Erweiterter
Projektbericht

Modellierung und Extraktion von Objekten aus Entfernungs- und Schichtdaten

(Sommersemester 1995)

(mit Erweiterungen vom Wintersemester 1995/96)

Automatische Erkennung von Objekten mit dem „Grünen Punkt“

Kjell Oppermann und Wolfram J. Schimke

Berlin, 19. Januar 1996.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Das Ziel dieser Arbeit ist es, mittels eines Programms weitgehend polyedrische Objekte anhand der Aufdrucke auf deren Flächen automatisch zu erkennen. Die Arbeit entstand an der TU–Berlin im Rahmen des Computer Vision Projekts „Modellierung und Extraktion von Objekten aus Entfernungs- und Schichtdaten“ bei Gunter Bellaire und Karsten Schlüns.
Den Hintergrund dieser Projektarbeit bildet die Erkennung von Müll, speziell die Erkennung von Objekten mit dem „Grünen Punkt“. Diese müssen aufgrund der Verschiedenartigkeit der Materialien sortiert werden, um sie recyceln zu können. In Berlin geschieht dies bisher durch Handarbeit am Fließband. Eine automatische Sortierung nach der Materialart wäre hier sinnvoll.
Die Lösung des Problems kann man sich folgendermaßen vorstellen: Der Müll wird so auf dem Fließband verteilt, daß jedes Teil von den anderen Objekten getrennt befördert wird, so daß eine Kamera über dem Band das Objekt einzeln aufnehmen kann. Ein Computersystem hat eine hohe Anzahl von möglichen Objekten in seiner Datenbank gespeichert, und ein Matching–Algorithmus probiert, das soeben aufgenommene Objekt einem bekannten Objekt aus der Datenbank zuzuordnen. War dies erfolgreich, so kann auch das Material, aus dem das aufgenommene Objekt besteht, bestimmt werden.
Unsere Aufgabe bestand darin, die Struktur der Datenbank zu entwerfen, eine kleine Datenbank zu erstellen sowie ein Programm zu entwickeln, welches die Zuordnung eines aufgenommenen Objekts zu den bekannten Objekten in der Datenbank realisiert.

2 Aufgabenstellung

Im gleichen Rahmen entstand bereits eine Arbeit mit dem Thema „Objekterkennung auf der Grundlage von Surface Normal Images“ von Klaus Gwinner und Arno Mitritz [1]. Darin werden topologische und numerische Merkmale für das Matching herangezogen. Zu diesen Merkmalen gehören z.B. die Anzahl und der Typ der Ecken bzw. Kanten, die Anzahl der Flächen, die Flächengrößenverhältnisse, die Kantenlängenverhältnisse etc.
Wir verfolgen einen anderen Ansatz, um die Objekte und deren Flächen zuzuordnen: Die Objekte werden anhand der (meist farbigen) Aufdrucke auf den Flächen identifiziert. Heutzutage sind die Verpackungen von Produkten vielseitig und bunt gestaltet, um die Aufmerksamkeit der Kunden auf ein Produkt zu lenken. Desweiteren kann man beobachten, daß sich die Verpackungen von Konkurrenzprodukten mit ähnlichen Inhalten stark in Farbe und Gestaltung unterscheiden, um Verwechslungen zu vermeiden. Somit sind die Voraussetzungen für eine eindeutige Zuordnung von derartigen Objekten anhand ihrer Aufdrucke gegeben.
Bei der Lösung der Aufgabe wird Wert auf die Minimierung der Rechenzeit gelegt, die für die Erkennung eines Objektes benötigt wird: Im praktischen Einsatz muß solch eine Objekterkennung im Sekundenbereich geschehen. Dabei soll das Verfahren keine hohen Anforderungen an die Rechentechnik stellen.

3 Surface Normal Images

Park, Chang und Choi diskutieren in [2] die Möglichkeiten, dreidimensionale Objekte in einem System geeignet zu repräsentieren. Bisherige Darstellungsvarianten wie z.B. 3D–Modelle (volume representation, surface boundary representation, generalized cylinders) bzw. charakteristische Ansichten sind nur mit hohem Aufwand zu verwalten und erfordern viel Rechenleistung beim Matching.
Um diesen Aufwand zu verringern, wird im oben genannten Artikel die Objektrepräsentation durch Surface Normal Images (SNIs) beschrieben. Dabei ist ein SNI die 2D–Ansicht (Parallelprojektion) eines 3D–Objekts, wobei genau eine Fläche unverzerrt (parallel zur Bildebene) abgebildet wird. Für diese eine Fläche, die als Basisfläche bzw. Surface Normal View (SNV) bezeichnet wird, liegt die Oberflächennormale parallel zur Blickrichtung. Jeder Fläche eines Objekts ist genau ein SNI zugeordnet, d.h. für jedes Objekt ist die Anzahl der SNIs gleich der Anzahl der Flächen.
Die folgende Grafik (Abb. 1) zeigt die SNI–Struktur eines Objekts mit sieben Flächen:

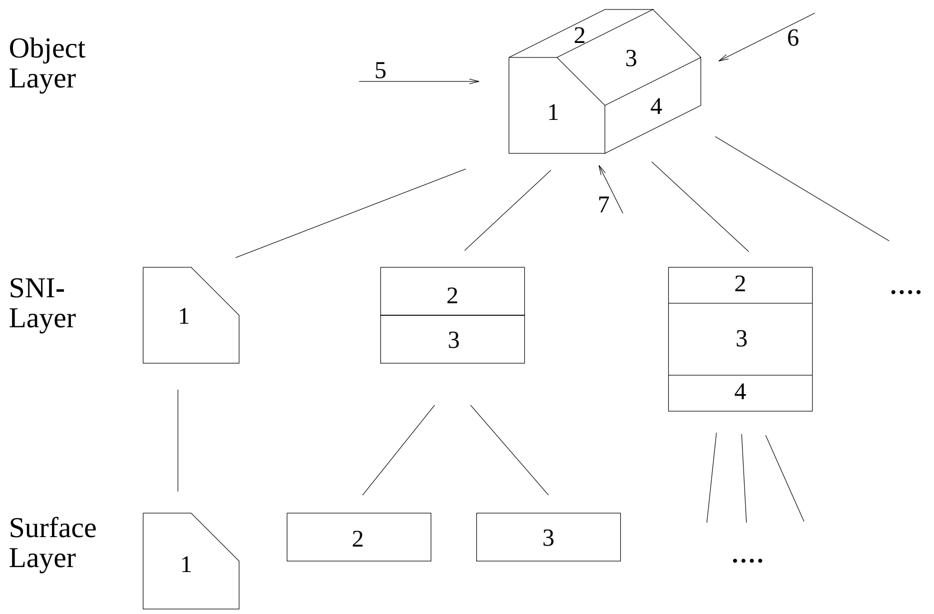


Abbildung 1: SNI–Struktur eines Objekts mit sieben Flächen (nach [1])

Mit dieser Struktur läßt sich eine leicht zu verwaltende Datenbank aufbauen, wobei in diese Baumstruktur weitere Daten integriert werden können (z.B. zu jeder Fläche deren Kantenanzahl, Kantenlängen, etc.).

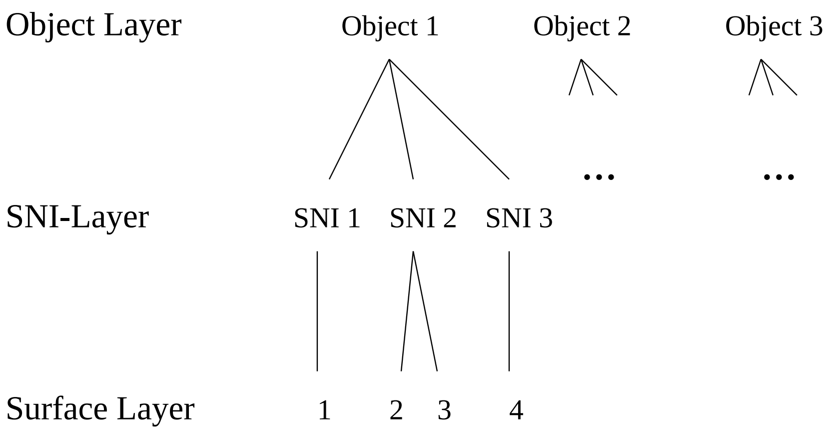


Abbildung 2: Baumstruktur der SNI–Daten (nach [1])

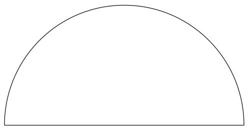
Um jetzt ein zu erkennendes Objekt mit der Datenbank zu vergleichen, wird das aufgenommene Objekt so gedreht, daß jeweils eine Fläche parallel zur Bildebene liegt. Solche Abbildungen werden Rotated Input Images (RIIs) genannt und sind topologisch gleich[1] zu dem entsprechenden Eintrag in der SNI–Struktur.

Abbildung 3: Erzeugung eines RII (nach [1])

Gekrümmte Flächen

Gekrümmte Flächen werden dabei durch planare approximiert, d.h. Oberflächennormalen mit einer bestimmten Abweichungstoleranz werden zu einer planaren Fläche zusammengefaßt. Je kleiner die Toleranz gewählt wird, desto mehr planare Flächen erhält man und desto genauer wird die Krümmung modelliert aber umso komplexer gestaltet sich die Verwaltung. Daher ist es nötig, die Anzahl der planaren Flächen relativ klein zu halten, um den Aufwand zu minimieren.
Diese Abbildung zeigt die Einteilung der Oberfläche einer Halbkugel:

[1]Die RIIs haben auch bis auf eine Rotation um die z–Achse die gleiche Lage wie die entsprechenden SNIs.



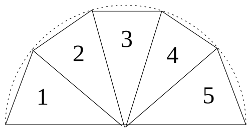


Abbildung 4: Approximation gekrümmter Flächen (nach [1])

Unsere Datenbank, die in Abschnitt 5 genau beschrieben ist, haben wir in Anlehnung an diese Struktur aufgebaut.

4 Aufnahme der Bilder

Sämtliche Aufnahmen von Objekten mit dem „Grünen Punkt“ (sowohl für die Erstellung einer Datenbank als auch für die Testdaten zur Objekterkennung) wurden unter weitgehend realitätsnahen Bedingungen im CV–Labor durchgeführt. Die Objekte befanden sich zur Aufnahme auf einem mit schwarzem Tuch bedeckten Tisch. Das Tuch füllte den Hintergrund des Kamerabildes vollständig aus. Beleuchtet wurde die Szene mit bis zu drei Diaprojektoren, die als Punktlichtquellen für das PHOTOMETRIC–STEREO–Verfahren[2] (PMS) erforderlich sind.

Wir bedanken uns an dieser Stelle bei Arno Mitritz und bei Tapani Hegewald für die Unterstützung bei der Erstellung der Aufnahmen der ersten bzw. zweiten Serie.

4.1 Aufnahme der Bilder für die Datenbank

Zunächst haben wir eine Reihe von Objekten zur Erstellung einer repräsentativen Datenbank ausgesucht. (Eine Liste der Objekte ist in Tabelle 1 (erste Serie) bzw. Tabelle 2 (erste Serie) zu finden.)

In der ersten Serie wurden von jedem Objekt alle relevanten Flächen einzeln aufgenommen. Als nicht relevant haben wir in der Regel den weißen, unbedruckten Boden vieler Objekte angesehen sowie extrem kleine Flächen[3]. In der zweiten Serie haben wir alle größeren Flächen aufgenommen, inklusive gleichartig bedruckter Flächen.

Die Objekte wurden so ausgerichtet, daß die Oberflächennormale der aufzunehmenden Fläche parallel zur optischen Achse der Kamera stand und sich in der Bildmitte befand. Der Abstand zwischen Kamera und Objekttisch sowie — in der ersten Serie — die Brennweite des Kameraobjektivs blieben unverändert, so daß kleinere Flächen entsprechend kleiner abgebildet wurden. Da wir nicht mit absoluten Längen arbeiten, wäre es jedoch, um eine verbesserte Detailauflösung zu erhalten, günstiger, alle Flächen bildfüllend aufzunehmen. Daher haben wir bei der Aufnahme der zweiten Serie die Brennweite des Kameraobjektivs an die Größe des Objekts angepaßt. Beleuchtet wurde das Objekt mit einem Diaprojektor in der Nähe der Kamera. Die aufgenommenen Bilder wurden anschließend nachbearbeitet: Die Flächen wurden per Hand vom Hintergrund separiert und ohne Hintergrund

[2]Dieses Verfahren dient zur Rekonstruktion der dreidimensionalen Gestalt (2.5D) der Objekte der aufgenommenen Szene. Dabei wird die Szene durch jeweils eine Punktlichtquelle aus verschiedenen Richtungen beleuchtet. Die Kamera und die Szene werden nicht bewegt.

[3]Diese Flächen sind auch in den Objekt–Modellen nicht enthalten.

Objekt	Name in der Datenbank
Giebelmilchtüte „Brandenburg“, 1l	`Milch_Brandbg_1.0`
Giebelmilchtüte „Reichelt“, 2l	`Milch_Reichelt_2.0`
Giebelmilchtüte „Reichelt“, 1l	`Milch_Reichelt_1.0`
Giebelmilchtüte „Reichelt“, Sahne, 0,5l	`Sahne_Reichelt_0.5`
Giebelmilchtüte „Uelzena“, 0,5l	`Milch_Uelzena_0.5`
Tetrapak „Penny Milch“, 1l, rot	`Milch_H_r_1.0`
Tetrapak „Penny Milch“, 1l, blau	`Milch_H_b_1.0`
Tetrapak, 1l	`Traubensaft_1.0`
Tetrapak „Valensina“, 0,2l	`Valensina_0.2`
Tabakdose, matt bedrucktes Blech	`Tabakdose`
Quarkbecher (weiß) mit bedrucktem Deckel	`Quarkbecher`
Papier–Schachtel, „Toblerone“	`Toblerone`
Joghurtbecher „Brandenburg“, 500g	`Joghurt`
Plastik–Flasche, „Johnson“–Autowachs	`Johnson_Wachs`
Plastik–Flasche, „Odol“	`Odol`
Geschirrspülmittel–Flasche (transparent)	`Pril`

Tabelle 1: Liste der Testobjekte — erste Serie

(also „ausgeschnitten“) gespeichert. Bei rechteckigen Flächen kann der Hintergrund vollständig beseitigt werden, insofern sie am Bildrand ausgerichtet sind. Bei nicht–rechteckigen Flächen wird die Fläche so gut wie möglich vom Hintergrund getrennt und das die Fläche umschreibende Polygon im „ausgeschnittenen“ Bild in der Datenbank angegeben (Clip–Path, siehe Abschnitt 5.1). Ist die Fläche zwar rechteckig, aber nicht am Bildrand ausgerichtet, so kann sie durch eine (2D–)Drehung in die ausgerichte Lage gebracht werden, um auf die Angabe eines Clip–Path verzichten zu können.
Da wir einen schwarzen Hintergrund verwendet haben, ist auch eine automatische Separierung denkbar, ähnlich der in [3] beschriebenen, solange nicht mehrere gleichzeitig sichtbare Flächen zu trennen sind. Der Operatorkern für die oben erwähnte Drehung befindet sich bei den zum Projekt gehörenden Quelltexten. Von einer Verwendung haben wir bislang aufgrund der geringen Abweichungen bei unseren Bildern abgesehen.

