with the established approaches like modelling or neural networks. Instead, any progress of Reinforcement Learning systems might be used to enhance the advantages of modelling or neural networks achieved so far or vice-versa. At the end, a combined system built upon e.g. modelling and Reinforcement Learning might provide better results than each approach alone. In this light, we strongly believe that Reinforcement Learning system will play a significant role in the near future in autonomous driving systems.

This research focuses purely on Reinforcement Learning in order to explore its benefits and limitations. All in all, the main targets of this research are:

- steering an autonomous vehicle along different types of road courses
- autonomous exploration of new actions for familiar as well as for new situations, therefore autonomous optimization (self-tuning of the system to any combination of environment and vehicle)
- learning from evaluative feedback (in contrast to instructive feedback/ teaching)
- coping with missing, disturbed or delayed feedback
- real-time processing.

According to Sutton/Barto [Sutton & Barto 98], a Reinforcement Learning system consists of an agent and an environment. The agent is the core of the learning system and receives at time t an input regarding the state (a discrete form of the situation) s_t as well as a reward r_t and determines an appropriate action a_t. This action will cause a reaction of the environment and consequently results in a change of state from s_t to s_{t+1}. Similarly, the environment will also issue a reward r_{t+1} corresponding to s_{t+1}. at time $t+1$

Since the determination of the state s and the reward r is to be provided by the environment but usually not being provided by any environment simulator, our system enhances the environment with methods of Image Processing, Pattern Matching and Reward-Generation.

In consequence, the current research implements the following major modules:

- Environment Simulator
- Image Processing
- Pattern Matching
- Reward Generation
- Reinforcement Learning

whose major aspects are being described as follows.

For the Environment Simulator, different driving simulator software is being used which always simulate the steering of an autonomous vehicle along a piece of road. Partially, an own software is being used which allows to gradually increase the complexity of the environment - partially, a commercially available driving simulator has been chosen. In any case, the Environment Simulator receives the actions (e.g. steering command) and calculates the resulting position of the vehicle on the road. The output of the Environment Simulator is an image representing the view a human driver would have if looking through the front window of such vehicle.

The Image Processing module retrieves a parametric scene description from any incoming image. Basically, those parametric scene descriptions represent the road marks. In order to extract those, the Image Processing Module uses a self-build statistical database to determine as to how likely a colour-contrast within a single image is supposed to be part of a road mark. The Image Proc-

essing module also recognizes the fact that the difference between camera coordinates and world coordinates distorts the representation of the road marks within the image and partially compensates such distortion.

The Pattern Matching module retrieves similar situations in comparison to the actual situation the vehicle is in. Nucleus of this module is an approximate nearest neighbour algorithm which continually determines the similarity between a current situation and previously experienced ones. In order to be able to do so, any parametric scene description of the previous module is being converted into a metric representation and further being referred to as the abstract complete situation description (ACSD). Those ACSD's are being continually stored in a pattern database for as long as the database does not yet contain a similar ACSD. Both, the ACSD of the current situation and the ACSD pattern database are the basis for the approximate nearest neighbour algorithm. Such way, the module Pattern Matching creates a pattern database whose size (i.e. number of ACSD-entries) automatically adapts to the complexity of the environment. Finally, the result of this module is a reference to a stored ACSD-entry as a representation of the current state – a parameter crucially needed by any Reinforcement Learning system.

The module Reward Generation provides a reward indicating how much the vehicle deviates horizontally from the road centre. Such reward is in line with the overall goal of road following. However, the module Reward Generation is also able to issue delayed or errand rewards in order to test the robustness of the learning system.

The module Reinforcement Learning determines a suitable behaviour in terms of actions for the current state or, since the state has been determined on the basis of the situation, the situation. This determination is based on the actual state in conjunction with previously recorded, similar states and their issued actions. For such task it is crucial to rate the appropriateness of any previously issued action, otherwise optimization of behaviour would not be possible. Therefore, this subsystem is also responsible for weighing the success, respectively the appropriateness of any calculated action and also copes with the difficulty that such appropriateness can often only be determined after a time delay.

Such achievements are key characteristics of a Reinforcement Learning system and therefore such system is being examined closely. In detail, the impact of the traditional parameters of Reinforcement Learning systems is being explored in conjunction with the concrete task of this research. This is being done while gradually increasing the complexity of the environment. In conjunction with the evaluation of the corresponding test-series, a measure of convergence is being proposed. Such measure of convergence is useful for Reinforcement Learning systems whose values function, which lead to behaviour, are supposed to converge to a smooth function within the behaviour space. Even though a Reinforcement System strives towards full convergence of such values function, it will not fully achieve such final state in any complex

environment and a stable measure of convergence is an important indication as to how trustful the determination of an action is.

All in all, this research provides extensive insights in coping with a Reinforcement Learning system. Autonomous exploration of the behaviour spaces is being achieved – even when rewards of the environment have been partly missing or errand. Many test-series have been performed to support the statements in this research.

In addition to the learning capabilities, this research also demonstrates the capabilities of a driving system purely based on Pattern Matching. A corresponding fast and robust algorithm is being developed to extract information regarding the road type into pattern and, based on a fast and reliable pattern matching algorithm, to quickly locate similar pattern in order to issue associated steering commands. Such combination proves remarkable steering capabilities based on the combination of Image Processing and Pattern Matching.

EINLEITUNG

1.1 Motivation

Aufgrund des technischen Fortschritts einerseits und der Forderung nach mehr aktiver und passiver Sicherheit anderseits hat die elektronische Ausstattung von Fahrzeugen im Laufe der vergangenen Jahrzehnte deutlich zugenommen. In heutigen Fahrzeugen wird eine Vielzahl von Funktionen durch die Elektronik übernommen oder unterstützt. Eine zentrale Aufgabe, nämlich die Fahrzeugsteuerung selber, ist derzeit noch dem menschlichen Fahrer vorbehalten.

Entwicklungen in Bezug auf eine autonome Roboter- oder Fahrzeugsteuerung existieren schon seit längerer Zeit und Kapitel 2 gibt einen Überblick über den aktuellen Status themenrelevanter Forschungsarbeiten. Dabei ist das Kernproblem bei allen Arbeiten, dass Situationen, in denen sich ein Fahrzeug befinden kann, beliebig komplex sind. *Sensordaten* von z.B. Kamera, Laserscannern oder Abstandssensoren enthalten Rahmendaten der Situationen und werden in Situationsbeschreibungen überführt. Dabei ist eine Situationsbeschreibung eine auf Parameter reduzierte Beschreibung der Umgebung, bzw. der Situation. Je komplexer die Umgebung oder eine mögliche Situation, desto komplexer auch die Situationsbeschreibung.

Diese Arbeit widmet sich der Implementierung eines lernenden Systems, im Folgenden auch als *Lernsystem* bezeichnet, welches im Kern zwei Problemfelder adressiert:

a) es wird **eigenständig** eine Anzahl von möglichen Situationsbeschreibungen anlegt
b) für diese Situationsbeschreibungen wird die Fahrzeugsteuerung, d.h. die Wahl der auszugebenden Steuerkommandos, autonom **optimiert**

Das eigenständige Anlegen von Situationsbeschreibungen ist elementar wichtig, um die hohe Komplexität der Umgebung nicht auf das lernende System durchschlagen zu lassen. Statt eine beliebig hohe Anzahl aller theoretisch möglichen Situationsbeschreibungen zu erfassen, wird das vorliegende System lediglich die tatsächlich vorkommenden, daher erlebten Situationsbeschreibungen erfassen. Die Anzahl der dadurch angelegten Situationsbeschreibungen passt sich an die Komplexität der Umgebung automatisch an und ermöglicht den selektiven, d.h. effizienten, Einsatz von Rechenkapazitäten.

Für die angelegten Situationsbeschreibungen - und nur für diese - wird das Verhalten optimiert. Das Optimieren von Verhalten bedeutet in diesem Zusammenhang, dass Steuerkommandos, im Speziellen die Lenkbefehle, da-

hingehend bewertet werden, wie angemessen diese in einer entsprechenden Situation sind, um ein übergeordnetes Ziel, z.B. das Fahren entlang einer Fahrbahn, zu erreichen. Entscheidend in dieser Arbeit ist dabei, dass das System keinerlei Vorwissen voraussetzt, z.B. aufgrund von Lernphasen. Vielmehr wird das mögliche Verhalten, das sich unter anderem aufgrund der unterschiedlichen Situationsbeschreibungen und deren möglichen Steuerkommandos ergibt, autonom erforscht. Dies wird als Erkundung (exploration) bezeichnet.

Zur Erkundung des möglichen Verhaltens gehört auch die bewusste Ausgabe von sub-optimalen Steuerkommandos. Das bedeutet, dass die Intensität der Erkundung auch davon abhängt, in wie weit auch unangemessene Steuerkommandos ausgegeben werden können, ohne z.B. einen Unfall oder eine Beschädigung der Testapparatur zu riskieren. Die Intensität der Erkundung entscheidet somit, wie intensiv das mögliche Verhalten erforscht wird.

Durch die Fokussierung auf Situationsbeschreibungen und Verhaltensoptimierung, wird die Erstellung eines mathematischen Modells des Fahrzeugs oder der Umgebung vermieden. Parameter zum Lenkverhalten des Fahrzeugs oder eine Modellierung des Fahrbahnverlaufs werden nicht explizit benötigt.

Dabei stellt sich das System ausdrücklich dem Problemfeld gestörter Situationsbeschreibungen, bzw. –bewertungen, wie sie z.B. aufgrund von Rauschen innerhalb der Sensordaten auftreten können.

Mit unterschiedlichen Testreihen wird die Funktionsweise des oben geschilderten Ansatzes nachgewiesen und eine robuste Bewertung der Situationsbeschreibungen erreicht.

In Bezug auf die oben aufgeführte Zielstellung liegt der Fokus dieser Arbeit auf einem System, das das Fahrverhalten selbständig, d.h. aufgrund von eigenen Erfahrungen, an ein beliebiges Fahrzeug und an beliebige Fahrbahnverläufe anpasst.

1.2 Lernfähigkeit als Fokus

Durch permanente Steigerung der Leistungsfähigkeit von Computern sowie dem gleichzeitigen Preisverfall im IT-Bereich werden Computersysteme zunehmend für komplexe Aufgaben, z.B. maschinelles Lernen, eingesetzt.

Bereits 1950 wurde in [Turing 50] ein Ausblick auf denkende Maschinen gegeben. Turing befasst sich zunächst mit der Definition des Denkens und ersetzt diese Frage durch eine Aufgabe, in der eine Maschine das Verhalten eines Menschen nachahmen muss, ohne als Maschine erkannt zu werden. Im Detail befragt ein Fragesteller über einen Fernschreiber eine Maschine oder eine Frau. Beide geben sich als Frau aus. Der Fragesteller hat die Aufgabe herauszufinden, ob der jeweilige Dialogpartner die Frau oder die Maschine ist.

Um den Test möglichst gut zu bestehen, muss die Maschine in der Lage sein, das menschliche Denken zu imitieren. Turing ist zuversichtlich, dass die Konstruktion einer solchen Maschine machbar ist – im Detail schätzt er, dass im Jahre 2000 eine Maschine in mindestens in 30% der Dialoge den Fragesteller täuschen könne, wobei in solch einem Fall die Maschine jeweils als denkend anzusehen ist.

Dabei diskutiert Turing die Notwendigkeit und Fähigkeit einer Maschine, menschliche Fehler vorzutäuschen – im Gegenzug dazu kann ein Mensch sehr leicht durch z.B. komplizierte arithmetische Fragen identifiziert werden. Interessanterweise geht Turing bereits auf den Programmieraufwand einer solchen Maschine ein. Er erkennt den sehr hohen Aufwand und schlägt vor, zunächst eine Maschine mit der Intelligenz eines Kindes zu programmieren. Dann wird diese Maschine einer autonomen Lernphase ausgesetzt, die über Belohnung und Bestrafung einen weiteren Lernprozess durchläuft und in Konsequenz die Gesamtfähigkeit des Systems deutlich erhöht.

Insbesondere seit Erfindung von IC-basierten Computern hat sich die Entwicklung der Leistungsfähigkeit von Maschinen rasant entwickelt. In [Kurzweil 99] wird die mehr oder weniger fix gegebene Rechenleistung des menschlichen Gehirns mit der stets wachsenden Rechenleistung von Computern verglichen. Dabei wird gezeigt, dass sich die Rechenleistung von Maschinen bereits seit 1900 beginnend mit mechanischen Rechen-Maschinen über Röhren-Computer bis hin zu heutigen IC-basierten Computern jedes Jahr zunächst alle 3 Jahre, inzwischen jedes Jahr verdoppelt. Diese exponentiell-ähnliche Leistungssteigerung wird laut Kurzweil, hochgerechnet auf die nächsten Jahre, dazu führen, dass die Rechenleistung eines gewöhnlichen Standardcomputers im Jahre 2020 mit der Rechenleistung des menschlichen Gehirns vergleichbar ist. Im Detail zeigt Kurzweil, dass ein Rechner im Jahre 1999 zu einem Preis von ca. 1000$ ca. 10^8 Rechenschritte pro Sekunde durchführen kann. Aufgrund der Evolution im Computerbereich ist anzunehmen, dass im Jahre 2020 die gleiche Klasse von Rechnern eine Anzahl von 20 * 10^{15} Rechenschritte pro Sekunde erbringen kann. Dies entspricht gemäß Kurzweil der geschätzten Rechenleistung des menschlichen Gehirns – letzteres ergibt sich aufgrund schätzungsweise 10^{11} Neuronen mit durchschnittlich ca. 10^3 Verbindungen zu anderen Neuronen mit jeweils schätzungsweise 200 Rechenschritten pro Neuronenverbindung.

Für eine effiziente Nutzung dieser immensen Rechenleistung ist neben der reinen Verarbeitungsgeschwindigkeit der Computer deren Algorithmenstruktur von entscheidender Bedeutung. Kurzweil unterscheidet dabei in drei Grundalgorithmen: Aktionsgenerierung aufgrund rekursiver Berechung (z.B. im Falle einen Schach-Programms, bei dem alle zukünftig möglichen Brettformationen und Spielzüge berechnet und analysiert werden), Aktionsgenerierung auf der Basis von Mustererkennung (z.B. auf Basis Neuronaler Netze, die vorher mit Hilfe von Trainingsmustern konditioniert werden) und Aktionsgenerierung auf der Basis evolutionärer Algorithmen. Die herausragende Schwierigkeit, so Kurzweil, wird dabei sein, die Architektur des lernenden Systems an die zu lernende Aufgabenstellung anzupassen. Diese Aufgabe wird noch lange dem

Menschen vorbehalten bleiben, bis auch diese über intelligente Maschinen erfolgen kann.

In seiner Arbeit geht Kurzweil sogar weiter und stellt die zukünftigen Grenzen zwischen Mensch und Maschine in Frage. Computer werden immer intelligenter und können mehr und mehr menschliche Fähigkeiten wie Erkennen, Interpretieren, Gestalten, Denken (gemäß Turing), etc. übernehmen. Dabei haben Computer schon jetzt den Vorteil einer nicht-flüchtigen Datenspeicherung. Hingegen kommen in der Medizin immer öfter technische Implantate zum Einsatz – angefangen von künstlichen Hüftgelenken, Herzschrittmachern, Hörimplantaten, möglichen Seh-Implantaten bis hin zu möglichen Hirnimplantaten. Was passiert, wenn das Gehirn inklusive der Erinnerungen sowie der kognitiven und kreativen Fähigkeiten eines Menschen eingescannt und innerhalb eines Computers nachgebildet wird und sich dort autonom weiterentwickelt? Wo hört der Computer auf und wo fängt der Mensch an?

Ohne solch eine sensible Frage an dieser Stelle beantworten zu wollen und können erkennt man an diesem Themenkomplex, dass Computer durch die exponentielle Entwicklung der Rechenleistung immer kostengünstiger komplexe Aufgaben ausüben können – oft in Verbindung mit maschinellem Lernen. Abbildung 1.1 zeigt die Kostenentwicklung maschineller Rechenleistung. Die In Zukunft werden alleine aufgrund wirtschaftlicher Aspekte komplexe, d.h. lernende, Aufgabenstellungen mehr und mehr auf den Computer übertragen.

Dieser Trend lässt sich bereits in Forschung und Produktentwicklung verfolgen: die eigentliche Lernfähigkeit von Computersystemen rückt mehr und mehr in den Fokus und somit auch in den Bereich der konkreten Umsetzung.

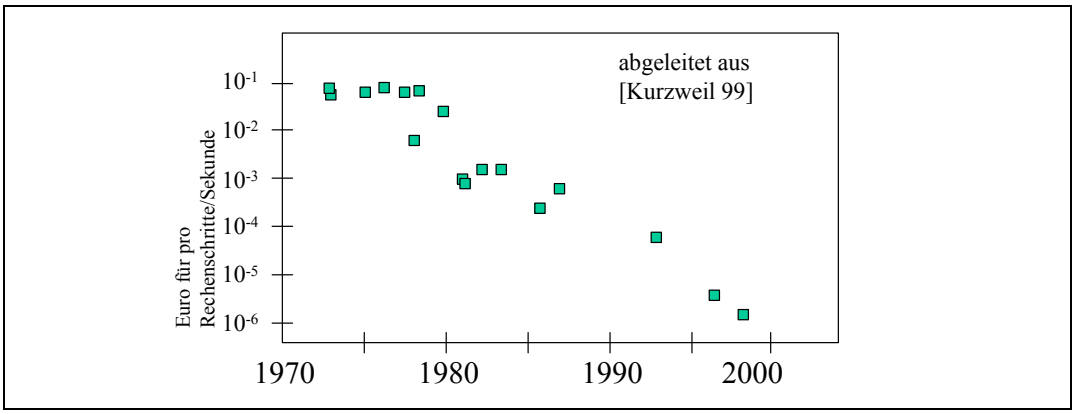


Abbildung 1.1 –Kostenentwicklung für die maschinelle Rechenleistung

Die vorliegende Arbeit befasst sich im Wesentlichen mit der Lernfähigkeit von Computersystemen. Dabei wird großer Wert darauf gelegt, dass das gelernte Wissen so abgelegt wird, dass es nachvollziehbar analysiert, d.h. auch in weiteren Prozessschritten weiter optimiert werden kann. Als zusätzliche Forderung wird erhoben, dass das System eigenständig lernt und nach Möglichkeit auf jegliche Form von Instruktion oder Trainingsphasen verzichtet.

Abbildung 1.2 zeigt diese beiden Anforderungen. Die erste Forderung, der Anspruch nach nachvollziehbar abgelegtem Wissen, wird üblicherweise durch einen modell-basierten Ansatz implementiert. Ein parametrisiertes Modell bildet die Umgebung näherungsweise ab und über konkrete Werte der Parameter kann das Gesamtverhalten des Systems entsprechend verändert werden. Je weiter sich das Modell der Umgebung annähert, desto komplexer wird die Parametrisierung und über deren Werte muss ein Großteil des benötigten Wissens bereits vorgegeben werden. Dies widerspricht dem Ansatz des komplett autonomen Lernens.

Das autonome Lernen wird üblicherweise über KI-basierte Verfahren (KI: künstliche Intelligenz) implementiert, z.B. über ein Neuronales Netz. Dabei wird lediglich die Struktur des Neuronalen Netzes vorgegeben; der eigentliche Lernvorgang wird vom Neuronalen Netz und dessen Aktualisierungsverfahren selbständig durchgeführt. Allerdings erfolgt der Lernvorgang des Neuronalen Netzes aufgrund einer Trainingsphase. Hauptmanko im Umfeld Neuronaler Netze ist, dass das gelernte Wissen nicht explizit, geschweige denn optimierbar, zur Verfügung steht. Es wird durch die gewichteten Verknüpfungen zwischen den Neuronen repräsentiert und kann lediglich angewendet, nicht aber durch zusätzliche Verfahren analysiert oder optimiert werden.

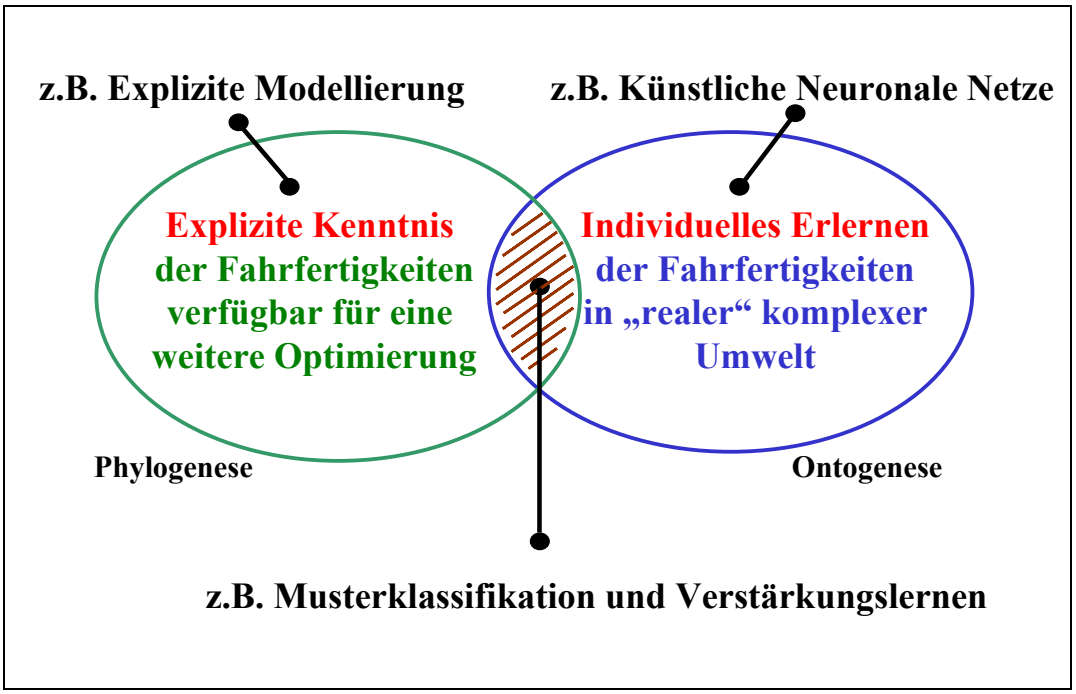


Abbildung 1.2 – Fokus der vorliegenden Arbeit: Musterklassifikation und Verstärkungslernen

Aufgrund der Anforderungen dieser Arbeit, insbesondere der Forderung nach Vermeidung einer expliziten Modellierung sowie nach eigenständiger Lernfähigkeit, kristallisierte sich die Verwendung eines Systems basierend auf *Verstärkungslernen* (Reinforcement Learning) heraus. Dabei wird durch Verstärkungslernen ein Lernsystem bezeichnet, welches rein auf Basis von Belohnungen aus der Umwelt lernt und diese zum Optimieren des Verhaltens nutz.

Durch Verstärkungslernen wird die Menge der möglichen Verhalten autonom erforscht, d.h. es wird ermittelt, in wie weit Steuerkommandos in einer Situation angemessen sind – Details werden in Kapitel 4 dargestellt.

Dabei klassifiziert das System die möglichen Situationen, in denen es sich befindet, in Situationsbeschreibungen und ordnet jeder Situationsbeschreibung eine Anzahl von Steuerkommandos, bzw. Aktionen zu. Aus diesen Kombinationsmöglichkeiten von Situationsbeschreibungen und Aktionen wird die Menge der möglichen Verhalten bestimmt und jede mögliche Kombination aus Situationsbeschreibung und Aktion wird einer Bewertung gemäß Verstärkungslernen-Verfahren unterzogen. Durch diese Struktur sind Situationsbeschreibungen, Aktionen und Bewertungen explizit verfügbar und erlauben einen externen Zugriff auf diese Daten z.B. für Analysen oder Verhaltensvoraussagen. Die Forderung nach Optimierung wird über die Verstärkungslernen-eigenen Methoden erfüllt: nach einem ausreichend oft durchgeführten Bewertungslauf konvergieren die Bewertungen und geben Aufschluss über die jeweilige Angemessenheit der Aktion in der betreffenden Situation. Dabei muss hervorgehoben werden, dass die Konvergenz der Bewertungen komplett eigenständig durchgeführt werden kann – d.h. es wird die Forderung nach autonomem Lernen erfüllt.

Trotz der hohen Erwartungshaltung an den Einsatz von Verstärkungslernen, soll mit dieser Arbeit keine Konkurrenz zu den bisher etablierten Methoden (z.B. Modellierung, Neuronale Netze, Mustererkennung) aufgebaut werden. Vielmehr sollen die Möglichkeiten und Grenzen von Verstärkungslernen ausreichend erforscht werden, um für zukünftige Systeme zu entscheiden, mit welcher Kombination von Lernmethoden sie ihre jeweilige Aufgabe am besten erfüllen. Denkbar sind hybride Systeme, wie z.B. der Einsatz von Verstärkungslernen zur Erlernung der Parameter eines Fahrzeugmodells.

1.3 Aufgabenstellung

Diese Arbeit widmet sich der Erforschung der Möglichkeiten und Grenzen von Verstärkungslernen im Umfeld von Fahrzeugsteuerungs- und Fahrzeugassistenzsystemen. Im konkreten Fall wird dazu ein lernendes Fahrsystem implementiert, das sich autonom an ein beliebiges Fahrzeug und an beliebige Streckenverläufe anpasst.

Wichtig dabei ist, dass die Stimuli für dieses System als Information, wie gut oder wie schlecht der aktuelle Fahrstil ist, einzig und allein aus den Bildinformationen gewonnen werden, die aus einer angeschlossenen Videokamera stammen. Die Videokamera digitalisiert die Straßenverkehrsszenen und liefert diese in Form von *Bildfolgen* an das in dieser Arbeit behandelte System. Das System muss sich dabei auch auf das aktuelle Fahrzeug anpassen, ohne dass mathematische Parameter über das Lenkverhalten verfügbar sind. Das bedeutet, dass kein explizites Fahrzeugmodell verwendet wird.

Das Ziel dieser Arbeit ist, die Quersteuerung eines Fahrzeugs über ein System erfolgen zu lassen, das im Wesentlichen die folgenden Anforderungen erfüllt:

- Erkennung des Straßenverlaufs, sowie Ableitung von sinnvollen Steuerkommandos bezüglich der Querlenkung.
- Aufbau und Optimierung von Verhaltensmustern für bekannte oder dazu ähnliche Situationen. Je öfter ein Fahrzeug einer gleichen Situation ausgesetzt ist, desto souveräner muss das Lenkverhalten erfolgen und desto schneller muss bei Abweichung von der Fahrbahnmitte zu dieser wieder zurückgefahren werden.
- Der Aufbau und die Optimierung des Verhaltens sollen eigenständig erfolgen. D.h. das System muss die Menge der möglichen Verhalten autonom erforschen und wird nicht in Bezug auf das Fahrzeug oder die Umgebung manuell parametrisiert.
- Das System muss gegenüber gestörten Umgebungsinformationen oder gestörten Bewertungen unempfindlich sein.

Alle Funktionen werden in Echtzeit bewältigt.

Um die Arbeit auf die Lernfähigkeit zu konzentrieren, werden dieser Arbeit noch zwei vereinfachende Bedingungen zugrunde gelegt:

- Der Straßenverlauf enthält keine Kreuzungen, Einmündungen, Verkehrsschilder, etc., die beachtet werden müssen.
- Zusätzlich zu dem eigenen Fahrzeug ist kein anderes Fahrzeug auf der gleichen Teilstrecke unterwegs – d.h. es bestehen keine Anforderungen an Erkennung von Objekten oder Hindernissen auf der Fahrbahn.

Zur Implementierung wird ein Computer benötigt, auf dem der Verstärkungslernen-Algorithmus implementiert ist und der im Weiteren als System-PC be-

zeichnet wird. Die Verkehrszenen werden über eine Videokamera aufgenommen und dem System-PC zugeführt. Hier werden die Verkehrszenen verarbeitet und die Steuerkommandos über eine parallele Schnittstelle ausgegeben.

Zur Simulation der Fahrzeugreaktion und des Streckenverlaufs wird ein zweiter Computer benötigt, der die Steuerkommandos des System-PC als Eingabe nutzt und basierend darauf die neuen Zustände für Geschwindigkeit und Position berechnet, sowie über einen Monitor ausgibt. Dieser zweite Computer wird als Simulator-PC bezeichnet.

Durch Simulationsprogramme ist inzwischen ein genügend hoher Realitätsgrad gegeben und es wird vermieden, einen aufwendigen Fahrzeugaufbau realisieren zu müssen. Außerdem steht eine Vielzahl von Streckenverläufen ohne Aufwand bezüglich Streckensperrung, Gefährdung anderer Verkehrsteilnehmer oder Gefahr der Beschädigung von Material zur Verfügung.

Außerdem ist es dadurch möglich, den Algorithmus auch mit extremen Lernimpulsen (z.B. Unfälle aufgrund Fehlverhaltens) zu testen, die in einer Simulation einfacher und kostengünstiger zu untersuchen sind als in der Realität.

Abbildung 1.3 zeigt den Versuchsaufbau, wie er für diese Arbeit verwendet wird. Die Videokamera zeichnet die auf dem Monitor dargestellten Verkehrszenen auf; der System-PC wertet diese Informationen aus und gibt die Steuerkommandos über eine Konverter-Box auf die Schnittstelle des Simulator-PC, auf dem das Simulationsprogramm läuft. Dieses ermittelt die nötigen Änderungen in Bezug auf den Fahrzeugzustand und gibt die neue Verkehrsszene über den angeschlossenen Monitor aus.

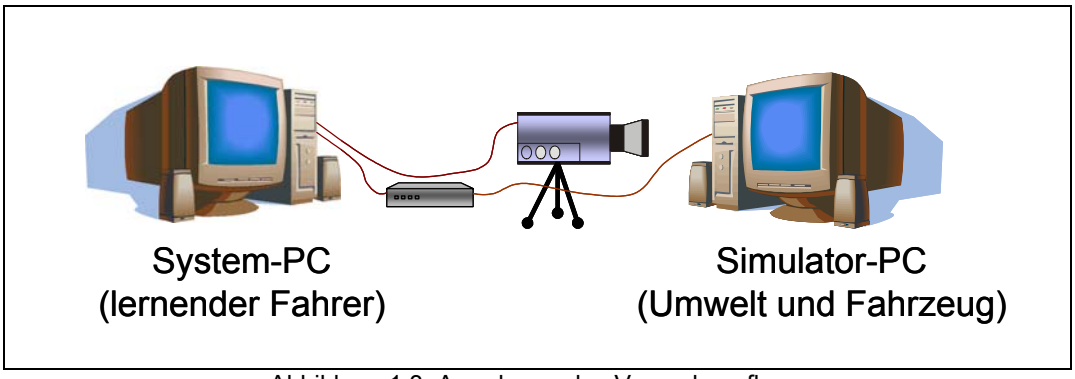


Abbildung 1.3: Anordnung des Versuchsaufbaus

1.4 Allgemeine Begriffsdefinitionen

An vielen Stellen in dieser Arbeit werden feststehende Begriffe verwendet, die vorab in ihrer Definition wie folgt festgelegt werden.

Systeme im Themenumfeld der vorliegenden Arbeit haben gemeinsam, Sensordaten aus der *Umgebung (environment)* aufzunehmen. Diese Sensordaten sind repräsentativ für die *Situation (situation)*, in der sich das autonome System befindet. Dabei kann die Situation sowohl den Zustand der Umgebung als auch den Zustand des autonomen Systems wie z.B. die Eigenposition oder den Zustand von Aktoren umfassen. Von diesen Sensordaten wird eine *Situationsbeschreibung (situation description)* abgeleitet. Eine solche Situationsbeschreibung ist eine quantisierte Untermenge der unendlichen Menge der Situationen und wird in dieser Arbeit auch als *Zustand (state)* des lernenden Systems bezeichnet. Andererseits interagieren die autonomen Systeme mit ihrer Umgebung, indem *Aktionen (actions)* ausgeführt werden. In der vorliegenden Arbeit sind diese Aktionen *Steuerkommandos (steering commands),* die im Detail ein Lenkmanöver ausführen. Nach Ausführen der Aktion werden die Sensordaten analysiert und es wird erneut eine Situationsbeschreibung abgeleitet.

Wenn nun die ausgeführten Aktionen in Beziehung zur Veränderung der Situationsbeschreibung bewertet werden und diese Informationen über die Zeit gesammelt und als *Situationsbewertung* strukturiert abgelegt werden, sprechen wir im Folgenden vom Aufbau eines *Verhaltensraumes (behaviour space)*. Somit ist der Verhaltensraum ein virtueller Raum, der aufgrund von Situationsbeschreibungen, Aktionen und Situationsbewertungen aufgespannt wird

Der Begriff *Verhalten (behaviour)* steht synonym für die situationsspezifische Ermittlung und Ausgabe von Aktionen. Dabei sagt der Begriff Verhalten noch nichts über die Angemessenheit der Aktionen aus – Ziel einer jeden autonomen Roboter- oder Fahrzeugsteuerung ist dabei das Optimieren von Verhalten, d.h. die Ermittlung und Ausgabe von möglichst situationsangemessenen Aktionen.

An dieser Stelle sei vermerkt, dass feststehende Begriffe bei ihrer ersten Nennung inklusive Definition *kursiv* dargestellt werden und somit bis zum Ende der Arbeit entsprechend verwendet werden können. Zusätzlich zur Definition im Text sind alle kursiv markierten Begriffe im Glossar (Annex A) in ihrer Definition nachzuschlagen.

1.5 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich wie folgt:

Kapitel 1 vermittelt einen Überblick über die Aufgabenstellung und die Strukturierung der Arbeit.

Kapitel 2 gibt einen Überblick über die bisherigen Systeme und bekannten Forschungsprojekte ähnlicher Thematik.

Kapitel 3 beschäftigt sich mit der Merkmalsextraktion aus visuellen Eingangsdaten

Kapitel 4 befasst sich mit Verstärkungslernen inklusive der mathematischen Hintergründe.

Kapitel 5 stellt die Konzeption der der Arbeit inklusive der Unterteilung in Teilsysteme dar.

Kapitel 6 bezieht sich auf die Implementierung und diskutiert die Ergebnisse der einzelnen Teilbereiche sowie deren Zusammenspiel im Gesamtsystem.

Kapitel 7 fasst die Erkenntnisse aus dieser Arbeit zusammen und gibt einen Ausblick auf weitere Entwicklungen.

2 BESTEHENDE SYSTEME

Der Schwerpunkt der vorliegenden Arbeit liegt auf lernenden autonomen Systemen zur Fahrzeugnavigation. In Konsequenz wird im Folgenden ein Quervergleich zu anderen Arbeiten des Bereichs autonomes Lernen und dabei in Kombination mit Robotersteuerung oder Fahrzeugnavigation vorgestellt. Diese Anwendungen haben gemeinsam, Steuerkommandos zur Navigation zu generieren und gegebenenfalls das Verhalten zu optimieren.

Dazu lassen sich die im Folgenden referenzierten Arbeiten in drei wesentliche Gruppen unterteilen:

- Systeme auf der Basis fester Modelle
- Systeme auf der Basis lernender Modelle (adaptive Modelle)
- Systeme auf der Basis lernender Systeme (d.h. ohne Modell)

2.1 Systeme auf der Basis fester Modelle

Beim modellbasierten Ansatz werden Kinematik und häufig auch die Dynamik des autonomen Systems sowie Interaktion mit der Umgebung durch ein Modell angenähert. Der Zustand des Modells in einer aktuellen Situation wird geschätzt und dient als Basis für Vorhersagen bezüglich der Auswirkungen von Aktionen. Auf dieser Basis erfolgt die Ermittlung der vermeintlich besten Aktion in der aktuellen Situation.

[Dickmanns & Zapp 87] beschreibt bereits einen frühen international anerkannten Erfolg eines Echtzeit-Fahrsystems BVV2 [Kuhnert 86].

[Dickmanns 02] beschreibt die Entwicklung von autonomen Fahrsystemen auf Basis von Bildauswertung und stellt unter anderem das eigene, modellbasierte Fahrzeug VAMP (Kurzform von VaMoRs-P - "Versuchsfahrzeug für autonome Mobilität und Rechnersehen - PKW") vor. Im Rahmen einer Dissertation an der gleichen Fakultät hat Gregor [Gregor 02] ein Konzept zur globalen Bahnführung eines autonomen Landfahrzeugs beschrieben. Dabei werden die Fähigkeiten des Systems in Bezug auf Wahrnehmung als auch Fortbewegung zunächst individuell implementiert und innerhalb einer hierarchischen Struktur zusammengefasst. Die Planung komplexer Fahrmissionen erfolgt aufgrund einer Modellierung des Operationsgebiets und der darin vorkommenden Objekte sowie unter Berücksichtigung der eigenen Fahreigenschaften.

Während der Durchführung einer Fahrmission ist eine Hierarchie von Entscheidungsinstanzen für die situationsgerechte Aktivierung der erforderlichen Fähigkeiten im Hinblick auf die aktuelle Situation und die spezifizierten langfristigen Ziele verantwortlich. Die resultierenden Fahrzeugbewegungen wer-

den mit den geplanten Aktionen permanent abgeglichen und durch die Positionsbestimmung wird der Fortschritt innerhalb der Mission überwacht.

Das entwickelte Konzept wurde im Rahmen des laboreigenen Vision Systems entwickelt und in das autonome Versuchsfahrzeug VaMoRs integriert. Die Leistungsfähigkeit der entwickelten Methoden und Algorithmen zur globalen Fahrzeugführung und der situationsgerechten Aktivierung einzelner Fähigkeiten wurde in ausgiebigen Fahrversuchen nachgewiesen.

Durch die Arbeit wurden mehrere Ziele erreicht. Zum einen die noch flexiblere Möglichkeit der Modell-Parametrisierung. Zum anderen wurde die Möglichkeit der Selbstdiagnose bei Ausfall einer Komponente geschaffen. Letztlich wurde ermöglicht, grundlegende Fähigkeiten neu zu kombinieren und damit das Fahrverhalten in Grenzen zu adaptieren.

Auf Basis des 4D-Ansatzes [Dickmanns 87] wurde die Bildverarbeitung erweitert und berücksichtigt die geometrische Anordnung von Objekten (drei Dimensionen) und deren zeitliches Verhalten (vierte Dimension). Merkmale eines Objekts in der 2D-Bildebene werden mit Hilfe von internen vierdimensionalen Modellen abgeglichen. Mit Hilfe von Objekthypothesen kann der Verfolgungsprozess gezielt im zweidimensionalen Kamerabild rechenaufwendige Merkmalsextraktionen starten. Für den Mechanismus der Blickfixierung und Objektverfolgung werden die durch die Bildverarbeitung und den 4D-Ansatz gemessenen Positionen und Geschwindigkeiten der Objekte im Bild herangezogen. Beim 4D-Ansatz werden mit Hilfe von rekursiven Schätzverfahren, bei denen Form und Bewegung beschreibende Objektmodelle und -hypothesen ihre Anwendung finden, Sensorsignale erwartungsbasiert ausgewertet.

Ein weitergehender Überblick über den internationalen Status modellbasierter Fahrersysteme und Fahrerassistenzsysteme ist in [Gregor 02] aufgeführt.

2.2 Systeme auf der Basis lernender Modelle

Lernende Modelle sind dadurch gekennzeichnet, dass die Architektur des lernenden Systems explizit auf die zu lernende Aufgabe zugeschnitten (d.h. modelliert) oder zumindest autonom parametrisiert wird.

S. Ramachandran beschreibt in seiner Arbeit [Ramachandran et al 05] ein System, welches Verstärkungslernen dazu verwendet, zu lernen, die beste Auswahl aus mehren Aktionsmöglichkeiten zu identifizieren. Ein mobiler Roboter für den Innenraumbereich hat die Aufgabe, sich in einem Gebäude zu bewegen und dabei Kollisionen zu vermeiden. Mehrere Software-Module zur Steuerung des mobilen Roboters, wie z.B. Linksdrehung, Rechtsdrehung, Stopp, Geradeausfahrt und Erkundungsfahrt bieten entsprechende Steuerkommandos für die Aktoren des mobilen Roboters an – dabei greifen diese Module jeweils auf die Daten verschiedener Sensoren zu. Ein übergeordnetes

lernendes System, basierend auf Verstärkungslernen, hat die Aufgabe, eine jeweils angemessene Aktion auszuwählen.

Der Hintergrund dieser Vorgehensweise liegt darin, dass einzelne Bewegungsabläufe von jeweils spezialisierten Modulen berechnet und vorgegeben werden – eine übergeordnete Fahrzeugsteuerung muss aber zwischen unterschiedlichen Manövern abwägen. In sich einer möglicherweise verändernden Umgebung ist es schwer, diese Aufgabe im Rahmen des Systemdesigns a-priori festzulegen und damit wird die Forderung nach einem adaptiven Lernverfahren erhoben.

Die Dissertation von [Gloye 05] diskutiert die erfolgreiche Anwendung von Lernmethoden zur Verhaltensoptimierung autonomer, mobiler Roboter im Rahmen des RoboCup Small-Size-Teams der Freien Universität Berlin (FU-Fighters).

Das Spektrum dieser Dissertation ist breit gefächert und zeigt, dass Lernverfahren bei realen mobilen Robotern erfolgreich eingesetzt werden können. Dabei verfügen die Roboter über eine modellbasierte Regelung, deren Parameter über unterschiedliche Lernmethoden optimiert werden. Im Detail kommen sowohl Neuronale Netze, als auch Verstärkungslernen-Algorithmen zum Einsatz. Die unterschiedlichen Lernmethoden übernehmen primär die Parameteranpassung der Regelkreise der Roboter.

Dabei sind die Aufgaben des Robotersystems vielschichtig: Zum einen muss gelernt werden, zukünftige Roboterpositionen und -orientierungen vorherzusagen. Für die Prognose werden neben den aktuellen und vergangenen Bewegungszuständen auch die vergangenen Fahrbefehle für den Roboter berücksichtigt. Es werden unterschiedliche Vorhersagemethoden miteinander verglichen. Durch die Messung der Bildverarbeitungsgenauigkeit wird eine obere Schranke für die Vorhersagegenauigkeit berechnet. Zum anderen wird die Bewegung des Roboters gelernt - unter anderem aufgrund des Zusammenhangs zwischen gesendeten Fahrbefehlen und der vollzogenen Fahrausführung. Dadurch können die Fahrbefehle korrigiert werden, um die Fahrweise des Roboters zu optimieren. Letztlich wird die bestmögliche Bremsfunktion gelernt. Die Bremsfunktion des Roboters dient dem perfekten Anhalten des Roboters, so dass er einerseits nicht über die Zielposition hinausfährt und anderseits nicht zu langsam an das Ziel heranfährt.

In Bezug zu modellbasierten Systemen wurde auch eine interessante Kombination mit einem Neuronalen Netz untersucht, deren Zielstellung es war, die Parameter für das gewählte Modell zu erlernen [Kuhnert & Dong 03]. Ein Fahrersystem übernimmt die Querregelung eines Fahrzeugs entlang von Geraden und unterschiedlichen Kurven. Dabei wird ein Sollpfad errechnet und die Sollpfad-Verfolgung wird mit vorgegebenem Geschwindigkeitsprofil durchgeführt. Auf Basis der Sollpfadbestimmung durch ein übergeordnetes Modul (aufgrund der Auswertung von visueller und odometrischer Daten), wird ein adaptiver Regler auf Basis eines Neuronalen Netzes vorgestellt, der obiges Pfadfolgeproblem löst.

2.3 Systeme auf der Basis lernender Systeme

Im Gegensatz zur Modellierung wird bei den autonomen Lernverfahren die Interaktion mit der Umfeld nicht vorgegeben (z.B. durch Modellparameter), sondern im Laufe der Zeit weitgehend autonom erlernt.

Ziel beim Einsatz von Lernverfahren ist es, das autonome System um die Fähigkeit zu erweitern, selbständig einen Verhaltsraum aufzubauen und nicht von vorgegebenem Wissen abhängig zu sein. Die eigenständige Interaktion mit der Umgebung ist trotz immenser Steigerung von Rechenleistungen und Vielzahl an Sensoren noch in keiner Weise mit der von biologischen Wesen vergleichbar. Insbesondere in einer komplexen Umgebung oder bei Systemen mit einem hohen Grad an Handlungsfreiheiten besteht noch ausgeprägter Forschungsbedarf.

In Verbindung mit autonomen, mobilen Systemen sind bisher insbesondere die Verfahren basierend auf Neuronalen Netzen, Sensordatenauswertung und Verstärkungslernen zum Einsatz gekommen, auf die im Folgenden weiter eingegangen wird.

2.3.1 Neuronale Netze

Neuronale Netze werden oft eingesetzt, Lernprozesse von biologischen Wesen nachzubilden. Vorwärtsgerichtete, mehrschichtige Neuronale Netze sind Funktionsapproximatoren, die zu einem Eingabevektor einen bestimmten Ausgabevektor erzeugen. Dabei sind die Grundfunktionen, aus denen sich das Netz zusammenbaut möglichst einfach gehalten.

Ein Neuronales Netz kann so entworfen werden, dass es eine bestimmte Abbildungsfunktion erfüllt. Populär sind die Neuronalen Netze dadurch, dass sie Funktionen an Hand von Beispielen lernen können. Zum Training eines Neuronalen Netzes wird oft der Backpropagation-Algorithmus verwendet, ein Verfahren, welches durch Zurück-Propagierung in die vorderen Schichten den Ausgabefehler des Netzes minimiert.

Eine ausführliche Erläuterung von Neuronalen Netzen und des Backpropagation-Algorithmus ist in z.B. [Rojas 96] zu finden.

In Bezug auf das Ziel einer autonomen Fahrzeugsteuerung wurde eine Vielzahl an Forschungen basierend auf Neuronalen Netzen durchgeführt.

Bereits frühzeitig wurden an der Carnegie Mellon University Forschungen zu Neuronalen Netzen für den Einsatz in Fahrsystemen betrieben - das Systems ALVINN (Autonomous Land Vehicle in a Neural Network) wurde Anfang der 90er Jahre in Betrieb genommen. [Pommerlau 91] beschreibt das System mit einer direkten Verbindung zwischen der Bildverarbeitung und einem Neuronalen Netz. Dieser Ansatz erbringt gute Fahrergebnisse, jedoch nur dann, wenn

die Einzelbilder der Trainingssequenzen den Bildern im Fahrmodus hinreichend ähnlich sind.

Dieser Ansatz wurde so weiterentwickelt, dass das Gesamtsystem aus mehreren Neuronalen Netzen bestand [Jochem et al. 93], aber auch dieser Ansatz konnte die Abhängigkeit des Lernerfolges von den Trainingsmustern nicht aufheben. Weitere Forschungen nutzten weiter Sensoren, z.B. ein GPS System [Jochem et al. 95], zur Unterstützung der Orientierung - andere Ansätze erweiterten das Neuronale Netz um Komponenten der Objekterkennung um zwischen Wege-Folgung und Hindernis-Erkennung besser unterscheiden zu können [Baluja & Pommerlau 97], [Franke et al. 98]. Allerdings konnte auch in diesen Variationen die Abhängigkeit von den Trainingsmustern nicht vermieden werden. Ebenso ist das gelernte Wissen in einem Neuronalen Netz nicht in einer Form hinterlegt, die von externen Prozessen interpretiert oder gar optimiert werden kann.

Auch im Umfeld anderer Lernaufgaben, stoßen Implementierungen Neuronaler Netze an ihre Grenzen. [Anderson & Hong 94] beschreibt den Vergleich eines einstufigen Neuronalen Netzes im Vergleich zu einem hierarchisch strukturierten Neuronalen Netz. Die Aufgabe bezieht sich auf das Aufschwingen und Balancieren eines einachsigen Pendels. Im Detail ist das Gesamtsystem aus mehreren Einzelsystemen basierend auf Neuronalen Netzen aufgebaut. Ein übergeordnetes System auf der Basis von Verstärkungslernen koordiniert die Generierung des auszugebenden Steuersignals unter Verwendung der Ergebnisse der untergeordneten Neuronalen Netze. Dabei zeigt sich, dass auch die Aufteilung des Neuronalen Netzes auf mehrere kleinere Einheiten keine Verbesserung bringt.

2.3.2 Aktionsgenerierung auf der Basis von Sensorenauswertung

Die Generierung von Aktionen wie z.B. Steuerkommandos kann auf der Basis der Sensordaten erfolgen, ohne diese im Detail zu interpretieren.

In [Graefe 99] wird ein System vorgestellt, welches eine kalibrierungsfreie Navigation eines Roboters auf der Basis visueller Informationen ermöglicht. Hintergrund ist der Bedarf einer universellen Robotersteuerung, die weder modelliert, noch kalibriert oder trainiert werden muss. Es wurde erkannt, dass insbesondere das Kalibrieren von modellbasierten Robotersteuerungen aufwendig und fehleranfällig ist. Zum einen ist selbst ein sehr aufwendig kalibriertes Modell lediglich eine Annäherung an den tatsächlichen Zustand eines Roboters oder dessen Umgebung. Zum anderen verfälschen Abnutzung und Alterung eines Roboters und dessen Komponenten die Kohärenz zum kalibrierten Modell.

