

FAHRZEUGKLASSIFIKATION UNTER VERWENDUNG DES ADA-BOOST ALGORITHMUS

Clemens Arth, Bakk. techn.

Betreuer & Begutachter:
Vertr. Prof. DI. Dr. Horst Bischof

Diplomarbeit am Institut für
Maschinelles Sehen und Darstellen,
Fakultät für Informatik



Technische Universität Graz
Erzherzog Johann Universität

August, 2004

ABSTRACT

An algorithm based on ADA-Boost is presented to perform classification of moving vehicles by their appearance. After starting with a system description used for data acquisition, different kinds of feature detectors based on Haar-Wavelets are discussed. A slightly modified version of the ADA-Boost algorithm is presented which selects the most discriminating features from a large pool. The classifier is trained on discriminating the four classes: CAR, TRUCK, MOTORBIKE, BUS. For this four class classification task a tree structure and a One-vs.-All structure are compared. These structures are evaluated extensively using different sets of parameters and different camera positions. The applicability of the algorithm is demonstrated by giving some results from the experimental analysis carried out on a huge set of real world data.

ZUSAMMENFASSUNG

Es wird ein Verfahren vorgestellt, welches auf dem ADA-Boost Algorithmus basiert und eine Klassifikation von Fahrzeugen anhand ihres Aussehens durchführt. Beginnend mit einer Beschreibung des Systems, welches zur Datenaufnahme verwendet wurde, werden unterschiedliche Arten von auf Haar-Wavelets basierenden Feature-detektoren diskutiert. Danach wird eine modifizierte Version des ADA-Boost Algorithmus präsentiert, welche aus einem Featurepool unter Einbeziehung einer Menge von Trainingsbeispielen die diskriminantesten Features auswählt und Unterschiede zwischen vier verschiedenen Klassen von Fahrzeugen lernt. Basierend auf den so trainierten Klassifikatoren werden zwei unterschiedliche Strukturen entworfen, die eine Kategorisierung der vier unterschiedlichen Fahrzeugtypen ermöglichen. Es folgt eine ausführliche experimentelle Evaluierung der Strukturen auf von unterschiedlichen Positionen aus aufgenommenen Fahrzeugbildern und eine Präsentation der Ergebnisse. Dabei wird die Einsatzmöglichkeit anhand einer Reihe von Experimenten auf einer definierten Testdatenbasis demonstriert.

DANKSAGUNG

Ich möchte mich hiermit bei meinem Betreuer und Begutachter Horst Bischof für seine Hilfe im Laufe der Arbeit an dieser Diplomarbeit und für seine Geduld bedanken. Weiters möchte ich Andreas Kröpfel danken, der mich ohne zu zögern freundlich im InfoA10-Team der Firma Frequentis aufgenommen hat und mir die Möglichkeit gegeben hat, diese Diplomarbeit zu schreiben. Mein Dank gilt ebenso Gustavo Fernandez für die interessanten und fruchtbaren Diskussionen im Rahmen der Entwicklung des Systems, und Thomas Weber, der einen großen Teil der Entwicklung und Implementierung des Systems übernommen und durchgeführt hat. Ebenfalls möchte ich Christopher Grunert von der BH Graz-Umgebung danken, der mir eine Betreuungsgenehmigung für ein an eine Autobahn angrenzendes Gelände zur Aufstellung der Kamera erteilt hat.

Meinem Bruder Gregor möchte ich dafür danken, dass er mir seine Video-Ausrüstung für die Aufnahme der Daten zur Verfügung gestellt hat. Mein besonderer Dank gilt meinen Eltern, die mir mein Studium finanziert und mich immer dabei unterstützt haben.

Meiner Freundin Tamara Riehl möchte ich für ihre Geduld danken, die sie während meines gesamten Studiums mit mir hatte, die mich immer wieder motiviert hat und die mich bis zuletzt beim Schreiben dieser Diplomarbeit unterstützt hat.

Inhaltsverzeichnis

Inhaltsverzeichnis	i
Tabellenverzeichnis	v
Abbildungsverzeichnis	vii
1 EINLEITUNG	1
1.1 Motivation	1
1.2 Aufgabenstellung	2
1.3 Kapitelübersicht	4
1.4 Mathematische Notation	5
2 ÜBERSICHT ÜBER ALGORITHMEN IM BEREICH DER DETEKTION UND KLASSIFIKATION VON FAHRZEUGEN	7
2.1 Allgemeines	7
2.2 Detection und Recognition Algorithmen	8
2.2.1 Knowledge Based	9
2.2.2 Feature Based	9
2.2.3 Model Based	11
2.2.4 Template Matching	11
2.2.5 Appearance Based	12

2.3	Motion Detection	14
2.3.1	Nicht-Rekursive BGM-Algorithmen	15
2.3.2	Rekursive BGM-Algorithmen	16
2.3.3	Shadow Modelling Algorithmen	17
2.4	Tracking Algorithmen	18
2.4.1	Der Kalman Tracker	18
2.4.2	Der Condensation Tracker	19
2.4.3	Der KLT Tracker	20
2.5	Zusammenfassung	21
3	SYSTEMARCHITEKTUR	23
3.1	Allgemeines	23
3.2	Vorraussetzungen und Randbedingungen	23
3.3	Systemstruktur	24
3.4	Das Real-World Modell	26
3.5	Das Bildaufnahme-Modul	26
3.6	Hintergrundmodell	28
3.7	Das Segmentierungs-Modul	29
3.8	Das Tracker-Modul	30
3.9	Das Klassifikations-Modul	33
3.10	Probleme beim Einsatz des Systems	33
4	FEATUREGENERIERUNG UND KLASSIFIKATORENTWURF	35
4.1	Allgemeines	35
4.2	Entwurf und Berechnung der verwendeten Features	35
4.2.1	Summed Area Table	37

4.2.2	Rotated Summed Area Table	38
4.2.3	Featuretypen und Interpretation	39
4.2.4	Datenpool und Selektion	41
4.3	Klassifikatorentwurf	43
4.3.1	Der ADA-Boost Algorithmus	43
4.3.2	Die Funktion der schwachen Klassifikatoren	45
4.4	Evaluierete Strukturen zur Multi-Class Klassifikation	47
4.4.1	Die <i>Baum</i> -Struktur	47
4.4.2	Die <i>1-vs-All</i> -Struktur	47
4.4.3	Vor- und Nachteile der einzelnen Strukturen	49
5	EXPERIMENTE UND RESULTATE	51
5.1	Allgemeines	51
5.2	Beschreibung der verwendeten Trainings- und Testdaten	51
5.3	Detailanalyse der Algorithmen	56
5.3.1	Aufteilung der Daten	56
5.3.2	Vergleich der Weak Classifiers	57
5.3.3	Performance in Abhängigkeit zur Anzahl verwendeter Featu- res (Level)	59
5.3.4	Performance in Abhängigkeit zur Größe der Trainingsdaten- menge	59
5.3.5	Gesamtperformance der einzelnen Strukturen	62
5.3.6	Sensitivität auf eine Veränderung des Blickwinkels	65
5.4	Untersuchung der Generalisierungsfähigkeit	66
5.4.1	Performance der Strukturen für Kameraposition 2	66
5.4.2	Performance der Strukturen für Kameraposition 3	69

5.5	Zusammenfassung und Schlussfolgerungen	71
6	ZUSAMMENFASSUNG UND AUSBLICK	75
6.1	Zusammenfassung	75
6.2	Ausblick	76
A	ANHANG	79
A.1	Der Kalman Tracker	79
A.2	Berechnung einer Homographie	82
	REFERENZEN	85

Tabellenverzeichnis

4.1	Featurepools von drei Bildern unterschiedlicher Größe	42
5.1	Anzahl aufgenommener Fahrzeuge verschiedenen Typs	55
5.2	Gegenüberstellung der benutzten und möglichen Anzahl an Features	55
5.3	Aufteilung Trainings-/Testdaten für die folgenden Experimente . . .	57
5.4	EER-Vergleich der Weak-Classifiers	58
5.5	EER-Vergleich auf verschiedenen Klassifikator-Levels	59
5.6	EER-Vergleich bei verschieden großen Trainingsdatenmengen	62
5.7	Anzahl korrekt klassifizierter Fahrzeuge	65
5.8	Gegenüberstellung der EER für Daten von Kameraposition 1 und 1*	65
5.9	Anzahl korrekt klass. Fahrzeuge auf Daten von Kameraposition 1* .	66
5.10	Aufteilung der Datenmenge für Kameraposition 2	67
5.11	Gegenüberstellung der EER	68
5.12	Anzahl korrekt klass. Fahrzeuge auf Daten von Kameraposition 2 .	68
5.13	Aufteilung der Datenmenge für Kameraposition 3	70
5.14	Gegenüberstellung der EER	70
5.15	Anzahl korrekt klass. Fahrzeuge auf Daten von Kameraposition 2 .	71
5.16	<i>Confusion Matrix</i> des empfohlenen Systems	74

Abbildungsverzeichnis

1.1	Systemübersicht und Aufgabenverteilung	3
2.1	Beispiel für den Algorithmus nach [FPZ03]	10
2.2	Beispiele für die Detektion eines Fahrzeugs nach [KDN93]	12
2.3	Beispiel der Kette binärer Klassifikatoren in [VJ01]	14
2.4	Beispiel der Schattendetektion in [PMGT01]	18
3.1	Modularer Aufbau des Systems zur Datenextraktion	25
3.2	Beispiel einer <i>Region Of Interest</i>	27
3.3	Beispiel für ein Differenzbild	28
3.4	Beispiel für ein <i>Growing Grid</i>	29
3.5	<i>Homographie</i> als Zusammenhang zwischen einem Bild und einem Modell der Szene	32
4.1	Lage eines Rechtecks	37
4.2	Berechnung der Pixelsumme eines Rechtecks	38
4.3	Berechnung des Rotierten Integral Images	40
4.4	Berechnung der Pixelsumme eines rotierten Rechtecks	40
4.5	Prototypen der 15 verwendeten Features	42
4.6	Modifizierter ADA-Boost Algorithmus	44
4.7	Beispiel für Schwellwertsetzung	46

4.8	Baum-Struktur zur Multi-Class Klassifikation	48
4.9	1-vs-All-Struktur zur Multi-Class Klassifikation	48
5.1	Drei ausgewählte Kamerapositionen	52
5.2	Beispiele aus der PKW Kategorie	53
5.3	Beispiele aus der BIKE Kategorie	53
5.4	Beispiele aus der LKW Kategorie	54
5.5	Beispiele aus der BUS Kategorie	54
5.6	Vergleich der Weak Classifiers	58
5.7	Performance bei unterschiedlichem Klassifikator-Level	60
5.8	Performance bei unterschiedlicher Anzahl an Trainingsdaten	61
5.9	ROC Kurven der Strukturen auf der Testdatenmenge	63
5.10	Featureselektion durch einzelne Klassifikatoren	64
5.11	Bilder falsch klassifizierter Fahrzeuge	64
5.12	Beispielbilder für Kameraposition 2	67
5.13	ROC Kurven für Klassifikatoren auf Daten von Kameraposition 2	68
5.14	Beispielbilder für Kameraposition 3	69
5.15	ROC Kurven für Klassifikatoren auf Daten von Kameraposition 3	70
A.1	Funktion des Kalman Filters	81

KAPITEL 1

Einleitung

1.1 Motivation

Mit der stetig zunehmenden Leistungsfähigkeit moderner Computer wird der Einsatz von Bildverarbeitungsalgorithmen zur Lösung alltäglicher Probleme immer populärer. Auch auf dem Gebiet der Verkehrsüberwachung werden zunehmend neue Techniken eingesetzt, die dabei die unterschiedlichsten Funktionen erfüllen.

In großen Einkaufszentren beispielsweise können solche Systeme zur automatisierten Parkraumüberwachung und als Parkleitsysteme eingesetzt werden. Videokameras filmen Parkplätze und Algorithmen registrieren ein- und ausparkende Fahrzeuge. Automatisch geschaltete Anzeigen steuern direkt oder indirekt den einfahrenden Verkehr zu freien Parkplätzen.

Ein großer Forschungsbereich ist auch der des Einsatzes von Videokameras und Verarbeitungsalgorithmen innerhalb von Fahrzeugen. In der Testphase befinden sich hier zurzeit Systeme, die LKW-Fahrer beim unbeabsichtigten Überfahren von seitlichen Leitlinien warnen, aber auch schon Systeme, die ein selbstständiges Fahren von Fahrzeugen ohne Zutun eines menschlichen Lenkers ermöglichen.

Das wohl größte Gebiet, auf welchem Bild- und Videoverarbeitungsalgorithmen heutzutage bereits erfolgreich eingesetzt werden, ist die Überwachung von Kreuzungen, Autobahnen, Tunnelabschnitten und wichtigen Verkehrsknotenpunkten.

Die Gründe für den massiven Einsatz von automatischen Systemen in diesem Bereich sind vor allem Sicherheitsaspekte, aber natürlich auch wirtschaftliche Gesichtspunkte. Mit dem steigendem Verkehrsaufkommen wächst die Gefahr von Unfällen und Geisterfahrern, gleichzeitig muss auch mehr in den Umgang mit diesen Proble-

men investiert werden und versucht werden, diese Risiken zu minimieren. Dies ist sowohl hinsichtlich der Sicherheit auf den Straßen wichtig, als auch in Bezug auf den zu minimierenden wirtschaftlichen Schaden, der durch Unfälle und Staus entstehen kann.

Ein dichteres Netz der Überwachung ist die Folge, damit verbunden ist eine regelrechte Datenflut, die ohne den Einsatz weiterer computertechnischer Hilfsmittel unmöglich bewältigt werden kann. So reicht das Spektrum der Applikationen von verhältnismäßig einfachen Systemen zur Zählung von Fahrzeugen über Anwendungen zur Unfall- und Stauerkennung bis hin zu komplexen Algorithmen zur Abstandsmessung, Geschwindigkeitsmessung, Geisterfahrererkennung, Mautverrechnung und vielem mehr.

Trotz des hohen Forschungsaufwands auf dem Gebiet der automatisierten Verkehrsüberwachung, lässt die Performance der meisten handelsüblichen Systeme sehr zu wünschen übrig. Das liegt aber nicht unbedingt daran, dass technische Ressourcen oder das Know-how fehlten, sondern ist vielmehr in der Komplexität der Aufgabenstellung selbst begründet. Da es sich hierbei um klassische *Outdoor*-Applikationen handelt, gilt es viele Probleme zu lösen, deren Existenz zumeist erst beim praktischen Einsatz eines Systems offensichtlich wird. Hier sei beispielsweise die Unterdrückung von unerwünschten Vibrationen einer Kamera genannt, welche durch starken Wind auf einem freistehenden Masten oder durch vorbeifahrende Schwerfahrzeuge in einem Tunnel hervorgerufen werden. Ein anderes Beispiel ist die korrekte Funktion bei unterschiedlichen Wetterbedingungen, sei es bei starker Sonneneinstrahlung oder, im Gegensatz dazu, bei absoluter Dunkelheit, sei es bei schwerem Regen, dichtem Nebel oder bei Schneefall.

In jedem Fall stellt der Entwurf eines solchen Systems für den Entwickler eine große Herausforderung dar. In letzter Zeit wurde eine Reihe neuer und besserer Algorithmen in der Bild- und Videoverarbeitung präsentiert. Durch die damit erzielten viel versprechenden Ergebnisse, wird deren Erprobung auch auf dem Gebiet der Verkehrsüberwachung motiviert.

1.2 Aufgabenstellung

Seit dem Fall des Eisernen Vorhangs zu Beginn der 90er Jahre steigt das Verkehrsaufkommen auf den Transitrouten durch Österreich unaufhörlich an. Mit dem Beitritt vieler osteuropäischer Länder zur Europäischen Union wird dieser Trend noch verstärkt. Um den damit verbundenen Problemen nicht nur durch den Bau breiterer

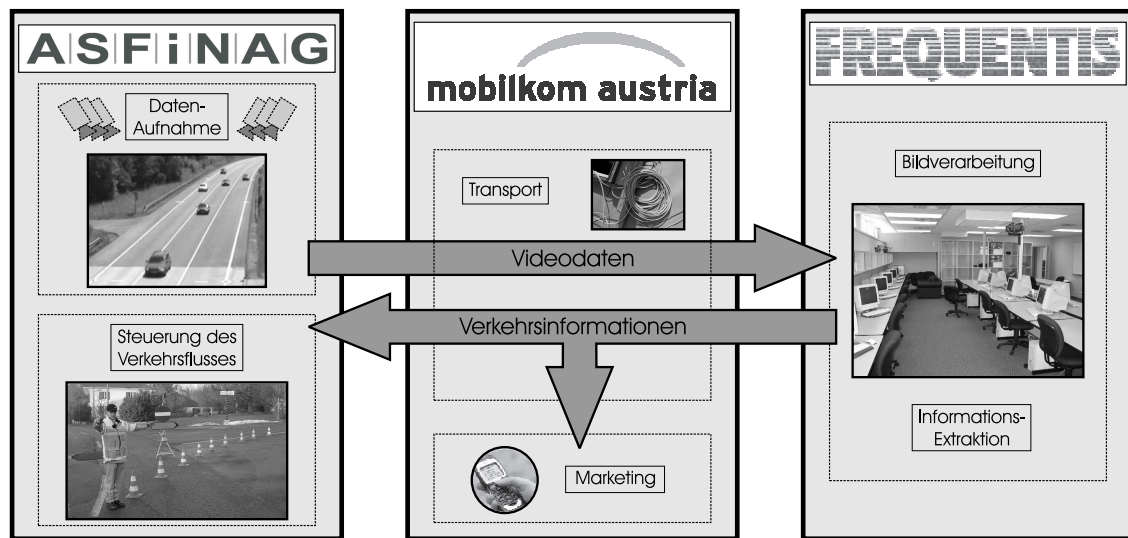


Abbildung 1.1: Übersicht über die Verteilung der Aufgabenbereiche im Rahmen des INFOA10 Projekts.

Straßen entgegenzutreten, initiiert die österreichische Autobahnen- und Schnellstraßen Finanzierungs-AG, kurz ASFINAG, in der letzten Zeit immer mehr Projekte, die dem Bereich der Verkehrstelematik zuzurechnen sind. Ziel ist es, eine Messung des Verkehrsflusses vornehmen und eine direkte oder indirekte Steuerung und Regelung des Verkehrsaufkommens auf den verschiedensten Routen durchführen zu können.

Die Firma FREQUENTIS entwickelt in Kooperation mit der Asfinag, der ÖSTERREICHISCHEN MOBILKOM und einigen anderen Firmen ein System zur Zählung von Fahrzeugen. Der Name dieses Projekts lautet INFOA10. Die Grundfunktion des Systems liegt dabei in der Messung des Verkehrsaufkommens auf einem überwachten Straßenabschnitt. Dabei verwendet das System einerseits von Laser-Sensoren aufgenommene Daten, andererseits durch den Einsatz von Bildverarbeitungsalgorithmen auf Videobildern gewonnene Informationen. Die Firma Frequentis ist dabei allein auf dem Gebiet des auf der Verarbeitung von Videobildern basierenden Teils tätig.

Neben der primären Funktion der Zählung von Fahrzeugen ist das vorrangige Ziel des INFOA10 Projekts, durch Ermittlung des prozentuellen Anteils des Schwerverkehrs am gesamten Verkehrsaufkommen eine durchschnittliche Reisezeitberechnung für einen definierten Streckenabschnitt durchführen zu können. Diese Information soll danach kommerziell genutzt werden und als Service einer breiten Schicht von Benutzern durch SMS-Versand oder Hotlines zugänglich gemacht werden.

Für den in dieser Diplomarbeit interessanten Teil des Projekts liegt im experimentellen Einsatz eine verteilte Struktur mit verschiedenen Verantwortungsbereichen

vor, die in Abbildung 1.1 zu sehen ist. Während die tatsächliche Positionierung der Kameras im Einflussbereich der Asfinag liegt, übernimmt die Frequentis die Verarbeitung der Bilddaten. Das Bindeglied dazwischen bildet die österreichische Mobilkom, welche den Transport des Bildmaterials und der daraus gewonnenen Informationen zwischen den einzelnen Institutionen durchführt. Ebenfalls in den Aufgabenbereich der Mobilkom fällt die weitere Vermarktung der gewonnenen Verkehrsinformationen.

Der Entwurf der verarbeitenden Einheit im Einflussbereich der Frequentis sieht den Einsatz eines Moduls vor, welches eine Klassifikation der Fahrzeuge vornehmen soll, um den Anteil des Schwerverkehrs am gesamten Verkehrsaufkommen bestimmen zu können. Der gesamte Entwurf und die anschließende Implementierung des Klassifikationsalgorithmus bilden, neben der Weiterentwicklung der bereits in den bestehenden Programmteilen eingesetzten Algorithmen, den Inhalt der vorliegenden Diplomarbeit.

Ausgehend von Bildern einzelner Fahrzeuge, die durch verschiedene Vorverarbeitungsschritte aus einer fortlaufenden Sequenz von Videobildern extrahiert werden, lautet die genaue Aufgabenstellung, einen Klassifikationsalgorithmus zu entwerfen, der in der Lage ist, Personenkraftwagen (PKWs) und Lastkraftwagen (LKWs) voneinander zu unterscheiden. Besonderes Augenmerk soll dabei auf den universellen Einsatz unter verschiedenen Aufnahmesituationen und Blickwinkeln gelegt werden.

Da sich die gesamte in Abbildung 1.1 gezeigte Struktur nach wie vor im Aufbaustadium befindet, standen zum Zeitpunkt der Entwicklung des Algorithmus keine benötigten Videodaten zur Verfügung. Deshalb wurden mit Hilfe eines handelsüblichen Camcorders an einem günstigen Aufstellungsort mehr als 50 Stunden Videomaterial aufgenommen und daraus Fahrzeugbilder mit Hilfe der bereits vorhandenen Programmteile extrahiert. Diese Daten wurden dann zur Lösung der gestellten Aufgabe verwendet.

1.3 Kapitelübersicht

In der Literatur finden sich Unmengen an Publikationen, die den Einsatz von Bildverarbeitungsalgorithmen im Bereich der Detektion und Klassifikation von Fahrzeugen zum Inhalt haben. In **Kapitel 2** wird versucht, eine grobe Einteilung in einzelne Gruppen vorzunehmen. Weiters werden Beispiele aus der Literatur angeführt, in denen die einzelnen Algorithmen verwendet wurden.

In **Kapitel 3** wird das System beschrieben, welches zur Extraktion der benötigten

Daten verwendet wurde. Nach einem Überblick über die Gesamtstruktur, werden die einzelnen Module im Detail thematisiert. Das Kapitel schließt mit einer Diskussion der Einsatzmöglichkeiten des Systems.

Kapitel 4 ist dem Kernthema dieser Diplomarbeit gewidmet. Einer detaillierten Erklärung der Methoden zur Featureberechnung folgt die Beschreibung des Entwurfs des Klassifikationsalgorithmus.

In **Kapitel 5** wird der entworfene Algorithmus im experimentellen Einsatz ausführlich evaluiert. Beginnend mit einer Beschreibung der verwendeten Daten, wird die Performance des Klassifikationsalgorithmus in mehreren Situationen untersucht.

Zusammenfassende Worte finden sich in **Kapitel 6**. Außerdem werden verschiedene Überlegungen angestellt, wie das System zur Datenextraktion, der entworfene Klassifikationsalgorithmus und die daraus gebildeten Strukturen verbessert werden könnten.

1.4 Mathematische Notation

Vektoren werden mit fetten Kleinbuchstaben notiert.

Matrizen werden mit fetten Großbuchstaben angeschrieben.

Partielle Ableitungen von Funktionen werden wahlweise in Indexform, wenn deutlich erkennbar ist, dass es sich um eine Ableitung handelt, oder in der ∂ -Schreibweise notiert. Das bedeutet, folgende Schreibweisen sind äquivalent:

$$\frac{\partial z}{\partial x} \Longleftrightarrow z_x \tag{1.1}$$

KAPITEL 2

Übersicht über Algorithmen im Bereich der Detektion und Klassifikation von Fahrzeugen

2.1 Allgemeines

Bereits Ende der 80er Jahre wurden erste Überlegungen angestellt, Bildverarbeitungsalgorithmen im Bereich der videobasierten Verkehrsüberwachung einzusetzen. Artikel von BOYCE [Boy88] oder BERGAN u.a. [BHKT88] sind Zeugen der damals futuristisch wirkenden Ideen und den großen Vorstellungen, die man von den Entwicklungen auf diesem Gebiet hatte. Mit der einsetzenden Massenproduktion immer leistungstärkerer Computer in den 90er Jahren wuchs die Menge der Einsatzmöglichkeiten von Bildverarbeitungsalgorithmen am Sektor der Verkehrsüberwachung enorm. Als Resultat davon gibt es heute eine unüberschaubare Zahl von Publikationen, die ausgehend von den verschiedensten Voraussetzungen und Zielsetzungen den experimentellen Einsatz von Bildverarbeitungsalgorithmen zum Verarbeiten von Verkehrsdaten zum Inhalt haben.

In diesem Kapitel soll nur ein grober Überblick über die wichtigsten Ansätze zur Detektion und Klassifikation von Fahrzeugen gegeben werden, wobei die Liste der hier angeführten und diskutierten Algorithmen und Referenzen keinerlei Anspruch auf Vollständigkeit erhebt. Trotz der vielfachen Überschneidungen und der breiten Fächerung der Literatur auf diesem Gebiet erscheint eine grobe Einteilung wie folgt sinnvoll:

1. Detection und Recognition Algorithmen:

Unter diesem Titel kann die Gruppe der Algorithmen zusammengefasst werden, welche ein definiertes Objekt in einem einzelnen Bild suchen und lokalisieren; weiters zählen jene Verfahren zu dieser Kategorie, welche ein bekanntes Objekt in einem einzelnen neuen Bild wieder zu erkennen versuchen. In Abschnitt 2.2 wird dieses Thema besprochen und eine Übersicht über die üblichsten Ansätze gegeben.

2. Motion Detection:

In diese Kategorie fallen Algorithmen die zur Detektion von bewegten Objekten in einer Sequenz von Bildern angewendet werden. Das gebräuchlichste Verfahren auf diesem Gebiet nennt sich *Background Modelling and Subtraction* und wird zusammen mit einer kurzen Übersicht über Schattendetektierung in Abschnitt 2.3 diskutiert.

3. Tracking Algorithmen:

Diese Algorithmen haben ihr Einsatzgebiet im Bereich der Verfolgung von sich bewegendenden Objekten. Durch ihren Einsatz wird ein Zusammenhang von aus einzelnen Frames extrahierter Information über die Bewegung von Objekten hergestellt. Unter anderem zählen

- der *Kalman-Tracker*, welcher aus dem Kalman Filter entstanden ist,
- der *Condensation-Tracker*, welcher eine Weiterentwicklung des Kalman-Trackers bildet und
- der *KLT-Tracker* (Kanade-Lucas-Tomasi) aus dem Bereich der Feature-Basierten Tracking Algorithmen

zu den wichtigsten Vertretern. Diesem Themengebiet ist der letzte Abschnitt dieses Kapitels gewidmet.

2.2 Detection und Recognition Algorithmen

Der Großteil der Ansätze in diesem Bereich entstammt einem der wohl am meisten thematisierten Forschungsbereiche der Bildverarbeitung, dem Bereich der Face Detection und Face Recognition. Deshalb kann die folgende grobe Kategorisierung für Verfahren auf diesem Gebiet nach YANG [YKA02] auch für den Bereich der Fahrzeu-erkennung übernommen werden. Eine klare Abgrenzung der Kategorien ist hier nicht möglich, deshalb können einzelne Algorithmen durchaus mehreren Klassen zugeordnet werden.

2.2.1 Knowledge Based

Bei diesem Top-down Ansatz wird Vorwissen über die Eigenschaften eines Objekts benutzt, um eine Reihe von Regeln aufzustellen, die die Grundlage für ein System zur Detektion dieses Objekts bilden. Dabei kommen einfache Hypothesen zum Einsatz, wie zum Beispiel die Existenz und Symmetrieeigenschaften von einzelnen Features oder die relativen Abstände einzelner Features zueinander.

