Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Ben Struss

Konzept und Entwicklung eines Verfahrens zur Suche von ähnlichen Bildern in Bilddatenbanken

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Ben Struss

Konzept und Entwicklung eines Verfahrens zur Suche von ähnlichen Bildern in Bilddatenbanken

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter: Prof. Dr.-Ing. Franz Korf

Abgegeben am 27. Februar 2009

Ben Struss

Thema der Bachelorarbeit
Konzept und Entwicklung eines Verfahrens zur Suche von ähnlichen Bildern in Bilddatenbanken
Stichworte
Bildvergleich Bilddatenbank Indizierung
Kurzzusammenfassung
In der vorliegenden Arbeit, wird ein hierarchisches Verfahren entworfen und realisiert, welches für die Suche von ähnlichen Bildern in großen Datenbanken geeignet ist. Dazu werden einige Techniken der Bildvorverarbeitung und eine Reihe von Deskriptoren entwickelt und vorgestellt. Diese werden mittels eines flexiblen Frameworks getestet und hinsichtlich der Sucheffizienz und Geschwindigkeit bewertet. Das Bildmaterial dieser Arbeit besteht aus ca. 31.000 Werbeanzeigen aus Publikumszeitschriften und stellt durch seine spezifischen Eigenschaften und die hohe Diversität der Motive besondere Anforderungen an das vorgestellte Verfahren.

Ben Struss

Title of the paper
Concept and design of a technique for similar image detection in image databases
Keywords
Imagecomparisation Advertising Database
Abstract
In this bachelor thesis a hierarchical approach for searching of similar images in large databases is presented. Some image preprocessing filters and a number of descriptors were implemented and introduced. For testing purposes and rating of the search efficiency a flexible framework was developed. This works image base consists of about 31.000 advertisements from magazines, which by its high diversity and specific characteristics has great demands on the proposed system.

Inhaltsverzeichnis

1 Einleitung

1.1 Einleitung

Ein Bild sagt mehr als tausend Worte...

So abgedroschen dieser Spruch auch ist, so gut beschreibt er doch das Dilemma der digitalen Bildverarbeitung. Für uns Menschen ist es ein leichtes, binnen Sekundenbruchteilen den Inhalt eines Bildes zu erfassen und die Ähnlichkeit zu einem anderen Bild zu bewerten. Für einen Computer dagegen ist es bisher unmöglich, von den bunten Pixelhaufen auf die Inhalte oder gar die Semantik des Bildes zu schließen.

Der Vorteil der digitalen Technik liegt viel mehr in dem perfekten Gedächtnis und damit der Möglichkeit viele Millionen Bilder abzuspeichern und wiederzufinden, während die meisten Menschen nicht einmal beim Memory mit 64 Kärtchen ohne Raten auskommen.

Problematisch wird es immer dann, wenn in dieser riesigen Datenmenge ein Bild gefunden werden soll, aber der Name vergessen wurde. Von Vorteil wäre es dann, wenn zu jedem Bild bestimmte Eigenschaften gespeichert wären und es dem Benutzer damit ermöglicht würde, dem Computer quasi zu „erklären“, welches Bild man gerne hätte.

Es gibt inzwischen eine Reihe von Systemen[1], welche es erlauben, Farben auszuwählen und grobe Formen zu zeichnen, die als Grundlage der Suche dienen. Auf sehr großen Datenbanken kann eine derart unscharfe Beschreibung des Bildes jedoch nur selten auf Anhieb ein zufriedenstellendes Ergebnis liefern, weshalb zur weiteren Verfeinerung der Anfrage meist ein oder mehrere Bilder aus dem ersten Ergebnis ausgewählt werden und eine neue Suche durchgeführt wird. Alternativ kann der Anwender auch selber ein Bild bereitstellen oder nach einer groben Stichwortsuche eines auswählen, dass dem Gesuchten möglichst ähnlich sieht.

Und obwohl die Erfassung semantischer Eigenschaften von Bildern Fortschritte macht (Vasconcelos, 2007), werden diese recht primitiven Verfahren wahrscheinlich auch noch viele Jahre lang Bestand haben, bevor Computer in der Lage sein werden die Inhalte eines Bildes zu „begreifen“ und Ähnlichkeiten anhand der Semantik zu erkennen.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, ein Verfahren zu entwerfen, welches die Erfassung von Werbeanzeigen aus Zeitschriften erleichtert. Bisher wird nach dem Scannen der Werbeanzeigen zunächst festgelegt, für welches Produkt geworben wird. Da jedoch ein bestimmtes Motiv nicht nur in einer Zeitschrift erscheint, sondern im Zuge einer Werbekampagne Schaltungen in vielen verschiedenen Magazinen sowie über längere Zeiträume haben kann, werden

[1] Unter anderem IBM QBIC (siehe Kapitel 2.2.1), VisualSEEK (Smith und fu Chang, 1996) und einige weitere, siehe (Veltkamp und Tanase, 2002)

in einem nachgelagerten Schritt regelmäßig alle Motive eines Produktes oder einer Marke aufgerufen und händisch nach gleichen Motiven durchsucht und gruppiert.

Um die bisher nötige mehrfache Zuordnung eines Motives zu einem Produkt zu vermeiden und die Häufigkeit und den Aufwand der nachgelagerten Gruppierung von gleichen Motiven auf ein Minimum zu reduzieren, soll bereits nach der Freistellung (dem Ausschneiden) der Anzeige in der Datenbank nach ähnlichen Bildern gesucht, die Ergebnisse präsentiert und eventuell übereinstimmende ausgewählt werden. Da zu diesem Zeitpunkt noch keinerlei semantische Informationen (Was ist auf dem Bild zu sehen?) über das Bild vorliegen, ist es nötig, möglichst markante Merkmale der Anzeigen zu extrahieren, indiziert zu speichern und in annehmbarer Zeit durchsuchbar zu machen. Es ist besonderer Wert darauf zu legen, dass wenig *false positives* präsentiert werden, jedoch gleichzeitig sichergestellt ist, dass vorhandene gleiche Motive auch gefunden werden.

1.3 Problemdarstellung

Da es sich bei dem Bildmaterial um digitalisierte Druckerzeugnisse handelt, weisen die Bilder durch Druck, Transport bzw. äußere Einwirkungen und das erneute Scannen häufig leichte bis ausgeprägte Störungen auf. Dies können Ränder, die nur schlecht oder gar nicht entfernt wurden (Abbildung 1), verfälschte Farben (Abbildung 2, links) oder auch leicht verschobene bzw. neu hinzugekommene Objekte sein (Abbildung 2, rechts). All diese Abweichungen in den Motiven müssen von dem Verfahren toleriert werden, da auch die meisten Menschen diese Motive als „gleich“ beschreiben würden.

Trotz dessen muss ein System, das auf einer Datenbank mit über einer halben Million Bildern eingesetzt wird, sehr selektiv arbeiten. Selbst wenn ein Suchvorgang nur 0,01% des Datenbestandes liefern würde, wären das immer noch über 50 Bilder, die von einem Menschen kaum in kurzer Zeit zu überblicken sind.

