Farbkonstante Objektlokalisierung und -identifikation aus lokalchromatischen Orientierungshäufigkeiten

Diplomarbeit
am Lehrstuhl für Kognitive Systeme
des Instituts für Informatik und Praktische Mathematik
der Christian–Albrechts–Universität zu Kiel

vorgelegt von
Harro Pahl

Betreuer: Dipl. Ing. Udo Mahlmeister

Januar 1997

Zusammenfassung

Visuelle Suche ist neben visueller Navigation eine wichtige Teilaufgabe **visuell geführter Robotik**, die sich mit dem visuell basierten Verhalten von Robotern in wechselnden Umgebungen beschäftigt. Voraussetzung für das Handhaben von Objekten ist deren **Lokalisierung** und **Identifikation** aus den Signalen von Kameras. Beide Verfahren müssen robust gegenüber Umweltbedingungen bzw. Aufnahmebedingungen wie Beleuchtung oder Verdeckung sein. Vor allem aber müssen sie in Echtzeit arbeiten, um mit dem zeitlichen Rahmen der dynamischen Umgebung schritthalten zu können.
Eine weitere wichtige Aufgabe ist die inhaltsbasierte Suche von Bildeigenschaften in Bilddatenbanken. Die textuelle Bildbeschreibung und Suche ist oft nur unzureichend, da mit Schlüsselwörtern Bildinhalte nur eingeschränkt wiedergegeben werden können. Eine nicht-textuelle Beschreibung von Bildern durch Bildmerkmale ist nötig. Wir stellen in dieser Arbeit zwei Verfahren von Swain und Ballard[SB91] vor – einen Algorithmus zur Lokalisierung und einen Algorithmus zur Identifikation von Objekten –, die den obigen Anforderungen z.T. entsprechen. Beide Verfahren arbeiten auf der Basis von **Farbhistogrammen**, wodurch sich der Begriff 'color indexing' geprägt hat. Die Verfahren können einzeln für sich angewendet werden; mit einer Koppelung beider Algorithmen ist aber auch eine attentive Suche nach bestimmten Objekteigenschaften in einem Bild möglich. Ferner untersuchen wir verschiedene Weiterentwicklungen der Swain'schen Verfahren, welche bestimmte Schwächen ausräumen sollen.
Eine große Schwäche beider Algorithmen ist deren Anfälligkeit gegenüber örtlicher oder spektraler Beleuchtungsänderung. Diese Anfälligkeit macht einen Einsatz der Verfahren nur in Umgebungen möglich, in denen die Beleuchtungsbedingungen vollständig kontrolliert werden. Um diesen Nachteil aufzuheben, entwickeln wir einen Algorithmus zur **Farbkonstanz**, der den beiden Verfahren vorangestellt wird. Wir führen in der Vorverarbeitung eine nichtlineare Farbraumkonvertierung durch und berechnen mit einem Basissatz von steuerbaren Filtern lokale Orientierungen auf den einzelnen Farbkanälen. Verglichen werden in den anschließenden Verfahren somit Orientierungshäufigkeiten statt Farbhäufigkeiten.
In einer Vorstudie implementieren wir sämtliche Algorithmen unter dem Bildverarbeitungssystem *Khoros II*. Ferner nehmen wir eine **Echtzeitimplementierung** der Swain'schen Algorithmen und unseres Verfahrens zur Farbkonstanz auf einem Pipelinerechner MV200 der Firma Datacube vor. Die MV200 ist ein Spezialrechner für Aufgaben der Bildverarbeitung und bietet sich vor allem wegen seiner Faltungskapazitäten für die intensive lineare Filterung unseres Farbkonstanzalgorithmus' an. Wir erzielen Gesamtlaufzeiten von ca. 0.1 s, wodurch das Bild einer Farbkamera *live* auf das Vorhandensein verschiedener Objekte getestet werden kann.

Anhand einer Datenbank – bestehend aus 15 realen Objekten – liefern wir Testergebnisse zu allen implementierten Algorithmen und belegen die Farbkonstanz unseres Verfahrens auch unter schwierigen Beleuchtungssituationen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In diesem einleitenden Kapitel geben wir Motivationen für die Verfahren, welche wir in dieser Arbeit beschreiben und entwickeln. Als zentrale Themen resultieren daraus die Echtzeitfähigkeit und -implementierbarkeit von Algorithmen sowie die farbkonstante Vorverarbeitung von Farbbildern.

1.1 Motivation

1.1.1 Suche in Bilddatenbanken

Neben textueller Information wird zunehmend auch visuelle Information in Form von Bildern und Videos in Datenbanken verwaltet. Textuelle Daten können zum einen effektiv mit sog. Schlüsseln beschrieben werden, zum anderen besteht die Möglichkeit, Texte schlicht nach dem Vorkommen von Wörtern oder Wortkombinationen zu durchsuchen, um die Informationsvielfalt auf einen gewissen Bereich zu begrenzen. Die Suche nach Bildinformationen mit bestimmten Eigenschaften ist wesentlich schwieriger als im textuellen Fall. Eine textuelle Bildbeschreibung ist jeweils vom Verfasser abhängig, die Schwerpunkte der Beschreibung können je nach Autor stark divergieren. So sind z.B. die beiden folgenden Suchaufgaben in einer Bilddatenbank vorstellbar:

- Ein Autoliebhaber möchte aus einer Vielzahl von innerstädtischen Aufnahmen diejenigen Bilder ausgewählt haben, in denen Autos von ganz bestimmtem Typ und Farbe vorkommen.
- Ein automatischer Pförtner soll mittels einer Überwachungskamera Menschen aufgrund ihrer Gesichter identifizieren.

In dem ersten Fall sind die gesuchten Objekte nicht wesentlicher Teil der Bilder. Es ist zu bezweifeln, daß eine textuelle Beschreibung auch diese gewünschten nebensächlichen Informationen enthält. Eine Anpassung der Bildabfrage ist im textuellen Fall nicht möglich, da die gesuchten Informationen nicht vorhanden sind. Im zweiten Fall ist eine textuelle Beschreibung zwar möglich, jedoch kann diese nicht

'maschinell' vom Computer vorgenommen werden und ist für diese Aufgabe nicht brauchbar.
Somit werden für diesen Problembereich Eigenschaften von Bildern benötigt, die zum einen vom Computer berechnet werden können, und die zum anderen eine ausreichende Beschreibung und Abgrenzung zu anderen Bildern liefern.

1.1.2 Maschinelles Greifen eines Objektes

Die Automatisierung von Arbeitsabläufen beeinhaltet oftmals das Einsetzen oder Bewegen von Gegenständen durch Roboterarme. Das Greifen eines Objektes kann dabei 'blind' erfolgen, wenn Position und Art des Objektes allein durch den bisherigen Fertigungsablauf bekannt sind. Diese Informationen sind jedoch nicht immer vorhanden, so z.B. bei einem automatischen Büroreiniger, der mittels Kamerasensoren u.a. auf dem Boden liegende Abfälle einsammeln, heruntergefallene 'wertvolle' Gegenstände (Locher, Disketten ...) jedoch liegenlassen soll. Unbekannte Objekte müssen sowohl lokalisiert als auch identifiziert werden.
Wiederum ist die Suche in Bilddatenbanken notwendig, lediglich der Umfang der Datenbanken wird bei derartigen Aufgaben wesentlich geringer sein als der von allgemeinen Bilddatenbanken. Bei Greifaufgaben spielt jedoch die Zeit, in der Objekte lokalisiert oder identifizert werden, eine wichtige Rolle. Bei dem Durchsuchen einer Bilddatenbank aus dem letzten Abschnitt ist es nicht maßgebend, wann die Resultate vorliegen. Kurze Rechenzeiten sind zwar wünschenswert aber nicht ausschlaggebend. Anders sieht dies bei den Greifproblemen aus. Ein automatisierter Arbeitsvorgang sollte zum einen nicht um ein vielfaches langsamer sein als von Menschenhand ausgeführt, zum anderen kann eine lange Reaktionszeit bei bewegten Objekten zum Fehlschlagen des Greifvorganges führen.
Algorithmen für diese Probleme müssen somit geringe Komplexitäten haben, sie müssen echtzeitfähig sein. Der Begriff der 'Echtzeit' ist hierbei in der Literatur nicht fest umrissen. Je nach Aufgabenstellung wird eine Bearbeitung von 10 Bildern pro Sekunde gefordert, es werden aber auch Verfahren mit Laufzeiten von mehr als $8\,s$ echtzeitfähig genannt (siehe z.B. [SR88]).

1.1.3 Ein echtzeitfähiges Bildmerkmal: Farbe

Durch die Bildaufnahme von Farbkameras ergeben sich grundsätzlich drei Farbkanäle mit insgesamt 256^3 Farben statt wie bei Grauwertbildern nur ein Grauwertkanal mit 256 Graustufen. Swain und Ballard[SB91] zeigen in ihrem Artikel 'Color Indexing' aus dem Jahre 1991, daß Farbe ein Bildmerkmal ist, welches sich sehr gut zur Objektlokalisierung und -identifikation eignet. In früheren Verfahren zur Objektdetektion werden Farben nicht verwendet, es wird sogar bezweifelt, daß das Merkmal Farbe etwas zur Lösung dieser Probleme beitragen kann. So schreiben z.B. Lee et al[LPSE94]

> *"For some real world objects, such as a banana, surface characteristics such as color contribute to the object's classification. But they do not*

> *contribute to the classification of many other objects, such as a chair. Indeed, surface characteristics such as color play at most a secondary role in the real-time recognition of intact object when it edges can be extracted.*" (S. 66) ,

nachdem bereits zahlreiche Autoren Farbe als Merkmal zur Objektdetektion auch in größeren Datenbanken mit einem Umfang von bis zu 2000 Bildern erfolgreich angewendet haben (siehe z.B. [SO95]).

Swain und Ballard verwenden in ihren Verfahren 3-dimensionale (3D) Farbhistogramme. Die Farbhäufigkeiten eines Bildes können durch Abzählen der vorkommenden Farben ermittelt werden und stellen somit ein einfach zu extrahierendes Merkmal dar. Farbhäufigkeiten sind gleichzusetzen mit Flächeninhalten von farbigen Oberflächen. Da der Flächeninhalt eines Objektes in der Ebene eine Eigenschaft ist, die invariant gegenüber Rotation in der Ebene ist (im 3D-Fall gegenüber der Rotation um die optische Achse), sind Farbhistogramme rotationsinvariant. Ferner ändert sich das Farbhistogramm eines Objektes nur langsam unter wechselnder Entfernung vom Objekt zur Kamera oder bei der Verdeckung durch andere Gegenstände. Somit sind Verfahren, welche Farbhistogramme verwenden, robust gegen Änderungen in diesen Parametern.

Aufgrund dieser Eigenschaften eignen sich Farbhistogramme sehr gut für die Objektlokalisierung und -identifikation in Echtzeit. Das Merkmal Farbe unterscheidet sich stark von anderen in der Objektdetektion verwendeten Merkmalen, so z.B. in Verfahren, welche in einer aufwendigen Vorverarbeitung Kanten und Ecken extrahieren und diese zu Objektkonturen zu verbinden oder eine Spline-Interpolation vornehmen, oder in Verfahren, die geometrische Objekte wie Ellipsen in Objektkonturen einpassen, um aufgrund dieser Ellipsen Objekte zu unterscheiden. Gerade diese Art der Objektmodellierung ist besonders anfällig gegenüber dem Problem der Verdeckung.

Größter Nachteil der Farbhistogramme ist deren Anfälligkeit gegenüber örtlicher und spektraler Beleuchtungsänderung. Schon geringste Änderungen in diesen Parametern verschieben die Histogramme derart, daß Algorithmen, welche auf Farbhistogrammen arbeiten, sehr unzuverlässig werden. Ein Einsatz von Farbhistogrammen ist nur unter Beleuchtungsbedingungen möglich, die vollständig kontrolliert werden – sobald Tageslicht ins Spiel kommt, sind die Resultate der Verfahren nicht verwendbar. Aus diesem Problem erwächst die Notwendigkeit einer farbkonstanten Vorverarbeitung von Eingangsbildern, auf welche die 'Color Indexing'-Verfahren angewendet werden können. Wir entwickeln dazu in dieser Arbeit einen Farbkonstanz-Algorithmus, der auf einer nichtlinearen Farbraumkonvertierung und anschließender linearer Filterung mit einem Basissatz von steuerbaren Filtern basiert. Mit der Filterung berechnen wir lokale Orientierungen, durch die zum einen Farbkonstanz erzielt wird und zum anderen neben dem photogrammetrischen Merkmal 'Farbe' das geometrische Merkmal 'lokale Orientierung' zur besseren Objektcharakterisierung eingeführt wird.

Der Farbkonstanz-Algorithmus ist in Hinsicht auf eine Echtzeitimplementierung entworfen, die wir auf einem Pipelinerechner vornehmen. Auf die Resultate der Vorverarbeitung wenden wir die urspünglichen Verfahren fast unverändert an – die positiven Eigenschaften der Farbhistogramme (jetzt Orientierungshistogramme) bleiben bis auf die Rotationsinvarianz erhalten. Um diese Eigenschaft zurückzugewinnen, entwickeln wir ein Verfahren, das auf den Orientierungshistogrammen Schwerpunkte berechnet, so daß Objekte daraufhin in eine Normlage rotiert werden können.

1.2 Überlegungen zu Komplexitätsangaben

Zwei Möglichkeiten, die Komplexität eines Algorithmus' anzugeben, sind die Anzahl der benötigten Grundoperationen und die Komplexität O in Abhängigkeit von bestimmten Parametern. Im ersten Fall wird schlicht die Anzahl der Operationen, die ein Computer für das Abarbeiten des Algorithmus benötigt, gezählt. Dabei kann z.B. zwischen Addition und Multiplikation unterschieden werden, da eine Multiplikation grundsätzlich mehr Rechenzeit in Anspruch nimmt als eine Addition. Die Berechnung des Landauschen Symbols O ist eine Grenzwertbetrachtung. Einige Parameter können vernachlässigt werden, gesucht wird eine obere Schranke für die Anzahl der benötigten Grundoperationen. Ein Algorithmus A_1, der z.B. $6n$ Grundoperationen benötigt, hat die Komplexität $O(n)$; ein Algorithmus A_2 mit $1024n$ Operationen hat die gleiche Komplexität. Bei immer größer werdendem n nähern sich beide Kostenfunktionen dem gleichen Grenzwert. Trotzdem ist A_1 83 mal so schnell wie A_2.
Bei einem Algorithmus, der im wesentlichen die Faltung eines Bildes mit einer höchstens halb so großen Maske beinhaltet, stellt sich die Frage, wie sinnvoll eine Komplexitätsangabe mit dem Landauschen Symbol ist. Die Parameter des Algorithmus' sind die Abmessungen des Bildes und der Maske. Schon aus Gründen der Speicherknappheit (sowohl Arbeits- als auch Massenspeicher) kann das Bild nicht unendlich groß werden. Ein 512×512 Pixel großes Bild enthält eine ausreichende Anzahl von Informationen – eine Erhöhung der Auflösung würde zum derzeitigen Stand der Technik keine Vorteile bringen. Demzufolge kann auch die Faltungsmaske nicht unendlich groß werden, da der Algorithmus nur sinnvoll ist, wenn die Maske kleiner gleich der halben Größe des Bildes bleibt. Eine Grenzwertbetrachtung für diese Parameter halten wir daher nicht für sehr aussagekräftig.
In unserer Arbeit befassen wir uns u.a. mit unterschiedlichen Algorithmen, die z.T. in ihrer Komplexität verglichen werden. Wir bevorzugen dabei das Maß der Anzahl der benötigten Grundoperationen und werden das Landausche Symbol nur in Ausnahmefällen aufführen.

Diese Arbeit ist im weiteren folgendermaßen aufgeteilt: Kapitel 2 beschreibt die ursprünglichen 'Color Indexing'-Verfahren von Swain und Ballard. In dem darauffolgenden Kapitel befassen wir uns mit verschiedenen Weiterententwicklungen dieser Verfahren, die u.a. Farbkonstanz oder höhere Genauigkeit bei der Lokalisierung beinhalten. Da wir in unserem Algorithmus zur Farbkonstanz steuerbare Filter benutzen, gehen wir in Kapitel 4 allgemein auf steuerbare Filter und speziell auf die von uns verwendeten Andersson-Filter ein. In Kapitel 5 entwickeln wie schließlich den Algorithmus zur Farbkonstanz. Die Echtzeitimplementierung auf dem Pipelinerechner MV200 der Firma Datacube ist Inhalt von Kapitel 6. Wir gehen dabei nicht zu sehr ins Detail, sondern wollen vielmehr Ideen vermitteln, wie die verschiedenen Algorithmen auf dem Spezialrechner umgesetzt werden. In Kapitel 7 beschreiben wir die umfangreichen Experimente zu sämtlichen Verfahren, die in dieser Arbeit behandelt werden. Abschließend diskutieren wir die Resultate dieser Arbeit und geben einen Ausblick auf mögliche Weiterentwicklungen.

Kapitel 2

Die Verfahren von Swain und Ballard

In diesem Kapitel stellen wir zwei Verfahren von Swain und Ballard[SB91], die Histogramm-Rückprojektion und die Histogramm-Intersektion, vor, die unter dem Begriff 'Color-Indexing' zusammengefaßt werden. Die Rückprojektion ist ein Algorithmus zur Lokalisierung, die Intersektion zur Identifizierung eines Objektes aufgrund seiner Farbverteilung. Beide Verfahren arbeiten auf Farbhistogrammen und haben daher geringe Komplexitäten. Sie gliedern sich jeweils – wie in Abbildung 2.1 dargestellt – in zwei Phasen: eine Offline- und eine Online-Phase.
Die Offline-Phase beinhaltet die Aufnahme von bestimmten Objekten in digitaler Form mittels einer Kamera, die Berechnung von evtl. mehrdimensionalen Histogrammen dieser Objekte und das Abspeichern der Histogramme in einer Datenbank. Die Datenbank dient als Grundlage für beide Verfahren. Für Aufgaben in der 'Active Vision' ist die Rückprojektion auch ohne Online-Phase einsetzbar. Soll z.B. ein in einer Szene aufgetauchtes Objekt beobachtet werden, genügt die Berechnung des Histogrammes nur dieses Objektes zur weiteren Verfolgung, ohne daß eine Datenbank mit den Histogrammen anderer Objekte vorliegen muß.
In der Online-Phase wird ein Bild von einer unbekannten Szene oder einem Objekt digital aufgenommen, das Histogramm davon berechnet und – je nach Algorithmus – die Koordinaten bestimmt, an denen sich ein gesuchtes Objekt am wahrscheinlichsten befindet (Rückprojektion), oder das Histogramm mit den Histogrammen der Datenbank verglichen, um das ähnlichste zu ermitteln (Intersektion).
Als Ausgangsobjekte können sowohl 2D- als auch 3D-Objekte gewählt werden. Beide Verfahren sind grundsätzlich invariant gegenüber Rotation um die optische Achse der Kamera (im folgenden kurz 'rotationsinvariant'), da die histogrammisierten Merkmale Farbpixel rotationsinvariant sind. Ferner zeigt sich eine Robustheit gegen Änderung der Entfernung Objekt/Szene-Kamera und der Verdeckung von Objekten. Ein 2D-Objekt ist daher durch ein Histogramm ausreichend charakerisiert, bei einem 3D-Objekt werden evtl. Histogramme von verschiedenen Ansichten des Objektes benötigt.
Obwohl die Algorithmen für die Lokalisierung und Identifikation von farbigen Ob-

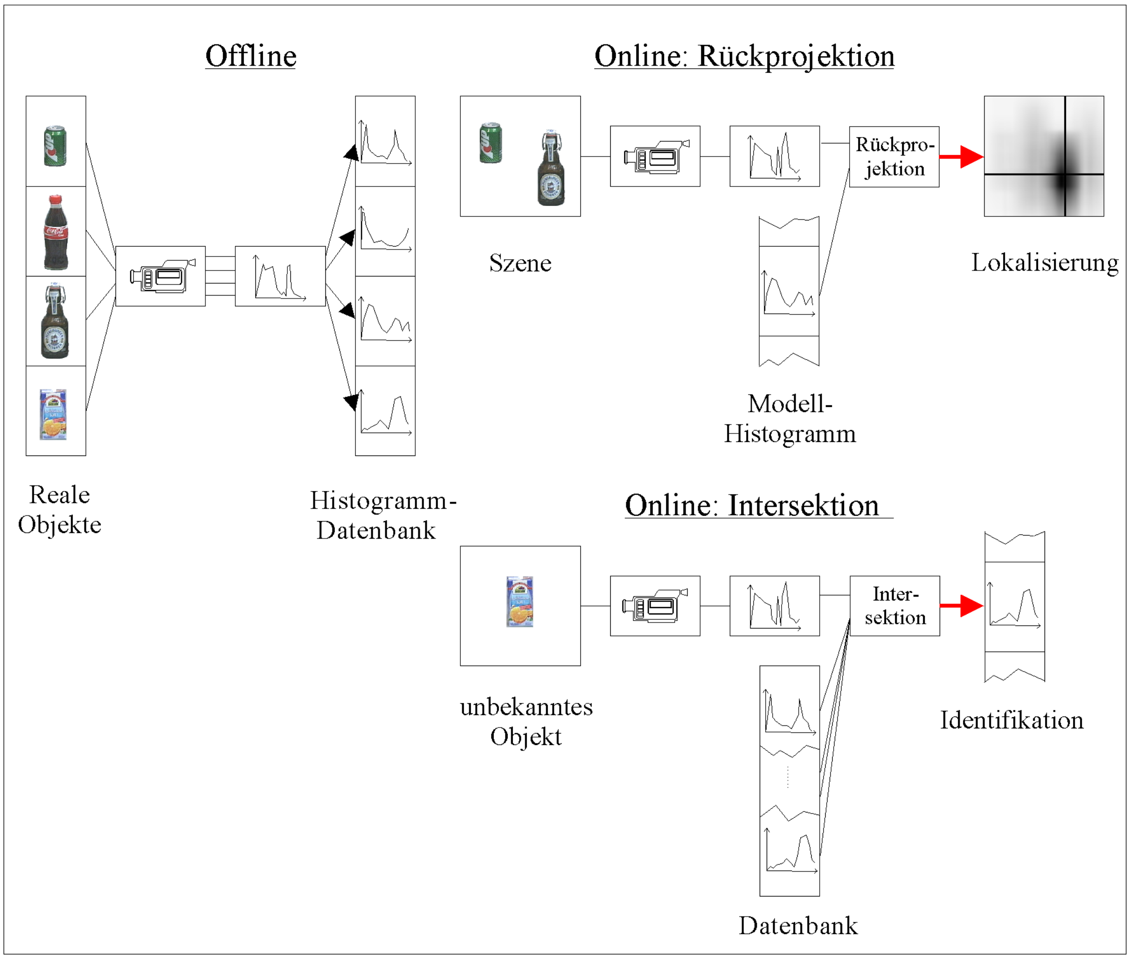


Abbildung 2.1: Offline- und Online-Phasen der Rückprojektion und der Intersektion. Die Rückprojektion ist in der 'Active Vision' auch ohne Online-Phase einsetzbar.

jekten entworfen sind, können sie auch auf Grauwertobjekte angewendet werden. Die Resultate sind allerdings nicht herausragend, da Grauwerthistogramme nicht charakeristisch genug sind: zu viele Farben werden bei der Digitalisierung dem gleichen Grauwert zugeordnet. Die Wahl eines bestimmten Farbraumes hingegen ist für die Ergebnisse nicht ausschlaggebend: Swain und Ballard wählten als Farbräume den RGB-und den 'Opponent Color Axes'-Raum (definiert in [BB82]), unsere Tests führen wir erfolgreich im RGB- und Lab-Raum durch. Die Ergebnisse dieser Versuche sind später in Kapitel 7 ausführlich dargestellt.

2.1 Die Rückprojektion

Die Rückprojektion ist ein Algorithmus zur Lokalisierung eines Objektes (Modell) in einer Szene (Bild) oder Verfolgung eines bekannten Objektes – als Resultat liefert das Verfahren ein oder mehrere Koordinatenpaare (x_i, y_i), welche das vermutete Zentrum des gesuchten Objektes anzeigen. Die Entfernung des Objektes zur Kame-

ra und somit seine Größe sollten ungefähr gleich sein wie die Entfernung bei der Modellaufnahme. Das Verfahren gibt Antwort auf die Frage, wo in einer Szene die Farben, die das gesuchte Objekt charakerisieren, häufig vorkommen. Dazu werden die Farben, die in dem Modell vorherrschen, in der Szene betont, alle anderen Farben abgeschwächt. Annahme ist dabei, daß die Farben des gesuchten Objektes in der Szene auch grundsätzlich nur in dem Objekt vorkommen. Je stärker diese Annahme verletzt wird, desto ungenauer werden die Ergebnisse. Das Objekt wird in Bereichen vermutet, in denen sich die verstärkten Farben anhäufen.
Im folgenden wird als Bin eines Histogrammes der evtl. mehrdimensionale Eintrag in einem Histogramm bezeichnet. Die Variable n stehe für die Anzahl der Bins eines Histogrammes. Wir gehen ferner davon aus, daß oBdA alle Histogramme die gleiche Anzahl von Bins besitzen.

2.1.1 Theoretischer Hintergrund

Der Algorithmus geht bei dem Vorliegen einer unbekannten Szene in vier Schritten vor:

1. Berechnung des Verhältnis-Histogramms von gesuchtem Modell und gegebener Szene.
2. Rückprojektion des Verhältnis-Histogramms auf die Szene.
3. Glättung der Rückprojektion.
4. Ermittelung der ersten k lokalen Maxima.

Aus den Histogrammen M und I des Modells und des Bildes wird das sog. Verhältnis-Histogramm R berechnet:

$$R_i = \begin{cases} \min\left(1, \frac{M_i}{I_i}\right), & \text{falls } I_i \neq 0 \\ 0 \quad , & \text{sonst} \end{cases} \qquad i \in \{1, \ldots, n\} \, . \tag{2.1}$$

Man erhält ein Histogramm, in dem abgelesen werden kann, wie das Verhältnis der Vorkommen von Farben des Modells und der Szene ist. Das Histogramm enthält Einträge zwischen 0 und 1, wobei ein Wert nahe 1 bedeutet, daß die betreffende Farbe das Modell in der Szene gut charakterisiert, bei einem Wert nahe 0 ist die Farbe nicht geeignet für eine Charakterisierung. Kommt z.B. in einem Modell häufig die Farbe Rot vor, und enthält der Hintergrund der Szene ebenfalls diese Farbe, trägt das Rot nichts dazu bei, den Fokus der Aufmerksamkeit auf das gesuchte Objekt zu lenken. Enthält unser Objekt nur wenig blau, ist diese Farbe ebenfalls kein gutes Merkmal. In beiden Fällen wären die Einträge im Verhältnishistogramm sehr niedrig. Enthält in dem obigen Beispiel der Hintergrund nicht die Farbe Rot, ist dies ein gutes Charakteristikum; das entsprechende Bin erhält einen Wert nahe 1.

Sei h eine Funktion, die einen Farbwert einem Eintrag in einem Histogramm zuordnet, ferner sei $c_{x,y}$ der Farbwert des Pixels an der Stelle (x, y) in der Szene. Dann ergibt sich die Rückprojektion zu

$$b_{x,y} = R_{h(c_{x,y})} \quad \in [0,1] \,. \tag{2.2}$$

Die Rückprojektion besteht nur noch aus einem Kanal, auch wenn vorher Modell und Szene evtl. mehrkanalig waren. Abbildung 2.2a) zeigt das typische Ergebnis einer Rückprojektion.

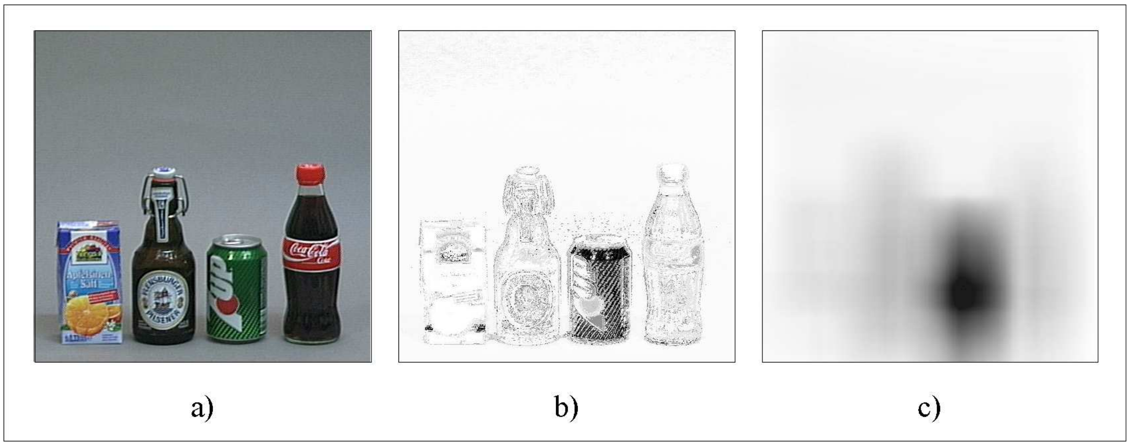


Abbildung 2.2: Ergebnisse der Rückprojektion eines RGB-Bildes und deren Glättung. Das gesuchte Objekt ist das 2. von rechts. a) RGB-Szene. b) Rückprojektion von a) mit dem gesuchten Objekt. Schwarz bedeutet Werte nahe 1, weiß nahe 0. c) Glättung von b) mit einem Boxfilter geeigneter Größe. Schwarz bedeutet sehr hohe Werte, weiß nahe 0.

Um zu ermitteln, wo in der Szene die betonten Farben im lokalen Mittel am häufigsten auftreten, wird die Rückprojektion mit einer speziellen Maske geglättet. Swain und Ballard schlagen einen Kreis mit Radius r vor, wobei r so gewählt sein sollte, daß der Kreis die ungefähre Größe des gesuchten Objektes hat.

$$D^r_{x,y} = \begin{cases} 1 & , \quad \text{falls} \quad \sqrt{x^2+y^2} < r \\ 0 & , \quad \text{sonst} \end{cases} \tag{2.3}$$

Stellt $*$ die Faltung dar, ergibt sich die geglättete Rückprojektion zu

$$b = D^r * b \,. \tag{2.4}$$

Die Glättung mit der Maske D^r bewirkt jeweils das Aufsummieren der rückprojizierten Werte um einen Aufpunkt. Häufen sich in der Rückprojektion viele hohe Werte in einem bestimmten Bereich, ergibt dies einen maximalen Ausschlag in der geglätteten Rückprojektion. Dies ist im Idealfall gegeben, wenn die Glättungsmaske genau über dem gesuchten Objekt liegt. Abbildung 2.2b) stellt die Glättung von Abbildung 2.2a) dar. Deutlich ist der Peak in dem Zentrum des gesuchten Objektes zu erkennen.

In der geglätteten Rückprojektion werden anschließend die ersten k lokalen Maxima $b_{(x_i,y_i)}$, $i \in \{1, \ldots k\}$ gesucht. Der Wert von k hängt davon ab, wieviele Objekte der gesuchten Art in der Szene vermutet werden, bzw. wieviele Aufmerksamkeitspunkte näher untersucht werden sollen.

2.1.2 Bewertung

2.1.2.1 Komplexität

Bei einer Bildgröße der Szene von $s \times s$ und des Modells von $m \times m$ ergibt sich zunächst die Berechnung des Objekt-Histogrammes mit s^2 Schritten. Danach muß das Verhältnis-Histogramm mit n Berechnungen gebildet und auf die Szene projiziert werden, was $s \times s$ Berechnungen erfordert. Nach der Faltung mit $s^2 + 2s^2m$ Schritten (inkrementelle Implementation) bleibt noch die Suche nach k lokalen Maxima mit ks^2 Schritten (vereinfachte Suche).
Da es Spezialrechner gibt, in denen extra Prozessoren für die Faltung vorhanden sind, und die selbst bei größeren Faltungsmasken nicht mehr als ein paar Millisekunden Rechenzeit benötigen, ist dieser Algorithmus auch für Echtzeitaufgaben anwendbar. In Kapitel 6 beschreiben wir u.a. die Echtzeitimplementierung der Rückprojektion auf einer MaxVideo 200 der Firma Datacube. Auf diesem Rechner hat der Algorithmus je nach Größe des gesuchten Objektes eine Laufzeit von ca. $0.005s$ bis $0.01s$. Die Zeitangaben beziehen sich hierbei nur auf die reine Laufzeit des Algorithmus'. Nicht berücksichtigt in dieser Angabe ist die Zeit, die benötigt wird, um ein Bild in digitaler Form zu erhalten (ca. $0.07s$). Auf die verschiedenen Laufzeiten gehen wir genauer in Kapitel 6 über die Implementierung auf dem Pipelinerechner ein.

2.1.2.2 Wahl der Faltungsmaske

In den Tests von Swain und Ballard sind die Objekte von gleicher Größe bzw. so skaliert, daß sie die gleiche Größe besitzen, und jeweils in einem gleichgroßen quadratischen Raum angeordnet. Die Objekte, welche wir in den späteren Versuchen verwenden, sind von unterschiedlicher Größe und meistens 2 bis 3 mal so hoch wie breit. Ferner haben die Objekte in den verschiedenen Testszenen unterschiedliche Abstände zueinander, so daß bei der Faltung mit D^r ein zu großer Einzugsbereich berücksichtigt wird. Dadurch befinden sich bei der Suche nach einem größeren Objekt teilweise bis zu drei Objekte im Einzugsbereich der Faltungsmaske. Das gefundene Zentrum des gesuchten Objektes stimmt deshalb oftmals nicht mit dem tatsächlichen Zentrum überein, sondern ist zu benachbarten Objekten verschoben.
Um diese Ungenauigkeit zu verhindern, wählen wir jeweils eine rechteckige Faltungsmaske, welche die ungefähren Abmessungen des gesuchten Objektes und nur Werte gleich 1 hat. Dies ist sinnvoll, da unsere Objekte grundsätzlich aufrecht auf einer Unterlage stehen und ihre Orientierung somit bekannt ist. Mit diesen Faltungsmasken wird das jeweilige Zentrum sehr genau lokalisiert.
Während die Rückprojektion unter Beutzung von D^r noch rotationsinvariant ist, hat sie diese Eigenschaft mit einer quadratischen Maske nicht mehr. Abbildung 2.3

verdeutlicht dies anschaulich: das gesuchte Objekt A befindet sich in Schräglage. Mit der Maske D^r wird der Attentionspunkt zum Objekt B verschoben, da beide Objekte einen Anteil gleicher Farben haben. Die quadratischen Masken hingegen haben die falsche Orientierung und beziehen immer nur einen Teil der rückprojezierten Werte in die Glättung ein.

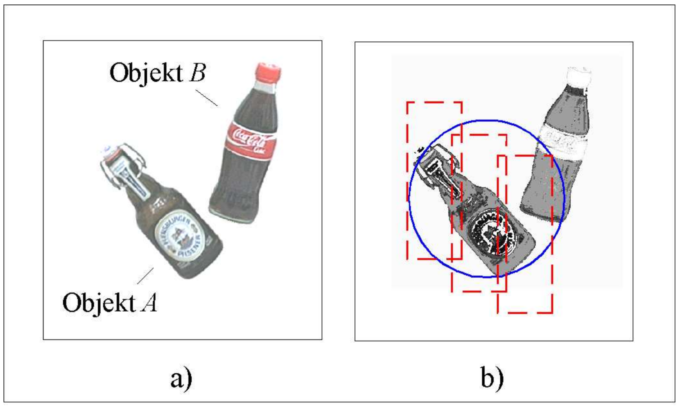


Abbildung 2.3: Einzugsbereiche von Glättungsmasken. Keine der Masken ist ideal für die Suche nach dem Objekt A. Die kreisförmige Maske hat einen zu großen Einzugsbereich, die quadratischen die falsche Orientierung. a) RGB-Szene; b) Rückprojektion der Szene mit dem Objekt A.

Dem Verfahren sind somit Grenzen gesetzt. Es kann nicht auf jede mögliche Szene erfolgreich angewendet werden, ohne daß vorher bestimmte Annahmen über Anzahl oder Orientierung der vorhandenen Objekte getroffen werden.

2.1.2.3 Aussagekraft der Maxima

Die Werte $b_{(x_i,y_i)}$ der geglätteten Rückprojektion sind abhängig von der Größe der Faltungsmaske: je größer die Maske desto größer die lokalen Maxima. Wir führen daher eine Normierung durch, indem wir die lokalen Maxima durch die Anzahl der Pixel der Faltungsmaske dividieren. Sind X und Y die Abmessungen der quadratischen Maske, ergibt sich dies zu

$$b^{rel}_{(x_i,y_i)} = \frac{b_{(x_i,y_i)}}{XY} \quad \in [0,1] \tag{2.5}$$

Im theoretischen Fall, daß die Farben des gesuchten Objektes in der Szene nur in dem Objekt vorkommen, zeigt sich im Zentrum des Objektes ein relatives Maximum von 1. Da weiterhin Lage und Entfernung des Objektes von der Kamera sowie die Beleuchtungsverhältnisse genau so wie zur Aufnahme des Modells sein müssen, sind dies Voraussetzungen, welche in der Praxis so gut wie nie erreicht werden.
Es stellt sich die Frage, ob nach der Normierung des oder der Maxima ein Schwellwert eingeführt werden kann, ab dessen Überschreiten mit großer Wahrscheinlichkeit angenommen werden kann, das gefundene Objekt sei das gesuchte? Dies müssen wir leider verneinen. Die Rückprojektion ist in höchstem Maße davon abhängig, welche

Farben neben denen des gesuchten Objektes noch in dem Bild vorkommen. Ein einfaches Beispiel soll verdeutlichen, wie leicht die Rückprojektion irregeführt werden kann. Abbildung 2.1.2.3a) zeigt eine Szene mit 4 verschiedenen Objekten, die der Einfachheit halber alle aus drei horizontal angeordneten Pixeln bestehen. Die vorkommenden Farben werden von 0 bis 3 durchnummeriert. Als gesuchtes Modell wird das Objekt 1 genommen, das sich farblich von den anderen unterscheidet.