Bei der Verwendung eines solchen Ansatzes treten einige Probleme auf. Eines davon ist, aus menschlichem Wissen heraus klare eindeutige Regeln zu definieren. Sind diese zu allgemein gehalten, so werden zu viele und falsche Objekte erkannt, sind sie aber zu detailliert, so werden etwas von der Norm abweichende Objekte nicht mehr als solche erkannt. Eine weitere Schwierigkeit liegt in der Generalisierung dieses Ansatzes auf unterschiedliche Ansichten eines Objekts, da für jede mögliche Situation bestimmte Regeln definiert werden müssen.

Ein Verfahren, bei dem die Helligkeitsverteilung einer Straße und die Schattenbildung unter Fahrzeugen modelliert und zur Detektion von Fahrzeugen verwendet wird, findet man in einer Arbeit von TZOMAKAS [TS98]. Die Verwendung von Ansätzen, bei denen die Symmetrieeigenschaften von Fahrzeugen eine große Rolle spielen, findet man in früheren Arbeiten von KUEHNLE [Kue91] oder ZIELKE [ZBS93].

2.2.2 Feature Based

Im Gegensatz zu dem im vorigen Abschnitt besprochenen Ansatz, werden hier einzelne Merkmale eines Objekts verwendet, die invariant gegenüber Blickwinkel und Beleuchtungssituation erscheinen sollten. Basierend auf der Extraktion solcher Features werden statistische Modelle über eventuelle Zusammenhänge erstellt, die dann zur Detektion des Objekts verwendet werden.

Erwähnenswert ist hier vor allem die Arbeit von FERGUS, PERONA und ZISSERMAN [FPZ03]. Bei diesem Framework zur Object Recognition wird eine Klasse von Objekten durch eine flexible Konstellation von einzelnen Features repräsentiert. Dabei werden die Position eines Features, sowie dessen Größe und Aussehen als Wahrscheinlichkeitsverteilungen modelliert. Anfangs wird eine Vielzahl von Features durch einen Interest Operator ausgewählt, wobei aus dieser Prozedur die *Location* und der *Scale* der Features gewonnen wird. Die Features werden dann als Templates aus dem Trainingsbild extrahiert und auf eine gemeinsame fixe Größe ($n \times n$) skaliert. Danach wird eine PCA (*Principle Component Analysis*) in diesem

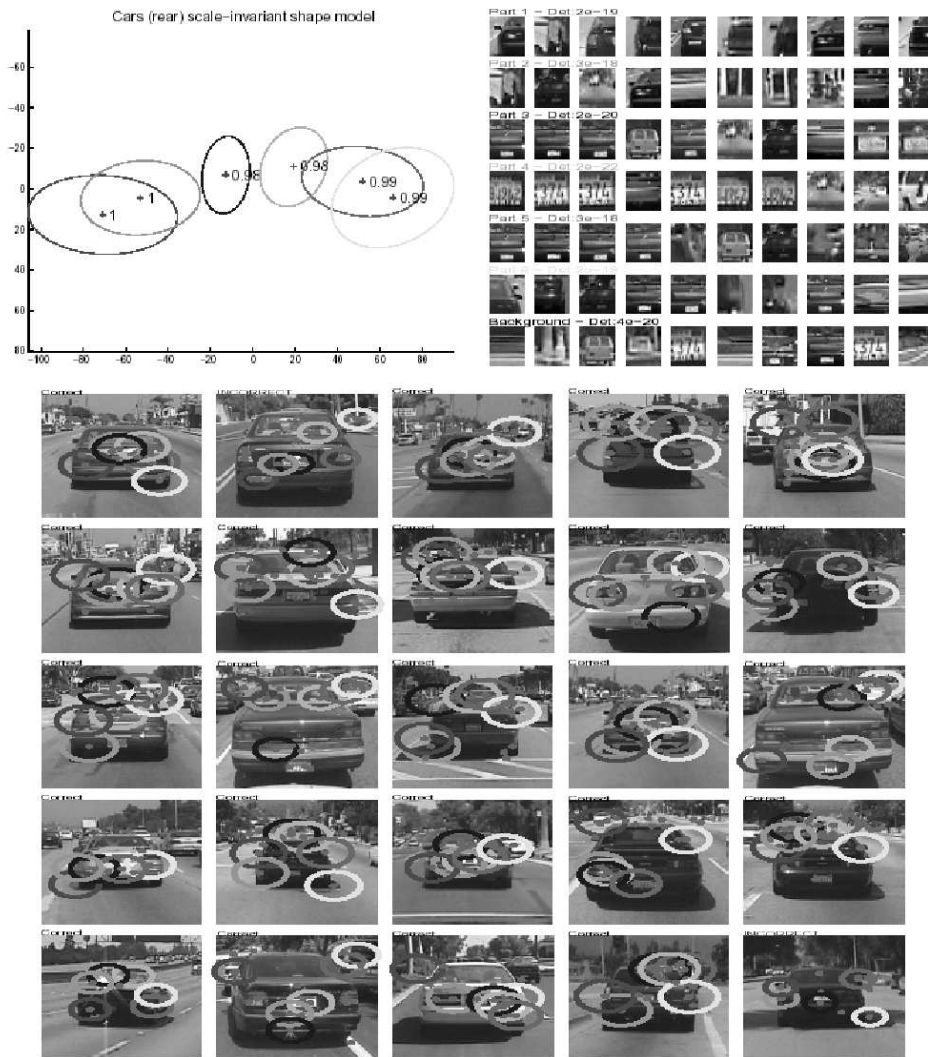


Abbildung 2.1: Beispiel für die Anwendung des Algorithmus nach FERGUS, PERONA und ZISSERMAN [FPZ03] auf Fahrzeuge in Heckansicht. Das hier dargestellte Modell benutzt 6 Features.

n^2 dimensionalen Raum durchgeführt und eine Untermenge von 10-20 Komponenten ausgewählt, welche die *Appearance* eines Features modellieren. Auf die durch die 3 Dimensionen, *Location*, *Scale* und *Appearance* aufgespannten Wahrscheinlichkeitsverteilungen wird schließlich eine Clustering mit Hilfe des EM-Algorithmus durchgeführt. Abbildung (2.1) zeigt die Anwendung des besprochenen Ansatzes unter Verwendung eines aus 6 Features bestehenden Modells für eine Klasse von Fahrzeugen in Heckansicht.

2.2.3 Model Based

Die Basis für diesen Ansatz wird durch Modelle von Objekten, welche aus Ecken, Kanten und Flächen zusammengesetzt werden, gebildet. Das Modell sollte dabei dem tatsächlichen Aussehen eines Objekts so nahe wie möglich kommen. Ist die Kamera-Abbildungsgeometrie bekannt, so kann eine solche Repräsentation in die Bildebene projiziert und mit dem tatsächlichen Erscheinungsbild eines mutmaßlichen Objekts verglichen werden. Eine Verbesserung der sich dadurch ergebenden Hypothese kann durch ein zumeist gradientenbasiertes Verfahren erreicht werden. Die Verwendung eines Modells lässt demnach eine einfache Lagebestimmung von erkannten Objekten zu. Ein Nachteil dieses Ansatzes ist es, dass zumindest eine grobe Kamerakalibrierung durchgeführt werden muss, um die erwähnte Kamera-Abbildungsgeometrie schätzen zu können.

Umfangreiche Studien dieses Ansatzes findet man in zahlreichen Arbeiten von NAGEL u.a. [KDN93] Hier wird ein parametrisiertes Modell vorgeschlagen, welches, bei entsprechender Einstellung der Parameter, einen großen Teil der verschiedenen Fahrzeugtypen repräsentieren kann. Ebenfalls thematisiert wird ein Modell für die Beleuchtungssituation und eine Beschreibung des verwendeten Matching Algorithmus. In einer weiteren Arbeit von NAGEL [KN97] wird ein spezielles Verfahren für das Fitten des Modells auf ein reales Objekt vorgeschlagen, welches eine robustere Detektion auch zum Teil verdeckter Objekte ermöglichen soll.

WORRALL, SULLIVAN und BAKER entwickelten ein auf geometrischen Überlegungen basierendes interaktives Tool, um eine einfache Kamerakalibrierung durchführen zu können [WSB94]. In einer darauf folgenden Arbeit von FERRYMAN u.a. [FWSB95] findet dieses Tool Verwendung. Die Autoren verwenden den modularen Aufbau eines ähnlichen Modells wie in den Arbeiten von NAGEL und evaluieren durch eine PCA, inwieweit die einzelnen Teile des Modells für eine jeweilige Fahrzeugkategorie, wie etwa Fahrzeuge mit Stufenheck oder mit Fließheck, signifikant sind.

2.2.4 Template Matching

Bei diesem Ansatz werden Bildausschnitte von bestimmten Ansichten eines Objekts vorbereitet und über ein Korrelationsmaß mit einem neuen Inputbild verglichen. Durch Schwellwertsetzung auf die Korrelationsfunktion wird versucht, die Existenz und die Position eines solchen Objekts festzustellen. Der größte Nachteil eines derartigen Algorithmus besteht in der weitestgehenden Unfähigkeit, mit verschiedenen

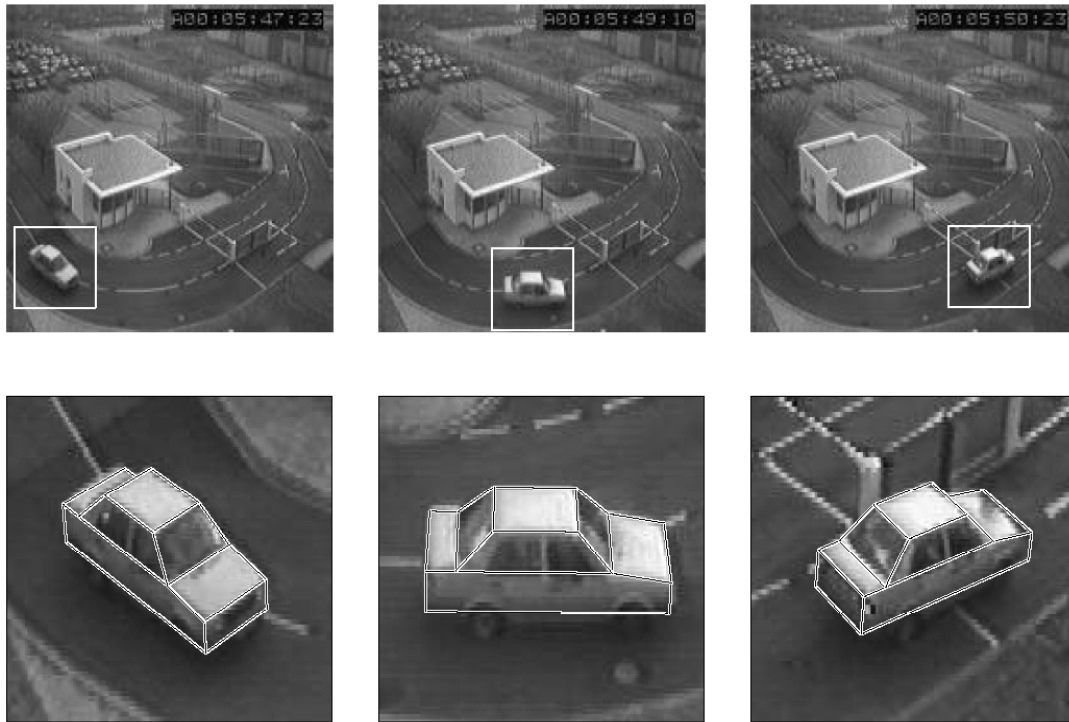


Abbildung 2.2: Beispiele für die Detektion eines vorbeifahrenden Fahrzeugs in der Arbeit von NAGEL [KDN93]

Blickwinkeln und Skalierungen eines Objekts umzugehen.

Dieses Verfahren findet sich beispielsweise in der Arbeit von BETKE u.a. [BHD96]. Ausgehend von einer Kamera, welche in einem sich selbst bewegenden Fahrzeug angebracht ist, werden überholte oder überholende Fahrzeuge detektiert. Dabei wird ein Datensatz von Graustufenbildern als Templates verwendet, welche verschiedene Fahrzeugtypen in Heckansicht zeigen, und die Existenz eines Fahrzeugs auf die oben beschriebene Art und Weise festgestellt.

2.2.5 Appearance Based

Diese Gruppe von Algorithmen erfährt in den letzten Jahren eine starke Zunahme an Popularität. Im Gegensatz zum *Template Matching* Ansatz werden Repräsentationsmodelle nicht vordefiniert, sondern aus einer Datenbank mit positiven und negativen Trainingsbeispielen gelernt. Dabei beruhen diese Verfahren stark auf Methoden aus dem Bereich der künstlichen Intelligenz und dem Bereich der Statistik.

Im Ansatz von MATTHEWS u.a. [MACH96] werden anfangs einzelne Ausschnitte eines Bildes, die möglicherweise Fahrzeugen entsprechen, auf eine vordefinierte Größe von 20×20 Bildpunkten skaliert. Jeder der skalierten Ausschnitte wird weiters in $25 \times 4 \times 4$ Fenster unterteilt, welche dann einer PCA unterzogen werden. Die daraus resultierenden Principle Components werden schließlich einem Neuralen Netz zur Klassifikation übergeben.

Neurale Netze werden auch in der Methode von GOERICK u.a. verwendet [GNW96]. Hierbei wird ein so genanntes *Local Oriented Coding* Verfahren verwendet, um Kanteninformation aus einem Bild zu extrahieren, wobei dann ein Histogramm in einem interessanten Bereich einem Neuralen Netz zugeführt wird.

Ein Verfahren, welches eine Kombination aus Gabor Filtern, Genetischen Algorithmen und Support Vector Machines beinhaltet, wurde von SUN, BEBIS und MILLER vorgeschlagen [SBM03]. Die Grundidee besteht darin, eine Filterpopulation einem Genetischen Algorithmus zuzuführen, der mit jedem Evolutionsschritt eine Performanceoptimierung durchführt. Dazu werden die Parameter einzelner Gabor Filter nacheinander binär kodiert, wobei ein Filter aus M Bit besteht. Eine aneinander gereihe Anzahl N einzeln binär kodierter Filter bildet als $N \cdot M$ langer Bitstream ein einzelnes Chromosom. Da im Laufe der Evolutionsschritte Filter mit stark redundantem Verhalten entstehen können, wird ein Clustering Algorithmus im Parameterraum der Gabor Filter angewandt und Cluster redundanter Gabor Filter in einzelne optimierte Gabor Filter umgewandelt. Mutation wird durch einzelne Bitflips durchgeführt, einfaches Crossover wird verwendet und im Selektionschritt werden die besten Individuen beibehalten, so dass die Anfangsgröße der Population immer beibehalten wird. Die Fitnessfunktion wird hier durch eine SVM berechnet, welche die Qualität der durch jeden einzelnen Gabor Filter extrahierten Features bewertet.

Das von PAPAGEORGIOU, OREN und POGGIO [POP98] publizierte Framework bildet den Ausgangspunkt vieler Forschungsarbeiten im Bereich der Detektion und Klassifikation von Objekten, bei welchen so genannte Haar-Wavelets zum Einsatz kommen. In dieser Arbeit wird ein rechenaufwendiges Verfahren zur Berechnung der Wavelets auf ein in einzelne Ausschnitte unterteiltes Inputbild verwendet und anschließend gute Features händisch selektiert. Danach wird eine SVM trainiert, welche das genaue Modell der zu beschreibenden Klasse lernen soll. Die Anwendung dieses Ansatzes wird schließlich in der Detektion von Gesichtern und Passanten demonstriert und erreicht durchaus vertretbare Ergebnisse.

In der Arbeit von VIOLA und JONES [VJ01] finden sich einige Ideen wieder, die zum Teil auf die Arbeit von PAPAGEORGIOU u.a. zurückgehen, aber auch neue Vor-

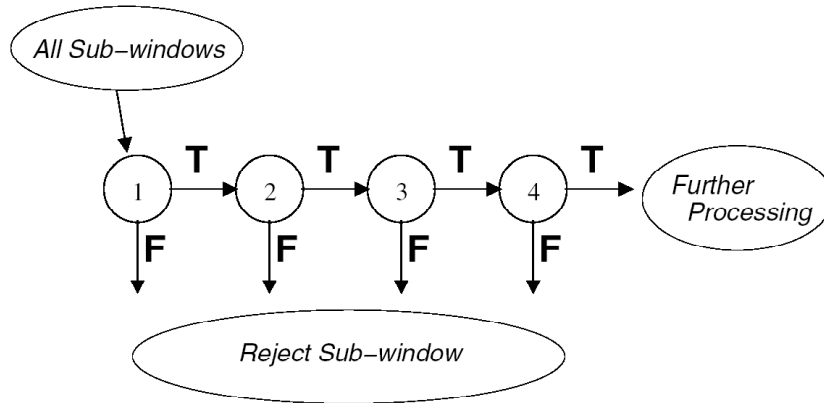


Abbildung 2.3: Schematische Darstellung der Kette von binären Klassifikatoren in der Arbeit von VIOLA und JONES [VJ01].

schläge, die vor allem zur Echtzeitfähigkeit eines entworfenen Systems beitragen sollen. Der erste Vorschlag ist eine andere Repräsentationsart eines Inputbildes, um die bereits erwähnte Klasse von Haar-Wavelets mit deutlich reduziertem Rechenaufwand und damit deutlich schneller berechnen zu können (siehe Abschnitt 4.2.1). Weiters wird für die Selektion von signifikanten Merkmalen die Benutzung einer leicht modifizierten Form des von SCHAPIRE entwickelten ADA-Boost Algorithmus [SF96] vorgeschlagen (siehe Abschnitt 4.3.1 für eine ausführliche Erklärung). Schließlich wird der Entwurf einer aus diesen selektierten Features gebildeten Kette binärer Klassifikatoren angeregt. Der in Abbildung (2.3) dargestellte Aufbau bildet sozusagen ein Sieb, durch welches negative Ausschnitte von Bildern aussortiert und nur vermeintlich positive Kandidaten aufbehalten werden. Während dieses System, evaluiert durch die Anwendung zur Detektion von Gesichtern, vergleichbare Ergebnisse in diesem Bereich erzielt, glänzt dieses Verfahren vor allem durch die schnelle Verarbeitung von Inputbildern und der damit verbundenen hohen Durchsatzrate.

2.3 Motion Detection

Im Gegensatz zu den im vorhergegangenen Abschnitt diskutierten Algorithmen, wird bei dieser Gruppe von Verfahren die Information über die Präsenz und die Position eines Objekts aus einer Reihe von Bildern und der sich in dieser Sequenz abbildenden Bewegung extrahiert. Ein gängiger Ansatz, bewegte Objekte zu identifizieren, nennt sich *Background Subtraction*. Hierbei wird ein aktuelles Bild mit

einem Referenz- oder Hintergrundmodell verglichen und Regionen mit starken Differenzen werden zu bewegten Objekten erklärt.

Da der Einsatz dieser Methode meist am Anfang einer Computer Vision Anwendung steht, ist es von großer Bedeutung, dass die extrahierten Bereiche realen bewegten Objekten möglichst gut entsprechen. Die Aufgabe, die diese Verfahren zu erfüllen haben, ist aber alles andere als trivial. Ausgehend von einer stationären Kamera, welche den Verkehrsfluss auf einer beliebigen Straße aufzeichnet, muss ein Background Subtraction Algorithmus mit zeitlich variierenden Beleuchtungsbedingungen umgehen können, ebenso muss er robust sein gegenüber sich ändernden Wetterverhältnissen. Weitere Probleme werden durch von bewegten Objekten geworfene Schatten verursacht, die sich trotz ihres immateriellen Wesens in Videobildern abbilden. Der Algorithmus muss es bewegten Objekten erlauben, sich in das Hintergrundmodell zu integrieren und aus diesem wieder zu lösen, wie zum Beispiel bei einem sich sehr langsam bewegendem oder gar stockenden Verkehrsfluss. Zuletzt muss ein solches Verfahren auch noch Echtzeitqualitäten besitzen, also effizient im Gebrauch von Speicher- und Rechenressourcen sein.

Grundsätzlich unterscheidet man 2 Gruppen von Algorithmen auf dem Gebiet des Background Modelling (BGM): *Nicht-Rekursive* und *Rekursive* Algorithmen. Für eine detaillierte Evaluierung der hier gelisteten Algorithmen auf dem Gebiet der Fahrzeugdetektion sei auf die Arbeit von CHEUNG und KAMATH [CK03] verwiesen. Eine spezielle Gruppe von Algorithmen, die nur bedingt mit dem Ansatz des Background Modelling verbunden ist, ist die der *Shadow Detection* und *Shadow Modelling* Algorithmen. Diese Verfahren beschäftigen sich mit der bereits erwähnten Problematik von Schatten in Bildern, welche keinen realen Objekten entsprechen.

2.3.1 Nicht-Rekursive BGM-Algorithmen

Bei dieser Klasse werden in einem Puffer eine gewisse Anzahl an Videobildern gespeichert und ein Hintergrundmodell basierend auf der zeitlichen Veränderung der einzelnen Bildpunkte generiert. Da diese Algorithmen abgesehen von den im Puffer gespeicherten Bildern keinerlei Information verwenden, sind sie sehr anpassungsfähig. Handelt es sich bei dem beobachteten System aber um einen sehr lange andauernden Prozess mit wenig Bewegung, so kann der Speicherbedarf dieser Algorithmen durchaus kritisch sein.

Die gebräuchlichsten Methoden in dieser Kategorie sind:

Frame Differencing: Dies ist die einfachste Background Modelling Methode. Zum

Zeitpunkt t wird ein Frame eines früheren Zeitpunkts $t - 1$ als Hintergrundmodell angenommen. Anwendung findet dieses Verfahren beispielsweise in der bereits erwähnten Arbeit von BETKE u.a. [BHD96].

Median Filter: Der Median Filter ist eines der meistverwendeten Verfahren zur Modellierung des Hintergrunds. Als Modell eines einzelnen Hintergrundpixels wird der Median über die Werte desselben Bildpunktes in allen Frames des Puffers berechnet. Beispiele für die Anwendung dieses Filters finden sich unter anderem in Arbeiten von AGGARWAL [AZ01] und CUCCHIARA [CPP03].

Linear Predictive Filter: Dieses Verfahren wird von TOYAMA u.a. vorgeschlagen. Hier wird das Hintergrundmodell durch das Anwenden eines *linear predictive filters* auf die Bildpunkte im Puffer gebildet. Dabei werden die Filterkoeffizienten für jedes Inputbild erneut berechnet [TKBM99].

Non-Parametric Model: Dieser Algorithmus wurde von ELGAMMAL u.a. [EHD99] vorgeschlagen und arbeitet folgendermaßen. Der gesamte Input vom Anlaufen des Systems bis zum aktuellen Zeitpunkt, $I_{t-L}, I_{t-L+1}, \dots, I_{t-1}$, wird verwendet um eine Schätzung der Dichtefunktion $f(I_t = u)$ für ein einzelnes Pixel zu bilden,

$$f(I_t = u) = \frac{1}{L} \sum_{i=t-L}^{t-1} K(u - I_i) \quad (2.1)$$

wobei $K(\cdot)$ einen Gaußschen Kernel bezeichnet. Ein Pixel I_t wird dann als Vordergrund bezeichnet, wenn es unwahrscheinlich ist, dass es von dieser Verteilung modelliert wird (Schwellwertsetzung über der Wahrscheinlichkeit). Statt solchen unimodalen, können natürlich auch multimodale Hintergrundmodelle aufgebaut werden.

2.3.2 Rekursive BGM-Algorithmen

Rekursive Algorithmen benötigen keinen Puffer, sondern führen rekursiv ein auf jedem neuen Input basierendes Update eines einzelnen Hintergrundmodells durch. Das bedeutet auf der einen Seite, dass zu einem bestimmten Zeitpunkt lange zurückliegende Eingangsbilder, meist exponentiell gewichtet, immer noch einen gewissen Einfluss auf das Modell nehmen, auf der anderen Seite aber, dass eventuelle Fehler im Modell sehr lange mitgeschleift werden können.

Als wichtigste Vertreter dieser Kategorie werden genannt:

Approximated Median Filter: Er bildet das rekursive Gegenbeispiel zum gerade erwähnten *Median Filter*. Entwickelt von MCFARLANE und SCHOFIELD [MS95],

beispielsweise von REMAGNINO u.a. [R⁺97] in einer Anwendung zur Verkehrsüberwachung eingesetzt, führt dieser Filter eine rekursive Approximierung des wahren Medians durch. Ein fortlaufender Schätzwert für den Median eines Pixels wird um 1 erhöht, wenn der Wert des Pixels in einem neuen Inputbild größer als der aktuelle Schätzwert ist, und um 1 verringert, wenn der Wert geringer ist. Diese Schätzung konvergiert gegen den wahren Median.

Kalman Filter: Eine genauere Diskussion der Funktion des Kalman Filters findet sich in Anhang A.1. Während die Hauptanwendung des Kalman Filters auf dem Gebiet der Verkehrsüberwachung die des Trackens von Fahrzeugen ist, kann der Kalman Filter auch zum “Tracken” von Eigenschaften des Hintergrunds, wie etwa von Helligkeit verwendet werden. Eine populäre Methode, welche das Tracken der Helligkeit und deren zeitlicher Ableitung benutzt, wurde von KARMANN und BRANDT [KB90] vorgeschlagen, eine Anwendung dieser Methode findet sich beispielsweise in der Arbeit von KOLLER, WEBER und MALIK [KWM93].

Mixture of Gaussians (MoG): Bei dieser Art der Modellierung wird wie bei der bereits zuvor erwähnten *Non-Parametric Model* Methode eine multimodale Pixel-Dichtefunktion für jedes einzelne Pixel erstellt. Die Parameter dieser Verteilungen erfahren ein adaptives Update basierend auf neuem Input. Für eine genauere Beschreibung dieses Verfahrens sei auf die Arbeit von STAUFFER und GRIMSON verwiesen [SG00].

2.3.3 Shadow Modelling Algorithmen

Diese spezielle Klasse von Algorithmen befasst sich mit der Problematik des Schattenswurfs von Objekten. Da es sich bei Systemen zur Verkehrsüberwachung um Outdoor-Applikationen handelt, ist der richtige Umgang mit Schatten von großer Bedeutung, da diese Fahrzeuge virtuell verschmelzen lassen oder die Größe einer Detektion unangenehm beeinflussen können (siehe Abbildung (2.4)).

Folgt man der Ansicht von JIANG und WARD [JW92] bzw. von STAUDER u.a. [SMO99], so können Schatten in 2 Klassen eingeteilt werden: *self shadow* wird der nicht beleuchtete, d.h. der Lichtquelle abgewendete Teil eines Objekts genannt, und *cast shadow* der von dem Objekt auf einen Untergrund geworfene Schatten, wobei Probleme meist durch Phänomene letzterer Kategorie verursacht werden. In der Literatur findet man vor allem eine Einteilung in die Gruppe der *statistischen* und die Gruppe der *deterministischen* Ansätze. Aus der langen Liste der Referenzen zu diesem Thema kann unter anderem auf die Arbeiten von MIKIĆ, PRATI, CUCCHIARA u.a. verwiesen werden [MCKT00] [PMGT01] [PMT03].

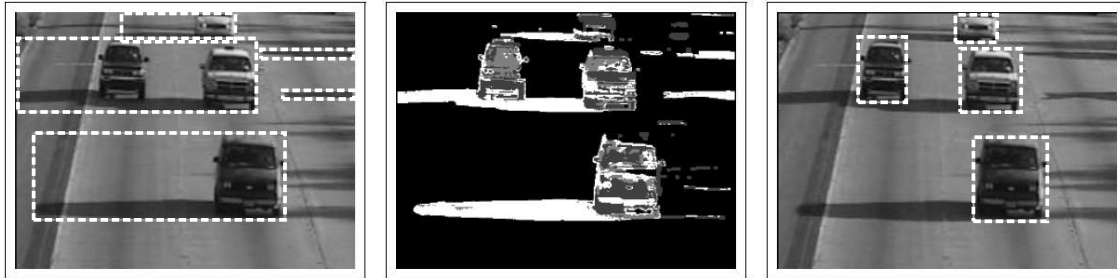


Abbildung 2.4: Beispiel für die Schattendetektion in der Arbeit von PRATI [PMGT01]. Links eine Aufnahme ohne Schattenunterdrückung, in der Mitte das Ergebnis der Schattendetektion (weiß) und rechts das Ergebnis nach erfolgreicher Unterdrückung der Schatten.