Es gilt daher Merkmale zu finden, die — jedes für sich genommen — tolerant gegenüber den genannten Störungen sind und sich in Kombination so ergänzen, dass die Ergebnisgröße handhabbar wird.

Abbildung 1: Störungen durch nicht oder schlecht entfernte Ränder

Abbildung 2: Störungen durch Farbabweichungen und Objektverschiebung

1.4 Gliederung der Arbeit

Die Gliederung der Arbeit richtet sich nach dem Entwicklungsprozess des Verfahrens und beginnt in Kapitel 2 mit einer Einführung in das Content based image retrieval. Es werden zwei Systeme vorgestellt und eine Übersicht über verbreitete Merkmale für die Bildbeschreibung gegeben.
Kapitel 3 befasst sich mit den Anforderungen an das zu entwickelnde Verfahren, beschreibt die Metriken der Erfolgsmessung, die Testumgebung und das entworfene Konzept.
In Kapitel 4 werden die Ansprüche an die Software definiert, die Designentscheidungen getroffen und das Ergebnis präsentiert.
Unter Einsatz der Software werden in Kapitel 5 Filter für die Vorverarbeitung entwickelt und auf ihre Tauglichkeit getestet.
Kapitel 6 stellt den Hauptteil der Arbeit dar. Es werden die verschiedenen Merkmale ausgewählt, implementiert und auf den Testdaten evaluiert.
Aus diesen Ergebnissen werden in Kapitel 7 die Merkmale und Deskriptoren auf Kombinierbarkeit untersucht und eine sinnvolle Konfiguration für das Verfahren erstellt.
Abschließend werden in Kapitel 8 die Ergebnisse des Verfahrens mit den zuvor aufgestellten Anforderungen verglichen und ein Ausblick auf die zukünftige Perspektive des Systems gegeben.

2 Grundlagen

2.1 Content based image retrieval

Für die Suche von Bildern in Bilddatenbanken gibt es nur zwei grundsätzliche Verfahren (Torres und Falcão, 2006).

Die erste Möglichkeit besteht darin, alle Bilder mit beschreibenden Metadaten zu versehen, was je nach Größe der Datenbank einen extremen Arbeitsaufwand darstellt und zu Konsistenzproblemen führt. So ist nur sehr schwer sicherzustellen, dass alle an der Verschlagwortung beteiligten Personen dieselben Begrifflichkeiten verwenden und ähnliche Situationen auch vergleichbar beschreiben. Des Weiteren ist es schlicht nicht realisierbar, jede mögliche Information, die ein Bild enthalten kann, zu erfassen und zu beschreiben. Zum Beispiel lässt sich etwas Abstraktes wie die Verteilung der dominanten Farben eines Bildes oder die „Rauhigkeit“ der Textur nur schwer objektiv beurteilen.[2]

Unter dem Begriff *Content Based Image Retrieval (CBIR)* werden allgemein alle Verfahren der zweiten Kategorie zusammengefasst. Hier werden für die Suche und Indizierung der Bilder keine Textbeschreibungen sondern zumeist numerische Merkmale aus den Bildern extrahiert und gespeichert bzw. verglichen. Mithilfe solcher Verfahren ist es möglich, auch sehr große Bilddatenbanken im Bereich von mehreren Hunderttausend bis einigen Millionen Bildern zu erfassen und durchsuchbar zu machen.

Für die Suche innerhalb von CBIR–Systemen wird zumeist das *Query-by-Example*[3]–Verfahren verwendet. Hierbei wählt der Nutzer entweder aus einer vorgegebenen Liste von Bildern eines aus oder kann ein eigenes Bild verwenden, von welchem dann zunächst noch die Merkmale extrahiert werden müssen. Das System wird dem Nutzer nun die gefundenen Ergebnisse, zumeist in absteigender Reihenfolge der Ähnlichkeit zum Suchbild sortiert, präsentieren. Falls die Suche nicht zu dem gewünschten Ergebnis führt, kann sie nun für ein Bild aus der Ergebnismenge fortgesetzt werden. Einige Systeme ermöglichen es dem Anwender auch, die Gewichtung der verschiedenen Merkmale zu modifizieren und die Suche so an die eigenen Bedürfnisse anzupassen. So kann z.B. der Fokus mehr auf ähnliche Farben gelegt und die Textur oder Form niedriger bewertet werden.

Eine weitere Variante der Suche ist das *Query-by-User-Sketch*–Verfahren. Der Nutzer kann hier mit einem speziellen Tool eine Skizze des gesuchten Bildes anfertigen oder bestimmte gewünschte Eigenschaften festlegen. Allgemein müssen die Suchalgorithmen in solchen Fällen eine viel größere Abweichung zum Suchbild zulassen, um noch brauchbare Ergebnisse liefern zu können. Da diese Art der Suche eher abstrakter Natur ist, wird sie nur selten verwendet.

[2] Siehe auch: (Niblack u. a., 1993)

[3] Siehe: (Datta u. a., 2008)

2.2 Verfügbare Systeme

Da der Bereich des CBIR seit vielen Jahren ein interessantes Thema für die Forschung ist und ein breites Spektrum an Einsatzmöglichkeiten bietet, haben sich viele Systeme entwickelt. Einen guten Überblick über 46 verschiedene Systeme bis zum Jahr 2002 bietet die Arbeit von Veltkamp und Tanase (Veltkamp und Tanase, 2002). Es werden im folgenden zwei Vertreter kurz vorgestellt.

2.2.1 QBIC

QBIC (Query By Image Content) ist ein CBIR–System, welches IBM Mitte der neunziger Jahre entwickelt und vorgestellt hat (Flickner u. a., 1995; Faloutsos u. a., 1994). Auf der (sehr spartanischen) Website[4] ist zu erfahren, dass das System in Form von *DB2 Image Extenders* in die Datenbank DB2[5] integriert wurde. Wie dem IBM Developer Forum jedoch zu entnehmen ist[6], wurde die Technologie nicht mehr weiterentwickelt und mit Erscheinen der Version 9 von DB2 wieder aus dem DBMS entfernt.

Features IBM setzt für QBIC auf eine breit gefächerte Auswahl von Features (Niblack u. a., 1993).

Um die Farbinhalte der Bilder zu beschreiben, wird der Farbmittelwert in den Farbdarstellungen RGB, Yiq, Lab und MTM berechnet, sowie ein Farbhistogram mit einer einstellbaren Anzahl an Werten genutzt.

Als Texturfeatures kommen abgewandelte Merkmale von Tamura zum Einsatz (Tamura u. a., 1978). Dafür wird die *coarseness*[7], der *contrast* sowie die *directionality* des Bildes berechnet.