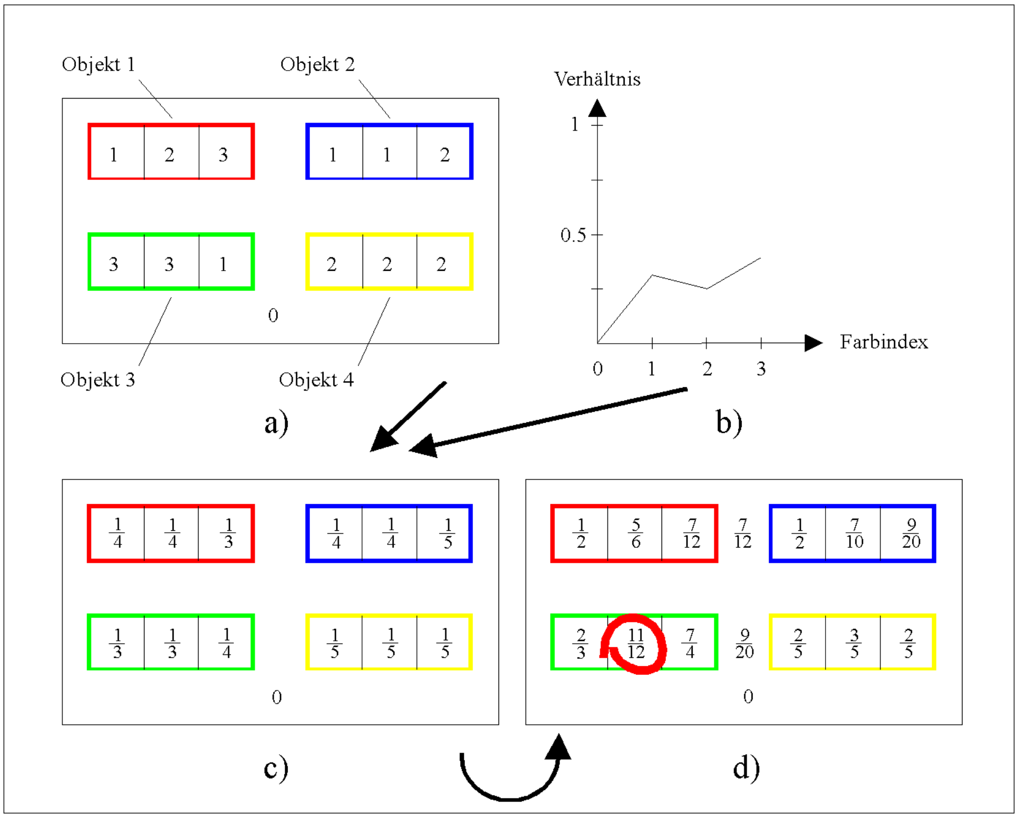


Abbildung 2.4: Abhängigkeit der Rückprojektion durch Farben, welche neben den Farben des gesuchten Objektes in der Szene vorkommen. a) Szene mit 4 einfachen Objekten zur Irreführung der Rückprojektion. Gesucht werden soll Objekt 1. Zahlen stellen Farbwerte der Pixel dar. b) Verhältnishistogramm aus der Szene und Objekt 1. c) Rückprojektion des Verhältnis-Histogramms auf die Szene. Zahlen sind Werte der Rückprojektion für jedes Pixel. d) Glättung der Rückprojektion mit mit einer 3×1-Maske. Im Zentrum von Objekt 3 ergibt sich das Maximum, das gesuchte Objekt 1 wird erst an zweiter Stelle lokalisiert.

An der geglätteten Rückprojektion der Szene mit Objekt 1 in Abbildung 2.1.2.3d) sieht man, daß Objekt 2 vor Objekt 1 erkannt wird: nach der Faltung mit einer 3×1 Maske ergibt sich im Zentrum von Objekt 2 ein Wert von $\frac{11}{12}$, in dem von Objekt 1 jedoch nur $\frac{10}{12}$. Es zeigt sich, daß die Farbe 2 in dieser Szene kein gutes Merkmal zur Lokalisierung von Objekt 1 ist, da diese Farbe auch in anderen Objekten häufig vorkommt (Wert des Bins im Verhältnis-Histogramm: 'nur' $\frac{1}{4}$). Die Farben 1 und 3 eignen sich zwar besser (Bin: $\frac{1}{3}$), jedoch besteht Objekt 3 nur aus diesen Farben und erhält insgesamt gesehen höhere Rückprojektionswerte als Objekt 1.

2.1.2.4 Feinheit der Quantisierung bei Bildung der Histogramme

Durch die Verwendung von Farbhistogrammen ist die Rückprojektion sehr anfällig gegenüber örtlicher und spektraler Beleuchtungsänderung. Schwankungen in der Helligkeit können aber durch eine gröbere Quantisierung bei der Bildung der Histogramme abgefangen werden. Farbhistogramme mit $8 \times 8 \times 8$ Bins spannen einen Raum auf, in dem eine gewisse Anzahl von Objekten unterschieden werden können. Durch diese Quantisierung fallen pro Farbachse 32 Farben in ein Bin, so daß bei Helligkeitsschwankungen noch ausreichend der ursprünglichen Farben dem selben Bin zugeordnet werden. So benutzen Swain und Ballard diese Quantisierung für eine Datenbank aus 66 Histogrammen und auch wir verwenden diese Einteilung in unsere späteren Versuchen. Die Kapazität eines Histogrammraumes werden wir genauer im Abschnitt der Intersektion untersuchen.

2.1.3 Schlußfolgerungen

Die Rückprojektion eignet sich sehr gut, um in Echtzeit in einer Szene eine visuelle Suche nach Objekten durchzuführen. Der Schnelligkeit der Methode steht entgegen, daß keine spezifischen Aussagen über das Vorhandensein von Objekten gemacht werden können, sondern die Attention nur auf Punkte maximaler Auftrittswahrscheinlichkeit des Objektes gerichtet wird. Diese Punkte ergeben sich je nach Wahl des Objektes, nach dem in der Szene gesucht wird, und daraus, welche Annahmen über Anzahl, Orientierung und Größe der vorkommenden Objekte gemacht werden. I.a. können die relativen Maxima, die sich aus der Rückprojektion ergeben, wegen des Einflusses von Hintergrundfarben und Farben anderer Objekte, die sich evtl. noch in der Szene befinden, nicht als schlüssiges Indiz für das Vorhandensein eines Objektes genommen werden. Lediglich bei Werten > 0.9 und < 0.1 kann davon ausgegangen werden, ein bestimmtes Objekt gefunden bzw. nicht gefunden zu haben. Wegen der Repräsentation der Modelle in einer Datenbank durch Histogramme ist die Datenbank wesentlich kleiner, als wenn darin die ursprünglichen Farbbilder der Modelle abgespeichert würden. Ein weiterer Vorteil der Repräsentation durch Histogramme ist die höhere Geschwindigkeit des Verfahrens gegenüber 'Template-Matching'-Verfahren, die einzelne Pixel von Objekt und Szene vergleichen.

2.2 Die Histogramm-Intersektion

Die Histogramm-Intersektion ist ein Algorithmus zur schnellen Identifikation eines Objektes. Voraussetzung ist, daß das zu vergleichende Objekt bereits lokalisiert und – wie bei der Rückprojektion – die Entfernung des Objektes zur Kamera ungefähr gleich der Entfernung während der Modellaufnahme ist. Der Algorithmus ist robust gegenüber Schwankungen in diesem Parameter. Das Verfahren vergleicht das Histogramm des Objektes mit den Histogrammen der Modelle einer Datenbank, woraus sich ein oder mehrere ähnliche Modelle ergeben. Da die Häufigkeit von Farbpixeln proportional zur Fläche der vorkommenden Farben ist, ergibt sich aus der Rotationsinvarianz der Flächen die Invarianz der Histogrammbildung. Wir bezeichnen im folgenden die Histogramm-Intersektion auch als Histogramm-Schnitt.

2.2.1 Theoretischer Hintergrund

Seien M und I die Histogramme des Modells und des Objektes. Grundsätzlich wird davon ausgegangen, daß die Histogramme über Bilder/Bildausschnitte derselben Größe gebildet werden, d.h. es gilt

$$\sum_{i=1}^{n} M_i = \sum_{i=1}^{n} I_i \ . \tag{2.6}$$

Dies ist für einige theoretische Überlegungen Voraussetzung, wird allerdings in der Praxis nicht immer erfüllt.
Als Intersektion ist die Summe der Pixel von Modell und Objekt definiert, die korrespondierende Farbwerte haben:

$$h(I, M) = \sum_{i=1}^{n} \min(I_i, M_i) \ . \tag{2.7}$$

Da dieser Wert wiederum von den Abmessungen des Objektes und des Modells abhängt, wird eine Normierung durch die Anzahl der Pixel des Modells durchgeführt.

$$H(I, M) = \frac{h(I, M)}{\sum_{i=1}^{n} M_i} \quad \in [0, 1] \tag{2.8}$$

Je kleiner $H(I, M)$ desto verschiedener sind die verglichenen Objekte in ihrer Farbzusammensetzung. Um nun das ähnlichste Objekt in einer Datenbank zu ermitteln, ist jedes Modellhistogramm mit dem Histogramm des Objektes zu schneiden. Je nach Aufgabenstellung und Größe der Datenbank kann ein Modell, welches den höchsten Intersektionswert erzielt, oder mehrere Modelle, deren Intersektionswert über einem bestimmten Schwellwert liegen, mit dem Objekt identifiziert werden.

2.2.2 Bewertung

2.2.2.1 Komplexität

Sei k die Anzahl der Modellhistogramme in der Datenbank, $s \times s$ die Bildgröße des Objektes. In der Online-Phase muß zunächst das Histogramm des unbekannten Objektes berechnet werden, also s^2 Schritte. Dann wird dieses Histogramm mit jedem aus der Datenbank geschnitten, was noch einmal $k \cdot n$ Berechnungen zur Folge hat. Insgesamt ergibt sich somit eine Komplexität von $s^2 + kn$. Swain und Ballard berichten von einer durchschnittlichen Laufzeit von 4 ms für das Erzeugen und 132 ms für das Matchen eines Histogrammes in einer Datenbank mit 66 Modellhistogrammen auf einer Sun Sparcstation 1 und einer Datacube FeatureMax. Angaben über die Dauer des Digitalisierungsprozesses werden nicht gemacht. Für größere Datenbaken schlagen Swain und Ballard eine Abänderung des Algorithmus' vor, die sogenannte *Inkrementelle Intersektion.* Da unsere Datenbank 'nur' aus 15 Modellen besteht, ist diese Methode für uns nicht interessant und wird hier nicht weiter behandelt. Ferner haben wir in unserer Implementierung auf dem Pipelinerechner eine durchschnittliche Laufzeit von $0.07s$ für das Erzeugen eines 3-dimensionalen Histogramms und das Schneiden mit sämtlichen Modellhistogrammen erreicht – eine Zeit, welche in der gleichen Größenordnung liegt wie die von Swain und Ballard. Wiederum bezieht sich die Zeitangabe auf die Laufzeit des Algorithmus' und beeinhaltet nicht die Dauer der Bilddigitalisierung.

2.2.2.2 Segmentierung

Bei der Bildung einer Datenbank können die Modelle 'per Hand' vom Hintergrund segmentiert werden, so daß in den Modellhistogrammen nur noch die Farben der jeweiligen Modelle vorkommen. Dadurch spielen bei der Intersektion eines Bildausschnittes, in dem sich eines unserer Objekte befindet, die Hintergrundfarben keine Rolle, da sie ignoriert werden. Das unbekannte Objekt muß vor der Identifikation nicht vom Hintergrund segmentiert werden, wie es z.B. bei Algorithmen, welche die Identifikation mittels 'template matching' vornehmen, der Fall ist. Auch die Größe des Fensters, das bewertet werden soll, ist nicht wesentlich. Es kann variabel gestaltet sein; Voraussetzung ist lediglich, daß das Objekt darin untergebracht werden kann. Dies unterscheidet die Intersektion von Algorithmen, bei denen Voraussetzung die Gleichheit der Bildgrößen ist, und macht den Algorithmus somit flexibler.

2.2.2.3 Aussagekraft des Maximums

Ein Intersektionswert nahe 1 bedeutet, daß das unbekannte Objekte und das zugeordnete Modell eine nahezu identische Farbzusammensetzung haben, nicht aber, daß das unbekannte Objekt mit dem in der Datenbank übereinstimmt. Es können Objekte sein, die in Geometrie und Form völlig verschieden sind und sich nur farblich sehr ähneln. Z.B. würde sich bei einem Kreis und einem Dreieck von gleicher Farbe und Flächeninhalt ein Intersektionswert von 1 ergeben. So sehen auch Swain und Ballard[SB91] die Schwäche in ihrem Verfahren:

> *"(...) the technique is capable of indexing into a large database, that is, eleminating most of the possible matches leaving only a small number for further consideration.“* (S. 15)
>
> *"Other, more expensive, matching techniques can be used to verify, which of the scoring models is the correct one, so it is not crucial that the correct model is always the best one.“* (S. 16)

Swain und Ballard führen einen Schwellwert ein, ab dessen Überschreiten das Ergebnis einer Intersektion erst für aussagekräftig gehalten wird. Sie wählen einen Wert von 0.6, den wir durch unsere Experimente bestätigt finden werden. In den Histogrammen einiger Modelle treten Farben des Hintergrundes auf (z.B. fällt die Farbe der Etiketten der Modelle *Claust* und *Flens* und die des Hintergrundes in dasselbe Bin). Dadurch würde bei dem Schnitt des leeren Hintergrundes mit den Modellen der Datenbank ein Intersektionswert von 0.47 erreicht und der Hintergrund als Objekt *Claust* identifiziert. Die Einführung eines Schwellwertes löst dieses Problem. Der Algorithmus liefert in diesem Fall das Ergebnis, daß ein unbekanntes Objekt vorliegt.

2.2.2.4 Objekte unterschiedlicher Größe

Swain und Ballard verwenden in ihren Experimenten Objekte, welche die gleiche Größe besitzen bzw. auf die gleiche Größe skaliert sind. Diese Einschränkung wird für viele praktische Fälle nicht einzuhalten sein, in denen auch Objekte unterschiedlicher Größe identifiziert werden sollen. Eine Automatisierung der Skalierung eines unbekannten Objektes ist schwierig, da das Objekt zunächst unbekannt ist und wenig Aussagen über seine Größe gemacht werden kann. Da kleine Objekte eine geringere Anzahl Pixel beinhalten als große Objekte, ist auch der Normierungsfaktor der zugehörigen Modelle $\sum_{i=1}^{n} M_i$ geringer. Kommen in einem großen Modell u.a. auch Farben eines kleinen Modells vor, kann es durch den geringen Normierungsfaktor des kleinen Modells passieren, daß bei der Identifikation ein großes Objekt fälschlich einem kleinen Modell zugeordnet wird. Diese Schwäche der Intersektion wird, wie wir später sehen werden, durch unseren Algorithmus zur Farbkonstanz behoben.

2.2.2.5 Statistische Größen

Um Aussagen über die Intersektion und Testergebnisse der Intersektion machen zu können, führen wir in diesem Abschnitt verschiedene Begriffe ein, die u.a. auch von Swain und Ballard sowie von anderen Autoren benutzt werden.

Rang, Ähnlichkeit und Trefferquote Der *Rang* eines Objektes gibt an, wie gut das Objekt erkannt wurde. So bedeutet ein Rang von 1, daß das zugehörige Modell aus der Datenbank den höchsten Intersektionswert mit dem Objekt hat, ein Rang von 2 den zweithöchsten usw. Damit verbunden ist auch der Begriff der *Ähnlichkeit*, welcher durch den Rang definiert wird. Sei k die Anzahl der Modelle in der Datenbank, r der Rang eines Objektes. Die Ähnlichkeit eines Objektes ist dann definiert als

$$e = \frac{k-r}{k-1} \quad \in [0,1]\,. \tag{2.9}$$

Eine Ähnlichkeit von 0.9 besagt, daß das zugeordnete Modell einen höheren Intersektions-Wert mit dem Objekt besitzt als 90% der übrigen Modelle der Datenbank. Die *Trefferquote* läßt Aussagen über Erfolg oder Nichterfolg von Testergebnissen zu. Sie ist der Durchschnitt der Ähnlichkeiten aller Objekte und ergibt sich zu

$$t = \frac{\sum_{i=1}^{k} e_i}{k}\,. \tag{2.10}$$

Betrachtet man den Rang eines Objektes für sich, kann dies allerdings zu Fehlschlüssen führen. Hat ein Objekt z.B. den Rang 1, jedoch einen Intersektionswert von 0.5, wird es wegen des Schwellwertes trotz seines Ranges nicht positiv identifiziert. In diesem Fall kann man dem Objekt künstlich dem niedrigsten Rang zuordnen. Dieses Phänomen tritt verstärkt bei den Tests mit den farbig beleuchteten Objekten auf und wird dort differenziert behandelt.

Maximale Anzahl der Zuordnungen Um Informationen darüber zu erhalten, wie ähnlich schon die Modelle in der Datenbank sind, hat Stricker[Str94] den Begriff der *Maximalen Anzahl der Zuordnungen* eingeführt. Unter der Annahme von (2.6), daß die Histogramme über Bilder der gleichen Größe gebildet wurden, definiert die Intersektion eine Metrik im Histogrammraum. Ist d_{L_1} die L_1-Metrik

$$d_{L_1}(M, I) = \sum_{i=1}^{n} |M_i - L_i| \; , \tag{2.11}$$

so gilt

$$1 - H(M, I) = \frac{1}{2T} \, d_{L_1}(M, I) \quad , \text{mit } T = \sum_{i=1}^{n} M_i \; . \tag{2.12}$$

Zwei Histogramme, die als Abstand der L_1-Metrik mindestens den Wert t haben, heißen t-verschieden. Dabei wird t relativ zum größtmöglichen Abstand $2T$ angegeben, d.h. sollen 2 Histogramme t-verschieden sein, müssen sie mindestens den Abstand $t \cdot 2T$ haben. Als maximale Anzahl der Zuordnungen für einen Schwellwert t ist die maximale Anzahl der Modelle einer Datenbank definiert, die in dem Histogrammraum innerhalb einer Hyperkuggel mit Radius t liegen. Im Idealfall sollte der Schwellwert t so gewählt werden, daß die maximale Anzahl der Zuordnungen 1 ist. Denn sollten sich schon die Modelle sehr ähnlich sein, kann das Matchen mit unbekannten Objekten keine eindeutige Aussage zulassen.

Toleranz Eine 3. Meßgröße ist die sog. *Toleranz*. Sie bezieht sowohl die getesteten Objekte als auch die Modelle in die Berechnung ein. Bezeichnet r den Rang eines Objektes und M_r den Intersektionswert des zugehörigen Modells sowie M_{r-1} den nächst niedrigeren Intersektionswert, so ist die Toleranz definiert als $M_r - M_{r-1}$. Je geringer die Toleranz eines Objektes ist, desto ähnlicher ist es neben dem dazugehörigen Modell einem anderen Modell in der Datenbank. Die durchschnittliche Toleranz läßt somit Aussagen auf die Robustheit des Verfahrens sowie Qualität der Datenbank und der unbekannten Objekte zu. Ergibt z.B. eine Testreihe mit Objekten, die weder unter dem Problem der Verdeckung noch unter verschiedenen Entfernungen zur Kamera aufgenommen werden, eine geringe Toleranz, wird die Trefferquote schon bei geringen Schwankungen dieser Parameter sinken, obwohl das Verfahren der Intersektion dagegen eigentlich robust ist.

Kapazität Bei der Bildung einer Histogrammdatenbank stellt sich die Frage, wie fein die Quantisierung der Histogramme gewählt werden sollte, oder von der anderen Seite gesehen, wie viele Histogramme bei vorgegebener Anzahl der Bins in einem Histogrammraum noch genügend großen Abstand voneinander haben.
Mit dieser Frage hat sich Stricker[Str94] befaßt. Dazu führte er den Begriff der *Kapazität eines Histogrammraumes* ein. Ein Histogrammraum $\mathcal{H}$ ist definiert als

$$\mathcal{H} = \{(\mathrm{b}_1, \cdots, \mathrm{b}_\mathrm{n}) \mid \mathrm{b}_\mathrm{i} \geq 0 \text{ für } 1 \leq \mathrm{i} \leq \mathrm{n}, \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{b}_\mathrm{i} = \mathrm{N}\}\,, \tag{2.13}$$

wobei n die Anzahl der Bins ist und davon ausgegangen wird, daß die Histogramme über Bilder gebildet werden, die die gleiche Anzahl von Bildpunkten N beinhalten. Für einen Histogrammraum $\mathcal{H}$, die zugehörige L_1-Metrik d und einen Schwellwert t ist die Kapazität $\mathcal{K}(\mathcal{H}, d, t)$ definiert als die maximale Anzahl von t-verschiedenen Histogrammen, die in den Histogrammraum passen. Stricker beweist folgende Ungleichung

$$\mathcal{K}(\mathcal{H}, d, t) \geq \max_{\omega \leq n} \frac{1}{q^{\lceil \frac{\omega t}{2T} \rceil - 1}} \binom{n}{\omega}\,, \tag{2.14}$$

wobei q die kleinste Primzahl mit $q \geq n$ und T wie in (2.12) definiert ist. Die rechte Seite von (2.14) ist eine untere Grenze für die Kapazität eines Histogrammraumes. In Abbildung 2.5 haben wir die Kapazitäten für 3 verschiedene Schwellwerte und einer Anzahl bis zu 512 Bins dargestellt. Selbst wenn jedes Histogramm von jedem anderen einen L_1-Abstand von 0.7 hat, ist die Kapazität bei 150 Bins bereits mindestens 888, bei 512 Bins sogar größer als 24757.

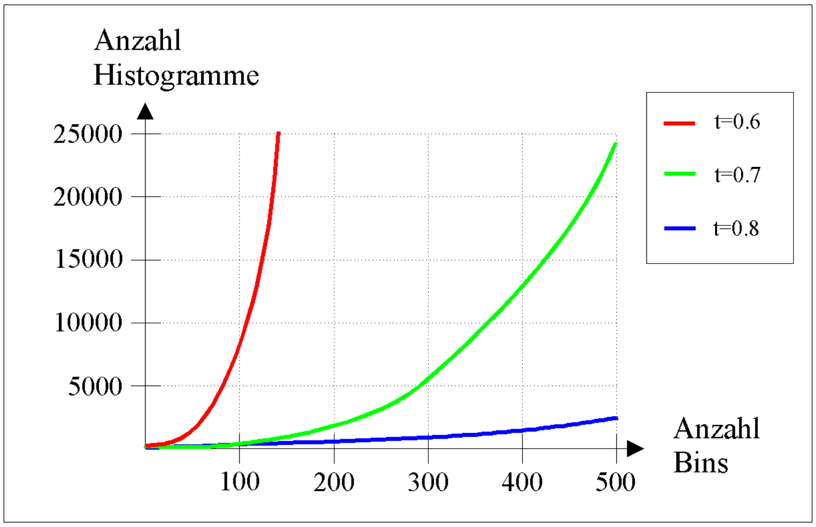


Abbildung 2.5: Kapazitäten eines Histogrammraumes bei Zugrundelegung von verschiedenen Mindestabständen der einzelnen Histogramme untereinander. Dargestellt sind die Verläufe für die Mindestabstände $t = 0.6, 0.7, 0.8$.

Der durchschnittliche Abstand der Histogramme unserer Datenbank ergibt sich zu 0.73. Demnach könnten nach der von Stricker definierten Kapazität wenigstens noch

24000 weitere Histogramme untergebracht werden. Dies erscheint ein wenig viel, wenn man bedenkt, daß die 32 Objekte von Swain und Ballard (in der Datenbank durch 66 Histogramme repräsentiert) schon nicht mehr eindeutig identifiziert werden konnten. Die Diskrepanz läßt sich durch zwei Gründe erklären:

- In den theoretischen Überlegungen zur Kapazität wird eine Gleichverteilung der Histogramme über dem Histogrammraum zugrundegelegt.
- Eine Vielzahl 'künstlicher' Histogramme, die in einer Histogrammdatenbank von Bildern realer Objekte nicht vorkommen, werden mitgezählt.

In Abbildung 2.6 haben wir 3 solcher Histogramme dargestellt. Sämtliche Bins haben den Wert 0, lediglich ein Bin hat den Wert N.

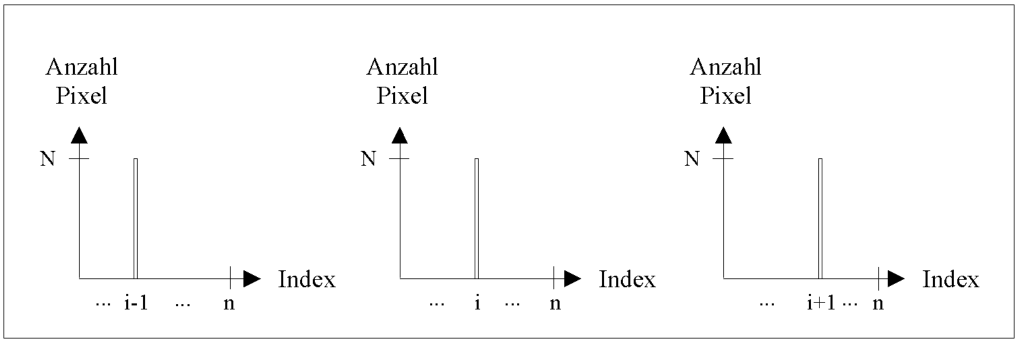


Abbildung 2.6: Drei Künstliche Histogramme, die jeweils den maximalen Abstand von 1 haben, und die in einer aus natürlichen Objekten bebildeten Datenbank nicht vorkommen würden.

Natürliche Objekte haben oftmals Oberflächen, die sich in verschiedene Farbbereiche unterteilen lassen. Solch ein Farbbereich verteilt sich aber nicht nur über ein einziges sondern gaussförmig über mehrere Bins eines Farbhistogrammes. Dies kann zum einen an einer verrauschten Aufnahme liegen und zum anderen daran, daß Punkte einer gleichfarbigen Oberfläche durch Krümmung oder Beleuchtungsvariationen unterschiedliche Reflektionen liefern.
Die Histogramme aus Abbildung 2.6 werden somit in einer Histogrammdatenbank realer Objekte nicht auftauchen. Die Abschätzung der Kapazität eines Histogrammraumes durch (2.14) ist mit Vorsicht anzuwenden. Der Mindestabstand t sollte eher zu hoch als zu niedrig gewählt werden.

2.2.2.6 Robustheit und Invarianz

Da sich Farbhistogramme von Objekten unter Rotation um die optische Achse nicht und mit Entfernung des Objektes von der Kamera und unter Rotation um andere Rotationsachsen nur langsam ändern, ist das Verfahren robust gegenüber diesen Einflüssen. Ferner stellt auch das Problem der Verdeckung erst ab einem gewissen Grad ein wirkliches Problem dar: Swain und Ballard erzielen selbst mit Bildern von Objekten, die zu 1/3 verdeckt sind, gute Ergebnisse. Da das Hauptziel dieser Arbeit in der Farbkonstanz liegt, haben wir diese Eigenschaften nicht weiter untersucht. Lediglich der Rotation um die optische Achse haben wir uns zugewendet, bedingt durch den Verlust dieser Invarianz durch unseren Algorithmus zur Farbkonstanz. Wird ein Objekt rotiert, treffen die Lichtstrahlen der Beleuchtungskörper in einem anderen Winkel auf die Punkte der Oberfläche als vor der Rotation. Es ändert sich somit auch das Farbsignal dieser Punkte, was sich – je nach der Ausleuchtung des Aufnahmeraumes – mehr oder weniger stark bemerkbar macht. In der Praxis zeigt sich das Verfahren der Intersektion auf Farbhistogrammen nicht rotationsinvariant.

2.2.3 Schlußfolgerungen

Die Histogramm-Intersektion ist ein effizienter Algorithmus, um ein lokalisiertes Objekt, dessen Entfernung zur Kamera ungefähr bekannt ist, mit Objekten einer Datenbank zu vergleichen. Je nach Größe der Datenbank und Verteilung der Farben der Modelle im jeweiligen Farbraum ist die Feinheit der Quantisierung bei der Bildung der Histogramme zu wählen. Meßgrößen wie die Toleranz oder die maximale Anzahl der Matches können herangezogen werden, um zu bestimmen, ob die vorgenommene Quantisierung ausreichend ist. Bei wachsendem Umfang einer Datenbank werden zwangsläufig auch Objekte mit ähnlichen Farbzusammensetzungen vorkommen, so daß eine feinere Quantisierung notwendig wird.
Wesentliche Vorteile des Histogramm-Schnittes gegenüber anderen Verfahren zur Objektidentifikation sind die Komplexität und die Implementierbarkeit auf Spezialrechnern wie der MV200, wodurch der Algorithmus auch für Echtzeitprobleme anwendbar ist. Bei einem unbekannten Objekt muß keine Segmentierung vor der eigentlichen Identifikation vorgenommen werden. Weiterhin ist das Verfahren robust gegen geringe Entfernungsschwankungen und Verdeckung des Objektes. Die theoretische Rotationsinvarianz wird in der Praxis nur erreicht, wenn die zu identifizierenden Objekte von allen Seiten gleichmäßig ausgeleuchtet werden.
Ein Nachteil ist, daß die Intersektion auf Farbhistogrammen nicht zwischen Objekten verschiedener Größe differenzieren kann. Ferner gibt der Intersektionswert nur Auskunft über die Ähnlichkeit der Farbverteilung eines unbekannten Objektes und dem gematchten Modell einer Datenbank, nicht jedoch über Farbanordnungen oder Formen.
Zusammen mit dem Verfahren der Rückprojektion ergibt sich die Möglichkeit einer attentiven Suche nach Objekten in einer Szene: in einem ersten Schritt wird mit der Rückprojektion eine gewissen Anzahl von Aufmerksamkeitspunkten ermittelt, welche in einem zweiten Schritt lokal mit der Intersektion untersucht werden können.
Größter Nachteil ist die Anfälligkeit gegenüber Beleuchtungsschwankungen, die einen Einsatz des Verfahrens nur unter Beleuchtungsbedingungen zulassen, die vollständig kontrolliert werden wie z.B. in industriellen Umgebungen. Ein Einsatz in natürlichen Beleuchtungsumgebungen ist jedoch ausgeschlossen.

Kapitel 3

Weiterentwicklungen der Verfahren

Nach der Veröffentlichung des Artikels von Swain und Ballard sind mehrere Artikel erschienen, die sich mit Weiterentwicklungen sowohl der Rückprojektion als auch der Intersektion befassen. Wir stellen hier vier dieser Arbeiten aus den Jahren 1992-1995 vor, welche die folgenden Aspekte behandeln:

- Objektlokalisierung mittels wahrscheinlichkeitstheoretischer Annahmen,
- genauere Lokalisierung eines Objektes durch lokale Histogrammbildung,
- Farbkonstanz durch Intersektion von Farbdifferenzen statt Farben und
- Ähnlichkeit von Objekten durch Berechnung von Momenten.

Obwohl zentraler Punkt dieser Arbeit die Farbkonstanz ist, sind auch die neueren Ansätze zur Objektlokalisierung und die Objektidentifizierung durch Momente von Interesse, die evtl. Schwächen der Rückprojektion und der Intersektion beheben. Wir betrachten kritisch die Vor- und Nachteile, die diese Verfahren mit sich bringen.

3.1 Lokalisierung durch Wahrscheinlichkeiten

Schiele und Crowley[SC96] verwenden multidimensionale Histogramme von Filterantworten verschiedener Gauss-Ableitungen zur Objektlokalisierung. Mittels dieser Histogramme schließen sie über Wahrscheinlichkeiten auf das Vorkommen von gesuchten Objekten. Die Wahrscheinlichkeiten müssen nicht auf den Gauss-Filterantworten berechnet werden, es können auch andere Bildeigenschaften Verwendung finden. Da wir dieses Verfahren explizit mit der Rückprojektion vergleichen wollen, benutzen wir die Farbeigenschaften der Objekte.

3.1.1 Theoretischer Hintergrund

Das Verfahren beantwortet die Frage, wo in einem Bild ein gesuchtes Objekt aufgrund bestimmter Objekteigenschaften am wahrscheinlichsten vorkommt und kann in zwei Schritte aufgeteilt werden:

1. Berechnung von Auftrittswahrscheinlichkeiten für jeden Bildpunkt und
2. Ermittelung von lokalen Anhäufungen der Wahrscheinlichkeiten.

Der 2. Schritt wird typischerweise genau wie bei der Rückprojektion durch die Glättung mit einer Maske erreicht, welche die ungefähren Abmessungen des gesuchten Objektes hat.
Zum ersten Schritt muß für jeden Bildpunkt die Frage beantwortet werden, mit welcher Wahrscheinlichkeit dieser Punkt, der einen bestimmten Farbwert hat, zu dem gesuchten Objekt gehört. Dazu führen wir zunächst folgende Variablen ein:

- O_n mit $n \in \{1, \dots, K\}$ bezeichnet das Objekt n aus einer Datenbank, welche K Objekte beeinhaltet.
- M_c mit $c \in \{0, \dots, N\}$ steht für das Merkmal c. Da wir Farbeigenschaften verwenden, ist M_c einer bestimmten von N möglichen Farben zuzuordnen.
- $H_j(O_n)$ bedeutet den Wert des j. Bins in dem Histogramm des Objektes n.

Die Beantwortung der obigen Frage stellt sich dar als die Berechnung der bedingten Wahrscheinlichkeit $P(O_n|M_c)$ unter Vernachlässigung des Auftretens von Hintergrundfarben. Nach der Bayes-Regel bestimmt sich diese zu

$$P(O_n|M_c) = \frac{P(M_c|O_n) \cdot P(O_n)}{P(M_c)} \,. \tag{3.1}$$

Dabei sind die einzelnen Wahrscheinlichkeiten folgendermaßen definiert:

$$\begin{aligned} P(O_n) &\leftrightarrow \text{W.keit, daß der Bildpunkt zum Objekt } n \text{ gehört.} \\ &= \frac{\sum_{j=0}^{N} H_j(O_n)}{\sum_{i=1}^{K}\sum_{j=0}^{N} H_j(O_i)} \quad , \end{aligned} \tag{3.2}$$

$$\begin{aligned} P(M_c) &\leftrightarrow \text{W.keit, daß ein Bildpunkt das Merkmal } c \text{ aufweist.} \\ &= \frac{\sum_{i=1}^{K} H_c(O_i)}{\sum_{i=1}^{K}\sum_{j=0}^{N} H_j(O_i)} \quad \text{und} \end{aligned} \tag{3.3}$$

$$\begin{aligned} P(M_c|O_n) &\leftrightarrow \text{W.keit, daß ein Bildpunkt das Merkmal } c \text{ hat, unter} \\ & \quad \text{der Annahme, daß er zum Objekt } n \text{ gehört.} \\ &= \frac{H_c(O_n)}{\sum_{j=0}^{K} H_j(O_n)} . \end{aligned} \tag{3.4}$$

Eingesetzt in (3.1) erhält man nach dem Kürzen gleicher Faktoren aus Zähler und Nenner

$$P(O_n|M_c) = \frac{H_c(O_n)}{\sum_{i=1}^{K} H_c(O_i)} \quad , \tag{3.5}$$

einen Wert, der für jeden Bildpunkt leicht zu berechnen ist. Die nachfolgende Glättung und Suche nach lokalen Maxima ergibt Punkte mit hoher Auftrittswahrscheinlichkeit für ein gesuchtes Objekt.

3.1.2 Bewertung

3.1.2.1 Komplexität

Wir nehmen eine Bildgröße der Szene von $s \times s$ und des Modells von $m \times m$ Punkten an. Für jede (in der Szene vorkommende) Farbe ist die bedingte Wahrscheinlichkeit aus (3.5) zu berechnen. Diese Berechnung kann jedoch offline durchgeführt und in der Online-Phase darauf zurückgegriffen werden. Es bleibt die 'Projektion' der Wahrscheinlichkeiten auf das Bild mit s^2-Berechnungen. Die anschließende Glättung und Suche nach k Maxima unterscheidet sich nicht von den Schritten, die an dieser Stelle bei der Rückprojektion durchzuführen sind (siehe Abschnitt 2.1.2.1). Insgesamt ergibt sich eine Komplexität von $(k+4)s^2$ im Vergleich zu der Komplexität der Rückprojektion von $(k+5)s^2 + K$. Die Methode ist somit über den Wert s^2 schneller als das Verfahren von Swain und Ballard.

3.1.2.2 Schlüsse zur Rückprojektion

Verglichen mit dem Wert der Rückprojektion

$$b_c = \begin{cases} \min\left(1, \frac{M_c}{I_c}\right), & \text{falls } I_c \neq 0 \\ 0 \quad , & \text{sonst} \end{cases} \tag{3.6}$$

mit M und I als Histogramme des Modells und der Szene erhält man das Ergebnis, daß die bedingte Wahrscheinlichkeit (3.5) ein Spezialfall der Rückprojektion ist für den Fall, daß in der Szene sämtliche Modelle und keine Hintergrundpixel vorhanden sind. Die Objekteigenschaften, die sich somit für die Lokalisierung eignen, bestimmen sich nur durch die Häufigkeiten der Farbvorkommen der Modelle. Hintergrundfarben fließen in diesen Prozeß nicht ein und können in der Wahrscheinlichkeitsberechnung evtl. ebenso hohe Werte erhalten wie einige Objektfarben.

