Institut für Visualisierung und Interaktive Systeme
Abteilung Intelligente Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3043

Hierarchische Klassifikation von Fahrzeugansichten

Michael Gabb

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Gunther Heidemann
Betreuer:	Dr.-Ing. Otto Löhlein Dipl.-Inf. Matthias Oberländer
begonnen am:	16. Juni 2010
beendet am:	16. Dezember 2010
CR-Klassifikation:	I.5.2, I.5.4, G.1.6

Sperrvermerk

Diese Diplomarbeit ist mit Einverständnis der Beteiligten bis zum 16. Dezember 2013 für die Veröffentlichung und Weitergabe an Dritte gesperrt.

Zusammenfassung

Automobile Assistenzsysteme bergen großes Potential bei der Vermeidung von Unfällen. Um effektiv und effizient arbeiten zu können, benötigen sie Informationen über ihre Umgebung. Von zentraler Bedeutung ist hierbei das Wissen um die Existenz, Lage und Bewegung anderer Verkehrsteilnehmer.

In dieser Arbeit wird ein echtzeitfähiges System vorgestellt, welches in monokularen Einzelbildern einer hinter der Windschutzscheibe angebrachten Kamera andere Fahrzeuge aller Orientierungen detektiert (*Multi-View Vehicle Detection*). Neben der Existenz der Fahrzeuge wird zusätzlich ihre Orientierung mit hoher Genauigkeit eingeschätzt. Ausgangspunkt hierfür bildet die von Viola und Jones vorgeschlagene Kaskadenstruktur, welche auf einen allgemeinen Klassifikationsbaum erweitert wird.

Der entwickelte *Decision-Boosted Cluster Boosted Tree* (DB-CBT) nimmt im Gegensatz zur Kaskadenstruktur eine weitergehende Trennung des Eingaberaums vor. Auf diese Weise wird die Geschwindigkeit des Klassifikators erhöht, ohne Einbußen bei der Detektionsleistung hinnehmen zu müssen. Der Baumaufbau erfolgt während des Trainings direkt auf das Klassifikationsproblem ausgerichtet, d. h. der gesamte Baum wird so aufgebaut, dass die Klassifikationsaufgabe mit möglichst hoher Geschwindigkeit gelöst werden kann. Hierfür wird ein Gütemaß definiert, welches die Eignung einer Trennung des Eingaberaums misst. Anhand diesem wird die Trennung gewählt, so dass eine schnellere Konvergenz von AdaBoost möglich ist. Um die Geschwindigkeit zusätzlich zu erhöhen, wird durch einen zweistufigen Boosting-Prozess (*Decision-Boosting*) die Anzahl zu traversierender Knoten des DB-CBT minimiert.

Die Vorteile des Ansatzes werden experimentell bestätigt. Bei vergleichbarer Detektionsleistung ist die erwartete Laufzeit je Hypothese um 25 %, die maximale Laufzeit um 64 % geringer als bei Einsatz eines Kaskadenklassifikators.

Danksagung

Ich möchte allen danken, die zum Gelingen dieser Diplomarbeit beigetragen haben.

Mein besonderer Dank gilt meinen Betreuern Dr. Otto Löhlein und Matthias Oberländer. Beide standen trotz voller Terminkalender stets für Fragen, Hilfestellungen und Diskussionen zur Verfügung.

Meinem Prüfer Prof. Dr. Gunther Heidemann danke ich dafür, dass er sich erklärt hat, diese externe Diplomarbeit zu prüfen sowie für seine interessanten Vorlesungen „Statistical Data Mining“ und „Computer Vision“, die mein Interesse für das Thema weckten.

Roland Schweiger danke ich für seine zahlreichen Hinweise und Ratschläge fachlicher und menschlicher Art; Markus Gressmann danke ich für die Hilfestellungen bei der Integration der EOH-Merkmale, seine Hilfe mit dem MSC-Framework und die immer offene Tür bei technischen Fragen.

Nikolas Engelhart, Raimar Wagner, Sebastian Klenk und Ferdinand Deger danke ich, dass sie sich bereit erklärt haben, Teile dieser Diplomarbeit gegenzulesen.

Meinen Geschwistern und Eltern danke ich, dass sie mich jederzeit unterstützt haben und somit mein Studium und diese Diplomarbeit erst ermöglichten.

Besonders danke ich Jessica Schmalhofer für ihre Unterstützung und Aufmunterung während der gesamten Diplomarbeit.

Inhaltsverzeichnis

Symbolverzeichnis

Allgemein

$\mathcal{C} = \{-1, 1\}$	Menge der Klassen. „-1“ ist Hintergrund, „1“ ist Objekt.
$\mathcal{M}$	Merkmalsraum.
$\mathcal{X}$	Eingaberaum.
X, Y, Z	Weltkoordinaten in Metern.
x, y	Spalte bzw. Zeile im Eingabebild, beginnend in der linken, oberen Ecke.

Aspektwinkelschätzung

$\|.\|$	L_2-Norm (Euklidischer Abstand).
Γ	Schwellwert der Detektionsassoziation.
Φ, Θ, Ψ	Roll-, Nick- und Gierwinkel.
α, β	Gesuchter Gierwinkel bzw. Gegenwinkel des gesuchten Gierwinkels.
γ_1, γ_2	Winkel, unter denen das Fahrzeug gesehen wird.
A, B, C, D	Maßgebende Fahrzeugkanten bzw. Eckpunkte des Fahrzeugs.
A', C'	Bildpunkte der Fahrzeugeckpunkte A bzw. C.
$\underline{a}$, $\underline{b}$, $\underline{c}$	Ortsvektoren der Punkte A', B, C'.
y_i	i-te Komponente des Vektors $\underline{y}$.
d	Distanz des Fahrzeugs zur Kamera (in Metern).
w, l	Breite bzw. Länge des Fahrzeugs (in Metern).

AdaBoost und Kaskadenklassifikator

α_t	Gewicht des in Runde t gewählten Weaklearners.
θ	Akzeptanzschwelle des Stronglearners.
D	Detektionsrate.
F	Falschalarmrate.
$H_T(.)$	Stronglearner.
$W_j^{\pm}$	Gewichteter Anteil an positiven/negativen Beispielen in der j-ten Teilmenge des Eingaberaums.
$Z(f)$	Z-Wert des Weaklearners f.
$h_t(.)$	In Runde t gewählter Weaklearner.
$w_t(j)$	Gewicht des i-ten Beispiels in Runde t.

Decision-Boosted Cluster Boosted Tree

η	Mindestanzahl Positivbeispiele für eine Trennung des Eingaberaums.
$\lambda(.)$	Regularisierung der Gütefunktion e$(.)$.
τ_i	Schwellwerte des Z-Werts.
bhat(p_a, p_b)	Bhattacharyya-Koeffizient der Wahrscheinlichkeitsdichten p_a und p_b.
e$(.)$	Gütefunktion zur Bewertung einer Zerlegung.
sep(A, B)	Separierbarkeit zweier Mengen A und B.

Experimentelle Ergebnisse

σ_{aff}	Ähnlichkeitsschwellwert.
$d(.,.)$	Distanzmaß für Winkelkategorien.
d_{max}	Maximal zulässige Abweichung der Winkelkategorie (Toleranz).

KAPITEL 1

Einleitung

> The hardest single part of building a software system is deciding precisely what to build.
>
> *(Fred Brooks, No Silver Bullet)*

Dieses Kapitel dient als Einleitung und gibt einen Überblick über diese Arbeit. Zunächst wird die Problemstellung motiviert (Abschnitt 1.1) und die Ziele dieser Arbeit zusammengefasst (Abschnitt 1.2). Im Anschluss daran werden grundlegende Begriffe definiert (Abschnitt 1.3), ein Überblick über das entwickelte System gegeben (Abschnitt 1.4) und die Ausgangsbasis sowie die vorhandenen Hilfsmittel bei der Anfertigung dieser Arbeit erläutert (Abschnitt 1.5). Abschließend wird ein Überblick über die folgenden Kapitel gegeben (Abschnitt 1.6).

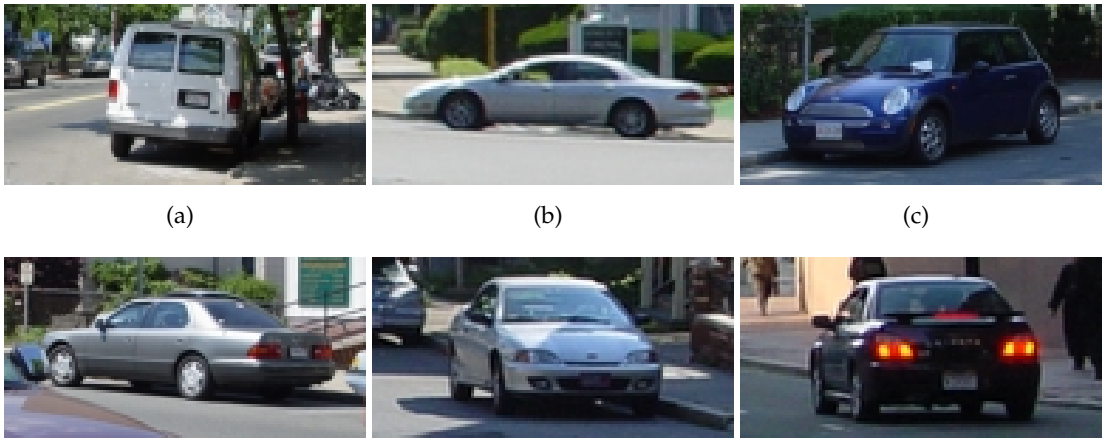
1.1. Motivation

Im Jahr 2008 wurden in Deutschland über 409 000 Personen bei Verkehrsunfällen verletzt, 4 477 verletzten sich dabei tödlich [Sta09, S. 440 ff.]. Weltweit sterben jährlich über 1 Mio. Menschen an den Folgen von Verkehrsunfällen. Ein Großteil der Unfälle hätte verhindert oder in den Folgen abgemildert werden können, wenn geeignete Notmanöver wenige Augenblicke früher eingeleitet worden wären [Sti07].

Assistenzsysteme, die den Fahrer eines Fahrzeugs bei seiner Aufgabe unterstützen und ihm so helfen, in allen Situationen adäquat und schnell zu reagieren, bergen großes Potential bei der Vermeidung von Unfällen. Um effektiv und effizient arbeiten zu können, benötigen sie eine Vielzahl von Informationen von verschiedenen Sensoren, mit deren Hilfe sie ein umfassendes Umgebungsmodell erstellen können [Wen08],[Wal08].

Da Fahrer einen Großteil der Informationen visuell, d. h. mit den Augen, aufnehmen, ist es naheliegend, diesen Informationskanal automobilen Assistenzsystemen zur Verfügung zu stellen. Die Erkennung anderer Fahrzeuge in Videoströmen ist hierbei eine der Kernaufgaben der Systeme. Neben der Erkennung der reinen *Existenz* anderer Fahrzeuge ist auch die korrekte Einschätzung ihrer *Lage* und *Bewegung* wichtig. Durch Kenntnis von Lage und Bewegung sind

Rückschlüsse über den möglichen Weg des erkannten Fahrzeugs und eventuelle Konsequenzen daraus (bspw. einen bevorstehenden Zusammenstoß) möglich, was bessere Reaktionen des Systems erlaubt. Damit diese Reaktionen rechtzeitig erfolgen können, wird im automobilen Umfeld die Echtzeitfähigkeit der Systeme gefordert [SZ06]. Da Fehler während der Fahrt fatale Folgen haben können, sind zudem die Anforderungen an die Robustheit und Zuverlässigkeit von Assistenzsystemen sehr viel höher als in anderen Bereichen wie bspw. der Consumer-Elektronik [Gre08, S. 1 f.].



(a) (b) (c)

(d) (e) (f)

Abbildung 1.1.: Beispiele für Ansichten verschiedener PKW, die ein Assistenzsystem erkennen muss. Die Variabilität zwischen den einzelnen Ansichten ist klar sichtbar. Quelle: Bilddatenbank von Kuo und Nevatia [KN09].

Um die Lage erkannter Fahrzeuge korrekt zu beurteilen, müssen sie unter allen Aspektwinkeln (siehe Abb. 1.1 für einige Beispiele) durch das Assistenzsystem erkannt werden. Für die Erkennung von Objekten in Videoströmen ist im automobilen Umfeld die Klassifikation mit dem von Viola und Jones vorgeschlagenen Kaskadenklassifikator [VJ01] – obwohl ursprünglich für die Gesichtserkennung konzipiert – weit verbreitet, da er hohe Geschwindigkeit mit hoher Detektionsleistung verbindet. Beispiele für einen erfolgreichen Einsatz sind die Erkennung von Insassen in einem Auto mit Hilfe einer omnidirektionalen Kamera [Wen03],[WL04] sowie die Detektion von Fußgängern in Infrarotbildern [MOL+05]. Ist die Variabilität in den zu detektierenden Objekten allerdings groß, reicht die Kaskadenstruktur nicht mehr aus [ZZ10],[HALL07]. Hier erweisen sich Baumstrukturen als flexibler und leistungsfähiger [Wen03, S. 79 ff.],[Chr07, S. 7],[Wu08, S. 25 f.],[PSR10]. Bspw. mussten Viola und Jones ihren ursprünglichen Ansatz auf eine Baumstruktur erweitern, um unterschiedliche Profile erkennen zu können (sog. *Multi-view Face Detection*) [JV03].

Um eine ausreichende Detektionsleistung sicherzustellen, bietet es sich also an, auch bei der optischen Lageschätzung von Fahrzeugen einen baumbasierten Ansatz zu wählen. Gleichzeitig muss die hohe Geschwindigkeit der Kaskadenstruktur erhalten bleiben, um die Echtzeitanforderungen an automobile Assistenzsysteme zu erfüllen. In dieser Arbeit wird ein solcher baumbasierter Ansatz entwickelt und vorgestellt sowie seine Detektionsleistung und Geschwindigkeit untersucht.

Im Gegensatz zu bisherigen Ansätzen [Tu05],[BFSO84] bleiben die Stärken des Kaskadenklassifikators erhalten. Durch einen automatischen Baumaufbau, basierend auf den Merkmalsvektoren der Hypothesen, wird eine hohe Detektionsleistung sichergestellt. Außerdem wird der Baumaufbau – im Gegensatz zu [Wen03],[Chr07],[WN07] – direkt bzgl. der Klassifikationsaufgabe optimiert und somit eine effizientere Klassifikation mit weniger Merkmalen ermöglicht. Um die Geschwindigkeit zu erhöhen, wird zudem sichergestellt, dass die Anzahl der Knoten, die eine Hypothese „besuchen“ muss, so gering wie möglich ist. Dies wird durch ein Boosting in zwei Phasen (*Decision-Boosting*, siehe Abschnitt 5.4) erreicht.

1.2. Zielsetzung der Arbeit

Ziel dieser Arbeit ist die Entwicklung eines Ansatzes zur Erweiterung der von Viola und Jones in [VJ01] vorgeschlagenen Kaskadenstruktur auf einen allgemeinen hierarchischen Klassifikator (siehe Kapitel 4). Dieser soll in der Lage sein, Fahrzeuge, d. h. sowohl PKW als auch LKW, unter beliebigen Aspektwinkeln zu erkennen. Dabei sollen die Vorteile der Kaskadenstruktur (siehe Abschnitt 3.4) – insbesondere die hohe Geschwindigkeit – erhalten bleiben.

Mit dem entwickelten Gesamtsystem soll es am Ende möglich sein, nicht nur Fahrzeuge in den Eingabebildern zu detektieren, sondern auch ihre Orientierung bzgl. der Kamera – d. h. den *Gierwinkel* (siehe Abschnitt 2.2) – mit hoher Genauigkeit und Geschwindigkeit zu schätzen. Die Gesamtzeit, die für die komplette Bearbeitung eines Einzelbildes zur Verfügung steht, ist hierbei auf 100 Millisekunden beschränkt.

(a) (b) (c)

Abbildung 1.2.: Eingabebild, Einbauposition und Kamera des Systems. (a) Beispiel für ein 752 mal 480 Pixel großes Eingabebild. (b) Versuchsaufbau zum Aufnehmen der Sequenzen. Die Kamera wird zentral hinter der Frontscheibe angebracht. Der gelbe Pfeil zeigt die Einbauposition. (c) Kamera des Herstellers Continental. Quelle: (c): www.conti-online.com

Das Training der einzelnen Knoten des Baums soll analog zum Training der Stufen einer Kaskade mit *AdaBoost* (siehe 3.1) erfolgen. Als Merkmale kommen neben den von Viola und Jones eingesetzten *Haarwavelets* (siehe Abschnitt 3.2.1) auch die von Levi und Weiss vorgeschlagenen [LW04] *Edge Orientation Histograms* (siehe Abschnitt 3.2.2) zum Einsatz.

Als Eingabedaten für das realisierte System werden Mono-Bildsequenzen mit 8 Bit Grauwertbildern der Größe 752 mal 480 Pixel (siehe Abb. 1.2(a) für ein Beispiel) verwendet. Die Bilder stammen alle von einer Kamera des Herstellers Continental mit der Modellbezeichnung „MFC“ (Multi-Function Camera). Aufgenommen wird aus einem LKW heraus, die Einbauposition der

Kamera ist hierbei zentral hinter der Frontscheibe. Abb. 1.2(b) zeigt die Einbauposition und den aufnehmenden LKW, Abb. 1.2(c) zeigt die Kamera.

Die Implementierung des Ansatzes soll objektorientiert in C++ [Str00] erfolgen und sich in die bestehende Software-Landschaft integrieren sowie diese ggf. erweitern. Für das Training ist der Einsatz der Skriptsprache Python [Lut01] vorgesehen.

1.3. Grundbegriffe

Um Missverständnisse zu vermeiden, werden in diesem Abschnitt einige Grundbegriffe dieser Arbeit definiert. Soweit nicht anders angegeben, gelten diese Definitionen für die gesamte Arbeit.

Mit *Fahrzeug* sind sowohl PKW als auch LKW gemeint. Falls eine Unterscheidung zwischen PKW und LKW bedeutsam ist, so wird explizit auf den korrekten Typ des Fahrzeugs eingegangen. Die Rückfront eines Fahrzeugs meint sowohl die Hinter- als auch die Vorderseite. D. h. in dieser Arbeit wird keine Unterscheidung zwischen „Vorne“ und „Hinten“ detektierter Fahrzeuge vorgenommen.

Eine *Hypothese* ist ein rechteckiger Teilbereich des Eingabebildes. Der Raum aller möglichen Hypothesen wird $\mathcal{X}$ oder *Eingaberaum* genannt. Hypothesen werden durch die *Merkmalsextraktion* (engl. *feature extraction*) in den *Merkmalsraum* $\mathcal{M}$ abgebildet.

Ein *Merkmal* ist die Reduktion einer Hypothese auf einen skalaren Wert. Wie die Hypothese auf einen skalaren Wert reduziert wird (die „Berechnungsvorschrift“), ist abhängig vom konkreten Merkmal (siehe auch Abschnitt 3.2). Der Vektor, der sich aus Kombinationen aller Merkmalswerte ergibt, heißt *Merkmalsvektor* der Hypothese.

Als *Klassifikation* wird eine Abbildung

$$K : \mathcal{M} \to \mathcal{C},\ K(\underline{x}) = y$$

von $\mathcal{M}$[1] auf ein Menge von *Klassen* $\mathcal{C}$ bezeichnet. Die Abbildung K wird *Klassifikator* oder *Modell*[2] genannt. $y \in \mathcal{C}$ ist die *Klasse*, die durch den Klassifikator für den Merkmalsvektor $\underline{x} \in \mathcal{M}$ vorgeschlagen wird.

Soweit nicht anders angegeben, wird in dieser Arbeit immer von *binärer* Klassifikation ausgegangen. In diesem Fall ist $\mathcal{C} = \{-1, 1\}$. Die Klassen „-1“ und „1“ heißen *Hintergrund* bzw. *Vordergrund* oder *Objekt*. Die Aussage $K(\underline{x}) = 1$ wird als *Detektion* bezeichnet, die Aussage $K(\underline{x}) = -1$ heißt *Zurückweisung* der zu $\underline{x}$ gehörenden Hypothese.

Als *Training* wird das Bilden eines Modells anhand von *Beispielen* bezeichnet. Beispiele (synonym: *Samples*) sind Hypothesen, für die die korrekte Klasse bekannt ist. Beispiele der Klasse „1“ heißen *Positivbeispiele*, alle anderen *Negativbeispiele*. Die Unterscheidung zwischen den Positiv- und den Negativbeispielen wird *Klassifikationsaufgabe* oder *Klassifikationsproblem* genannt. Die Klassifikationsaufgabe ist also eine *Trennung* zweier Mengen (Positiv- gegen Negativbeispiele). Durch das Training soll also ein Modell konstruiert werden, welches die gegebene Klassifikationsaufgabe löst.

[1] Für manche Klassifikatoren gilt $\mathcal{M} = \mathcal{X}$, d. h. sie kombinieren Merkmalsextraktion und Klassifikation in einem Schritt und arbeiten direkt auf den Hypothesen.

[2] Die Literatur unterscheidet zwischen *diskrimininativen* und *generativen* Modellen (vgl. z. B. [Bis06, S. 43], [BK08, S. 461 f.] und [EG09]). In dieser Arbeit sind alle Modelle ausschließlich diskriminativer Art.

In der Phase der *Anwendung* werden die Hypothesen gemäß des im Training gewonnenen Modells klassifiziert. Die tatsächlichen Klassen sind in dieser Phase nicht bekannt.

Eine *Sequenz* ist eine Abfolge von hintereinander aufgenommenen Bildern, wobei zwischen den einzelnen Bildern 40 Millisekunden liegen. Das entspricht 25 Bildern in der Sekunde. Eine Sequenz ist bei den in dieser Arbeit verwendeten Daten immer genau 5 Sekunden – d. h. 125 Bilder – lang.

Die *Performance* eines Klassifikators ist seine Leistung (siehe auch Abschnitt 6.2) bzgl. der Klassifikationsaufgabe (Falschalarmrate etc.). Sie ist nicht zu verwechseln mit der *Geschwindigkeit*, die angibt wie schnell Eingaben verarbeitet werden können.

x und y (kleingeschrieben) sind *Bildkoordinaten* (in Pixeln). Hierbei ist x die Spalte, y die Zeile im Bild. Der Ursprung ist die linke, obere Ecke. X, Y und Z (großgeschrieben) sind Weltkoordinaten (in Metern) im Weltkoordinatensystem (siehe Abschnitt 2.2).

1.4. Aufbau des Systems

Ziel dieser Arbeit ist die Entwicklung eines Systems, welches, basierend auf einer hierarchischen Erweiterung der Viola-Jones-Kaskade, die Orientierung von Fahrzeugen bzgl. der Kamera schätzt (vgl. Abschnitt 1.2).

Zu diesem Zweck werden sowohl das Gesamtfahrzeug als auch die Fahrzeugrückfront mit dem entwickelten hierarchischen Klassifikator *getrennt* in den Eingabebildern detektiert. Im Anschluss daran werden die gefundenen Fahrzeuge mit den gefundenen Rückfronten assoziiert, d. h. jedem Fahrzeug wird die zugehörige Rückfront zugewiesen. Aus der so gewonnenen Information (Position der Rückfront zur Position des zugehörigen Fahrzeugs) lässt sich die gesuchte Orientierung bestimmen (siehe Abb. 1.3). Abb. 1.4 zeigt das entwickelte System im Überblick.

Abbildung 1.3.: Aus getrennter Detektion von Rückfront und Fahrzeug lässt sich der Aspektwinkel schätzen. Sind Rückfront (gelb) und Gesamtfahrzeug (rot) detektiert, so lässt sich aus der Beziehung zwischen den Detektionen der Aspektwinkel schätzen.

1.5. Ausgangsbasis und Hilfsmittel

Ausgangspunkt dieser Arbeit ist der von Viola und Jones vorgestellte Kaskadenklassifikator [VJ01]. In der Arbeit von Kallenbach [Kal05] (eine Kurzfassung findet sich in [KSPL06]) werden

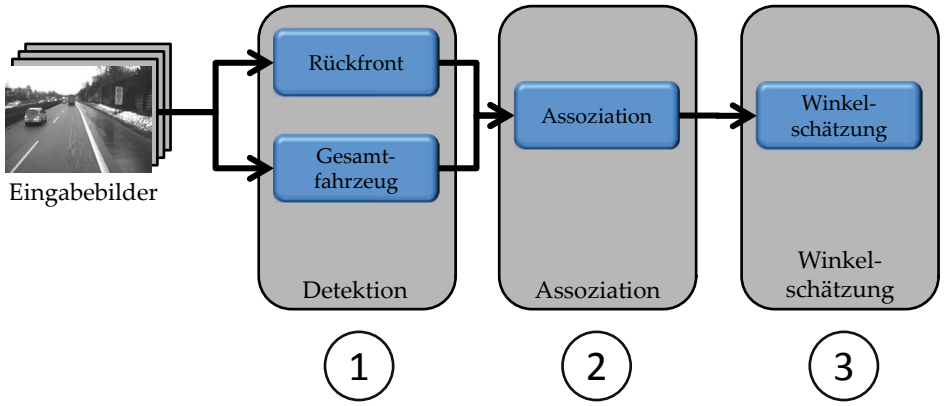


Abbildung 1.4.: Überblick über das entwickelte Gesamtsystem. (1) Fahrzeuge und Fahrzeugrückfronten werden *getrennt* in den Eingabebildern detektiert. (2) Im Anschluss daran werden den Fahrzeugen die zugehörigen Rückfronten zugeordnet. (3) Die gewonnene Beziehung von Fahrzeug und zugehöriger Rückfront lässt sich nutzen, um eine Schätzung des Aspektwinkels vorzunehmen.

Ansätze, die Viola-Jones-Kaskade um Multiklassenfähigkeit zu erweitern, untersucht. Die hierbei entwickelten Konzepte bilden die Grundlage dieser Arbeit. Wender [Wen03] (eine Kurzfassung findet sich in [WL04]) liefert die Motivation für die Erweiterung der Kaskade auf einen hierarchischen Klassifikator. Wender untersucht theoretisch und experimentell wie sich eine Teilung der Trainingsbeispiele auf die Konvergenz von AdaBoost auswirkt. Die hierbei gewonnenen Erkenntnisse sind die Motivation für das vorgestellte Konzept.

Das weiterentwickelte Konzept basiert in Teilen auf dem von Wu und Nevatia entwickelten *Cluster Boosted Tree* [WN07] (ausführlicher beschrieben in [Wu08]) sowie dessen Weiterentwicklung von Yang et al. (*Voting Cluster Boosted Tree*) [YHN09].

Für die Umsetzung des Klassifikators stand eine Implementierung des RealAdaBoost-Algorithmus [SS99] mit verbesserten Weaklearnern (siehe Abschnitt 3.4.2) in C++ zur Verfügung. Ferner waren Implementierungen der verwendeten Merkmale (siehe Abschnitt 3.2) sowie der Hypothesengenerierung (siehe Abschnitt 3.5) bereits vorhanden. Zudem war ein Beispiel für das Training einer Kaskade auf Basis der vorhandenen Einzelteile verfügbar. Dieses bildete den Ausgangspunkt für die entwickelte Software.

Um den entwickelten Klassifikator zu trainieren stehen 246 Sequenzen mit insgesamt ca. 35 000 Bildern (ca. 58 Gigabyte komprimiert im TIFF-Format) zur Verfügung. Die Bilder zeigen typische Straßenszenen bei Tag und bei Dämmerung. In jedem Bild sind die sichtbaren Fahrzeuge und (soweit zu sehen) die zugehörige Rückfront bzw. Front durch umschließende Rechtecke gelabelt. Die Label unterscheiden zwischen LKW und PKW. Zusätzliche Daten (Radar für Distanzinformationen etc.) sind nicht verfügbar.

1.6. Gliederung der Arbeit

Die restliche Arbeit ist wie folgt aufgebaut.

In Kapitel 2 werden die Schritte 2 und 3 des Gesamtsystems (siehe Abb. 1.4) beschrieben. Hierfür wird zunächst anhand eines Fahrzeugmodells untersucht, wie genau eine Aspektwin-

kelberechnung maximal sein kann, d. h. wie groß der inhärente und damit unvermeidbare Fehler bei einer Berechnung des Aspektwinkels ist. Darauf aufbauend wird die vereinfachte Winkelschätzung mittels Kategorisierung in Winkelkategorien beschrieben. Ferner wird auf das Verfahren zur Assoziation von Gesamtfahrzeug und Rückfront eingegangen.

In Kapitel 3 werden die Grundlagen, welche als Ausgangspunkt für den entwickelten Klassifikator dienen, erläutert. Vertiefend wird hierbei auf den Lernalgorithmus AdaBoost sowie die Viola-Jones-Kaskade eingegangen, da diese von zentraler Bedeutung für diese Arbeit sind.

Kapitel 4 gibt eine Einleitung in Klassifikationsbäume und stellt generelle Konzepte für die Erweiterung der Viola-Jones-Kaskade auf einen Klassifikationsbaum vor. Die unterschiedlichen Möglichkeiten werden diskutiert und ihre Vor- und Nachteile erläutert. Abschließend werden Training und Anwendung hierarchischer Klassifikatoren erläutert.

Kapitel 5 beschreibt den entwickelten Klassifikationsbaum. Es wird auf die unterschiedlichen Teilaspekte im Detail eingegangen sowie das Training und die Anwendung erläutert. Die Leistungsfähigkeit des entwickelten Konzeptes wird in Kapitel 6 experimentell bestätigt.

Kapitel 7 gibt eine abschließende Zusammenfassung und dient als Ausblick für weitere Entwicklungen.

KAPITEL 2

Aspektwinkelschätzung

In diesem Kapitel wird untersucht, wie die Aspektwinkelschätzung im realisierten Gesamtsystem (vgl. Abschnitt 1.4) durchgeführt werden kann und was hierbei theoretische Untergrenzen für die Genauigkeit der Winkelschätzung sind.

Hierfür wird das Koordinatensystem für die Winkelberechnung eingeführt (Abschnitt 2.2), die Assoziation von zusammengehörigen Detektionen beschrieben (Abschnitt 2.3) sowie ein vereinfachendes Fahrzeugmodell aufgestellt (Abschnitt 2.4). Dieses erlaubt die Berechnung des Aspektwinkels auf verschiedene Arten (Abschnitte 2.5 und 2.6). Abschließend wird der Fehler bei der (modellbasierten) Berechnung untersucht (Abschnitt 2.7) sowie das Vorgehen bei der Einordnung von Detektionen beschrieben (Abschnitt 2.8).

2.1. Einleitung

Ziel dieser Arbeit ist u. a. die Schätzung des Aspektwinkels detektierter Fahrzeuge. Zu diesem Zweck werden Fahrzeugrückfront und Gesamtfahrzeug getrennt detektiert und anschließend assoziiert. Die Position der Rückfront innerhalb der Detektion des Gesamtfahrzeugs lässt Rückschlüsse auf den Aspektwinkel des Fahrzeugs zu (vgl. Abschnitt 1.4).

In diesem Kapitel wird untersucht, wie genau eine modellbasierte Berechnung des Aspektwinkels auf dieser Basis sein kann. Hierfür wird ein vereinfachtes Fahrzeugmodell aufgestellt, mit dem der Aspektwinkel berechnet werden kann. Hierbei können zwei Ansätze unterschieden werden:

- Der gesuchte Winkel kann aus Fahrzeuglänge oder -breite berechnet werden (Abschnitt 2.5).
- Das Verhältnis von Fahrzeuglänge zu -breite kann als Ausgangspunkt für die Berechnung des Winkels dienen. In diesem Fall ist die Kenntnis von Fahrzeuglänge bzw. -breite nicht nötig (Abschnitt 2.6).

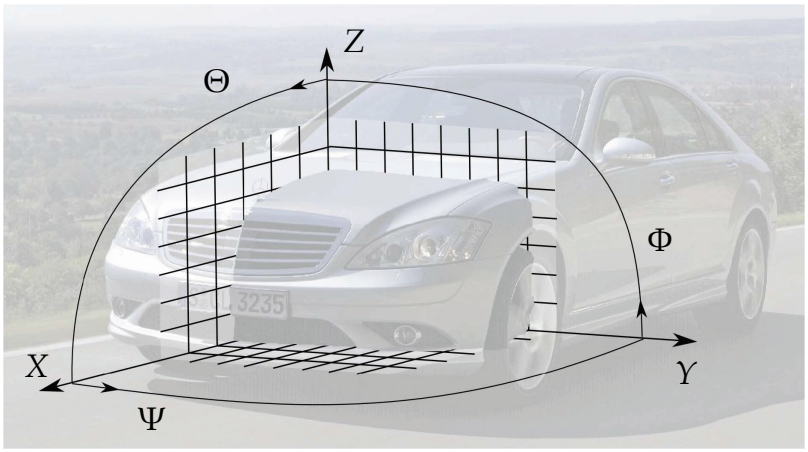


Abbildung 2.1.: Koordinatensystem nach DIN 70 000. Der Ursprung des Koordinatensystems befindet sich mittig auf der Vorderachse. Die X, Y und Z-Achsen sind in Fahrtrichtung, nach links bzw. nach oben ausgerichtet. Zusätzlich sind die Winkel für die rotatorischen Bewegungen „Rollen“ (Φ), „Nicken“ (Θ) und „Gieren“ (Ψ) definiert. Für diese Arbeit ist der Gierwinkel Ψ detektierter Fahrzeuge maßgeblich.
Quelle: [Nem07, S. 6]

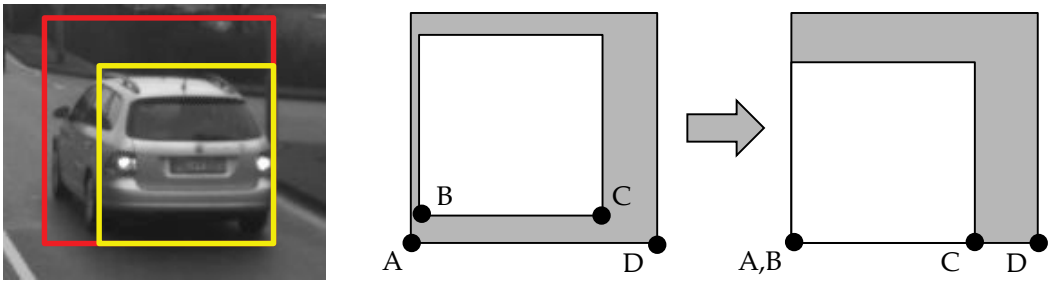
Die modellbasierten Berechnung des Aspektwinkels unterliegt einem unvermeidbaren *Modellfehler*. Dieser stellt die Untergrenze des Fehlers bei der Berechnung des Aspektwinkels dar. Der Modellfehler wird anhand dreier Modellprobleme untersucht (Abschnitt 2.7).

2.2. Koordinatensystem

Um den Aspektwinkel detektierter Fahrzeuge schätzen zu können, ist das Weltkoordinatensystem, in dem sie sich aus Sicht des detektierenden Fahrzeugs befinden, festzulegen. Die geschätzte Lage detektierter Fahrzeuge bezieht sich auf dieses Weltkoordinatensystem.

In dieser Arbeit wird das im automobilen Umfeld übliche aufbaufeste Koordinatensystem nach DIN 70 000 verwendet. Der Ursprung des Koordinatensystems ist mittig zwischen den vorderen Rädern, auf Höhe der Radachse. Von diesem Ursprung ausgehend ist die X-Achse in Fahrtrichtung, die Y-Achse nach links und die Z-Achse nach oben ausgerichtet (rechtshändisches Koordinatensystem). Die Winkel zwischen den Koordinatenachsen sind jeweils 90°.

Zusätzlich zum Koordinatensystem legt DIN 70 000 noch drei rotatorische Bewegungen „Rollen“, „Nicken“ und „Gieren“ fest. Rollen ist als Drehung um die X-Achse, Nicken als Drehung um die Y-Achse, Gieren als Drehung um die Z-Achse definiert. Die zugehörigen Winkel heißen Φ (Rollwinkel), Θ (Nickwinkel) und Ψ (Gierwinkel). Die Drehrichtung ist der mathematisch positive Sinn (gegen den Uhrzeigersinn). Abb. 2.1 zeigt das Koordinatensystem sowie die definierten Drehbewegungen und -richtungen.



(a) (b)

Abbildung 2.2.: Assoziation von Rückfronten und Gesamtfahrzeugen. (a) Perfekte Detektionen. (b) Für das Gesamtsystem müssen Rückfronten (weiß) zu Gesamtfahrzeugen (grau) assoziiert werden. Hierbei sind die unteren Ecken der Detektionen (A, B, C und D) maßgeblich. Entsprechen die Detektionen exakt den realen Ausmaßen von Gesamtfahrzeug bzw. Rückfront, sind entweder die Ecken A und B oder die Ecken C und D deckungsgleich (rechte Seite).

Durch Angabe der Koordinaten $(X, Y, Z)^T$ und der Drehwinkel $(\Phi, \Theta, \Psi)^T$ ist ein Objekt eindeutig im dreidimensionalen Raum lokalisier- und ausrichtbar (*6 Degrees of Freedom*, 6DoF) [Nem07, S. 3 ff.].

In diesem Kontext lässt sich die gesuchte Größe „Aspektwinkel“ detektierter Fahrzeuge wie folgt fassen: Der Aspektwinkel ist der Gierwinkel Ψ detektierter Fahrzeuge, bezogen auf das am detektierenden Fahrzeug angebrachte Koordinatensystem.

2.3. Assoziation von Fahrzeug und Fahrzeugrückfront

Detektionen von Gesamtfahrzeugen und Fahrzeugrückfronten müssen einander zugeordnet werden, nachdem diese getrennt voneinander gefunden wurden (vgl. Abschnitt 2.1). PKW werden PKW-Rückfronten zugeordnet, LKW werden mit LKW-Rückfronten assoziiert, d. h. die Assoziation findet nicht zwischen verschiedenen Fahrzeugtypen statt.

Für die Zuordnung sind die unteren Ecken der Detektionen maßgeblich (vgl. Abschnitt 2.4). Abb. 2.2(b) stellt diese Ecken A, B, C und D dar. Die Detektion von Rückfront und Gesamtfahrzeug sind weiß bzw. grau dargestellt. Entsprechen die Detektionen exakt den realen Ausmaßen von Gesamtfahrzeug bzw. Rückfront, sind entweder die Ecken A und B oder die Ecken C und D deckungsgleich (Abb. 2.2(b), rechte Seite).

Um auch nicht perfekt zueinander passende Detektionen zu assoziieren, wird die L_1-Distanz (*Manhattan Distance*) zwischen den Punkten A und B bzw. C und D berechnet. Der kleinere der berechneten Abstände wird im Anschluss durch die Breite der Gesamtfahrzeugdetektion dividiert, um den Abstand in Bezug zur absoluten Größe der Detektionen zu setzen (Skalierungsinvarianz). Ist die so berechnete Größe kleiner als eine Schwelle Γ, so werden die Detektionen einander zugeordnet. Bei mehreren möglichen Paaren wird ein *best-fit*-Ansatz gewählt, d. h. die am besten zueinander passenden Paare werden assoziiert.

