STATISTISCHE MODELLIERUNG VON BILDINHALTEN FÜR DIE BILDERKENNUNG

DIPLOMARBEIT AM LEHRSTUHL FÜR INFORMATIK VI
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND NATURWISSENSCHAFTEN
RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

GUTACHTER: PROF. DR.-ING. H. NEY, PROF. DR. L. KOBBELT
BETREUER: DIPL.-INFORM. D. KEYSERS

VORGELEGT VON: MICHAEL MOTTER MATR.NUMMER: 211 109

Inhalt

In dieser Arbeit wird ein ganzheitliches Modell vorgestellt, um Objekte in komplexen Bildszenen zu klassifizieren und damit einen objektbasierten Bildzugriff zu ermöglichen. Dieses Modell beruht auf einem statistischen, erscheinungsbasierten Ansatz, in dem alle Pixel einer Bildszene mit dem Modell erklärbar sind. Die beschriebenen Verfahren stellen einen neuen Ansatz dar, erscheinungsbasiert Objekt- und Hintergrundmodell miteinander zu kombinieren.

Basierend auf diesem Ansatz wurde ein Verfahren zum automatischen Objekttraining und zur Objektklassifikation entwickelt und implementiert. Mit dem Verfahren zum automatischen Objekttraining können Prototypen für Objekte, die in komplexen Bildszenen vorkommen, trainiert werden. Der Klassifikator ermöglicht es, Objekte in komplexen Bildszenen zu lokalisieren und einer Klasse zuzuordnen.

Um diese Verfahren zu testen, wurde eine neue Bilddatenbank erstellt, weil bis zum Zeitpunkt dieser Arbeit keine Datenbank existierte, die automatisches Objekttraining und Objektklassifikation mit realem Hintergrund in sich vereinte. Die erzeugte Datenbank ermöglicht es, neue Ansätze auf diesem Gebiet systematisch in verschiedenen Schwierigkeitsstufen zu testen und zu bewerten.

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, im Dezember 2001

Michael Motter

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Das Interesse an digitalen Bildern ist in den letzten Jahren enorm gewachsen. Durch die rapide Zunahme der Prozessorgeschwindigkeiten und der Speichermedien ist man in der Lage, immer schneller eine hohe Anzahl von digitalen Bildern zu erzeugen. Zum Beispiel wird im Bereich der Computer-Tomographie bei jedem Messvorgang eine Fülle von Bildern erzeugt, die in digitaler Form abgespeichert werden. Selbst moderne Röntgengeräte liefern heutzutage Bilder direkt in digitaler Form. In privaten Haushalten erfreut sich die digitale Kamera immer größerer Beliebtheit, deren Bilder in digitaler Form auf dem heimischen PC gespeichert werden.

Gleichzeitig wächst die Popularität des Internets (*World Wide Web*, WWW), durch das sich sehr einfach Daten wie digitale Bilder austauschen lassen. Dadurch ist jedem Benutzer, egal ob privat oder beruflich, die Möglichkeit gegeben, Bilder im Internet zur Verfügung zu stellen oder sich Bilder aus dem Internet zu beschaffen. Es entstehen große Bilddatenbanken, die unstrukturiert über viele Computer verteilt abgespeichert sind. Diese Bilddatenbanken zu strukturieren, so dass Bilder mit bestimmten Inhalten schnell gefunden werden können, ist Aufgabe der Forschungsbereiche *Image Retrieval*, Bildindizierung, Bildklassifikation, *Data Mining* und *Computer Vision*. Die Anwendungsgebiete lassen sich grob in 3 Kategorien einteilen [21].

In der einen Kategorie besteht nur eine vage Vorstellung von dem Bild, das man sucht. Es wird ein grobes Muster, z.B. ein Beispielbild, vorgegeben, und das System soll alle Bilder suchen, die es mit dieser Vorgabe assoziiert [2, 10, 13]. Solch ein System könnte z.B. die Diagnose eines Arztes erleichtern, der auf einem Röntgenbild einen Tumor vermutet. Er gibt dem System dieses Röntgenbild als Vorlage und erhält eine Menge von Bildern zurück, auf denen ähnliche Anzeichen zu erkennen sind. Mit Hilfe dieser Bilder und den zugehörigen vorhandenen Diagnosen kann der Arzt eine Entscheidung für seinen jetzigen Patienten treffen.

Eine weitere Kategorie ist die präzise Suche nach einem Muster. Es wird ein Bild vorgegeben, und das System soll dieses Bild in einer Datenbank wiederfinden [4, 9]. Typische Anwendungsgebiete sind Authentifizierungssysteme. Ein solches System entscheidet z.B. anhand eines Fingerabdrucks, ob einer Person der Zutritt in einen Sicherheitsbereich gewährt werden soll. Es sucht in einer Datenbank, ob der Fingerabdruck der Person dort gespeichert ist.

Ein drittes Anwendungsgebiet ist die automatische Kategorisierung von Bildern. Bilder werden automatisch in vorgegebene Klassen einsortiert [20, 23]. Ein Bild, das neu in die Datenbank aufgenommen werden soll, wird dann automatisch in eine der Klassen einsortiert. Man könnte sich vorstellen, dass ein System z.B. ein neues Röntgenbild automatisch in eine der Klassen „Schädel“, „Rumpf“ oder „Extremitäten“ einordnet.

All diese Anwendungen haben eine Gemeinsamkeit. Sie basieren auf Verfahren aus dem Forschungsgebiet der Mustererkennung. Das „Muster“, welches erkannt werden soll, ist ersteinmal

nicht näher definiert. Es kann ein Sonnenuntergang oder die rote Kravatte einer Person sein. In dieser Arbeit werden diese „Muster“ Objekte genannt. Dies können Bauklötze, Sparschweine, Autos oder ähnliches sein.

Damit ein Objekt in einer Bildszene wiedergefunden werden kann, muss für das Objekt ein Prototyp erstellt werden. Dieser Prototyp muss charakteristisch für das Objekt sein. Möchte man z.B. Gesichter erkennen, dann muss dieser Prototyp Merkmale eines Gesichts, wie Augen, Nase und Mund, in sich vereinen. Die Erstellung eines Prototyps wird durch ein Training erzielt. Im Training wird aus einer Menge von Beispielbildern/Trainingsdaten ein Prototyp erzeugt.

Um zu überprüfen, wie gut ein Prototyp trainiert wurde, wird noch eine Menge von Beispielbildern/Testdaten benötigt. Diese enthalten nicht nur Gesichter sondern z.B. auch Autos und Schiffe. Für jedes Testbild wird gezählt, wie oft mit Hilfe dieses Prototyps ein Gesicht richtig erkannt wird. Aus der Anzahl der richtigen und falschen Klassifikationen errechnet sich die Fehlerrate.

Bis zum jetzigen Zeitpunkt wird in der Objekterkennung und Klassifikation der Schwerpunkt auf die Beschreibung bzw. Modellierung der Objekte selber gelegt. Dies setzt voraus, dass für das Training eines Prototyps eines Objekts das Objekt bekannt sein muss. Bilddaten, die ein Objekt enthalten, müssen im Vorfeld manuell segmentiert werden, um das Objekt von seinem Umfeld zu trennen. Dabei geht jede Kontextinformation verloren. Ausnahmen bildet hier die Handschrifterkennung, die die Information berücksichtigt, mit welchen Buchstaben zusammen der zu erkennende Buchstabe auftritt. In die Modellierung fließt demnach ausschließlich die Objektinformation ein, ohne den Hintergrund, auf dem sich das Objekt befindet, zu berücksichtigen. Dies kann zu unerwünschten Effekten führen, die für Fehlklassifikationen verantwortlich sind, wie sie in Abbildung 1.1 gezeigt werden.

Abbildung 1.1: In dieser Abbildung sind Fehlklassifikationen zu sehen, die daraus resultieren, dass der Hintergrund bei der Klassifikation nicht berücksichtigt wurde[12].

Diese Art von Fehler könnte ein Klassifikator vermeiden, wenn er Kontextinformation, z.B. den Hintergrund, bei der Entscheidung berücksichtigen würde. Im Fall von USPS[1] ist a-priori bekannt, dass die Ziffern auf schwarzem, homogenem Hintergrund liegen. Wenn ein Klassifikator eine Hypothese, wie in Abbildung 1.1, betrachten würde, bliebe oberhalb des Ausschnitts eine weiße Pixelreihe übrig, die dem Hintergrund zugeordnet würde. Diese weiße Reihe passt aber nicht zu der Annahme, dass es sich um einen homogenen, schwarzen Hintergrund handeln muss. Daher ist es unwahrscheinlich, dass es sich bei dieser Hypothese um das gesuchte Objekt handelt. Eine insgesamt bessere Bewertung würde eine Hypothese erhalten, die die „9“ komplett umfassen und damit ausschließlich schwarze Pixel im Hintergrund auftreten lassen würde.

Ein Ziel dieser Arbeit ist es, geeignete Modelle für solchen Hintergrund zu trainieren und zu testen, wie diese sich auf die Klassifikation auswirken.

[1] USPS (United States Postal Service) ist eine Datensammlung aus isolierten, handgeschriebenen Ziffern, die unter `ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data` frei zur Verfügung steht.

Eine weitere Problematik ist das Objekttraining selber. Wie bereits erwähnt, findet das Objekttraining bisher weitgehend auf manuell vorverarbeiteten Daten statt, so dass gewährleistet ist, dass die Objektinformation direkt zum Training verwendet werden kann. Die Vorverarbeitung kann z.B. durch Segmentierung geschehen, bei der die Objekte von Hand ausgeschnitten werden, wie die Ziffern der USPS-Datenbank, oder durch Positionsangaben, an denen sich das Objekt im Bild befindet. In beiden Fällen ist ein hoher manueller Aufwand notwendig, um die Daten richtig aufzubereiten.

Aus dieser Problematik ergibt sich ein weiteres Ziel dieser Arbeit. Nur mit dem Wissen, dass ein Objekt in allen Trainingsdaten vorhanden ist, soll das Objekt richtig lokalisiert und trainiert werden. Ein Verfahren, das diese Aufgabe löst, wäre somit in der Lage, eine Bildszene automatisch in Hintergrund und Objekt zu segmentieren, so dass das Objekt vom Hintergrund getrennt wird. Für diese Vorgehensweise ist die Hintergrundinformation ebenso wichtig wie das Objekt selber, so dass auf Algorithmen der Objektklassifikation mit Hintergrundmodell zurückgegriffen werden kann. Dieser Ansatz ist in gewisser Weise mit einem Problem aus der automatischen Spracherkennung vergleichbar. Auch dabei ist im Training zunächst lediglich bekannt, dass eine bestimmte Wortfolge gesprochen wurde, jedoch nicht, wann die einzelnen Wörter anfangen bzw. enden.

Im Zusammenhang mit der Bearbeitung dieser Aufgabe ergibt sich noch ein weiteres Problem, ohne dessen Lösung keine Bewertung der aufgestellten Algorithmen vorgenommen werden kann. Es fehlt eine geeignete Datenbank zum Trainieren und Testen der Modelle. Ein weiteres Ziel war demnach, eine brauchbare Datenbank zu finden, mit der automatisches Objekttraining und Klassifikation mit Hintergrundinformation durchgeführt werden kann. Es stellte sich im Verlauf der Arbeit heraus, dass eine solche Datenbank offenbar nicht existiert. Die wenigen Gruppen, die an einer ähnlichen Thematik arbeiten, haben ihre eigenen Datenbanken generiert, jedoch nicht immer allgemein zur Verfügung gestellt. Dies verdeutlicht, dass die Forschung auf diesen Gebieten noch in den Anfängen steckt. Andererseits gibt es aber ohne eine standardisierte Datenbank keine Möglichkeit, die unterschiedlichen Verfahren der einzelnen Gruppen miteinander zu vergleichen. Aus diesem Grund wurde im Rahmen dieser Arbeit eine Datenbank generiert, mit dem Ziel, diese frei zur Verfügung zu stellen und erste Ergebnisse auf dieser Datenbank zu präsentieren.

Kapitel 2

Datenbanken

Ein wesentlicher Bestandteil, um die Qualität von neuen Methoden in der Objektklassifikation zu bewerten, ist eine geeignete Bilddatenbank. Standardisierte Bilddatenbanken für Multiobjekterkennung und automatisches Objekttraining sind bis zum jetzigen Zeitpunkt nicht verfügbar. Gruppen, die sich mit diesen oder ähnlichen Themen befassen, erzeugen fast ohne Ausnahme ihre eigenen Bilddatenbanken. Diese sind speziell auf die Problemstellungen der Gruppe angepasst und lassen sich schlecht auf andere Probleme übertragen.

Einen Haupteinstiegspunkt für Bilddatenbanken aller Art findet man auf der Homepage „*Computer Vision*“ [1]. Diese Homepage wurde seit 1994 an der Carnegie Mellon Universität aufgebaut, um eine zentrale Anlaufstelle für Forschungsgruppen auf dem Gebiet „Computer Vision“ zu schaffen. „Computer Vision“ bezeichnet ein Teilgebiet der KI und der Bildverarbeitung, in dem Bildverstehen anhand realer Bilddaten erforscht wird.

In den folgenden Abschnitten wird erläutert, warum die in [1] referenzierten Bilddatenbanken nur zu einem sehr geringen Anteil für die Problemstellung in dieser Arbeit von Nutzen sind. Aus diesem Grund wurde im Rahmen dieser Arbeit eine Bilddatenbank erzeugt, mit dem Ziel, diese zu veröffentlichen, so dass andere Gruppen diese Datenbank verwenden und Vergleichsergebnisse erzeugen können. Schließlich werden die verwendeten Datenbanken kurz vorgestellt.

2.1 Datenbanken auf „Computer Vision“

Wie schon erwähnt, ist es bis zum jetzigen Zeitpunkt nicht möglich gewesen, eine Bilddatenbank zu finden, die für automatisches Objekttraining und Multiobjekterkennung geeignet ist. Ein zentraler Anlaufpunkt für die Suche war die Homepage von „*Computer Vision*“ [1]. Dort sind Verweise von ca. 70 Bilddatenbanken angegeben, von denen in Tabelle 2.1 ein Teil dokumentiert ist. In Anhang B ist die vollständige Liste beschrieben. Zusammenfassend ergibt sich aus den untersuchten Datenbanken folgendes Bild. Forschungsgruppen haben für ihre speziellen Aufgaben jeweils eine eigene Datenbank erzeugt, diese jedoch nicht öffentlich zugänglich gemacht, so dass diese für Tests im Rahmen dieser Arbeit nicht in Frage kommen. Datenbanken, die für „Image Retrieval“ erzeugt wurden, sind nicht in Trainings- und Testdaten unterteilt, so dass sich die Güte der Klassifikatoren nicht messen lässt. Diese Gruppen verlassen sich auf den subjektiven, visuellen Eindruck, ob geeignete Bilder gefunden wurden. Vielfach sind diese Datenbanken auch zu klein, so dass erscheinungsbasierte Objekterkennung kaum möglich ist. Mit z.B. 40 Bildern lassen sich die Parameter in einem 1024-dimensionalen Raum nicht sinnvoll schätzen. Ein Großteil der Datenbanken wurde für andere Aufgabenstellungen geschaffen. So gibt es Datenbanken, die

Tabelle 2.1: Diese Tabelle enthält eine Teilmenge der auf der Homepage von 'Computer Vision' gefundenen Datenbanken.

Homepage	**Inhalt / Bemerkung**
`xtreme.gsfc.nasa.gov/`	atmosphärische Aufnahmen / keine Klassifikationsaufgabe
`www.cs.waikato.ac.nz/~singlis`	Faxe / keine Klassifikationsaufgabe
`www.cs.cmu.edu/afs/cs.cmu.edu`	Stereodaten / Datenbank ungeeignet
`www.vasc.ri.cmu.edu/idb`	Gesichtsaufnahmen / ungeeignet
`www.vision.caltech.edu`	Objekte vor realem Hintergrund / brauchbar, aber nicht erhältlich
`www.cs.columbia.edu/CAVE/curet`	Texturen / ungeeignet
`www.cs.sfu.ca/~colour`	Bilder für Farbanalysen / keine Klassifikationsaufgabe
`www.cs.washington.edu/research/imagedatabase/groundtruth`	Image Retrieval / zu wenig Bilder und keine klar definierbaren Objekte
`vision.psych.umn.edu/www/kersten-lab/demos`	3D-Daten / keine Klassifikationsaufgabe
`earthrise.sdsc.edu`	Satellitenaufnahmen / keine Klassifikationsaufgabe
`www.gastrointestinalatlas.com`	medizinische Aufnahmen / Atlas und keine Klassifikationsaufgabe
`bias.csr.unibo.it/fvc2000`	Fingerabdrücke / ungeeignet

Fingerabdrücke, Faxe, Stereobilder usw. enthalten, die ebenfalls für die hier verfolgten Zwecke ungeeignet sind.

Nach der Durchforschung all dieser und vieler weiterer Links wurde keine für diese Arbeit geeignete Datenbank gefunden. Dies ist die Motivation, eine neue Datenbank zu erzeugen und diese öffentlich zur Verfügung zu stellen.

2.2 Zur Verfügung stehende Datenbanken

2.2.1 `ERLANGEN`-Datenbank

Am Lehrstuhl für Mustererkennung an der Universität Erlangen-Nürnberg wurde eine Datenbank bestehend aus 5 Klassen generiert, um Experimente mit heterogenem Hintergrund und Überdeckung durchzuführen. Zum Training stehen 2 Korpora mit jeweils einem Umfang von 85 Trainingsdaten zur Verfügung. Die Objekte wurden über einem homogenen, schwarzen Hintergrund und um die eigenen Achse gedreht aufgenommen. Die Kamera ist senkrecht über den Objekten fixiert worden, so dass die Objekte in den Bildern in 2-dimensionalen Rotationen vorliegen. Der einzige Unterschied der beiden Korpora sind die unterschiedlichen Beleuchtungsquellen bei der Aufnahme der Objekte. Während der erste Korpus mit gleicher Beleuchtung aufgenommen wurde, wurden beim zweiten Aufnahmen mit unterschiedlichen Lichtquellen verwendet. In Abbildung 2.1 sind einige Beispiele der einzelnen Klassen zu sehen.

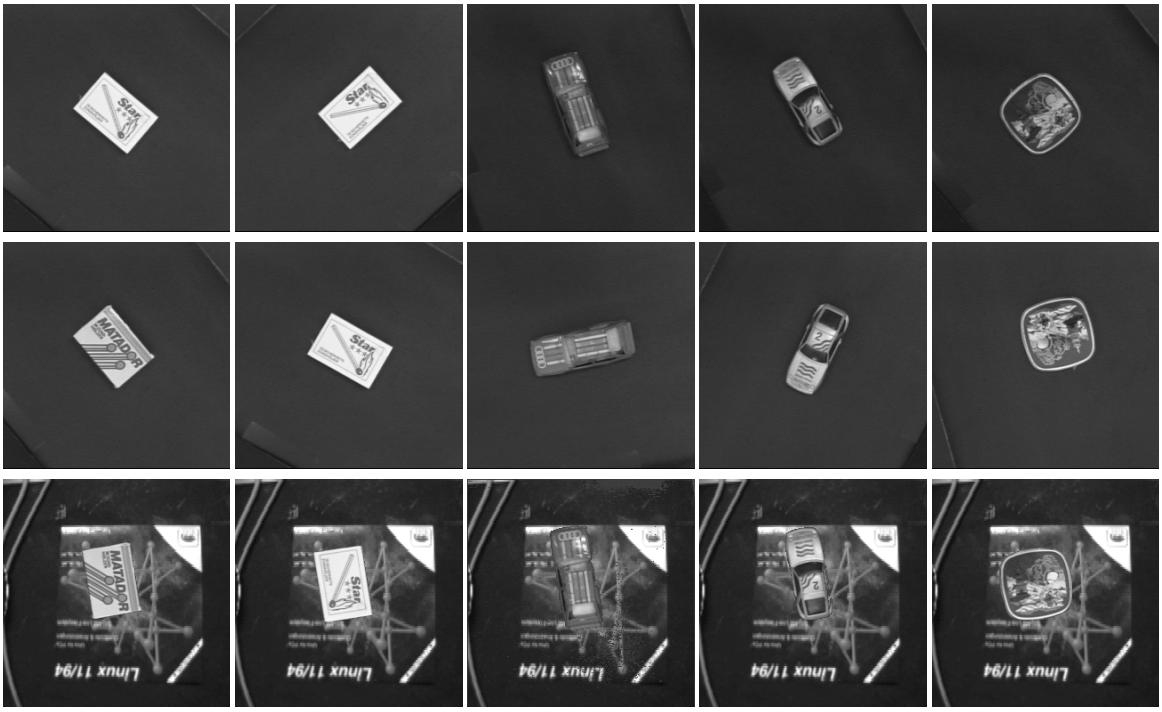


Abbildung 2.1: In der Abbildung sind die 5 Klassen der Datenbank `ERLANGEN` zu sehen. In der 1. Zeile befinden sich Trainingsobjekte, die mit einer einheitlichen Beleuchtung aufgenommen wurden. In der 2. Zeile sind die Trainingsobjekte zu sehen, die mit Beleuchtung von verschiedenen Lichtquellen aufgenommen wurden und in der 3. Zeile befinden sich Testobjekte auf einem Hintergrundbild.

Zum Testen gibt es 6 verschiedene Korpora mit drei verschiedenen Kategorien. Zu klassifizieren sind Objekte auf heterogenem Hintergrund, Objekte auf homogenem Hintergrund mit 25% Überdeckung durch andere Objekte und Objekte auf homogenem Hintergrund mit 50% Überdeckung. Diese 3 Gruppen wurden wieder in zwei Beleuchtungsklassen eingeteilt. Hierbei ist anzumerken, dass die Verdeckungen und der heterogene Hintergrund den Objekten nachträglich hinzugefügt wurden, so dass diese Objekte nicht in ihrem realen Kontext aufgenommen wurden. Ebenso taucht der Hintergrund nicht in den Trainingsdaten auf, so dass diese Datenbank nur teilweise zu gebrauchen ist. Zu bemerken ist noch, dass der Hintergrund alleine zum Training zur Verfügung steht und in den Testbildern immer identisch ist. Hinzu kommt noch die Tatsache, dass Objekt und Hintergrund eine unterschiedliche Auflösung haben, was je nach der Wahl der Merkmale (z.B. *Wavelets*) ein Vorteil sein kann, weil in den Objektmerkmalen hohe Frequenzen häufiger auftreten.

Als Vergleichsergebnisse werden die in [18, 19] veröffentlichten Resultate genommen.

2.2.2 `IRMA`-Bilddatenbank

Die `IRMA`-Bilddatenbank (*Image Retrieval for Medical Applications*) [5] wurde zur Entwicklung eines *Image-Retrieval*-Systems aufgebaut und umfasst bisher 1617 Röntgenbilder, die in 6 Klassen untergliedert sind. Von den Röntgenbildern sind 110 Abdomen-, 706 Extremitäten-, 103 Brust-, 110 Schädel-, 410 Becken- und 178 Wirbelsäulenaufnahmen. Die Bilder liegen in 256 Graustufen vor und besitzen eine Auflösungsspannweite von 200x200 bis 2000x2000 Pixeln. Zusätzlich gibt es einen Testkorpus, der ein Gesamtvolumen von 332 Bildern enthält.

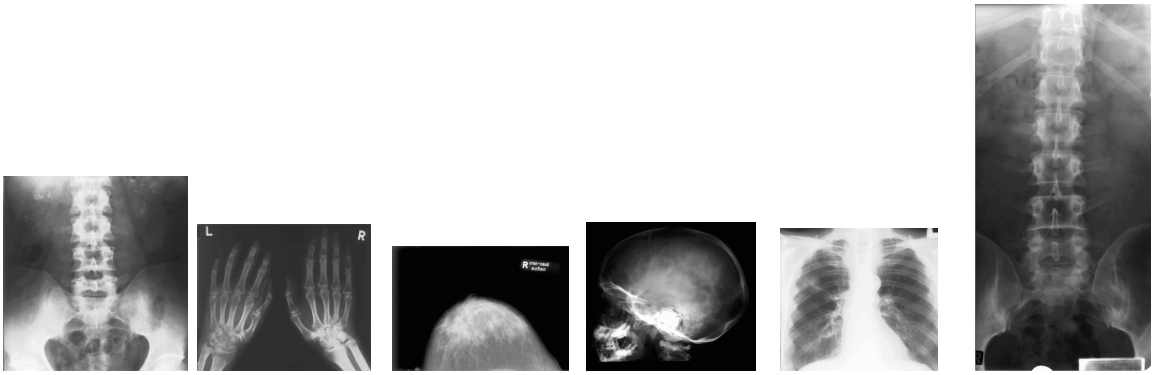


Abbildung 2.2: Die 6 Klassen aus der IRMA-Datenbank

Beispiele der einzelnen Klassen sind in Abbildung 2.2 zu sehen. In den Veröffentlichungen [6, 5] werden Klassifikationsergebnisse auf dieser Datenbank präsentiert, die als Vergleichsergebnisse herangezogen werden können.

2.2.3 COIL-20-Datenbank

Columbia Object Image Library (COIL-20) ist eine Datenbank von Grauwertbildern, bestehend aus insgesamt 20 verschiedenen Objektklassen (siehe Abbildung 2.3). Sie wurde erzeugt, um Experimente zu dem von Murase und Nayar in [15] dargestellten Verfahren durchzuführen.

Abbildung 2.3: In der 1. Zeile sind Beispiele der 5 Objektklassen aus dem COIL-20-Testkorpus zu sehen. Die Bilder darunter zeigen die 20 Objektklassen aus dem Trainingskorpus.

Die Objekte wurden auf einem motorisierten Drehteller vor einen schwarzen Hintergrund gestellt. Der Drehteller wurde schrittweise um 360 Grad gedreht, und alle 5 Grad wurde eine Aufnahme von jedem Objekt gemacht. Daraus erhält man 72 Bilder pro Objektklasse. Aus diesen Bildern wurden zwei Datenbanken erstellt. Die eine ist ein Testkorpus („*unprocessed images*“), der 360 Bilder aus 5 Objektklassen enthält. Die andere ist ein Trainingskorpus („*processed images*“), der aus 1440 Bildern aus 20 Objektklassen besteht.

Aus der Datenbankbeschreibung ist nicht ersichtlich, ob sich die Kameraeinstellungen bei den Aufnahmen der Trainings- und Testbilder unterscheiden. Um sicherzustellen, dass sich der Trainingskorpus von dem Testkorpus unterscheidet, wurden in dieser Arbeit die Korpora so aufgeteilt

(vgl. [12]), dass Trainings- und Testaufnahmen jeweils zu anderen Winkeleinstellungen gehören. Für das Training wurden die Bilder mit den ungeraden Winkeleinstellungen und für das Testen die geraden Winkeleinstellungen verwendet. Diese Selektion reduziert den Trainingskorpus auf 720 und den Testkorpus auf 180 Bilder. Als Vergleichsergebnisse werden die in [12] präsentierten Ergebnisse herangezogen.

2.3 Generierung einer neuen Datenbank: `COIL-I6`

Aus Gründen, wie sie in Abschnitt 2.1 und Anhang B beschrieben sind, ist es notwendig, eine eigene Datenbank zu generieren, um die in dieser Arbeit aufgestellten Modelle zu trainieren und zu testen.

Weiterhin wurde darauf geachtet, dass nicht nur eine Trainings- und Testmenge generiert wird, die alle möglichen Features in sich vereinen, sondern es wurden verschiedene Korpora erzeugt. Diese Korpora ermöglichen es, auf verschiedenen Schwierigkeitsstufen zu experimentieren und damit die Effekte z.B. von realem und homogenem Hintergrund unabhängig voneinander zu untersuchen. In einer beiliegenden Datei werden die Anzahl der Objekte, Skalierung, Rotation und Position der Objekte im jeweiligen Bild aufgeführt. Dadurch kann das Training und Testen zusätzlich erleichtert werden, indem Information über die vorhandenen Abbildungen dem Trainingsalgorithmus oder Klassifikator hinzugefügt wird. Dies ermöglicht es, die einzelnen Effekte der Abbildungen getrennt voneinander zu betrachten.

Als einfachste Schwierigkeitsstufe ist der originale `COIL-20` Trainings- und Testdatensatz anzusehen. Auf Basis dieser Datenbank (siehe Abschnitt 2.2.3) wurden für das automatische Objekttraining 2 Trainingskorpora erzeugt und für das Testen 6 Korpora, die in den folgenden Abschnitten beschrieben werden.

2.3.1 Trainingskorpora: `COIL-I6-Train(x)`

Die Trainingsobjekte in der `COIL-20`-Datenbank liegen in segmentierter Form und auf schwarzem Hintergrund vor. Für das Objekttraining ist dies eine sehr einfache Darstellung der Objekte. Diese Art von Daten findet man in der Regel dann vor, wenn die Bilder künstlich erzeugt oder manuell vorverarbeitet wurden. Mit den Trainingskorpora `COIL-I6-Train1` und `COIL-I6-Train2` werden erste Schritte in eine realistischere Darstellung der Objekte angeboten.