3.1.3 Schlußfolgerungen

Die Lokalisierung mittels Annahmen über Auftrittswahrscheinlichkeiten ist ein Spezialfall der Rückprojektion. Somit gelten für das Verfahren die Aussagen, welche wir in den Abschnitten 2.1.2.2, 2.1.2.3 und 2.1.2.4 über die Wahl der Faltungsmaske, die lokalen Maxima und Anzahl der möglichen Farben (Feinheit der Quantisierung der Farbhistogramme) getroffen haben. Die Komplexität des Algorithmus' ist geringer als die der Rückprojektion. Haben Objekte gleiche Merkmale, wird dies in der Berechnung berücksichtigt, indem diese Farben geringere Wahrscheinlichkeiten erhalten. Das Problem der Einflußnahme anderer Objektfarben bleibt jedoch bestehen, wenn sich neben dem gesuchten Objekt weitere Objekte in einer Szene befinden. Betrachten wir unser Beispiel aus Abbildung 2.1.2.3 auf Seite 12 und nehmen an, daß die vier Objekte den gesamten Inhalt einer Datenbank darstellen, ergibt sich weiterhin die Lokalisierung des richtigen Objektes erst an zweiter Stelle. Treten in dem Hintergrund Farben des gesuchten Objektes auf, wird dies in Wahrscheinlichkeiten nicht berücksichtigt: das einmal berechnete Merkmal behält seine 'Güte', auch wenn es offensichtlich verstärkt im Hintergrund auftaucht. Dies wird bei der Rückprojektion vermieden.

3.2 Lokale Histogramme

Ennesser und Medioni[EM95] entwickeln einen Algorithmus zur Lokalisierung eines Objektes in einer Szene – basierend auf den Methoden von Swain und Ballard – und vergleichen diesen mit der Rückprojektion. Das Verfahren schneidet lokale Histogramme der Szene mit dem Histogramm des gesuchten Modells und liefert so Punkte, an denen, wie bei der Rückprojektion, das Objekt mit höherer Wahrscheinlichkeit zu finden ist als an anderen Punkten. Ennesser und Medioni postulieren ihrem Verfahren eine höhere Genauigkeit und geringere Komplexität als der Rückprojektion.

3.2.1 Theoretischer Hintergrund

Seien M und I die Histogramme des gesuchten Modells und der Szene. Von jedem Punkt einer Szene wird für einen bestimmten Einzugsbereich, der i.a. die Abmessungen des gesuchten Objektes hat, ein lokales Histogramm L_i gebildet. Jedes lokale Histogramm wird mit dem des gesuchten Modells geschnitten. Ennesser und Medioni wählen eine gewichtete Intersektion, um das Verhältnis der Farbverteilung von Szene und Modell zu berücksichtigen. Für die Gewichte W_i schlagen sie die Einträge des Verhältnishistogramms R aus (2.1) auf Seite 8 vor, um das Vorkommen von Hintergrundfarben zu berücksichtigen. Je nach Aufgabenstellung können durchaus andere Gewichte bestimmt werden. Somit ergibt sich für jeden Punkt der Szene ein lokaler Schnitt von

$$l = \sum_{i=1}^{n} W_i \min(L_i, M_i) \tag{3.7}$$

Die ersten k lokalen Maxima stellen Punkte dar, an denen das gesuchte Objekt vermutet wird.

3.2.2 Bewertung

3.2.2.1 Komplexität

Seien $s \times s$ und $m \times m$ die Abmessungen der Szene und des Modells. Ennesser und Medioni geben für die lokale Histogrammbildung eine Komplexität $O((s+m)^2 m)$ an, wobei sie die Berechnung der lokalen Histogramme und des jeweiligen Schnittes durch Updates implementieren und somit statt $m^2 + n$ Berechnungen für einen lokalen Schnitt nur $2m$ Schritte benötigen. Im Vergleich zur Rückprojektion, die eine Komplexität $O((s+m)^2 m)$ besitzt, hat der lokale Histogrammschnitt eine geringere Komplexität, falls $\log_2(s+m) < m$, also $s > (2^m - m)$. Dies ist der Fall, falls das Objekt sehr viel kleiner als die Szene ist, was, so Ennesser und Medioni, in den meisten Suchproblemen der Fall sei. Für unsere Experimente, die wir später beschreiben werden, mit 251×168 großen Szenen würde dies bedeuten, die Objekte dürfen nicht mehr als 8×8 Pixel beinhalten. Da unsere Modelle jedoch Abmessungen zwischen 20×20 und 40×100 Pixeln haben, ist die Komplexität des lokalen Histogrammschnittes in diesem Fall wesentlich höher als die der Rückprojektion. Die unüblichen Maße der Bildgrößen ergeben sich aus technischen Gründen, die wir im Kapitel über die Implementation auf dem Pipelinerechner näher erleutern.
Die Rückprojektion läßt sich effektiv auf einem Spezialrechner implementieren. Der lokale Histogrammschnitt würde sich zwar ebenfalls auf unserem Pipelinerechner implementieren lassen, jedoch müßte das Histogramm und der Schnitt mit den Modellen der Datenbank jeweils neu berechnet werden, d.h. die Möglichkeit der Updates ist hier nicht gegeben. Wir schätzen dadurch eine Laufzeit auf der MV200 von mehr als 10 s.

3.2.2.2 Genauigkeit

Ennesser und Medioni geben neben der Komplexität einen höheren Genauigkeitsgrad als wesentlichen Vorteil des lokalen Histogrammschnittes gegenüber der Rückprojektion an. Ferner soll ihr Verfahren bei falschen Annahmen (z.B. der Größe des gesuchten Objektes) wenigstens so gut sein wie die Rückprojektion.

> *"On the other hand, when the scaling is badly overestimated, we can prove that the degenerate Local Histograms algorithm still performs as well as backprojection: (...)"* (S. 887)

Diesen 'Beweis' gilt es kritisch zu hinterfragen: Ennesser und Medioni argumentieren für den Fall, daß das gesuchte Objekt kleiner als angenommen ist, der maximal mögliche Intersektionswert selbst im Zentrum des Objektes nicht erreicht wird. Daher gilt – so die Autoren – insbesondere für den Aufpunkt im Zentrum des Objektes

$$\min(L_i, M_i) = L_i\,. \tag{3.8}$$

Somit ergibt sich für den lokalen Schnitt mit den Gewichten $W_i = R_i$

$$\sum_{i=1}^{n} W_i \min(L_i, M_i) = \sum_{i=1}^{n} \frac{M_i}{I_i} L_i \tag{3.9}$$

Der rechte Teil von (3.9) ist genau der Wert der geglätteten Rückprojektion, wobei eine quadratische Glättungsmaske benutzt wird, wie wir sie in Abschnitt 2.1.2.2 vorgeschlagen haben.
Der Argumentation von (3.8) können wir uns nicht anschließen. Gerade für dieses Beispiel (Objekt ist kleiner als erwartet) treten im Einzugsbereich des lokalen Histogrammes vermehrt Pixel des Hintergrundes auf, die zu dem Schnitt $\min(L_i, M_i)$ nichts beitragen. (3.8) bedeutet aber, daß alle Hintergrundpixel auch im Modell vorkommen, was nicht sein kann. Für den Fall hingegen, daß das gesuchte Objekt in Größe und Farbzusammensetzung gleich dem Modell ist, ist (3.8) richtig, da dann $M_i = L_i$. Somit ist auch (3.9) gültig, was bedeutet: selbst wenn Modell und gesuchtes Objekt vollkommen in ihrer Farbzusammensetzung und Größe übereinstimmen, ist der lokale Histogrammschnitt mit den Gewichten $W_i = R_i$ 'nur' so gut wie die Rückprojektion.
Der Leser mag sich davon anhand unseres Beispiels aus Abbildung 2.1.2.3 auf Seite 12 überzeugen. Mit den Gewichten $W_i = R_i$ ergeben sich die gleichen Werte wie in der Rückprojektion bedingt durch (3.8) und (3.9). Wählt man hingegen $W_i = 1, i \in \{1, \ldots, n\}$, ist die lokale Histogrammbildung genauer als die Rückprojektion; in unserem theoretischen Beispiel wird dann das richtige Objekt an erster Stelle erkannt, da Farben außerhalb des Einzugsbereiches der Maske keine Rolle spielen.

3.2.3 Schlußfolgerungen

Der lokale Histogrammschnitt hat – bzgl. der Szenen und Objekte, die wir in unseren Experimenten benutzen – einen höheren Aufwand als die Rückprojektion. Lediglich für sehr kleine Objekte liegt eine geringere Komplexität dieses Verfahrens gegenüber der Rückprojektion vor. Eine Implementierung auf der MV200 wäre nicht echtzeitfähig. Grundsätzlich ist die lokale Histogrammbildung genauer als die bloße Rückprojektion, bei ungünstiger Wahl der Gewichte tritt jedoch auch bei diesem Verfahren eine Abhängigkeit von den Farben des Hintergrundes oder anderer vorhandener Objekte auf. Die höhere Genauigkeit kann aber mit einer zweistufigen Attentionsstrategie erreicht werden, dessen Aufwand wesentlich geringer ist:

- Ermittlung von k Aufmerksamkeitspunkten durch die Rückprojektion und
- Errechnung des lokalen Histogrammschnittes nur für diese Punkte.

Aus diesen Gründen ergeben sich durch die Bildung von lokalen Histogrammen, wie sie Ennesser und Medioni vorschlagen, für uns keine Vorteile gegenüber den ursprünglichen Verfahren von Swain und Ballard.

3.3 Farbkonstanz mit Farbdifferenzen

Funt und Finlayson[FF95] entwickeln ein Verfahren zur Farbkonstanz, welches sie der Histogramm-Intersektion voranstellen, um die bekannte Schwäche des Algorithmus' von Swain und Ballard zu beheben. Ihr Verfahren basiert auf der Theorie des (menschlichen) Sehens und verschiedener, darauf aufbauender Annahmen, die wir kurz behandeln wollen. Aus dem sog. *Koeffizienten-Modell von Sensorantworten* leiten Funt und Finlayson ab, daß die Verhältnisse von Farben benachbarter Punkte einer Oberfläche auch bei Beleuchtungsänderung konstant bleiben. Verglichen werden somit in der Intersektion Farbverhältnisse statt Farben.

3.3.1 Theoretischer Hintergrund

Der Algorithmus zur Identifikation eines unbekannten Objektes geht in der Online-Phase in vier Schritten vor:

1. Berechnung des natürlichen Logarithmus des Bildes,
2. isotrope Ableitung des logarithmierten Bildes,
3. Bildung des Histogrammes mit spezieller Quantisierung und
4. Berechnung des Histogrammschnittes mit sämtlichen Modellhistogrammen.

In der menschlichen Retina oder in CCD-Chips befinden sich optische Rezeptoren, die auf Licht verschiedener Wellenlänge unterschiedlich reagieren. Die Rezeptoren können grundsätzlich in 3 Klassen eingeteilt werden, die verstärkt auf rotes, grünes und blaues Licht antworten. Die Antwort eines Sensors der Klasse k wird im folgenden mit ρ_k, die spektrale Empfindlichkeit des Sensors als $S_k(\lambda)$ bezeichnet. Ist $E(\lambda)$ die spektrale Leistungsverteilung der beleuchtenden Strahlung, $R(\lambda)$ der spektrale Reflexionsfaktor der Oberfläche eines Körpers, ergibt sich das Farbsignal des von einem Körper reflektierten Lichts $C(\lambda) = E(\lambda)R(\lambda)$. Funt und Finlayson legen das folgende Sensor-Modell zugrunde

$$\rho_k = \int_{\lambda_1}^{\lambda_2} C(\lambda) S_k(\lambda)\, d\lambda \ , \tag{3.10}$$

wobei $[\lambda_1, \lambda_2]$ der sichtbare Bereich des elektromagnetischen Spektrums ist.
Unter der Annahme, daß für zwei Punkte x und y einer Oberfläche folgende Approximation gültig ist

$$\frac{\int R^x(\lambda)S_k(\lambda)\,d\lambda}{\int R^y(\lambda)S_k(\lambda)\,d\lambda} \approx \frac{\int E(\lambda)R^x(\lambda)S_k(\lambda)\,d\lambda}{\int E(\lambda)R^y(\lambda)S_k(\lambda)\,d\lambda}\,, \tag{3.11}$$

und da im diskreten Fall das Integral zur endlichen Summe wird, leiten sie daraus das 'Koeffizienten-Modell' ab

$$\rho_k = e_k r_k\,. \tag{3.12}$$

e_k und r_k sind hierbei Gewichtungsvektoren, die sich aus endlichen Approximationen der Beleuchtungsdichte und Reflektanz ergeben.
In diesem Koeffizienten-Modell, das nur unter verschiedenen Einschränkungen gilt, ist das Verhältnis der Sensorantworten zweier benachbarter Punkte x und y einer Oberfläche unter derselben Beleuchtung gleich dem Verhältnis der Oberflächenreflektanzen.

$$\frac{\rho_k^x}{\rho_k^y} = \frac{e_k r^x}{e_k r^y} = \frac{r^x}{r^y} \tag{3.13}$$

Nimmt man hiervon den Logarithmus, werden aus den Verhältnissen Differenzen:

$$\ln(\rho_k^x) - \ln(\rho_k^y) = \ln(r^x) - \ln(r^y)\,. \tag{3.14}$$

Somit ergibt sich für zwei benachbarte Pixel selbst bei ändernder Beleuchtung eine konstante Differenz, die mit einem Ableitungsoperator (Laplace, LoG, ...) berechnet werden kann. Da bei den Objekten, die Swain und Ballard aufgenommen haben und die auch weitgehend von Funt und Finlayson benutzt werden, grundsätzlich weniger Farbkanten als Farbplateaus vorkommen, nehmen die Autoren eine spezielle Quantisierung bei der Bildung der Histogramme vor, die sie nach der Wahrscheinlichkeitsverteilung der auftretenden Differenzen berechnen.
Als letzter Schritt folgt die Intersektion, wobei der Intersektionswert nicht wie in (2.8) auf Seite 14 normiert wird. Diese Normierung, so Funt und Finlayson, würde zu sehr auf das Vergleichen von Farboberflächen abstellen. Da Punkte verschiedenfarbiger Oberflächen nach dem Farbkonstanzalgorithmus grundsätzlich Werte nahe 0 haben, würden Farbkanten nicht genug gewürdigt. Deshalb normieren sie statt dessen durch das maximale Bin des Objetes und des Modells.

3.3.2 Bewertung

Funt und Finlayson nehmen bei der Berechnung der Quantisierungsabstände an, die Farben der Modelle in der Datenbank seien gleichverteilt im Histogrammraum. Abbildung 3.1 zeigt die Farbverteilung der Modelle, welche wir in unseren Experimenten verwenden werden; eine gleichmäßige Verteilung liegt nicht vor und wird i.a. auch bei anderen Datenbanken nicht der Fall sein. Eine differenziertere Betrachtung je nach Farbverteilung scheint angeraten.
Wie sich in unseren Experimenten zeigen wird, sind die Resultate des Verfahrens in einigen Beleuchtungsumgebungen nicht sehr gut. Dies steht in einem krassen Gegensatz zu den Resultaten, mit welchen Funt und Finlayson die Farbkonstanz ihres

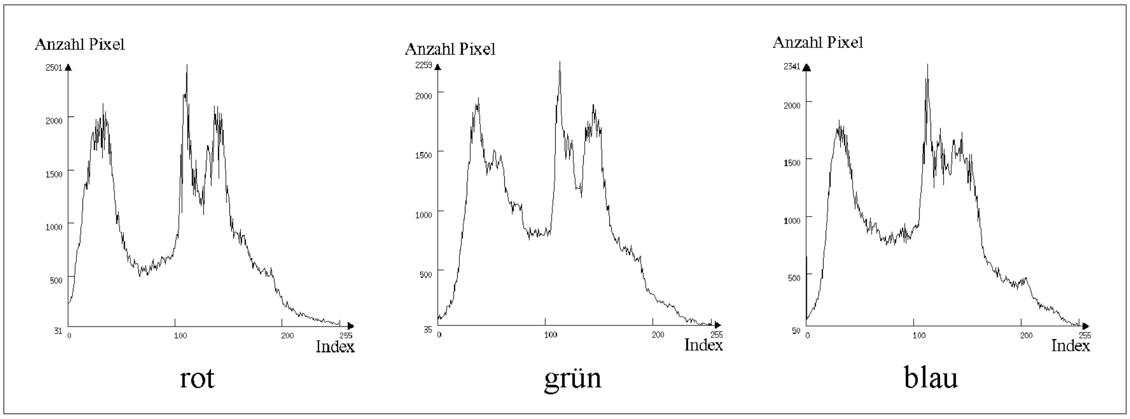


Abbildung 3.1: Bimodale Farbverteilung der Modelle unserer Datenbank auf den RGB-Kanälen, welche nicht mit der Annahme einer Gleichverteilung der Farben von Funt und Finlayson übereinstimmt. Der maximale Peak in der Mitte des jeweiligen Farbindexes stellt die Farbe des Hintergrundes dar, die nur bei wenigen Modellen vollständig entfernt ist.

Algorithmus' belegen. Die Diskrepanz läßt sich an den Testobjekten erklären: die Autoren konstruieren sich 60 künstliche Farbobjekte unter verschiedenen Farbtemperaturen (sog. 'mondrians'), für welche die benutzten Ableitungsfilter ideal sind, und erreichen damit eine Trefferquote von 1 (!), d.h. alle künstlichen Objekte werden unter sämtlichen Farbtemperaturen korrekt identifiziert. Darüber hinaus könnte in den Aufnahmen einiger unserer farbig-beleuchteten Objekte eine Schwachstelle liegen. Wir benutzen neben der Deckenbeleuchtung in einem Aufnahmeraum 2 Scheinwerfer zur gleichmäßigen Ausleuchtung der Szenen und Modelle. Dabei strahlen diese Scheinwerfer grundsätzlich nicht die Objekte direkt an, sondern lieferen indirektes Licht. Für die farbig-beleuchteten Aufnahmen befestigen wir je zwei rote, grüne oder blaue Plastikfolien vor den Scheinwerfern. Um mit den Filtern aber Wirkungen zu erzielen, müssen die Objekte direkt angestrahlt werden. Dabei kommt es bei einigen Objekten auf dem jeweiligen Farbkanal zur Sättigung, die wir auch nach mehrmaligen Anpassungsversuchen nicht ganz verhindern können. Abbildung 3.2 zeigt die Auswirkung der farbigen Beleuchtung auf den G-Kanal des Modells *7Up*. Gesättigte Pixel, die vorher evtl. unterschiedliche Farbwerte besaßen, haben den gleichen maximalen Farbwert. Liegen derartige Pixel im Einzugsbereich des Ableitungsfilters, ergibt sich dort ein konstanter Wert von 0. Funt und Finlayson sehen diese Anfälligkeit ihres Algorithmus ebenfalls und entfernen deshalb aus der Datenbank von Swain und Ballard 11 Objekte, in denen Sättigungen vorkommen. Die Modelle unserer Datenbank sind zwar frei von Sättigung, nicht jedoch die farblich beleuchteten Objekte. Diese Objekte aus den Experimenten auszuschließen würde zur Folge haben, daß wir keine Ergebnisse über die Farbkonstanz vorweisen könnten. Eine praktische Farbkonstanz schließt eine gewisse Toleranz des Verfahrens gegenüber gesättigten Farbkanälen ein.

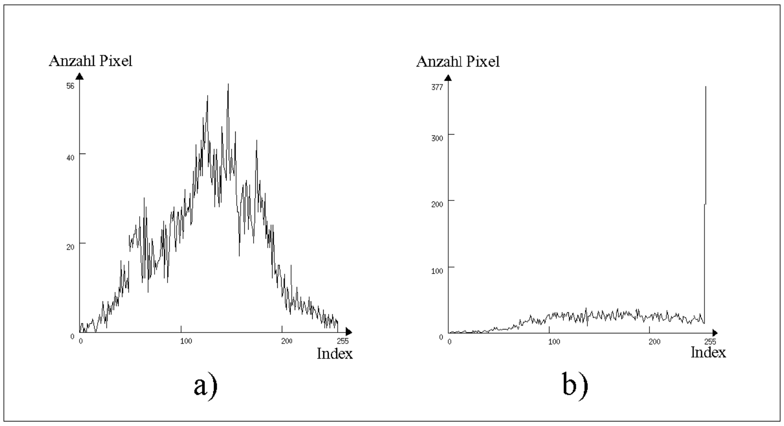


Abbildung 3.2: Histogramme des G-Kanals des Modells *7Up* zur Verdeutlichung des Problems der Sättigung, mit welcher der Algorithmus von Funt und Finlayson nicht gut zurechtkommt. a) normale Beleuchtung; b) grüne Beleuchtung.

3.3.3 Schlußfolgerungen

Der Algorithmus von Funt und Finlayson unterscheidet sich von der Histogramm-Intersektion in einer Vorverarbeitung des Eingangsbildes durch ein Ableitungsfilter und einer besonderen Quantisierung bei der Bildung der Histogramme auf der Filterantwort. Durch die Verwendung eines isotropen Filters bleibt die Rotationsinvarianz erhalten. Da in einem Bild grundsätzlich weniger Farbkanten als Farboberflächen vorhanden sind, findet trotz der speziellen Quantisierung ein Vergleich der Oberflächen statt, welche nach der Filterung jedoch alle in einem Wertebereich nahe 0 liegen. Objekte werden somit nicht durch das Hauptmerkmal der Histogramme (maximale Einträge) sondern durch die verbleibenden Bins unterschieden. Somit sinkt auch die Anzahl der Objekte, welche mit diesem Verfahren unterschieden werden können. Die Intersektion ist ein Verfahren zum Vergleichen von farbigen Oberflächen, welches nicht ohne weiteres in ein Vergleichen von Farbkanten gewandelt werden kann.
Die praktische Farbkonstanz der Methode bleibt zu bezweifeln, da eine starke Anfälligkeit gegenüber gesättigten Farbkanälen besteht.

3.4 Ähnlichkeit durch Momente

Der Histogrammschnitt hat nach Stricker und Orengo[SO95] einige Nachteile, die hier kurz aufgeführt werden:

- Die Suche nach den optimalen Parametern für Bildung der Histogramme ist oft schwierig.
- Durch die Quantisierung fällt eine gewisse Anzahl von Farben in ein einziges Bin des Histogrammes.

- Histogramme mit Verteilungen, die lediglich auf der X-Achse verschoben sind, werden durch den Histogrammschnitt nicht als ähnlich erkannt.
- Bei großen Datenbanken liefert die Intersektion viele positive Antworten für falsche Modelle.

Aufgrund dieser Nachteile ändern Stricker und Orengo die Methode des Histogrammschnittes dahingehend ab, daß statt Farbhistogrammen andere statistische Eigenschaften der Bilder verglichen werden. Anhand dieser Eigenschaften definieren sie ein Abstandsmaß für Bilder, mit dem sie auf Ähnlichkeit schließen.

3.4.1 Theoretischer Hintergrund

Der Farbwert eines Pixels und somit die Farbverteilung eines Bildes kann als Zufallsvariable aufgefaßt werden. Zufallsvariablen lassen sich durch ihre Momente charakterisieren. Stricker et. al. benutzen den Erwartungswert sowie das 2. und 3. zentrale Moment der einzelnen Farbkanäle eines Bildes. Sei i der Index des Farbkanals und p_{ij} der Wert des Pixels j des i-ten Kanals. Dann ergeben sich die Momente zu

$$E_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij} \quad , \quad \sigma_i = \Big(\frac{1}{N}\sum_{j=1}^{N}(p_{ij}-E_i)^2\Big)^{\frac{1}{2}} \quad , \quad s_i = \Big(\frac{1}{N}\sum_{j=1}^{N}(p_{ij}-E_i)^3\Big)^{\frac{1}{3}} . \tag{3.15}$$

Somit wird ein 3-kanaliges Bild durch 9 Werte repräsentiert. Mit den Momenten definieren Stricker et. al. eine gewichtete Ähnlichkeitsfunktion

$$d_{Mom}(O,M) = \sum_{i=1}^{r} \omega_{i1}|E_i^O - E_i^M| + \omega_{i2}|\sigma_i^O - \sigma_i^M| + \omega_{i3}|s_i^O - s_i^M| \, , \tag{3.16}$$

wobei E^O, σ^O und s^O die Momente des Objektes und E^M, σ^M und s^M die Momente des Modells darstellen. Die Gewichte ω_{ij} können dazu benutzt werden, die Funktion an bestimmte Gegebenheiten anzupassen. Werden z.B. Bilder verglichen, die nicht unter den gleichen Lichtverhältnissen aufgenommen sind, ist der Farbmittelwert keine zuverlässige Information; die jeweiligen Gewichte sollten auf kleine Werte gesetzt werden. Benutzt man einen Farbraum, in dem ein Kanal eine stärkere Bedeutung hat als ein anderer (z.B. der Luminanzkanal der Räume *Lab* oder *Luv*), sollte dessen Gewicht auch in der Ähnlichkeitsfunktion größer sein.

3.4.2 Bewertung

3.4.2.1 Komplexität

Wir vergleichen die Komplexität der Ähnlichkeitsfunktion mit der des Histogramm-Schnittes in der Online-Phase für eine Anzahl bis zu 2000 Modellen. Ausgegangen wird von quadratischen RGB-Bildern mit den Abmessungen $x \times y = 251 \times 168$ Bildpunkten. Ferner nehmen wir der Einfachheit halber an, daß alle mathematischen Operationen die gleiche Rechenzeit beanspruchen.

Zunächst muß der Erwartungswert des Bildes in $3xy$ Schritten berechnet werden. Danach folgen die Berechnungen des 2. und 3. Momentes mit $3[6(xy + 1)]$ Operationen. Das Vergleichen mit den Momenten von m gespeicherten Modellen einer Datenbank kostet $24m$ Schritte.
Um 2000 Modelle in einer Histogrammdatenbank zu repräsentieren, wählen wir eine Auflösung von 1000 Bins für jedes Histogramm. Die Kapazität dieses Histogrammraumes schätzen wir vorsichtig mit einem sehr hohen Schwellwert $t = 0.8$ ab: als untere Grenze ergibt sich ein Wert von 8031. Da für kleine Datenbanken 1000 Bins eine zu feine Quantisierung darstellt, wählen wir für die kleinste Datenbank (1 Modellhistogramm) 512 Bins und erhöhen die Anzahl linear mit steigender Anzahl der Modelle bis 1000 Bins bei 2000 Modellen.
Für die Intersektion ist zunächst das Histogramm des unbekannten Objektes zu ermitteln, was xy Berechnungen kostet. Danach muß das Histogramm mit m Histogrammen aus einer Datenbank geschnitten werden. Dies benötigt mn_{var} Schritte, wobei n_{var} die linear steigende Anzahl der Bins ist.
In Abbildung 3.3 sind beide Kostenfunktionen eingezeichnet. Die Methode von Stricker und Orengo ist schneller – allerdings erst ab einer Anzahl von 1141 Modellhistogrammen in der Datenbank.

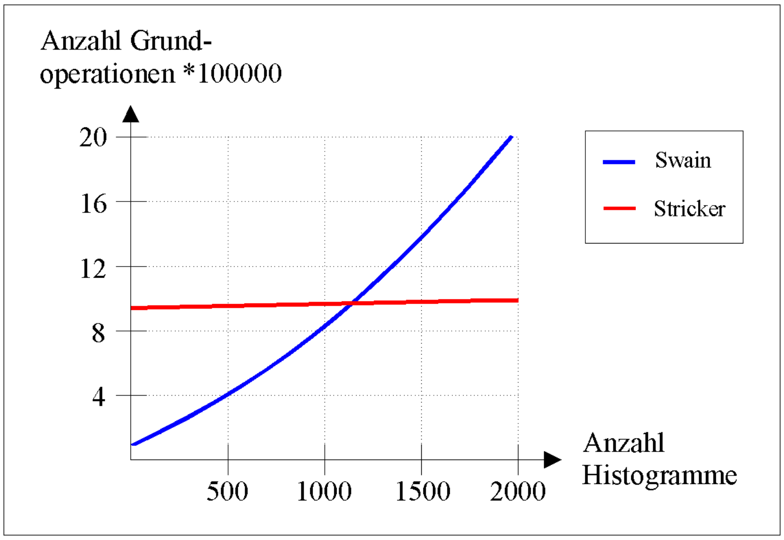


Abbildung 3.3: Vergleich der Anzahl der Operationen der Ähnlichkeitsfunktion und des Histogrammschnittes.

3.4.2.2 Einsetzbarkeit des Verfahrens

Die Methode von Stricker vergleicht Bilder aufgrund globaler statistischer Werte. Dies hat Auswirkungen auf die Einsetzbarkeit der Verfahrens bezogen auf evtl. variierende Größen der zu bewertenden Bilder oder Bildausschnitte.
Bei der Intersektion können Modelle per Hand segmentiert werden, um deren Histogramme in einer Datenbank zu speichern. Objekte können in unbekannten Szenen

identifiziert werden, ohne daß eine vorherige Segmentierung Hintergrund/Objekt notwendig ist. In der Online-Phase werden Hintergrundfarben, die nicht in dem Objekt vorkommen, ignoriert. Demzufolge ist für den bewerteten Bildausschnitt lediglich Voraussetzung, daß das unbekannte Objekt darin Platz findet. Die Größe des Ausschnittes kann ansonsten frei variiert werden. Die Ähnlichkeitsfunktion bezieht jedoch sämtliche Bildpunkte in die Berechnung ein. Somit können verschiedene Bildgrößen allein durch die Änderung der Anzahl der Hintergrundpixel verschiedene Ähnlichkeiten ergeben, obwohl an dem eigentlich interessierten Objekt keine Änderung vorgenommen wird. Das gleiche Problem stellt sich, wenn in der Online-Phase ein Hintergrund vorliegt, welcher unterschiedlich ist zu dem Hintergrund während der Berechnung der Modell-Momente. Eine Segmentierung der Modelle vor Berechnung ihrer Momente bietet keine Lösung, da die Ähnlichkeitsfunktion im Gegensatz zur Intersektion Hintergrundfarben nicht ignoriert.
Das Verfahren eignet sich somit grundsätzlich besser für den Vergleich von Objekten oder Szenen, die einen Großteil des Bildes einnehmen, wie es in Strickers Experimenten der Fall war. Nehmen Objekte jedoch nur einen kleinen Teil des bewerteten Bildausschnittes ein, ergibt sich eine starke Dominanz des Hintergrundes in der Berechnung der Momente.

3.4.3 Schlußfolgerungen

Das Verfahren hat für größere Datenbanken eine geringere Komplexität als die Intersektion. Ebenso wie der Histogrammschnitt ist es rotationsinvariant und robust gegenüber Schwankungen des Abstandes vom Objekt zur Kamera. Für einen Online-Einsatz ist der Algorithmus nicht geeignet, da nicht immer garantiert werden kann, daß die aktuellen Bildgrößen und diejenigen, unter denen die Momentdatenbank gebildet worden ist, übereinstimmen. Da sämtliche Bildpunkte in die Berechnung der Momente des unbekannten Objektes einbezogen werden, ist die Anzahl der Bildpunkte ein wesentlicher Faktor. Berechnet man einerseits die Momente für einen Bildausschnitt, der von dem unbekannten Objekt eingenommen wird, und andererseits einen vergrößerten Ausschnitt, in dem sich nach wie vor das Objekt befindet, werden die Momente nachhaltig durch den Hintergrund beeinflußt. Genau entgegengesetzt verhält es sich bei der Intersektion, bei der nur Punkte in den Schnitt einbezogen werden, die einen Farbwert des Modells besitzen. Wegen dieser Eigenschaft kann eine Segmentierung des unbekannten Objektes bei der Intersektion entfallen, bei Strickers Algorithmus scheint dies wieder notwendig zu werden.

Kapitel 4

Steuerbare Filter

In unserem Verfahren zur Farbkonstanz benutzen wir steuerbare Filter. Bevor wir dieses Verfahren in Kapitel 5 beschreiben, gehen wir nun ausführlich auf steuerbare Filter im allgemeinen und das Andersson-Filter im besonderen ein.

4.1 Theorie der steuerbaren Filter

Orientierte Filter finden in vielen Aufgaben der Computer-Vision ihre Anwendung, so z.B. in der Kantendetektion, der Texturanalyse, der Segmentierung eines Bildes in verschiedene Bereiche o.a. Oftmals werden in einem Verfahren orientierte Filter benötigt, deren Filterantworten sich lediglich darin unterscheiden, daß sie um einen best. Winkel rotiert sind. Ist die Anzahl dieser benötigten Filter, deren jeweilige Orientierung sich zur Laufzeit des Algorithmus' nicht ändert, fest, können diese offline berechnet und zum gewünschten Zeitpunkt eingesetzt werden. Verfahren jedoch, die eine adaptive Kontrolle der Orientierung einzelner Filter beinhalten, verlangen nach einer effizienteren Lösung, als alle möglichen Orientierungen des oder der Filter im Voraus zu bestimmen.
Eine Lösung für dieses Problem ist der Einsatz von steuerbaren Filtern. Durch eine Linearkombination von sog. Basisfiltern läßt sich jede gewünschte Orientierung eines steuerbaren Filters berechnen. Da wir in unserer Arbeit 2D-Andersson-Filter benutzen, beschränken wir uns auf 2D-Filter und auf Steuerung der Orientierung, obwohl die Theorie auch für Filter höherer Dimension und beliebiger Deformation (z.B. Skala) herzuleiten ist.

4.1.1 Voraussetzungen für die Steuerbarkeit

Ein 2D-Filter f ist eine Funktion, die von zwei Variablen abhängt. Wir benutzen in den folgenden Abschnitten sowohl kartesische Koordinaten (x_1, x_2) als auch Polarkoordinaten (r, φ), die bekanntermaßen zusammenhängen durch die Gleichungen

$$r = \sqrt{x_1^2 + x_2^2} \quad \text{und} \quad \varphi = \arg(x_1, x_2) \; . \tag{4.1}$$

Mit f^θ bezeichnen wir im Folgenden eine Funktion, die um den Winkel θ rotiert ist, also $f^\theta(r,\varphi) = f(r,\varphi-\theta)$. Eine Funktion $f(r,\varphi)$ ist steuerbar nach Freeman, falls es Funktionen $k_j(\theta) \mid [0,2\pi] \rightarrow \mathbb{R}$ gibt mit

$$f^\theta(r,\varphi) = \sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(r,\varphi) \quad , \tag{4.2}$$

d.h. jede rotierte Version von f kann als eine Linearkombination von rotierten Versionen ihrer selbst geschrieben werden. Es gilt somit zu bestimmen, ob eine Funktion überhaupt steuerbar ist, wenn dies der Fall ist, welche und wie viele Basisfunktionen benötigt werden und welches die korrekten Koeffizienten sind. Diese Koeffizienten $k_j(\theta)$ werden Interpolationsfunktionen genannt, da durch sie ein Filter beliebiger Orientierung interpoliert wird.
Sind diese Probleme gelöst, kann die Filterantwort eines beliebig orientierten steuerbaren Filters auf ein Bild I wegen der Linearität der Faltung folgendermaßen berechnet werden:

$$\begin{aligned} f^\theta(r,\varphi) * I &= \left(\sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(r,\varphi)\right) * I \\ &= \sum_{j=1}^{M} \Big(k_j(\theta) f^{\theta_j}(r,\varphi)\Big) * I \\ &= \sum_{j=1}^{M} \Big(f^{\theta_j}(r,\varphi) * I\Big)\, k_j(\theta) \, . \end{aligned} \tag{4.3}$$

Voraussetzung ist für diesen Fall, daß das Filter im Ortsraum definiert ist. Wie wir später sehen werden, ist die Steuerbarkeit eines Filters im Ortsraum äquivalent mit der Steuerbarkeit des fouriertransformierten Filters im Frequenzraum und somit (4.3) wegen des Faltungssatzes der Fouriertransformation analog auf Filter im Frequenzraum anwendbar.
Abbildung 4.1 veranschaulicht den Vorgang der Steuerung der Orientierung graphisch: ein gegebenes Bild wird nur einmal mit jedem Basisfilter gefaltet, die Ausrichtung wird durch Multiplikation mit den Interpolationsfunktionen vorgenommen, die interpolierten Filterantworten zum Schluß addiert.