Neben dem Wegfall des Kalibrierungsaufwandes ist davon auszugehen, dass sich kalibrierungsfreie Roboter auch durch eine autonome Anpassungsfähigkeit an Veränderungen von Umgebung oder Roboter auszeichnen. In der Ar-

beit von 1999 wird ein 5-achsiger Roboter beschrieben, der über einen Greifarm Objekte in Reichweite anfährt und greift.

Konzeptionell ist entscheidend, das der Controller des Roboters nicht versucht, auf Basis der erhaltenen Sensordaten (hier: die Bildfolgen zweier Kameras) auf die Position von Objekt, Roboter oder Kameras im dreidimensionalen Raum zurückzuschließen. Vielmehr werden die erhaltenen Sensordaten als Daten einer Black Box behandelt ohne dieser weiter zu interpretieren. Aus diesen Datenströmen wird lediglich der Erfüllungsgrad der Mission ermittelt. Nach Ausgabe eines weiteren Steuerkommandos werden erneut die Sensordaten auf den Erfüllungsgrad der Mission untersucht und ermittelt, ob das letzte Steuerkommando in Bezug auf die übergeordnete Mission dienlich oder schädlich war. Die Kombination von Sensordaten, ausgegebenem Steuerkommando und resultierender Sensordatenänderung wird als Datensatz gespeichert und baut autonom über die Zeit eine Wissensbasis auf. Das System lernt somit auf Basis von statistisch verteilten Versuchen; über die Analyse der Wissensbasis lassen sich Auswirkungen von Aktionen vorhersagen, bzw. zu einer durchzuführenden Aufgabe die entsprechenden Steuerkommandos identifizieren oder interpolieren.

In Konsequenz liegt die Wissensbasis in Form von explizit analysierbaren Datensätzen vor, die jederzeit spezifisch modifiziert oder interpretiert werden können. Eine Trainingsphase ist nicht nötig – das Gesamtsystem baut seine Wissensbasis komplett autonom auf. Die Weiterentwicklung der Wissensbasis ist jederzeit möglich, d.h. die Wissensbasis ist jederzeit erweiterbar.

Das beschriebene Konzept wurde weiterentwickelt und [Bischoff & Graefe 04] beschreibt die Implementierung für ein mobiles System, welches autonom lernt, einen Korridor oder Gang möglichst gleichmäßig entlangzufahren, ohne an Wände oder seitlichen Türen anzustoßen. Dabei muss sich das System an verschiedene Arten von Korridoren (d.h. unterschiedliche Korridore in unterschiedlichen Gebäuden) autonom anpassen. Trotz der zusätzlichen Komplexität (aufgrund der Eigenbewegung der Gesamtanordnung) konnte das System erfolgreich auf zwei unterschiedlichen mobilen Robotern in verschiedenen Gebäuden eingesetzt werden.

Offen scheinen noch die Fragen, wann, im Falle von Änderung von Robotersystem oder Umgebung, die Wissensbasis weiterentwickelt und wann sie verworfen und komplett neu aufgebaut werden sollte (d.h. es bedarf einer Entscheidungsinstanz, wie situationsangemessen die ermittelten Systemkommandos sind). Auch ist noch offen, wie das System mit dem lokalen Minimum/ Maximum-Problem umgeht, wie es z.B. aus zeitweise gestörten oder fehlenden Sensordaten resultieren kann.

2.3.3 Verstärkungslernen

Verstärkungslernen (Reinforcement Learning) ist eine Lernmethode bei der in Umgebung und lernenden Agenten unterschieden wird. Diese bestehen aus

einer Menge von Zuständen (states), einer Menge von Aktionen (actions), einer Bewertungsfunktion (value function), einer Strategiefunktion (policy) und einer Belohnungsfunktion (reward function). Auf die Details wird in Kapitel 4 genauer eingegangen.

Grob dargestellt wird ein Verstärkungslernen-System dadurch implementiert wird, dass ein Agent einen Zustand übermittelt bekommt, der sich aus der Situation ableitet, in der sich das lernende System befindet. Zusätzlich erhält der Agent eine Belohnung für diesen Zustand, z.B. in wie weit ein Lernziel bereits erreicht oder eingehalten wurde. Auf der Basis von Zustand und Belohnung wählt der Agent eine Aktion aus der Menge von endlich vielen möglichen Aktionen aus, die meist beim nächsten (diskreten) Zeitschritt in einem anderen Zustand resultiert. Es gibt unendlich viele Zustände. Daraus folgt, dass bei vielen Anwendungen die Umgebung diskretisiert werden muss. Die Beziehung zwischen Umgebung, Situationsbeschreibung und Zustand ist in Kapitel 1.4 beschrieben.

Der Agent verwendet die erhaltenen zustands-aktions-spezifischen Belohnungen zum Aufbau einer Bewertungsfunktion im Verhaltensraum. Auf diese Weise wird eine zeitlich längerfristige Bewertung der Aktion zum gegebenen Zustand vorgenommen. Deshalb dient diese Bewertungsfunktion auch der Identifizierung der angemessenen Aktion innerhalb eines Zustands. Die Auswahl wird durch die Strategiefunktion (policy) vorgenommen – üblicherweise wird bei einer Schar von möglichen Aktionen diejenige ausgewählt, die durch den höchsten Wert der Bewertungsfunktion identifiziert wird.

Die Art, wie der Agent lernt - ob er sich vergangene Bewertungen merkt, wie er die Aktionen berechnet usw. ist nicht festgelegt. Das wichtigste Kriterium zur Unterscheidung von Verstärkungslernen von anderen Lernmethoden ist dabei, dass dem Agenten das Ziel nicht direkt, sondern nur indirekt über die Belohnungen mitgeteilt wird.

In [Mitsunaga et al 05] wird z.B. ein Verstärkungslernen-System beschrieben, welches die Interaktion mit einem Menschen lernt – d.h. die eigene Gestikulation an eine mit dem Roboter interagierende Person anpasst. Dabei werden in Bezug auf eine menschliche Person Elemente der Körpersprache wie z.B. Abstand zum Roboter, Blickrichtung der menschlichen Person, Gestikulierung, etc. erfasst und als Eingangsgröße für das lernende System genutzt. Als Ziel dieser Arbeit wurde das adaptive Verhalten des Roboters in Bezug auf menschliche Körpersprache festgesetzt. Dabei erlaubt die Adaptionsfähigkeit des Systems das Anpassen der Körpersprache des Roboters auf unterschiedliche Personengruppen.

[Gasket et al 00] beschreibt ein autonomes mobiles System, welches die Aufgabe hat, eigenständig in einem Raum zu manövrieren und Objekte zu erkennen. Diesen Objekten ist entweder auszuweichen oder zu folgen. Das beschriebene System umfasst zwei Grundkomponenten, die in Objektverfolgung und Erkundung unterschieden werden. Der Modus der Objektverfolgung wird jeweils dann aktiviert, wenn ein nach vorgegeben Kriterien als interessant

eingestuftes Objekt erkannt wird. Diesem Objekt ist entweder zu folgen oder im Falle eines festen Objektes ist dieses zu umfahren, um Informationen aus unterschiedlichen Perspektiven aufzunehmen. Der Modus Erkundung ist jeweils dann aktiv, wenn der Modus Objektverfolgung inaktiv ist. In diesem Modus bewegt sich das mobile System durch den Raum und klassifiziert erkannte Objekte nach vorgegeben Regeln. Ursprünglich wurde die Steuerung für beide Modi auf Basis eines modellbasierten Reglers implementiert – diese Implementierung erforderte Kalibrierungs- und Parametrisierungsaktivitäten. Im Zuge der Weiterentwicklung wurde die Steuerung der beiden Modi auf ein Lernverfahren basierend auf Verstärkungslernen umgesetzt. Interessant ist dabei die gewählte Implementierung, in der der für Verstärkungslernen aufzubauende Verhaltensraum über ein Neuronales Netz approximiert wird.

Eine ähnliche Vorgehensweise wird auch in [Onat 98] beschrieben, d.h. dem Ersatz der Verhaltensraum-Wertetabelle durch ein Neuronales Netz. Letztere Arbeit befasst sich zwar mit einer anderen Anwendung (im Detail: dem Aufschwingen und Balancieren eines einachsigen Pendels) – es wird aber ein direkter Vergleich des Speicherbedarfs zwischen Wertetabelle und Neuronalem Netz erstellt, um die Werte des Verhaltensraums eines lernendes Systems auf Basis von Verstärkungslernen aufzunehmen, bzw. zu erhalten. Der Vorteil des Neuronalen Netzes konnte aber nicht nachgewiesen werden – konkret benötigte die Implementierung des Neuronalen Netzes sogar mehr Speicher als eine Wertetabelle und es wurde lediglich ein positiver Ausblick zugunsten dem Speicherbedarf Neuronaler Netze bei komplexeren Aufgabenstellungen angedeutet.

Eine detaillierte Abhandlung von Verstärkungslernen in Kombination mit Neuronalen Netzen ist in [Kretchmar 00] dargestellt. Das Neuronale Netz dient als Funktionsapproximator für den Verhaltensraum des lernenden Systems auf Basis von Verstärkungslernen. Auf theoretischer Basis wird nachgewiesen, dass dieser kontinuierliche Verlauf der Werte im Verhaltensraum ein robusteres Gesamtsystem ermöglicht.

Die Arbeit von Vollbrecht [Vollbrecht 99] beschreibt eine Architektur auf Basis von Verstärkungslernen, um eine komplexe Lernaufgabe ohne Modell, d.h. ohne Vorgabe von Systemdynamiken zu meistern. Dabei wird die komplexe Lernaufgabe in mehrere weniger komplexe Teilaufgaben zerlegt. Hintergrund ist die Erwartung, dass Aufgabenstellungen mit ausgeprägter Komplexität besser über eine Kombination aus mehreren System basierend auf Verstärkungslernen adressiert werden können – dabei lernt jedes untergeordnete Teilsystem eine Teilaufgabe. Die Gesamtarchitektur des Systems weist eine hierarchische Anordnung auf und eine übergeordnete Instanz entscheidet über die Selektion, bzw. Kombination der Steuerkommandos.

Die Schwerpunkte der Arbeit liegen zum einen in der Aufteilung der komplexen Aufgabenstellung in mehrere Unteraufgabenstellungen. In der vorliegenden Arbeit wird diese Aufgabe manuell durchgeführt – im Detail für eine konkret Aufgabenstellung: das Rückwärtsfahren eines LKW's inklusive Andocken an einen virtuellen Anhänger. Dabei ist zu beachten, dass sowohl ein maxi-

maler Lenkwinkel als auch ein maximaler Winkel zwischen Fahrerkabine und Hänger nicht überschritten werden darf. Zudem muss sich das Fahrzeug innerhalb einer vorgegeben Fläche bewegen, d.h. darf diese nicht überschreiten. Bezüglich der Unterteilung der Aufgabe wird angeregt, auch diese Aufgabe zukünftig erlernen zu lassen, d.h. Regeln und Verfahrensweisen zur Unterteilung von Aufgaben automatisiert durchführen zu lassen.

Ein weiterer Schwerpunkt der Arbeit liegt zum anderen darin, wie die Ausgabegrößen, bzw. Steuerkommandos der unterschiedlichen Teilsysteme kombiniert werden. Dazu werden 3 Prinzipien vorgestellt. Das Veto-Prinzip (veto principle) ermöglicht einem Untersystem die Ausführung eines Steuerkommandos eines anderen Untersystems zu unterbinden. Sollte z.B. von einem Untersystem erkannt werden, dass die Ausführung eines Steuerkommandos eines anderen Untersystems zu einer Kollision führt, dann kann das erkennende Untersystem dies über ein Veto verhindern. Im Rahmen des Teilaufgaben-Prinzips (subtask principle) wird situationsspezifisch die Steuerung des Gesamtsystems einem Teilsystem übertragen. Dabei wird eine übergeordnete Kontroll- und Entscheidungsinstanz benötigt, die die Situationserkennung und Untersystemzuweisung durchführt – auch diese Instanz wird auf Basis von Verstärkungslernen implementiert. Ein drittes Prinzip, das Beeinflussungsprinzip (perturbation principle) ermöglicht die Beeinflussung des Lernziels eines Untersystems durch ein anderes. Durch diese Interaktion zwischen den Untersystemen wird eine dynamische Anpassung bzgl. der Lernziele der Untersysteme ermöglicht.

In Summe wird in der Arbeit gezeigt, dass die Aufteilung der komplexen Lernaufgabe in Unteraufgaben erreicht wird und dass die Komplexität der Unteraufgabenstellungen jeweils geringer als die Gesamtkomplexität ist. In Folge sind die Verhaltensräume der jeweiligen Teilsysteme auf Basis Verstärkungslernens kleiner und das jeweilige Lernziel kann schneller erreicht werden.

John Shakleton beschreibt in [Shackleton & Gini 97] ebenso eine Implementierung von Verstärkungslernen, die als hierarchische Anordnung von Teilsystemen aufgebaut ist. Ein autonomes System wird vorgestellt, welches innerhalb gegebener Flächen unterschiedlicher Abgrenzungen platzierte Kisten verschieben muss. Dabei wird als Lernziel vorgeben, in einer festgelegten Zeit möglichst viele Kisten zu verschieben. Basierend darauf wird zunächst in drei grundlegende Zustände unterschieden: Kisten finden (Finder), Kisten verschieben (Pusher) und Befreien (Unwedger). Jedem der grundlegenden Zustände wird eigenes lernendes System mit jeweils eigenem Verhaltensraum zugeordnet und eine übergeordnete Instanz entscheidet, welche der berechneten Aktionen tatsächlich ausgegeben wird. Dabei fokussiert sich die Arbeit unter anderem auf die Ermittlung der Effektivität des Lernverhaltens – d.h. wie kann bei lernendes Systemen auf Basis von Verstärkungslernen ermittelt werden, wie verlässlich die ermittelte Aktion ist. Diese Anforderung wird auf Basis der Zielerreichung beantwortet, d.h. einem Vergleich zwischen Testreihen, wie viele Kisten verschoben wurden. Alternativ, da vom Gesamtsystem auch eine Erkundung der Umgebung erwartet wird, wird ermittelt, welche Strecke vom autonomen System zurückgelegt wurde. Eine Bewertung der

Daten des Verhaltensraumes erfolgt nicht. Eine weitere Frage der Arbeit beschäftigt sich mit der Architektur des Systems. Wie eingangs beschrieben, wird eine übergeordnete komplexe Aufgabe in Unteraufgaben unterteilt. Diese Aufteilung erfolgt manuell – zunächst in 3 Teilaufgaben und später in 6 Teilaufgaben. Vergleichbar mit der Arbeit von [Vollbrecht 99] wird gezeigt, dass der Erfolg eines hierarchischen Systems klar auf eine sinnvolle Aufteilung angewiesen ist – dabei wird diese Aufteilung ebenso noch manuell durchgeführt. Letztlich behandelt die Arbeit die Frage der Anpassungsfähigkeit des Systems an unterschiedliche Formen der Umgebung. Dabei kann diese Frage nicht eindeutig beantwortet werden, da in diesem Zusammenhang unterschiedliche Ergebnisse im Falle einer direkten Übertragung von gelerntem Verhalten auf eine neue Umgebung erzielt wurden. Durch die allgemeine Adaptionsfähigkeit von Verstärkungslernen sollte aber in allen Fällen eine mittelfristige Anpassung der Verhaltensräume erfolgen.

Verstärkungslernen wird oft in Kombination mit der Aufgabe eingesetzt, ein Pendel zu balancieren. In [Szita et al 02] wird ein 2-Achsen Pendel beschrieben, welches vom stabilen Zustand (hängend) in den instabilen Zustand (stehend) gebracht und in diesem gehalten, bzw. balanciert werden muss. Dabei hat das Pendel zwei Pendelelemente – ein horizontales und vertikales. Das horizontale Pendelelement kann durch einen Stellmotor horizontal beliebig gedreht werden. An einem Ende des horizontalen Pendelelements befindet sich über ein Gelenk das vertikale Pendelelement, an dessen anderem Ende eine Masse m befestigt ist. Über die Drehung des horizontalen Pendelelements kann das vertikale Pendelelement in Schwingungen und zum Aufschwingen und Balancieren gebracht werden. Dies entspricht der Aufgabe des lernenden Systems – d.h. das Aufschwingen und Balancieren des vertikalen Pendelelements über die Drehung des horizontalen Pendelelementes. Eine übergeordnete Überwachung übernimmt die Belohnungsfunktion. Mittels entsprechender Testreihen konnte die vorgegebene Aufgabe erfolgreich gelernt werden.

Eine Implementierung eines Fahrsystems, d.h. die Aufgabe einer Streckenverfolgung wird in [Oh et al 00] beschrieben. Das System auf Basis von Verstärkungslernen erhält als Situation die Seitenablage, d.h. die Abweichung des Fahrzeugs von der Fahrbahnmitte, und optimiert über die auszugebenden Aktionen das Zurückfahren zur Fahrbahnmitte. Dabei wird in der Beschreibung der Arbeit der Aspekt der lernenden Fahrzeugsteuerung betont – im Unterschied zu Aspekten der Bildauswertung und Objekterkennung bei vielen anderen Arbeiten in ähnlichem Umfeld. Im Detail wird ein System basierend auf Verstärkungslernen in Kombination mit einem Neuronalen Netz vorgestellt, welches nachweisbare Erfolge erzielte, gute (im Sinne von angemessen), aber nicht zwangsläufig die besten Aktionen zu identifizieren. Hintergrund für dieses Verhalten ist eine adaptive Erkundung des Verhaltensraumes – d.h. das System begnügt sich mit der Identifikation eines lokalen Maximums im Verhaltensraum. Trotzdem konnten gute Ergebnisse sowohl auf Basis eines Simulators, als auch in Kombination mit einem echten Fahrzeug nachgewiesen werden.

Ein in doppelter Hinsicht autonomes System wird in der Arbeit von [Goerke & Henne 05] beschrieben, in der das autonome Ausführen von ganzen Bewegungsabläufen gelernt wird. Dabei ist das Ziel nicht ein finaler Zustand sondern ein harmonischer Wechsel zwischen verschiedenen Zuständen. Ein mobiler Roboter bewegt sich autonom in einer abgegrenzten Umgebung und bekommt an verschiedenen Stellen unterschiedliche Signale übermittelt. Auf der Basis dieser Signale wird das Verhalten des Roboters autonom verändert. Diese Signale sind Informationen über die Umgebung (z.B. erkannte Ecken, Passagen, Hindernisse, etc.) aber auch Information wie z.B. Heimat, Nahrung oder Energieverbrauch. Diese Signale beeinflussen die Motivation des Roboters, wobei in vier Grundmotivationen unterschieden wird: Erschöpfung, Hunger, Heimweh und Neugier. So führt z.B. das Ausbleiben des Signals Nahrung zum Anstieg der Grundmotivation Hunger oder das übermäßige Erkunden der Umgebung zur Grundmotivation Erschöpfung. Je nach Kombination der 4 Grundmotivationen wird ein übergreifender Zustand des Roboters definiert, dabei wird in Furcht, Verärgerung, Langeweile und Zufriedenheit unterschieden. Dieser übergreifende Zustand entscheidet über ganze Bewegungsabläufe, die in einen jeweils spezifischen Erhalt von den eingangs beschriebenen Signalen münden. Da die Aufgabe des Roboters darin besteht, ein eigenes Gleichgewicht zwischen den Zuständen zu finden – unabhängig davon, wie dieses Gleichgewicht sich im Detail abzeichnet. Aus diesem Grund wäre ein überwachtes Lernen oder eine Vorgabe von Lernmustern nicht angebracht und die eigentliche Lernfunktion wird dabei auf Basis von Verstärkungslernen implementiert.

Dabei kann Verstärkungslernen auch dazu verwendet werden, Modellparameter zu lernen. In [Atkeson & Santamaria 97] wird der Vergleich zwischen purem Verstärkungslernen und Erlernen der Modellparameter am Beispiel des invertierten Pendels verglichen. Dabei wird im Versuch eine vereinfachte Variante des invertierten Pendelproblems verwendet, da über einen Motor direkt ein Drehmoment auf das im Ruhezustand nach unten hängende Pendel ausgeübt wird (normalerweise wird kein direktes Drehmoment auf das Pendel ausgeübt, sondern es ist auf einem seitlich steuerbaren Wagens angebracht). Für diese Anordnung kann nachgewiesen werden, dass das Lernen von Modellparametern das Ziel (d.h. das Balancieren des Pendels) schneller und stabiler erreicht wird, als beim Einsatz des puren Verstärkungslernens. Auch bei einer Änderung von Umgebungsparametern adaptiert das System, bei dem die Parameter eines Modells gelernt werden, schneller und stabiler. Dabei ist festzuhalten, dass diese sehr einfache Anordnung nicht repräsentativ für alle möglichen Lernaufgaben ist und einige Vorteile des Verstärkungslernens (wie z.B. das Kompensieren von fehlerhaften oder gestörten Belohnungen) nicht zum Einsatz kommen.

3 MERKMALSEXTRAKTION AUS VISUELLEN EINGANGSDATEN

Die dieser Arbeit zugrunde liegenden Sensordaten sind Bildfolgen einer am Fahrzeug befestigten Kamera. Aus diesen Bildfolgen sind die bedeutungstragenden Merkmale zur Interpretation der Bildfolgen zu extrahieren.

Kamera und Fahrzeug verfügen über ein jeweils eigenes Koordinatensystem. Um aufgenommene *Einzelbilder* (einzelnes Bild einer Bildfolge) richtig interpretieren zu können, muss die Beziehung zwischen den beiden Koordinatensystemen berücksichtigt werden. Die Beziehung zwischen dem Koordinatensystem der Kamera und dem Koordinatensystem des Fahrzeugs wird in Beug auf die vorliegende Arbeit in Kapitel 3.1 erläutert.

Die wesentlichen Informationen des Streckenverlaufs werden aus den Fahrbahnmarkierungen ermittelt. Von diesen Fahrbahnmarkierungen wird ein gewisser Verlauf im Bild erwartet (z.B. bei Geraden auf einen Fluchtpunkt am Horizont zulaufend). Diese Tatsache wird beim Extrahieren der Fahrbahnmarkierungen verwendet: Bildpunkte, die aufgrund von Farbunterschieden als mögliche Fahrbahnmarkierungspunkte identifiziert werden, müssen in eine vorgegebene Richtung verkettbar sein. Diese vorgegebenen Verkettungsrichtungen werden aus Daten gebildet, die bereits vorher (durch gezielte Analyse von Einzelbildern oder Bildfolgen) aufgebaut werden und für spätere Testläufe in einer Datenbank gespeichert werden. Hintergrund und Verfahrensweise dieses Verkettungsverfahren werden in Kapitel 3.2 konzeptionell dargestellt.

3.1 Koordinatentransformation

Da sich die vorliegende Arbeit auf die Auswertung von Bildfolgen bezieht, die über eine Videokamera gewonnen werden, müssen die unterschiedlichen Koordinatensysteme von Kamera und Fahrzeugs berücksichtigt werden.

Die generelle Thematik der Koordinatentransformation bei der Auswertung von Bildfolgen wird unter anderem in [Faugeras 93], [Enkelmann 97] und [Henke 98] dargestellt. Dieser Abschnitt erläutert diesen Hintergrund und wendet ihn explizit auf die in dieser Arbeit gegebenen Rahmenbedingungen an.

Dazu sei ein Koordinatensystem des Fahrzeugs mit den Komponenten (x_v, y_v, z_v) gegeben. Der Ursprung dieses Koordinatensystem liegt in der Mitte der Flächenmittelpunkte der beiden Berührungsflächen von Vorderreifen und Boden. Die X-Achse verläuft in Richtung Flächenmittelpunkt der Berührungsfläche von rechtem Vorderrad und Boden, die Y-Achse verläuft entgegen der Gravitation nach oben und die Z-Achse verläuft orthogonal dazu nach vorne.

Des Weiteren sei ebenfalls das Koordinatensystem einer im oder am Fahrzeug befestigten Kamera mit den Komponenten (x_c, y_c, z_c) gegeben. Der Ursprung dieses Koordinatensystems liegt in der Mitte des Bildsensors einer über der Vorderradachse befestigten Kamera. Dieses Kamera-Koordinatensystem sei um die x-Achse gedreht (Neigungswinkel der Kamera ist gesenkt) und in y-Richtung um die Distanz d_y verschoben. Beide Koordinatensysteme sind in Abbildung 3.1 dargestellt.

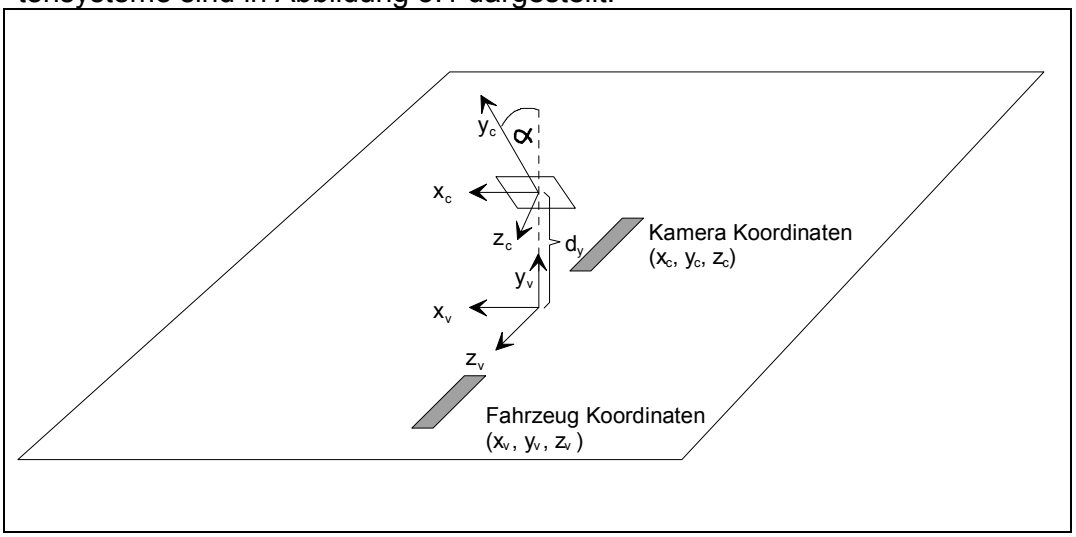


Abbildung 3.1: Koordinatensysteme von Kamera und Fahrzeug

Die Beziehung zwischen den Koordinatensystemen des Fahrzeugs (x_v, y_v, z_v) und der Kamera (x_c, y_c, z_c) ist gegeben durch:

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x_v \\ y_v - d_y \\ z_v \end{pmatrix} \tag{3.1}$$

Durch die Abbildung mittels Kamera wird die dreidimensionale Welt auf ein zweidimensionales Bild mit dem Sensorkoordinatensystem (x_s, y_s) abgebildet. Die Beziehung zwischen Welt-Koordinatensystem und Sensor-Koordinatensystem ist in Abbildung 3.2 anhand eines einfachen Lochkameramodells dargestellt.

Damit gilt für die Beziehung zwischen Kamera-Koordinatensystem und Sensor-Koordinatensystem:

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \frac{f_1}{f_2} \begin{pmatrix} x_c \\ y_c \end{pmatrix} \tag{3.2}$$

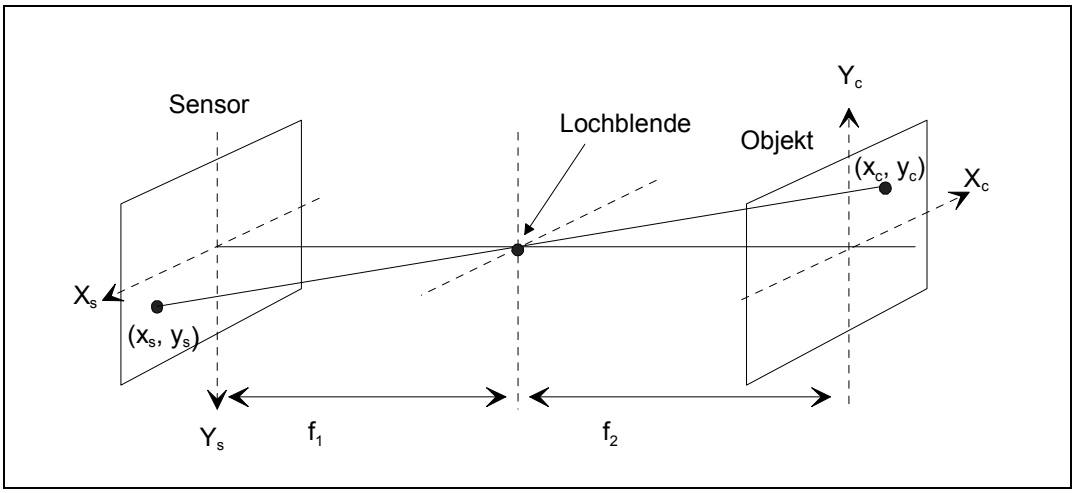


Abbildung 3.2: Koordinatensysteme von Sensor und Objekt

Formel (3.2) kann mit Hilfe von Formel (3.1) wie folgt umgeformt werden:

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \frac{f_1}{f_2} \begin{pmatrix} x_v \\ y_v \cdot \cos\alpha - d_y \cdot \cos\alpha + z_v \cdot \sin\alpha \end{pmatrix} \tag{3.3}$$

Es wird nun angenommen, dass die Fahrbahn keinen ausgeprägten hügeligen Verlauf hat (d.h. keine großen Kuppen oder Täler). Weil sich die Fahrbahnmarkierungen auf der Fahrbahn befinden, kann ohne Beschränkung der Allgemeinheit y_v gleich Null gesetzt werden, was Formel (3.3) wie folgt vereinfacht:

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \frac{f_1}{f_2} \begin{pmatrix} x_v \\ z_v \cdot \sin\alpha - d_y \cdot \cos\alpha \end{pmatrix} \tag{3.4}$$

Schließlich wird das über die Videokamera digitalisierte Bild auf einem Bildschirm oder im Speicher dargestellt, wobei sich Höhe und Breite entsprechend verändern können. Auch liegt der Ursprung des Koordinatensystems eines sich im (Bild-)Speicher befindlichen Bildes in der linken oberen Ecke und nicht mehr im Zentrum des Bildes wie in Abbildung 3.3 gezeigt.

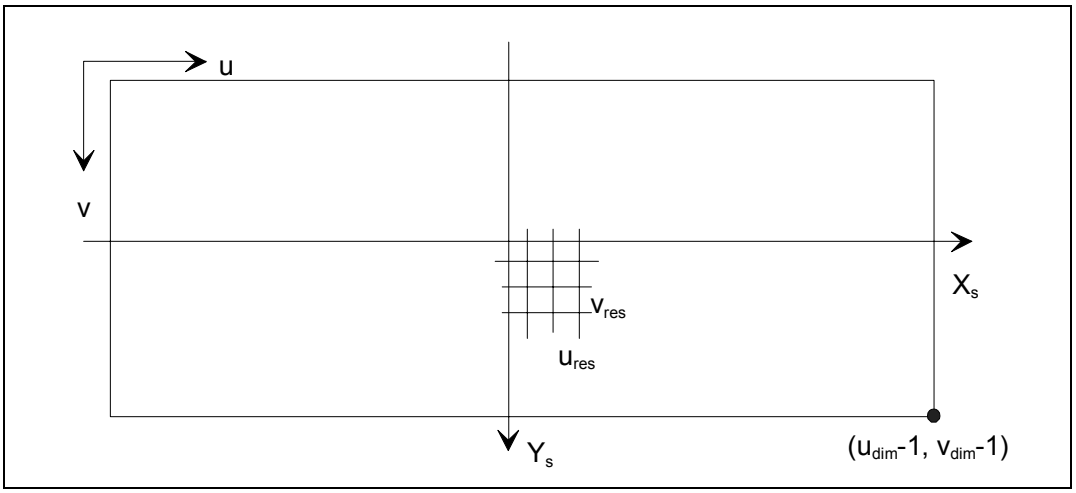


Abbildung 3.3: Koordinatensysteme von Sensor und Bildschirm

Damit erhält man:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{x_s}{u_{res}} \\ \frac{y_s}{v_{res}} \end{pmatrix} + \begin{pmatrix} \frac{u_{\dim}}{2} \\ \frac{v_{\dim}}{2} \end{pmatrix} \tag{3.5}$$

$$= \frac{f_1}{f_2} \begin{pmatrix} \frac{x_v}{u_{res}} \\ \frac{z_v \cdot \sin\alpha - d_y \cdot \cos\alpha}{v_{res}} \end{pmatrix} + \begin{pmatrix} \frac{u_{\dim}}{2} \\ \frac{v_{\dim}}{2} \end{pmatrix}$$

Mit: u und v in Pixel; x_w, z_v und d_y in Meter,
u_{res} und v_{res} in Meter/Pixel

Formel (3.5) kann in die Komponenten u und v aufgeteilt werden:

$$u = \frac{f_1}{f_2} \frac{x_v}{u_{res}} + \frac{u_{\dim}}{2} \tag{3.6}$$

$$v = \frac{f_1}{f_2} \frac{z_v \cdot \sin\alpha - d_y \cdot \cos\alpha}{v_{res}} + \frac{v_{\dim}}{2} \tag{3.7}$$

Die in den Formeln (3.6) und (3.7) dargestellten Beziehungen zwischen einem von der Kamera erfassten Objekt und dessen Abbildung auf dem Bildschirm werden später bei der Interpretation der Bildinformationen weiter verwendet.

3.2 Verkettungsverfahren

Die für die vorliegende Arbeit wichtigste Information aus den Bildfolgen ist der Verlauf der Fahrbahn, der durch die Fahrbahnmarkierungen gekennzeichnet ist. Um die Fahrbahnmarkierungen zu lokalisieren wäre ein klassischer Ansatz, eingehende Bilder auf horizontale Farbunterschiede zu untersuchen, da von den Fahrbahnmarkierungen erwartet wird, dass sich diese von der Fahrbahn farblich deutlich abheben. Da aber auch andere Objekte der Grund für deutliche Farbunterschiede sein können, ist ein deutlicher Farbunterschied nicht notwendigerweise eine Fahrbahnmarkierung. In Konsequenz: ein rein horizontales Scannen nach Farbunterschieden ist für eine qualitativ hochwertige Suche nach Fahrbahnmarkierungen nicht ausreichend.

Deshalb nutzt das System eine Datenbasis mit statistischen Informationen über den Verlauf von Fahrbahnmarkierungen. Es wird die Tatsache genutzt, dass von Fahrbahnmarkierungen ein gewisser Verlauf im Bild erwartet wird (z.B. im Allgemeinen auf einen Fluchpunkt am Horizont zulaufend).

Diese Information ist von der Position im Bild abhängig – auch gibt es Bildbereiche, in denen zu keiner Zeit eine Fahrbahnmarkierung erwartet wird. In Konsequenz müssen die statistischen Informationen dahingehend unterschieden werden, auf welchen Bildbereich sie sich beziehen. Dazu wird der zu untersuchende Teil des Bildes in Teilbereiche quadratischer Form unterteilt und es werden für jeden Teilbereich unterschiedliche statistische Informationen gebildet.

Diese statistischen Informationen werden dabei einmalig aufgrund von Analysen von Bildfolgen erstellt. Im Detail geben sie die jeweils bedingte Wahrscheinlichkeit darüber an, ob zwei Punkte, die aufgrund der horizontalen Suche nach Farbunterschieden als Kandidaten für eine Fahrbahnmarkierung identifiziert wurden, Teil einer Fahrbahnmarkierung sind. Solche Bildpunkte, d.h. die Kandidaten für Fahrbahnmarkierungen, müssen im Umkehrschluss in eine vorgegebene Richtung verkettbar sein.

Die Auswertung der statistischen Informationen, d.h. die Info, in welche Richtung Fahrbahnmarkierungspunkte innerhalb eines Teilbereichs verkettbar sind, werden als einmalig als *Verkettungsvektoren* gespeichert und später vom Lernsystem dazu genutzt, *Verkettungen,* d.h. kettenartige Verbindung von Markierungspunkten, bzw. Pixel, zu bilden. Mit Hilfe dieser Verkettungen werden drei entscheidende Ziele erreicht.

Zum einen kann aufgrund einer Verkettungsmöglichkeit eine Aussage abgeleitet werden, ob die zur Verkettung verwendeten Bildpunkte mit hoher Wahrscheinlichkeit zu einer Fahrbahnmarkierung gehören oder nur aufgrund von Farbschwankungen oder anderer im Bild befindlicher Objekte stammen.

Zum anderen lässt sich durch eine gerichtete Suche nach Fahrbahnmarkierungen der Suchaufwand minimieren und es wird ein Zeitvorteil beim Ablauf des Algorithmus erzielt.

Als dritter Vorteil: es werden unsinnige Verkettungen (z.B. zickzackförmig oder kreisförmig) vermieden.

Die Details zum Verkettungsverfahren werden in Kapitel 5.1.1.3 dargestellt.

4 VERSTÄRKUNGSLERNEN

Wie bereits eingangs in Kapitel 1.3 dargestellt, liegt der Fokus der vorliegenden Arbeit auf der Lernfähigkeit eines Gesamtsystems. Eine der bedeutendsten Anforderung lautet dabei, dass das System den Verhaltensraum (d.h. gemäß Kapitel 1.4 den virtuellen Raum, der sich über Situationsbeschreibungen, Aktionen und Situationsbewertungen aufspannt) autonom erforscht. Bei vielen anderen Arbeiten auf dem Gebiet der autonomen Fahrzeugsteuerung wird Wissen über Fahrzeug, Umgebung oder erwartetes Fahrverhalten dem System explizit vorgegeben (Modellierung) oder in Form von Trainingssequenzen vermittelt. In diesen Fällen wird von instruierendem Lernen (instructive learning) gesprochen. Ein System auf Basis instruierenden Lernens kann beliebig komplex werden, da die Instruktionen letztlich Handlungsanweisungen vorgeben, die vom Lernverfahren verstanden werden sollten. In vereinfachter Form werden die Instruktionen (d.h. die Handlungsanweisungen) nicht ausgewertet und interpretiert sondern lediglich gespeichert – in Folgesituationen werden dann die gespeicherten Handlungsanweisungen einer der aktuellen Situation ähnlichen Situation ausgegeben. In diesem Fall ist der Anspruch an die Lernfähigkeit des Systems begrenzt. Im idealen Fall hingegen muss das lernende System die Instruktionen verstehen, d.h. eine Interpretation der Handlungsanweisung durchführen.

In der vorliegenden Arbeit soll das System das Wissen über Fahrverhalten des Fahrzeugs und Interaktion mit der Umgebung autonom erforschen – d.h. in Bezug auf Aktionen und deren Auswirkung auf Fahrzeug und Umgebung analysieren. Dies wird auch als auswertendes Lernen (evaluative learning) bezeichnet, da die einzige Information, die das System zur Optimierung des Fahrverhaltens erhält, aus den Sensordaten der Umgebung stammt und diese entsprechend ausgewertet werden. In diesem Zusammenhang müssen zeitliche Verzögerungen bei Erhalt oder Auswertung der Sensordaten sowie mögliches Rauschen berücksichtigt werden, d.h. das System muss diesbezüglich entsprechend robust sein. Auswertendes Lernen ist ein evolutionstechnisch sehr altes Lernverfahren, da die meisten Lernvorgänge bei Organismen auf diesem Verfahren aufsetzen.

Diese Anforderungen und Überlegungen führten zur Auswahl des Verstärkungslernens als grundlegendes Lernverfahren und sind die Basis des in dieser Arbeit implementierten und untersuchten lernenden Systems. Bevor dieses in Bezug auf die vorliegende Aufgabenstellung im Detail angepasst und erforscht wird, wird die generelle Verfahrensweise von Verstärkungslernen dargestellt und es werden zusätzlich zu Kapitel 1.4 weitere Begriffsdefinitionen festgelegt.

4.1 Verfahrensweise und weitere Begriffsdefinitionen

Eine umfangreiche Darstellung des Verstärkungslernens wurde 1998 von Richard Sutton und Andrew G. Barto veröffentlicht [Sutton & Barto 98].

Dabei basiert die Architektur eines lernenden Systems auf Basis von Verstärkungslernen, wie in Abbildung 4.1 dargestellt, auf einem *Agenten* (der zu einem Zeitpunkt t die Information über einen Zustand s_t als auch einen *Belohnungswert* r_t erhält) sowie der *Umgebung* (environment)

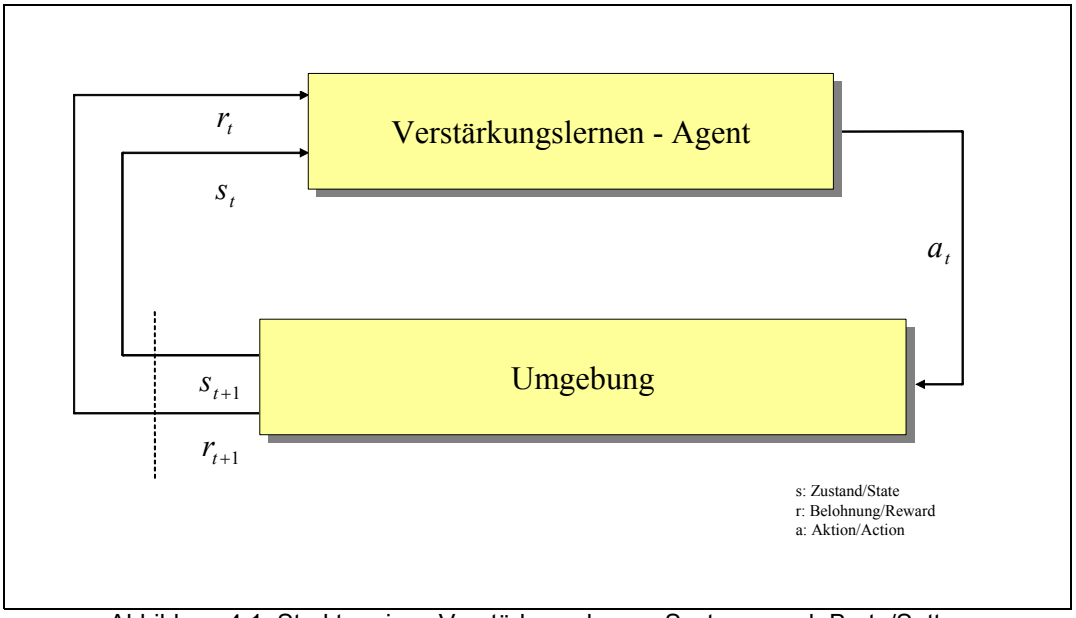


Abbildung 4.1: Struktur eines Verstärkungslernen-Systems nach Barto/Sutton

Der *Zustand* s_t entspricht einer Situationsbeschreibung (eine auf Parameter reduzierte Beschreibung der Situation) zum Zeitpunkt t.

Bisher wurde der Verhaltensraum so definiert, dass dies ein virtueller Raum ist, der sich aufgrund von **Situationsbeschreibungen**, Aktionen und Situationsbewertungen aufspannt. Da beim Verstärkungslernen die Zustände die Situationsbeschreibungen ersetzen, wird die weitere Verwendung des Verhaltensraums als virtueller Raum verstanden, der sich aufgrund von **Zuständen**, Aktionen und Situationsbewertungen aufspannt.

Zeitgleich wird dem Agenten ein Belohnungswert r_t (reward) zugeführt, der eine Bewertung der aktuellen Situation in Bezug auf das zu lernende Ziel abgibt. Soll z.B. gelernt werden, ein Fahrzeug entlang einer Strecke zu fahren, könnte die Bewertung umgekehrt proportional zur Abweichung von der Fahrbahnmitte sein. Im Folgenden wird dieser Wert als *Belohnung* bezeichnet, wobei ein numerisch niedriger, bzw. negativer Wert einer Bestrafung gleichkommt.

Die Belohnungswerte werden in eine Situationsbewertung überführt. Eine Situationsbewertung ist die Summe kumulierter Belohnungen einer situationsspezifischen Aktion wobei zukünftige Belohnungen abgeschwächt berücksichtigt werden (d.h. umso später eine Belohnung eintritt, desto geringer geht diese in die Bewertung einer aktuellen situationsspezifischen Aktion ein).

Basierend auf diesen Eingangsgrößen wird vom Agenten eine Aktion a_t (action) ermittelt und ausgegeben, die im Einsatzbereich für Fahrzeugsteuerungen einem Steuerkommando der Lenkung entspricht. Die Aktion wirkt auf die Umgebung und bewirkt dort eine Situationsänderung. Zum Zeitpunkt $t+1$ wird erneut der Zustand (auf Basis der Sensordaten) ermittelt und zusammen mit dem Belohnungswert - d.h. s_{t+1} sowie r_{t+1} - dem Agenten zugeführt.

Die Ermittlung der Aktion wird von der *Strategiefunktion* (policy) durchgeführt, die die Regeln zur Auswahl der nächsten Handlung aus der Menge der Handlungsalternativen zum Zeitpunkt t vorgibt. Üblicherweise wird dabei die Aktion mit der höchsten Situationsbewertung aus der Menge der möglichen Aktionen der aktuellen Situation ausgewählt.

Im Bezug auf die Aktionsselektion wird in zwei wesentliche Vorgehensweisen unterschieden: Zum einen kann ein Verstärkungslernen-System das situationsspezifische Verhalten weiter optimieren, indem auch bisher unbekannte oder nicht-optimale Aktionen (d.h. Aktionen mit relativ niedriger Situationsbewertung) ausgeübt werden, um diese im Anschluss einer Bewertung zu unterziehen. Dieser Moduls wird *Erkundung* (exploration mode) genannt und ermöglicht die eigenständige Erkundung des Verhaltensraumes. Im Gegensatz dazu kann ein Verstärkungslernen-System den Grad der Erkundung des Verhaltensraumes einstellen und lediglich die Aktion mit der aktuell höchsten Situationsbewertung ausgeben. Dieser Modus wird *Datennutzung* (exploitation mode) genannt.

In diesem Zusammenspiel wird die Bedeutung der Belohnung r_t deutlich: über die Differenz, bzw. Entwicklung der Belohnungswerte erhält der Agent eine Rückmeldung über die Angemessenheit der einzelnen situationsspezifischen Aktionen, bzw. Aktionssequenzen.

Im Detail werden diese Belohnungen dazu verwendet, die Situationsbewertungen (die Bewertung von situationsspezifischen Aktionen) zu aktualisieren und nach einer genügend oft durchgeführten *Aktualisierung* eine Konvergenz der Situationsbewertungen im Verhaltensraum anzustreben. Sofern die Umgebung in irgendeiner Form gesetzmäßig auf die Aktionen des Verstärkungslernen-Systems reagiert, kann dieses Ziel erreicht werden. Dazu wird nach jedem Erhalt einer Belohnung eine *Aktualisierungsfunktion* (update function) aufgerufen, die eine zentrale Bedeutung im Rahmen der Lernfähigkeit wahrnimmt.

Aus Sicht des Agenten ist es zwar legitim, die Informationen über den Zustand und die Belohnung in aufbereiteter Form zu erwarten – diese Art von Information kann aber nicht direkt in dieser Form von einem Sensor eines

Fahrzeugs oder einem entsprechenden Simulator zur Verfügung gestellt werden. Aus diesem Grund muss die Umgebung weiter aufgeteilt und um eine Vorverarbeitung der Sensordaten erweitert werden.

In der vorliegenden Arbeit wird zunächst ein Fahrzeugsimulator eingesetzt, der die Steuerkommandos des lernenden Systems entgegennimmt und die Umgebungsauswirkungen simuliert. Der Simulator erzeugt daraufhin eine visuelle Darstellung der neuen Situation. Diese wird von einer Bildverarbeitung aufgenommen und in eine Situationsbeschreibung umgewandelt, die wiederum in einer Datenbank abgelegt wird. Mit Hilfe von Ähnlichkeitsuntersuchungen können ähnliche Situationen zusammengefasst und als übergreifende Situationsbeschreibung abgelegt werden.

In Bezug auf den Agenten kommen dabei unterschiedliche Implementierungsvariationen von Verstärkungslernen in Betracht.

Verstärkungslernen auf Basis des Monte-Carlo-Prinzips führt die Lernschritte **in Episoden** durch – dabei ist das Ende einer Episode durch das Erreichen eines finalen Lernziels bestimmt. Die Lernvorgänge, d.h. beim Verstärkungslernen die Aktualisierungen der Situationsbewertungen, finden jeweils nur am Ende einer Episode und nicht kontinuierlich, d.h. nicht nach jedem Erhalt einer Belohnung, statt. Das Monte-Carlo-Prinzip erfordert, dass ein finales Lernziel eindeutig formulierbar ist und mit hoher Häufigkeit erreicht wird - für die vorliegende Arbeit ist diese Voraussetzung nicht erfüllt. Viele Lernziele, wie auch das der vorliegenden Arbeit, sind komplexer und können erst nach einer hohen Anzahl von Aktionsausübungen erreicht werden. Je mehr Aktionen während einer Episode ausgeübt werden und je mehr Situationen durchlaufen werden, desto schwerer die spätere Zuordnung der Aktualisierungen zu den Situationsbewertungen. Auch ist es denkbar, dass in einer komplexen Umwelt mehrere optimale Zustände vorliegen, so dass ein eindeutiger Zielzustand nicht vorgegeben werden kann. Für diese Aufgabenstellungen stößt das Monte-Carlo-Prinzip an seine Grenzen.