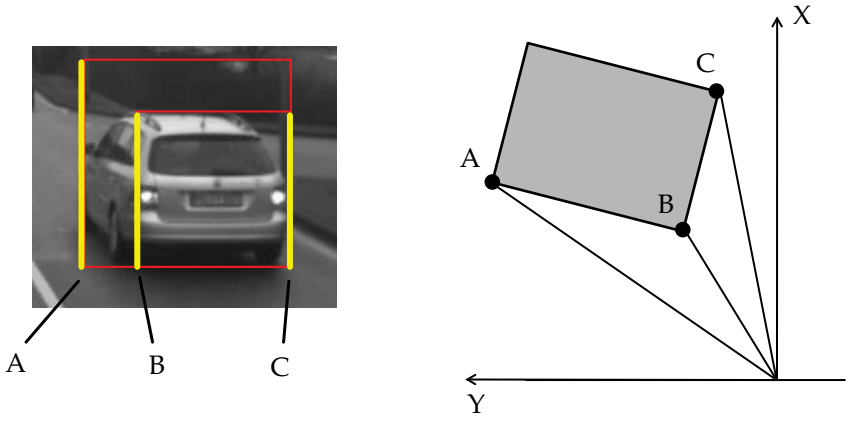


Abbildung 2.3.: Geometrische Ausgangssituation bei der Schätzung des Aspektwinkels. Bei der Schätzung des Aspektwinkels sind drei Kanten (gelb) maßgeblich. Diese entsprechen aus der Vogelperspektive betrachtet drei Punkten A, B und C.

2.4. Geometrische Ausgangssituation und Fahrzeugmodell

Da lediglich der Gierwinkel detektierter Fahrzeuge berechnet werden soll, werden der Roll- und Nickwinkel nicht weiter untersucht. Unter Annahme eines ungekrümmten Erdbodens (engl. *Ground Plane*, siehe 3.5) lässt sich die Situation aus der Vogelperspektive betrachten. Hierfür ist Voraussetzung, dass waagrechte Linien im Bild auch tatsächlich waagrechten Linien in der Realität entsprechen, d. h. die Kamera annähernd waagrecht verbaut ist. Das war beim Versuchsaufbau dieser Arbeit der Fall (vgl. 1.2).

Für die Schätzung des Gierwinkels sind drei Fahrzeugkanten von Bedeutung. Diese sind in Abb. 2.3 gelb hervorgehoben. Wird die Situation aus der Vogelperspektive betrachtet, so entsprechen diese drei Punkten A, B und C. Ohne Einschränkung der Allgemeinheit ist hierbei der Punkt A immer links, der Punkt B mittig und der Punkt C rechts. Abb. 2.3 rechts zeigt die Situation im Weltkoordinatensystem aus der Vogelperspektive. Die Z-Achse kann in diesem Fall vernachlässigt werden. Die Kamera ist im Ursprung angebracht und filmt in Fahrtrichtung, d. h. entlang der X-Achse. Die Verbindungslinien zwischen den Punkten A, B und C und dem Ursprung in Abb. 2.3 rechts sind die Sichtlinien der Kamera.

Es sind nicht immer drei Punkte sichtbar. Steht das Fahrzeug in einem bestimmten Winkel zur Kamera, sind lediglich zwei Punkte zu sehen. Abb. 2.4 zeigt diesen Fall. Analog zum ersten Fall seien dies die Punkte A und B. Ohne Einschränkung der Allgemeinheit ist A der linke und B der rechte der Punkte.

Durch die perspektivische Projektion der Kamera gehen Tiefeninformationen verloren, d. h. die X- und Y-Koordinaten der Punkte sind nicht genau bestimmbar. Alle Punkte werden auf eine Bildebene in gleicher Tiefe abgebildet. Im hier verwendeten Modell sei diese durch den Punkt B gehend, d. h. das Fahrzeug hat im Bild die in Abb. 2.5 rot eingezeichneten horizontalen Ausmaße. Die Punkte A und C werden auf die Bildpunkte A' bzw. C' abgebildet. Die Ortsvektoren von A', B und C' seien $\underline{a}$, $\underline{b}$ und $\underline{c}$. Die Distanz der Bildebene (= X-Koordinate des Punkts B) sei mit d bezeichnet. Der gesuchte Winkel ist α. β ergibt sich als $\beta = 90° - \alpha$.

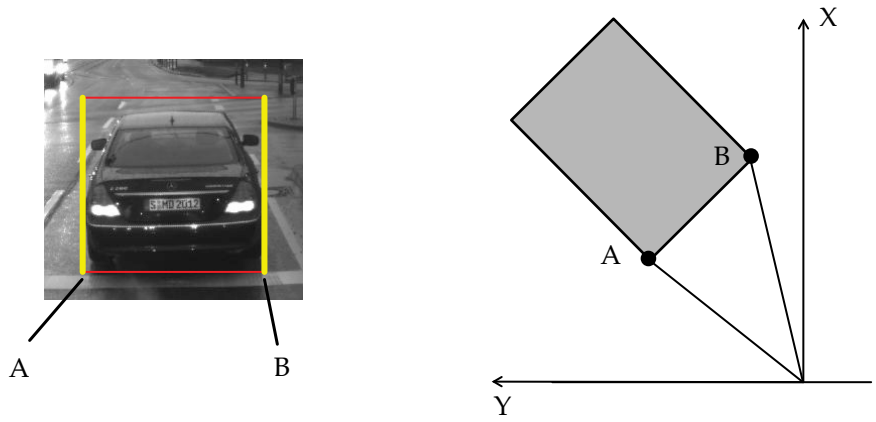


Abbildung 2.4.: Geometrische Ausgangssituation bei der Schätzung des Aspektwinkels. Steht das Fahrzeug in einem bestimmten Winkel, sind nur zwei der drei maßgebenden Kanten sichtbar.

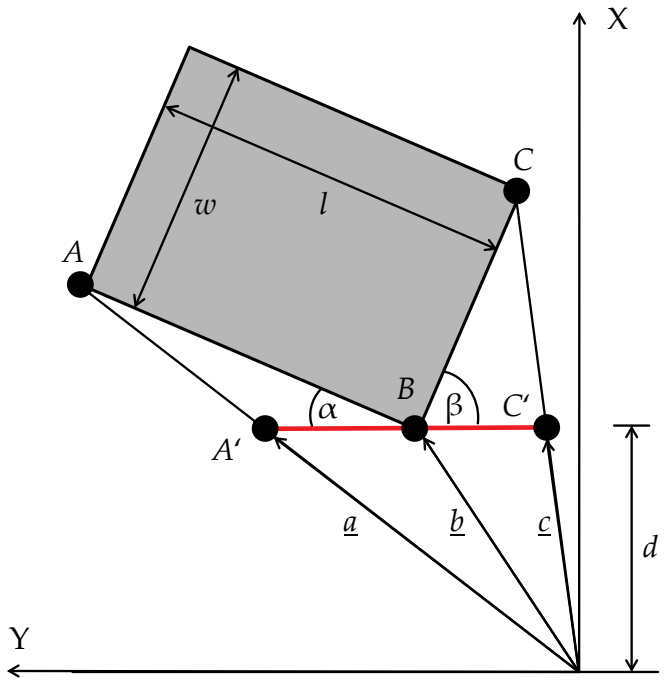


Abbildung 2.5.: Maßgebende Variablen bei der Berechnung des Aspektwinkels. Durch die perspektivische Projektion auf eine Bildebene gehen Tiefeninformationen verloren. Die Bildebene sei durch den Punkt B gehend. Die Punkte A und C werden auf die Bildpunkte A′ bzw. C′ abgebildet. Die rote Linie zeigt die sichtbaren Ausmaße des Fahrzeugs. Der gesuchte Winkel ist dann durch α gegeben.

Die Breite w und die Länge l des Fahrzeugs können nur geschätzt werden. In der Realität weichen die angenommenen w und l von den tatsächlichen Werten ab. D. h. bei der Berechnung des Winkels ergibt sich ein inhärenter *Modellfehler*, da keine Unterscheidung von Fahrzeugmodellen (Smart versus S-Klasse etc.) vorgenommen wird. Die Auswirkungen des Modellfehlers werden anhand dreier Modellprobleme untersucht (siehe Abschnitt 2.7).

2.5. Berechnung des Aspektwinkels aus Fahrzeuglänge und -breite

Wird die Breite w oder die Länge l eines Fahrzeugs als bekannt angenommen, lässt sich der gesuchte Winkel α (siehe Abschnitt 2.4) berechnen, wenn $\underline{b}$ und $\underline{a}$ bekannt sind. Ist $\underline{c}$ statt $\underline{a}$ bekannt, ist β berechenbar, womit α bestimmt werden kann. Im Folgenden wird auf die Berechnung von α eingegangen. Das Bestimmen von β verläuft analog.

Sei $\underline{A}$ der Ortsvektor des Punkts A, $\|.\|$ sei die L_2-Norm (*Euklidischer Abstand*), x_i sei die i-te Komponente eines Vektors $\underline{x}$.

Da bekannt ist, dass $\|\underline{A} - \underline{b}\| = l$ und es einen Faktor k gibt, für den $\underline{A} = k\underline{a}$ gilt, lässt sich die quadratische Gleichung

$$\|k\underline{a} - \underline{b}\| = \sqrt{(ka_1 - b_1)^2 + (ka_2 - b_2)^2} = l \tag{2.1}$$

aufstellen. Von den Lösungen

$$k_{1,2} = \frac{(a_1b_1 + a_2b_2) \pm \sqrt{2a_1a_2b_1b_2 - a_1^2b_2^2 + a_1^2l^2 - a_2^2b_1^2 + a_2^2l^2}}{a_1^2 + a_2^2}$$

ist k_1 das gesuchte k. $k_2 \leq k_1$ ist eine Lösung der Gleichung, bei der der Vektor $\underline{a}$ verkürzt statt verlängert wird (siehe Abb. 2.6(a)). Diese Lösung kann ignoriert werden. Hat Gleichung (2.1) keine Lösung, weicht das angenommene l so stark von der tatsächlichen Länge des Fahrzeugs ab, dass sich die Annahme nicht mehr in Einklang mit dem Bild bringen lässt (siehe Abb. 2.6(b))

Mit dem Faktor k_1 lässt sich α als Winkel zwischen den Vektoren $(k_1\underline{a} - \underline{b})$ und $(\underline{a} - \underline{b})$ berechnen. D. h.

$$\alpha = \arccos \frac{(k_1\underline{a} - \underline{b})(\underline{a} - \underline{b})}{\|k_1\underline{a} - \underline{b}\| \ \|\underline{a} - \underline{b}\|}.$$

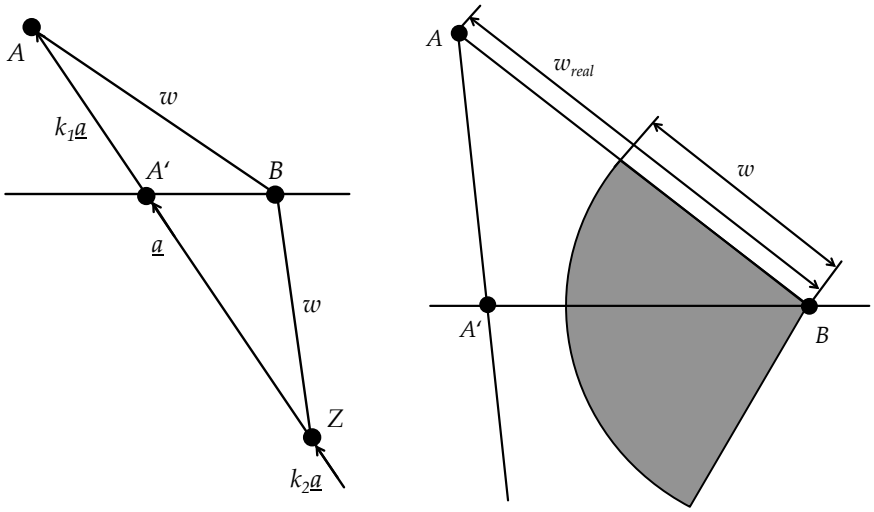
2.6. Berechnung des Aspektwinkels aus dem Verhältnis von Fahrzeuglänge und -breite

Statt w oder l direkt als bekannt anzunehmen, lässt sich der Aspektwinkel auch aus dem Verhältnis von Fahrzeuglänge zu Fahrzeugbreite berechnen. D. h. indem q mit

$$q = \frac{l}{w}$$

festgelegt wird, ist α berechenbar, ohne l und w zu kennen.

Hierfür müssen – im Gegensatz zum ersten Ansatz (Abschnitt 2.5) – alle drei maßgebenden Punkte A, B und C sichtbar sein. Sind lediglich zwei Punkte zu sehen, kann α nur in einem Intervall angegeben werden (siehe Abschnitt 2.6.2).



(a) (b)

Abbildung 2.6.: Probleme bei der Berechnung des Aspektwinkels aus der Fahrzeuglänge. (a) Bei der Berechnung des Aspektwinkels ergeben sich zwei mögliche Lösungen, von denen eine zu verwerfen ist. Sie ergibt den Faktor k_2, welcher zum Punkt Z führt, der „vor" B liegt. Der gesuchte Faktor k_1 ergibt den korrekten Punkt A. (b) Geometrisch entspricht die Lösung von Gleichung (2.1) den Schnittpunkten eines Kreises mit dem Radius w um den Punkt B mit der Sichtgeraden durch den Punkt A'. Hat der (gedachte) Kreis keinen Schnittpunkt, gibt es keine Lösung. Dies kann geschehen, wenn das angenommene w zu stark vom tatsächlichen w_{real} abweicht.

2.6.1. Berechnung des Aspektwinkels bei drei sichtbaren Punkten

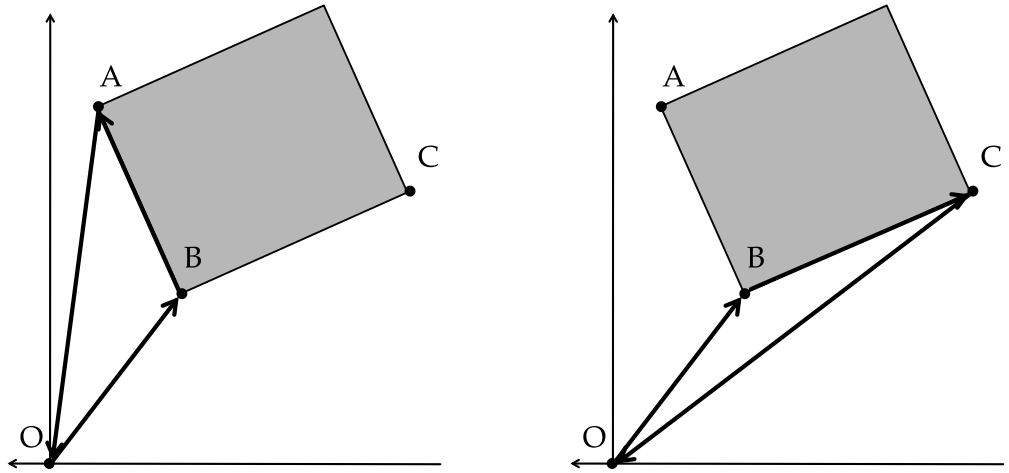
Ähnlich zum ersten Ansatz müssen für die Berechnung von α zuerst die Faktoren $k_{1,2}$ gefunden werden, womit die Koordinaten der Punkte $\underline{A} = k_1\underline{a}$ und $\underline{C} = k_2\underline{c}$ berechnet werden können.

Die Dreiecke OBA und OBC bilden jeweils einen geschlossenen Vektorzug (siehe Abb. 2.7). Damit lässt sich das System

$$\begin{aligned} \underline{b} + l\begin{pmatrix}\sin\alpha\\ \cos\alpha\end{pmatrix} - k_1\underline{a} &= \underline{0} \\ \underline{b} + w\begin{pmatrix}\sin\beta\\ -\cos\beta\end{pmatrix} - k_2\underline{c} = \underline{b} + w\begin{pmatrix}\sin(90^\circ-\alpha)\\ -\cos(90^\circ-\alpha)\end{pmatrix} - k_2\underline{c} &= \underline{0} \end{aligned}$$

aufstellen. $\underline{0}$ ist hierbei der zweidimensionale Nullvektor $(0,0)^T$. Durch Einsetzen der Gleichungen

$$\begin{aligned} l &= qw \\ \sin(90^\circ-\alpha) &= \cos\alpha \\ -\cos(90^\circ-\alpha) &= -\sin\alpha \end{aligned}$$



(a) (b)

Abbildung 2.7.: Die Dreiecke OBA und OBC bilden geschlossene Vektorzüge. (a) Vektorzug des Dreiecks OBA. (b) Vektorzug des Dreiecks OBC.

lässt sich das System auf die Form

$$k_1 a_1 = b_1 + qw \sin\alpha \tag{2.2}$$
$$k_1 a_2 = b_2 + qw \cos\alpha \tag{2.3}$$
$$k_2 c_1 = b_1 + w \cos\alpha \tag{2.4}$$
$$k_2 c_2 = b_2 - w \sin\alpha \tag{2.5}$$

bringen. Elimination mittels $(2.2) + q(2.5)$ und $(2.3) - q(2.4)$ ergibt das lineare System

$$\begin{aligned} k_1 a_2 - k_2 q c_1 &= -q b_1 + b_2 \\ k_1 a_1 + k_2 q c_2 &= b_1 + q b_2 \end{aligned}$$

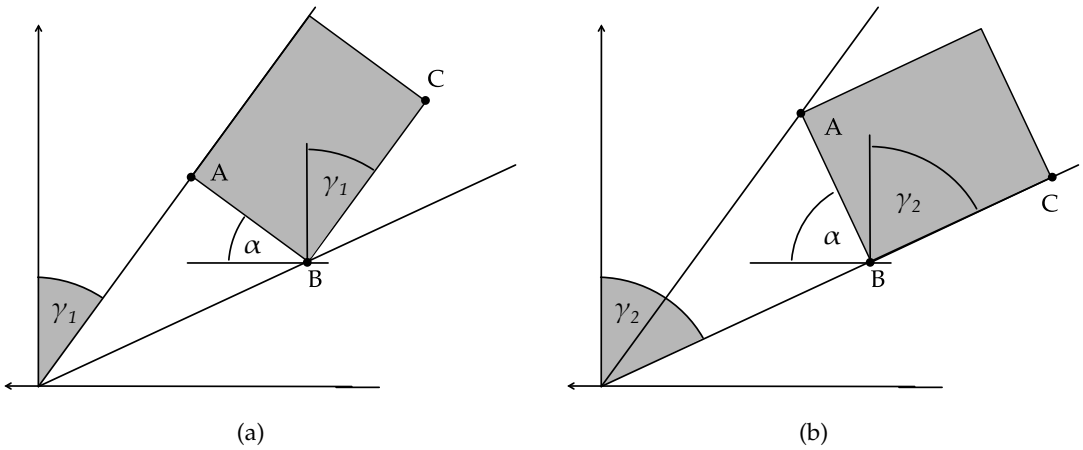
mit der Lösung

$$\begin{aligned} k_1 &= -\frac{q c_2 b_1 - c_2 b_2 - b_1 c_1 - q b_2 c_1}{a_2 c_2 + a_1 c_1} \\ k_2 &= \frac{a_2 b_1 + q a_2 b_2 + q a_1 b_1 - a_1 b_2}{q(a_2 c_2 + a_1 c_1)}. \end{aligned}$$

Mit den so berechneten Faktoren $k_{1,2}$ können, analog zum ersten Ansatz, die gesuchten Winkel α und β berechnet werden.

Diese Berechnung von α setzt keine Kenntnis der Entfernung der Fahrzeugdistanz d voraus. D. h. für die Berechnung kann $\|\underline{b}\| = 1$ angenommen werden[1]. Dies lässt sich damit erklären, dass das im Bild sichtbare Seitenverhältnis als Folge des Strahlensatzes invariant gegenüber Streckungen und Stauchungen ist. Aus der sichtbaren Länge nur einer Seite lässt sich nicht auf den Winkel α schließen, ohne Distanzinformationen zu Hilfe zu nehmen. Daher wird für die Berechnung mittels Ansatz 1 die Fahrzeugdistanz d benötigt.

[1] Die Vektoren $\underline{a}$ und $\underline{c}$ sind in diesem Fall entsprechend zu skalieren.



(a) (b)

Abbildung 2.8.: Abschätzung des Aspektwinkels α bei nur zwei sichtbaren Punkten. Sind nur zwei Punkte sichtbar, lässt sich der Winkel α nur in einem Intervall angeben. Dieses ergibt sich geometrisch aus dem Spielraum, den ein Fahrzeug haben kann, ohne dass drei Punkte sichtbar werden. (a) Untere Schranke γ_1. (b) Obere Schranke γ_2.

2.6.2. Schätzung das Aspektwinkels bei zwei sichtbaren Punkten

Sind lediglich zwei der drei maßgebenden Punkte sichtbar, lässt sich α ohne Kenntnis von w und l nicht berechnen. Ist nur $q = \frac{l}{w}$ bekannt, kann lediglich ein Intervall für den Wert von α angegeben werden.

Bei zwei sichtbaren Punkten ist α durch die beiden Sichtwinkel γ_1 und γ_2 eingeschränkt, d. h.

$$\gamma_1 \leq \alpha \leq \gamma_2. \tag{2.6}$$

Die Grenzen der Abschätzung (2.6) ergeben sich dabei geometrisch aus dem Spielraum, den ein Fahrzeug haben kann, ohne dass einer der Punkte A oder C sichtbar wird. Abb. 2.8 zeigt die Grenzfälle $\alpha = \gamma_1$ und $\alpha = \gamma_2$, zwischen denen sich α bewegen kann.

2.7. Vergleich der Ansätze der modellbasierten Aspektwinkelschätzung

Die beiden vorgestellten Ansätze werden anhand dreier Modellprobleme verglichen, um einen Eindruck über die Genauigkeit und Stabilität der Winkelschätzung unter abweichenden Fahrzeuglängen/-breiten in der Realität zu vermitteln.

Die Modellprobleme sind

1. Ein Fahrzeug mit einem Aspektwinkel von 45° (siehe Abb. 2.9(a)),
2. ein Fahrzeug mit einem Aspektwinkel von 90° (siehe Abb. 2.9(b)) und
3. ein Fahrzeug mit einem Aspektwinkel von 10° (siehe Abb. 2.9(c)).

Die reale Fahrzeugbreite ist 3 Meter, die reale Länge 5 Meter; Die erwartete Fahrzeugbreite w ist 3.1 Meter, die erwartete Länge l 5.1 Meter. Das entspricht einer Modellabweichung von ca. 3 % in der Breite und 2 % in der Länge. Abb. 2.9 zeigt die drei Modellprobleme.

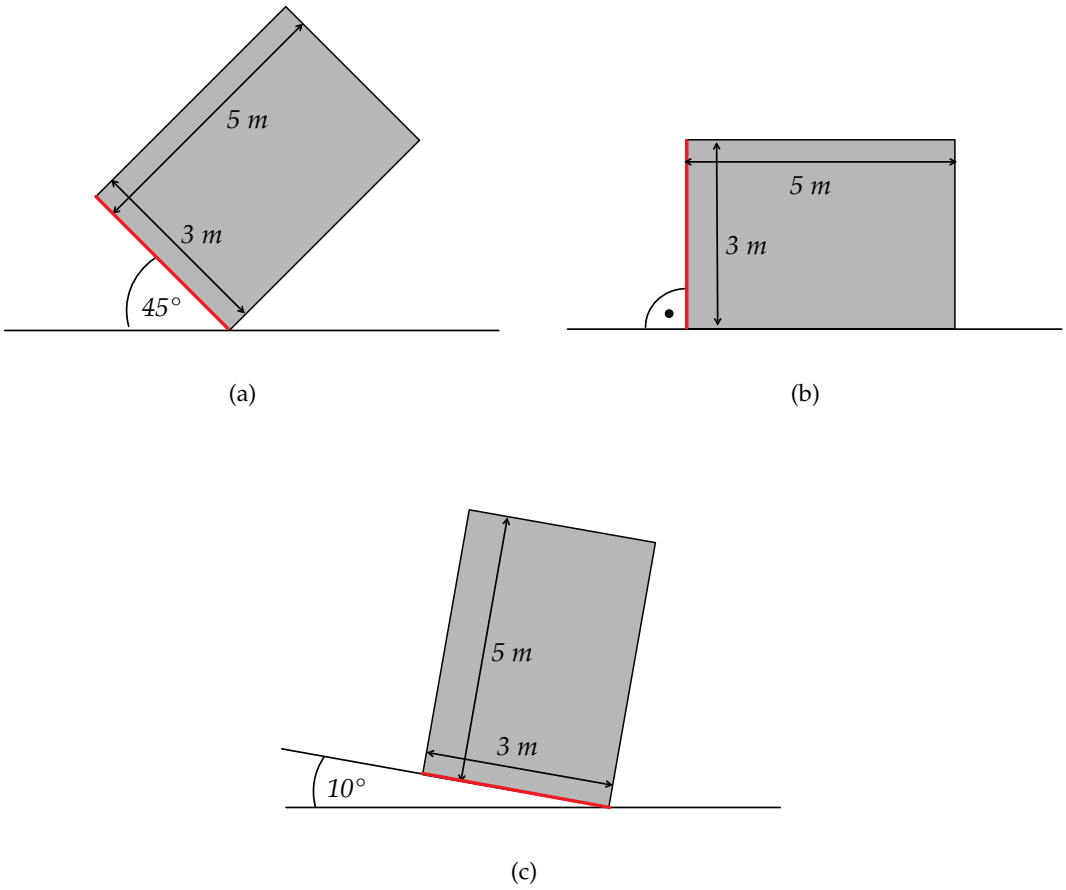


Abbildung 2.9.: Modellprobleme. Um einen Eindruck von der Genauigkeit einer modellbasierten Aspektwinkelschätzung zu vermitteln, werden die vorgestellten Ansätze anhand dreier Modellprobleme untersucht. Diese sind (a) ein Fahrzeug mit Aspektwinkel 45°, (b) ein Fahrzeug mit Aspektwinkel 90° und (c) ein Fahrzeug mit Aspektwinkel 10°.

Für die Untersuchung werden die in Abb. 2.9 dargestellten Fahrzeuge an alle Positionen innerhalb des Sichtbereichs der Kamera verschoben. Der Kamera-Öffnungswinkel ist hierbei auf 90° festgelegt, die maximale Distanz beträgt 100 Meter. An jeder Position wird mit beiden Ansätzen der Winkel berechnet und anschließend die Abweichung vom tatsächlichen Winkel ausgewertet. Bei beiden Ansätzen wird der Winkel α berechnet. Mit dem ersten Ansatz wäre es prinzipiell möglich, den Gegenwinkel β (mit Hilfe der Punkte B und C) zu berechnen. Diese Option wird allerdings nicht weiter untersucht, um die Vergleichbarkeit der Ergebnisse sicherzustellen.

Abb. 2.10 zeigt den Fehler beim ersten Modellproblem. Die rechte Graphik zeigt den Fehler bei der Berechnung mit dem ersten Ansatz. Die Abweichung beträgt, je nach Position des

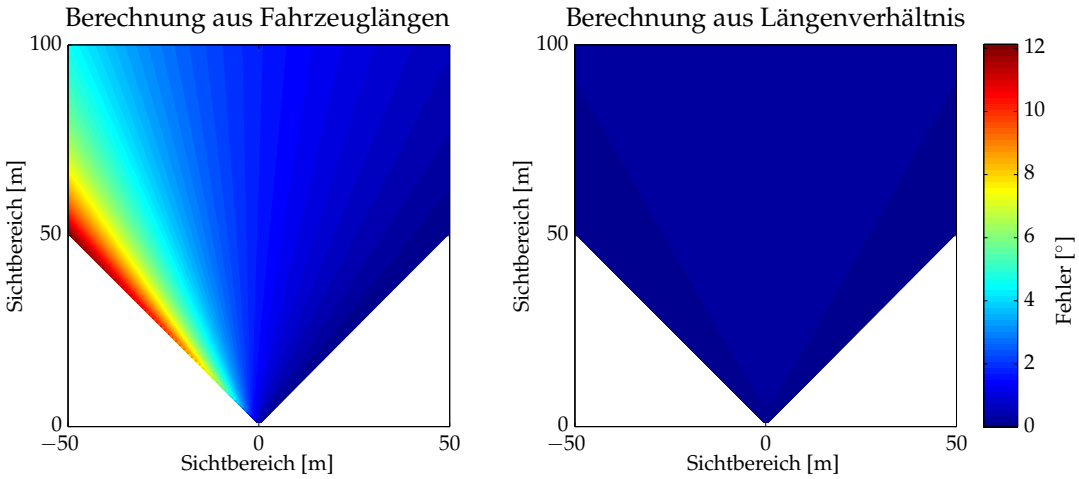


Abbildung 2.10.: Fehler (in Grad) der Aspektwinkel von Modellproblem 1. Bei der Berechnung durch die Fahrzeuglängen (linke Graphik) zeigt sich ein größerer Fehler als bei der Berechnung durch das Längenverhältnis (rechte Graphik). Der Fehler ist abhängig von der Position des Fahrzeugs. Je „spitzer" auf die maßgebende Seite des Fahrzeugs geblickt wird, desto größer ist der Fehler. Wird statt des Winkels α der Gegenwinkel β berechnet, sind die großen Fehler am entgegengesetzten Ende des Sichtbereichs, da in diesem Fall rechts der spitzere Winkel ist.

Fahrzeugs, bis zu 12°. Je spitzer die Seite des Fahrzeugs zu sehen ist, desto größer ist der Fehler. Bei der Berechnung mit dem zweiten Ansatz (rechte Graphik) ist der Fehler geringer als 2°. Für das erste Modellproblem ist also der zweite Ansatz besser für die Berechnung geeignet.

Der Fehler beim zweiten Modellproblem (Abb. 2.11) ist bei beiden Ansätzen gering. Bei Berechnung aus den Fahrzeuglängen ist der maximale Fehler kleiner als 2°, wobei dieser Fall nur am Rand des Sichtbereichs auftritt. Die Berechnung aus den Längenverhältnissen weicht nur marginal vom Sollergebnis ab. Somit ist auch für das zweite Modellproblem der zweite Berechnungsansatz besser geeignet, wobei die Unterschiede weniger stark ausfallen.

Das dritte Modellproblem zeigt die größten Unterschiede zwischen den beiden Möglichkeiten der Aspektwinkelberechnung. Wird α alleine aus der Fahrzeugbreite berechnet, so beträgt die Abweichung über 60°. Auch hier zeigt sich wieder der schon bei der Untersuchung von Modellproblem 1 sichtbare Effekt, dass der Fehler umso größer wird, je „spitzer" auf die maßgebende Seite geblickt wird. Beim dritten Modellproblem ist der korrekte Winkel mit 10° allerdings deutlich spitzer als bei Modellproblem 1 (45°). Folgerichtig zeigt sich im dritten Modellproblem dieses Problem sehr viel deutlicher. Wird der Aspektwinkel aus den Längenverhältnissen berechnet, liegt der Fehler, wie schon bei den beiden ersten Modellproblemen, deutlich unter dem Fehler vom ersten Ansatz.

Insgesamt gesehen ist der zweite vorgestellte Ansatz deutlich robuster gegenüber Modellfehlern. In allen untersuchten Modellproblemen ist der Fehler kleiner als bei der Alternative, der Berechnung unter Annahme der Fahrzeuglänge/-breite. Zusätzlich müssen für den zweiten Ansatz keine Entfernungsinformationen vorliegen, was im ersten Ansatz der Fall sein muss. Nachteilig ist allerdings, dass für eine Berechnung immer alle drei maßgebenden Punkte sicht-

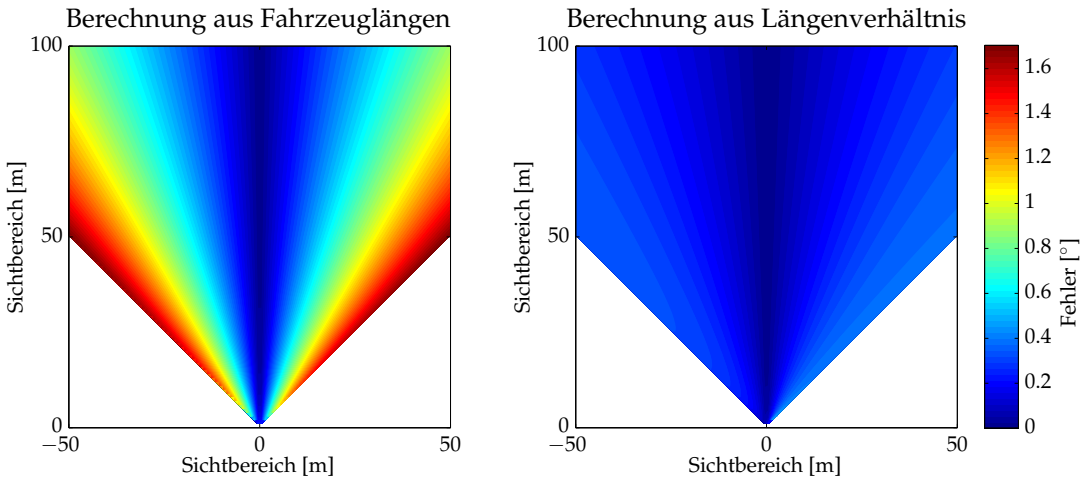


Abbildung 2.11.: Fehler (in Grad) der Aspektwinkel von Modellproblem 2. Der Modellfehler wirkt sich beim zweiten Modellproblem nicht sehr stark aus. Bei beiden Ansätzen der Aspektwinkelberechnung ist er geringer als 2°, wobei der zweite Ansatz (rechte Graphik) leicht besser abschneidet.

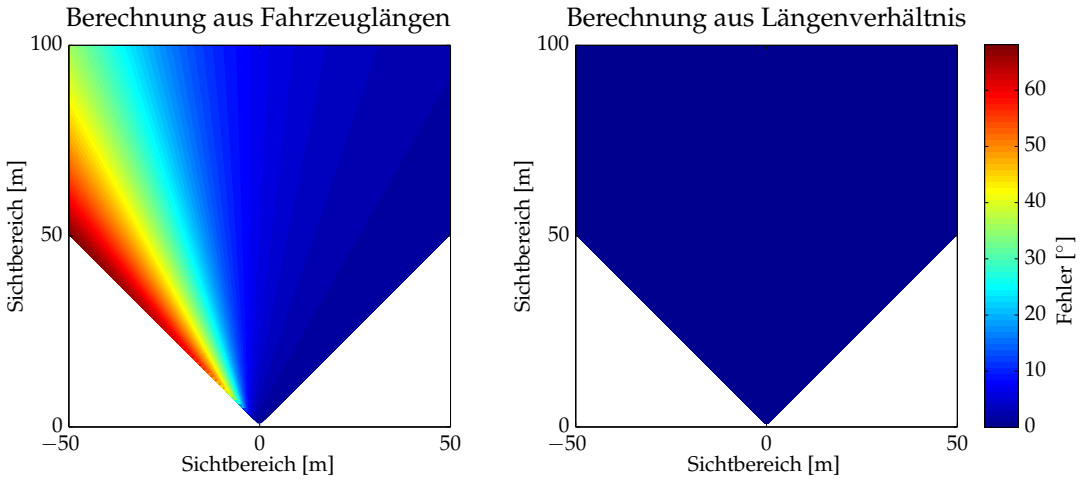


Abbildung 2.12.: Fehler (in Grad) der Aspektwinkel von Modellproblem 3. Auch hier zeigt sich der bereits beim ersten Modellproblem (Abb. 2.12) zu beobachtende Effekt, dass sich der Modellfehler stark auswirkt, sobald die maßgebende Seite unter einem spitzen Winkel zu sehen ist. Wird der Askpektwinkel mit dem ersten Ansatz berechnet (linke Graphik) führt das zu einer Abweichung von über 60°. Die Berechnung mit dem zweiten Ansatz (rechte Graphik) zeigt, wie bei den anderen Modellproblemen ebenfalls, keine starken Abweichungen.

left 16 **left 14** **left 12** **straight** **right 12** **right 14** **right 16**

Abbildung 2.13.: Beispiele für kategorisierte PKW. Wie zu sehen ist, spiegelt die vorgenommene Kategorisierung den Aspektwinkel wider. Je größer die Zahl, desto „schräger" steht das Fahrzeug zur Kamera. Die Kategorien *left* bzw. *right* geben die sichtbare Seite des Fahrzeugs an.

bar sein müssen. Sind nur zwei Punkte sichtbar, kann der Aspektwinkel lediglich nach oben und unten abgeschätzt, jedoch nicht exakt berechnet werden (vgl. Abschnitt 2.6.2).

Die in diesem Abschnitt vorgenommene Untersuchung betrachtet lediglich den inhärenten Modellfehler, d. h. es wird angenommen, dass sich die Bildpunkte A', B und C' exakt bestimmen lassen. Dies ist in der Realität nicht der Fall. Da Detektionen von Fahrzeugen, die die maßgebenden Kanten vorgeben (vgl. Abschnitt 2.4), nicht immer die exakten Ausmaße der Fahrzeuge wiedergeben, weichen schon die Koordinaten von A', B und C' von den tatsächlichen Fahrzeugkanten ab. Der Modellfehler ist lediglich eine *theoretische* Untergrenze des Fehlers. In der Praxis ist zu erwarten, dass der berechnete Winkel noch sehr viel stärker vom tatsächlichen Winkel abweicht.

Da die Winkelberechnung nur mit sehr großen Ungenauigkeiten verbunden ist, wird im realisierten System keine exakte Winkelberechnung vorgenommen. Stattdessen werden Fahrzeuge lediglich in grobe Kategorien eingeteilt (siehe Abschnitt 2.8), welche den Aspektwinkel detektierter Fahrzeuge widerspiegeln, ohne die fehleranfällige und ungenaue Winkelberechnung durchführen zu müssen.

2.8. Kategorisierung assoziierter Detektionen

Statt den Winkel direkt auszurechnen, werden im realisierten System den erkannten Fahrzeugen Kategorien zugewiesen. Diese spiegeln den gesuchten Aspektwinkel wider. Nach der Assoziation (siehe Abschnitt 2.3) sind die Höhe und Breite von zusammengehörigen Detektionen bekannt. Damit kann eine Kategorisierung wie folgt formuliert werden. Die Bezeichnungen der Punkte beziehen sich auf Abb. 2.2.

- Sind die Punkte A und B deckungsgleich, bekommt das Fahrzeug die Kategorie *right*, sind die Punkte C und D deckungsgleich, wird die Kategorie *left* vergeben. Gilt sowohl $A = B$ als auch $C = D$, wird die Kategorie *straight* vergeben. In diesem Fall fährt das Fahrzeug in gerader Linie von der Kamera weg bzw. auf sie zu[2].

[2] Vorder- und Rückfront werden nicht unterschieden, vgl. Abschnitt 1.3.

- Wird in Schritt 1 die Kategorie *left* oder *right* vergeben, lässt sich der Aspektwinkel genauer fassen. Daher wird eine weitere Unterkategorie gewonnen, indem

$$k = \left\lceil 10 \, \frac{\text{Breite der Fahrzeugdetektion (in Pixeln)}}{\text{Breite der Rückfrontdetektion (in Pixeln)}} \right\rceil$$

  berechnet wird. Das so bestimmte k gibt die Abweichung von einer Ausrichtung in gerader Linie zur Kamera an. Je größer k, desto „schräger“ steht das Fahrzeug zur Kamera.