4.2 Aufnahme der Bilder für die Testreihen

Zu einem späteren Zeitpunkt wurden die Testobjekte in beliebigen Lagen aufgenommen. Da wir für die Segmentierung Tiefeninformationen benötigen, wurden die Objekte mit einem Photometric–Stereo–Verfahren mit

Objekt	Name in der Datenbank
Margarine–Dose „Rama“	`Rama`
Butter–Dose	`Butter`
Tetrapak „DieWeissen Milch“, 1l	`DieWeissen_Milch_1.0`
Tetrapak „Milram Milch“, 1l	`Milram_Milch_1.0`
Tetrapak „Kirberg Orangennektar“, 1l	`Kirberg_Orangennektar_1.0`
Tetrapak „Kirberg Traubensaft“, 1l	`Kirberg_Traubensaft_1.0`
Tetrapak „Urbanus Mineralwasser“, 1l	`Urbanus_Wasser_1.0`
Tetrapak „Pfanner Eistee“, 1l	`Pfanner_EisTee_Ananas_1.0`
Tetrapak „Penny Milch“, 1l, rot	`Penny_HMilch_r_1.0`
Tetrapak „Penny Milch“, 1l, blau	`Penny_HMilch_b_1.0`
Tetrapak „Milsani Milch“, 1l	`Milsani_Milch_1.0`
Tetrapak „Valensina Multiv.“, 0,75l	`Valensina_Multi_0.75`
Tetrapak „Valensina Orange“, 0,75l	`Valensina_Orange_0.75`
Tetrapak „Valensina Orange“, 0,2l	`Valensina_Orange_0.2`
Tetrapak „Milsani Kondensmilch“ 0,2l	`Milsani_Kondens_0.2`
Quarkbecher (weiß) mit bedrucktem Deckel	`Milram_Quark`
Eispackung „Mövenpick Himbeer“, 1l	`Moevenpick_Himbeer`
Giebelmilchtüte „Reichelt“, 1l	`Reichelt_Milch_1.0`
Giebelmilchtüte „Brandenburg“, 2l	`Brandenburg_Milch_2.0`
Giebelmilchtüte „Erlenhof“, 1l	`Erlenhof_Milch_1.0`
Giebelmilchtüte „Emzett“, 1l	`Emzett_Milch_1.0`
Giebelmilchtüte „Reichelt“, Sahne, 0,5l	`Reichelt_Sahne_0.5`
Giebelmilchtüte „Uelzena“, 0,5l	`Uelzena_Milch_0.5`
„Toblerone“ (dreiseitiges Prisma)	`Toblerone_400`
Joghurtbecher „Brandenburg“	`Brandenburg_Joghurt_Erdbeer`
Milchbecher „Müller“	`Mueller_Milch_Schoko`
Joghurtbecher „Brandenburg“, groß	`Brandenburg_Joghurt_500`
Geschirrspülmittel–Flasche (transparent)	`Pril`
Glas–Flasche, „Odol“	`Odol`

Tabelle 2: Liste der Testobjekte — zweite Serie

drei Lichtquellen aufgenommen. Die Anordnung von Objekt, Kamera und Lichtquellen zeigt Abb. 5. Als Lichtquellen dienten Diaprojektoren.
Für das PHOTOMETRIC–STEREO–Verfahren wurden bei der ersten Serie drei Aufnahmen erstellt, jeweils eine mit genau einer der drei Lichtquellen. Eine vierte Aufnahme wurde mit allen drei Lichtquellen erstellt. Die Blende war für alle Aufnahmen konstant so eingestellt, daß alle Bildpunkte in allen

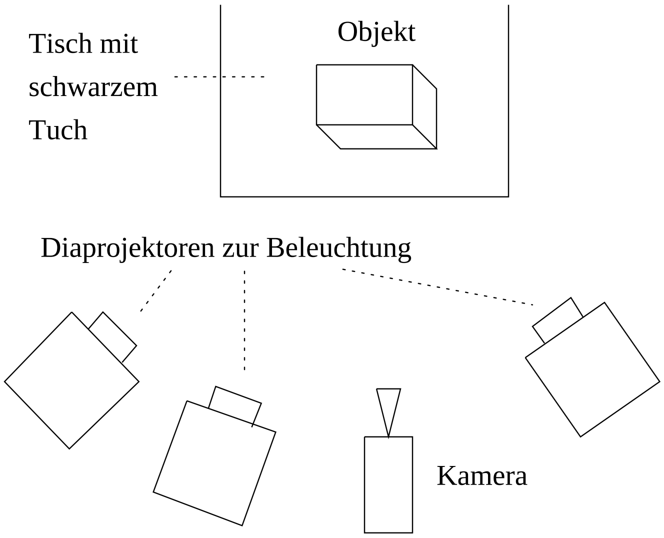


Abbildung 5: Anordnung für PHOTOMETRIC–STEREO–Aufnahmen

Kanälen Werte unter 190 aufwiesen, und sie somit im linearen Bereich der Kamera blieben. Bei der zweiten Serie wurden zweimal drei Aufnahmen mit jeweils einer der drei Lichtquellen erstellt, wobei unterschiedliche Blenden verwendet wurden. Die erste Blendeneinstellung war so gewählt, daß alle Bildpunkte in allen Kanälen Werte unter 190 aufwiesen. Bei der zweiten Einstellung wurde die Blende etwas geöffnet, um auch dunklere Objektteile ausreichend hell abzubilden. Eine siebente Aufnahme wurde mit allen drei Lichtquellen bei einer eigenen Blendeneinstellung gemacht. Die Blende wurde so eingestellt, daß alle Bildpunkte in allen Kanälen Werte unter 190 aufweisen. Die vierte bzw. siebente Aufnahme wird als Farbbild für das Matching verwendet.

Bei der ersten Serie wurden aus den ersten drei Aufnahmen die Orientierungen berechnet (sogenannte „pq–Bilder“[4]), die zur Segmentierung (siehe Abschnitt 7) benötigt werden.

Bei der zweiten Serie wurde zuerst für jeden Bildpunkt eine der beiden Blendeneinstellungen gewählt: Aus allen Blendeneinstellungen, deren sämtliche Bilder in allen Kanälen Werte unter 190 besitzen, wird die hellste gewählt, d.h. diejenige, bei der die Summe aller Kanäle über alle Bilder maximal[5] ist. Ist bei allen Blendeneinstellungen in mindestens einem Bild mindestens ein Kanal mit einem Wert größer als 190 behaftet, wird diejenige Blende gewählt, bei der die Summe der Differenz (mit Vorzeichen) jedes

[4]teilweise etwas unkorrekt als Nadelkarte bezeichnet

[5]Ist das Maximum nicht eindeutig, wird die erste Blende gewählt.

Kanals zu 190 über alle Bilder minimal[6] ist. Auf diese Weise erhält man eine einfache Glanzlichtelimination und eine mit dem PMS–Verfahren konforme Aufhellung dunkler Objektbereiche. (Die ist möglich, weil wir ein Albedo–unabhängiges PMS–Verfahren einsetzen.)
Anschließend wird für die so erhaltenen drei Farbbilder für jeden Bildpunkt der in der Summe über alle drei Bilder hellste Kanal bestimmt und in die Ergebnisbilder übernommen. Auf die so erhaltenen Grauwertbilder wird dann das PMS–Verfahren angewandt.
Da sich unter den Objekten einige mit einem hohen Anteil weißer Flächen befinden, haben wir bei der ersten Serie relativ stark abblenden müssen, so daß dunklere Objekte wie der `Traubensaft` zu dunkel abgebildet wurden. In solchen Bildern können dann stellenweise keine pq–Werte berechnet werden, was zu Fehlern bei der Segmentierung führt. (Das Objekt `Traubensaft` ist in dieser Beziehung besonders problematisch, da es selbst größere weiße Flächen besitzt, so daß eine optimale Blendeneinstellung der Kamera kaum möglich ist.) Diese Probleme konnten mit dem oben beschriebenen Verfahren bei der zweiten Serie beseitigt werden.
Die Wahl der drei Beleuchtungsrichtungen ist von entscheidender Bedeutung. Denn nur, wenn alle Flächen des Objekts von allen drei Lichtquellen vollständig ausgeleuchtet werden, ist eine Berechnung der pq–Werte für alle sichtbaren Objektpunkte möglich. Gegebenenfalls ist ein PMS–Verfahren mit mehr als drei Lichtquellen zu verwenden.

4.3 Aufnahmebedingte Probleme beim Matching

Bei der Betrachtung beider Aufnahmereihen (für die Datenbank und für die Tests) der ersten Serie fiel uns ein erheblicher Unterschied der Farbbalance auf. Die Bilder für die Tests hatten ein „gutes Weiß“, wohingegen die Bilder der Datenbank einen zu geringen Blauanteil aufwiesen sowie generell etwas dunkler waren als die Bilder für die Tests.
Da alle unsere Matching–Verfahren auf eine gute Übereinstimmung der Farbbalance aller Aufnahmen angewiesen sind, haben wir die Bilder der Datenbank der ersten Serie dahingehend nachbearbeitet. Beim Blauanteil wurden die Werte im Bereich 0 bis 205 auf den gesamten Bereich (von 0 bis 255) gestreckt und anschließend bei allen drei Farbanteilen die Werte im Bereich 0 bis 200 auf den gesamten Bereich gestreckt. Die Genauigkeit der Zuordnung konnte damit erheblich verbessert werden.
Bei der zweiten Serie waren die Farbabweichungen gering, obwohl beide Aufnahmeserien wieder zu verschiedenen Zeitpunkten erfolgen mußten. Wir

[6] Ist das Minimum nicht eindeutig, wird die erste Blende gewählt.

haben jedoch jedesmal die drei Diaprojektoren auf möglichst gleiche Helligkeit und ausgeglichenes Weiß eingestellt. Mangels einer Einstellmöglichkeit am Gerät mußte die Helligkeit von zwei der drei Diaprojektoren mit dem Abstand zum Objekt eingestellt werden.
In der späteren Anwendung sollten die Bilder für die Datenbank unter den gleichen Lichtbedingungen aufgenommen werden wie die zu erkennenden Objekte. Wie die zweite Serie gezeigt hat, ist eventuell auch ein korrekter Weißabgleich sowie eine genaue Einstellung der Blende der Kamera trotz verschiedener Beleuchtung ausreichend. Eventuell ist auch die Verwendung eines gegenüber Abweichungen der Farbtemperatur toleranteren Matching–Verfahrens in Erwägung zu ziehen.
Ein schwerwiegendes Problem sind größere Abweichungen der aus den Aufnahmen für die Tests berechneten Orientierungen (siehe auch Abschnitt 8.2.2). Die Winkel zwischen den einzelnen Flächen stimmen auf ± 5 Grad mit den realen Werten überein. Die absoluten Werte der Orientierungen (x– und y–Anteil) haben zum Teil gravierende Abweichungen. Als Ursachen kommen, neben Problemen mit der Aufnahmegeometrie, auch Ungenauigkeiten bei der Beleuchtung in Frage.
Um die Qualität der Orientierungen zu verbessern, wurde eine zweite Aufnahmeserie erstellt, mit äußerster Sorgfalt bezüglich der Einstellungen und der Beleuchtung.

5 Die Datenbank

5.1 Struktur der Datenbank

Die Datenbank besteht aus einer Reihe von Dateien, die hierarchisch aufgebaut sind und aufeinander verweisen. Die „Wurzel“ ist die Objektdatei[7]. Sie enthält pro Objekt einen Eintrag, in dem der Objektname und Verweise auf eine pol–, rel– sowie sim–Datei angegeben sind (siehe Abb. 6). Durch Verwendung mehrerer Objektdateien können verschiedene Datenbanken angelegt werden, die gemeinsame Objekte haben, ohne daß die Daten der Objekte mehrfach abgelegt werden müssen.

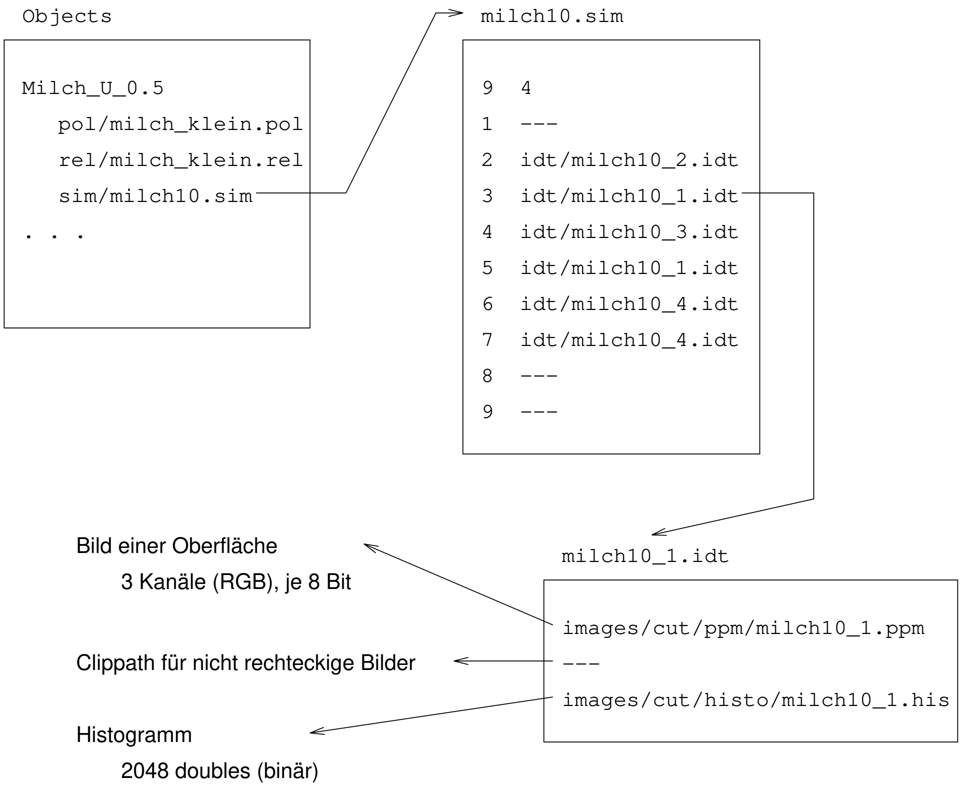


Abbildung 6: Struktur der Datenbank

In den pol–Dateien ist das dreidimensionale, in der Regel vereinfachte Modell des Objekts gespeichert. Die Modelle wurden hauptsächlich mit dem Programm `Polygraph` erzeugt, dessen Format übernommen wurde.
Die pol–Dateien werden momentan nicht direkt benutzt. Stattdessen wird

[7] Bei uns heißt sie `Objects`. Sie kann jedoch einen beliebigen Namen haben.

mit dem Programm `pol2rel` aus jeder pol–Datei eine rel–Datei generiert. Zu jeder pol–Datei gehört somit genau eine rel–Datei. Die rel–Datei enthält für jede Fläche des Objektmodells die Nachbarflächen und die dazugehörigen Winkel. Diese Daten werden für eine Konsistenzprüfung nach dem Matching verwendet (siehe Abschnitt 8.1.2, z.Z. noch nicht implementiert).
Die sim–Dateien[8] geben die Zuordnung von Objektflächen zu den Bildern in der Datenbank an. Für jedes Bild in der Datenbank existiert eine Beschreibung, seine idt–Datei[9]. Sie verweist auf die Bild–Datei und gegebenenfalls auf die Histogramm–Datei. Ist die Objektfläche nicht rechteckig, so muß noch angegeben werden, welcher Teil des Bildes in der Bild–Datei zur Fläche gehört. Dafür wird der sogenannte Clip–Path verwendet, der dann auch in der idt–Datei vermerkt ist. Der Clip–Path ist eine Liste von Koordinaten des Polygons, das die Fläche umschließt. Weitere charakteristische Bildmerkmale können in der idt–Datei ebenfalls gespeichert werden.
Die Aufteilung der Datenbank in mehrere Dateien dient nicht nur der Übersichtlichkeit sondern auch der Wiederverwendbarkeit. Objekte mit gleicher Form und Größe aber mit verschiedenen Aufdrucken haben die gleichen pol– und rel–Dateien und nur verschiedene sim–Dateien. Einzelne Bilder können innerhalb einer sim–Datei und auch in verschiedenen sim–Dateien mehrfach verwendet werden. So z.B., wenn verschiedene Flächen eines Objektes den gleichen Aufdruck haben oder wenn mehrere Objekte gleich bedruckte Flächen aufweisen. Ein typisches Beispiel ist der weiße Boden gleich großer Objekte.

5.2 Verzeichnisstruktur im Dateisystem

Es hat sich als praktisch erwiesen, die einzelnen Dateien der Datenbank in verschiedenen Verzeichnissen abzulegen. Als Beispiel ist in Abb. 7 die bei uns verwendete Verzeichnisstruktur angegeben.
Das Verzeichnis `camera` enthält die für die Datenbank aufgenommenen Bilder. Sie wurden für die Verwendung in der Datenbank noch bearbeitet (siehe Abschnitt 4.1) und dann im Verzeichnis `cut` abgelegt. Das Verzeichnis `camera` ist somit kein Bestandteil der Datenbank.
Für umfangreiche Datenbanken kann es sinnvoll sein, die einzelnen Verzeichnisse noch weiter zu unterteilen, z.B. durch Bildung von Gruppen innerhalb der Objekte. Die Geschwindigkeit des Einlesens der Datenbank wird dadurch (fast) nicht beeinflußt.

[8] sim steht für Surface IMages
[9] idt steht für Image DaTa

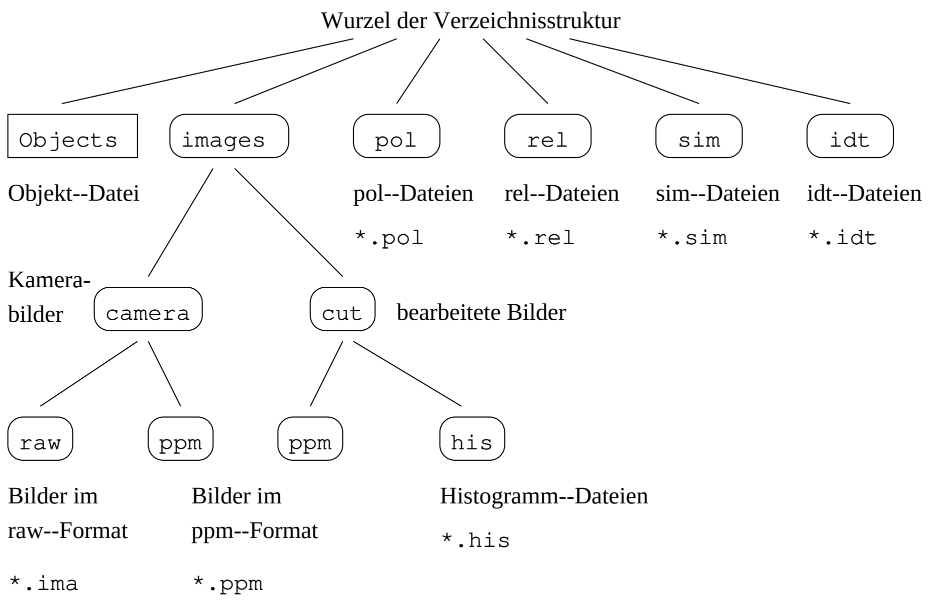


Abbildung 7: Verzeichnisstruktur der Datenbank

Alle Pfade in der Datenbank sind relativ zur Wurzel der Verzeichnisstruktur der Datenbank angegeben.