2.4 Tracking Algorithmen

Tracking Algorithmen werden eingesetzt, um einen Zusammenhang zwischen der aus einzelnen Bildern extrahierten Information über die Bewegung eines Objekts herzustellen, wobei der gebräuchlichste Tracking Algorithmus der *Kalman*-Tracker ist. Aus dem Kalman Filter entstanden, hat er seine Wurzeln im Bereich der Navigation und bildet ein allgemeines Framework zum Tracken von Systemen. Im Gegensatz dazu haben der *Condensation*-Tracker und der *KLT*-Tracker ihren Ursprung im Bereich der Bildverarbeitung. Der Condensation-Tracker kann als Weiterentwicklung des Kalman Trackers verstanden werden und versucht, den Umriss von Objekten zu erkennen und zu verfolgen. Der KLT-Tracker ist auf der Extraktion einzelner optimaler Features und deren Verfolgung aufgebaut. Die folgenden Unterkapitel sind einer kurzen Beschreibung der hinter den einzelnen Ansätzen liegenden Ideen gewidmet.

2.4.1 Der Kalman Tracker

Die mathematischen Grundlagen des nach seinem Erfinder Rudolph E. KALMAN [Kal60] benannten diskreten Kalman Filters bilden die Grundlage für diesen Tracking Algorithmus. Die Filtergleichungen bilden eine auf dem Predictor-Corrector Prinzip beruhenden *optimalen* Schätzer für den Zustand (*State*) $\mathbf{x} \in \mathbb{R}^n$ eines diskreten Prozesses, wobei der Mittelwert des resultierenden Fehlers minimiert wird. Eine Übersicht über die Funktion des Kalman Filters wird in Anhang A.1 gegeben.

Beim Tracking modelliert der Kalman Filter die Bewegung eines Objekts ausgehend von einer Anfangsposition zu einem gegebenen Zeitpunkt, wobei die Performance

stark davon abhängt, wie gut die Charakteristik der Bewegung durch die Systemgleichung modelliert wird (siehe auch Abschnitt 3.8).

Je nach Anwendung ist ein mehr oder weniger komplexes Modell notwendig. In der Arbeit von COIFMAN u.a. [CBMM98] findet ein einfaches Modell Verwendung. Für einen Frame werden ein Ein- und ein Ausgangsbereich definiert, respektive Bereiche, in die Fahrzeuge ein- und ausfahren. Ein Feature Detektor wird zur Auswahl auffälliger Punkte im Eingangsbereich verwendet und für jeden einzelnen detektierten Punkt wird ein neuer Tracker eingeführt. Die Systemgleichung dieser Tracker berücksichtigt nur Position und Geschwindigkeit des Features, enthält also einen 4-dimensionalen Zustandsvektor. Bewegt sich ein getracktes Feature in den Ausgangsbereich, so wird ein aus dem Verlauf und der Ähnlichkeit der Bewegung der einzelnen Features schließender Gruppierungsalgorithmus initiiert, welcher aus einer Gruppe von Features eine Fahrzeughypothese generiert.

Von DELLAERT und THORPE [DT97] stammt eine Arbeit über ein System zur Detektion und zum Tracking von Fahrzeugen von einem sich selbst bewegenden Fahrzeug aus. Aufgrund der Komplexität der Aufgabe findet hier ein Kalman Tracker mit einem 10-dimensionalen Zustandsvektor Anwendung, wobei nicht nur die Position des getrackten Fahrzeugs in einem Frame, sondern neben anderem auch dessen Geschwindigkeit und die Größe des umschreibenden Rechtecks in der Systemgleichung berücksichtigt werden.

2.4.2 Der Condensation Tracker

Dieses Verfahren wurde erstmals von ISARD und BLAKE vorgeschlagen [IB98]. Es basiert auf der Problematik, dass bei dem oben genannten Kalman Filter eine unimodale gaußsche Repräsentation der Wahrscheinlichkeitsdichtefunktion verwendet wird und damit nur eine einzelne Hypothese über die Position und Form eines Objekts unterstützt wird. Dieser Nachteil führt dazu, dass in Szenen mit grobtexturiertem Hintergrund der Kalman Filter allein nicht mehr in der Lage ist, ein Objekt fehlerfrei zu verfolgen.

In diesem neuen Verfahren wird nun die Idee des Kalman Filters weiterverfolgt und eine multimodale Dichtefunktion eingeführt, die es erlaubt, mehrere Hypothesen über den Zustand eines Objekts zu verfolgen. Weiters werden so genannte *B*-Splines verwendet, um eine Hypothese über den Umriss eines Objekts aufzustellen. An den Kurven, welche eine einzelne Hypothese bilden, wird eine Kantendetektion in Normalrichtung durchgeführt und ein Likelihood-Maß errechnet, welches die Hypothese unterstützt oder schwächt.

In der Arbeit von MEIER [MA99] wird der Condensation Algorithmus von Isard und Blake erweitert, um mit neu auftauchenden Objekten umgehen zu können. Dies ist im Bereich der Verkehrsüberwachung in jedem Fall nötig, bei Anwendung einer stationären Kamera, um in den Bildbereich ein- und ausfahrende Fahrzeuge tracken zu können, bei Verwendung einer bewegten Kamera wie in dieser Arbeit, um Hindernisse und andere Fahrzeuge erkennen und tracken zu können. Die Evaluierung der vorgeschlagenen Technik erfolgt hier auf einer computergenerierten Simulation eines sich bewegenden Fahrzeugs.

2.4.3 Der KLT Tracker

Der so genannte "Kanade-Lucas-Tomasi Tracker" (KLT Tracker) hat seine Wurzeln in der Arbeit von LUCAS und KANADE [LK81], wurde von TOMASI und KANADE [TK91] entwickelt und in geschlossener Form von SHI und TOMASI [ST94] publiziert.

Diesem Algorithmus liegt die Idee zugrunde, dass die Aufgabe der Featureextraktion nicht von der Aufgabe des Trackings getrennt werden sollte, da die Performance im Bereich des Trackings unmittelbar von der Eignung der Features abhängt; mit anderen Worten - gute Features sind jene, die auch gut getrackt werden können. Ein gutes Feature ist ein stark texturierter Bereich mit großen Helligkeitsschwankungen in beiden Achsenrichtungen. Bezeichnet man nun die Helligkeitsfunktion mit $g(x, y)$ und eine lokale Änderungsmatrix mit

$$\mathbf{Z} = \begin{bmatrix} g_x^2 & g_x \cdot g_y \\ g_x \cdot g_y & g_y^2 \end{bmatrix}, \quad (2.2)$$

so muss ein $(m \times n)$ großer Bereich eines Bildes folgendes Kriterium erfüllen, um als Feature akzeptiert zu werden: beide Eigenwerte λ_1 und λ_2 der Matrix $\mathbf{Z}$, evaluiert in der Mitte des selektierten Bereiches, müssen größer als ein festgelegter Schwellwert λ_{thr} sein:

$$\min(\lambda_1, \lambda_2) > \lambda_{thr} \quad (2.3)$$

Außerdem werden nur eine festgelegte Anzahl N Features selektiert, die auch einen gewissen Mindestabstand zueinander haben müssen. Ein rein translatorisches Modell wird für das Tracken der Features durch eine Sequenz von Bildern verwendet, während durch ein Maß für die Unterschiedlichkeit (*Dissimilarity*) eines Patches zwischen dem aktuellen und dem vorhergegangenen Frame ein affines Veränderungsmodell des Image Patches gesteuert wird. Die Bewegung, die ein Feature zwischen

aufeinander folgenden Frames vollziehen kann, ist hier deutlich durch den erforderlichen Rechenaufwand beschränkt.

In der Literatur oftmals referenziert und verwendet wird die Implementierung des KLT-Trackers von BIRCHFELD [Bir97]. In der Arbeit von TOMMASINI u.a. [TFTR98] wird ein Algorithmus zur Eliminierung von Feature-Outliern behandelt, mit dem Ziel robusteren Trackings. BELL u.a. [BFH99] setzten den KLT-Tracker erfolgreich in einem Projekt zur Verfolgung von Luftfahrzeugen ein.

2.5 Zusammenfassung

In den vorangegangenen Abschnitten wurden die wichtigsten Algorithmen aus dem breiten Spektrum der Literatur herausgegriffen und thematisiert. Zwar können mit einigen Ansätzen durchaus viel versprechende Erfolge erzielt werden, trotzdem ist eine Universallösung für eine zuverlässige Detektion und Klassifikation von Fahrzeugen mittelfristig nicht in Sicht. Ausgehend vom Umfang der zu lösenden Problematik, drängt sich aber der Verdacht auf, dass ein schnelles und vor allem sehr performantes System nur aus Kombinationen der jeweiligen Algorithmen entstehen kann.

KAPITEL 3

Systemarchitektur

3.1 Allgemeines

Das gesamte System wurde in C++ für den Einsatz unter LINUX entwickelt. Unter anderem wurden die INTEL PERFORMANCE PRIMITIVES Bibliotheken [Int04], aber auch die DEVIL Image Libraries [Woo04] und die XVISION2 Bibliotheken [Yal98] verwendet. Die Benutzeroberfläche wurde mit Trolltechs QT erstellt [Tro04].

Zu Beginn dieses Kapitels wird eine grobe Übersicht über die Vorgaben und Randbedingungen gegeben, die beim Entwurf des Systems beachtet werden mussten.

Darauf folgt ein Überblick über die Struktur des Systems. Da es sich um einen modularen Aufbau handelt, ist die Funktion durch das Zusammenwirken einzelner Sektionen gegeben.

Im Anschluss daran, werden die einzelnen Module beschrieben und die jeweils verwendeten Algorithmen im Detail thematisiert. Außerdem wird versucht, das Zusammenspiel der einzelnen Module möglichst klar zu zeichnen.

Den Schluss bildet eine Diskussion der beim praktischen Einsatz des Systems aufgetretenen Probleme.

3.2 Vorraussetzungen und Randbedingungen

Obwohl beim Entwurf des Systems größter Wert auf die universelle Einsetzbarkeit gelegt wurde, mussten einige Randbedingungen vorab festgelegt werden, die im Folgenden kurz genannt werden:

- Es wird eine Aufnahmeposition in einer Ebene über der Fahrbahn eingenommen, wobei die Kamera vor Erschütterungen so gut wie möglich geschützt ist.
- Die Beleuchtung der Szene erfolgt durch Tageslicht oder durch künstliches Licht. Nächtliche Bedingungen werden nicht behandelt.
- Es herrschen keine extremen Wetterbedingungen vor wie beispielsweise dichter Nebel oder dichter Schneefall.
- Der Verkehrsfluss bewegt sich prinzipiell auf die Kamera zu oder von der Kamera weg, der optische Fluss in der Bildebene verläuft also von oben nach unten oder umgekehrt.
- Es handelt sich bei der Szene in der überwiegenden Zeit des praktischen Einsatzes um fließenden Verkehr und nicht um eine stockende oder stauende Verkehrssituation.

Als weitere Vorgabe sollte im Bereich der Algorithmik auf die möglichst einfache Umsetzbarkeit auf einem DSP (Digitaler Signalprozessor) Rücksicht genommen werden. Deshalb sollten möglichst geringe Speicher- und Rechenressourcen verwendet werden.

3.3 Systemstruktur

Das Programm besteht aus sechs einzelnen Modulen, deren Zusammenwirken in einer groben Struktur in Abbildung 3.1 zu sehen ist.

Ein übergeordnetes parametrisiertes Modul, das *Real-World-Modell*, enthält vom Benutzer eingegebene Informationen über die aufgenommene Szene. Jedes andere Modul des Systems wird von dieser Einheit mit diverser Information versorgt (siehe Abschnitt 3.4).

Die Basis zur Datenerfassung wird von einer selbstständigen Einheit, dem *Bildaufnahme-Modul* gebildet, welches zur Aufnahme von Bildern verschiedener Quellen verwendet wird (Abschnitt 3.5).

Ein weiteres Modul, das *Hintergrundmodell* übernimmt die Modellierung des Hintergrundes basierend auf einer Folge von Inputbildern (siehe Abschnitt 3.6). Außerdem stellt das Modul ein binäres Bild mit Information über Bewegung in der Szene zur Verfügung.

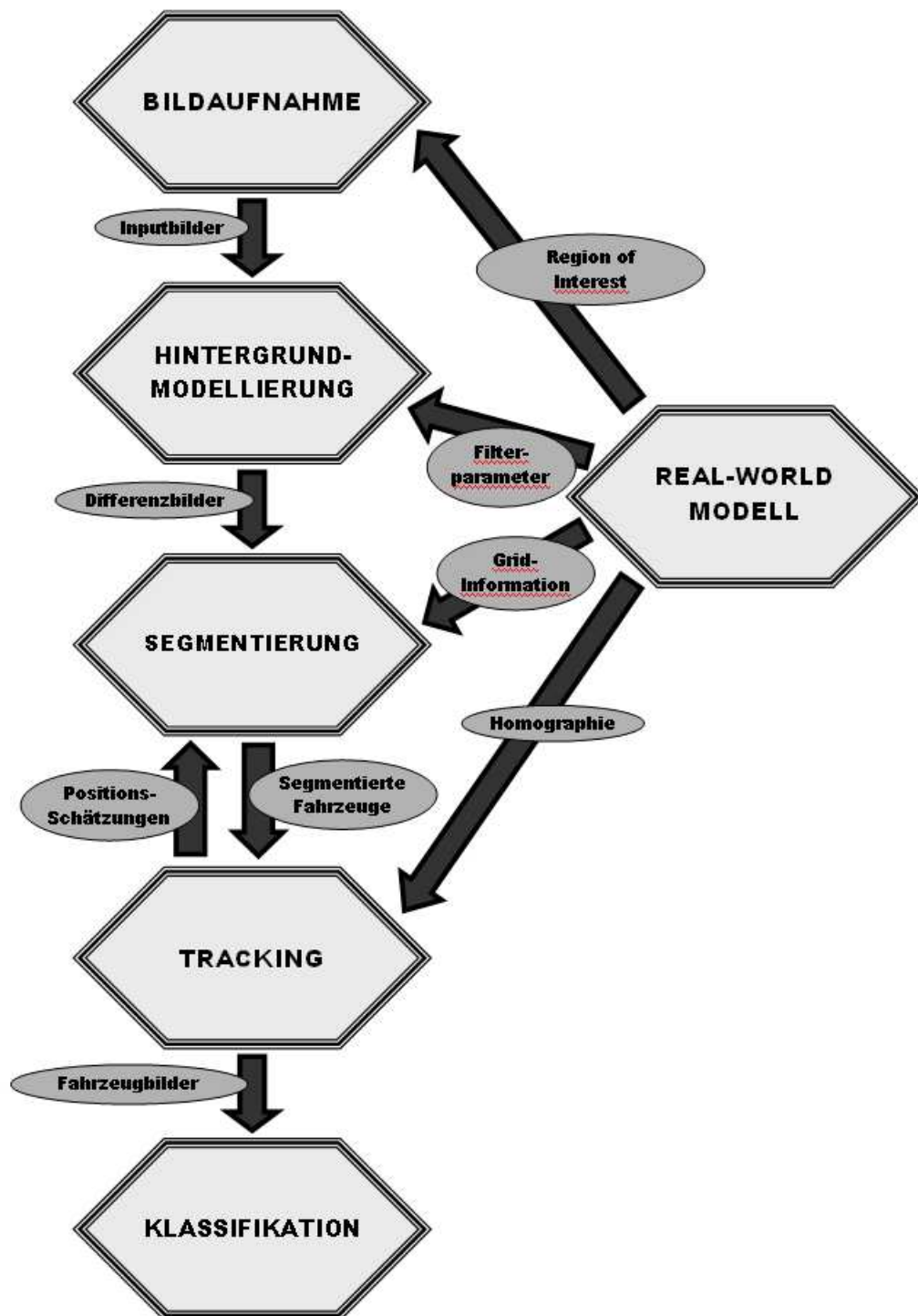


Abbildung 3.1: Struktur und modularer Aufbau des Systems zur Datenaufnahme

Das *Segmentierungs-Modul* (Abschnitt 3.7) übernimmt die genaue Detektion und Lokalisation eines bewegten Objekts in dem Inputbild. Als Information dient sowohl das binäre Bild des vorverarbeitenden *Hintergrund-Moduls*, als auch Information aus dem nachgeschalteten *Tracker-Modul*.

Das *Tracker-Modul* besteht aus einer Menge einzelner Kalman-Tracker, welche das Tracking von Objekten durch die Szene übernehmen (Abschnitt 3.8). Außerdem erstellt das Modul eine History über die Bewegung eines Objekts von der ersten Detektion bis zum Verlassen des Bildbereichs.

Das sechste und letzte Modul des Systems dient zur *Klassifizierung* von Fahrzeugen. Die Funktion des Moduls wird hier nur kurz angerissen (siehe Abschnitt 3.9). Eine genaue Beschreibung der verwendeten Methoden und Algorithmen zum Entwurf und zur Implementierung einer Klassifikationsstruktur findet sich in Kapitel 4.

3.4 Das Real-World Modell

Im *Real-World Modell* werden alle benutzerdefinierten Informationen über die beobachtete Szene gespeichert und den einzelnen anderen Modulen zur Verfügung gestellt. Die drei wichtigsten Definitionen sind:

- *Region Of Interest*; ein ausgewählter Bildbereich, in den sich die zu untersuchende Bewegung abbildet; wird im *Bildaufnahme-Modul* verwendet (Abschnitt 3.5).
- *Growing Grid*; ein spezieller Raster zur besseren Segmentierung von potenziellen Fahrzeugen; wird im *Segmentierungs-Modul* verwendet (Abschnitt 3.7).
- *Homographie*; Informationen zur Abbildungsgeometrie der Straße in das Bild; wird zur Schätzung der wirklichen Position eines Fahrzeugs im *Tracker-Modul* verwendet (Abschnitt 3.8).

Das *Real-World Modell* bildet sozusagen eine Datenbank von Umgebungsinformationen, die von den einzelnen Modulen verwendet werden.

3.5 Das Bildaufnahme-Modul

Bei der Konstruktion des *Bildaufnahme-Moduls* müssen einige Einschränkungen in Kauf genommen werden, die vor allem durch die technischen Ressourcen definiert

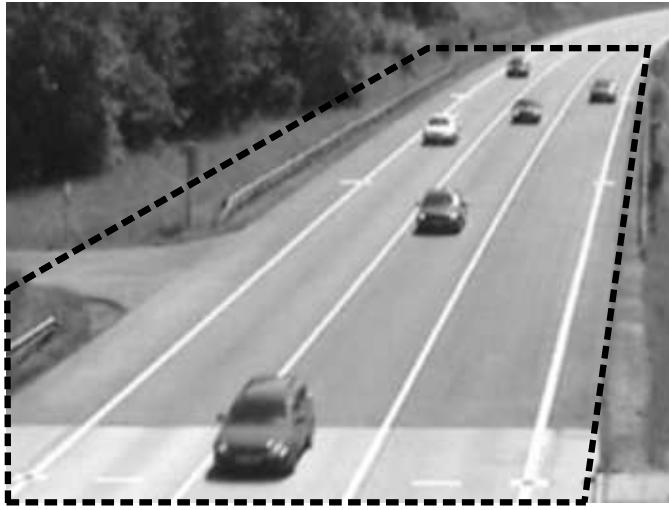


Abbildung 3.2: Beispiel für die Definition einer *Region Of Interest*. Nur der Bereich innerhalb der unterbrochenen Linien wird untersucht. Die Bereiche außerhalb werden verworfen.

werden, aber auch durch die Komplexität der Aufgabenstellung sinnvoll erscheinen.

Während sowohl die Verarbeitung von auf einem externen Datenträger in beliebigem Format gespeicherten Bildern möglich ist, ist auch die direkte Anbindung an eine Kamera für den Echtzeit-Online-Betrieb möglich.

Beim Lesen von einem Datenträger ist die Größe eines einzelnen Bildes nicht beschränkt, das Speicherformat ist sowohl aus komprimierten, als auch unkomprimierten Typen frei wählbar. Die Geschwindigkeit, mit der Bilder von einem Datenträger gelesen werden können, ist allerdings durch die dem Datenträger eigene Zugriffsgeschwindigkeit und die benötigte Menge an zu ladendem Datenmaterial beschränkt.

Da beim Kamerabetrieb die *XVision2*-Libraries [Yal98] verwendet werden, ist das Bildformat auf 320×240 Pixel beschränkt. Die Anzahl der bereitgestellten Bilder pro Sekunde ist mit 30 FPS (*Frames per Second*) jedoch mehr als ausreichend. Außerdem erlauben diese Libraries den Einsatz einer handelsüblichen TV-Karte mit einem *Brooktree BT8x8* Chipsatz als Framegrabber, was eine sehr kostengünstige Anschaffung ermöglicht.

Nach der Bildaufnahme werden die Aufnahmen so zurechtgeschnitten, dass sie nur mehr den im *Real-World Modell* definierten interessanten Ausschnitt, die *Region of Interest* zeigen.

Durch die Auswahl einer Region eines Bildes unmittelbar nach der Aufnahme wird erreicht, dass nur für den Algorithmus relevante Bildbereiche den weiteren Ver-



Abbildung 3.3: Beispiel für ein Differenzbild. Links ist das aus den drei Kanälen bestehende Referenzbild zu sehen, in der Mitte ein neues Inputbild. Rechter Hand befindet sich das resultierende Differenzbild.

arbeitungsschritten zugeführt werden. Bereiche, die uninteressant für die gestellte Aufgabe sind, werden verworfen, da sie nur Ressourcen unnötigerweise verschwenden würden.

In Abbildung 3.2 ist ein Beispiel für die Definition einer *Region Of Interest* in einer Szene zu sehen. Die Videobilder werden auf den Bereich innerhalb der unterbrochenen Linien zugeschnitten, die Bereiche außerhalb werden verworfen. Diese Zuschnitte werden dann dem nächsten Modul zur Verfügung gestellt.

3.6 Hintergrundmodell

Hier wird das bereits in Abschnitt 2.3.2 erwähnte *Approximated Median* Verfahren verwendet, um ein Modell des Hintergrunds zu erstellen.

Es wird ein Referenzbild für jeden einzelnen der *RGB*-Kanäle gebildet und die Kanäle eines neuen Inputbildes von den jeweiligen Referenzkanälen abgezogen. Findet man zumindest in einem Kanal eine Abweichung größer als ein vordefinierter Schwellwert, so wird das ausgegebene Binärbild an der entsprechenden Stelle auf 1 gesetzt, dieser Bildpunkt also aus bewegter Vordergrund definiert (Abbildung 3.3).

Anschließend wird ein Algorithmus aus einer Arbeit von CUCCHIARA u.a. [CGPP01] zur Schattendetektion angewandt. Dieser Algorithmus erlaubt es, Schatten zu großen Teilen als solche zu erkennen und aus dem Differenzbild zu entfernen.

Liegt ein Bereich im Schatten eines Objekts, so bedeutet dies zumeist eine größere Änderung der Helligkeit in diesem Bereich, während die Farbgebung selbst nur wenig verändert wird. Ein bereits als Vordergrund definierter Bildpunkt wird demnach als Schatten erkannt, sofern nur die Änderung der Helligkeitskomponente zwischen einem neuen Inputbild und dem Referenzbild einen gewissen Schwellwert

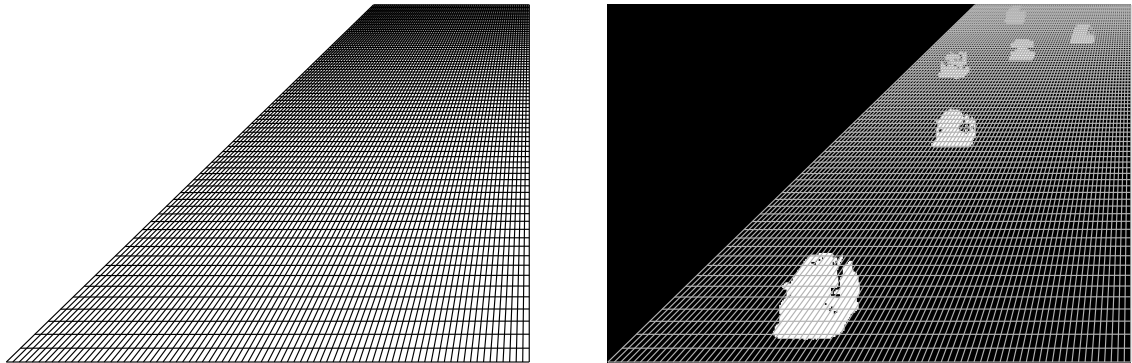


Abbildung 3.4: Beispiel für ein dem Straßenverlauf angepasstes *Growing Grid* (links), welches über das Differenzbild gelegt wird (rechts).

überschreitet und die Farbunterschiede niedriger als ein zweiter Schwellwert sind. Die so definierten Bereiche des Bildes werden danach aus dem Differenzbild wieder entfernt.

Größere bewegte Objekte finden sich schließlich als weiße *Blobs* in dem resultierenden Differenzbild wieder, welches der Segmentierungseinheit übergeben wird.

3.7 Das Segmentierungs-Modul

Als Input für das *Segmentierungs-Modul* dient in erster Linie das vom darüberliegenden *Hintergrund-Modell* gelieferte binäre Differenzbild. Das *Tracker Modul* liefert für bereits getrackte Objekte eine Schätzung ihrer Position, und erleichtert dadurch die Suche nach den korrespondierenden Blobs.

Vom *Real-World Modell* wird ein dem Straßenverlauf angepasstes Grid übernommen, welches benutzt wird, um einzelne Blobs besser segmentieren zu können. Dieses *Growing Grid* ist im Prinzip ein neues Koordinatensystem, ein feiner Raster, welcher ein neues Nachbarschaftsverhältnis zwischen den einzelnen Bildpunkten definiert. Dieser Raster ist dabei dem Straßenverlauf bzw. dem Verlauf des optischen Flusses im Bild angepasst und spannt sich über die gesamte *Region of Interest*. Der dahinter liegende Grundgedanke ist der, dass sich Fahrzeuge im Normalfall in einem gewissen geometrischen Verhältnis zur Straße befinden und dementsprechend in ein Bild abgebildet werden. Informationen über dieses Abbildungsverhältnis können weiters in einem Segmentierungsverfahren verwendet werden.

Die Segmentierung des Differenzbildes läuft rekursiv ab. Durch die vom *Tracker-Modul* gelieferte Schätzung der Größe und der Position eines Fahrzeugs wird ein

Ausschnitt des Differenzbildes definiert. Die einzelnen horizontalen Reihen im *Growing Grid* bilden Pixelreihen, so genannte *Scanlines*. Innerhalb des definierten Ausschnitts werden die Scanlines und die Anzahl der gesetzten Pixel mit einem Schwellwert verglichen. Ist die Anzahl der gesetzten Pixel größer als dieser Schwellwert, so wird der Mittelpunkt der Scanline als Ausgangspunkt eines Growing-Algorithmus benutzt.

Ausgehend von dem Knotenpunkt, der einen Ausgangspunkt bildet, werden jeweils die Pixelwerte der 4 benachbarten Knoten im *Growing Grid* untersucht. An jedem Knoten, an dem sich wieder ein gesetztes Pixel befindet, wird dieser Vorgang dann wiederholt. Schließlich ergibt sich der rechteckige Bildausschnitt, in welchem der Blob zur Gänze enthalten ist, durch den Mittelpunkt dieser Menge gesetzter Pixel und dem maximalen Durchmesser in horizontaler und vertikaler Richtung.

