“Multi-Class-Boosting” für die konturbasierte Objektkategorisierung

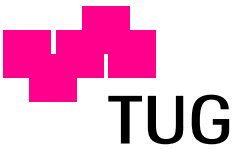
Michael Eberle
michael@eberle.at

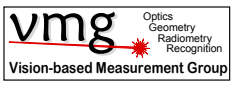
Institut für Elektrische Meßtechnik
und Meßsignalverarbeitung
Technische Universität Graz
Kopernikusgasse 24/IV
8010 Graz

Masterarbeit

Betreuer: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Axel Pinz

Graz, Oktober 2010





Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

Deutsche Fassung:

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Abstract

Automatic systems often use cameras to gather information about their environment. The images have to be understood to provide only useful parts of the digitalized information to the user. Object categorization tries to find objects in images and to assign a class label to them, e.g. “this is a horse”. In 2006, a new algorithm was published, which can classify objects by their contours, but each category has to be learnt separately. This Boundary-Fragment-Model (BFM) is very good in separating objects from the background. A problem occurs, when more than one learnt model is voting for the same object. There is no principled method to decide, which class is the right one. In this work, the used learning-algorithm is replaced with a multi-class-algorithm. The goal of this thesis is to jointly train different models, so that the resulting multi-class-model can decide the correct class label for a given detection. First, the existing BFM algorithm had to be adapted because of its excessive computational complexity of several weeks of training time for one model. Next, the constellation of the parts in the model was redesigned. Many pairs of boundary-fragments, called weak detectors, form a BFM. The learning-algorithm combines them to minimize the training-error. Thus, it can happen that two highly similar boundary-fragments are selected as one weak detector. In the original BFM-algorithm this was avoided by clustering the individual fragments before the learning starts. Experiments showed that the comparison of two boundary-fragments is not easy. By introducing a geometric relation between the boundary-fragments in this work, the learning algorithm should prefer combinations, which are more distributed over the object, like head and tail of a horse. New experiments show, that the optical quality of the learnt models is comparable, so that clustering is unnecessary. However, the detection rates decrease, compared to the original BFM. The BFM works very well, if the chosen boundary-fragments are very discriminative. On the other hand, in multi-class training, the learning-algorithm selects boundary-fragments, which are good on all classes, meaning, they are less discriminative. In conclusion, similar categories might benefit from joint training, while quite different classes should be trained separately. For separately trained models, a principled method to compare competing detections has yet to be found.

Keywords: Object recognition, Object detection, Boundary-Fragment-Model, Multi-Class, Boosting

Kurzfassung

Automatische Systeme verwenden zunehmend Kameras, um Informationen über ihre Umgebung zu sammeln. Um die Daten für Benutzer auch verwendbar zu machen, müssen die gesammelten Informationen, die Bilder, zuerst verstanden werden. Die Objektkategorisierung versucht, Objekte in Bildern zu finden, und ihnen eine Klasse zuzuweisen, z.B. "das ist ein Pferd". Im Jahr 2006 wurde eine solche Methode vorgestellt, die anhand der im Bild gefundenen Konturen Objekte klassifizieren kann. Es muss jedoch jede Klasse getrennt trainiert werden. Dieses Boundary-Fragment-Model (BFM) kann sehr gut einzelne Objekte vom Hintergrund unterscheiden, allerdings fehlt eine Möglichkeit, zu entscheiden, welcher Klassifikator gültig ist, wenn mehrere an derselben Stelle ansprechen. In dieser Arbeit wird das verwendete Lernverfahren durch ein Multi-Class-Verfahren ersetzt und untersucht, ob es möglich ist, Modelle gemeinsam zu trainieren. Das bietet den Vorteil, dass die Entscheidung vom Klassifikator übernommen wird. Dafür sind einige Änderungen am Algorithmus notwendig, denn die Laufzeit zum Trainieren eines einzigen Modells beträgt mehrere Wochen. Des Weiteren wird auch die Zusammensetzung der einzelnen Teile eines Modells überarbeitet. Ein BFM besteht aus vielen Paaren von Konturstücken, die zusammen den Klassifikator ergeben. Diese werden vom Lernverfahren so kombiniert, dass der Trainingsfehler minimiert wird, d.h. es kann passieren, dass ein solches Paar aus zwei einander sehr ähnlichen Fragmenten besteht. Im BFM wurde versucht, durch vorheriges Clustern der Konturstücke, dies zu verhindern. Experimente haben gezeigt, dass es schwierig ist, Konturstücke miteinander zu vergleichen. Durch Einführung einer geometrischen Beziehung zwischen den Konturstücken, sollen in dieser Arbeit Paare bevorzugt werden, die besser am Objekt verteilt sind, wie z.B. Kopf und Schweif eines Pferdes. Die Resultate der Experimente zeigen, dass durch Einführung dieser Beziehung die optische Qualität der Modelle vergleichbar bleibt, allerdings sinkt die Erkennungsrate des Multi-Class-Klassifikators im Vergleich zum Original. Das BFM funktioniert gut, wenn sehr diskriminative Konturstücke ausgewählt werden. Werden Modelle aber gemeinsam trainiert, werden Konturstücke bevorzugt, die in möglichst allen Klassen passen, d.h. die Diskriminativität geht verloren. Ähnliche Klassen profitieren möglicherweise von dieser Methode, für stark unterschiedliche Klassen ist das gemeinsame Training aber nicht zielführend. Vielversprechender ist es, einen Weg zu finden, um einzelne Modelle vergleichbar zu machen.

Stichwörter: Objekterkennung, Objektlokalisierung, Boundary-Fragment-Model, Multi-Class, Boosting

Inhaltsverzeichnis

Kapitel 1

Einleitung

Objekterkennung ist in der Computer-Vision ein blühendes Forschungsgebiet. Nahezu überall wo Kameras im Einsatz sind, wird Objekterkennung benötigt. Einige Beispiele sind in Abbildung 1.1 zu sehen. Im Bereich der Robotik z.B., können diese nur mit der Umwelt agieren, wenn sie verstehen, was sie sehen. Auch in der Überwachung von Verkehr, Gebäuden und anderen sicherheitsrelevanten Szenen, wie z.B. beim Erkennen von Fußgängern vor einem Auto, wird Objekterkennung benötigt. Erste Ansätze lieferte der englische Mathematiker David Marr. Er beschrieb bereits 1982 in seinem Buch Vision [Marr, 1982] Methoden zur Objekterkennung. Die Technologie ist heute soweit ausgereift, dass es möglich ist, ein spezifisches Objekt sehr schnell und mit einer hohen Genauigkeit wieder zu erkennen. Computer sind in der spezifischen Objekterkennung sogar besser als Menschen, d.h. sie können z.B. das Cover einer bestimmten CD unter vielen mit einer höheren Genauigkeit wieder finden [Nistér u. Stewénius, 2006], als Menschen es könnten. Die Aufgabe wird aber ungemein schwieriger, wenn nicht ein spezifisches Objekt, sondern eine ganze Klasse von Objekten erkannt werden soll. In diesem Bereich ist der Mensch dem Computer noch weit überlegen. Wir können einem unbekannten Objekt sofort seine Klasse, also z.B. Fahrrad, Auto, Lebewesen, usw., zuweisen. Die Aufgabenstellung der generischen Objektkategorisierung ist v.a. für die gezeigten Beispiele von Interesse.

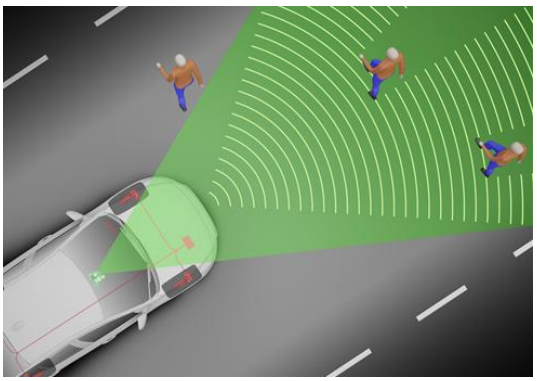
Ein Einsatzgebiet der spezifischen Objekterkennung ist z.B. die Augumented Reality, also eine virtuelle Abbildung der Umgebung in Computern. Gerade im aufblühenden Smartphone-Markt wird dieses Thema stark beworben. Solche Programme ermöglichen es z.B., zusätzliche Informationen zur tatsächlichen Umgebung zu erfahren. Mit Hilfe von Bildern, Position und Orientierung wird die Umgebung analysiert. Werden Objekte erkannt, können vom Internet zusätzliche Informationen über diese bezogen werden, wie z.B. die Namen der Häuser in Abbildung 1.2. Im Gegensatz dazu wird die Objektkategorisierung bereits in Digitalkameras verwendet. Sie erkennen Personen bzw. deren Gesichter und stellen automatisch auf diese scharf, bzw. starten den Selbstauslöser, wenn eine neue Person das Bild betritt.

Die Objektkategorisierung in der Computer-Vision lässt sich meistens in folgende drei Punkte aufteilen:

1. Extrahieren markanter Merkmale von den zu lernenden Objekten
2. Trainieren eines Modelles mit diesen Merkmalen
3. Finden der Merkmale bzw. des Modelles in neuen Bildern

(a) Ein humanoider Roboter der Firma Aldebaran. Eingesetzt wird er hauptsächlich in der Forschung und im RoboCup, einem Fußballturnier für Roboter.

(b) Um Sicherheitspersonal unterstützen zu können, muss die Software hinter Überwachungskameras verstehen, was passiert. Das Bild stammt vom britischen Künstler Banksy [2007].



(c) Der Volvo S60 erkennt Fußgänger vor dem Auto und kann notfalls automatisch bremsen.

Abbildung 1.1: Visuelle Objekterkennung wird nicht nur in der Forschung, sondern bereits auch in der Praxis eingesetzt. Beispiele sind u.a. die Robotik (a), Sicherheits- bzw. Überwachungskameras (b) und im Auto (c) z.B. zur Erkennung von Fußgängern.

Abbildung 1.2: Das Programm Wikitude zeigt zusätzliche Informationen zu den erkannten Objekten im Bild an.

Markante Merkmale können beispielsweise Kontur, Textur oder Ausschnitte von Objekten sein. Diese werden mit einem Lernverfahren zu einem Modell zusammengestellt. Die Lernverfahren basieren meist auf statistischen Betrachtungen, Boosting ist z.B. ein solches. Die einzelnen Teile, also die extrahierten Merkmale, eines Modelles, werden häufig weak-classifier genannt, denn jedes Merkmal für sich ist nicht aussagekräftig genug.

1.1 Motivation

Im Jahr 2006 veröffentlichten Opelt u. a. [2006a] eine neue Methode zur Objektkategorisierung, das Boundary-Fragment-Model (BFM). Im Gegensatz zu vielen anderen, basiert ihre Methode nicht auf Ausschnitten der Objekte sondern auf deren Kontur. Die zu Grunde liegende Idee, Kanten und nicht Ausschnitte zu verwenden war, dass Menschen Objekte sehr gut anhand deren Kontur bzw. Silhouette erkennen können. Versuchen Sie selbst, das Objekt in Abbildung 1.3 zu erraten. Die Resultate von Opelt u. a. [2006a] waren sehr vielversprechend, allerdings dauerte das Trainieren neuer Modelle mehrere Wochen. Versuche, neue Experimente durchzuführen und auch alte nachzuvollziehen wurden nicht nur wegen der langen Trainingsdauer, sondern auch wegen dem fehlenden Verständnis für den Sourcecode abgebrochen. Durch eine strukturierte Reimplementierung konnte die Trainingsdauer auf einige Tage verkürzt werden. Es war nun möglich, neue Experimente in einer akzeptablen Zeit durchzuführen und das Verhalten des BFMs zu untersuchen. Während diesen Experimenten tauchte die Fragestellung auf, ob es möglich ist, mehrere Modelle gleichzeitig und in Abhängigkeit voneinander zu trainieren. Hinter der Idee, die Modelle in Abhängigkeit voneinander zu trainieren, stand der Gedanke, dass ähnliche Objekte so besser voneinander unterschieden werden können. Das Lernverfahren soll also nicht nur die Klasse vom Hintergrund trennen, sondern auch Unterschiede zwischen den Klassen, also z.B. zwischen Pferd und Kuh, lernen. Opelt u. a. [2006b] haben sich ebenfalls mit diesem Thema auseinandergesetzt. Sie verwendeten zwar teilweise die gleichen Konturstücke für alle Modelle, allerdings geschah das Training unabhängig von den anderen Klassen.

Eine weitere Frage, die während den Experimenten auftauchte, war, ob es notwendig ist, Konturstücke vor dem Lernen zu clustern. Ein BFM besteht aus vielen Konturstücken,

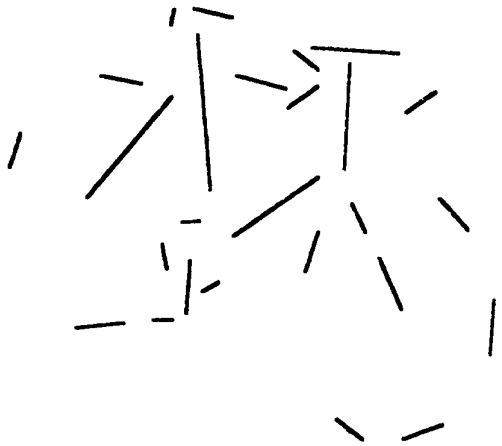


Abbildung 1.3: Menschen können Objekte sehr gut anhand deren Kontur erkennen [Lowe, 1985]. Erraten Sie, um was für ein Objekt es sich hier handelt?

und um zu verhindern, dass nur ähnliche Konturstücke, also z.B. verschiedene Köpfe von Pferden, zu einem Modell zusammengesetzt werden, haben Opelt u. a. [2006a] diese vorher geclustert. Experimente haben gezeigt, dass der Vergleich von verschiedenen Konturstücken nicht einfach ist. Eine mögliche Umgehung des Clusterns ist, im Lernverfahren Konturstücke zu bevorzugen, welche unterschiedliche Positionen am Objekt haben, wie z.B. Kopf und Schweif eines Pferdes.

Aufgrund der immer noch langen Trainingsdauer der Reimplementierung des BFMs konnte ein Multi-Class-Ansatz nicht direkt eingebaut werden. Die vorliegende Arbeit beschäftigt sich mit den notwendigen Adaptionen der Reimplementierung und den anschließenden Experimenten mit dem neuen Lernverfahren. Im ersten Teil wird ein kurzer Überblick über das BFM und das verwendete Lernverfahren, sowie anderer Arbeiten zum Thema Multi-Class-Objekterkennung gegeben. Danach werden die Änderungen beschrieben, um das vorhandene Lernverfahren durch ein Multi-Class-Verfahren zu ersetzen. Im letzten Teil der Arbeit werden die vorgenommenen Änderungen anhand verschiedener Experimente getestet und diskutiert.

1.2 Aufgabenstellung

Die Aufgabenstellung lässt sich in zwei Teile gliedern: Einerseits soll das verwendete Lernverfahren adaptiert und andererseits die Trainingsgeschwindigkeit optimiert werden. Das primäre Ziel dieser Arbeit soll aber nicht sein, die Erkennungsleistung des BFMs zu steigern, sondern zu untersuchen, ob es mit dem BFM möglich ist, mehrere Klassen in Abhängigkeit voneinander zu lernen. Als Ausgangspunkt der Arbeit, soll die Reimplementierung des BFMs verwendet werden. Diese zwei Punkte werden im Folgenden detaillierter beschrieben.

1.2.1 Adaption des Lernverfahrens

Das von Opelt u. a. [2006a] verwendete Lernverfahren zum Finden von Konturstücken soll adaptiert werden. Anstelle des binären Lernverfahrens, welches nur eine Klasse vom Hin-

tergrund trennen kann, soll ein Multi-Class-Verfahren verwendet werden. Da ein solches Verfahren rechenintensiver ist, müssen zusätzlich einige Änderungen an der Reimplementierung vorgenommen werden. Des Weiteren soll bei der Wahl geeigneter Konturstücke die geometrische Verteilung von diesen berücksichtigt werden.

1.2.2 Optimierung der Geschwindigkeit

Die meiste Rechenzeit im BFM wird für das Lernverfahren benötigt. Deshalb mussten Opelt u. a. [2006a] die Anzahl der Konturstücke stark begrenzen. Es konnten nur ca. 350 verarbeitet werden. In dieser Arbeit soll ein Multi-Class-Lernverfahren verwendet werden, deshalb kann es von Vorteil sein, wenn wesentlich mehr Konturstücke zur Verfügung stehen. Um die Laufzeit zu verbessern, sollen mögliche Vereinfachungen untersucht und die Struktur des Algorithmus dahingehend optimiert werden, dass Matlab-Spezifische Vorteile ausgenützt werden können.

Kapitel 2

Literaturrecherche

Zahlreiche Forscher haben sich bereits mit dem Thema der Multi-Class-Objekterkennung beschäftigt. In diesem Kapitel werden vorangegangene Arbeiten, die als Grundlagen dieser Arbeit dienen, vorgestellt. Es unterteilt sich in die Vorstellung des BFMs und des verwendeten Lernverfahrens sowie weiterer, relevanter Literatur anderer Autoren zum Thema Multi-Class-Objekterkennung. Es sei allerdings darauf hingewiesen, dass hier nur ein Überblick zu finden ist, der interessierte Leser wird daher auf die angegebenen Quellen verwiesen.

2.1 Das Boundary-Fragment-Model

Opelt u. a. [2006a] verfolgten in ihrer Arbeit die Idee, dass Objekte nur durch ihre Kontur beschrieben werden können. Sie trainierten einen Klassifikator, der aus mehreren Paaren von Konturstücken, sogenannten Boundary-Fragments (BFs), eines zu lernenden Objektes besteht. Die Information, wo sich ein bestimmtes BF am Objekt befindet, wird in Form eines Centroid-Votes (CVs) gespeichert, d.h. jedes BF besitzt einen Vektor, der zum Zentrum des Objektes zeigt. Im linken Bild der Abbildung 2.1, sind zwei solcher BFs mit ihren CVs dargestellt.

Gelernt wird ein BFM in zwei Schritten: (i) Zuerst werden viele BFs, normalerweise mehr als 1000, mit ihren CVs aus den Trainingsbildern ausgeschnitten. Von diesen BFs werden dann Kosten auf positiven (Objekt ist vorhanden) und negativen (nur Hintergrund) Validierungs-Bildern berechnet. Die Kosten beinhalten, wie gut ein BF in einem Bild übereinstimmt und wie gut der bekannte Objektmittelpunkt in den positiven Validierungs-Bildern mit dem CV des BFs getroffen wird. Jene 300 bis 350 BFs, die geringe Kosten in den positiven und hohe Kosten in den negativen Validierungs-Bilder besitzen, werden in den zweiten Schritt übernommen, d.h. BFs die nicht diskriminativ für eine Klasse sind, scheiden aus. (ii) Mit AdaBoost (siehe Kapitel 2.2) werden gute Kombinationen von zwei BFs gesucht und zu einem weak-classifier[1] zusammengesetzt. In Abbildung 2.1 ist dargestellt, wie ein solcher weak-classifier funktioniert: Die zwei BFs voten in einem Bild für das Zentrum des Objektes (linkes Bild). Der Vote ist allerdings nur gültig, wenn alle BFs des weak-classifiers für dasselbe Objektzentrum voten (Bild in der Mitte). Im Gegensatz zu Shotton u. a. [2004], die ein sehr ähnliches Verfahren zur Objektkategorisierung entwickelten, verwenden Opelt u. a. [2006a] nicht nur ein BF als weak-classifier. Dadurch

[1] Ein weak-classifier ist eine schwache Hypothese für das Vorkommen eines Objektes. Mehr dazu in Kapitel 2.2.

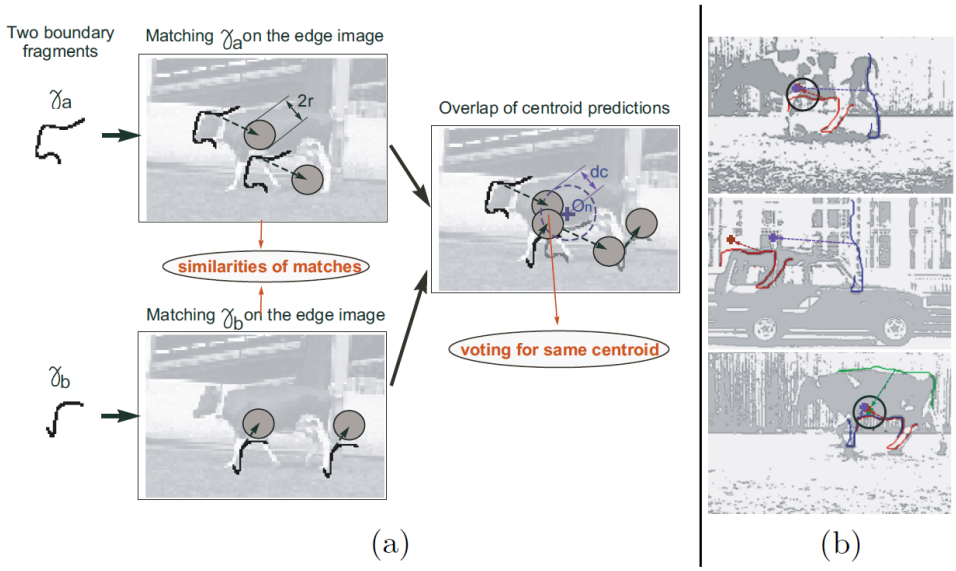


Abbildung 2.1: Die Funktion eines weak-classifiers. Zwei BFs γ_a und γ_b werden auf einem Bild getestet (links). Überlappen sich ihre CVs (Mitte), votet der weak-classifier an dieser Stelle für das Vorkommen eines Objektes (rechts oben). Tritt kein Überlapp der CVs auf, ist auch kein Objekt vorhanden (rechts Mitte). Es ist auch möglich, den weak-classifier aus mehr als zwei BFs zusammen zu setzen (z.B. rechts unten mit drei BFs). [Opelt u. a., 2006a].

sind sie weniger anfällig gegen Störungen, denn einzelne BFs werden sehr häufig auch im Hintergrund gefunden.

Die Vorgehensweise des Detektors ist in Abbildung 2.2 schematisch dargestellt. Die BFs des gelernten Modells werden mit Chamfer-Matching [Borgefors, 1988; Opelt u. a., 2006a] im Bild gesucht. Überlappen sich die CVs der BFs eines weak-classifiers, feuert[2] dieser an der Stelle in den Voting-Space und somit steigt dort die Wahrscheinlichkeit für das Vorhandensein eines Objektes. Nachdem alle Feuerungen der weak-classifiers in den Voting-Space eingetragen wurden, werden die Maxima mit einem Mean-Shift-Clustering-Algorithmus [Cheng, 1995] gesucht. Jene Maxima, die über einem definierten Threshold liegen, werden als Detektionen gewertet. Die Bounding-Box ergibt sich durch zurück projizieren jener BFs, die zu dieser Detektion beitragen.

2.2 Boosting - Ein Lernverfahren

Boosting gehört in der Computer Vision zum Lernen generativer Modelle, zu den gängigsten Lernverfahren und wird erfolgreich in zahlreichen Arbeiten wie z.B. in [Opelt u. a., 2004; Shotton u. a., 2004; Torralba u. a., 2004; Lin u. Liu, 2005] verwendet. Die Idee hinter dem Verfahren ist, mehrere schwache Hypothesen zu einer aussagekräftigen, starken Hypothese zu vereinigen, wobei die schwachen Hypothesen nur besser sein müssen, als zufälliges raten. Unter Raten wird in diesem Fall verstanden, ob die CVs der zwei BFs das Objektzentrum treffen oder nicht.

Eine schwache Hypothese ist eine Funktion h, die einzelnen Instanzen einen Klassen-

[2] Feuern bedeutet, dass ein weak-classifier bzw. ein BF im Bild gefunden wird, bzw. diese eine Detektion markieren.

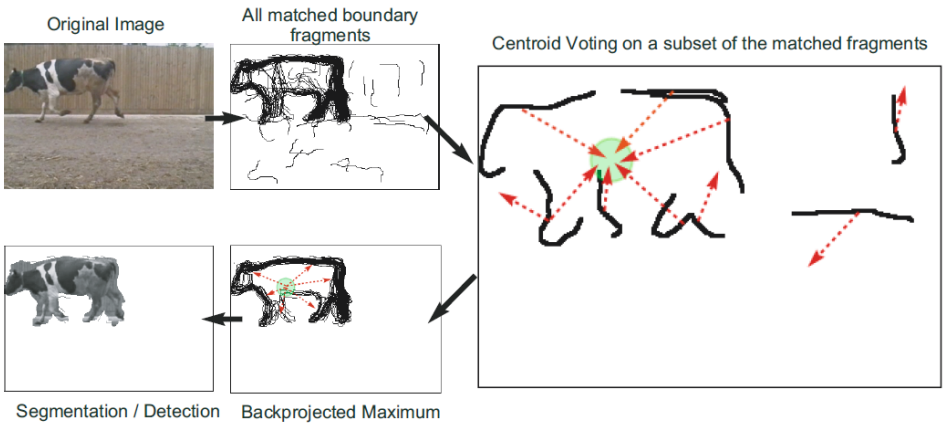


Abbildung 2.2: Die Prinzipielle Funktion des BFM-Detektors. Vom Eingabebild werden die Kanten berechnet und darauf die BFs der weak-classifier des gelernten Detektors gesucht. Die Feuerungen eines weak-classifiers werden in einen Voting-Space eingetragen. Die Maxima im Voting-Space entsprechen einer Detektion und durch zurück projizieren der BFs, die zu dieser Detektion beitragen, erhält man die Bounding-Box, bzw. die Segmentation. [Opelt u. a., 2006a].

label zuweisen kann, also

$$h(x_i) = \begin{cases} +1 & \text{wenn } x_i \text{ ein Objekt ist} \\ -1 & \text{wenn nicht} \end{cases}$$

Das Paar (x_i, y_i) definiert eine Instanz $x_i \in X$ und den zugehörigen Klassenlabel $y_i \in Y = \{-1, +1\}$. Die Menge X umfasst alle Instanzen des Objektes, wie z.B. verschiedene Autos, Fahrräder, Flaschen, usw. In einem iterativen Verfahren werden nun schwache Hypothesen h gesucht, die eine Anzahl "wichtiger" Trainingsdaten richtig klassifiziert. Die Wichtigkeit der einzelnen Trainingsdaten wird mit einer Verteilungsfunktion D bewertet, welche nach jeder Iteration aktualisiert wird. Werden Trainingsdaten mit der schwachen Hypothese der aktuellen Boosting-Runde richtig klassifiziert, werden die Gewichte dieser verringert, ansonsten werden sie vergrößert. Nach T Iterationen entsteht so ein Klassifikator H, der wesentlich besser ist, als die schwachen Hypothesen selbst.