Auf die vorher segmentierten Objekte wurden für beide Trainingsmengen zufällig gleichverteilt folgende Operationen angewendet:

- zufällige Translationen
- zufällige 2-dimensionale Rotationen (0-360 Grad)
- zufälliges Skalieren (60-100%) der Achsen unter Einhaltung des Seitenverhältnisses

Die Auflösung der Trainingsdaten ist für alle Bilder gleich und beträgt 192x192 Pixel.

Insgesamt wurden 5760 Trainingsdaten je Korpus erzeugt, so dass von jedem Objekt jede Winkelaufnahme im Durchschnitt 8 mal in den Daten auftaucht. Der Unterschied beider Korpora liegt in der Wahl des Hintergrundes. In der ersten Datenbank `COIL-I6-Train1` wurde ein homogener, schwarzer Hintergrund gewählt. Beispiele aus diesem Korpus sind in Abbildung 2.4 zu sehen. Für das Training schwieriger ist die zweite Datenbank, `COIL-I6-Train2`, in der die Objekte auf einem realen Hintergrund positioniert wurden. Beispiele aus diesem Korpus sind in Abbildung 2.5 zu sehen.

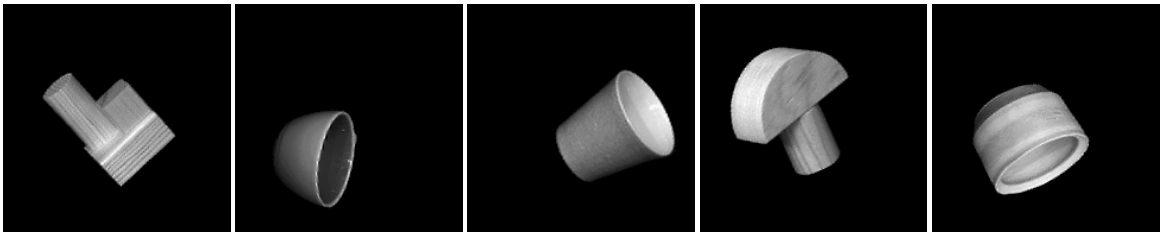


Abbildung 2.4: Beispiele von Bildern aus COIL-I6-Train1.

Abbildung 2.5: Beispiele von Bildern aus COIL-I6-Train2.

2.3.2 Testkorpora: COIL-I6-Test(x)

Ähnlich wie die Trainingskorpora basieren die Testkorpora auf dem COIL-20 Testkorpus (siehe Abschnitt 2.2.3). Es wurden insgesamt 6 Korpora erzeugt, die als verschiedene Schwierigkeitsstufen aufgefasst werden können. Die Operationen, die auf die Objekte angewandt wurden, sind die gleichen wie bei den Trainingskorpora:

- zufällige Translationen
- zufällige 2-dimensionalen Rotationen (0-360 Grad)
- zufälliges Skalieren der Achsen (60-100%) unter Einhaltung des Seitenverhältnisses

In der folgenden Tabelle ist angegeben, welche Testkorpora erzeugt wurden.

	Hintergrund	
Anzahl der Objekte pro Bildszene	**schwarz**	**real**
1 Objekt	COIL-I6-Test1	COIL-I6-Test2
0-2 Objekte ohne Überlagerung	COIL-I6-Test3	COIL-I6-Test4
0-2 Objekte mit Überlagerung	COIL-I6-Test5	COIL-I6-Test6

Insgesamt wurden 180 Testbilder pro Korpus kreiert. Die Bilder haben eine Größe von 448×336 Pixeln. In den Fällen, in denen sich Objekte überlagern dürfen, was bei den letzten beiden Korpora der Fall ist, beträgt die Überlagerung nicht mehr als 25%. Beispiele der einzelnen Testkorpora sind in Abbildung 2.6 zu sehen. Dort werden zeilenweise von oben nach unten Beispiele aus den einzelnen Testmengen gezeigt.

2.4 Zusammenfassung

Insgesamt stehen für diese Arbeit 3 Datenbanken[1] zur Verfügung. Die Datenbanken, die vor Beginn dieser Arbeit erhältlich waren, sind für andere Ziele generiert worden, so dass lediglich

[1] Die Datenbanken COIL-20 und COIL-I6 werden als eine Datenbank aufgefasst.

Abbildung 2.6: In der Abbildung sind Beispiele aus der COIL-I6-Test-Datenbank aufgelistet. In den Zeilen von oben nach unten sind Beispiele aus den jeweiligen Testkorpora zu sehen.

für die Klassifikatoren Vergleichsergebnisse vorliegen. Für das automatische Objekttraining gibt es keine vergleichbaren Ergebnisse.

Der Nachteil aller zur Verfügung stehenden Korpora ist der unrealistische Kontext, in dem die Objekte erscheinen. Entweder liegen sie vor einem homogen schwarzen Hintergrund oder sie wurden nachträglich vor einen inhomogenen Hintergrund gesetzt. Im Fall der ERLANGEN-Datenbank ist dies eine Zeitschrift und im Fall von COIL-I6 sind dies Aufnahmen in Innenräumen oder auf dem Universitätsgelände. Eine Datenbank aufzubauen, in der die Objekte in einem realen Kontext liegen, ist aus zeitlichen Gründen im Rahmen dieser Arbeit nicht möglich gewesen.

Ein Vorteil der künstlich erzeugten Daten ist die Option zur Trennung der Effekte. Dadurch, dass Rotation, Größe und Position der Objekte bekannt sind, lässt sich untersuchen, wie sich diese Abbildungen auf das Training und die Klassifikation auswirken.

Kapitel 3

Statistische Modellierung und Mustererkennung

Dieses Kapitel beschreibt die Klassifikatoren und die zugehörigen statistischen Modelle, die in dieser Arbeit benutzt werden. Dabei wird zuerst der Aufbau eines Erkenners beschrieben, um zu zeigen, in welchem Kontext ein Klassifikator verwendet wird. Da die Modellierung von Objektinhalten direkt im Training und im Klassifikator zum Tragen kommt, werden diese Komponenten des Erkenners in dieser Arbeit genauer beschrieben. Auf das Training wird detaillierter im 4. Kapitel eingegangen.

Von einer allgemeinen Beschreibung eines Klassifikators ausgehend, werden spezielle Klassifikatoren hergeleitet, die im Rahmen dieser Arbeit verwendet werden. Eine ausführliche Herleitung der verwendeten Klassifikatoren für die Einzelobjekterkennung lässt sich in [8, 11] nachlesen.

3.1 Klassifikatoren für Einzelobjekterkennung

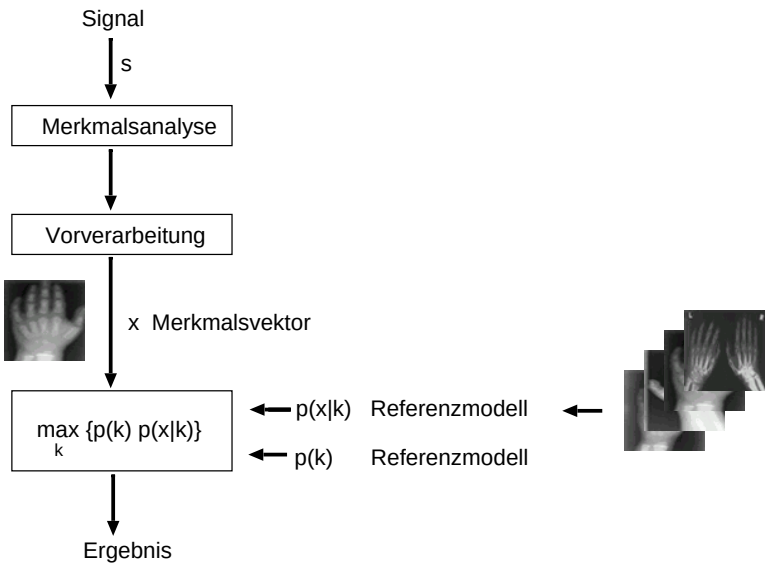


Abbildung 3.1: In dieser Abbildung ist der allgemeine Aufbau eines Einzelobjekterkenners dargestellt.

Betrachtet man ein $K-$Klassen-Problem, dann soll ein Erkenner in der Lage sein, ein beobachtetes Signal s in die richtige Klasse $k = 1 \ldots K$ einzuordnen. In Abbildung 3.1 ist der Aufbau eines Erkenners schematisch dargestellt.

Im ersten Schritt wird das eingehende Signal s (z.B. die Ausgabe einer Kamera) vorverarbeitet und anschließend eine Merkmalsanalyse, $s \mapsto \mathbf{x}$, durchgeführt. Der resultierende Merkmalsvektor $\mathbf{x} \in \mathbb{R}^D$ wird als Eingabe einer Entscheidungsfunktion genommen und einer Klasse zugeordnet.

In dieser Arbeit wurde für die Klassifikation ein erscheinungsbasierter Ansatz gewählt. Das bedeutet, dass die Bilddaten direkt als Merkmalsvektoren verwendet werden. Zum Beispiel werden bei Grauwertbildern die Werte der einzelnen Pixel als Merkmale verwendet. Damit fällt der Schritt der Merkmalsanalyse, wie sie in Abbildung 3.1 zu sehen ist, weg. Es sei jedoch erwähnt, dass eine Merkmalsanalyse zusätzlich zu den Pixelwerten verwendet werden kann, um die Qualität der Klassifikation zu verbessern. Hierzu bieten sich zum Beispiel lokale Texturmerkmale oder Gradienten an.

Ein Klassifikator wird durch eine Entscheidungsfunktion $r : \mathbb{R}^D \longrightarrow \{1 \ldots K\}$ beschrieben, die jedem Merkmalsvektor eine Klasse zuordnet. Möchte man das Risiko einer Fehlentscheidung unter Annahme einer 0-1-Kostenfunktion minimieren, erhält man die *Bayes'sche Entscheidungsregel [8, S.26ff]*. Dabei bedeutet die Verwendung der 0-1-Kostenfunktion, dass nur Klassifikationsfehler gezählt werden. Wird ein Klassifikationsfehler gemacht, wird dies mit den Kosten 1 bestraft. Ist die Klassifikation richtig, so wird sie mit 0 bewertet.

Die Bayes'sche Entscheidungsregel entscheidet sich für die Klasse mit der höchsten *posteriori-Wahrscheinlichkeit* $p(k|\mathbf{x})$. Da die *posteriori-Wahrscheinlichkeit* in der Praxis auf direktem Wege meist schwer zu ermitteln ist, führt man diese mittels der $\mathrm{Bayes'schen\ Regel}$ auf die *a-priori*-Wahrscheinlichkeit $p(k)$ und die *klassenbedingte Wahrscheinlichkeit* $p(\mathbf{x}|k)$ zurück.

$$p(k|\mathbf{x}) \overset{Bayes}{=} \frac{p(\mathbf{x}|k)p(k)}{p(\mathbf{x})} \tag{3.1}$$

Die klassenbedingte und *a-priori*-Wahrscheinlichkeiten sind in der Praxis leichter zu ermitteln. Man erhält durch einfaches Umformen den folgenden Ausdruck für die *Bayes'sche Entscheidungsregel*:

$$\begin{aligned} r(\mathbf{x}) &= \underset{k}{\text{argmax}} \left\{ p(k|\mathbf{x}) \right\} \\ &\overset{(3.1)}{=} \underset{k}{\text{argmax}} \left\{ \frac{p(k)p(\mathbf{x}|k)}{p(\mathbf{x})} \right\} \\ &= \underset{k}{\text{argmax}} \left\{ p(k)p(\mathbf{x}|k) \right\} \end{aligned} \tag{3.2}$$

In der letzten Umformung kann die Verteilung $p(\mathbf{x})$ vernachlässigt werden, da sie unabhängig von k ist. Sind die wahren *a-priori*- und klassenbedingten Verteilungen bekannt (damit auch die *posteriori*-Verteilung), so lässt sich zeigen [8, 16], dass die zu erwartende Fehlerrate minimal ist. In der Praxis sind diese Verteilungen in der Regel nicht bekannt. Die hohe Kunst in der Mustererkennung besteht darin, die realen Daten durch eine möglichst passende Modellannahme, $\hat{p}(k)$, $\hat{p}(\mathbf{x}|k)$, zu beschreiben.

Solange nicht explizit anders erwähnt, wird als Modell für die *a-priori*-Wahrscheinlichkeit $\hat{p}(k)$ eine Gleichverteilung angenommen. Der Klassifikator vereinfacht sich dann zu folgendem Ausdruck:

$$r(\mathbf{x}) \quad = \quad \underset{k}{\text{argmax}}\{p(\mathbf{x}|k)\} \tag{3.3}$$

In den nächsten Abschnitten werden die verwendeten Modelle der klassenbedingten Wahrscheinlichkeitsfunktionen $\hat{p}(\mathbf{x}|k)$ genauer beschrieben.

3.1.1 Gauß'sche Verteilung für die Einzelobjekterkennung

Ein einfaches, aber häufig verwendetes Modell ist die multivariate Gauß-Verteilung.

$$\hat{p}(\mathbf{x}|k) \quad = \quad \frac{1}{\sqrt{|2\pi\mathbf{\Sigma}_k|}} \cdot \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big) \quad \sim \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \tag{3.4}$$

Bei dieser Modellannahme müssen insgesamt $\frac{d(d+1)}{2}$ Parameter für die Kovarianzmatrix geschätzt werden. Ein Problem kann die Inverse der Kovarianzmatrix werden, die bei zu wenig Trainingsdaten nicht richtig geschätzt und leicht singulär werden kann.

Aus diesen Gründen und um die Anzahl der Parameter zu reduzieren, wird bei allen folgenden Modellen eine über alle Komponenten „gepoolte" Varianz verwendet. Das bedeutet, dass $\mathbf{\Sigma}_k = \sigma_k^2\mathbf{I}$ gesetzt wird, und es ergibt sich folgendes Modell:

$$\hat{p}(\mathbf{x}|k) \quad = \quad \frac{1}{\sqrt{(2\pi\sigma_k^2)^D}} \cdot \exp\big(-\frac{1}{2\sigma_k^2}\cdot \parallel \mathbf{x}-\boldsymbol{\mu}_k \parallel^2\big) \quad \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2\mathbf{I}) \tag{3.5}$$

Für dieses Modell muss nur noch 1 Parameter für die Kovarianzmatrix geschätzt werden. Das Training der Parameter wird in Kapitel 4 beschrieben.

3.1.2 Gauß'sche Mischverteilung für die Einzelobjekterkennung

Mit einer einfachen Gauß'schen Modellannahme ist man in der Lage, unimodal verteilte Zufallsgrößen zu beschreiben. In der Praxis sind die Verteilungen jedoch komplizierter, so dass unimodale Beschreibungen nicht mehr ausreichen. Wenn z.B. ein Objekt in verschiedenen Rotationen vorliegt, dann lassen sich mit einer unimodalen Verteilung nicht alle Rotationswinkel erfassen. Sie kann bestenfalls eine Objektansicht mit festem Rotationswinkel oder nur kleinen Rotationen beschreiben. Eine bessere Beschreibung für die klassenbedingten Wahrscheinlichkeitsverteilungen $\hat{p}(\mathbf{x}|k)$ sind dann Gauß'sche Mischverteilungen (GMV). Im Gegensatz zu einer einfachen Gauß-Verteilung handelt es sich bei der GMV um eine multimodale Beschreibung der Daten, die jede Verteilung beliebig genau approximieren kann [16]. Die GMV ist eine Linearkombination aus einzelnen Gauß-Verteilungen, die unterschiedlich gewichtet werden.

$$\hat{p}(\mathbf{x}|k) \quad = \quad \sum_{i=1}^{I_k} c_{ik}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{ik}, \mathbf{\Sigma}_{ik}) \quad \text{und} \quad \sum_{i=1}^{I_k} c_{ik} = 1 \tag{3.6}$$

Unter der Annahme der „gepoolten" Varianz folgt (hier auch über alle Dichten „gepoolt", d.h. $\sigma_{ik}^2 = \sigma_k^2$):

$$\hat{p}(\mathbf{x}|k) \quad = \quad \sum_{i=1}^{I_k} c_{ik}\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{ik}, \sigma_k^2\mathbf{I}) \quad \text{und} \quad \sum_{i=1}^{I_k} c_{ik} = 1 \tag{3.7}$$

Wenn man z.B. ein Objekt nimmt, das verschieden rotiert vorliegt, dann kann für die Beschreibung jedes Rotationswinkels eine Dichte verwendet werden.

Neben den Parametern $\{\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}\}$ bzw. $\{\boldsymbol{\mu}_{ik}, \sigma_k^2\mathbf{I}\}$ der Dichten gibt es noch die Gewichte, die als Parameter geschätzt werden müssen. Eine effiziente Einstellung der Parameter lässt sich mit dem *Expectation Maximization*-Algorithmus (EM-Algorithmus) mit integriertem „Clustering" berechnen. Eine Beschreibung dieses Algorithmus wird in Kapitel 4 gegeben.

3.1.3 Kernel-Densities für Einzelobjekterkennung

Möchte man keine Annahme über die Verteilung der Daten machen und stehen nicht viele Trainingsdaten zur Verfügung, dann kann man auf folgendes, nicht parametrische Verfahren zurückgreifen. Aus den GMVen lassen sich die *Kernel-Densities* (KD, *Kernschätzverfahren*) als Spezialfall herleiten. Für jede GMV einer Klasse setzt man die Mittelwerte jeder Dichte gleich einem Trainingsdatum.

Seien $\mathbf{x}_{1k}, \ldots, \mathbf{x}_{N_k k}$ mit $x \in \mathbb{R}^D$ die Trainingsdaten der k-ten Klasse, dann ergibt sich aus (3.7):

$$\hat{p}(\mathbf{x}|k) \quad = \quad \sum_{n=1}^{N_k} c_{nk}\mathcal{N}(\mathbf{x}|\mathbf{x}_{nk}, \sigma_k^2\mathbf{I}) \tag{3.8}$$

Mit der Annahme, dass jedes Trainingsdatum gleich wahrscheinlich ist, folgt:

$$\hat{p}(\mathbf{x}|k) \quad = \quad \frac{1}{N_k}\sum_{n=1}^{N_k} \mathcal{N}(\mathbf{x}|\mathbf{x}_{nk}, \sigma_k^2\mathbf{I}) \tag{3.9}$$

Jedes Trainingsdatum liefert explizit einen Beitrag bei jeder Klassifikation. Wie stark dieser Beitrag ist, hängt von der gewählten Varianz ab. Die Varianz gibt an, wie stark der Einfluss des Trainingsdatums auf seine Umgebung im Merkmalsraum ist.

3.1.4 Nächste-Nachbar-Regel für die Einzelobjekterkennung

Die *Nächste-Nachbar-Regel* (NN,*Nearest Neighbor*) lässt sich als Vereinfachung aus den KD herleiten. Dabei wird eine über alle Klassen „gepoolte" Varianz $\sigma_k^2 := \sigma^2$ vorausgesetzt. Anstelle der Linearkombination der Einzelverteilungen wird die *Maximum-Approximation* verwendet. Die *a-priori*-Klassenverteilung entspricht den relativen Häufigkeiten. Dann erhält man folgende Schätzer für die Modelle:

$$\begin{aligned}
\hat{p}(k) \quad &= \quad \frac{N_k}{N} \quad \text{mit} \quad N = \sum_{k=1}^{K} N_k \\
\hat{p}(\mathbf{x}|k) \quad &= \quad \frac{1}{N_k}\sum_{n=1}^{N_k} \frac{1}{\sqrt{(2\pi\sigma^2)^D}} \cdot \exp\big(-\frac{1}{2\sigma^2} \cdot \|\mathbf{x} - \mathbf{x}_{nk}\|\big)
\end{aligned}$$

Damit ergibt sich für den Klassifikator:

$$
\begin{aligned}
r(\mathbf{x}) &\overset{3.2}{=} \operatorname*{argmax}_k \Big\{\hat{p}(k)\hat{p}(\mathbf{x}|k)\Big\} \\
&= \operatorname*{argmax}_k \Big\{\frac{N_k}{N}\cdot\frac{1}{N_k}\sum_{n=1}^{N_k}\frac{1}{\sqrt{(2\pi\sigma^2)^D}}\cdot\exp\big(-\frac{1}{2\sigma^2}\cdot\|\mathbf{x}-\mathbf{x}_{nk}\|^2\big)\Big\} \\
&= \operatorname*{argmax}_k \Big\{\sum_{n=1}^{N_k}\exp\big(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}_{nk}\|^2\big)\Big\} \\
&\approx \operatorname*{argmax}_k \Big\{\max_{n=1,\dots,N_k}\exp\big(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}_{nk}\|^2\big)\Big\} \quad \text{(Maximum-Approximation)} \\
&= \operatorname*{argmin}_k \Big\{\min_{n=1,\dots,N_k}\|\mathbf{x}-\mathbf{x}_{nk}\|^2\Big\} \quad (\exp(-\mathbf{x}) \text{ monoton fallend}) \qquad (3.10)
\end{aligned}
$$

Diese aus den KD hergeleitete Approximation heißt Nächste-Nachbar-Regel, weil sie für jede Beobachtung $\mathbf{x}$ diejenige Klasse k wählt, zu der der nächste Nachbar unter den Trainingsdaten $\mathbf{x}_{nk}$ gehört. Als Verfeinerung und Erweiterung der NN-Regel kann man die *k-Nächste-Nachbar-Regel (k-NN)* betrachten. Diese berücksichtigt für die Zuordnung der Beobachtung nicht die Klasse, die zu dem Trainingsvektor mit der geringsten Distanz gehört, sondern betrachtet für die Entscheidung die k geringsten Distanzen[1]. Weiterhin ist zu beachten, dass die NN-Regel von dem gewählten Distanzmaß abhängt. Die einfachste Annahme als Distanzmaß ist die *euklidische Distanz*.

3.2 Klassifikatoren für Multiobjekterkennung ohne Überlagerung

In den Abschnitten zuvor wurden reine Einzelobjektklassifikatoren behandelt. In der Bildverarbeitung bedeutet dies, dass ein Merkmalsvektor genau *ein* Objekt darstellt. In der Multiobjekterkennung hat man als Merkmalsvektor eine Bildszene, die mehrere Objekte enthalten kann. Die Merkmalsvektoren der Objekte müssen demnach erst lokalisiert werden. Hinzu kommt, dass die Objekte nicht immer auf homogenem, schwarzem Hintergrund vorkommen, sondern dass sie in einer realen Umgebung eingebettet sind. Dieser Sachverhalt muss ebenfalls in der Modellierung mitberücksichtigt werden. Eine formelle Beschreibung liefern die folgenden Abschnitte.

Sei $\mathbf{X} \in \mathbb{R}^{IJ}$ mit $I, J \in \mathbb{N}$ eine solche Bildszene, die M unbekannte Objekte der Klassen $k_1 \dots k_M =: k_1^M$ enthalten kann. Da die Objektklassen nicht immer alle Pixel des Bildes abdecken, werden die restlichen Pixel einer Klasse k_0 zugeordnet, die den Hintergrund repräsentiert. Die Bildszene $\mathbf{X}$ lässt sich dann in $M+1$ disjunkte Partitionen einteilen, so dass gilt:

$$
S = S_0^M := \dot{\bigcup}_{m=0}^{M} S_m \quad \text{mit} \quad S_m \subset \big\{(i,j) : i = 0\dots I-1, j = 0\dots J-1\big\} \quad \text{mit } I, J \in \mathbb{N} \tag{3.11}
$$

Ferner sei $p(\mathbf{X}_{S_m}|\boldsymbol{\mu}_k)$ die Modellannahme für ein Objekt der jeweiligen Klasse k, dass ein Bildausschnitt $\mathbf{X}_{S_m} := \big\{x_{ij} : (i,j) \in S_m\big\}$ das Objekt der Klasse k enthält. $\boldsymbol{\mu}_k$ ist der Mittelwertsvektor des zugehörigen Referenzmodells. Um alle zulässigen Transformationen eines Objektes zu erfassen, wird eine Abbildung

$$
\vartheta_m : \boldsymbol{\mu}_{k_m} \mapsto \boldsymbol{\mu}_{k_m}(\vartheta_m) \quad \text{mit} \quad m = 0\dots M
$$

[1] Die Anzahl k der betrachteten Nachbarn ist nicht zu verwechseln mit dem Klassenindex k. Die Bezeichnung „k-NN“ hat sich in der Literatur durchgesetzt.

eingeführt, die auf die Referenzen angewendet wird.

Daraus folgt für den Multiobjekt-Klassifikator als Verallgemeinerung der Bayes'schen Entscheidungsregel auf diesen Fall:

$$r(\mathbf{X}) \quad = \quad \underset{M,k_0^M}{\text{argmax}}\Big\{\max_{\vartheta_0^M}\{p(\vartheta_0^M)\cdot p(k_0^M)\cdot \prod_{m=0}^{M} p(\mathbf{X}_{S_m}|\boldsymbol{\mu}_{k_m}(\vartheta_m)\}\Big\} \tag{3.12}$$

In der Formel 3.12 stellt $p(\vartheta_0^M)$ die *a-priori*-Wahrscheinlichkeit dar, mit der eine Abbildung auf die jeweilige Referenz angewendet wird. Diese wird hier als unabhängig von k_0^M angenommen, d.h. $p(\vartheta_0^M) = p(\vartheta_0^M|k_0^M)$. Die *a-priori*-Wahrscheinlichkeit $p(k_0^M)$ gibt an, mit welcher Wahrscheinlichkeit eine bestimmte Klassenkombination in der Bildszene auftreten kann.

Diese Entscheidungsregel beinhaltet ein komplexes Suchproblem. Alleine die *a-priori*-Verteilung mit festem m lässt sich als geordnete Probe aus $K+1$ Klassen[2] vom Umfang m mit Wiederholung auffassen. Das ergibt $(K+1)^m$ Kombinationsmöglichkeiten, wie die Objekte in der Bildszene auftreten können. Bei einer Anzahl von maximal M möglichen Objekten, die in einer Bildszene auftreten können, müssen $\sum_{m=1}^{M}(K+1)^m$ Kombinationsmöglichkeiten ausgetestet werden.

Weiterhin wird das Maximum über alle Transformationsparameter gesucht. Alle Abbildungen zu betrachten ist unmöglich und auch nicht immer wünschenswert. Soll z.B. die Ziffer „6“ von der „9“ unterschieden werden, hat man es schwer, wenn man Drehungen um einen Winkel von 180 Grad zulässt. Aus diesen Gründen werden Einschränkungen bzgl. der Wahl der Abbildung notwendig sein.

Als ein Spezialfall des Multiobjektklassifikators ist der *Einzelobjektklassifikator mit Hintergrundmodell* anzusehen. Im Gegensatz zum reinen Einzelobjektklassifikator muss das Objekt gleichzeitig lokalisiert und klassifiziert werden. Diese kombinierte Lokalisation und Klassifikation berücksichtigt neben den Referenzmodellen der Objekte ein Referenzmodell für den Hintergrund. Aus der Formel 3.12 lässt sich der *Einzelobjektklassifikator mit Hintergrundmodell* direkt herleiten, wenn folgende Voraussetzungen erfüllt werden:

- die Anzahl der Objekte in der Bildszene ist $M = 1$
- zugelassene Abbildungen für ϑ sind Translationen, Skalierungen und 2-dimensionale Rotationen

Dann gilt:

$$\begin{aligned} r(\mathbf{X}) \quad &\overset{(3.12)}{=} \quad \underset{k}{\text{argmax}}\Big\{\max_{\vartheta_0^1}\{\hat{p}(\vartheta_0^1)\cdot \hat{p}(k)\cdot \prod_{m=0}^{1} \hat{p}(\mathbf{X}_{S_m}|\boldsymbol{\mu}_{k_m}(\vartheta_m))\}\Big\} \\ &= \quad \underset{k}{\text{argmax}}\Big\{\max_{\vartheta_0^1}\{\hat{p}(\vartheta_0^1)\cdot \hat{p}(k)\cdot \hat{p}(\mathbf{X}_{S_0}|\mu_{k_0}(\vartheta_0))\cdot \hat{p}(\mathbf{X}_{S_1}|\boldsymbol{\mu}_{k_1}(\vartheta_1))\}\Big\} \end{aligned} \tag{3.13}$$

In Formel 3.13 stellt $\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0}(\vartheta_0))$ das Modell für den Hintergrund und $\hat{p}(\mathbf{X}_{S_1}|\boldsymbol{\mu}_{k_1}(\vartheta_1))$ das Modell für das Objekt einer Klasse dar. $\hat{p}(k)$ ist die *a-priori*-Wahrscheinlichkeit, mit der die Objektklasse auftritt, und $p(\vartheta_0^1)$ die *a-priori*-Wahrscheinlichkeit der Abbildungen, die auf das Objekt und den Hintergrund angewandt werden.

Als Modellannahmen wurden in dieser Arbeit mehrere Varianten getestet. Für die Objektmodellierung wurden KD, NN, multivariate Gauß-Verteilung und Gauß'sche Mischverteilungen getestet.

[2] Die Klasse, die den Hintergrund repräsentiert, muss mitberücksichtigt werden.

Der Hintergrund wurde mit univariater Gauß-Verteilung, Gleichverteilung und Histogrammen modelliert.

In den folgenden Abschnitten wird beschrieben, wie sich die einzelnen Modelle auf den Klassifikator auswirken.

