Oberkörperdetektion mittels Histogram of Oriented Gradients und Haar-Feature Kaskaden

Studienarbeit

Institut für theoretische Informatik
Prof. A. Waibel

Falkutät für Informatik
Universität Karlsruhe (TH)

von

Ngoc Thang Vu

20. Dezember 2008

Betreuer:

Dipl.-Inf. Keni Bernardin
Dr.-Ing. Rainer Stiefelhagen
Prof. Dr. Alex Waibel

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen verwendet zu haben.

Karlsruhe, 20. Dezember 2008 .....

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Die heutige Gesellschaft wird immer stärker durch die Entwicklung der Rechentechnik beeinflusst. Viele Anwendungen wie intelligente Räume oder Überwachungsysteme, bei denen Menschen im Mittelpunkt stehen, werden Tag für Tag bekannter. Deshalb entsteht die Anforderung, menschliche Struktur nicht nur in Bilddaten, sondern auch in Bildsequenzen zu erkennen und als Eingabe für weitere Aufgaben zu nutzen. Dabei existieren bereits heute zahlreiche Anwendungen wie Smartrooms, Zugangskontrolle, Kameraüberwachungsystem usw. Während in einem Smartroom die Menschen bei der Arbeit oder zu Hause unterstüzt werden sollen, übernimmt die Kameraüberwachungsystem die Kontrolle.
In allen oben genannten Systemen ist eine erste Anforderung, die Menschen mit Hilfe von Kameras und Rechnern zu detektieren. Die Frage ist, welche Informationen der Menschen in solchen Anwendungen erforderlich sind und deshalb erkannt werden sollen. Während bei einer Software für Geschlechtserkennung das Gesicht der Menschen ausreichend ist, wird für eine Bewegungsanalyse der ganze Körper benötigt. In anderen Anwendungen, beispielweise die Gestenerkennung, sind nur die Hände wichtig und werden deshalb detektiert. In unserem Fall ist es wichtig zu erkennen, ob Menschen in einem intelligeten Raum, einem sogenannten Smartroom sind und wo sie sich befinden. Während der ganze Körper schwierig zu sehen ist, da die Beine oft durch Tische, Stühle, etc. verdeckt sind, ist der Oberkörper meistens sichtbar. Aus diesen Gründen ist der sich frei bewegende Oberkörper interessant und sollte deshalb detektiert werden. Eine mögliche Szene, in der Oberkörper erkannt werden sollen, ist in Abbildung 1.1 zu sehen.

1.2 Ziel der Arbeit

Ziel dieser Studienarbeit ist es, durch verschiedene Experimente einen relativen Vergleich der Leistung und Geschwindigkeit zweier bekannter Detektionsansätze durchzuführen. Der eine Ansatz wurde von Paul Viola und Michael Jones im Jahr 2001 veröffentlicht und ist im Bereich Objekt- und Gesichtserkennung bekannt geworden. Der Vorteil dieses Verfahrens ist die kurze Detektionszeit. Dafür wird

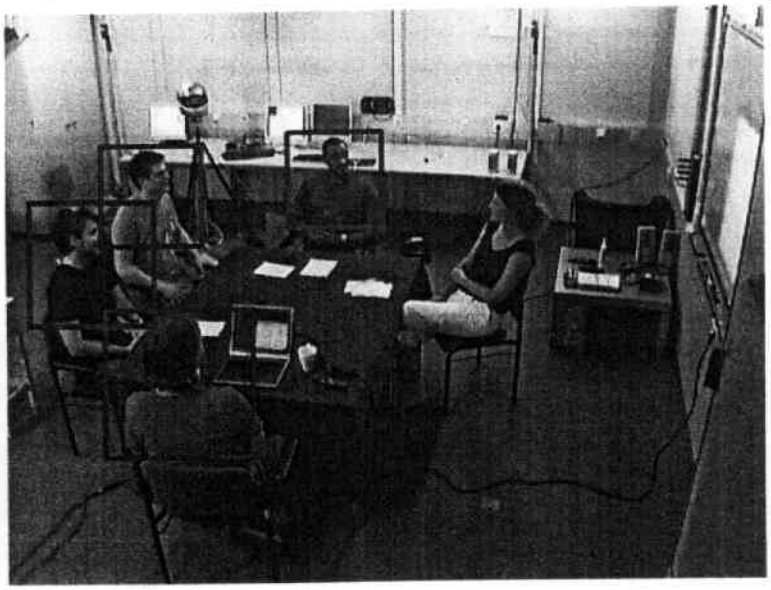

Abb. 1.1: Eine mögliche Szene im Smartroom mit detektierten Oberkörpern

allerdings eine sehr lange Trainingszeit benötigt und es kann nur sinvoll eingesetzt werden, falls das zu detektierende Muster eindeutig ist. Das heißt, dass es unmöglich ist, Objekte mit verschiedener Orientierung oder variabler Erscheinung mit einem einzigen Detektor zu detektieren. Der andere Ansatz wurde von Dalal und Triggs im Jahr 2005 erfunden und funktioniert hervorragend für den gesamten Körper. So wie bei Viola&Jones ist dieses Verfahren robust und funktioniert nicht nur für die Detektion von Menschen, sondern auch für andere Objekte und Tiere. Aber auf der anderen Seite ist die benötigte Detektionszeit vergleichsweise hoch. In dieser Arbeit werden auf diesen 2 Ansätzen basierende Systeme mit Oberkörpern von Bildsequenzen trainiert und evaluiert. Die Oberkörper in diesen Sequenzen bewegen sich frei und sind in verschiedenen Orientierungen zu sehen. Die Studienarbeit gliedert sich wie folgt. In Kaptitel 2 werden die grundlegenden Kenntnisse der zwei oben genannten Ansätze und die dazu benutzten Klassifikationverfahren erläutert. Das dritte Kapitel beschreibt die Vorbereitung der Daten. In Kapitel 4 wird das gesamte Training vorgestellt, die Detektion experimentell ausgewertet und dadurch ein relativer Vergleich zwischen den zwei Verfahren vorgenommen. Kapitel 5 fasst die gewonnen Erkenntnisse nochmal zusammen.

1.3 Verwandte Arbeiten

Der Ansatz von Paul Viola und Michael Jones [1] zur robusten Echtzeiterkennung kam in den letzten Jahren vielfach zur Anwendung. Die Arbeit von Viola und Jones basiert auf der Erkennung von Gesichtern aus der Frontalansicht. Neben der Erkennung von Gesichtern eignet sich dieser Ansatz auch zur Detektion beliebiger

Objekte[2]. Stan Z.Li und seine Kollegen [5] haben diesen Ansatz zur Erkennung von Gesichtern mit beliebiger Orientierung verwendet. Ihre Vorgehensweise erlaubt es, die Drehung des Kopfes mit einer Genauigkeit von bis zu 10 Grad zu bestimmen. Sie stellen auch ein Verfahren zur Reduktion der insgesamt verwendeten Merkmale vor. Lienhart und Kollegen haben die Merkmalstypen von Viola und Jones erweitert. Sie erhalten durch diese Merkmale einen kleineren und damit schnelleren Detektor, benötigen aber für die effiziente Berechnung zwei Vorverarbeitungsschritte [4]. Kobi Levi und Yair Weiss [6] erweitern den Detektor ebenfalls um einen neuen Merkmalstyp. Sie trainieren einen Detektor mit diesen Merkmalen für die Erkennung von Stühlen. Der Vorteil des Merkmalstyps besteht darin, dass für die Trainingsphase ein kleiner Datensatz genügt. Barreto und Kollegen [7] zeigen die Erkennung von Gesichtern und Händen für ein Interaktionsszenario mit einem Roboter. Für die Handerkennung verwenden sie einen auf dem Viola&Jones Ansatz basierten Detektor.
Im Verlgeich zu dem Ansatz von Viola und Jones wurde der Ansatz von Dalal und Triggs [9] erst im Jahr 2005 vorgestellt. Ihre Arbeit basiert auf der Erkennung des gesamten menschlichen Körpers. Daneben hat es auch für Objekte, wie Autos oder Motorräder und Tiere, wie Katzen richtig funktioniert. Qiang Zhu und Kollegen [10] haben ähnliche Merkmale wie beim Histogram Of Gradients Ansatz(kurz: HOG) von Dalal& Triggs verwendet, aber dafür zur Klassifikation ein kaskadenbasiertes System aufgebaut. Ausserdem wurde ein sogenanntes Intergralbild benutzt um ein schnelleres und besseres System zu erreichen. In der folgenden Ausarbeitung werden 2 Systeme mit angepassten Parametern zur Oberkörperdetektion trainiert und evaluiert. Eins basiert auf dem Ansatz von Dalal&Triggs und verwendet die Implementierung von INRIA Object Detection and Localization Toolkit [14]. Das andere wurde in OpenCV [12] implementiert und basiert auf dem Ansatz von Viola&Jones. Anhand der Ergebnisse von verschiedenen Experimenten wird ein Vergleich der Leistung und Geschwindigkeit dieser 2 Verfahren durchgeführt.

2 Grundlegende Konzepte

2.1 Viola&Jones Ansatz

2.1.1 Einführung

Um Objekte in Bildern oder Bildsequenzen detektieren zu können veröffentlichten Paul Viola und Michael Jones im Jahr 2001 einen Ansatz, der sehr schnell funktioniert und robust ist. Es werden dafür sehr schnell zu berechnende Merkmale in Kombination mit dem AdaBoost Lernalgorithmus verwendet. Mit Hilfe von einem sogenannten Integralbild wird die gesammte Detektionszeit deutlich verringert. Um die Leistung des Detektors zu verbessern wird eine Kaskade von verschiedenen hintereinander liegenden Detektoren zusammengebaut.

2.1.2 Merkmale

Die Objektdetektion basiert auf den Werten einfacher Merkmale. Es gibt viele Gründe dafür, warum man statt direkt Pixels zu nehmen, Merkmale verwendet, um Objekte zu detektieren. Der Hauptgrund liegt darin, dass mit bestimmten Trainingsdaten durch Merkmale mehr Ad-hoc Domänenwissen als durch Pixels erfasst werden kann. Zudem ergibt sich dadurch speziell bei diesem Ansatz ein bedeutender Geschwindigkeitsvorteil im Vergleich zu andereren pixelbasierten Systemen [1].
Vier verschiedene Arten von Merkmalen werden verwendet(siehe Abbildung 2.1). Die ersten zwei Mermale basieren auf 2 verschiedenen Rechtecken, die entweder horizontal oder vertikal benachbart sind. Es wird dabei für jedes Rechteck die Summe über alle Pixelwerte gebildet und die Differenz der Summen ergibt das Merkmalsergebniss. Im Vergleich dazu basiert das dritte Merkmal auf 3 verschiedenen benachbarten Rechtecken. Die Werte der zwei äusseren Rechtecke werden aufsummiert und von dem Wert des in der Mitte liegenden Rechtecks subtrahiert. Bei vier Rechtecken ergibt sich der Wert aus der Differenz der diagonal liegenden Flächen [1] [2].
In einem Bildfenster mit einer Auflösung von 24 x 24 ergeben sich über 18000 mögliche Merkmale, da nicht nur die Form sondern auch die Position und Größe

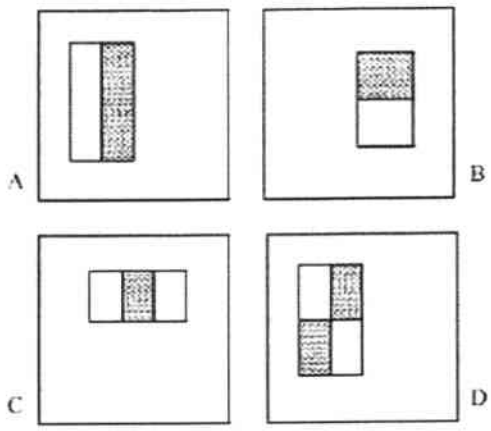

Abb. 2.1: 4 verschiedene Arten von Merkmalen. Die 2-Rechteck-Merkmale sind in (A) und (B) zu sehen. (C) zeigt das 3-Rechteck-Merkmal, und (D) das 4-Rechteck-Merkmal

der Rechtecke im Bildfenster variabel ist. Eine schnelle Berechnung dieser Merkmale ist erforderlich, um die Detektion in Echtzeit zu ermöglichen. Eine Losüng dafür ist die Verwendung eines Integralbilds (siehe Abbildung 2.2).
Das Intergralbild liefert an der Stelle x, y die Summe aller Pixelswerte links über und neben dem Punkt x,y:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{2.1}$$