Formal betrachtet ist eine Winkelkategorie c ein Tupel

$$c = (c_1, c_2) \in \{\text{right}, \text{left}, \text{straight}\} \times \mathbb{N}.$$

Für Kategorien mit $c_1 = \text{straight}$ wird $c_2 = 10$ gesetzt. Diese formale Definition ist bei der Evaluation des Gesamtsystems von Bedeutung (siehe Abschnitt 6.2.4).

Abb. 2.13 zeigt Beispiele für PKW zusammen mit den ihnen zugewiesenen Kategorien. Es ist zu sehen, wie die Kategorisierung den Aspektwinkel widerspiegelt.

KAPITEL 3

Detektion von Fahrzeugen mit Kaskadenklassifikatoren

In diesem Kapitel werden die Grundlagen der Detektion von Fahrzeugen mit der Viola-Jones-Kaskade beschrieben. Es wird auf den Algorithmus AdaBoost (Abschnitt 3.1), die verwendeten Merkmale (Abschnitt 3.2), das in dieser Arbeit zum Einsatz kommende Fensterkonzept (Abschnitt 3.3), die Viola-Jones-Kaskade (Abschnitt 3.4) und das verwendete Verfahren zur Generierung von Hypothesen (Abschnitt 3.5) eingegangen. Die vorgestellten Konzepte und Verfahren bilden die Grundlage für die in dieser Arbeit entwickelte hierarchische Erweiterung der Viola-Jones-Kaskade.

3.1. AdaBoost-basiertes Lernen

Boosting ist eine Methode, mehrere schwache Klassifikatoren (sog. *Weaklearner*) zu einem starken Klassifikator, dem sog. *Stronglearner* zu kombinieren. Dabei ist es ausreichend, wenn die Performance der Weaklearner wenig besser als zufällig ist [MR03],[Web02, S. 294 f.]. Basierend auf der Idee des Boosting, stellen Freund und Schapire in [FS95] mit *AdaBoost* (**Ada**ptive **Boost**ing) den ersten praktisch einsetzbaren Algorithmus für die Konstruktion des Stronglearners vor.

Im Folgenden wird zunächst der ursprüngliche Algorithmus aus [FS95] beschrieben (Abschnitt 3.1.1) und einige grundlegende Aussagen über den Trainingsfehler sowie die statistische Interpretierbarkeit (Abschnitt 3.1.2) erläutert. Abschließend wird der auf [FS95] basierende und in dieser Arbeit eingesetzte Boosting-Algorithmus *RealAdaBoost* vorgestellt (Abschnitt 3.1.3).

Für eine ausführlichere Einleitung in Boosting sei auf die Arbeit von Meir und Rätsch [MR03] sowie die Einführung von Schapire [Sch02] verwiesen. Eine mathematische Analyse von AdaBoost bieten die Arbeiten von Friedman et al. [FHT00] und Schapire und Singer [SS99]. Ein Überblick über weitere Boosting-Algorithmen und aktuelle Entwicklungen findet sich in der Arbeit von Zhang und Zhang [ZZ10].

3.1.1. Original AdaBoost von Freund und Schapire

Aus einer Menge von Beispielen

$$\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}\,,\ \underline{x}_i \in \mathcal{M},\ y_i \in \mathcal{C}$$

mit einem Gewicht $w_1(i) = \frac{1}{N},\ i = 1, \ldots, N$ für jedes der Beispiele wird durch AdaBoost ein additives Modell

$$\sum_{t=1}^{T} \alpha_t h_t(\underline{x})$$

konstruiert. Dieses kombiniert T Weaklearner $h_t : \mathcal{M} \to \mathcal{C}$ linear zu einem Stronglearner. Bei der Konstruktion des Modells geht AdaBoost iterativ vor. In jeder Iteration wird derjenige Weaklearner mit dem kleinsten gewichteten Fehler bei der Klassifikation der Beispiele zum finalen Stronglearner hinzugefügt. AdaBoost gehört somit zur Klasse der *Greedy*-Algorithmen [Sch01, S. 185 ff.],[MR03]. Die Gewichtung α_t, mit der h_t in das Modell eingeht, wird hierbei umso größer gewählt, je geringer der gewichtete Trainingsfehler des Weaklearners ist.

Im Anschluss an die Auswahl des neuen Weaklearners h_t werden die Gewichte der Beispiele angepasst. Das Gewicht derjenigen Beispiele, die durch den neuen Weaklearner falsch klassifiziert wurden, wird erhöht; die korrekt klassifizierten Beispiele werden geringer gewichtet. So wird der Fokus für die nächste Iteration auf die bisher falsch klassifizierten Beispiele gelegt.

Der finale Stronglearner H_T ergibt sich nach T Runden zu

$$H_T : \mathcal{M} \to \mathcal{C},\ H_T(\underline{x}) = \operatorname{sign} \sum_{t=1}^{T} \alpha_t h_t(\underline{x}).$$

Abb. 3.1 zeigt den Ablauf von AdaBoost graphisch, Algorithmus 3.1 beschreibt AdaBoost in Pseudocode. Die Begründung der Berechnungsvorschriften von α_t und $w_{t+1}(i)$ befindet sich in Abschnitt 3.1.2.

3.1.2. AdaBoost als Minimierung des exponentiellen Fehlers

Friedman et al. liefern eine Erklärung für die Wahl von α_t und $w_{t+1}(i)$ [FHT00]. Demnach lässt sich AdaBoost interpretieren als schrittweise Minimierung der exponentiellen Fehlerfunktion

$$E(k) = \sum_{i=1}^{N} \exp(-y_i H_T^k(\underline{x}_i)). \tag{3.1}$$

H_T^k ist hierbei das nach k von T Schritten (teilweise) konstruierte Modell

$$H_T^k(\underline{x}) = \sum_{q=1}^{k} \alpha_q h_q(\underline{x}).$$

Die Berechnungsvorschriften für α_t und $w_{t+1}(i)$ ergeben sich von Gleichung (3.1) ausgehend, wenn man die bisherigen α_j und $w_j(i)$ $(j = 1, \ldots, k-1)$ als konstant ansieht und $E(k)$ bzgl.

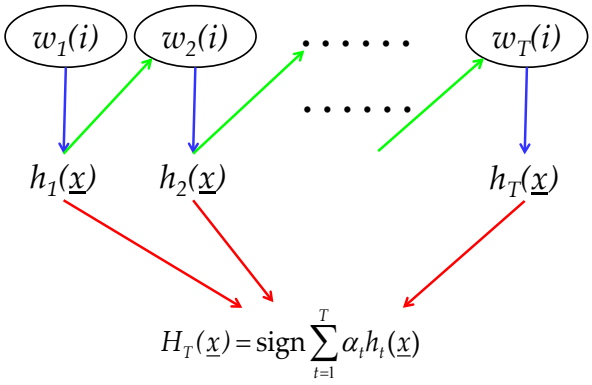


Abbildung 3.1.: Ablauf von AdaBoost. In jeder Runde wird ein Weaklearner h_t basierend auf den Beispielgewichten $w_t(i)$ ausgewählt (blaue Pfeile). Der Fehler von h_t beeinflusst die Gewichte der Beispiele für die nächste Runde (grüne Pfeile). Am Ende werden alle ausgewählten Weaklearner linear zu einem Stronglearner kombiniert (rote Pfeile).
Quelle: Darstellung basierend auf [Bis06, S. 658]

α_k bzw. $w_k(i)$ minimiert [Bis06, S. 659 ff.]. Der Trainingsfehler Δ^k_{train} von H^k_T ist in diesem Fall durch

$$\begin{aligned} \Delta^k_{train} = \frac{1}{N} \left| \left\{ y_i \neq \text{sign}\, H^k_T(\underline{x}_i) \right\} \right| &\leq \Delta^k_{max} \\ &= \prod_{q=1}^{k} \sum_{i=1}^{N} w_q(i) \exp(-\alpha_q y_i h_q(\underline{x}_i)) \\ &= \prod_{q=1}^{k} Z_q \end{aligned} \tag{3.2}$$

beschränkt [SS99],[HALL07]. Der Normalisierungsfaktor der Beispielgewichte Z_k ist also eine obere Schranke für den Trainingsfehler in Runde k. Es gilt zudem

$$Z_{k+1}\Delta^k_{max} = \Delta^{k+1}_{max} \leq \Delta^k_{max}$$

sowie

$$0 \leq \Delta^k_{max} < 1$$

und damit

$$\lim_{k\to\infty} \Delta^k_{max} = 0.$$

Folglich verschwindet bei Wahl eines ausreichend großen T der Trainingsfehler [Sch02]. Für einen ausführlichen und kommentierten Beweis dieser Aussage sei auf [Wen03, S. 31 ff.] verwiesen.

In der Praxis ist der tatsächliche Fehler oftmals kleiner als die durch Gleichung (3.2) vorgegebene Schranke [Wen03, S. 62 f.]. Dadurch kann es allerdings sein, dass ein neu hinzugefügter Weaklearner h_{k+1} den Trainingsfehler Δ^{k+1}_{train} *vergrößert*, ohne Gleichung (3.2) zu verletzen. Um

Algorithmus 3.1 Original AdaBoost von Freund und Schapire

Eingabe : Beispiele: $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$, $\underline{x}_i \in \mathcal{M}$, $y_i \in \mathcal{C}$
Anzahl der Lernschritte: T

Ausgabe : Stronglearner H_T

Initialisiere: $w_1(i) \leftarrow \frac{1}{N}$, $i = 1, \ldots, N$

for $t = 1$ **to** T **do**

`// Wähle Weaklearner` h_t `mit dem kleinsten gewichteten Fehler` ϵ

$h_t \leftarrow \underset{h_j}{\operatorname{argmin}}\, \epsilon_j$, mit $\epsilon_j = \sum_{i=1}^{N} w_t(i) \left[h_j(\underline{x}_i) \neq y_i\right]$

`// Bestimme Gewichtung des neuen Weaklearners`

$\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

`// Adaptiere und normalisiere die Gewichte für die nächste Runde`

$Z_t \leftarrow \sum_{i=1}^{N} w_t(i) \exp(-\alpha_t y_i h_t(\underline{x}_i))$

$w_{t+1}(i) \leftarrow \frac{1}{Z_t} w_t(i) \exp(-\alpha_t y_i h_t(\underline{x}_i))$

end

`// Stronglearner` H_T `ergibt sich zu:`

$H_T(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\underline{x})\right)$

dieses Phänomen zu begrenzen und Weaklearner mit negativem Einfluss zu entfernen, schlagen Li et al. einen Backtracking-Mechanismus vor [LZSZ02],[LZZ+02]. Eine ausführliche Erklärung des resultierenden *FloatBoost*-Algorithmus findet sich bspw. in [Ape05, S. 24 ff.]. Außerdem setzt der Konvergenzbeweis die Existenz *beliebig vieler* Weaklearner, deren Performance besser als zufällig sein muss, voraus. Diese Voraussetzung ist in der praktischen Anwendung nicht gegeben; Praktisch existieren nur begrenzt viele Weaklearner. D. h. ab einer gewissen Runde wird der Trainingsfehler nicht mehr kleiner werden.

3.1.3. RealAdaBoost von Schapire und Singer

Ausgehend vom ursprünglichen AdaBoost [FS95], entwickeln Schapire und Singer mit RealAdaBoost [SS99] eine generalisierte Version des Verfahrens, das eine höhere Performance als der Originalalgorithmus aufweist [LZZ+02],[WAHL04],[MKH05]. Im Gegensatz zu diesem geht RealAdaBoost von reellen Weaklearnern $f_t : \mathcal{M} \to \mathbb{R}$ aus. Hierbei ist sign $f_t(\underline{x})$ die Klasse, die f_t für $\underline{x}$ vorschlägt; $|f_t(\underline{x})|$ ist interpretierbar als die Sicherheit (engl. *confidence*), die f_t bei dieser Aussage hat [FHT00].

Statt der exponentiellen Fehlerfunktion $E(k)$ (Gleichung (3.1)) minimiert RealAdaBoost direkt die durch Z_t gegebene obere Schranke für den Trainingsfehler in Runde t. Da $f_t(\underline{x})$ reell ist, lässt sich ohne Verlust der Allgemeinheit die Berechnung von α_t in den Weaklearner verlagern. Somit lässt sich der vom gewählten Weaklearner f_t abhängige Fehler Z_t schreiben als [SS99]

$$Z_t(f_t) = \sum_{i=1}^{N} w_t(i) \exp(-y_i f_t(\underline{x}_i)) \,. \tag{3.3}$$

Mit diesem Maß lässt sich ein iterativer Algorithmus formulieren: In jeder Runde wird derjenige Weaklearner ausgewählt und zum finalen Stronglearner hinzugefügt, für den Gleichung (3.3) minimal wird.

Beschränkt man sich auf eine bestimmte Klasse von Weaklearnern, die sog. Eingaberaum-zerlegenden (engl. *domain-partitioning*) Weaklearner, lässt sich diese recht allgemeine Aussage noch konkretisieren und RealAdaBoost dazu nutzen, effiziente Weaklearner dieser Art zu konstruieren.

Eingaberaum-zerlegende Weaklearner

Stützen die Weaklearner ihre Aussage allein auf eine Zerlegung des Eingaberaums, so lässt sich Gleichung (3.3) minimieren, indem für jeden Teilbereich ein spezieller Wert $c_j \in \mathbb{R}$ gewählt wird. Beispiele für Weaklearner dieser Art sind Klassifikationsbäume (siehe Abschnitt 4.1) sowie die von Viola und Jones [VJ01] verwendeten Decision-stumps (siehe Abschnitt 3.4).

Sei ein auf einer Zerlegung des Eingaberaums basierender Weaklearner f gegeben durch

$$f(\underline{x}) = \begin{cases} c_j, & \text{falls } \underline{x} \text{ in einer bestimmten Teilmenge des Eingaberaums} \\ 0, & \text{sonst} \end{cases}$$

wobei die Werte c_j nicht näher bestimmt sind und durch die Minimierung von Z_t gefunden werden. Sei weiter $W_j^{\pm}$ der (gewichtete) Anteil an positiven/negativen Beispielen in der betrachteten Teilmenge des Eingaberaums. Dann ist $Z_t(f)$ minimal, wenn

$$c_j = \frac{1}{2} \ln \left(\frac{W_j^+}{W_j^-} \right). \tag{3.4}$$

In diesem Fall lässt sich $Z_t(f)$ durch

$$Z_t(f) = 2 \sum_j \sqrt{W_j^+ W_j^-} \tag{3.5}$$

berechnen [SS99]. Zu beachten ist, dass hierbei keine Aussage über die Zerlegung selbst gemacht wird. Die Wahl der c_j gemäß Gleichung (3.4) minimiert Z_t für eine beliebige Zerlegung; es bleibt weiterhin Aufgabe des Weaklearners, diese zu finden.

Die mittels Gleichung (3.5) berechnete Größe wird *Z-Wert* des Weaklearners f genannt und kann als die „Stärke“ oder „Mächtigkeit“, mit der f zwischen den Positiv- und Negativbeispielen unterscheiden kann, interpretiert werden [WN07].

3.2. Verwendete Merkmale

In diesem Abschnitt werden die in dieser Arbeit zum Einsatz kommenden Merkmale vorgestellt. Neben den von Viola und Jones verwendeten [VJ01] Haarwavelets (Abschnitt 3.2.1) werden die von Levi und Weiss eingeführten EOH-Merkmale (Abschnitt 3.2.2) [LW04] eingesetzt.

Eng mit den verwendeten Merkmalen verknüpft ist das dieser Arbeit zu Grunde liegende Konzept der Merkmalsfenster. Auf dieses wird in Abschnitt 3.3 eingegangen.

3.2.1. Haarwavelets

Die von Oren et al. als Merkmal vorgeschlagenen [OPS+97] und von Papageorgiou et al. weiterentwickelten [POP98] Haarwavelets bestehen aus gewichteten Rechtecken, die Helligkeitsbeziehungen zwischen Bildregionen erfassen. Sie sind somit eine Verallgemeinerung des klassischen Prewitt-Filters [Jä05, S. 365 ff.]. Analog zum Prewitt-Filter ergibt sich der Merkmalswert, indem die Pixel unter den Rechtecken aufsummiert und durch die Rechteckgröße geteilt werden. Anschließend werden die so normalisierten Summen voneinander abgezogen [Kal05, S. 6 ff.].

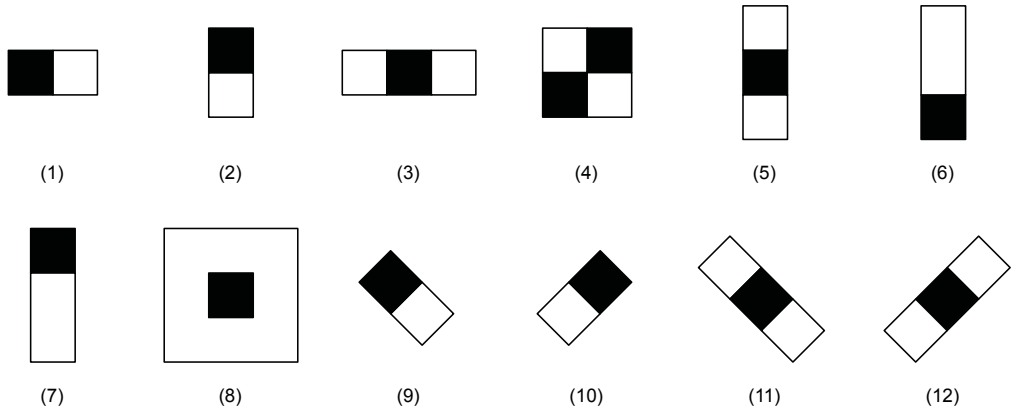
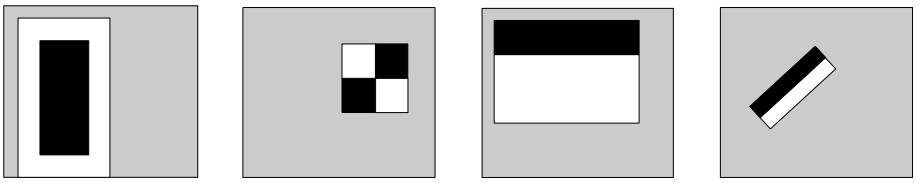


Abbildung 3.2.: Typen von Haarwavelets. Haarwavelets erfassen Helligkeitsunterschiede in Bildern, indem sie die Pixel unter den weißen bzw. schwarzen Rechtecken aufsummieren, die entstehenden Werte durch die jeweilige Rechteckgröße teilen und anschließend voneinander abziehen [VJ01],[Kal05, S. 6 ff.]. Die Abbildung zeigt mögliche Basistypen, die in verschiedenen Beiträgen [VJ01],[Ape05],[LKP03] vorgeschlagen wurden.
Quellen: (1) – (4): [VJ01], (5) – (7): [Ape05], (8) – (12): [LKP03]

Einen Überblick über die Basistypen an Haarwavelets, von denen ausgehend die verschiedenen Merkmale definiert werden können, gibt Abb. 3.2. Zusätzlich zum Basistyp wird die Position innerhalb des Merkmalsfensters (siehe Abschnitt 3.3) sowie die Skalierung in x- und y-Richtung benötigt, um ein Merkmal eindeutig zu identifizieren. Abb. 3.3 zeigt beispielhaft vier der möglichen Merkmale innerhalb eines quadratischen Merkmalsfensters.

Die Menge der möglichen Merkmale, die auf diese Weise definiert werden können, ist sehr groß. Allein für die von Viola und Jones verwendeten Typen (1) – (4) können innerhalb eines 24 mal 24 Pixel großen Merkmalsfensters über 160 000 verschiedene Kombinationen unterschieden werden [VJ01].

Der Hauptvorteil der Verwendung von Haarwavelets als Merkmale ist die hohe Geschwindigkeit, mit der diese ausgewertet werden können. Mit dem von Viola und Jones verwendeten *Integralbild* lässt sich ein normales, d. h. nicht-diagonales Haarwavelet mit nur vier atomaren Rechenoperationen berechnen [VJ01]. Um zusätzlich die diagonalen Typen (9) – (12) effizient zu ermöglichen, erweitern Lienhart et al. die originale Datenstruktur auf ein diagonales Integralbild [LKP03].



(a) Typ (8) (b) Typ (4) (c) Typ (7) (d) Typ (9)

Abbildung 3.3.: Beispiele für mögliche Haarwavelets in einem quadratischen Merkmalsfenster. Ein Merkmal ist definiert duch seinen Typ (siehe Abb. 3.2), der Skalierung in x- und y-Richtung sowie der Position innerhalb des Merkmalsfensters.

Von den in Abb. 3.2 gezeigten Basistypen kommen in dieser Arbeit nur die Typen (1) – (8) zum Einsatz. Auf die diagonalen Haarwavelets wurde verzichtet um die Gesamtanzahl der Merkmale in einem beherrschbaren Bereich zu halten.

3.2.2. Edge Orientation Histograms

Merkmale, die auf orientierten Gradientenhistogrammen (*Histogram of Oriented Gradients*, HOG) basieren, werden schon an verschiedenen Stellen, z. B. der Fußgängererkennung [DT05] oder um Objekte anhand lokaler Merkmale (sog. Keypoints) zu finden [Low04], erfolgreich eingesetzt. Sie werden gebildet, indem in einer Bildumgebung Histogramme der Gradientenrichtung aufgestellt werden (Abb. 3.4). Diese Histogramme erfassen die vorherrschenden Richtungen und spiegeln somit die „Form“ der Kanten in der betrachteten Umgebung wider [ZZ10]. Für eine umfassende Einführung in Gradientenhistogramme sei auf [Wal08, S. 66 ff.] verwiesen.

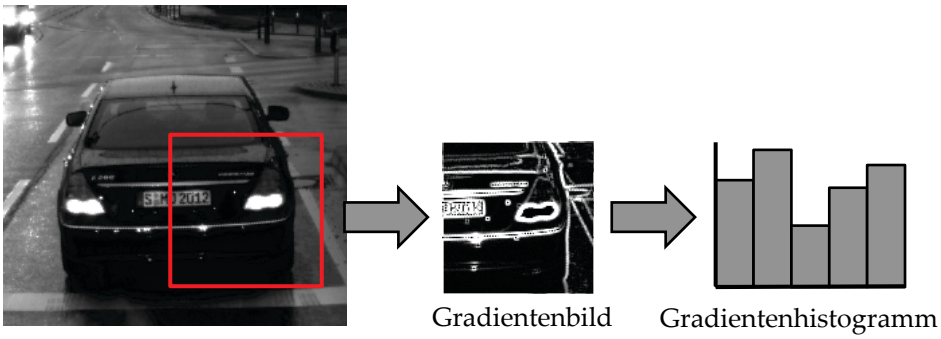


Abbildung 3.4.: Merkmalsextraktion von Gradientenhistogrammen. Gradientenhistogramme werden gebildet, indem in einer Bildumgebung (rot) das Gradientenbild berechnet wird. Damit kann das Histogramm der Gradientenrichtungen aufgestellt werden.

Um Gradientenhistogramme für das Training mit AdaBoost einsetzen zu können, müssen die eigentlich vektorwertigen Histogramme auf einen skalaren Wert reduziert werden. Diesen Schritt vollziehen Levi und Weiss in [LW04]. Die von ihnen vorgeschlagenen *Edge Orientation Histograms* (EOH) gehen von Gradientenhistogrammen aus und setzen die einzelnen Komponenten – sprich: die einzelnen Gradientenrichtungen – in Beziehung zueinander. Bspw. kann der

Anteil einer einzelnen Gradientenrichtung k an der gesamten Gradientenstärke aller Richtungen ausgedrückt werden, indem

$$A_k(R) = \frac{E_k(R)}{\sum_i E_i(R)} \tag{3.6}$$

berechnet wird [LW04]. Hierbei ist $E_i(R)$ der i-te Eintrag des Gradientenhistogramms der Region R. Eine weitere Möglichkeiten ist die Beziehung zwischen zwei einzelnen Gradientenrichtungen k und l, ausdrückbar als

$$B_{k,l}(R) = \frac{E_k(R)}{E_l(R)}.$$

Durch diesen Ansatz können sehr viele verschiedene Merkmale innerhalb eines Merkmalsfensters (siehe Abschnitt 3.3) definiert werden. Selbst bei Beschränkung auf die Beziehung von Gleichung (3.6) sind für jede mögliche Umgebung innerhalb des Merkmalsfensters so viele Merkmale, wie das zugrunde liegende Histogramm Einträge hat, möglich. Die Zahl der möglichen EOH-Merkmale ist somit ähnlich groß wie die Zahl der möglichen Haarwavelets. Aus diesem Grund wird in dieser Arbeit lediglich die durch Gleichung (3.6) definierte Beziehung verwendet.

Auch EOH-Merkmale lassen sich sehr effizient berechnen. Ähnlich zur Struktur des Integralbildes für Haarwavelets kommt hierbei für EOH-Merkmale das von Porikli [Por05] entwickelte *Integralhistogramm* zum Einsatz.

Analog zur Definition der Haarwavelets ist ein EOH-Merkmal eindeutig durch die Position und Skalierung in x- und y-Richtung sowie zusätzlich durch eine Gradientenrichtung, die in Beziehung zu den anderen gesetzt wird, definiert. Als Umgebung wird hierbei immer von einer quadratischen Grundform ausgegangen, die in x- und y-Richtung skaliert wird. Die Skalierung erfolgt unabhängig voneinander, d. h. es sind auch nicht-quadratische Formen möglich.

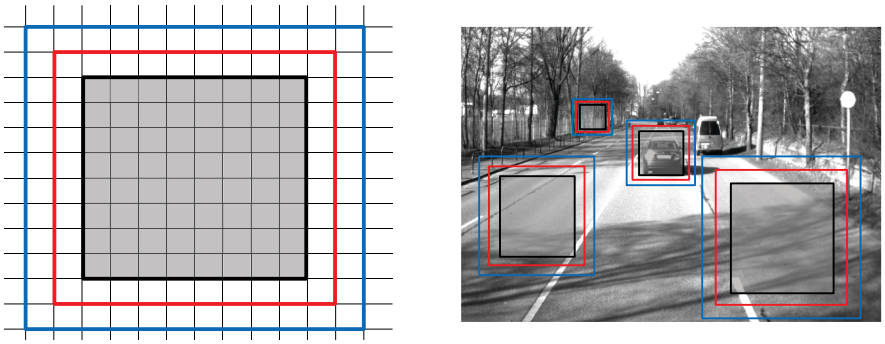
Die zum Einsatz kommenden EOH-Merkmale unterscheiden Gradientenrichtungen, jedoch nicht *Gradientenorientierungen* (die Vorzeichen der Gradienten) [Wal08, S. 67 f.] bei der Extraktion der Gradientenhistogramme. Die Anzahl Richtungen, die unterschieden werden, ist auf 5 festgelegt; Die Histogramme haben folglich 5 Einträge.

3.3. Norm-, Merkmals- und Objektfenster

Um Merkmale eindeutig und skalierungsinvariant angeben zu können, muss die Beziehung zwischen einem Merkmal, den Hypothesen und der Position des gesuchten Objektes innerhalb einer Hypothese festgehalten werden. Hierfür bedarf es eines Ansatzes, der Größen- und Positionsangaben für Merkmale innerhalb einer Hypothese skalierungsinvariant erlaubt. In diesem Abschnitt wird das Konzept der Norm-, Merkmals- und Objektfenster erläutert. Es geht zurück auf [Kal05, S. 25 ff.].

Das *Normfenster* gibt das Seitenverhältnis der Hypothesen vor. Es kann als normierte Hypothese angesehen werden. Innerhalb des Normfensters werden sowohl das *Merkmalsfenster* als auch das *Objektfenster* pixelgenau definiert. Das Merkmalsfenster ist der Bereich innerhalb des Normfensters, in dem sich die Merkmale (siehe Abschnitt 3.2) befinden. Das Objektfenster gibt die Position des zu detektierenden Objektes, bezogen auf das Normfenster, an. Die Anzahl der Merkmals- bzw. Objektfenster je Normfenster ist nicht begrenzt. Es ist daher möglich, mehrere

Merkmals- und Objektfenster für ein Normfenster zu definieren. Das ist bspw. dann sinnvoll, wenn mehrere Objekt-Typen mit stark unterschiedlichem Seitenverhältnis mit dem gleichen Klassifikator erkannt werden sollen. Gemeinsame Merkmale müssen dann nur einmal ausgewertet werden (*Feature Sharing* [TMF07]). In [Kal05] werden auf diese Weise PKW gemeinsam mit Fußgängern detektiert. In dieser Arbeit wird diese Technik genutzt, um LKW gemeinsam mit PKW zu verarbeiten (siehe Abschnitt 6.1.2).



(a) (b)

Abbildung 3.5.: Norm-, Merkmals- und Objektfenster. (a) Norm- (blau, äußeres Fenster), Merkmals- (rot, mittleres Fenster) und Objektfenster (schwarz, grau hinterlegt) geben pixelgenau den Zusammenhang zwischen Hypothese, Bereich der Merkmale und Objekt an. Typischerweise wird die hier gezeigte Konstellation Objektfenster $\subset$ Merkmalsfenster $\subseteq$ Normfenster gewählt. (b) Eine Hypothese ergibt sich, indem das Normfenster skaliert und an eine Position innerhalb des Suchtunnels (siehe Abschnitt 3.5) verschoben wird. Merkmals- und Objektfenster werden dabei im gleichen Maße skaliert und entsprechend ihrer Position innerhalb des Normfensters in der Hypothese positioniert.

Typischerweise wird die Konstellation Objektfenster $\subset$ Merkmalsfenster $\subseteq$ Normfenster gewählt[1]. Durch die Wahl des Merkmalsfensters als Obermenge des Objektfensters ist es möglich, die „Umgebung“ (charakteristische Kanten von Fahrzeugen etc.) der Objekte bei der Merkmalsextraktion mit einzubeziehen [Kal05, S. 29]. Abb. 3.5(a) zeigt diese Konstellation beispielhaft an einem quadratischen Normfenster.

Die Beziehung zwischen den Fenstern bleibt auch bei der Hypothesengenerierung (siehe Abschnitt 3.5) erhalten. Eine Hypothese ergibt sich, indem das Normfenster skaliert und an eine Position innerhalb des Suchtunnels verschoben wird. Die für das skalierte Normfenster definierten Merkmals- und Objektfenster werden dabei im gleichen Maße skaliert und entsprechend ihrer Position innerhalb des Normfensters im Bild positioniert. Die im Merkmalsfenster definierten Merkmale werden gemeinsam mit dem Merkmalsfenster skaliert. Das ist ein Unterschied zum „klassischen“ Vorgehen, bei dem das Bild mit Hilfe einer Bildpyramide in verschiedenen Skalierungen untersucht wird: Statt das Bild zu skalieren werden die Hypothesen und damit die Merkmale in ihrer Größe angepasst, um Objekte auf verschiedenen Skalierungsstufen zu detektieren.

[1] Die Mengensymbole sind in diesem Fall als „Ist-In“-Beziehung zu verstehen.

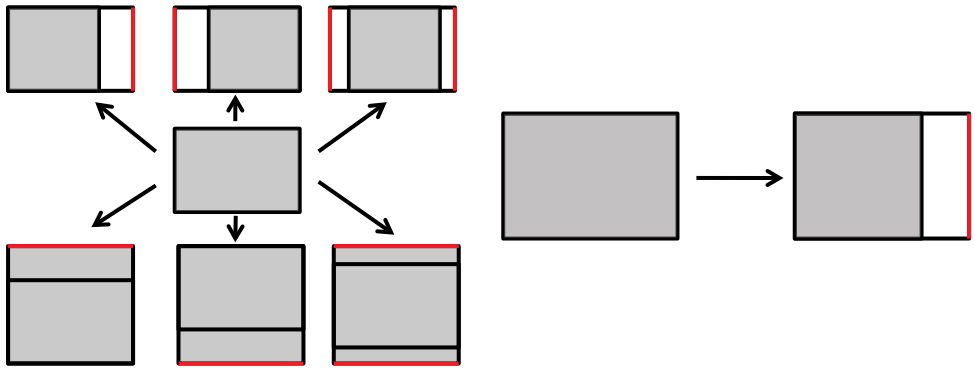


Abbildung 3.6.: Anpassung der Label auf das Objektfenster. (a) Soll das Label (Mitte) an das Format des Objektfensters (in diesem Beispiel: quadratisch) angepasst werden, ergeben sich sechs Möglichkeiten (äußere Rechtecke). Die roten Kanten werden jeweils so gewählt, dass das geforderte Seitenverhältnis hergestellt ist. (b) Bei Wahl eines ungünstigen Objektfensters oder einer ungünstigen Anpassungsmethode kann das Objektfenster nicht das gesamte Label enthalten.
Quelle: (a): Eigene Darstellung basierend auf [Kal05, S. 27]

Um die Daten für das Training aufzubereiten und die vorhandenen Label (siehe Abschnitt 1.5) auf das durch das Objektfenster vorgegebene Format zu bringen, wird in einem Vorverarbeitungsschritt eine Anpassung der Label vorgenommen. Diese Anpassung kann auf sechs verschiedene Arten erfolgen. Abb. 3.6(a) zeigt die Möglichkeiten graphisch. Dabei ist nicht gewährleistet, dass das gesamte Label im Objektfenster enthalten ist (Abb. 3.6(b)). Dies muss durch Wahl eines geeigneten Objektfensters und einer geeigneten Anpassung sichergestellt werden.

3.4. Die Viola-Jones-Kaskade

Viola und Jones stellten mit dem Kaskadenklassifikator [VJ01] erstmals ein echtzeitfähiges System zur Gesichtserkennung vor, das als Durchbruch auf dem Gebiet der Gesichtserkennung angesehen ist [ZZ10]. Auch im automobilen Umfeld hat die Viola-Jones-Kaskade zu neuen Impulsen und Verbesserungen, z. B. für automobile Assistenzsysteme [WL04],[MOL+05],[KSPL06], geführt. Diese Arbeit orientiert sich ebenfalls zu großen Teilen an der von Viola und Jones vorgeschlagenen Struktur.

Der Ansatz von Viola und Jones basiert auf der Erkenntnis, dass nur ein kleiner Teil aller möglichen Hypothesen eines Eingabebildes tatsächlich das gesuchte Objekt zeigen; die große Mehrheit zeigt Hintergrund. Folglich ist es ausreichend, sehr schnell entscheiden zu können, welche Hypothese *kein* Objekt zeigt, damit ein Klassifikator – im Mittel über alle Hypothesen gesehen – sehr schnell arbeitet.

Von dieser Idee ausgehend, strukturieren Viola und Jones ihren Klassifikator in kaskadierten Stufen, die hintereinander abgearbeitet werden; d. h. Hypothesen werden von Stufe zu Stufe weitergegeben. In jeder Stufe wird durch einen mittels AdaBoost (siehe Abschnitt 3.1.1) konstruierten Klassifikators entschieden, ob eine Hypothese in die nächste Stufe weitergereicht wird:

wird sie als Hintergrund klassifiziert, wird ihre Bearbeitung abgebrochen. Dadurch nimmt die Zahl der Hypothesen mit jeder Stufe kontinuierlich ab. Passiert eine Hypothese die letzte Stufe, wird ihr die Klasse „Objekt“ zugeordnet. Abb. 3.7 veranschaulicht den Aufbau der Viola-Jones-Kaskade.

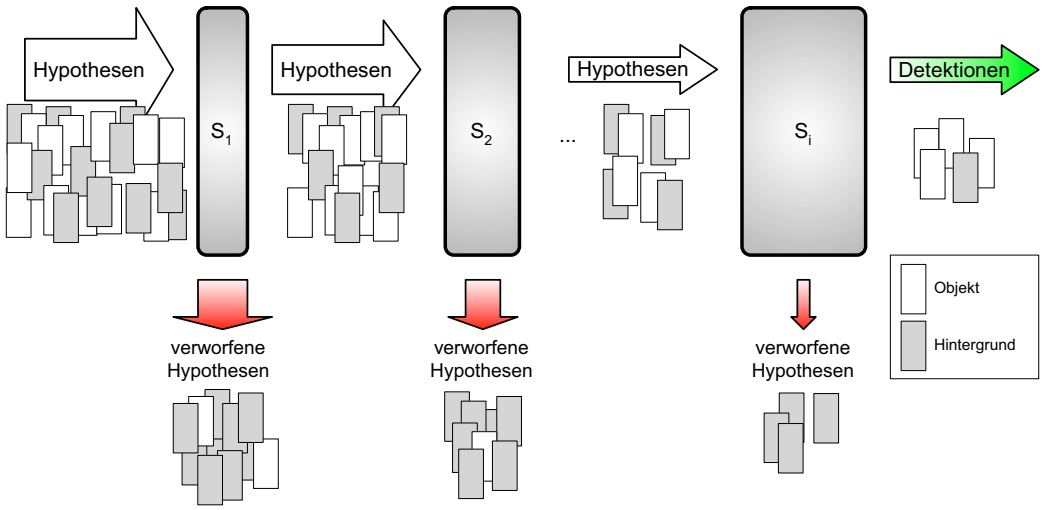


Abbildung 3.7.: Struktur der Viola-Jones-Kaskade und Ablauf der Hypothesenverarbeitung. Hypothesen werden von Stufe zu Stufe weitergereicht. Weist eine Stufe eine Hypothese zurück, so wird sie nicht an die nächste Stufe weitergegeben. Dadurch nimmt die Zahl der Hypothesen in den späteren Stufen immer weiter ab. Hypothesen, die die letzte Stufe passieren, werden als Objekt klassifiziert. Quelle: [Kal05]

Durch den kaskadierten Aufbau des Klassifikators wird die Entscheidung „Hintergrund“ schnellstmöglich, d. h. mit wenigen Weaklearnern, getroffen. In den späteren Stufen kann sich der Klassifikator auf die „schweren“ Hintergrund-Hypothesen, also solche, die Objekten ähneln, konzentrieren.

Um die komplexeren Entscheidungen in den späten Stufen treffen zu können, wird die Anzahl der Weaklearner je Stufe zunehmend größer gewählt. Viola und Jones bspw. verwenden Klassifikatoren mit 2, 10 und 25 Weaklearnern für die Stufen 1, 2 und 3. Insgesamt erstellen sie eine Kaskade mit 38 Stufen für ihre Experimente [VJ01].

Als Weaklearner für AdaBoost verwenden Viola und Jones die in Abschnitt 3.2.1 vorgestellten Haarwavelets zusammen mit einem einzelnen Schwellwert, der positive von negativen Beispielen trennt. Das entspricht einem Entscheidungsbaum der Tiefe 1, dem sog. *Decision-stump*. Durch diese Kombination fungiert AdaBoost als Algorithmus zur Auswahl von Merkmalen: In jeder Runde wird aus der Liste aller möglichen Merkmale dasjenige, das am besten zwischen Hintergrund und Objekt trennen kann, ausgewählt und mit einem Schwellwert versehen. Im Fall von Haarwavelets heißt das, dass alle verfügbaren Typen (siehe Abb. 3.2) in allen Skalierungen an allen möglichen Positionen innerhalb des Merkmalsfensters (siehe Abschnitt 3.3) aufgezählt und bzgl. ihrer Diskriminanz beurteilt werden müssen.