Nachdem sämtliche vom *Tracker-Modul* gelieferte Schätzungen bearbeitet wurden, werden noch mal sämtliche durch das *Growing Grid* im gesamten Bild definierten Scanlines untersucht. Finden sich unter diesen Scanlines welche, in denen eine gewisse Anzahl an Pixeln gesetzt sind und die nicht zu bereits abgearbeiteten Blobs gehören, so bilden diese wieder Ausgangspunkte für den Growing Algorithmus.

Schließlich sollten alle Objekte bestimmter Größe segmentiert worden sein. Die neu gewonnene Information über die Position der detektierten Objekte wird so an das *Tracker-Modul* weitergeleitet.

Der Grund, warum in diesem Modul kein übliches *Region Growing* oder *Region Labeling* (siehe dazu beispielsweise [SHB98]) verwendet wird, liegt einzig und allein im Bereich des erforderlichen Rechenaufwands. Die meisten Algorithmen lassen sich nur schwer in Echtzeit berechnen, und sind deshalb nur schwer im Bereich der Videoverarbeitung anwendbar. Das hier verwendete Verfahren ist aber äußerst schnell und kann auch bei Verwendung einer höheren Auflösung problemlos in Echtzeit angewendet werden.

3.8 Das Tracker-Modul

In diesem Modul wird für jedes einzelne bewegte Objekt ein Kalman-Tracker mit zwei Teilen verwendet. „*Kalman Tracker*“ nennt man einen Kalman Filter, der die Bewegung eines Objekts auf einer Systemgleichung basierend modelliert. Für eine nähere Beschreibung des Kalman Filters sei auf Anhang A.1 verwiesen.

Im ersten Teil des Kalman-Trackers wird die Position (x, y) eines vermeintlichen

Fahrzeugs in der Bildebene, im zweiten Teil die Größe (w, h) des umschreibenden Rechtecks modelliert. In dem Teil, in welchem die Position modelliert wird, wird eine Systemgleichung zweiter Ordnung verwendet, in der die Geschwindigkeit und die Beschleunigung berücksichtigt werden. Bezeichnet man beispielsweise die Geschwindigkeit in Richtung der x-Achse v_x mit

$$v_x = \frac{\partial x}{\partial t} \quad (3.1)$$

bzw. die Beschleunigung in Richtung der x-Achse a_x mit

$$a_x = \frac{\partial^2 x}{\partial^2 t} \quad , \quad (3.2)$$

dann ergibt sich folgende Systemgleichung für die x -Position:

$$\underbrace{\begin{pmatrix} \tilde{x} \\ \tilde{v}_x \\ \tilde{a}_x \end{pmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{pmatrix} 1 & t & \frac{t^2}{2} \\ 0 & 1 & t \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{pmatrix} x \\ v_x \\ a_x \end{pmatrix}}_{\mathbf{x}_{k-1}} \quad (3.3)$$

Ähnliche Gleichungen ergeben sich auch für den Teil des Trackers, der die Größe modelliert. Hier wird beispielsweise die Breite w , die Geschwindigkeit mit der sich diese ändert und ein Beschleunigungsfaktor berücksichtigt.

Die Schätzungen der Position und der Größe eines Fahrzeugs bilden den Ausgangspunkt für die Suche nach dem getrackten Fahrzeug in einem neuen Input-Differenzbild im *Segmentierungs-Modul*. Nach der Segmentierung werden diese “Messungen” als Update für die einzelnen Filter verwendet. Für jedes neue Objekt, über das vor der Segmentierung noch keine Information im *Tracker-Modul* vorhanden war, wird ein neuer Tracker initialisiert.

Durch die rekursive Natur des Tracking-Algorithmus kann eine History über die Bewegung eines Objekts aufgezeichnet werden, bis ein Objekt den Bildbereich verlässt.

Um eine ungefähre Berechnung der Position eines Objekts in der aufgenommenen Szene und eine Schätzung des Abstands zur Kamera durchführen zu können wird eine *Homographie* benötigt. Durch die Definition von 4 Punkten in der Szene, welche zumeist durch Bodenmarkierungen auf der Straße gegeben sind, und dem Vorwissen des genauen realen Abstands dieser Punkte zueinander, wird damit ein mathematischer Zusammenhang zwischen der Ebene, in welcher die Straße liegt, und einem Modell der Szene hergestellt. Für eine detaillierte Beschreibung der Berechnung

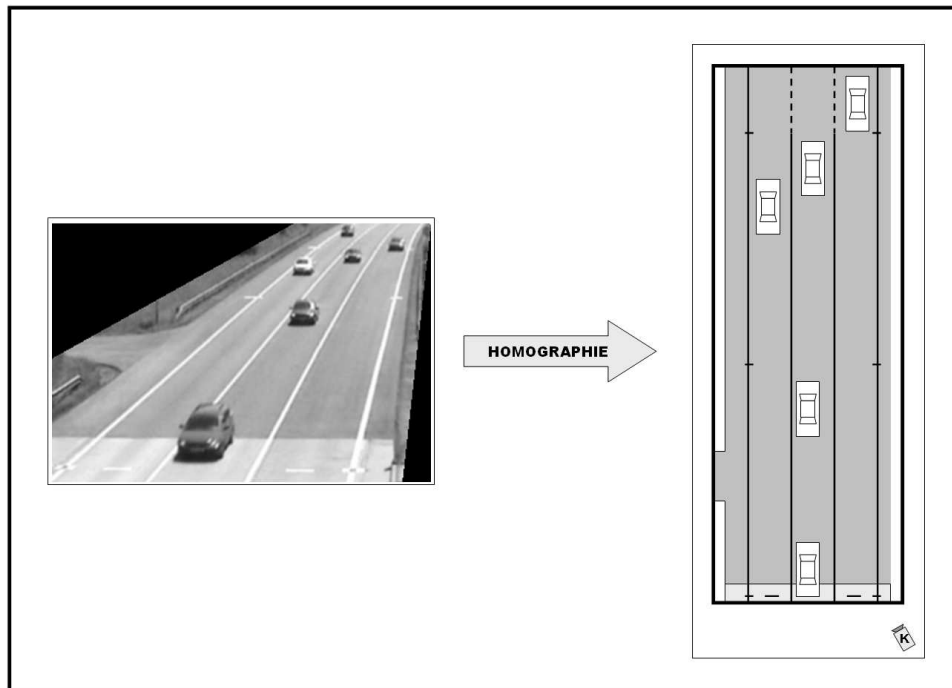


Abbildung 3.5: Beispiel für den Zusammenhang zwischen einem Bild und einem Modell der Szene, welcher durch die *Homographie* gegeben ist.

einer *Homographie* sei auf Anhang A.2 verwiesen. In Abbildung 3.5 ist zu sehen, welcher Zusammenhang durch die *Homographie* hergestellt wird.

Durch Kombination der *Homographie* und der Geschichte eines Objekts lässt sich die gesamte Bewegung durch die Szene nachvollziehen. Die Position eines Objekts in der Szene kann allein durch die Transformation des Mittelpunktes des segmentierten Blobs in einem einzelnen Bild errechnet werden. Unter Einbeziehung der History lassen sich weitere Informationen berechnen. So kann eine Schätzung der mittleren Geschwindigkeit eines Objekts errechnet werden, oder Information über die benutzten Fahrspuren oder getätigte Überholmanöver gewonnen werden.

Für die Weiterverarbeitung im *Klassifikations-Modul* werden jeweils Bilder von Fahrzeugen weitergeleitet, welche nahezu im gleichen Abstand zur Kameraposition aufgenommen werden. Die Idee ist die, alle Fahrzeuge möglichst weit im Vordergrund aufzunehmen, um eine möglichst hohe Auflösung zu bewahren.

Die Systemgleichungen sind relativ aufwendig gestaltet. Allerdings versetzen sie das System in die Lage, nach einer leichten Anpassung der Parameter der Filter prinzipiell in jeder Aufnahmesituation Fahrzeuge genau genug tracken zu können. Dies ist besonders wichtig, da dadurch Kamerapositionen, die einen sich von der

Kamera wegbewegenden Verkehrsfluss aufnehmen, genauso eingenommen werden können wie Positionen für seitliche Aufnahmen oder Aufnahmen von oben.

3.9 Das Klassifikations-Modul

Das Klassifikationsmodul hat die Aufgabe, ein vom *Tracker-Modul* geliefertes Bild zu verarbeiten und das darauf abgebildete Fahrzeug einer gewissen Kategorie zuzuordnen. Dabei wird das Bild als erstes auf die vordefinierte Größe des Klassifikators skaliert und anschließend eine entsprechende Menge an Features berechnet. Die verwendete Klassifikationsstruktur weist auf diesem Featurepool basierend dem Bild und damit dem abgebildeten Fahrzeug ein Label zu, das der vorliegenden Fahrzeugkategorie entspricht.

In Kapitel 4 wird die genaue Vorgangsweise zum Entwurf einer Klassifikator-Struktur und die genaue Funktion beschrieben.

3.10 Probleme beim Einsatz des Systems

Der Entwurf des gesamten Systems ist bewusst möglichst allgemein gehalten. Dadurch gibt es prinzipiell keine Einschränkungen, was den Blickwinkel auf eine Szene betrifft. Trotzdem sind die Einsatzmöglichkeiten deutlich beschränkt, wie die aufgetretenen Probleme zeigen.

Das *Bildaufnahme-Modul* ist rein für den Einsatz unter Tageslicht bzw. künstlichem Licht gedacht, deshalb ist ein Einsatz unter nächtlichen Bedingungen in dieser Form nicht möglich. Außerdem ist kein Algorithmus zur Stabilisierung des Kamerabildes implementiert, so dass nur Aufnahmepositionen verwendet werden können, an denen die Kamera keinen Erschütterungen ausgesetzt ist.

Die Implementierung des *Hintergrundmodells* enthält einen Algorithmus zur Behandlung von Schatten (siehe 2.3.3). Dadurch können Probleme, die durch den Schattenwurf von Fahrzeugen entstehen, größtenteils verhindert werden. Es können allerdings weder durch Lichtkegel von Fahrzeugscheinwerfern hervorgerufene Blendwirkungen, noch durch Fahrbahnmaterialien hervorgerufene Spiegelungen oder aus widrigen Wetterbedingungen resultierende Störungen behandelt werden.

Die Segmentierungseinheit kann einzelne reale Objekte gut voneinander trennen. Allerdings ist das Zusammenwirken des *Segmentierungs-* bzw. des *Tracker-Moduls* nicht optimal. Probleme, die durch im Laufe der Bewegung entstehende Verdeckun-

gen von Fahrzeugen gegeben sind, können nur teilweise bereinigt werden. Ebenso problematisch gestaltet sich die Behandlung von Stausituationen.

Trotz dieser Probleme zeigt das System unter Einhaltung der in Abschnitt 3.2 beschriebenen Randbedingungen durchaus eine gute Performance. Deshalb können die zum Entwurf der Klassifikationseinheit verwendeten Daten auch relativ problemlos aufgenommen werden.

KAPITEL 4

Featuregenerierung und Klassifikatorentwurf

4.1 Allgemeines

In diesem Kapitel wird der algorithmische Hintergrund der Featureberechnung und des Klassifikatorentwurfs in 3 Abschnitten behandelt.

Im ersten Abschnitt wird eine ausführliche Übersicht über die Algorithmen gegeben, die zur Featureberechnung verwendet werden. Auf eine Beschreibung verschiedener Repräsentationsarten und Transformationen folgt eine kurze Diskussion und Interpretation der verwendeten Featuretypen.

Das zweite Unterkapitel gibt eine Übersicht über den entworfenen Klassifikationsalgorithmus. Beginnend mit der Erläuterung des generellen Ansatzes wird eine detaillierte Erklärung der Funktion der einzelnen Schritte gegeben.

Der letzte Abschnitt ist einer Beschreibung der aus einzelnen Klassifikatoren aufgebauten Strukturen gewidmet, deren Performance in der Folge in Kapitel 5 evaluiert wird.

4.2 Entwurf und Berechnung der verwendeten Features

Das Aussehen eines Objekts kann zu einem großen Teil durch Kanten beschrieben werden. Während diese Kanten vor allem den Umriss eines Objekts ausdrücken

können, können sie zu einem gewissen Teil auch das textuelle Aussehen eines Objekts wiedergeben. Deshalb haben Kanten prinzipiell große Aussagekraft über das Erscheinungsbild eines Objekts.

Die Detektion von Kanten kann mitunter sehr effizient durchgeführt werden, allerdings reduzieren sich nahezu alle Verfahren auf ein mehr oder weniger komplexes Schwellwertproblem. Ein Detektionsalgorithmus stellt zumeist nur die *Existenz* einer Helligkeitsänderung allein fest, sagt aber nichts über die *Richtung* oder die *Stärke* der Änderung aus. Deshalb ist die einfache Detektion einer Kante allein nicht ausreichend. Ausgehend von dieser Tatsache muss auf eine andere Methode zur Repräsentation von Helligkeitsunterschieden zurückgegriffen werden, auf die der *Wavelet*-Transformation.

In der Computer Vision bilden so genannte *Haar Wavelets* der Definition nach eine Menge von natürlichen Basis-Funktionen, die die Differenz in der durchschnittlichen Helligkeit zwischen verschiedenen Regionen eines Bildes kodieren. Für eine detaillierte Beschreibung von Wavelets sei auf die Publikation von MALLAT [Mal89] verwiesen; eine Einführung über den Einsatz von Wavelets in der Computer Vision wird in den Publikationen von STOLLNITZ u.a. gegeben [SDS95a] [SDS95b].

Bei der Klasse der *Haar*-Features handelt es sich somit um einzelne Zahlenwerte, welche die Differenz zwischen den Pixelsummen in verschiedenen Bildausschnitten, die in einem bestimmten geometrischen Verhältnis zueinander stehen, kodieren. Die in Kapitel 2 bereits erwähnte Arbeit von VIOLA und JONES [VJ01] enthält einen interessanten Ansatz zur effizienten Berechnung dieser Pixelsummen, welcher hier aufgegriffen wird. Durch eine Transformation eines Inputbildes in eine andere Repräsentationsform ist es möglich, die Pixelsumme einer horizontal oder vertikal liegenden rechteckigen Fläche in **konstanter** Zeit zu berechnen. Diese Repräsentationsform wird von den Autoren das *Integral Image* genannt. Da diese Idee aber eigentlich einer viel früheren Arbeit von CROW [Cro84] über Texture Mapping entstammt, wird in dieser Arbeit der dort verwendete Begriff des *Summed Area Table* (SAT) verwendet.

Um die Pixelsummen in diagonal liegenden Flächen effizient berechnen zu können, wird auf einen weiteren Ansatz aus den Arbeiten von LIENHART, KURANOV, PISAREVSKY und MAYDT [LKP02] [LM02] zurückgegriffen. Darin wird eine weitere, auf der Idee des SAT basierende Transformation in eine andere Repräsentationsart vorgeschlagen, in den so genannten *Rotated Summed Area Table* (RSAT). Dadurch ist es möglich, auch um 45° gedreht liegende rechteckige Flächen zu denselben Konditionen zu berechnen, also ebenfalls in **konstanter** Zeit.

Abbildung 4.1 zeigt zwei Beispiele für die mögliche Lage eines Rechtecks in einem

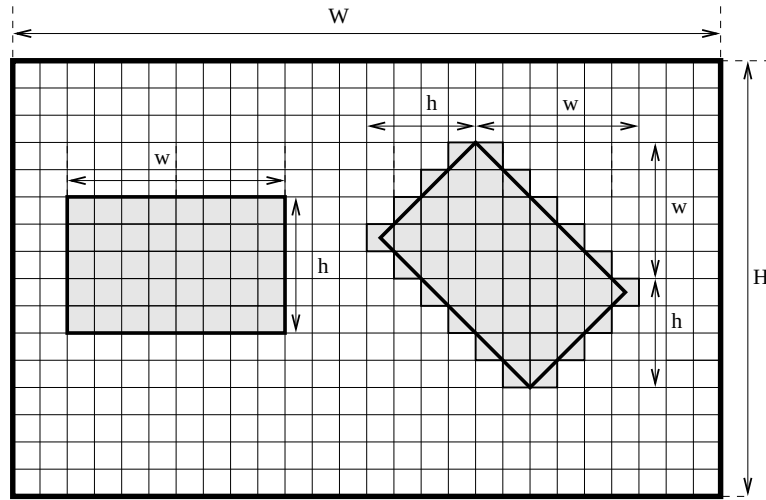


Abbildung 4.1: Beispiele für eine mögliche Lage eines normalen und eines um 45° verdrehten Rechtecks in einem Bild

Bild. Durch die beiden Transformationen wird der Grundstein für eine sehr effiziente Berechnung von Features gelegt, welche aus in solchen Rechtecken liegenden Pixelsummen gebildet werden. Dabei steht der Rechenaufwand dieser Transformationen selbst in konstantem Verhältnis zu der Anzahl der Bildpunkte in einem Bild, wie in den nächsten beiden Abschnitten erläutert wird.

4.2.1 Summed Area Table

Der Wert des *Summed Area Tables* berechnet sich an der Stelle $SAT(x, y)$ zu

$$SAT(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') . \quad (4.1)$$

Nutzt man die folgenden 2 Rekursionen

$$s(x, y) = s(x, y - 1) + I(x, y) \quad (4.2)$$

$$SAT(x, y) = SAT(x - 1, y) + s(x, y) , \quad (4.3)$$

wobei $s(x, y)$ die kumulative Reihensumme und

$$s(x, -1) = 0 \quad (4.4)$$

$$SAT(-1, y) = 0 \quad (4.5)$$

ist, kann der SAT in einem Durchlauf über ein Bild berechnet werden.

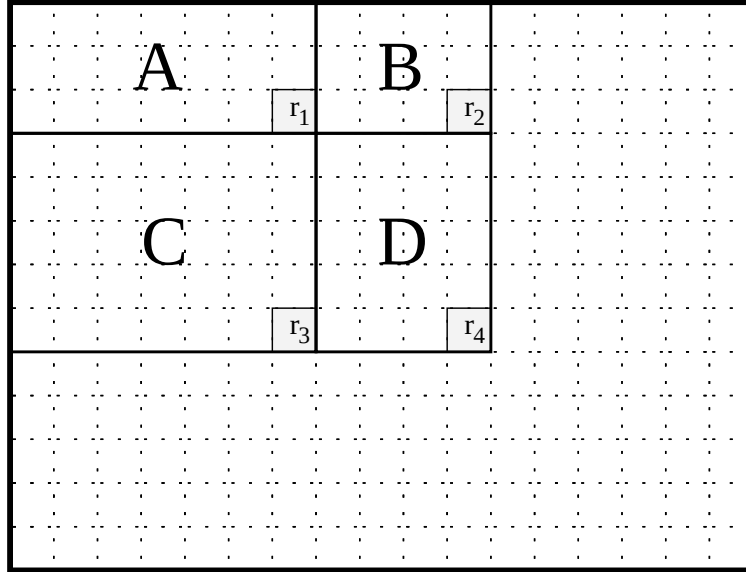


Abbildung 4.2: Berechnung der Pixelsumme in einem vorgegebenen Rechteck. Mit Hilfe der 4 Referenzen kann diese in konstanter Zeit berechnet werden (siehe Formel (4.6))

Nach der Transformation kann jede rechteckige Summe durch eine einfache Referenzierung von 4 Werten berechnet werden. Abbildung 4.2 illustriert dies.

Zur Berechnung der Pixelsumme im Rechteck D sind folgende 4 Referenzen notwendig:

- r_1 entspricht der Pixelsumme im Rechteck A
- r_2 entspricht der Summe der Rechtecke $A + B$
- r_3 entspricht $A + C$
- r_4 entspricht $A + B + C + D$.

Es ergibt sich für die Pixelsumme im Rechteck D somit

$$\text{pixsum}(D) = r_4 + r_1 - (r_2 + r_3) . \quad (4.6)$$

4.2.2 Rotated Summed Area Table

Der *Rotated Summed Area Table* berechnet sich an der Stelle $RSAT(x, y)$ zu

$$RSAT(x, y) = \sum_{x' \leq x, y' \leq y - |x - x'|} I(x', y') \quad (4.7)$$

und kann mit der Formel

$$RSAT(x, y) = RSAT(x - 1, y - 1) + RSAT(x + 1, y - 1) - RSAT(x, y - 2) + I(x, y) + I(x, y - 1) \quad (4.8)$$

und unter Zuhilfenahme folgender Bedingungen

$$RSAT(-1, y) = RSAT(x, -1) = RSAT(x, -2) = 0 \quad (4.9)$$

$$RSAT(-1, -1) = RSAT(-1, -2) = 0 \quad (4.10)$$

in einem Durchlauf über ein Bild berechnet werden (Abbildung 4.3).

Hiermit kann die Pixelsumme eines beliebigen um 45° rotierten Rechtecks $rect = (x, y, w, h, 45^\circ)$ durch 4 Referenzen berechnet werden (Abbildung 4.4):

$$\begin{aligned} \text{pixsum}(rect) = & RSAT(x - h + w, y + w + h - 1) + \\ & + RSAT(x, y - 1) - RSAT(x - h, y + h - 1) \\ & - RSAT(x + w, y + w - 1) . \end{aligned} \quad (4.11)$$

4.2.3 Featuretypen und Interpretation

Mit Hilfe der neuen Repräsentationsarten können Features, die auf Pixelsummen in Rechtecken basieren, aus einem Bild sehr effektiv berechnet werden. Das Layout der einzelnen Features hängt dabei von den zu suchenden Strukturen ab.

Abbildung 4.5 zeigt Prototypen der 15 in dieser Arbeit verwendeten Featurearten. 8 davon können unter Zuhilfenahme des SAT berechnet werden, die restlichen 7 über Einbeziehung des RSAT. Weiters können die Features folgendermaßen unterteilt werden:

Edge Features: Grundsätzlich stellt die Berechnung dieser Gruppe einen einfachen Kantendetektionsalgorithmus dar. Dabei erfüllen die Typen (a) und (b) ihre

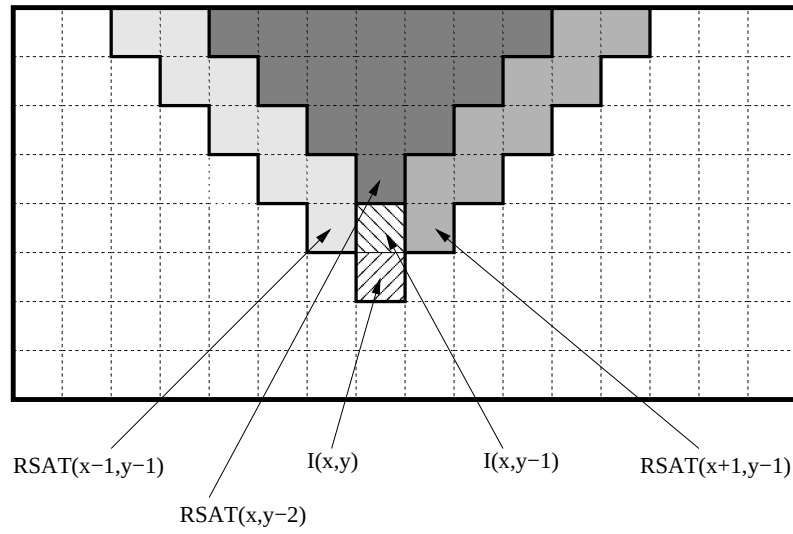


Abbildung 4.3: Berechnung des Rotierten Integral Images in einem Durchlauf nach Gleichung (4.8)

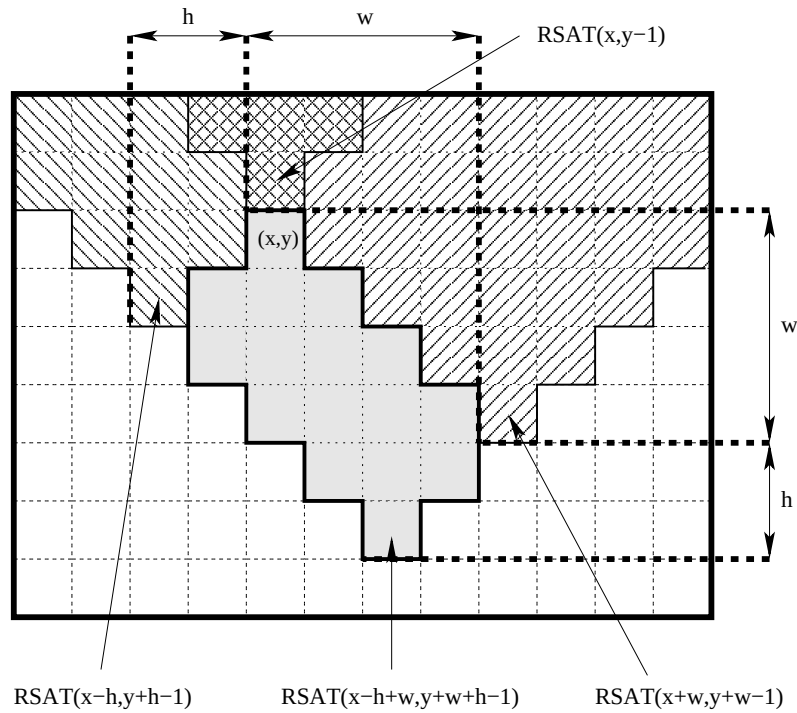


Abbildung 4.4: Berechnung der Pixelsumme in einem rotierten Rechteck mit Hilfe des Rotierten Integral Images und unter Verwendung von Gleichung (4.11)

Funktion in horizontaler und vertikaler Richtung, die Typen (c) und (d) hingegen in diagonalen Richtung.

Line Features: Sie sollen auf strichartige Strukturen ansprechen. Die Gattungen (a), (c), (e) und (g) werden zur Detektion dünner, die Gattungen (b), (d), (f) und (h) zur Detektion dicker linienförmiger Merkmale verwendet.

Center Surround Features: Das Ziel dieser Features ist es, auf besondere punktförmige Merkmale zu reagieren.

Corner Feature: Als spezielles Feature ist dieser Typ als Ersatz für alle diagonalen Features verwendbar, im Besonderen für die diagonalen Edge Features, da es sowohl auf diagonale Kanten, als auch auf markante Ecken anspricht.

Basierend auf der Anzahl zusammenhängender Regionen gleichen Vorzeichens, sind zur Berechnung der Features eine gewisse Anzahl an Zugriffen auf die Repräsentationen SAT und RSAT vonnöten. Wie leicht nachzuvollziehen ist, können *Edge* Features durch 6 Zugriffe berechnet werden, während für *Line* und *Center Surround* Features 8 und für das *Corner* Feature 9 Referenzierungen benötigt werden.

4.2.4 Datenpool und Selektion

Die Gesamtheit aller auf ein Bild bestimmter Größe berechenbaren Features entsteht aus der theoretischen Möglichkeit, jeden dieser Prototypen, ausgehend von einer Basisgröße, in Breite und Höhe beliebig ganzzahlig zu skalieren und an jeder möglichen Stelle im Bild zu berechnen. Dies ist allerdings nicht unbedingt notwendig.

In den Arbeiten von LIENHART werden zwei Featuresets für das Detektion-Problem definiert und deren Performance evaluiert. Dieselben Gruppen von Features werden auch hier verwendet, allerdings für das Klassifikationsproblem.

Das erste Featureset wird durch die Arbeiten von VIOLA & JONES motiviert und wird in der Folge **Viola-Set** genannt. Es besteht aus den 5 Featuretypen 1(a)(b), 2(a)(c) und 4. Da keine diagonalen Features berechnet werden, braucht auch der *Rotated Summed Area Table* nicht berechnet zu werden. Dadurch wird der Berechnungsaufwand deutlich reduziert.