Bei der Variante des Temporal Difference Lernens (TD-Learning) des Verstärkungslernens werden diese Grenzen dadurch überwunden, dass die Aktualisierungen der Situationsbewertungen **jeweils unmittelbar nach Erhalt einer Belohnung** erfolgen und somit unabhängig von einem finalen Ziel-Zustand sind. Dabei wurden in diesem Umfeld zwei unterschiedliche Varianten entwickelt: Sarsa und Q-Lernen (Q-Learning). Beide Verfahren ermitteln auf Basis der aktuellen Situationsbewertungen eine bestmögliche Aktion. Beim Sarsa ist diese Aktion zwingend auszuüben – eine Abweichung von der durch den Sarsa-Algorithmus ermittelten Aktion ist nicht zulässig. Beim Q-Lernen besteht diese Vorgabe nicht und die ermittelte Aktion wird lediglich als Aktionsempfehlung verstanden. Deshalb ist es beim Q-Lernen möglich, beliebige Aktionen auszugeben, um die Reaktion der Umgebung bewusst zu testen – eine Eigenschaft, die in dieser Arbeit explizit genutzt werden soll. Aus diesem Grund wird in der vorliegenden Arbeit das Lernsystem auf Basis des Q-Lernens aufgebaut und beschrieben.

4.2 Mathematischer Hintergrund des Q-Lernens

In diesem Unterkapitel wird der mathematische Hintergrund des in dieser Arbeit verwendeten Lernsystems (d.h. lernendes System auf Basis von Q-Lernen) beschrieben und ergänzend hergeleitet. Zunächst werden die bisherigen Begriffsdefinitionen weiter ergänzt.

4.2.1 Zustände, Aktionen, Belohnungen und Situationsbewertungen

In Abbildung 4.2 sind die wesentlichen Elemente des Lernsystems dargestellt. Zunächst verfügt ein System auf Basis von Verstärkungslernen über eine beliebige Anzahl von Zuständen (states) s_i. Diese Zustände entsprechen den eingangs erwähnten Situationsbeschreibungen die über die Sensordaten von der Situation abgeleitet werden, in der sich ein Fahrzeug befindet. In der vorliegenden Arbeit werden die Zustände dahingehend unterschieden, auf welchem Streckentyp sich das Fahrzeug befindet (Geraden- oder Kurventyp) sowie in welcher Position sich das Fahrzeug befindet (Seitenablage, Fahrwinkel).

Die Zustände werden im Speicher des System-PC's gespeichert und dabei in Bezug auf den Index durchnumeriert. Die Speicherreihenfolge der Zustände ist dabei willkürlich, d.h. zwei in Bezug auf den Index benachbarte Zustände entsprechen möglicherweise zwei komplett unterschiedlichen Situationen. Der Zustand wird dabei durch die Situationsbeschreibung beschrieben, die im Wesentlichen die Nachbildung der Fahrbahnmarkierungen enthält. Diese Situationsbeschreibungen sind im Speicher abgelegt und können über den Index des Zustandes angesprochen werden.

Sofern vor der Laufzeit des Systems die Menge der möglichen Situationen bekannt ist, können die Zustände in geordneter Form angelegt werden. Insbesondere in komplexen Umgebungen ist dies aber oft nicht möglich (die mögliche Anzahl von Situationen ist beliebig groß). In diesen Fällen werden Zustände erst während der Laufzeit des Lernsystems angelegt – nämlich jeweils erst dann, wenn erstmals eine neue Situation auftritt. Dies führt zu einer eigenständigen Anpassung des Zustandsraums an die Komplexität der Umgebung.

In jedem Zustand wird von der Umgebung eine zustandspezifische Belohnung r_i erzeugt und dem Agenten zugeführt. Im Folgenden wird grundsätzlich davon ausgegangen, dass diese Belohnung in hohem Maß deterministisch ist, d.h. für jeden Zustand eine jeweils ähnliche Belohnung ermittelt wird. Diese Belohnungen werden von der Umgebung berechnet. In der vorliegenden Arbeit ist die Zielvorgabe, das Fahrzeug möglichst nahe zu einer vorgegebenen Linie zu führen (path tracking). Aus diesem Grunde entsprechen die Bewertungen der Seitenablage, d.h. geben Rückmeldung über die Entfernung des Fahrzeugs von der Fahrbahnmitte. Eine andere Bewertungsfunktion wäre denkbar (z.B. eine explizite Forderung nach Minimierung der Schwingungs-

frequenz um die Fahrbahmitte), wurde in dieser Arbeit aber nicht gewählt, da eine Varianz der Bewertungsfunktion keinen Einfluss auf die Art sondern lediglich auf das Resultat des Lernprozesses hätte.

Der Übergang von einem Zustand in einen anderen wird durch die Ausübung von Aktionen $a_{i \to j}$ veranlasst, wobei $a_{i \to j}$ den Zustand s_i in den Zustand s_j überführt. Eine Aktion $a_{i \to i+2}$ bezeichnet eine Aktion die einen Zustandswechsel vom Zustand s_i auf den Zustand s_{i+2} bewirkt.

Selbstverständlich ist nicht jeder beliebige Zustandswechsel möglich, d.h. ein gewünschter Zustand kann oft nicht direkt (mit nur einer einzigen Aktion) erreicht werden, sondern nur indirekt über den Umweg weiterer Zustände.

Die Menge aller Zustände ist $s=\{s_i \mid s_i \text{ ist ein Zustand mit } 0 < i < n\text{-}1\}$ mit n = Anzahl der möglichen Zustände.

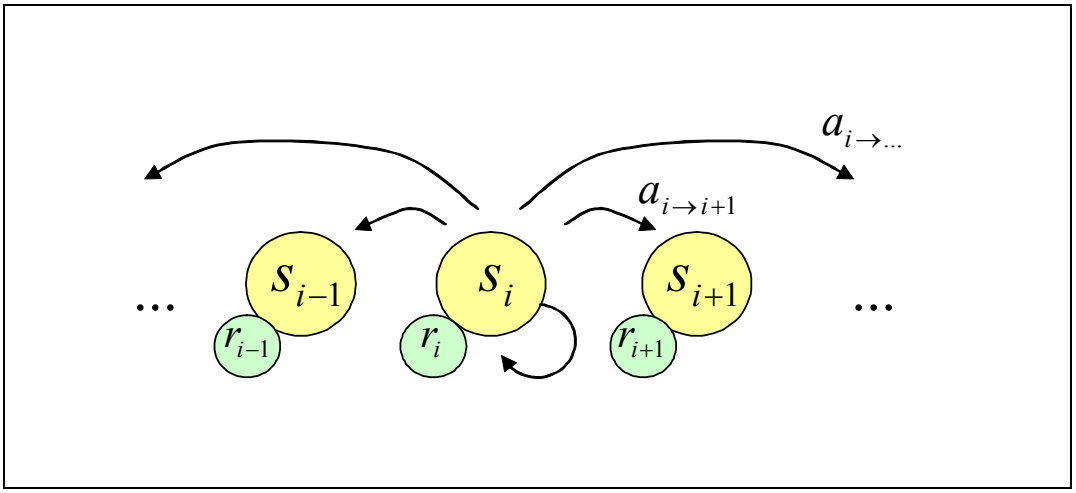


Abbildung 4.2: Zustandsbezogene Darstellung von Zuständen, Belohnungen und Aktionen

In Abbildung 4.3 werden ebenfalls Zustände, Belohnungen und Aktionen dargestellt, diesmal allerdings in einer zeitlichen Abhängigkeit. Diese Darstellung entspricht der Tatsache, dass zu jedem Zeitintervall während der Laufzeit des Lernsystems eine Aktion ausgeben wird, aufgrund dessen der Zustand im nächsten Zeitintervall bestimmt wird.

Die Beziehungen zwischen zustandsbezogener Darstellung und zeitbezogener Darstellung, bzw. die Kombination beider Darstellungen, werden später wesentlicher Bestandteil der Konvergenzuntersuchungen.

Basierend auf den Belohnungen wird vom Verstärkungslernen jeder situationsspezifischen Aktion eine Situationsbewertung Q zugeordnet. Da im Weiteren die Situationsbewertungen zur Auswahl der Aktion dienen soll, muss jeder Handlungsalternative $a_{i \to j}$ in einer Situation s_i eine Bewertung $Q(s_i, a_{i \to j})$ zugeordnet werden. Deshalb wird bei Zustandsbezug die Darstellung $Q(s_i, a_{j \to j})$, bzw. bei Zeitbezug die Darstellung $Q(s_t, a_t)$ verwendet.

Wenn bezüglich Verhaltensraum das Thema Konvergenz diskutiert wird, ist jeweils die Konvergenz der Situationsbewertungen $Q(s, a)$ gemeint. Da diese Situationsbewertungen auch als Teile der Q-Funktion im Verhaltensraum bezeichnet werden, kann in Bezug auf die Konvergenz auch von der Konvergenz der Q-Funktion im Verhaltensraum gesprochen werden.

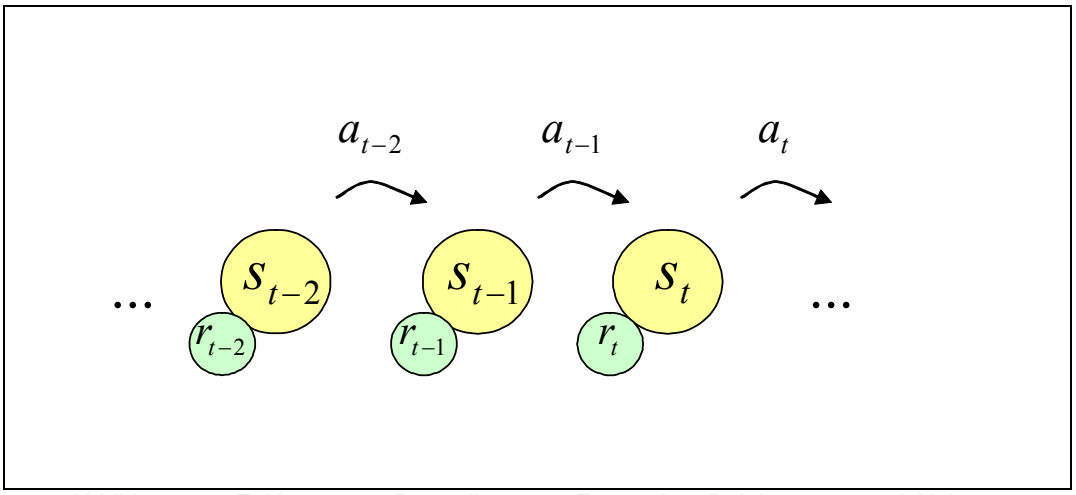


Abbildung 4.3: Zeitbezogene Darstellung von Zuständen, Belohnungen und Aktionen

Der Zustandsbezug der Situationsbewertungen wird bei Betrachtungen des Zustandsraums verwendet – wie z.B. bei Analyse eines erreichten Status. In diesem Fall ist der Bezug auf einen Zustand, bzw. dessen Aktionen wichtig und die Information, wann dieser Zustand wie oft angesprochen wurde oder wann die entsprechende Aktion ausgegeben wurde ist nicht relevant. Der Zeitbezug der Situationsbewertungen wird immer dann verwendet, wenn eine Zustands- oder Aktionsreihenfolge untersucht wird – z.B. bezüglich der Dynamik einer Testreihe. Im Folgenden wird die Darstellungsweise je nach Aufgabenstellung der Analyse unterschieden.

Beim Verstärkungslernen gemäß Q-Lernen wird die Situationsbewertung $Q(s_t, a_t)$ als Summe aller zukünftigen Belohnungen berechnet. Dabei werden zukünftige Belohnungen umso stärker abgeschwächt, je weiter sie in der Zukunft liegen. Dies wird mit Multiplikation mit einem Faktor erreicht, der im Folgenden als *Diskontierungsfaktor* (discount factor) γ bezeichnet wird $(0 \leq \gamma \leq 1)$. In einem Verhaltensraum, dessen Situationsbewertungen nicht vollständig konvergiert sind, weicht die aktuelle Situationsbewertung von der zumindest theoretisch erreichbaren konvergierten Situationsbewertung um einen Fehlerwert ab, der im Folgenden als *Fehlerwert* $TDerr$ bezeichnet wird. Diese Beziehung ist in Formel (4.1) darstellt.

$$Q(s_t, a_t) = r_{t+1} + \sum_{i=1}^{\infty} \gamma^i r_{t+i+1} - TDerr \qquad (4.1)$$

4.2.2 Aktualisierung

Das zentrales Thema im Umfeld Verstärkungslernen ist die Konvergenz der Situationsbewertungen $Q(s,a)$. In Formel (4.1). wird diese zunächst in Abhängigkeit aller zukünftigen Belohnungen berechnet und beinhaltet einen unbekannten Fehlerwert $TDerr$. Diese Formel kann wie folgt umgestellt und nach $TDerr$ aufgelöst werden:

$$Q(s_t, a_t) = r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - TDerr \tag{4.2}$$

$$TDerr = r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{4.3}$$

Der über Formel (4.3) berechnete Fehlerwert wird zum Zeitpunkt $t+1$ rückwirkend berechnet und zur nachträglichen Aktualisierung der Situationsbewertung zum Zeitpunkt t verwendet. Sofern aber $Q(s_{t+1},a_{t+t})$ noch nicht konvergiert, d.h. fehlerbehaftet ist, ist in Konsequenz auch $TDerr$ noch fehlerbehaftet. Deshalb wird zur Aktualisierung der Situationsbewertungen der Fehlerwert $TDerr$ mit einem weiteren Faktor, der *Lernrate* (learning rate) α (mit $0 < \alpha \leq 1$), gedämpft.

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \cdot TDerr \tag{4.4}$$

Diese Vorgehensweise ermöglicht eine schrittweise Konvergenz; setzt aber die geeignete Parametrisierung des Lernsystems voraus.

Dabei erreicht man eine schnellere Konvergenz, wenn zum Zeitpunkt $t+1$ nicht nur die Situationsbewertung $Q(s_t,a_t)$, sondern auch alle zeitlich davor liegenden Situationsbewertungen aktualisiert werden, d.h. $Q(s_{t-1},a_{t-1})$, $Q(s_{t-2},a_{t-2})$, und so weiter. Je weiter eine Situation, bzw. deren ausgeführte Aktion zeitlich zurückliegt, desto geringer ist der Einfluss der Aktualisierung auf die jeweilige Situationsbewertung. Deshalb wird beim rückwärtigen Aktualisieren der Fehlerwert $TDerr$ um einen weiteren Faktor, dem *Verfallsparameter* (trace decay parameter) λ ($0 \leq \lambda \leq 1$), gedämpft:

$$\begin{aligned} &Q(s_{t-n}, a_{t-n}) := Q(s_{t-n}, a_{t-n}) + \alpha \cdot \lambda^n \cdot TDerr \\ &n \in \{1,2,...,\infty\} \end{aligned} \tag{4.5}$$

Spätere Tests zeigen, dass das rückwärtige Aktualisieren dazu führt, dass die gleiche Konvergenz in einer kürzeren Zeit erreicht wird – verliert dafür aber an Robustheit in Bezug auf gestörte oder zeitlich verzögerte Belohnungen.

4.2.3 Q_{max}

Da in Formel (4.1) der Wert der Situationsbewertung $Q(s,a)$ als Summe aller zukünftigen Belohnungen definiert wird, gibt der Wert der Situationsbewertung

Auskunft darüber, in wie weit eine situationsspezifische Aktion zum Erreichen einer Situation mit dauerhaft maximaler Belohnung beiträgt.

In diesem Zusammenhang werden zwei Definitionen vereinbart.

$$Q_{\mathrm{a-max}} = \max_a Q(s_i, a) \tag{4.6}$$

$$Q_{\max} = \max_s Q_{\mathrm{a-max}} \tag{4.7}$$

Formel (4.6) definiert die höchste Situationsbewertung über alle Aktionen einer Situation; Formel (4.7) definiert die höchste Situationsbewertung überhaupt (d.h. über alle Situationen und alle Aktionen).

4.2.4 Strategiefunktion

Übergeordnetes Ziel eines Verstärkungslernen-Systems ist das Erreichen einer Situation, in der das System dauerhaft die höchstmöglichen Belohnungen erhält.

In einem konvergierten System mit einem einzigen Q_{max} und keinen lokalen Maxima müsste dazu lediglich in jeder Situation diejenige Aktion ausgeführt werden, deren Situationsbewertung als $Q_{a\text{-}max}$ identifiziert werden kann.

In diesem Fall vertraut das System den Situationsbewertungen, d.h. dem dadurch aufgebauten Wissen über die Angemessenheit der Aktionen innerhalb einer Situation. Aus diesem Grunde bezeichnet man diese Vorgehensweise als Datennutzung (exploitation) und die Beschreibung der Strategiefunktion (policy) ergibt sich wie folgt:

$$\pi(s_i) = a \;\; mit \; Q(s_i, a) = Q_{a-\max} \tag{4.8}$$

Wie in späteren Tests nachgewiesen wird, kann ein Lernsystem im Datennutzungsmodus sehr wohl zu einer stabilen Konvergenz einiger Situationsbewertungen führen. Dabei läuft das Lernsystem Gefahr, dass es sich auf eine geringe Anzahl von Aktionen und somit eine geringe Anzahl von alternativ erreichbaren Situationen beschränkt und dadurch in ein sub-optimales Verhalten verfällt. Dies gilt insbesondere dann, wenn noch keine Konvergenz der Situationsbewertungen über einen größeren Bereich des Verhaltensraums eingetreten ist.

Um den Verhaltensraum so weit wie möglich zu erkunden, wird die Strategiefunktion wie folgt verändert:

$$\pi\ (s_i) = a \begin{cases} mit\ \ Q(s_i, a) = Q_{a-\max}\ \ \text{wenn}\ \ rand() \geq \varepsilon \\ mit\ \ a = a_{random}\ \ wenn\ rand() < \varepsilon \end{cases} \tag{4.9}$$

Diese Variante der Strategiefunktion wird Erkundung genannt, da über einen Zufallsgenerator das explizite Erkunden aller Aktionen erzwungen wird. Wie ausgeprägt der Erkundungsmodus eingesetzt werden kann, hängt im Wesentlichen von der Testumgebung ab, d.h. wie weit das bewusste Ausführen und Ausprobieren von nicht-optimalen Aktionen akzeptiert werden kann. Bei Simulationen sind üblicherweise keine Grenzen gesetzt; in der Praxis gibt es oft Grenzen in Bezug auf Materialverschleiß aber auch Gefahr- und Sicherheitsaspekte für Personen. Der Grad der Erkundung wird dabei über den *Erkundungsparameter* ε (mit $0 \leq \varepsilon \leq 1$) gesteuert.

4.2.5 Konvergenz

Zentraler Fokus im Umfeld von Systemen basierend auf Verstärkungslernen ist das Erzielen einer Konvergenz der Situationsbewertungen im Verhaltensraums, d.h. die Konvergenz aller Situationsbewertungen $Q(s,a)$.

Deshalb wird im Folgenden zunächst berechnet, welche unterschiedlichen Werte die Situationsbewertungen von Aktionen und Situationsbeschreibungen annehmen. Diese ausführlichen Untersuchungen der Konvergenz der Situationsbewertungen sind die Basis für die nachfolgenden Experimente und wurden in dieser Art erstmalig in der vorliegenden Arbeit hergeleitet.

Dabei ist eine Situationsbewertung genau dann vollständig konvergiert, wenn der Fehlerwert $TDerr$ gleich Null ist, d.h. $\lim_{t\to\infty} TDerr \to 0$.

In diesem Fall wird die konvergierte Situationsbewertung im folgenden als $Q^*(s,a)$ bezeichnet.

$$Q^*(s_t, a_t) = Q(s_t, a_t)\ \text{wenn}\ TDerr = 0 \tag{4.10}$$

In der Praxis wird eine vollständige Konvergenz infolge gestörter Belohnungen oder Variationen bei der Zustandsbestimmung nicht eintreten. Diese Effekte werden später im Rahmen der experimentellen Ergebnisse diskutiert.

Im Folgenden werden zunächst die zu erwartenden konvergierten Situationsbewertungen ermittelt. Dabei wird die Konvergenz von Q_{max}, bzw. $Q_{a\text{-}max}$ unter-

sucht; im Anschluss daran aber auch die Situationsbewertungen der alternativen Aktionen.

4.2.5.1 $Q^*(s_i, a_{i\to i})$

Im ersten Fall wird hergeleitet, auf welchen Wert eine Situationsbewertung konvergiert, dessen Zustand fortwährend die gleiche Aktion ausführt und damit auch im gleichen Zustand bleibt.

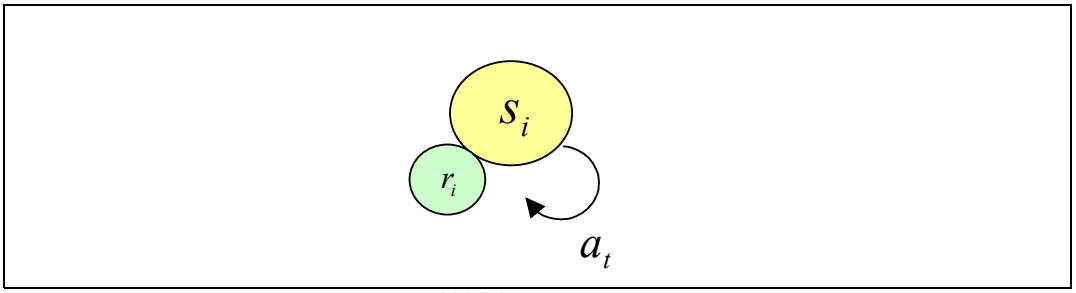


Abbildung 4.4: $a_{i\to i}$

Es wird vorausgesetzt:

$$s_{t+1} = s_t = s_i$$

$$a_{t+1} = a_t = a_{i\to i}; r_{t+1} = r_t = r_i$$

Da die konvergierte Situationsbewertung berechnet werden soll, wird $TDerr$ zu Null gesetzt und nach der konvergierten Situationsbewertung aufgelöst:

$$TDerr = r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) = 0 \tag{4.11}$$

$$r_i + \gamma \cdot Q^*(s_i, a_{i\to i}) - Q^*(s_i, a_{i\to i}) = 0 \tag{4.12}$$

$$Q^*(s_i, a_{i\to i}) = \frac{r_i}{1-\gamma} \tag{4.13}$$

Ein solch stationärer Fall kann nur dann auftreten, wenn die Situationsbewertung $Q(s_i)$ einem lokalen Maximum entspricht, d.h. größer ist als alle anderen $Q(s_j)$ für $j \neq i$. Demnach gilt:

$$\frac{r_i}{1-\gamma} = Q_{a-\max} \text{ von } s_i \tag{4.14}$$

Für den Fall, dass die Belohnung in Zustand s_i größer oder zumindest gleich allen anderen Belohnungen der anderen Zustände ist, entspricht das lokale Maximum der Situationsbewertung dem absoluten Maximum.

$$\text{wenn } r_i \geq r_j \quad \text{für alle } i \neq j$$

$$Q_{a-\max} \text{ von } s_i \geq Q_{a-\max} \text{ von } s_j$$

$$\begin{aligned} Q^*(s_i, a_{i\to i}) &= Q_{\max} \\ s_i &= s_{\max} \end{aligned} \tag{4.15}$$

4.2.5.2 $Q^*(s_i, a_{i\to \max})$

Im Folgenden wird die konvergierte Situationsbewertung eines Zustandes berechnet, dessen Belohnungswert nicht der maximalen Belohnung entspricht, einen solchen Zustand aber innerhalb einer Aktionsausführung erreicht und anschließend in diesem Zustand verbleibt.

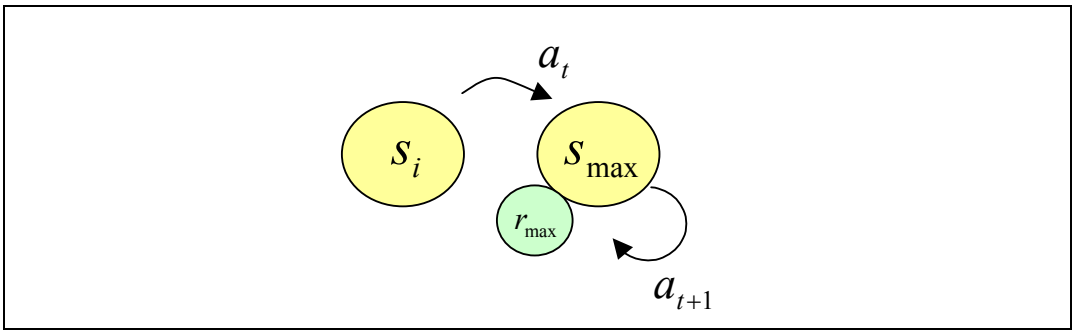


Abbildung 4.5: $a_{i\to \max}$

Es wird vorausgesetzt:

$$s_t = s_i; \quad s_{t+1} = s_{\max}$$

Aufgrund der Konvergenzanforderung kann *TDerr* wieder zu Null gesetzt werden. Nach Umformung der Gleichung und Auflösung nach der Situationsbewertung erhält man:

$$TDerr = r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) = 0 \tag{4.16}$$

$$r_{\max} + \gamma \cdot Q_{\max} - Q^*(s_i, a_{i\to\max}) = 0$$

$$\begin{aligned} Q^*(s_i, a_{i\to\max}) &= r_{\max} + \gamma \cdot Q_{\max} \\ &= r_{\max} + \gamma \cdot \frac{r_{\max}}{1-\gamma} = \frac{r_{\max}(1-\gamma) + \gamma \cdot r_{\max}}{1-\gamma} \\ &= \frac{r_{\max} - \gamma \cdot r_{\max} + \gamma \cdot r_{\max}}{1-\gamma} = \frac{r_{\max}}{1-\gamma} \end{aligned} \tag{4.17}$$

Dies entspricht der unter 4.2.5.1 hergeleiteten Konvergenz für $Q^*(s_i, a_{i\to i})$.

4.2.5.3 $Q^*(s_i,a_{i\to j})$

Im nächsten Schritt wird die konvergierte Situationsbewertung eines Zustandes betrachtet, der zwei Aktionsausübungen vom Zustand mit maximalem Belohnungswert entfernt ist.

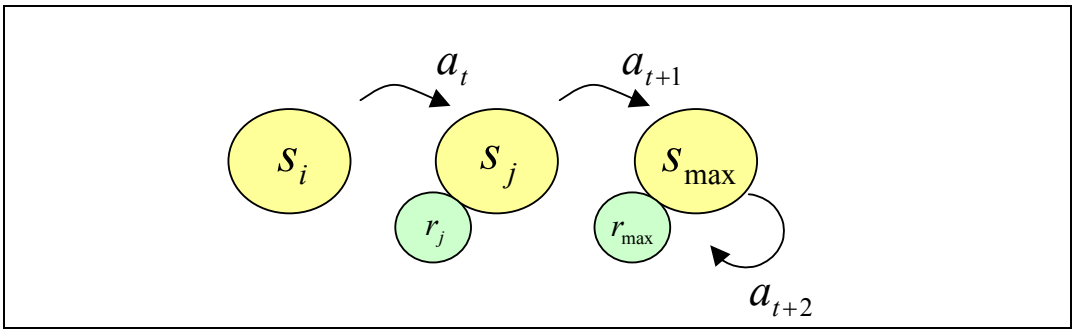


Abbildung 4.6: $a_{i\to j}$

Es wird vorausgesetzt:

$$s_t = s_i;\ \ s_{t+1} = s_j;\ \ s_{t+2} = s_{\max}$$

Wie bereits zuvor, kann aufgrund der Konvergenzanforderung $TDerr$ zu Null gesetzt und die Gleichung nach der konvergierten Situationsbewertung aufgelöst werden.

$$TDerr = r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) = 0 \tag{4.18}$$

$$r_j + \gamma \cdot Q_{a-\max(j)} - Q^*(s_i, a_{i\to j}) = 0$$

$$Q^*(s_i, a_{i\to j}) = r_j + \gamma \cdot Q_{a-\max}\ von\ s_j \tag{4.19}$$

Die konvergierte Situationsbewertung für die Situation $s = s_i$ entspricht der um den Faktor γ diskontierten maximalen Situationswertung der Situation $s = s_j$ plus der Belohnung der Situation $s = s_j$.

Wenn beachtet wird, dass gemäß Kapitel 4.2.5.2 die konvergierte Situationsbewertung $Q_{a\text{-}max(j)}$ gleich Q_{max} ist und die Differenz zwischen r_{max} und r_j als Parameter d definiert wird, erhält man:

$$\begin{aligned} Q^*(s_i, a_{i\to j}) &= r_{\max} - d + \gamma \cdot Q_{\max} \\ &= r_{\max} + \gamma \cdot Q_{\max} - d \end{aligned} \tag{4.20}$$

$$Q^*(s_i, a_{i\to j}) = Q_{\max} - d \tag{4.21}$$

4.2.5.4 $Q^*(s_t,a_{t\rightarrow t+1})$

Im Folgenden wird die zeitliche Konvergenzentwicklung der Situationsbewertungen untersucht. Unabhängig von den exakten Situationen, die bis zum Erreichen einer Situation mit maximaler Belohnung durchlaufen werden, können diese in einer zeitlichen Reihenfolge betrachtet werden. Dies wird in der folgenden Abbildung dargestellt:

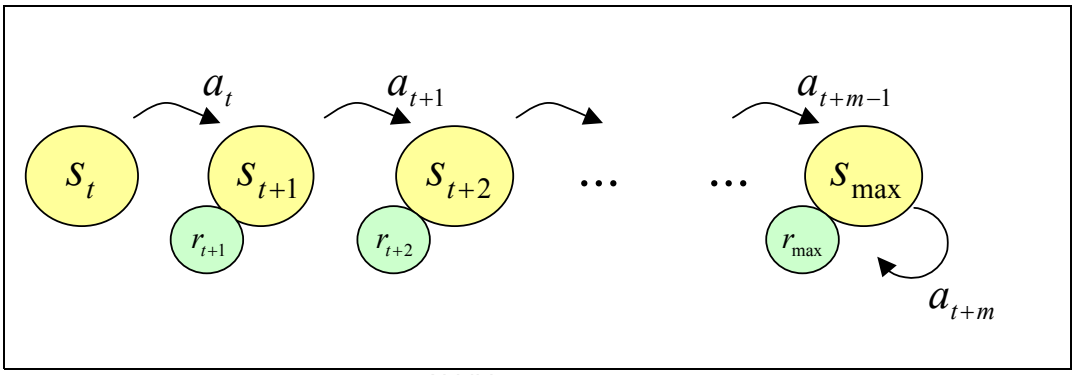


Abbildung 4.7: $a_{t\rightarrow t+1}$

Dabei ist der Parameter m gleich der Anzahl von Aktionen zwischen dem Anfangszustand $s = s_t$ und dem Erreichen des Zustandes $s = s_{max}$.

Es gilt:

$$Q^*(s_t, a_{t\rightarrow t+1}) = r_{t+1} + \gamma \cdot Q_{\max}{}^*(s_{t+1}, a) \tag{4.22}$$

$$= \gamma^0 \cdot r_{t+1} + \gamma^1 \cdot r_{t+2} + \ldots + \gamma^{m-1} \cdot Q_{\max}$$

$$= \sum_{v=0}^{m-2} \gamma^v \cdot r_{t+1+v} + \gamma^{m-1} \cdot Q_{\max}$$

Da alle Belohnungen r kleiner r_{max} sind, ist die konvergierte Situationsbewertung $Q^*(s_t,a_{t\rightarrow t+1})$ umso kleiner, je kleiner diese Belohnungen r sind. Dieser generelle Effekt wird durch den Diskontierungsfaktor γ in seiner Ausprägung entsprechend beeinflusst.

4.2.5.5 ΔQ^*, Δr

Zuletzt werden noch die Differenzen der konvergierten Situationsbewertungen betrachtet:

Es gilt die gleiche zeitliche Abfolge der Situationen wie in Kapitel 4.2.5.4., nur werden alle Werte relativ und nicht absolut betrachtet:

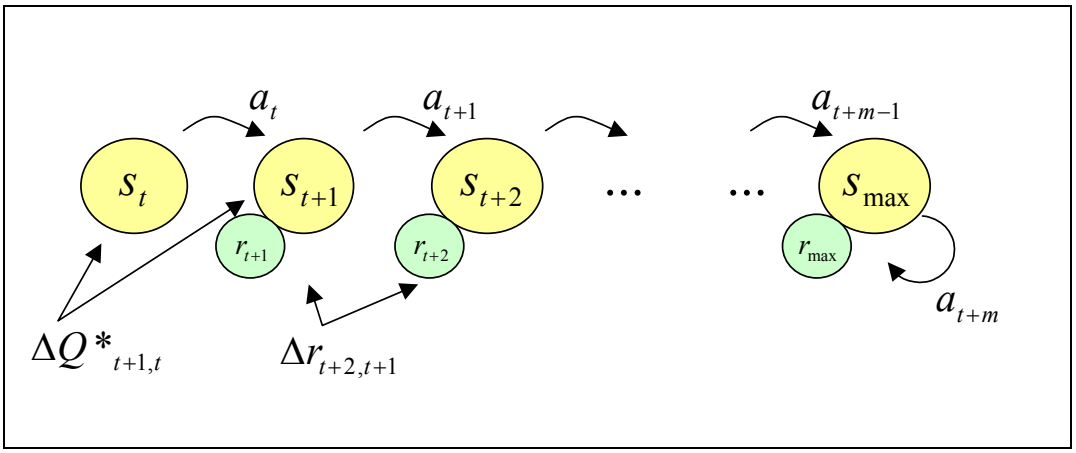


Abbildung 4.8: ΔQ*, Δ r

Die Differenz ΔQ^* zwischen den Situationen $s = t$ und $s = t+1$ kann als Differenz der Belohnungen der Zeitpunkte $t = t+2$ und $t = t+1$ sowie der abgewerteten Differenz zwischen den Situationen $s = s_{t+1}$ und $s = s_{t+2}$ hergeleitet werden:

$$\begin{aligned}\Delta Q^*_{t+1,t} &= Q^*(s_{t+1},a) - Q^*(s_t,a) \\ &= r_{t+2} + \gamma \cdot Q^*(s_{t+2},a) - (r_{t+1} + \gamma \cdot Q^*(s_{t+1},a)) \\ &= \Delta r_{t+2,t+1} + \gamma \cdot \Delta Q^*_{t+2,t+1}\end{aligned} \tag{4.23}$$

Nun sei der Parameter m wieder gleich der Anzahl der Aktionen zwischen dem Anfangszustand $s = s_t$ und dem Erreichen des Zustandes $s = s_{max}$. Damit erweitert sich die Formel zu:

$$\Delta Q^*_{t+1,t} = \sum_{v=0}^{m-2} \gamma^v \cdot \Delta r_{t+v+2,t+v+1} + \gamma^{m-1} \cdot \Delta Q^*_{t+m,t+m-1} \tag{4.24}$$

Sofern das Verstärkungslernen-System zum Zeitpunkt $t = t + m$ den Zustand s_{max} erreicht, gilt:

$$\Delta Q^*_{t+m,t+m-1} = \Delta Q^*_{\max,\to\max} = 0$$

Und damit:

$$\Delta Q^*_{t+1,t} = \sum_{v=0}^{m-2} \gamma^v \cdot \Delta r_{t+v+2,t+v+1} \tag{4.25}$$

Die Differenz der konvergierten Situationsbewertungen zweier zeitlich aufeinander folgenden Situationen ist demnach gleich der Summe aller diskontierten Belohnungsdifferenzen bis zum Erreichen des Zustandes s_{max}.

4.2.6 Diskontierungsfaktor γ

Gemäß der in Kapitel 4.2.4 beschriebenen Strategiefunktion strebt ein Lernsystem auf Basis Verstärkungslernen nach Belohnungsmaximierung. Dabei können zwischen dem aktuellen Zustand $s = s_t$ und dem angestrebten Zustand $s = s_{max}$ eine beliebige Anzahl von Zuständen liegen, deren Belohnungswerte in der Praxis nur selten monoton zunehmen. Oft sind Belohnungen gestört oder fehlen komplett. Trotzdem muss ein System auf Basis von Verstärkungslernen in der Lage sein, zu lernen, die richtige Aktion in Bezug zum langfristigen Ziel, somit zur langfristigen Belohnungsmaximierung, zu treffen.

Mit dieser Anforderung wird der eigentliche Kern der Aufgabenstellung formuliert, denn wenn die Belohnungswerte bereits unmittelbar dem Aspekt der langfristigen Auswirkung entsprächen, könnte man die Aktionsselektion zu jedem Zeitpunkt direkt nach den Belohnungswerten ausrichten. Diese Anforderung an die Belohnungswerte kann aber nicht ohne weiteres von einer Belohnungsfunktion erwartet werden. Somit reduziert sich der Anspruch an die Belohnungsfunktion insofern, dass sie lediglich die aktuelle Situation bewertet und der Anspruch der Lernfähigkeit bezüglich der langfristig besten Aktion wird dem Lernsystem zugeschlagen.

Damit kommt dem Diskontierungsfaktor γ eine besondere Bedeutung zu, da im Kapitel 4.2.2 der Einfluss von γ im Rahmen der Aktualisierungsfunktion beschrieben wurde: über diesen Parameter wird festgelegt, welche Bedeutung zukünftigen Belohnungen zugemessen wird. Dabei gilt: $0 \leq \gamma \leq 1$.

Im folgenden Fall soll im Detail untersucht werden: in einem Zustand $s = s_i$ stehen zwei Aktionen zur Verfügung. Die eine Aktion $a_{i \to i}$ führt zum Verharren im aktuellen Zustand; eine andere Aktion führt zum Zustandswechsel zum Zustand s_j, der aber zu einer (verfälschten) Belohnung $r_j < r_i$ führt. Kurzfristig gesehen wäre somit ein Verharren im aktuellen Zustand in Bezug auf Belohnungsmaximierung sinnvoll.

Vom Zustand s_j besteht nun aber die Möglichkeit einer Aktionsselektion $a_{j \to k}$, die zu einem Zustand s_k führt mit $r_k > r_i$ und ein Verharren in diesem höherwertigen Zustand ermöglicht. Langfristig ist somit ein Zustandswechsel von s_i zu s_k erstrebenswert.

Dieses Szenario verdeutlicht die folgende Abbildung.

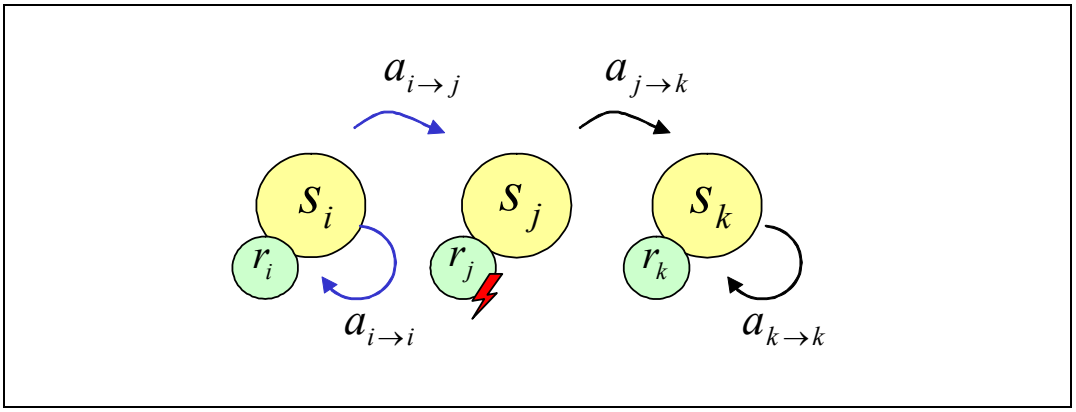


Abbildung 4.9: gestörte Belohnungen

Gemäß vorher ermittelten Formeln konvergiert die maximale Situationsbewertung über alle möglichen Aktionen für den Zustand $s = s_i$ wie folgt:

$$Q^*_{a-\max(i)} = \max\{\frac{r_i}{1-\gamma}; r_j + \gamma\frac{r_k}{1-\gamma}\} \tag{4.26}$$

Damit die Situationsbewertung der Aktion $a_{i\to j}$ höher bewertet wird als die der Aktion $a_{i\to i}$ (d.h. der Zustandswechsel höher bewertet wird als das Verharren im aktuellen Zustand) muss gelten:

$$\frac{r_i}{1-\gamma} < r_j + \gamma\frac{r_k}{1-\gamma} \Rightarrow r_i < r_j(1-\gamma) + \gamma\cdot r_k$$
$$\Rightarrow \gamma > \frac{r_i - r_j}{r_k - r_j} \tag{4.27}$$

Im Endeffekt bedeutet dies, dass ab einem durch obige Formel bestimmbaren Wert von γ eine temporär geringere Belohnung zugunsten einer langfristig erreichbaren höheren Bewertung in Kauf genommen wird.

Nach ähnlichem Schema lässt sich nachweisen, dass auch die Störungen von Belohnungen während einer Kette von Aktionen kompensiert werden. Die Abbildungen 4.10 und 4.11 zeigen diesen Effekt in einem Verhaltensraum mit 7 Zuständen. Die Säulen stellen die dem jeweiligen Zustand zugeordneten Belohnungen dar, die durch Linien verbundenen blauen Punkte die konvergierten Werte von Q^*, d.h. die konvergierten maximalen Werte der Situationsbewertung der jeweiligen Situation (über alle möglichen Aktionen).

In Abbildung 4.10-a sind die Belohnungen ungestört und sind in Richtung Situation i streng monoton steigend. Folgerichtig sind auch die Werte von Q^* monoton steigend. In Abbildung 4.10-b sind die Belohnungen gestört, die maximale Bewertung erfolgt aber immer noch zur Situation i. Trotz diesen Störungen bleibt die Monotonie für die Werte von Q^* erhalten.

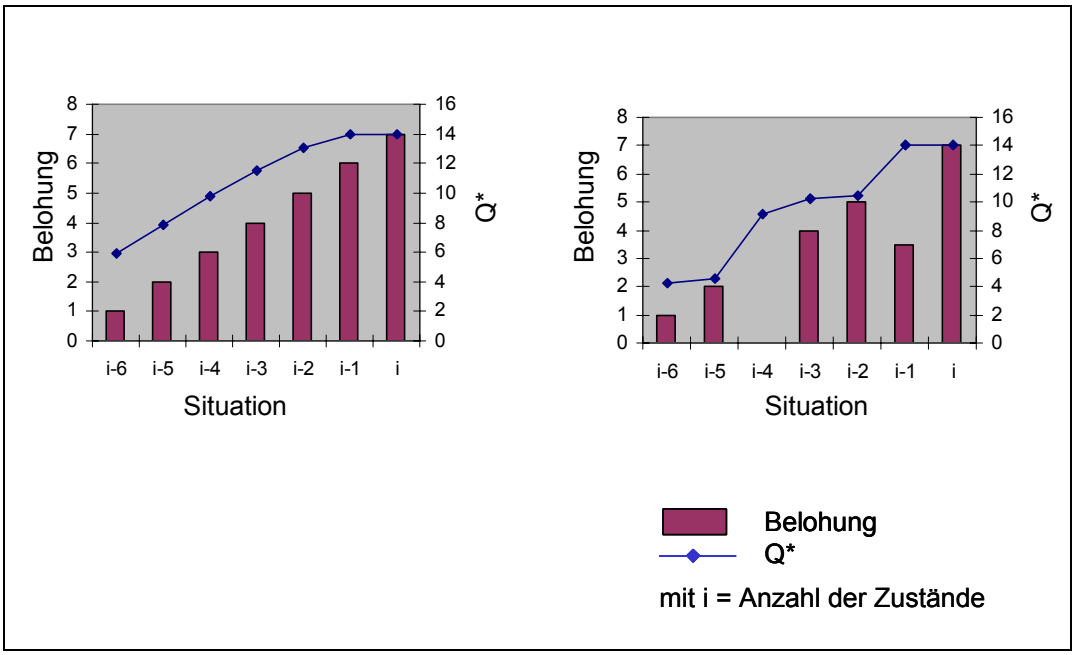


Abbildung 4.10-a und -b: Beziehung von Situationsbewertungen und ungestörten (a), bzw. gestörten (b) Belohnungen bei einem Diskontierungsfaktor von $\gamma = 0{,}5$

In den Abbildungen 4.11-a und 4.11-b ist ein ähnlicher Fall aufgezeigt, nur ist der Wert des Diskontierungsfaktors mit $\gamma = 0{,}9$ deutlich höher als zuvor. In Konsequenz kann die Störung der Belohnungen deutlich ausgeprägter sein, um trotzdem noch eine Monotonie der Situationsbewertungen zu erhalten (Abbildung 4.11-b).

4.2.7 Verfallsfaktor λ

Um eine schnellere Konvergenz der Situationsbewertungen zu erreichen, wird nach jedem Erhalt einer Belohnung nicht nur die Aktualisierung einer Situationsbewertung, sondern gleich eine ganze Kette von Aktualisierungen durchgeführt. Dabei wird die Auswirkung der Aktualisierung umso stärker abgeschwächt, je weiter eine ausgeübte Aktion zurückliegt.

Gegeben sei eine Abfolge von $m+1$ Zuständen (d.h. von s_t bis s_{t+m}). Gemäß Kapitel 4.2.2 wird zum Zeitpunkt $t = m$ zunächst eine Aktualisierung der Situationsbewertung $Q(s_{t+m-1})$ durchgeführt. Zum gleichen Zeitpunkt erfolgt nun auch eine Aktualisierung der Situationsbewertungen $Q(s_{t+m-2})$, $Q(s_{t+m-3})$, etc. Diese Aktualisierungen erfolgen dadurch gedämpft, dass diese Aktualisierungen jeweils mit dem Verfallsfaktor (trace decay parameter) λ multipliziert werden.

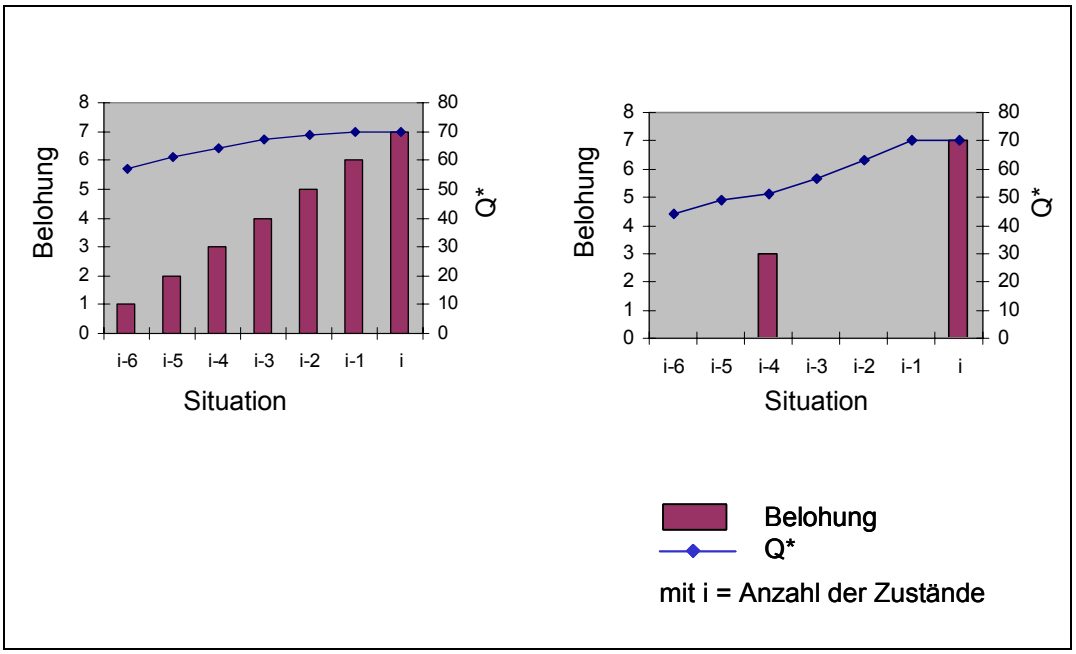


Abbildung 4.11-a und -b: Beziehung von Situationsbewertungen und ungestörten (a), bzw. gestörten (b) Belohnungen bei einem Diskontierungsfaktor von $\gamma = 0{,}9$

Die obigen Ausführungen zeigen einen wesentlichen Vorteil des Verstärkungslernens, der deshalb an dieser Stelle nochmals betont wird. Die Kompensationsmöglichkeit von gestörten oder fehlerhaften Belohnungen ist elementar wichtig für den Einsatz eines lernenden Systems in natürlicher Umgebung. Ohne diese Fähigkeit müsste ein zusätzlicher Aufwand auf die Berechnung, bzw. Validierung der Belohnungen erbracht werden, welches in den meisten Fällen nicht ohne Rückfall auf modellbasierte Ansätze möglich ist.