Die ersten Algorithmen entstanden in den frühen 90er Jahren [Schapire, 1990; Freund, 1995] und wurden im Laufe der Zeit immer wieder verbessert und abgeändert, bis Freund u. Schapire [1997] den sogenannten AdaBoost[3]-Algorithmus vorstellten. Die Funktionsweise zum "Boosten" eines binären Klassifikators ist als Pseudocode in Algorithmus 1 dargestellt.

Es gibt zahlreiche Bemühungen, den binären AdaBoost-Algorithmus auf den Multi-Class-Fall zu erweitern, d.h. es treten nicht nur die Klassenlabel $Y = \{-1, +1\}$ auf, sondern $Y = \{1, 2, 3, \dots\}$. Schapire u. Singer [1998] versuchten z.B. das Multi-Class-Problem auf K binäre Probleme zu reduzieren, wobei K der Anzahl Klassen entspricht. Sie nannten den Algorithmus AdaBoost.MH. Einen ähnlichen, aber etwas robusteren Algorithmus namens SAMME haben Zhu u. a. [2006] vorgestellt. Wird nämlich der Fehler ϵ in Formel (2.1) einer schwachen Hypothese größer als $1/2$, wird laut Formel (2.2) der Gewichtungsfaktor $\alpha \leq 0$ und die Gewichte der Trainingsdaten werden falsch aktualisiert. Zhu u. a. [2006] führen einen Korrekturfaktor ein und begründen diesen damit, dass für K Klassen zufälliges Raten

[3]AdaBoost bedeutet Adaptive Boosting, d.h. der Klassifikator wird schrittweise zusammengesetzt.

Algorithmus 1 Der AdaBoost-Algorithmus für einen binären Klassifikator. [Freund u. Schapire, 1990].

Given Training-Data $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in Y = \{-1, +1\}$
Initialize Training-Weights $D_1(i) = 1/m$
for $t = 1$ to T **do**
Train weak learner using training-data and their weights D_t
Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \cdot \begin{cases} 1 & \text{if } h_t(x_i) \neq y_i \\ 0 & \text{otherwhise} \end{cases} \tag{2.1}$$

Choose

$$\alpha_t = \frac{1}{2} \cdot ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{2.2}$$

Update

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

Renormalize D_{t+1} so that it will be a distribution
end for
Output the final hypothesis

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right)$$

nicht mehr einem Fehler von $\epsilon = 1/2$ entspricht, sondern $\epsilon = 1 - 1/K$. In Algorithmus 2 ist wieder der Pseudocode dargestellt. Der zusätzliche Faktor ist in Formel (2.4) zu finden. Eine detaillierte Begründung für diese Erweiterung kann in [Zhu u. a., 2006] nachgelesen werden.

In dieser Arbeit wurde diese abgewandelte Form von AdaBoost.MH verwendet. Der Grund dafür war derselbe wie in [Zhu u. a., 2006] erwähnt wurde: AdaBoost.MH war nach einigen Iterationen nicht mehr in der Lage, eine schwache Hypothese zu finden, deren Fehler unter $\epsilon = 1/2$ war.

Algorithmus 2 Der SAMME-Algorithmus für einen Multi-Class-Klassifikator. [Zhu u. a., 2006].

Given Training-Data $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in Y = \{-1, +1\}$
Initialize Training-Weights $D_1(i) = 1/m$
for $t = 1$ to T **do**
Train weak learner using training-data and their weights D_t
Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \cdot \begin{cases} 1 & \text{if } h_t(x_i) \neq y_i \\ 0 & \text{otherwhise} \end{cases} \tag{2.3}$$

Choose

$$\alpha_t = \frac{1}{2} \cdot ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + log(K-1) \tag{2.4}$$

Update

$$\begin{aligned} D_{t+1}(i) &= D_t(i) \cdot \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= D_t(i) \cdot e^{-\alpha_t \cdot y_i \cdot h_t(x_i)} \end{aligned} \tag{2.5}$$

Renormalize D_{t+1} so that it will be a distribution
end for
Output the final hypothesis

$$H(x) = \arg\max_k \sum_{t=1}^{T} \alpha_t \cdot \begin{cases} 1 & \text{if } h_t(x) = k \\ 0 & \text{if } h_t(x) \neq k \end{cases} \tag{2.6}$$

2.3 Multi-Class-Objekterkennung

Für das BFM haben Opelt u. a. [2006b] bereits einen Multi-Class-Ansatz vorgestellt. Ihr Ziel war es, den Lernschritt des BFMs so zu adaptieren, dass BFs für mehrere Klassen verwendet werden können. Außerdem soll, aufgrund der extrem langen Trainingsdauer, das Modell nicht immer komplett neu trainiert werden müssen, wenn eine neue Klasse hinzugefügt werden soll. Dazu wurden, wie in [Opelt u. a., 2006a], BFs mit ihren CVs aus

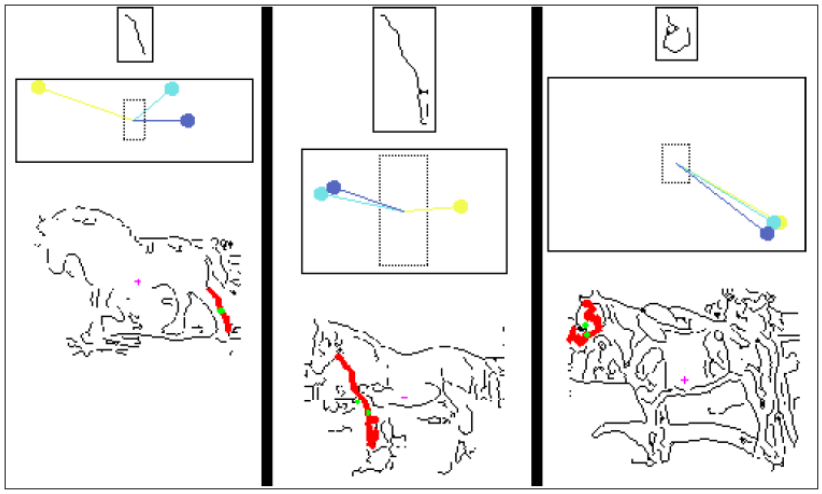


Abbildung 2.3: Drei verschiedene BFs (oben) mit ihren CVs (Mitte) und den Bildern, aus denen sie ausgeschnitten wurden (unten). [Opelt u. a., 2006b].

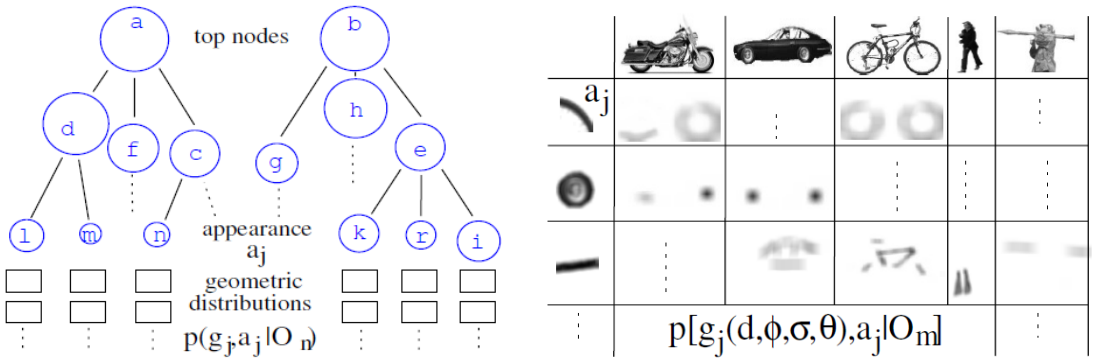
den einzelnen Klassen ausgeschnitten. Sind sich zwei BFs von der Form her ähnlich, erhält der Repräsentant einen zusätzlichen CV. Auch wenn ein BF einer Klasse in einer anderen Klasse geringe Kosten hat (siehe Kapitel 2.1), wird ein zusätzlicher CV hinzugefügt. Abbildung 2.3 zeigt drei Beispiele von BFs, die mehrere CVs besitzen. Es können aber nicht nur einzelne BFs mehreren Klassen angehören, sondern auch, die für den Lernschritt verwendeten weak-classifier, d.h. dem Lernverfahren wird erlaubt, weak-classifier anderer Klassen zu verwenden.

Die Idee, bereits vorhandene Informationen anderer Klassen für das Lernen neuer Klassen zu verwenden, haben Stark u. a. [2009] weitergeführt. Sie teilen neben der Information über das Aussehen der Objekte (z.B. die Form der Beine) und den Symmetrien (z.B. die Symmetrie zwischen Vorder- und Hinterbeinen bei Vierbeinern) auch noch die Anordnung einzelner Teile (z.B. die Anordnung von Kopf, Körper und Beinen). Durch verwenden dieser vorhandenen Informationen ist es ihnen möglich, neue Klassen mit sehr wenig Trainingsbildern zu lernen.

Eine weitere Methode der Multi-Class Objekterkennung haben Torralba u. a. [2006] beschrieben. Sie versuchen ebenfalls mit einem Boosting-Algorithmus weak-classifier zu finden, die möglichst viele Klassen richtig erkennen. Als Merkmal verwenden sie nicht Konturstücke, sondern extrahieren Patches von Objekten aus den Trainingsbildern.

Neben dem weit verbreiteten Boosting-Algorithmus, kommen auch noch andere Lernverfahren zum Einsatz. Mikolajczyk u. a. [2006] z.B., verwenden zwar kantenbasierte Merkmale, lernen aber einen bayes'schen Klassifikator. Sie beschreiben die Canny-Edges der Objekte mittels SIFT-Deskriptoren[4] und bauen durch ein agglomeratives Clusterverfahren eine hierarchische Struktur, bezogen auf die Ähnlichkeit der Deskriptoren, auf, dargestellt in Abbildung 2.4a. Gelernt werden anschließend zwei Parameter: (i) Die Wahrscheinlichkeit, wie ähnlich sich die extrahierten Deskriptoren aus den Trainings-Bildern der verschiedenen Klassen und jene aus den Validierungs-Bildern der verschiedenen Klassen sind, sowie (ii) die Wahrscheinlichkeit, dass ein aus den Trainings-Bildern extrahierter Deskriptor in den Validierungs-Bildern an derselben Stelle vorkommt (Abbildung 2.4b). Die Klassifikation neuer Bilder wird anschließend mit Hilfe von bayes'schen Entscheidungen durchgeführt.

[4]Scale Invariant Feature Transform. Mehr dazu in [Lowe, 1999].



(a) (b)

Abbildung 2.4: (a) Die hierarchische Struktur nach dem Clustern und (b) die geometrische Verteilung der Deskriptoren (Zeile) in verschiedenen Klassen (Spalte). [Mikolajczyk u. a., 2006].

Für die Modellierung von Objekten haben Shotton u. a. [2009] eine neue Methode vorgestellt: Sie modellieren Textur, Anordnung und den Zusammenhang zwischen Objekten. Mit einem Boosting-Algorithmus werden diese drei Merkmale zu einem Modell zusammengefügt, um damit in einem späteren Schritt neue Bilder segmentieren zu können, d.h. sie wollen nicht nur das Objekt und die Position bestimmen, sondern, wie in Abbildung 2.5 dargestellt ist, dieses auch segmentieren.

Dieser kurze Literaturüberblick zeigt, dass auch andere Forschungsgruppen in diesem Gebiet sehr stark tätig sind und hervorragende Ergebnisse liefern. Um die Multi-Class-Variante des BFMs von Opelt u. a. [2006b] ebenfalls weiter zu entwickeln, wird in den nächsten Kapiteln eine Methode untersucht, die aus Erfahrungen mit dem BFM entstanden ist.

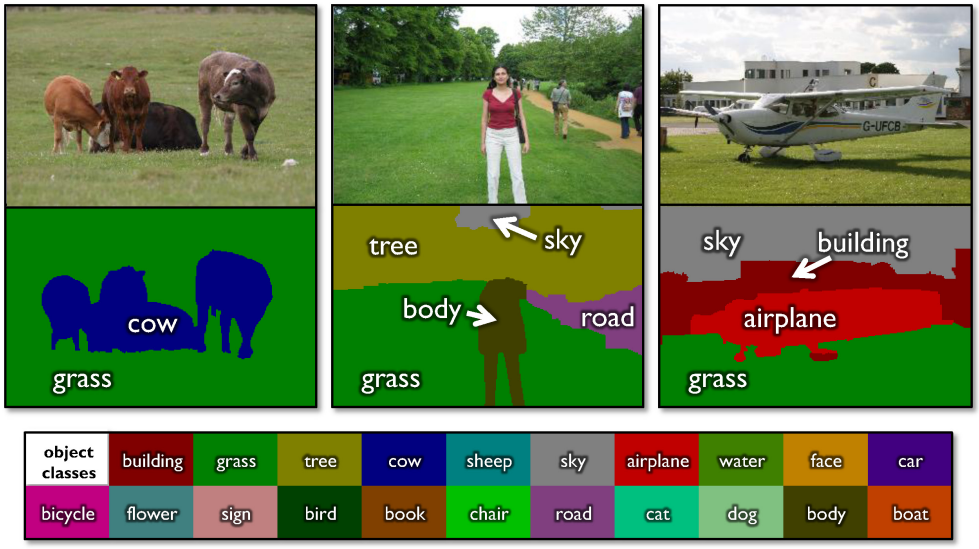


Abbildung 2.5: Beispiele von erfolgreicher Segmentation neuer Bilder mit dem Algorithmus von Shotton u. a. [2009].

Kapitel 3

Lernen von Modellen - Änderungen am Boundary-Fragment-Model

Wie bereits in Kapitel 1.1 auf der Seite 3 beschrieben, soll in dieser Arbeit untersucht werden, ob mit dem BFM unterschiedliche Klassen in Abhängigkeit voneinander trainiert werden können. Dazu wird das von Opelt u. a. [2006a] verwendete Lernverfahren durch das in Kapitel 2.2 auf der Seite 7 bzw. in [Zhu u. a., 2006] beschriebene Multi-Class-Verfahren ersetzt. Die dafür notwendigen Maßnahmen werden in diesem Kapitel detailliert beschrieben. Nach einigen Überlegungen zur Verbesserung der Trainingsgeschwindigkeit, entstand eine neue Struktur des BFMs, welche in Abbildung 3.1 dargestellt ist. Ausgehend von den annotierten Trainingsbildern verschiedener Klassen, werden BFs ausgeschnitten und mit Hilfe der positiven Validierungs-Bilder mögliche CVs berechnet. Da in den positiven Validierungs-Bildern bereits sehr viel Hintergrund vorhanden ist, soll getestet werden, ob auf die negativen Validierungs-Bilder verzichtet werden kann. In dem Pool von ausgeschnittenen BFs, von denen jedes mehrere CVs für unterschiedliche Klassen besitzen kann, werden mit einen Multi-Class-Boosting-Algorithums gute zweier-Kombinationen gesucht, und zu einem Klassifikator zusammengesetzt. Im Laufe dieses Kapitels werden die einzelnen Schritte, vom Ausschneiden der BFs bis zum fertigen Klassifikator, detailliert erklärt. Die Experimente in diesem Kapitel sollen nur der Erklärung dienen, eine genaue Untersuchung wird im Kapitel 5 ab der Seite 32 durchgeführt.

3.1 Ausschneiden von Boundary-Fragments

Die Grundidee des BFMs ist die Beschreibung der Objekte durch ihre Kontur, d.h. es werden BFs als Deskriptor für Objekte verwendet[1]. Diese BFs werden aus den Trainingsbildern extrahiert. Dazu ist es erforderlich, dass die Trainingsbilder vorher annotiert wurden, d.h. die Objekte sind durch eine Bounding-Box (vgl. Abbildung A.1 auf der Seite 51) und den zugehörigen Klassenlabel[2] als solche gekennzeichnet. Die Vorgehensweise um aus einem Trainingsbild ein BF zu extrahieren ist folgendermaßen:

1. Die annotierten Objekte werden aus den Trainingsbildern ausgeschnitten und auf

[1] Genauer gesagt werden Paare von BFs verwendet (siehe Kapitel 3.5 auf der Seite 24)
[2] Die Bezeichnung der Klasse, z.B. Auto, Fahrrad, ...

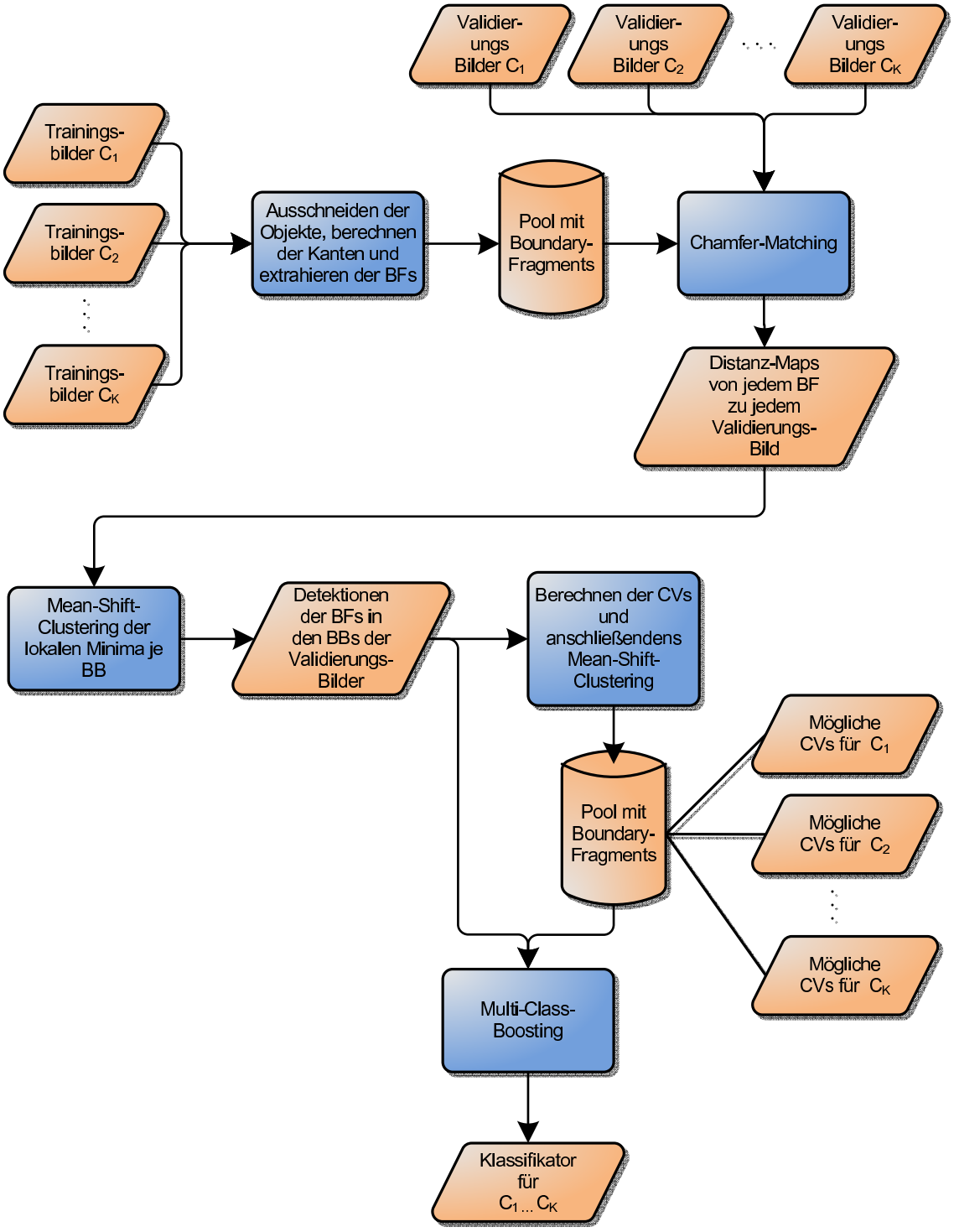


Abbildung 3.1: Schematischer Aufbau des neuen BFM-Algorithmus. Aus den Trainingsbildern der Klassen C_1 bis C_K werden zuerst BFs ausgeschnitten und anschließend in den Validierungs-Bildern gesucht. Werden BFs in der Bounding-Box gefunden, werden für diese CVs berechnet und mit den BFs abgespeichert. Anschließend werden mit einem Multi-Class-Boosting-Algorithmus gute Kombinationen von BFs und CVs gesucht und zu einem Klassifikator zusammengesetzt. Die Abkürzung BB steht für Bounding-Box.

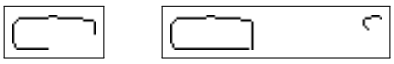


Abbildung 3.2: Ein BF unterschiedlicher Länge. Das Linke BF hat eine Länge von 69 Pixeln. Da die Kontur in den Bildern nicht immer zusammenhängend ist, kann es passieren, dass BFs beim Wachsen (rechts mit einer Länge von 109 Pixeln) auseinander reißen.

die gleiche Größe skaliert.

2. Mit einem Canny-Edge-Detektor werden die Kanten berechnet und anschließend zu einer zusammenhängenden Kontur verknüpft. Dieser Schritt ist notwendig, um im nächsten BFs ausschneiden zu können.

3. Auf die Kontur werden zufällig eine Anzahl von Startpunkten gesetzt. Ausgehend von diesen Punkten werden die Kanten mit verschiedenen Längen extrahiert. Diese Kanten stellen die BFs dar. In Abbildung 3.2 links ist ein solches BF zu sehen. Rechts ist dasselbe BF abgebildet, jedoch in beide Richtungen um 20 Pixel verlängert.

Die Anzahl der verwendeten Startpunkte pro Bild und die Anzahl der Längen die ausgeschnitten werden, sind für die Qualität des zu lernenden Klassifikators von entscheidender Bedeutung. Die von uns verwendeten Trainingsbilder wurden unter realen Bedingungen aufgenommen und von Hand annotiert, d.h. trotz Bounding-Box um das Objekt, ist noch Hintergrund vorhanden. Umso weniger Punkte pro Bild verwendet werden, desto stärker sinkt die Wahrscheinlichkeit, dass eine tatsächliche Objektkante ausgeschnitten wird. Es werden also im späteren Lernschritt BFs zu einem Klassifikator zusammengesetzt, die eigentlich gar nicht von den Objekten stammen.

Natürlich wird es bei einer großen Anzahl von BFs passieren, dass sich mehrere sehr ähnlich sind, wie z.B. die Vorderräder von drei unterschiedlichen Fahrrädern in Abbildung 3.3. Wenn nun ein BF im Lernschritt sehr gut abschneidet, schneidet ein ähnliches ebenfalls sehr gut ab, und es werden zwei nahezu gleiche BFs, vgl. Abbildung 3.4 rechts, zu einem weak-classifier zusammengesetzt. Das ist aber nicht zielführend, denn dann wird praktisch, wie in [Shotton u. a., 2004], nur ein einziges BF als weak-classifier verwendet. Um das zu vermeiden haben Opelt u. a. [2006a] die BFs vor der weiteren Verwendung geclustert. Bei der Reimplementierung des BFMs haben wir bemerkt, dass es nicht ganz trivial ist, zwei BFs miteinander zu vergleichen. Der Mensch kann sofort erkennen, dass sich die drei BFs in Abbildung 3.3 ähnlich sind, und kann ein neues BF skizzieren, welches diese drei repräsentiert. In der Computer-Vision gibt es zahlreiche Algorithmen, welche die Ähnlichkeit zweier Konturen beschreiben, wie z.B. die Hausdorff-Distanz [Huttenlocher u. a., 1993], die Fréchet-Distanz [Alt u. a., 2001] oder die Earth-Movers-Distance [Grauman u. Darrell, 2004]. Diese sind aber entweder sehr rechenintensiv (im Sekundenbereich) oder können nicht mit Konturen umgehen, die unterschiedliche Start- bzw. Endpunkte besitzen[3], wie z.B. das BF in Abbildung 3.2. Einen guten Überblick über all diese Verfahren gibt Veltkamp [2001]. Ein weiteres Problem beim Clustern von BFs ist die Wahl des Repräsentanten eines Clusters, also welcher von z.B. den drei BFs in Abbildung 3.3 soll weiter verwendet werden? Im folgenden Kapitel ist ein neuer Ansatz beschrieben, der das Clustering, und die damit verbundenen Probleme, überflüssig machen soll.

[3]Vor allem die Fréchet-Distanz vergleicht Polygone, die dieselben Start-, bzw. Endpunkte besitzen.

Abbildung 3.3: Drei ähnliche BFs von unterschiedlichen Vorderrädern von Fahrrädern. Im Boosting-Algorithmus muss irgendwie verhindert werden, dass zwei ähnliche BFs zu einem weak-classifier zusammengesetzt werden.

