Universität Ulm
Abteilung Neuroinformatik
Prof. Dr. Günther Palm

Visuelle Objekterkennung in dynamischen Umgebungen

Dissertation
zur Erlangung des Doktorgrades
Dr. rer. nat.
der Fakultät für Ingenieurwissenschaften
und Informatik der Universität Ulm



vorgelegt von
Gerd Mayer
aus Ulm
2006

Dekan:	Prof. Dr. Helmuth Partsch
Gutachter:	Prof. Dr. Günther Palm
Gutachter:	Prof. Dr. Heiko Neumann
Tag der Promotion:	3. Mai 2007

Zusammenfassung

Objekte in ihrer Umgebung erkennen zu können, ist eine der wesentlichen Fähigkeiten autonomer, mobiler Roboter. Die Roboter müssen die Gegenstände, die sie manipulieren, oder andere Roboter, mit denen sie interagieren sollen, erkennen, um ihre Aufgabe zu erfüllen. Ist dieser Vorgang bereits in kontrollierten, statischen Umgebungen nicht trivial, sind in dynamischen Umgebungen weitere Probleme zu erwarten. Geringere Bildqualität, Rechenzeitbeschränkung und hohe Objektvariabilität erschweren die Aufgabe zusätzlich. Solche Umgebungen stellen aber ein ideales Umfeld für Untersuchungen über zukünftige, realweltliche Anwendungsgebiete dar.

Ziel der hier vorgestellten Arbeit ist die Entwicklung eines Objekterkennungssystems für die Anwendung innerhalb einer solchen dynamischen Umgebung. Am Beispiel fußballspielender Roboter im RoboCup werden die Besonderheiten solcher Einsatzszenarien untersucht und ein robuster und flexibler Lösungsansatz vorgestellt.

Die bisher gebräuchlichen Ansätze im RoboCup und in vielen anderen Einsatzbereichen autonomer, mobiler Roboter sind oftmals stark heuristisch und lassen sich nur schwer an veränderte Umweltbedingungen anpassen oder auf andere Szenarien anwenden. Die hier vorgestellte modulare Architektur basiert auf einem mehrstufigen Bearbeitungsschema, so daß sich einzelne Schritte, ohne Einfluß auf andere Schichten, an veränderte Bedingungen anpassen lassen. Die enge Kopplung der einzelnen Stufen mit einer rückgekoppelten zeitlichen Integration der Ergebnisse erhöht die Sicherheit über die gefundenen Objekte und reduziert gleichzeitig den für die Erkennung nötigen Bearbeitungsaufwand. Zusätzlich läßt sich so statistisches Modellwissen über das Einsatzszenario bereits während der Objekterkennung berücksichtigen.

Das im Rahmen dieser Arbeit entwickelte System ist in der Lage, alle für den Roboterfußball relevanten Objekte robust und in ausreichend hoher Geschwindigkeit zu erkennen. Durch die Verwendung eines Frameworks zur Videobildverarbeitung auf Robotern und die Implementation möglichst genereller Filterkomponenten werden große Teile der entstandenen Umsetzung aber auch erfolgreich in anderen Projekten eingesetzt.

Danksagung

Nach jahrelanger Arbeit an einer Dissertation, gibt es viele Personen, die zu deren Gelingen beigetragen haben und denen man am Schluß danken sollte. Bestimmt habe ich einen oder mehrere davon vergessen. Ich bitte das zu entschuldigen, ich bin für jede erhaltene Hilfe dankbar.

Danken möchte ich der DFG für die Finanzierung des SPP-1125-Projekts „Adaptivität und Lernen in Teams kooperativer mobiler Roboter“. Ebenfalls geht ein Dank an die Universität Ulm und die Ulmer Universitätsgesellschaft für die stetige Unterstützung des Ulmer RoboCup-Teams „Ulm Sparrows“. Ohne die regelmäßige Teilnahme an nationalen und internationalen Turnieren wäre ich nicht in der Lage gewesen, geeignetes Material für die Erstellung und Auswertung dieser Arbeit zu sammeln.

Weiter geht mein Dank an Herrn Professor Palm. Nicht nur, daß er mir die Möglichkeit zur Promotion gab, sondern auch, daß er sich so engagiert um die Robotik in der Abteilung bemüht. Dank auch an Herrn Professor Neumann für die Bereitschaft, die Rolle des zweiten Gutachters zu übernehmen. Ganz besonders danken möchte ich Herrn Kraetzschmar für die eigentliche Betreuung der Arbeit und viele fruchtbare Diskussionen in seiner Zeit an der Universität Ulm.

Danke den ganzen Ulm Sparrows für ihre unzähligen Arbeitsstunden für das Team und dafür, daß sie meine Launen auch während der Turniere ausgehalten haben. Danke auch den Münchnern „Agilo Robocuppers“, den Grazern „Mostly Harmless“ und dem Team „Carpe Noctem“ aus Kassel für die gute Zusammenarbeit, die Möglichkeit Daten zu sammeln, ihre Mitarbeit an der verwendeten Middleware und den gemeinsamen Publikationen.

Auch meinen Kollegen Uli Kaufmann und Hans Utz gilt mein Dank. Uli Kaufmann legte mit seiner Diplomarbeit den Grundstein für diese Arbeit und wird bestimmt auch in Zukunft die Ulm Sparrows am Laufen halten. Ganz besonders hilfreich war aber die Zusammenarbeit mit Hans Utz, die weit über die normalen Arbeitszeiten hinausging. Hans: „send it to the moon“. Weiter möchte ich mich bei Markus Müller und Pierre Bayerl bedanken, daß sie die Arbeit so tapfer Korrektur gelesen haben.

Mein ganzer Dank gilt aber meiner kleinen Familie: Alexandra und Emily. Entschuldigung, daß ich am Schluß so wenig Zeit für euch hatte. Und danke, daß ihr mich trotzdem ausgehalten habt.

Daumenkino

Im Laufe der Arbeit werden Ihnen die Bilder an den unteren, linken und rechten Rändern auffallen. Diese Bilder sind Teil eines Daumenkinos. Wenn sie die Seiten mit dem Daumen schnell durchblättern, sollte der Eindruck von Bewegung entstehen. Die ausführliche Beschreibung dieser Sequenzen ist Teil des Auswertungskapitels und im Detail in Abschnitt 9.5 erklärt.

Falls sie die Bilder lieber in Form eines Films sehen wollen, so stehen ihnen unter http://smart.informatik.uni-ulm.de/DataSets/RoboCup/Movies/ alle Sequenzen in verschiedenen Formaten zum Herunterladen zur Verfügung.

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Motivation . . . 1
- 1.2 Problemstellung . . . 2
- 1.3 Lösungsansatz . . . 4
- 1.4 Ergebnisse . . . 5
- 1.5 Überblick . . . 6

2 Problemanalyse **9**
- 2.1 Grundlegende Rahmenbedingungen im RoboCup . . . 9
- 2.2 Farben und Beleuchtung . . . 12
- 2.3 Die Roboter . . . 14
- 2.4 Regeladaption . . . 17
- 2.5 Kernpunkte . . . 19

3 Lösungsansatz **21**
- 3.1 Objektklassifikation . . . 22
  - 3.1.1 Finden von Regionen . . . 22
  - 3.1.2 Merkmalsberechnung . . . 23
  - 3.1.3 Neuronale Klassifikation . . . 24
- 3.2 Zeitliche Integration der Klassifikationsergebnisse . . . 25
  - 3.2.1 Klassifikationsverbesserung . . . 26
  - 3.2.2 Reduktion des Ressourcenverbrauchs . . . 27
  - 3.2.3 Informationsintegration . . . 28
- 3.3 Kernpunkte . . . 29

4 Implementationsrahmen **31**
- 4.1 Miro . . . 31
- 4.2 Video Image Processing Framework . . . 32
- 4.3 Sonstige Bibliotheken . . . 35
- 4.4 Kernpunkte . . . 35

5 Regionensuche **37**
- 5.1 Farbindizierung . . . 38
  - 5.1.1 Farbkonvertierung . . . 38
  - 5.1.2 Farbklassifikation . . . 40
  - 5.1.3 Bereichserkennung . . . 41
- 5.2 Modellbasierte Regionenselektion . . . 41

5.3 Objekttypen ohne neuronalen Klassifikator . . . 43
5.4 Verwandte Arbeiten . . . 45
5.4.1 Automobilbereich . . . 45
5.4.2 RoboCup . . . 46
5.5 Kernpunkte . . . 49

6 Objektklassifikation **51**
6.1 Merkmale . . . 51
6.1.1 Farbhistogramme . . . 53
6.1.2 Orientierungshistogramme . . . 54
6.1.3 Kombinierte Orientierungs- und Krümmungshistogramme . . . 56
6.2 Merkmalsextraktion . . . 57
6.3 Neuronale Klassifikation . . . 59
6.4 Modellbasierte Selektionsverfeinerung . . . 59
6.5 Verwandte Arbeiten . . . 60
6.5.1 RoboCup . . . 60
6.5.2 Automobilbereich . . . 62
6.6 Kernpunkte . . . 63

7 Objektverfolgung und -prädiktion **65**
7.1 Ablaufschema . . . 65
7.2 Kalman-Filter . . . 67
7.2.1 Zustandsvorhersage in linearen Systemen . . . 68
7.2.2 Kalman-Filter und Prädiktor-Korrektor-Struktur . . . 69
7.2.3 Systemmodelle . . . 69
7.3 Multi-Target-Tracking . . . 71
7.4 Verwandte Arbeiten . . . 73
7.4.1 RoboCup . . . 73
7.4.2 Automobilbereich und Robotik allgemein . . . 75
7.5 Kernpunkte . . . 75

8 Informationsextraktion und -verwendung **77**
8.1 Wahrscheinlichkeit und Sicherheit . . . 77
8.2 Aufgabenbasierte Informationsextraktion . . . 79
8.3 Kernpunkte . . . 82

9 Empirische Evaluation **85**
9.1 Beleuchtungssituation . . . 85
9.1.1 Übersicht der Beleuchtungsbedingungen . . . 86
9.1.2 Untersuchungsmethode . . . 87
9.1.3 Resultate . . . 88
9.1.4 Abschließende Bemerkungen . . . 94
9.2 Regionensuche . . . 95
9.2.1 Ballsuche . . . 96

9.2.2 Eckpfostensuche . . . 98
9.2.3 Abschließende Bemerkungen . . . 99
9.3 Objektklassifikation . . . 100
9.3.1 Parametrisierung . . . 100
9.3.2 Training . . . 101
9.3.3 Klassifikationsergebnisse . . . 102
9.3.4 Ballerkennung . . . 105
9.3.5 Eckpfostenerkennung . . . 106
9.3.6 Abschließende Bemerkungen . . . 107
9.4 Objektverfolgung, -prädiktion und Informationsintegration . . . 108
9.4.1 Kalman-Filter . . . 108
9.4.2 Informationsintegration . . . 112
9.4.3 Abschließende Bemerkungen . . . 114
9.5 Gesamtsystem . . . 114
9.5.1 Berechnungsdauer . . . 115

10 Zusammenfassung **119**
10.1 Kernpunkte . . . 119
10.2 Ausblick . . . 120

Abbildungs- und Tabellenverzeichnis

Abbildungen

1.1 Illustration der Problemstellung 3

2.1 Objekte im RoboCup aus der Sicht eines Roboters 10
2.2 Verschiedene Robotertypen in der RoboCup-middle-size-Liga 11
2.3 Verschiedene Beleuchtungsbedingungen 13
2.4 Generationen der verwendeten Roboter 14
2.5 Kamerasystem eines Sparrows-2003 Roboter 15
2.6 Grenzbereiche der Kameraleistung 18

3.1 Illustration des gesamten Bearbeitungsschemas 28

4.1 Schematische Filterkonfiguration für eine Kantendetektion 34

5.1 Struktur des HSV-Farbraums 39
5.2 Abbildung von Original- nach Indexbild 41
5.3 Nicht optimale Farbindizierung 42
5.4 Falsche Objekthypothese durch farblich ähnliches Objekt 43
5.5 Verschiedene Ansichten desselben Objekts (gelbes Tor) 44
5.6 Überlagerung der Kantenbilder aus Grauwert- und Farbindexbild . . . 45

6.1 Artifizielles und reales Farbhistogramm 53
6.2 Beispielbilder und korrespondierende Orientierungshistogramme 55
6.3 Artifizielles und reales Orientierungs-/Krümmungshistogramm 57
6.4 Vorverarbeitungsschritte bei der Merkmalsberechnung 58

7.1 Illustration der Zuordnung von Messdaten zu Tracks 72

8.1 Schematische Darstellung der gesamten Filterkonfiguration 80
8.2 Detaillierte Ansicht verschiedener Filterkonfigurationen 81

9.1 Verglastes Gebäude der TU München 86
9.2 Beispiel für verschiedene Beleuchtungsbedingungen 87
9.3 Manuelle Kalibrierungen für verschiedene Beleuchtungsbedingungen . . 89
9.4 Räumliche Verteilung der Farbklassen 89
9.5 Räumliche Verteilung der idealisierten Farbsegmentierung 91
9.6 Distanzabhängige Variation der Farbwahrnehmung 93
9.7 Einfluß der Beleuchtung auf den Farbabstand zweier Gegenstände . . . 94

9.8 Wechselnde Beleuchtungsbedingungen während des Spiels 95
9.9 Falsche Ergebnisse der Ballsuche 98
9.10 Observationen und Trajektorien für ruhenden Ball 110
9.11 Observationen und Trajektorien für bewegten Ball 111
9.12 Wahrscheinlichkeitswerte neuronaler Klassifikatoren und Integration . 113
9.13 Wahrscheinlichkeitswerte verschiedener Objekte 116

Tabellen

9.1 Mittelwerte und Varianzen der räumlichen Verteilungen 90
9.2 Mittelwerte und Varianzen idealisierter Kalibrierungen 92
9.3 Verwechslungsmatrizen der Regionensuche für den Ball 97
9.4 Verwechslungsmatrizen der Regionensuche für Eckpfosten 99
9.5 Verwechslungsmatrizen auf den Daten der Orientierungshistogramme . 103
9.6 Verwechslungsmatrizen für Histogramme mit bestem Parametersatz . . 103
9.7 Verwechslungsmatrizen für die einfachen Merkmale 104
9.8 Verwechslungsmatrizen für komplette Robotererkennung 104
9.9 Verwechslungsmatrizen für Einzelnetze der Ballerkennung 106
9.10 Verwechslungsmatrizen für komplette Ballerkennung 106
9.11 Verwechslungsmatrizen für Einzelnetze der Eckpfostenerkennung . . . 107
9.12 Verwechslungsmatrizen für komplette Eckpfostenerkennung 107
9.13 Berechnungszeit einzelner Filter am Beispiel Eckpfostenerkennung . . . 118

1 Einführung

1.1 Motivation

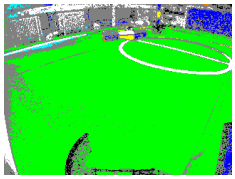
Die Entwicklung von Programmen für autonome, mobile Roboter ist ein komplexes Unterfangen. Solche Programme sind sehr umfangreich, sie reichen von der Informationsverarbeitung über die Aktionsauswahl bis hin zur eigentlichen Bewegungsausführung. Aus diesem Grund würde man normalerweise eine vereinfachte, kontrollierte Umgebung für die Programmentwicklung und Evaluation der einzelnen Komponenten entwerfen. Allerdings würde eine solche kontrollierte, zu artifizielle Umgebung das Problem der Programmentwicklung für reale hochdynamische Szenarien konterkarieren. Vor genau diesem Problem standen jedoch bereits viele Forscher und riefen deshalb vor einigen Jahren den RoboCup ins Leben.

Im RoboCup spielen autonome, mobile Roboter Fußball gegeneinander. Er ist als Testumgebung gedacht, welche Entwicklungen und Algorithmen zwischen verschiedenen Roboterplattformen austauschbar und verschiedene Entwicklungsrichtungen vergleichbar machen soll. Dies bildet die Basis für Lösungen, wie sie später in anderen, weit realistischeren Bereichen eingesetzt werden sollen.

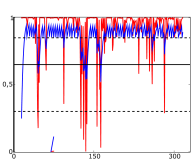
Die Probleme, mit denen man im RoboCup konfrontiert ist, gleichen denen, wie sie in realen Umgebungen vorherrschen. Genauso wie Teammitglieder und gegnerische Spieler in einem Fußballspiel, so gibt es auch in der Realität Personen, welche kooperativ sind und andere, die es nicht sind. Man denke an einen Roboter, der als Führer in einem Museum eingesetzt wird und dazu den Besucher durch ein sehr belebtes Museum führen muß.

Ein anderes wichtiges Merkmal ist die hohe Dynamik der Umgebung. Die Person in unserem Museumsbeispiel mag sich noch kooperativ verhalten und sich in gemäßigtem Tempo bewegen; die anderen Personen im Museum werden das mit Sicherheit nicht tun. Ein anderes Szenario, in dem diese hohe Dynamik eine entscheidende Rolle spielt, ist der gesamte Sektor der autonomen Fahrzeuge. Diese bewegen sich mit noch weit höheren Geschwindigkeiten in ihrer Umgebung. Vom sensorischen Gesichtspunkt aus sind sich beide Beispiele jedoch sehr ähnlich. Ein Roboter muß sich in seiner Umgebung mit vielen mehr oder weniger bekannten Objekten auseinandersetzen. Die einzelnen Individuen, kleine Kinder oder große Erwachsene im ersten Beispiel, verschiedene Automodelle im zweiten, können aber sehr unterschiedlich aussehen.





Das letztgenannte Problem, die *zuverlässige Erkennung relevanter Objekte innerhalb einer hochdynamischen Umgebung*, ist das Thema der vorliegenden Arbeit. Einige der Objekte, die während eines Fußballspiels erkannt werden müssen, sind offensichtlich. Genauso wie bei echten Fußballspielen, sind natürlich der Ball und die Tore Grundvoraussetzung für das Spiel. Die einzelnen Roboter, die Teamkollegen sowie



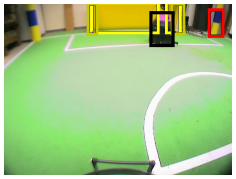
die Gegenspieler zu erkennen ist wichtig, um Taktik sowie Teamverhalten vernünftig umsetzen zu können. Alle anderen Objekte sind vielleicht weniger offensichtlich, aber für das Spiel eines Roboters sehr wichtig. Eckpfosten, Torpfosten und die Bodenlinien dienen dem Roboter zur eigenen Positionsbestimmung auf dem Spielfeld. Alle anderen Objekte, die nicht näher bestimmt werden können, müssen zuverlässig als Hindernisse modelliert werden, da, wie im richtigen Fußball, eine übertriebene Härte gegenüber anderen Robotern und insbesondere gegenüber dem Schiedsrichter streng geahndet wird.

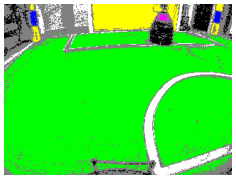
Das Ziel dieser Arbeit ist es, ein System zu entwickeln, das alle obengenannten Objekte in ausreichender Geschwindigkeit und mit hoher Zuverlässigkeit erkennen kann. Dafür können und sollen auch zusätzliche Informationen über das Einsatzgebiet sowie temporale Zusammenhänge berücksichtigt werden. Das wiederum ermöglicht nicht nur eine zuverlässigere Objekterkennung. Es erlaubt auch, die Position von Gegenständen in die Zukunft zu projizieren und diese Information anderen Modulen der Robotersoftware zur Verfügung zu stellen. Alle eingesetzten Verfahren sollten außerdem so flexibel wie möglich sein, so daß sie auch in anderen Einsatzgebieten verwendet werden können.

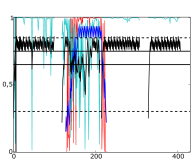
1.2 Problemstellung

Da der RoboCup ein sehr dynamisches Umfeld ist, ist eine „ausreichende Geschwindigkeit“ in diesem Zusammenhang de facto eine Echtzeitanforderung. Da der Begriff „Echtzeit“ in verschiedenen Forschungsbereichen aber sehr unterschiedlich verwendet wird, soll er im folgenden als „so schnell wie möglich“ verstanden werden (was im allgemeinen als *weiche Echtzeit* beschrieben wird). Es ist nicht erforderlich, Garantien für eine maximal zu erreichende Bildrate oder minimale Schwankungen in der Bearbeitungszeit zu erreichen, wie sie bei der sogenannten *harten Echtzeit* gefordert werden. In unserem konkreten Fall bedeutet Echtzeit, daß 30 Einzelbilder pro Sekunde verarbeitet werden müssen.

Kameras sind die Hauptsensoren der in dieser Arbeit eingesetzten Fußballroboter, den Ulm Sparrows [83, 84, 137]. Sie besitzen eine ganze Reihe von Vorteilen. Die Spielfelddiagonale ist derzeit etwa 20 Meter lang. Der Ball ist auf diese Distanz im Bild nur noch einige wenige Pixel groß, kann aber durchaus noch als solcher erkannt werden. Außer der hohen Ortsauflösung bieten sie meist auch eine relativ große zeitliche Auflösung mit üblicherweise 30 Bildern pro Sekunde.







Dieser Menge erfassbarer Informationen stehen aber einige nicht zu vernachlässigende Nachteile gegenüber. Der Wichtigste ist der hohe Rechenaufwand, der nötig ist, um alle diese Informationen aus dem Bild zu extrahieren. Deswegen aber nur die einfach zu bestimmenden Farbinformationen zu verwenden ist aber wiederum auch nicht möglich. Farben sind, abhängig von den herrschenden Beleuchtungsbedingungen, starken Schwankungen unterworfen. Schon kleine Änderungen in der Beleuchtungsstärke oder Schwankungen in der Farbtemperatur haben einen großen Einfluß auf die Farbwiedergabe im aufgenommenen Bild. Diese Tatsache stellt alle Forschungsgruppen im RoboCup vor große Probleme und ist deshalb ebenfalls Thema dieser Arbeit.

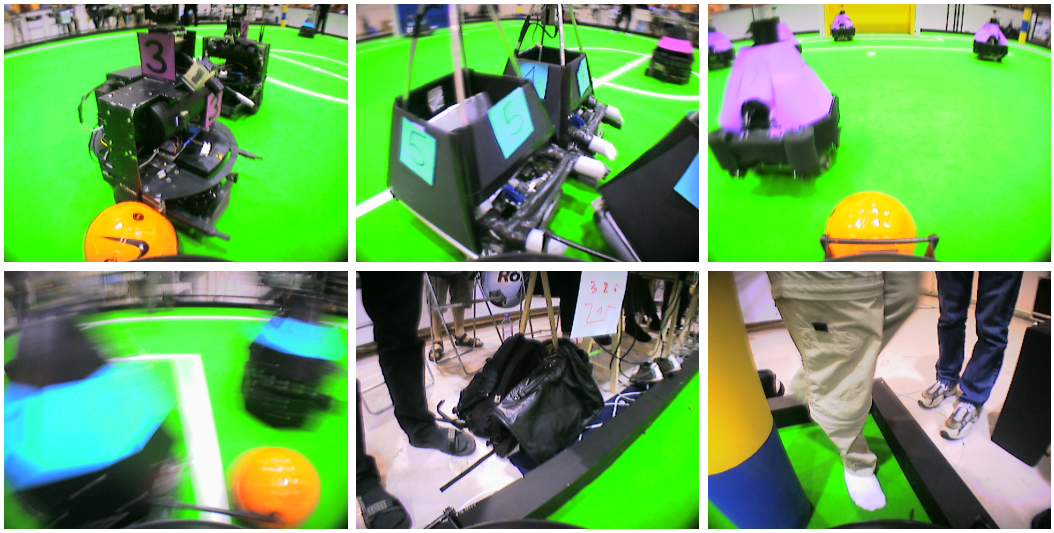
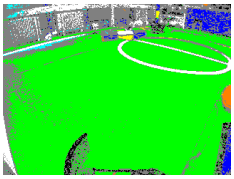


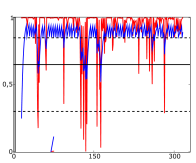
Abbildung 1.1: Roboteraufnahmen als Beispiele für die im RoboCup auftretenden Schwierigkeiten. Die ersten beiden Bilder zeigen die häufig auftretenden Überdeckungen, wie sie beim Kampf um den Ball entstehen. Im dritten Bild (oben rechts) sind besonders die extremen Größenvariationen zu sehen. Mehrere Roboter des gleichen Typs direkt vor dem Roboter oder quer über das halbe Spielfeld unterscheiden sich stark in ihrer Größe. Das nächste Bild unten links illustriert die teilweise starke Bewegungsunschärfe. Weitere Bilder zeigen unbekannte Objekte, wie sie vor allem am Spielfeldrand auftreten, sowie die starke Verzerrung von Objekten, hervorgerufen durch das extrem weitwinklige Objektiv.

Um noch einmal zum Thema Echtzeit und anderen Schwierigkeiten der visuellen Objekterkennung in einer dynamischen Umgebung zurück zu kehren. Das Erkennen von Objekten in einem schnellen und dynamischen Szenario bedeutet nicht nur, mehr Bilder innerhalb kürzerer Zeit zu bearbeiten. Die gesamte Dynamik ändert sich, wenn die Dinge schneller ablaufen. Ein besonderes Problem im RoboCup ist die Bewegungsunschärfe. Durch die physikalisch begrenzte minimale Belichtungszeit und eine sehr hohe Eigenbewegung, insbesondere hohe Winkelgeschwindigkeiten bei Kurvenfahrten oder bei Kameradrehungen, werden Bilder schnell unbrauchbar. Hohe Eigengeschwindigkeiten der zu erfassenden Objekte, speziell die des Balls, verschärfen dieses Problem noch. Um die Geschwindigkeit eines Balls zu berechnen, muß dieser aber nicht nur erkannt werden, sondern eine Geschwindigkeitsschätzung auch mit sehr wenigen, aufeinanderfolgenden Bildern möglich sein.

Im RoboCup sind noch andere Besonderheiten zu beachten. Diese Schwierigkeiten machen ihn zu einer ausgesprochen realistischen Test- und Entwicklungsumgebung für zukünftige Einsatzgebiete. Abbildung 1.1 zeigt einige dieser Probleme. Die Aufnahmen sind mit einem der für diese Arbeit verwendeten Robotern erstellt worden, um einen ersten Eindruck zu geben, wie das Szenario im RoboCup aussieht. Gegenstände im Bild sind oft partiell verdeckt oder am Rand der Aufnahme abgeschnitten.







Die Objekte variieren sehr stark in ihrer Größe und Form und befinden sich meist vor einem unstrukturierten Hintergrund. Die Gegenstände im Bildhintergrund sind in der Regel nicht bekannt. Weitere Schwierigkeiten entstehen durch eine extreme Linsenverzerrung[1].

1.3 Lösungsansatz

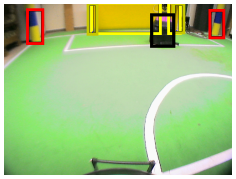
Wie bisher dargestellt wird schnell klar, daß es sich hierbei um ein dreidimensionales Szenario handelt. Merkmale eines solchen Szenarios sind, daß immer nur eine Seite des Objekts sichtbar ist. Außerdem ist eine teilweise Verdeckung von Objekten in diesem Szenarien immer ein großes Problem. Ein Objekterkennungssystem für diese Umgebung arbeitet zwangsläufig immer auf unvollständigen Daten.

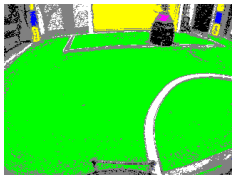
Die in der Literatur beschriebenen Vorgehensweisen für solche Szenarien lassen sich grob in zwei große Gruppen einteilen: Entweder eine objektzentrierte oder eine beobachterzentrierte Repräsentation der zu suchenden Objekte und damit verbunden die jeweils passenden Lösungsschritte. Umfangreiche Übersichtsartikel zu diesem Themenkomplex bieten [10], [19], [57] oder auch [111].

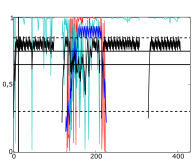
Objektzentrierte, strukturbasierte Ansätze basieren auf einem de-kompositorischen Ansatz. Dabei wird versucht, das Gesamtobjekt durch die dreidimensionale Modellierung einzelner Primitive zu beschreiben. Vorstellen kann man sich das ungefähr wie eine Modellierung in einem CAD-Programm. Die einzelnen Primitive werden oft mit dem Begriff Geons bezeichnet. Ein Problem dieser objektzentrierten Ansätze ist, daß keine Einigkeit besteht, wie diese Dekomposition am Besten und Effektivsten erfolgen soll. Außerdem herrscht Uneinigkeit unter den Psychophysikern, ob ein solcher Ansatz überhaupt biologisch plausibel ist.

Im Gegensatz dazu wird bei beobachterzentrierten oder auch ansichtenbasierten Verfahren überlegt, wie ein Objekt dem Beobachter erscheint, das heißt wie ein dreidimensionales Objekt in seiner zweidimensionalen Abbildung aussieht. Auch hier wird meist wieder unterschieden zwischen dekompositorischen und holistischen Ansätzen. Während beim dekompositorischen Ansatz das zu erkennende Objekt in verschiedene Strukturprimitive zerlegt wird, wird beim holistischen Ansatz das Objekt als Ganzes betrachtet, und die Ähnlichkeit zur repräsentierten Abbildung untersucht.

Der in dieser Arbeit verfolgte Ansatz läßt sich am besten diesem zuletzt beschriebenen, holistischen Ansatz zuordnen. Vorteil einer solchen Vorgehensweise ist, daß kein explizites Modell der zu erkennenden Objekte nötig ist. Konkret werden dazu abstrakte, niederdimensionale Ansichtsrepräsentationen verschiedener Objekte durch unterschiedliche neuronale Klassifikatoren bewertet. Dadurch wird der Aufwand der Zuordnung von Objekteigenschaften zum Objekt quasi in die Lernphase des künstlichen neuronalen Netzes verschoben und in der Ausführung nur noch das Netz in der Vorwärtsphase durchlaufen.







[1] Um einen möglichst großen Sichtbereich zu erhalten, ist die verwendete gerichtete Kamera mit einem besonders weitwinkligen Objektiven (f = 2,8 mm) versehen.

Dazu wurde eine mehrschichtige Architektur verwendet, welche die Teilbereiche Regionensuche, Merkmalsextraktion und neuronale Klassifikation umfasst. Die Verwendung neuronaler Klassifikatoren verspricht hierbei eine hohe Robustheit bei gleichzeitiger Flexibilität des Gesamtsystems. Vorteile einer solchen mehrschichtigen Architektur sind die einfache Austauschbarkeit von Komponenten sowie die leichtere Wiederverwendung bestehender Komponenten in anderen, verwandten Projekten. Dieser Gedanke ist eine der Grundlagen dieser Arbeit und wird in nachfolgenden Kapiteln noch des öfteren thematisiert werden.

Um Mißverständnissen vorzubeugen, soll an dieser Stelle noch erwähnt werden, daß das gesamte System hier nicht als *aktives Sehen* verstanden werden soll, wie es als Begriff von Aloimonos und Bajcsy in ihren Arbeiten geprägt wurde [1, 4]. Zwar ist auch hier die Kamera eine aktive Komponente, wird aber nur dazu eingesetzt, relevante Objekte in Bildbereich zu verfolgen, nicht aber, um aktiv neue Objektansichten des gleichen Objekts zu erfassen.

Im beschriebenen Gesamtsystem werden keine Bildverarbeitungsoperatoren verwendet, die explizit auf Bildfolgen arbeiten, sondern immer nur Einzelbilder ausgewertet. Dieses Vorgehen bedingt, daß Bildinformationen aus aufeinanderfolgenden Bildern sehr präzise miteinander korreliert werden müssen. Eine solch hohe Genauigkeit der Bewegungsschätzung kann in dem beschriebenen Umfeld aber nicht garantiert werden. Um aber trotzdem temporale Informationen nutzen zu können, die, wie später beschrieben, die Klassifikation signifikant verbessern können, wird die Objektverfolgung und die darausfolgende Informationsintegration auf symbolischer Ebene mit Hilfe eines Kalman-Filter-basierten Trackingverfahrens direkt innerhalb des Gesamtsystems realisiert. Dadurch läßt sich die Objekterkennung deutlich stabilisieren, indem beispielsweise vorübergehende Verdeckungen überbrückt oder das Erkennungsrauschen eliminiert werden kann. Zusätzlich läßt sich so das Gesamtsystem beschleunigen.

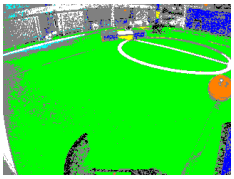
1.4 Ergebnisse

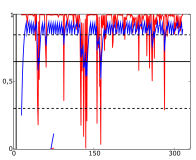
Das im Rahmen dieser Arbeit entwickelte Objekterkennungssystem stellt beinahe das komplette sensorische System der Ulm Sparrows RoboCup-Mannschaft dar. Jedes relevante Objekt auf dem Fußballfeld wird schnell und robust erkannt. Mit diesem System ist es gelungen, als beste deutsche Mannschaft im Turnier auf dem zweiten Platz bei den offenen deutschen Meisterschaften 2003 in Paderborn abzuschließen.

Das System erlaubt es, den Ball und die Tore mit voller Geschwindigkeit, also mit 30 Bildern pro Sekunde, zu erkennen. Alle anderen Objekte wie die Eck- und Torpfosten und die Bodenlinien werden in einer für die Selbstlokalisierung auf dem Spielfeld ausreichenden Geschwindigkeit ermittelt. Diese Selbstlokalisation basiert ausschließlich auf diesen gesehenen Gegenständen und einer kontinuierlichen Bewegungsfortschreibung mittels Radsensoren – der sogenannten Odometrie.

Zusätzlich existieren Methoden zur Erkennung von eigenen und gegnerischen Robotern. Die Komplexität dieser Aufgabe, insbesondere durch die hohe Variabilität der







Roboter bezüglich Größe und Gestalt, läßt sich daran sehen, daß wir die erste Mannschaft in der RoboCup-middle-size-Liga waren, die eine solche Erkennung besaß, die nicht nur auf einer simplen Farbbloberkennung basierte.

Daneben ist dieses gesamte Erkennungssystem noch hinreichend schnell, so daß anderen Modulen noch ausreichend Ressourcen zur Verfügung stehen. Andere besonders rechenaufwendige Module sind die Selbstlokalisation, die Weltmodellierung oder die abschließende Entscheidungsfindung über die endgültige Bewegungsaktion.

Damit sind alle oben kurz skizzierten Problemfälle erfolgreich abgearbeitet worden.

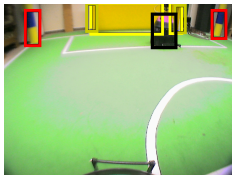
Wie in späteren Kapiteln der vorliegenden Arbeit noch ausführlich dargestellt, ist es gelungen, die Suche nach interessanten Regionen auf dem gesamten Bild sehr schnell zu berechnen. Zusätzlich erlaubt die neuronale Klassifikation eine schnelle und robuste Erkennung aller relevanten Gegenstände auf dem Spielfeld. Durch eine enge Verknüpfung dieser Objekterkennung und eines Moduls zur zeitlichen Integration ließ sich die Erkennungsleistung weiter verbessern und gleichzeitig die benötigte Rechenleistung und durchschnittliche Berechnungsdauer pro Bild weiter reduzieren. Die Flexibilität der in dieser Arbeit entstandenen Lösungen ist dadurch ersichtlich, daß ein Großteil der entwickelten Filter in einem verwandten Projekt mit großem Erfolg eingesetzt werden [77, 28].

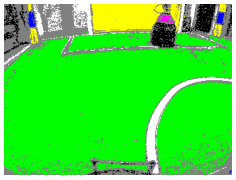
Zusätzlich stellt die Verwendung von realen, während der Spiele gewonnenen Daten zum Training und zur Auswertung der Verfahren einen wesentlichen Unterschied zu vielen anderen, auf diesem Gebiet geleisteten Arbeiten dar. Diese realistischen Daten erlauben eine sinnvolle Bewertung der Leistung des entwickelten Verfahrens.

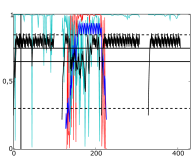
1.5 Überblick

Nachdem dieses Kapitel einen groben Überblick über das Einsatzszenario und die daraus folgenden Probleme gegeben hat, soll diese Analyse im nächsten Kapitel im Detail weiterverfolgt werden. Nach einer kurzen Vorstellung des RoboCup werden insbesondere die momentanen und zukünftigen Beleuchtungsbedingungen vorgestellt. Eine Erläuterung der verwendeten technischen Infrastruktur gibt weitere Einblicke in die Voraussetzungen für diese Arbeit.

In Kapitel 3 wird dann der Lösungsansatz im Gesamten vorgestellt. Darauf folgt eine detaillierte Erklärung der einzelnen dort eingeführten Schichten. Kapitel 4 beschreibt die softwaretechnischen Grundlagen, auf denen diese Arbeit beruht. Obwohl diese Softwarebasis nicht direkt Teil der vorliegenden Arbeit ist, ist die Vorstellung von Konzepten wie Filtern oder Metadaten für das weitere Verständnis Voraussetzung. Kapitel 5 beschreibt die Suche nach interessanten Regionen im Bild, Kapitel 6 die Merkmalsberechnung und die Objektklassifikation mit Hilfe künstlicher neuronaler Netze. In Kapitel 7 schließlich wird das verbindende Element, die zeitliche Integration der Sicherheitswerte, im Detail erklärt. Nach einer Erläuterung der Kalman-basierten Filterung und der verwendeten Multi-target-Objektzuordnung wird die Integration in das Erkennungssystem beschrieben. Wie diese extrahierten Informationen innerhalb des Gesamtsystems verwendet werden und wann die jeweiligen Informationen



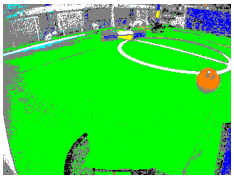


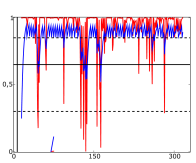


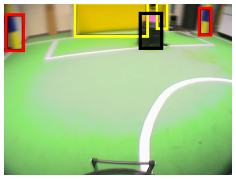
ausgewertet werden, ist Thema des Kapitels 8. Da zu jedem der in diesen Kapiteln vorgestellten Teilbereichen bereits viele Arbeiten existieren, ist in jedem Kapitel ein eigener Abschnitt „verwandte Arbeiten“ enthalten. In diesen Abschnitten wird die vorliegende Arbeit im Vergleich zu diesen Lösungen diskutiert und die getroffenen Entscheidungen begründet.

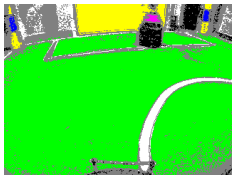
Eine umfangreiche Auswertung der einzelnen Komponenten sowie der Leistungsfähigkeit des Gesamtsystems erfolgt in Kapitel 9. Jedes Modul wird daraufhin untersucht, wie robust und flexibel es ist. Abschließend werden in Kapitel 10 die Ergebnisse mit dem verglichen, was als Ziel für diese Arbeit ausgegeben wurde. Eine Beschreibung der noch offenen Probleme und Ideen für zukünftige Verbesserung und Entwicklungsrichtungen runden die Arbeit ab.

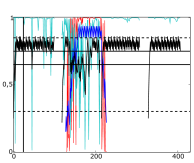











0,5
0
0
200
400

2 Problemanalyse

Objekterkennungssysteme sind in der Regel stark durch ihr jeweiliges Einsatzszenario geprägt. Insbesondere die Objekterkennung in hochdynamischen Systemen muß zwangsläufig die vorherrschenden Eigenschaften der Umgebung in die Entscheidungsfindung mit einbeziehen. Viele der auftretenden Probleme sind aber nicht auf ein spezielles Einsatzgebiet beschränkt, sondern treten so oder so ähnlich in allen denkbaren Szenarien wieder auf. Im folgenden Kapitel soll deswegen eine detaillierte Beschreibung der Umgebung gegeben werden, in der diese Arbeit entwickelt wurde. Dabei wird auch diskutiert, weshalb der RoboCup auch eine ideale Entwicklungsumgebung für andere, realweltliche Einsatzgebiete ist.

Im folgenden wird erst der RoboCup selbst vorgestellt. Hier sind insbesondere die vorherrschenden Beleuchtungsbedingungen von Interesse, weshalb ihnen ein extra Abschnitt gewidmet ist. Danach werden die technischen Daten der verwendeten Roboter im Detail erläutert, wie die Bilder aufgezeichnet werden und wie dies die verwendeten Algorithmen beeinflusst. Zum Schluß werden die Regeländerungen im RoboCup der letzten Jahre vorgestellt.

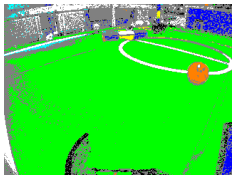
2.1 Grundlegende Rahmenbedingungen im RoboCup

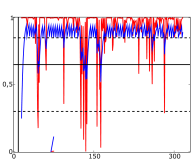
Für alle Leser, denen der RoboCup nicht vertraut ist, soll dieser hier kurz vorgestellt werden. In den verschiedenen Ligen des RoboCup spielen Teams autonomer, mobiler Roboter oder Agenten selbstständig Fußball gegeneinander.

Die Liga, in der diese Arbeit entstanden ist, ist die sogenannte middle-size-Liga. Hier spielen vier bis sechs Roboter (drei bis fünf Feldspieler plus ein Torwart) gegen ein anderes Team. Alle Spieler haben jeweils ihr gesamtes Repertoire an Sensoren selbst an Bord. Ein globaler Sensor ist nicht erlaubt. Die besonderen Herausforderungen in der middle-size-Liga bestehen hauptsächlich in der Sensordatenverarbeitung sowie in der Integration des Gesamtsystems. Solche aufwendigen Roboter ergeben ein sehr komplexes Gesamtsystem.

Eine andere noch vergleichbare Liga ist die Sony-Legged-Liga, in der sogenannte Aibo-Roboter der Firma Sony gegeneinander antreten. Aibo-Roboter sind hundeartige, laufende Roboter. Auch hier ist die ganze Sensorik in den Roboter integriert, wobei sich die Sensorausstattung aber auf eine gerichtete Kamera und Neigungs- und Winkelmesser an den Füßen der Roboter beschränkt. Das Spiel selbst ist durch die begrenzte Geschwindigkeit der Roboter nur von geringer Dynamik. Der Spielfeldaufbau und damit die Sensoreindrücke der Roboter sind aber durchaus mit denen in der middle-size-Liga vergleichbar.







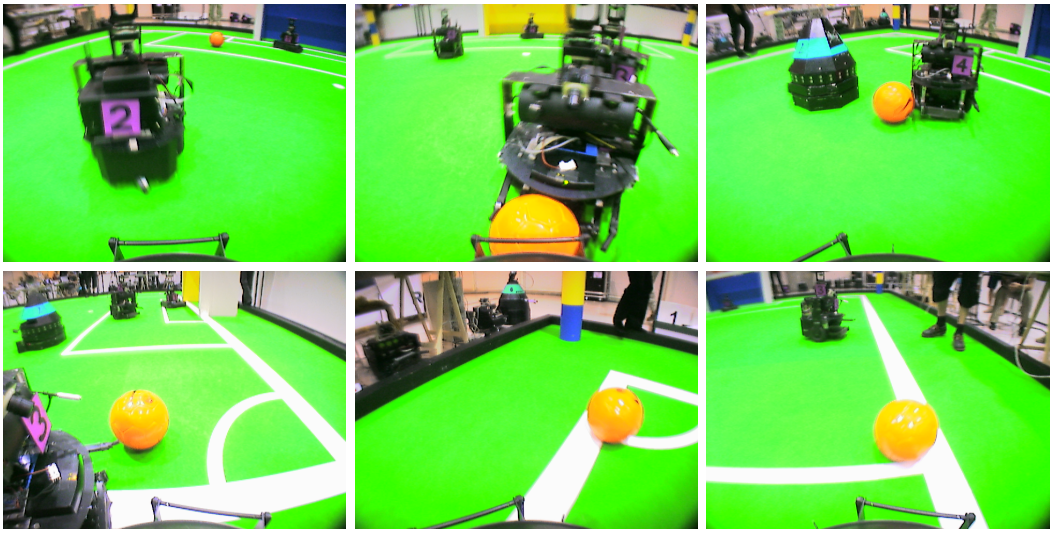
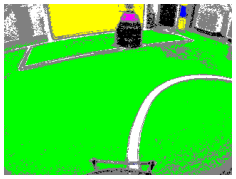


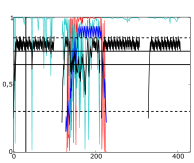
Abbildung 2.1: Bilder der Objekte auf dem Spielfeld aus der Sicht des Roboters. In den ersten beiden Bildern sind die zwei verschiedenen Tore zu sehen. In Bild 3 und 4 sind die verschiedenen Farbmarkierungen an der Oberseite der Roboter deutlich zu erkennen. In der fünften Aufnahme ist einer der Eckpfosten im Bild. Im Letzten sieht man, wie gleichzeitig Objekte sowohl auf dem Spielfeld als auch solche neben dem Feld in derselben Aufnahme auftauchen können.

Die Regeln der middle-size-Liga sind ähnlich denen des richtigen Fußballs, erweitert um ein paar zusätzliche Regeln, welche auf die Besonderheiten des Roboterfußballs eingehen. So gibt es erst seit 2005 Pendants zu Einwürfen und Freistößen. Zusätzliche Regeln befassen sich mit roboterspezifischen Besonderheiten wie beispielsweise der Frage, wie weit der Roboter den Ball umfassen darf, oder welche Mechanismen zum Ballführen oder -annehmen erlaubt sind.

Alle Gegenstände auf dem Spielfeld sind farbkodiert: Der Ball ist orange, der Boden grün mit weißen Begrenzungslinien, die Tore blau und gelb und die Eckfahnen abwechselnd blau und gelb. Die Roboter müssen zum größten Teil schwarz sein und eine magenta- oder cyanfarbene Markierung besitzen, welche die Teamzugehörigkeit festlegt. Das Spielfeld ist nur durch eine kleine, sehr niedrige Bande begrenzt, welche die Roboter davon abhält, aus dem Feld zu fahren. Direkt um das Spielfeld können sich Zuschauer, Tische oder abgestellte Gerätschaften der verschiedenen Mannschaften befinden. Einige Beispiele aus der Roboterperspektive sind in Abbildung 2.1 zu sehen.







Zusätzlich zu den Farben sind auch die Größen der einzelnen Objekte streng begrenzt. Roboter dürfen maximal 80 cm hoch sein und eine maximale Breite bzw. Tiefe von 50 cm besitzen. Zusätzliche Beschränkungen sowie praktische und physikalische Gründe schränken die Formenvielfalt weiter ein[1]. Die Größe aller anderen Objekte ist ebenfalls im Detail festgeschrieben. Die Tore sind 2 Meter breit und einen Meter

[1] Die genauen Regeln kann der interessierte Leser in [21] finden.

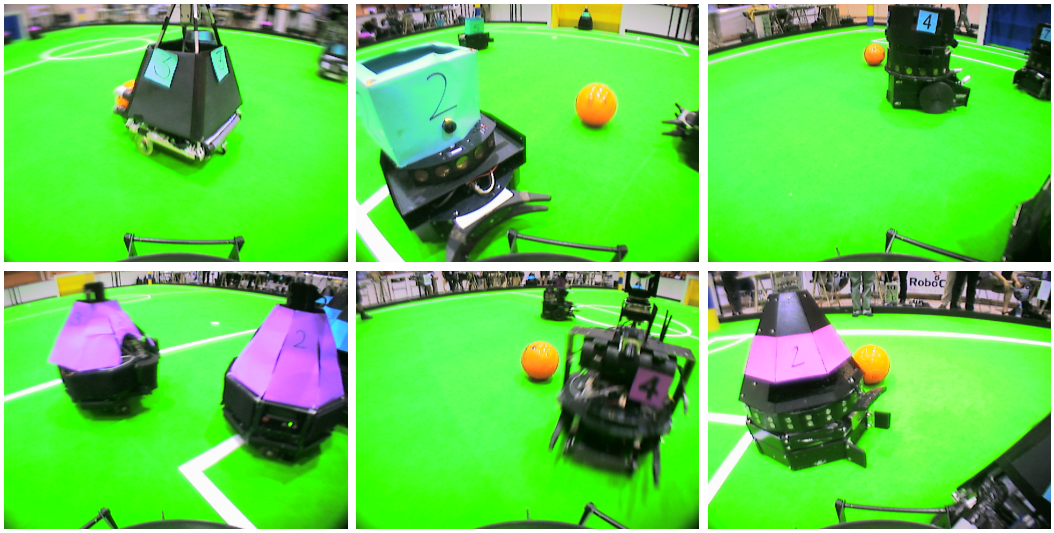


Abbildung 2.2: Beispiele für verschiedenen Robotertypen, wie sie an der RoboCup-middle-size-Liga teilnehmen. Von links nach rechts und oben nach unten: Roboter der Abteilung AIS des Fraunhofer Instituts, der TU München, der Universität Delft, dem japanischen Team Trakkies der Universität Osaka, der Universität Tübingen und der Universität Graz. Diese Robotertypen, sowie unsere eigenen Roboter, sind auch für das Training der Robotererkennung, wie sie in den folgenden Kapiteln beschrieben wird, verwendet worden.

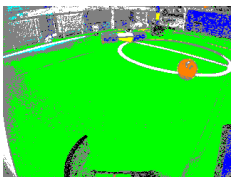
hoch. Die Eckfahnen sind 90 cm hoch und haben einen Durchmesser von 20 Zentimetern. Das Spielfeld ist im Moment 16 Meter lang und 12 Meter breit (plus einen ein Meter breiten Rand, bevor die Bande beginnt). Alle diese Größen und Beschränkungen verändern sich jedoch, wie bereits erwähnt, von Jahr zu Jahr (mehr darüber in Abschnitt 2.4).

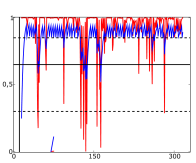
Problematik

Ausgehend von der obigen Beschreibung läßt sich rekapitulieren, welche praxisrelevanten Probleme im RoboCup existieren. Während Objekte, welche zum Spielfeld gehören wie Eckfahnen, Tore und Ball noch recht genau definiert sind und damit eine optimale Anpassung der Erkennungsalgorithmen im Vorfeld möglich ist, sind die Eigenschaften gegnerischer Roboter nur noch in weiten Grenzen spezifiziert. Zwar ist die Gestalt der Maschinen einer gewissen Konsolidierung unterworfen, trotzdem variiert ihr Äußeres stark. In Abbildung 2.2 sind einige Robotertypen dargestellt, wie sie bei den Weltmeisterschaften in Lissabon 2004 verwendet wurden.

Eine Herausforderung sind aber nicht nur diese grob definierten Objekte, sondern auch und vor allem die nicht zum Spiel gehörigen. Rund um das Spielfeld können beliebige Gegenstände vorhanden sein: Personen, Tische, Kabeltrommeln, Kisten mit Material der teilnehmenden Mannschaften und vieles mehr. Besonders kleine Kinder tragen erstaunlich oft orange Hosen mit gelben T-Shirts. Die tatsächlichen, relevanten







Gegenstände, wie der orange Ball und das gelbe Tor, von diesen Objekten abzugrenzen erzwingt eine Abwägung zwischen Spezifität und Variabilität der verwendeten Algorithmen.

Der zweite Problembereich ist die hohe Geschwindigkeit der Objekte auf dem Spielfeld. Viele Roboter, darunter unsere Eigenen, die Ulm Sparrows, können bis zu 3 Meter pro Sekunde fahren. Durch den eingebauten Schußmechanismus kann der Ball sogar noch stärker beschleunigt werden. Um in dieser Umgebung die Gegenstände noch ausreichend sicher detektieren zu können, müssen mindestens 10 bis 15 Bilder pro Sekunde verarbeitet werden. Da für eine schnelle und valide Geschwindigkeitsschätzung aber mehrere aufeinanderfolgende Bilder benötigt werden, ist eine noch höhere Bildrate wünschenswert.

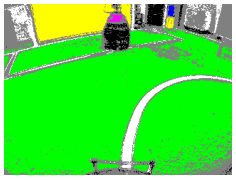
Durch die hohe Dynamik verschlechtern sich aber auch die gemessenen Odometriewerte stark. Starke Beschleunigungen, abrupte Bremsmanöver, hohe Kurvengeschwindigkeiten und teils harte Kontakte mit anderen Robotern können die Ergebnisse der Messung beträchtlich von der tatsächlich zurückgelegten Wegstrecke abweichen lassen. Da diese Messungen für eine zeitliche Integration benötigt werden, müssen die gewählten Objektverfolgungs- und -prädiktionsverfahren sehr robust gegen solche, stark verrauschten Daten sein.

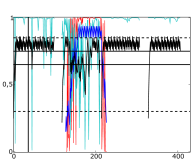
2.2 Farben und Beleuchtung

Im Moment werden Spiele der middle-size-Liga in einem Umfeld mit genau kontrollierten Beleuchtungsbedingungen abgehalten [70]. Die momentan gültigen Regeln (Stand 2006) schreiben eine Lichtintensität zwischen 300 und 1200 Lux vor. Ausgehend von der dann eingestellten Intensität darf die Schwankung innerhalb eines Spielfelds dann aber nur 300 Lux betragen. Weitere Regelungen, z.B. zur Farbtemperatur, existieren nicht [21]. Es wird außerdem besonders darauf geachtet, daß die Beleuchtung möglichst diffus ist, um Glanzlichter auf glatten Oberflächen, wie etwa dem Ball, zu vermeiden. Außerdem wird dadurch vermieden, daß die Scheinwerfer direkt in die Kameras leuchten.

Während die Beleuchtung im Voraus nicht genau bekannt ist, bleibt sie aber während eines gesamten Turniers konstant. Mannschaften können zu Beginn eines Wettkampfs einmalig eine Farbkalibrierung ihrer Kameras vornehmen und beispielsweise Farbzuordnungstabellen vor Ort optimieren. Solche Farbzuordnungstabellen müssen dann nur noch mit den verbleibenden Variationen, die auf einem Spielfeld herrschen, zurechtkommen. Umgekehrt können solche statischen Verfahren aber nicht mit Variationen außerhalb der erwarteten Varianzen umgehen.







Spiele unter solch definierten Beleuchtungsbedingungen abzuhalten ist aber natürlich nicht im Sinne der langfristigen RoboCup-Vision, nämlich in noch nicht einmal 50 Jahren ein Fußballspiel gegen den amtierenden menschlichen Weltmeister zu gewinnen. Aus diesem Grund müssen die Spiele früher oder später ins Freie verlagert werden oder zumindest unter natürlichen Beleuchtungsbedingungen abgehalten werden. Ein pragmatischer Grund ist auch, daß es für die Organisatoren eines Turniers

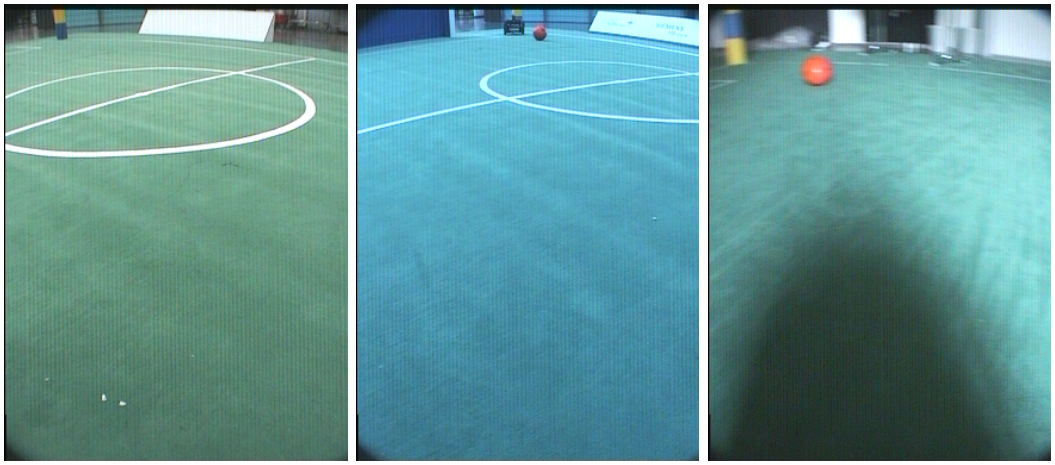


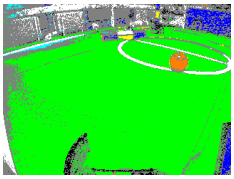
Abbildung 2.3: Beispiele für drei verschiedene Beleuchtungsbedingungen, wie sie für die Experimente benutzt wurden. Im linken Bild Neonlicht während eines Freundschaftsspiels gegen das Münchner RoboCup Team. In der Mitte dasselbe Spielfeld bei natürlichem Sonnenlicht. Rechts ebenfalls natürliches, aber stark gerichtetes Licht in unserem Labor. Die verwendeten Tore, Eckfahnen, Ball und Teppich sind in allen Bildern identisch.

mitunter sehr aufwendig ist, solche Beleuchtungsbedingungen bereitzustellen. Große Hallen haben meist lange Fensterfronten, die nur mit erheblichem Aufwand abgeklebt werden können.

Aus diesem Grund wurde bereits vor einiger Zeit diskutiert, ob Spiele zukünftig nicht bei Tageslicht abgehalten werden sollen. Ein erster Schritt wurde auf den diesjährigen Weltmeisterschaften 2006 in Bremen getan, da hier eine komplette Seite der Halle eine offene, nicht abgedunkelte Fensterfront aufwies. Um eine fundierte Datenbasis für diese Diskussion bereitzustellen, haben Hans Utz, Gerhard Kraetzschmar und ich eine umfangreiche Auswertung vorgenommen, welchen Einfluß wechselnde, natürliche Beleuchtungsbedingungen auf solche statischen Farbzuordnungstabellen haben [94]. Die wichtigsten Resultate daraus sind in Abschnitt 9.1 zu sehen. Um aber einen ersten Eindruck zu vermitteln, sind einige Beispielbilder in Abbildung 2.3 dargestellt. Auf den Bildern sind verschiedene künstliche und natürliche Beleuchtungen zu sehen. Das verwendete Spielfeld, Teppich, Ball, Tore und Eckfahnen sind in allen Bildern identisch, um nur den Einfluß des Lichts zu illustrieren.

Problematik





Die unterschiedliche Wahrnehmung der Farben unter wechselnden Beleuchtungsbedingungen ist ein großes Problem in realweltlichen Einsatzgebieten. Versuche in unserem Labor haben gezeigt, daß der Einsatz von Farbkonstanzalgorithmen auf den uns zur Verfügung stehenden Rechnern in Echtzeit nicht realisierbar sind. Wie in Abschnitt 5.4 im Detail beschrieben, arbeiten viele Gruppen folglich an adaptiven Verfahren zur Zuordnung von wahrgenommenen Farben zu definierten Farbklassen.

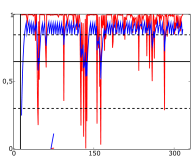




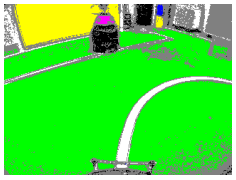
Abbildung 2.4: Auf der linken Seite ist ein Sparrows-2003-Roboter zu sehen, auf denen fast die gesamte Auswertung für diese Arbeit gemacht wurde. Auf der rechten Seite ist das ältere Modell, der Sparrows-99 abgebildet. Einige der Testbilder für die Evaluation der Beleuchtungsbedingungen wurden mit diesem Robotertyp aufgezeichnet.