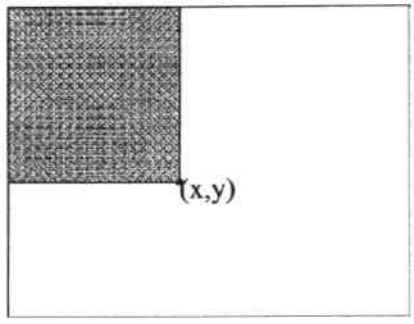

Abb. 2.2: Intergralbild: Der Wert des Punkts (x,y) ist die Summe aller Pixelwerte über und links von (x,y)

wobei ii(x,y) das Intergralbild ist und i(x,y) das Originalbild.
Durch die Benutzung folgender Funktionen:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{2.2}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{2.3}$$

kann das Intergralbild in einem Durchgang mit Hilfe des Originalbilds berechnet werden.[1]
Mit dem Integralbild kann wiederum eine beliebige Rechteck-Summe mit vier Array-Referenzen (siehe Abb. 2.3) berechnet werden. Die Differenz zwischen zwei Rechteck-Summen kann mit acht Referenzen eindeutig berechnet werden. In unserem Fall haben die beiden Recktecke eine gemeinsame Seite, d.h., das Merkmal läßt sich mit sechs Array-Referenzen bestimmen. Für das 3-Rechteck-Merkmal bzw. 4-Rechteck-Merkmal sind 8 bzw. 9 Array-Referenzen nötig.[1]

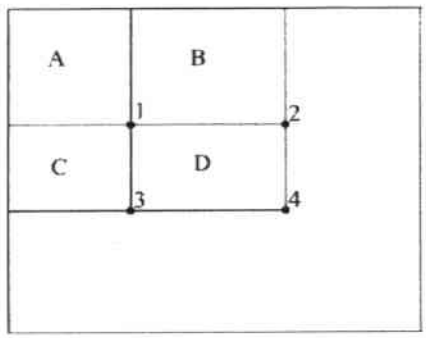

Abb. 2.3: Die Summe der Pixel in D kann mit vier Array-Referenzen berechnet werden. In dem Intergralbild ist der Wert von Punkt 1 die Summe aller Pixel in A. Der Wert von Punkt 2 ist A + B, der von Punkt 3 ist A + C und der von Punkt 4 ist A + B + C + D. Daraus folgt, die Summe aller Pixel in D ist 4 + 1 - (2 + 3)

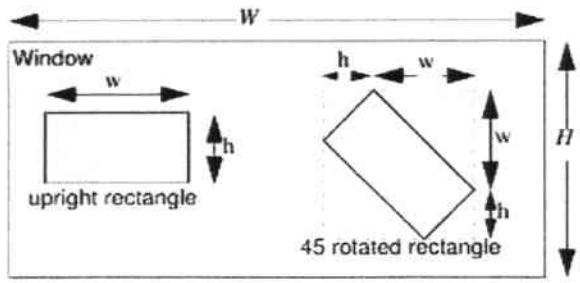

Abb. 2.4: Ein Beispiel für das Original-Merkmal und das um 45 Grad rotierte Merkmal.

Im Jahr 2002 haben R.Lienhart und sein Kollege die Merkmale des Original-Verfahrens erweitert, indem sie die vorherigen Merkmale um 45 Grad rotiert haben [3]. In der Abbildung 2.4 ist ein Beispiel zu sehen. Nach dieser Erweiterung ist die Menge der Merkmalen, die im ganzen System verwendet werden können, mehr als 4. Alle Merkmale sind in der Abbildung 2.5 zu sehen. Diese Erweiterung ermöglicht die bessere Detektion von rotierten Objekten und macht aus diesem

Grund das ganze Verfahren robuster.

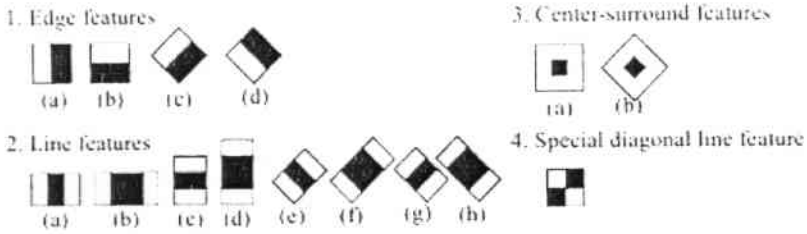

Abb. 2.5: Übersicht über mögliche Merkmale

2.1.3 Boosting Lernverfahren

Der Boosting Lernalgorithmus trainiert Klassifikatoren für 2-Klassen Probleme. Die Theorie des Boostings besagt, dass die Leistungen von mehreren schwachen Klassifikatoren kombiniert werden können, um eine endgültige mächtige Entscheidung zu treffen. Die schwachen Klassifikatoren sind sehr einfach aufgebaut und berücksichtigen meist nur ein einziges Merkmal der Objekte oder eine Kombination der Merkmalen als Entscheidungsbaum. Für sich genommen liefern sie deswegen einerseits schlechte Ergebnisse, können aber andererseits sehr schnell ausgewertet werden. Im Boosting Verfahren werden in der Trainingsphase die schwachen Klassifikatoren gewichtet. Die Gewichtung besagt, wie gut der Klassifikator im Vergleich zu allen anderen schwachen Klassifikatoren ist.
AdaBoost ist ein Boosting Lernalgorithmus, der im Jahr 1995 von Freund und Schapire veröffentlicht wurde. Der Pseudocode ist in der Tabelle 2.1 zu sehen.

Gegeben sind m gelabelte Trainingsdaten, wobei x_i ein Element der Menge X und y_i das entsprechende Label ist. Im Fall der 2-Klassen Probleme ist y_i entweder 1 oder -1. Zu trainieren sind T verschiedene schwache Klassifiktoren F : X $\rightarrow$ {-1;1}. Gesucht sind die T Gewichtungsfaktoren $\alpha_1, \alpha_2, ..., \alpha_T$ der entsprechenden schwachen Klassifikatoren.
Am Anfang des Trainings wird eine Initialverteilung: $D_1(i) = 1/m$ definiert. In der Trainingsphase werden T Iterationen durchgeführt. Bei jeder Iteration wird der Fehler ε_t des schwachen Klassifikators berechnet:

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i) \tag{2.4}$$

Danach wird $\alpha_t = \frac{1}{2} ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ definiert. Intuitiv gibt α_t an, wie gut der schwache Klassifikator ist. Je kleiner ε_t ist, desto größer ist α_t. Anhand von α_t und der

Gegeben: (x_1, y_1), .. , (x_m, y_m) mit $x_i \in$ X, $y_i \in Y = \{-1, +1\}$
Initialisere: D_1(i) = 1/m
Für t = 1,..,T:
Trainiere schwachen Klassifikator mit der Verteilung D_t
Nehme schwache Hypothese $h_t : X \rightarrow \{-1, +1\}$ mit den Fehler:
$\varepsilon_t = Pr_{i \sim D_i}[h_t(x_i) \neq y_i]$
Wähle $\alpha_t = \frac{1}{2} ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
Aktuallisiere:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t(x_i) = y_i \\ e^{\alpha_t} \text{ if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

,mit Z_t dem Normalisierungsfaktor (wird gewählt, so dass D_{t+1} Wahrscheinlichkeitsverteilung.).
Ausgabe der Endhypothese:
$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Tabelle 2.1: Der Pseudocode von AdaBoost - ein Boosting Algorithmuß [8]

alten Verteilung wird die neue Verteilung berechnet(siehe 2.1.3). Die gesamte Hypothese ist wie folgt definiert:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{2.5}$$

Die Gewichtungsfaktoren sollen so optimiert werden, dass der Fehler des gesamten Klassifikators so gering wie möglich ist.
Nach T schrittweisen Optimierungen wird die Gewichtung von allen T schwachen Klassifikatoren bestimmt.

2.1.4 Klassifikatorkaskade

Grundlegendes Konzept

Dieser Abschnitt beschreibt einen Algorithmus für den Bau einer Kaskade von Klassifikatoren. Damit wird eine erhöhte Leistung erreicht, während die Klassifikationszeit radikal verringert wird. In diesem Ansatz wird eine Menge N von verschiedenen Klassifikatoren, sogenannten Stages mit steigender Komplexität hintereinander geschaltet und zu einer Kette zusammengesetzt, wie in der Abblidung 2.6

zu sehen ist. Die einfachen Klassifikatoren werden am Anfang der Kette verwendet, um so viele falsche Objekte wie möglich heranszufiltern aber auch gleichzeitig fast alle richtigen Objekte zu erkennen. Danach kommen die komplexeren Klassifikatoren zum Einsatz, um die Falschalarmrate zu reduzieren.

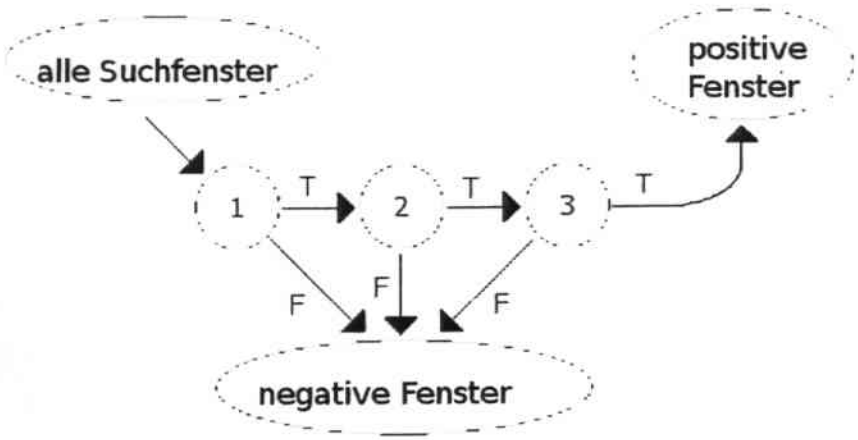

Abb. 2.6: Kaskadestruktur

Die Klassifikatoren in jedem *Stage* werden nach dem Prinzip des AdaBoost-Lernalgorithmus trainiert.

Training einer Klassifikatorkaskade

Das Ziel einer Kaskade ist nicht nur die erhöhte Erkennungsrate sondern auch die geringe Detektionszeit. Für die Gesichtserkennung-Aufgabe haben vorangegangene Systeme gute Erkennungsraten (zwischen 85 und 95 Prozent) und eine extrem niedrige Falschalarmrate (in der Größenordnung von 10^{-5} oder 10^{-6}) erreicht. Die Anzahl von Stages und die Leistung der einzelnen Stages muss deshalb so gewählt werden, dass die Erkennungsrate maximiert und die Detektionszeit minimiert wird.
Angenommen, der Klassikator der i-ten Stufe hat eine maximale Falschalarmrate f_i bzw. eine minimale Hitrate d_i und insgesamt werden K Stages hintereinander geschaltet, dann ergeben sich die gesamte Falschalarm- bzw. Hitrate wie folgt:

$$F = \prod_{i=1}^{K} f_i \tag{2.6}$$

$$D = \prod_{i=1}^{K} d_i \tag{2.7}$$

Ein beliebiges Objekt, das klassifiziert werden soll, wird durch verschiedene Stages der Kaskade überprüft, um die Entscheidung zu treffen, ob es positiv oder negativ

klassifiziert ist. Das Objekt, das in allen Stufen als positiv erkannt wird, ist positiv in der Gesamt-Entscheidung.
Die Anzahl der evaluierten Merkmale ist dann:

$$N = n_0 + \sum_{i=1}^{K}(n_i \prod_{j<i} p_j) \tag{2.8}$$

, wobei N die erwartete Anzahl von Merkmalen ist, K für die Anzahl der Klassifikatoren steht, p_j die Hitrate des j-ten Klassifikators ist, und n_i die Anzahl der Merkmale des i-ten Klassifikators ist.
Der allgemeine Trainingsprozess umfasst zwei Arten von Tradeoffs. In den meisten Fällen erreichen Klassifikatoren mit mehreren Merkmalen höhere Erkennungsraten und niedrigere Falschalarmraten. Allerdings benötigen solche Klassifikatoren auch mehr Zeit zur Berechnung. Im Prinzip könnte eine Optimierung im Rahmen von folgenden Parametern vorgenommen werden.

- die Anzahl von Stages,
- die Anzahl der Merkmale n_i jeder Stufe,
- der Schwellwert jeder Stufe

In der Praxis wird, um die Kaskade zu trainieren, die Anzahl von Stages, die minimale Hitrate und die maximale Falschalarmrate je Stage angepasst. Jede Stage der Kaskade wird durch AdaBoost trainiert, bis bei jedem Stage die vorher definierte Erkennungs- und Falschalarmrate erreicht wurden.([1])

2.1.5 Zusammenfassung

Der oben beschriebene Ansatz von Viola und Jones ist ein hervorragender Ansatz, der es ermöglicht, Gesichter bzw. Objekte in Bildern zu erkennen. Durch die Veröffentlichung des Ansatzes haben die beiden Autoren zu neuen Erkenntnissen für drei verschiedene Probleme in der Bildverarbeitung und bei der Objekterkennung besonders der Gesichtserkennung beigetragen. Die erste Verbesserung ist die neue Methode, mit Integralbildern Merkmale zu extrahieren. Der zweite Beitrag ist der einfache und schnelle auf AdaBoost basierte Klassifikationsalgorithmus. Der dritte Beitrag ist die Benutzung von Kaskaden, um die Leistung des Detektors zu verbessern und gleichzeitig eine Minimierung der Berechnungszeit zu erreichen.