3.2.1 Einzelobjektklassifikator mit multivariater Gauß-Verteilung und Hintergrundmodell

Eine einfache Modellannahme ist die multivariate Gauß-Verteilung. Sie ist sinnvoll, wenn ein Objekt in genau einer Perspektive vorliegt und man alle Abbildungen, die auf dieses Objekt angewendet werden, in die Modellierung einarbeitet. Hat man als Objekt z.B. eine Streichholzschachtel senkrecht von oben an verschiedenen Positionen fotografiert vorliegen, dann lassen sich die Translationen mittels ϑ_1 ausdrücken. Die inversen Translationen positionieren die Streichholzschachtel in der jeweiligen Abbildung auf eine einheitliche Stelle. An dieser Stelle lässt sich die Streichholzschachtel mit einer einfachen Gauß-Verteilung beschreiben.

Für den Klassifikator ergibt sich:

$$r(\mathbf{X}) = \underset{k}{\text{argmax}}\Bigg\{\max_{\vartheta_0^1}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0})\cdot \frac{1}{\sqrt{|2\pi\sigma_{1k}^2\mathbf{I}|}}\cdot\exp\big(-\frac{1}{2\sigma_{1k}^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_{1k}(\vartheta_1)\|^2\big)\Big\}\Bigg\} \tag{3.14}$$

Man beachte, dass der Normierungsterm $\frac{1}{\sqrt{|2\pi\sigma_{1k}^2\mathbf{I}|}}$ indirekt von ϑ_1 abhängt und deswegen nicht vernachlässigt werden kann. Da ϑ_1 für die Partitionierung der Bildszene verantwortlich ist, ändert sich mit der Partitionierung die Dimension der Referenz und damit die Dimension der Einheitsmatrix $\mathbf{I}$.

3.2.2 Einzelobjektklassifikator mit Gauß'schen Mischverteilungen und Hintergrundmodell

Betrachtet man reale Bilddaten, dann fällt auf, dass auf Objekte nicht nur Translationen angewendet werden, sondern wesentlich komplexere Abbildungen, z.B. unterschiedliche Beleuchtungsverhältnisse. Diese Abbildungen ließen sich zwar direkt in die Modelle integrieren, würden aber die Laufzeiten drastisch in die Höhe treiben. Aus diesem Grund muss man einen Kompromiss finden, welche Abbildungen direkt modelliert werden oder welche gelernt werden sollen. Sollen die Abbildungen gelernt werden, müssen sie zum einen in den Trainingsdaten auftreten, zum anderen muss die Modellannahme für das Objekt mächtig genug sein, die unterschiedlichen Abbildungen zu lernen. Eine einfache Gauß-Verteilung erfüllt diese Ansprüche nicht. Besser geeignet ist die Gauß'sche Mischverteilung. Setzt man die Definition der GMV in den Klassifikator 3.13 ein, ergibt sich der folgende Ausdruck:

$$r(\mathbf{X}) = \underset{k}{\text{argmax}}\Bigg\{ \max_{\vartheta_0^1}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0})\cdot \frac{1}{\sqrt{|2\pi\sigma_{1k}^2\mathbf{I}|}}\sum_{i=1}^{I_k} c_{ik}\exp\big(-\frac{1}{2\sigma_{1k}^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_{1ik}(\vartheta_1)\|^2\big)\Big\}\Bigg\} \tag{3.15}$$

Mit $\sum_{i=1}^{I_k} c_{ik} = 1$ ist die Voraussetzung für eine Wahrscheinlichkeitsdichte gegeben.

3.2.3 KD und NN-Regel für Einzelobjekterkennung mit Hintergrundmodell

Möchte man keine Annahmen über die reale Verteilung der Objekte machen, dann kann man besser auf ein nicht parametrisches Verfahren zurückgreifen. Ähnlich wie bei den reinen Einzelobjekterkennern folgen KD und NN als Spezialfall aus den GMVen.

Seien $\mathbf{x}_{1k},\ldots,\mathbf{x}_{N_k k}$, mit $x \in \mathbb{R}^D$ die Trainingsdaten der k-ten Klasse, dann ergibt sich aus (3.7):

$$r(\mathbf{X}) = \underset{k}{\text{argmax}}\Bigg\{ \max_{\vartheta_0^1}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0})\cdot \frac{1}{\sqrt{|2\pi\sigma_{1k}^2\mathbf{I}|}}\cdot\sum_{n=1}^{N_k}\frac{1}{N_k}\cdot\exp\big(-\frac{1}{2\sigma_{1k}^2}\|\mathbf{X}_{S_1}-\mathbf{x}_{nk}(\vartheta_1)\|^2\big)\Big\}\Bigg\} \tag{3.16}$$

Mit den Voraussetzungen aus Abschnitt 3.1.4 folgt die NN-Regel wieder aus den KD durch Maximum-Approximation, und man erhält folgenden Klassifikator:

$$\begin{aligned}
r(\mathbf{X}) &\overset{(3.16)}{=} \underset{k}{\text{argmax}}\Bigg\{ \max_{\vartheta_0^1}\Big\{\hat{p}(\vartheta_0^1)\cdot\frac{N_k}{N}\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0})\cdot \sum_{n=1}^{N_k}\frac{1}{N_k}\cdot\frac{1}{\sqrt{|2\pi\sigma_1^2\mathbf{I}|}}\cdot\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\mathbf{x}_{nk}(\vartheta_1)^2\|\big)\Big\}\Bigg\} \\
&\approx \underset{k}{\text{argmax}}\Bigg\{ \max_{\vartheta_0^1;n=1\ldots N_k}\Big\{\hat{p}(\vartheta_0^1)\cdot\frac{N_k}{N}\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0})\cdot \frac{1}{N_k}\frac{1}{\sqrt{|2\pi\sigma_1^2\mathbf{I}|}}\cdot\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\mathbf{x}_{nk}(\vartheta_1)\|^2\big)\Big\}\Bigg\} \\
&= \underset{k}{\text{argmin}}\Bigg\{ \min_{\vartheta_0^1;n=1\ldots N_k}\Big\{-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(\mathbf{X}_{S_0}|\mu_{k_0}) + \frac{D}{2}\log(2\pi\sigma_1^2)+\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\mathbf{x}_{nk}(\vartheta_1)\|^2\Big\}\Bigg\}
\end{aligned} \tag{3.17}$$

3.2.4 Hintergrundmodelle

Der Hintergrund ist, nach Festlegung in Abschnitt 3.2, als die Menge definiert, die durch die Positionierung der Objekte in der Bildszene übrig bleibt. Demnach ist der Hintergrund keine zusammenhängende, wohlgeordnete Menge von Pixeln, die in jeder Bildszene identisch ist, und kann somit nicht durch eine multivariate Funktion beschrieben werden. Aus diesem Grund werden für die Hintergrundmodellierung im wesentlichen 3 verschiedene, univariate Verteilungen auf Pixelebene getestet. Als Voraussetzung für alle Typen gilt die vereinfachende Annahme der statistischen Unabhängigkeit der einzelnen Hintergrundpixel.

Der erste Typ ist die univariate Gauß-Verteilung, die durch die Parameter μ_0 und σ_0^2 festgelegt ist. Für das Modell ergibt sich folgender Ausdruck:

$$\hat{p}(\mathbf{X}_{S_0}|\mu_0) \quad = \quad \prod_{x \in \mathbf{X}_{S_0}} \frac{1}{\sqrt{2\pi\sigma_0^2}} \cdot \exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right) \tag{3.18}$$

Als zweite Möglichkeit lässt sich der Hintergrund als gleichverteilt auffassen. Sei G die Anzahl der Farbwerte, die im Bild vorkommen können, dann sieht das Modell folgendermaßen aus:

$$\hat{p}(\mathbf{X}_{S_0}|\mu_0) = \hat{p}(\mathbf{X}_{S_0}) \quad = \quad \prod_{x \in \mathbf{X}_{S_0}} \frac{1}{G} \tag{3.19}$$

Als dritter Typ werden Histogramme verwendet, die über die Anzahl der Grauwerte G normiert sind. Es seien $x_1, \ldots, x_n$ mit $x_n \in \mathbb{R}$ eine Menge von Trainingsdaten und $[l, u]$ mit $l, u \in \mathbb{R}$ das Intervall, auf dem das Histogramm definiert ist. Dann lässt sich das Intervall $[l, u]$ in B Teilintervalle $\Delta_b = [l + b\frac{u-l}{B}, l + (b+1)\frac{u-l}{B})$ mit $b = 0, \ldots, B-1$ unterteilen.

Die empirische Verteilungsfunktion sieht dann wie folgt aus:

$$h^B(x) \quad := \quad \frac{|\{x_n \mid x, x_n \in \Delta_b\}|}{N} \tag{3.20}$$

Als Hintergrundmodell ergibt sich:

$$\hat{p}(\mathbf{X}_{S_0}|\mu_0) = \hat{p}(\mathbf{X}_{S_0}) \quad = \quad \prod_{x \in \mathbf{X}_{S_0}} h^B(x) \tag{3.21}$$

Zu erwähnen sei noch, dass sich die Gleichverteilung als Histogramm mit einem Intervall auffassen lässt. Das hat den Vorteil, dass für die Gleichverteilung keine neue Funktion implementiert werden muss.

3.3 Zusammenfassung

In diesem Kapitel wurden einige Verfahren der Einzelobjekterkennung auf Basis der Bayes'schen Entscheidungsregel vorgestellt. Diese reichen in der Praxis nicht aus, da Objekte meistens nicht einzeln und in segmentierter Form vorliegen. Aus diesem Grund wurde aus diesen Einzelobjekterkennern eine allgemeine Formel eines Multiobjekterkenners hergeleitet. Dieser Multiobjekterkenner stellt ein sehr komplexes Suchproblem. Um erste Experimente mit diesem Erkenner

durchführen zu können, wurde daraus zur Vereinfachung ein Einzelobjekterkenner mit Hintergrundmodell abgeleitet.

Als Spezialisierung der Objektmodelle wurden die multivariate Gauß-Verteilung und die Gauß'schen Mischverteilungen, für die Hintergrundmodelle eine univariate Gauß-Verteilung, Gleichverteilung und Histogramme vorgestellt.

Als nichtparametrischen Ansatz wurden *Kernel Densities* und die Nächste-Nachbar-Regel erklärt. Bei diesen Verfahren hat man wiederum den Nachteil, dass die Trainingsdaten vorher segmentiert sein müssen.

Kapitel 4

Automatisches Objekttraining

Wenn Objekte in einer Bildszene in unterschiedlichen Transformationen vorliegen können, kann die Bildszene nicht mehr im Ganzen als Objekt behandelt werden. Dies ist jedoch bei allen Trainingsalgorithmen der Fall, die mit vorsegmentierten Daten arbeiten. Da diese Vorsegmentierung manuell vorgenommen wird, bedeutet dies eine sehr zeit- und kostenaufwändige Vorverarbeitung. Wünschenswert wäre, wenn diese Segmentierung automatisch abliefe. Ein Trainingsalgorithmus für komplexe Szenen sollte daher in der Lage sein, eigenständig zwischen Objekt und Hintergrund zu unterscheiden, um das Objekt trainieren zu können. Wie schon in der Einleitung erwähnt, reicht es nicht aus, sich nur auf das Objekt zu beziehen. Vielmehr muss neben dem Objektmodell auch ein Hintergrundmodell miteinbezogen und trainiert werden. Daraus ergibt sich folgende Fragestellung: Wie kann ein Algorithmus formuliert werden, der in der Lage ist, nur aus der Information, dass in allen Trainingsbildern das gleiche Objekt vorhanden ist, dieses zu erkennen und zu trainieren?

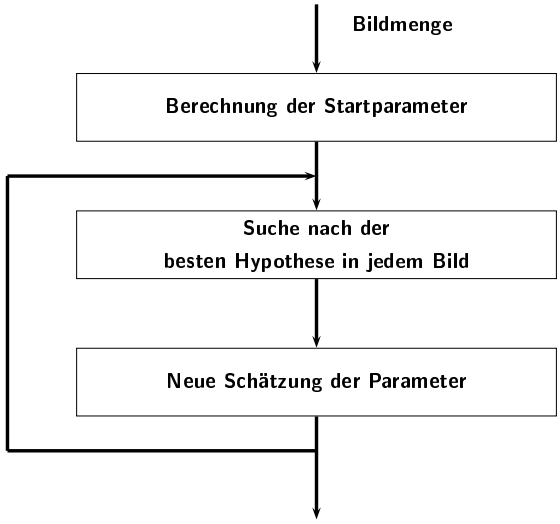


Abbildung 4.1: Schema eines iterativen Algorithmus zur automatischen Segmentierung.

Die nächsten Abschnitte beschäftigen sich damit, wie so ein Algorithmus aussehen kann. Da die Objekte in den Trainingsdaten erst lokalisiert werden müssen, lassen sich die Einzelobjekterkenner mit Hintergrundmodell in leicht abgewandelter Form dafür verwenden.

Um das Objektmodell immer besser an das reale Objekt anzunähern, wird ein iterativer Algorithmus gewählt, der mit einer ungenauen Beschreibung des Objekts anfängt und sich durch eine

iterative Verbesserung an das tatsächliche Modell anpasst.

In Abb. 4.1 ist das Prinzip der Iteration schematisch dargestellt. Als Eingabe liegt eine Menge von Bildern vor, von denen lediglich bekannt ist, dass ein bestimmtes Objekt in allen Bildern enthalten ist. Aus dieser Eingabemenge werden als erstes die Startparameter für die Iteration bestimmt. Mittels dieser Parameter wird für jedes Bild eine neue Hypothese für das Objekt aufgestellt. Aus diesen Hypothesen werden die Modellparameter neu trainiert. Mit den trainierten Parametern werden im nächsten Iterationsschritt wiederum bessere Hypothesen für das Objekt generiert usw. Ziel ist es, durch mehrere Iterationsschritte eine immer genauere Beschreibung des Objektes zu erhalten. Dieses Verfahren ähnelt dem bekannten „*Expectation-Maximization*-Algorithmus“, der z.B. bei der iterativen Schätzung von Mischverteilungen eingesetzt wird (siehe 4.2.2).

4.1 Berechnung der Startparameter

Liegen keinerlei Informationen über das zu trainierende Objekt vor, so müssen erst einmal die Startparameter des iterativen Algorithmus aus der gegebenen Bildmenge geschätzt werden. Mit diesen Schätzwerten können anschließend die Objekte in jedem Bild neu hypothetisiert werden.

In dieser Arbeit werden zwei einfache Varianten verwendet, um geeignete Startwerte der Verteilungen zu finden. Zum einen werden die Parameter auf konstante Werte gesetzt, die sich durch Berechnung des Mittelwertes und der Varianz aus allen Trainingsdaten empirisch berechnen lassen. Zusätzlich lassen sich Histogramme über den Trainingsdaten heranziehen, um die Parameter zu bestimmen. Anhand der Verteilung der Pixelwerte, die aus dem Histogramm ersichtlich ist, lassen sich *a-priori* Aussagen über das Objekt machen, die genutzt werden können, um die Startparameter einzustellen.

Zum anderen wählt man ein Bild aus den Trainingsdaten aus und segmentiert das Objekt manuell, um eine genauere Beschreibung des Objektes zu Beginn der Iteration festzulegen.

Diese Vorgehensweisen sorgen für eine passende Initialisierung der Modelle, brauchen aber deutlich weniger manuelle Vorarbeit als die Segmentierung aller Trainingsdaten.

4.2 Suche nach der besten Hypothese

Die Suche nach der besten Hypothese findet in einem komplexen Suchraum statt, da theoretisch eine hohe Anzahl von Abbildungen in Betracht gezogen werden muss. Bei der Aufstellung von Modellen hat man zum einen die Wahl, die Abbildungen direkt in die Modelle einzuarbeiten oder indirekt die Abbildungen aus den Trainingsdaten zu lernen. Werden die Abbildungen direkt in die Modelle eingearbeitet, müssen alle möglichen Abbildungen explizit hypothetisiert werden. Dies bedeutet, dass alle Abbildungen auf die Referenzen angewendet und durchgetestet werden müssen.

Bei der indirekten Methode dagegen werden die Abbildungen von den Modellen gelernt, so dass weniger Hypothesen getestet werden müssen, die Modelle aber komplizierter werden.

Beide Varianten haben ihre Vor- und Nachteile, die in Tablle 4.2 aufgelistet sind.

In dieser Arbeit werden Translationen, Skalierungen unter Einhaltung des Seitenverhältnisses und 2-dimensionale Rotationen direkt in das Modell eingearbeitet. Zusätzliche Abbildungen, die in den Trainingsdaten vorkommen, wie z.B. die 3-dimensionalen Rotationen in den `COIL-` und `COIL-I6-`Daten, werden durch die Verteilungsfunktionen gelernt.

Hinzu kommen noch folgende Rahmenbedingungen:

Tabelle 4.1: Vor- und Nachteile der (in-)direkten Modellierung

	direkte Modellierung	**indirekte Modellierung**
Verteilungsfunktionen	Beschränkung auf einfache Funktionen (z.B. Gauß-Verteilungen)	komplizierte Funktionen notwendig (z.B. Mischverteilungen)
Trainingsdaten	im Vergleich weniger notwendig, da Abbildungen nicht gelernt werden müssen	viele Trainingsdaten notwendig, um Abbildungen sinnvoll zu lernen
Aufwand	hoher Aufwand zum Testen der modellierten Abbildungen	Aufwand steckt im Training der Verteilung

- es befindet sich genau ein Objekt im Bild
- das Objekt liegt in einem Rechteck mit festem Seitenverhältnis
- als Referenzmodelle für die Objekte werden eine multivariate Gauß-Verteilung oder Gauß'sche Mischverteilungen verwendet
- als Referenzmodell für den Hintergrund werden eine univariate Gauß-Verteilung, Gleichverteilung oder Histogramm verwendet

4.2.1 Training der Referenzmodelle

Das Training der Referenzmodelle ist in dieser Arbeit nicht diskriminativ. Dies erlaubt es, das Training der einzelnen Objektklassen unabhängig voneinander durchzuführen und zu betrachten. Für das Training werden Datenpaare $(\mathbf{x}_n, k)$ mit $n = 1, \ldots, N_k$ bereit gestellt. Dabei ist $k = 1, \ldots, K$ die Menge aller Objektklassen und $\{\mathbf{x}_n\}$ eine Menge von Merkmalsvektoren. Das Tupel $(\mathbf{x}_n, k)$ bedeutet, dass der Merkmalsvektor $\mathbf{x}_n$ der Klasse k zugeordnet ist und wird im Folgenden mit $\mathbf{x}_{nk}$ bezeichnet. Da für das Training bekannt ist, zu welcher Klasse ein Merkmalsvektor gehört, wird diese Art des Trainings auch *überwachtes Lernen* genannt.

Da das Training für jede Klasse analog abläuft, wird im Folgenden auf den Klassenindex k verzichtet. Wird als Referenzmodell eine multivariate Gauß-Verteilung vorausgesetzt, so sind zwei Parameter unbekannt, der Mittelwertsvektor $\boldsymbol{\mu}$ und die Kovarianzmatrix $\boldsymbol{\Sigma}$. In Kapitel 3 wurde eine „gepoolte" Kovarianzmatrix vorausgesetzt. Die gleichen Voraussetzungen gelten auch für das Training, so dass $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ gilt. Seien $\mathbf{x}_1, \ldots, \mathbf{x}_N$ Trainingsdaten einer Objektklasse gegeben, dann erfolgt das Training nach der *Maximum-Likelihood-Methode* (ML-Methode). Diese wählt die Parameter so, dass das Modell die beste Erklärung für die Trainingsdaten liefert. Im Fall der Gauß-Verteilung erhält man als Schätzer für $\boldsymbol{\mu}$ und σ^2:

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \\ \hat{\sigma}^2 &= \frac{1}{N}\sum_{n=1}^{N}\| \mathbf{x}_n - \hat{\boldsymbol{\mu}} \|^2 \end{aligned}$$

Analog lassen sich die Schätzer für eine univariate Gauß-Verteilung berechnen, indem die Vektoren gegen skalare Variablen ausgetauscht werden.

Ähnlich einfach lassen sich Histogramme trainieren. Mit $\Delta_b = [l + b\frac{u-l}{B}, l + (b+1)\frac{u-l}{B})$ und $b = 0, \ldots, B-1$ gilt:

$$\hat{h}^B(x) = \frac{|\{x_n \in \mathbb{R} \mid x, x_n \in \Delta_b\}|}{N} \tag{4.1}$$

Aufwendiger wird das Training, wenn eine Gauß'sche Mischverteilung zu Grunde gelegt wird. Eine solche Mischverteilung ist eine Linearkombination aus einzelnen Dichten. Die Koeffizienten c_i dieser Linearkombination sind allerdings zu Beginn des Trainings nicht bekannt. Die Zuordnung der Beobachtung zu den Einzeldichten wird dabei als Zufallsvariable behandelt. Die Wahrscheinlichkeitsdichte, einen Trainingsvektor $\mathbf{x}_n$ zu beobachten, wenn die Modellparameter $(\{c_i\}, \{\boldsymbol{\mu}_i\}, \sigma^2)$ gegeben sind, lässt sich mit dem folgenden Ausdruck beschreiben:

$$p(\mathbf{x}_n|\{c_i\}, \{\boldsymbol{\mu}_i\}, \sigma^2\mathbf{I}) = \sum_{i=1}^{I} c_i \cdot \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_i, \sigma^2\mathbf{I}) \tag{4.2}$$

Gesucht sind die Modellparameter, die die Trainingsdaten am besten beschreiben. Dies führt mittels *Maximum-Likelihood*-Kriterium zu einem unhandlichen Maximierungsproblem, das es zu lösen gilt. Eine effiziente Möglichkeit, ein lokales Optimum zu finden, bietet der *Expectation-Maximization*-Algorithmus (EM-Algorithmus). Eine allgemeine Herleitung des Algorithmus kann man in [16, 96ff] nachlesen.

Der EM-Algorithmus beruht auf dem Prinzip einer „versteckten“ Zufallsvariablen. Im Fall der Gauß'schen Mischverteilung beschreibt die „versteckte“ Zufallsvariable die Zuordung des Trainingsvektors zu einer einzelnen Dichte. Diese Zuordnung ist aus den Daten nicht beobachtbar, was dieser Variablen den Namen „versteckte“ Zufallsvariable gibt.

Es gilt:

$$\begin{aligned} p(\mathbf{x}_n|\{c_i\}, \{\boldsymbol{\mu}_i\}, \sigma^2\mathbf{I}) &= \sum_{i=1}^{I} p(\mathbf{x}_n, i|\{\boldsymbol{\mu}_i\}, \sigma^2\mathbf{I}) \\ &= \sum_{i=1}^{I} \underbrace{p(i|\boldsymbol{\mu}_i, \sigma^2\mathbf{I})}_{\equiv c_i} \cdot p(\mathbf{x}_n|i, \boldsymbol{\mu}_i, \sigma^2\mathbf{I}) \end{aligned} \tag{4.3}$$

Der EM-Algorithmus basiert auf einem iterativen Ansatz, der in jedem Iterationsschritt versucht, die Parameter immer besser zu schätzen. Für die versteckten Variablen werden Erwartungswerte berechnet (*Expectation*), mit denen ein neuer *Maximum-Likelihood*-Parametersatz in Abhängigkeit von dem vorhergehenden Parametersatz (*Maximization*) berechnet wird.

Die Übertragung des EM-Algorithmus auf die Gauß'schen Mischverteilungen kann man in [7] nachlesen. Im Folgenden sind lediglich die Reestimationsformeln angegeben, deren Berechnungen im Anhang A gezeigt werden. Es sei $\bar{\lambda} \equiv (\{\bar{c}_i\}, \{\bar{\boldsymbol{\mu}}_i\}, \bar{\sigma}^2)$ der Parametersatz, der aus dem alten Parametersatz $\lambda \equiv (\{c_i\}, \{\boldsymbol{\mu}_i\}, \sigma^2)$ neu geschätzt werden soll, dann gilt:

$$
\begin{aligned}
p(i|\mathbf{x}_n, \lambda) &= \frac{c_i \cdot p(\mathbf{x}_n|i, \boldsymbol{\mu}_i, \sigma^2)}{\sum_{i'=1}^{I} c_{i'} \cdot p(\mathbf{x}_n|i', \boldsymbol{\mu}_{i'}, \sigma^2)} \\
&= \frac{c_i \cdot \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_i, \sigma^2\mathbf{I})}{\sum_{i'=1}^{I} c_{i'} \cdot \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{i'}, \sigma^2\mathbf{I})} \\
\gamma_i(n) &= \frac{p(i|\mathbf{x}_n, \lambda)}{\sum_{n'=1}^{N} p(i|\mathbf{x}_{n'}, \lambda)} \\
\bar{c}_i &= \frac{1}{N} \sum_{n=1}^{N} p(i|\mathbf{x}_n, \lambda) \\
\bar{\boldsymbol{\mu}}_i &= \sum_{n=1}^{N} \gamma_i(n) \cdot \mathbf{x}_n \\
\bar{\sigma}^2 &= \frac{1}{D} \sum_{n=1}^{N} \gamma_i(n) \|\mathbf{x}_n - \bar{\boldsymbol{\mu}}_i\|^2
\end{aligned}
$$

Für den nächsten Iterationsschritt wird $\lambda := \bar{\lambda}$ gesetzt. Im einzelnen bedeuten die Parameter:

- $\gamma_i(n)$ sind die Anteile, mit denen eine Beobachtung $\mathbf{x}_n$ in die Schätzung der $\bar{\boldsymbol{\mu}}_i$ und $\bar{\sigma}^2$ eingeht.
- $\boldsymbol{\mu}_i$ sind die empirischen Mittelwertsvektoren.
- σ^2 sind die empirischen Varianzen.
- c_i sind die gewichteten, relativen Häufigkeiten.

Mit dem EM-Algorithmus lassen sich die Parameter effizient trainieren. Jedoch wird vorausgesetzt, dass die Anzahl der Dichten fest vorgegeben ist. Zu Beginn eines Trainings ist jedoch nicht klar, wie viele Dichten verwendet werden sollen und wie die Startparameter zu wählen sind. Je mehr Dichten verwendet werden, desto vorsichtiger müssen die Startparameter gewählt werden, um ein gutes Optimum zu erreichen.

In der Diplomarbeit von M. O. Güld [12] wurde ein Verfahren vorgestellt, das die Wahl der Startparameter erleichtert und bei einer gegebenen, maximalen Anzahl von Dichten eine optimale Anzahl von Dichten findet.

In diesem Ansatz wird mit einer einzelnen Dichte gestartet. Diese wird solange aufgespalten, bis eine optimale Anzahl von Dichten erreicht ist. Dadurch dass ein Mittelwertsvektor einer Dichte entlang ihrer maximalen Varianz leicht „gestört“ wird, wird ein Aufspalten der Dichte erreicht. Nach dem Teilen einer Dichte werden die Parameter der neuen Mischverteilung mittels EM-Algorithmus trainiert. Die Dichten der neu geschätzten Parameter werden wiederum aufgespalten und trainiert. Dieser iterative Vorgang wiederholt sich solange, bis entweder die maximale Anzahl von Dichten erreicht ist oder das Aufspalten der Dichten beendet wird, weil nicht genügend Trainingsdaten den einzelnen Dichten zugeordnet werden können.

4.2.2 Automatisches Training der Objekte

Im vorherigen Abschnitt wurde beschrieben, wie sich die Referenzmodelle trainieren lassen. Auf welchen Trainingsdaten diese Referenzmodelle jedoch trainiert werden, wurde noch nicht

erwähnt. Die gegebenen Bildszenen können nicht verwendet werden, da nicht bekannt ist, wo sich die Objekte befinden. Aus diesem Grund muss vorher eine Lokalisation der Objekte stattfinden. Dazu lassen sich die Einzelobjektklassifikatoren mit Hintergrundmodell aus Abschnitt 3.2 in leicht abgewandelter Form verwenden. Diese suchen in der Bildszene nach der Hypothese, die am besten zur aktuellen Objektbeschreibung passt. In Abbildung 4.2 ist diese Suche schema-

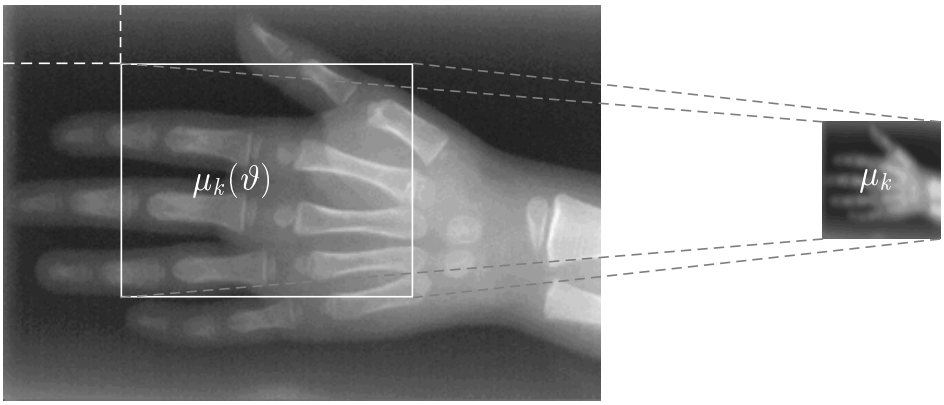


Abbildung 4.2: Suche nach der wahrscheinlichsten Abbildung

tisch dargestellt. Sie findet in jedem Iterationsschritt und separat für alle Trainings-Bildszenen statt, so dass im Folgenden auf den Bildindex n und Klassenindex k verzichtet wird. Auf der rechten Seite der Abbildung ist das Referenzmodell abgebildet. Dieses wird mit einer Abbildung ϑ (in der Abbildung 4.2 sind das Skalierung und Translation) auf die Beobachtung projiziert. Die projizierte Position ist eine mögliche Objektposition, d.h. eine Hypothese. Diese wird in Abhängigkeit des Referenzmodelles bewertet. Der Ausschnitt mit der besten Bewertung bildet die neue Objekthypothese für die gegebene Bildszene.