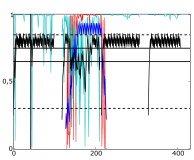
Für das Fußballspielen unter natürlichen Beleuchtungsbedingungen wird das aber voraussichtlich nicht ausreichen, sondern bedarf grundlegend anderer Verfahren.

Die hohe Variabilität der Lichtstärke beeinflusst aber nicht nur die Farbwahrnehmung, sondern übersteigt oftmals auch die dynamische Bandbreite heute verfügbarer Kameras. Fällt die Beleuchtungsstärke zu stark ab, steigt sie über ein bestimmtes Niveau oder ist im Bild allgemein eine zu hohe Dynamik, sind handelsübliche Kameras überfordert. Homogen dunkle oder überstrahlte Bereiche im Bild oder zumindest ein starker Anstieg des Kamerarauschens sind die Folge. In einer Arbeit aus dem Jahre 2002 rechnen Bertozzi et. al. [9] vor, daß heute übliche Kameras einen Intensitätskontrast von etwa 500 : 1 liefern. Für Arbeiten im Außenbereich auf einzelnen Bildern wären aber eher Kontrastbereiche von 10000 : 1, für schnelle Bildfolgen gar 100000 : 1 nötig.

2.3 Die Roboter







In diesem Abschnitt wird die Roboterplattform vorgestellt, auf der diese Arbeit entstanden ist. Obwohl im Laufe der Entstehung dieser Arbeit mehrere Plattformen, nämlich die Sparrows-99 und die Sparrows-2003-Roboter verwendet wurden, ist fast die gesamte Auswertung auf den Sparrows-2003 erfolgt. Aus diesem Grunde sollen diese hier im Detail vorgestellt werden (eine weit detailliertere Beschreibung findet sich in [109, 118]). Die Sparrows-2003 (Abbildung 2.4) sind in unserem Labor an der Universität Ulm entworfen und gebaut worden. Der Roboter ist mit einem Differential-



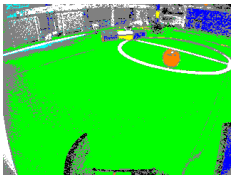
Abbildung 2.5: Großaufnahme des Kamerasystems der Sparrows-2003. Der obere Teil zeigt die gerichtete Kamera auf der Schwenkeinheit montiert. Im unteren Teil ist das omnidirektionale System zu sehen. Diese Kamera erlaubt eine 360°-Rundumsicht um den gesamten Roboter.

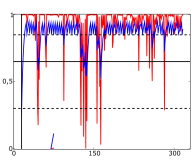
antrieb mit zwei unabhängig voneinander angetrieben Rädern und zwei Schleifrädern ausgestattet. Der Roboter hat eine maximale Geschwindigkeit von etwa 3 Metern pro Sekunde und eine maximale Beschleunigung von ungefähr 1,8 m/s^2. Für seinen Einsatz als Fußballspieler besitzt er außerdem einen pneumatischen Kicker. Rund um den Roboter ist ein Ring von 16 Infrarotsensoren für die Hindernisvermeidung in unmittelbarer Umgebung angebracht. Diese decken genau den Bereich rund um den Roboter ab, den die Kameras nicht einsehen können.

Neben den erwähnten Infrarotsensoren sind zwei verschiedene Kameras oben auf dem Roboter angebracht. Diese Kameras sind neben den IRs die einzigen Sensoren des Roboters und mit Abstand die Wichtigsten. Im Gegensatz zu einem eindimensionalen Abstandsmaß liefern Kameras eine wesentlich größere Anzahl verschiedenartiger Informationen wie räumliche Lage im Bild, Farbe oder Gestalt über einen wesentlich größeren Entfernungsbereich.

Die erste Kamera ist eine normales, gerichtetes Modell, das oben auf der Spitze des Roboters auf einer Dreheinheit, dem sogenannten Pan, montiert ist (Abbildung 2.5). Die Kamera ist eine SONY DFW V500, wie sie standardmäßig im Handel erhältlich ist und auch von vielen anderen Gruppen im RoboCup verwendet wird. Sie ist ausgestattet mit einem $1/3''$-CCD-Chip. Zusammen mit dem verwendeten 2,8-Millimeter Objektiv ergibt das einen ungefähren Öffnungswinkel von etwa 90° in der Horizontalen und circa 60° in der Vertikalen. Das erlaubt uns, die Kamera so zu montieren, daß gleichzeitig Gegenstände sowohl direkt vor dem Roboter als auch quer über das gesamte Feld gesehen werden können. Der weite Öffnungswinkel des Objektives führt aber auch zu einer extremen Verzerrung der Bilder an den Bildrandbereichen (siehe Abbildung 1.1).





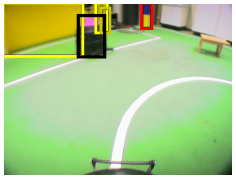


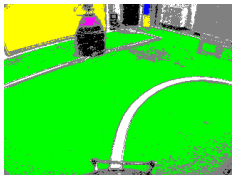
Die meisten der erhältlichen Bildbearbeitungsbibliotheken arbeiten standardmäßig auf rechtwinkligen Bildausschnitten. Würde man diese Linsenverzerrungen deswegen nun explizit herausrechnen, dann würden große Teile des Bildes durch die entstehende Kissenstruktur wieder aus dem Bild herausragen. Das wiederum würde die Vorteile des großen Öffnungswinkels fast komplett wieder zunichte machen. Aus diesem Grund wird die Linsenentzerrung nicht direkt im Kamerabild korrigiert, sondern erst bei der Projektion von Gegenständen aus dem Bild in das lokale, roboterzentrische Koordinatensystem. Das bedeutet natürlich auch, daß die verwendeten Objekterkennungsverfahren robust gegenüber solch verzerrten Bildern sein müssen.

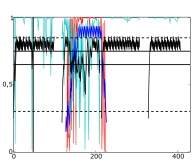
Die zweite verwendete Kamera ist ein einfache Logitech-Webcam. Diese Kamera ist unterhalb eines konkaven Spiegels mit Blickrichtung nach oben angebracht, was eine 360° Rundumsicht ermöglicht. Der Spiegel ist so konstruiert, daß er in radialer Richtung eine lineare Abbildung der Umgebung auf dem Spiegel selbst liefert. Der Sichtbereich dieses Systems reicht von minimal 40 Zentimetern bis maximal etwa 2,5 Metern. Die minimale Distanz garantiert, daß sich der Roboter *nicht* selbst sehen kann. Der maximal mögliche Abstand wurde gewählt, da die Hauptaufgabe dieser Kamera die Hindernisvermeidung im mittleren Entfernungsbereich darstellt. Dies wurde notwendig, da die erreichbare Maximalgeschwindigkeit von 3 Meter pro Sekunde mit den Infrarotsensoren, welche eine maximale Erkennungweite von etwa 40 Zentimetern besitzen, nicht mehr zu kontrollieren ist. Bei eben diesen 3 m/s und 40 Zentimetern Bremsweg müsste der Roboter mit $a = s/(s/v)^2 \cdot 2 = 45$ m/s^2 Bremsbeschleunigung reagieren, was definitiv nicht möglich ist. Außerdem sind in dieser Rechnung die begrenzte Auswertungsgeschwindigkeit der Sensoren mit 30 Hz sowie die interne Latenz durch das System noch gar nicht berücksichtigt. Durch die Erweiterung auf 2.5 Meter Bremsweg sinkt dieser Wert auf realistische 7 m/s^2.

Der Spiegel wurde außerdem so konstruiert, daß die Bildbereiche, die nahe am Roboter sind, im Spiegel außen abgebildet werden. Weiter entfernte Gegenstände werden auf den inneren Bereich projiziert. Dadurch ist die höchste Ortsauflösung des Spiegels im Nahbereich rund um den Roboter und nimmt mit zunehmender Entfernung zum Roboter ab, was für den gewählten Einsatzzweck in der Hindernisvermeidung oder auch für eine spezielle Ballerkennung während des Balldribblings optimal ist. Sonst liegt die Hauptlast der Objekterkennung aber bei der gerichteten Kamera.

Die Kameras sind über unterschiedliche Bussysteme an den Rechner angeschlossen, um eine maximale Bildwiederholrate beider Kameras zu garantieren. Die gerichtete SONY Kamera ist über Firewire (auch unter den Namen IEEE 1394 oder iLink bekannt) mit dem Computer verbunden. Die maximal mögliche Geschwindigkeit dieses Systems von 400 MBit/s erlaubt 30 Bilder pro Sekunde bei einer Auflösung von 640×480 Pixeln. Dazu wird das Bild in einem verlustbehafteten YUV-4:2:2-Farbmodell übertragen. Die Logitech-Kamera besitzt nur einen USB1-Anschluß. Dieses recht langsame Bussystem ermöglicht 30 Bilder pro Sekunde mit einer Auflösung von 320×240 Bildpunkten bei Verwendung des YUV-4:2:0-Farbmodells[2].







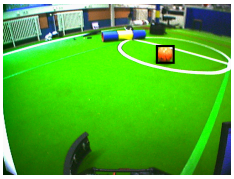
[2] In Wirklichkeit ist das Ganze noch aufwendiger, da ein USB1-Anschluß auch für diese verringerte Auflösung eigentlich noch zu langsam ist. Daher wird das Bild durch den Treiber kamerasei-

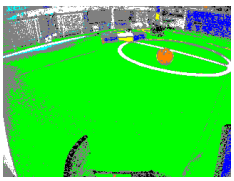
Beide Kameras sind mit einem IBM X31 Sub-Notebook verbunden. Dieser Rechner ist mit einem nicht mehr ganz aktuellen 1,4 GHz Intel Pentium M Prozessor (auch bekannt als Centrino) ausgestattet. Da der Rechner mit 512 MB RAM bestückt ist, sollte der Speicherverbrauch nicht der limitierende Faktor sein (wobei sogenannte "cache-misses" natürlich trotzdem möglichst vermieden werden sollten). Alle aufgeführten Ergebnisse (siehe Kapitel 9) zu Rechengeschwindigkeit und Berechnungsdauer wurden auf einem solchen Laptop ermittelt. Die zur Verfügung stehende Rechenleistung darf aber nicht komplett von der Bildverarbeitung verbraucht werden. Andere gleichzeitig laufende Prozesse im Gesamtsystem, wie etwa die Selbstlokalisation, benötigen ebenfalls einen nicht unerheblichen Teil der Ressourcen.

Problematik

Die heute verfügbaren CCD-Chip-Kameras sind leider nicht optimal für den Einsatz auf autonomen, mobilen Robotern geeignet. Insbesondere ihr hoher Lichtbedarf und die dadurch nötige lange Belichtungszeit führen schnell zu Bewegungsunschärfe im Bild. Besonders bei hohen Winkelgeschwindigkeiten, wie sie bei Kurvenfahrten oder beim Drehen der Kameraeinheit auftreten, muß unter Umständen auf eine Auswertung der Bilder komplett verzichtet werden. Die verwendeten billigen und sehr kleinen Bildsensoren erlauben es aber auch nicht, die Empfindlichkeit des Sensors zu erhöhen, da sonst die Rauschneigung stark zunimmt. Zusätzlich neigen solche Kameras zu starken Blooming-Effekten an Stellen starker lokaler Helligkeitsänderungen im Bild. Dieses Übersprechen läßt leicht den Übergang zwischen Boden und Bodenlinien verwischen. Abbildung 2.6 verdeutlicht einige dieser Kameralimitierungen.

Zusätzlich sind Probleme zu beobachten, die so oder so ähnlich auch in anderen, realweltlichen Einsatzgebieten auftreten. Durch die Verwendung von mehreren unterschiedlichen Sensoren entstehen teilweise widersprüchliche Daten für denselben Sachverhalt. Im Frontbereich der Sparrows-2003-Roboter überlappen etwa die verschiedenen Kamerasysteme. Sollen Informationen beider Sensoren berücksichtigt werden, muß ein Sensorfusionsalgorithmus die Daten entsprechend ihrer Validität entweder korrekt miteinander verrechnen oder zumindest entscheiden, welches Datum jeweils verwendet werden soll. Ähnlich verfährt man, wenn sich verschiedene Detektoren gegenseitig ergänzen sollen, wie etwa bei den Sparrows-2003 beim Übergang zwischen Kamerabild und Infrarotwerten. Falls Daten im Bereich dieser Nahtstelle verwendet werden, muß durch geeignete Maßnahmen sichergestellt sein, daß unterschiedliches Sensorrauschen oder unterschiedliche Genauigkeiten die verwendeten Algorithmen nicht verwirren.





2.4 Regeladaption

Um den wissenschaftlichen Fortschritt im RoboCup und damit auch in der gesamten Robotik voranzutreiben, werden die Regeln für das Roboterfußballspiel regelmäßig

tig noch jpeg-kodiert und erst im Rechner wieder in ein normales Bild zurückgewandelt – was wiederum die Latenz bis zur Bildauslieferung deutlich erhöht.

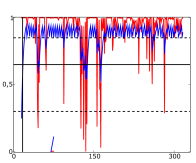
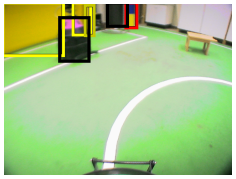


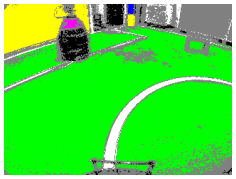


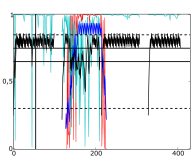
Abbildung 2.6: Einige Beispielbilder für besonders kritisches Verhalten heute verfügbarer CCD-Chip-Kameras. Oben links Bewegungsunschärfe, hervorgerufen durch den meist recht hohen Lichtbedarf dieser Kameras. Rechts oben ein stark verrauschter Bildausschnitt, nachdem die Verschlußzeit verkürzt, gleichzeitig aber die Empfindlichkeit erhöht wurde. Zweite Reihe links: Testbild eines besonders großen Spielfelds. Aufgrund begrenzter Ortsauflösung ist die eigentlich in der Mitte des Spielfelds vorhandene weiße Linie nicht zu sehen. Außerdem ist die Abschattung zum Bildrand hin, die sogenannte Vignettierung, deutlich erkennbar. Unten rechts ein Ausschnitt aus diesem Bild, der die entstehenden Artefakte bei einer YUV-4:2:2-Kodierung gut verdeutlicht (siehe dazu Abschnitt 5.1.1).

verändert und die Randbedingungen von Jahr zu Jahr erschwert. Die meisten der Änderungen in den letzten Jahren betrafen das sensorische System der Roboter.

Die wichtigste Veränderung war das Entfernen der Wände rund um das Spielfeld. Während die ersten Turniere im RoboCup noch mit einer soliden, hohen Wand um die Spielfelder ausgetragen wurden, ist inzwischen nur noch eine kleine, niedrige Bande um das Feld angebracht, welche die Roboter am Verlassen des Spielfelds hindert. Wo vorher nur Objekte gesehen werden konnten, die auch tatsächlich zum eigentlichen Fußballspiel gehörten, sind nun alle Zuschauer und andere Objekte, teilweise sogar Feldmarkierung von benachbarten Spielfeldern, zu sehen. Das macht nicht nur die Objekterkennung um einiges schwieriger, sondern auch Verfahren unbrauchbar, die auf der Annahmen basierten, daß alle gesehenen Gegenstände zum Spiel gehören [95].







Ein weiterer, die Sensorik betreffender Punkt ist, daß die Spielfeldgröße in mehreren Schritten um einige Meter in beide Richtung verlängert wurde. Statt der ursprünglichen 4×7 Meter (Stand 2000) misst es nun bis zu 12×16 Meter (Stand 2006). Viele der bestehenden Kamerasysteme, insbesondere omnidirektionale Kameras [102], stoßen durch ihre begrenzte Ortsauflösung damit langsam an ihre Grenzen.

Wie in Abschnitt 2.2 bereits erwähnt, wurden auch die Beleuchtungsbedingungen über die Zeit immer mehr de-reguliert. Während die Beleuchtung während eines kompletten Turniers immer noch konstant gehalten wird, ist der maximal erlaubte Beleuchtungsunterschied innerhalb eines Spielfelds mehrmals erhöht worden. Diskussionen, ob zukünftig Spiele im Freien unter natürlichen Lichtverhältnissen ausgetragen werden sollen, sind bereits im Gange (wenn auch im Moment noch als zu schwierig abgelehnt – verschiedene Testspiele und Demonstrationswettbewerbe (auch von uns) sind aber bereits ausgetragen worden).

Obwohl die Bilderkennung immer schwieriger wird, wird die Einhaltung der Regeln immer strenger überwacht und die Schiedsrichter sind angehalten, grobe Fouls, wie etwa zu starker Kontakt der Roboter untereinander, strenger zu ahnden. Um während eines Spiels nicht zu viele Roboter durch Platzverweise zu verlieren, sind die Teams gezwungen, ihre Objekt- und Hindernisvermeidung immer weiter zu verfeinern.

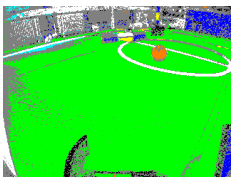
Durch die Einführung von Standardsituationen wie Freistöße, Eckbälle und Einwürfe, die – im Widerspruch zu den FIFA-Regeln – alle indirekt ausgeführt werden, das heißt es darf durch einen Einwurf nicht direkt ein Tor erzielt werden, sollen nun auch die motorischen Fähigkeiten der Roboter und das Zusammenspiel untereinander verbessert werden. Diese Änderungen erzwingen aber auch einen Fortschritt in der Weltmodellierung. Da die funkbasierte Vernetzung der Roboter während eines Turniers selten reibungslos funktioniert, sind die Roboter gezwungen, aus eigenen Beobachtungen möglichst valide Weltmodelle zu extrahieren. Auch wenn Standardsituationen wie ein Eckstoß von einem elektronischen Schiedsrichter angekündigt werden, müssen sich die Roboter für das dann folgende Paßspiel in ausreichender Genauigkeit untereinander detektieren und ihre gegenseitigen Absichten, wie Paßspiel und Ballannahme, erkennen.

Problematik

Durch die ständigen Veränderungen sind alle Mannschaften nicht nur dazu gezwungen, ihre Verfahren immer wieder an die geänderten Umgebungen anzupassen. Es zwingt quasi nebenher dazu, Software und Algorithmen zu entwickeln, die sich schnell und leicht an geänderte Umgebungsbedingungen und Regeln anpassen lassen.

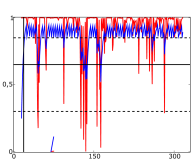
Das entspricht auch den Zielen, welche sich die RoboCup Community selbst gesetzt hat. Schließlich sollen nicht primär Lösungen für das Fußballspiel entwickelt werden, sondern die Robotik an sich vorangebracht werden. Einige der im RoboCup entstandenen Lösungen scheinen sich hier zu stark auf das jeweilige Szenario zu konzentrieren. Ob sich diese Verfahren auch in anderen Bereichen anwenden lassen, erscheint teilweise zweifelhaft.





2.5 Kernpunkte

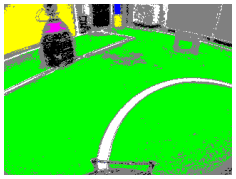
Viele der oben beschriebenen Herausforderungen illustrieren deutlich, daß die im RoboCup auftretenden Schwierigkeiten den Roboterfußball zu einem sehr interessanten Umfeld für die Erforschung realer Roboteranwendungen machen. Hohe Dy-

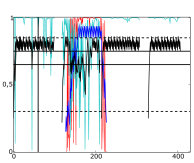


namik, unerwartete, teilweise widersprüchliche Sensorinformationen und technische Limitierung der Informationserfassung sind so oder so ähnlich auch in realweltlichen Einsatzgebieten zu erwarten. Zusätzlich müssen alle diese Probleme in einer robusten und flexiblen Art und Weise gelöst werden, um das verwendete System möglichst einfach an veränderte Umgebungsbedingungen anpassen zu können. Gleichzeitig ist klar geworden, daß sich das beschriebene Szenario deutlich von den sonst üblichen, artifiziellen Umgebungen anderer Roboterdemonstratoren unterscheidet [115, 117, 91]. Viele der beschriebenen Probleme sind in einem anderen Umfeld wie etwa einem weitgehend statischen Büroszenario nicht oder nur in geringerer Art zu beobachten. Insbesondere die hohe Dynamik sorgt für eine Reihe neuer Einflüsse, welche bei der Entwicklung eines Objekterkennungssystem berücksichtigt werden müssen.

Wie diese Probleme adressiert wurden, wird im weiteren Verlauf dieser Arbeit aufgezeigt. Daß das verwendete Verfahren leicht verändert werden kann, ist zum Beispiel dadurch sichergestellt, daß das Gesamtsystem strukturell und algorithmisch in mehrere Stufen aufgeteilt ist. Dies erlaubt es, einzelne Stufen beliebig gegen neue, geeignetere Module auszutauschen. Die mehrschichtige Architektur wird in Kapitel 3 diskutiert. Zusätzlich beruht die ganze geschriebene Software auf einer sehr mächtigen und flexiblen Softwarebasis, wie sie in Kapitel 4 beschrieben wird. Bereits auf dieser Ebene ist ein flexibles und modulares Konzept in Zusammenarbeit mit meinem Kollegen Hans Utz entstanden.







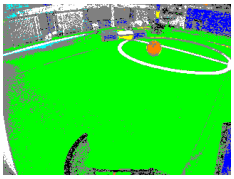
3 Lösungsansatz

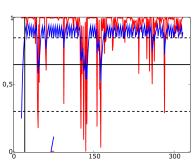
Nachdem im vorherigen Kapitel die zu berücksichtigenden Randbedingungen diskutiert wurden, wird in diesem Kapitel nun der deswegen gewählte Lösungsansatz beschrieben. Im folgenden wird nun erst einmal das gesamte Verfahren in einer kurzen Übersicht skizziert.

Als Basis dient ein Verfahren für die neuronale Klassifikation extrahierter Bildmerkmale. Diese Vorgehensweise hat sich bereits in anderen Robotikanwendungen bewährt [79, 80]. Anders als im hier untersuchten Szenario wurden diese Anwendungen aber für weit weniger dynamische Umgebungen entwickelt. Zusätzliche Schwierigkeiten, wie sie im vorherigen Kapitel beschrieben wurden, hohe Variabilität der Objektgrößen, Bewegungsunschärfe oder andere Kameralimitierungen, waren dort nicht Teil der Problemstellung. Aber auch wenn die verwendeten Verfahren wie in dieser Arbeit sorgfältig auf ihre Robustheit gegenüber allen Randbedingungen optimiert werden, ist das Verfahren für hochdynamische Umgebungen noch nicht hinreichend geeignet. Aus diesem Grund wurde der gewählte Ansatz mit einem Modul zur zeitlichen Integration erweitert. Dadurch entsteht eine geschlossene, rekursive Bearbeitungsschleife. Die Vorteile dieser direkten Eingliederung der zeitlichen Integration in das Klassifikationsverfahren sind vielfältig. Die erzielte Sicherheit über die beobachteten Objekte wächst natürlich über die Zeit. Im Gegensatz zur getrennten Auswertung in zwei Schritten lassen sich so aber situationsabhängig viele verschiedene Informationsquellen zur zeitlichen Integration heranziehen. Über nur einen einfachen Parameter läßt sich steuern, wie schnell die Integration den aktuellen Werten folgt oder wie lange Zeiträume durch die Integration überbrückt werden können. Gleichzeitig kann auch die Auswertungstiefe des Ablaufschemas durch die bereits integrierten (Un-)Sicherheitswerte gesteuert werden. Damit ist eine sehr einfache Möglichkeit gegeben, den Ressourcenverbrauch und die benötigten oder gewünschten Konfidenzwerte gegeneinander auszubalancieren und an das jeweilige Einsatzgebiet anzupassen. Zusätzlich verringert sich die durchschnittlich benötigte Zeit zwischen Bildaufzeichnung und Erhalt der Endergebnisse. Diese Latenzreduktion ist ein weiterer wichtiger Vorteil des Gesamtsystems für die Verwendung in hochdynamischen Umgebungen.

Im folgenden wird zuerst das verwendete Objekterkennungssystem eingeführt. Dessen einzelnen Module, die Suche nach Objekthypothesen, die Extraktion von Merkmalen aus diesen gefundenen Bereichen sowie die eigentliche Klassifikation werden ausführlich beschrieben. Anschließend folgt das Verfahren zur zeitlichen Integration. Dessen Konzepte sind Thema der Unterkapitel in Abschnitt 3.2. Allen diesen Teilbereichen sind dann im Anschluß weitere eigene Kapitel gewidmet. Abschließend wird das vorgestellte Konzept im Kontext anderer existierender Arbeiten in vergleichbaren Einsatzgebieten diskutiert.







3.1 Objektklassifikation

Die grundlegende Vorgehensweise für das Objekterkennungssystem, wie es in dieser Arbeit verwendet wird, läßt sich grob in die folgenden Einzelschritte unterteilen:

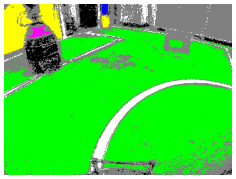
> Mehrstufige Klassifikation
> 1 Finden von potentiellen Objektpositionen
> 2 Extraktion der Merkmale aus diesen Regionen
> 3 Klassifikation der Merkmale mit Hilfe neuronaler Netzwerke

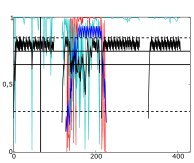
Im ersten Schritt (1) werden mögliche Objektpositionen im aufgezeichneten Bild gesucht, um die Aufmerksamkeit des Roboters auf die Objekthypothesen zu lenken. Alle folgenden Bildoperatoren werden nur noch auf diesen Regionen (sogenannten *regions of interest*, rechteckigen Bildausschnitten, kurz ROIs) angewendet. Das spart einerseits Rechenzeit, da in den folgenden Schritten nicht mehr das gesamte Bild ausgewertet werden muß, und läßt andererseits Bildbereiche außer acht, welche keine sinnvollen Informationen liefern würden. Da aber alle nachfolgenden Schritte *nur* noch auf diesen Regionen arbeiten, darf diese Regionensuche nicht zu restriktiv ausfallen, da keine Chance mehr besteht, Objekte außerhalb dieser gewählten Regionen zu finden. In der Liste der gefundenen ROIs werden daher zwangsläufig auch falsch positive Bereiche enthalten sein.

Im nächsten Schritt (2) werden verschiedene Merkmale aus diesen gefundenen ROIs extrahiert. Diese Merkmale beschreiben verschiedene Attribute der Objekte so allgemein wie möglich, um verschiedene Ausprägungen eines Objekts zu beschreiben. Andererseits sollten die Merkmale aber auch so selektiv wie möglich sein, um die gewünschten Objekte möglichst eindeutig von anderen, hier nicht relevanten Objekten zu trennen. Das Finden von geeigneten Merkmalen, die beide Voraussetzungen möglichst gut erfüllen, ist eine sehr schwierige Aufgabe, insbesondere unter den in Abschnitt 1.2 beschriebenen Randbedingungen.

Im letzten Schritt (3) werden alle berechneten Merkmale an ein künstliches neuronales Netz übergeben. In diesem Netzwerk wird die eigentliche Objektklassifikation berechnet, indem das Netz für jeden Merkmalsvektor entscheidet, ob er zur gesuchten Objektklasse gehört oder nicht. Das gesamte Verfahren ist im Detail in [78] beschrieben.







3.1.1 Finden von Regionen

Im ersten Bearbeitungsschritt soll die Aufmerksamkeit des Erkennungssystems auf Regionen gelenkt werden, welche mit hoher Wahrscheinlichkeit das gesuchte Objekt beinhalten. Das ist notwendig, da erstens die folgenden Arbeitsschritte möglicherweise sehr berechnungsaufwendig sind, und zweitens große Teile des Bildes sowieso keine relevanten Informationen enthalten. Andererseits werden eben auch nur diese Regionen weiterverarbeitet. Deshalb sollte ein solches Modul immer gleichzeitig möglichst

alle ROIs finden, um die eigentliche Objekterkennung zu garantieren und dabei möglichst wenig falsche ROIs zu identifizieren, und um den Gesamtberechnungsaufwand minimal zu halten.

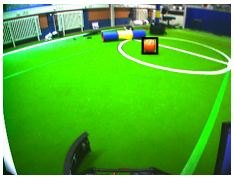
Angenommen, E sei ein Eingangsbild der Dimension $x \times y$ (im konkreten Fall $x = 640$, $y = 480$), dann ist eine solche Region definiert als Quadrupel $r = (a, b, c, d)$ mit $0 \leq a \leq x$, $0 \leq b \leq y$ als die Position der Region innerhalb des Bildes und $0 \leq c \leq x - b$, $0 \leq d \leq y - b$ als die Höhe und Breite der Region. Da mehrere solcher Regionen pro Bild existieren können, wird die Gesamtheit aller Regionen r_x mit $R = (r_1, r_2, \ldots, r_n)$ beschrieben. r^o beschreibt dann eine Hypothese r, die mit hoher Wahrscheinlichkeit ein Objekt des Typs $o = \{Ball, Tor, Roboter, \ldots\}$ repräsentiert. Eine solche Hypothese r^o ist dann korrekt positiv, wenn die Region tatsächlich ein Objekt des Typs o enthält. Die Hypothese ist falsch positiv, wenn sie den Objekttyp nicht enthält. Umgekehrt ist eine fehlende Hypothese $\tilde{r}$ falsch negativ, wenn der Bereich $\tilde{r}^o$ des existierenden Objekts nicht in R enthalten ist: $R \cap \tilde{r}^o = \{\}$.

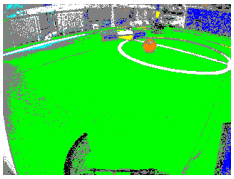
Die Suche nach Regionen wird immer auf dem gesamten Bild ausgeführt. Aus diesem Grund müssen die hier verwendeten Verfahren und Merkmale sehr einfach zu berechnen sein, worunter aber wiederum die Spezifität leidet. Um beide erwähnten Voraussetzungen zu erfüllen – möglichst alle Regionen finden, möglichst keine falschen Regionen finden – ist die Regionensuche meist stark modellbasiert und abhängig von vielen Voraussetzungen, die sich aus dem jeweiligen Einsatzgebiet ergeben. Um diese Voraussetzungen und Heuristiken in der Implementation noch schnell und wartbar zu halten, existiert ein eigenes Suchmodul für jeden gesuchten Objekttyp.

Im RoboCup sind alle relevanten Objekte auf dem Spielfeld durch eindeutige Farben gekennzeichnet. Der Ball ist orange, das Spielfeld grün und die Tore blau und gelb. Eine einfache und schnelle Regionensuche verwendet daher fast zwangsläufig diese Farben. Folgerichtig basieren die hier benutzten Aufmerksamkeitsmodule im Prinzip auf einer Farbbereichssuche in einem farbindizierten Bild. Zusätzliche Modellparameter reduzieren die Liste der gefundenen ROIs dann weiter. Dieses Verfahren ist in Kapitel 5 ausführlich beschrieben.

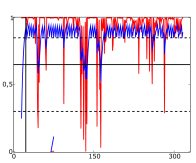
3.1.2 Merkmalsberechnung

Im zweiten Schritt werden verschiedene Merkmale aus den Objekthypothesen berechnet. Da auch Objekte des gleichen Typs sich unter Umständen erheblich voneinander unterscheiden oder verschiedene Ansichten des gleichen Gegenstandes aus verschiedenen Blickwinkeln sehr unterschiedlich sein können, müssen die verwendeten Merkmale dieser Tatsache gerecht werden. Alle verwendeten Merkmale müssen daher flexibel genug sein, diese Unterschiede zu tolerieren, andererseits aber aussagekräftig genug, um die gesuchten Objekte von anderen Objekte zu unterscheiden.





Merkmale in diesem Sinne sind zum Beispiel sehr einfach zu berechnende Attribute wie räumliche bzw. geometrische Eigenschaften wie die Breite oder Höhe eines Objekts bzw. der Region. Es können aber auch sehr aufwendig zu berechnende Merkmale wie Histogrammrepräsentationen der Farbverteilung in Bereichsunterregionen sein. Im Prinzip lassen sich dafür alle Attribute verwenden, die sich in irgendeiner Form



in einer oder mehreren Kennzahlen ausdrücken lassen. Berechnete Merkmale sind damit beliebige mehrdimensionale Beschreibungen $m \in \mathbb{R}^n$, die gemäß einer Abbildung $E \mapsto m$ aus dem Eingangsbild E berechnet werden können. Da normalerweise mehrere Merkmalsvektoren pro Region berechnet werden, ist $M = (m_1, m_2, \ldots, m_p)$ die Gesamtmenge aller Merkmale m_x, $\forall x \in \mathbb{N}^p$.

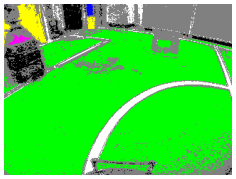
Da aber viele der existierenden Objekte im RoboCup sehr artifiziell sind, ist die Auswahl der richtigen Merkmale extrem schwierig. Ein Beispiel sind die Torpfosten, die für die Selbstlokalisation benötigt werden. Solch ein Torpfosten ist kaum mehr als eine schmale vertikale Linie, die auf der einen Seite eine homogene Fläche in der Torfarbe hat und auf der anderen Seite viel unvorhersehbaren Hintergrunds. Stattdessen einfach nach dem ganzen Tor zu suchen ist aber nicht möglich, da dieses nur sehr selten komplett im Bild zu sehen ist.

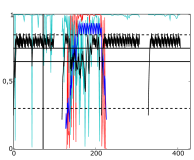
Oft sind vor der eigentlichen Merkmalsextraktion noch einige Vorverarbeitungsschritte notwendig, um die Qualität der Attribute zu verbessern oder den Suchbereich noch weiter einzuschränken. Da die ROIs hier immer rechteckige Bildbereiche sind, die darin enthaltenen Objekte aber jede beliebige Form haben können, ist in der Region meist zusätzlich noch viel Hintergrund zu sehen. Um dieses Rauschen zu reduzieren, werden binäre Masken auf dem Bildausschnitt erstellt. Merkmale werden dann nur an den Stellen berechnet, welche einen positiven Wert in der Binärmaske haben. Andere Vorverarbeitungsschritte sind wegen stark unterschiedlicher Größen der Regionen notwendig. Die Nahansicht des Balls enthält wesentlich mehr Texturinformation als ein Ball aus 5 Meter Entfernung. Eine Größenangleichung der Suchregionen kann diesen Unterschied deutlich verringern. Welche Attribute letztendlich verwendet werden und welche Vorverarbeitungsschritte nötig sind, werden im Detail in den Abschnitten 6.1 und 6.2 erklärt.

3.1.3 Neuronale Klassifikation

Nachdem die gewünschten Merkmale aus den Regionen extrahiert wurden, werden diese an ein oder mehrere neuronale Netze weitergereicht. Da für jeden Objekttyp ein separates Aufmerksamkeitsmodul existiert und folglich nur noch eine binäre Entscheidung nötig ist, ob das Objekt vom gesuchten Typ ist oder nicht, werden hier nur einzelne Netze und keine hierarchischen Netzwerke verwendet. Dadurch sinkt der nötige Berechnungsaufwand für die Klassifikation deutlich. Aus demselben Grund besitzen die meisten Netzwerke auch nur ein Ausgabeneuron. Die Anzahl der Eingabeneuronen ist außerdem durch die Länge des Merkmalsvektors vorgegeben. Um auch hier den Berechnungsaufwand möglichst gering zu halten, wird für alle Netze die minimal nötige Anzahl von Neuronen in den inneren Schichten gesucht. Das garantiert nicht nur eine geringe Klassifikationsdauer, sondern kann unter Umständen auch die Generalisierungsleistung des Netzes positiv beeinflussen. Das neuronale Netz kann als eine weitere Abbildung $n = [0, 1]$ aufgefasst werden, in der ein oder mehrere Merkmalsvektoren auf einen Wahrscheinlichkeitswert umgerechnet werden $m_1, m_2, \ldots, m_n \mapsto n$.







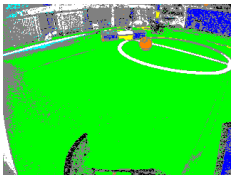
Falls mehrere, getrennte neuronale Netze für verschiedene Teile des Merkmalsvektors verwendet werden, muß die abschließende Entscheidungsfindung über ein getrenntes Arbitrierungsmodul geschehen. Da hier jedem Netzwerk nur eine Teilmenge der Objektattribute zur Verfügung stehen, müssen alle beteiligten Netze einen möglichst hohen Einzelwert liefern, um ein positives Gesamtergebnis zu erhalten. Aus diesem Grund werden hohe Schwellenwerte für alle Teilergebnisse festgelegt. Falls eines der Netzwerke einen Wert unterhalb dieser Schwellen zurückgibt, wird die Region als falsch zurückgewiesen. Da nur sehr wenige, dafür aber sehr selektive Merkmale pro Objekttyp verwendet werden, sind alle Merkmale mit Sicherheit auch sehr wichtig für die Erkennung. Aus diesem Grund können, wie beschrieben, die Schwellenwerte auch fest auf einem sehr hohen Schwellenniveau angesetzt werden, da durch die hohe Selektivität der Merkmale jeder schlechte Wert eines neuronalen Netzes mit Sicherheit die Existenz des gesuchten Objekttyps ausschließt. Mit der Menge $N = (n_1, n_2, \ldots, n_p)$ aller berechneten Netzantworten ist dann das abschließende Arbitrierungsergebnis $N \mapsto e$ wiederum ein Wahrscheinlichkeitswert $e = [0, 1]$ für die eigentliche Hypothesenkorrektheit.

Obwohl für eine direkte Entscheidungsfindung nur eine binäre Entscheidung der Netzwerke bzw. des Arbitrierungsschemas verlangt ist, enthalten die tatsächlich ausgegebenen, kontinuierlichen Zwischenwerte wichtige Informationen über die Sicherheit, mit der die neuronalen Netze die Entscheidung getroffen haben. Im weiteren werden daher nicht absolute Binärwerte, sondern direkt die Netzausgabe bzw. das Mittel der Entscheidungen, falls mehrere Netzwerke benutzt werden und alle Ausgaben über dem gewünschten Schwellenwert liegen, verwendet. Wie diese Informationen weiter genutzt werden, ist im nächsten Abschnitt beschrieben.

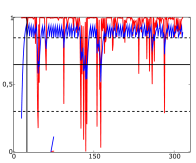
Alle neuronalen Netzwerke, die in dieser Arbeit verwendet werden, werden mit einem überwachten Lernverfahren trainiert. Aus diesem Grund muß eine ausreichende Menge an Trainingsmaterial vorbereitet werden. Große Teile davon lassen sich halbautomatisch durch die existierenden Aufmerksamkeitsmodule erstellen. Eine gründliche, manuelle Inspektion dieser Ergebnisse ist aber trotz allem noch nötig. Neben ausreichend positiven sind auch gute negative Trainingsbeispiele wichtig. Dazu werden ähnlich aussehende Objekte gesucht (hauptsächlich automatisch gefundene Objekte, die anschließend aussortiert wurden) und außerdem eine große Anzahl von zufällig ausgewählten Bildbereichen gesammelt (mehr dazu in Abschnitt 6.3).

3.2 Zeitliche Integration der Klassifikationsergebnisse





Wie bereits angesprochen, ist auch ein sorgfältig abgestimmter Klassifikator noch nicht optimal für den Einsatz in hochdynamischen Umgebungen geeignet. Erstens lassen sich gelegentliche, falsche Klassifikationsergebnisse durch das völlige Ignorieren jeglicher zeitlichen Information nicht erkennen oder vermeiden. Zweitens steht die benötigte Rechenleistung im Kontrast zu einer gewünschten, möglichst hohen Anzahl von ausgewerteten Bildern. Wird es möglich, diese benötigte Rechenleistung zu reduzieren, sinkt gleichzeitig die Latenz zwischen Bildaufzeichnung und dem Erreichen



des Ergebnisses. Die dadurch gewonnenen Ergebnisse sind aktueller und verbessern damit die Ergebnisse abgeleiteter Module, was sich positiv auf das Gesamtsystem des Roboters auswirkt.

Die folgende Erklärung ist in zwei Teile aufgespalten, um die Verbesserung für die oben beschriebenen zwei Hauptbereiche besser illustrieren zu können.

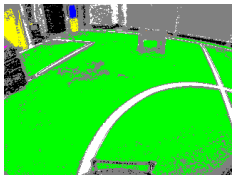
3.2.1 Klassifikationsverbesserung

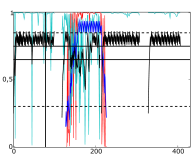
Der oben skizzierte Algorithmus wird mit einem Kalman-basierten Objektverfolgungs- und -prädiktionsmodul [97] erweitert. Dieses Modul wird zwischen der Aufmerksamkeitssteuerung und dem nachfolgenden Objektklassifikationsteil eingefügt. Der erste Vorteil, den man dadurch ausnutzt, ist eine verbesserte und stabilisierte Objektklassifikation. Um von diesen zusätzlichen zeitlichen Informationen zu profitieren, wird der bereits bekannte Ablauf folgendermaßen erweitert.

KLASSIFIKATIONSVERBESSERUNG
1 Finden von potentiellen Objektpositionen
2 Vorhersage der aktuellen Position von bisher gefundenen Objekten
3 Zuordnung von neuen zu alten Tracks
4 Extraktion der Merkmale aus diesen Regionen
5 Klassifikation der Merkmale mit Hilfe neuronaler Netzwerke
6 Integration der bisherigen Klassifikationsergebnisse
7 Starten/Beenden von neuen/alten Tracks
8 Korrektur vorhergesagter Positionen anhand aktueller Beobachtungen

Der erste Schritt (1) wird weiter wie bisher abgearbeitet. Im nächsten Schritt (2) werden alle bisher bekannten Objekttrajektorien T_{t-1} auf ihre wahrscheinlichste Position T_t zum aktuellen Zeitpunkt vorhergesagt. Eine Objekttrajektorie t (auch kurz Track genannt) ist eine Trajektorie aus der gesamten Trackmenge $T = (t_1, t_2, \ldots, t_n)$ mit der aktuellen egozentrischen Position (z_x, γ_x) mit Distanz z und Winkel γ zum Roboter. Dann (3) wird versucht, diese prädizierten Objekthypothesen den neuen, gefundenen ROIs zuzuordnen. Das dabei verwendete Verfahren garantiert, daß jede Objekthypothese nur einer neuen Objektposition zugeordnet werden kann und umgekehrt jede Objektposition mit maximal einer Hypothese assoziiert wird. Die beiden nächsten Stufen (4-5) werden wiederum wie oben beschrieben ausgeführt. Die aktuelle Entscheidung des neuronalen Klassifikators e wird nun mit dem bisherigen (Un-)Sicherheitsniveau $(s_{t-1}, e) \mapsto s_t$ verrechnet. So erhält man über die Zeit eine wesentlich bessere Einschätzung, wie sicher das Objekt als solches bestätigt oder falsifiziert wurde. Falls in Schritt (3) kein Track gefunden wurden, der mit hoher Wahrscheinlichkeit der bisherigen Bewegung einer gefundenen Region entspricht $(T_t \cap R = \{\})$, werden für diese ROIs nun neue Tracks initiiert. Umgekehrt enden Tracks, wenn sie über einen längeren Zeitraum keiner neuen Region zugeordnet werden konnten (7). Abschließend wird die prädizierte Objektposition mit der tatsächlich







beobachteten und zugeordneten Position korrigiert (8), um auch im nächsten Schritt wieder eine verlässliche Vorhersage treffen zu können ($(t_{t-1}, r) \mapsto t_t$). Damit schließt sich der Kreis zu einer vollständig rückgekoppelten Bearbeitungsschleife. Details zur Realisierung des Kalman-Filters und der Track-Regionen-Zuordnung folgen in den Abschnitten 7.2 und 7.3.

3.2.2 Reduktion des Ressourcenverbrauchs

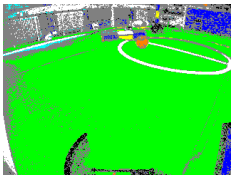
Der zweite wichtige Vorteil, den eine solche Integration der Positionsvorhersage in die Klassifikationsarchitektur bringt, ist es, daß man dadurch einen erheblichen Teil des Rechenaufwands einsparen kann. Die Idee dahinter ist, daß man Teile der Bearbeitungskette einsparen kann, wenn eine neu gefundene Region einem Track entspricht, bei dem man sich bereits sehr sicher ist, das heißt daß er einem Objekt gehört, dessen tatsächliche Existenz man bereits mehrfach bestätigt oder negiert hat (was wiederum bedeutet, daß es einen Sicherheitswert s nahe 1,0 bzw. 0,0 besitzt). Der nochmals erweiterte Ablauf sieht nun wie folgt aus:

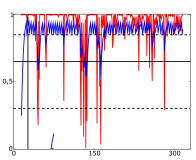
Reduktion des Ressourcenverbrauchs

1 Finden von potentiellen Objektpositionen
2 Vorhersage der aktuellen Position von bisher gefundenen Objekten
3 Zuordnung von neuen zu alten Tracks
4 **if** Sicherheitswerte für Tracks liegen zwischen zwei Schwellenwerten:
5 **then** Extraktion der Merkmale aus diesen Regionen
6 Klassifikation der Merkmale mit Hilfe neuronaler Netzwerke
7 Integration der bisherigen Klassifikationsergebnisse
8 **else** Reduziere Sicherheitswerte
9 Starten/Beenden von neuen/alten Tracks
10 Korrektur vorhergesagter Positionen anhand aktueller Beobachtungen

Hierzu wird nun der Ablauf in zwei Bearbeitungsstränge aufgespalten. Wenn die Sicherheitswerte der zugeordneten Tracks noch nicht einer oberen bzw. unteren Schranke genügen ($\theta_1 \leq s \leq \theta_2$), das heißt die (Nicht-)Existenz eines Objekt noch nicht ausreichend bestätigt wurde, werden die Klassifikations- und Integrationsschritte (5-7) wie oben beschrieben ausgeführt. Dasselbe gilt, falls der Region kein Track zugeordnet wurde, es sich also um ein neues, noch nicht bestätigtes Objekt handelt. Alle Regionen, deren Sicherheitswert bereits über oder unter dem gewünschten Sicherheitsniveau liegen ($s < \theta_1 \cup s > \theta_2$), die einen Sicherheitswert nahe 0,0 oder 1,0 besitzen, werden *nicht* ausgewertet. Der gesamte Aufwand der Merkmalsberechnung und Klassifikation entfällt damit. Damit ein einmal bestätigtes Objekt nicht für immer über der gewählten Schwelle verbleibt, wird der Sicherheitswert mit einem Wert aktualisiert, welcher zwischen den zwei Schrankenwerte liegt ($s_{t-1}, e_\theta \mapsto s_t$ mit $\theta_1 < e_\theta < \theta_2$). Dadurch wird eine Art „Vergessen“ modelliert, so daß Objekte nach einer gewissen Zeit automatisch wieder neu klassifiziert werden müssen. In Abbildung 3.1 ist der gesamte Ablauf zum besseren Verständnis noch einmal graphisch illustriert.







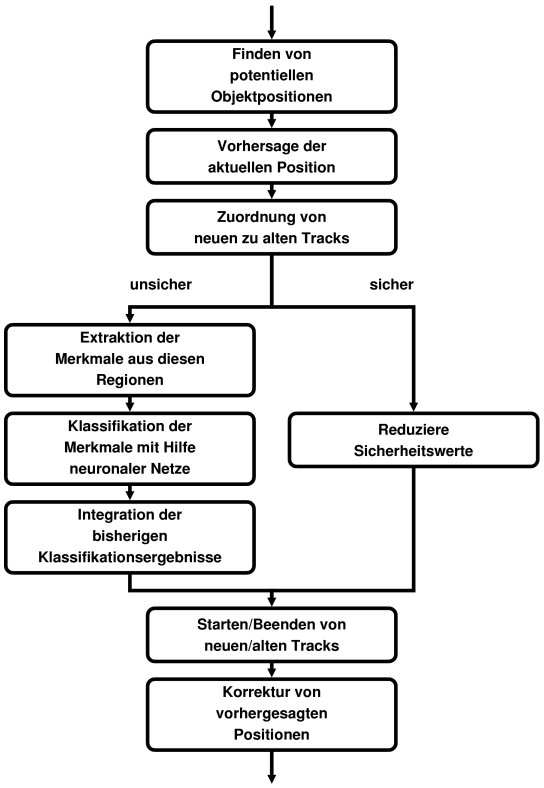
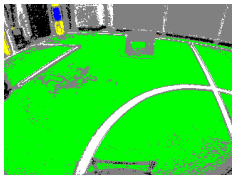


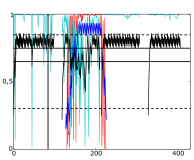
Abbildung 3.1: Illustration des gesamten Bearbeitungsschemas. Zu sehen sind alle Module der Regionensuche, zur zeitlichen Integration und für die neuronale Klassifikation.

3.2.3 Informationsintegration

Die enge Kopplung der Objektklassifikation mit der Objektprädiktion erlaubt eine sehr detaillierte Integration der (Un-)Sicherheitswerte innerhalb dieser Bearbeitungsschleife. Wenn der Klassifikator ein Objekt als solches erkennt oder zurückweist, wird dieser erhaltene Wert direkt zur zeitlichen Integration verwendet. Wird der Klassifikationsschritt nicht ausgeführt, wird ein a-priori-Wert verwendet, welcher die durchschnittliche Korrektheit des Aufmerksamkeitsmoduls repräsentiert. Solche Werte lassen sich leicht aus aufgezeichneten Spieldaten ermitteln.







Was oben bisher nicht berücksichtigt wurde, sind Tracks, denen in einem Durchgang keine Region zugeordnet worden ist. Falls die vorhergesagte Objektposition außerhalb des Sichtbereichs des Roboters liegt, eine Erkennung als gar nicht möglich ist, wird ein vorher bestimmter Wert für die Wahrscheinlichkeit, daß sich ein Objekt auf dem Spielfeld befindet, integriert. Wenn sich eine Objekthypothese aber im Sichtbereich des Roboters befindet, die Regionensuche aber keine ROI dafür gefunden hat, läßt sich die Entscheidung sogar noch weiter auftrennen. Im hier beschriebenen Szenario ist es sehr einfach, ein kleines Modul zu schreiben, das entscheidet, ob man an der vorausberechneten Position wahrscheinlicher das Spielfeld oder einen anderen Ge-

genstand sieht – ob man das Objekt sehen müsste oder es sich hinter einem anderen Gegenstand verbirgt. Der zu integrierende Wert ist dann wieder eine vorher bestimmte Wahrscheinlichkeit, daß ein Objekt an dieser Stelle ist oder die Wahrscheinlichkeit, daß sich die Regionensuche geirrt und das gesuchte Objekt übersehen wurde. Zusätzlich werden die Wahrscheinlichkeitswerte gewichtet, je nachdem, wie exakt der Track der Objektverfolgung ist. Die verwendete Wahrscheinlichkeit reduziert sich, je länger ein Track nicht mehr über neue Positionen aktualisiert wurde. Details zur zeitlichen Integration finden sich in Abschnitt 8.1.

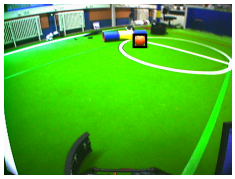
In manchen Einsatzgebieten mag eine niedrige obere Sicherheitsschwelle nicht ausreichend sein; die zeitliche Einsparung für tatsächliche Objekte ist dann gering. Für fast alle Szenarien wird es aber gleichgültig sein, ob eine ablehnende Klassifikation mit einer Sicherheit von 0,0 oder 0,3 erfolgt, das heißt besonders bei fortlaufenden Falsifikationen durch den Klassifikator ist ein hohe Zeiteinsparung möglich.

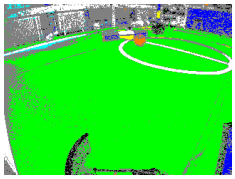
3.3 Kernpunkte

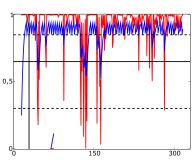
Das beschriebene mehrstufige Basisverfahren aus Regionensuche, Merkmalsextraktion und neuronaler Klassifikation beweist seine Flexibilität bereits dadurch, daß es auch von anderen in vielen verschiedenen Szenarien realisiert wurde [79, 116, 117]. Die meisten Anwendungen auf mobilen Robotern sind aber im statischeren und kontrollierteren Bürobereich angesiedelt, in dem viele der beschriebenen Probleme wie Bewegungsunschärfe, stark verrauschte Odometriebewegungsinformation und das Auftreten von unbekannten Objekten und Objekttypen nicht berücksichtigt werden müssen. Die Anwendung einer solchen Architektur im RoboCup ist einzigartig. Zwar wird auch in der Forschung zu autonomen Automobilen verstärkt mit neuronalen Klassifikatoren gearbeitet, aber hier sind meist völlig andere Voraussetzungen gegeben. Viele Arbeiten konzentrieren sich nur auf bewegte Objekte, da nur diese auf den zu überwachenden Bereich Einfluß nehmen können, wodurch sie Informationen wie beispielsweise den optischen Fluß nutzen können [59, 126]. Andere Arbeiten setzen eine deutlich andere sensorische Ausstattung wie etwa Stereokameras voraus [32, 44].

Die konkrete Implementation wiederum zeigt ihre Flexibilität dadurch, daß sie auch zu großen Teilen in anderen Bereichen wie dem EU-Projekt *MirrorBot – Biomimetic multimodal learning in a mirror neuron-based robot*[1] eingesetzt wird [29, 30, 31]. Zwar werden dort hierarchische neuronale Klassifikatoren anstatt der hier benutzten flachen, binären Klassifikatoren verwendet, die entwickelte Basisarchitektur *VIP* sowie die zur Merkmalsberechnung verwendeten Filter werden aber auch dort verwendet (mehr dazu im nächsten Kapitel).

Neben der Anwendung einer solchen Architektur innerhalb einer Umgebung wie dem RoboCup [78, 92] ist die enge Verknüpfung der Klassifikation mit einer zeitlichen Integration eine der wesentlichen Neuerungen des hier vorgestellten Verfahrens [93, 96]. Diese Integration stabilisiert und beschleunigt das Gesamtverfahren gleicher-



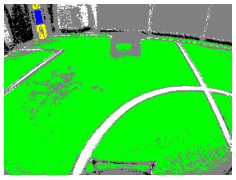


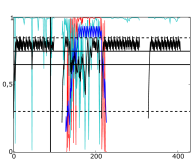


[1] Ein Projekt der Universitäten Sunderland, MCR Cambridge, Universität Parma, INRIA Nancy und der Universität Ulm. EU-IST-FET Projekt des EU Neuro-IT Network of Excellence.

maßen. Es erlaubt eine einfache Abwägung zwischen der gewünschten Konfidenz für jeden einzelnen Objekttyp gegenüber dem Rechenaufwand, der nötig ist, um diese Information aus den Eingangsbildern zu extrahieren. Gleichzeitig ermöglicht diese direkte Kopplung von Klassifikation und temporaler Integration das Einbeziehen einer Vielzahl unterschiedlicher Informationen, wie sie in Kapitel 8 beschrieben wird. Wären diese Komponenten zwei getrennte, nacheinander ausgeführte Module, so würden sich auf der einen Seite nur die Endresultate der Klassifikation zur zeitlichen Integration berücksichtigen lassen. Auf der anderen Seite kann man nur so bereits während der Erkennungsausführung durch die integrierte Information Einfluß auf die Bearbeitungskette nehmen. So können Teile der sonst nötigen Operatoren zugunsten einer deutlich höheren Bearbeitungsgeschwindigkeit bei Bedarf übersprungen oder alternative Auswertungspfade einschlagen werden.







4 Implementationsrahmen

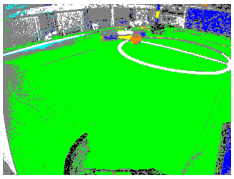
Die verwendete Softwarebasis eines Projekts wird nicht nur dessen Implementation maßgeblich beeinflussen. Ein durchdachtes Konzept kann Entwurf, Entwicklung und Portierung wesentlich vereinfachen und verbessern. Um das im vorherigen Kapitel beschriebene Vorgehen der strukturierten und modularen Entwicklung auch auf Softwareebene fortzusetzen, wurde in Zusammenarbeit mit meinem Kollegen Hans Utz eine Infrastruktur für die Bildverarbeitung auf autonomen, mobilen Robotern entwickelt. Entwicklungsziele waren die Entwicklerunterstützung bei der Entstehung und während der Testphase unter besonderer Berücksichtigung der Modularisierungsmöglichkeiten sowie der Echtzeitfähigkeit.

Grundidee einer solchen Infrastruktur ist eine möglichst weitgehende Entkoppelung der eigentlichen Bildverarbeitung von speziellen Anforderungen des Einsatzgebietes. Zu solchen Anforderungen gehören nicht nur die angesprochenen Echtzeitfähigkeiten, sondern auch Schwierigkeiten, die sich aus dem Einsatz auf autonomen, mobilen Robotern ergeben: Ansprechen der Kameras, Synchronisation der Bilder mehrerer Kameras, parallele Abarbeitung der Bearbeitungsstränge dieser Kameras und vieles mehr.

Ein weiterer wichtiger Grundgedanke bei der Entwicklung dieser Arbeit war die Verwendung bestehender, bewährter und flexibler Komponenten. Solch ein Ansatz garantiert die leichtere Portierung und Adaption des Gesamtsystems an geänderte Umgebungsbedingungen oder Einsatzgebiete. Aus diesem Grund basieren Teile des entwickelten Frameworks für die Bildverarbeitung auf einer Middleware namens Miro. Obwohl Miro nicht Teil dieser Arbeit ist, wird es im Folgenden kurz vorgestellt, um die daraus verwendeten Komponenten zu beschreiben. Weitere Bibliotheken, die ebenfalls diesem Gedanken der Verwendung bestehender Infrastrukturen geschuldet sind, werden ebenfalls später in diesem Kapitel beschrieben. Der für diese Arbeit interessanteste Teil von Miro, das *video image processing framework (VIP)*, die Basissoftware für die Bildverarbeitung, ist im zweiten Teil beschrieben.

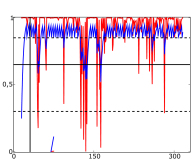
4.1 Miro





Middlewaresysteme[1] werden hauptsächlich verwendet, um den Aufwand für die Entwicklung und Portierung einer Software zu reduzieren. Autonome, mobile Roboter laufen unter verschiedenen Betriebssystemen, werden in vielen verschiedenen Programmiersprachen programmiert und können sich in der verwendeten Hardware stark

[1] eine passende deutsche Bezeichnung für viele der hier verwendeten englischen Begriffe existiert leider nicht; der Name Abstraktionsschicht trifft den Sachverhalt nur ungenügend.



unterscheiden. Sie sind außerdem inhärent verteile Systeme; einige Roboter haben bereits mehrere Rechner eingebaut, die Bedienungsoberfläche wird oft auf anderen Rechnern dargestellt und in multi-robot-Szenarien wird die Verteilung dann offensichtlich. Aus diesem Grund wurde MIRO speziell dafür entwickelt, die meisten dieser Anforderungen vor dem Entwickler zu verbergen und einfache, gut getestete Komponenten für die am häufigsten benötigten Aufgaben zur Verfügung zu stellen.