5.3 Formate der Datenbankdateien

5.3.1 Objektdatei

Die Objektdatei enthält pro Objekt einen einzeiligen Eintrag. Dieser Eintrag besteht aus dem Objektnamen und den Namen und Pfaden der pol–, rel– und sim–Dateien, jeweils durch White–Space getrennt. Alle Pfade sind relativ zur Wurzel der Datenbank. Dort sollte sich auch die Objektdatei befinden.
Da in der momentanen Programmversion pol– und rel–Dateien nicht benutzt werden, können sie in der Objektdatei durch jeweils drei Minuszeichen („`---`“) ersetzt werden. Ohne rel–Dateien stehen jedoch in der eingelesenen Datenbank keine Flächeninformationen zur Verfügung.

5.3.2 pol–Datei

Die pol–Dateien liegen im ASCII–Format vor. Alle Einträge sind Dezimalzahlen der Form *nnn.nnnnnn*. Einträge innerhalb einer Zeile werden durch White–Space getrennt.
In der ersten Zeile werden die Anzahl der Eckpunkte und die Anzahl der

Flächen angegeben. In den folgenden Zeilen werden die x–, y– und z–Koordinaten der Eckpunkte abgelegt — jeder Punkt in einer eigenen Zeile. Danach folgt die Liste der Flächen mit jeweils zwei Zeilen pro Fläche. In der jeweils ersten Zeile steht die Anzahl der Eckpunkte der Fläche. Danach folgen die Nummern der Eckpunkte in der Eckenliste. Die Zählung beginnt bei 1. In der jeweils zweiten Zeile stehen die Kantenlängen in Millimetern.

5.3.3 rel–Datei

Die rel–Dateien liegen ebenfalls im ASCII–Format vor. Alle Einträge sind natürliche Zahlen. Einträge innerhalb einer Zeile werden durch White–Space getrennt.
In der ersten Zeile steht die Anzahl der Flächen. Sie muß identisch sein mit der Anzahl der Flächen in der zugehörigen pol–Datei. Ab der zweiten Zeile beginnt die Liste der Flächen mit jeweils zwei Zeilen pro Fläche. In der jeweils ersten Zeile stehen die Nummer der Fläche laut pol–Datei[10], die Anzahl der angrenzenden Flächen (gleich der Anzahl der Eckpunkte in der pol–Datei) und die Nummern (laut pol–Datei) der angrenzenden Fläche. In der jeweils zweiten Zeile steht zu jeder angrenzenden Fläche der Winkel, den die Oberflächennormalen der Fläche und der angrenzenden Fläche bilden.

5.3.4 sim–Datei

Auch die sim–Dateien liegen im ASCII–Format vor. Einträge innerhalb einer Zeile werden durch White–Space getrennt.
In der ersten Zeile einer sim–Datei stehen die Anzahl der Flächen und die Anzahl verschiedener Bilder. (Da bei gleichartig bedruckten Flächen nur ein Bild für beide Flächen gespeichert wird, wird in der sim–Datei ein Bild mehrfach eingetragen. Es wird jedoch nur das erste Auftreten eines Bildes in der sim–Datei gezählt.)
In den folgenden Zeilen erfolgt eine Zuordnung von Flächen zu Bildern, jeweils eine Fläche pro Zeile. Die Zeile wird mit der Nummer der Fläche laut pol–Datei eingeleitet. (Die Flächen müssen in aufsteigender Reihenfolge angegeben werden.) Danach wird das Bild durch Angabe des Namens und Pfads einer idt–Datei zugeordnet. Existiert zu einer Fläche kein Bild in der Datenbank, so werden stattdessen drei Minuszeichen (ohne Leerzeichen o.ä. dazwischen) eingetragen.

[10] Die Numerierung beginnt mit 1. Die Flächen müssen in aufsteigender Reihenfolge angegeben werden.

5.3.5 idt–Datei

Eine idt–Datei ist die Beschreibung zu einem Datenbankbild. Sie besteht aus vier Zeilen. In der ersten Zeile sind Name und Pfad der Bild–Datei und in der dritten Zeile Name und Pfad der Histogramm–Datei angegeben. Die Pfade sind, wie bereits erwähnt, relativ zum Pfad der Objektdatei. Ist zu dem Bild kein Histogramm gespeichert, so werden in der dritten Zeile drei Minuszeichen (ohne Leerzeichen o.ä. davor, dazwischen oder danach) angegeben.
In der zweiten Zeile wird der sogenannte Clip–Path angegeben. Er besteht aus der Anzahl der Punkte gefolgt von den durch Leerzeichen getrennten Koordinatenangaben. Dabei sind x– und y–Anteil wiederum durch Leerzeichen getrennte natürliche Zahlen. Ist der Clip–Path identisch mit dem Rand des Bildes in der Bild–Datei, so muß er nicht angegeben werden. Stattdessen können drei Minuszeichen (wie oben) eingetragen.
In der vierten Zeile werden durch Leerzeichen getrennt „Main Hue“ und „Average Hue“ angegeben. „Main Hue“ ist eine natürliche Zahl im Bereich von 0 bis 255 und „Average Hue“ eine reelle Zahl im Bereich von 0.0 bis 255.0. Die beiden Werte dienen der Vorauswahl beim Matching (siehe dazu Abschnitt 8.1.1).
Hinweis: In einer früheren Programmversion war die vierte Zeile der idt–Datei noch nicht vorhanden. Daher kann zur Verwendung älterer Datenbanken auch eine Programmversion für dreizeilige idt–Dateien generiert werden.

5.3.6 Bild

Als Format für die Bild–Dateien wurde PPM (für Farbbilder) bzw. PGM (für Grauwertbilder) gewählt. Farbbilder werden dabei als Bilder mit drei Grauwert–Kanälen betrachtet, die den Rot–, Grün– und Blau–Anteil repräsentieren. Pro Kanal wird eine maximale Auflösung von 8 Bit zugelassen.
Für beide Formate ist eine binäre und eine ASCII–Variante definiert. Von unseren Programmen wird aber nur die binäre Variante unterstützt, da das Laden und Speichern so schneller möglich ist[11]. Da das Binärformat byteorientiert ist, ist es auch (weitgehend) maschinenunabhängig.
Eine Bild–Datei beginnt mit einem aus einem ASCII–Text bestehenden Header. Alle mit `#` beginnenden Zeilen sind dabei Kommentare und werden ignoriert. Die Angabe der Zeilennummern im nachfolgenden Text bezieht daher Kommentarzeilen *nicht* mit ein.

[11] Es sind weniger Bytes zwischen Hauptspeicher und Sekundärspeicher zu transferieren, und die Konvertierung von der bzw. in die Maschinenrepräsentation entfällt.

In der ersten Zeile steht der sogenannte Magic Code. Er ist für Grauwertbilder `P5` und für Farbbilder `P6`.
In der zweiten Zeile geben zwei natürliche Zahlen die Bildgröße an, zuerst die Breite (x) und dann die Höhe (y). Die beiden Zahlen sind durch White–Space voneinander getrennt.
In der dritten Zeile gibt eine natürliche Zahl den maximalen Wert an, der in einem Kanal auftreten kann. Bei uns darf dieser Wert 255 (aufgrund der maximalen Auflösung von 8 Bit) nicht überschreiten.

5.3.7 Histogramm

Der Aufbau der Histogramm–Datei orientiert sich an dem der Bild–Datei. Es steht wahlweise ein binäres oder ein ASCII–Format zur Verfügung, allerdings wird von den Programmen bisher nur das Binärformat unterstützt. Das Binärformat läßt sich schneller einlesen, ist jedoch von der internen Repräsentation des Typs `double` auf der jeweiligen Maschine abhängig. Da die Programme i.d.R. aber nur auf gleichartigen Maschinen zum Einsatz kommen werden, ist dies kein Nachteil.
Sowohl beim Binär- als auch beim ASCII–Format enthalten die ersten vier Zeilen Headerinformationen als ASCII–Text. In der ersten Zeile steht der Identifikationsstring. Die zweite Zeile enthält einen String, der das Farbmodell angibt. Die dritte und vierte Zeile geben die Dimension und die Anzahl der Bins im Histogramm an. Zur Zeit werden nur eindimensionale Histogramme unterstützt. Statt ein dreidimensionales Histogramm (entsprechend den drei Farbkanälen) zu verwenden, wird aus den drei Indizes ein Index gebildet, so daß ein eindimensionales Histogramm entsteht. Die Anzahl der Bins ist beliebig, muß aber zur in der fünften Zeile angegebenen Auflösung „passen“. (Das wird z.Z. jedoch nicht überprüft.) In der fünften Zeile stehen die Auflösungen der drei Farbkanäle als drei natürliche Zahlen im Bereich von 0 bis 8. Sie geben die Anzahl der zu verwendenden Bits (vom MSB aus gezählt) des jeweiligen Farbkanals an. Eine Auflösung von 0 bedeutet, daß dieser Kanal überhaupt nicht in das Histogramm eingeht, 8 bedeutet die volle Auflösung.
Nach dem Header (unmittelbar nach dem letzten Byte der Zeilenendekennung der fünften Zeile) folgen die eigentlichen Histogrammdaten. Die Zahlenwerte der einzelnen Bins werden in der Reihenfolge der Bins (aufsteigend) geschrieben, beim Binärformat ohne Trennzeichen, beim ASCII–Format durch White–Space getrennt. Im Binärformat besteht eine Zahl aus der entsprechenden Anzahl Bytes eines `double`s in maschinenabhängiger Reihenfolge. Im ASCII–Format wird die Zahl in der für Werte vom Typ

Zeile	Name	Inhalt	Bedeutung	Einschränkungen
1.	Identifikation	„HISTO-BIN“	Binär–Format	
		„HISTO-ASCII“	ASCII–Format	nicht realisiert
2.	Farbmodell	„RGB“	Standard RGB	
		„HSI“	Standard HSI	
		„RG-BY-WK“	Gegenfarbenmodell	
3.	Dimension	nat. Zahl ≥ 1		nur 1
4.	Anzahl der Histogramm–Bins	nat. Zahl ≥ 1		
5.	Auflösung	3 nat. Zahlen $0 \geq n \geq 8$	für jeden Kanal in Bits	

Tabelle 3: Aufbau des Headers einer Histogramm–Datei

`double` in C üblichen Notation geschrieben.

5.4 Anmerkung zu extrem großen Datenbanken

Bei einer größeren Anzahl von Objekten in der Datenbank spielt der Speicherplatzbedarf eine zunehmende Rolle. Es könnte dann notwendig sein, ein komprimiertes Bild– und Histogramm–Dateiformat zu wählen. Allerdings sollte der Dekompressionsvorgang beim Laden nicht die für den physikalischen Transfer von der Platte benötigte Zeit übersteigen, um keine Performance–Verluste zu verursachen.
Der Speicherplatzbedarf der Histogramme könnte auch durch die Verwendung des Typs `float` statt `double` auf die Hälfte reduziert werden. Die Anzahl signifikanter Stellen des Typs `float` ist für unsere Zwecke ausreichend. (Die Histogramme könnten dann eventuell ständig im Speicher gehalten werden, siehe Abschnitt 9.3.2 auf Seite 38.)

6 Ablauf der Objekterkennung

Ausgehend von der Aufnahme des zu erkennenden Objekts wird in diesem Abschnitt kurz der Ablauf der Erkennung beschrieben. Eine Skizze des Ablaufs zeigt Abb. 8.

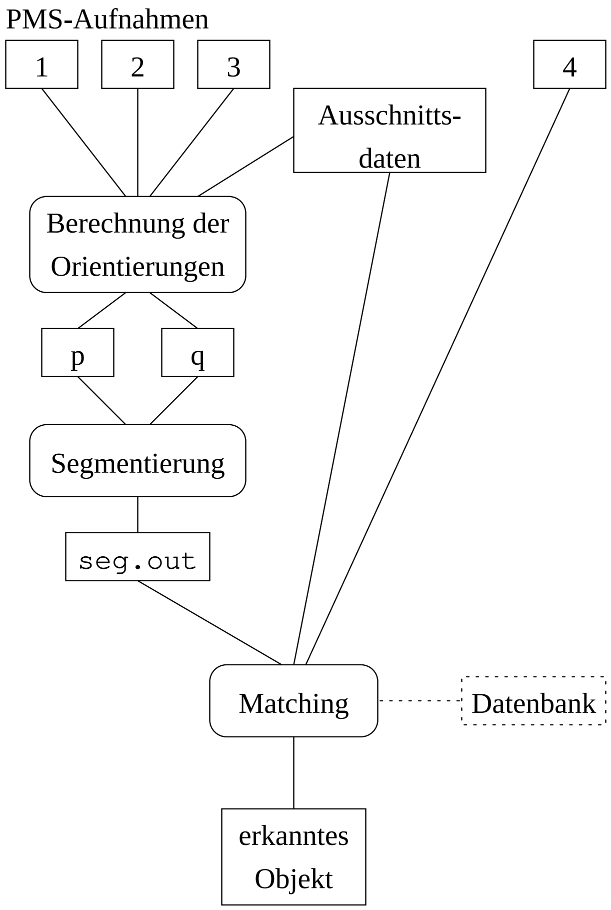


Abbildung 8: „Datenfluß“ bei der Erkennung von Objekten

Ausgehend von den PMS–Aufnahmen (wie in Abschnitt 4.2 beschrieben) werden die Orientierungen der Flächen des Objekts an allen ausreichend beleuchteten Punkten berechnet. Dazu wurden an der TU Berlin entwickelte Programme verwendet. Aufgrund der im Umgang mit Photometric–Stereo–Verfahren und ihren Programmen nötigen Erfahrungen haben uns Tapani Hegewald und Karsten Schlüns bei der Berechnung der Orientierungen der zweiten Serie sehr unterstützt. Arno Mitritz hat für uns die Orientierungen der ersten Serie berechnet, in der die Orientierungen teilweise nur für einen (manuell gewählten) Auschnitt der Aufnahme berechnet

wurden.
Die Orientierungen werden, jeweils x– und y–Anteil (p bzw. q genannt) getrennt, in zwei Dateien gespeichert, die als Eingabe für die anschließende Segmentierung dienen. Bei der Segmentierung werden die einzelnen Punkte anhand ihrer Orientierungen zu Flächen zusammengefaßt. Eine genaue Beschreibung der Segmentierung ist in Abschnitt 7 zu finden.
Die Ergebnisse der Segmentierung werden zusammen mit einer Farbaufnahme des zu erkennenden Objekts (und gegebenenfalls den Koordinaten des Ausschnitts) dem Matching zugeführt, das mit Hilfe der Datenbank das aufgenommene Objekt einem Objekt der Datenbank zuordnet.

7 Segmentierung

Das zu erkennende Objekt wird segmentiert, um die genaue Lage der Flächen zu bestimmen. Dabei verwenden wir das Programm `seg` von Arno Mitritz, das wir an unser Projekt anpaßten.

7.1 Das Programm seg

Da unsere Objekterkennung automatisch abläuft, ist dieses Programm nicht mehr interaktiv, sondern wird mit einem für unsere Objekte passenden vorinitialisierten Parametervektor ausgeführt. Die Parameter von `seg` können über die Kommandozeile verändert werden. Der allgemeine Aufruf sieht jetzt folgendermaßen aus:

`seg [-nographics] [-(w|d|f|p|b|v|n|e|t)`*Wert*`]` *Dateiname*,

wobei der Datei–Name zwei Raw–float–Bilder identifiziert, ein Bild, das auf `.p` endet und ein Bild, das auf `.q` endet. Dabei wird der Dateiname ohne `.p` bzw. `.q` angegeben.
Um die Rechenzeit zu verkürzen, entfallen (durch Angabe von `-nographics`) wahlweise die grafischen Ausgaben des Programms. (Damit entfällt auch insgesamt die Notwendigkeit eines grafischen Displays.) Nach einer Segmentierung wird eine Datei `seg.out` im aktuellen Verzeichnis erzeugt, in der Informationen über die segmentierten Flächen zur Weiterverarbeitung gespeichert sind. Darauf wird im übernächsten Abschnitt näher eingegangen.

7.2 Wahl der Parameter

Unsere Objekte sind — wie im Abschnitt 1 beschrieben — Objekte, die mehr oder weniger starke Verformungen gegenüber einem synthetischen Objekt gleicher Topologie aufweisen. Der Segmentierer hat die Aufgabe, trotz

fehlerhafter Bereiche die Flächenform und -größe bestmöglich zu rekonstruieren. Die Parameter müssen hierbei sehr feinfühlig abgestimmt werden. Dabei wird gefordert, daß ein einziger Parametervektor für beliebige Objekte verwertbare Ergebnisse liefert, da die Objekterkennung automatisch abläuft, ohne eine interaktive Anpassung der Paramter für ein bestimmtes Objekt.
Zum Matching werden nur die größeren Flächen herangezogen, so daß die Parameter so gewählt wurden, daß große Flächen gut, kleinere aber schlecht bzw. gar nicht extrahiert werden. Kleine Flächen — wie z.B. der Giebel einer Milchtüte — eignen sich zum Matching kaum, da die Fehler hierbei einen hohen Einfluß haben und die Flächen oft keine bedeutenden Aufdrucke besitzen (z.B die Bodenfläche einer Milchtüte).
Wir verwenden folgenden Parametervektor, der für unsere Auswahl von Objekten die bestmöglichen Ergebnisse liefert:

```
Nadelkarten glätten (Nein)
Max. Nadelabweichung innerhalb einer Fläche (28.0 Grad)
Max. Standardabweichung in planaren Flächen (6.0 Grad)
Minimale Flächengröße (700)
Maximale Konturentfernung von polygon. Approx. (7)
Flächen ausbalancieren (50)
Max. Winkel für Verdeckungskanten (15.0 Grad)
Nadelkartentyp (real)
Eliminierung von Ausläufern der Dicke 1 (Ja)
Tiefenberechnung mit konstanter Linienzeichung
Segmentierungsbereich (1, 254, 1 254)
```

Es lassen sich alle Objekte segmentieren, wobei die Qualität der Ergebnisse schwankt. Probleme ergeben sich vor allem bei Objekten mit Rundungen, wo gekrümmte Flächen teilweise sehr unregelmäßig approximiert werden. Desweiteren sind Objekte mit dunklen Aufdrucken, wie z.B. die Nadelkarte des Objekts `Traubensaft_1.0`, nicht geeignet. Der Segmentierer kann die Begrenzungen der einzelnen Flächen nicht erkennen, da für die dunklen Bereiche keine pq–Werte zur Verfügung stehen. Bei der zweiten Serie konnte dieses Problem, wie in Abschnitt 4.2 und 4.3 beschrieben, gelöst werden.