Im Folgenden ist beschrieben, wie sich diese Suche aus dem Einzelobjektklassifikator mit Hintergrundmodell herleiten lässt.

In der Klassifikation wird die Klasse gesucht, die am besten durch das Modell erklärt wird. Im Training kennen wir die Klasse und sind an der besten Projektion interessiert, so dass sich die Lokalisation aus der Formel 3.13 wie folgt verwenden lässt:

$$\underset{\vartheta_0^1}{\text{argmax}}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\hat{p}(\mathbf{X}_{S_0}|\mu_0)\cdot\hat{p}(\mathbf{X}_{S_1}|\boldsymbol{\mu}_1(\vartheta_1))\Big\} \tag{4.4}$$

Sei als Voraussetzung für das Referenzmodell eines Objektes die multivariate Gauß-Verteilung und für das Referenzmodell des Hintergrundes eine univariate Gauß-Verteilung gegeben, dann lässt sich die Suche nach der besten Hypothese in der Bildszene wie folgt herleiten. Mit Formel 4.4

erhält man die beste Hypothese $\hat{\mathbf{X}}_{S_1}$ durch Verwendung der optimalen Parameter $\hat{\vartheta}_1$.

$$
\begin{aligned}
\hat{\vartheta}_1 &\overset{4.4}{=} \underset{\vartheta_1}{\mathsf{argmax}}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\mathcal{N}(\mathbf{X}_{S_0}|\mu_0,\sigma_0^2)\cdot\mathcal{N}\big(\mathbf{X}_{S_1}|\boldsymbol{\mu}_1(\vartheta_1),\sigma_1^2\mathbf{I}\big)\Big\}\\
&= \underset{\vartheta_1}{\mathsf{argmax}}\Big\{\hat{p}(\vartheta_0^1)\cdot\hat{p}(k)\cdot\prod_{x\in\mathbf{X}_{S_0}}\Big(\frac{1}{\sqrt{2\pi\sigma_0^2}}\cdot\exp\big(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\big)\Big)\cdot\\
&\qquad\qquad\frac{1}{\sqrt{|2\pi\sigma_1^2\mathbf{I}|}}\cdot\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_1(\vartheta_1)\|^2\big)\Big\}\\
&= \underset{\vartheta_1}{\mathsf{argmax}}\Big\{\log\hat{p}(\vartheta_0^1)+\log\hat{p}(k)-\frac{1}{2}|S_0|\log(2\pi\sigma_0^2)-\frac{(x-\mu_0)^2}{2\sigma_0^2}\\
&\qquad\qquad-\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_1(\vartheta_1)\|^2\Big\}\\
&= \underset{\vartheta_1}{\mathsf{argmin}}\Big\{-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)+\frac{1}{2}|S_0|\log(2\pi\sigma_0^2)+\frac{(x-\mu_0)^2}{2\sigma_0^2}\\
&\qquad\qquad+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)+\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_1(\vartheta_1)\|^2\Big\}
\end{aligned}
\tag{4.5}
$$

Analog zur Formel 4.5 kann die Herleitung auf die Gleichverteilung (siehe Formel 4.6) und auf die Histogramme (siehe 4.7) angewandt werden.

$$
\hat{\vartheta}_1 = \underset{\vartheta_1}{\mathsf{argmin}}\Big\{-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)+|S_0|\log(G)+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)+\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_1(\vartheta_1)\|^2\Big\}
\tag{4.6}
$$

$$
\begin{aligned}
\hat{\vartheta}_1 = \underset{\vartheta_1}{\mathsf{argmin}}\Big\{&-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)-\sum_{x\in\mathbf{X}_{S_0}}\log h^B(x)\\
&+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)+\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_1(\vartheta_1)\|^2\Big\}
\end{aligned}
\tag{4.7}
$$

Ebenso ergeben sich die Formeln für die Gauß'sche Mischverteilung mit den Hintergrundmodellen: Gauß-Verteilung (4.8), Gleichverteilung (4.9) und Histogramm (4.10).

$$
\begin{aligned}
\hat{\vartheta}_1 = \underset{\vartheta_1}{\mathsf{argmin}}\Big\{&-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)+\frac{1}{2}|S_0|\log(2\pi\sigma_0^2)+\frac{(x-\mu_0)^2}{2\sigma_0^2}\\
&+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)-\log\sum_{i=1}^{I}c_i\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_i(\vartheta_1)\|^2\big)\Big\}
\end{aligned}
\tag{4.8}
$$

$$
\begin{aligned}
\hat{\vartheta}_1 = \underset{\vartheta_1}{\mathsf{argmin}}\Big\{&-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)+|S_0|\log(G)\\
&+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)-\log\sum_{i=1}^{I}c_i\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_i(\vartheta_1)\|^2\big)\Big\}
\end{aligned}
\tag{4.9}
$$

$$
\begin{aligned}
\hat{\vartheta}_1 = \underset{\vartheta_1}{\mathsf{argmin}}\Big\{&-\log\hat{p}(\vartheta_0^1)-\log\hat{p}(k)-\sum_{x\in\mathbf{X}_{S_0}}\log h^B(x)\\
&+\frac{1}{2}|S_1|\log(2\pi\sigma_1^2)-\log\sum_{i=1}^{I}c_i\exp\big(-\frac{1}{2\sigma_1^2}\|\mathbf{X}_{S_1}-\boldsymbol{\mu}_i(\vartheta_1)\|^2\big)\Big\}
\end{aligned}
\tag{4.10}
$$

4.3 Zusammenfassung

In diesem Kapitel wurde ein iterativer Algorithmus vorgestellt, mit dem Objekte automatisch trainiert werden sollen. Dies erspart eine manuelle Vorverarbeitung aller Trainingsdaten, was eine mühevolle Arbeit ist, wenn zum Beispiel alle Objekte einzeln segmentiert werden müssen. Lediglich für die Wahl der Startparameter muss eine wenig aufwändige, manuelle Vorverarbeitung der Daten durchgeführt werden.

Weiterhin wurde beschrieben, wie sich die verschiedenen äußeren Erscheinungen der Objekte in das Training integrieren lassen. Dabei gibt es die Möglichkeiten der direkten und indirekten Modellierung, die in Tabelle 4.2 gegenübergestellt werden. In dieser Arbeit werden 2-dimensionale Rotationen, Skalierungen unter Einhaltung des Seitenverhältnisses und Translationen direkt in die Modelle integriert, so dass für diese Abbildungen die Hypothesen explizit generiert werden müssen. Andere Abbildungen wie 3-dimensionale Rotationen oder Beleuchtungsverhältnisse werden durch eine Gauß'sche Mischverteilung trainiert. Diese Abbildungen müssen demnach aus den Trainingsdaten hervorgehen.

Es wurde beschrieben, wie die verwendeten Modelle im einzelnen mittels *Maximum-Likelihood*-Kriterium und EM-Algorithmus trainiert werden.

Kapitel 5

Stand der Technik

Statistische Klassifikation von Objekten unter Berücksichtigung eines Hintergrundmodells ist ein recht junges Forschungsgebiet. Dies wurde zum einen deutlich bei der Suche nach einer standardisierten Bilddatenbank für diese Problematik, zum anderen aber auch in der Anzahl der Veröffentlichungen zu diesem Thema. Es gibt bisher nur wenige Gruppen, die an dieser Thematik arbeiten.

In den folgenden Abschnitten werden zwei Ansätze zu diesem Thema kurz vorgestellt. Der erste Ansatz klassifiziert Objekte unter Berücksichtigung eines Hintergrundmodells, wobei Hintergrundmodell und Objektmodell getrennt voneinander trainiert werden. Das Training der Objekte findet auf Daten statt, die realen Hintergrund beinhalten. Der Hintergrund wird als Gleichverteilung modelliert. In dem zweiten vorgestellten Ansatz wird zuerst versucht, die Objekte automatisch zu trainieren. Als Voraussetzung hat man, wie in Abb. 5.1, eine Menge von Bildern, die das Objekt enthalten, und eine Menge von Bildern, die das Objekt nicht enthalten. Das Verfahren soll automatisch erkennen, welche Objekte in der einen Menge auftreten und in der anderen Menge nicht. Diese gefundenen Objekte werden schließlich trainiert und für die Klassifikation verwendet.

In [19] wird ein Verfahren beschrieben zur Lokalisation und Klassifikation von 2-dimensionalen Objekten. Als zusätzliche Schwierigkeit wurden die Objekte entweder auf heterogenem, realem Hintergrund positioniert oder von anderen Objekten überdeckt.

Es wird ein Multi-Skalen-Ansatz verwendet, auf den im Folgenden nicht näher eingegangen wird, da dieser lediglich zur Beschleunigung der Algorithmen dient und keinen Einfluss auf die Modellierung hat.

Sei $\mathbf{X} := \{x_m \in \mathbb{R} \mid m = 0, \ldots, I \cdot J - 1\}$ eine Bildszene der Dimension $I \times J$. Ein Objekt ist eine geschlossene Kontur $S_1 \subset \{0, \ldots, I \cdot J - 1\}$ und $\mathbf{X}_{S_1} := \{x_m \in \mathbb{R} \mid m \in S_1\}$ die Menge der Pixel, die in dieser Kontur liegen und als Objekt interpretiert werden.

Der hier vorgestellte Ansatz arbeitet nicht direkt auf den Pixeldaten, sondern es werden Merkmalsanalysen durchgeführt, so dass theoretisch pro Pixel mehrere Merkmale bei der Modellierung einbezogen werden können. Der Merkmalsvektor an der Position m wird mit $\mathbf{c}_m$ bezeichnet. Dann ist $\mathbf{c}_{S_1} := \{\mathbf{c}_m \in \mathbb{R}^D \mid m \in S_1\}$ die Menge der Merkmalsvektoren, die zum Objekt gehören.

In der Arbeit [19] werden die Merkmale mittels *diskreten Johnston Wavelets* berechnet. Als Voraussetzung gilt, dass die Merkmalsvektoren innerhalb von S_1 statistisch unabhängig von den Merkmalsvektoren außerhalb von S_1 sind, und dass die Merkmalsvektoren selber auch statistisch unabhängig voneinander sind.

Abbildung 5.1: Welche Objekte erscheinen im linken Bild, jedoch nicht auf der rechten Seite? Ist eine Maschine im Stande, die entscheidenden Charakteristiken der zwei Objektklassen (Auto und Gesicht) zu lernen, ohne dass noch zusätzliche Information gegeben wird? (aus [23])

Die Modellannahme kann dann durch eine Dichtefunktion $p(\mathbf{c}_{S_1} \mid \boldsymbol{\mu}(\Phi, t))$ beschrieben werden, und es gilt:

$$p(\mathbf{c}_{S_1} \mid \boldsymbol{\mu}(\Phi, t)) \quad = \quad \prod_{m \in S_1} p(\mathbf{c}_m \mid \boldsymbol{\mu}(\Phi, t)) \tag{5.1}$$

Die Variable Φ beschreibt die Rotationen und t die Translationen, die berücksichtigt werden. Als Klassifikator ergibt sich:

$$r(S) \quad = \quad \operatorname*{argmax}_k \left\{ \max_{\Phi, t} \prod_{m \in S_1} p(\mathbf{c}_m \mid \boldsymbol{\mu}_k(\Phi, t)) \right\} \tag{5.2}$$

Der Klassifikator (5.2) ist ein Einzelobjektklassifikator mit automatischer Lokalisation der Objekte. Um den Hintergrund zu modellieren, wird eine Zuweisungsfunktion $\zeta \in \{0, 1\}^{IJ}$ definiert, die einen Merkmalsvektor $\mathbf{c}_m$ dem Objekt ($\zeta_m = 1$) oder dem Hintergrund ($\zeta_m = 0$) zuordnet. Diese Zuweisungsfunktion wird als *versteckte Variable* dem Modell hinzugefügt, dann gilt:

$$p(\mathbf{c}_{S_1} \mid \boldsymbol{\mu}(\Phi, t)) \quad = \quad \sum_{\zeta} p(\mathbf{c}_{S_1}, \zeta \mid \boldsymbol{\mu}(\Phi, t)) \tag{5.3}$$

Aus (5.1) und (5.3) folgt schließlich:

$$\begin{aligned} p(\mathbf{c}_{S_1} \mid \boldsymbol{\mu}(\Phi,t)) &= \sum_{\zeta} \prod_{m \in S_1} p(\mathbf{c}_m, \zeta \mid \boldsymbol{\mu}(\Phi,t)) \\ &= \prod_{m \in S_1} \sum_{\zeta} p(\mathbf{c}_m, \zeta \mid \boldsymbol{\mu}(\Phi,t)) \\ &= \prod_{m \in S_1} \sum_{\zeta} p(\zeta_m) p(\mathbf{c}_m \mid \zeta, \boldsymbol{\mu}(\Phi,t)) \end{aligned} \tag{5.4}$$

Damit ist noch nicht entschieden, ob ein Merkmalsvektor dem Objekt oder Hintergrund zugeordnet wird. Diese Entscheidung wird durch folgende Maximierung getroffen:

$$\hat{\zeta}_m = \underset{\zeta_m}{\operatorname{argmax}} \left\{ p(\mathbf{c}_m \mid \zeta_m = 1, \boldsymbol{\mu}_1(\Phi,t)), p(\mathbf{c}_m \mid \zeta_m = 0, \boldsymbol{\mu}_0) \right\} \tag{5.5}$$

Diese Wahrscheinlichkeitsdichte $p(\mathbf{c}_m \mid \zeta_m = 0, \boldsymbol{\mu}_0)$ bewertet einen Merkmalsvektor, der zum Hintergrund gehört. Da für den Hintergrund eine Gleichverteilung angenommen wurde, ist die Bewertung gleich einem konstanten Wert, so dass $p(\mathbf{c}_m \mid \zeta_m = 0, \boldsymbol{\mu}_0) \equiv const$ gilt. Die Entscheidungsfunktion lässt sich demnach wie folgt ausdrücken:

$$\hat{\zeta}_m = \underset{\zeta_m}{\operatorname{argmax}} \left\{ p(\mathbf{c}_m \mid \zeta_m = 1, \boldsymbol{\mu}_1(\Phi,t)), const \right\} \tag{5.6}$$

Dies bedeutet, dass ein Merkmalsvektor dem Objekt zugeordnet wird, wenn die Wahrscheinlichkeitsdichte des Objektes $p(\mathbf{c}_m \mid \zeta_m = 1, \boldsymbol{\mu}_1(\Phi,t))$ einen Wert liefert, der größer als eine Konstante ist. Diese Entscheidung kommt demnach einem *tresholding* gleich.

Liegt der Merkmalsvektor $\mathbf{c}_m$ innerhalb der Objektgrenzen und ist $\zeta_m = 1$, dann wird $p(\zeta_m) := 1$ gesetzt, ansonsten gilt $p(\zeta_m) := 0$.

Sei nun S_1' die Indexmenge der Merkmalsvektoren, die durch die Zuweisungsfunktion ζ dem Objekt zugeordnet werden. Aufgrund von Überlagerung und Hintergrundinformation besteht die Möglichkeit, dass $|S_1| \neq |S_1'|$. Dies bedeutet, dass die Objekte in der Beobachtung in unterschiedlichen Größen erkannt und lokalisiert werden. Damit die verschieden großen Objekte vergleichbar sind, wird eine Normalisierung des Modells vorgenommen und man erhält den Klassifikator:

$$r(S) = \underset{k}{\operatorname{argmax}} \left\{ \max_{\Phi,t} \sqrt[|S_1'|]{p(\mathbf{c}_{S_1} \mid \boldsymbol{\mu}_k(\Phi,t))} \right\} \tag{5.7}$$

Die Datenbank, auf der dieser Klassifikator getestet wurde, wird genauer im Abschnitt 2.2.1 vorgestellt. Betrachtet man die Testdaten, dann fällt auf, dass das Hintergrundbild eine geringere Auflösung hat als die Objekte in den Bildern. Durch die Verwendung von *Wavelets* als Merkmale ist dies ein leichter Vorteil, denn sowohl bei homogenem Hintergrund als auch bei heterogenem Hintergrund treten keine hohen Frequenzen auf, so dass Objekt und Hintergrund anhand der hohen Frequenzen unterscheidbar sind. Die Ergebnisse sind in Abschnitt 7.3.4 aufgelistet.

In den Veröffentlichungen [23, 24, 22] wird ein Verfahren vorgestellt, automatisch Klassenmodelle für Objekte aus beliebigen, unverarbeiteten und unsegmentierten Bildszenen zu lernen. Die gelernten Modelle werden dann zur Klassifikation verwendet. Ziel ist es, Objekte, die zu einer Klasse gehören, in einer Bildszene zu erkennen (s. Abb. 5.1). Eine Objektklasse ist als Menge von Objekten definiert, die visuell ähnlich sind und die gleichen räumlichen Anordnungen besitzen.

Zum Beispiel setzt sich die Objektklasse „Gesicht“ aus den wohl angeordneten Objekten Augen, Nase und Mund zusammen. In der Praxis werden die Objekte einer Objektklasse automatisch gefunden und lassen sich in der Regel nicht so leicht interpretieren.

Um diese Aufgabenstellung zu lösen, treten 3 Probleme auf:

1. Segmentierung und Registrierung der Trainingsbilder: Welche Objekte müssen erkannt werden und wo befinden sie sich in der Bildszene?
2. Auswahl von signifikanten Objektteilen: Welche Objektteile sind charakteristisch für ein Objekt und welche tragen dazu bei, das Objekt von anderen zu unterscheiden?
3. Schätzung der Modellparameter: Mit welcher Parameterwahl werden die Objekte am besten beschrieben?

Eine Objektklasse wird als eine Komposition von *Parts* und *Shapes* definiert. *Parts* sind Bildausschnitte, die von einem Erkenner erkannt und charakterisiert werden. *Shapes* beschreiben die Lage der *Parts* untereinander. Die *Parts* und *Shapes* werden als gemeinsame Wahrscheinlichkeitsdichte modelliert.

Mittels eines *Parts*-Detektors werden *Parts* in der Bildszene gefunden. Dieser sucht nach interessanten Punkten in der Bildszene wie Kanten, Ecken, Texturen usw., die er in einer vorher definierten Größe ausschneidet. Die gefundenen *Parts* lassen sich entweder dem Objekt/Vordergrund oder dem Hintergrund zuordnen. Ein Objekt ist demnach eine Menge von *Parts* mit deren Positionen in der Bildszene und deren Typ. Es wird angenommen, dass es T verschiedene Typen von *Parts* gibt.

Die Positionen aller *Parts*, die aus einem Bild extrahiert werden, sind in einer „Matrix“ beschrieben (dies ist keine Matrix im üblichen Sinn, da die Zeilen unterschiedlich viele Elemente enthalten können):

$$\mathbf{X}^o = \begin{pmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1N_1} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \cdots & \mathbf{x}_{2N_2} \\ \vdots & \vdots & & \vdots \\ \mathbf{x}_{T1} & \mathbf{x}_{T} & \cdots & \mathbf{x}_{TN_T} \end{pmatrix} \quad \text{mit } \mathbf{x}_{tj} \in \mathbb{N}^2 \tag{5.8}$$

In der Bezeichnung $\mathbf{X}^o$ bedeutet das Superskript “o”, dass die Werte $\mathbf{x}_{tj}$ im Bild auch beobachtbar (*observable*) sind. Sollten Werte nicht direkt beobachtbar sein, dann wird dies durch das Superskript “m” beschrieben.

Die Abbildung, ob ein *Part* dem Vordergrund oder dem Hintergrund zugeordnet ist, wird in einem Vektor $\mathbf{h}$ festgehalten, in dem jeder Eintrag $h_t = j$ mit $j > 0$ bedeutet, dass $\mathbf{x}_{tj}$ zum Vordergrund gehört. Wird ein *Part* eines Objektes in der Bildszene nicht gesehen, weil er z.B. verdeckt wird, dann ist der entsprechende Eintrag in $\mathbf{h}$ gleich Null.

Da bei einer beliebigen Bildszene nicht bekannt ist, welche *Parts* zum Objekt gehören, ist $\mathbf{h}$ direkt nicht beobachtbar. Alle nicht gefundenen *Parts* eines Objektes werden in einem Vektor $\mathbf{x}^m$ festgehalten. Die Dimension von $\mathbf{x}^m$ variiert von Bildszene zu Bildszene.

Diese Vektoren lassen sich in einer gemeinsamen Wahrscheinlichkeitsdichte $p(\mathbf{X}^o, \mathbf{x}^m, \mathbf{h})$ ausdrücken. Dabei ist zu beachten, dass sowohl die Einträge von $\mathbf{x}^m$ und $\mathbf{h}$ als auch ihre Dimensionen Zufallsvariablen sind.

Um die Wahrscheinlichkeitsdichten besser beschreiben zu können, werden zusätzlich Hilfsvektoren definiert, die sich direkt aus dem Vektor $\mathbf{h}$ ergeben. Der binäre Vektor $\mathbf{b}$ speichert die

Information, welche *Parts* gefunden wurden, d.h. $h_t > 0 \;\Rightarrow\; b_t = 1$ und $h_t = 0 \;\Rightarrow\; b_t = 0$. Der Vektor $\mathbf{n}$ enthält die Komponenten n_t, die die Hintergrundkandidaten der t-ten Zeile von $\mathbf{X}^o$ zählen. Es gilt:

$$\begin{aligned} p(\mathbf{X}^0, \mathbf{x}^m, \mathbf{h}) &= p(\mathbf{X}^0, \mathbf{x}^m, \mathbf{h}, \mathbf{n}, \mathbf{b}) \\ &= p(\mathbf{X}^0, \mathbf{x}^m \mid \mathbf{h}, \mathbf{n}) p(\mathbf{h} \mid \mathbf{n}, \mathbf{b}) p(\mathbf{n}) p(\mathbf{b}) \end{aligned} \tag{5.9}$$

Die Anzahl der *Parts*, die dem Hintergrund zugeordnet werden, lässt sich als Poisson-Verteilung modellieren. Dann ist

$$p(\mathbf{n}) = p((n_1, \ldots n_T)^t) \;=\; \prod_{t=1}^{T} \frac{\lambda_t^{n_t}}{n_t!} \mathbf{e}^{-\lambda_t} \tag{5.10}$$

die Wahrscheinlichkeit, mit der die $\mathbf{n}$ *Parts* dem Hintergrund zugeordnet werden, wobei λ_t die mittlere Anzahl von Hintergrunderkennungen der *Parts* vom Typ t sind. Die Wahrscheinlichkeitsdichte $p(\mathbf{b})$ wird explizit als Tabelle mit 2^T Einträgen modelliert.

Sei $\mathcal{H}(\mathbf{b}, \mathbf{n})$ die Menge aller Hypothesen $\mathbf{h}$, die mit $\mathbf{b}$ und $\mathbf{n}$ konsistent sind, und N_t die gesamte Anzahl von *Parts* eines Typs, die in der Bildszene gefunden wurden. Dann gilt mit der Annahme, dass alle mit $\mathbf{b}$ und $\mathbf{n}$ in Einklang stehenden Hypothesen gleichverteilt sind:

$$p(\mathbf{h} \mid \mathbf{n}, \mathbf{b}) \;=\; \begin{cases} \frac{1}{|\mathcal{H}(\mathbf{b},\mathbf{n})|} & : \; \mathbf{h} \in \mathcal{H}(\mathbf{b}, \mathbf{n}) \quad \text{und} \quad |\mathcal{H}(\mathbf{b}, \mathbf{n})| = \prod_{t=1}^{T} N_t^{b_t} \\ 0 & : \; \text{sonst} \end{cases} \tag{5.11}$$

Sei $\mathbf{x}_{fg}$ der Vektor, der Koordinaten aller *Parts* enthält, die zum Vordergrund gehören, und $\mathbf{x}_{bg}$ die Koordinaten aller *Parts*, die zum Hintergrund gehören. Dann gilt unter der Voraussetzung, dass die gefundenen Vordergrund-*Parts* unabhängig von den Hintergrund-*Parts* sind:

$$p(\mathbf{X}^0, \mathbf{x}^m \mid \mathbf{h}, \mathbf{n}) \;=\; p(\mathbf{x}_{fg}) p(\mathbf{x}_{bg}) \tag{5.12}$$

Die Dichte $p(\mathbf{x}_{fg})$ wird als multivariate Gauß-Verteilung mit den Parametern $\boldsymbol{\mu}$ und $\boldsymbol{\Sigma}$ modelliert, der Hintergrund wird als Gleichverteilung angenommen.

$$\begin{aligned} p(\mathbf{x}_{fg}) &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ p(\mathbf{x}_{bg}) &= \prod_{t=1}^{T} \frac{1}{(I \cdot J)^{n_t}} \qquad I, J \text{ sind die Dimensionen der Bildszene} \end{aligned} \tag{5.13}$$

Um zu entscheiden, ob ein Objekt vorliegt, wird ein 2-Klassenproblem betrachtet. Sei k_0 die Klasse, die die Entscheidung "das Objekt ist präsent" enthält. Dann ist k_1 die Entscheidung, dass das Objekt nicht gefunden wurde. Aus der Bayes'schen Entscheidungsregel folgt:

$$\frac{p(k_1|\mathbf{X}^o)}{p(k_0|\mathbf{X}^o)} \;\approx\; \frac{\sum_{\mathbf{h}} p(\mathbf{X}^0, \mathbf{h}|k_1)}{p(\mathbf{X}^0, \mathbf{h}_0|k_0)} \tag{5.14}$$

Die Nullhypothese $\mathbf{h}_0$ besagt, dass alle *Parts* zum Hintergrund gehören.

Beim Training der Modelle muss die Parametermenge $\theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, p(\mathbf{b}), \boldsymbol{\lambda}\}$ geschätzt werden. Eine effiziente Methode bietet der EM-Algorithmus, der in [24] ausführlich beschrieben ist.

Experimente wurden auf einer eigenen Datenbank durchgeführt, die von der Gruppe selber generiert wurde. Sie ist nicht erhältlich, so dass keine vergleichenden Ergebnisse produziert werden können.

Kapitel 6

Optimierungsstrategien

In Abschnitt 3.2 wurde bereits erwähnt, dass die Multiobjektklassifikation ein komplexes Suchproblem darstellt, selbst in dem Fall, dass nur ein Objekt mit Hintergrundmodell betrachtet wird. Durch die direkte Modellierung der Translationen, Rotationen und Skalierungen in dem in dieser Arbeit verwendeten Ansatz liegt ein breites Spektrum der zu hypothetisierenden Abbildungen vor. Um aus dieser großen Menge an Hypothesen die beste zu finden, gibt es zwei Ansatzmöglichkeiten, damit die Suche in akzeptabler Zeit ein Ergebnis liefert.

Die eine Möglichkeit ist, die Suche algorithmisch zu beschleunigen, so dass Berechnungen schneller durchgeführt werden können. Die andere Möglichkeit liegt in der Einschränkung des Suchraumes, so dass nicht mehr alle Hypothesen in Betracht gezogen werden und die Suche schneller ein Ergebnis findet. Hierbei ist jedoch je nach Ansatz nicht immer gewährleistet, dass das globale Optimum gefunden wird.

Eine wesentliche Beschleunigung des Trainings bzw. der Klassifikation wird durch die schnelle Fourier-Transformation (*fast fourier transform* ,FFT) erzielt. Durch die geschickte Anwendung des Faltungstheorems ist es möglich, die euklidische Distanz effizienter zu berechnen [3].

Weiterhin treten bei der Berechnung von Distanzen Summen und Quadratsummen auf, die in dem auftretenden Kontext ebenfalls effizient berechnet werden können, ohne jedesmal explizit diese Summen auszurechnen.

Am Ende dieses Kapitels wird noch vorgestellt, wie die Suche mittels „Pruning“ (d.h. mittels Begrenzung des Suchraumes) noch schneller durchzuführen ist.

6.1 Euklidische Distanz und FFT

Ein Ziel der Maximierung aus den Abschnitten 3.2 und 4.2.2 ist die Suche nach der besten Projektion der Referenz auf die Bildszene. Für die Objektmodellierung wird hierfür eine multivariate Gauß-Verteilung oder eine Gauß'sche Mischverteilung verwendet. Aus den Formeln der Abschnitte geht hervor, dass in den Modellen euklidische Distanzen berechnet werde müssen. Für alle Bildausschnitte $S_1 \subset S$ müssen demnach die quadratischen euklidischen Distanzen $\|\mathbf{X}_{S_1} - \boldsymbol{\mu}_1(\vartheta_1)\|^2$ ausgewertet werden.

