HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
INSTITUT FÜR INFORMATIK

Visuelle Detektion humanoider Roboter basierend auf Histogrammen orientierter Gradienten

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

eingereicht von:	Dominik Krienelke
geboren am:	20.11.1986
geboren in:	Berlin
Gutachter/innen:	Prof. Dr. Verena Hafner
	Prof. Dr. Hans-Dieter Burkhard

eingereicht am: ..... verteidigt am: .....

Abstract

This work concerns the shape-based, visual object detection of robots in the context of the Standard Platform League (SPL) of the international RoboCup competition.

Within the RoboCup competition, color-based object detection plays a key role in the detection of opponents, the goals or the ball. In the future, a fixed coloring of objects will no longer be compulsory, hence a shape-based object detection has to be emphasized. In the field of machine vision, a classification based on histograms of oriented gradients (HOG) has established itself as the standard for pedestrian detection. Therefore this work examines Felzenszwalb's HOG based methodology (fHOG) to develop a visual, shape-based detection for the anthropomorphic Nao robot.

A total of three different variants of the method are examined. The fHOG method builds a cascade of classifiers upon a deformable parts model (DPM) of HOG features, which enables the classifier to detect even partly visible robots. The second variant reduces the cascade by successively discarding the weakest filters and for the third variant, the classifiers are subjected to regularization during training. The three variants are experimentally investigated for two application scenarios. Those scenarios are the detection of robots by an external camera, which is observing the whole field and the detection from the robot's point of view. Consequently, two databases are build, representing those applications. The quantitative evaluation of the detectors confirms HOG based algorithms as a feasible method for visual robot detection.

The created databases will be made available to the public in order to promote comparative work. Furthermore, ongoing research projects have begun to integrate HOG based classifiers in order to detect or track robots and to classify their posture.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der visuellen Objekterkennung in der Robotik. Es werden Verfahren für die formbasierte Detektion der Nao Roboter, dem offiziellen Roboter der Standard Platform League (SPL) des internationalen RoboCup Wettbewerbes, untersucht.

Im Kontext des RoboCup Wettbewerbes spielt die farbbasierte Objekterkennung eine tragende Rolle bei der Erkennung von Gegnern, der Tore oder des Balles. In Zukunft wird die feste Farbgebung nicht länger vorgegeben sein und eine formbasierte Objekterkennung rückt in den Vordergrund. Im Bereich des maschinellen Sehens hat sich die Klassifikation basierend auf Histogrammen orientierter Gradienten (HOG) als wichtiger Standard für die Erkennung von Fußgängern etabliert.

Mit dem Ziel einer schnellen, formbasierten Detektion wird das Verfahren auf den anthropomorphen Nao Roboter übertragen und es wird eine moderne Variante genannt fHOG umgesetzt. Insgesamt werden drei verschiedene Varianten des Verfahrens untersucht. Beim fHOG Verfahren werden die Merkmale in verformbare Einzelteile zerlegt. Eine entsprechend aufgebaute Klassifikatorkaskade detektiert ein Objekt auch dann, wenn es nur in Teilen sichtbar ist. In der zweiten untersuchten Variante wird die aufgebaute Klassifikatorkaskade reduziert, indem sukzessive die Filter mit der geringsten Gewichtung verworfen werden. Bei der dritten Variante werden die Klassifikatoren beim Training einer Regularisierung unterzogen.

In Experimenten werden die drei Verfahren am Beispiel der Nao Roboter evaluiert. Zur Durchführung der Experimente werden zwei Bild-Datenbanken von markierten Nao Robotern angelegt. Die beiden Datenbanken repräsentieren dabei zwei unterschiedliche Anwendungsszenarien. Eine Datenbank enthält Szenen eines Roboterfußballspiels aufgenommen mit einer externen Weitwinkelkamera aus der Vogelperspektive. Die andere Datenbank besteht aus Aufnahme aus der Ich-Perspektive des Roboters. Jede Variante wird hinsichtlich der Qualitätsmaße ihrer Detektoren empirisch analysiert. Die quantitative Auswertung bestätigt, dass sich auf fHOG basierende Algorithmen ohne Qualitätsverlust auf die visuelle Detektion von Robotern übertragen lassen.

Das Ergebnis der Arbeit sind die zwei Datenbanken von markierten Robotern, sowie eine Sammlung von Klassifikatoren und Detektoren. Die erstellten Datenbanken werden öffentlich verfügbar gemacht, um die Durchführung vergleichender Arbeiten zu fördern. Darüber hinaus hat die Einbindung von auf fHOG basierenden Klassifikatoren in laufende Forschungsprojekte und die Arbeit an einer integrierten Lösung auf der Roboterplattform begonnen. In den Projekten werden Roboter erkannt, verfolgt und Roboterposen werden klassifiziert.

Inhaltsverzeichnis

1 Einleitung

Das Ziel des internationalen Robotik-Wettbewerbes RoboCup ist es im Rahmen eines Fußballturniers zunehmend fortschrittlichere Forschungsaufgaben zu formulieren und zu lösen. Seit der Gründung des Projekts gibt es Bestrebungen bis Mitte dieses Jahrhunderts eine Mannschaft autonomer, humanoider Roboter zu stellen, die den amtierenden FIFA Fußballweltmeister in einem Fußballspiel schlagen kann. Für die Standard Platform League (SPL), in der die Arbeitsgruppe des Nao Team Humboldt (NaoTH) antritt, bedeutet dies, dass sich die besten und progressivsten Teams für Regeländerungen einsetzen, die den Robotern das Spiel erschweren und es gleichzeitig näher an das menschliche Vorbild heranführen. Möchte ein Team sich selber zu diesen progressiven Gruppen zählen, muss es mittelfristig nicht nur dem aktuellen Regelwerk gerecht werden, sondern aktiv den Stand der Technik voranbringen, zukünftige Regeländerungen vorausahnen und schließlich herbeiführen.

In der absehbaren Zukunft wird es Regeländerungen hinsichtlich der standardisierten Farbgebung verschiedener Objekte geben. Die gelben und blauen Tore wurden bereits abgeschafft, ebenso der uniform orangene Ball. Es ist vorstellbar, dass die klar unterscheidbaren, einfarbigen Trikots in Zukunft durch eine Mannigfaltigkeit an Färbungen, Mustern und Logos ersetzt werden. Beim menschlichen Fußballspiel wird beinahe jede gut sichtbare Fläche auch als Werbetafel genutzt. Darüber hinaus erlaubt das Regelwerk neuerdings den Einsatz eines Roboters als Trainer, der außerhalb des Spielfeldes stehend das Geschehen beobachten und an die Spieler kommunizieren kann.

Eine Objekterkennung über komplexe Formen wird in Zukunft also eine zunehmend tragende Rolle spielen und die bisher vorherrschende Detektion über die Farbe und fundamentale Geometrie ergänzen oder ersetzen. Für die Arbeitsgruppe ist es daher erstrebenswert, verschiedene Ansätze zu vergleichen und schließlich ein flexibel einsetzbares Verfahren zu erarbeiten, um im Rahmen des RoboCups prävalente Objekte an Hand der Form zuverlässig zu erkennen und wiederzufinden.

1.1 Ziel und Aufbau der Arbeit

Das maßgebliche Ziel dieser Arbeit ist die Entwicklung, Ausarbeitung und Evaluation einer visuellen, formbasierten Roboterdetektion. Nach erfolgreicher Suche des geeigneten Modells, soll das Resultat eine fertige Implementierung sein, die in Projekten und Wettbewerben im Umfeld des RoboCups praktischen Einsatz finden kann und dabei die Erkennung von Nao Robotern ermöglicht.

Als sekundäres Ziel versteht sich dieses Werk auch als Vorbereitung weiterführender Arbeiten und will eine Grundlage schaffen, auf welcher die auf Interest-Operatoren

beruhende Objekterkennung im Kontext des RoboCups erforscht werden kann. Diese Anforderung findet sich im Besonderen in den theoretischen Teilen reflektiert, deren Aufbereitung über die reine Vorbereitung der Implementierung hinausgeht.

Die vorliegende Arbeit gliedert sich in die folgenden Abschnitte: In den ersten Kapiteln wird zunächst eine theoretische Basis geschaffen, indem der aktuelle Forschungsstand, sowie die in späteren Kapiteln praktisch umgesetzten Konzepte, Modelle und Begrifflichkeiten detailliert vorgestellt werden. Als erster Schritt auf der Suche nach einer passenden Lösung, wird die Auseinandersetzung mit der Problemstellung in einem breiten wissenschaftlichen Kontext verstanden. Nachdem die der Arbeit zugrundeliegenden Ziele und Rahmenbedingungen (Kapitel 1.2) einleitend beschrieben wurden, wird mit der Vorstellung thematisch relevanter Werke (Kapitel 1.3) begonnen.

Im zweiten Schritt werden im folgenden Kapitel die notwendigen elementaren Grundlagen des maschinellen Sehens aufgearbeitet und vorgestellt (Kapitel 2.1), um dann in späteren Abschnitten die speziell im Kontext des RoboCups und der humanoiden Robotik vielversprechendsten Verfahren der Merkmalsextraktion im Detail untersuchen zu können. Beim maschinellen Sehen hat sich die Klassifikation mittels Histogrammen orientierter Gradienten (HOG, aus dem Englischen Histograms of Oriented Gradiants) als wichtiger Standard für die Erkennung von Fußgängern etabliert. Aufgrund der anthropomorphen Gestalt des Naos, ist anzunehmen, dass sich diese Methode auch zur Erkennung von Naos bzw. allgemein humanoider Roboter eignet. Es wird daher der Fokus auf HOG und verwandte, robuste, lokale Detektoren wie die skaleninvariante Merkmalstransformationen (SIFT, aus dem Englischen Scale-invariant feature transform) gelegt (Kapitel 2.2, 2.3). In der Praxis ist häufig die Gesamtleistung der Objektklassifizierung das ausschlaggebende Qualitätsmerkmal. Diese hängt neben der Geschwindigkeit der Merkmalsextraktion auch von der Zuverlässigkeit, der Genauigkeit und der Wiederholbarkeit der extrahierten Merkmale, so wie von den eingesetzten Klassifikatoren und deren Qualitäten ab. Da der Einsatz eines Algorithmus im Rahmen des RoboCups oft sowohl eine qualitative als auch eine zeitkritische Anwendung ist, wird mögliches Optimierungspotential beim Training und bei der Verwendung von Klassifikatoren im letzten Abschnitt des Kapitels untersucht (Kapitel 2.4).

Aufbauend auf dieser theoretischen Vorarbeit im Bereich des maschinellen Sehens, werden im Weiteren verschiedene Algorithmen zur Detektion humanoider Roboter praktisch umgesetzt (Kapitel 3). Zunächst werden die drei ausgewählten Lösungsverfahren beschrieben. Im ersten Abschnitt des Kapitels werden lösungsspezifische Algorithmen vorgestellt (Kapitel 3.1). Anschließend wird das Training der Klassifikatoren, die bei der Detektion zum Einsatz kommen, dargelegt (Kapitel 3.2). Der letzte Teil des Kapitels widmet sich dann den praktischen Details bei der Implementierung der Verfahren (Kapitel 3.3). Es werden die Technologien präsentiert, die bei der Implementierung effektiv zum Einsatz kamen und erklärt, warum diese zwischen vergleichbaren Technologien in der Praxis ausgewählt wurden (Kapitel 3.3.1). Zu den betrachteten, realisierten Implementierungsdetails

zählt exemplarischer Pseudocode, der den entstandenen Programmcode repräsentiert, Speicherüberlegungen, sowie das Laden und protokollieren von Trainingssitzungen (Kapitel 3.3.2 bis 3.3.6).

Der praktischen Umsetzung verschiedener Detektoren folgt die empirische Untersuchung der selbigen (Kapitel 4). In einer Reihe von Experimenten, werden die einzelnen Varianten getestet und anschließend miteinander verglichen. Für den speziellen Anwendungsfall der Detektion humanoider Roboter, wurden zwei Datenbanken mit Bildern von Nao Robotern erstellt, die den Algorithmen als Trainings- und Testbeispiele dienen. Diese Datenbanken werden im ersten Abschnitt des Kapitels vorgestellt (Kapitel 4.1). Die durchgeführten Experimente bestehen im Wesentlichen aus erschöpfenden Suchen in ausgewählten Parameterräumen. Die drei implementierten Verfahren werden auf den zwei Datenbanken trainiert und die ermittelten Detektoren werden einer Kreuzvalidierung unterzogen (Kapitel 4.2, 4.3). Auf Basis der gemessenen Genauigkeit und Sensitivität werden objektive Qualitätsmaße wie die F- und G-Maße bestimmt (Kapitel 4.4).

Mit den ermittelten Qualitätsmaßen wird die Auswertung der empirischen Analyse vorgenommen (Kapitel 5). Jedem Verfahren wird ein Abschnitt gewidmet, in welchem die ermittelten Qualitätsmaße grafisch aufgearbeitet und die Ergebnisse verschiedener Parametrisierungen innerhalb der jeweiligen Lösungsvariante miteinander verglichen werden (Kapitel 5.1, 5.2, 5.3). Anschließend werden die Varianten zueinander in Relation gesetzt, indem für die am besten abschneidenden Klassifikatoren vollständige PR-Kurven berechnet und einander gegenübergestellt wurden (Kapitel 5.4). Das Fazit gibt Auskunft darüber, in welcher Form und mit welcher zu erwartenden Leistung die entstandenen Detektoren in Zukunft eingesetzt werden können. Es werden die Vor- und Nachteile, die möglichen Einsatzgebiete und die Problemfelder der verschiedenen Detektoren diskutiert (Kapitel 5.5).

Im letzten Kapitel werden die Ergebnisse und die gewonnenen Erkenntnisse dieses Werkes zusammengefasst (Kapitel 6.1). Es wird eine Einschätzung darüber gegeben, in welchen Bereichen in kommenden Arbeiten die interessantesten und vielversprechendsten Fortschritte erzielt werden können und zukünftige Projekte, die sich auf diese Arbeit beziehen, werden in Aussicht gestellt. Außerdem werden erste durchgeführte Experimente zur auf HOG basierenden Balldetektion, zur Klassifikation von Posen und zum Tracking kurz vorgestellt. (Kapitel 6.2, 6.3).

1.2 Rahmen der Arbeit

Zum Verständnis der Motivation und der Rahmenbedingungen bei der Ausarbeitung der Lösung wird in diesem Abschnitt das Arbeitsumfeld beschrieben, dem das Thema entstammt.

Der in der Einleitung bereits erwähnte RoboCup ist ein 1997 gegründeter Verband zur Förderung der Forschung in den Feldern der Robotik und der Künstlichen Intelligenz. Der

Grundgedanke war ein gleichermaßen öffentlichkeitswirksames wie wissenschaftlich anspruchsvolles Forschungsprojekt ins Leben zu rufen, weshalb im Zentrum des RoboCups die Roboterfußball-Wettbewerbe stehen. Im Fußballspiel müssen die Roboter komplexe Bewegungen koordinieren, sie müssen in einer hochdynamischen Umgebung autonom agieren und sind limitiert bezüglich ihrer Sensorik und Rechenkapazität. Roboterfußball ist daher ein ergiebiges Aufgabenfeld in der Entwicklung intelligenter Roboter.

Ursprünglich formuliertes Langzeitziel der Initiative ist es bis 2050 den amtierenden Fußballweltmeister in einem fairen Fußballspiel zu schlagen. Fair in diesem Kontext heißt, die spielenden Roboter agieren autonom und sind annähernd humanoid. Das Fußballspiel stellt die Roboter vor die Aufgabe in einer sich dynamisch verändernden Umgebung mit unvollständigem Wissen zu planen und zu handeln. Die gewonnenen Erkenntnisse sind grundlegend und lassen sich auf vielerlei Anwendungsgebiete außerhalb des RoboCups übertragen. Im Rahmen diverser Veranstaltungen und Wettbewerbe werden verschiedenen Forschungszweige zusammengeführt. Neben der jährlich ausgetragenen Weltmeisterschaft findet eine Zahl von lokalen Meisterschaften und assoziierter Treffen statt. Mittlerweile wurde der RoboCup um weitere Ligen ergänzt, die den unmittelbaren Nutzen von mobilen, autonomen Robotern im Haushalt, in gefährlichen Umgebungen oder im industriellen Umfeld betonen.

Das NaoTH ist ein Projekt der Arbeitsgruppe Künstliche Intelligenz der Humboldt-Universität zu Berlin, das an verschiedenen Wettbewerben des RoboCups teilnimmt. Die Forschungsschwerpunkte liegen im Roboterfußball, teilgenommen wird in der SPL und in der 3D Simulationsliga. In der Ausarbeitung der praktischen Teile dieser Arbeit findet das NaoTH-Framework Verwendung. Diese Softwarepalette, die in den letzten Jahren im Rahmen der Teilnahme am RoboCup durch das NaoTH entwickelt wurde, stellt neben einer Vielzahl an Modulen zur Steuerung des Roboters selber, auch umfangreiche Werkzeuge zur Analyse und zur Weiterverarbeitung der von den Robotern generierten Daten zur Verfügung.

Die SPL (Standard Platform League) zeichnet sich durch die Verwendung einer einheitlichen Roboterplattform aus. Im Vergleich mit anderen RoboCup-Ligen, in denen humanoide Roboter zum Fußballspiel gegeneinander antreten, verschiebt sich durch die Vereinheitlichung der Plattform der Fokus von der Ausarbeitung der Roboter-Hardware auf die Entwicklung der Software. Seit 2009 ist der Nao Roboter die exklusive Plattform der SPL. Der Nao Roboter wurde von Aldebaran Robotics (mittlerweile Teil der Softbank Gruppe, Tokio) entwickelt. Der zweibeinige Nao Roboter befindet sich bereits in der fünften Iteration und verfügt über 25 Freiheitsgrade, 4 Mikrofone, je 2 Ultraschall-Emfänger und Sender, 9 Tastsensoren, 8 Kraftsensoren, diverse Schnittstellen und 2 hochauflösende Kameras. Die Kameras sind im beweglichen Kopf des Roboters verbaut. Genauere Spezifikationen zu den Vorgaben der SPL als auch zu der Plattform Nao können den Anlagen A.1 und A.2 entnommen werden.

1.3 Vorausgehende Arbeiten

Vor der praktischen Umsetzung einer Detektion humanoider Roboter steht die Vorstellung thematisch relevanter Werke, sowie der notwendigen mathematischen Voraussetzungen, die zur finalen Implementierung geführt haben. Im folgenden Abschnitt werden die untersuchten Verfahren in einen möglichst breiten, wissenschaftlichen Kontext gesetzt.

Eine der ersten Arbeiten, die sich explizit mit dem Einsatz von HOG Merkmalen innerhalb des RoboCups auseinandersetzt, ist ein 2013 im Rahmen der International Conference on Humanoid Robots eingereichtes Paper von Estivill-Castro und Radev[1]. Die Autoren prognostizieren korrekt, dass diverse Hilfestellungen zur Objekterkennung, wie die unifarbene Kolorierung des Balles, aufgehoben würden und wollen daher frühzeitig eine Objekterkennung an Hand der Form in ihrem Modell evaluieren. Die Wahl fällt auf HOG gegenüber SIFT und es wird auf die robusten Eigenschaften bei der Erkennung anthropomorpher Formen verwiesen. [2] Weiterhin merken die Autoren an, dass Techniken des maschinellen Sehens zur Erkennung komplexer Formen in der Regel wesentlich rechenaufwendiger sind, als solche zur Erkennung und Unterscheidung basierend auf Farben. Sie kommen daher zu dem Kompromiss HOG einzusetzen, um in einer dedizierten Initialisierungsphase Nao Roboter zu erkennen, dann aber die Trikotfarben zu lernen, um im späteren Spiel auf die erprobte, farbbasierte Erkennung zurückzugreifen. Die Roboter werden zuverlässig erkannt und die manuelle Farbkonfiguration kann somit automatisiert werden, der Einsatz von HOG im Spiel steht aber noch aus.

Um eine realistische Erwartungshaltung an den Detektor zu entwickeln, ist es sinnvoll, sich mit vergleichenden Arbeiten zu beschäftigen, die die Qualitäten verschiedener moderner Algorithmen der Fußgängererkennung in Relation setzen. Dollár et al. haben dazu 2012 die Studie „Pedestrian Detection An Evaluation of the State of the Art“ [3] veröffentlicht. Die Studie vergleicht 16 repräsentative Detektoren auf 6 öffentlichen Datenbanken und attestiert den besten Detektoren noch stark ausbaufähige Sensitivität. Selbst bei guten Bedingungen werden nur 70-80% der Fußgänger erkannt, wenn als Einschränkung eine falschpositive Erkennung auf 10 Bilder festgelegt wird. Wird darüber hinaus die Erkennung im Fernbereich, also von Fußgängern von unter 80 Pixeln Größe betrachtet, erhält man eine praktisch unverwertbare Erkennungsrate von 20% oder schlechter. Vergleichbare Resultate erzielen die Detektoren auf Bildern in denen die Fußgänger teils stark verdeckt sind, hier erreichen die besten Detektoren eine Erkennungsrate von 35%. Es wird vorgeschlagen die Forschung in die Richtungen Skalierung, Okklusion, bessere Merkmale, bessere Trainingsdaten, Bewegungserkennung und kontextsensitive Zusatzinformationen auszubauen.

Bereits zwei Jahre später evaluieren Benenson et al. in einer Arbeit [4] die Fortschritte der letzten Jahre in über 40 Detektoren auf 7 der am meisten verbreiteten Datenbanken. Die Familie der HOG-Merkmalsdetekoren ist nach wie vor prävalent, wobei die Merkmale selber in den besten Detektoren um Bewegungs- oder Farbinformationen erweitert worden sind. Die Anwendung des Deformable Parts Model (DPM) [5, 6], also die Zerlegung der

Merkmale in verformbare Einzelteile, erzielt gute Ergebnisse bei der Verdeckungsproblematik. Deep Learning Architekturen und Entscheidungswälder bieten keinen empirisch belastbaren Fortschritt gegenüber DPM. Ebenso verhält es sich mit nicht-linearen SVM-Kernels (Stützvektormaschinen, Support Vector Machines) gegenüber linearen SVMs. Skalierende Lösungen werden als gute, generische Methode betrachtet, um den erfolgreichen Einsatzbereich der Detektoren zu verbessern. Der Einsatz von Kontext- und Domänenwissen wird im Besonderen hervorgehoben und scheint für beste Ergebnisse ebenso unabdingbar zu sein, wie spezifisch nach Problemstellung ausgewählte Merkmale und Trainingsdaten. Damit sind bei guten Bedingungen mittlerweile Erkennungsraten von über 90% aller Fußgänger bei einer falsch positiven Erkennung auf 10 Bilder möglich.

Das DPM, wie es heute in der Fußgängerdetektion Verwendung findet, wurde maßgeblich durch eine von Felzenszwalb et al. im Rahmen der IEEE CVPR Konferenz 2010 eingereichte Arbeit geprägt. [6]. Unter Verwendung dieses DPM bleibt die Prozesskette äußerlich unverändert. In den Trainingsdaten werden die Objekte als Ganzes markiert und später in der Detektion als Ganzes auch wiedergefunden. Intern wird ein Objekt nun als variable Hierarchie von Bildteilen strukturiert, die lose und flexibel miteinander verknüpft sind. Aufbauend auf den grundlegenden Arbeiten von Dalal und Triggs [7] dient HOG der Erstellung einer Kaskade von Objektmerkmalen der einzelnen Bildteile. Als Wurzeldetektoren der Kaskade dienen hierbei noch der unmodifizierte HOG-Detektor, das Ergebnis der Kaskade wird aber von den Teildetektoren beeinflusst. Für die Detektion eines Fußgängers bedeutet dies, dass ein verdeckter Arm etwa nicht sofort das gesamte Objekt disqualifiziert, ein erkannter Arm dagegen die Zuversicht in die Detektion verbessert. Es spielt dabei kaum eine Rolle wo der Arm im Objekt gefunden wird, ob auf Kopfhöhe oder neben den Beinen pendelnd. Weiterhin können die konkreten Attribute der einzelnen Detektoren variieren, so kann die Auflösung des HOG-Merkmalsbildes des Wurzeldetektors kleiner sein, als die der Teildetektoren.

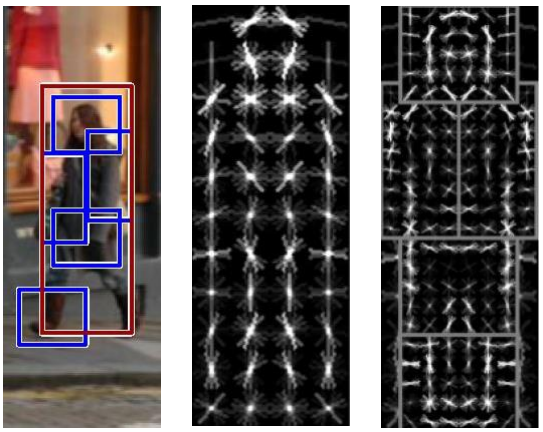


Abbildung 1.1: *Resultate eines DPM-basierten HOG-Detektors. Links: Beispielhafte Erkennung durch den Detektor, rot markiert ist die eigentliche Erkennung, blau markiert sind die Detektionen der Kaskade. Mitte: Ausgabe der HOG-Merkmale des grobaufgelösten Wurzelfilters, Rechts: Darstellung der Ansammlung von gelernten Teilfiltern, räumlich in Relation gesetzt*[1]

Eine andere Form die Detektion mittels HOG-Merkmalen zu verbessern wurde 2006 von Zhu et al. vorgestellt. [8] Auch hier wird eine Klassifikatorkaskade aufgebaut (vgl. Abbildung 1.1), allerdings mit dem expliziten Ziel die Detektion zu beschleunigen. Die per AdaBoost-Verfahren aufgebaute Kaskade verwendet Merkmalsbilder variabler Blockgrößen, um möglichst schnell die Bildbereiche zu verwerfen, in denen keine Detektionen zu erwarten sind. Müssen beim ursprünglichen Verfahren, bei dem ein Erkennungsfenster über das ganze Bild geschoben wird, unter den von Dalal und Triggs verwendeten Parametern, noch 105 Teilbereiche ausgewertet werden, so gelingt es Zhu et al. bei vergleichbarer Erkennungsqualität die Arbeit im Schnitt auf 4,6 Bereiche zu reduzieren und damit das Verfahren um den Faktor 20 zu beschleunigen.

Mögliche Verbesserungen der Merkmale selber werden unter anderem in zwei Arbeiten von Gary Overett und Lars Petersson [9, 10] beschrieben. Unter besonderer Berücksichtigung der Integralbilder, der Charakteristiken der HOG-Merkmale und möglicher schwacher Klassifizierer werden die Speicher- und Leistungsengpässe identifiziert. Als in dieser Hinsicht optimiert werden die LiteHOG-Merkmale vorgestellt, bei denen eine Diskriminanzanalyse der Orientierungshistogramme durchgeführt wird und komplexere schwache Klassifizierer eingesetzt werden. Die Größe des HOG-Merkmalsvektors kann durch lokalitätserhaltene Projektion (LPP) reduziert werden. [11] Nach den Autoren von LPP-HOG soll damit die Auswertung durch die Klassifizierer beschleunigt werden ohne dass die Erkennungsrate negativ beeinflusst wird.

Zusätzliche Invarianz gegenüber verschiedenen Beleuchtungsszenarien kann durch den Einsatz von farbbasierten Gradienten erzielt werden. [12] Bei dieser Methode wird das Gradientenbild als euklidischer Abstand über die Ableitungen der einzelnen Farbkanäle

[1] Darstellung unverändert übernommen aus 6. Felzenszwalb, P.F.G., R. B.; McAllester, D.; Ramanan, D., *Object detection with discriminatively trained part-based models.* IEEE transactions on pattern analysis and machine intelligence, 2010. **32**(9): p. 1627-1645.

berechnet und so nicht nur der Einfluss der Beleuchtung, sondern auch der des oft vorteilhaften Eigenschattens reduziert.

Alternativ zum DPM können lokale, binäre Muster (im Englischen Local Binary Patterns, LBP) verwendet werden. [13] Es werden dabei parallel zu den Gradientenbildern auch Integralbilder über den LBPs des Eingangsbildes erzeugt, die später verwendet werden, um die Textur eines Erkennungsfensters zu untersuchen. Stark segmentierte Regionen werden verworfen und bei unsicheren Detektionen können Unterbereiche mit positiver Antwort an einen Körperteildetektor weitergereicht werden.

Ein Hauptaugenmerk bei der praktischen Umsetzung der Roboterdetektion wurde auf die Modellauswahl und die Hyperparameteroptimierung des Klassifikators bzw. des HOG-Detektors gelegt. Beim Design der Experimente zur Modellauswahl via k-facher Kreuzvalidierung erwiesen sich insbesondere die Beschreibungen von Ricardo Gutierrez-Osuna [14], sowie bei der Auswahl der Wertebereiche der Parameter, dienten die Darlegungen aus „A Practical Guide to Support Vector Classification“ [15] als praxisnahe Vorlagen.

Ein exemplarisches System zur kooperativen Erkennung und Verfolgung von Objekten in der RoboCup SPL wird 2012 von Marchant et al. vorgeschlagen. [16] Diese Umsetzung verwendet die am Spiel teilnehmenden Roboter als verteiltes System und führt für die visuellen Wahrnehmungen und die Sonardetektionen eine Sensordatenfusion in ein gemeinsames Modell durch.

Eine aktuelle Publikation mit direktem Bezug zur vorliegenden Arbeit wurde beim 20th RoboCup International Symposium eingereicht. [17] Farazi und Behnke beschreiben ihre Umsetzung eines HOG-basierten Algorithmus zur Verfolgung humanoider Roboter. Die Autoren weisen auf die besondere Herausforderung beim Verfolgen und identifizieren von mehreren untereinander identisch aussehenden Humanoiden hin. Für die initiale Detektion der Roboter wurden zunächst mehrere konkurrierende Ansätze getestet, unter denen sich eine auf HOG-Merkmalen basierende Klassifikatorkaskade durchsetzen konnte. Beim Training der Kaskade werden die Eingabebilder nebst anderen Transformationen um bis zu $\pm 15°$ rotiert, um zumindest eine grundlegende Rotationstoleranz herbeizuführen und es wird darauf hingewiesen, dass die Erkennung verschiedener Posen wie Sitzen oder Liegen vorstellbar sind. Zur Verfolgung der Roboter wird zusätzlich ein Multiklassen-Detektor trainiert, der anhand von HOG die ungefähre Ausrichtung der Roboter bestimmt. Außerdem findet eine Kommunikation zwischen Robotern und Beobachter statt, bei der sich die Roboter als aktiv melden und ihre lokale Bewegung bekannt geben, umgekehrt berichtet auch der Beobachter seine Schätzungen an die Roboter. Über diese Rückkopplung sollen zukünftig die lokalen Modelle und das externe Modell des Beobachters synchronisiert werden. Insbesondere wird so auch die kontinuierliche Identifikation der Roboter im Bild ermöglicht.

2 Theoretische Grundlagen

Das folgende Kapitel beschreibt die zugrundeliegenden mathematischen Modelle, der in späteren Kapiteln umgesetzten Implementierung einer Roboterdetektion. Weiterhin dient dieses Kapitel dem NaoTH als Referenz im Umgang mit Interest-Operatoren im Bereich des maschinellen Sehens.

Es werden zuerst elementare Grundlagen der maschinellen Bildverarbeitung wie Filter und Kantendetektionen herausgearbeitet und anschließend Methoden zur Merkmalsextraktion vorgestellt. Ein besonderer Stellenwert wird dabei robusten, lokalen Operatoren basierend auf HOG und dem ähnlichen, skaleninvarianten Merkmalsextraktor SIFT eingeräumt. Im letzten Abschnitt des Kapitels werden Klassifikatoren und Verfahren zur Optimierung von diesen besprochen.

Die Bildverarbeitung ist ein tragendes Element des Forschungsgebietes des maschinellen Sehens. Die Bildverarbeitung beschreibt verschiedene fundamentale Techniken, die eine schnelle Analyse von Bildern zulassen, gleichzeitig bietet sie aber auch hochkomplexe Operationen, um z.B. eine subtile Charakteristik in einem Bild zu finden und differenziert zu beschreiben. Die für die Verarbeitung eines Bildes verwendeten Verfahren reichen von der Pixelanalyse und -manipulation über die Bildwiederherstellung und das Extrahieren von Merkmalen bis hin zur Klassifizierung von der Bewegung und den Merkmalen in einer Sequenz von Bildern.

Im Prozess des maschinellen Sehens durchläuft ein Bild die folgenden Schritte: Aufnahme, Vorverarbeitung, Segmentierung, Merkmalsextraktion, Klassifizierung.

Die theoretischen Grundlagen dieser Schritte werden in den nächsten Kapiteln beschrieben. Zur vereinfachten Darstellung wird bei den kommenden Erklärungen nur über einen Farbkanal gesprochen. Dies entspricht z.B. einem Graustufenbild oder eben einem Farbkanal eines Farbbildes. Die Definitionen und Prinzipien lassen sich generell aber auf beliebig viele parallele Farbkanäle anwenden.

2.1 Filter und Kantendetekion

Zur Vorverarbeitung eines Bildes gehört neben der Beseitigung von unerwünschtem Rauschen oder Artefakten auch die Bestimmung von Gradientenbildern. Ist ein Bild entsprechend prozessiert, lassen sich anschließend effizient Merkmale aus diesem extrahieren. Beide Aufgaben lassen sich durch das Filtern des Bildes bewältigen.

Filter in der Bildverarbeitung sind sogenannte Faltungsmatrizen. Es handelt sich dabei meist um quadratische Matrizen unterschiedlicher Größe. Da Bildverarbeitungsoperationen häufig als lineares System dargestellt werden können, lassen sich diese für diskrete, zweidimensionale Funktionen, wie es zweidimensionale Bilder sind, als folgende Formel über die diskrete Faltung beschreiben:

$$I^*(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{n} I\,(x+i-a+1, y+j-a+1)k(i,j) \tag{2.1}$$

$I^*(x,y)$ ist hier das Ergebnispixel, I ist das Bild, auf welches der Filterkernel angewandt wird, a ist die Koordinate des Mittelpunktes in der quadratischen Faltungsmatrix, und $k(i,j)$ ist ein Element der Faltungsmatrix.

2.1.1 Glättungsfilter

Glättungsfilter bereinigen ein Bild vom Rauschen und unterdrücken feine Details. Rauschen als zufällige Veränderung der Helligkeitswerte einzelner Bildpunkte lässt sich durch Mittelwertbildungen innerhalb eines Bildbereichs gut unterdrücken, weshalb Glättungsfilter in der Regel auf der Berechnung ungewichteter oder gewichteter Mittelwerte basieren.

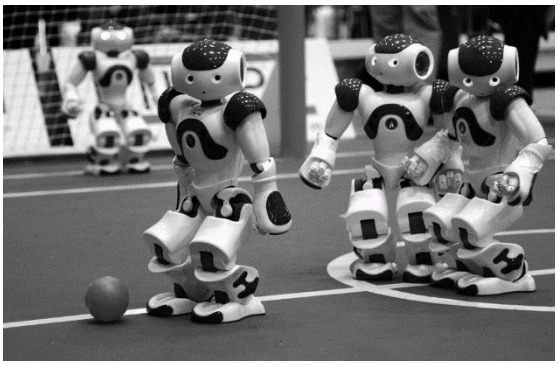
Mittelwertfilter: Die Faltung eine universelle Technik. So lässt sich die Berechnung des Mittelwertes μ eines Bildausschnitts

$$\mu = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} I(i,j) \tag{2.2}$$

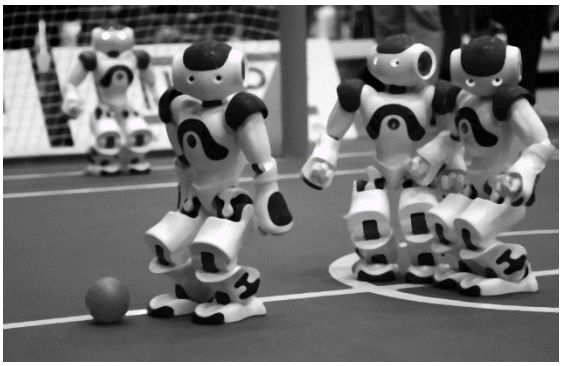
leicht mit einer Faltungsmatrix bestimmen. Für $n = 3$ wäre diese:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.3}$$

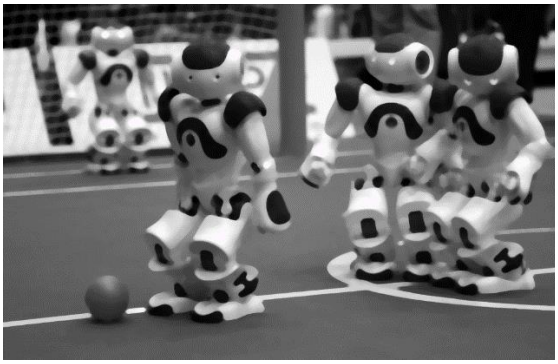
Mit dieser Matrix wird also jeder Bildpunkt zu gleichen teilen gewichtet. Je nach Faltungsmatrix kann ein Filter auf eine beliebig große Umgebung wirken. In der Regel gilt, je größer die Umgebung desto mächtiger wirkt der Filter (vgl. Abbildung 2.1). Zur Berechnung eines Ergebnisbildes bewegt sich diese Umgebung Pixel für Pixel über das Ausgangsbild wie in Abbildung 2.3 gezeigt, weshalb sie auch Sliding-Window genannt wird.



Original



3x3 Filterkernel



5x5 Filterkernel

10x10 Filterkernel

Abbildung 2.1: *Anwendung eines simplen Glättungsfilters, jeder Punkt ist der Mittelwert seiner direkten Umgebung. Oben links ist das scharfe Ausgangsbild zu sehen. Es wird progressiv ein zunehmend großer Filterkernel gewählt. Mit zunehmend großer Umgebung wirkt das Ergebnisbild zunehmend unschärfer.*[2]

Gaußfilter: Es ist vorstellbar, dass bei einer Glättung, bei der zwar störendes Rauschen entfernt wird, das Ursprungsbild aber so gut wie möglich erhalten bleiben soll, die Pixel in der direkten Umgebung einen höheren Einfluss haben sollten, als ein weit entfernter Pixel. Zu diesem Zweck wird die nähere Umgebung und das Ursprungspixel selber stärker gewichtet als die ferne Umgebung.

Es wird daher für jedes Umgebungspixel ein Gewicht w_{ji} eingeführt, um ein gewichtetes Mittel zu erhalten:

$$I^*(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{n} I\,(x+i-a+1, y+j-a+1) k(i,j) w_{ij} \tag{2.4}$$

Die Gewichtung kann direkt in die Faltungsmatrix eingebracht werden. Das wichtigste Beispiel hierfür ist eine gaußsche Gewichtung gemäß der Gaußschen Normalverteilung:

[2] Bilder programmgestützt erzeugt, Verwendung Mittelwertfilter

$$\frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix} \tag{2.5}$$

Die Visualisierung eines Kernels wie dem hier dargestellten kann als Gauß'sche Glockenkurve identifiziert werden (vgl. Abbildung 2.2). Allgemein lässt sich ein diskreter Gaußfilter beliebiger Größe und mit beliebiger Varianz über die folgende Formel berechnen:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.6}$$

Der Wert der Varianz σ beeinflusst direkt die Stärke der Weichzeichnung des Bildes. Zusammenfassend kann gesagt werden, dass der Gaußfilter einer der nützlichsten und am weitesten verbreiteten Weichzeichnungsfilter ist. Er fungiert als Tiefpassfilter, der ungewünschtes Rauschen entfernt. Details in kleineren Strukturen gehen je nach Konfiguration verloren, gröbere Strukturen bleiben erhalten. Insgesamt erzielt der Filter bessere Ergebnisse als z.B. vergleichbare Mittelwertfilter.

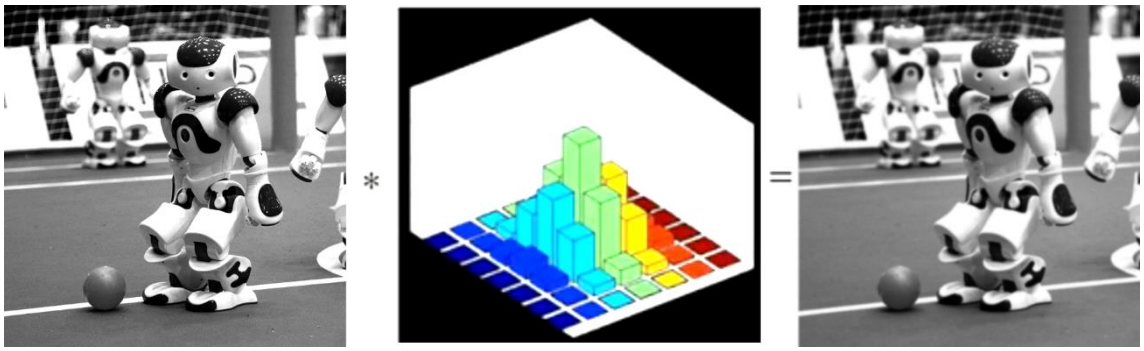


Abbildung 2.2: *Anwendung eines Gaußfilters. Links: Das Ausgangsbild auf das der Filter angewandt wird. Mitte: Visualisierung des Filterkernels, der als Gauß'sche Glocke wiederzuerkennen ist*[3]*. Rechts: Das weichgezeichnete Ergebnisbild enthält noch die allermeisten Details des Ausgangsbildes.*

Medianfilter: Beim Medianfilter werden die Werte der ausgewählten Umgebung ihrer Größe nach sortiert und der in der Mitte liegende Wert wird verwendet. Die anderen Werte werden verworfen. Der Vorteil des Medianfilters ist, dass extreme Ausreißer in jedem Fall verworfen werden und nicht wie etwa beim Gaußfilter ungünstig betont werden, wenn sich diese im Zentrum der ausgewählten Umgebung befinden. Der Medianfilter lässt sich aber nicht einfach durch eine Faltungsmatrix implementieren. Das Sortieren und das Verwerfen der Einträge muss algorithmisch erfolgen und ist daher weniger weit verbreitet.

[3] Visualisierung des Filterkernels aus der Vorlesungsreihe Computer Vision von Dr. Alper Yilmaz, Dr. Mubarak Shah, UCF 2014, http://crcv.ucf.edu/courses/CAP5415/Fall2014/Lecture-2-Filtering.pdf, Seite 138, abgerufen am 22.01.2017

2.1.2 Gradientenfilter

Gradientenfilter werden eingesetzt, wenn das Ziel der Vorverarbeitung die Hervorhebung von Kanten und Linien im Bild ist. Gradientenfilter basieren auf der ersten Ableitung der Helligkeitswerte eines Bildes. Funktionsähnliche Filter, die auf der zweiten Ableitung basieren und eher Linienenden und Punkte betonen, gehören zur Kategorie der Laplacefilter.

Gradient: Der Gradient einer Funktion mit zwei Variablen x, y ist ein Vektor der partiellen Ableitungen in x- und y-Richtung. Da ein Bild eine Darstellung einer solchen diskreten Funktion ist, lässt sich auch über den Gradienten in Bildern sprechen.

Der Gradient eines Bildes hat also die zwei Komponenten:

- f_x die Ableitung in x-Richtung
- f_y die Ableitung in y-Richtung

Damit lässt sich die Magnitude des Gradienten beschreiben als:

$$m(x,y) = |\nabla \mathrm{f(x,y)}| = \sqrt{f_x^2 + f_y^2} \tag{2.7}$$

Die Richtung des Gradienten ist somit beschrieben durch:

$$\theta = \tan^{-1}\frac{f_y}{f_x} \tag{2.8}$$

Intuitiv lässt sich der Gradient für die Ableitung in x-Richtung als der horizontale Farbverlauf bzw. –differenz, der für die Ableitung in y-Richtung als der vertikale Farbverlauf, auffassen.

In ihrer einfachsten Form dienen Filter der Berechnung des oben beschriebenen Gradienten oder Abwandlungen desselbigen.

Beispiele hierfür sind die Filterkernels der Rückwärtsdifferenz: $[-1 \quad 1]$, der Vorwärtsdifferenz: $[1 \quad -1]$, sowie der Zentraldifferenz: $[-1 \quad 0 \quad 1]$.

Eine beispielhafte Berechnung der Rückwärtsdifferenz für eine gegebene diskrete Funktion f sieht damit so aus:

$f(x) =$	10	15	10	10	25	10	10	10
$f'(x) =$	0	5	-5	0	15	-15	0	0

Prewitt-Filter: Die folgenden Filterkernel sind eine 3×3 Variante der Zentraldifferenz und werden im Prewitt-Operator verwendet. Da zwischen den 3 Zeilen allerdings der Mittelwert gebildet wird, wird das Bild in jedem Pixel auch implizit geglättet:

$$f_x \to \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},\ f_y \to \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[4]

***Abbildung 2.3**: Anwendung des Prewitt-Filters auf einer Matrix. Links: Die Werte des Ausgangsbildes, grün eingerahmt ist ein betrachtetes Ausgangspixel, rot eingerahmt ist die vom Filterkernel einbezogene Umgebung. Mitte: Die berechneten x-Werte, grün eingerahmt das betrachtete Ergebnispixel. Rechts: Die berechneten y-Werte, da kein vertikaler Farbverlauf im Ausgangsbild vorzufinden ist, sind alle Werte 0.*

Mit diesen Filterkerneln lässt sich also sowohl der Verlauf in x- als auch in y-Richtung bestimmen, das finale Ergebnisbild ergibt sich dann in der Regel aus der Magnitude des Gradienten (siehe Abbildung 2.4).