Die zweite Gruppe von Features wird von LIENHART preferiert und deshalb mit **Lienhart-Set** bezeichnet. Bis auf das Feature des Typs 4 zählen alle Featuretypen zu dieser Gruppe. Zur Berechnung werden beide Repräsentationsarten, SAT und RSAT, benötigt. Durch die verhältnismäßig große Anzahl unterschiedlicher Featu-

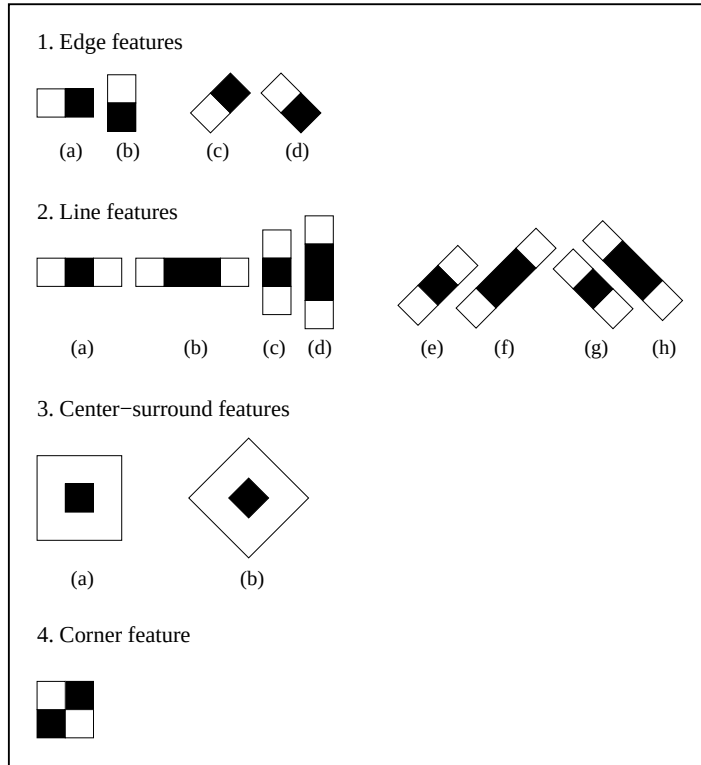


Abbildung 4.5: Prototypen der 15 verwendeten Features; schwarze Flächen haben negatives, weiße Flächen positives Vorzeichen

Feature-Typ	Viola-Set			Lienhart-Set		
	24 × 24	48 × 48	64 × 64	24 × 24	48 × 48	64 × 64
1.(a)(b) je	43.200	677.376	2.129.920	43.200	677.376	2.129.920
1.(c)(d) je	—	—	—	11.572	202.952	655.712
2.(a)(c) je	27.600	442.176	1.397.760	27.600	442.176	1.397.760
2.(e)(g) je	—	—	—	7.028	129.480	423.150
2.(b)(d) je	—	—	—	19.800	324.576	1.031.680
2.(f)(h) je	—	—	—	4.780	92.840	307.040
3.(a)	—	—	—	8.464	141.376	451.584
3.(b)	—	—	—	1.932	39.480	132.090
4.	20.736	331.776	1.048.576	—	—	—
Summe:	162.336	2.570.880	8.103.936	238.356	3.919.656	12.474.198

Tabelle 4.1: Beispiel für die Größe der möglichen Featurepools für 3 Bilder unterschiedlicher Größe. In der linken Spalte ist die Bezeichnung des Featuretyps aus Abbildung 4.5 aufgetragen, In den Spalten 2 bis 5 ist die Anzahl der Features für das **Viola-Set**, in den rechten drei Spalten die Anzahl der berechenbaren Features für das **Lienhart-Set** aufgetragen.

reotypen resultiert auch ein deutlich größerer Berechnungsaufwand.

Selbst nach einer Einteilung in zwei Feature-Sets ist die Größe der jeweiligen resultierenden Featurepools enorm. In Tabelle 4.1 wird aufgelistet, welche Anzahl an Features aus der Verwendung der Featuregruppen für zwei verschiedene Bildgrößen entstehen würden. Diese Featurepools enthalten einerseits Unmengen redundanter Information, andererseits bilden sie technisch nur sehr schwer bewältigbare Datenmengen. Deshalb ist es empfehlenswert, die jeweiligen Prototypen bei einer vordefinierten Bildgröße nur an bestimmten Positionen und in festgesetzten Skalierungen zu berechnen. Dadurch kann der erforderlichen Speicheraufwand und der ohnehin nicht zu unterschätzende Rechenaufwand in Grenzen gehalten werden.

4.3 Klassifikatorentwurf

Grundlage des gesamten Klassifikationsschemas bildet der *ADA-Boost* Algorithmus. Eine detaillierte Beschreibung der verwendeten Version wird in den nächsten beiden Abschnitten gegeben.

4.3.1 Der ADA-Boost Algorithmus

Der *ADA-Boost* Algorithmus wurde von SCHAPIRE und FREUND entwickelt [SF95] [SF96]. Diesem Algorithmus liegt die Idee zugrunde, eine Gruppe von schwachen Klassifikatoren, so genannten *Weak Classifiers*, die nur unwesentlich bessere Entscheidungen treffen als Zufallsentscheidungen, zu einem einzelnen starken Klassifikator zu kombinieren, dessen Performance wesentlich besser ist als die jedes einzelnen aus dieser Gruppe.

In Abbildung 4.6 ist der etwas modifizierte ADA-Boost Algorithmus zu sehen, der durch die Arbeit von VIOLA und JONES [VJ01] angeregt wird und hier Verwendung findet. Der Algorithmus übernimmt sowohl die Auswahl einer vorgegebenen Anzahl T signifikanter Features, als auch das Training eines Klassifikators.

Ausgehend von einem Trainingsdatensatz an Tuples (x_i, y_i) , welche je aus einem Bild und dem Label der Fahrzeugkategorie bestehen, wird für jedes einzelne Tupel eine von der Anzahl der positiven und negativen Beispielen abhängige Anfangsgewichtung $w_{1,i}$ berechnet. Für einen starken Klassifikator, welcher aus T schwachen Klassifikatoren besteht (Level- T Klassifikator), wird folgende Schleife T mal durchlaufen.

- Gegeben: Satz Trainingsbilder $(x_1, y_1), \dots, (x_n, y_n)$, wobei $y_i = 1$ für positive und $y_i = 0$ für negative Beispiele
- Initialisiere die Gewichte für jedes Trainingsbeispiel i , wobei l und m die Anzahl an positiven und negativen Beispielen ist
$$w_{1,i} = \begin{cases} \frac{1}{2l} & \text{falls } y_i = 1 \\ \frac{1}{2m} & \text{falls } y_i = 0 \end{cases}$$
- Für einen Klassifikator bestehend aus T schwachen Klassifikatoren:
für $t = 1 \dots T$

- ▣ Normalisiere die Gewichte

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

sodaß eine Wahrscheinlichkeitsverteilung w_t entsteht

- ▣ Für jedes Feature j trainiere einen schwachen Klassifikator h_j ,
der nur ein einzelnes Feature verwenden darf und berechne den Fehler
in Bezug auf w_t

$$\epsilon_j = \sum_i (w_i \cdot |h_j(x_i) - y_i|)$$

- ▣ Selektiere den schwachen Klassifikator h_j mit dem geringsten Fehler ϵ_j

$$h_t = h_j \quad \epsilon_t = \epsilon_j$$

- ▣ Berechne die Gewichte der Trainingsbeispiele neu

$$w_{t+1,i} = w_{t,i} \cdot \beta_t^{1-e_i} \quad \text{mit} \quad \beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$$

$$\text{und} \quad e_i = \begin{cases} 0 & \text{wenn } h_t \text{ Beispiel } i \text{ korrekt klassifiziert} \\ 1 & \text{wenn } h_t \text{ Beispiel } i \text{ nicht korrekt klassifiziert} \end{cases}$$

- Der gesamte starke Klassifikator ergibt sich damit zu

$$h(x) = \begin{cases} 1 & \text{falls } \sum_{t=1}^T \alpha_t \cdot h_t(x) \geq p_{thr} \sum_{t=1}^T \alpha_t \\ 0 & \text{sonst} \end{cases} \quad \text{mit} \quad \alpha_t = \log \frac{1}{\beta_t}$$

$$p(x=1) = \begin{cases} 0.5 \cdot \left(\frac{\left(1 - 2p_{thr} + \left(\frac{\sum_{t=1}^T \alpha_t \cdot h_t(x)}{\sum_{t=1}^T \alpha_t} \right) \right)}{1 - p_{thr}} \right) & \text{wenn } h(x) = 1 \\ 0.5 \cdot \left(\frac{\left(\frac{\sum_{t=1}^T \alpha_t \cdot h_t(x)}{\sum_{t=1}^T \alpha_t} \right)}{p_{thr}} \right) & \text{wenn } h(x) = 0 \end{cases}$$

Abbildung 4.6: Modifizierter ADA-Boost Algorithmus für einen starken Klassifikator bestehend aus T schwachen Klassifikatoren (Level- T Klassifikator).

Beginnend mit einer Normalisierung der Gewichte zu einer Wahrscheinlichkeitsverteilung wird für jedes einzelne Feature ein schwacher Klassifikator trainiert (siehe Abschnitt 4.3.2) und dessen Fehler auf den Trainingsdatensatz evaluiert. Dabei wird eine Fehlklassifikation durch die verschiedene Gewichtung der einzelnen Trainingsbeispiele unterschiedlich bewertet. Schließlich wird jener schwache Klassifikator h_j aus der gesamten Menge herausgegriffen, der den geringsten Fehler ϵ_j aufweist und eine Umverteilung der Gewichte anhand der richtigen oder falschen Klassifikation durch dieses eine Modul vorgenommen. Dadurch wird erreicht, dass mit jedem Durchlauf *schwere* Beispiele stärker gewichtet werden und im nächsten Schritt auf die *richtige* Klassifikation genau dieser Tupels mehr Wert gelegt wird.

Nach Durchlaufen der Schleife ergibt sich der Level- T Klassifikator $h(x)$ durch eine serielle Aneinanderreihung der einzelnen schwachen Klassifikatoren, wobei jedes dieser Module eine fehlerabhängige logarithmische Gewichtung erhält. Dabei kann ein Wahrscheinlichkeitsmaß definiert werden, die angibt, mit welcher Wahrscheinlichkeit $p(x)$ ein Beispiel eine positive Klassifikation $x = 1$ erhält. Eine Wahrscheinlichkeit, die größer als oder gleich einer vorgegebenen Wahrscheinlichkeit p_{thr} ist, führt automatisch zu einer Klassifikation $h(x) = 1$.

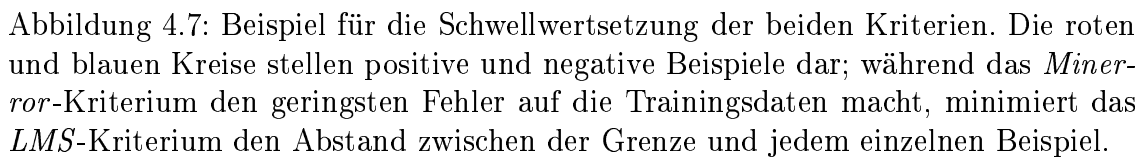
4.3.2 Die Funktion der schwachen Klassifikatoren

Da ein schwacher Klassifikator bei Verwendung des in Abbildung 4.6 beschriebenen Boostings nur ein einzelnes Feature verwenden darf, handelt es sich um eine eindimensionale Problemstellung. Während eine ganze Reihe von Algorithmen zur Lösung verwendet werden können, werden hier die folgenden 2 Ansätze vorgeschlagen, die zum Training und zur Optimierung eines *Weak Classifiers* bei einer gegebenen Menge von N Tuples verwendet werden können:

LMS Kriterium: Unter Verwendung des *Least-Mean-Square* Kriteriums wird derjenige Schwellwert θ_{min} gesucht, bei dem der quadratische Fehler zwischen den Werten der Features f in den einzelnen Trainingsbildern i und der gezogenen Grenze θ minimal ist.

$$\theta_{min} = \min_{\theta} \sum_{i=1}^N \|f_i - \theta\|^2 \quad (4.12)$$

Dadurch wird nicht unbedingt der kleinste mögliche Fehler auf dem Trainingsdatensatz erzielt, prinzipiell ist aber zu erwarten, dass dadurch Features ausgewählt werden, die einerseits diskriminant sind und deren Werte gleichzeitig kompaktere



Minimum Error Kriterium: Hier wird der optimale Schwellwert θ_{min} gesucht, damit die *minimale Anzahl* an Trainingsbeispielen falsch klassifiziert wird.

$$\theta_{min} = \min_{\theta} \sum_{i=1}^N |h_{\theta}(f_i) - y_i| \quad (4.14)$$

Falls es einen optimalen Schwellwert gibt, an dem kein einziges Trainingsbeispiel

falsch klassifiziert wird, dann wird dieser bei Verwendung dieses Kriteriums gefunden. Die Performance hängt allerdings stark von der Repräsentationsfähigkeit der Trainingsdaten ab. Ist der Trainingsdatensatz wenig repräsentativ, so kann unter Umständen ein optimaler Schwellwert leicht gefunden werden, die Performance auf einem Testdatensatz muss deshalb aber nicht unbedingt gut sein.

In Abbildung 4.7 wird die unterschiedliche Verhaltensweise der beiden Ansätze demonstriert.

4.4 Evaluierete Strukturen zur Multi-Class Klassifikation

In Abschnitt 4.3 wurde die Vorgangsweise zum Bau eines binären Klassifikators besprochen. Da es mehr als zwei Klassen von Fahrzeugen zu unterscheiden gilt, ist eine Kombination einzelner Module notwendig, um die gestellte Aufgabe erfüllen zu können. Durch die oben beschriebene Methode werden einzelne binäre Klassifikatoren trainiert und diese anschließend in zwei verschiedenen Strukturen angeordnet, welche in der Folge kurz beschrieben werden.

4.4.1 Die *Baum*-Struktur

In Abbildung 4.8 ist die *Baum*-Struktur aus einzelnen Klassifikatoren zu sehen. Hier wird zuerst eine größenabhängige Klassifikation durchgeführt und im Anschluss daran eine Einordnung in die jeweilige Fahrzeugkategorie.

Die einzelnen Klassifikatoren werden jeweils mit positiven und negativen Beispielen für die zu treffende Entscheidung trainiert. Die jeweilige Anzahl an Features (Level M, N und O) der einzelnen Klassifikatoren beschreibt nur die Unabhängigkeit der einzelnen Module voneinander, sie kann also unabhängig voneinander gewählt werden.

4.4.2 Die *1-vs-All*-Struktur

Abbildung 4.9 zeigt die andere Anordnung binärer Klassifikatoren. 4 unabhängige Module berechnen parallel die Wahrscheinlichkeit, dass ein Inputbild zu einer gewissen Fahrzeugkategorie gehört. Die maximale Zugehörigkeitswahrscheinlichkeit ergibt schließlich die Zuordnung des Bildes zu einer Klasse.

4.4. EVALUIERTE STRUKTUREN ZUR MULTI-CLASS KLASSIFIKATION

48

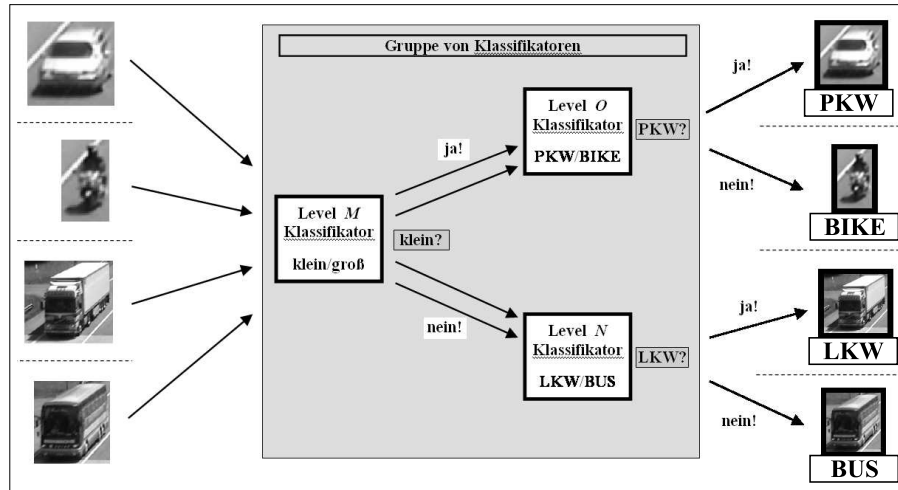


Abbildung 4.8: Baum-Struktur zur Multi-Class Klassifikation.

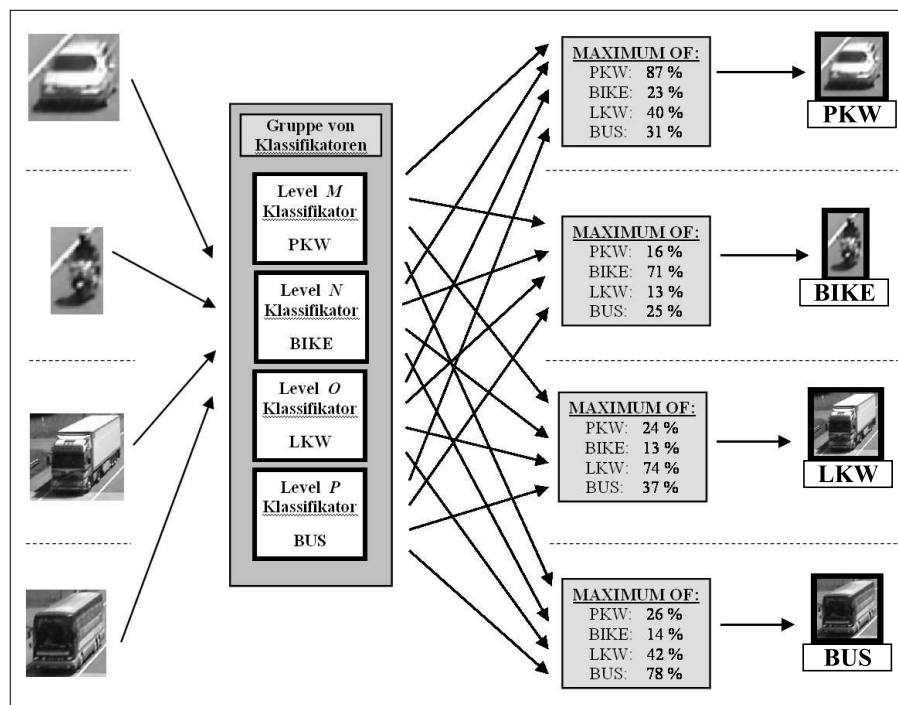


Abbildung 4.9: 1-vs-All-Struktur zur Multi-Class Klassifikation.

Zum Training der einzelnen Klassifikatoren werden jeweils Gruppen positiver Beispiele, also der jeweiligen Klasse zugehöriger Fahrzeuge, und Gruppen negativer Beispiele, eben nicht zu der Klasse gehöriger Fahrzeuge, verwendet. Auch hier ist die Anzahl der Features (Level M, N, O und P) frei wählbar und unabhängig voneinander.

4.4.3 Vor- und Nachteile der einzelnen Strukturen

Während die Zuordnung eines Inputbildes zu einer Fahrzeugkategorie durch die *Baum*-Struktur einen rechnerischen Vorteil gegenüber der zweiten evaluierten Struktur hat, hat die serielle Anordnung der Klassifikatoren einen entscheidenden Nachteil. Für die Kategorisierung eines Inputbildes müssen nur 2 Klassifikationen durchgeführt werden, was nur den halben Rechenaufwand im Vergleich zur *1-vs-All*-Struktur darstellt. Da eine Einordnung in eine Fahrzeugkategorie aber nur durch zwei serielle binäre *ja/nein*-Entscheidungen getroffen wird, ist eine Fehlentscheidung in der ersten Stufe der Klassifikatorengruppe nicht mehr korrigierbar und führt unweigerlich zu einer Fehlklassifikation.

Die *1-vs-All*-Struktur benötigt für die Zuordnung eines Inputbildes zu einer Fahrzeugkategorie 4 Klassifikationen und damit den doppelten Rechenaufwand wie die *Baum*-Struktur. Die parallele Anordnung der Module macht allerdings eine wahrheitswahrscheinlichkeitsbasierte Entscheidungsfindung möglich. Aus diesem Grund kann die Einordnung eines Inputbildes in eine Klasse anhand der maximalen Zugehörigkeitswahrscheinlichkeit einen deutlich positiven Einfluss auf die Performance dieser Struktur haben.

KAPITEL 5

Experimente und Resultate

5.1 Allgemeines

In diesem Abschnitt werden die in Kapitel 4 ausführlich beschriebenen Verfahren und Strukturen in Bezug auf verschiedene Kriterien evaluiert.

Zu Beginn wird die Menge der Fahrzeugbilder beschrieben, welche zum Training und zum Test der Algorithmen verwendet wird, und Beispiele aus den jeweiligen Fahrzeugkategorien gezeigt.

In Abschnitt 5.3 werden verschiedene Experimente durchgeführt. Es wird unter anderem ein Vergleich der Weak Classifiers durchgeführt und die Abhängigkeit zwischen dem Klassifikator-Level und der Performance untersucht. Außerdem wird die Performance in Abhängigkeit der Anzahl an Trainingsdaten untersucht.

Die universelle Einsatzmöglichkeit der Algorithmen wird in den Abschnitten 5.4.1 und 5.4.2 auf den von den beiden anderen Kamerapositionen aus aufgenommenen Daten demonstriert.

Den Schluss in Abschnitt 5.5 bildet eine Zusammenfassung der Ergebnisse und der daraus resultierenden Erkenntnisse.

5.2 Beschreibung der verwendeten Trainings- und Testdaten

Datenaufnahme: Zur Aufnahme der Fahrzeugbilder wurden drei verschiedene Kamerapositionen und damit drei grob verschiedene Blickwinkel auf die Fahrbahn

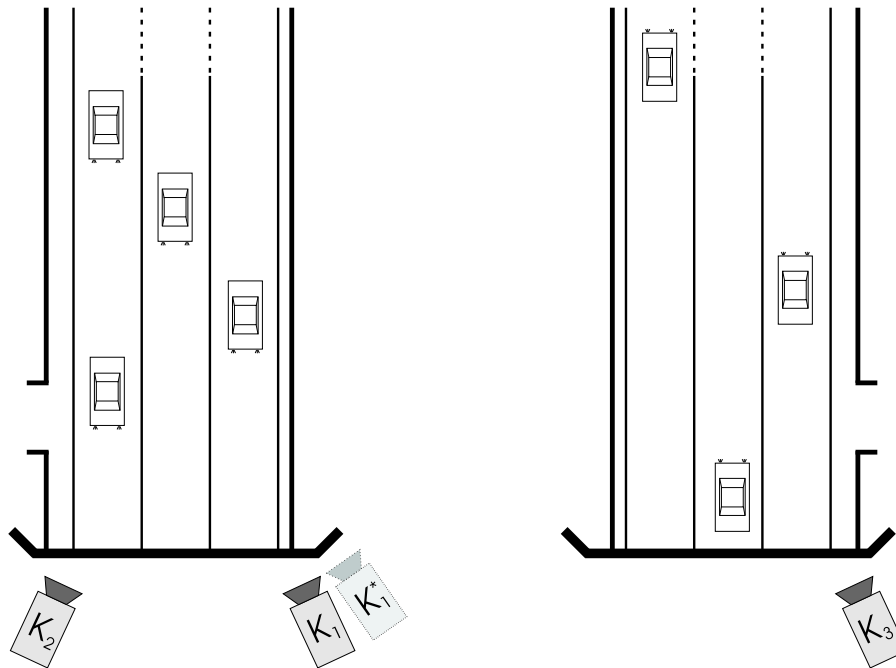


Abbildung 5.1: Drei unterschiedliche Standorte der Kamera für die Aufnahme von Daten. Von den ersten beiden Positionen aus wird der Verkehrsfluss von vorne, bei der dritten Position von hinten aufgenommen.

ausgewählt. Die ersten beiden Positionen befanden sich jeweils rechts und links der Fahrbahn und ermöglichten die Aufnahme des Verkehrsflusses von vorne. Die dritte Position wurde so gewählt, dass der Verkehrsfluss auf der Gegenfahrbahn gefilmt wurde, die Fahrzeuge also von hinten aufgenommen wurden. In Abbildung 5.1 sind die verschiedenen Kamerapositionen gegenüber der Fahrbahn zu sehen.

Ausgehend von den 3 Kamerapositionen wurden an mehreren Tagen bei gleich bleibendem Blickwinkel zur Fahrbahn und veränderlichem Sonnenstand einige Stunden Videomaterial aufgezeichnet und Fahrzeugbilder daraus extrahiert. Für Kameraposition 1 wurden für einen besonderen Test unter einem leicht veränderten Blickwinkel (ca. 5°) weitere Daten aufgenommen.

Kategorisierung: Nach der Extraktion der Fahrzeugbilder wurde von Hand eine Zuordnung von Labels zu den Bildern gemäß der abgebildeten Fahrzeugkategorie vorgenommen. Die Abbildungen 5.2 - 5.5 zeigen Beispiele aus den gesammelten Daten für die jeweilige Kategorie und die erste Kameraposition. In Tabelle 5.1 ist die gesamte Anzahl der aufgenommenen Bilder für die einzelnen Positionen aufgelistet.

Durch die Einschränkung auf 4 zu unterscheidende Fahrzeugkategorien gestaltet sich



Abbildung 5.2: Beispiele für Fahrzeuge aus der PKW Kategorie

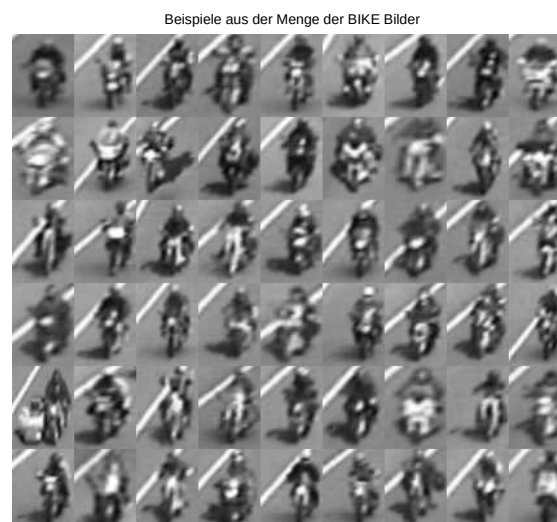


Abbildung 5.3: Beispiele für Fahrzeuge aus der BIKE Kategorie



Abbildung 5.4: Beispiele für Fahrzeuge aus der LKW Kategorie

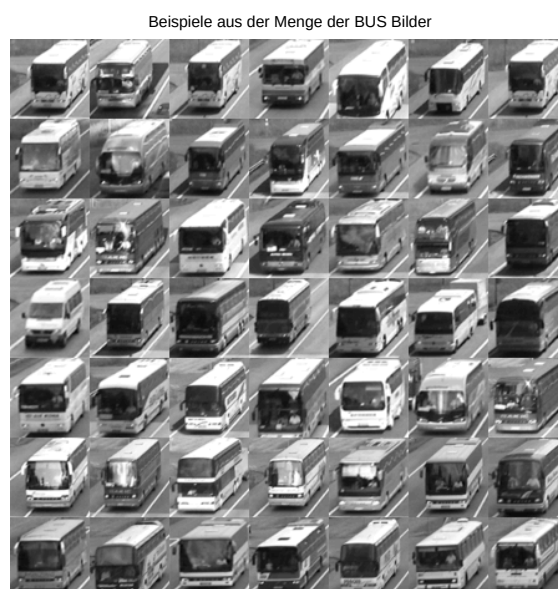


Abbildung 5.5: Beispiele für Fahrzeuge aus der BUS Kategorie

Kategorie	Pos. 1	Pos. 1*	Pos. 2	Pos. 3
PKW	20329	3368	5000	3853
BIKE	242	23	63	4
LKW	1555	444	571	529
BUS	94	12	20	11
Gesamt:	22220	3847	5654	4397

Tabelle 5.1: Anzahl aufgenommener Fahrzeuge verschiedenen Typs für die drei verschiedenen Kamerapositionen

Feature-Typ	Viola-Set		Lienhart-Set	
	genutzt	möglich	genutzt	möglich
1.(a)(b) je	7.396	677.376	7.396	677.376
1.(c)(d) je	—	—	5.360	202.952
2.(a)(c) je	3.698	442.176	3.698	442.176
2.(e)(g) je	—	—	2.672	129.480
2.(b)(d) je	—	—	3.526	324.576
2.(f)(h) je	—	—	5.060	92.840
3.(a)	—	—	6.889	141.376
3.(b)	—	—	4.506	39.480
4.	7.396	331.776	—	—
Summe:	29.584	2.570.880	66.819	3.919.656

Tabelle 5.2: Gegenüberstellung der benutzten und der möglichen Anzahl an Features für die festgelegte Bildgröße von 48×48 Pixel. Die Mengen der benutzten Features sind deutlich kleiner als die ursprünglichen Pools.

die Zuordnung von Fahrzeugen, die eigentlich zu keiner Kategorie gehören oder zu mehreren Kategorien gleichzeitig gehören können, zum Teil problematisch. Deshalb werden beispielsweise Wohnwägen und Lieferwägen prinzipiell zur Klasse der PKWs gezählt, obwohl diese durchaus größen- und aussehensmäßig in die Kategorie der LKW eingeordnet werden können.