Zur Beschreibung von Formen von Objekten muss das Bild zunächst segmentiert werden. Dies geschieht bei QBIC in einem manuellen oder semi-automatischen Verfahren, das den Anwender in der Auswahl und Kennzeichnung relevanter Objekte unterstützt. Diese werden zur Berechnung der Merkmale als binäre Bildmasken verwendet, bei denen alle Pixel gesetzt sind, die sich innerhalb der Objektgrenzen befinden. Von diesen Binärbildern werden die Fläche, die Zirkularität, die Exzentrität, die Hauptachsenausrichtung, sowie einige invariante algebraische Momente bestimmt.

Zusätzlich wird noch ein stark verkleinertes Kantenbild von 64×64 Pixeln erstellt, um die *Query-by-User-Sketch*–Funktionalität zu ermöglichen.

[4] http://wwwqbic.almaden.ibm.com/

[5] IBM DB2: http://www-01.ibm.com/software/data/db2/

[6] http://www.ibm.com/developerworks/forums/thread.jspa?threadID=102964&tstart=0

[7] am besten als Grobkörnigkeit zu übersetzen

Besonderheiten IBM legt den Fokus bei QBIC nicht ausschließlich auf den *Query-by-Example* Bereich, sondern ermöglicht es über angebotene Werkzeuge auch, nach bestimmten Farbanteilen in den Bildern (z.B. 20% Blau, 50%Grün, 30% Rot) zu suchen, sowie das gesuche Bild grob zu skizzieren. Zusätzlich lassen sich in dem System die Suchvorgänge auch mit händisch vergebenen Schlüsselwörtern kombinieren.

Ein Beispiel des QBIC Systems findet sich auf der Website des Eremitage Museum St. Petersburg [8] unter der URL: `http://www.hermitagemuseum.org/fcgi-bin/db2www/qbicSearch.mac/qbic?selLang=English`.

2.2.2 FIRE

FIRE (Flexible Image Retrieval Engine) wurde 2004 von Dr. Thomas Deselaers und Anderen an der RWTH Aachen entwickelt (Deselaers u. a., 2005). Das System soll vor allem der Evaluierung verschiedener Merkmale und Distanzfunktionen dienen und wird hauptsächlich zu Forschungszwecken eingesetzt. Es ist unter der GNU Public License auf der Homepage von Dr. Deselaers[9] verfügbar.

Features FIRE verwendet kein festes Feature–Schema, sondern bietet eine große Auswahl bereits implementierter Merkmale. Diese können je nach Bedarf beliebig kombiniert werden.

Diese Sammlung enthält unter anderem Farbhistogramme, Korrelogramme, Tamura Textur Merkmale, Gabor Features und lokale Features wie SIFT[10] (Deselaers u. a., 2008).

Besonderheiten Das System beinhaltet zusätzlich eine beachtliche Anzahl an Distanzfunktionen für die Bewertung der Bildähnlichkeiten. Um die Suchergebnisse weiter den persönlichen Erwartungen anpassen zu können, gibt es zudem die Möglichkeit des Relevanz–Feedbacks. Hier wird dem Benutzer einer Auswahl der n besten Bilder präsentiert, die dieser in die Kategorien relevant, neutral und nicht relevant einteilt. Anhand dieser Klassifikation wird die Suche verfeinert und erneut ausgeführt.

2.2.3 Fazit

Das größte Problem von CBIR–Systemen liegt in der Vielzahl unterschiedlicher Anwendungsgebiete und dem jeweils zugehörigen Bildmaterial. So sind die Anforderungen an ein System zur Speicherung von medizinischen Daten wie Röntgenbildern oder CT-Scans völlig anders geartet als z.B. die Suche von Blumenbildern oder Pressefotos von Prominenten.

[8] http://www.hermitagemuseum.org/html_En/index.html

[9] `http://www-i6.informatik.rwth-aachen.de/~deselaers/fire/`

[10] Scale–invariant feature transform

Es ist nahezu unmöglich, Systeme zu entwickeln, die ohne spezielle Anpassungen in allen Bereichen zufriedenstellende Ergebnisse vorweisen können. Das vorgestellte System FIRE (Kapitel sec:syst:fire) ist ein gutes Beispiel dafür. Es folgt genau diesem Ansatz und stellt nur Basisfunktionalitäten bereit. Die Auswahl der Merkmale wird hier dem Benutzer überlassen.

Diese Vielfältigkeit der Einsatzgebiete zeigt sich auch an öffentlich verfügbaren Bilddatenbanken, die sehr unterschiedliche Bereiche abdecken (Deselaers u. a., 2008). So enthält beispielsweise die WANG–Datenbank[11] insgesamt 1.000 Bilder, die in 10 Gruppen zu je 100 Bildern eingeteilt ist. Diese Gruppen sind Afrika, Strand, Monumente, Busse, Essen, Dinosaurier, Elefanten, Blumen, Pferde und Berge. Verfahren, die mit dieser Datenbank testen, stehen also vor einem Klassifizierungsproblem mit relativ wenigen Gruppen.

Dagegen enthält die UW–Datenbank der Universität Washington 1.109 Bilder, welche jedoch nicht zu Gruppen zusammgefasst, sondern mit insgesamt 6383 Schlagwörtern versehen sind[12]. Die Überprüfung der Ergebnisse findet hier durch einen Vergleich der Schlagworte miteinander statt, bei dem all die Ergebnisse als positive Übereinstimmung gezählt werden, welche eine nicht leere Schnittmenge von Wörtern aufweisen.

Des Weiteren gibt es noch die IRMA–Datenbanken[13], welche über 10.000 medizinische Röntgenaufnahmen enthalten (je nach Version), die ZuBuD–Datenbank[14], bestehend aus 1.120 Fotos von Züricher Gebäuden, sowie die UCID–Datenbank (Schaefer und Stich, 2004), die extra als Testdatenbank für CBIR erstellt wurde und in der ersten Version manuell vergebene Relevanzwerte für jede mögliche Kombination von zwei der 264 Bilder enthält. Die zweite Version dieser Datenbank enthält bereits 1.338 unkomprimierte Bilder, zu welchen jeweils eine oder mehrere Mengen relevanter Bilder angegeben sind.

Die genannten Einschränkungen betreffen diese Arbeit gleich in mehrfacher Hinsicht. Es gibt unter den verbreiteten Testdatenbanken keine, die auch nur ansatzweise die Eigenheiten von gescanntem Bildmaterial unterschiedlicher Druckherkunft nachbildet. Des Weiteren sind die Datenbanken mit einem Umfang von maximal gut 10.000 Bildern relativ klein und eignen sich kaum, um Performancemessungen sowie Indizierungstests durchzuführen. Auch der geforderte Grad der Ähnlichkeit differiert sehr stark von dem, was mit dieser Arbeit angestrebt wird, nämlich in einer sehr großen Datenbank mit einer Vielzahl von Motivgruppen die fast gleichen Bilder zu finden.