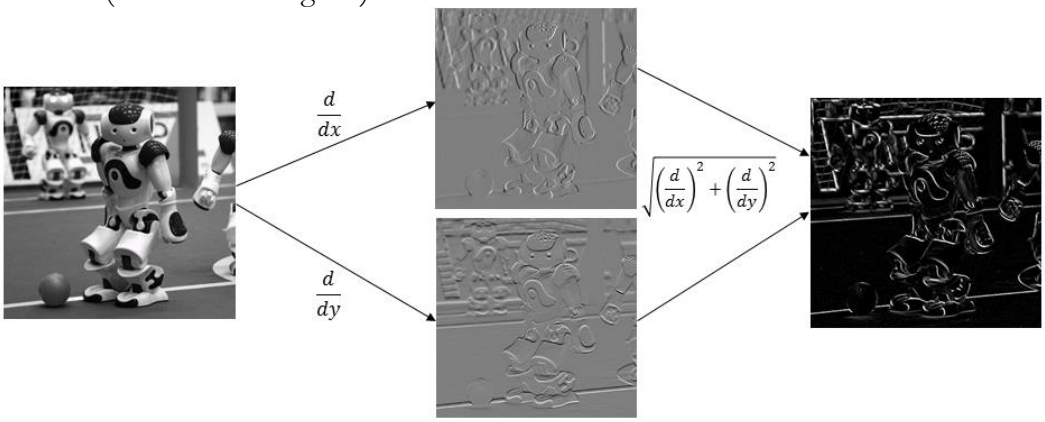


***Abbildung 2.4:** Anwendung eines etwas komplexeren Grafikfilters, der ein Gradientenbild berechnet, das implizit auch leicht geglättet ist. Links: Das Ausgangsbild auf das der Filter angewandt wird. Mitte: Die beiden Bilder, die sich als Zwischenschritt ergeben und den Helligkeitsverlauf in horizontale Richtung (oben) und in vertikale Richtung (unten) darstellen. Rechts: Das Magnitudenbild des Gradienten ist das finale Ergebnisbild. Kanten treten hier bereits deutlich hervor.*

2.1.3 Kantendetektion

Nach Abschluss der Vorverarbeitung eines Bildes ist die Segmentierung der nächste Prozessschritt im maschinellen Sehen. Durch die Segmentierung werden Strukturen und Eigenschaften von Objekten in einem Bild zu Merkmalen zusammengefasst und können als solche identifiziert, wiedererkannt und ausgewertet werden. Eines dieser Merkmale sind die Kanten. Per Definition sind Kanten erhebliche lokale Veränderungen der Intensitäten in einem Bild. Kanten treten als Grenzen zwischen flächigen Regionen eines Bildes auf. Die bereits eingeführten Gradienten sind ideal, um solche Intensitätsänderungen auszumachen.

[4] Bilder aus der Vorlesungsreihe Computer Vision von Dr. Alper Yilmaz, Dr. Mubarak Shah, UCF 2014, http://crcv.ucf.edu/courses/CAP5415/Fall2014/Lecture-2-Filtering.pdf, abgerufen am 22.01.2017

Die Qualitätsmerkmale einer guten Kantenerkennung sind: Robustheit gegenüber Rauschen, genaue Lokalisierung und hohe Sensitivität mit niedriger Ausfallrate. Die hier aufgeführten Kantendetekoren verbessern diese Qualitätsmerkmale sukzessiv und finden sich schließlich in der Merkmalsgenerierung mittels SIFT und HOG wieder.

2.1.4 Sobel-Operator

Der Sobel-Operator war, gemeinsam dem sehr ähnlichen Prewitt-Operator aus Abschnitt 2.1.2, aufgrund seiner Einfachheit und Effizienz einer der am weitesten verbreiteten Kantendetektoren. Zunächst wird ein Gradientenbild erzeugt mit den Faltungsmatrizen:

$$horizontal: \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, vertikal: \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.10}$$

Schwellenwertverfahren: Die einfachste Form der Bildsegmentierung ist das Schwellenwertverfahren. In seiner primitivsten Variante wird das Bild mit einem Schwellenwert t binarisiert, das heißt jeder Bildpunkt wird mit t verglichen. Ist das Pixel kleiner als t, wird der Wert auf schwarz gesetzt, ansonsten auf weiß. Ist $I(x, y)$ ein Bildpunkt, wird für jeden Punkt im Bild die folgende Berechnungsvorschrift angewandt:

$$I_s(x, y) = \begin{cases} 0, & I(x, y) < t \\ 1, & I(x, y) \geq t \end{cases}. \tag{2.11}$$

Mit zusätzlichen Werten t_i kann das Bild natürlich in beliebig viele Segmente unterteilt werden. Eine weitere Variante verwirft alle Bildpunkte unterhalb des Schwellwerts, behält für die Werte oberhalb von t aber die Originalwerte bei, also

$$I_{s_2}(x, y) = \begin{cases} 0, & I(x, y) < t \\ I(x, y), & I(x, y) \geq t \end{cases}. \tag{2.12}$$

Man spricht vom Sobel-Operator wenn auf das Magnitudenbild anschließend eine solche Schwellwertfunktion angewandt wird, so dass ein klares Kantenbild entsteht (siehe Abbildung 2.5).

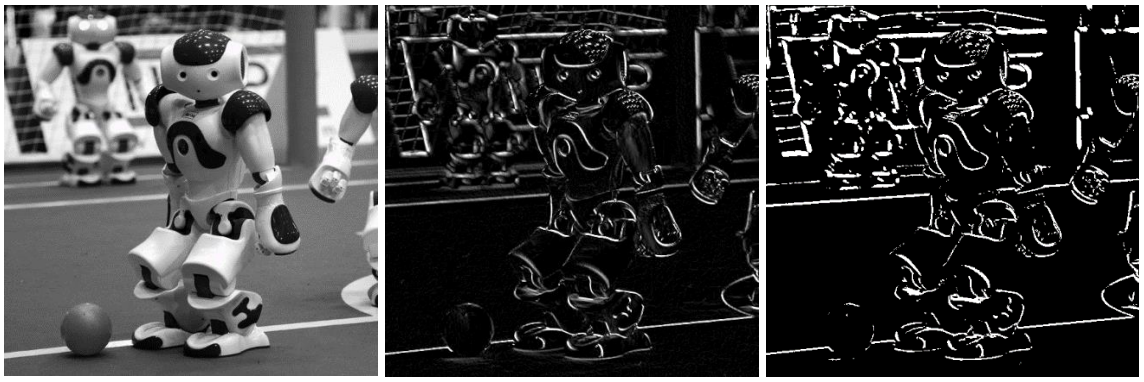


Abbildung 2.5: *Anwendung des Sobel-Operators. Links: Das Ausgangsbild auf das der Operator angewandt wird. Mitte: Darstellung des mit Sobel-Filterkernels erzeugten Gradientenbildes. Rechts: Das finale Kantenbild nach Anwendung einer Schwellwertfunktion mit Schwellwert 70.*[5]

2.1.5 Marr-Hildreth-Operator

Während die Glättung beim Prewitt- bzw. Sobel-Filter noch so gering ist, dass man sie vernachlässigen kann, entsteht der Filterkernel des Marr-Hildreth-Operators als echte Verbindung der Glättungs- und Gradientenfilter durch die Anwendung des Laplace-Operators auf die zweidimensionale Gaußfunktion (2.6):

$$g(x,y) = \Delta G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

$$g(x,y) = -\frac{1}{\pi\sigma^4}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \left(1 - \frac{x^2+y^2}{2\sigma^2}\right) \tag{2.13}$$

Der Filter wird daher Laplacian of Gaussian (LOG) genannt.

In dem gefilterten Bild entstehen durch den LOG als Maß der zweiten Ableitung Nullstellenübergänge dort wo sich Objektkanten befinden. Um das finale Kantenbild zu erhalten, wird der Betrag der Stärke der Nullstellenübergänge bestimmt und auf dieses Bild wiederrum eine Schwellwertfunktion angewandt (vgl. Abbildung 2.6). Zusammenfassend wird also wieder mittels Gauß geglättet, es werden Nullstellenübergänge und deren Intensität mittels Laplace-Operator gefunden und schließlich final Kanten anhand einer Schwellwertfunktion herausgestellt. Die diskrete Approximation des LOG als Filterkernel wird in der Regel für quadratische Matrizen ungerader Seitenlängen durchgeführt. Ein einfaches Beispiel für einen 5×5 LOG-Filterkernel mit $\sigma = 1$ ist:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.14}$$

[5] Bilder erzeugt mit Adobe Photoshop CC 2014, Verwendung eigener Filter, Korrekturen: Schwellenwert

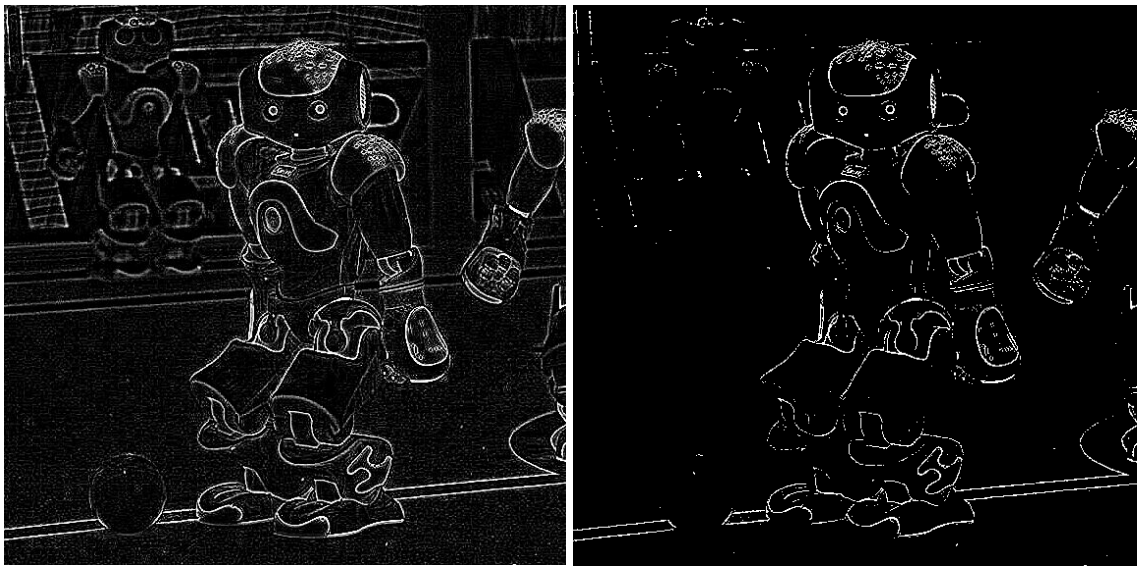


Abbildung 2.6: *Beispielhafte Anwendung des Filterkernels (2.14). Links: Gradientenbild, Rechts: Das Kantenbild nach Anwendung des Schwellenwerts 200.*[6]

Je größer die gewählte Varianz σ, desto größer muss auch der Filterkernel gewählt werden, um die breitere Glockenkurve angemessen abzubilden (siehe Abbildung 2.7).

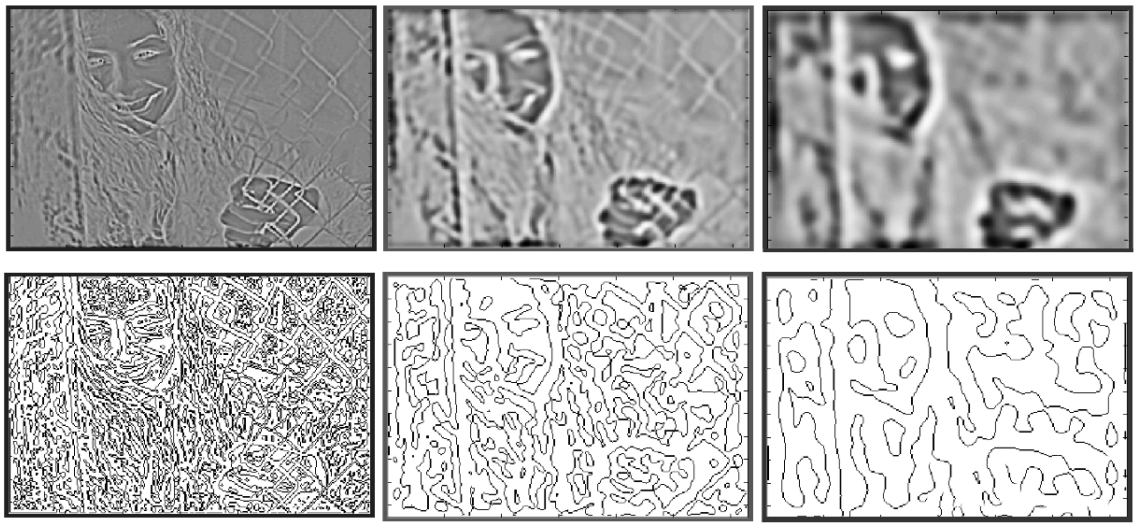


Abbildung 2.7: *Eine Reihe erzeugter Kantenbilder mit verschiedener Varianz. Von links nach rechts ist die Varianz* $\sigma_1 = 1$, $\sigma_2 = 3$, $\sigma_3 = 6$. *Die obere Reihe zeigt die Bilder direkt nach der Filteranwendung. Die höhere Varianz führt zu einer breiteren Glockenkurve und damit zu weniger steilen Übergängen. In der unteren Reihe sieht man entsprechend weniger erzeugte Kanten als Ergebnisse der Anwendung der identischen Schwellwertfunktion auf die Nullstellenübergänge.*[7]

[6] Bilder programmgestützt erzeugt, Verwendung eigener Filter, Verwendung Schwellwertfunktion

[7] Bilder aus der Vorlesungsreihe Computer Vision von Dr. Alper Yilmaz, Dr. Mubarak Shah, UCF 2014, http://crcv.ucf.edu/courses/CAP5415/Fall2014/Lecture-3-EdgeDetection.pdf, abgerufen am 22.01.2017

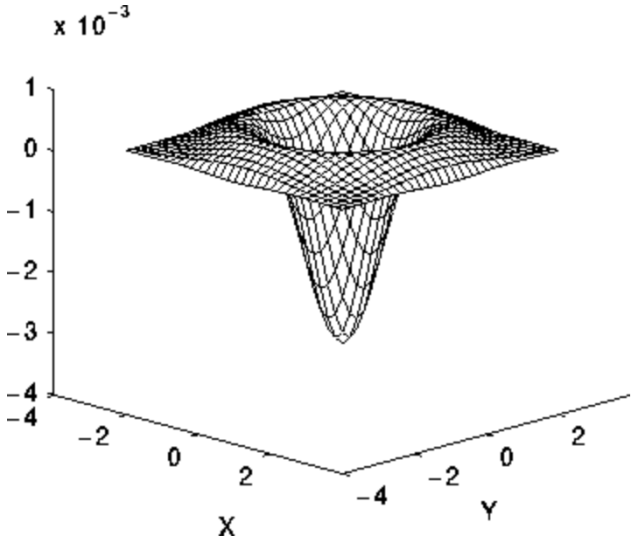


Abbildung 2.8: *2D-Plot der LOG-Funktion. Die Visualisierung erklärt, warum der LOG aufgrund seiner Form auch Sombrerofilter genannt wird.*[8]

2.1.6 Canny-Algorithmus

Basierend auf den Qualitätsmerkmalen einer guten Kantenerkennung hat John Canny ein mathematisches Optimierungsproblem formuliert, aus dem der Canny-Algorithmus hervor ging. [18] Der eigentliche Algorithmus ist zunächst identisch mit den bisher beschriebenen Verfahren. Als erstes wird das Bild mit einem Gaußschen Filterkernel geglättet und die Intensität der Kantenübergänge im Gradientenbild herausgestellt. Zusätzlich zur Gradientenmagnitude (2.7) wird bei Canny auch die Gradientenrichtung (2.8) berechnet und kann dann bei der Non-Maximum-Unterdrückung und der Merkmalsextraktion eingesetzt werden.

Non-Maximum-Unterdrückung: Die Non-Maximum-Unterdrückung nach Canny dient der Ausdünnung der erhaltenen Kanten. Im Genauen stellt die Non-Maximum-Unterdrückung sicher, dass eine Kante nie breiter als ein Pixel ist. Um dies zu erreichen, wird für jedes Ergebnispixel überprüft, ob es das lokale Maximum auf der Kante ist. Es wird dazu jedes Ergebnispixel mit dem direkt benachbarten Pixeln in positiver und negativer Gradientenrichtung verglichen. Ist es das Maximum der drei Werte, bleibt es erhalten, ansonsten wird es auf null gesetzt. Aufwändigere Ansätze filtern mit einem Sliding-Window entlang der Gradientenrichtungen die lokalen Maxima heraus oder interpolieren gar zwischen konkurrierenden Pixeln hoher Intensität. Die Non-Maximum-Unterdrückung ist jedoch bereits in ihrer einfachsten Form sehr effektiv, sie erzeugt scharfe Kanten und sorgt insbesondere auch dafür, dass keine Objektkante im gefilterten Bild mehrfach auftaucht.

[8] Darstellung aus der HIPR2 Bilderbibliothek von R. Fisher et al., University of Edinburgh, http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, abgerufen am 22.01.2017

Hysterese-Schwellwertoperation: Ein weiterer Verbesserungsansatz, der mit Canny eingeführt wurde, ist das Anwenden der Hysterese an Stelle der einfachen Schwellwertfunktion. Bei der Hysterese-Schwellwertoperation werden zwei Schranken S_1, S_2 mit $S_2 > S_1$ gewählt. Im Bild wird nach Werten größer S_2 gesucht. Ausgehend von solch einem Pixel wird die Kante in beide Richtungen abgefahren. Solange die Pixel größer als der untere Schwellwert S_1 sind, gehören diese zur Kante und werden nicht verworfen. Trifft man beim Abfahren der Kante auf einen Wert kleiner S_1 wird dieser verworfen und die Kante dort unterbrochen. Abschließend werden alle Werte verworfen, die zu keiner Kante gehören (vgl. Abbildung 2.9). Der Vorteil bei diesem Verfahren ist, dass der Algorithmus toleranter gegenüber Pixeln ist, die zwar unter dem ursprünglichen Schwellwert liegen, aber Teil einer zusammenhängenden Kante sind. Unnötiges und fehlerhaftes Aufbrechen von Kanten wird vermieden.

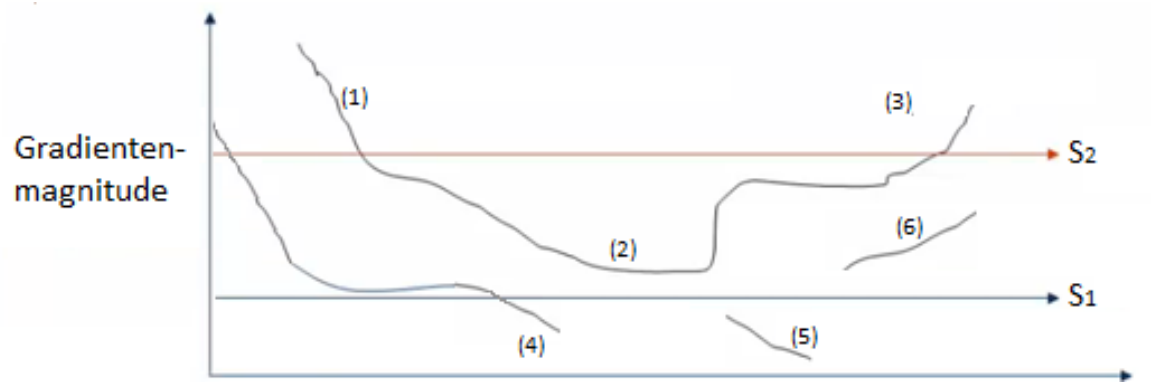


Abbildung 2.9: *Exemplarische Darstellung der Hysterese-Schwellwertoperation. Die Abschnitte (1), (2) und (3) bilden eine Kante und werden vollständig akzeptiert, da mindestens ein Wert über dem oberen Schwellwert S_2 liegt und kein Wert unter den unteren Schwellwert S_1 fällt. Die Kante (4) bleibt größtenteils erhalten, erst die sehr niedrigen Werte gegen Ende der Kante, die unterhalb von S_1 liegen, werden verworfen. Die Kanten (5) und (6) werden komplett verworfen, da diese zu keinem Zeitpunkt größer als S_2 sind. Nach der einfachen Schwellwertfunktion wäre der Abschnitt (2) und der größte Teil von (4) komplett verworfen worden. Die eigentlich zusammenhängende Kante wäre in mehrere Teilstücke zersetzt worden oder es hätte ein so niedriger Schwellwert gewählt werden müssen, dass sogar Abschnitt (6) als Kante im Bild aufgetaucht wäre.*

2.2 Merkmalsextraktion und Interest-Operatoren

Interest-Operatoren extrahieren lokale Bildmerkmale, die auch als Features bezeichnet werden, verstreut um markante Regionen, den regions of interest. Klassifikatoren stützen die Berechnung ihrer Ergebnisse auf generierte Merkmalsvektoren dieser Schlüsselpunkte. Entscheidend bei diesem Ansatz ist, dass die regions of interest einzigartig und besonders informativ in ihrer lokalen Umgebung sind und sich diese Eigenschaft dann auch in einzigartigen Merkmalsvektoren wiederspiegelt.

Die Gesamtperformance der Objektklassifizierung hängt von der Zuverlässigkeit, Genauigkeit und Wiederholbarkeit der Regionen und ihrer Features für jede der Objektklassen ab. Diese Eigenschaften sind abhängig von der Menge an relevanten Informationen die von den erfassten Merkmalen mitgeführt werden. Der große Vorteil der auf Interest-Operatoren basierenden Methoden ist die Kompaktheit der Darstellung und die damit beschleunigte Prozessierung. Die Dimension des Merkmalsvektors auf Basis wichtiger Punkte oder Bereiche ist in der Regel sehr viel kleiner, als die Gesamtzahl aller Bildpixel. Die verschiedenen Verfahren weisen jedoch häufig Einschränkungen für allgemeinere Objektklassen auf, weil sie meistens konstruiert wurden, um bestimmte Objektklassen zu verarbeiten.

Einige gängige Techniken in der Merkmalsextraktion mittels Interest-Operatoren sind Moravec [19], Plessy, Förstner [20], Laplace [21], Difference of Gaussian (DOG) [22], so wie der skaleninvariante Harris-Laplace [21] und Kombinationen der selbigen. Zu den fortschrittlichsten und beliebtesten Algorithmen unter Verwendung dieser Art von Merkmalsextraktion zählen SIFT [22, 23], sowie HOG [7] und Verfahren, die auf diesen aufbauen.

2.2.1 Moravec-Operator

Der Moravec-Operator [19] fasst die gesuchten markanten Stellen als die Punkte im Bild auf, deren direkte Umgebung eine geringe Ähnlichkeit zu ihnen selbst aufweist. Ist die Umgebung eines Punktes in jeder Richtung homogen, also z.B. wenn der Punkt zentral in einer Farbfläche liegt, soll dieser nicht als markante Stelle bzw. interest point gewählt werden. Um die Selbstähnlichkeit einer Region zu bestimmen, legt der Moravec-Operator um den zu prüfenden Punkt ein lokales Fenster und bestimmt die mittleren quadratischen Gradientensummen in den vier Hauptrichtungen innerhalb des Fensters. Die Operatorantwort ist das Minimum dieser vier Gradientensummen. Ist dieses größer als der bestimmte Schwellwert, das heißt in jeder Richtung ist die Intensitätsänderung wenigstens so stark wie im Schwellwert festgelegt, liegt eine markante Stelle und damit ein interest point vor (vgl. Abbildung 2.10).

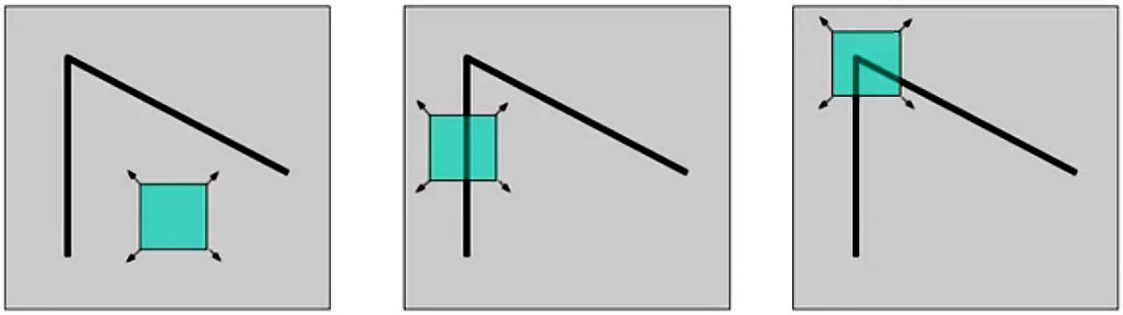


Abbildung 2.10: *Bestimmung der Selbstähnlichkeit einer Region im Moravec-Operator. Links: Das Umgebungsfenster ist in einer flachen, homogenen Region, in keiner Richtung liegt eine große Intensitätsänderung vor. Mitte: Liegt der untersuchte Punkt auf einer Kante, gibt es entlang der Kante keinen hohen Gradienten. Rechts: Das Umgebungsfenster liegt auf einer Ecke und in jeder untersuchten Gradientenrichtung wird eine signifikante Intensitätsänderung gemessen.* [9]

Der Moravec-Operator ist effizient zu implementieren, weist allerdings einige wichtige Probleme auf. Der Operator neigt dazu zu rauschen, da alle Pixel innerhalb des rechteckigen Umgebungsfensters gleich gewichtet werden und da nur das Minimum der berechneten Summen ausgewertet wird. Außerdem ist der Operator nicht rotationsinvariant und das Ergebnis wird anisotrop, da die Gradienten nur in den vier diskreten Richtungen bestimmt werden.

2.2.2 Harris Corner-Detektor

Für Harris und Stephens [24] waren die Probleme des Moravec-Operators gewichtig genug, um zwei konkrete Verbesserungen vorzunehmen. Zum einen wird vorgeschlagen die rechteckige Fensterfunktion durch eine kreisförmige Gaußfunktion zu ersetzen, zum anderen wird die diskrete Verschiebung zur Bestimmung der Selbstähnlichkeit analytisch durch die Autokorrelationsfunktion gelöst.

Zuerst werden dabei die Gradienten in x- und y-Richtung G_x, G_y mit dem Sobel-Operator bestimmt. Die symmetrische Autokorrelationsmatrix A beschreibt dann die Nachbarschaftsstruktur um den untersuchten Punkt:

$$A = \begin{bmatrix} G_x^2 & G_x G_y \\ G_x G_y & G_y^2 \end{bmatrix} \tag{2.15}$$

Aus dem Rang der Autokorrelationsmatrix lässt sich ablesen, ob eine homogene Fläche (Rang 0), eine gerade Kante (Rang 1) oder ein markanter Punkt (Rang 2) vorliegt. Alternativ kann diese Klassifikation auch aus dem Verhältnis der Eigenwerte λ_1, λ_2 von A abgeleitet werden. Sind beide Eigenwerte signifikant und in etwa gleich groß, muss eine hohe Intensitätsänderung in jeder Richtung vorliegen und die untersuchte Stelle kann als markanter Punkt betrachtet werden. Die gewünschte Merkmalsextraktion durch den Harris

[9] Bilder aus der Vorlesungsreihe Computer Vision von Dr. Alper Yilmaz, Dr. Mubarak Shah, UCF 2014, http://crcv.ucf.edu/courses/CAP5415/Fall2014/Lecture-4-Harris.pdf, Seite 26, abgerufen am 22.01.2017

Corner-Detektor lässt sich daher vereinfachen durch die Auswertung der Punktstärke V mit einem Sensibilitätsfaktor k in der Operatorfunktion:

$$V = \det(A) - k\ spur^2(A) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{2.16}$$

Eine abschließende Non-Maximum-Unterdrückung wie beim Canny-Algorithmus (2.1.6) liefert dann die gesuchten interest points.

Der ähnliche Förstner-Operator [20] verwendet ebenfalls die Autokorrelationsmatrix, wählt aber eine Bestimmung über die Eigenwerte der Kovarianzmatrix von A. Shi und Thomasi [25] schlagen dagegen vor die Operatorfunktion noch weiter zu vereinfachen zu:

$$V = \min(\lambda_1, \lambda_2) \tag{2.17}$$

In ihrer Veröffentlichung demonstrieren sie experimentell, dass diese Vereinfachung bessere Ergebnisse liefert, als die ursprüngliche Variante von Harrison und Stephens.

2.2.3 Scale-invariant feature transform

Der Harris-Corner-Detekor ist bei der Merkmalsextraktion bereits invariant gegenüber der Rotation und der Translation der markanten Regionen, er erweist sich weiterhin als robust gegenüber Helligkeitsschwankungen. Ein echter Nachteil des in 2.2.2 beschriebenen Harris-Corner-Detektors ist jedoch die nicht vorhandene Invarianz gegenüber der Skalierung. In der Praxis heißt das, bewegt sich etwa die Kamera auf ein Objekt zu oder entfernt sich von diesem oder umgekehrt das Objekt bewegt sich relativ zur Kamera, verändern sich die extrahierten Merkmale. Neue interest points werden erkannt, andere werden nicht mehr erkannt und das Wiederfinden von Objekten in einer Folge von Aufnahmen wird problematisch.

Der von Lowe entwickelte [23] und mittlerweile unter Patentschutz[10] stehende Scale-invariant feature transform (SIFT) Algorithmus begegnet dieser Problemstellung mit der Einführung sogenannter Scale-Spaces. Der Algorithmus erzeugt so für jeden interest point einen Merkmalsvektor genannt SIFT-Deskriptor. Dieser ist robust gegenüber Rotation, Translation, Skalierung und Helligkeitsschwankungen und kann als solcher für die Klassifikation oder das Wiederfinden von Objekten verwendet werden.

2.2.3.1 Scale-Space

Das Ziel der Scale-Space-Theorie ist die kontinuierlich skalierte Darstellung eines Objektes und seiner Struktur. [26] Die Ausarbeitungen von Koenderink [27] und später Lindeberg [28] belegen, dass sich die Gaußfunktion unter einigen begründeten Annahmen ideal zur

[10] Lowe, D.G., Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. 2004, Google Patents

Erstellung des Scale-Space eines Objektes eignet [22]. Bilder verschiedener Größe desselben Objekts können als ein Bild des Objekts, das skaliert und bei der Skalierung mit der Gaußfunktion geglättet wird, interpretiert werden. Mit σ ist ein kontinuierlicher Parameter vorhanden, um einen potentiell kontinuierlichen Scale-Space aufzubauen (siehe Abbildung 2.11).

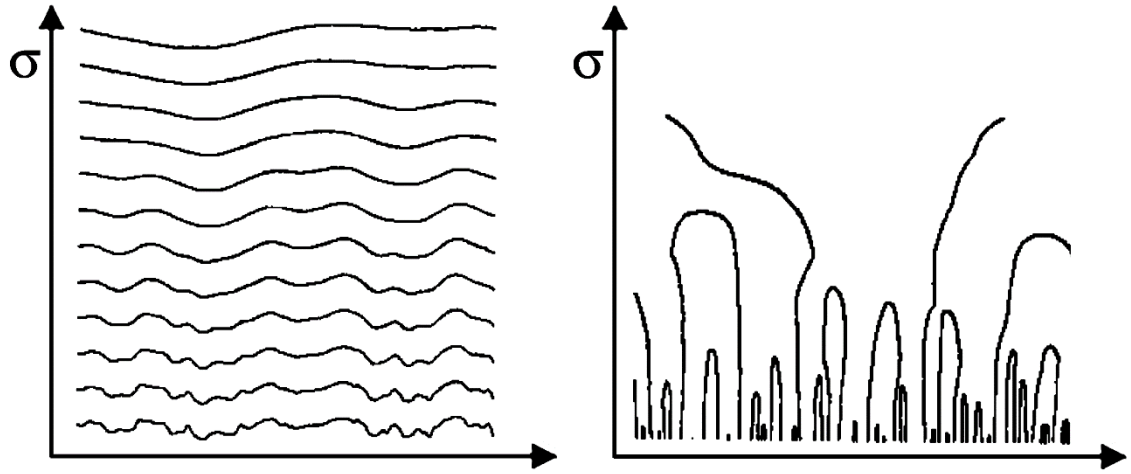


Abbildung 2.11: *Betrachtung eines kontinuierlichen Scale-Spaces. Links: Darstellung eines mit Gaußfunktion geglätteten Signals. Rechts: Die Betrachtung der Nullstellenübergänge zeigt, dass mit wachsendem σ Nullstellenübergänge zusammenfallen, umgekehrt mit fallendem σ aber stabil vorhanden bleiben. Eine gute Merkmalsbeschreibung eines Objektes sollte das Objekt möglichst eindeutig beschreiben und dabei über einen möglichst breiten Bereich des Scale-Space stabil bleiben, damit das Objekt später in verschiedensten Größen identifiziert werden kann* [11]

Der Scale-Space eines Bildes lässt sich damit als folgende Funktion definieren:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.18}$$

wobei $I(x, y)$ wie gewohnt das Bild und $G(x, y, \sigma)$ die diesmal in σ variable Gaußfunktion (2.6) ist.

In der Praxis wird eine konkrete Bildpyramide berechnet. Als Basis dient das Bild mit der höchsten Auflösung. Dieses wird sukzessiv bei konstant steigendem σ mit der Gaußfunktion geglättet, bis eine weitere Faltung keine erhebliche Veränderung der Intensität benachbarter Bildpunkte mehr bewirkt. An diesem Punkt wird das Bild in horizontaler und vertikaler Richtungen halbiert, wobei jedes zweite Pixel verworfen wird und dient wiederrum als Basis für eine neue sogenannte Oktave in der Bildpyramide. So wird nach und nach der Scale-Space als Folge von auf die Hälfte skalierter Oktaven mit jeweils zunehmend geglätteten Skalenbildern innerhalb einer Oktave aufgebaut.

Der SIFT-Algorithmus berechnet parallel zum Aufbau des gaußschen Scale-Space eine Pyramide von Bildern entstehend durch die Differenzbildung zwischen den jeweiligen

[11] Darstellung verändert nach Witkin, A.P., Scale-space filtering, in Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2. 1983, Morgan Kaufmann Publishers Inc.: Karlsruhe, West Germany. p. 1019-1022.

aufeinander folgenden Skalenbildern, die als Difference of Gaussians bezeichnet werden (vgl. Abbildung 2.12). DOG hat die nützliche Eigenschaft den bereits im Marr-Hildreth-Operator vorgestellten Laplacian of Gaussian zu approximieren, weshalb die Pyramide der Differenzbilder auch Laplace-Pyramide genannt wird und schließlich in SIFT zur Bestimmung der interest points eingesetzt wird.

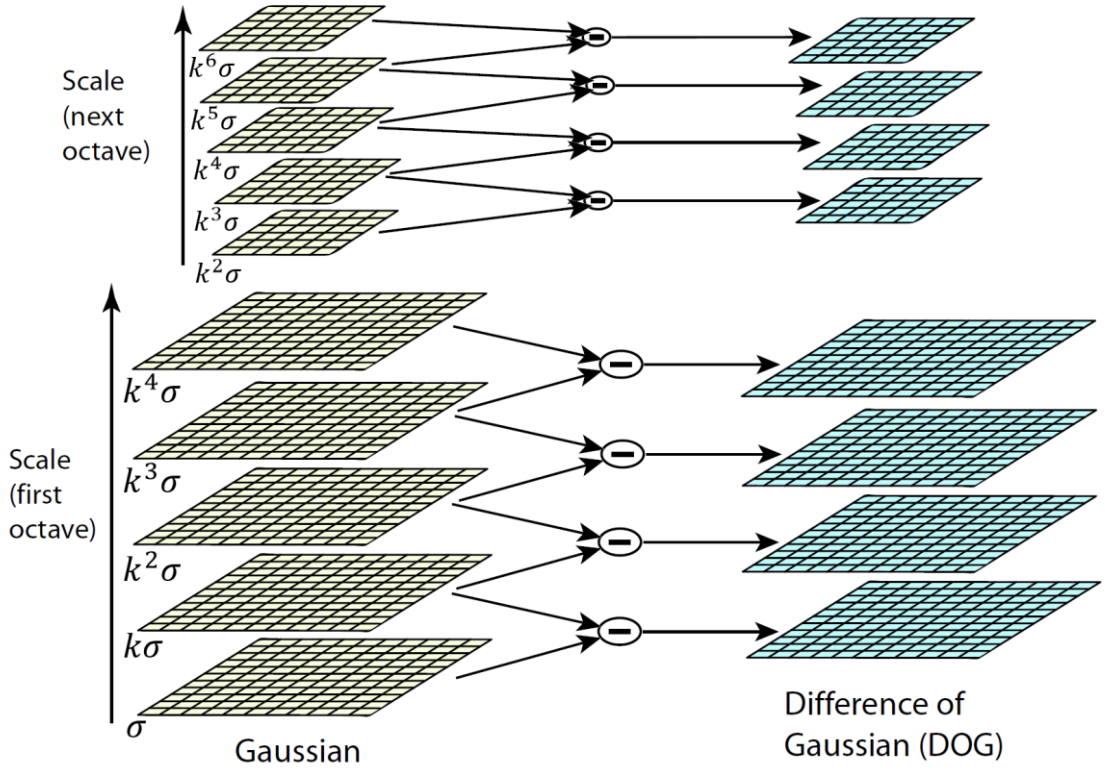


Abbildung 2.12: *Aufbau der Laplace-Pyramide beim SIFT. Auf der linken Seite sind je 5 Skalenbilder in zwei Oktaven der Bildpyramide dargestellt. Diese werden mit der Gaußfunktion zunehmend stärker geglättet, wie man am Faktor vor* σ *ablesen kann. Aus diesen Bildern wird die Laplace-Pyramide von Differenzbildern berechnet, die rechts zu sehen ist. Zwischen den Oktaven findet ein Skalensprung statt, die Basis der folgenden Oktave ist auf halbe Auflösung runterskaliert.*[12]

2.2.3.2 Histogramm

Ein weiterer elementarer Baustein von SIFT ist die Berechnung der Orientierung eines Merkmals basierend auf Histogrammen. Bei Histogrammen wird die Häufigkeitsverteilung eines Merkmals, bei SIFT ist dies das Vorkommen von Gradientenrichtungen, auf diskrete Klassen aufgeteilt und kann dann grafisch anschaulich dargestellt werden.

Die häufigste Darstellungsart ist das klassische Balkendiagramm. Bei der Verteilung von Richtungen bietet sich eine radiale Darstellungsweise an, aus der sich die dominante Richtung eines Gradienten ablesen lässt (siehe Abbildung 2.13).

[12] Darstellung verändert nach Lowe, D., Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004. 60(2): p. 91-110.

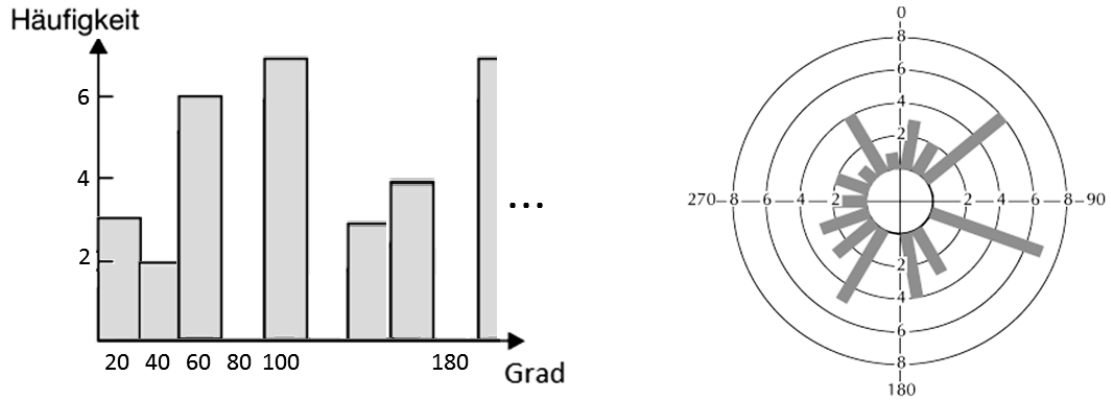


Abbildung 2.13: *Beispielhafte Darstellung eines Histogramms von Gradientenrichtungen. Links: Darstellung als einfaches Balkendiagramm, Rechts: Radiale Darstellung desselben Histogramms, in der sich die primären Richtungen ablesen lassen*

Bei SIFT und HOG kann das Merkmal der Gradientenrichtungen Werte im Bereich 0° bis 360° annehmen. Die Klassen, häufig auch aus dem Englischen Bins genannt, sind dann gleichbreite Intervalle von z.B. je 20°, in die sich eine konkrete Ausprägung des Merkmals, also ein Gradient, einsortieren lässt. Ein Histogramm bei SIFT bzw. HOG repräsentiert dann die Häufigkeitsverteilung der Gradientenrichtungen in einem bestimmten Bildabschnitt, wobei die konkreten Merkmalsausprägungen teilweise unterschiedlich gewichtet werden.

2.2.3.3 Funktionsweise

Der gesamte Ablauf des SIFT-Algorithmus von der Detektion der interest points bis hin zur Beschreibung der extrahierten Merkmale lässt sich in vier Abschnitte gliedern.

Detektion der Scale-Space Extrema: Als erstes muss ein wie in Abschnitt 2.2.3.1 beschriebener Scale-Space aufgebaut werden. Empirisch haben sich die folgenden Konfigurationen als besonders gut bewährt [22]: $Anzahl\ der\ Oktaven\ =\ 4$, $Anzahl\ der\ Bilder\ pro\ Oktave\ =\ 5, \sigma = 1.6, k = \sqrt{2}$.

Nachdem die Laplace-Pyramide berechnet wurde, können die lokalen Maxima dieser Pyramide bestimmt werden. Ein Pixel ist ein Scale-Space Extremum, wenn es kleiner respektive größer als alle seine direkten Nachbarn im Scale-Space ist. Ein Pixel wird daher nicht nur mit seinen Nachbarn im selben Bild verglichen, sondern auch mit der korrespondierenden Region im darunter und darüber liegenden Differenzbild (vgl. Abbildung 2.14). Ist ein Pixel bereits kein Extrema in der eigenen Bildnachbarschaft, wird der Vergleich mit den benachbarten Bildern aus Effizienzgründen übersprungen.

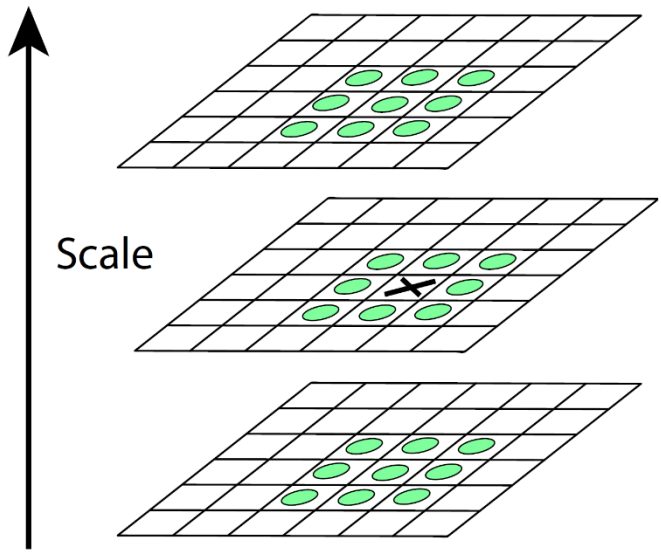


***Abbildung 2.14**: Maxima und Minima der Differenzbilder im SIFT. Die Maxima und Minima werden ermittelt, indem ein Pixel, hier mit X markiert, mit seinen 26 Nachbarn, hier als Kreise dargestellt, aus der aktuellen und den benachbarten Skalierungen verglichen wird.*[13]

Die bisher ermittelten Extrema sind die potentiellen Kandidaten für markante Stellen, im folgenden Arbeitsschritt werden diese bezüglich ihrer Stabilität und dem Kontrast zu ihrer Umgebung untersucht.

Festlegung der interest points: In neueren Iterationen des Algorithmus wird die Lokalisation der Extrema zunächst überprüft, indem die Scale-Space DOG-Funktion $D(x, y, \sigma)$ des jeweiligen Punktes mithilfe einer Taylorreihe approximiert wird. Über ein lineares Gleichungssystem lässt sich damit die Verschiebung zwischen dem über die Laplace-Pyramide ermitteltem Extremum und der tatsächlichen Lage finden. [22] Ist diese Verschiebung $\hat{x}$ in allen Komponenten größer als $0{,}5$ wird der Punkt, der der Verschiebung am nächsten liegt, als neuer Kandidat ausgewählt und muss erneut approximiert werden. Ist die Verschiebung geringer, wird diese zum bisherigen Extremum addiert. Das Ergebnis sind interpolierter Kandidatenpunkte, die schließlich zu stabileren und leichter wiederfindbareren interest points führen. [29]

Die Taylorreihe wird nochmals bemüht, um $D(\hat{x})$ zu berechnen. Ist der Wert sehr gering, wird das Extremum aufgrund eines zu geringen Kontrastes und damit hoher Anfälligkeit gegenüber Rauschen verworfen. Als guter Schwellenwert hat sich hier $|D(\hat{x})| < 0{,}03$ bewährt.

Der durch DOG approximierte Laplace-Operator liefert wieder Extrema im Bereich der Kanten und Ecken eines Bildes (vgl. Marr-Hildreth-Operator im Abschnitt 2.1.5). Kantenpunkte sind anfällig dafür sich entlang der Kante zu verschieben, wenn sich die

[13] Darstellung aus Lowe, D., Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004. 60(2): p. 91-110.

Aufnahmebedingungen wie die Helligkeit, Rauschen oder Kameraposition teilweise geringfügig ändern. Als SIFT interest points sind diese Punkte daher nur bedingt geeignet. Wie bereits beim Moravec-Operator und Harris Corner-Detektor beschrieben, lassen sich Kanten gut anhand der Intensitätsänderungen der Gradienten in den unterschiedlichen Hauptrichtungen charakterisieren (vgl. Abbildung 2.10, Harris Corner-Detektor Abschnitt 2.2.2).

Ähnlich dem Harris Corner-Detektor werden auch bei SIFT diese Gradienten bestimmt. Die auszuwertende Autokorrelationsmatrix H der Nachbarschaftsstruktur im Scale-Space ist hier definiert als die Hessematrix. Die partiellen Ableitungen werden geschätzt durch die Differenz der benachbarten Punkte:

$$H = \begin{bmatrix} D_x^2 & D_x D_y \\ D_x D_y & D_y^2 \end{bmatrix} \tag{2.19}$$

Genau wie beim Harris-Corner Detektor wird nun das Verhältnis der Eigenwerte von H betrachtet und Kantenpunkte, die eine große Differenz in ihren Eigenwerten aufweisen, werden herausgefiltert.

Bestimmung der Orientierung: Der nächste Schritt im Algorithmus weist den ermittelten, markanten Punkten eine Orientierung zu. Der Merkmalsvektor eines Punktes wird anschließend in Relation zur Orientierung erzeugt und damit invariant gegenüber der Bildrotation.