3.4.1. Training einer Kaskade

Eine Kaskade wird iterativ von der ersten Stufe an trainiert. Für jede Stufe wird mittels AdaBoost ein Klassifikator erstellt. Die Trainingsbeispiele für die neue Stufe werden gesammelt, indem Hypothesen durch die bereits erstellten Stufen klassifiziert werden. Wird eine Hypothese durch eine der früheren Stufen abgewiesen, so ist sie nicht Teil der Beispiele für die neue Stufe. Auf diese Weise wird jede Stufe nur für Hypothesen trainiert, die sie auch während der Anwendung als Eingabe bekommen kann. Dieser Ablauf wird entweder so lange wiederholt, bis keine Negativbeispiele mehr gefunden werden (sog. *erschöpfendes Training*), oder bis eine zuvor festgelegte Stufenanzahl erreicht ist.

Für das Training der einzelnen Stufen können ebenfalls verschiedene Vorgaben gemacht werden. Neben einer festen Anzahl Weaklearner kann eine minimale Detektionsrate zusammen mit einer maximalen Falschalarmrate für die Trainingsbeispiele vorgeben werden. In diesem Fall fügt AdaBoost so lange weitere Weaklearner hinzu, bis die geforderten Werte erreicht sind. Zudem ist eine Kombination, also Performance-Vorgaben gepaart mit einer maximalen Anzahl Weaklearner, möglich. Mit dieser Strategie haben Viola und Jones den Großteil der von ihnen verwendeten 38 Stufen trainiert [VJ01].

Kommt solch eine kombinierte Trainings-Vorgabe zum Einsatz, wird das von AdaBoost erstellte Modell H_T derart erweitert, dass

$$H_T(\underline{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\underline{x}) - \theta_s\right),$$

mit einer Schwelle θ_s für die Stufe s. θ_s wird nach jeder Boosting-Runde so gewählt, dass die geforderte Detektionsrate erreicht ist. Ist für das so gewählte θ_s die Falschalarmrate kleiner als die Vorgabe, so kann das Training der Stufe beendet werden.

Die finale Detektionsrate D des Kaskadenklassifikators ist durch

$$D = \prod_i d_i,$$

die Falschalarmrate F durch

$$F = \prod_i f_i$$

bestimmt [VJ01],[Kal05, S. 15]. d_i und f_i sind hierbei die Detektions- bzw. Falschalarmraten der einzelnen Stufen. Eine hohe Detektionsrate D ist folglich nur bei nahezu perfekten Detektionsraten in den einzelnen Stufen erreichbar. So werden bspw. bei einer Kaskade mit zehn Stufen für $D \geq 90\,\%$ bereits $d_i \geq 99\,\%$ benötigt ($0.99^{10} \approx 0.9$). Dieser ungünstige Umstand wird allerdings durch die Tatsache entkräftet, dass auch sehr hohe Falschalarmraten in den einzelnen Stufen zu einer niedrigen Falschalarmrate des kompletten Kaskadenklassifikators führen. Bereits bei einer Falschalarmrate von 50 % je Stufe ergibt sich für zehn Stufen $F = 0.001 \approx 0.5^{10}$. Daher ist durch Wahl eines geeigneten θ_s eine hohe Detektionsrate sehr gut erzielbar.

3.4.2. Verbesserung der Weaklearner durch mehrere Schwellwerte

Obwohl Viola und Jones den originalen AdaBoost (siehe Abschnitt 3.1.1) zur Auswahl der Merkmale verwenden, ist der von ihnen vorgeschlagene Trainingsaufbau besser für den später entwickelten RealAdaBoost geeignet [ZZ10]. Dieser macht Aussagen über die optimale Wahl

der Weaklearner-Ausgaben, falls diese Eingaberaum-zerlegend (siehe Abschnitt 3.1.3) arbeiten. Die von Viola und Jones verwendeten Decision-stumps sind von dieser Art.

Offen bleibt allerdings die Wahl der Zerlegung, d. h. im Fall von Decision-stumps die Wahl des Schwellwerts. Üblicherweise wird ein Schwellwert gewählt, der die Wahrscheinlichkeit eines Klassifikationsfehlers minimiert (*Maximum-A-Posteriori*-Regel, kurz: MAP-Regel) [RPP06],[DHS00, S. 87 f.].

Oftmals ist die Beschränkung auf lediglich einen Schwellwert, d. h. eine Zerlegung in zwei Teilmengen, nicht optimal. Durch eine feinere Zerlegung werden die Wahrscheinlichkeiten für Objekt bzw. Hintergrund genauer erfasst [HALL07]. Damit können auch Mengen getrennt werden, die sich mit einem einzelnen Schwellwert nur mit großem Fehler trennen lassen. Abb. 3.8 zeigt ein Beispiel für eine solche Konstellation und die Problematik, die bei Wahl eines einzelnen Schwellwerts besteht.

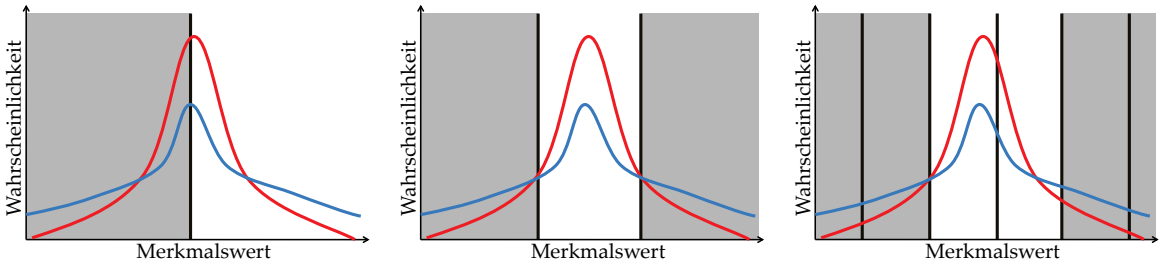


Abbildung 3.8.: Beispiel für eine Wahrscheinlichkeitsverteilung, die sich nicht durch einen Schwellwert trennen lässt. Die roten und blauen Kurven sind die Wahrscheinlichkeitsdichten der Klassen „Objekt“ bzw. „Hintergrund“, abgetragen über dem Wert des gewählten Merkmals. Die grau hinterlegten Bereiche werden dem Hintergrund zugeordnet. (a) Deutlich zu sehen ist, dass in diesem Fall eine einzelne Schwelle nicht optimal zwischen den Klassen trennen kann. (b) Optimale Wahl gemäß der Maximum-A-Posteriori-Regel. (c) Schwellen bei den in der Praxis häufig verwendeten äquidistanten Bins.

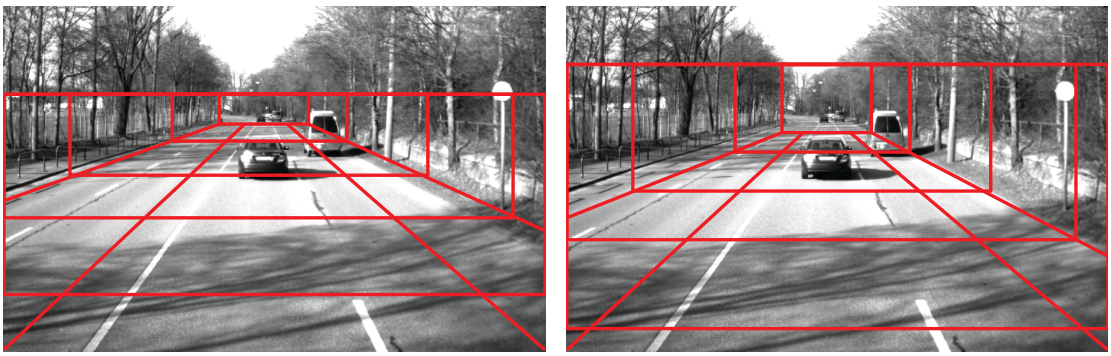
Neben der Möglichkeit, beliebig viele Schwellwerte gemäß der MAP-Regel zu wählen [RPP06], d. h. für jeden Merkmalswert eine optimale Wahl der Klassenzugehörigkeit zu treffen, wird in der Praxis (bspw. von [HALL05],[WN07],[YHN09]) häufig aus Geschwindigkeits- und Einfachheitsgründen ein äquidistantes Histogramm, die sog. *Bins* (siehe Abb. 3.8(c)) verwendet. Die Anzahl der Schwellwerte ist dann um eins kleiner als die Anzahl der Bins. Für jeden Bin wird mittels Gleichung (3.4) der Ausgabewert des Weaklearners bestimmt.

In dieser Arbeit wird allerdings ein alternatives, an [RPP06] angelehntes Verfahren angewandt. Es versucht, bei Vorgabe einer festen Anzahl Bins, die in Abb. 3.8(b) gezeigten, optimalen Schwellwerte zu finden. D. h. die Bins werden nicht notwendigerweise äquidistant gewählt. Da die *konkrete* Vorgehensweise des Verfahrens für die weitere Arbeit unerheblich ist, wird im Weiteren nicht darauf eingegangen. Für weitere Informationen sei auf [RPP06] verwiesen.

3.5. Hypothesengenerierung

Um Objekte in Eingabebildern erkennen zu können, ist es nötig, dass mindestens eine Hypothese das zu erkennende Objekt tatsächlich zeigt. Daher ist die Vorgehensweise, mit der Hypothesen für die Eingabebilder generiert werden, mitentscheidend für die Erkennungsleistung des Gesamtsystems. In diesem Abschnitt wird das verwendete Verfahren zur Generierung von Hypothesen skizziert. Auf eine weitergehende Einführung, speziell der mathematischen Aspekte, wird verzichtet. Hierfür sei bspw. auf [BK08, S. 370 ff.], [Kal05, S. 17 ff.] sowie [EG09] verwiesen.

Wird bei der Generierung der Hypothesen naiv vorgegangen (d. h. indem Hypothesen in allen Skalierungen über das gesamte Eingabebild verteilt werden), übersteigt die Zahl der Hypothesen schnell mehrere Millionen. Damit wäre keine Erkennung in Echtzeit mehr möglich [Kal05, S. 18]. Die Zahl der Hypothesen lässt sich allerdings drastisch reduzieren, wenn der Bildbereich, in dem Objekte erwartet werden, eingeschränkt wird. Da sich die gesuchten Fahrzeuge alle auf der Straße befinden, lässt sich der Bildbereich, für den Hypothesen generiert werden müssen, auf einen Suchtunnel einschränken (siehe Abb. 3.9). Dabei wird angenommen, das der Erdboden eine ungekrümmte Ebene ist (*Ground-Plane-Annahme*). Im verwendeten Weltkoordinatensystem (siehe Abschnitt 2.2) entspricht diese der XY-Ebene.



(a) (b)

Abbildung 3.9.: Suchtunnel, in dem Hypothesen generiert werden. Dieser wird abhängig von den Kameraparametern und der Objekthöhe (z. B. bei PKW ca. 2 Meter) aufgebaut. (a) Suchtunnel bei normaler Ground-Plane-Annahme. (b) Suchtunnel bei relaxierter Ground-Plane-Annahme. Es ist deutlich zu sehen, wie der Suchtunnel einen breiteren Bildbereich umschließt und somit Nickbewegungen ausgeglichen werden.

Aufgrund der in der Realität durch Nickbewegungen des aufnehmenden Fahrzeugs sowie durch Hügel etc. verletzen Ground-Plane-Annahme befinden sich speziell weit entfernte Fahrzeuge schnell außerhalb des Suchtunnels und würden damit nicht mehr detektiert werden. Um diesen Effekt zu vermeiden, wird der Suchtunnel erweitert, indem ein Abweichwinkel ϵ vorgegeben wird, um den die xy-Ebene nach oben und unten geneigt wird (*Relaxed Ground-Plane-Annahme*, siehe Abb. 3.9) [Kal05, S. 18 ff.]. Das ist insbesondere im in dieser Arbeit verwendeten Versuchsaufbau (siehe Abschnitt 1.2) nötig, da LKW im Vergleich zu PKW zu verstärkten Nickbewegungen neigen.

Damit der (relaxierte) Suchtunnel aufgebaut werden kann, sind die Parameter (Einbauposition und -winkel, Brennweite usw.) der bildgebenden Kamera nötig [BK08, S. 370 ff.]. Für alle in dieser Arbeit verwendeten Sequenzen lagen diese vor. Zusätzlich wird die erwartete Objekthöhe benötigt [Kal05, S. 19 ff.], damit die „Ausmaße“ des Suchtunnels berechnet werden können.

Die Dichte, mit der Hypothesen innerhalb des Suchtunnels generiert werden (die sog. *Quantisierung*), wird relativ zur Größe der Hypothesen angegeben. Bspw. wird bei einer Hypothesengröße von 100 Pixeln mit einer Quantisierung von 0.5 die nächste Hypothese in dieser Größe um $50 = 0.5 \times 100$ Pixel verschoben erzeugt. Sind mit dieser Quantisierung innerhalb des Suchtunnels alle Hypothesen dieser Größe erzeugt worden, wird sie skaliert und das gleiche Verfahren in der nächsten Skalierungsstufe wiederholt. Die Größe der Hypothesen der nächsten Skalierungsstufe wird, analog zur Quantisierung in x- und y-Richtung, ebenfalls relativ zur aktuellen Hypothesengröße angegeben (sog. z-Quantisierung) [Kal05, S. 21 f.].

KAPITEL 4

Hierarchische Klassifikation

In diesem Kapitel werden Klassifikationsbäume eingeführt (Abschnitt 4.1) sowie diskutiert, wie sich die Viola-Jones-Kaskade auf einen hierarchischen Klassifikator erweitern lässt (Abschnitt 4.2). Hierfür werden mögliche Ansätze kategorisiert (Abschnitt 4.2.1) und untersucht, welche Strategien des Baumdurchlaufs möglich sind (Abschnitt 4.2.2). Im Anschluss daran wird das Training und die Anwendung hierarchischer Erweiterungen des Kaskadenklassifikators erläutert (Abschnitt 4.3).

4.1. Einleitung

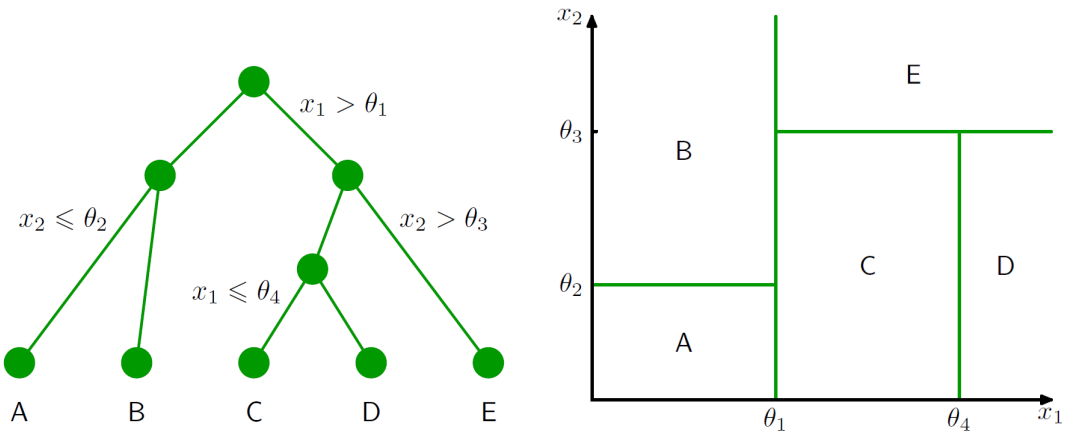
Im Folgenden wird ein kurzer Überblick über Klassifikationsbäume gegeben. Für eine weitergehende Einführung sei auf [DHS00, S. 394 ff.], [Bis06, S. 663 ff.] sowie [BK08, S. 486 ff.] verwiesen.

Unter einem hierarchischen Klassifikator oder Klassifikationsbaum wird ein Klassifikator verstanden, der durch eine Serie von hintereinander ausgeführten Entscheidungen zu einer Klasse für eine Hypothese findet. Die Antwort jeder einzelnen Entscheidung bestimmt die nächste Entscheidung. Somit ergibt sich eine Baumstruktur mit einem Knoten für jede Entscheidung. Die Wurzel des Baums entspricht der ersten Entscheidung. Diese muss für jede Hypothese getroffen werden. Den Blättern des Baums werden Klassen zugeordnet. Erreicht eine Hypothese ein Blatt, d. h. es gibt keine weiteren Entscheidungen für sie zu treffen, wird ihr die Klasse des Blattes zugewiesen. Die Zuordnung von Klassen zu Blättern ist nicht exklusiv oder eineindeutig, d. h. verschiedenen Blättern können die gleiche Klasse zugeordnet sein; außerdem ist eine Zuordnung von mehreren Klassen zu einem einzigen Blatt möglich.

In dieser Arbeit sind die zu treffenden Entscheidungen ausschließlich binär. Somit ist die entstehende Struktur ein Binärbaum. Da sich jede n-äre Entscheidung als Serie von binären Entscheidungen darstellen lässt [ASS01], ist durch diese Einschränkung kein Verlust der Allgemeinheit gegeben.

In jedem Knoten wird der Eingaberaum in zwei Teile geteilt. Eine Abfolge von Entscheidungen entspricht somit einer sich verfeinernden, hierarchischen Zerlegung des Eingaberaums. Das

Paradigma hinter dieser Herangehensweise ist *Divide-and-Conquer*, also die Annahme, dass die Klassifikationsaufgabe durch sukzessive Teilung vereinfacht wird [Wu08, S. 16]. Abb. 4.1 zeigt ein Beispiel für einen Klassifikationsbaum sowie die zugehörige Zerlegung des Eingaberaums.



(a) (b)

Abbildung 4.1.: Beispiel eines Klassifikationsbaums für zweidimensionale Eingaben. (a) Eine Eingabe $\underline{x} = (x_1, x_2)^T$ wird durch eine Abfolge von binären Entscheidungen klassifiziert. Den Blättern sind Klassen (in diesem Beispiel *A*–*E*) zugeordnet. (b) Ein Klassifikationsbaum entspricht einer Zerlegung des Eingaberaums. In diesem Beispiel erfolgt diese durch die Schwellwerte θ_{1-4}.
Quelle: [Bis06, S. 663 f.]

Die Struktur eines Klassifikationsbaums, d. h. die Reihenfolge der Entscheidungen, wird gewöhnlich aus den vorhandenen Trainingsbeispielen ermittelt. Hierfür gibt es verschiedene Verfahren. Beispiele sind der von Breiman et al. entwickelte CART-Algorithmus [BFSO84] sowie die von Quinlan entwickelten Verfahren ID3 [Qui86] und C4.5 [Qui93]. Alle Verfahren haben gemein, dass sie durch eine geschickte Wahl der Entscheidungen (z. B. durch Auswahl einer oder mehrerer der Eingabevariablen, anhand derer der Eingaberaum geteilt wird) die *Unreinheit* (engl. *impurity*) der entstehenden Teilmengen minimieren und auf diese Weise die Anzahl der Entscheidungen kleinstmöglich zu halten versuchen (sog. Regel von *Occam's razor* [DHS00, S. 398]).

Ein mögliches und weitverbreitetes Maß für die Unreinheit einer Menge ist die Shannon'sche[1] Entropie. Sie ist definiert als

$$i(N) = -\sum_j P(c_j) \log P(c_j) \,,$$

wobei $P(c_j)$ der Anteil an Trainingsbeispielen der Klasse c_j ist. Mögliche Alternativen sind u. a. der *Gini*-Index oder das Risiko einer Fehlklassifikation [DHS00, S. 398].

Wird die Zerlegung des Eingaberaums so lange fortgesetzt, bis die Unreinheit in allen Knoten so klein wie möglich ist, ist der resultierende Klassifikationsbaum typischerweise überadaptiert [DHS00, S. 402]. Als Extremfall kann in jedem Blatt lediglich eines der Trainingsbeispiele liegen.

[1] C. Shannon (* 12. April 1916; † 24. Februar 2001) gilt als der Begründer der Informationstheorie. Außerdem ist er durch zahlreiche Erfindungen wie z. B. eine Jongliermaschine bekannt.

Zahlreiche Strategien existieren, um dieses Problem zu umgehen. Sie lassen sich einteilen in Strategien, die weitere Teilungen des Eingaberaums verhindern (*Pre-Pruning*) und Strategien, die komplett gewachsene (d. h. mit hoher Wahrscheinlichkeit überadaptierte) Bäume wieder komprimieren, indem sie Knoten zur Wurzel hin verschmelzen (*Post-Pruning*).

Im Folgenden wird die Viola-Jones-Kaskade im Kontext von hierarchischen Klassifikatoren betrachtet. Es wird diskutiert, welche Möglichkeiten einer Erweiterung auf einen Klassifikationsbaum sich prinzipiell ergeben. Weiter wird ein Überblick über bereits vorgeschlagene hierarchische Erweiterungen der Kaskade gegeben.

4.2. Erweiterung der Kaskade auf einen hierarchischen Klassifikator

Die Viola-Jones-Kaskade kann als Spezialfall eines Klassifikationsbaums gesehen werden [VJ01]. In jeder Stufe wird hierbei entschieden, ob eine Hypothese Hintergrund zeigt. Ist das der Fall, wird sie einem impliziten Blatt an dieser Stufe zugeordnet und verworfen. Zeigt die Hypothese allerdings ein Objekt, so wird ihre Bearbeitung mit der nächsten Entscheidung, d. h. in der nächsten Stufe, fortgesetzt. Die Kaskadenstruktur entspricht somit einem linearisierten, binären Klassifikationsbaum.

Dieser Aufbau hat allerdings Nachteile, sobald die Positivbeispiele sich nicht effizient vom Hintergrund trennen lassen [ZZ10],[PSR10]. Durch die Konvergenz von AdaBoost (siehe Abschnitt 3.1.2) ist ein Erreichen der Trainingsvorgaben zwar immer möglich, allerdings werden dafür u. U. sehr viele Merkmale benötigt. Dieser Fall tritt z. B. dann ein, wenn die Positivbeispiele multimodal verteilt sind [Wen03, S. 56 ff.]. Um solche Probleme zu vermeiden, bietet sich eine weitergehende Teilung des Eingaberaums, also die Erweiterung der Kaskadenstruktur auf einen hierarchischen Klassifikator, an.

Diese Erweiterung darf allerdings nicht die Vorteile der Kaskadenstruktur außer Kraft setzen. Sie muss die *Reject-Fast*-Eigenschaft (das schnellstmögliche Zurückweisen von Hintergrund) beibehalten, um die für automobile Assistenzsysteme benötigte Geschwindigkeit zu erreichen. Folglich sind die „klassischen“ Ansätze zur hierarchischen Klassifikation (CART, ID3, C4.5 etc.) nicht geeignet, da bei diesen die Klasse einer Hypothese erst nach dem Durchlaufen des gesamten Baums feststeht. Sie ignorieren die asymmetrische Fragestellung des Problems Objekte in Bildern zu detektieren.

Die Kaskade teilt den Eingaberaum lediglich, um positive von negativen Beispielen zu trennen. Durch eine zusätzliche Unterteilung der in einer Stufe akzeptierten Hypothesen (*Divide*) kann eine schnellere Konvergenz von AdaBoost in den folgenden Stufen erreicht werden (*Conquer*) [WL04]. Somit werden weniger Boosting-Runden bis zum Erreichen der Trainingsvorgaben benötigt, was in höherer Geschwindigkeit des erstellten Klassifikators resultiert[2]. Zusätzlich zur potentiell höheren Geschwindigkeit ist eine höhere Detektionsrate erreichbar [WN07]. Dies erklärt sich dadurch, dass die Detektionsraten entlang des Pfades einer Hypothese multipliziert werden (siehe auch Abschnitt 3.4.1). Durch die schnellere Konvergenz von AdaBoost können in den einzelnen Stufen bei gleicher Anzahl Merkmale schwieriger zu erfüllende Vorgaben gemacht werden. D. h. bei gleicher Detektionsrate ist eine geringere Falschalarmrate

[2] Die Anzahl benötigter Boosting-Runden entspricht der Anzahl zum Einsatz kommender Merkmale. Bei weniger Boosting-Runden benötigt der finale Klassifikator folglich weniger Merkmale für die Klassifikation einer Hypothese.

erzielbar. Folglich werden weniger Stufen benötigt, bis keine Negativbeispiele mehr zu finden sind.

4.2.1. Kategorisierung möglicher Erweiterungen

Hierarchische Erweiterungen der Kaskade lassen sich durch zwei Kriterien kategorisieren:

Baumaufbaukriterium: Die Struktur des Baums kann fest vorgegeben sein oder sich automatisch durch das Training entwickeln, dann erfolgt die Einflussnahme auf den Baumaufbau lediglich indirekt über die Wahl der Trainingsparameter.

Teilungskriterium: In den Knoten eines Klassifikationsbaums erfolgt eine Teilung des Eingaberaums. Diese Teilung kann anhand der möglichen Ansichten eines Objektes (bspw. ein Kindknoten für Frontansichten von PKW, ein Kindknoten für Seitansichten usw.) oder anhand der Merkmale erfolgen.

Es ergeben sich dadurch vier prinzipielle Möglichkeiten einer hierarchischen Erweiterung der Kaskade (in Abb. 4.2 graphisch veranschaulicht). Im Folgenden werden diese erläutert.

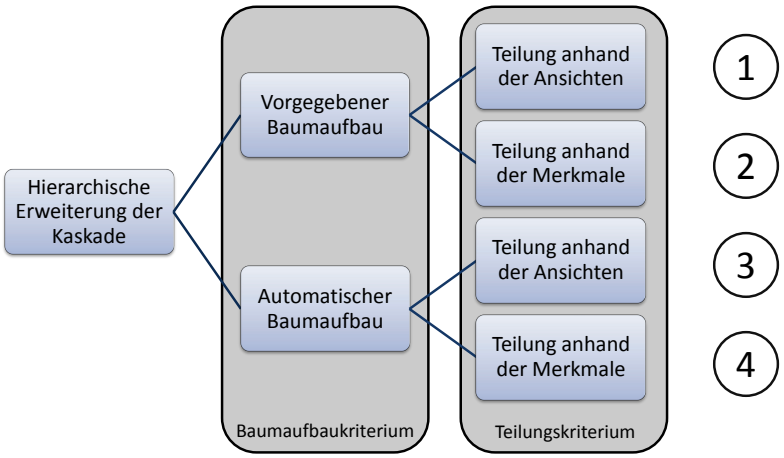


Abbildung 4.2.: Taxonomie möglicher hierarchischer Erweiterungen der Kaskade. Erweiterungen lassen sich anhand zweier Kriterien einordnen: 1. Baumaufbaukriterium und 2. Teilungskriterium. Somit ergeben sich vier prinzipielle Möglichkeiten.

Vorgegebener Baumaufbau mit einer Teilung anhand der Ansichten

Ansätze dieser Art gehören zu den ersten Versuchen (u. a. [JV03]), die Nachteile der Kaskadenstrukur durch eine hierarchische Erweiterung auszugleichen. Sie verbinden eine Schätzung der vorliegenden Ansicht (engl. *pose-estimation*) mit ansichtsspezifischen Kaskaden oder Teilbäumen. Die Struktur des Baums ist hierbei fest vorgegeben. Beispiele sind die von Jones und Viola in [JV03] vorgeschlagene Struktur, die eine vorgelagerte Schätzung der Ansicht mit nachgelagerten Kaskaden für die Detektion von Gesichtern nutzt, sowie der von Huang et al. entwickelte *WFS-Baum* [HALL05],[HALL07], bei dem durch die vektorwertige AdaBoost-Variante *VectorBoosting* Ansichtsschätzung und Trennung der Objekte von Hintergrund in einem Schritt erfolgen.

Auch die für diese Arbeit in vielen Punkten grundlegende Arbeit von Kallenbach [Kal05] kann dieser Kategorie zugeordnet werden. Im Gegensatz zu den oben genannten Ansätzen wird in [Kal05] allerdings auf die Schätzung der Ansicht verzichtet. Hypothesen werden somit automatisch in alle nachfolgenden Teilbäume weitergereicht (siehe auch Abschnitt 4.2.2).

Vorgegebener Baumaufbau mit einer Teilung anhand der Merkmale

Als Alternative zur Teilung anhand der Ansichten kann der Eingaberaum (bei vorgegebenem Baumaufbau) auch anhand der Merkmale geteilt werden. Damit erfolgt die Spezialisierung der nachfolgenden Knoten nicht durch die Ansichten. Die nachfolgenden Knoten können sich stattdessen auf einen Teilbereich des Merkmalsraums spezialisieren.

In der Arbeit von Wender [Wen03] werden Erweiterungen dieser Art untersucht. Von einer Kaskade ausgehend, wird an bestimmten vorgegebenen Stellen während des Trainings eine Teilung des Merkmalsraums vorgenommen. Diese wird so gewählt, dass die Entropie in den entstehenden Teilmengen minimal ist.

Automatischer Baumaufbau mit einer Teilung anhand der Ansichten

Bei Ansätzen dieser Kategorie ist der Baumaufbau nicht vorgegeben, d. h. in welchen der Knoten eine Auftrennung erfolgt ist nicht von vornherein klar. Wird in einem Knoten eine Teilung vorgenommen, so wird diese anhand der Ansichten der Objekte durchgeführt. Beispiele sind die von Fleuret und Geman vorgeschlagene Struktur [FG01] der hierarchischen Zerlegung von Profilen für die Gesichtserkennung (*Coarse-to-Fine Face Detection*) sowie der von Kuo und Nevatia vorgeschlagene Ansatz [KN09], der hierarchisches Clustering in Verbindung mit nichtlinearer Dimensionsreduktion nutzt, um PKW-Ansichten zu kategorisieren.

Leider ließen sich die Ergebnisse von Kuo und Nevatia auf dem verwendeten Datensatz (siehe Abschnitt 1.5) nicht nachvollziehen. Das von ihnen vorgestellte Verfahren zeigte zwar eine Struktur in den Daten (siehe Abb. 4.3), diese war jedoch (anders als von den Autoren in [KN09] beschrieben) stark von den Parametern der Dimensionsreduktion abhängig.

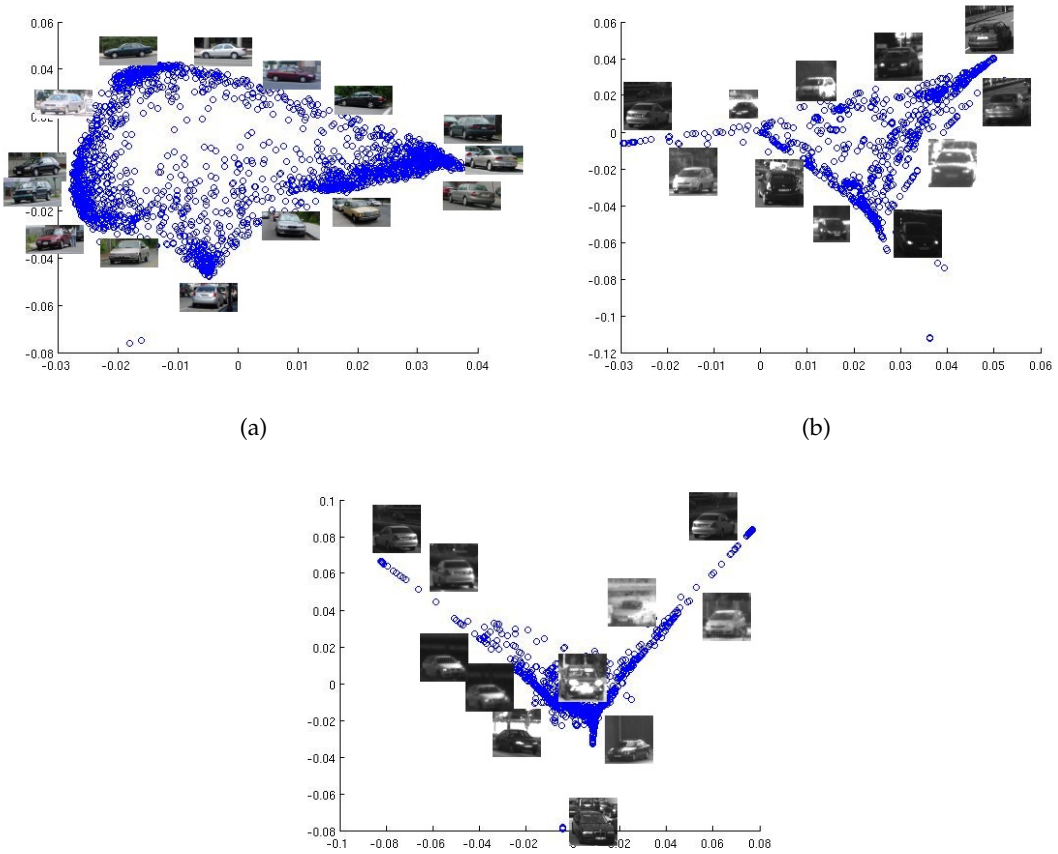
Automatischer Baumaufbau mit einer Teilung anhand der Merkmale

Bei Erweiterungen dieser Kategorie ist die Struktur des Baums ebenfalls nicht vorgegeben, ergibt sich allerdings im Gegensatz zu den ansichtsbasierten Ansätzen indem der Merkmalsraum geteilt wird. Somit ist es möglich, dass der Baumaufbau vorhandene Ansichten komplett ignoriert. Daher lässt sich keine klare Zuordnung von Blättern zu Ansichten mehr vornehmen.

Der CART-Algorithmus [BFSO84] ist ein Beispiel für einen Ansatz dieser Kategorie. In jedem Knoten wird, unabhängig von möglichen Ansichten, eines der Merkmale ausgewählt, anhand dem der Merkmalsraum geteilt wird. Andere Beispiele sind der von Wu und Nevatia entwickelte *Cluster Boosted Tree* (CBT) [WN07] und die von Yang et al. weiterentwickelte Variante *Voting Cluster Boosted Tree* (VCBT) [YHN09].

4.2.2. Traversierungsstrategien möglicher Erweiterungen

In der Phase der Anwendung durchläuft eine Hypothese den Baum top-down, d. h. von der Wurzel an. Hat ein Knoten (neben der immer vorhandenen Möglichkeit, die Hypothese zurückzuweisen) nur einen Nachfolger ist keine Entscheidung nötig, an welche Knoten die



(a) (b)

(c)

Abbildung 4.3.: Ansatz von Kuo und Nevatia [KN09] zur Kategorisierung von Fahrzeugansichten. Der Ansatz zeigt eine den Fahrzeugansichten inhärente Struktur, ist jedoch nicht sehr robust gegenüber Parameteränderungen. (a) Struktur bei Anwendung auf den Datensatz aus [KN09]. (b), (c) Anwendung des Verfahrens auf den Datensatz dieser Arbeit. Der Unterschied in den Bildern kommt durch eine leichte Variation eines der Eingabeparameter (Dimension der HOG-Merkmale) zustande.

Hypothese weitergereicht wird. Existieren dagegen mehrere Nachfolger, so ergeben sich drei mögliche Strategien, die Hypothese weiterzuverarbeiten.

Die erste mögliche Strategie ist der Verzicht auf eine Entscheidung. Eine in einem Knoten akzeptierte Hypothese wird folglich immer an beide Nachfolger weitergereicht. Die Trainingsbeispiele der einzelnen Knoten werden je nach Aufgabenstellung des Klassifikationsproblems gewählt. Ist eine Unterscheidung zwischen den einzelnen Objektkategorien – d. h. zwischen den einzelnen Zweigen – wichtig, werden Positivbeispiele des einen Zweiges als Negativbeispiele für den anderen Zweig verwendet. Somit wird – zusätzlich zur Trennung zwischen Hintergrund und Objekt – eine Trennung zwischen den einzelnen Objektkategorien vorgenommen. In der Arbeit von Kallenbach [Kal05] wurde auf diese Weise ein multiklassenfähiger Kaskadenklassifi-

kator konstruiert. Ist eine Unterscheidung zwischen den einzelnen Zweigen nicht nötig, kann allerdings auf diese zusätzlichen Negativbeispiele verzichtet werden [WN07].

Der Verzicht auf eine Entscheidung hat allerdings wesentliche Nachteile. Zum einen ist die Geschwindigkeit eines so konstruierten Klassifikators wegen der höheren Anzahl Knoten, die eine Hypothese durchlaufen muss, tendenziell geringer als bei den anderen Strategien. Im schlimmsten Fall erreicht die Hypothese jedes vorhandene Blatt. Bei einem vollständigen Binärbaum der Tiefe t entspricht dies $2^t - 1$ besuchten Knoten. Zum anderen gehen die Falschalarme der einzelnen Zweige bei einem Weiterreichen der Hypothese in beide Nachfolger additiv in die Falschalarme des gesamten Baums ein (siehe auch Abschnitt 4.3). Der Klassifikationsbaum benötigt damit eine im Vergleich zu den anderen Strategien geringere Falschalarmrate in den möglichen Pfaden von der Wurzel zu den Blättern, um insgesamt die gleiche Falschalarmrate zu erreichen.

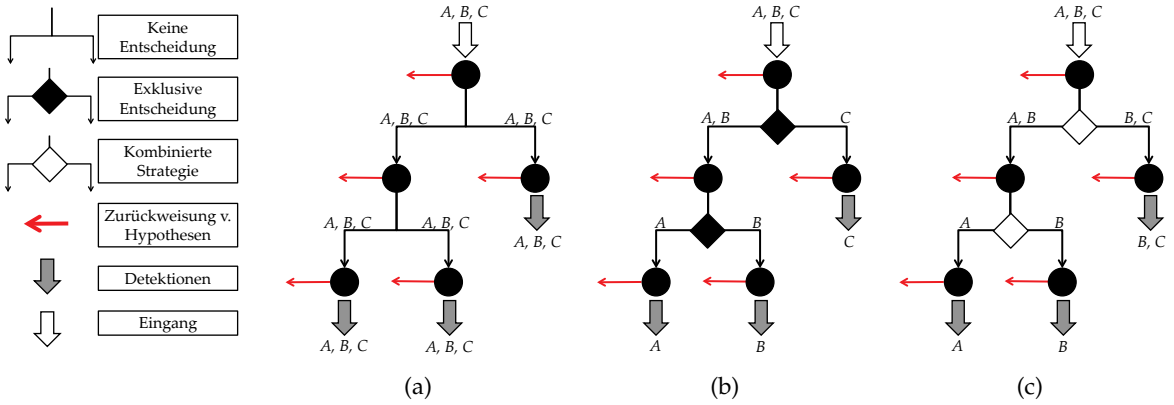


Abbildung 4.4.: Mögliche Entscheidungsstrategien in den Knoten. Es existieren verschiedene Strategien, wie Hypothesen an Kindknoten weitergereicht werden können. Dies wird hier anhand von drei Positivbeispielen A, B und C verdeutlicht. Die immer vorhandene Möglichkeit des Zurückweisens einer Hypothese ist rot dargestellt. (a) Weiterreichen an alle Nachfolger. (b) Treffen einer exklusiven Entscheidung. Eine Hypothese wird an genau einen der Nachfolger weitergereicht. (c) Kombinierte Strategie. Lässt sich eine Hypothese klar einem der Nachfolger zuordnen, so wird eine exklusive Entscheidung getroffen, andernfalls wird sie an beide Nachfolger weitergereicht.

Diese Nachteile werden durch das Treffen einer exklusiven Entscheidung, an welchen Knoten eine Hypothese weitergereicht werden soll, vermieden. Eine Hypothese wird somit nur an maximal einen der Nachfolger weitergereicht. Daher entspricht die maximale Anzahl zu besuchender Knoten dem längsten vorkommenden Pfad von der Wurzel zu einem Blatt. Die Falschalarme der einzelnen Pfade gehen nicht additiv in die gesamten Falschalarme ein.