Bildgröße: Für die Wahl der Bildgröße, die zum Entwurf der Klassifikationsstrukturen verwendet wird, sind folgende zwei Aspekte ausschlaggebend. Für den Betrachter sollen die geometrischen und die aussehensmäßigen Merkmale der abgebildeten Fahrzeuge auch bei einer Reduzierung der Bildgröße zu großen Teilen erkennbar bleiben. Auf der anderen Seite muss auf die beschränkten technischen Speicher- und Rechenressourcen Rücksicht genommen werden.

Beide Gesichtspunkte gleichermaßen berücksichtigend wird deshalb für alle folgen-

den Experimente eine Standard-Bildgröße von 48×48 Pixel gewählt. Als Vorverarbeitungsschritt werden alle Bilder in 8-bit Grauwertbilder konvertiert und auf die festgelegte einheitliche Größe skaliert.

Featurepools: Um die Größe der Featurepools zu reduzieren, müssen weitere Einschränkungen bezüglich der Berechnung der Prototypen getroffen werden.

Für alle Experimente werden deshalb folgende Parameter festgelegt. Ausgehend von der erwähnten Bildgröße wird der Skalierungsfaktor so eingeschränkt, dass die Features maximal bis zu einem Fünftel der Bildgröße skaliert werden (maximale Dimension eines Features damit 9×9 Pixel). Weiters wird eine fixe Anfangsskalierung definiert und eine Schrittweite in der Skalierung festgelegt. Eine Definition einer Anfangsposition und von Schrittweiten in x- und y-Richtung schränkt schließlich die Berechnung von Features auf bestimmte Positionen im Bildbereich ein.

Die angeführten Reduktionen zusammengefasst resultieren in Featurepools, die wesentlich kleiner sind als die Mengen aller berechenbaren Möglichkeiten, für das **Viola-Set** nur etwa 1.15% und für das **Lienhart-Set** etwa 1.7% der ursprünglichen Größe. Tabelle 5.2 werden die resultierenden Featurezahlen pro Bild aufgelistet, die für das weitere Training und die anschließende Evaluierung verwendet werden.

5.3 Detailanalyse der Algorithmen

In diesem Abschnitt werden verschiedene Experimente durchgeführt, die Aufschluss über das Verhalten der beschriebenen Algorithmen unter verschiedenen Bedingungen geben sollen. Dazu werden ausschließlich Daten von Kameraposition 1 und 1* benutzt.

5.3.1 Aufteilung der Daten

Für die folgenden Experimente werden die Daten von Kameraposition 1 folgendermaßen aufgeteilt:

Es werden 3 Trainingsdatensätze gebildet, wobei der 1. Datensatz der kleinste, der 2. fünfmal und der 3. Trainingsdatensatz zehnmal so groß wie der 1. Datensatz ist. Die restlichen Daten werden als Testdaten verwendet, wobei aus der überdimensional großen Menge von PKWs nur ein Bruchteil herausgenommen wird, um das Ergebnis der einzelnen Fehler-Berechnungen nicht zu sehr zu verfälschen. Außerdem

	Trainm. 1	Trainm. 2	Trainm. 3	Testm.	Testm. 1*
PKW	80	400	800	5425	3368
BIKE	16	80	160	82	23
LKW	40	200	400	1155	444
BUS	6	30	60	34	12
Gesamt	132	710	1420	6696	3847

Tabelle 5.3: Trainings- und Testdatenmengen für jede einzelne Fahrzeugklasse für die folgenden Experimente.

wird nur diese einzige Testmenge definiert, um einen für *alle* Experimente *eindeutig* festgelegten Testdatensatz verwenden zu können.

Die von Kameraposition 1* aus aufgenommenen Daten bilden eine spezielle Testdatenmenge. Eine genaue Auflistung der einzelnen Zahlen ist in Tabelle 5.3 zu sehen.

5.3.2 Vergleich der Weak Classifiers

In diesem Experiment werden die verwendeten Weak Classifiers, das **LMS**- und das **Minimum Error** Kriterium, miteinander verglichen. Ziel ist es, eine generelle Aussage über die Performanceunterschiede der beiden Ansätze treffen zu können.

Durchführung: Aus den beiden unterschiedlichen Strukturen werden sämtliche Einzelklassifikatoren herausgenommen. Diese Klassifikatoren werden mit der Trainingsdatenmenge 3 unter Verwendung der unterschiedlichen Featuresets und der unterschiedlichen Weak Classifier Kriterien trainiert. Durch eine Evaluierung auf den Testdaten wird die Performance der Klassifikatoren miteinander verglichen.

Ergebnisse: In Abbildung 5.6 sind die ROC Kurven (*Receiver Operator Characteristic Curves*) für zwei ausgewählte Klassifikatoren bei Verwendung der verschiedenen Weak Classifier Kriterien und der beiden Featuresets zu sehen. Als Parameter wird hier der Schwellwert über der Wahrscheinlichkeit p_{thr} zwischen positiver und negativer Klassifikation von 0% bis 100% variiert. Die Klassifikatoren basieren dabei auf der Berechnung von 30 Features (Level 30 Klassifikatoren).

In Tabelle 5.4 ist die resultierende *Equal Error Rate* (EER) für die einzelnen Klassifikatoren zu sehen. Der kleinere Fehler ist dabei jeweils **fett** herausgehoben.

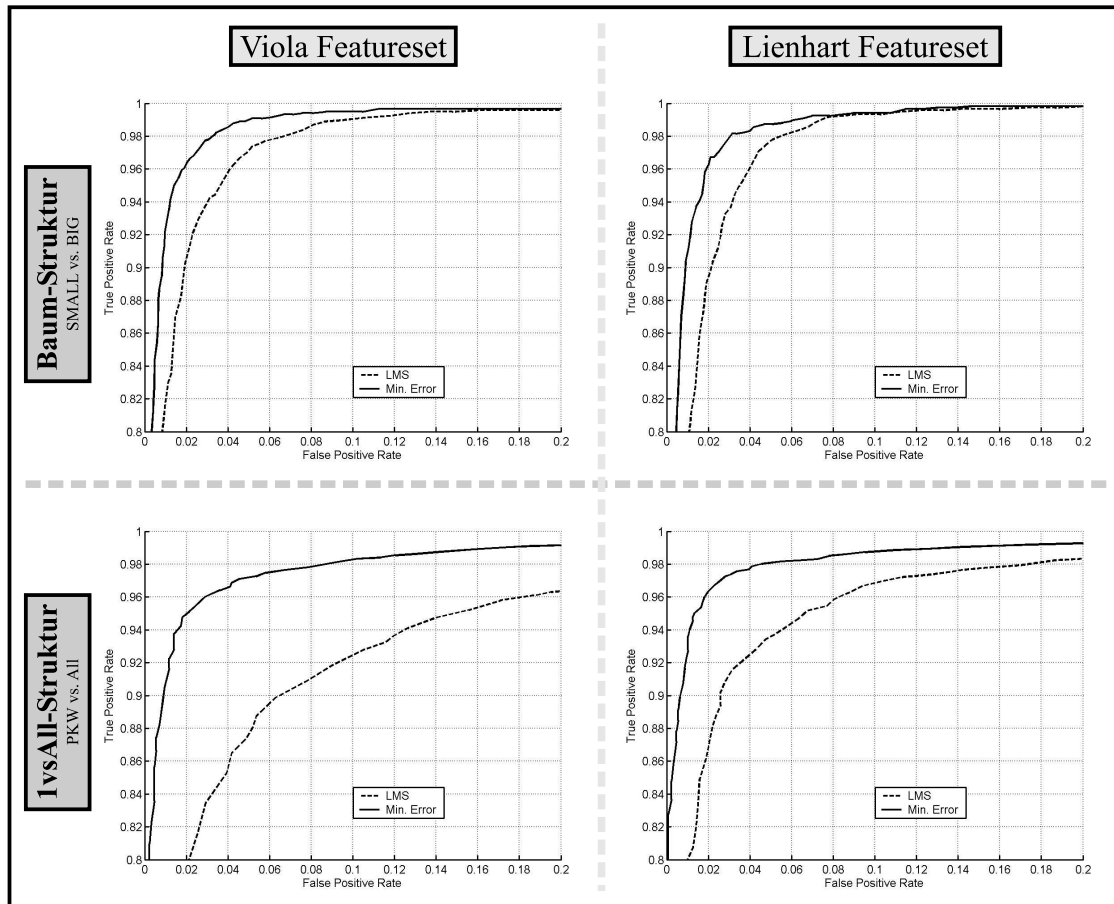


Abbildung 5.6: Vergleich der Weak Classifiers für zwei ausgewählte Einzelklassifikatoren. In der linken Spalte wird das *Viola*-Featureset, in der rechten Spalte das *Lienhart*-Featureset verwendet.

		VIOLA FEATURESET		LIENHART FEATURESET	
		<i>LMS</i>	<i>Min. Error</i>	<i>LMS</i>	<i>Min. Error</i>
<i>Baum</i> Struktur	“SMALL/BIG”	4,03 %	2,63 %	3,94 %	2,65 %
	“PKW/BIKE”	1,21 %	2,41 %	1,12 %	1,18 %
	“LKW/BUS”	5,92 %	5,84 %	5,98 %	2,93 %
<i>1-vs-All</i> Struktur	“PKW/All”	8,58 %	3,60 %	5,77 %	2,77 %
	“BIKE/All”	1,37 %	2,83 %	2,16 %	1,27 %
	“LKW/All”	5,14 %	6,23 %	4,93 %	4,29 %
	“BUS/All”	8,86 %	6,84 %	8,89 %	5,64 %

Tabelle 5.4: Vergleich der *Equal Error Rate* (EER) bei Verwendung der unterschiedlichen Weak Classifier Kriterien. Der kleinere Fehler ist jeweils **fett** herausgehoben.

		VIOLA FEATURESET			LIENHART FEATURESET		
		Level 10	Level 20	Level 30	Level 10	Level 20	Level 30
<i>Baum</i> Struktur	“SMALL/BIG”	4.23 %	3.21 %	2.63 %	3.56 %	3.18 %	2.65 %
	“PKW/BIKE”	2.26 %	2.47 %	2.41 %	2.35 %	1.27 %	1.18 %
	“LKW/BUS”	5.83 %	5.20 %	5.84 %	5.80 %	3.32 %	2.93 %
<i>1-vs-All</i> Struktur	“PKW/All”	6.37 %	3.99 %	3.60 %	4.54 %	3.54 %	2.77 %
	“BIKE/All”	5.81 %	2.63 %	2.83 %	3.71 %	1.18 %	1.27 %
	“LKW/All”	9.59 %	7.48 %	6.23 %	6.71 %	4.93 %	4.29 %
	“BUS/All”	11.80 %	8.71 %	6.84 %	8.85 %	5.82 %	5.64 %

Tabelle 5.5: Vergleich der *Equal Error Rate* (EER) auf verschiedenen Klassifikator-Levels. Der kleinste Fehler ist jeweils **fett** herausgehoben.

5.3.3 Performance in Abhängigkeit zur Anzahl verwendeter Features (Level)

In diesem Abschnitt wird die Abhängigkeit zwischen der Performance eines Klassifikators und der Anzahl an verwendeten Features (Klassifikator-Level) untersucht.

Durchführung: In diesem Experiment werden sämtliche Einzelklassifikatoren aus den Strukturen herausgenommen. Diese werden mit dem *Minimum Error* Kriterium und den unterschiedlichen Featuresets auf der Trainingsdatenmenge 3 trainiert. Die Performance der einzelnen Klassifikatoren auf dem Testdatensatz wird auf drei verschiedenen Klassifikator-Levels untersucht und gegenübergestellt.

Ergebnisse: In Abbildung 5.7 ist die Performance für drei ausgewählte Klassifikatoren bei unterschiedlicher Anzahl berechneter Features zu sehen. Als Parameter wird wieder der Schwellwert über der Wahrscheinlichkeit verwendet.

In Tabelle 5.5 wird die EER auf den verschiedenen Levels für die einzelnen Klassifikatoren einander gegenübergestellt. Der kleinste Fehler ist jeweils **fett** markiert.

5.3.4 Performance in Abhängigkeit zur Größe der Trainingsdatenmenge

In diesem Experiment wird untersucht, wie die Performance der Klassifikatoren von der Größe der Trainingsdatenmenge abhängt.

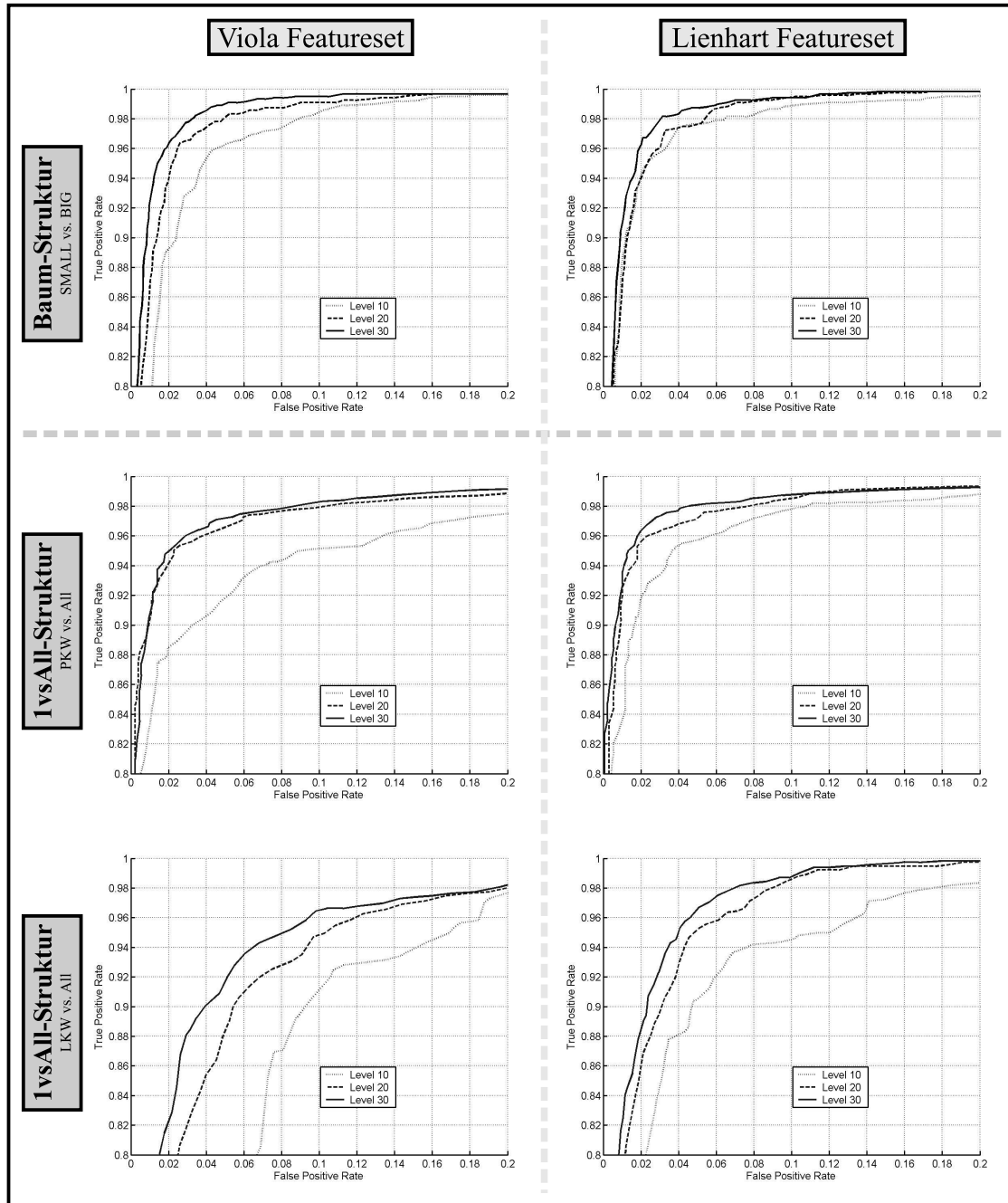


Abbildung 5.7: Performancevergleich bei unterschiedlichen Klassifikator-Levels für drei ausgewählte Klassifikatoren. In der linken Spalte wird das *Viola*-Featureset, in der rechten Spalte das *Lienhart*-Featureset verwendet.

Durchführung: Es werden sämtliche Einzelklassifikatoren der *Baum*-Struktur und der *1-vs-All*-Struktur herausgenommen und unter Einsatz des *Minimum Er-*

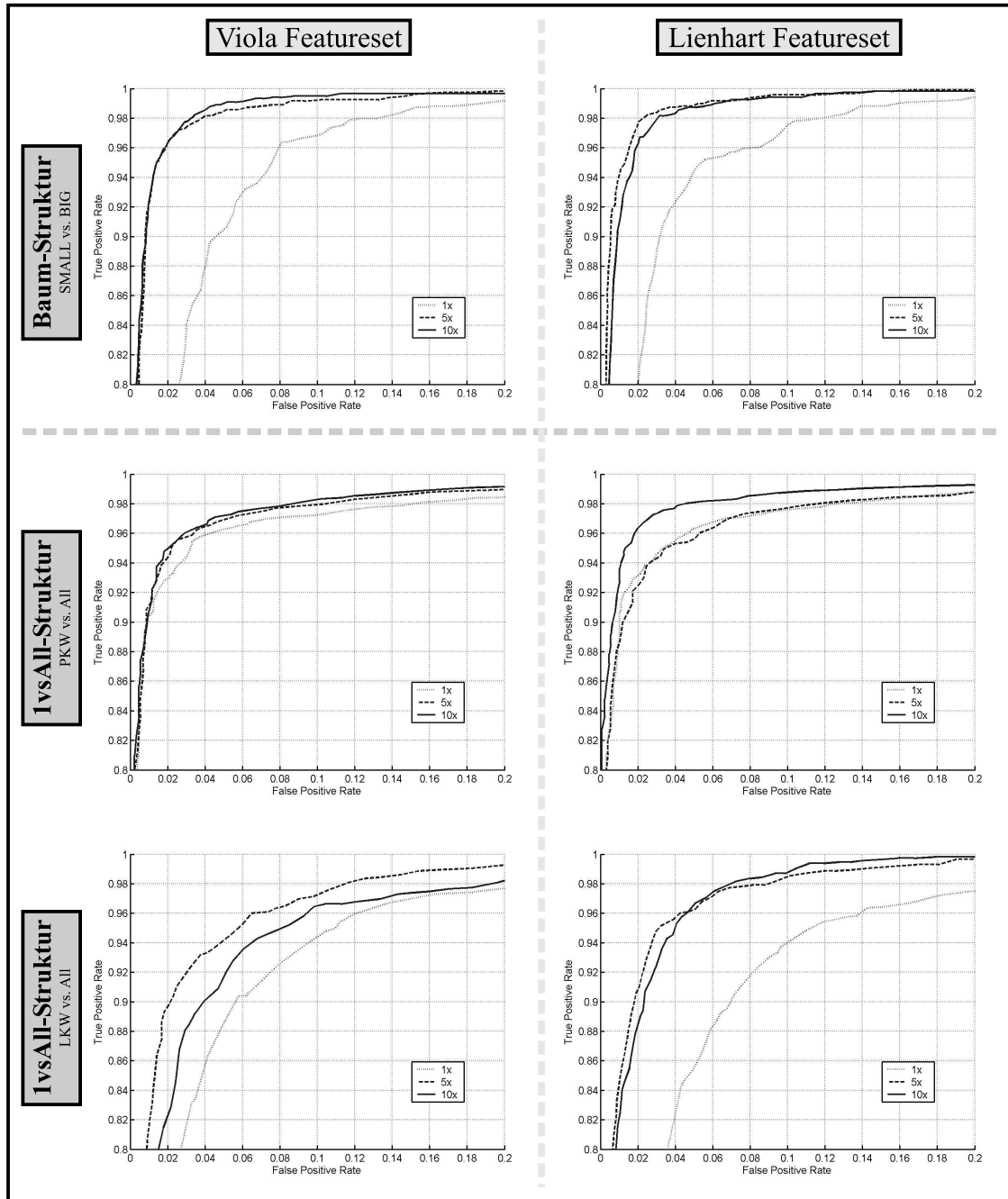


Abbildung 5.8: Performancevergleich bei drei unterschiedlich großen Trainingsdatensätzen. In der linken Spalte wird das *Viola*-Featureset, in der rechten Spalte das *Lienhart*-Featureset verwendet.

Die Klassifikatoren werden mit den Trainingsdatensätzen 1, 2 und 3 trainiert. Danach wird die Performance der Klassifikatoren auf der Testdatensatzmenge evaluiert.

		VIOLA FEATURESET			LIENHART FEATURESET		
		1x	5x	10x	1x	5x	10x
<i>Baum</i> Struktur	“SMALL/BIG”	6.57 %	2.71 %	2.63 %	5.21 %	2.13 %	2.65 %
	“PKW/BIKE”	3.36 %	1.61 %	2.41 %	2.70 %	2.23 %	1.18 %
	“LKW/BUS”	8.75 %	5.72 %	5.84 %	31.45 %	1.50 %	2.93 %
<i>1-vs-All</i> Struktur	“PKW/All”	4.01 %	3.65 %	3.60 %	4.29 %	4.62 %	2.77 %
	“BIKE/All”	4.03 %	3.68 %	2.83 %	7.35 %	4.04 %	1.27 %
	“LKW/All”	7.69 %	5.41 %	6.23 %	8.08 %	4.11 %	4.29 %
	“BUS/All”	30.04 %	8.46 %	6.84 %	13.00 %	1.28 %	5.64 %

Tabelle 5.6: Vergleich der *Equal Error Rate* (EER) der einzelnen Klassifikatoren, welche auf verschieden großen Trainingsdatenmengen trainiert wurden. Der kleinste Fehler ist jeweils **fett** herausgehoben.

Ergebnisse: In Abbildung 5.8 ist die Performance von 3 ausgewählten Level-30 Klassifikatoren in Abhängigkeit der Größe der Trainingsdatenmenge dargestellt. Als Parameter wird der Schwellwert über der Wahrscheinlichkeit p_{thr} verwendet.

In Tabelle 5.6 wird die resultierende EER für die mit den unterschiedlich großen Trainingsmengen trainierten Klassifikatoren aufgelistet. Der kleinste Fehler ist wieder **fett** markiert.

5.3.5 Gesamtperformance der einzelnen Strukturen

In diesem Versuch wird die Performance der gesamten *Baum*-Struktur und der gesamten *1-vs-All*-Struktur auf der Testdatenmenge untersucht.

Durchführung: Zunächst werden die beiden Klassifikatorstrukturen auf der Trainingsdatenmenge 3 nach dem *Minimum Error* Kriterium trainiert. Danach wird die Performance der Strukturen auf der Testdatenmenge evaluiert. Alle Einzelklassifikatoren beruhen dabei auf der Berechnung von 30 Features (Level 30).

Ergebnisse: In Abbildung 5.9 sind die ROCs für die Einzelklassifikatoren der einzelnen Strukturen zu sehen. In der linken Spalte wurde das *Viola*-Featureset, in der rechten Spalte das *Lienhart*-Featureset verwendet.

Die *Equal Error Rate* (EER) für die einzelnen Klassifikatoren ist bereits in den Spalten 5 und 8 (VIOLA-10x bzw. LIENHART-10x) von Tabelle 5.6 angegeben. Wird eine Einstellung der Klassifikatorparameter zur Erreichung der EER gewählt, so erhält man die Ergebnisse, die in Tabelle 5.7 aufgelistet sind. Hier findet man die

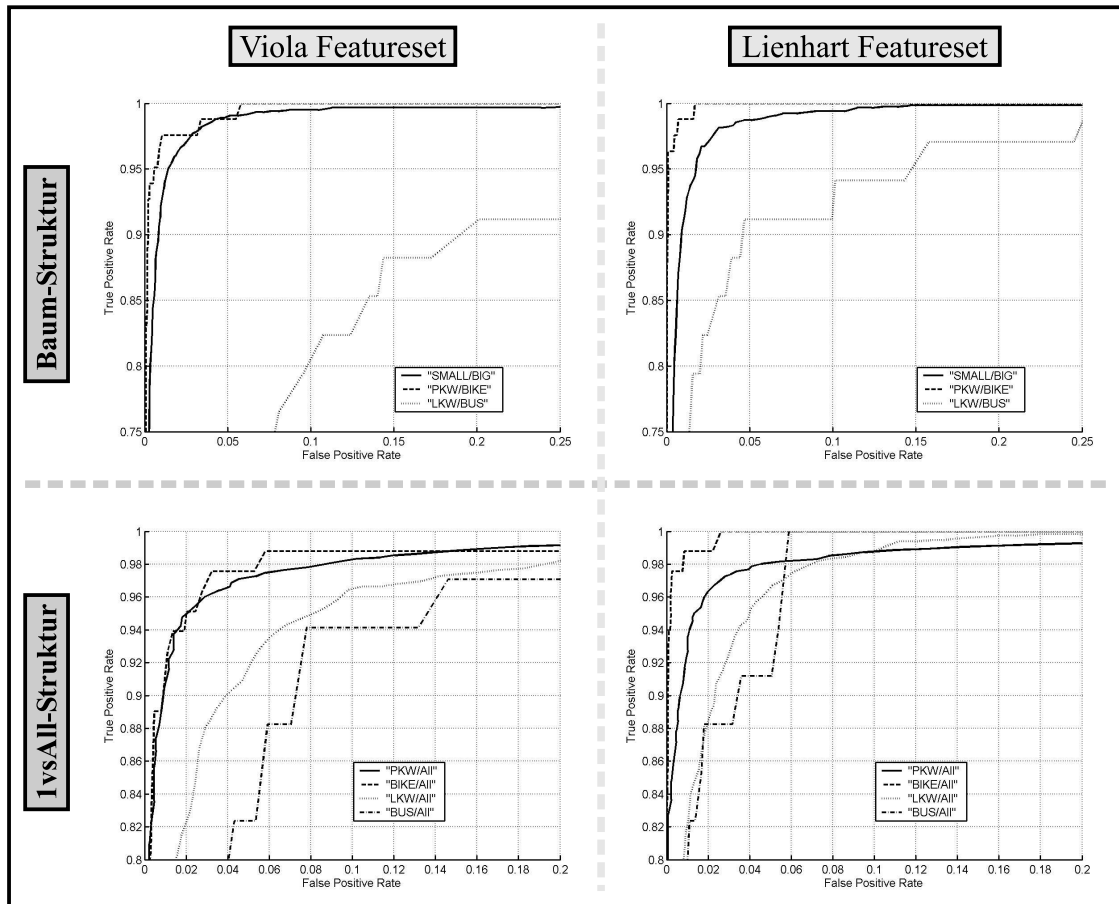


Abbildung 5.9: *Receiver Operator Characteristic* Kurven aller einzelnen Klassifikatoren auf der Testdatenmenge. In der linken Spalte wird das *Viola*-Featureset, in der rechten Spalte das *Lienhart*-Featureset verwendet.

Anzahl der korrekt klassifizierten Fahrzeuge und die prozentuelle Quote für die einzelnen Strukturen.