Die ermittelten Punkte wurden in $D(x, y, \sigma)$, dem Scale-Space des DOG, berechnet. Da die Orientierung basierend auf den lokalen Gradienten des ursprünglichen Bildes geschehen soll, wird der entsprechende Punkt in der Pyramide der nur mit Gauß geglätteten Skalenbilder, also in $L(x, y, \sigma)$, ausgewählt. Die Gradientenmagnitude $m(x, y)$ und seine Richtung $\theta(x, y)$ werden diskret über Pixeldifferenzen approximiert. Mit

$$\begin{aligned} \Delta x &= L(x+1, y, \sigma) - L(x-1, y, \sigma) \\ \Delta y &= L(x, y+1, \sigma) - L(x, y-1, \sigma) \end{aligned} \tag{2.20}$$

werden $m(x, y)$ und $\theta(x, y)$ berechnet durch:

$$\begin{aligned} m(x, y) &= \sqrt{\Delta x^2 + \Delta y^2} \\ \theta(x, y) &= \tan^{-1} \frac{\Delta y}{\Delta x} \end{aligned} \tag{2.21}$$

In einer von der Skalierung abhängigen Umgebung um den interest point, werden nun Gradientenmagnitude und Gradientenrichtung berechnet. In einem Histogramm (vgl. Abschnitt 2.2.3.2) mit 36 Klassen, die je 10° der möglichen Winkel repräsentieren, werden die Magnituden kumuliert, wobei weiter entfernte Punkte der Umgebung mittels Gauß geringer gewichtet werden. Die Klasse mit dem höchsten Wert ist die Hauptrichtung des interest points und wird diesem zugewiesen. Liegen im Histogramm weitere Gruppen vor,

also weitere Winkel deren Größe mindestens 80% des Maximums betragen, dann werden mit diesen Winkeln neue interest points am selben Ort erzeugt. So können aus einem interest point mehrere Merkmale hevorgehen.

Merkmalsvektor erstellen: Die Merkmale besitzen jetzt neben einer Position im Scale-Space auch eine Orientierung. Es werden erneut die Gradienten in der Umgebung um den markanten Punkt betrachtet, diesmal werden allerdings die Gradientenrichtungen und damit das Koordinatensystem des späteren Deskriptors relativ zur ermittelten Orientierung des Merkmals rotiert, so dass ein Merkmal unabhängig von der aktuellen Bildrotation denselben Merkmalsvektor hervorbringen wird. Die Gradienten werden wieder in Orientierungshistogramme sortiert, wobei dieses Mal die gesamte Umgebung in mehrere gleichgroße Unterregionen aufgeteilt wird, die jeweils in einem Histogramm zusammengefasst werden.

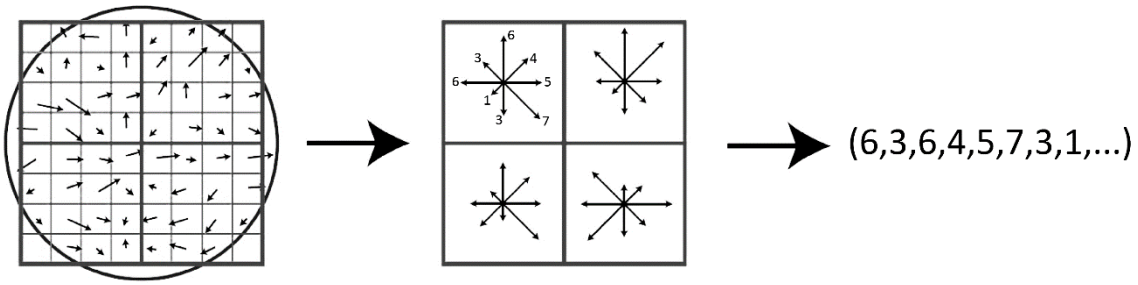


Abbildung 2.15: *Erstellung des SIFT-Deskriptors. Links: Beispielhafte Darstellung der Bildgradienten in einer* $8x8$ *Umgebung um das Merkmal herum. Die Umgebung ist hier in* $2x2$ *Unterregionen von je 16 Pixeln aufgeteilt. Mitte: Jede Unterregion bringt ein Orientierungshistogramm hervor. Rechts: Die* $2x2x8$ *Werte der Histogrammklassen werden der Reihe nach in den Merkmalsvektor geschrieben, der damit den SIFT-Deskriptor des Merkmals darstellt.*[14]

In der Regel wird eine $16x16$ Umgebung um den Vektor betrachtet und diese in $4x4$ Unterregionen und damit Histogramme aufgeteilt. Da jedes Histogramm die Gradienten in je 8 Klassen à 45° aufteilt, entsteht als Ergebnis ein 128-dimensionaler Merkmalsvektor (vgl. Abbildung 2.15). Bei Lowe [22] lieferten Experimente mit einem kleineren Merkmalsvektor signifikant schlechtere Ergebnisse, während ein größerer Merkmalsvektor keine signifikante Verbesserung mit sich brachte.

Um den Einfluss der Beleuchtung auf den Deskriptor zu reduzieren, wird der Merkmalsvektor abschließend auf eins normiert. Um weiterhin den Einfluss nichtaffiner Beleuchtungsänderungen zu reduzieren, werden die Werte im Merkmalsvektor auf $0{,}2$ begrenzt und der Vektor wird abermals normiert. Dies hat zur Folge, dass die Magnitude eines Gradienten an Wichtigkeit verliert und die Verteilung der Orientierungen einen höheren Stellenwert bekommt.

[14] Darstellung verändert nach Lowe, D., Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004. 60(2): p. 91-110

2.3 Histogramme orientierter Gradienten

Im Rahmen dieser Arbeit nehmen die auf HOG basierenden Verfahren einen besonderen Stellenwert bei der Roboterdetektion ein, weshalb diese Form der Merkmalsextratkion im folgenden Abschnitt im Detail vorgestellt wird.
Speziell zur visuellen Detektion von Menschen haben Dalal und Triggs 2005 einen Ansatz vorgestellt, der einen Merkmalsvektor basierend auf HOG hervorbringt. [7] Bereits dieser erste Ansatz lieferte außerordentliche Erfolge bei der Erkennung von Fußgängern. In den folgenden Jahren entwickelten Zhu et al. basierend auf den Konzepten von Viola und Jones [30] eine Beschleunigung des Verfahrens über die Kaskadierung der Klassifizierer. [8] Moderne Varianten, die sich noch immer auf gradientenbasierte HOG-Merkmale stützen, aber z.B. die partielle Erkennung von Objekten ermöglichen [5], stellen in diesem Bereich bis heute den aktuellen Stand der Technik dar. Lediglich die Zusammenführung mit weiteren Merkmalstypen und Techniken bietet zumindest eine bessere Erkennungsrate. [3]
Der Einsatz von HOG speziell zur Erkennung anthropomorpher Objekte ist heute daher weit etabliert und gilt als bewährt, ist allerdings nicht auf diese Objekte beschränkt.
Aufgrund der anthropomorphen Gestalt des Nao Roboters ist anzunehmen, dass sich HOG auch sehr gut zur Erkennung von Naos bzw. allgemein humanoider Roboter eignet. Weiterhin ermöglicht ein HOG Deskriptor die Objekterkennung über die Form und ist ein guter Kandidat zur Detektion anderer im Rahmen des RoboCups relevanter Objekte, wie z.B. Bällen oder Teamlogos [31]. Mit wachsender Komplexität in der Färbung der Spieler, der Tore oder der Bälle wird der formbasierten Objekterkennung eine zunehmend gewichtigere Rolle zuteil (vgl. Abschnitt 1.2).

Die grundlegende Methodik von HOG ist sehr ähnlich zu dem bereits vorgestellten SIFT-Deskriptor. In der unmodifizierten Form unterscheidet sich HOG aber hinsichtlich der globalen Anwendung auf das ganze Bild unter der Verwendung eines Sliding-Window, während SIFT einen Scale-Space basierten interest point detector implementiert hat. Im Detail unterscheiden sich beide Algorithmen insbesondere hinsichtlich der tatsächlichen Erstellung der Histogramme, so wie der Interpolation der selbigen.

2.3.1.1 Funktionsweise

Die Berechnung des HOG-Merkmalsvektors lässt sich in einer Handvoll klar unterscheidbarer Schritte beschreiben.

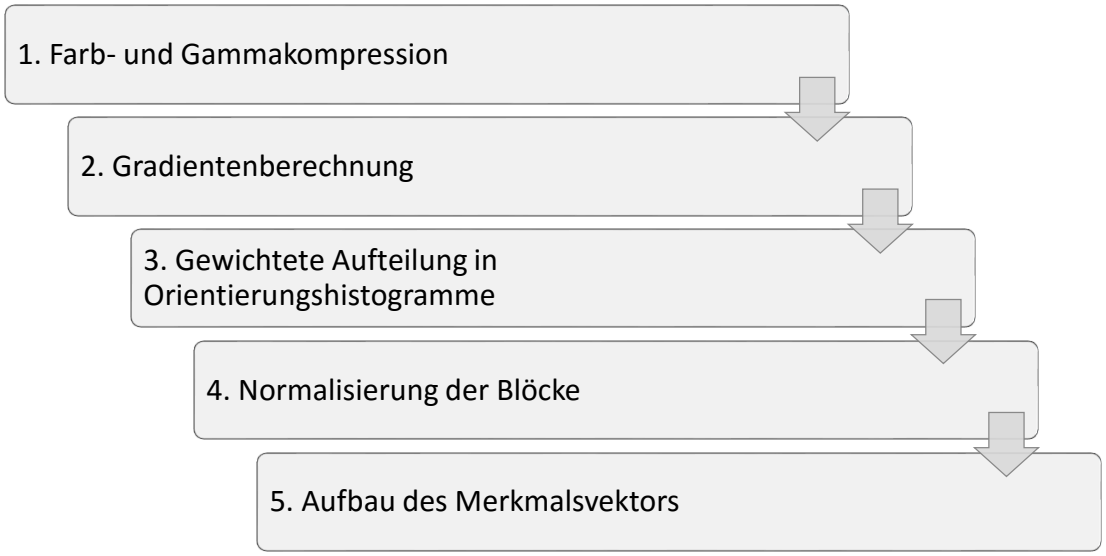


Abbildung 2.16: *Ablaufdiagramm der Prozessschritte bei der Berechnung des HOG-Merkmalsvektors*

Farb- und Gammakompression: Die Farb- und Gammakompression dienen der Normalisierung der Helligkeit des Eingabebildes. Für jeden Farbkanal wird jeder Bildpunkt nach $I_\gamma(x,y) = I(x,y)^\gamma$ (2.22) abgebildet. Bei der Gammakompression im Verfahren nach Dalal und Triggs werden die Bildpunkte mit der Quadratwurzelfunkion normiert. Die Gammakompression hat nur einen moderaten Einfluss auf das Ergebnis des Merkmalsvektors und die Qualität der Klassifikation, vermutlich weil die später folgende Blocknormalisierung eine ähnliche Aufgabe erfüllt. Auch die Verwendung von Graustufenbildern gegenüber Farbbildern beeinträchtigt die Leistung kaum (um weniger als $1{,}5\%$ bei 10^{-4} falsch positiven Detektionen pro Erkennungsfenster [7]). Häufig wird dieser Prozessschritt mittlerweile übersprungen.

Gradientenberechnung: HOG stützt seine Berechnung wieder auf die Bestimmung der horizontalen und vertikalen Ableitungen. Bemerkenswert ist, dass die Gradienten komplett ohne Glättung ermittelt werden. Verwendet wird die Zentraldifferenz in ihrer einfachsten Form, so wie im Abschnitt 2.1.2 beschrieben, also $[-1 \quad 0 \quad 1]$ und $[-1 \quad 0 \quad 1]^T$.

Experimente mit komplexeren Filtern, etwa einem Sobel-Filter oder Gaußfiltern führten in der Praxis zu schlechteren Ergebnissen bei der Geschwindigkeit, in der Erkennungsrate oder in beidem [7].

Im Anschluss werden aus diesen für jeden Bildpunkt die Gradientenmagnitude m (vgl. Formel (2.7) und die Gradientenrichtung θ (2.8) bestimmt.

Gewichtete Aufteilung in Orientierungshistogramme: Jedes Pixel trägt jetzt eine gewichtete Orientierung. Im zweiten Berechnungsschritt wird das Bild in Blöcke von Zellen

unterteilt und es folgt die Quantifizierung dieser gewichteten Orientierungen in die Orientierungshistogramme. Dieser Berechnungsschritt ist damit namensgebend für HOG.

Die Zellen können entweder rechteckig sein, dann spricht man von einem Rectangular HOG (R-HOG) oder sie sind radial, in diesem Fall wird das Verfahren Circular HOG (C-HOG) genannt (vgl. Abbildung 2.17). In den meisten Implementierungen wird der R-HOG verwendet und oft wird synonym für den R-HOG einfach von HOG gesprochen.

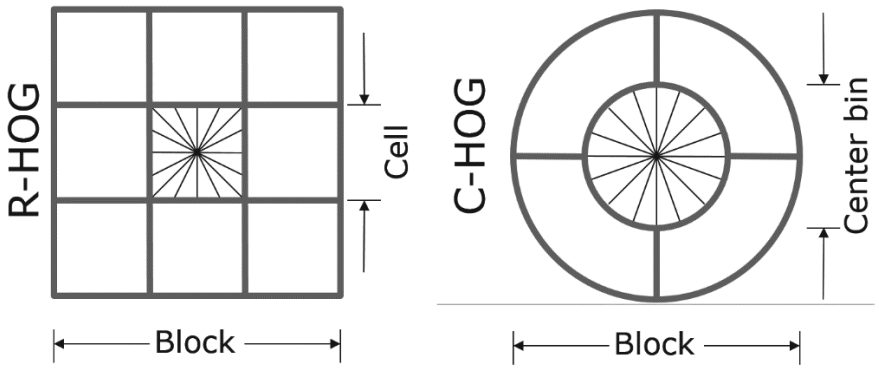
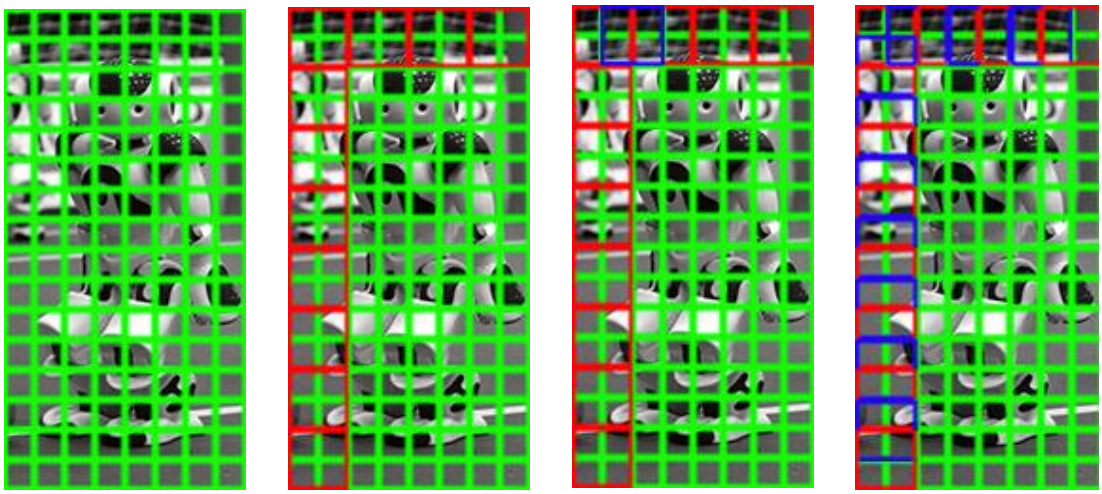


***Abbildung 2.17**: Gegenüberstellung eines R-HOG und eines C-HOG Blocks. Links: Darstellung eines R-HOG Blocks von* 3×3 *Zellen, Rechts: Darstellung eines C-HOG Blocks mit einer zentrischen Zelle und vier umgebende Sektoren* [15]

Die Zellen werden weiterhin in Blöcke zusammengefasst. Ein R-HOG Block besteht aus $m \times m$ Zellen jeweils bestehend aus $n \times n$ Pixeln. Die R-HOG Blöcke sind mit den in Abbildung 2.15 dargestellten Regionen des SIFT-Deskriptors vergleichbar, sie sind allerdings nicht nach der dominanten Richtung orientiert. Die Blöcke selber überlagern sich meistens zu 50%, so dass jede Zelle in mehrere Blöcke einfließt (vgl. Abbildung 2.18). In den Tests von Dalal und Triggs [7] lieferten Blöcke aus 2×2 Zellen von 8×8 Pixeln bei 50%iger Überlagerung die besten Resultate und sind daher ein guter Ausgangspunkt für alternative Implementierungen.



[15] Darstellung verändert nach Dalal, N., Triggs, B., Pascal VOC 2006 Workshop, ECCV 2006, p. 6

***Abbildung 2.18**: Beispielhafte Einteilung eines Bildes in Blöcke und Zellen beim HOG-Verfahren. Links: Ein* 64×128 *Pixel großes Bild, in dem die Zellen bestehend aus* 8×8 *Pixeln grün eingezeichnet sind, Mitte links: Zusätzlich sind hier rot einige Blöcke eingezeichnet, bestehend aus jeweils* 2×2 *Zellen, Mitte rechts: Blau wurde ein zusätzlicher Block eingezeichnet, der die ersten beiden roten Blöcke halb überlappt, Rechts: Alle Blöcke wurden in der ersten horizontalen und vertikalen Reihe eingezeichnet, bei* 2×2 *Zellen von* 8×8 *Pixeln und einer 50%igen Überlagerung sind das insgesamt* $7 \times 15 = 105$ *Blöcke und* $4 \times 105 = 420$ *Histogramme. Mit* 9 *Klassen pro Histogramm ergibt dies einen* 3780*-dimensionalen Merkmalsvektor.*

Ist das Bild in Blöcke unterteilt erfolgt eine trilineare Interpolation der Gradientenmagnitude und der Gradientenrichtung jedes Bildpunkts in Histogramme pro Zelle pro Block. Die Histogramme bestehen aus 9 Klassen, die sich gleichmäßig über das 360° große Intervall verteilen, häufig wird aber auch ein vorzeichenloses 180° großes Intervall angewandt. Die Werte eines Pixels werden anhand ihrer räumlichen Lage bilinear in die Zellen eines Blocks interpoliert. Dann wird zwischen den zwei nächsten Intervall-Klassen des Histogramms einer Zelle linear interpoliert, um Alias-Effekten entgegenzuwirken (siehe Abbildung 2.19).

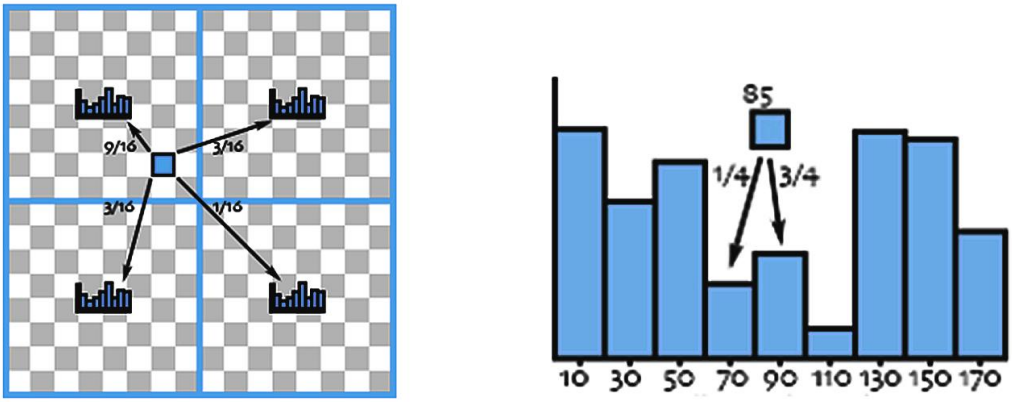


***Abbildung 2.19**: Trilineare Interpolation beim HOG Verfahren. Links: Dargestellt ist die bilineare Interpolation innerhalb eines Blocks. Der blau markierte Bildpunkt geht gewichtet in die vier Histogramme der Zellen des Blocks ein. Es wird zwischen der Lage des Pixels und seinem Abstand zur Mitte einer Zelle interpoliert. Rechts: Innerhalb des Histogramms wird linear zwischen der Gradientenrichtung und der Mitte des Intervalls der zwei nächsten Klassen interpoliert.*[16]

Durch die trilineare Interpolation variieren die Histogramme einzelner Zellen abhängig davon in welchem Block sie berechnet werden, es landen also keine redundanten Informationen im Merkmalsvektor.

Normalisierung der Blöcke: Bevor die entstandenen Histogramme der einzelnen Blöcke zum fertigen HOG-Deskriptor zusammengesetzt werden, wird im vorletzten Arbeitsschnitt jeder lokale Block normalisiert. Die Normalisierung soll den lokalen Schwankungen der Gradientenmagnituden aufgrund schwankender Helligkeits- und Kontrastwerte durch

[16] Verändert nach Yilmaz, A., Shah, M., Vorlesungsreihe Computer Vision, UCF 2012, http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-6a-Hog.pdf, Seite 5, abgerufen am 22.01.2017

örtlich variierende Beleuchtung und variierenden Fokus entgegenwirken. Es wurden die vier folgenden Methoden zur Block-Normalisierung getestet.

Sei v der nicht-normalisierte Vektor, der alle Histogramme eines Blocks enthält und $||v||_k$ sei seine k-Norm für $k = 1,2$, sei e eine kleine Normierungskonstante, dann können die folgenden Normalisierungsabbildungen gewählt werden:

$$L2-norm: \quad v \rightarrow \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{2.23}$$

$$L2-hys: \quad v \rightarrow \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{2.24}$$

Anschließendes setzen der Werte größer als $0{,}2$ auf $0{,}2$ und erneutes Normalisieren

$$L1-norm: \quad v \rightarrow \frac{v}{||v||_1 + e} \tag{2.25}$$

$$L1-wurzel: \quad v \rightarrow \sqrt{\frac{v}{||v||_1 + e}} \tag{2.26}$$

In den Tests von Dalal und Triggs verbessert sich mit der Normalisierung der Blöcke die Erkennungsrate erheblich gegenüber nicht-normalisierten Versuchen. Alle Varianten erreichten dabei annähernd gleiche Ergebnisse.

Aufbau des Merkmalsvektors: Sind die Vektoren der lokalen Blöcke normalisiert, werden diese im letzten Schritt zu einem einzigen Merkmalsvektor konkateniert. Sei $m \times m$ die Anzahl der Zellen pro Block, n_b die Anzahl der Blöcke im Bild und N die Anzahl der Klassen im Histogramm, dann hat der berechnete HOG-Deskriptor
$m \times m \times n_b \times N$ Dimensionen. Der berechnete HOG-Merkmalsvektor kann verwendet werden um einen Klassifikator zu trainieren oder er wird eben durch einen solchen Klassifikator ausgewertet. In ihrer ersten Vorstellung von HOG-Merkmalen setzen Dalal und Triggs eine lineare SVM zur Erkennung von Fußgängern ein.

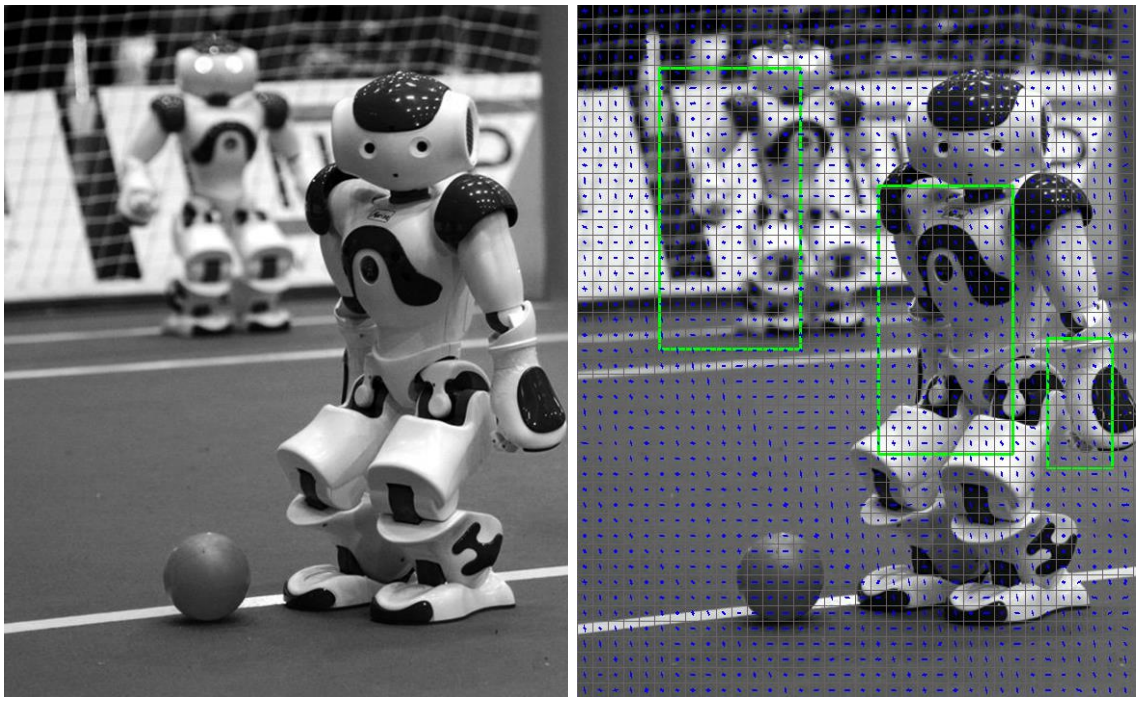


Abbildung 2.20: *Darstellung der R-HOG-Zellen auf einem Eingabebild. Links: beispielhaftes Eingabebild einer Spielsituation im RoboCup, Rechts: Eine erste Implementierung eines Detektors basierend auf HOG-Merkmalen wurde auf das Bild angewandt. Das eingezeichnete Gitter entspricht den R-HOG-Zellen. Blau eingezeichnet sind die entstandenen Histogramme. Als Klassifikator wurde eine SVM zur Erkennung von Fußgängern geladen. Die grünen Rechtecke zeigen wo dieser im Bild humanoide Formen erkannt hat.*[17]

2.4 Klassifikation

Nachdem aus einem Bild Merkmale extrahiert worden sind, können diese Merkmale auch gelernt und wiedergefunden werden. Der nächste notwendige Schritt im Prozess des maschinellen Sehens sind daher die Klassifizierungsverfahren. Klassifikatoren ordnen die Merkmalsvektoren verschiedenen Objektklassen hinzu, so dass diese als Objekte identifiziert und ausgewertet werden können.

Die grundlegendste Form von Klassifikatoren basiert auf der Verwendung einer Distanzfunktion. Zwei der am häufigsten verfolgten Ansätze des maschinellen Lernens zur Objekterkennung im Zusammenhang mit HOG-Merkmalen sind die Support Vector Machines und das AdaBoost-Verfahren.

2.4.1 Nächste-Nachbarn-Klassifikation

Bei der Nächste-Nachbarn-Klassifikation wird ein Objekt, etwa ein Merkmalsvektor $x \in \mathbb{R}^n$ über eine Distanz zu bereits klassifizierten Trainingsobjekten einer Klasse

[17] Bilder aus einer Implementierung unter Verwendung der OpenCV Version 3.1 Programmbibliothek in C++ und Visual Studio 2013

zugeordnet. Als Distanzfunktion sind verschiedene Abstandsmaße denkbar, allen voran der euklidische Abstand, aber z.B. auch die Manhattan- oder die Minkowski-Distanz. In der einfachsten Form wird der nächste Nachbar gesucht, also der Trainingsvektor mit dem geringsten Abstand gemäß der ausgewählten Distanzfunktion zum Objekt x. Das Objekt x wird dann der Klasse dieses nächsten Nachbarn zugeordnet.

Da die Variante besonders anfällig ist für Streuungen in den Trainingsdaten, wird in der Regel eine modifizierte Variante, genannt k-Nächste-Nachbarn (KNN), implementiert. Bei dieser wird nicht nur der erste Nachbar, sondern es werden die k ersten Nachbarn, hinsichtlich ihrer Klasse betrachtet (vgl. Abbildung 2.21). Die Häufigkeit der Klassen in den ausgewählten Nachbarn bestimmt dann die Klassenzuordnung von x, wobei die Häufigkeitsverteilung der Klassen innerhalb der Trainingsdaten berücksichtig werden muss.

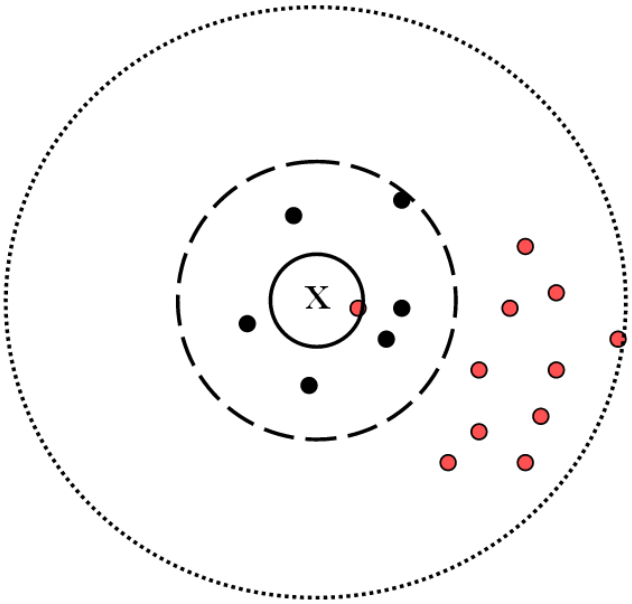


Abbildung 2.21: *Dargestellt ist der zu klassifizierende Vektor X mit schwarzen und roten Trainingsbeispielen in einem zweidimensionalen Raum.*[18] *Der erste Ring zeigt die Klassifizierung mit* $k = 1$ *in die Klasse der roten Objekte, da nur der nächste Nachbar betrachtet wird. Es folgt der gestrichelte Ring für* $k = 7$ *mit der korrekten Klassifizierung zu den schwarzen Objekten. Wird* $k = 17$ *zu groß gewählt, wie der äußere, gepunktete Ring darstellt, würde X wieder als rotes Objekt klassifiziert werden.*

2.4.2 AdaBoost

Das AdaBoost-Verfahren (Adaptive Beschleunigung) wurde 1995 von Schapire und Freund als universelle Methode zur Erstellung eines Klassifikators durch die adaptive Zusammenführung mehrerer schwacher Klassifikatoren vorgestellt [32]. Durch diese

[18] Visualisierung der Nächster-Nachbarn-Klassifikation aus der Vorlesungsreihe Knowledge Discovery in Databases I von Dr. Peer Kröger, Dr. Matthias Schubert, Ludwig-Maximilians-Universität München WS 07/08, http://www.dbs.ifi.lmu.de/Lehre/KDD/WS0708/skript/kdd-3-klassifikation.pdf, Seite 89, abgerufen am 22.01.2017

Verschmelzung wird die Gesamtperformance der Objekterkennung verbessert. Es entsteht ein binärer Klassifikator, so dass das Verfahren dann eingesetzt werden kann, wenn die automatische Klassifikation in zwei Klassen ein zufriedenstellendes Ergebnis darstellt. Im Rahmen von z.B. HOG-basierten Verfahren, kann damit bewertet werden, ob ein humanoider Roboter im Bild vorhanden ist oder nicht.

Eine formale Definition des AdaBoost-Klassifikators K lässt sich geben durch:

$$K(x) \coloneqq sgn\left(\sum_{i=1}^{n} w_i k_i(x)\right) \tag{2.27}$$

wobei $k_1, k_2, \dots, k_n$ die schwachen binären Klassifikatoren darstellen und $w_1, w_2, \dots, w_n$ die zu lernenden Gewichte sind.

Die schwachen Klassifikatoren eignen sich dann fürs Boosting, wenn sie einfach aufgebaut sind und schnell, z.B. an Hand eines einzigen Merkmals des Objekts, eine Entscheidung treffen. Außerdem muss diese Entscheidung zumindest besser sein als einfaches Raten, das heißt, die Fehlerrate muss kleiner als 50% sein. Das Verfahren kombiniert diese schwachen Klassifikatoren linear und führt dabei für jeden eine Gewichtung w_i ein. In einem schrittweisen Optimierungsverfahren werden diese Gewichte mithilfe von Trainingsdaten angepasst, so dass dabei der starke Klassifikator optimiert wird. Bei jedem Schritt werden insbesondere die schwachen Klassifikatoren stärker gewichtet, die die bisher falsch klassifizierten Merkmalsvektoren korrekt zuordnen. Anzumerken ist, dass in dem Algorithmus die schwachen Klassifikatoren sukzessive dem starken Klassifikator hinzugefügt werden. Der Algorithmus kann daher vorzeitig beendet werden, wenn er bereits zufriedenstellende Ergebnisse liefert, ohne alle schwachen Klassifikatoren verwenden zu müssen. Weiterhin kann der starke Klassifikator noch verbessert werden, indem schwache Klassifikatoren herausgeworfen werden. Wenn ein schwacher Klassifikator z.B. sehr ähnliche Resultate liefert, wie ein anderer oder wenn seine Gewichtung verschwindend klein wird, kann dieser Klassifikator verworfen werden.

Zhu et al. haben AdaBoost in ihrer Kaskade von HOG-Klassifikatoren zur Erkennung von Passanten erfolgreich implementiert, weitere Gruppen folgten diesem Beispiel [8].

2.4.3 Stützvektormaschinen

SVMs sind lineare, binäre Klassifikatoren, die im Vektorraum der Trainingsvektoren nach einer optimalen Hyperebene suchen, um diese als Entscheidungsfunktion für die Klassifikation von Merkmalsvektoren einzusetzen. Als optimale Trennebene kann man jene Hyperebene verstehen, die den größten räumlichen Abstand zu allen Klassen von Trainingsobjekten hat. Sie wird deshalb (aus dem Englischen) als Maximum-Margin-Hyperebene bezeichnet (vgl. Abbildung 2.22). Diese ergibt sich als die Lösung des Optimierungsproblems der Maximierung des kleinsten Abstands der Trainingsdaten zur Hyperebene.

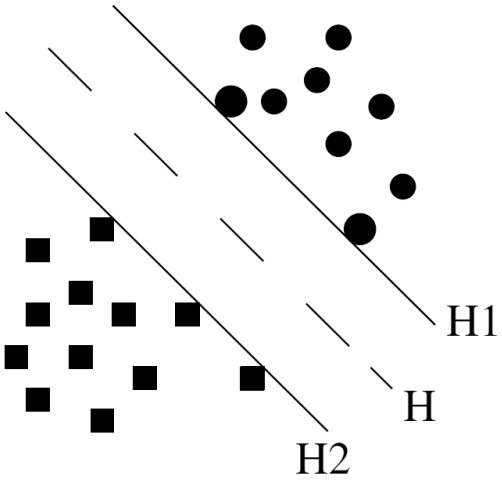


Abbildung 2.22: *Die dargestellten Trenngeraden H, H1 und H2 separieren alle korrekt die zwei verschiedenen Klassen von Trainingsobjekten. H ist die Maximum-Margin-Hyperebene, sie hat von allen möglichen Hyperebenen den maximalen Abstand zu nächstliegenden Trainingsobjekten beider Kategorien.* [19]

Voraussetzung für die Lösung des Optimierungsproblems ist, dass die Daten überhaupt linear trennbar sind, damit eine Maximum-Margin-Hyperebene existiert. Sind die Trainingsvektoren nur nicht-linear separierbar, besteht die Möglichkeit, diese in einen neuen Vektorraum mit einer höheren Dimensionalität zu projizieren (vgl. Abbildung 2.23). Die Daten werden dabei solange in eine höhere Dimension abgebildet, bis sie linear trennbar sind. Man spricht dabei auch von der Merkmalsabbildung mittels Kernelfunktion.

Eine weitere Problematik ist, dass ein einzelner Ausreißer in den Trainingsdaten die Ausprägung der Hyperebene stark beeinflussen kann. Es ist daher nützlich eine bestimmte Anzahl an Fehlern zuzulassen. Mathematisch wird dem Optimierungsproblem zu diesem Zweck ein Fehlerterm hinzugefügt. Dieser besteht aus den Schlupfvariablen ξ_i, die m-viele Verletzungen des Optimierungsproblems als Fehler abhängig vom Abstand zur Hyperebene quantifizieren und dem Parameter C, der die Summe dieser Fehler gewichtet. Beim Training einer SVM wird C häufig so gewählt, dass ein Overfitting (aus dem Englischen, Überanpassung) an die Trainingsdaten vermieden wird. Man spricht bei einem niedrigen C-Wert auch von einem weichen Rand und bei einem hohen Wert von einem harten Rand.

[19] Grafik verändert nach Fassbender, T., Videobasierte Fußgängererkennung für autonome Fahrzeuge, Freie Universität Berlin 2012, Seite 22

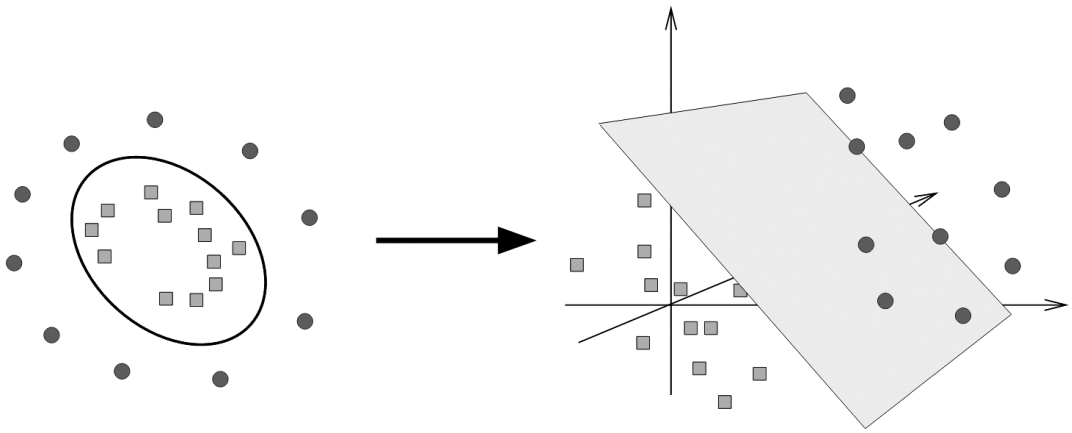


Abbildung 2.23: *Merkmalsabbildung mittels Kernelfunktion. Links: Die Trainingsdaten zweier Klassen, hier als Punkte bzw. Quadrate dargestellt, in ihrem ursprünglichen, zweidimensionalen Vektorraum. In diesem Vektorraum lassen sich die Daten nicht linear separieren. Eine mögliche Entscheidungsfunktion, hier elliptisch eingezeichnet, lässt sich nur aufwendig finden und optimieren. Rechts: Die gleichen Merkmalsvektoren der Trainingsobjekte wurden geschickt in einen dreidimensionalen Raum projiziert. Durch das günstige Auseinanderziehen der Daten auf der neuen Achse, lässt sich jetzt eine trennende Ebene finden, die beispielhaft eingezeichnet wurde.* [20]

Seien x_i Trainingsbeispiele, w der Normalenvektor der trennenden Hyperebene, b ein Bias, $y_i(\langle w, x_i \rangle + b)$ eine Entscheidungsfunktion, die den Trainingsbeispielen abhängig von ihrer Lage zur Trennebene ein Vorzeichen zuordnet, dann ist die mathematische Formulierung des oben beschriebenen Minimierungsproblems gegeben durch:

$$Minimiere\ \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m}\xi_i \qquad (2.28)$$

$$unter\ der\ Nebenbedingung\ \ y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i\ f\ddot{u}r\ alle\ i, 1 \leq i \leq m$$

Eine weitere Alternative zur linearen Trennung ist die Suche nach einer passenden nichtlinearen Trennfunktion. Zum Beispiel bietet sich für das in Abbildung 2.23 dargestellte Problem eine radiale Basisfunktion (RBF) an. Andere Probleme lassen sich dagegen besser mit einer polynomialen Kernelfunktion lösen.

Wie in Abschnitt 2.3.1.1 erwähnt, wird in der ersten Vorstellung der HOG-Merkmalsvektoren in der Arbeit von Dalal und Triggs [7] aber eine lineare SVM als Klassifikator vorgeschlagen und erfolgreich getestet. Da SVMs mittlerweile gut erforscht sind und gute Implementierungen in vielen Programmiersprachen und Bibliotheken offen zur Verfügung stehen, verwenden auch heute fast alle Anwendungen einer Objekterkennung basierend auf HOG-Merkmalen eine Form von Klassifikation durch SVMs.

[20] Grafik verändert nach Spang, R., Vorlesungsreihe Einführung in die Statistik für Bioinformatiker, Max-Planck-Institut für molekulare Genetik SS 03, Kapitel 16, Seite 24

2.4.4 Kreuzvalidierungsverfahren

Als Kreuzvalidierungsverfahren wird eine statistische Methode zur Modellvalidierung bezeichnet. Mit diesem Testverfahren kann eine zuverlässige Aussage darüber getroffen werden, wie gut ein Modell funktionieren wird, das auf einer limitierten Anzahl von Trainingsdaten angepasst wurde. Der Vorteil der k-fachen Kreuzvalidierung gegenüber einer einfachen Einteilung des bekannten Datensatzes in Trainings- und Testdaten ist, dass durch geschicktes Subsampling der Trainingsdaten selbst bei vergleichsweise kleinen Datenmengen eine verlässliche Aussage über die Qualität des Modells getroffen werden kann.

Grundsätzlich kann dadurch beim Training eines parametrisierten Modells, wie z.B. einer SVM, Überanpassung vermieden und die Aussagequalität des Klassifikators generalisiert werden.

k-fache Kreuzvalidierung: Die k-fache Kreuzvalidierung zeichnet sich dadurch aus, dass die gegebene Datenmenge T in k viele möglichst gleich mächtige Teilmengen $T_1, \ldots, T_k$ aufgeteilt wird. Im Anschluss folgen k viele komplette Testdurchläufe. In diesen wird die jeweils aktuelle Menge T_i mit dem Laufindex i herausgenommen. Mit den verbliebenden Elementen, also der Menge $\{T_1, \ldots, T_k\} \setminus \{T_i\}$ wird das Modell trainiert und gegen T_i wird es getestet. Für jeden der k Testdurchläufe ergibt sich also eine Fehlerquote bzw. es lassen sich die verschiedenen Gütemaße bestimmen. Der Durchschnitt ist dann die Gesamtfehlerquote oder Kreuzvalidierungsfehler der Methode und kann als Schätzung verstanden werden, wie gut die Methode auf unbekannten, realen Daten funktionieren wird.

Werden zusätzliche Maßnahmen ergriffen, damit die Verteilung der Elemente auf die Teilmengen $T_1, \ldots, T_k$ annähernd gleich ist, spricht man von einer stratifizierten Kreuzvalidierung.

Eine extreme Variante der k-fachen Kreuzvalidierung ist die Leave-one-Out-Kreuzvalidierung, bei der $k = |T|$ gewählt wird, so dass jedes einzelne Element einmal isoliert und getestet wird. Dabei kann natürlich keine Stratifizierung stattfinden. Die Schätzung kann schlecht sein und der Rechenaufwand ist in der Regel höher als für kleinere k, dafür bleibt die Menge der Trainingsdaten quasi unverändert groß, da auf keine Testmenge verzichtet werden muss.

Vorzugsweise wird k so gewählt, dass die Varianz und der Bias der Schätzung nicht zu groß werden. Ein kleines k resultiert meistens in weniger Varianz und einem größeren Bias, ein großes k dagegen in höherer Varianz mit geringerem Bias. [33] Als gute Werte für k gelten daher Werte zwischen 5 und 10 [33, 34], wobei ein Wert gewählt werden sollte, für den sich eine gut balancierte Stratifizierung durchführen lässt. Die Varianz lässt sich zusätzlich bei gleichbleibendem Bias reduzieren, indem die komplette k-fache Kreuzvalidierung mehrfach mit gleichem k aber zufällig permutierten Teilmengen durchgeführt wird. [33]

Hyperparameteroptimierung: Wenn das k-fache Kreuzvalidierungsverfahren zur Auswahl konkurrierender Vorhersagemodelle eingesetzt wird, dann ist davon auszugehen, dass der geschätzte Kreuzvalidierungsfehler des Modells optimistisch ist gegenüber dem realen Anwendungsfall. Beispiele dafür wären die Auswahl zwischen SVM gegenüber KNN oder die Optimierung von Hyperparametern wie etwa den Variablen der Kernelfunktion einer SVM. Durch die Auswahl des besten Modells für die gegebene Datenmenge T würde implizit ein Bias eingeführt werden. Wenn eine weniger optimistische Einschätzung der realen Qualität des Modells notwendig ist, die Modellauswahl aber mit dem Kreuzvalidierungsverfahren stattfinden soll, dann ist es erforderlich, vor dem Kreuzvalidierungsverfahren eine weitere Menge an Testdaten T_{k+1} zur Validierung des Modells zu separieren. Diese Validierungsmenge wird dann nie in der Modellauswahl betrachtet. Für die k-Iterationsschritte wird $\{T_1, \ldots, T_k\} \setminus \{T_i, T_{k+1}\}$ verwendet, für die Ermittlung der Fehlerquote wird weiterhin gegen T_i getestet. Abschließend kann nun das ausgewählte Modell auf der Validierungsmenge T_v überprüft und die entsprechenden Qualitätsmaße als Vorhersage für eine Anwendung des Modells in realen Daten interpretiert werden.

2.4.5 Kaskade von Klassifikatoren

Die Kaskadierung stammt ursprünglich aus der Elektrotechnik und bezeichnet die Hintereinanderschaltung von mehreren Modulen, um eine Verbesserung der Leistung gegenüber den einzelnen Modulen zu erreichen. Auch bei der Kaskade von Klassifikatoren wird dieser Ansatz verfolgt. Mehrere Klassifikatoren werden verkettet, wobei die gesammelten Informationen und Auswertungen eines Klassifikators dem folgenden Klassifikator zur Verfügung gestellt werden. Die erste popularisierte Klassifikatorenkaskade zur Erkennung von Gesichtern wurde 2001 von Viola und Jones vorgestellt. [30]

Die Annahme mehrere Klassifikatoren ausführen zu müssen, verlangsame die Identifizierung von Objekten, ist nicht korrekt. Durch geschicktes kaskadieren kann sowohl die Geschwindigkeit, als auch die Detektionsrate verbessert werden (vgl. Abbildung 2.24).