4.1.2 Klassifikation steuerbarer Filter

Freeman und Adelson beweisen, daß nur Funktionen $f(r,\varphi)$, die sich durch eine Partialsumme ihrer Fourierreihe (entwickelt nach der Variable φ) schreiben oder hinreichend approximieren lassen

$$f(r,\varphi) = \sum_{n=-N}^{N} c_n(r) e^{in\varphi} \quad , \tag{4.4}$$

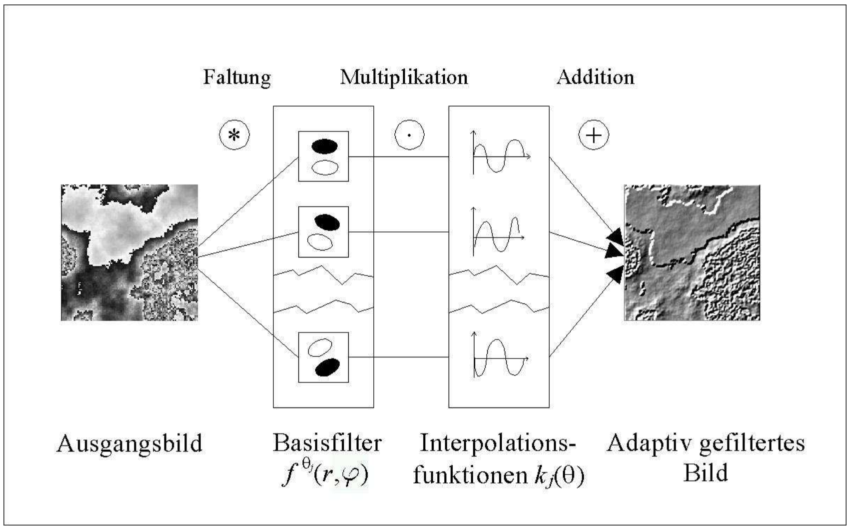


Abbildung 4.1: Adaptive Filterung mit dem Basissatz eines steuerbaren Filters. Die Filterantworten des Basissatzes werden durch eine gewichtete Addition mit den Interpolationsfunktionen als Gewichte in eine gewählte Orientierung ausgerichtet.

steuerbar sind. Diesen Beweis werden wir im nächsten Abschnitt nachvollziehen.
Aus (4.4) ergibt sich unmittelbar die Anzahl der benötigten Basisfilter: ist T die Anzahl der von 0 verschiedenen Koeffizienten $c_n(r)$, so werden mindestens T Basisfilter zur Steuerung benötigt.
Damit ist die Klasse der steuerbaren Filter erheblich eingeschränkt. Bekannte Filter dieser Klasse sind z.B. alle Ableitungen der Gauss-Funktion oder 2-dimensionale Polynome, die mit einer radialsymmetrischen Fensterfunktion multipliziert sind. Bezeichnen wir mit

$$\mathcal{F} := \{\mathrm{f} : \mathbb{R} \to \mathbb{C} \mid \mathrm{f}\ 2\pi\text{-periodisch}\} \tag{4.5}$$

die Menge aller 2π-periodischen Funktionen von $\mathbb{R}$ nach $\mathbb{C}$, besitzt zwar jedes $f \in \mathcal{F}$ komplexe Fourierkoeffizienten $c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)e^{-ikt}\,dt$ und damit auch eine Fourierreihe $\sum_{-\infty}^{\infty} c_k e^{ikt}$, ob jedoch allgemein gilt $f(t) = \sum_{-\infty}^{\infty} c_k e^{ikt}$ oder ab welchem N die Partialsumme $\sum_{-N}^{N} c_k e^{ikt}$ die Funktion hinreichend approximiert, sind z.T. noch nicht gelöste Probleme. Von exakter Steuerbarkeit spricht man bei Funktionen mit endlicher Partialsumme, von approximativer Steuerbarkeit hingegen, wenn die Partialsumme lediglich approximiert wird.
Für eine Funktion $f(x,y)$ und ihre Fouriertransformierte $F(u,v)$ gilt

$$f^{\theta}(x,y) \circ\!\!-\!\!\bullet\ F^{\theta}(u,v)\ , \tag{4.6}$$

d.h., eine Rotation um den Winkel θ im Ortsraum bewirkt ebenfalls eine Rotation um θ im Frequenzraum. Die ausführliche Herleitung von (4.6) ist in Anhang A zu finden. Durch diese Eigenschaft erhält man zusammen mit dem Additionstheorem der Fouriertransformation die Korrespondenzen

$$\begin{array}{ccc} f^\theta(r,\varphi) & = & \sum\limits_{j=1}^{M} k_j(\theta) f^{\theta_j}(r,\varphi) \\ \circ\!\!-\!\!\bullet & & \circ\!\!-\!\!\bullet \\ F^\theta(\rho,\vartheta) & = & \sum\limits_{j=1}^{M} k_j(\theta) F^{\theta_j}(\rho,\vartheta) \end{array} . \tag{4.7}$$

Somit ist F genau dann steuerbar, wenn f steuerbar ist.

4.1.3 Berechnung der Interpolationsfunktionen

Sei f eine Funktion, die sich wie in (4.4) in eine Partialsumme entwickeln läßt. Eine derartige Funktion ist steuerbar nach Freeman und Adelsondie gleichzeitig ein Verfahren zur Berechnung der Koeffizienten $k_j(\theta)$ angeben. Dazu wird zunächst der Ausdruck aus (4.4) in beide Seiten von (4.2) substituiert:

$$\left(\sum_{n=-N}^{N} c_n(r) e^{in\varphi}\right)^{\theta} = \sum_{j=1}^{M} k_j(\theta) \left(\sum_{n=-N}^{N} c_n(r) e^{in\varphi_j}\right)^{\theta_j} . \tag{4.8}$$

In dem unendlich-dimensionalen Vektorraum $\mathcal{F}$ aus (4.5) wird für je zwei Funktionen $x, y \in \mathcal{F}$ durch folgende Abbildungsvorschrift ein Skalarprodukt definiert:

$$x \cdot y := \frac{1}{2\pi} \int_{-\pi}^{\pi} x(t) y(t)^* \, dt . \tag{4.9}$$

Für alle Funktionen $x, y \in \mathcal{F}$ mit $x(\varphi) = e^{im\varphi}, y(\varphi) = e^{in\varphi}$ und $m \neq n$ gilt die sog. Orthogonalitätsrelation $x \cdot y = 0$, d.h. je zwei Funktionen $e^{im\varphi}$ und $e^{in\phi}$ mit $m \neq n$ sind orthogonal zueinander, die Folge $\left(e^{in\varphi}\right)_{n \in \mathbb{Z}}$ ist also eine Basis von $\mathcal{F}$. Wir können somit (4.8) für alle $m \in \{-N, \ldots, N\}$ auf die Funktion $e^{im\theta}$ projizieren und erhalten ein lineares Gleichungssystem

$$c_m(r) e^{im\theta} = \sum_{j=1}^{M} k_j(\theta) c_m(r) e^{im\varphi_j} \quad , \ m \in \{-N, \ldots, N\} . \tag{4.10}$$

Falls $c_m(r) = 0$ für ein $m \in \{-N, \ldots, N\}$, kann diese Gleichung aus dem System entfernt werden, sonst wird jede Gleichung durch $c_m(r)$ dividiert und es ergib sich:

$$\begin{pmatrix} e^{-iN\theta} \\ \vdots \\ e^{-i\theta} \\ 1 \\ e^{i\theta} \\ \vdots \\ e^{iN\theta} \end{pmatrix} = \begin{pmatrix} e^{-iN\varphi_1} & e^{-iN\varphi_2} & \cdots & e^{-iN\varphi_M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i\varphi_1} & e^{-i\varphi_2} & \cdots & e^{-i\varphi_M} \\ 1 & 1 & \cdots & 1 \\ e^{i\varphi_1} & e^{i\varphi_2} & \cdots & e^{i\varphi_M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{iN\varphi_1} & e^{iN\varphi_2} & \cdots & e^{iN\varphi_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ k_2(\theta) \\ \vdots \\ k_M(\theta) \end{pmatrix} . \tag{4.11}$$

Betrachtet man eine Zeile des Gleichungssystems mit dem Index m und nimmt davon den konjugiert komplexen Wert, erhält man

$$\begin{aligned} \left(e^{im\theta}\right)^* &= \left(\sum_{j=1}^{M} k_j(\theta)e^{im\varphi_j}\right)^* \text{, also} \\ e^{-im\theta} &= \sum_{j=1}^{M} \left(k_j(\theta)e^{im\varphi_j}\right)^* \\ &= \sum_{j=1}^{M} k_j(\theta)e^{-im\varphi_j} \; . \end{aligned} \tag{4.12}$$

Daraus resultieren die Voraussetzungen, welche die Zeile mit dem Index $-m$ liefert. Aus diesem Grund können die Gleichungen für $m < 0$ außer Acht gelassen und folgendes gekürztes Gleichungssystem betrachtet werden

$$\begin{pmatrix} 1 \\ e^{i\theta} \\ \vdots \\ e^{iN\theta} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ e^{i\varphi_1} & e^{i\varphi_2} & \cdots & e^{i\varphi_M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{iN\varphi_1} & e^{iN\varphi_2} & \cdots & e^{iN\varphi_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ k_2(\theta) \\ \vdots \\ k_M(\theta) \end{pmatrix} , \tag{4.13}$$

aus dem sich die Interpolationsfunktionen bestimmen lassen. Interessanterweise ergeben sich die Interpolationsfunktionen unabhängig von den Fourierkoeffizienten $c_m(r)$. Dies kann so interpretiert werden, daß sich ein steuerbares Filter in einen angularen und einen radialen Anteil aufspalten läßt; für das reine Ausrichten des Filters auf einen best. Winkel sind die angularen Anteile der Basisfilter nötig.

4.2 Das Andersson-Filter

M. T. Andersson[And92] entwickelt einen Basissatz von steuerbaren Filtern, den sog. Andersson-Filtern. Die Filter sind nach den Gesichtspunkten der

- Steuerbarkeit in Orientierung und
- Steuerbarkeit in Skala

entworfen. Aus letzterem Grund definiert Andersson die Filter im Fourierraum, weil sich dieses Problem dort in bekannten Begriffen wie Bandbreite, Mittelfrequenz, Amplitude u.a. wiederspiegelt. In den folgenden Abschnitten werden wir auf die Filterdefinition, wichtige Eigenschaften sowie der Transformation der Filter in den Ortsraum eingehen.

4.2.1 Definition

Um Orientierung und Skalierung getrennt behandeln zu können, ist ein Andersson-Filter polar separabel gestaltet. Seine Übertragungsfunktion kann in einen radialen und einen angularen Teil mit Radialfrequenz ρ und Angularfrequenz ϑ aufgespalten werden

$$H(\rho, \vartheta) = Q(\rho) \cdot A(\vartheta) \; , \tag{4.14}$$

wobei $Q(\rho)$ die Eigenschaften eines Bandpasses hat.
Ein Andersson-Filter ist variabel bzgl. der Parameter

- Mittelfrequenz q, die umgekehrt proportional zur Skalierung σ ist,
- logarithmischer Radialbandbreite B und
- Orientierung θ.

Da sich ein Andersson-Filter aus einem geraden und einen ungeraden Anteil, die zusammen ein Hilbertpaar bilden, zusammensetzt, muß bei dem angularen Teil zusätzlich zwischen gerade und ungerade unterschieden werden, was durch den Index $\alpha \in \{\textit{gerade, ungerade}\}$ angedeutet wird. Bei dem radialen Filteranteil kann dies entfallen, weil die Amplitudenfunktionen eines Hilbertpaares identisch sind.
(4.14) kann somit präzisier formuliert werden durch

$$H_{\sigma,\theta,B,\alpha}(\rho, \vartheta) = Q_{\sigma,B}(\rho) \cdot A_{\theta,\alpha}(\vartheta) \; . \tag{4.15}$$

Der radiale und angulare Filteranteil sind dann definiert als

$$Q_{\sigma,B}(\rho) = \begin{cases} \cos^2 \dfrac{\pi \ln(\sigma\rho)}{2B \ln 2} & \text{, falls } 2^{-B}\sigma^{-1} \le \rho < 2^{B}\sigma^{-1} \\ 0 & \text{, sonst} \end{cases} \quad \text{und} \tag{4.16}$$

$$A_{\theta,\alpha}(\vartheta) = \begin{cases} \cos^2(\vartheta - \theta) & \text{, falls } \alpha = \text{gerade} \\ j\cos^2(\vartheta - \theta)\,\mathrm{sign}(\cos(\vartheta - \theta)) & \text{, sonst} \end{cases} \; . \tag{4.17}$$

4.2.2 Steuerbarkeit

Für die Steurerbarkeit der Orientierung genügt es, lediglich den angularen Teil des Filters zu betrachten. Aus der Formel $\cos^2(\vartheta) = \frac{1}{4}e^{j2\vartheta} + \frac{1}{4}e^{-j2\vartheta} + \frac{1}{2}e^{0}$ ergeben sich sofort die Koeffizienten c_{-2}, c_0 und c_2 aus (4.4), die restlichen Koeffizienten sind gleich 0.
Der ungerade Anteil hat nicht diese einfache Darstellung der Partialsumme. Es gilt $c_k = 0$ für alle geraden $k \in \mathbb{Z}$ und

$$c_k = \frac{1}{\pi} \cdot \begin{cases} \dfrac{4}{k^3 - 4k} & \text{, falls} \quad k = 2n - 1,\ n \text{ gerade} \\ -\dfrac{4}{k^3 - 4k} & \text{, falls} \quad k = 2n - 1,\ n \text{ ungerade} \end{cases} \tag{4.18}$$

für alle ungeraden $k \in \mathbb{Z}$. Somit hat der ungerade Andersson-Filter unendlich viele Fourierkoeffizienten und es gilt zu bestimmen, welche Approximation für das Filter ausreichend ist. Dazu schätzen wir den absoluten Fehler ab mit der Formel

$$\Delta_u = \int_{-\pi}^{\pi} \left(A_{\theta,\alpha}(\vartheta) - \sum_{n=-N}^{N} c_n(r) e^{in\vartheta} \right)^2 d\vartheta \,, \quad N \in \mathbb{N} \text{ und } \alpha = \mathit{ungerade}\,. \tag{4.19}$$

Bedingt durch die Potenzen im Nenner der Koeffizienten sinkt der Fehler sehr schnell: für $N = 3$ beträgt er 0.00212, für $N = 5$ bereits 0.00027. Somit wählen wir die vier Koeffizienten a_{-3}, a_{-1}, a_1 und a_3, welche eine ausreichende Approximation der unendlichen Summe aus (4.4) darstellen.

4.2.3 Wahl der Parameter

Wir beabsichtigen, lokale Orientierungen mittels der Andersson-Filter zu berechnen. Dazu benutzen wir den analytischen Filter. Es stellt sich die Frage, wie die Parameter gewählt werden sollten, um unserer Aufgabenstellung gerecht zu werden.
Aus der Anzahl der Basisfilter (vier) bestimmen sich zunächst die Radialanteile der Filter: die Orientierungen θ_i werden äquidistant verteilt im Intervall $[0, \pi]$, so daß $\theta_i = \frac{n\pi}{4}, n \in \{0, \ldots, 3\}$, um mögliche Interferenzen der einzelnen Filter gering zu halten und alle Richtungen vom Filtersatz gleichgewichtig zu erfassen.
Die Wahl der Skala bzw. Skalen fällt etwas komplexer aus. Hierzu gibt es verschiedene Ansätze, von denen wir kurz zwei erläutern wollen:

- Andersson[And92] schlägt vor, bei dem Eingangsbild pyramidenartig in mehreren Schritten eine Unterabtastung vorzunehmen, um auf jeder Abtastung den gleichen Basisfiltersatz anzuwenden. Vor jeder Unterabtastung wird das Bild mit einem Tiefpaßfilter geglättet, um Aliasing zu vermeiden. Die Filterantworten sind anschließend auf die ursprüngliche Bildgröße zu interpolieren.

- Mahlmeister et. al.[MSS95] benutzen Basisfilter auf drei verschiedenen Skalen σ_i, wobei die Bandbreite proportional zur Mittelfrequenz gewählt wird. Aufgrund von mehrdimensionalen Merkmalskarten wird dann nach bestimmten Eigenschaften von Objekten gesucht.

Beide Ansätze untersuchen Ereignisse in mehreren Auflösungen. Wir dürfen dabei die angestrebte Echtzeitimplementierung nicht außer Acht lassen, welche uns gewisse Grenzen setzt. Betrachten wir z.B. die letztgenannte Vorgehensweise, wären bei einem 3-kanaligen Bild die Filterantworten jedes Kanals mit 12 Basisfiltern, also 36, zu bestimmen. Legt man eine Filtergröße von 16×16 zugrunde, bedeutet dies für unsere Implementierung, daß 144 Faltungen durchgeführt werden müßten, da auf der MV200 eine 16×16-Maske aus vier 8×8-Masken zusammengesetzt wird. Die Faltung eines 251×170-Bildes dauert auf der MV200 ca. $0.005\,s$. Dies ergäbe selbst bei vier parallelen Faltungen eine Zeit von $0.018\,s$, wobei die Faltungsergebnisse

noch nicht weiterverarbeitet wären. Bei der Vielzahl der auszuführenden Datenflüsse schätzen wir für derartige Berechnungen wenigstens $1\,s$. Damit ergäbe sich allein für die Vorverarbeitung des Eingangsbildes, auf das noch die Rückprojektion und die Intersektion angewendet werden sollen, eine Laufzeit von ca. $0.3\,s$. Eine Rate von 3 Bildern pro Sekunde erscheint uns im Hinblick auf 'Echtzeit' zu gering.
Aus diesen Gründen haben wir uns entschieden, einen Satz von Basisfiltern nur in einer Skala auf das Eingangsbild anzuwenden. Dieses Vorgehen läßt sich ferner stützen durch die Algorithmen, die wir anwenden wollen: beide sind nicht skalierungsunabhängig sondern lediglich 'robust' gegenüber Änderungen in diesem Parameter. Für eine genaue Lokalisierung wird u.a. die Entfernung des Objektes zur Kamera benötigt. Weicht diese Annahme stärker von der tatsächlichen Entfernung ab, ist auch die Größe der Glättungsmaske falsch, was sich in einer ungenauen Lokalisierung bemerkbar macht (siehe Abschnitt 2.1.2.2). Die Intersektion kann nicht oder nur eingeschränkt verschieden große Objekte identifizieren, bedingt durch den Normierungsfaktor der Modellhistogramme (siehe Abschnitt 2.2.2.4). Somit sind die gleichbleibende Entfernung der Objekte zur Kamera sowie Auflösung des Bildes wichtige Voraussetzungen, bei deren Nichteinhaltung die Algorithmen nichtdeterministische Ergebnisse liefern; unsere Objekte müssen in einer Skala bleiben. Es stellt sich nicht die Notwendigkeit, ein ganz bestimmtes Merkmal eines Objektes (z.B. einen Schriftzug) auf verschiedenen Skalierungen zu suchen, weil dieses Merkmal bei der Bildung der Datenbank auch nur auf einer bestimmten Skala gebildet wird.
Bei dieser Vorgehensweise werden Ereignisse, die nicht in die Bandbreite der Filter fallen, nicht detektiert. Da wir als Merkmal für den Farbkonstanzalgorithmus die lokale Orientierung, also lokale Farbänderungen, verwenden wollen, fallen in diesen Bereich mittel- bis hochfrequente Ereignisse. Wir wählen daher eine Mittelfrequenz von 50 und eine Bandbreite von 3, um sowohl einen gewissen Anteil höher- als auch niederfrequenter Ereignisse zu detektieren. Die verwendeten Basisfilter sind u.a. in Abbildung 4.2 dargestellt.

4.2.4 Transformation der Filter in den Ortsraum

In der Entwicklungsphase unseres Algorithmus' unter dem Bildverarbeitungstool *Khoros II* führen wir die Filterung mit dem Satz von Basisfiltern im Frequenzraum durch. Dies ist bei einer Echtzeitimplementierung aus zwei Gründen nicht möglich:

1. Die Fouriertransformation ist eine Operation, die auf der MV200 nicht implementiert werden kann. Deswegen müßten alle drei Farbkanäle über eine serielle Datenleitung auf den Host-Computer transferiert werden, was ca. $4\,s$ dauern würde.

2. Der Host-Computer der MV200 ist 'nur' eine *Sun Sparcstation 10*, welcher für eine Hin- und mehrere Rücktransformationen eines Eingabebildes über 2 *Min* benötigen würde.

Um die Faltungskapazitäten der MV200 nutzen zu können, ist eine Rücktransformation der Andersson-Filter in den Ortsraum erforderlich. Die 2D-Fouriertransformierte in der Rückrichtung

$$h_{\sigma,\theta,B,\alpha}(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q_{\sigma,B}(\sqrt{u^2+v^2})A_{\theta,\alpha}(arg(u,v))e^{i2\pi(ux+vy)}\,du\,dv \tag{4.20}$$

ist in kartesischen Koordinaten definiert. Die Substitution des Doppelintegrals in Polarkoordinaten mit

$$\begin{array}{ll} u = \rho\cos(\vartheta) & x = r\cos(\varphi) \\ v = \rho\sin(\vartheta) & y = r\sin(\varphi) \end{array} \tag{4.21}$$

ergibt zunächst

$$h_{\sigma,\theta,B,\alpha}(r,\varphi) = \int_0^{\infty}\int_0^{2\pi} Q_{\sigma,B}(\rho)\mathcal{A}_{\theta,\alpha}(\vartheta)e^{i2\pi\rho r\cos(\vartheta-\varphi)}\,\rho\,d\vartheta\,d\rho\,. \tag{4.22}$$

In dieser Darstellung kann die Separabilität des Andersson-Filters ausgenutzt und die 2D- in zwei 1D-Transformationen zerlegt werden, so daß auch das Filter im Ortsraum polar separabel wird:

$$q_{\sigma,B}(r) = \int_0^{\infty} Q_{\sigma,B}(\rho)\rho e^{i2\pi\rho r}\,d\rho \tag{4.23}$$

$$a_{\theta,\alpha}(\varphi) = \int_0^{2\pi} \mathcal{A}_{\theta,\alpha}(\vartheta)e^{i2\pi\cos(\vartheta-\varphi)}d\vartheta\,. \tag{4.24}$$

Analytisch lassen sich diese Teilformeln nicht weiterentwickeln, da die Funktionen unter den Integralen zu verschachtelt sind. Auch eine evtl. Darstellung des angularen Teils durch die Hankeltransformation wird nicht aussagekräftiger. Aus diesen Gründen kann im Ortsraum lediglich mit numerisch transformierten Filtern gearbeitet werden. Die rücktransformierten Basisfilter, welche wir in unseren Experimenten verwenden, sind u.a. in Abbildung 4.2 dargestellt.

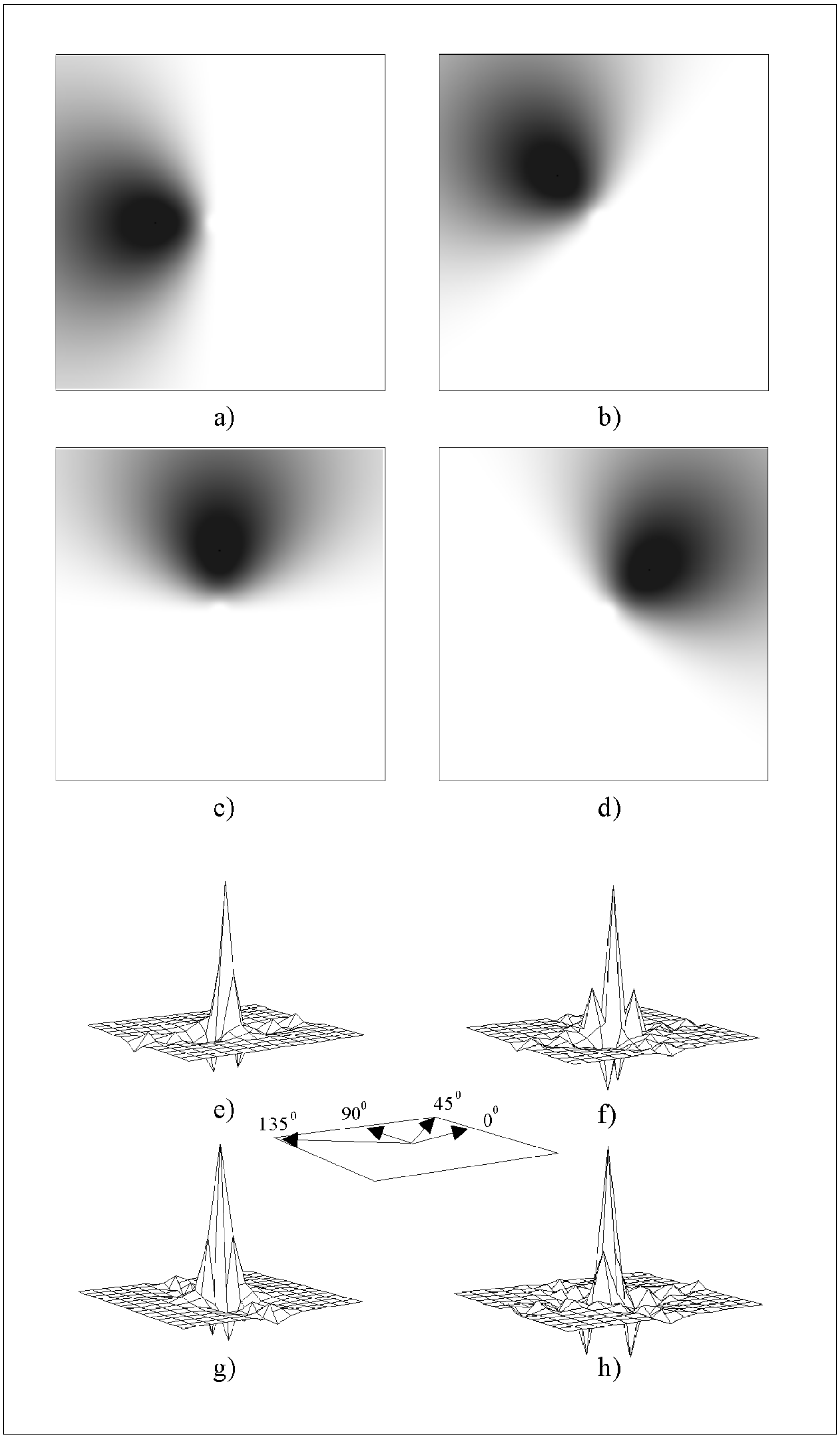


Abbildung 4.2: Darstellung des verwendeten Basissatzes von analytischen Andersson-Filtern im Frequenz- und im Ortsraum mit den Orientierungen $\theta_i = \frac{n\pi}{4}$, Mittelfrequenz $q = 50$ und Bandbreite $B = 3$. a) - d) Filter im Frequenzraum. Schwarz bedeutet Werte nahe 1, Weiß Werte nahe 0. e) - h) 15×15-Filter im Ortsraum.

Kapitel 5

Farbkonstanz durch lokale Orientierung

In diesem Kapitel stellen wir ein zweistufiges Verfahren vor, welches wir für die Lokalisierung und Identifikation von Objekten mittels der Methoden von Swain und Ballard entworfen haben. Dabei können die Resultate des Verfahrens nicht nur für histogrammbasierte Ansätze verwendet werden sondern Grundlage anderer Algorithmen sein. Erster Schritt unseres Verfahrens ist die Konvertierung des RGB-Farbraumes, da dieser Raum einige Nachteile mit sich bringt, die wir kurz erläutern. Zweiter Schritt ist die Berechnung der lokalen Orientierung mit anschließender Vektorquantisierung.
Im Anschluß an den Algorithmus beschreiben wir Eigenschaften von Objekten oder Bildern, die mit unserer Methode extrahiert werden können und leiten daraus eine Methode zur Rückgewinnung der Rotationsinvarianz ab, welche durch die Nutzung der lokalen Orientierung gegenüber dem ursprünglichen Verfahren verloren gegangen ist.

5.1 Nichtlineare Farbraumkonvertierung

5.1.1 Begriff der Farbkonstanz

Das Farbsignal einer Lambert'schen Oberfläche ist das Produkt aus spektraler Beleuchtungsdichte und spektraler Reflektanz der Oberfläche, $C(\lambda) = E(\lambda) \cdot R(\lambda)$, was sich als subtraktive Farbmischung darstellt. Dabei variiert die Beleuchtungsdichte in natürlichen Szenen i.a. langsamer als die Reflektanz; in der Farbanalyse wird deshalb für lokale Bereiche einer Oberfläche eine konstante Beleuchtung angenommen.
Farbkonstanz ist somit die Gewinnung der Faktoren $E(\lambda)$ und $R(\lambda)$ aus einem gegebenen Farbsignal.
Liegt ein Bild in digitalisierter Form vor, ist dabei der Vorgang der Digitalisierung zu berücksichtigen. Dieser Vorgang ist mathematisch gesehen eine Integration des Farbsignals, projeziert auf die jeweiligen spektralen Empfindlichkeiten der Kamerasensoren $S_k(\lambda)$, so daß für die Sensorantworten q_k der Kamerasensoren gilt:

$$q_k = \int_0^\infty C(\lambda) \cdot S_k(\lambda)\, d\lambda \quad , \quad k \in \{\mathrm{R,G,B}\}\,. \tag{5.1}$$

Unter der Annahme, daß die spektralen Empfindlichkeiten orthogonal zueinander sind, und Approximationen von $E(\lambda)$ und $R(\lambda)$ durch

$$E(\lambda) \approx \sum_{k\in\{\mathrm{R,G,B}\}} e_k S_k \quad \text{und} \quad R(\lambda) \approx \sum_{k\in\{\mathrm{R,G,B}\}} r_k S_k \tag{5.2}$$

(ausführlich siehe [MSS95]) ergeben sich die Sensorantworten der Kamera zu

$$\begin{pmatrix} q_R \\ q_G \\ q_B \end{pmatrix} = \begin{pmatrix} e_R \cdot r_R \\ e_G \cdot r_G \\ e_B \cdot r_B \end{pmatrix}\,, \tag{5.3}$$

wobei e_k und r_k Gewichtungsvektoren zur Approximation der Beleuchtung und Reflektanz sind. Somit sind nun die Gewichtungsvektoren statt der ursprünglichen Faktoren zu bestimmen.

5.1.2 Farbraumeigenschaften

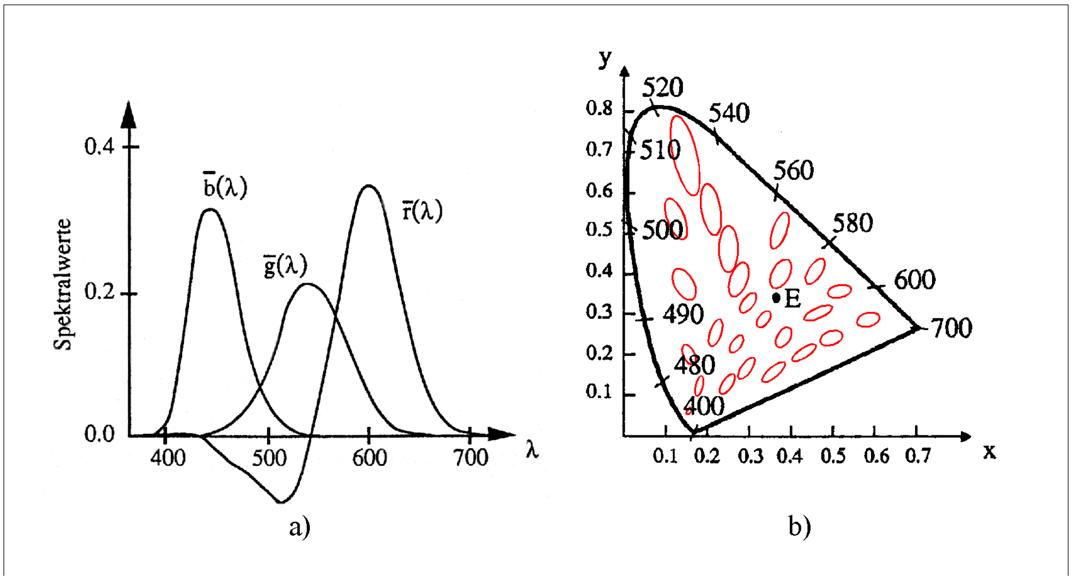
In der Farbbildverarbeitung gibt es verschiedene technisch basierte Farbräume, z.B. RGB, YIQ oder YUV. Größte Bedeutung hat hierbei der RGB-Farbraum. In diesem Raum bilden die Primärfarben Rot, Grün und Blau die Basisvektoren des 3D-Vektorraumes. Ein Farbvektor $\vec{M}$ wird durch additive Farbmischung der Primärfarben gebildet

$$\vec{M} = R\vec{R} + G\vec{G} + B\vec{B}\,, \tag{5.4}$$

wobei R, G und B die Anteile der Primärfarben sind. Ein Nachteil ist, daß nicht alle Farben mit einem Satz von Primärfarben erzeugt werden können, sondern daß ein Anteil nachgemischt werden muß (uneigentliche Farbmischung). Dies zeigt sich auch in den Spektralwertkurven der Primärfarben, wie in Abbildung 5.1 dargestellt. Die Darstellung von 3D-Farbräumen ist schwierig. Deswegen wird auf 2D-Ebenenschnitte eines Raumes, den sog. Farbtafeln, zurückgegriffen. Für das RGB-Modell ist das Maxwell'sche Farbdreieck die allgemein verwendete Schnittebene, woraus sich für einen Farbvektor $\vec{M}$ die Normfarbanteile

$$r_M = \frac{R}{R+G+B} \quad , \quad g_M = \frac{G}{R+G+B} \quad , \quad b_M = \frac{B}{R+G+B} \tag{5.5}$$

berechnen, die linear abhängig voneinander sind: aus zwei gegebenen Anteilen läßt sich der fehlende dritte bestimmen. Somit ist eine 2D-Darstellung ausreichend. Abstände verschiedener Farben in diesem Modell werden vom menschlichen Auge oft unterschiedlich wahrgenommen. So unterscheidet das menschliche Sichtsystem im Grünbereich sehr viel genauer als im Blaubereich. Dieses Phänomen spiegelt sich in den sog. *MacAdam-Ellipsen* wieder: Farben innerhalb dieser Ellipsen können



(aus [KS94])

Abbildung 5.1: a) Spektralwertkurven für die Valenzen $R = 700\,nm$, $G = 546.1\,nm$ und $B = 435.8\,nm$. b) Normfarbtafel mit farbig eingezeichneten MacAdam-Ellipsen.

vom menschlichen Auge nicht unterschieden werden. Für theoretische Betrachtungen wäre eine euklidische Metrik nützlich; die Ellipsen würden sich dann als Kreise darstellen. Eine derartige Metrik ist für den RGB-Farbraum jedoch nicht möglich. In dem CIE[1]-*Lab*-Raum hingegen kann eine euklidische Metrik approximiert werden. Die Abkürzung *Lab* steht für den Luminanzkanal sowie die beiden Chrominanzkanäle Rot-Grün und Gelb-Blau. Der *Lab*-Raum ist für die Modellierung der Verarbeitung von visuellen Reizen im menschlichen Cortex entworfen. Ein 24-Bit-RGB-NTSC[2]-Eingabevektor $\mathbf{r}(\mathbf{x}) = (r_R(\mathbf{x}), r_G(\mathbf{x}), r_B(\mathbf{x}))$ wird zunächst in den CIE-XYZ-Farbraum linear umgerechnet durch

$$\mathbf{r}'(\mathbf{x}) = \begin{pmatrix} 0.607 & 0.174 & 0.201 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.117 \end{pmatrix} \mathbf{r}(\mathbf{x}) \,, \tag{5.6}$$

anschließend erfolgt der nichtlineare Teil mit

$$\rho_L(\mathbf{x}) \;=\; 25\left(\frac{100 r'_Y(\mathbf{x})}{r'_{Y_0}}\right)^{\frac{1}{3}} - 16 \quad \text{für} \quad 0.01 \leq \frac{r'_Y(\mathbf{x})}{r'_{Y_0}} \leq 1\,, \tag{5.7}$$

$$\rho_a(\mathbf{x}) \;=\; 500\left[\left(\frac{100 r'_X(\mathbf{x})}{r'_{X_0}}\right)^{\frac{1}{3}} - \left(\frac{100 r'_Y(\mathbf{x})}{r'_{Y_0}}\right)^{\frac{1}{3}}\right] \quad \text{und} \tag{5.8}$$

[1]Commision Internationale d'Eclairage
[2]National Television Systems Community

$$\rho_b(\mathbf{x}) \quad = \quad 200 \left[\left(\frac{100 r'_Y(\mathbf{x})}{r'_{Y_0}} \right)^{\frac{1}{3}} - \left(\frac{100 r'_Z(\mathbf{x})}{r'_{Z_0}} \right)^{\frac{1}{3}} \right], \tag{5.9}$$

wobei r'_{X_0}, r'_{Y_0} und r'_{Z_0} die Farbanteile des Referenzweiß' im XYZ-Raum sind.
Durch die Kubikwurzeln in (5.7), (5.8) und (5.9), welche ein ähnliches Verhalten aufweisen wie der Logarithmus, bestimmen sich die Sensorantworten nunmehr durch Addition statt durch Multiplikation der Gewichtungsvektoren

$$\begin{pmatrix} q_L \\ q_a \\ q_b \end{pmatrix} \approx \begin{pmatrix} e_L + r_L \\ e_a + r_a \\ e_b + r_b \end{pmatrix}, \tag{5.10}$$

wobei q_k, e_k und r_k, $k \in \{L,a,b\}$ durch Transformation der entsprechenden RGB-Werte in den *Lab*-Raum entstehen. Die subtraktive Farbmischung hat sich somit in eine additive Mischung gewandelt. In der anschließenden Andersson-Filterung, welche eine Hochpaßfilterung ist, werden tiefe Frequenzen unterdrückt. Da wir die Annahme getroffen haben, daß die Beleuchtung langsamer variiert als die Reflektanz, werden bei der Filterung nur die für uns wesentlichen Oberflächenreflektanzen beachtet.