Es sei $\mathbf{X} \in \mathbb{R}^{I \times J}$ eine Bildszene mit den Indexvariablen $i = 0 \ldots I-1$ und $j = 0 \ldots J-1$. Ferner sei $\mathbf{T} := \boldsymbol{\mu}_1(\vartheta_1) \in \mathbb{R}^{M \times N} \subset \mathbb{R}^{I \times J}$ ein Template mit den Indizes $m = 0 \ldots M-1$ und $n = 0 \ldots N-1$. Außerhalb der Definitionsbereiche der Bildszene $\mathbf{X}$ und des Templates $\mathbf{T}$ werden die Werte als 0 definiert.

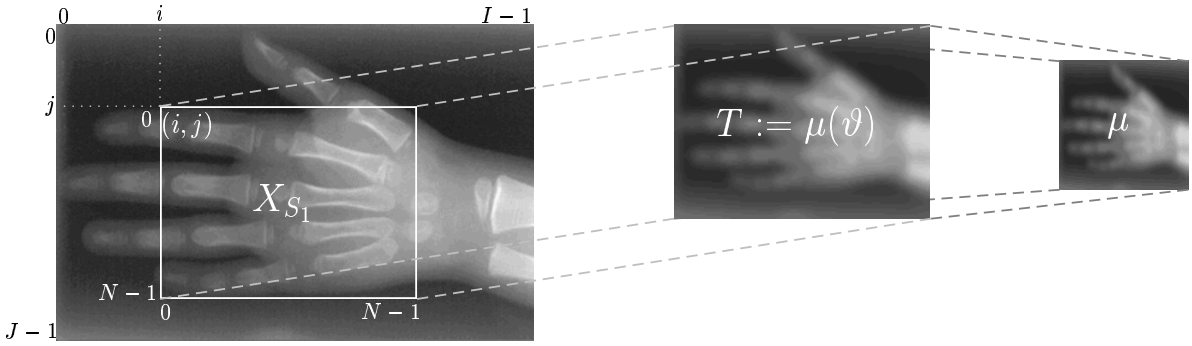


Abbildung 6.1: Suche nach der besten Projektion, die die Referenz auf die Beobachtung abbildet.

Für die Suche ergibt sich dann die in Abbildung 6.1 dargestellte Situation. Ganz rechts ist die aktuelle Referenz bzw. der aktuelle Prototyp des zu trainierenden Objektes abgebildet. Auf diesen Prototyp werden alle zu modellierenden Abbildungen angewandt. In Abbildung 6.1 sind dies Translation und Skalierung. Der größte Teil der Rechenzeit wird für die Berechnung der euklidischen Distanz zwischen dem Template $\mathbf{T}$ und der Projektionsfläche des Templates auf der Beobachtung $\mathbf{X}$ benötigt.

Betrachtet man eine feste Skalierungsstufe und beliebige Translationen, dann muss an jeder Position (i, j) in der Beobachtung die euklidische Distanz $D(i, j)$ berechnet werden und es gilt:

$$\begin{aligned} D(i,j) &= \sum_{n=j}^{N+j} \sum_{m=i}^{M+i} \big(\mathbf{X}[m,n] - \mathbf{T}[m-i, n-j]\big)^2 \\ &= \underbrace{\sum_{n=j}^{N+j} \sum_{m=i}^{M+i} \mathbf{X}^2[m,n]}_{\in\mathcal{O}(MN)} + \underbrace{\sum_{n=j}^{N+j} \sum_{m=i}^{M+i} \mathbf{T}^2[m-i,n-j]}_{\in\mathcal{O}(MN)} - \underbrace{2\sum_{n=j}^{N+j} \sum_{m=i}^{M+i} \mathbf{X}[m,n]\mathbf{T}[m-i,n-j]}_{\in\mathcal{O}(NM)} \end{aligned} \tag{6.1}$$

Um die beste Translation zu finden, muss über alle Distanzen $D(i, j)$ mit $I = 0, \ldots, I - M$ und $j = 0, \ldots J - N$ minimiert werden. Dies bedeutet, dass die Distanzen an jeder Position (i, j) berechnet werden müssen. Die im ersten Term der Formel 6.1 berechnete Quadratsumme kann effizient mit dem im Abschnitt 6.2 vorgestellten Verfahren berechnet werden und bedeutet einen einmaligen Rechenaufwand von $\mathcal{O}(IJ)$. Die Quadratsumme des zweiten Terms ist unabhängig von der Position in der Beobachtung, so dass diese nur einmal vorweg mit einem Rechenaufwand von $\mathcal{O}(MN)$ berechnet werden muss. Der letzte Term muss für jede Position neu berechnet werden. Dies ist ein Rechenaufwand von $\mathcal{O}((I - M)(J - N) \cdot MN)$. Insgesamt haben die Berechnungen der euklidischen Distanzen mit diesem Verfahren eine Komplexität von $\mathcal{O}\big((IJ)^2\big)$.

Eine effizientere Berechnung der Distanzen wird ermöglicht, durch einen Umweg über den Fourier-Raum. Durch diesen Umweg und die schnelle Fourier-Transformation ist die Berechnung ab einer bestimmten Bildgröße zu beschleunigen.

Betrachtet man den letzten Term der Formel 6.1, dann fällt auf, dass dieser sich als Kreuzkor-

relation auffassen und auf eine endliche Faltung zurückführen lässt (vgl. [17, Seite 158]).

$$
\begin{aligned}
R_{\mathbf{XT}}(i,j) &= \sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{X}[m,n]\mathbf{T}[m-i,n-j] \\
&= \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} \mathbf{X}[m,n]\mathbf{T}[m-i,n-j] \\
&= \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} \mathbf{X}[m,n]\tilde{\mathbf{T}}[i-m,j-n] \quad \text{und} \quad \tilde{\mathbf{T}}[i,j] := \mathbf{T}[-i,-j]
\end{aligned}
$$

Das Eingangssignal $\mathbf{X}$ und der Faltungskern $\mathbf{T}$ sind nur über einen endlichen Bereich definiert. Damit das Ausgangssignal vollständig rekonstruiert wird, muss die Faltung über den Indexbereich $i = 0 \ldots M + I - 2$ und $j = 0 \ldots N + J - 2$ berechnet werden (vgl. [17, Seite 103ff]). Es ergibt sich:

$$
\begin{aligned}
R_{\mathbf{XT}}(i,j) &= \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} \mathbf{X}[m,n]\tilde{\mathbf{T}}[i-m,j-n] \\
&= \sum_{n=j}^{N+J-2}\sum_{m=0}^{M+I-2} \mathbf{X}[m,n]\tilde{\mathbf{T}}[i-m,j-n] \\
&= F^{-1}\big\{F\{\mathbf{X}\}\cdot F(\{\tilde{\mathbf{T}}\})\big\}
\end{aligned} \tag{6.2}
$$

Die Faltung in Formel 6.2, in der F die Fourier-Transformation und F^{-1} deren Umkehrung berechnet, lässt sich also mittels der schnellen Fourier-Transformation effizient auswerten. Die Transformation des Bildes $\mathbf{X}$ und des Templates $\tilde{\mathbf{T}}$ in den Fourier-Raum hat jeweils eine Komplexität von $\mathcal{O}((I+M-1)(J+N-1)\log[(J+N-1)(I+M-1)])$. Für die Multiplikation im Fourier-Raum sind $(I+M-1)(J+N-1)$ Operationen notwendig, und die Rücktransformation des Produktes liegt in $\mathcal{O}((I+M-1)(J+N-1)\log[(J+N-1)(I+M-1)])$. Insgesamt hat man einen Aufwand von $(I+M-1)(J+N-1)(3\log[(J+N-1)(I+M-1)]+1)$.

Zur Berechnung der quadratischen euklidischen Distanz muss demnach der folgende Ausdruck ausgewertet werden:

$$
\begin{aligned}
D(i,j) &= \sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{X}^2[m,n] + \sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{T}^2[m-i,n-j] - 2\sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{X}[m,n]T[m-i,n-j] \\
&= \underbrace{\sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{X}^2[m,n]}_{\in\mathcal{O}(MN)} + \underbrace{\sum_{n=j}^{N+j}\sum_{m=i}^{M+i} \mathbf{T}^2[m-i,n-j]}_{\in\mathcal{O}(MN)} - \underbrace{2\cdot R_{\mathbf{XT}}(i,j)}_{\in\mathcal{O}\left((I+M)(J+N)\log((J+N)(I+M))\right)}
\end{aligned} \tag{6.3}
$$

Der erste Term in Formel 6.3 kann effizient mit dem im Abschnitt 6.2 vorgestellten Verfahren berechnet werden. Der zweite Term ist unabhängig von der Position und kann demnach vorweg bestimmt werden. Ebenso wird für den dritten Term ein einziges Mal die Faltung berechnet, und man erhält daraus die Werte für $R_{\mathbf{XT}}(i,j)$. Insgesamt erhält man folgende Komplexität:

$$
\begin{aligned}
&MN + IJ + 2(I+M-1)(J+N-1)\log((J+N-1)(I+M-1)) \\
\in&\mathcal{O}\big((I+M)(J+N)\log((J+N)(I+M))\big) \\
\in&\mathcal{O}\big(IJ\log(IJ)\big)
\end{aligned} \tag{6.4}
$$

Dies bedeutet, dass man durch die FFT in eine bessere Komplexitätsklasse fällt im Vergleich zu den Berechnungen in Formel 6.1.

6.2 Effiziente Berechnung von Quadratsummen

Für die Berechnungen der euklidischen Distanz in den Formeln 6.1 und 6.3 muss an jeder Position in der Bildszene die Quadratsumme der Pixel in der Projektionsfläche des Templates berechnet werden.

Dies ergibt bei Auswertung des Terms $\sum_{n=j}^{N+j}\sum_{m=i}^{M+i} S^2[m,n]$ an jeder Position (i,j) einen Aufwand von

$$
(I-M)\cdot(J-M)\cdot MN \in \mathcal{O}((IJ)^2). \tag{6.5}
$$

Eine effizientere Berechnung erreicht man, wenn vorweg eine Quadratsummenmatrix $M \in \mathbb{R}^{I\times J}$ von der Bildszene berechnet wird, die wie folgt definiert ist:

$$
\mathbf{M} := \begin{pmatrix}
\mathbf{X}^2[0,0] & \sum\limits_{n=0}^{1}\mathbf{X}^2[0,n] & \sum\limits_{n=0}^{2}\mathbf{X}^2[0,n] & \dots & \sum\limits_{n=0}^{J}\mathbf{X}^2[0,n] \\
\sum\limits_{m=0}^{1}\mathbf{X}^2[m,0] & \sum\limits_{m=0}^{1}\sum\limits_{n=0}^{1}\mathbf{X}^2[m,n] & \sum\limits_{m=0}^{1}\sum\limits_{n=0}^{2}\mathbf{X}^2[m,n] & \dots & \sum\limits_{m=0}^{1}\sum\limits_{n=0}^{J}\mathbf{X}^2[m,n] \\
\sum\limits_{m=0}^{2}\mathbf{X}^2[m,0] & \sum\limits_{m=0}^{2}\sum\limits_{n=0}^{1}\mathbf{X}^2[m,n] & \sum\limits_{m=0}^{2}\sum\limits_{n=0}^{2}\mathbf{X}^2[m,n] & \dots & \sum\limits_{m=0}^{2}\sum\limits_{n=0}^{J}\mathbf{X}^2[m,n] \\
\vdots & \vdots & \vdots & & \vdots \\
\sum\limits_{m=0}^{I}\mathbf{X}^2[m,0] & \sum\limits_{m=0}^{I}\sum\limits_{n=0}^{1}\mathbf{X}^2[m,n] & \sum\limits_{m=0}^{I}\sum\limits_{n=0}^{2}\mathbf{X}^2[m,n] & \dots & \sum\limits_{m=0}^{I}\sum\limits_{n=0}^{J}\mathbf{X}^2[m,n]
\end{pmatrix} \tag{6.6}
$$

Die Berechnung der Matrix $\mathbf{M}$ bedeutet einen einmaligen Rechenaufwand von $O(IJ)$. Die Quadratsumme an Position (i,j), d.h. der erste Term in Formel 6.1, ergibt sich aus folgender Berechnung:

$$
\begin{aligned}
&\mathbf{M}[i+M,j+N] - \mathbf{M}[i-1,j+N] - \mathbf{M}[i+M,j-1] + \mathbf{M}[i-1,j-1] \qquad (6.7)\\
=&\sum_{m=0}^{i+M}\sum_{n=0}^{j+N}\mathbf{X}^2[m,n] - \sum_{m=0}^{i-1}\sum_{n=0}^{j+N}\mathbf{X}^2[m,n] - \sum_{m=0}^{i+M}\sum_{n=0}^{j-1}\mathbf{X}^2[m,n] + \sum_{m=0}^{i-1}\sum_{n=0}^{j-1}\mathbf{X}^2[m,n] \\
=&\sum_{m=i}^{i+M}\sum_{n=0}^{j+N}\mathbf{X}^2[m,n] - \sum_{m=i}^{i+M}\sum_{n=0}^{j-1}\mathbf{X}^2[m,n] \\
=&\sum_{m=i}^{i+M}\sum_{n=j}^{j+N}\mathbf{X}^2[m,n]
\end{aligned}
$$

Das bedeutet, dass die Berechnung der Quadratsummen an einer Position (i, j) mit konstantem Zeitaufwand durchgeführt werden kann, vorausgesetzt die Quadratsummenmatrix ist schon erzeugt worden. Für die Berechnung aller Quadratsummen ist demnach nur noch ein Zeitaufwand von

$$IJ + 4 \cdot (I - M)(J - N) \in \mathcal{O}(IJ) \tag{6.8}$$

notwendig. In diesem Fall ist es effizienter, eine Quadratsummenmatrix vorwegzu berechnen als die Quadratsummen an jeder Position explizit auszurechnen.

6.3 Einschränkung des Suchraumes

Mit den oben beschriebenen Optimierungen gelangt man zwar in eine bessere Komplexitätsklasse, die Rechenzeit jedoch ist immer noch sehr hoch. Um die Laufzeit weiter zu verkürzen, wird der Suchraum eingeschränkt. Durch die direkte Modellierung der Translationen, Rotationen und Skalierungen ergibt sich eine große Menge an Hypothesen, die in Betracht gezogen werden müssen. Dadurch dass man die zu betrachtenden Abbildungen einschränkt, lassen sich das Training und die Klassifikation beschleunigen. Zum Beispiel kann die Anzahl der Skalierungstufen oder der zu betrachtenden Rotationen eingeschränkt werden. Dies bedeutet aber, dass eine Menge an Hypothesen unberücksichtigt bleibt und das globale Optimum nicht unbedingt gefunden wird.

Eine weitere Möglichkeit, den Suchraum einzuschränken, lässt sich durch die Betrachtung verschiedener Skalierungsstufen bei der Suche erreichen. In diesem Fall werden die Bildszenen in eine niedrigere Auflösung skaliert, so dass die Dimensionen des Bildes kleiner werden. In dieser niedrigeren Auflösung läuft die Suche wesentlich schneller, so dass mehr Abbildungen berücksichtigt werden können. Die beste Hypothese oder eine Menge der besten Hypothesen wird in der hohen Auflösung anschließend genauer betrachtet. Die Gefahr bei diesem Ansatz liegt darin, dass man in der niedrigeren Auflösungsstufe nicht notwendigerweiser in der Nähe des globalen Optimums liegt. In diesem Fall bringt die Betrachtung in der höheren Auflösungsstufe keine Verbesserung der Bewertung.

6.4 Zusammenfassung

In diesem Kapitel wird beschrieben, wie sich die Suche nach der besten Hypothese beschleunigen lässt. Es wird gezeigt, dass sich die „naive“ Berechnung der euklidischen Distanzen, im Falle der in dieser Arbeit verwendeten Algorithmen, wesentlich schneller durchführen lässt. Diese Beschleunigung wird durch eine Transformation in den Fourier-Raum erzielt, die effizient mit der FFT berechnet werden kann. Bei einer Bildszene der Dimension $I \times J$ und einem Template fester Dimension $M \times N$ liegt der Aufwand in $\mathcal{O}((IJ)\log(IJ))$.

Desweiteren lässt sich die Effizienz steigern, indem die Quadratsummen geschickt berechnet werden. Das in Abschnitt 6.2 vorgestellte Verfahren hat eine Komplexität von $\mathcal{O}(IJ)$.

In der Praxis zeigt sich jedoch, dass diese Beschleunigungsmaßnahmen nicht ausreichen, um das Training und die Klassifikation auf den gegebenen Trainings- und Testdaten in annehmbarer Zeit durchzuführen. Aus diesem Grunde müssen zusätzlich noch *Pruning*-Maßnahmen getroffen werden, die den Suchraum einschränken. Durch diese Einschränkungen ist das Finden des Optimums jedoch nicht mehr gewährleistet.

Kapitel 7

Ergebnisse

In diesem Kapitel werden die Ergebnisse präsentiert, die mit den in dieser Arbeit vorgestellten Verfahren erzielt wurden. Es gliedert sich in vier Abschnitte. Im ersten Abschnitt wird gezeigt, welchen Beschleunigungsgewinn man durch die Verwendung der FFT hat. Der nächste Abschnitt beschäftigt sich ausführlich mit den Ergebnissen des automatischen Objekttrainings. Wie die trainierten Prototypen sich auf die Klassifikation auswirken, wird im dritten Abschnitt dargestellt. Im vierten Abschnitt findet eine Analyse der Ergebnisse statt.

7.1 FFT zur schnellen Berechnung der euklidischen Distanz

Um den Unterschied der beiden im Abschnitt 6.1 beschriebenen Ansätze zur effizienten Berechnung von euklidischen Distanzen aufzuzeigen, wurde die jeweilige verbrauchte CPU-Zeit in der Praxis gemessen. Für die Zeitmessung der Funktionen bietet sich die '*profiling*'-Option des C++-Compilers 'g++' an. Mit Hilfe dieser Option ist es möglich, die Zeit zu messen, die ein Programm in einer bestimmten Funktion verbringt.

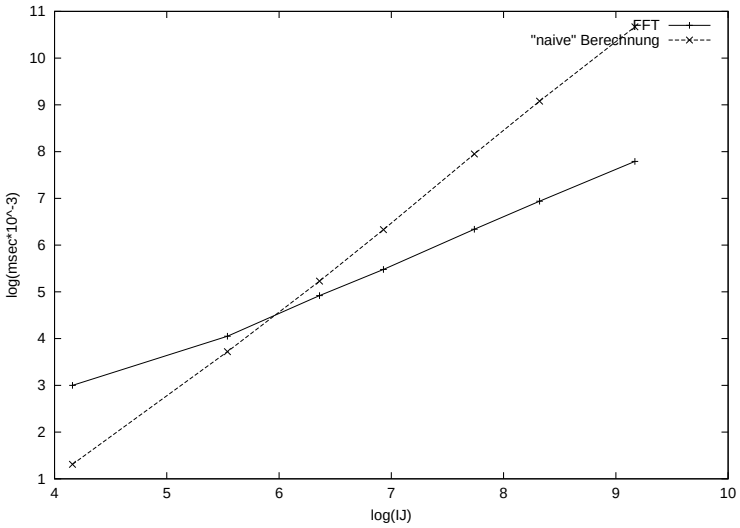


Abbildung 7.1: Beschleunigung durch Verwendung der FFT.

Die Messungen wurden auf einem Linux-System mit einem 1GHz-Athlon Prozessor und 128MB Hauptspeicher durchgeführt. Da die vorhandene Linux-Version nicht echtzeitfähig ist, wurden

die Messungen 1000 Mal wiederholt, und die mittleren Zeiten aus den einzelnen Durchläufen zum Vergleich verwendet.

In Abb. 7.1 sind die Ergebnisse der Messungen zusammengetragen. Auf der horizontalen Achse sind die getesteten Bildgrößen logarithmisch, auf der vertikalen Achse die gemessenen Zeiten logarithmisch in μs aufgetragen. Bei einer Bildgröße von ca. 20×20 Pixeln kreuzen sich die beiden Geraden. Ab dieser Bildgröße sollten die euklidischen Distanzen immer mittels der FFT berechnet werden.

7.2 Automatisches Objekttraining

Das automatische Segmentieren und Trainieren von Objekten ist keine leichte Aufgabe. Liegen die Objekte zusätzlich noch in ihrer realen Umgebung vor, wird die Aufgabe noch schwieriger. Für das automatische Objekttraining wurden in Kapitel 4 mehrere Verfahren vorgestellt, Objekt und Hintergrund zu modellieren. Durch die Verwendung eines iterativen Algorithmus werden schrittweise immer bessere Prototypen für das Objekt trainiert. Um das Verhalten der vorgestellten Verfahren genauer zu analysieren, wird mit einfachen Aufgabenstellungen angefangen. Das bedeutet, dass dem Trainingsalgorithmus zusätzliche Informationen über das zu trainierende Objekt mitgegeben wird. Nach und nach werden dann die Anforderungen erhöht, indem man dem Algorithmus Informationen vorenthält. Die Ergebnisse des automatischen Trainings auf der `COIL-I6`-Datenbank sind in den folgenden Abschnitten aufgelistet.

7.2.1 Ergebnisse auf der `COIL-I6-Train1`-Datenbank

Die ersten Experimente werden mit der `COIL-I6-Train1`-Datenbank durchgeführt. In dieser Datenbank liegen die Objekte auf homogenem, schwarzem Hintergrund (vgl. Abschnitt 2.3) vor. Diese Information kann direkt ausgenutzt werden, um die Parameter des Hintergrundmodells geeignet zu wählen.

Training mit einem 3-dimensionalen Rotationswinkel

Um das Training zusätzlich zu vereinfachen, wird auf die 3-dimensionale Rotation, die in den Bilddaten vorhanden ist, verzichtet. Aus dem Trainingskorpus wurden alle Trainingsbilder, die zur Klasse des 1. Objektes gehören und den gleichen 3D-Rotationswinkel von 0 Grad besitzen, herausgefiltert und zum Training verwendet.

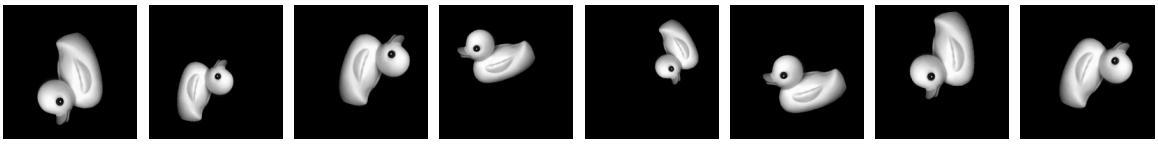


Abbildung 7.2: `COIL-I6-Train1`-Trainingsdaten des 1. Objektes mit einem 3D-Rotationswinkel von 0 Grad.

Für das Objekttraining stehen insgesamt 8 Trainingsbilder zur Verfügung, die in Abb. 7.2 zu sehen sind. Die Bilder haben eine Größe von 192x192 Pixeln und Intensitätswerte, die zwischen 0 und 1 liegen.

Da genau ein 3-dimensionaler Rotationswinkel in den Daten vorkommt, reicht es aus, eine multivariate Gauß-Verteilung zur Beschreibung des Objektes zu verwenden. Ferner lassen sich klare

Angaben über die Parameter der Hintergrundverteilung machen. Zur Bestimmung des Startmittelwertsvektors für das Objektmodell wurde ein Histogramm aus allen Trainingsdaten erzeugt. Der Mittelwertsvektor wird dann konstant auf den Pixelwert gesetzt, der für das Objekt am wahrscheinlichsten ist. Es ergab sich der Wert $\boldsymbol{\mu}_1 \equiv 0.83$ für den Mittelswertsvektor. Die Varianz wurde auf $\sigma_1^2 = 0,001$ gesetzt. Dies entspricht bei 256 Graustufen einer Standardabweichung von 33 Grauwerten.

Tabelle 7.1: Tabelle der auf `COIL-I6-Train1` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und univariater Gauß-Verteilung als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**	**5.**
$\boldsymbol{\mu}_1$						
σ_1^2	0,0100	0,0862	0,0563	0,0044	0,0036	0,0028

Im ersten Versuch wurde für den Hintergrund eine univariate Gauß-Verteilung verwendet. Die Ergebnisse sind in Tabelle 7.1 aufgelistet. In der ersten Zeile der Tabelle sind die Iterationsschritte zu sehen. Dabei bedeutet der 0. Iterationsschritt, dass die Startparameter verwendet werden. In der zweiten Zeile sind die in jedem Iterationsschritt neu geschätzten Prototypen des Objektes abgebildet. Die neu geschätzten Varianzen sind der dritten Zeile zu entnehmen.

Weil in der `COIL-I6-Train1`-Datenbank ein schwarzer Hintergrund modelliert werden soll, macht es keinen Sinn, die Parameter der Gauß-Verteilung zu trainieren. Die Varianz würde nach dem ersten Iterationsschritt auf 0 geschätzt werden, und die Gauß-Verteilung wäre somit entartet. Aus diesem Grund bleibt die Hintergrundverteilung in allen Iterationsschritten gleich. Mit diesen Modellen konvergiert das Verfahren in wenigen Iterationschritten gegen das gesuchte Objekt.

Im nächsten Versuch wurde für das Hintergrundmodell eine Gleichverteilung angenommen. Das bedeutet, dass jedes Pixel mit der gleichen Wahrscheinlichkeit dem Hintergrund zugeordnet werden kann. Für die Wahl der besten Hypothese in jedem Trainingsbild ist demnach das Vordergrundmodell ausschlaggebend. Auch in diesem Fall muss das Hintergrundmodell nicht trainiert werden. In der Tabelle 7.2 sind die Versuchsergebnisse aufgelistet.

Tabelle 7.2: Tabelle der auf `COIL-I6-Train1` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Gleichverteilung als Hintergrundmodell.

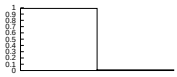
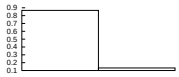
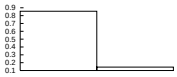
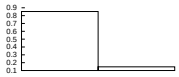
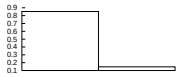
	0.	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**
$\boldsymbol{\mu}_1$										
σ_1^2	0,0100	0,0818	0,0326	0,0140	0,0112	0,0105	0,0076	0,0074	0,0074	0,0074

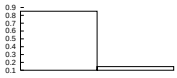
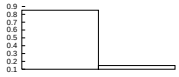
In diesem Fall wird das Objekt zwar gefunden, aber nicht vollständig segmentiert. Es liegt in der Natur der Gleichverteilung, dass Pixel, die zum Objekt gehören, dem Hintergrund zugeteilt werden können. Bei der Gleichverteilung ist die Bewertung eines Pixels immer die gleiche, egal ob dieses Pixel zum Objekt oder zum Hintergrund gehört. Dadurch dass zu Beginn keine genaue Beschreibung des Objektes vorliegt, sondern das Objekt nur mit einem konstanten Wert beschrieben wird, lässt sich das Objekt im ersten Iterationsschritt nicht genau beschreiben. Es werden nur Objekthypothesen generiert, die nicht das vollständige Objekt enthalten. Aus diesem Grund ist mit dieser Modellierung keine bessere Segmentierung zu erwarten.

In den nächsten beiden Experimenten werden verschiedene Histogramme getestet. Die Gleichverteilung kann als Histogramm mit einem Intervall aufgefasst werden. Deshalb beschreiben Histogramme die wahre Pixelverteilung der betrachteten Daten immer besser, je mehr Intervalle man zulässt. Dabei ist nicht klar, ob Histogramme mit vielen Intervallen im Vergleich zu Histogrammen mit wenigen Intervallen bessere Resultate im Training liefern.

Übertragen auf den Fall des homogenen, schwarzen Hintergrundes wird ein Histogramm mit mehreren Intervallen, Objektpixel und Hintergrundpixel nicht mehr gleich bewertet. Aus diesem Grund wird das Objekt mit zunehmenden Intervallen immer besser segmentiert werden. In der Tabelle 7.3 sind die Resultate eines Histogramms mit 2 Intervallen als Hintergrundmodell aufgelistet. Die dritte Zeile enthält diesmal die Realisierung des Hintergrundmodells.

Tabelle 7.3: Tabelle der auf `COIL-I6-Train1` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Histogramm mit 2 Intervallen als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,0100	0,0813	0,0321	0,0072	0,0039
$h^2(\cdot)$					

	5.	**6.**	**7.**	**8.**	**9.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,0031	0,0032			
$h^2(\cdot)$					

Aus den Ergebnissen wird deutlich, dass man mit Histogrammen mit zwei Intervallen noch nicht in der Lage ist, das Objekt vollständig zu segmentieren, wie dies bei der Gauß-Verteilung der Fall ist. Erst mit einer höheren Anzahl von Intervallen kann das Objekt vollständig segmentiert werden.

Zum Schluss wird ein Histogramm mit 64 Intervallen verwendet. In diesem Fall hat die geschätzte Hintergrundverteilung eine feinere Struktur im Vergleich zu Histogrammen mit einem Intervall. Durch diese feinere Strukturierung fällt die Bestrafung viel höher aus, wenn Objektpixel fälschlicherweise dem Hintergrund zugeordnet werden. Das Objekt wird daher auch komplett gefunden. In der Tabelle 7.4 sind die Ergebnisse aufgelistet.

Um einen Gesamtüberblick zu geben, wie gut die Objekte mittels der unterschiedlichen Hintergrundmodelle gefunden werden konnten, sind in Tabelle 7.5 alle durchgeführten Trainingsvarianten und deren Resultate zusammengestellt. In der Tabelle sind in der linken Spalte die verwendeten Hintergrundmodelle und in der ersten Zeile die Startparameter angegeben. Insgesamt kann man festhalten, dass die Objekte gefunden und gute Prototypen erstellt wurden. Dabei spielte im Fall des homogenen Hintergrundes die Wahl der Startparameter kaum eine Rolle.