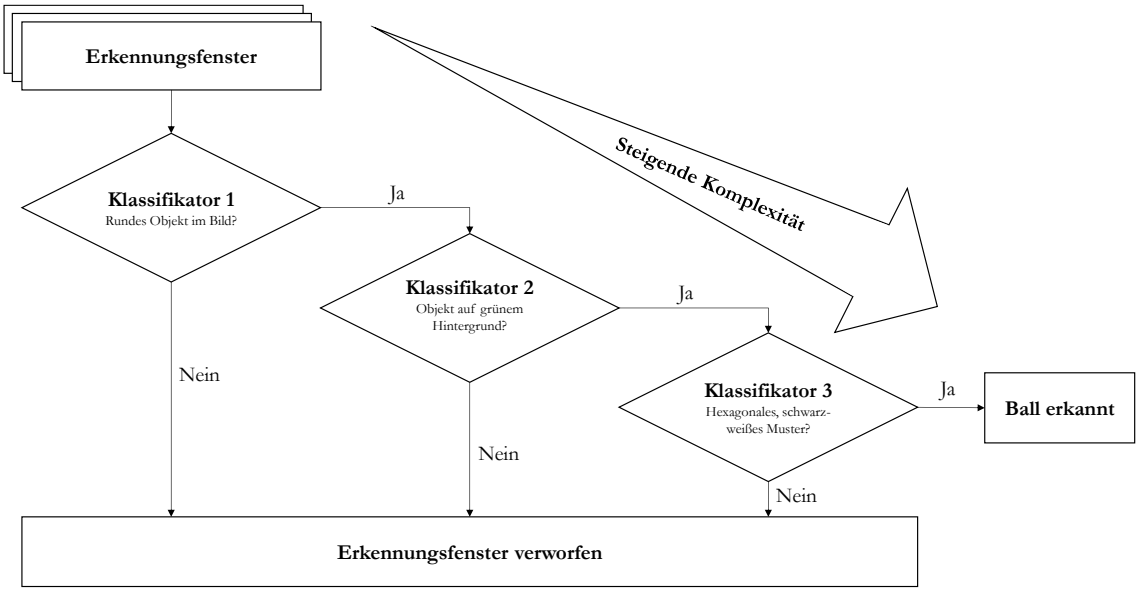


Abbildung 2.24: *Beispiel für eine Klassifikatorkaskade, die einen Fußball im Bild findet. Der Wurzelklassifikator bestimmt, ob überhaupt ein rundes Objekt im Bild ist. Erst die tieferen Klassifikatoren führen zeitintensivere Suchen durch, z.B. ob das runde Objekt ein hexagonales, schwarz-weißes Muster trägt. Negative Fälle werden schnell durch den Wurzelklassifikator verworfen und die richtig positiven Fälle werden mit hoher Genauigkeit durch die tieferen Ebenen der Kaskade bestimmt.*

Im Unterschied zum Boosting, wie etwa in Abschnitt 2.4.2 beschrieben, sind die Klassifikatoren nicht parallel geschaltet und gehen damit nicht zwingend in die Auswertung ein, sondern es existiert eine Hierarchie zwischen den Klassifikatoren. Es ist außerdem möglich die beiden Konzepte zu vereinen. Es kann z.B. eine Ebene der Kaskade ein starker Klassifikator sein, der aus der Verbindung von schwachen Klassifikatoren per AdaBoost-Verfahren hervorgegangen ist.

3 Visuelle Detektion humanoider Roboter

In diesem Kapitel wird die Umsetzung einer auf HOG basierenden, visuellen Roboterdetektion vorgestellt, die dem NaoTH zur Verfügung gestellt wird.

Untersucht wurden insgesamt drei Verfahren. Ein Verfahren basiert auf Felzenszwalbs Kaskadierung zur Abbildung eines DPMs [5, 6] genannt fHOG. Beim zweiten Verfahren wird die gelernte Filterkaskade einer Reduktion unterzogen und beim dritten Verfahren wird die gelernte SVM regularisiert. Zusätzlich kann der HOG nach Dalal und Triggs [7] durch die Reduktion der fHOG Filterkaskade angenähert werden (vgl. Abschnitt 3.1.1). Die Regularisierung der SVM geschieht durch die Einführung einer nuklearen Norm (vgl. Abschnitt 3.1.2).

Die implementierte Detektion humanoider Roboter verwendet als Klassifikatoren entsprechend trainierte SVMs. Wie ein Klassifikator Schritt für Schritt trainiert wird, ist im Abschnitt 3.2 veranschaulicht. Die verwendete Trainingsmethode löst das Optimierungsproblem der SVM durch den Max-Margin Object Detection (MMOD) Algorithmus. [35], der im Kern ein Schnittebenenverfahren zur iterativen Lösung linearer Optimierungsprobleme beschreibt (vgl. Abschnitt 3.2.4).

Nach den einleitend beschriebenen Algorithmen, wird die praktische Umsetzung ausgeführt (vgl. Abschnitt 3.3). Es wird die Auswahl der tatsächlich eingesetzten Technologien begründet und es werden realisierte Implementierungsdetails wie das Laden der Trainingsdaten, Speicherüberlegungen und das Protokollieren von Trainingsresultaten und gelernten Klassifikatoren dargelegt.

3.1 Verwendete Verfahren

Zur visuellen Detektion humanoider Roboter wurden drei Varianten untersucht, deren zugrundeliegenden Algorithmen in diesem Abschnitt beschrieben sind.

Die erste Variante ist der von Felzenszwalb et al. [5] formulierte Algorithmus zur Erstellung eines Objektdetektors unter Berücksichtigung der Modellierung eines DPM. Der Algorithmus führt, wie in Abschnitt 1.3 dargelegt, eine Kaskadierung im Inneren des Detektors ein.

Um zu evaluieren, wie sich ein Vereinfachen der inneren Komplexität des Detektors auswirkt, wurde neben dem modernen fHOG Ansatz, als zweites eine die Kaskade reduzierende Variante (rfHOG) untersucht. Die reduzierte Kaskade wird dann hinsichtlich der Qualität und allgemeinen Performance mit der fHOG Implementierung verglichen. Da die Implementierung von fHOG im Kern auf der unmodifizierten HOG Variante nach Dalal und Triggs [7] aufsetzt und diese analog in Form der Wurzelfilter bei Felzenszwalb in das DPM integriert ist, lässt sich die vollständige Reduktion auf den Wurzelfilter als Rückführung auf den klassischen HOG verstehen. Die fertige Umsetzung bietet dabei eine vollständige

Reduktion der Kaskade auf bis zu einen Filter oder eine partielle Rückführung auf die starken Filter der Kaskade an. Es können dafür entsprechend Parameter in einer Konfigurationsdatei gesetzt werden. Zwei exemplarische Konfigurationen werden später in der experimentellen Analyse vorgestellt.

In der dritten untersuchten Variante wird der gelernte Klassifikator einer Regularisierung unterzogen. Der Zweck einer Regularisierung liegt darin, Überanpassung zu vermeiden und evtl. die Generalisierung des Klassifikators zu verbessern, indem versucht wird, eine anschaulich möglichst einfache Klassifikationsfunktion zu erlernen. In der fertigen Implementierung wird eine SVM im Training, unter Verwendung eines Spurklassenoperators, einer nuklearen Norm Regularisierung (NNR) unterzogen.

3.1.1 Reduktion der Klassifikatorkaskade

Die auf fHOG Merkmalen gelernten Klassifikatoren stellen eine Filterkaskade dar, deren Entscheidung sich aus den Antworten der einzelnen Filter berechnet, vergleichbar dem starken Klassifikator, der im AdaBoost-Verfahren in Formel (2.27) beschrieben wurde. Die maßgeblichen Filter im fHOG sind die Wurzelfilter, die bestimmen wo im Bild das DPM den Wurzelort, etwa den Kopf oder Torso, vermutet und in Relation dazu nach passenden Unterteilen, etwa den Extremitäten, filtert.

Aus einem trainierten Klassifikator K_i lassen sich wahlweise nach und nach die schwächsten Filter entfernen, die den geringsten Einfluss auf die Klassifikation haben. Dies entspricht der Reduktion eines Suchbaumes um die am wenigsten einflussreichen Blätter bzw. Zweige. Zu diesem Zweck wurden die Werte c, als prozentuale Annäherung an das klassische Ein-Filter-Modell und s, der minimalen zu erhaltenden Filterstärke eingeführt. In der empirischen Analyse werden gleich eine Menge K von Filtern K_i der Reduktion unterzogen.

Sei außerdem $|K_i|$ die Anzahl der Filter (f_j, w_j) in K_i, wobei f_j der jeweilige Filter ist und w_j die Gewichtungen des Filters.

Algorithmus 1: Reduktion von K

Eingabe: K, c, s
Ausgabe: K'
1 $K' \coloneqq \{\}$
2 **foreach** $K_i \in K$ **do**
3 $K_i' \coloneqq K_i$
4 **while** $|K_i'| > 1$ **and** $\frac{|K_i'|}{|K_i|} > 1 - c$ **and** $\exists\, w_j > \min(w_x | (f_x, w_x) \in K_i')$ **and** $\exists\, w_j < s$ **do**
5 Entferne jedes (f_j, w_j) aus K_i' mit $w_j = \min(w_x | (f_x, w_x) \in K_i')$
6 **end while**
7 $K' \coloneqq K' \cup K_i'$
8 **end while**

Das Ergebnis des Algorithmus 1 ist dann eine Menge von Klassifikatoren K_i', deren Filterstärken wenigstens so groß sind wie s und die wenigstens einen Bruchteil der am stärksten gewichteten Filter aus K_i enthalten, der $1-c$ entspricht, wobei mindestens ein Filter erhalten bleibt. Werden z.B. $c = 1$ und genügend große Werte für s gewählt, bleibt nur der stärkste Wurzelfilter aus K_i erhalten (bzw. die stärksten wenn mehrere gleichstark gewichtete existieren). Für $c = 0$ oder genügend kleine s darf dagegen kein Filter entfernt werden und es gilt $K = K'$.

3.1.2 Regularisierung des Klassifikators mittels nuklearer Norm

Ziel einer Regularisierung des Klassifikators ist eine bessere Generalisierung. Eine Überanpassung an die Trainingsdaten soll vermieden werden, indem bei der Hypothesenauswahl im Trainingsverlauf die weniger komplexen Filter bevorzugt werden (vgl. Abbildung 3.1). Bei der NNR werden die Filter glatter und die Kaskade wird weniger tief, dies führt zu einer schnelleren Klassifikation.

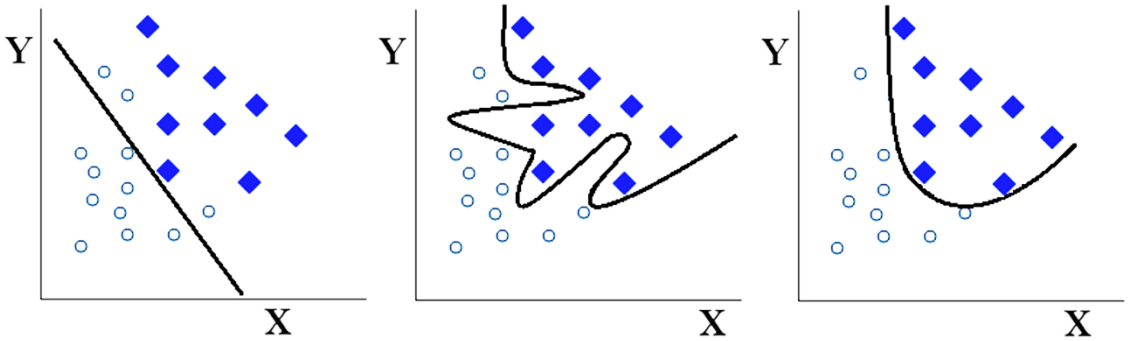


Abbildung 3.1: *Von links nach rechts sind drei mögliche Hypothesen für eine gesuchte Klassifikationsfunktion eingezeichnet. Der linke Klassifikator ist unterangepasst, einige Trainingsbeispiele werden falsch eingeteilt. Der mittlere Klassifikator teilt alle Daten richtig ein, jedoch erscheint die Funktion zu komplex. Der Klassifikator ist ggf. auch an inkorrekte Trainingsbeispiele überangepasst und die Generalisierung auf neuen Daten könnte fehleranfällig sein. Ziel einer Regularisierung ist der rechte Klassifikator. Die Trainingsbeispiele werden korrekt klassifiziert, aber die trainierte Funktion lässt sich viel einfacher beschreiben.*

In der Implementierung durchläuft das gesamte Verfahren die gleichen Schritte wie beim Training eines fHOG Klassifikators. Es wird lediglich dem instanziierten Trainerobjekt ein weiterer Parameter, die Regularisierungsstärke R übergeben, der die NNR einschaltet und gewichtet. Dies fügt dem zu minimierendem Term, der der Formel (2.28) zur Optimierung einer nicht-linearen SVM entspricht, eine weitere Nebenbedingung in Form des Regularisierungsterms $\lambda R(w)$ hinzu. Die Regularisierungsfunktion $R(w)$ ist dabei definiert als $R(w) = ||\sigma(w)||_1$, wobei $||\sigma(w)||_1$ die Schatten-1-Norm oder auch die sogenannte Spurnorm ist, so dass an dieser Stelle im Algorithmus nun optimiert wird nach:

$$Minimiere\ \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m}\xi_i + \lambda R(w) \tag{3.1}$$

$$unter\ der\ Nebenbedingung\ \ y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i\ f\ddot{u}r\ alle\ i, 1 \leq i \leq m$$

Die Regularisierungsstärke λ verhält sich also zur Nebenbedingung der NNR ähnlich wie C zu den Schlupfvariablen ξ_i. Eine hohe Regularisierungsstärke λ betont somit die Suche nach einer kleineren Spurnorm. Da eine kleinere Spurnorm allerdings glattere und ggf. schnellere Filter bewirkt, folgt damit in der Regel eine bessere Generalisierung (vgl. [36]). Ein zu hoher λ-Wert führt dagegen zu einer schlechten Anpassung an die Trainingsdaten und verhält sich damit in etwa analog zu einem zu kleinen C-Wert.

3.2 Training eines Klassifikators

Das Training des Klassifikators ist ein elementarer Schritt bei der formbasierten Roboterdetekion. Wie in Abschnitt 2.4.3 erwähnt, kann das Training einer SVM als die Suche nach der Maximum-Margin-Hyperebene an Hand von vorhandenen Trainingsdaten verstanden werden. Zur Übersicht werden die Kernabläufe des Trainings in Abbildung 3.2 gelistet. In der experimentellen Durchführung erweitert sich dieses Schema noch um das Laden der Trainingsdaten und um das Speichern des gelernten Klassifikators.

Abbildung 3.2: *Ablaufplan der elementaren Schritte beim Training einer SVM*

Das Ergebnis des Trainings ist ein Klassifikator K_i, der für ein gegebenes Erkennungsfenster entscheidet, ob eine Detektion vorliegt oder nicht.

3.2.1 Erkennungsfenster und Auswertungsfunktion

Vor dem eigentlichen Training steht die Findung des idealen Erkennungsfensters des späteren Detektors. Bei der Suche nach Detektionen gleitet ein Erkennungsfenster mit festem Seitenverhältnis über das gesamte Bild (Sliding-Window-Verfahren). Nur dieser aktuelle Teilbereich wird an die SVM weitergereicht und von dieser ausgewertet. Bei allen umgesetzten Verfahren wird außerdem nicht nur das Bild in Originalgröße durchlaufen,

sondern es wird eine wie in Abschnitt 2.2.3.1 beschriebene Bildpyramide aufgebaut. Die Verfahren werden dadurch skaleninvariant.

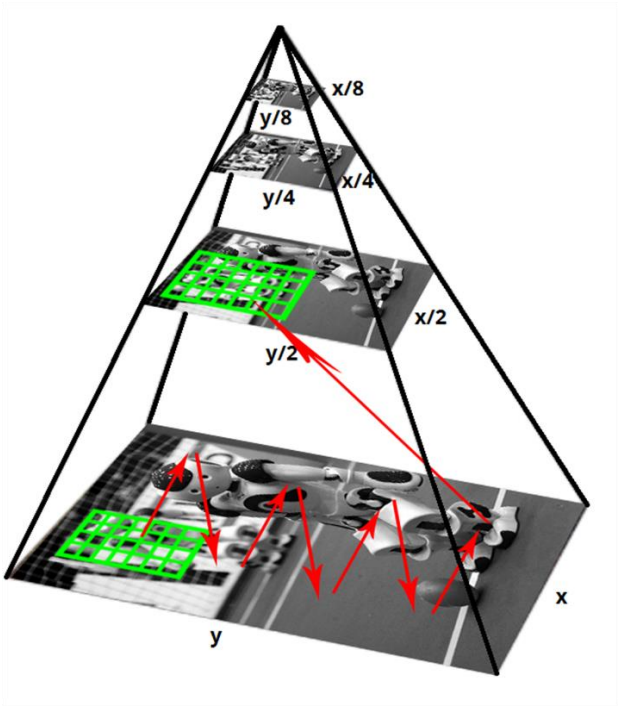


Abbildung 3.3: *Bildpyramide wie sie von einem HOG-Detektor aufgebaut wird. Das Erkennungsfenster ist grün eingezeichnet, sein Raster entspricht den im HOG-Verfahren berechneten Blöcken von Histogrammen. Die roten Pfeile zeigen den Pfad in dem das Erkennungsfenster über das Bild fährt und Merkmalsvektoren auswählt, die dem Klassifikator übergeben werden.*

Die Bildpyramide hat ein Skalenbild pro Oktave. Die Größe des Erkennungsfensters entspricht einer festen Anzahl von HOG-Blöcken und damit Pixeln (vgl. Abschnitt zum Aufbau des HOG-Merkmalsvektors 2.3.1.1). Die Größe des Erkennungsfensters bleibt von Oktave zu Oktave der Bildpyramide gleich und deckt mit kleiner werdendem Bild einen größeren Bildbereich ab (vgl. Abbildung 3.3). Der durch das Erkennungsfenster ausgewählte HOG-Merkmalsvektor bleibt also gleich lang. Das Erkennungsfenster gleitet blockweise über die Bilder. Klassifiziert die SVM den jeweiligen Merkmalsvektor als Roboter, wird genau das aktuelle Erkennungsfenster als eine positive Detektion vom Algorithmus zurückgeliefert.

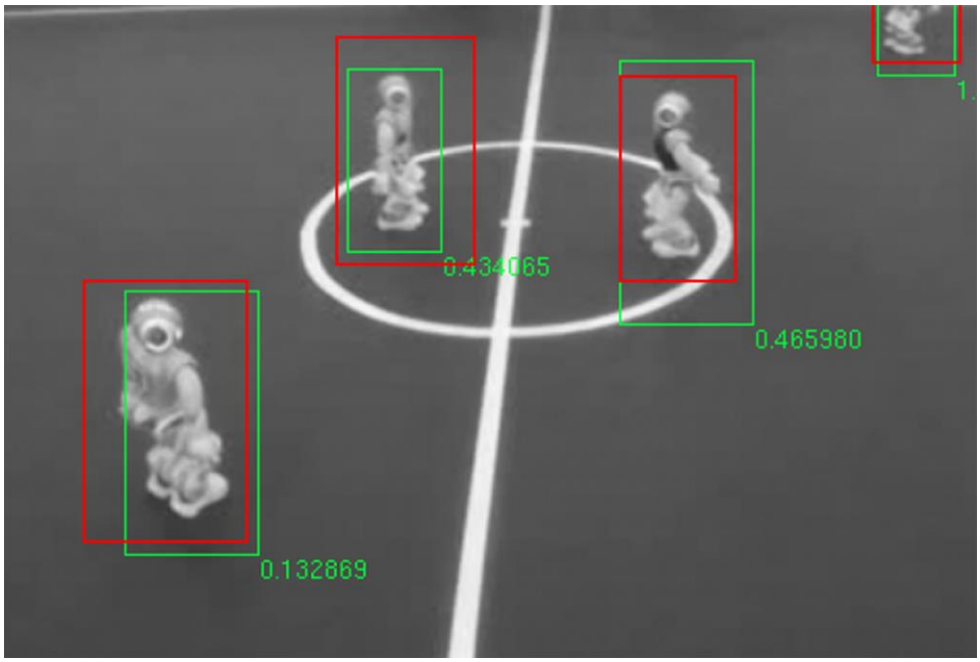


Abbildung 3.4: *Exemplarische Auswertung eines Detektors. Grün eingezeichnet sind die tatsächlichen Erkennungen eines Detektors, die Zahlen sind dabei als Konfidenz in die Detektion zu verstehen. Rot gegenübergestellt sind die Markierungen der Positivbeispiele wie sie manuell in die Trainingsdatei eingetragen wurden. Beim Roboter unten links überlappen sich Markierung und Detektion nicht vollständig. Beim Roboter in der Mitte ist das markierte Positivbeispiel viel größer als die Detektion und bei dem Roboter rechts ist es genau umgekehrt, die Markierung liegt vollständig innerhalb der Detektion. Die grünen Rechtecke der Detektion sind zwar unterschiedlich groß, stehen aber im selben Seitenverhältnis. Bei den roten Markierungen ist das Seitenverhältnis willkürlich.*

Interessant ist dies vor allem für die statistischen Auswertung des Detektors. Hier stellt sich die Frage, welche Detektion überhaupt als richtig positiv qualifiziert werden kann (vgl. Abbildung 3.4). Eine Detektion, die das gesamte Bild zurückliefert, ist erst einmal nicht falsch, wenn das Bild an einer beliebigen Stelle einen Roboter enthält. Dennoch ist sie praktisch falsch bzw. schwer verwertbar. Die Detektion kann demnach genauso gut als falsch positiv bewertet werden oder sie wird bei der Auswertung ignoriert. Umgekehrt kann eine Detektion aber auch zu eingeschränkt sein und nur Teile des Roboters beinhalten, etwa nur den Kopf.

Es ist also notwendig eine Funktion zu definieren, die ein richtig positives Beispiel beschreibt. Dollar, P., et al. [3] legen im Abschnitt "Evaluation Methodology" eine Möglichkeit dar, dieses Thema anzugehen. Es wird die konkrete Implementierung der Auswertungsfunktion einsehbar gemacht und völlig richtig dargelegt, dass es notwendig ist, den exakt selben Auswertungscode für jedes eingesetzte Verfahren zu verwenden, um innerhalb der eigenen Experimente Unvoreingenommenheit zu garantieren. Andere Arbeiten aus der verwiesenen Literatur [8, 11, 13], etwa zur Beschleunigung des Verfahrens durch Formen der Kaskadierung, erschweren den Vergleich mit der eigenen Implementierung. Ohne Angaben zur Auswertungsfunktion kann eine falsch positive Detektion auf 1000 Bildern entweder für einen qualitativ sehr guten Detektor oder für eine

zu großzügige Auswertungsfunktion sprechen. Führt eine strengere Auswertungsfunktion zu z.B. 5 falsch positiven Detektionen pro 1000 Bildern, verändert dies die gemessenen Qualitätsmaße erheblich.

Möchte man die Qualität der eigenen Detektion mit anderen Verfahren vergleichen, ist man daher gezwungen, diesen Algorithmus oder zumindest einen Algorithmus der bereits zu diesem in Relation gesetzt wurde, selber umsetzen.

Die für diese Arbeit verwendete Definition einer richtig positiven Detektion wurde bewusst streng gewählt. Im Zweifel soll die praktische Detektion bessere Ergebnisse liefern, als nach den Schätzungen der Qualitätsmaße zu erwarten ist. Sei die Fläche D eine rechteckige Detektion und sei die Fläche M eine rechteckigen Markierung. Seien $|M|$, $|D|$, $|M \cap D|$, $|M \cup D|$ die jeweiligen Flächeninhalte. Seien außerdem $0 \geq \tau_v, \tau_g \leq 1$ frei wählbare Schwellwerte, die bestimmen, zu welchen Anteilen die Flächen sich überschneiden müssen.

Mit

$$d_1(M, D) \coloneqq \frac{|M \cap D|}{|M \cup D|} \tag{3.2}$$

und

$$d_2(M, D) \coloneqq \max\left[\frac{|M \cap D|}{|M|}, \frac{|M \cap D|}{|D|}\right] \tag{3.3}$$

zählt die Auswertungsfunktion $a(D)$ eine Detektion D dann als richtig (1) bzw. falsch (0) positiv, wenn

$$a(D) = \begin{cases} 1, & \exists M\colon d_1(M, D) \geq \tau_v \wedge d_2(M, D) \geq \tau_g \\ 0, & sonst. \end{cases} \tag{3.4}$$

Für die Auswertung in späteren Kapiteln des vorliegenden Textes wurden die Schwellwerte $\tau_v = 0{,}5$ und $\tau_g = 1$ gewählt. Das bedeutet, die gemeinsame Schnittfläche muss wenigstens halb so groß sein wie die Vereinigung beider Flächen (Bedingung (3.2)) und eines der Rechtecke muss das andere Rechteck komplett enthalten (Bedingung (3.3)). Gerade die zweite Bedingung ist besonders restriktiv, eine subjektiv gute Erkennung, wie unten links in der Abbildung 3.4, würde als falsch positiv gezählt werden.

Hervorzuheben ist, dass die Zählung einer Detektion als richtig oder falsch positiv schon beim Trainingszyklus der SVM eine wichtige Rolle spielt. Eine Veränderung der Schwellwerte führt damit auch zu einer veränderten Anpassung der SVM an die Trainingsdaten bis hin zum Overfitting. Diese Anpassung wird, wie im Abschnitt 2.4.3 beschrieben, über die Schlupfvariablen und den Wert C gesteuert und kann daher bei einer Parameteroptimierung des Klassifikators automatisch reguliert werden.

Es entstehen jedoch durch die Einschränkungen eines festen Seitenverhältnisses des Erkennungsfensters und durch die Auswertungsfunktion Situationen, in denen Markierungen von einer SVM nie als richtig positiv erkannt werden können. In der

Implementierung werden daher zunächst die Zielmaße des Erkennungsfensters als Mittelwerte der Breiten und Höhen der gegebenen, markierten Positivbeispiele bestimmt. Diese Maße können dann, falls notwendig, noch vom Entwickler skaliert werden, danach ist das Erkennungsfenster im Seitenverhältnis fest.

3.2.2 Vorbereitung der Trainingsdaten

Die Vorbereitung der Trainingsdaten geschieht in mehreren Etappen. Bereits mit dem Laden selber beginnt der erste vorbereitende Verarbeitungsschritt. Es werden die Farbinformationen der Bilder verworfen bzw. in ein Graustufenbild umgewandelt, die Bilder werden in einem 2D-Feld gespeichert, ein Pixel entspricht dabei einem Byte.

Wenn nur wenige Eingabebilder vorhanden sind, werden aus diesen neue erzeugt. Die Bilder und die Markierungen werden dazu horizontal gespiegelt. Spiegelung ist dabei die einfachste Möglichkeit neue Positivbeispiele zu generieren. Falls es notwendig ist, noch weitere Beispiele zu erzeugen, gibt es dafür elaborierte Methoden. Farazi und Behnke rotieren die Eingabebilder [17]. Nach Viola und Jones [30] lassen sich große Mengen an Beispielen erzeugen, indem die ausgeschnittenen Positivbeispiele gegebenenfalls perspektivisch verzerrt, rotiert oder skaliert auf neue Hintergründe kopiert werden. Entsprechende Skripte und Algorithmen finden sich online zur Genüge.[21] Die erstellten Datensätze erzielten bereits gute Ergebnisse beim Training der Klassifikatoren. Zur Frage, ob die Generierung zusätzlicher Eingabedaten bessere Detektoren liefert, sind noch weitere Untersuchungen vorzunehmen.

In den ersten Tests hat die Skalierung der Eingabedaten auf doppelte Größe deutlich bessere Erkennungsraten ermöglicht, was daher in allen Verfahren vor dem Training durchgeführt wird. Damit der HOG-Detektor ein gelerntes Objekt in verschiedenen Skalierungen wiederfinden kann, bietet es sich an, bei der Detektion eine Skalenpyramide ähnlich den Oktaven im SIFT (vgl. Abbildung 2.12) aufzubauen und zu durchlaufen. Im Unterschied zu SIFT werden dabei aber nicht skaleninvariante Merkmale als lokale Extrema des Scale-Space erzeugt, sondern es wird die Detektion iterativ auf kleiner werdenden Skalierungen ausgeführt. In der Regel führt die Vergrößerung der Basis, also des originalen Eingabebildes, dabei zu mehr Detektionen, denn die kleinere Version sollte trotzdem noch als Spezialfall durchlaufen werden. Gleichzeitig führt dies aber unmittelbar zu einem langsameren Detektor und damit zu einem direkten Tausch zwischen schnellerem Durchlauf oder mehr Detektionen. Die Gütekriterien der Klassifikation sind so erheblich verbessert worden, dass die Vergrößerung der Bilder fest in den Trainingsablauf aufgenommen wurde. In einigen Experimenten (siehe Abschnitt 5.3) werden im Verlaufe der k-fachen Kreuzvalidierung die ausgelassenen Bilder der übersprungenen Teilmenge nicht größer skaliert. Dadurch können

[21] Zum Beispiel bietet OpenCV diese Funktionalität an und beschreibt sie in ihrer Dokumentation zum Thema "Cascade Classifier Training", http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html, abgerufen am 22.01.2017

mit geringem Aufwand Datenreihen gewonnen werden, die sich direkt mit der hochskalierten Variante aus dem gleichen Testdurchlauf vergleichen lassen.

In einem weiteren Vorbereitungsschritt werden aus der Liste der positiven Markierungen diejenigen entfernt, die für den HOG-Algorithmus gemäß der Auswertungsfunktion nicht erreichbar sind. Vorstellbar ist z.B. die Situation, dass der HOG-Algorithmus generell nach einem aufrecht stehenden Roboter sucht, das Erkennungsfenster ist doppelt so hoch wie breit, eines der markierten Trainingsbeispiele aber ein umgefallener Roboter ist und die Markierung damit in etwa doppelt so breit wie hoch ist. Es wird sich daher keine Detektion finden lassen, so dass diese die Markierung zur Hälfte überschneidet und vollständig in ihr enthalten ist (vgl. Abschnitt 3.2.1).

3.2.3 Berechnung des HOG-Merkmalsvektors

Im letzten Vorbereitungsschritt werden für alle eingelesenen Bilder die HOG-Merkmalsvektoren berechnet (siehe Abbildung 3.5). Der implementierte Extraktor folgt bei der Berechnung grundsätzlich dem in Abschnitt 2.3.1.1 beschriebenen Verfahren nach Dalal und Triggs, die Blöcke sind dabei 2×2 Zellen à 5×5 Pixel groß und die Histogramme unterscheiden die Gradientenrichtung in 9 Klassen. Der lokale HOG-Merkmalsvektor eines insgesamt 10×10 Pixel großen Blocks ist damit 36-dimensional und die Dimensionalität des gesamten HOG-Deskriptors eines Bildes bzw. eines Erkennungsfensters ergibt sich entsprechend.

Die lokalen Merkmalsvektoren werden allerdings in einer Bag-of-visual-words (BoW) Repräsentation zusammengefasst. Das Verfahren ist beschrieben bei Lazebnik und Ponce [37]. Für die Aufteilung in ein BoW ist die aktuelle Lage eines lokalen Merkmals relativ unwichtig, lokale Merkmale die sich ähneln, werden in einer höheren Struktur gebündelt, etwa in Form von Histogrammen von Merkmalen. BoW entspricht damit der Zerlegung nach Felzenszwalbs DPM. Eine weitere Form dieser Repräsentation lokaler Merkmalsvektoren wurde schon im Abschnitt 2.2.3.3, bei der Zusammensetzung des Deskriptors im SIFT-Algorithmus, vorgestellt. Ist die Zusammensetzung des HOG-Merkmalsvektors abgeschlossen, sind alle Vorbereitungen getroffen, um den Trainingszyklus des Klassifikators zu beginnen.

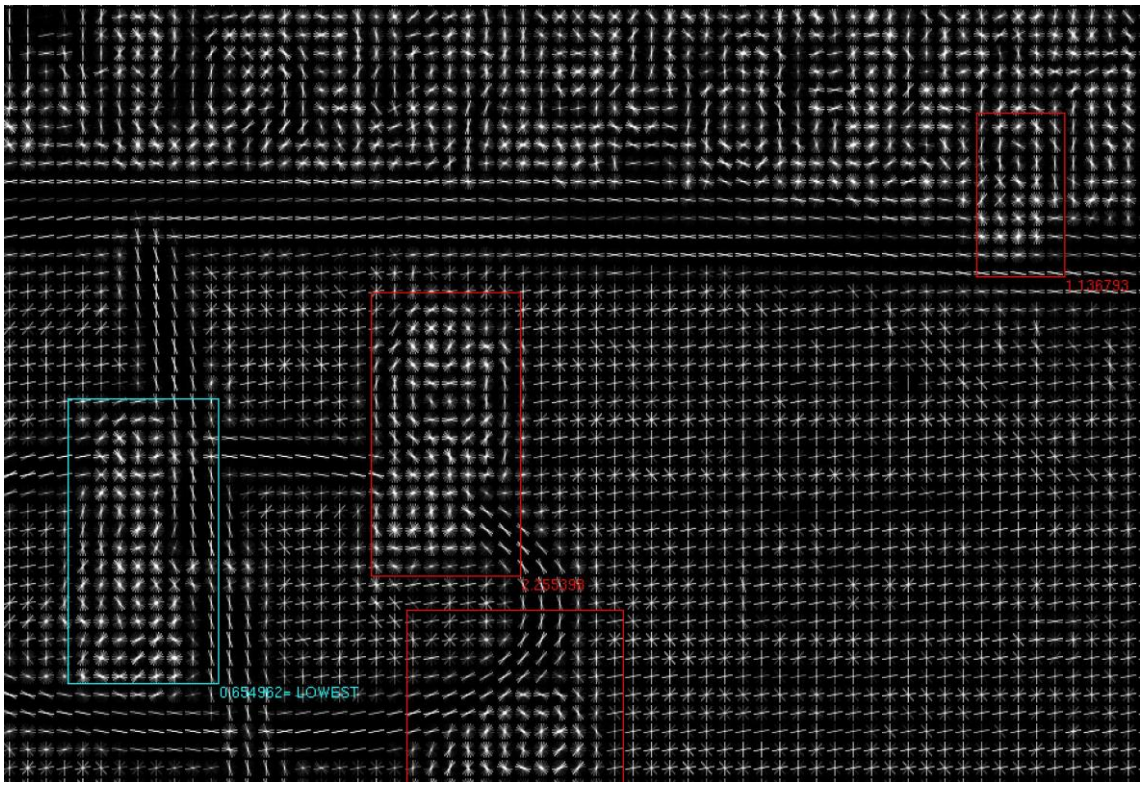


Abbildung 3.5: *Visualisierung der berechneten HOG-Merkmale aus einem Spiel der RoboCup EuropeanOpen 2016. In diesem vergrößerten Ausschnitt lassen sich blau und rot markiert die Roboter schemenhaft erkennen. Die Zahlen geben die Konfidenz in die Detektion an. Am oberen Rand befindet sich eine Zuschauermenge, die sich in ihren Merkmalen stark den Kopf- und Schulterpartien der anthropomorphen Roboter ähnelt, weshalb es in diesen Bereichen häufiger zu Fehldetektionen kommt.*

3.2.4 Trainingszyklus

Das eigentliche Training findet in Form von wiederholten Trainingsiterationen statt. In jedem Trainingsschritt wird dabei versucht, das Optimierungsproblem der Maximierung des kleinsten Abstands der Trainingsdaten zur Hyperebene zu lösen, wie im Kapitel über Stützvektormaschinen 2.4.3 beschrieben.

Typischerweise würden dafür aus nicht markierten Flächen in den Eingabedaten mindestens so viele Negativbeispiele generiert werden, dass in einem balancierten Satz aus Positiv- und Negativbeispielen diese Ebene gesucht werden kann. In folgenden Iterationen werden anschließend die Negativbeispiele um falsch positive Erkennungen, also die harten Negativbeispiele, ergänzt und es wird ein weiteres Mal gesucht. Die verwendete Trainingsmethode löst dieses Problem basierend auf dem 2015 von Davis King formulierten MMOD-Algorithmus. [35] Es wird dabei auf eine Variation der Auswertungsfunktion (3.4) zurückgegriffen und das Optimierungsproblem als Maximierung der Summe der Bewertungen aller möglichen, sich nicht überschneidenden Erkennungsfenster formuliert. Zum Finden der optimalen Bewertungsfunktion wird dabei ein Schnittebenenverfahren eingesetzt, Details zu diesem lassen sich den Arbeiten [35, 38] entnehmen. Das

Schnittebenenverfahren bietet eine iterative Lösung linearer Optimierungsprobleme. Die Relaxation der Ganzzahligkeitsbedingung entspricht der Einführung des Fehlerterms, bestehend aus den Schlupfvariablen ξ_i, wie es zum Umgang mit schwer linear trennbaren oder verrauschten Daten in Abschnitt 2.4.3 beschrieben wurde. Mit jedem Iterationsschritt wird die Nebenbedingung verschärft und die Approximation verbessert.

Hervorzuheben ist, dass diese Suche über alle möglichen Erkennungsfenster optimiert und damit wesentlich mehr Informationen aus den gegebenen Trainingsbildern extrahiert, als das herkömmliche Verfahren, bei dem nur eine Teilmenge in Form negativer Beispiele generiert wird. Das Resultat jeder Iteration ist neben einem möglichen Klassifikator auch ein Risikowert und der Fehlerterm, der über den Parameter C skaliert werden kann. Es bietet sich daher an die Suche zu beenden, wenn dieser Fehlerterm klein genug ist. Bei einer Parameteroptimierung kann entsprechend nach verschiedene Abbruchbedingungen gesucht werden.

3.3 Implementierung

Nachdem die entwickelte Roboterdetektion bisher überwiegend in ihrer Algorithmik beschrieben wurde, widmet sich der folgende Abschnitt praktischen Details bei der Implementierung der Verfahren.

Zunächst wird die Auswahl der tatsächlich eingesetzten Technologien begründet. In der Abwägung wurde in erster Linie im Sinne einer gut funktionierenden Detektion ausgewählt. Sekundär wurden die im Abschnitt 1.2 vorgestellten Rahmenbedingungen berücksichtigt.

Anschließend werden implementierungsspezifische Prozessabläufe und einige konkret realisierte Implementierungsdetails vorgestellt. Insbesondere wird das Laden der Trainingsdaten beschrieben, es werden Überlegungen zum physischen Speicher dargelegt und das Protokollieren der Trainingsresultate, das heißt trainierter Klassifikatoren und etwaiger Messwerte, wird ausgeführt.

3.3.1 Auswahl und Vorstellung eingesetzter Technologien

Für die Implementierung wurden zwischen verschiedenen Programmiersprachen, Entwicklungsumgebunden und Bibliotheken diejenigen erwogen, die am besten die Ansprüche der angestrebten Lösung (vgl. Abschnitte 1.1 und 1.2) abgedeckt haben.

Unter Berücksichtigung, dass die gesamte Implementierung dem NaoTH zur Verfügung gestellt wird und dass die Algorithmen zu Teilen auf den Nao Robotern ausgeführt werden, fiel die Wahl der verwendeten Programmiersprache auf C++. Die Codebasis des NaoTH und der Großteil der im Wettkampf auf dem Roboter ablaufenden Programme sind in C++ implementiert. Der entstandene Programmcode ist daher auf den Roboter portierbar und lässt sich außerdem in die Werkzeuge des Teams einbinden. Weiterhin finden sich für die C++-Programmiersprache ausgearbeitete, frei verfügbare Bibliotheken für die

Bildverarbeitung und das maschinelle Sehen an. Die populärste Bibliothek ist OpenCV, die unter der BSD-Lizenz[22] (Berkeley Software Distribution, University of California, Berkeley) zur Verfügung steht. Die ersten durchgeführten Implementierungsansätze basieren auf OpenCV und griffen dabei in Teilen auf die Module zur HOG-Merkmalsgewinnung, zum Training der SVM-Klassifikatoren und zum Boosting (vgl. Abschnitt 2.4.2) zurück. Einige Bilder aus dem Grundlagenkapitel, wie etwa die in die R-HOG-Zellen eingezeichneten Histogramme in Abbildung 2.20, entstammen dieser Version.

Für die finale Version der Implementierung wurde dann auf die Dlib Bibliothek [39] umgestiegen. Diese Programmbibliothek steht unter der mit der BSD-Lizenz vergleichbaren Boost Software Lizenz[23] und bietet neben hochportablem Code umfangreiche Komponenten aus den Bereichen maschinelles Lernen, Bildverarbeitung, Numerik und anderen. Die Schwerpunkte liegen dabei klar in den Bereichen der Statistik und des maschinellen Lernens und Sehens. Je nach Anwendungsfall finden sich häufig hochspezialisierte Module in der Programmbibliothek, die jedoch problemlos erweitert oder umgeschrieben werden können. Weil durchweg das Programmierparadigma Design by contract (vertragsbasierter Entwurf) eingehalten wird, ist, über einfache Schnittstellen hinausgehend, das Zusammenspiel der Komponenten sichergestellt. So war es möglich, die Dlib-Klasse zur Kreuzvalidierung reibungslos durch eine eigene Implementierung zu ersetzt, als diese die notwendige Flexibilität zur Verwendung mit den für die Arbeit erstellten Datenbanken vermissen ließ, ohne dabei auf die Generierung der fHOG-Merkmale nach Felzenszwalb direkt aus der Bibliothek heraus verzichten zu müssen.

Die Bibliotheksroutinen sind gut getestet und hardwarenah umgesetzt, es erfordert nur ein paar Compilermakros, damit eine Vielzahl der Instruktionen durch den aktuellsten Befehlssatz gängiger, moderner Prozessoren ausgeführt werden. Die Bibliothek bietet außerdem Möglichkeiten zur Parallelisierung einzelner Programmteile. Auf das nebenläufige Ausführen des Programms wurde vor allem bei den Algorithmen zum Training der Klassifikatoren zurückgegriffen. Ein weiterer praktischer Nebeneffekt war die Reduktion der manuellen Arbeit beim Erstellen der Positiv- und insbesondere der schweren Negativbeispiele durch Hilfsmethoden der Dataminingkomponenten der Bibliothek. Wurden so bei der OpenCV Implementierung noch alle Negativbeispiele von Hand erstellt und die schweren Negativbeispiele programmatisch hinzugefügt, wodurch unfreiwillig ein Bias durch nicht balancierte Trainingsdaten eingeführt wurde, konnte diese Fehlerquelle durch die Befähigung zum statistischen Bootstrapping (aus dem Englischen, Verfahren zur Generierung von Stichproben) der Dlib ausgemerzt werden.

Die Dlib Bibliothek ist vollständig in C++ und zu Teilen in Python verfügbar. Im Unterschied zu OpenCV gibt es für Dlib zwar keinen Java Wrapper, dies ist aber für den geplanten Verwendungszweck nicht relevant. Außerdem besteht theoretisch die Möglichkeit, Dlib in Form einer dynamisch gelinkten Bibliothek anderen Sprachen zur Verfügung zu

[22] https://opensource.org/licenses/bsd-license.php, abgerufen am 22.01.2017
[23] http://www.boost.org/users/license.html, abgerufen am 22.01.2017

stellen. Weiterhin sind Dlib und OpenCV gut miteinander kompatibel. Mathematische Objekte wie die rechteckigen Detektionen oder Matrizen lassen sich problemlos in das passende Format der jeweils anderen Bibliothek überführen. Für einige Objekte wie z.B. Bilder, also im Prinzip Matrizen mit Metadaten, die etwa den Farbraum bestimmen, besitzt Dlib sogar explizit Klassen und Methoden zur Konvertierung von und nach OpenCV. Beispielsweise wurde in einem implementierten Test eines auf fHOG basierenden Trackers OpenCV bemüht, um zur Laufzeit aus einem Eingabevideo einzelne Bilder zu extrahieren. Diese Funktionalität ist so nicht durch Dlib abgedeckt. Die Bilder wurden dann den Dlib-Methoden konvertiert übergeben.

Zusammenfassend wurden die folgenden Technologien zur Umsetzung der im Rahmen dieser Arbeit gesteckten Ziele gewählt. Die eingesetzte Programmiersprache ist C++. Der Programmcode selber wurde in der Visual Studio 2013 Entwicklungsumgebung geschrieben. Diese ist funktional und wird den Studenten durch den Softwareservice der Humboldt-Universität zu Berlin frei zur Verfügung gestellt. Es wurde der Dlib Programmbibliothek der Vorzug gegeben, die sich als Sammlung von Algorithmen zum maschinellen Lernen mit Erweiterungen in Bereiche wie dem maschinellen Sehen versteht, gegenüber OpenCV, die sich als Sammlung von Algorithmen im Bereich maschinelles Sehen mit Erweiterungen in Bereiche wie maschinelles Lernen versteht. Zusätzlich wurde auf die Werkzeuge des NaoTH-Frameworks zurückgegriffen. Verwendet wurden Python-Skripte zur Interpretation der Roboter-Protokolldateien, das Analyseprogramm RobotControl und C++-Methoden aus dem Wahrnehmungsmodul der Nao Roboter. In der Auswertung der Verfahren wurde zur Erstellung der Diagramme auf die Programmiersprache R und die Module gplots[24] und plotly[25] (Plotly Inc, Montreal) zurückgegriffen.

3.3.2 Implementierter Programmablauf

Das in Abschnitt 3.2 beschriebene Training einer SVM wurde in der Praxis in ein vollständiges Programm eingebettet. In der einfachsten Version lädt ein solches Programm ausgewählte Trainingsdaten in den Speicher und stellt sie dem Trainingsprozess zur Verfügung (vgl. Abbildung 3.6). Nach abgeschlossenem Training wird der gelernte Klassifikator gespeichert. Für die empirische Analyse wurde dieser Prozess weiter ausgebaut, indem automatisiert ein Parameterraum von Klassifikatoren trainiert wird und indem ein jeweils trainierter Klassifikator auf den bereits geladenen Trainingsdaten evaluiert wird.