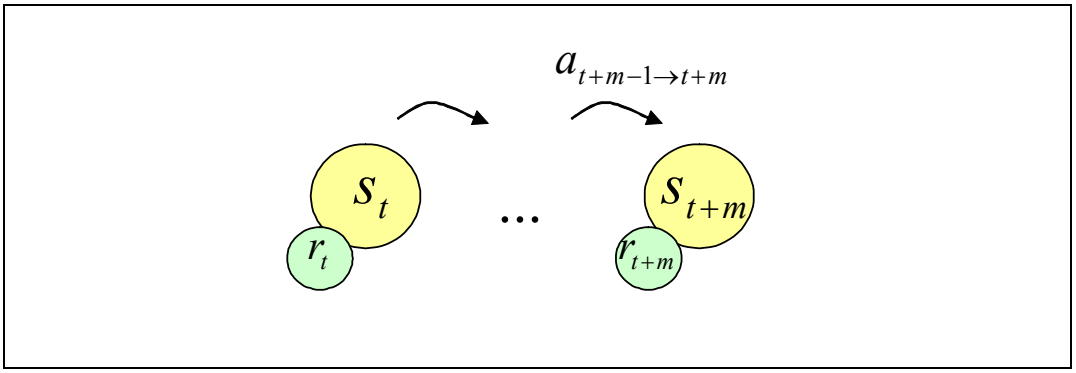


Abbildung 4.12: Zeitliche Darstellung der Zustandswechsel
als Basis für das rückwärtige Aktualisieren

Im Detail lautet die entsprechende Formel:

$$Q(s_{t+m-n)}, a_{t+m-n}) := Q(s_{t+m-n}, a_{t+m-n}) + \alpha \cdot \lambda^n \cdot TDerr \qquad (4.28)$$
$$n \in \{1,...,m\}$$

Dadurch wird erreicht, dass die Aktualisierungen aufgrund einer Belohung zum Zeitpunkt $t = t + m$ bereits schneller in zuvor durchlaufene Situations-Bewertungen zurückgetragen werden.

In den Abbildungen 4.13 und 4.14 ist die Entwicklung der Q^* –Werte von 5 Zuständen s_0 bis s_4 aufgezeigt, die jeweils in der gleichen Reihenfolge durchlaufen werden. In jeder dieser Situationen wird ein Belohnungswert $r = 0$ angesetzt, in allen nach s_4 auftretenden Situationen $s > s_4$ wird ein Belohnungswert von $r = 10$ angesetzt. Zusammen mit Werten von $\alpha = 0{,}5$ und $\gamma = 0{,}7$ ergibt sich je nach Wert des Verfallsfaktors λ ein unterschiedlicher Verlauf der Situationsbewertungen. Dabei wird über der Abszisse die Anzahl Durchläufe abgetragen.

In der Testreihe der Abbildung 4.13 gilt: Verfallsfaktor $\lambda = 0$. Es zeigt sich, dass die Situationsbewertung des Zustandes s_4 bereits nach dem ersten Durchlauf einer Veränderung ausgesetzt ist – aufgrund der Belohnung von s_5, die erstmals ungleich Null ist. Eine Veränderung der Situationsbewertung des Zustandes von s_3 kann erst beim nächsten Durchlauf erfolgen, usw.

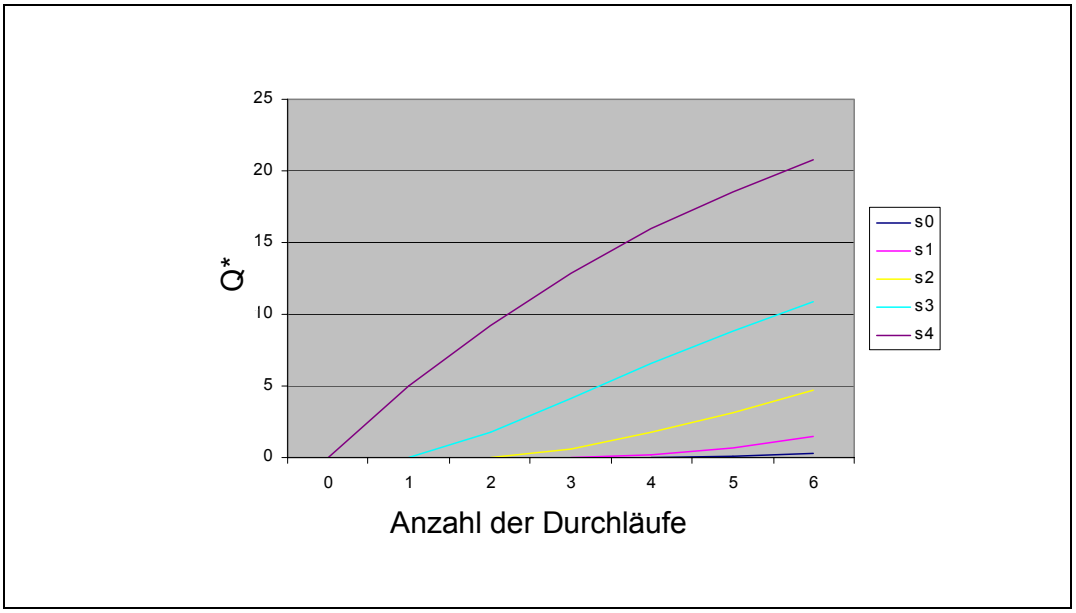


Abbildung 4.13: Situationsbewertungen mit λ = 0

In der Testreihe der Abbildung 4.14 gilt: Verfallsfaktor $\lambda = 0{,}9$. Durch die zusätzlichen Aktualisierungen aller früheren Situationsbewertungen, wird bereits gegen Ende des ersten Durchlaufs die auf die Situationsbewertung von Zustand s_4 angewandte Aktualisierung auch auf die Situationsbewertungen der

Zustände, s_3, s_2, s_1 und s_0 angewendet. Dadurch beginnen die Situationsbewertungen aller Zustände bereits früher zu konvergieren.

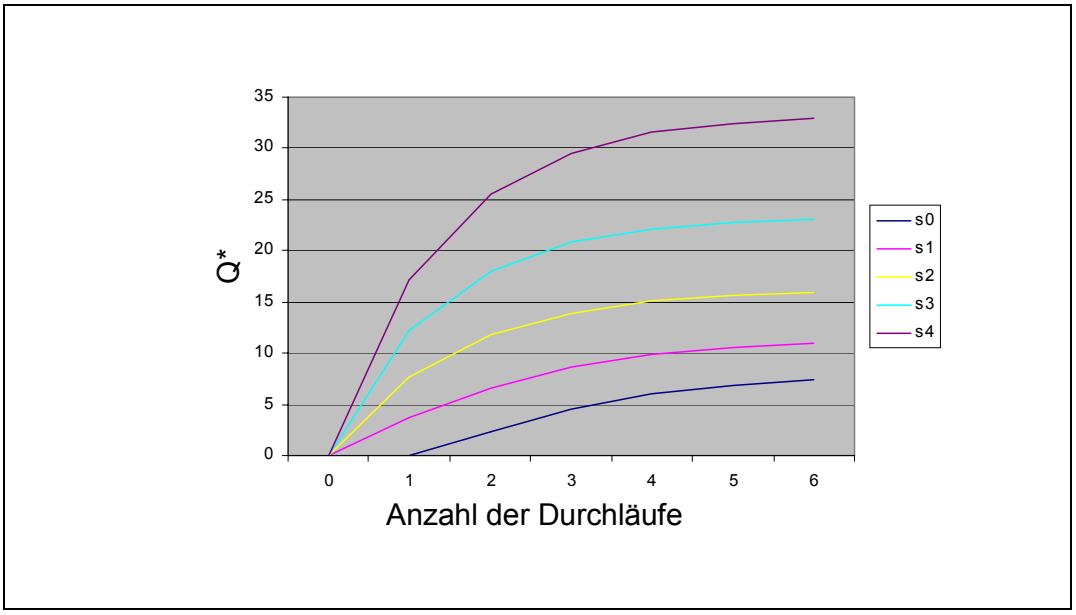


Abbildung 4.14: Situationsbewertungen mit $\lambda = 0{,}9$

Mit Hilfe der zusätzlichen Aktualisierungen kann bei deterministischen Belohnungen eine schnellere Konvergenz der Situationsbewertungen erreicht werden. Allerdings werden bei fehlerhaften oder nicht deterministischen Belohnungen deren Einfluss ebenso nicht nur auf die letzte, sondern auf alle zeitlich vorher durchlaufenen Situationsbewertungen zurückgeschrieben. In der Konsequenz bedeutet dies, dass durch den Einsatz des Verfallsfaktors λ ein schneller, aber dadurch weniger robuster Lernvorgang ermöglicht wird. Dabei liegt der Verfallsfaktor λ im Bereich von $0 \leq \lambda \leq 1$.

4.2.8 Auswirkung verzögerter Belohnungen

In Kapitel 4.2.5.4 wurde die allgemeine Konvergenzformel hergeleitet. Im Falle von verzögerten Belohnungen muss r_t durch r_{t-n} ersetzt werden. Der Parameter n ist damit die Anzahl von Zeittakten, um die eine Belohnung verzögert eintritt.

Im Falle von $n = 1$ können die konvergierten Situationsbewertungen des Verhaltensraums noch hergeleitet werden. Der für die Aktualisierung verwendete Fehlerwert $TDerr$ berechnet sich wie bisher auf Basis der bisherigen Q-Werte der aktuellen sowie der folgenden Situation sowie des Belohnungswertes der folgenden Situation. Wird anstelle des Belohnungswertes der folgenden Situation der Belohnungswert der aktuellen Situation verwendet, kann gemäß Kapitel 4.2.5.5 die Auswirkung auf die Q-Werte aufgrund der Differenz der Belohnungswerte konkret bestimmt werden.

Im Falle von $n > 1$ ist für die Bestimmung der Situationsbewertungen des Verhaltensraums eine Wahrscheinlichkeitsberechnung der Belohnungsverzögerung zu erstellen. Da zum Zeitpunkt $t+1$ die Aktualisierung für die Situation s_t berechnet wird und dabei der Belohnungswert der Situation $t-1$ oder früher eingeht, kann ohne Wahrscheinlichkeitsbetrachtung, in welchen Zuständen sich das System zu diesem früheren Zeitpunkt befunden hat, keine Bestimmung des Verhaltensraum erfolgen. Diese Analyse wird in dieser Arbeit nicht weiter ausgeführt und müsste, bei Bedarf, in einer weiterführenden Arbeit aufgenommen werden.

5 KONZEPTION

Die Aufgabenstellung für das Gesamtsystem umfasst mehrere Teilaufgaben, die jeweils über eigene Ansätze gelöst werden. Folgerichtig kann das Gesamtsystem in mehrere Teilsysteme unterteilt werden.

Das Teilsystem, welches sich mit der Auswertung der Sensordaten (hier: Bildfolgen) beschäftigt, liefert als Ergebnis eine parametrische Szenenbeschreibung und wird im Folgenden als *Teilsystem Bildverarbeitung* bezeichnet. Eine *parametrische Szenenbeschreibung* ist in dieser Arbeit die Nachbildung von Fahrbahnmarkierungen in Form von verketteten Bildpunkten in einem festen, d.h. vom nachfolgenden Teilsystem weiterverarbeitbarem Format.

Ein weiteres Teilsystem wandelt die parametrischen Szenenbeschreibungen in Situationsbeschreibungen um und speichert diese in einer Datenbank. Die Situationsbeschreibungen unterscheiden sich von den parametrischen Szenenbeschreibungen dadurch, dass sie eine feste Metrik als Speicherformat aufweisen und somit als Basis für Ähnlichkeitsanalysen zwischen aktueller Situationsbeschreibung und gespeicherten Situationsbeschreibungen verwendet werden können. Dieses Teilsystem wird im Folgenden als *Teilsystem Mustererkennung* bezeichnet.

Ein drittes Teilsystem implementiert die eingangs bereits beschriebene Lernfähigkeit des Systems und optimiert das Verhaltens, d.h. optimiert die Identifikation von angemessenen Aktionen für die aktuelle Situation. Dieses Teilsystem wird im Folgenden als *Teilsystem Verstärkungslernen* bezeichnet.

Zwei weitere Teilsysteme implementieren die Funktion der Umweltsimulation (*Teilsystem Simulator*) und Belohnungsfunktion (*Teilsystem Belohnungsgenerierung*).

Das Gesamtsystem ist in Abbildung 5.1 dargestellt und gliedert sich in die folgenden Teilsysteme:

- Simulator
- Bildverarbeitung
- Mustererkennung
- Belohnungsgenerierung
- Verstärkungslernen

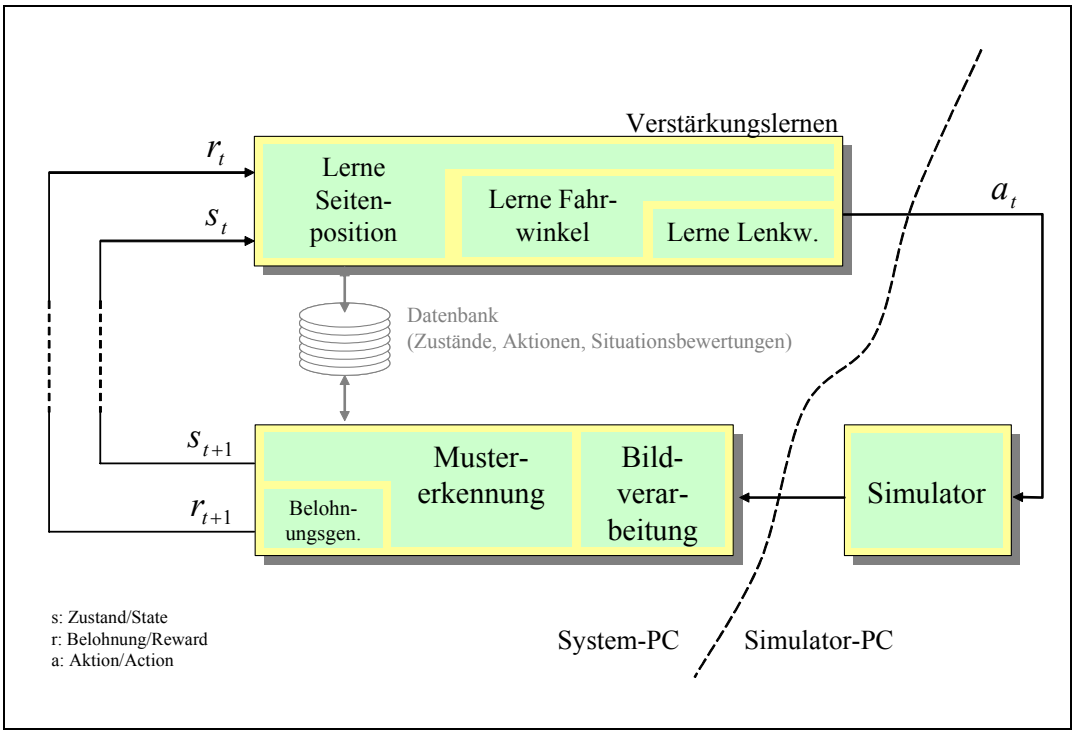


Abbildung 5.1: Untergliederung der Gesamtarbeit in Teilsysteme

Das Teilsystem Simulator ist ein Programm, das für die Simulation der Umgebung, d.h. des Fahrzeuges inklusive der Umwelt, zuständig ist. Dabei simuliert dieses Teilsystem die Position und den Zustand eines Fahrzeugs während einer Fahrt entlang eines Streckenabschnittes. Das Teilsystem Simulator unterstützt unterschiedliche Eingangsgrößen, wie sie später zur Untersuchung der Algorithmen des Lernsystems benötigt werden (siehe auch Kapitel 5.3). Es erzeugt ein simuliertes Bild der Fahrbahn aus Sicht einer im Fahrzeug montierten Kamera.

Im Teilsystem Bildverarbeitung werden die vom Teilsystem Simulator gelieferten digitalisierten Verkehrsszenen weiterverarbeitet. Da eine periodische Digitalisierung der Verkehrsszenen durchgeführt wird, entstehen zeitlich aufeinander folgende Bilder, die als Bildfolgen bezeichnet werden. Als Ergebnis des Teilsystems Bildverarbeitung werden parametrische Szenenbeschreibungen zur Verfügung gestellt, die im Wesentlichen den Verlauf der Fahrbahnmarkierungen nachbilden. Eine konzeptionelle Beschreibung dieses Teilsystems erfolgt in Kapitel 5.1.

Im Teilsystem Mustererkennung werden aus den parametrischen Szenenbeschreibungen zunächst Situationsbeschreibungen gebildet, wobei eine Situationsbeschreibung eine auf Parameter reduzierte Beschreibung der Situation mit fester Metrik des Speicherformats ist. Auf Basis aktueller sowie zuvor gespeicherter Situationsbeschreibungen werden über einen Vergleich ähnliche Situationsbeschreibungen der Vergangenheit identifiziert. Sofern in Bezug auf ein definierbares Ähnlichkeitskriterium ähnliche Situationsbeschreibungen identifiziert werden, werden die Referenzen auf diese an das nachfolgende

Teilsystem übergeben. Sollte keine ähnliche Situationsbeschreibung aus der Vergangenheit erfasst worden sein, wird die aktuelle Situationsbeschreibung gespeichert und eine Referenz auf diese an das folgende Teilsystem übergeben. Dabei dienen die referenzierten Situationsbeschreibungen als Zustand im Umfeld des im folgenden Teilsystem beschriebenen Lernverfahrens.

An dieser Stelle sei noch vermerkt, dass durch das Teilsystem Mustererkennung auch ohne nachfolgendes Lernsystem Fahrfähigkeiten nachweisbar sind. Wenn zu jeder gespeicherten Situationsbeschreibung ein der Situation angemessenes Steuerkommando hinterlegt wird, kann nach der Identifizierung der ähnlichsten gespeicherten Situationsbeschreibung dieses Steuerkommando ausgegeben werden. Im Rahmen der vorliegenden Arbeit wurden diesbezügliche Implementierungen und Untersuchungen durchgeführt und die Ergebnisse eines solchen Fahrsystems (d.h. Fahren auf Basis von Mustererkennung) werden in Kapitel 6.4.2 erläutert

Das Teilsystem Belohnungsgenerierung ist für die Generierung des Belohnungswertes einer jeden Situation zuständig. Da in der vorliegenden Arbeit das Verstärkungslernen-System lernen soll, entlang einer Fahrbahn zu fahren, ist der Belohnungswert dann am höchsten, wenn sich das Fahrzeug in der Fahrbahnmitte befindet. Je weiter sich das Fahrzeug von der Fahrbahnmitte entfernt, desto geringer der Belohnungswert. In der vorliegenden Arbeit wird eine umgekehrt proportionale Belohnungsfunktion verwendet, d.h. der Belohnungswert ist umgekehrt proportional zum Abstand von der Fahrbahnmitte. Eine nicht-lineare Belohnungsfunktion führt zu einer anderen Q-Funktion im Verhaltensraum, wobei die für die Aktionsselektion relevanten Verortungen der Maxima identisch sind.

Der Hauptfokus dieser Arbeit liegt auf der Lernfähigkeit; somit ist der Schwerpunkt der vorliegenden Arbeit das Teilsystem Verstärkungslernen. Da pro Situationsbeschreibung mehrere Steuerkommandos hinterlegt werden können (d.h. in jeder Situation ist eine feste Anzahl von verschiedenen Verhalten möglich) müssen diese in Bezug auf ihre Situationsangemessenheit unterschieden werden. Außerdem kann nicht davon ausgegangen werden, dass das jeweils beste Steuerkommando bereits bekannt ist; somit muss das System existente Steuerkommandos nicht nur bewerten, sondern auch neue Steuerkommandos eigenständig ausprobieren. Dieser Anspruch auf Bewertung, bzw. Optimierung von Steuerkommandos für die einzelnen Situationen wird durch Verstärkungslernen (Reinforcement Learning) erreicht.

5.1 Teilsystem Bildverarbeitung

Die Hauptaufgabe des Teilsystems Bildverarbeitung ist es, in einem Einzelbild (d.h. dem Teil einer Bildfolge) die Fahrbahnmarkierungen zu finden und eine parametrische Szenenbeschreibung zu erstellen, die im Wesentlichen die Fahrbahnmarkierungen nachbildet.

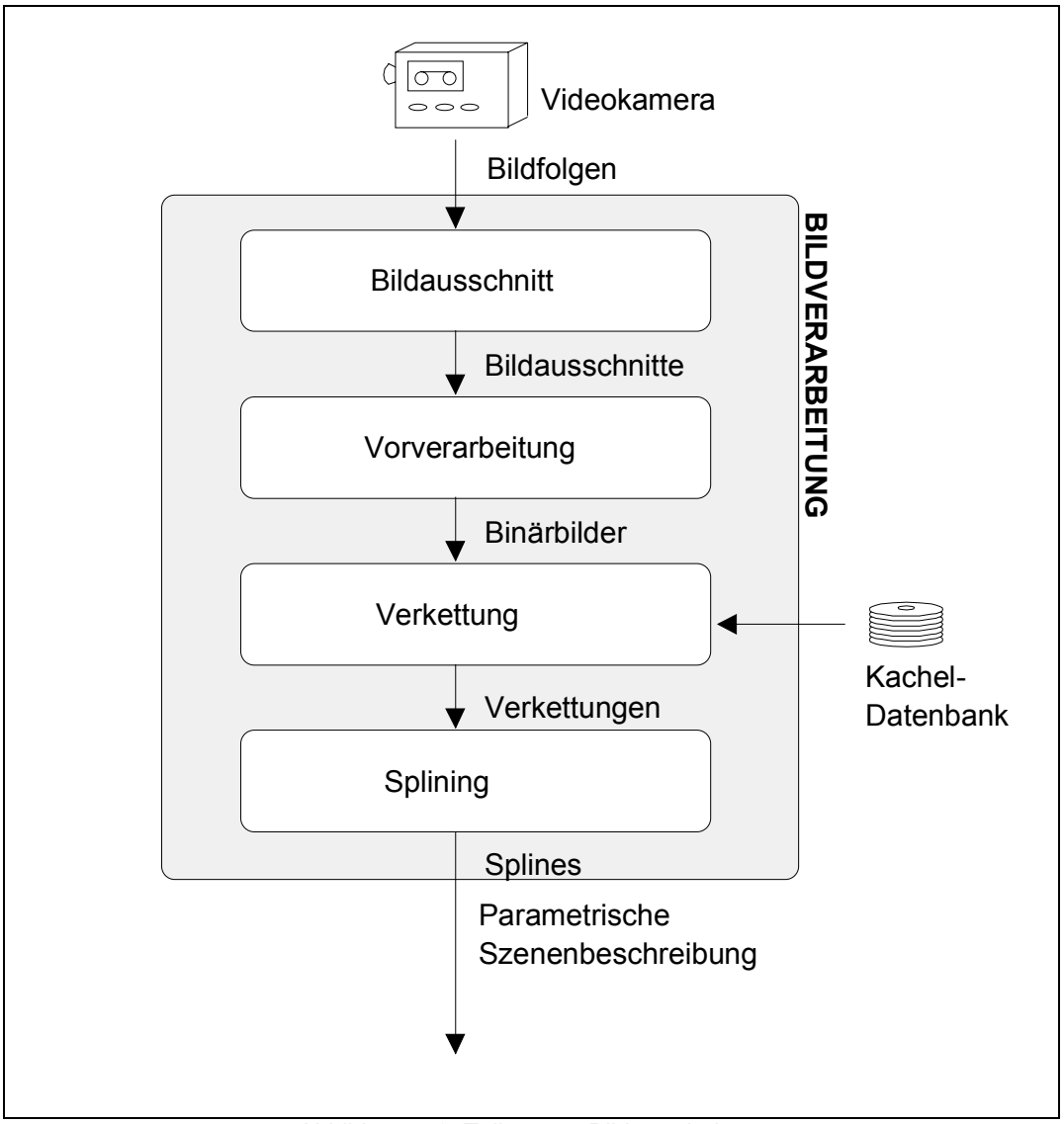


Abbildung 5.2: Teilsystem Bildverarbeitung

Aus den Bildfolgen werden zunächst Bildbereiche ausgeschnitten. Diese Bildausschnitte werden einer Vorverarbeitung unterworfen, die im Wesentlichen in vertikaler Richtung eine Mittelwertbildung über wenige Bildzeilen durchführt und anschließend horizontale Farbdifferenzen ermittelt.

Die gefundenen Farbdifferenzen beruhen entweder auf Fahrbahnmarkierungen oder auf sonstigen starken Farbschwankungen im Originalbild (wobei durch die oben genannte vertikale Mittelwertbildung geringe Schwankungen ignoriert werden). Es wird die Tatsache genutzt, dass Fahrbahnmarkierungen einen spezifischen Verlauf im Bild erwarten lassen (z.B. laufen Geraden auf einen Fluchpunkt am Horizont zu). Wenn die gefundenen Farbdifferenzen auf einer Fahrbahnmarkierung im Originalbild basieren, müssen sie in eine bestimmte Richtung verkettbar sein.

In einem Vorbereitungsschritt wird eine Statistik gebildet, in welcher Richtung Fahrbahnmarkierungen verlaufen und in einer Datenbank abgelegt. Diese Information wird dann bei der während der Bildung von Verkettungen wieder abgerufen und zur Ermittlung der Fahrbahnmarkierungen verwendet.

Im letzten Schritt werden diese Verkettungen in Splines umgewandelt, um einen harmonischen Kurvenverlauf zu erreichen. Dabei ist ein *Spline* eine stückweise zusammengesetzte Funktion, deren Einzelstücke über jeweils eigene Polynome gebildet werden. Die Aufgabenstellung bei der Bildung eines Splines ist es, eine mathematische Darstellung einer Kurve zu ermitteln, die in möglichst glatter Form durch eine vorgegebene Anzahl von Punkten verläuft oder sich zumindest an diese anschmiegt. Letzterer Fall wird als Spline-Approximation bezeichnet. In der vorliegenden Arbeit werden Polynome dritten Grades verwendet, damit wird der Spline als kubischer Spline bezeichnet. Details zu Eigenschaften und der Bildung von Splines ist in [Engeln & Uhlig 96] dargestellt. Diese Splines (die in der vorliegenden Arbeit augrund von kubischer Spline-Approximation gebildet werden) sind dadurch eine Annäherung an im Eingangsbild erkannte Fahrbahnmarkierungen.

In Bezug auf die Darstellung einer Verkehrszene in einem Bild, bzw. in einer Bildfolge müssen die perspektivischen Verzerrungen berücksichtigt werden. Die in der vorliegenden Arbeit zugrunde liegende Koordinatentransformation von Kamera zu Bild ist in Kapitel 3.1 behandelt.

Um eine Fahrbahnmarkierung als Spline annähern zu können, müssen Punkte einer Fahrbahnmarkierung gefunden und diese zu Verkettungen umgewandelt werden. Um solch einen Algorithmus so schnell wie möglich ausführen zu können, wird von diesem Teilsystem die Tatsache genutzt, dass selbst Fahrbahnmarkierungen verschiedener Verkehrsszenen letztendlich ähnlich verlaufen. Ist erst einmal ein Anfangspunkt einer Fahrbahnmarkierung gefunden, kann ein Verkettungsalgorithmus auf vorgegebene Suchrichtungen angesetzt werden. Das hierbei verwendete Konzept für den Verkettungsalgorithmus nutzt Verkettungsvektoren, die in Kapitel 3.2 im Detail dargestellt wurden.

Im Folgenden wird detailliert auf die vier Module des Teilsystems Bildverarbeitung eingegangen.

5.1.1.1 Modul Bildausschnitt

Dieses Modul schneidet aus dem eingehenden Einzelbild (als Teil der Bildfolge) einen *Bildausschnitt* aus, da die Fahrbahn oft nur in einem Teilbereich des Einzelbildes sichtbar ist. Randinformationen zu Landschaft oder Himmel werden nicht benötigt und würden bei der weiteren Auswertung nur stören und unnötig Rechenzeit in Anspruch nehmen. Das Ergebnis dieses Moduls sind die Bildausschnitte.

5.1.1.2 Modul Vorverarbeitung

Um eine Weiterverarbeitung in Richtung parametrische Szenenbeschreibung zu ermöglichen, werden die Bildausschnitte binarisiert und im Folgenden als *Binärbild* bezeichnet.

Da Fahrbahnmarkierungen grundsätzlich in vertikaler und nicht in horizontaler Richtung verlaufen, wird ein horizontaler Differenzoperator verwendet.

Da aber selbst auf einheitlichen Objekten (z.B. Fahrbahn, Fahrbahnmarkierung, etc.) Farbschwankunken in gewissen Grenzen normal sind und zudem während des Prozesses der Digitalisierung durch die Videokamera Pixelrauschen auftritt, ist eine vorherige Mittelwertbildung von Pixel mit mehreren Nachbarpixel im Bildausschnitt sinnvoll. Aufgrund des horizontalen Differenzoperators wird eine vertikale Mittelwertbildung durchgeführt.

Je nach verwendetem Differenzoperator sind im Binärbild die gefundenen vertikalen Fahrbahnmarkierungen mehrere Pixel breit. Dies führt später bei der Bildung von Verkettungen zu mehreren parallelen Verkettungen und somit zu Redundanz. Deshalb erlaubt dieses Modul die Ausdünnung des Ergebnisses der Binarisierung.

Das Binärbild unterliegt nach wie vor der in Kapitel 3.1 dargestellten perspektivischen Verzerrung. Bei der Weiterverarbeitung ist es unter Umständen vorteilhaft, diese Verzerrung wieder rückgängig zu machen, bzw. wenigstens größtmöglich zu kompensieren. Genau dies ist eine weitere Funktion von diesem Modul.

Zusammengefasst realisiert dieses Modul die folgenden Funktionen

- Mittelwertbildung & Binarisierung
- Ausdünnung
- Kompensierung der perspektivischen Verzerrung

Das Ergebnis dieses Moduls sind die Binärbilder.

5.1.1.3 Modul Verkettung

Dieses Modul setzt die im Kapitel 3.2 dargestellte Thematik des Verkettungsmechanismus um.

Im Detail wird die Funktion der bedingen Wahrscheinlichkeitsdichte $f(P_{x,y} \mid P_{x+i,y+j})$ angenähert, wobei $P_{x,y}$ und $P_{x+i,y+j}$ die Wahrscheinlichkeiten sind, dass die Bildpunkte *(x , y)*, bzw. *(x + i , y + j)* jeweils Teil der Fahrbahnmarkierung sind. Von der Funktion der bedingten Wahrscheinlichkeiten werden Verkettungsvektoren abgeleitet, die die Suchrichtungen für Verkettungen vorgeben. Dabei werden diese Verkettungsvektoren aus Daten gebildet, die aus einer Datenbank ausgelesen werden, die eigens über das gleiche Teilsys-

tem vorher (durch gezielte Analyse von Einzelbildern oder Bildfolgen) aufgebaut wurde.

Die Verkettungsvektoren sind je nach Position im Eingangsbild unterschiedlich. Verkettungsvektoren in einen perspektivisch unbehandelten Bild links unten sind z.B. tendenziell nach rechts oben ausgerichtet während die Ausrichtung der Verkettungsvektoren im gleichen Bild rechts unten tendenziell nach links oben erfolgt.

Deshalb wird ein Eingangsbild zunächst in gleichmäßige quadratische Unterbereiche der Dimension $n \bullet n$, sogenannte *Kacheln*, unterteilt (siehe Abbildung 5.3). Bei der Bildung von Verkettungsvektoren wird jede Kachel individuell untersucht, d.h. es wird für jede Kachel eine individuelle Schar von Verkettungsvektoren erstellt.

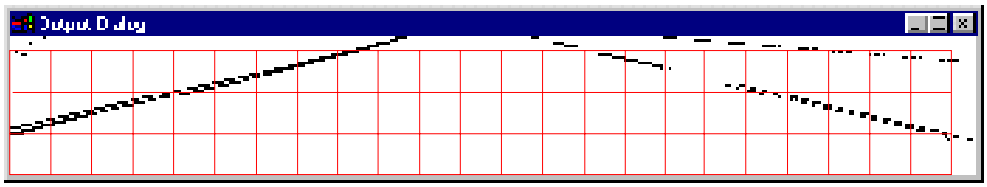


Abbildung 5.3: Unterteilung des Eingangsbildes in Kacheln

Bei der Unterteilung in Kacheln zeigt sich, dass die Kachelgröße als Kompromiss zweier Rahmenbedingungen gewählt werden muss. Zum einen sollte die Kachelgröße so klein wie möglich sein, damit Kurvenverläufe möglichst genau nachgebildet werden können. Andererseits ist die Länge einer Verkettung von der Größe der Kachel abhängig, da die Funktion der bedingten Wahrscheinlichkeit als Basis für die Verkettungsvektoren und somit die Verkettung nur für die jeweils vorliegende Kachel und nicht Kachel-übergreifend gilt. Deshalb sollte die Kachel so groß wie möglich sein, um möglichst große Verkettungssprünge (aus Rücksicht auf Datenvolumen der Verkettungen und benötigter Rechenzeit) durchführen zu können. Letztendlich muss sich im Rahmen der Experimente die Größe der Kachel als Kompromiss beider Rahmenbedingung ergeben.

Bei der Erstellung der Verkettungsvektoren für jede Kachel wird wie folgt vorgegangen:

Zunächst wird für die gesamte weitere Berechnung das Binärbild (dabei repräsentiert ein gesetztes Pixel einen möglichen Teil einer Fahrbahnmarkierung) zugrunde gelegt und diesem Binärbild das erwähnte Raster der Kacheln überlagert.

Nun wird für jede Kachel eine Matrix mit der Dimension $n \bullet n$ (gleiche Dimension wie die Kachel) erstellt, die die so genannten *Kacheldaten* für die jeweilige Kachel aufnimmt. Diese Matrix der Kacheldaten gibt an, wie oft ein Umge-

bungspixel gesetzt war und für die Bildung eines Verkettungsvektors verwendet werden sollte.

Im Detail wird die Matrix der Kacheldaten so aufgebaut, dass jedes einzelne Pixel einer Kachel untersucht wird:

- Ist das aktuell untersuchte Pixel nicht gesetzt, wird der Vorgang für dieses Pixel abgebrochen und es wird zum nächsten Pixel übergegangen.
- Ist das aktuell untersuchte Pixel gesetzt, werden die Umgebungspixel untersucht. Für jedes gesetzte Umgebungspixel wird ein Element der Matrix inkrementiert, wobei die relative Position des Elements innerhalb der Kacheldaten mit der relativen Position des gesetzten Umgebungspixels zum aktuell untersuchten Pixel übereinstimmt. Ist also z.B. ausgehend vom aktuell untersuchten Pixel ein Umgebungspixel mit der relativen Position (1,2) gesetzt, wird das Element der Kacheldaten mit der relativen Position (1,2) inkrementiert.

Abbildung 5.4 verdeutlicht diese Vorgehensweise. Dabei ist es erforderlich, dass die für die Bildung von Kacheldaten verwendeten Einzelbilder (bzw. die daraus abgeleiteten Binärbilder) ausschließlich Fahrbahnmarkierungen enthalten, da sich die statistischen Daten ansonsten sich nicht rein auf den Verlauf von Fahrbahnmarkierungen beziehen.

Dieser Vorgang wird zunächst für alle Kacheln des Binärbildes durchgeführt; anschließend werden mit der gleichen Vorgehensweise weitere Binärbilder ausgewertet. Über diese Vorgehensweise verfügen die Kacheldaten über statistischen Daten der jeweils bildbereichspezifischen Verläufe der Fahrbahnmarkierungen.

Diese statistischen Daten werden zur Bildung der Verkettungsvektoren verwendet. Wie bereits geschildert, sind die die Daten einer Kachel eine Annäherung an die Funktion der bedingen Wahrscheinlichkeitsdichte $f(P_{x,y} \mid P_{x+i,y+j})$, wobei $P_{x,y}$ und $P_{x+i,y+j}$ die Wahrscheinlichkeiten sind, dass die Bildpunkte (x , y), bzw. $(x + i , y + j)$ jeweils ein Teil der Fahrbahnmarkierung sind. Deshalb werden innerhalb einer Kachel die Kacheldaten nach den Zählerständen der Elemente sortiert und für jedes Element wird ein Verkettungsvektor gebildet. Der jeweilige Verkettungsvektor ergibt sich aufgrund der jeweiligen relativen Position des Elements innerhalb der Kacheldaten – dies ist beispielhaft in Abbildung 5.4 dargestellt.

Die gebildeten Kacheldaten werden initial gebildet und als innerhalb der *Kachel-Datenbank* gespeichert. Zur Laufzeit des Lernsystems wird auf diese Daten zurückgegriffen und basierend darauf werden die Verkettungsvektoren gebildet - mit deren Hilfe wird versucht, möglichst schnell Kandidaten von Fahrbahnmarkierungen als solche zu qualifizieren und Verkettungen herzustellen.

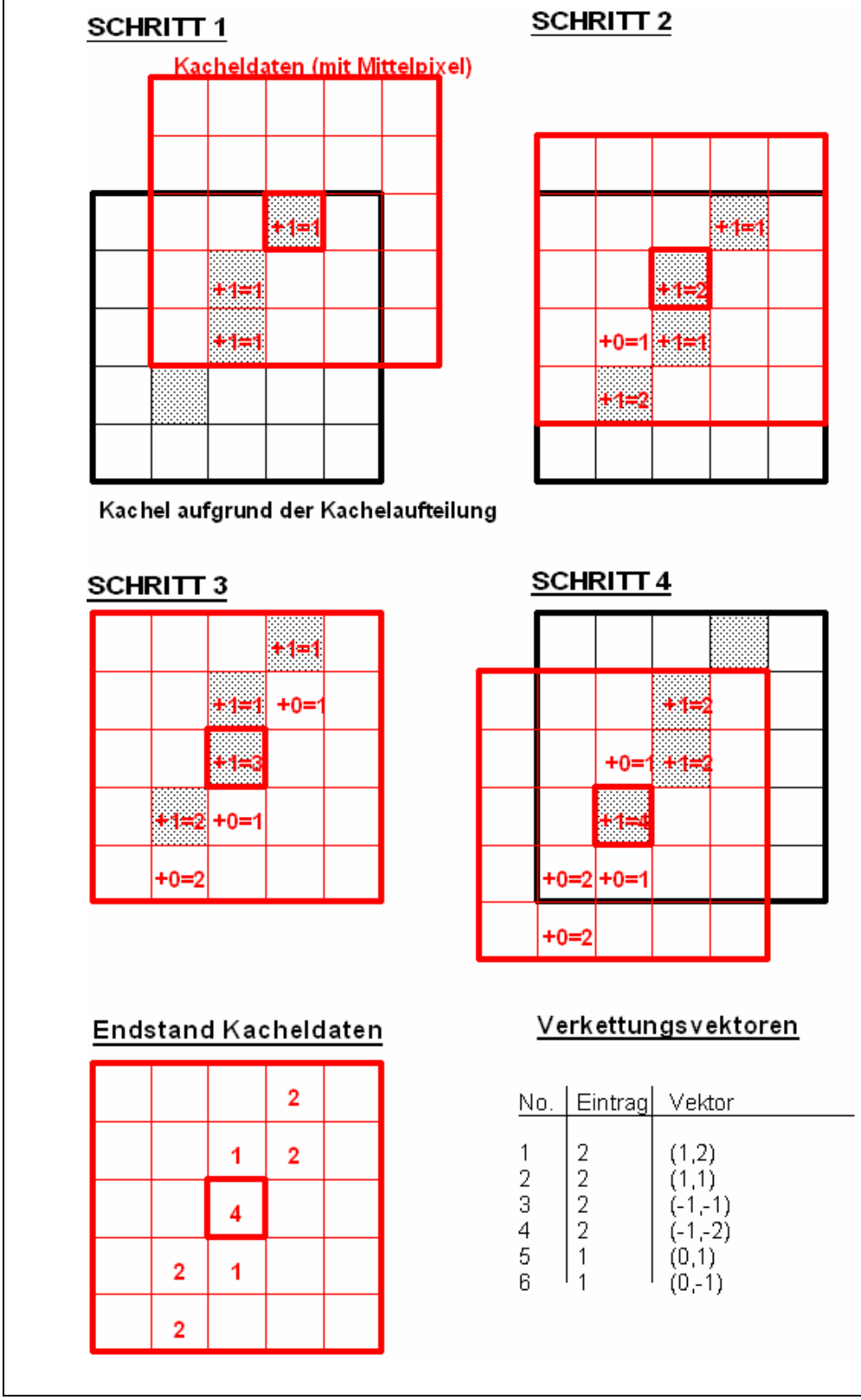


Abbildung 5.4: Kacheldaten und Verkettungsvektoren

5.1.1.4 Modul Splining

Die bisher gebildeten Verkettungen bestehen aus verketten Geradenstücken und ergeben dadurch einen kantigen oder auch zackigen Graphen. Diese Kantigkeit liegt bei realen Fahrbahnmarkierungen nicht vor sondern muss über die bisher erfolgten Schritte der Digitalisierung, Binarisierung und Verkettung entstanden sein.

Aus diesem Grund ermöglicht das Modul Splining die Verkettungen in Splines zu approximieren und einen weicheren Verlauf der Graphen zurückzugewinnen. Das Ergebnis von diesem Modul sind die Splines.

Die Implementierung von diesem Teilsystem wird im Kapitel 6.3.4 behandelt.

5.2 Teilsystem Mustererkennung

Das Teilsystem Mustererkennung hat die Aufgabe, die für das Verstärkungslernen-System benötigte aktuelle Situationsbeschreibung s_t zu bestimmen. Die parametrischen Szenenbeschreibungen des Teilsystems Bildverarbeitung werden in eine Darstellungsform umgewandelt, die es später ermöglichen, Ähnlichkeitsuntersuchungen untereinander durchzuführen. Dazu werden die aus einem Bild ermittelten Splines mit einem festen Raster zur Überlappung gebracht und die Verortungen und Winkel der Schnittpunkte identifiziert. Die variable Anzahl und individuelle Form der Splines eines jeden Bildes kann dadurch in eine einheitliche Darstellung mit fester Größenordnung umgewandelt werden. Details werden in Kapitel 6.4 behandelt.

Diese umgewandelten parametrischen Szenenbeschreibungen werden im Folgenden als *Abstract Complete Situation Description* (*ACSD*) bezeichnet.

Die ACSD's werden vom Teilsystem Mustererkennung immer dann in einer Datenbank gespeichert, wenn in dieser Datenbank nicht bereits eine ähnliche ACSD existiert. Kapitel 6.4 geht auf den Verlauf der Anzahl der ACSD-Einträge über die Zeit ein. Durch diese Vorgehensweise erweitert sich die Datenbank autonom um alle neuen ACSD und nähert sich im Laufe der Zeit an alle möglichen ACSD's einer Umgebung an.

Eine weitere wesentliche Funktion des Teilsystems Mustererkennung ist die Ausgabe eines numerischen Wertes zur Identifikation der aktuellen Situation. Dies wird ebenso durch eine Ähnlichkeitsbetrachtung der aktuellen ACSD sowie aller gespeicherten ACSD's erreicht. Da aufgrund des vorher beschriebenen Verfahrens entweder die aktuelle ACSD oder eine ähnliche ACSD in der Datenbank gespeichert sein muss, wird der Index dieses Speichereintrages als numerischer Wert ausgegeben. Damit ist eine wesentliche Eingangsgröße für das später folgende Verstärkungslernen-System erreicht. Dabei beinhaltet der Index keinerlei qualitative Aussage über die referenzierte ACSD, sondern

lediglich die Information über die Reihenfolge der Aufzeichnung, bzw. Speicherposition in der Datenbank.

Zur Ähnlichkeitsbestimmung wird das unter [Mount 98] beschriebene Verfahren des Approximate Nearest Neighbour (ANN) verwendet. ANN ist eine unter C++ erstellte Bibliothek zur Abstandsberechung von Datenpunkten in einem mehrdimensionalen Raum. ANN unterstützt sowohl die Bestimmung des exakten Abstandes, als auch den innerhalb einer vorgebbaren Toleranzbreite ungefähren Abstand. Auf Basis dieser Abstandsberechnungen, bzw. –schätzungen kann eine festgelegte Anzahl von exakten Nachbarn, bzw. ähnlichen Nachbarn ermittelt werden.

Gegeben seien n Punkte in einem d-dimensionalen Raum $S \in E^d$ und einem weiteren Datenpunkt gleicher Dimension $q \in E^d$. Bei der exakten Suche der nächsten Nachbarn wird durch ANN der nächste Nachbar von q bestimmt, d.h. der Punkt aus S, dessen Abstand zu q minimal ist. Bei der Schätzung der nächsten Nachbarn wird zusätzlich eine Konstante $\varepsilon > 0$ berücksichtigt, die den maximalen Fehler bei der Abstandbestimmung eingrenzt, d.h. der Quotient von ermitteltem Abstand zu tatsächlichem Abstand ist maximal $(1+ \varepsilon)$. Dabei ist die von ANN verwendete Konstante ε nicht mit dem später unter Verstärkungslernen verwendeten Erkundungsparameter ε zu verwechseln.

Es wird davon ausgegangen, dass die Dimension d nicht variabel und unabhängig von der Anzahl n der Datenpunkte des Raumes S ist.

[Arya & Mount 93] und [Arya et al 98] beschreiben den grundsätzlichen Algorithmus und zeigen dabei, dass eine ähnliche Suche im Vergleich zur exakten Suche deutlich schneller durchgeführt werden kann.

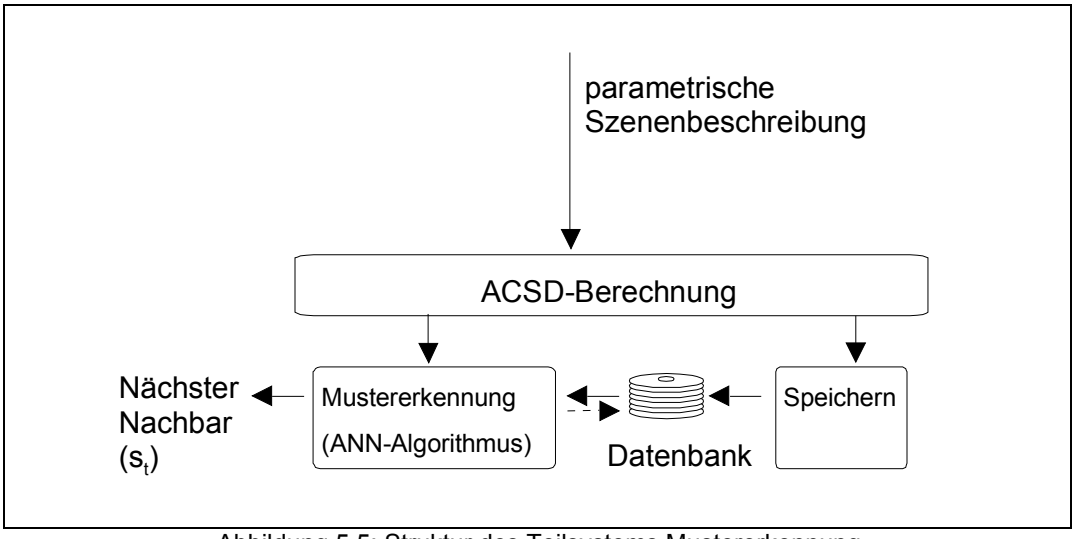


Abbildung 5.5: Struktur des Teilsystems Mustererkennung

Zusätzlich zur ACSD erlaubt die Datenbank auch noch das Speichern von zusätzlichen Daten pro ACSD. Diese Funktion wird von dem später beschriebenen Teilsystem Verstärkungslernen verwendet, da pro ACSD (d.h. pro Situationsbeschreibung) zusätzlich die situationsspezifischen Aktionen und die Situationsbewertungen gespeichert werden.

5.3 Teilsystem Verstärkungslernen

In diesem Teilsystem werden die Funktionen des Verstärkungslernens implementiert. Da in der vorliegenden Arbeit die Auswirkungen der Verstärkungslernen-Parameter im Mittelpunkt stehen, werden drei unterschiedliche Szenarien untersucht.

Hintergrund der Unterscheidung ist die Erkenntnis, dass insbesondere bei komplexen Umgebungen die Auswirkungen der einzelnen Verstärkungslernen-Parameter nicht mehr isoliert untersucht werden können. Deshalb wird zunächst ein extrem einfacher Umgebungstyp angesetzt, um diesen dann über zwei weitere Schritte in einen komplexen und realitätsnahen Umgebungstyp zu überführen. Alle Umgebungstypen sind dabei Umgebungssimulatoren für ein Fahrzeug, das entlang einer Fahrbahn fährt und mit der Aufgabe des Verstärkungslernens, das Fahrzeug zur Fahrbahnmitte zu steuern und bei gegebener Längsgeschwindigkeit das Fahrzeug dort zu halten.

Für die Umgebungstypen gemäß Abbildung 5.1 wurden drei Szenarien unterschieden:

a) Lerne die optimale Seitenposition
b) Lerne den optimalen Fahrwinkel
c) Lerne den optimalen Lenkwinkel

In Bezug auf diese Simulation bezeichnet im Folgenden die Seitenposition die seitliche Position des Fahrzeugs auf der Fahrbahn. Der Fahrwinkel bezeichnet den Winkel, mit dem die Fahrtrichtung des Fahrzeugs vom Fluchtpunkt des Fahrbahnteilstücks abweicht. Der Lenkwinkel bezeichnet die Abweichung von der Fahrtrichtung, in die das Fahrzeug gelenkt wird.