5.2 Berechnung der lokalen Orientierung

Lokale Orientierung bedeutet das Vorhandensein einer bestimmten Richtung in der Signaländerung einer Nachbarschaft, z.B. eine Kante, Ecke o.a. Den Idealfall einer lokalen Orientierung modellieren wir im folgenden durch die Funktion g, für die gilt

$$g(\mathbf{x}) = g(\mathbf{x} \cdot \mathbf{k}), \tag{5.11}$$

wobei $\cdot$ das Skalarprodukt bedeutet und $\mathbf{k} = (k_1, k_2)$ die Richtung der Struktur angibt. Jeder Punkt, der auf einer Geraden senkrecht zu dem Vektor k liegt, weist als konstanten Grauwert den Minimalabstand der Geraden zum Ursprung auf.
Das Verfahren zur Berechnung der lokalen Orientierung basiert auf Überlegungen der Existenz von lokaler Orientierung im Fourierraum. Eine derartige Existenz spiegelt sich wieder im Vorhandensein einer Energiekonzentration auf einer Geraden, welche durch den Ursprung verläuft. Andersson[And92] schlägt eine Schwerpunktberechnung der gewichteten Filterantworten vor, Jähne[Jäh93] gibt für den Fourierraum ein Geradenfit an, welcher aus Überlegungen der Physik zu Trägheitsmomenten an Körpern rührt.
Wir benutzen ein Verfahren von Jähne[Jäh93], bei welchem die Berechnungen im Ortsraum durchgeführt werden. Die Filterantworten h_k unseres Basissatzes aus K Filtern werden als Orientierungsenergien interpretiert und mittels gewichteter Vektoraddition aufsummiert zu

$$o' = \sum_{k=0}^{K-1} h_k e^{i\varphi_k} \in \mathbb{C} \quad \text{mit} \quad \varphi_k = \frac{\pi k}{K}. \tag{5.12}$$

Hierbei stellt sich ein Phänomen ein, welches sich ebenfalls bei den Verfahren im Fourierraum ergibt: die Winkel φ_k der zugehörigen Filterantworten müssen verdoppelt werden. Dies zeigt sich anschaulich, wenn in einer Nachbarschaft eine ausgeprägte lokale Orientierung nicht vorliegt. Für diesen Fall sind die Filterantworten in den jeweiligen Richtungen vom Betrag her gleich und eine Vektoraddition nach (5.12) sollte den Nullvektor ergeben. Tatsächlich resultiert die Addition jedoch in einem Vektor mit der Orientierung $arg(o') = \frac{\varphi_{K-1}}{2}$.

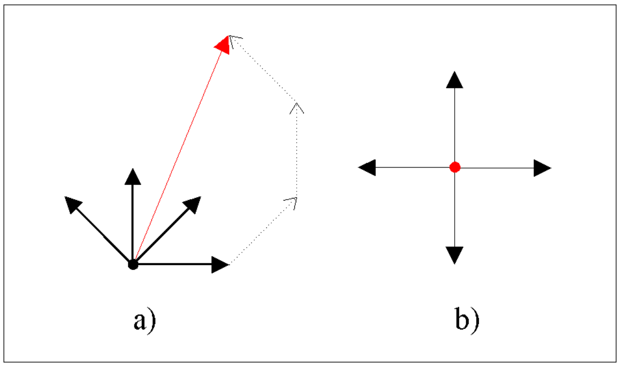


Abbildung 5.2: Notwendigkeit der Verdoppelung der Eingangsinkel φ_k. Farbige Vektoren sind Resultate von Vektoradditionen. a) Ergebnis der Vektoraddition der vier Baisisfilterantworten beim Fehlen einer lokalen Orientierung in einer Nachbarschaft nach (5.12). b) Erwünschtes Ergebnis durch Verdoppelung der Winkel.

Somit modifizieren wir (5.12) zu

$$o = \sum_{k=0}^{K-1} h_k e^{i2\varphi_k} \tag{5.13}$$

und erhalten ein komplexes Ergebnis, aus dem Amplitude $|o|$ und Orientierung $arg(o)$ berechnet werden können. Aus der Verdoppelung der Eingangswinkel erwächst nun die Notwendigkeit, die berechnete Orientierung zu halbieren. So ergibt z.B. eine ideale lokale Orientierung in der Richtung $\frac{\pi}{4}$ mit unserem Basissatz die Orientierungsenergien $h_0 = h_2$ und $h_3 = 0$, woraus eine Orientierung von $\frac{\pi}{2}$ resultiert:

$$\sum_{k=0}^{K-1} h_k e^{i2\varphi_k} = \sum_{k=0}^{3} h_k e^{i\pi k/2} = h_1 e^{i\pi/2}\,. \tag{5.14}$$

Die lokale Orientierung berechnet sich somit zu

$$\theta = \frac{arg(o)}{2} \in [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \tag{5.15}$$

und umfaßt einen Winkelbereich von 180 Grad. Die Einschränkung diese Bereiches ist ein wesentlicher Vorteil gegenüber den Resultaten anderer Verfahren (z.B. des Gradientoperators), welche Winkel in einem Bereich von 360 Grad liefern, die Orientierung eines Musters jedoch nur sinnvoll in dem halben Bereich anzugeben ist.

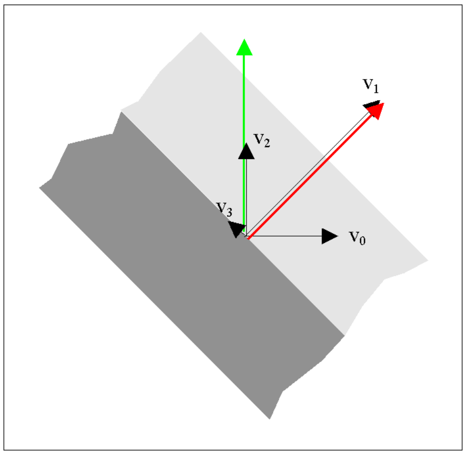


Abbildung 5.3: Notwendigkeit der Halbierung der berechneten Orientierung $arg(o)$. Die Vektoren v_0 bis v_3 stellen zur besseren Anschauung die Summanden $h_k \cdot e^{i\varphi_k}$ dar. Grüner Vektor: Orientierung nach (5.13). Roter Vektor: erwünschte Orientierung unter Hinzunahme von (5.15).

Der 2D-Vektor aus Amplitude und Orientierung ist für die Algorithmen, die wir letztendlich anwenden wollen, noch nicht brauchbar. Wir benötigen einen 1D-Wert (bevorzugt in dem diskreten Bereich $\{0, \ldots, 255\}$), welcher die Ergebnisse der lokalen Orientierung codiert, um über diesen Wertebereich Histogramme zu bilden. Da wir die Information der Orientierung nutzen wollen, kommen zwei Methoden der Vektorquantisierung (VQ) in Frage:

- Polare Quantisierung der Amplitude und Orientierung in einem Skalar oder
- Codierung nur nach der Orientierung.

Der diskrete Wertebereich $\{0, \ldots, 255\}$ setzt uns dabei Grenzen. Bei einer reinen Codierung der Orientierung könnten immerhin noch 256 Winkel mit den jeweiligen Abständen von 0.7 Grad unterschieden werden. Bei einer polaren Quantisierung, in der auch die Größe der Amplitude codiert wird, fallen die Winkelabstände größer aus. Problematisch ist hierbei auch, ob die Bereiche hoher Amplitude feiner quantisiert werden sollten als diejenigen mit niedriger Amplitude (z.B. logarithmisch-polare VQ), oder ob die Anzahl der codierten Winkel pro Stufe gleich bleibt. Weiterhin sollte berücksichtigt werden, daß das Ergebnis der Vektorquantisierung einer erneuten Quantisierung bei der Bildung der 3D-Histogramme unterzogen wird, welches ein relativ grobes Verfahren ist. Die Winkelcodierung sollte bei der VQ noch einen gewissen Feinheitsgrad besitzen.
Der entscheidende Nachteil der polaren VQ liegt in der Intention der Berechnung der lokalen Orientierungen: mehrkanalige Farbbilder sollen mit dieser Methode einer Vorverarbeitung unterzogen und nachfolgende Verfahren farbkonstant werden. Änderungen in der Beleuchtung führen zu Änderungen in der Amplitude der lokalen Orientierung, z.B. auf dem Luminanzkanal eines *Lab*-Bildes oder allgemein auf den Kanälen von RGB-Bildern. Konstant hingegen bleibt die berechnete Orientierung.

So hat eine Abschwächung der Beleuchtungsstärke eine Verkleinerung von Farbkanten in einem Bild zur Folge, wodurch auch die berechneten Amplituden sinken. Unser Algorithmus zur Farbkonstanz sollte – bis zu einem bestimmten Grad – gegen derartige Einflüsse robust sein.

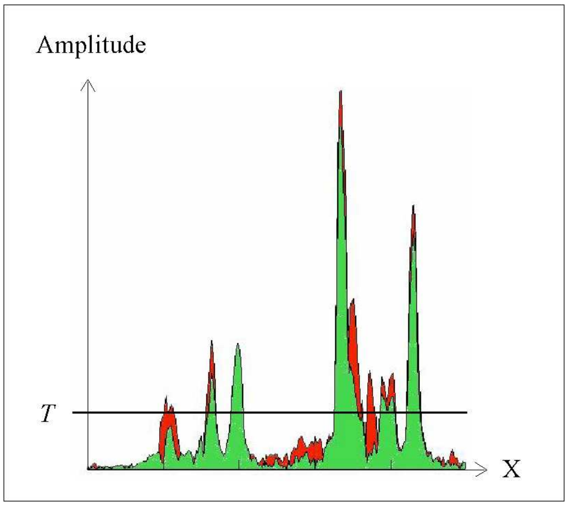


Abbildung 5.4: Querschnitt von Amplituden der lokalen Orientierung des L-Kanals einer Szene unter normaler (grün) und starker Beleuchtung (rot). Deutlich sind die Unterschiede in den Amplitudenverläufen sichtbar. Relativ konstant hingegen sind die Bereiche mit überschwelliger Amplitude, deren Orientierungsinformationen genutzt werden sollen.

Die Schwankungen der Amplitude scheinen im Widerspruch zu den Ergebnissen aus Abschnitt 5.1.2 zu stehen, wonach durch die Andersson-Filterung eines *Lab*-Bildes grundsätzlich nur die Oberflächenreflektanzen detektiert und die Beleuchtungsdichte ignoriert werden sollte. Dies bedingt sich zum einen durch die Annahmen zum *Lab*-Farbraum, daß die spektralen Empfindlichkeiten der Kamerasensoren orthogonal zueinander stehen, und die endlichen Approximationen der Reflektanzen und Beleuchtungsdichten aus (5.2), welches stärkere Einschränkungen in der Modellierung des menschlichen Warnehmungssystems sind. Neben der 'unkonstanten' Amplitude wird aus den Filterantworten des Basissatzes jedoch auch die Richtung der Orientierung bestimmt, welche unabhängig von der Steilheit von Kanten erhalten bleibt.
Aus diesem Grund bietet sich eine Codierung der Amplituden nicht an. Die Amplitude ist jedoch ein Bestimmtheitsmaß für das Vorliegen einer Orientierung und wird bei der Codierung berücksichtigt: erst ab einer gewissen Ausprägung (Steilheit einer Kante) ist die Information der Orientierung zuverlässig. Wir führen daher einen Schwellwert T ein, welcher von der Amplitude überschritten werden muß, damit wir die Orientierungsinformation nutzen. Unterschwellige Amplituden bewirken somit eine Ignorierung der zugehörigen Orientierung, welche wir im folgenden *Nullorientierung* nennen.
Sei $f_d(x, y)$ eine zweistellige Grauwert-Treppenfunktion mit einer einzigen Treppe in Richtung des Winkels α. Die Steilheit der Treppe sei d. Für ein derartiges Modell

wollen wir die Orientierungsinformation nutzen, wenn die Treppe mindestens eine Steilheit von 20 Grauwerteinheiten hat. OBdA nehmen wir die Orientierung der Treppe in X-Richtung an. In diesem Fall können die übrigen Filter vernachlässigt werden, da dies die Richtung eines Basisfilters ist und die übrigen Filter somit nichts zu der Berechnung beitragen (siehe (5.14)). Zu bestimmen bleibt somit die Amplitude in Abhängigkeit von der Steilheit d

$$|o_d| = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_d(u,v) R_{\sigma,B}(\sqrt{u^2+v^2}) A_{\theta,\alpha}(arg(u,v)) e^{i(ux+vy)}\, du\, dv\,, \qquad (5.16)$$

wobei F_d die Fouriertransformierte von f_d ist. Die resultierende Funktion hat einen monotonen, linearen Anstieg. Der gesuchte Schwellwert der Amplituden liegt bei ≈ 6.

Da wir auch in diesem Ansatz die Information der Amplitude nutzen, bleibt eine gewisse Abhängigkeit von der Beleuchtung. Diese fällt jedoch wesentlich geringer aus als bei einer polaren VQ. Lediglich Amplituden, die in dem Grenzbereich des Schwellwertes liegen, sind dem Wechsel von überschwelliger Orientierung zur Nullorientierung ausgesetzt, so daß die quantisierten Orientierungen bei Beleuchtungsänderungen konstant bleiben und sich nur die Anzahl der verwendbaren Informationen verschiebt.

Lösung dieses Dilemmas ist evtl. eine adaptive Steuerung des Schwellwertes. Dieser könnte z.B. von der maximal berechneten Amplitude oder der durchschnittlichen Amplitude eines Bildes abhängig gemacht werden. Dadurch würde jedoch eine Störanfälligkeit unseres Verfahrens entstehen, welche die bereits erläuterten Vorteile aufhebt. Befinden sich z.B. in einem Bildbereich mehrere Objekte, würde der Schwellwert nach dem Objekt mit der höchsten Amplitude ausgerichtet werden. Ist dieses Objekt nicht mehr im Bild, ändert sich auch der Schwellwert. Ähnliches ergibt sich, wenn wechselnde Bildhintergründe mit evtl. vorhandenen Orientierungen nicht ausgeschlossen werden können.

Somit führen wir eine Codierung der Orientierung unter Berücksichtigung der Amplitude gem. folgender Rechenvorschrift durch:

$$s(|o|,\theta,T) = \begin{cases} 1 + \left\lfloor \dfrac{\theta+\pi/2}{\pi} \right\rfloor \cdot 254 & \text{, falls } \quad |o| \geq T \\ 0 & \text{, sonst} \end{cases} \in \{0,\ldots,255\}\ . \qquad (5.17)$$

Dies ist eine Abbildung der überschwelligen Orientierungen von $[-\frac{\pi}{2}, \frac{\pi}{2}]$ in den diskreten Wertebereich $\{1,\ldots,255\}$, in der Null wird das Fehlen von Orientierung codiert. Im folgenden bezeichnen wir diese codierten Ergebnisse auch als *Orientierungsbilder* oder einfach *Orientierungen*. Wir haben damit Resultate, auf welche die histogrammbasierten Verfahren angewendet werden können. Statt der Häufigkeit von Farben wird die Häufigkeit von überschwelligen Orientierungen als wesentliche Information verwendet. Das Vorhandensein von Nullorientierung kann dabei je nach Farbraum unterschiedlich gehandhabt werden. Z.B. sollten in dem *Lab*-Raum bei unterschwelliger Amplitude auf dem Luminanzkanal evtl. vorhandene Orientierungen

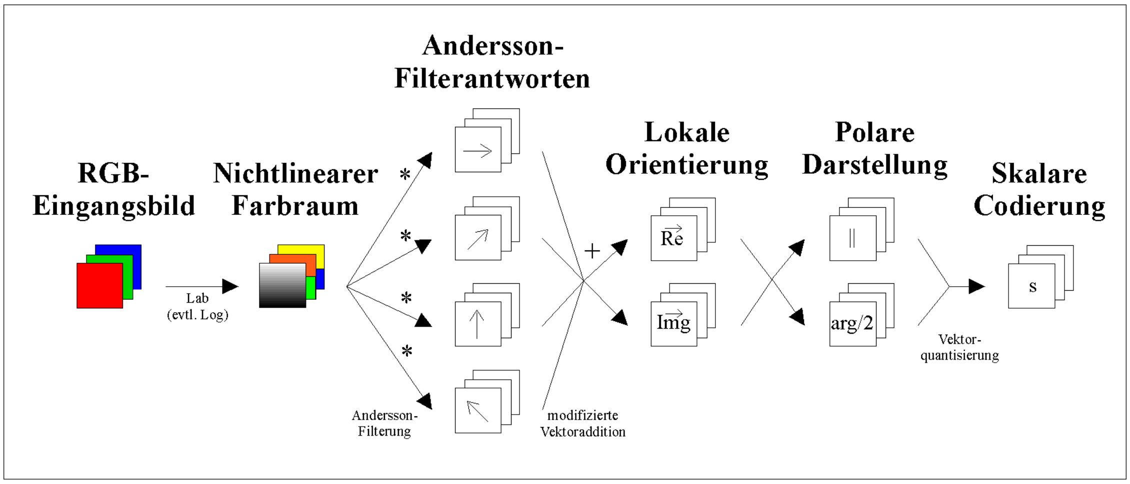


Abbildung 5.5: Zusammenfassung der einzelnen Schritte zur Berechnung und Codierung der lokalen Orientierung.

auf den Chrominanzkanälen ignoriert werden. Im RGB-Farbraum, in dem gleichzeitig über- und unterschwellige Amplituden in den verschiedenen Kanälen vorkommen können, sollte die Nullorientierung nur für den Fall bei der 3D-Histogrammbildung nicht beachtet werden, wenn diese auf allen drei Kanälen vorherrscht.
Unser Verfahren besteht somit aus fünf verschiedenen Schritten, welche wir zur Übersicht in Abbildung 5.5 dargestellt haben.

5.3 Eigenschaften der lokalen und vektorquantisierten Orientierung

Aus dem obigen Verfahren zur Berechnung der lokalen Orientierung ergeben sich für die Orientierungsbilder folgende Eigenschaften:

- Es werden Ereignisse in einer bestimmten Auflösung erfaßt. Darunter fallen nicht nur gerade Kanten, sondern ebenfalls Kurven, Ecken oder das Aufeinandertreffen von drei oder mehreren Kanten, so daß keine Einschränkung auf bestimmte Formen besteht.

- Die Addition sämtlicher Basisfilter ergibt ein isotropes Filter, welches nur noch aus dem Radialanteil besteht. Da die Einteilung der Orientierungen der einzelnen Filter äquidistant ist, wird bei dem Vorliegen einer idealen lokalen Orientierung θ diese mit der modifizierten Vektoraddition exakt bestimmt (zur ausführlichen Berechnung siehe [Jäh93]):

$$\sum_{k=0}^{K} h_k = e^{2\theta} . \tag{5.18}$$

- Wir codieren u.a. den Winkelbereich $[-\frac{\pi}{2}, \frac{\pi}{2}]$, der sich an den jeweiligen Grenzen periodisch fortsetzt. Der Winkel $\frac{\pi}{2}$ erhält den Grauwert 255, $-\frac{\pi}{2}$ entspricht dem Wert 1. In einem Bildbereich, in welchem eine Struktur rotiert und die Orientierung die Grenze von $-\frac{\pi}{2}, \frac{\pi}{2}$ überschreitet, findet ein krasser Wechsel in der Darstellung von Schwarz auf Weiß statt. In mehrkanaligen Bildern deuten grauwertige Orientierungen das Vorhandensein der gleichen Orientierung auf allen Kanälen an, farbige Orientierung auf unterschiedliche Vorkommen.

- Die Verwendung des Schwellwertes T kann je nach Applikation verschieden gewählt werden. Sind z.B. Hintergrundmuster bekannt oder nicht vorhanden, kann ein höherer Schwellwert zur genauen Segmentierung von Objekten herangezogen werden. Unterscheidet sich der Hintergrund farblich stark von den darin vorkommenden Objekten, ist insbesondere an den Konturpunkten der Objekte eine hohe Amplitude vorhanden. Mittels Konturverfolgungsalgorithmen kann so die äußere Form eines Objektes ermittelt werden. Ferner ist direkt aus den Orientierungsbildern mittels der Häufigkeiten überschwelliger Amplituden auf die Größe von Objekten zu schließen. Diese Information kann z.B. bei der Intersektion zusätzlich herangezogen werden, um das Vergleichen unterschiedlich großer Objekte zu vermeiden (siehe Abschnitt 2.2.2.4).

- Die Orientierung berechnet sich unabhängig von der Beleuchtung, lediglich die Anzahl der Bildpunkte, die ungleich der Nullorientierung sind, kann bei Beleuchtungsänderung schwanken. So bewirkt z.B. Verdunkelung die Verringerung der Bildpunkte mit überschwelliger Amplitude auf dem Luminanzkanal, zusätzliche Beleuchtung erhöht diese Anzahl. Betroffen von diesen Änderungen sind nur die Bildpunkte, deren Amplitude im Grenzbereich des Schwellwertes liegt.

- Die einzelnen Schritte sind grundsätzlich alle für eine Implementierung auf dem Pipelinerechner MV200 geeignet und können durch Faltung, arithmetische Operationen oder Lookup-Tabellen realisiert werden. Lediglich die Konvertierung des RGB- in den *Lab*-Farbraum ist nicht möglich. Als Ersatz bietet sich auf diesem System eine Logarithmisierung der Eingangsbilder an.

5.4 Problem der Rotationsinvarianz

Der Algorithmus der Intersektion ist in der Theorie rotationsinvariant aufgrund der Rotationsinvarianz von Farbhistogrammen. Diese Eigenschaft geht verloren, wenn das Verfahren auf Orientierungs- statt Farbbildern angewendet wird, da sich die berechneten Orientierungen durch Rotation der Kamera oder der Objekte ändern. Wir versuchen im folgenden, mittels statistischer Aussagen Aufschluß über das Vorkommen einer oder mehrerer dominanter Orientierungen in einem Bild zu gewinnen, um anhand dieser Information das Bild in eine Normlage zu drehen, die sich unabhängig von der Orientierung des Bildes ergibt.

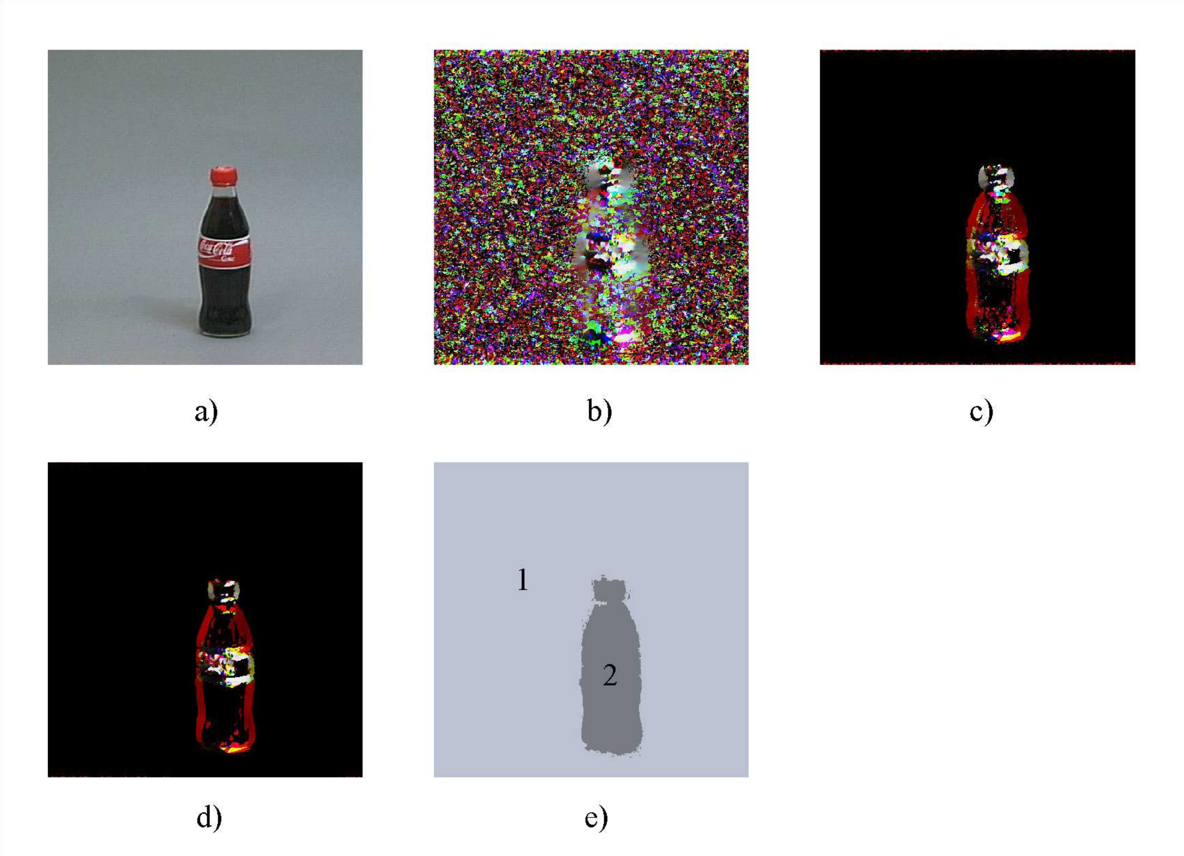


Abbildung 5.6: Berechnete Vektorquantisierung eines *Lab*-Bildes unter verschiedenen Schwellwerten T. a) RGB-Ausgangsbild; B) Quantisierung mit T=0, c) T=6 und d) T=10. e) Segmentierung von d) mit einem einfachen Konturverfolgungsalgorithmus in zwei Bereiche.

5.4.1 Schwerpunktberechnung von Orientierungshistogrammen

Zunächst bietet sich die Berechnung des Erwartungswertes eines Orientierungsbildes an

$$E = \frac{\sum\limits_{s_{xy} \neq 0} s_{xy}}{N_a}, \tag{5.19}$$

wobei s_{xy} die Orientierung an der Stelle (x, y) und N_a die Anzahl der Bildpunkte mit überschwelliger Amplitude ist. Die mittlere Orientierung läßt allerdings die Tatsache außer Acht, daß sich der Wertebereich $\{1, \ldots, 255\}$ zyklisch zu beiden Seiten fortsetzt. Sind z.B. in einem Bild lediglich lokale Orientierungen der Richtung $-\frac{\pi}{2} \pm \triangle_o$ vorhanden, ergibt dies Orientierungen bei $1 + \triangle_o'$ und $255 - \triangle_o'$, wobei sich $\triangle_o'$ aus Umrechnung von $\triangle_o$ durch (5.17) ergibt. Der Erwartungswert bestimmt sich somit zu ≈ 128, was einer lokalen Orientierung von 0 Grad entspricht.
Um die zyklische Fortsetzung des Wertebereiches besser behandeln zu können, be-

trachten wir Orientierungshistogramme statt Orientierungsbilder, wobei ein Orientierungshistogramm eine Funktion $h : \{1, \ldots, 255\} \rightarrow \mathbb{N}_0$ ist. Die dominante Orientierung setzen wir gleich mit dem Schwerpunkt des Histogramms, welcher z.B. mit den 0. und 1. diskreten Momenten

$$M_{p,q} = \sum_{(x,y)\in B} x^p y^q \tag{5.20}$$

berechnet werden kann. Hierbei ist $M_{1,0}/M_{0,0}$ die X-Koordinate und $M_{0,1}/M_{0,0}$ die Y-Koordinate des Schwerpunktes. Auch bei diesem Ansatz ergibt sich das gleiche Problem wie bei dem Erwartungswert, was allerdings nicht verwundert, da der Erwartungswert gleich der X-Koordinate des Schwerpunktes ist.

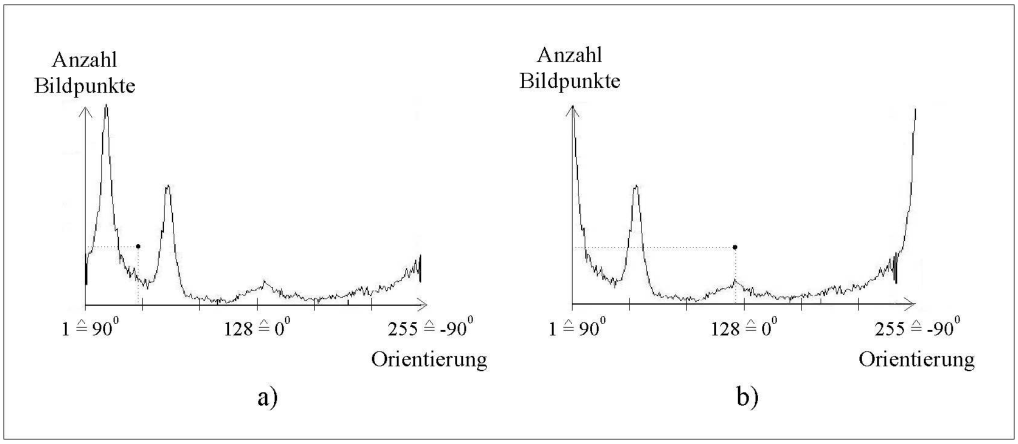


Abbildung 5.7: Problem der zyklischen Fortsetzung des Definitionsbereiches der Orientierungshistogramme. a) Korrekt berechneter Schwerpunkt. b) Verschobenes Orientierungshistogramm von a) mit falschem Schwerpunkt.

Die bisherigen Ansätze beziehen jeweils den gesamten Definitionsbereich in die Berechnungen mit ein, wodurch die Fortsetzung des Definitionsbereiches nicht beachtet wird. Daher betrachten wir lokale Abschnitte von Histogrammen. Eine mathematische Funktion, welche lokale Nachbarschaften und die zyklische Fortsetzung berücksichtigt, ist die zyklische Faltung. Wird das Histogramm mit einer 1D-Maske S der Größe l, welche nur aus 1en besteht, gefaltet, finden sich in dem Faltungsergebnis maximale Werte in den Bereichen, in denen eine Anhäufung von Orientierungen vorherrscht. Das Maximum s mit

$$(h \circledast S)(s) = \max_{x\in\{1,\ldots,255\}} \{(h \circledast S)(x)\}, \tag{5.21}$$

wobei $\circledast$ die zyklische Faltung darstellt, wird als 'Schwerpunkt' interpretiert, obwohl es kein Schwerpunkt im herkömmlichen Sinne ist. Bestimmt wird somit unter Beachtung der zyklischen Fortsetzung ein Bereich des Histogramms, in dem sich Orientierungen anhäufen. Anhand dieses Merkmals kann auf die Ausrichtung von Objekten in einem Bildausschnitt geschlossen werden.

5.4.2 Auswirkungen auf die Intersektion

Mit der obigen Berechnung ergibt sich die Möglichkeit, die Abweichung der Orientierung in einem Bildausschnitt von einer Normorientierung oder Normlage zu bestimmen. Diese Normlage legen wir im Hinblick auf die Intersektion fest bei 0 Grad oder dem 128. Bin eines Orientierungshistogrammes. Ein Bild kann somit in die Normlage gedreht und die Orientierung neu berechnet werden. Da die Intersektion ausschließlich Informationen aus Histogrammen verwertet und diese bei der Bestimmung des Schwerpunktes bereits vorliegen, ist eine Rotation des Bildes nicht notwendig – eine (zyklische) Verschiebung des Histogrammes ist den beiden Schritten gleichwertig.
Die Histogramm-Intersektion definiert im Histogrammraum eine Metrik, welche die L_1-Metrik impliziert (siehe Abschnitt 2.2.2.5), somit werden Abstände von Histogrammen berechnet. Durch die Rotation in die Normlage sind die Histogrammeinträge jedes Histogramms im Bereich des Normbins sehr hoch und gleichen sich stark. Dadurch werden die Histogramme ähnlicher und schwieriger zu unterscheiden. Hauptkriterium ist nicht mehr der Bereich mit den höchsten Einträgen, differenziert wird nach den verbleibenden Bins.
Die Anzahl der Objekte, die auf diese Weise unterschieden werden können, sinkt damit deutlich im Vergleich zur Intersektion auf Orientierungshistogrammen ohne Rotationsinvarianz. So stellt auch Stricker[SO95] fest, daß die Histogramme der zu identifizierenden Objekte gleichverteilt im Histogrammraum sein müssen, damit die Intersektion erfolgreich angewendet werden kann. Eine Gleichverteilung der Orientierung ist nach der Rotation in die Normlage jedoch nicht mehr gegeben. Dies ist eine Schwäche der Intersektion, welche nicht behoben werden kann. Andere Verfahren, welche nicht nur die Information der Häufigkeit von Orientierungen sondern evtl. aus den Orientierungen auf die Form von Objekten schließen, sind für den Fall anzuwenden, daß die rotierten Histogramme zu ähnlich sind und nicht mehr unterschieden werden können.

Kapitel 6

Echtzeitimplementierung

Wir implementieren ein Programm auf dem Pipelinerechner MaxVideo200 der Firma Datacube, welches die folgenden Schwerpunkte beinhaltet:

- Erstellung und Verwaltung von Histogrammdatenbanken, basierend auf RGB- als auch Orientierungsbildern,
- Online-Berechnung von lokalen Orientierungen,
- Online-Lokalisierung von Objekten mittels Rückprojektion von RGB- und Orientierungshistogrammen sowie
- Online-Identifikation von Objekten mittels Intersektion von RGB- und Orientierungshistogrammen.

Die Online-Algorithmen erzielen Laufzeiten von 0.05 bis $0.12\,s$, also Raten von 15 bis zu 8 Bildern pro Sekunde. Wir beschreiben in diesem Kapitel das Programm, welches wir *orientation_indexing* genannt haben. Das Programm setzt sich aus 30 einzelnen Files zusammen, in denen jeweils logisch zusammenhängende Probleme bearbeitet werden. Da die Files insgesamt einen Umfang von über 13.000 Zeilen ergeben, haben wir das Programm nicht in einem Anhang ausgedruckt (ca. 250 Seiten). Es besteht aber die Möglichkeit für den interessierten Leser, sich die Quelltexte über den Lehrstuhl für Kognitive Systeme des Instituts für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel zu besorgen.
In der Programmbeschreibung werden oftmals Fachbegriffe oder Terminologien verwendet, die sich speziell auf die MV200 beziehen. Deshalb geben wir zunächst einen Überblick über die Fähigkeiten des Pipelinerechners und einige Grundbegriffe. Da sämtliche Manuals und Beschreibungen über die MV200 in englischer Sprache verfaßt sind, haben sich bestimmte englische Fachausdrücke eingebürgert, die wir nicht ins Deutsche übersetzen sondern unverändert benutzen. Danach gehen wir auf die Implementierung der einzelnen Probleme ein. Wir beschränken uns dabei auf eine oberflächliche Beschreibung und gehen nicht zu sehr ins Detail, weil dies zum einen sehr umfangreich werden würde, und wir zum anderen bei dem Leser ausgiebige Kenntnisse des Pipelinerechners voraussetzen müßten.

6.1 Terminologien

6.1.1 Pipelineprozessor

Ein Pipelineprozessor unterscheidet sich grundsätzlich von einem 'normalen' Prozessor (CPU), der allg. auch in Workstations oder PCs vorhanden ist. Eine CPU kann jeweils eine Berechnung pro Zeiteinheit auf einer bestimmten Datenmenge vollziehen. Alle Daten müssen vorbereitet, berechnet und anschließend gespeichert werden. Dies geschieht sequentiell. Die Geschwindigkeit, mit der eine Berechnung durchgeführt werden kann, wird dabei durch die Taktfrequenz des Prozessors bestimmt und in Mhz oder Flops (floating point operations per second) angegeben. In der Bildverarbeitung sind herkömmliche CPUs jedoch schnell überfordert: allein die Aufnahme und Darstellung eines 768×576-Bildes benötigt auf einer Sun Sparcstation10 $0.16\,s$, so daß nur eine Rate von 6 Bildern pro Sekunde erreicht wird, ohne eine einzige Berechnung auf dem Bild vorgenommen zu haben.

Im Gegensatz dazu besteht ein Pipelineprozessor aus einer Vielzahl von einzelnen Prozessoren, die variabel miteinander verbunden werden können oder bereits verbunden sind. Die vorliegende Konfiguration der MV200 besteht aus mehr als 1000 Einzelprozessoren. Jeder Prozessor ist auf eine Aufgabe spezialisiert. So gibt es z.B. einfache Prozessoren, die lediglich eine Addition zweier Zahlen durchführen können, oder komplexere, die ein Eingangsbild mit einer Maske falten. Je nach Problemstellung werden diese Prozessoren miteinander verknüpft; während Prozessor 1 Berechnungen auf einem Pixel durchführt, berechnet der nachfolgende Prozessor 2 bereits auf dem Ergebnis des letzten Pixels von Prozessor 1 usw. Somit arbeitet ein Pipelineprozessor auf einem kontinuierlichen Datenstrom. Dieser Datenstrom kann sich aus verschiedenen einzelnen Strömen zusammensetzen, die parallel abgearbeitet werden. Alle einzelnen Prozessoren und Datenleitungen werden durch ein Taktsignal gesteuert. Die Geschwindigkeit des Pipelinerechners wird dadurch in Mhz angegeben; die MV200 ist grundsätzlich mit 20 Mhz getaktet, was einer Verarbeitungskapazität von 20 Mio Pixel pro Sekunde entspricht.

6.1.2 Imageflow

Die MV200 wird von einem Host-Computer initialisiert und gesteuert. Dies geschieht mittels eines dort erstellten C-Programmes. Die Firma Datacube liefert dafür ca. 2000 Bibliotheksfunktionen für die Programmiersprache C, die unter dem Begriff *Imageflow* zusammengefaßt werden. *Imageflow* bietet einen objektorientierten Ansatz zur Programmierung; sämtliche Komponenten der MV200 oder *Imageflows* werden als Objekte behandelt, die bestimmte Eigenschaften besitzen. Der Programmierer kann solche Objekte erzeugen, manipulieren und verwerfen, ohne zu große Kennnisse über die Hardware haben zu müssen. Gleichzeitig ist *Imageflow* sehr anpassungsfähig gegenüber Änderungen in der Konfiguration einzelner Teile der MV200.