Das Problem der Verteiltheit löst MIRO durch die Verwendung von CORBA [48, 49, 107] als Standardprotokoll für die Kommunikation zwischen den Komponenten. Mit CORBA lassen sich Objektmethoden unabhängig von ihrem tatsächlichen physikalischen Ausführungsort aufrufen. Zusätzlich besitzt CORBA Anbindungen für die verschiedensten Programmiersprachen, was beispielsweise die Kombination eines Kontrollprogrammes in C++ mit einer Bedienoberfläche in Java erlaubt.

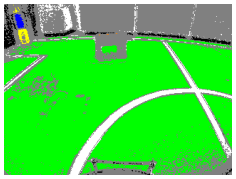
Hardwareaufrufe, um Motoren oder Sensoren anzusprechen, werden in entsprechenden Services mit generischen Interfaces gekapselt. Um den Roboter zu bewegen, ist es dann nicht nötig zu wissen, welche Antriebsart der Roboter besitzt. Ein Aufruf der entsprechenden Klasse mit dem Befehl „fahren“ genügt.

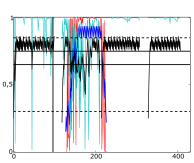
Um die Fehlersuche zu vereinfachen, läßt sich der komplette Zustand des Roboters während der Programmausführung in einer Datei mitschreiben. Dazu gehören alle Zustandsinformationen, die zwischen den Programmkomponenten ausgetauscht werden, also die Position des Roboters, die gesehenen Objekte oder sein Spielzustand. Später können diese Informationen dann sehr einfach wieder in das System eingespielt werden, ohne daß die Komponenten dafür geändert werden müssen [136].

Eine andere Hilfestellung für Programmentwickler existiert durch die Möglichkeit der einfachen Parametrisierung der Programme. Insbesondere Programme für autonome, mobile Roboter besitzen eine gewaltige Menge an Parametern in allen Komponenten, wie die maximal erreichbare Geschwindigkeit oder die Position der Sensoren. Alle diese Parameter können nun mit Hilfe eines XML-Dialekts spezifiziert werden. Ein in MIRO enthaltenes Programm übersetzt diese Beschreibungen dann in passende Softwareklassen, welche sich um das Einlesen der Parameter zu Programmbeginn kümmern.

Eine wesentlich ausführlichere Beschreibung der Möglichkeiten von MIRO findet der interessierte Leser in [85], [139] oder der Dissertationsschrift von Hans Utz [132]. Ein integraler Bestandteil von MIRO und technischer Unterbau für die hier vorliegende Arbeit, das *video image processing framework (VIP)*, wird im nachfolgenden Abschnitt vorgestellt.







4.2 Video Image Processing Framework

Aus Softwaresicht ist Bildverarbeitung mehr, als nur ständig neue Bilder von einer Kamera anzufordern. Bildoperatoren müssen miteinander verbunden werden, Bildbereiche definiert und Teilergebnisse durch das System propagiert werden. Besonders der Einsatz auf autonomen, mobilen Robotern bringt eine Reihe von neuen Problemen mit sich. Außer der oben beschriebenen Abstraktion der Hardware kommen

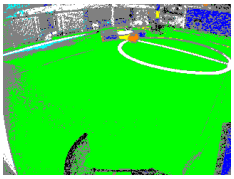
andere Schwierigkeiten dazu wie das parallele Ausführen von Bearbeitungssträngen mehrerer Kameras oder der Synchronisierung der Bilder dieser Kameras. Diese Bearbeitungsstränge können für ausgereifte Bildverarbeitungssysteme sehr aufwendig werden. Eine zusätzliche Hilfe in der Organisation des Bearbeitungs- und Datenflusses ist daher wünschenswert. Aus diesem Grund erfolgt im *VIP-Framework* die Entwicklung der Bildbearbeitungsoperatoren und die Spezifikation des Bearbeitungsablaufs in getrennten Schritten.

Die Basiskomponente im *VIP-Framework* ist ein Filter. Diese Filter sind in einer baumartigen Struktur organisiert, die dann automatisch vom Framework in der richtigen Reihenfolge abgearbeitet werden. Der Datenfluß durch diesen Baum kann wesentlich komplexer, in Form eines azyklischen, gerichteten Graphen geplant werden. Dazu kann jeder Filter ein oder mehrere Eingabebilder auf ein Ausgabebild abbilden. Auch die Organisation dieses Datenflusses wird anschließend vom *VIP-Framework* übernommen. Zusätzlich können sogenannte Meta-Daten zwischen den Filtern durchgereicht werden. Diese Datenstrukturen können mit Hilfe des oben beschriebenen Parametrisierungskonzepts komplett frei definiert werden und beinhalten dann Listen von gefundenen Regionen oder bestimmte Eigenschaften von Bildbereichen. Wie der gesamte Bearbeitungsbaum werden auch die für jeden Filter möglichen Parameter dann in einer XML-Datei definiert. So lassen sich sehr einfach verschiedene Konfigurationen oder Parametrisierungen während der Entwicklungsphase durchtesten.

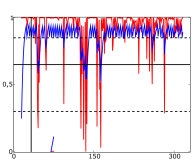
Damit hebt sich das Filterkonzept deutlich von der informationstheoretischen Filterdefinition ab. Ein Filter innerhalb des Frameworks ist definiert als jede beliebige Abbildung der Eingangsinformationen entweder auf ein Ausgabebild oder eine auszugebene Meta-Datenstruktur. Sei f eine beliebige Abbildungsfunktion, so ist nicht nur die Abbildung eines Eingangsbildes a auf ein Ausgabebild y möglich ($y = f(a)$), sondern ebenfalls eine Abbildung mehrerer Bilder $a_{x|x=\{1,2,\ldots\}}$ auf ein Ergebnis $y = f(a_1, a_2, \ldots)$. Zusätzlich kann noch mit jedem Ausgabebild a_x ein n-dimensionaler Ausgabevektor $t_x \in \mathbb{R}^n$ assoziiert sein. Damit ist ein Ausgabebild und ein damit wiederum assoziierter Ausgabevektor $(y, z) = f((a_1, t_1), (a_2, t_2), \ldots)$. Die Bandbreite der implementierten Filtern reicht von einfachen Bildverarbeitungsoperationen wie einer Gaußglättung oder einer Farbraumkonvertierung bis hin zu komplexen Merkmalshistogrammen, die zwei Eingabebilder auf eine Histogrammdatenstruktur abbilden.

Von besonderer Bedeutung im Rahmen dieser Arbeit ist auch die bedarfsabhängige Auswertung. Dies bedeutet, daß komplette Teilbäume nur dann ausgewertet werden, wenn eine andere Komponente des Gesamtsystems Ergebnisse dieses Teilbaumes anfordert. Damit lassen sich Zweige im Baum für jeden denkbaren Einsatzbereich definieren, ohne daß die Gesamtleistung des Systems dadurch beeinträchtigt wird. Sie werden nur dann berechnet, wenn der Roboter tatsächlich in der jeweiligen Situation die Resultate daraus benötigt. Über die oben angesprochene Ortstransparenz lassen sich diese Resultate übrigens auch sehr einfach im laufenden Betrieb von jedem beliebigen Rechner aus kontrollieren.





Ein interessantes Detail ist auch die Tatsache, daß sich die für den Roboter definierten Filterbäume einfach durch den Austausch des Wurzelknotens, welcher die



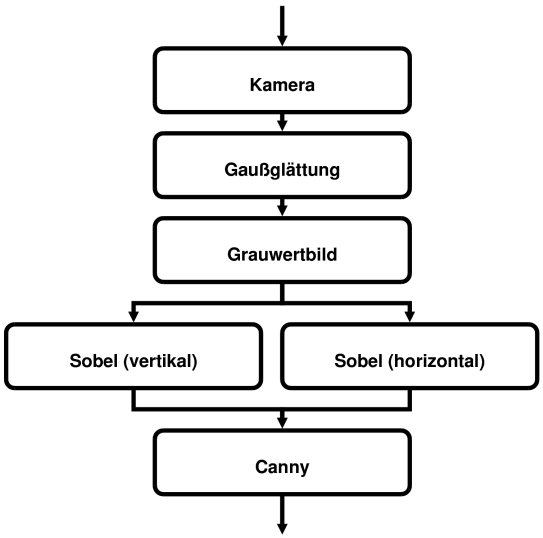


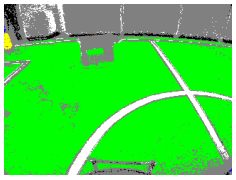
Abbildung 4.1: Beispiel für eine simple Filterkonfiguration zur Detektion von Kanten im Bild. Als Baumknoten ist ein spezieller Kamerafilter vorgesehen. Danach folgen eine Gaußglättung, eine Grauwertkonvertierung, mehrere Sobelfilter und ein abschließender Cannyfilter. Die Filter verwenden entweder ein oder mehrere Bilder als Eingabewerte.

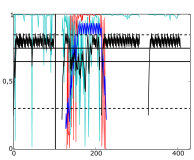
Bildquelle definiert, für die dateibasierte Entwicklungsarbeit benutzen lassen. Aus Geschwindigkeitsgründen werden Bilddaten nicht in der im vorigen Abschnitt beschriebenen Art mitprotokolliert, sondern in normale Bilddateien geschrieben. Diese lassen sich dann mit einem einfachen dateibasierten Baumknoten wieder in den Bearbeitungsbaum einspeisen.

Zusätzlich werden auch Mehrkamerasysteme unterstützt. Da sich Bilder und Meta-Daten auch zwischen den Filterbäumen zweier Kameraströme austauschen lassen, existieren verschiedene Mechanismen für die Synchronisation zwischen den Bäumen. Da verschiedene Kameras ihre Bilder unter Umständen mit verschiedenen Geschwindigkeiten liefern, unterstützt das *VIP-Framework* verschiedene Synchronisationsmechanismen, die speziell für die Stereobildverarbeitung oder das parallele Bearbeiten eines schnellen und eines langsamen Verarbeitungstrangs optimiert sind.

Ein kleines Beispiel für das oben Angesprochene ist in Abbildung 4.1 zu sehen. Ausgehend von der Wurzel des Baums, einem Kamera-Devicefilter, wird eine einfache Kantendetektion auf dem Bild berechnet. Dazu werden nacheinander die einzelnen Schritte Gaußglättung, Grauwertkonvertierung, Sobelfilterung und abschließende Cannyfilterung durchgeführt. Zu sehen ist, wie verschiedene Filter eine unterschiedliche Anzahl von Eingabebildern verwenden. Derselbe Filter, in diesem Fall der Sobel-Filter, kann mehrmals mit unterschiedlicher Parametrisierung, in diesem Fall die Richtung der Sobelmaske, instantiiert werden. Würde man den Wurzelknoten austauschen, könnte man den Rest des Baums unverändert für einen Einsatz mit







gespeicherten Bildern verwenden. Eine ausführlichere Darstellung der verschiedenen Aspekte des *VIP-Frameworks* findet sich in [133], [134] oder [135].

4.3 Sonstige Bibliotheken

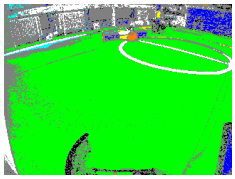
Um eine möglichst kurze Bearbeitungszeit der entwickelten Filtermodule zu garantieren, wird zusätzlich eine von der Firma Intel stammende Bibliothek [64] verwendet. Diese *Intel Performance Primitives* (IPP) stellen grundlegende Bildverarbeitungsprimitive, Methoden zur Signalverarbeitung und Funktionen zur Berechnung kleiner Matrizen zur Verfügung. Alle Funktionen sind durch die Verwendung der MMX- und SSE-Einheiten [12, 100, 127] hochgradig geschwindigkeitsoptimiert. Zwar ist der Geschwindigkeitsgewinn auf Intel-Prozessoren am größten, die Funktionalität ist aber auf allen x86-Prozessoren dieselbe.

Die zur Verfügung stehenden Bildbearbeitungsoperatoren sind unter anderem logische und arithmetische Operationen, Farbkonvertierungsfunktionen, Vergleichsfunktionen, morphologische und Filter-Funktionen und Bildstatistiken. Die Wahl der verwendeten Verfahren ist daher teilweise auch durch den Einsatz dieser Bibliothek begründet. So wird zur Kantendetektion der Canny-Operator verwendet, obwohl vermutlich auch ein einfacheres Verfahren über einen festen Schwellenwert auf den Gradientenbildern ausreichen würde. Ein solcher selbstprogrammierter Algorithmus erwies sich aber als deutlich langsamer als der sehr schnelle Canny-Operator aus der IPP Bibliothek.

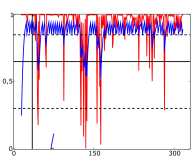
Die zweite verwendete Bibliothek ist EpsiloNN [124, 125]. EpsiloNN umfasst eine Beschreibungssprache für neuronale Netze, die anschließend in den für die jeweilige Plattform passenden C-Code übersetzt wird. Die Syntax der Beschreibungssprache ist an C++ angelehnt. Der resultierende Code ist jeweils speziell an die verwendete Plattform angepasst, ohne daß in der Netzwerkbeschreibung speziell darauf eingegangen werden muß. Desweiteren wird das entstehende neuronale Netz immer durch dieselben Aufrufmethoden angesprochen. Ein Austausch der kompletten Netzwerkarchitektur läßt sich damit einzig durch eine Änderung der Netzbeschreibung erreichen, ohne daß zusätzliche Anpassungen in den verwendeten Filtern nötig sind.

4.4 Kernpunkte





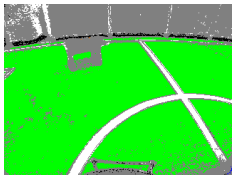
Um die gesamte hier vorgestellte Arbeit auch in anderen Bereichen verwenden zu können und eine Anpassung an verschiedene Szenarien so leicht wie möglich zu machen, wurde bereits softwareseitig auf die Verwendung standardisierter Komponenten und Bibliotheken geachtet. Jedes der hier benutzten Softwarepakete läßt sich leicht auf andere Plattformen, Prozessoren oder anderen Hardwarekomponenten anpassen. So läuft MIRO beispielsweise auf allen uns zur Verfügung stehenden Robotern wie den Sparrows-99 und Sparrows-2003, einem B21 Roboter der Firma RWI und einem PeopleBot von ActivMedia. In anderen Laboren wurden zusätzliche Anbindungen,

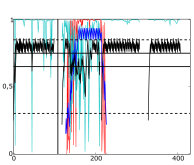


sowohl für weitere Roboter aus dem RoboCup-Umfeld als auch andere Plattformen auch für den Einsatz im Freien entwickelt.

Mit der gleichen Designphilosophie wurde das *VIP-Framework* für die Bildverarbeitung auf autonomen mobilen Robotern entwickelt. Auch hier standen Punkte wie Wiederverwendbarkeit von Komponenten, Abstraktion von hardwarespezifischen Roboterbelangen sowie die Konzentration auf weiche Echtzeitfähigkeit im Vordergrund. Echtzeitfähigkeit wird unter anderem durch die bedarfsabhängige Auswertung von Teilbäumen oder das konsequente Vermeiden unnötiger Kopieroperation erreicht (siehe auch Abschnitt 8.2). Wiederverwendbarkeit wird durch das Konzept von leicht zu wartenden und austauschbaren Modulen erreicht. Dabei ist die kleinste Moduleinheit ein Filter, wie er in Abschnitt 4.2 beschrieben wurde. Alle im Laufe dieser Arbeit entstandenen Filter sind in einer Bibliothek namens libVideoFilters[2] zusammengefasst. Zusätzlich erleichtert der Einsatz von wiederverwendbaren Komponenten den Austausch mit anderen Projekten und Arbeitsgruppen. So wird ein Großteil der für diese Arbeit entwickelten Filtermodule etwa auch im EU-Projekt *MirrorBot* erfolgreich eingesetzt.







[2] libVideoFilters und Miro sind beide unter der GNU Public License veröffentlicht und können unter http://smart.informatik.uni-ulm.de von jedem Interessierten heruntergeladen werden.

5 Regionensuche

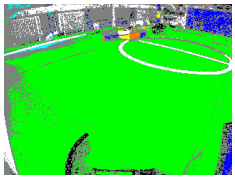
In diesem Kapitel wird der erste Berechnungsschritt des oben vorgestellten Gesamterkennungssystems im Detail vorgestellt: die Suche nach interessanten Regionen im Bild. Interessante Regionen sind in dieser Arbeit stets rechteckige Bereiche innerhalb des aufgezeichneten Bildes, in denen sich mit hoher Wahrscheinlichkeit ein Objekt des gesuchten Objekttyps befindet. Wichtig in diesem Zusammenhang ist, daß jeweils eine spezielle Regionensuche für jeden Objekttyp existiert. Die gefundenen Regionen sind daher nicht Bereiche, die unspezifisch einen möglicherweise interessanten Bereich darstellen, sondern jeweils eine Hypothese für das Vorhandensein eines Objekts einer konkreten Objektklasse in diesem Bildbereich. Die Bezeichnung „Aufmerksamkeitssteuerung“ wird hier synonym zu dieser Regionensuche verwendet, auch wenn diese Terminologie normalerweise für stärker biologisch motivierte Verfahren verwendet wird [66].

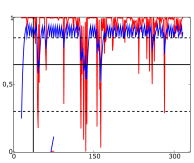
Die Suche nach Regionen wird immer auf dem gesamten Bild ausgeführt. Aus diesem Grund muß die Aufmerksamkeitssteuerung sehr schnell sein, um alle Bilder in ausreichender Geschwindigkeit und Genauigkeit bearbeiten zu können. Umgekehrt sind die verwendeten Merkmale deswegen meist auch sehr einfach, wie beispielsweise Farbinformationen oder einfache geometrische Größeninformationen.

Natürlich kann ein derart einfaches Verfahren auch nicht perfekt sein. Es ist immer eine Abwägung zwischen den verschiedenen Zielen, wie schnell und wie genau diese Aufmerksamkeitssteuerung sein soll. Da alle nachfolgenden Arbeitsschritte nur noch auf diesen gefundenen Regionen ausgeführt werden, ist es wichtig, daß alle vorkommenden Objekte im Bild auch gefunden werden. Zu viele Regionen, die keine interessanten Bereiche des Bildes markieren, erhöhen den Rechenaufwand aber deutlich. Zusätzlich steigt die Wahrscheinlichkeit für Fehlklassifikationen in den folgenden Arbeitschritten.

Im folgenden Abschnitt wird zuerst die Bildvorverarbeitung, die Farbindizierung des Eingangsbildes, die dazu verwendeten Farbräume sowie die Festlegung der Bereichsgrenzen, erklärt. Im zweiten Abschnitt folgen dann die eigentliche Auswahl der Objekthypothesen sowie die dafür verwendeten modellbasierten Ausschlußkriterien. Danach wird in Abschnitt 5.3 beschrieben, warum für einige Objekte kein nachfolgender neuronaler Klassifikator verwendet wird, sondern die Aufmerksamkeitssteuerung bereits die komplette Berechnung darstellt. Im darauffolgenden Abschnitt wird das verwendete Verfahren dann im Kontext anderer Arbeiten zur Aufmerksamkeitssteuerung diskutiert. Hier sind insbesondere Verfahren aus dem RoboCup und anderen dynamischen Szenarien wie dem Automobilbereich von Interesse. Abschließend werden im letzten Abschnitt die Kernpunkte des beschrieben Verfahrens zusammengefaßt.







5.1 Farbindizierung

Wie bereits erwähnt, sind alle Objekte im RoboCup durch eindeutige Farben gekennzeichnet. Der Ball ist orange, das Spielfeld grün und die Tore blau und gelb. Aus diesem Grund ist die Aufmerksamkeitssteuerung als Farbbereichssuche in einem farbindizierten Bild realisiert. Das verwendete Verfahren besteht im Wesentlichen aus drei Schritten:

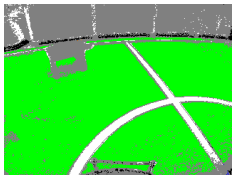
> Farbindizierung
> 1 Farbkonvertierung (Abbilden von Farbwerten in den HSV-Farbraum)
> 2 Farbklassifikation (Zuordnen von Farbwerten zu Farbklassen)
> 3 Bereichserkennung (Finden von Regionen mit gleicher Farbklasse)

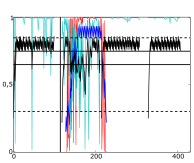
Im ersten Schritt werden die von den Kameras gelieferten YUV-Farben in den HSV-Farbraum transformiert [46]. Anschließend wird jedem HSV-Farbwert eine Farbklasse zugeordnet. Im letzten Schritt werden dann zusammenhängende Bereiche, also Bereiche mit homogenem Farbklassenwert gesucht. Diese drei Schritte werden im folgenden erläutert.

5.1.1 Farbkonvertierung

Abhängig von der verwendeten Kamera werden in unserem Robotersystem die Kamerabilder im YUV-Farbraum mit verschiedenen Kodierungsformaten geliefert. Um die Qualitätsverluste, wie sie in Abbildung 2.6 illustriert sind, zu verstehen, muß man wissen, daß die gerichtete Kamera die Bilder im sogenannten YUV-4:2:2-Format liefert, die omnidirektionale Kamera im YUV-4:2:0-Format. Die Y-Komponente des YUV-Farbraums ist der Helligkeitswert eines Bildpunktes, die U- und V-Komponenten sind die Farbwerte des Farbraums. Da sich 30 Bilder pro Sekunde in voller Auflösung nicht über die verwendete Firewire-Schnittstelle, bzw. bei reduzierter Auflösung über die USB1-Schnittstelle übertragen lassen, werden die Aufnahmen zusätzlich noch verlustbehaftet komprimiert. Bei der YUV-4:2:2-Kodierung werden jeweils vier quadratisch angeordnete Pixel durch vier Y-Werte, aber nur je zwei U- und V-Werte kodiert. Bei der YUV-4:2:0-Kodierung werden jeweils vier wiederum quadratisch angeordnete Pixel gar nur durch vier Y-Werte und jeweils einen U- und V-Wert repräsentiert.







Der HSV-Farbraum wiederum ist ein zirkulärer, kegelförmiger Raum mit den Dimensionen *Hue* (Farbwinkel), *Saturation* (Sättigung) und *Brightness* (Helligkeit). *Hue* ist der eigentliche Farbwert des Bildpunktes. *Saturation* bestimmt den Grad der Sättigung der Farbe. Komplett gesättigte Farben sind die Farben in ihrer Reinform. Je weniger Sättigung eine Farbe besitzt, desto eher erscheint sie unbunt (schwarz über grau bis weiß). *Brightness* (oft auch einfach Value genannt) beschreibt als dritte Dimension den Helligkeitswert des Pixels. Schwarz liegt in der Spitze des Kegels, Weiß im Zentrum des Bodens des Kegels und die reinen Farben an den Außenseiten des Kegelbodens. Die kegelartige Struktur versinnbildlicht, daß der Farbwert mit abneh-

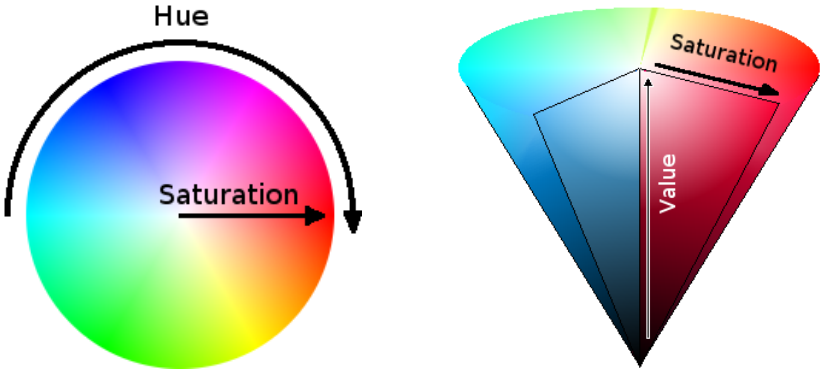


Abbildung 5.1: Illustration der Struktur des HSV-Farbraums. Links die Draufsicht auf den Farbkegel, rechts die 3-dimensionale Struktur bzw. ein Schnitt durch diesen Kegel. Zu sehen ist, wie an der Außenkante des Schnittes die reine Farbe, an der Innenseite reines Weiß oben, und reines Schwarz unten liegt.

mender Helligkeit an Bedeutung verliert. In Abbildung 5.1 ist der HSV-Farbraum graphisch illustriert.

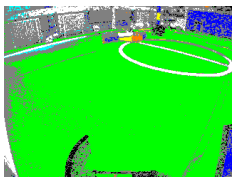
Der Vorteil des HSV-Farbraums besteht nun darin, daß Farben unabhängig von ihrer Ausleuchtung immer im gleichen Winkelbereich des Farbraums zu liegen kommen. Ein dunkles Rot hat den gleichen *Hue*-Wert (und denselben anteiligen radialen *Saturation*-Wert) wie ein helles Rot.

Die Konvertierung des YUV- in den HSV-Farbraum wird am einfachsten in zwei Schritten erklärt. Erst wird der YUV-Farbraum mittels folgender Vorschrift in den RGB-Farbraum abgebildet:

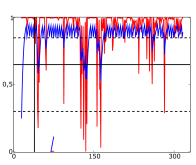
$$R = Y + 1{,}140V \tag{5.1}$$
$$G = Y - 0{,}394U - 0{,}581V \tag{5.2}$$
$$B = Y + 2{,}032U \tag{5.3}$$

Dabei sind R, G, B die entsprechenden Komponenten des RGB-Farbraums, Y, U, V jeweils die YUV-Komponenten. Alle Werte sind normiert, so daß gilt: $R, G, B, Y \in [0, 1]$, $U \in [-0{,}436, 0{,}436]$, $V \in [-0{,}615, 0{,}615]$. Die zur Konvertierung verwendeten Koeffizienten entsprechen denen für das Dekodieren eines analogen PAL-Signals. Für andere Kameramodelle können andere Koeffizienten nötig sein.





Im zweiten Schritt wird nun der RGB-Farbwert in den endgültigen HSV-Farbraum übertragen. Entsprechend der oben genannten Skalierung sind die resultierenden Farbwerte auch hier skaliert auf $H \in [0, 360[$, $S, V \in [0, 1]$. Sei nun MAX das Maximum der drei Farbkomponenten (R, G, B) und MIN entsprechend das Minimum, so gilt



folgende Transformationsvorschrift[1]:

$$H = \begin{cases} \text{undefiniert} & \text{falls } MAX = MIN \\ 60 \cdot \frac{G-B}{MAX-MIN} & \text{falls } MAX = R \text{ und } G \geq B \\ 60 \cdot \frac{G-B}{MAX-MIN} + 360 & \text{falls } MAX = R \text{ und } G < B \\ 60 \cdot \frac{B-R}{MAX-MIN} + 120 & \text{falls } MAX = G \\ 60 \cdot \frac{R-G}{MAX-MIN} + 240 & \text{falls } MAX = B \end{cases} \tag{5.4}$$

$$S = \begin{cases} 0 & \text{falls } MAX = 0 \\ 1 - \frac{MIN}{MAX} & \text{sonst} \end{cases} \tag{5.5}$$

$$V = MAX \tag{5.6}$$

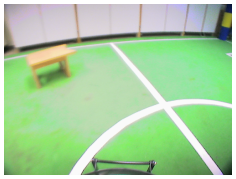
Für den Fall daß alle RGB-Farbwerte identisch sind, das heißt $MIN = MAX$ (entspricht Grau), liegt der resultierende HSV-Farbwert genau in der Mitte des Kegels. Der *Hue*-Wert ist damit bedeutungslos und folglich nicht definiert.

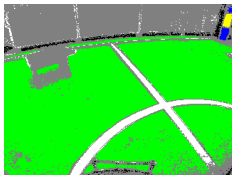
5.1.2 Farbklassifikation

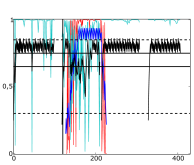
Bei der Farbklassifikation werden alle Farbwerte auf einen Satz von Farbklassen abgebildet. Diese Farbklassen beschreiben alle für den RoboCup relevanten Zielfarben zuzüglich einer Sammelklasse für Farbwerte, die nicht zugeordnet werden können. Durch die spezielle Struktur des HSV-Farbraums lassen sich Farbklassen ausreichend gut durch die Festlegung von oberen und unteren Bereichsgrenzen in allen Dimensionen definieren.

Im Detail bedeutet dies: $FR \mapsto FK$ aus einem Farbraum FR auf einen endlichen Satz FK von Farbklassen. In unserem Fall gilt: $FR = H \times S \times V$ sei der HSV-Farbraum mit $H = [0, \ldots, 360[$ und $S, V = [0, \ldots, 255]$ $\forall$ $(h, s, v) \in FR$ und $FK = \{Blau, Gelb, \ldots, Magenta\} \cup \{Grau\}$ bestehend aus den relevanten Farbklassen sowie der Sammelklasse $Grau$.

Die Festlegung dieser Bereichsgrenzen geschieht manuell. Ein ursprünglich entwickeltes Verfahren zum automatischen Finden dieser Regionen [95] ist durch die kontinuierlichen Regeländerungen, wie sie in Kapitel 2.4 bereits beschrieben wurden, nicht mehr anwendbar. In Abbildung 5.2 ist ein Beispiel für eine solche Abbildung von Originalfarbwerten nach Farbklassen zu sehen. Deutlich zu sehen ist, wie das gesamte Spielfeld grün, das Tor gelb und die Eckpfosten gelb und blau eingefärbt werden. Ebenfalls zu sehen ist die magentafarbene Markierung an der Oberseite des Roboters.







Wenn sich die Zielklassen stark voneinander unterscheiden, muß beim Festlegen nur zwischen der jeweiligen Farbklasse und der Sammelklasse abgewogen werden. Wenn aber aufgrund einer schwierigen Ausleuchtung des Spielfelds bereits die Zielklassen sehr ähnlich sind oder sich gar überlappen, ähnliche oder gleiche Farbwerte im Ausgangsbild daher eigentlich zwei verschiedenen Farbklassen zugeordnet werden sollten, muß bei der Festlegung der Bereichsgrenzen eine Abwägung zwischen den möglichen Zuordnungen erfolgen.

[1] Transformationsformel sowie Abbildung 5.1 nach http://de.wikipedia.org/wiki/HSV-Farbraum

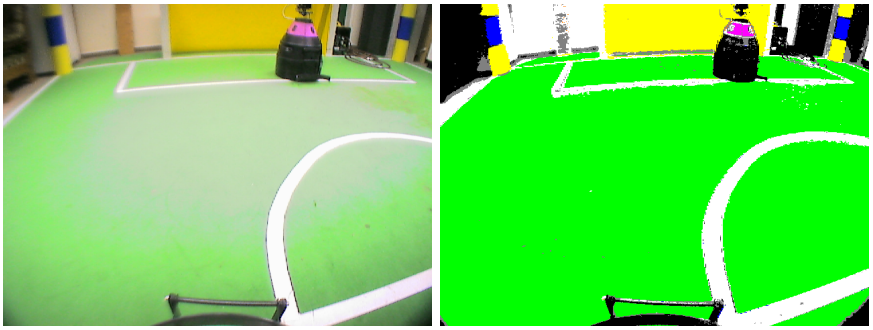


Abbildung 5.2: Beispiel für die Abbildung von Farbwerten des Originalbilds (links) auf Farbklassen im farbindizierten Bild (rechts).

Der Vorteil einer solchen Klassendefinition durch feste Bereichsgrenzen besteht darin, daß sich die Abbildungsfunktion mit Hilfe einer vorab berechneten Abbildungstabelle realisieren läßt. Variable Abbildungsfunktionen, wie sie von anderen Mannschaften eingesetzt werden (siehe Abbschnitt 5.4), erfordern wesentlich mehr Berechnungsaufwand. Durch die nachgeschaltete, sehr selektive Objektklassifikation kann ein solcher Aufwand an dieser Stelle aber minimiert werden.

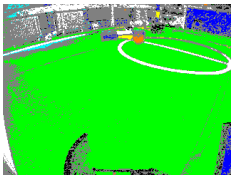
5.1.3 Bereichserkennung

Zur Farbblobsuche existieren viele Ansätze in der Literatur. In der IPP-Bibliothek (siehe Abschnitt 4.3) findet sich eine Funktion zum Füllen von Bereichen mit gleicher Farbe im Bild (ein sogenanntes flood-filling), die sich optimal für diese Zwecke verwenden läßt. Vorab wird das farbindizierte Bild einmal erodiert und dilatiert, um feine Übergänge zwischen den einzelnen Farbbereichen zu eliminieren. Danach werden alle Bereiche der gesuchten Farbe im Bild mit Hilfe dieser Füllfunktion eingefärbt. Das Resultat der Funktion enthält die Größe des gefüllten Bereichs sowie den Füllgrad. Weitere realisierte Versionen, die über eine reine Farbblobsuche hinausgehen, sind in der Diplomarbeit von Ulrich Kaufmann [76] nachzulesen.

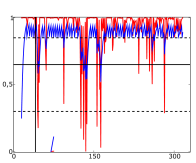
In weiteren einfachen Bearbeitungsschritten werden nahe beieinanderliegende oder übereinanderliegende Bereiche ähnlicher Größe bzw. überlappende Bereiche ähnlicher Breite zu einem Bereich zusammengefasst. Ebenso werden Bereiche, die innerhalb anderer Bereiche liegen, verworfen.

5.2 Modellbasierte Regionenselektion





Durch eine schlecht gewählte Farbindizierung sowie unbekannte Objekte auf und neben dem Spielfeld können bei der Regionensuche, wie sie bisher beschrieben wurde, auch Bereiche gefunden werden, die nicht den gesuchten Objekten entsprechen. Die Abbildungen 5.3 und 5.4 zeigen zwei Beispiele, die entweder durch schlecht eingestellte Farbbereichsgrenzen oder durch unbekannte Objekte im Bild falsche Regionen



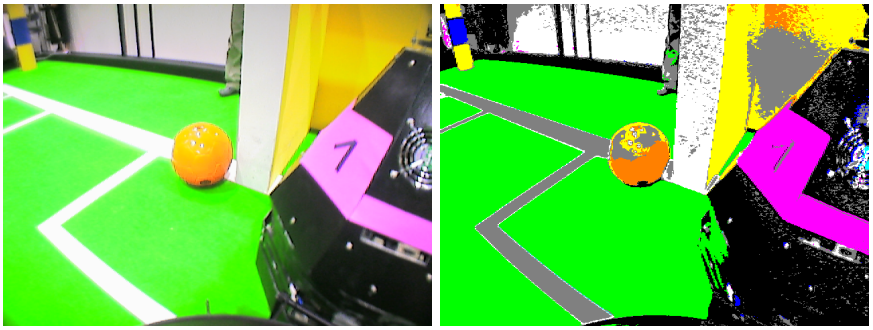


Abbildung 5.3: Beispiele für eine nicht optimale Farbindizierung durch sich stark ähnelnde Farbwerte von Gelb und Orange. Teile des Balls und große Bereiche des Tores sind durch die schlechte Indizierung in der Farbe des jeweils anderen Objekts eingefärbt. Dadurch steigt die Gefahr, daß einerseits Objektinstanzen nicht korrekt erkannt werden (Ball nur unzureichend orange) und andererseits, daß vermehrt falsche Objekthypothesen generiert werden (Teile des Tores eventuell orangener Ball).

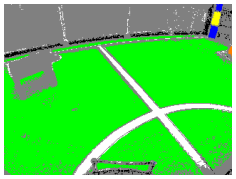
liefern würden. Abbildung 5.3 zeigt, was passieren kann, wenn sich zwei Farbwerte, in diesem Fall Rot und Gelb, sehr stark ähneln. Teile des Balls und große Bereiche des Tores sind durch die schlechte Indizierung in der Farbe des jeweils anderen Objekts eingefärbt. Dies kann unter Umständen dazu führen, daß die erkannte Größe des Balls zu stark von seiner tatsächlichen Größe abweicht und er deswegen nicht berücksichtigt wird. Zusätzlich sind die Bodenlinien auf dem Spielfeld nicht korrekt als Weiß wiedergegeben.

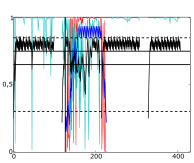
In Abbildung 5.4 ist zu sehen, wie ein Gegenstand im farbindizierten Bild einem gesuchten Objekttyp sehr ähnlich sein kann. In diesem Fall gleicht die rote Hose eines kleinen Kindes am Spielfeldrand einem etwa gleich großen Ball.

Durch zusätzliche, einfach zu berechnende Eigenschaften sollen nun modellbasiert möglichst viele solcher unerwünschten Regionen von der weiteren Verarbeitung ausgeschlossen werden. Die mangelnde Generalisierbarkeit dieser zusätzlichen, heuristischen Parameter ist dabei der realzeitlichen Anforderung des Szenarios geschuldet.

Der Füllgrad der Region, wie er direkt von der IPP-Füllfunktion geliefert wird, ist die erste Eigenschaft, die sich hierzu verwenden läßt. Bereiche, welche nur wenige Bildpunkte der eigentlich gesuchten Farbe enthalten, entsprechen mit hoher Wahrscheinlichkeit nicht einem Bereich gleicher, homogener Farbe im Originalbild, wie er für Objekte im RoboCup typisch ist. Diese lassen sich folglich über einen einfachen Schwellenwert für den relativen Füllgrad verwerfen.







Die zweite einfach zu berechnende Eigenschaft ist die Größe des Objekts. Derzeit wird davon ausgegangen, daß sich alle gesuchten Objekte auf dem Boden befinden. Es läßt sich damit aus der Lage im Bild und der bekannten Montagehöhe und -neigung der Kamera sowie der vorab ermittelten Linsenverzerrung über eine Abbildungsfunktion die reale Größe des Objekts ermitteln. Da die Größe der meisten Objekte bekannt ist, oder zumindest durch die RoboCup-Regeln in ihrer Minimal- und Maximalgröße

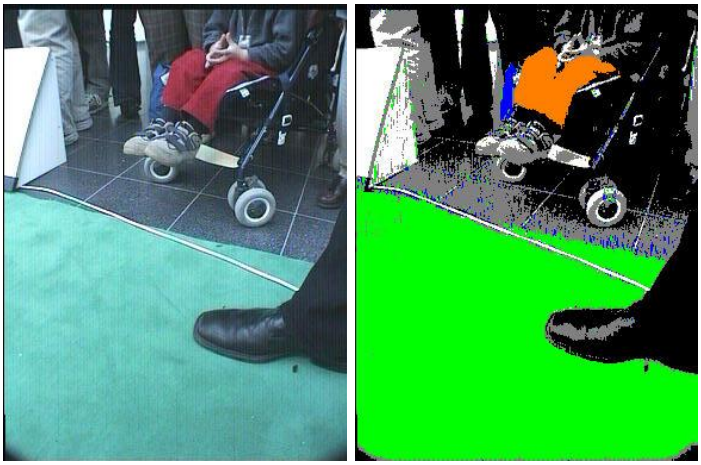


Abbildung 5.4: Beispiel, wie ein farbiger Gegenstand im farbindizierten Bild sehr ähnlich einem gesuchten Objekttyp erscheinen kann. In diesem Fall gleicht die rote Hose eines kleinen Kindes am Spielfeldrand einem etwa gleich großen Ball.

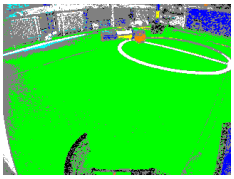
beschränkt sind, kann eine Region, deren reale Größe stark davon abweicht, durch einen einfachen Test ebenfalls verworfen werden. Gegeben sei eine Abbildungsfunktion $(a_1, b_1) \mapsto (z_1, \gamma_1)$, welche den linken unteren Bildpunkt (a_1, b_1) auf einen egozentrischen Satz von Distanz z_1 und Winkel γ_1 zum Roboter abbildet. Eine weitere Berechnung für die rechte untere Ecke der Region ergibt $(a_1 + c_1, b_1) \mapsto (z_2, \gamma_2)$. Lassen sich beide unteren Eckpunkte der Region tatsächlich auf den Boden projizieren, ist die reale Breite des Objekts dann $B = \sqrt{z_1^2 + z_2^2 - 2z_1 z_2 \cos|\gamma_1 - \gamma_2|}$.

Weitere Beschränkungen ergeben sich durch die räumliche Lage der Objekte im Bild. Durch die eingestellte Neigung der Kamera auf dem Roboter beginnen Tore unabhängig von der tatsächlichen Entfernung immer am oberen Bildrand. Alle Objekthypothesen für Tore, die diesen Anforderungen nicht genügen, etwa nur im unteren Drittel des Bildes liegen, können ebenfalls aussortiert werden.

5.3 Objekttypen ohne neuronalen Klassifikator

Nicht alle Gegenstände werden anschließend durch einen neuronalen Klassifikator validiert. Hierzu gehören die Torbereiche, die Torpfosten sowie die Bodenlinien. Diese Objekte besitzen entweder nicht genügend unterschiedliche Eigenschaften, die sich sinnvoll durch das verwendete Verfahren der Merkmalsextraktion und Klassifikation ermitteln und validieren lassen, oder sind durch ihre Größe im Bild selten komplett zu sehen. Abbildung 5.5 zeigt verschiedene Ansichten des gelben Tores, wie sie im Laufe eines Spiels durch den Roboter zu sehen sind. Die Spannbreite der Ansichten reicht von der Sicht quer über das komplette Spielfeld im ersten Bild, von außerhalb des Spielfelds im zweiten Bild bis zu formatfüllenden Ansichten, wenn der Roboter selbst beinahe im Tor, steht wie in den letzten Bildern. Zusätzlich können Teile des Tores durch eigene oder gegnerische Roboter verdeckt sein. Diese teilen den zu erkennenden freien Bereich im Tor in mehrere Teile.





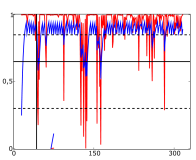
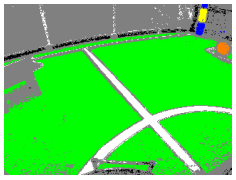


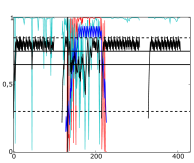


Abbildung 5.5: Verschiedene Ansichten des gelben Tores, wie sie im Laufe eines Spieles entstehen können. Hierbei können Roboter die freie Sicht auf das Tor behindern und das Tor in der Ansicht in zwei oder mehr Teile gliedern.

Torpfosten und Bodenlinien wiederum sind nicht mehr als abrupte Farbübergänge entlang einer Kante im Bild. Genau diese Kanteneigenschaft wird nun im weiteren verwendet, um gefundene Bereichshypothesen für diese Objekttypen weiter einzuschränken. Reine Farbkanten können auch durch eine nur unzureichende Indizierung des Farbbilds entstehen. Durch die binäre Verknüpfung von Kanten eines Farbindexbildes und denen eines Grauwertbildes können die gefundenen Bereiche nun auf tatsächliche Farbübergänge im Originalbild reduziert werden. Abbildung 5.6 zeigt die aus einem ungenau indizierten Bild resultierenden Kanten im Grauwertbild, im Farbindexbild sowie im Ergebnis der Kombination beider Kantenresultate. Über die Gradientenrichtung der Farbübergänge im Bild lassen sich weitere Rückschlüsse ziehen. So kann zum einen zwischen linkem und rechtem Torpfosten des jeweiligen Tores unterschieden werden, sowie eine grobe Richtung der gefundenen Bodenlinien als weiteres Merkmal für die Weiterverarbeitung derselben extrahiert werden.







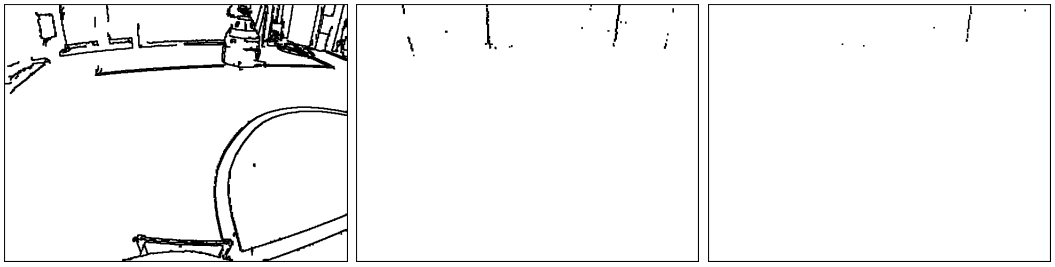


Abbildung 5.6: Im linken Bild ist das Resultat der Kantensuche im Grauwertbild zu sehen. Das Originalbild entspricht dem aus Abbildung 5.2 bzw. den Abbildungen 6.1 und 6.2. Das mittlere Bild wiederum ist das Ergebnis der Kantenberechnung auf einem Indexbild mit den Farben gelb und weiß. Das rechte Bild ist das Resultat der binären Addition der beiden ersten Bilder und das Ergebnis der Suche nach Torpfosten eines gelben Tores im Bild. Damit die Linien in der Abbildung besser erkannt werden können, sind diese hier verbreitert dargestellt.

5.4 Verwandte Arbeiten

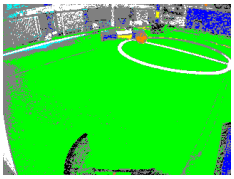
Wie zu Beginn erwähnt, bezeichnet der Begriff Aufmerksamkeitssteuerung eigentlich eher biologisch motivierte Verfahren. Solche Verfahren sind meist nicht auf Geschwindigkeit optimiert und durch ihre inhärent parallele Struktur auch sehr aufwendig [18, 26, 66]. Für das hier untersuchte Einsatzszenario scheiden diese Verfahren daher aus.

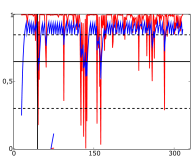
5.4.1 Automobilbereich

Andere, eher pragmatische, technisch motivierte Verfahren sind für das hier untersuchte Anwendungsgebiet deutlich relevanter. Solche Verfahren sind meist maßgeblich durch die zur Verfügung stehende physikalische Plattform und die sensorische Ausstattung sowie Eigenschaften der Einsatzumgebung beschränkt. So besteht eine der Möglichkeiten, um Objekte von ihrem Hintergrund zu trennen, darin, ihre räumliche Tiefe zu bestimmen. Eine solche Tiefenwahrnehmung erfolgt dabei meist mit Hilfe eines Stereokamerasystems [32, 44, 50]. Da ein solches Stereosystem aber die Verarbeitung doppelt so vieler Bilder bedingt wie einfache monokulare Kamerasysteme, scheidet ein solches Verfahren im RoboCup ebenfalls aus. Daneben wäre für viele Mannschaften aber auch der finanzielle Aufwand für die doppelte Anzahl an Kameras für die Roboter schlichtweg zu hoch.

In gewisser Weise damit verwandt ist der optische Fluß [60]. Durch die zeitliche Verfolgung charakteristischer Merkmale in Bildsequenzen lassen sich Bewegungsinformationen ermitteln, und so Objekte anhand ihrer unterschiedlichen Bewegungen voneinander trennen. Beim optischen Fluß werden diese Informationen aber mit Hilfe einer einzigen Kamera ermittelt [59, 126, 142]. Aber auch die Regionensuche mit Hilfe des optischen Flusses ist im RoboCup nur bedingt möglich. Grundlegende Voraussetzung für eine Schätzung des Flusses ist die Bewegung von entweder beobachtendem





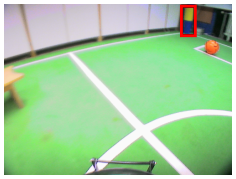


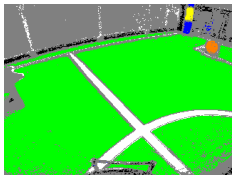
Roboter oder zu beobachtendem Objekt. Bei einem Stillstand von Objekt und eigenem Roboter kann keine solche Information geschätzt werden. Zusätzlich braucht man jeweils eine gewisse Menge an aufeinanderfolgenden Bildern, um Objekte zu finden. Dies widerspricht aber bereits dem Gedanken der Echtzeitfähigkeit und einer schnellen Reaktion auf veränderte Umgebungsbedingungen. Ein anderes Problem ist außerdem, daß wir auf unseren Robotern eine gerichtete und schwenkbare Kamera verwenden. Jedesmal, wenn die Kamera kurz von einem Objekt wegbewegt wird, müßte man danach erst wieder eine gewisse Zeit auf den vorherigen Punkt schauen, um weitere Informationen zu erhalten. Außerdem kann sich während dieser Zeit sehr wohl etwas an der Szenerie verändert haben, ohne daß man dies dann detektiert. Dieser Umstand, zusammen mit dem Problem der unbedingt nötigen Bewegung im Bild, machen auch den optischen Fluß für einen Einsatz im RoboCup eher ungeeignet.

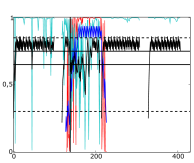
Weitere im RoboCup nicht praktikable Verfahren sind auch alle, die auf Wärmebildkameras zur einfacheren Objekt-/Hintergrundtrennung setzen [45, 131] sowie Verfahren, die entweder ausschließlich [41] oder in Kombination mit normalen Kameras [87] einen hochpräzisen Laserscanner einsetzen[2].

5.4.2 RoboCup

Die üblicherweise verwendeten Verfahren im RoboCup müssen die Regionensuche zumeist mit Hilfe nur einer Kamera und möglichst schnell auf wenigen Bildern durchführen. Da im Moment noch alle Objekte auf dem Spielfeld farbkodiert sind, bietet sich eine farbbasierte Suche natürlich an. Daher bauen auch beinahe alle anderen Mannschaften auf einem Verfahren ähnlich dem in dieser Arbeit vorgestellten auf. Die Frage, wie man das Originalbild in ein segmentiertes, farbindiziertes Bild umwandelt und die eigentliche Objekterkennung gehen dabei oft fließend ineinander über. Jamzad et. al. [67, 68] etwa beschreiben in mehreren Arbeiten, wie sie mit Hilfe perspektivisch angeordneter Unterabtastungspunkte das Bild nach bestimmten Indexfarben absuchen. Ist eine solche Farbe gefunden, bestimmt sich die Objektklasse durch die Anzahl an gleichfarbigen Scanpunkten, und damit die reale Objektgröße in der direkten Nachbarschaft des gefundenen Punktes. Eine algorithmisch besonders schnelle Form der Farbsegmentierung und anschließender Farbblobsuche findet sich in der Arbeit von Bruce et. al. [16]. Dabei werden die Farbindexzugehörigkeiten in einem Bitvektor kodiert, womit die Nachbarschaftsbeziehungen, und damit die Konnektivität gleichfarbiger Farbbereiche, in einem „run-length encoded“ Bild untersucht werden kann.







Ein interessanter Ansatz, wie versucht wird, dieses Abbildungsproblem möglichst einfach und eindeutig zu gestalten, wurde von Grillo et. al. veröffentlicht [47]. Anhand eines kleinen Satzes von manuell vorgegebenen Farbbereichen und den zugehörigen Objektklassen auf dem Kamerabild eines ruhenden Roboters werden die Kameraparameter wie etwa Belichtungszeit, Sättigung usw. in einem iterativen Prozess mit Hilfe

[2] Teilweise werden von einigen Mannschaften allerdings noch Laserscanner verwendet [34, 123], auch wenn seit dem Wegfall der Spielfeldbanden und aus Kostengründen dieser Sensor immer seltener verwendet wird.

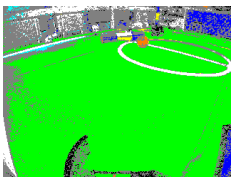
einfacher genetischer Algorithmen solange optimiert, bis der mittlere quadratische Abstand der Farbbereiche untereinander möglichst weit auseinander liegt. Dadurch vereinfacht sich der nachfolgende Schritt der Erstellung einer solchen Farbabbildung signifikant.

Desweiteren gibt es viele Arbeiten, wie nun die eigentliche Farbzuordnung definiert wird. Ähnlich wie auch in den bereits oben erwähnten Arbeiten von Jamzad versuchen viele, erst einen passenden Farbraum zu finden, in dem die Farbsegmentierung dann deutlich einfacher möglich ist. Ein Beispiel ist die Arbeit von Dahm et. al. [22], in der die ursprünglichen YUV-Farbwerte in einen sogenannten TSL*-Farbraum konvertiert werden. Dadurch lassen sich die Bereichsgrenzen einfacher durch horizontale und vertikale Schwellenwerte definieren. Die Parameter, die für diese Konvertierung nötig sind, werden mit Hilfe genetischer Algorithmen und einem Satz von Beispielfarben, bzw. Bildern, die nur bestimmte Farben enthalten, auf einen möglichst großen Abstand zwischen den Farben optimiert.

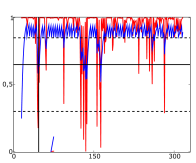
Ähnlich ist die Vorgehensweise in der Arbeit von Brusey und Padgham [17]. Auch sie wählen einen möglichst beleuchtungsunabhängigen Farbraum, um anschließend die Bereichsgrenzen semiautomatisch mit Hilfe des bekannten C4.5-Algorithmus durch einen hierarchischen Entscheidungsbaum festzulegen. Amoroso et. al. [2] gehen gar von einer nur eindimensionalen Repräsentation der *Hue*-Werte aus und lernen dann mit Hilfe von neuronalen feed-forward-Netzwerken die Abbildung von Farbwert nach Farbklassenzugehörigkeit. In der Arbeit von Sridharan und Stone [120, 122] werden sogar unterschiedliche Farbräume für verschiedene Objekte benutzt. In dieser Arbeit läuft ein AIBO-Roboter zu Beginn selbstständig auf dem Spielfeld herum, sucht dabei anhand vorgegebener Formbeschreibungen die relevanten Objekte und sammelt so selbstständig die zur Segmentierung nötigen Farbbeispiele. Ebenfalls in diese Gruppe fällt die Arbeit von Kikuchi et. al. [82], die ebenfalls durch einen neuen Farbraum eine erhöhte Beleuchtungsunabhängigkeit erwarten.

Den bisher vorgestellten Verfahren ist gemeinsam, daß sie alle mit festen, vorab definierten Abbildungsfunktionen arbeiten. Wie bereits erwähnt (und auch in Kapitel 9.1 untersucht), ist dieser Ansatz bei wechselnden Beleuchtungsbedingungen oder bei natürlichen Lichtverhältnissen nicht länger praktikabel. Um auch dann noch nur auf Basis farbindizierter Bilder eine valide Objekterkennung zu erreichen, gibt es viele Versuche, die Farbtabellen automatisch zu generieren oder über die Zeit an wechselnde Verhältnisse anzupassen. Eine interessante Mischform hin zu komplett variablen Bereichsgrenzen ist in der Arbeit von Sridharan und Stone [121] beschrieben. Im Prinzip werden hier mehrere feste Kalibrierungen für verschiedene Beleuchtungsbedingungen erstellt. Dann wird die Farbverteilung der aktuellen Aufnahme in bestimmten Abständen mit gelernten Farbverteilungen typischer Situationen verglichen. Anhand dieser Ähnlichkeiten wird dann zwischen den vorher festgelegten Kalibrierungen umgeschaltet.





Eine einfache, sich wirklich ändernde Kalibrierung beschreiben Jüngel et. al. [73]. Anhand einer modellbasierten Suche nach dem immer sichtbaren grünen Fußboden, welcher mit Hilfe einfacher Heuristiken kurz unterhalb des Horizonts vermutet wird, wird der die grünen Farbwerte repräsentierende Farbbereich jeweils mit den aktuell gefundenen Grünwerten verglichen und dann die Konfiguration entsprechend innerhalb



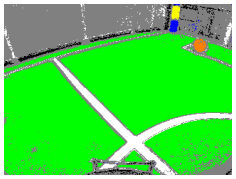
des YUV-Farbraums verschoben. Hierbei bleibt die Größe der definierten Farbbereiche sowie ihre relative Lage zueinander aber unverändert. In einer späteren Arbeit [72] wird dieser Ansatz dann noch mit Hilfe mehrerer hierarchisch aufgebauten Farbräume erweitert und verfeinert.

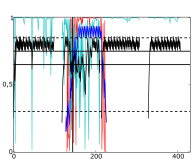
Gönner et. al. beschreiben ein Verfahren, welches auf Wahrscheinlichkeitsverteilungen basiert [51, 52]. Dabei entspricht die Auftrittshäufigkeit innerhalb eines Histogramms den a-priori-Wahrscheinlichkeiten, daß eine solche Farbwertkombination einem bestimmten Objekt zugehörig ist. Mittels Bayes'scher Verknüpfung aller Verteilungen der verschiedenen Objekte, kann dann so die benutzte a-posteriori-Wahrscheinlichkeit berechnet werden. Die Erstellung dieser Wahrscheinlichkeitsverteilungen geschieht durch eine enge Kopplung mit der späteren Objekterkennungsschicht (in diesem Fall eine Hough-Transformation zur Ballerkennung). Alle dadurch dem Objekt zugeordneten Farbwerte werden im adaptiven Fall wiederum dem Verteilungshistogramm hinzugefügt. Durch einen zusätzlichen Vergessensparameter werden alte, lange nicht gesehene Farb-/Objektzuordnungen wieder aus dem Histogramm entfernt.

Weitere Arbeiten dieser Art gibt es von Anzani et. al. [3], die mit Hilfe eines iterativen maximum-likelihood-Schätzers die Änderungen in der Beleuchtung mitverfolgen. Heinemann und Kollegen [58] beschreiben ebenfalls ein Verfahren das mit Hilfe einer bekannten, initialen Position und einer ganzen Reihe von szenarioabhängigen Annahmen die Abbildungsfunktion über die Zeit variieren.

Eine weitere interessante Mischform findet sich in der Arbeit von Lovell [90]. Durch die Untersuchung mehrerer Bilder, die unter verschiedenen Beleuchtungsbedingungen aufgenommen wurden, werden erst Farbwerte gesucht, die bei unterschiedlicher Illumination konstant immer der gleichen Farbklasse zugeordnet werden. Dann werden Bereiche auf einem Kantenbild gesucht. Die Mehrheit dieser darin enthaltenen beleuchtungsunabhängigen Farbwerte bestimmen die gesuchte Farbe des Bereichs. Es gibt aber auch Arbeiten, in denen auf die Berechnung einer Farbzuordnungstabelle für bestimmte Aufgaben völlig verzichtet wird. Einige Roboter lokalisieren sich inzwischen auf dem Spielfeld nur noch anhand der Linien auf dem Boden. Diese Linien werden nun nicht durch den grün-weißen Farbübergang detektiert, sondern anhand eines starken Gradienten im Luminanzkanal entlang von sogenannten scanlines auf dem Bild. Beschrieben wird dieses Verfahren unter anderem in der Arbeit von Merke, Welker und Riedmiller [98]. So gefundene Linienhypothesen müssen aber ebenfalls wieder mit Hilfe einfacher Heuristiken auf ihre Validität überprüft werden.







Interessant in diesem Zusammenhang ist die Tatsache, daß diese RoboCup-relevanten Arbeiten in der Zeit von 1999 bis 2006 publiziert wurden. Dies zeigt deutlich, daß auch nach Jahren der Forschung selbst in einem so definierten Szenario das zuverlässige Erkennen und Verarbeiten von Farbinformationen noch lange kein gelöstes Problem darstellt. Gleichzeitig wird aber bei einem Großteil dieser zitierten Arbeiten die nachfolgende Objekterkennung dann nur noch durch eine Farbblobsuche, scanlinebasierte Objektgrenzenerkennung oder ähnlich einfache Verfahren durchgeführt.