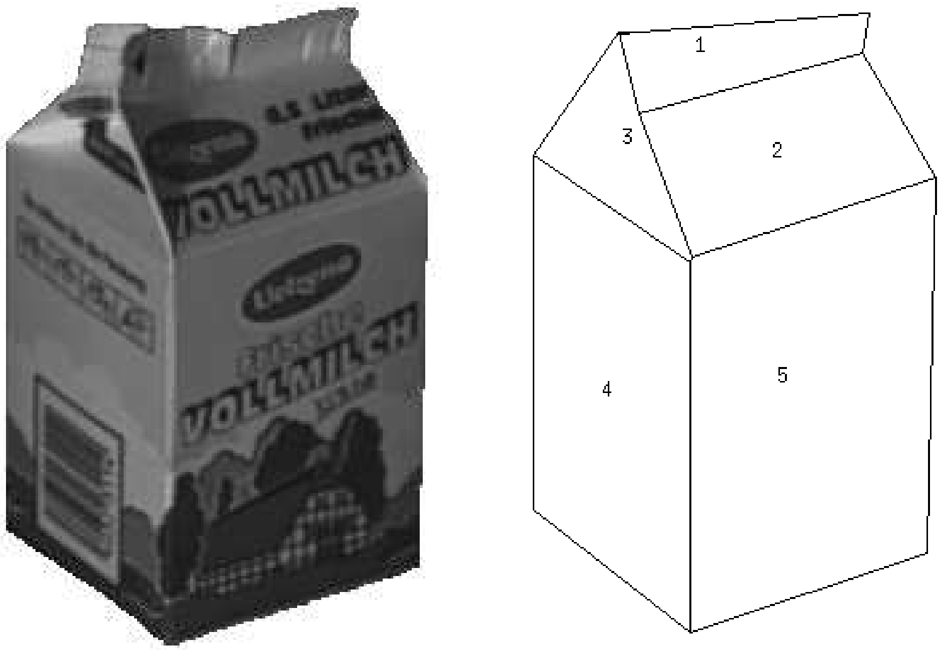


Abbildung 9: Milchverpackung: links aufgenommenes Bild (im Original farbig, Hintergrund manuell separiert), rechts Ergebnis der Segmentierung

7.3 Aufbau von `seg.out`

Das modifizierte Programm erzeugt nach jeder erfolgreichen Segmentierung eine Datei `seg.out`[12] im ASCII–Format, die die für unser Matching relevanten Informationen über die extrahierten Flächen enthält:
Die erste Zeile der Datei enthält die Anzahl der Flächen eines Objekts. Jede Folgezeile entält die Informationen für genau eine Fläche mit folgenden Angaben:

- durchschnittlicher p-Wert der Fläche
- durchschnittlicher q-Wert der Fläche
- Anzahl der Eckpunkte der Fläche
- x- und y-Koordinate des 1. Eckpunktes
- x- und y-Koordinate des 2. Eckpunktes

usw. Dabei sind alle Werte durch ein White–Space getrennt.

[12] Diese Datei wird vor einer eventuellen Korrektur der Orientierungen geschrieben.

8 Matching

Das Matching ist — wie in Abb. 8 auf Seite 19 zu sehen — der abschließende und zugleich auch umfangreichste Schritt. Ziel des Matchings ist es, zu jedem Objekt aus der Datenbank einen Wert anzugeben, der die Übereinstimmung mit dem zu erkennenden Objekt angibt. Das aufgenommene Objekt wird dann mit dem Objekt aus der Datenbank mit der besten Übereinstimmung identifiziert oder als nicht erkennbar zurückgewiesen.
Für das Matching verwenden wir die Farbhistogramme der Flächen sowie Template–Matching-Verfahren. Bei den Bildern der ersten Serie konnte das Template–Matching jedoch noch nicht eingesetzt werden, da die errechneten Orientierungen relativ schlecht waren (siehe auch Abschnitt 8.2.2).

8.1 Matching mittels Farbhistogrammen

Das Verfahren wurde von Michael J. Swain und Dana H. Ballard entwickelt und in [4] beschrieben. Es wurde an der TU Berlin bereits in einem Projekt von Rafał Sałustowicz, Christian Krüger und Alexander Bachem [3] eingesetzt. Von ihnen haben wir auch die Datenstrukturen und Teile der Funktionen zum Histogrammvergleich übernommen.
Für die zwei zu vergleichenden Bilder (bzw. Bildausschnitte) wird jeweils ein Farbhistogramm berechnet. Die Ähnlichkeit der beiden Histogramme wird dann mit Hilfe der Histogrammschnitt–Methode (histogram intersection) bestimmt. Dabei wird für jedes Bin das Minimum beider Histogramme bestimmt und über alle Bins summiert. Da wir relative Histogramme verwenden, ist eine weitere Normierung nicht erforderlich.
Der Einfachheit halber verwenden wir (wie in [3]) kein mehrdimensionales Histogramm, sondern ein eindimensionales, dessen Index eineindeutig aus den drei Farbkomponenten berechnet wird.
Wir haben drei Farbmodelle mit verschiedenen Auflösungen in den jeweils drei Farbkanälen untersucht. Als Farbmodelle haben wir neben dem Standard–RGB–Modell das HSI– und ein auf Gegenfarben beruhendes Farbmodell (Rot–Grün (RG), Blau–Gelb (BY) und Weiß–Schwarz (WK)) verwendet, das wir den Quellen [4] und [3] entnommen haben.
Swain und Ballard haben in [4], bezogen auf das Gegenfarbenmodell, eine weitgehende Unabhängigkeit der Ergebnisse von der Farbauflösung beschrieben, die wir nicht nachvollziehen konnten. In [3] wird beim Gegenfarbenmodell eine Auflösung von Rot–Grün (RG) und Blau–Gelb (BY) in 16 Stufen (4 Bits) und Weiß–Schwarz (WB) in 8 Stufen (3 Bits) verwendet. Die verringerte Farbauflösung dient, wie Sałustowicz, Krüger und Bachem

in [3] darlegen, vor allem dazu, das Verfahren toleranter gegenüber kleineren Farbschwankungen zu gestalten und die zu verarbeitende Datenmenge zu reduzieren.
Wir haben selbst verschiedene Farbauflösungen untersucht. Die Ergebnisse haben wir in Abschnitt 10.1 zusammengefaßt. Allgemein kann man sagen, daß eine höhere Farbauflösung eine bessere Unterscheidung der Farben gestattet, jedoch auch eine größere Beleuchtungsabhängigkeit bedingt. Die Farbauflösung sollte so klein wie möglich gewählt werden, so daß aber noch alle wesentlichen Farben der Objekte unterschieden werden können. Auch in Bezug auf den Speicheraufwand sind geringere Auflösungen von Vorteil.

8.1.1 Realisierung des Matchings

Da ein Vergleich der Histogramme aller Segment–Flächen[13] des zu erkennenden Objekts mit denen aller Flächen der Objekte der Datenbank zu zeitaufwendig ist, muß eine Vorauswahl in Frage kommender Flächen der Datenbank getroffen werden.
Kleine Segment–Flächen tragen wenig zum Matching–Ergebnis bei und können es sogar negativ beeinflussen, da sie wenig Informationen enthalten und der Segmentierer sie nicht klar extrahieren kann. Daher werden Segment–Flächen die kleiner als ein bestimmter wählbarer Bruchteil der größten Segment–Fläche sind von vornherein vom Matching ausgeschlossen.
Als weiteres Kriterium haben wir das sogenannte Edge–Area–Ratio, das Verhältnis zwischen längster Kante und Wurzel des Flächeninhalts der Fläche, gewählt. Es ist ein Maß für die „Länglichkeit" der Fläche. Weichen die Edge–Area–Ratios einer Segment–Fläche und einer Fläche der Datenbank um mehr als einen bestimmten Faktor voneinander ab, so wird diese Paarung von Flächen nicht weiter betrachtet, da sie offensichtlich nicht übereinstimmen können.
Für die Bestimmung des Edge–Area–Ratios der Segment–Fläche wird die entzerrte Fläche herangezogen. Für die Berechnung des Histogramms zum Vergleich kann hingegen auch die aufgenommene Fläche verwendet werden. Wir benutzen dabei die Invarianz des Histogramms gegenüber geringen Änderungen des Blickwinkels. Bei einer größeren Neigung der Fläche zur Bildebene sollte jedoch besser die entzerrte Fläche verwendet werden — oder noch besser: die Datenbankfläche entsprechend verzerrt werden, um den Informationsverlust bei der Bildaufnahme zu modellieren. Eine Ausrichtung

[13]Der besseren Verständlichkeit wegen werden die segmentierten Flächen des zu erkennenden Objekts hier Segmente (oder Segment–Flächen) genannt. Die Bezeichnung Flächen bezieht sich dann in der Regel auf die Flächen in der Datenbank.

ist für die Histogrammberechnung allerdings nicht notwendig, da die Histogramme invariant bezüglich Translation und Rotation um die optische Achse sind.
Für weitere Kriterien zur Vorauswahl wird der Hue–Wert (aus dem HSI–Farbmodell) verwendet. Der „Main Hue“ ist der Hue–Wert, der im Histogramm der Hue–Werte der Fläche den größten Peak hat. Es ist sehr unwahrscheinlich, daß zwei verschiedene Hue–Werte genau gleich oft vorkommen. In diesem Fall wird der kleinere der beiden Hue–Werte als „Main Hue“ definiert. Der „Average Hue“ ist der durchschnittliche Hue–Wert der Fläche (eine reelle Zahl).
Sowohl für die Flächen der Datenbank als auch für die Segment–Flächen werden „Average Hue“ und „Main Hue“ mit der vollen 8–Bit–Auflösung des Hue–Kanals berechnet. Für jede Paarung einer Segment– und einer Datenbank–Fläche wird die Differenz der „Average Hues“ und der „Main Hues“ berechnet. Bleiben beide unter dem jeweiligen Schwellenwert, so wird die Paarung für das Matching verwendet.
Die beiden „Hue–Kriterien“ wurden definiert, um nicht unnötig eine z.B. weitgehend rötliche Fläche mit einer weitgehend bläulichen zu vergleichen, da beide kaum übereinstimmen werden. (An dieser Stelle spielt die Gleichheit der Lichtfarbe der Beleuchtung bei der Aufnahme der Datenbankbilder und der Bilder für die Tests eine große Rolle.) Wir haben zwei Kriterien definiert, um zu untersuchen, welches besser geeignet ist.
Im Ergebnis haben wir feststellen können, daß beide Kriterien annähernd gleichwertig sind und durch die Verwendung beider Kriterien keine signifikanten Verbesserungen zu erreichen sind. Vor allem ergibt sich jedoch eine Verringerung der Rechenzeit, da weniger Flächen–Paarungen einem Matching zu unterziehen sind. Allerdings erhöht sich auch wieder der Rechenaufwand durch die Bestimmung der Kriterien, falls das Matching nicht im HSI–Farbmodell erfolgt. In diesem Fall sind zusätzliche Farbtransformationen notwendig, wobei jedoch optimiert werden kann, da nur der Hue–Wert benötigt wird.

8.1.2 Erweiterungsmöglichkeiten

Konsistenzprüfung

Nach den Ergebnissen des Matchings der Segment–Flächen mit den Datenbank–Flächen könnten die Ergebnisse noch durch Konsistenzprüfungen verbessert werden. Wenn z.B. eine Segment–Fläche am besten zu einer bestimmten Fläche eines Objekts der Datenbank paßt und eine andere Segment–Fläche jedoch besser zu einer Fläche eines anderen Objekts, dann

wäre zu überprüfen, welches der beiden Objekte insgesamt besser mit dem zu erkennenden Objekt übereinstimmt. Das jetzige Verfahren realisiert dies nur indirekt: Es werden die Match–Werte für jedes Objekt summiert, so daß dann das Objekt mit dem höchsten Wert meist auch das mit der größten Übereinstimmung ist.
Eine weitere Konsistenzprüfung könnte anhand der Nachbarschaften und der Winkel zwischen benachbarten Flächen erfolgen: Benachbarte Segment–Flächen sollten bei einem korrekt erkannten Objekt auch benachbarten Flächen des Objekts zugeordnet werden, und die Winkel zwischen den Segment–Flächen sollten annähernd gleich den Winkeln zwischen den Flächen des Objekts sein. Dabei sollte sich die zulässige Abweichung der Winkel an der Genauigkeit des Photometric–Stereo–Verfahrens orientieren, bei unseren Tests ca. 5–10 Grad.

Berücksichtigung der Verzerrung

Eine Verbesserung der Ergebnisse ist zu erwarten, wenn man die perspektivische Verzerrung der Flächen berücksichtigt. In der jetzigen Implementierung wird das Histogramm der unverzerrten Fläche aus der Datenbank mit dem der verzerrt aufgenommenen Fläche verglichen. Dabei können aufgrund der Verzerrung signifikante Farbanteile fehlen[14], was zu Fehlern beim Matching führen kann.
Es wäre keine Lösung, die aufgenommene Fläche zu entzerren, da dabei keine zusätzliche Information gewonnen wird. (Die Farbe der zusätzlichen Punkte kann nur aus der Farbe der Nachbarpunkte interpoliert werden.) Daher muß die Fläche aus der Datenbank entsprechend der aufgenommenen Fläche verzerrt werden, so daß die gleiche Informationsreduktion eintritt und die Histogramme besser vergleichbar werden. Allerdings könnten dann keine vorberechneten Histogramme mehr verwendet werden, da jede Datenbank–Fläche für jede Segment–Fläche erneut verzerrt werden muß, siehe Abschnitt 8.2.1 auf Seite 28. (Die Berechnung der Histogramme ist jedoch relativ schnell. Ein signifikanter Unterschied würde sich erst ergeben, wenn alle vorberechneten Histogramme ständig im Speicher gehalten würden, siehe Abschnitt 9.3.2 auf Seite 38.)

8.2 Template-Matching

Neben dem Histogramm–Matching untersuchten wir als zweite Methode den pixelweisen Vergleich der Flächen (Template–Matching). Dafür muß die zu

[14] Wir haben dies jedoch bisher nicht beobachten können.

erkennende Fläche entzerrt, skaliert und in die gleiche Lage gebracht werden, in der sich die entsprechende Fläche in der Datenbank befindet.

8.2.1 Voraussetzungen

Die mit dem PHOTOMETRIC–STEREO–Verfahren aufgenommenen Objekte in beliebiger Lage müssen zunächst so gedreht werden, daß jeweils eine Fläche unverzerrt, gemäß der Abb. 3 auf Seite 4 dargestellt wird. Diese Rotation ist durch die pq–Werte der Flächen, die vom Segmentierer ermittelt werden, eindeutig bestimmt. Nach dieser Drehung erhält man die Rotated Input Images (RIIs), die sich bis auf eine Rotation um die z–Achse nicht von den entsprechenden Flächen in der Datenbank unterscheiden.

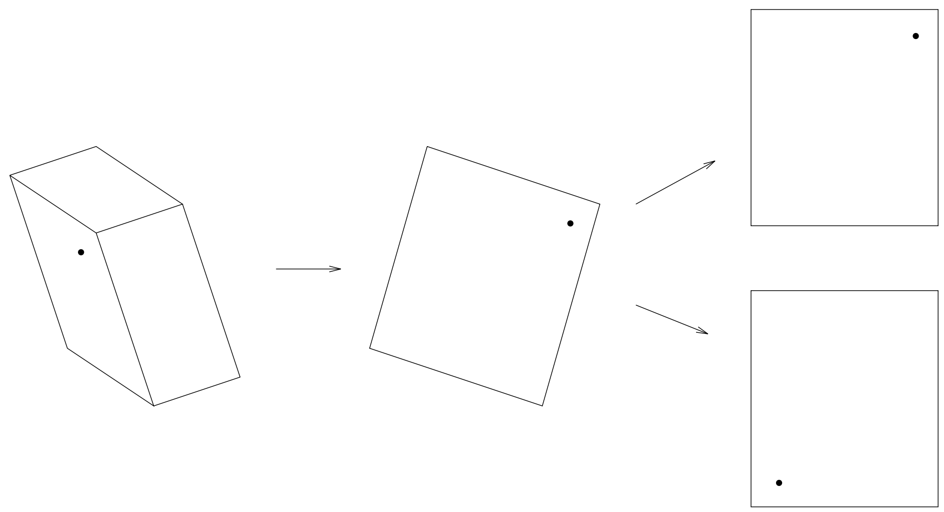


Abbildung 10: Entzerrung und Ausrichtung der Flächen

Dies korrigiert man, indem die längste Kante des RIIs parallel zur längsten Kante der Datenbankfläche ausgerichtet wird. Damit reduziert sich das Matching auf genau die Anzahl von Fällen, wie es gleichlange längste Kanten gibt. Bei einer rechteckigen Fläche sind es zwei Fälle, wie die Abb. 10 zeigt. Tatsächlich kann sich die Anzahl der Fälle erhöhen, da alle diejenigen Kanten als längste Kanten aufgefaßt werden, die annähernd so lang sind wie die absolut längste Kante, um leichte Fehler bei der Segmentierung auszugleichen.
Eine Alternative dazu ist die Ausrichtung entlang der Hauptachse der Fläche. Wenn man die Hauptachse mit der Parallelen zur y–Achse, die durch die Bildmitte verläuft, in Übereinstimmung bringt, sind nur noch zwei Fälle

zu betrachten, die sich durch eine Drehung um 180 Grad um die z–Achse unterscheiden. Da der Rechenaufwand zur Bestimmmung der Hauptachse erheblich höher ist, wird dieses Verfahren im Programm nicht verwendet.
In beiden Varianten wird als gegeben vorausgesetzt, daß die Bilder der Datenbank bereits ausgerichtete Flächen enthalten.
Damit Objektfläche und Datenbankfläche, sofern sie zueinander passen, für das Template–Matching deckungsgleich übereinanderliegen, erfolgt abschließend eine Skalierung, um jetzt bereits geometrisch ähnliche Flächen noch auf die gleiche Größe zu transformieren.