6.1.3 Boards und Devices

Ein Pipelinerechner der Firma Datacube läßt sich modular zusammensetzen. Je nach Bedarf kann die Konfiguration erweitert, geändert oder verringert werden. Größtes Bauteil ist ein sog. *Board*. Der 'Lehrstuhl für Kognitive Systeme' am Informatik-Institut der Christian-Albrechts-Universität zu Kiel, an dem diese Arbeit geschrieben wird, hat z.Zt. eine Konfiguration von 7 *Boards*, von denen wir in unserer Implementierung 5 benutzen. Typische *Boards* sind z.B. das *Digicolor-Board*, mit dem RGB-Bilder digitalisiert werden können, oder ein *AB-Board*, auf dem sich Speichermodule, ein arithmetisch-logischer Teil o.a. befinden.
Ein *Board* läßt sich wiederum in verschiedene *Devices* untergliedern. *Devices* sind z.B. die eben erwähnten Speichermodule (*AM-Device*) oder die arithmetischen Einheiten (*AU-Device*). Diese lassen sich modular auf den *Boards* anbringen, so daß es vom Typ her gleiche *Boards* geben kann, die jedoch mit unterschiedlichen *Devices* besetzt sind. Durch diese Konzeption kann ein Rechner je nach der Problemanforderung individuell zusammengestellt werden, was eine flexible Einsatzfähigkeit der Pipelinerechner zur Folge hat. Nachteil dieser modularen Bauweise ist die Tatsache, daß Programme von verschiedenen Rechnern nicht ohne weiteres übertragbar sind. Eine Anpassung muß – wenn überhaupt möglich – für jeden programmierten Datenstrom vorgenommen werden.

6.1.4 Datenfluß

Die Verknüpfung einzelner Prozessoren zu einem Datenstrom wird *Pipeline* oder kurz *Pipe* genannt. Die Auswahl, welche Prozessoren mit welchen verbunden werden können, ist eingschränkt und muß dem jeweiligen Hardware-Manual entnommen werden. Eine *Pipeline* startet grundsätzlich von einem Bereich in einem *Speichermodul*, einem sog. *Surface*, und endet in einem *Surface*. Dazwischen werden Berechnungen durch die zugeschalteten Prozessoren durchgeführt. Durch die Größe eines rechteckigen Fensters, dem sog. *Rectangle*, und dessen linker oberen Ecke, dem sog. *Alignment-Point*, wird bestimmt, welche Daten eines *Surfaces* bei einem Datentransfer berücksichtigt werden.
Eine *Pipeline* kann eine verschiede Anzahl von Start- und Ziel-*Surfaces* besitzen:

- ein oder mehrere Start-*Surfaces* (*Single Source Pipe* und *Multi Source Pipe*) und
- ein oder mehrere Ziel-*Surfaces* (*Single Destination Pipe* oder *Multi Destination Pipe*).

Dadurch sind 4 Topologien für eine *Pipeline* möglich, die wir in Abbildung 6.1 dargestellt haben.
Ist der Pfad für eine *Pipeline* korrekt aufgebaut und alle Einstellungen an den dort benutzten Prozessoren vorgenommen, muß die *Pipe* zunächst *gearmed* werden. Das Durchfließen des Datenstroms durch die Pipeline wird dann durch das *Feuern* der Pipe erreicht. Dabei gibt es zwei Typen von *Pipes*, die unterschieden werden können:

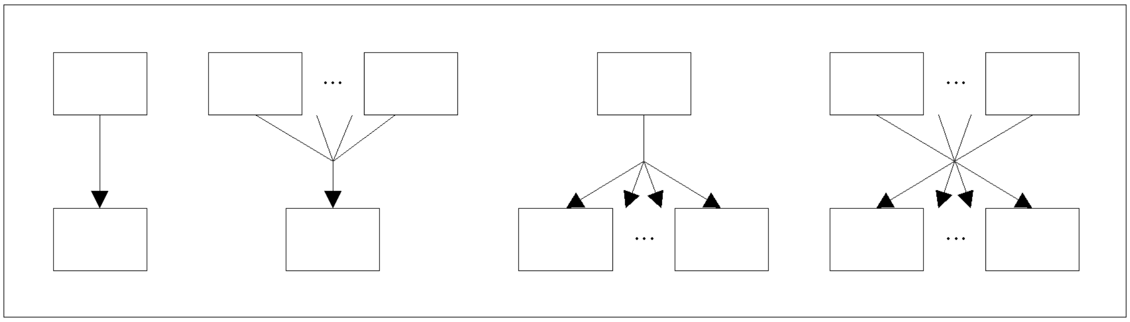


Abbildung 6.1: Mögliche Topologien einer *Pipeline*, bedingt durch die unterschiedliche Anzahl von Start- und Ziel-*Surfaces*.

- *One-Shot-Pipes* und
- *Continuous-Pipes*.

Bei einer *One-Shot-Pipe* wird der Inhalt des oder der Start-*Surfaces* nach dem *Feuern* einmalig durch die Prozessoren innerhalb der *Pipe* bearbeitet und in dem oder den Ziel-*Surfaces* gespeichert. Eine erneute Abarbeitung dieser *Pipeline* setzt widerum das *Armen* und *Feuern* dieser *Pipe* voraus. Dazwischen können bestimmte Parameter wie z.B. die Größe und Position des *Rectangles* oder Einstellungen von Prozessoren geändert werden. Eine *Continuous-Pipe* braucht nur ein Mal *gefeuert* zu werden. Sobald sie beendet ist, wird sie automatisch erneut *gefeuert*. Dies geschieht so lange, bis die *Pipe* verworfen wird. Eine weitere Kontrolle über eine derartige *Pipe* ist nach dem *Feuern* nicht möglich.
Grundsätzlich werden in der Programmierung des Pipelinerechners nur *One-Shot-Pipes* verwendet. Dies begründet sich zum einen an der Hardware (maximal vier *Pipes* können gleichzeitig aktiv sein), und zum anderen an den Problemstellungen, die der Pipelinerechner lösen soll. Kaum ein Algorithmus läßt sich durch eine einzige *Pipe* realisieren, sondern benötigt mehrere sequentiell ablaufende *Pipelines*. *Continuous Pipes* bieten sich z.B. bei der Digitalisierung von Bildern oder bei der Darstellung der Inhalte von *Surfaces* auf dem Bildschirm an.
Wie bereits oben erwähnt, können maximal vier *Pipes* parallel laufen. Dies geht allerdings nur unter der Voraussetzung, daß die einzelnen *Pipes* verschiedene Hardwarekomponenten benutzen, z.B. verschiedene Speichermodule als Start- und Ziel-*Surfaces*. Sollten dennoch einmal *Pipes* mit konkurrierender Hardware parallel laufen, ist ein Programmabsturz wahrscheinlich, zumindest aber sind die Ergebnisse der *Pipes* nicht mehr deterministisch vorhersagbar.

6.1.5 Datentypen

Die Speichermodule der MV200 können Daten mit einer Tiefe bis zu 8 Bits speichern, d.h. die *Surfaces* enthalten nur ganzzahlige Werte zwischen 0 und 255 (bzw. -128 und $+127$ im Zweierkomplement). Dies begründet sich in den MaxBus-Datenleitungen des Pipelinerechners, welche die einzelnen *Devices* und *Boards* miteinander verbinden. Diese Leitungen haben lediglich eine Breite von 8 Bits, nur innerhalb einzelner *Devices* gibt es Datenleitungen mit größeren Breiten. Da die Ausgabe bei vielen Operationen der MV200 (z.B. der Faltung, Addition oder Multiplikation) sehr schnell Werte $\gg 255$ liefert, kann man einem Bild zwei *Surfaces* zuordnen, die das höherwertige und niederwertige Byte eines 16-Bit-Wertes enthalten. Dies erweitert den Wertebereich auf 0 bis 65535 (bzw. -32768 bis $+32767$). Die Handhabung von 16-Bit-*Surfaces* ist jedoch komplizierter als vorher, weil viele Prozessoren nur 8 Bit als Eingabe zulassen und die Datenleitungen nicht 16 Bits gleichzeitig transportieren können. Dadurch müssen zwei getrennte Pfade gelegt und zum geeigneten Zeitpunkt, etwa vor dem Eintritt in einen Prozessor, zusammengefügt werden. Weiterhin stehen aber nur ganzzahlige Werte zur Verfügung – Floatig-Point-Arithmetik ist auf der MV200 nicht möglich.

6.1.6 PATs

Die einzelnen Prozessoren der MV200 befinden sich entweder im Zustand des Arbeitens oder des Wartens. Das Warten ergibt sich durch Zeiten, welche durch das Aufbauen und *Armen* der *Pipes* durch den Host-Computer entstehen. Benötigt z.B. ein Problem die Abarbeitung von mehreren sequentiellen *Pipes*, kann eine *Pipe* erst nach Beendigung der letzten aufgebaut werden. Um die Wartezeit dazwischen möglichst gering zu halten, werden sog. *PATs* (Pipeline Altering Threads) benutzt. Ein *PAT* ist ein Prozeß auf Betriebssystemebene, der speziell für die Steuerung von *Pipelines* auf der MV200 entworfen wurde. Innerhalb eines *PATs* können mehrere *Pipes* aufgebaut, gearmed und gefeuert, sowie begrenzt Einstellungen der verschiedenen Prozessoren vorgenommen werden. Ein *PAT* wird in einer Offline-Phase definiert und in der Online-Phase durch einen einzigen Befehl gestartet. Es können keine herkömmlichen C-Variablen benutzt werden, um Abläufe innerhalb des *PATs* in der Online-Phase zu steuern. Dazu werden spezielle *PAT*-Variablen herangezogen. Diese *PAT*-Variablen sind 24-Bit-Variablen, die ganzzahlige Werte speichern können. Der Zugriff erfolgt etwas umständlich über Akkumulator-Funktionen, wie sie aus der Maschinensprache bekannt sind.
Wesentlicher Vorteil der *PATs* ist die Tatsache, daß bereits zur Laufzeit einer *Pipe* die Befehle zum Aufbau der nächsten *Pipe* abgearbeitet werden können und so ein erheblicher Geschwindigkeitsvorteil gegenüber der 'normalen' Methode des Aufbaus von *Pipes* entsteht. Leider ergeben sich durch den Einsatz von *PATs* auch einige Nachteile:

- Nachdem ein *PAT* definiert ist, können keine Einstellung darin mehr vorgenommen werden, also weder an den *Pipes* noch den benutzten Prozessoren. Das *PAT* müßte dazu gelöscht und anschließend neu gebildet werden, wodurch der Geschwindigkeitsvorteil verloren ginge.

- Innerhalb eines *PATs* kann nur ein begrenzter Teil von *Imageflow*-Funktionen eingesetzt werden. U.a. läßt sich die Größe eines *Rectangles* nicht mehr und der *Alignment-Point* nur noch mit großem Aufwand verändern, wodurch die Anwendbarkeit von *PATs* erheblich eingeschränkt wird.

- Der Source-Code des Programmes wird durch den Einsatz der *Imageflow*-Funktionen bzgl. *PATs* sehr unübersichtlich. Normaler C-Code vermischt mit *PAT*-Funktionen verhält sich anders als erwartet und ist dadurch sehr fehleranfällig.

- Es gibt keine Möglichkeit des Debuggens. Stürzt ein *PAT* ab, lassen sich keine Variablenwerte oder Registerzustände heranziehen, um die Fehlerursache zu finden.

Nichtsdesweniger sind *PATs* die schnellste Möglichkeit, um verschiedene, voneinander abhängige *Pipes* zu steuern. Wir hatten bei unserer Implementierung als Ziel, die Online-Algorithmen vollständig innerhalb von *PATs* laufen zu lassen. Wegen der begrenzten Auswahl an Funktionen, die zur *PAT*-Definition benutzt werden können, ist uns dies zu 98% gelungen. In den verbleibenden 2% werden jedoch keine *Pipes* aufgebaut oder gefeuert, so daß hierdurch kein Zeitverlust entsteht.

6.2 Implementierung

Folgende Programmteile werden von uns in diesem Abschnitt beschrieben, die speziell für den Pipelinerechner entwickelt sind:

- Digitalisierung und Skalierung von RGB-Bildern,
- Berechnung von 3-dimensionalen Histogrammen,
- Berechnung der lokalen Orientierung,
- Rotation in den Histogrammschwerpunkt,
- Rückprojektion mit anschließender Glättung sowie
- Intersektion von 3D-Histogrammen.

Weniger interessant ist dagegen die Erzeugung und Verwaltung von Histogrammdatenbanken, die auch auf herkömmlichen Rechnern implementiert werden können. Wir gehen somit nur auf die Online-Phasen der MV200-spezifischen Algorithmen ein.

6.2.1 Digitalisierung

Wichtiger Bestandteil der vorliegenden Konfiguration der MV200 ist das *Digicolor-Board*. An dieses Board können verschiedene Farb-Vidoekameras mit unterschiedlichen Signalausgängen (z.B. Pal oder RGB) oder ein Videorekorder angeschlossen und daraufhin Bilder digitalisiert werden. Wir benutzen eine Kamera der Firma Sony (DXC-151A). Die *Digicolor* kann Bilder der Abmessungen 502×576 einlesen. Da aus einem unbekannten Grund die letzten 16 Zeilen keine Bildinformationen enthalten (RGB-Wert der Pixel = $(0, 0, 0)$), ergibt sich somit ein 502×560-Farbbild.
Das *Digicolor-Board* verfügt nicht über Speichermodule zum Zwischenspeichern der digitalisierten Bilder. Daher wird aus dem Board eine *Multi-Destination-Pipe* in ein *AB-Board* gelegt, in dem die drei Farbanteile Rot, Grün und Blau gespeichert werden. Wir haben bei der Konstruktion aller anderen *Pipes* darauf geachtet, daß diese keine Hardware benutzen, die von der Digitalisierungs-*Pipe* benötigt wird. Dadurch ergibt sich die Möglichkeit, diese als *Continuous-Pipe* einmalig zu starten und somit permanent laufen zu lassen. Während der Entwicklung unseres Programmes haben wir davon jedoch aus folgenden Gründen wieder Abstand genommen:

- Die Andersson-Filterung wird nacheinander auf die drei Farbkanäle angewendet. Dabei soll vermieden werden, daß während der Berechnung des 1. oder 2. Kanals bereits ein neues Bild in den Speichermodulen abgelegt wird, so daß die nachfolgende Filterung des 2. oder 3. Kanals auf einem anderen Bild stattfindet.

- U.a. wird auf einem Monitor das digitalisierte Bild ausgegeben. Der Transfer der Daten der einzelnen Farbkanäle erfolgt als 'Nebenprodukt' der Skalierung und Andersson-Filterung, also zu unterschiedlichen Zeitpunkten. Bei einer kontinuierlichen Digitalisierung kann wegen fehlender Abstimmung der Fall auftreten, daß Farbkanäle unterschiedlicher Bilder dargestellt werden. Dies ist insbesondere störend bei Szenen, in denen Bewegung vorkommt, weil dann die Farbkanäle der bewegten Bildausschnitte gegeneinander verschoben sind.

Die *Digicolor* ist mit $10Mhz$ getaktet, alle anderen *Boards* jedoch mit $20Mhz$. Aus diesem Grund muß das Signal beim Eintritt in das *AB-Board* künstlich verdoppelt werden, um die verschiedenen Taktfrequenzen anzugleichen. Rein rechnerisch ergibt sich somit eine Pipelaufzeit von $0.028\,s$, die tatsächliche Laufzeit liegt bei $0.069\,s$. Diese Zeitdifferenz resultiert aus dem Aufbau und dem *Armen* der Pipe sowie verschiedenen Verzögerungselementen, die von der MV200 in den Datenfluß zwischengeschaltet werden und auf die der Programmierer keinen Einfluß hat.
Die Einstellungen, die an der *Digicolor* vorgenommen werden können, sind etwas begrenzt. Die Möglichkeit, horizontale oder vertikale Abtastfrequenzen der Kamera einzustellen, wie z.B. in dem *Device* für Grauwertdigitalisierung auf einem *AB-Board*, gibt es nicht. Die Sony-Farbkamera wurde für die Aufnahme von Bildern für Video-Rekorder und Fernsehgeräte konzipiert. Fernseher haben grundsätzlich ein Verhältnis der horizontalen und vertikalen Bildröhre von 2/3. Demzufolge liefert die

Kamera ein Bild, das mit der 1.5-fachen vertikalen Frequenz abgetastet wird wie in der Horizontalen. Wir erhalten in der Digicolor ein in die Höhe verzerrtes Bild, das sich durch bloße Unterabtastung nicht entzerren läßt.
Aus diesem Grund werden die drei Farbanteile mit einem *Mini-Warper*, der sich auf einem *AP-Device* befindet, nachbearbeitet. Der *Mini-Warper* bietet die Möglichkeit, Bilder in Echtzeit geometrischen Transformationen wie Rotation, Translation oder Skalierung zu unterwerfen. Wir skalieren das Bild in X- und Y-Richtung um die Faktoren 1/2 und 1/3 und nehmen neben der Entzerrung gleichzeitig eine Unterabtastung vor. Dies geschieht aus dem Grunde, da auf dem Ausgabemonitor Bilder der maximalen Größe 512×512 darstellen werden können, wir aber neben dem aufgenommenenen RGB-Bild noch andere Darstellungen (z.B. die lokale Orientierung, die Rückprojektion o.a.) gleichzeitig betrachten wollen. Somit ergibt sich die Größe unseres RGB-Ausgangsbildes zu 251×186.
Eine Farbraumkonvertierung des RGB-Bildes in den *Lab*-Raum ist mit reiner Integer-Arithmetik und den Möglichkeiten des Pipelinerechners nicht realisierbar. Um die Vorteile eines nichtlinearen Farbraums trotzdem zu erhalten, nehmen wir eine Logarithmisierung mit anschließender Ausweitung des Wertebereiches auf $\{0, \ldots, 255\}$ vor. Dieser Schritt erfolgt direkt nach der Skalierung eines Farbkanals mit einer 8-Bit-Lookuptable (LUT).
Die Laufzeiten der *Skalierungs-Pipes* beträgt $0.055\,s$ (im Vergleich zur rechnerischen Zeit von $0.042\,s$).
Die eben beschriebenen Pipes und PATs finden sich wieder in den Files *acq.c* und *warp.c*.

6.2.2 Berechnung von 3D-Histogrammen

Grundlage sowohl für die Intersektion als auch die Rückprojektion ist die Erstellung von Farbhistogrammen. Eine Histogrammberechnung bietet sich grundsätzlich auf der MV200 an, da für diese Aufgabe spezielle Prozessoren in den sog. *AP-Devices* vorhanden sind. Einziger Nachteil dieser Prozessoren ist, daß sie lediglich Histogramme von Grauwertbildern mit insgesamt 256 Bins, also Bildern mit einer Auflösung von 8 Bit, erzeugen können. Wir benötigen jedoch 3D-Histogramme mit 512 Bins.
Bei einer Auflösung von 8 Bins pro Farbkanal berechnet sich das Bin eines RGB-Wertes (c_r, c_g, c_b) zu $(c_r/32, c_g/32, c_b/32)$ (/ bedeutet in diesem Fall ganzzahlige Division ohne Rest). Die 3D-Darstellung des Histogrammes wird im Computer eindimensional gespeichert. Um das 3D-Bin (i, j, k) anzusprechen, wird das $i + j * 8 + k * 64$. Bin des 1D-Histogramms ausgewählt. Diese Rechenvorschrift machen wir uns zu Nutze, indem wir bereits vor Berechnung des Farbhistogrammes den RGB-Wert in einen 1D-'Grauwert' umrechnen:

$$(c_r, c_g, c_b) \rightarrow (c_r/32) + (c_g/32) \cdot 8 + (r_b/32) \cdot 64\,. \tag{6.1}$$

Wir erhalten ein Bild mit einer Tiefe von 9 Bit, in dem die ursprünglichen Farbvektoren aufgrund der Quantisierung der 3D-Histogramme in einen Farbwert umgewandelt worden sind. Die Bildung eines 'herkömmlichen' Histogrammes über diese

Werte kommt der vorherigen Histogrammberechnung mit anschließender linearer Speicherung gleich.
Diese Berechnung wird in einer *Multi-Source-Multi-Destination-Pipe* vorgenommen. Wir leiten die RGB-Anteile im artithmetischen Teil des *AB-Boards* dem sog. *AU-Device* zusammen und führen die Umwandlung aus (6.1) durch. Anschließend wird das niederwertige Byte und das Höchstwertige Bit (MSBit) in zwei *Surfaces* gespeichert.
Als einziges Problem bleibt die Tiefe von 9 Bit, da die Statistik-Prozessoren nur 8 Bit verarbeiten können. Dies kann mit einem sog. *MOSC* (Multiple Operation State Controller) gelöst werden, der mit dem Statistik-Prozessor gekoppelt ist. Der Controller kann für jedes Pixel bestimmen, ob dieses in die Histogrammberechnung einbezogen wird oder nicht. Wir leiten dazu das Surface mit den MSBits aus der vorherigen Rechnung in den Controller und das mit den niederwertigen Bytes in den Statistik-Prozessor. Ferner progammieren wir den Controller so, daß Pixel nur dann in die Histogrammberechnung einfließen, falls das zugehörige MSBit gleich 0 ist. Dadurch erreichen wir, daß ein Histogramm über die ersten 8 Bits gebildet wird, also über Werte zwischen 0 und 255. In einer zweiten *Pipe* invertieren wir das MSBit und erhalten so ein Histogramm über Werte zwischen 256 und 511. Diese zwei einzelnen Histogramme können dann als ein 512 Bin großes Histogramm betrachtet werden.
Da in der vorliegenden Konfiguration der MV200 sechs *AP-Devices* vorhanden sind, kann die Erzeugung der beiden Histogramme parallel erfolgen.
Grundsätzlich wird ein 3D-Orientierungshistogramm genauso berechnet wie ein Farbhistogramm. Lediglich die Einträge, die in allen drei Kanälen 0 sind (*Null-Orientierung*), sollen nicht beachtet werden. Dazu ist es notwendig, vor der Histogrammberechnung und somit vor der Quantisierung diese Bildpunkte zu kennzeichnen. Dies geschieht mit einer *Multi-Source-Pipe*, welche in einem *AU-Device* die drei Farbkanäle schlicht addiert und nach dem Durchgang durch eine *LUT* binär die *Nullorientierung* und das 9. Bit der RGB-Quantisierung für jeden Bildpunkt codiert. Bei der anschließenden Berechnung der Histogramme wird dieses *Surface* statt des *Surfaces*, welches nur die Information des 9. Bits enthält, in die *MOSCs* eingespeist, so daß Bildpunkte mit der *Nullorientierung* herausgefiltert werden können.
Den zwei- bzw. dreistufigen Vorgang der Histogrammberechnung haben wir graphisch in Abbildung 6.2 dargestellt. Separate Zeitmessungen für diesen Vorgang haben wir nicht erstellt sondern nur im Zusammenhang mit der Rückprojektion und Intersektion vorgenommen.
Die soeben beschriebenen Pipes und dazugehörigen PATs sind in den Files *histo.c* und *histo.h* definiert.

6.2.3 Berechnung der lokalen Orientierung

Wohl einer der größten Vorteile der MV200 gegenüber herkömmlichen Computern in der Bildverarbeitung ist durch die Faltungsprozessoren gegeben. Mit wachsender Bild- oder Maskengröße steigt die Zahl der Rechenoperationen stark an. In der

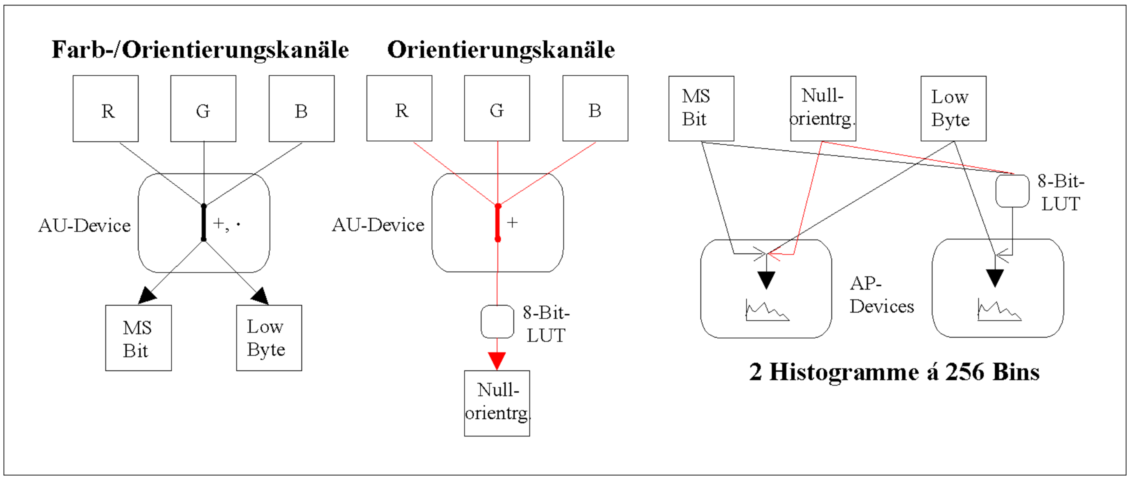


Abbildung 6.2: Pipes zur Berechung von 3D-Farbhistogrammen. Farbig dargestellte Datenflüsse werden nur bei Orientierungshistogrammen benötigt und ersetzen evtl. vorhandene Verbindungen.

vorliegenden Konfiguration der MV200 sind 18 Faltungsprozessoren vorhanden, die mit Masken der Größe 8×8 programmiert werden können. Dadurch ist es möglich, auch mit größeren Faltungsmasken durch Aufsplittung in mehrere kleine zu arbeiten.
Wir benötigen in unserer Berechnungen die Faltung jedes Farbkanals mit vier Andersson-Basisfiltern, welche die Abmessungen 15×15 haben. Ein einzelnes Filter wird dabei durch vier 8×8-Filter realisiert. Dies wird durch geeignete Verschiebung der *Alignment-Points* der Masken und anschließender Addition der Filterantworten erreicht, wie in Abbildung 6.3 veranschaulicht.
Aus Gründen der Genauigkeit benutzen wir 16 Bit der Faltungsergebnisse in den anschließenden Berechnungen. Da in der Konfiguration zwei *AU-Devices* vorhanden sind, können parallel vier Faltungen mit anschließender Addition von je zwei Faltungsergebnissen vorgenommen werden. In einer darauf folgenden *Pipe* werden die Resultate der Additionen nochmals addiert, so daß schließlich das Ergebnis einer 16×16-Faltung vorliegt. Die Berechnungen haben wir in Abbildung 6.4 dargestellt.
Dieser Vorgang wird für die restlichen drei Basisfilter wiederholt. Anschließend gewichten wir die vorliegenden Filterantworten mit Interpolationsfunktionen und addieren diese. Da die Interpolationsfunktionen in unserem Fall konstant 0 oder 1 sind (vgl. Abschnitt 5.2), bleiben lediglich zwei Additionen vorzunehmen. Die beiden Additionsergebnisse werden auf 8 Bit begrenzt, weil im nächsten Schritt eine höhere Auflösung nicht möglich ist.
Aus dem komplexen Ergebnis bleibt die Amplitude und Phase zu berechnen. Dazu werden die zwei Additionsresultate in einer 16-Bit-LUT zusammengeführt. Dort kann in einem einzigen Schritt Phase und Amplitude berechnet, eine Vektorquantisierung vorgenommen und Orientierungen mit unterschwelliger Amplitude ausgeblendet werden.
Somit haben wir die lokale Orientierung für einen Farbkanal berechnet. Dieses ist zusammengefaßt in Abbildung 6.5 dargestellt. Dieser Vorgang ist für die übrigen

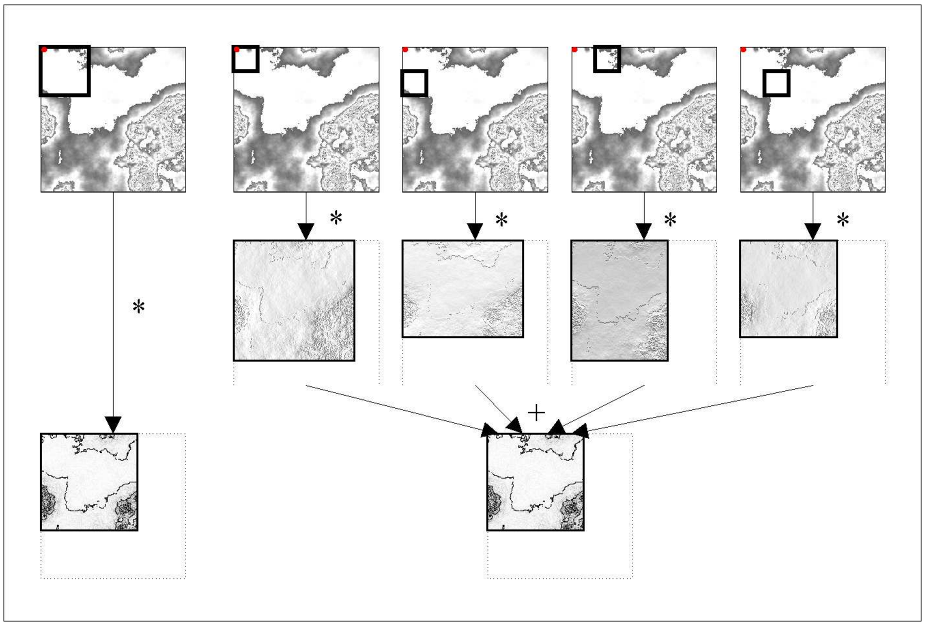


Abbildung 6.3: Zusammensetzen einer 16×16 Faltungsmaske durch vier einzelne 8×8 Masken mit verschobenen Aufpunkten. Rote Punkte stellen die Aufpunkte der Faltungsmasken dar.

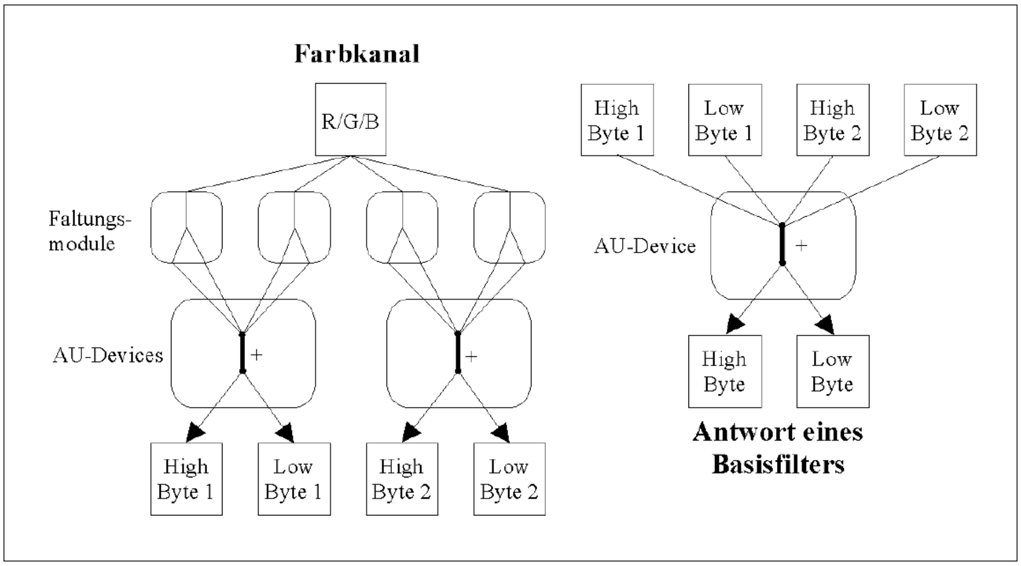


Abbildung 6.4: Pipes zur Berechung der Filterantwort eines Andersson-Basisfilters. Da in der vorliegenden Konfiguration nur 2 *AU-Devices* vorhanden sind, können die Ergebnisse der vier 8×8-Filterungen nicht in einer *Pipe* addiert werden. Eine Zwischenspeicherung ist notwendig.

zwei Kanäle zu wiederholen. Insgesamt werden 48 Faltungen, 42 Additionen und 3 *LUTs* benötigt. Daß die Laufzeit lediglich bei $0.09\,s$ (rechnerisch $0.025\,s$) liegt, demonstriert eindringlich die Stärken des Pipelinerechners.

Die beschriebenen *Pipes* und *PAT*s zur Andersson-Filterung sind definiert in den Files *anders.c* und *anders.h*.

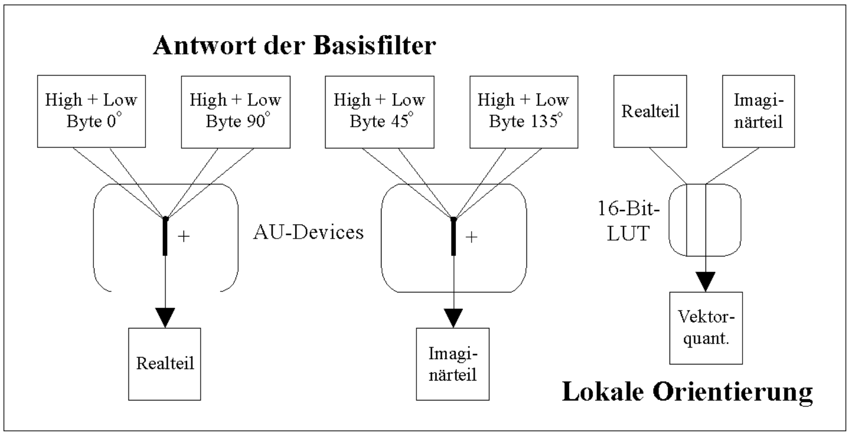


Abbildung 6.5: Pipes zur Berechung der lokalen Orientierung eines Farbkanals.

6.2.4 Rotation in die Normlage

In Abschnitt 5.4 haben wir ein Verfahren beschrieben, welches den Verlust der Rotationsinvarianz durch die Berechnung lokaler Orientierung für die Histogrammintersektion aufheben soll. Dieses Verfahren ist auf der MV200 implementiert und kann der Intersektion von Orientierungshistogrammen vorgeschaltet werden. Der Algorithmus besteht aus dem Finden von Schwerpunkten der Orientierungshistogramme der einzelnen Farbkanäle, woraus ein gemeinsamer Schwerpunkt berechnet wird. Das unbekannte Objekt im Einzugsfenster kann daraufhin in eine Normlage gedreht und das Identifikationsverfahren auf dieses 'rotierte' Objekt angewendet werden.
Zunächst werden die einzelnen Orientierungshistogramme der RGB-Farbkanäle benötigt. Dieser Schritt wird durch drei *Pipes* realisiert, die jeweils mit einem Statistik-Prozessor die Histogramme berechnen. Innerhalb eines *PATs* bestimmen wir die Schwerpunkte der drei Histogramme mittels *PAT*-Variablen und mitteln die Resultate zu einem Schwerpunkt.
Wir kennen somit die Abweichung zu der Normlage (0° oder Bildwert = 128) und müssen das Bild lediglich noch rotieren. Dies könnte mit einem *Mini-Warper* geschehen, den wir bereits bei der Skalierung des RGB-Eingangsbildes benutzt haben. Das Verhalten des *Warpers* wird durch mehrere Parameter gesteuert – neben Skalierungen ist dieser auch in der Lage, Rotationen um beliebige Winkel auszuführen. Das RGB-Bild könnte in die Normlage rotiert, die neuen lokalen Orientierungen berechnet und anschließend die Intersektion durchgeführt werden. Diese Methode hat jedoch zwei gravierende Nachteile:

1. Die Parameter des *Warpers* lassen sich nicht aus einem *PAT* heraus verändern, so daß eine adaptive Rotation je nach Lage des Objektes oder Neigung der Kamera nur durch den Host-Computer vorgenommen werden kann.

2. Die erneute Andersson-Filterung kostet noch einmal ca. $0.09\,s$ Rechenzeit.

Diese Probleme können umgangen werden, indem wir die Information der Orientierung in den vektorquantisierten *Surfaces* ausnutzen. Die Rotation des Bildes um

einen Winkel φ bedeutet für die Orientierungsbildpunkte die Addition von φ modulo dem Winkelbereich $[-\frac{\pi}{2}, \frac{\pi}{2}]$, was einer zyklischen Verschiebung des Histogrammes gleichkommt. Diese Addition kann auf der MV200 in einem *AU-Device*, die Modulo-Operation mittels einer *LUT* vorgenommen werden. Somit benötigen wir für die Rotation jedes Orientierungskanals eine *Pipe.* Auf das entstandene 3-kanalige Bild kann sofort die Histogrammbildung mit anschließender Intersektion angewendet werden.
Den zweistufigen Vorgang der Rotation haben wir graphisch in Abbildung 6.6 dargestellt.

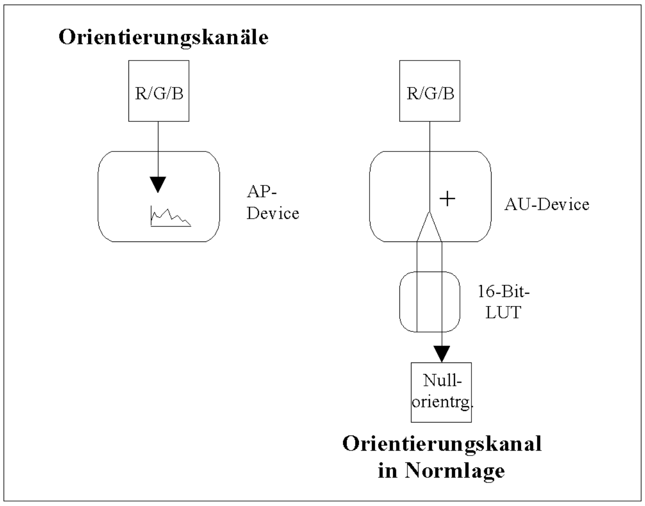


Abbildung 6.6: Pipes zur Drehung eines vektorquantisierten Bildes in eine Normlage.