[11] Benannt nach James Z. Wang (`http://wang.ist.psu.edu/docs/related.shtml`)

[12] Verfügbar unter: `http://www.cs.washington.edu/research/imagedatabase/`

[13] Projekt Website und Download: `http://libra.imib.rwth-aachen.de/irma/datasets_en.php`

[14] `http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html`

2.3 Verbreitete Merkmale

Da der Bereich des CBIR bzw. der Indizierung von Bildmaterial schon seit einigen Jahrzehnten ein beliebtes Forschungsthema ist, sind entsprechend viele, mehr oder weniger verschiedene Merkmale zur Bildbeschreibung bekannt.

Die meisten dieser Merkmale lassen sich in 4 große Kategorien einordnen (Deselaers u. a., 2008; Veltkamp und Tanase, 2002; Birgale u. a., 2006; Rui u. a., 1999).

2.3.1 Farbe

Farbe ist (sofern vorhanden) eines der wichtigsten Kriterien für die Wahrnehmung von Ähnlichkeit in Bildern (Manjunath u. a., 2001).

Eines der einfachsten und am häufigsten verwendeten Farbmerkmale ist das *Farbhistogramm* (Deselaers u. a., 2008; Hafner u. a., 1995). Die Erstellung eines solchen ist sehr einfach und erfordert kaum Rechenaufwand, da das Bild Pixel für Pixel eingelesen wird und nur der Zähler für die entsprechende Farbe erhöht werden muss. Um die Dimensionalität des Histogramms in Grenzen zu halten, werden die Werte der Farbkanäle auf einige wenige Bereiche quantisiert. Je nach Farbdarstellung kann die Quantisierung symmetrisch oder asymmetrisch erfolgen. So kann der RGB–Farbraum auf jeweils 8 Abstufungen pro Kanal reduziert werden, was zu $8 \times 8 \times 8 = 512$ Histogrammwerten führen würde. Bei HSV– oder HSL–Darstellung bietet sich eine feinere Abstufung des Hue-Kanals an, wobei der Saturation und Value/Luminance Anteil entsprechend gröber eingeteilt wird. Das könnte z.B. zu einem $16 \times 4 \times 4 = 256$ Bin–Histogramm führen.

Der große Nachteil der *Farbhistogramme* ist, dass keinerlei Aussagen über die räumliche Verteilung der Farben im Bild gemacht werden.

Um diese Verteilung der Farben mit zu berücksichtigen, wurden unter anderem *Farb–Korrelogramme* entwickelt (Huang u. a., 1997). Hier werden für alle Kombinationen zweier (quantisierter) Farben i und j die Wahrscheinlichkeiten berechnet, dass ein Pixel der Farbe i in einer Entfernung k von einem Pixel der Farbe j gefunden wird. Jedoch hat auch dieser Ansatz den Nachteil, relativ hochdimensionale Merkmalsvektoren zu erzeugen.

Die *Moving Picture Experts Group (MPEG)* begann 1996 mit der Entwicklung von MPEG–7, welches 2002 zum ISO-Standard wurde.[15] In MPEG–7 werden unter anderem 4 Deskriptoren für Farbinformationen in Bildern definiert (Manjunath u. a., 2001).
Diese sind im Einzelnen:

Dominant Color ist ein Deskriptor, welcher bis zu 8 repräsentative Farben des Bildes und ihre Eigenschaften wie den Anteil am Gesamtbild, die räumliche Homogenität sowie optional die Varianz innerhalb der Farbe speichert.

[15] `http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm`

Scalable Color ist ein Haar-transformiertes, skalierbares Farbhistogramm, welches in der Größe an die Bedürfnisse der Anwendung angepasst werden kann und kompakt speicherbar ist.

Color Layout ist eine sehr kompakte Darstellung der Farbverteilung im Bild. Hierfür wird das Bild zunächst in 64 (8×8) Blöcke zerlegt und für jeden eine repräsentative Farbe im YCbCr–Farbraum bestimmt. Dieses 8×8 Feld wird mittels einer diskreten Kosinustransformation in Koeffizienten zerlegt, von denen die ersten 6 (Y–Kanal) bzw. 3 (Cr– sowie Cb–Kanal) gespeichert werden. In der Standardeinstellung beträgt die Länge des Deskriptors nur 63 Bit.

Color-Structure ist ein Histogramm mit 32, 64, 128 oder 256 Bins, welche nichtlinear im HMMD-Farbraum verteilt sind. Im Gegensatz zu einem normalen Farbhistogramm liefert es jedoch unterschiedliche Ergebnisse für zwei Bilder mit den gleichen Farbanteilen, aber einer unterschiedlichen räumlichen Verteilung innerhalb des Bildes. Dazu wird ein 64 Felder großes (8×8) Strukturelement über das Bild bewegt und in den Histogramm-Bins gezählt, an wievielen Positionen des Strukturelements eine entsprechende Farbe abgedeckt war.

2.3.2 Textur

Textur–Features machen Aussagen über die Beschaffenheiten der Oberflächen und der abgebildeten Strukturen. Bereits 1978 hat Tamura (Tamura u. a., 1978) sechs Merkmale von Oberflächen beschrieben, die dem menschlichen Sehen nachempfunden sind. Laut Deselaers (Deselaers u. a., 2008) sind besonders die ersten drei Merkmale, namentlich *coarseness*, *contrast* und *directionality*, für den Einsatz in CBIR–Systemen relevant. Die *coarseness* ist ein Maß für die „Grobkörnigkeit“ oder „Rauheit“ einer Textur und wird über das komplette Bild gemittelt. Die Priorität liegt hierbei auf den gröberen Strukturen.

Der *contrast* macht Aussagen über die Verteilung der Grauwerte des Bildes und die *directionality* ist ein 16–Bin Histogramm, welches die Anteile von Kanten in bestimmten Richtungen zählt.

Auch unter die Textur–Features fallen Deskriptoren, die auf der Verteilung und den Eigenschaften von Kanten im Bild basieren. Der im MPEG–7 Standard enthaltene Edge Histogram Descriptor ist ein solcher. Er kategorisiert Kanten nach ihrer Ausrichtung in 5 verschiedene Gruppen und macht Aussagen über die lokale Verteilung der Kanten im Bild.

2.3.3 Form

Für die Extraktion von Merkmalen, die die Formen eines Objekts beschreiben, ist es zunächst erforderlich, das Bild zu segmentieren. Da eine vollautomatische Segmentierung

komplexer Bilder wie Fotos noch nicht möglich ist, werden diese Features eher in Bereichen der Bildmustererkennung eingesetzt (Torres und Falcão, 2006; Deselaers u. a., 2008).

Eine der am weitesten verbreiteten Methoden ist die Berechnung invarianter Momente für die segmentierten Objekte (Sebe und Lew, 2002). Hierzu können die 7 von Hu in (Hu, 1962) vorgestellten Momente verwendet werden, welche invariant bezüglich Translation, Skalierung und Rotation sind.

2.3.4 Lokale Merkmale

Lokale Features erfreuen sich spätestens seit der Veröffentlichung der SIFT–Features 1999 durch Lowe (Lowe, 1999) großer Beliebtheit.