3.2 Eine geometrische Beziehung zwischen zwei Boundary-Fragments

Bei der Wahl guter weak-classifier soll es nicht passieren, dass zwei ähnliche BFs mit ähnlichen CVs gewählt werden. Vielmehr ist es gewünscht, wenn ein weak-classifier möglichst viel vom zu lernenden Objekt abdeckt, wie z.B. Kopf und Fuß einer Person, Kopf und Schweif eines Pferdes oder Hals und Boden einer Flasche. In Abbildung 3.4 sind zwei Beispiele eines weak-classifiers zu sehen. Die Kombination auf der linken Seite ist gut, denn der Kopf eines Menschen ist vorstellbar, während jene auf der rechten Seite nicht sehr aussagekräftig ist. Sie soll ein Teil eines Fahrrades darstellen. Im Lernverfahren sollen also Kombinationen bevorzugt werden, die unterschiedliche CVs besitzen, egal ob sich die BFs ähnlich sind. Dadurch ist ein vorheriges Clustern der BFs nicht mehr notwendig. Um die gewünschte geometrische Streuung zu erreichen, wird beim Suchen einer geeigneten Kombination von BFs (Kapitel 3.4 auf der Seite 22) diese durch ein zusätzliches Gewicht mit der Formel

$$\delta = \frac{\|\boldsymbol{\zeta}_1 - \boldsymbol{\zeta}_2\|_2}{\|\boldsymbol{\zeta}_1\|_2 + \|\boldsymbol{\zeta}_2\|_2} \tag{3.1}$$

bewertet, wobei $\boldsymbol{\zeta}_1$ und $\boldsymbol{\zeta}_2$ die Vektoren der CVs der beiden BFs sind. Je kleiner der Winkel zwischen den CVs ist, und je ähnlicher die Längen, umso kleiner wird δ. Die Verteilung der Werte von δ mit unterschiedlichen Winkel und Längen ist in Abbildung 3.5 grafisch dargestellt. Dabei wurde $\boldsymbol{\zeta}_1 = [0\ 1]^T$ fixiert und die Formel (3.1) für jedes mögliche $\boldsymbol{\zeta}_2 \in \mathbb{R}^2$ mit $\|\boldsymbol{\zeta}_2\|_2 <= \|\boldsymbol{\zeta}_1\|_2$ ausgewertet. Wie zu sehen ist, haben Winkel und Längen, die sich im oberen Drittel des Kreises befinden ein $\delta < 0.4$ und solche, die sich im unteren Drittel befinden ein $\delta > 0.8$, d.h. Kombinationen die gewünscht sind, werden nur wenig bestraft, alle anderen sehr viel mehr.

3.3 Optimierung der Performance - Vorbereiten der Daten für das Boosting

Eine Notwendigkeit, um das Multi-Class-Lernverfahren zu verwenden, ist die Optimierung der Geschwindigkeit zum Auswählen von weak-classifiers. Eine Schwachstelle von [Opelt u. a., 2006b] war, dass die Anzahl von BFs im Lernschritt stark begrenzt werden musste (auf ca. $n = 300 \ldots 350$). Im BFM werden Paare von BFs als weak-classifier verwendet,

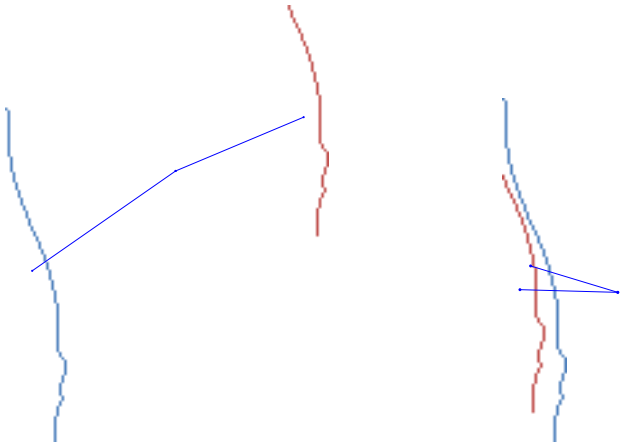


Abbildung 3.4: Zwei ähnliche BFs mit einer erwünschten (links) und einer unerwünschten (rechts) geometrischen Beziehung. Die linke Kombination stammt aus einem Klassifikator für Gesichter, die rechte aus einem für Fahrräder.

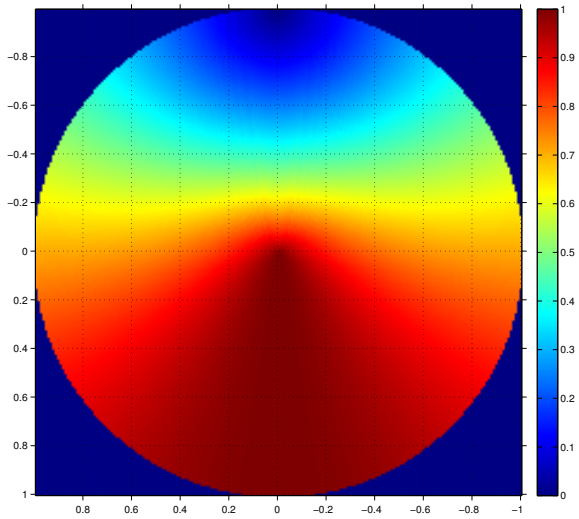


Abbildung 3.5: Grafische Darstellung der Gewichtungsformel (3.1) für die geometrische Beziehung, ausgewertet für alle $\boldsymbol{\zeta}_2 \in \mathbb{R}^2$ mit $\|\boldsymbol{\zeta}_2\|_2 <= \|\boldsymbol{\zeta}_1\|_2$. Kombinationen, deren Winkel zwischen den CVs groß sind, bzw. je unterschiedlicher die Längen der CVs sind, sollen bevorzugt werden.

d.h. es gibt pro Boosting-Schritt entsprechend dem Binomialkoeffizienten

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} 350 \\ 2 \end{array} \right) = 61075$$

mögliche Kombinationen. Um die Beste auszuwählen, müssen alle Kombinationen auf den Validierungs-Bildern getestet werden, was in der originalen BFM-Software sehr lange dauert. In der vorliegenden Arbeit wollen wir aber in der Lage sein, mehrere Klassen gleichzeitig zu trainieren, d.h. es werden auch mehr BFs benötigt, um alle Klassen beschreiben zu können. Wird davon ausgegangen, dass 350 BFs zum Trainieren einer Klasse ausreichen, würden wir für unsere Experimente mit 10 Klassen (siehe Kapitel 5 auf der Seite 32) 3500, also ca. 4000 BFs benötigen. Es ist selbstverständlich, dass bei 4000 BFs einige redundant sind, allerdings werden bei 20 Trainingsbildern pro Klasse nur 20 BFs pro Objekt ausgeschnitten, und das ist bei Bildern mit stark strukturiertem Hintergrund bereits sehr wenig.

Bei $n = 4000$ BFs sind

$$\left(\begin{array}{c} 4000 \\ 2 \end{array} \right) = 7998000$$

Kombinationen möglich. Um alle in einer vernünftigen Zeit (innerhalb von wenigen Stunden) testen zu können, müssen einige Daten für den Lernschritt vorbereitet werden. Erstens werden die ausgeschnittenen BFs bereits vor dem Lernverfahren mit den Validierungs-Bildern verglichen und die möglichen CVs extrahiert und zweitens, werden die Daten so organisiert, dass eine effiziente Implementierung möglich ist.

3.3.1 Vorausrechnen der Chamfer-Distanz

Chamfer-Matching ist eine gängige Methode, um Templates (in diesem Fall ein BF) in einem binären Kantenbild zu finden. Die Methode ist in [Borgefors, 1988] ausführlich beschrieben und wird im Folgenden nur kurz skizziert:

Auf ein binäres Kantenbild (Abbildung 3.6) wird eine Distanztransformation angewandt, d.h. nach der Transformation enthält jedes Pixel die euklidische Distanz zum nächsten Kantenpunkt. Wird dieses Bild mit dem gesuchten Template gefaltet entsteht eine Distanz-Map, die an jeder Stelle im Bild die Ähnlichkeit mit dem Template beinhaltet. Je geringer die Distanz, desto besser passt das Template an dieser Stelle in das Bild. Probleme treten allerdings auf, wenn Bereiche im Bild stark strukturiert sind, denn dort sind die Werte der Distanz-Transformation sehr klein, d.h. die Chamfer-Distanz ist dort auch sehr klein und es entstehen falsche Detektionen. Opelt u. a. [2006a] haben dieses Problem mittels Orientierungsebenen entschärft. Das Kantenbild und das BF werden jeweils in acht verschiedene Orientierungen unterteilt. Von diesen wird die Chamfer-Distanz berechnet und anschließend wieder zusammengeführt. Da die Chamfer-Distanz eine lineare Funktion ist, ist dieses Vorgehen erlaubt. Des Weiteren sind auch die Längen der BFs nicht immer dieselben, deshalb wird die Distanz vor der weiteren Verwendung noch mit der Länge des BFs normiert.

Um später die CVs der BFs zu bestimmen, muss jedes BF auf jedem Validierungs-Bild gematcht werden. Das kann vorab und auch auf mehreren CPUs parallel passieren. Die Dauer zum Berechnen einer Chamfer-Map hängt von der Größe des BFs und des Bildes ab und beträgt bei den Experimenten in Kapitel 5.4 ab der Seite 39 zwischen 0.1s und 0.3s pro BF und Bild. Die berechneten Chamfer-Maps werden gespeichert und können anschließend vom Lernschritt schnell wieder geladen werden. Durch das Vorausrechnen der Distanz von

Abbildung 3.6: Beispiel eines Validierungs-Bildes des Fahrraddatensatzes mit stark strukturiertem Hintergrund. Die Kanten wurden mit einem Canny-Edge-Detektor berechnet. Das rote BF wurde im Bild mittels Chamfer-Matching (aufgeteilt auf 8 Orientierungs-Ebenen) gesucht, und an der Stelle mit der geringsten Distanz eingezeichnet.

BFs zu den Validierungs-Bildern ist es später auch möglich, zeitaufwändigere Verfahren als Chamfer-Matching zu verwenden.

3.3.2 Berechnen möglicher Centroid-Votes

Die Erkennung und Lokalisierung von Objekten mit dem BFM geschieht mit Hilfe von BFs, die zum Zentrum des Objektes zeigen. Opelt u. a. [2006b] haben in ihrer Multi-Class-Variante diese CVs bereits beim Extrahieren der BFs gespeichert. Da es aber keine diskriminativen BFs geben kann, die bei verschiedenen Klassen an derselben Stelle auftreten, sollen in dieser Arbeit für jede zu trainierende Klasse unterschiedliche CVs zugelassen werden. Es ist also notwendig, die CVs der weak-classifier im Lernschritt zu wählen. Dazu müssen zuerst mögliche CVs für die extrahierten BFs berechnet werden.

Die zuvor berechneten Chamfer-Maps werden geladen und die lokalen Minima[4] gesucht. Dieser Schritt kann wieder parallelisiert werden. Es kann vorkommen, dass mehrere Minima sehr nah beieinander liegen. Deshalb werden sie vor der weiteren Verwendung mit Mean-Shift geclustert und jene Minima, die außerhalb der Bounding-Box liegen, verworfen. Von den verbleibenden Positionen werden die CVs berechnet und gespeichert. Es

[4]Lokale Minima sind dadurch gekennzeichnet, dass die Werte aller Nachbarpixel größer oder gleich groß sind.

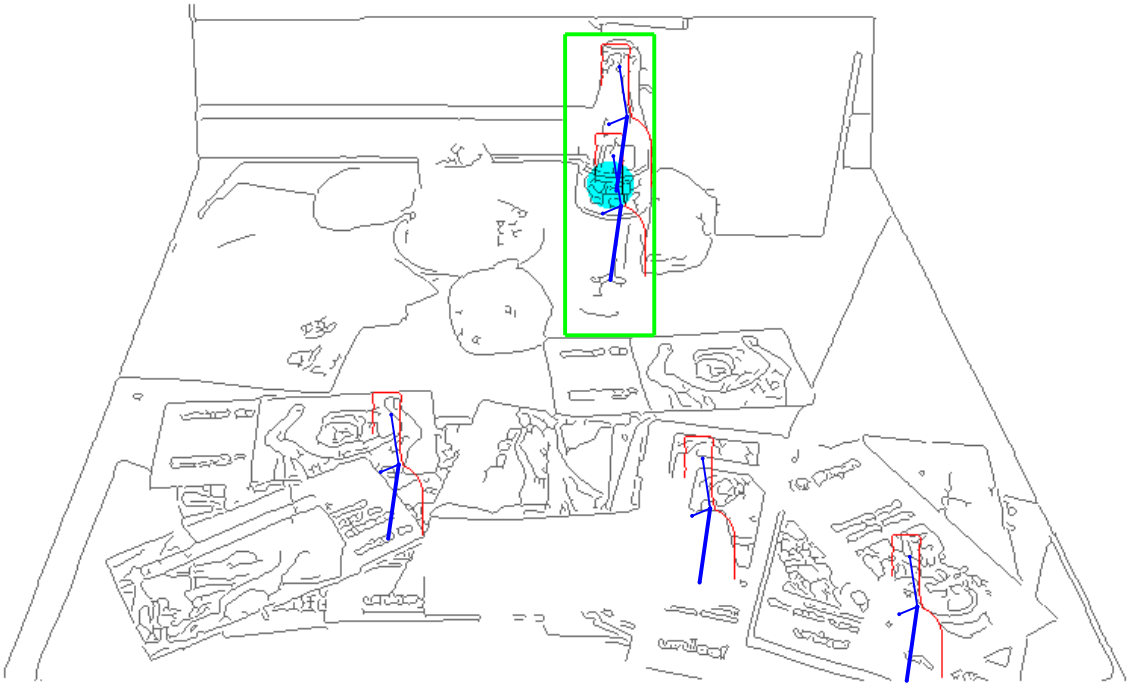


Abbildung 3.7: Ein Validierungs-Bild der Flaschen-Datenbank mit der Bounding-Box (grün), den fünf besten Treffern eines BFs (rot) mit drei möglichen CVs (blau). Der CV, der in die Zentrums-Unsicherheit mit dem Radius $r_c = 15$ (hellblau) des Objektes trifft, ist dicker gezeichnet.

entsteht so eine Anzahl möglicher CVs pro BF und Validierungs-Bild. Da ein, für eine Klasse diskriminatives BF, logischerweise sehr oft an derselben Stelle vorkommen soll, werden die berechneten CVs eines BFs nochmals geclustert. Die Auswahl des am besten geeigneten CVs geschieht im Lernschritt und ist in Kapitel 3.4 beschrieben.

3.3.3 Datenstruktur für das Boosting

Die Software wurde komplett mit Matlab geschrieben. Da Matlab eine Skript-basierte Programmiersprache ist, ist die Laufzeit von vornherein langsamer als von Programmen, die in Maschinencode vorliegen. Die Stärke von Matlab sind Operationen mit Matrizen, diese können sehr schnell durchgeführt werden. Deshalb werden die für das Boosting benötigten Daten vorher in Matrizen organisiert.

Zum einen werden die extrahierten CVs in einer dreidimensionalen Matrix $\boldsymbol{M}_{CV}$ gespeichert. Die Dimensionen stehen für die BFs, die Klassen und die Validierungs-Bilder. Des Weiteren wird eine zweite dreidimensionale Matrix $\boldsymbol{M}_{match}$ für jede Klasse aufgebaut. Diese enthält die Information, welcher CV das wahre Zentrum des Objektes in den Validierungs-Bildern getroffen hat. Ein Treffer liegt dann vor, wenn der CV in den Umkreis mit dem Radius r_c des Zentrums trifft (Abbildung 3.7). Die Treffer werden wieder für jeden CV der BFs in allen Validierungs-Bilder berechnet. Die Matrizen $\boldsymbol{M}_{CV}$ und $\boldsymbol{M}_{match}$ werden im nächsten Abschnitt zur effizienten Suche nach weak-classifier benötigt.

3.4 Effiziente Auswahl von weak-classifiers

Die Auswahl der weak-classifier ist der zeitaufwendigste Schritt, denn wie in Kapitel 3.3 bereits erwähnt wurde, steigt die Anzahl möglicher Kombinationen von BFs mit dem Binomialkoeffizienten. Um in jeder Boosting-Runde das beste Paar auszuwählen, muss laut Schapire u. Singer [1998] für jedes Paar die Kostenfunktion

$$Z_n = \sum_{i=1}^{m} D_t(i) \cdot e^{-y_i h_n(x_i)}. \tag{3.2}$$

berechnet werden. $D_t(i)$ ist die Gewichtung des Validierungs-Bildes i in der Boosting-Runde t, y_i ist der Label des Validierungs-Bildes x_i und h_n ist der weak-classifier, für den Z_n berechnet wird. In dieser Arbeit wurde diese Kostenfunktion mit der Bewertung der geometrischen Beziehung (3.1) zwischen den BFs auf

$$Z_n = \sum_{i=1}^{m} D_t(i) \cdot e^{-y_i h_n(x_i)} \cdot \frac{1}{\chi} \tag{3.3}$$

$$\chi = \frac{1}{K} \cdot \sum_{k=1}^{K} [1 - \alpha \cdot (1 - \cdot\delta_k)] \tag{3.4}$$

erweitert. Da jede Klasse andere CVs besitzt, ist auch δ für jede Klasse anders. In der Kostenfunktion wird deshalb der Mittelwert über alle Klassen verwendet. Mit dem Faktor α kann der Einfluss der Gewichtung verändert werden. $\alpha = 0$ bedeutet, die Gewichtung wird nicht berücksichtigt und mit $\alpha = 1$ wird sie voll berücksichtigt.

Um die gewünschte Performance zu erreichen, muss eine Vereinfachung angenommen werden: Es werden nicht alle möglichen CVs pro BF getestet, sondern immer nur jener, der in den wichtigsten Validierungs-Bildern gültig ist. Würde diese Vereinfachung nicht gemacht werden, müsste jedes Paar mit all den unterschiedlichen CVs und auch die einzelnen BFs selbst mit ihren CVs getestet werden, d.h. die Rechenzeit wäre nicht mehr tragbar. Um den besten CV je Klasse auszuwählen, wird die dreidimensionale Matrix $\boldsymbol{M}_{match}$ zuerst mit einer, um eine dritte Dimension erweiterten Gewichtsmatrix $\boldsymbol{D}$, d.h. die zweidimensionale Matrix $\boldsymbol{D}$ wird entsprechend der Anzahl CVs erweitert, multipliziert. Das Ergebnis wird anschließend entlang der zweiten Dimension, also entlang der Validierungs-Bilder, aufsummiert. Die zwei Schritte sind in Abbildung 3.8 dargestellt. Es entsteht für jede Klasse eine $n \times j$ Matrix (n ist die Anzahl BFs und j ist die der möglichen CVs), die in den Spalten die aufsummierten Gewichte der mit dem entsprechenden CV richtig klassifizierten Validierungs-Bilder beinhaltet. Diese Matrix wird entlang der CV-Dimension sortiert, d.h. jene CVs, die die wichtigsten Validierungs-Bilder richtig Klassifizieren, stehen nun an erster Stelle, alle anderen werden verworfen.

Der Term $-y_i h_n(x_i)$ in der Formel (3.3) ergibt -1, wenn der weak-classifier h_n das Validierungs-Bild x_i richtig klassifiziert und 1 wenn nicht, d.h. es muss überprüft werden, bei welcher Kombination von BFs beide in das wahre Objektzentrum feuern. Dazu wird zuerst eine $l \times 2$ Matrix $\boldsymbol{V}$, mit

$$l = \begin{pmatrix} n \\ 2 \end{pmatrix}$$

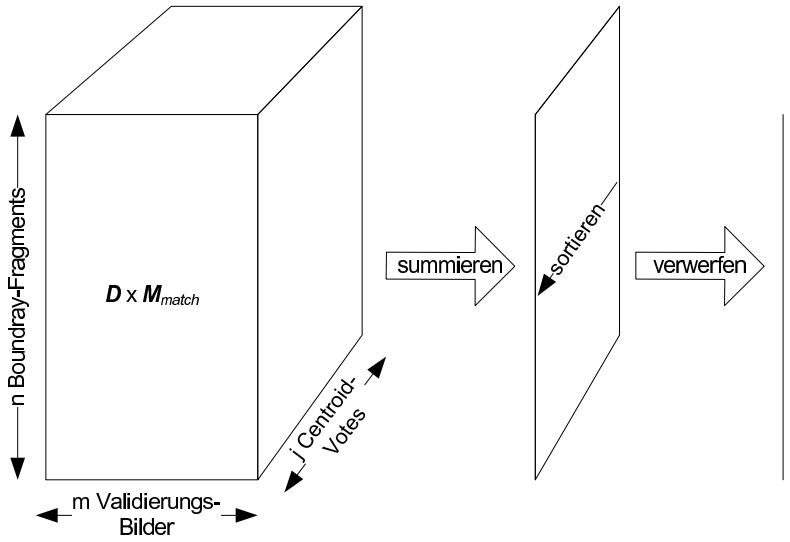


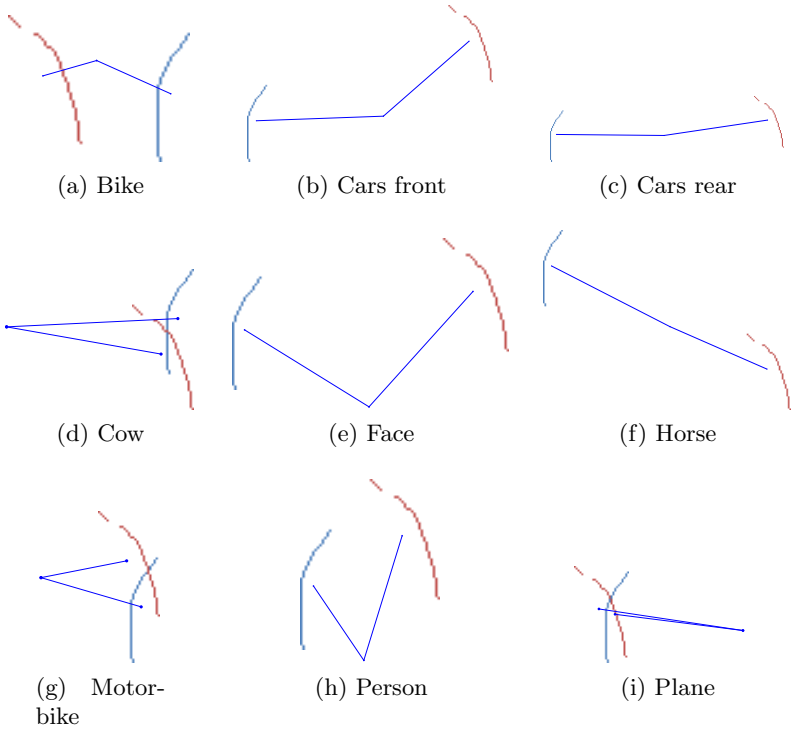
Abbildung 3.8: Auswählen des besten CVs. Die dreidimensionale Matrix $\boldsymbol{M}_{match}$ wird zuerst mit der Gewichtsmatrix $\boldsymbol{D}$ multipliziert und entlang der Validierungs-Bilder-Dimension aufsummiert. Die so entstandene Matrix enthält große Werte für jene CVs, welche oft in den Validierungs-Bildern feuern. Schlussendlich wird nur jener CV behalten, welcher den größten Wert hat, also jener, der, in Abhängigkeit der Gewichtsverteilung, in den meisten Validierungs-Bildern feuert.

aufgebaut und in diesen jede mögliche Kombination eingetragen, also

$$\boldsymbol{V} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ \vdots & \\ n-1 & n \end{bmatrix}.$$

Als nächstes wird eine weitere Matrix der Größe $l \times m$ (m ist die Anzahl der Validierungs-Bilder) initialisiert. Diese wird spaltenweise, das bedeutet entlang der Validierungs-Bilder, abgearbeitet. Mit Hilfe der Matrix $\boldsymbol{V}$ und der Matrix $\boldsymbol{M}_{match}$ wird nun in einem Schritt für alle Kombinationen überprüft, ob zuerst das erste und dann das zweite BF mit den zuvor ausgewählten CVs das Objektzentrum in den positiven Validierungs-Bildern treffen. Ist das der Fall, wird e^{-1} in die Matrix eingetragen, wenn das Objektzentrum nicht von beiden BFs getroffen wird, e^{+1}. Diese Matrix wird wieder mit den Gewichten $\boldsymbol{D}$ multipliziert und entlang der Validierungs-Bilder aufsummiert. Die Berechnung von χ kann ebenfalls für alle Kombinationen pro Validierungs-Bild gleichzeitig durchgeführt werden und durch Division erhält man laut Formel (3.3) Z. Ein Beispiel eines möglichen weak-classifiers, der die Funktion Z für alle Klassen minimiert, ist in Abbildung 3.9 dargestellt. Die Gewichtung δ wird beim Berechnen von Z gemittelt (Formel (3.4)), daher kann es passieren, dass, wie in Abbildung 3.9i, sich CVs trotzdem ähnlich sind. Auch wenn die Gewichte der Validierungs-Bilder so verteilt sind, dass die Kostenfunktion Z trotz eines weak-classifiers mit einer schlechten durchschnittlichen geometrischen Beziehung minimiert wird, wird dieser dem Modell hinzugefügt.

Da mit Hilfe des beschriebenen Algorithmus alle Kombinationen in einem Schritt auf einem Validierungs-Bild getestet werden können, wird die Laufzeit erheblich reduziert. Es ist nun möglich, alle Kombinationen von 4000 BFs, also ca. 8 Millionen, auf 200 Validierungs-



(a) Bike (b) Cars front (c) Cars rear

(d) Cow (e) Face (f) Horse

(g) Motorbike (h) Person (i) Plane

Abbildung 3.9: Die verschiedenen CVs (blau) eines weak-classifiers für unterschiedliche Klassen. Die BFs sind für jede Klasse dieselben. Um die Übersicht zu bewahren, ist die Darstellung für jede Klasse anders skaliert. Für die Klasse Bottle wurden für diese Kombination der BFs keine gültigen CVs gefunden.

Bilder in ungefähr 5 Minuten mit einem aktuellen Standard-PC[5] zu testen.