Die soeben beschriebenen *Pipes* und *PATs* sind in den Files *rotate.c* und *rotate.h* definiert.

6.2.5 Berechnung der Rückprojektion

Erster Schritt der Rückprojektion ist die Bildung des 3D-Histogramms aus Abschnitt 6.2.2. Die Histogrammdaten werden aus den *AU*-Registern in *PAT*-Variablen gespeichert. In der Offline-Phase ist bereits das Histogramm des gesuchten Objektes in *PAT*-Variablen transferiert, so daß innerhalb eines *PATs* das Verhältnis-Histogramm berechnet werden kann. Da *PAT*-Variablen nur ganzzahlige Werte aufnehmen können, bilden wir den Wertebereich $[0, 1] \subseteq \mathbb{R}$ der Rückprojektion auf $\{0, \ldots, 255\} \subseteq \mathbb{N}_0$ ab. Die 512 Bins des Verhältnis-Histogramms werden in eine *LUT* geschrieben, durch die schließlich das 9-Bit-Zwischenergebnis der Histogrammberechnung geleitet wird. Wir erhalten damit die Rückprojektion, die eine Tiefe von 8 Bit hat.
Komplizierter ist die anschließende Glättung der Rückprojektion. Wir müssen dabei Boxfilter bis zu einer Größe von 40×100 realisieren. Eine bloße Zusammensetzung aus 8×8-Masken würde im obigen Fall $5 \cdot 13 = 65$ Faltungen erfordern, was ein

größerer Aufwand als die Andersson-Filterung wäre. Eine Rückprojektion auf Orientierungsbildern würde somit wenigstens $0.2\,s$ Rechenzeit benötigen.
Um dieses Problem zu lösen, greifen wir einerseits auf die Zerlegung von 16×16-Masken aus Abschnitt 6.5 und andererseits auf die konstanten Koeffizienten der Boxfilter zurück:

1. In einem ersten Schritt realisieren wir mit vier parallelen Faltungen und drei Additionen die Glättung mit einer 16×16-Maske.

2. Um eine in X-Richtung größere Maske zu simulieren, z.B. eine der Größe 32×16, könnten wir den ersten Schritt wiederholen (die *Alignment-Points* der einzelnen Masken müßten lediglich um 16 Bildpunkte nach links verschoben werden) und das Ergebnis mit dem aus Schritt 1 addieren. Wegen der Gleichheit der Koeffizienten der jeweiligen 16×16-Masken sind die Faltungsergebnisse bis auf eine Translation gleich. Diese Translation ergibt sich aus den unterschiedlichen *Alignment-Points* und ist in Abbildung 6.7 dargestellt.

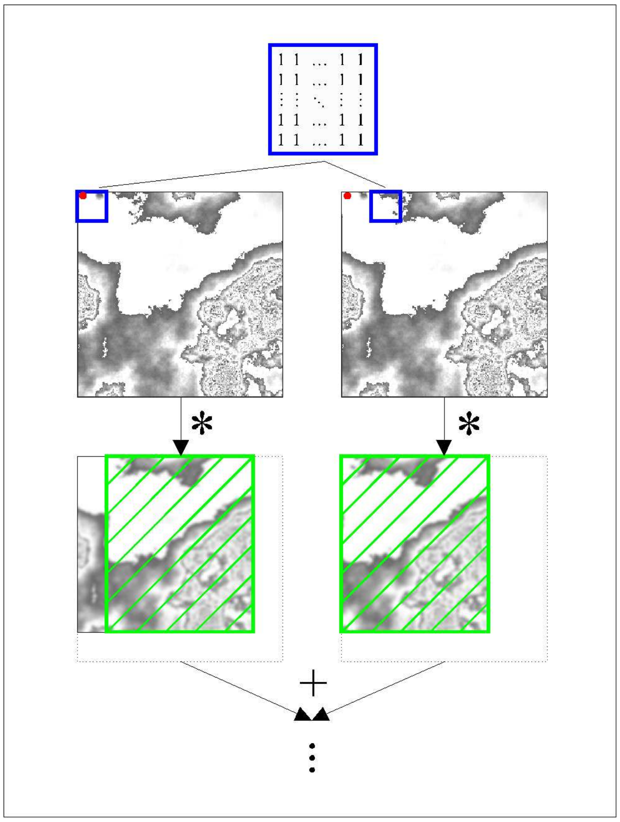


Abbildung 6.7: Filterantworten von Glättungsmasken, welche die gleichen Koeffizienten haben und sich lediglich im Aufpunkt unterscheiden. Dadurch sind verschobene Teile der Filterantworten (grün schraffiert) identisch.

Somit können wir das 1. Glättungsergebnis erneut verwenden, indem wir es mit einem verschobenen Teil seiner selbst addieren, und erhalten die Filterantwort eines 32×16-Filters. Führen wir eine derartige Addition auf diesem Ergebnis erneut durch – den 2. Summanden um 32 Bildpunkte nach links verschoben – , ergibt dies die Antwort eines 64×16-Filters; jede derartige Addition verdoppelt die Breite des simulierten Filters. Dabei muß die Filterbreite nicht immer verdoppelt werden: indem nicht die aktuellste Filterantwort sondern eine frühere als 2. Summand benutzt wird, können auch Zwischengrößen berechnet werden. Addieren wir z.B. auf die Antwort des 32×16-Filters die um 16 Punkte verschobene Antwort des 16×16-Filters, simuliert dies einen Filter der Abmessungen 48×16.

Berechnet man im 1. Schritt die Antwort eines kleineren Filters (z.B. 11×11) lassen sich mit dem obigen Verfahren weitere Zwischengrößen bestimmen. In unserer Implementierung benutzen wir die Masken-Dimensionen 11 bis 16 und können jedes Vielfache dieser Größe mit den obigen Additionen berechnen.

3. Hat die Maske in der X-Dimension die gewünschte Größe, läßt sich das gleiche Verfahren auf das Faltungsergebnis anwenden, um die Dimension in Y-Richtung zu vergrößern.

Damit sind wir in der Lage, Bilder mit sehr großen Masken in wenigen Schritten zu glätten: für die oben erwähnte 40×100-Maske benötigen wir lediglich 7 *Pipes*.
Zum Schluß verbleibt die Suche nach lokalen Maxima in der geglätteten Rückprojektion. Wir implementieren eine Suche nach dem absoluten Maximum, da die Ergebnisse der Rückprojektion auf den Orientierungsbildern bereits in der Vorstudie unter *Khoros II* sehr gut sind und die Suche nach weiteren lokalen Maxima nicht notwendig erscheint. Weil sich das 16-Bit-Glättungsergebnis in zwei *Surfaces* befindet, gehen wir in drei Schritten vor:

1. Bestimmung des maximalen Wertes in dem *Surface* mit den höherwertigen Bytes.

2. Ausblenden aller Bildpunkte in dem niederwertigen Byte, deren höherwertiger Teil nicht Maximum war.

3. Lokalisierung des absoluten Maximums in dem ausgeblendeten niederwertigen Byte.

Bei jedem Datentransfer in ein *Surface* berechnet der Pipelinerechner automatisch bestimmte Statistiken, u.a. auch den maximalen Wert des Surfaces. Somit ergibt sich der erste Schritt bereits mit Beendigung der letzten *Pipe* des Glättungsverfahrens. Der zweite Schritt läßt sich in einem *AU-Device* realisieren: ein *MOSC* wird mit dem höherwertigen Byte gefüllt und steuert einen Multiplikationsprozessor, der das niederwertige Byte mit 0 oder 1 multipliziert, je nachdem, ob das höherwertige Byte den maximalen Wert enthält oder nicht. In dem so ausgeblendeten niederwertigen Byte wird wiederum das Maximum ermittelt. Dieses Maximum wird daraufhin mit dem Statistik-Prozessor in einem *AU-Device* lokalisiert. Dieser kann programmiert

werden, in einem Surface die Koordinaten von Bildpunkten mit bestimmten Eigenschaften (hier das Maximum) zu finden. Sollten sich mehrere Maxima ergeben, wird das erstauftretende gewählt.
Das gesamte Verfahren (Histogrammberechnung, Rückprojektion, Glättung und Lokalisierung des Maximums) benötigt je nach Größe des gesuchten Objektes bis zu 0.036 *s*. Sämtliche Berechnungen und *Overlay-Graphiken*[1] finden innerhalb von *PATs* statt, lediglich das Schreiben der 512 Bins des Verhältnis-Histogramms in eine *LUT* wird von dem Host-Computer vorgenommen.
Die beschriebenen *Pipes* und *PATs* sind in den Files *backpro.c* und *backpro.h* definiert.

6.2.6 Berechnung der Intersektion

In einer Offline-Phase werden die Modell-Histogramme in Speichermodule des Pipelinerechners geschrieben. Da ein Histogrammeintrag leicht über dem Wert 255 liegen kann, benötigen wir zwei Surfaces, die das höher- und niederwertige Byte der Histogramme repräsentieren. In diese Surfaces werden nacheinander die 512 Bins sämtlicher Modell-Histogramme gespeichert.
In der Online-Phase ist zunächst das Histogramm des bereits lokalisierten Objektes zu bestimmen. Dieses geschieht genau wie bei der Rückprojektion in einem *PAT*. Anschließend wird durch den Host-Computer der Inhalt der *PAT*-Variablen, in denen das Histogramm untergebracht wurde, getrennt nach höher- und niederwertigen Bytes in zwei *Surfaces* transferiert. Dieser Vorgang ähnelt dem Schreiben des Verhältnis-Histogramms in eine *LUT* bei der Rückprojektion und kann nicht innerhalb eines *PATs* vorgenommen werden. Die *Rectangles* dieser *Surfaces* haben die Abmessungen 512×1, so daß bei der anschließenden Pipe genau ein Modellhistogramm in den Schnitt einbezogen wird.
Um den Histogrammschnitt eines Modells und des unbekannten Objektes zu berechnen (siehe (2.7) auf Seite 14), wird eine *Multi-Source-Multi-Destination-Pipe* jeweils von den zwei Histogramm-*Surfaces* in ein *AU-Device* geleitet. Weil die *ALU*-Prozessoren nur einen 10-Bit-Datenstrom verarbeiten können, bestimmen wir vorher das Minimum des höherwertigen Bytes in einer *LUT*. Dieses Ergebnis wird zum einen in einem Surface gespeichert (und dort automatisch die Summe berechnet) und steuert zum anderen über einen *MOSC* das Verhalten eines *ALU*-Prozessors, durch den die beiden niederwertigen Bytes fließen. Der Prozessor filtert das Minimum aus den beiden Datenströmen heraus, dessen Summe am Ende der *Pipe* wiederum automatisch berechnet wird. Wir können beide Summationsergebnisse gewichtet addieren und erhalten den Intersektionswert.
Der Gesamtvorgang der Intersektion ist in Abbildung 6.8 dargestellt. In einer *Pipe* wird ein Histogrammschnitt berechnet. Da in der Konfiguration der MV200 sowohl zwei 16-Bit-*LUTs* als auch zwei *AU-Devices* vorhanden sind, wäre es möglich gewesen, parallel zwei Schnitte zu bestimmen. Wir haben in dieser Implementierung

[1]Die Position eines lokalisierten Objektes wird durch ein rotes Kreuz, das über das RGB-Bild gelegt wird, angezeigt; daher die Bezeichnung *Overlay*.

aus zwei Gründen davon abgesehen: erstens ist unsere Datenbank noch sehr klein ist und zweitens sind die *Pipelaufzeiten* von *Surfaces* mit 512 Bildpunkten so gering, daß die vorangehende Histogrammbildung und die Intersektion mit allen Modellhistogrammen nur $0.0122\,s$ (rechnerisch $0.005\,s$) dauert.

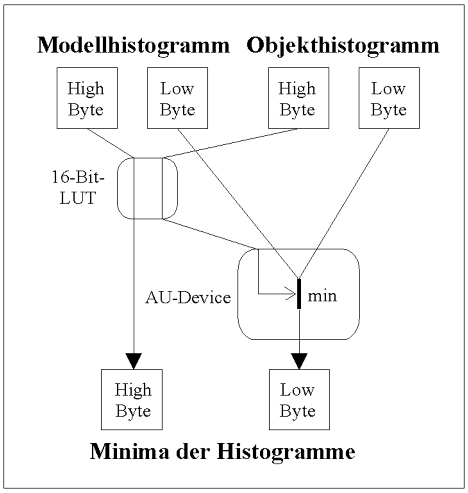


Abbildung 6.8: *Pipe* zur Berechnung des Schnittes von Modell- und Objekthistogramm.

Um das Objekt mit sämtlichen n Modellhistogrammen zu schneiden, ist die *Pipe* n-mal auszuführen, wobei der *Alignment-Point* des Surfaces der Modellhistogramme jeweils auf das nächste Histogramm verschoben werden muß. Dazu wird die *Pipe* innerhalb eines *PATs* n-mal gefeuert, die jeweiligen Intersektionsergebnisse in *PAT*-Variablen gespeichert, um zum Schluß den maximalen Wert zu ermitteln.
Das Ergebnis der Intersektion (die Bezeichnung des identifizierten Objektes) wird mit *Overlay*-Graphiken auf dem Bildschirm angezeigt. Die Intersektion von Orientierungshistogrammen ist ein wenig schneller als die von RGB-Histogrammen, da bei Vorliegen eines kleinen Objektes der Histogrammschnitt mit großen Objekten (und umgekehrt) gar nicht bestimmt wird, wodurch weniger *Pipes gefeuert* werden. Da bei Bildung von Orientierungshistogrammen jedoch eine *Pipe* mehr auszuführen ist als im RGB-Fall (siehe Abschnitt 6.2.2), hebt sich dieser Geschwindigkeitsvorteil wieder auf. Sämtliche Berechnungen und *Overlay*-Graphiken finden innerhalb von *PATs* statt; lediglich der Transfer des Objekthistogramms in die Speichermodule wird von dem Host-Computer vorgenommen.
Die beschriebenen *Pipes* und dazugehörigen *PATs* finden sich wieder in den Files *inter.c* und *inter.h*.

6.2.7 Weitere Programmabschnitte

Die folgenden Files sind in der obigen Beschreibung nicht vorgekommen, sollten aber der Vollständigkeit halber genannt und kurz beschrieben werden:

main.c und main.h: Definition und Initialisierung sämtlicher globaler Variablen.

menus.c, menus.h, keycontrol.c und key.h: Funktionen zur Steuerung des Programmes durch Menüs.

patcontrol.c: Steuerung aller zeitkritischen Algorithmen durch Kontroll-*PATs*.

ovly.c und ovly.h: Overlay-Grafiken für die Anzeige der Lokalisierung und Identifikation von Objekten.

disp.c: Definition der *Pipe* zur Anzeige verschiedener Inhalte von *Surfaces* auf einem Monitor.

cut.c, cut.h und database.c: Funktionen zur Bildung und Verwaltung von Histogrammdatenbanken.

func.c und fun.c: Funktionen, die keiner speziellen Kategorie zugeordnet werden konnten.

6.3 Bewertung

Da wir die verschiedenen Algorithmen bisher nur für sich alleine betrachtet haben, wollen wir an dieser Stelle einen Überblick über das Laufzeitverhalten sämtlicher, evtl. gekoppelter Verfahren geben. Auf dem Pipelinerechner gibt es die Möglichkeit, insgesamt vier *Pipes* gleichzeitig parallel laufen zu lassen, wodurch auch verschiedene Algorithmen parallel abgearbeitet werden können. Dies bietet sich insbesondere schon bei der Digitalisierung mit anschließender Skalierung an, welche sequentiell eine Laufzeit von $0.13\,s$ hätten. In unserer Echtzeitimplementierung gibt es die Auswahl, bestimmte Verfahren allein oder gekoppelt mit anderen zu starten, so z.B. die Rückprojektion zusammen mit der Intersektion auf RGB-Bildern, oder nur die Intersektion auf den vektorquantisierten Eingabebildern. In Abbildung 6.9 haben wir das Laufzeitverhalten der einzelnen Algorithmen und verschiedener Kombinationsvarianten graphisch dargestellt.
Da ständig eine *Pipe* zur Darstellung verschiedener *Surface*-Inhalte auf einem Monitor aktiv ist, können maximal drei parallele Verarbeitungen erfolgen, was bei allen Kombinationen ausgenutzt wird.
Wie aus Abbildung 6.9 hervorgeht, sind die Andersson-Filterung und die Digitalisierung die zeitaufwendigsten Verfahren. Die Lokalisierung und Identifikation sind bis zu einem Faktor 3 schneller. Dies hat mehrere Gründe:

1. Die MV200 ist ausgelegt für eine Verarbeitung von Grauwertbildern. Bei mehrkanaligen Farbbildern werden schnell die Grenzen des Pipelinerechners offengelegt: Da oftmals die gleichen Verarbeitungsschritte auf alle Farbkanäle angewendet werden müssen, sollte jedes *Device* in dreifacher Ausfertigung vorhanden sein, um bereits hier die Vorteile der parallelen Ausführung nutzen zu können. Dies kann nicht geschehen, da grundsätzlich 'nur' eine doppelte Ausführung der *Devices* vorliegt. Eine Verarbeitung von Farbbildern hat somit i.a. die dreifache Laufzeit im Vergleich zur Grauwertverarbeitung.

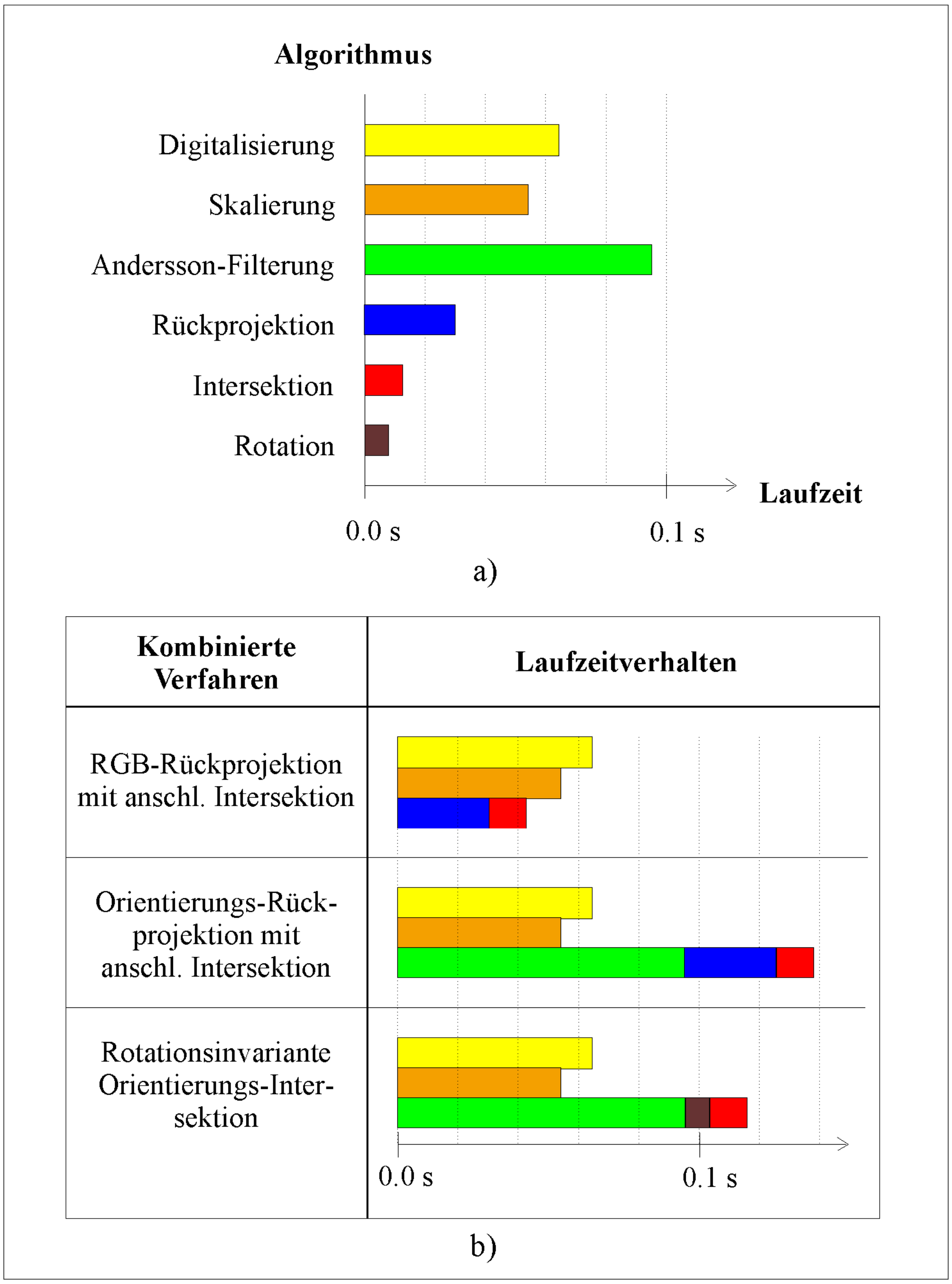


Abbildung 6.9: Laufzeitverhalten der implementierten Algorithmen auf der MV200. a) Die jeweiligen Verfahren für sich betrachtet. b) Kombinationsmöglichkeiten der Algorithmen. Parallele Verarbeitung ist durch vertikale Anordnung gekennzeichnet, sequentielle Verarbeitung durch horizontaleAnordnung.

2. Selbst für den Fall, daß alle *Devices* in dreifacher Ausfertigung vorliegen, wird die parallele Verarbeitung nur stark eingeschränkt möglich sein, bedingt durch die maximale Anzahl von vier gleichzeitig aktiven *Pipes*. Bei paralleler Abarbeitung der Digitalisierung und Skalierung stehen keine drei *Pipes* mehr zur Verfügung, um Verfahren auf mehreren Farbkanälen gleichzeitig zu starten.

3. Die vorliegende Konfiguration hat insgesamt 12 Speichermodule auf zwei *AB-Devices*. Davon sind bereits 7 durch die Digitalisierung und Darstellung auf dem Monitor belegt und können von anderen *Pipes* nicht mehr benutzt werden. Die verbleibenden 5 Module reichen gerade für unsere Anwendungen aus: allein bei der Filterung eines Farbkanals mit einem Basisfilter sind 4 *Surfaces* zum Zwischenspeichern der Filterantwort nötig. Ähnliches ergibt sich bei der Rückprojektion. Obwohl noch 8 weitere Faltungsmodule zur Verfügung stehen, können wir diese nicht nutzten, da die Ergebnisse der Faltungen nicht gespeichert werden können. Eine größere Anzahl von Speichermodulen würde eine Halbierung der Laufzeit der Andersson-Filterung auf $0.05\,s$ bewirken.

Dennoch bietet der Pipelinerechner Möglichkeiten, die normale Workstations oder PCs nicht ereichen. Massive Parallelisierung und Aufsplittung in Kleinstprozessoren sind hervorragende Voraussetzungen für echtzeitfähige Implementierungen.

Kapitel 7

Experimente

Wir testen die Verfahren von Swain und Ballard, einige Weiterentwicklungen sowie unseren Algorithmus zur Farbkonstanz anhand einer Datenbank aus 15 Objekten. Die lokale Histogrammbildung ist zwar implementiert, wird jedoch keinen umfangreichen Experimenten unterzogen, da eine Koppelung der Rückprojektion und Intersektion dem Verfahren gleich kommt (siehe Abschnitt 3.2). Die Versuchsobjekte unterscheiden sich z.T. in Farbe, Form und Größe, einige Objekte sind sich in diesen Eigenschaften aber auch ähnlich. In Anhang B sind sämtliche Modelle abgebildet.
Der Umfang von nur 15 Objekten scheint ein wenig gering, um statistisch aussagekräftige Ergebnisse liefern zu können. Wir müssen jedoch durch die Vielzahl der Verfahren und unterschiedlichen Testsituationen Einschränkungen vornehmen, um die Zahl der Experimente in Grenzen zu halten. Der Vergleich zu der Datenbank von Swain und Ballard, welche 66 Histogramme umfaßt, fällt nicht so schlecht aus, wenn man bedenkt, daß 33 Objekte durch je zwei Histogramme von verschiedenen Ansichten repräsentiert werden, die Datenbank also etwas mehr als die doppelte Anzahl von Objekten enthält wie unsere. Aus statistischen Werten wie der Toleranz lassen sich die Ergebnisse für eine größere Datenbank abschätzen.
Im folgenden beschreiben wir kurz die Entwicklungsumgebungen, unter denen die Algorithmen implementiert worden sind, die Versuchsaufbauten sowie ausführlich die Ergebnisse sämtlicher Verfahren unter verschiedenen Testsituationen.

7.1 Herkunft der Resultate

In einer Vorstudie sind die obigen Verfahren in Form von *Khoros II* -Programmen unter Unix implementiert und in *Cantata-Workspaces* integriert worden. *Khoros II* ist eine Softwareentwicklungsumgebung, welche sich zusammen mit der visuellen Programmiersprache *Cantata* sehr gut eignet, Algorithmen der Computer Vision in modularartiger Form zu entwickeln, auszutesten und zu erweitern. Dabei können bereits vorhandene *Kohros II* -Resourcen wie z.B. die 2D-Fouriertransformation, komplexe Operatoren oder Programme zur Visualisierung von Daten aller Art genutzt und problemlos mit eigenen Programmen verknüpft werden, die wiederum

anderen Entwicklern zur Verfügung stehen. Der Datenfluß zwischen verknüpften Programmen wird übersichtlich in Form von direkten Graphen dargestellt.
Obwohl sich dieses System zum Entwickeln neuer Algorithmen sehr gut eignet, ist das ausgiebige Testen unter *Khoros II* etwas mühsam. So benötigt die Andersson-Filterung (im Orts- oder Frequenzraum) eines 256×256-Eingangsbildes benötigt unter *Khoros II* auf eine Sun Sparcstation 10 über zwei Minuten. Da in der Vorstudie die Ergebnisse der Rückprojektion und der Intersektion sowohl mit als auch ohne Vorschaltung des Farbkonstanzalgorithmus' wesentlich besser sind als die der übrigen Verfahren, haben wir neben der lokalen Orientierungsberechnung auch nur die Swain'schen Algorithmen auf dem Pipelinerechner MV200 implementiert. Lediglich die Objektlokalisierung mittels auftretender Wahrscheinlichkeiten von Schiele und Crowley, welche ein Spezialfall der Rückprojektion ist (siehe Abschnitt 3.1), kann nach der Implementierung der Rückprojektion ohne großen Aufwand auch auf der MV200 realisiert werden. Somit ergeben sich die Testergebnisse der Rückprojektion und Intersektion aus dem Programm *orientation_indexing* auf der MV200, alle anderen Resultate werden mit *Khoros II* -Programmen erzielt.

7.2 Versuchsaufbau

Eine Kamera vom Typ DCX-151A der Firma Sony, die an das *Digicolor-Board* der MV200 angeschlossen ist, liefert Farbbilder im RGB-Format. Die Kamera ist in ca. $50\,cm$ Höhe auf einem Stativ befestigt und ausgerichet auf einen einfarbigen Photokarton, welcher sich in ca. $4.60\,m$ Entfernung auf dem Boden befindet. Der Photokarton nimmt fast den gesamten Bildbereich der Kamera ein und bewirkt, daß keine störenden Hintergrundfarben oder -kanten in den Testszenen erscheinen. Auf diesem Karton können mehrere Objekte neben- oder hintereinander plaziert werden.
Der Aufnahmeraum ist grundsätzlich nur durch die eingeschaltete Deckenbeleuchtung und zwei Scheinwerfer beleuchtet, die Fenster werden durch Vorhänge verdunkelt. Da die Farbkonstanz ein Hauptaspekt dieser Arbeit ist, stellen wir mit den Scheinwerfern, welche mit durchsichtigen Folien der Farben Rot, Grün, Blau und Weiß versehen werden können, insgesamt sechs verschiedene Beleuchtungssituationen her:

normal: Die Scheinwerfer liefern weiches, indirektes Licht. Unter diesen Verhältnissen werden die Modelle der Datenbank aufgenommen.

hell: Die Objekte werden von den Scheinwerfern direkt angestrahlt, wodurch Schlagschatten entstehen.

dunkel: Die Scheinwerfer sind ausgeschaltet.

rot: Vor den Scheinwerfern werden rote Folien befestigt, die Objekte direkt angestrahlt. Wieder entstehen Schlagschatten.

grün: Wie vorhergehende Versuchsdurchführung, jedoch Verwendung von grüner Folie.

blau: Verwendung von blauer Folie.

Die Resultate der Rückprojektion und Intersektion sind sofort auf dem Farbmonitor, der an der MV200 angeschlossen ist, abzulesen. Für die übrigen Algorithmen, die nur unter *Khoros II* implementiert sind, müssen zunächst Modellbilder und Szenen aufgenommen und anschließend in *Cantata-Workspaces* eingebunden werden.

7.3 Resultate der Objektlokalisierung

Um die Lokalisierung ausgiebig zu testen, stellen wir fünf verschiedene Szenen zusammen, in denen sich jeweils vier Objekte unserer Datenbank nebeneinander mit unterschiedlichen Abständen zueinander in dem Aufnahmebereich der Kamera befinden. Jede Szene wird unter sämtlichen Beleuchtungsvariationen auf die Lokalisierung der darin vorkommenden Objekte untersucht. Diese Szenen sind in Anhang C abgebildet. Da die Resultate der Rückprojektion und der Methode von Schiele und Crowley identisch sind, stellen wir nur die Ergebnisse der Rückprojektion dar.

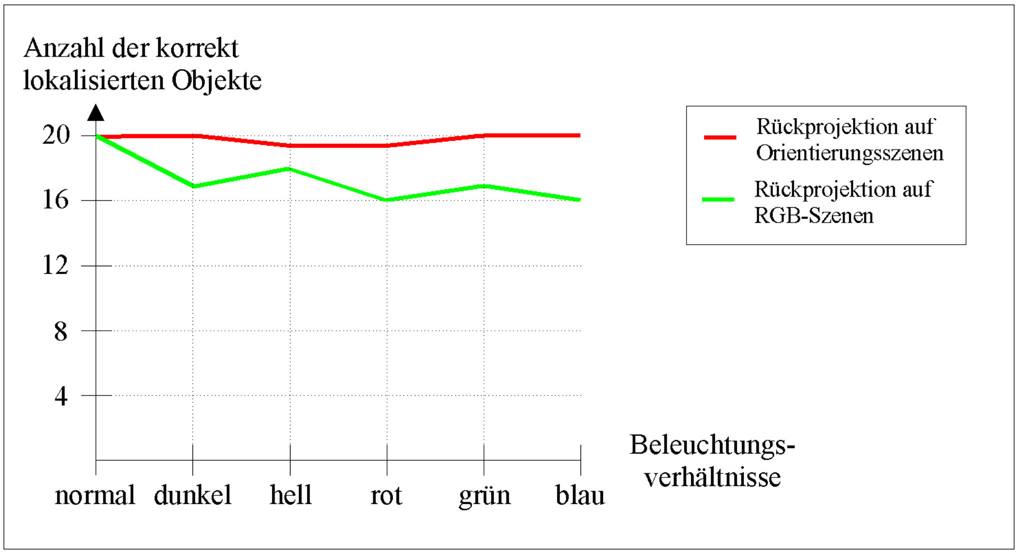


Abbildung 7.1: Statistik über die Ergebnisse der Rückprojektion unter sämtlichen Beleuchtungsverhältnissen. Pro Beleuchtung sind maximal 20 Objekte zu lokalisieren. Angezeigt werden die Anzahl der korrekt lokalisierten Objekte für das Verfahren auf Farb- und Orientierungshistogrammen.

Die Lokalisierung mittels Farbhistogrammen funktioniert bei den farbig beleuchteten Szenen wie zu erwarten nicht sehr gut: bis zu 20% der Objekte werden nicht korrekt gefunden. Helligkeitsschwankungen hingegen können durch die grobe Quantisierung bei der Bildung der 3D-Histogramme noch abgefangen werden. Die relativen Maxima der Rückprojektion sinken auch in diesen Situationen deutlich, jedoch ist die Höhe

der Maxima kein Garant für das tatsächliche Vorkommen eines Objektes (siehe Abschnitt 2.1.2.3) und insofern nicht ausschlaggebend.
Die Resultate der Rückprojektion mit vorgeschaltetem Farbkonstanzalgorithmus sind wesentlich besser: lediglich ein Objekt wird jeweils in den hellen und roten Szenen nicht an erster Stelle erkannt. Interessant ist hierbei auch die Genauigkeit der Zentrumsermittlung eines Objektes. Durch die Quantisierung bei der Farbhistogrammbildung fallen teilweise Objektfarben in die gleiche Bins, welche auch die Hintergrundfarben enthalten, so daß der Hintergrund in der Rückprojektion nicht immer den erwünschten Wert 0 hat. Somit wirkt sich auch der Hintergrund in der anschließenden Glättung aus, und das Objektzentrum wird nicht immer exakt gefunden. Zusammen mit dem Kamerarauschen ergibt dies in der Echtzeitimplementierung ein Zentrum, welches z.T. stark um das tatsächliche Zentrum des gesuchten Objektes 'rotiert'. Da der Hintergrund in den Orientierungsbildern grundsätzlich der Nullorientierung zugeordnet und diese in den Orientierungshistogrammen vollständig vernachlässigt wird, ist das ermittelte Zentrum in Orientierungsszenen wesentlich stabiler und daher geringeren Schwankungen unterworfen.

7.4 Resultate der Algorithmen zur Objektidentifikation

7.4.1 Ähnlichkeit durch Momente

Stricker et. al. wählen für ihre Versuche eine Datenbank mit 3000 Bildern, welche alle die gleichen Abmessungen besitzen, und testen zwei unbekannte Bilder u.a. mit der Histogramm-Intersektion und ihrer Ähnlichkeitsfunktion. Die Ergebnisse des Histogrammschnittes sind nicht so gut wie die der Ähnlichkeitsfunktion. Wir testen den Algorithmus mit unseren normal beleuchteten Objekten und stellen fest, daß die Resultate der Intersektion (siehe später Tabelle 7.3 auf Seite 82) wesentlich besser als die der Ähnlichkeitsfunktion sind. Da die Beleuchtung von uns kontrolliert wird und die Experimente im RGB-Raum durchgeführt werden, wählen wir als Gewichte der Ähnlichkeitsfunktion $\omega_{ij} = 1$ für $i, j \in \{1, ..., 3\}$. Tabelle 7.1 belegt die Ergebnisse.
Nur 4 Objekte werden korrekt identifiziert, 8 Objekte haben einen Rang ≥ 8. Bei diesem Verfahren zeigt sich eine starke Anfälligkeit gegenüber Beleuchtungsschwankungen, obwohl wir die Objekte dieses Experimentes unter den gleichen Bedingungen aufgenommen haben wie die Modelle der Datenbank. 'Gleiche Bedingungen' bedeutet hierbei, daß wir versucht haben, die Beleuchtungsbedingungen wie zur Modellaufnahme herzustellen. Dies ist uns jedoch nicht immer gelungen. Die Verdunkelung der Fenster durch Vorhänge kann bei starker Sonnenstrahlung zu bestimmten Zeiten eine zusätzliche Beleuchtung nicht vollständig verhindern. Ferner bewirkt auch das erneute Aufstellen der Scheinwerfer an verschiedenen Tagen durch Positionsschwankungen veränderte Beleuchtungssituationen. Zusammen mit der Schwäche des Verfahrens, Bildausschnitte mit Objekten korrekt zu bewerten, in denen die Objekte nur einen kleinen Teil ausmachen (siehe Abschnitt 3.4.2.2), ergibt dies für

Objekt	**Rang**	d
7Up	11	145.55
8x4	6	36.38
Ball	7	35.14
Claust	7	55.54
Coke03	3	65.03
Coke05	2	13.21
Flens	6	42.89
Iso	12	72.30
Kinks	12	147.55
Locher	1	15.86
SunOr	7	45.12
SunZi	11	141.55
Target	1	258.16
Tesa	1	32.61
Teekanne	1	40.67

Tabelle 7.1: Tests der Ähnlichkeitsfuntion unter normaler Beleuchtung. Die Werte der Ähnlichkeitsfunktion d aus (3.16) zeigen den Abstand des Objektes vom zugehörigen Modell an. Nur selten wird ein Objekt korrekt identifiziert (Rang = 1).

unsere Experimente nicht überragende Resultate.

7.4.2 Intersektion auf Farbdifferenzen

Funt und Finlayson testen die Farbkonstanz ihres Verfahren an 60 künstlichen Farbobjekten unter verschiedenen Farbtemperaturen und belegen darin eine Trefferquote von 1 (!), d.h., alle Objekte werden unter allen verschiedenen Farbtemperaturen richtig erkannt.
Wir testen den Farbkonstanzalgorithmus mit sämtlichen, unter verschiedenen Beleuchtungen aufgenommenen Objekten. Dabei benutzen wir einen 5×5-Laplace-Filter zur Ableitung. Die Ergebnisse der ausgewerteten Ränge ist in Tabelle 7.2 dargestellt.

Beleuchtung	**Zuordnungen mit Rang = 1**	**Zuordnungen mit Rang = 2**	**Zuordnungen mit Rang ≥ 3**
normal	13	2	0
dunkel	5	5	5
hell	12	3	0
rot	5	3	7
grün	7	6	2
blau	6	5	4

Tabelle 7.2: Ränge aller Objekte unter allen Beleuchungsvariationen der Funt&Finlayson-Methode mit einem 5×5-Laplace-Filter.

Die Einträge der letzten 3 Zeilen belegen, daß der Algorithmus mit unseren realen

Objekten unter verschiedener spektraler Beleuchtung nicht gut zurechtkommt. Die durchschnittliche Trefferquote liegt unter 0.6; ob das richtige Objekt erkannt wird, ist Zufall.