Zur Extraktion von SIFT–Features, werden zunächst sogenannte Interest–Points gesucht. Dafür wird ein Bild in verschiedenen Stärken mit einem gaußschen Weichzeichner gefiltert und in mehreren Stufen verkleinert. Wenn ein Pixel im Vergleich zu seinen direkten Nachbarn und denen in der darüber und darunter liegenden Ebene ein lokales Maximum oder Minimum ist, dann ist er ein Kandidat für einen Interest–Point. Die Kandidatenmenge wird nun um instabile Punkte bereinigt und für jeden Punkt die Orientierung und ein Deskriptor der Umgebung bestimmt. Diese Deskriptoren bestehen aus 16 Histogrammen mit je 8 Bins, die die Orientierungen von Pixeln im Umkreis darstellen. Pro Bild können einige hundert bis deutlich über tausend solcher Punkte identifiziert werden.

In (Foo und Sinha, 2007) wird gezeigt, dass auch mit deutlich weniger Punkten eine relativ zuverlässige Suche nach fast gleichen Bildern möglich ist.

Mit SURF[16] wurde 2006 ein sehr ähnliches Feature vorgestellt (Bay u. a., 2006). Im Gegensatz zu SIFT ist dieses hauptsächlich auf hohe Geschwindigkeit ausgelegt und versucht, trotz kompakterer Darstellung, eine ähnlich hohe Präzision zu gewährleisten. Ein Vergleich dieser zwei Features in verschiedenen Implementierungen wurde in Bauer u. a. vorgenommen. Dort wird gezeigt, dass die SURF–Features etwas ungenauer, jedoch auch deutlich schneller sind.

2.3.5 Andere Verfahren

Abseits der 4 großen Gruppen, gibt es noch einige hochspezialisierte Verfahren, welche die Bilder ausschließlich auf sehr bestimmte Merkmale untersuchen und den Rest komplett ignorieren. Ein Beispiel ist die biometrische Gesichtserkennung, die als Unterstützung einer Suche nach Bildern bestimmter Personen in Agenturdatenbanken denkbar ist.

Auch die durch Optical Character Recognition (OCR) gewonnenen Textinhalte von Bildern könnten für einen Vergleich verwendet werden. Ein System auf dieser Basis kann für

[16] Speeded Up Robust Features

die Klassifizierung von Eingangsdokumenten in Dokumenten–Management–Systemen eingesetzt werden.

3 Konzeptüberlegung

3.1 Basis

Die Datenbasis, die für diese Arbeit zur Verfügung steht[17], umfasst knapp 800.000 in 300dpi gescannte Werbeanzeigen der letzten 2,5 Jahre. Ein produktiver Einsatz auf diesem ständig wachsenden Datenbestand ist das große Ziel für das Verfahren, das im Rahmen dieser Arbeit entwickelt werden soll.

3.2 Zielsetzung

Es soll erreicht werden, dass das Verfahren sich in den täglichen Workflow integrieren lässt und ein flüssiges Arbeiten ermöglicht. Hierfür ist es erforderlich, dass die Merkmale der Bilder in akzeptabler Zeit extrahiert werden können und die Suche nach den ähnlichen Bildern relativ schnell geht. Da die Vorverarbeitung und Merkmalsextraktion bereits parallel zum Ausschneiden der Bilder erfolgen kann, ist die Dauer dessen nicht ganz so entscheidend für den gesamten Ablauf. Dennoch sollte sich dieser Vorgang im Bereich von unter 5 Sekunden pro Bild abspielen.

Für die eigentliche Suche in der Datenbank ist eine schnelle Reaktionszeit wesentlich wichtiger. Hier sollten die Ergebnisse nach Möglichkeit binnen höchstens einer Sekunde präsentiert werden, um keine zu großen Wartezeiten aufkommen zu lassen. Des Weiteren ist eine Positionierung des gesuchten Bildes, sofern in der Datenbank vorhanden, unter den ersten 5 Ergebnissen wünschenswert, so dass keine großen Bildmengen vom Mitarbeiter durchgesehen werden müssen.

3.3 Anforderungen

Technisch ergeben sich daraus die folgenden Anforderungen:

- Die Vorverarbeitung und Merkmalsextraktion darf nicht länger als ca. 5 Sekunden dauern.
- Nach der Aufnahme des Bildes in die Datenbank muss es sofort anhand seiner Merkmale findbar sein.

[17] Freundlicherweise bereitgestellt von der Firma: AdVision digital GmbH (`http://www.advision-digital.de/`)

- Um den Speicherplatzbedarf der Datenbank in Grenzen zu halten, sollten die Deskriptoren möglichst kompakt sein.
- Die Struktur und der Detailgrad der Deskriptoren muss eine effiziente Indizierung erlauben.
- *False-Negatives* d.h. die Ablehnung eines passendes Bildes sollten auf ein Minimum reduziert werden.
- Der gesamte Vorgang der Suche sowie die Sortierung der Ergebnisse nach Relevanz soll im Mittel innerhalb einer Sekunde abgeschlossen sein.
- Falls Treffer vorhanden sind, sollten diese im Ergebnis innerhalb der ersten 5 Einträge auftauchen.

3.4 Vorgehen

Um die Ergebnisse der Arbeit objektiv beurteilen zu können, werden zunächst Metriken definiert, die die technischen Anforderungen überprüfbar und bewertbar machen.

Da für die Berechnung der Metriken gewisse Informationen über die zugrundeliegende Bilddatenbank nötig sind, werden mehrere Testsets von Bildern definiert, für die diese Informationen bekannt sind und die für verschiedene Zwecke verwendet werden können.

Zuletzt wird auf Basis der Metriken eine mögliche Struktur eines Verfahrens entwickelt, das den gestellten Anforderungen gerecht wird.

3.4.1 Metriken

Zwei der einfachsten und am weitesten verbreiteten Metriken zur Beurteilung von CBIR-Systemen sind die *precision* und der *recall* (Chen und Stentiford, 2006; Deselaers u. a., 2008).

Precision Sei Q eine Anfrage an die Datenbank und $R(Q)$ das Ergebnis, dann ist die *precision* $P(Q)$ definiert als:

$$P(Q) = \frac{\text{Anzahl der relevanten Ergebnisse in } R(Q)}{\text{Anzahl der Ergebnisse in } R(Q)}$$

Sind alle Ergebnisse der Anfrage relevant, nimmt die *precision* den Wert 1 an. Diese Metrik ist sehr einfach, hat aber auch einige gravierende Nachteile. So ist die *precision* nur eingeschränkt brauchbar für Anwendungsfälle, in denen mehr Ergebnisse abgefragt werden als tatsächlich relevante Ergebnisse für die Suche vorhanden sind, da hier der Maximalwert nie erreicht werden kann. Des Weiteren wird keine Aussage über die Position der Treffer im

Ergebnis gemacht, so dass eine Anfrage mit 10 Ergebnissen, bei denen die 5 relevanten Treffer auf den Plätzen 1-5 stehen, genauso gut bewertet wird, wie ein Ergebnis, bei dem die Treffer sich auf den Plätzen 6-10 befinden.