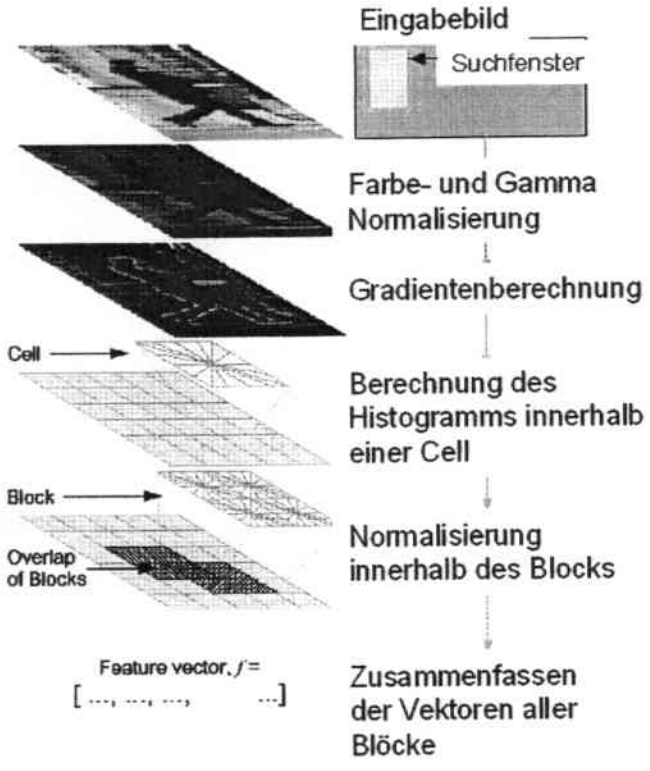

Abb. 2.7: Eine Übersicht über die Merkmalsextraktions- und Objekterfassungskette

2.2 Histogram of Oriented Gradients (*kurz* : HOG)

2.2.1 Einführung

Um Menschen zu detektieren haben Dalal und Triggs im Jahr 2005 einen Ansatz veröffentlicht, der auf Testdaten des MIT (Eine Bilder-Datenbank des Massachusetts Institute of Technology, für Details siehe [15]) fast perfekte Ergebnisse geliefert hat. Der grundlegende Gedanke bei HOG ist, dass das Aussehen und die Form der Objekte innerhalb eines Bildes, ohne genaues Wissen über die Positionen der Kanten oder Ecken, durch die Verteilung der lokalen Intensität oder der Anordnung der Kanten dargestellt werden kann.

Die Abbildung 2.7 gibt einen Überblick über das gesamte Verfahren. Für das Eingabebild wird zuerst eine Gamma-/Farb-Normalisierung und danach eine Gradientenberechnung durchgeführt. Das resultierende Bild wird in kleine sogenannte *Zellen* unterteilt. Innerhalb jeder Zelle wird für die Pixel das Histogramm der Kantenorientierungen berechnet. Zur Verbesserung der Leistung des Verfahrens wird das lokale Histogramm durch die Berechnung der Intensität über eine größe-

re Region des Bildes ermittelt, dem sogenannten *Block*. Mit Hilfe dieses Wertes werden alle Zelle in dem Block normalisiert. Diese Normalisierung führt zu einer besseren Robustheit gegenüber Veränderungen der Beleuchtung. Die Histogramme orientierter Gradienten aller Blöcke werden zu einem Merkmalvektor zusammengeführt.

2.2.2 Merkmale

Der HOG-Merkmalsvektor besteht aus den Histogrammen der normalisierten Zellen aus allen Blöcken. Üblicherweise überschneiden sich die Blöcke, das heißt, die einzelnen Zelle kommen nicht nur ein Mal im Merkmalsvektor vor. Im Allgemeinen unterscheidet man zwei verschiedene geometrische Arten von Blöcken: Rectangular HOG (R-HOG) und Circular HOG (C-HOG).

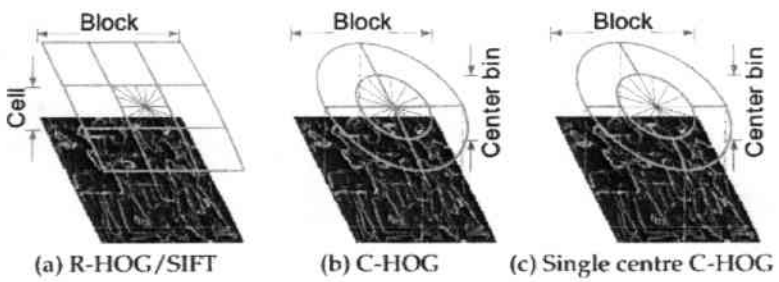

Abb. 2.8: Varianten von HOG Blöcken: a) RHOG mit 3x3 Zellen je Block b) C-HOG, bei dem die zentrische Zelle in verschiedene Sektoren unterteilt wird, c) Single-centre C-HOG mit einer einzigen zentrischen Zelle.

R-HOG

R-HOG Blöcke verwenden überlappende quadratische oder rechteckige Gitter von Zellen. Die Werte der Blöcke werden über dichte gleichmäßige Gitter ermittelt. Die Blöcke überlappen sich üblicherweise und jeder ist unäbhangig von den anderen normalisiert. RHOG berechnet ς x ς Netze (die Festlegung der Anzahl von Cells in jedem Block) von η x η Pixel-Cells mit β Ausrichtungen von bins, in denen ς, η, β Parameter sind. Die Abbildung 2.8 (a) zeigt ein 3x3 Zelle RHOG Block als Beispiel.

C-HOG

Im Vergleich dazu sind die Zellen bei C-HOG als Gitter von Kreisen definiert. Es gibt 2 Varianten von C-HOG: C-HOG, indem das zentrische Cells in verschiedene Sektoren unterteilt wird (siehe (b) in Abbildung 2.8) und Single-centre C-HOG, also C-HOG mit einem einzigen zentrischen Cell.
Der Anzahl der Cells und die Größe eines Cells in einem Block sind abhängig von 4 folgenden Parametern:

- Anzahl der Ausrichtungen
- Anzahl der Kreise
- Radius der zentrischen Zelle
- Skalierungsfaktor des nächsten Radius

Im Vergleich zu C-HOG hat das System mit R-HOG laut Ergebnissen von Dallal und Triggs mit gleicher Falschalarmrate eine bessere Erkennungsrate. Aus diesem Grund wird das System mit R-HOG in dieser Studienarbeit aufgeführt.

2.2.3 Implementierung

Im folgenden Abschnitt wird detailiert beschrieben, wie die Merkmale aus einem Bild bei der Implementierung von Dalal und Triggs extrahiert werden. Anbei sind einige Ergebnisse von Dalal&Triggs verwendet.

Gamma- / Farbnormalisierung

Gamma/Farbnormalisierung ist der 1. Schritt in der Objekterfassungskette. Zwei Varianten von Gamma-Normalisierung, Quadratwurzel-und Log-Komprimierung der einzelnen Farbkanäle, werden betrachtet.
Die Quadratwurzel-Gamma-Komprimierung ermöglicht eine bessere Leistung für Objektklassen wie Fahrräder, Motorräder, Autos, Busse und Menschen. Für Tiere, bei denen viele Variationen der Farbklasse vorkommen, wie zum Beispiel Katzen, Hunde oder Pferde, lieferte der Detektor mit unnormalisierten RGB eine bessere Leistung.(Siehe [9])

Gradientenberechnung

Der Gradient des Eingabebilds wird, mit optionaler Gaußscher Glättung mit der Standardabweichung σ, von mehreren diskreten Filtern, berechnet. Für die Farbbilder (RGB Raum) werden die getrennten Gradienten für jeden Farbkanal berechnet und danach mit der größten Norm als Pixel-Gradienten-Vektor gewählt.

Von Dalal&Triggs wurden u.a. folgende Filter implementiert und evaluiert.

- 1-D abgeleiteter Punkt:
  $([-1 \quad 1], [-1 \quad 0 \quad 1])$
- 3 x 3 Sobel Masken:
  $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
- 2 x 2 Diagonale:
  $(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix})$

Insgesamt ist die Leistung des Detektors unterschiedlich und abhänging von dem Filter und der Glättung. Das einfachste System, der 1-D Maske [-1, 0, 1] mit $\sigma = 0$, lieferte das beste Ergebnis(Für mehr Details siehe [9]).

Orientierte Zellen

Jedes Pixel trägt eine gewichtete Orientierung. Die Häufigkeit der Werte akkumuliert sich in den Orientierungsbehälter über lokale räumliche Regionen, Zelle. Die Orientierungsbehälter sind gleichmäßig von 0 bis 180 Grad (unsigned Gradient) oder von 0 bis 360 Grad (signed Gradient) verteilt. Zur Reduzierung des Aliasing wird trilineare Interpolation zwischen den benachbarten zentrischen Cells in Orientierung und Position durchgeführt. Trilineare Interpolation bedeutet lineare Interpolation in 3D Raum. (Für mehr Details siehe [9]).
Eine gute Auswahl der Orientierungsrepräsentation erweist sich als wesentlich für eine gute Leistung für alle Objekt-Klassen.

Blocks

Wie oben erwähnt wird in dieser Studienarbeit R-Quadrat HOG verwendet. Deshalb werden ς x ς Netze (die Festlegung der Anzahl der Zellen in jedem Block) von η x η Pixel-Cells mit β Bin-Ausrichtungen verwendet, in denen ς, η, β anzupassende Parameters sind.

Block-Normalisierung

Die Größe des Gradienten variiert über einen weiten Bereich aufgrund von örtlichen Schwankungen der Beleuchtung und des Vordergrund-Hintergrund-Kontrasts. Daher erweist sich eine effektive lokale Normalisierung als wesentlich für eine gute

Leistung. Vier verschiedene Normalisierungsmethoden wurden implementiert und evaluiert.(Für mehr Details siehe [9]).

- L2-norm, v $\leftarrow$ v / $\sqrt{||v||_2^2 + \epsilon^2}$
- L2-Hys, L2-Norm gefolgt von Clipping (Begrenzung der maximalen Werte von v auf 0,2) und Renormalisierung;
- L1-norm, v $\leftarrow$ v / $(||x||_1 + \epsilon)$
- L1-Sqrt, v $\leftarrow \sqrt{v/(||x||_1 + \epsilon)}$

Je nach Aufgabe kann man eine der oben genannten Normalisierungsmethoden nutzen, indem man die entsprechenden Parameter anpasst. (Für mehr Details siehe Anhang).
Nach der gesamten Berechnungen werden die Histogrammen orientierter Gradient aller Blöcke zu einem Merkmalsvektor zusammengeführt. Dann werden dieser Vektor mit Hilfe des auf einem Support-Vektor-Maschine basierten Klassifiktors untersucht, ob das Eingabebild zu erkennende Objekt ist oder nicht.

2.2.4 Support-Vektor-Maschine (*kurz*: SVM)

Grundlagen

In der Sprache des maschinellen Lernens schließen sich die SVMs an das Feld des Supervised Learning (überwachtes Lernen) an. Gemeint ist damit ein Lernen durch Beispiele. Es gibt von vorneherein einen Trainingsdatensatz mit Beispielen für die zu unterscheidenden Klassen. In erster Linie ist es nicht das Ziel die Trainingsdaten sondern die Testdaten zu trennen. Deshalb ist es nötig eine gute Generalisierbarkeit zukünftiger Daten zu erreichen. Es sollen Klassenunterschiede mathematisch kompakt ausgedrückt werden. Jedes Element einer Klasse wird durch einen Merkmalsvektor dargestellt, der die Eigenschaften der Klasse gut repräsentiert. In unserem Fall sind die Merkmalsvertoren die Histogramme von orientierten Gradienten des zu klassifizierenden Fensters. Gelernt werden soll der Normalvektor w der optimalen Hyperebene von allen möglichen Ebenen, die die beiden Klassen voneinander trennen(siehe Abbildung 2.9).
In der Abbildung 2.9 stellen die Punkte zwei verschiedene Klassen dar. Die Hyperebene soll es ermöglichen, später hinzukommende Elemente einer der beiden Klassen zuordnen zu können. Bei hochdimensionalen Räumen besteht das Problem darin, dass die Klassen auf verschiedene Arten getrennt werden können (siehe Abbildung 2.9). Das Ziel von SVM ist, die beste Hyperebene mit entsprechendem w herauszufinden, um die beiden Klassen klar voneinander zu trennen.