[24] https://cran.r-project.org/web/packages/gplots/index.html, abgerufen am 22.01.2017
[25] https://plot.ly/r/, abgerufen am 22.01.2017

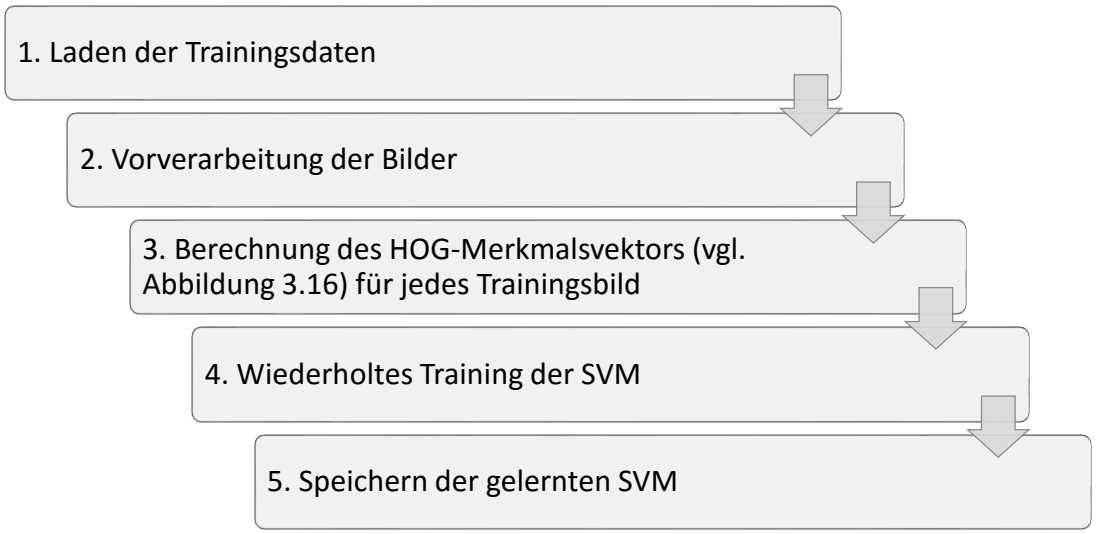


Abbildung 3.6: *Ablaufdiagramm des implementierten Programms zur Erstellung eines Klassifikators. Das Training aus Abbildung 3.2 wird um die notwendigen Schritte zum Laden der Trainingsdaten und zum Speichern des Klassifikators erweitert.*

3.3.3 Laden der Trainingsdaten

Die verwendeten Trainingsdaten liegen in zwei Datensätzen vor, wie diese erstellt wurden, wird in Abschnitt 4.1 beschrieben. Hier sei bereits vorweggenommen, dass die Trainingsdaten im Wesentlichen aus zwei Komponenten bestehen: Den aus Videos fußballspielender Roboter extrahierten Bildern und den in XML-Dateien beschriebenen Markierungen der Positivbeispiele.

Der erste Schritt einer Trainingssitzung ist das Laden der Trainingsdaten, so wie diese in den Datensätzen vorbereitet sind. Der Aufbau innerhalb der XML-Datei ist wie für das Dateiformat üblich hierarchisch strukturiert. Sie enthält zunächst Metadaten zur XML-Version und -Codierung und Beschreibungen des Datensatzes. Die eigentlichen Markierungen sind in Bildobjekten untergebracht, die ihrerseits auf den Pfad der Datei selber verweisen und außerdem Objekte von rechteckigen Boxen enthalten. Der Pfad kann dabei relativ oder absolut sein. Die Rechtecke sind durch einen Punkt und durch ihre Höhe und Breite beschrieben. Innerhalb jeder Box ist im Bild ein aktiver Roboter zu finden, das heißt jedes Rechteck repräsentiert ein Positivbeispiel, mit Ausnahme von solchen Boxen deren Status als zu ignorieren gesetzt ist. Der Auszug 1 aus einer der XML-Dateien zeigt zwei Bilder mit insgesamt 6 markierten Rechtecken, das erste Bild enthält zwei solcher zu ignorierenden Bereiche. Eine Markierung nicht-positiver, aber auch nicht-negativer Bereiche war deshalb notwendig, weil im Training die Negativbeispiele aus allen nicht als Positivbeispiel markierten Flächen generiert werden. Befinden sich in den Bildern stark an die zu erlernenden aktiven Roboter erinnernde Bereiche, etwa andere humanoide Roboter, Nao Roboter die passiv sind oder auf T-Shirts gedruckte Roboterfiguren (vgl. Abbildung 4.1), dann werden diese gegebenenfalls in der Trainingsphase der SVM als Negativbeispiel

gesamplet. Im schlechtesten Fall führt dies dazu, dass solche Negativbeispiele als harte Negativbeispiele immer wieder der SVM vorgeführt werden, die ihrerseits diese Beispiele kaum von den korrekten Positivbeispielen unterscheiden kann. Der Klassifikator würde sich sukzessive verschlechtern, bis diese unscharfen, hart negativen Beispiele erlernt sind.

Auszug 1: Beispielhafte XML-Datei

```
1  <?xml version='1.0' encoding='ISO-8859-1'?>
2  <?xml-stylesheet type='text/xsl' href='metadata_stylesheet.xsl'?>
3  <dataset>
4  <name>2016-03-29 RC-EuropeanOpen-Eindhoven dataset</name>
5  <comment>Author: Dominik Krienelke</comment>
6  <images>
7    <image file='half1\00019.jpg'>
8      <box top='324' left='32' width='68' height='79'/>
9      <box top='331' left='212' width='89' height='105'/>
10     <box top='118' left='725' width='37' height='27' ignore='1'/>
11     <box top='125' left='763' width='29' height='26' ignore='1'/>
12   </image>
13   <image file='half1\00020.jpg'>
14     <box top='284' left='62' width='63' height='76'/>
15     <box top='333' left='215' width='89' height='95'/>
16   </image>
17 </images>
18 </dataset>
```

Die XML-Dateien werden von einem Parser verarbeitet und alle gelisteten Bilder und Markierungen werden in den Arbeitsspeicher geladen. In der Regel wird nur eine XML-Datei geladen, mit der trainiert werden kann. Die Datensätze wurden so aufgebaut, dass eine XML-Datei die Bilder aus genau einem Video beschreibt. Für den Parser und die Programmlogik im Allgemeinen stellt es kein Problem dar mehrere Videos in einer Markierungsdatei zusammenzufassen. Für das bei der späteren Modellauswahl durchgeführte, umfangreichere Kreuzvalidierungsverfahren, wurde das Programm dahingehend erweitert, nicht eine Datei entgegenzunehmen, sondern einen Verzeichnispfad. Alle in dem Pfad gefundenen XML-Dateien werden dabei durch den Parser geschleust und geladen.

3.3.4 Optimierung der Ladezeiten

Beim Laden der Trainingsdaten in den physischen Speicher wird versucht, alle Bilder, deren Skalierungen, die HOG-Merkmale, so wie alle generierten Positiv- und Negativbeispiele über die gesamte Dauer des Trainings hinweg im Arbeitsspeicher zu halten, um schnelle Zugriffszeiten auf alle Daten zu ermöglichen.

Ein Bild aus den Trainingsdaten ist mit 1920×1080 Pixeln bereits über 2 MB groß. Weil dazu außerdem noch die daraus gezogenen Daten kommen, können, je nach Limitierungen durch das Betriebssystem und durch die vorhandene Maschine, die Grenzen des zur

Verfügung stehenden Arbeitsspeichers überschritten werden. Falls dieser Punkt erreicht wird, ist das Betriebssystem fortan permanent damit beschäftigt, dem Programm virtuellen Arbeitsspeicher zur Verfügung zu stellen und die Daten sinnvoll zwischen diesem und dem echten RAM zu verteilen. Da die Zugriffszeit selbst auf schnellen Festplatten um Potenzen langsamer ist, als auf den Arbeitsspeicher, wird der Trainingsalgorithmus in diesem Falle erheblich ausgebremst. Das Training einer einzelnen SVM, das in der Regel nur Minuten in Anspruch nimmt, kann so Stunden dauern. Bei dem Training einer SVM auf einem Video ist dies, natürlich abhängig vom System, nie der Fall. Bei der Kreuzvalidierung, bei der Teildatensätze aus allen Videos aufgebaut werden, ist dies jedoch ein realistisches Szenario. Im Gesamtkontext der Aufgabenstellung ist dies aber ein vernachlässigbares Problem. Das Training und insbesondere die Suche nach optimalen Parametern, ist eine einmalig durchzuführende Aufgabe. Wenn der resultierende Detektor gute Erkennungen liefert und performant läuft, dann ist es irrelevant, dass sein Training gegebenenfalls Tage gedauert hat.

Soll mit der umgesetzten Lösung in Zukunft unterwegs ein Klassifikator auf die besonderen örtlichen Gegebenheiten trainiert werden, sollte dies auch mit z.B. mobilen Rechnern funktionieren, die vielleicht mit nur 4 bis 8 GB RAM ausgestattet sind. Beim Training sollte dann die Speicherlast beobachtet werden und unter Umständen muss die Anzahl der Trainingsbilder reduziert werden. Für die im Rahmen der Diplomarbeit durchgeführten Experimente, die zum Teil auf den gesamten Datensätzen operieren, wurde eine relativ potente Maschine mit einer Ausstattung von 12 GB Arbeitsspeicher bemüht (vgl. Anlage A.3).

3.3.5 Speichern des Klassifikators

Das Training einer SVM ist ein einmaliger Vorgang. Nach beendetem Training wird der gelernte Klassifikator serialisiert im persistenten Speicher abgelegt. Die trainierte SVM kann immer wieder geladen und zur Detektion humanoider Roboter auf neuen Daten verwendet werden. Für alle im Rahmen der Arbeit trainierten Klassifikatoren wurde darüber hinaus auch eine Visualisierung des gelernten HOG-Merkmalsvektors berechnet und als PNG-Datei gespeichert. Die Visualisierung konvertiert die HOG-Zellen in ein Graustufenbild in einer Weise, die die Magnitude und Orientierung der Gradientenenergie in jeder Zelle zeigt. Dies entspricht in etwa der radialen Darstellung des entsprechenden Histogramms. Eine solche Darstellung ist objektiv nicht gut zu verwerten, bietet aber auf einen Blick die Möglichkeit subjektiv problematische oder interessante Klassifikatoren zu identifizieren (vgl. Abbildung 3.7).

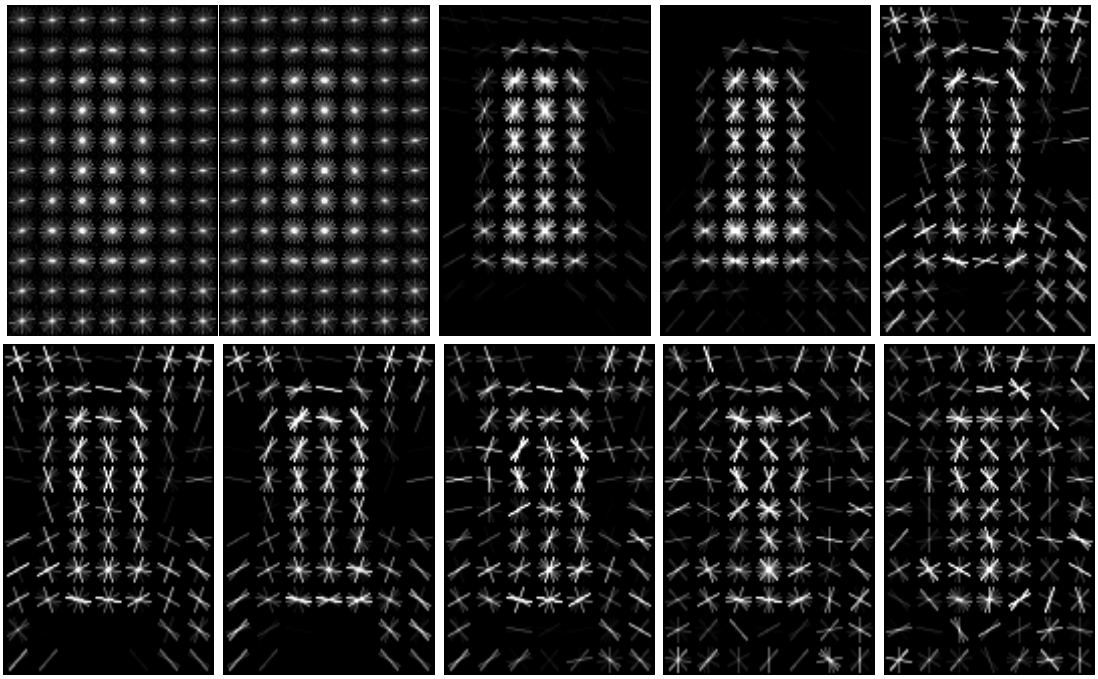


Abbildung 3.7: *Sukzessive Progression durch eine Folge von Trainingsexperimenten mit steigender Gewichtung des Fehlerterms in der SVM. Die ersten beiden Klassifikatoren (in Leserichtung) sind augenscheinlich uniform in ihren Merkmalen, es ist schwer vorstellbar, dass diese in der praktischen Anwendung gute Resultate liefern. Mit besserer Parametrisierung verdichtet sich die Gradientenenergie im Zentrum (Bilder 3-4) und entwickelt von dort Ausläufer (Bilder 5-7). Die letzten Klassifikatoren haben eine sehr geringe Fehlertoleranz und trennen die Trainingsbeispiele perfekt, sie versagen aber bei neuen Daten. Ihre Merkmalsausprägungen wirken willkürlich.*

3.3.6 Protokollierung von Messwerten

Wenn die gelernten Klassifikatoren auf Testbildern ausgeführt werden, erzeugt das implementierte Programm Protokolldateien dieser Tests. In der Regel werden diese Tests im Stapel durchgeführt.

Zum Beispiel wird in der experimentellen Analyse in Kreuzvalidierungstests ein Raum von (C_m, ε_n)-Paaren untersucht. Wie die Endergebnisse des jeweiligen Trainingsdurchlaufs festgehalten wurden, kann dem nachfolgenden Auszug entnommen werden:

Auszug 2: Exemplarischer Auszug aus einer Protokolldatei

```
1   Starting logging at: 21-09-2016 16-50-02
2   NNR active with strength: 10
3   Cross-validation tests done with:
4   C Epsilon Fold# Time Train Fold Real (prec., recall, avg. prec.)
5   100.000000 0.001000 1 215: 0.981289 0.618206 0.614816
6                         203: 0.927694 0.533673 0.518004
7                         215: 0.974194 0.594488 0.58975
8   100.000000 0.001000 2 215: 0.971664 0.679668 0.674011
9                         213: 0.910326 0.637622 0.620715
10                        213: 0.970646 0.650919 0.642624
```

Dieser Datei ist zu entnehmen, dass mit einer Parametrisierung $C_m = 100$ und $\varepsilon_n = 0{,}001$ bei 2-facher Kreuzvalidierung trainiert wurde (Zeilen 5 und 8 zeigen die gleiche Parametrisierung in zwei Durchläufen).

Bei der Kreuzvalidierung wurden diese auf (unterschiedlichen) Teildatensätzen $\{T_1, \ldots, T_k\} \setminus \{T_i\}$ trainiert (vgl. Abschnitt 2.4.4) und es sind zwei Klassifikatoren dieser Parametrisierung entstanden. Beide wurden danach insgesamt dreimal ausgeführt (siehe Zeilen 5-7 bzw. 8-10). Ein Test lief auf dem Teildatensatz (also ihrer Trainingsmenge), einer auf der ausgelassene Menge T_i und einer auf den eigentlichen Testdaten T_{k+1}, die vorher separiert wurden. Bei einer 2-fachen Kreuzvalidierung entstehen so sechs abschließende Messungen pro Parameterpaar, die durchgeführt und festgehalten wurden.

Dem Auszug 2 kann man außerdem entnehmen, dass pro Messung eine Zeit gespeichert wurde (die Zahl vor dem Doppelpunkt in einer Zeile). Dies ist die Zeit in Millisekunden, die der Detektor im Schnitt gebraucht hat, um ein Bild des jeweiligen Datensatzes vollständig auszuwerten. Die ersten beiden Messzahlen hinter der Zeit sind die Genauigkeit und die Sensitivität des Klassifikators bei der Anwendung auf den Testdaten. Die letzte Messzahl einer Zeile ist die durchschnittliche Genauigkeit.

4 Empirische Analyse

Im vorhergehenden Kapitel wurde die Implementierung von drei Varianten einer auf HOG basierenden Roboterdetektion vorgestellt. Hier folgt die experimentelle Analyse dieser Detektoren.

Um die drei Varianten zu trainieren und zu evaluieren, wurden zwei Bilddatenbanken samt markierten Robotern aufgebaut, die repräsentativ zwei mögliche Anwendungsszenarien für die gelernten Detektoren darstellen würden (siehe 4.1). Eine dieser Datenbanken zeigt die Ansicht eines externen Beobachters auf die Roboter. Die Bilder der zweiten Datenbank wurden dagegen auf den Robotern selber aufgezeichnet. Sie geben die Ich-Perspektive des Roboters wieder. Alle im Folgenden beschriebenen Experimente wurden auf den Bildern dieser beiden Datenbanken durchgeführt.

Für jedes der drei beschriebenen Verfahren wurden Experimente auf den zwei verschiedenen Datensätzen durchgeführt. In den Experimenten wird ein Raum möglicher Parameterpaare, deren Auswahl in 4.2.1 beschrieben ist, nach optimalen Konfigurationen durchsucht. Für jedes Paar wurden die gelernten Klassifikatoren per 2-facher Kreuzvalidierung überprüft.

Jeder Durchgang der Suche beinhaltet dabei den kompletten Lern- und Testvorgang eines HOG-Detektors von der Vorverarbeitung der Eingabebilder. Eine Iteration führt über die Berechnung des HOG-Merkmalsvektors auf den Bildern, dem wiederholten Training der SVM mit den markierten Trainingsbeispielen, so dass diese schließlich den gewählten Parametern genügt, bis hin zum Testen des Detektors auf dafür separierten Teilen der Datenbank.

Um die Qualität der finalen Ergebnisse dieser Parameteroptimierung miteinander vergleichen zu können, wurde für jeden trainierten Detektor die Genauigkeit (engl. precision) und die Sensitivität (engl. recall) bestimmt und aus diesen das F_1-, F_2-, $F_{0,5}$- und G-Maß berechnet (vgl. 4.4) und in Protokolldateien festgehalten.

In Kapitel 5 folgt die vollständige Auswertung der Experimente.

4.1 Aufbau und Vergleich erstellter Datensätze

Um das Potenzial in realen Anwendungen zu messen, muss ein neu entwickelter Detektor zuerst auf Datensätzen validiert werden, die den realen Anwendungsfall abbilden. Für verschiedene Anwendungsfälle, etwa der Erkennung von Fußgängern oder der Detektion und Klassifikation von Vehikeln, stehen bereits umfangreiche Datenbanken an Bildern zur Verfügung. Diese können in Trainingsrunden und Testverfahren mit dem Algorithmus der Wahl durchexerziert und mit der Leistung anderer Techniken auf derselben Datenbank verglichen werden. Für den speziellen Fall der Detektion humanoider Roboter, insbesondere im Rahmen der SPL des RoboCups, existieren solche Datenbanken nicht. Um die Klassifikatoren trainieren und testen zu können, mussten daher entsprechende Datenbanken

zunächst erstellt werden. Als erster Schritt der empirischen Untersuchung wurden daher zwei Datenbanken mit Bildern von Nao Robotern erstellt und die Roboter auf diesen Bildern markiert.

Eine der Datenbanken wurde aufgebaut aus gesammelten Videoaufnahmen des NaoTH. Die einzelnen Videoaufnahmen wurden bei Spielen mit einer auf Höhe der Spielfeldmittellinie angebrachten hochauflösenden Weitwinkelkamera in der Vogelperspektive aufgenommen. Die Kamera[26] produziert Bilder der Größe 1920×1080 Pixel (1080p). Es sind das gesamte Spielfeld, inklusive beider Tore und aller Linien, sowie der Zuschauer- bzw. Teilnehmerbereich gegenüber der Kamera zu sehen. Diese Aufnahmen wurden zum einen ausgewählt, da jederzeit alle am Spiel beteiligten Roboter zu sehen sind. Zum anderen sind die Bilder perspektivisch dem am besten empirisch belegten Anwendungsfall für HOG-Merkmale, der Erfassung von Fußgängern durch Verkehrskameras wie beispielsweise auf den Fußgänger-Datenbanken INRIA und des MIT [40, 41] sehr ähnlich.

Aus einer Vielzahl an Videos wurden dabei repräsentativ solche ausgewählt, die ein möglichst breites Spektrum an Anwendungsfällen für den Klassifikator darstellen. Die Videos unterscheiden sich dabei hinsichtlich der Spielfeldfarbe, der Beleuchtung, des Zuschaueraufkommens und des Vorkommens anderer Störfaktoren, wie etwa der Häufigkeit mit der Schiedsrichter über das Spielfeld laufen und so z.B. Roboter verdecken (vgl. Abbildung 4.1). Es wurden 5 Videos aus drei Veranstaltungen, dem RoboCup 2015, der Make Munich 2016 und der European Open 2016, selektiert. Die Videos sind im Durchschnitt 13 Minuten lang und zeigen eine Halbzeit eines Spiels.

Für den Aufbau der tatsächlichen Trainingsdaten wurden zunächst mit dem Programm FFmpeg[27] die Vor- und Nachbereitungsphasen, also Phasen in denen kein Spiel stattfand, herausgeschnitten. Die Auflösung der Videos wurde dabei in Länge und Breite auf 960×540 Pixel halbiert. Aus dem laufenden Spiel wurde anschließend alle drei Sekunden ein Standbild extrahiert, so dass je Video zwischen 150 und 250 einzelne Bilder in die Datenbank eingegangen sind.

In den insgesamt 1059 einzelnen Aufnahmen wurden dann manuell alle am Spiel teilnehmenden, aktiven Nao Roboter markiert.[28] Das heißt insbesondere umgefallene und aus dem Spiel herausgestellte Roboter wurden als zu ignorieren markiert. Die entstandenen Markierungen wurden in XML-Dateien geschrieben. Insgesamt wurden so 9086 einzigartige Nao Roboter markiert, die im späteren Training als Positivbeispiele dienten. Die markierten Roboter sind zwischen 25 bis 80 Pixel breit und zwischen 40 bis 105 Pixel hoch (siehe Abbildung 4.2).

---

[26] Die Kamera ist eine GoPro HERO, mit 1080p Auflösung bei bis zu 30 Bildern pro Sekunde im 16:9 Format. Weitere Spezifikationen lassen sich dem Handbuch entnehmen, http://cbcdn2.gp-static.com/uploads/product_manual/file/349/UM_HERO_GER_REVA_WEB.pdf, abgerufen am 22.01.2017

[27] Aktuelle Version der Anwendung zu finden unter https://ffmpeg.org/, abgerufen am 22.01.2017

[28] Zum Markieren verwendet wurde die auf dlib basierende Anwendung imagelab, https://github.com/davisking/dlib/tree/master/tools/imglab, abgerufen am 22.01.2017

Abbildung 4.1: *Repräsentative Bilder aus der Datenbank 1. Oben links: Beispielhafte Aufnahme eines Wettbewerbsspiels vom RoboCup 2015, das Spielfeld ist gut und gleichmäßig ausgeleuchtet, zahlreiche Zuschauer sind um das Spielfeld versammelt. Oben rechts: Eine Aufnahme von der Make Munich Messe, die Zuschauer sind hier im Dunkeln kaum zu sehen, das Spielfeld ist von einem großen Spot ungleichmäßig ausgeleuchtet, die Roboter werfen teils tiefe Schatten. Unten links: Aufnahme eines Testspiels beim RoboCup 2015, ideale Bedingungen, mit wenigen Zuschauern, die roten Markierungen zeigen aktive Roboter, die im Training gelernt werden sollen. Ein inaktiver Roboter ganz hinten soll ignoriert werden und ist deshalb durchgestrichen. Unten rechts: Eine Aufnahme von einem Testspiel bei den European Open 2016 in Eindhoven. Das Spielfeld ist umgeben von inaktiven Robotern, die aufgeladen oder programmiert werden. Die Schiedsrichter laufen vor der Kamera umher und tragen Aufdrucke der Nao Roboter auf ihrer Kleidung.*

Abbildung 4.2: *Eine Serie von Positivbeispielen wie sie tatsächlich vom Trainingsalgorithmus verwendet werden. Die Eingabedaten variieren in ihrer Rotation, Pose oder Ausrichtung zur Kamera. Die Nao Roboter sind teilweise verdeckt oder abgeschnitten, stehen vor Linien sowie Torpfosten und die Auflösung und Schärfe hängen vom Abstand zur Kamera ab. Einzig das Seitenverhältnis von Höhe zu Breite ist annähernd konstant.*

Zum Aufbau der zweiten Datenbank wurden verschiedene Protokolldateien direkt auf den Nao Robotern aufgezeichnet. Im Kopf des Roboters sind zwei Kameras vertikal angeordnet, die jeweils mit bis zu 1280×960 Pixeln (720p) auflösen und parallel 30 Bilder pro Sekunde liefern können. Diese Kameras verwendet der Roboter für die visuelle Analyse seiner Umgebung, die aufgenommenen Bilder sind daher aus der subjektiven Kameraperspektive (Ich-Perspektive) des Roboters. Für die Aufnahmen wurde im Labor des Nao Team Humboldt eine Reihe von zwölf progressiv komplexeren Spielsituationen aufgebaut und in Protokolldateien im lokalen Speicher eines Roboters aufgezeichnet und anschließend extrahiert. Ermöglicht wurde dies durch die Verwendung des NaoTH-Frameworks (vgl. Abschnitt 1.2), insbesondere durch Skripte die der automatisierten Extraktion der von den Robotern generierten Daten dienen.

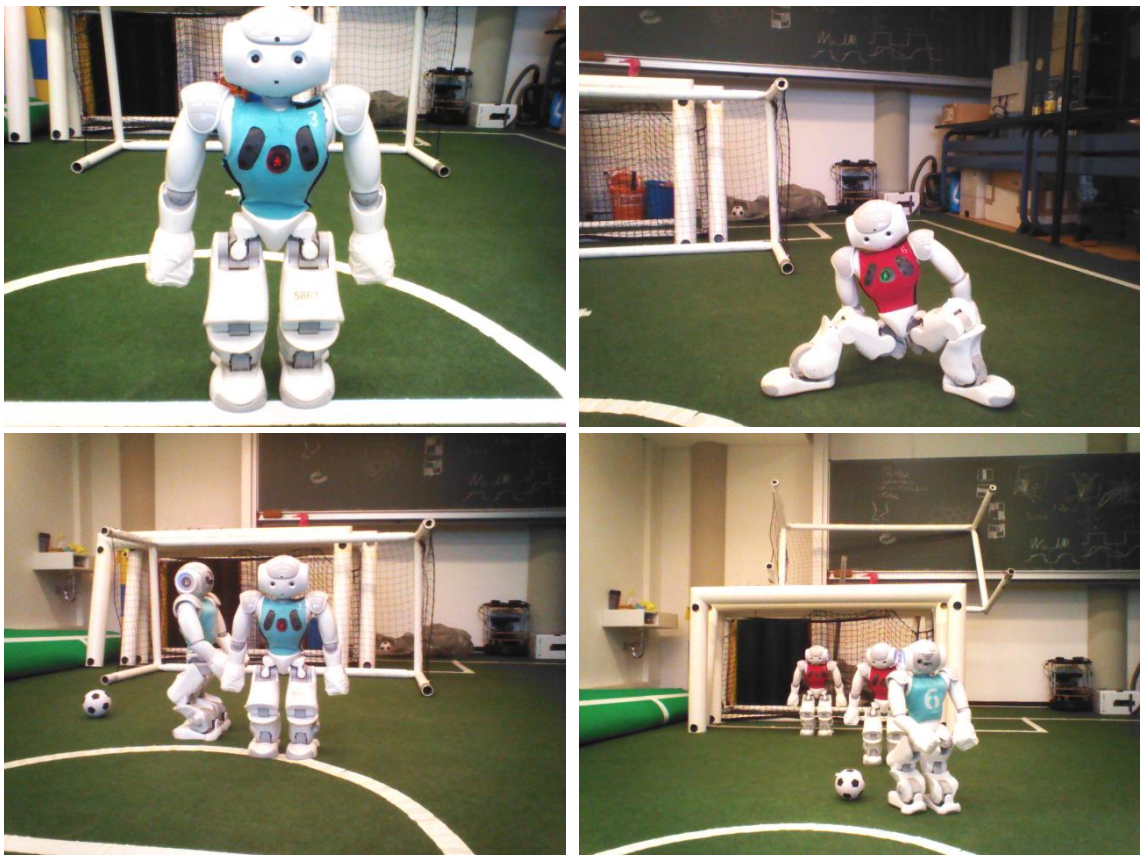


Abbildung 4.3: *Vier exemplarische Bilder aus den verschiedenen Videoaufnahmen der zweiten Datenbank aus der Ich-Perspektive des Nao Roboters. Von oben links nach unten rechts nimmt in Leserichtung die Entfernung von der Kamera, so wie die Anzahl der potentiell zu erkennenden Objekte und der am Spiel teilnehmenden Roboter zu. Das Spielfeld und die Beleuchtung der Bilder sind zwischen den Videos unverändert.*

Aus den zunächst über 2000 extrahierten Bildern wurden repräsentativ 857 Bilder der oberen Kamera ausgewählt und in diesen, wie bereits bei der ersten Datenbank beschrieben, 1578 aktive Roboter als zu erkennende Positivbeispiele markiert. Bei der Auswahl wurde darauf Wert gelegt, dass zwischen aufeinander folgenden Aufnahmen eine Bewegung stattfand. Die markierten Roboter sind zwischen 30cm und 200cm von der aufnehmenden Kamera

entfernt und im Bild zwischen 50 und 300 Pixel breit und zwischen 100 und 450 Pixel hoch (vgl. Abbildung 4.4). Im Unterschied zu den Videos der ersten Datenbank sind die Videoaufnahmen homogener in Bezug auf die Beleuchtung und die Spielfeldfarbe, da alle Aufnahmen am selben Ort und unter kontrollierten Bedingungen entstanden sind. Des Weiteren wurde darauf verzichtet die Aufnahmen der beiden Kameras in der Nachbearbeitung zusammenzufügen, da die Spielsituationen so konstruiert wurden, dass nahezu keine Roboterteile auf den Bildern der unteren Kamera vorzufinden sind.

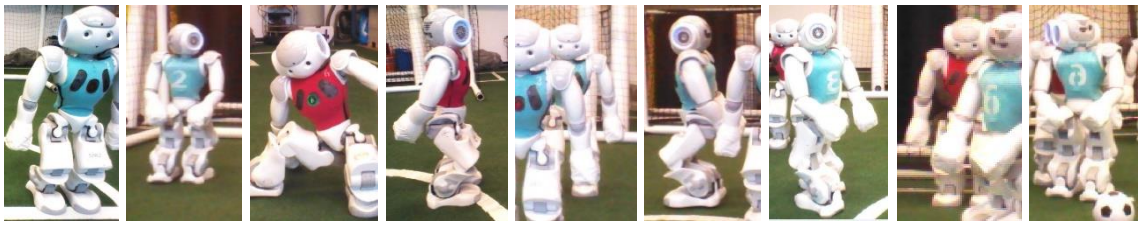


Abbildung 4.4: *Eine Reihe der markierten Positivbeispiele aus der zweiten Datenbank. Die Roboter sind in typischen Posen aus vielen Rotationen um die eigene Längsachse zu sehen. Teilweise verdecken sich die Roboter gegenseitig oder sind abgeschnitten, wenn sie sich nahe der Kamera befinden.*

4.2 Modellauswahl

Beim maschinellen Lernen lässt sich die Modellauswahl von gewöhnlichen Optimierungsproblemen unterscheiden, die oft mit der Suche nach den besten Parametern für ein Problem, z.B. beschrieben durch eine Kostenfunktion, gleichzusetzen sind. Beim Training der SVM auf den HOG-Merkmalen liegt so ein Optimierungsproblem in Form der Maximierung des kleinsten Abstands der Trainingsdaten zur Hyperebene bzw. beim Durchführen des MMOD-Algorithmus (vgl. Abschnitt 3.2.4) vor. Es kann dabei auch von der Anpassung der Modellparameter an die Trainingsdaten gesprochen werden. Darüber hinaus ist für den Erfolg des Trainings eine Anzahl von Parametern relevant, die abstrakt betrachtet auf einer höheren Ebene liegen, nämlich bei der Konfiguration der Eigenschaften des Modells selbst. Zu diesen Eigenschaften zählen etwa wie schnell und mit welcher Komplexität das Modell lernt, wie drastisch Fehler zu einer Anpassung führen oder in welchen Parameterräumen überhaupt nach möglichen Lösungen für das Optimierungsproblem gesucht werden kann.

4.2.1 Parameterauswahl

Relevante Parameter, die im Zuge der Modellauswahl für die Detektion humanoider Roboter mittels HOG-Merkmalen begutachtet werden können, sind unter anderem die den MMOD-Algorithmus bestimmenden Parameter $C, \varepsilon, \varepsilon_{\mathrm{qp}}, L_{fa}$ und L_{miss} [35]. Neben dem bereits in Abschnitt 2.4.3 beschriebenen Parameter C, also der Gewichtung des SVM-Fehlerterms, bietet sich insbesondere die Suche nach einer geeigneten Trainingsabbruchbedingung an. Beim MMOD-Algorithmus entspricht dies ε, dem maximalen Fehlerterm gegen den ggf.

mehrere Iterationsschritte geprüft wird. L_{miss} und L_{fa}, die die relative Wichtigkeit des Erreichens einer hohen Sensitivität bzw. einer hohen Genauigkeit steuern, bieten sich dagegen weniger an, um in erster Instanz das Verfahren auszuwählen. Stattdessen könnten Klassifikatoren mit stark abweichenden Bewertungen in ihren verschiedenen Qualitätsmaßen hinsichtlich dieser Parameter überprüft werden. Weitere Modelle die verglichen werden können, finden sich in der Auswahl der Kernelfunktion der SVM. Die SVM kann einen linearen Kernel haben oder eine RBF oder sie hat eine komplexere polynomiale Basisfunktion. Weiterhin kann geprüft werden, ob es vorteilhaft ist, den zu lernenden Klassifikator einer Regularisierung zu unterziehen.

Wenn man die aufgeführten Möglichkeiten zur Modellauswahl betrachtet, kann man zu der Erkenntnis kommen, dass einige davon das Training auf ganz konzeptioneller Ebene verändern, wie die Regularisierung oder die Verwendung einer nichtlinearen Basisfunktion. Diese können vor allem dann gezielt optimiert werden, wenn ein neues oder besseres Verständnis über die allgemeine Beschaffenheit der Eingabedaten oder über die gelernten Klassifikatoren gewonnen wurde.

Dem gegenüber stehen jene Modellparameter, die sich systematisch hinsichtlich ihres Einflusses auf den Lernerfolg prüfen lassen, wie etwa die Parameter des MMOD-Algorithmus. Eine Art diese Modellparameter gezielt nach optimalen Konfigurationen zu durchsuchen ist die Rastersuche. Die Rastersuche ist eine erschöpfende Suche in einer manuell festgelegten Teilmenge des gesamten Parameterraums. Die zu optimierenden Parameter spannen bei der Rastersuche einen Raum auf, welcher entlang der Parameterachsen in gleichmäßigen Abständen geteilt wird. Die Schnittpunkte der entstehenden Rasterlinien definieren die Abtastpunkte auf der Oberfläche der Zielfunktion. Das Ziel ist dabei zum einen die Betrachtung der Verfahren in Abhängigkeit von verschiedenen Parametrisierungen. Zum anderen dient die Suche dem Auffinden des globalen Optimums innerhalb des abgesteckten Raumes bzw. in Abhängigkeit von der Engmaschigkeit des Rasters findet eine Annäherung an das Optimum statt.

Eine Voraussetzung dafür ist, dass die Zielfunktion in weiten Bereichen monoton ist und die Optima sich aus weiten Tälern hervorheben. Auf die untersuchten Gütemaße der Klassifikatoren in Abhängigkeit von den aufspannenden Modellparametern C und ε trifft dies zu. Ein weiterer Schritt, der bei der Rastersuche zu beachten ist, ist die sinnvolle Auswahl des Wertebereichs der Parameter. Ein gefundenes Optimum sollte nicht am Rande des Rasters liegen. Je nach benötigtem Rechenaufwand zur Messung der Zielfunktion kann es sich anbieten, diese nicht in gleichmäßigen Abständen abzutasten, sondern unter Berücksichtigung des Einflusses der Parameter auf das Modell. In ihrer Arbeit zur praktischen Verwendung von SVM Klassifikatoren [15] schlagen Hsu et al. deshalb eine exponentiell wachsende Sequenz für die Modellparameter vor. Weiterhin wird beschrieben, dass komplexere Heuristiken als die erschöpfende Rastersuche bei nur zwei zu optimierenden Parametern verhältnismäßig wenig Rechenaufwand sparen und gleichzeitig schlechter parallelisiert werden können als die Rastersuche, bei der sich jedes Ergebnis

unabhängig von einer anderen Iteration berechnen lässt. Wenn die erschöpfende Suche dennoch zu viel Zeit in Anspruch nimmt, empfehlen Hsu et al., erst in einem groben Raster nach vielversprechenden Regionen zu suchen und in diesen dann eine neue Suche mit einem dichteren Raster durchzuführen.

4.3 Durchgeführte Experimente

Für beide Datenbanken (vgl. Abschnitt 4.1) wurden die drei implementierten Verfahren untersucht. Ein Verfahren, das Roboter basierend auf den fHOG-Merkmalen von Felzenszwalb detektiert, eines bei dem die gelernte Klassifikatorkaskade reduziert wird und eines bei dem im Zuge des Trainings des fHOG-Klassifikators die SVM einer NNR unterzogen wird.

Für die durchgeführten Experimente wurden in einer erschöpfenden Rastersuche die Werte $C = 10^{-6}, 10^{-5}, \dots, 10^{8}$ und $\varepsilon = 10^{-8}, 10^{-7}, \dots, 10^{6}$ durchlaufen, um das Verhalten der Klassifikatoren in diesem Parameterraum zu untersuchen und die besten (C_m, ε_n)-Paare zu bestimmen. Für jedes Paar in dieser Menge H der zu optimierenden Parameter wurde dabei eine 2-fache Kreuzvalidierung durchgeführt. Für Datensatz 1 wurde eine optimale Erkennungsfenstergröße von 45×64 ($Breite \times Höhe$) Pixeln berechnet und verwendet. Für Datensatz 2 wurde die optimale Erkennungsfenstergröße 40×72 Pixel ermittelt. Die HOG-Merkmale wurden auf 5×7 (Datenbank 1) und 4×8 (Datenbank 2) Blöcken von 2×2 Zellen mit 5×5 Pixeln berechnet, um den optimalen Erkennungsfenstern zu entsprechen. In allen Experimenten ist die Ratio $\frac{5}{6}$, mit der eine Oktave in der Bildpyramide (vgl. Abschnitt 3.2.1) herunterskaliert wird. Die Histogramme sind in 9 Klassen unterteilt. Der resultierende HOG-Merkmalsvektor für Datenbank 1 ist $5 \times 7 \times 2 \times 2 \times 9 = 1260$ Einträge groß. Der HOG-Merkmalsvektor von Datenbank 2 ist 1152-dimensional.

Zur Durchführung eines Experiments wurde der vorgestellte Prozess zum Training eines Klassifikators (vgl. Abschnitt 3.3.2) in den Rastersuchlauf eingebettet. Das umgesetzte Kreuzvalidierungsverfahren erweitert den bisherigen Trainingsablauf um die notwendigen Prozessschritte, damit ein Trainingsdatensatz automatisch k-fach durchlaufen wird und die Gesamtfehlerquote korrekt aus den Einzelfehlerquoten bzw. Messwerten bestimmt werden kann (siehe Abbildung 4.5).

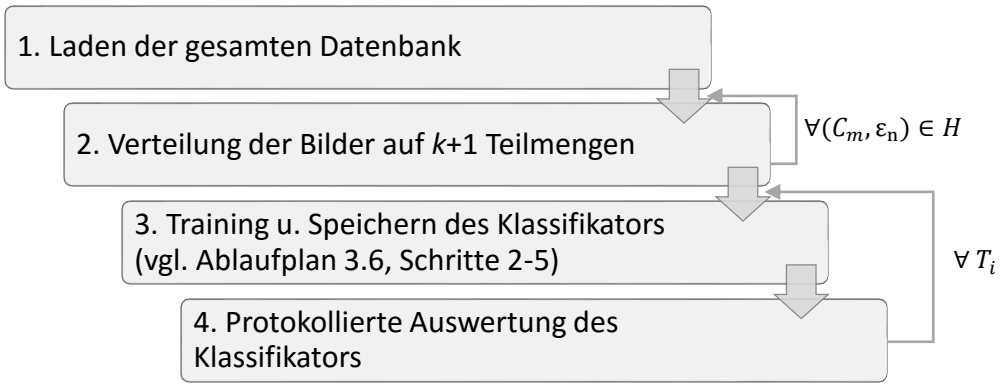


Abbildung 4.5: *Ablaufplan eines Experiments. In einer Rastersuche werden die Parameter durchlaufen, Ergebnisklassifikatoren werden per Kreuzvalidierung bestimmt und getestet.*

Die erste Erweiterung besteht darin, statt eines einzelnen Videos einen Verzeichnispfad von XML-Dateien angeben zu können. Alle in diesem Verzeichnispfad gefundenen Videos samt Markierungen werden als Trainingsbeispiele geladen (vgl. Abschnitt 3.3.3). Bei der Modellauswahl soll ein Klassifikator nicht mehr auf nur einem Video trainiert werden, sondern es wird das auf der gesamten Datenbank am besten funktionierende Modell gesucht.

Nachdem alle Bilder geladen wurden, werden diese zufällig auf die Teilmengen $T_1, \ldots, T_k$ und die Testmenge T_{k+1} verteilt. Alle Teilmengen sind balanciert bezüglich der Anzahl ihrer Elemente. Wenn sich die Zahl der Elemente nicht ohne Rest auf die $k+1$ Teilmengen verteilen lässt, werden zufällig Elemente verworfen, bis dies möglich ist. Darüber hinaus wird versucht, die Elemente der verschiedenen Quellvideos gemäß der originalen Häufigkeitsverteilung in der Gesamtmenge auf die Teilmengen zu verteilen. Das heißt, bei den durchgeführten Experimenten ist jedes Quellvideo in jeder Teilmenge anteilig vorhanden und zwar in etwa im selben Maße wie der Anteil an den gesamten Trainingsdaten war. Es fand eine stratifizierte k-fache Kreuzvalidierung statt.

Der Algorithmus 2 verweist in Zeile 10 auf das in den in den Arbeiten [35, 38] beschriebene Schnittebenenverfahren, das im MMOD-Algorithmus zum Lösen des SVM-Optimierungsproblems verwendet wird. Die Implementierung in der eingesetzten Bibliothek ist dabei eine getreue Umsetzung des in [38] beschriebenen Algorithmus 3, genannt: Training strukturierter SVMs (mit wiederholter Randskalierung) mittels der 1-Lockerung-Formulierung.

Algorithmus 2: k-faches Kreuzvalidierungsverfahren

```
    Eingabe: H, k, Verzeichnis
    Ausgabe: K
1   T := {}, K := {}
2   foreach Datei in Verzeichnis do
3     extrahiere Trainingsdaten aus Datei nach T_temp
4     T := T ∪ T_temp
5   end foreach
6   balanciertes Verteilen von T auf k + 1 Teilmengen T_i
7   foreach (C_m, ε_n) ∈ H do
8     for i = 1 to k do
9       präpariere Trainingsdaten in {T_1,...,T_k} \ {T_i} für SVM Training
10      trainiere Klassifikator K_m,n,i nach MMOD-Algorithmus auf
        {T_1,...,T_k} \ {T_i}
11      Teste K_m,n,i auf {T_1,...,T_k} \ {T_i} und speichere Messdaten
12      Teste K_m,n,i auf T_i und speichere Messdaten
13      Teste K_m,n,i auf T_k+1 und speichere Messdaten
14      K := K ∪ K_m,n,i
15    end for
16  end foreach
```

Das Ergebnis des Kreuzvalidierungsverfahrens ist, wie dem Algorithmus 2 zu entnehmen, eine Menge von Klassifikatoren K bestehend aus dem für jedes Parameterpaar (C_m, ε_n) und auf der jeweiligen Trainingsmenge $\{T_1, \ldots, T_k\} \setminus \{T_i\}$ gelernten Klassifikator $K_{m,n,i}$ und es gilt $|K| = k \cdot |H|$. Außerdem werden die Genauigkeit und die Sensitivität des Klassifikators sowie die Gesamtgeschwindigkeit und Geschwindigkeit pro Bild auf der Trainingsmenge $\{T_1, \ldots, T_k\} \setminus \{T_i\}$, (Zeile 11) auf der ausgelassenen Menge T_i (Zeile 12) und den gesondert separierten Testdaten T_{k+1} (Zeile 13) gemessen.

Alle diese Messungen werden dann pro Klassifikator in einer Protokolldatei festgehalten (vgl. 3.3.6). Auf den Trainingsdaten, auf denen der Klassifikator trainiert wurde, schneidet ein Detektor erwartungsgemäß am besten ab. Eine Voreingenommenheit bei diesen Daten ist gegeben und das Protokollieren der Ergebnisse auf den Trainingsdaten dient ausschließlich der Validierung des Trainings. Bei der späteren Auswertung der Daten liefern diese Messungen keine gewichtige Aussage. Ähnlich verhält es sich mit dem Satz Daten der vor dem Training im jeweiligen Durchlauf i ausgelassene Teilmenge T_i. In der Auswertung sind ausschließlich die Testdaten, die vor dem Beginn der Kreuzvalidierung komplett separiert wurden und die für alle Detektoren damit gleich und vollkommen unbekannt sind, aussagekräftig. Die erzielten Ergebnisse auf den Testdaten liefern praktisch relevante Gütemaße, die als Schätzung der zu erwartenden Leistung eines Klassifikators beim Einsatz unter nicht streng kontrollierten Bedingungen zu verstehen sind.

Wenn möglich, wurde ein Satz von Klassifikatoren im Stapel ausgewertet. In diesem Kontext heißt das, die HOG-Merkmalsvektoren der zu testenden Bilder wurde nur einmal für jedes

Bild berechnet und die Detektoren wurden dann der Reihe nach alle auf diese Merkmalsvektoren angewandt und ausgewertet. Diese Vorgehensweise ist eine erhebliche Zeitersparnis gegenüber der Methode für jeden Klassifikator jeweils das gesamte HOG-Bild neu zu berechnen.