3.5 Multi-Class-Boosting

Das verwendete Lernverfahren wurde bereits im Algorithmus 2 auf der Seite 10, vorgestellt. In T Runden wird durch systematisches Hinzufügen von weak-classifiers ein Klassifikator zusammengestellt. Der Vorgang ist in Abbildung 3.10 dargestellt. Die Auswahl des besten weak-classifiers, in Abhängigkeit der Verteilung der Trainingsgewichte D, ist in Kapitel 3.4 beschrieben. Nachdem ein weak-classifier gefunden wurde, der die Kostenfunktion (3.3) minimiert, muss noch geprüft werden, für welche Klasse der gefundene weak-classifier gültig ist, also welche CVs dem weak-classifier tatsächlich zugewiesen werden sollen. Weak-classifiers, die Z minimieren, klassifizieren jene Bilder richtig, die hohe Trainingsgewichte besitzen, d.h. es kann vorkommen dass Bilder anderer Klassen nicht erkannt werden. Wie bereits in Kapitel 2.2 auf der Seite 7 beschrieben, müssen weak-classifier im binären Fall besser als zufälliges Raten sein, damit der Fehler des Klassifikators abnimmt. Betrachten wir die erste Boosting-Runde, dann sind die Trainingsgewichte aller Bilder gleich verteilt, nämlich $D_1 = 1/m$, m ist die Anzahl Validierungs-Bilder. Setzt man die Überlegung fort, muss ein weak-classifier im Multi-Class-Fall mehr als die Hälfte der Validierungs-Bilder einer Klasse erkennen können, um für diese Klasse gültig zu sein. Ein weak-classifier kann

[5] Core i7 860 (4x2.8GHz) und 8GB Arbeitsspeicher

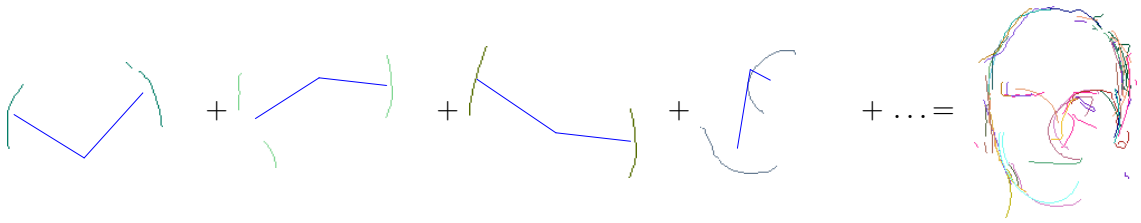


Abbildung 3.10: Ein Beispiel für die Zusammensetzung eines Klassifikators für die Klasse Face. Die einzelnen weak-classifier zusammen, ergeben den Klassifikator. Um die Übersicht zu bewahren, wurden die weak-classifier unterschiedlich skaliert.

eine Klasse also klassifizieren, wenn gilt

$$\sum_{i=1}^{m_k} D_1(i) \cdot \begin{cases} 1 & \text{if } h_1(x_i) = y_i \\ 0 & \text{otherwhise} \end{cases} \geq \frac{m_k}{m} \cdot \frac{1}{2}.$$

m_k ist dabei die Anzahl Validierungs-Bilder in der Klasse k. Nach jeder Boosting-Runde werden die Validierungs-Bilder neu gewichtet. Jene die richtig klassifiziert wurden, werden nach unten und jene die falsch klassifiziert wurden, werden nach oben gewichtet. Behalten wir die Überlegung der ersten Runde für die nächsten bei, ergibt sich folgende Vorschrift, zum Bestimmen, wann einem Klassifikator die CVs einer Klasse zugewiesen werden sollen:

$$\sum_{i=1}^{m_k} D_t(i) \cdot \begin{cases} 1 & \text{if } h_t(x_i) = y_i \\ 0 & \text{otherwhise} \end{cases} \geq \frac{m_k}{m} \cdot \frac{1}{2}. \tag{3.5}$$

Was bedeutet dies nun für nicht gleich verteilte Trainingsgewichte? Wenn die Validierungs-Bilder einer Klasse bereits sehr oft erkannt wurden, sind die Gewichte sehr klein. Kommt nun ein neuer weak-classifier, der wieder alle Bilder erkennen kann, ist die Summe über die Gewichte ab einer bestimmten Boosting-Runde t nicht mehr größer als $m_k/m \cdot 0.5$ und somit werden dem weak-classifier für diese Klasse keine CVs mehr zugewiesen. So wird verhindert, dass leichter zu lernende Klassen übertrainiert werden. Andererseits sind die Gewichte von schwierigen Validierungs-Bildern sehr hoch und deshalb wird der Schwellwert für solche Klassen bereits überschritten, wenn nur sehr wenig Bilder mit einem weak-classifier klassifiziert werden. Es werden also einfachere Klassen nicht bevorzugt und schwierigere nicht vernachlässigt, vielmehr wird versucht, die Anzahl weak-classifiers an die Schwierigkeit der Validierungs-Bilder anzupassen.

Nachdem die CVs der entsprechenden Klassen dem weak-classifier zugewiesen und die Trainingsgewichte entsprechend der Formel (2.5) auf der Seite 10 aktualisiert wurden, wird der gefundene weak-classifier h_t mit seinem Gewicht α_t dem Klassifikator H hinzugefügt. Eine mögliche Entwicklung der Gewichte ist in Abbildung 3.11 dargestellt. Einige schwierige Bilder können in diesem Beispiel mit keinem der gefundenen weak-classifier erkannt werden, deshalb steigen ihre Gewichte immer weiter an. Das passiert so lange, bis die Gewichte so groß sind, dass ein weak-classifier gefunden wird, der diese Bilder klassifizieren kann.

Normalerweise müssen für Lernalgorithmen wie Boosting, positive und negative Trainingsdaten zur Verfügung stehen. Wir verwenden aber nur positive Validierungs-Bilder denn diese beinhalten nicht nur das Objekt selbst, sondern auch genügend Hintergrund. BFs bzw. CVs die nur im Hintergrund feuern, wurden bereits im Vorverabreitungsschritt

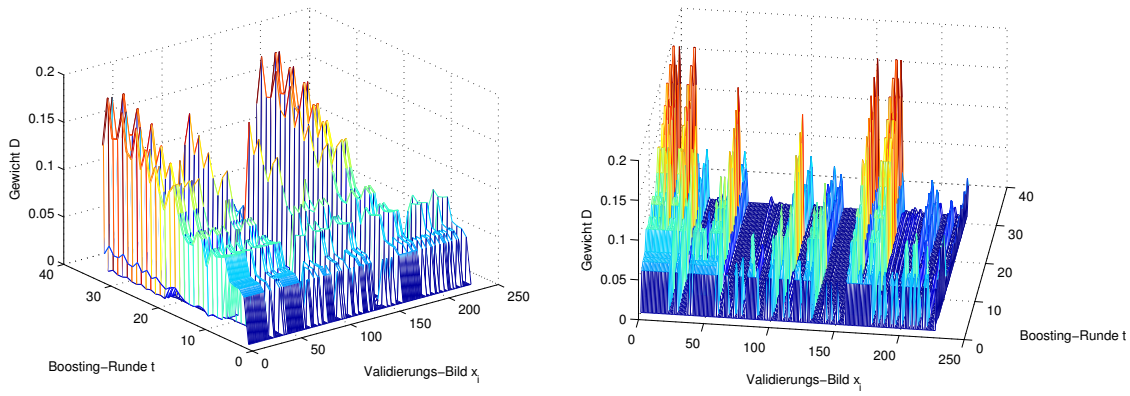


Abbildung 3.11: Eine mögliche Entwicklung der Gewichte der Validierungs-Bilder über die Boosting-Runden. Die zwei Darstellungen sind dieselben, nur von verschiedenen Ansichten betrachtet. Es ist gut zu erkennen, dass einige Bilder vom Algorithmus nicht erkannt werden können und ihre Gewichte deshalb ansteigen. Andere Bilder sind sehr einfach, d.h. ihre Gewichte sind von Anfang an niedrig.

(Kapitel 3.3.2 auf der Seite 20) verworfen, d.h. im Boosting-Algorithmus stehen hauptsächlich BFs zur Verfügung, die diskriminativ für die zu trainierenden Klassen sind. Die Klassifikatoren werden also darauf trainiert, Klassen voneinander zu unterscheiden und nicht Klassen vom Hintergrund.

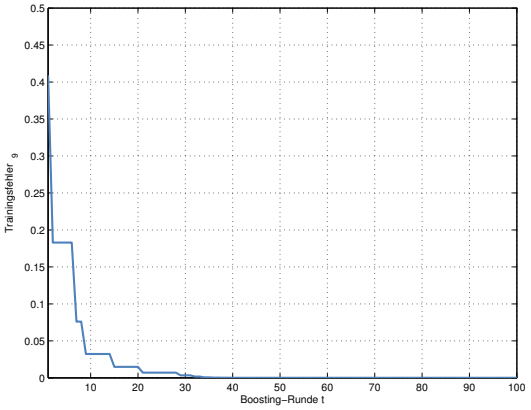
Um die Performance des Verfahrens zu analysieren, kann ein Trainingsfehler für die einzelnen Klassen berechnet werden. Freund u. Schapire [1997] geben in ihrem Paper die obere Grenze des Trainingsfehlers für AdaBoost mit $\epsilon \leq exp(-2\sum_{t=1}^{T}\gamma_t^2)$ an. Der Trainingsfehler der einzelnen Klassen ist demnach

$$\epsilon_k \leq e^{-2\sum_{t=1}^{T}\gamma_{t,k}^2}, \tag{3.6}$$

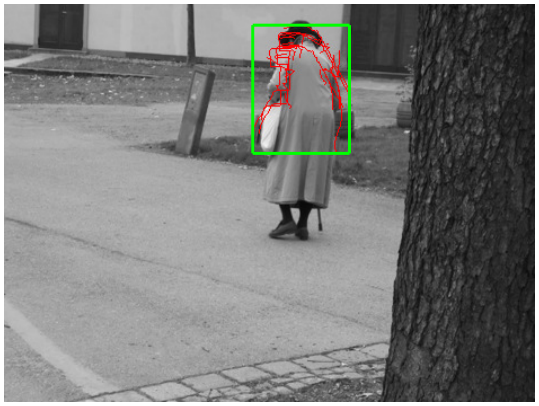
wobei $\gamma_{t,k}$ mit $1/2 - \epsilon_{t,k}$ berechnet wird. $\epsilon_{t,k}$ ist der Fehler des weak-classifiers in der Boosting-Runde t auf der Klasse k. Aus der Formel (3.6) ist ersichtlich, dass der Trainingsfehler einer Klasse exponentiell abnimmt, solange weak-classifiers gefunden werden, deren Fehler in der Klasse kleiner als $1/2$ sind. Bei der Berechnung des Trainingsfehlers (3.6) ist darauf zu achten, dass nicht in jedem Boosting-Schritt weak-classifiers gefunden werden, die jede Klasse erkennen können, d.h. die Formel darf nur für gültige t berechnet werden.

Der Trainingsfehler darf allerdings nicht mit dem Detektionsfehler verwechselt werden. Zum Berechnen des Trainingsfehlers ist es nur erforderlich, dass die weak-classifier das Zentrum der Objekte in den Validierungs-Bildern treffen, während für die Berechnung der Detektionsrate die Bounding-Box (siehe Kapitel 5.2 auf der Seite 33) herangezogen wird. Es kann also vorkommen, dass, wie in Abbildung 3.12 dargestellt ist, Modelle mit einem sehr kleinen Trainingsfehler trainiert werden, diese aber nur einen Teil des Objektes beschreiben, wie in diesem Beispiel den Oberkörper einer Person. Solche Modelle haben aber immer eine sehr schlechte Detektionsrate, denn der fehlende Teil des Objektes wird nicht in die Bounding-Box mit eingeschlossen (Abbildung 3.12c), was zu einer falschen Detektion führt.

(a) Die weak-classifier eines Personen-Modells. BFs die zum selben weak-classifier gehören, besitzen die gleiche Farbe.



(b) Der Trainingsfehler des Modells in (a) aufgetragen über die Boosting-Runden. Der stufenförmige Verlauf resultiert daher, dass nicht alle weak-classifier die Klasse Persons erkennen können.



(c) Die Bounding-Box (grün) und die zugehörigen BFs (rot) einer Detektion mit dem Modell aus (a).

Abbildung 3.12: Das Personenmodell in (a) kann trotz eines sehr geringen Trainingsfehlers (b) die Objekte nicht erkennen. Grund dafür ist, dass nur ein Teil des Modelles, hier der Oberkörper, gelernt wurde, und deshalb die Bounding-Box nicht das ganze Objekt umfasst (c).

Kapitel 4

Klassifizieren neuer Bilder

In den vorherigen Kapiteln wurde erklärt, wie ein Klassifikator für mehrere Klassen trainiert werden kann. In diesem Teil wird nun beschrieben, wie ein gelerntes Modell neue Bilder klassifizieren kann, also wie Instanzen der Objekte erkennt werden können. In Abbildung 4.1 ist der Detektions-Vorgang, vom Berechnen der Kanten über das Matchen der BFs des gelernten Klassifikators bis zur Bounding-Box der Detektion, für eine Klasse grob dargestellt und wird im Folgenden detailliert beschrieben. Auch hier dienen die gezeigten Beispiele nur der Erklärung, genauere Experimente folgen in Kapitel 5 ab der Seite 32.

4.1 Ablauf

Die Detektion unterscheidet sich kaum von der aus [Opelt u. a., 2006a]. Vom Testbild x werden zuerst die canny-edges berechnet. Danach werden die BFs des Klassifikators H im binären Kantenbild mittels Chamfer-Matching (wieder mit acht Orientierungs-Ebenen) gesucht und die lokalen Minima der Distanz-Map weiter verwendet. Wie bereits in Kapitel 3.3.2 auf der Seite 20 erwähnt, werden die Minima vor der weiteren Verwendung mit Mean-Shift geclustert, um zu vermeiden, dass sehr nahe beieinander liegende Minima als eigenständige Matches gewertet werden. Es soll auch möglich sein, mehrere Instanzen desselben Objektes erkennen zu können. Deshalb wird nach dem Clustern nicht nur der beste Match eines BFs, also jener mit der geringsten Chamfer-Distanz, verwendet, sondern die n besten, die über einem Threshold th_{BF} liegen. Die Anzahl n hängt unter anderem von der Anzahl vorkommender Objekte, der Bildgröße und von der Struktur des Hintergrundes ab. Den Threshold th_{BF} haben Opelt u. a. [2006a] im Lernschritt ausgewählt, allerdings war dieser immer so hoch, dass er praktisch keinen Einfluss hatte. Des Weiteren funktioniert das hier vorgestellte Lernverfahren bzw. die Auswahl der weak-classifiers anders als in [Opelt u. a., 2006a], deshalb wird der Parameter th_{BF} hier fix eingestellt.

Von den gefundenen Positionen aller BFs ausgehend, werden ihre CVs in einen eigenen Voting-Space je Klasse eingetragen, d.h. ein BF, welches an der Position $\boldsymbol{u}$ gefunden wird, produziert einen Eintrag in Form einer Kreisscheibe mit dem Radius r_{CV} im Voting-Space Ω_k an der Stelle $\boldsymbol{u} + \boldsymbol{\zeta}_k$ für $k = 1 \dots K$, wie in Abbildung 4.2 gezeigt wird. $\boldsymbol{\zeta}_k$ entspricht dabei dem CV-Vektor der Klasse k. Enthält ein BF kein CV für eine Klasse, wird nichts eingetragen. Da ein weak-classifier nur feuert, wenn sich die Unsicherheiten der CVs seiner BFs überlappen, muss überprüft werden, an welchen Stellen das der Fall ist. Der endgültige Eintrag in den Voting-Space Ω_k ist dann nicht die Kreisscheibe, sondern das Gewicht des

weak-classifiers α_t (Formel (2.4) auf der Seite 10) an der Stelle

$$\boldsymbol{u}_{BF1} + \boldsymbol{\zeta}_{BF1,k} + \frac{\boldsymbol{u}_{BF2} + \boldsymbol{\zeta}_{BF2,k} - \boldsymbol{u}_{BF1} - \boldsymbol{\zeta}_{BF1,k}}{2},$$

falls $\left\|\boldsymbol{u}_{BF2} + \boldsymbol{\zeta}_{BF2,k} - \boldsymbol{u}_{BF1} - \boldsymbol{\zeta}_{BF1,k}\right\|_2 \leq 2 \cdot r_{CV}$ ist. In Abbildung 4.3 ist diese Stelle mit einem grünen Kreuz markiert. Nachdem alle weak-classifier überprüft wurden, entsteht für jede Klasse ein Voting-Space, wie in Abbildung 4.1c zu sehen ist.

Anschließend wird der Voting-Space mit Mean-Shift geclustert (als Radius für den Mean-Shift-Kernel wird wieder r_{CV} verwendet) und um jene BFs, die innerhalb der Unsicherheit um das Cluster-Zentrum liegen, die Bounding-Box gezeichnet. Die Konfidenz einer Detektion entspricht der Summe aller Gewichte α_t innerhalb des Unsicherheitsradius r_{CV} um das Cluster-Zentrum, also

$$conf = \sum_{\Gamma} \alpha. \tag{4.1}$$

Γ ist die Fläche eines Kreises mit dem Radius r_{CV} und dem Mittelpunkt im Cluster-Zentrum.

4.2 Behandlung von Skalierung und Rotation

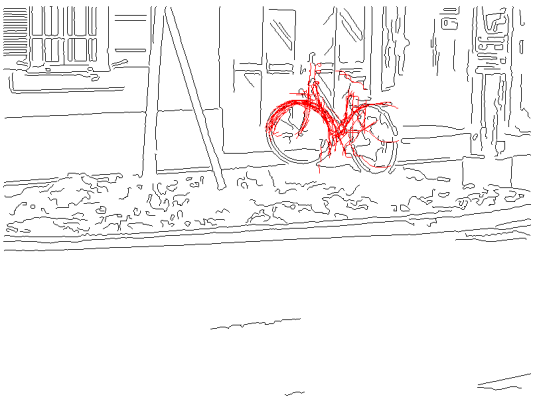
In natürlichen Szenen treten Objekte immer in verschiedenen Größen (Scale) und Orientierungen (Rotation) auf. Betrachtet man z.B. ein Bild von weidenden Kühen, sind jene Kühe, die näher beim Betrachter stehen, größer, als jene, die weiter hinten sind. Es ist deshalb sinnvoll, wenn Algorithmen zur Objekterkennung unabhängig vom Scale und Rotation der Objekte funktionieren. Die Möglichkeiten der Scale- und Rotations-Behandlung wurden aber in dieser Arbeit nicht betrachtet und sind nur der Vollständigkeit halber erwähnt.

Der verwendete Deskriptor[1] ist von sich aus nicht invariant gegen Scale und Rotation, denn die BFs stammen von Objekten einer bestimmten Größe und Rotation und feuern auch nur auf annähernd gleichen Objekten. Der einfachste Weg, um Objekte in verschiedenen Größen und Rotationen dennoch zu erkennen, ist die Testbilder in Schritten zu skalieren (vgl. Bildpyramide [Adelson u. a., 1984]) und zu rotieren und diese Varianten mit dem Klassifikator zu testen. Diese Methode haben auch Opelt u. a. [2006a] verwendet. Eine andere Möglichkeit ist, nicht die Testbilder zu ändern, sondern die BFs. Bei den Experimenten mit der Reimplementierung hat sich gezeigt, dass bessere Resultate erreicht werden, wenn die Testbilder, und nicht die BFs, skaliert werden. Es ist allerdings klar, dass der Rechenaufwand bei beiden Methoden mit der Anzahl Skalierungen, bzw. Rotationen zunimmt.

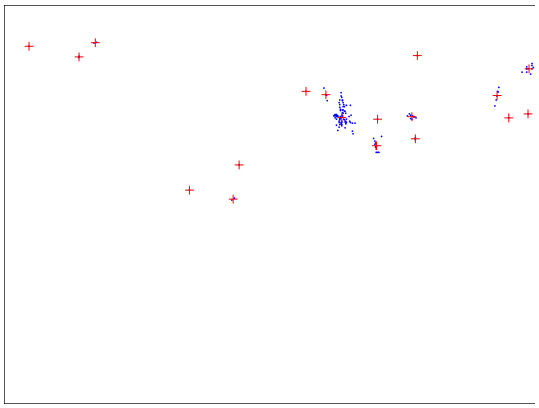
Des Weiteren ist die Entscheidung, welcher Scale nun der Richtig ist, nicht ganz trivial. In Kapitel 4.1 wurde beschrieben, dass mit Mean-Shift und einem fixen Radius r_{CV} Detektionen gesucht werden. Bei mehreren Scales muss der Radius ebenfalls skaliert werden. Eine mögliche Methode, um den Radius des Mean-Shift-Kerns anzupassen, haben Leibe u. Schiele [2004] vorgestellt. Da die Behandlung von Scale und Rotation aber nicht Teil dieser Arbeit ist, wird darauf nicht näher eingegangen.

[1] Ein Paar von BFs.

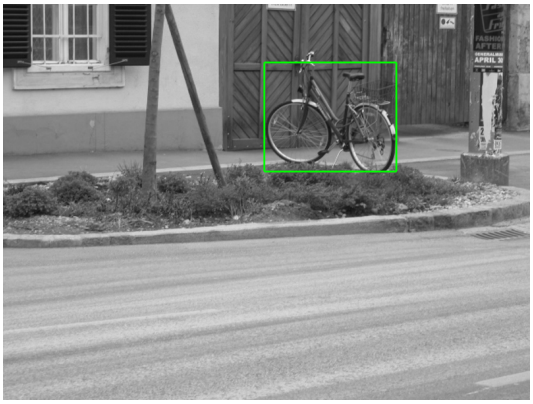
(a) Testbild x



(b) Kantenbild mit den BFs (rot), welche für eine Detektion der Bike-Klasse ($k = 1$) feuern



(c) Voting-Space Ω_1 mit den Votes der weak-classifier (blau) und den Cluster-Zentren (rot)



(d) Ausgabebild mit der Bounding-Box

Abbildung 4.1: Die Schritte der Detektion für eine Klasse (hier anhand der Klasse Bike). Ausgehend vom Testbild x (a) wird das Kantenbild berechnet und die BFs des Klassifikators mittels Chamfer-Matching gematcht (b). Überlappen sich die CVs der BFs eines weak-classifiers, wird die Position der Überlappung in den Voting-Space Ω_k der entsprechenden Klasse eingetragen (blaue Punkte in (c)). Nachdem Detektionen mittels Mean-Shift-Clustering im Voting-Space gefunden wurden (rote Kreuze in (c)), kann die Bounding-Box um jene BFs, die für die Detektion feuern, gezeichnet werden (d).

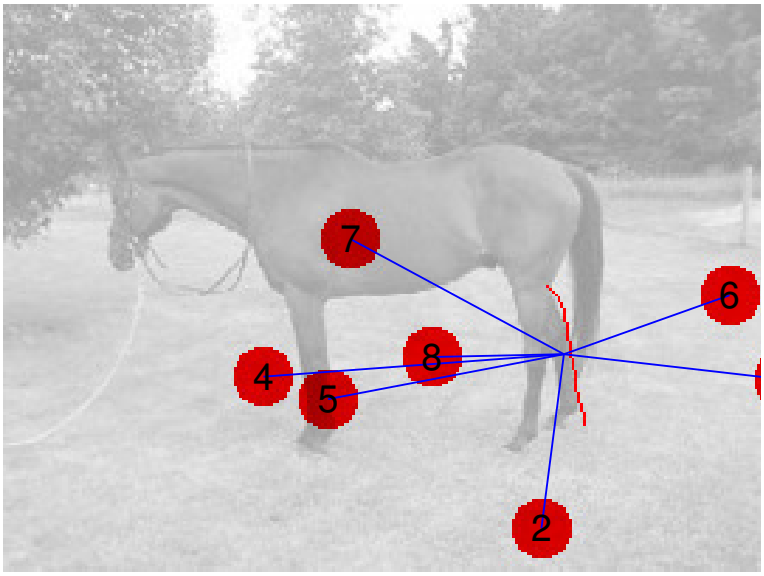


Abbildung 4.2: Ein BF (rot) eines weak-classifiers feuert mit den zugehörigen CVs (blau) in den Voting-Space Ω_k. Zur Orientierung ist das Testbild dem Voting-Space hinterlegt. Die Unsicherheit der CVs mit dem Radius $r_{CV} = 10$ Pixel ist als rote Kreisscheibe dargestellt. Der nach rechts zeigende CV ist zu lang und feuert deshalb nicht mehr vollständig in den Voting-Space. Die Nummern in den Kreisscheiben entsprechen der Klasse des jeweiligen CVs (2: Bottle; 4: Cars rear; 5: Cow; 6: Face; 7: Horse; 8: Motorbike).

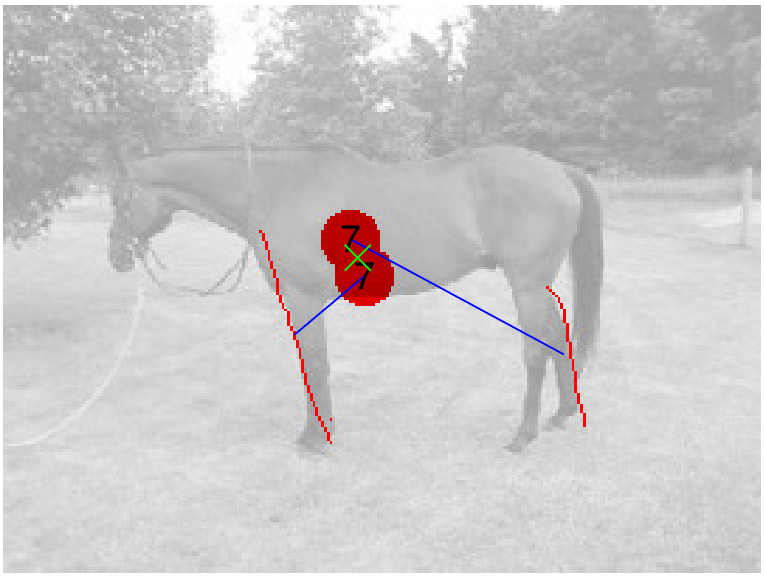


Abbildung 4.3: Ein weak-classifier mit den zwei BFs (rot) und ihren CVs (blau) feuert mit dem Gewicht α_t an der mit dem grünen Kreuz markierten Stelle in den Voting-Space Ω_7. Das Testbild ist wieder zur Orientierung hinterlegt, die Unsicherheit der CVs mit dem Radius $r_{CV} = 10$ Pixel ist als rote Kreisscheibe dargestellt. Die Nummer in den Kreisscheiben entspricht der Klasse (7: Horse).

Kapitel 5

Experimente und Resultate

In den vorangegangenen Kapiteln wurden die Unterschiede dieser Methode gegenüber der Reimplementierung des BFMs beschrieben. Diese liegen vor allem in einem Multi-Class-Boosting-Algorithmus und einer anderen Wahl geeigneter weak-classifier. Es ist daher naheliegend, diese Methode einem direkten Vergleich mit der Reimplementierung zu unterziehen. Der Vorteil in diesem Vergleich liegt darin, dass zum Trainieren und Testen exakt dieselben Bilder und Parameter verwendet werden können.