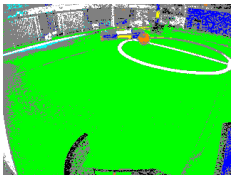
5.5 Kernpunkte

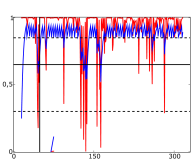
Das in diesem Kapitel dargestellte Verfahren erlaubt eine einfache und dabei extrem schnelle Suche nach Objekthypothesen im Bild (Zeitmessungen siehe Abschnitt 9.5.1). Durch die nachfolgende Objektklassifikation kann dieser Schritt mit einfachen, wenig rechenaufwendigen Verfahren realisiert werden. Die Verwendung fester Bereichsgrenzen für die Farbindizierung erlaubt eine extrem schnelle Indizierung des Bildes sowie zusammen mit der Verwendung hochoptimierter Füllalgorithmen und modellbasierter Regionenselektionsverfahren die problemlose Auswertung kompletter Bilder in diesem ersten Schritt. Für alle Objekte, welche nicht durch eine nachgeschaltete neuronale Klassifikation validiert werden, wurde eine Möglichkeit beschrieben, wie solche, nur durch Farbübergänge definierten Gegenstände mit Hilfe von Kanteninformation in einer ebenfalls sehr schnellen Art und Weise validiert werden können.

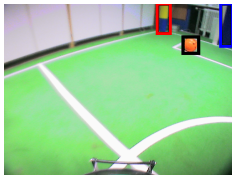
Wie in Kapitel 5.4 beschrieben, würden andere existierende Verfahren entweder einen weit größeren Rechenaufwand erfordern, setzen eine andere sensorische Ausstattung (z.B. mit Stereokamerasystemen) voraus oder basieren auf Annahmen, die im RoboCup-Umfeld nicht zutreffen.

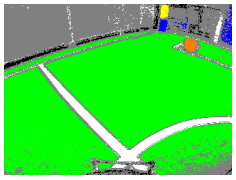
Auch aus diesem Grund stellen Verfahren, ähnlich der hier beschriebenen Regionensuche, immer noch die komplette Bildverarbeitung für einen Großteil der am RoboCup teilnehmenden Mannschaften dar. Wie von Kollegen und mir in [94] am Beispiel einer umfangreichen Fallstudie dargestellt, lassen sich solche Verfahren beim Einsatz unter wechselnden Beleuchtungsbedingungen aber nicht mehr sinnvoll mit Hilfe fester Abbildungstabellen für die Farbindizierung realisieren. Aus diesem Grunde existieren, wie oben beschrieben, viele unterschiedliche Verfahren zur dynamischen Anpassung dieser Farbbereichsgrenzen während des Spiels. Viele dieser Verfahren sind aber durch implizite und explizite Annahmen wahrscheinlich nicht auf Anwendungen außerhalb des RoboCup übertragbar. Eigene Arbeiten auf diesem Gebiet [95] sind durch Regeländerungen, insbesondere dem Wegfall der Banden rund um das Spielfeld, inzwischen ebenfalls nicht weiter anwendbar.

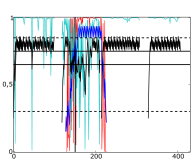











0,5
0
200
400

6 Objektklassifikation

In diesem Kapitel wird die Merkmalsextraktion auf den gefundenen Objekthypothesen beschrieben, sowie die anschließende Klassifikation dieser Merkmale erklärt. Objekte, für die dieser Bearbeitungsschritt ausgeführt wird, sind der Ball, die Eckpfosten und die Roboter auf dem Spielfeld. Da alle Objekte sehr spezielle Eigenschaften besitzen, unterscheiden sich diese Bearbeitungsschritte im Detail. Abhängig von der nötigen Flexibilität und Spezifität unterschieden sich die verwendeten Merkmale zwischen den unterschiedlichen Objekttypen.

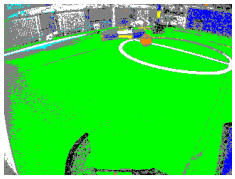
Die nötige Flexibilität ist ein wichtiger Grund für den Einsatz neuronaler Klassifikatoren. Neuronale Netze besitzen eine gute Generalisierungsleistung. Das bedeutet, daß sie nach einer initialen Trainingsphase auch solche Ansichten von Objekten korrekt validieren können, welche so in den Trainingsdaten nicht enthalten waren. Es ist weder möglich, für alle denkbaren Ansichten eines Roboters noch für alle möglichen Robotertypen eine ausreichende Anzahl von Trainingsbildern aufzunehmen, da viele Mannschaften zwischen zwei Turnieren komplett neue Roboter entwickeln, welche dann erst auf dem nächsten Turnier zum Einsatz kommen. Auch solche noch unbekannten Roboter müssen aber korrekt als solche erkannt werden.

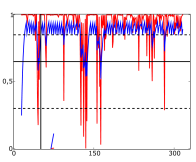
In den folgenden Abschnitten werden zuerst die verwendeten Merkmale beschrieben und die Gründe für die Verwendung eben dieser Merkmale dargelegt, sowie alle für die Extraktion dieser Merkmale notwendigen Vorverarbeitungsschritte erläutert. In einem weiteren Abschnitt folgt die Beschreibung der neuronalen Klassifikatoren: Der verwendeten Netzarchitektur, der eingesetzten Lernverfahren sowie eine Beschreibung des eigentlichen Lernvorgangs. Darauf folgt eine kurze Darstellung weiterer Regeln zur modellbasierten Selektionsverfeinerung. Abschließend wird wiederum ein Überblick über andere existierende Arbeiten in diesem Bereich gegeben, bevor der letzte Abschnitt die Kernpunkte der beschriebenen Verfahren mit einer Auflistung der hierbei relevanten Neuerungen zusammenfasst.

6.1 Merkmale

Aufgrund der Echtzeitanforderungen des Gesamtsystems können nicht beliebig viele Merkmale pro ermittelter Region berechnet werden. Daher müssen die tatsächlich berechneten Merkmale für jeden Objekttyp sorgfältig individuell ausgewählt werden. Insbesondere müssen diese dann besonders aussagekräftig sein, um eine möglichst hohe Spezifität für den jeweiligen Objekttyp zu garantieren, ohne ihre Generalisierungsleistung zu verlieren.







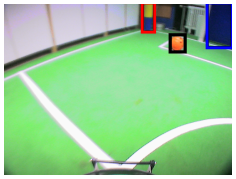
Für die drei zu klassifizierenden Objekttypen Ball, Eckpfosten sowie Roboter kommen die folgenden Merkmale zu Einsatz:

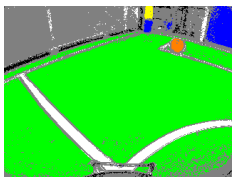
- Reale Größe bzw. Größenverhältnisse
- Farbanteile (der Index-Farbwerte)
- Histogramme der Originalfarben
- Orientierungshistogramme
- Kombinierte Orientierungs-/Krümmungshistogramme

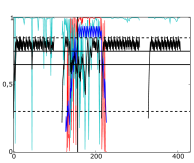
Die realen Größenverhältnisse berechnen sich, wie bereits in Abschnitt 5.2 beschrieben, über eine Abbildungstabelle, welche für jeden Punkt im Bild den korrespondierenden Punkt im egozentrischen, roboterbezogenen Weltmodell liefert[1]. Die Farbanteile der Index-Farbwerte ist ein Maß für die Anzahl der Pixel einer bestimmten Farbklasse innerhalb der gesamten Region oder verschiedenen Unterbereichen der gefundenen Region. Beide Kennzahlen werden für die Roboterklassifikation verwendet. Die Farbanteile stellen dabei unter anderem sicher, daß im Bereich der Roboterhypothese auch der vorgeschriebene Farbmarker des Roboters enthalten ist. Die Unterteilung in Unterbereiche ist deshalb sinnvoll, da die Regel vorschreibt, daß der Farbmarker eine bestimmte Größe (10×10 Zentimeter) besitzt und von allen Seiten zu sehen sein muß. Ein hoher Anteil der Farbmarkierung im mittleren Regionenbereich kann damit stärker bewertet werden; zusätzliche Vorkommen an den Randbereichen unterstützen die Hypothese weiter.

Im folgenden sollen nun die anderen, deutlich komplexeren Merkmale, die Histogramme, und deren Berechnung im Einzelnen vorgestellt werden. Alle diese Features werden hierbei nicht auf der gesamten Region, sondern gesondert auf mehreren Unterbereichen berechnet, der Bereich wird also in beispielsweise 3×3 zusätzliche Unterbereiche aufgegliedert. Diese Unterbereiche können sich an den Randbereichen um einen bestimmten Prozentsatz überlappen, um eine höhere Robustheit gegenüber nicht optimal bestimmten Regionen zu erhalten.

Die jeweiligen Histogrammwerte werden dann in eine frei wählbare Anzahl von Werteklassen diskretisiert. Es wird also die gewünschte Auflösung, z.B. der Orientierungsrichtungen im Orientierungshistogramm, je nach Einsatzzweck bestimmt. Diese sollten genau genug sein, um noch zwischen verschiedenen Richtungen unterscheiden zu können, aber auch möglichst wenige, um den Rechenaufwand nicht unnötig in die Höhe zu treiben.







Die Berechnung der Merkmale in Histogrammform stellt eine robuste, aber dennoch sehr generelle Form der Merkmalsbeschreibung dar. Im Gegensatz zu einer reinen Suche nach beispielsweise einer mehr oder weniger senkrechten Linie erlaubt die

[1] Ein egozentrisches Koordinatensystem ist roboterbezogen, der Ursprung des Koordinatensystems liegt in der Mitte des Roboters und Objekte verändern ihre Lage im Koordinatensystem bereits dadurch, daß sich der Roboter bewegt. Im Gegensatz dazu ist ein allozentrisches Koordinatensystem durch einen externen, globalen Referenzpunkt definiert. Roboter und Objekte bewegen sich innerhalb dieses Koordinatensystems.

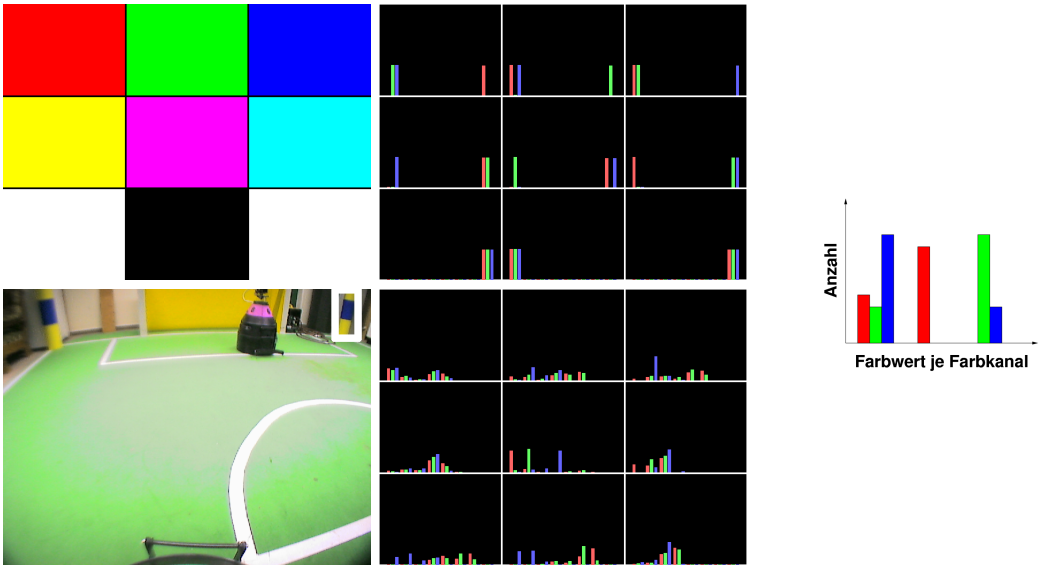


Abbildung 6.1: Farbhistogramme, extrahiert aus einer einfachen Illustration (oben) sowie aus der eingezeichneten Region einer tatsächlich gefundenen Aufmerksamkeitshypothese (unten). Die kleine Illustration rechts beschreibt die Bedeutung der Achsen der einzelnen Histogramme. Weitere Erklärungen hierzu entnehmen sie bitte dem Text.

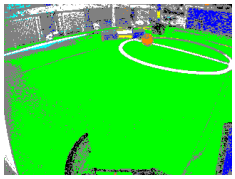
Darstellung in Histogrammform die Erkennung aller Linienbereiche in allen beliebigen Orientierungen. Was das „mehr oder weniger senkrecht“ dann zu bedeuten hat, wird während der Lernphase durch das neuronale Netz automatisch bestimmt und muß nicht mehr mühsam und fehleranfällig vom Entwickler selbst bestimmt werden.

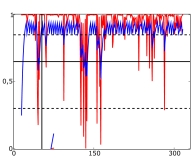
6.1.1 Farbhistogramme

Das erste der beschriebenen, komplexeren Merkmale ist das Farbhistogramm. Dieses Histogramm wird auf dem Originalbild bzw. den Originalfarben der gefundenen Regionen bestimmt. Ein Farbhistogramm wird für die Klassifikation des Balls sowie der Eckpfosten verwendet, da deren Farbe fest definiert ist und damit im voraus bereits bekannt ist. Die Farbe der Roboter sollte nach den RoboCup-Regeln zwar auch hauptsächlich aus Schwarz bestehen, wird in der Praxis aber nicht so streng gehandhabt.

Am einfachsten läßt sich dieses und die folgenden Merkmale jeweils anhand von Beispielen erklären. Abbildung 6.1 zeigt Beispiele für ein gefundenes Objekt sowie das daraus berechnete Farbhistogramm. Das Histogramm ist unterteilt in 3×3-Unterfenster, deren Anordnung den Bereichen im Originalbild entsprechen. Das linke, obere Teilhistogramm ist aus Werten des linken, oberen Bereichs der Originalregion berechnet. Entsprechend resultiert das mittlere, obere Teilhistogramm aus Farbwerten des mittleren, oberen Bereichs des Originalbildes und so weiter. Obwohl im realen Einsatz eine Überlappung der Teilbereiche die Erkennungsleistung deutlich stabilisie-







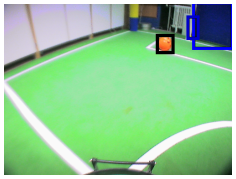
ren kann, sind in den folgenden Beispielen keine Überlappungen vorhanden, so daß sie leichter zu verstehen sind. In der oberen Abbildung ist ein artifizielles Bild zu sehen, an dem sich die Eigenschaften des Histogramms einfacher beschreiben lassen, im unteren Bild ein reales, aus dem eingezeichneten Bereich extrahiertes Farbhistogramm.

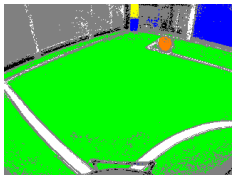
Auf der x-Achse des Histogramms ist die Intensität der Farbwerte aufgetragen, auf der y-Achse die Häufigkeit ihres Auftretens. Die drei verschiedenen Balken rot, grün und blau entsprechen den Farbwerten des jeweiligen RGB-Farbkanals. Im obersten linken Bereich des oberen Bilds wird die rein rote Farbe durch einen großen Wert des roten Balkens am rechten Ende der Histogrammskala entsprechend dem hohen Farbwert der roten Farbe repräsentiert. Blau und grün, die überhaupt nicht vorkommen, sind entsprechend durch ihren extrem kleinen Wert (Nullwert) am linken Ende der Skala aufgetragen. Bei Mischfarben, wie etwa Cyan in der mittleren Reihe ganz rechts sind entsprechend die grünen und blauen Histogrammbalken sehr häufig und dementsprechend weit rechts, während das Rot, das in Cyan nicht vorkommt, ganz links aufgetragen ist.

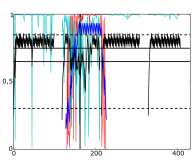
Das untere Histogramm in Abbildung 6.1 ist aus der weiß umrandeten Region extrahiert. Hier sind die Farbwerte nicht mehr so eindeutig verteilt wie in der oberen Illustration. Trotzdem lassen sich auch hier noch die verschiedenen Bereiche erkennen. Das mittlere zentrale Teilhistogramm, entstanden aus dem mittleren, zentralen Bereich der gefundenen Region, zeigt deutlich die wesentlich höheren Intensitäten der blauen Farbe im Vergleich zu den fast nicht existierenden Intensitäten (sehr kleinen Werte) im roten und grünen Farbkanal. Im mittleren Histogramm der unteren Reihe wiederum sind die roten und grünen Farbanteile deutlich über denen der blauen Farbe, entsprechend der gelben Farbe im Originalausschnitt.

6.1.2 Orientierungshistogramme

Der nächste verwendete Histogrammtyp ist das Orientierungshistogramm [38, 81]. Im Unterschied zu den Intensitäten der Farbwerte im vorhergehenden Abschnitt sind hier die Richtungen der Orientierungen in einem Histogramm aufgetragen. Entsprechend sind auf der x-Achse die verschiedenen Orientierungsrichtungen abgebildet, auf der y-Achse wieder die Häufigkeit ihres Auftretens. Wie im obigen Abschnitt sind auch hier in der Abbildung 6.2 zwei Beispiele angegeben; ein artifizielles zum besseren Verständnis sowie eine reale Roboteraufnahme (dieselbe Region des gleichen Originalbilds wie in Abbildung 6.1). Um wichtige Orientierungen am Rand einer Region sicher detektieren zu können, wird die gefundene Region hierzu um einen kleinen Prozentsatz vergrößert. Der genaue Vergleich der Region um den Eckpfosten in diesem und im obigen Bild läßt diesen Unterschied deutlich erkennen.







Zur Berechnung der Orientierungen wird das Bild bzw. die Region zuerst mit Hilfe eines waagrechten und eines senkrechten Sobeloperators gefaltet. Die resultierenden Bilder seien X für die horizontale Ableitung der Bildfunktion sowie Y für die entsprechende vertikale Ableitung. Die Richtung r des Gradienten an jeder Stelle läßt sich dann einfach trigonometrisch wie folgt berechnen: $r = tan^{-1}(X/Y)$. Entsprechend ist die Stärke s des Gradienten an der jeweiligen Stelle definiert über: $s = \sqrt{X^2 + Y^2}$.

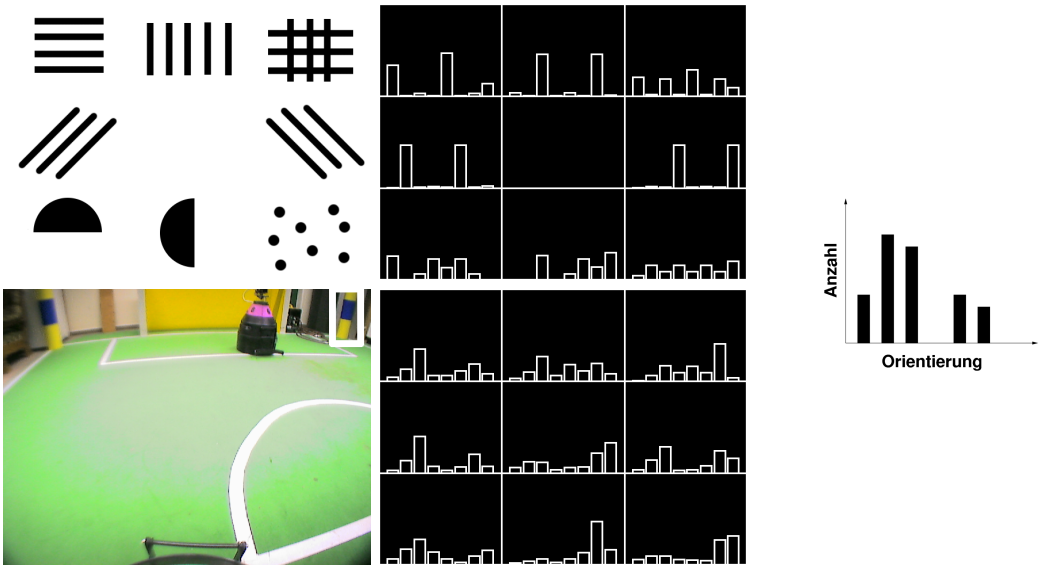


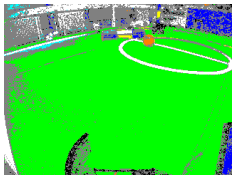
Abbildung 6.2: Orientierungshistogramme, extrahiert aus einem beispielhaften Gesamtbild (oben) sowie aus der eingezeichneten Region einer tatsächlich gefundenen Aufmerksamkeitshypothese (unten). Die kleine Illustration rechts beschreibt die Bedeutung der Achsen der einzelnen Histogramme. Weitere Erklärungen hierzu entnehmen sie bitte dem Text.

Beim Aufsummieren der Gradientenrichtung entsprechend ihrer Gradientenrichtung können diese dann über die jeweilige Gradientenstärke gewichtet werden.

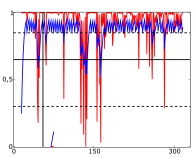
Das obere, linke Unterhistogramm der oberen Abbildung zeigt zwei Balken entsprechend den waagrechten Orientierungen im oberen, linken Bereich des Originalbildes. Die zwei Balken entstehen, da sowohl die Übergänge zwischen Weiß und Schwarz als auch der Übergang zwischen Schwarz und Weiß als zwei verschiedene Richtungen interpretiert werden[2]. Entsprechend sieht man im oberen, mittleren Histogramm ebenfalls zwei Balken, die aber entsprechend der gewählten Richtungsauflösung des Histogramms um zwei Einheiten verschoben sind. Das bedeutet, daß ein Balken im Histogramm einen Bereich von 45° der Orientierung umfasst. Die zusätzlichen kleineren Werte in den jeweils anderen Bereichen kommen durch die Endbereiche der Linienstücke sowie Diskretisierungsfehler an den Ecken der Linien zustande. Bei den Kreisen in der unteren, rechten Abbildung sind die Orientierungen im Histogramm entsprechend beinahe gleichverteilt.

In der realen Aufnahme sind entsprechend in den jeweils linken Unterhistogrammen die vorherrschenden, eher vertikalen Linien an den entsprechenden großen Werten der jeweiligen Orientierungen zu sehen. Die zusätzliche zweite Linie im mittleren, linken Histogramm kommt durch die im Hintergrund erkennbare Säule zustande, da hier der Richtungswechsel um 180° verschoben ist. Wie sich solche Artefakte vermeiden lassen,





[2] Die Filterimplementierung kann für andere Einsatzzwecke aber auch so parametrisiert werden, daß beide Gradienten als ein und dieselbe Richtung interpretiert werden.



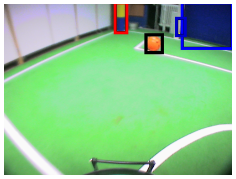
ist in Abschnitt 6.2 erklärt. Hier ist auch gut zu sehen, daß eine Suche nach genau vertikalen Linien, wie man sie für einen senkrecht stehenden Eckpfosten erwarten würde, nicht ausreichend ist. Welchem Teilhistogramm die waagrechten Linien an den blau-gelben sowie gelb-blauen Übergängen des Eckpfostens zugeordnet sind, ist nicht sofort erkennbar. Aus diesem Grund wird normalerweise die oben bereits erwähnte Überlappung der Teilhistogramme verwendet.

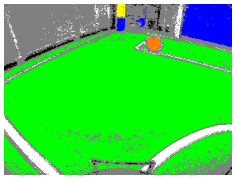
6.1.3 Kombinierte Orientierungs- und Krümmungshistogramme

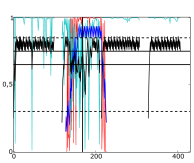
Der letzte verwendete Histogrammtyp ist ein mehrdimensionales Histogramm aus Orientierungsrichtungen und lokalen Krümmungen[3]. Entsprechend sind in den folgenden Abbildungen 6.3 auf der x-Achse die Orientierungsrichtungen, auf der y-Achse die Stärke der lokalen Krümmungen aufgetragen. Die Stärke der Krümmung steigt von unten nach oben; schwache Krümmungen entsprechen den kleinsten y-Werten im Histogramm, starke Krümmungen einem hohen y-Wert. Bildbereiche ohne irgendeine lokale Krümmung werden nicht berücksichtigt. Die Häufigkeit ihres Auftretens ist in diesem Fall über die Helligkeit des jeweiligen Histogrammwertes symbolisiert. Je heller der Histogrammwert dargestellt ist, desto häufiger kommt das entsprechende Merkmal im Bild vor. Um wichtige Strukturen am Rand der gefundenen Regionen nicht zu übersehen und eine unzureichende Erkennungsleistung der Bereichssuche zu kompensieren, wird auch hier die Region um einen bestimmten Prozentsatz vergrößert, bevor die eigentliche Merkmalsextraktion beginnt.

Die Orientierungsrichtungen und -stärken werden wie oben über die entsprechenden trigonometrischen Funktionen berechnet. Die lokalen Krümmungen werden entsprechend den Berechnungen für den Strukturtensor über die Verhältnisse der Eigenwerte ermittelt [71]. Ausgehend von den durch die zwei Sobeloperatoren gefalteten Bildern X und Y werden diese Bilder multipliziert ($XY_1 = X \cdot Y$) sowie jeweils quadriert ($X_1 = X^2$ bzw. $Y_1 = Y^2$). Diese Bilder werden nun jeweils mit einer 2-dimensionalen Gaußfunktion zu X_2, Y_2, XY_2 gefaltet. Die Eigenwerte sind dann einfach die Lösung des Polynoms 2. Ordnung. Die Größe der Gaußfunktion bestimmt dabei den Einflußbereich des zu beachtenden Umfelds rund um das jeweilige Pixel. Da sich die Faltung mit einem 2-dimensionalen Gauß separieren läßt, wird in der eigentlichen Implementation eine Faltung der Spalten und Reihen mit einer entsprechend großen, eindimensionalen Gaußfunktion ausgeführt.

Da im ersten Originalbild jeweils Kreise dargestellt sind, sind die Orientierungen im Histogramm jeweils gleichverteilt. Unterschiede in den Helligkeiten resultieren hier aus Diskretisierungsfehlern. Mit sinkendem Durchmesser der dargestellten Kreise im Bild steigt der Wert der lokalen Krümmungen, die gefunden werden und damit der y-Wert im Histogramm. Sind die Kanten der Kreise wie in den letzten vier Bildberei-







[3] Der Name Krümmung ist in diesem Zusammenhang nicht ganz korrekt. Eigentlich handelt es sich eher um die Eckigkeit einer lokalen Struktur, der Name Eckigkeitshistogramm scheint mir aber ungeeignet. Tatsächlich wird, wie später erklärt, die Hauptrichtung der lokalen Gradienten sowie ein dazu senkrechte stehender, zweiter Vektor ermittelt, und über das Verhältnis dieser beiden Gradientenstärken eine Aussage über die zugrundeliegende lokale Struktur ermittelt.

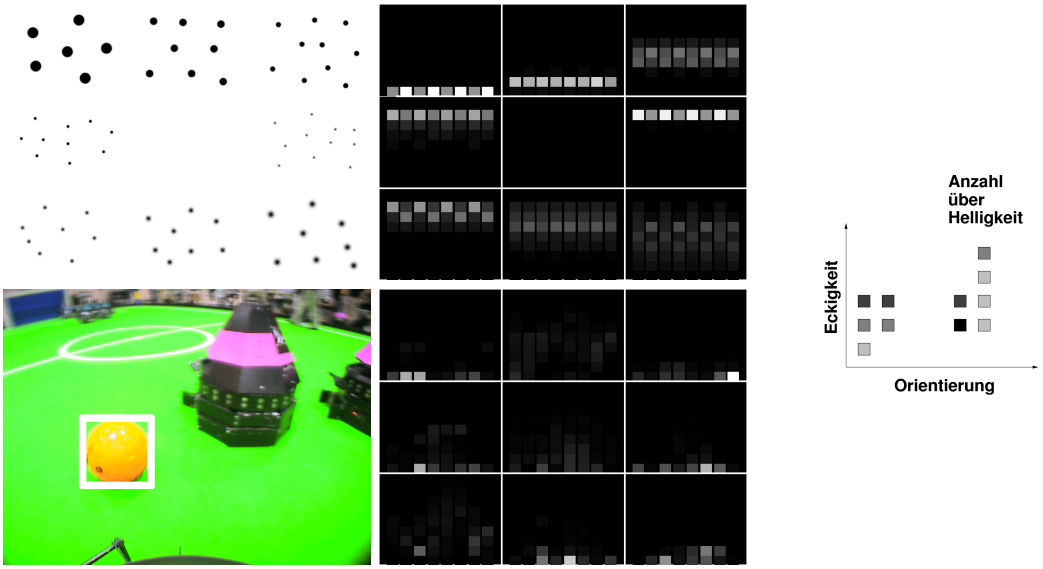


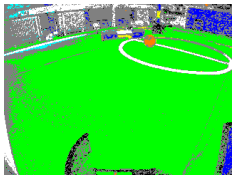
Abbildung 6.3: Kombinierte Orientierungs-/Krümmungshistogramme, extrahiert aus einem beispielhaften Gesamtbild (oben) sowie aus einer tatsächlich gefundenen Aufmerksamkeitshypothese (unten). Die kleine Illustration rechts beschreibt die Bedeutung der Achsen der einzelnen Histogramme. Weitere Erklärungen hierzu entnehmen sie bitte dem Text.

chen verschwommen, erstrecken sich die gefundenen lokalen Krümmungen über einen deutlich größeren Bereich.

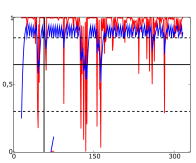
Die reale Aufnahme zeigt im Gegensatz zu den Beispielaufnahmen der vorherigen Abschnitte eine andere Ansicht des Spielfelds; die Aufmerksamkeitsregion enthält dieses mal einen Ball. Man erkennt an den Histogrammen aus den äußeren Bereichen, daß hier vornehmlich kleinere lokale Krümmungen vorherrschen, entsprechend den kleinen y-Werten im Histogramm. Spätestens hier zeigt sich aber, daß die Bewertung solcher Histogramme durch einen Menschen nicht möglich ist. Ein neuronales Netz kann hier aber sehr wohl noch Regelmäßigkeiten ableiten und die Merkmale entsprechend sinnvoll bewerten (siehe Kapitel 9).

6.2 Merkmalsextraktion





Wie man am Beispiel des Histogramms für den Eckpfosten in Abbildung 6.2 bereits sehen konnte, ist ein Nachteil der rechteckigen Aufmerksamkeitsregionen, daß Teile des Hintergrunds ebenfalls in die Merkmalsberechnung mit einfließen. Dadurch werden die Merkmale teilweise sehr stark verfälscht. Insbesondere Regionen der meist kegelartigen Roboter sowie stark geneigte Objekte am linken oder rechten Bildrand enthalten einen großen Anteil eines solchen Hintergrundrauschens. Um diesen Nachteil auszugleichen, wird eine binäre Maske erstellt, die das gesuchte Objekt möglichst vollständig enthält, im Gegenzug aber möglichst wenige Teile des Hintergrunds mit



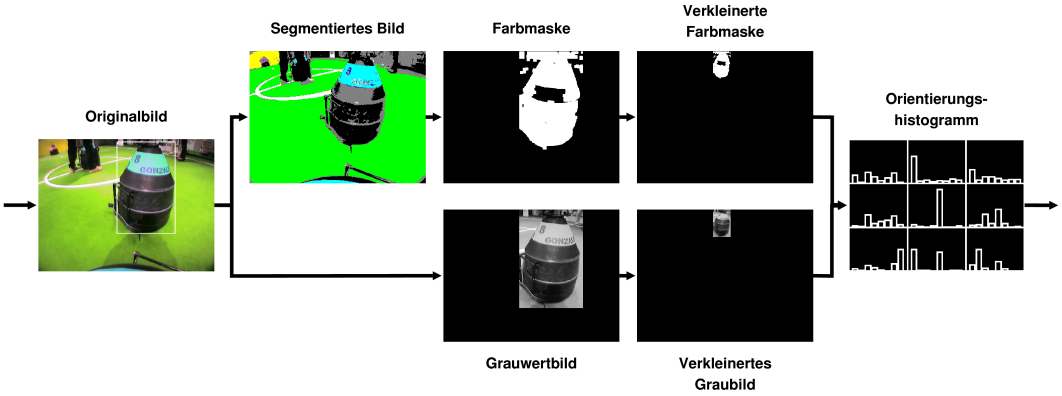
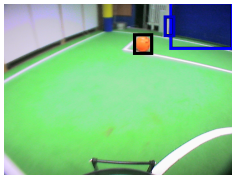


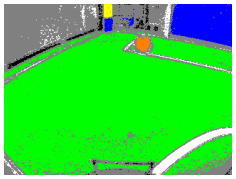
Abbildung 6.4: Vorverarbeitungsschritte, die vor der Merkmalsberechnung ausgeführt werden. Alle gefundenen Objekthypothesen werden auf eine maximale Obergrenze skaliert, um die starken Größenvariationen der Objekte im Bild auszugleichen. Zusätzlich werden die Merkmale nur auf Bildbereichen berechnet, die in einer korrespondierenden binären Maske einen positiven Wert besitzen.

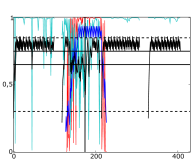
erfasst. Die eigentlichen Merkmalswerte werden dann nur noch an den Stellen der Objekthypothese ermittelt, die einen positiven Eintrag in der korrespondierenden Binärmaske besitzen. Die binäre Maske selbst wird mit Hilfe der Werte des farbindizierten Bildes erstellt. Alle Stellen, an der die gesuchte Farbklasse im Indexbild vorkommt, erhalten einen positiven Wert in der Binärmaske. Im Beispiel des Eckpfostens erzeugen alle Pixel der gelben und blauen Farbklasse positive Werte an den entsprechenden Stellen im Binärbild. Um kleinere Fehler in der Farbindizierung auszugleichen, wird die binäre Maske vor der Benutzung noch großzügig mit Hilfe eines morphologischen Operators dilatiert [71] um vereinzelte Lücken aufzufüllen.

Ein weiteres Problem ist die starke Größenvarianz der zu erkennenden Objekte. Beispiele hierzu sind in Abbildung 1.1 aufgezeigt. Objekte, die eine starke Struktur aufweisen, wie die aus einzelnen Teilflächen bestehende Oberfläche des Balls, ergeben so abhängig vom Aufnahmeabstand deutlich unterschiedliche Merkmale. Um diesen Nachteil auszugleichen, werden alle gefundenen Regionen auf eine maximale Obergrenze skaliert. Die Obergrenze stellt je nach bearbeitetem Objekt einen guten Kompromiß zwischen übermäßigem Detailreichtum und ausreichender Formerkennung dar.

In Abbildung 6.4 ist diese Bearbeitungskette für die Berechnung eines Orientierungshistogramms zur Robotererkennung illustriert. Ausgehend vom Eingangsbild wird das Bild sowohl farbindiziert als auch die gefundene Region in ein Grauwertbild übersetzt. Aus dem Indexbild bzw. dem Unterbereich der gefundenen Region werden alle Pixel der schwarzen, cyan- sowie magentafarbenen Klasse in positive Werte der Binärmaske übertragen. Die anschließende Dilatation der binären Maske schließt kleinere Lücken innerhalb der Maske. Anschließend werden beide Bilder, das Grauwertbild sowie die binäre Maske auf die maximale Obergrenze skaliert (zur Illustration übertrieben dargestellt). Die Berechnung des resultierenden Orientierungshistogramms erfolgt nun nur noch an den Stellen, an der die korrespondierende binäre Maske positive Werte enthält.







6.3 Neuronale Klassifikation

Mit Hilfe der so gewonnenen Merkmalsvektoren können nun die neuronalen Netze entweder trainiert oder zur Klassifikation verwendet werden. Da für jeden Objekttyp eine eigene Regionensuche implementiert ist und die gefundenen Regionen damit bereits eine Hypothese für einen bestimmten Objekttyp darstellen, muß in der anschließenden Klassifikation nur noch eine binäre Entscheidung getroffen werden. Die Klassifikation entscheidet nur noch, ob die jeweilige Region das vermutete Objekt enthält oder nicht.

Zur Klassifikation werden ein oder mehrere Multilayer-Perzeptron-Netzwerke verwendet. Wie in Abschnitt 3.1.3 beschrieben, ergibt sich die Anzahl der Eingabeneuronen aus der Länge der Merkmalsvektors, die Ausgabeschicht wiederum enthält nur ein Neuron, entsprechend der binären Entscheidung die getroffen werden muß. Für den Fall, daß mehrere einzelne neuronale Netze verwendet werden, ist eine abschließende Entscheidung über das Gesamtergebnis mit Hilfe fest eingestellter Schwellenwerte nötig. Alle MLP-Netze enthalten die üblichen sigmoiden Schwellenwertfunktionen und werden mit einem normalen, überwachten „backpropagation“-Lernverfahren trainiert [56, 110]. Da die Untersuchung verschiedener Netzarchitekturen den Rahmen dieser Arbeit überstiegen hätte, wurde auf Tests mit anderen Netzarchitekturen verzichtet. Die in Kapitel 9 aufgezeigten Resultate zeigen aber auch deutlich, daß die verwendeten Multilayer-Perzeptron-Netzwerke (MLP) bereits eine sehr gute Erkennungsleistung bieten.

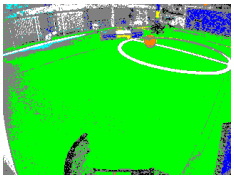
Zum Training der jeweiligen Netze ist eine ausreichende Anzahl an Trainingsmaterial nötig. Dieses Trainingsmaterial wurde aus Bildern gewonnen, die während echter Wettkampfspiele aufgezeichnet wurden. Dies garantiert, daß etwa eine hohe Zahl verschiedenartiger Robotertypen, Objektansichten und unerwarteter Situationen im Trainingsmaterial enthalten waren. Insbesondere die während eines Turniers doch hin und wieder fälschlicherweise erkannten Objekte liefern wertvolle Daten, um eine hohe Spezifität der trainierten neuronalen Netze zu erreichen.

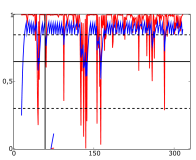
6.4 Modellbasierte Selektionsverfeinerung

Durch einen weiteren modellbasierten Abgleich aller gefundenen Objekte am Ende der Objekterkennungskette lassen sich weitere, wenig rechenaufwendige Validierungsregeln erstellen. So muß etwa ein erkannter Torpfosten direkt an einen Torbereich anschließen. Desweiteren können Gelbübergänge an Kanten, die einem Eckpfosten zugeordnet wurden, ebenfalls ignoriert werden. Ebenso sind Torpfosten des blauen und gelben Tores direkt über- oder untereinander mit hoher Wahrscheinlichkeit Randbereiche eines Eckpfosten an dieser Stelle, selbst wenn der Eckpfosten selbst nicht als solcher erkannt worden ist. Weitere Regeln ergeben sich aus der Position des Roboters auf dem Spielfeld. Bei ausreichender Selbstlokalisation können unter anderem Ballhypothesen außerhalb des Spielfelds ignoriert werden.

In weniger einfachen Fällen, wenn der Roboter beispielsweise einen Eckpfosten der







einen Seite sowie den Torpfosten der anderen Seite erkennt, können auch die ermittelten Erkennungsgenauigkeiten aus Kapitel 9 herangezogen werden. Da die Eckpfosten durch die zusätzliche neuronale Klassifikation deutlich sicherer erkannt werden, kann hier der gefundene Torpfosten mit hoher Sicherheit verworfen werden.

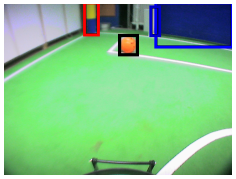
Im realen Einsatz geschieht diese abschließende, modellbasierte Selektionsverfeinerung aber nicht auf den jeweiligen Wahrscheinlichkeitswerten, wie sie direkt aus dem neuronalen Klassifikator gewonnen werden, sondern erst nach der zeitlichen Integration, wie es in den nachfolgenden Kapiteln 7 und 8 beschrieben wird.

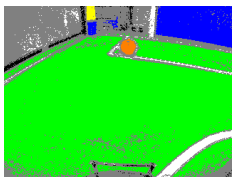
6.5 Verwandte Arbeiten

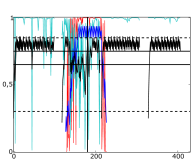
Das Gesamtsystem, wie es bis zu dieser Stelle erklärt wurde (ohne die in den nächsten Kapiteln beschriebene zeitliche Integration), wurde bereits von anderen Gruppen erfolgreich im Robotikbereich angewendet. Kestler et. al. beschreiben in [79, 80] ebenfalls ein System, das auf eine farbbasierte Regionensuche eine Merkmalsextraktion und eine anschließende Klassifikation der gefundenen Merkmale realisiert. Diese und andere Arbeiten [30] sind aber alle in einem recht statischen, büroartigen Umfeld entstanden. Nicht nur die fehlende Dynamik der Umgebung unterscheidet sie von dem hier vorausgesetzten Umfeld, sondern auch eine teilweise stark vereinfachte Aufgabenstellung macht den Unterschied zur hier vorliegenden Arbeit deutlich. So waren etwa bunte Objekte jeweils auf und vor einem neutralen, unstrukturierten Hintergrund platziert. Das macht nicht nur die zeitliche Integration weniger wichtig, sondern auch zusätzliche Vorarbeiten, wie sie in Abschnitt 6.2 dargestellt wurden, unnötig.

6.5.1 RoboCup

Vergleichbare Arbeiten aus dem RoboCup-Umfeld gibt es deutlich weniger. Bereits die Aufteilung in eine mehrschichtige Architektur, das heißt irgendeine Form von Regionensuche, Merkmalsberechnung und Klassifikation zu verwenden, ist kaum verbreitet. Natürlich wird meist nach der sehr einfachen Blobsuche noch eine daran anschließende Verifikation und heuristische Regionenbewertung durchgeführt, aber dies als Klassifikationsschritt zu bezeichnen scheint übertrieben. Deutlich ausgefeiltere Verfahren gibt es beispielsweise von Zagal et. al. [144]. Sie geben die Auflagen für eine positive Verifikation der Regionen nicht von Hand vor, sondern lassen diese mit Hilfe genetischer Algorithmen in einem überwachten Trainingsprozess lernen. Vorgegeben werden die Eigenschaftsarten wie Mindestgröße, Farbklassenzugehörigkeit oder Seitenverhältnisse. Die Parameterwerte, sowie die Gewichtung der einzelnen Parameter, werden dann automatisch ermittelt.







Einen Schritt weiter gehen Wilking und Röfter [141]. Hier werden nicht mehr nur einfache, geometrische Eigenschaften der Regionen verwendet, sondern zusätzlich eine abstrakte Repräsentation der Form der Regionen durch Anzahl der Ecken, eine Konvexitätskriterium, sowie die Anzahl verschiedener Winkelklassen zwischen den

Liniensegmenten[4]. Zur anschließenden Klassifikation wird, mit Hilfe des bekannten C4.5-Algorithmus', ein Entscheidungsbaum angelegt, in dem alle diese Merkmale als Attribute für die Baumaufteilung verwendet werden.

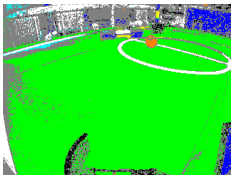
Nicht in einer eigenen Publikation veröffentlicht, aber im Rahmen einer anderen Arbeit [33] wird auch von Ferrein, Hermanns und Lakemeyer angegeben, daß in ihrem System einmal gefundene Regionen anschließend in einem weiteren Schritt validiert werden. Dazu wird für Ball, Tore und die Eckpfosten mittels einer Hough-Transformation der Umriß der Objekte untersucht.

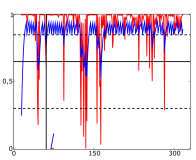
Alle drei oben vorgestellten Arbeiten beziehen sich auf alle in diesen Ligen relevanten Objekte wie Ball, Eckpfosten und Tore. Bei der Arbeit von Wilking und Röfer werden mit diesen Merkmalen sogar die Roboterhunde klassifiziert. Die Arbeit von Mitri et. al. [99] dagegen konzentriert sich auf die Ballerkennung. Auch hier wird eine mehrschichtige Architektur ähnlich der von mir vorgestellten Aufteilung verwendet. Die Regionensuche ist eine Kombination aus top-down- und bottom-up-Vorgehensweisen. Welche Merkmale in welcher Gewichtung dann für die Regionenauswahl verwendet werden, wird mit Hilfe manuell ausgewählter Regionen trainiert, und anschließend automatisch ermittelt. Das Maximum des erreichten Saliencywertes wird dann als Unterregion für die Weiterverarbeitung ausgewählt. Was bei mehreren zu findenden Objekten im Bild passiert, bleibt unklar. Sowieso scheint die Qualität der Aufmerksamkeitssteuerung ein wesentliches Kriterium für die Spezifität der gesamten Verarbeitungskette zu sein. Die anschließende Klassifikation mit Hilfe eines Adaboost-Verfahrens [39, 40], mit dem ein Satz von Regressions- und Klassifikationsbäumen aufgebaut wird, kann beispielsweise nicht zwischen einem Ball und ebenfalls runden Köpfen von Personen unterscheiden. Außerdem ist das Verfahren nach Aussage der Autoren auf heute erhältlichen Rechnern nicht echtzeitfähig. Alleine die sehr aufwendige Regionensuche benötigt etwa 1,5 Sekunden auf einem, dem hier Verwendeten vergleichbar schnellen Rechner.

Einen ähnlichen Ansatz verfolgen Treptow, Masselli und Zell in [128]. Auch hier werden Haar-Wavelets-ähnliche Merkmale [20] mit Hilfe eines Adaboost-Verfahrens gelernt. Der Unterschied zur vorherigen Arbeit besteht darin, daß hier keine separate Regionensuche zur Auswahlsreduktion verwendet wird, sondern das Verfahren mit einem Kondensationsalgorithmus kombiniert wird. Leider ist die Leistungsfähigkeit des Gesamtverfahrens nur sehr schwer zu beurteilen. Auf der einen Seite wird eine korrekte Klassifikation von 97% angegeben, gleichzeitig werden aber ganze Sequenzen abgebildet, auf denen der Ball nicht korrekt gefunden wird. Darüber hinaus ist das Verfahren nur auf Bälle mit einem maximalen Abstand von 2 Metern anwendbar. Zusätzlich ist der Zustand des System undefiniert, wenn überhaupt kein Ball im Bild zu sehen ist.

Weitere Arbeiten, die völlig ohne vorherige Regionensuche auskommen, gibt es von Hanek et. al. [53, 54]. Sie verwenden einen sogenannten „Contracting Curve Density"-Algorithmus, um einen Polygonzug um das gesuchte Objekte zu legen. In einem ite-







[4] Genauere Angaben zu den verwendeten Merkmalen sowie zum Aufbau des Entscheidungsbaums lassen sich aus der Arbeit leider nicht entnehmen.

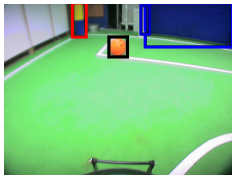
rativen Vorgang wird eine parametrisierte Kurve ähnlich einem EM-Algorithmus anhand lokaler Bildstatistiken an die Kontur des gesuchten Objekts angepasst. Der Algorithmus benötigt dazu nur eine ungefähre Startposition für die Suche, was im Umkehrschluß aber auch bedeutet, daß keine globale Suche nach Objekten möglich ist. Weitere Annahmen über die Farbe des Objekts, dessen Oberflächenstruktur oder den Bildhintergrund sind nicht nötig. Unklar bleibt aber, wie das Verfahren mit Objekten klarkommt, die verschiedene Ansichten aus verschiedenen Blickwinkeln besitzen. Durch die Verwendung ausgewählter Punkte anstatt aller Konturpixel wurde das Verfahren in einer späteren Arbeit [55] auch auf Echtzeitfähigkeit optimiert.

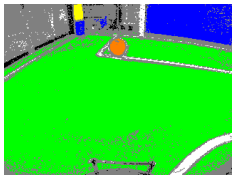
6.5.2 Automobilbereich

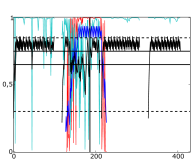
Im Automobilbereich wurden bereits deutlich mehr für dieses Kapitel relevante Arbeiten vorgestellt. Vom Schwierigkeitsgrad liegt die Objekterkennung im RoboCup in etwa zwischen der Erkennung von Verkehrsschildern und der von Fußgängern. Verkehrsschilder sind alle bereits vorab bekannt[5]. Außerdem haben sie keine wirkliche dreidimensionale Struktur. Die Schildansicht wird aus verschiedenen Blickwinkeln nur perspektivisch verzerrt, es werden aber keine neuen Ansichten des Objekts sichtbar. Fußgänger wiederum sind zwar im Prinzip bekannt, kommen aber in unterschiedlichen Ausprägungen vor. Dies entspricht in etwa den verschiedenen Modellen der Roboter im RoboCup. Zusätzlich sind Fußgänger aber auch über die Zeit veränderlich, was die Erkennung weiter erschwert. Diese meist rhythmische Bewegung läßt sich aber auch zur Erkennung ausnutzen, sie ist sogar alleiniges Thema einer Arbeit [43]. Zusätzlich bedingen wechselnde Beleuchtungsbedingungen, bis hin zur vollständigen Dunkelheit, aber teilweise die Verwendung anderer Sensoren, andere Verfahren zur Regionensuche oder die Verwendung anderer Merkmale zur Klassifikation.

Aus der Menge unterschiedlichster Ansatzpunkte sollen hier nur einige direkt vergleichbare, ansichtenbasierte Methoden beschrieben werden. Die wohl größte Gruppe dieser Verfahren basiert im Prinzip auf einem Pattern Matching gefundener Regionen mit bekannten Ansichten der verschiedenen Objekte. Piccioli et. al. realisierten dazu einen mehrstufigen Prozess ähnlich dem in dieser Arbeit vorgestellten [105]. Ausgehend von einem festen Satz von 60 kreisrunden und 47 dreieckigen Schildern wird eine normalisierte Kreuzkorrelation als Distanzmaß zwischen diesen Schildaufnahmen und den gefundenen Regionen berechnet und anschließend über eine feste Schwelle die wahrscheinlichsten Treffer ausgewählt.

Ein ähnlicher Ansatz für die Verkehrsschilderkennung ist von Franke, Gavrila et. al. in [37] beschrieben. Anstatt aber den Vergleich direkt auf den Originalbildern auszuführen, werden hier gespeicherte, binäre Umrißansichten mit distanz-transformierten Bildansichten verglichen. Derselbe Ansatz wird nun auch auf die Fußgängererkennung übertragen. Durch die hohe Variabilität der Ansichten von Fußgängern kann







[5] Verkehrsschilder sind in diesem Zusammenhang immer fest definierte Schilder wie Stopschilder oder Tempobeschränkungen. Mit ist keine Arbeit bekannt, in der auch Wegweiser oder zusätzliche, einschränkende Schilder, wie sie manchmal unterhalb von Tempolimitierungen angebracht sind, erkannt werden würden.

aber nicht mehr mit einem kleinen Satz von vorgegebenen Umrissen der Passanten gearbeitet werden. Stattdessen werden Hierarchien von solchen Umrissen automatisch aus einem Satz Trainingsmaterial bestimmt und aufgebaut [44, 45]. In der obigen Arbeit ebenfalls noch vorgestellt (siehe auch [143]) ist ein spatiotemporaler Ansatz zur Fußgängererkennung, der mit Hilfe von sogenannten time-delay-neural-networks (TDNNs) das Aussehen sowie die Bewegung der Beine gleichzeitig erkennt. Dazu werden mehrere aufeinanderfolgende Ansichten der Beine direkt dem neuronalen Netz übergeben. Durch eine nicht vollständige Konnektivität der verschiedenen Schichten des neuronalen Netzes werden rezeptive Felder ähnlich denen des menschlichen Auges nachgebildet.

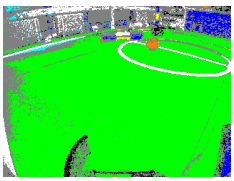
Andere Ansätze zur Klassifikation mit Hilfe neuronaler Netze sind in der Arbeit von Zhao und Thorpe [145] zu finden. Analog zur hier vorgestellten Arbeit erfolgt die Klassifikation mit Hilfe mehrschichtiger MLP-Netzwerke auf Gradientenbildern. Dort werden aber nicht wie in dieser Arbeit abstrakte Repräsentationen in Form eines Orientierungshistogramms verwendet, sondern direkt das ganze, auf eine feste Größe skalierte, Gradientenbild. Auch hier ist das Netzwerk als binärer Klassifikator ausgelegt. Ein Ausgabeneuron gibt einen Wahrscheinlichkeitswert für das Auftreten eines Passanten, die Entscheidung für oder wider einer Erkennung wird dann über einen festen Schwellenwert geregelt.

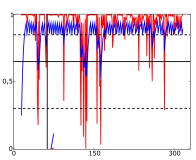
Auch Verfahren ähnlich den im obigen Abschnitt von Treptow oder Mitri Beschriebenen finden Anwendung im Bereich der intelligenten Transportsysteme. Papageorgiou, Evgeniou und Poggio [104] verwenden ebenfalls Haar-Wavelets zur Extraktion simpler Kanten- und Eckenbeschreibungen. Im Gegensatz zu den erwähnten Arbeiten werden hier aber Merkmale ausgewählter Regionenpositionen mit Hilfe von Support-Vektor-Netzwerken klassifiziert. Zum Zeitpunkt der Veröffentlichung dieser Arbeit 1998 war aber noch ein erheblicher Aufwand nötig, um dieses Verfahren annähernd echtzeitfähig zu machen. Im Bereich der autonomen Fahrzeuge gibt es noch eine Unmenge anderer Arbeiten mit den verschiedensten Lösungsansätzen; der interessierte Leser soll an dieser Stelle auf die sehr umfangreichen Übersichtsarbeiten von Lombardi [89] oder Bertozzi et. al. [9] verwiesen werden.

6.6 Kernpunkte

Die neuronale Klassifikation ermöglicht eine stabile Entscheidungsfindung, ob die gefundene Objekthypothese tatsächlich den gesuchten Objekttyp enthält oder nicht. Ein Vorteil der neuronalen Bewertung ist die allgemein sehr gute Generalisierungsleistung solcher Netze, das heißt, das neuronale Netz kann auch Ansichten eines Objekts in der Ausführungsphase korrekt bewerten, selbst wenn diese Ansichten nicht im Trainingsmaterial enthalten waren. Dieses Verhalten gegenüber unbekannten Objektansichten kann in der Anwendung aber auch zu Fehlklassifikationen führen. Deswegen ist es besonders wichtig, daß im Trainingsmaterial genügend negative Beispiele vorhanden sind, also Bilder, die explizit nicht das gesuchte Objekt enthalten. Dadurch, daß für das Training Bilder aus realen Spielen vergangener Wettkämpfe verwendet wurden,



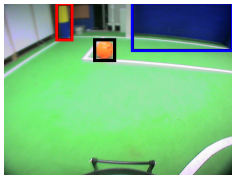


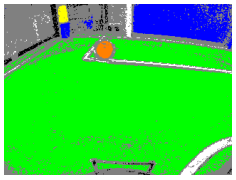


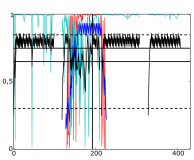
konnte dies in großen Bereichen sichergestellt werden (siehe Kapitel 9).

Der wichtigste Beitrag in diesem Kapitel ist die Anwendung solcher neuronaler Klassifikatoren innerhalb einer hochdynamischen Umgebung wie dem RoboCup [78, 92]. Wie bereits im vorherigen Kapitel beschrieben, entspricht die komplette Objekterkennung für einen Großteil der am RoboCup teilnehmenden Mannschaften in etwa der oben beschriebenen Suche nach Objekthypothesen. Wie in der Diskussion über verwandte Arbeiten zu sehen war, ist eine solch hochentwickelte Objekterkennung im RoboCup noch immer die Ausnahme.

Eine andere Neuerung ist die Verwendung kombinierter Orientierungs- und Krümmungshistogramme. Da aufgrund der Echtzeitbedingungen des Gesamtsystems nur wenige Merkmale berechnet werden können, müssen diese sehr aussagekräftig sein. Umgekehrt müssen diese Merkmalsberechnungen aufgrund der sehr einfach strukturierten Objekte auch besonders die wesentlichen geometrischen Strukturen wiedergeben. Die Kombination von Orientierungs- und Krümmungsinformationen erlauben genau dies ohne großen Berechnungsaufwand.







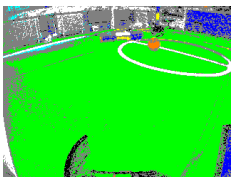
7 Objektverfolgung und -prädiktion

Die Aufgabe einer Objektverfolgung ist es, sich bewegende Objekte über die Zeit zu beobachten und aus diesen gewonnenen Informationen, den sogenannten Tracks, die weitere Bewegung des Objekts vorhersagen zu können [14]. Die Objektverfolgung muß daher ein kinematisches Modell des zu verfolgenden Gegenstandes aus der zeitlichen Abfolge der beobachteten Positionen ermitteln. Zu solch einem kinematischen Modell gehört nicht nur die momentane Position eines Objekts, sondern insbesondere auch seine Bewegungsgeschwindigkeit und -richtung. Diese Werte sind für die Vorhersage der wahrscheinlichsten zukünftigen Positionen wichtig. Durch Ungenauigkeiten in der Erkennung des Gegenstandes, der ungenauen Ermittlung der eigenen Bewegung und anderer, schwer zu modellierender Einflußparameter scheidet eine einfache Interpolation über bisherige Objektpositionen als Vorhersagemodell aus.

Sollen mehrere Objekte gleichzeitig beobachtet und verfolgt werden, wird ein sogenanntes *multi-target tracking* verwendet [5]. Eine Möglichkeit ist es, ein Verfahren zu verwenden, das von sich aus mehrere Positionshypothesen verwaltet, wie es zum Beispiel eine partikelbasierte Monte-Carlo-Methode kann, die oft in der Selbstlokalisation (auch von uns, wie in [27, 138] beschrieben) verwendet wird. Ein solches Verfahren ist aber sehr rechenintensiv. Da die Objektverfolgung direkt in die Bildverarbeitung integriert werden soll, muß ein Verfahren verwendet werden, welches wesentlich weniger Rechenleistung benötigt. Im vorliegenden Fall wird jede Instanz eines Objekts durch einen eigenen, Kalman-basierten Objektverfolgungsfilter beobachtet. Ein separates Modul muß daher die momentan gesehenen Objekte den jeweiligen Filterinstanzen zuordnen.

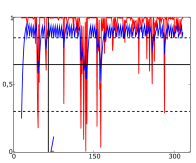
In den folgenden Abschnitten werden die einzelnen Teile dieses Vorgehens näher beschrieben. Der nächste Abschnitt illustriert den Ablauf des gesamten Systems, wie er bereits im Abschnitt 3.2 illustriert wurde. Abschnitt 7.2 erklärt dann den verwendeten Kalman-Filter sowie die verschiedenen implementierten Filtermodelle im Detail. In Abschnitt 7.3 wird die Zuordnung von Objekthypothesen zu existierenden Objektbahnen, sogenannten Tracks, erläutert. Abschließend folgt, wie in allen anderen Kapiteln, eine Übersicht über existierende Arbeiten in diesem Gebiet, gefolgt von einer Zusammenfassung und Bewertung der besprochenen Verfahren.