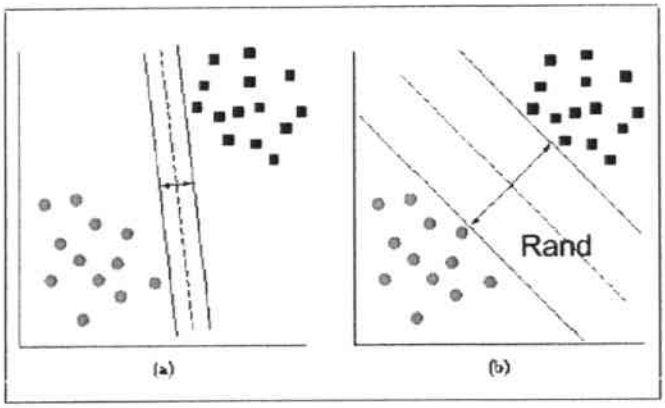

Abb. 2.9: Trennung der Klassen mittels Hyperebene

Lineare Support-Vektor-Maschine

Im Falle der linearen Klassifikation mit der SVM liegen die Daten durch ihre Darstellung als Vektor so im Raum, dass sie voneinander linear separierbar sind. Es wird dabei nach einer Entscheidungsfunktion f : $R^n \rightarrow R$ gesucht, die für die Trainingsdaten x=$(x_1, .., x_n)$ das Label 1, spricht ein positives Label, falls f(x) > 0 und anderseits ein negatives Lablel, falls f(x) < 0 vergibt und dadurch die Zugehörigkeit zu einer Klasse bestimmt.
Konkret werden gelabelte Trainingsdaten in der folgende Form dargestellt:

$$\{(\overrightarrow{x_m}, y_m), ..(\overrightarrow{x_N}, y_N)\}\, x \in X, y_i \in \{1. - 1\} \tag{2.9}$$

wobei $\overrightarrow{x_i}$ die jeweiligen Merkmalsvektoren und y_i die zugehörigen Labels sind. Die lineare Entscheidung bzw. Entscheidungsfunktion ist:

$$f(x) = \langle \overrightarrow{w}, \overrightarrow{x} \rangle + b \tag{2.10}$$

w beschreibt dabei den Normalvektor und b den Abstand vom Ursprung (Bias). Durch Verwendung der Signumfunktion sgn erfolgt nun die binäre Klassifikation. Der Abstand eines Merkmalsvertors x zur Hyperebene H lässt sich wie folgt berechnen:

$$dist(x, H) = |\frac{1}{||\overrightarrow{w}||}((\overrightarrow{w}, \overrightarrow{x}) + b)| \tag{2.11}$$

Wie in Abblidung 2.10 zu sehen ist, ist es möglich, unendlich viele Hyperebenen (Trennlinien) zwischen zwei Klassen anhand der Trainingsdaten zu erstellen. Die durch die SVM berechneter Hyperbene weist sich jedoch durch die Eigenschaft aus, so weit wie möglich von allen Objekten entfernt zu liegen. Das heißt, der geringste Abstand zu den Objekten, sowohl positiv als auch negativ gelabelter, soll maximiert werden.

Die am nächsten zur Hyperebene gelegenen Vektoren werden Support-Vektoren

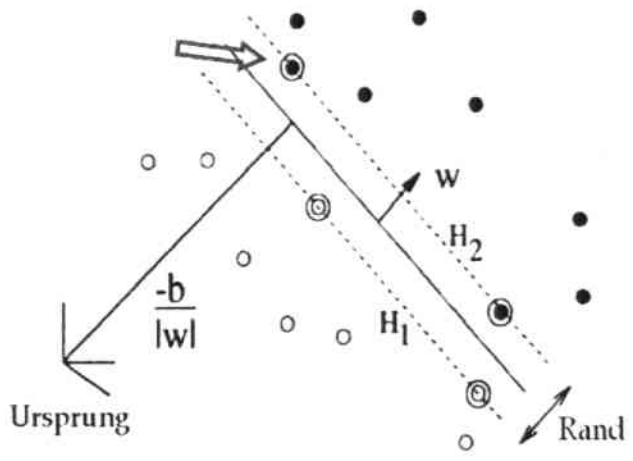

Abb. 2.10: Durch die Support-Vektoren wird die optimale Hyperebene festgelegt.

genannt, und nur solche Vektoren spielen für die Bestimmung der Hyperebene eine Rolle. Alle anderen Daten sind für die Berechnung nicht relevant und werden deshalb nicht weiter betrachtet. Der Abstand der Support-Vektoren zur Hyperebene ist $\frac{+1}{||\vec{w}||}$ für einen Vektor mit positivem Label bzw. $\frac{-1}{||\vec{w}||}$ für den anderen Fall. Die gesamte Breite ist $\frac{2}{||\vec{w}||}$ und wird Rand genannt.
Das Problem der Berechnung der optimalen Hyperbene ist jetzt äquivalent zu dem der Maximierung des Randes. Das heißt, dass $\frac{2}{||\vec{w}||}$ maximiert werden muss, d.h. $||\vec{w}||$ muss minimiert werden.
Das neue Problem ist jetzt:

$$min\ ||\vec{w}||^2 \tag{2.12}$$

unter den Bedingungen:

$$y_i(\langle \vec{w} \cdot \vec{x_i} \rangle + b) >= 1 \tag{2.13}$$

Dieses Optimierungsproblem kann durch die Lagrange Methode gelöst werden, indem der Sattelpunkt der folgenden Funktion gefunden wird:

$$L_p = L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{N} \alpha_i(y_i(\vec{w}\,\vec{x_i} + b) - 1) \tag{2.14}$$

mit $\alpha_1, \alpha_2, .., \alpha_N \geq 0$. Die Sattelpunkt-Bedingungen sind durch folgende Gleichungen charakterisiert:

$$\frac{\delta L}{\delta b} = \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{2.15}$$

$$\frac{\delta L}{\vec{w}} = \vec{w} - \sum_{i=1}^{N} \alpha_i y_i \vec{x_i} = 0 \tag{2.16}$$

Mit Hilfe von diesen Gleichungen können $\vec{w}$ und b durch α_i, x_i, y_i dargestellt werden. Dadurch ensteht die duale Lagrange-Gleichung:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j} \tag{2.17}$$

$W(\alpha)$ soll unter den Bedingungen:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \text{ und } \alpha_i \geq 0 \; i = 1, .., m \tag{2.18}$$

maximiert werden. Unter den Sattel-Bedingungen sind die meisten $\alpha_i = 0$. Die Support-Vektoren sind die Vektoren $\vec{x_i}$ mit $\alpha_i > 0$ und der gesuchte Vektor $\vec{w}$ ist eine lineare Kombination dieser Support-Vektoren.

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x_i} \tag{2.19}$$

Nichtlineare Klassifikation oder Klassifikation im höherdimensionalen Raum

Wenn die zu klassifizierenden Daten nicht linear serparierbar sind, wie in Abbildung 2.11, ermöglicht die SVM entweder mit Hilfe von zusätzlichen Parametern oder mit Hilfe von Kernelfunktionen die Problemlösung anzugehen.
Falls das Problem nicht deutlich serparierbar, aber doch noch separierbar ist, wird ein c-Wert hinzugefügt, um die Fehlklassifikationen der einen Klassen schwer zu bestrafen. Der Abstand der Fehlklassifikation von der Hyperebene sei im Folgenden (ζ).
Die Summe von allen Fehlklassikation kann wie folgt berechnet werden:

$$c * \sum \zeta^i \tag{2.20}$$

Diese Summe soll minimiert werden.
Es kommt oft vor, dass die zu lernenden Klassen unterschiedlich häufig repräsentiert vorliegen. In diesem Fall wird die Summe in den gewichteten Abstand der Fehlklassifikation auf die verschiedenen Klassen gespaltet. Es gibt nun zwei Summen:

$$c_+ * \sum \zeta_{i+} + c_- * \sum \zeta_{i-} \tag{2.21}$$

Durch den j-Wert werden Fehler auf den beiden Klassen unterschiedlich schwer gewichtet. j stellt das relative Verhältnis zwischen c_+ und c_- dar: j=c_+/c_-. Nun kann anstatt $||w||^2$ zu minimieren $||w||^2 + C(\sum_i \zeta_i)$ minimiert werden.
Falls die Daten deutlich nicht linear serparierbar sind (siehe Abbildung 2.12), soll das Problem mit Hilfe von Kernelfunktionen gelöst werden.

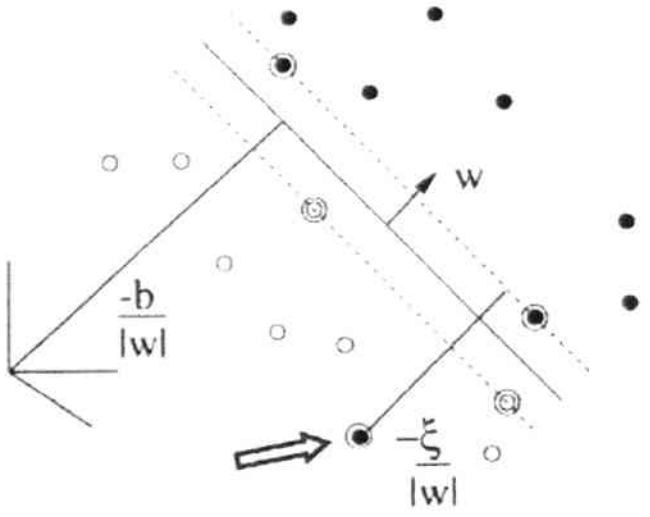

Abb. 2.11: Abstand der Missklassifikation

Die Kernelfunktion Φ beschreibt also eine Abbildung $\Phi : R^n \rightarrow R^N$ wobei n < N und R^n den Eingaberaum darstellt.
Eine solche Kernelfunktion existiert,wenn es zu einem Skalarprodukt <u,v> im Ursprungsraum eine Abbildung gibt, so dass:

$$\forall u, v \in R^n : K(u, v) = \langle \Phi(u) \cdot \Phi(v) \rangle \tag{2.22}$$

Es gibt vier verschiedene grundlegende Kernelfunktionen:
Linear:

$$K(x, y) := \langle x, y \rangle = x_T \cdot y \tag{2.23}$$

Polynomiell:

$$K(x, y) = (\gamma x^T y + r)^d, \gamma > 0 \tag{2.24}$$

Sigmoid:

$$K(x, y) = tanh(\gamma x^T y + r) \tag{2.25}$$

Gauss/radiale Basisfunktion(RBF):

$$K(x, y) = exp(-\gamma ||x - y||^2), \gamma > 0 \tag{2.26}$$

Hierbei sind die Variablen γ, r und d Parameter der verschiedenen Kernels. Für verschiedene Aufgaben sollen die Parameter angepasst werden, um die erwartete Leistung zu erreichen.
Für weitere Details siehe [11].

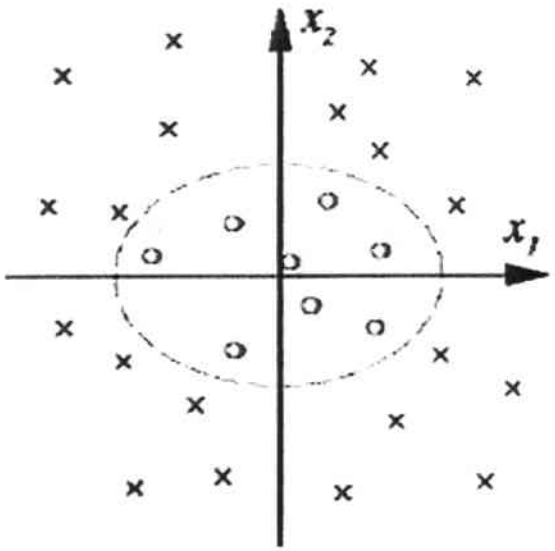

Abb. 2.12: Nicht linear seperierbare Daten im 2-dimensionale Raum.

2.2.5 Zusammenfassung

Im Vergleich zu dem Viola&Jones Ansatz basiert der Ansatz von Dalal&Triggs auf Histogrammen von orientierten Gradienten. Die Position der Gradienten wurde nicht betrachtet, sondern nur die Anzahl und die Richtung von Gradienten. Trotz des Informationsverlusts erreicht das System von Dalal und Triggs eine sehr gute Erkennungsrate. Zur Klassifikation wird eine lineare SVM verwendet. Mit anderen Kernels wird die Hitrate verbessert aber dafür steigt die Detektionszeit. Der Vorteil des Verfahrens ist die Stabilität. Solange die Form des Objekts vorhanden ist, kann das Objekt mit genügend Trainingsdaten eingelernt und erkannt werden. Allerdings ist die lange Detektionzeit ein Nachteil, da für jedes Fenster das Gradientenhistogramm berechnet und danach durch den SVM-Klassifikator untersucht werden muss.

2.3 Anwendung der Klassifikation

Um Oberköper in einem Bild zu detektieren müssen verschiedene Suchfenster mit verschiedenen Größen und an verschiedenen Orten des Bildes anhand des Klassifikators überprüft werden. Das bedeutet, dass Merkmale mit den vorher definierten Verfahren aus dem Fenster berechnet und extrahiert werden. Danach ist es anhand der Merkmale zu unterscheiden, ob es sich bei dem gerade geprüften Fenster um einen Oberkörper handelt.