Ein Vergleich mit den Arbeiten, die in Kapitel 2.3 auf der Seite 10 beschrieben wurden, ist nicht sinnvoll. In dieser Arbeit geht es nicht primär darum, die Erkennungsleistung des BFMs zu steigern, sondern zu testen, ob das BFM so erweitert werden kann, dass auch mehrere Klassen gleichzeitig trainiert werden können.

Um die Experimente auch später nachvollziehen zu können, werden zuerst die verwendeten Daten und die Methode, die zum Auswerten verwendet wurde, beschrieben. Anschließend werden verschiedene Experimente durchgeführt, die erstens den Vergleich mit der Reimplementierung des BFMs und zweitens die Auswirkung der vorgestellten Änderungen zeigen sollen.

5.1 Die verwendeten Daten

Die Experimente wurden auf den Bildern der Graz02-Multi-Class[1] Datenbank durchgeführt. Diese beinhaltet Fotos der Objekte von 17 verschiedenen Klassen, die in Graz von Mitarbeitern der Vision-based Measurement Group (VMG) am Institut für Elektrische Meßtechnik und Meßsignalverarbeitung (EMT) zusammengestellt und teilweise von Hand annotiert wurden. Die Schwierigkeit beim Experimentieren mit solchen Bildern liegt an der nicht konstanten Beleuchtung, dem Schattenwurf der Objekte und v.a. an dem stark strukturierten und dominanten Hintergrund. Eine Zusammenfassung der verwendeten Klassen und Anzahl Bilder je Klasse ist in Tabelle 5.1 zu finden. Da in dieser Arbeit die negativen Validierungs-Bilder nicht verwendet wurden, sind sie auch in der Tabelle nicht aufgelistet.

Das Trainieren und Testen neuer Modelle liegt trotz signifikanten Verbesserungen noch immer in der Größenordnung von Tagen[2]. Um diesen Algorithmus gründlich zu testen, wurde die Anzahl der zu trainierenden Klassen auf Zehn (Bike, Bottle, Cars front, Cars

[1] Die Datenbank ist auf der DVD (Seite 63) zu finden, bzw. steht unter `http://www.emt.tugraz.at/~pinz/data/multiclass` zum Download zur Verfügung.

[2] Die Laufzeit des Forschungscodes von Opelt u.a. betrug Wochen.

Nr.	Klasse	Trainings-bilder	positive Validierungs-Bilder	Testbilder	Größe der Objekte in Pixel
1	Bike	45	45	53	200 w
2	Bottle	24	30	64	200 h
3	Cars front	17	17	16	200 w
4	Cars rear	50	50	166	240 w
5	Cow	20	25	65	200 w
6	Face	50	50	166	200 h
7	Horse	30	25	96	200 w
8	Motorbike	50	50	400	200 w
9	Person	19	20	18	200 h
10	Plane	50	50	400	200 w

Tabelle 5.1: Die verwendeten Klassen und Anzahl Bilder der Graz02-Multi-Class Datenbank. Die Größe der Objekte bezieht sich entweder auf die Breite (w) oder auf die Höhe (h) der Bounding-Box. Zum Trainieren der Modelle wurde nicht die komplette Anzahl der Trainings- und positiven Validierungsbilder verwendet, sondern jeweils maximal 20.

rear, Cow, Face, Horse, Motorbike, Person und Plane) beschränkt. Die Wahl der Klassen wurde so gestaltet, dass die Unterscheidbarkeit zwischen den Klassen (Cars front vs. Cars rear und Cow vs. Horse) und die Performance auf schwierigen Klassen (Bike und Person) getestet werden kann.

Um zu vermeiden, dass bestimmte Klassen im Lernschritt bevorzugt werden, wurden für jede Klasse gleich viele Bilder zum Trainieren und Validieren verwendet. Cars front ist mit 17 Bildern die Klasse mit den wenigsten. Da der Trainingsfehler mit der Anzahl Bilder sinkt [Opelt u. a., 2006a], wurden für alle Klassen 20 Trainings- und 20 positive Validierungsbilder verwendet, außer bei den Klassen Cars front nur 17 und bei Person nur 19. Um die Modelle zu testen, wurden immer gleich viel negative und positive Testbilder getestet, jedoch maximal 166, denn in der Datenbank sind nur 166 negative Testbilder vorhanden. Die tatsächlich verwendeten Bilder sind ab der Seite 50 im Anhang A aufgelistet.

Zum Trainieren und Testen von Modellen wurden die Bilder vorher so skaliert, dass die vorkommenden Objekte der Größe in Spalte 6 der Tabelle 5.1 entsprechen. Die Größe der Bilder selbst liegt nach dem Skalieren zwischen 204×74 und 1543×1157 Pixel, d.h. es sind Bilder vorhanden, auf denen das Objekt sehr dominant ist und solche, in denen der Hintergrund dominiert. Einige Beispiele aus dieser Datenbank sind ab Seite 51 in den Abbildungen A.1 bis A.3 dargestellt.

5.2 Auswertung der Detektionen

Die Auswertung geschieht mit Hilfe von Recall-Precision-Curves (RPCs). Diese Kurven zeigen den recall aufgetragen über 1-precision. Zur Ermittlung von richtigen bzw. falschen Detektionen wird der Detektions-Threshold kontinuierlich erhöht, d.h. der recall steigt und die precision sinkt. Eine Detektion ist richtig, wenn die geschätzte Bounding-Box B_P mehr als 50% Überlapp mit der richtigen Bounding-Box B_{GT} hat, also

$$\frac{area(B_P \cap B_{GT})}{area(B_P \cup B_{GT})} \geq 0.5.$$

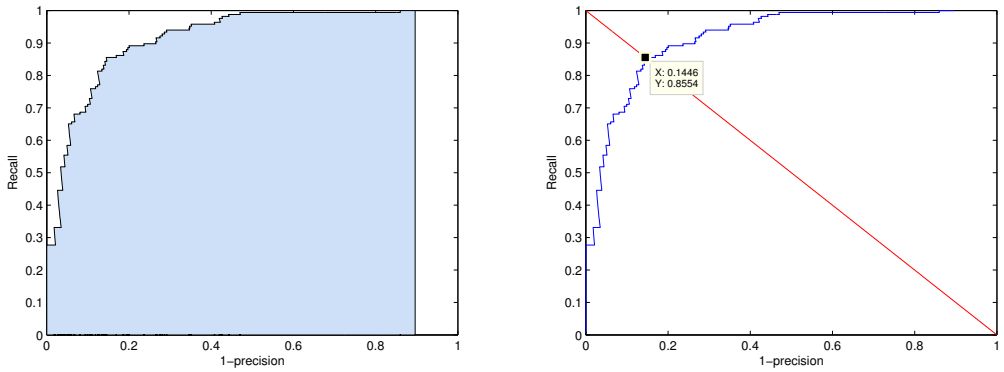


Abbildung 5.1: Eine mögliche Recall-Precision-Curve. Wird der Detektions-Threshold kontinuierlich erhöht, steigt der recall, aber die precision sinkt. Um die Detektionsrate in Prozent anzugeben kann entweder die Fläche unter der Kurve (links mit 86.6%) oder jener Punkt, bei dem recall=precision ist (rechts bei 85.5%) herangezogen werden.

Dieses Kriterium zur Ermittlung von korrekten Detektionen wurde von Agarwal u. Roth [2002] zum ersten Mal verwendet und ist heute weit verbreitet. Der recall berechnet sich mit

$$\text{recall} = \frac{\text{Anzahl richtiger Detektionen}}{\text{Anzahl Objekte}}$$

und die precision mit

$$\text{precision} = \frac{\text{Anzahl richtiger Detektionen}}{\text{Anzahl aller Detektionen}},$$

bzw. in der RPC verwendete 1-precision mit

$$\text{1-precision} = \frac{\text{Anzahl falscher Detektionen}}{\text{Anzahl aller Detektionen}}.$$

Um die Performance des Algorithmus in Prozent ausdrücken zu können, gibt es zwei Möglichkeiten: Zum einen kann die Fläche unter der RPC als Detektionsrate verwendet werden und zum anderen jener Punkt, bei dem recall=precision ist. Dieser Punkt wird RPC-equal-Error Rate (RPCeER) genannt. In Abbildung 5.1 sind beide Methoden dargestellt. Da in [Opelt u. a., 2006a; Mikolajczyk u. a., 2006] die RPCeER als Maß verwendet wurde, wird diese auch in dieser Arbeit zur Ermittlung der Detektionsrate herangezogen.

5.3 Referenzexperiment mit dem Boundary-Fragment-Model

Mit der reimplementierten Version des originalen BFMs wurden für jede Klasse aus der Tabelle 5.1 neue Modelle trainiert. Diese werden als Referenz für den Vergleich mit dem neuen Algorithmus herangezogen und anhand dieser Modelle werden die Unterschiede diskutiert. Die Parameter zum Trainieren und Testen dieser Modelle wurden empirisch ermittelt und sind in der Tabelle B.2 auf der Seite 58 aufgelistet. Bei der Wahl der Parameter wurde darauf geachtet, dass alle Modelle mit den gleichen Parametern trainiert werden, denn für den Multi-Class-Ansatz gelten ebenfalls für alle Klassen dieselben Parameter.

Die Rechenzeit zum Trainieren und Testen der Modelle ist in Tabelle 5.2 dargestellt. Die Vorverarbeitung für alle zehn Modelle dauert ca. 88 Stunden. Die lange Laufzeit

Nr.	Klasse	Vorverarbeitung	Boosting	Summe	Detektieren
1	Bike	14:01	00:19	14:20	01:48
2	Bottle	06:15	00:15	06:30	01:18
3	Cars front	12:08	00:18	12:26	01:52
4	Cars rear	08:40	00:15	08:55	03:05
5	Cow	07:53	00:20	08:13	01:39
6	Face	08:04	00:17	08:21	02:08
7	Horse	06:20	00:19	06:39	01:34
8	Motorbike	06:20	00:20	06:40	03:00
9	Person	08:49	00:21	08:10	01:17
10	Plane	06:48	00:16	07:04	03:29
		85:18	03:00	88:18	21:17

Tabelle 5.2: Die benötigte Rechenzeit des Experiments 20100511 in hh:mm zum Trainieren und Testen der Modelle mit dem originalen BFM. Die Dauer zum Trainieren eines Modells ergibt sich aus Vorverarbeitung+Boosting. Die Gesamtdauer steht in der letzten Zeile. Die unterschiedliche Laufzeit bei der Vorverarbeitung resultiert aus der unterschiedlichen Größe der Bilder und beim Detektieren zusätzlich aus der unterschiedlichen Anzahl an Testbildern.

resultiert daher, dass alle 2000 ausgeschnittenen BFs auf den 20 positiven und 20 negativen Validierungs-Bilder gematcht werden müssen. Für das Boosting werden dann nur noch die besten 350 verwendet, d.h. es werden also nur

$$\begin{pmatrix} 350 \\ 2 \end{pmatrix} = 61075$$

Kombinationen getestet. Das Testen einer Kombination dauert demnach

$$\frac{18\ min}{61075\ komb} = 17.7 \frac{ms}{komb}.$$

Für Kombinationen aus zwei BFs steigt der Binomialkoeffizient quadratisch mit der Anzahl zur Verfügung stehender BFs, und somit auch die Rechenzeit, wie in Abbildung 5.2 dargestellt ist. Wenn anstelle von 350 BFs 4000 verwendet werden, würden ca. 39 Stunden zum Boosten von nur einem Modell benötigt.

Die Ergebnisse des Referenzexperiments mit der Reimplementierung des BFMs sind in Tabelle 5.3 zusammengefasst. Um die Unterscheidbarkeit der einzelnen Detektoren untereinander zu untersuchen, wurden die einzelnen Klassifikatoren auch auf den Testbildern aller anderen Klassen getestet. Damit die RPCeER für die einzelnen Klassifikatoren berechnet werden kann, wurden zusätzlich gleich viele Hintergrundbilder wie Testbilder in der jeweiligen Klasse vorhanden sind, getestet. Die resultierenden RPCeER sind in der Diagonale der Tabelle 5.3 eingetragen. Der dafür notwendige Detektions-Threshold ist in der Spalte “Th” zu finden. Da es in den Testbildern einer anderen Klasse keine richtigen Detektionen geben kann, kann für diese auch keine RPC berechnet werden. Deshalb wurden in den entsprechenden Feldern die Klassifikationsraten eines Klassifikators (Zeile) auf den Testbildern der jeweils anderen Klassen (Spalte) eingetragen. Ist die Konfidenz einer Detektion auf den Testbildern einer anderen Klasse über dem Detektions-Threshold des Klassifikators, und überlappen sich die Bounding-Boxen der Detektion und des Objektes um mehr als 50%, bedeutet dass, das der Klassifikator das Objekt einer anderen Klasse fälschlicherweise klassifiziert.

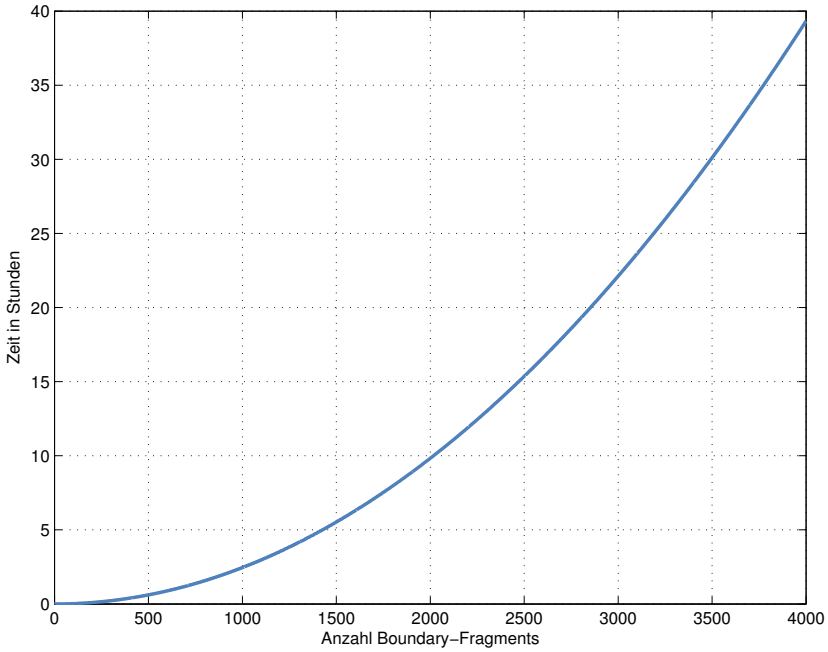


Abbildung 5.2: Die für das Boosting benötigte Zeit beim originalen BFM aufgetragen über die zur Verfügung stehenden BFs. Da immer zweier-Kombination von BFs getestet werden, resultiert ein quadratischer Anstieg mit der Anzahl BFs. Zum Trainieren der Referenz-Modelle wurden 350 BFs verwendet. Die dafür benötigte Zeit ist markiert und beträgt für ein Modell ca. 18 Minuten.

Nr.	**Kl.**	**Klassifikationsraten in %**										**Th**
		1	2	3	4	5	6	7	8	9	10	
1	Bike	75.9	1.6	6.3	0.0	4.6	19.9	6.3	72.3	4.3	1.8	9.737
2	Bottle	0.0	89.1	0.0	0.0	0.0	2.4	1.0	0.0	21.7	0.0	45.898
3	CarsF	33.3	1.6	87.5	14.5	44.6	20.5	35.4	64.5	0.0	13.9	23.857
4	CarsR	16.7	0.0	56.3	90.4	53.9	7.8	58.3	32.5	0.0	15.7	53.273
5	Cow	16.7	0.0	0.0	6.0	96.9	2.4	44.8	27.1	0.0	30.1	52.087
6	Face	3.7	0.0	50.0	16.9	9.2	96.4	25.0	0.6	13.0	0.0	58.098
7	Horse	53.8	0.0	43.8	21.1	36.9	13.3	82.4	60.2	0.0	14.5	57.598
8	Mb	18.5	0.0	0.0	0.0	6.2	0.6	5.2	89.8	0.0	12.7	98.823
9	Person	14.8	39.1	6.3	2.4	3.1	21.7	13.5	5.4	30.4	0.0	1.216
10	Plane	13.0	0.0	25.0	1.2	10.8	0.6	10.4	37.4	0.0	76.5	83.333

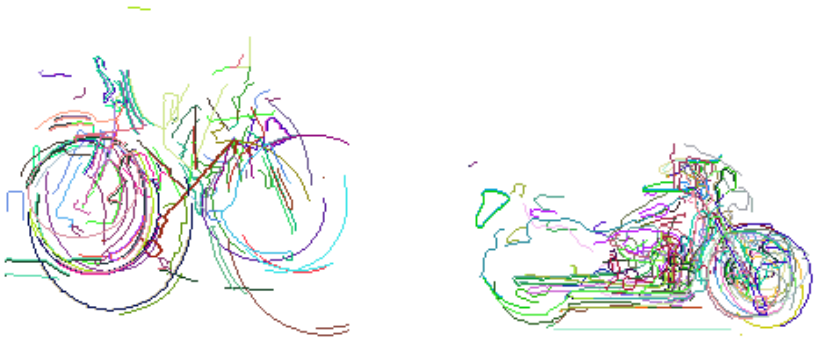
Tabelle 5.3: Die Klassifikationsraten des Experiments 20100511 mit der Reimplementierung des BFMs. Zur besseren Visualisierung sind die Felder mit dem entsprechenden Grauwert hinterlegt. Die Modelle wurden einzeln trainiert und gegeneinander getestet. Die Diagonale beinhaltet die RPCeERs der Modelle. In den übrigen Spalten stehen die Erkennungsraten der Modelle, auf den Testbildern der anderen Klassen. In der Spalte "Kl." stehen die Namen der Klassen mit den Abkürzungen CarsF für Cars front, CarsR für Cars rear und Mb für Motorbike. Die Spalte "Th" beinhaltet den zum Erreichen der RPCeER benötigten Detektions-Threshold. Die zum Lernen und Detektieren verwendeten Parameter sind in Tabelle B.2 auf der Seite 58 aufgelistet.

Nr.	**Kl.**	**Klassifikationsraten in %**										**Th**
		1	2	3	4	5	6	7	8	9	10	
1	Bike	15.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.737
2	Bottle	0.0	89.1	0.0	0.0	0.0	0.0	1.0	0.0	26.7	0.0	45.898
3	CarsF	1.9	0.0	20.0	0.6	0.0	0.0	0.0	0.6	0.0	0.6	23.857
4	CarsR	0.0	0.0	13.3	89.2	6.2	0.6	7.3	0.0	0.0	1.8	53.273
5	Cow	11.3	0.0	0.0	0.0	81.5	0.6	3.1	0.6	0.0	3.0	52.087
6	Face	3.8	0.0	20.0	1.2	0.0	94.0	10.4	0.6	20.0	0.0	58.098
7	Horse	37.7	0.0	20.0	1.2	6.2	1.2	62.5	16.3	0.0	7.8	57.598
8	Mb	15.1	0.0	0.0	0.0	4.6	0.0	2.1	78.3	0.0	3.6	98.823
9	Person	0.0	1.6	0.0	0.0	0.0	0.0	1.0	0.0	6.7	0.0	1.216
10	Plane	1.9	0.0	26.7	0.6	1.5	0.0	3.1	2.4	0.0	69.3	83.333

Tabelle 5.4: Die Klassifikationsraten des Experimentes 20100511 wenn jedem Bild nur eine Klasse zugewiesen wird. Als Entscheidung, welcher der zehn Klassifikatoren gültig ist, wenn mehr als einer am Objekt feuern, wurde die höhere Konfidenz verwendet.

Nr.	**Kl.**	**Klassifikationsraten in %**										**Th**
		1	2	3	4	5	6	7	8	9	10	
1	Bike	45.3	0.0	0.0	0.0	0.0	0.0	0.0	7.2	6.7	0.0	9.737
2	Bottle	0.0	87.5	0.0	0.0	0.0	0.0	1.0	0.0	13.3	0.0	45.898
3	CarsF	3.8	0.0	66.7	0.6	6.2	0.6	2.1	7.8	0.0	2.4	23.857
4	CarsR	0.0	0.0	13.3	89.2	4.6	1.2	9.4	3.0	0.0	1.8	53.273
5	Cow	3.8	0.0	0.0	0.6	84.6	0.6	5.2	1.8	0.0	9.0	52.087
6	Face	3.8	0.0	6.7	1.2	0.0	88.6	8.3	0.0	13.3	0.0	58.098
7	Horse	28.3	0.0	6.7	1.2	3.1	1.2	61.5	42.8	0.0	11.4	57.598
8	Mb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	32.5	0.0	1.8	98.823
9	Person	1.9	3.1	0.0	0.0	1.5	4.2	3.1	1.2	20.0	0.0	1.216
10	Plane	0.0	0.0	6.7	0.0	0.0	0.0	0.0	2.4	0.0	59.6	83.333

Tabelle 5.5: Die Klassifikationsraten des Experimentes 20100511 wenn jedem Bild nur eine Klasse zugewiesen wird. Als Entscheidung, welcher der zehn Klassifikatoren gültig ist, wenn mehr als einer am Objekt feuern, wurde die höhere normierte Konfidenz verwendet.



(a) Bike (b) Motorbike

Abbildung 5.3: Die Klassifikatoren für Bike und Motorbike aus dem Referenzexperiment. Da die Objekte in den Validierungs-Bildern für Bike nicht einheitlich ausgerichtet sind, bleibt nur die Gemeinsamkeit, nämlich die Räder, über. Die Validierungs-Bilder der Klasse Motorbike beinhalten die Objekte hingegen sehr dominant, deshalb ist in diesem Modell wesentlich mehr Information vorhanden.

Idealerweise sollten die Werte in der Diagonale sehr groß und alle anderen sehr klein sein. Das würde bedeuten, dass die Klassifikatoren nur bei der gelernten Klasse ansprechen. Wie in Tabelle 5.3 zu sehen ist, ist das aber nicht der Fall. Die meisten Klassifikatoren haben auch in den Testbildern anderer Klassen Detektionen, die über dem Detektions-Threshold liegen. Die Asymmetrie in der Tabelle resultiert daher, dass die einzelnen Klassifikatoren nicht aus denselben BFs bestehen. Als Beispiel sind die Klassifikatoren für Bike und Motorbike in Abbildung 5.3 angegeben. Der Klassifikator für Bike feuert auf den Testbildern von Motorbike öfters, als umgekehrt. Werden diese zwei verglichen, erkennt man, dass der Klassifikator für Motorbike viel spezifischer ist (innere Struktur und Form), als jener für Bike, welcher praktisch nur die Räder beinhaltet. Die starken Unterschiede in der optischen Qualität der Modelle resultieren aus den für das Boosting verwendeten Validierungs-Bildern. Während die Objekte in der Motorbike-Datenbank alle in dieselbe Richtung zeigen (nach rechts) und sehr wenig Hintergrund aufweisen, sind die Objekte in der Bike-Datenbank nicht einheitlich ausgerichtet. Das Lernverfahren kann sich beim Modell für Motorbike wesentlich mehr auf detailliertere Strukturen konzentrieren, wo hingegen beim Modell für Bike nur Gemeinsamkeiten zwischen den verschieden orientierten Objekten, nämlich Vorder- und Hinterrad, übrig bleiben.

Auch Klassifikatoren, die sehr allgemeine BFs, wie z.B. waagrechte oder senkrechte Linien, besitzen, feuern auf nahezu allen Klassen. Beispiele dafür sind die Modelle für Cars front und Cars rear. Die einzelnen Modelle des Referenzexperiments sind in Abbildung B.1 auf der Seite 59 dargestellt.

Mit dem ermittelten Threshold zum Erreichen der RPCeER kann nun für jeden Klassifikator einzeln bestimmt werden, wann eine Detektion in einem Testbild gültig ist (Nebendiagonalen in der Tabelle 5.3). Es stellt sich nun die Frage, welche Detektion gültig ist, wenn mehrere Klassifikatoren am selben Objekt feuern, was durchaus passiert, wie Tabelle 5.3 zeigt. Es ist mathematisch nicht richtig, jene als die Richtige zu verwenden, welche die höchste Konfidenz laut Formel (4.1) auf der Seite 29 besitzt, denn die Modelle wurden einzeln trainiert und ihre Gewichte sind deshalb nicht vergleichbar. Um dennoch zu untersuchen, ob der Multi-Class-Klassifikator besser zwischen den Klassen unterscheiden kann, wurden zwei Methoden zum Treffen einer Entscheidung verwendet:

1. Jener Klassifikator, mit der höchsten Konfidenz ist gültig.

2. Die berechnete Konfidenz einer Detektion wird mit jener zum Erreichen der RP-CeER des jeweiligen Klassifikators normiert. Jener Klassifikator, mit der höchsten normierten Konfidenz gewinnt.

Die Ergebnisse für Punkt 1 sind in Tabelle 5.4 und jene für Punkt 2 in Tabelle 5.5 zusammengefasst. Im Gegensatz zu den Ergebnissen in Tabelle 5.3 bedeuten die Angaben in diesen Tabellen, wie viele Testbilder von welchem Klassifikator erkannt wurden. Es wurden also keine Hintergrundbilder mehr verwendet und jedem Bild wurde nur eine Klasse zugewiesen. Wie deutlich zu erkennen ist, liefert die Normierung der Konfidenz bessere Resultate. Das entspricht den Erwartungen, denn Modelle, die beim Trainieren einen geringeren Fehler aufweisen, haben immer höhere Gewichte α (Formel (2.2) auf der Seite 9) und deshalb auch höhere Konfidenzen.

5.4 Experimente mit dem neuen Algorithmus

Die ursprüngliche Idee des BFMs wurde bereits in [Opelt u. a., 2006a,b] diskutiert und mit vielversprechenden Experimenten hinterlegt. In diesem Kapitel werden die vorgestellten Änderungen, nämlich die Einführung einer geometrischen Beziehung zwischen den BFs eines weak-classifiers und einem Multi-Class-Boosting-Algorithmus, untersucht. Im ersten Teil wird der Einfluss der geometrischen Beziehung gezeigt. Dazu werden Modelle trainiert und die Auswirkung anhand der optischen Qualität der Modelle betrachtet. Danach werden die Ergebnisse des Multi-Class-Verfahrens mit jenen vom Referenzexperiment verglichen, und die Unterschiede untersucht.