Alternative: Verzerrung der Bilder der Datenbank

Die verzerrt aufgenommenen Flächen enthalten (aufgrund der geringeren Pixelanzahl) weniger (Farb)Information als die Flächen der Datenbank. Statt die aufgenommenen Flächen zu entzerren und die fehlenden Pixel zu interpolieren, können auch die Flächen aus der Datenbank jeweils entsprechend der Lage der aufgenommenen Fläche verzerrt werden. Der Informationsgehalt entspricht jetzt genau dem der verzerrten Aufnahme. Insofern ist ein besseres Ergebnis zu erwarten. Allerdings steigt auch der Rechenaufwand erheblich, denn statt einer geringen Anzahl von Segment–Flächen muß nun die relativ große Anzahl von Flächen aus der Datenbank einer affinen Transformation unterzogen werden.
Das Problem der Ausrichtung stellt sich hier ebenso. Eine Ausrichtung an der längsten Kante ist genauso vorstellbar. Eine bezüglich der Abweichungen der Verzerrung günstigere Variante ist eventuell die bereits erwähnte Ausrichtung mit dem Schwerpunkt in der Bildmitte und der Hauptachse parallel zur y–Achse.

8.2.2 Oberflächennormalen

Wie im Abschnitt 6 bereits beschrieben, wurden die Nadelkarten (pq–Bilder) mit Hilfe des Photometric–Stereo–Verfahrens berechnet.
Mit der ersten Aufnahmeserie von ca. 15 verschiedenen Objekten konnten die pq–Werte der Flächen nur unzureichend rekonstruiert werden und genügten somit nicht den Anforderungen an das Template–Matching, das eine genaue Entzerrung des aufgenommenen Bildes erfordert, um die einzelnen Pixel der Bilder vergleichen zu können.
Die folgende Abbildung zeigt das Beispiel einer Milchtüte. Links befindet sich die Fläche vor der Entzerrung, in der Mitte die Ergebnisfläche nach der Entzerrung mit den uns zur Verfügung stehenden pq–Bildern. Rechts

ist die Fläche in Originalform dargestellt, also so, wie unsere Ergebnisfläche mit genauen pq–Werten aussehen würde.

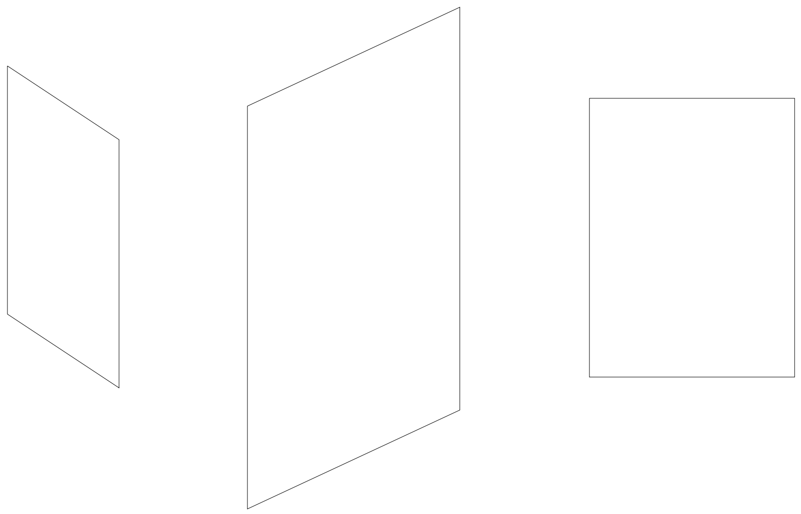


Abbildung 11: links: aufgenommene Fläche, Mitte: entzerrte Fläche, rechts: Originalfläche

Hierbei werden die Grenzen des PHOTOMETRIC–STEREO–Verfahrens deutlich, welches für die 3D-Rekonstruktion von natürlichen Objekten (z.B. Personen, Gesichter, etc.) entwickelt wurde, wo keine exakten pq–Werte erforderlich sind.
Wie im Abschnitt 4 beschrieben, wurde eine zweite Aufnahmeserie von jetzt insgesamt 29 verschiedenen Objekten erstellt; zum einen, um das Histogramm–Matching mit einer größeren Datenbank zu verifizieren, und zum anderen, um das Template–Matching anwenden zu können. Dabei wurden die Aufnahmen mit äußerster Sorgfalt durchgeführt, um bestmögliche pq–Werte zu erhalten.
Die Resultate der Bemühungen werden im Abschnitt 10.2 erläutert.

8.2.3 Matching

Für die weiteren Betrachtungen steht jetzt eine entzerrte Fläche zur Verfügung, die ausgerichtet über der Datenbankfläche liegt, mit welcher der pixelweise Vergleich stattfindet.

Vorauswahl durch topologische Merkmale

Um das Verfahren performant zu halten, ist eine Vorauswahl nötig, damit nicht jede Fläche mit jeder verglichen wird. Prizipiell kommen die im Abschnitt 8.1.1 verwendeten Verfahren zur Vorauswahl auch hier zum Einsatz. Überdecken sich Objektfläche und Datenbankfläche nicht ausreichend, so wird das Matching vorzeitig abgebrochen.

Unterabtastung der Flächen

Sind alle Kriterien der Vorauswahl erfüllt, erfolgt der pixelweise Vergleich beider Flächen, wobei beispielsweise bei rechteckigen Flächen der Vergleich mit einer um 180 Grad gedrehten Fläche wiederholt werden muß (vgl. Abb. 10). Vergleicht man zwei Flächen mit einer Größe von beispielsweise 300x300 Pixel, so sind 270000 Vergleiche nötig, 90000 pro Farbkanal. Bei einer hohen Anzahl von Datenbankobjekten ist dies ein äußerst zeitintensives Verfahren. Diese Zeit kann mittels Unterabtastung leicht verringert werden: Vergleicht man nur jedes achte Pixel, so beträgt der Zeitgewinn Faktor 64. Eine Unterabtastung in diesem Rahmen ist ohne nennenswerten Qualitätsverlust möglich, da die Bildinhalte größtenteils niederfrequent sind.
Für wenige Flächen, die in die engere Auswahl gehören, bzw. für Flächen, bei denen eine eindeutige Zuordnung beim Vergleich mit Unterabtastung nicht möglich war, kann ein zweiter Vergleich mit allen Pixeln zur Auswertung herangezogen werden.

Verwendung der Gauss–Funktion

Eine weitere Möglichkeit, genauere Matching–Ergebnisse zu erzielen, ist die Verwendung der zweidimensionalen Gauss–Funktion (Gauss–Glocke) zur Wichtung der Differenzen in der Umgebung der Pixel. Dies bedeutet, daß ein Pixel der Datenbankfläche nicht nur mit dem korrespondierenden Pixel der Objektfläche verglichen wird, sondern zusätzlich mit seinen Nachbarn. Die Wichtung wird mit zunehmendem geometrischen Abstand geringer, entsprechend der zweidimensionalen Gauss–Funktion. Damit können in beschränktem Maße Fehler verringert werden, die durch hochfrequentes Punktrauschen und durch leichte Verzerrungen, Verschiebungen und Verdrehungen entstehen. Leichte Verzerrungen sind bei unseren Objekten typisch, da die Oberflächen immer leicht gekrümmt oder verformt sind, auch bei behutsam behandelten Objekten.
Für die praktische Durchführung der Objekterkennung ist dieses Verfahren nicht relevant, da der Zeitaufwand mit der heute verfügbaren Rechentechnik unverhältnismäßig hoch ist.

Verwendung eines adaptiven Verfahrens

Hier wird das Problem aus dem vorherigen Abschnitt nochmal aufgegriffen, kleine Verzerrungen, Verschiebungen und Verdrehungen auszugleichen, mit der Forderung, den Rechenaufwand in Grenzen zu halten.
Die Idee besteht darin, einen Punkt der Datenbankfläche nicht direkt mit dem korrespondierenden Punkt der Objektfläche zu vergleichen, sondern ihn in einer bestimmten Umgebung zu suchen. So kann für einen Punkt lokal ein Verschiebungsvektor gefunden werden, der kleine Verschiebungen ausgleicht. Beim Vergleich der nächsten Pixel wird dann dieser Verschiebungsvektor berücksichtigt, so daß ein pixelgenaues Matching erfolgt.
Dabei ist bei der Bestimmung des Verschiebungsvektors besondere Vorsicht geboten. Eine Anpassung dieses Vektors bei jedem Pixelvergleich führt bei unseren realen Objekten zu falschen Vektoren. (Statt zu konvergieren, verhält sich das System dann chaotisch.) Vielmehr sollte zunächst über eine Vielzahl von Pixeln der Verschiebungsvektor berechnet werden, und erst, wenn sich über viele Vergleiche hinweg ein konstanter Vektor einstellt, sollte dieser Berücksichtigung finden.

9 Die Programme

In diesem Abschnitt wird die Verwendung der von uns im Rahmen des Projekts entwickelten Programme beschrieben. Das von uns modifizierte Programm `seg` wurde bereits in den Abschnitten 7.1 und 7.2 beschrieben.

9.1 `pol2rel`

Wie in Abschnitt 5.1 erwähnt, wird das Programm `pol2rel` für die Erstellung der Datenbank benutzt. `pol2rel` liest eine pol–Datei ein und erzeugt die zugehörige rel–Datei. Zu den Formaten beider Dateien siehe Abschnitt 5.3.2 bzw. 5.3.3.
Aufruf des Programms:

`pol2rel` *pol-Datei rel-Datei*

Die Quelltexte befinden sich im Verzeichnis `programs/pol2rel`.

9.2 `mkhist`

Auch das Programm `mkhist` wird für die Erstellung der Datenbank benutzt. Es berechnet Histogramme im Gegenfarben- und im HSI–Modell für ein Bild und speichert es im in Abschnitt 5.3.7 angegebenen Binär–Format. Die Einstellungen zur Auflösung werden, soweit sie nicht den Vorgabewerten entsprechen, der Parameter–Datei entnommen.
`mkhist` berechnet die Histogramme für ein Bild oder für eine Fläche eines Objekts der Datenbank. Der bei nicht–rechteckigen Flächen in der idt–Datei angegebene Clip–Path (siehe auch Abschnitt 5.3.5) wird berücksichtigt, falls die idt–Datei angegeben wurde.
Aufruf des Programms:

`mkhist` *Bild-Datei*
Histogramm-Datei(RGBYWK) Histogramm-Datei(HSI)
Parameter-Datei [*idt-Datei*]

Die Angabe der idt–Datei ist optional. Wird sie nicht angegeben, so wird davon ausgegangen, daß das Bild von der Fläche vollständig ausgefüllt wird. Ansonsten wird der Clip–Path der idt–Datei entnommen.
Die Quelltexte stehen im Verzeichnis `programs/match`, da ein Teil des Quelltextes von `poly_match` mitbenutzt wird.

9.3 `poly_match`

Dieses Programm realisiert die Erkennung der Objekte. Es liest zunächst eine Datenbank ein und versucht dann ein oder mehrere Objekte mit Ob-

jekten aus der Datenbank zu identifizieren.
Aufruf des Programms:

`poly_match` *`Datenbank-Pfad Objektdatei Parameter-Datei Segment-Datei Bild-Datei Daten-Datei`*

oder

`poly_match` *`Datenbank-Pfad Objektdatei Parameter-Datei Pipe`*

Das erste Argument beim Aufruf ist der Pfad der Datenbank (ohne abschließenden Schrägstrich „`/`“). Alle Pfadangaben in der Datenbank beziehen sich auf diesen Pfad (siehe Abschnitt 5.2). Das zweite Argument ist der Name der Objektdatei[15] — bei unserer Datenbank z.B. `Objects`.
Als drittes Argument ist der Name und der Pfad einer Parameter–Datei anzugeben. Parameter, die in dieser Datei nicht aufgelistet werden, behalten ihre Vorgabewerte. Somit kann, wenn nur mit den Vorgabewerten gearbeitet werden soll, einfach eine leere Datei (z.B. `/dev/null`) angegeben werden.
Bei der ersten Aufruf–Variante werden im vierten, fünften und sechsten Argument die Dateien zum zu erkennenden Objekt angegeben. Das vierte Argument besteht aus dem Namen und dem Pfad der Segment–Datei. Diese Datei beinhaltet die bei der Segmentierung gewonnen Daten im unter Abschnitt 7.3 beschriebenen Format. Das Programm `seg` nennt diese Datei grundsätzlich `seg.out`, `poly_match` kann jedoch eine Datei beliebigen Namens lesen.
Das fünfte Argument enthält den Namen und den Pfad der Bild–Datei, die mit allen drei Lichtquellen aufgenommen wurde. Das sechste Argument schließlich besteht aus dem Namen und dem Pfad der Daten–Datei, die die Koordinaten des Ausschnitts enthält, für den die Orientierungen („pq–Werte“) berechnet wurden. Ihr Name endet bei den von uns verwendeten Programmen zur Berechnung der Orientierungen auf `_Data_R_1`.
Da bei jedem Programmaufruf die Datenbank eingelesen wird, wäre es wünschenswert, mit einem Programmaufruf mehr als ein Objekt zu erkennen. Dafür ist die zweite Aufruf–Variante gedacht. Das vierte Argument („Pipe“) kann eine reguläre Datei, Named Pipe oder ähnliches sein, aus der die Namen und Pfade der Segment–, Bild– und Daten–Dateien ausgelesen werden. Wir haben aufgrund der inhärenten Synchronisationsmöglichkeit eine Named Pipe verwendet.
Die Named Pipe wird angelegt und `poly_match` gestartet. Dann wird (per Shell–Skript oder per Hand) die Datei `seg.out` erzeugt, und die Namen und Pfade der Segment–, Bild– und Daten–Dateien werden in dieser Reihenfolge

[15] Die Objektdatei sollte den im ersten Argument angegebenen Pfad besitzen. Andernfalls ist noch der relative Pfad der Objektdatei (ausgehend vom Pfad der Datenbank) anzugeben.

durch Leerzeichen getrennt in die Named Pipe geschrieben. Sobald das Programm alle drei Dateinamen gelesen hat, beginnt das Matching.

9.3.1 Format der Parameter–Datei

Jede Zeile der Parameter–Datei definiert eine Einstellung. In jeder Zeile steht zuerst der Name der Einstellung und dann durch White–Space getrennt ihr Wert, Leerzeilen werden ignoriert. Optional können nach einem weiterem White–Space noch Kommentare folgen. Ist kein Wert angegeben, wird die Einstellung auf ihren Vorgabewert gesetzt. Groß- und Kleinschreibung spielt generell keine Rolle.
Jede Einstellung entspricht einer in Abschnitt 9.3.5 erklärten Präprozessor–Konstanten. Die Einstellung trägt denselben Namen wie ihre zugehörige Präprozessor–Konstante. Sie definiert auch den Vorgabewert der Einstellung.
In Tabelle 4 sind die Namen der möglichen Einstellungen mit ihren zulässigen Werten aufgelistet. `double*` bedeutet dabei eine reelle Zahl > 0 und `double+` eine reelle Zahl ≥ 0. Wertebereiche ohne Dezimalpunkt bedeuten Werte aus dem angegebenen Bereich der natürlichen Zahlen und Wertebereiche mit Dezimalpunkt entsprechend aus dem Bereich der reellen Zahlen. Bei der Einstellung `COLOR_MODEL` sind die zulässigen Werte Zeichenketten.
Sobald ein ungültiger Name oder Wert erkannt wird, wird das Einlesen der Parameter–Datei abgebrochen.

Name der Einstellung	zulässige Werte
COLOR_MODEL	COL_MODEL_RGB, COL_MODEL_HSI oder COL_MODEL_RGBYWK
SEGMENT_FRACTION_LIMIT	double*
EA_RATIO_LIMIT	double*
MHUE_MAX_EPS	0 – 255
AHUE_MAX_EPS	0.0 – 255.0
NUM_VOTINGS	5
WEIGHT_CONST	double+
WEIGHT_AREA	double+
WEIGHT_PQ	double+
WEIGHT_MULTIPLIER_AREA	double+
WEIGHT_MULTIPLIER_PQ	double+
WEIGHT_HISTO	double+
WEIGHT_TEMPL	double+
AREA_THRESHOLD	double+
HISTO_BITS_RG	0 – 8
HISTO_BITS_BY	0 – 8
HISTO_BITS_WK	0 – 8
HISTO_BITS_H	0 – 8
HISTO_BITS_S	0 – 8
HISTO_BITS_I	0 – 8
HISTO_BITS_R	0 – 8
HISTO_BITS_G	0 – 8
HISTO_BITS_B	0 – 8
TEMPL_WEIGHT_RG	double+
TEMPL_WEIGHT_BY	double+
TEMPL_WEIGHT_WK	double+
TEMPL_WEIGHT_H	double+
TEMPL_WEIGHT_S	double+
TEMPL_WEIGHT_I	double+
TEMPL_WEIGHT_R	double+
TEMPL_WEIGHT_G	double+
TEMPL_WEIGHT_B	double+

Tabelle 4: Einstellungen in der Parameter–Datei

9.3.2 Der Programmablauf im Detail

Wie bereits erwähnt, gliedert sich der Programmablauf in zwei Hauptteile: in das Einlesen der Datenbank und die Erkennung von Objekten. (Es sind jeweils die wichtigsten zugehörigen Funktionen im Quelltext angegeben.)