Bei dieser Strategie ergeben sich allerdings Probleme, sobald in einem Knoten eine falsche Entscheidung getroffen wird [ZZ09],[PSR10]. Diese kann nicht mehr korrigiert werden. Ist eine Unterscheidung zwischen den einzelnen Objektkategorien bedeutsam, kann dieser Effekt sehr unerwünscht sein [Kal05, S. 54 f.] und sich zudem negativ auf die Performance des Klassifikators auswirken [HALL07]. Zusätzlich kann bei ungünstiger Wahl der Entscheidung die Klassifikationsaufgabe (Trennung von Hintergrund und Objekt) in den beiden entstehenden

Teilmengen schwieriger sein als ohne eine Entscheidung [Chr07, S. 30 f.]. Somit werden in den einzelnen Teilbäumen mehr Merkmale nötig, was den potentiellen Geschwindigkeitsvorteil gegenüber einer gemeinsamen Verarbeitung wieder egalisiert.

Die dritte mögliche Strategie ist eine Kombination aus den beiden oben genannten Strategien. Es wird versucht, eine exklusive Entscheidung zu treffen. Ist eine Zuordnung der Hypothese zu den Nachfolgern nicht eindeutig möglich, wird die Hypothese an beide Nachfolger weitergereicht. Auf diese Weise kommen die Vorteile beider Strategien zum tragen. Nachteilig wirkt sich allerdings auch bei dieser Strategie die potentiell größere Anzahl Knoten, die besucht werden müssen, aus [HALL07].

Abb. 4.4 zeigt die drei möglichen Entscheidungsstrategien anhand von drei Positivbeispielen *A*, *B* und *C*.

4.3. Training und Anwendung hierarchischer Erweiterungen

In diesem Abschnitt wird der generelle Trainingsablauf beim Training hierarchischer Erweiterungen erläutert. Diese Grundstruktur ist für alle vorgestellten Erweiterungen gleich. Im Anschluss wird die Anwendung hierarchischer Erweiterungen diskutiert. Hierbei ist v. a. von Bedeutung, wann eine Hypothese als Objekt und wann als Hintergrund gewertet wird. Abhängig von der Entscheidungsstrategie ergeben sich Unterschiede, die in diesem Abschnitt aufgezeigt werden.

Analog zum Training der Kaskade, bei der die einzelnen Stufen nur mit Beispielen trainiert werden, die von allen vorherigen Stufen akzeptiert wurden (vgl. Abschnitt 3.4.1), dürfen die Knoten eines Klassifikationsbaums nur mit solchen Beispielen trainiert werden, die sie auch erreichen. D. h. es dürfen nur solche Beispiele verwendet werden, die von allen Knoten des Pfades von der Wurzel bis zum zu trainierenden Knoten akzeptiert werden. Hierbei spielt die Entscheidungsstrategie des Klassifikationsbaums (siehe Abschnitt 4.2.2) eine entscheidende Rolle.

Ist die Struktur des Baums nicht vorgegeben, wird diese während des Trainings ermittelt. Damit erfüllt das Training eine Doppelfunktion. Zum einen wird in den Knoten des Baums die Unterscheidung zwischen Hintergrund und Objekten trainiert, zum anderen wird durch das Training die Struktur des Baums festgelegt. Dies erfolgt von der Wurzel an. In jedem Knoten wird ermittelt, ob weitere Nachfolger benötigt werden. Ist das der Fall (etwa weil noch Negativbeispiele gefunden wurden), so werden neue Knoten als Kinder des aktuellen Knotens zum finalen Klassifikationsbaum hinzugefügt. Die Anzahl der Nachfolger wird durch den Algorithmus selbst bestimmt. Wird nur ein Nachfolger hinzugefügt, so wird keine Teilung des Eingaberaums vorgenommen. Das Training einer Kaskade kann also als Spezialfall eines Baumtrainings mit der Beschränkung auf maximal einen Nachfolger je Knoten gesehen werden. Abb. 4.5 zeigt den Ablauf des Baumaufbaus graphisch.

Wie auch das Training einer Kaskade, kann das Training eines Klassifikationsbaums so lange fortgesetzt werden, bis keine Negativbeispiele mehr zu finden sind. Alternativ kann eine maximale Tiefe des Baums vorgegeben werden. Zusätzlich ist eine Begrenzung der Anzahl Blätter möglich, z. B. um eine Überadaption zu vermeiden [WN07].

In der Phase der Anwendung werden Hypothesen gemäß seiner Entscheidungsstrategie durch den Baum geleitet. Sobald ein Knoten eine Hypothese zurückweist, wird ihre Bearbeitung in diesem Knoten abgebrochen. Sind die Entscheidungen in den Knoten nicht-exklusiv,

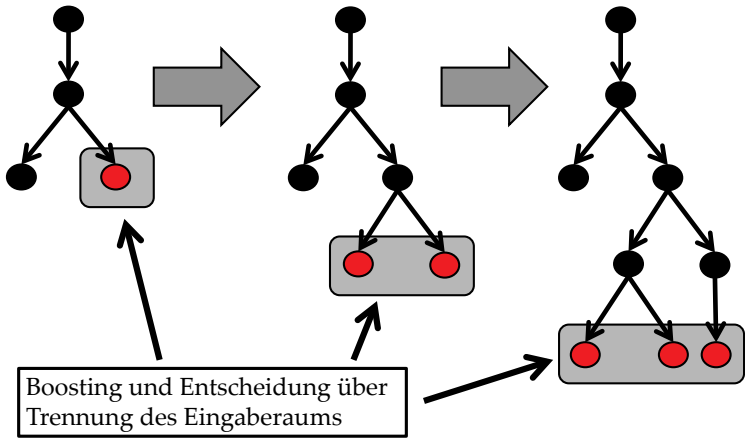


Abbildung 4.5.: Trainingsablauf eines automatisch aufgebauten Klassifikationsbaums. Der Klassifikationsbaum wird von der Wurzel an trainiert. In jedem Knoten wird ein Boosting durchgeführt, mit dem die Trennung zwischen Hintergrund und Objekt gelernt wird. Zusätzlich wird über die Auftrennung des Eingaberaums, also die Anzahl der Nachfolger, entschieden.

bedeutet dies allerdings nicht die Zurückweisung der Hypothese. Eine Hypothese ist erst dann zurückgewiesen, wenn sie in *allen* Knoten, an die sie weitergereicht wurde, zurückgewiesen wird. Dementsprechend wird eine Hypothese als Objekt klassifiziert, sobald sie von mindestens einem Blatt akzeptiert wird.

KAPITEL 5

Decision-Boosted Cluster Boosted Tree zur Klassifikation

In diesem Kapitel wird der entwickelte Baumklassifikator (*Decision-Boosted Cluster Boosted Tree*) vorgestellt. Zuerst wird ein Überblick gegeben (Abschnitt 5.1). In Anschluss daran wird das zum Einsatz kommende Kriterium für die Auftrennung des Eingaberaums eingeführt (Abschnitt 5.2), das entwickelte Verfahren zum Finden von vorteilhaften Auftrennungen beschrieben (Abschnitt 5.3) sowie das Lernen der gefundenen Auftrennung erläutert (Abschnitt 5.4). Den Abschluss bildet eine Beschreibung von Training und Anwendung des Klassifikators (Abschnitt 5.5).

5.1. Überblick

Der in dieser Arbeit entwickelte *Decision-Boosted Cluster Boosted Tree* (DB-CBT) basiert auf Ansätzen [WN07],[YHN09] der vierten Kategorie (automatischer Baumaufbau mit einer Teilung anhand der Merkmale) der eingeführten Kategorisierung möglicher Erweiterungen (vgl. Abschnitt 4.2). Ansätze dieser Kategorie weisen eine höhere Performance auf als die Vorschläge anderer Kategorien [WN07],[YHN09],[PSR10]. Dies lässt sich damit erklären, dass Ansätze dieser Kategorie die maßgebenden Faktoren (Merkmale und zu lösende Teilprobleme) sehr viel stärker berücksichtigt als andere Ansätze dies tun [YHN09].

Der Aufbau des DB-CBT erfolgt top-down, d. h. das Wachstum des Baums erfolgt von der Wurzel aus (siehe auch Abschnitt 4.3). Während des Trainings der Knoten des Baums wird entschieden, ob eine Auftrennung des Eingaberaums erfolgen soll, oder ob eine gemeinsame Weiterverarbeitung des gesamten Eingaberaums sinnvoller ist. In den ersteren Fällen hat der neu zum finalen Klassifikator hinzugefügte Knoten zwei Nachfolger, andernfalls bleibt die Kaskadenstruktur erhalten. Dadurch ist sichergestellt, dass eine Teilung des Eingaberaums nur dann erfolgt, wenn sie nötig ist. Die Entscheidung, wann eine Auftrennung erfolgt, wird mit einem in Abschnitt 5.2 vorgestellten Kriterium getroffen.

Die Auftrennung des Eingaberaums hat das Ziel, die Klassifikationsaufgabe in den dadurch entstehenden Teilbereichen des Eingaberaums zu erleichtern. Diese Vorgabe ist nicht

bei jeder der möglichen Auftrennungen erfüllt. Ist die Zerlegung ungünstig gewählt, so kann das Klassifikationsproblem durch die Auftrennung schwieriger zu lösen sein als ohne sie [Chr07, S. 30 f.]. Die Zerlegung ist also von entscheidender Bedeutung für die Performance des resultierenden Klassifikationsbaums. Wu und Nevatia schlagen vor, die Zerlegung durch den k-Means-Algorithmus auf den Merkmalsvektoren zu finden [WN07]. Die Einteilung in die Teilmengen erfolgt somit durch die Ähnlichkeit der Merkmalsvektoren untereinander. Yang et al. dagegen wählen die Auftrennung so, dass sie die Klassifikationsaufgabe vereinfacht [YHN09].

In dieser Arbeit wird die Zerlegung, analog zu [YHN09], ebenfalls so gewählt, dass die Klassifikationsaufgabe durch sie vereinfacht wird. Im Unterschied zu [YHN09] wird diese allerdings nicht durch Würfeln und anschließendes Testen gefunden. Stattdessen wird die Zerlegung durch Lösung eines Optimierungsproblems ermittelt. Hierfür wird in Abschnitt 5.3.1 ein Gütemaß definiert, das (orientiert an Annahmen von Yang et al. [YHN09]) die Eignung von Zerlegungen angibt. Dieses Gütemaß wird genutzt, um eine vorteilhafte Zerlegung zu finden (Abschnitt 5.3.2).

In den einzelnen Knoten muss eine Entscheidung über das weitere Vorgehen getroffen werden, um eine hohe Geschwindigkeit in der Phase der Anwendung zu gewährleisten (vgl. Abschnitt 4.2.2). Die Entscheidung muss die im Training gewonnene Zerlegung widerspiegeln, d. h. die – bisher unbekannte – Hypothese muss an denjenigen Teilbaum weitergereicht werden, der für sie trainiert wurde. Daher wird nach dem eigentlichen Training zur Lösung der Klassifikationsaufgabe, d. h. der Trennung von Objekt und Hintergrund, ein weiteres Boosting, das sog. *Decision-Boosting* (DB, siehe Abschnitt 5.4), durchgeführt. Mit diesem wird die Aufteilung der Positivbeispiele gelernt. Die Merkmale sind hierbei die gleichen wie beim Lernen der Unterscheidung zwischen Objekt und Hintergrund. D. h. durch das DB werden abermals aus der Menge aller Merkmale die aussagekräftigsten für die Trennung des Eingaberaums ausgewählt. Mit diesen ist in der Phase der Anwendung das Treffen einer Entscheidung über den weiteren Weg einer Hypothese möglich.

5.2. Kriterium für die Durchführung einer Trennung

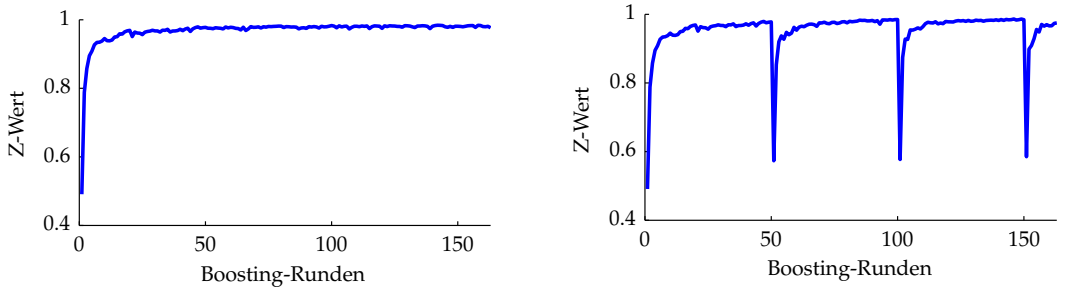
Der Aufbau des DB-CBT erfolgt automatisch, daher muss automatisch bestimmbar sein, wann eine Auftrennung des Eingaberaums erfolgen muss. Dies sollte nur geschehen, wenn eine Auftrennung die Lösung der Klassifikationsaufgabe vereinfacht. Ist dies nicht der Fall, so kann die Kaskadenstruktur beibehalten werden (Occam's razor, vgl. Abschnitt 4.1). Daher wird ein Kriterium benötigt, das zeigt, wie schwierig die Lösung des Klassifikationsproblems in einem Knoten ist. Anhand diesem Kriterium kann über die Auftrennung des Eingaberaums entschieden werden: Ist das Klassifikationsproblem schwierig, so werden sehr viele Merkmale für die Lösung benötigt (vgl. Abschnitt 4.2). Es ist in diesen Fällen also sinnvoll, eine Auftrennung vorzunehmen [PSR10].

Wie bereits in Abschnitt 3.1.3 diskutiert, wird bei RealAdaBoost in jeder Runde der Weaklearner mit dem kleinsten Z-Wert

$$Z = 2 \sum_i \sqrt{W_i^+ W_i^-} \ , \ Z \in [0, 1] \tag{5.1}$$

ausgewählt. Der Z-Wert kann interpretiert werden als die Fähigkeit des Merkmals, zwischen Hintergrund und Objekt zu trennen [WN07] und ist verbunden mit der Wahrscheinlichkeit

einer Fehlklassifikation (vgl. Abschnitt 5.2.1). Im Verlauf des Boostings nimmt der Z-Wert der gewählten Merkmale immer weiter zu (siehe Abb. 5.1). Der Beitrag, den neue Merkmale zum Stronglearner leisten, wird also immer geringer. Dies lässt sich damit erklären, dass das Gewicht der schwierigen Beispiele im Verlauf der Runden immer weiter steigt. Folglich ist nach einigen Runden der Z-Wert ein Maß für die Schwierigkeit der Klassifikationsaufgabe. Diese bemisst sich primär an den schwierigen Teilen der zu trennenden Mengen.



(a) (b)

Abbildung 5.1.: Verlauf der Z-Werte über die Boosting-Runden. (a) Verlauf der Z-Werte beim Training einer Stufe. (b) Verlauf über mehrere Kaskadenstufen hinweg. Die Einbrüche des Z-Werts sind auf den Beginn einer neuen Stufe zurückzuführen (hier: nach jeweils 50 Runden).

Aus diesem Grund kann der Z-Wert eingesetzt werden, um zu erkennen, wann eine Trennung des Eingaberaums sinnvoll ist [WN07]. Es werden n Schwellwerte $\tau_1, \ldots, \tau_n$ für den Z-Wert vorgegeben. Aus Gründen der Einfachheit werden diese gleich gewählt, d. h. $\tau_i = \tau_j$ für $i, j = 1, \ldots, n$. Ist der Z-Wert der n letzten Merkmale größer als τ_i, wird kein weiterer Weaklearner zum finalen Stronglearner hinzugefügt. Das Boosting wird in diesem Fall abgebrochen. Im Anschluss wird eine Teilung des Eingaberaums vorgenommen (siehe Abschnitt 5.3).

Im Folgenden werden weitere Eigenschaften des Z-Werts diskutiert, um die Wahl als Indikator für die Schwierigkeit des Klassifikationsproblems weiter zu begründen (Abschnitt 5.2.1). Im Anschluss daran wird die Strategie zur Vermeidung einer Überadaption vorgestellt (Abschnitt 5.2.2).

5.2.1. Eigenschaften des Z-Werts

Der Z-Wert eines Merkmals ist eng verbunden mit dem Bhattacharyya[1]-Koeffizient (BK) [Bha43], einem Ähnlichkeitsmaß für Wahrscheinlichkeitsdichten [HAWL04].

Seien $p_a(x), p_b(x)$ Wahrscheinlichkeitsdichten einer kontinuierlichen Zufallsvariablen x. Dann ist der BK $\mathrm{bhat}(p_a, p_b)$ definiert als

$$\mathrm{bhat}(p_a, p_b) = \int \sqrt{p_a(x) p_b(x)} dx.$$

[1] A. Bhattacharyya war indischer Mathematiker und Statistiker.

Für diskrete Variablen ist er durch

$$\text{bhat}(p_a, p_b) = \sum_i \sqrt{p_a^i p_b^i} \tag{5.2}$$

berechenbar [Bha43]. Sowohl im kontinuierlichen als auch im diskreten Fall gilt $\text{bhat}(.,.) \in [0, 1]$ [Kai67].

Der Zusammenhang zwischen dem Z-Wert (Gleichung (5.1)) und dem BK (Gleichung (5.2)) ist ersichtlich. Damit lässt sich der Auswahlschritt von RealAdaBoost folgendermaßen formulieren: Wähle in jeder Runde den Weaklearner mit dem kleinsten BK (durch die Multiplikation mit einem konstanten Faktor – in diesem Fall: 2 – wird die „Position“ des Minimums nicht geändert) [HAWL04]. Die für die Berechnung von $\text{bhat}(.,.)$ nötigen Dichtefunktionen sind hierbei die bereits in Abschnitt 3.4.2 vorgestellten gewichteten a-posteriori Wahrscheinlichkeiten für Hintergrund bzw. Objekt über den Merkmalswerten.

Der BK ist eng verbunden mit dem Risiko eines Klassifikationsfehlers. Ein ein hoher (=schlechter) BK ist gleichbedeutend mit einem hohen Risiko einer Falschklassifikation [Kai67],[DSG90]. Abb. 5.2 zeigt den Zusammenhang an Beispielen von Wahrscheinlichkeitsdichten mit dem zugehörigen BK.

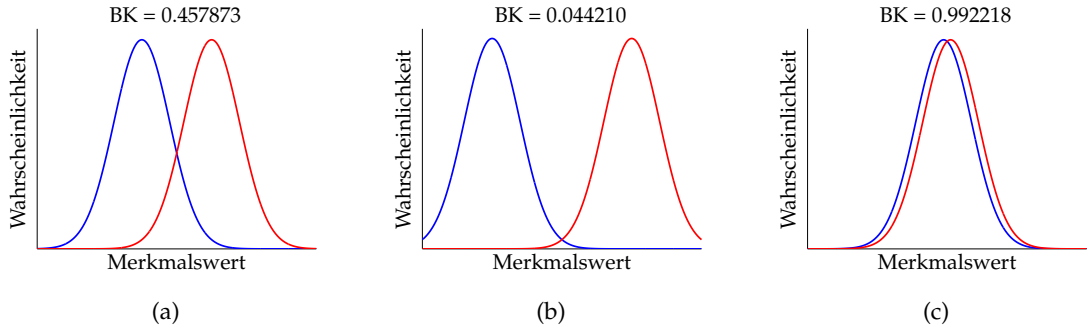


Abbildung 5.2.: Beispiele für Bhattacharyya-Koeffizienten. Der Bhattacharyya-Koeffizient (BK) ist ein Maß für die Ähnlichkeit zweier Wahrscheinlichkeitsdichten [Bha43]. Je größer der BK, desto ähnlicher sind sich beide Dichten. (a)-(c) Beispiele für Wahrscheinlichkeitsdichten und zugehörige BK. Klar zu sehen ist der Zusammenhang zwischen BK, Ähnlichkeit der Dichte und dem Risiko einer Falschklassifikation [Kai67],[DSG90].

Damit ist der Zusammenhang zwischen dem Z-Wert der gewählten Weaklearner und der Schwierigkeit des Klassifikationsproblems klar: Je höher der Z-Wert, desto höher ist das Risiko einer Falschklassifikation auf den *gewichteten* Beispielen des zugehörigen Weaklearners. In den ersten Boosting-Runden ist der Z-Wert noch klein, da sowohl die leicht als auch die schwierig zu trennenden Beispiele gleich gewichtet sind. Die Gewichte verschieben sich aber im Verlauf des Boostings immer weiter zu den schwierigen Teilen (vgl. Abschnitt 3.1). Daher ist der Z-Wert ein Maß für das Risiko einer Falschklassifikation auf den schwierigen Teilen. Folglich ist es sinnvoll, die Schwierigkeit des Klassifikationsproblems an den Z-Werten zu bemessen. Die Schwierigkeit eines Klassifikationsproblems bemisst sich nicht an den leichten sondern an den schwierigen Teilen der zu trennenden Mengen.

5.2.2. Verhinderung der Überadaption

Bäume zeigen eine Tendenz zur Überadaption (vgl. Abschnitt 4.1). Um dieses Problem zu vermeiden, sind Trennungen nur bei einer ausreichend großen Menge an Positivbeispielen in einem Knoten möglich. D. h. für das Training wird eine Schwelle η vorgegeben, anhand derer entschieden wird, ob ein Abbruch des Boosting mit anschließendem Auftrennen des Eingaberaums erlaubt ist. Ist die Anzahl der Positivbeispiele kleiner als η, wird das Boosting bis zum Erreichen der Trainingsvorgaben fortgesetzt. Es findet kein vorzeitiger Abbruch statt. Der resultierende Knoten hat nur einen Nachfolger. Folglich entsteht ab diesem Zeitpunkt durch das weitere Training eine Kaskade unterhalb des Knotens. Die Anzahl der Positivbeispiele kann im weiteren Verlauf nur abnehmen.

Dieses Vorgehen hat mehrere Vorteile: Durch die Vorgabe von η ist garantiert, dass ein gewisser repräsentativer Anteil der insgesamt vorhandenen Positivbeispiele als atomar betrachtet wird. Die Gefahr des Auswendiglernens der Trainingsbeispiele ist folglich gemindert; das Problem der Überadaption wird vermieden. Hinsichtlich der Strategien zur Überadaptionsvermeidung ist dieses Vorgehen dem Pre-Pruning zuzuordnen (vgl. 4.1).

Außerdem wird ein wiederholtes Abbrechen des Boosting durch sehr schwierig zu erkennende Objekte ab einer gewissen Stufe verhindert. Für schwierige Probleme wird das Boosting stets nach wenigen Runden abgebrochen, weshalb in den einzelnen Knoten nur ein geringer Teil des Hintergrunds zurückgewiesen wird. Effektiv wird in diesen Fällen die Lösung des Klassifikationsproblems immer weiter auf die nachfolgenden Knoten verlagert. Durch die Vorgabe von η wird verhindert, dass dies zu oft geschieht. Stattdessen wird in diesen Fällen das Problem zuerst zerlegt und anschließend – nach dem Unterschreiten von η – gelöst.

5.3. Zerlegung des Eingaberaums durch Optimierung bzgl. der Klassifikationsaufgabe

Die Zerlegungen des Eingaberaums werden so gewählt, dass sie die Klassifikationsaufgabe vereinfachen. Auf diese Weise wird die Anzahl der nötigen Merkmale bis zur endgültigen Klassifikation einer Hypothese möglichst gering gehalten. Ist die Trennung von Hintergrund und Objekt in den entstehenden Teilmengen einfach, so ist durch die schnellere Konvergenz des Boostings die Anzahl der Weaklearner geringer (vgl. Abschnitt 4.2).

Für diesen Zweck wird ein Gütekriterium definiert (Abschnitt 5.3.1), das die Eignung einer Zerlegung angibt. Anhand diesem wird mittels *Simulated Annealing* [DHS00, S. 354 ff.] eine Zerlegung gesucht, die die Klassifikationsaufgabe in den nachfolgenden Knoten vereinfacht (Abschnitt 5.3.2).

5.3.1. Gütekriterium für eine Zerlegung

Um die Zerlegung so zu wählen, dass die Klassifikationsaufgabe in den beiden entstehenden Teilmengen möglichst einfach wird, ist ein Gütekriterium für Zerlegungen nötig. Anhand diesem kann eine Optimierung der Zerlegung durchgeführt werden.

Für potentielle Gütekriterien müssen folgende Anforderungen erfüllt sein:

1. Sie müssen möglichst aussagekräftig sein, d. h. bei einer hohen Zerlegungsgüte muss die Trennung von Objekt und Hintergrund in den entstehenden Teilmengen tatsächlich einfacher sein als bei einer Zerlegung mit geringerer Güte.

2. Sie müssen sich effizient berechnen lassen, um eine Optimierung (siehe Abschnitt 5.3.2) zu ermöglichen.

Sei z eine beliebige Zerlegung. Sie zerlegt die Menge der Positivbeispiele P in zwei Teile P_A^z und P_B^z, wobei $P = P_A^z \cup P_B^z$ und $P_A^z \cap P_B^z = \emptyset$. Sei weiter N die Menge der Negativbeispiele und $\text{sep}(A, B) \in [0, 1]$ eine (beliebige) *Separierbarkeitsfunktion*. Diese gibt an, wie gut sich die Mengen A und B trennen lassen. Per definitionem sei die Trennbarkeit zweier Mengen umso besser, je größer $\text{sep}(.,.)$ ist. Dann ist bei geeigneter Wahl von $\text{sep}(.,.)$ eine Gütefunktion $\text{e}(.)$, die die obigen Bedingungen erfüllt, gegeben durch

$$\text{e}(z) = -\text{sep}(P_A^z, N) - \text{sep}(P_B^z, N) + \lambda(z),\ \text{e}(z) \in \left[-2, \frac{1}{4}\right]$$

mit

$$\lambda(z) = \left(\max\left\{\frac{|P_A^z|}{|P|}, \frac{|P_B^z|}{|P|}\right\} - \frac{1}{2}\right)^2,\ \lambda(z) \in \left[0, \frac{1}{4}\right].$$

$\lambda(z)$ ist hierbei ein Regularisierungsterm, der unbalancierte Zerlegungen bestraft. Ohne Beschränkung der Allgemeinheit ist also die Güte einer Zerlegung z umso größer, je kleiner $\text{e}(z)$ ist.

$\text{e}(.)$ bezieht durch $\text{sep}(.,.)$ die potentielle Vereinfachung der Klassifikationsaufgabe bei getrennter Weiterverarbeitung in die Güte einer Zerlegung mit ein. Daher werden die beiden Teilmengen P_A^z und P_B^z getrennt betrachtet. Für beide Teilmengen wird bewertet, wie gut sie sich von den Negativbeispielen trennen lassen. Da die Negativbeispiele der beiden entstehenden Knoten nicht bekannt sind, wird hierbei auf die Negativbeispiele des aktuellen Knotens zurückgegriffen. Diese Vereinfachung geht zurück auf [YHN09].

Durch Hinzunahme des Regularisierungsterms $\lambda(z)$ werden Zerlegungen in gleich große Teilmengen favorisiert. Dadurch wird zum Einen die Gefahr der Überadaption vermindert, da bei unbalancierten Zerlegungen stärker die Tendenz besteht, die kleinere der beiden Teilmengen auswendig zu lernen [Tu05]. Zum Anderen wirken sich unbalancierte Zerlegungen negativ auf die Performance des Gesamtklassifikators aus [Wu08, S. 39 ff.]. Es ist daher sinnvoll, die Balance einer Zerlegung in die Bewertung ihrer Güte miteinzubeziehen.

Die Wahl der Separierbarkeitsfunktion ist der entscheidende Faktor für die Gütefunktion. Sie muss die Trennbarkeit durch das Boosting widerspiegeln. Die Trennfähigkeit von AdaBoost hängt von der Trennfähigkeit der zur Verfügung stehenden Weaklearner ab (vgl. Abschnitt 3.1.2). Folglich muss $\text{sep}(A, B)$ genau dann groß sein, wenn die verfügbaren Weaklearner gut zwischen A und B trennen können.

Wie in Abschnitt 5.2.1 dargelegt, ist der eng mit AdaBoost zusammenhängende Bhattacharyya-Koeffizient (BK) ein Maß für die Trennfähigkeit eines Weaklearners. Daher lässt er sich nutzen, um eine Separierbarkeitsfunktion zu definieren: Je geringer der BK der Weaklearner bei der Trennung der Mengen A und B, desto leichter sind diese durch AdaBoost insgesamt zu trennen. Da nicht bekannt ist, welche der verfügbaren Weaklearner beim Training der neuen Knoten ausgewählt werden (das Boosting geschieht erst *nach* der Wahl der Zerlegung), wird die Menge der möglichen Weaklearner hierbei auf die bereits ausgewählten eingeschränkt. D. h.

die Zerlegung wird so gewählt, dass die im vorherigen Boosting gewählten Weaklearner eine hohe Trennfähigkeit besitzen.

Seien $f_1, \ldots, f_N$ die bereits gewählten Weaklearner. Die Separierbarkeitsfunktion $\mathrm{sep}(.,.)$ ist dann[2]

$$\begin{aligned}\mathrm{sep}(A,B) &= 1 - \overline{\mathrm{bhat}}(A,B) \\ &= 1 - \frac{1}{N}\sum_{k=1}^{N} \underset{f_k}{\mathrm{bhat}}(A,B) \\ &= 1 - \frac{1}{N}\sum_{k=1}^{N} \int \sqrt{p_A^k(x)p_B^k(x)}dx,\end{aligned}$$

also der durchschnittliche BK der Weaklearner $f_1, \ldots, f_N$ bei der Trennung der beiden Mengen A und B.

Um $\mathrm{sep}(.,.)$ möglichst effizient zu berechnen, werden äquidistante Histogramme zur Approximation der Integrale eingesetzt. Damit lässt sich $\mathrm{bhat}(.,.)$ schreiben als

$$\mathrm{bhat}(A,B) \approx \sum_i \sqrt{A_i B_i},$$

wobei A_i bzw. B_i der Anteil von A bzw. B ist, deren Merkmalswert des Merkmals f in den i-ten Bin des Histogramms fällt [CRM00].

Das auf diese Weise konstruierte $\mathrm{e}(.)$ erfüllt somit die oben genannten Anforderungen an eine Gütefunktion. Durch die Verwendung des BK der bereits gewählten Weaklearner wird die Zerlegung dahingehend optimiert, dass beide Teilmengen gut von den Negativbeispielen trennbar sind. Die Verwendung von $\mathrm{bhat}(.,.)$ hat zudem den Vorteil, dass damit $\mathrm{sep}(.,.)$ bei gleicher Aussagekraft effizienter berechenbar ist als bei Verwendung anderer möglicher Maße (bspw. der Kullback-Leibler-Divergenz) [Kai67].

5.3.2. Optimierung mittels Simulated Annealing

Die im vorherigen Abschnitt definierte Gütefunktion $\mathrm{e}(.)$ ermöglicht es, eine Zerlegung so zu wählen, dass sie die Klassifikationsaufgabe in den entstehenden Teilmengen möglichst einfach macht. Da die Wahl einer Zerlegung für ein beliebiges Gütemaß NP-schwer ist [YHN09], fällt auch die Minimierung von $\mathrm{e}(.)$ in die Kategorie der NP-schweren Probleme. Es ist folglich nur mit heuristischen Verfahren möglich, in akzeptabler Zeit eine Zerlegung zu finden die bzgl. $\mathrm{e}(.)$ optimal ist.

Als heuristisches Optimierungsverfahren kommt in dieser Arbeit Simulated Annealing (SA) zum Einsatz. Für eine Einführung in SA sei auf [DHS00, S. 354 ff.] verwiesen.

Für SA ist neben der Zielfunktion die Definition des Vorgehens bei der Durchführung des zufälligen Schrittes von einem Zustand in den anderen nötig. In dieser Arbeit wird hierfür wie folgt vorgegangen. Es seien P_A und P_B die beiden Teilmengen der Positivbeispiele P in einem beliebigen Zustand während des Verfahrens:

1. Generiere einen zufälligen boolschen Wert $r \in \{-1, 1\}$.

[2] Aus Gründen der Lesbarkeit wird hier $\mathrm{bhat}(A,B)$ von zwei Mengen A, B angegeben. Diese verkürzte Schreibweise meint den BK zwischen den zu A bzw. B gehörenden a-posteriori Dichten über dem Merkmalswert.

2. Ist $r = 1$, dann vertausche zwei zufällig gewählte Beispiele der Mengen P_A und P_B.

3. Ist dagegen $r = -1$, dann verschiebe ein zufälliges Beispiel von einer Menge in die andere. Generiere hierfür einen zufälligen rellen Wert $v \in [0, 1]$, um die Richtung des Verschiebens – von P_A nach P_B oder umgekehrt – festzulegen. Ist $v \leq \frac{|P_A|}{|P|}$, dann verschiebe von P_A nach P_B, andernfalls von P_B nach P_A.

Dieses Vorgehen hat den Vorteil, dass zufällige Schritte immer zu einer Zerlegung mit gleich großen Teilmengen tendieren. Die Wahrscheinlichkeit eines Verschiebens von einer großen Menge in die kleinere ist größer als die Wahrscheinlichkeit für das Verschieben in die umgekehrte Richtung. Da die Zielfunktion $\mathrm{e}(.)$ balancierte Zerlegungen besser bewertet, wird der zufällige Schritt also tendenziell in Richtung eines kleineren Werts von $\mathrm{e}(.)$ ausgeführt.

Die weiteren Parameter von SA (Minimal- und Maximaltemperatur, Kühlfaktor usw.) werden experimentell oder durch Erfahrung bestimmt. Der Startzustand des Verfahrens ist eine zufällige Zerlegung in zwei gleich große Hälften.

5.4. Lernen der Zerlegung mittels Decision-Boosting

Aus Geschwindigkeits- und Performance-Gründen ist das Treffen einer Entscheidung über den weiteren Weg einer Hypothese sinnvoll (vgl. Abschnitt 4.2.2). Daher wird nach dem Finden einer Zerlegung ein erneutes Boosting (Decision-Boosting, kurz: DB) durchgeführt. Es hat das Ziel, die gefundene Zerlegung zu lernen. In einem Knoten mit zwei Nachfolgern befinden sich folglich zwei Klassifikatoren. Der durch das erste Boosting konstruierte Klassifikator dient dazu, Hintergrund zurückzuweisen. Der zweite trifft die Entscheidung, an welchen der Nachfolger eine Hypothese weitergereicht wird, sofern der erste Klassifikator sie nicht zurückweist. Die Entscheidungen, die getroffen werden, sind hierbei exklusiv (vgl. Abschnitt 4.2.2), um eine möglichst hohe Geschwindigkeit zu gewährleisten. Die potentiell mögliche Erweiterung des DB-CBT auf eine kombinierte Entscheidungsstrategie ist nicht Teil dieser Arbeit und kann in weiterführenden Arbeiten untersucht werden.

Seien P_A und P_B die Teilmengen der Positivbeispiele P unter der gefundenen Zerlegung. Ohne Beschränkung der Allgemeinheit ist P_A die Menge der Positivbeispiele, P_B die Menge der Negativbeispiele für das DB. Die Begriffe Detektions- und Falschalarmrate sind dementsprechend zu verstehen.

Es stehen alle Merkmale als Weaklearner für das DB zur Verfügung. Somit wird durch das DB (in der Phase der Anwendung) die Entscheidung über den weiteren Weg einer Hypothese nur aufgrund von Merkmalswerten getroffen. Daher sind für den DB-CBT keine anderen atomaren Operationen als für die klassische Viola-Jones-Kaskade nötig. Diese Tatsache ist v. a. im Hinblick auf eine mögliche Implementierung in Hardware von Vorteil: Die ohnehin vorhandenen Bausteine für die hocheffiziente Auswertung von Merkmalen können auch für das Treffen von Entscheidungen herangezogen werden.

Die Vorgaben, die für das DB gemacht werden, weichen von den Vorgaben des normalen Boostings ab. Statt zu erreichende Detektions- und Falschalarmraten vorzugeben, wird für das DB die benötigte *Korrektheit* der Entscheidung festgelegt. Die Korrektheit ist der Anteil der Positivbeispiele, die gemäß der ermittelten Zerlegung an die korrekten Teilbäume weitergereicht werden. Bei repräsentativer Auswahl der Positivbeispiele entspricht die erreichte Korrektheit der Wahrscheinlichkeit einer korrekten Entscheidung in der Phase der Anwendung.

Seien $c \in [0, 1]$ die geforderte Korrektheit, D die Detektionsrate sowie F die Falschalarmrate des mittels DB konstruierten Klassifikators. Dann ist die Korrektheit c erreicht, wenn (evtl. durch Wahl eines geeigneten Schwellwerts θ, siehe Abschnitt 3.4.1)

$$D \geq c$$

und

$$F \leq 1 - c$$

erfüllt sind.

Analog zum normalen Boosting werden auch beim DB so lange Merkmale zum Stronglearner hinzugefügt, bis diese Gleichungen erfüllt sind. Es werden allerdings keine Schwellwerte für den Z-Wert vorgegeben, d. h. das DB wird nicht vorzeitig abgebrochen. Um die Anzahl der Merkmale zu begrenzen, wird eine maximale Anzahl von Runden vorgegeben.

5.5. Training und Anwendung des Decision-Boosted Cluster Boosted Tree

Das Training des DB-CBT folgt dem in Abschnitt 4.3 vorgestellten Schema, d. h. das Training erfolgt ab der Wurzel. Für jeden neu hinzugefügten Knoten werden sowohl Positiv- als auch Negativbeispiele gesammelt. Mit diesen wird im Anschluss ein Boosting durchgeführt.

Im Unterschied zum Training einer Kaskade (vgl. 3.4.1) wird das Boosting nicht in jedem Fall bis zum Erreichen der Vorgaben bzw. der maximalen Anzahl Weaklearner fortgesetzt: Es wird vorzeitig angebrochen, sobald das Klassifikationsproblem als zu schwierig erkannt wird und eine Überadaption unwahrscheinlich ist, d. h. sobald die vorgegebenen Schwellwerte $\tau_i, \ldots, \tau_n$ für den Z-Wert verletzt und noch ausreichend ($\geq \eta$) Positivbeispiele vorhanden sind. Wird das Boosting vorzeitig abgebrochen, hat der trainierte Knoten zwei Nachfolger. Andernfalls wird die Kaskadenstruktur beibehalten.

Wird das Boosting vorzeitig abgebrochen, erfolgt eine Trennung des Eingaberaums, um das Klassifikationsproblem zu vereinfachen. Zu diesem Zweck wird die Menge der Positivbeispiele in zwei Teile zerlegt (Abschnitt 5.3). Die gefundene Zerlegung wird mittels DB (Abschnitt 5.4) gelernt.

Algorithmus 5.1 stellt das Training des DB-CBT in Pseudocode dar.

Durch diesen Trainingsaufbau und die Wahl der Parameter lassen sich drei Phasen während des Trainings unterscheiden:

1. Zu Beginn ist das Klassifikationsproblem in den Knoten leicht zu lösen; die Negativbeispiele sind durch wenige Weaklearner von den Positivbeispielen trennbar. Daher wird die Kaskadenstruktur beibehalten.