5.4.1 Wahl der Parameter

Die Ergebnisse mit dem neuen Algorithmus werden mit jenen aus dem Referenzexperiment verglichen. Deshalb wurden auch zum Trainieren und Testen dieselben Parameter verwendet. Eine genaue Auflistung der eingestellten Parameter für die folgenden Experimente ist im Anhang B ab Seite 57 zu finden. In der neuen Methode sind zwei Parameter hinzugekommen, die noch bestimmt werden müssen: Der Einfluss der geometrischen Beziehung bei der Wahl von weak-classifiers und der Threshold für die Chamfer-Distanz bei der Suche von BFs in neuen Bildern.

Die geometrische Beziehung ist eine Neuerung, die zum Einen das Clustering von BFs überflüssig machen soll und zum anderen sicherstellen soll, dass ein weak-classifier bereits möglichst viel vom Objekt abdeckt. Auf die Wahl des Parameters zum Steuern des Einflusses der geometrischen Beziehung wird in Kapitel 5.4.2 genauer eingegangen.

Um die Auswirkung des Thresholds für die Chamfer-Distanz zu untersuchen, wurde im Experiment 20100614 ein Klassifikator trainiert und die Detektion mit unterschiedlichen Thresholds durchgeführt. In Abbildung 5.4 sind die RPCeERs aller Klassen in Abhängigkeit des verwendeten Thresholds für die Chamfer-Distanz zu sehen. Aus dieser Abbildung ist klar ersichtlich, dass sich die Erkennungsleistung verschlechtert, wenn der Threshold für das Chamfer-Matching zu groß wird. Grund dafür sind falsche Matches von BFs im Bild. Abbildung 5.5 zeigt sehr gut, dass ab einem bestimmten Threshold zu viele BFs matchen und diese die Bounding-Box fälschlicherweise vergrößern, was schlussendlich zu einer falschen Detektion führt. Die Trainings- und Testparameter des Experimentes

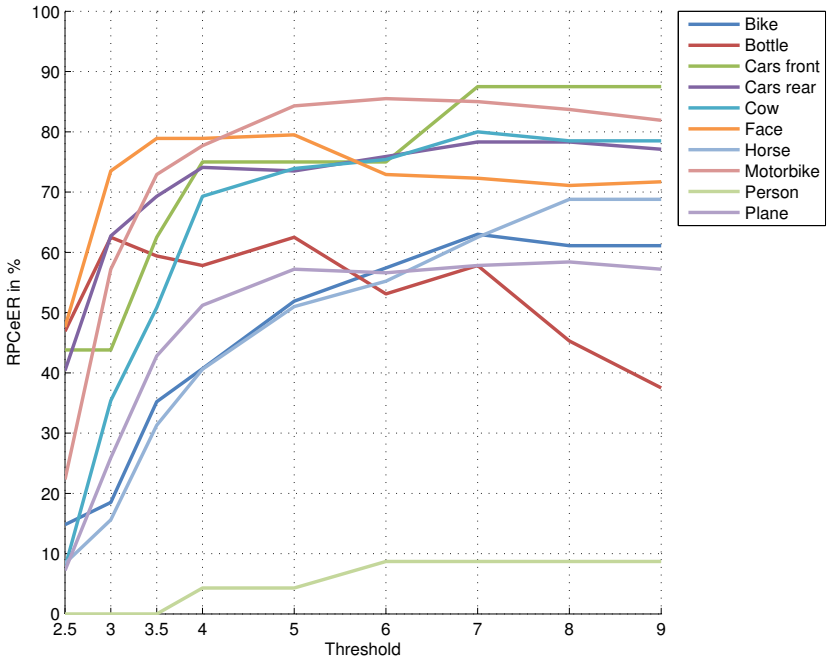


Abbildung 5.4: Die RPCeERs des Experimentes 20100614 in Abhängigkeit des Thresholds th_{BF}, der in der Detektion für die Chamfer-Distanz verwendet wurde.

sind wieder im Anhang B auf der Seite 60 zu finden. Für die weiteren Experimente wurde $th_{BF} = 7$ gewählt.

5.4.2 Auswirkung der geometrischen Beziehung

In Kapitel 3.2 auf der Seite 17 wurde die Schwierigkeit beim Clustern von ähnlichen BFs aufgezeigt und mit einer geometrischen Beziehung zwischen den CVs eine mögliche Lösung für das Problem präsentiert. Durch diese Beziehung zwischen den CVs eines weak-classifiers sollen im Lernschritt nun Kombinationen von CVs bevorzugt werden, die sich nicht ähnlich sind. Um die Funktion zu testen, wurde im Experiment 20100818 zuerst der Einfluss der geometrischen Beziehung mit $\alpha = 0$ deaktiviert und ein Klassifikator trainiert. Die ersten 14 weak-classifiers, die CVs für die Klasse Bottle besitzen, sind in Abbildung 5.6a abgebildet. Bei der Auswahl von weak-classifiers, die die Kostenfunktion Z minimieren, werden nun jene BFs mit den entsprechenden CVs für jede Klasse ausgewählt, die auch tatsächlich den geringsten Fehler haben. Es können Kombinationen, wie z.B. die Zweite aus Abbildung 5.6a, entstehen, die eigentlich vermieden werden sollen, denn die Information in diesen ist redundant. Wird α erhöht, sprich ähnliche CVs werden mehr bestraft, werden solche Kombinationen nicht mehr ausgewählt. Abbildung 5.6b zeigt ausgewählte Alternativen, die zwar nicht mehr den absolut geringsten Trainingsfehler aufweisen (in der Kostenfunktion wird jetzt auch die geometrische Beziehung zwischen den CVs berücksichtigt), dafür aber dem Klassifikator neue Informationen hinzufügen.

Experimente mit unterschiedlichen Werten für α (vgl. Abbildung 5.7) haben gezeigt, dass mit $\alpha = 0.3$ ein guter Kompromiss zwischen der Vermeidung von ähnlichen CVs und der Verfälschung der Kostenfunktion laut Schapire u. Singer [1998] gegeben ist.

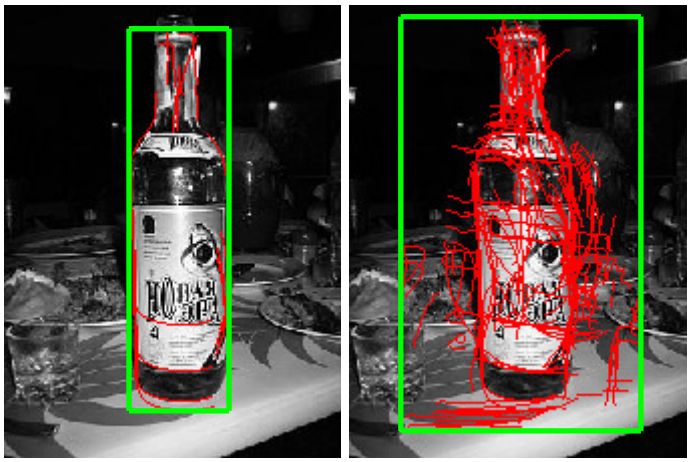


Abbildung 5.5: Das Bild Bottle_67 aus dem Experiment 20100614 mit unterschiedlichen Thresholds für das Chamfer-Matching. Die Detektion links wurde mit einem Threshold von 3 durchgeführt, rechts mit 9. Da rechts viel mehr BFs matchen (rot), wird die Bounding-Box (grün) zu groß und die Detektion wird als falsch gewertet.

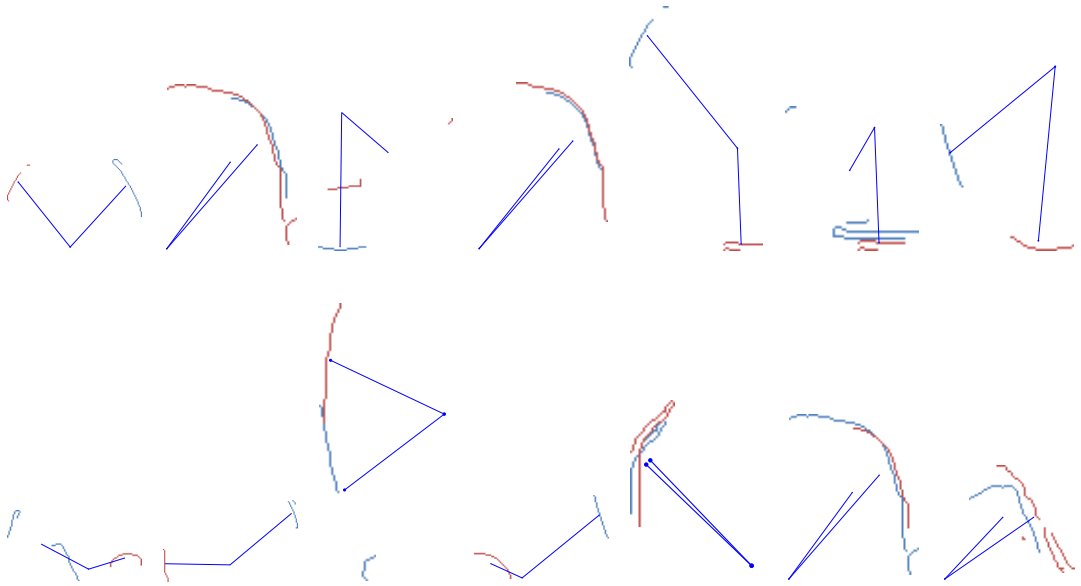
Vorverarbeitung	Boosting	Summe	Detektieren
110:36	008:33	119:09	016:32

Tabelle 5.6: Die benötigte Rechenzeit des Experiments 20100824 in hhh:mm zum Trainieren und Testen des Multi-Class-Klassifikators. Die Dauer zum Trainieren eines Modells ergibt sich aus Vorverarbeitung+Boosting.

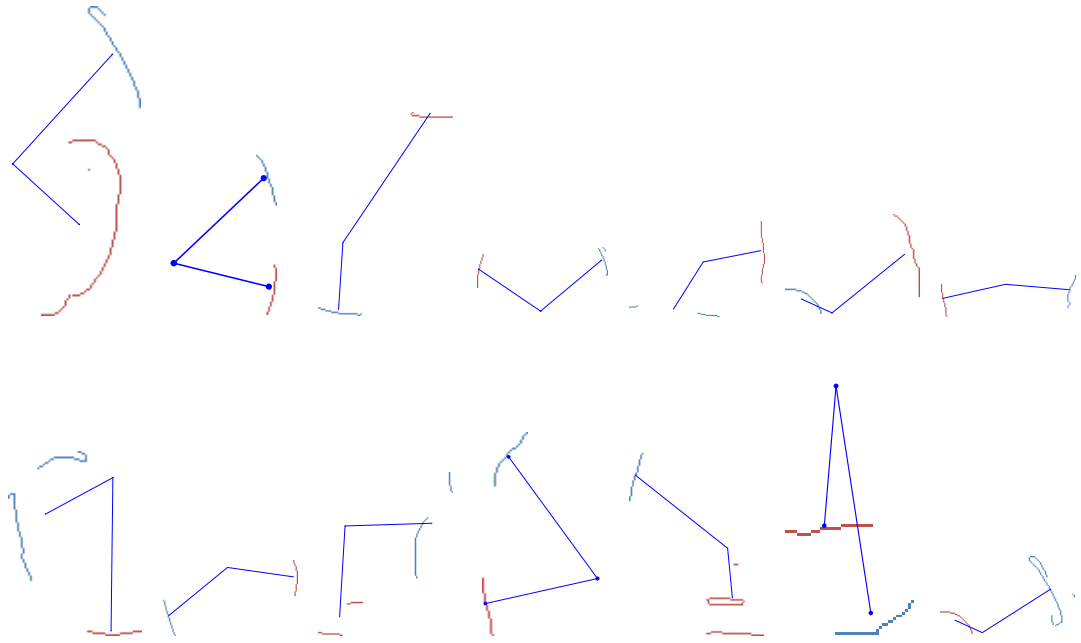
5.4.3 Wie gut funktioniert die Multi-Class-Erweiterung?

Mit den Parametern aus dem Referenzexperiment und den im vorherigen Kapitel bestimmten neuen Parametern $\alpha = 0.3$ und $th_{BF} = 7$ wird nun das Experiment 20100824 durchgeführt. In diesem wird ein Multi-Class-Klassifikator wie er in dieser Arbeit beschrieben wurde trainiert und auf den Testbildern aller Klassen getestet. Die Dauer zum Trainieren und Testen des Klassifikators ist in Tabelle 5.6 aufgelistet. Wie bereits in Kapitel 3.3 auf der Seite 17 beschrieben wurde, müssen in der Vorverarbeitung die ausgeschnittenen BFs auf allen Validierungs-Bildern getestet werden, um mögliche CVs auszuschneiden und zu prüfen, ob diese das Objektzentrum in den Validierungs-Bildern treffen. Das ist notwendig, damit die benötigte Performance im Boosting-Algorithmus erreicht werden kann. Die lange Dauer für die Vorverarbeitung resultiert daher, dass 4000 BFs auf 200 positiven Validierungs-Bilder (20 je Klasse) zuerst mit Chamfer-Matching gefunden, dann die Minima der Distanz-Map mit Mean-Shift geclustert und schlussendlich die CVs berechnet und wieder geclustert werden müssen. Die dafür benötigte Dauer beträgt durchschnittlich 0.5s.

Nachdem das Modell trainiert wurde, kann überprüft werden, wie die weak-classifier, bzw. die CVs, über die Klassen verteilt wurden. Es sollte, wie in Kapitel 3.5 auf der Seite 24 erwähnt, nicht passieren, dass schwierige Klassen über- und leichte Klassen untertrainiert werden. Die Abbildung 5.8a zeigt, in welchem Boosting-Schritt einer Klasse CVs hinzugefügt wurden. Nur in der ersten Runde wird eine Kombination von BFs gefunden, die für acht Klassen gültige CVs besitzt. Danach werden laut Abbildung 5.8b im Durchschnitt nur mehr 3.6 Klassen pro weak-classifier erkannt. Schaut man sich die Abbildung 5.8a genauer an, erkennt man, dass schwierige Klassen, wie Bike (1) und Person (9) mehr weak-classifier



(a) $\alpha = 0$



(b) $\alpha = 0.3$

Abbildung 5.6: Die ersten 14 weak-classifiers des Experimentes 20100818, die CVs für die Klasse Bottle besitzen. Aus dem Bild ist ersichtlich, dass durch die geometrische Beziehung ($\alpha > 0$) zwischen den CVs verhindert wird, dass BFs mit ähnlichen CVs, wie die Zweite, Vierte und die letzten Drei aus (a), zu einem weak-classifier zusammengefügt werden.

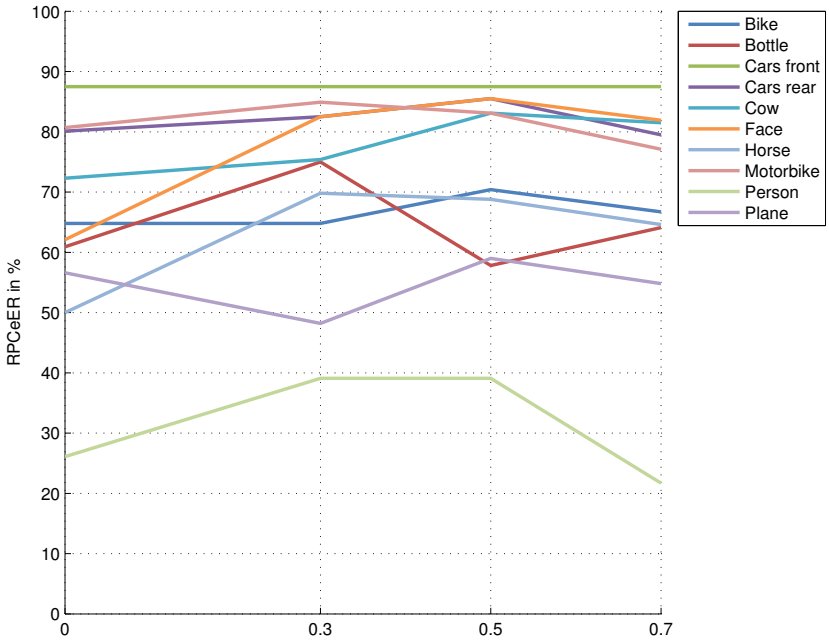
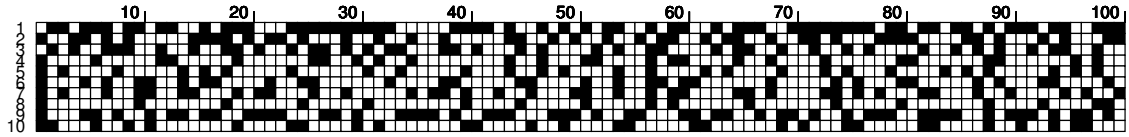


Abbildung 5.7: Die RPCeERs des Experimentes 20100811 in Abhängigkeit des Einflusses der geometrischen Beziehung zwischen den CVs der BFs.

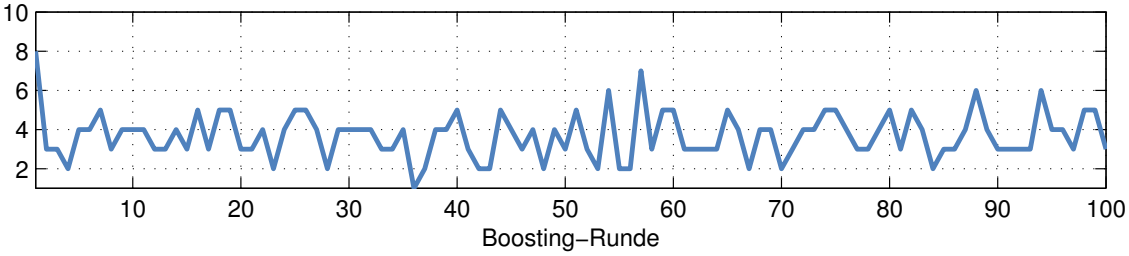
besitzen, als einfachere, wie z.B. die Klasse Motorbike (8), was, wie in Kapitel 3.5 auf der Seite 24 bereits erwartet wurde.

Zur Auswertung der Detektionsergebnisse werden wieder dieselben Tabellen wie beim Referenzexperiment erzeugt. Tabelle 5.7 zeigt in der Diagonale die RPCeERs des Klassifikators auf den positiven und negativen Testbildern der einzelnen Klassen. In den Nebendiagonalen stehen wieder die Klassifikationsraten einer Klasse, auf den Testbildern der anderen Klassen.

Ein Vergleich der Tabelle 5.7 mit den Ergebnissen des Referenzexperimentes in Tabelle 5.3 auf der Seite 36 zeigt, dass die RPCeERs des Multi-Class-Klassifikators auf den einzelnen Klassen um ca. 12% schlechter sind. Am meisten Unterschied tritt bei den Klassen Bike (11.1%), Bottle (14.1%), Cow (21.5%), Face (13.9%), Horse (12.6%) und Plane (28.3%) auf. Betrachtet man die Klassifikatoren des Referenzexperimentes auf Seite 59 und jene des Multi-Class-Klassifikators auf Seite 62 erkennt man starke Unterschiede in der optischen Qualität. Betrachtet man zusätzlich die durchschnittliche Länge der BFs in den trainierten Klassifikatoren in Tabelle 5.8 erkennt man, dass die BFs des Multi-Class-Klassifikators wesentlich kürzer sind, als jene der einzeln trainierten Modelle. Das bedeutet, dass die BFs in den Klassifikatoren des Referenzexperiments wesentlich diskriminativer sind, als jene des Multi-Class-Klassifikators. Dies ist auch in der grafischen Darstellung der Modelle, z.B. anhand des fehlenden Kopfes der Kuh und des Pferdes, an den vielen kleinen BFs im Modell für Plane oder an dem fehlenden Nummernschild in der Klasse Cars front, zu sehen. Die Ursache, warum der Multi-Class-Klassifikator die Testbilder der Klasse Motorbike, trotz signifikanter Unterschiede des Modells zum Motorbike-Klassifikator des Referenzexperimentes, fast gleich gut klassifiziert, ist einfach zu finden. Werden die Testbilder auf der Seite 55 betrachtet, erkennt man, dass diese das Objekt sehr dominant und mit wenig Hintergrund beinhalten, d.h. der Klassifikator kann nur am Objekt und



(a)



(b)

Abbildung 5.8: Die Abbildung (a) zeigt die Verteilung der CVs im Multi-Class-Klassifikator des Experimentes 20100824. Ein schwarzes Feld bedeutet, dass der Klasse (Zeile) im jeweiligen Boosting-Schritt (Spalte) CVs zugewiesen wurden. Die Nummern der Klassen stehen für Bike (1), Bottle (2), Cars front (3), Cars rear (4), Cow (5), Face (6), Horse (7), Motorbike (8), Person (9) und Plane (10). In (b) ist die Anzahl Klassen, die vom weak-classifier der jeweiligen Boosting-Runde klassifiziert werden kann, aufgetragen.

Nr.	**Kl.**	**Klassifikationsraten in %**										**Th**
		1	2	3	4	5	6	7	8	9	10	
1	Bike	64.8	0.0	7.4	1.9	24.1	3.7	27.8	18.5	27.8	29.6	69.135
2	Bottle	3.1	75.0	0.0	0.0	1.6	3.1	0.0	3.1	42.2	6.3	92.437
3	CarsF	12.5	6.3	87.5	18.8	62.5	43.8	25.0	18.8	6.3	62.5	38.093
4	CarsR	1.8	0.0	26.5	82.5	7.2	3.6	6.6	0.0	0.6	3.0	35.572
5	Cow	30.8	0.0	43.1	0.0	75.4	0.0	6.1	27.7	6.2	32.2	72.857
6	Face	7.8	3.6	9.0	4.2	1.8	82.5	17.5	1.8	18.1	4.8	50.568
7	Horse	53.1	1.0	40.6	14.6	41.7	35.4	69.8	10.4	20.8	36.5	45.652
8	Mb	64.5	0.6	41.0	5.4	56.6	4.2	0.6	84.9	23.5	12.7	117.276
9	Person	4.3	21.7	4.3	0.0	0.0	8.7	4.3	0.0	39.1	4.3	42.916
10	Plane	9.0	0.6	59.0	0.6	37.4	0.6	1.2	7.2	7.2	48.2	63.157

Tabelle 5.7: Die Klassifikationsraten des Experiments 20100824 mit dem neuen Multi-Class-Lernverfahren. Zur besseren Visualisierung sind die Felder mit dem entsprechenden Grauwert hinterlegt. Der trainierte Klassifikator wurde auf den Testbildern aller Klassen getestet. Die Diagonale beinhaltet die RPCeERs für die einzelnen Klassen. In den übrigen Spalten stehen die Erkennungsraten des Klassifikators, auf den Testbildern der anderen Klassen. In der Spalte "Kl." stehen die Namen der Klassen mit den Abkürzungen CarsF für Cars front, CarsR für Cars rear und Mb für Motorbike. Die Spalte "Th" beinhaltet den zum Erreichen der RPCeER benötigten Detektions-Threshold. Die zum Lernen und Detektieren verwendeten Parameter sind in Tabelle B.5 auf der Seite 61 aufgelistet.

	Bike	**Btl**	**CF**	**CR**	**Cow**	**Face**	**Horse**	**Mb**	**Pers**	**Plane**
Ref	85.1	121.9	110.0	128.9	111.4	97.1	111.8	113.9	116.2	114.5
MC	51.6									

Tabelle 5.8: Die durchschnittliche Länge in Pixel aller BFs in den Modellen des Referenzexperimentes (Ref) und des Multi-Class-Klassifikators (MC) aus dem Experiment 20100824. Die Abkürzungen stehen für Bottle (Btl), Cars front (CF), Cars rear (CR), Motorbike (Mb) und Person (Pers).

nicht im Hintergrund feuern. Auch mag die grafische Darstellung der Klasse Face beim Multi-Class-Klassifikator auf den ersten Blick besser ausschauen, als die des Referenzexperimentes. Betrachtet man das Modell aber genauer, erkennt man, dass Augen, Nase und Mund beim Multi-Class-Klassifikator aus sehr kurzen BFs bestehen, während der Mund im Modell des Referenzexperimentes im Wesentlichen aus einem längeren BF besteht.

Der Vorteil des neuen Klassifikators soll aber nicht bei einer besseren Erkennungsleistung liegen, sondern in einer mathematisch korrekten Methode, um zwischen Klassen zu unterscheiden. Wird die Klassifikationsregel (2.6) von der Seite 10 angewandt, ergeben sich die Detektionsraten laut Tabelle 5.9. Es wird also immer die Detektion mit der höchsten Konfidenz gewertet, sofern diese über dem Detektions-Threshold der jeweiligen Klasse liegt. Ein Vergleich mit der Tabelle 5.5 auf der Seite 37 zeigt, dass die Unterscheidbarkeit nicht besser funktioniert, als im Referenzexperiment, wenn bei diesem der normierte Detektions-Threshold als Entscheidung verwendet wird. Berechnet man, wie viel Prozent der Detektionen auf den Testbildern einer Klasse von dem zugehörigen Klassifikator stammt, schneidet, wie in Tabelle 5.10 zu sehen ist, das Referenzexperiment mit den normierten Detektions-Thresholds sogar leicht besser ab.

Warum das so ist, bedarf noch einer Erklärung. Prinzipiell funktioniert der Multi-Class-Ansatz, denn die Unterscheidbarkeit zwischen den Klassen ist durchaus gegeben. Wie bereits weiter oben beschrieben, ist der Grund für die geringen Klassifikationsraten auf zu wenig diskriminative BFs im Klassifikator zurückzuführen. Die etwas schlechtere Unterscheidbarkeit gegenüber den Klassifikatoren aus dem Referenzexperiment kann daher nur auf das Multi-Class-Lernverfahren zurückzuführen sein. Für die Unterscheidung zwischen den Klassen ist es demnach schlechter, das Risiko einzugehen, mit einem Multi-Class-Lernverfahren generellere BFs auszuwählen, als die Modelle unabhängig voneinander zu trainieren und dafür sehr diskriminative weak-classifier zu erhalten. Des Weiteren hat auch die Größe der zu lernenden Objekte einen Einfluss auf das Ergebnis. Zum Beispiel besitzen Fahrrad und Auto (jeweils von der Seite betrachtet) Vorder- und Hinterräder. Sind die Räder beider Klassen im Training gleich groß skaliert, können diese vom Lernalgorithmus verwendet werden, denn es ändert sich zwischen Auto und Fahrrad nur die Position der Räder, nicht aber die Form. Ist die Größe der Räder hingegen nicht dieselbe, wird das Lernverfahren andere, nicht so diskriminative BFs auswählen, die in beiden Klassen vorkommen. Ein Beispiel, aus dem gelernten Multi-Class-Klassifikator sind die Klassen Person und Face. Beide Klassen sind auf eine Höhe von 200 Pixeln skaliert, d.h. BFs die in der Klasse Face den Kopf darstellen, können niemals für den Kopf in der Klasse Person verwendet werden.