Einlesen der Datenbank

Das Einlesen der Datenbank übernimmt die Funktion `read_database()`. Zuerst wird die Objektdatei eingelesen. Ihr werden dann die Namen und Pfade der weiteren einzulesenden Dateien entnommen. Von diesen werden zuerst alle rel–Dateien eingelesen (`read_rel()`–Aufrufe in `read_database()`) und verarbeitet und anschließend alle sim–Dateien (`read_sim()`–Aufrufe in `read_database()`). Die Funktion `read_sim()` liest nach der vollständigen Bearbeitung einer sim–Datei die zugehörigen idt–Dateien ein (`read_idt()`). Dabei werden bereits bei anderen Objekten eingelesene sim– und idt–Dateien nicht erneut bearbeitet. Alle Funktionen zum Einlesen der Datenbank befinden sich in der Datei `read_db.c`.

Einlesen der aufnahmespezifischen Daten

Die Erkennung beginnt mit dem Einlesen der Daten für das zu erkennende Objekt. Zuerst werden die Ausschnittsdaten gelesen (`read_offset()`) und anschließend die Ergebnisse der Segmentierung (`read_seg()`). Wurde zur Berechnung der Orientierungen ein verkleinertes Bild benutzt, was daran erkannt wird, daß das gesamte Bild als Ausschnitt in den Ausschnittsdaten definiert ist, so werden die Koordinaten der Segmentflächen gleich beim Einlesen umgerechnet. Ansonsten wird die Verschiebung später berücksichtigt.

Entzerrung

Danach werden die Segment–Flächen entzerrt (`top_view()`), das Bild des zu erkennenden Objekts eingelesen (`ppm_read_file()`), die Ausschnittsbilder der einzelnen Flächen erzeugt und die Flächeninhalte der Segment–Flächen bestimmt (`mk_cut_pics()`). Mit der Funktion `mk_trans_ea_ratios()` werden dann die Edge–Area–Ratios der entzerrten Segmente bestimmt. Dies muß in einer gesonderten Funktion geschehen, da für die entzerrten Segment–Flächen die längsten Kanten noch bestimmt werden müssen.
An dieser Stelle werden die internen Ausschnitts- und Segmentbilder ausgegeben. Dies sind im einzelnen: seg*n*.pgm, die Grauwertbilder der Segment–Flächen, seg*n*.ppm, die Farbbilder der ausgeschnittenen Segmente und sec*n*.pgm, die Grauwertbilder der entzerrten Segment–Flächen sowie sec*n*.ppm, die

Farbbilder der entzerrten ausgeschnittenen Segmente, wobei n die Nummer des Segments ist.

Matching

Der nächste Schritt bei der Erkennung des Objekts ist das Matching. Dabei werden für jede Paarung von einer Segment–Fläche und einer Fläche aus der Datenbank Werte für die mögliche Übereinstimmung berechnet (`do_match()`). Anschließend müssen aus diesen Werten noch Werte für die Übereinstimmung auf der Ebene der Objekte berechnet werden. Dieser Prozeß wird als Voting bezeichnet (`select_object()`).

Voting

In der vorliegenden Implementierung werden fünf verschiedene Votings parallel verwendet. In der späteren Anwendung sollte man sich auf ein oder zwei Votings beschränken. Bei der Verwendung von mehr als einem Voting wäre die Behandlung von Widersprüchen in der Plazierung der Objekte zu definieren. Wir haben lediglich für die Tests mehrere Votings verwendet, um das günstigste Verfahren herauszufinden.
Zunächst werden die vom Matching erhaltenen Werte mit einem von der beteiligten Segment–Fläche abhängigen Gewicht multipliziert. Das Gewicht setzt sich aus einem unabhängigen Anteil und aus von der Größe und von der Verzerrung der Segment–Fläche abhängigen Anteilen zusammen. Die Matching–Werte werden umso schwächer gewichtet, je kleiner bzw. je verzerrter die beteiligte Segment–Fläche ist, da bei kleinen und bei stark verzerrten Segment–Flächen größere Fehler zu erwarten sind. Ein Schwellenwert legt die relative Größe einer Segment–Fläche fest, bei deren Unterschreitung das Gewicht für dieses Segment auf Null gesetzt und das Segment somit ausgeschlossen wird. Effizienter ist jedoch der Ausschluß vor dem Matching. Mit der Berücksichtigung der Größe der Segment–Fläche wird ein ähnlicher Effekt wie beim Ausschluß kleiner Segment–Flächen erzielt, jedoch kann auch der geringe Einfluß kleiner Flächen noch stören. Die genaue Berechnung des Gewichts ist in Abschnitt 9.3.5 beschrieben.
Die Votings 0 und 1 werden als „Maximum Votings“ bezeichnet. Dabei wird pro Segment das am besten passende Bild aus der Datenbank bestimmt. Dies geschieht durch eine Maximumsbildung, woher auch der Name der Voting–Verfahren herrührt.
Beim Voting 0 wird der gewichtete Matching–Wert zu den Werten aller zur

Datenbank–Fläche gehörenden Objekte[16] addiert und ein Zähler inkrementiert. Am Ende des Votings werden die Werte für jedes Objekt durch den Wert des Zählers geteilt.
Beim Voting 1 wird für jedes zur Datenbank–Fläche gehörende Objekt das Maximum aus dem bisherigen Wert und dem gewichteten Matching–Wert eingetragen. Somit erhält das bzw. die Objekt(e), zu dem bzw. denen das Datenbank–Bild mit dem besten Matching–Wert paßt, den höchsten Wert.
Die Votings 2, 3 und 4 sind sogenannte „Selection Votings". Für jedes Objekt werden dabei die n besten Paarungen von Segment– und Datenbank–Fläche bestimmt, wobei n das Minimum aus der Anzahl der in der Datenbank erfaßten Flächen des Objekts und der Anzahl der Segment–Flächen ist. Beim Voting 2 werden die gewichteten Matching–Werte aufaddiert, beim Voting 3 werden die Kehrwerte der „globalen Platzierungen" und beim Voting 4 die Kehrwerte der „Platzierungen im Segment" aufaddiert. Anschließend werden die Werte durch n geteilt. Als „Platzierung" wird die Anzahl größerer Matching–Werte +1 definiert. Bei der „Platzierung im Segment" werden dabei nur die Matching–Werte des betreffenden Segments betrachtet.

Speicherverwendung

Bei jedem Matching werden die Bilder der Datenbank neu geladen. Der Bedarf an virtuellem Speicher wäre (gerade im Hinblick auf die zu erwartende Größe der Datenbank im realen Einsatz) viel zu groß, um alle Bilder der Datenbank ständig im Speicher zu halten. Es wird jeweils ein Bild und das zugehörige Histogramm geladen und danach mit allen Segment–Flächen verglichen. Kann das Bild nicht geladen werden, so wird auf das Matching mit diesem Bild verzichtet, kann ein Histogramm nicht geladen werden, so wird es berechnet, allerdings an dieser Stelle grundsätzlich ohne Berücksichtigung eines Clip–Paths.
Bei den Histogrammen wäre es prinzipiell möglich, alle Histogramme der Datenbank im Speicher zu halten. Bei den momentan verwendeten Histogrammen mit 2048 Bins vom Typ `double` ergibt sich ein Speicherbedarf von 16KByte pro Histogramm. Das entspricht bei einer geschätzten Zahl von 10 000 Flächen in einer realen Datenbank 160MB. Durch Verwendung des Typs `float` und Reduzierung der Auflösung auf 1024 Bins verringert sich der Speicherbedarf auf 40MB, ein durchaus realisierbarer Wert.

[16] Normalerweise ist eine Datenbank–Fläche nur einem Objekt zugeordnet. Um aber gleichartige Flächen verschiedener Objekte in der Datenbank adäquat zu repräsentieren, wird dann nur eine Aufnahme gemacht, die mehreren Objekten zugeordnet wird.

9.3.3 Ausgaben des Programms

Das Programm gibt für jedes Objekt der Datenbank eine reelle Zahl an, die die Übereinstimmung des zu erkennenden Objekts mit dem jeweiligen Datenbankobjekt angibt. Je höher die Zahl ist, desto besser ist die Übereinstimmung. Die Zahl Null bedeutet, daß keinerlei Übereinstimmung gefunden wurde. Dies kommt in der Regel nur vor, wenn keine der Flächen des Objekts der Datenbank dem Matching unterzogen wurde, weil sie die Bedingungen zur Vorauswahl nicht erfüllten. Kleinere Zahlen als Null werden nicht vergeben.
Zusätzlich können Ausgaben zur Zeitmessung und zur Verfolgung des Programmablaufs (sogenannte Debug–Ausgaben) erzeugt werden. Ob solche Ausgaben erzeugt werden (und wenn ja welche), wird bei der Übersetzung des Programms mit Präprozessor–Konstanten gesteuert (siehe Abschnitt 9.3.5).

9.3.4 Fehler beim Programmablauf

Tritt ein Fehler beim Einlesen der Datenbank auf, so wird das Programm abgebrochen. Eine Ausnahme bilden lediglich unbedeutende Fehler, wie z.B., daß eine Datei nicht geschlossen werden kann. Dieses Verhalten wird im wesentlichen in der Datei `match_main.c` festgelegt. Die Funktionen zum Einlesen der Datenbank melden alle aufgetretenen Fehler im `return`–Wert und versuchen, soweit wie möglich trotz des Fehlers weiter zu arbeiten. So wird, wenn die Dateien zu einem Objekt nicht lesbar sind, dieses Objekt aus der eingelesenen Datenbank entfernt[17].
Fehler, die bei der Erkennung eines Objekts auftreten, führen nur zum Abbruch der Erkennung dieses Objekts. Das Programm fährt gegebenenfalls mit der Erkennung des nächsten Objekts fort. Lediglich schwerwiegende Fehler, wie Probleme beim Alloziieren von Speicher oder beim Lesen aus der Pipe, führen zu einem Programmabbruch. Auch hier wird das Verhalten im Fehlerfall durch die Anweisungen in `match_main.c` gesteuert.

Fehlerbehandlung im Programm

Es gilt der Grundsatz: „Fehler werden dort gemeldet, wo sie auftreten, aber an einer übergeordneten Stelle behandelt.“ Zur Meldung der Fehler dienen die Funktionen `warn()` und `error()` für Warnungen und Fehler. Sie werden numerisch codiert, die entsprechenden Konstanten sind in `errors.h` definiert. Die numerischen Codes für die Warnungen können gleichzeitig

[17] Genau genommen ist es dann ein Objekt, zu dem keine Bilder gespeichert sind und das somit auch nicht erkannt werden kann.

verwendet werden, um mehrere verschiedene Warnungen im `return`–Wert anzuzeigen. Von den Fehlern kann immer nur einer angezeigt werden und auch nicht gleichzeitig mit einer Warnung.

9.3.5 Übersetzung des Programms

Für die Programme `poly_match` und `mkhist` wird ein gemeinsames `Makefile` verwendet.
Das Verhalten des Programms wird über eine Reihe von Präprozessor–Konstanten gesteuert.

Steuerung des Matchings und Votings

Nahezu alle Steuerungsparameter des Matchings und Votings lassen sich über die Parameter–Datei (siehe Abschnitt 9.3.1) beeinflussen. Die unten genannten Dateien definieren nur die Vorgabewerte (Defaults).
Zur Steuerung des Matchings dienen die in `match_def.h` definierten Konstanten. Die Konstante `SEGMENT_FRACTION_LIMIT` wird benutzt, um kleinere Segmente vor dem Matching zu eliminieren. Segmente, deren Fläche mindestens um einen bestimmten Faktor kleiner ist als die größte Segment–Fläche, werden dann ausgeschlossen. (Mit einem Wert von 100 würden alle Segmente deren Fläche kleiner als ein Hundertstel der größten Fläche ist eliminiert.)
Ist die Konstante `USE_EA_RATIO_LIMIT` definiert, so wird das in Abschnitt 8.1.1 beschriebene Edge–Area–Ratio–Kriterium angewandt. Die Konstante `EA_RATIO_LIMIT` definiert dabei die maximal mögliche Abweichung. Ist der Quotient (bzw. sein Kehrwert, falls der Quotient größer als eins ist) der beiden Edge–Area–Ratios größer als der Wert von `EA_RATIO_LIMIT`, so wird die entsprechende Paarung Datenbankfläche/Segmentfläche nicht weiter betrachtet.
`COLOR_MODEL` definiert das für das Matching zu verwendende Farbmodell. Es kann zwischen RGB, HSI und dem bereits beschriebenen Gegenfarbenmodell gewählt werden.
`MHUE_MAX_EPS` und `AHUE_MAX_EPS` sind die maximalen Differenzen für die beiden Hue–Kriterien (siehe Abschnitt 8.1.1). Die Hue–Kriterien werden nur verwendet, wenn die entsprechende(n) `USE_...`–Konstante(n) definiert ist bzw. sind.
Folgende Definitionen werden von der vorliegenden Implementierung als Vorgabewerte verwendet:

```
/* match_def.h */
#define COLOR_MODEL COL_MODEL_RGBYWK
```

```
#define SEGMENT_FRACTION_LIMIT   100
#define EA_RATIO_LIMIT             5
#define MHUE_MAX_EPS    100
#define AHUE_MAX_EPS     50.0

#define USE_EA_RATIO_LIMIT
#define USE_MHUE_LIMIT
#define USE_AHUE_LIMIT
```

Die `USE_...`–Konstanten dienen nur der Optimierung. Wird der Quelltext ohne ihre Definition kompiliert, so werden die betreffenden Programmteile ausgeschlossen, was zu einem etwas schnelleren Programmablauf führt.
Zur Steuerung des Votings werden die Definitionen aus `vote_def.h` eingesetzt. Es werden beliebig viele Voting–Varianten unterstützt. Die verwendete Anzahl ist in `vote_def.h` als `NUM_VOTINGS` definiert.
Den Schwerpunkt der Definitionen bilden die Konstanten zur Gewichtung der beim Matching ermittelten Werte. Nach folgendem Verfahren wird ein Gewicht bestimmt:

```
rel_area = seg_area / max_area;
weight   = WEIGHT_CONST;
weight  += WEIGHT_AREA * WEIGHT_MULTIPLIER_AREA * rel_area;
weight  += WEIGHT_PQ   * WEIGHT_MULTIPLIER_PQ
   * 1.0 / sqrt( p * p  +  q * q  +  1.0 );
weight  /= WEIGHT_ALL;
```

Die vom Matching erhaltenen Werte werden dann vor der Weiterverarbeitung mit dem Gewicht des beteiligten Segments multipliziert.
Der Anteil `WEIGHT_CONST` ist unabhängig von weiteren Parametern. `WEIGHT_AREA` und `WEIGHT_PQ` gewichten jeweils den Anteil der relativen Flächengröße und der Neigung der Segment–Fläche.
Die abschließende Division durch `WEIGHT_ALL` dient der Normierung, so daß nur die Verhältnisse zwischen den Gewichtungsfaktoren von Bedeutung sind.
Die `WEIGHT_MULTIPLIER`–Konstanten dienen dazu, den Einfluß der entsprechenden Größe überproportional zu wählen und unabhängig von der Normierung mittels `WEIGHT_ALL` zu steuern.
Die Parameter `WEIGHT_HISTO` und `WEIGHT_TEMPL` dienen zur Steuerung des Anteils des Histogramm– und Template–Matchings am Gesamtergebnis. Die Matching–Werte der beiden Verfahren werden jeweils mit ihrem zugehörigen Gewicht multipliziert und anschließend addiert.
Die Konstante `AREA_THRESHOLD` dient zum Ausschluß sehr kleiner Segment–Flächen. Für Segment–Flächen, deren Quotient aus der Fläche der größten

Segment–Fläche und ihrer Fläche größer als der Wert von AREA_THRESHOLD ist, wird das Gewicht auf Null gesetzt. Dies entspricht einem Ausschluß der betreffenden Segment–Flächen. Da dasselbe Ergebnis auch mit Hilfe des Parameters SEGMENT_FRACTION_LIMIT wesentlich effizienter erreicht werden kann, wird empfohlen, AREA_THRESHOLD nicht mehr zu verwenden. Der Parameter AREA_THRESHOLD war in einer früheren Version, die kein SEGMENT_FRACTION_LIMIT hatte, benutzt worden. Bei der Verwendung von SEGMENT_FRACTION_LIMIT entfallen die Matching–Versuche für die ausgeschlossenen Segmente.
Folgende Definitionen werden von der vorliegenden Implementierung als Vorgabewerte verwendet:

```
/* vote_def.h */
#define NUM_VOTINGS 5

#define WEIGHT_CONST     1.0
#define WEIGHT_AREA      0.5
#define WEIGHT_PQ        0.5
#define WEIGHT_ALL       WEIGHT_CONST + WEIGHT_AREA + WEIGHT_PQ
#define WEIGHT_MULTIPLIER_AREA      1.0
#define WEIGHT_MULTIPLIER_PQ        1.0

#define AREA_THRESHOLD   9.5

#define WEIGHT_HISTO     1.0
#define WEIGHT_TEMPL     0.0
```

Ausgaben für Messungen und zum Debugging

Die Ausgaben zum Debugging und zur Zeitmessung werden mit folgenden Präprozessorkonstanten gesteuert:

Konstante	Ausgabe von ...
DB_DEBUG	Meldungen zum Einlesen der Datenbank
TOPVW_DEBUG	Meldungen zur Entzerrung
MATCH_DEBUG	Meldungen zum Matching
PROFILING	Ergebnissen der Zeit– und Performance–Messungen
PROFILING_DETAIL	wie oben, jedoch mit einer detailierteren Ausgabe

Ist die entsprechende Konstante definiert, so erfolgen die Ausgaben, ansonsten nicht.
Die PROFILING–Ausgaben sind die von der Funktion getrusage() gemeldeten Werte. Ist nur PROFILING definiert, so werden nur die Differenzen

der Profiling–Werte für bestimmte Programmteile (Einlesen der Datenbank, komplette Erkennung eines Objekts) ausgegeben. Bei `PROFILING_DETAIL` werden zusätzlich auch noch die absoluten Werte ausgegeben.