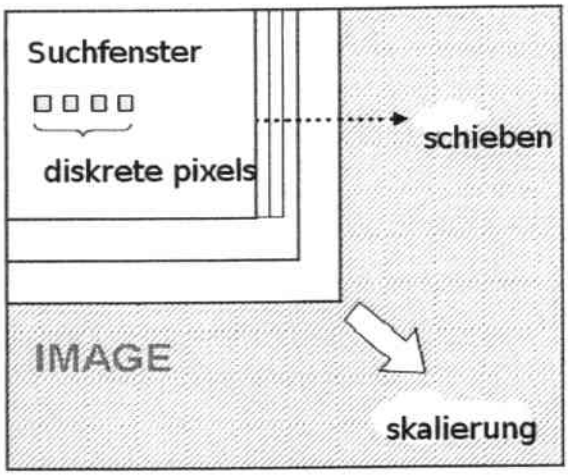

Abb. 2.13: Anwendung der Klassifikation

Jedes Fenster hat die Koordinaten (x,y), die Breite w und die Höhe h. Die Initialisierung ist der Startwert für (x,y) und die Mindestgröße des Fensters. Die 1.Iteration wird durchgeführt, in dem alle Fenster mit der Mindestgröße von der Anfangspostion bis zum Ende des Bildes überprüft werden (siehe Abbildung 2.13). Es kann von unten nach oben oder umgekehrt und von links nach rechts oder umgekert durchgeführt werden. Die Größe der Schritte entscheidet über die Genauigkeit des Detektors und deshalb auch über die Hitrate und die Falschalarmrate. Je größer der Schritt, desto grober ist die Detektion. Die nächsten Iterationen funktionieren genau wie die 1.Iteration nur mit anderen Fenstergrößen. Diese Größe ist von einem vorher definierten Skalierungsfaktor abhängig. Das Scanen wird so lange durchgeführt, bis Abbruchkriterien erfüllt sind. Das bedeutet, dass ein Objekt im Bild mehrere Male gescannt wird und deshalb auch mehrere Male erkannt werden kann. Zur Vermeidung dieses Effekts wird ein sogenannter Parameter *min_neighbor* festgelegt, der angibt, wie viele Male ein Objekt in einer lokalen Nachbarschaftsregion mindestens detektiert werden muss, damit es als Objekt in der endgültigen Entscheidung erkannt wird.
Die Parametisierung des Detektors beim Scannen des Bildes ist sehr wichtig. Sie entscheidet über die Leistung und die Geschwindigkeit des Detektors. Je genauer der Detektor funktioniert, desto länger ist die Laufzeit. Auch die Hitrate und die Falschalarmrate erhöhen sich. Deshalb ist es nötigt, ein Trade-Off zwischen Hitrate, Falschalarmrate und Geschwindigkeit zu finden.

3 Daten und Annotation

Bei den Detektionsaufgaben spielen die Evalutionsdaten eine große Rollen. Sie entscheiden sowohl über die Leistung als auch das Potential des neuen Systems. In dieser Studienarbeit werden zwei verschiedene Datensätze verwendet, um die Systeme zu trainieren und zu evaluieren. Beide bestehen aus Bildern von Videosequenzen. Im folgenden Kapitel wird beschrieben, wie die Daten für unsere Bewertung verwendet und wie die Annotationen durchgeführt werden.

3.1 Daten

Für die Bewertung wurden zwei verschiedene Datensätze verwendet. Während der 1.Datensatz aus Videosequenzen von einzelnen Mitarbeitern entnommen wurde, besteht der 2.Datensatz aus Bildern der Videosquenzen von verschiedenen Meetings im intelligenten Raum des Instituts.

3.1.1 1.Datensatz

Dieser Datensatz besteht aus 7 verschiedenen Videosquenzen von 7 verschiedenen Mitarbeitern, die auf dem Flur des Instituts aufgenommen wurden. Die Kamera steht in Kopfhöhe (ungefähr 1m80). Die Videosquenzen dauern durchschnittlich jeweils 1 Minute und bestehen durchschnittlich aus 900 Frames. Von jeder Sequenz wird jedes 5.Frame als Testbild bzw. Trainingsbild verwendet. Die Bilder haben eine Auflösung von 640x480 Pixel und das Format RGB. Ein Beispielsbild des 1.Datensatzes ist in der Abbildung 3.1 zu sehen.

Die Videosequenzen werden von 1 bis 7 nummeriert. Die Tabelle 3.1 stellt die Anzahl Bilder und Oberkörper in Frontalaufnahme für jede Videosequenz dar.
Alle Frontaloberköper dieser Bilder werden annotiert, ausgeschnitten und als Trainingsdaten verwendet.

Abb. 3.1: Ein Bild aus dem 1.Datensatz

Videosequenz	1	2	3	4	5	6	7
Anzahl Bilder	173	94	211	136	167	218	186
Anzahl Frontaloberkörper	46	29	61	34	49	100	50

Tabelle 3.1: Die Anzahl aller Bilder und dazugehöriger Frontaloberkörper im 1. Datensatz

3.1.2 2.Datensatz

Im Vergleich zum 1. Datensatz ist der 2. Datensatz deutlich komplexer. Er besteht aus 40 verschiedenen Videosequenzen von 10 verschiedenen Meetings im intelligenten Raum des Instituts. Jedes Meeting wurde von 7 verschiedenen Kameras aufgenommen. Für diese Bewertung wurden 4 Kameras, an den oberen Ecken des Raums, in einer Höhe von 2,70 m, verwendet. 4 mögliche Bilder von den Kameras in einem Zeitpunkt ist in der Abbildung 3.2 zu sehen. Wie die Bilder im 1. Datensatz haben diese Bilder auch die Auflösung von 640x480 Pixel und das Format RGB. Die Videosequenzen dauern durchschnittlich jeweils 10 Minuten bzw. bestehen aus 9000 Frames. In dieser Arbeit wurden nur die ersten 2000 Frames als Test- bzw, Trainingsbilder verwendet, da die Annotation der Bilder sehr zeitaufwändig ist.

Jedes 15te Frame (1 s) in den Videosequenzen der sieben ersten Meetings wird als Trainingsbild verwendet. Hier werden alle Oberkörper in verschiedenen Orientierungen annotiert, ausgeschnitten. In dieser Studienarbeit wird die Drehung des Oberkörpers in 8 Klassen aufgeteilt. Falls der Oberkörper zur Kamera eine Drehung von -22,5 Grad bis 22.5 Grad bzw. -162,5 bis 162,5 Grad hat, wird er zur Klasse 0 Grad bzw. 180 Grad eingeordnet. Falls der Oberkörper im Vergleich zur

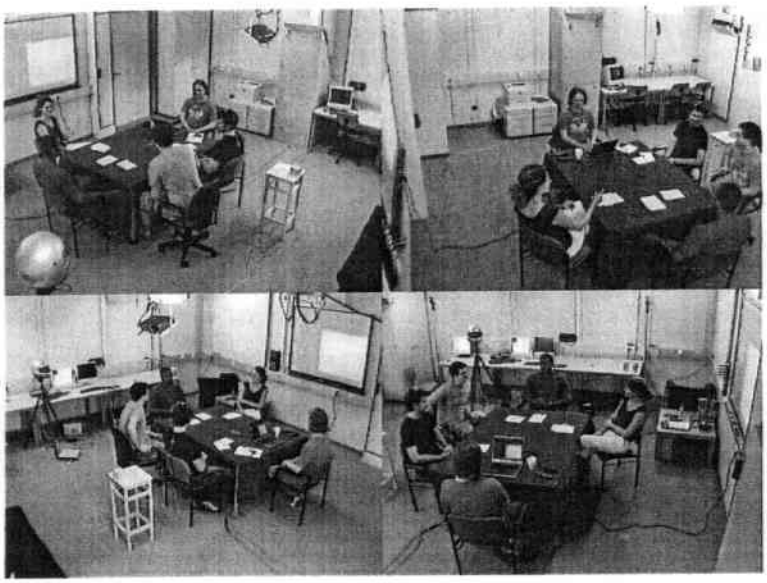

Abb. 3.2: 4 mögliche Bilder von den Kameras in einem Zeitpunkt

Kamera einen Winkel von 45 (± 22.5) Grad (oder 90 (± 22.5), 135(± 22.5) Grad) hat und nach links gedreht ist, dann wird der Oberkörper zur Klasse +45 Grad (oder +90, +135 Grad) eingeordnet. Umgekehrt, falls der Oberkörper rechts gedreht ist, wird er zur Klasse -45 Grad (oder -90, -135 Grad) eingeordnet.
In diesem Datensatz ist die Anzahl von Oberköpern mit entsprechenden Klassen zum Trainieren in der Tabelle 3.2 zu sehen.

Klasse(Grad)	0	45	90	135	180	-45	-90	-135
Anzahl Oberkörper	569	394	270	244	455	331	83	330

Tabelle 3.2: Die Anzahl der als Trainingsdaten verwendeten Oberkörper im 2. Datensatz

Ausserdem werden 1946 Bilder annotiert, ausgeschnitten und als negative Trainingsbilder verwendet. Einige Beispiele für positiven Trainingsbilder sind in der Abbildung 3.3 zu sehen.

Abb. 3.3: Einige positive Trainingsbilder aus dem 2.Datensatz

Jedes 75te Frame (5 s) der Videosequenzen der letzten 3 Meetings wird als Testbild verwendet. Insgesamt gibt es 324 Testbilder mit 710 Oberkörper mit verschiedenen

Orientierungen. Die Tabelle 3.3 stellt die Anzahl der Oberkörper in dazugehörigen Klassen dar.

Klasse(Grad)	0	45	90	135	180	-45	-90	-135
Anzahl Oberkörper	138	101	51	103	82	97	35	113

Tabelle 3.3: Die Anzahl der als Testdaten verwendeten Oberkörper im 2. Datensatz

3.2 Annotation

Allgemein wurden alle Annotationen auf den Originalbildern durchgeführt. Jede Annotation besteht aus einem Fenster, das den Kopf und den Oberkörper der Person umfasst. Die 4 Kanten des Fensters werden wie folgt bestimmt:

- die obere Kante ist die horizontale Tangente zu dem Kopf
- die untere Kante ist parallel zu der oberen Kante und schneidet auf Höhe der unteren Achsel
- die linke bzw. rechte seitliche Kante schneiden die äußersten Punkte der Schultern bei den Klassen von 0, 45,90, 135 und 180 Grad bzw. von -45, -90, -135 Grad

Wichtig ist zur Vermeidung von Verzerrungen, dass das Fenster so angepasst wird, dass es ein Quadrat bildet. Die Fenster werden je nach Drehung des Oberkörpers in ensprechende Klassen aufgeteilt.
Die Größe der Fenster ist verschieden und abhängig von der Größe des Oberkörpers. In dieser Studienarbeit wurden nach der Annotation den Fenstern zusätzlich noch ein Rand auf jeder Seite hinzugefügt, dann wurden sie aus dem Bild ausgeschnitten und auf 80x80 bzw.20x20 skaliert. Bei dem 1.Datensatz beträgt der Rand 10 % der Größe des Oberkörpers, während bei dem 2.Datensatz mit 2 verschiedenen Größen von Rand experimentiert wurde. Beim 1.Versuch beträgt der Rand 10 % der Größe des Oberkörpers und beim 2.Versuch 20 %.

4 Experimente

4.1 Vorbereitung

Um einen relativen Vergleich zwischen den oben genannten Ansätzen durchzuführen werden zwei verschiedene Systeme traniert und evaluiert. Das 1. System basiert auf dem Ansatz von Viola & Jones, während das 2. System auf dem HOG Ansatz basiert.

4.1.1 Klassifikatorkaskade (Viola&Jones)

Für dieses System wird die Implementierung von OpenCV ([12]) verwendet. Das System besteht aus 3 Hauptteilen.
Im 1. Teil werden die Trainingsdaten vorbereitet. Hierbei ist die Größe des Oberköpers festgelegt. Diese Festlegung ist nicht nur für die gesamte Trainingsphase sehr wichtig sondern auch für die Testphase. Dabei gilt, je größer die Trainingsbilder sind, desto länger ist die Trainingszeit. Aus diesem Grund werden in unserem Fall alle Experimente mit Oberkörperbildern durchgeführt, die auf die Größe 20x20 skaliert wurden.

Der 2.Teil ist die Trainingsphase. Die wichtigen zu beoachtenden Parameter sind:

- Anzahl von Splits (nsplits): Dieser Parameter gibt an, welche Arten von schwachen Klassikatoren in jedem Stage verwendet werden. Im Fall nsplits $= 1$, verwendet jeder schwache Klassifikator nur ein einziges Merkmal. Falls nsplits > 1, wird eine binäre Baumstruktur als schwacher Klassifikator verwendet, wobei bei jedem Knoten ein Merkmal und dazugehöriger Schwellwert angegeben werden.
- Anzahl von Stages (nstages): Dieser Parameter gibt die Anzahl von Stages in der Kaskadenstruktur an.
- Im OpenCV ist zu unterscheiden, welche Merkmale für die Klassifikatoren verwendet werden. Zur Wahl stehen hier Original-Merkmale von Viola&Jones und die erweiterten Merkmale von R. Lienhart
- Breite und Höhe von Trainingsbildern

- Minimale Hitrate pro Stage
- Maximal erlaubte Falschalarmrate pro Stage
- In OpenCV werden 3 verschiedene Versionen des Boosting-Algorithmus implementiert. Deshalb ist es auch wichtig zu unterscheiden, welche Version im gesamten Experiment verwendet wird.

(Für weitere Parameter siehe Anhang A)

Der 3. Teil ist die Testphase. Entscheidende Faktoren dieses Teils sind die Parameter für die Skalierung und die Mindestgröße des Suchfensters. Mit gut angepassten Parametern wird die Hitrate erhöht und die Falschalarmrate erniedrigt. Außerdem spielt der Parameter *min_neighbors* eine große Rolle (siehe Kapitel 2.3).