7.4.3 Intersektion auf RGB-Histogrammen

Bei der Bildung unserer RGB-Datenbank schneiden wir die Objekte vor der Berechnung der Histogramme grundsätzlich quadratisch aus. Da die Mehrzahl unserer Objekte von quadratischer Form sind, kommt dies einer Segmentierung vom Hintergrund gleich. Lediglich bei 4 Objekten verändern wir nachträglich die Histogramme, um Farben des Hintergrundes zu entfernen. Bei den 'unbekannten' Objekten während der Online-Versuche ist dies nicht möglich. Da in unserer Datenbank Objekte unterschiedlicher Größe enthalten sind, bilden wir die Histogramme der zu identifizierenden Objekte über einen Bildausschnitt, der die ungefähren Abmessungen des größten Objektes hat.
Wir testen die Intersektion unter den 6 unterschiedlichen Lichtverhältnissen, unter denen auch die vorherigen Versuche stattfinden. Die Ergebnisse für normale Lichtverhältnisse sind in Tabelle 7.3 ausführlich dargestellt.

Objekt	**Rang**	**Inters.-Wert**	**Toleranz**
7Up	1	0.831	0.328
8x4	1	0.828	0.246
Ball	1	0.777	0.352
Claust	1	0.926	0.239
Coke03	2	0.854	0.100
Coke05	1	0.891	0.032
Flens	1	0.861	0.040
Iso	1	0.823	0.353
Kinks	1	0.848	0.281
Locher	1	0.897	0.215
SunOr	1	0.830	0.117
SunZi	1	0.856	0.291
Scheibe	1	0.883	0.410
Tesa	1	0.836	0.381
Teekanne	1	0.906	0.128

Tabelle 7.3: Tests der Intersektion unter normaler Beleuchtung auf RGB-Histogrammen. 14 von 15 Objekten werden korrekt erkannt (Rang= 1). Der 'ideale' Intersektionswert von 1 wird von keinem Objekt erreicht. Toleranzwerte belegen, welchen Abstand im Histogrammraum das Objekt vom nächst-ähnlichen Modell hat.

Bis auf ein Objekt werden alle anderen korrekt erkannt, die Intersektions-Werte der richtigen Matches liegen dabei zwischen 0.926 und 0.777. Das negative Ergebnis des Objektes *Coke03* läßt sich durch die unterschiedlichen Größen der Modelle in unserer Datenbank erklären (siehe Abschnitt 2.2.2.4). Zugeordnet wird in diesem

Fall das Modell *Tesa*, das sehr klein ist und sich fast ausschließlich aus einem rötlichen Farbton zusammensetzt, der vermehrt auch in dem Modell *Coke03* auftaucht. Deutlich ist an niedrigen Toleranzwerten zu erkennen, daß einige Objekte im Histogrammraum dicht zusammenliegen, so ist z.B. der Abstand des Objektes *Coke05* zum Objekt *Coke03* sehr gering, das gleiche gilt für die Objekte *Flens* und *Claust*.
In Abbildung 7.2 haben wir die Resultate unter sämtlichen Beleuchtungsverhältnissen graphisch dargestellt, da bei der Vielzahl der Einträge eine Tabelle keine Übersicht mehr bietet. Bei den wenig und stark beleuchteten Objekten sinkt der durchschnittliche Intersektionswert etwas, bei farbiger Beleuchtung sogar unter 0.4. Obwohl das eine oder andere Objekt noch einen Rang von 1 hat, wird unter diesen Bedingungen wegen der niedrigen Intersektionswerte keines (!) der 15 Objekte mehr erkannt, was sich in der roten Kurve wiederspiegelt.

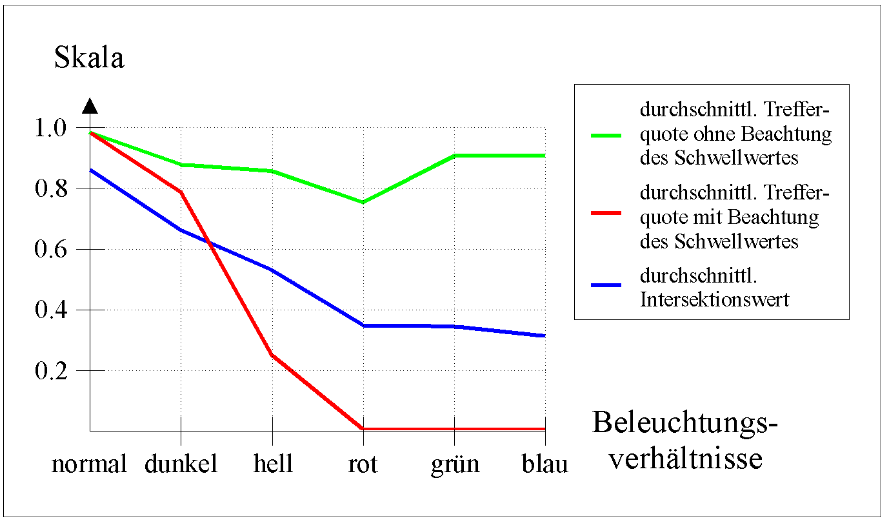


Abbildung 7.2: Statistik über die Ergebnisse der Intersektion auf RGB-Histogrammen unter sämtlichen Beleuchtungsverhältnissen. Der durchschnittliche Intersektionswert sinkt bei den farbig beleuchteten Szenen deutlich unter 0.6. Interessant ist die Diskrepanz der durchschnittlichen Trefferquoten mit und ohne Beachtung des Schwellwertes von 0.6, der zur Ausgrenzung von fremden Objekten und des Hintergrundes notwendig ist. Obwohl ein Objekt einen hohen Rang (z.B. 1) haben kann, wird es nicht erkannt, da der Intersektionswert geringer als 0.6 ist.

Wir testen ebenfalls unter normalen Bedingungen die Rotationsinvarianz der Intersektion, indem die zu identifizierenden Objekte (sofern möglich) um 90° gekippt werden. Unter diesen Voraussetzungen verändern sich die Farbsignale so stark, daß 4 Objekte, die in der vorherigen Testsituation einen Rang von 1 haben, nun nicht mehr korrekt erkannt werden.

7.4.4 Intersektion auf Orientierungshistogrammen

Bei dem Vergleich von Orientierungshistogrammen unserer Objekte haben wir neben dem Intersektionswert zusätzlich die Information über die Anzahl von Bildpunkten mit überschwelliger Amplitude, woraus wir auf die Größe von Objekten schließen.

Somit wird ein Objekthistogramm grundsätzlich nur mit den Modellhistogrammen geschnitten, welche – mit einem gewissen Spielraum – die gleiche Größe haben wir das unbekannte Objekt.
Um abzuschätzen, wie dicht die Orientierungshistogramme verglichen mit den RGB-Histogrammen der Modelle im Histogrammraum liegen, führen wie die Resultate der normal-beleuchteten Objekte ebenso ausführlich auf wie im vorherigen Abschnitt.

Objekt	**Rang**	**Inters.-Wert**	**Toleranz**
7Up	1	0.839	0.290
8x4	1	0.738	0.159
Ball	1	0.823	0.448
Claust	1	0.906	0.410
Coke03	1	0.867	0.111
Coke05	1	0.875	0.162
Flens	1	0.884	0.032
Iso	1	0.840	0.313
Kinks	1	0.804	0.183
Locher	1	0.880	0.250
SunOr	1	0.708	0.300
SunZi	1	0.835	0.200
Scheibe	1	0.879	0.221
Tesa	1	0.730	0.229
Teekanne	1	0.925	0.290

Tabelle 7.4: Tests der Intersektion unter normaler Beleuchtung auf den Orientierungshistogrammen. Alle 15 Objekte werden korrekt erkannt (Rang=1).

Die durchschnittliche Toleranz liegt mit 0.2399 in der gleichen Größenordnung wie die Toleranz 0.2324 der RGB-Histogramme. Es gibt auch hier Objekte, die sich durch den Histogrammschnitt nicht sehr gut unterscheiden lassen (z.B. die Modelle *Claust* und *Flens*); das Problem der unterschiedlichen Objektgrößen stellt sich nicht mehr.
Unser Algorithmus zur Farbkonstanz zeigt sich sehr robust unter den hergestellten, z.T. krassen Beleuchtungssituationen. Die Gesamtergebnisse sind dargestellt in Abbildung 7.3. Das Problem der gesättigten Pixel wird vollständig ignoriert und tritt selbst bei den farbig-beleuchteten Objekten nicht auf. Damit erhalten wir fast unter allen Beleuchtungsvariationen eine Trefferquote von 1. Unter der roten Beleuchtung ergibt sich bei drei Objekten ein 'Chamäleon-Effekt': der eigentlich graue Photokarton nimmt einen hellen rötlichen Farbton an, so daß sich die Objekte *Ball*, *SunOr* und *Tesa* nur noch schwach vom Hintergrund abheben. Dadurch werden die Amplituden der Konturpunkte extrem niedrig, was zu einer Fehlidentifikation dieser Objekte führt.
Die Rückgewinnung der Rotationsinvarianz durch Verschiebung der Orientierungshistogramme in den Histogrammschwerpunkt vermindert die Farbkonstanz des Verfahrens nicht. Der Erfolg der Schwerpunktberechnung hängt von den jeweiligen Objekten ab. Besitzt ein Objekt einen ausgeprägten Schwerpunkt, erfolgt auch eine

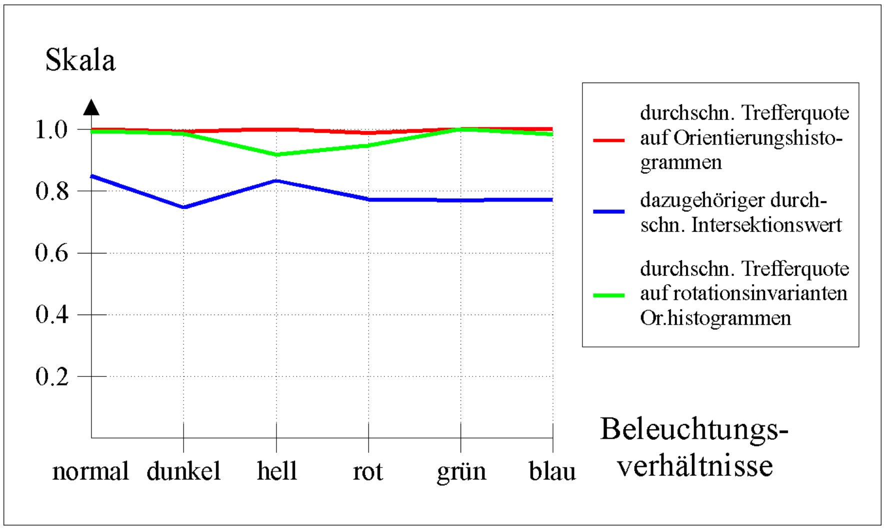


Abbildung 7.3: Statistik über die Ergebnisse der Intersektion unter sämtlichen Beleuchtungsverhältnissen mit Vorschaltung des Farbkonstanzalgorithmus'. Der durchschnittliche Intersektionswert bleibt bei allen Beleuchtungen über 0.7. Zusätzlich aufgeführt sind die Resultate der Intersektion auf Orientierungshistogrammen, die zur Rückgewinnung der Rotationsinvarianz in den Histogrammschwerpunkt verschoben sind.

korrekte Berechnung dieses Wertes. Bei anderen Objekten hingegen, die keinen ausgeprägten Schwerpunkt haben, wie das Objekt *Ball*, sind die Orientierungen auf den einzelnen Farbkanälen gleichverteilt. Es macht dann keinen Unterschied, ob das Orientierungshistogramm in die Normlage verschoben wird oder nicht — die Intersektion liefert mit oder ohne Verschiebung gleiche Ergebnisse. Einzig bei Objekten, deren Histogramme mehrere gleichausgeprägte Schwerpunkte besitzen, ist das Verfahren der Schwerpunktberechnung nicht erfolgreich. Durch Kamerarauschen, welches in begrenztem Umfang auch die Orientierungsberechnung beeinflußt, wird nicht deterministisch mal der eine oder andere Schwerpunkt gefunden. Je nachdem, ob dies der Schwerpunkt ist, welcher für das Modellhistogramm zugrundegelegt ist, wird das richtige Objekt identifiziert oder nicht. Dieses Phänomen tritt bei den Modellen *Locher* und *Kinks* auf, die wir daraufhin für die Experimente zur Rotationsinvarianz kurzzeitig aus der Datenbank entfernen.
Untersucht haben wir weiterhin, wie sich Entfernungsänderungen der Objekte zur Kamera auf das Verfahren ohne Rotationsinvarianz auswirken. Durch Veränderungen in diesem Parameter fallen Ereignisse in andere Skalen, was die Orientierungsberechnung beeinflußt. Da ferner die Größe eines Objektes zur Identifikation beiträgt, sinkt zwangsweise auch die durchschnittliche Trefferquote. Wir testen dieses Verhalten unter normaler Bedingung und variieren dabei die Entfernung Kamera/Objekte um $\pm 2\ m$. Die Resultate sind in Abbildung 7.4 dargestellt.

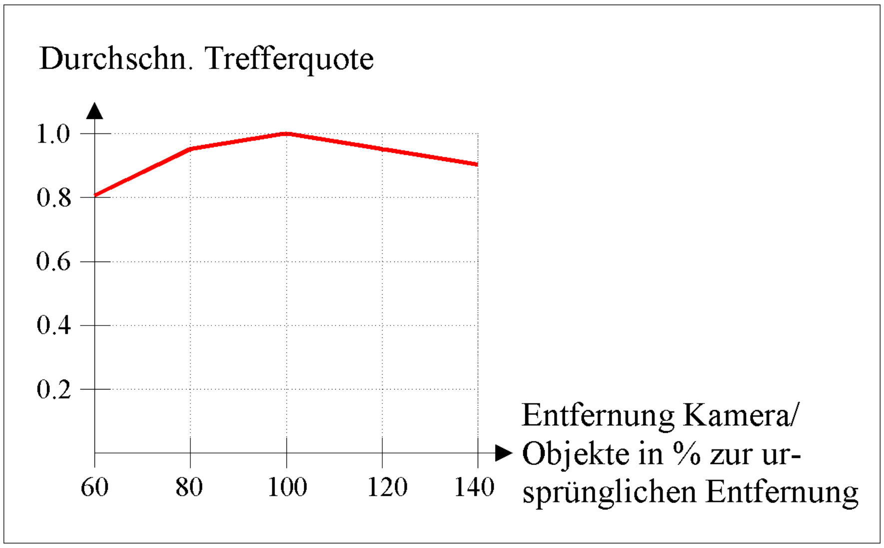


Abbildung 7.4: Statistik über das Verhalten der Intersektion auf Orientierungshistogrammen unter normaler Beleuchtung und Veränderung der Entfernung Kamera/Objekt.

Kapitel 8

Diskussion und Ausblick

Dieses abschließende Kapitel bewertet sämtliche vorgestellte Verfahren, insbesondere die gekoppelten Algorithmen zur farbkonstanten attentiven Suche. Es wird ein Ausblick gegeben auf mögliche zukünftige Entwicklungen oder Forschungsarbeiten.

8.1 Verfahren zur Objektlokalisierung und -identifikation

In dieser Arbeit sind insgesamt sechs verschiedene Algorithmen zur Lokalisierung oder Identifikation eines Objektes vorgestellt, welche sich teilweise miteinander verbinden lassen. Sie können bzgl. der zu lösenden Aufgaben folgendermaßen untergliedert werden:

Aufgabe: Lokalisierung eines Objektes

- **Verfahren:** Swain und Ballards Rückprojektion
  Unter Benutzung von Häufigkeiten der im Bild codierten Merkmale (z.B. Farbe, Helligkeit, Orientierung) werden Objektmerkmale verstärkt, alle anderen Merkmale abgeschwächt und anschließend lokale Anhäufungen der Verstärkungen gesucht.
- **Verfahren:** Schiele und Crowleys Wahrscheinlichkeitsberechnungen
  Bestimmung von Auftrittswahrscheinlichkeiten eines gesuchten Objektes über Merkmalshäufigkeiten mit anschließender lokaler Suche wie bei der Rückprojektion.

Aufgabe: Identifikation eines Objektes

- **Verfahren:** Swain und Ballards Intersektion
  Über den Vergleich von Histogrammen der Bildmerkmale wird auf die Ähnlichkeit von Objekten geschlossen.
- **Verfahren:** Funt und Finlaysons farbkonstante Intersektion
  Gleich der ursprünglichen Intersektion, nur daß die Merkmalshäufigkeiten

zwecks Farbkonstanz auf den Fiterantworten von Ableitungsfiltern auf logarithmisierten Eingangsbildern berechnet werden.

Verfahren: Strickers Ähnlichkeitsfunktion
Vergleich von Bildern aufgrund statistischer Eigenschaften der 1. bis 3. zentralen Momente mit einer gewichteten Abstandsfunktion.

Aufgabe: Lokalisierung und Identifikation

Verfahren: Ennesser und Medionis lokale Histogrammbildung
Gleichzeitige Suche und Identifikation eines Objektes durch Bildung lokaler Histogramme über das gesamte Eingangsbild.

Verfahren: Rückprojektion mit anschließender Intersektion
Attentive Suche durch den Vergleich nur weniger lokaler Histogramme von Bereichen mit den höchsten Auftrittswahrscheinlichkeiten eines Objektes.

Bis auf den Algorithmus von Funt und Finlayson, der bereits eine Vorverarbeitung zur Farbkonstanz besitzt, kann unser Verfahren zur Berechnung lokaler Orientierung jedem Algorithmus vorangestellt werden. Die Algorithmen verwenden lediglich statistische Merkmale wie Histogramme oder Momente. Dadurch haben sie geringe Komplexitäten und sind echtzeitfähig.
Für die visuelle Suche von Objekten in der Robotik zeigen sich die ursprünglichen Methoden der Rückprojektion und der Intersektion durch unsere Experimente am geeignetsten, nicht zuletzt durch die Kombinationsmöglichkeit beider Verfahren zu einer attentiven Suche und die Implementierbarkeit auf Spezialcomputern wie der MV200. Mit Laufzeiten der Algorithmen von $0.0122\,s$ bis $0.036\,s$ liegen diese Werte weit unter den Zeiten, welche z.B. für die reine Digitalisierung von Bildern benötigt wird.

8.2 Farbkonstanz

Unser Algorithmus zur Berechnung lokaler Orientierung führt ein geometrisches Merkmal ein, das die Unterscheidungskapazität von histogrammbasierten oder anderen Verfahren steigert. Die wirkungsvolle Vorverarbeitung von Eingangsbildern liefert farbkonstante Ausgangsbilder für anschließende Verfahren. Neben der Robustheit gegenüber örtlicher und spektraler Beleuchtungsänderung ergeben sich noch die folgenden Eigenschaften eines Orientierungsbildes:

- Unter Beachtung der *Nullorientierung* kann eine Segmentierung des Bildes in einzelne zusammenhängende Bereiche vorgenommen werden.

- Die Rotationslage eines Objektes um die optische Achse läßt sich durch Schwerpunktberechnung der Orientierungshistogramme effektiv und schnell bestimmen.

- Aus der Anzahl der Bildpunkte mit überschwelliger Amplitude läßt sich auf die Größe vorhandener Objekte schließen, wodurch u.a. die Schwäche der Histogrammintersektion bzgl. des Unterscheidens verschieden großer Objekte aufgehoben wird.

Durch die Rotation von Histogrammschwerpunkten in eine Normlage wird die Rotationsinvarianz der Intersektion zurückgewonnen. Es zeigt sich jedoch eine Schwäche der Intersektion, welche Histogramme lediglich aufgrund der höchsten Einträge unterscheidet. Dadurch sind in die Normlage rotierte Histogramme sehr ähnlich. Für diesen Fall sollten Verfahren verwendet werden, die nicht nur Merkmalshäufigkeiten sondern auch räumliche Anordnung der Merkmale zur Identifikation nutzen.
Mit der Implementierung des Farbkonstanzalgorithmus' auf dem Pipelinerechner haben wir gezeigt, daß intensive Filterungen eines Eingangsbildes (in unserem Fall 48(!) Faltungen mit 8×8-Masken pro Bild) nicht zu langen Laufzeiten eines Verfahrens führen müssen. Die Faltungskapazitäten des Spezialrechners und parallele Implementierung ermöglichen eine Vorverarbeitung, welche unter $0.1\,s$ liegt. Durch den Einsatz zusätzlicher Speicher- und Additionskapazitäten ist nochmals eine Halbierung dieser Zeit möglich.

8.3 Mögliche Weiterentwicklungen

Die Algorithmen von Swain und Ballard unterliegen verschiedenen Einschränkungen. So wird z.B. die gleiche Objektgröße wie zur Bildung der Datenbank vorausgesetzt, bei der Rückprojektion ist die Rotationslage des Objektes zur optischen Achse zur genauen Lokalisierung notwendig. Zu einigen dieser Einschränkungen geben wir Lösungsmöglichkeiten.

8.3.1 Genaue Lokalisierung bei unbekannter Rotationslage

Zur exakten Objektlokalisierung durch die Rückprojektion ist die Kenntnis der Rotationslage des Objektes um die optische Achse notwendig (siehe Abschnitt 2.1.2.2). Da wir die Lage des Objektes durch Schwerpunktberechnung der Orientierungshistogramme bestimmen können, ist folgendes dreistufige Verfahren auf den Orientierungsbildern vorstellbar:

1. Ungenaue Lokalisierung eines Objektes durch die Rückprojektion. Trotz evtl. Rotation des Objektes gibt es durch die Quantisierung bei der Bildung der 3D-Histogramme noch genügend korrespondierende Bins, so daß die Werte der ungeglätteten Rückprojektion im Objektbereich nicht gleich Null sind.

2. Identifizierung des Objektes durch Schwerpunktberechnung und Rotation der Orientierungshistogramme sowie anschließender Intersektion.

3. Rückprojektion der Szene mit einem rotierten Modellhistogramm sowie einer rotierten Glättungsmaske.

Der letzte Schritt setzt die korrekte Ermittlung der Rotationslage voraus. Dies ist mit unserem Verfahren aus Abschnitt 5.4.1 zwar möglich, jedoch ist dort Voraussetzung, daß der Hintergrund keine wesentlichen Orientierungen beeinhaltet.

8.3.2 Analyse einer Szene unter gleichbleibender Objekt/ Kamera-Entfernung

Die Rückprojektion – angewendet auf Farb- oder Orientierungsbilder – beantwortet die Frage, wo sich in einem Bild ein ganz bestimmtes Objekt A am wahrscheinlichsten befindet. Voraussetzung ist dabei die implizite Kenntnis der Entfernung vom Objekt zur Kamera, indem die gleiche Objektgröße wie zur Bildung der Histogrammdatenbank angenommen wird. In der Robotik oder 'Active Vision' wird oftmals jedoch die Analyse einer unbekannten Szene auf das Vorkommen sämtlicher möglicher Objekte benötigt. Eine einmalige Rückprojektion mit anschließender Intersektion auf einigen vielversprechenden Bereichen wird diesen Anforderungen jedoch nicht gerecht. Die exakte Lokalisierung eines Objektes A setzt auch die Suche nach diesem Objekt voraus und kann nicht als Nebenprodukt der Suche nach einem anderen Objekt erreicht werden.
Auf Orientierungsbildern könnte eine Lösung ein mehrstufiges Verfahren sein, welches aus folgenden Schritten besteht:

1. In der Offline-Phase Bildung eines 'universellen' Objektes, welches die Merkmale aller Orientierungshistogramme der Datenbank enthält.

2. Berechnung der Rückprojektion von Szene und 'universellem' Objekt, wobei mit einer Maske geglättet wird, welche die Abmessungen der größten Objekte hat. Es ergeben sich in den ungefähren Zentren vorhandener Objekte lokale Maxima, deren Bereiche mit der Intersektion analysiert werden. Die Position der somit identifizierten Objekte ist noch ungenau, bedingt durch die Größe der Glättungsmaske.

3. Exakte Lokalisierung durch nacheinander ausgeführte Rückprojektion aller identifizierten Objekte. Da bereits durch die Intersektion bestimmten Bereichen Modelle der Datenbank zugeordnet sind, ist eine erneute Intersektion nicht erforderlich.

Geht man von fünf Objekten in einer Szene aus, hätte dieses mehrstufige Verfahren mit unseren Möglichkeiten auf dem Pipelinerechner eine ungefähre Laufzeit von $0.4\,s$.

8.3.3 Analyse einer Szene unter unbekannter Objekt/Kamera-Entfernung

Bei einem sich bewegenden Fahrzeug wird das Problem der Lokalisierung und Identifizierung von Objekten mit unterschiedlichen Entfernungen zu lösen sein. Die

Entfernungs- oder Tiefeninformation sind grundsätzlich nicht vorhanden, es gibt aber verschiedene Verfahren, diese zu erlangen. So benutzen z.B. Hansen und Sommer[HS96] Gabor-Filter, um mit einem System von Stereo-Kameras auf einem Roboter-Kopf Disparitäten-Karten verschiedener Auflösung zu berechnen. Die Implementierung ist auf demselben Pipelinerechner vorgenommen, auf dem auch wir unser Programm schreiben. Potsdamer und Altschuler[PA82] berechnen Tiefeninformationen mit einer Kamera und einem getakteten Laser-Projektor.
Mit den somit gewonnenen Tiefenkarten (siehe z.B. [WWB84]) kann eine Szene in verschiedene Tiefenabschnitte eingeteilt werden. In dem Kamerabild werden je nach Ebene, die analysiert werden soll, abschnittsfremde Bildpunkte ausgeblendet, so daß auf dieses reduzierte Bild das obige Verfahren aus Abschnitt 8.3.2 angewendet werden kann.

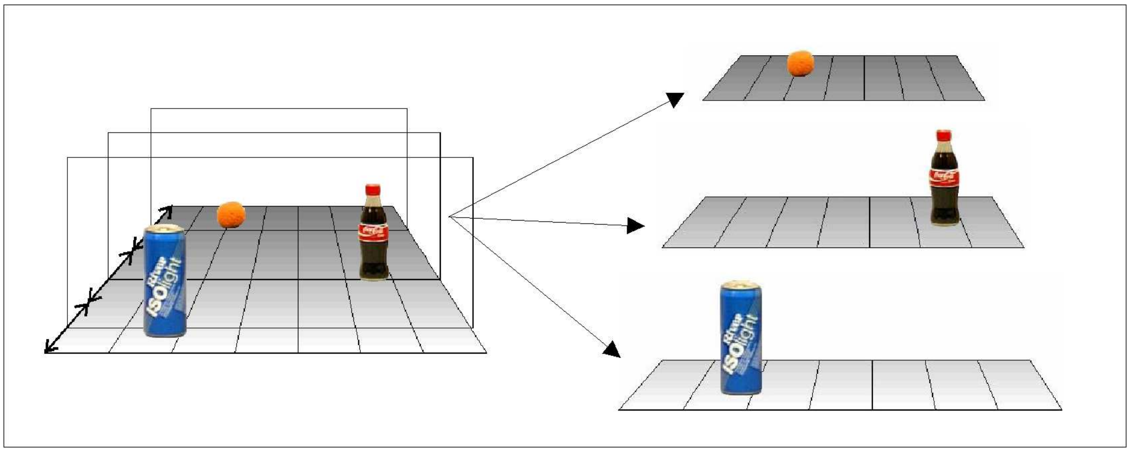


Abbildung 8.1: Ideale Darstellung einer mögliche Integration der Rückprojektion und Intersektion in ein Verfahren zur Analyse von 3D-Räumen. Mittels Entfernungsinformation wird der Raum in verschiedene Tiefenebenen eingeteilt, auf welche das Verfahren zur Analyse von Szenen mit bekannten Objektentfernungen angewendet werden kann.

Probleme könnten sich dabei durch die unterschiedliche Anzahl von Merkmalspunkten je nach Entfernung eines Objektes zur Kamera ergeben. Mögliche Lösung hierfür ist z.B. die Normierung der Bins sämtlicher Objekt- und Modellhistogramme, was einer nachträglichen Skalierung der Objekte gleichkommt, oder die separate Erstellung von Abschnitten der Datenbank mit Histogrammen von Modellen aus den einzelnen Tiefenebenen. In jeder Tiefenebene werden dann lediglich die Modellhistogramme der zugehörigen Ebene verwendet.

Literaturverzeichnis

[And92] M.T. Andersson. *Controllable multidimensional filters and models in low level computer vision.* PhD thesis, Linkoping University, 1992.

[BB82] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice Hall, 1982.

[EM95] E. Ennesser and M. Medioni. Finding waldo. *IEEE Transactions on Pattern Analysis and Machine Intelligence unknown*, 17(5):522–529, 1995.

[FF95] B.V. Funt and G.D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, 1995.

[HS96] M. Hansen and G. Sommer. Active depth estimation with gaze and vergence control using gabor filters. In *13. th ICPR*, volume A, pages 344–348, 1996.

[Jäh93] B. Jähne. *Digitale Bildverarbeitung.* Springer Verlag, 1993.

[KS94] A. Koschan and K. Schlüns. Grundlagen und Voraussetzungen für die digitale Bildverarbeitung. Technical report, Technische Universität Berlin, 1994.

[LPSE94] C.-M. Lee, T.-C. Pong, J.R. Slagle, and A. Esterling. An experimental study of an object recognition system that learns. *Pattern Recognition*, 27(1):65–89, 1994.

[MSS95] U. Mahlmeister, B. Schmidt, and G. Sommer. Preattentive colour features by steerable filters. In *Mustererkennung 1995*, pages 464–472. 17. DAGM-Symposium, 1995.

[PA82] J.L. Potsdamer and M.D. Altschuler. Surface measurement by space-encoded projected beam system. *Computer Graphics and Image Processing*, 18:1–17, 1982.

[SB91] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[SC96] B. Schiele and J. L. Crowley. Probabilistic object recognition using multidimensional receptive field histograms. In *ICPR'96, Proceedings of the 13th International Conference on Pattern Recognition, Volume B*, pages 50–54, August 1996.

[SO95] M. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392. SPIE Proceedings Series, 1995.

[SR88] S. Simon and J.P. Rogala. Model-based prediction-verification scheme for real-time inspection. In *Pattern Recognition Letters*, volume 7, pages 305–311, 1988.

[Str94] M. Stricker. Bounds for the discrimination power of color indexing techniques. In *Storage and Retrieval for Image and Video Databases II*, volume 2185, pages 15–24. SPIE Proceedings Series, 1994.

[WWB84] C.K. Wu, D.Q. Wang, and R.K. Bajacsy. Acquiring 3-d spatial data of a real object. *Computer Vision, Graphics, and Image Processing*, 28:126–133, 1984.

Anhang A

Berechnung der Fourierkorrespondenz $f^\theta(x,y) \circ\!\!-\!\!\bullet\, F^\theta(u,v)$

Seien f und und F zweistellige Funktionen mit $f(x,y) \circ\!\!-\!\!\bullet\, F(u,v)$. f^θ (bzw. F^θ) bezeichne die um den Winkel θ rotierte Funktion f (bzw. F). Dann gilt:

$$f^\theta(x,y) \circ\!\!-\!\!\bullet\, F'(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f^\theta(x,y) e^{-i2\pi(ux+vy)}\, dx\; dy\ . \tag{A.1}$$

Substituiert man in dem Doppelintgral die kartesischen Koordinaten in Polarkoordinaten durch

$$\begin{array}{ll} x = r\cos(\varphi) & u = \rho\cos(\vartheta) \\ y = r\sin(\varphi) & v = \rho\sin(\vartheta) \end{array} \tag{A.2}$$

erhält man

$$\begin{aligned} F'(\rho,\vartheta) &= \int_0^\infty\int_0^{2\pi} f^\theta(r,\varphi) r e^{-i2\pi r\rho\cos(\vartheta)\cos(\varphi)+\sin(\vartheta)\sin(\varphi)}\, dr\; d\varphi \\ &= \int_0^\infty\int_0^{2\pi} f^\theta(r,\varphi) r e^{-i2\pi r\rho\cos(\varphi-\vartheta)}\, dr\; d\varphi \\ &= \int_0^\infty\int_0^{2\pi} f(r,\varphi-\theta) r e^{-i2\pi r\rho\cos(\varphi-\vartheta)}\, dr\; d\varphi \end{aligned} \tag{A.3}$$

Substitution von $\varphi' = \varphi - \theta$ mit anschließender Substitution zurück in die kartesischen Koordinaten mit

$$\begin{array}{ll} u' = \rho\cos(\vartheta-\theta) & x' = r\cos(\varphi') \\ v' = \rho\sin(\vartheta-\theta) & y' = r\sin(\varphi') \end{array} \tag{A.4}$$

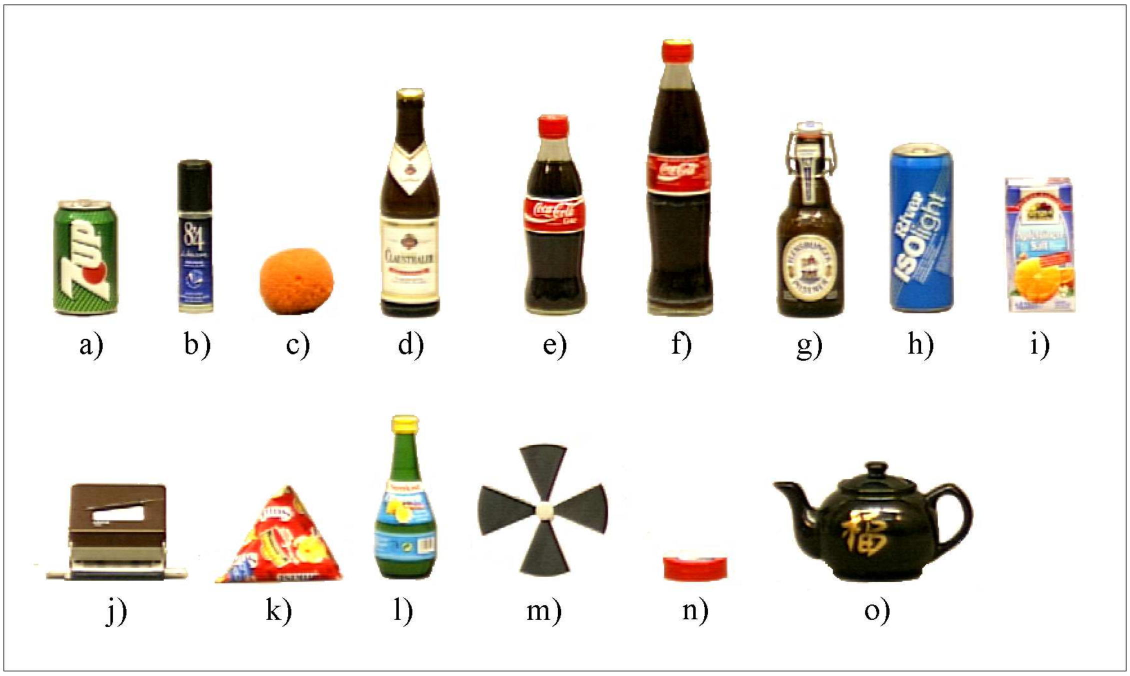
ergibt schließlich

$$
\begin{aligned}
F'(\rho,\vartheta) &= \int\limits_0^{\infty}\int\limits_{\theta}^{2\pi+\theta} f(r,\varphi')re^{-i2\pi r\rho\cos(\varphi'-\theta-\vartheta)}\,dr\;d\varphi' \\
&= \int\limits_0^{\infty}\int\limits_{\theta}^{2\pi+\theta} f(r,\varphi')re^{-i2\pi r\rho(\cos(\vartheta-\theta)\cos(\varphi')+\sin(\vartheta-\theta)\sin(\varphi')}\,dr\;d\varphi' \\
&= \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} f(x',y')e^{-i2\pi(u'x'+v'y')}\,dx'\;dy' \\
&= F(u',v') \\
&= F^{\theta}(u,v)\;. \qquad \text{(A.5)}
\end{aligned}
$$

Das letzte Gleichheitszeichen resultiert aus der Ersetzung von (u', v') in (u, v) durch Teile der Substitutionsformeln (A.2) und (A.4).

Anhang B

Modelle der Datenbank



Die Modelle werden in der vorliegenden Arbeit mit folgenden Abkürzungen bezeichnet:
a) 7Up; b) 8x4; c) Ball; d) Claust; e) Coke03; f) Coke05; g) Flens; h) Iso; i) Kinks; j) Locher; k) SunOr; l) SunZi; m) Scheibe; n) Tesa; o) Teekanne.

Anhang C

Testszenen der Rückprojektion

Das Verfahren der Rückprojektion wurde an folgenden fünf Szenen unter unterschiedlicher Beleuchtung getestet:

Szene 1:

Szene2:


normal


dunkel


hell


rot


grün


blau

Szene3:


normal


dunkel


hell


rot


grün


blau

Szene 4:

normal

dunkel

hell

rot

grün

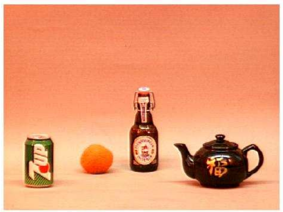
blau

Szene 5:

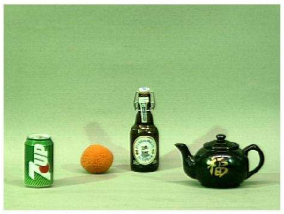
normal

dunkel

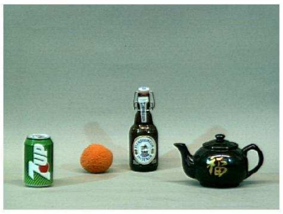
hell



rot



grün



blau