Besonderheit bei SunOS 5.x

Aufgrund der Verwendung der Funktion `getrusage()` zur Zeit– und Performance–Messung sind unter SunOS 5.x einige Vorkehrungen zu treffen, da es sich um eine BSD–Funktion handelt. Details dazu finden sich im `Makefile`. Wird auf die Messungen verzichtet (`PROFILING` nicht definiert), so kann auch unter SunOS 5.x ohne besondere Vorkehrungen übersetzt werden.

10 Ergebnisse und Fazit

Wie in Abschnitt 4 beschrieben, wurden zwei Aufnahmeserien durchgeführt, jeweils unter verschiedenen Bedingungen mit unterschiedlicher Vorverarbeitung. Daher wird die Auswertung der Ergebnisse beider Serien getrennt behandelt, wobei der Schwerpunkt auf der zweiten Serie liegt, die mehr Objekte umfaßt und bei der Aufnahmen, Vorverarbeitung und Matching mit erhöhtem Aufwand durchgeführt wurden.

10.1 Ergebnisse der ersten Aufnahmeserie

Mittels der Objekterkennung anhand der farbigen Aufdrucke konnten in der ersten Serie neun von 13 Objekten richtig zugeordnet werden, vgl. Tab. 10 (e) im Anhang. Dieses Ergebnis kann als zufriedenstellend gewertet werden, da erhebliche Helligkeits- und Farbschwankungen zwischen den Aufnahmen der Datenbankbilder und den Testaufnahmen festzustellen sind und bei der Segmentierung einige Flächen nur stark approximiert erkannt wurden (vgl. Abschnitt 7).
Die Abweichungen konnten durch die Verringerung der Farbauflösung und durch die Einführung von Parametern in den Matchingalgorithmus entscheidend kompensiert werden.
Die vier nicht erkannten Objekte unterscheiden sich in Ihrer Farbzusammensetzung nur gering von entsprechend anderen, als falsch erkannten Objekten. So bestehen einige Objekte (z.B. `Sahne_Reichelt_0.5`) aus überwiegend weißen Flächen mit roten Aufdrucken. Die Fehler in der Segmentierung und die durch die Ausleuchtung verursachten Farb- und Helligkeitsschwankungen führen in diesem Fall zu den wenigen Fehlzurodnungen. (Siehe dazu Testreihen in den Tabellen 10 (a)–(e) und 10 (u)–(x).) Mit einer homogenen Ausleuchtung und einer genauen Segmentierung könnten auch diese Objekte richtig erkannt werden. Zieht man zusätzlich topologische Merkmale zum Matching heran, so könnten diese Fehlzuordungen ebenfalls vermieden werden.
Das Objekt `Pril` wird richtig erkannt, allerdings mit nur geringem Abstand zum Objekt `Johnson_Wachs`. Alle anderen Objekte werden eindeutig erkannt, mit deutlichen Abständen zu den folgenden Objekten. Die genauen Plazierungen sind den Testergebnissen im Anhang zu entnehmen.

Abhängigkeit von der Farbauflösung

Bei verschiedenen Aufnahmen eines Objektes können infolge von veränderten Lichtverhältnissen und Positionen der Lichtquellen Differenzen in den

Farb- und Helligkeitswerten auftreten. Diese Differenzen sind trotz der im Abschnitt 4.3 beschriebenen Farbkorrektur bei unseren Aufnahmereihen deutlich, da die Aufnahmen für die Datenbank und die Aufnahmen für das Matching zu verschiedenen Zeitpunkten erstellt wurden. Aufgrund dessen werden die besten Ergebnisse durch eine recht niedrige Auflösung der Histogramme erzielt, mit jeweils 3 Bits für den Rot–Grün– und Blau–Gelb–Anteil und mit nur 2 Bits für den Weiß–Schwarz–Anteil, da die Abweichungen zum Datenbankbild hier am größten sind. Diese Histogramme mit einer Größe von 256 Bins kompensieren die Differenzen weitgehend. Durch die Verwendung von Histogrammen mit einer Größe von 2048 Bins (Auflösung: 4,4,3) erreichen wir ebenfalls noch akzeptable Ergebnisse. Besser aufgelöste Histogramme sollten nur dann Verwendung finden, wenn die Objekte unter exakt den gleichen Bedingungen ohne nennenswerte Farb- und Helligkeitdifferenzen aufgenommen wurden. (Siehe dazu die Testreihen in den Tabellen 10 (a)–(e) und 10 (u)–(x).)

Ergebnisse durch Veränderung der Matching-Parameter

Mit Hilfe des Paramters `SEGMENT_FRACTION_LIMIT` können kleinere Flächen vom Matching ausgeschlossen werden. Das ist bei solchen Objekten sinnvoll, die kleine Flächen im Bereich des Bodens oder des Giebels besitzen. Diese Flächen weisen kaum markante Farbaufdrucke auf und sind in vielen Fällen einfarbig, so daß diese Flächen wegen der Helligkeitsabweichung, die daraus resultiert, daß aufgrund der einseitigen Beleuchtung bei den Aufnahmen je nach Lage des Objektes eine Fläche mehr oder weniger stark belichtet wird, falsch zugeordnet werden und somit die richtige Zuordnung der großen markanten Flächen teilweise wieder aufheben. Für die erste Testreihe wurde der von der Wirkung her gleichwertige Parameter `AREA_THRESHOLD` verwendet. Der Wert 1.5 führt hierbei zu den besten Testergebnissen (vgl. Testreihen in der Tabelle 10 (c), (e), (f) und (g)). Bei diesem Wert werden alle Flächen, die kleiner 66,7% der größten Fläche dieses Objektes sind, beim Matching nicht berücksichtigt.
Die mehr oder weniger starke Bevorteilung von großen Flächen gegenüber kleinen durch die Paramter `WEIGHT_AREA` und `WEIGHT_MULTIPLIER_AREA` (siehe Abschnitt 9.3.5) wirkt sich in den Testergebnissen nur gering aus (vgl. Testreihen in den Tabellen 10 (h), (i), (j), (n) und (o)). Gleiches gilt für die Parameter `WEIGHT_PQ` und `WEIGHT_MULTIPLIER_PQ`, wobei diese Bevorteilung von unverzerrten Flächen auf der Grundlage der unbrauchbaren weil fehlerhaften pq–Werte beruht.

10.2 Ergebnisse der zweiten Aufnahmeserie

10.2.1 Überblick

Die Aufnahmeserie umfaßt insgesamt 50 Aufnahmen von 29 verschiedenen Objekten. Dabei wurden die Objekte sehr vielfältig ausgewählt: Es sind neben annähernd polyedrischen Objekten sowohl mehrere runde Objekte als auch verschutzte und verformte Objekte vorhanden. Damit ist es möglich, infolge umfangreicher Tests sichere Aussagen darüber zu finden, welche Objektgruppen erkannt werden und welche nicht. Da die Qualität der pq–Werte gegenüber der ersten Serie etwas verbessert wurde, fand neben dem Histrogramm–Vergleich auch das Template–Matching Berücksichtigung.

10.2.2 Gesamtergebnis

Polyedrische Objekte

Bei polyedrischen Objekten erhalten wir bis auf wenige Ausnahmen gute Ergebnisse. Bei diesen Objekten ist es größtenteils möglich, mittels des uns zur Verfügung gestellten Segmentierers die Objektflächen zufriedenstellend zu rekonstruieren. Von den insgesamt 23 annähernd polyedrischen Objekten werden 17 Objekte korrekt erkannt.

Farbmodell	Gegenfarben	RGB	HSI
exakt erkannt	72%	69%	62%
unter den besten 3	93%	93%	86%
nicht erkannt	7%	7%	14%

Tabelle 5: Ergebnisse mit polyedrischen Objekten

Dabei werden solche Objekte ausgezeichet erkannt, bei denen große und detailreiche Flächen extrahiert werden können. Ein Beispiel dafür ist das Objekt `Milram_Milch_1.0`. Probleme treten auf, wenn die Objektflächen falsch segmentiert wurden. Schwierig zu erkennen sind außerdem Aufnahmen, bei denen aufgrund der Lage des Objektes zur Kamera eine weiße Bodenfläche gut erkannt wird aber wichtige Aufdrucke auf den Seitenflächen durch zu starke Neigung zu dunkel und damit kaum erkennbar sind.

Runde Objekte

Bei runden Objekten — ein Beispiel ist der Becher `Brandenburg_Joghurt_500` — müssen gekrümmte Flächen in planare aufgeteilt werden. Der uns zur Verfügung gestellte Segmentierer ist dazu nicht geeignet. Die Ansicht einer gekrümmten Fläche kann auch nicht als eine einzige planare Fläche dargestellt werden; stattdessen werden die runden Flächen willkürlich in beliebige Vielecke mit einer hohen Anzahl an Ecken unterteilt. Somit ist ein Template–Matching nicht möglich. Verwendet man das Histogramm–Matching, so hängt es vom Zufall ab, ob die vom Segmentierer gefundenen Flächen in etwa die Farbzusammensetzung haben, wie die Datenbankflächen. Der Becher `Brandenburg_Joghurt_Erdbeer` wird im Gegenfarbenmodell erkannt. (vgl. Tabelle 11 (l)).

Farbmodell	Gegenfarben	RGB	HSI
exakt erkannt	14%	14%	0%
unter den besten 3	29%	29%	14%
nicht erkannt	71%	71%	86%

Tabelle 6: Ergebnisse mit runden Objekten

Um Verbesserungen zu erreichen, ist es nötig, einen Algorithmus für die Segmentierung von gekrümmten Objekten zu entwickeln. Dennoch bleibt das Problem ungelöst, immer einen gemeimsamen Startpunkt auf der (endlosen) gekrümmten Fläche zu finden.

Verschmutzte und verformte Objekte

In der Aufnahmeserie sind drei Objekte enthalten, die vor der Aufnahme mit Buntstiften verschmutzt wurden. Wie aus der Tabelle 11 (l) ersichtlich, werden zwei von drei verschmutzten Objekten richtig erkannt. Beim Objekt `Milsani_Kondens_0.2` (Aufnahme Nr. 2) sind nur ca. 10% jeder Fläche mit schwarzem Stift bemalt. Somit stehen 90% der Hauptfläche zur Verfügung, und das Objekt wird richtig erkannt. Dieser Grad der Verschmutzung bewegt sich innerhalb der Fehlertoleranz der Matchingverfahren.

Farbmodell	Gegenfarben	RGB	HSI
exakt erkannt	71%	29%	43%
unter den besten 3	86%	43%	57%
nicht erkannt	14%	57%	43%

Tabelle 7: Ergebnisse mit verschmutzten und verformten Objekten

Das Verfahren ist gegen kleine Verunreinigungen resistent. Nach eigenen Erfahrungen beschränken sich die Verunreinigugen in der Praxis auf Flüssigkeiten, Fette und kleinere Inhaltsreste, die die Oberflächen kaum verändern.

In der Praxis problematischer sind dagegen Verformungen, da in bestimmten Haushalten aus Platzgründen die Objekte zusammengedrückt werden. Das Tetrapak `Milsani_Milch_1.0` (Aufnahme Nr. 25) zeigt, daß zusammengedrückte Objekte mit Hilfe des Histogramm-Matchings exakt erkannt werden können, solange mindestens eine große Fläche einigermaßen in ihrer ursprünglichen Form erhalten bleibt (siehe Tabelle 11 (a)). Gleiches gilt für die Aufnahme Nr. 37 (Objekt `Toblerone_400`). Auch kleinere Verformungen bzw. wellige Oberflächen bei den Aufnahmen 2, 3, 7 und 26 werden vom Verfahren kompensiert.

10.2.3 Parameterauswahl

Auswahl des Farbmodells

In unserem Verfahren werden drei verschiedene Farbmodelle verwendet. Die Testreihen in Tabelle 11 (a)–(c) wurden im RGB–Farbraum durchgeführt. Die Helligkeit der Objektflächen variiert aufgrund der verschiedenen Betrachtungswinkel. Je größer die Winkel zwischen Objektfläche und Kamera sind, desto dunkler erscheinen die Flächen, da gerichtete Lichtquellen verwendet werden müssen. Die Datenbankflächen hingegen wurden immer mit maximaler Lichtstärke ausgeleuchet. Das Gegenfarbenmodell erlaubt die Extraktion des Weiß–Schwarz–Anteils und somit eine Benachteiligung der Intensität.
Insgesamt verbessert das Gegenfarbenmodell die Ergebnisse nur geringfügig (vgl. Testreihen in Tabelle 11 (a) und (o)). So wird das Objekt `Butter` (Aufnahme 49), welches sehr dunkle Flächen besitzt, im RGB–Modell nur auf Platz drei erkannt, im Gegenfarbenmodell plaziert sich dieses Objekt jedoch an erster Stelle.
Das dritte Farbmodell HSI besitzt kaum Vorteile im Vergleich zu den anderen beiden Modellen (vgl. Tabelle 11 (a), (e) und (o)). Das Objekt `Milram_Quark` (Aufnahme 3) wird zwar nur in diesem Model exakt erkannt, bei einigen anderen Objekten verschlechtern sich jedoch die Ergebnisse etwas. Hue–Wert (Buntton) und Sättigung enthalten hier die meisten Informationen, da auch in diesem Modell die Intensität unterdrückt wird. Der Hue–Wert reagiert sehr empfindlich auf leichte Farbschwankungen bei hellen Objekten. Die weißen Flächen unserer Objekte bekommen je nach Lichtquelle und deren Positionen leichte Farbstiche. So ermittelten wir zwischen einer weißen Fläche mit leichtem Rot-Stich und der gleichen weißen Fläche (andere Aufnahme) mit leichtem Blau-Stich eine hohe Differrenz im Hue–Wert (Differenz: 100 von maximal 255). Somit enthält auch der Hue–Wert bei einigen Objekten keine verwendbaren Informationen, woraus die etwas

schlechteren Ergebnisse im HSI–Modell resultieren.
In den Tabellen 5, 6 und 7 werden die Ergebnisse verglichen.

Ausschluß kleiner Flächen

Mit Hilfe des `SEGMENT_FRACTION_LIMIT`s werden kleine Flächen vom Matching ausgeschlossen, die vom Segmentierer überhaupt nicht bzw. nur unzureichend extrahiert werden, und damit das Ergebnis negativ beeinflussen würden. Dadurch wird auch die Rechenzeit verringert, da weniger Vergleiche mit den Objektflächen nötig sind.
Mit Werten zwischen 1.5 und 3.0 für diesen Parameter werden die besten Ergebnisse erreicht. Das Limit 3.0 bedeutet, daß diejenigen Flächen eines Objektes zum Matching verwendet werden, die nicht kleiner als ein drittel der größten Fläche des Objektes sind (reziproker Wert des Limits).

Ausschluß von Flächen unterschiedlicher Form

Als einziges geometrisches Merkmal wird das EA_RATIO als Verhältnis von längster Kante zur Wurzel des Flächeninhalts zum Matching verwendet. Ein Quadrat besitzt ein EA_RATIO von 1.0, ein Rechteck mit einem Kantenlängenverhältnis von 4 zu 1 ein EA_RATIO von 2.0. Vor dem Vergleich zweier Flächen wird der Quotient aus den beiden EA_RATIOs bestimmt und mit dem `EA_RATIO_LIMIT` verglichen. Übertrifft dieser Quotient (bzw. sein reziproker Wert, falls der Quotient kleiner als eins ist) das Limit, so sind beide Flächen entsprechend dieses Merkmals verschieden und ein Histogramm– bzw. Template–Matching ist nicht notwendig. Aufgrund der auch in der zweiten Serie schwankenden Qualität der berechneten Oberflächennormalen muß dieses Limit innerhalb unserer Testreihen auf den sehr hohen Wert 2.5 gesetzt werden, um in jedem Fall nur ungleiche Flächen nicht weiter zu betrachten.

Template–Matching

In der zweiten Aufnahmeserie konnten die pq–Werte der Objekte verbessert werden, so daß das Template–Matching in die Testreihen mit einbezogen wurde. Die Auswertung der Testreihen ergab, daß die alleinige Anwendung des Template–Matchings für die Objekte mit dem "Grünen Punkt" nicht sinnvoll ist. Die Objekte besitzen runde Kanten, leichte Vorformungen etc. Eine exakte Entzerrung und Überlagerung der zu vergleichenden Flächen ist somit nicht möglich. Die verbesserten, aber in der Qualität noch schwankenden pq–Werte und der verwendete Segmentierer führen zu nicht korrekten

Überlagerungen. So treten bei theoretisch deckungsgleichen Flächen Verschiebungen von bis zu 30 Pixel auf. Bei großen Schriften, z.B. das Wort "Vollmilch" beim Objekt `Milsani_Milch_1.0`, liegen die Buchstaben dadurch genau nebeneinander.