2. Je mehr Stufen im Stile der Kaskade trainiert wurden, desto schwieriger wird das Klassifikationsproblem (vgl. Abschnitt 3.4.1). Daher wird ab einer bestimmten Stufe eine Trennung des Eingaberaums vorgenommen. Diese vereinfacht das Klassifikationsproblem. Der Eingaberaum wird so lange geteilt, bis das Klassifikationsproblem sehr einfach ist oder bis nicht mehr genügend Positivbeispiele vorhanden sind.

Algorithmus 5.1 Training des DB-CBT

Initialisiere: Menge der zu trainierenden Knoten $\mathcal{Q} \leftarrow \{q_0\}$ (q_0 = Wurzel des Baums)

while $\mathcal{Q} \neq \emptyset$ **do**

- Wähle den zu trainierenden Knoten $q \in \mathcal{Q}$.
- Sammle $P \cup N$ = Positiv- und Negativbeispiele für den Knoten q.

if $N \neq \emptyset$ *und* $P \neq \emptyset$ **then**

`// Deaktiviere Trennung wenn zu wenige Positivbeispiele vorhanden.`

if $|P| < \eta$ **then**

- Führe normalen *RealAdaBoost* auf den Beispielen aus, bis die Vorgaben erfüllt sind.
- Füge einen neuen Knoten q_{neu} als Nachfolger von q hinzu.
- $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_{neu}$

else

- Führe normalen *RealAdaBoost* auf den Beispielen aus, bis die Vorgaben erfüllt sind. Brich vorzeitig ab, wenn die Schwellwerte $\tau_1, \ldots, \tau_n$ überschritten werden.

if *Boosting wurde vorzeitig abgebrochen* **then**

- Finde mittels Simulated Annealing eine Zerlegung z von P, für die $\mathrm{e}(z)$ minimal ist.
- Lerne die Zerlegung z mittels Decision-Boosting.

`// Füge zwei neue Knoten` $q_{neu}^{A,B}$ `als Nachfolger von` q `hinzu.`

- $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_{neu}^{A} \cup q_{neu}^{B}$

else

`// Füge einen neuen Knoten` q_{neu} `als Nachfolger von` q `hinzu.`

- $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_{neu}$

end

end

end

`// Entferne Knoten` q `aus` $\mathcal{Q}$

- $\mathcal{Q} \leftarrow \mathcal{Q} \setminus q$

end

3. Sobald weniger als η Positivbeispiele vorhanden sind, entstehen wieder Kaskaden. Da der Eingaberaum in Phase 2 getrennt wurde, sind diese für Teilbereiche des Eingaberaums zuständig.

Es werden nicht zwangsläufig alle Phasen durchlaufen. Ist das Klassifikationsproblem sehr schwierig zu lösen, entfällt Phase 1. In diesem Fall wird bereits nach dem ersten Knoten eine Teilung des Eingaberaums vorgenommen. Ist das Klassifikationsproblem dagegen sehr einfach zu lösen, so werden die Phasen 2 und 3 nicht durchlaufen – der finale Baum hat eine Kaskadenstruktur. Abb. 5.3 zeigt einen Beispielbaum. Es ist zu sehen, wie die drei Phasen des Trainings die Baumstruktur widerspiegeln.

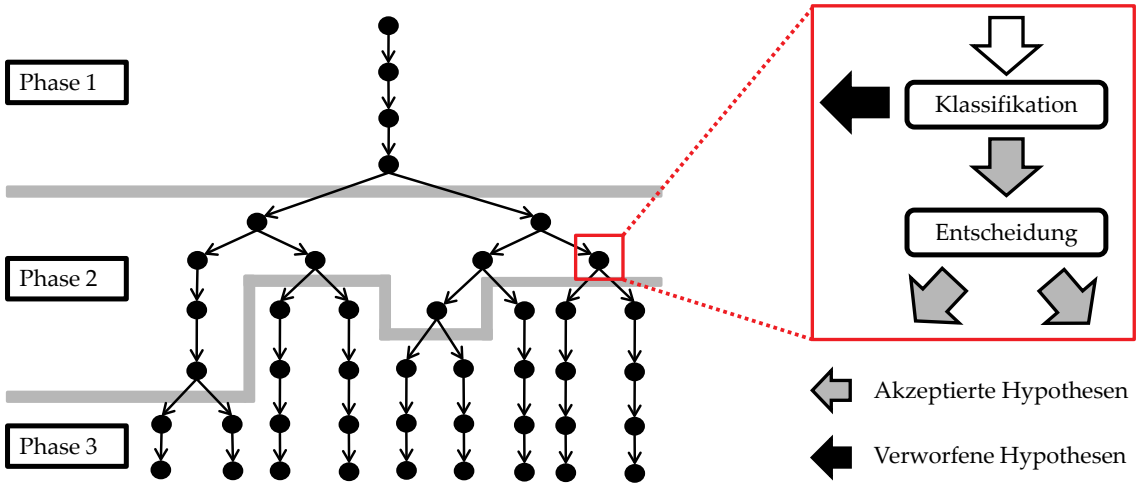


Abbildung 5.3.: Struktur eines DB-CBT. Das Training läuft in drei Phasen ab. Diese spiegeln sich in der Struktur des entstehenden Baums wider. In Phase 1 ist das Klassifikationsproblem einfach lösbar: Die Kaskadenstruktur wird beibehalten. In Phase 2 wird das Klassifikationsproblem durch Teilen vereinfacht. Phase 3 beginnt, sobald die Anzahl Positivbeispiele kleiner als η ist.

In der Phase der Anwendung wird der Baum durchlaufen. In jedem Knoten wird zuerst entschieden, ob eine Hypothese Hintergrund zeigt. Ist das der Fall, wird sie verworfen. Andernfalls wird sie gemäß der Entscheidung, der durch DB gefundenen Klassifikatoren, an einen Nachfolger weitergereicht (siehe Abb. 5.3, vergrößerter Knoten). Da Entscheidungen exklusiv sind, wird die Bearbeitung einer Hypothese abgebrochen, sobald sie in einem Knoten zurückgewiesen wird. Eine Hypothese ist akzeptiert, sobald sie in einem Blatt akzeptiert wurde. Durch Exklusivität der Entscheidungen wird jede Hypothese nur von genau einem Blatt akzeptiert (vgl. Abschnitt 4.2.2).

Die Anwendung des DB-CBT wird in Algorithmus 5.2 verdeutlicht.

Algorithmus 5.2 Anwendung des DB-CBT

Eingabe : Hypothese $h \in \mathcal{X}$
Ausgabe : Klasse $c \in \mathcal{C}$ der Hypothese h
Initialisiere: $q \leftarrow$ Wurzel des DB-CBT
while $q \neq \emptyset$ **do**
  if *Klassifikator in q weist h zurück* **then**
    `// Hypothese wird als Hintergrund klassifiziert`
    return -1
  end
  if *q hat nur einen Nachfolger* **then**
    $q \leftarrow$ Nachfolger von q
  else if *q hat zwei Nachfolger* **then**
    $x \leftarrow$ Ausgabe des zweiten Klassifikators in q
    if $x = 1$ **then**
      $q \leftarrow$ erster Nachfolger von q
    else
      $q \leftarrow$ zweiter Nachfolger von q
    end
  else
    `// Kein Nachfolger mehr vorhanden. Beende Baumdurchlauf`
    $q \leftarrow \emptyset$
  end
end
`// Hypothese wird als Objekt klassifiziert`
return 1

KAPITEL 6

Experimentelle Ergebnisse

To measure is to know.

(Sir William Thompson)

Dieses Kapitel beschreibt die durchgeführten Experimente sowie ihre Ergebnisse. Zuerst wird der Aufbau der Experimente erläutert (Abschnitt 6.1). Im Anschluss werden die eingesetzten Metriken und Kenngrößen für die Evaluierung von Klassifikatoren beschrieben (Abschnitt 6.2) sowie die eigentlichen Ergebnisse vorgestellt (Abschnitt 6.3). Das Kapitel endet mit Eindrücken des realisierten Systems in der Phase der Anwendung (Abschnitt 6.4).

6.1. Experimentaufbau

6.1.1. Trainingsparameter und Datensätze

Um die trainierten Klassifikatoren zu evaluieren, wurde ein Teil der verfügbaren Sequenzen nicht für das Training verwendet. Die ausgelassenen Sequenzen bilden einen unabhängigen Testdatensatz. Alle Auswertungen werden auf diesem durchgeführt.

Bei der Auswahl der Testsequenzen wurde darauf geachtet, dass sie einen repräsentativen Ausschnitt der gesamten Daten darstellen. D. h. der Testdatensatz umfasst Sequenzen aller Tageszeiten und Situationen (Stadt, Landstraße, Autobahn usw.). Insgesamt enthält der Testdatensatz 9 der 246 Sequenzen. Die restlichen 237 Sequenzen bilden den Trainingsdatensatz. Tabelle 6.1 zeigt eine Übersicht über den Trainings- und den Testdatensatz.

Das Training wird mit den Parametern von Tabelle 6.2 durchgeführt. Die Wahl von τ_i (4×0.97) orientiert sich an Angaben von [YHN09]. Die minimale Anzahl an Positivbeispielen η wird variiert, um den Einfluss auf die Performance des resultierenden Klassifikationsbaums festzustellen (siehe Abschnitt 6.1.4). Experimente zeigen, dass eine Variation von τ_i keinen großen Einfluss auf die Struktur des Baums hat. Daher wird der Einfluss auf die Performance der resultierenden Klassifikatoren nicht weiter untersucht. Die beobachtete Robustheit der Struktur des DB-CBT gegenüber Änderungen von τ_i deckt sich mit den Untersuchungen von [WN07].

	Trainingsdatensatz	Testdatensatz
Sequenzen	237	9
Bilder	32 875	1 525
PKW (ohne Rückfronten)	33 394	1 606
PKW-Rückfronten	36 077	1 968
LKW (ohne Rückfronten)	2 346	25
LKW-Rückfronten	3 884	410

Tabelle 6.1.: Trainings- und Testdatensatz. Bei der Auswahl der Testsequenzen wurde darauf geachtet, dass sie einen repräsentativen Ausschnitt der Daten darstellen.

Aus Zeitgründen wird die Anzahl an Positiv- und Negativbeispielen für das Training der einzelnen Knoten auf jeweils maximal 30 000 beschränkt. Das Training eines Teilbaums wird abgebrochen, sobald weniger als 15 Negativbeispiele für einen neuen Knoten gefunden wurden; es war insgesamt beendet, sobald keine zu trainierenden Knoten mehr vorhanden sind (vgl. Algorithmus 5.1).

Parameter	Wert
Maximale Baumtiefe bzw. Anzahl Stufen	40
Vorgaben für das Boosting	
Weaklearner je Tiefe/Stufe	3, 5, 12, 14, 20, 24, 32, 42, 55, 5×64, 84, 92, 3×128, 3×156, 3×182, 256, ...
Detektionsrate	0.98, 0.99, 3×0.995, 0.999, ...
Falschalarmrate	5×0.1, 2×0.5, 2×0.4, 0.1, ...
Schwellen für den Z-Wert τ_i	4×0.97
(Maximale) Anzahl Positivbeispiele	30 000
(Maximale) Anzahl Negativbeispiele	30 000
Hypothesen je Bild	131 117
Merkmale	40 292
davon Haarwavelets	29 042
davon EOHs	11 250

Tabelle 6.2.: Die wichtigsten Trainingsparameter. Bei Angaben mit „...“ wird der vorher angegebene Wert wiederholt; Die Schreibweise $k \times u$ meint die k-fache Wiederholung des Werts u. Die Parameter zur Generierung der Hypothesen finden sich in Tabelle 6.3.

6.1.2. Grundstruktur der Klassifikatoren

Für das Gesamtsystem sind insgesamt vier parallel arbeitende Klassifikatoren nötig: jeweils einer für PKW (gesamt), PKW-Rückfront, LKW (gesamt) und LKW-Rückfront (vgl. Abschnitt 1.4). Äquivalent zur parallelen Verarbeitung in Einzelklassifikatoren kann ein (einzelner) nicht-exklusiver Multiklassenklassifikator verwendet werden [HALL07]. Nicht-exklusiv bedeutet,

dass eine Hypothese mehreren Klassen zugeordnet werden kann. Eine Hypothese wird als Hintergrund klassifiziert, sobald sie keiner der anderen Klassen zufällt.

Diese Äquivalenz wird genutzt, um die Geschwindigkeit des Gesamtsystems zu erhöhen. Die ersten Knoten, also diejenigen, die einen Großteil der Hypothesen verarbeiten, werden für alle benötigten Klassifikatoren gemeinsam durchlaufen. Die Merkmale in diesen Knoten müssen somit nur einmal ausgewertet werden und werden für alle Klassen geteilt [TMF07]. Mit dieser Strategie konnte in [Kal05] ein Geschwindigkeitsgewinn von bis zu 23.5 % erzielt werden [Kal05, S. 56].

Um diesen Vorteil zu nutzen, haben alle in dieser Arbeit evaluierten Klassifikatoren die gleiche Grundstruktur: Die ersten vier Knoten werden für *alle* Einzelklassifikatoren gemeinsam durchlaufen. Nach dem vierten Knoten werden Hypothesen in zwei Schritten auf die vier Einzelklassifikatoren verteilt (Abb. 6.1). Hierbei wird auf eine Entscheidung verzichtet, d. h. Hypothesen werden an beide Nachfolger weitergereicht (vgl. Abschnitt 4.2.2).

Durch diesen Aufbau wird ein Großteil der Hypothesen (ca. 87 %, siehe Abschnitt 6.3.2) mit gemeinsamen Merkmalen in den ersten vier Knoten zurückgewiesen. Die Knoten der Tiefe 6 (schwarz in Abb. 6.1) bilden die Wurzel der untersuchten Klassifikatoren (siehe Abschnitt 6.1.4).

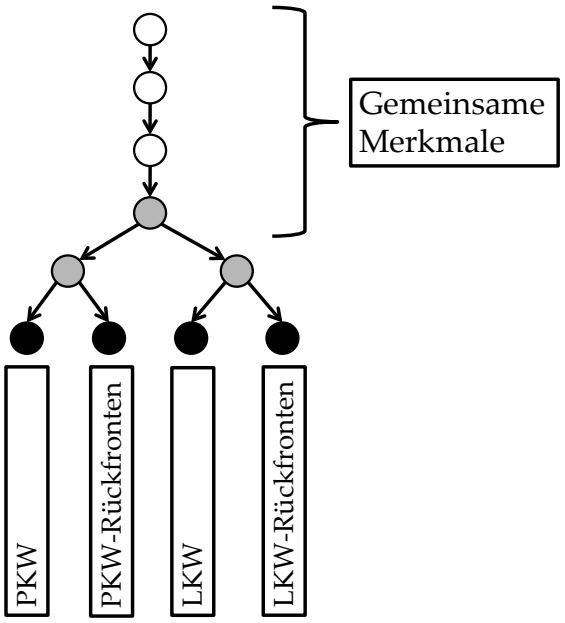
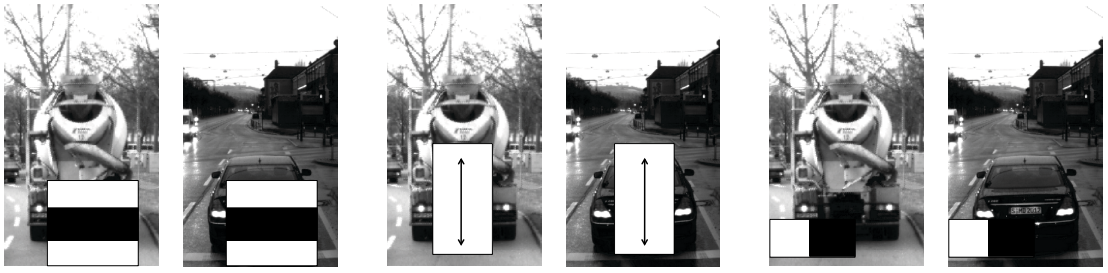


Abbildung 6.1.: Grundstruktur der evaluierten Klassifikatoren. In den ersten Knoten werden alle Hypothesen aller nötigen Einzelklassifikatoren gemeinsam verarbeitet (weiße Knoten). Die Merkmale dieser Knoten werden geteilt. In zwei Schritten (graue Knoten) werden die Hypothesen auf die vier Einzelklassifikatoren verteilt. Hierbei werden Hypothesen an beide Nachfolger weitergereicht. In den schwarzen Knoten beginnen die Einzelklassifikatoren.

Die Positivbeispiele für das Boosting der Knoten, die für mehrere Klassen zuständig sind (d. h. die Knoten der Tiefen 1–5) wurden gleichmäßig aus den Daten der verschiedenen Klassen gewählt. Bspw. kamen für das Training der Wurzel (Tiefe 1) jeweils maximal 7 500 Positivbeispiele für die Klassen „PKW“, „PKW-Rückfront“, „LKW“ und „LKW-Rückfront“ zum Einsatz ($4 \times 7\,500 = 30\,000$).

Die in den ersten Stufen gewählten Merkmale spiegeln „gemeinsame“ Eigenschaften aller Klassen wider. Abb. 6.2 zeigt die drei Merkmale des ersten Knotens.



(a) (b) (c)

Abbildung 6.2.: Merkmale des ersten Knotens. Im ersten Knoten werden Hypothesen gemeinsam für PKW und LKW verarbeitet. Daher sind die gewählten Merkmale charakteristisch für beide Fahrzeugtypen. (a) Untere Kante (Haarwavelet). (b) Vorwiegend vertikale Kanten (EOH). (c) Markante Ecke (Haarwavelet).

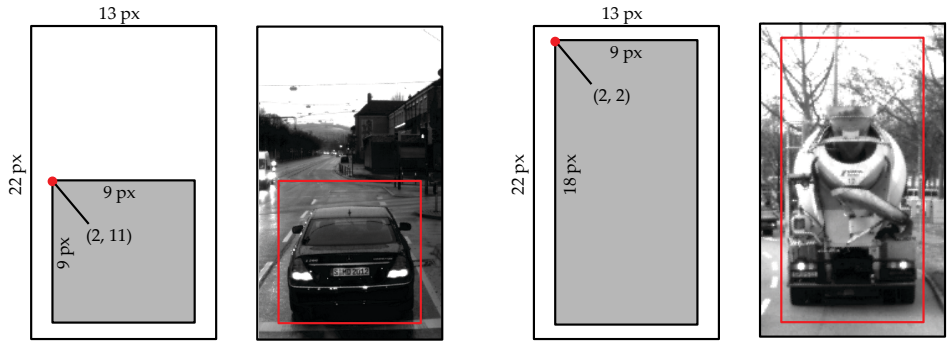
6.1.3. Gewählte Fenster

Um im gemeinsamen Bereich der Klassifikatoren (d. h. in den ersten vier Knoten) sowohl PKW als auch LKW zu detektieren, müssen die Norm- und Objektfenster so gewählt werden, dass gemeinsame Merkmale möglich sind. Problematisch ist, dass LKW höher als breit sind, wohingegen PKW eine tendenziell quadratische Grundform (von hinten gesehen) besitzen. Diese beiden Seitenverhältnisse müssen kombiniert werden.

Daher ist das Normfenster für alle untersuchten Klassifikatoren auf 13 mal 22 Pixel festgelegt. Die Objektfenster von PKW und LKW unterscheiden sich. Um der Höhe von LKW Rechnung zu tragen, wird ihr Objektfenster im Verhältnis 2 : 1 zentral im Normfenster platziert (Abb. 6.3(b)). Das Objektfenster für PKW ist quadratisch gewählt (Abb. 6.3(a)). Die Position der Objektfenster wird so festgelegt, dass möglichst viele gemeinsame Kanten vorhanden sind. Abb. 6.3 zeigt das verwendete Normfenster und die Objektfenster für PKW bzw. LKW. Die Objektfenster der Rückfronten wurden entsprechend den Objektfenstern des zugehörigen Fahrzeugtyps gewählt.

Die gesuchten Objekte haben Einfluss auf die Hypothesengenerierung. Insbesondere die maximale und minimale Objekthöhe sind relevant (vgl. Abschnitt 3.5). Die Angabe der Objekthöhe muss sich am höheren der beiden Objektfenster bemessen, da das kleinere dann notwendigerweise innerhalb des Suchtunnels liegt. Daher ist die Vorgabe der Höhe von LKW nötig. Diese wurde auf minimal 2.5, maximal 5 Meter festgelegt. Damit ist die Höhe von PKW minimal 1.25, maximal 2.5 Meter (das Objektfenster von PKW ist halb so hoch wie das Objektfenster von LKW). Tabelle 6.3 gibt eine Übersicht über die Parameter der Hypothesengenerierung. Die gewählte minimale Hypothesengröße ergibt eine maximale Detektionsreichweite von ca. 100 Metern, berechnet bei einer erwarteten Fahrzeugbreite von 1.8 Metern.

Das Merkmalsfenster ist gleich dem Normfenster gewählt. D. h. an jeder Position innerhalb des Normfensters können Merkmale existieren.



(a) (b)

Abbildung 6.3.: Gewählte Norm- und Objektfenster. Um eine gemeinsame Detektion von PKW und LKW zu ermöglichen, wird das Normfenster auf 13 mal 22 Pixel festgelegt. Die Objektfenster sind für PKW und LKW unterschiedlich gewählt. Der „Rand“ um die Objekte ist auf 2 Pixel im Normfenster festgelegt. Die Position der Objektfenster ist so gewählt, dass möglichst viele gemeinsame Kanten vorhanden sind. (a) Das Objektfenster für PKW ist quadratisch gewählt. (b) LKW besitzen ein Objektfenster mit dem Seitenverhältnis 2 : 1 um ihrer Höhe Rechnung zu tragen.

Parameter	**Wert**
Hypothesenhöhe	44 – 480 Pixel
Objekthöhe	2.5 – 5 Meter (LKW, $\hat{=}$ 1.25 – 2.5 für PKW)
Quantisierungen (in x-, y- und z-Richtung)	0.05, 0.05, 0.09
Relaxierungswinkel	2°

Tabelle 6.3.: Parameter der Hypothesengenerierung. Die Objekthöhe bemisst sich an LKW, da diese das höhere Objektfenster besitzen. Die Wahl von 2.5 – 5 Meter für LKW entspricht einer Objekthöhe von 1.25 – 2.5 Meter für PKW.

6.1.4. Untersuchte Klassifikatoren

Ausgehend von der gemeinsamen Grundstruktur (siehe Abschnitt 6.1.2) werden folgende Klassifikatoren sowohl für PKW als auch für PKW-Rückfronten untersucht:

- Eine Viola-Jones-Kaskade.
- Ein DB-CBT mit $\eta = 4000$ und ein DB-CBT mit $\eta = 8000$. Die Wahl von η entspricht ungefähr 10 bzw. 20 % der Trainingsbeispiele für PKW/PKW-Rückfronten.
- Ein Klassifikationsbaum, dessen Teilung des Eingaberaums mit dem k-Means-Algorithmus auf den Merkmalsvektoren statt mit dem Verfahren des DB-CBT vorgenommen wird. Als Kriterium für die Initiierung einer Trennung kommt der gleiche Schwellwert auf den Z-Wert zum Einsatz. In der Phase der Anwendung werden Hypothesen demjenigen Nachfolger zugeordnet, dem sie auch in der Phase des Trainings

zugeteilt worden wären. D. h. die Entscheidung wird anhand der L_2-Distanz der Hypothese zu den Clusterzentren getroffen. Wie auch beim DB-CBT wird eine weitere Teilung unterbunden, wenn die Anzahl Positivbeispiele unter 4000 bzw. 8000 fällt.

Im weiteren Verlauf werden die trainierten Baumklassifikatoren mit 4000-er bzw. 8000-er Baum bezeichnet. Die Bezeichnung spiegelt das jeweilige η des Trainings wider. 4000-er Bäume meint also sowohl den DB-CBT als auch den k-Means-Baum, trainiert mit $\eta = 4000$.

Für LKW und LKW-Rückfronten sind bei den gewählten Trainingsparametern nicht ausreichend Beispiele für das Training eines DB-CBT oder eines k-Means-Baums vorhanden. Daher wird für LKW und LKW-Rückfronten lediglich eine Kaskade untersucht.

Das Training der Klassifikatoren nahm über sieben Wochen auf 16 Computern in Anspruch. Sowohl das Boosting als auch das Sammeln der Beispiele wurde im Zuge dieser Arbeit parallelisiert, um der enormen Datenmassen und dem Bedarf an Rechenleistung Herr zu werden. Dennoch überstieg das Training der 8000-er Bäume und der Kaskade zeitlich den Rahmen dieser Arbeit. Die 4000-er Bäume wurden vollständig und erschöpfend trainiert.

Das Training der Kaskade wurde nach 30 Stufen für PKW und nach 29 Stufen für PKW-Rückfronten abgebrochen. Die 8000-er Bäume werden bei einer Tiefe von 23 evaluiert. Der 4000-er DB-CBT hat Tiefe 22 (PKW) bzw. Tiefe 21 (PKW-Rückfronten); der 4000-er k-Means-Baum hat Tiefe 23 sowohl für PKW als auch für PKW-Rückfronten. Die Struktur der untersuchten Klassifikatoren wird in Anhang A visualisiert.

6.2. Metriken und Kenngrößen für die Performance-Messung

6.2.1. Bewertung von Detektionen

Bei der Evaluierung von Klassifikatoren ist zunächst festzulegen, was eine korrekte Detektion ist. Hierfür ist ein Abstandsmaß für Hypothesen nötig. Mit diesem kann der Abstand zwischen einer Hypothese, welche das Objekt „perfekt“ zeigt (sog. *Grundwahrheit*, engl. *ground truth*, im Folgenden auch oft *Label* genannt), und einer Detektion berechnet werden. Ist der Abstand kleiner als eine vorgegebene Schwelle, so ist die Detektion korrekt (*True Positive*, TP), andernfalls ist sie es nicht (*False Positive*, FP).

In der Literatur wird häufig (u. a. von [Kal05],[SHL07],[KN09],[EG09]) die *Überdeckung* (engl. *coverage*) als Abstandsmaß verwendet. Sie ist für zwei Hypothesen $B_{1,2}$ definiert als

$$\mathrm{cov}(B_1, B_2) = \frac{\mathrm{A}(B_1 \cap B_2)}{\mathrm{A}(B_1 \cup B_2)} \in [0,\, 1]\,.$$

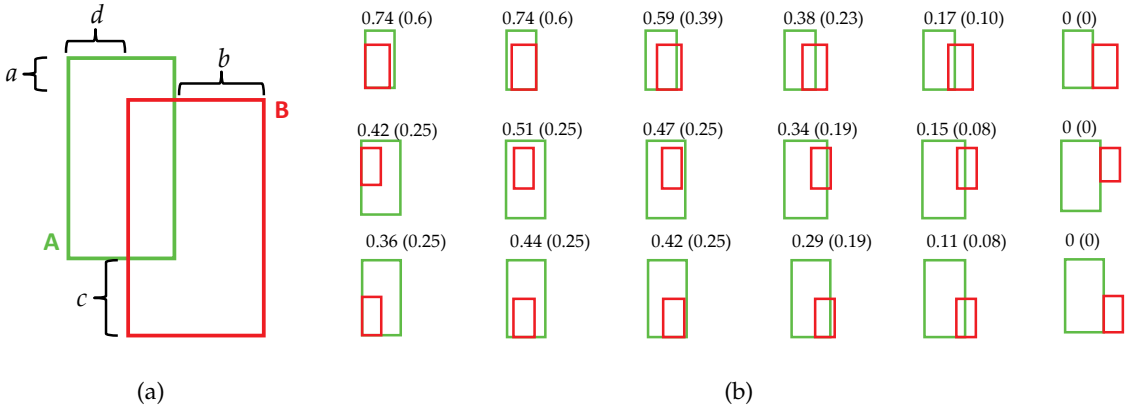
A ist hierbei der Flächeninhalt eines Rechtecks. $\mathrm{cov}(B_1, B_2)$ ist also der gemeinsame Anteil am (gesamten) Flächeninhalt von B_1 und B_2.

Nachteilig an der Verwendung von $\mathrm{cov}(.,.)$ ist allerdings, dass die Überdeckung unabhängig von der Position des gemeinsamen Bereichs zweier Hypothesen ist, d. h. zentral gelegene Übereinstimmungen werden nicht favorisiert (siehe auch Abb. 6.4). Um diesen Nachteil zu vermeiden, schlägt Löhlein in [Lö09] die *Ähnlichkeit* (engl. *affinity*) als alternatives Abstandsmaß vor. Diese bemisst die Distanz zweier Hypothesen an der Distanz zwischen ihren Rändern.

Sie sind umso ähnlicher, je näher sich ihre Ränder sind. Für zwei Hypothesen A und B ist die Ähnlichkeit $\mathrm{aff}(A, B)$ zwischen ihnen definiert als

$$\mathrm{aff}(A,B) = \max\left(0, 1 - \sqrt{\frac{\left(\frac{a}{h_{max}}\right)^2 + \left(\frac{b}{w_{max}}\right)^2 + \left(\frac{c}{h_{max}}\right)^2 + \left(\frac{d}{w_{max}}\right)^2}{\left(\frac{h_A}{h_{max}}\right)^2 + \left(\frac{h_B}{h_{max}}\right)^2}}\right) \in [0,\,1]\,. \tag{6.1}$$

Die Bedeutung der einzelnen Variablen a – d ist in Abb. 6.4(a) dargestellt. $h_{A,B}$ ist die Höhe der Hypothese A bzw. B. Dementsprechend ist $w_{A,B}$ deren Breite. Damit ist h_{max} durch $h_{max} = \max(h_A, h_B)$, w_{max} durch $w_{max} = \max(w_A, w_B)$ bestimmt.



(a) (b)

Abbildung 6.4.: Berechnung und Beispiele der Ähnlichkeit zweier Hypothesen. Die Ähnlichkeit $\mathrm{aff}(.,.)$ ist eine Metrik auf Hypothesen. Der Abstand zweier Hypothesen ist umso geringer, je ähnlicher sich diese sind. (a) Bedeutung der Variablen von Gleichung (6.1). (b) Beispiele für die Ähnlichkeit zweier Hypothesen. In Klammern ist die Überdeckung der Hypothesen angegeben. Es ist deutlich zu sehen, dass die Ähnlichkeit ein geeigneteres Maß darstellt, da sie „zentrale" Übereinstimmungen favorisiert.
Quelle: [Lö09]

Mit dem so definierten Abstandsmaß $\mathrm{aff}(.,.)$ können Detektionen bewertet werden. Hierbei werden drei mögliche Fälle unterschieden:

1. Detektionen, deren Ähnlichkeit mit *allen* Labels unter einem Schwellwert σ_{aff} liegt, bekommen die Klasse *Negative*. Bei Detektionen dieser Art handelt es sich um Falschalarme.

2. Hat eine Detektion eine Ähnlichkeit $\geq \sigma_{\mathrm{aff}}$ mit einem Label und entspricht das Label der Klasse der Detektion – bspw. eine PKW-Detektion und ein PKW-Label – so wird der Detektion die Klasse *Positive* zugewiesen. Detektionen dieser Art sind korrekt.

3. Ist die Ähnlichkeit zwischen Detektion und Label größer als σ_{aff}, passen die Klassen jedoch nicht zueinander – etwa die Detektion einer PKW-Rückfront mit einem Label für PKW – so wird die Detektion ignoriert (*Ignore*), d. h. diese Detektionen werden weder als Falschalarm noch als korrekte Detektion gewertet.

In dieser Arbeit wird die Performance der Klassifikatoren mit $\sigma_{\text{aff}} = 0.2$ ausgewertet. Die Einordnung der Detektionen unterscheidet sich je nach Evaluierung und wird bei der jeweiligen Auswertung direkt erläutert.

6.2.2. Kenngrößen der Performance

In diesem Abschnitt werden die relevanten Kenngrößen zur Bewertung der Performance erläutert. Die hier vorgestellten Maße sind nur ein kleiner Ausschnitt aus den möglichen Kenngrößen für die Evaluierung von Klassifikatoren. Für eine Übersicht über weitere sei auf [Wal08, S. 121 ff.] und [Kal05, S. 44 ff.] verwiesen.

TP-Rate: Die TP-Rate ist der Anteil der detektierten Objekte an der Menge der Hypothesen, die ein Objekt zeigen (TP).

Falschalarmrate: Die Falschalarmrate ist der Anteil der Detektionen an der Menge der Hypothesen, die *kein* Objekt zeigen (FP).

Detektionsrate: Die Detektionsrate ist der Anteil an detektierten Objekten an insgesamt im Datensatz vorhandenen Objekten. Bei einer Detektionsrate von 100 % werden somit alle im Datensatz vorhandenen Objekte detektiert.

Präzision: Die Präzision ist die Wahrscheinlichkeit, dass eine Detektion einem Objekt entspricht. Bei einer Präzision von 1 zeigt jede Detektion auch tatsächlich ein Objekt.

6.2.3. Visualisierung der Performance

Zur Visualisierung der Performance kommen ROC[1]- und PR[2]-Diagramme zum Einsatz. ROC-Diagramme visualisieren den Trade-off zwischen Detektions- und Falschalarmrate [Wal08, S. 184 ff.], PR-Diagramme stellen die Präzision der Detektionsrate gegenüber [WF05, S. 171 f.]. Für eine weitergehende Einführung in ROC-Analyse sei auf [Faw06] verwiesen. Eine ausführlichere Diskussion von PR-Diagrammen bietet [WF05, S. 171 f.].

Die Generierung von ROC- und PR-Diagrammen ist sehr aufwendig, da für jeden Punkt im ROC- bzw. PR-Raum ein neuer Klassifikator trainiert werden muss [Wal08, S. 186 f.]. Eine Vereinfachung bietet die im Folgenden beschriebene Vorgehensweise, bei der die Tatsache genutzt wird, dass eine „Akzeptanzschwelle“ ρ existiert, ab der eine Hypothese als Objekt klassifiziert wird. Wird ρ verringert, werden alle Hypothesen, die für das frühere ρ als Objekt klassifiziert wurden, weiter als Objekt eingestuft[3]. Zusätzlich werden nun allerdings diejenigen Hypothesen, die über dem neuen, nicht jedoch über dem alten ρ liegen, akzeptiert. Folglich steigt der Anteil der akzeptierten Hypothesen. D. h. durch Absenken von ρ wird ein neuer Punkt im ROC- und PR-Raum gefunden. Dieser Prozess lässt sich wiederholen, bis jede Hypothese akzeptiert wird, die TP-Rate also 100 % ist. Für eine weitergehende Analyse des skizzierten Verfahrens sei auf [Faw06] verwiesen.

[1] Receiver Operating Characteristic
[2] Precision over Recall
[3] Ohne Einschränkung der Allgemeinheit wird hierbei davon ausgegangen, dass Hypothesen als Objekt gewertet werden, sobald sie *über* der Akzeptanzschwelle liegen.

Für eine Kaskade lässt sich das schrittweise Absenken der Akzeptanzschwelle durch sukzessives Löschen von Stufen (von der letzten Stufe an) realisieren. Für Bäume wird das Verfahren analog angewandt, indem schrittweise alle Knoten einer Tiefe (von der maximalen Tiefe an) gelöscht werden.

Um die Interpretierbarkeit zu erhöhen, werden die erstellten ROC-Diagramme leicht verändert dargestellt. Statt der Falschalarmrate werden Falschalarme je Bild auf der x-Achse aufgetragen. Damit wird ein besserer Eindruck von der Systemleistung vermittelt, da die Anzahl Falschalarme je Bild (im Gegensatz zur Anzahl Hypothesen) besser interpretierbar und damit aussagekräftiger ist. Zudem kann von Falschalarmen je Bild leichter auf die im automobilen Bereich verbreitete [Wal08, S. 121 f.],[SZ06, S. 94 ff.] Kennzahl „Fehler je Zeiteinheit“ umgerechnet werden (je Bild vergehen 40 Millisekunden, vgl. Abschnitt 1.3). Auf der y-Achse der ROC-Diagramme wird die Detektionsrate aufgetragen. Dies weicht von der „traditionellen“ Darstellung ab, da die Detektionsrate nicht der normalerweise verwendeten TP-Rate entspricht. Die TP-Rate erfasst nicht diejenigen Objekte, für die keine Hypothese existiert, sie ist also tendenziell höher als die Detektionsrate. Da die TP-Rate allerdings nicht dem menschlichen Verständnis von Korrektheit entspricht (nicht erkannte Objekte können bei einer TP-Rate von 100 %, nicht jedoch bei einer Detektionsrate von 100 % auftreten), wird für die hier verwendeten ROC-Diagramme die Detektionsrate als Kennzahl gewählt.

6.2.4. Ein Abstandsmaß für Winkelkategorien

Für die Untersuchung der Genauigkeit der Winkelschätzung ist ein Abstandsmaß nötig, das die Distanz zwischen Winkelkategorien (siehe Abschnitt 2.8) angibt. Anhand diesem kann der Anteil korrekt kategorisierter Detektionen ausgewertet werden.

Daher sei folgendes Maß $m(c)$ für eine Kategorie $c = (c_1, c_2)$ mit $c_1 \in \{\text{right}, \text{left}, \text{straight}\}$ und $c_2 \in \mathbb{N}$ definiert:

$$m(c) = (10 - c_2) \times \begin{cases} -1, & \text{falls } c_1 = \text{left} \\ 1, & \text{sonst} \end{cases} .$$

Damit lässt sich die Distanz $d(a, b)$ zwischen zwei Kategorien a und b wie folgt berechnen:

$$d(a, b) = |m(a) - m(b)| .$$

Tabelle 6.4 gibt einige Beispiele für Abstände zwischen Kategorien.

a	b	$d(a, b)$
(left, 12)	(left, 12)	0
(left, 11)	(left, 12)	1
(straight, 10)	(left, 12)	2
(right, 12)	(left, 12)	4

Tabelle 6.4.: Beispiele für Abstände mit dem definierten Abstandsmaß $d(.,.)$.

6.3. Ergebnisse

In diesem Abschnitt werden die Ergebnisse der experimentellen Untersuchung der trainierten Klassifikatoren angegeben. Neben der Performance (Abschnitt 6.3.1) wird die Geschwindigkeit der Klassifikatoren (Abschnitt 6.3.2) untersucht. Zusätzlich wird die Qualität, d. h. Genauigkeit und Verlässlichkeit, der Kategorisierung in Winkelkategorien (Abschnitt 6.3.3) und die Verwechslungsgefahr zwischen den Fahrzeugtypen (Abschnitt 6.3.4) evaluiert.

6.3.1. Performance der Klassifikatoren

Die Performance der Klassifikatoren wird für PKW, PKW-Rückfronten, LKW und LKW-Rückfronten einzeln untersucht. Ignoriert werden hierbei alle Detektionen, die nicht vom jeweils korrekten Klassifikator stammen oder die sich mit einem Label überschneiden, welches nicht zum untersuchten Klassifikator passt. Bspw. werden bei der Untersuchung der PKW-Klassifikatoren alle Detektionen der Klassifikatoren für PKW-Rückfront, LKW und LKW-Rückfront ignoriert. Ferner werden alle Detektionen des PKW-Klassifikators ignoriert, die sich mit einem Label einer PKW-Rückfront, eines LKW oder einer LKW-Rückfront überschneiden, nicht jedoch mit einem PKW-Label.