4.4 Untersuchte Qualitätsmaße

Wichtige Mittel zur Evaluierung der in den durchgeführten Experimenten trainierten Klassifikatoren sind die Laufzeit und die Qualitätsmaße. Nach abgeschlossenem Training werden die Klassifikatoren daher auf den Datenbanken ausgeführt und die Messergebnisse in Form von Protokolldateien festgehalten. Aus diesen Messwerten werden kombinierte Maße berechnet, die in der Auswertung eine Beurteilung des Klassifikators zulassen.

Im letzten Prozessschritt der Experimente wurde ein trainierter oder geladener Klassifikator auf einen Satz von Bildern mit markierten Positivbeispielen angewandt und mithilfe der in Abschnitt 3.2.1 beschriebenen Auswertungsfunktion wurden die Genauigkeit p (auch Präzision) und die Sensitivität r (auch Trefferquote) des Klassifikators bestimmt.

Seien t_p die Anzahl der richtig positiven Detektionen, f_p die Anzahl der falsch positiven Detektionen und f_n die Anzahl der falsch negativen Detektionen, dann sind p und r bestimmt durch:

$$p(t_p, f_p) = \frac{t_p}{t_p + f_p} \quad (4.1) \quad \text{und} \quad r(t_p, f_n) = \frac{t_p}{t_p + f_n} \quad (4.2).$$

Zusätzlich wurde die Zeit gemessen, die der Klassifikator benötigt, um den Datensatz bzw. ein Bild aus dem Datensatz zu analysieren. Hervorzuheben ist, dass für jedes Parameterpaar k-viele Klassifikatoren validiert wurden. Die Messwerte für ein Paar sind die gemittelten Messwerte zwischen diesen Klassifikatoren. Die Qualitätsmaße eines Paares sind die gemittelten Qualitätsmaße der einzelnen Klassifikatoren.

4.4.1 Definition der Qualitätsmaße

Aus der Genauigkeit und der Sensitivität wurden für jede Messung verschiedene F_β-Maße und das G-Maß bestimmt. Weil verschiedene Gütekriterien miteinander korrelieren, gehört die Berechnung solcher kombinierten Gütemaße zu den bevorzugten Werkzeugen, um eine Beurteilung der Güte des Klassifikators mit einer einzigen Kennzahl zu erlauben. Das F_β-Maß verbindet dabei Genauigkeit p und Sensitivität r als gewichtetes harmonisches Mittel.

$$F_\beta = (1 + \beta^2) \cdot \frac{p \cdot r}{(\beta^2 \cdot p) + r} \quad (4.3)$$

Als natürlichster Spezialfall berechnet sich das F_1-Maß (oder auch nur F-Maß genannt) als harmonisches Mittel aus Genauigkeit und Sensitivität zu:

$$F_1 = 2 \cdot \frac{p \cdot r}{p + r} \tag{4.4}$$

Für F_1 sind Genauigkeit und Sensitivität gleich stark gewichtet. Soll die Genauigkeit stärker gewichtet werden, kann das $F_{0,5}$-Maß bemüht werden. Soll dagegen mehr Wert auf die Sensitivität gelegt werden, kann das F_2-Maß verwendet werden. Aus Formel (*4.3*) ergibt sich $F_2 = 5 \cdot \frac{p \cdot r}{4 \cdot p + r}$ (4.5), bzw. $F_{0,5} = 1{,}25 \cdot \frac{p \cdot r}{0{,}25 \cdot p + r}$ (4.6) und somit, dass jeweils eine der Kenngrößen viermal stärker gewichtet wird als die andere.

Im Unterschied zum F-Maß stellt das G-Maß das geometrische Mittel aus Genauigkeit und Sensitivität dar und berechnet sich als:

$$G = \sqrt{p \cdot r} \tag{4.7}$$

Die vierte der protokollierten Messzahlen ist die durchschnittliche Genauigkeit. Dieser Wert lässt sich anhand der PR-Kurve ermitteln. Die PR-Kurve trägt die Genauigkeit in Abhängigkeit von der Sensitivität ab. Sei $P(r)$ diese Funktion, dann ist die durchschnittliche Genauigkeit p_{avg} definiert als der Durchschnitt der Werte die $P(r)$ über das Intervall $r = 0$ bis $r = 1$ annimmt, also genau als:

$$p_{avg} = \int_0^1 P(r)\, dr \tag{4.8}$$

In der Praxis wird das Integral häufig über den Mittelwert einer Reihe von Messungen approximiert. Um den Wert ermitteln zu können, wird die Funktionalität benötigt, die Sensitivität r_i beliebig zu verschieben, um dann einen Messwert der Genauigkeit $P(r_i)$ zu bestimmen. Es ist möglich r auf dem ganzen Intervall zu verschieben, indem die akzeptierte Bewertung eines Objektes durch die Klassifikatorkaskade verschoben wird. Aus den Einzelbewertungen der Klassifikatorkaskade geht als gewichtete Summe die Gesamtbewertung eines geprüften Datenpunktes durch die Klassifikatorkaskade hervor. Diese Summe entscheidet letztlich darüber wie ein Datenpunkt durch den binären Klassifikator klassifiziert wird. Die Gesamtbewertung wird dabei mit einem Schwellwert S_k verglichen. Liegt die Bewertung über diesem Schwellwert, wird der Datenpunkt als Positivbeispiel klassifiziert, liegt die Bewertung darunter, ist er ein Negativbeispiel. Wenn nicht explizit angegeben, kann $S_k = 0$ angenommen werden. Beim Training der Detektoren und bei der Modellauswahl war dies der Fall. Das protokollierte (p, r)-Paar gilt für $S_k = 0$ und ist damit wiederrum eigentlich nur eine Messung aller möglichen Ausprägungen des jeweiligen Klassifikators. Die durchschnittliche Genauigkeit p_{avg} des Klassifikators ist dagegen ein unabhängiges Qualitätsmaß des Klassifikators und wurde daher zusätzlich ermittelt und ausgewertet. Unabhängig von der Approximation von p_{avg}, die für jeden Klassifikator und jede Teilmenge vorgenommen wurde, wurde zusätzlich für die nach den Qualitätsmaßen besten Klassifikatoren die vollständige PR-Kurve approximiert und dargestellt, wie den Abschnitten des folgenden Kapitels entnommen werden kann.

5 Auswertung

In den vorherigen Kapiteln wurden mehrere Verfahren zur visuellen Detektion humanoider Roboter vorgestellt. Drei dieser Verfahren wurden implementiert und damit Klassifikatoren eines ausgewählten Parameterraumes auf zwei Datenbanken trainiert. Die auf den Klassifikatoren basierenden Detektoren wurden auf den Datensätzen ausgeführt, kreuzvalidiert und so experimentell diverse qualitative Messwerte der Detektoren ermittelt.

In diesem Kapitel werden die kombinierten F_1-, F_2-, $F_{0,5}$- und G-Maße, sowie die durchschnittliche Präzision und der Zeitaufwand der Detektion jedes einzelnen Verfahrens auf den beiden Datenbanken evaluiert. In Heatmaps (aus dem Englischen, Farbdiagramm) werden die Maße zum besseren Vergleich visualisiert.

Für eine Auswahl der in den verschiedenen Gütekriterien am besten abschneidenden Klassifikatoren wurden anschließend vollständige PR-Kurven (aus dem Englischen, precision-recall curve) in weiteren Experimenten ermittelt. In der Gegenüberstellung der PR-Kurven der verschiedenen Implementierungen wird ein wertendes Fazit zwischen den Varianten gezogen.

Abschließend werden die Vor- und Nachteile, die möglichen Einsatzgebiete und die Problemfelder der verschiedenen Detektoren besprochen.

5.1 Roboterdetektion basierend auf fHOG

In den ersten beiden Experimenten wurde der Algorithmus nach Felzenszwalb [5] auf den Datensätzen 1 (Vogelperspektive) und 2 (Ich-Perspektive) ausgeführt. In beiden Experimenten wurde zur Hyperparameteroptimierung der trainierten SVM eine erschöpfende Rastersuche der Werte $C = 10^{-6}, 10^{-5}, \dots, 10^{8}$ und $\varepsilon = 10^{-8}, 10^{-7}, \dots, 10^{6}$ durchgeführt.

Die untersuchten Wertebereiche wurden sehr großzügig gewählt, so dass an den Rändern für Werte von $C < 10^{-3}$ und $\varepsilon > 10^{3}$ keine brauchbaren Klassifikatoren trainiert wurden. Im ersten Fall ist die Gewichtung des Fehlerterms so gering, dass keinerlei Anpassung an die Trainingsbeispiele geschieht. Im zweiten Fall ist die akzeptierte Abbruchbedingung so hoch, dass der Algorithmus bereits nach der ersten Trainingsiteration abbricht. Die resultierenden Klassifikatoren sind dort hinsichtlich Präzision und Sensitivität annähernd 0. Beispiele für diese Klassifikatoren sind die in Abbildung 3.7 dargestellten uniformen Visualisierungen.

Für weniger extreme Werte lassen sich in den Messdaten, die hier exemplarisch für die $F1$- und p_{avg}-Maße in Heatmaps dargestellt wurden (siehe Abbildung 5.1), deutlich die am besten abschneidenden Wertebereiche ausmachen. Die dargestellten Messungen wurden auf den separierten Testdaten T_{k+1} gemacht, die den realen Einsatz auf unbekannten Eingaben repräsentieren.

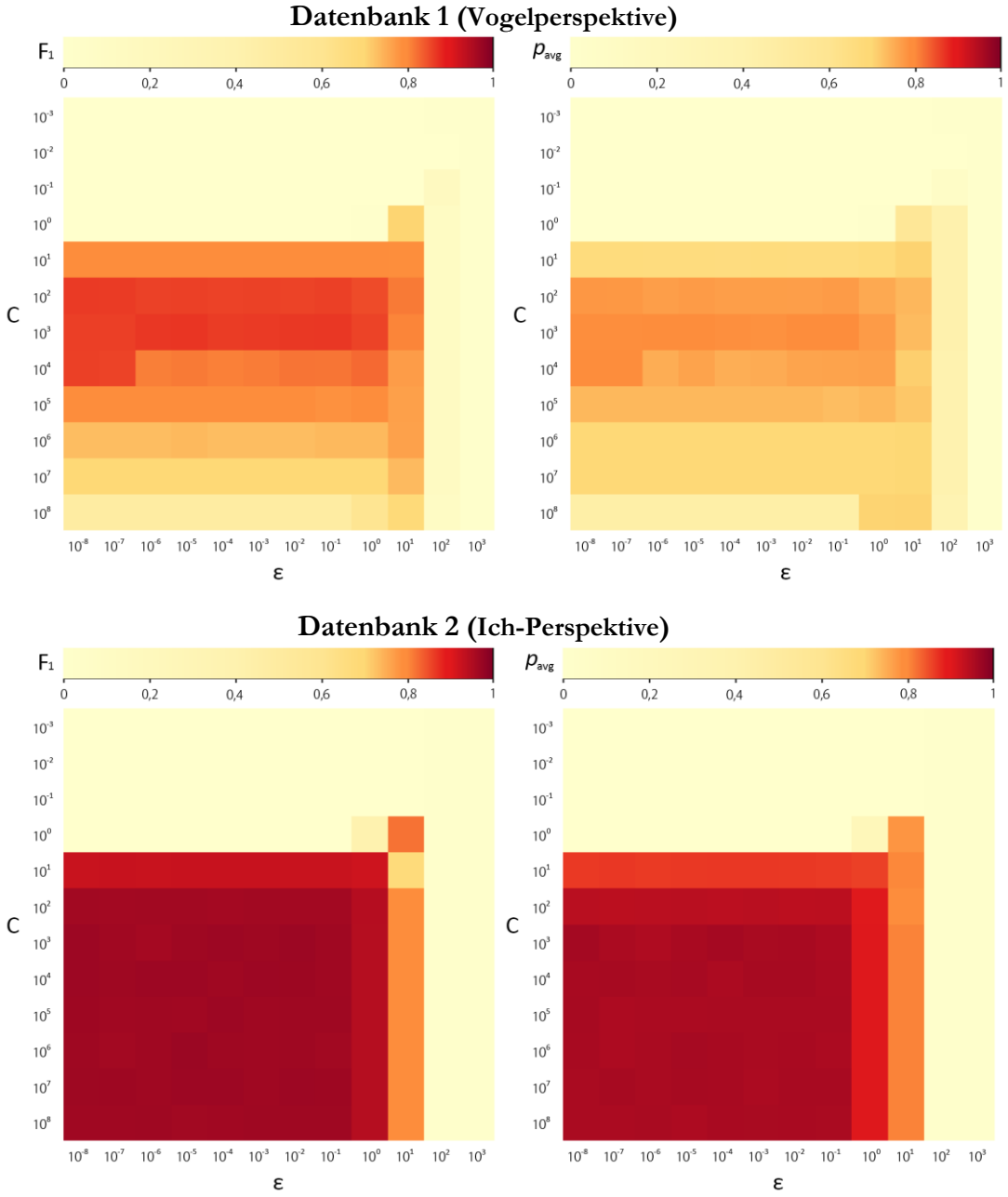


***Abbildung 5.1**: Dargestellt sind Heatmaps gemessener Maße der fHOG-Detektoren. Oben links: F_1-Maße auf der Datenbank 1. Oben rechts: Durchschnittliche Präzisionen p_{avg} auf der Datenbank 1. Unten links: F_1-Maße auf der Datenbank 2. Unten rechts: Durchschnittlichen Präzisionen p_{avg} auf der Datenbank 2. In der Gegenüberstellung von links und rechts ist zu erkennen, dass die F_1-Maße im Schnitt höher ausfallen als die durchschnittliche Präzision, die über die gesamte PR-Kurve bestimmt wird und die für Sensitivitätswerte gegen 1 zumeist 0 annimmt. Dennoch ist zu erkennen, dass die jeweiligen Maxima in den selben Bereichen für (C, ε) auftreten. Im Vergleich zwischen Datenbank 1 und 2, fällt auf, dass auf der Datenbank 2 breitflächiger höhere Maxima erreicht werden. Die tiefdunklen Flächen, in denen Werte von annähernd 1 erreicht werden, sind nur in den unteren Heatmaps zu finden.*

Auf Datenbank 1 ist die die Maximal erreichte Präzision eines fHOG-Detektors 98,54% und die maximal erreichte Sensitivität ist 84,09%. Die Spitzenwerte in den vereinigten

Qualitätsmaßen und damit die auf fHOG-basierende Detektion sind im Anwendungsfall gut, die Werte können der folgenden Tabelle entnommen werden (vgl. außerdem Abbildung 5.2, Tabelle 5 und Anlage A.4):

Tabelle 1: *Erreichte Maxima bei fHOG*

Erreichte Maxima bei fHOG						
	Datenbank 1 (Vogelperspektive)			**Datenbank 2 (Ich-Perspektive)**		
Qualitätsmaß	C	ε	**Maximalwert**	C	ε	**Maximalwert**
F_1	10^3	10^{-5}	0,8681	10^6	10^{-5}	0,9720
F_2	10^4	10^{-8}	0,8301	10^3	10^{-8}	0,9676
$F_{0,5}$	10^2	10^{-8}	0,9199	10^2	10^{-2}	0,9727
G	10^3	10^{-5}	0,8709	10^6	10^{-5}	0,9720
p_{avg}	10^4	10^{-8}	0,7967	10^3	10^{-8}	0,9627

In jedem Qualitätsmaß gibt es mehrere Wertepaare, die bis auf einen Bruchteil die jeweiligen Maxima erreichen, so dass ein höher aufgelöstes Raster durchaus noch neue Maxima hervorbringen kann. Die Maße bewegen sich allerdings in der Spitze nur noch in der dritten oder vierten Nachkommastelle. Auf den Trainingsdaten wurden erwartungsgemäß bessere Ergebnisse erzielt. Die Schätzungen auf den Trainingsdaten sind einige Prozente zu optimistisch, im F_1-Maß werden Werte von $F_1 > 0{,}94$ erreicht. Im Mittel brauchen die Detektoren (solche mit $F_1 < 0{,}01$ ausgenommen) 220 ms um ein gesamtes Bild der Größe 1920×1080 Pixel auszuwerten. Die schnellsten Detektoren schaffen dies in 214 ms, die langsamsten in 240 ms. Die Verarbeitungszeit hängt dabei vom eingesetzten Testcomputer ab, dessen Spezifikationen der Anlage A.3 entnommen werden können.

Auf der Datenbank 2 werden im Vergleich noch einmal deutlich bessere Messergebnisse erzielt. Die höchste gemessene Präzision auf den Testdaten ist $99{,}55\%$, die höchste gemessene Sensitivität ist $96{,}68\%$. Die 1280×960 großen Testbilder werden im Schnitt in 129,5 ms ausgewertet, wobei die Werte um 1-2 ms zwischen den Detektoren schwanken.

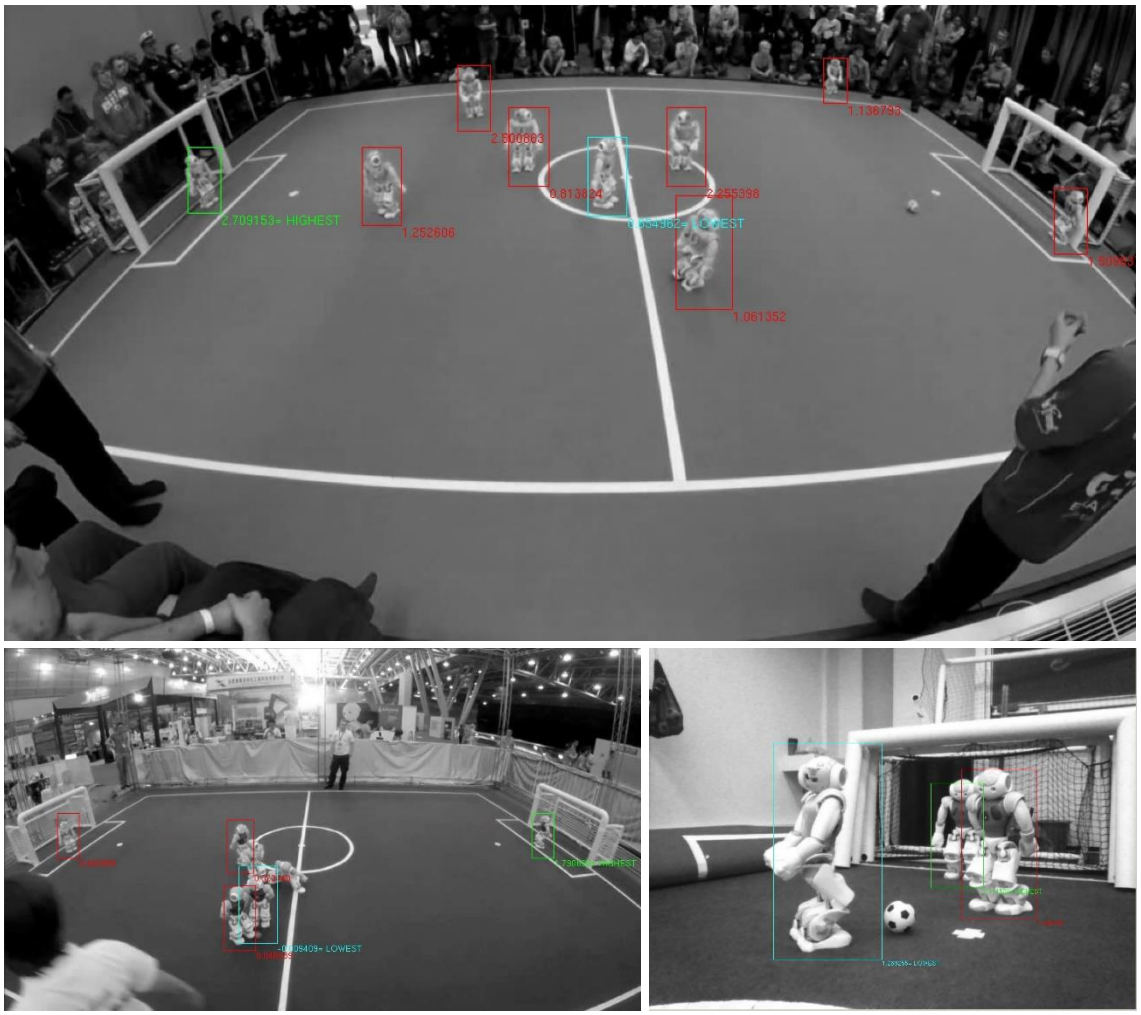


***Abbildung 5.2**: Auswahl an mittels fHOG-Verfahren klassifizierten Bildern. Verwendet wurde jeweils der Klassifikator, der am besten im F_1-Maß abgeschnitten hat. Oben: Zu sehen ist, dass die Detektion zuverlässig funktioniert, wenn die Roboter relativ frei und unverdeckt auf dem Spielfeld sichtbar sind. Die inaktiven Roboter ganz links werden korrekt nicht detektiert, der Roboter, der hinten zwischen den Kindern steht, dagegen schon. Unten links: Wenn sich mehrere Roboter zu großen Teilen verdecken, kann es vorkommen, dass ein Roboter nicht detektiert wird, allerdings bleibt ein Roboter nur selten über mehrere Sekunden hinweg unerkannt. Unten rechts: Auf der Datenbank aus der Ich-Perspektive eines Roboters funktioniert die Detektion auch bei verdeckten Robotern sehr zuverlässig. Der Torwart weist hier sogar die höchste Konfidenz auf.*

Nach der Berechnung der Qualitätsmaß-Maxima des Verfahrens, wurden zur Ermittlung der vollständigen PR-Kurve die (gespeicherten) Klassifikatoren der jeweiligen Maxima noch einmal auf den Testdaten der Datenbank 1, respektive Datenbank 2, ausgeführt. Es wurde jedoch iterativ der Schwellwert S_k, der über die Einordnung als Positiverkennung des Klassifikators bestimmt, von -5 auf 5 in 0,05er-Schritten erhöht. Mit jedem dieser S_{k_i} wurde für den Klassifikator ein (p, r)-Paar gemessen. Die ermittelten PR-Kurven finden sich am Ende des Kapitels in den Abbildungen 5.5 und 5.7. In diesen Abbildungen werden die besten Klassifikatoren der verschiedenen Varianten in Form ihrer PR-Kurven einander gegenübergestellt.

5.2 Roboterdetektion durch reduzierten fHOG

In den folgenden vier Experimenten wurde der vorgestellte Algorithmus zur Reduktion der Kaskaden (rfHOG, vgl. Abschnitt 3.1.1) auf alle in Abschnitt 5.1 gewonnenen Klassifikatoren angewandt. Die beiden Mengen der auf Datensatz 1 (Vogelperspektive) und Datensatz 2 (Ich-Perspektive) trainierten Klassifikatoren wurden dabei einmal reduziert mit der Parametrisierung $s_1 = 0{,}6$ und $c_1 = 0{,}9$ und ein anderes Mal mit $s_2 = 0{,}4$ und $c_2 = 0{,}5$. Ziel dieser Versuche war es, ein Verständnis dafür zu entwickeln, wie sich die Filterkaskade, sowie die Zeit und Qualität der Detektion verändert, wenn die Klassifikatoren einmal stark mit (s_1, c_1) und einmal weniger stark mit (s_2, c_2) reduziert werden.

Für (s_1, c_1) auf Datensatz 1 wurde die Kaskade von im Schnitt 214 Filter auf 81 Filter reduziert. Auf Datensatz 2 wurde sie von 209 Filter auf 80 Filter reduziert. In beiden Experimenten wurden also rund 62% der Filter verworfen. Der durchschnittliche Zeitaufwand konnte auf Datensatz 1 dagegen nur um $5{,}5\%$ und auf Datensatz 2 um 4% reduziert werden.

Mit (s_2, c_2) wurden die Filterkaskaden weniger stark auf im Schnitt 135 bzw. 132 Filter um 47% reduziert. Die Verarbeitungszeit eines Bildes durch einen Detektor verbessert sich um ca. 5-6 ms auf Datensatz 1 und auf Datensatz um ca. 2 ms, also um rund 3% bzw. $1{,}5\%$.

In der qualitativen Betrachtung der Detektionen wird bereits in den Visualisierungen (siehe Abbildung 5.3 und Anlage A.5) augenscheinlich, dass die Maxima kleiner ausfallen als vor der Reduktion und gute Klassifikatoren in einem weniger breiten Wertebereich auftreten.

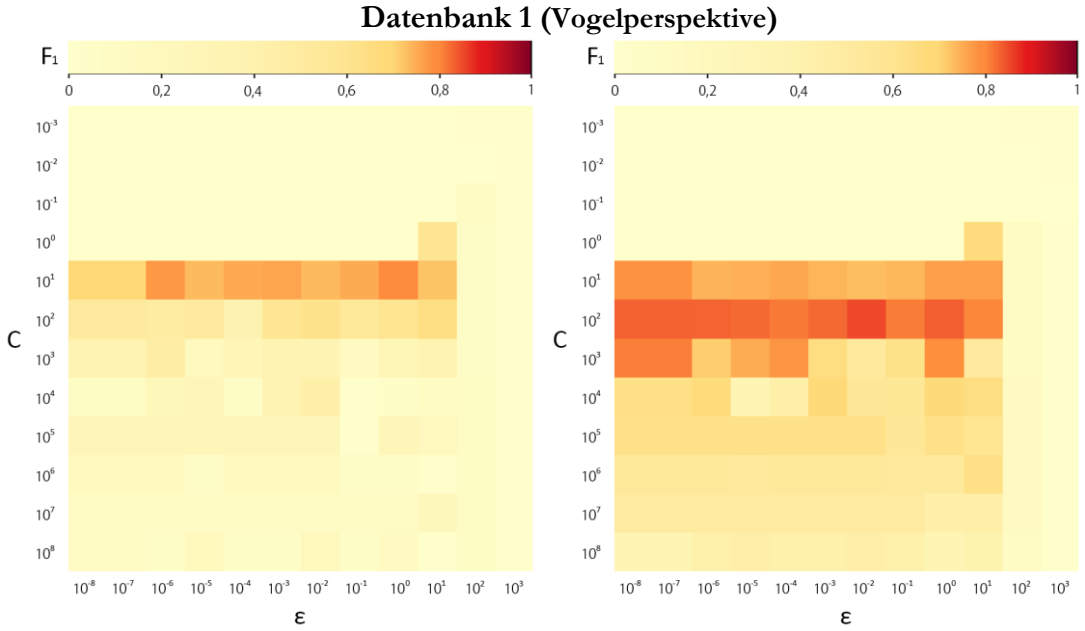


***Abbildung 5.3**: Heatmaps der auf Datenbank 1 gemessenen F_1-Maße der rfHOG-Detektoren. Links: Nach starker Reduktion erreicht kein Parameterpaar Werte von $F_1 > 0{,}8$. Nur für Parametrisierungen von $10^0 \leq C \leq 10^2$ werden überhaupt Detektoren mit $F_1 > 0{,}5$ trainiert. Rechts: Das gleiche Maß visualisiert für die moderate Reduktion. Hier sieht die Heatmap dem entsprechenden Pendant vor der*

Reduktion (oben links in Abbildung 5.1) schon deutlich ähnlicher, wenngleich alle Messungen noch immer niedriger sind. Vergleichbare Entwicklungen bei der Reduktion lassen sich auch in den Heatmaps auf der Datenbank 2 und im p_{avg}-Maß erkennen (vgl. Anlagen, Abbildungen A.13, A.14, A.15, A.16).

Quantitativ drückt sich dies in den Qualitätsmaßen wie folgt aus:

Tabelle 2: *Erreichte Maxima bei stark reduziertem fHOG*

Erreichte Maxima bei stark reduziertem fHOG						
	Datenbank 1 (Vogelperspektive)			**Datenbank 2 (Ich-Perspektive)**		
Qualitätsmaß	C	ε	**Maximalwert**	C	ε	**Maximalwert**
F_1	10^1	10^0	0,7945	10^2	10^{-3}	0,9631
F_2	10^1	10^0	0,7422	10^2	10^{-1}	0,9551
$F_{0,5}$	10^1	10^{-6}	0,8977	10^2	10^{-3}	0,9818
G	10^1	10^0	0,8431	10^2	10^{-3}	0,9636
p_{avg}	10^1	10^0	0,7004	10^2	10^{-1}	0,9499

Tabelle 3: Erreichte Maxima bei moderat reduziertem fHOG

Erreichte Maxima bei moderat reduziertem fHOG						
	Datenbank 1 (Vogelperspektive)			**Datenbank 2 (Ich-Perspektive)**		
Qualitätsmaß	C	ε	**Maximalwert**	C	ε	**Maximalwert**
F_1	10^2	10^{-2}	0,8517	10^4	10^{-5}	0,9672
F_2	10^2	10^{-2}	0,8011	10^4	10^{-5}	0,9579
$F_{0,5}$	10^2	10^{-6}	0,9104	10^2	10^{-3}	0,9820
G	10^2	10^{-2}	0,8565	10^4	10^{-5}	0,9674
p_{avg}	10^2	10^{-2}	0,7608	10^4	10^{-5}	0,9500

Auffällig ist, dass die rfHOG-Detektoren gegenüber fHOG im $F_{0,5}$-Maß weniger stark abfallen als im F_2-Maß. Da im $F_{0,5}$-Maß die Sensitivität r viermal stärker gewichtet wird (vgl. Formel (4.6)) und im F_2-Maß die Präzision p betont wird (vgl. Formel $F2= 5 \cdot \frac{p \cdot r}{4 \cdot p + r}$ (4.5)), lässt sich schließen, dass nach der Reduktion die Menge der falsch positiven Detektionen ansteigt, während die Menge der richtig positiven Detektionen unverändert bleibt (vgl. Formel (4.1)).

Auch für die im F_1-Maß besten aus der Reduktion hervorgegangenen Klassifikatoren wurden im Anschluss vollständige PR-Kurven ermittelt, die in der zusammenfassenden Auswertung in Abschnitt 5.4 zu finden sind.

5.3 Roboterdetektion unter Verwendung von NNR

Für die letzten Experimente wurden die trainierten Klassifikatoren einer NNR unterzogen, wie in Abschnitt 4.2. beschrieben. Wiederrum wurde der bekannte Parameterraum jeweils auf Datensatz 1 und Datensatz 2 in einer erschöpfenden Rastersuche durchlaufen, die Regularisierungsstärke wurde auf $\lambda = 10$ festgesetzt.

Die Experimente dauerten insgesamt erheblich länger. Die Trainingsphasen verlängerten sich bis um den Faktor 10, was darauf zurückzuführen ist, dass die zusätzliche Nebenbedingung des Regularisierungsterms (vgl. Formel (*3.1*)) erfüllt werden muss, ehe das Training als abgeschlossen betrachtet wird. Für die Qualität der resultierenden Detektoren ist die Dauer der Trainingsphase selbst aber unerheblich und wird im Folgenden deshalb nicht weiter betrachtet.

In der Visualisierung ist die Ausprägung der Qualitätsmaße im Parameterraum homogener auf Datensatz 1 und weniger homogen auf Datensatz 2 (vgl. Abbildung 5.4), bei den vorhergehenden Experimenten verhielt es sich umgekehrt.

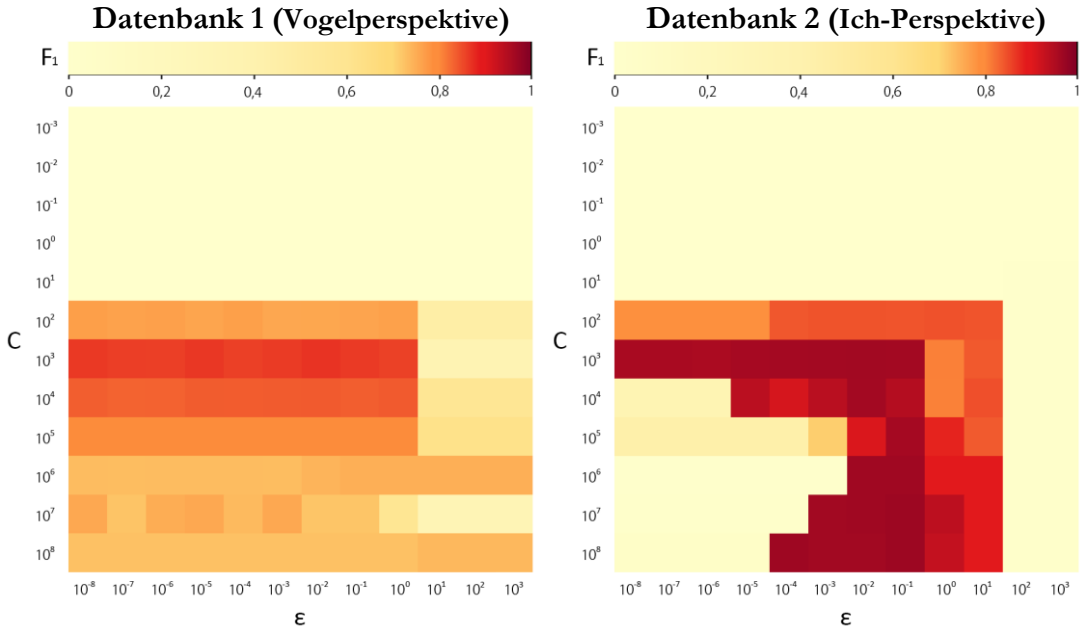


Abbildung 5.4: *Heatmaps der gemessenen F_1-Maße der NNR Detektoren. Links: Auf der Datenbank 1 scheint C in Teilbereichen nahezu ausschlaggebend für das Resultat zu sein. Für $C = 1$ werden z.B. konstant Werte von $F_1 > 0{,}8$ erreicht. Rechts: Das F_1-Maß auf der Datenbank 2 ist weniger uniform, insbesondere verglichen mit den Heatmaps der anderen Verfahren auf Datenbank 2 (vgl. Abbildung 5.1, untere Reihe). Insgesamt werden aber hohe Maxima von annähernd 1 erreicht (vgl. Tabelle 4).*

Die quantitative Veränderung in den Maxima lässt sich der nachstehenden Tabelle 4 entnehmen. Auf Datenbank 1 steigen die gemessenen Bestwerte bei F_1, F_2 und G minimal an und fallen in den Maßen $F_{0,5}$ und p_{avg} minimal ab.

Auch in der Testreihe auf Datenbank 2 werden in der Spitze mit der ersten Variante vergleichbare Werte erreicht.

Tabelle 4: *Erreichte Maxima bei NNR*

Erreichte Maxima bei NNR						
	Datenbank 1 (Vogelperspektive)			Datenbank 2 (Ich-Perspektive)		
Qualitätsmaß	C	ε	Maximalwert	C	ε	Maximalwert
F_1	10^3	10^{-2}	0,8699	10^8	10^{-1}	0,9696
F_2	10^3	10^{-2}	0,8321	10^8	10^{-1}	0,9654
$F_{0,5}$	10^3	10^{-2}	0,9113	10^3	10^{-3}	0,9813
G	10^3	10^{-2}	0,8724	10^8	10^{-1}	0,9696
p_{avg}	10^3	10^{-5}	0,7940	10^8	10^{-1}	0,9595

Die Filterkaskade wird auf Datenbank 1 im Durchschnitt 191 Filter groß und die durchschnittliche Verarbeitungszeit eines Bildes liegt bei 217 ms. Bei der Datenbank 2 sind die entstandenen Kaskaden rund 155 Filter groß und die Verarbeitungszeit liegt im Schnitt bei 127 ms. Erneut wurden bei der Ermittlung dieser Werte praktisch untrainierte Filter mit $F_1 < 0{,}01$ verworfen.

In einer weiteren Untersuchung wurden die NNR Klassifikatoren noch einmal auf Bilder der Größe 640×480 angewandt. Die Eingabebilder der Datenbank 2 wurden dazu in der Vorverarbeitung nicht in ihrer Breite und Höhe verdoppelt. Im Ergebnis werden diese Bilder ihrer Skalierung entsprechend in einem Viertel der Zeit verarbeitet. Die Qualitätsmaße bleiben in der Spitze trotzdem gut, sie verschlechtern sich nur um $2 - 3\%$ (vgl. Anlage, Abbildung A.17).

Auch für die besten unter NNR produzierten Klassifikatoren wurden PR-Kurven berechnet, indem der Schwellwert S_k systematisch verschoben wurde.

5.4 Vergleich der Varianten

In diesem Abschnitt werden die Klassifikatoren, die in den verschiedenen Verfahren die Maxima der jeweiligen Maße hervorgebracht haben, einander gegenübergestellt.

Eine etablierte Methode ist es die PR-Kurven der zu vergleichenden Klassifikatoren gemeinsam abzutragen [3, 42]. Die PR-Kurven illustrieren die mögliche Austauschbeziehung zwischen der Präzision und der Sensitivität eines Klassifikators. Ein Algorithmus, der eine große Fläche unter der PR-Kurve liefert, ermöglicht es, passend zum Anwendungsfall die Präzision oder die Sensitivität zu betonen, ohne das andere Gütekriterium zu stark negativ zu beeinflussen (siehe Abbildung 5.5, 5.7).

Auf der Datenbank 1 stechen diesbezüglich der fHOG-Klassifikator, der das beste $F_{0,5}$-Maß hervorgebracht hatte und der NNR Klassifikator, der im F_1-, F_2-, $F_{0,5}$- und G-Maß das Maximum hatte, hervor. Für beide Klassifikatoren ist es möglich die Sensitivität auf bis zu 90% zu regulieren, ohne dass die Präzision unter 90% fällt. Ein solcher Schnittpunkt mit der Identitätsfunktion (auch PR-Breakeven-Punkt genannt) eröffnet in der praktischen

Anwendung ein breites Einsatzspektrum (vgl. Abbildung 5.6). Die ideale PR-Kurve wäre die Gerade $P(r) = 1$ (5.1) mit dem PR-Breakeven-Punkt $(1,1)$.

PR-Kurven der Klassifikatoren, Datenbank 1 (Vogelperspektive)

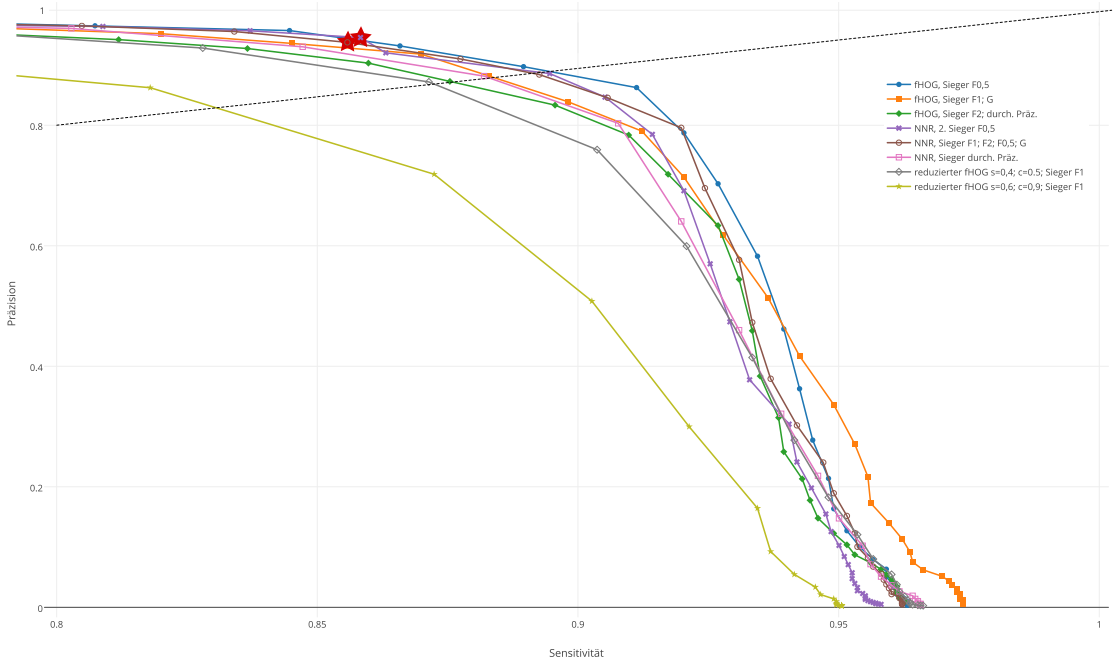


Abbildung 5.5: *Ausschnitt aus den PR-Kurven der auf Datensatz 1 am besten abschneidenden Detektoren. Verschiebt man von einem Messpunkt aus nach links, wird die Präzision auf Kosten der Sensitivität betont. Verschiebt man nach rechts, betont dies die Sensitivität. Auffällig sind die Kurven des fHOG $F_{0,5}$-Maß Siegers (blau) und die des NNR Siegers im F_1-,F_2-,$F_{0,5}$- und G-Maß (braun). Ihr PR-Breakeven-Punkt liegt weit rechts. Lange lässt sich die Sensitivität ohne große Kosten hinsichtlich der Präzision verbessern. Als rote Sterne sind zusätzlich die beiden Messpunkte mit dem höchsten und zweithöchstem gemessenen F_1-Maß markiert. Die Identitätsfunktion ist als gestrichelte, schwarze Linie eingezeichnet.*

Möchte man nur nach dem hinsichtlich Präzision und Sensitivität neutralem F_1-Maß optimieren, findet man bei $S_k = -0{,}25$ das maximale $F_1 = 0{,}9015$ aller Kurven bei dem NNR Klassifikator, der in der Kreuzvalidierung das maximale $F_{0,5}$-Maß hatte.

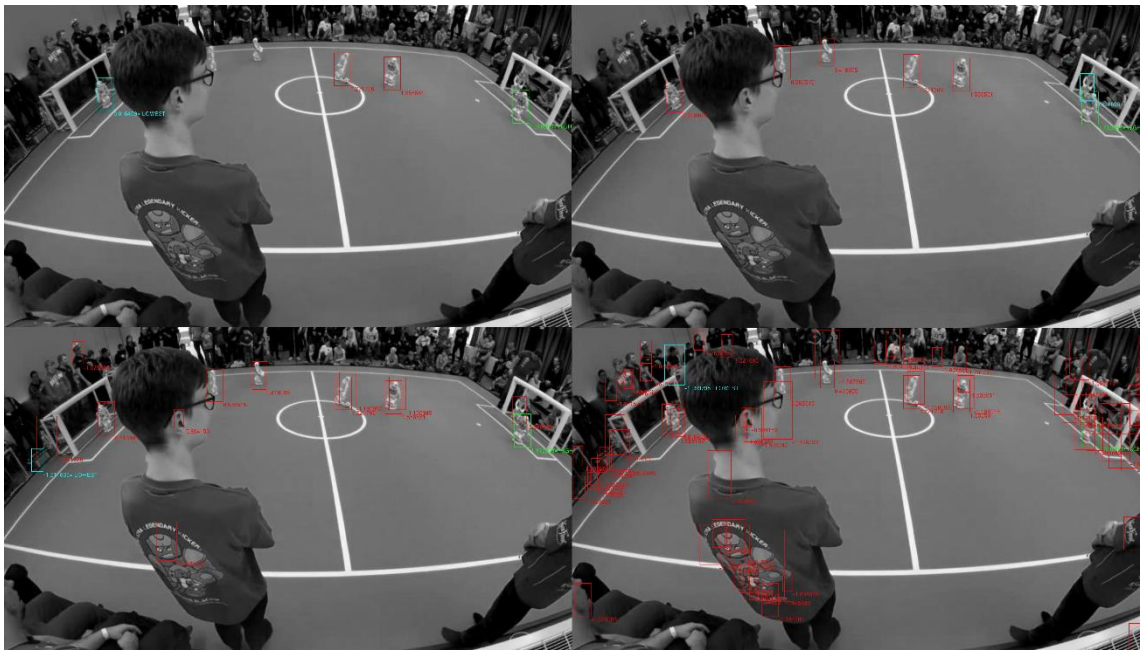


Abbildung 5.6: *Derselbe Klassifikator durchläuft (in Leserichtung) von links oben nach rechts unten die PR-Kurve durch Verschiebung des Schwellwertes S_k. Hellblau markiert ist die Detektion mit der geringsten Konfidenz, grün markiert ist die Detektion mit der höchsten Konfidenz und rot markiert sind alle anderen Detektionen. Für hohe Schwellwerte (oben links) ist der Klassifikator sehr präzise, aber findet nicht alle Roboter. Da dieser Detektor sehr stabil über die PR-Kurve ist, lässt sich ein Schwellwert finden (oben rechts) für den alle aktiven Roboter, gefunden werden, ohne dass es zu falsch positiven Detektionen kommt. In der unteren Reihe wird der Schwellwert immer weiter gesenkt. Dies generiert immer mehr Detektionen. Diese Detektoren haben die höchste Sensitivität, es werden sogar die inaktiven Roboter links hinter dem Tor detektiert. Auch ein solcher Detektor kann Verwendung finden, indem z.B. die vielen generierten Detektionen einem anderen Modul als Eingabe dienen.*

Die PR-Kurven der Klassifikatoren lassen sich auf Datenbank 2 schwerer voneinander trennen (vgl. Abbildung 5.7). Die fHOG und die NNR Klassifikatoren separieren sich erst für sehr hohe $r > 0{,}95$. Den besten PR-Breakeven-Punkt bei $r = 0{,}973$ hat der NNR Klassifikator, der im F_1-, F_2- und G-Maß die Maxima hatte. Der Klassifikator erreicht bei $S_k = 0{,}1$ auch das beste F_1-Maß auf der gesamten PR-Kurve mit $F_1 = 0{,}9804$ und ist damit die erste Wahl für die Detektion anderer Roboter aus der Ich-Perspektive.