7.1 Ablaufschema

Das Tracking wird in einem allozentrischen Koordinatensystem durchgeführt. Dadurch läßt sich die Eigenbewegung des Roboters sowie seine Kameradrehung leicht in jedem Aktualisierungsschritt mit einbeziehen. Um Unstetigkeiten in der Eigenpositionsermittlung zu vermeiden, wird eine unkorrigierte globale Position des Ro-



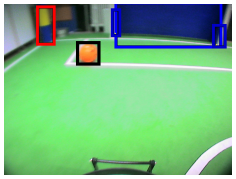
boters verwendet. Die Position wird laufend anhand der Radumdrehungsmessungen, der sogenannten Odometrie, ermittelt. Eine globale Neupositionierung anhand von Lokalisationsmerkmalen, wie gesehenen Eckfahnen oder Torpfosten, wird nicht vorgenommen. Die so ermittelte Position unterscheidet sich im Laufe der Zeit durch Fehler in der Odometriemessung zwar erheblich von der tatsächlichen Roboterposition, der Fehler der relativen Verschiebung zwischen einigen aufeinanderfolgenden Bildern ist aber vernachlässigbar gering.

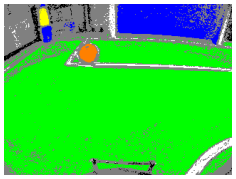
Die einzelnen Schritte für einen vollständigen Durchlauf des Objektverfolgungsmoduls sind wie folgt:

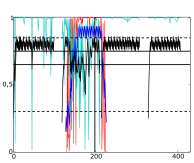
Ablaufschema Tracking
1 Daten in globales Koordinatensystem konvertieren
2 Zustandsvorhersage der existierenden Tracks
3 Datenzuordnung der Beobachtungen zu den Tracks
4 Zustandskorrektur der existierenden Tracks
5 Rückkonvertierung der Daten ins Bildkoordinatensystem

Um die im aktuellen Bild gefundenen Objekte mit den existieren Tracks, der Liste von bisher berücksichtigten Bewegungsabläufen, vergleichen zu können, müssen die Objekthypothesen aus dem Bildkoordinatensystem in das verwendete, globale Koordinatensystem übertragen werden. Im realen Einsatz geschieht das über eine einfache Abbildungstabelle, welche die Bildpositionen auf Positionen relativ zum Robotermittelpunkt transformiert. Die Berechnung dieser Abbildungstabelle geschieht mit Hilfe des Verfahrens von Tsai [129, 130], welches hierzu ein Modell der Kamera, der Position der Kamera auf dem Roboter sowie der verwendeten Kameralinse verwendet. Da sich mit einer einzelnen Kamera nur mit Hilfe aufwendiger Verfahren die tatsächliche 3-dimensionale Position schätzen läßt, wird hier davon ausgegangen, daß sich das gesuchte Objekt direkt auf dem Boden befindet. Unter Berücksichtigung der Roboterposition und der Ausrichtung der Kamera zum Aufnahmezeitpunkt läßt sich so die Position des gesehenen Objekts im globalen Weltmodell ermitteln.

Im nächsten Schritt muß die wahrscheinlichste Position zum aktuellen Zeitpunkt für jede der bisher verfolgten Messungen ermittelt werden. Genau hierzu sind die letzten bekannten Positionen sowie das gefundene kinematische Modell nötig. Die Tracks werden um genau die Zeitspanne in die Zukunft projiziert, die zwischen der letzten und der aktuellen Aufnahme vergangen ist.







Nun können im dritten Schritt die neu gefundenen Objektpositionen mit den korrigierten, nun aktuellen Positionen der Tracks verglichen werden. Diese Positionsvergleiche entscheiden, welche aktuelle Observation als zu welchem Track zugehörig bestimmt wird (mehr dazu in Abschnitt 7.3). Vereinfacht wird diese Zuordnung dadurch, daß für alle gefundenen Positionen ebenso wie für die Tracks der Objekttyp bekannt ist, den sie repräsentieren. Eine Positionshypothese für einen Ball kann damit nur einem Track zugeordnet werden, der ebenfalls einer Ballbewegung entspricht. Kann an

dieser Stelle keine aktuelle Beobachtung zugewiesen werden, wird ein neuer Track begonnen, der mit der aktuellen Position initialisiert wird. Im Gegensatz dazu erhalten eventuell nicht alle existierenden Tracks eine aktuelle Beobachtung zugewiesen. Diese Tracks werden trotzdem noch einen gewissen Zeitraum berücksichtigt, um kurzzeitige Fehler in der Regionensuche oder Kameraschwenks weg vom Objekt zu überbrücken. Tracks, denen über einen längeren Zeitraum keine Beobachtung zugeordnet wurde, werden beendet. Da nun durch diese Zuordnung für jeden Track die aktuelle Position bekannt ist, können die bisherige Position und der bisherige Zustand mit Hilfe dieser neuen Position korrigiert werden. Dieser Schritt garantiert eine erneut möglichst korrekte Vorhersage der Trackpositionen für das folgende aufgenommene Bild.

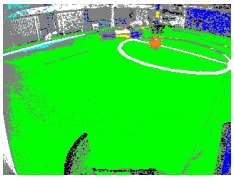
Im letzten, abschließenden Schritt werden nun die so ermittelten, neuen Trackpositionen wieder in Bildkoordinaten zurücktransformiert. Wie in Abschnitt 8.1 zu sehen, kann diese Position verwendet werden, um alternative Berechnungspfade in der Objekterkennung zu aktivieren. Wenn also ein Objekt an einer bestimmten Stelle vorhergesagt wird, dieses Objekt von der Regionensuche aber nicht detektiert wird, dann können für diesen Bereich im Bild weitere Bildbearbeitungsschritte durchgeführt werden. Dies kann unter Umständen weitere wichtige Informationen für die zeitliche Integration der Sicherheitswerte liefern.

7.2 Kalman-Filter

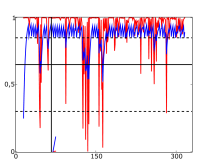
Entsprechend der Echtzeitanforderung an alle verwendeten Verfahren bietet sich für das Trackingmodul hier ein Kalman-Filter-basiertes System an. Kalman-Filter sind stochastische Zustandsschätzer für dynamische Systeme, die aufgrund ihrer iterativen Struktur für den Einsatz in Echzeitanwendungen prädestiniert sind. Die Schätzung und Vorhersage des Zustandes des zu verfolgenden Objekts geschieht allein mit Hilfe der aktuellen Messung und des bisherigen Zustands. Alle anderen, älteren Zustände werden nicht mehr benötigt.

Aufgrund einer Unmenge an existierenden Arbeiten zum Thema Kalman-Filterung soll hier keine umfassende Erklärung dieses Verfahrens gegeben werden. Stattdessen werden insbesondere die einzelnen Entscheidungen für oder wider einer bestimmten Implementation erläutert. Weit detailliertere mathematische und theoretisch fundierte Erklärungen finden sich zum einen natürlich in den Originalarbeiten von Kalman [74] zum anderen in einer ganzen Reihe von Büchern wie zum Beispiel von Bar-Shalom und Fortman [5] oder Blackman und Popoli [14]. Bei der verwendeten Notation orientiere ich mich an der Arbeit von Welch und Bishop [140].





Die Arbeitsweise eines solchen Filters ist derart, daß er das Rauschen von einem gemessenen Signal entfernt und dabei die eigentliche, nützliche Information erhält. Im idealen Fall erhält man damit ein Modul, welches eine genauestmögliche Vorhersage des tatsächlichen Systemzustands liefert, ohne diesen tatsächlich messen zu können. Man erhält damit einen mittleren Wert für den geschätzten Systemzustand, welcher identisch zum mittleren Wert des tatsächlichen Zustandes ist. Außerdem variiert der geschätzte Zustand so wenig wie möglich vom tatsächlichen Zustand.



Um die Echtzeitfähigkeit des Filters noch weiter zu erhöhen, wird in dieser Arbeit eines der einfachsten Modelle, der sogenannte lineare, zeitdiskrete Kalman-Filter verwendet. Ein solcher Filter schätzt den Zustand eines linearen Systems. Das hier vorgestellte Szenario ist jedoch kein lineares System. Da ein solcher Filter aber bereits in den verschiedensten Umgebungen erfolgreich genau dafür eingesetzt wurde [42, 103], wird auch hier dieser berechnungstechnisch sehr einfache Ansatz verwendet.

7.2.1 Zustandsvorhersage in linearen Systemen

Ein lineares System ist ein System, welches sich durch folgende Formel beschreiben läßt:

$$x_k = Ax_{k-1} + Bu_k + Gw_{k-1} \tag{7.1}$$

Dabei ist x_k der gesuchte Zustand bzw. der Zustandsvektor zum aktuellen Zeitpunkt, x_{k-1} der bisherige Zustand zum letzten Messzeitpunkt. Der Vektor u_k ist ein gegebener Einfluß auf das System, zum Beispiel Steuerkommandos auf das zu verfolgende Objekt. In unserem speziellen Szenario ist solch ein Einfluß nicht bekannt, ist darum nur der Vollständigkeit halber hier angegeben. A, B und G sind jeweils Matrizen, um die Werte zu koppeln und w_{k-1} ist das sogenannte Prozessrauschen, ein Einfluß auf das System, welcher sich nur grob in Form einer Zufallsvariablen beschreiben läßt.

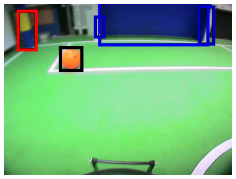
Der n-dimensionale Vektor x enthält alle Information über den aktuellen Zustand des Systems. Dieses x läßt sich nun aber eben nicht direkt messen. Was gemessen wird, ist ein m-dimensionaler Vektor z:

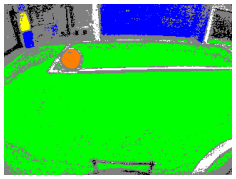
$$z_k = Hx_k + v_k \tag{7.2}$$

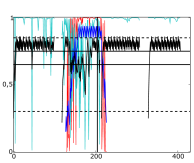
Dieses z_k ist wiederum vom aktuellen Zustand x_k (gekoppelt über eine weitere Matrix H) und einem weiteren Rauscheinfluß v_k (dem sogenannten Messrauschen) abhängig. Beide Zufallsgrößen w_k und v_k sind voneinander unabhängig, weiß und normalverteilt:

$$p(w) \sim \mathcal{N}(0, Q) \tag{7.3}$$
$$p(v) \sim \mathcal{N}(0, R) \tag{7.4}$$

Beide Kovarianzen, das Prozessrauschen Q sowie das Messrauschen R, sind in unserem Fall als konstant angenommen und können entsprechend über konstante Matrizen für jedes Filtermodell konfiguriert werden.







Nehmen wir nun an, $\hat{x}_k^-$ beschreibt die sogenannte a-priori-Zustandsvorhersage zum Zeitpunkt k, bei der nur alles Wissen vor dem aktuellen Zeitpunkt k genutzt wird. Im Gegensatz dazu ist $\hat{x}_k$ die a-posteriori-Zustandsvorhersage zum aktuellen Zeitpunkt *inklusive* der aktuellen Messung z_k. Damit können wir nun zwei Fehlermaße definieren, die den Abstand vom tatsächlichen Systemzustand beschreiben:

$$e_k^- \equiv x_k - \hat{x}_k^- \tag{7.5}$$
$$e_k \equiv x_k - \hat{x}_k \tag{7.6}$$

Die jeweiligen Kovarianzen dieser beiden Vorhersagefehler sind dann entsprechend:

$$P_k^- = E[e_k^- e_k^{-T}] \tag{7.7}$$
$$P_k = E[e_k e_k^T] \tag{7.8}$$

7.2.2 Kalman-Filter und Prädiktor-Korrektor-Struktur

Grundgedanke bei einem Kalman-Filter ist nun, eine a-posteriori-Zustandsvorhersage $\hat{x}_k$ als eine Linearkombination von einer a-posteriori-Vorhersage $\hat{x}_k^-$ und einer gewichteten Differenz zwischen der eigentlichen, aktuellen Messung z_k und der Messvorhersage $H\hat{x}_k^-$ wie folgt zu berechnen:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{7.9}$$

Diese gewichtete Differenz $(z_k - H\hat{x}_k^-)$ wird *residual* sowie *innovation* genannt. Sie beschreibt die Abweichung der Messvorhersage von der tatsächlichen Messung. Die noch verbleibende Matrix K ist der sogenannte *Kalman-Gain*, welcher die a-posteriori-Fehlerkovarianz minimiert:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{7.10}$$

Betrachtet man sich diesen *Kalman-Gain* näher, so sieht man folgendes: wenn das Messrauschen groß ist, wird K klein sein, und die aktuelle Messung wird bei der Berechnung des Systemzustands nicht sehr stark berücksichtigt. Andersherum wird bei kleinem Messrauschen das K größer und damit das Vertrauen in den aktuellen Messwert deutlich stärker.

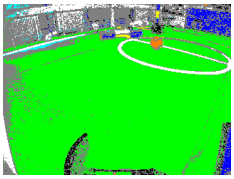
Nimmt man nun alle diese bisher vorgestellten Formeln zusammen, ergibt sich die eigentliche, sogenannte Prädiktor-Korrektor-Struktur des Kalman-Filters. Diese Struktur entspricht genau den Punkten (2) und (4) aus dem obigen Ablaufschema in Abschnitt 7.1, Seite 66. Der eigentliche Kalman-Filter, wie er hier verwendet wurde, besteht somit aus den Blöcken Zustandsvorhersage:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{7.11}$$
$$P_k^- = AP_{k-1}A^T + GQG^T \tag{7.12}$$

sowie Zustandskorrektur:

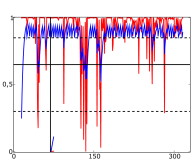
$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{7.13}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{7.14}$$
$$P_k = (I - K_k H)P_k^- \tag{7.15}$$





7.2.3 Systemmodelle

Was nun noch fehlt, sind die verschiedenen Systemmodelle, das heißt die unterschiedlichen Bewegungsmodelle der verschiedenen zu verfolgenden Objekte. In dieser Arbeit



werden hierzu zwei verschiedene Modelle verwendet, einmal für die statischen Objekte wie Eckfahnen und Torpfosten und die Tore selbst, sowie ein anderes Modell für alle bewegten Objekte wie Ball und Roboter. Auf den ersten Blick wirkt die Verwendung eines Filters für unbewegte Objekte vielleicht etwas übertrieben. Durch die Fähigkeit eines Kalman-Filters, das eigentliche Nutzsignal von seinem Erkennungsrauschen zu befreien, wird aber auch hier die Erkennungsleistung über die Zeit wesentlich verbessert.

Der andere Punkt ist, daß sich die Bewegung von Ball und Robotern deutlich von einer linearen Bewegung, wie sie Grundannahme für einen Kalman-Filter ist, abweicht. Bälle können von Wänden abprallen, Roboter spontan ihre Bewegungsrichtung ändern. Kalman-Filter wurden aber, wie oben erwähnt, oft erfolgreich in solchen Szenarien eingesetzt. Auch die Auswertungen in Kapitel 9.4 werden zeigen, daß diese Verletzung der Grundannahme keinen großen negativen Effekt auf die Filterleistung hat. Grund ist, daß sich durch die hohe Updaterate des Filters mit voller Bildwiederholrate die relative Bewegung des Objekts in diesem kurzen Zeitraum nur unwesentlich von einer linearen Bewegung unterscheidet. Die verbleibenden kleinen Abweichungen werden über das Systemrauschen modelliert. So lassen sich nun auch die höherdimensionalen Werte wie etwa die Beschleunigung in den Kalman-Filter einbringen. Das ganze Verfahren nennt sich *discrete white noise acceleration* und ist an verschiedenen Stellen in der Literatur [6] beschrieben.

Statische Objekte

Für alle statischen Objekte ist der Systemzustand einfach über die 2-dimensionale Position (x_x, x_y) der Objekte auf dem Spielfeld definiert.

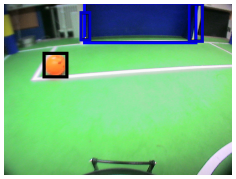
$$x = [x_x \; x_y]^T \tag{7.16}$$

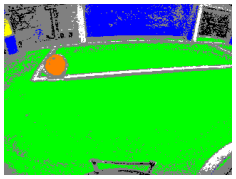
Damit entsprechen die Kopplungsmatrizen aus Formel 7.1 bzw. 7.11 einfach der Identitätsmatrix I. Die Kovarianz Q des Systemrauschens aus Formel 7.12 ist damit:

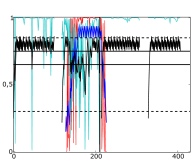
$$\begin{aligned} Q =& IE[w_{k-1} {w_{k-1}}^T] I^T \\ \hat{=}& \begin{bmatrix} q_s & 0 \\ 0 & q_s \end{bmatrix} \end{aligned} \tag{7.17}$$

wobei q_s für statische Objekte sehr klein ist. Ein Systemrauschen ist in diesem Fall zwar nicht physikalisch sinnvoll, aber es gilt, daß ein zusätzlicher Rauschanteil einen stabilisierenden Effekt auf den Kalman-Filter hat. Analog dazu ist die Matrix H aus Formel 7.2 ebenfalls die Identitätsmatrix, was wiederum die folgende Kovarianz R für das Messrauschen ergibt:

$$\begin{aligned} R =& IE[v_k v_k^T] I^T \\ \hat{=}& \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix} \end{aligned} \tag{7.18}$$







Da für beide Systemmodelle, egal ob im statischen oder bewegten Fall, keine Bewegungseingabe bekannt ist oder keine Steuerungseingabe erfolgt, entspricht der komplette Term Bu_k aus Formel 7.1 $Bu_k \hat{=} 0$.

Bewegte Objekte

In diesem Fall ist der Systemzustand wiederum über die 2-dimensionale Position des Objekts auf dem Spielfeld (x_x, x_y) und zusätzlich über seine Geschwindigkeit $(\dot{x}_x, \dot{x}_y)$, gemessen im selben Koordinatensystem, definiert:

$$x = [x_x \ \dot{x}_x \ x_y \ \dot{x}_y]^T \tag{7.19}$$

Die beiden Matrizen A und G aus Formel 7.1 sind in diesem Fall:

$$A = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix} \tag{7.20}$$

T ist dabei der Zeitabstand zwischen zwei aufeinanderfolgenden Messungen. Die Matrizen für das Systemrauschen mit den Elementen q_b auf der Diagonalen für bewegte Objekte sowie für das Messrauschen sind quasi identisch wie im obigen Fall. Nur werden hier die Elemente q_b gemäß der maximalen angenommenen Beschleunigung auf einen Wert von etwa 2 $\mathrm{m/s^2}$ gesetzt. Außerdem ist in diesem Fall

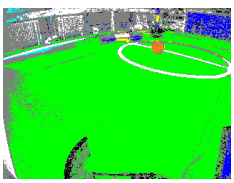
$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{7.21}$$

7.3 Multi-Target-Tracking

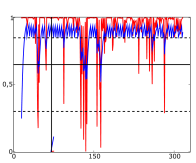
Der verbleibende Punkt des Ablaufschemas aus Abschnitt 7.1, der noch nicht erklärt wurde, ist der dritte Punkt „Datenzuordnung von Beobachtungen zu Tracks“. Welche Messung wird also für die Zustandskorrektur des Kalman-Filters verwendet. Entsprechend unserer Vorgabe eines echtzeitfähigen Algorithmus' wird auch hier ein berechnungstechnisch simples Verfahren verwendet, auch wenn die Zuordnungsleistung damit eventuell nicht optimal ist. Wiederum gilt aber auch hier, daß die Ergebnisse in Kapitel 9.4 die Wahl dieses Verfahrens unterstützen.



Bei der Zuordnung von neuen Messungen zu bestehenden Tracks kann bereits vor Beginn eine deutliche Reduktion der Kandidaten erreicht werden. Dadurch, daß für jeden Objekttyp ein eigenes Modul zur Regionensuche existiert, haben auch gefundene Regionen bereits einen Objekttyp assoziiert. So kann sichergestellt werden, daß nur Objekte des passenden Typs überhaupt für eine Datenzuordnung verwendet werden.



Anschließend wird um jede vorhergesagte Position eines existierenden Tracks $\hat{x}_k^-$ eine elliptische Region definiert. Nur Beobachtungen innerhalb dieser Region sind



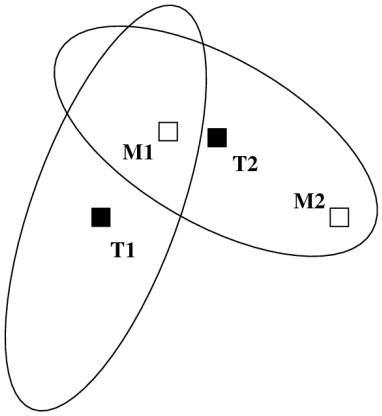


Abbildung 7.1: Illustration, wie Messdaten den Tracks zugeordnet werden. Die beiden Tracks T1 und T2 (die schwarzen Quadrate) stellen die Mittelpunkte der beiden Ellipsen dar. Die Messpunkte sind als weiße Quadrate M1 und M2 dargestellt.

Kandidaten für eine nachfolgende Datenzuordnung. Hierzu wird die gewichtete Norm der Innovation, auch Mahalanobis-Distanz genannt, verwendet:

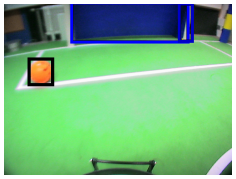
$$d^2 = \tilde{z}_k^T S_k^{-1} \tilde{z}_k \tag{7.22}$$

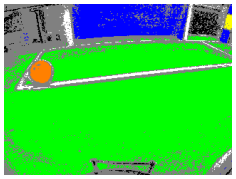
$\tilde{z}_k$ bezeichnet hier die aus Formel 7.9 bekannte Innovation $(z_k - H\hat{x}_k^-)$. Die Kovarianz dieser Innovation $\tilde{z}_k$ ist das verwendete $S_k = E[\tilde{z}_k \tilde{z}_k^T]$. Damit ist die um die Position einer Vorhersage definierte Region abhängig von der Genauigkeit der Messvorhersage.

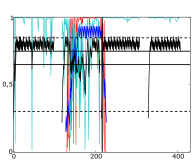
Die eigentliche Datenzuordnung ist nun wie folgt: Nacheinander wird für jeden existierenden Track berechnet, welche Messungen innerhalb dieser Ellipse liegen. Wichtig dabei ist, daß die Tracks hierfür zeitlich sortiert sind. Zuerst wird der Track berücksichtigt, der am längsten existiert. Die Annahme ist, daß ältere Tracks sehr oft bestätigt wurden und daher deutlich sicherer eine tatsächliche Bewegungsspur darstellen.

Von all diesen möglichen Messungen wird nun diejenige zugeordnet, welche die geringste Distanz zur Positionsvorhersage besitzt. Danach wird diese Messung sowie dieser Track aus der Liste der Kandidaten gestrichen. Dadurch wird sichergestellt, daß jede Messung nur einem Track zugeordnet wird, und umgekehrt, jeder Track nur eine Messung zugeordnet bekommt. Erst dann wird mit dem nächsten Track in der Liste fortgefahren.

Wie oben bereits angesprochen, kann dieses Verfahren zu Ergebnissen führen, die nicht global optimal sind. Für den Fall, daß sich die Ellipsen mehrerer Tracks überlappen, kann es sein, daß ein Track nicht den für ihn eigentlich nächstliegenden Messpunkt erhält. Abbildung 7.1 zeigt dies an einem einfachen Beispiel. Die beiden Tracks T1 und T2 (die schwarzen Quadrate) stellen die Mittelpunkte der beiden Ellipsen dar. Die Messpunkte sind als weiße Quadrate M1 und M2 dargestellt. Ist nun der Track T1 älter als T2, so wird ihm der Messpunkt M1 zugeordnet. T2 verbleibt damit nur noch die Messung M2, obwohl die Messung M1 einen deutlich kleineren Abstand hätte. Hier wird das Alter der Tracks aber höher gewichtet.







Sind alle Tracks durchgerechnet, kann es vorkommen, daß sowohl Tracks als auch Messpunkte nicht berücksichtigt worden sind. Für alle nicht verwendeten Messungen wird ein neuer Track gestartet. Oftmals werden solche neuen Tracks, bzw. die Geschwindigkeit dieser Tracks, über die Differenz zweier aufeinanderfolgender Messungen initialisiert. Da im RoboCup plötzlich auftretende hohe Geschwindigkeiten für das restliche Gesamtsystem aber deutlich schlimmer sind, als etwas langsamer startende Geschwindigkeiten, werden neue Tracks immer direkt aus einem Messpunkt und einer Geschwindigkeit von Null initialisiert.

Für Tracks, denen kein Messpunkt zugeordnet wurde, wird ein interner Zähler inkrementiert, welcher bei einer neuen Zuordnung zurückgesetzt wird. Übersteigt dieser Zähler eine bestimmte Schwelle, erhält der Track also über einen längeren Zeitraum keine neue Messung zugeordnet, so wird er beendet.

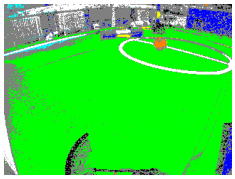
7.4 Verwandte Arbeiten

Verschiedene Arbeiten beschäftigten sich bereits mit der Frage, inwieweit sich Kalman-Filter-basierte Systeme überhaupt für die visuelle Objektverfolgung eignen. Eine dieser Arbeiten ist die Veröffentlichung von Nathan Funk [42]. Die Auswertungen unterscheiden sich von der vorliegenden Arbeit darin, daß der Autor nur Fälle konstanter Bewegung und konstanter Beschleunigung untersucht hat. Auch wurde das teilweise eingebrachte, zusätzliche Rauschen nur auf die Originalbilder, und damit auf die Erkennung angewendet, nicht aber auf die Geschwindigkeit oder die Beschleunigung selbst. Fazit der Arbeit ist, daß sich Kalman-Filter sehr wohl für die visuelle Objektverfolgung eignen, diese aber exakt konfiguriert und parametrisiert sein müssen.

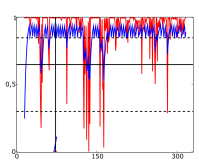
Eine weitere, ausführliche Auswertung gibt es in einer Veröffentlichung von Ferrein et. al. [33]. Obwohl diese Arbeit eigentlich von globaler Sensorfusion und Roboterlokalisation handelt, sind die Ergebnisse auch für das hier vorgestellte Szenario interessant. Verglichen wurden Verfahren wie unterschiedliche Varianten des arithmetischen Mittels, gitterbasierte Diskretisierungen, Monto-Carlo-Methoden und eben ein einfacher, Kalman-basierter Ansatz. Dabei erreichte der Kalman-Filter zusammen mit der Monte-Carlo-Methode die besten Fusionsleistungen. Zusätzlich zeigte sich aber auch, daß die benötigte Rechenleistung, zusammen mit dem arithmetischen Mittel, die mit Abstand geringsten Werte erreichte. Ein Kalman-basiertes Verfahren erwies sich damit als am exaktesten und gleichzeitig am schnellsten von allen untersuchten Methoden.

7.4.1 RoboCup





Auch auf den einschlägigen RoboCup-Konferenzen wurden bereits einige Arbeiten vorgestellt, die sich mit dem Thema Objektverfolgung und Tracking beschäftigen. In einer Arbeit von Solar und Vallejos [119] ist das Tracking ebenfalls mit Hilfe von Kalman-Filtern realisiert. Diese sind integriert in ein System, das über direkte Bildvergleiche sich bewegende Objekte sucht. Dabei wird versucht, aufeinanderfolgende Bilder auszurichten und dann zur Deckung zu bringen. Nach einigen heuristischen

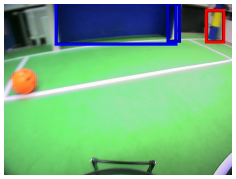


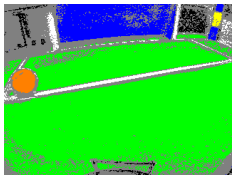
Operationen zur Rauscheliminierung und Gruppierung der Bildunterschiede deuten die verbleibenden Unterschiede auf bewegte Objekte hin. Diese Objekte werden analog dem oben beschriebenen Verfahren über einen Kalman-Filter je Objekt sowie einem ähnlichen Verfahren zur Objekt-Track-Zuordnung über die Zeit beobachtet. Einer der Unterschiede besteht darin, daß offensichtlich auch die Beschleunigung sowie ein Maß für die Änderung der Objektgröße über die Zeit mitverfolgt wird. Leistungsvergleiche sind später im Auswertungskapitel in Abschnitt 9.4 aufgeführt.

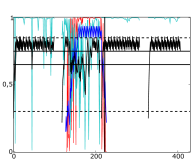
Einen wesentlich komplexeren Ansatz verfolgen Schmitt et. al. [112, 113]. Zur zeitlichen Verfolgung gegnerischer Roboter werden hier die Sensorinformationen aller im eigenen Team vorhandenen Roboter zu einem gemeinsamen Weltmodell fusioniert. Im Unterschied zum hier vorgestellten Verfahren werden neue Beobachtungen nicht gemäß einem „1 : n“-Ansatz nur jeweils einem existierenden Track zugeordnet, sondern alle Beobachtungen werden, gewichtet mit ihrer Zugehörigkeitswahrscheinlichkeit, allen Tracks aufgeschlagen. Dieses probabilistische Multi-Hypothesen-Tracking erzeugt damit bei jeder Bildauswertung neue mögliche Hypothesen. Diese wachsende Menge an Tracks muß dann über die Zeit wieder über sogenannte Pruning-Verfahren auf die wahrscheinlichsten Hypothesen eingegrenzt werden. Dazu werden etwa Schwellen für die minimal notwendige Wahrscheinlichkeit eingeführt. Außerdem wird berücksichtigt, wieviele neue solcher Nachfolge-Tracks aus einem bestehenden Track entstehen. Dabei wird ausgenutzt, daß sich jede Mehrdeutigkeit bei einer gewissen Anzahl von Tracks nach einer bestimmten Zeit durch die Menge der überhaupt möglichen Zuordnung wieder auflösen muß. Wird diese Zeitschwelle überschritten, können solche mehrdeutigen Nachfolge-Tracks wieder gelöscht werden.

Viele weitere Ansätze existieren, die nicht Kalman-Filter, sondern Partikelfilter auf Basis einer Monte-Carlo-Methode verwenden. Beispiele dafür, wenn auch nicht aus dem RoboCup, finden sich in den Arbeiten von Black und Jepson [13] oder auch Isard und Blake [65], die beide visuell detektierte Merkmale mit Hilfe solcher Kondensationsalgorithmen verfolgen. Im Falle von Black et. al. sind es Gesten, also kurze, skizzierte Bewegungsmuster, mit denem man beispielsweise einen Computer steuern kann. Im Falle von Isard et. al. sind es ganz allgemeine Liniensegmente oder -kurven, die ein Objekt oder Objektteil repräsentieren.

Eine noch aufwendigere Mischform wird in der Arbeit von Kwok und Fox [86] vorgestellt. Am Beispiel der Ballprädiktion wird ein System beschrieben, das ein partikelbasiertes Filterverfahren so mit Kalman-Filtern kombiniert, daß jedes einzelne der Partikel durch einen eigenen Kalman-Filter repräsentiert wird. Dieser Ansatz soll die Mächtigkeit eines Partikelfilters in der Beschreibung mit der Effizienz und Genauigkeit des Kalman-Filters vereinen. Zusätzlich existieren unterschiedliche Filtermodelle für alle denkbaren Situationen wie einen rollenden Ball, einen abprallenden Ball oder einen gerade eben gekickten Ball. Durch eine solche Aufteilung lassen sich auch Nichtlinearitäten des Systems modellieren. Ein offensichtlicher Nachteil dieses Vorgehens ist die unglaubliche Berechnungskomplexität. Selbst nach Berücksichtigung eigens entwickelter Vereinfachungen benötigt das System nach Aussage der Autoren immer noch etwa 50 Partikel um die eigene Roboterposition zu verfolgen, sowie weitere 20 Partikel für einen beobachteten Ball. Ein solcher Aufwand ist nicht nur auf der be-







nutzten AIBO-Roboter Plattform wesentlich zu hoch, um das System in Echtzeit zu betreiben.

Keine der oben vorgestellten Arbeiten integriert das Modul zur zeitlichen Objektverfolgung aber direkt in die Objekterkennung. Selbst Arbeiten wie das zuerst vorgestellte Verfahren von Solar und Vallejos [119], die dadurch deutlich einfacher wahrscheinliche Hypothesen aus den Bildfolgen extrahieren könnten, lassen beide Aufgabenbereiche strikt getrennt. So ist es nicht möglich, aus den zeitlichen Beobachtungen unmittelbaren Nutzen für die Objekterkennung zu ziehen.

7.4.2 Automobilbereich und Robotik allgemein

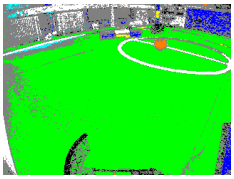
Da auch in anderen Bereichen der Robotik allgemein und im Bereich der autonomen Transportsysteme vergleichbare Probleme zu lösen sind, hier noch ein paar weitere Beispiele verwandter Arbeiten.

Einer der bekanntesten Ansätze kommt hier mit dem erwartungsbasierten 4D-Verfahren aus der Arbeitsgruppe um Ernst Dickmanns [23, 24]. Auch hier werden Symbole in einer spatiotemporalen Repräsentation, also mit ihren 3-dimensionalen Koordinaten sowie der 1-dimensionalen, zeitlichen Komponente, gespeichert. So werden Computerressourcen nur für potentiell wichtige Bereiche im Bild aufgewendet. Zusätzlich läßt sich so auch die Eigenbewegung der hier eingesetzten beweglichen Kamera kompensieren. Angewendet wurde dieses Verfahren aber hauptsächlich für die Straßenverfolgung und einfache Hinderniserkennungen. Eine Erweiterung dieser Idee ist das sogenannte „Expectation-based, Multi-focal, Saccadic - EMS-System“, also ein System mit zusätzlichen speziellen Kameras für die Nah- und Fernsicht [7, 25]. Damit werden nun auch einzelne, definierte Objektklassen erkannt und verwaltet.

Auch in der bereits erwähnten Arbeit von Piccoli et. al. [105] werden Objekte, in diesem Fall Verkehrsschilder, nach der Erkennung mit Hilfe von Kalman-Filtern verwaltet. Identisch zur Arbeit von Bertozzi et. al. [8] werden diese Objekttrajektorien dazu verwendet, im nachfolgenden Bild den möglichen Suchraum für die Objekterkennung einzuschränken. Die eigentlich recht ausführliche Auswertung bei Bertozzi leidet aber unter der Tatsache, daß diese im Innenbereich mit einem stehenden Auto, also in einem sehr artifiziellen Szenario, durchgeführt wurde.

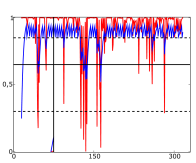
In diesem Bereich gibt es einige Verfahren, die zumindest das Tracking direkt in die Objekterkennung integrieren. Meist bedeutet dies aber, daß nur eine Vorauswahl der zu bearbeitenden Bildbereiche vorgenommen wird. Der komplette weitere Ablauf der Bearbeitungskette bleibt aber immer der gleiche. Die temporale Informationsintegration beeinflusst dabei nicht die weitere Verarbeitung, also etwa in der Art, daß Teile der Bearbeitungskette ausgelassen werden, oder alternative Bearbeitungsschritte durchgeführt werden.





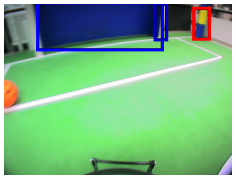
7.5 Kernpunkte

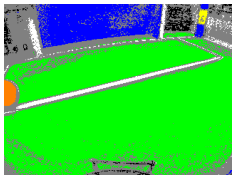
In diesem Kapitel wurde ein Objektverfolgungs- und -prädiktionssystem vorgestellt, welches es erlaubt, die zeitliche Integration direkt in das Gesamtsystem zur Objek-

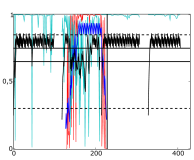


terkennung zu integrieren. Durch eine sorgfältige Abwägung zwischen theoretischer Leistungsfähigkeit und praktischem Berechnungsaufwand kann dieses Gesamtsystem sehr leistungsfähig realisiert werden. Dazu gehören Punkte wie die Verwendung eines einfachen, linearen Kalman-Filter, die Modellierung der relativen Beschleunigung mittels der sogenannten „discrete white noise acceleration“-Methode oder die Bevorzugung von älteren, bewährten Tracks bei der Datenzuordnung. Wie im folgenden Auswertungskapitel zu sehen sein wird, ist die Wahl der Filtermodelle und der gewählte Ansatz für die Daten-Track-Zuordnung für dieses Szenario angemessen.

Selbstverständlich existieren Situationen, die den gewählten Annahmen widersprechen und damit beispielsweise das Zuordnungssystem nicht mehr wie erwartet funktioniert. So kann etwa dann, wenn ein Ball von einem Gegenstand abprallt, durch die starke Nichtlinearität ein zusätzlicher neuer Track entstehen. Durch eine schnelle Konvergenz des neuen Kalman-Filters und die wenig rechenintensive Verwaltung des alten Tracks bleibt das Gesamtsystem aber auch in solchen Situationen stabil und schnell.







8 Informationsextraktion und -verwendung

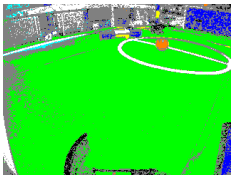
In diesem Kapitel sollen abschließend alle Punkte angesprochen werden, die mit der Extraktion und Verwendung der gewonnenen Informationen in Zusammenhang stehen. Dazu wird im folgenden Abschnitt 8.1 der letzte Schritt des gesamten Objekterkennungsvorgangs, die Integration der Sicherheits- und Wahrscheinlichkeitswerte über die Zeit, beschrieben. Da der gesamte Prozess der Kalman-Filterung eine rekursive, geschlossene Bearbeitungsschleife darstellt, können Wahrscheinlichkeitswerte, wie sie von den neuronalen Klassifikatoren geliefert werden, von Schritt zu Schritt eine immer validere Basis für die abschließende Bewertung liefern. Zusätzlich sind, abhängig von diesen kumulierten Sicherheitswerten, andere, alternative Bearbeitungspfade direkt in der Bildverarbeitung möglich.

Abschnitt 8.2 zeigt dann konkret, wie das nun komplett vorgestellte Objekterkennungssystem im Detail aussieht. Hier soll einmal die gesamte Filterkonfiguration gezeigt werden, wie sie tatsächlich während offizieller Meisterschaften verwendet wird, und wie sie komplett implementiert in dem Projekt LIBVIDEOFILTERS jedem Interessierten zu Verfügung steht. Anhand dieser Gesamtarchitektur läßt sich zeigen, wie sich alle möglichen Bearbeitungspfade in einem großen Bearbeitungsbaum definieren lassen, aber nur solche Teilbäume bearbeitet werden, die vom Rest des Systems tatsächlich gebraucht werden. Am Ende des Kapitels dann noch ein Abschnitt mit einer kurzen Zusammenfassung dieser Themen und eine Auflistung der wichtigsten hier vorgestellten Fakten und Neuerungen.

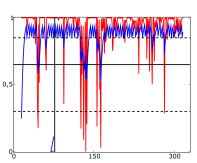
8.1 Wahrscheinlichkeit und Sicherheit

Die enge Kopplung der Objektklassifikation mit der Objektprädiktion erlaubt eine sehr umfassende Integration der (Un-)Sicherheitswerte innerhalb dieser Bearbeitungsschleife. In Abschnitt 3.2.3 wurde dieses Verfahren bereits kurz angesprochen. Abhängig von den Werten der bisherigen, integrierten Objektpfade und den Wahrscheinlichkeitswerten der aktuellen neuronalen Klassifikation können Bearbeitungsschritte entweder ausgelassen, ausgeführt oder Alternativen ausgewählt werden. Im Einzelnen läßt sich folgende Unterscheidung für die verschiedenen, zu integrierenden Werte treffen:





- Wird eine neuronale Klassifikation ausgeführt, kann das Ergebnis dieser Klassifikation direkt als neuer Wahrscheinlichkeitswert c_i für die Integration verwendet werden. Falls der Klassifikator aus mehreren neuronalen Netzen besteht, ist die-



ser Wert der Mittelwert der Einzelergebnisse aller Netze, sofern alle Netze einen Wert über der gewählten Schwelle von 0,75 besitzen. Ansonsten wird hier der kleinere der beiden Werte verwendet.

- Wenn der Wert des bisherigen Tracks, das bereits kumulierte Ergebnis der Sicherheitswerte des vorher erkannten Objektpfads, über oder unterhalb einer bestimmten festgelegten Schwelle liegt, wird keine neuronale Klassifikation durchgeführt. In diesem Fall wird ein im voraus bestimmter a-priori-Wert für die Aufmerksamkeitssteuerung als c_i verwendet. Dieser Wert beschreibt die durchschnittliche Korrektheit der Aufmerksamkeitssteuerung bezüglich dieses Objekttyps.

In beiden Fällen wird der neue Wert über folgende Integrationsformel berechnet:

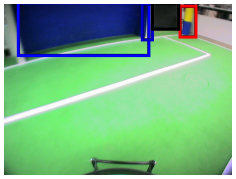
$$c_{t+1} = c_t + \tau(-c_t + c_i),$$

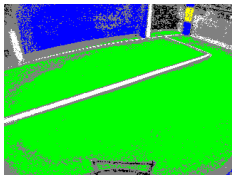
wobei c_t der alte Sicherheitswert ist (der Wert des zugeordneten Tracks), c_i eben jener neu ausgewählte Einzelwert (entweder Klassifikationergebnis oder a-priori-Aufmerksamkeitskorrektheit) und τ ein Gewichtungsfaktor, der bestimmt, wie stark das aktuelle Ergebnis in das neue Gesamtergebnis c_{t+1} einfließt.

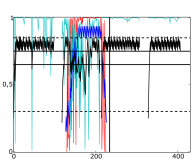
Für Tracks, die nicht mit Hilfe aktueller Observationen aktualisiert werden, denen aktuell also keine neue Positionshypothese zugeordnet wurde, wird der zu integrierende Sicherheitswert c_i anders bestimmt:

- Falls die Trackvorhersage die Position als außerhalb des Kamerasichtbereichs liegend vorhersagt, der Roboter das Objekt daher gar nicht erkennen konnte, wird ein a-priori-Wert $\tilde{c_i}$ für die Wahrscheinlichkeit, daß sich an dieser Stelle tatsächlich ein Objekt befindet, verwendet. Dieser Wert wird zusätzlich mit einem Gewichtungsfaktor verrechnet, der die momentane Vorhersagegenauigkeit berücksichtigt. Je länger ein Objekt nicht gesehen wurde, desto weniger sicher wird die Integration der Wahrscheinlichkeiten ausfallen.

- Wenn die Trackvorhersage die Position aber als innerhalb des Sichtbereichs der Roboters annimmt, das Objekt deswegen hätte gesehen werden können, unterscheidet sich die Vorgehensweise abhängig vom Ergebnis einer weiteren, nur dann nötigen Berechnung. Ein extra Modul entscheidet dann, ob das gesuchte Objekt hätte gesehen werden müssen, oder ob es sich eventuell hinter einem anderen Objekt befindt. In diesem Fall hätte der Roboter gar keine Möglichkeit, es zu sehen.







  - Ist das Objekt mit hoher Wahrscheinlichkeit durch ein anderes verdeckt, so wird derselbe a-priori-Sicherheitswert wie im obigen Fall verwendet. Dieser kann auch hier wieder über die Trackgenauigkeit gewichtet werden.
  - Wird aber andersherum mit hoher Wahrscheinlichkeit der Fußboden gesehen (hinter dem sich kein Objekt verstecken kann), so wird hier der, normalerweise kleinere, a-priori-Wert genommen, welcher beschreibt, wie

oft die Aufmerksamkeitssteuerung ein eigentlich vorhandenes Objekt übersieht (wiederum gewichtet mit der Vorhersagegenauigkeit).

Dieser gewichtete Sicherheitswert wird dann wie folgt ermittelt:

$$c_i = \begin{cases} \tilde{c}_i \frac{1}{e_m}(e_m - e_f) & \text{falls } e_m > e_f \\ 0 & \text{sonst} \end{cases}$$

wobei e_f der Vorhersagefehler ist und e_m das Maximum des tolerierbaren Fehlers, also die Obergrenze, ab wann der Vorhersage keinerlei Gewicht mehr beigemessen wird. Alle diese a-priori-Werte für die durchschnittliche positive und negative Korrektheit der Aufmerksamkeitssteuerung lassen sich recht einfach über repräsentative Datensammlungen ermitteln. Im vorliegenden Fall sind diese Bilder echte Aufnahmen aus vielen Spielen vergangener Meisterschaften (siehe auch Abschnitt 9.2).

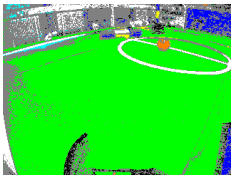
Aktuell ist die Entscheidung, ob sich ein im Sichtfeld des Roboters befindliches Objekt frei zu sehen, oder aber verdeckt ist, über eine relativ einfache Heuristik realisiert. Da das Spielfeld im RoboCup grün ist, alle anderen Objekte aber kein Grün enthalten, ist die Entscheidung über einen einfachen Schwellenwert des Anteils enthaltener grüner Farbe im entsprechenden Ausschnitt der Indexbilds realisiert. Eine solch einfache Heuristik funktioniert aber natürlich nur im konkreten Anwendungsfall. Trotzdem ist das Prinzip aber auch in anderen Szenarien anwendbar. Falls sich die Entscheidung nicht oder nur schwer über visuelle Features realisieren läßt, kann sie beispielsweise auch über zusätzliche Distanzsensoren erfolgen, welche dieselbe Blickrichtung wie die Kamera besitzen. Sollte ein solcher Sensor ein Objekt in einem Abstand entdecken, welcher sich wesentlich vor der vermutlichen Position befindet, kann mit großer Wahrscheinlichkeit von einer Verdeckung ausgegangen werden. Nebenbei bemerkt, macht sich auch hier die gute Integration von Bildverarbeitungssytem und Robotermiddleware bezahlt. In einem Filtermodul lassen sich ohne Probleme alle möglichen Werte des Robotersystems wie eben solche Distanzsensorwerte abfragen.

8.2 Aufgabenbasierte Informationsextraktion

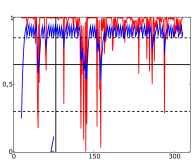
Eine Besonderheit des hier vorgestellten Gesamtverfahrens ist es, daß die Auswertung aller definierten Komponente nur Bedarfsabhängig geschieht. Bedarfsabhängig läßt sich hierbei in zwei Gruppen unterteilen:

- Berücksichtigung der Erwartung oder der Information aus anderen Teilkomponenten des Objekterkennungssystems.
- Berücksichtigung der Anforderung der von der Objekterkennung getrennten Komponenten wie der Selbstlokalisation.





Zur ersten Rubrik gehört die oben erwähnte, bedarfsabhängige Untersuchung, ob sich an einer Stelle potentiell ein Objekt befindet oder doch mit hoher Wahrscheinlichkeit der Fußboden des Spielfelds. Diese Untersuchung erfolgt aber nur, wenn Ergebnisse



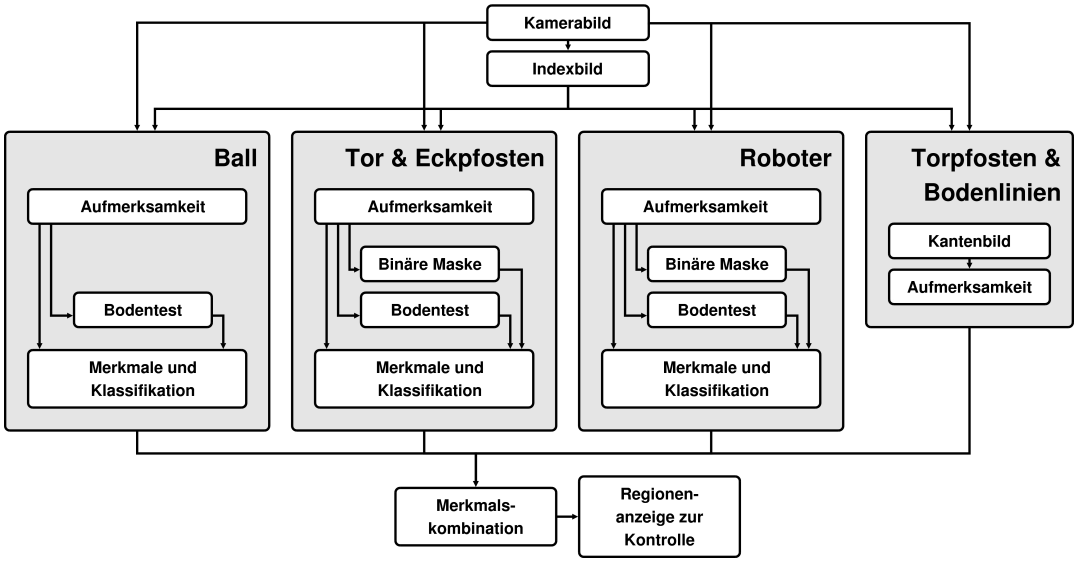
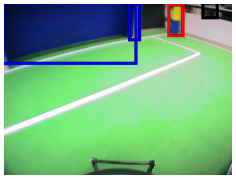
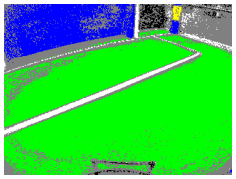


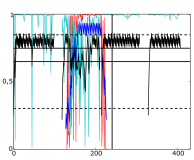
Abbildung 8.1: Schematische Darstellung der gesamten Filterkonfiguration. Zu sehen sind die jeweiligen Arbeitsbereiche Ball, Tor-, Eckpfosten und Roboterdetektion sowie die Erkennung der Torpfosten und Bodenlinien. Alle Gegenstände, die durch einen neuronalen Klassifikator validiert werden, besitzen dabei einen ähnlichen Aufbau.

der Aufmerksamkeitssteuerung und der Objektverfolgung eine solche Untersuchung erfordern. Die zweite Rubrik der Anforderung anderer, von der Objekterkennung unabhängiger Komponenten kann aber ebenfalls Einfluß auf die zu bearbeitenden Filter nehmen. Je nachdem, welche Filter von anderen Komponenten abgefragt werden, werden die zu diesem Filter gehörigen Teilbäume der Filterkonfiguration auch nur dann abgearbeitet.

Abbildung 8.1 zeigt eine schematische Darstellung der gesamten Filterkonfiguration, wie sie für das nun vollständig erklärte Gesamtsystem verwendet wird. Die Verbindungslinien symbolisieren sowohl den Bilddatenfluß als auch Fluß der sogenannten Meta-Daten wie zum Beispiel Listen von Aufmerksamkeitshypothesen. Von links nach rechts aufgezeigt ist die Ball-, die Tor- und Eckpfostenerkennung sowie die Roboterdetektion. Der Bereich ganz rechts ist für die Erkennung der Torpfosten sowie der Bodenlinien zuständig (Abbschnitt 5.3). Grundsätzlich können nun Ergebnisse an jeder beliebigen Stelle innerhalb dieses Filterbaums abgefragt werden. Im realen Einsatz werden aber wohl meist fertig klassifizierte und validierte Objekte gesucht sein, das heißt die Abfrage wird am Ausgang der hier grau dargestellten Blöcke erfolgen. Blöcke, deren aktuelles Endergebnis für keines der auf dem Roboter laufenden Module von Bedeutung ist, werden erst gar nicht abgearbeitet. Solche nicht benutzen Arbeitsblöcke werden dann komplett von der Bearbeitung ausgenommen. Analog existiert ein solcher Baum für die Auswertung der omnidirektionalen Rundumkamera. Deren Filterkonfiguration ist zwar wesentlich weniger komplex, die Möglichkeit zur selektiven Auswertung gilt aber selbstverständlich auch dort.







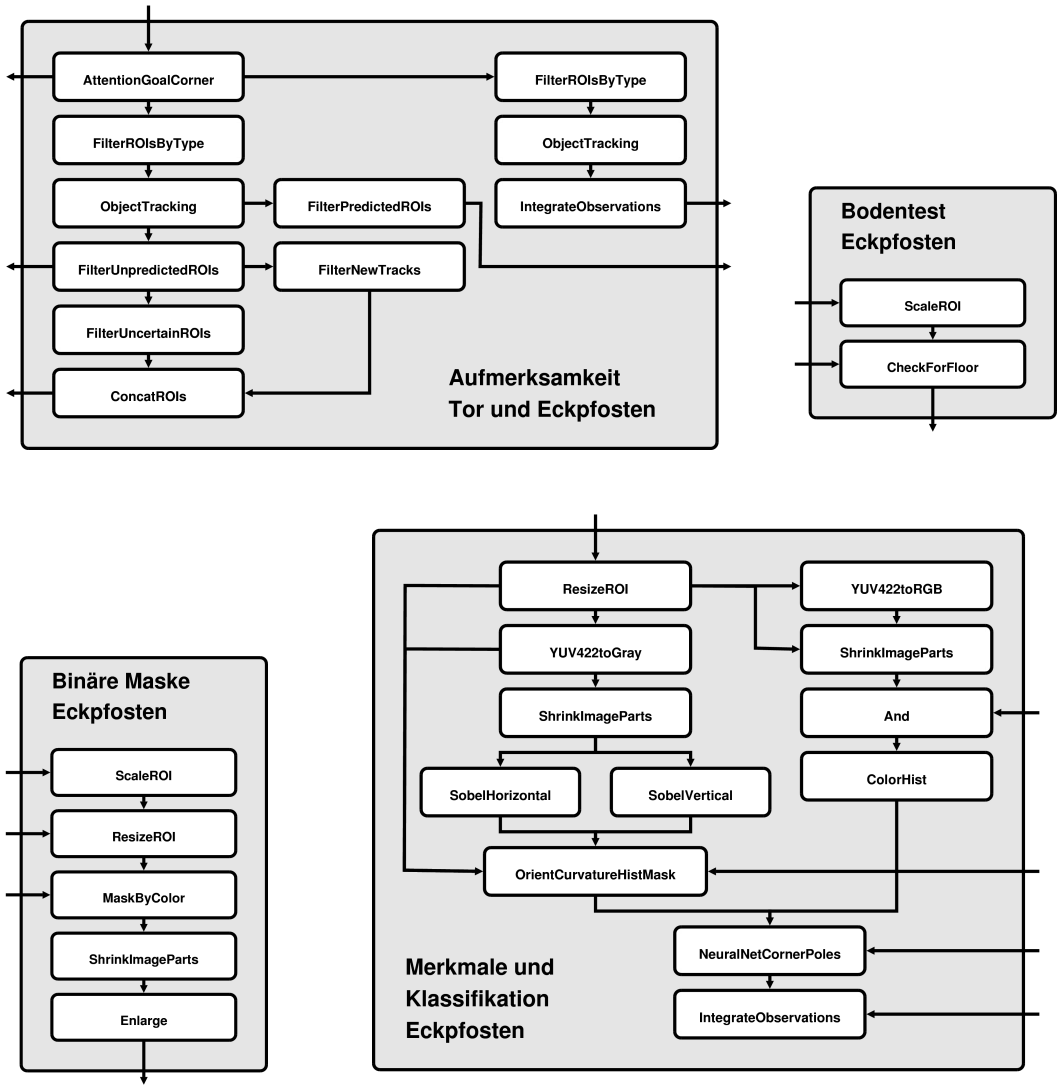
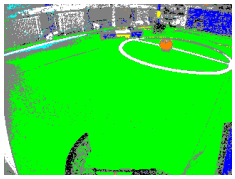
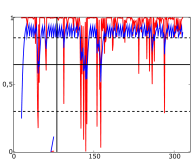


Abbildung 8.2: Detaillierte Ausschnitte verschiedener Arbeitsbereiche der Filterkonfiguration am Beispiel der Eckpfostenerkennung. Links oben die kombinierte Regionensuche für das Tor und die Eckpfosten basierend auf denselben Indexfarben. Rechts oben der simple Test, ob der Boden gesehen wird. In der zweiten Reihe links die Erstellung der binären Maske, rechts die eigentliche Merkmalsberechung und neuronale Klassifikation.







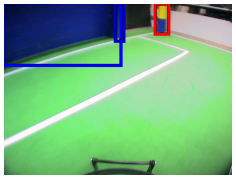
Zusätzlich läßt sich in dieser schematischen Darstellung gut erkennen, wie sich die einzelnen Arbeitsblöcke der neuronal zu klassifizierenden Objekte in ihrer Grundstruktur ähneln. Für den Ball wird aber beispielsweise der Hintergrund nicht über eine binäre Maske ausgeblendet, da durch dessen Form sowieso kein großer Anteil an Hintergrund im Bild sein kann.

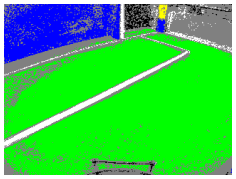
Der Vollständigkeit halber sind in Abbildung 8.2 nun noch die einzelnen Elemente filtergenau am Beispiel der Tor- und Eckpfostenerkennung dargestellt. Die einzelnen hier grau hinterlegten Blöcke entsprechen den einzelnen Unterpunkten aus dem Gesamtblock *Tor & Eckpfosten* der obigen Illustration. Links oben dargestellt die kombinierte Regionensuche für das Tor und die Eckpfosten. Da die Suche nach beiden Objekttypen auf der gleichen Indexfarbe beruht, ist die Hypothesenbildung hier aus Geschwindigkeitsgründen in ein Modul integriert. So muß nicht zweimal das gesamte Bild nach den eigentlich gleichen Farbblobs abgesucht werden. Die jeweils ein- und ausgehenden Pfeile entsprechen wiederum dem Bilddaten- und Meta-Datenfluß. In diesem Beispiel etwa erhält die eigentliche Regionensuche *AttentionGoalCorner* das Indexbild als Eingabe, der Filter *ConcatROIs* gibt die Gesamtliste aller vorhergesagten und prädizierten Eckpfostenhypothesen an die nächsten Filter aus. Rechts oben abgebildet ist das sehr einfache Bodentest-Modul, welches, wie im vorherigen Kapitel erklärt, nur genau dann ausgewertet wird, wenn die Regionensuche kein Objekt an einer bestimmten Stelle erkannt hat, die Trackverfolgung aber an dieser Stelle ein solches Objekt vorhergesagt hat. Links in der unteren Reihe ist dargestellt, wie die binäre Maske erstellt wird, rechts die eigentliche Merkmalsberechung und neuronale Klassifikation. Auch hier sind die beiden benutzen Merkmale gut zu erkennen: Links die Berechnung der kombinierten Orientierungs-/Krümmungshistogramme, rechts die Erstellung des Farbhistogramms, beide werden anschließend durch die neuronalen Netze validiert.

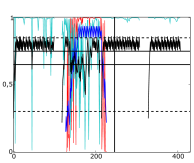
Kurze Bemerkung am Rande: Wie in Abschnitt 4.2 dargestellt, muß nur der oberste Wurzelknoten namens Kamerabild durch einen anderen Filter ausgetauscht werden, und schon kann die gesamte Filterkonfiguration ebenso dateibasiert während der Entwicklungs- und Testphase verwendet werden.