4.1.2 HOG

Die Implementierung von "INRIA Object Detection and Localization Toolkit" von Dalal ([14]) wird für das 2.System verwendet. Dabei ist die frei verfügbare SVM-Bibliothek SVMLight [13] für das Lernen und die Klassifikation zuständig. Wie das 1.System enthält dieses System auch 3 Haupteile.

Der 1.Teil bereitet die Trainingsdaten für den SVM Klassifikator vor. Die Merkmale werden extrahiert, berechnet und in das richtige Format (für mehr Details siehe [13]) gebracht. Viele wichtige Parameter liegen in diesem Teil, da hier entschieden wird, welche Größe die Trainingsbilder, die *Cells* und die Blöcke haben und wie groß die Überlappung ist. Die wichtigen zu beoachtenden Parameters sind:

- Auflösung einer Zelle
- Anzahl von Zellen in einem Block
- Anzahl von Orientierungsbehältern im Histogramm
- Größe der Blockverschiebung
- Glättung bei der Gradientenberechnung
- Normalisierungsmethode
- Methode zur Gradientenberechnung

Der 2.Teil ist für die Lernphase zuständig und aus diesem Grund kommt das SVMLight-Tool zum Einsatz. In diesem Fall wird ein linearer Klassifikator trainiert und deshalb ist, wie im Kapitel 2.2.4 erwähnt, das Verhältnis zwischen c_+ und c_- ein wichtiger Parameter.

Der 3.Teil ist die Testphase. Die folgenden Parameter sind wichtig und entscheiden über die Hitrate und die Falschalarmrate.

- Schwellwert des Klassifikators
- Mindestgröße des Suchfensters
- Skalierungsfaktor des Suchfensters
- Verschiebung des Suchfensters

Anhand der Tranings- und Testdaten wurde die Auflösung der Oberkörper für das Training auf 80x80 angesetzt.

4.2 Evalutionsmethode

4.2.1 Bewertung

Hit/Falschalarm-Berechnung

Um zu entscheiden, ob das detektierte Fenster ein Treffer (Hit) oder ein falscher Alarm ist, soll die gemeinsame Fläche zwischen den detektierten und referenzierten Fenstern berechnet werden. Angenommen, das detektierte Fenster bzw. das referenzierte Fenster hat die Fläche A bzw. B. Die gemeinsame Fläche soll G sein. Der Score S ist wie folgt definiert:

$$S = \frac{1}{2} \cdot (\frac{G}{A} + \frac{G}{B}) \tag{4.1}$$

Falls $S \geq 0.7$ wird entschieden, dass es ein Hit ist, falls $S < 0.7$ dann wird es als falscher Alarm abgewiesen. In dieser Studienarbeit wird die Falschalarmrate als die Anzahl der falsch detektierten Oberkörper pro Bild berechnet. Die Hitrate ist der Prozent der detektierten Oberkörper von gesamten Oberkoper im Testsätzen.

Konfusionsmatrix

Da es für Oberkörper kein festes Muster der Erscheinung gibt, werden die Oberkörper im 2.Datensatz in verschiedene Klassen je nach ihrer Orientierung eingeteilt. Für jede Klasse wird ein Klassifkator trainiert. Die Bewertung von jedem Klassifikator wird so erstellt, dass die Falschalarmrate in verschiedenen Klassen aufgeteilt wird. Beispielsweise wird ein 0 Grad Oberkörper Klassifikator evaluiert. Die Hitrate stellt die Anzahl von detektierten Oberkörpern mit 0 Grad Orientierung dar. Die Falschalarmrate ist die Summe der Anzahl A1 der detektierten Oberkörper der anderen Klassen und der Anzahl A2 der detektierten Objekte, die keinen Oberkörper darstellen. Von A1 wird festgehalten, welcher Klasse die annotierten Oberkörper zugeordnet wurden. Mit anderen Klassifikatoren wird genauso vorgegangen und dadurch entsteht eine Konfusionmatrix, die in der Tabelle 4.1

exemplarisch dargestellt ist.
In der 1.Zeile stehen alle möglichen Klassen der Orientierung, in der sich die Referenzoberkörper befinden können. '-' bedeutet, dass das evaluierte Fenster keinen Oberkörper darstellt. Die 1.Spalte ist die Liste aller Klassifikatoren, die trainiert wurden. Beispielweise kann die 4.Zeile wie folgt interpretiert werden:
Der 135 Grad Oberkörperklassifikator wird im gesamten Testset evaluiert. 25 % der gesamten Oberkörper in Orientierung von 0 Grad, 29 % von 45 Grad, 47 % von 90 Grad, 37 % von 135 Grad, 68 % von 180 Grad, 14 % von -135 Grad, 33 % von -90 Grad, 45 % von -45 und durchschnittlich 1.1 falsche Alarme pro Bild wurden detektiert.

Klasse	0	45	90	135	180	-135	-90	-45	-
0	x	x	x	x	x	x	x	x	x
45	x	x	x	x	x	x	x	x	x
90	x	x	x	x	x	x	x	x	x
135	0.25	0.29	0.47	0.37	0.68	0.14	0.33	0.45	1.1
180	x	x	x	x	x	x	x	x	x
-135	x	x	x	x	x	x	x	x	x
-90	x	x	x	x	x	x	x	x	x
-45	x	x	x	x	x	x	x	x	x

Tabelle 4.1: Eine mögliche Konfusionmatrix

Mit Hilfe der Konfusionsmatrix wird herausgefunden, wie viele Oberkörper tatsächlich von allen Klassikatoren detektiert wurden und wie hoch die Falschalarmrate ist. Für die Berechnung der klassenunabhängigen Hit- und Falschalarmraten werden die Hitrate und die Falschalarmrate aller Klassifikatoren werden so zusammenaddiert, dass jedes Fenster des Testbildes höchsten einmal detektiert wird. D.h., falls ein Fenster vom O Grad Klassifikator und vom 45 Grad Klassifikator detektiert wird, wird es nur einmal im Gesamtergebnis aufgenommen.

4.3 Ergebnisse

4.3.1 1. Datensatz

Um die beiden Systeme zu trainieren und zu evaluieren wird bei dem 1.Datensatz das Verfahren Round-Robin durchgeführt. Da der Datensatz aus 7 verschiedenen Videosequenzen mit 7 verschiedenen Personen besteht, werden dementsprechend 7 Iterationen durchgeführt. Bei jeder Iteration werden die Oberkörper von 6 Videosequenzen als Trainingsdaten und die Bilder der letzten Videosequenz als

Testsbilder verwendet. Das endgültige Ergebnis ist das durchschnittliche Ergebnis von allen 7 Iterationen. Bei diesem Experiment wurden nur frontale Oberkörper zum Trainieren verwendet.
Die Experimente wurden teilweise auf 5 Quadcore-Computern durchgeführt, wobei jeder Core die Taktfrequenz von 2.4 GHz und 4 MB Cache hat.

Klassifikatorkaskaden

Die Parameter des 1.Systems werden angepasst, so dass das System folgende Eigenschaften besitzt:

- Alle Oberkörper werden auf die Größe 20x20 skaliert.
- Bei jedem Stage verwendet jeder schwache Klassikator ein einziges Merkmal.
- Der gesamte Klassifikator wird aus 17 Stage aufgebaut.
- minimal zu erreichende Hitrate pro Stage ist 0.995
- maximal erlaubte Falschalarmrate pro Stage ist 0.5
- Gentle Adaboost wird verwendet
- Erweiterte Merkmale werden verwendet.

Die Trainingsphase wurde mit 7 verschiedenen Klassifikatoren paralell auf 5 Quadcore-Computer durchgeführt. Um einen Klassifikator fertig zu trainieren, wurden durchschnittlich 15 Minuten benötigt. Das folgende Ergebnis (siehe Tabelle 4.2) wurde erhalten.

Iteration	1	2	3	4	5	6	7	Gesamtergebnis
Hitrate	0.3	0.89	0.75	0.85	0.86	0.51	0.88	0.72
Falschalarmrate	0.26	0.14	0.21	0.44	0.2	0.28	0.44	0.28

Tabelle 4.2: Ergebnis des 1.Systems mit dem 1.Datensatz

Das 1. System erreicht eine maximale Hitrate von 89 %, während die minimale erreichte Falschalarmsrate 0.14 pro Bild ist. In der Abbildung 4.1 ist ein Beispiel mit detektiertem Oberkörper zu sehen. Bei der 1. und 5. Person kann das System nur eine Hitrate von 30 % bzw. 51 % erreichen. Der Grund liegt darin, dass die Anzahl der Trainingsbilder in diesem Experiment relativ gering ist. Außerdem ist ein Problem bei dem Kontrast zu sehen. Bei der 1.Person ist die Farbe des Hemdes ähnlich wie die Farbe des Lichtes im Raum. Die 5. Person hat ein T-Shirt an, bei dem die Schultern frei sind. In der 1. Hälfte der Videosequenz war der Hintergund hellblau und deshalb wurden die meisten Oberkörper erkannt. Im Gegensatz dazu war der Hintergrund in der 2.Hälfte der Videosequenz eine Tafel, deren Farbe zur

Abb. 4.1: Ein Beispielbild mit detektieren Oberkörper im 1. Datensatz

Hautfarbe ähnlich ist, und die meiste Zeit wurde der Oberkörper nicht erkannt. Es lässt sich feststellen, dass zu geringer Kontrast zwischen Vorder- und Hintergund eine Ursache für schlechte Erkennungsraten ist. Eine mögliche Lösung dafür ist, so viele Bilder wie möglich in verschiedener Helligkeit und Kontrast zum Trainieren zu verwenden.

HOG

Die Paramter des 2. Systems werden wie folgt angepasst:

- Jede Zelle hat die Auflösung 4x4 Pixel ($\eta = 4$)
- Jeder Block besitzt 2x2 Zellen ($\varsigma = 2$)
- Die Blockverschiebung beträgt 4 Pixel, d.h. 50 Prozent des Blocks wird überlappt
- Die Orientierung jedes Pixels wird gleichmäßig 9 Behältern von 0 bis 180 Grad zugeordnet ($\beta = 9$)
- L2Hys als Normalisierungsmethode
- RGB_sqrt als Gradientenberechnungsmethode

Anhand der Ergebnissen von Dalal & Triggs wird gezeigt, dass $\beta = 9$, L2Hys, RGB_sqrt und Blocküberlappung = 50 % die passende Paramenter für die Erkennung von Menschen sind. In den Experimenten von Dalal & Triggs haben die Suchfenster eine Auflösung von 128x64 Pixel und deshalb werden die Parameters $\eta = 8$ und $\varsigma = 2$ verwendet. In unserem Fall hat das Suchfenster eine kleinere Auflösung (80x80 Pixel) und damit auch ein kleineres η (=4).
In der Trainingsphase wurden alle Klassifikatoren sequenziell auf einem Quardcore Computer trainiert, wobei wegen der Nichtparallelität der Berechnung nur ein Core verwendet wurde. Die gesamte Trainingszeit beträgt 5 Minuten. Die Ergebnisse der Testphase sind in Tabelle 4.3 dargestellt.

Iteration	1	2	3	4	5	6	7	Gesatmergebnis
Hitrate	0.89	0.93	0.85	1	0.92	0.94	1	0.93
Falschalarmrate	0.24	0.26	0.27	0.49	0.11	0.29	0.27	0.3

Tabelle 4.3: Ergebnis des 2.Systems mit dem 1.Datensatz

Die maximal erreichte Hitrate des 2. Systems ist 100 %, während die minimale Falschalarmrate 0.11 pro Bild ist. Die durchschnittliche Hitrate ist 93 % und die Falschalarmrate ist 0.3 pro Bild. Ein Beispiel-Bild ist in der Abbildung 4.2 zu sehen. Die meisten falsch detektierten Fenster sind Oberkörper, die nicht in Frontalansicht sondern in davon verschiedenen Orientierungen sich befinden.

Abb. 4.2: Beispielbild: Mit 2.System detektierter Oberkörper des 1. Datensatzes

Bei der 4. Person ist die Falschalarmrate etwas höher als bei den anderen, da ein falscher Alarm mehrere Male in der Bildsequenz evaluiert wurde.

4.3.2 2. Datensatz

Bei dem 2.Datensatz lassen sich die Ergebnisse durch Konfusionsmatrizen darstellen. Für beide Ansätze werden genau 5 verschiedene Klassifikatoren für verschiedene Ansichten des Oberkörpers (Orientierung von 0. 45. 90. 135 und 180 Grad) trainiert und evaluiert. Um Oberkörper mit negativer Orientierung zu detektieren, werden die Bilder um die in der Mitte des Bildes liegende y-Achse rotiert und lassen sich damit durch die Klassifikatoren für 45, 90 und 135 Grad Orietierungen erkennen. Am Ende werden die Ergebnisse der verschiedenen Klassifikatoren zusammen getragen.

Klassifikatorkaskaden

Für den komplexen 2.Datensatz werden die Parameter des 1.Systems folgendermaßen angepasst:

- Alle Oberkörper werden auf der Größe 20x20 skaliert.
- Jeder schwache Klassikator verwendet eine Baumstruktur mit 2 Terminalknoten.
- Der gesamte Klassifikator wird aus 17 Stage aufgebaut.
- Minimal zu erreichende Hitrate pro Stage ist 0.995.
- Maximal erlaubte Falschalarmrate pro Stage ist 0.5.
- Gentle Adaboost wird verwendet
- Erweiterte Merkmale werden verwendet.