PR-Kurven der Klassifikatoren, Datenbank 2 (Ich-Perspektive)

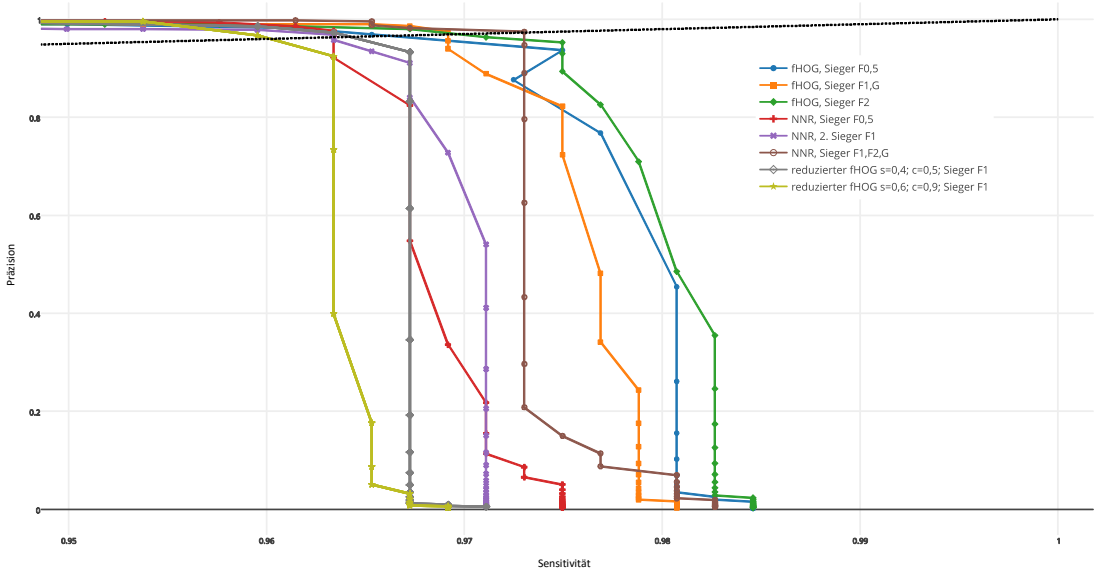


Abbildung 5.7: *Ausschnitt aus den PR-Kurven der auf Datensatz 2 am besten abschneidenden Detektoren. Für Werte der Sensitivität* $r < 0{,}95$ *ist die Präzision sehr hoch, es gibt weniger als eine falsch positive Detektion auf 100 Detektionen, also etwa alle 50 Bilder. Der braun eingezeichnete NNR Klassifikator, der die Maxima im* F_1*-,* F_2*- und* G*-Maß hatte, hält dieses Niveau am längsten, bis hin zu einer Sensitivität von* $r = 0{,}9653$*. Dort liegt auch sein Schnittpunkt mit der schwarz gestrichelten Identitätsfunktion.*

Insgesamt liefern alle der besten Klassifikatoren vergleichbare Ergebnisse auf der Datenbank 2. Der Unterschied zwischen allen Detektoren beläuft sich in absoluten Zahlen auf dem gesamten Datensatz von 1578 Robotern auf ± 10 falsch positive oder falsch negative Erkennungen. Optimierungspotenzial liegt daher über die Sensitivitätsanpassung hinaus in anderen Bereichen (siehe Abschnitt 5.5).

In der Tabelle 5 sind noch einmal die Klassifikatoren, die in den drei Verfahren Maximalwerte in einem Qualitätsmaß erzielt haben, einander gegenübergestellt. Grau hinterlegt wurden die über alle Verfahren hinweg auf einer Datenbank erreichten Maximalwerte. Zur besseren Einordnung sind zusätzlich der arithmetische Mittelwert $\emptyset$ und die empirische Standardabweichung σ der gelisteten Werte aufgeführt.

Tabelle 5: *Quantitative Gegenüberstellung der Verfahren*

Quantitative Gegenüberstellung der Verfahren									
Datenbank 1 (Vogelperspektive)									
Verfahren	**C**	**ε**	**p**	**r**	**F_1**	**F_2**	**$F_{0,5}$**	**G**	**p_{avg}**
fHOG	10^2	10^{-8}	0,9611	0,7850	0,8642	0,8149	0,9199	0,8687	0,7771
fHOG	10^3	10^{-5}	0,9409	0,8059	0,8681	0,8297	0,9104	0,8709	0,7930
fHOG	10^4	10^{-8}	0,9135	0,8116	0,8600	0,8301	0,8911	0,8610	**0,7967**
rfHOG(stark)	10^1	10^0	0,9186	0,7133	0,7945	0,7422	0,8616	0,8431	0,7004
rfHOG(stark)	10^1	10^{-6}	0,9708	0,6844	0,7764	0,6956	0,8811	0,7934	0,6457
rfHOG(mod.)	10^2	10^{-2}	0,9521	0,7706	0,8517	0,8011	0,9093	0,8565	0,7608
rfHOG(mod.)	10^2	10^{-6}	**0,9718**	0,7272	0,8318	0,7657	0,9104	0,8406	0,7198
NNR	10^3	10^{-2}	0,9412	0,8087	0,8699	0,8321	0,9113	0,8724	0,7937
NNR	10^3	10^{-5}	0,9377	0,8054	0,8288	0,8665	0,9079	0,8691	0,7940
NNR($s_k = -0,25$)	10^3	10^{-8}	0,9379	**0,8679**	**0,9015**	**0,8810**	**0,9230**	**0,9022**	0,7846
Ø			0,9446	0,7780	0,8475	0,8122	0,9043	0,8582	0,7587
σ			0,0198	0,0550	0,0398	0,0634	0,0197	0,0339	0,0565
Datenbank 2 (Ich-Perspektive)									
fHOG	10^2	10^{-2}	0,9868	0,9425	0,9641	0,9510	0,9727	0,9644	0,9415
fHOG	10^3	10^{-8}	0,9745	**0,9659**	0,9702	0,9676	0,9727	0,9702	**0,9627**
fHOG	10^6	10^{-5}	0,9802	0,9639	0,9720	0,9671	0,9769	0,9720	0,9608
rfHOG(stark)	10^2	10^{-1}	0,9687	0,9518	0,9602	0,9551	0,9653	0,9602	0,9499
rHOG(stark)	10^2	10^{-3}	0,9948	0,9133	0,9631	0,9451	0,9818	0,9636	0,9313
rfHOG(mod.)	10^2	10^{-3}	0,9959	0,9306	0,9621	0,9430	0,9820	0,9627	0,9290
rfHOG(mod.)	10^4	10^{-5}	0,9833	0,9518	0,9672	0,9579	0,9768	0,9674	0,9500
NNR	10^3	10^{-3}	0,9929	0,9377	0,9645	0,9483	0,9813	0,9649	0,9365
NNR	10^8	10^{-1}	0,9767	0,9626	0,9696	0,9654	0,9738	0,9696	0,9595
NNR($s_k = 0,1$)	10^8	10^{-1}	**0,9960**	0,9653	**0,9804**	**0,9713**	**0,9897**	**0,9806**	0,9595
Ø			0,9850	0,9485	0,9655	0,9573	0,9774	0,9657	0,9436
σ			0,0098	0,0175	0,0046	0,0116	0,0073	0,0045	0,0147

5.5 Diskussion der Ergebnisse

In beiden getesteten Anwendungsszenarien ermöglichen die umgesetzten Verfahren grundsätzlich die formbasierte, visuelle Detektion von Nao Robotern. Auf beiden Datensätzen scheint zwischen fHOG und reduziertem HOG ein Austausch hinsichtlich der Qualität der Detektion und der Verarbeitungszeit möglich. Dieser ist allerdings in beiden Richtungen auf den einstelligen Prozentbereich beschränkt. Spart man z.B. auf Datensatz 1 bei der moderaten Reduktion noch 3% Zeitaufwand bei nur 2% Verlust im F_1-Maß, so

bricht das F_1-Maß bei der starken Reduktion um deutliche $8{,}5\%$ ein, mit einem Zeitgewinn von gerade $5{,}5\%$. Wenn man die Ausdünnung hoher Maxima in der Rastersuche als zusätzliche Tendenz betrachtet, ist auszuschließen, dass sich die Filterkaskade noch viel weiter reduzieren lässt, ohne dass die Detektoren unbrauchbar werden. In der Betrachtung der PR-Kurven fallen die reduzierten Klassifikatoren gegenüber allen anderen Klassifikatoren merklich ab.

Durch NNR lässt sich die Filterkaskade effizienter reduzieren. Auch hier kann der Zeitaufwand im Millisekundenbereich verbessert werden, allerdings ohne dass damit eine Verschlechterung der Qualität einhergeht. Im Gegenteil platzieren sich die NNR Klassifikatoren in den PR-Kurven-Betrachtungen in der Spitze.

Was den geplanten Einsatzweck betrifft, sind weniger als 220 ms Verarbeitungszeit pro hochaufgelöstem 1080p Bild zweckmäßig. Ein Video von 15 Minuten Länge lässt sich auf dem Testcomputer bei 24 Bildern pro Sekunde in ca. 79 Minuten komplett auswerten. Wenn man die Auswertung auf 5 Bilder pro Sekunde reduziert, lässt sich ein 1080p Video in Echtzeit auswerten, also z.B. parallel zum gerade ablaufenden Spiel. Falls notwendig, lässt sich die Bildfrequenz auch durch einen entsprechend potenteren Computer verbessern.

Bei der Detektion aus der Vogelperspektive ist die Qualität der Erkennung zufriedenstellend, kann allerdings hinsichtlich der Sensitivität in der Spitze noch verbessert werden. Gelegentlich werden Roboter über mehrere Bilder hinweg nicht detektiert. Falsch negative Roboter sind vor allem solche, die verdeckt und nur in Teilen sichtbar sind, sowie Roboter die am Bildrand vorkommen. Falsch positive Detektionen treten am häufigsten bei Robotern auf, die nicht aktiv am Spiel teilnehmen, also umgefallene, liegende oder aus dem Spielfeld gestellte Roboter. Wird die Sensitivität hoch reguliert, kommen auch Zuschauer und andere formähnliche Objekte als Detektionen dazu. Gute Beispiele für diese falsch negativen, bzw. falsch positiven Roboter finden sich in den Anlagen in Abbildung A.7 und in Abbildung A.8. Hier bleibt zu diskutieren, wie die als zu ignorieren markierten Roboter ausgewertet werden sollen. Exemplarisch wurde für den NNR Klassifikator, der im F_1-, F_2-, $F_{0,5}$- und G-Maß das Maximum hatte, ermittelt, dass 280 von 568 falsch positiven Detektionen solche Roboter waren, die als zu ignorieren markiert sind. Legt man zu Grunde, dass das Verfahren solche Roboter nicht unterscheiden kann und verwirft diese falsch positiven Detektionen, verbessert sich die Präzision des Detektors in der Auswertung von 94% auf 97%.

Aus den PR-Kurven lässt sich ablesen, dass sich die Sensitivität auf Werte von $r > 0{,}9$ steigern lässt, allerdings verringert sich dabei zunehmend die Präzision. Abhilfe schaffen kann hier die Anwendung von spezifischem Domänenwissen. Reguliert man den Akzeptanzschwellwert nach unten, um mehr Detektionen zu generieren, müssen neue falsch positive Detektionen verworfen werden. Zum Beispiel können falsch positive Detektionen, die nicht auf dem Spielfeld liegen, mit einem Grünwertfilter oder durch die Spielfelddetektion eleminiert werden.

Die Detektion aus der Ich-Perspektive der Roboter funktioniert in den Gütekriterien bereits ausgezeichnet. Im Vergleich zur Datenbank 1 ist die Datenbank 2 insgesamt homogener. Obwohl versucht wurde verschiedene Spielszenen nachzustellen, fehlen die unvorhersehbaren Situationen, die im Wettbewerb durch Zuschauer, Schiedsrichter oder andere Umstände einfließen. Im Besonderen sind in Datenbank 2 keine inaktiven, zu ignorierenden Roboter enthalten, die bei den Detektoren auf der Datenbank 1 die Hälfte der falsch positiven Detektionen ausmachen. Generell sind die Roboter aber auch größer im Bild, als bei der entfernt aufgestellten Kamera, was die Detektion ebenso positiv beeinflussen kann.

Ein weiteres Problem betrifft die Geschwindigkeit der Detektoren. Die Anwendung der reduzierten Kaskaden verbessert die Geschwindigkeit nicht signifikant, daher ist die Reduktion der Bildgröße in Kombination mit einer Spezifizierung der zu untersuchenden Bildbereiche vielversprechender. Werden die Bilder nicht zusätzlich hochskaliert, können sie in einem Viertel der ursprünglichen Zeit verarbeitet werden. Wird das Wahrnehmungsmodell des Roboters mit einbezogen, so dass Roboter oder Bildbereiche über mehrere Bilder hinweg untersucht werden können, dann muss nicht mehr das gesamte Bild ausgewertet werden und der Einsatz auf dem Nao Roboter wird praktikabel. In ersten Tests dauerte die Verarbeitung eines 100×100 Pixel Teilbereichs auf dem Testcomputer weniger als 3 ms. In den weiterführenden Experimenten in Abschnitt 6.2 wird ein auf HOG basierendes Trackingverfahren (aus dem Englischen, Verfolgung) eingesetzt, das auf lokalen Bildbereichen arbeitet. Weiterhin ist es denkbar, den Detektor selektiv, in Abhängigkeit von der Spielsituation, auf dem Roboter einzuschalten. Eine integrierte Lösung auf dem Roboter ist möglich.

Es ist schwer die Ergebnisse mit anderen Verfahren in Relation zu setzen. Vergleichende Arbeiten für visuelle, formbasierte Detektoren wurden zur Erkennung von Fußgängern auf entsprechenden Datenbanken durchgeführt [3, 4] und sind daher nicht komparabel. Die im Rahmen dieser Arbeit erstellten Datenbanken (vgl. Abschnitt 4.1) werden öffentlich verfügbar gemacht, so dass in Zukunft weitere Verfahren auf diesen evaluiert werden können. Entsprechende Arbeiten stehen noch aus.

6 Zusammenfassung und Ausblick

Mit der Auswertung der empirischen Resultate der implementierten Verfahren, wurde die Entwicklung einer geeigneten, auf HOG basierenden Roboterdetektion erfolgreich vollzogen. Die Kernpunkte der Arbeit werden hier noch einmal zusammengetragen. Abschließend wird beschrieben wie die Ergebnisse dieser Arbeit weitergeführt werden können. Neben Optimierungen der Detektion selbst, werden einige bereits durchgeführte Experimente beschrieben und es werden weiterführende Projekte im Ausblick vorgestellt.

6.1 Zusammenfassung

Zu Beginn dieser Arbeit wurde die Motivation für die Entwicklung einer formbasierte, visuelle Detektion humanoider Roboter dargelegt. Wesentlicher Antrieb findet sich im Anwendungsgebiet des RoboCups begründet. Die stetige Progression in den Regularien der SPL stellt Roboter und Forscher vor zunehmend komplexere Aufgaben. Die Rahmenbedingungen und bisherigen Lösungen wurden im ersten Kapitel dieser Arbeit beschrieben.

Im zweiten Kapitel wurden die theoretischen Grundlagen der im Kontext des RoboCups und der humanoiden Robotik vielversprechendsten Verfahren der Merkmalsextraktion und des maschinellen Sehens aufgearbeitet. Aufgrund der universellen Erfolge der Klassifikation verschiedenartiger Objekte basierend auf HOG-Merkmalen und des besonderen Fortschritts in der Detektion von Fußgängern und humanoiden Robotern, wurden drei auf HOG basierende Verfahren zur Evaluation ausgewählt. Die Verfahren unterscheiden sich hinsichtlich der Komplexität in ihren Klassifikatoren.

Das erste Verfahren modelliert ein DPM und erzeugt fHOG genannte Merkmale. Das Ergebnis ist eine Filterkaskade, die Roboter auch dann erkennt, wenn diese nur in Teilen zu sehen sind. Solange der Roboter nicht vollständig verdeckt ist und sich die erkannten Teile zu einem Roboter zusammenfügen lassen, kann das DPM den Roboter detektieren. In der zweiten Variante wird die Filterkaskade reduziert. Es entsteht eine flachere Filterkaskade bis hin zu einem einzigen Filter. Das dritte Verfahren regularisiert den Klassifikator, indem beim Training eine NNR durchgeführt wird. Ziel des Verfahrens ist eine bessere Generalisierung bei weniger komplexen Filtern. Im dritten Kapitel wurde die praktische Implementierung der verschiedenen Verfahren beschrieben. Im Training der Klassifikatoren wurden nochmals einige sehr praxisnahe Probleme analysiert, unter anderem die Dauer des Trainings, die Speicherlast und die Protokollierung aller Resultate.

Im realen Einsatz wurden zwei grundsätzlich verschiedene Anwendungsszenarien ausgemacht. Es gibt die externe Beobachtung der Roboter durch eine außerhalb des Spielfeldes stehende Kamera aus der Vogelperspektive und die Detektion anderer Roboter durch einen selber am Fußballspiel teilnehmenden Roboter aus der Ich-Perspektive. Beiden Szenarien liegen unterschiedliche Perspektiven zugrunde. Die jeweilige Aufgabenstellung

unterscheidet sich dadurch fundamental. In einem Bild des weit vom Spielfeld entfernt stehenden Beobachters sind meistens viele Roboter gleichzeitig und in Gänze zu sehen. Das Bild reicht über die Grenzen des Spielfeldes hinaus, so dass Zuschauer oder inaktive Roboter zu sehen sind. Die Roboter selbst sind im Bild relativ klein. Aus der Perspektive des Roboters sind andere Roboter dagegen größer, nehmen große Bildbereiche ein und sind oft nur in Teilen zu sehen. Ein Roboter sieht meistens nicht alle anderen Roboter, außerdem verdecken die Roboter einander häufiger.

In der empirischen Untersuchung im vierten Kapitel wurden alle Verfahren für beide Anwendungsfälle getestet. Zur Durchführung der Experimente wurden zwei repräsentative Datenbanken von markierten Nao Robotern erstellt und die Verfahren auf diesen durchgeführt. Auf den Datenbanken wurden dabei erschöpfende Rastersuchen zur Betrachtung eines ausgewählten Parameterraums und zur Ermittlung von optimalen Parametrisierungen ausgeführt und die gelernten Klassifikatoren wurden mittels Kreuzvalidierung kontrolliert. Aus den gesammelten Daten wurden qualitative Messzahlen für alle Verfahren ermittelt.

In der Auswertung dieser Qualitätsmaße im fünften Kapitel der Arbeit wurde ein abschließendes Fazit zur Detektion und zum Einsatzgebiet der jeweiligen Verfahren gezogen. Die Detektion war hinsichtlich ihrer Qualität in beiden Anwendungsfällen gut. Die Roboter wurden auch in komplizierten Situationen zuverlässig erkannt. Für die Detektion aus der Vogelperspektive ist die Verarbeitungszeit bereits zweckmäßig. Es wurde Potential zur Verbesserung der Detektionssensitivität in der Regulierung der Detektionsschwelle und in der Weiterverarbeitung der Detektionen beschrieben.

Der Einsatz auf dem Roboter selber ist weiterhin durch den verhältnismäßig hohen Zeitaufwand bei der Berechnung der HOG-Merkmale auf großen Bildern beschränkt. In der Verwendung auf dem Roboter muss daher entweder eine verkleinerte Version des Eingabebildes verwendet werden oder der Einsatz muss auf Teilbereiche des Bildes beschränkt werden. In der Integration der qualitativ guten Detektionen in das Wahrnehmungs- und schließlich Weltmodell des Roboters liegt der nächste Entwicklungsschritt.

6.2 Weiterführende Experimente

Über die in Kapitel 5 ausgewerteten Experimente hinaus wurde eine Reihe erster Experimente durchgeführt, die auf den implementierten Verfahren zur visuellen, formbasierten Detektion mittels HOG basieren und diese erweitern.

Neben der Roboterdetektion wurde die Arbeit an weiteren Projekten begonnen, in denen die fHOG-Merkmale und Klassifikatoren zum Einsatz kommen. Die Erkennung anderer im Rahmen des RoboCups relevanter Objekte ist möglich. Hervorzuheben sind dabei Teamlogos, die in ihren minimalistischen, klaren Formen Straßenschildern ähneln, für deren Erkennung HOG bereits mehrfach erfolgreich eingesetzt wurde. [31, 43]

Eine erste auf HOG basierende Balldetektion wurde bereits getestet. Verwendet wurden Testspielaufnahmen eines Roboters. Die erreichte Qualität der Balldetektion liegt auf den Testdaten im F_1-Maß bei $F_1 \approx 0{,}7 - 0{,}75$ mit einer Genauigkeit von 100%. Der Zeitaufwand disqualifiziert allerdings die Methode als erste Wahl zur Ballerkennung auf dem Roboter.

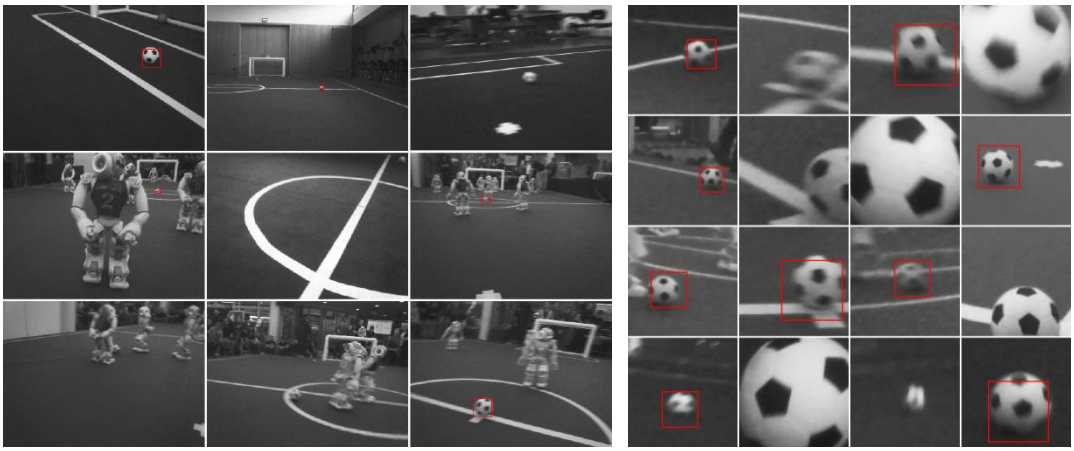


Abbildung 6.1: *Beispielhafte Detektionen eines auf HOG basierenden Balldetektors. Links: Anwendung der Balldetektion auf das gesamte Bild, die Detektionen sind korrekt, allerdings werden unscharfe und verdeckte Bälle nicht zuverlässig gefunden, Rechts: Ergebnisse der Klassifikation von Bildausschnitten der Größe* 100×100 *Pixel*

Interessant wird hier ein weiterer Aspekt der auf HOG-Merkmalen basierenden Objekterkennung. Mehrere Klassifikatoren lassen sich sehr gut parallelisieren. Der Aufwand die Merkmale zu berechnen, ist im Verfahren maßgeblich für die gesamte Verarbeitungszeit. Gleichzeitig heißt das, nachdem diese einmal berechnet wurden, kann eine beliebige Anzahl an Klassifikatoren darauf angewandt werden. Jeder Klassifikator nach dem ersten Klassifikator ist somit sehr zeiteffizient.

Das heißt, wird HOG eingesetzt, um z.B. die Roboter in einem Bildbereich zu detektieren, können, fast ohne zusätzlichen Zeitaufwand, weitere Objekte wie z.B. Ball, Teamlogos oder die Pose der Roboter identifiziert werden. Auf Datenbank 1 wurden dafür bereits Tests durchgeführt, um die detektierten Roboter hinsichtlich verschiedener Eigenschaften zu klassifizieren. Es wurden Klassifikatoren trainiert, um die Roboter als stehend, sitzend, liegend oder als Torhüter zu unterscheiden (vgl. Abbildung 6.2). Der zeitliche Aufwand stieg dabei pro zusätzlichem Klassifikator um ca. 10% gegenüber dem Aufwand mit einem Detektor an.

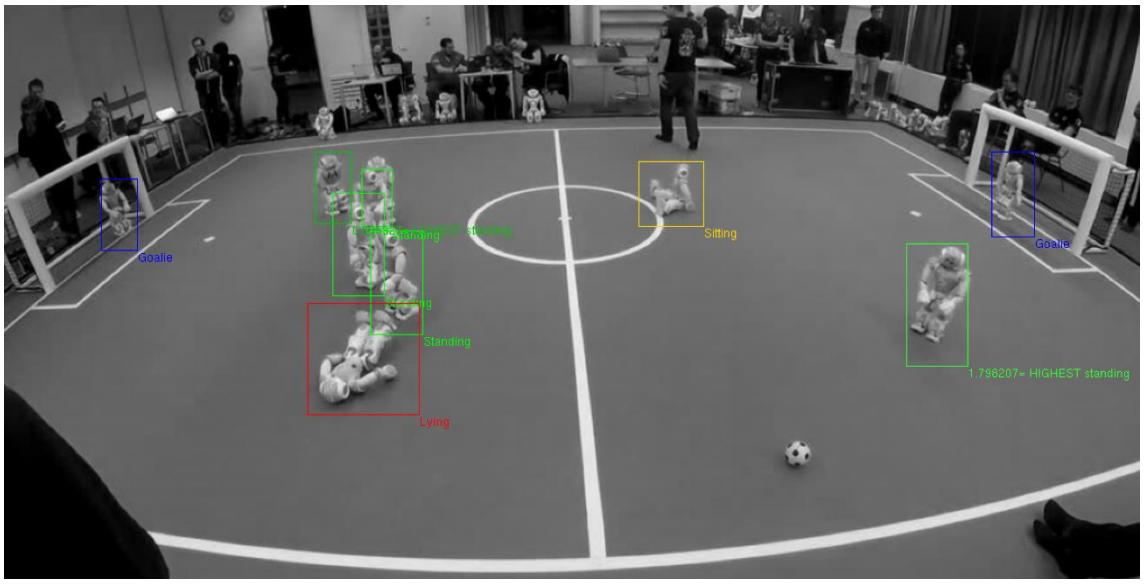


Abbildung 6.2: *Beispielhafte Detektionen von vier parallel betriebenen Klassifikatoren zur Detektion von Posen. Korrekt identifiziert sind die Torhüter in Blau, der liegende Roboter in Rot, der sitzende Roboter in Gelb und die aktiven, stehenden Roboter in Grün.*

Ein weiteres Verfahren, das bereits experimentell umgesetzt wurde, ist das Tracking der Roboter über mehrere Bilder hinweg. Das Verfahren verwendet einen in [44] beschriebenen Korrelationsfilter auf den generierten HOG-Merkmalen. Das Verfahren untersucht für eine Detektion im nachfolgenden Bild die Umgebung und verschiebt die Detektion in Richtung der höchsten Korrelation. Wurde ein Roboter bereits detektiert, benötigt dieser lokal arbeitende Algorithmus auf dem Testcomputer ca. 5-8 ms zum Tracken eines Roboters. Das heißt in einem Spiel 5 gegen 5, können alle Roboter in weniger als 80 ms verfolgt werden.

6.3 Ausblick

Neben der Weiterentwicklung der implementierten Verfahren wie im vorhergehenden Abschnitt beschrieben, besteht die Möglichkeit, die erstellten Detektoren zu optimieren und in existierende Projekte einzubinden.

Ohne weitere Modifizierungen, sondern nur durch Zeitaufwand auf einem Trainingscomputer, lassen sich nochmal die Wertebereiche mit der höchsten Dichte an guten Klassifikatoren in einem höher aufgelösten Raster durchlaufen, in der Erwartung neue Maxima zu finden. Aus den Heatmaps lassen sich diese als rote und dunkelrote Bereiche ablesen. Zum Beispiel wären für das fHOG-Verfahren auf Datenbank 1 die Intervalle $[10^1, 10^5]$ für C und $[10^{-8}, 10^{-1}]$ für ε von Interesse (vgl. Abbildung 5.1).

Ein weiterer Ansatz ist das Ausweiten des Parametersuchraums auf eine höhere Dimensionalität. Der am besten geeignete Parameter ist der Akzeptanzschwellwert S_k, so dass für jedes Paar (C_m, ε_n) implizit die PR-Kurve ermittelt wird. In den berechneten PR-Kurven lagen alle Maxima innerhalb des Intervalls $[-2,1]$, das auf der Datenbank 1

(Vogelperspektive) die Kurven von $r = 0{,}5$ bis $r = 0{,}95$ und auf der Datenbank 2 die Kurven von $r = 0{,}75$ bis $r = 0{,}98$ abbildet, so dass dies ein guter Wertebereich für die Suche wäre.

Zusätzliche Parameter die optimiert werden können, sind L_{miss} und L_{fa} des eingesetzten MMOD-Algorithmus, wobei L_{miss} im MMOD die Betonung der falsch negativen Detektionen und damit der Sensitivität steuert. Da in beiden Datenbanken und über alle Klassifikatoren hinweg die Genauigkeit größer war, als die Sensitivität für die F_1-Maxima, liegt es nahe, $L_{miss} > 1$ zu durchsuchen.

Außerdem könnte die Untersuchung weiterer SVM Kernelfunktionen, z.B. einem Kernel mit einer RBF, in die Modellauswahl einbezogen werden. Da jedoch die größten Sprünge in der praktischen Verwertbarkeit der Detektoren nicht in der weiteren Parameteroptimierung erwartet wird, verfolgen wir aktuell andere Ansätze.

Eine merkliche Verbesserung der Qualitätsmaße kann durch die Weiterverarbeitung der Detektionen erreicht werden. Nach Tests mit einem einfachen Grünwertfilter (vgl. Abbildung 6.3), wird aktuell an der Einbindung der Spielfelddetektion des NaoTH-Frameworks gearbeitet. In einem weiteren Schritt ist es denkbar, Objekte zu filtern, die nicht der Größe des Nao Roboters entsprechen, z.B. wenn der Kameraabstand bekannt ist.

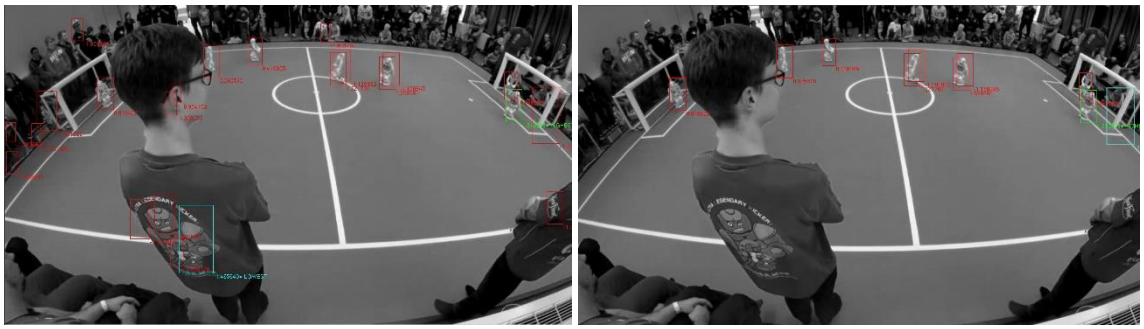


Abbildung 6.3: *Gegenüberstellung einer Detektion mit und ohne nachgeschaltetem Grünwertfilter. Links: Szene aus der RoboCup EuropeanOpen 2016 mit stark hochregulierter Sensitivität, Rechts: dieselbe Szene nach dem Zuschalten des Grünwertfilters.*

Für den Betrieb auf dem Roboter ist das Ziel der Aufbau eines integrierten Systems bestehend aus Detektion, Weltmodell des Roboters und lokalem Tracking. Die Detektionen können weiterverarbeitet werden und der Zeitaufwand soll durch die Auswahl von Bildbereichen reduziert werden. Eine weiterführende Anwendung der Detektoren aus der Vogelperspektive findet sich in einem aktuellen Projekt des NaoTH. Das NaoTH arbeitet an einem Werkzeug[29] für die systematische Erfassung und Analyse von Daten des RoboCups [45]. Das Projekt hat zur Weiterentwicklung Unterstützung durch den RoboCup erfahren. Es ist geplant die zuvor beschriebenen Verfahren zur Detektion, zum automatisierten Tracking und zur Klassifikation der Posen in dieses Projekt einzubinden.

[29] https://www2.informatik.hu-berlin.de/~naoth/videolabeling/, abgerufen am 22.01.2017

A. Anlagenverzeichnis

In den Anlagen wird diese Arbeit ergänzendes Material vorgestellt. Die Anlagen enthalten Beschreibungen der SPL, Spezfikationen der Roboterplattform Nao und des in den Experimenten verwendeten Computers, sowie exemplarische Detektionsergebnisse und weitere Heatmaps der empirischen Analyse.

A.1 Standard Platform League

Die SPL ist eine der Ligen des RoboCup Wettbewerbes. Die Liga zeichnet sich durch den Einsatz baugleicher Roboter als Zielplattform aller teilnehmenden Teams aus. Im Jahr 1999 begann die Liga als Einladungsturnier und verwendete die hundeähnliche Roboterplattform AIBO (Sony Corporation, Tokio). Drei Jahre später eröffnete die Liga einen Qualifikationsprozess, so dass weitere Teams beitreten konnten. Mit dem Wechsel auf die Roboterplattform Nao in den Jahren 2008-2009 änderte sich auch die vierbeinige Fortbewegung der Roboter hin zur Bipedie.

Die SPL reguliert nicht nur die Art der eingesetzten Roboter, sondern macht im jährlich aktualisierten Regelwerk[30] auch allgemeine Vorgaben zum Spiel- und Turnierablauf, dem Ball und dem Spielfeld. Es werden die für die experimentelle Analyse dieser Arbeit relevanten Regeln in Kürze aufgeführt.

Grundregeln: Gespielt wird eine Partie zwischen zwei Mannschaften, bestehend aus bis zu 5 Spielern und einem Trainer. Es darf maximal 1 Spieler als Torwart auftreten, die anderen Spieler werden als Feldspieler bezeichnet. Der Torwart darf den Ball innerhalb des eigenen Strafraums mit der Hand berühren. Der Trainer darf das Spiel beobachten und taktische Anweisungen kommunizieren. Roboter tragen ihrem Team entsprechend einheitlich gefärbte Trikots. Das Spiel besteht aus zwei Halbzeiten von 10 Minuten Länge. Ein Tor ist erzielt, wenn der Ball die Torlinie eine volle Umdrehung überschritten hat. Die Regeln garantieren einen flüssigen Spielablauf. Spieler die den Spielablauf oder gegnerische Spieler irregulär behindern, werden temporär vom Spielfeld entfernt. Spieler können zu diesem Zweck vom Spielleiter über ihren bzw. den Spielstatus in Kenntnis gesetzt werden.

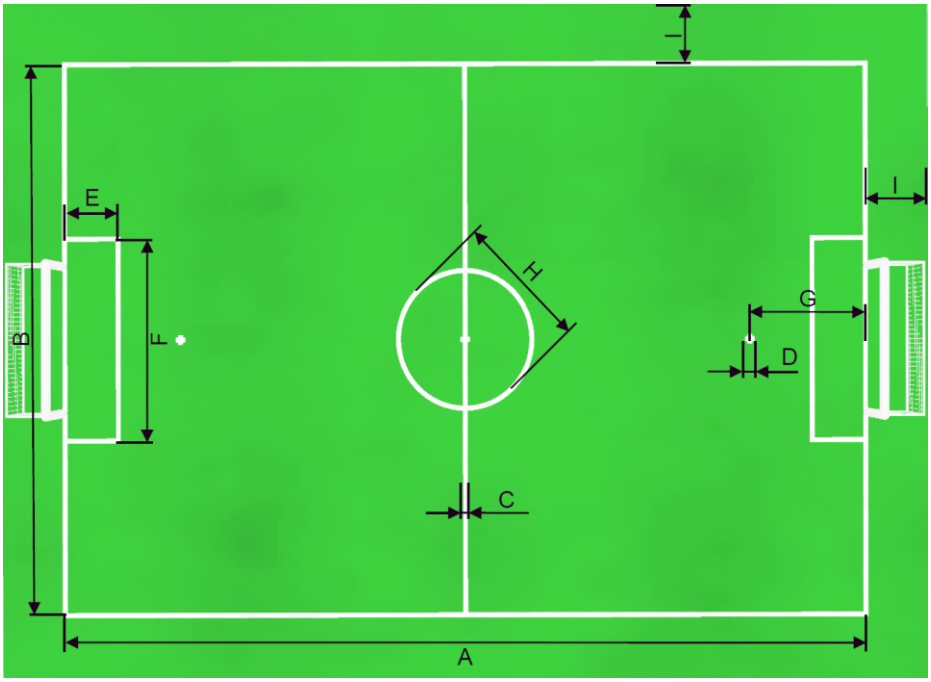
Beleuchtung: Ziel ist es, die SPL Spielfelder den natürlichen Lichtbedingungen anzupassen. Wenn möglich werden die Spielfelder daher im Freien oder an Fenstern aufgebaut. Für den Wettbewerb soll der Veranstalter Sorge tragen, dass die meisten Bereiche des Feldes wenigstens zu 300 Lux ausgeleuchtet sind. Die Beleuchtung des gesamten Spielfeldes muss nicht gleichmäßig sein.

[30] Alle beschriebenen Regeln und Abbildungen des Abschnitts unter Veränderung dem offiziellen RoboCup Standard Platform League (NAO) Rule Book entnommen. Die vorläufige Version für 2017 ist zu finden unter http://www.tzi.de/spl/pub/Website/Downloads/Rules2017.pdf, abgerufen am 22.01.2017

Ball: Der offizielle Spielball ist ein weicher Schaumstoffball von 44g Gewicht und 100mm Durchmesser. Optisch ist der Ball einem weißen Fußball mit schwarzen Flicken nachempfunden (vgl. Abbildung 4.3 und Abbildung 5.2.)

Roboter: Eingesetzt werden dürfen nur die von Aldebaran Robotics (mittlerweile Softbank Gruppe, Tokio) produzierte Nao Roboter (vgl. Anlage A.2). Keinerlei Modifikationen der Roboter, ihrer Hardware wie Sensorik, ist zulässig. Die passiven Handgelenke dürfen festgemacht und die Finger mit weißem oder durchsichtigem Klebeband geschützt werden.

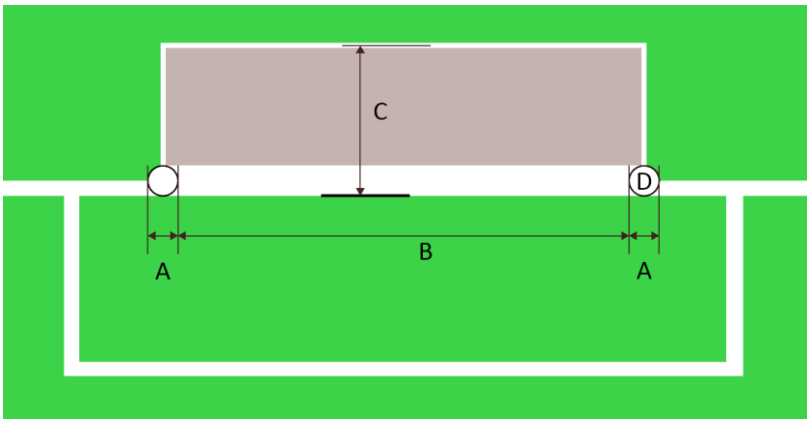
Spielfeld:



Maße (in mm)					
ID	Bezeichnung	Länge	ID	Bezeichnung	Länge
A	Feldlänge	9000	F	Strafraumbreite	2200
B	Feldbreite	6000	G	Elfmeterkreuzdistanz	1300
C	Linienbreite	50	H	Mittelkreisdurchmesser	1500
D	Elfmeterkreuzgröße	100	I	Grenzstreifenbreite	700
E	Strafraumlänge	600			

Abbildung A.4 *Schematische Darstellung des Spielfeldes (nicht maßstabsgetreu)*

Tor:



Maße (Draufsicht, in mm)					
ID	Bezeichnung	Länge	ID	Bezeichnung	Länge
A	Torpfostenbreite	100	C	Tortiefe	500
B	Torbreite	1500	D	Torhöhe	800

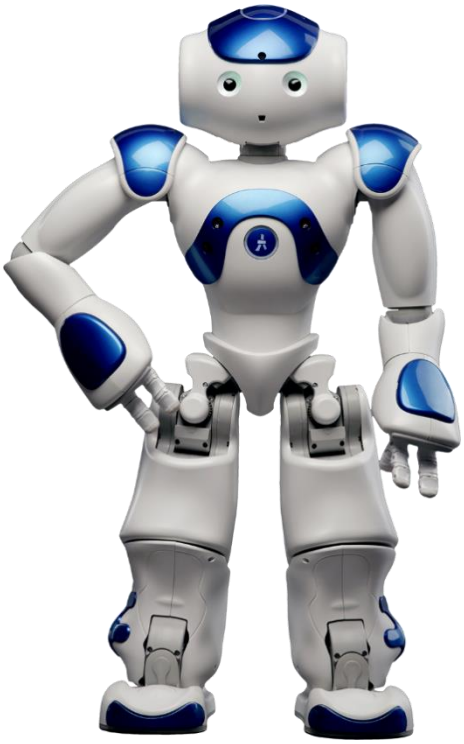
Abbildung A.5: *Draufsicht auf den Strafraum mit Maßen des Tores*

A.2 Nao V5

Der Roboter Nao dient der SPL als exklusive Roboter Plattform. Der von Aldebaran Robotics entwickelte humanoide Roboter existiert mittlerweile in fünf Versionen. Die technischen Angaben in der folgenden Tabelle sind der offiziellen Dokumentation[31] zum fünften Modell entnommen. Auch ältere Versionen dürfen in der SPL eingesetzt werden.

Nao V5

[32]



Betriebssystem	
Name	NAOqi (Embedded Linux)
Version	2.0

Prozessor	
Model	Intel Atom Z530
Architektur	Silverthorne, x86
Takt	1,6 Ghz 533 MHz FSB
Kerne	1
Cache	512 kB

Speicher	
Arbeitsspeicher	1 GB
Hauptspeicher	2 GB (Flash) 8 GB (SDHC)

Netzwerk	
LAN	RJ45 10/100/1000 base T
WLAN	802.11a/b/g/n

Maße	
Dimensionen	573mm × 311mm × 275mm (Höhe × Tiefe × Breite)
Gewicht	5,4kg
Freiheitsgrade	25
Akku	Lithium Ionen (48,6 Wh, 60-90min Betrieb)

Sensoren	
Kameras	2 á 1280 Pixel × 960 Pixel, 30 Bilder/s 60,9° horizontaler Öffnungswinkel 47,6° vertikaler Öffnungswinkel 30cm bis unendlicher Fokusbereich
Mikrofone	4 (20 mV/Pa ± 3dB Sensitivität, 150Hz bis 12kHz Frequenzbereich)

[31] http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html, abgerufen am 22.01.2017

[32] Bild des Roboters von https://www.ald.softbankrobotics.com/en/press/gallery/nao, abgerufen am 22.01.2017

Infrarot	2 (940nm Wellenlänge, ± 60° Öffnungswinkel)
Ultraschall	2 × 2 (1cm – 4cm Auflösung, 40kHz Frequenz, 60° Öffnungswinkel)
Kraftsensoren	2 × 4 (4 je Fuß, 0N - 25N Sensitivität)
Inertialsensoren	3-Achsen Gyrometer (~500°/s Winkelgeschwindigkeit) 3-Achsen Beschleunigungssensor (~2g Beschleunigung)
Taktile Sensoren	9 (3 Kopf, 3 je Hand)
Propriozeptive Sensoren	Wahrnehmung der eigenen Gelenkstellung per Hall-Sensoren (ca. 0,1° Genauigkeit)

Interaktion	
Lautsprecher	2 (1 je Ohr)
Knöpfe	3 (1 Brust, 1 je Fuß)
LEDs	51 (12 Kopf, 8 je Auge, 10 je Ohr, 1 Brust, 1 je Fuß)
Akku	Lithium Ionen (48,6 Wh, 60-90min Betrieb)

A.3 Spezifikationen des Testcomputers

Betriebssystem	
Name	Microsoft Windows 10 Professional (x64)
Version	Build 10240.17236

Hauptplatine	
Model	ASRock Z77 Extreme4
Chipsatz	Intel Z77 (Panther Point DH)
Steckplätze	2 PCI, 5 PCI Express x1, 2 PCI Express x16
PCI Express Version	3.0
USB Version	2.0/3.0

Prozessor	
Model	Intel Core i7-3770
Architektur	Ivy Bridge
Takt	3,4 GHz 3,8 GHz (Turbo)
Kerne	4 8 (Logisch)
Cache	L1: Instruktionen: 4 × 32 kB, Daten: 4 × 32 kB L2: Integriert: 4 × 256 kB L3: 8 MB

Speicher	
Arbeitsspeicher	12 GB, 800 MHz
Hauptspeicher	1 TB, 108/156 MB/s (SATA 3.0) 180 GB, 450/500 MB/s (SSD)

Grafikkarte[33]	
Model	Radeon R9 390X
Takt	1050 Mhz
Speicher	8 GB (GDDR5)

Netzwerk	
LAN	PCI-E Gigabit Ethernet Controller, 10/100/1000 base T
WLAN	PCI-E 802.11b/g/n

[33] Die Grafikkarte wurde in den Experimenten nicht von der Software angesprochen und wird hier nur aus Gründen der Vollständigkeit aufgeführt.

A.4 Exemplarische Detektionen

Die folgenden Aufnahmen sind im Rahmen der empirischen Auswertung entstanden. Die Aufnahmen illustrieren Detektionsergebnisse in interessanten Spielsituationen.