In den nächsten Experimenten werden die Anforderungen an das Training erhöht, indem mehr 3-dimensionale Rotationswinkel zugelassen werden.

Tabelle 7.4: Tabelle der auf `COIL-I6-Train1` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Histogramm mit 64 Intervallen als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,01	0,0711	0,0553	0,0315	0,026

	5.	**6.**	**7.**	**8.**	**9.**
$\boldsymbol{\mu}_1$					
σ_1^2	0.017	0,0074	0,0053	0,0027	0,0027

Tabelle 7.5: Gegenüberstellung der auf `COIL-I6-Train1` trainierten Prototypen des 1. Objektes mit festem 3D-Rotationswinkel von 0 Grad.

	Rotation bekannt		**Skalierung bekannt**		**keine Vorgaben**	
Startprototyp						
Gauß-Verteilung						
Gleichverteilung						
$h^2(\cdot)$						
$h^{64}(\cdot)$						

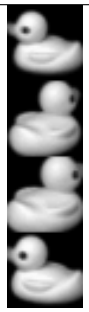
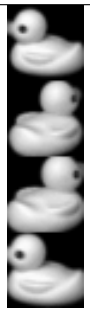
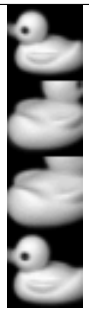
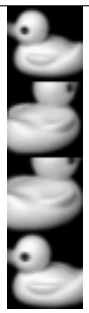
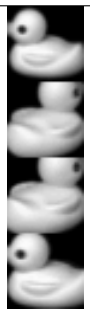
Training mit vier 3-dimensionalen Rotationswinkeln

Um die Trainingsaufgabe zu erschweren, wurden für die nächsten Experimente Trainingsdaten verwendet, die insgesamt vier 3-dimensionale Rotationswinkel enthalten. Diese vier Rotationswinkel lassen sich nicht mehr mit einer unimodalen Verteilung erfassen, so dass für dieses Training auf die Gauß'schen Mischverteilungen zurückgegriffen wird.

Die vier Rotationswinkel wurden so ausgewählt, dass die ersten beiden und die letzten beiden Winkel so nah wie möglich beieinander liegen. In der `COIL-I6`-Datenbank bedeutet dies, dass sich die Objektansichten um 10 Grad voneinander unterscheiden. So kann gleichzeitig getestet werden, ob der Algorithmus nahe beieinanderliegende Winkel differenzieren kann.

Insgesamt wurden die gleichen Testserien durchgeführt wie im vorigen Abschnitt. Die trainierten Prototypen sind in Tabelle 7.6 in Form der Dichten der Gauß'schen Mischverteilung dargestellt.

Tabelle 7.6: Gegenüberstellung der auf `COIL-I6-Train1` trainierten Prototypen mit vier festen 3D-Rotationswinkeln des 1. Objektes.

	Rotation bekannt		**Skalierung bekannt**		**keine Vorgaben**	
Startprototyp						
Gauß-Verteilung						
Gleichverteilung						
$h^2(\cdot)$						
$h^{64}(\cdot)$						

Zusammenfassend lässt sich sagen, dass die Gauß-Verteilung und die Histogramme mit 64 Intervallen die besten Resultate liefern. Die Gleichverteilung und die Histogramme mit 2 Intervallen haben das Problem, dass die Objekte nicht vollständig segmentiert werden. Dies hängt damit zusammen, dass die Zuweisung von Objektpixeln zum Hintergrund bei der Gleichverteilung gar nicht und bei dem Histogramm mit 2 Intervallen erst ab einer Pixelintensität von 0.5 bestraft

wird. Der Algorithmus ist auch in der Lage, kleine 3D-Rotationen voneinander zu trennen.

Eine objektive Bewertung der Güte der trainierten Prototypen lässt sich nur mit einer Klassifikation durchführen (siehe Abschnitt 7.3)

7.2.2 Ergebnisse auf der `COIL-I6-Train2`-Datenbank

Mit der `COIL-I6-Train2`-Datenbank wird das Training schwieriger, da es keinen homogenen Hintergrund mehr gibt. Die Startparameter für das Hintergrundmodell lassen sich a-priori nicht mehr so einfach festlegen. Aus diesem Grund haben sie im Vergleich zum Training auf der `COIL-I6`-Datenbank eine wesentlich größere Bedeutung. Um die Auswirkungen der Wahl der Startparameter auf das Training zu untersuchen, werden die ersten Experimente unter den vereinfachten Voraussetzungen durchgeführt (vgl. Abschnitt 7.2.1).

Training mit einem 3-dimensionalen Rotationswinkel

In den ersten Versuchen wird auf die Betrachtung der 3-dimensionalen Rotation verzichtet. Aus allen Trainingsdaten werden die Bilder zum Training herausgefiltert, die das Objekt 7 beinhalten und in einem 3-dimensionalen Rotationswinkel von 27 Grad vorliegen. In Abbildung 7.3 sind die verwendeten Trainingsdaten zu sehen. Es stehen 8 Trainingsbilder mit einer Größe von 192x192 Pixeln zur Verfügung, deren Intensitätswerte zwischen 0 und 1 liegen.

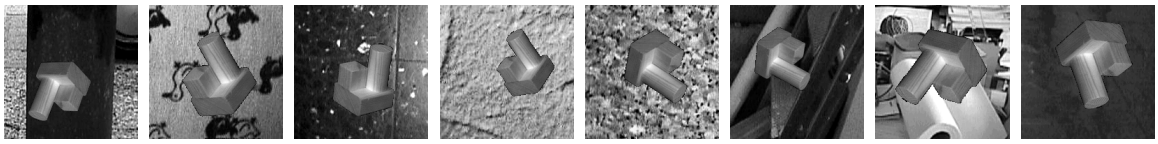


Abbildung 7.3: `COIL-I6-Train2`-Trainingsdaten des 7. Objektes mit einem 3D-Rotationswinkel von 27 Grad.

Dadurch dass genau eine 3D-Rotation vorliegt, kann für das Objektmodell auf eine multivariate Gauß-Verteilung zurückgegriffen werden. Im Folgenden sind die Ergebnisse aufgezeigt, die bei verschiedenen Startparametern und Hintergrundmodellen entstehen.

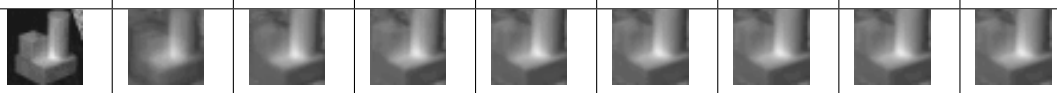
Für die ersten Versuche wird das erste Bild aus den Trainingsdaten manuell segmentiert, und das ausgeschnittene Objekt als Startparameter des Algorithmus verwendet. Man erhält zwar eine gute Beschreibung des Objektes, aber durch die Voraussetzung, dass das Objekt in einem Quadrat liegen muss, werden Pixel dem Objekt zugeschrieben, die eigentlich zum Hintergrund gehören. Diese Hintergrundpixel sind keine gute Beschreibung für das Objekt und können von Trainingsbild zu Trainingsbild sehr stark variieren und dadurch die Lokalisation des Objektes negativ beeinflussen.

Dennoch ist durch dieses manuell segmentierte Bild eine Möglichkeit gegeben, schnell und einfach eine Beschreibung des Objektes zu finden. Man hofft, dass sich die Hintergrundpixel, die im Objektquadrat liegen, durch die Menge der Trainingsdaten glätten und bei der Lokalisation nicht mehr so stark ins Gewicht fallen.

Um das Training weiter zu vereinfachen, wurden die 2-dimensionalen Rotationen als bekannt vorausgesetzt, so dass lediglich Translationen und Skalierungen richtig gefunden werden müssen.

Als erstes wird eine univariate Gauß-Verteilung als Hintergrundmodell verwendet. Die Startparameter dieser Gauß-Verteilung lassen sich aus allen Trainingsdaten schnell berechnen. Die Ergebnisse des Trainings sind in Tabelle 7.7 festgehalten.

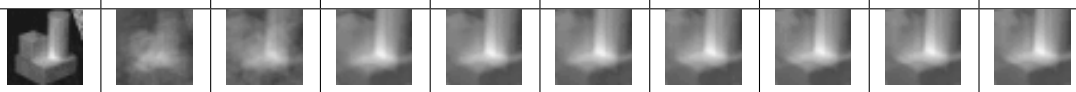
Tabelle 7.7: Tabelle der auf `COIL-I6-Train2` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Gauß-Verteilung als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**
$\boldsymbol{\mu}_1$									
σ_1^2	0,0100	0,0156	0,0095	0,0074	0,0063	0,0063	0,0063	0,0063	0,0063
μ_0	0,3982	0,4018	0,3998	0,3997	0,4001	0,4000	0,4000	0,4000	0,4000
σ_0^2	0,0304	0,1481	0,1470	0,1470	0,1472	0,1472	0,1472	0,1472	0,1472

Mit einem manuell segmentierten Objekt als Startparameter lässt sich nach 8 Iterationen das Objekt im Prototypen wiederfinden. Die anfänglich sehr dunkelen Pixel, die zum Hintergrund gehörten, wurden mit der Zeit herausgemittelt und das Objekt gefunden.

Der Tabelle 7.8 sind die Ergebnisse zu entnehmen, die man erhält, wenn als Hintergrundmodell eine Gleichverteilung verwendet wird.

Tabelle 7.8: Tabelle der auf `COIL-I6-Train2` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Gleichverteilung als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**
$\boldsymbol{\mu}_1$										
σ_1^2	0,0100	0,0337	0,0192	0,0117	0,0110	0,0104	0,0098	0,0100	0,0099	0,0099

Als Verfeinerung der Gleichverteilung werden im Folgenden die Histogramme als Hintergrundmodell getestet. Es wird zum einen ein Histogramm mit 2 Intervallen und zum anderen ein Histogramm mit 64 Intervallen getestet. Die Ergebnisse der Histogramme mit 2 Intervallen findet man in Tabelle 7.9, die Ergebnisse der Histogramme mit 64 Intervallen in Tabelle 7.10.

Tabelle 7.9: Tabelle der auf `COIL-I6-Train2` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Histogramm mit 2 Intervallen als Hintergrundmodell.

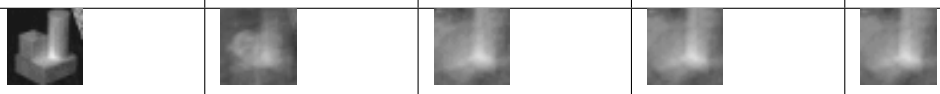
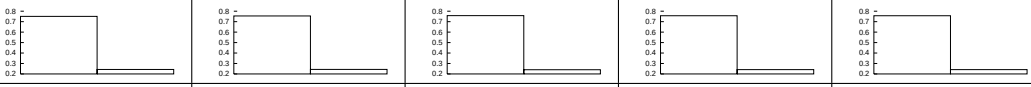
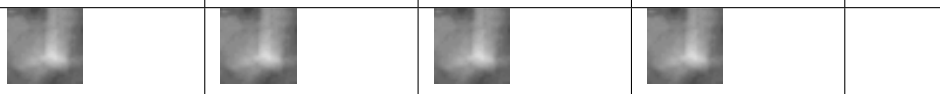
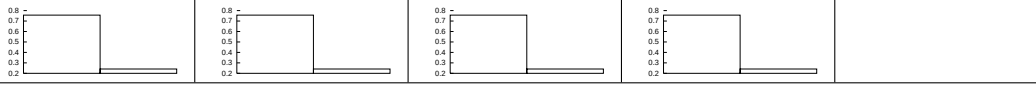
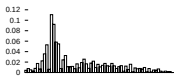
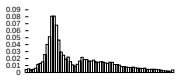
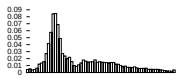
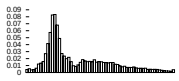
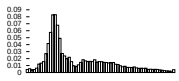
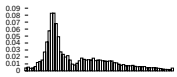
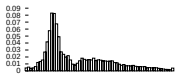
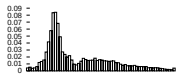
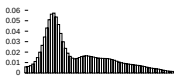
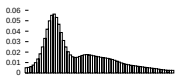
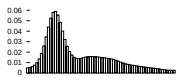
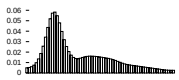
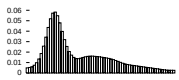
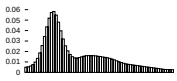
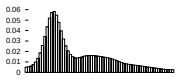
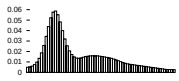
	0.	**1.**	**2.**	**3.**	**4.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,0100	0,0324	0,0174	0,0166	0,0166
$h^2(\cdot)$					
	5.	**6.**	**7.**	**8.**	
$\boldsymbol{\mu}_1$					
σ_1^2	0,0166	0,0166	0,0166	0,0166	
$h^2(\cdot)$					

Tabelle 7.10: Tabelle der auf `COIL-I6-Train2` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und Histogramm mit 64 Intervallen als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,0100	0,0306	0,0130	0,0134	0,0128
$h^{64}(\cdot)$					
	5.	**6.**	**7.**		
$\boldsymbol{\mu}_1$					
σ_1^2	0,0125	0,0122	0,0122		
$h^{64}(\cdot)$					

Werden Histogramme mit vielen Intervallen trainiert, kann es vorkommen, dass einige Intervalle nicht belegt werden. Dies kann bei der Distanzberechnung zu unerwünschten Nebeneffekten führen, wenn der Logarithmus aus einer sehr kleinen Zahl berechnet wird bzw. eine Wahrscheinlichkeitsdichte von 0 auftritt. Aus diesem Grund werden die Histogramme $h_s^B(\cdot)$ geglättet. Die Ergebnisse, die bei geglätteten Histogrammen herauskommen, sind in Tabelle 7.11 aufgelistet.

Tabelle 7.11: Tabelle der auf `COIL-I6-Train2` trainierten Prototypen mit multivariater Gauß-Verteilung als Vordergrundmodell und geglättetem Histogramm mit 64 Intervallen als Hintergrundmodell.

	0.	**1.**	**2.**	**3.**	**4.**
$\boldsymbol{\mu}_1$					
σ_1^2	0,0100	0,0294	0,0125	0,0134	0,0128
$h_s^{64}(\cdot)$					
	5.	**6.**	**7.**		
$\boldsymbol{\mu}_1$					
σ_1^2	0,0125	0,0122	0,0122		
$h_s^{64}(\cdot)$					

Dies sind die Experimente, in denen ein manuell segmentiertes Bild als Startparameter verwendet wurde. Das gesuchte Objekt lässt sich in den Prototypen bei allen Modellen wiederfinden. Eine Tendenz über die Güte der trainierten Prototypen, die sich später noch bestätigen wird, lässt sich im Fall der vorgegebenen Rotation schon ausmachen. Wird eine Gauß-Verteilung als Hinter-

grundmodell eingesetzt, wird das Objekt am besten gefunden und trainiert. Die Gleichverteilung als Modell ist eher ungeeignet, da diese zuviel Spielraum für die Wahl der Hintergrundpixel zulässt.

In den weiteren Experimenten werden dem Trainingsalgorithmus weniger Informationen über das Objekt zur Verfügung gestellt. Der Mittelwertsvektor des Objektmodells wird auf einen konstanten Wert gesetzt. Dieser Wert ist die wahrscheinlichste Pixelintensität des Objektes und wurde aus einem Histogramm des Objektes ermittelt. Für das Objekt ergab sich eine Pixelintensität von μ =0,37, und für die Varianz wurde ein Wert von σ^2 =0.01 angenommen.

Eine Übersicht über alle trainierten Prototypen ist in Tabelle 7.12 zu sehen. Um diese Ergeb-

Tabelle 7.12: Gegenüberstellung der auf `COIL-I6-Train2` trainierten Prototypen des 7. Objektes mit festem 3D-Rotationswinkel von 37 Grad.

	Rotation bekannt		**Skalierung bekannt**		**keine Vorgaben**	
Startprototyp						
Gauß-Verteilung						
Gleichverteilung						
$h^2(\cdot)$						
$h^{64}(\cdot)$						
$h_s^{64}(\cdot)$						

nisse zu erzeugen, wurde bewusst das Objekt 7 ausgewählt, um zu veranschaulichen, wie sich das Training in einem schwierigen Fall verhält. Mit diesem Objekt ist gleichzeitig die Situation gegeben, dass die mittlere Pixelintensität des Objektes fast gleich der mittleren Pixelintensität des Hintergrundes ist. Die mittlere Intensität des Objektes liegt bei 0.37, die des Hintergrundes bei 0,398. Dadurch dass Objekt und Hintergrund ähnliche Intensitätswerte besitzten, ist eine Differenzierung sehr schwierig. In Tabelle 7.13 sind die Prototypen aufgelistet, die für Objekt 1 trainiert wurden. Mit einem mittleren Pixelwert von 0,87 des Objektes 1 und einer mittleren Hintergrundintensität von 0,391 ist die Differenz wesentlich größer als bei Objekt 7. Wenn man die Ergebnisse aus Tabelle 7.12 und 7.13 insgesamt betrachtet, fällt auf, dass es den Prototypen an Schärfe fehlt. Dies liegt daran, dass das Objekt in einigen Trainingsdaten nicht richtig gefunden wurde. Die besten Resultate ergeben sich mit der Gauß-Verteilung als Hintergrundmodell. In diesen Prototypen ist das Objekt deutlich zu erkennen. Es zeigt sich aber auch schon das Phänomen, dass ein manuell segmentierter Prototyp als Startwert nicht notwendigerweise bessere Resultate liefert als ein Startwert mit konstanter Pixelintensität. Durch die Voraussetzung, dass ein Prototyp eine quadratische Form haben muss, sind sowohl Hintergrund- als auch Objektpixel im Protoyp enthalten, was zu einer falschen Lokalisation des Objektes führen kann.

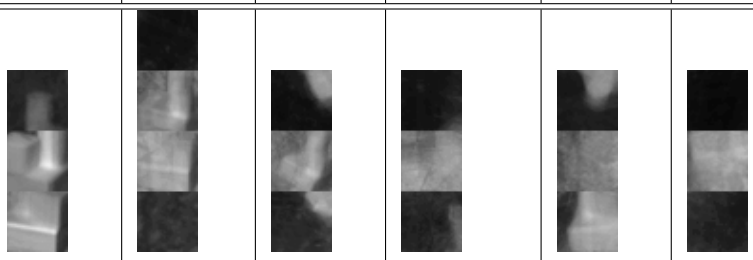
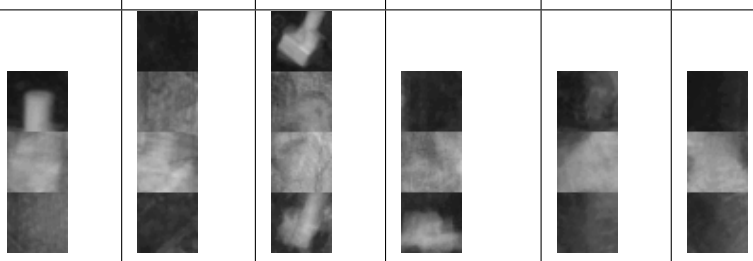
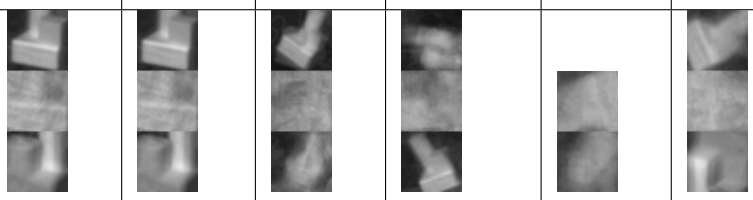
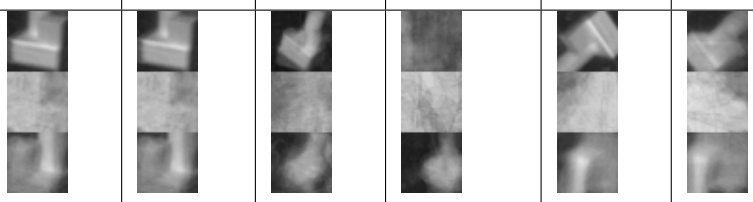
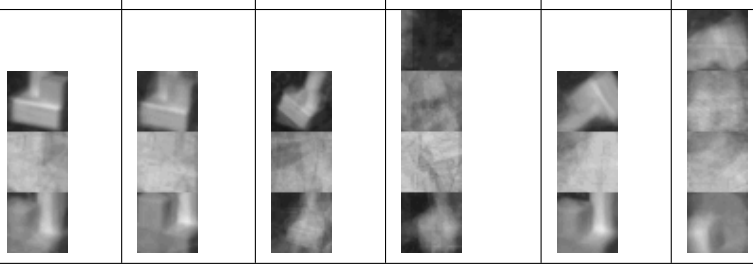
Tabelle 7.13: Gegenüberstellung der auf `COIL-I6-Train2` trainierten Prototypen des 1. Objektes mit festem 3D-Rotationswinkel von 0 Grad.

	Rotation bekannt		**Skalierung bekannt**		**keine Vorgaben**	
Startwert						
Gauß-Verteilung						
Gleichverteilung						
$h^2(\cdot)$						
$h^{64}(\cdot)$						
$h_s^{64}(\cdot)$						

Training mit vier 3-dimensionalen Rotationswinkeln

In den Experimenten in diesem Abschnitt wurden die Anforderungen an das Training weiter gesteigert. Dies wurde erreicht, indem vier 3-dimensionale Rotationswinkel in den Trainingsdaten auftraten. Es wurden analog zu Abschnitt 7.2.1 zwei Paare untersucht, die sich vom Rotationswinkel stark unterscheiden. Die Rotationswinkel der Objekte jedes Paares liegen dagegen so nah wie möglich (d.h. 10 Grad) beieinander. Dies lässt eine Analyse zu, wie gut ähnliche Winkel differenziert werden können. Wie in Abschnitt 7.2.1 gesehen, ist es bei homogenem Hintergrund möglich, Winkel, die nahe beieinander liegen, zu unterscheiden. In Tabelle 7.14 sind die Ergebnisse mit realem Hintergrund aufgezeigt. Gibt man zum Training als Startwert lediglich einen konstanten Wert an, hat der Trainingsalgorithmus es schwer, das Objekt zu lokalisieren. Die Objektbeschreibung ist zu ungenau, und es werden hauptsächlich homogene Flächen gefunden, was der Struktur des Startprototypen entspricht. Der Anteil der gefundenen Objekte steigt mit Hinzugabe von Information an den Algorithmus. Wird als Startwert ein manuell segmentierter Prototyp verwendet, fällt auf, dass das segmentierte Objekt in den Prototypen gefunden wird, die anderen Erscheinungsformen des Objektes jedoch nicht. Die Startinformation ist demnach immer noch zu ungenau für die anderen Erscheinungsformen des Objektes. Dies liegt auch darin begründet, dass sich verschiedene 2D-Ansichten eines 3D-Objektes stark unterscheiden können. Diese Tatsache wird auch als sensorische Lücke (*sensoric gap*) bezeichnet[21].

Tabelle 7.14: Gegenüberstellung der auf `COIL-I6-Train2` trainierten Prototypen mit vier festen 3D-Rotationswinkeln des 7. Objektes.

	Rotation bekannt		**Skalierung bekannt**		**keine Vorgaben**	
Startprototyp						
Gauß-Verteilung						
Gleichverteilung						
$h^2(\cdot)$						
$h^{64}(\cdot)$						
$h_s^{64}(\cdot)$						

7.3 Klassifikation

7.3.1 Klassifikationsergebnisse auf COIL-20

Die in Abschnitt 2.3 erstellten Datenbanken COIL-I6 basieren auf den originalen COIL-20 Daten. Das bedeutet, dass bei der Generierung der Korpora keine neuen Informationen über die Objekte hinzugekommen sind. Aus diesem Grund können die Klassifikationsfehlerraten auf COIL-I6 nicht besser sein als auf den originalen Daten. Die Fehlerraten auf der COIL-20-Datenbank, die mit den Verfahren aus dieser Arbeit erzeugt wurden, werden daher als Referenz für die Fehlerraten auf den COIL-I6-Datenbanken genommen.

Tabelle 7.15: In dieser Tabelle sind die Ergebnisse auf der COIL-20-Datenbank aufgelistet

Parameter	**Fehlerrate [%]**
NN,Energienormierung, Handicap-Reject-Schwelle 1:1 [12]	0
KD, $\sigma_0 = 0,001$, $\mu_0 = 0$, $\sigma_1 = 0.001$, COIL-20 Trainingsdaten	0

Das Ergebnis der letzten Zeile in Tabelle 7.15 wurde mit dem in Abschnitt 3.2.3 vorgestellten Ansatz erreicht. Für das Hintergrundmodell wurde eine Gauß-Verteilung verwendet. Mit dem Wissen, dass die Objekte auf einem schwarzen, homogenen Hintergrund positioniert sind, haben sich die Parameter σ_0 = 0,001, μ_0=0 als optimal herausgestellt. Als Varianz für die Objektmodelle hat sich eine Varianz von σ_1 =0.001 als gut erwiesen.

Es wurde eine Fehlerrate von 0% erreicht, so dass auf den Testdatenbanken COIL-I6-Test(x) mit diesem Verfahren eine Fehlerrate von 0% theoretisch möglich ist.

Auf dem originalen COIL-20-Testkorpus wurden weitere Experimente durchgeführt, um die Güte automatisch trainierter Prototypen auf den originalen Testdaten zu bewerten. Diese Prototypen wurden auf dem COIL-20-Train1-Korpus mit einer Gauß-Verteilung als Hintergrundmodell trainiert. Weil der Hintergrund homogen schwarz ist, wird als Mittelwert μ_0=0 und als Varianz σ_0^2=0,001 verwendet. Das Training wurde mit einem konstanten Wert $\boldsymbol{\mu}_1$ ≡0,87 als Startprototyp begonnen. Diese Resultate sind in Tabelle 7.16 zu sehen.

Tabelle 7.16: In dieser Tabelle sind die Klassifikationsergebnisse auf der COIL-20-Test1-Datenbank aufgelistet. Zum Training wurden die COIL-20-Trainingsdaten verwendet.

Trainingskorpus	**Hintergrund-modell**	**Parameter**	**Fehlerrate [%]**
COIL-20-Train1 (v.R.)	Gauß-Verteilung	$\sigma_0 = 0,001$, $\mu_0 = 0$, $\sigma_1 = 0.001$	4,4
COIL-20-Train1	Gauß-Verteilung	$\sigma_0 = 0,001$, $\mu_0 = 0$, $\sigma_1 = 0.001$	7,8

7.3.2 Klassifikationsergebnisse auf COIL-I6-Test1

Analog zur Vorgehensweise beim Training werden zur Trennung der Effekte bei den Klassifikationen die Anforderungen an den Klassifikator im Verlauf der Experimente steigen. Die ersten Versuche bestehen darin, zu testen, wie gut die Klassifikation auf der COIL-20-Test1-Datenbank ohne Training der Objekte verläuft. Aus diesem Grund wird der in Abschnitt 3.2.3 vorgestellte, nicht parametrische Ansatz der *Kernel-Densities* verwendet. Die Dichten setzen sich aus den

Tabelle 7.17: In dieser Tabelle sind die Klassifikationsergebnisse auf der COIL-I6-Test1-Datenbank aufgelistet. Zum Training wurden die COIL-20-Trainingsdaten verwendet.

Trainingskorpus	**Hintergrund-modell**	**Parameter**	**Bemerkung**	**Fehlerrate [%]**
COIL-20 KD	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$	Rotation bekannt	1,1
COIL-20 KD	Gleichvert.	$\sigma_1 = 0.001$	Rotation bekannt	1,1
COIL-20 KD	$h^{64}(\cdot)$	$\sigma_1 = 0.001$	Rotation bekannt	0
COIL-20 KD	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$	Skalierung bekannt	21,1
COIL-20 KD	Gleichvert.	$\sigma_1 = 0.001$	Skalierung bekannt	16,1
COIL-20 KD	$h^{64}(\cdot)$	$\sigma_1 = 0.001$	Skalierung bekannt	15,6
COIL-20 KD	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$		24,4
COIL-20 KD	Gleichvert.	$\sigma_1 = 0.001$		85,6
COIL-20 KD	$h^{64}(\cdot)$	$\sigma_1 = 0.001$		33,3

originalen COIL-20-Daten und der Varianz $\sigma_1 = 0.001$, die sich als günstig erwiesen hat, zusammen. Weiterhin wurden, um die Anforderungen herunterzusetzen, die Rotationen als bekannt vorausgesetzt. In diesem Fall erreicht man eine Fehlerrate von 0%, wenn Histogramme als Hintergrundmodell eingesetzt werden. Ein anderes Mal wurde die Skalierung als bekannt vorausgesetzt. Im besten Fall fällt die Fehlerrate auf 15,6%. Zuletzt wurden weder Rotationen noch Skalierungen bekannt gegeben. Mit einer Gauß-Verteilung als Hintergrundmodell wurde eine Fehlerrate von 24,4% erzielt. Um die Laufzeit für die Klassifikation des Testkorpus akzeptabel zu halten, wurde die Suche auf 36 Rotationswinkel in 10-Grad-Abständen und 30 Skalierungsstufen beschränkt. Mit diesen Begrenzungen des Suchraumes wurden für die Klassifikation des Testkorpus 4 Tage Rechenzeit[1] benötigt. Die Zusammenfassung aller Ergebnisse der Experimente wird in Tablle 7.17 gezeigt.