Die Abb. 6.5 und 6.6 zeigen die Performance der PKW-Klassifikatoren. Alle Klassifikatoren zeigen in etwa die gleiche Performance. Im ROC-Diagramm (Abb. 6.5) ist erkennbar, dass das Training der PKW-Kaskade nicht beendet wurde – es existieren keine Arbeitspunkte mit weniger als zwei Falschalarmen je Bild. Es ist allerdings anzunehmen, dass die Kurve der Kaskade bei weiterem Training einen ähnlichen Verlauf nimmt wie die Kurven der anderen Klassifikatoren. Alle Klassifikatoren garantieren Detektionsraten $> 80\,\%$, im Fall der 4000-er Bäume bei weniger als 0.3 Falschalarmen je Bild. Die erreichte Präzision liegt für diesen Arbeitspunkt bei ca. 80 % (siehe Abb. 6.6).

Auch bei der Untersuchung bei der Detektion von PKW-Rückfronten unterscheiden sich die Klassifikatoren nicht stark voneinander. Allerdings sind hier die Unterschiede zwischen den einzelnen Klassifikatoren dennoch etwas größer als bei den PKW-Klassifikatoren (Abb. 6.7 und 6.8). Hervorzuheben ist der 8000-er k-Means-Baum, der eine Detektionsrate von ca. 90 % bei 0.3 Falschalarmen je Bild aufweist. Die übrigen Klassifikatoren garantieren eine Detektionsrate von ca. 85 % bei dieser Falschalarmrate. Für die Kaskade existiert kein Arbeitspunkt in diesem Bereich, weswegen kein quantitativer Vergleich möglich ist. Anhand des Kurvenverlaufs lässt sich aber eine Detektionsrate von ca. 90 % extrapolieren.

Die Performance bei der Detektion von PKW-Rückfronten liegt generell über der Performance bei der Detektion von PKW. Dies lässt sich damit erklären, dass die Variabilität der Objekt-Ansichten bei Detektion kompletter PKW sehr viel höher ist (siehe auch Abb. 1.1, Seite 14) als bei Einschränkung auf PKW-Rückfronten. Folglich ist die Detektion von PKW ein schwierigeres Problem als die Detektion von PKW-Rückfronten.

Die Abb. 6.9 und 6.10 zeigen die Performance bei der Detektion von LKW bzw. LKW-Rückfronten. Beide Experimente sind aufgrund der geringen Anzahl Test- und Trainingsdaten (vgl. Tabelle 6.1) nur eingeschränkt aussagekräftig[4]. Die Ergebnisse werden dennoch, aus Gründen der Vollständigkeit, angegeben. Die Detektion von LKW funktioniert sehr gut –

[4] Für Untersuchungen über den Zusammenhang zwischen Trainingsdatengröße und Performance sei auf [EG08] verwiesen.

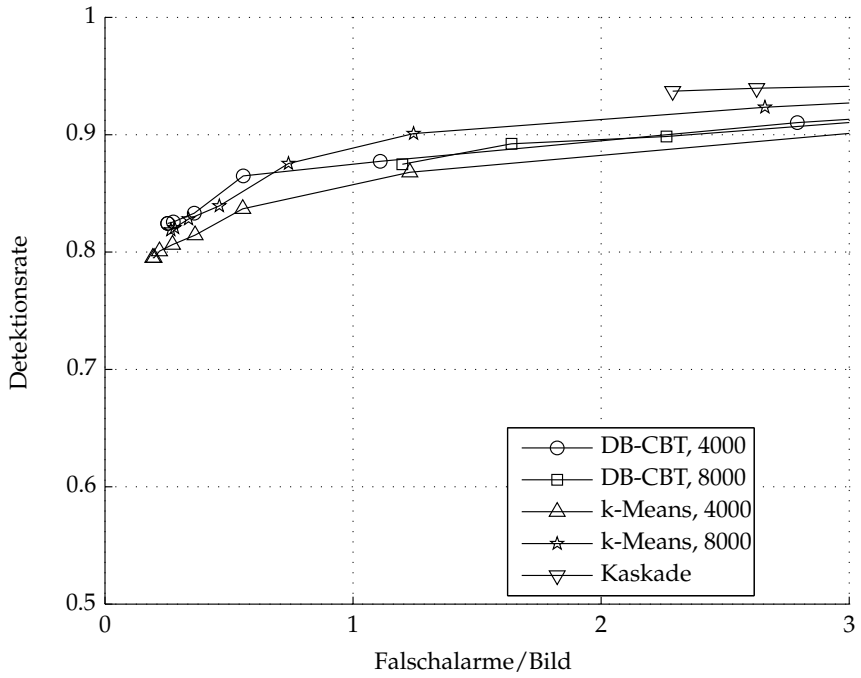


Abbildung 6.5.: ROC-Diagramm der PKW-Klassifikatoren. Die PKW-Klassifikatoren unterscheiden sich nur geringfügig voneinander. Die Kaskade ist den anderen Klassifikatoren leicht überlegen, allerdings ist sichtbar, dass ihr Training nicht beendet werden konnte. Es ist anzunehmen, dass die Kurve der Kaskade bei weiterem Training einen ähnlichen Verlauf nimmt wie die Kurven der anderen Klassifikatoren.

die Kaskade zeigt Detektionsraten von 92 % bei weniger als einem Falschalarm auf hundert Bildern (Abb. 6.9). Demgegenüber steht eine geringe Performance bei der Detektion von LKW-Rückfronten. Hier werden lediglich 65 % der Objekte detektiert (Abb. 6.10).

Neben der Untersuchung der einzelnen Klassifikatoren wird im Folgenden zusätzlich untersucht, welche Performance erreicht wird, wenn die Unterscheidung zwischen Rückfront und Gesamtfahrzeug aufgehoben wird. D. h. Detektionen werden als Positiv gewertet, sobald der Fahrzeugtyp der Detektion mit dem Fahrzeugtyp des Labels übereinstimmt. Somit werden Detektionen des Klassifikators für PKW-Rückfronten auch bei PKW-Labeln als korrekte Detektion gezählt. Stimmt der Fahrzeugtyp nicht überein, werden Detektionen weiterhin ignoriert.

Die Abb. 6.11 und 6.12 zeigen die Performance bei der Detektion von PKW. Erwartungsgemäß ist die Detektionsrate bei diesem Experimentaufbau größer als bei einer getrennten Detektion: Alle Klassifikatoren erreichen Detektionsraten von über 89 %. Mithin ist allerdings die Falschalarmrate schlechter. Maximal werden 0.5 Falschalarme je Bild erreicht, was einer Präzision von ca. 75 % entspricht (Abb. 6.12). Wie auch bei den anderen Experimenten zu beobachten, sind die Unterschiede der Performance zwischen den Klassifikatoren sehr gering.

Abb. 6.13 zeigt die Ergebnisse bei der Durchführung des Experiments mit LKW. Wie bei den anderen Ergebnissen für LKW, gilt auch für dieses Ergebnis die Einschränkung bzgl. der Aussagekraft aufgrund des geringen Datenumfangs. Die Performance bei der Detektion

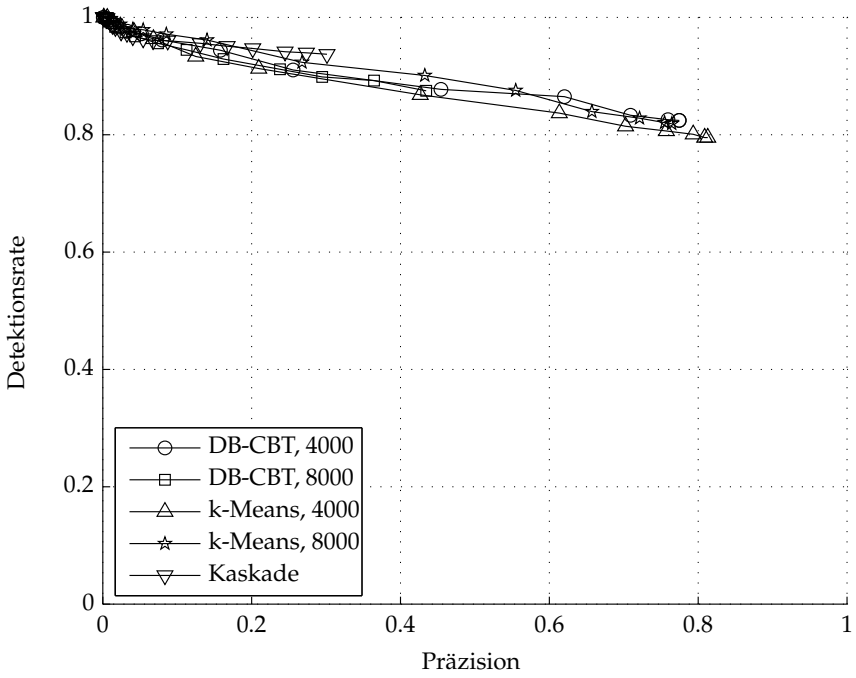


Abbildung 6.6.: PR-Diagramm der PKW-Klassifikatoren. Auch im PR-Raum betrachtet unterscheiden sich die untersuchten Klassifikatoren nur wenig.

entspricht in etwa der Performance bei der Detektion von LKW-Rückfronten. Das sich keine deutlichere Verbesserung gegenüber der getrennten Detektion zeigt, lässt sich damit erklären, dass die Anzahl LKW im Vergleich zur Anzahl LKW-Rückfronten vergleichsweise gering ist (25 gegenüber 410, vgl. Tabelle 6.1).

Zusammenfassend lässt sich sagen, dass die Detektionsleistung für PKW und PKW-Rückfronten für alle untersuchten Klassifikatoren annähernd gleich ist. Für PKW-Rückfronten sind die Unterschiede etwas größer als für PKW, die Performance der Kaskade und des 8000-er k-Means-Baums ist etwas größer als die der restlichen Klassifikatoren. Die Detektionsleistung lässt sich noch weiter erhöhen, wenn auf die Unterscheidung zwischen PKW und PKW-Rückfront verzichtet wird. In diesem Fall ist eine Detektionsrate von mindestens 89 % bei 0.5 Falschalarmen je Bild mit allen Klassifikatoren möglich.

6.3.2. Geschwindigkeit der Klassifikatoren

Neben der Performance ist die Geschwindigkeit eines Klassifikators von entscheidender Bedeutung. Diese wird in benötigten Weaklearnern je Hypothese angegeben. Da jeder Weaklearner genau einem Merkmal entspricht, ist die Laufzeit einer Hypothese genau die Anzahl benötigter Merkmale bis zu ihrer Klassifikation. Diese ist proportional zur Zeit je Hypothese, d. h. durch Auswertung der benötigten Weaklearner je Hypothese werden Rückschlüsse auf die benötigte Zeit je Hypothese ermöglicht.

Folgende Fragestellungen sind bei der Geschwindigkeitsuntersuchung von Interesse:

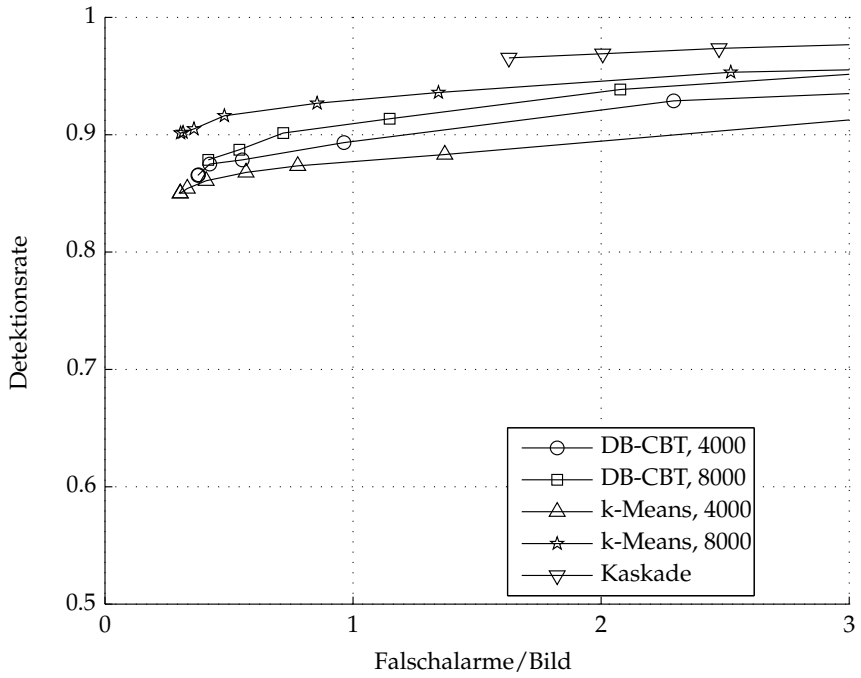


Abbildung 6.7.: ROC-Diagramm der Klassifikatoren für PKW-Rückfronten. Bei der Detektion von PKW-Rückfronten zeigt der 8000-er k-Means-Baum leicht bessere Ergebnisse im Vergleich zu den anderen Bäumen. Für die Kaskade liegen im interessanten Arbeitsbereich (weniger als ein Falschalarm je Bild) keine Daten vor, es ist jedoch aus dem Kurvenverlauf sichtbar, dass die Detektionsrate ähnliche Bereiche wie die des 8000-er k-Means-Baums erreichen kann. Die übrigen Klassifikatoren garantieren eine Detektionsrate von ca. 85 % bei 0.3 Falschalarmen je Bild.

1. Wie groß ist der durch Feature-Sharing (siehe Abschnitt 6.1.2) erreichte Geschwindigkeitsvorteil, d. h. wie groß ist der Anteil gemeinsamer Merkmale, gemessen an den benötigten Merkmalen?

2. Was sind erwartete und maximale Laufzeit einer Hypothese? Während die erwartete Laufzeit einem Durchschnittswert entspricht und einen Eindruck über die Geschwindigkeit des Klassifikators vermittelt, ist die – potentiell selten eintreffende – maximale Laufzeit v. a. für Aussagen über garantierte Antwortzeiten von Interesse – eine Fragestellung, die für automobile Assistenzsysteme zentral ist [SZ06, S. 62 ff.].

Die ersten vier Knoten der Klassifikatorgrundstruktur werden für alle Hypothesen gemeinsam durchlaufen (vgl. Abschnitt 6.1.2). In diesen Knoten werden 36, 24, 19 bzw. 8 % aller Hypothesen zurückgewiesen; Lediglich 13 % der Hypothesen erreichen die Tiefe 5. Folglich werden ca. 87 % der Merkmale gemeinsam von allen Klassifikatoren verwendet. Das Feature-Sharing spart also einen Großteil der Merkmale ein.

Da für die beiden LKW-Klassifikatoren lediglich Kaskaden trainiert wurden, beziehen sich die nachfolgenden Geschwindigkeitsuntersuchungen auf die PKW-Klassifikatoren. Für LKW

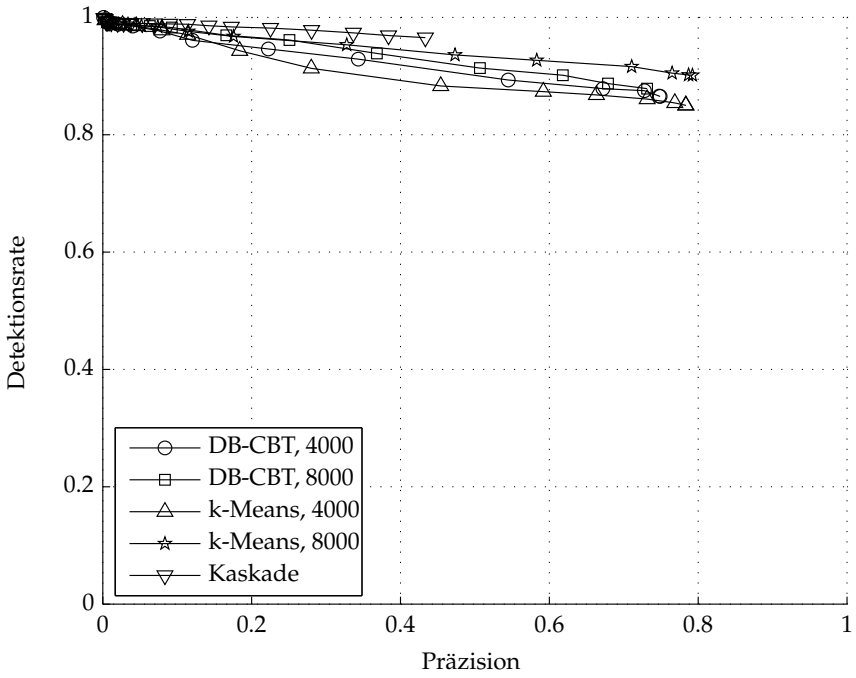


Abbildung 6.8.: PR-Diagramm der Klassifikatoren für PKW-Rückfronten.

ergab sich eine erwartete und maximale Laufzeit von 17.03 bzw. 506 Weaklearnern für den LKW-Klassifikator sowie eine erwartete und maximale Laufzeit von 16.74 bzw. 387 Weaklearnern für den LKW-Rückfronten-Klassifikator.

Kennwert	**Kaskade**	**DB-CBT, 4000**	**k-Means, 4000**
Erwartete Laufzeit (inkl. Feature-Sharing)	29.93	22.36	21.37
Vergleich zu Kaskade	100 %	75 %	71 %
Erwartete Laufzeit (exkl. Feature-Sharing)	170.55	106.71	98.33
Vergleich zu Kaskade	100 %	62 %	58 %
Maximale Laufzeit	3 637	1 321	1 524
Vergleich zu Kaskade	100 %	36 %	42 %

Tabelle 6.5.: Geschwindigkeitsvergleich zwischen Kaskade und den 4000-er Bäumen (PKW).

Die Tabellen 6.5 und 6.6 fassen die Laufzeitdaten für PKW zusammen. Bei Verwendung des 4000-er DB-CBT ergibt sich ein erwarteter Laufzeitvorteil von 25 % gegenüber der Kaskade, bei Einsatz des 8000-er DB-CBT ist die erwartete Laufzeit um 23 % geringer. Rechnet man den gemeinsamen und für alle Klassifikatoren gleichen Bereich heraus, ergibt sich ein erwarteter Laufzeitvorteile von 38 bzw. 34 % des 4000-er bzw. 8000-er DB-CBT.

Bei der maximalen Hypothesenlaufzeit zeigen sich die Geschwindigkeitsvorteile des DB-CBT nocht deutlicher. Für Hypothesen, die den längsten Weg durch den Baum nehmen (*worst-case*-Laufzeit) spart man mit dem DB-CBT 64 bzw. 53 % (4000-er bzw. 8000-er DB-CBT) der

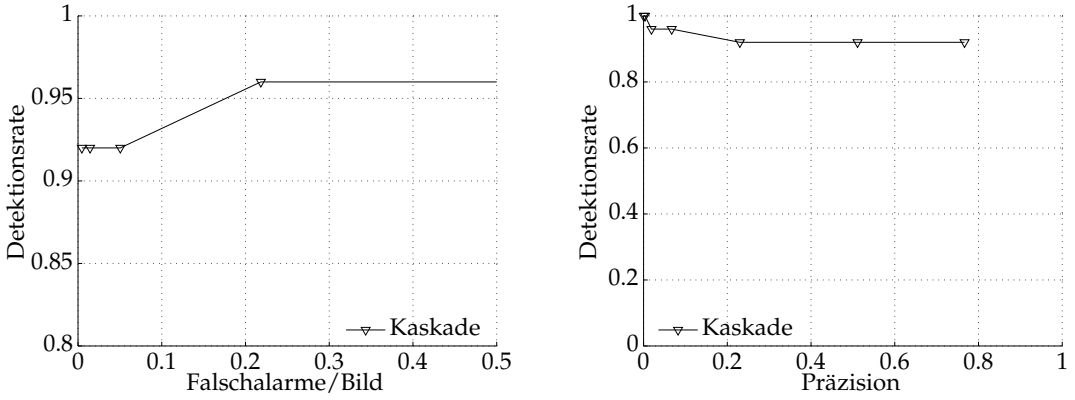


Abbildung 6.9.: ROC- und PR-Diagramm des LKW-Klassifikators. Die Performance bei der Detektion von LKW ist sehr hoch, die Ergebnisse sind jedoch aufgrund der geringen Anzahl Test- und Trainingsdaten (vgl. Tabelle 6.1) nur eingeschränkt aussagekräftig.

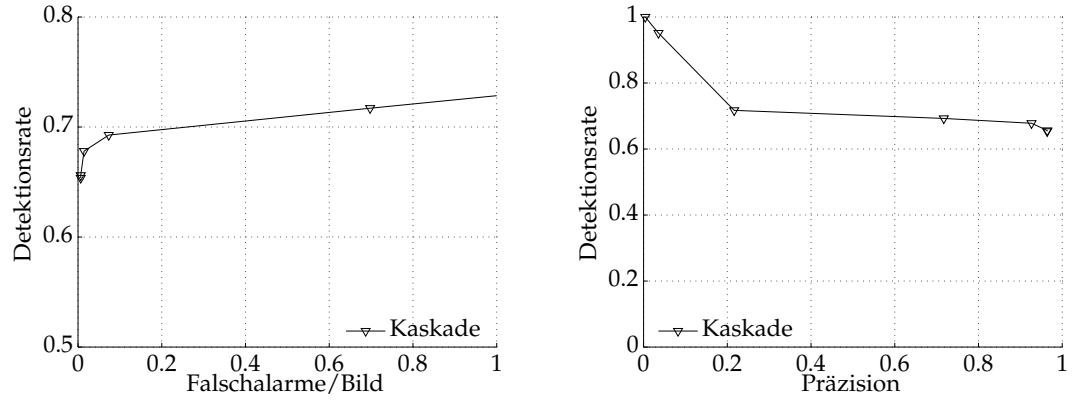


Abbildung 6.10.: ROC- und PR-Diagramm des Klassifikators für LKW-Rückfronten. Die Performance bei der Detektion von LKW-Rückfronten ist nicht so hoch wie bei der Detektion von PKW-Rückfronten, was sich allerdings mit der geringeren Anzahl Trainingsbeispiele erklären lässt [EG08].

Weaklearner ein. Zu beachten ist hierbei, dass bei einer Kaskade dieser Fall für *jede* akzeptierte Hypothese auftritt, beim DB-CBT hingegen auch Hypothesen mit kürzeren Laufzeiten akzeptiert werden.

Die erwarteten Laufzeiten der k-Means-Bäume sind leicht unter den erwarteten Laufzeiten der beiden DB-CBT. Zu beachten ist hierbei allerdings, dass die maximale Laufzeit des 4000-er DB-CBT 15 % geringer als die des 4000-er k-Means-Baums und 32 % geringer als die des 8000-er k-Means Baums ist, während der Vorteil der k-Means-Bäume bei der erwarteten Laufzeit lediglich 5 % bzw. 8 % beträgt. D. h. die Laufzeit des DB-CBT weist eine im Vergleich zum k-Means-Baum geringere Variabilität auf.

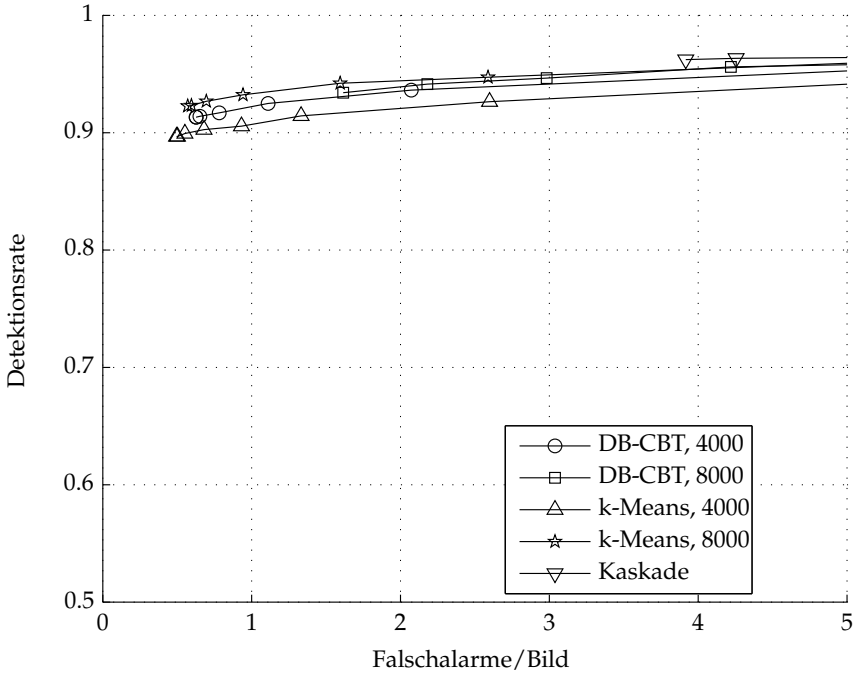


Abbildung 6.11.: ROC-Diagramm der Detektion von PKW ohne Unterscheidung zwischen Rückfront und Gesamtfahrzeug. Erwartungsgemäß ergibt sich eine höhere Detektionsrate, gepaart mit einer schlechteren Falschalarmrate, wenn die Unterscheidung zwischen Fahrzeug und Rückfront aufgehoben wird.

Kennwert	Kaskade	DB-CBT, 8000	k-Means, 8000
Erwartete Laufzeit (inkl. Feature-Sharing)	29.93	23.15	21.61
Vergleich zu Kaskade	100 %	77 %	72 %
Erwartete Laufzeit (exkl. Feature-Sharing)	170.55	113.33	100.36
Vergleich zu Kaskade	100 %	66 %	59 %
Maximale Laufzeit	3 637	1 724	1 600
Vergleich zu Kaskade	100 %	47 %	44 %

Tabelle 6.6.: Geschwindigkeitsvergleich zwischen Kaskade und den 8000-er Bäumen (PKW).

Für PKW-Rückfronten bestätigt sich die bereits bei der Untersuchung der Performance beobachtbare Tatsache, dass die Klassifikation von PKW ein komplexeres Problem ist als die Klassifikation von PKW-Rückfronten. Folglich werden für die Klassifikation von PKW-Rückfronten weniger Weaklearner benötigt – im Schnitt 21.67 gegenüber 22.36 bei Verwendung eines 4000-er DB-CBT.

Bei Verwendung eines DB-CBT ergibt sich ein erwarteter Laufzeitvorteil von 21 bzw. 19 % (4000-er bzw. 8000-er DB-CBT) gegenüber der Kaskade. Die erwartet Laufzeit bei Verwendung eines k-Means-Baums ist etwas geringer als die erwartete Laufzeit der DB-CBT. Auch hier ist jedoch festzuhalten, dass die maximale Laufzeit des 4000-er DB-CBT um 25 % bzw. 27 %

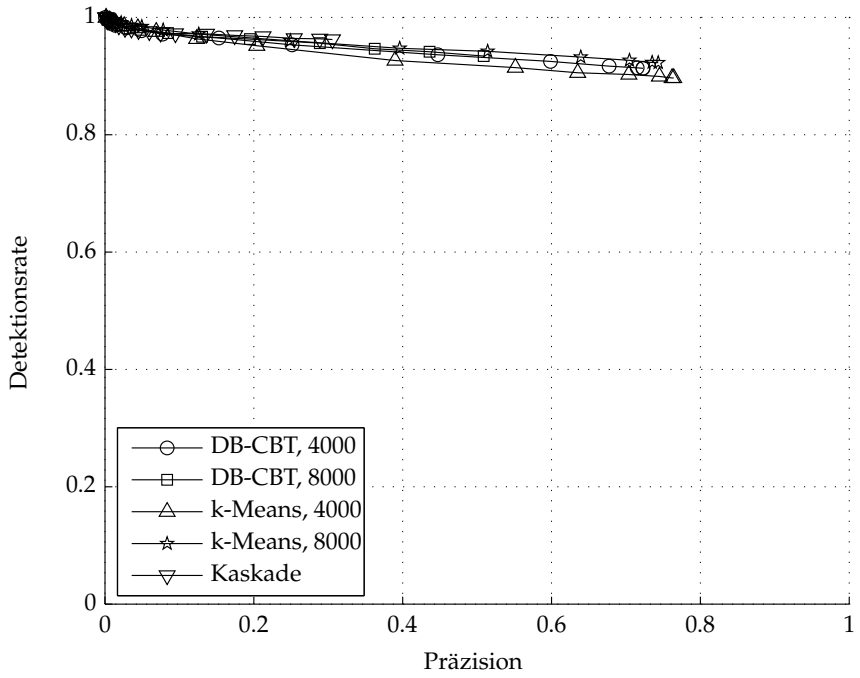


Abbildung 6.12.: PR-Diagramm der Detektion von PKW ohne Unterscheidung zwischen Rückfront und Gesamtfahrzeug. Die Präzision ist durch die höhere Falschalarmrate (vgl. Abb. 6.11) geringer als bei einer getrennten Detektion von PKW und PKW-Rückfront.

Kennwert	Kaskade	DB-CBT, 4000	k-Means, 4000
Erwartete Laufzeit (inkl. Feature-Sharing)	27.47	21.67	20.75
Vergleich zu Kaskade	100 %	79 %	76 %
Erwartete Laufzeit (exkl. Feature-Sharing)	149.74	100.88	93.11
Vergleich zu Kaskade	100 %	67 %	62 %
Maximale Laufzeit	3 381	1 204	1 514
Vergleich zu Kaskade	100 %	36 %	45 %

Tabelle 6.7.: Geschwindigkeitsvergleich zwischen Kaskade und den 4000**-er Bäumen (PKW-Rückfront).**

geringer ist als die maximale Laufzeit der k-Means-Bäume. Der Vorteil des DB-CBT gegenüber der Kaskade beträgt 64 bzw. 50 %.

Insgesamt ergibt sich in jedem Fall ein Geschwindigkeitsvorteil bei Einsatz des DB-CBT. Dieser beträgt (je nach Parametern) bis zu 25 % gegenüber der Kaskade. Neben den Vorteilen bei der erwarteten Laufzeit der Hypothesen ergibt sich ein Vorteil bei der maximalen Laufzeit: die maximale Laufzeit – und damit auch die bestmöglich zu garantierende Antwortzeit – ist beim DB-CBT um bis zu 64 % geringer.

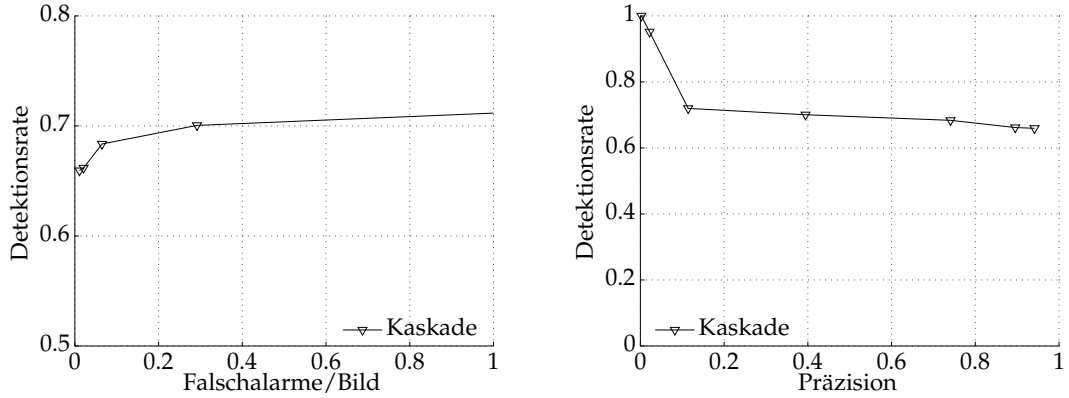


Abbildung 6.13.: Ergebnisse der Detektion von LKW ohne Unterscheidung zwischen Rückfront und Gesamtfahrzeug. (a) ROC-Diagramm. (b) PR-Diagramm.

Kennwert	Kaskade	DB-CBT, 8000	k-Means, 8000
Erwartete Laufzeit (inkl. Feature-Sharing)	27.47	22.30	21.25
Vergleich zu Kaskade	100 %	81 %	77 %
Erwartete Laufzeit (exkl. Feature-Sharing)	149.74	106.20	97.30
Vergleich zu Kaskade	100 %	71 %	65 %
Maximale Laufzeit	3 381	1 698	1 532
Vergleich zu Kaskade	100 %	50 %	45 %

Tabelle 6.8.: Geschwindigkeitsvergleich zwischen Kaskade und den 8000-er Bäumen (PKW-Rückfront).

6.3.3. Genauigkeit der Winkelschätzung

Neben der Performance und Geschwindigkeit der Klassifikatoren ist auch die Qualität der Kategorisierung in Winkelkategorien von Bedeutung. Um diese zu untersuchen, wurden alle assoziierten und korrekten Detektionen, also solche, die auch tatsächlich ein Objekt zeigen, dahingehend untersucht, wie stark die ihnen zugewiesene Winkelkategorie von der tatsächlichen abweicht. Zum Einsatz kam dabei das in Abschnitt 6.2.4 eingeführte Abstandsmaß. Unter Vorgabe einer maximalen Abweichung d_{max} kann damit der Anteil korrekt kategorisierter Detektionen an den kategorisierten Detektionen insgesamt betrachtet werden. Als korrekt kategorisiert gelten dabei alle Detektionen, deren Winkelkategorie höchstens um d_{max} von der tatsächlichen abweicht.

Tabelle 6.9 zeigt die Qualität der Winkelschätzung für PKW. Diese ist sehr hoch, alle Klassifikatoren kategorisieren schon bei $d_{max} = 0$ rund 28 % korrekt. Bei Einräumung einer Toleranz lässt sich der Korrektheitsgrad weiter steigern. Alle untersuchten Klassifikatoren zeigen in etwa die gleiche Leistung bei der Kategorisierung, keiner der Klassifikatoren kann hervorgehoben werden.

Tabelle 6.9 zeigt die Qualität der Winkelschätzung für LKW. Sie ist sehr viel höher als die der Winkelkategorisierung von PKW. Aufgrund der geringen Zahl an auswertbaren Detektionen

Klassifikator	$d_{max} = 0$	$d_{max} = 1$	$d_{max} = 2$	$d_{max} = 3$
DB-CBT, 4000	28 %	50 %	68 %	85 %
DB-CBT, 8000	28 %	48 %	68 %	86 %
k-Means, 4000	27 %	45 %	66 %	85 %
k-Means, 8000	28 %	47 %	69 %	89 %
Kaskade	28 %	49 %	69 %	85 %

Tabelle 6.9.: Qualität der Winkelkategorisierung (PKW). Angegeben ist jeweils der Anteil korrekt kategorisierter Detektionen an den kategorisierten Detektionen insgesamt.

Klassifikator	$d_{max} = 0$	$d_{max} = 1$	$d_{max} = 2$	$d_{max} = 3$
Kaskade	89 %	94 %	98 %	99 %

Tabelle 6.10.: Qualität der Winkelkategorisierung (LKW). Angegeben ist jeweils der Anteil korrekt kategorisierter Detektionen an den kategorisierten Detektionen insgesamt. Die Qualität der Winkelkategorisierung ist sehr viel höher als die der Winkelkategorisierung von PKW (Tabelle 6.9). Aufgrund der geringen Zahl an auswertbaren Detektionen – lediglich 265 Detektionen konnten Daten der Grundwahrheit zugeordnet werden – sind diese Zahlen allerdings nur eingeschränkt aussagekräftig.

– lediglich 265 Detektionen konnten Daten der Grundwahrheit zugeordnet werden – sind diese Zahlen allerdings nur eingeschränkt aussagekräftig und lediglich aus Gründen der Vollständigkeit angegeben.

6.3.4. Verwechslungsgefahr zwischen den Fahrzeugtypen

Tabelle 6.11 zeigt die Verwechslungsmatrix (engl. *confusion matrix*) der Klassifikatoren. Der Anteil an Verwechslungen zwischen den Fahrzeugtypen ist hervorgehoben. Für die Auswertung wird der Arbeitspunkt mit der geringsten Falschalarmrate jedes Klassifikators zugrunde gelegt.

Klassifikator	Anteil PKW	Anteil LKW	Falschalarme	Hypothesen
DB-CBT, 4000	94.3 %	**3.2 %**	2.5 %	48 981
DB-CBT, 8000	91.2 %	**5.6 %**	3.2 %	91 902
k-Means, 4000	92.4 %	**5.4 %**	2.2 %	45 454
k-Means, 8000	93.0 %	**5.2 %**	1.8 %	67 163
Kaskade (PKW)	90.3 %	**5.3 %**	4.4 %	183 045
Kaskade (LKW)	**20.6 %**	77.9 %	1.5 %	1 296

Tabelle 6.11.: Verwechslungsmatrix der Klassifikatoren. Der Anteil an Verwechslungen zwischen den Fahrzeugtypen ist hervorgehoben. Für die Auswertung wird der Arbeitspunkt mit der geringsten Falschalarmrate jedes Klassifikators zugrunde gelegt.

Es ist zu sehen, dass die Klassifikatoren für PKW keine Probleme mit Verwechslungen zeigen. Nur ein geringer Anteil der akzeptierten Hypothesen zeigt LKW statt PKW. Hervorzuheben

ist an dieser Stelle der 4000-er DB-CBT, der eine leicht höhere Zuverlässigkeit aufweist als die anderen Klassifikatoren. Insgesamt sind die Unterschiede zwischen den Klassifikatoren jedoch vernachlässigbar klein.

Der LKW-Klassifikator hingegen zeigt Probleme mit Verwechslungen zwischen der Fahrzeugtypen. Über 20 % der Detektionen zeigen PKW statt LKW.

6.4. Exemplarische Systemleistung

Die Abb. 6.14 und 6.15 zeigen Beispiele des kompletten Systems in der Phase der Anwendung. Als Klassifikator kam der 4000-er DB-CBT zum Einsatz. In Abb. 6.14(a) ist zu erkennen, dass sich die Objektfenster von PKW und LKW in der Höhe unterscheiden. Abb. 6.14(b) zeigt zwei korrekt detektierte PKW zusammen mit einem Falschalarm („Gullideckel“). Abb. 6.15 zeigt die im System vorgenommene Winkelschätzung durch Kategorisierung assoziierter Detektionen. Im Beispiel wurde dem PKW die Kategorie *right 13* zugewiesen.

(a) (b)

Abbildung 6.14.: Typische Szenen beim Einsatz des Gesamtsystems. Zum Einsatz kam der 4000-er DB-CBT. (a) Unterschiedliche Objektfenster für LKW und PKW. (b) Korrekt detektierte PKW zusammen mit einem Falschalarm.