Nr.	**Kl.**	**Klassifikationsraten in %**										**Th**
		1	2	3	4	5	6	7	8	9	10	
1	Bike	42.3	0.0	0.0	0.0	1.5	1.2	7.3	7.8	0.0	1.8	69.135
2	Bottle	0.0	67.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	92.437
3	CarsF	0.0	0.0	80.0	4.8	10.8	2.4	6.3	6.0	0.0	45.2	38.093
4	CarsR	0.0	0.0	0.0	78.9	0.0	0.0	1.0	0.0	0.0	0.0	35.572
5	Cow	1.9	0.0	0.0	0.0	69.2	0.0	10.4	4.2	0.0	9.6	72.857
6	Face	0.0	0.0	0.0	0.0	1.5	75.9	3.1	0.0	0.0	0.0	50.568
7	Horse	0.0	0.0	0.0	0.0	1.5	2.4	31.3	0.0	0.0	0.0	45.652
8	Mb	5.8	0.0	0.0	0.0	0.0	0.0	1.0	71.1	0.0	1.2	117.276
9	Person	19.2	18.8	0.0	0.6	1.5	6.0	15.6	7.8	55.6	0.6	42.916
10	Plane	1.9	0.0	13.3	0.0	3.1	0.0	11.5	2.4	0.0	24.7	63.157

Tabelle 5.9: Die Klassifikationsraten des Experimentes 20100824 wenn die Klassifikationsregel laut Formel (2.6) auf Seite 10 angewandt wird. Jedem Bild wird also nur eine Klasse zugewiesen, nämlich jene mit der höchsten Konfidenz. Eine Detektion ist aber nur gültig, wenn diese über dem Detektions-Threshold Th der jeweiligen Klasse liegt.

	Bike	**Btl**	**CF**	**CR**	**Cow**	**Face**	**Horse**	**Mb**	**Pers**	**Plane**
Ref	52.2	**96.6**	66.7	**96.1**	**84.6**	**91.9**	**67.8**	32.9	37.5	**69.2**
MC	**59.5**	78.2	**85.7**	93.6	77.6	86.3	35.7	**71.5**	**100.0**	29.7

Tabelle 5.10: Die Tabelle zeigt, wie viel Prozent der Detektionen einer Klasse vom entsprechenden Klassifikator gemacht wurden. In der Zeile "Ref" wurden die Klassifikationsraten des Referenzexperimentes aus Tabelle 5.5 verwendet und in der Zeile "MC" jene des Multi-Class-Klassifikators aus der Tabelle 5.9. Die Abkürzungen stehen für Bottle (Btl), Cars front (CF), Cars rear (CR), Motorbike (Mb) und Person (Pers).

Kapitel 6

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war zu testen, ob das BFM von Opelt u. a. [2006a] so erweitert werden kann, dass mehrere Klassen in Abhängigkeit voneinander trainiert werden können. Das originale BFM konnte sehr gut gelernte Klassen vom Hintergrund unterscheiden, allerdings fehlte eine Methode, um die einzeln trainierten Klassifikatoren miteinander zu vergleichen. Es war also unklar, was passieren soll, wenn mehrere Klassifikatoren am selben Objekt feuern. In den vorangegangenen Kapiteln wurden die notwendigen Maßnahmen beschrieben, um zu versuchen, das Problem mit einem Multi-Class-Lernverfahren zu lösen. Des Weiteren wurden einige Ideen, die während den Experimenten mit der reimplementierten Version des BFM entstanden, umgesetzt: (i) Die BFs eines weak-classifiers sollen nicht beliebig zusammengesetzt werden, sondern einer geometrischen Beschränkung unterliegen und (ii) die negativen Validierungs-Bilder werden nicht mehr verwendet.

Die geometrische Beziehung zwischen den CVs der BFs eines weak-classifiers hat folgenden Grund: Dadurch soll das vorherige Clustern der BFs entfallen, denn BFs mit ähnlichen CVs werden so im Lernverfahren mehr bestraft und deshalb eher vermieden. In den Experimenten auf Seite 40 wurde gezeigt, dass bei Berücksichtigung der formulierten Beziehung zwischen den CVs eines weak-classifiers, tatsächlich auf das Clustering verzichtet werden kann. Allerdings wird die ursprüngliche Kostenfunktion zur Bestimmung von weak-classifiers beim Boosten umso mehr verfälscht, je größer der Einfluss der geometrischen Beziehung ist. Mittels Experimenten, bei denen dieser Einfluss verändert wurde, konnte schlussendlich ein Wert bestimmt werden, mit dem sehr wenig ähnliche CVs vorhanden sind und gute Detektionsergebnisse erreicht werden.

Durch Weglassen der negativen Validierungs-Bilder wird nicht nur die Laufzeit des Algorithmus verringert, sondern auch gezeigt, dass in den positiven Validierungs-Bildern genügend Hintergrundinformation vorhanden ist, wenn diese auch verwendet wird. In dieser Arbeit wurden alle BFs verworfen, die in den positiven Validierungs-Bildern nur außerhalb der Bounding-Box gefunden wurden. Im Lernverfahren standen also nur noch BFs zur Auswahl, die bereits auf den Objekten gefeuert haben.

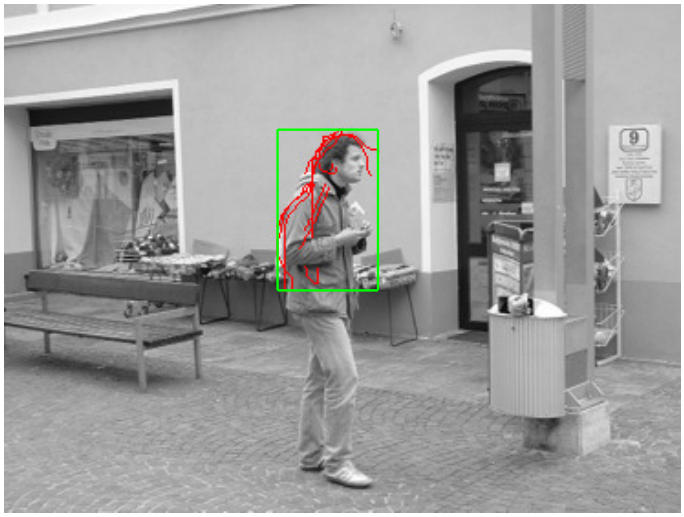
Die Auswertung der Detektionsergebnisse mit dem neuen Verfahren geschah mit Hilfe eines Referenzexperimentes, welches mit der reimplementierten Version des BFMs von Opelt u. a. [2006a] durchgeführt wurde. In diesem wurden Klassifikatoren der zehn Klassen einzeln trainiert und auf den Testbildern aller Klassen getestet. Da auf den einzelnen Testbildern mehrere Klassifikatoren feuerten, wurden die Konfidenzen der Detektionen mit dem Detektions-Threshold normiert und jene Detektion, mit der maximalen normierten Konfidenz, als die Gültige verwendet. Mit dem in dieser Arbeit vorgestellten Klassifikator wurde das selbe Experiment durchgeführt, allerdings wird die Entscheidung, welche De-

tektion bei mehreren gültig ist, vom Klassifikator übernommen. Dieser wird so trainiert, dass immer die Detektion mit der höchsten Konfidenz die Richtige ist. Der Vergleich beider Experimente zeigt, dass erstens die RPCeERs des Referenzexperimentes besser sind als die des Multi-Class-Klassifikators und zweitens dass das Referenzexperiment auch etwas besser zwischen den Klassen unterscheiden kann. Der Grund für das schlechtere Abschneiden des Multi-Class-Klassifikators liegt im Wesentlichen darin, dass die BFs im Durchschnitt kürzer, also nicht so diskriminativ, sind, als jene aus den einzeln trainierten Klassifikatoren des Referenzexperimentes. Es ist also nicht sinnvoll, Multi-Class-Modelle zu trainieren, die sehr unterschiedliche Klassen und damit auch sehr unterschiedliche BFs beinhalten, denn das Lernverfahren versucht, Kombinationen von BFs zu finden, die für möglichst viele Klassen gültig sind. Eine mögliche Anwendung des vorgestellten Verfahrens könnte die Erkennung bzw. Unterscheidung von sehr ähnlichen Klassen sein. Wenn solche Klassen gemeinsam trainiert werden, können, aufgrund der Ähnlichkeit der Klassen, sehr diskriminative BFs ausgewählt und trotzdem Unterschiede zwischen den Klassen gelernt werden. Diese Anwendung muss aber in zukünftigen Untersuchungen genauer analysiert werden.

Eine weitere Schwachstelle im vorgestellten Lernverfahren, ergibt sich aus dem Unterschied zwischen Trainingsfehler und Detektionsfehler. Damit ein weak-classifier im Lernverfahren einen geringen Fehler aufweist, ist es nur erforderlich, dass beide CVs der BFs das Objektzentrum der positiven Validierungs-Bilder treffen. Eine Detektion ist aber erst gültig, wenn die Bounding-Box der Detektion mit der Annotierung übereinstimmt. Lernverfahren funktionieren nur dann gut, wenn für das Lernen und Detektieren dieselben Methoden verwendet werden. Zukünftige Überlegungen sollten vor allem auch in diese Richtung gehen.

Möglichkeiten zur Verbesserung der Detektionsleistung stecken aber nicht nur im Lernverfahren. Oft feuern weak-classifier nicht am ganzen Objekt verteilt, sondern nur in bestimmten Bereichen. Als Beispiel soll die Abbildung 6.1 dienen. Das passiert vor allem, wenn Teile des Objektes schwer vom Hintergrund zu trennen sind, und deshalb der Kantendetektor an dieser Stelle keine Information liefert. Die resultierende Bounding-Box wird dann nur um die BFs der weak-classifier gezeichnet und umfasst deshalb nicht das ganze Objekt. Im gelernten Modell steckt aber die Information, an welcher Stelle sich der restliche Teil des Objektes befinden muss, und die Bounding-Box könnte korrigiert werden.

(a) Das gelernte Modell der Klasse Person.



(b) Nur ein Teil der weak-classifier des Modells feuern an der Person.

Abbildung 6.1: Es kommt oft vor, dass nur ein Teil der weak-classifier des Modells am Objekt feuern und dadurch die Bounding-Box nicht das ganze Objekt umschließt. Durch verwenden der vorhandenen Information des Modells, könnte auf die tatsächliche Bounding-Box geschlossen werden.

Anhang A

Die verwendete Bild-Datenbank

In Kapitel 5.1 auf der Seite 32 wurde beschrieben, wie viel Bilder zum Trainieren und Testen der Modelle verwendet wurden. Um die Experimente nachvollziehbar zu machen, sind die verwendeten Bilder hier aufgelistet. In der Tabelle A.1 sind die Indizes dieser Bilder zu finden, wenn das entsprechende Verzeichnis mit Matlab ausgelesen[1] wird. Einige ausgewählte Beispielbilder mit der zugehörigen Annotierung aus jeder Klasse sind in den Abbildungen A.1 bis A.3 zu sehen.

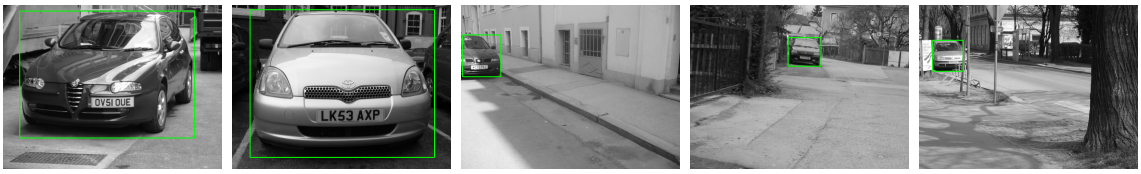
Nr.	**Klasse**	**Lernen**		**Testen**	
		Trainingsbilder	positive Validierungsbilder	positiv	negativ
1	Bike	1 ... 20	1 ... 20	1 ... 53	1 ... 53
2	Bottle	1 ... 20	1 ... 20	1 ... 64	1 ... 64
3	Cars front	1 ... 17	1 ... 17	1 ... 16	1 ... 16
4	Cars rear	1 ... 20	1 ... 20	1 ... 166	1 ... 166
5	Cow	1 ... 20	1 ... 20	1 ... 65	1 ... 65
6	Face	1 ... 20	1 ... 20	1 ... 166	1 ... 166
7	Horse	1 ... 20	1 ... 20	1 ... 96	1 ... 96
8	Motorbike	1 ... 20	1 ... 20	1 ... 166	1 ... 166
9	Person	1 ... 19	1 ... 20	1 ... 18	1 ... 18
10	Plane	1 ... 20	1 ... 20	1 ... 166	1 ... 166

Tabelle A.1: Die zum Trainieren und Testen verwendeten Bilder aus der Graz02-MC-Datenbank. 1 ... 20 bedeutet, dass die ersten 20 Bilder verwendet wurden. Es ist allerdings darauf zu achten, dass die Sortierung von Matlab/Linux verwendet wird und nicht jene von Windows.

[1] Mit dem Befehl `images=dir('*.png')`. Es ist darauf zu achten dass Windows anders sortiert, als Matlab.

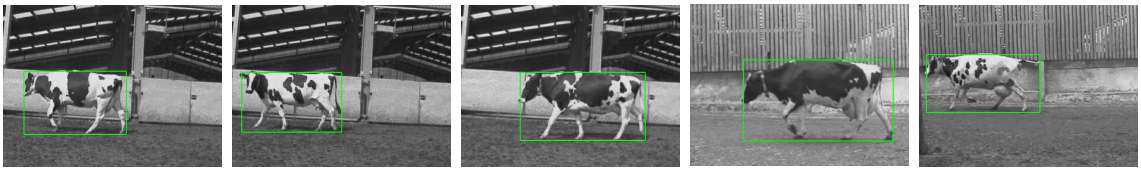
(a) Klasse “Bike” (bike_057.png; bike_131.png; bike_178.png; bike_206.png; bike_209.png)

(b) Klasse “Bottle” (Bottle_1.png; Bottle_118.png; Bottle_19.png; Bottle_3.png; Bottle_60.png)



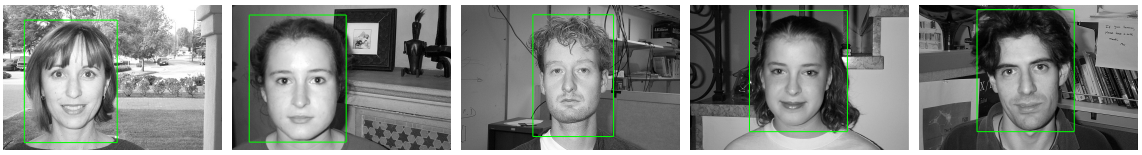
(c) Klasse “Cars front” (CarFront_105_0105.png; CarFront_47_0047.png; carsgraz_010.png; carsgraz_123.png; carsgraz_285.png)

(d) Klasse “Cars rear” (image_0109.png; image_0124.png; image_0130.png; image_0135.png; image_0149.png)

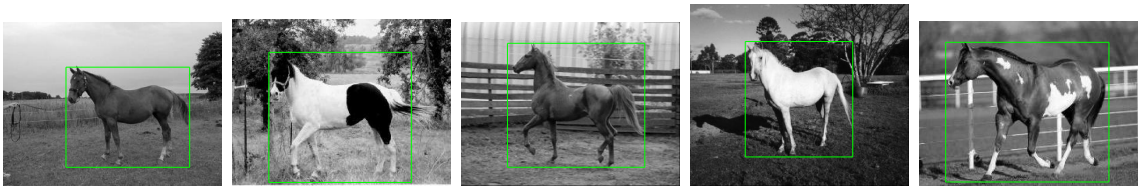


(e) Klasse “Cow” (cow-pic122-sml-lt.png; cow-pic131-sml-lt.png; cow-pic171-sml-lt.png; cow-pic182-sml-lt.png; cow-pic280-sml-lt.png)

Abbildung A.1: Einige Beispiele von Trainingsbildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet.



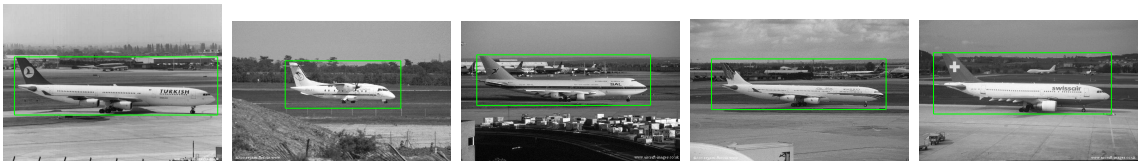
(f) Klasse “Face” (image_0085.png; image_0115.png; image_0258.png; image_0366.png; image_0397.png)



(g) Klasse “Horse” (Horse_horse038.png; Horse_horse042.png; Horse_horse086.png; Horse_horse134.png; Horse_horse135.png)

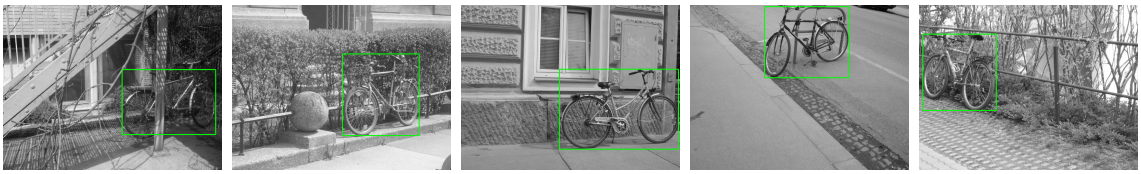
(h) Klasse “Motorbike” (0136.png; 0284.png; 0343.png; 0356.png; 0541.png)

(i) Klasse “Person” (person_048.png; person_073.png; person_080.png; person_090.png; person_101.png)

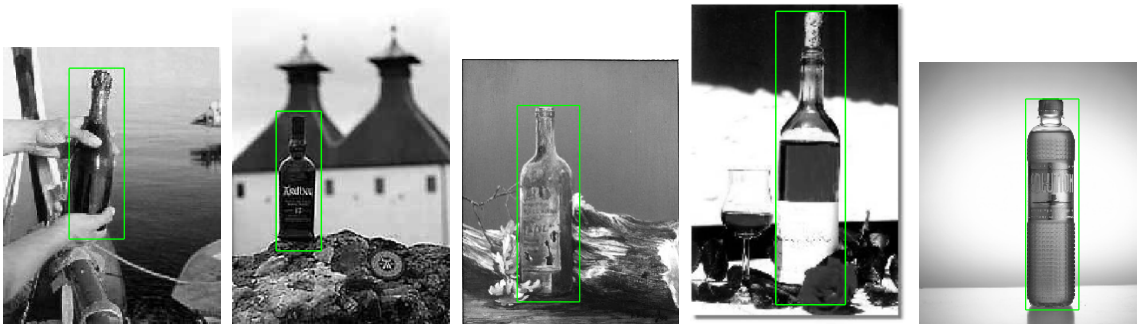


(j) Klasse “Plane” (0201.png; 0329.png; 0344.png; 0370.png; 0505.png)

Abbildung A.1: Einige Beispiele von Trainingsbildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet (Fortsetzung).

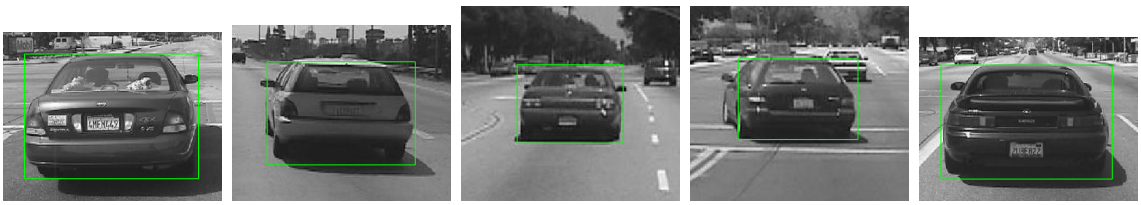


(a) Klasse “Bike” (bike_031.png; bike_037.png; bike_058.png; bike_093.png; bike_108.png)

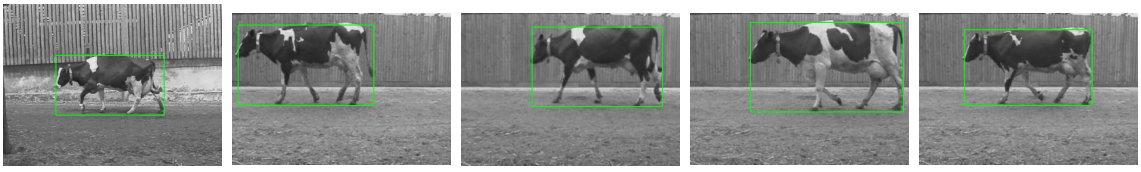


(b) Klasse “Bottle” (Bottle_103.png; Bottle_119.png; Bottle_12.png; Bottle_30.png; Bottle_36.png)

(c) Klasse “Cars front” (CarFront_111_0111.png; carsgraz_023.png; carsgraz_124.png; carsgraz_166.png; carsgraz_271.png)

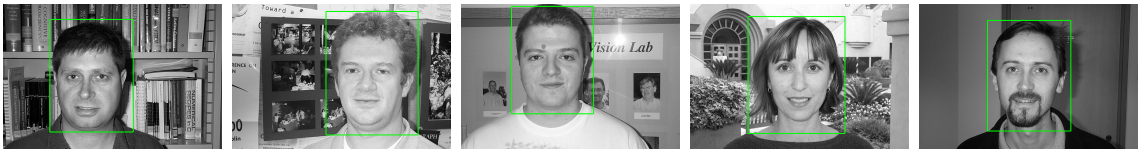


(d) Klasse “Cars rear” (image_0024.png; image_0035.png; image_0044.png; image_0066.png; image_0079.png)

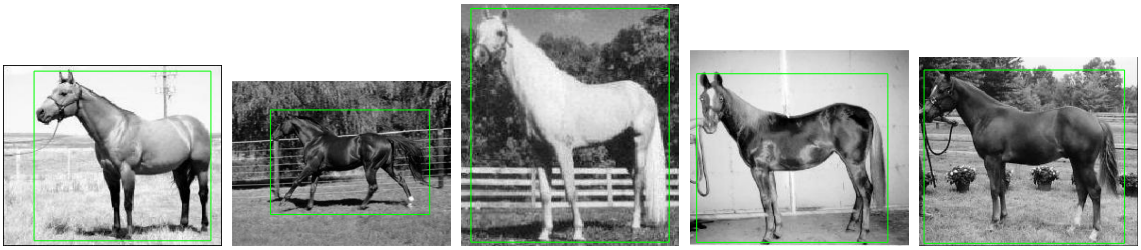


(e) Klasse “Cow” (cow-pic312-sml-lt.png; cow-pic421-sml-lt.png; cow-pic441-sml-lt.png; cow-pic481-sml-lt.png; cow-pic570-sml-lt.png)

Abbildung A.2: Einige Beispiele von positiven Validierungs-Bildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet.



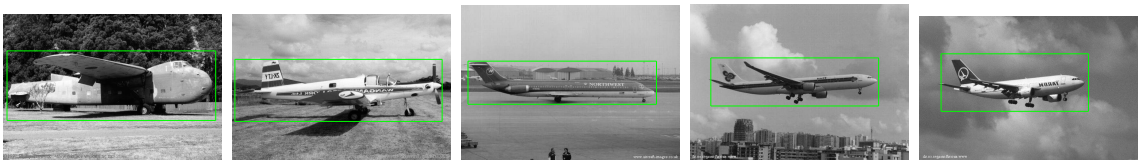
(f) Klasse “Face” (image_0002.png; image_0031.png; image_0048.png; image_0075.png; image_0145.png)



(g) Klasse “Horse” (Horse_horse013.png; Horse_horse025.png; Horse_horse031.png; Horse_horse037.png; Horse_horse043.png)

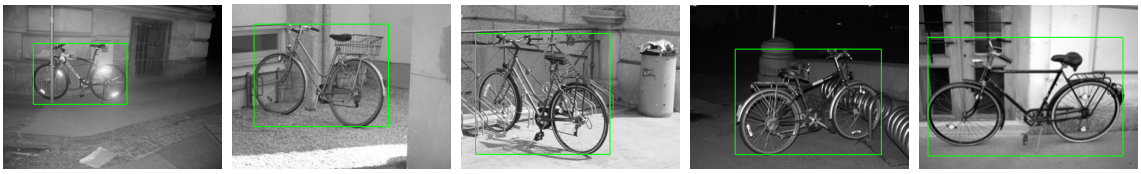
(h) Klasse “Motorbike” (0010.png; 0111.png; 0129.png; 0171.png; 0252.png)

(i) Klasse “Person” (person_047.png; person_056.png; person_078.png; person_095.png; person_115.png)

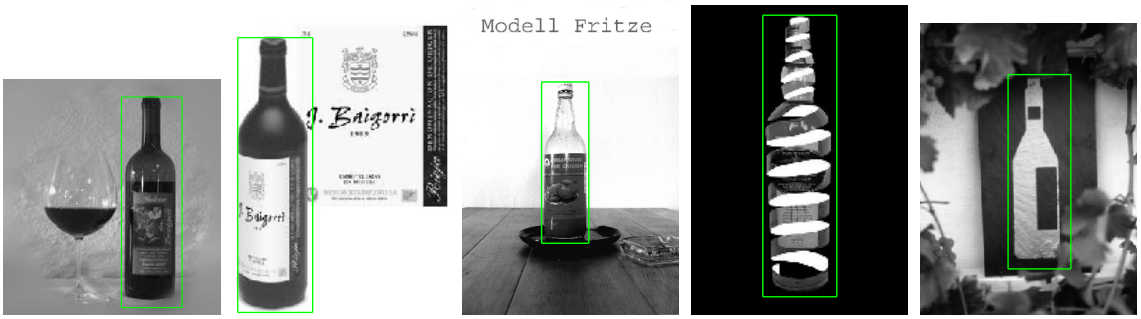


(j) Klasse “Plane” (0034.png; 0096.png; 0144.png; 0229.png; 0285.png)

Abbildung A.2: Einige Beispiele von positiven Validierungs-Bildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet (Fortsetzung).