8.3 Kernpunkte

In diesem Kapitel sind zwei wesentliche Kernpunkte beschrieben, die das dargestellte Verfahren ausreichend schnell und reaktiv für den Einsatz in hochdynamischen Umgebungen machen. Wie gesehen, ist die Grundidee dabei, nur die nötigsten Rechenschritte tatsächlich auszuführen.





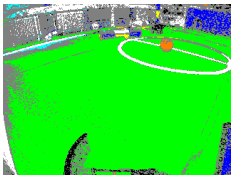


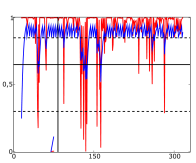
Algorithmisch wurde dies dadurch erreicht, daß Objekte nur dann neuronal validiert werden, wenn ihre Existenz noch nicht ausreichend oft verifiziert wurde. Iterationen ohne einen solchen Klassifikationsschritt verwenden stattdessen eine ausgeklügelte Mischung aus szenariospezifischem a-priori-Wissen und Informationen über die Genauigkeit der Objektvorhersage. Dieser Schritt ist so nur möglich, wenn Objekterkennung und Objektverfolgung ein integriertes Gesamtsystem darstellen. Daß dies

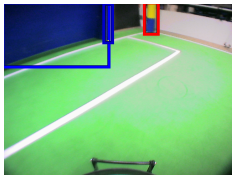
keine Selbstverständlichkeit darstellt, wurde bei der Untersuchung anderer, relevanter Arbeiten in Abschnitt 7.4 deutlich.

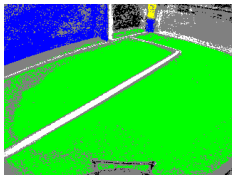
Die Infrastruktur garantiert unterdessen, daß komplette Teilbäume des gesamten Erkennungssystem nur dann ausgeführt werden, wenn andere, davon unabhängige Komponenten des auf dem Roboter laufenden Systems die Ergebnisse eines solchen Teilbaums benötigen. So werden etwa die für die Selbstlokalisation des Roboter nötigen Merkmale nur dann ausgewertet, wenn das Lokalisierungsmodul diese von der Bildverarbeitung anfordert. Zwar ist eine bedarfsabhängige Auswertung wohl auch bei vielen anderen im RoboCup teilnehmenden Mannschaften realisiert. Eine Umsetzung auf Infrastrukturebene entlastet aber den Entwickler davon, das Gesamtsystem kennen zu müssen und reduziert Fehler, wenn in anderen, unabhängigen Modulen Änderungen vorgenommen werden.

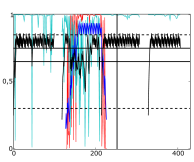








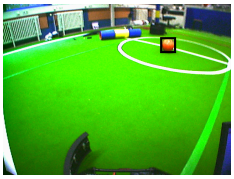


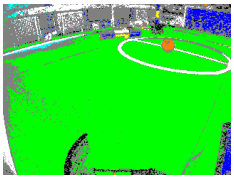

0,5
0
0
200
400

9 Empirische Evaluation

In diesem Kapitel folgt nun die Auswertung aller Einzelkomponenten sowie des resultierenden Gesamtsystems. In Abschnitt 9.1 will ich zuerst die bereits in Abschnitt 2.2 angesprochene Untersuchung der Beleuchtungseigenschaften derzeitiger RoboCup-Spiele bzw. der zu erwartenden Beleuchtungssituation während zukünftiger Spiele unter natürlichen Bedingungen ausführen. Insbesondere ist interessant, wie sich eine solche zu erwartende Beleuchtungssituation auf die bestehenden Verfahren im RoboCup auswirken wird. Diesem Abschnitt folgt eine Untersuchung der durchschnittlichen Leistungen der verschiedenen Aufmerksamkeitsmodule in Abschnitt 9.2. Einige negative Beispiele zeigen, mit welchen Objekten die nachfolgende neuronale Klassifikation zurechtkommen muß. Diese Objektklassifikation wird dann in Abschnitt 9.3 ausgewertet. Von besonderem Interesse ist vor allem die Leistung bei der Roboterkennung, da hier die weitaus größte Varianz an unterschiedlichen Objekttypen und -instanzen vorherrscht. Erste Beispiele hierzu waren bereits in Abschnitt 2.1 zu sehen. Als letzte Einzelkomponente folgt dann die Bewertung der Objektprädiktion und -verfolgung in Abschnitt 9.4. Die Untersuchungen werden auch hier zeigen, daß das gewählte System unter den gegebenen Bedingungen, wie etwa der Anzahl der maximal auftretenden Instanzen eines Objekttyps sehr leistungsfähig bleibt. Abschließend folgt die Untersuchung des Gesamtsystems inklusive einer Auswertung der verbrauchten Rechenzeit einiger, besonders aufwendiger Berechnungsschritte. Ein quantitativer Vergleich mit anderen Arbeiten im RoboCup ist leider nicht möglich, da nach meinem Kenntnisstand nur in sehr wenigen vergleichbaren Publikationen überhaupt eine sinnvolle, quantitative Auswertung der vorgestellten Verfahren erfolgt. Aufgrund unterschiedlicher Annahmen und Voraussetzungen, wie etwa die Verwendung omnidirektionaler Kameras, lassen sich aber auch diese Daten nicht mit den hier ermittelten Werten vergleichen.

9.1 Beleuchtungssituation





Häufige Debatten in der RoboCup-middle-size-Liga über natürliche Beleuchtungsbedingungen zeigten, daß eine detaillierte Untersuchung dieses Problems nötig war, um das weitere Vorgehen innerhalb dieser Liga, und damit die zukünftige Forschungsrichtung, vernünftig planen zu können. Auf der Basis von realen Bildern, die unter verschiedenen Beleuchtungsbedingungen aufgenommen wurden, ist eine statistische und beschreibende Analyse der zu erwartenden Effekte auf die vornehmlich farbbasierten Verfahren im RoboCup entstanden. Die Ergebnisse zeigen, daß solche rein auf Farben basierenden Verfahren in einem Umfeld variabler Beleuchtung nicht mehr

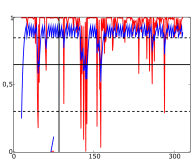


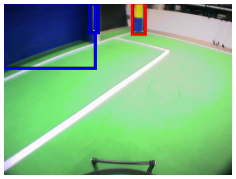


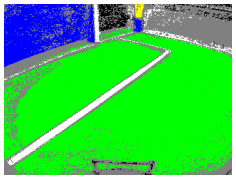
Abbildung 9.1: Das neue Gebäude der Fakultät für Informatik der TU München in Garching, in dem die meisten Bilder für die Untersuchung in diesem Kapitel aufgenommen wurden. Front und Dach bestehen fast vollständig aus Glas (http://de.wikipedia.org/wiki/TU_München, aufgenommen von Martin Röll).

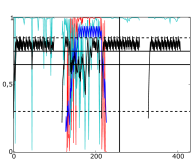
zuverlässig funktionieren werden, selbst wenn deren Parameter vor jedem Spiel wieder neu kalibriert werden. Diese Erkenntnis war die eigentliche Motivation für das in dieser Arbeit vorgestellte Gesamtsystem.

9.1.1 Übersicht der Beleuchtungsbedingungen

Um ein besseres Verständnis für die Effekte von und durch variierende Beleuchtungsbedingungen zu erhalten, haben wir verschiedene unterschiedliche Bedingungen untersucht. Das erste Szenario ist jenes, innerhalb dem im Moment die RoboCup Spiele ausgetragen werden: Ein Fußballfeld wird ausschließlich durch künstliches Licht beleuchtet. Die aktuellen Regeln begrenzen die Varianz des Lichts auf Werte zwischen 300 und 1200 Lux, legen aber sonst keine weiteren Werte wie etwa für die Farbtemperatur fest. Es wird aber versucht, zu harte Schatten durch eine möglichst diffuse Beleuchtung zu vermeiden. Nachdem das Licht einmalig zu Beginn eines Turniers eingestellt wird, wird es im Weiteren über den gesamten Verlauf nicht mehr verändert. Das nächste Szenario ist charakterisiert durch kleine Veränderungen über die Zeit, wie unter natürlichem Licht an einem bewölkten, nicht allzu sonnigen Tag. Auch wenn hier keine schnellen Veränderungen zu erwarten sind, ändert sich das Licht doch zumindest durch den Lauf der Sonne im Verlauf des Tages. Das dritte mögliche Szenario zeigt schnelle Änderungen, harte Schatten und extreme Schwankungen über die Zeit und über die einzelnen Positionen auf dem Spielfeld. Solche Bedingungen herrschen an sonnigen Tagen, an denen vereinzelte Wolken vor die Sonne treten können und die Beleuchtungsstärke von einer Sekunde auf die andere wechseln kann.







Auf Basis dieser drei Szenarien wurden eine Reihe von Bildern aufgenommen, die diese Situationen möglichst gut nachbilden. Zwei Sets von Bildern wurden während eines Freundschaftsspiels mit dem Team Agilo der TU München zum Anlass der Einweihung ihres neuen Gebäudes der Fakultät für Informatik aufgenommen. Ein Foto dieses Gebäudes ist in Abbildung 9.1 zu sehen. Deutlich erkennbar ist, daß das komplette Dach sowie die Front vollständig aus leicht bläulich eingefärbtem Glas bestehen.

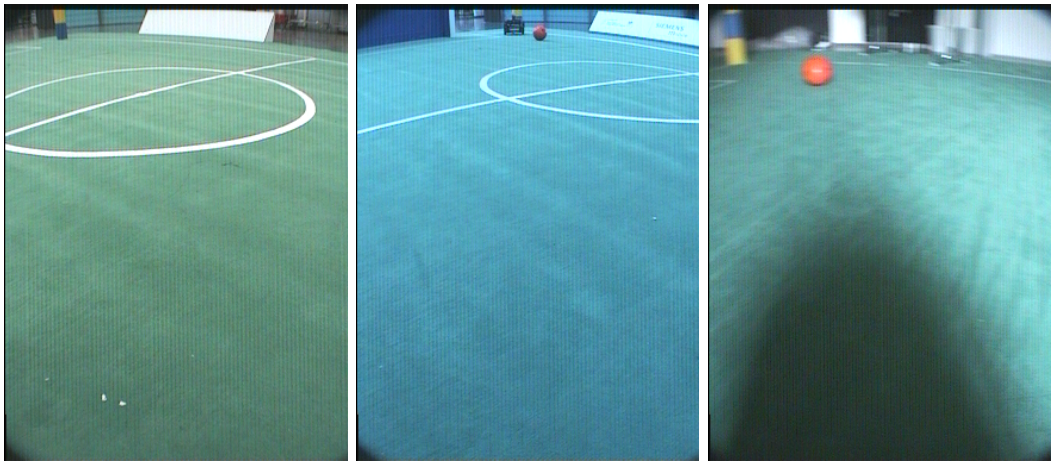


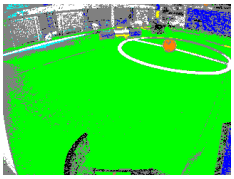
Abbildung 9.2: Beispiele für die drei verschiedenen Beleuchtungsbedingungen, wie sie für die Auswertung hier verwendet wurden. Im linken Bild Neonlicht während eines Freundschaftsspiels gegen das Münchner RoboCup-Team. In der Mitte dasselbe Spielfeld bei natürlichem Sonnenlicht. Rechts ebenfalls natürliches, aber stark gerichtetes Licht in unserem Labor. Die verwendeten Tore, Eckfahnen, Ball und Teppich sind in allen Bildern identisch.

Die Bildfolgen wurden während des Aufbaus am Vorabend unter künstlichem Licht (Set I) und während etwa 5 Stunden am Mittag des Folgetages (ein relativ sonniger Tag mit gelegentlicher Bewölkung – Set II) aufgezeichnet. Die dritte Bilderreihe wurde in unserem Labor während eines besonders sonnigen Tages aufgenommen. Dieses Labor hat nur eine Fensterreihe an einer Seite des Raums. Dadurch entstehen starke Schlagschatten und die gesamte Ausleuchtung des Raumes ist extrem inhomogen (Set III). Bei beiden Gelegenheiten wurden sowohl der gleiche Teppich als auch die gleichen Tore und Eckpfosten verwendet. Die wahrgenommenen Farbunterschiede sind also allein auf die wechselnden Beleuchtungen zurückzuführen. In Abbildung 9.2 sind hierzu Beispiele abgebildet.

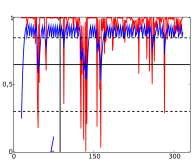
9.1.2 Untersuchungsmethode

Um den Effekt unterschiedlicher Beleuchtungsbedingungen zu untersuchen, wäre es am besten, eine Reihe von Bildern für exakt dieselben Situationen auf dem Spielfeld, aber unter wechselnden, genau bekannten und kontrollierten Bedingungen aufzunehmen. Dadurch ließe sich eine genaue Bewertung der Unterschiede durchführen. Aber der Wunsch nach einer natürlichen Beleuchtung macht dieses Vorhaben unmöglich. Schließlich läßt sich das Wetter nicht kontrollieren. Da aber eine farbbasierte Auswertung der Bilder im RoboCup immer noch die Methode der Wahl ist, können wir zumindest den Einfluß der verschiedenen Situationen auf solche Verfahren testen.





Im folgenden gehen wir davon aus, daß jede Farbklasse nur durch eine obere und untere Schwelle in jeder Dimension des Farbraums definiert ist. Obwohl viele Gruppen im RoboCup die Farbklassen deutlich flexibler bestimmen können, illustriert dieses



Vorgehen die Effekte am besten und ändert nichts an der grundsätzlichen Aussage. Im Einzelnen wurden die folgenden Experimente durchgeführt:

1. **Manuelle Kalibrierung der verschiedenen Beleuchtungssituationen:** Vergleich der Bereichsgrenzen der verschiedenen Farbklassen. Für jedes Szenario wurde eine Einstellung basierend auf einem repräsentativen Satz von Bildern durchgeführt.

2. **Räumliche Verteilung innerhalb der Farbklassen:** Mit Hilfe dieser Kalibrierungen wurde ein großer Satz von Bildern farbklassifiziert. Die räumliche Verteilung der innerhalb der Schwellen liegenden Farbpunkte wird durch ihren Mittelwert und deren Varianz bestimmt.

3. **Räumliche Verteilung der idealisierten Klassifikation:** Um zu zeigen, daß diese statistischen Werte nicht wesentlich von der Art der automatischen Kalibrierung beeinflusst werden, wurden einige Bilder von Hand „quasi perfekt“ indiziert. Die räumliche Verteilung dieser idealen sowie der automatisch indizierten Bilder läßt sich nun im HSV-Raum vergleichen.

4. **Einige ausgewählte Beispiele:** Abschließend sind noch einige ausgewählte Einzelbeispiele diskutiert, die verschiedene Effekte besser beschreiben als einfache statistische Maßzahlen dies könnten.

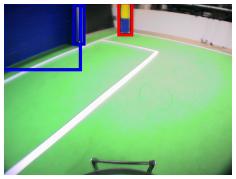
9.1.3 Resultate

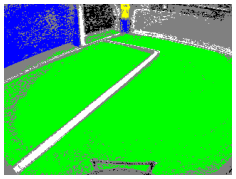
Manuelle Kalibrierung der verschiedenen Beleuchtungssituationen

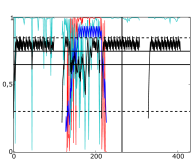
Abbildung 9.3 zeigt die resultierenden Bereichsgrenzen für die verschiedenen Beleuchtungsszenarien. Die verschiedenen Farben symbolisieren Bereichsgrenzen für die jeweilige Zielfarbe. Der weiße Kreis in der Mitte der Kegelbodens ist der kalibrierte Bereich der weißen Farbe. Wenn farbige Bereiche diesen weißen Kreis überlappen, werden diese Farben gegenüber Weiß bevorzugt.

Deutlich zu sehen ist, wie sich die Bereichsgrenzen zwischen den verschiedenen Szenarien deutlich unterscheiden. So rücken die blauen und grünen Bereiche in der rechten Abbildung (Set III) extrem nahe zusammen. Dies würde unter Umständen zu Fehlklassifikationen im realen Einsatz führen.

Räumliche Verteilung innerhalb der Farbklassen







Für diesen Versuch wurden jeweils mehr als einhundert Bilder jeder Beleuchtungssituation ausgewertet. Dazu wurden die Bilder mit Hilfe der vorbereiteten Kalibrierung farbsegmentiert. Alle Farbwerte des Originalbildes, die der dadurch bestimmten Farbklasse zugeordnet wurden, wurden zu einer Farbverteilung zusammengefasst. Wie eine solche räumliche Verteilung der Farben im HSV aussieht, ist in Abbildung 9.4 am Beispiel eines Bildes aus Set I (der künstlichen Beleuchtung) illustriert. Gut zu sehen ist die ansatzweise kegelartige Struktur des Farbraums sowie die Gruppierung der bunten Farbwerte rund um den Bereich der weißen Farbe.

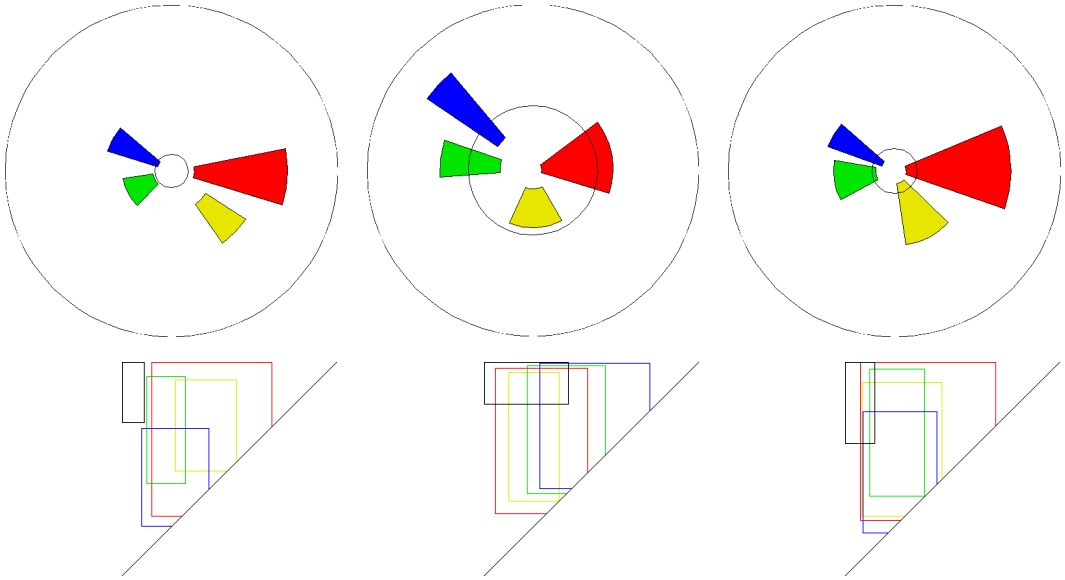


Abbildung 9.3: Resultat der manuellen Kalibrierung für die drei verschiedenen Beleuchtungsbedingungen. Die obere Reihe zeigt die Draufsicht auf den HSV-Kegel, also eine Projektion der Bereichsgrenzen auf die Hue/Saturation-Ebene. Die untere Reihe wiederum zeigt eine Projekte derselben Bereichsgrenzen in die Saturation/Brightness-(Halb)-Ebene. Die Bilder links beziehen sich auf das Bilderset I, die mittleren auf Set II und die Bilder rechts auf das Set III.

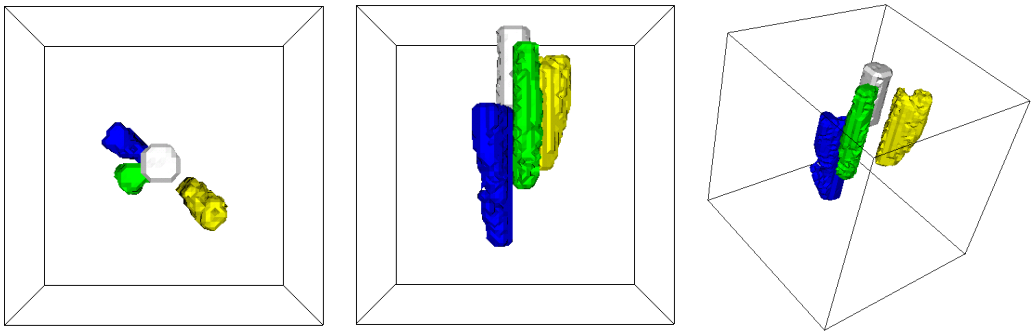
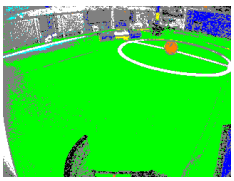
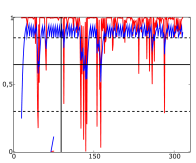


Abbildung 9.4: Die 3-dimensionale Abbildung der Pixelfarbwerte im HSV-Raum für ein ausgewähltes Bild aus dem Bilderset I. Die Farbe der Bereiche entsprechen den Farben der jeweiligen Objektklasse, also etwa grün für die Farbe des Spielfelds. Das linke Bild zeigt eine Draufsicht auf den Kegel des HSV-Farbraums (entsprechend der oberen Reihe in Abbildung 9.3), die mittlere Abbildung die Seitenansicht auf dieselbe Verteilung, der rechte Plot eine isometrische Darstellung, um einen besseren räumlichen Eindruck der Daten zu bekommen.







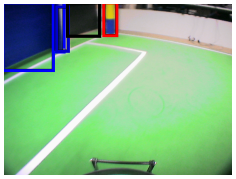
	Hue				Saturation				Brightness			
	G	B	E	O	G	B	E	O	G	B	E	O
						Standardabweichung						
Set I	6,3	4,6	3,2	6,4	7,1	10,0	12,4	47,2	22,8	28,4	15,2	59,1
Set II	3,3	2,8	9,8	11,4	17,6	34,9	13,3	25,7	30,6	37,4	26,4	45,1
Set III	6,9	6,6	8,3	8,3	16,0	13,7	12,1	46,2	31,7	49,7	22,8	64,2
						Gewichteter Mittelwert						
Set I	151	210	44	4	49	62	112	96	157	116	181	167
Set II	193	218	78	352	94	119	62	64	169	175	160	155
Set III	176	206	64	5	67	37	37	99	160	116	85	174

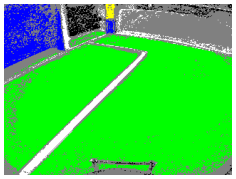
Tabelle 9.1: Gewichtete Mittelwerte und Standardabweichungen über die drei Dimensionen des HSV-Farbraums (Hue, Saturation und Brightness). Jeder Wert ist für die vier verschiedenen Farbklassen ermittelt (G=Grün, B=Blau, E=Gelb, O=Orange).

Für jede Farbverteilung jeder Beleuchtungssituation wurde nun das gewichtete arithmetische Mittel und die dazugehörige Standardabweichung ermittelt. Tabelle 9.1 zeigt diese Werte für die drei verschiedenen Szenarien und jeweils vier verschiedene Farbklassen. Deutlich zu sehen ist, daß sich die Standardabweichung für die verschiedenen Farben erhöht, je schwieriger die jeweilige Beleuchtungssituation wird. Die Standardabweichung für den Hue-Farbwert bleibt über alle Szenarien relativ konstant, aber die Werte für Saturation und besonders Brightness steigt signifikant an. Insbesondere die Werte für das grüne Spielfeld sind hier besonders auffällig. Ebenfalls gut zu erkennen ist, daß sich der Hue-Wert für den blauen Farbton durch die blau eingefärbte Fensterfront nicht wesentlich verändert, wohl aber die Hue-Werte für die meisten anderen Farbwerte.

Räumliche Verteilung der idealisierten Klassifikation

Um zu zeigen, daß die oben angeführten statistischen Werte nicht maßgeblich durch die gewählte Art der Kalibrierung beeinflusst werden, wurden dieselben Werte für einige, von Hand „quasi perfekt“ farbsegmentierte Bilder ermittelt. Für jedes Szenario wurden dazu einige Bilder ausgewählt, welche eine ähnliche Situation auf dem Spielfeld darstellen. Jedes Bild ist dann von Hand maskiert und die Farbwerte den jeweiligen Farbklassen zugeordnet worden. Daraufhin wurden, genau wie oben bereits beschrieben, die den jeweiligen Farbklassen zugeordneten Farbwerte zu einem Farbbereich zusammengefasst.





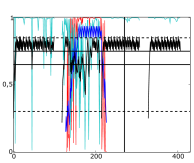


Abbildung 9.5 zeigt die räumliche Verteilung innerhalb der drei verschiedenen Szenarien für die Farbklassen grün (welche den grünen Spielteppich repräsentieren, oben links), blau (eines der Tore, oben rechts) sowie gelb (das andere Tor, unten Mitte). Die dreidimensionale Struktur ist in einer isometrische Darstellung angezeigt. Man beachte, daß die Farben der Farbbereiche diesmal nicht die Zielfarben darstellen, sondern verschiedene Beleuchtungssituationen repräsentieren. Die grünen Bereiche beschrei-

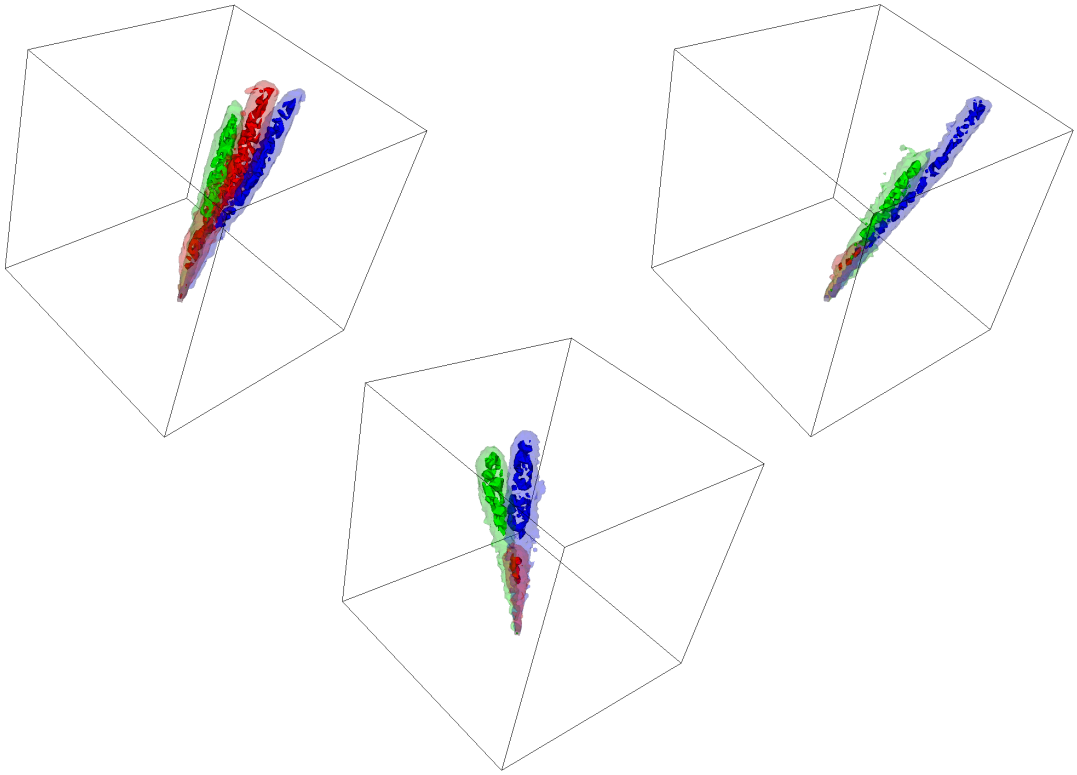
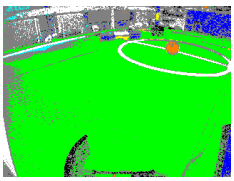


Abbildung 9.5: Dreidimensionale Verteilung derselben Farbklassen unter verschiedenen Beleuchtungsbedingungen innerhalb des HSV-Farbraums. Oben links für die grüne Farbklasse, oben rechts für das blaue Tor und unten Mitte entsprechend für die gelbe Farbklasse.

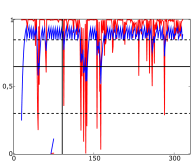
ben die Verteilung unter künstlichem Licht (Set I), der blaue Cluster dasselbe Objekt bei Mittagslicht (Set II) und der rote Bereich die Szenerie bei gerichtetem Sonnenlicht (Set III). Die verschiedenen Transparenzbereiche in der Abbildung symbolisieren verschiedene Iso-Level der Dichte der Verteilungsfunktion (der undurchsichtige Teil besitzt eine höhere Dichte als der halbtransparente Bereich, stellt aber denselben Cluster derselben Farbklasse dar).

Abbildung 9.5 bestätigt die Aussagen der statistischen Werte aus Tabelle 9.1. So sieht man im der ersten Abbildung (oben links, welche die grüne Farbklasse repräsentiert), daß der halbtransparente Bereich des roten Clusters (welcher das Bilderset III beschreibt) im Verlauf der „Brightness“-Dimension einen deutlich größeren Raum einnimmt. Dies reflektiert die extrem unterschiedlichen Helligkeitsbedingungen in diesem Fall. Beachtenswert ist auch, daß dasselbe rote Cluster für die gelbe Farbklasse (unten, Mitte) bei sehr kleinen Brightness-Werten verbleibt. Dieses Phänomen wird weiter unten noch erklärt werden.





Identische statistischen Kennzahlen wie in obiger Tabelle – Standardabweichung der verschiedenen Farbklassen im HSV-Farbraum und gewichteter Mittelwert – wurden auch für diese idealisierten, handgemachten Segmentierungen berechnet; Tabelle 9.2



	Hue				Saturation				Brightness			
	G	B	E	O	G	B	E	O	G	B	E	O
	Standardabweichung											
Set I	6,5	3,9	2,9	3,6	8,3	18,0	12,0	29,5	24,5	26,4	16,4	42,3
Set II	3,6	3,3	6,7	28,6	19,1	40,0	12,1	26,4	32,0	49,7	28,7	50,0
Set III	10,7	13,1	8,4	25,4	18,9	7,9	7,4	59,3	41,9	10,9	9,8	85,1
	Gewichteter Mittelwert											
Set I	151	211	44	7	49	72	115	148	152	124	183	221
Set II	194	218	74	348	92	135	68	63	162	182	166	151
Set III	174	208	65	2	60	25	36	92	149	58	82	173

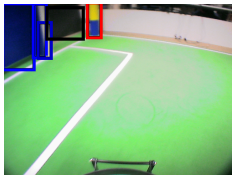
Tabelle 9.2: Gewichtete Mittelwerte und Standardabweichungen über die drei Dimensionen des HSV-Farbraums (Hue, Saturation und Brightness). Jeder Wert ist für die vier verschiedenen Farbklassen ermittelt (G=Grün, B=Blau, E=Gelb, O=Orange).

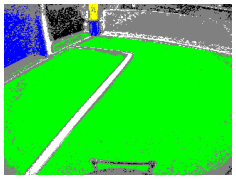
beschreibt diese. Da in diesen idealen Segmentierungen auch die Farbwerte der gelb-blauen Eckpfosten berücksichtigt wurden, ändern sich dadurch die Werte für diese zwei Farbklassen ein wenig. Der Brightness-Wert für die gelbe Klasse bleibt in Set III aber auch hier auf einem sehr kleinen Bereich. Der Hue-Wert für das blaue Tor bleibt ebenfalls in etwa gleich, die übrigen Werte ändern sich hingegen signifikant. Die rechte, obere Darstellung in Abbildung 9.5 spiegelt deutlich den großen Wert für die Standardabweichung für den Fall natürlichen Lichts wieder (der blaue Bereich).

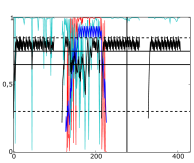
Die Ähnlichkeit der Werte, die aus den idealisierten Segmentierungen gewonnen wurden (Tabelle 9.2), und denen der automatischen Segmentierung (Tabelle 9.1) ist aber deutlich zu erkennen. Das ist ein klares Indiz dafür, daß die beschriebenen Effekte nicht signifikant durch die gewählte Art der Farbkalibrierung beeinflusst wurden oder gar daraus resultieren.

Einige ausgewählte Beispiele

Abschließend möchte ich ein paar Beispiele präsentieren, welche hier als Illustration bestimmter Effekte dienen können. Die ersten Bilder zeigen das Problem distanzabhängiger Variationen der Farbwahrnehmung. In Abbildung 9.6 sind zwei Originalaufnahmen und das jeweils dazugehörige, segmentierte Bild mit Blick auf das gelbe Tor zu sehen (beide unter denselben Bedingungen in unserem Labor aufgenommen; Set III). Das Tor selbst ist nur im zweiten Farbindexbild zu erkennen, bei dem der Roboter näher vor dem Tor steht. Dieses Problem ist aber keines, welches durch eine schlechte Kalibrierung der Farbklassen entstanden ist. Im Hintergrund werden bereits Teile als blau klassifiziert, was deutlich zeigt, daß die Kalibrierung für dunkle Bereiche bereits am Rande des Möglichen arbeitet. Man sieht ebenfalls deutlich, daß die Eckpfosten, welche direkt vom Sonnenlicht angestrahlt werden, in beiden segmentierten Bildern zu erkennen sind. Nicht erkannt wird das Tor, welches durch seine Seitenwände teils deutlich abgeschattet wird.







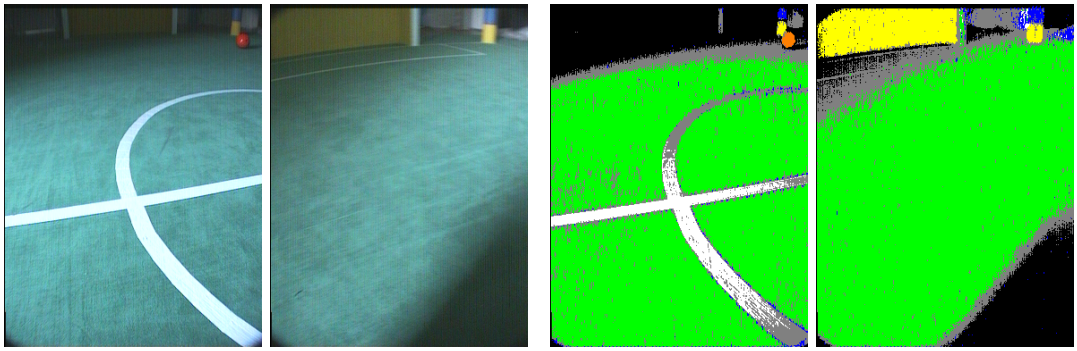
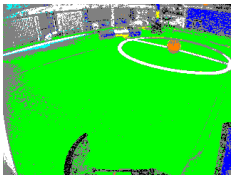


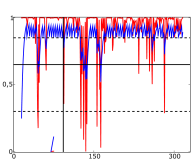
Abbildung 9.6: Beispiel, wie dieselbe Farbe, abhängig von der Aufnahmedistanz, unterschiedlich wahrgenommen wird. Die jeweils linken Bilder zeigen die Originalaufnahmen, die jeweils rechten Bilder die entsprechenden farbindizierten Ergebnisse.

Ein anderes Beispiel illustriert den Einfluß der leicht blau eingefärbten Fenster während unseres Freundschaftsspiels in München. Abbildung 9.7 zeigt zwei Bilder des exakt gleichen Roboters mit exakt demselben Farbmarker vor derselben weißen Absperrung. Das linke Bild ist aus dem Bilderset I (künstliche Beleuchtung) entnommen, das zweite Bild aus Set II (Tageslicht). Der vergrößert dargestellte Ausschnitt zeigt, wie dramatisch sich die Farbwahrnehmung unter wechselnden Bedingungen ändert, speziell der Abstand zwischen Weiß und Cyan. Bei künstlichem Licht beträgt der räumliche, arithmetische Abstand zwischen den RGB-Farbwerten $(255, 255, 255)$ und $(199, 255, 255)$ noch 56 (alle Werte skaliert zwischen $[0 \ldots 255]$). Unter natürlichem Licht (welches durch die erwähnten, blau eingefärbten Fensterscheiben fällt) verringert sich dieser Abstand zwischen dem Wert der weißen Absperrung $(150, 221, 255)$ und dem Ausschnitt des Farbmarkers $(117, 226, 255)$ deutlich auf nur noch 33, bzw. 31 für die anderen Werte. Als direkte Konsequenz waren wir nicht mehr in der Lage, eine Kalibrierung zu erstellen, welche diese beiden Farbklassen noch zuverlässig unterscheiden konnte.

Das letzte Beispiel zeigt, wie eine natürliche Beleuchtung durch die wechselnde Helligkeit innerhalb kürzester Zeit eine einmal gemachte, statische Kalibrierung an ihre Grenzen führt. In Abbildung 9.8 sind zwei Aufnahmen dargestellt, welche im Abstand von nur etwa 14 Minuten aufgezeichnet wurden. Beide Bilder zeigen ungefähr dieselbe Situation auf dem Spielfeld. In der zweiten Aufnahmen im rechten Bild ließ die Sonneneinstrahlung für einen Moment etwas nach. Im linken Bild beträgt der Mittelwert aller grünen Farbwerte des Spielfelds im RGB-Farbraum etwa $(81, 190, 168)$, wohingegen er in der rechten Aufnahme bei $(88, 149, 155)$ liegt. Beachten sie den deutlichen Unterschied im mittleren, grünen Farbwert. Eine statische Kalibrierung funktioniert in diesem Fall nicht mehr für beide, sehr unterschiedlichen Beleuchtungssituationen (insbesondere, da zweiter Fall nicht während der Kalibrierungsphase auftrat und daher nicht in den Lernbeispielen enthalten war). Deutlich zu sehen ist, wie im farbsegmentierten Bild des linken Beispiels praktisch alle Teile des Spielfelds korrekt grün eingefärbt sind, wohingegen im rechten Segmentbild große Teile des Bodens nurmehr







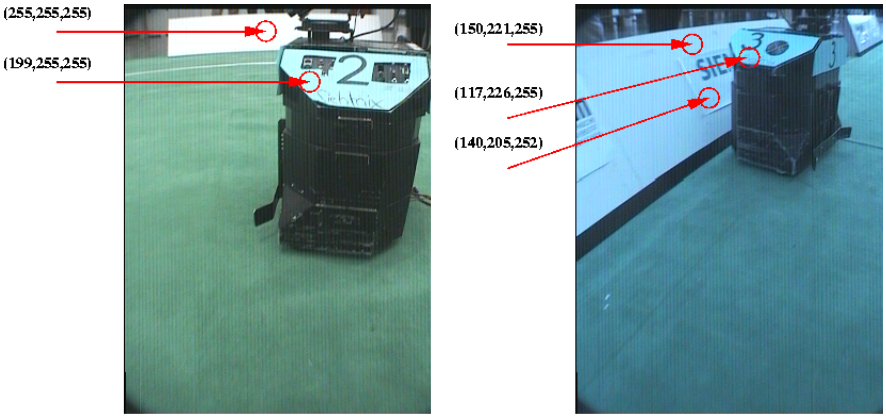
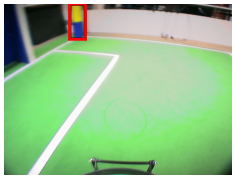


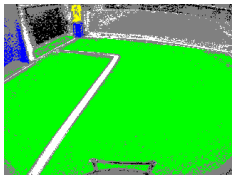
Abbildung 9.7: Beispiel, wie durch äußere Einflüsse die Farbwahrnehmung erschwert werden kann. Das linke Bild zeigt den Roboter (mit cyanfarbenem Farbmarker) vor einer weißen Wand bei Kunstlicht. Das rechte Bild die beinahe gleiche Szene (selber Roboter, selbe weiße Wand) bei natürlichem Tageslicht. Der Abstand der vergrößert dargestellten Ausschnitte verringert sich deutlich durch den Wechsel der Beleuchtungsbedingungen.

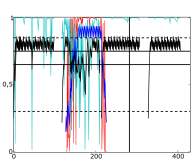
der grauen Klasse (der sogenannten catch-all Klasse, welche alle nicht zugeordneten Farbwerte sammelt) zugeordnet wurden. Dieser Effekt geschah, wie bereits erwähnt, innerhalb von nur 14 Minuten. Im ungünstigsten Fall wäre das innerhalb einer einzigen Halbzeit, ohne die Möglichkeit, die Farbzuordnungstabellen entsprechend zu korrigieren.

9.1.4 Abschließende Bemerkungen

Ausgehend von den obigen Experimenten und ermittelten Werten kann folgendes Fazit gezogen werden. Variationen der Beleuchtungssituation auf dem Spielfeld können einen dramatischen Effekt auf die Wahrnehmung der Farben haben. Oftmals überlappen sich die räumlichen Bereiche zwischen korrespondieren Farbklassen bei unterschiedlicher Beleuchtung nicht einmal mehr (Vergleiche Abschnitt 9.3). Auch wenn sich einige der dargestellten Effekte durch eine variable, automatisch adaptierende Farbkalibrierung ausgleichen lassen werden, sind viele Umstände auch für solche Verfahren eine große Herausforderung. Das Beispiel des sich verkleinernden Farbabstands, wie in Beispiel 9.7 aufgezeigt, betrifft selbstverständlich auch solche Verfahren.







Eine Möglichkeit, diesen Problemen Herr zu werden, ist sicherlich der Einsatz sogenannter Farbkonstanzalgorithmen. Diese Verfahren versuchen, die Farbeindrücke mit Hilfe unterschiedlicher Ansätze auch unter wechselnden Bedingungen konstant zu halten. Beispiele sind in den Arbeiten von Forsyth [35, 36] oder von Jobson, Rahman und Woodell [108, 69] zu finden. Ein Einsatz solcher Verfahren ist aufgrund der algorithmischen Komplexität im Moment aber noch schwierig. Ein von uns untersuchter Ansatz [95] verlagert diesen Berechnungsaufwand deshalb auf die Vorbereitungsphase.

Ein anderer Ansatz ist der in dieser Arbeit Gewählte, nämlich die Objekterkennung

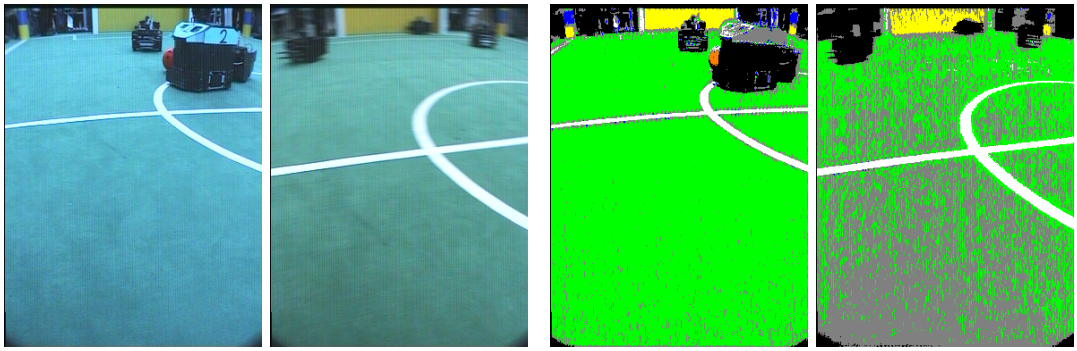


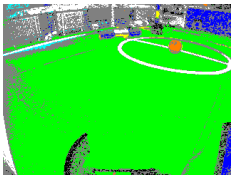
Abbildung 9.8: Beispiel, wie sich die Beleuchtungssituation innerhalb kürzester Zeit ändern kann. Die jeweils linken Bilder zeigen die Originalaufnahmen, die rechten Bilder jeweils die entsprechenden, farbsegmentierten Bilder. Deutlich zu sehen ist, wie im linken Bild die Segmentierung des Spielfelds quasi perfekt funktioniert, im rechten Bild hingegegen große Teile des Spielfelds nicht mehr korrekt klassifiziert werden können.

nicht mehr nur von Farbinformationen abhängig zu machen, sondern sich bei der Entscheidung auch auf andere Informationen zu stützen, wie etwa Größe und Form. Nur wenige andere Gruppen folgten bisher diesem Beispiel wie etwa [53, 54] (siehe auch weiter oben in Abschnitt 6.5).

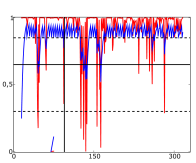
9.2 Regionensuche

Im folgenden soll die Regionensuche für verschiedene Objekttypen ausgewertet werden. Die Auswertung erfolgt dabei auf realen, vom Roboter aufgenommenen Daten. Die Frage stellt sich aber, woher diese Daten genommen werden sollen. Während eines echten Fußballspiels ist der Roboter quasi immer auf den Ball fixiert. Die anderen Objekte tauchen nur eher zufällig in seinem Sichtbereich auf. Erkennt der Roboter einen für ihn unerreichbaren Ball (oder einen Gegenstand, den er für einen Ball hält), dann wird er an den Spielfeldrand fahren und möglicherweise bis zur nächsten Spielunterbrechung vergeblich versuchen, diesen Ball anzufahren. Aufnahmen im Labor sind andererseits aufgrund der dort deutlich kontrollierteren Umgebung aber nicht repräsentativ, insbesondere was unbekannte Objekte am Spielfeldrand betrifft. Als Kompromiß wurden die Aufnahmen eines Feldspielers von der Weltmeisterschaft 2004 in Lissabon verwendet, aus denen teilweise einige, quasi-statische Sequenzen vom Spielfeldrand, beim Warten auf den Anstoß und als er sich eine Zeitlang an einem Eckpfosten verklemmt hatte, aus dem Datenset gelöscht wurden. In der Summe enthält diese Sequenz 825 Bilder.





Im Kampf um den Ball ist dieser oftmals teilweise von anderen Robotern verdeckt. Als Faustregel wurde angenommen, daß ein Ball, der *weniger* als die Hälfte verdeckt ist, als solcher erkannt werden sollte (in der Annahme, daß er später auch vom neuronalen Netz korrekt validiert werden würde). Bälle, die *mehr* als die Hälfte verdeckt sind, sind nicht als solche gewertet worden. Wurde der Ball von der Regionensuche



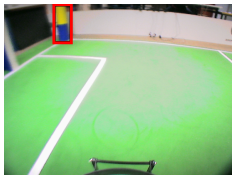
aber trotzdem als solcher erkannt, dann wird dies ebenfalls korrekt bewertet und nicht als falsch positiver Treffer interpretiert. Ausnahmen von diesen Regeln wurden dann gemacht, wenn der Ball wegen seiner (mangelnden) Größe oder aufgrund starker Bewegungsunschärfe auch für einen menschlichen Beobachter quasi nicht mehr als solcher erkannt werden kann. Ähnlich wurden auch die anderen Objekte, wie etwa die Eckpfosten interpretiert.

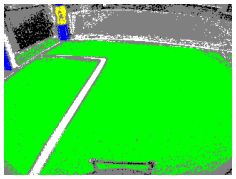
Die Erkennungsleistung der Regionensuche ist natürlich grundlegend von der Auswahl der Farbbereichsgrenzen abhängig. Die folgenden Auswertungen wurde alle mit zwei verschiedenen Kalibrierungen vorgenommen. Die als „Original“ bezeichnete Kalibrierung ist jene, mit der tatsächlich damals 2004 in Lissabon gespielt wurde (die daher auch erklärt, warum sich der Roboter so oft von vermeintlichen Bällen außerhalb des Spielfelds ablenken ließ – eine neuronale Validierung existierte damals noch nicht). Diese Kalibrierung ist insbesondere dadurch charakterisiert, daß für jede Farbklasse nur ein einziger Farbbereich definiert ist (wie auch in Kapitel 5 beschrieben). Im Gegensatz dazu enthält die als „Aktuell“ bezeichnete, zweite Kalibrierung, möglicherweise mehrere Farbbereiche pro Farbklasse. Diese Erweiterung wurde in Zusammenarbeit mit Ulrich Kaufmann entwickelt und ändert nichts an den grundsätzlichen Aussagen zur Regionensuche. Insbesondere ändert sich dadurch das Laufzeitverhalten nicht, da die eigentliche Abbildung von Originalfarben in Farbklassen mittels lookup-Tabelle vorgenommen wird. Diese Tabelle wiederum muß nur einmal beim Programmstart erstellt werden. Die zweite Kalibrierung entspricht in etwa einer, wie wir sie zum jetzigen Zeitpunkt bei diesen Bedingungen erstellen würden.

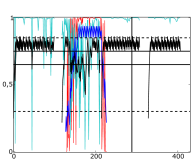
9.2.1 Ballsuche

Wie bereits in Kapitel 5.2 beschrieben, werden gefundene Regionen anschließend über bestimmte Größen- und Füllgradbeschränkungen aussortiert. Im Detail gilt bei der Ballsuche für alle Regionen eine minimale Breite und Höhe von jeweils 3 Pixeln ($x \geq 3$, $y \geq 3$). Die minimale Anzahl von entsprechend klassifizierten, orangenen Pixeln beträgt 5 Stück. Der minimale Füllgrad der gesamten Region mit orangenen Pixeln muß mindestens 30% betragen.

Zusätzlich gibt es Maximalbeschränkungen für die Regionendimension. Unterhalb eines Abstands von 4,5 m wird die Regionenausdehnung über die reale, projizierte Größe des Objekts bestimmt. Hier darf das umschließende Rechteck eine reale Größe von mindestens 120 cm und maximal 380 cm besitzen. Dies entspricht in etwa den Ballwerten, beginnend bei einem halben Balldurchmesser bis hin zum kompletten Durchmesser zuzüglich eines gewissen Sicherheitsabstandes. Bei einer Distanz von mehr als 4 m wird nicht mehr über die reale, projizierte Größe entschieden, sondern über die Pixelwerte. Über 4 m muß die Region kleiner als 14 Pixel sein, über 6 m kleiner als 10 Pixel, über 8 m gar kleiner als 8 Pixel.







Die Erkennungsraten sind in Tabelle 9.3 einzeln aufgeführt. Die unterschiedliche Anzahl möglicher zu erkennender Bälle entsteht dadurch, daß teilweise Bälle doch erkannt wurden, obwohl sie nach obiger Definition nicht hätten erkannt werden müssen.

Demnach wurden für die Originalkalibrierung 404 von 421 Ballregionen korrekt als

Regionensuche – Ball		
Originale Kalibrierung	Positiv	Negativ
Korrekte Entscheidung	404	168
Falsche Entscheidung	1234	17
Aktuelle Kalibrierung	Positiv	Negativ
Korrekte Entscheidung	371	398
Falsche Entscheidung	71	21

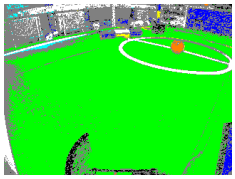
Tabelle 9.3: Verwechslungsmatrizen der Regionensuche für den Ball. Falsch positive Entscheidungen sind in diesem Zusammenhang „erfundene“ Regionen, das heißt Regionen, die in Wahrheit keinen Ball enthalten. Umgekehrt sind falsch negative Entscheidungen Bälle, die von der Regionensuche „übersehen“ wurden.

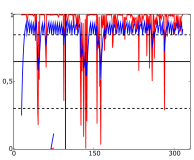
solche erkannt. Das bedeutet eine korrekt positive Erkennungsrate von 96% aller Ballkandidaten, umgekehrt wurden nur 4% aller Bälle in dieser Bearbeitungsstufe nicht als solche erkannt. Daneben wurde für 168 Bilder korrekterweise keine Region ausgewählt, zusätzlich aber in den übrigen Bildern noch 1234 eigentlich uninteressante Regionen gefunden. Bei der aktuellen Kalibrierung wurden 371 von 392 Regionen korrekt als solche erkannt, was einer fast identischen Erkennungsrate von 95% entspricht. Gleichzeitig wurden durch die deutlich exaktere Eingrenzung auf die relevanten Farbanteile aber nur 71 falsch positive Regionen detektiert.

Der größte Teil der falsch negativen Regionen, der nicht gefundenen Bälle, wurde aufgrund einer nicht ausreichenden Größe des Farbblobs wieder verworfen. Jedoch waren im Testdatensatz viele, auch noch kleinere Bälle enthalten. Es bedarf deswegen einer Kombination von kleiner Ballabbildung und ungünstiger Lichtverhältnisse, damit eine solch falsch negative Bewertung zustande kommt. Nur in wenigen Fällen sind die Bälle durch andere Umstände nicht erkannt worden. Abbildung 9.9 zeigt in der oberen Reihe zwei Beispielbilder, einmal ein kleiner, weit entfernter Ball, einmal ein Ball, der durch die Bauteile eines Roboters in zwei Teilbereiche zerlegt wird.

Die falsch positiven Regionen der aktuellen Kalibrierung sind allesamt aufgrund der starken Ähnlichkeit der gelben Tor- und Eckpfostenfarbe zur orangenen Farbe des Balls entstanden. Die untere Reihe in Abbildung 9.9 zeigt ein Beispiel für eine falsch positiv bewertete Region. Der orangene Bereich an der Seitenwand des Tores wurde aufgrund seiner Größe und seines Füllgrades nicht als Ballregion gewertet, sehr wohl aber der kleinere Bereich an der Rückseite des Tores. Dieser passt mit seiner Größe in das einfache Ballschema der Regionensuche. Im Falle der Originalkalibrierung wurden die unterschiedlichsten Objekte irrtümlich als Ballregion interpretiert: am Spielfeldrand herumliegende Kabelhaufen, Hautpartien, Teile der magentafarbenen Robotermarkierungen und so weiter. Beide Beispiele zeigen aber deutlich die Notwendigkeit einer zweiten, weit selektiveren Bearbeitungsschicht für das Gesamtsystem.







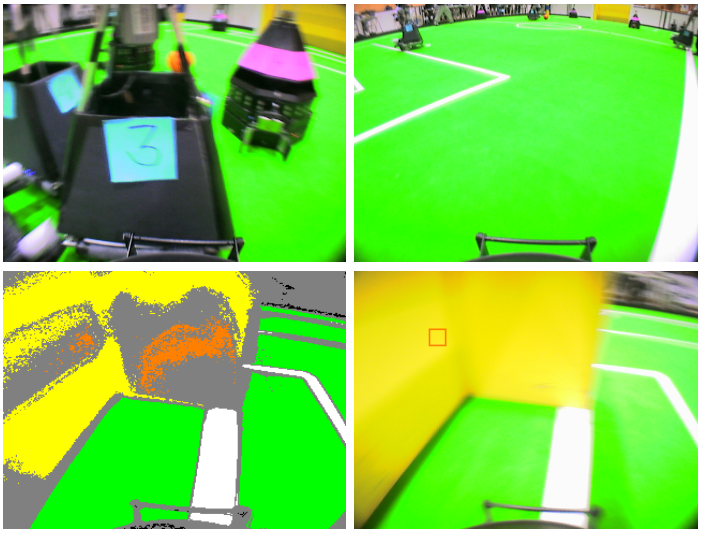
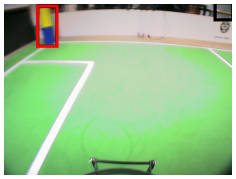


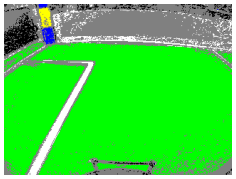
Abbildung 9.9: In der oberen Reihe zwei Beispiele für falsch negative Ergebnisse der Regionensuche. In diesem Fall Bälle, welche die Regionensuche nicht als solche erkannt hat. In der unteren Reihe einmal das indizierte Bild sowie das Originalbild mit eingezeichneter, falsch positiv erkannter Region.

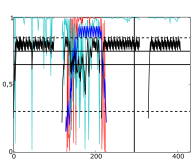
9.2.2 Eckpfostensuche

Was die Suche nach Eckpfosten von der Ballsuche unterscheidet ist, daß gleich zwei Farbklassen korrekt segmentiert, sowie zusätzlich die räumliche Lage von gelben und blauen Bereichen zueinander bewertet werden müssen. Die folgende Auswertung für die Regionensuche nach Eckpfosten wurde mit exakt derselben Farbkalibrierung wie die obige Ballregionensuche durchgeführt. Wie im obigen Abschnitt dargestellt, wird durch die starke Ähnlichkeit von gelber und orangener Farbe gelb teilweise als Ballfarbe interpretiert. Im Umkehrschluß bedeutet das, daß viele Eckpfosten durch diese Verwechslung nicht gefunden werden. Da in realen Spielen die Ballerkennung aber deutlich wichtiger ist als die korrekte Eckpfostenerkennung, wurde auch hier eine möglichst realistische Kalibrierung verwendet.

Entsprechend der Ballsuche im vorherigen Abschnitt gibt es auch bei der Suche nach Eckpfosten einen ganzen Satz von Parametern, welcher unter anderem minimale und maximale Größe der Regionen definiert. Es gilt: $x \geq 2$, $y \geq 3$, Mindestanzahl an farbigen Pixeln ist 5, Mindestfüllgrad 30%. Alle diese Werte beziehen sich *jeweils* auf die blauen bzw. gelben Regionen alleine. Auf die Aufzählung der einzelnen Werte für die Größenoberwerte sowie für erlaubte Werte zwischen den beiden Farbblöcken wird an dieser Stelle verzichtet.







Damit ergibt sich für die Originalversion eine korrekt positive Erkennungsrate von 81%, für die neue Kalibrierung von 60%. Nicht in der Zählung berücksichtigt wurde dabei, ob der Eckpfosten jeweils korrekt als beispielsweise blau-gelb-blauer Pfosten oder umgekehrt als gelb-blau-gelber Pfosten erkannt wurde, da diese Unterscheidung erst an späterer Stelle gesondert überprüft wird. Nach einer qualitativen Prüfung

Regionensuche – Eckpfosten		
Originale Kalibrierung	Positiv	Negativ
Korrekte Entscheidung	463	322
Falsche Entscheidung	15	107
Aktuelle Kalibrierung	Positiv	Negativ
Korrekte Entscheidung	270	424
Falsche Entscheidung	0	174

Tabelle 9.4: Verwechslungsmatrizen der Regionensuche für Eckpfosten. In den oberen Spalten die Werte für die Originalkalibrierung, in den Unteren für die Aktuelle.

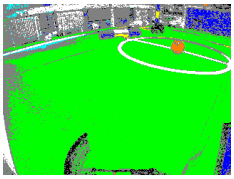
der Daten kann aber festgestellt werden, daß im Falle der aktuellen Kalibrierung beinahe alle Eckpfosten korrekt erkannt wurden, im Originalfall nur etwa $^2/_3$ aller Pfosten. Damit wäre auch hier der Wert der korrekt positiv erkannten Objekte in etwa auf demselben 60%-Niveau. Abhängig von der nachfolgenden Validierung kann dieser Wert aber durch das Aussortieren der falschen Eckpfosten und der zeitlichen Integration immer noch für die Selbstlokalisierung ausreichen.

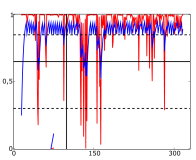
9.2.3 Abschließende Bemerkungen

Da die manuelle Interpretation und Kontrolle mehrerer tausend Bilder sehr mühsam ist, soll die Untersuchung zweier Objekttypen an dieser Stelle genügen. Auch so lassen sich bereits einige Schlussfolgerungen ziehen: Hypothesen für einfache Objekte wie etwa den Ball lassen sich mit hoher Präzision finden. Besteht die Möglichkeit, Farben mittels mehrerer Bereichsgrenzen zu definieren, hält sich auch die Menge an falsch positiven Objekthypothesen in engen Grenzen, ohne dabei die korrekt positive Erkennungsrate zu beeinträchtigen. Nach subjektivem Empfinden ist das manuelle Einstellen dieser multiplen Bereichsgrenzen für den Anwender überdies auch wesentlich einfacher. Im Umkehrschluß bedeutet dies aber auch eine möglicherweise größere Anfälligkeit für wechselnde Beleuchtungsbedingungen (vgl. dazu auch den vorherigen Abschnitt).