Als Beispiel zeigt Abbildung 5.10 die 15 wichtigsten Features, welche durch die *BIKE-vs-All* und *BUS-vs-All* Klassifikatoren der auf dem *Lienhart*-Featureset basierenden *1-vs-All* Struktur ausgewählt werden. In ersterem Fall befinden sich diese vor allem im Kopf- und Schulterbereich der Motorradfahrer sowie im Bereich des geworfenen Schattens, in zweiterem Fall vor allem im Bereich der Frontscheibe und der Seitenfenster der Busse.

In Abbildung 5.11 sind die Bilder von 30 Fahrzeugen zu sehen, die durch die auf dem *Lienhart*-Featureset basierenden *1-vs-All* Struktur falsch klassifiziert werden.

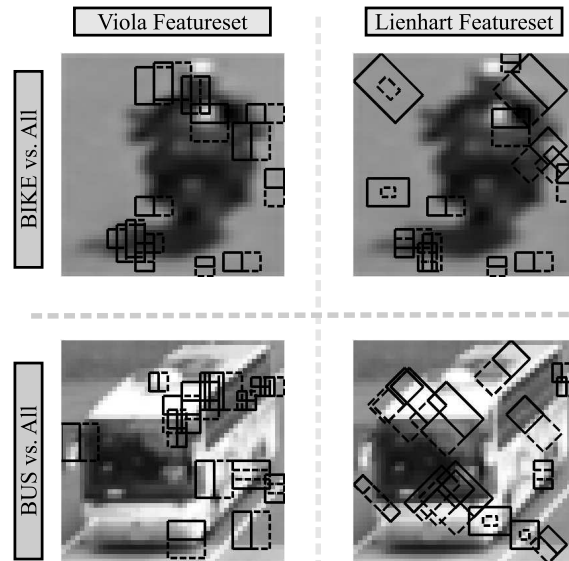


Abbildung 5.10: Die 15 wichtigsten Features für den *BIKE-vs-All* und *BUS-vs-All* Klassifikator der *1-vs-All*-Struktur. Links wird das *Viola*- und rechts das *Lienhart*-Featureset verwendet. Flächen positiven Vorzeichens sind mit durchgezogenen Linien, negativen Vorzeichens mit unterbrochenen Linien umrandet.

30 falsch klassifizierte Fahrzeuge



Abbildung 5.11: Bilder von 30 Fahrzeugen, welche durch die auf dem *Lienhart*-Featureset basierenden *1-vs-All* Struktur falsch klassifiziert werden.

		<i>Baum</i> Struktur		<i>1-vs-All</i> Struktur	
		Anzahl	%	Anzahl	%
VIOLA FEATURESET	PKW	5159	95,11	5211	96,07
	BIKE	71	86,59	72	87,80
	LKW	782	67,71	1049	90,82
	BUS	31	91,18	24	70,59
LIENHART FEATURESET	PKW	5234	96,50	5266	97,09
	BIKE	78	95,12	82	100,00
	LKW	934	80,87	1090	94,37
	BUS	33	97,06	28	82,35

Tabelle 5.7: Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge für die einzelnen Strukturen unter Verwendung der unterschiedlichen Featuresets.

	VIOLA FEATURESET		LIENHART FEATURESET	
	norm. Testset	Testset 1*	norm Testset	Testset 1*
“SMALL/BIG”	2,63 %	2,61 %	2,65 %	2,80 %
“PKW/All”	3,60 %	3,88 %	2,77 %	4,03 %
“LKW/All”	6,23 %	7,63 %	4,29 %	4,92 %

Tabelle 5.8: Vergleich der *Equal Error Rate* ausgewählter Klassifikatoren auf der normalen Testdatenmenge und der Menge an Daten von Kameraposition 1*.

5.3.6 Sensitivität auf eine Veränderung des Blickwinkels

Hier wird untersucht, wie sich die Performance der im vorigen Abschnitt trainierten Strukturen auf Daten verhält, welche unter einem leicht veränderten Blickwinkel aufgenommen wurden. Dabei soll der Einfluss einer kleinen Blickwinkeländerung auf die Performance einer bereits trainierten Struktur betrachtet werden.

Durchführung: Die im vorigen Abschnitt 5.3.5 beschriebenen Strukturen werden in diesem Experiment auf den Daten evaluiert, welche von Kameraposition 1* aus aufgenommen wurden. Als Vergleich werden die *Equal Error Rates* auf diesen Daten mit den EERs auf dem normalen Testdatensatz gegenübergestellt.

Ergebnisse: In Tabelle 5.8 wird die *Equal Error Rate* ausgewählter Level-30 Klassifikatoren auf den beiden Datensätzen gegenübergestellt. Die Daten der anderen Klassifikatoren haben verhältnismäßig geringe Aussagekraft, da sie auf sehr wenig Testdaten beruhen, und wurden aus diesem Grund hier nicht angegeben.

		<i>Baum</i> Struktur		<i>1-vs-All</i> Struktur	
		Anzahl	%	Anzahl	%
VIOLA FEATURESET	PKW	3223	95,69	3216	95,49
	BIKE	21	91,30	20	86,96
	LKW	366	82,43	386	86,94
	BUS	11	91,67	9	75,00
LIENHART FEATURESET	PKW	3287	97,60	3231	95,93
	BIKE	17	73,91	20	86,96
	LKW	361	81,31	399	89,86
	BUS	10	83,33	10	82,33

Tabelle 5.9: Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge für die einzelnen Strukturen unter Verwendung der unterschiedlichen Featuresets und einer Evaluierung auf Daten von Kameraposition 1*.

Werden für die Klassifikatoren Parameter zur Erreichung der EER gewählt, erhält man Resultate, die in Tabelle 5.9 angegeben sind. Hier ist die Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge für die einzelnen auf den unterschiedlichen Featuresets basierenden Strukturen aufgelistet.

5.4 Untersuchung der Generalisierungsfähigkeit

Um die universale Einsatzmöglichkeit der Algorithmen zu demonstrieren, werden diese in den nächsten beiden Abschnitten auf Daten angewandt, welche von den beiden anderen Kamerapositionen 2 und 3 aus aufgenommen wurden.

5.4.1 Performance der Strukturen für Kameraposition 2

In diesem Kapitel werden Daten von Kameraposition 2 behandelt (siehe Abbildung 5.12 für einige Beispiele von Fahrzeugbildern).

Durchführung: Die gesamte verfügbare Datenmenge wird in eine Trainings- und eine Testmenge aufgeteilt. Die genaue Anzahl der jeweiligen Fahrzeuge ist in Tabelle 5.10 aufgelistet.

Da für die BIKE und die BUS Kategorie insgesamt verhältnismäßig wenig Beispiele zur Verfügung stehen, wird eine weitere Unterscheidung zwischen PKW und BIKE bzw. LKW und BUS unterlassen. Stattdessen werden die BIKE-Fahrzeugbilder zur



Abbildung 5.12: Einige Beispielbilder für Daten, die von Kameraposition 2 aus aufgenommen wurden.

	Trainingsmenge	Testmenge
Kleinfahrzeuge	560	4503
Großfahrzeuge	295	296
Gesamt	855	4799

Tabelle 5.10: Aufteilung der Datenmenge in Trainings- und Testdaten für Kameraposition 2.

Kategorie der PKWs und die BUS-Fahrzeugbilder zur Kategorie der LKWs hinzugefügt. Damit vermindert sich die Anzahl der zu unterscheidenden Kategorien auf die der *Kleinfahrzeuge* (PKWs und BIKes), und die der *Großfahrzeuge* (LKWs und BUSse).

Durch die getroffene Einschränkung reduziert sich auch die Zahl der Einzelklassifikatoren in der *Baum*-Struktur auf einen, den “SMALL-vs-BIG” Klassifikator, und in der *1-vs-All*-Struktur auf einen, den “SMALL-vs-All” Klassifikator. Diese werden demnach auf der angegebenen Trainingsdatenmenge unter Verwendung des *Minimum Error* Kriteriums trainiert.

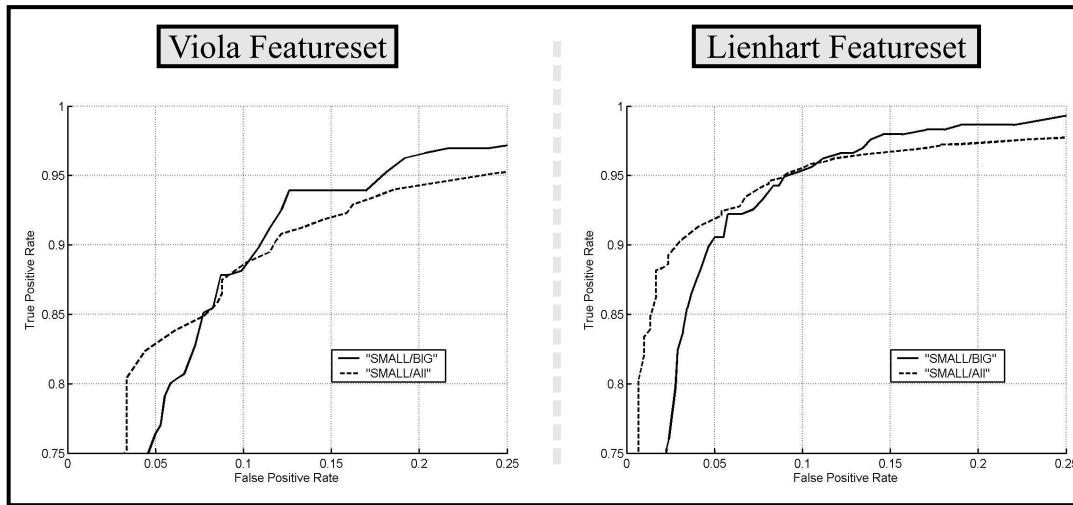


Abbildung 5.13: ROC Kurven für die beiden Klassifikatoren der *Baum* und der *1-vs-All* Struktur.

	VIOLA FEATURESET	LIENHART FEATURESET
"SMALL/BIG"	10,51 %	7,31 %
"SMALL/All"	11,00 %	6,64 %

Tabelle 5.11: Vergleich der *Equal Error Rate* der Klassifikatoren auf der Testdatensmenge.

		<i>Baum</i> Struktur		<i>1-vs-All</i> Struktur	
		Anzahl	%	Anzahl	%
VIOLA FEATURESET	Kleinfahrzeuge	4057	90,10	4162	92,43
	Großfahrzeuge	261	88,47	266	90,17
LIENHART FEATURESET	Kleinfahrzeuge	4208	93,45	4260	94,60
	Großfahrzeuge	273	92,54	278	94,24

Tabelle 5.12: Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge auf Daten von Kameraposition 2. Eine korrekte Klassifizierung bezieht sich hierbei auf eine richtige Zuordnung zur Gruppe der *Klein-* bzw. der *Großfahrzeuge*.

Ergebnisse: In Abbildung 5.13 sind die ROCs für die beiden Level-30 Klassifikatoren zu sehen.

In Tabelle 5.11 ist die *Equal Error Rate* für die beiden Klassifikatoren und die unterschiedlichen Featuresets aufgetragen. Eine Evaluierung auf den Testdaten, wobei die Klassifikator-Parameter zur Erreichung der EER benutzt werden, ergibt die Resultate, die in Tabelle 5.12 aufgelistet sind. Darin ist die Anzahl und die prozentuelle



Abbildung 5.14: Einige Beispielbilder für Daten, die von Kameraposition 3 aus aufgenommen wurden.

Quote der korrekt klassifizierten Fahrzeuge angegeben. Eine korrekte Klassifizierung bezieht sich hierbei auf eine richtige Zuordnung zur Gruppe der *Klein-* bzw. der *Großfahrzeuge*.

5.4.2 Performance der Strukturen für Kameraposition 3

In diesem Kapitel werden die Algorithmen auf Daten evaluiert, welche von Kameraposition 3 aus aufgenommen wurden (siehe Abbildung 5.14 für einige Beispiele von Fahrzeugbildern).

Durchführung: Es erfolgt eine Einteilung der verfügbaren Daten wie in Tabelle 5.13 angegeben. Wie bereits in dem in Abschnitt 5.4.1 beschriebenen Experiment werden auch hier die BIKes und BUSse zu den PKW- und LKW- Kategorien hinzugefügt. Dies führt zu denselben Einschränkungen bezüglich der Klassifikatoren, wie im vorigen Abschnitt beschrieben. Das Training der Klassifikatoren erfolgt ebenfalls unter Verwendung des *Minimum Error* Kriteriums.

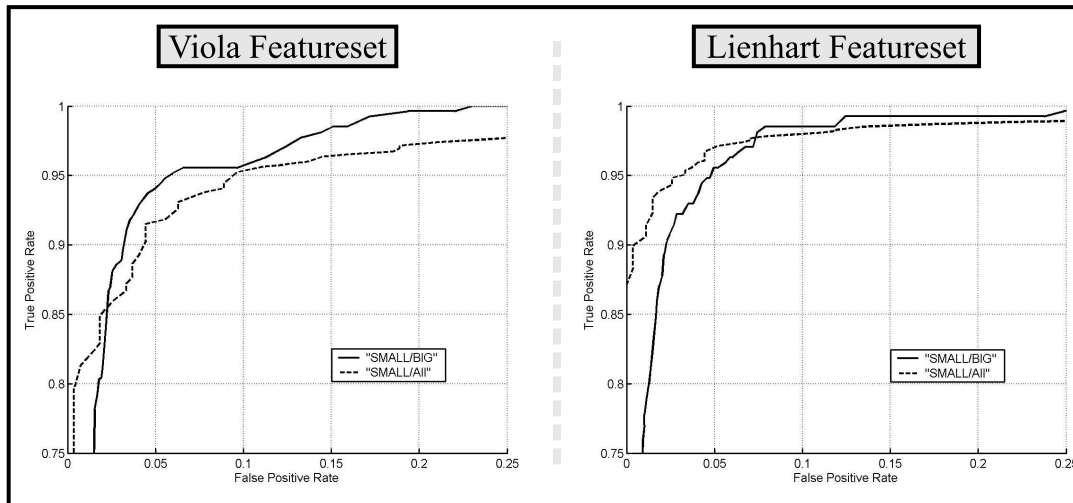


Abbildung 5.15: ROC Kurven für die beiden Klassifikatoren der *Baum* und der *1-vs-All* Struktur.

	Trainingsmenge	Testmenge
Kleinfahrzeuge	482	3375
Großfahrzeuge	270	270
Gesamt	752	3645

Tabelle 5.13: Aufteilung der Datenmenge in Trainings- und Testdaten für Kamera-position 3.

	VIOLA FEATURESET	LIENHART FEATURESET
“SMALL/BIG”	5,41 %	4,95 %
“SMALL/All”	6,60 %	4,05 %

Tabelle 5.14: Vergleich der *Equal Error Rate* der Klassifikatoren auf der Testdatenmenge.

Ergebnisse: In Abbildung 5.15 sind die ROC Kurven für die beiden Klassifikatoren zu sehen.

In Tabelle 5.14 ist die *Equal Error Rate* für die Klassifikatoren angegeben. In Tabelle 5.15 ist die Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge angegeben, wenn für die Klassifikatoren Parameter zur Erreichung der *Equal Error Rate* verwendet werden.

		<i>Baum</i> Struktur		<i>1-vs-All</i> Struktur	
		Anzahl	%	Anzahl	%
VIOLA FEATURESET	Kleinfahrzeuge	3207	95,02	3188	94,46
	Großfahrzeuge	254	94,07	258	95,56
LIENHART FEATURESET	Kleinfahrzeuge	3221	95,44	3229	95,67
	Großfahrzeuge	256	94,81	263	97,41

Tabelle 5.15: Anzahl und prozentuelle Quote korrekt klassifizierter Fahrzeuge auf Daten von Kameraposition 3. Eine korrekte Klassifizierung bezieht sich hierbei auf eine richtige Zuordnung zur Gruppe der *Klein-* bzw. der *Großfahrzeuge*.

5.5 Zusammenfassung und Schlussfolgerungen

In den Abschnitten 5.3 wird der Einfluss der verschiedensten Parameter auf die Performance der Klassifikatoren untersucht.

Die in Abschnitt 5.3.2 durchgeführte Untersuchung zeigt, dass als Weak Classifier das *Minimum Error* Kriterium dem *LMS* Kriterium fast immer überlegen, oder zumindest gleich gut ist. Offenbar sind die durch das *Minimum Error* Kriterium ausgewählten Features meist viel diskriminanter als die durch das *LMS* Kriterium ausgewählten Features. Deshalb ist die Verwendung des *Minimum Error* Kriteriums dem anderen Weak Classifier vorzuziehen.

Die Untersuchung der Abhängigkeit der Performance eines Klassifikators von der Anzahl an verwendeten Features (Levels) in Abschnitt 5.3.3 ergibt, dass die Performance eines Klassifikators mit steigendem Level, also mit steigender Anzahl berechneter Features deutlich besser wird. Obwohl ein Klassifikator hohen Levels demnach eine ausgezeichnete Performance zeigen sollte, ist dem der erforderliche Rechen- und Speicheraufwand entgegenzuhalten, sowohl in der Trainingsphase, als auch im praktischen Einsatz. Des Weiteren kann in den angeführten Grafiken ein gewisser Sättigungseffekt ausgemacht werden. Es scheint also für die Aufgabe der Fahrzeugklassifikation in diesem Rahmen ein Klassifikator-Level zwischen 20 und 40 durchaus ausreichend und sinnvoll zu sein.

Der Einfluss der Größe der Trainingsdatenmenge auf die Performance eines Klassifikators wird in Abschnitt 5.3.4 untersucht. Hier zeigt sich, dass bereits eine kleine Trainingsdatenmenge für einen Klassifikator ausreicht, um eine gewisse Performance zu erreichen. Weiters zeigt sich, dass eine Performancesteigerung nur bis zu einem gewissen Grad durch eine Vergrößerung der Trainingsdatenmenge erreicht werden kann und dass auch hier ein gewisser Sättigungseffekt eintritt. Deshalb gilt es eine Abwägung zwischen der Anzahl an Trainingsdaten zum Erreichen einer gewissen

Performance und des damit verbundenen Aufwands durchzuführen. Da mit zunehmender Größe der Trainingsdatenmenge die Performance eines Klassifikators aber allgemein besser wird, empfiehlt es sich eine möglichst große Menge an Daten für das Training zu verwenden.

Eine Evaluierung der gesamten Strukturen auf dem Testdatensatz in Abschnitt 5.3.5 zeigt, dass die *1-vs-All*-Struktur im Allgemeinen bessere Ergebnisse erzielt als die *Baum*-Struktur. Dies wird besonders dann offensichtlich, wenn man die Summen der durch die unterschiedlichen Strukturen korrekt klassifizierten Großfahrzeuge, also LKWs und BUSse, miteinander vergleicht. Der Grund für den höheren Fehler der *Baum*-Struktur ist in erster Linie, dass sich eine Fehlentscheidung in der ersten Stufe nicht mehr ändern lässt und sich die Fehler der beiden Klassifikatorstufen dadurch akkumulieren.

Weiters kann festgestellt werden, dass durch die Klassifikatoren Features ausgewählt werden, welche für eine spezifische Fahrzeugkategorie besonders auffällig zu sein scheinen. Außerdem zeigt sich, dass vor allem Fahrzeuge durch die Algorithmen falsch klassifiziert werden, die in irgendeiner Form von der Norm stark abweichen. Im Prinzip sind dies zumeist Fahrzeuge, die auch in Wirklichkeit in keine der angegebenen Kategorien einzuordnen sind, wie beispielsweise mittelgroße Kastenwagen oder LKWs mit speziellen Aufbauten oder außergewöhnlichem Ladegut. Eine wichtige Folgerung daraus ist aber auch, dass das allgemeine Aussehen eines Fahrzeugs für eine korrekte Klassifikation viel wichtiger ist als der Einfluss von Störparametern, wie etwa Teile von Schatten oder Teile von Fahrbahnmarkierungen.

Die Untersuchung der Sensitivität trainierter Strukturen auf eine leichte Änderung des Blickwinkels in Abschnitt 5.3.6 zeigt, dass kein wesentlicher Performanceeinbruch bei einer so kleinen Änderung besteht. Der Grund dafür ist die Tatsache, dass die Trainings- und Testbilder Fahrzeuge auf allen Fahrspuren zeigen. Da sich die Spuren selbst nicht alle im selben Winkel zur Kamera befinden, zeigen die Bilder auch die Fahrzeuge selbst aus leicht unterschiedlichen Winkeln. Deshalb führt eine weitere kleine Änderung des Blickwinkels nur zu einer sehr geringen Veränderung der Gesamtpformance.

Die in Abschnitt 5.4 durchgeführten Versuche mit von anderen Kamerapositionen aufgenommenen Daten demonstrieren die universale Einsatzmöglichkeit der beschriebenen Algorithmen. Unter Verwendung einer Trainingsdatenmenge mittlerer Größe und Level-30 Klassifikatoren können durchaus respektable Ergebnisse erzielt werden.

Auffallend ist dabei, dass die in Abschnitt 5.4.1 präsentierten Ergebnisse deutlich schlechter sind als alle in den anderen Experimenten erzielten Resultate. Der Grund dafür ist, dass aufgrund der Wahl dieser Aufnahmeposition und des Sonnenverlaufs in sehr vielen Fahrzeugbildern Sonnenspiegelungen, so genannte *Highlights* vorhanden sind. Dies ist auch aus den angeführten Beispielbildern erkennbar. Diese störenden Einflüsse beeinflussen offensichtlich die durch die Klassifikatoren durchgeführte Featureberechnung und haben deshalb auch einen deutlich negativen Einfluss auf die Gesamtperformance.

Das in Abschnitt 5.4.2 durchgeführte Experiment zeigt, dass Fahrzeuge auch von Kamerapositionen aus aufgenommen und klassifiziert werden können, von welchen aus einige wichtige Merkmale von Fahrzeugen nicht sichtbar sind. So ist eine korrekte Klassifikation von LKWs auch dann möglich, wenn beispielsweise das für die LKW-Kategorie sehr markante Führerhaus nicht direkt abgebildet ist. Die Featureauswahl erfolgt in diesem Fall eben nach signifikanten Merkmalen im Bereich der hinteren Achsen und der Rückleuchten.

In sämtlichen Experimenten werden bewusst das *Viola*- und das *Lienhart*-Featureset gegenübergestellt, um einen Gesamteindruck über die durch die Verwendung der unterschiedlich gearteten Features resultierende Performance der Klassifikatoren zu erhalten.

Es kann festgehalten werden, dass die auf dem *Viola*-Featureset basierenden Klassifikatoren eine etwas schlechtere Performance zeigen als die auf dem *Lienhart*-Featureset basierenden Klassifikatoren. Allerdings scheint das LIENHART-Featureset empfindlicher auf kleine Blickwinkeländerungen zu reagieren als das VIOLA-Featureset, was aus den in Abschnitt 5.3.6 präsentierten Ergebnissen ersichtlich ist. Für den Einsatz des *Viola*-Featuresets sprechen weiters der geringere Speicherbedarf und der niedrigere Berechnungsaufwand.

Zusammenfassend ist das *Lienhart*-Featureset dem *Viola*-Featureset trotzdem vorzuziehen. Die einfache Begründung liegt darin, dass die Fähigkeit des *Lienhart*-Featuresets, charakteristische Eigenschaften eines Objekts besser beschreiben zu können, für eine weitere Unterteilung der groben Fahrzeugkategorien entscheidend sein kann. Außerdem ist der Performancegewinn höher zu bewerten als die Einsparungen auf dem Sektor des Aufwands und da es sich bei der entwickelten Gesamtapplikation nicht um ein für den mobilen Einsatz gedachtes System handelt, bilden die durch eine kleine Änderung des Blickwinkels hervorgerufenen Performanceeinbußen keinen entscheidenden Faktor.

		KLASSIFIKATION					
		PKW	BIKE	LKW	BUS	GESAMT	KORREKT
WIRKLICHE KLASSE	PKW	18649	398	377	105	19529	95,49 %
	BIKE	0	82	0	0	82	100,00 %
	LKW	18	3	1090	44	1155	94,37 %
	BUS	0	0	6	28	34	82,35 %
GESAMT		19849				20800	95,43 %

Tabelle 5.16: *Confusion Matrix* des empfohlenen Systems. Die Elemente der Hauptdiagonale geben die Anzahl der korrekt klassifizierten Fahrzeuge an und sind hier **fett** herausgehoben.

Die Wahl des verwendeten Weak Classifiers fällt eindeutig zugunsten des *Minimum Error* Kriteriums aus und als Struktur ist sicherlich die *1-vs-All* Struktur der *Baum*-Struktur aus den besprochenen Gründen vorzuziehen.

Die vorgeschlagene Konfiguration einer Klassifikationsstruktur wird nun benutzt um zu einer Aussage über die Gesamtperformance eines solchen Systems zu gelangen.

Eine *1-vs-All* Struktur wird unter Verwendung des *Minimum Error*-Kriteriums und des LIENHART-Featuresets mit der in Abschnitt 5.3.1 definierten Trainingsmenge 3 trainiert, wobei sämtliche Einzelklassifikatoren auf der Berechnung von 30 Features basieren (Level 30). Benutzt man dann die übrigen Daten von Kameraposition 1 als Testdaten (wobei nun sämtliche zur Verfügung stehenden PKW-Fahrzeugbilder verwendet werden) und verwendet Parameter zur Erreichung der *Equal Error Rate*, so ergibt sich die in Tabelle 5.16 dargestellte *Confusion Matrix*.

Aus dieser *Confusion Matrix* ist ersichtlich, dass von der gesamten Menge an Testdaten demnach **95,43%** der abgebildeten Fahrzeuge die richtige Kategorie zugewiesen wird. Fasst man die PKW und BIKE bzw. die LKW und BUS Kategorien zu *Kleinfahrzeug*- und *Großfahrzeug*-Klassen zusammen und konzentriert sich damit nur auf die Bestimmung des Anteils des Schwerverkehrs am Gesamtverkehrsaufkommen, so können sogar **97,58%** aller Fahrzeuge richtig klassifiziert werden.

KAPITEL 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

In dieser Arbeit werden verschiedene Ansätze aus der Bildverarbeitung zur Detektion von Objekten aufgegriffen und für den Bereich der Fahrzeugklassifikation adaptiert. Neben der Beschreibung der Umgebungsbedingungen, der Zielsetzungen dieser Diplomarbeit und einem Überblick über die in der Literatur vorgeschlagenen Algorithmen, werden in den einzelnen Kapiteln mehrere zusammenhängende Kernthemen behandelt.

Das erste davon ist das übergeordnete System, welches im Rahmen des INFOA10-Projekts entwickelt wurde und in dieser Arbeit zur Datenaufnahme verwendet wurde. Ausgehend von gewissen Bedingungen, die bei der Entwicklung des Systems vorausgesetzt wurden, wird der gesamte modulare Aufbau detailliert beschrieben und die Funktionsweise der in den einzelnen Modulen verwendeten Algorithmen im Detail erklärt. Anschließend werden die Probleme beschrieben, welche beim praktischen Einsatz des Systems auftreten, und die Gründe dafür thematisiert.

Ein weiteres Thema bildet die Extraktion charakteristischer Features aus Bildern. Beginnend mit einer Beschreibung der Transformation von Bildern in unterschiedliche Repräsentationsarten, wird eine Menge verschiedener Features vorgestellt, welche auf Haar-Wavelets basieren, und deren genaue Funktion und Berechnung erklärt. Danach werden zwei unterschiedliche Featuresets, das VIOLA- und das LIENHART-Featureset, definiert und eine Gegenüberstellung der verwendeten Bildgröße und der Anzahl berechenbarer Features durchgeführt.

Die Aufbereitung des ADA-Boost Algorithmus zur Klassifikation von Fahrzeugen bildet einen weiteren Kernpunkt dieser Diplomarbeit. Einem generellen Überblick

über die Funktion des Algorithmus folgt die Definition zweier unterschiedlicher Weak Classifier Kriterien, des *Least Mean Square* und des *Minimum Error* Kriteriums, und eine Beschreibung ihrer genauen Funktionsweise.