Mit den nächsten Klassifikationsergebnissen wird die Qualität untersucht, wie gut die Prototypen, die auf der COIL-I6-Train1-Datenbank mit vorgegebenen Rotationen (v.R.) trainiert wurden, für die Klassifikation geeignet sind. Die Prototypen wurden mit einer Gauß-Verteilung als Hintergrundmodell trainiert, die den Mittelwert μ_0=0 und die Varianz σ_0^2=0,001 hatte. Trainiert wurde mit dem in Kapitel 4 vorgestellten iterativen Algorithmus und einer Gauß'schen Mischverteilung als Objektmodell.

Um die Anforderungen zu variieren und die Hintergrundmodelle gegeneinander zu testen, wurden die Klassifikationen in verschiedenen Modi durchgeführt. Die Resultate sind in Tabelle 7.18 zusammengefasst.

Es fällt auf, dass in sämtlichen Versuchen die Gauß-Verteilung als Hintergrundmodell deutlich bessere Klassifikationsergebnisse liefert als die anderen Modelle. Man kann aus diesen Ergebnissen jedoch nicht auf eine Überlegenheit der Gauß-Verteilung schließen. Die schlechten Fehlerraten der anderen Modelle haben jedoch möglicherweise ihre Ursache darin, dass die Prototypen mit der Gauß-Verteilung als Hintergrundmodell trainiert wurden. Verwendet man für die Klassifikation ein anderes Hintergrundmodell als für das Training, kommt es zu Unterschieden bei der Suche

[1] Die Klassifikationen wurden auf einem 1GHz-Intel Prozessor mit 2GB Hauptspeicher durchgeführt.

Tabelle 7.18: In dieser Tabelle sind die Klassifikationsergebnisse auf der COIL-I6-Test1-Datenbank aufgelistet. Zum Training wurden die COIL-I6-Train1-Trainingsdaten mit vorgegebenen Rotationen (v.R.) verwendet.

Trainingskorpus	**Hintergrund-modell**	**Parameter**	**Bemerkung**	**Fehlerrate [%]**
COIL-20-Train1 (v.R.)	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$	Rotation bekannt	7,8
COIL-20-Train1 (v.R.)	Gleichvert.	$\sigma_1 = 0.001$	Rotation bekannt	88,9
COIL-20-Train1 (v.R.)	$h^{64}(\cdot)$	$\sigma_1 = 0.001$	Rotation bekannt	41,7
COIL-20-Train1 (v.R.)	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$	Skalierung bekannt	65,6
COIL-20-Train1 (v.R.)	Gleichvert.	$\sigma_1 = 0.001$	Skalierung bekannt	40,6
COIL-20-Train1 (v.R.)	$h^{64}(\cdot)$	$\sigma_1 = 0.001$	Skalierung bekannt	22,8
COIL-20-Train1 (v.R.)	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$		20
COIL-20-Train1 (v.R.)	Gleichvert.	$\sigma_1 = 0.001$		92,2
COIL-20-Train1 (v.R.)	$h^{64}(\cdot)$	$\sigma_1 = 0.001$		81,7

nach der besten Hypothese. Bei unterschiedlichen Modellen werden unterschiedliche Hypothesen generiert (vgl. die Ergebnisse in Tabellen 7.12, 7.13, 7.14).

Tabelle 7.19: In dieser Tabelle sind die Klassifikationsergebnisse auf der COIL-I6-Test1-Datenbank aufgelistet. Zum Training wurden die COIL-I6-Train1-Trainingsdaten verwendet.

Trainingskorpus	**Hintergrund-modell**	**Parameter**	**Bemerkung**	**Fehlerrate [%]**
COIL-20-Train1	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$	Skalierung bekannt	29,4
COIL-20-Train1	Gleichvert.	$\sigma_1 = 0.001$	Skalierung bekannt	30,6
COIL-20-Train1	$h^{64}(\cdot)$	$\sigma_1 = 0.001$	Skalierung bekannt	37,2
COIL-20-Train1	Gauß-Vert.	$\sigma_0 = 0,001$, $\mu_0 = 0$ $\sigma_1 = 0.001$		33,9
COIL-20-Train1	Gleichvert.	$\sigma_1 = 0.001$		92,8
COIL-20-Train1	$h^{64}(\cdot)$	$\sigma_1 = 0.001$		54,4

Für die folgenden Klassifikationsergebnisse wurden Prototypen wie auf der COIL-I6-Train1-Datenbank trainiert. Um die Anforderungen an das Training zu erhöhen, wurden diesmal die Rotationswinkel nicht vorgegeben. Daraus folgt gleichzeitig, dass die Rotationen für die Klassifikation nicht als bekannt vorausgesetzt werden können. Die Prototypen werden automatisch trainiert und sind daher nicht an bekannten Rotationswinkeln ausgerichtet. Es ist daher unmöglich,

die Rotationen der Testdaten mit den Trainingsdaten anzupassen. In Tabelle 7.19 sind die Ergebnisse zusammengefasst. An den Ergebnissen lassen sich die gleichen Beobachtungen machen wie in Tabelle 7.18. Die Gauß-Verteilung als Hintergrundmodell hat die besseren Klassifikationsergebnisse, weil die Prototypen ebenfalls mit einer Gauß-Verteilung als Hintergrundmodell trainiert wurden. Es scheint daher nötig, diesen Effekt durch entsprechende Trainingsdurchläufe auszuschalten. Im Rahmen dieser Arbeit war dies leider aufgrund der Zeitbeschränkung nicht möglich.

Für alle Klassifikationsergebnisse gilt, dass die Objektsuche in einem stark eingeschränkten Suchraum stattgefunden hat, damit die benötigte Rechenzeit niedrig gehalten wird. Eine Vorauswahl der besten Hypothese hat auf 10% der originalen Bildgröße stattgefunden. Die Anzahl der Rotationen wurde auf 60 beschränkt, so dass in 6-Grad-Schritten alle Rotationen erfasst werden. Die besten Hypothesen wurden anschließend in voller Auflösung erneut getestet, wobei nochmal 5 Rotationswinkel in unmittelbarer Umgebung berücksichtigt wurden.

7.3.3 Klassifikationsergebnisse auf `COIL-I6-Train2`

Die Klassifikationsergebnisse in diesem Abschnitt wurden alle auf dem `COIL-I6-Train2`-Korpus berechnet. Trainiert wurden die Prototypen einmal mit vorgegebenen Rotationen (v.R.), ein

Tabelle 7.20: In dieser Tabelle sind die Klassifikationsergebnisse auf der `COIL-I6-Test2`-Datenbank aufgelistet. Zum Training wurden die `COIL-I6-Train2`-Trainingsdaten mit/ohne vorgegebenen Rotationen (v.R.) verwendet.

Trainingskorpus	**Hintergrundmodell**	**Bemerkung**	**Fehlerrate [%]**
`COIL-20-Train2` (v.R.)	Gauß-Verteilung	Rotation bekannt	92,8
`COIL-20-Train2` (v.R.)	Gleichverteilung	Rotation bekannt	93,3
`COIL-20-Train2` (v.R.)	$h^{64}(\cdot)$	Rotation bekannt	93,3
`COIL-20-Train2` (v.R.)	Gauß-Verteilung	Skalierung bekannt	89,4
`COIL-20-Train2` (v.R.)	Gleichverteilung	Skalierung bekannt	95,2
`COIL-20-Train2` (v.R.)	$h^{64}(\cdot)$	Skalierung bekannt	91,1
`COIL-20-Train2` (v.R.)	Gauß-Verteilung		89,4
`COIL-20-Train2` (v.R.)	Gleichverteilung		95
`COIL-20-Train2` (v.R.)	$h^{64}(\cdot)$		90,1
`COIL-20-Train2`	Gauß-Verteilung	Skalierung bekannt	93,3
`COIL-20-Train2`	Gleichverteilung	Skalierung bekannt	90,5
`COIL-20-Train2`	$h^{64}(\cdot)$	Skalierung bekannt	92,2
`COIL-20-Train2`	Gauß-Verteilung		91,1
`COIL-20-Train2`	Gleichverteilung		92,8
`COIL-20-Train2`	$h^{64}(\cdot)$		93,3

anderes Mal ohne Vorgaben. Weil sich als Hintergrundmodell die Gauß-Verteilung als günstig erwiesen hat (vgl. Tabellen 7.12, 7.13, 7.14), wurde dieses Modell für das Training verwendet. Als Startprototyp, der für den iterativen Trainingsalgorithmus benötigt wird, wurde für jede Klasse das Objekt aus der ersten Bildszene der Trainingsdaten manuell segmentiert. Die Parameter des Hintergrundmodells wurden aus allen Trainingsdaten der jeweiligen Klasse berechnet.

Die Klassifikationen wurden mit allen Hintergrundmodellen durchgeführt. Um die Anforderungen zu variieren, wurde mit bekannten Rotationen oder Skalierungen getestet. Die Resultate sind in Tabelle 7.20 festgehalten.

Die extrem hohen Fehlerraten in diesen Experimenten verdeutlichen die Schwierigkeit der Lernaufgabe. Allerdings kann man erkennen, dass das automatische Objekttraining etwas über die Objekte gelernt hat, da die Fehlerraten kleiner als 95% ist. Diese Fehlerrate würde sich ergeben, wenn der Algorithmus lediglich aufgrund von Raten seine Entscheidung treffen würde, also keinerlei Bildinformation betrachtet.

7.3.4 Klassifikationsergebnisse auf `ERLANGEN`

Die `ERLANGEN`-Datenbank hat im Vergleich zur `COIL-I6`-Datenbank ein paar Besonderheiten, die die Klassifikation zum Teil erleichtern und zum Teil erschweren.

Eine Erleichterung ist, dass die Objekte nur in 2-dimensionalen Rotationen auftreten. Dies bedeutet, dass die Modellierung der Objekte einfacher ist. In den Trainingsdaten liegen die Objekte auf homogenem Hintergrund vor, was für das automatische Training von Vorteil ist. Der Hintergrund der Testdaten ist nachträglich zu den Objekten hinzugefügt worden und kann separat trainiert werden. Hinzu kommt noch, dass es sich immer um das gleiche Hintergrundbild handelt.

Erschwerend kommt hinzu, dass sich die Beleuchtungsverhältnisse der Objekte ändern. Daher reicht es nicht aus, als Objektmodell eine einfache Gauß-Verteilung zu verwenden. Es muss auf Gauß'sche-Mischverteilungen zurückgegriffen werden, die aufwendiger zu trainieren sind. Ein weiterer Punkt ist, dass die Objekte keine quadratische Form besitzen und es ungünstig ist, als Prototyp nur quadratische Ausschnitte zu berücksichtigen. Da in den Trainingsdaten lediglich homogener Hintergrund vorhanden ist, werden sich die Hintergrundpixel, die im quadratischen Objektprototyp liegen, nicht herausmitteln. Diese Pixel wirken sehr störend, wenn in der Klassifikation die Objekte auf realem Hintergrund liegen.

In Tabelle 7.21 sind die Klassifikationsergebnisse zu sehen, die mit den Verfahren in dieser Arbeit erzielt wurden.

Tabelle 7.21: Klassifikationsergebnisse auf der `ERLANGEN`-Datenbank.

	einheitliche Beleuchtung [%]	**verschiedene Beleuchtungsquellen [%]**
Erlangen [19]	0,0	0,0
manuell erstellte Gauß-Verteilung	0,0	23.5

7.3.5 Klassifikationsergebnisse auf `IRMA`

Auf der `IRMA`-Datenbank ist eine andere Methode gewählt worden, um die Verfahren aus dieser Arbeit zu testen. Da die Datenbank klein ist und keine Testdaten existieren, wird ein *Kernel-Density*-Ansatz mit *leaving-one-out* verwendet. Dies bedeutet, dass der Korpus gleichzeitig Trainings- und Testmenge ist. Alle Bilder in diesem Korpus sind einmal Beobachtung und werden klassifiziert. Wird ein Bild klassifiziert, wird es nicht als Trainingsdatum verwendet.

Am Lehrstuhl für Informatik VI der RWTH-Aachen wurde Software entwickelt [14], die nach diesem Ansatz arbeitet. Ein weiteres Merkmal dieser Software ist u.a. die Integration der Tangenten-Distanz und des *Image-Distortion-Models* in den Klassifikator.

Tabelle 7.22: Klassifikationsergebnisse auf der IRMA-Datenbank.

Ansatz	Daten	Fehlerrate [%]
KD mit TD und IDM	original IRMA-Daten	7,7
KD mit TD und IDM	trainiert auf IRMA-Daten	7,7

Eine Einschränkung für die Verwendung dieser Software war bisher die Skalierung aller Bilder auf eine einheitliche Größe. Mit Hilfe der Verfahren aus dieser Arbeit ist es möglich, diese Einschränkung zu umgehen. Durch das automatische Objekttraining ist man in der Lage, im Vorfeld den für das Objekt relevanten Ausschnitt aus den Bildern zu extrahieren. Diese Ausschnitte werden auf eine einheitliche Größe skaliert und dem Klassifikator als Eingabe gegeben.

Genau betrachtet wird bei diesem Verfahren die Trennung von Trainings- und Testdaten verletzt, da das Training auf allen Daten beruht. Das Ergebnis kann jedoch als Indikator dafür gelten, dass das Training wirklich die für die Klassifikation relevanten Bereiche erkennt.

In Tabelle 7.22 ist in der ersten Zeile das Klassifikationsergebnis zu sehen, welches die Software berechnet, wenn auf den originalen Daten klassifiziert wird. In der zweiten Zeile sind die Ergebnisse zu sehen, die man erhält, wenn nur die im Training als objektspezifisch erkannten Daten verwendet werden.

7.4 Analyse der Ergebnisse

Um qualitative Aussagen über den Erfolg der trainierten Prototypen zu machen, müssen die Klassifikationsergebnisse herangezogen werden. Diese jedoch machen deutlich, wie schwierig das Problem ist, wenn automatisch trainierte Prototypen in der Klassifikation eingesetzt werden sollen. Auf homogenem Hintergrund scheint dies im Vergleich zu realem Hintergrund eine leichte Aufgabe zu sein. Auf den ersten Blick sehen die Ergebnisse aus Abschnitt 7.2.1 zufriedenstellend aus. Werden diese Prototypen jedoch für die Klassifikation eingesetzt, dann stellt man fest, dass optimale Fehlerraten noch nicht erreicht werden (vgl. 7.16).

Werden Prototypen für Objekte trainiert, die auf realem Hintergrund positioniert sind, stellt sich das Training noch komplizierter dar. Um sich systematisch an die schwierigsten Aufgaben heranzutasten, wurde die Aufgabenstellung in den ersten Versuchen erleichtert. Es wurde vorausgesetzt, dass das Objekt in einem 3D-Rotationswinkel vorliegt. Dadurch konnte ein Objektmodell verwendet werden, das leicht trainiert werden kann, aber für die Objektbeschreibung völlig ausreicht. Unter diesen Bedingungen wurden visuell gute Prototypen trainiert. Danach wurden die Schwierigkeitsstufen angehoben. Es wurden jetzt mehrere Facetten des Objektes zugelassen. Für das Objektmodell bedeutete dies, dass vier 3D-Rotationswinkel trainiert werden mussten. Es konnte nun nicht mehr mit einfachen Modellen gearbeitet werden. Es wurden Gauß'sche Mischverteilungen verwendet, die mittels EM-Algorithmus trainiert wurden. Die Schwierigkeit dieser Aufgabe ist den Ergebnissen anzusehen (vgl. 7.2.2). Um zu überprüfen, ob die trainierten Prototypen für die Klassifikation einsetzbar sind, wurden die Objekte schließlich auf der gesamten Datenbank trainiert. Die Ergebnisse in Tabelle 7.20 verdeutlichen nochmals die Schwierigkeit der Lernaufgabe. Ein wenig wurde über die Objekte gelernt, was allerdings noch nicht ausreicht.

Wo aber liegen die Probleme und wie könnten die Fehlerraten verbessert werden?

Ein großes Problem ist die Einschränkung des Suchraumes. Damit der Testkorpus in akzeptabler Zeit klassifiziert werden konnte, musste der Suchraum klein gehalten werden. Durch eine Vorsondierung in einer niedrigeren Auflösungsstufe ist es möglich, ca. 30 Skalierungsstufen und 60

Rotationswinkel in annehmbarer Zeit zu untersuchen. Dennoch können grobe Fehler, die während dieser Sondierung gemacht werden, nacher in der hohen Auflösung nicht mehr rückgängig gemacht werden, da dort nicht mehr die Möglichkeit besteht, ausreichend viele Rotationen und Skalierungen zu betrachten.

Ein weiteres Problem ist die Objektbeschreibung. In den Ansätzen in dieser Arbeit wird für ein Objekt lediglich vorausgesetzt, dass es innerhalb eines Quadrates liegt. Da Objekte nicht immer quadratisch sind, werden neben dem eigentlichen Objekt noch Pixel hinzugefügt, die zum Hintergrund gehören. Diese Hintergrundpixel können die Lokalisation des Objektes negativ beeinflussen. In den vorgestellten Modellen ist nicht die Möglichkeit enthalten, diese störende Hintergrundinformation herauszurechnen. Ein einfacher Ansatz, der diese Störungen nicht vollständig beheben, aber verbessern kann, ist die Voraussetzung, dass das Objekt innerhalb eines Rechtecks liegt. Dadurch lässt sich das Objekt etwas genauer beschreiben, aber die Problematik wird dadurch nicht völlig behoben. Eine weitere Möglichkeit ist die Aufhebung der Voraussetzung der „gepoolten“ Varianz. Durch die Verwendung einer Kovarianzmatrix ließen sich die störenden Hintergrundpixel direkt in die Verteilungsfunktion integrieren, indem z.B. an den Hintergrundstellen höhere Varianzen erlaubt würden. Die Einführung der Kovarianzmatrix hat jedoch zwei Nachteile. Zum einen müssen ausreichend Trainingsdaten vorhanden sein, damit die Kovarianzmatrix vernünftig geschätzt werden kann. Zum anderen muss man sich über neue Optimierungsstrategien Gedanken machen, da die Fourier-Transformation nicht mehr in gleicher Weise benutzt werden kann.

Hinzu kommt noch, dass ein erscheinungsbasierter Ansatz alleine nicht ausreicht, um die Objekte zu charakterisieren. Eine Merkmalsanalyse im Vorfeld könnte zusätzlich die Leistung der Lokalisation erhöhen. Zum Beispiel könnten als Merkmale die diskrete Ableitung, Texturmerkmale, Wavelets oder Ähnliches mit in die Entscheidung einfließen.

Kapitel 8

Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurde eine Bilddatenbank erstellt, die Trainings- und Testkorpora enthält, mit denen automatisches Objekttraining und Klassifikation in verschiedenen Schwierigkeitsstufen durchgeführt werden kann. Das besondere an dieser Datenbank ist, dass die Objekte nicht in segmentierter Form und unter anderem auf realem Hintergrund positioniert sind.

Es wurde ein Basissystem entwickelt und implementiert, mit dem automatisch Prototypen von Objekten trainiert werden, die an beliebiger Position in einer Bildszene auftreten können. Für die Modellierung einer Bildszene wurde ein ganzheitlicher Ansatz gewählt. Das bedeutet, dass alle Pixel der Szene durch das Modell erklärbar sind und in Objekt- und Hintergrundpixel aufgeteilt werden. Es wurden verschiedene statistische, erscheinungsbasierte Szenenmodelle vorgestellt, um ein Objekt und den Hintergrund zu beschreiben.

Dazu wurde ein Klassifikator entwickelt, der auf diesem ganzheitlichen Ansatz basiert und die automatisch trainierten Prototypen verwendet, um in komplexen Bildszenen ein Objekt zu lokalisieren und zu klassifizieren.

Dieses Basissystem wurde mit den verschiedenen Szenenmodellen und den vorhandenen Datenbanken getestet, und es lassen sich folgende Schlüsse aus den Ergebnissen ziehen. Die Szenenanalyse mit den vorgestellten Modellen ist ein komplexes Problem. Sie liefert jedoch dem Stand der Technik angemessene Ergebnisse. Auf der `COIL-20`-Datenbank und der `ERLANGEN`-Datenbank wurden jeweils Fehlerraten von 0% erzielt.

Schwieriger wird die Problematik noch beim automatischen Objekttraining, das eingehend untersucht wurde. Die ersten Ergebnisse sind vielversprechend und Vorschläge für weiterführende Tests werden noch gemacht werden.

Die Forschungsgebiete der Klassifikation mit Hintergrundmodell und dem automatischen Objekttraining sind junge Forschungsbereiche. Die im Rahmen dieser Arbeit vorgestellten Verfahren sind als Einstieg in diese Bereiche zu sehen und lassen sich weiter fortsetzen.

Im Folgenden sind einige Anregungen aufgeführt, die aus zeitlichen Gründen in dieser Arbeit nicht mehr untersucht werden konnten.

Ein großes Problem, das aber aus der zeitlichen Begrenzung dieser Arbeit folgte, ist die starke Einschränkung des Suchraumes. Damit Training und Klassifikation in akzeptabler Zeit durchgerechnet werden konnten, wurde die Anzahl der zu betrachtenden Abbildungen stark reduziert und die Auflösungsstufe in der Vorauswahl auf 10% der originalen Bildgröße heruntergesetzt. Dadurch konnte zwar nicht mehr gewährleistet werden, dass das Optimum gefunden wurde, aber der Zeitaufwand war akzeptabel. Eine Möglichkeit, die Leistung des Trainings und der Klassifikation zu verbessern, wäre ein weniger restriktiv begrenzter Suchraum.

Ein anderes Problem sind die Hintergrundpixel in den Prototypen. Die jetzige Modellierung reicht wegen der „gepoolten“ Varianz nicht aus, um diese Situation zu beschreiben. Mit der Verwendung einer Kovarianzmatrix könnten diese Effekte durch das Modell abgefangen werden. Verwendet man jedoch eine Kovarianzmatrix, hat man das Problem der vernünftigen Schätzung bei zu geringen Datenmengen. Hinzu kommt, dass man die effiziente Berechnung der euklidischen Distanz mittels FFT auf die Mahalanobis-Distanz übertragen muss.

Ein anderer Ansatzpunkt ist die Wahl der Startparameter. In dieser Arbeit wurden einfache heuristische Methoden verwendet, indem die Startparameter von Hand gesetzt wurden. Es ist vorstellbar, für die Auswahl automatische Verfahren zu verwenden, die z.B. „wichtige“ Punkte in den Bildern finden und daraus eine Inititialisierung vornehmen.

Dann ist es weiterhin möglich, andere Distanzmaße zu testen. In diesem Zusammenhang ist z.B. die Tangenten-Distanz zu nennen, durch die auf verschiedenen Datenbanken hervorragende Ergebnisse zustande gekommen sind. Durch die lineare Struktur der TD ist es möglich, diese effizient mittels FFT berechnen zu lassen.

Die Klassifikation in dieser Arbeit beschränkt sich auf Einzelobjekterkennung. Eine Erweiterung des Klassifikators wäre die Multiobjekterkennung. Dabei stößt man auf neue Probleme, die modelliert werden müssen. Es steht noch offen, wie die Überlagerung oder die Superposition von Objekten zu modellieren ist.

Anhang A

EM-Algorithmus

Allgemein dient der EM-Algorithmus *Expectation Maximization* dazu, die optimalen Parameter der Likelihood-Funktion zu bestimmen. Darüber hinaus ist der EM-Algorithmus in der Lage, Parameter, die nicht direkt aus den Trainingsdaten beobachtbar sind, effizient zu schätzen. Im Fall der GMV (siehe Abschnitt 4.2.1) ist die Zuordnung eines Trainingsdatums zu einer Dichte nicht direkt aus den Daten ersichtlich. Damit diese Zuordnung bei der Schätzung mitberücksichtigt wird, kann man den EM-Algorithmus verwenden.

Als Basis für den EM-Algorithmus dient die *Likelihood*-Parameterschätzung. Es sei $p(\mathbf{x}|\lambda)$ das zu schätzende Modell mit der Parametermenge λ. Die Vektoren $\mathbf{x}_1, \ldots, \mathbf{x}_N$ mit $\mathbf{x}_i \in \mathbb{R}^D$ seien Realisierungen bzw. Trainingsdaten der zu schätzenden Verteilung. Dann ist die *Log-Likelihood-Funktion* definiert als:

$$L(\lambda) := L(\lambda; \mathbf{x}_1, \ldots, \mathbf{x}_N) \quad = \quad \log \prod_{n=1}^{N} p(\mathbf{x}_n|\lambda) \tag{A.1}$$

Die Parameter $\hat{\lambda}$, die diese Funktion maximieren, sind *die* Parameter, die die Trainingsdaten mit dem Modell $p(\mathbf{x}|\hat{\lambda})$ am besten beschreiben. Das Maximum

$$\hat{\lambda} \quad = \quad \underset{\lambda}{\text{argmax}}\Big\{\log \prod_{n=1}^{N} p(\mathbf{x}_n|\lambda)\Big\} \tag{A.2}$$

wird auch als *Maximum-Likelihood-Kriterium* (ML-Kriterium) bezeichnet.

Im besonderen Fall, dass die Parameter einer GMV, wie sie in Formel 4.3 gegeben ist, geschätzt werden sollen, ist folgendes Maximierungsproblem zu lösen. Da kein klassenübergreifendes Training gemacht wird, wird der Klassenzugehörigkeitsindex k analog zur Formel 4.3 in den folgenden Berechnungen weggelassen. Der EM-Algorithmus wird für jede Klasse einzeln durchgeführt. Es sei $\lambda := (\{c_i\}, \{\boldsymbol{\mu}_i\}, \sigma^2)$ die Menge der zu schätzenden Parameter. Der beste Schätzer $\hat{\lambda}$ ergibt sich aus:

$$\hat{\lambda} \quad = \quad \underset{\lambda}{\text{argmax}}\Big\{\log \prod_{n=1}^{N} \sum_{i=1}^{I} c_i \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_i, \sigma^2\mathbf{I})\Big\} \quad \text{und} \quad \sum_{i=1}^{I} c_i = 1 \tag{A.3}$$

Eine Lösung dieses unhandlichen Optimierungsproblems lässt sich effizient mit dem EM-Algorithmus berechnen. Zu bemerken ist, dass es sich bei dieser Lösung um ein lokales Optimum

handelt. Ausgangspunkt ist die Differenz zweier *Log-Likelihood*-Funktionen (vgl. [16, .96ff]). Es sei λ ein gegebener Parametersatz, dann wird $\bar{\lambda}$ neu geschätzt, so dass $L(\bar{\lambda}) \geq L(\lambda)$ gilt (d.h. die neuen Parameter beschreiben die Verteilung besser als die alten). Mit $\lambda := \bar{\lambda}$ wird der neue Parametersatz als der gegebene Parametersatz verwendet, und $\bar{\lambda}$ wird wiederum neu geschätzt. Dieser iterative Vorgang wiederholt sich solange, bis kein besserer Parametersatz gefunden wird. Werden diese iterativen Berechnungen direkt mit den *Log-Likelihood*-Funktionen durchgeführt, bleibt das unhandliche Optimierungsproblem bestehen. In [16, S.96ff] wird gezeigt, dass sich dieses Problem durch folgende Ungleichungen auf ein einfacheres Problem zurückführen lässt. Es gilt:

$$L(\bar{\lambda}) - L(\lambda) \;\geq\; Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) \;\geq 0 \tag{A.4}$$

Und $Q(\lambda, \bar{\lambda})$ ist in diesem Fall definiert als:

$$Q(\lambda, \bar{\lambda}) \;=\; \sum_{n=1}^{N} \sum_{i=1}^{I} p(i|\mathbf{x}_n, \lambda) \cdot \log\big(\bar{c}_i \cdot p(\mathbf{x}_n|i, \bar{\boldsymbol{\mu}}_i, \bar{\sigma}^2)\big) \tag{A.5}$$

Anstatt direkt mit den *Log-Likelihood*-Funktionen zu rechnen, verwendet man die Funktion $Q(\lambda, \bar{\lambda})$ und berechnet iterativ einen immer besseren Parametersatz unter Beachtung der Bedingung $Q(\lambda, \bar{\lambda}) > Q(\lambda, \lambda)$. In der Praxis hat sich gezeigt, dass die Funktion $Q(\lambda, \bar{\lambda})$ wesentlich handlicher ist, und sich ihr Maximum im Fall der GMV explizit berechnen lässt. Der Kern des EM-Algorithmus besteht im wesentlichen aus 3 Schritten. Es sei λ ein bekannter Parametersatz:

1. Bildung des Erwartungswertes über i: (*Expectation*)
   berechne $Q(\lambda, \bar{\lambda})$
2. Maximierung über $\bar{\lambda}$: (*Maximization*)
   $\underset{\bar{\lambda}}{\text{argmax}}\big\{Q(\lambda, \bar{\lambda})\big\}$
3. setze $\lambda := \bar{\lambda}$ und gehe zu Schritt 1

Für eine GMV lassen sich die neuen, optimalen Parameter $\bar{\lambda}$ explizit berechnen, indem die Ableitung von $Q(\lambda, \bar{\lambda})$ unter Berücksichtigung der Nebenbedingung $\sum_{i=1}^{I} c_i = 1$ gleich 0 gesetzt wird. Weil $Q(\lambda, \bar{\lambda})$ eine konvexe Funktion ist [7], ist dieses Maximum eindeutig bestimmt.