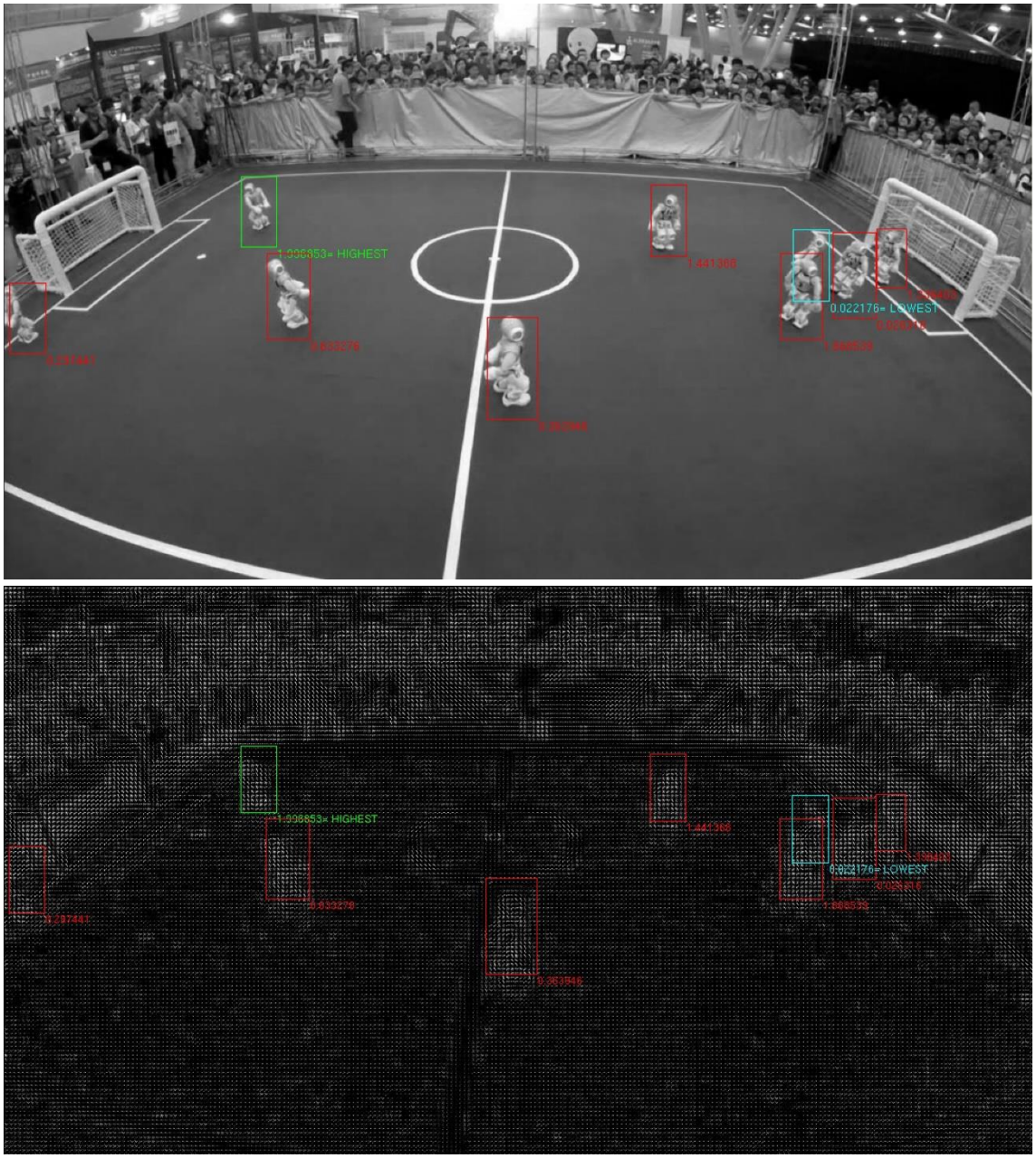


Abbildung A.6: *Zwei exemplarische Detektionen. Oben: Detektionen aus einem Video des RoboCup 2015. Der Klassifikator war der fHOG Sieger im* $F_{0,5}$*-Maß. Der Detektor schneidet in diesen Situationen sehr gut ab, hier werden alle Roboter korrekt detektiert. Unten: Illustration der HOG-Merkmale in dieser Szene. Auch in dieser Darstellung sind Spielfeld, Tore und Roboter erkennbar.*

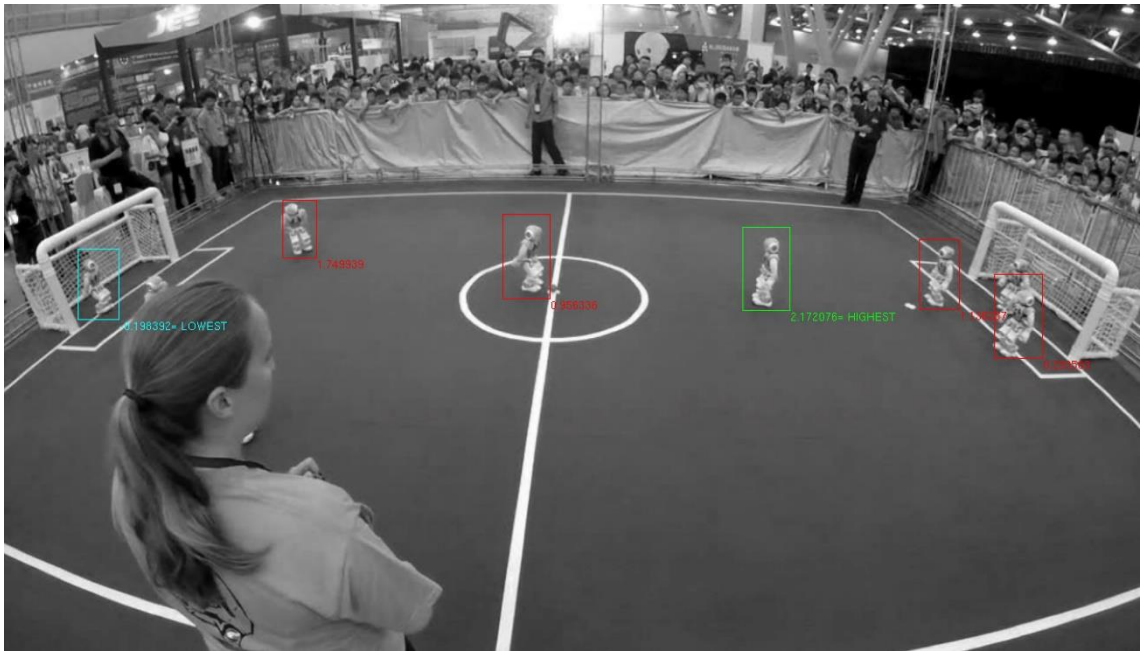


Abbildung A.7: *Eine weitere Aufnahme vom RoboCup 2015 illustriert Fälle in denen Roboter fälschlicherweise nicht erkannt werden. Die falsch negativen Fälle sind der Roboter hinter dem Kopf der Schiedsrichterin und der halb verdeckte Torwart rechts im Tor.*

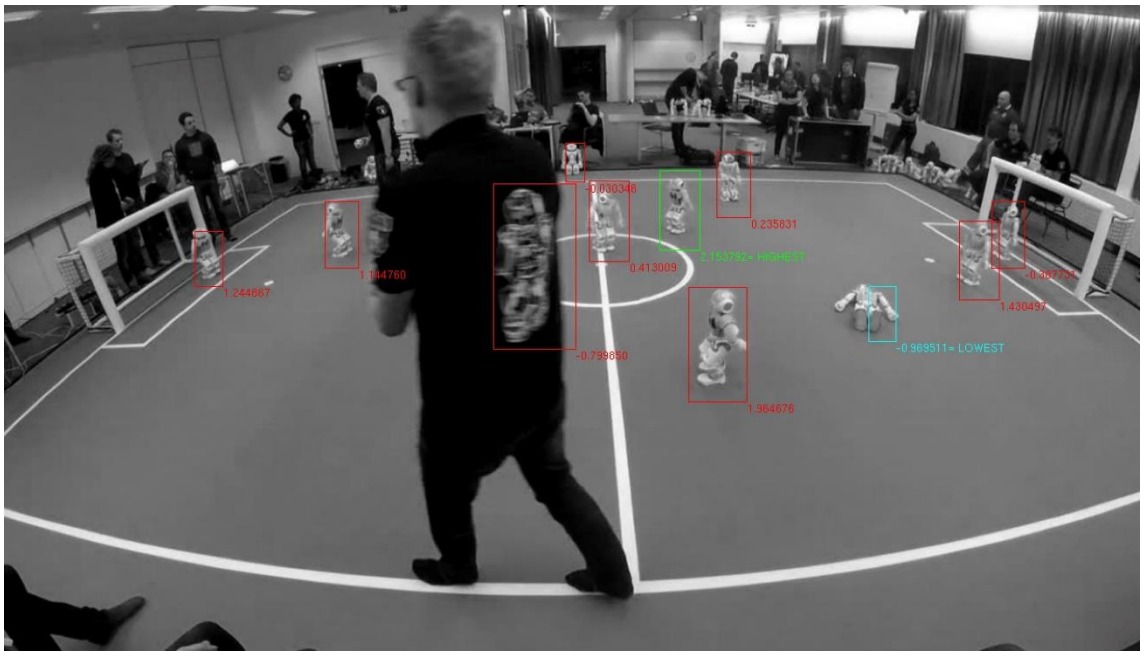


Abbildung A.8: *Diese Aufnahme von der RoboCup EuropeanOpen 2016 zeigt drei falsch positive Detektionen. Erstens wird der Roboter auf dem T-Shirt des Schiedsrichters erkannt. Zweitens wird einer der inaktiven Roboter, der außerhalb des Spielfeldes auf Höhe der Mittellinie sitzt, erkannt. Drittens wird der umgefallene Roboter in der Mitte rechts detektiert. Solche Detektionen treten gehäuft auf, wenn die Sensitivität erhöht wird. Der Umgang mit inaktiven Robotern in der Auswertung der Detektoren wird in Abschnitt 5.5 diskutiert.*

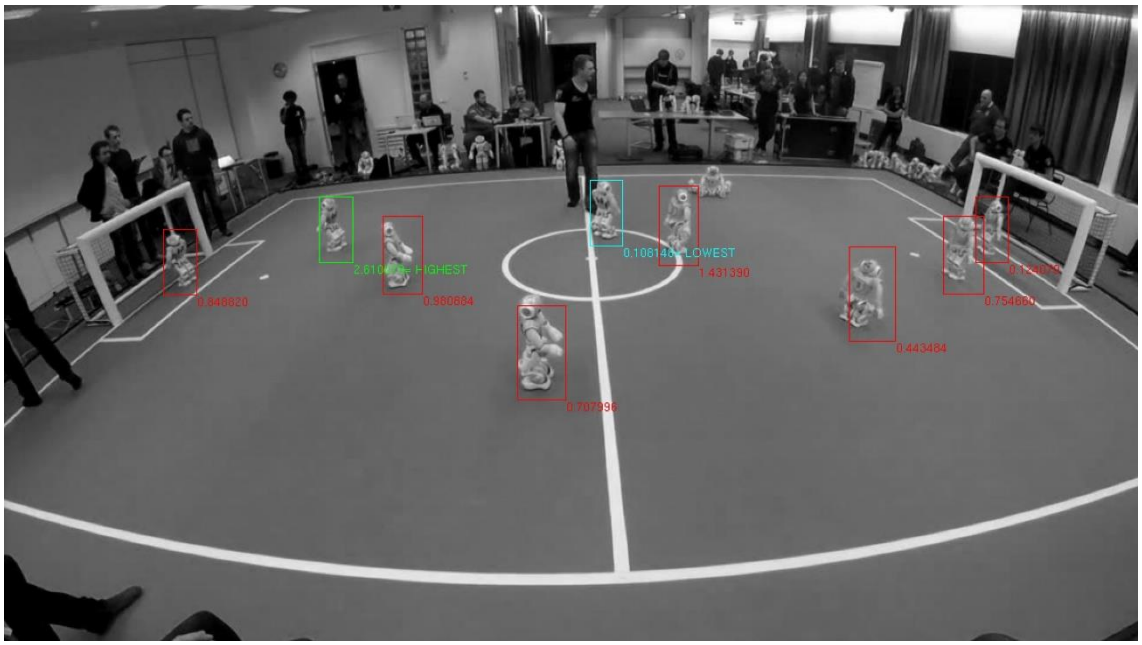


Abbildung A.9: *Fehlerfreie Detektion aus dem Video der RoboCup EuropeanOpen 2016.*

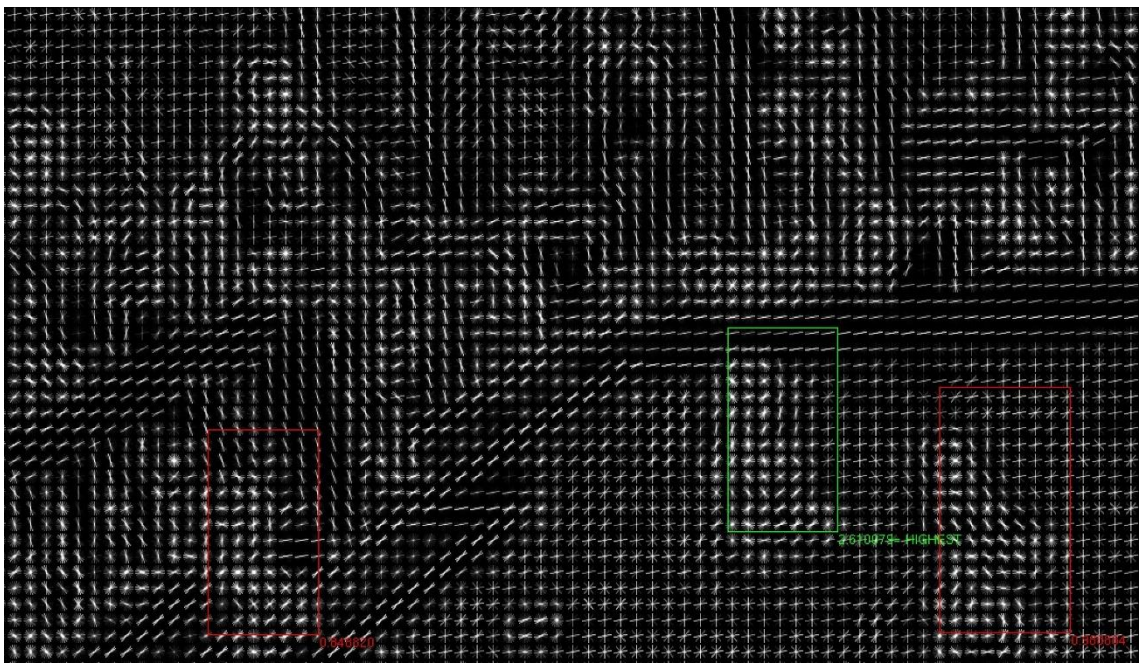


Abbildung A.10: *Vergrößerung des linken Strafraums der in Abbildung A.9 dargestellten Szene. In dieser Visualisierung lassen sich die Kopf- und Schulterpartien der detektierten Roboter in den HOG-Merkmalen ausmachen. Am oberen Rand zeigen sich die Zuschauer mit ähnlichen Merkmalen, weshalb in diesen Bereichen vermehrt Fehldetektionen auftreten können.*

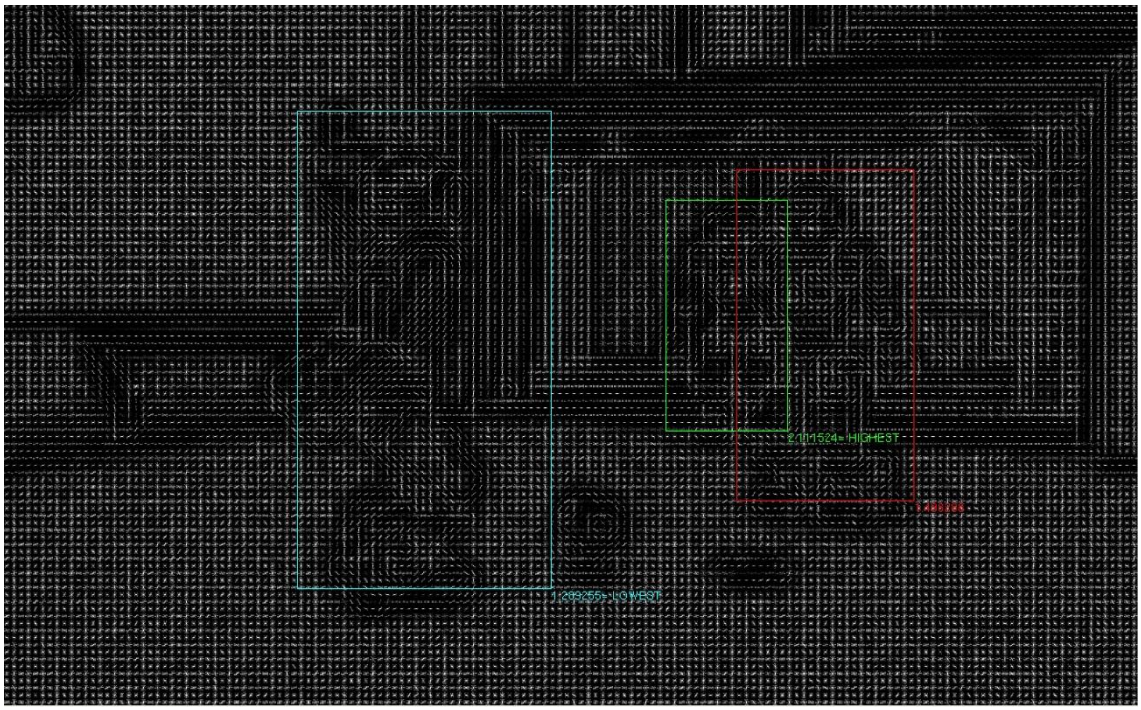


Abbildung A.11: *Visualisierung der HOG-Merkmale der in Abbildung 5.2 unten rechts gezeigten Aufnahme aus der Ich-Perspektive des Roboters. Die Roboter nehmen große Teile des Bildes ein, selbst der verdeckte Torwart (grün) hat noch stark ausgeprägte Merkmale. Auch der Ball, Elfmeterpunkt und das Tor lassen sich augenscheinlich ausmachen.*

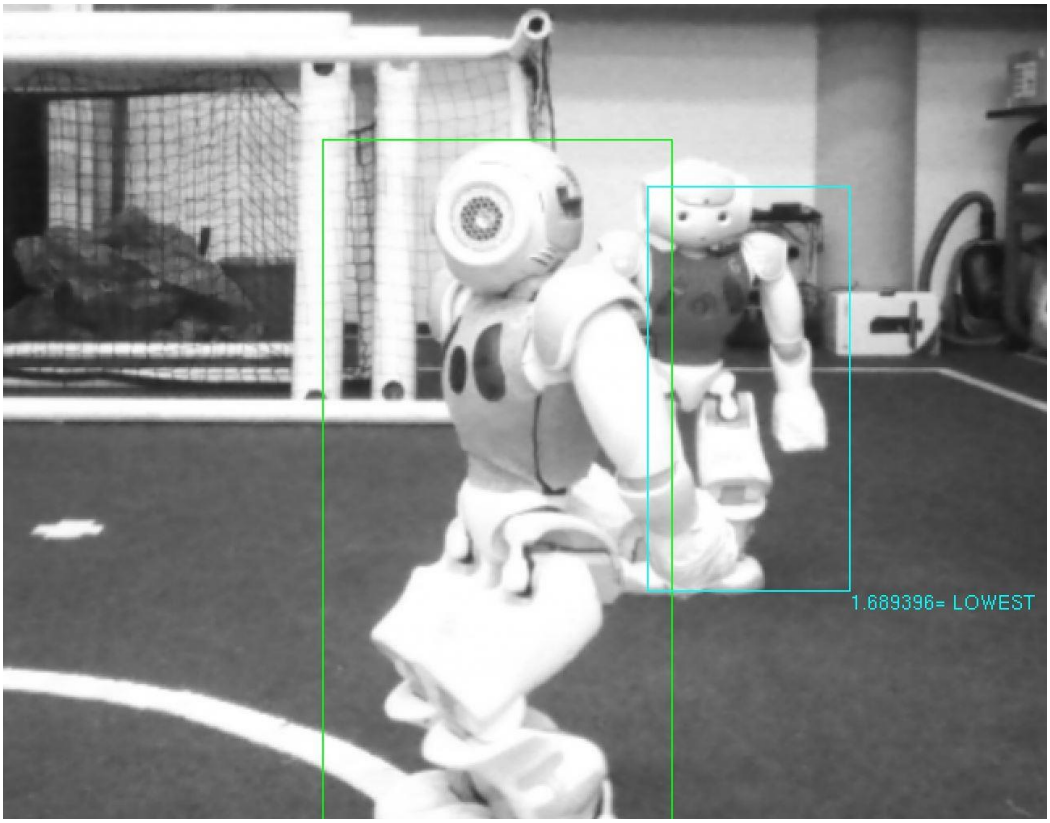


Abbildung A.12: *Eine weitere Aufnahme aus Ich-Perspektive des Roboters. Hier steht ein Roboter direkt im Sichtfeld, trotzdem werden beide Roboter korrekt detektiert.*

A.5 Weitere Heatmaps

In diesem Abschnitt werden die fehlenden Heatmaps dargestellt, die zu den in Kapitel 5.2 und 5.3 beschriebenen Experimenten angefertigt wurden.

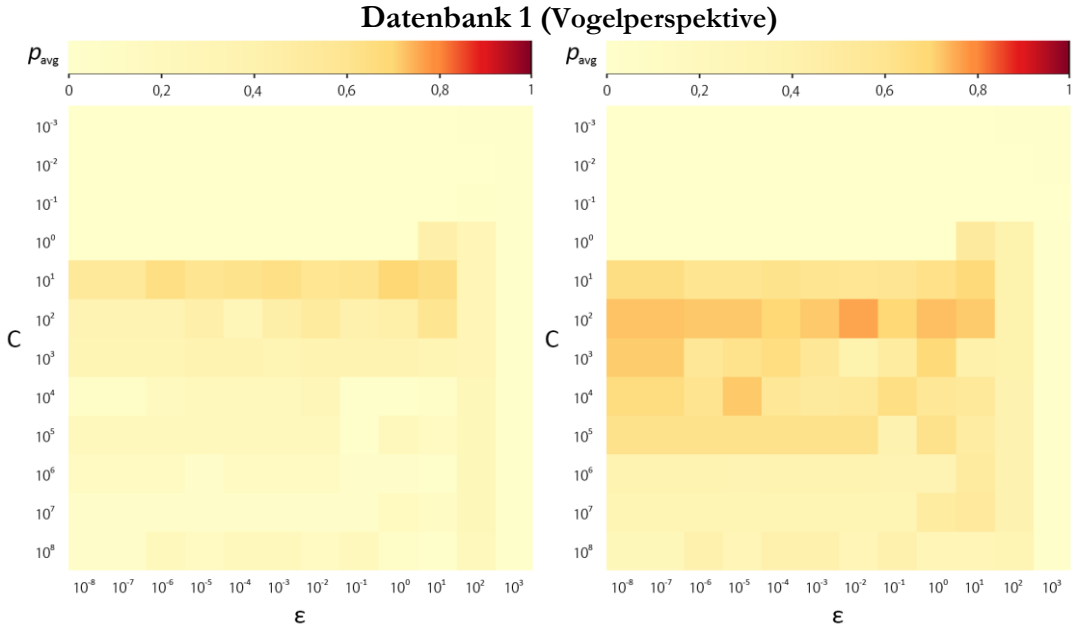


Abbildung A.13: *Heatmaps der durchschnittlichen Präzision* p_{avg} *auf der Datenbank 1 der rfHOG Detektoren. Links: Durchschnittliche Präzision* p_{avg} *auf der Datenbank 1 bei starker Reduktion, Rechts: Durchschnittliche Präzision* p_{avg} *auf der Datenbank 1 bei moderater Reduktion*

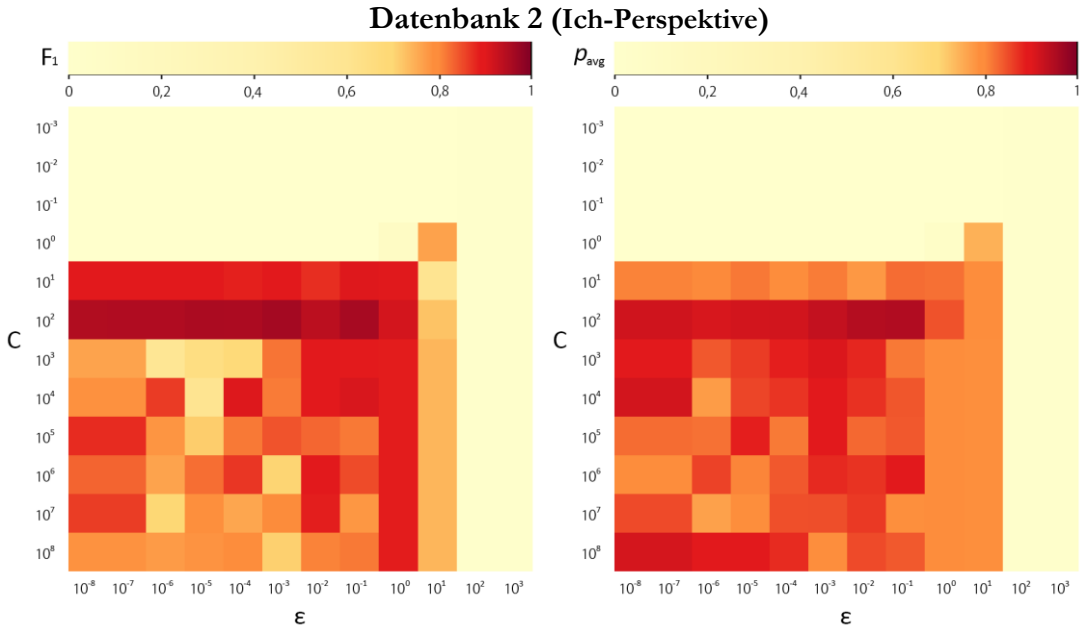


Abbildung A.14: *Heatmaps bei starker Reduktion auf der Datenbank 2. Links: Durchschnittliches* F_1*-Maß auf der Datenbank 2 bei starker Reduktion, Rechts: Durchschnittliche Präzision* p_{avg} *auf der Datenbank 2 bei starker Reduktion*

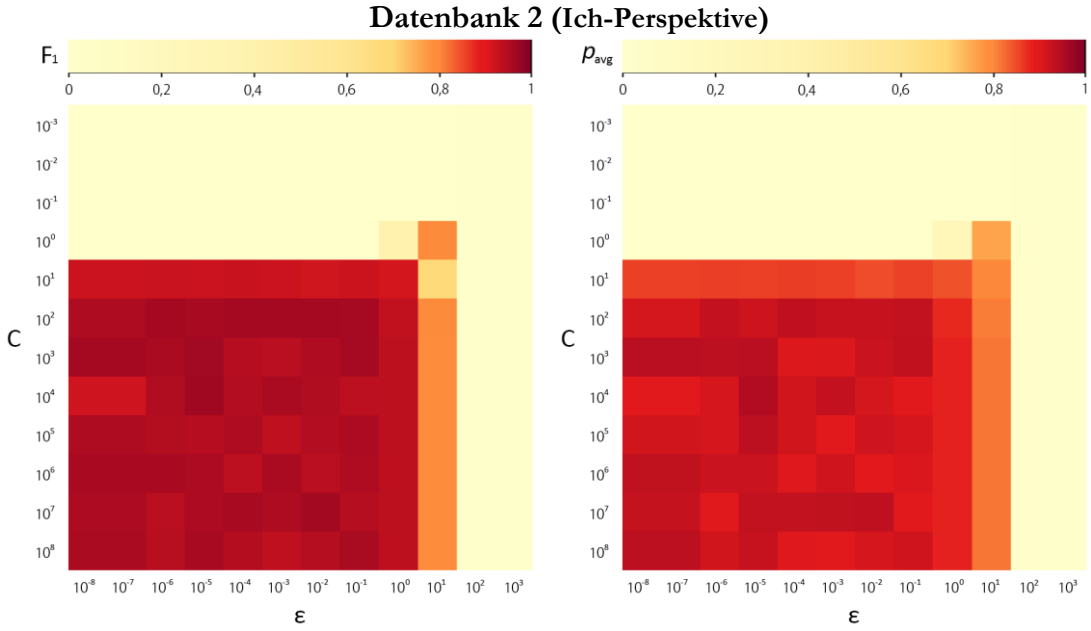


Abbildung A.15: *Heatmaps bei moderater Reduktion auf der Datenbank 2. Links: Durchschnittliches F_1-Maß auf der Datenbank 2 bei moderater Reduktion, Rechts: Durchschnittliche Präzision p_{avg} auf der Datenbank 2 bei moderater Reduktion*

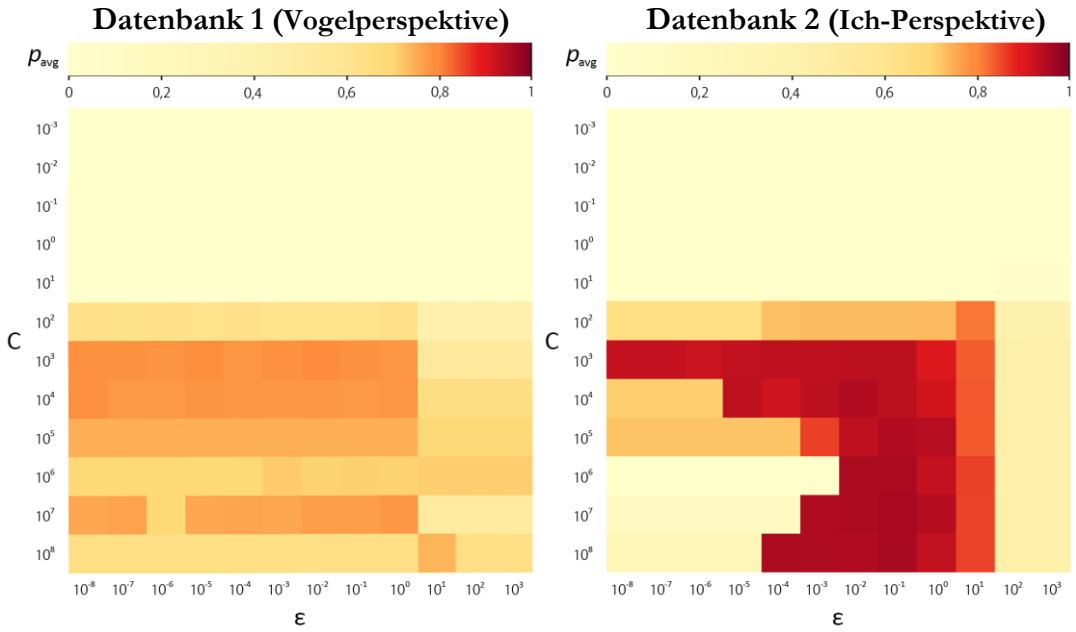


Abbildung A.16: *Heatmaps der durchschnittlichen Präzision p_{avg} bei NNR. Links: Durchschnittliche Präzision p_{avg} auf der Datenbank 1 bei NNR, Rechts: Durchschnittliche Präzision p_{avg} auf der Datenbank 2 bei NNR*

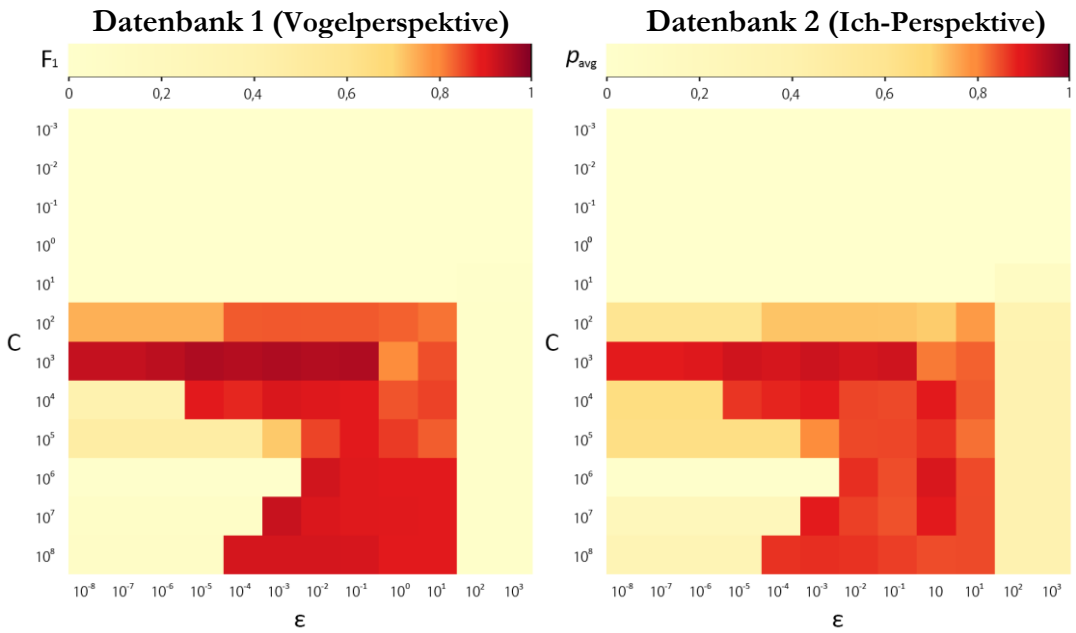


Abbildung A.17: *Heatmaps auf Bildern der Größe* $640 \times 480\ Pixel$ *bei NNR. Links: Durchschnittliches* F_1*-Maß auf der Datenbank 2, Rechts: Durchschnittliche Präzision* p_{avg} *auf der Datenbank 2*

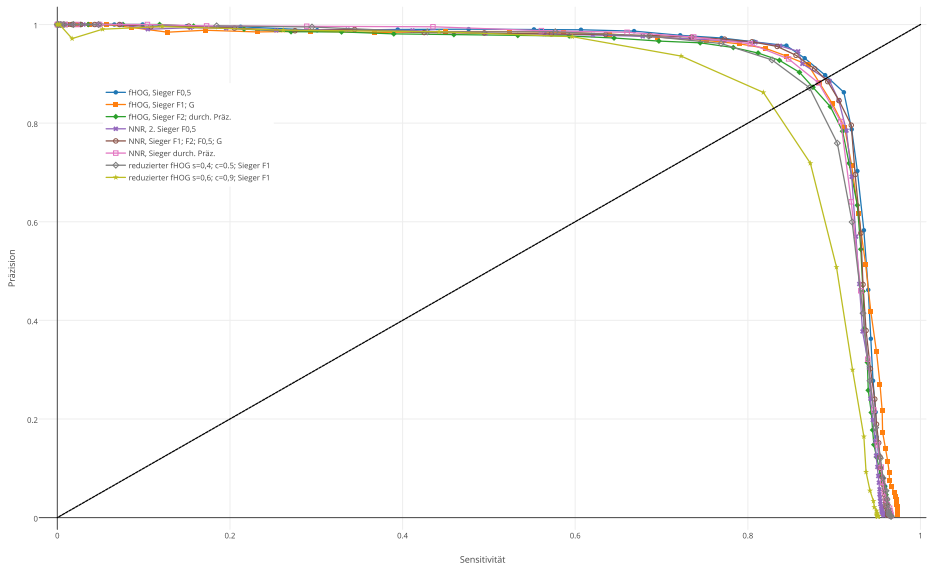


Abbildung A.18*: Vollständige PR-Kurven der auf Datensatz 1 am besten abschneidenden Detektoren.*

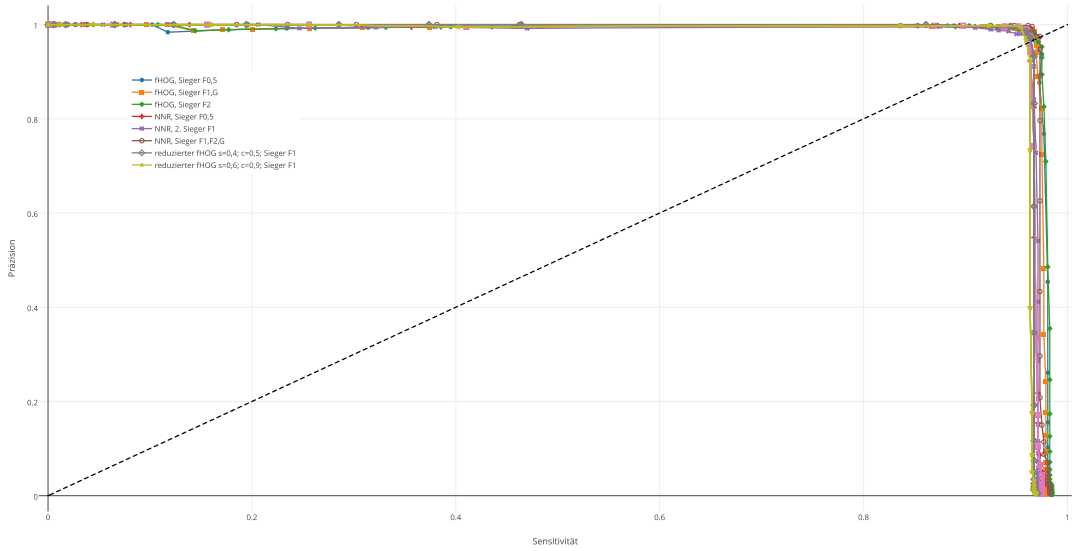


Abbildung A.19: *Vollständige PR-Kurven der auf Datensatz 2 am besten abschneidenden Detektoren.*

Literaturverzeichnis

1. Estivill-Castro, V.; Radev, J. *Humanoids Learning who are Teammates and who are Opponents.* in The 8th Workshop on Humanoid Soccer Robots 13th at IEEE-RAS International Conference on Humanoid Robots. 2013.
2. Siegwart, R.; Nourbakhsh, I.R.; Scaramuzza, D. *Introduction to autonomous mobile robots*. 2011. MIT press.
3. Dollar, P.; et al. *Pedestrian Detection: An Evaluation of the State of the Art.* Pattern Analysis and Machine Intelligence, IEEE Transactions. 2012. 34(4): p. 743-761.
4. Benenson, R.; et al. *Ten years of pedestrian detection, what have we learned?* in *European Conference on Computer Vision*. 2014. Springer.
5. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D. *Cascade object detection with deformable part models*. in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference*. 2010.
6. Felzenszwalb, P.F.; et al. *Object detection with discriminatively trained part-based models.* IEEE transactions on pattern analysis and machine intelligence, 2010. 32(9): p. 1627-1645.
7. Dalal, N.; Triggs, B. *Histograms of oriented gradients for human detection*. in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference*. 2005.
8. Zhu, Q.; et al. *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients*. in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*. 2006.
9. Overett, G.; Petersson, L.; Andersson, L.; Pettersson, N. *Boosting a heterogeneous pool of fast hog features for pedestrian and sign detection*. in *Intelligent Vehicles Symposium*. 2009. IEEE.
10. Overett, G.; Petersson, L. *Fast features for time constrained object detection*. in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2009. IEEE.
11. Wang, Q.; Zhang, R.B. *LPP-HOG: A new local image descriptor for fast human detection*. in *Knowledge Acquisition and Modeling Workshop, 2008. KAM Workshop 2008. IEEE International Symposium on*. 2008. IEEE.
12. Villamizar, M.; et al. *Combining color-based invariant gradient detector with HoG descriptors for robust image detection in scenes under cast shadows*. in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. 2009. IEEE.
13. Wang, X.; Han, T.; Yan, S. *An HOG-LBP human detector with partial occlusion handling.* in *2009 IEEE 12th International Conference on Computer Vision*. 2009. IEEE.

14. Gutierrez-Osuna, R. *Validation.* Intelligent Sensor Systems, 2002.

15. Hsu, C.; Chang C.; Lin C. *A practical guide to support vector classification.* 2003.

16. Marchant, R.; Guerrero, P.; Ruiz-del-Solar, J. *Cooperative Global Tracking Using Multiple Sensors*, in *RoboCup 2012: Robot Soccer World Cup XVI*. 2013. Springer. p. 310-321.

17. Farazi, H.; Behnke, S. *Real-Time Visual Tracking and Identification for a Team of Homogeneous Humanoid Robots.* 2016.

18. Canny, J. *A Computational Approach to Edge Detection*. 1986.

19. Moravec, H.P. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.* 1980.

20. Förstner, W.; Gülch, E. *A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features*. in *Proceedings of the ISPRS Intercommission Workshop on Fast Processing of Photogrammetric Data*. 1987.

21. Lindeberg, T. *Scale Selection Properties of Generalized Scale-Space Interest Point Detectors.* Journal of Mathematical Imaging and Vision, 2012. 46(2): p. 177-210.

22. Lowe, D.G. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, 2004. 60(2): p. 91-110.

23. Lowe, D.G. *Object recognition from local scale-invariant features*. in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference*. 1999.

24. Harris, C.; Stephens, M. *A combined corner and edge detector*. in *In Proc. of Fourth Alvey Vision Conference*. 1988.

25. Shi, J.; Tomaso. C. *Good Features to Track*. in *9th IEEE Conference on Computer Vision and Pattern Recognition*. 9th IEEE Conference on Computer Vision and Pattern Recognition. 1994 Springer.

26. Witkin, A. *Scale-space filtering*, in *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2*. 1983, Morgan Kaufmann Publishers Inc.: Karlsruhe, West Germany. p. 1019-1022.

27. Koenderink, J. *The structure of images.* Biological cybernetics, 1984**.** 50(5): p. 363-370.

28. Lindeberg, T. *Scale-space theory: a basic tool for analyzing structures at different scales.* Journal of Applied Statistics, 1994. 21(1-2): p. 225-270.

29. Brown, M.; Lowe, D.G. *Invariant Features from Interest Point Groups*. *BMVC*. 2002.

30. Viola, P.; Jones, M. *Rapid object detection using a boosted cascade of simple features*. in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. 2001.

31. Greenhalgh, J.; Mirmehdi, M. *Real-Time Detection and Recognition of Road Traffic Signs.* Intelligent Transportation Systems, IEEE Transactions on, 2012. 13(4): p. 1498-1506.

32. Freund, Y.; Schapire, R.E. *A desicion-theoretic generalization of on-line learning and an application to boosting.* in *Computational learning theory*. 1995. Springer.

33. Kohavi, J., *A Study of Cross-Validation and Bootstrap.* 1995.

34. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning.* 2008: p. 106-113.

35. King, D., *Max-margin object detection.* arXiv preprint arXiv:1502.00046, 2015.

36. Hastie, T; et al. *The entire regularization path for the support vector machine.* Journal of Machine Learning Research, 2004. 5(Oct): p. 1391-1415.

37. Lazebnik, S.; Schmid, C.; Ponce, J. *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.* in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006. IEEE.

38. Thorsten, J.; Thomas, F.; Yu, C. *Cutting-plane training of structural SVMs.* Machine Learning, 2009. **77**(1): p. 27-59.

39. King, D. *Dlib-ml: A machine learning toolkit.* Journal of Machine Learning Research, 2009. 10(Jul): p. 1755-1758.

40. Dalal, N.; Triggs, B. *INRIA Person Dataset.* 2005.

41. Papageorgiou, C.; Poggio, T. *A Trainable System for Object Detection.* International Journal of Computer Vision, 2000. 38(1): p. 15-33.

42. Kalra, P.; Peleg, S. *Computer Vision, Graphics and Image Processing: 5th Indian Conference, ICVGIP 2006, Madurai, India, December 13-16, 2006, Proceedings*. 2007: Springer Berlin Heidelberg.

43. Chen, C.H.; Chen, M.; Gao, T. *Detection and Recognition of Alert Traffic Signs.* Publication of Standford University, 2008.

44. Danelljan, M.; et al. *Accurate scale estimation for robust visual tracking.* in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press.

45. Mellmann, H.; Schlotter, B.; Blum, C. *Simulation Based Selection of Actions for a Humanoid Soccer-Robot.* RoboCup 2016: Robot Soccer World Cup XX, 2016.

Abbildungsverzeichnis

Abkürzungsverzeichnis

AdaBoost	Adaptive Beschleunigung
BSD	Berkeley Software Distribution
BoW	Bag-of-visual-words
DOG	Difference of Gaussians
DPM	Deformable Parts Model
fHOG	HOG nach P. Felzenszwalb
HOG	Histograms of Oriented Gradients
LOG	Laplacian of Gaussian
LPP	lokalitätserhaltene Projektion
MMOD	Max-Margin Object Detection
NaoTH	Nao Team Humboldt
NNR	nukleare Norm Regularisierung
RBF	radiale Basisfunktion
rfHOG	reduzierter fHOG
SIFT	Scale-invariant feature transform
SPL	Standard Platform League
SVM	Support Vector Machine

Danksagung

Ich hatte das große Glück früh in meinem Hauptstudium eine *Einführung in die Künstliche Intelligenz* zu belegen. Die Begeisterung mit der Prof. Dr. Hans-Dieter Burkhard die Themenfelder der Künstlichen Intelligenz und allem voran der Robotik vermittelt, ist ansteckend. Folgerichtig musste ich danach jede Vorlesung und jeden Kurs besuchen, die es mir ermöglichten, meinem Interesse an diesem Betätigungsfeld nachzugehen.

So begegnete ich auch Prof. Dr. Verena Hafner, die mir ihre nicht minder inspirierenden Arbeiten zur Embodied AI in den wunderbaren *ShanghAI Lectures* nahebringen konnte. An dieser Stelle möchte ich Ihnen beiden ausdrücklich danken für alles vermittelte Wissen, für jede investierte Stunde in meine Person und dafür, dass sie ein gleichsam motivierendes, wie stimulierendes Arbeitsumfeld geschaffen haben, das die Grundlage dieser Arbeit darstellt.

Es war mir ein besonderes Vergnügen zweimal dem Nao Team Humboldt beizutreten. Ich habe jeweils eine Gruppe hilfsbereiter, talentierter und engagierter Menschen kennen lernen dürfen, durch die das tägliche Arbeiten zu einem Vergnügen wurde. Ausdrücklichen Dank an alle Mitglieder, die mir über die Jahre mit Rat und Tat zur Seite standen: Christian Rekittke, Claas-Norman Ritter, Thomas Krause, Marcus Scheunemann, Paul Schütte, Yuan Xu, Florian Holzhauer, Kirill Yasinovskiy, Martin Schneider, Benjamin Schlotter, Steffen Kaden, Tobias Hübner, Peter Woltersdorf, Robert Martin und Philipp Strobel.

Ich möchte in ganz besonderem Maße Heinrich Mellmann danken. Ohne dich wäre diese Arbeit nicht entstanden. Du bist nicht nur das Herz der Arbeitsgruppe, sondern wichtiger Lehrer, Diskussionspartner und Freund. Deine Geduld und dein Engagement sind inspirierend und keinem Menschen hören ich gebannter zu als dir, wenn aus einem kleinen Denkanstoß in Sachen Robotik oder Programmierung ein wissenschaftlicher Diskurs wird.

Frauke Fuchs, ich möchte dir danken für deine Freundschaft, für alle wichtigen Hinweise und für das Entfernen von vielen, vielen Fehlern.

Ich danke auch der Familie Szymansky, eure Liebe und Hilfsbereitschaft waren für die Fertigstellung der Arbeit unabdingbar. Annabell, du bist nicht nur die strengste Lektorin, sondern auch die großartigste Wissenschaftlerin und Person, die mir in meinem Leben begegnet ist. Du bist meine beste Freundin und mein größtes Glück, vielen Dank für deine Unterstützung.

Der wichtigste Dank geht an meine Familie, an meine Eltern und an meinen Bruder. Ohne euch hätte diese Arbeit nicht entstehen können. Eure Liebe und Unterstützung sind die Basis meines Schaffens. Ich danke Euch.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 26.02.2017

……

Dominik Krienelke