Abbildung Bild 5.6 (a) zeigt das Simulationsmodell. Die Lenkung motorgetriebener Kraftwagen wird seit Beginn des 20. Jahrhunderts über die Achsschenkellenkung erreicht. Dabei wird berücksichtigt, dass sich für eine gleitfreie Kurvenfahrt die Achsverlängerungen gelenkter Vorderräder in einem imaginären Kurvenmittelpunkt treffen müssen. Deshalb muss bei der Einzelradaufhängung gelenkter Vorderrädern der Einschlagwinkel des kurveninneren Rades größer sein als der des kurvenäußeren. Teilweise kompensierend wirkt allerdings die Tatsache, dass aus Stabilitätsgründen heutiger Fahrzeuge ein erhöhter Einschlagwinkel des kurvenäußeren Rades durchgeführt wird. Beides berücksichtigend wird das kurveninnere Rad bis zu 2° stärker eingelenkt. Details zum Status und Historie von Lenksystemen sind in [Ekermann 98] dargestellt.

Der Zusammenhang zwischen Lenkwinkel und Winkel der Vorderräder ergibt sich wie folgt:

Es sei d_{Rad} der Abstand vom Lenkgelenk eines Vorderrads zur Fahrzeugmittelachse. Es sei d_x der seitliche Abstand von der Fahrzeugmittelachse zum Kurvenmittelpunkt. Es sei d_y der Abstand einer durch beide Lenkgelenke gehende Achse zum Kurvenmittelpunkt. Diese Beziehungen sind ebenso in Abbildung 5.6 (b) dargestellt.

Damit ergibt sich

$$\varphi_{Rad1} = \tan\frac{d_x + d_{Rad}}{d_y};\ \varphi_{Rad2} = \tan\frac{d_x - d_{Rad}}{d_y} \tag{5.1}$$

sowie

$$\frac{\varphi_{Rad\,1} + \varphi_{Rad\,2}}{2} = \frac{\tan\frac{d_x + d_{Rad}}{d_y} + \tan\frac{d_x - d_{Rad}}{d_y}}{2} \tag{5.2}$$

$$\approx \tan\left(\frac{(\frac{d_x + d_{Rad}}{d_y}) + (\frac{d_x - d_{Rad}}{d_y})}{2}\right) = \tan\frac{d_x}{d_y} = \Delta\varphi$$

für kleine Radwinkel. D.h. der Lenkwinkel kann als arithmetisches Mittel aus den Radwinkeln für kleine Radwinkel angenähert werden.

Eine der elementaren Bestandteile eines Verstärkungslernen-Systems sind die Zustände. In der vorliegenden Arbeit werden als Zustände die Situationsbeschreibungen verwendet, d.h. im Detail die ACSD's.

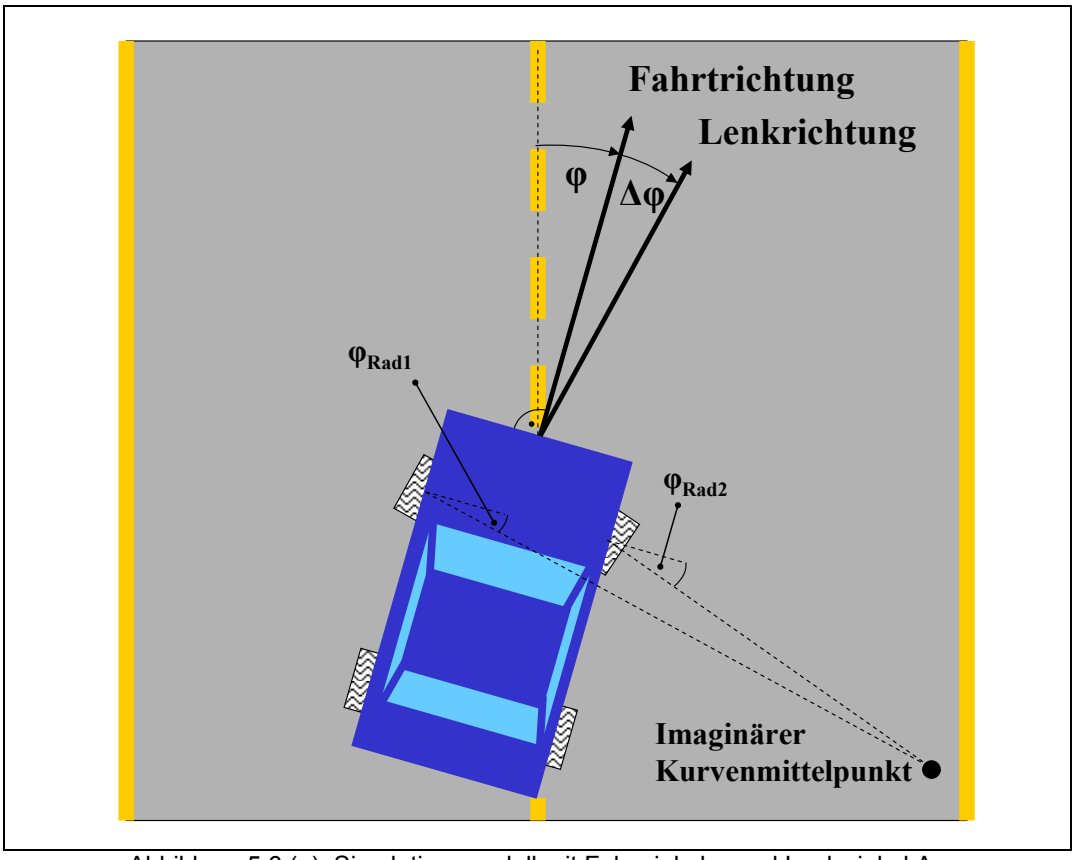


Abbildung 5.6 (a): Simulationsmodell mit Fahrwinkel φ und Lenkwinkel Δφ

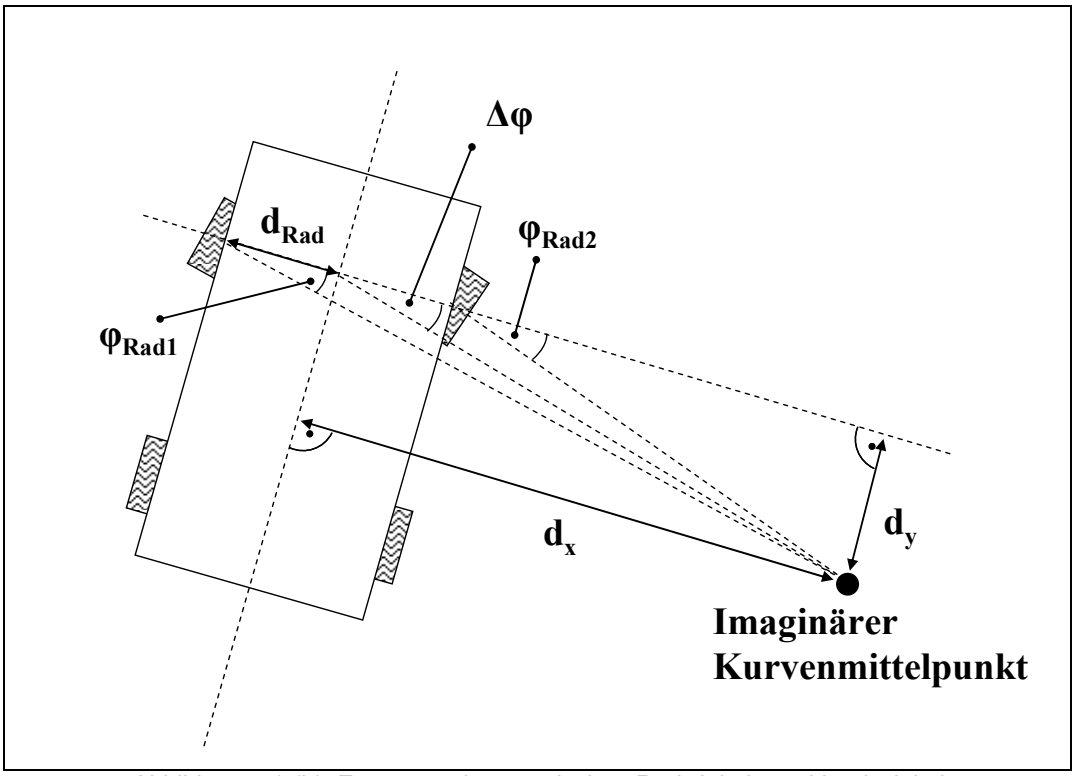


Abbildung 5.6 (b): Zusammenhang zwischen Radwinkeln und Lenkwinkel

5.3.1 Szenario a): Lerne die optimale Seitenposition

In diesem Szenario wird der Simulator für die Umgebung so eingestellt, dass jeder möglichen Aktion a eine bestimmte Seitenposition des Fahrzeugs zugeordnet wird. Wird die Aktion ausgeführt, springt der Simulator direkt diese Seitenposition an (anstatt diese über die Zeit anzufahren). Dieses sehr einfache Szenario mag zwar zunächst nicht realistisch erscheinen, erlaubt aber grundsätzliche Experimente zum Verhalten von Verstärkungslernen-Systemen.

Als Abhängigkeit des Zustandes von den Aktionen gilt:

$$s_{t+1} = k_x a_t$$

mit k_x als fester Konstante. s_{t+1} und a_t sind dabei numerische Werte, nämlich die Referenzen (Index) auf einen Zustand aus der Menge der möglichen Zustände, bzw. auf eine Aktion aus der Menge der möglichen Aktionen. Das Verhalten des Simulators ist dem eines P-Reglers vergleichbar.

Gemäß Q-Lernen ergibt sich der Verhaltensraum auf Basis von Zuständen, Aktionen und Situationsbewertungen. Deshalb wird eine Situationsbewertungsmatrix $Q = Q(s,a)$ gebildet.

Abbildung 5.7 stellt diese Situationsbewertungsmatrix schematisch dar – dabei wird in den Zeilen nach Zuständen und in den Spalten nach Aktionen unterschieden. Auf der linken Seite sind die unterschiedlichen Situationen abgetragen: beginnend oben mit einer Position des Fahrzeugs am linken Fahrbahnrand über eine mittige Position bis hin zur Position des Fahrzeugs am rechten Fahrbahnrand.

Die Spalten der Matrix unterscheiden sich nach den möglichen Aktionen. Beginnend mit einem Steuerkommando nach links (d.h. den linken Fahrbahnrand anfahrend) über entsprechende Zwischenkommandos bis hin zu einem Steuerkommando nach rechts (d.h. den rechten Fahrbahnrand ansteuernd).

Innerhalb der Matrix ist auszugsweise die Berechnung der jeweiligen Situationsbewertungen angegeben.

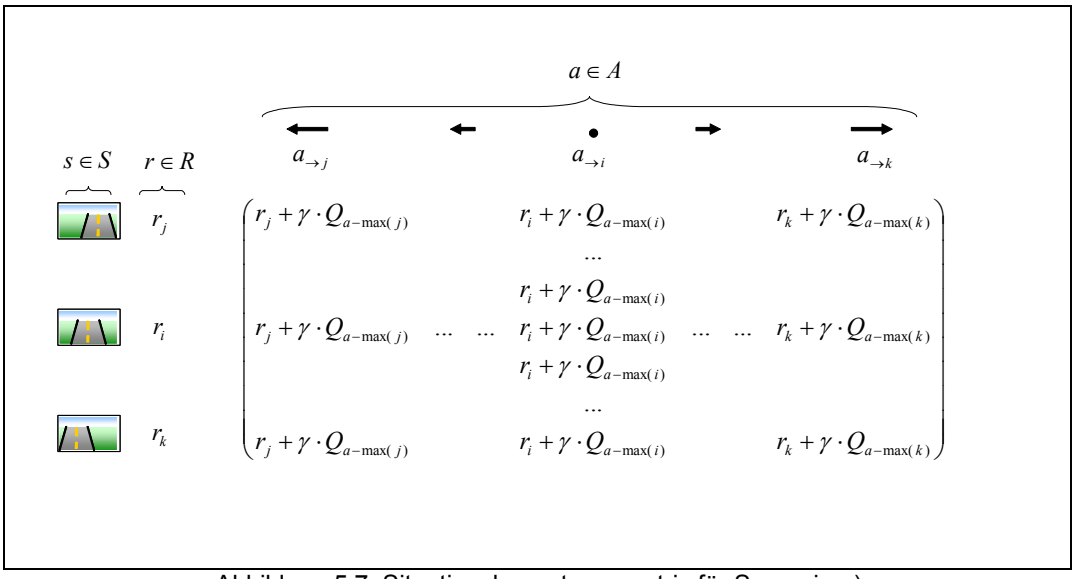


Abbildung 5.7: Situationsbewertungsmatrix für Szenario a)

Wenn nun gilt:

$$r_{j-1} < r_j \text{ für } j < i$$
$$r_{k+1} < r_k \text{ für } k > i$$

mit i, j und k als gültige Indices der Zustände wird gemäß des in Kapitel 4 dargestellten mathematischen Hintergrundes deutlich, wie die graphische Darstellung der konvergierten Q-Funktion qualitativ aussehen sollte und ist in Abbildung 5.8. dargestellt.

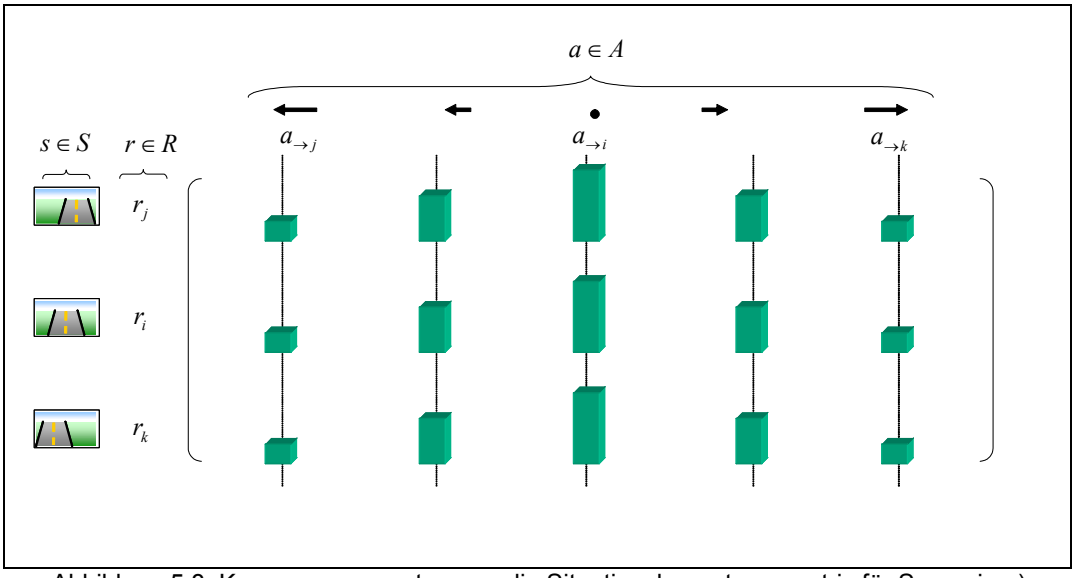


Abbildung 5.8: Konvergenzerwartung an die Situationsbewertungsmatrix für Szenario a)

5.3.2 Szenario b): Lerne den optimalen Fahrwinkel

In diesem Szenario wird der Simulator so betrieben, dass durch eine Aktion a das simulierte Fahrzeug direkt einen bestimmten Fahrwinkel einschlägt. Das Verstärkungslernen-System ist somit in der Lage, eine bestimmte Fahrrichtung anzuspringen. Die Änderung der Situation (vertikale Position des Fahrzeugs auf der Fahrbahn) ergibt sich über den Fahrwinkel und die Verweildauer Δt in diesem Zustand.

Als Abhängigkeit des Zustandes von den Aktionen gilt:

$$s_{t+1} = s_t + k_v a_t \Delta t$$

mit k_v als fester Konstante. Der Simulator simuliert eine Seitenbewegung.

Für den Fall, dass gilt:

$$r_{j-1} < r_j \text{ für } j < i; r_{k+1} < r_k \text{ für } k > i$$

$$j \le w \le k-2$$

ergeben sich die Inhalte für die qualitative Form der Situationsbewertungsmatrix gemäß Abbildung 5.8. Dabei sind am oberen Rand des Bildes wieder die möglichen Aktionen aufgeführt, auf die aber nur auszugsweise eingegangen wird. Die Aktion $a_{w \to w}$ erzielt einen Verbleib in der aktuellen Situation (gleiche vertikale Position des Fahrzeugs auf der Fahrbahn) während die Aktionen $a_{w \to w+1}$ oder $a_{w \to w+2}$ einen geringeren, bzw. stärkeren Positionswechsel in Richtung des rechten Fahrbahnrandes erwirken.

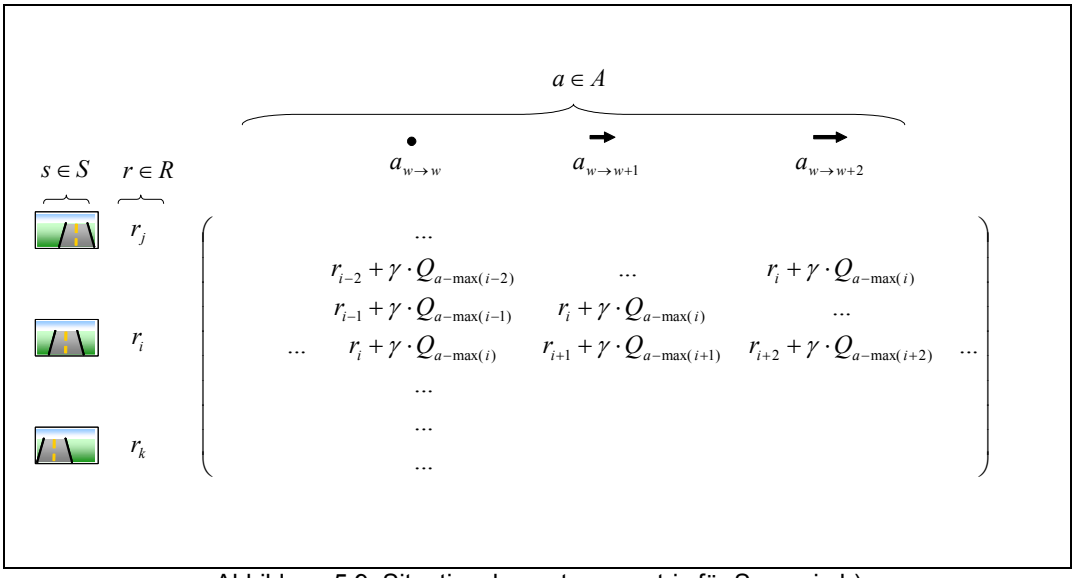


Abbildung 5.9: Situationsbewertungsmatrix für Szenario b)

In Abbildung 5.10 ist der graphische Verlauf der konvergierten Situationsbewertungen dargestellt.

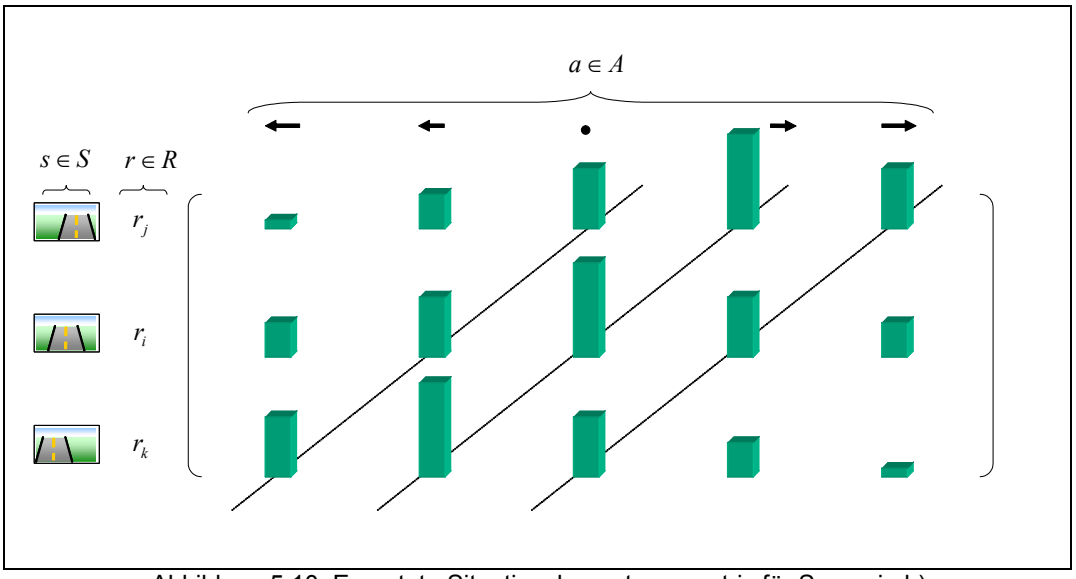


Abbildung 5.10: Erwartete Situationsbewertungsmatrix für Szenario b)

5.3.3 Szenario c): Lerne den optimalen Lenkwinkel

In diesem Szenario wird der Simulator so eingestellt, dass eine Aktion a direkt einen bestimmten Lenkwinkel des Fahrzeugs auswählt. Das Verstärkungslernen-System muss somit eine gewünschte Situation (vertikale Position des Fahrzeugs auf der Fahrbahn) explizit anfahren – die Situation ergibt sich über den Fahrwinkel, der wiederum durch den Lenkwinkel beeinflusst wird, und die Verweildauer Δt in diesem Zustand.

Dabei wird ein 2-Achsen-Fahrzeug durch ein 1-Achsen-System wie folgt angenähert:

$$\varphi_t = \varphi_{t-1} + \Delta\varphi_{t-1} \quad \text{bei mitgelenkten Hinterrädern}$$

$$x_t = x_{t-1} + v_{x_{t-1}} \cdot \Delta t = x_{t-1} + v_{t-1} \cdot \sin(\varphi_t) \cdot \Delta t$$
$$= x_{t-1} + v_{t-1} \cdot \sin(\varphi_{t-1} + \Delta\varphi_{t-1}) \cdot \Delta t$$

$$x_t \approx x_{t-1} + v_t \cdot k_\varphi \cdot (\varphi_{t-1} + \Delta\varphi_{t-1}) \cdot \Delta t \text{ für k}leine\ \varphi$$

Der Simulator simuliert eine Seitenbeschleunigung. Als Abhängigkeit des Zustandes von den Aktionen gilt:

$$s_t \approx s_{t-1} + v_{t-1} \cdot k_\varphi \cdot (\varphi_{t-1} + a_{t-1}) \cdot \Delta t$$

mit k_φ als fester Konstante.

Für den Fall, dass gilt:

$$j \leq w \leq k-2$$

ergibt sich ein graphischer Verlauf der konvergierten Situationsbewertungen wie in Abbildung 5.11 dargestellt. Dabei wird erstmals auch eine dritte Dimension aufgespannt, die durch den bisherigen Fahrwinkel des Fahrzeugs gegeben ist.

Dies liegt daran, dass sich in der obigen Formel Fahrwinkel und Lenkwinkel addieren und in jeder Situation nach bisherigem Fahrwinkel unterschieden werden muss.

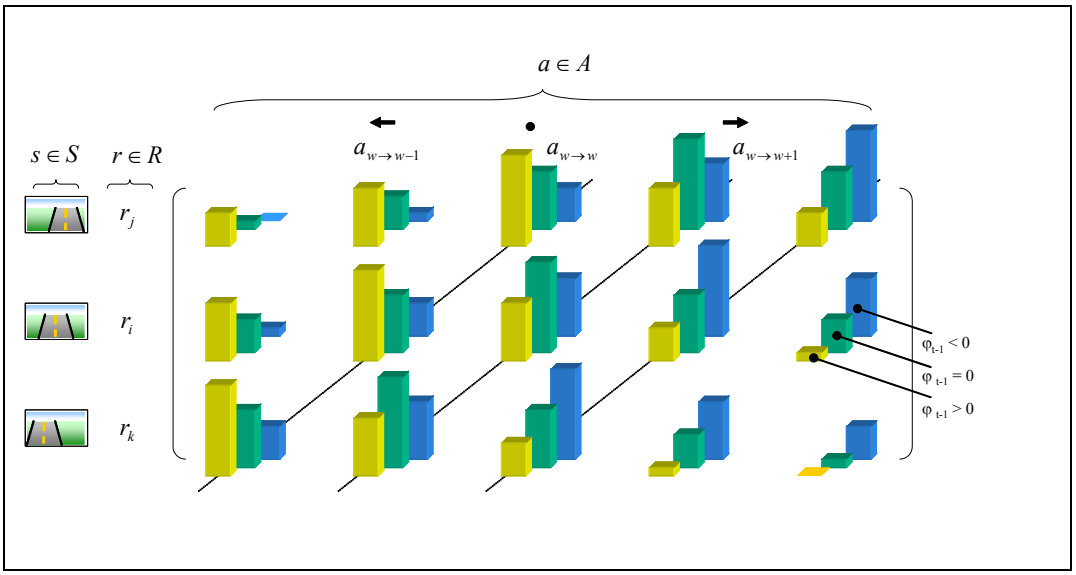


Abbildung 5.11: Erwartung an die Situationsbewertungsmatrix für Szenario c)

6 IMPLEMENTIERUNG UND EXPERIMENTELLE ERGEBNISSE

In diesem Kapitel wird die Implementierung des im Kapitel 3 vorgestellten Konzepts behandelt. Dabei wird getrennt auf die Teilsysteme Bildverarbeitung, Mustererkennung und Verstärkungslernen eingegangen.

6.1 Verwendete Computerkonfiguration

Der grundsätzliche Aufbau der Versuchsanordnung ist in Abbildung 1.3 dargestellt.

Im Detail wurde für den System-PC ein Pentium4-Computer mit 1,5 GHz Taktung unter WindowsMe und für den Simulator-PC ein Pentium-II-Computer mit 400 MHz Taktung unter Windows98 eingesetzt.

Als Video-Verbindung wurde teilweise ein S-VHS-Verbindungskabel (von der Grafikkarte des Simulator-PC's zum Framegrabber des System-PC's) und teilweise eine Videokamera eingesetzt (angeschlossen am Framegrabber des System-PC's).

Für die Übertragung der Steuerkommandos wurde eine an der Universität Siegen entwickelt Verbindungsbox eingesetzt, die zum einen am Parallelport des System-PC's und zum anderen am Gameport des Simulator-PC's angeschlossen wurde.

6.2 Taktung des Systems

In Kapitel 5.3 wurden die unterschiedlichen Berechnungsformeln für die Situationsänderungen dargestellt. In den Szenarien „Lerne den optimalen Fahrwinkel" und „Lerne den optimalen Lenkwinkel" ist die Dauer der Aktionseinwirkung ein entscheidendes Kriterium. Da die Durchlaufzeiten durch die einzelnen Teilsysteme variabel sind, wird das realisierte System durch einen festen 2-Phasen-Takt betrieben. Diese beiden Phasen sind in Abbildung 6.1 visualisiert und stellen einen Zyklus dar.

Zu Beginn der Phase 0 wird das aktuell anliegende Einzelbild eingelesen und die seitliche Position des Fahrzeugs auf der Fahrbahn bestimmt, um die aktuelle Situationsbewertung zu aktualisieren. Unmittelbar danach wird eine Aktion a ausgegeben, die einem Steuerbefehl des Geradeausfahrens entspricht. Dies erfolgt, um eine aus dem letzten Intervall noch wirkende Aktion zurückzusetzen und eine deterministische Interaktion mit der Umgebung während der Phase 0 zu erreichen. Im Anschluss werden die Methoden der Teilsyste-

me Bildverarbeitung bis Verstärkungslernen durchlaufen und die auszugebende Aktion a zwischengespeichert (aber noch nicht ausgegeben).

Die Ausgabe der in der Phase 0 ermittelten Aktion erfolgt erst beim Übergang auf die Phase 1. Diese Vorgehensweise ist nötig, um jede Aktion zu einem festen Zeitpunkt - und nicht nach einer variablen Durchlaufzeit abhängig von der Dauer der Berechnungen in Phase 0 – auszugeben.

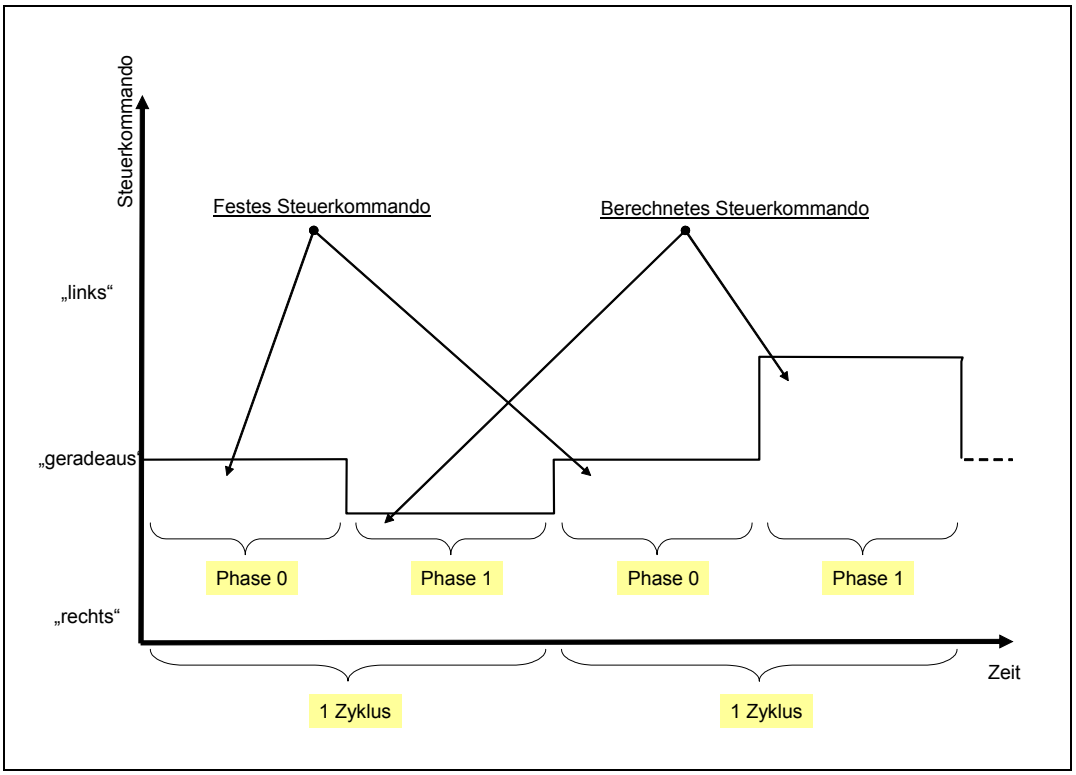


Abbildung 6.1: 2-Phasen-Taktung des Gesamtsystems

6.3 Teilsystem Bildverarbeitung

Das Konzept des Teilsystems Bildverarbeitung ist in Abbildung 5.2 graphisch dargestellt. Bei der Implementierung wird die die perspektivische Verzerrung größtenteils kompensiert (siehe Kapitel 3.1 - Koordinatentransformation) sowie das Verkettungsverfahren (siehe Kapitel 3.2) angewendet. Dabei wird getrennt auf die einzelnen Module eingegangen.

6.3.1 Modul Bildausschnitt

Durch dieses Modul wird aus einem Einzelbild (als Teil einer Bildfolge) ein Teilbild ausgeschnitten – der Bildausschnitt. Der Bildausschnitt wird durch das Rechteck bestimmt, das durch die Parameter *Left*, *Right*, *Top* und *Bottom* bestimmt wird (siehe Abbildung 6.2). Der Ursprung des Koordinatensystems

liegt in der linken oberen Ecke des Einzelbildes. Diese Festlegung wird durchgehend im ganzen Programm verwendet. Alle Parameter werden dabei in der Einheit Pixel angegeben.

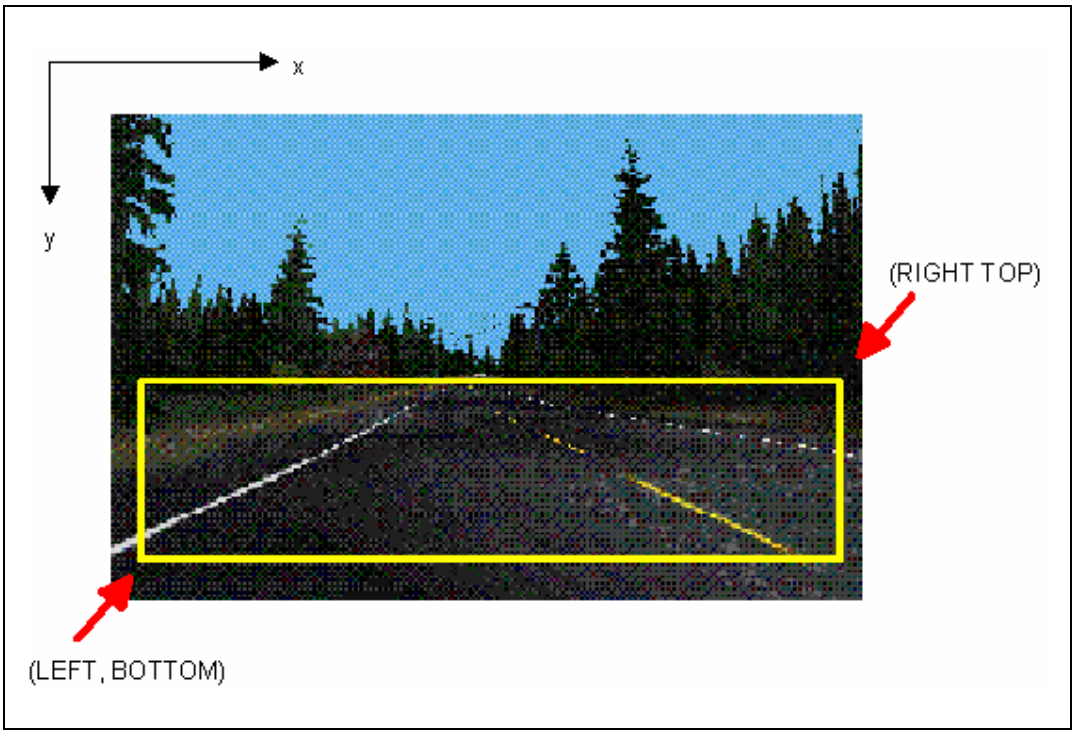


Abbildung 6.2: Festlegung des Bildausschnitts

Das Ausschnittbild sollte so dimensioniert und positioniert sein, dass der relevante Teil der Fahrbahn (d.h. der Bereich der Fahrbahnmarkierungen) erfasst wird. Welche Werte für die Parameter *Left*, *Right*, *Top* und *Bottom* im Detail für das Gesamtsystem sind, hängt von der gewählten Kameraposition und der damit verbundenen perspektivischen Verzerrung ab.

6.3.2 Modul Vorverarbeitung

Nachdem ein Bildausschnitt gewählt wurde, müssen durch das Modul Vorverarbeitung die Kandidaten für Fahrbahnmarkierungen herausgefiltert werden.

Dabei realisiert das Modul Vorverarbeitung die bereits in Kapitel 5.1.1.2 erwähnten drei Grundfunktionen:

- Mittelwertbildung & Binarisierung
- Kompensierung der perspektivischen Verzerrung
- Ausdünnung

6.3.2.1 Mittelwertbildung und Binarisierung

Da sich Fahrbahnmarkierungen farblich von der Fahrbahn abheben, lässt sich die Suche nach Fahrbahnmarkierungen grundsätzlich durch die Faltung mit einem Differenzoperator durchführen. In [Jähne 97] oder [Haberäcker 91] ist die Anwendung von Faltungen im Detail beschrieben.

Die Faltung eines Grauwertbildes $S = s(x,y)$ mit einer Filtermaske H mit der Dimension m ist laut [Haberäcker 91] definiert zu

$$s'(x,y) = \frac{1}{m^2} \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s(x+k-u, y+k-v) \cdot h(u,v) \tag{6.1}$$

mit $k = (m - 1) / 2; m= 3,5,7,...$

Dabei entspricht s einem Grauwertpixel des ungefalteten Bildes und s' einem Grauwertpixel des gefalteten Bildes und ist nicht mit dem Zustand s (state) des Verstärkungslernen-Systems zu verwechseln.

Da in der vorliegenden Arbeit primär die horizontalen Grauwertdifferenzen interessieren (Fahrbahnmarkierungen verlaufen tendenziell parallel zur Fahrtrichtung, daher in vertikaler Richtung im Bildausschnitt), wird die Faltung im eindimensionalen Bereich angewendet und kann auf die folgende Formel reduziert werden:

$$s'(x) = \frac{1}{m} \sum_{u=0}^{m-1} s(x+k-u) \cdot h(u) \tag{6.2}$$

Um auch gerade Werte für m zuzulassen gilt: $k = (m-1)/2$ für ungerade Werte von m und $k = (m-2)/2$ für gerade Werte von m.

Als Filtermaske H werden in der vorliegenden Arbeit dabei folgende Masken unterstützt:

m	***H***
2	$\begin{bmatrix} 1 & -1 \end{bmatrix}$
3	$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$
4	$\begin{bmatrix} 1 & 0 & 0 & -1 \end{bmatrix}$
5	$\begin{bmatrix} 1 & 0 & 0 & 0 & -1 \end{bmatrix}$
6	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$

Tabelle 6.1: Mögliche Filtermasken für H

Dadurch, dass der Bildausschnitt kein Grauwertbild, sondern ein Bild mit den drei Farbkomponenten RGB ist, wird diese Faltung für alle drei Farbkomponenten eines Pixel angewendet und die jeweiligen Absolutbeträge der Differenzen aufsummiert:

$$s'_{RGB}(x) = abs(s'_R(x)) + abs(s'_G(x)) + abs(s'_B(x)) \tag{6.3}$$

Formel (6.3) ist graphisch in Abbildung 6.3 dargestellt.

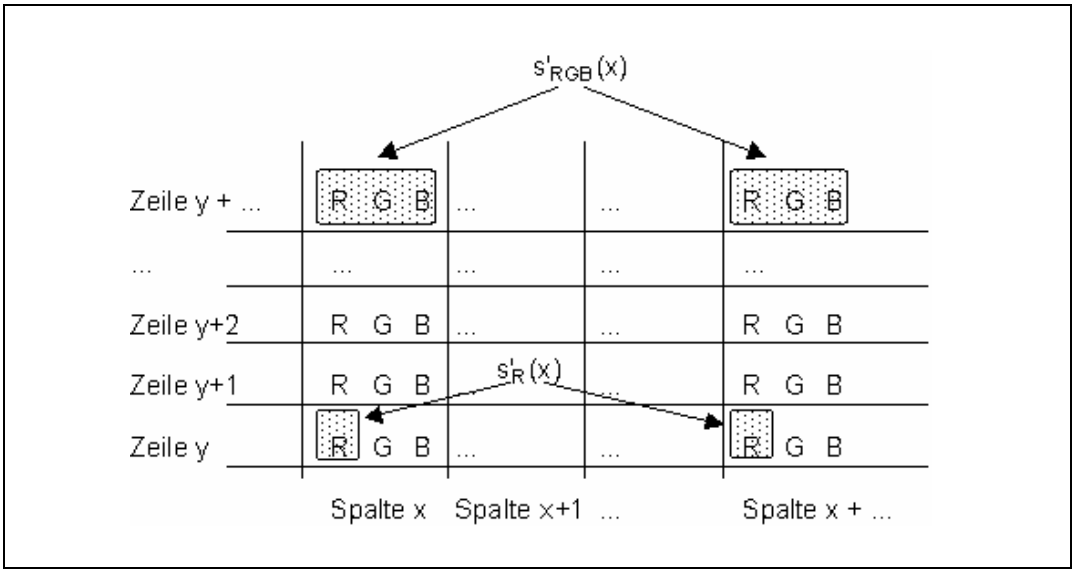


Abbildung 6.3: Berechnung der Grauwertdifferenzen

Der Wert $s'_{RGB}(x)$, der die Grauwertdifferenz der drei Farbkomponenten R, G und B zweier durch die Faltungsmaske H festgelegten Pixel darstellt, wird anschließend binarisiert. Dazu wird $s'_{RGB}(x)$ mit einem Schwellwert verglichen:

$$bin(x) = \begin{cases} 0 \, falls \, s'_{RGB}(x) < Schwellwert \\ 1 \, falls \, s'_{RGB}(x) \geq Schwellwert \end{cases} \tag{6.4}$$

Da sich die Faltungsmaske H jeweils nur auf zwei Pixel bezieht, ist sie relativ anfällig für Pixelrauschen, welches im unverarbeiteten Bildausschnitt vorliegt. Um dieses Pixelrauschen (z.B. erzeugt bei der Digitalisierung des Bildes) bzw. geringe Farbschwankungen von Fahrbahn oder Fahrbahnmarkierung auszugleichen, wird in vertikaler Richtung zusätzlich eine Mittelwertbildung durchgeführt. Die Grauwertdifferenzen werden in vertikaler Richtung aufsummiert und das Ergebnis durch die Anzahl der verwendeten Zeilen geteilt.

$$\overline{s'_{RGB}}(x,y) = \frac{1}{v} \sum_{y=0}^{v} s'_{RGB}(x,y) \tag{6.5}$$

mit v = Anzahl der in vertikaler Richtung für die Mittelwertbildung berücksichtigter Zeilen.

Entsprechend erweitert sich ebenso die Funktion (6.4) wie folgt:

$$bin(x) = \begin{cases} 0 \, falls \, \overline{s'_{RGB}(x)} < Schwellwert \\ 1 \, falls \, \overline{s'_{RGB}(x)} \geq Schwellwert \end{cases} \quad (6.6)$$

Dabei ist zu berücksichtigen, dass das Binärbild (Ergebnisbild nach der Binarisierung) im Falle der vertikalen Mittelwertbildung über weniger Zeilen als der Bildausschnitt (Eingangsbild) verfügt. Die Mittelwertbildung speichert das Ergebnis von v Eingangszeilen in nur einer Ausgangszeile.

Der Wert von v für die vertikale Mittelwertbildung (Anzahl der zu verwendeten Zeilen) kann innerhalb des Prozesses der Binarisierung einheitlich sein, kann aber auch in Abhängigkeit von der vertikalen Position im Bildausschnitt variieren. Eine Notwendigkeit zur Variation des Wertes v lässt sich aufgrund der perspektivischen Verzerrung ableiten. In Kapitel 3.1 zeigt Formel (3.5) dass eine feste Anzahl von Zeilen im oberen Teil des Bildausschnittes eine größere Strecke der Fahrbahn repräsentiert, als die gleiche Anzahl von Zeilen im unteren Teil des Bildes. Deshalb lässt sich bei der hier gewählten Implementierung der Mittelwertbildung der Wert v im Bildausschnitt von oben nach unten erhöhen.

Bildlich gesprochen kann man sich dies so vorstellen, dass der Bildausschnitt in vertikale Streifen unterteilt wird, die in der Breite von oben nach unten linear zunehmen. Diese Breite repräsentiert dabei v, demnach die Anzahl der Zeilen, die für die Mittelwertbildung verwendet wird.

Dabei muss v_{max} als oberer Wert und v_{min} als unterer Wert festgelegt werden. Der Bildausschnitt wird dann in $n = v_{max} - v_{min} + 1$ Streifen geteilt. Für den obersten Streifen kommt v_{min} zum Einsatz; für den untersten Streifen kommt v_{max} zum Einsatz. Alle Streifen dazwischen werden linear angepasst.

6.3.2.2 Kompensierung der perspektivischen Verzerrung

Um die perspektivische Verzerrung zu kompensieren, können die in Kapitel 3.1 hergeleiteten Formeln nicht direkt verwendet werden, da deren Eingangsgrößen nicht bekannt sind und eine Modellierung der Fahrbahn (um die Eingangsgrößen abzuschätzen) vermieden werden soll.

Somit wird das Wissen um die perspektivische Verzerrung dazu genutzt, eine vertikale Stauchung innerhalb des Binärbildes durchzuführen. Dabei hängt die Intensität der Stauchung von der vertikalen Position im Binärbild ab. Es wird zunächst ein Stauchfaktor (*S-Faktor*) festgelegt, der die vertikale Stauchung der untersten Bildzeile angibt. Ein Wert von $S\text{-}Faktor = 50$ entspricht einer Stauchung der untersten Bildzeile auf 50% der ursprünglichen Breite, ein Wert von $S\text{-}Faktor = 100$ entspricht einer Stauchung auf 0%, d.h. Stauchung der kompletten Zeile auf einen einzelnen Punkt.

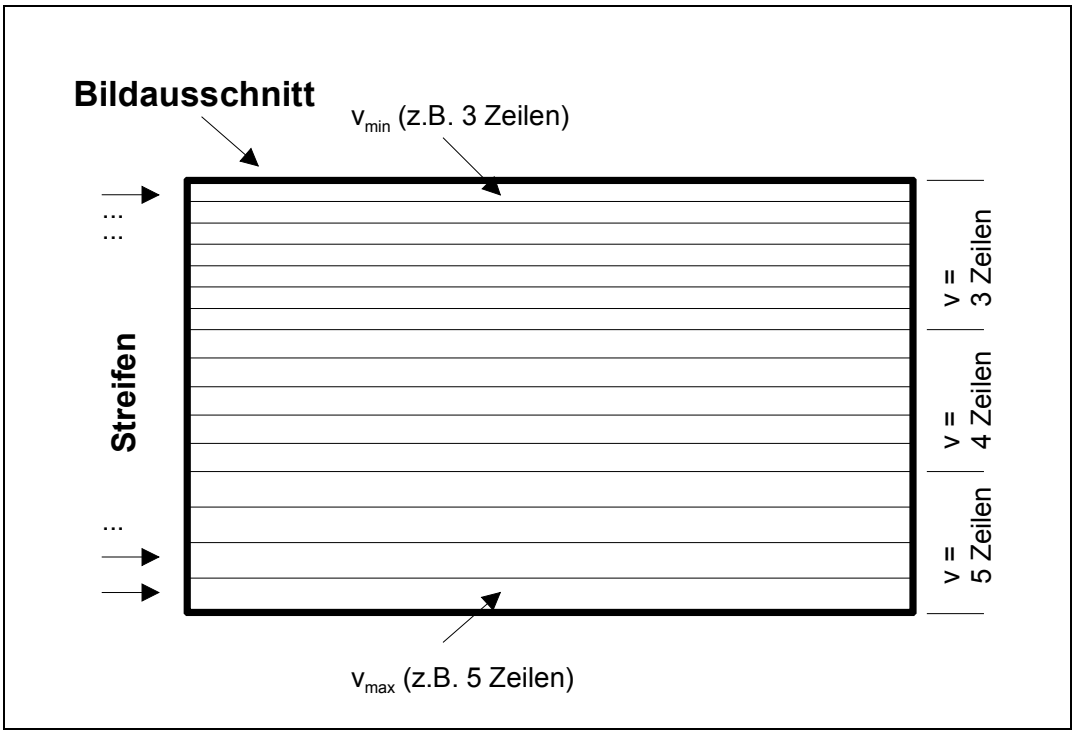


Abbildung 6.4: Unterteilung des Bildausschnittes in Streifen

Für alle anderen Bildzeilen ergibt sich eine geringere Stauchung, die auf Basis eines virtuellen Scharniers berechnet wird. Abbildung 6.5 zeigt die Auswirkungen des Wertes für das virtuelle Scharnier auf die Stauchung des Gesamtbildes. Dabei gibt der Wert des virtuellen Scharniers die horizontale Position an den Bildrändern an. Ein Wert von $Scharnier = 50$ entspricht der Position auf halber Bildhöhe; ein Wert von $Scharnier = 100$ entspricht der Position am oberen Bildrand; ein Wert von $Scharnier = 200$ entspricht einer Position auf doppelter Bildhöhe.

6.3.2.3 Ausdünnung

Als weitere Funktion des Moduls Vorverarbeitung wird die Ausdünnung beschrieben.

Hintergrund ist, dass je nach verwendeter Filtermaske H Grauwertdifferenzen im Bildausschnitt durch mehr oder weniger breite Pixelstreifen repräsentiert werden.