Matching-Verfahren	Histogramm	Template
exakt erkannt	72%	14%
unter den besten 3	93%	21%
nicht erkannt	7%	79%

Tabelle 8: Vergleich zwischen Histogramm– und Template–Matching

Die Testreihe in Tabelle 11 (f) zeigt ein reines Template–Matching. Das Objekt `Erlenhof_Milch_1.0` liefert ein Beispiel dafür, daß man bei ausreichend genauer Überlagerung gute Ergebnisse erhält.
Andererseits wirkt sich das Ergebnis des Template–Matchings in Kombination mit einem anderen Verfahren nicht negativ aus. So tritt der Fall nicht auf, daß eine Fläche eines anderen Objekts erheblich besser erkannt wird, als die Fläche des richtigen Objekts. Somit ist das Template–Matching hilfreich in Kombination mit dem Histogramm–Matching, da es das Ergebnis des Histogramm–Matching bei guter Überlagerung verbessert, bei ungenauer Überlagerung aber nicht verschlechtert.

Histogrammauflösung

In der zweiten Aufnahmeserie liefert eine Histogramm–Auflösung von jeweils 4 Bit für den Rot–Grün– bzw. Blau–Gelb–Anteil sowie 3 Bit für den Schwarz–Weiß–Anteil im Gegenfarbenmodell die besten Ergebnisse. Der Schwarz–Weiß–Anteil ist etwas gröber aufgelöst, um kleine Helligkeitunterschiede beim Matching zu ignorieren. Die Auflösung konnte somit gegenüber der ersten Testserie auf insgesamt 11 Bit gesteigert werden, da die bessere Qualität der Objektaufnahmen jetzt eine genauere Differenzierung der Farbwerte zuläßt.

Histogrammauflösung	4-4-3	3-3-2
exakt erkannt	72%	66%
unter den besten 3	93%	86%
nicht erkannt	7%	14%

Tabelle 9: Histogramm–Matching im Gegenfarbenmodell

10.3 Geschwindigkeit

Die Erkennung eines Objekts beansprucht auf der Grundlage der ersten Aufnahmeserie mit 16 Datenbankobjekten 17 Sekunden Rechenzeit auf einer SUN SPARCStation 10 unter SunOS 5.4. Für die zweite Aufnahmeserie mit 29 Datenbankobjekten dautert dieser Vorgang knapp 25 Sekunden. Diese Zeiten gelten für das Histogramm–Matching und können je nach Flächenanzahl und -größe des zu erkennenden Objektes bis zu 50% nach oben und unten abweichen. Werden Template-Matching oder gar beide Verfahren parallel verwendet, so erhöht sich die Laufzeit um wenige Sekunden. Wird das Programm auf leistungsschwächeren SUN SPARCStations ausgeführt (z.B. IPC), so vergrößert sich die Rechenzeit höchstens um den Faktor 2.
Die Verwendung verschiedener Histogrammauflösungen hat bis zu einer Größe von 14 Bits (5 Bits Rot–Grün, 5 Bits Blau–Gelb, 4 Bits Schwarz–Weiß) keinen meßbaren Einfluß auf die Rechenzeit.

Ansätze zur Geschwindigkeitssteigerung

Wie bereits in Abschnitt 9.3.2 erwähnt, werden die Bilder der Datenbank nicht ständig im Speicher gehalten, sondern bei jedem Matching neu geladen. An dieser Stelle wäre natürlich am meisten Zeit zu sparen, indem die Bilder geladen bleiben. Das ist aber in der Regel ein Speicherproblem. Allerdings läßt sich unser Verfahren noch etwas verbessern, indem man nicht jedesmal die gesamte Bilddatei inklusive Header liest, sondern nur die Bilddaten selbst. Die im Header enthaltenen Informationen wie Bildgröße etc. werden sowieso ständig im Speicher gehalten. Dazu wäre die Funktion `ppm_read_file()` in der Datei `rw_ppm.c` anzupassen.

10.4 Fazit

Mit hauptsächlich oder vollkommen polyedrischen Objekten werden gute bis sehr gute Ergebnisse erreicht. Dabei erweist sich das Histogramm–Matching als ein einfaches und schnelles Verfahren zum Vergleich von Objektflächen, welches sehr robust bzgl. Verzerrungen, ungenauer Erkennung der Flächengrenzen sowie leichten Verformungen und Verunreinigungen ist. Dieses Verfahren versagt genau dann, wenn Flächen verschiedener Objekte eine ähnliche Farbverteilung aufweisen.
Das Template–Matching als zweite Methode erfordert eine exakte Erkennung der Flächengrenzen, die der uns zur Verfügung gestellte Segmentierer nur teilweise beherrschte. So bietet sich dieses Verfahren als Ergänzung

zum Histogramm–Matching, um in den Fällen zu entscheiden, bei denen das Histogramm–Matching kein eindeutiges Ergebnis liefert.
Aufgrund der ungenauen Flächenextraktion und der nicht exakten Bestimmung der Oberflächennormalen haben die geometischen Mermale zur Vorauswahl bisher nur eine geringe Bedeutung, so daß eine Vielzahl von Histogramm–Vergleichen notwendig ist und damit der Rechenaufwand beträchtlich wird.
Daher gibt es bezüglich des in der Einführung beschrieben praktischen Einsatzes an zwei Stellen Ansätze zur Verbesserung:
Zum einen ist ein schneller Algorithmus zur Segmentierung zu entwickeln, der speziell für diese Objekte angepaßt ist, um die Erkennung von gekrümmten Flächen zu ermöglichen und um durch eine verläßliche Vorauswahl durch geometrische Merkmale die Anzahl der Vergleiche zu minimieren und damit die Rechenzeit zu senken.
Zum anderen ist ein genaues Verfahren zur Bestimmung der Oberflächennormalen notwendig, um die Objektflächen vollkommen entzerren zu können, so daß die geometrischen Merkmale und der pixelweise Vergleich zuverlässiger sind.

A Testergebnisse

	a	b	c	d	e	f	g	h	i	j	k	l	m
Histogrammauflösung	443	333	332	554	332	332	332	332	332	332	332	332	332
WEIGHT_CONST	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WEIGHT_AREA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	2.0	0.0	0.0	0.0
WEIGHT_PQ	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	2.0
WEIGHT_MULTIPLIER_AREA	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
WEIGHT_MULTIPLIER_PQ	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
AREA_THRESHOLD	2.0	2.0	1.0	2.0	1.5	3.0	6.0	2.0	2.0	2.0	2.0	2.0	2.0

	n	o	p	q	r	s	t	u	v	w	x	y
Histogrammauflösung	332	332	332	332	332	332	332	322	232	222	221	440
WEIGHT_CONST	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0
WEIGHT_AREA	0.5	0.5	0.0	0.0	0.5	0.5	0.5	0.0	0.0	0.0	0.0	0.0
WEIGHT_PQ	0.0	0.5	0.5	0.5	0.0	0.5	0.25	0.0	0.0	0.0	0.0	0.0
WEIGHT_MULTIPLIER_AREA	2.0	5.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WEIGHT_MULTIPLIER_PQ	1.0	1.0	2.0	5.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
AREA_THRESHOLD	2.0	2.0	2.0	2.0	1.0	1.5	1.5	1.5	1.5	1.5	1.5	1.5

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
Milch_H_r_1.0	2	3	3	2	3	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Milch_H_b_1.0	6	7	7	6	9	7	7	8	8	8	8	8	8	8	8	8	8	9	9	9	8	7	7	8	9
Milch_Brandbg_1.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Milch_Reichelt_1.0	9	9	10	9	7	7	7	5	5	5	7	7	7	5	5	7	7	7	7	7	5	7	6	7	7
Milch_Reichelt_1.0	12	14	14	10	5	5	5	5	5	5	5	5	5	5	5	5	5	3	5	5	7	10	11	11	4
Sahne_Reichelt_0.5	3	2	2	2	2	2	2	2	2	2	1	1	1	2	2	1	1	2	2	2	1	2	2	2	2
Pril	4	3	1	3	1	1	1	2	2	2	1	1	1	2	2	1	2	1	2	2	1	2	1	2	4
Pril	4	5	3	5	3	3	5	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	4	3	3
Traubensaft_1.0	2	3	3	2	3	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	5	2
Traubensaft_1.0	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
Quarkbecher	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	1
Quarkbecher	2	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1
Milch_Uelzena_0.5	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Milch_Uelzena_0.5	3	3	2	6	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	2
Milch_Uelzena_0.5	3	3	2	3	1	2	2	2	2	3	3	3	3	2	3	3	3	1	1	1	2	1	3	1	1
Valensina_0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Valensina_0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Johnson_Wachs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1
Johnson_Wachs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Odol	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1
Odol	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Toblerone	2	1	1	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Joghurt	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabelle 10: Testreihe 1

Varibalenname	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
COLOR_MODEL	RGB	RGB	RGB	HSI	HSI	HSI	HSI	HSI	HSI	HSI	GFM	GFM	GFM	GFM	GFM	GFM
SEGMENT_FRACTION_LIMIT	1.5	3.0	3.0	1.5	1.5	1.5	3.0	3.0	3.0	3.0	1.5	1.5	1.5	3.0	3.0	1.5
EA_RATIO_LIMIT	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
WEIGHT_AREA	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0
WEIGHT_PQ	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0
WEIGHT_MULTIPLIER_AREA	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
WEIGHT_MULTIPLIER_PQ	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
WEIGHT_HISTO	0.7	0.7	1.0	0.7	1.0	0.0	1.0	0.7	0.7	0.7	0.7	1.0	0.7	0.7	0.7	1.0
WEIGHT_TEMPL	0.3	0.3	0.0	0.3	0.0	1.0	0.0	0.3	0.3	0.3	0.3	0.0	0.3	0.3	0.3	0.0
HISTO_BITS_RG	3	3	3	4	4	4	3	4	4	3	3	4	3	4	4	4
HISTO_BITS_BY	3	3	3	4	4	4	3	4	4	3	3	4	3	4	4	4
HISTO_BITS_WK	2	2	2	3	3	3	2	3	3	2	2	3	2	3	3	3
HISTO_BITS_H	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
HISTO_BITS_S	4	4	4	3	3	3	4	3	3	4	4	3	4	3	3	3
HISTO_BITS_I	2	2	2	1	1	1	2	1	1	2	2	1	2	1	1	1
HISTO_BITS_R	3	3	3	4	4	4	3	4	4	3	3	4	3	4	4	4
HISTO_BITS_G	3	3	3	4	4	4	3	4	4	3	3	4	3	4	4	4
HISTO_BITS_B	3	3	3	4	4	4	3	4	4	3	3	4	3	4	4	4
TEMPL_WEIGHT_RG	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
TEMPL_WEIGHT_BY	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
TEMPL_WEIGHT_WK	1.0	1.0	1.0	3.0	3.0	3.0	1.0	3.0	3.0	1.0	1.0	3.0	1.0	3.0	3.0	3.0
TEMPL_WEIGHT_H	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
TEMPL_WEIGHT_S	4.0	4.0	4.0	3.0	3.0	3.0	4.0	3.0	3.0	4.0	4.0	3.0	4.0	3.0	3.0	3.0
TEMPL_WEIGHT_I	2.0	2.0	2.0	1.0	1.0	1.0	2.0	1.0	1.0	2.0	2.0	1.0	2.0	1.0	1.0	1.0
TEMPL_WEIGHT_R	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TEMPL_WEIGHT_G	2.0	2.0	2.0	1.0	1.0	1.0	2.0	1.0	1.0	2.0	2.0	1.0	2.0	1.0	1.0	1.0
TEMPL_WEIGHT_B	3.0	3.0	3.0	1.0	1.0	1.0	3.0	1.0	1.0	3.0	3.0	1.0	3.0	1.0	1.0	1.0

Tabelle 11: Testreihe 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
Milsani_Kondens_0.2	2	3	7	2	2	9	8	5	4	3	5	2	5	6	4	10
Milram_Quark	3	2	2	1	1	10	8	3	1	1	3	3	3	4	2	2
Valensina_Orange_0.2	1	1	2	2	2	7	2	2	2	2	1	2	1	2	2	1
Odol	2	3	1	1	1	-	3	2	2	2	3	1	7	3	2	9
Reichelt_Sahne_0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1
Reichelt_Sahne_0.5	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1
Uelzena_Milch_0.5	1	1	1	1	1	7	1	1	1	1	1	1	1	1	1	1
Pril	1	1	1	1	1	-	1	1	1	1	1	1	1	1	1	1
Valensina_Multi_0.75	1	1	1	3	3	14	3	3	3	3	1	1	3	1	2	3
Valensina_Multi_0.75	1	1	1	3	2	11	2	3	3	3	1	1	1	1	1	1
Valensina_Orange_0.75	1	1	1	1	1	9	1	1	1	1	1	1	1	2	2	1
Moevenpick_Himbeer	2	2	4	3	4	3	2	3	3	3	2	3	1	2	2	1
Moevenpick_Himbeer	1	1	1	2	1	6	1	2	1	1	1	2	2	1	1	2
Milsani_Milch_1.0	1	1	1	1	1	8	1	1	1	1	1	1	1	1	1	1
Milsani_Milch_1.0	-	3	14	-	-	-	12	7	8	8	-	-	-	4	6	10
Kirberg_Traubensaft_1.0	2	4	11	1	1	9	3	4	3	1	1	1	1	1	1	1
Kirberg_Traubensaft_1.0	1	1	1	1	1	-	1	1	1	1	1	1	1	1	1	1
Pfanner_EisTee_Ananas_1.0	1	1	1	1	1	6	1	1	1	1	1	1	1	1	1	1
Erlenhof_Milch_1.0	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1
Emzett_Milch_1.0	1	1	1	1	1	5	1	1	1	1	1	1	1	1	1	1
Reichelt_Milch_1.0	13	12	10	3	3	6	16	5	5	12	10	9	9	9	10	12
Reichelt_Milch_1.0	1	1	1	11	9	-	12	11	11	12	3	1	3	2	3	3
Toblerone_400	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Brandenburg_Milch_2.0	1	1	1	1	1	9	1	1	1	1	1	1	1	1	1	1
Brandenburg_Milch_2.0	1	1	1	1	1	13	1	1	1	1	1	1	1	1	1	1
DieWeissen_Milch_1.0	6	5	6	4	5	14	7	4	4	6	7	8	7	5	5	6
DieWeissen_Milch_1.0	1	1	1	1	1	8	3	1	1	4	2	1	2	1	1	2
Milram_Milch_1.0	1	1	1	2	2	9	2	2	2	2	1	1	1	1	1	1
Butter	3	7	2	6	6	5	4	7	7	2	3	1	3	3	1	5
Butter	3	3	3	5	5	1	2	5	5	1	3	3	3	2	2	3

Tabelle 12: Polyedrische Objekte

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
Milsani_Kondens_0.2	2	5	5	1	1	7	5	3	3	3	1	2	1	7	4	1
Milsani_Kondens_0.2	9	11	2	4	1	9	1	4	4	11	6	1	6	8	8	8
Brandenburg_Joghurt_Erdbeer	1	1	1	5	7	-	3	5	7	7	3	1	6	2	1	6
Milsani_Milch_1.0	1	1	1	1	1	6	1	1	1	1	1	1	1	1	1	1
Milsani_Milch_1.0	-	9	8	-	-	-	3	10	15	15	-	-	-	12	15	16
Toblerone_400	1	1	1	1	4	1	4	1	1	1	1	1	1	1	1	1
Brandenburg_Milch_2.0	4	4	1	2	2	17	2	2	2	2	3	1	3	2	2	3

Tabelle 13: Verschmutzte und verformte Objekte

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
Brandenburg_Joghurt_Erdbeer	5	5	5	12	10	-	9	10	8	9	11	10	10	10	9	8
Brandenburg_Joghurt_Erdbeer	1	1	1	5	7	-	3	5	7	7	3	1	6	2	1	6
Mueller_Milch_Schoko	-	-			-	-	-	-	-	-	-	-	-	-	-	-
Mueller_Milch_Schoko	-	-			-	-	-	-	-	-	-	-	-	-	-	-
Brandenburg_Joghurt_500	7	6	4	8	7	-	8	10	10	9	9	12	9	14	15	14
Brandenburg_Joghurt_500	16	18	12	17	14	-	18	18	17	19	15	12	15	18	17	20
Rama	3	3	3	2	2	-	2	2	2	2	1	2	1	2	2	1

Tabelle 14: Runde Objekte

Literatur

[1] Klaus Gwinner und Arno Mitritz. Objekterkennung auf der Grundlage von Surface Normal Images. Projektbericht: Modellbasierte Objekterkennung, TU Berlin, Oktober 1994.

[2] Jong Hoon Park, Tae Gyu Chang und Jong Soo Choi. Three-Dimensional Object Representation and Recognition based on Surface Normal Images. *Pattern Recognition 26*, 6 (1993), 913–921.

[3] Rafał Sałustowicz, Christian Krüger und Alexander Bachem. Analyse von Farbhistogrammen zur 3D–Gestaltserkennung. Projektbericht, TU Berlin, 1993.

[4] Michael J. Swain und Dana H. Ballard. Color Indexing. *International Journal of Computer Vision 7*, 1 (1991), 11–31.