Für Objekte, die aus mehreren, zusammengesetzten Farbbereichen bestehen, konnte eine solch hohe Erkennungsrate nicht mehr erreicht werden. Die im obigen Eckpfostenbeispiel gezeigten Werte sind mit Sicherheit auch aufgrund der damaligen schwierigen Beleuchtungsbedingungen entstanden. In vielen Bildern waren große Teile der (eigentlich) gelben Bereiche mit roten Bereichen durchsetzt und wurden deshalb nicht mehr korrekt als solche (gelben) Bereiche erkannt. Es gab aber auch viele Eckpfosten, die trotz korrekter Farbindizierung keine zugeordnete Hypothese erhielten. Die riesige Variabilität der Erscheinungsformen und die Kameraverzerrung bei Nahaufnahmen machen eine Auswahl der Testparameter für die Bewertung der räumlichen Lage extrem schwierig. Eventuell sollte man Parameter bereits an dieser Stelle besser lernen, als sie manuell vom Entwickler vorzugeben. Eine große Auswahl an Bildern unterschiedlichster Erscheinungsformen existiert bereits.







9.3 Objektklassifikation

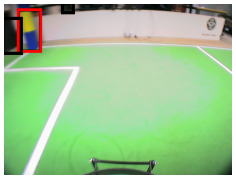
In der hier vorgestellten Arbeit werden aktuell drei verschiedene Objekttypen durch einen neuronalen Klassifikator validiert: Der Ball, die Eckpfosten und eigene und gegnerische Roboter. Auch wenn die Erkennung der Roboter für das eigentliche Fußballspiel nicht die wichtigste Aufgabe darstellt, sind sie aus Sicht der Objektklassifikation doch mit Abstand der spannendste Objekttyp. Die besonders hohe Formenvielfalt der möglichen Roboterarten erzwingt eine besondere hohe Generalisierungsleistung der neuronalen Klassifikatoren. Es ist quasi unmöglich, von allen im RoboCup teilnehmenden Mannschaften und ihren Robotern geeignete Testbilder aufzunehmen. In der Trainingsphase steht daher nur ein Teilbereich der tatsächlich vorkommenden Robotertypen zur Verfügung. Ein geeigneter neuronaler Klassifikator muß daher auch mit dieser begrenzten Anzahl von Trainingsmaterial eine möglichst gute Erkennungsleistung im späteren Einsatz erreichen.

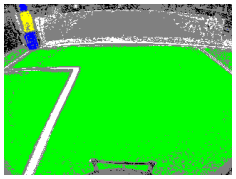
Aus diesem Grund wird die Robotererkennung im folgenden besonders ausführlich untersucht. Nach einer ausführlichen Erklärung der verwendeten Parametrisierung und der Beschreibung der Lern- und Trainingsphase folgt die Auswertung der Erkennungsleistung. Im Anschluß daran werden dann in deutlich kürzerer Form noch die Erkennungsleistungen für Ball und Eckpfosten ermittelt.

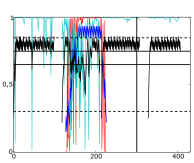
9.3.1 Parametrisierung

Wie üblich in einem Robotersystem, besonders wenn es zusammen mit künstlichen neuronalen Netzen verwendet wird, gibt es ein große Anzahl von Parametern, die passend eingestellt werden können und müssen.

Das wichtigste Merkmal im Zusammenhang mit der Robotererkennung sind die Orientierungshistogramme. Im folgenden werden wir uns die hier möglichen Parameter genauer anschauen: Je größer die Überlappung der einzelnen Unterfenster für die Histogrammberechnung ist, desto geringer sind Fehler durch ROIs, welche nicht exakt auf dem zu untersuchenden Objekt liegen (bezüglich ihrer Größe oder ihrer Position). Auf der anderen Seite werden die Resultate dadurch für die einzelnen Unterbereiche weniger spezifisch. Die Anzahl dieser Unterbereiche entscheidet, wie exakt die Form des Roboters repräsentiert werden kann. Die Anzahl dieser Unterbereiche ist daher wiederum eine Abwägung zwischen Spezifität und Generalizität. Wenn viele der möglichen Robotertypen bekannt sind, kann eine große Zahl an Unterfenstern sinnvoll sein, da dann die Formrepräsentation möglichst exakt wird. Bei einer kleinen Zahl an bekannten Roboterinstanzen wird man lieber eine kleine Zahl an Unterfenstern wählen, um eine größtmögliche Flexibilität zu erreichen. Genau das gleiche gilt für die Anzahl der Histogrammwerte, welche die vorkommenden Orientierungsrichtungen innerhalb eines Unterfensters darstellen. Auch hier kann zwischen Flexibilität und Erkennungsgenauigkeit abgewogen werden.







Der zweite Ansatzpunkt für die Parametrisierung sind die neuronalen Netzwerke. Hier sind die Anzahl der Eingabeneuronen durch die Länge des Merkmalsvektors und die Anzahl der Ausgabeneurone durch die Art der gewünschten Entscheidung (in

diesem Fall eine Binäre, daher ein Ausgabeneuron) festgelegt. Frei wählen läßt sich aber die Anzahl der Neuronen in der mittleren Schicht. Je mehr Neuronen in dieser Zwischenschicht liegen, desto besser kann ein neuronales Netz die präsentierten Trainingsmerkmale lernen. Hierbei steigt aber das Risiko eines sogenannten Overfittings. Das Netzwerk lernt dabei die Trainingsdaten mehr oder weniger auswendig, ohne sich auf die wesentlichen Eigenschaften der Merkmale einzustellen [56, 110]. Das Ergebnis ist ein Verlust an Plastizität und eine schlechte Generalisierungsleistung, was die Erkennung von Roboterinstanzen, die nicht im Trainingsmaterial vorhanden waren, deutlich verschlechtert. Zusätzlich dazu steigt der Berechnungsaufwand für Netze mit vielen Neuronen in der Zwischenschicht auch deutlich an, da in der verwendeten Netzarchitektur immer alle Neuronen aller Schichten mit allen Neuronen der anderen Schichten verbunden sind. Der Aufwand wächst quadratisch mit der Größe der mittleren Schicht. Konkret werden die folgenden Werte für die jeweiligen Parameter der Orientierungshistogramme und der Anzahl der Neuronen in der Zwischenschicht verwendet:

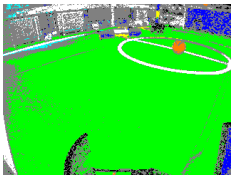
- Überlappung der Unterfenster: 30%, 20%, 10% oder 0% (keine Überlappung).
- Anzahl der Unterfenster: 2, 3 oder 4 (in beiden Richtungen).
- Anzahl der Orientierungswerte: 6, 8, 10 oder 12.
- Größe der Zwischenschicht: Identisch zur Anzahl der Eingabeneuronen, die Hälfte, ein Drittel oder ein Viertel.

Die Angabe der Zwischenschichtgröße bezieht sich auf das neuronale Netz für die Bewertung des Orientierungshistogramms. Für das andere Netz, welches die sogenannten „einfachen Merkmale“ wie die Größe der ROIs bewertet, werden die Parameter weiter unten benannt.

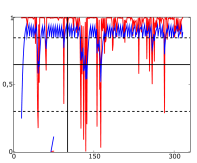
9.3.2 Training

Da in diesem Kapitel nur die Leistungsfähigkeit der Objekterkennung untersucht werden soll, wurden die Objekthypothesen, die ROIs aller Bilder, die im folgenden für das Training der Klassifikatoren verwendet werden, von Hand ermittelt. Im Trainingsmaterial waren Bilder von sechs verschiedenen Robotertypen mit deutlich unterschiedlichem Aussehen enthalten. Bilder dieser unterschiedlichen Roboter waren bereits in Abbildung 2.2 zu sehen. Die Daten enthielten Roboter der Abteilung AIS des Fraunhofer Instituts (84 ROIs), des Teams Trackies der Universität Osaka (138), des Teams Attempto Tübingen (116), von Clockwork Orange der Universität Delft (148), Mostly Harmless Graz (102), unserer eigenen Ulm Sparrows Roboter (94) sowie 656 Bilder weiteren negativen Trainingsmaterials (Regionen, in denen *kein* Roboter vorkam – 164 sorgfältig ausgesuchte Regionen mit Beispielen für wahrscheinliche Verwechsler (z.B. schwarze Kisten am Spielfeldrand) sowie 492 zufällig ausgewählte Bereiche).





In der Trainingsphase wurden fünf der sechs Robotertypen ausgewählt und davon 90% der Regionen entnommen. Zusätzlich wurden ebenfalls 90% der negativen



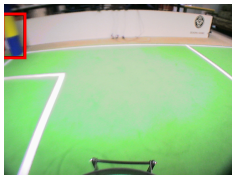
Beipiele dazugemischt. Mit diesen Daten wurde das neuronale Netz trainiert. Diese Trainingsphase wurde dreimal für jede Kombination von Robotertypen und Parametern wiederholt, um den Einfluß der zufälligen Initialisierung zu Trainingsbeginn zu eliminieren. Danach wurde eine komplette Verwechslungsmatrix der Erkennungsleistung der trainierten Netzwerke ermittelt. Es wurden Werte für die Leistungen auf den Trainingsdaten (ein Indiz für die prinzipielle Lernfähigkeit der Netze), auf den Evaluationsdaten, den restlichen 10% der noch unbekannten ROIs (ein Indiz für die Generalisierungsleistung auf den bekannten Robotertypen), sowie auf den Daten des nicht zum Training verwendeten (sechsten) Robotertyps (als Maß für eine noch weitergehende Generalisierungsleistung) berechnet.

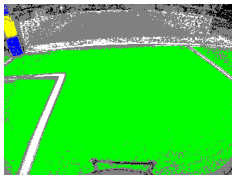
Die Klassifikationsergebnisse sind zusätzlich 10-fach kreuzvalidiert. Dabei wird der oben beschriebene Ablauf 10 Mal mit jeweils neuen, zufälligen 10% - 90% Aufteilungen wiederholt und die Ergebnisse gemittelt. Damit fallen auch hier zufällige, extrem positive oder besonders schlechte Aufteilungen nicht mehr ins Gewicht.

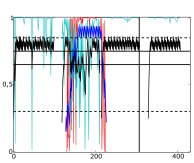
9.3.3 Klassifikationsergebnisse

Die ersten Klassifikationsergebnisse, die präsentiert werden sind für jenes Netzwerk, welches für die Orientierungshistogramme zuständig ist. In Tabelle 9.5 sind die durchschnittlichen Werte über alle möglichen Parametrisierungen für die verschiedenen Datensätze (der Trainingsdatensatz sowie der Evaluationsdatensatz) aufgelistet. Da das neuronale Netz einen Wahrscheinlichkeitswert zwischen Null und Eins ausgibt, wird eine positive Antwort dann angenommen, wenn der Wahrscheinlichkeitswert über 75%, das heißt $\geq 0{,}75$ beträgt (der Wert, der auch für das abschließende Arbitrierungsmodul verwendet wird, wie es in Abschnitt 3.1.3 beschrieben wurde). Der jeweils linke, obere Wert in der Tabelle zeigt den Wert, der korrekt als positiv erkannten Roboterhypothesen, wohingegen der rechte, untere Wert in der Tabelle, die verbleibende Anzahl an fälschlicherweise als falsch klassifizierten Regionen darstellt, obwohl in diesen Aufnahmen tatsächlich ein Roboter enthalten war. Dasselbe gilt entsprechend für die anderen Zellen der Tabelle, für das korrekte Erkennen eines nicht enthaltenen Roboters innerhalb der präsentierten ROIs bzw. einer falschen positiven Antwort auf diesen ROIs.

Der erste Versuch testet, wie sich das künstliche neuronale Netz verhält, wenn ihm dieselben Daten präsentiert werden, die für das Training selbst verwendet wurden. Die Ergebnisse sind im oberen Teil der Tabelle 9.5 dargestellt. Die Ergebnisse sind gemittelte Werte für alle möglichen, oben aufgeführten Parametereinstellungen. Gut zu sehen ist, daß das Netzwerk bereits über diesen großen Bereich in der Lage ist, die präsentierten Trainingsdaten fast perfekt zu lernen. Der zweite Teil der Tabelle beschreibt die ermittelte Erkennungsleistung auf den Evaluationsdaten, also dem Teil der Datensätze, welche *nicht* für das Training verwendet wurden. Auch wenn die Leistung hier nicht mehr für alle Parametereinstellungen optimal ist, zeigt sich bereits, daß die Summe aller Netzwerke eine hohe Generalisierungsleistung erbringt.







Man erinnere sich, daß das Netz jeweils abwechselnd nur mit 5 der 6 möglichen Robotertypen trainiert wurde. Wir können nun denjenigen Parametersatz suchen, für

Roboterklassifikation – Orientierungen (I)		
Trainingsdaten	Positiv (%)	Negativ (%)
Korrekte Entscheidung	98,5	99,2
Falsche Entscheidung	0,8	1,5
Evaluationsdaten	Positiv (%)	Negativ (%)
Korrekte Entscheidung	91,6	92,9
Falsche Entscheidung	7,1	8,4

Tabelle 9.5: Verwechslungsmatrizen für die unterschiedlichen Datensätze, welche dem künstlichen neuronalen Netz präsentiert werden, das für die Bewertung des Orientierungshistogramms zuständig ist. Die Ergebnisse sind über alle möglichen Kombinationen der verschiedenen Parameter gemittelt.

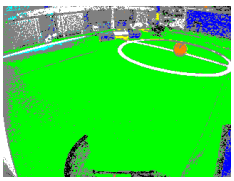
Roboterklassifikation – Orientierungen (II)		
Bekannte Robotertypen	Positiv (%)	Negativ (%)
Korrekte Entscheidung	92,6	94,8
Falsche Entscheidung	5,2	7,4
Unbekannter Robotertyp	Positiv (%)	Negativ (%)
Korrekte Entscheidung	86,7	-
Falsche Entscheidung	-	13,3

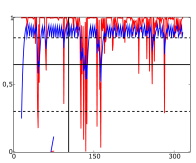
Tabelle 9.6: Verwechslungsmatrizen bei Verwendung der bestmöglichen Parameter für das neuronale Netz des Orientierungshistogramms. Die Ergebnisse sind über alle möglichen Permutationen der Robotertypen gemittelt.

den sowohl die Erkennungsleistung für die bekannten Robotertypen (sowohl aus dem Trainingsdatensatz als auch dem Evaluationsdatensatz) sowie auch für den sechsten, unbekannten Robotertyp am besten ist. Tabelle 9.6 zeigt die Resultate für diese, als Beste bestimmte Parametereinstellung. Im Einzelnen sind diese Parameter wie folgt: Überlappung der Unterfenster 10%, Anzahl der Unterfenster in beiden Richtungen: 3, Anzahl an Orientierungswerten: 10, Anzahl an Neuronen in der Zwischenschicht: 45 (entspricht der Hälfte der Anzahl der Eingabeneuronen). Es gab allerdings auch einige andere, recht ähnliche Parametereinstellungen, die nur geringfügig schlechter abgeschnitten haben.

Die Ergebnisse für das neuronale Netz, welches die einfacheren, geometrischen und farbbasierten Merkmale klassifiziert, lassen sich wesentlich leichter ermitteln. Der einzige Parameter, der hier verändert werden kann, ist die Anzahl der Neuronen in der Zwischenschicht. Entsprechend den Möglichkeiten für das Histogrammnetzwerk wurden die Versuche hier mit 3, 5, 10 und 15 Neuronen durchgerechnet. Da die Unterschiede in der Erkennungsleistung marginal waren, wurden letztendlich 3 Neuronen in der mittleren Schicht verwendet. Die Ergebnisse dafür sind in Tabelle 9.7 zu sehen.







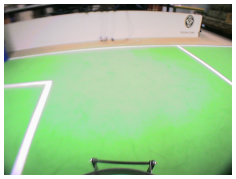
Roboterklassifikation – Einfache Merkmale		
Bekannte Robotertypen	Positiv (%)	Negativ (%)
Korrekte Entscheidung	95,0	95,9
Falsche Entscheidung	4,1	5,0
Unbekannter Robotertyp	Positiv (%)	Negativ (%)
Korrekte Entscheidung	90,4	-
Falsche Entscheidung	-	9,6

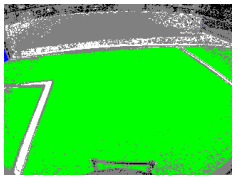
Tabelle 9.7: Verwechslungsmatrizen für das neuronale Netz, welches für die einfachen Merkmale zuständig ist. Die Ergebnisse sind über alle möglichen Permutationen der Robotertypen gemittelt.

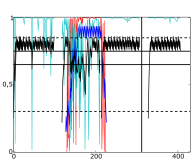
Roboterklassifikation – Gesamtsystem		
Bekannte Robotertypen	Positiv (%)	Negativ (%)
Korrekte Entscheidung	88,2	99,5
Falsche Entscheidung	0,52	11,8
Unbekannter Robotertyp	Positiv (%)	Negativ (%)
Korrekte Entscheidung	78,9	-
Falsche Entscheidung	-	21,1

Tabelle 9.8: Verwechslungsmatrizen für die gesamte Robotererkennung durch beide neuronalen Netze sowie die abschließende Arbitrierung. Die Werte wurden für den als das Beste ermittelten Parametersatz sowie die gewählten Roboterpermutationen und einen unbekannten Robotertyp ermittelt.

Abschließend folgt nun das Gesamtergebnis für beide Netze und der anschließenden Arbitrierung. Training und Parametereinstellungen sind jeweils identisch zu den obengenannten Resultaten. Wiederum wurde das Training jeweils mit fünf Roboterklassen durchgeführt und jeweils ein Typ nur bei der Evaluation verwendet. Diesmal wurden aber nicht die Einzelresultate der beiden neuronalen Netze gewertet, sondern das Gesamtergebnis *nach* der Arbitrierungsinstanz. Nur wenn – im positiven Fall – *beide* neuronalen Netze eine Ausgabe überhalb der Schwelle von 0,75 liefern, wird auch das Gesamtergebnis als positiv gewertet. Umgekehrt reicht es im negativen Fall, daß bereits dann ein negatives Gesamtergebnis vorliegt, wenn *mindestens ein* Netz einen Wert unterhalb von 0,75 zurückgibt.







Da bereits ein negatives Votum ausreicht, um ein negatives Gesamturteil zu erreichen, ist zu erwarten, daß die Anzahl der falsch-positiv erkannten Objekte deutlich sinkt. Wie in Tabelle 9.8 zu sehen ist, ist dies auch tatsächlich der Fall. Hier sind fast keinerlei Fehlklassifikationen mehr vorhanden. Zusätzlich ist aber auch zu sehen, daß die Gesamtanzahl an korrekt klassifizierten Regionen, seien sie nun positiv oder negativ, stark sinkt. Leider existieren noch nicht genügend Anwendungen, die eine

Robotererkennung benötigen, um abschließend sagen zu können, ob ein solch niedriger falsch-positiver Wert sinnvoll ist, oder doch eine höhere korrekt-positive Rate wünschenswerter wäre. Im Falle der unten gezeigten Eckpfostenerkennung ist aber genau eine solche, sehr niedrige falsch-positive Erkennungsrate wichtig, da das gewählte Verfahren mit falschen Angaben schlechter umgehen kann, als mit selten, aber korrekt erkannten Objekten.

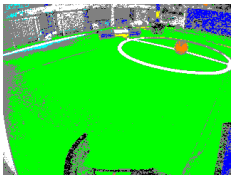
9.3.4 Ballerkennung

Die Leistungsfähigkeit der Ballerkennung soll nun im folgenden in wesentlich kompakterer Form beschrieben werden, als die gerade bewertete Roboterklassifikation. Das Training entspricht der oben dargestellten Methode. Dieses Training jedes einzelnen Netzes wurde 3 Mal wiederholt und das Netz mit dem jeweils kleinsten mittleren quadratischen Fehler weiterverwendet. Einflüsse der zufälligen Gewichtsinitialisierung zu Trainingsbeginn sollten so weitgehend ausgeschlossen sein. Alle Versuche wurden zusätzlich 10-fach kreuzvalidiert, das heißt 10 Mal wurden Trainings- und Evaluationsdaten zufällig aus der Gesamtmenge der vorliegenden Bilder ausgewählt und damit das neuronale Netz trainiert. Damit sollten die Ergebnisse auch gut gegen zufällige Ausreisser in den verschiedenen Datensätzen abgesichert sein.

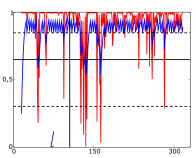
Zur Ballerkennung werden wiederum zwei getrennte neuronale Netze verwendet, die anschließend über einen einfachen Schwellenwertvergleich arbitriert werden. Eines der neuronalen Netze validiert dabei die kombinierten Orientierungs-/Krümmungshistogramme[1], das andere die Verteilung der Farbwerte in den einzelnen Unterbereichen. Im Einzelnen waren die besten Parameter für den Orientierungsvektor: Überlappung der Unterfenster 10%, Anzahl der Unterfenster in beiden Richtungen: 3, Anzahl der Orientierungswerte: 10, Größe der Gaußfunktion für die Krümmungsberechnung: 11, Anzahl der Neuronen in der Zwischenschicht: 225 (dies entspricht einem Viertel der Anzahl der Eingabeneuronen). Für das Farbhistogramm verwendet wurden 20% Überlappung der Unterfenster, zwei Unterfenster in beiden Richtungen, sowie eine 8-fache Unterteilung der einzelnen Farbwerte. Die Anzahl der Neuronen in der Zwischenschicht beträgt 24, damit wiederum ein Viertel der Anzahl der Eingabeneuronen. Im Gegensatz zur Parametrisierung der Netze für die Robotererkennung war hier bei der Auswahl der Parameter für die Orientierungs-/Krümmungshistogramme eine deutlich größere Varianz der Erkennungsleistung zu beobachten.

Der gesamte Datensatz umfaßte dieses Mal 1191 positive Bilder, also Bilder, die einen Ball enthielten. Außerdem 494 negative, teils zufällig ausgewählte Bildbereiche ohne Ball. Tabelle 9.9 zeigt nun die Ergebnisse der einzelnen Netze. Fast durchgängig sind sehr gute Erkennungsraten von weit über 90% zu erkennen. Insbesondere die Leistung auf den Farbhistogrammen ist beinahe perfekt. Nur die korrekt negativ validierten Orientierungs-/Krümmungshistogramme fallen etwas gegen die anderen Werte ab. Das Gesamtergebnis nach der abschließenden Arbitrierung ist in Tabelle 9.10 zu sehen. Wie erwartet, und wie auch bei der vorherigen Robotererkennung zu





[1] Siehe hierzu bitte auch die Erklärung in Abschnitt 6.1.3 auf Seite 56



Ballklassifikation – Einzelnetze		
Orientierungsmerkmale	Positiv (%)	Negativ (%)
Korrekte Entscheidung	96,6	86,4
Falsche Entscheidung	13,6	3,4
Farbmerkmale	Positiv (%)	Negativ (%)
Korrekte Entscheidung	99,7	99,1
Falsche Entscheidung	0,9	0,3

Tabelle 9.9: Verwechslungsmatrizen der beiden Teilnetze für die Ballerkennung mit den jeweils optimalen Parametereinstellungen. Die Erkennungsrate wurde auf Basis der jeweiligen Evaluationsdatensätzen ermittelt.

Ballklassifikation – Gesamtsystem		
	Positiv (%)	Negativ (%)
Korrekte Entscheidung	96,4	99,5
Falsche Entscheidung	0,5	3,6

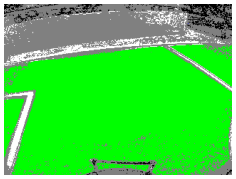
Tabelle 9.10: Verwechslungsmatrizen für die gesamte Ballerkennung durch beide neuronalen Netze sowie die abschließende Arbitrierung. Die Werte wurden für den als Besten ermittelten Parametersatz errechnet.

sehen, ist die Kombination beider Netze bei der korrekten Klassifikation der negativen Beispiele den Einzelnetzen überlegen. Die Anzahl der korrekt negativ erkannten Objekte steigt, gleichzeitig fällt die Anzahl der korrekt positiv erkannten Objekte nur unwesentlich.

9.3.5 Eckpfostenerkennung

Abschließend folgt nun die Ergebnisse für die Eckpfostenklassifikation. Wiederum wurde hier 3-fach trainiert sowie 10-fach kreuzvalidiert, diesmal auf einer Gesamtanzahl von 2818 positiven und 620 negativen Beispielbildern. Die Parameter für die Einzelnetze sind entsprechend für das Orientierungs-/Krümmungshistogramm: 20% Überlappung, 3×3 Unterfenster, 6 Orientierungswerte, 13 als Größe des Kernels zur Strukturerkennung und 108 Neurone in der verdeckten Schicht. Für das Farbhistogramm 0% Überlappung, 3×3 Unterfenster, 12 Werte für die Farbabstufungen sowie 40 Neuronen in der Zwischenschicht.





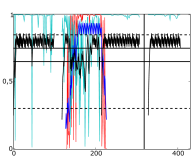


Tabelle 9.11 zeigt die Verwechslungsmatrix für die Einzelnetze. Die Leistungsfähigkeit der Farbhistogramme ist wieder vergleichbar mit den obigen Werten für die Orientierungswerte. Diesmal können aber viele der positiven Beispiel nicht mehr korrekt als solche klassifiziert werden. Die Werte der Gesamtleistung in Tabelle 9.12 zeigen erneut eine quasi perfekte Leistung, ohne dabei die korrekt positive Erkennungsleistung wesentlich zu beeinflußen.

Eckpfostenerkennung – Einzelnetze		
Orientierungsmerkmale	Positiv (%)	Negativ (%)
Korrekte Entscheidung	87,1	96,4
Falsche Entscheidung	3,6	12,9
Farbmerkmale	Positiv (%)	Negativ (%)
Korrekte Entscheidung	99,1	98,3
Falsche Entscheidung	1,7	0,9

Tabelle 9.11: Verwechslungsmatrizen der beiden Teilnetze für die Eckpfostenerkennung mit den jeweils optimalen Parametereinstellungen. Die Erkennungsrate wurde auf Basis der jeweiligen Evaluationsdatensätze ermittelt.

Eckpfostenerkennung – Gesamtsystem		
	Positiv (%)	Negativ (%)
Korrekte Entscheidung	87,0	100,0
Falsche Entscheidung	0,0	13,0

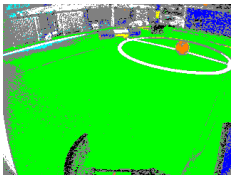
Tabelle 9.12: Verwechslungsmatrizen für die gesamte Eckpfostenerkennung durch beide neuronalen Netze sowie die abschließende Arbitrierung. Die Werte wurden für den als Besten ermittelten Parametersatz errechnet.

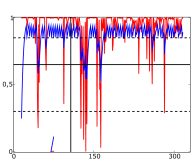
9.3.6 Abschließende Bemerkungen

Alle oben aufgeführten Ergebnisse bieten ein ausgesprochen gutes Bild von der Leistungsfähigkeit der neuronalen Klassifikatoren. Insbesondere, spricht für eine hohe Spezifität der verwendeten Methoden, daß beinahe keine falsch positiven Beispiele erkannt werden. Nur die korrekt positive Bewertung der Eckpfosten fällt hier ein wenig ab. Da die verwendete Methode zur Selbstlokalisation des Roboters aber mit fehlenden Merkmalen deutlich besser zurecht kommt als mit falschen Merkmalen, ist auch dieser Wert noch ausreichend für die gestellte Aufgabe. Die eigentliche Vorgehensweise ist daher dem Szenario angemessen.

Zugegeben ist die Erkennungsrate in realen Spielen aber nicht ganz so hoch wie hier dargestellt. Dies hat mit Sicherheit mehrere Ursachen. Einerseits ist es sehr schwierig, eine ausreichende Anzahl an verschiedenen Beleuchtungsbedingungen zu sammeln. Der Farbbildersatz enthielt hier zum Beispiel hauptsächlich zwei verschiedene, nämlich Neon- und Tageslicht. Zu einer vollständigen Sammlung aller möglichen Beleuchtungen fehlen aber natürlich noch einige Bedingungen. Zusätzlich wurde versucht, möglichst viele Bilder von echten Wettkampfspielen zu benutzen, um ein realistisches Abbild des tatsächlichen Einsatzgebietes zu erhalten. Da während eines solchen Turniers die Beleuchtungsbedingungen aber immer konstant bleiben, ist auch hier keine weitere Variation zu erwarten. Außerdem ändert sich damit auch die gesamte Kameraparametrisierung für die vielfältigen Einstellmöglichkeiten wie Weißabgleich,





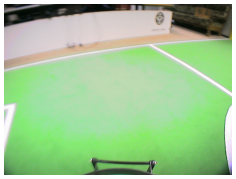


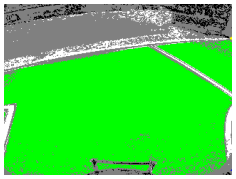
Farbsättigung oder Verschlußgeschwindigkeit nicht. Diese Einstellungen beeinflussen nicht nur die Farbwahrnehmung, sondern ebenfalls die gesamte Qualität der Aufnahmen und damit auch zum Beispiel die Kantenwahrnehmung. Das wiederum hat direkten Einfluß auf die Klassifikationsleistung der Netze auf den Orientierungs- und Krümmungshistogrammen.

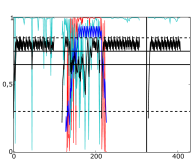
Leider sagen gemittelte Auswertungen über alle Trainings- und Evaluationsdaten wie die oben ermittelten Verwechslungsmatrizen auch nichts über die Verteilung der Fehler innerhalb der Datensätze aus. So ist nicht klar, ob die falsch klassifizierten Beispiele gleichmäßig über den ganzen Datensatz verteilt sind, oder ob es spezielle Bereiche, etwa einen bestimmten Entfernungsbereich gibt, in dem die Objekte immer falsch erkannt werden. So sind unter den nicht korrekt positiv erkannten Eckpfosten besonders viele aus dem Nahbereich dabei, da hier die Verzeichnung durch die Linsenkrümmung besonders stark ist. Ein weiterer Ausbau der Trainingsdatenbasis würde hier wohl noch etwas Verbesserung bringen. Die andere Möglichkeit, nämlich zeitliche Informationen zu verwenden, wird im folgenden Abschnitt untersucht.

9.4 Objektverfolgung, -prädiktion und Informationsintegration

Der gesamte Bereich Objektverfolgung, -prädiktion und Informationsintegration gliedert sich in zwei Bereiche. Zum einen muß die Vorhersagegenauigkeit der ermittelten Trajektorien bestimmt werden, zum anderen die Verbesserungen, die sich durch die so gewonnenen Informationen bezüglich des zeitlichen Verhaltens ergeben. Für die Analyse der Prädiktionsleistung ist es wichtig, daß reale Aufnahmen des Roboters verwendet werden. Künstlich generierte Bilder, selbst wenn sie anschließend stark verrauscht sind, können nie die wirklichen Verhältnisse wie unstetige Beschleunigungen oder das Erkennungsrauschen nachbilden. Für echte Kameraaufnahmen stehen aber leider nicht die tatsächlichen Werte für Position, Geschwindigkeit und Beschleunigung zur Verfügung. Es ist daher nicht möglich, die gesammelten Daten mit echten, sogenannten ground-truth Daten zu vergleichen. Außerdem stehen in diesem Fall keine Sequenzen aus echten Wettbewerbssituationen zur Verfügung. Die in heute üblichen Laptops verbauten Festplatten sind zu langsam, um auch nur jedes zweite von der Kamera aufgenommene Bild in Echtzeit zu speichern. Solche Sequenzen mit voller zeitlicher Auflösung müssen daher im Hauptspeicher des Laptops vorgehalten werden. Die während echter Spiele nötigen laufenden Prozesse verhindern aber leider eine solche Speichernutzung. Aus diesem Grund sind im folgenden Sequenzen verwendet, die im Labor unter kontrollierteren Bedingungen aufgenommen wurden. An der prinzipiellen Aussage ändert sich dadurch jedoch nichts. Die Sequenzen sind sorgfältig so gewählt, daß sie realen Spielsituationen möglichst orginialgetreu entsprechen.







9.4.1 Kalman-Filter

Das erste Beispiel zeigt, wie der Roboter auf einen ruhenden Ball zufährt. Im Idealfall wären alle Observationen auf einen einzigen Punkt konzentriert, die entstehende

Trajektorie ebenfalls nur ein einziger Punkt. Das tatsächliche Ergebnis ist in Abbildung 9.10 zu sehen. Die obige Darstellung zeigt alle durch die Aufmerksamkeitssteuerung erkannten Objekthypothesen sowie die daraus resultierenden Trajektorien. Die untere Darstellung ist ein Ausschnitt des eigentlich beobachteten Balls *ohne* die fälschlicherweise erkannten Regionen und ohne die eingezeichnete Roboterposition. Der zeitliche Verlauf der Kurve ist an der jeweiligen Linienfarbe zu erkennen. Beginnend mit gelb, färbt sie sich im weiteren Verlauf grün, blau und wird am Schluß rot.

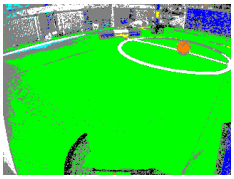
Wie im nächsten Abschnitt zu sehen sein wird, werden alle Beobachtungen, die hier nicht dem eigentlich gesuchten Ball entsprechen, nach der Regionensuche von den neuronalen Netzen korrekt negativ klassifiziert. Trotzdem wird auch für sie eine eigene Trajektorie erstellt. Schließlich sollen auch solche Beobachtungen von den in der Bildverarbeitung möglichen Optimierungen profitieren.

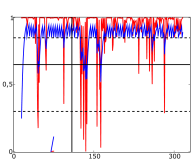
Die Bewegung des Roboters ist durch die zwei großen blauen Punkte dargestellt. Beginnend an Position $(0, 0)$ links, fährt der Roboter in gerader Linie auf den zweiten blauen Punkt zu und hält dort. Diese eigene Roboterposition ist nur durch die ermittelten Odometriemessungen bestimmt, und ist daher zwangsläufig nicht völlig korrekt. Der gesamte Erkennungsfehler ist auch an der irrtümlich ermittelten Bewegung des Balls zu sehen. Der Ball scheint sich über die Zeit leicht vom Roboter weg zu bewegen, liegt in Wirklichkeit aber genau an immer derselben Stelle. Entweder misst die Odometrie einen leicht falschen Wert, oder der Winkel der Kamera relativ zum Boden ist nicht korrekt eingestellt. Nichtsdestotrotz integriert der Kalman-Filter die jeweilige Position aber korrekt und liefert trotz teilweise stark verrauschter Daten eine gute mittlere Ballposition zurück. Das Erkennungsrauschen ist teilweise auf Diskretisierungsfehler der Regionensuche zurückzuführen. Hier werden nur ganzzahlige Pixelwerte verwendet. Zusätzlich beeinflussen Kipp- und Neigebewegungen des Roboters beim Anfahren und Abbremsen die Entfernungsschätzung aus der Bildposition stark.

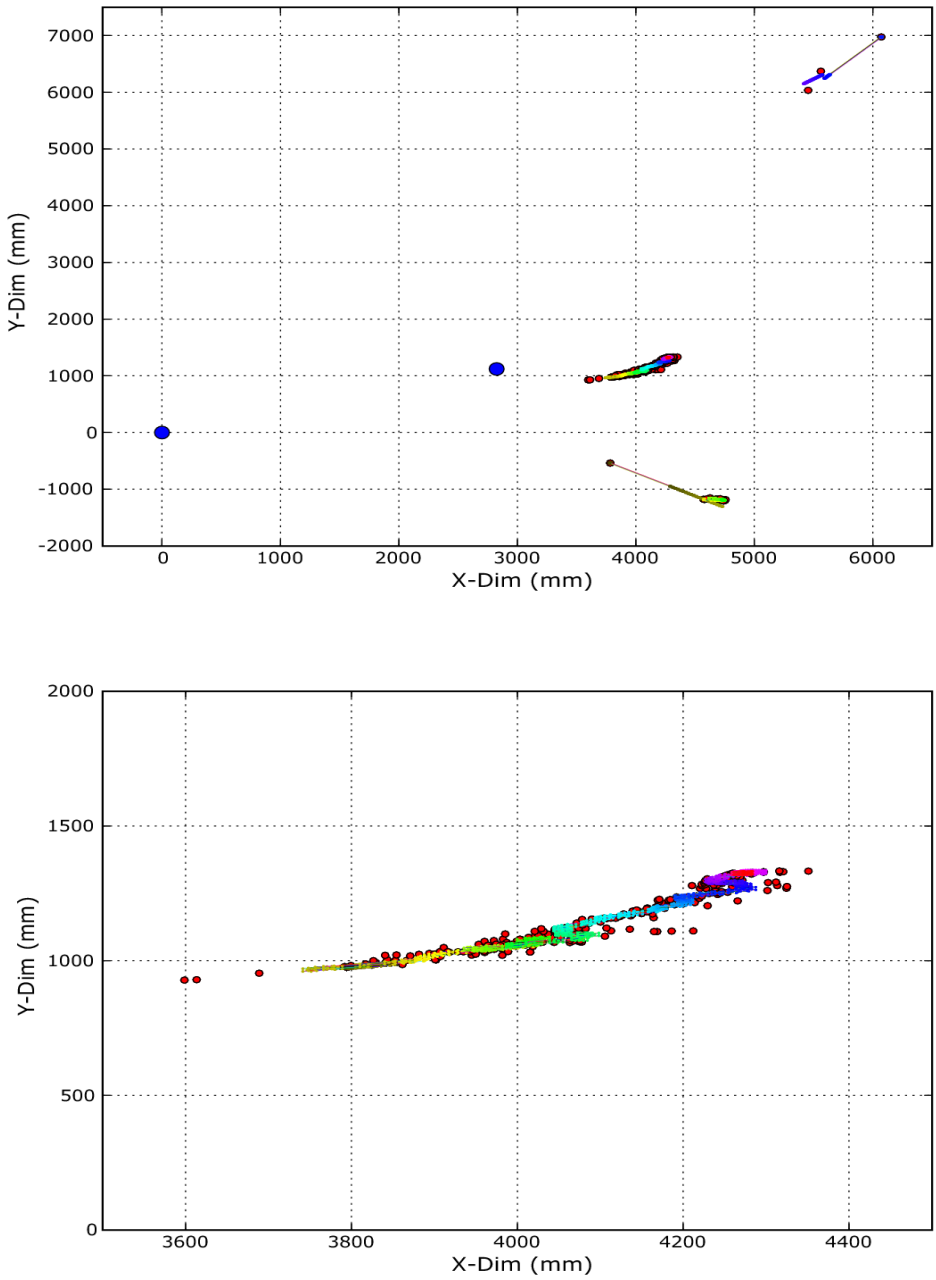
Im zweiten Versuch soll die Reaktion und Prädiktionsleistung des Kalman-Filters bei einer stark nichtlinearen Bewegung des Balls untersucht werden. Abbildung 9.11 zeigt diesen Versuch. Dabei steht der Roboter wiederum an Position $(0, 0)$, der Ball rollt von der rechten Seite ins Blickfeld des Roboters, prallt dann von einem Hindernis ab, bewegt sich beinahe im 180°-Winkel wieder zurück, läuft langsam aus und bleibt schließlich an einer Stelle liegen. Wieder stellen die roten Punkte die durch die Regionensuche gefundenen Objekthypothesen dar, die farbigen Linien die daraus resultierenden Objekttrajektorien. Wieder ist der zeitliche Verlauf der Bewegung über den Farbverlauf wie oben beschrieben von gelb bis blau illustriert.

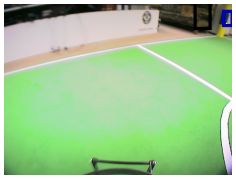
An zwei Stellen des beschriebenen Szenarios ist die Bewegung des Balls nicht linear. Einmal in dem Moment, in dem er an dem Hindernis abprallt, sowie auch wenn er durch die Reibung am Schluß der Sequenz ausrollt. Beide Stellen sind besonders interessant, da diese der Grundannahme eines linearen Kalman-Filters eigentlich widersprechen. In beiden Situationen passt die resultierende Objekttrajektorie aber sehr gut zu den jeweiligen, erkannten Positionen. Der Kalman-Filter macht beim Abpraller keinen abrupten Knick, sondern schwingt sich nach nur wenigen aufeinanderfolgenden

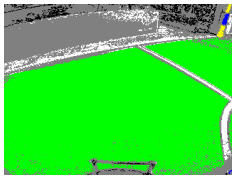












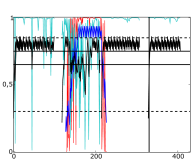


Abbildung 9.10: Der Roboter fährt direkt auf einen ruhenden Ball zu. Im oberen Bild sind alle durch die Regionensuche erkannten Objekthypothesen (rote Punkte), sowie die daraus entstandenen Objekttrajektorien (farbige Linien) dargestellt. Zusätzlich eingezeichnet ist die Startposition (blauer Punkt links), sowie die Endposition des Roboters (rechter blauer Punkt). Im unteren Bild ist der Ausschnitt der eigentlichen Ballposition vergrößert dargestellt.

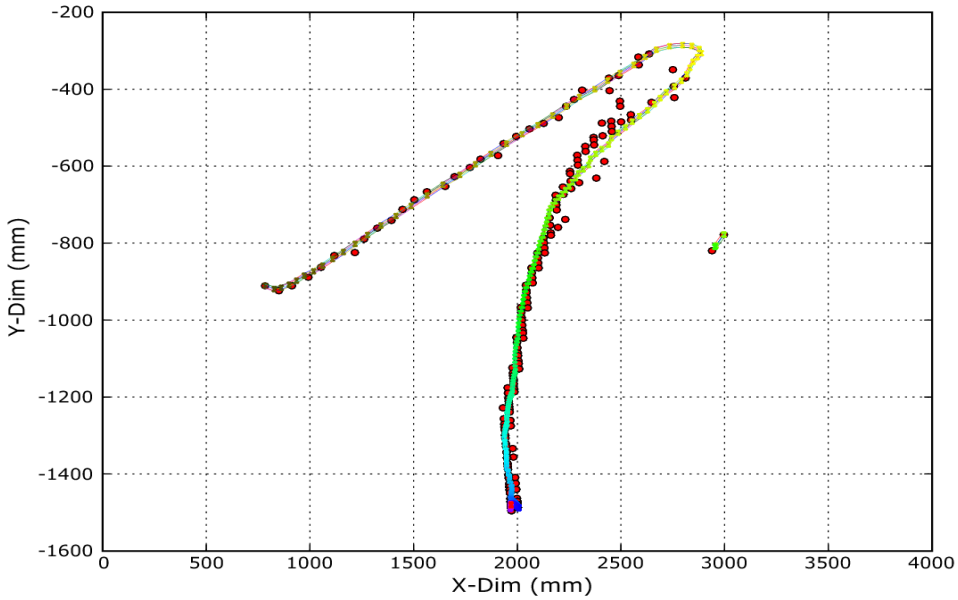


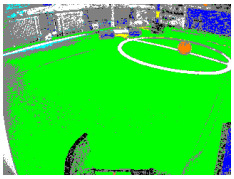
Abbildung 9.11: Der Ball bewegt sich, von rechts kommend, am Roboter vorbei, prallt an einem Hindernis ab und rollt wieder nach rechts zurück. Der Roboter steht dabei an Position $(0, 0)$. Die roten Punkte sind die durch die Regionensuche erkannten Objekthypothesen, die farbigen Linien die daraus resultierenden Tracks.

Bildern wieder auf die korrekte Position ein. Zusätzlich erschwert wird dies dadurch, daß der Ball durch den Aufprall auf das Hindernis beginnt, leicht zu hüpfen. Dadurch variiert die Entfernungsmessung innerhalb weniger aufeinanderfolgender Bilder stark.

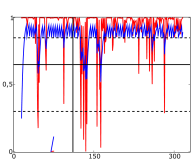
Wie schnell eine solche Reaktion auf die aufgenommenen Bilder erfolgen soll, läßt sich über die jeweiligen Parameter, wie in Kapitel 7 beschrieben, einstellen. In diesem Fall ist es das Systemrauschen, das für höhere Werte die Nichtlinearitäten gut zu modellieren vermag.

In beiden aufgezeigten Szenarien wird die eigentliche Ballposition durch die immer gleiche Trajektorie beschrieben. Das in Abschnitt 7.5 beschriebene Szenario, daß durch starke Nichtlinearitäten eventuell ein neuer Track für die gegensätzliche Bewegung initiiert wird, tritt hier nicht ein. Das bedeutet aber nicht, daß dieser Fall nicht doch irgendwann auftreten könnte. Wenn die Bewegung des Balls vor dem Aufprall noch höher wäre, könnte dies trotzdem passieren. Erfahrungsgemäß sind die Geschwindigkeiten dann aber so hoch, daß auch Bewegungsunschärfe und damit verbunden eine schlechte Klassifikationsleistung einen nicht unwesentlichen Anteil daran haben.





Darüberhinaus macht eine getrennte Analyse der Zuordnungsleistung von Tracks zu Beobachtungen im beschriebenen Szenario eher wenig Sinn. Die Menge an auftretenden Objekten auf dem Spielfeld grenzt die mögliche Menge an Objekthypothesen



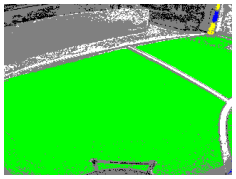
pro Objekttyp bereits stark ein. Wie in den obigen Beispielen bereits zu sehen war, werden auch einige wenige Fehlklassifikationen noch gut verarbeitet und sorgen nicht für eine Verwechslung der Tracks bei der Zuordnung. Sind im Kamerabild viele, sehr große Objekte wie etwa Roboter zu sehen, so ist in den allermeisten Fällen die Objekterkennung der limitierende Faktor. Selbst wenn die Objekte bezüglich des globalen Koordinatensystems noch eindeutig zu trennen wären, sind sie durch die Überlappung im Bild bereits lange vorher nur noch sehr schwer zu trennen und der eigentliche Einfluß der Zuordnungsleistung ist nur noch sehr schwer zu ermitteln. Aus diesem Grund wird hier auf eine getrennte Auswertung dieser Zuordnungsleistung verzichtet.

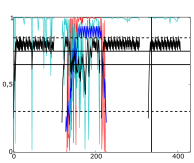
9.4.2 Informationsintegration

Weiter von Interesse ist nun noch, wie die zeitliche Integration die Sicherheit über das Auftreten eines Objekts im Bild verbessert, und wie der zweite dargestellte Vorteil, die Einsparung von Rechenressourcen, den Rechner entlastet. Abbildung 9.12 zeigt dafür drei Kurven, die den Verlauf dieser verschiedenen Sicherheitswerte über die Zeit, also die aufeinanderfolgenden Aufnahmen beschreibt. Der Versuch ist exakt derselbe wie bereits im obigen Abschnitt in Abbildung 9.11 beschrieben. Zu sehen ist, wie der Ball in Bild 15 zum ersten Mal in das Blickfeld des Roboters kommt und bis zum letzten Frame (Bild 316) immer zu sehen ist. Die durchgezogene der dicken schwarzen, waagrechten Linien bei 0,65 ist die Grenze, überhalb der eine positive Erkennung des jeweiligen Objekts angenommen wird. Die untere, gestrichelte waagrechte Linie bei 0,3 ist entsprechend der untere Schwellenwert, unterhalb dem angenommen wird, daß die erkannte Region kein Objekt des erwarteten Objekttyps enthält. Diese, sowie der andere, später beschriebene Wert, lassen sich frei für das jeweilige Szenario und auch für den jeweiligen Objekttyp wählen.

Die rote Kurve zeigt nun die Einzelergebnisse des neuronalen Klassifikators ohne irgendeine zeitliche Integration. Gut zu sehen ist, wie er in 21 Fällen unter diese Schwelle von 0,65 fällt. Das geschieht bereits, wenn nur ein einziges Klassifikationergebnis einen niedrigen Wert annimmt. Würde ein anderes Modul der Robotersoftware in diesem Moment die Ballposition benötigen, erhielte sie keine sinnvolle Position zurückgeliefert. Die blaue Linie zeigt die über die Zeit integrierte Wahrscheinlichkeit für das Vorhandensein eines Balls (siehe Abschnitt 8.1). Diese Kurve ist nur noch 9 Mal unter der Erkennungsschwelle (plus die drei Bilder zu Beginn der Sequenz). Zu sehen ist aber auch, daß sie ganz links, wenn der Ball zum ersten mal gesehen wird, einige wenige aufeinanderfolgende Bilder benötigt (in diesem Fall 4 Aufnahmen), bis dessen Existenz ausreichend validiert wurde. Diese blaue Kurve entspricht in etwa dem, was an Möglichkeiten bleibt, wenn Klassifikation und zeitliche Integration als zwei getrennte Module implementiert sind. Die zweite der beschriebenen Verbesserungen (Abschnitt 3.2), die zusätzliche und gleichzeitig mögliche Reduktion des Ressourcenverbrauchs, zeigt die dritte, schwarze Kurve. Dabei wird eine erneute Klassifikation nur dann durchgeführt, wenn der kumulierte Sicherheitswert unter die zweite gestrichelte, obere Linie von 0,85 zu sinken droht. Diese zweite Schwelle ist nötig, damit eine einzige Fehlklassifikation die Gesamtsicherheit nicht genau dann un-







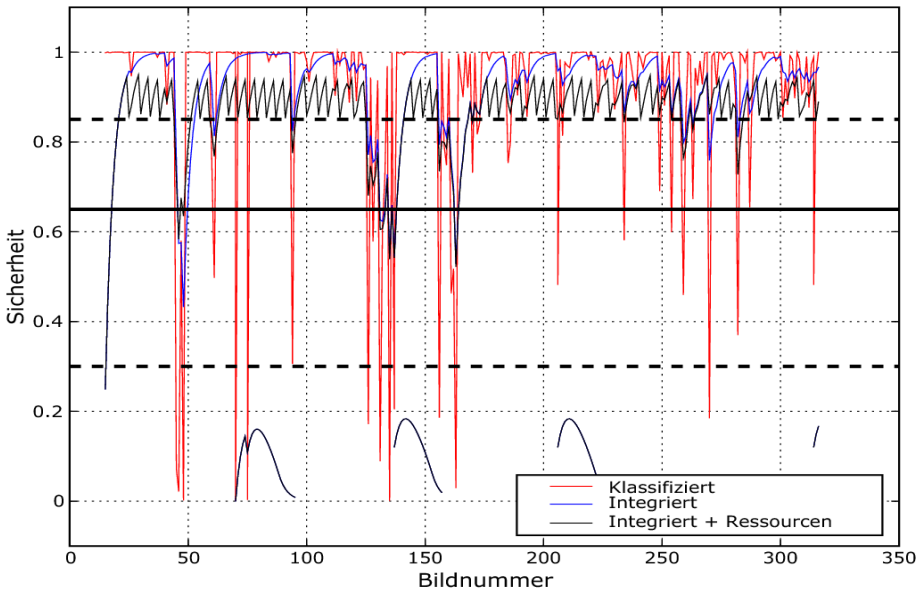
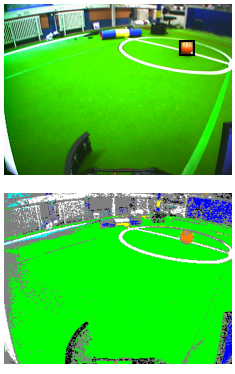
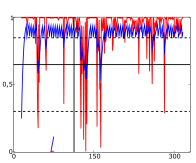


Abbildung 9.12: Wahrscheinlichkeitswerte der Ballerkennung, aufgetragen über die Zeit. Die rote Linie zeigt die jeweiligen einzelnen neuronalen Klassifikationsergebnisse. Die blaue Linie beschreibt die, gemäß oben vorgestellter Formel, integrierten Wahrscheinlichkeitswerte. Die schwarze Linie enthält zusätzlich die als „Reduktion des Ressourcenverbrauchs“ beschriebene Vorgehensweise.

ter die Erkennungsschwelle sinken läßt, wenn gerade eben wenige aufeinanderfolgende Fehler die Kurve bereits an diese untere Schwelle angenähert haben. Der charakteristische sägezahnartige Verlauf dieser letzten Kurve illustriert deutlich das abwechselnde Klassifizieren und Verwenden von a-priori-Werten. Im Schnitt wird hier daher nur in etwa jedem zweiten Bild der Ball tatsächlich durch den neuronalen Klassifikator validiert. Gleichzeitig sinkt aber auch hier in nur 9 Fällen der Sicherheitswert unter die Erkennungsschwelle, ist also praktisch identisch zum Verlauf der blauen Kurve. Der Ressourcenverbrauch läßt sich damit je nach Bedarf für ein bestimmtes Szenario über einen einzigen, einfachen Schwellenwert (die obere waagrechte Linie) steuern. Trotzdem ist die Gesamterkennungsrate immer noch signifikant besser, als wenn nur ständig die neuronale Klassifikation wiederholt würde.

Die unteren, kurzen schwarzen Liniensegmente sind Trajektorien, die aus später verworfenen Objekthypothesen entstanden sind. Gut zu sehen ist, wie diese Linien nie den unteren Schwellenwert nach oben durchbrechen. Sie werden also nie fälschlicherweise als ein solcher Objekttyp angesehen (auch nicht als Unbestimmt, wie es der Fall wäre, wenn die Kurve im mittleren Bereich zwischen oberem und unterem Schwellenwert liegt).





9.4.3 Abschließende Bemerkungen

Das vorgestellte Verfahren zur zeitlichen Prädiktion und Integration erlaubt eine durchgängige Verfolgung von Objekten auch unter ungünstigen Bedingungen. Das gewählte, berechnungstechnisch wenig aufwendige Kalman-Filter-basierte Verfahren erzielt hierbei eine für das Szenario ausreichende Leistung, obwohl die tatsächlich auftretenden Bedingungen den ursprünglichen Annahmen widersprechen. Auch das Auftreten mehrerer Instanzen des gleichen Objekttyps ist, wie gesehen, kein Problem für das Zuordnungsverfahren. Die Menge an gleichzeitig zu verfolgenden Objekten ist für das Einsatzgebiet ebenfalls ausreichend.

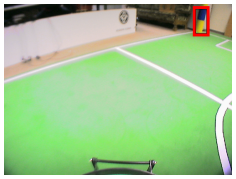
Die Informationsintegration über die Zeit verbessert die Erkennungsleistung des Gesamtsystems signifikant. Selbst bei zusätzlichen Optimierungen wie der beschriebenen Möglichkeit zur Ressourcenreduktion wird dieser Informationsgewinn nicht wieder aufgebraucht. Gut zu sehen war auch, wie sich die gewünschte Gesamtsicherheit für jeden Objekttyp mit Hilfe nur eines einzigen Parameters gegenüber dem benötigten Berechnungsaufwand abwägen läßt.

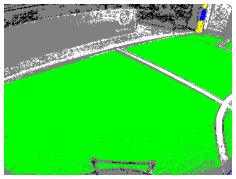
Einzig wenn ein Objekt neu im Blickfeld des Roboters auftaucht, oder wenn doch einmal ein neuer Track für ein eigentlich schon verfolgtes Objekt initiiert werden würde, sinkt die Reaktivität des Systems. Zu Beginn eines neuen Tracks werden immer ein paar wenige aufeinanderfolgende Observationen benötigt, um die Existenz dieses Objekts zu bestätigen. Die gewonnene Gesamtsicherheit wiegt diesen kleinen Nachteil aber mehr als auf.

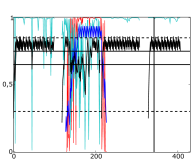
9.5 Gesamtsystem

Im folgenden soll abschließend die Erkennungsleistung der gesamten Berechnungskette vorgestellt werden. Wiederum nur positive und negative Treffer in Tabellenform darzustellen, beschreibt die Leistungsfähigkeit aber nur unzureichend. Weder sagen sie etwas über die Verteilung innerhalb der Sequenzen aus, noch bekommt der Leser ein wirkliches Gefühl für die Leistung des Systems. Aus diesem Grund sind einige Sequenzen hier in Form eines Daumenkinos an den beiden Rändern unten, jeweils links und rechts abgebildet.

Die erste Bilderfolge ist dieselbe Abpraller-Sequenz, wie sie bereits im vorherigen Abschnitt beschrieben wurde. Auch die zweite Bilderfolge ist eine im Labor aufgezeichnete Folge, da, wie gesagt, die Aufnahme von Sequenzen in voller Framerate nicht möglich ist, wenn die gesamte Robotersoftware läuft. Dadurch sind eventuell nicht genauso viele Schwierigkeiten in den Bildern enthalten, wie sie normalerweise in echten Spielen vorhanden wären. In dieser Sequenz sind aber alle Objekttypen enthalten, die erkannt werden sollen. Dabei fährt der Roboter einmal auf einer Kreisbahn rund um den Mittelpunkt des Spielfelds. Er sieht dort beide Tore, den Ball, alle Eckpfosten und einen auf dem Spielfeld stehenden Roboter.







Jeweils links auf den geraden Seiten ist diese Kreis-Sequenz abgebildet, auf den rechten, ungeraden Seiten die Abpraller-Sequenz. Von oben nach unten ist jeweils das Originalbild mit eingezeichneten Objektpositionen, das farbsegmentierte Bild und der bereits von oben bekannte Plot der Konfidenzwerte abgebildet. Dabei ist ein erkannter

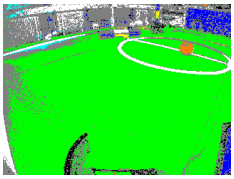
Ball oder Roboter mit einem schwarzen Rahmen markiert, das gelbe Tor mit einem hellgelben Rahmen, das blaue Tor mit einem hellblauen Rahmen, die Eckpfosten mit roten, und die jeweils linken und rechten Torpfosten von gelbem und blauem Tor mit jeweils dunkelgelben und dunkelblauen Rahmen.

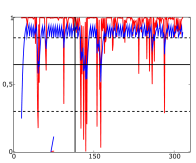
Die Plots enthalten jeweils bereits den gesamten Kurvenverlauf und zeigen die aktuelle Position über eine breite, schwarze, senkrechte Linie an. Im Falle der Abpraller-Sequenz ist das die bereits von oben bekannte Kurve. Abgebildet sind diesesmal nur die einzelnen Klassifikationswerte in rot sowie die Endergebnisse des kompletten Objekterkennungsmoduls in blau inklusive zeitlicher Integration und dem Verfahren zur Rechenzeiteinsparung. Im Plot der Dreh-Sequenz sind nicht nur die Ergebnisse der Ballerkennung, sondern auch die Ergebnisse der Eckpfostenerkennung eingetragen. Die roten und blauen Linien entsprechen denen der anderen Kurve. Die cyanfarbene Kurve beschreibt die einzelnen Klassifikationsergebnisse für die Eckpfosten, die schwarze Linie wiederum die integrierten Werte dafür. Die Wahrscheinlichkeitsschwelle, ab der die Eckpfosten neu klassifiziert werden, ist niedriger gewählt als jene für den weitaus wichtigeren Ball. Hier werden mehr Fehlklassifikationen zugunsten eines niedrigeren Rechenaufwands in Kauf genommen. Zu sehen ist dies gut daran, daß die schwarze, sägezahnartige Linie ein deutliches Stück unter der blauen Linie liegt. Die einzelnen Segmente der cyanfarbenen und schwarzen Linien entstehen dadurch, daß immer wieder Eckpfosten den Sichtbereich des Roboters verlassen und andere Pfosten im Bild auftauchen.