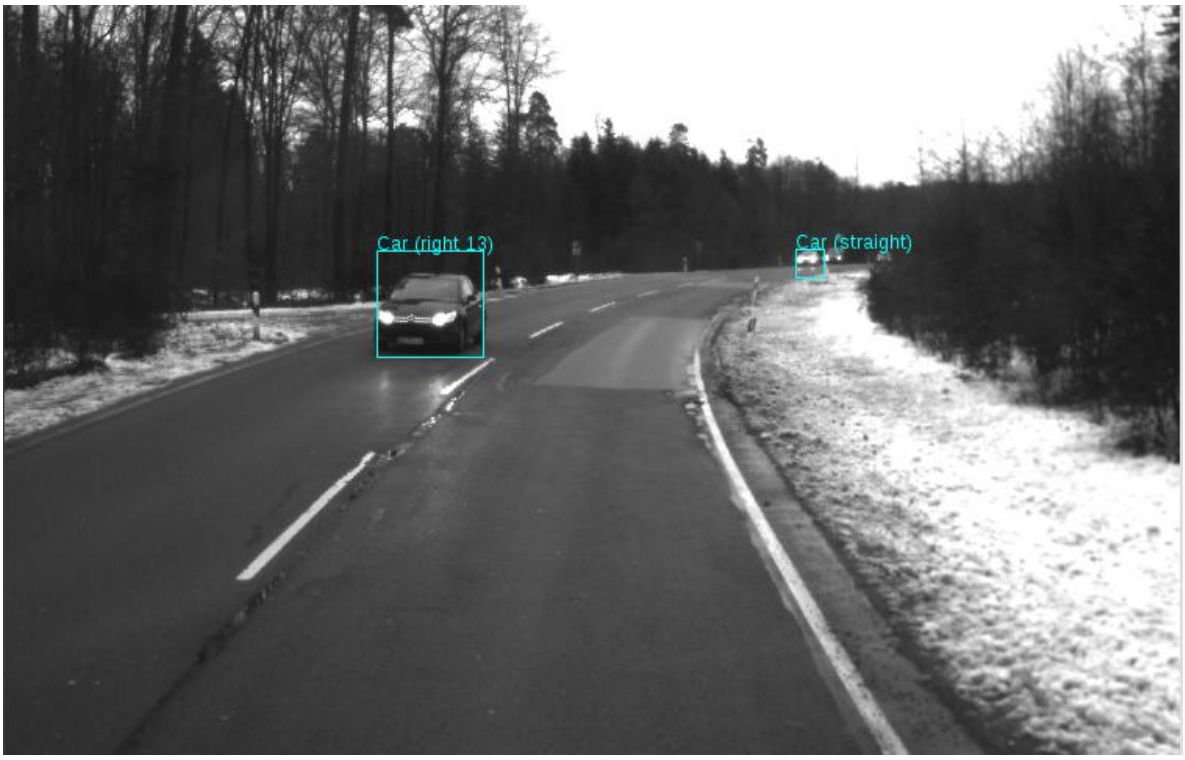


Abbildung 6.15.: Kategorisierung von Fahrzeugen. Zum Einsatz kam der 4000-er DB-CBT. Zu sehen ist eine vollständig assoziierte und kategorisierte Detektion eines PKW. Im Beispiel wurde der Detektion die Kategorie *right 13* zugewiesen.

KAPITEL 7

Zusammenfassung und Ausblick

Das Wissen um andere Verkehrsteilnehmer ist eine zentrale Komponente moderner Fahrerassistenzsysteme. Die Detektion von Fahrzeugen in Mono-Bildern ist hierbei ein wichtiger Baustein, der zur Weiterentwicklung von automobilen Assistenzsystemen beiträgt.

In dieser Arbeit wurde ein echtzeitfähiges System vorgestellt, das PKW und LKW in Mono-Grauwertbildern unter allen Orientierungen bzgl. der Kamera detektiert (*Multi-View Vehicle Detection*). Es arbeitet hierbei einzelbildbasiert, d. h. ohne Tracking über mehrere Bilder hinweg. Um sowohl PKW als auch LKW effizient zu klassifizieren, wird die Anzahl benötigter Merkmale durch Feature-Sharing stark reduziert sowie eine neuartige, hierarchische Klassifikatorstruktur eingesetzt.

Neben der reinen Detektion der Fahrzeuge wird zusätzlich ihre Orientierung eingeschätzt. Hierfür wurde der Zusammenhang zwischen Fahrzeugrückfront und Gesamtfahrzeug genutzt, welcher Rückschlüsse über die Orientierung des Fahrzeugs erlaubt. Daher werden Fahrzeugrückfront und Gesamtfahrzeug getrennt detektiert. Im Anschluss daran werden Detektionen einander zugeordnet und aus ihrer relativen Lage auf die Orientierung des Fahrzeugs geschlossen. Die so gewonnene Information lässt sich für zukünftige Verbesserungen von Tracking und Sensordatenfusion in Mono-Systemen nutzen und ermöglicht somit in Zukunft eine genauere und zuverlässigere Gefährdungseinschätzung im Straßenverkehr.

Um Fahrzeuge unter allen Orientierungen effizient detektieren zu können, wurde die Kaskade auf einen hierarchischen Klassifikator erweitert. Der entwickelte Klassifikator (*Decision-Boosted Cluster Boosted Tree*, DB-CBT) nimmt zusätzliche Auftrennungen des Eingaberaums vor, um eine effizientere Klassifikation zu ermöglichen. Hierfür wurde ein neues Gütemaß entwickelt, das die Eignung einer Auftrennung misst. Anhand diesem wird die Trennung mittels dem heuristischen Optimierungsverfahren *Simulated Annealing* so gewählt, dass das Klassifikationsproblem möglichst einfach wird.

Für die Entscheidung, wann eine Auftrennung des Eingaberaums erfolgen soll, wurde der Bhattacharyya-Koeffizient, ein eng mit dem Boosting zusammenhängendes Maß, verwendet. Dieser wird genutzt, um die Schwere des Klassifikationsproblems zu bewerten. Durch Schwellwertbildung ist es möglich, die Auftrennung des Eingaberaums nur dann durchzuführen,

wenn das Klassifikationsproblem schwierig zu lösen ist und eine Auftrennung mit hoher Wahrscheinlichkeit eine Vereinfachung verspricht.

Um die Anzahl zu traversierender Knoten zusätzlich zu reduzieren, wurde ein zweistufiges Verfahren vorgestellt, bei dem nach dem eigentlichen Boosting zur Trennung von Objekt und Hintergrund ein zweites Boosting durchgeführt wird. Dieses hat das Ziel, die gewonnene Auftrennung des Eingaberaums zu erlernen.

Die Leistungsfähigkeit des vorgestellten Konzepts konnte durch experimentelle Untersuchungen nachgewiesen werden. Die Detektionsleistung eines DB-CBT ist vergleichbar mit der Detektionsleistung einer Kaskade. Bis zu einer Entfernung von 100 Metern ist es mit dem DB-CBT möglich, mit einer Detektionsrate von 85 % bei weniger als 0.3 Falschalarmen je Bild die Rückfronten von PKW zu detektieren. Komplette PKW können mit einer Detektionsrate von über 80 % bei weniger als 0.3 Falschalarmen je Bild detektiert werden. Bei Einsatz des DB-CBT ergibt sich allerdings – bei vergleichbarer Detektionsleistung – ein Laufzeitvorteil gegenüber des Kaskade: im Schnitt müssen 25 % weniger Weaklearner je Hypothese ausgewertet werden; die worst-case-Laufzeit des DB-CBT ist sogar um 64 % geringer als die der Kaskade.

Die Schätzung der Orientierung durch das realisierte System ist sehr zuverlässig. Abhängig von der maximalen Toleranz werden zwischen 28 % (keine Toleranz) und 86 % (Toleranz von 3) der detektierten Fahrzeug korrekt kategorisiert.

Insgesamt gesehen konnte duch den entwickelten DB-CBT eine Verbesserung der Kaskade erreicht werden. Bei vergleichbarer Detektionsleistung ist die Geschwindigkeit deutlich höher. Durch das vorgestellte Konzept ist es zudem möglich, die Orientierung von Fahrzeugen mit hoher Qualität zu schätzen.

7.1. Ausblick

Aus dieser Arbeit lassen sich eine Reihe neuer Fragestellungen ableiten. Diese bieten sich als Ausgangspunkt für mögliche Folgearbeiten an. Sie lassen sich unterscheiden bzgl. dem DB-CBT als entwickelter *Methode* der Klassifikation und dem Gesamtsystem, welches eine *Funktion* – die Schätzung der Orientierung detektierter Fahrzeuge – erfüllt. Für beide gibt es verschiedene offene Fragen, die durch weitere Untersuchungen geklärt werden könnten. Ferner traten im Zuge der Evaluierung der Klassifikationsbäume Möglichkeiten zur Weiterentwicklung auf.

Weiterführende Untersuchungen des DB-CBT

Ein Problem bei Klassifikationsbäumen ist die erhöhte Gefahr der Überadaption. In dieser Arbeit wird dieser Gefahr durch Vorgabe einer Mindestanzahl an Positivbeispielen begegnet (siehe Abschnitt 5.2.2, Seite 65). In der Literatur sind mehrere alternative Ansätze beschrieben, z. B. der χ^2-Test. Diese könnten auf ihre Eignung für den DB-CBT untersucht werden. Als Ausgangspunkt für eine solche Untersuchung sei auf [DHS00, S. 402 ff.] verwiesen.

Die Entscheidungsstrategie des DB-CBT ist exklusiv, d. h. Hypothesen werden an maximal einen Nachfolger weitergereicht (siehe Abschnitte 4.2.2 und 5.4, Seiten 55 und 68). Da für die Entscheidung ein RealAdaBoost-Stronglearner zum Einsatz kommt, enthält die Aussage des „Entscheiders“ bereits die Sicherheit der Entscheidung. Damit ist eine Erweiterung auf eine kombinierte Entscheidungsstrategie leicht möglich. Hierfür muss lediglich ein Schwellwert ϵ definiert werden, welcher die notwendige Sicherheit für das Treffen einer exklusiven Entschei-

dung ausdrückt. Eine Hypothese wird nur dann an lediglich einen Nachfolger weitergereicht, wenn die Aktivierung um mindestens ϵ von der Entscheidungsschwelle abweicht – der trainierte Klassifikator also sicher bei seiner Entscheidung ist. Dieses Vorgehen wurde bereits in [Tu05] erfolgreich eingesetzt, es bietet sich folglich auch für den DB-CBT an, diese Strategie zu evaluieren.

Außerdem kann untersucht werden, ob und wie sich alternative Verfahren für die Optimierung der Zerlegung einsetzen lassen. Neben anderen heuristischen Optimierungsverfahren (genetische Algorithmen etc.) könnten hierbei auch Verfahren der diskreten Optimierung untersucht werden. Bei Nachweis der Konvexität von $e(.)$ gelänge es in diesem Fall, effizient die *optimale* Lösung, d. h. die bzgl. $e(.)$ beste Zerlegung in zwei Teilmengen zu erhalten.

Weiterführende Untersuchungen des Gesamtsystems

Auf Seiten des Gesamtsystems bietet sich zur Verbesserung der Performance und Winkelkategorisierung ein Tracking der Detektionen an. Hierbei muss untersucht werden, wie groß die nötige Vorlaufzeit bei Einsatz eines Trackers ist und welche Verbesserung der Performance und Winkelkategorisierung sich dadurch erreichen lassen.

Zudem kann der örtliche Zusammenhang zwischen Rückfront und Gesamtfahrzeug besser genutzt werden. In dieser Arbeit wurden Rückfront und Gesamtfahrzeug vollkommen unabhängig voneinander detektiert. Als potentielle Weiterentwicklung bietet sich an, den Detektionsschritt des Systems (siehe Abb. 1.4, Seite 18) in zwei Stufen durchzuführen. Zuerst wird das Gesamtfahrzeug detektiert, im Anschluss daran wird *innerhalb* der ersten Detektion die zugehörige Rückfront gesucht. Da bei einem solchen Aufbau der Rückfronten-Klassifikator sehr viel genauer trainiert werden könnte – als Negativbeispiel kommt alles in Frage, was nicht exakt die Rückfront, aber immer noch ein Fahrzeug zeigt – könnte eine höhere Genauigkeit der Winkelkategorisierung möglich sein.

Effiziente Evaluierung hierarchischer Klassifikatoren

In dieser Arbeit wurden ROC- und PR-Diagramme für Klassifikationsbäume erzeugt, indem schrittweise Stufen deaktiviert wurden (siehe Abschnitt 6.2.3, Seite 80). Dieses Vorgehen ist nicht optimal, da hierbei mehrere Knoten des Baums auf einmal deaktiviert werden. Durch bestmögliche Wahl der Reihenfolge, in der die Knoten einer Stufe deaktiviert werden, ließe sich die Performance eines hierarchischen Klassifikators genauer erfassen. Hierbei könnte untersucht werden, ob und wie es effizient möglich ist, die optimale Reihenfolge zu bestimmen.

Eine Alternative könnte an dieser Stelle der Einsatz von Rückschlusswahrscheinlichkeiten sein. Durch diesen Ansatz können ROC- und PR-Diagramme effizient generiert werden, indem die Wahrscheinlichkeitsschwelle einer Detektion sukzessive abgesenkt wird. Ausgangsbasis für weitere Untersuchungen sind [Tu05] und [SHL07].

ANHANG A

Struktur der erstellten Klassifikatoren

In diesem Anhang wird die Struktur der untersuchten Klassifikatoren visualisiert. Abb. A.1 zeigt den 4000-er DB-CBT, Abb. A.2 den 4000-er k-Means-Baum. Die 8000-er Bäume sind in Abb. A.3 (DB-CBT) bzw. Abb. A.4 (k-Means-Baum) dargestellt. Abb. A.5 zeigt die untersuchte Kaskade. Jeder Klassifikator ist inklusive der gemeinsamen Grundstruktur (vgl. 6.1.2) dargestellt.

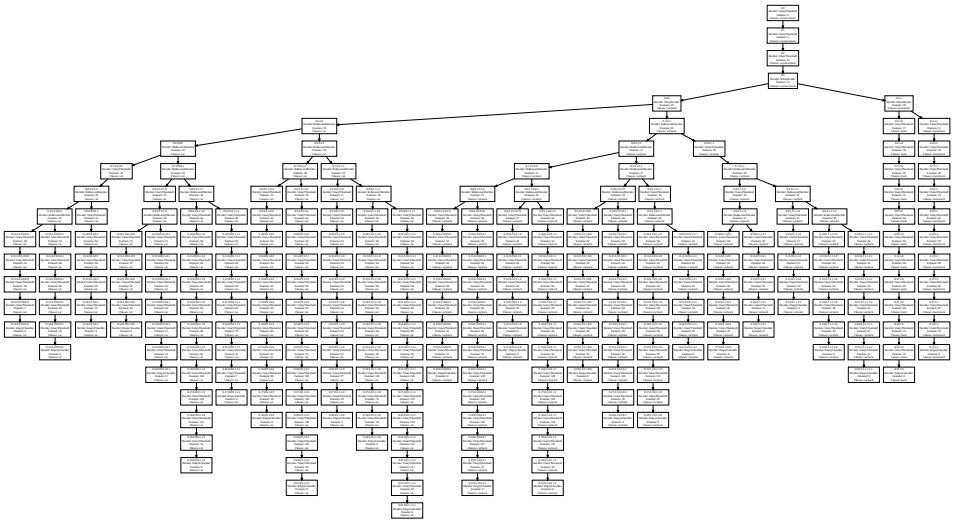


Abbildung A.1.: 4000-er DB-CBT.

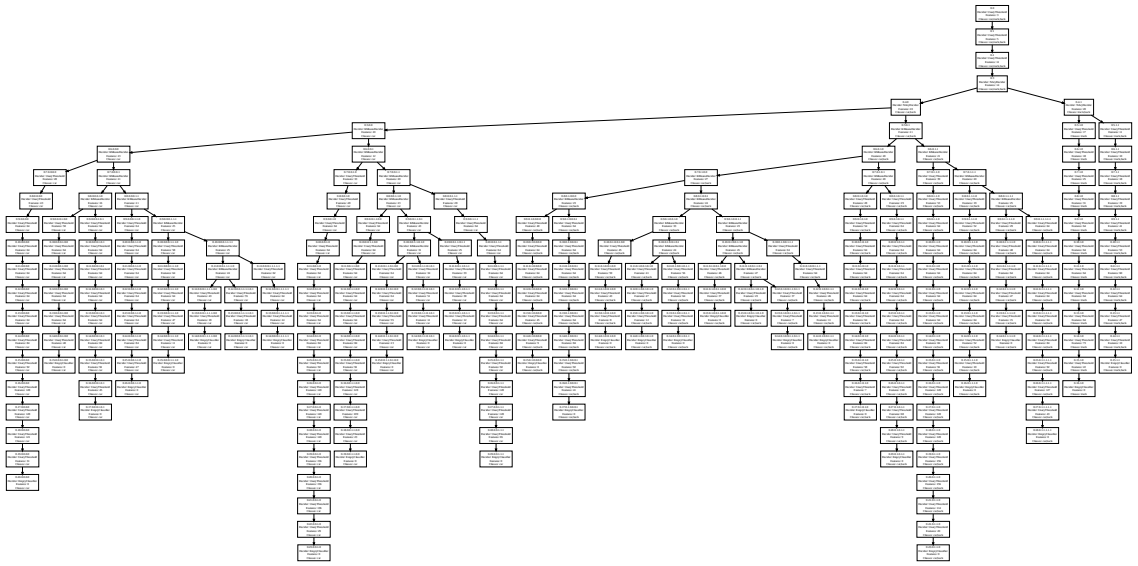


Abbildung A.2.: 4000-er k-Means-Baum.

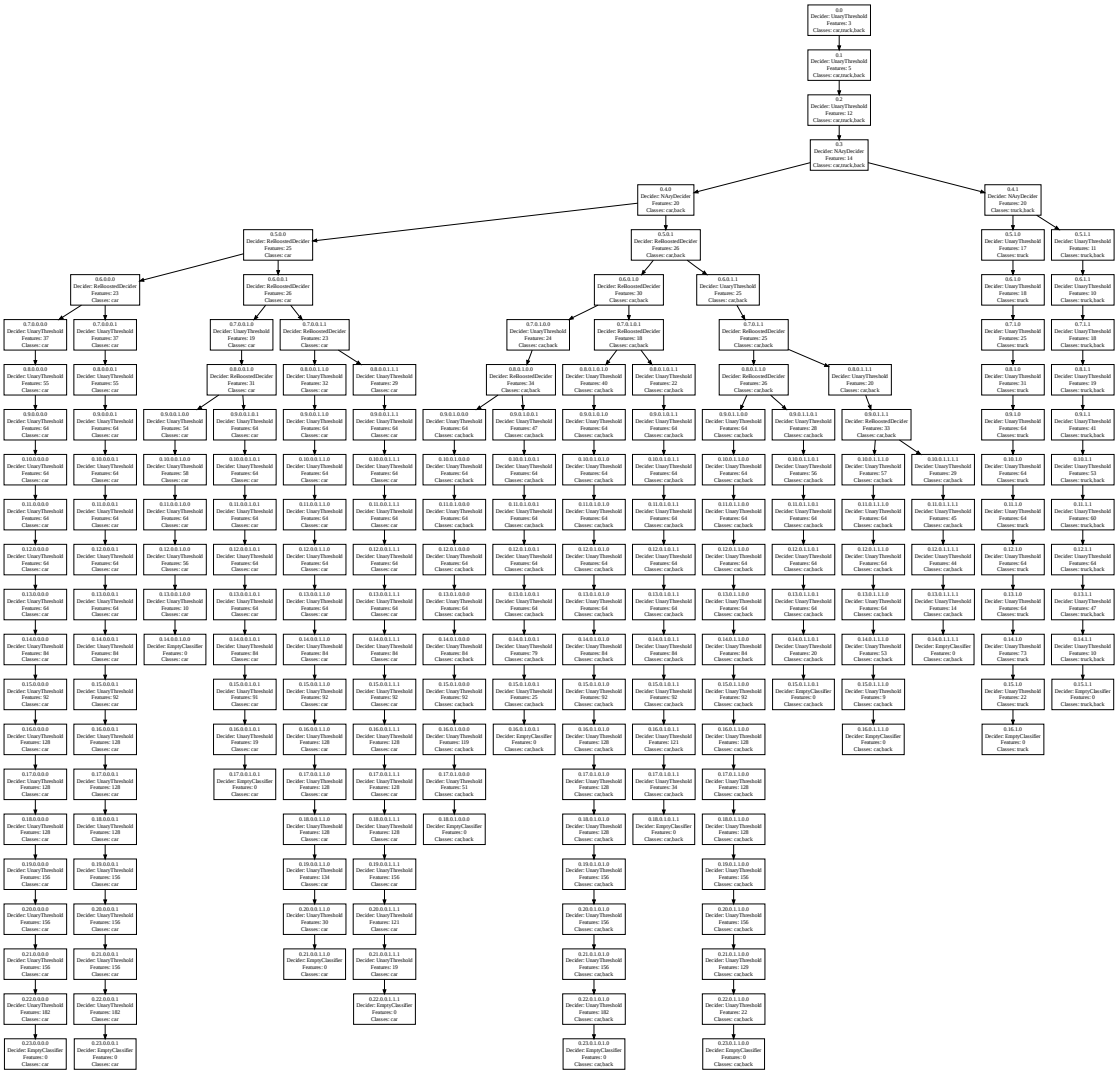


Abbildung A.3.: 8000-er DB-CBT.

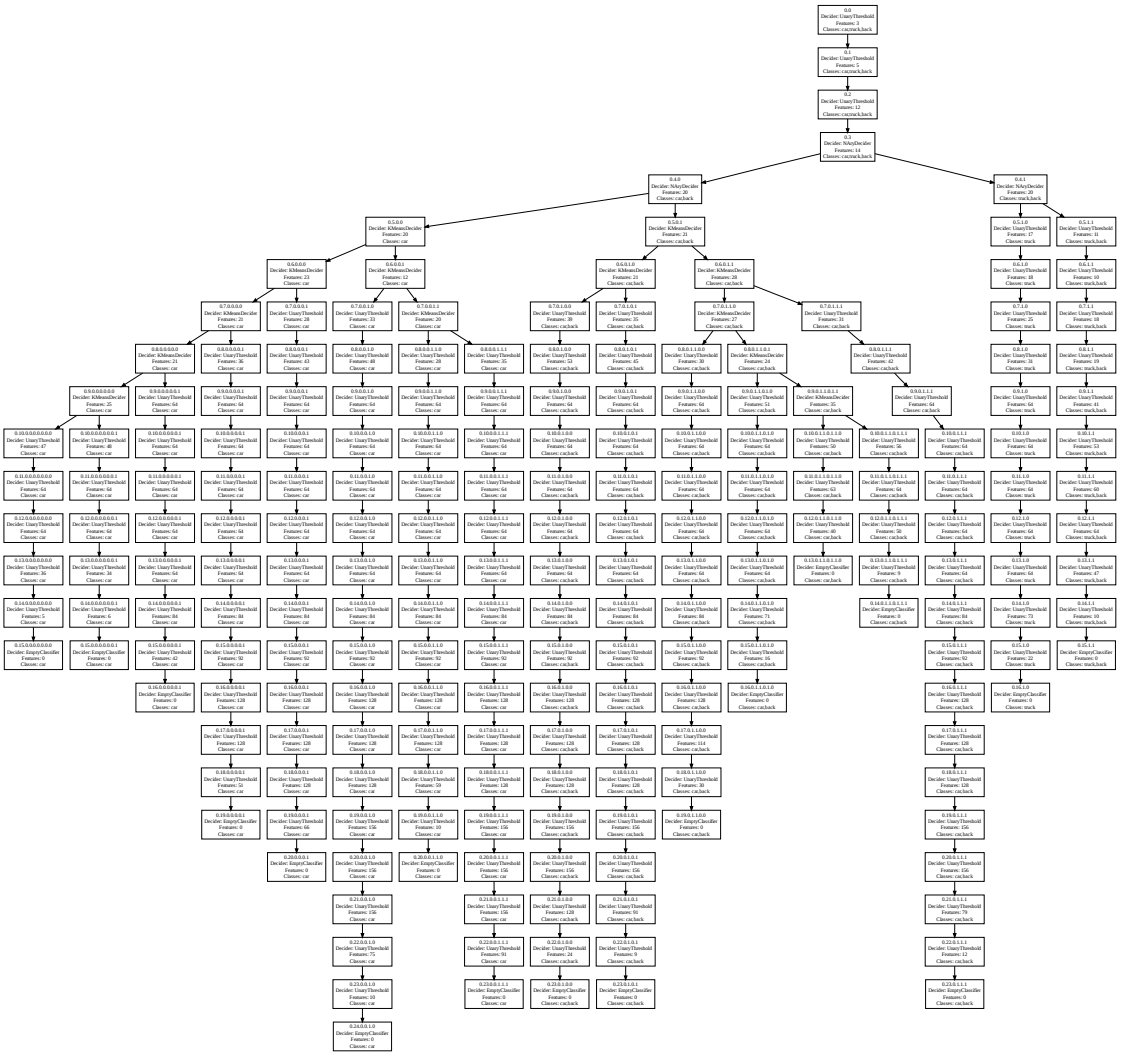


Abbildung A.4.: 8000-er k-Means-Baum.

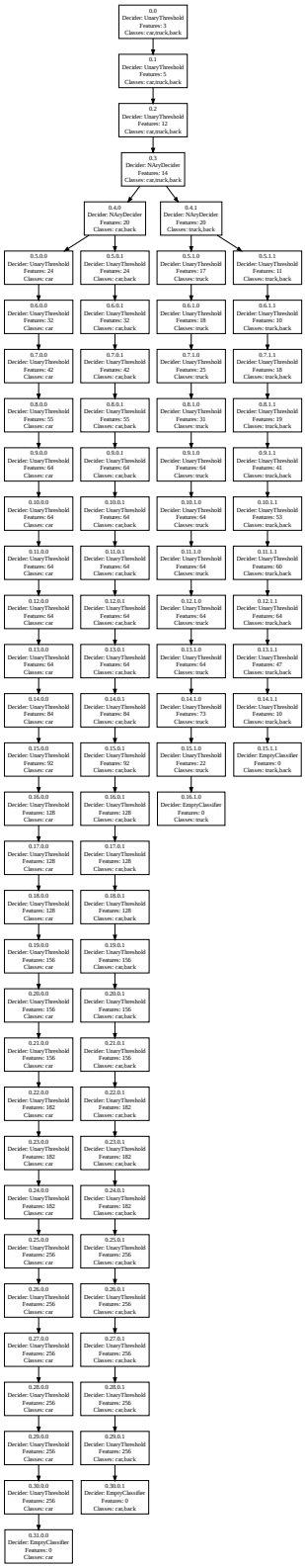


Abbildung A.5.: Kaskade.

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

Literaturverzeichnis

[Ape05] N. Apel. *Kaskadierte Komponenten-Klassifikatoren*. Abschlussarbeit, TU Ilmenau, 2005. (Zitiert auf den Seiten 38 und 40)

[ASS01] E. L. Allwein, R. E. Schapire, Y. Singer. Reducing Multiclass to Binary: a Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. (Zitiert auf Seite 51)

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, R. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. (Zitiert auf den Seiten 15, 52 und 55)

[Bha43] A. Bhattacharyya. On a Measure of Divergence Between Two Statistical Populations Defined by Their Probability Distributions. In *Bulletin of the Calcutta Mathematical Society*, S. 99–109. 1943. (Zitiert auf den Seiten 63 und 64)

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2 Auflage, 2006. (Zitiert auf den Seiten 16, 37, 51 und 52)

[BK08] G. Bradski, A. Kaehler. *Learning OpenCV*. O'Reilly, 2008. (Zitiert auf den Seiten 16, 48, 49 und 51)

[Chr07] C. Christmann. *Lokal-Adaptive Boosting-Trees*. Abschlussarbeit, Universität Ulm, 2007. (Zitiert auf den Seiten 14, 15, 58 und 62)

[CRM00] D. Comaniciu, V. Ramesh, P. Meer. Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *Proc. of Computer Vision and Pattern Recognition*, S. 142–149. 2000. (Zitiert auf Seite 67)

[DHS00] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2. Auflage, 2000. (Zitiert auf den Seiten 47, 51, 52, 65, 67 und 96)

[DSG90] A. Djouadi, O. Snorrason, F. D. Garber. The Quality of Training-Sample Estimates of the Bhattacharyya Coefficient. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):92–97, 1990. (Zitiert auf Seite 64)

[DT05] N. Dalal, B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Proc. of Computer Vision and Pattern Recognition*, S. 886–893. 2005. (Zitiert auf Seite 41)

[EG08] M. Enzweiler, D. M. Gavrila. A mixed generative-discriminative framework for pedestrian classification. In *Proc. of IEEE Conf. Computer Vision and Pattern Recognition CVPR 2008*. 2008. (Zitiert auf den Seiten 82 und 87)

[EG09] M. Enzweiler, D. M. Gavrila. Monocular Pedestrian Detection: Survey and Experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009. (Zitiert auf den Seiten 16, 48 und 78)

[Faw06] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. (Zitiert auf Seite 80)

[FG01] F. Fleuret, D. Geman. Coarse-to-Fine Face Detection. *International Journal of Computer Vision*, 41(1-2):85–107, 2001. (Zitiert auf Seite 55)

[FHT00] J. Friedman, T. Hastie, R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 28(2):337–407, 2000. (Zitiert auf den Seiten 35, 36 und 38)

[FS95] Y. Freund, R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of Second European Conference on Computational Learning Theory*, S. 23–37. Springer-Verlag, London, UK, 1995. (Zitiert auf den Seiten 35 und 38)

[Gre08] M. Gressmann. *Training eines Kaskadenklassifikators zur Fußgängererkennung unter Verwendung von Active Learning und virtuellen Beispielen*. Abschlussarbeit, Universität Ulm, 2008. (Zitiert auf Seite 14)

[HALL05] C. Huang, H. Ai, Y. Li, S. Lao. Vector Boosting for Rotation Invariant Multi-View Face Detection. In *Proc. of IEEE International Conference on Computer Vision*, S. 446–453. 2005. (Zitiert auf den Seiten 47 und 54)

[HALL07] C. Huang, H. Ai, Y. Li, S. Lao. High-Performance Rotation Invariant Multiview Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4):671–686, 2007. (Zitiert auf den Seiten 14, 37, 47, 54, 57, 58 und 74)

[HAWL04] C. Huang, H. Ai, B. Wu, S. Lao. Boosting Nested Cascade Detector for Multi-View Face Detection. In *Proc. of International Conference on Pattern Recognition*, S. 415–418. 2004. (Zitiert auf den Seiten 63 und 64)

[Jä05] B. Jähne. *Digitale Bildverarbeitung*. Springer, 6. Auflage, 2005. (Zitiert auf Seite 40)

[JV03] M. Jones, P. Viola. Fast Multi-view Face Detection. In *Proc. of Computer Vision and Pattern Recognition*. 2003. (Zitiert auf den Seiten 14 und 54)

[Kai67] T. Kailath. The Divergence and Bhattacharyya Distance Measures in Signal Selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967. (Zitiert auf den Seiten 64 und 67)

[Kal05] I. Kallenbach. *Multiklassen-Detektion mit verketteten Kaskadenklassifikatoren*. Abschlussarbeit, Universität Ulm, 2005. (Zitiert auf den Seiten 17, 40, 42, 43, 44, 45, 46, 48, 49, 55, 56, 57, 75, 78 und 80)

[KN09] C.-H. Kuo, R. Nevatia. Robust multi-view car detection using unsupervised sub-categorization. In *Proc. of IEEE Workshop on Applications of Computer Vision*. 2009. (Zitiert auf den Seiten 14, 55, 56, 78 und 106)

[KSPL06] I. Kallenbach, R. Schweiger, G. Palm, O. Löhlein. Multi-class Object Detection in Vision Systems Using a Hierarchy of Cascaded Classifiers. In *Proc. of IEEE Intelligent Vehicles Symposium*, S. 383–387. 2006. (Zitiert auf den Seiten 17 und 44)

[Lö09] O. Löhlein. Vergleich von Überdeckungsmaß und Ähnlichkeitsmaß für die Bewertung und Zusammenfassung von Detektionen, 2009. Daimler AG, Abteilung GR/EAP. Nicht veröffentlicht. (Zitiert auf den Seiten 78 und 79)

[LKP03] R. Lienhart, A. Kuranov, V. Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In *DAGM 25th Pattern Recognition Symposium*, S. 297–304. 2003. (Zitiert auf Seite 40)

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. (Zitiert auf Seite 41)

[Lut01] M. Lutz. *Programming Python*. O'Reilly, 2. Auflage, 2001. (Zitiert auf Seite 16)

[LW04] K. Levi, Y. Weiss. Learning Object Detection from a Small Number of Examples: the Importance of Good Features. In *Proc. of IEEE International Conference on Computer Vision*. 2004. (Zitiert auf den Seiten 15, 39, 41 und 42)

[LZSZ02] S. Z. Li, Z. Zhang, H.-Y. Shum, H. Zhang. FloatBoost Learning for Classification. In *Advances in Neural Information Processing Systems 15*, S. 993–1000. 2002. (Zitiert auf Seite 38)

[LZZ+02] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, H.-Y. Shum. Statistical Learning of Multi-View Face Detection. In *Proc. of 7th European Conference on Computer Vision*, S. 67–81. 2002. (Zitiert auf Seite 38)

[MKH05] T. Mita, T. Kaneko, O. Hori. Joint Haar-like Features for Face Detection. In *Proc. of International Conference on Computer Vision*, S. 1619–1626. 2005. (Zitiert auf Seite 38)

[MOL+05] M. Mählisch, M. Oberländer, O. Löhlein, D. Gavrilla, W. Ritter. A Multiple Detector Approach to Low-resolution FIR Pedestrian Recognition. In *Proc. of IEEE Intelligent Vehicles Symposium*. 2005. (Zitiert auf den Seiten 14 und 44)

[MR03] R. Meir, G. Rätsch. An Introduction to Boosting and Leveraging. In *Advanced Lectures on Machine Learning*, S. 118–183. Springer, 2003. (Zitiert auf den Seiten 35 und 36)

[Nem07] M. Nemec. *Time-delayed cascaded classifier*. Abschlussarbeit, Universität Ulm, 2007. (Zitiert auf den Seiten 22 und 23)

[OPS+97] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, T. Poggio. Pedestrian Detection Using Wavelet Templates. In *Proc. of Computer Vision and Pattern Recognition*, S. 193–199. 1997. (Zitiert auf Seite 40)

[POP98] C. P. Papageorgiou, M. Oren, T. Poggio. A General Framework for Object Detection. In *Proc. of International Conference on Computer Vision*, S. 555–562. 1998. (Zitiert auf Seite 40)

[Por05] F. Porikli. Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces. In *Proc. of Computer Vision and Pattern Recognition*, S. 829–836. 2005. (Zitiert auf Seite 42)

[PSR10] X. Perroton, M. Sturzel, M. Roux. Implicit Hierarchical Boosting for Mulit-view Object Detection. In *Proc. of Computer Vision and Pattern Recognition*. 2010. (Zitiert auf den Seiten 14, 53, 57, 61 und 62)

[Qui86] J. R. Quinlan. Introduction of decision trees. *Machine Learning*, 1(1):81–106, 1986. (Zitiert auf Seite 52)

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. (Zitiert auf Seite 52)

[RPP06] B. Rasolzadeh, L. Petersson, N. Pettersson. Response Binning: Improved Weak Classifiers for Boosting. In *Proc. of IEEE Intelligent Vehicles Symposium*, S. 344–349. 2006. (Zitiert auf Seite 47)

[Sch01] U. Schöning. *Algorithmik*. Spektrum, 2001. (Zitiert auf Seite 36)

[Sch02] R. E. Schapire. The Boosting Approach to Machine Learning: An Overview. In *Proc. of MSRI Workshop on Nonlinear Estimation and Classification*. 2002. (Zitiert auf den Seiten 35 und 37)

[SHL07] R. Schweiger, H. Hamer, O. Löhlein. Determining Posterior Probabilities on the Basis of Cascaded Classifiers as used in Pedestrian Detection Systems. In *Proc. of IEEE Intelligent Vehicles Symposium*, S. 1284–1289. 2007. (Zitiert auf den Seiten 78 und 97)

[SS99] R. E. Schapire, Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37:297–336, 1999. (Zitiert auf den Seiten 18, 35, 37, 38 und 39)

[Sta09] *Statistisches Jahrbuch 2009*. Statistisches Bundesamt, 2009. (Zitiert auf Seite 13)

[Sti07] C. Stiller. Intelligente Fahrzeuge - Technik, Chancen und Grenzen. In *Proc. of Autonome Mobile Systeme*, Informatik aktuell, S. 163–170. Springer Berlin, Heidelberg, 2007. (Zitiert auf Seite 13)

[Str00] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, 3. Auflage, 2000. (Zitiert auf Seite 16)

[SZ06] J. Schäuffele, T. Zurawka. *Automotive Software Engineering*. Vieweg, 3. Auflage, 2006. (Zitiert auf den Seiten 14, 81 und 85)

[TMF07] A. Torralba, K. P. Murphy, W. T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007. (Zitiert auf den Seiten 43 und 75)

[Tu05] Z. Tu. Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In *Proc. of IEEE International Conference on Computer Vision*, S. 1589–1596. 2005. (Zitiert auf den Seiten 15, 66 und 97)

[VJ01] P. Viola, M. Jones. Robust Real-time Object Detection. In *Proc. of IEEE Workshop on Statistical and Computational Theories of Vision*. 2001. (Zitiert auf den Seiten 14, 15, 17, 39, 40, 44, 45, 46 und 53)

[WAHL04] B. Wu, H. Ai, C. Huang, S. Lao. Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboost. In *Proc. of IEEE International Automatic Face and Gesture Recognition Conference*, S. 79–84. 2004. (Zitiert auf Seite 38)

[Wal08] L. Walchshäusl. *Maschinelle Erkennung von Verkehrsteilnehmern mittels heterogener Sensorik*. Dissertation, TU München, 2008. (Zitiert auf den Seiten 13, 41, 42, 80 und 81)

[Web02] A. R. Webb. *Statistical Pattern Recognition*. John Wiley & Sons, 2. Auflage, 2002. (Zitiert auf Seite 35)

[Wen03] S. Wender. *Insassendetektion mit kaskadierten Klassifikatoren*. Abschlussarbeit, TU Ilmenau, 2003. (Zitiert auf den Seiten 14, 15, 18, 37, 53 und 55)

[Wen08] S. Wender. *Multisensorsystem zur erweiterten Fahrzeugumfelderfassung*. Dissertation, Universität Ulm, 2008. (Zitiert auf Seite 13)

[WF05] I. H. Witten, E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques*. Elsevier, Morgan Kaufmann, 2 Auflage, 2005. (Zitiert auf Seite 80)

[WL04] S. Wender, O. Löhlein. A Cascade Detector Approach Applied to Vehicle Occupant Monitoring with an Omnidirectional Camera. In *Proc. of IEEE Intelligent Vehicles Symposium*, S. 345–350. 2004. (Zitiert auf den Seiten 14, 18, 44 und 53)

[WN07] B. Wu, R. Nevatia. Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In *Proc. of IEEE International Conference on Computer Vision*. 2007. (Zitiert auf den Seiten 15, 18, 39, 47, 53, 55, 57, 58, 61, 62, 63 und 73)

[Wu08] B. Wu. *Part based Object Detection, Segmentation, and Tracking by Boosting Simple Feature based Weak Classifiers*. Dissertation, University of Southern California, 2008. (Zitiert auf den Seiten 14, 18, 52 und 66)

[YHN09] B. Yang, C. Huang, R. Nevatia. Extensive Articulated Human Detection by Voting Cluster Boosted Tree. In *Proc. of IEEE Workshop on Applications of Computer Vision*. 2009. (Zitiert auf den Seiten 18, 47, 55, 61, 62, 66, 67 und 73)

[ZZ09] C. Zhang, Z. Zhang. Winner-Take-All Multiple Category Boosting for Multi-view Face Detection. Tech Report MSR-TR-2009-190, Microsoft Research, 2009. (Zitiert auf Seite 57)

[ZZ10] C. Zhang, Z. Zhang. A Survey of Recent Advances in Face Detection. Tech Report MSR-TR-2010-66, Microsoft Research, 2010. (Zitiert auf den Seiten 14, 35, 41, 44, 46 und 53)

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Michael Gabb)