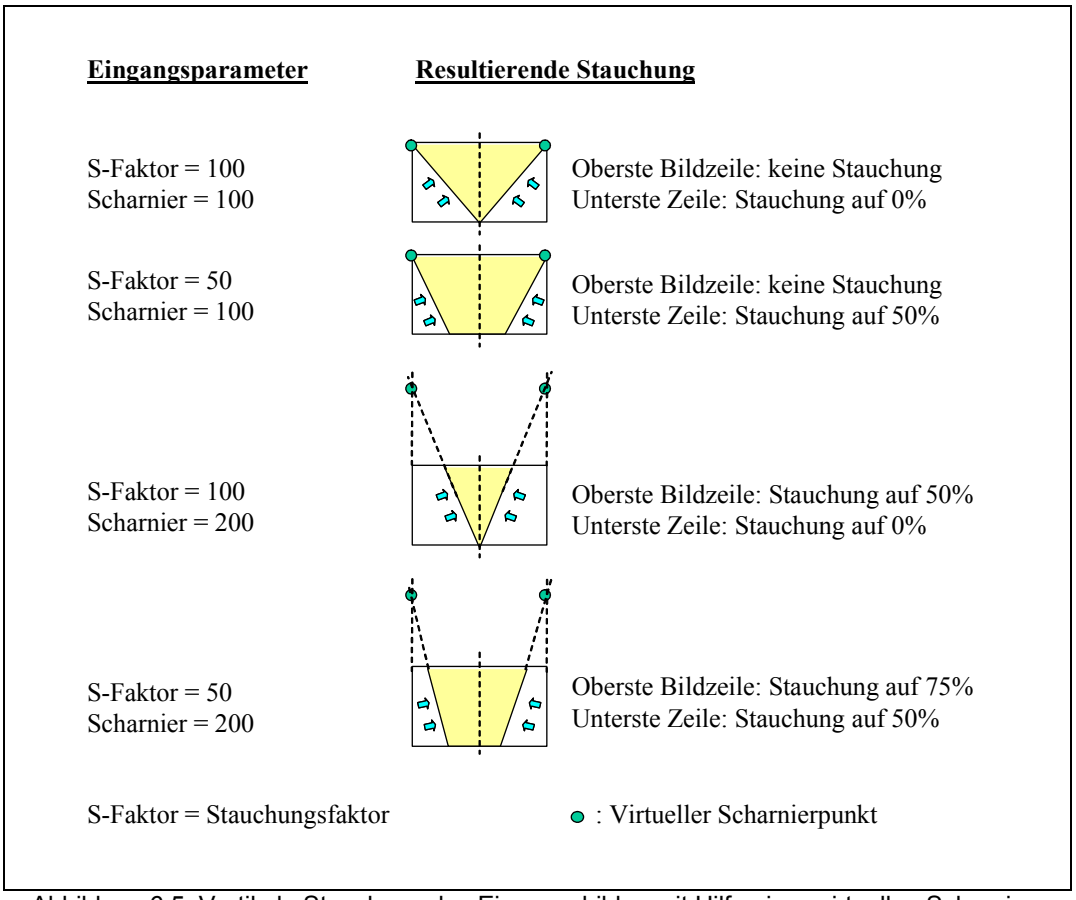


Abbildung 6.5: Vertikale Stauchung des Eingangsbildes mit Hilfe eines virtuellen Scharniers

Abbildung 6.6 zeigt ein Beispiel einer Stauchung eines Binärbildes.

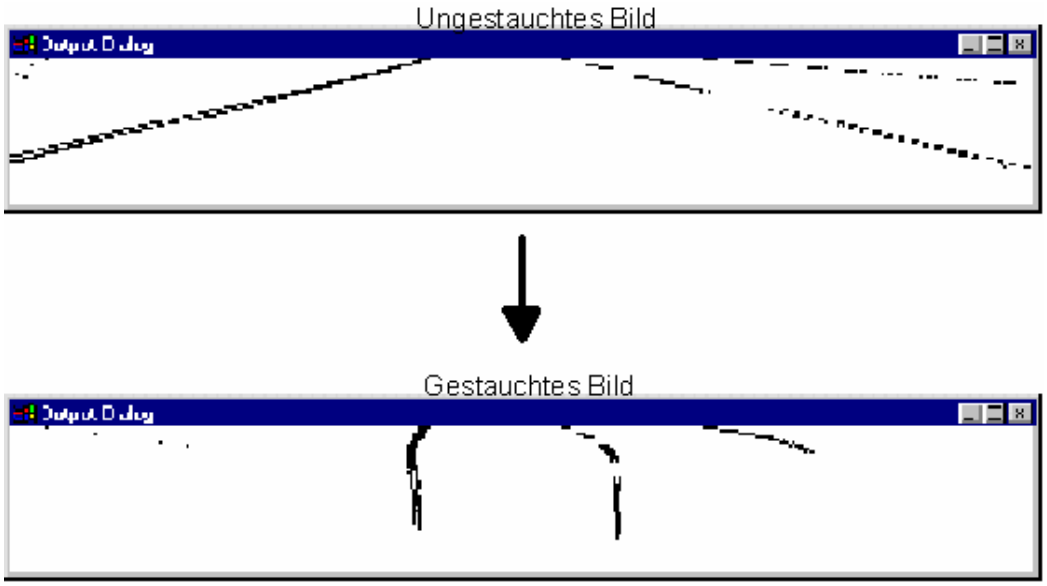


Abbildung 6.6: Stauchung des Binärbildes

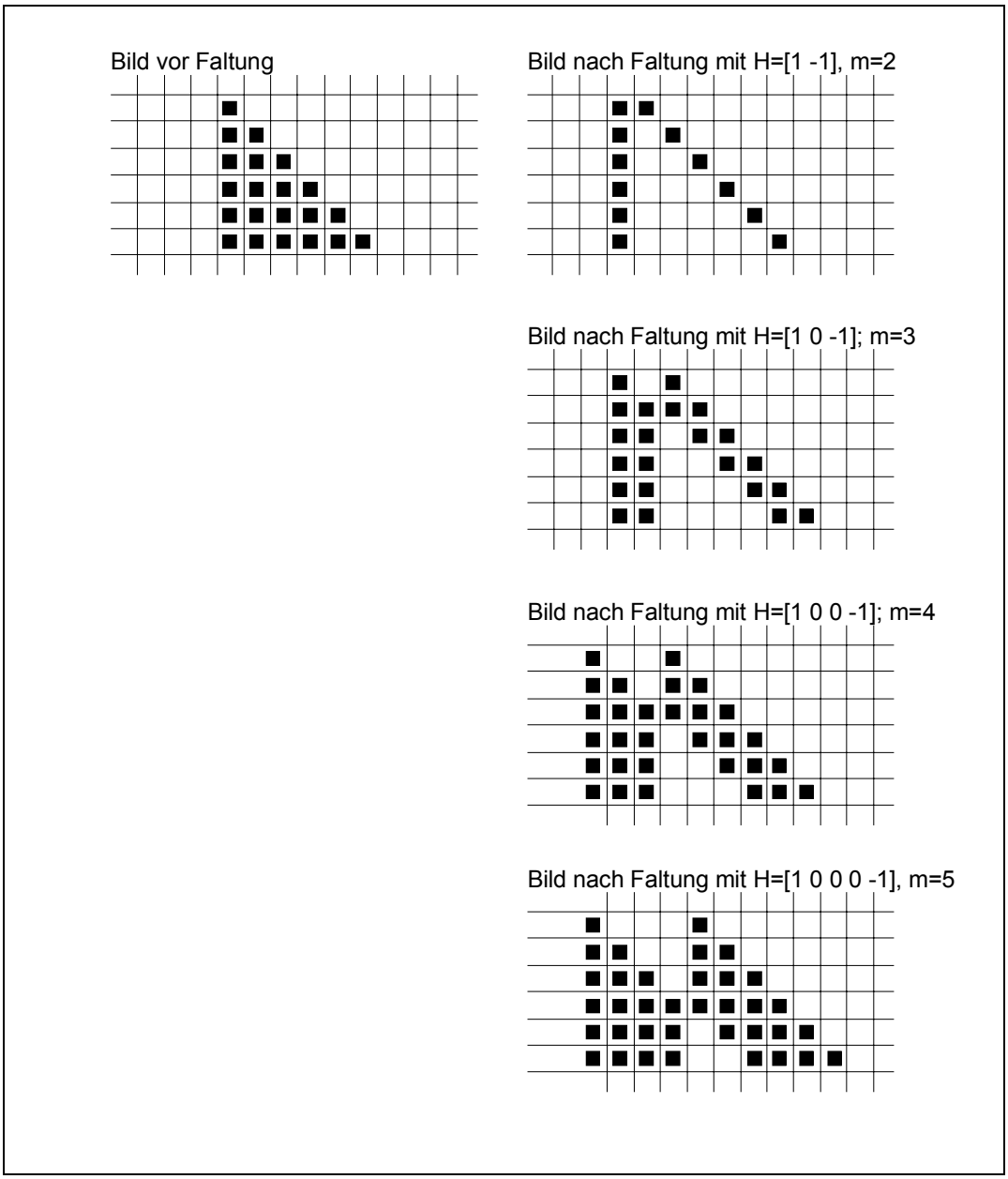


Abbildung 6.7: Breite der Pixelstreifen aufgrund der Faltung

Abbildung 6.7 verdeutlicht an mehreren Beispielen, dass der Pixelstreifen nach einer Faltung die folgenden Breiten annimmt:

$$Breite_{Pixelstreifen} = \begin{Bmatrix} n \; falls \; n < (m-1) \\ (m-1) \; sonst \end{Bmatrix} \tag{6.7}$$

mit m laut Tabelle 6.1 und n der Breite eines Objekts in Eingangsbild der Faltung.

Je breiter ein Pixelstreifen aufgrund einer Grauwertdifferenz im Bildausschnitt ist, desto mehr Verkettungen können später beim Prozess der Verkettung gebildet werden (weil für alle Pixel Verkettungen gesucht werden). Meist sind diese Verkettungen parallel (enthalten also Redundanz), es ist aber auch möglich, dass Zickzack-Verkettungen durchgeführt werden.

Der obere Teil von Abbildung 6.8 verdeutlicht dies an einem Beispiel. Letztendlich ist beides für die spätere Verarbeitung störend und muss vor der Verkettung so weit wie möglich verhindert werden.

Eine einfache Möglichkeit der Minimierung dieses Effektes ist eine Ausdünnung der Pixelstreifen. Dieses Vorgehen ist im unteren Teil von Abbildung 6.8 dargestellt.

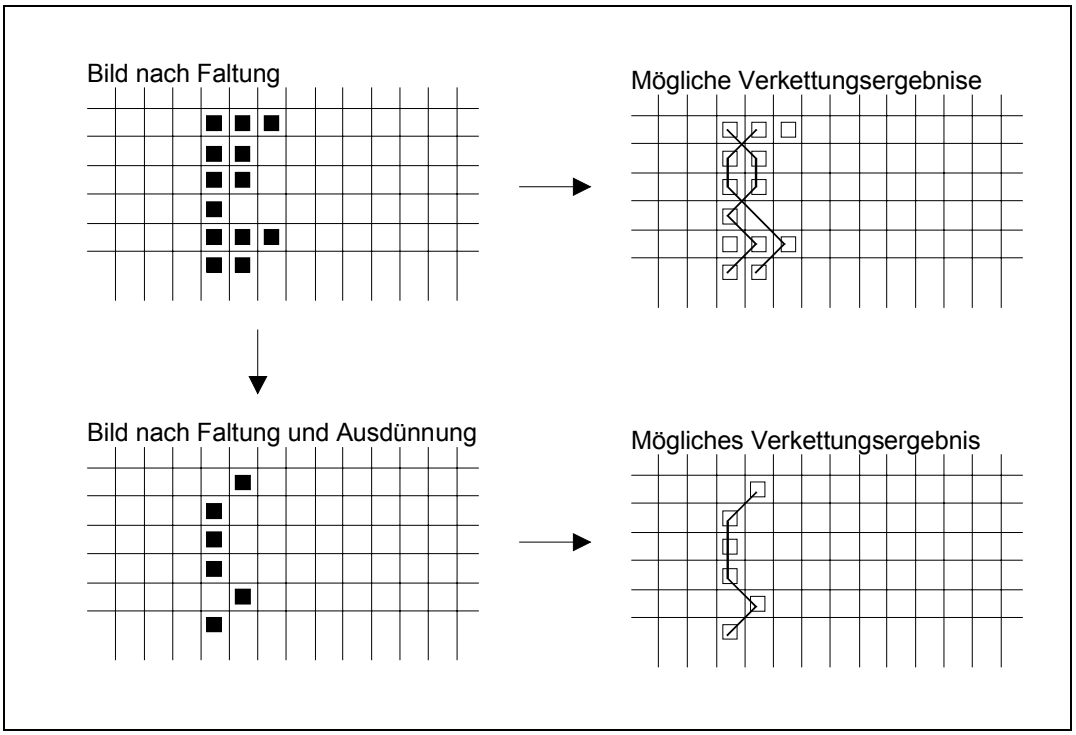


Abbildung 6.8: Ausdünnung

6.3.3 Modul Verkettung

Das Konzept der Verkettungsvektoren wurde in Kapitel 3.2 erläutert. Sie werden im Modul Verkettung zur schnellen und effizienten Bildung von Verkettungen verwendet. Für die Implementierung müssen noch weitere Details beachtet werden, die im Folgenden dargestellt sind.

6.3.3.1 Erstellung der Kacheldaten - Manuelle Vorverarbeitung

Der Verkettungsalgorithmus benötigt als Basis für die Verkettungen die Verkettungsvektoren, die ihrerseits aus den Kacheldaten gebildet werden (siehe Kapitel 5.1.1.3). Das bedeutet, dass die Kacheldaten vor der eigentlichen Laufzeit des Verkettungsalgorithmus gebildet werden müssen. Dazu werden Einzelbilder geladen und die bereits beschriebene Funktion der Binärbildauswertung und Inkrementierung der Kacheldaten aufgerufen. Um die Kacheldaten korrekt aufzubauen, dürfen die Einzelbilder lediglich Fahrbahnmarkierungen enthalten (d.h. keine weiteren Objekte) - dadurch lässt sich die Qualität der Kacheldaten verbessern, da sich diese ausschließlich auf den Verlauf der Fahrbahnmarkierungen beziehen.

6.3.3.2 Invertierung und Initialisierung

Trotz aller durch die Bildung der Kacheldaten gelernten Fahrahnverläufe im Bild kann es ratsam sein, dass jede Kachel über eine Grundschar an Verkettungsvektoren verfügt. Insbesondere bei Unterteilung des Binärbildes in eine Vielzahl von kleinen Kacheln ist es möglich, dass trotz intensivem Aufbau der Kacheldaten (d.h. manuelle Auswertung vieler Einzelbilder), einige Kacheln keine Daten besitzen. Fehlende Daten innerhalb der Kacheldaten bedeuten, dass durch diese Kacheln auch keine Verkettungsvektoren gebildet werden können und somit im späteren Verkettungsalgorithmus keine einzige Verkettung gebildet werden kann. Eine Grundinitialisierung der Kacheldaten jeder Kachel kann in diesen Fällen sinnvoll sein und wird deshalb als optionale Funktion eingeführt.

Ein weiterer Punkt ist die Tatsache, dass der in Kapitel 3.2 beschriebene Algorithmus zur Inkrementierung der Kacheldaten in Verkettungsvektoren in positiver und negativer y-Richtung resultiert. Um kreisförmige Verkettungen zu vermeiden, und wissend, dass die Fahrbahn normalerweise immer tendenziell in eine Richtung (Richtung Horizont, d.h. positive y-Richtung) verläuft, machen Verkettungsvektoren in zwei verschiedene Richtungen (positive und negative y-Richtung) keinen Sinn. Der Ursprung des Koordinatensystems der Verkettungsvektoren liegt dabei in der linken unteren Bildecke. Die positive x-Richtung verläuft nach rechts, die positive y-Richtung nach oben.

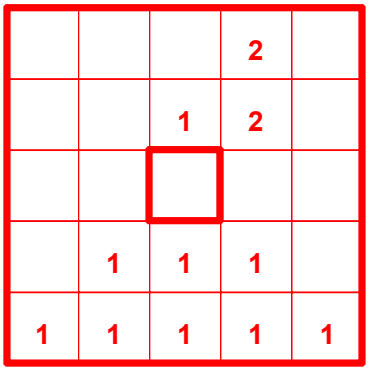
Als Konsequenz wird eine gerichtete Verkettung in rein positiver y-Richtung vorgegeben. In Konsequenz werden die Inkremente der Kacheldaten nur im oberen Teil der Kachel vorgenommen.

Diese beiden Punkte lassen sich im Detail wie folgt kombinieren: Inkremente der Kacheldaten erfolgen, wie oben geschildert, nur oberhalb der Mittellinie der Kacheln. Der untere Teil der Kachel wird V-förmig mit einem Grundwert (z.B. ‚1‘) einmalig initialisiert.

Bei der Bildung der Verkettungsvektoren, werden diese zunächst aufgrund der Daten der oberen Kacheldatenhälfte gebildet. Anschließend werden wei-

tere Verkettungsvektoren aufgrund der Daten der unteren Kacheldatenhälfte, d.h. der Grundinitialisierung, gebildet (wobei diese invertiert werden, um wieder die positive y-Richtung zu gewährleisten). Abbildung 6.9 verdeutlicht dies an einem Beispiel.

Endstand Kacheldaten (*)



(*): ähnlich zu Beispiel lt. Bild 5.4, aber Inkremente nur oberhalb der Mittellinie durchgeführt und Kacheldaten vorher im unteren Bereich initialisiert

Verkettungsvektoren

No.	Eintrag	Vektor
1	2	(1,2)
2	2	(1,1)
3	1	(0,1)
4	1	(-1,-1) → (1,1)
5	1	(0,-1) → (0,1)
6	1	(1,-1) → (-1,1)
7	1	(-2,-2) → (2,2)
8	1	(-1,-2) → (1,2)
9	1	(0,-2) → (0,2)
10	1	(1,-2) → (-1,2)
11	1	(2,-2) → (-2,2)

Abbildung 6.9: Initialisierung von Kacheldaten und Bildung von Verkettungsvektoren

6.3.3.3 Lineare Gewichtung

Um Verkettungsvektoren großer Länge priorisieren zu können, ist es nötig, beim Kumulieren der Kacheldaten das Inkrement stärker zu gewichten, wenn das jeweilige Umgebungspixel weiter vom Mittelpixel entfernt ist. Ebenso ist das Inkrement geringer zu gewichten, wenn der jeweilige Umgebungspixel näher am Mittelpixel liegt.

Dies kann dadurch erreicht werden, dass in dem im Kapitel 3.2 beschriebenen Verfahren der Inkrementierung der Kacheldaten nicht immer der gleiche Wert verwendet wird, sondern ein Wert, der linear zum Abstand zwischen untersuchtem Pixel und Umgebungspixel zunimmt. Dadurch wird eine lineare Abstandsgewichtung berücksichtigt und resultiert, je nach Anstieg der linearen Gewichtung, in der Priorisierung von Verkettungsvektoren größerer Länge.

6.3.3.4 Laden und Speichern der Kacheldaten

Neben den oben erwähnten Funktionen des Moduls Verkettung, lassen sich die Kacheldaten speichern, bzw. laden. Die Tatsache, dass es sich beim

Speichern oder Laden um die Kacheldaten, und nicht um die Verkettungsvektoren handelt, ermöglicht es, diese geladenen Kacheldaten weiter zu entwickeln. Das Speichern erfolgt in die Kachel-Datenbank.

Beim Speichern werden neben den Kacheldaten alle Parameter der bisher beschriebenen Funktionen des aktuellen Teilsystems gespeichert (z.B. Koordinaten des Bildausschnitts, verwendete Filtermaske H, etc.).

6.3.3.5 Berechnung der Verkettungsvektoren

Die Verkettungsvektoren werden immer dann berechnet, wenn sich die Kacheldaten ändern, bzw. diese neu geladen werden.

Dabei kann es vorkommen, dass ein Vielfaches der Kachelgröße nicht exakt mit den Dimensionen des Binärbildes übereinstimmen. Es resultieren Randbereiche, für die keine eigenen Kacheldaten existieren.

Bei der späteren Bildung von Verkettungen gibt es Pixel in diesen Randbereichen, die als Kandidaten für eine Verkettung berücksichtigt werden müssen; somit werden auch für diese Pixel eigene Verkettungsvektoren benötigt. Da zu den Verkettungsvektoren der nächstgelegenen Nachbarskachel keine großen Änderungen zu erwarten sind, werden diese verwendet. Da während des Verkettungsprozesses Ausnahmebehandlungen aus Laufzeitgründen zu vermeiden sind, werden schon bei der Berechnung der Verkettungsvektoren für die Randbereiche die Verkettungsvektoren der nächstgelegenen Nachbar-Kachel kopiert. Abbildung 6.10 zeigt diese Vorgehensweise.

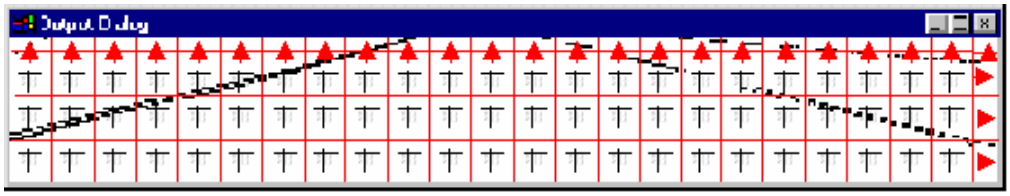


Abbildung 6.10: Kopie der Verkettungsvektoren in die Randbereiche

6.3.3.6 Bildung der Verkettungen

Mit Hilfe der oben beschriebenen Verkettungsvektoren, greift das Modul Verkettung auf die Binärbilder des Moduls Vorverarbeitung zu und es werden aufbauend auf den Pixel des Binärbildes die Verkettungen gebildet.

Selbst bei erfolgter Ausdünnung im Binärbild kann es immer noch zu mehrfachen Verkettungen aufgrund einer einzigen Fahrbahnmarkierung im Bildausschnitt kommen. Jedes Pixel im Binärbild, welches bereits zur Bildung einer

Verkettung verwendet wurde, wird blockiert, um eine mehrfache Nutzung zu vermeiden.

Wird bei der Bildung der Verkettungen durch einen Verkettungsvektor eine gewisse Anzahl von Pixel übersprungen, können diese in einem weiteren Lauf die Basis für eine weitere Verkettung sein.

Dies lässt sich dadurch reduzieren, dass bei Bildungen von Verkettungen übersprungene Pixel in Binärbild ebenso blockiert werden.

Wenn sowohl Ausdünnung als auch Blockade vom übersprungenen Pixel genutzt wird, ist die Anforderung erfüllt, aus einer Fahrbahnmarkierung im Bildausschnitt möglichst wenige Verkettungen zu erzeugen. Diesen Zusammenhang zeigen Abbildung 6.11 (a) und Abbildung 6.11 (b).

Der in dieser Arbeit implementierte Algorithmus stellt die Informationen über die Verwendung der einzelnen Pixel im Binärbild wie folgt zur Verfügung.

- Ein Pixel, das für einen Startpunkt einer Verkettung verwendet wurde, wird rot markiert.
- Ein Pixel, das für einen weiteren Verkettungspunkt verwendet wurde, wird grün markiert.
- Ein Pixel, das für eine weitere Verwendung blockiert wurde, wird blau markiert.
- Ein Pixel, dass nicht verwendet wurde, weil es mit keinem Verkettungsvektor zu einer Verkettung kommen konnte, bleibt schwarz markiert.

Ein Beispiel hierzu ist in Abbildung 6.12 dargestellt. Diese Farbinformationen können zur Verdeutlichung der Funktionsweise analysiert werden.

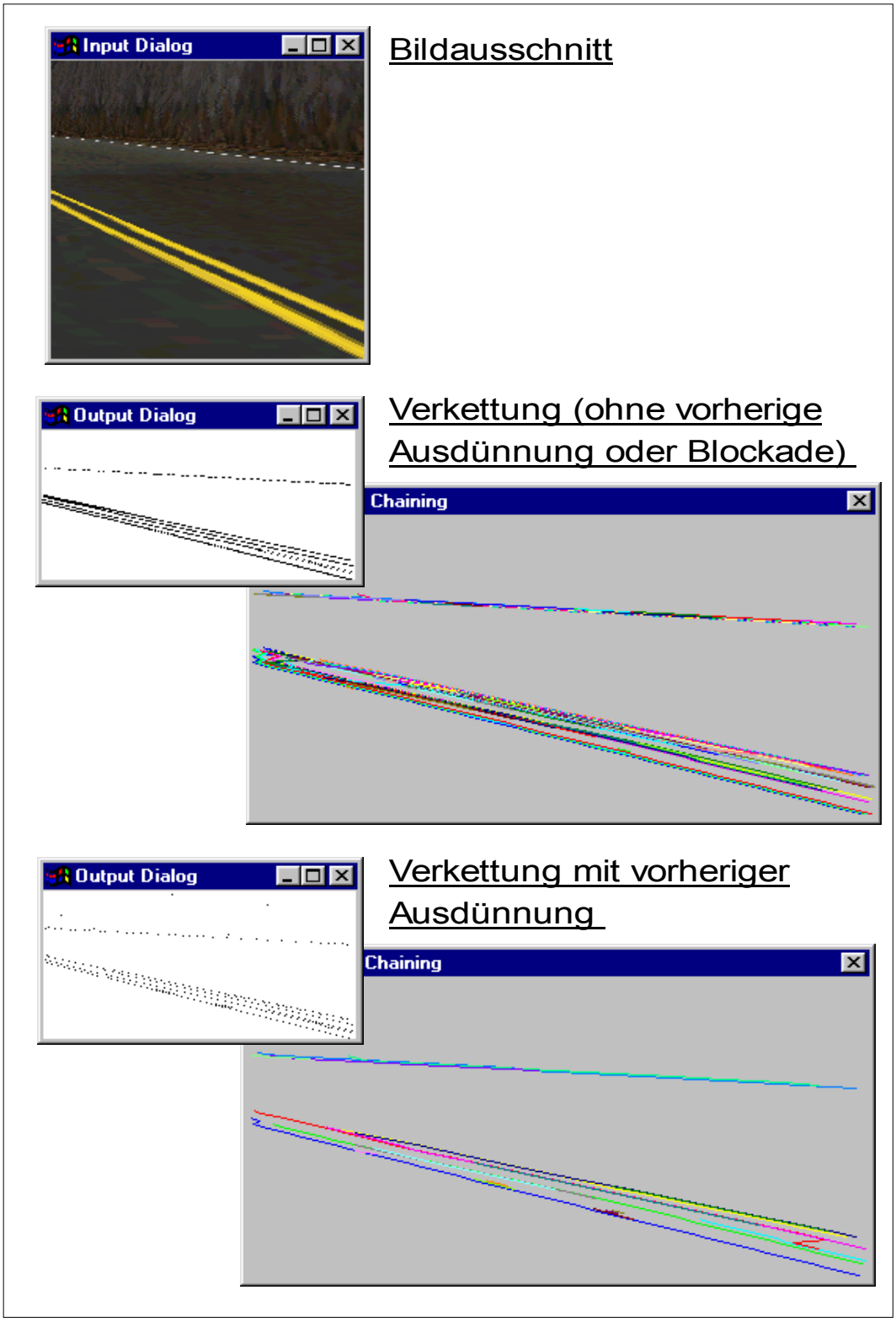


Abbildung 6.11 (a): Bildung von Verkettungen (mit/ohne Ausdünnung)

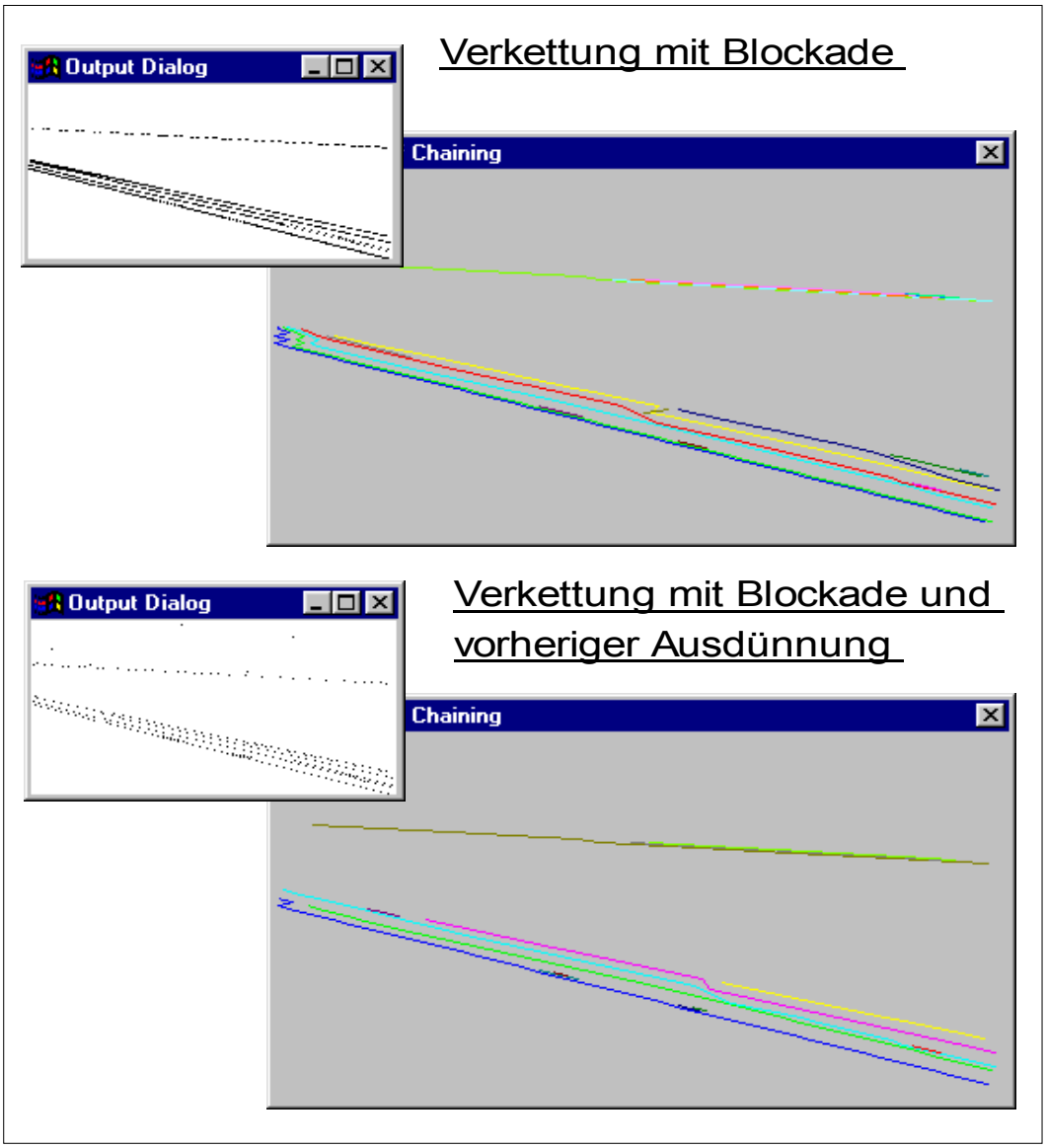
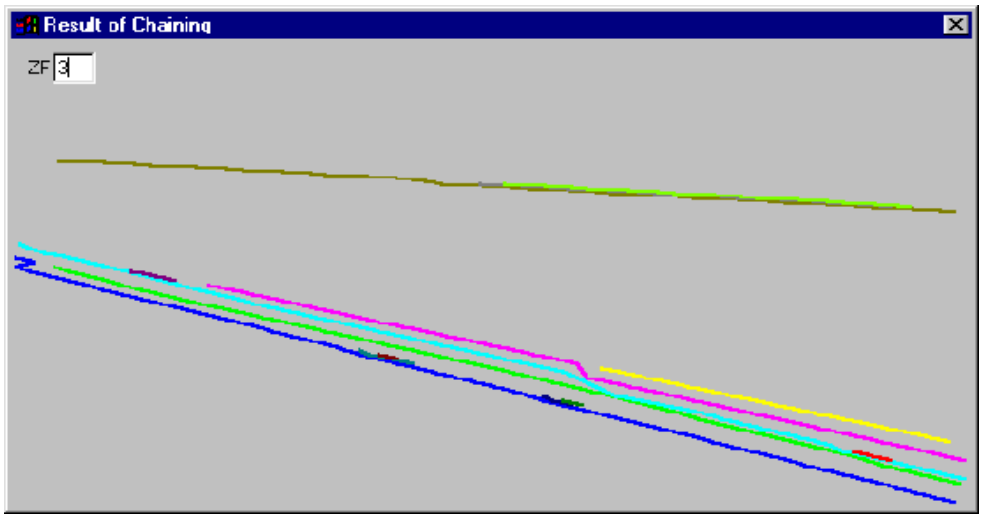
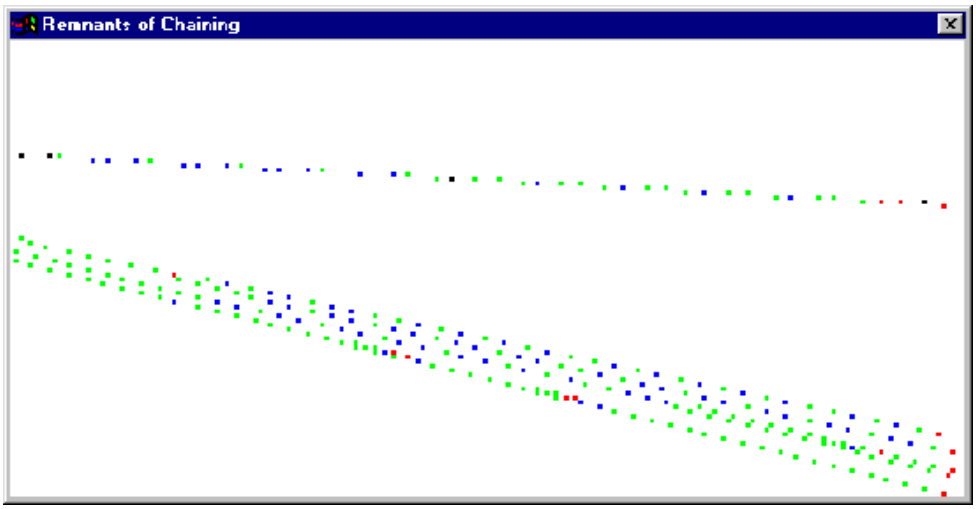


Abbildung 6.11 (b): Bildung von Verkettungen mit Blockade (mit/ohne Ausdünnung)

Ergebnis der Verkettung



Farbinformation bezüglich der Verwendung der Pixel



Legende der Pixelfarben:

- Rot: Verwendet als Anfangspunkt einer Verkettung
- Grün: Verwendet als Verkettungspunkt
- Blau: Blockiert
- Schwarz: Nicht genutzt

Abbildung 6.12: Farbinformationen in Bezug auf die Verwendungsart der Pixel

6.3.4 Modul Splining

Die durch das Modul Verkettung gebildeten Verkettungen bestehen aus einzelnen Verkettungspunkten, die bei Verbindung durch Geraden zackige Linien ergeben. Da in der Realität Fahrbahnmarkierungen glatt und geschwungen

verlaufen, muss diese Kantigkeit durch die bisher durchlaufenen Schritte (Digitalisierung, Binarisierung, Verkettung) erzeugt worden sein. Das Ziel des Moduls Splining ist es, diese Kantigkeit wieder herauszufiltern und die Verkettungen durch einen harmonischen Verlauf zu ersetzen.

Da eine Fahrbahnmarkierung nicht durch ein einziges Polynom dargestellt werden kann, wird der Verlauf zwischen den Verkettungspunkten durch mehrere, zusammengesetzte Polynome dargestellt. Genügen diese einzelnen Polynome weiteren Bedingungen, werden diese in der Mathematik als Splines bezeichnet.

Bei der Auswahl der geeigneten Splines stehen die folgenden Typen zur Verfügung:

- Splines mit vorgegebenen ersten Ableitungen an den Endpunkten
- Splines mit vorgegebenen zweiten Ableitungen an den Endpunkten; sind diese gleich Null spricht man von einem natürlichen Spline
- Splines mit vorgegebenen ersten, zweiten und dritten Ableitungen an den Endpunkten

Über die erste Ableitung kann die Steigung des Splines am Anfang und Ende des Splines vorgegebenen werden. Über die zweite Ableitung kann die Steigungsänderung vorgegebenen werden und mit Vorgabe der dritten Ableitung wiederum die Änderung der Steigungsänderung.

Die Verkettungen aufgrund von Fahrbahnmarkierungen sind wie folgt charakterisiert:

- monoton steigend in Y-Richtung
- keine vorgegebenen Winkel am Anfang und Ende
- tendenziell geradliniger Verlauf

Eine Vorgabe der Steigung (Spline mit vorgegebener erster Ableitung) ist in dieser Arbeit nicht sinnvoll, da nicht bekannt ist, mit welchem Winkel eine Fahrbahnmarkierung beginnt oder endet.

Die Vorgabe der zweiten Ableitung macht dann Sinn, wenn diese zu Null gesetzt wird. Dadurch wird gefordert, dass sich der Spline am Anfang und Ende gedanklich in eine Gerade verlängern ließe. Da die Fahrbahnmarkierungen tendenziell geradlinig verlaufen, ist dieser Ansatz am sinnvollsten.

Aus diesem Grund wird der natürliche Spline verwendet.

Weiterhin wird zwischen Spline-Interpolation und Spline-Approximation unterschieden. Bei der Spline-Interpolation, werden die Stützstellen fest vorgegeben und es werden nur die Zwischenpunkte interpoliert. Da in der vorliegenden Arbeit die Verkettungspunkte durch die vorher durchlaufenen Zwischenschritte eventuell in x- oder y-Richtung Ortsfehler aufweisen, wird in der vorliegenden Arbeit die Spline-Approximation verwendet. Bei der Spline-Approximation müssen die einzelnen Splines nicht exakt durch die vorgege-

benen Stützstellen verlaufen, sondern sich lediglich in möglichst glatter Form an diese anschmiegen.

Eine genaue Übersicht über Splines vermittelt [Engeln & Uhlig 96].

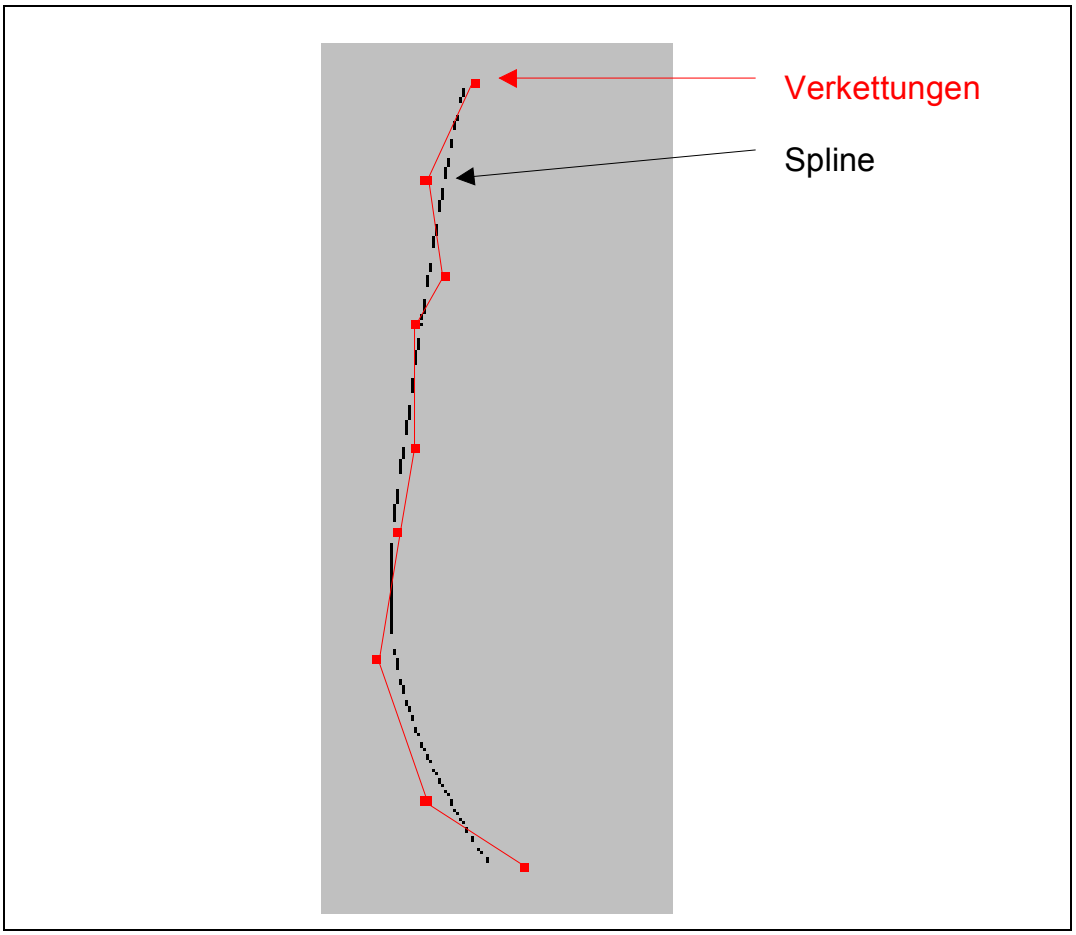


Abbildung 6.13: Umwandlung von Verkettungen in Splines

6.3.5 Zusammenfassung Teilsystem Bildverarbeitung

Wie in der Einleitung dieses Teilsystems erwähnt, ist die Hauptaufgabe an dieser Stelle die Ausgabe von Splines in Echtzeit, die die Fahrbahnmarkierungen der Einzelbilder (als Teil der Bildfolgen) repräsentieren.

Durch die Vorverarbeitung und Nutzung der Verkettungsvektoren wird das System stark auf typische Fahrbahnverläufe geprägt (d.h. es lernt, in welcher Form der Verlauf der Fahrbahnmarkierungen erwartet wird). Der Vorteil dieser Vorgehensweise liegt in einer schnellen Erkennung und Nachbildung der Fahrbahnmarkierungen.

Der Nachteil liegt darin, dass im Falle von stark von der Norm abweichenden Bildinformationen (extrem starke Kurven, extrem starke Kuppen oder Täler,

sehr stark verschmutzte Fahrbahn oder Fahrbahnmarkierungen, keine Fahrbahnmarkierungen, etc.) diese nicht beliebig detailliert analysiert werden und somit zu Fehlinformationen oder fehlenden Informationen führen.

Eine Möglichkeit an dieser Stelle wäre in solchen Situationen ein weiteres Modul dazuzuschalten, welches solche Bilder, wenn nötig, weiter analysiert – dann dafür aber mehr Zeit benötigt. Auch werden bisher die Farben der Bereiche zwischen den Fahrbahnmarkierungen nicht ausgewertet, die Hinweise darüber liefern, auf welcher Seite einer Markierung die Fahrbahn ist. Auch dies könnte in einem weiteren, optional zuschaltbaren, Modul implementiert werden.

In dieser Arbeit wird darauf nicht weiter eingegangen, da die Komponente Bildverarbeitung dazu gedacht ist, die Grundinformationen aus den Bildfolgen zu extrahieren, um den eigentlichen Fokus der Arbeit, nämlich die lernende Steuerung von Fahrzeugen, zu ermöglichen. Daher werden die fehlenden Informationen oder gar Fehlinformationen durch den lernenden Charakter der anderen Teilsysteme aufgefangen.

Mit der in für die vorliegende Arbeit vorhandenen Computerkonfiguration (siehe 6.1) wurden für das Modul Bildverarbeitung folgende Durchlaufzeiten gemessen:

- Bildausschnitt aus einem über den Framegrabber gelieferten Einzelbild: ca. 2 ms
- Vorverarbeitung (Mittelwertbildung und Binarisierung; Kompensierung der perspektivischen Verzerrung, Ausdünnung) ca. 10 ms
- Verkettung und Splining: ca. 2 ms

6.4 Teilsystem Mustererkennung

Um einen Vergleich zwischen zwei parametrischen Szenenbeschreibungen (also z.B. den Splines eines Bildausschnitts) zu ermöglichen, wird ein Speicherformat mit fester Metrik aufgebaut. Dazu werden die Splines, die von Situation zu Situation im Bildausschnitt unterschiedlich oft vorkommen können und dabei auch unterschiedlich lang sein können, mit einem festen Gitter überlagert und es werden die Winkel und die Orte der Schnittpunkte erfasst. Die dadurch gewonnene Darstellungsform der Situation sind die bereits erwähnten ACSD.

Diese ACSD's werden zum einen jeweils in einer Datenbank abgelegt – d.h. eine einmal vorgekommene Situation kann zu späteren Zeitpunkten identifiziert werden.

Zum anderen werden aktuelle ACSD und ACSD's aus der Datenbank an einen Algorithmus übergeben, der zwischen diesen eine Ähnlichkeitsbestimmung durchführt. Der dazu verwendete Mustererkennungsalgorithmus ist

ANN (Approximate Nearest Neighbour) [Mount 98]; eine weiterentwickelte Version des Nearest Neighbour Algorithmus.

Im Detail werden die ACSD's durch die Überlagerung eines Gitters mit den Splines gebildet. Das Gitter besteht aus den Gitterpunkten und den dazwischen liegenden Gitterschenkeln. Sofern der Schnittpunkt nicht direkt auf einem Gitterkreuz zustande kommt, werden statt den exakten Koordinaten des Schnittpunkts die Koordinaten des nächst-kleineren Gitterkreuzes vermerkt. Über diese Vorgehensweise wird eine Quantisierung der Schnittpunkte erreicht. Zusätzlich wird der Winkel des Splines im Schnittpunkt erfasst und ebenso vermerkt. Jeder Gitterpunkt des Gitters entspricht einem Element im ACSD und in diesem Element wird der Schnittwinkel vermerkt.

Im Falle eines Gitters mit i Zeilen und j Spalten erhält man ein ACSD mit $i \bullet j$ Elementen. Da die Winkel als DWORD abgespeichert werden, benötigt ein Gitter mit 20 Zeilen und 40 Spalten in Summe 3200 Bytes. Bei einem Systemtakt von 100 ms, d.h. 10 Bildern pro Sekunde ergibt dies ein Volumen von 1,92 MByte/Minute, bzw. 115,2 MByte/Stunde.

Diese Datenmenge würde erreicht, wenn tatsächlich jede ACSD in der Datenbank gespeichert würde. Allerdings wird vor jedem Speichern überprüft, ob eine ähnliche ACSD bereits in der Datenbank vorhanden ist und in diesem Fall ein erneutes Speichern vermieden. Die Anzahl der ACSD in der Datenbank wächst deshalb nicht linear an, sondern flacht je nach Varianz der Situationen sowie der Ähnlichkeitskriterien über die Zeit ab. Abbildung 6.14 zeigt eine solche Entwicklung über die Zeit. Aufgrund des festen Speicherbedarfs für eine ACSD nimmt auch die kumulierte Größe der ACSD-Datenbank nicht linear zu sondern flacht über die Zeit ab.

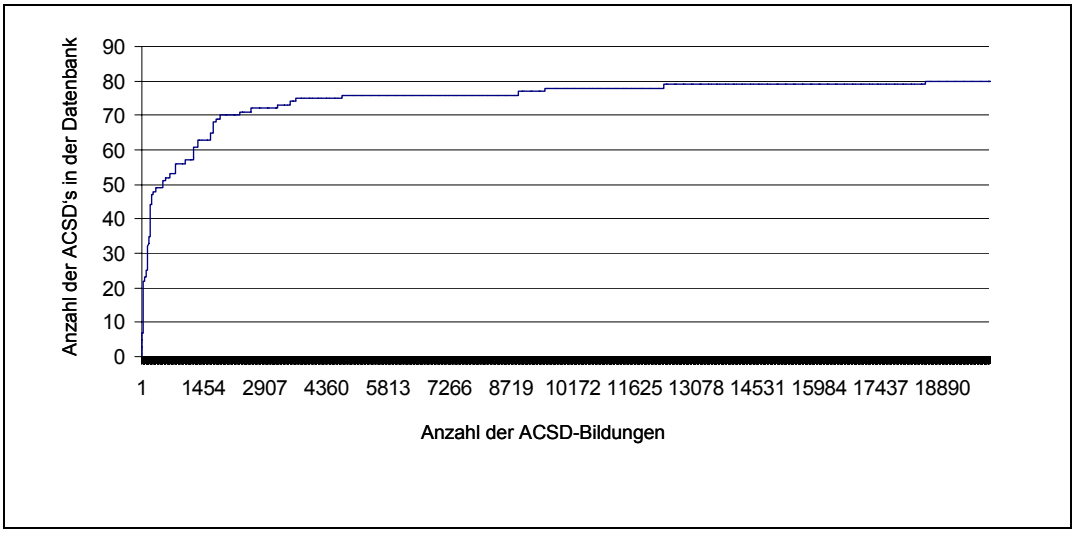


Abbildung 6.14: Verlauf der Anzahl von ACSD-Einträge über die Zeit

Die Größe des Gitters bei der ACSD-Bildung hat eine unterschiedliche Wirkung. Je kleiner das Gitter ist, desto genauer ist die Nachbildung der ursprünglichen Fahrbahnmarkierungen aber eine zu hohe Genauigkeit kann

kontraproduktiv bei der Identifikation ähnlicher Situationen sein (aufgrund von unterschiedlichem Rauschen oder leichtem Versatz in den Bildern).

In Abbildung 6.15 wird der Ausschnitt einer ACSD-Bildung dargestellt. In den Teil-Bildern links oben und links unten wird ein Teil einer parametrischen Szenenbeschreibung (d.h. ein Teil eines Splines) mit zwei unterschiedlichen Gittern überlagert. Die sich ergebenden quantisierten Schnittpunkte und deren Winkel sind in blau eingezeichnet.