Da der 2.Datensatz viel komplexer als der 1. ist, verwendet jeder schwache Klassifikator des 1.Systems eine Baumstruktur mit 2 Terminalknoten. Damit spielt bei schwachen Klassifikatoren nicht nur ein Merkmal allein sondern auch die Kombination von 2 Merkmalen eine Rolle bei der Entscheidung. Im Laufe der Experimente hat sich gezeigt, das ein System mit Baumstruktur eine bessere Hitrate liefert.
Bei der Testphase werden die Testbilder 1.5-fach verkleinert. Der Grund dafür liegt in der Trainingsphase, in der die Oberköperbilder auf 20x20 skaliert und damit auch geglättet werden. Der entscheidende Punkt dabei ist, dass damit auch das Testbild geglättet wird. Die Ergebnissse der Experimente haben gezeigt, dass die Hitrate tatsächlich verschlechtert wird, falls das originale Testbild ohne Verkleinerung direkt evaluiert wird. Die Mindestgröße des Suchfensters beträgt 40x40 Pixel, der Skalierungsfaktor ist 1.01 und '*min_neighbors* ist 3. Mit diesen Parametern wurden 5 Klassifikatoren parallel auf 5 verschiedenen Quadcore-Computer trainiert. Durchschnittlich dauert das Training 20 Stunden. Das Gesamtergebnis, bei dem den Trainingsbildern zusätzlicher Rand von 20 % beigefügt wurde, wird als Konfusionmatrix in der Tabelle 4.4 dargestellt. Bei der Testphase benötigt das System pro Bild mit einer Auflösung von 640x480 Pixel 0.47 s. Insgesamt haben die 5 verschiedenen Klassifikatoren 568 von 710 Oberkörpern (d.h. 80 %) und 2.5 falsche Alarme pro Bild detektiert. Eine mögliche Szene ist in der Abbildung 4.3 zu sehen.
In dieser Arbeit wurde auch ein Leistungsvergleich zwischen zwei verschiedenen Randgrößen von 10 % bzw. 20 % Rand durchgeführt. Die gesamten Ergebnisse sind in der Tabelle 4.5 zu sehen. Hieraus lässt sich folgern, dass Klassifikatoren, die Trainingsbilder mit 20 % Rand zum Trainieren verwenden, eine bessere Hitrate von bis zu 5 % mit fast gleicher Falschalarmrate haben. Deshalb lässt sich feststellen, dass die Randgröße der Trainingsbilder einen Einfluss auf die Leistung der Klassifikatoren hat.

Klasse	0	45	90	135	180	-135	-90	-45	-
0	.68	.26	.31	.12	.26	.22	.37	.26	.70
45	.36	.51	.41	.12	.10	.22	.36	.35	.81
90	.14	.25	.53	.09	.13	.17	.35	.11	.40
135	.14	.16	.17	.25	.37	.53	.37	.12	.74
180	.07	.11	.10	.12	.49	.20	.16	.04	.19
-135	.17	.16	.25	.14	.39	.56	.51	.15	.70
-90	.14	.12	.31	.24	.21	.43	.46	.33	.51
-45	.32	.22	.27	.13	.62	.40	.63	.65	.60
gesamte Hitrate	.85	.70	.88	.60	.89	.90	.86	.81	

Tabelle 4.4: Ergebnis des 1.Systems mit dem 2.Datensatz

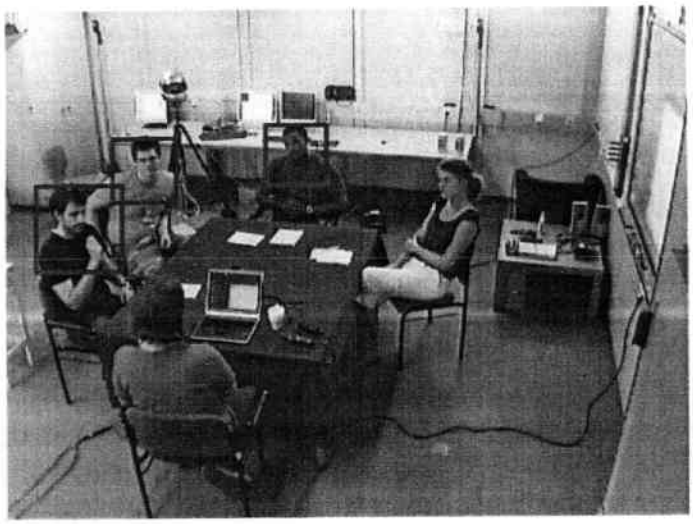

Abb. 4.3: Beispielbild: Mit 1.System detektierter Oberkörper des 2. Datensatzes

Randgröße der Trainingsbilder	10 %	20 %
Hitrate	.75	.80
Falschalarmrate	2.4	.2.5

Tabelle 4.5: Gesamtergebnis des 1.Systems mit dem 2.Datensatz und 2 verschiedenen Randgrößen.

Klasse	0	45	90	135	180	-135	-90	-45	-
0	.83	.30	.08	.1	.35	.2	.31	.33	.55
45	.46	.55	.5	.31	.26	.32	.17	.43	.52
90	.24	.42	.86	.32	.18	.05	.26	.08	.52
135	.2	.25	.42	.48	.39	.08	.2	.21	.44
180	.43	.22	.1	.22	.49	.32	.17	.28	.48
-135	.16	.12	.08	.09	.33	.77	.37	.26	.42
-90	.17	.09	.1	.09	.24	.46	.71	.44	.43
-45	.51	.18	.14	.16	.42	.52	.57	.72	.64
gesamte Hitrate	.92	.66	.88	.63	.82	.87	.83	.87	

Tabelle 4.6: Ergebnis des 2.Systems mit dem 2.Datensatz

HOG

Die Parameter des 2.Systems werden angepasst, so dass das System wie folgt funktioniert:

- Jede Zelle hat eine Auflösung von 4x4 Pixels.
- Jeder Block besitzt 2x2 Zellen.
- Die Blockverschiebung ist 4 Pixel, d.h. 50 Prozent der Blocks wird überlappt.
- Die Orientierung jedes Pixels wird gleichmäßig 9 Behältern von 0 bis 180 Grad zugeordnet.
- L2 als Normalisierungsmethode
- RGB_sqrt als Gradientenberechnungsmethode

Bei der Testphase hat das 2.System folgende Parameter:

- Mindestgröße des Suchfensters ist 80x80 Pixel
- Verschiebung des Suchfensters ist 4 Pixels
- Skalierungsfaktor ist 1.2
- Endfaktor der Skalierung ist 4

Mit den oben genannten Parametern lieferte das 2.System eine Hitrate von 80.9 % und 1.9 falsche Alarme pro Bild. Das detaillierte Ergebnis ist in der Tabelle 4.6 zu sehen.
Um die Falschalarmrate zu erniedrigen, wurde das System nach der Traingsphase noch einmal trainiert. Diese 2.Trainingstteration wurde schrittweise folgendermaßen durchgeführt:

- 1.Schritt: Vorbereitung von Hintergrundbildern, in denen schwierige Negativbeispiele gesucht werden.

Klasse	0	45	90	135	180	-45	-90	-135	-
0	.72	.22	.04	.05	.13	.24	.25	.28	.31
45	.64	.53	.38	.2	.22	.19	.22	.41	.31
90	.51	.54	.88	.34	.18	.06	.19	.31	.32
135	.29	.26	.44	.55	.43	.19	.34	.26	.21
180	.49	.15	.08	.16	.39	.19	.25	.26	.18
-135	.3	.20	.14	.1	.33	.66	.46	.3	.33
-90	.4	.16	.14	.09	.26	.44	.74	.53	.24
-45	.5	.22	.18	.11	.26	.39	.49	.72	.28
gesamte Hitrate	.91	.72	.88	.66	.74	.83	.80	0.92	

Tabelle 4.7: Ergebnis des 2.Systems mit dem 2.Datensatz nach dem 2.Training

Abb. 4.4: Beispielbild: Mit 2.System detektierten Oberkörpern des 2.Datensatzes

- 2.Schritt: Die zuvor trainierten Klassifikatoren scannen alle Hintergrundbilder.
- 3.Schritt: Da auf den Bilder nur Hintergrund zu sehen ist, sind alle detektierten Fenster falsche Alarme. Diese Fenster werden aus den Bildern ausgeschnitten und als Negativbeispiele bei der 2.Iteration des Trainings verwendet.
- 4.Schritt: Das System wird noch einmal mit den neuen schwierigen Negativbeispiele trainiert.

Nach dem Trainieren wurde das System mit den Testdaten evaluiert und lieferte das in der Tabelle 4.7 zu sehende Ergebnis.
Insgesamt wurden 573 von 710 Oberkörpern (d.h. 80,7 %) und 0.9 falsche Alarme pro Bild detektiert. Eine mögliche Szene ist in Abbildung 4.4 zu sehen. Für jedes Bild mit einer Auflösung von 640x480 Pixeln benötigt das System 2.8s mit einem Core des Quadcore-Computers. Genau wie beim 1.System wurde hier mit verschiedenen Randgrößen experimentiert. Das gesamte Ergebnis ist in der Tabelle 4.8 zu sehen. Das Ergebnis zeigt, dass bei dem 2.System die Randgröße der

Randgröße der Trainingsbilder	10 %	20 %
Hitrate	.77	.80
Falschalarmrate	1.1	0.9

Tabelle 4.8: Gesamtergebnis des 2.Systems mit dem 2.Datensatz und 2 verschiedenen Randgrößen.

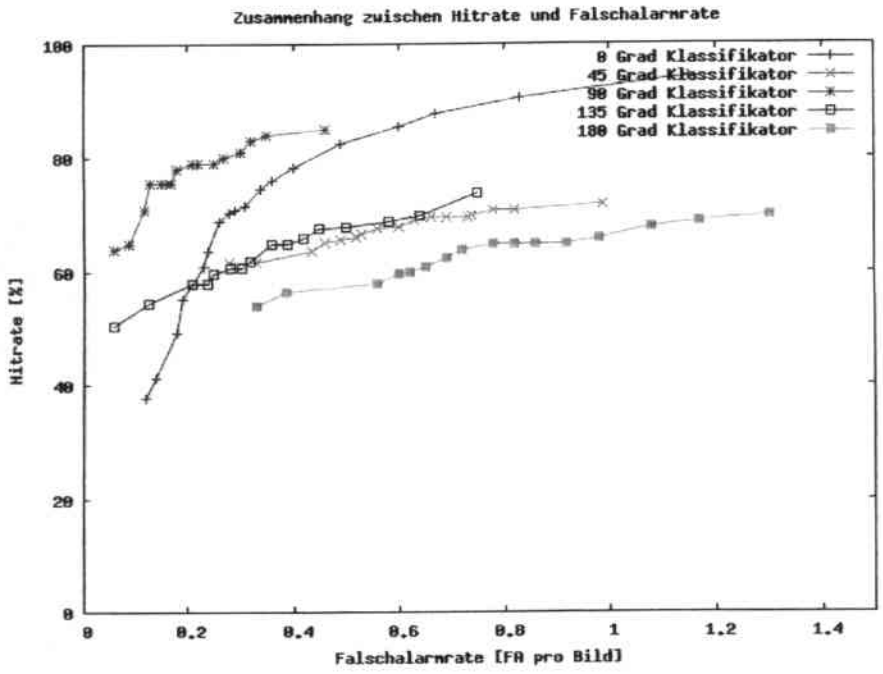

Abb. 4.5: Zusammenhang zwischen Hitrate und Falschalarmrate durch die Änderung des Schwellwertes der Klassifikatoren

positiven Trainingsbilder eine Rolle spielt. Bei einer Randgröße von 20% ist die Hitrate 3 % höher und die Falschalarmrate 0.2 falscher Alarm pro Bild kleiner als bei der Randgröße von 10 %.
Außerdem spielt beim HOG System der Schwellwert der Klassifikatoren eine große Rolle. Es gilt, je größer der Schwellwert, desto kleiner sowohl die Falschalarmrate als auch die Hitrate. Durch einen gut gewählten Schwellwert ergibt sich ein gutes Ergebnis. In der Abbildung 4.5 ist der relative Zusammenhang zwischen Hitrate und Falschalarmrate durch die Änderung des Schwellwerts des Klassifikators zu sehen.