Auf der Basis einzelner Klassifikatoren werden zwei unterschiedliche Strukturen, die *Baum* und die *1-vs-All* Struktur vorgestellt, die eine Multiclass-Klassifikation ermöglichen. Des Weiteren werden die für die einzelnen Strukturen benötigten technischen Ressourcen verglichen und die Vor- und Nachteile der einzelnen Strukturen angesprochen.

Die Performance der entwickelten Verfahren wird schließlich einer ausführlichen experimentellen Untersuchung unterzogen. Zu Beginn werden die verwendeten Trainingsdaten und Testdaten beschrieben und Beispiele für die unterschiedlichen Fahrzeugkategorien angeführt. Danach werden sowohl die beiden Weak Classifier Kriterien miteinander verglichen, als auch Untersuchungen der Abhängigkeit der Performance von der verwendeten Menge an Trainingsdaten und der Anzahl der durch einen Klassifikator verwendeten Features (Level) durchgeführt. Dabei werden sowohl die Einflüsse der Parameter auf die Performance der Einzelklassifikatoren untersucht, als auch die Gesamtperformance der *Baum* und der *1-vs-All* Struktur auf den Testdaten ermittelt. Außerdem wird die Sensitivität einer trainierten Struktur auf eine kleine Änderung des Blickwinkels untersucht und die universelle Einsatzmöglichkeit der vorgestellten Algorithmen durch die Anwendung auf von unterschiedlichen Kamerapositionen aus aufgenommene Fahrzeugbilder demonstriert.

Zum Schluss werden die Resultate der einzelnen Experimente zusammengefasst, diskutiert, und die beste für den praktischen Einsatz geeignete Struktur ermittelt und vorgeschlagen. Diese wird dann auf einer festgelegten Testdatenmenge evaluiert und die Ergebnisse zusammengefasst und angegeben.

6.2 Ausblick

Eine Reihe von Vorschlägen zur Erweiterung bzw. Verbesserung der Algorithmen stehen zur Diskussion.

Im Bereich des Systems, welches zur Datenaufnahme verwendet wurde, sollte das Zusammenspiel der einzelnen Module optimiert werden. Vor allem in der Verbesserung des Zusammenwirkens zwischen dem *Segmentierungs*- und dem *Tracker*-Modul steckt viel Potential zur Steigerung der Gesamtperformance. Dadurch könnte ein Großteil der vor allem bei der Verfolgung sich gegenseitig verdeckender Fahrzeuge auftretenden Probleme beseitigt werden. Außerdem könnte ein Algorithmus im

Bereich des *Bildaufnahme*-Moduls eingebunden werden, der eine Stabilisierung des Videobildes vornehmen und das gesamte System dadurch robuster gegen Vibrationen machen könnte.

Vorausgesetzt, man hätte die technischen Möglichkeiten, die gesamte Menge der auf der verwendeten Bildgröße berechenbaren Features für das Training eines Klassifikators zu verwenden, so würden sicher noch viel diskriminanzere Features durch den Algorithmus ausgewählt werden, wodurch ein Performancegewinn erreicht werden würde. Außerdem könnten die Ergebnisse einer genaueren Untersuchung der statistischen Verteilungen der Featurewerte in den einzelnen Fahrzeugkategorien und der in einem solchen Verfahren ausgewählten Featuregrößen und Featuretypen benutzt werden, um einen verhältnismäßig kleinen Featurepool zu bilden, in welchem nur die allgemein diskriminantesten Features vorhanden sind. Durch Umsetzung dieser beiden Ideen könnte theoretisch ein Performancegewinn erreicht und gleichzeitig die Trainingsphase deutlich verkürzt werden.

Wie sich durch die Versuche gezeigt hat, können Fahrzeuge der vier verwendeten Kategorien problemlos mit geringem Fehler klassifiziert werden. Davon angeregt könnte eine weitere Unterteilung in Subklassen wie zum Beispiel in eine "Klein-LKW"- oder "Wohnwagen"-Klasse vorgenommen werden. Als Anfang dazu wäre schon die Verwendung einer einfachen Rückweisungsklasse für alle nicht eindeutig zuordenbare Fahrzeuge denkbar.

Da es sich bei dem zum Training der Klassifikatoren verwendeten Verfahren um einen Supervised-Learning-Algorithmus handelt und mit einer kleinen Menge an Trainingsbeispielen bereits gute Resultate erzielt werden können, könnte für den Einsatz in der Trainingsphase ein Bootstrapping-Algorithmus entwickelt werden. Dadurch könnte unter Umständen die Trainingsphase, abgesehen von einem beschränkten Eingreifen durch den Supervisor, zu großen Teilen verselbstständigt werden.

Interessant wäre weiters eine Untersuchung, ob es möglich ist, einen trainierten Klassifikator auf unter einem anderen Blickwinkel aufgenommene Daten anzuwenden. Unter Umständen könnte durch einen Algorithmus zum Re-Trainieren ein Weg gefunden werden, einen bereits Basisklassifikator mit mittelmäßiger Performance universal einsetzbar zu machen und durch diesen Algorithmus zu optimieren.

ANHANG A

Anhang

A.1 Der Kalman Tracker

Für eine ausführliche Dokumentation und Diskussion des Kalman Filters sei auf die Arbeiten von GELB [Gel74] und JACOBS [Jac93] verwiesen. Eine kurze Erläuterung seiner Funktionsweise wird hier nach einem Tutorial von BISHOP und WELCH [WB95] geführt.

Ein diskreter Prozess ist charakterisiert durch eine Differenzengleichung der Form

$$\mathbf{x}_k = \mathbf{A} \cdot \mathbf{x}_{k-1} + \mathbf{B} \cdot \mathbf{u}_k + \mathbf{w}_{k-1} \quad (\text{A.1})$$

und einer Messung $\mathbf{z} \in \mathbb{R}^m$

$$\mathbf{z}_k = \mathbf{H} \cdot \mathbf{x}_k + \mathbf{v}_k. \quad (\text{A.2})$$

Die Zufallsvariablen $\mathbf{w}_k$ und $\mathbf{v}_k$ beschreiben das Prozess- und das Messrauschen und können als voneinander unabhängig und normalverteilt angenommen werden

$$p(\mathbf{w}) \sim N(0, \mathbf{Q}) \quad (\text{A.3})$$

$$p(\mathbf{v}) \sim N(0, \mathbf{R}), \quad (\text{A.4})$$

wobei $\mathbf{Q}$ und $\mathbf{R}$ die Kovarianzmatrizen des Prozessrauschens und des Messrauschens sind. Die $(n \times n)$ Matrix $\mathbf{A}$ in Gleichung (A.1) stellt den mathematischen Zusammenhang zwischen dem Zustand im aktuellen Schritt k und dem im vorhergegangenen

Schritt $k - 1$ her. Die $(n \times l)$ Matrix $\mathbf{B}$ stellt die Verbindung zwischen optionalen Inputparametern $\mathbf{u}_k \in \mathbb{R}^l$ und dem Zustand $\mathbf{x}_k$ her und die $(m \times n)$ Matrix $\mathbf{H}$ in Gleichung (A.2) gibt den Zusammenhang zwischen der Messung $\mathbf{z}_k$ und dem Zustand $\mathbf{x}_k$ wieder.

Bezeichnet man nun eine “a priori” Vorhersage für den Zustand im Schritt k mit $\hat{\mathbf{x}}_k^- \in \mathbb{R}^n$ und die “a posteriori” Schätzung mit $\hat{\mathbf{x}}_k \in \mathbb{R}^n$, so kann man den “a priori” Fehler $\mathbf{e}_k^-$ und den “a posteriori” Fehler $\mathbf{e}_k$ mit

$$\mathbf{e}_k^- \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k^- \quad (\text{A.5})$$

$$\mathbf{e}_k \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k \quad (\text{A.6})$$

anschreiben, wobei $\mathbf{x}_k \in \mathbb{R}^n$ den unbekannten wahren Zustand des Prozesses angibt. Die “a priori” und die “a posteriori” Fehlerkovarianz lässt sich sonach wie folgt anschreiben:

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \cdot \mathbf{e}_k^{-T}] \quad (\text{A.7})$$

$$\mathbf{P}_k = E[\mathbf{e}_k \cdot \mathbf{e}_k^T] \quad (\text{A.8})$$

Sucht man nun nach einer Gleichung, die die “a posteriori” Schätzung des Zustands $\hat{\mathbf{x}}_k$ als Linearkombination der “a priori” Vorhersage $\hat{\mathbf{x}}_k^-$ und einer gewichteten Differenz zwischen der Messung $\mathbf{z}_k$ und einer Prediction der Messung $\mathbf{H} \cdot \hat{\mathbf{x}}_k^-$ bildet, so findet man:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_k^-) \quad (\text{A.9})$$

Die Differenz $(\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_k^-)$ nennt man die so genannte *Innovation*, welche die Diskrepanz zwischen der Vorhersage der Messung und der wirklichen Messung beschreibt. Die $(n \times m)$ Matrix $\mathbf{K}$ nennt sich *Kalman Gain* und minimiert die “a posteriori” Fehlerkovarianz aus Gleichung (A.8). Eine mögliche Form der Berechnung von $\mathbf{K}$, durch welche Gleichung (A.8) minimiert wird, ist

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^- \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_k^- \cdot \mathbf{H}^T + \mathbf{R})^{-1} \\ &= \frac{\mathbf{P}_k^- \cdot \mathbf{H}^T}{\mathbf{H} \cdot \mathbf{P}_k^- \cdot \mathbf{H}^T + \mathbf{R}} \end{aligned} \quad (\text{A.10})$$

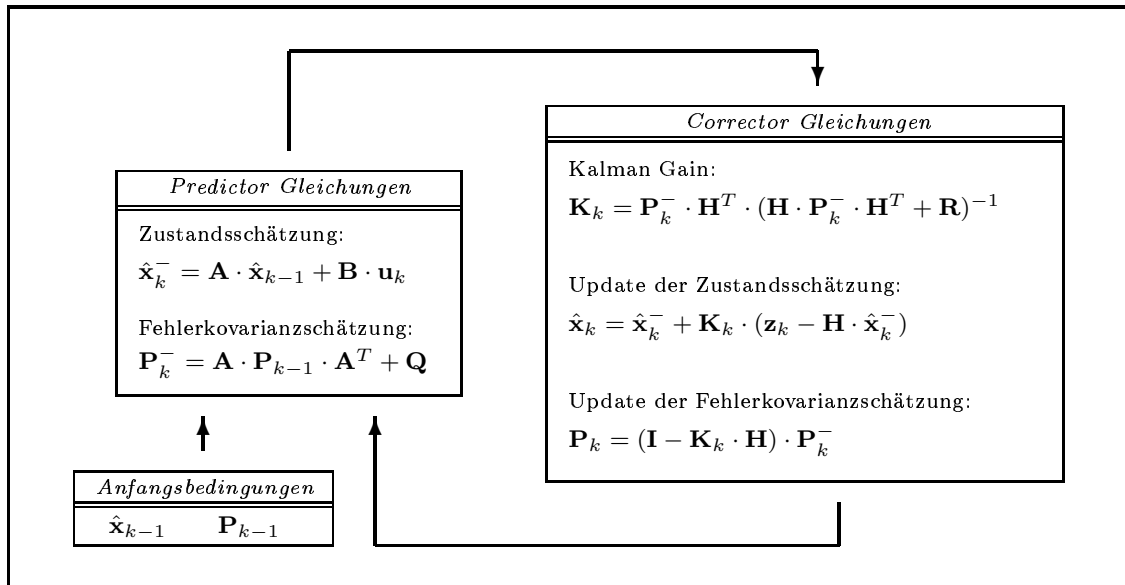


Abbildung A.1: Funktion des Kalman Filters; linker Hand die Gleichungen für die “a priori” Vorhersage, rechter Hand die Gleichungen für die “a posteriori” Korrektur.

Zusammengefasst besteht der Algorithmus aus 2 Gruppen von Gleichungen:

- *Time Update* (oder *Predictor*) Gleichungen, welche vom aktuellen Zustand $\mathbf{x}$ und der aktuelle Fehlerkovarianz $\mathbf{P}$ eine Vorhersage für den Zustand im nächsten Schritt treffen (“a priori” Schätzung), und
- *Measurement Update* (oder *Corrector*) Gleichungen, die aus diesen Schätzungen unter Einbeziehung eines neuen Messergebnisses eine verbesserte Schätzung errechnen (“a posteriori” Schätzung).

Diese beiden Gruppen von Gleichungen bilden eine Regelschleife, welche in Abbildung A.1 dargestellt ist.

In dieser Arbeit wird ein Kalman Filter zweiter Ordnung als *Kalman Tracker* verwendet, um ein Objekt durch ein Bild zu verfolgen. Dabei modelliert der Tracker den Mittelpunkt eines Bildausschnitts, der ein vermeintliches Fahrzeug zeigt, und die Breite und Höhe dieses Ausschnitts. Die Systemgleichung berücksichtigt die Geschwindigkeit, mit der sich ein Fahrzeug in der Bildebene bewegt (also die Bewegung des Bildausschnitts) und einen Beschleunigungsfaktor.

Die Verwendung eines Systems zweiter Ordnung ist aus zwei Überlegungen heraus sinnvoll. Einerseits handelt es sich bei der Bewegung eines Fahrzeugs prinzipiell um

ein System, welches sehr gut durch Geschwindigkeit und Beschleunigung beschrieben werden kann. Auf der anderen Seite kann durch die freie Wahl des Blickwinkels eine Verzerrung einer gleichförmigen Bewegung eines Fahrzeugs in der Bildebene erfolgen, die ebenfalls durch ein System zweiter Ordnung gut beschrieben werden kann. Aus diesen beiden Gründen eignet sich eine solche Konfiguration eines *Kalman Trackers* für die verwendete Anwendung sehr gut.

A.2 Berechnung einer Homographie

Die hier angeführte Definition und gekürzte Anleitung zur Berechnung einer Homographie ist dem Buch von HARTLEY und ZISSERMAN [HZ00] entnommen.

Als *2D-Homographie* bezeichnet man die projektive Transformation aller Punkte $\mathbf{x}_i$ einer Ebene in korrespondierende Punkte $\mathbf{x}'_i$ in einer anderen Ebene.

Zur Berechnung dieser Transformation kann beispielsweise der DLT-Algorithmus (*Direct Linear Transform*) verwendet werden. Dieser funktioniert prinzipiell so:

Gegeben seien $n \geq 4$ korrespondierende 2D Punkte $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$. Man berechne die 2D-Homographie-Matrix $\mathbf{H}$ so dass für alle korrespondierenden Punktpaare die Formel

$$\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i \quad (\text{A.11})$$

gilt. Bei $\mathbf{x}_i$ und $\mathbf{x}'_i$ handelt es sich um homogene Vektoren, diese haben also 3 Elemente. Die Vektoren $\mathbf{x}'_i$ und $\mathbf{H}\mathbf{x}_i$ haben zwar dieselbe Richtung, können sich aber in ihrer Länge um einen Skalierungsfaktor ungleich 0 unterscheiden. Die Formel (A.11) kann auch unter Verwendung des Kreuzprodukts angeschrieben werden:

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0} \quad (\text{A.12})$$

Schreibt man nun die Matrix $\mathbf{H}$ mit

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad (\text{A.13})$$

an, bezeichnet man die erste, zweite und dritte Zeile der Matrix $\mathbf{H}$ mit $\mathbf{h}^{1T}$, $\mathbf{h}^{2T}$ und $\mathbf{h}^{3T}$ und schreibt man den Vektor $\mathbf{x}'_i$ mit

$$\mathbf{x}'_i = \begin{pmatrix} x'_i \\ y'_i \\ w'_i \end{pmatrix} \quad (\text{A.14})$$

an, so ergibt sich für Gleichung (A.12)

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i \mathbf{h}^{3T} \mathbf{x}_i - w'_i \mathbf{h}^{2T} \mathbf{x}_i \\ w'_i \mathbf{h}^{1T} \mathbf{x}_i - x'_i \mathbf{h}^{3T} \mathbf{x}_i \\ x'_i \mathbf{h}^{2T} \mathbf{x}_i - y'_i \mathbf{h}^{1T} \mathbf{x}_i \end{pmatrix} = \mathbf{0} . \quad (\text{A.15})$$

Da $\mathbf{h}^{jT} \mathbf{x}_i = \mathbf{x}_i^T \mathbf{h}^j$ für $j = 1, 2, 3$, kann die Formel auch angeschrieben werden mit $\mathbf{A}_i \mathbf{h} = \mathbf{0}$,

$$\underbrace{\begin{bmatrix} \mathbf{0}^T & -w'_i \mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ w'_i \mathbf{x}_i^T & \mathbf{0}^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix}}_{\mathbf{A}_i} \underbrace{\begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix}}_{\mathbf{h}} = \mathbf{0} . \quad (\text{A.16})$$

Die Matrix $\mathbf{A}_i$ ist dabei eine 3×9 Matrix, die aber nur 2 linear unabhängige Zeilen besitzt. Das bedeutet, man erhält für jede Punktkorrespondenz 2 linear unabhängige Gleichungen. Aus diesem Grund kann man, sofern es sich bei der Wahl der verwendeten Punkte nicht ungünstigerweise um kollineare Punkte handelt, die dritte Zeile der Matrix $\mathbf{A}_i$ eliminieren und man erhält eine 2×9 Matrix $\mathbf{A}_i$.

Man kann dann alle aus n Punktkorrespondenzen gewonnenen 2×9 Matrizen $\mathbf{A}_i$ in eine einzige große $2n \times 9$ Matrix $\mathbf{A}$ zusammenfassen.

Berechnet man nun eine SVD (*Singular Value Decomposition*) auf der Matrix $\mathbf{A}$, so dass

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T , \quad (\text{A.17})$$

wobei die Matrix $\mathbf{D}$ eine Diagonalmatrix mit nach rechts unten absteigend sortierten Einträgen ist, dann ist der Vektor $\mathbf{h}$ die letzte Spalte der Matrix $\mathbf{V}$ und die Matrix $\mathbf{H}$ kann aus diesem Vektor nach (A.13) gewonnen werden.

Für eine ausführlichere Diskussion der Berechnung und der Eigenschaften einer Homographie und einer *Singular Value Decomposition* sei auf [HZ00] verwiesen.

Die *Homographie* wird hier benutzt um einen mathematischen Zusammenhang zwischen der Bildebene und einem Modell der aufgenommenen Szene herzustellen. Da-

bei werden zumindest 4 Punkte in der Szene, welche zumeist durch Bodenmarkierungen auf der Straße gegeben sind, definiert und deren Abbildung in der Bildebene und reale Abstände in der Szene zur Berechnung der *Homographie* verwendet. Dadurch lässt sich eine grobe Kalibrierung der Kamera vornehmen und eine Schätzung des Abstandes des Fahrzeugs zur Kamera errechnen. Außerdem kann die Bewegung eines Objekts durch die Szene im Modell nachvollzogen werden und zusätzliche Information, wie zum Beispiel über die Durchführung eines Überholmanövers oder das Tätigen eines Spurwechsels, abgeleitet werden.

REFERENZEN

- [AZ01] J. Aggarwal and Q. Zhou. Tracking and Classifying Moving Objects from Videos. In *Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.
- [BFH99] W. Bell, P. Felzenszwalb, and D. Huttenlocher. Detection and Long Term Tracking of Moving Objects in Aerial Video. Technical report, Cornell University Ithaca, NY, 1999.
- [BHD96] M. Betke, E. Haritaglu, and L. Davis. Multiple Vehicle Detection and Tracking in Hard Real-Time. *IEEE Intelligent Vehicles Symposium*, pages 351–356, 1996.
- [BHKT88] A. T. Bergan, L. Henion, M. Krukar, and B. Taylor. Electronic License Plate Technology: Automatic Vehicle Location and Identification. *Canadian Journal of Civil Engineering*, Vol. 15 (6):1035–1042, 1988.
- [Bir97] S. Birchfeld. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. <http://www.ces.clemson.edu/~stb/klt>, 1997.
- [Boy88] D. E. Boyce. Route Guidance Systems for Improving Urban Travel and Location Choices. *Transportation Research A*, Vol. 22A (4):275–281, 1988.
- [CBMM98] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance. *Transportation Research: Part C*, 6 - 4:271–288, 1998.
- [CGPP01] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting Color and Motion Information. In *CIAP01*, pages 360–365, 2001.
- [CK03] S. S. Cheung and C. Kamath. Robust Techniques for Background Subtraction in Urban Traffic Video. *Lawrence Livermore National Laboratory; Livermore, CA*, 2003.
- [CPP03] R. Cucchiara, M. Piccardi, and A. Prati. Detecting Moving Objects, Ghosts and Shadows in Video Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1337–1342, 2003.

- [Cro84] F. Crow. Summed-Area Tables for Texture Mapping. *Proceedings of SIGGRAPH*, Vol. 18:207–212, 1984.
- [DT97] F. Dellaert and C. Thorpe. Robust Car Tracking using Kalman Filtering and Bayesian Templates. *Conference on Intelligent Transportation Systems*, 1997.
- [EHD99] A. Elgammal, D. Harwood, and L. Davis. Non Parametric Model for Background Subtraction. In *Proceedings of IEEE ICCV '99 Frame-rate Workshop*, 1999.
- [FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.
- [FWSB95] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker. A Generic Deformable Model for Vehicle Recognition. In *Proceedings of the British Machine Vision Conference*, volume Vol. 1, pages 127–136, September 1995.
- [Gel74] A. Gelb. *Applied Optimal Estimation*. MIT Press Cambridge, MA, 1974.
- [GNW96] C. Goerick, D. Noll, and M. Werner. Artificial Neural Networks in Real-Time Car Detection and Tracking Applications. *Pattern Recognition Letters*, 17(4):335–343, April 1996.
- [HZ00] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [IB98] M. Isard and A. Blake. Condensation – Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, Vol. 29 (1):5–28, 1998.
- [Int04] Intel. Intel Performance Primitives Version 4.0. <http://www.intel.com>, 2004.
- [Jac93] O. L. R. Jacobs. *Introduction to Control Theory, 2nd Edition*. Oxford University Press, 1993.
- [JW92] C. Jiang and M. O. Ward. Shadow Identification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 606–612, 1992.
- [Kal60] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME–Journal of Basic Engineering*, pages 35–45, 1960.
- [KB90] K.P. Karmann and A. Brandt. Moving Object Recognition Using an Adaptive Background Memory. *Time-Varying Image Processing and Moving Object Recognition*, Vol. 2:289–307, 1990. Elsevier Science Publishers B.V.
- [KDN93] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes. *International Journal of Computer Vision*, Vol. 10:257–281, 1993.

- [KN97] H. Kollnig and H.H. Nagel. 3D Pose Estimation by Directly Matching Polyhedral Models to Gray Value Gradients. *International Journal of Computer Vision*, 23(3):283–302, June 1997.
- [Kue91] A. Kuehnle. Symmetry-Based Recognition for Vehicle Rears. *Pattern Recognition Letters*, Vol. 12:249–258, 1991.
- [KWM93] D. Koller, J. Weber, and J. Malik. Robust Multiple Car Tracking with Occlusion Reasoning. Technical report, UCB/CSD-98-780, EECS Department, University of California, Berkeley, Oct. 1993.
- [LK81] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
- [LKP02] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. Technical report, MRL, 2002.
- [LM02] R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. *IEEE ICIP*, Vol. 1:900–903, 2002.
- [MA99] E. B. Meier and F. Ade. Tracking Cars in Range Images Using the Condensation Algorithm. *IEEE/IEEEJ/JSAI International Conference on Intelligent Transportation Systems ITSC'99 Tokyo, Japan*, pages 129–134, Oct. 1999.
- [MACH96] N. D. Matthews, P. E. An, D. Charnley, and C. J. Harris. Vehicle Detection and Recognition in Greyscale Imagery. In *Proceedings of 2nd Int. Workshop on Intelligent Autonomous Vehicles*, 1996.
- [Mal89] S. G. Mallat. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.
- [MCKT00] I. Mikić, P.C. Cosman, G.T. Kogut, and M.M. Trivedi. Moving Shadow and Object Detection in Traffic Scenes. In *ICPR00*, volume Vol. 1, pages 321–324, 2000.
- [MS95] N. McFarlane and C. Schofield. Segmentation and Tracking of Piglets in Images. *Machine Vision and Applications*, 8(3):187–193, 1995.
- [PMGT01] A. Prati, I. Mikic, C. Grana, and M. Trivedi. Shadow Detection Algorithms for Traffic Flow Analysis: A Comparative Study. In *Proceedings of the IEEE Intelligent Transportation Systems Conference, Oakland, CA*, Aug. 2001.
- [PMT03] A. Prati, I. Mikic, M.M. Trivedi, and R. Cucchiara. Detecting Moving Shadows: Algorithms and Evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 25(7):918–923, July 2003.

- [POP98] C. Papageorgiou, M. Oren, and T. Poggio. A General Framework for Object Detection. *International Conference on Computer Vision*, 1998.
- [R⁺97] P. Remagnino et al. An Integrated Traffic and Pedestrian Model-Based Vision System. In *Proceedings of the Eighth British Machine Vision Conference*, pages 380–389, 1997.
- [SBM03] Z. Sun, , G. Bebis, and R. Miller. Evolutionary Gabor Filter Optimization with Application to Vehicle Detection. In *Proceedings of the Third IEEE International Conference on Data Mining*, page 307. IEEE Computer Society, 2003.
- [SDS95a] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for Computer Graphics: A primer, Part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, May 1995.
- [SDS95b] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for Computer Graphics: A Primer, Part 2. *IEEE Computer Graphics and Applications*, 15(4):75–85, July 1995.
- [SF95] R.E. Schapire and Y. Freund. A Decision-Theoretic Generalization of on-line Learning and an Application to Boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, March 1995.
- [SF96] R. E. Schapire and Y. Freund. Experiments with a New Boosting Algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156. Morgan Kaufman, San Francisco, 1996.
- [SG00] C. Stauffer and W. Grimson. Learning Patterns of Activity Using Real-Time Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22:747–757, Aug. 2000.
- [SHB98] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision, Second Edition*. PWS Publishing, 1998.
- [SMO99] J. Stauder, R. Mech, and J. Ostermann. Detection of Moving Cast Shadows for Object Segmentation. *IEEE Transactions on Multimedia*, Vol. 1 (1):65–76, Mar. 1999.
- [ST94] J. Shi and C. Tomasi. Good Features to Track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.
- [TFTR98] T. Tommasini, A. Fusiello, E. Trucco, and Vito Roberto. Making Good Features Track Better. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 178–183. IEEE Computer Society Press, June 1998.
- [TK91] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

- [TKBM99] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and Practice of Background Maintenance. *ICCV*, Vol. 1:255–261, 1999.
- [Tro04] Trolltech. QT. <http://www.trolltech.com>, 2004.
- [TS98] C. Tzomakas and W. V. Seelen. Vehicle Detection in Track Scenes Using Shadows. *Internal Report 98-06, Institut für Neuroinformatik, Ruhr Universität Bochum, Deutschland*, 1998.
- [VJ01] P. Viola and M. J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE CVPR*, 2001.
- [WB95] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, TR95-041, 1995.
- [Woo04] Denton Woods. DevIL: A Full Featured Cross-Platform Image Library. <http://openil.sourceforge.net>, 2000–2004. formally known as "OpenIL".
- [WSB94] A. D. Worrall, G. D. Sullivan, and K. D. Baker. A Simple, Intuitive Camera Calibration Tool for Natural Images. In *Proceedings of the British Machine Vision Conference*, volume Vol. 2, pages 781–790. BMVA Press, 1994.
- [Yal98] University Yale. The XVision2 Project. <http://www.cs.jhu.edu/CIRL/XVision2>, 1993-1998. XVision originally developed by Greg Hager, completely rewritten at Yale University.
- [YKA02] M. H. Yang, D. J. Kriegman, and N. Ahuja. Detecting Faces in Images: A Survey. *IEEE Transactions on pattern analysis and machine vision*, 24(1):34–58, Jan. 2002.
- [ZBS93] T. Zielke, M. Brauckmann, and W. V. Seelen. Intensity and Edge-Based Symmetry Detection with an Application to Car-Following. *CVGIP:Image Understanding*, Vol. 58:177–190, 1993.