In den beiden Abbildungen rechts oben und unten, wird das gleiche Spline-Teilstück – diesmal aber leicht seitlich versetzt, mit dem gleichen Gitter überlagert. Im oberen Falle ergibt sich der gleiche ACSD-Eintrag – im der unteren Abbildung verschiedene ACSD-Einträgen.

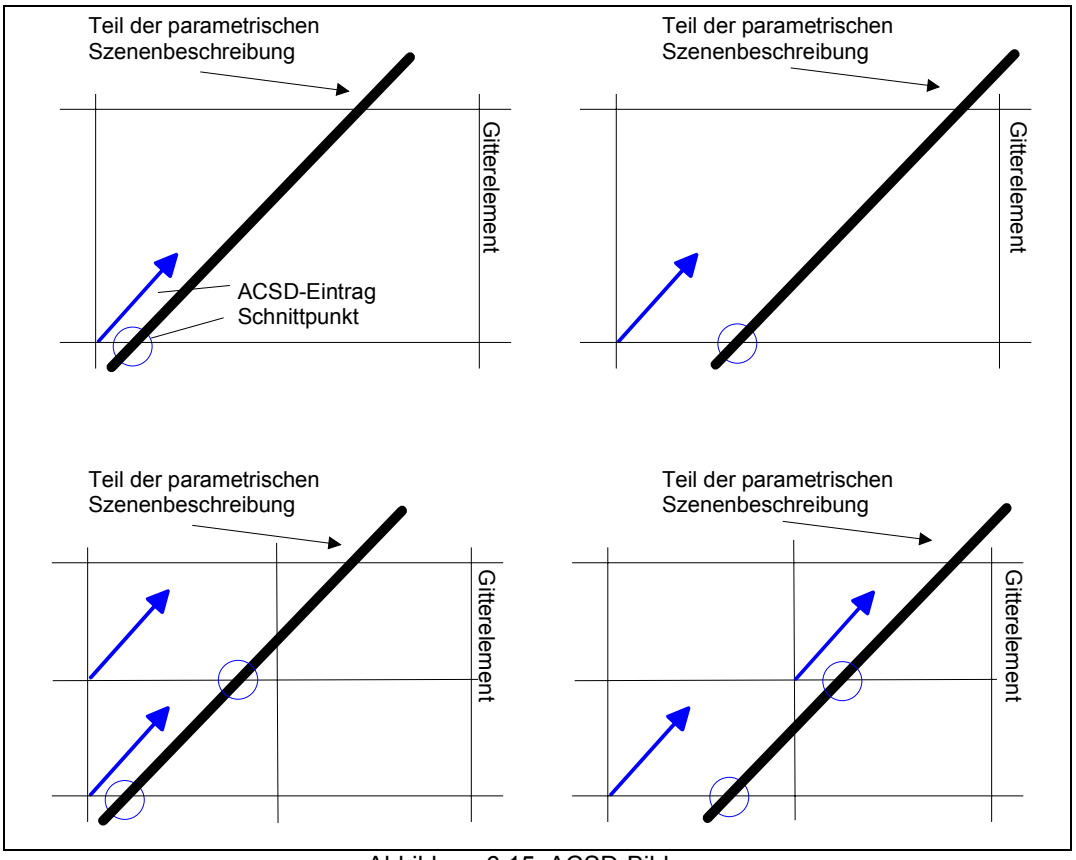


Abbildung 6.15: ACSD-Bildung

6.4.1 Experimentelle Ergebnisse der Mustererkennung

In der vorliegenden Arbeit erfolgt die Ähnlichkeitsbestimmung in zwei Durchläufen– dabei werden von jeder parametrischen Szenenbeschreibung zwei ACSD's erstellt. Eine ACSD aufgrund eines gröberen Gitters für den ersten Durchlauf und eine ACSD aufgrund eines feineren Gitters für einen zweiten Durchlauf. Nach dem ersten Durchlauf werden nur diejenigen ACSD's für den

zweiten Durchlauf qualifiziert, die einem ersten Ähnlichkeitskriterium genügen. D.h. es wird zunächst eine Grobsondierung und dann eine Feinsondierung der ähnlichsten Situation durchgeführt. In experimentellen Testreihen ergab sich, dass unter den 30-50 ersten Treffern der Grobsondierung die höchste Ausbeute an ähnlichen Situationen vorlag. Somit wurde für die meisten Testreihen die Anzahl an qualifizierten ACSD für die Feinsondierung auf 40 beschränkt.

Abbildung 6.16 zeigt zwei ACSD-Berechnungen in Bezug zum ursprünglichen Einzelbild. Da während der Bildverarbeitung eine Ausschnittsbildung sowie eine seitliche Stauchung des Bildes durchgeführt werden, haben sich die äußeren Abmaße der ACSD gegenüber dem Eingangsbild verringert.

In Abbildung 6.17 sind ebenso zwei ACSD-Berechnungen dargestellt, die sich durch die Wahl des Gitterabstandes unterschieden. Im oberen Fall ist das Gitter feiner und dadurch ergibt sich eine Vielzahl von Schnittpunkten zwischen Raster und parametrischer Szenenbeschreibung. Im unteren Fall ist das Gitter gröber und die Anzahl der Schnittpunkte ist in Konsequenz geringer.

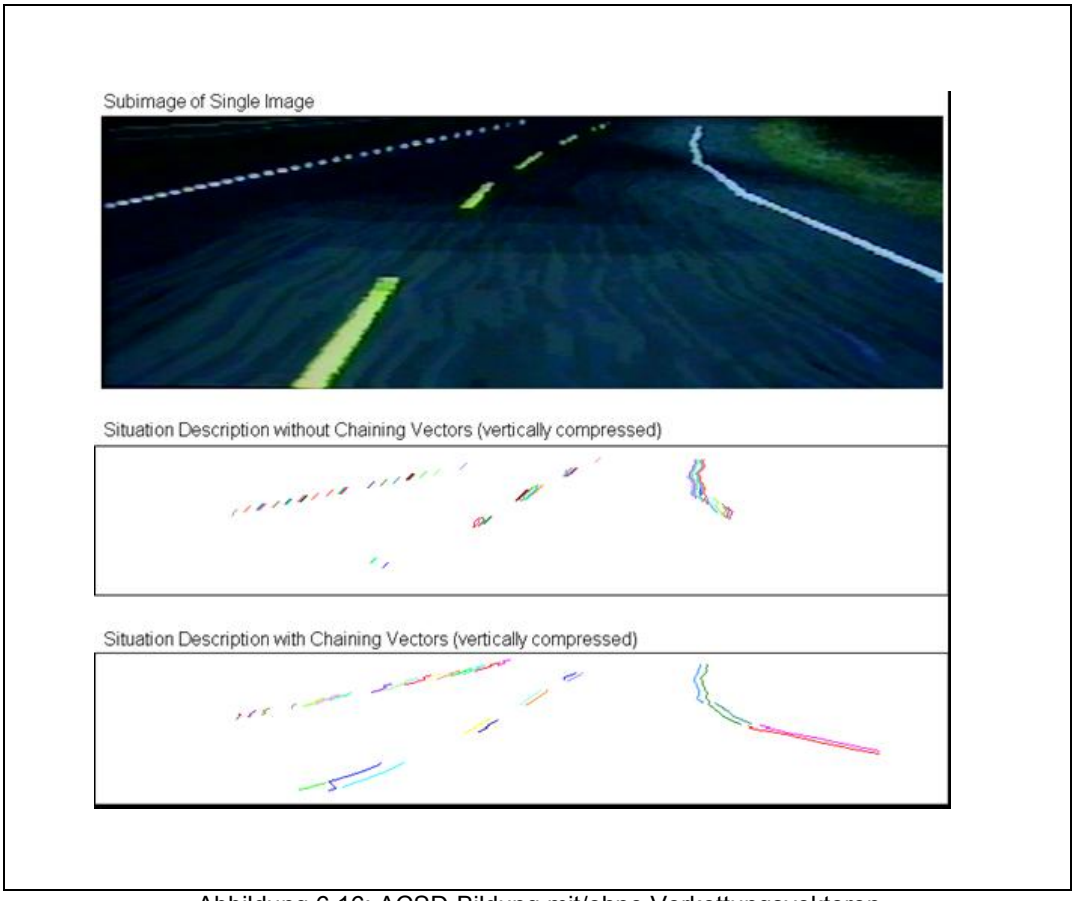


Abbildung 6.16: ACSD-Bildung mit/ohne Verkettungsvektoren

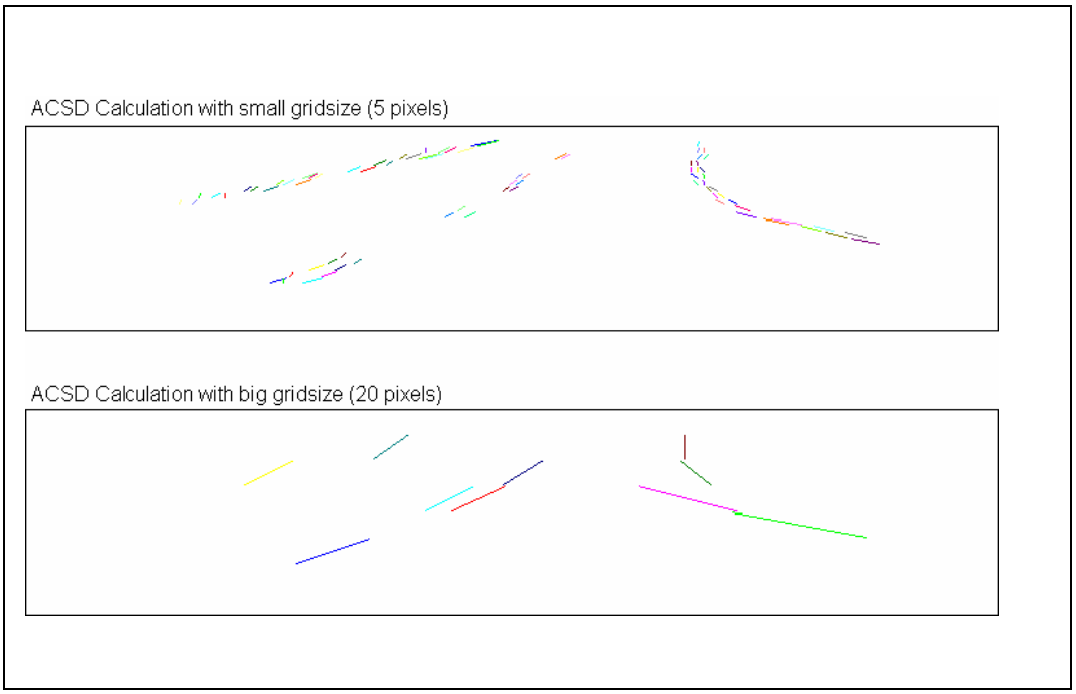


Abbildung 6.17: ACSD-Bildung mit kleinerem/größerem Raster

Die folgenden Abbildungen zeigen das Ergebnis eines Durchlaufs, bei dem aus einer Datenbank von ca. 10.000 ACSD's die 8 ähnlichsten Nachbarn zur ACSD eines neuen Eingangsbildes selektiert werden. In Abbildung 4.16 ist das Eingangsbild dargestellt – in diesem Falle eine gestreckte Rechtskurve. Die Datenbank umfasst dabei die ACSD's von unterschiedlichen Streckentypen, d.h. rangiert von starker Linkskurve bis hin zu starker Rechtskurve.

In den Abbildungen 6.19 sind die 8 besten Treffer dargestellt; dabei wurden durch den ANN-Algorithmus die ACSD's identifiziert. Die hier gezeigten Bilder sind die damit verbundenen Originalbilder.

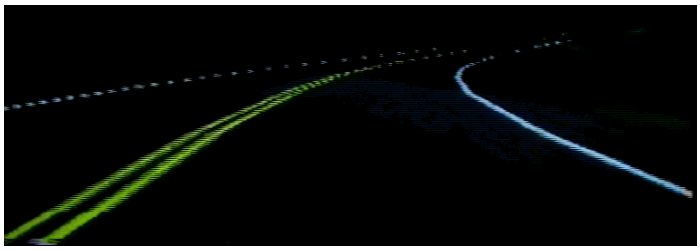


Abbildung 6.18: Streckenabschnitt als Eingangsbild

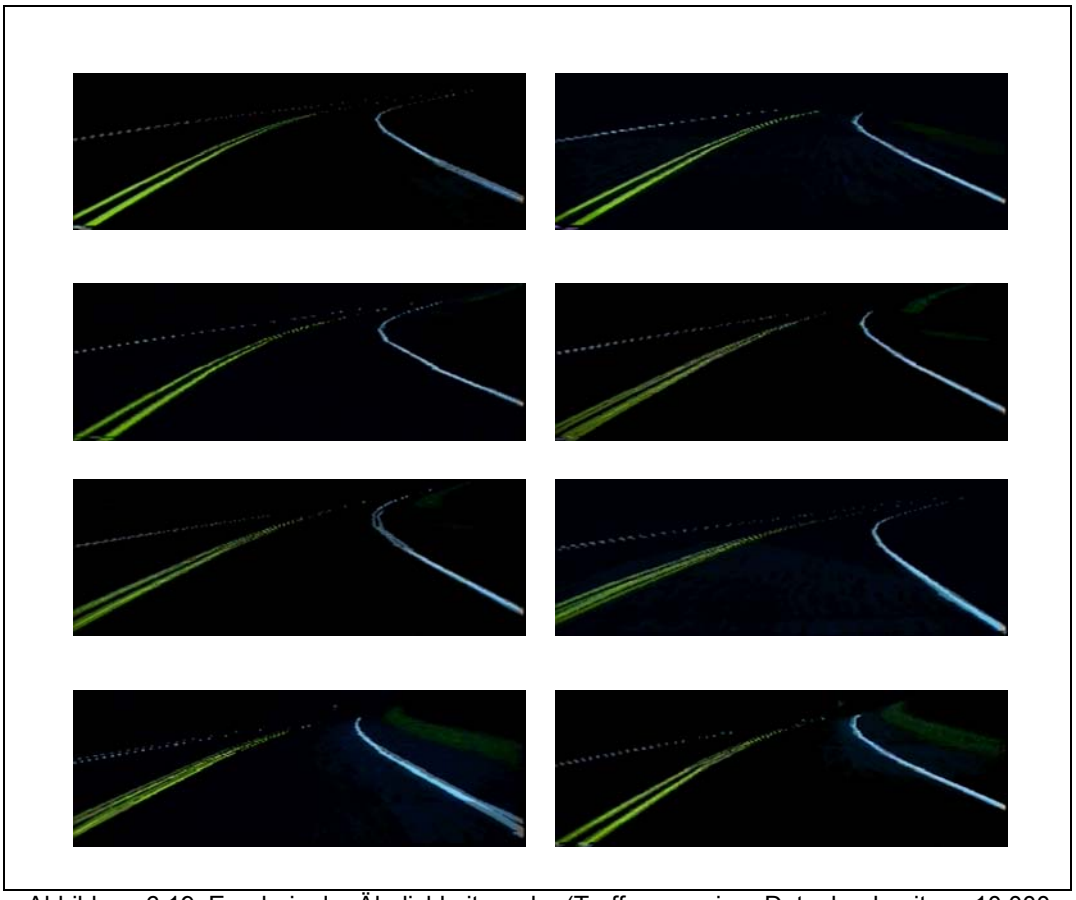


Abbildung 6.19: Ergebnis der Ähnlichkeitssuche (Treffer aus einer Datenbank mit ca. 10.000 unterschiedlichen Streckenabschnitten)

Die Ähnlichkeit der 8 Treffer im Vergleich zum Eingangsbild ist deutlich. Mit entsprechenden Testreihen wurde folgende Vertauschungsmatrix ermittelt:

	Kurventyp Eingangsbild					
Kurventyp Nächster Nachbar		Stark Links	Links	Geradeaus	Rechts	Stark Rechts
	Stark Links	369	180	23	8	2
	Links	158	2754	285	18	2
	Geradeaus	24	252	1530	234	0
	Rechts	7	16	199	1654	63
	Stark Rechts	4	7	3	47	196

Tabelle 6.2: Vertauschungsmatrix

6.4.2 Fahren aufgrund Mustererkennung

Um erste Fahreigenschaften zu implementieren, wurde zunächst eine Lernphase durchgeführt, in der jedem Bildmuster jeweils eine Aktion zugeordnet wurde. Diese Aktion war dabei die jeweils optimale Aktion, d.h. die beste Aktion in Bezug auf Situationsangemessenheit. Nach dem Durchführen einer solchen Lernphase wurde das System angewiesen, bei Identifikation einer Situation die eingangs hinterlegte Aktion auszugeben. Im Rahmen dieser Tests konnte ein akzeptables Fahrverhalten nachgewiesen werden, welches lediglich auf den Komponenten Bildverarbeitung und Mustererkennung beruht.

Unter anderem wurden die Fahreigenschaften in einer Testkonfiguration mit 5.400 ACSD Einträgen und einer Systemtaktung von 100 ms durchgeführt, in dem der System-PC die Aufgabe hat, ein Fahrzeug entlang eines kurvigen und hügeligen Fahrbahnverlaufs zu steuern. Dabei war der Fahrbahnverlauf teilweise unbekannt, daher es wurden die ACSD's auf Basis ähnlicher, nicht aber identischer Situationen gebildet.

Die wichtigsten Ergebnisse einer exemplarischen Testreihe ergaben sich wie folgt:

- Zeitdauer des autonomen Fahrzeugsteuerung: 300 Sekunden (5 Minuten)
- Anzahl ausgegebener Steuerkommandos: 3000 (10 pro Sekunde)
- Anzahl signifikanter Fehler (Fahrbahn komplett verlassen): 1
- Anzahl allgemeiner Fehlern (z.B. äußere Fahrbahnmarkierung überschritten): 4

Wenn angenommen wird, das jeder dieser insgesamt 5 Fehler nicht nur durch das letzte Steuerkommando, sondern aus einer Kombination der letzten 10 Steuerkommando resultiert, liegt die Fehlerrate bei $(5 \cdot 10) / 3000 = 1.6$ %. Die Tatsache, dass diese Fahreigenschaften rein auf Basis Bildauswertung und Mustererkennung, d.h. ohne weitere Lernfähigkeit des Systems implementiert wurden, zeigt die Mächtigkeit von Mustererkennung als Kernelement autonomer Fahrzeugsteuerungen.

6.4.3 Zusammenfassung Teilsystem Mustererkennung

Zunächst wandelt das Teilsystem Mustererkennung die parametrischen Szenenbeschreibungen in Situationsbeschreibungen fester Metrik als Speicherformat um. Die Forderung nach fester Metrik resultiert aus einem nachfolgenden Algorithmus zur Bestimmung der Ähnlichkeit von Situationsbeschreibungen.

Diese Situationsbeschreibungen werden als ACSD's (Abstract Complete Situation Description) bezeichnet. Nach der Generierung wird jede ACSD in einer ACSD-Datenbank gespeichert, sofern dort nicht schon eine ähnliche ACSD vorliegt. Die Ähnlichkeitskriterien wurden in den Testreihen so gewählt, dass

sich langfristig eine Anzahl von etwa 1.000 ACSD-Einträgen in der ACSD-Datenbank aufbaut.

Für die Ähnlichkeitsbestimmung wurde ein 2-stufiger Prozess implementiert, d.h. zunächst wird eine Grobanalyse bezüglich der Ähnlichkeit durchgeführt und nur eine feste Anzahl der besten Treffer wird für eine Feinanalyse verwendet. Über diese Vorgehensweise wurde das Ziel erreicht, ähnliche Situationen einer Datenbank im Vergleich zu einer aktuellen Situation zu identifizieren.

Die Durchlaufzeiten für ein System mit ca. 1.000 ACSD-Einträgen bewegen sich dabei für das Teilsystem Mustererkennung im Bereich weniger Millisekunden.

Zusätzlich wurden durch die Kombination von Bildauswertung und Mustererkennung bereits erste Fahreigenschaften ermöglicht.

6.5 Verstärkungslernen

Im Folgenden werden die experimentellen Ergebnisse der Implementierung des Verstärkungslernen-Systems dargestellt. Es werden zunächst die drei unter Kapitel 5.3 dargestellten Szenarien untersucht und die Ergebnisse der Konvergenz diskutiert. Dabei wird nachgewiesen, dass sich das Verstärkungslernen-System automatisch an die unterschiedlichen Umgebungen anpasst und komplett selbständig die Charakteristika der jeweiligen Umgebung erforscht.

Im Rahmen der Testserien hat sich herausgestellt, dass eine besondere Schwierigkeit darin liegt, den Grad der Konvergenz der Situationsbewertungen, bzw. den Verlauf der Konvergenz über die Zeit zu bestimmen. Deshalb befasst sich ein weiterer Teil dieses Kapitel mit dieser Thematik und der in dieser Arbeit entwickelten Lösung.

Die vorliegende Arbeit erforscht weiterhin die Auswirkungen der Verstärkungslernen Parameter (d.h. α, γ, ε und λ). Diese Ergebnisse werden ebenfalls im Detail dargestellt.

6.5.1 Generelle Konvergenz der 3 Szenarien

Das in dieser Arbeit verwendete Lernsystem auf Basis Verstärkungslernen bewertet wie in den Kapiteln 4.1 und 4.2 beschrieben, die situationsspezifischen Aktionen in Form von Situationsbewertungen $Q(s,a)$. Im Detail spannen Zustände (d.h. die Situationsbeschreibungen), Aktionen und Situationsbewertungen einen Verhaltensraum auf und der Verlauf der Q-Werte wird auch als Q-Funktion bezeichnet. Anzustrebendes Ziel eines jeden Verstärkungslernen-Systems ist die Konvergenz der Situationsbewertungen, d.h. die Konvergenz

der Q-Funktion. Hintergrund ist, dass die Q-Funktion die Basis der Aktionsselektion ist, da wie in Kapitel 4.2 die Q-Funktion die diskontierte Summe aller zukünftigen Belohnungen darstellt und ein Verstärkungslernen-System nach Belohnungsmaximierung strebt.

Unter Kapitel 5.3 sind die 3 unterschiedlichen, zu lernenden Szenarien und die Erwartung an den graphischen Verlauf der Q-Funktion im Verhaltensraum dargestellt. Aufgrund der Implementierung des Verstärkungslernen-Systems können alle diese Erwartungen an die Konvergenz im Verhaltensraum experimentell nachgewiesen werden.

Abbildung 6.20 zeigt einen größtenteils konvergierten Verlauf der Q-Funktion im Verhaltensraum des Szenarios a (Lerne die Seitenposition). Der Verhaltensraum wird dabei über die möglichen Aktionen jeder Situation (von links nach rechts), von den Situationen (von hinten nach vorne) und den Q-Werten jeder Situation-Aktion-Kombination aufgespannt.

Zur Verdeutlichung der Situationen sind diese auf der linken Seite aufgetragen. Dabei entspricht die oberste Situation, bzw. im dreidimensionalen Raum die hinterste Situation, dem Zustand, dass sich das Fahrzeug am linken Fahrbahnrand befindet. Die unterste Situation, bzw. im dreidimensionalen Raum die vorderste Situation, entspricht dem Zustand, dass sich das Fahrzeug am rechten Fahrbahnrand befindet. Zwischen diesen beiden extremen Situationen verläuft der Übergang gleitend.

Jeder Situation ist wie eingangs dargestellt ein Belohnungswert zugeordnet. Diese sind ebenso in Abbildung 6.20 aufgeführt.

Ebenso ist der Verlauf des Fehlerwertes $TDerr$ über der Zeit dargestellt. Dabei ist die Konvergenz der Q-Funktion im Verhaltensraum im Verlauf von $TDerr$ nicht zu erkennen – mit diesem Aspekt befasst sich Kapitel 6.5.2.

Das Ergebnis zeigt, dass eine deutlich erkennbare Konvergenz eingesetzt hat. Dabei wurde dem Verstärkungslernen-System außer den Belohnungen keinerlei Information über die Umgebung übermittelt – somit wurde die Konvergenz der Q-Funktion im Verhaltensraum tatsächlich komplett autonom durchgeführt.

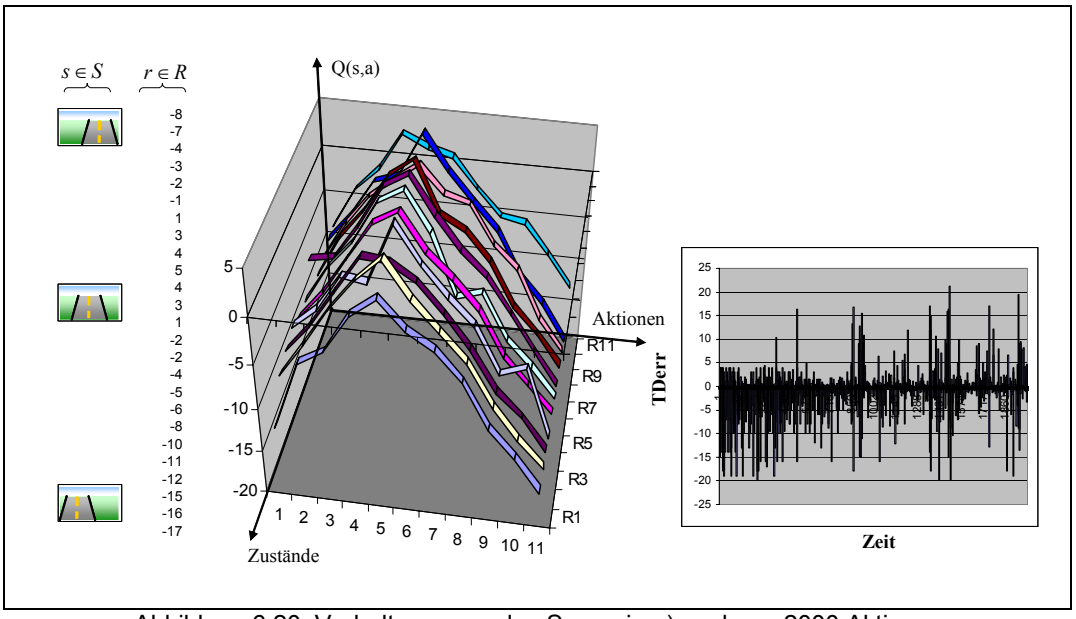


Abbildung 6.20: Verhaltensraum des Szenario a) nach ca. 2000 Aktionen

Abbildung 6.21 zeigt die Weiterführung der Testserie von Abbildung 6.20. Dabei ist die Verbesserung der Konvergenz ersichtlich (da sich das System extrem deterministisch verhält nimmt auch der Wert von *TDerr* über der Zeit ab).

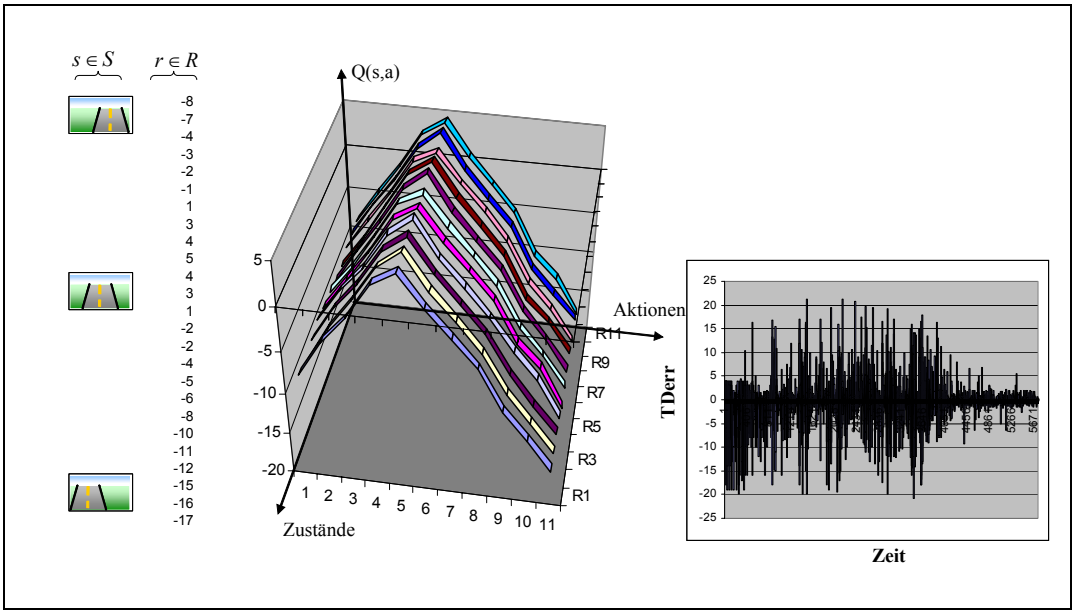


Abbildung 6.21: Verhaltensraum des Szenario a) nach ca. 6.000 Aktionen

Abbildung 6.22 zeigt eine größtenteils konvergierte Q-Funktion im Verhaltensraum des Szenarios b (Lerne den Fahrwinkel). Gemäß der in Kapitel 5.3 dargestellten Erwartung prägt sich eine sattelförmige Konvergenz der Q-Funktion aus, wobei das jeweilige Maximum der situationsspezifischen Aktion über die Menge der Aktionen wandert. Dabei wurde dem System eine einfache Umge-

bung zugrunde gelegt, die in 25 unterschiedliche Situationen klassifiziert werden kann.

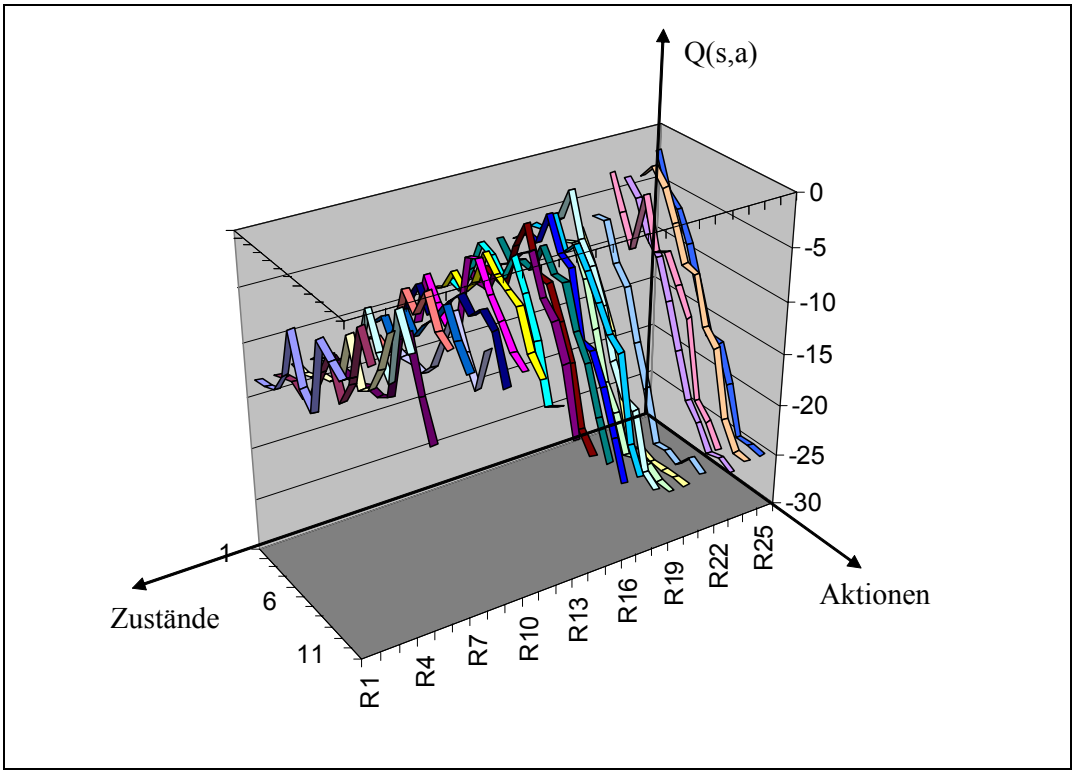


Abbildung 6.22: Verhaltensraum des Szenario b) nach ca. 25.000 Aktionen

Abbildung 6.23 zeigt ebenso den Verhaltensraum des Szenarios b) – diesmal aber auf Basis einer komplexeren Umgebung. Aus dieser wurden durch das System mehr als 200 unterschiedliche Situationsbeschreibungen extrahiert, die als Basis für den Verhaltensraum genutzt werden. Trotz der deutlich erhöhten Komplexität zeichnet sich auch in diesem Fall wieder eine sattelförmige Konvergenz mit wanderndem Maximum der Situationsbewertungen über die Aktionen.

Letztlich zeigt Abbildung 6.24 die konvergierte Q-Funktion im Verhaltensraum des Szenarios c (Lerne den Lenkwinkel). Es wurde bereits in Kapitel 5.3 dargestellt, dass der Verhaltensraum auch vom aktuellen Fahrwinkel abhängig ist. Deshalb wurde in der vorliegenden Arbeit eine Unterteilung in 7 mögliche Fahrwinkel vorgenommen. Die Klassifikation „3“ entspricht dabei einer Geradeausfahrt und die Klassifikationen „0“ und „6“ einem ausgeprägten Fahrwinkel nach links, bzw. nach rechts. Die anderen Klassifikationen sind zwischen diesen Vorgaben gleichmäßig verteilt. Dabei wurde die Testreihe von Abbildung 6.24 auf einem System durchgeführt, auf dem Simulator und Lernsystem auf einem PC implementiert wurden - dadurch wurden für diese Testreihe mögliche Signalverzögerungen vermieden.

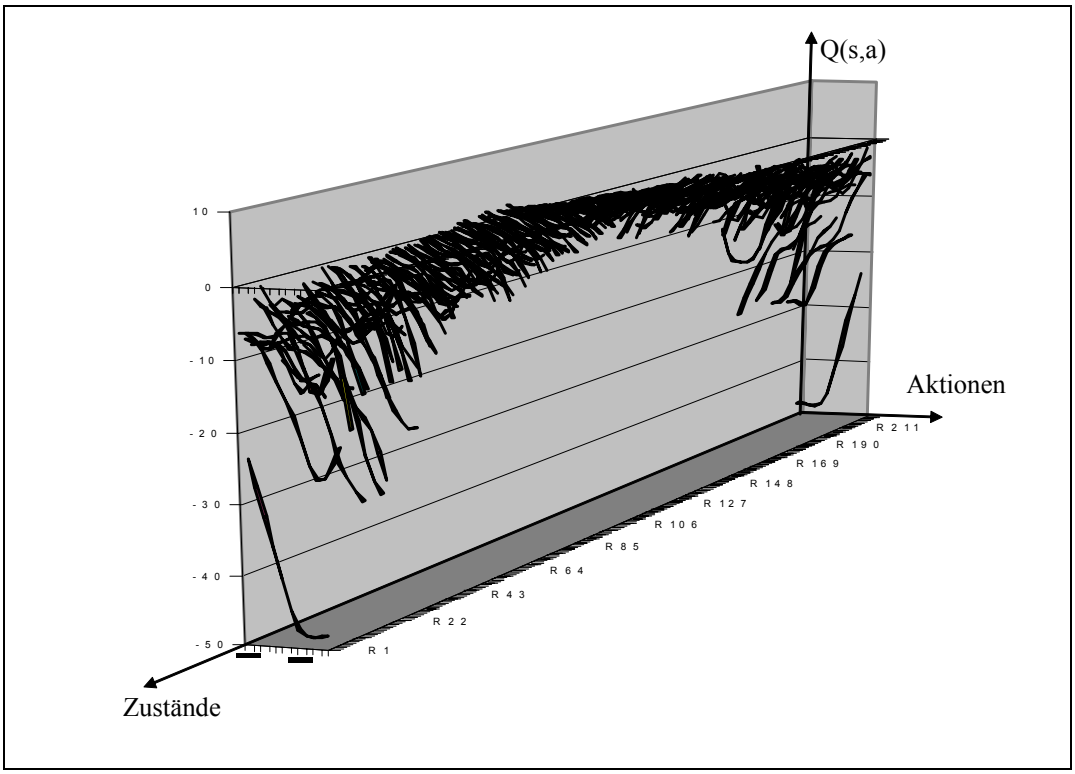


Abbildung 6.23: Verhaltensraum des Szenario b) nach ca. 23.000 Aktionen

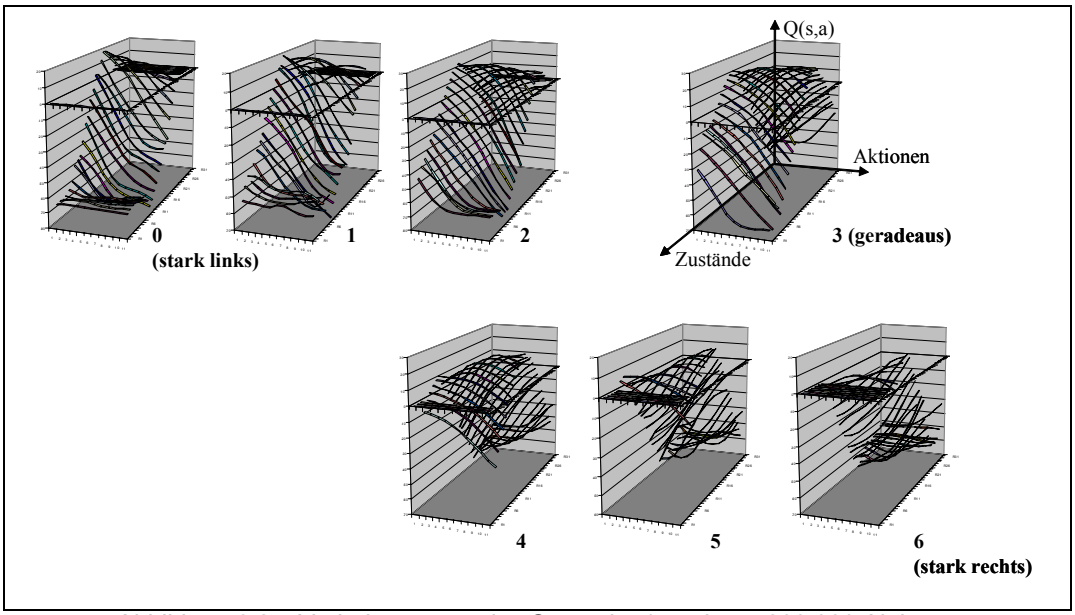


Abbildung 6.24: Verhaltensraum des Szenario c) nach ca. 200.000 Aktionen

Letztlich konnte mit den beschriebenen Testreihen nachgewiesen werden, dass die Vorhersagen aus Kapitel 5.3 richtig sind. In keinem der Fälle wurde ein anderer oder zusätzlicher Input außer den Belohnungen zur Verfügung gestellt. In allen Fällen wurde der Verhaltensraum autonom erkundet – d.h. es fand in keinem der Beispiele eine Instruktion oder Modellierung statt.

Damit werden Leistungsfähigkeit und vor allem Flexibilität eines Verstärkungslernen-Systems unterstrichen.

6.5.2 Glattheitsmaß

Im Laufe der Testreihen wurde der Bedarf an ein einfaches Konvergenzkriterium deutlich. Der Fehlerwert *TDerr* kann dazu nicht verwendet werden, da es sich gezeigt hat, dass dessen Wert nur in komplett deterministischen Umgebungen zurückgeht (siehe Abbildung 6.21). Aber selbst in solch komplett deterministischen Umgebungen (d.h. Umgebungen, deren situationsspezifische Belohnungen weder verzögert noch gestört sind) tritt dieser Effekt erst nach einiger Zeit ein – ein allmähliches Erreichen der Konvergenz im Laufe der Testserie ist nicht zu erkennen.

Gravierender ist der Nachteil, dass in komplexeren Umgebungen, deren Verhalten entweder nicht komplett deterministisch (d.h. Belohnungswerte sind gestört oder verzögert) oder deren Situationen aufgrund von natürlichem Rauschen in unterschiedliche ACSD klassifiziert werden, ein Abschwellen des Fehlerwerts *TDerr* nicht ersichtlich ist; auch nicht, wenn aus der graphischen Darstellung des Verhaltensraums eine ansatzweise Konvergenz ersichtlich ist.

Aus diesem Grunde werden die Q-Werte des Verhaltensraums nach jedem Update über die jeweilige Situation mittels eines kubischen Splines approximiert. Durch das Splining wird der Verlauf der Q-Werte des Verhaltensraums geglättet – ein Prozess, der ebenso im Laufe der Konvergenz erreicht wird. Zwischen den originalen Q-Werten und der Spline-Approximation wird der kumulierte quadratische Abstand bestimmt und in Zukunft als Glattheitsmaß *G(Q)* (Smoothness) bezeichnet.

$$G(Q) = \sum_{s \in S} \sum_{a \in A} (Q(s,a) - Q_{Spline}(s,a))^2 \tag{6.8}$$

In Abbildung 6.25 ist die Q-Funktion vor und nach der Approximation gezeigt. Ebenso der Verlauf des Fehlerwerts *TDerr* und des Glattheitsmaßes über der Zeit. Während aus den graphischen Darstellungen der Verhaltensräume sehr deutlich ein hoher Grad an Konvergenz der Situationsbewertungen ersichtlich ist, ist der Verlauf des Fehlerwertes *TDerr* wenig aufschlussreich. Im Gegensatz dazu kann über den Verlauf des Glattheitsmaßes ein Urteil über den Grad der Konvergenz geschlossen werden, da für die vorliegenden Verhaltenräume angenommen wird, dass die jeweilige Q-Funktion glatt ist.

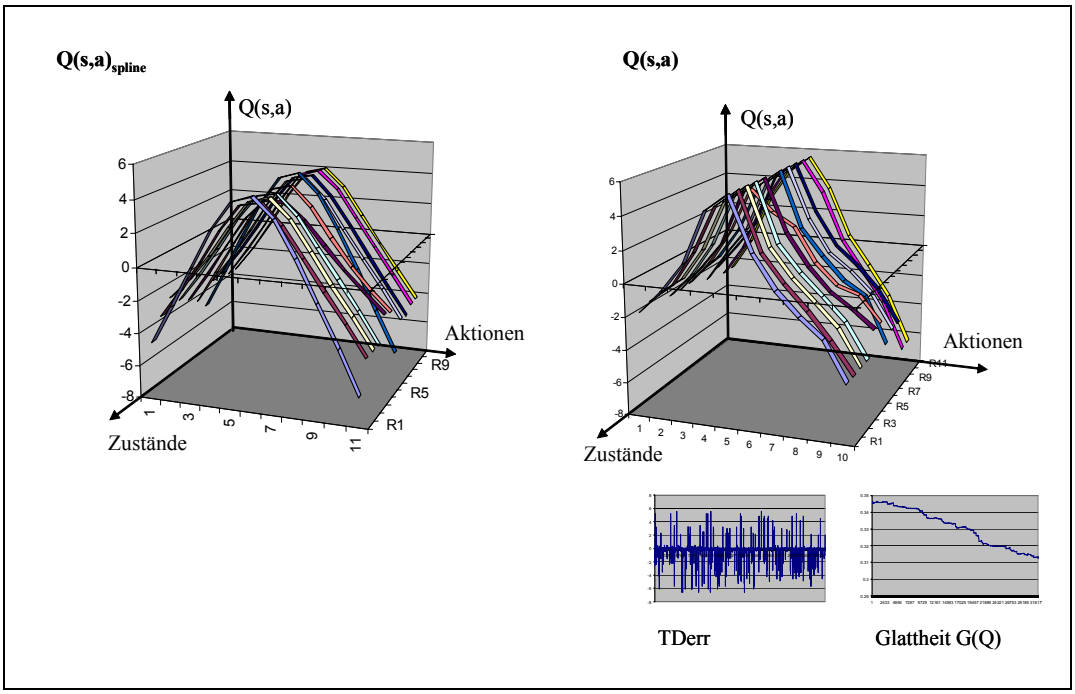


Abbildung 6.25: Fehlerwert und Glattheitsmaß bei hoher Erkundung (Szenario a)

Grundvoraussetzung für das Konvergieren der Situationsbewertungen ist dabei eine ausreichende Erkundung des Verhaltensraums. In Abbildung 6.26 ist das Ergebnis einer Testreihe mit sehr geringer Erkundung des Verhaltensraums gezeigt. Dabei haben sich die jeweils besten Aktionen einer Situation (d.h. die lokalen Maxima der Situationsbewertungen) bereits stabilisiert – die anderen Aktionen der Situationen wurden aufgrund der geringen Erkundung jedoch nicht oft genug ausgewählt, um von einem konvergierten Q-Wert sprechen zu können. Durch diese deutlich unterschiedliche Erkundung des Verhaltensraums ist auch eine stark unterschiedliche Konvergenz der unterschiedlichen Situations-Aktions-Kombinationen entstanden. Das gleichmäßige Splining des Verhaltensraums verfälscht die graphische Darstellung – allerdings zeigt auch der Verlauf des Glattheitsmaßes die fehlende Konvergenz.

Das Thema der Erkundung und die Auswirkungen auf die Konvergenz wird in Kapitel 6.5.3 dargestellt.

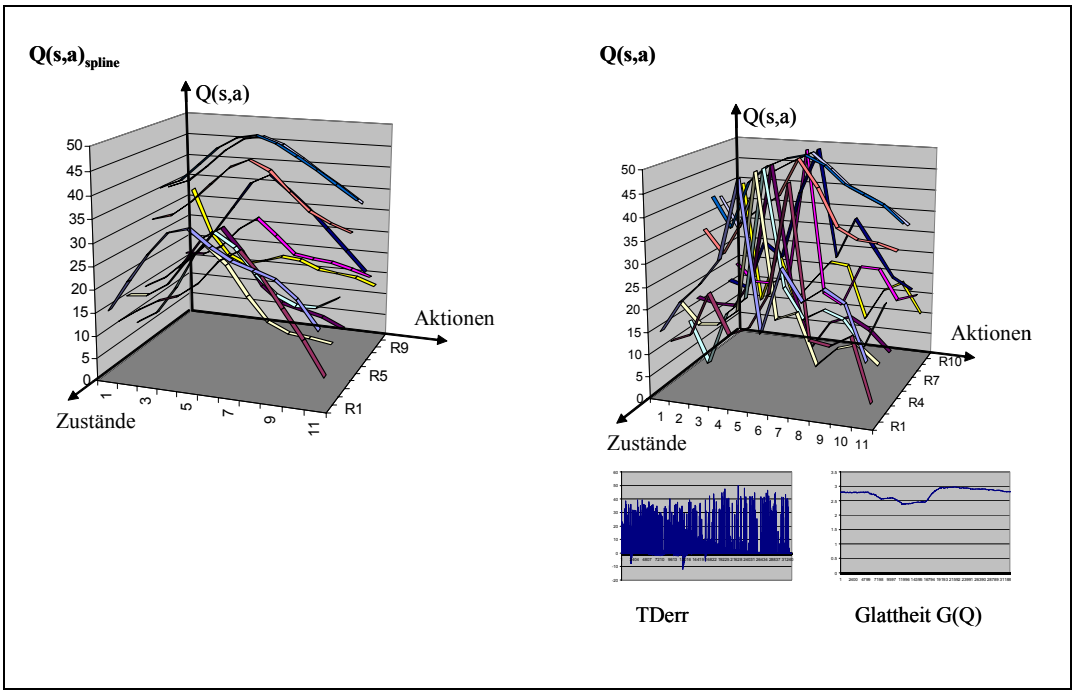


Abbildung 6.26: Fehlerwert und Glattheitsmaß bei niedriger Erkundung (Szenario a)

In Abbildung 6.27 wird auch wieder eine positiver Verlauf der Konvergenz dargestellt, diesmal wieder für das Szenario b (Lerne den optimalen Fahrwinkel).

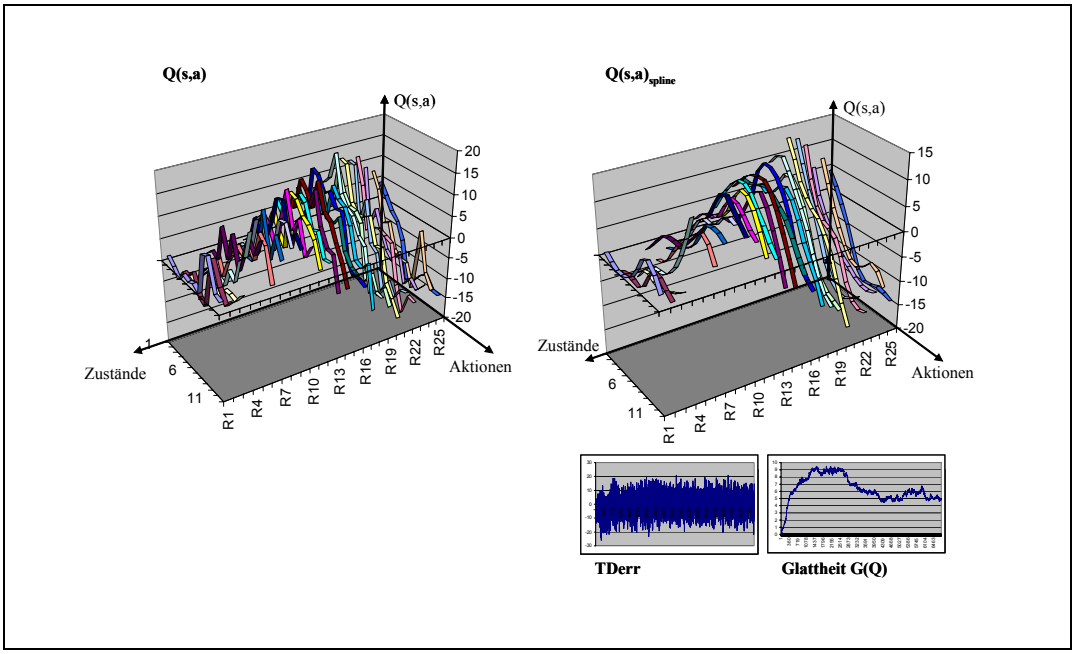


Abbildung 6.27: Fehlerwert und Glattheitsmaß (Szenario b)

6.5.3 Auswirkung des Erkundungsparameters ε

In Kapitel 4.2.4 ist der mathematische Hintergrund zu Erkundung und Datennutzung beschrieben. Dabei wird durch den Parameter ε der Grad der Erkundung festgelegt.