Da in dem Anwendungsbereich dieser Arbeit die Anzahl relevanter Treffer im Vergleich zur Größe der Ergebnisse relativ klein sein wird, ist die *precision* als Metrik für die Effizienz des Verfahrens nicht geeignet.

Recall Der *recall* berechnet sich ähnlich. Es gilt:

$$Rec(Q) = \frac{\text{Anzahl der relevanten Ergebnisse in } R(Q)}{\text{Anzahl der relevanten Ergebnisse insgesamt}}$$

Auch der *recall* macht keinerlei Aussage über die Verteilung der Treffer im Suchergebnis, kann jedoch bei der Vermeidung von *False-Negatives* helfen. Dafür muss die Menge der Ergebnisse allein durch die maximale Distanz zwischen Suchbild und Ergebnisbild bestimmt werden, und es darf kein festes Fenster (Zeige die ersten 20 Treffer) verwendet werden. In dem Fall ist ein *recall* von $Rec(Q) < 1$ ein Zeichen dafür, dass relevante Treffer durch eine zu niedrige Entfernungsgrenze ausgeschlossen wurden.

Da für diese Arbeit noch ein gesondertes Kriterium für den fehlerhaften Ausschluss definiert und zumeist eine Ergebnisgröße — basierend auf der Größe der Gruppen — verwendet wird, spielt auch der *recall* keine besondere Rolle.

Normalized Modified Retrieval Rank Im Zuge der Entwicklung von MPEG-7 wurde eine Metrik zur Beurteilung der eigenen Deskriptoren entwickelt, welche sowohl die Vollständigkeit des Ergebnisses, als auch die Positionen innerhalb der Ergebnismenge berücksichtigt.[18]

Der *Normalized Modified Retrieval Rank (NMRR)* kann Werte zwischen 0 und 1 annehmen, wobei 0 für ein perfektes Ergebnis steht, in dem sich alle gesuchten Bilder auf den ersten Positionen befinden, und 1 ein Ergebnis darstellt, in dem kein einziges gesuchtes Bild gefunden wurde.

Aufgrund der Universalität dieser Metrik wird sie in der vorliegenden Arbeit als eines der beiden Hauptkriterien für den Erfolg des Verfahrens verwendet.

Goal Success Rate Um die Erfüllung der gesetzten Anforderung an das Verfahren überprüfen zu können, wird für diese Arbeit eine *Goal Success Rate* auf einer Menge von Suchanfragen definiert, welche beschreibt, wie hoch der Anteil der Suchanfragen ist, in denen das gesetzte Ziel erreicht wurde. Hierzu wird zunächst die Reihenfolge der Gruppen G innerhalb des Ergebnisses R bestimmt, indem, beginnend mit dem ersten Element, über das Ergebnis iteriert wird und sobald eine Gruppe $G(R_i) \notin L$ gefunden wird, diese in die Liste der Gruppen L aufgenommen wird.

[18] Für die Herleitung und genaue Beschreibung siehe (Manjunath u. a., 2001)

Auf dieser Liste L definiert sich die binäre Metrik *Goal Success* $GS(Q)$ wie folgt:

$$GS(R(Q)) = \begin{cases} 1 & \text{wenn die Position von } G(Q) \text{ in } L \leq 5 \\ 0 & \text{sonst} \end{cases}$$

Wobei $G(Q)$ die Gruppe des gesuchten Bildes bezeichnet.

Für die Berechnung der *Goal Success Rate GSR* auf einer Menge von Ergebnissen $RS = \{R(Q_1), \ldots, R(Q_n)\}$ wird der Mittelwert aller $GS(R(Q))$ berechnet.

$$\text{GSR}(RS) = \frac{1}{n} \sum_{i=1}^{n} GS(R(Q_i))$$

Dies ist das zweite Hauptkriterium, welches es zu optimieren gilt und direkt eine der Anforderungen an das Verfahren beschreibt.

False Exclusion Rate Um ferner die Eignung einer Vorfilterung (siehe 3.5) beurteilen zu können und sicherzustellen, dass die relevanten Ergebnisse gefunden werden können, wird die *False Exclusion Rate (FER*(Q)*)* definiert, die das Verhältnis zwischen der Anzahl relevanter Ergebnisse insgesamt und der Menge der fälschlicherweise ausgeschlossenen Elemente beschreibt.

$$\text{FER}(Q) = \frac{\text{Anzahl der fälschlicherweise ausgeschlossenen Elemente}}{\text{Anzahl der relevanten Ergebnisse Insgesamt}}$$

Im Unterschied zum *recall* wird bei dieser Metrik keine Rücksicht auf eine eventuell festgelegte Beschränkung der Ergebnisse auf die ersten n Elemente genommen. Es werden nur solche Elemente gezählt, die durch eine gesonderte Vorfilterung des Gesamtbestandes ausgeschlossen wurden.

Performance Metriken Damit die Einhaltung der gesetzten zeitlichen Anforderungen an die einzelnen Schritte des Verfahrens überprüft werden kann, werden für die Teilbereiche Vorverarbeitung, Merkmalsextraktion und Suche in der Datenbank jeweils die Zeiten gestoppt und angezeigt.

Überall wo eine automatisierte Verarbeitung mehrerer Elemente erfolgt, wird die durchschnittliche Zeit pro Element berechnet.

3.4.2 Testdatensätze

Für alle definierten Metriken ist es erforderlich, dass für jedes Element eine Zuordnung zu einer Gruppe definiert ist. Jede Gruppe hat $1 - n$ Elemente, die untereinander als relevant eingestuft sind. Glücklicherweise stehen neben den Anzeigen selber auch die händisch

gepflegten Informationen über die Zusammenfassung gleicher bzw. ähnlicher Anzeigen zu Motivgruppen zur Verfügung.

Da 800.000 Anzeigen jedoch einen enormen Anspruch an den Speicherplatz und die Rechenzeit stellen würden, wurden als repräsentative Untermenge alle gescannten Anzeigen des Monats September 2008 kopiert. Die 31.411 Bilder aus dem September sind zwar schon wesentlich besser handhabbar, aber wenn man von optimistischen 0,5 Sekunden Bearbeitungszeit pro Bild ausgeht, würde ein Testlauf immer noch über 4 Stunden Rechenzeit beanspruchen.

Damit es trotzdem möglich ist, in annehmbarer Zeit zu belastbaren Ergebnissen zu kommen, wurden zwei Testmengen zusammengestellt, die jeweils unterschiedliche Bereiche abdecken.

Die erste Testmenge besteht aus 2.796 Bildern, von denen je zwei das gleiche Motiv zeigen und in einer Gruppe angeordnet sind. Diese Zusammenstellung eignet sich besonders für die Bewertung der Suchgenauigkeit, da für jede Suche das Verhältnis zwischen gewünschten Ergebnissen zu den nicht relevanten Bildern mit 1:2.794 sehr hoch ist.