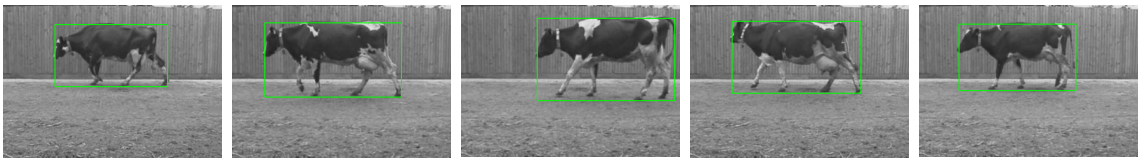
(a) Klasse “Bike” (bike_052_resized.png; bike_113_resized.png; bike_125_resized.png; bike_183_resized.png; bike_208_resized.png)



(b) Klasse “Bottle” (Bottle_108.png; Bottle_26.png; Bottle_41.png; Bottle_5.png; Bottle_74.png)

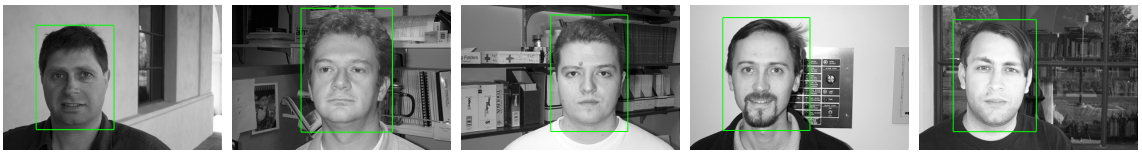
(c) Klasse “Cars front” (CarFront_26_0026_resized_548.png; CarFront_2_0002_resized_509.png; CarFront_43_0043_resized_693.png; carsgraz_005_resized_516.png; carsgraz_149_resized_973.png)

(d) Klasse “Cars rear” (image_0013_resized.jpg; image_0059_resized.jpg; image_0075_resized.jpg; image_0150_resized.jpg; image_0314_resized.jpg)



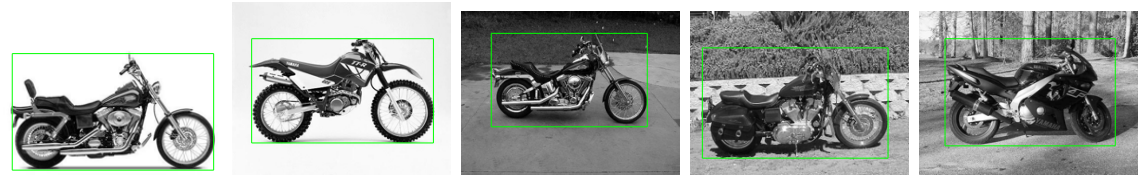
(e) Klasse “Cow” (cow-pic461-sml-lt_resized_93.png; cow-pic571-sml-lt_resized_690.png; cow-pic581-sml-lt_resized_895.png; cow-pic601-sml-lt_resized_309.png; cow-pic631-sml-lt_resized_784.png)

Abbildung A.3: Einige Beispiele von Testbildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet.

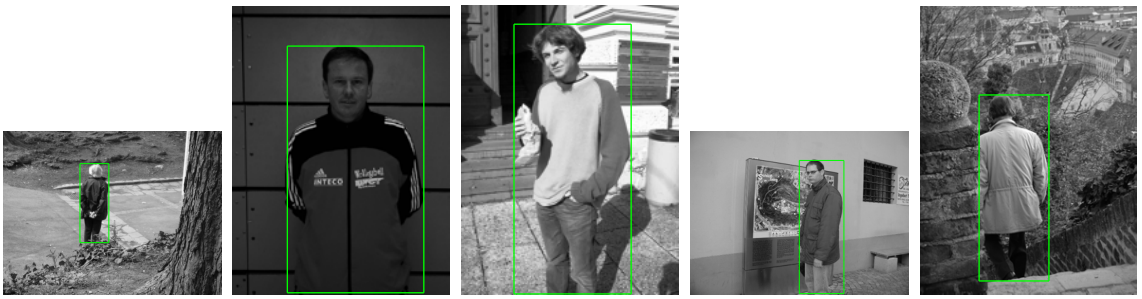


(f) Klasse “Face” (image_0014_resized_965.png; image_0037_resized_916.png; image_0052_resized_679.png; image_0147_resized_251.png; image_0288_resized_513.png)

(g) Klasse “Horse” (Horse_horse048_resized_654.png; Horse_horse075_resized_323.png; Horse_horse085_resized_686.png; Horse_horse104_resized_136.png; Horse_horse140_resized_144.png)



(h) Klasse “Motorbike” (0011_resized_171.png; 0040_resized_865.png; 0206_resized_427.png; 0263_resized_490.png; 0295_resized_136.png)



(i) Klasse “Person” (person_034_resized.png; person_046_resized.png; person_074_resized.png; person_117_resized.png; person_152_resized.png)

(j) Klasse “Plane” (0010_resized_711.png; 0061_resized_582.png; 0073_resized_676.png; 0146_resized_233.png; 0199_resized_674.png)

Abbildung A.3: Einige Beispiele von Testbildern aus der Graz02-MC-DB. Die Bounding-Box um die Objekte ist in grün eingezeichnet (Fortsetzung).

Anhang B

Parameter und Resultate der Experimente

Im Folgenden sind alle relevanten Parameter, die zum Durchführen der einzelnen Experimente benötigt werden, in Tabellen aufgelistet. Des Weiteren sind eventuell noch ergänzende Grafiken zu den Experimenten abgebildet. Eine detaillierte Beschreibung der Experimente wurde bereits im Kapitel 5 ab der Seite 32 durchgeführt und wird an dieser Stelle nicht noch einmal wiederholt. Die Bedeutungen der einzelnen Spalten der Parameter-Tabellen sind in Tabelle B.1 beschrieben.

Spalte	**Beschreibung**
Klasse	Name der Klasse
# seeds	Die Anzahl Startpunkte, die verwendet wird, um BFs aus der Kontur zu extrahieren (siehe Kapitel 3.1 auf der Seite 14)
# incr	Gibt an, wie oft ein BF vergrößert werden soll.
start	Die erste Länge eines BFs in Pixel.
Δ incr	Die Anzahl Pixel, um die das BF in beide Richtungen vergrößert wird.
r_{CV}	Der Unsicherheitsradius des CVs eines BFs.
r_C	Der Unsicherheitsradius des Objektzentrums in den Bildern.
T	Die Anzahl der durchzuführenden Boosting-Schritte (siehe Kapitel 2.2 auf der Seite 7).
α	Gibt den Einfluss der geometrischen Beziehung auf die Auswahl der weak-classifiers an. Mit $\alpha = 0$ hat sie keinen Einfluss und mit $\alpha = 1$ maximalen. Mehr dazu in Kapitel 3.4 auf der Seite 22.
th_{BF}	Threshold, den die Chamfer-Distanz eines BFs auf ein Bild unterschreiten muss, damit das BF als gefunden gilt.
r_{MS}	Radius des Mean-Shift-Kernels, der zum Finden von Detektionszentren verwendet wird.
th_{det}	Threshold, den eine Detektion überschreiten muss, um gültig zu sein.

Tabelle B.1: Die Bedeutung der Spalten in den nachfolgenden Parameter-Tabellen.

Einige Parameter wurden für alle durchgeführten Experimente fixiert. Diese sind:

Die Anzahl Orientierungs-Ebenen beim Chamfer-Matching. In Kapitel 3.3.1 auf der Seite 19 wurde beschrieben, dass das Kantenbild und das BF vor dem Chamfer-

Matching in verschiedene Orientierungs-Ebenen unterteilt wird. Opelt u. a. [2006a] haben bei Ihren Experimenten acht Ebenen verwendet, deshalb werden auch für die Experimente in dieser Arbeit so viel verwendet.

Wie oft kann ein BF im Bild feuern. Der Algorithmus soll auch in der Lage sein, mehrere Instanzen einer Klasse in einem Bild zu finden. Daher ist es notwendig, BFs mehrfach finden zu lassen. Opelt u. a. [2006a] haben immer zehn BFs gesucht und da in dieser Arbeit zehn Klassen getestet werden und alle aus denselben BFs bestehen, werden auch hier die zehn besten[1] Treffer gesucht.

Der Radius zum Clustern von BF-Matches. Um zu vermeiden, dass beim Suchen der besten BF-Matches direkt nebeneinander liegende Minima in der Chamfer-Map als separate Matches gewertet werden, werden die Minima vorher mit Mean-Shift geclustert (vgl. Kapitel 3.3.2 und 4.1 auf den Seiten 20 bzw. 28). Der Radius des Mean-Shift-Kernels wurde durch Experimente ermittelt und mit 5 fixiert.

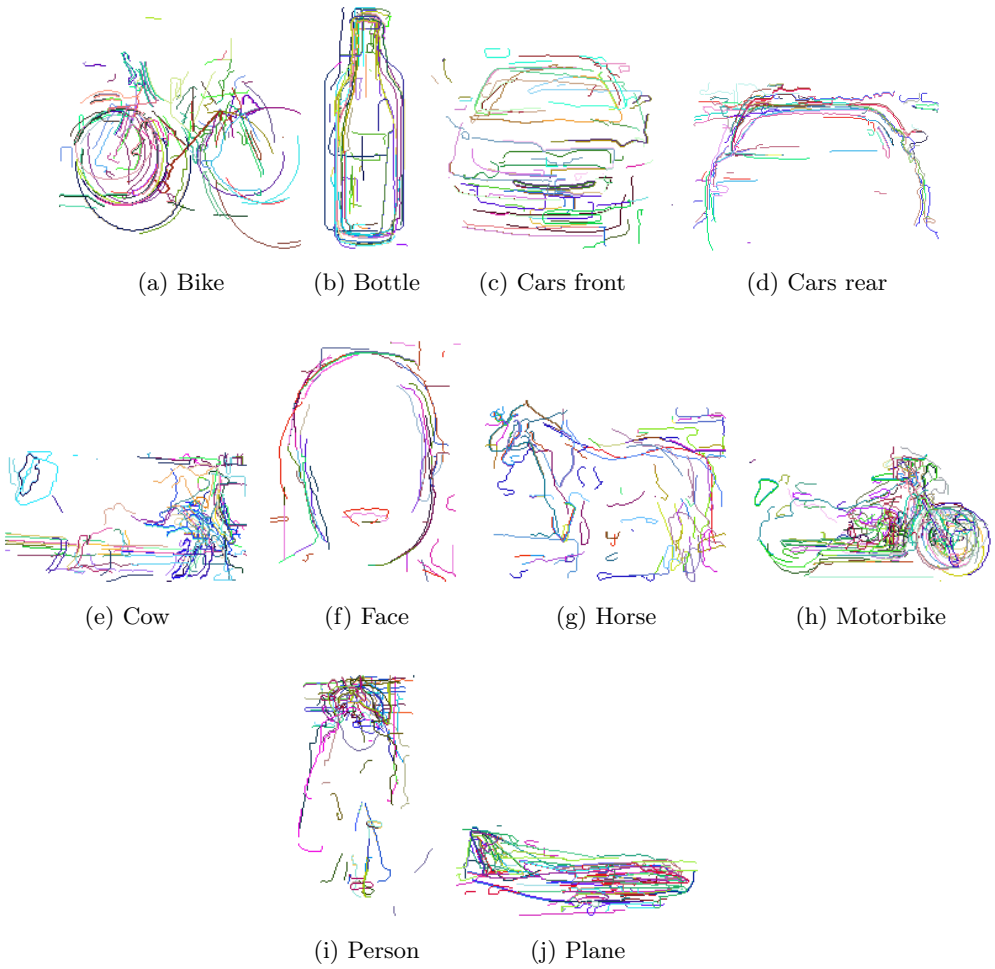
B.1 Experiment 20100511

Dieses Experiment wurde mit der reimplementierten Version des BFMs durchgeführt und dient als Referenz für den Vergleich mit dem neuen Algorithmus.

Klasse	**Vorverarbeitung**				**Lernen**			**Detect**	
	# seeds	# incr	start	Δ incr	r_{CV}	r_C	T	r_{MS}	th_{det}
Bike	20	4	50	30	15	15	100	15	0
Bottle	20	4	50	30	15	15	100	15	0
Cars front	20	4	50	30	15	15	100	15	0
Cars rear	20	4	50	30	15	15	100	15	0
Cow	20	4	50	30	15	15	100	15	0
Face	20	4	50	30	15	15	100	15	0
Horse	20	4	50	30	15	15	100	15	0
Motorbike	20	4	50	30	15	15	100	15	0
Person	20	4	50	30	15	15	100	15	0
Plane	20	4	50	30	15	15	100	15	0

Tabelle B.2: Parameter des Experimentes 20100511. Die Bedeutungen der Spalten sind in Tabelle B.1 auf Seite 57 beschrieben.

[1] Vorausgesetzt es gibt zehn Treffer, die beim Chamfer-Matching unter dem eingestellten Threshold liegen (vgl. Kapitel 5.4.1 auf Seite 39).



(a) Bike (b) Bottle (c) Cars front (d) Cars rear

(e) Cow (f) Face (g) Horse (h) Motorbike

(i) Person (j) Plane

Abbildung B.1: Die weak-classifier der mit der Reimplementierung trainierten Klassifikatoren des Experimentes 20100511. Die BFs eines weak-classifiers sind in derselben Farbe dargestellt. Die Skalierung ist bei allen Modellen dieselbe.

B.2 Experiment 20100614

Mit diesem Experiment wurde ein optimaler Wert für den Threshold th_{BF} bestimmt.

Klassen	**Vorverarbeitung**				**Lernen**			**Detect**			
	# seeds	# incr	start	Δ incr	r_{CV}	α	T	th_{BF}	# BF-M	r_{MS}	th_{det}
Bike, Bottle, Cars front, Cars rear, Cow, Face, Horse, Motorbike, Person, Plane	5	3	50	30	20	0.30	100	2.5, 3, 3.5, 4, 5, 6, 7, 8	10	20	0

Tabelle B.3: Parameter des Experimentes 20100614. Die Bedeutungen der Spalten sind in Tabelle B.1 auf Seite 57 beschrieben.

B.3 Experiment 20100818

Dieses Experiment soll die Auswirkung der geometrischen Beziehung zwischen den CVs der weak-classifiers zeigen.

Klassen	**Vorverarbeitung**				**Lernen**			**Detect**		
	# seeds	# incr	start	Δ incr	r_{CV}	α	T	th_{BF}	r_{MS}	th_{det}
Bike, Bottle, Cars front, Cars rear, Cow, Face, Horse, Motorbike, Person, Plane	5	3	30	30	15	0, 0.3, 0.5	100	7	15	0

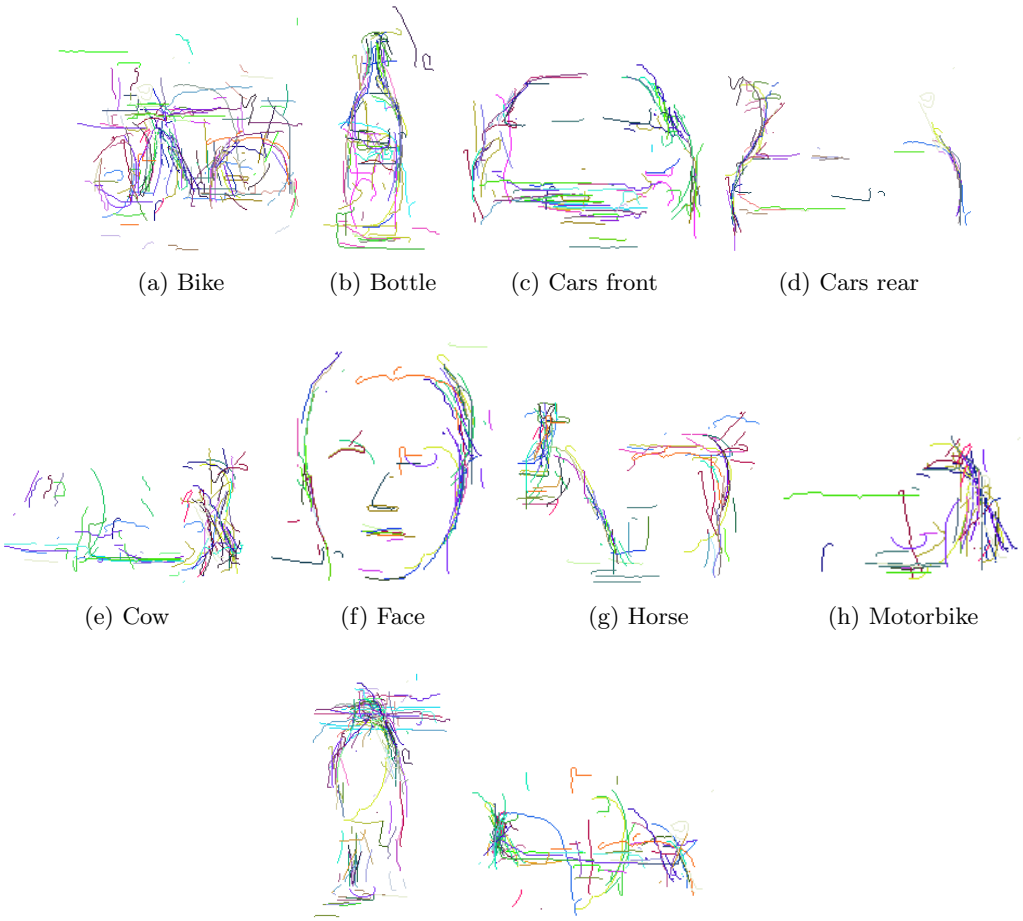
Tabelle B.4: Parameter des Experimentes 20100818. Die Bedeutungen der Spalten sind in Tabelle B.1 auf Seite 57 beschrieben.

B.4 Experiment 20100824

Dieses Experiment dient dem Vergleich mit dem Referenzexperiment. Anhand der Ergebnisse dieses Experiments werden die Vor- und Nachteile des neuen Algorithmus diskutiert.

Klassen	**Vorverarbeitung**				**Lernen**			**Detect**		
	# seeds	# incr	start	Δ incr	r_{CV}	α	T	th_{BF}	r_{MS}	th_{det}
Bike, Bottle, Cars front, Cars rear, Cow, Face, Horse, Motorbike, Person, Plane	5	3	30	30	15	0.30	100	7	15	0

Tabelle B.5: Parameter des Experimentes 20100824. Die Bedeutungen der Spalten sind in Tabelle B.1 auf Seite 57 beschrieben.



(a) Bike (b) Bottle (c) Cars front (d) Cars rear

(e) Cow (f) Face (g) Horse (h) Motorbike

(i) Person (j) Plane

Abbildung B.2: Die weak-classifier des Klassifikators des Experimentes 20100824. Die BFs eines weak-classifiers sind in derselben Farbe dargestellt. Die Skalierung ist bei allen Modellen dieselbe.

Anhang C

DVD

Auf der Innenseite des Rückdeckels ist eine DVD mit allen Daten, die zur Erstellung dieser Arbeit benötigt wurden, zu finden. Sollte die DVD nicht vorhanden sein, können die Daten auch direkt von Professor Pinz (`axel.pinz@tugraz.at`) bezogen werden. Um die Experimente zu starten, gehen Sie bitte wie in der Datei `readme.txt` beschrieben ist, vor.

Inhalt der DVD:

- Sourcecode in Matlab
- Bild-Datenbank
- Daten zum Durchführen der Experimente
- Anleitung zum Durchführen der Experimente (`readme.txt`)
- Literatur

Abkürzungsverzeichnis

BF Boundary-Fragment (siehe Kapitel 2.1 auf der Seite 6)

BFM Boundary-Fragment-Model (siehe Kapitel 2.1 auf der Seite 6)

CV Centroid-Vote (siehe Kapitel 2.1 auf der Seite 6)

EMT Institut für Elektrische Meßtechnik und Meßsignalverarbeitung

RPC Recall-Precision-Curve (siehe Kapitel 5.2 auf der Seite 33)

RPCeER RPC-equal-Error Rate (siehe Kapitel 5.2 auf der Seite 33)

VMG Vision-based Measurement Group

Literaturverzeichnis

[Adelson u. a. 1984] Adelson, E. H. ; Anderson, C. H. ; Bergen, J. R. ; Burt, P. J. ; Ogden, J. M.: 1984, Pyramid methods in image processing. In: *RCA Engineer* 29 (1984), Nr. 6, S. 33–41

[Agarwal u. Roth 2002] Agarwal, Shivani ; Roth, Dan: Learning a Sparse Representation for Object Detection. In: *European Conference on Computer Vision* (2002)

[Alt u. a. 2001] *Kapitel* Matching Polygonal Curves with Respect to the Fréchet Distance. In: Alt, Helmut ; Knauer, Christian ; Wenk, Carola: *Symposium on Theoretical Aspects of Computer Science*. Springer, 2001, S. 63–74

[Banksy 2007] `http://www.farleex.com/wordpress/2007/09/18/genie-des-humors`

[Borgefors 1988] Borgefors, Gunilla: Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10 (1988), S. 849–865. – ISSN 0162–8828

[Cheng 1995] Cheng, Yizong: Mean Shift, Mode Seeking, and Clustering. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (1995), Nr. 8, S. 790–799

[Freund 1995] Freund, Yoav: Boosting a weak learning algorithm by majority. In: *Information and Computation* 121 (1995), Nr. 2, S. 256–285

[Freund u. Schapire 1990] Freund, Yoav ; Schapire, Robert E.: A Short Introduction to Boosting. In: *Journal of Japanese Society for Artificial Intelligence* 14 (1990), S. 771–780

[Freund u. Schapire 1997] Freund, Yoav ; Schapire, Robert E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *Journal of Computer and System Sciences* 55 (1997), Nr. 1, S. 119–139

[Grauman u. Darrell 2004] Grauman, Kristen ; Darrell, Trevor: Fast Contour Matching Using Approximate Earth Mover's Distance. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2004)

[Huttenlocher u. a. 1993] Huttenlocher, Daniel P. ; Klanderman, Gregory A. ; Rucklidge, William J.: Comparing Images Using the Hausdorff Distance. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), Nr. 9, S. 850–863

[Leibe u. Schiele 2004] Leibe, Bastian ; Schiele, Bernt: Scale-Invariant Object Categorization using a Scale-Adaptive Mean-Shift Search. In: *Symposium of the German Association for Pattern Recognition* (2004)

[Lin u. Liu 2005] Lin, Yen-Yu ; Liu, Tyng-Luh: Robust Face Detection with Multi-Class Boosting. In: *IEEE Conference on Computer Vision and Pattern Recognition* 1 (2005), S. 680–687

[Lowe 1985] Lowe, David G.: *Perceptual Organization and Visual Recognition.* Norwell, MA, USA : Kluwer Academic Publishers, 1985

[Lowe 1999] Lowe, David G.: Object Recognition from Local Scale-Invariant Features. In: *Proceedings of the International Conference on Computer Vision*, 1999, 1150–1157

[Marr 1982] Marr, David: *Vision. A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman and Company, 1982

[Mikolajczyk u. a. 2006] Mikolajczyk, Krystian ; Leibe, Bastian ; Schiele, Bernt: Multiple Object Class Detection with a Generative Model. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2006)

[Nistér u. Stewénius 2006] Nistér, David ; Stewénius, Henrik: Scalable Recognition with a Vocabulary Tree. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2006)

[Opelt u. a. 2004] Opelt, Andreas ; Pinz, Axel ; Fussenegger, Michael ; Auer, Peter: Generic object recognition with boosting. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2004)

[Opelt u. a. 2006a] Opelt, Andreas ; Pinz, Axel ; Zisserman, Andrew: A Boundary-Fragment-Model for Object Detection. In: *European Conference on Computer Vision* (2006), S. 575–588

[Opelt u. a. 2006b] Opelt, Andreas ; Pinz, Axel ; Zisserman, Andrew: Incremental learning of object detectors using a visual shape alphabet. In: *IEEE Conference on Computer Vision and Pattern Recognition* 1 (2006), S. 3–10

[Schapire 1990] Schapire, Robert E.: The strength of weak learnability. In: *Machin Learning* 5 (1990), Nr. 2, S. 197–227

[Schapire u. Singer 1998] Schapire, Robert E. ; Singer, Yoram: Improved Boosting Algorithms Using Confidence-rated Predictions. (1998)

[Shotton u. a. 2004] Shotton, Jamie ; Blake, Andrew ; Cipolla, Roberto: Contour-Based Learning for Object Detection. In: *IEEE International Conference on Computer Vision* (2004)

[Shotton u. a. 2009] Shotton, Jamie ; Winn, John ; Rother, Carsten ; Criminisi, Antonio: TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. In: *International Journal of Computer Vision* 81 (2009), January, Nr. 1, S. 2–23

[Stark u. a. 2009] Stark, Michael ; Goesele, Michael ; Schiele, Bernt: A Shape-Based Object Class Model for Knowledge Transfer. In: *IEEE International Conference on Computer Vision* (2009), S. 373–380

[Torralba u. a. 2004] Torralba, Antonio ; Murphy, Kavin P. ; Freeman, William T.: Sharing features: efficient boosting procedures for multiclass object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2004)

[Torralba u. a. 2006] *Kapitel* Shared Features for Multiclass Object Detection. In: Torralba, Antonio ; Murphy, Kevin P. ; Freeman, William T.: *Toward Category-Level Object Recognition*. 2006

[Veltkamp 2001] Veltkamp, Remco C.: Shape Matching: Similarity Measures and Algorithms. In: *International Conference on Shape Modeling and Applications* 0 (2001), S. 188–197. ISBN 0–7695–0853–7

[Zhu u. a. 2006] Zhu, Ji ; Rosset, Saharon ; Zou, Hui ; Hastie, Trevor: Multi-class AdaBoost. (2006)