Abbildung 6.28 zeigt zwei Verhaltensräume von Testserien mit jeweils 2.000 Aktionen des Szenario a (Lerne die optimale Seitenposition). Im linken Teil der Abbildung wurde die Erkundung komplett ausgeschaltet – das System findet relativ schnell eine (sub-)optimale Aktion und führt eine Konvergenz einzig und allein für diese Aktion durch. Im rechten Teil der Abbildung hingegen führt eine starke Erkundung zur guten Erforschung des Verhaltensraumes.

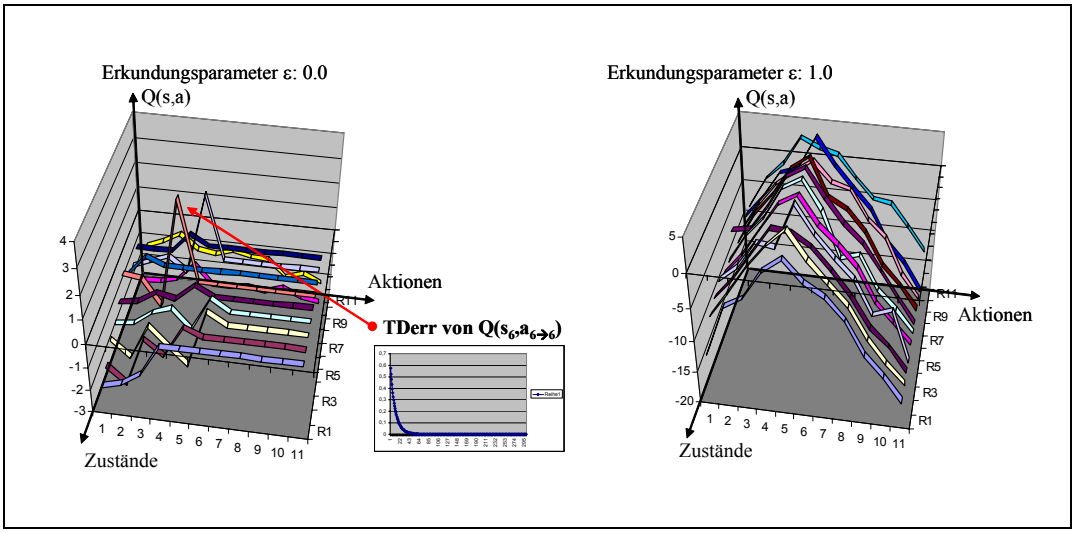


Abbildung 6.28: Erkundung versus Datennutzung (Szenario a)

Ein ähnliches Bild ergibt sich für die in Abbildung 6.29 dargestellte Testreihe des Szenario b). Eine geringe Erkundung (ε im Bereich von 0.1 und 0.3) führt zur Erkundung des Verhaltensraums um die optimale Situation herum– dieser Bereich ist im rechten unteren Teil der Abbildung nochmals vergrößert hervorgehoben – die anderen Situationen werden aber schlecht, bzw. teilweise überhaupt nicht erkundet.

Ähnlich verhält es sich bei komplexeren Umgebungen. Abbildung 6.30 zeigt das Ergebnis einer Testreihe mit über 100 unterschiedlichen Situationen. Auf der linken Seite der Abbildung ist der Verhaltensraum einer Testserie mit ε gleich 0,1 dargestellt – aus Gründen der Übersichtlichkeit ist ein Extrakt mit nur jeder 10.ten Situation eingeblendet. Auf der rechten Seite der Abbildung ist das Ergebnis der gleichen Testreihe, lediglich mit $\varepsilon = 1{,}0$ abgebildet.

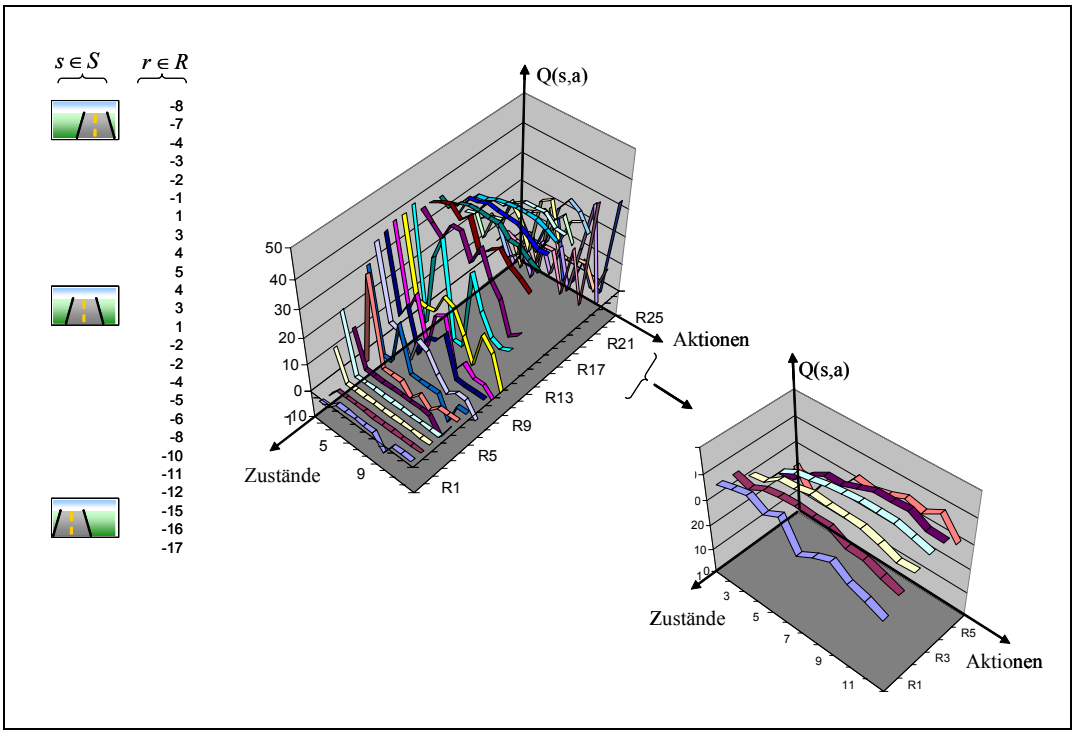


Abbildung 6.29: Geringe Erkundung (Szenario b)

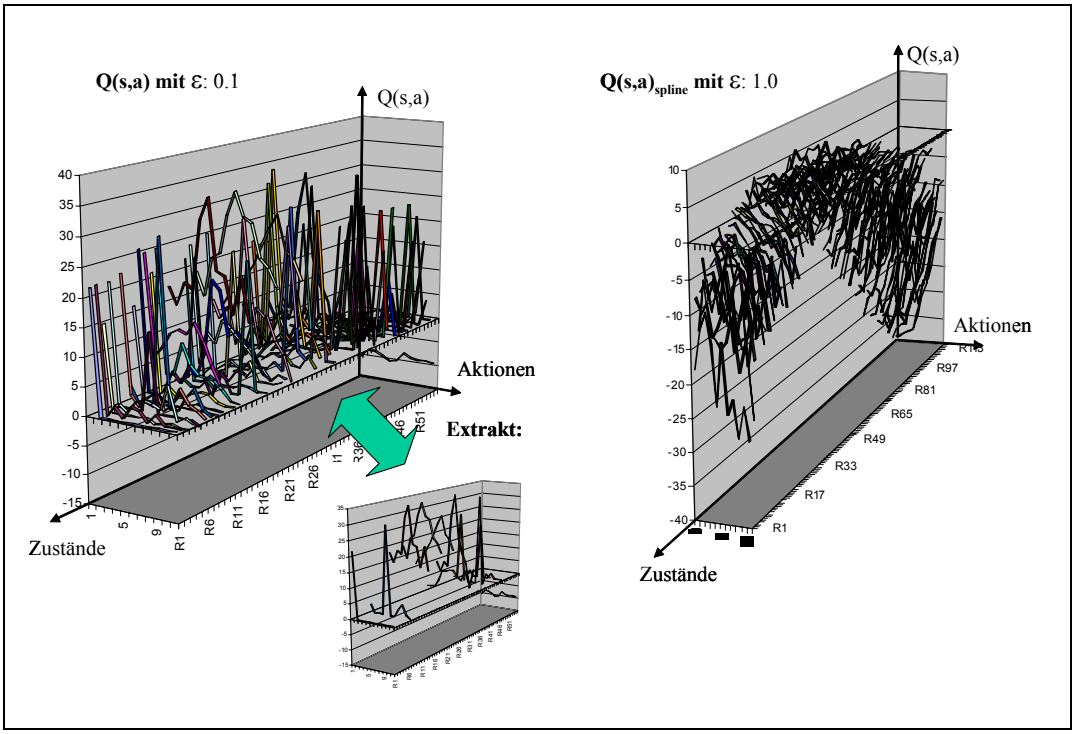


Abbildung 6.30: Erkundung versus Datennutzung (Szenario b)

6.5.4 Auswirkung der Lernrate α

Der mathematische Hintergrund der Lernrate α ist in Kapitel 4.2.2 beschrieben. Sie gibt an, mit welcher Gewichtung der ermittelte Fehlerwert $TDerr$ in die Aktualisierung eingeht. Je größer der Wert von α ist (dabei gilt: $0 < \alpha \leq 1$), desto schneller wird der Lernprozess des Verstärkungslernens vollzogen.

Je kleiner der Wert von α, desto geringer auch der Einfluss von $TDerr$ bei den Aktualisierungen der Situationsbewertungen und desto langsamer konvergieren diese. Da sich aber der Fehlerwert $TDerr$ ebenso von den Situationsbewertungen ableitet, ist der Fehlerwert selber fehlerbehaftet und sollte somit nicht ungedämpft in die Aktualisierung eingehen. Die Lernrate beeinflusst somit die Robustheit des Lernprozesses.

Abbildung 6.31 zeigt das Ergebnis einer Testreihe des Szenarios a) mit jeweils ca. 2.000 ausgeübten Aktionen sowie den Verhaltensraum für drei unterschiedliche Werte von α.

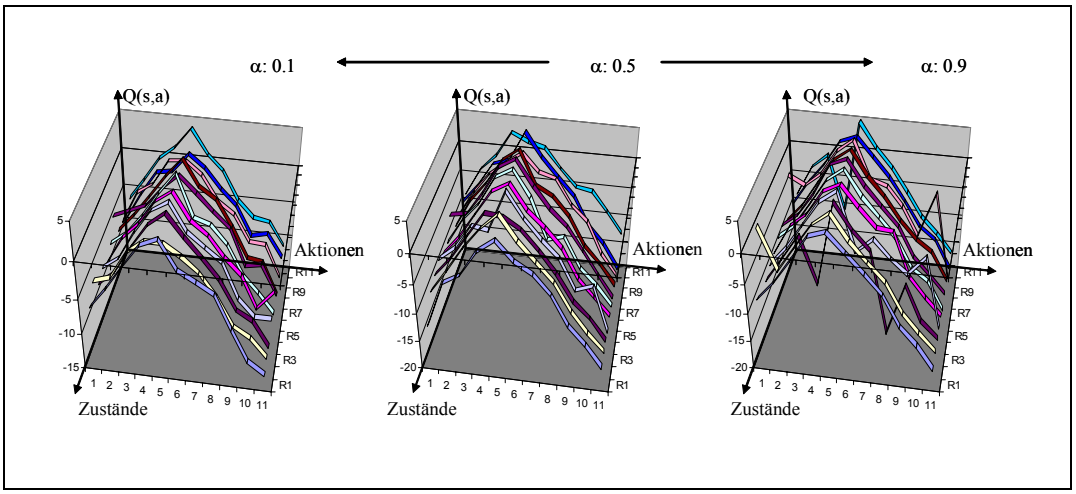


Abbildung 6.31: Einfluss der Lernrate (α) auf den Lernprozess (Szenario a)

Die höhere Robustheit des Lernprozesses ist insbesondere in nicht komplett deterministischen Umgebungen von hoher Bedeutung. Da dies für Szenario b (Lerne den optimalen Fahrwinkel) gilt, behindert ein zu großer Wert von α den Konvergenzprozess der Situationsbewertungen. Abbildung 6.32 zeigt zwei ähnliche Testreihen für das Szenario b), die sich nur durch den Wert von α unterscheiden. Während sich im linken Teil der Abbildung ($\alpha = 0.5$) die Konvergenz des Verhaltensraums nur grob ausbildet, wird diese im rechten Teil der Abbildung ($\alpha = 0.1$) recht deutlich erreicht. Die Verläufe des Glattheitsmaßes unterstreichen diese Bewertung.

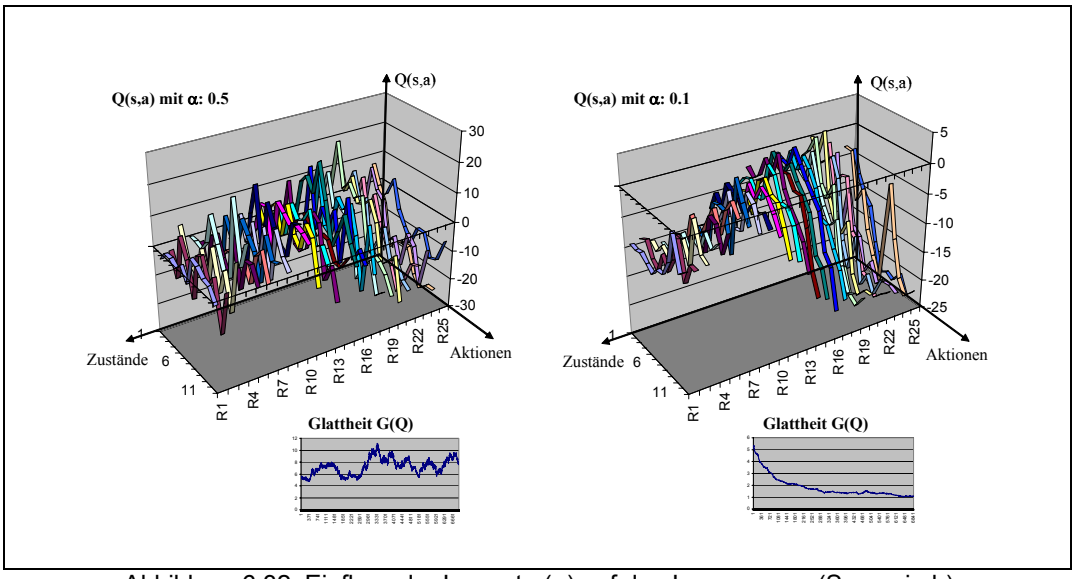


Abbildung 6.32: Einfluss der Lernrate (α) auf den Lernprozess (Szenario b)

6.5.5 Auswirkung des Diskontierungsfaktors γ

Wie mathematisch in 4.2.6 hergeleitet, entscheidet der Diskontierungsfaktor γ über den Einfluss zukünftiger Belohnungen – somit über die langfristige Belohnungsmaximierung. Dies trägt auch maßgeblich dazu bei, dass gestörte Belohnungen kompensiert werden.

In Abbildung 6.33 ist das Ergebnis einer Testreihe gezeigt, in der die Belohnungsfunktion bewusst gestört ist – d.h. es werden in einigen Situationen bewusst falsche Belohnungswerte erzeugt.

Im Detail wird in der dargestellten Testreihe in 25 unterschiedliche Situationen s_0 bis s_{24} klassifiziert. Bei der Situation s_0 befindet sich das Fahrzeug am linken Fahrzeugrand, bei der Situation s_{24} am rechten Fahrzeugrand. Beim Übergang von s_0 auf s_{24} steigt die Belohnungsfunktion zunächst an und fällt anschließend wieder ab. Dabei ist die Belohnungsfunktion bei den Situationen s_6, s_7 und s_8 komplett gestört.

Abbildung 6.33 zeigt die konvergierte Q-Funktion. Trotz der elementaren Störung der Belohnungswerte der Situationen s_6, s_7 und s_8, haben sich die Q-Werte dieser Situationen harmonisch zum übrigen Verhaltensraum ausgeprägt. Im Details ergibt sich selbst für die belohnungsgestörten Situationen eine korrekte Ausprägung der Q-Werte mit einem diagonalen Verlauf der Maxima über die Situationen hinweg.

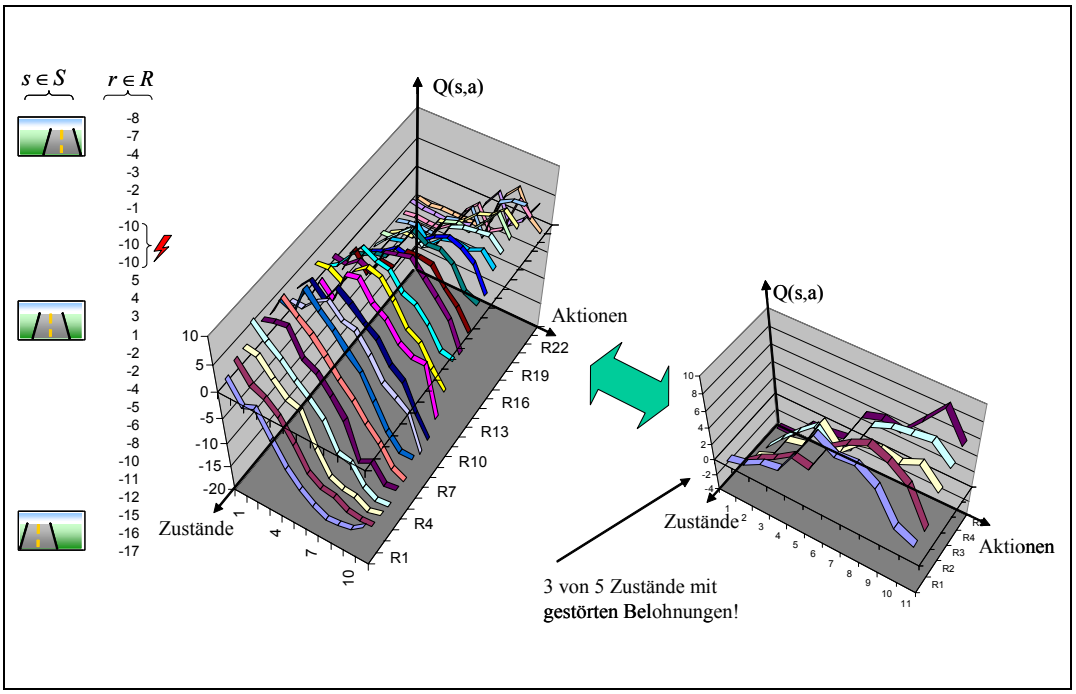


Abbildung 6.33: Kompensation gestörter Belohnungen durch den Diskontierungsfaktor (γ)

6.5.6 Auswirkung des Verfallsfaktors λ

Der Einfluss des Verfallsfaktors λ ist in Kapitel 4.2.7 beschrieben – letztlich wird durch die Aktualisierungen aller zeitlich zurückliegenden Situationen eine schnellere Konvergenz der Q-Funktion erreicht. Dabei ist zu beachten, dass durch zusätzliche Updates ein ähnliches Verhalten wie bei hohen Werten von α erreicht wird: der Lernprozess erfolgt schneller, aber auch weniger robust.

In den durchgeführten Testreihen wurde somit durch Einführung des Verfallsfaktors λ der die Lernrate α wie folgt gedämpft: $\alpha := \alpha \bullet (1-\lambda)$.

Die Ergebnisse der entsprechenden Testreihen zeigt exemplarisch Abbildung 6.34. Durch das Einführen der zusätzlichen Aktualisierungen aller zeitlich zurückliegenden Situationen wird eine schnellere Konvergenz (ersichtlich aus dem zeitlichen Verlauf des Glattheitsmaßes) erreicht.

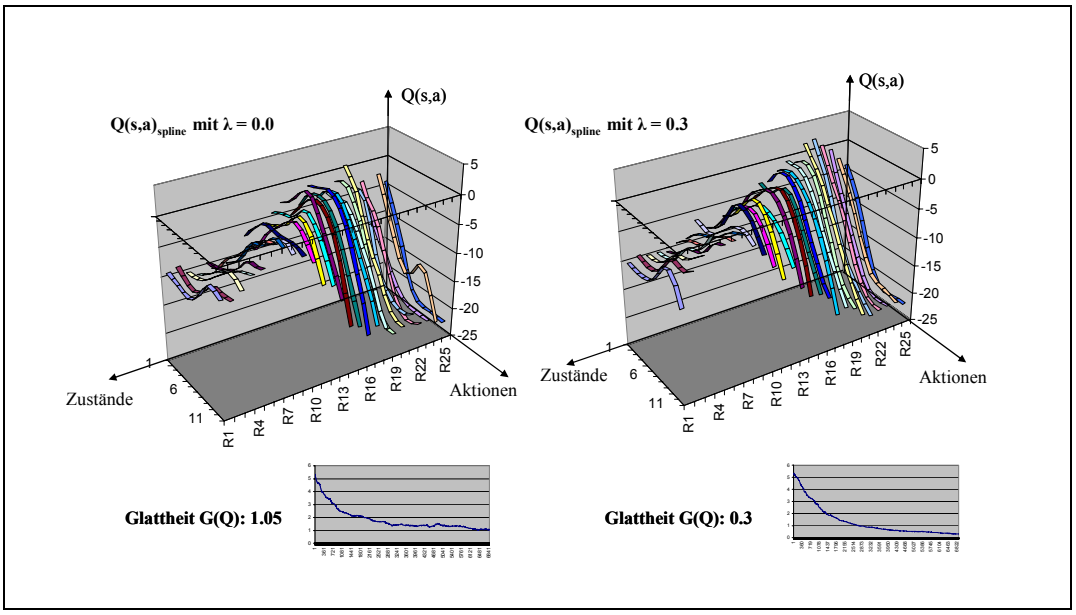


Abbildung 6.34: Schnellere Konvergenz durch den Einsatz des Verfallsfaktors λ

6.5.7 Zusammenfassung Teilsystem Verstärkungslernen

Zunächst wurde im Teilsystem Verstärkungslernen die individuelle Konvergenz der Situationsbewertungen, d.h. der Q-Funktion im Verhaltensraum nachgewiesen. Dabei wurde die Komplexität der Umgebung schrittweise gesteigert und die Konvergenz der Q-Funktion sowie deren Verlauf im Verhaltenstraum untersucht.

Im Rahmen der Testreihen der unterschiedlichen Umgebungen wurden die Parameter des Verstärkungslernens und deren Auswirkungen auf die Konvergenz und die Konvergenzgeschwindigkeit der Q-Funktion im Detail untersucht. Zur besseren Bewertung des Konvergenzgrades der Q-Funktion wurde die Einführung eines Glattheitsmaßes vorgestellt.

Die Durchlaufzeiten für die vorliegende Implementierung des Verstärkungslernens bewegten sich jeweils im Bereich weniger Millisekunden.

7 ZUSAMMENFASSUNG UND AUSBLICK

Im Rahmen der Entwicklung und Erforschung eines Systems zur lernenden Fahrzeugsteuerung und der in Kapitel 1.3 dargestellten Aufgabenstellung konnten umfassende Erkenntnisse im Umgang mit einem Verstärkungslernen-System gewonnen werden.

Eines der wesentlichen erreichten Ziele ist, dass das Verstärkungslernen-System den jeweiligen Verhaltensraum komplett autonom erforscht. Obwohl im Rahmen der Testreihen unterschiedliche Umgebungsbedingungen (die unter Kapitel 5.3 dargestellten Szenarien) angesetzt wurden, konnte das jeweilige Verstärkungslernen-System die angemessenen Verhaltensweisen allein aufgrund der zugeführten Belohnungen ermitteln.

Ein besonderer Schwerpunkt der Arbeit war dabei auch, die Auswirkungen der Parameter des Verstärkungslernen-Systems auf die Konvergenz der Situationsbewertungen zu erforschen. Dabei ist hervorzuheben, dass eine wesentliche Anforderung an das Verstärkungslernen-System ist, fehlende oder fehlerhafte Belohnungen soweit wie möglich zu tolerieren. Diese Fähigkeit unterscheidet Verstärkungslernen-Systeme von anderen Lernverfahren und hat für die korrekte Aktionsselektion elementare Bedeutung. Diese Fähigkeit konnte im Rahmen der Testserien entsprechend nachgewiesen werden.

Im Rahmen der Arbeit wurde zusätzlich die Einführung eines Glattheitsmaßes vorgestellt, um den Grad der Konvergenz der Situationsbewertungen zu quantifizieren. Ohne dieses Maß mangelt es an Aussagekraft, ob, bzw. wie weit eine Konvergenz der Situationsbewertungen erfolgt ist und in Konsequenz, mit welcher Sicherheit die durch Verstärkungslernen ermittelten optimalen Aktionen verlässlich sind.

Ein weiterer Aspekt war die Wechselwirkung zwischen Erkundung und Datennutzung. Insbesondere bei komplexen Verhaltensräumen kann eine vollständige Erkundung sehr aufwendig sein und dadurch ist gegebenenfalls die Beschränkung auf eine lokale Erkundung und somit eine Beschränkung auf lokale Konvergenz akzeptabel.

Zur Komplettierung des Lernsystems wurden in der Arbeit eine komfortable Testumgebung, eine vollständige Bildverarbeitung zur Fahrbahnerkennung und eine Fahrzeugregelung durch direkte Kopplung von Bildverarbeitung und einem Lernsystem geschaffen.

Zum einem wurde ein Algorithmus entworfen und implementiert, der schnell und robust die merkmalsrelevanten Informationen eines Einzelbildes nachbildet – im vorliegenden Fall die Fahrbahnmarkierungen. Zum anderen wurde nachgewiesen, dass ein ausschließlich auf Bildverarbeitung und Mustererkennung basierendes System bereits beachtliche Fahreigenschaften aufweist.

Die im Rahmen dieser Arbeit erhaltenen Erkenntnisse ermutigen zu weiteren Forschungen im Rahmen Verstärkungslernen – insbesondere im Umfeld von Fahr-, bzw. Fahrerassistenzsystemen.

Zum einen wäre eine selektive Erkundung des Verhaltensraumes denkbar. Dort, wo im Verhaltensraum die Konvergenz der Q-Funktion noch nicht vollzogen wurde, ist eine höhere Erkundung durchzuführen. Dort, wo die Konvergenz der Q-Funktion schon eingetreten ist, ist der Übergang auf den Datennutzungsmodus möglich. Damit würde erreicht, dass die Q-Funktion im Verhaltensraum zum einen so schnell wie möglich konvergiert– zum anderen aber möglichst früh von bedeutungslos gewordenen Erkundungen befreit wird.

Im Rahmen einer solchen Erweiterung wäre es auch möglich, eine Aussage über den Konvergenzgrad einer einzelnen Situationsbewertung, d.h. eines Teils des Verhaltensraums zu machen. Dies würde dazu führen, dass ein Urteil über die Verwendbarkeit einer Aktion getroffen werden kann. Eine Aktion, die auf Basis einer konvergierten Situationsbeschreibung bestimmt wird, ist mit Sicherheit angemessener als eine Aktion, die auf Basis einer nichtkonvergierten Situationsbeschreibung bestimmt wird.

Alternativ zur selektiven Erkundung wäre auch eine Approximation der Q-Funktion denkbar. Die Herausforderung in diesem Umfeld liegt dabei auf der Wahl aussagekräftiger Stützstellen in Bezug auf die gewählte Funktionsapproximation.

Ein Aspekt weiterer Forschung könnte auch das Verhalten von Verstärkungslernen im Rahmen von nicht-deterministischen, bzw. stark verzögerten Belohnungen sein. Dazu ist ein wahrscheinlichkeits-basierter Ansatz in die unter Kapitel 4.2.5 entwickelten Formeln einzubringen und durch experimentelle Ergebnisse zu untermauern.

Ebenso wurde in dieser Arbeit der Fokus auf die Querregelung (Lenkung) eines Fahrzeugs gelegt. Eine Erweiterung in Bezug auf die Längsregelung wäre denkbar – entweder dadurch, dass der Verhaltensraumes um eine zweite Aktion erweitert wird oder dadurch, dass ein parallel komplettes zweites Verstärkungslernen-System für die Längsregelung implementiert wird.

Die erarbeiteten Ergebnisse sollten aber auch im Rahmen konkreter Projekte nutzbar sein. So wäre z.B. denkbar, ein Fahrerassistenzsystem die Situationen und korrekten Aktionen lernen zu lassen. Dies wäre im normalen Fahrbetrieb – d.h. auch unbemerkt vom Fahrer eines Fahrzeugs möglich. Sobald eine Konvergenz der Q-Funktion im Verhaltensraum eingetreten ist, könnte das System in den Datennutzungsmodus übergehen und die Aktionen des Fahrers mit den optimal ermittelten Aktionen vergleichen und bei größeren Diskrepanzen informierend oder handelnd eingreifen.

In Konsequenz sollten die Ergebnisse der vorliegenden Arbeit ein besseres Verständnis im Umgang mit Verstärkungslernen-Systemen bewirken und weitere Forschung bzw. konkreter Anwendungsfälle ermutigen.

ANHANG A GLOSSAR

ACSD (Abstract Complete Situation Description)
Situationsbeschreibung in einem vorgegebenen Format, um Ähnlichkeitsuntersuchungen zwischen ACSD's untereinander zu ermöglichen. Um ein ACSD zu erhalten, werden die aus einem Bild ermittelten *parametrischen Szenenbeschreibungen* (z.B. *Splines*) mit einem festen Raster zur Überlappung gebracht und die Verortungen und Winkel der Schnittpunkte identifiziert. Die variable und individuelle Form der *parametrischen Szenenbeschreibung* kann dadurch in eine einheitliche Darstellung mit fester Metrik umgewandelt werden und wird in dieser Arbeit im Rahmen als *Zustand* verwendet.

Agent (agent)
Zentraler Teil eines Verstärkungslernen-Systems

Aktion (action)
Ausgabewerte - in der vorliegenden Arbeit sind diese Aktionen *Steuerkommandos* zur Ausführung von Lenkmanövern.

Aktualisierung, Aktualisierungsfunktion (update, update function)
Veränderung der *Situationsbewertung* in Bezug auf eine erhaltene *Belohnung*.

Belohnung, Belohnungswert (reward)
Numerische Bewertung der aktuellen *Situation* in Bezug auf die zu lernende Aufgabe. In der vorliegenden Arbeit ist die Belohnung umgekehrt proportional zum Abstand von der Fahrbahnmitte.

Bildausschnitt
Teilausschnitt aus einem *Einzelbild*.

Bildfolge
Eine Bildfolge ist eine Mehrzahl von zeitlich aufeinanderfolgenden Bildern, die in der vorliegenden Arbeit durch die periodische Digitalisierung von Verkehrsszenen entstehen.

Binärbild
Ein binarisiertes Bild (z.B. einer Verkehrsszene). In der vorliegenden Arbeit wird beim Prozess der Binarisierung eine Faltung mit einem horizontalen Differenzoperator durchgeführt – deshalb repräsentiert ein gesetzter Pixel im Binärbild einen Kandidaten für die Nachbildung der Fahrbahnmarkierung.

Datennutzung (exploitation)
Bestimmung einer *Aktion* aufgrund des aktuellen *Verhaltensraums*, d.h. aufgrund der aktuellen Werte der *Situationsbewertungen* im *Verhaltensraum* in Kombination mit der *Strategiefunktion*.

Diskontierungsfaktor (discount factor) γ
Numerischer Wert ($0 \leq \gamma \leq 1$) zur abschwächenden Berücksichtigung zukünftiger *Belohnungswerte*.

Einzelbild
Einzelbild einer *Bildfolge*.

Erkundung (exploration), Erkundungsparameter ε
Gezielte Erkundung des *Verhaltensraums* durch Ausgabe und anschließende Bewertung von beliebigen (d.h. nicht-optimalen) *Aktionen*. Der Grad der Erkundung wird über den Erkundungsparameter ε ($0 \leq \varepsilon \leq 1$) gesteuert, der das Verhältnis zwischen Erkundung und *Datennutzung* regelt.

Fehlerwert TDerr
Sich ergebende Fehlergröße im Umfeld des *Verstärkungslernens* aufgrund von nicht vollständig konvergierten *Situationsbewertungen* im *Verhaltensraum*. Der Fehlerwert TDerr wird als Basis für die *Aktualisierung* der *Situationsbewertungen* verwendet.

Kachel
Gleichmäßiger quadratischer Unterbereich eines Bildes mit der Dimension $n \bullet n$.

Kachel-Datenbank
Die im Systemspeicher oder als Datei gespeicherten *Kacheldaten*.

Kacheldaten
Eine Mehrzahl von Matrizen der Dimension $n \bullet n$ (gleiche Dimension wie die *Kacheln*), die die bedingte Wahrscheinlichkeit angibt, ob ein referenzierter Pixel Teil einer Fahrbahnmarkierung ist. Die Kacheldaten werden für die Bildung von *Verkettungsvektoren* verwendet.

Lernrate (learning rate) α
Wert, mit dem der *Fehlerwert TDerr* multipliziert wird, bevor dieser in das *Update* der *Situationsbewertungen* eingeht – dabei gilt: $0 < \alpha \leq 1$. Je größer der Wert von α, desto schneller der Lernprozess beim *Verstärkungslernen*. Je kleiner der Wert von α, desto langsamer, aber auch desto robuster, ist der Lernprozess.

Lernsystem
Oberbegriff für das in der vorliegenden Arbeit implementierte und untersuchte lernende System auf Basis Verstärkungslernen, welches im Kern zwei Problemfelder adressiert: Zum einen wird **eigenständig** eine Anzahl von möglichen Situationsbeschreibungen anlegt und zum anderen wird für diese Situationsbeschreibungen die Fahrzeugsteuerung, d.h. die Wahl der auszugebenden Steuerkommandos, autonom **optimiert**.

Parametrische Szenenbeschreibung
Darstellung der relevanten Informationen eines Einzelbildes durch Parameter – wie z.B. durch *Verkettungen* oder *Splines*. Die Parameter der Szenenbeschreibung werden so ermittelt, dass sie zur Bildung von *ACSD's* verwendet werden können, die wiederum eine Annäherung an in der Verkehrsszene erkannten Fahrbahnmarkierungen darstellen. Die parametrischen Szenenbeschreibungen verfügen über keine feste Metrik als Speicherformat.

Sensordaten
Daten vorhandener Sensoren die wiederum repräsentativ für die *Situation* sind, in der sich das autonome System befindet. Von diesen Sensordaten wird in der Regel eine *Situationsbeschreibung* abgeleitet.

Situation (situation)
Zustand von Umgebung und autonomem System wie z.B. die Eigenposition oder der Daten von Aktoren.

Situationsbeschreibung (situation description)
Auf Parameter reduzierte Beschreibung der *Situation*, daher eine quantisierte Untermenge der unendlichen Menge der *Situationen*, in der sich das autonome System befinden kann.
In der vorliegenden Arbeit werden diese in Form von *ACSD* beschrieben und als *Zustand* des Lernsystems verwendet. Die Situationsbeschreibungen, bzw. die ACSD's verfügen über keine feste Metrik als Speicherformat.

Situationsbewertung
Situationspezifische Bewertung einer *Aktion*. Die Situationsbewertungen einer ausgeführten *Aktion* werden nach Erhalt einer *Belohnung* einem *Update* unterzogen. In Bezug auf die vorliegende Arbeit, die im Wesentlichen auf der Variante Q-Lernen des Verstärkungslernens aufbaut, wird die Situationsbewertung auch als $Q(s_t, a_t)$ oder auch allgemein als Q-Funktion bezeichnet.

Spline
Ein Spline ist eine stückweise zusammengesetzte Funktion, deren Einzelstücke über jeweils eigene Polynome gebildet werden. Die Aufgabenstellung bei der Bildung eines Splines ist es, eine mathematische Darstellung einer Kurve zu ermitteln, die in möglichst glatter Form durch eine vorgegebene Anzahl von Punkten verläuft oder sich zumindest an diese anschmiegt. Letzterer Fall wird als Spline-Approximation bezeichnet. In der vorliegenden Arbeit werden Polynome dritten Grades verwendet, damit wird der Spline als kubischer Spline bezeichnet. Details zu Eigenschaften und der Bildung von Splines ist [Engeln & Uhlig 96] dargestellt. Diese Splines (die in der vorliegenden Arbeit augrund von kubischer Spline-Approximation gebildet werden) sind dadurch eine Annäherung an im Eingangsbild erkannte Fahrbahnmarkierungen.

Steuerkommando (steering command)
Ausgabewert des Lernsystems an die Umwelt; in der vorliegenden Arbeit entspricht das Steuerkommando der Anweisung zur Querlenkung.

Strategiefunktion (Policy)
Übergeordnete Verhaltensrichtlinie beim *Verstärkungslernen* in Bezug auf die Selektion von *Aktionen*.

Umgebung (environment)
Umwelt außerhalb des Lernsystems mit dem das Lernsystem über *Aktionen* und *Sensordaten* interagiert.

Verfallsfaktor (Trace decay parameter) λ
Multiplikationswert für *Aktualisierungen* zeitlich zurückliegender *Situationen* $0 \leq \lambda \leq 1$).

Verhalten (behaviour)
Der Begriff Verhalten steht synonym für die situationsspezifische Ermittlung und Ausgabe von *Aktionen*. Dabei sagt der Begriff Verhalten noch nichts über die Angemessenheit der *Aktionen* aus – Ziel einer jeden autonomen Roboter- oder Fahrzeugsteuerung ist das Optimieren von Verhalten, d.h. die Ermittlung und Ausgabe von möglichst situationsangemessenen *Aktionen*.

Verhaltensraum (behaviour space)
Virtueller Raum, der aufgrund von *Situationsbeschreibungen/Zuständen*, *Aktionen* und *Situationsbewertungen* aufgespannt wird.

Verkettung
Verbindung einzelner Pixel in einem *Binärbild*. Idealerweise besteht eine Verkettung nicht nur aus der Verbindung von zwei Pixel sondern aus der mehrfachen Weiterverbindung zu Folgepixel, d.h. einer Kettenbildung.

Verkettungsvektor
Vektor zur Durchführung von *Verkettungen* – dabei geben die Verkettungsvektoren die bedingten Wahrscheinlichkeiten darüber an, ob zwei aufgrund horizontaler Farbunterschiede selektierte Pixel Teil einer Fahrbahnmarkierung sind. Die Verkettungsvektoren werden auf Basis statistischer Methoden in Kombination mit Bildauswertungen ermittelt.

Verstärkungslernen (reinforcement learning)
Lernverfahren zum Lernen und Selbstoptimieren eines Systems wie im Detail in [Sutton & Barto 98] beschrieben. In der vorliegenden Arbeit wird die Variante des Q-Learning verwendet.

Zustand (state)
Zustandsidentifikation - in der vorliegenden Arbeit werden die *ACSD's* als Zustand verwendet.

ANHANG B LITERATURVERZEICHNIS

[Anderson & Hong 94] C. Anderson and Z. Hong. Reinforcement Learning with Modular Neural Networks for Control. Proceedings of NNACIP'94, the IEEE International Workshop on Neural Networks Applied to Control and Image Processing, 1994

[Arya & Mount 93] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In Proc. 4th ACM-SIAM Symposium Discrete Algorithms, pages 271-280, 1993

[Arya et al 98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. J. ACM, 45:891-923, 1998

[Atkeson & Santamaria 97] C. G. Atkeson, J. C. Santamaria, A Comparison of Direct and Model-Based Reinforcement Learning', International Conference on Robotics and Automation, 1997

[Baird 99] L. Baird III; Reinforcement Learning through Gradient Descent; Dissertation; Carnegie Mellon University, Pittsburgh/USA, 1999

[Baluja & Pomerlau 97] S. Baluja, D. A. Pomerleau; Expectation-based selective attention for visual monitoring and control of a robot vehicle; Robotics and Autonomous System Vol.22, No.3-4, December, 1997

[Bischoff & Graefe 04] R. Bischoff, V. Graefe, Robot Navigation without Calibration. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai/Japan, 2004

[Dickmanns & Zapp 87] E. D. Dickmanns, A. Zapp, Autonomous High Speed Road Vehicle Guidance by Computer Vision, Preprints of the 10th World Congress on Automatic Control, Vol.4, International Federation of Automatic Control, Munich/Germany, July 27-31, 1987

[Dickmanns 02] E. D. Dickmanns, The Development of the Sense of Vision for Ground Vehicles over the Last Decade, Intelligent Vehicles Symposium (IV) 2002, Versailles/France May 18-20, 2002,

[Dickmanns 87] Dickmanns, E. D.: 4-D Szenenanalyse mit integralen raum-/ zeitlichen Modellen. DAGM Symposium, 1987

[Dickmanns et al. 94] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, J. Schiehlen, The Seeing Passenger Car 'VaMoRs-P', Intelligent Vehicles Symposium (IV) 1994, Paris/France, October 24-26, 1994

[Dong & Kuhnert 04] W. Dong, K.-D. Kuhnert, Robust adaptive control of honholonomic mobile robot with parameter and non-parameter uncertainties, IEEE Transaction on Robotics and Automation, 2004

[Engeln & Uhlig 96] G. Engeln-Müllges, F. Uhlig, Numerical Algorithms with C; Springer Verlag; 1996

[Enkelmann 97] W. Enkelmann, Entwicklung von Systemen zur Interpretation von Straßenverkehrsszenen durch Bildfolgenauswertung; infix Verlag, 1997

[Ekermann 98] E. Ekermann, Die Achsschenkellenkung und andere Fahrzeug-Lenkssysteme, Deutsches Museum München, 1998

[Faugeras 93] O. Faugeras, Three-Dimensional Computer Vision: A geometric viewpoint; The MIT Press, London/UK, 1993

[Franke et al. 98] U. Franke, D. Gavrilla, S. Görzig, F. Lindner, F. Paetzold, C. Wöhler, Autonomous Driving Goes Downtown, IEEE Intelligent Vehicles Systems, v.13 n.6, p.40-48, November 1998

[Gasket et al 00] C. Gaskett, L. Fletcher, A. Zelinsky, Reinforcement learning for a vision based mobile robot, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Takamatsu/Japan, 2000

[Gloye 05] A. Gloye, Lernmethoden für autonome mobile Roboter, Dissertation FU Berlin/Germany, 2005

[Goerke & Henne 05] N. Goerke, T. Henne, Controlling an Autonomous Agent using Internal Value based Action Selection, International workshop on Automatic Learning and Real-Time (ALART), Siegen/Germany, September 7-8,2005

[Graefe & Bischoff 03] V. Graefe, R. Bischoff, Past, Present and Future of Intelligent Robots. IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA, Kobe/Japan 2003

[Graefe 1999] V. Graefe, Calibration-Free Robots. Proceedings, the 9th Intelligent System Symposium, Japan, 1999

[Gregor 02] R. Gregor, Fähigkeiten zur Missionsdurchführung und Landmarkennavigation, Dissertation, Universität der Bundeswehr München, 2002

[Haberäcker 91] P. Haberäcker, Digitale Bildverarbeitung; Hanser Verlag; 1991

[Henke 98] M. Henke, Erkennung von Fahrbahnmarkierungen im Totwinkelbereich eines Straßenfahrzeuges; Diplomarbeit; Universität-Gesamthochschule Siegen, Fachbereich 12, Fachgruppe Prozeßdatenverarbeitung; Oktober 1998

[Jähne 97] B. Jähne, Digitale Bildverarbeitung; Springer Verlag; 1997

[Jochem et al. 93] T. M. Jochem, D. A. Pomerleau, C. E. Thorpe. MANIAC: A Next Generation Neurally Based Autonomous Road Follower, IAS-3, Int. Conference on Intelligent autonomous Systems, Pittsburgh/USA, February 15-18, 1993

[Jochem et al. 95] T. M.Jochem, D. A.Pomerleau, C. E.Thorpe, Vision Guided Lane Transition, Intelligent Vehicles ’95 Symposium, Detroit/USA, September 25-26, 1995

[Kretchmar 00] R. M. Kretchmar, A Synthesis of Reinforcement Learning and Robust Control Theory, Ph.D. Dissertation, Department of Computer Science, Colorado State University, Fort Collins, CO 80523, 2000

[Kroedel & Kuhnert 00] M. Krödel, K.-D. Kuhnert, Towards a Learning Autonomous Driver System, IEEE International Conference on Industrial Electronics, Control and Instrumentation, Nagoya/Japan, October 22-28, 2000

[Kroedel & Kuhnert 01] M. Krödel, K.-D. Kuhnert, Autonomous Driving through Intelligent Image Processing and Machine Learning, Int’l Conference on Computational Intelligence, Dortmund/Germany, October 1-3 2001

[Kroedel & Kuhnert 02] M. Krödel, K.-D. Kuhnert, Pattern Matching as the Nucleus for either Autonomous Driving or Drive Assistance Systems, IEEE Intelligent Vehicle Symposium, Versailles/France, June 17-21, 2002

[Kuhnert & Dong 03] K.-D. Kuhnert, W. Dong, Über die lernende Regelung autonomer Fahrzeuge mit Neuronalen Netzen, 18. Fachgespräch Autonome Mobile Systeme (AMS), Karlsruhe/Germany, December 4-5, 2003

[Kuhnert & Kroedel 6-02] K.-D. Kuhnert, M. Krödel, Autonomous Driving by Pattern Matching and Reinforcement Learning, International Colloquium on Autonomous and Mobile Systems, Magdeburg/Germany, June 25-26, 2002

[Kuhnert & Kroedel 9-02] K.-D. Kuhnert, M. Krödel, Reinforcement Learning to drive a car by pattern matching, Anual symposium of Pattern recognition of DAGM, Zurich/Switzerland, September 16-18, 2002

[Kuhnert 86] K.-D.-Kuhnert, A Vision System for Real Time Road and Object Recognition for Vehicle Guidance, Proc. Mobile Robots, Cambridge, Massachusetts, Society of Photo-Optical Instrumentation Engineers, SPIE Volume 727, Oct 30-31, 1986

[Kurzweil99] R. Kurzweil, The Age of Spiritual Machines: When Computers Exceed Human Intelligence, Penguin paperback, 1999

[Langenhagen 01] J. Langenhangen, Nearly nearest Neighbour search with principal components, Student Thesis, Siegen, 2001

[Mitsunaga et al 05] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro and N. Hagita: Robot Behavior Adaptation for Human-Robot Interaction Based on Policy Gradient Reinforcement Learning, IROS 2005, , Edmonton/Canada, August 2-6 2005

[Mount 98] D. M. Mount, ANN Programming Manual, Department of Computer Science and Institute for Advance Computer Studies, University of Maryland, 1998

[Oh et al 00] S-Y. Oh, J-H. Lee, D-H. Choi, A new reinforcement learning vehicle control architecture for vision based road following. IEEE Transactions on vehicular technology 49, 2000

[Onat 98] A. Onat, Q-learning with recurrent Neural Networks as a Controller for the Inverted Pendulum Problem, The Fifth International Conference on Neural Information Processing, pp 837-840, October 21-23, 1998

[Pommerlau 91] D. A. Pommerleau, Efficient Training of Artificial Neural Networks for Autonomous Navigation, Neural Computation 3, 1991

[Ramachandran et al 05] S. Ramachandran, D. Bree, N. P. Gopalan, Learning action selection in autonomous agents, Preprint for IEEE Conference on Intelligent Transportation Systems, 2005

[Rojas 96] R. Rojas, Neural Networks — A Systematic Introduction, Springer, Berlin, 1996

[Shackleton & Gini 97] J. Shakleton, M. Gini, Measuring the Effectiveness of Reinforcement Learning for Behavior-Based Robots, Adaptive Behavior 1997

[Sutton & Barto 98] R. Sutton, A. G. Barto, Reinforcement Learning, An introduction, MIT-Press, Cambridge/USA, 1998

[Szita et al 02] I. Szita, B. Takács, A. Lőrincz, MDPs, Learning in Varying Environments, Journal of Machine Learning Research, 2002

[Turing 50] A. M. Turing, Computing Machinery and Intelligence, Mind 59, page 433-460, 1950

[Vollbrecht 99] H. Vollbrecht, Hierarchic Task Composition in Reinforcement Learning for Continous Control Problems, Ulmer SFB Bericht, 1999