In Abbildung 3 sind zwei Bildpaare aus dem Set abgebildet, von denen das zweite auch direkt als Beispiel für die geforderte Toleranz dient. So ist das Hauptmotiv zwar gleich, jedoch unterscheidet sich der untere Bereich deutlich.

Abbildung 3: Zwei Bildpaare aus der ersten Testzusammenstellung

Als zweite Zusammenstellung wurden 2.110 Bilder ausgewählt, welche jeweils zu Gruppen von 10–20 Anzeigen zusammengefasst sind. Diese Bildsammlung wird vor allem dazu verwendet, die Streuung der Merkmale innerhalb der Gruppen zu erfassen und zu bewerten. Dies ist besonders zur Findung der zulässigen Toleranzen wichtig.

Die Bilder einer Gruppe in Abbildung 4 vermitteln einen guten Eindruck davon, in welchem Bereich sich die Abweichungen innerhalb einer Gruppe bewegen können. So variiert nicht nur die Helligkeit zwischen den Bildern, sondern auch oft die Breite der Ränder um das eigentliche Motiv herum.

Abbildung 4: Bilder einer 20er Gruppe des zweiten Bildsets

3.4.3 Testumgebung

Die Basis jeden CBIR-Systems, welche über Erfolg oder Misserfolg entscheidet, sind die Merkmale, die zur Beschreibung der Bilder verwendet werden. Um verschiedene Deskriptoren auf den gewählten Testdaten testen zu können und die definierten Metriken zu ermitteln, wird ein Framework benötigt, das alle nötigen Funktionalitäten bereitstellt.

Da es leider keine frei verfügbare Software gibt, welche die gestellten Anforderungen erfüllen könnte, wird sich ein Teil dieser Arbeit mit der Entwicklung einer geeigneten Softwareumgebung befassen. Aufgrund des Umfangs wird diesem Punkt ein eigenes Kapitel gewidmet, siehe Kapitel 4, S. 21.

3.5 Konzept

Da die Zielsetzung vorsieht, dass nicht nur beliebige ähnliche Bilder gefunden werden sollen, sondern die beinahe Duplikate der gesuchten Bilder an den ersten Stellen auftauchen müssen, liegt es nahe, dass die Deskriptoren für einen derart genauen Vergleich komplexer und in der Berechnung aufwendiger sind. Um diese rechenintensiven Vergleiche möglichst auf ein Minimum zu reduzieren, ist die Entscheidung auf ein hierarchisches Verfahren gefallen, das die Deskriptoren nach ihrer Komplexität untereinander anordnet. Für einen ähnlichen Ansatz siehe (Patino-Escarcina und Costa, 2007).

So wird mit jeder Stufe i der Merkmalshierarchie eine Menge A_i von Kandidaten aus der Gesamtmenge D bzw. der Kandidatenmenge A_{i-1} der vorhergehenden Ebene herausgefiltert. Nach dem letzten Filterungsschritt werden die verbleibenden Kandidaten anhand eines oder mehrerer komplexer Merkmale ihrer Entfernung zum Suchbild nach aufsteigend sortiert und die ersten n Bilder dem Anwender präsentiert (Abbildung 5).

Es muss in jedem Fall sichergestellt werden, dass möglichst wenige „beinahe Duplikate“ durch die Filterung fälschlicherweise ausgeschlossen werden. Hierzu ist es erforderlich Toleranzen der Merkmale innerhalb von Gruppen fast gleicher Bilder zu ermitteln und diese in dem Ausschlussprozess zu berücksichtigen.

Ein weiterer Vorteil dieses Ansatzes ist, dass zunächst nur wenig Aufwand in komplexe Indexstrukturen investiert werden muss. Wenn für die Vorfilterung der Kandidatenmenge ausschließlich Merkmale verwendet werden, die sich als reelle Zahlen darstellen lassen, so kann man einen n–dimensionalen Raum aufspannen, wobei n gleich der Anzahl der Merkmale ist. Werden die Merkmale auf einen Bereich von z.B. 0–255 normiert und auf ganze Zahlen gerundet, so lässt sich diese Struktur auch als ein n–dimensionales Array der Größe 255^n nachbilden. Je nach ermittelter Toleranzanforderung an die einzelnen Merkmale, kann das Merkmal durchaus auf einen kleineren Bereich normiert werden und entsprechend Speicherbedarf einsparen.

Eine solche Arraystruktur ermöglicht einen verhältnismäßig schnellen Zugriff auf die Menge der Kandidaten, die durch die komplexen Merkmale verglichen werden. Auch erfüllt diese Art Index die formulierte Anforderung, dass neu eingefügte Objekte sofort auffindbar sind, da hier nichts weiter geschehen muss, als das neue Objekt anhand seiner Merkmale in die entsprechende Zelle des Arrays einzufügen.

Da der Fokus dieser Arbeit zunächst auf der Entwicklung und Ermittlung geeigneter Merkmale zum Vergleich der Bilder liegt, wird hier nicht weiter auf komplexere Indizierungsmethoden für metrische Räume[19] eingegangen. Dem geneigten Leser kann ich die folgenden Arbeiten empfehlen (Batko u. a., 2004; White und Jain, 1996; Chavez und Navarro, 2000; Hjaltason und Samet, 2003).

[19] Ein metrischer Raum ist eine Menge D von Objekten $o_1, \ldots, o_n$ auf denen eine Distanzfunktion $d(o_i, o_j)$ definiert ist (Batko u. a., 2004).