Daraus lassen sich die Reestimationsformeln der neuen Parameter wie folgt herleiten.

Für die Wichtungsfaktoren c_i der einzelnen Dichten muss die Nebenbedingung $\sum_{i=1}^{N} c_i = 1$ gelten. Es liegt ein Maximierungsproblem mit Nebenbedingung vor, für dessen Lösung ein *Lagrange-Multiplikator* ρ verwendet wird. Für die partielle Ableitung nach ρ gilt:

$$\begin{aligned} &\frac{\partial}{\partial \rho}\left(\sum_{n=1}^{N}\sum_{i=1}^{I} p(i|\mathbf{x}_n, \lambda) \log \bar{c}_i p(\mathbf{x}_n|i, \bar{\boldsymbol{\mu}}_i, \bar{\sigma}^2) - \rho(\sum_{i=1}^{I} \bar{c}_i - 1)\right) \;\stackrel{!}{=}\; 0 \\ \Leftrightarrow\; &\sum_{i=1}^{I} \bar{c}_i \;=\; 1 \end{aligned} \tag{A.6}$$

Für die partiellen Ableitungen nach den Wichtungsfaktoren c_i gilt:

$$
\begin{aligned}
& \frac{\partial}{\partial \bar{c}_i}\Big(\sum_{n=1}^{N}\sum_{i=1}^{I} p(i|\mathbf{x}_n,\lambda)\log \bar{c}_i p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2) - \rho(\sum_{i=1}^{I}\bar{c}_i - 1)\Big) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\frac{\partial}{\partial \bar{c}_i}\log \bar{c}_i p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2) - \rho \;=\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} \frac{p(i|\mathbf{x}_n,\lambda)}{\bar{c}_i} \;=\; \rho \\
\Leftrightarrow \quad & \bar{c}_i \;=\; \sum_{n=1}^{N}\frac{p(i|\mathbf{x}_n,\lambda)}{\rho} \qquad \text{(A.7)}
\end{aligned}
$$

Mit Bedingung A.6 gilt:

$$
\begin{aligned}
\sum_{i=1}^{I}\bar{c}_i \stackrel{(A.6)}{=} \sum_{i=1}^{I}\sum_{n=1}^{N}\frac{p(i|\mathbf{x}_n,\lambda)}{\rho} = 1 \\
\rho = \sum_{n=1}^{N}\sum_{i=1}^{I} p(i|\mathbf{x}_n,\lambda) \;=\; N \qquad \text{(A.8)}
\end{aligned}
$$

Das Ergebnis aus A.8 eingesetzt in A.7 ergibt:

$$
\bar{c}_i \;=\; \frac{1}{N}\sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda) \;=\; \frac{1}{N}\sum_{n=1}^{N}\frac{c_i \cdot p(\mathbf{x}_n|i,\boldsymbol{\mu}_i,\sigma^2)}{\sum_{i'=1}^{I} c_{i'} \cdot p(\mathbf{x}_n|i',\boldsymbol{\mu}_{i'},\sigma^2)} \qquad \text{(A.9)}
$$

Analog werden die Extrema der anderen Parameter berechnet. Im Fall der Mittelwertsvektoren der Dichten erhält man:

$$
\begin{aligned}
& \frac{\partial}{\partial \bar{\boldsymbol{\mu}}_i}\left(\sum_{n=1}^{N}\sum_{i=1}^{I} p(i|\mathbf{x}_n,\lambda)\cdot\log\big(\bar{c}_i \cdot p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2)\big)\right) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial \bar{\boldsymbol{\mu}}_i}\log\big(\bar{c}_i \cdot p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2)\big) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial \bar{\boldsymbol{\mu}}_i}\log\big(p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2)\big) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial \bar{\boldsymbol{\mu}}_i}\log\big(\mathcal{N}(\mathbf{x}_n|\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2)\big) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\|\mathbf{x}_n - \bar{\boldsymbol{\mu}}_i\| \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow \quad & \bar{\boldsymbol{\mu}}_i \;=\; \frac{\sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\mathbf{x}_n}{\sum_{n'=1}^{N} p(i|\mathbf{x}_{n'},\lambda)} \\
\Leftrightarrow \quad & \bar{\boldsymbol{\mu}}_i \;=\; \sum_{n=1}^{N}\gamma_i(n)\cdot\mathbf{x}_n \quad \text{mit} \quad \gamma_i(n) := \frac{p(i|\mathbf{x}_n,\lambda)}{\sum_{n'=1}^{N} p(i|\mathbf{x}_{n'},\lambda)} \qquad \text{(A.10)}
\end{aligned}
$$

Die $\gamma_i(n)$ können als Gewichte interpretiert werden, mit welchem Anteil die Beobachtung $\mathbf{x}_n$ in die Schätzung von $\bar{\boldsymbol{\mu}}_i$ eingeht.

Für die Schätzung der Varianz ergibt sich folgendes:

$$
\begin{aligned}
& \frac{\partial}{\partial \bar{\sigma}^2}\left(\sum_{n=1}^{N}\sum_{i=1}^{I} p(i|\mathbf{x}_n,\lambda)\cdot\log\big(\bar{c}_i\cdot p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2)\big)\right) \;\stackrel{!}{=}\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial\bar{\sigma}^2}\log p(\mathbf{x}_n|i,\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2) \;=\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial\bar{\sigma}^2}\log\mathcal{N}(\mathbf{x}_n|\bar{\boldsymbol{\mu}}_i,\bar{\sigma}^2\mathbf{I}) \;=\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\partial}{\partial\bar{\sigma}^2}\Big(-\frac{D}{2}\log(2\pi\bar{\sigma}^2)-\frac{1}{2\bar{\sigma}^2}\|\mathbf{x}_n-\bar{\boldsymbol{\mu}}_i\|^2\Big) \;=\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\Big(-\frac{D}{2\bar{\sigma}^2}+\frac{\|\mathbf{x}_n-\bar{\boldsymbol{\mu}}_i\|^2}{2\bar{\sigma}^4}\Big) \;=\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\cdot\frac{\|\mathbf{x}_n-\bar{\boldsymbol{\mu}}_i\|^2-D\bar{\sigma}^2}{2\bar{\sigma}^4} \;=\; 0 \\
\Leftrightarrow\quad & \sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda)\|\mathbf{x}_n-\bar{\boldsymbol{\mu}}_i\|^2 \;=\; D\bar{\sigma}^2\sum_{n=1}^{N} p(i|\mathbf{x}_n,\lambda) \\
\Leftrightarrow\quad & \bar{\sigma}^2 \;=\; \frac{1}{D}\sum_{n=1}^{N}\gamma_i(n)\|\mathbf{x}_n-\bar{\boldsymbol{\mu}}_i\|^2 \qquad \text{(A.11)}
\end{aligned}
$$

Insgesamt bilden die Ergebnisse A.9, A.10 und A.11 die Reestimationsformel für den EM-Algorithmus.

Anhang B

Datenbanken auf 'Computer Vision'

Auf der Suche nach geeigneten Test- und Trainingsdaten für das automatische Objekttraining und die Multiobjekterkennung wurden unter anderem die Bilddatenbankverweise auf der Homepage von *Computer Vision* genauer untersucht. Das Ergebnis dieser Analyse ist in der folgenden Tabelle aufgelistet. Bei der Untersuchung wurden folgende Kriterien zugrunde gelegt.

Zum einen sollte es in der Datenbank eine klare Trennung zwischen Trainings- und Testdaten geben, um Klassifikationsergebnisse bewerten und gegebenenfalls vergleichen zu können. Zum anderen sollten die Bilddaten eindeutig in Klassen mit eindeutig definierbaren Objekten eingeteilt sein.

Das Ergebnis der Analyse zeigt deutlich, dass die Forschung auf diesen Gebieten noch am Anfang steht. Für das automatische Objekttraining und die Multiobjekterkennung gibt es bis zu diesem Zeitpunkt noch keine geeigneten Trainings- und Testdatenbanken. Forschungsgruppen generieren für ihre Zwecke kleine Datenbanken, um ihre Verfahren zu überprüfen. Einen Vergleich der Verfahren untereinander gibt es bis zum jetzigen Zeitpunkt noch nicht.

Homepage	**Inhalt / Bemerkungen**
`xtreme.gsfc.nasa.gov`	atmosphärische Aufnahmen / keine Klassifikationsaufgabe
`www.cs.waikato.ac.nz/~singlis/ccitt.html`	Faxe / keine Klassifikationsaufgabe
`www.cs.cmu.edu/afs/cs.cmu.edu/project/cil/project/cil/ftp/html/cil-ster.html`	Stereodaten / Datenbank ungeeignet
`www.vasc.ri.cmu.edu/idb`	Gesichter / ungeeignet
`www.vision.caltech.edu`	Objekte vor realem Hintergrund / brauchbar, aber nicht erhältlich
`www.cs.columbia.edu/CAVE/curet`	Texturen / keine Klassifikationsaufgabe
`www.cs.sfu.ca/ colour`	Farbanalyse / keine Klassifikationsaufgabe
`www.cs.washington.edu/research/`	Bilder verschiedener, grober Kategorien

imagedatabase/groundtruth	keine Trennung zw. Test- und Trainingdaten
vision.psych.umn.edu/www/ kersten-lab/demos/digitalembryo.html	3D-Bilddaten / keine Klassifikationsaufgabe
earthrise.sdsc.edu	Erdaufnahmen von einem Sateliten / keine Klassifikationaufgabe
www.gastrointestinalatlas.com	medizinische Aufnahmen / Bildatlas
bias.csr.unibo.it/fvc2000	Fingerabrücke / ungeeignet
ftp.eedsp.gatech.edu	Datenbank nicht mehr vorhanden
www.TUGraz.at	Datenbank nicht mehr vorhanden
hlab.phys.rug.nl/archive.html	Landschaftsaufnahmen / ungeeignet
www.crs4.it/ gjb/ftpJOSA.html	Hyperspektralbilder / keine Klassifikationsaufgabe
www.ien.it/iengf/is/vislib.html	Bildersequenzen / keine Klassifikationsaufgabe
krakatoa.inria.fr/pub/IMAGES_ROBOTVIS	keine Zugriffsrechte für diese Seite
www.mic.atr.co.jp/ mlyons/jaffe.html	Gesichtsdatenbank / nicht geeignet
file:///home/michel ftp.vislist.com/IMAGERY/JISCT	Stereobilder / keine Klassifikationsaufgabe
www.ks.informatik.uni-kiel.de/∼/images.html	eine Szene unter verschiedenen Bedingungen / unbrauchbar
www.cs.cmu.edu/afs/cs.cmu.edu/ project/cil/ftp/html/v-images.html	kein Verbindungsaufbau möglich
ftp.c3.lanl.gov/pub/WSQ/print_data	Fingerabdrücke / ungeeignet
www-white.media.mit.edu/vismod/ imagery/VisionTexture/vistex.html	Texturen u. Hintergrundszenen / vorhanden
whitechapel.media.mit.edu/pub/images	viele Bilder / keine konkrete Aufgabenstellung
ftp.cse.psu.edu/pub/vision/MACHINE_VISION	Bilder von Texten / Verbindungsaufbau nicht erlaubt
marathon.csee.usf.edu/Mammography	digitale Mammographien / hier keine Klassifikationsaufgabe
ftp.cps.msu.edu/pub/prip	viele verschiedene Bilder / keine klare Aufgabenstellung
ltpwww.gsfc.nasa.gov/MODIS/MAS/gal_new.html	Flugzeuge, Erdaufnahmen / nur eine Bildergalerie
sequoyah.ncsl.nist.gov/pub/databases/data	Fingerabdrücke, handschriftliche Texte / nicht geeignet

ftp.cs.columbia.edu/jpeg/other/uuencoded	Fingerabdrücke / ungeeignet
www.nlm.nih.gov/research/ visible/visible_human.html	Personenaufnahmen / keine Klassifikationsaufgabe
eewww.eng.ohio-state.edu/ flynn/3DDB/Models	mehre Aufnahmen von 3D-Objekten / keine Klassifikationsaufgabe
eewww.eng.ohio-state.edu/ flynn/3DDB/RID	*Range*-Bilder / keine Klassifikationsaufgabe
sampl.eng.ohio-state.edu/∼sampl/database.html	Brodatz-Texturen / keine Klassifikationsaufgabe
www.cam-orl.co.uk/facedatabase.html	Gesichterdatenbank / nicht geeignet
ftp.limsi.fr/pub/quenot/opflow/testdata/piv	Ausschnitte von Bildersequenzen / keine Klassifikationsaufgabe
ww-dbv.cs.uni-bonn.de/stereo_data	Stereobilder / keine Klassifikationsaufgabe
rvl1.ecn.purdue.edu/∼aleix/aleix_face_DB.html	Gesichterdatenbank / ungeeignet
freebie.engin.umich.edu/pub/misc/textures	Texturen / keine Klassifikationsaufgabe

Anhang C

Verwendete Software

Die im Rahmen dieser Diplomarbeit erstellte Software wurde unter dem Betriebssystem *Linux* (`http://www.linux.org`) in der Programmiersprache *C++* implementiert. Als *Compiler* wurde der auf der Internetseite `http://www.gnu.org` frei verfügbare *GNU C++-Compiler* verwendet. Zusätzlich wurden folgende Bibliotheken benutzt, die die Implementierung erleichterten:

- `Blitz++` (`http://www.oonumerics.org/blitz`), eine frei verfügbare Bibliothek zur Verwaltung von mehrdimensionalen Vektoren.
- `Magick++` (`http://www.imagemagick.org`), eine frei verfügbare Bibliothek zum Einlesen und Bearbeiten von Bildern.
- `interpol` (`http://ioasun2.epfl.ch/thevenaz/interpolation`) stellt eine Funktion zur Verfügung, die eine Interpolation mittels B-Splines durchführt.
- `fftw` (`http://www.fftw.org`), Funktionen zur Durchführung der schnellen Fourier-Transformation für beliebige Bildgrößen (*„fastes Fourier Transformation in the West“*).

Dank Thomas Deselaers war es möglich, den EM-Algorithmus aus [12] in meinen Programmen zu verwenden. Er hat die vorhandene Software so angepasst, dass sie sich problemlos in meine Programme integrieren ließ.

Die Diplomarbeit wurde mit Hilfe von LaTeX (`http://www.latex-project.org`) geschrieben.

Anhang D

Erstellte Software

Der Kern der geschriebenen Programme ist die Klasse `MMMatching`. Diese Klasse berechnet aus einer gegebenen Beobachtung und einem Referenzmodell die wahrscheinlichste Projektion der Referenz auf die Beobachtung. Zur Unterstützung der `MMMatching`-Klasse wurden noch folgende Klassen implementiert, auf deren Methoden jedoch nicht näher eingegangen werden kann.

> `MMImage`: eine Klasse zur Verwaltung eines Bildes. Diese Klasse ermöglicht es, Bilder in allen Formaten, die auch von dem Programmpaket *ImageMagick* gelesen werden können, zu bearbeiten.
>
> `MMImageRecTemplate`: eine Klasse zur Verwaltung von rechteckigen Bildausschnitten (Templates), die auf einem Bild definiert werden können.
>
> `MMModels`: eine Klasse, die die Verwaltung von Gauß-Verteilungen und Histogrammen ermöglicht.
>
> `mm_tools`:eine Sammlung von Funktionen, die u.a. das Training von Modellen beinhaltet.

Aufbauend auf diesen Klassen wurde das Programm `trainObj` zum automatischen Objekttraining und das Programm `classifier` zur Klassifikation geschrieben.

D.1 `class MMMatching`

In der Tabelle D.1 sind die wichtigsten Methoden und Optionen dieser Klasse beschrieben. Diese Klasse ist die Realisierung der in Kapitel 4 beschriebenen Lokalisation. Integriert sind die in Kapitel 6 beschriebenen Optimierungsstrategien. `MMMatching` bildet die Grundlage für das automatische Objekttraining und die Klassifikation.

D.2 `trainObj`

Das Programm `trainObj` ist eine Realisisierung des in Kapitel 4 beschriebenen iterativen Algorithmus zum automatischen Trainieren von Objekten. Es lässt sich für das Training der Objektmodelle von einzelnen multivariaten Gauß-Verteilungen und Gauß'schen Mischverteilungen verwenden. Für den Hintergrund sind die 3 beschriebenen Modelle Gauß-Verteilung, Gleichverteilung und Histogramm implementiert. Zusätzlich zu diesen Modellen lässt sich die Methode des Trainings variieren. Normalerweise wird das *Likelihood*-Kriterium zum Training der Referenzmodelle verwendet (siehe Abschnitt 4.2.1). Es besteht jedoch auch die Möglichkeit, die

Verteilungen adaptiv zu trainieren. Das bedeutet, dass die Hintergrundverteilung nur aus dem gerade betrachteten Bild neu geschätzt wird.

In der Tabelle D.2 sind die Parameter des Programmes `trainObj` aufgelistet. Der Trainingskorpus wird aus einer Datei eingelesen, die wie folgt aufgebaut ist. In der 1. Zeile steht die Anzahl der Trainingsdaten, die eingelesen werden sollen. Für die weiteren Zeilen setzen sich die Einträge aus der Klassennummer und dem Dateinamen des Bildes zusammen. Zum Beispiel:

```
4
1 bild1.pgm
1 bild2.pgm
1 bild3.pgm
1 bild4.pgm
```

D.3 `classifier`

Das Programm `classifier` leistet die im Kapitel 3 beschriebenen Verfahren zur Einzelobjekterkennung mit Hintergrundmodell. In Tabelle D.3 sind die Parameter des Programmes aufgelistet. Die Datei, die den Testkorpus enthält, hat das gleiche Format wie die in Abschnitt D.2 beschriebene Datei für den Trainingskorpus. Die Datei, die die Modelle enthält, ist wie folgt aufgebaut. In der 1. Zeile steht die Anzahl der Objekte, für die Modelle eingelesen werden sollen. In den weiteren Zeilen wird jeweils ein Trippel angegeben, das zuerst die Klassennummer enthält, gefolgt vom Dateinamen des Objektmodells und dem Namen des Hintergrundmodells. Zum Beispiel:

```
3
1 fgModel1.gmv bgModel1.gauss
2 fgModel2.gmv bgModel2.gauss
3 fgModel3.gmv bgModel3.gauss
```

Für das Hintergrundmodell kann jeweils eine Gauß-Verteilung als auch ein Histogramm angegeben werden.

Tabelle D.1: Liste der wichtigsten Methoden der Klasse `MMMatching`.

Methode	**Beschreibung**
`AdaptBestSet2-Observation`	Übertragung der gefundenen Hypothesen auf ein anderes Bild. Mit der Funktion lassen sich z.B. Hypothesen, die in einer geringeren Auflösung gefunden wurden, auf eine höhere Auflösungstufe abbilden.
`Calculate`	Berechnet die beste Hypothese bei gegebener Referenz und Beobachtung.
`GetBestDist`	Man erhält die Distanz der besten Hypothese zurück.
`GetBestHyp`	Man erhält die beste Hypothese zurück.
`PostIteration`	Mit dieser Methode lässt sich auf den gefundenen, besten Hypothesen eine genauere Analyse durchführen. Es werden der Reihe nach die Hypothesen durchlaufen. Aus jeder Hypothese werden neue Hypothesen generiert, die in unmittelbarer Nachbarschaft liegen. Dadurch ist die Möglichkeit gegeben, für wenige Hypothesen eine wesentlich aufwändigere Distanzberechnung durchzuführen, wie z.B. die Tangentendistanz.
`SetBgType`	Legt den Type des Hintergrundmodells fest. Zur Zeit sind 2 Arten implementiert. Zum einen sind Histogramme möglich, zum anderen eine Gauß'sch Verteilung. Die Gleichverteilung lässt sich als Histogramm mit einem Intervall modellierten.
`SetBgModel`	Schnittstelle zur Übergabe des Hintergrundmodells an die Klasse.
`SetEuklid`	Die Suche wird mit der „naiven“ Berechnung der euklidischen Distanzen durchgeführt. Diese Einstellung kann dann verwendet werden, wenn die Dimension des Bildes kleiner als 400 Pixel ist.
`SetFFTEuklid`	Die Berechnung der Distanzen finden mit der FFT statt. Ab einer Bildgröße von mehr als 400 Pixeln bringt dies einen enormen Geschwindigkeitsvorteil gegenüber der „naiven“ Berechnung.
`SetMaxHyp`	Legt die maximale Anzahl von Hypothesen fest, die gespeichert werden sollen. Diese Hypothesen können dann z.B. mit der Methode `PostIteration` genauer untersucht werden.
`SetMaxObjSize`	Legt die maximale Größe eines Objektes fest.
`SetMinObjSize`	Legt die minimale Größe eines Objektes fest.
`SetNoRotations`	Legt die Anzahl der Rotationen fest, die betrachtet werden sollen. Die zu untersuchenden Rotationswinkel liegen in äquidistanten Abständen voneinander entfernt, so dass insgesamt die gewünschte Anzahl von Rotationen betrachtet werden.
`SetRotations`	Mit dieser Funktion lässt sich ein Vektor von Winkeln angeben, die betrachtet werden sollen.
`SetScaleLevels`	Setzt die Anzahl von Skalierungsebenen, die untersucht werden sollen. Mit dieser Methode lässt sich der Suchraum einschränken. Als Parameter wird die Anzahl der Skalierungsstufen eingestellt, die untersucht werden sollen. Die Skalierungsstufen werden nicht linear durchlaufen, sonder nach folgender Formel: $\left(\frac{max}{min}\right)^{\frac{l}{L}} \cdot min$. In der Formel ist max die größte und min die kleinste zu untersuchende Skalierungsstufe. Die Variable L gibt die Anzahl der zu untersuchenden Skalierungsstufen an. Werden mehr Skalierungsstufen angegeben als möglich sind, erreicht man eine lineare Durchsuchung der Skalierungsstufen. Es wird keine Stufe mehrfach betrachtet.
`SetVerbose`	Nach dem Aufruf dieser Methode werden zusätzliche Informationen über den Stand der Suche auf den Bildschirm ausgegeben.

Tabelle D.2: Liste der Optionen des Programms `trainObj`.

Option	**Beschreibung**
`--trainCorpus <file>`	der zu verwendende Trainingskorpus
`--startBgModel <gauss>`	Gauß-Verteilung für das Hintergrundmodell, mit der das Training beginnen soll.
`--startFgModel <gmv>`	Gauß'sche Mischverteilung für das Vordergrundmodell, mit der das Training beginnen soll.
`--bgHistogram <hist>`	Histogramm für das Hintergrundmodell, mit dem das Training beginnen soll.
`--smoothHistogram`	Glättung der Histogramme nach dem Training.
`--scaleLevels <n>`	Anzahl der Skalierungsstufen, die berücksicht werden sollen.
`--notTrainBgModel`	Hintergrundmodell wird nicht trainiert. Es werden die Startparameter verwendet.
`--noRotations <n>`	Anzahl der Rotationen, die berücksichtigt werden sollen.
`--maxHyp <n>`	Anzahl von Hypothesen, die gespeichert werden sollen.
`--minObjSize <n>`	minimale Objektgröße
`--maxObjSize <n>`	maximale Objektgröße
`--maxDensities <n>`	maximale Anzahl der Dichten der Gauß'schen Mischverteilung
`--minObservations <n>`	minimale Anzahl von Beobachtungen, die einer Dichte zugeordnet werden müssen
`--FFTEuklid`	Berechnung der euklidischen Distanzen mittels FFT
`--Euklid`	„naive“ Berechnung der euklidischen Distanzen

Tabelle D.3: Liste der Optionen des Programms `classifier`.

Option	**Beschreibung**
`--testCorpus <file>`	der zu verwendende Testkorpus
`--model <file>`	`<file>` enthält die zu verwendenden Modelle
`--useHistogram`	verwendete Histogramme als Hintergrundmodell
`--adaptiveBgModel`	adaptive Berechnung des Hintergrundmodells
`--maxHyp <n>`	maximale Anzahl von Hypothesen, die gespeichert werden sollen
`--minObjSize <n>`	minimale Objektgröße
`--maxObjSize <n>`	maximale Objektgröße
`scaleLevels <n>`	Anzahl der Skalierungsstufen, die berücksicht werden sollen.
`--noRotations <n>`	Anzahl der Rotationen, die berücksichtigt werden sollen
`--FFTEuklid`	Berechnung der euklidischen Distanzen mittels FFT
`--Euklid`	„naive“ Berechnung der euklidischen Distanzen

Literaturverzeichnis

[1] Computer Vision Homepage.
URL: `http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html`.

[2] J. Corridoni, A. Del Bimbo und P. Pala. Image Retrieval by Color Semantics. *Multimedia Systems*, Band 7(3):175–183, Mai 1999.

[3] G. S. Cox. Template Matching and Measures of Match in Image Processing. Technischer Bericht, Department of Electrical Engineering, University of Cape Town, South Africa, 1995.

[4] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas und P. N. Yianilos. The Bayesian Image Retrieval System, PicHunter: Theory, Implementation and Psychophysical Experiments. *IEEE Trans. Image Processing*, Band 9(1):20–37, Januar 2000.

[5] J. Dahmen, T. Theiner, D. Keysers, M. Motter, H. Ney, T. Lehmann und B. Wein. Classification of Radiographs in the '*Image Retrieval in Medical Applications*'-System (IRMA). In *6th International RIAO Conference on Content-Based Multimedia Information Access*, Seiten 551–566. Paris, Frankreich, April 2000.

[6] J. Dahmen, D. Keysers und H. Ney. An Automatic Approach to Invariant Radiograph Classification. In H. Handels, A. Horsch, T. Lehmann und H.-P. Meinzer (Hg.), *Bildverarbeitung für die Medizin 2001*, Seiten 337–341. Springer-Verlag, Berlin, Lübeck, März 2001.

[7] A. Dempster, N. Laird und D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, Band 39:1–38, 1977.

[8] R. O. Duda, P. E. Hart und D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, 2. Auflage, 2001.

[9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele und P. Yanker. Query by image content: The QBIC system. *IEEE Computer*, Band 28(9):23–31, September 1995.

[10] G. Frederix, G. Caenen und E. Pauwels. PARISS: Panoramic, Adaptive and Reconfigurable Interface for Similarity Search. In *Proc. Int'l Conf. Image Processing*. Vancouver, Canada, September 2000.

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer Science and Scientific Computing Academic Press Inc., 2. Auflage, 1990.

[12] M. O. Güld. *Inhaltsbasierter Bildzugriff mittels Statistischer Objekterkennung*. Diplomarbeit, RWTH-Aachen, Lehrstuhl für Informatik VI, Juli 2000.

[13] A. Hiroike, Y. Musha, A. Sugimoto und Y. Mori. Visualization of Information Spaces to Retrieve and Browse Image Data. In D. P. Huijsmans und A. W. M. Smeulders (Hg.), *Proc. Visual '99: Information and Information Systems*, Band 1614 von *Lecture Notes in Computer Science*, Seiten 155–162. Springer, Amsterdam, Niederlande, Juni 1999.

[14] D. Keysers. *Approaches to Invariant Image Object Recognition*. Diplomarbeit, RWTH-Aachen, Lehrstuhl für Informatik VI, Juni 2000.

[15] H. Murase und S. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *International Journal of Computer Vision*, Band 14(1):5–24, Januar 1995.

[16] H. Ney. Mustererkennung und Neuronale Netze. Skript zur Vorlesung, RWTH Aachen, Lehrstuhl für Informatik VI, 1999.

[17] H. Ney. Digitale Signalverarbeitung für Sprache und Bilder. Skript zur Vorlesung, RWTH Aachen, Lehrstuhl für Informatik VI, 2000.

[18] J. Pösl. *Erscheinungsbasierte statistische Objekterkennung*. Shaker Verlag, Aachen, 1998.

[19] M. Reinhold, D. Paulus und H. Niemann. Appearance-Based Statistical Object Recognition by Heterogeneous Background and Occlusion. Band 2191 von *Lecture Notes in Computer Science*, Seiten 254–261. München, Deutschland, September 2001.

[20] D. J. Swets und J. Weng. Hierarchical Discriminant Analysis for Image Retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Band 21(5):386–401, Mai 1999.

[21] A. Smeulders, M. Worring, S. Santini, A. Gupta und R. Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Band 22(12), Dezember 2000.

[22] M. Weber, M. Welling und P. Perona. Unsupervised Learning of Models for Visual Object Class Recognition. 6th Annual Joint Symposium on Neural Computation, JNSC99, Pasadena, CA, May 1999.

[23] M. Weber, M. Welling und P. Perona. Towards Automatic Discovery of Object Categories. In *Proc. Computer Vision and Pattern Recognition*, Seiten 101–108. Hilton Island, SC, 2000.

[24] M. Weber, M. Welling und P. Perona. Unsupervised Learning of Models for Recognition. In *Proc. 6th European Conf. Computer Vision*. Dublin, Ireland, Juni 2000.