4.4 Analyse

Die Detektion von Oberkörpern stellt eine interesssante Lösung für verschiedene weitere Aufgaben dar, wie zum Beispiel die Detektion von Körperdrehungen. Anhand der Ergebnisse kann man sehen, dass beide Ansätze mögliche Lösungen für die Problemstellung sind. Außerdem zeigte sich, dass die Oberkörperdetektion mit verschiedenen Orientierungen und Oberkörperdrehungschätzungen besonders schwierig ist, wenn sich in der Umgebung viele andere Dinge oder sogar andere Körper befinden. In dieser Situation existiert kein festes Muster und damit ist die Wahrscheinlichkeit einer Verwechselung mit den Objetkten in der Umgebung relativ hoch. Beim 1.Datensatz, bei dem nur Frontaloberköper detektiert werden und die Daten ziemlich einfach sind, funktioniert das 2.System ganz hervorragend. Es lieferte eine Hitrate von 93 % mit 0.3 falschen Alarmen pro Bild. Im Vergleich dazu lieferte das 1.System in 5 von 7 Fällen auch gute Ergebnisse. Die Hitraten liegen zwischen 75 % und 89 % mit einer Falschalarmrate von 0.28 pro Bild. Die anderen 2 Fällen haben nur eine Hitrate von 30 % bzw. 50 %. Das Problem liegt darin, dass die Anzahl von Trainingsdaten relativ gering ist. Deshalb ist das System nicht robust gegen große Änderungen des Kontrastes. Durch die Ergebnisse des 1.Datensatzes lässt sich feststellen, dass das 2.System flexibler als das 1.System ist. Mit einer gleichen, kleinen Menge von Trainingsdaten lieferte das 2.System eine bessere Leistung.
Im Vergleich zum 1.Experiment ist das 2. Experiment deutlich komplexer. Die Traingsbilder bzw. Testbilder beim 2.Datensatz sind variabler, da sich die Oberkorper in verschiedenen Orientierungen befinden. Aus diesem Grund gibt es kein festes Muster für alle Oberkörper und deshalb ist es schwierig mit dem klassischen Ansatz, eine gute Hitrate für Oberkörperdetektion bei diesem Datensatz zu erreichen. Durch die Kombination von 5 verschiedenen Klassifikatoren erreichen die beiden Systeme eine Hitrate von ungefähr 80 %, aber die Falschlarmrate akkumuliert sich ebenfalls. Mit der Klassifikatorkaskade ist das bestmöglich Ergebnis 80 % Hitrate und 2.5 falsche Alarme pro Bild. Im Vergleich dazu hat das auf HOG basierte System als bestes Ergebnis 80.9 % Hitrate und 1.9 falsche Alarme pro Bild. Die wichtigsten Gründe, weshalb die Falschalarmrate relativ hoch ist, sind folgende:

- Die Anzahl der Trainingsdaten ist nicht genügend, da der Aufwand der Annotation der Daten zum Trainieren sehr groß ist.
- Die Oberkörper überlappen sich in vielen Bildern.
- Der Mensch ist beweglich, während die andere Objekte im Raum statisch sind. D.h. falls eines von solchen statischen Objekten als Oberkörper erkannt wird, ist es in der gesamten Bildsequenz ein falscher Alarm. Ein Beispiel dafür ist in der Abbildung 4.6 zu sehen.

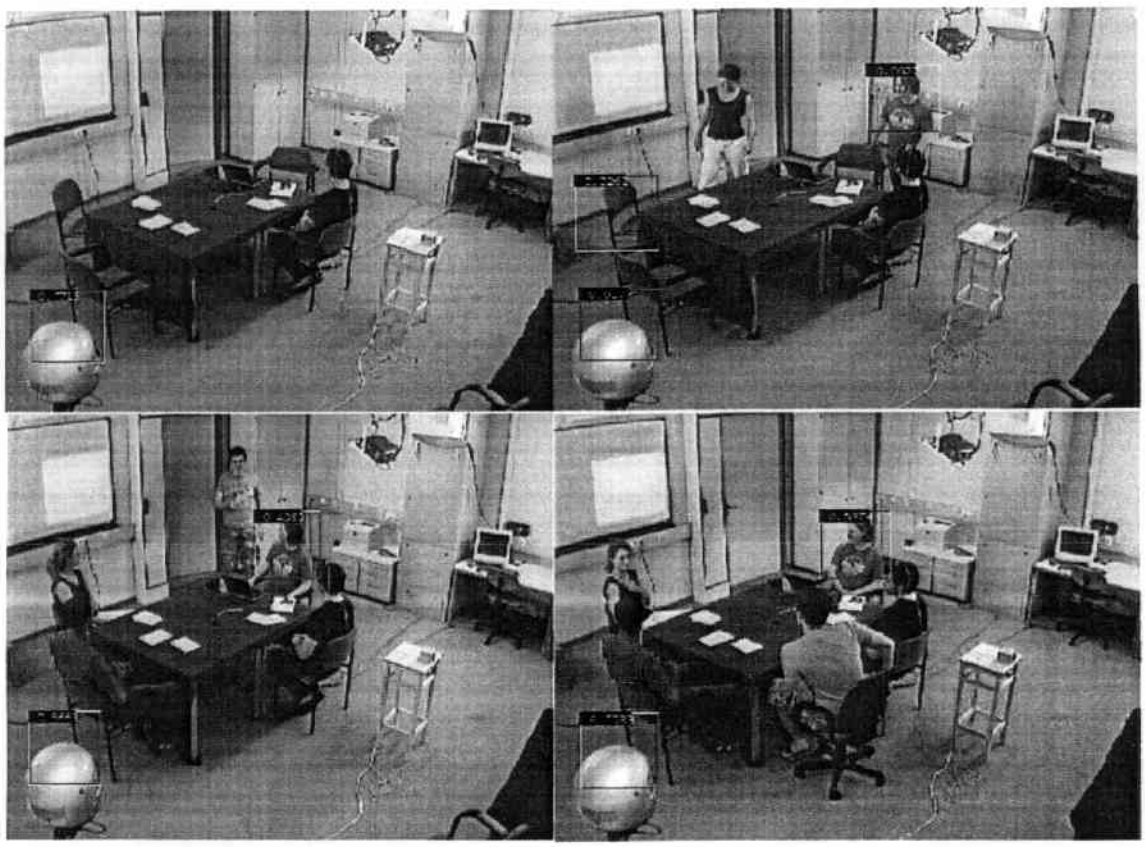

Abb. 4.6: Ein statisches Objekt wurde als Oberkörper in der ganzen Bildsequenz erkannt.

Unter der Bedingung, dass ein statisches Objekt bei mehrmaliger Erkennung nur einmal als falscher Alarm in das Gesamtergebnis einbezogen wird, erscheint die Leistung der Systeme nochmals besser. Das unter dieser Bedingung erhaltene Ergebnis ist in der Tabelle 4.9 zu sehen.

Ergebnis	Hitrate	Falschalarmrate
1.System	.80	0.6
2.System	.80	0.38

Tabelle 4.9: Gesamtergebnis des 1. und 2. Systems unter der Bedingung, dass statische Objekte nur einmal innerhalb einer Bildsequenz als falscher Alarm gezählt werden.

Anhand der Ergebnisse ist die Leistung des 2.Systems besser als die des 1.Systems. Allerdings ist das 2.System deutlich langsamer. Während das 2.System pro Bild mit dem schon oben erwähnten Rechner 2.58s benötigt, ist die Detektionszeit des 1.Systems pro Bild nur durchschnittlich 470 ms. Allerdings ist die Trainingszeit des 1.Systems deutlich länger. Das ist der Grund, weshalb in diesem Fall die positiven Trainingsdaten auf die Größe 20x20 eingestellt werden müssen. Im Gegensatz dazu ist die Trainingszeit des 2.Systems viel kürzer und die Trainingsdaten können auf die durschnittliche Oberköprergröße von 80x80 eingestellt werden.

5 Zusammenfassung und Ausblick

In vorliegender Arbeit wurde ein relativer Vergleich zwischen dem Haarkaskaden-Ansatz von Viola&Jones und den HOG-Ansatz von Dalal&Triggs für die Oberkörperdetektion durchgeführt, indem 2 vorhandenen Implementierungen mit angepassten Parametern experimentiert wurden. Dies sind zum einen die Haar-Feature Klassifikatorkaskadentools der OpenCV Bibliothek und zum anderen INRIA Object Detection and Localization Toolkit. Beide wurden zur Oberkörperdetektion in verschiedenen Orientierungen trainiert und evaluiert. 2 Datensätze mit steigender Komplexität von verschiedenen Videosequenzen wurden verwendet, um die Trainingsdaten und Testdaten zu extrahieren.
Anhand der Ergebnisse zeigte sich, dass beide Ansätze mögliche Lösungen für die Problemstellung darstellen. Jedes besitzt eigene Vor- und Nachteile. Das auf Haarkaskaden basierende System ist zwar schneller, hat aber die schlechteren Ergebnisse und deutlich längere Trainingszeiten. Im Gegensatz dazu hat das auf HOG basierende System bessere Hitraten mit ausreichend niedrigen Falschalarmraten und eine lange Detektionzeit. Aus diesen Gründen ist es nötig, ein System zu entwickeln, das die Vorteile von beiden oben genannten Systemen hat und die Nachteile eliminiert.
Zukünftige Arbeiten sollten sich deshalb mit folgenden möglichen Verbesserungsmöglichkeiten beschäftigen:
Klassifikation: In der Implemtierung von Dalal&Triggs werden nur lineare Kernelfunktionen unterstüzt. Das Ergebnis von Dalal&Triggs zeigt auch, dass mit anderen Kernelfunktionen mögliche Verbesserungen entstehen. Dafür erhöht man aber die Laufzeit.
Laufzeit: Das Ergebnis von Dalal&Triggs Implementierung auf unseren Datensätzen ist relative gut. Aber im Vergleich zu Haarkaskaden ist es leider nicht echtzeitfähig. Was verbessert werden könnte, wäre die Methode zur Berechnung der Merkmale.
Informationsfusion: In dieser Studienarbeit wurde das 2.Experiment in einem Smartroom durchgeführt, in dem für die Betrachtung 4 Kameras mit fester Position genutzt wurden. Das bedeutet, dass die Informationen von 4 Kameras zusammengeführt werden könnten. Dies wurde hier nicht gemacht, aber es wäre eine möglicher Ansatz, um die Falschalarmrate zu erniedrigen.

A Verwendete Parameterkonfigurationen

Die wichtigsten verwendeten Parameter, die für das Tranining der 2 oben genannten Systeme verwendet wurden, sollen in folgenden kurz aufgelistet werden.

A.1 OpenCV Parameter

- Anzahl von Stage -nstage 15
- maximale Falschalarmrate pro Stage -maxfalsealarm 0.5
- minimale Hitrate pro Stage -minhitrate 0.995
- schwacher Klassifikator verwendet nur einen Merkmal -nsplit 1

A.2 HOG Parameter

A.2.1 Parameter für Training

- die Breite von Traingsbildern -width 80
- die Höhe von Trainingsbildern -height 80
- die Auflösung von Cells -C 4,4
- die Anzahl von Cells pro Block -N 2,2
- die Anzahl von Bins -B 9
- die Blocksverschiebung -G 4,4
- Die Methode zur Gradientenberechnung –proc rgb_sqrt
- Art von Blocknormalisierung –norm l2Hys

A.2.2 Parameter für die Detektion

- Der Schwellwert des Klassifikators -t 0.1
- Skalierungsfaktor –scaleratio 1.1
- Verschiebung des Suchfensters –winstride 8
- Maximale Skalierungsfaktor –endscale 2.2
- Anfangskalierungsfaktor –startscale 1
- Größe von Rand –margin 0,0
- kleinste Größe von Fenster–avsize 60,60

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Paul Viola and Michael Jones, *Robust Realtime Face Detection*, International Journal of Computer Vision, 2003.

[2] Paul Viola amd Michael Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features*, International Journal of Computer Vision, 2001.

[3] Rainer Lienhart and Jochen Maydt, *An Extended Set of Haar-like Features for Rapid Object Detection*, IEEE International Conference on Image Processing, 2002.

[4] Rainer Lienhart, Alexander Kuranov, Vadim Pisarevsky, *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection*, DAGM'03, 25th Pattern Recognition Symposium, Madgeburg, Germany, 2003.

[5] Stan Z. Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum, *Statistical Learning of Multi-View Face Detection*, In Proceedings of The 7th European Conference on Computer Vision. Copenhagen, Denmark, 2002.

[6] Kobi Levi, Yair Weiss, *Learning Object Detection from a Small Number of Examples: the Importance of Good Features*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004

[7] Jos Barreto, Paulo Menezest, Jorge Dias, *Human-Robot Interaction based on Haar-like Features and Eigenfaces*, International Conference on Robotics and Automation, 2004.

[8] Yoav Freund Robert and E.Schapire, *A Short Introduction to Boosting*, Journal of Japanese Society for Artificial Intelligence, 1999.

[9] N.Dalal and B.Triggs *Histograms of oriented gradients for human detection*, Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[10] Qiang Zhu, Shai Avidan, Mei-Chen Yeh and Kwang-Ting Cheng, *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients*, International Conference on Computer Vision and Pattern Recognition, 2006.

[11] J.C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, Vol. 2, Number 2, p. 121-167, Kluwer Academic Publishers, 1998.

[12] Open source computer vision library, `http://www.intel.com/technology/computing/opencv/index.htm`.

[13] SVM Light Support Vector Machine
`http://svmlight.joachims.org/`

[14] Learning and Recognition in Vision
`http://lear.inrialpes.fr/index.php`

[15] MIT Pedestrian Database
`http://cbcl.mit.edu/software-datasets/PedestrianData.html`