Die Gesamtseitenanzahl dieser Arbeit und die Menge an aufgenommen Bildern erzwingt, daß nur etwa jedes sechste Bild für die Dreh-Sequenz abgebildet werden kann. Falls sie die Bilder größer und in Form eines Films sehen wollen, so stehen ihnen unter http://smart.informatik.uni-ulm.de/DataSets/RoboCup/Movies/ alle Sequenzen in verschiedenen Formaten zum Herunterladen zur Verfügung.

Um die Plots für die Dreh-Sequenz einmal noch in groß zu sehen, sind in Abbildung 9.13 einzelne Kurven für die verschiedenen Objekte aufgetragen: Links oben die Werte für die Ballerkennung, rechts für die Eckpfostenerkennung und unten für eine zusätzliche Robotererkennung, jeweils rot die einzelnen Klassifikationsergebnisse, schwarz die Endergebnisse von Integration und Ressourceneinsparung. Bei der Robotererkennung sind teilweise mehrere, sich überlagernde Linien zu erkennen, da oft mehrere Regionen pro Bild zu verwalten sind. Gut zu sehen ist, daß gleich zu Beginn der Sequenz und am Ende der Folge jeweils tatsächlich ein Roboter erkannt wird. Dazwischen sind einige wenige Stellen, an denen die Integrationskurve fälschlicherweise über die Signifikanzschwelle kommt. An der Menge der zu verwaltenden, unterhalb dieser Schwelle liegenden Trajektorien ist aber gut zu erkennen, daß die meisten dieser falsch positiven Regionen anschließend korrekt als negativ bewertet werden. Der Anteil von falsch positiv zu korrekt negativ klassifizierten Roboterhypothesen entspricht hier in etwa dem in Tabelle 9.8 ermittelten Verhältnis dieser Werte.







9.5.1 Berechnungsdauer

Laufzeitmessungen sind prinzipiell immer mit etwas Vorsicht zu genießen. Je nach sonstiger Software, die auf dem Rechner noch läuft, je nach Prozessorarchitektur

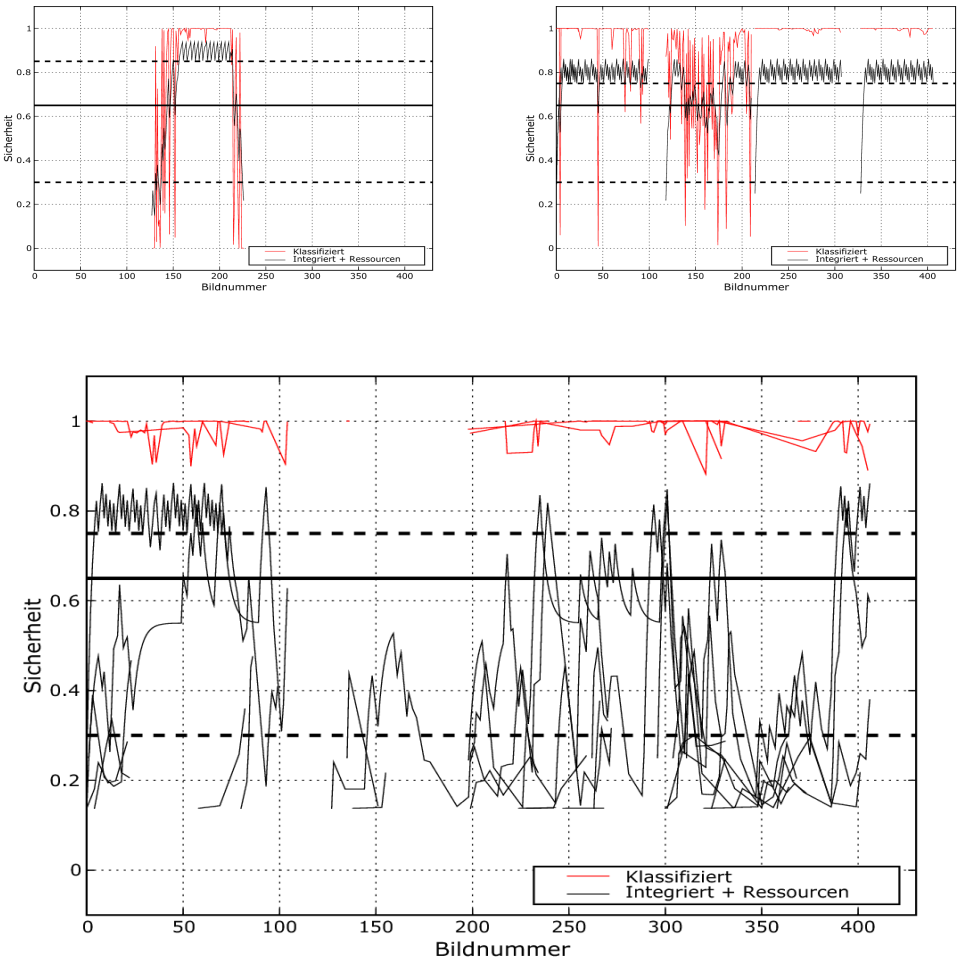
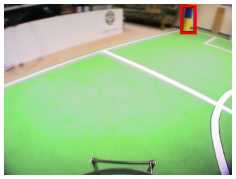
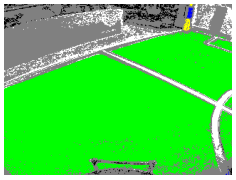
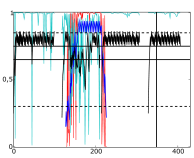


Abbildung 9.13: Wahrscheinlichkeitswerte der verschiedenen Objekte über die Zeit für die Dreh-Sequenz. Links oben sind die Werte für die Ballerkennung, rechts für die Eckpfostenerkennung und unten für die Robotererkennung dargestellt, jeweils rot die einzelnen Klassifikationsergebnisse, schwarz die Endergebnisse von Integration und Ressourceneinsparung.







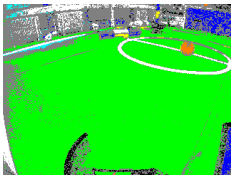
oder auch je nach Compileroptionen variieren die Berechnungszeiten schnell um einige Prozent. Da die Berechnungsdauer der meisten der hier verwendeten Filter auch davon abhängig ist, was in vorhergehenden Stufen der Bearbeitungskette gefunden wurde, sind absolute Zeitangaben immer auch direkt von der verwendeten Bildersequenz abhängig. Die nachfolgenden Ergebnisse sind alle anhand der oben beschriebenen Dreh-Sequenz ermittelt. Dabei sind alle Werte über 5000 Bilder gemittelt, damit lief die Sequenz mehr als 12 Mal komplett durch, bevor die Statistik erstellt wurden. Wie im realen Einsatz waren alle Bilder 640×480 Pixel groß, die Regionensuche wird auf Bildern halber Seitenlänge, das heißt 320×240, durchgeführt.

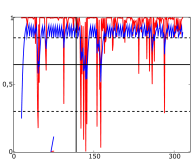
Im Mittel über alle diese Bilder dauert die Berechnung eines einzigen Bildes damit 18,8 Millisekunden, was deutlich kürzer als die maximal möglichen 33 ms bei einer Framerate von 30 Bildern pro Sekunde ist. Darin enthalten ist die Erkennung aller Objekte, wie sie im obigen Beispiel eingezeichnet sind: Ball, Tore, Eckpfosten, Torpfosten und die Roboter. Zusätzlich wurden in diesem Versuch zur besseren Vergleichbarkeit alle Objekttypen immer in jedem Bild ausgewertet. Da im realen Einsatz die zur Selbstlokalisation nötigen Merkmale aber nur bei Bedarf berechnet werden, ist der gesamte Berechnungsaufwand im Mittel deutlich geringer.

Da die Regionensuche für den Roboter im Großen und Ganzen eine Suche nach schwarzen Bereichen im Bild ist, und diese Bereiche natürlich an allen möglichen dunklen Stellen im Bild einen Treffer findet, ist es nicht überraschend, daß hier besonders viele Aufmerksamkeitsregionen gefunden werden. Folgerichtig benötigt der Filter für die Berechnung der Roboter-Orientierungshistogramme im Mittel mit Abstand am längsten. Pro Bild waren das rund 4 ms. Dabei lag der Minimalwert bei etwa 3,4 ms, der Maximalwert bei 10,7 ms. Solche Minimal- und Maximalwerte sind aber nicht über die Zeit gemittelt und damit sehr wahrscheinlich auch durch andere noch laufende Prozesse des Systems beeinflußt. Die beiden nächsten Filter in dieser Rangfolge waren die kombinierte Regionensuche für Tor und Eckpfosten mit 2,5 ms, sowie die Robotersuche mit 1,7 ms. Nur zwei weitere Filter für die Torpfostendetektion benötigten im Schnitt überhaupt noch mehr als eine Millisekunde, alle weiteren Filter werden in weniger als einer Millisekunde berechnet.

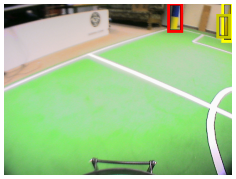
Tabelle 9.13 zeigt eine Auflistung der einzelnen Filterkomponenten für die Eckpfostenerkennung. Diese Filter entsprechen exakt denen aus Abbildung 8.2. Von oben nach unten sind die einzelnen funktionalen Einheiten zu sehen: Regionensuche, Bodentest, Erstellung der binären Suchmaske und die eigentliche Merkmalsextraktion und Klassifikation. Dabei arbeitet die Regionensuche auf einem Indexbild, welches innerhalb von etwas weniger als einer Millisekunde erstellt werden kann. Dieses Indexbild wird aber nicht nur exklusiv von der Eckpfostensuche verwendet, sondern ist die Basis aller Regionensuchmodule aller Objekttypen.

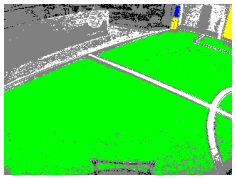






Filter	Zeit (ms)
Regionensuche	
AttentionGoalCornerPoles	2,507
FilterROIsByType (Tor)	0,004
ObjectTracking (Tor)	0,063
IntegrateObservations (Tor)	0,013
FilterROIsByType (Eckpfosten)	0,004
ObjectTracking (Eckpfosten)	0,149
FilterPredictedROIs (Eckpfosten)	0,004
FilterUnpredictedROIs (Eckpfosten)	0,004
FilterUncertainROIs (Eckpfosten)	0,004
FilterNewTracks (Eckpfosten)	0,004
ConcatROIs (Eckpfosten)	0,005
Bodentest	
ScaleROI	0,005
CheckForFloor	0,005
Binäre Maske	
ScaleROI	0,005
ResizeROI	0,006
MaskByColor	0,208
ShrinkImageParts	0,027
Enlarge	0,858
Merkmale und Klassifikation	
ResizeROI	0,005
YUV422toRGB	0,064
ShrinkImageParts	0,093
And	0,057
ColorHist	0,246
YUV422toGray	0,021
ShrinkImageParts	0,040
SobelVertical	0,025
SobelHorizontal	0,023
OrientCurvatureHistMask	0,550
NeuralNet	0,178
IntegrateObservations	0,018





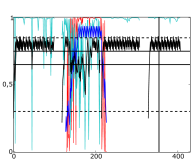


Tabelle 9.13: Die Berechnungszeit einzelner Filter am Beispiel der Eckpfostenerkennung. Die einzelnen Bereiche entsprechen den einzelnen funktionalen Einheiten, wie sie in Abbildung 8.2 bereits illustriert wurden. Von oben nach unten: Regionensuche, Bodentest, Erstellung der binären Suchmaske und die eigentliche Merkmalsextraktion und Klassifikation.

10 Zusammenfassung

Abschließend soll in diesem Kapitel noch ein Fazit der gesamten Arbeit gezogen werden. Nach einer kurzen Rekapitulation von Problemstellung und Lösungsansatz folgt eine kurze Einordnung der vorliegenden Arbeit im Vergleich zu anderen Gebieten. Im Ausblick werden einige Punkte angesprochen, die in unserer Arbeitsgruppe bereits diskutiert und teilweise auch implementiert oder zumindest ansatzweise versucht wurden. Im Gegensatz dazu sind andere, offene Probleme noch weiter von einer Lösung und Realisierung entfernt.

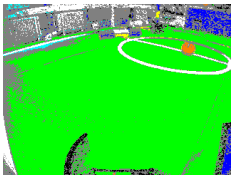
10.1 Kernpunkte

Im Rahmen dieser Arbeit ist ein funktionierendes Gesamtsystem zur *visuellen Objekterkennung in dynamischen Umgebungen* entstanden. Das heißt, das Basissoftware und Objekterkennung umfassende Gesamtsystem funktioniert innerhalb einer dynamischen und unstrukturierten Umgebung. Es erkennt alle relevanten Objekte robust und schnell. Insbesondere erreicht das Gesamtsystem seine Leistung unter den gegebenen Schwierigkeiten.

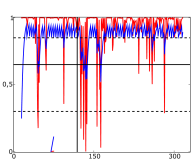
Die Schwierigkeiten, noch einmal kurz rekapituliert, sind die Folgenden: Schlechte Bildqualität, insbesondere durch Bewegungsunschärfe und eine starke Linsenverzerrung. Diese Bilder entstehen in einer unstrukturierten Umgebung, innerhalb derer die gesuchten Objekte zusammen mit anderen, unbekannten Objekten abgebildet werden, vor diesen liegen, oder von diesen teilweise verdeckt werden. Zusätzlich weisen die eigentlich bekannten Objekttypen eine große Variabilität innerhalb ihrer existierenden Instanzen auf. Die zur zeitlichen Integration nötigen Eigenbewegungsinformationen sind außerdem durch die hohe Dynamik und den starken Kontakt der Roboter untereinander extrem ungenau. Weiter verschärft wird die Situation durch ständige Regeländerungen, die eine kontinuierliche Adaption der verwendeten Verfahren erfordert.

Grundlegende Voraussetzung, um die gestellte Aufgabe innerhalb einer solchen Umgebung zu erfüllen, war es, nicht nur einfach ein Erkennungssystem zu realisieren, sondern ein komplettes Gesamtsystem von Basissoftware, Objekterkennung und temporaler Integration zu entwickeln. Dabei ist ein System entstanden, welches den modularen und komponentenbasierten Grundgedanken sowohl auf Softwareseite als auch auf algorithmischer Seite umsetzt. Damit ist eine hohe Flexibilität und Wiederverwendbarkeit aller dieser Komponenten für zukünftige Erweiterungen und andere Einsatzgebiete garantiert.





Ein wichtiger Punkt innerhalb des gesamten Systems ist die enge Verknüpfung von Objekterkennung und zeitlicher Integration. Wie dargestellt, läßt sich damit nicht nur eine deutliche Steigerung der Robustheit erreichen, sondern auch eine deutliche Reduktion der benötigten Rechenzeit. Dies ist nicht nur eine wichtige Voraussetzung für

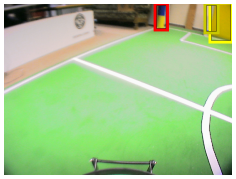


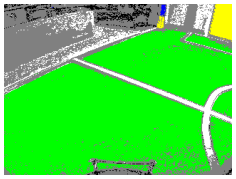
die erreichte Echtzeitfähigkeit, sondern auch Garant für möglichst geringe Latenzen zwischen dem Aufzeichnen der Bilder und dem Erhalt des fertigen Resultats. Zusätzlich lassen sich so bekannte, statistische Informationen über die Umgebung direkt in das Objekterkennungssystem einbringen und damit verbunden die Berechnung zur Laufzeit auf die jeweilige Situation anpassen.

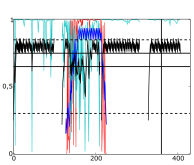
Weiter bieten die verwendeten Merkmale die Möglichkeit, Eigenschaften der Objekte zwar flexibel, gleichzeitig aber auch sehr selektiv zu beschreiben. Insbesondere die Verwendung von kombinierten Orientierungs-/Krümmungshistogrammen ermöglicht eine mächtige und doch mit begrenztem Aufwand zu berechnende Beschreibung der Objekte in der Umgebung.

Die Kombination aller dieser Entwicklungen führte zu einem System, welches wie kaum ein anderes im RoboCup komplexe Verfahren der Bildverarbeitung und Neuroinformatik verwendet, und als solches auch tatsächlich während echter RoboCup-Turniere eingesetzt wird. Besonders dieser Aspekt der Verwendung und Auswertung auf Basis realer, während echter Turniere gewonnener Daten, verdeutlicht die Einsatzfähigkeit des gesamten Verfahrens und liefert eine fundierte Aussage über die Robustheit und Fähigkeit des Systems. Dazu gehört auch die ausführliche Untersuchung der Beleuchtungsbedingungen im RoboCup, welche Grundlage und willkommene Hilfestellung für eine angeregte Debatte innerhalb der RoboCup-Community war.

Damit füllt die vorliegende Arbeit ein wichtige Lücke zwischen anderen Arbeiten aus dem RoboCup und verwandten Forschungsgebieten. Im Gegensatz zu Lösungsansätzen für den Bürobereich ist diese Arbeit innerhalb einer dynamischen Umgebung mit all seinen Schwierigkeiten wie schlechter Bildqualität und unstrukturierter Umgebung entstanden. Auch spielt die gesamte Echtzeitfähigkeit im Bürobereich keine so große Rolle, obwohl dort die zeitliche Integration aufgrund deutlich besserer Odometriewerte wesentlich einfacher wäre. Im Gegensatz zum Bereich der autonomen Transportsysteme muß im vorliegenden Fall mit der gegebenen Hardwareausstattung gearbeitet werden. Wirtschaftliche und technische Beschränkungen verbieten den Einsatz beispielsweise von Stereokamerasystemen oder besonders rechenaufwendigen Verfahren. Im Gegensatz zu vielen anderen Arbeiten im RoboCup aber kommen in dieser Arbeit trotzdem wesentlich anspruchsvollere Verfahren zum Einsatz, um das Gesamtsystem robuster und flexibler zu machen. Gleichzeitig wurde hier besonders darauf geachtet, daß sich die entwickelten Lösungen auch in anderen, artverwandten Bereichen einsetzen lassen. Entstanden ist so ein Objekterkennungssystem, welches, wie ich hoffe, einen wertvollen Beitrag zum besseren Verständnis der Herausforderungen solcher Szenarien, aber auch der Möglichkeiten eines solchen Lösungsansatzes liefert.







10.2 Ausblick

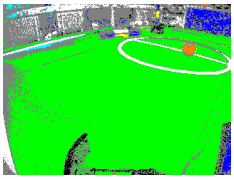
Zur weiteren Steigerung und Stabilisierung der Erkennungsleistung des Systems sind bereits einige Punkte angedacht oder prototypisch realisiert. Dazu gehört beispielsweise die Idee, verschiedene Klassifikatoren, verschiedene Merkmalsparametrisierungen oder sogar verschiedene Merkmale für verschiedene Objektgrößen und -instanzen

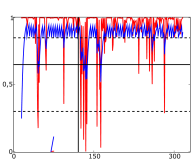
zu verwenden. Große Bälle direkt vor dem Roboter und weit entfernte Bälle könnten so mit individuell trainierten Klassifikatoren vielleicht noch robuster erkannt werden. Zusätzlich bietet sich ein separater Klassifikator für die eigenen Roboter an, da deren Ausprägung exakt bekannt ist. Eine weitere, inzwischen nötige Erweiterung wird es sein, die Suche nach Objekten auch auf solche auszudehnen, die sich nicht auf dem Boden befinden. Da inzwischen immer mehr Mannschaften dazu übergehen, den Ball auch hoch zu schießen, ist es besonders für den Torhüter wichtig, auch diese zu erkennen. Mit dem hier vorgestellten Verfahren ist es nun möglich, solche Objekte sicher und robust zu erkennen, und auf die Grundannahme liegender Objekte zu verzichten.

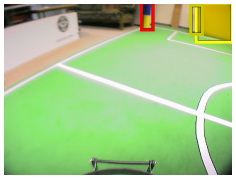
Einer der zukünftig wichtigsten Arbeitspunkte wird es aber sein, die Regionensuche weiter zu verbessern. Nicht nur die angesprochene statische Konfiguration der Farbbereichsgrenzen ist zukünftig nicht mehr ausreichend, sondern die gesamte Farbblobsuche muß durch ein besseres System ersetzt werden. Zwar gibt es dazu bereits einige Arbeiten im RoboCup-Umfeld, aber alle diese Lösungsvorschläge scheinen durch die vielen verwendeten Heuristiken und Annahmen zu stark auf den RoboCup selbst fokussiert zu sein. Eine Übertragbarkeit dieser Verfahren auf andere, realweltliche Bereiche ist fraglich. Mein Kollege Ulrich Kaufmann hat sich im Rahmen seiner laufenden Doktorarbeit dieses Problems angenommen. Hauptpunkte dieser Arbeit werden sein, sich besser auf sich verändernde Beleuchtungsbedingungen einstellen zu können und gleichzeitig weg von einer rein farbbasierten Bereichserkennung zu kommen.

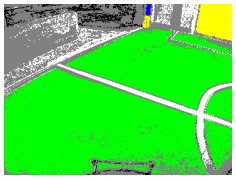
Genau das ist eines der wichtigsten offenen Probleme für die Zukunft. Es ist nicht unerwünscht, szenariospezifisches Wissen in ein System zu integrieren. Die Frage ist aber, wie sich solches Kontextwissen in einem ansichtenbasierten System repräsentieren läßt, ohne daß dadurch das Gesamtsystem unnötig stark auf das jeweilige Einsatzgebiet festgelegt wird, sondern sich trotzdem einfach auf veränderte Umgebungsbedingungen anpassen läßt.

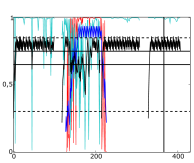











0,5
0
0
200
400

Literaturverzeichnis

[1] Aloimonos, J. ; Weiss, I. ; Bandyopadhyay, A.: Active Vision. In: *Proceedings of DARPA Image Understanding Workshop*, 1987, S. 552 – 573

[2] Amoroso, C. ; Chella, A. ; Morreale, V. ; Storniolo, P.: A Segmentation System for Soccer Robot Based on Neural Networks. In: *IJCAI-99 Third International Workshop on RoboCup*, 1999

[3] Anzani, Federico ; Bosisio, Daniele ; Matteucci, Matteo ; Sorrenti, Domenico G.: On-Line Color Calibration in Non-stationary Environments. In: [15], S. 396 – 407

[4] Bajcsy, R.: Active perception. In: *Proceedings of the IEEE* 76 (1988), Nr. 8, S. 996 – 1006

[5] Bar-Shalom, Y. ; Fortman, T. E.: *Tracking and Data Association*. New York : Academic Press, 1988

[6] Bar-Shalom, Y. ; Li, X. R.: *Estimation with applications to Tracking and Navigation*. John Wiley and Sons, Inc., 2001

[7] Baten, S. ; Mandelbaum, R. ; Luetzeler, M. ; Burt, P. ; Dickmanns, E.: Techniques for Autonomous, Off-Road Navigation. In: *IEEE Intelligent Systems magazine, special issue on Vision-Based Driving Assistance* (1998), November - Dezember, S. 57 – 65

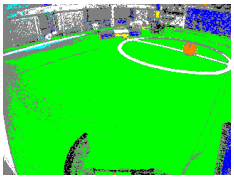
[8] Bertozzi, M. ; Broggi, A. ; Fascioli, A. ; Tibaldi, A. ; Chapuis, R. ; Chausse, F.: Pedestrian Localization and Tracking System with Kalman Filtering. In: *P*roceedings of IEEE Intelligent Vehicles Symposium[62]

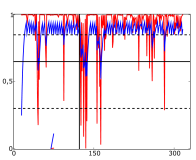
[9] Bertozzi, Massimo ; Broggi, Alberto ; Cellario, Massimo ; Fascioli, Alessandra ; Lombardi, Paolo ; Porta, Marco: Artificial Vision in Road Vehicles. In: *Proceedings of the IEEE - Special issue on Technology and Tools for Visual Perception* 90 (2002), Juli, Nr. 7, S. 1258 – 1271

[10] Besl, P. J. ; Jain, R. C.: Three-dimensional object recognition. In: *Computing Surveys* 17 (1985), Nr. 1, S. 75 – 145

[11] Birk, A. (Hrsg.) ; Coradeschi, S. (Hrsg.) ; Tadokoro, S. (Hrsg.): *Lecture Notes in Computer Science*. Bd. 2377: *RoboCup-2001: Robot Soccer World Cup V*. Berlin, Heidelberg, Germany : Springer-Verlag, 2002







[12] Bistry, David: *The Complete Guide to MMX Technology*. 1. McGraw-Hill/TAB Electronics, 1997

[13] Black, Michael J. ; Jepson, Allan D.: A Probabilistic Framework for Matching Temporal Trajectories: CONDENSATION-Based Recognition of Gestures and Expressions. In: *5th European Conference on Computer Vision (ECCV'98)*. Freiburg, Germany : Springer-Verlag, 1998 (Lecture Notes in Computer Science)

[14] Blackman, S. S. ; Popoli, R.: *Design and Analysis of Modern Tracking Systems*. Norwood, Massachussets : Artech House, 1999

[15] Bredenfeld, Ansgar (Hrsg.) ; Jacoff, Adam (Hrsg.) ; Noda, Itsuki (Hrsg.) ; Takahashi, Yasutake (Hrsg.): *Lecture Notes in Computer Science*. Bd. 4020: *RoboCup 2005: Robot Soccer World Cup IX*. Springer-Verlag, 2006

[16] Bruce, James ; Balch, Tucker ; Veloso, Manuela: Fast and inexpensive color image segmentation for interactive robots. In: *Proceedings of IROS-2000*. Japan : IEEE Press, Oktober 2000

[17] Brusey, James ; Padgham, Lin: Techniques for obtaining robust, real-time, colour-based vision for robotics. In: Veloso, M. (Hrsg.) ; Pagello, E. (Hrsg.) ; Kitano, H. (Hrsg.): *RoboCup-1999: Robot Soccer World Cup III* Bd. 2019. Berlin, Heidelberg, Germany : Springer-Verlag, 2000 (Lecture Notes in Computer Science)

[18] Cave, K. R.: The FeatureGate Model of Visual Selection. In: *Psychological Research* 62 (1999), S. 182 – 194

[19] Chin, R. T. ; Dyer, C. R.: Model-Based Recognition in Robot Vision. In: *Computing Surveys* 15 (1986), Nr. 1, S. 67 – 108

[20] Chui, Charles K.: *An Introduction to Wavelets*. San Diego : Academic Press, 1992

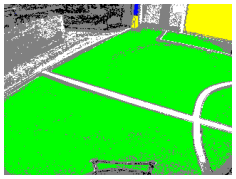
[21] Committee, MSL T.: *Middle Size Robot League, Rules and Regulations*. Erhältlich unter http://www.er.ams.eng.osaka-u.ac.jp/rc2004msl/-index.cgi?page=Regulations+and+Rules, 2004

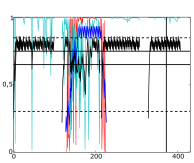
[22] Dahm, I. ; Deutsch, S. ; Hebbel, M. ; Osterhues, A.: Robust Color Classification for Robot Soccer. In: [106]

[23] Dickmanns, Ernst D.: An advanced vision system for ground vehicles. In: *Proceedings of 1st International Workshop on In-Vehicle Cognitive Computer Vision Systems (IVCCVS)*. Graz, Austria, April 2003, S. 1 – 12

[24] Dickmanns, Ernst D.: A general dynamic vision architecture for UGV and UAV. In: *Applied Intelligence* 2 (2004), September, Nr. 3, S. 251 – 270. – 1992







[25] Dickmanns, Ernst D.: Three-Stage Visual Perception for Vertebrate-type Dynamic Machine Vision. In: *Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*. Madeira, Portugal, Februar - März 2004

[26] Driscoll, J. A. ; II, R. A. P. ; Cave, K. R.: A Visual Attention Network for a Humanoid Robot. In: *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robotic Systems*. British Columbia, Kanada, 1998

[27] Enderle, Stefan ; Ritter, Marcus ; Fox, Dieter ; Sablatnög, Stefan ; Kraetzschmar, Gerhard K. ; Palm, Günther: Soccer-Robot Localization Using Sporadic Visual Features. In: Pagello, Enrico (Hrsg.) ; Groen, Frans (Hrsg.) ; Arai, Tamio (Hrsg.) ; Dillmann, Rdiger (Hrsg.) ; Stentz, Anthony (Hrsg.): *Intelligent Autonomous Systems 6 (IAS-6)*. Amsterdam, The Netherlands : IOS Press, 2000, S. 959 – 966

[28] Fay, Rebecca ; Kaufmann, Ulrich ; Knoblauch, Andreas ; Markert, Heiner ; Palm, Günther: Combining Visual Attention, Object Recognition and Associative Information Processing in a NeuroBotic System. In: Wermter, Stefan (Hrsg.) ; Palm, Günther (Hrsg.) ; Elshaw, Mark (Hrsg.): *Biomimetic Neural Learning for Intelligent Robots, Intelligent Systems, Cognitive Robotics, and Neuroscience* Bd. 3575. Berlin, Heidelberg : Springer-Verlag, 2005, S. 118 – 143

[29] Fay, Rebecca ; Kaufmann, Ulrich ; Markert, Heiner ; Palm, Günther: Adaptive Visual Attention Based Object Recognition. In: [114]

[30] Fay, Rebecca ; Kaufmann, Ulrich ; Schwenker, Friedhelm ; Palm, Günther: Learning Object Recognition in a NeuroBotic System. In: Gross, Horst-Michael (Hrsg.) ; Debes, Klaus (Hrsg.) ; Böhme, Hans-Joachim (Hrsg.): *3rd Workshop on SelfOrganization of AdaptiVE Behavior SOAVE 2004* Bd. 743. Düsseldorf : VDI Verlag, 2004, S. 198– – 209

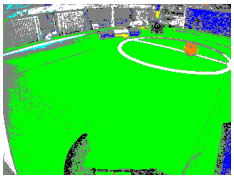
[31] Fay, Rebecca ; Schwenker, Friedhelm ; Palm, Günther: Incremental Learning in Hierarchical Neural Networks for Object Recognition. In: *2nd International Conference on Informatics in Control, Automation and Robotics ICINCO 2005* Bd. III, 2005, S. 298– – 303

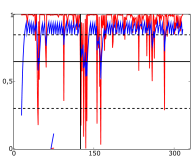
[32] Fernández, D. ; Parra, I. ; Sotelo, M. A. ; Bergasa, L. M. ; Revenga, P. ; Nuevo, J. ; Ocana, M. A.: Pedestrian recognition for intelligent transportation systems. In: Filipe, Joaquim (Hrsg.) ; Andrade-Cetto, Juan (Hrsg.) ; Ferrier, Jean-Louis (Hrsg.): *ICINCO*, INSTICC Press, 2005, S. 292 – 297

[33] Ferrein, Alexander ; Hermanns, Lutz ; Lakemeyer, Gerhard: Comparing Sensor Fusion Techniques for Ball Position Estimation. In: [15], S. 154 – 165

[34] Ferrein, Alexander ; Lakemeyer, Gerhard ; Schiffer, Stefan: AllemaniACs 2006 Team Description. In: [88]







[35] Forsyth, D.: A Novel Approach for Color Constancy. In: *International Journal of Computer Vision* 5 (1990), Nr. 1, S. 5 – 36

[36] Forsyth, D.: Sampling, resampling and colour constancy. In: *Proceedings Computer Vision and Pattern Recognition*, 1999, S. 300 – 305

[37] Franke, U. ; Gavrila, D. M. ; Görzig, S. ; Lindner, F. ; Paetzold, F. ; Wöhler, C.: Autonomous Driving Approaches Downtown. In: *IEEE Intelligent Systems* 13 (1998), Nr. 6, S. 40 – 48

[38] Freeman, W. ; Roth, M.: Orientation histograms for hand gesture recognition / Mitsubishi Electric Research Laboratorys, Cambridge Research Center. 1994 (94-03a). – Forschungsbericht

[39] Freund, Yoav ; Schapire, Robert E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *Second European Conference on Computational Learning Theory*, 1995

[40] Freund, Yoav ; Schapire, Robert E.: A Short Introduction to Boosting. In: *Journal of Japanese Society for Artificial Intelligence* 14 (1999), September, Nr. 5, S. 771 – 780

[41] Fuerstenberg, K. C. ; Dietmayer, K. C. J. ; Willhoeft, V.: Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In: *IEEE Intelligent Vehicles Symposium*, 2002, S. 31– 35

[42] Funk, Nathan: A Study of the Kalman Filter applied to Visual Tracking / University of Alberta, Department of Computing Science. 2003. – Forschungsbericht

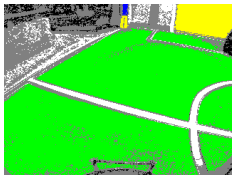
[43] Gavrila, D. M.: The Visual Analysis of Human Movement: A Survey. In: *Computer Vision and Image Understanding* 73 (1999), Nr. 1, S. 82 – 98

[44] Gavrila, D. M. ; Giebel, J. ; Munder, S.: Vision-based Pedestrian Detection: the PROTECTOR+ System. In: *P*roceedings of IEEE Intelligent Vehicles Symposium[62]

[45] Gavrila, D. M. ; Philomin, V.: Real-time Object Detection for Smart Vehicles. In: *Proceedings of IEEE International Conference on Computer Vision*. Kerkyra, Greece : IEEE Press, 1999, S. 87 – 93

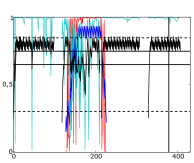


[46] Gonzalez, Rafael C. ; Woods, Richard E.: *Digital Image Processing*. Prentice Hall, 2003



[47] Grillo, Erio ; Matteucci, Matteo ; Sorrenti, Domenico G.: Getting the Most from Your Color Camera in a Color-Coded World. In: [101], S. 221 – 235

[48] Group, Object M.: Minimum CORBA Specification V1.0 / OMG. Framingham, MA : OMG Press, August 2002. – Forschungsbericht



[49] Group, Object M.: CORBA/IIOP Specification / OMG. Framingham, MA : OMG Press, April 2004. – Forschungsbericht

[50] Grubb, G. ; Zelinsky, A. ; Nilsson, L. ; Rilbe, M.: 3D vision sensing for improved pedestrian safety. In: *P*roceedings of IEEE Intelligent Vehicles Symposium[62], S. 19 – 24

[51] Gönner, Claudia ; Rous, Martin ; Kraiss, Karl-Friedrich: Robuste farbbasierte Bildsegmentierung für mobile Roboter. In: Dillmann, Rüdiger (Hrsg.) ; Wörn, Heinz (Hrsg.) ; Gockel, Tilo (Hrsg.): *AMS2003, 18. Fachgespräch Autonome Mobile Systeme*. Karlsruhe : Springer-Verlag, 2003 (Informatik aktuell), S. 64 – 74

[52] Gönner, Claudia ; Rous, Martin ; Kraiss, Karl-Friedrich: Real-Time Adaptive Colour Segmentation for the RoboCup Middle Size League. In: [101], S. 402 – 409

[53] Hanek, R.: The Contracting Curve Density Algorithm and its Application to Model-based Image Segmentation. In: *IEEE Conference Computer Vision and Pattern Recognition* 1 (2001), S. 797 – 804

[54] Hanek, R. ; Schmitt, T. ; Buck, S. ; Beetz, M.: Towards RoboCup without Color Labeling. In: [75], S. 179 – 194

[55] Hanek, Robert ; Schmitt, Thorsten ; Buck, Sebastian ; Beetz, Michael: Fast Image-based Object Localization in Natural Scenes. In: *I*nternational Conference on Intelligent Robots and Systems (IROS 2002)[63], S. 116 – 122

[56] Haykin, Simon S.: *Neural Networks : A Comprehensive Foundation*. Prentice Hall, 1998

[57] Hebert, M. (Hrsg.) ; Ponce, J. (Hrsg.) ; Boult, T. (Hrsg.) ; Gross, A. (Hrsg.): *Objekt Representation in Computer Vision*. Berlin : Springer-Verlag, 1995

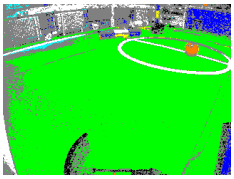
[58] Heinemann, P. ; Sehnke, F. ; Zell, A.: Towards a Calibration-Free Robot: The ACT Algorithm for Automatic Online Color Training. In: [88]

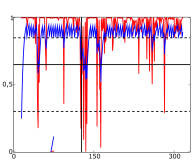
[59] Heisele, B. ; Ritter, W.: Obstacle Detection based on Color Blob Flow. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. Detroit, USA : IEEE Press, September 1995, S. 282 – 286

[60] Horn, B. K. P. ; Schunck, B. G.: Determining optical flow / Massachusetts Institue of Technology. 1980. – Forschungsbericht

[61] IEEE (Veranst.): *Proceedings of IEEE Intelligent Vehicles Symposium*. Stuttgart, Germany : IEEE Press, Oktober 1998







[62] IEEE (Veranst.): *Proceedings of IEEE Intelligent Vehicles Symposium*. Parma, Italy : IEEE Press, Juni 2004

[63] IEEE/RSJ (Veranst.): *International Conference on Intelligent Robots and Systems (IROS 2002)*. EPFL Lausanne, Switzerland : SPIE, September-Oktober 2002

[64] *Intel Performance Primitives (IPP)*. Mehr Informationen unter http://www.intel.com/software/products/perflib/, Oktober 2006

[65] Isard, Michael ; Blake, Andrew: Contour tracking by stochastic propagation of conditional density. In: *Proceedings of the European Conference of Computer Vision (ECCV'96)*. Cambridge, UK : Springer-Verlag, 1996 (Lecture Notes in Computer Science), S. 343 – 356

[66] Itti, L. ; Koch, C. ; Niebur, E.: A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Nr. 11, S. 1254 – 1259

[67] Jamzad, M. ; Sadjad, B. S. ; Mirrokni, V. S. ; Kazemi, M. ; Chitsaz, H. ; Heydarnoori, A. ; Hajiaghai, M. T. ; Chiniforooshan, E.: A Fast Vision System for Middle Size Robots in RoboCup. In: [11], S. 71 – 80

[68] Jamzad, Mansour ; Lamjiri, Abolfazal K.: An Efficient Need-Based Vision System in Variable Illumination Environment of Middle Size RoboCup. In: [106], S. 654 – 661

[69] Jobson, D. J. ; Rahman, Z. ; Woodell, G. A.: Properties and Performance of a Center/Surround Retinex. In: *IEEE Transactions on Image Processing* (1997), März

[70] Johansson, A. J. ; Rassmus-Gröhn, K. ; Balkenius, C.: Lighting of RoboCup Games / Lund University Cognitive Science, Sweden. 2002. – Forschungsbericht

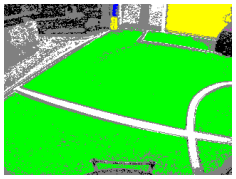
[71] Jähne, Bernd: *Digitale Bildverarbeitung*. Berlin : Springer-Verlag, 2005

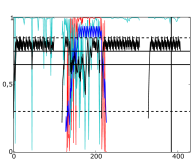
[72] Jüngel, Matthias: Using Layered Color Precision for a Self-Calibrating Vision System. In: [101], S. 209 – 220

[73] Jüngel, Matthias ; Hoffmann, Jan ; Lötzsch, Martin: A Real-Time Auto-Adjusting Vision System for Robotic Soccer. In: [106], S. 214 – 225

[74] Kalman, Rudolph E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME–Journal of Basic Engineering* 82 (1960), Nr. Series D, S. 35 – 45







[75] Kaminka, Gal (Hrsg.) ; Lima, Pedro (Hrsg.) ; Rojas, Raúl (Hrsg.): *Lecture Notes in Computer Science*. Bd. 2752: *RoboCup-2002: Robot Soccer World Cup VI*. Berlin, Heidelberg, Germany : Springer-Verlag, 2003

[76] Kaufmann, Ulrich: *Untersuchung, Bewertung und Nutzung von visuellen Merkmalen der Robotererkennung*. Germany, Computer Science Faculty, Department of Neural Information Processing, University of Ulm, Diplomarbeit, 2003

[77] Kaufmann, Ulrich ; Fay, Rebecca ; Schwenker, Friedhelm ; Palm, Günther: Neural Networks for Visual Object Recognition Based on Selective Attention. In: *1st International SenseMaker Workshop on Life-Like Perception Systems 2005*, 2005

[78] Kaufmann, Ulrich ; Mayer, Gerd ; Kraetzschmar, Gerhard ; Palm, Günther: Visual Robot Detection in RoboCup using Neural Networks. In: [101], S. 262 – 273

[79] Kestler, H. A. ; Simon, S. ; Baune, A. ; Hagenbuchner, M. ; Schwenker, F. ; Palm, G.: A Hierarchical Neural Object Classifier for Subsymbolic–Symbolic Coupling. In: Förster, W. (Hrsg.) ; Buhmann, J. M. (Hrsg.) ; Faber, A. (Hrsg.) ; Faber, P. (Hrsg.): *Mustererkennung 1999*. Berlin : Springer-Verlag, 1999, S. 27 – 35

[80] Kestler, H. A. ; Simon, S. ; Baune, A. ; Schwenker, F. ; Palm, G.: Object Classification Using Simple, Colour Based Visual Attention and a Hierarchical Neural Network for Neuro–Symbolic Integration. In: Burgard, W. (Hrsg.) ; Christaller, T. (Hrsg.) ; Cremers, A. B. (Hrsg.): *KI–99 Advances in Artificial Intelligence*. Berlin : Springer-Verlag, 1999, S. 267 – 279

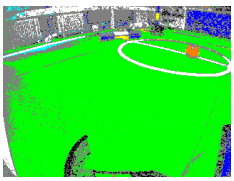
[81] Kestler, Hans A. ; Borst, Marcus ; Neumann, Heiko: Einfache Handgestikerkennung mit einem zweistufigen Nearest-neighbour Klassifikator / Universität Ulm. Ulm, Germany : Universität Ulm, 1999 (1999/6). – Forschungsbericht

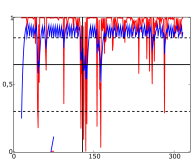
[82] Kikuchi, T. ; Umeda, K. ; Ueda, R. ; Jitsukawa, Y. ; Osumi, H. ; Arai, T.: Improvement of Color Recognition Using Colored Objects. In: [15], S. 537 – 544

[83] Kraetzschmar, Gerhard ; Mayer, Gerd ; Utz, Hans ; Baer, Philipp ; Claus, Martin ; Kaufmann, Ulrich ; Lauer, Markus ; Natterer, Simon ; Przewoznik, Sebastian ; Reichle, Roland ; Reisser, Alexander ; Roth, Axel ; Schmidt, Miriam ; Sitter, Christoph ; Sterk, Florian ; Palm, Günther: The Ulm Sparrows 2003. In: [106]

[84] Kraetzschmar, Gerhard ; Mayer, Gerd ; Utz, Hans ; Baer, Philipp ; Claus, Martin ; Kaufmann, Ulrich ; Lauer, Markus ; Natterer, Simon ; Przewoznik, Sebastian ; Reichle, Roland ; Sitter, Christoph ; Sterk, Florian ; Palm, Günther: The Ulm Sparrows 2004. In: [101]







[85] Kraetzschmar, Gerhard K. ; Utz, Hans ; Sablatnög, Stefan ; Enderle, Stefan ; Palm, Günther: Miro – Middleware for Cooperative Robotics. In: [11], S. 411 – 416

[86] Kwok, Cody ; Fox, Dieter: Map-Based Multiple Model Tracking of a Moving Object. In: [101], S. 18 – 33

[87] Labayrade, Raphaël ; Royere, Cyril ; Gruyer, Dominique ; Aubert, Didier: Cooperative Fusion for Multi-Obstacles Detection With Use of Stereovision and Laser Scanner. In: *Autonomous Robots* Bd. 19. Springer-Verlag Netherlands, September 2005, S. 117 – 140

[88] Lakemeyer, G. (Hrsg.) ; Sklar, E. (Hrsg.) ; Sorrenti, D. (Hrsg.) ; Takahashi, T. (Hrsg.): *RoboCup-2006: Robot Soccer World Cup X (in Druck)*. Berlin, Heidelberg, Germany : Springer-Verlag, 2006 (Lecture Notes in Computer Science)

[89] Lombardi, Paolo: A Survey on Pedestrian Detection for Autonomous Driving Systems / University of Pavia, Computer and Systems Engineering Department. 2001. – Forschungsbericht

[90] Lovell, Nathan: Illumination Independent Object Recognition. In: [15], S. 384 – 395

[91] Matarić, Maja J.: Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents. In: Sun, Ron (Hrsg.): *Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning* Bd. 2, 2001, S. 81 – 93

[92] Mayer, Gerd ; Kaufmann, Ulrich ; Kraetzschmar, Gerhard ; Palm, Günther: Neural Robot Detection in RoboCup. In: *27th German Conference on Artificial Intelligence (KI2004), NeuroBotics Workshop*. University of Ulm, 2004

[93] Mayer, Gerd ; Melchert, Jonas ; Utz, Hans ; Kraetzschmar, Gerhard ; Palm, Günther: Neural Object Classification and Tracking. In: [114]

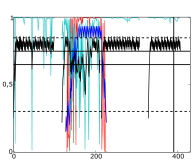
[94] Mayer, Gerd ; Utz, Hans ; Kraetzschmar, Gerhard: Playing Robot Soccer under Natural Light: A Case Study. In: [106], S. 238 – 249



[95] Mayer, Gerd ; Utz, Hans ; Kraetzschmar, Gerhard K.: Towards Autonomous Vision Self-Calibration for Soccer Robots. In: *I*nternational Conference on Intelligent Robots and Systems (IROS 2002)[63], S. 214 – 219



[96] Mayer, Gerd ; Utz, Hans ; Palm, Günther: Information Integration in a Multi-Stage Object Classifier. In: *19. Fachgespräch Autonome Mobile Systeme AMS 2005*. Stuttgart, Germany : Springer-Verlag, Dezember 2005 (Informatik Aktuell)



[97] Melchert, Jonas: *Video-based Motion Estimation and Object Tracking in RoboCup*. Germany, Computer Science Faculty, Department of Neural Information Processing, University of Ulm, Diplomarbeit, 2005

[98] Merke, A. ; Welker, S. ; Riedmiller, M.: Line Based Robot Localization under Natural Light Conditions. In: *European Conference on Artificial Intelligence Machine Learning (ECAI) 2004, Workshop on Agents in real-time and dynamic environments*, 2004

[99] Mitri, Sara ; Frintrop, Simone ; Pervölz, Kai ; Surmann, Hartmut ; Nüchter, Andreas: Robust Object Detection at Regions of Interest with an Application in Ball Recognition. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain, April 2005

[100] Mittal, Millind ; Peleg, Alex ; Weiser, Uri: MMX Technology Architecture Overview. In: *Intel Technology Journal* 1 (1997), Nr. 1

[101] Nardi, D. (Hrsg.) ; Riedmiller, M. (Hrsg.) ; Sammut, C. (Hrsg.) ; Santos-Victor, J. (Hrsg.): *Lecture Notes in Computer Science*. Bd. 3276: *RoboCup-2004: Robot Soccer World Cup VIII*. Berlin, Heidelberg, Germany : Springer-Verlag, 2005

[102] Nayar, Shree K.: Omnidirectional vision. In: *Proceedings of the Eight International Symposium of Robotics Research*. Hayama, Japan, Oktober 1997

[103] Pao, L. Y. ; Powers, R. M.: A Comparison of Several Different Approaches for Target Tracking with Clutter. In: *Proceedings of American Control Conference*. Denver, CO, Juni 2003, S. 3919 – 3924

[104] Papageorgiou, C. ; Evgeniou, T. ; Poggio, T.: A Trainable Pedestrian Detection System. In: *P*roceedings of IEEE Intelligent Vehicles Symposium[61], S. 241 – 246

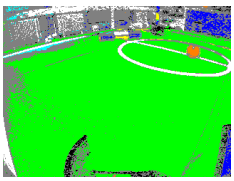
[105] Piccioli, G. ; Micheli, E. D. ; Parodi, P. ; Campani, M.: A Robust Method for Road Sign Detection and Recognition. In: Buxton, Bernard F. (Hrsg.) ; Cipolla, Roberto (Hrsg.): *4th European Conference on Computer Vision*. Cambridge, UK : Springer-Verlag, April 1996

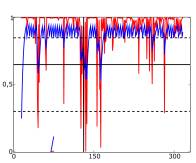
[106] Polani, Daniel (Hrsg.) ; Browning, Brett (Hrsg.) ; Bonarini, Andrea (Hrsg.) ; Yoshida, Kazuo (Hrsg.): *Lecture Notes in Computer Science*. Bd. 3020: *RoboCup-2003: Robot Soccer World Cup VII*. Berlin, Heidelberg, Germany : Springer-Verlag, 2004

[107] Pope, A.: *The CORBA Reference Guide*. Reading, MA : Addison-Wesley, 1997

[108] Rahman, Z. ; Jobson, D. J. ; Woodell, G. A.: Multiscale Retinex for Color Image Enhancement. In: *International Conference on Image Processing*, 1996







[109] Reichle, Roland ; Sterk, Florian ; Natterer, Simon: *Autonomer mobiler Fußballroboter Sparrow 2003.* Erhältlich unter http://smart.informatik.uni-ulm.de/Publications/Reichle_Sterk_Natterer_03.pdf, 2003

[110] Rojas, Raúl: *Neural Networks - A Systematic Introduction.* Springer-Verlag, 1996

[111] Rosenfeld, A.: Robot Vision. In: Wong, A. K. C. (Hrsg.) ; Pugh, A. (Hrsg.): *Machine Intelligence and Knowledge Engineering for Robotic Applications.* Berlin : Springer-Verlag, 1987, S. 1 – 19

[112] Schmitt, T. ; Hanek, R. ; Buck, S. ; Beetz, M.: Probabilistic Vision-based Opponent Tracking in Robot Soccer. In: [75], S. 179 – 194

[113] Schmitt, Thorsten ; Hanek, Robert ; Beetz, Michael ; Buck, Sebastian: Watch their Moves: Applying Probabilistic Multiple Object Tracking to Autonomous Robot Soccer. In: *Eighteenth National Conference on Artificial Intelligence.* Edmonton, Alberta, Canada, 2002

[114] Siddique, N. H. (Hrsg.) ; Amavasi, B. P. (Hrsg.) ; Hessami, A. G. (Hrsg.) ; IEEE Systems, Man and Cybernetics Society, United Kingdom and Republic of Ireland Chapter (Veranst.): *Proceedings of the IEEE SMC UK-RI Chapter Conference 2005 on Applied Cybernetics.* London, United Kingdom, September 2005

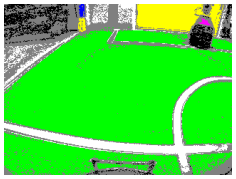
[115] Simon, S. ; Vogt, F. ; Kestler, H. A. ; Schwenker, F. ; Palm, G.: Aufmerksamkeitssteuerung für die Objektklassifikation auf einem autonomen mobilen Roboter. In: Baratoff, G. (Hrsg.) ; Neumann, H. (Hrsg.) ; Akademische Verlagsgesellschaft (Veranst.): *Dynamische Perzeption.* Berlin, 2000, S. 195–198

[116] Simon, Steffen ; Kestler, Hans A. ; Baune, Axel ; Schwenker, Friedhelm ; Palm, Günther: Werkzeuge zur visuellen Objektklassifikation / Universität Ulm. Ulm, Germany, 1999 (1999/3). – Forschungsbericht

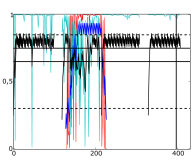
[117] Simon, Steffen ; Schwenker, Friedhelm ; Kestler, Hans A. ; Kraetzschmar, Gerhard K. ; Palm, Günther: Hierarchical Object Classification for Autonomous Mobile Robots. In: *Processing of international conference on artificial neural networks (ICANN).* Madrid, Spain, August 2002, S. 831 – 836



[118] Sitter, Christoph: *Design eines optischen Distanzsensors.* Erhältlich unter http://smart.informatik.uni-ulm.de/Publications/Sitter_04.pdf, 2004. – Computer Science Faculty, Department of Neural Information Processing, University of Ulm



[119] Solar, Javier R. ; Vallejos, Paul A.: Motion Detection and Tracking for an AIBO Robot Using Camera Motion Compensation and Kalman Filtering. In: [101], S. 619 – 627



[120] Sridharan, M. ; Stone, P.: Autonomous Planned Color Learning on a Legged Robot. In: [88]

[121] Sridharan, Mohan ; Stone, Peter: Towards Illumination Invariance in the Legged League. In: [101], S. 196 – 208

[122] Sridharan, Mohan ; Stone, Peter: Towards Eliminating Manual Color Calibration at RoboCup. In: [15], S. 673 – 681

[123] Steinbauer, Gerald ; Brandstötter, Mathias ; Buchleitner, Martin ; Galler, Stefan ; Jantscher, Simon ; Krammer, Gerlad ; Mörth, Martin ; Weber, Jörg ; Weiglhofer, Martin: Mostly Harmless Team Description 2006 - Robust Control of Mobile Robots. In: [88]

[124] Strey, A.: EpsiloNN — A Tool for the Abstract Specification and Parallel Simulation of Neural Networks. In: *Systems Analysis Modelling Simulation (SAMS), special issue on Simulation of Artificial Neural Networks, Gordon & Breach* 34 (1999), Nr. 4

[125] Strey, A.: A unified model for the simulation of artificial and biology-oriented neural networks. In: Mira, J. (Hrsg.) ; Sánchez-Andrés, J. V. (Hrsg.): *Engineering Applications of Bio-Inspired Artificial Neural Networks*, Springer-Verlag LNCS 1607, 1999, S. 1 – 10

[126] Suzuki, T. ; Kanada, T.: Measurement of vehicle motion and orientation using optical flow. In: *Proceedings of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*. Tokyo, Japan, 1999, S. 25 – 30

[127] Thakkar, Shreekant ; Huff, Tom: The Internet Streaming SIMD Extensions. In: *Intel Technology Journal, Pentium III Processor Overviews* 3 (1999), Mai, Nr. 2

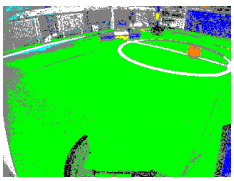
[128] Treptow, André ; Masselli, Andreas ; Zell, Andreas: Real-Time Object Tracking for Soccer-Robots without Color Information. In: *Proceedings of the European Conference on Mobile Robotics (ECMR 2003)*, 2003

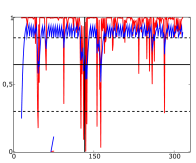
[129] Tsai, Roger Y.: An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. Miami Beach, FL, 1986, S. 364 – 374

[130] Tsai, Roger Y.: A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. In: *IEEE Journal of Robotics and Automation* RA-3 (1987), August, Nr. 4, S. 323 – 344

[131] Tsuji, Takayuki ; Hattori, Hiroshi ; Watanabe, Masahito ; Nagaoka, Nobuharu: Development of night-vision system. In: *IEEE Transactions on Intelligent Transportation Systems* 3 (2002), September, Nr. 3, S. 203– – 209







[132] Utz, Hans: *Advanced Software Concepts and Technologies for Autonomous Mobile Robots.* Germany, Computer Science Faculty, Department of Neural Information Processing, University of Ulm, Diss., 2005

[133] Utz, Hans ; Kaufmann, Ulrich ; Mayer, Gerd: *Advanced Video Image Processing on Autonomous Mobile Robots.* Edinburgh, Scotland : 19th International Joint Conference on Artificial Intelligence (IJCAI), August 2005. – Workshop on Agents in Real-time and Dynamic Environments

[134] *Kapitel* VIP: The Video Image Processing Framework based on the MIRO Middleware. In: Utz, Hans ; Kaufmann, Ulrich ; Mayer, Gerd ; Kraetzschmar, Gerhard: *Software Engineering for Experimental Robotics.* Springer Tracts on Advanced Robotics, 2006. – (in Druck)

[135] Utz, Hans ; Kaufmann, Ulrich ; Mayer, Gerd ; Kraetzschmar, Gerhard K.: VIP - A Framework-Based Approach to Robot Vision. In: *International Journal of Advanced Robotic Systems, Special Issue on Software Development and Integration in Robotics* 3 (2006), März, Nr. 1, S. 67 – 73

[136] Utz, Hans ; Mayer, Gerd ; Kraetzschmar, Gerhard: Middleware Logging Facilities for Experimentation and Evaluation in Robotics. In: *27th German Conference on Artificial Intelligence (KI2004), Workshop on Methods and Technology for Empirical Evaluation of Multiagent Systems and Multirobot Teams.* University of Ulm, 2004

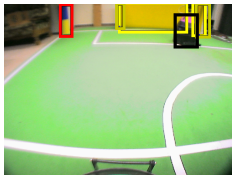
[137] Utz, Hans ; Mayer, Gerd ; Maschke, Dominik ; Neubeck, Alexander ; Schaeffer, Peter ; Baer, Philipp ; Baetge, Ingmar ; Fischer, Jan ; Holzer, Roland ; Lauer, Markus ; Reisser, Alexander ; Sterk, Florian ; Palm, Günther ; Kraetzschmar, Gerhard K.: The Ulm Sparrows 2001. In: [11], S. 677 – 680

[138] Utz, Hans ; Neubeck, Alexander ; Mayer, Gerd ; Kraetzschmar, Gerhard K.: Improving Vision-based Self-Localization. In: [75], S. 24 – 30

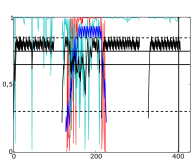
[139] Utz, Hans ; Sablatnög, Stefan ; Enderle, Stefan ; Kraetzschmar, Gerhard K.: Miro – Middleware for Mobile Robot Applications. In: *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures* 18 (2002), August, Nr. 4, S. 493 – 497

[140] Welch, G. ; Bishop, G.: An Introduction to the Kalman Filter / Department of Computer Science, University of North Carolina at Chapel Hill, USA. 2004. – Forschungsbericht

[141] Wilking, D. ; Röfer, T.: Real-time Object Recognition Using Decision Tree Learning. In: [101], S. 556 – 563







[142] Willersinn, D. ; Enkelmann, W.: Robust obstacle detection and tracking by motion analysis. In: *IEEE Conference on Intelligent Transportation System (ITSC 97)*, 1997, S. 717 – 722

[143] Wohler, C. ; Aulanf, J. ; Portner, T. ; Franke, U.: A Time Delay Neural Network Algorithm for Real-time Pedestrian Recognition. In: *P*roceedings of IEEE Intelligent Vehicles Symposium[61], S. 247 – 251

[144] Zagal, Juan C. ; Ruiz-del-Solar, Javier ; Guerrero, Pablo ; Palma, Rodrigo: Evolving Visual Object Recognition for Legged Robots. In: [106], S. 181 – 191

[145] Zhao, Liang ; Thorpe, Chuck: Stereo and Neural Network-based Pedestrian Detection. In: *IEEE Transactions on Intelligent Transportation Systems* 1 (2000), September, Nr. 3, S. 148 – 154