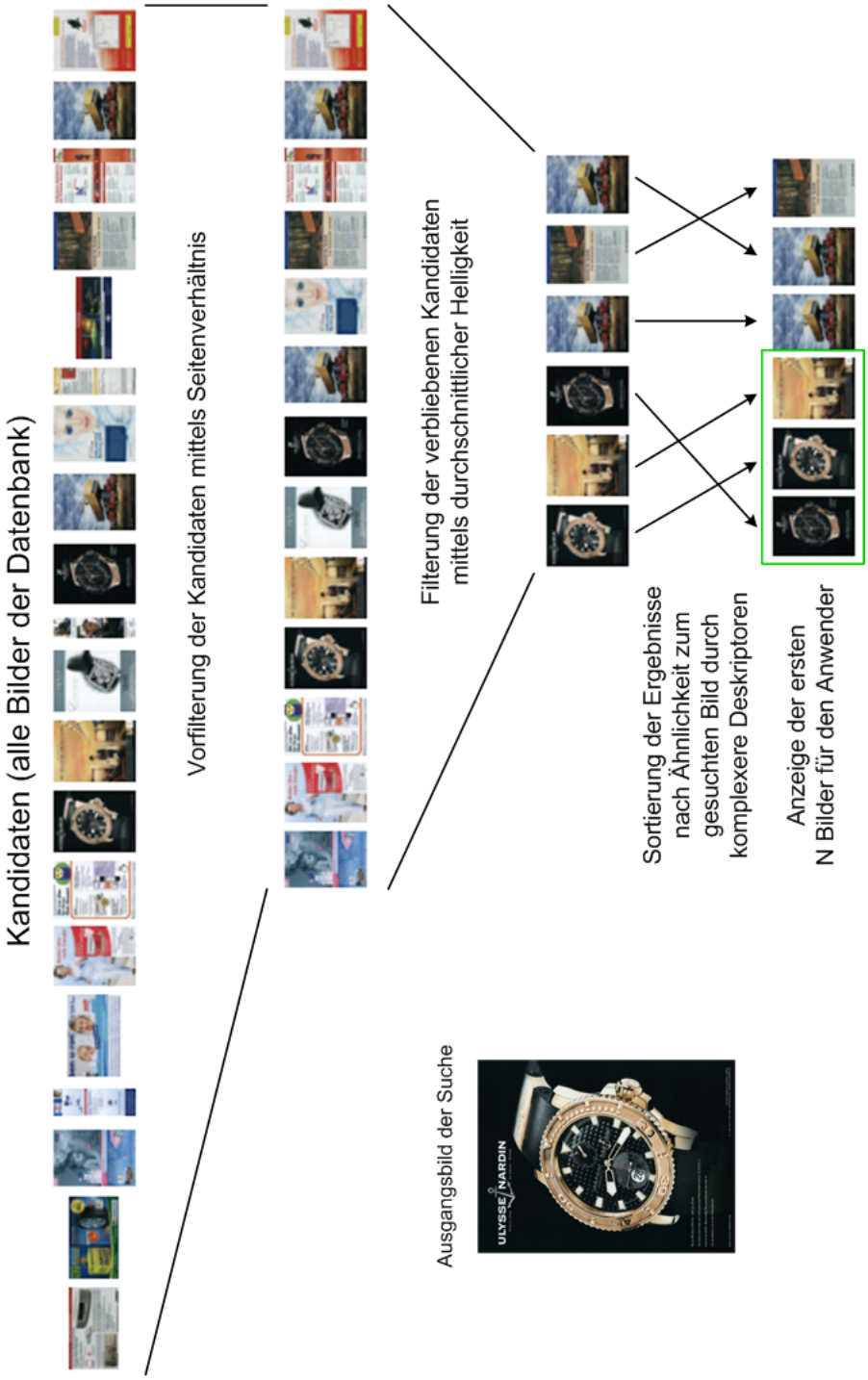


Abbildung 5: Schematischer Ablauf des Verfahrens

4 Die Software

4.1 Ziel

Die Software soll die Findung und die Bewertung von Merkmalen zum Vergleich von Bildern unterstützen. Hierfür ist es erforderlich, den gesamten Prozess von der Vorverarbeitung über die Merkmalsextraktion bis zur automatisierten Sammlung von Performanzmetriken zur Bewertung des aktuellen Verfahrens abzubilden.

Damit das Ergebnis kein starres Konstrukt ist, sondern vielfältig eingesetzt werden kann und flexibel erweiterbar ist, sollte die Möglichkeit bestehen, weitere Methoden und Merkmale hinzuzufügen.

Die Daten sollten anschaulich dargestellt werden können und ein intuitives Arbeiten nach kurzer Zeit möglich sein.

4.2 Anforderungen

Die Aufteilung der Anforderung erfolgt wie üblich in die drei Kategorien *„must“*, *„should“* und *„nice-to-have“*.

must Alle Features, die zur Zielerfüllung unbedingt erforderlich sind:

- Laden von Bildern und Darstellung derselben
- Laden von Gruppendefinitionen, die die Bilder in Gruppen einteilen
- Ausführung einer oder mehrerer Vorverarbeitungsschritte in beliebiger Reihenfolge auf den geladenen Bildern
- Extraktion eines oder mehrerer Merkmale für alle geladenen Bilder
- (optionale) Anzeige der extrahierten Merkmale
- Anzeige sowohl von Bild- als auch Gruppeninformationen
- Statistische Auswertungen der extrahierten Merkmale
- Einstellbare Suchoptionen wie die Reihenfolge und Toleranzen der Vorfilterung und der Gewichtung der Deskriptoren
- Durchführung von Suchanfragen auf den geladenen Bildern für eines der Bilder unter Verwendung der angepassten Suchoptionen
- Erstellung von globalen Metriken, indem automatisierte Suchen für jedes geladene Bild durchgeführt werden

should Features, die bei der Nutzung behilflich, aber nicht zwingend nötig sind:

- Laden der Vorverarbeitungsfilter und der Merkmalsextraktoren als Plugins
- Speichern und Laden der erstellten Konfigurationen und Optionen
- Multiprozessorunterstützung für besonders rechenaufwendige Vorgänge
- Manuelle Festlegung von Gruppenzugehörigkeiten im Programm
- Speicherung der extrahierten Merkmale in einem standardisierten Format, damit es von anderen Applikationen weiterverwendet werden kann

nice-to-have

- Speicherung der Konfigurationen und Einstellung in menschenlesbarem Format (z.B. XML)
- Automatische Generierung von HTML oder ähnlichen Reports über die Statistiken der Suche

4.3 Programmiersprache

Die Entscheidung für eine Sprache und eine Entwicklungsumgebung für die Software war hauptsächlich davon geprägt, eine gut bedienbare, grafische Windowsanwendung zu entwickeln. Die Wahl fiel daher auf die Sprache C# aus dem .NET Framework von Microsoft[20].

Features von C# bzw. des .NET Frameworks, welche sich für diese Software gut eignen, sind unter anderem

- die gute und einfache Entwicklung von grafischen Oberflächen mittels Mircosoft Visual Studio,
- eine sehr große und aktive Entwicklergemeinde und damit verbunden eine Vielzahl verfügbarer Bibiliotheken und vorgefertigter Lösungen,
- ein sehr hilfreiches Eventsystem[21], welches das Zusammenspiel verschiedener Komponenten deutlich vereinfacht,
- eine relativ einfache Methode, Plugins zu entwickeln und in das laufende Programm einzubinden,

[20] `http://msdn.microsoft.com/de-de/netframework/default.aspx`

[21] Jede Klasse kann Events veröffentlichen, für die sich andere Objekte mit speziellen Eventhandlern registrieren, welche im Falle eines entsprechenden Ereignisses auch über Thread-Grenzen hinweg aufgerufen werden können

- die Möglichkeit gewisse Bereiche als *unsafe* zu deklarieren und dann Zugriff auf Pointer(-arithmetik) zu haben, (gerade für performante Bildverarbeitung extrem wichtig)
- sowie akzeptable Möglichkeiten, die Arbeit auf mehrere Threads aufzuteilen und damit zu beschleunigen und/oder die Anwendung reaktiv zu halten[22].

4.4 Ergebnis

4.4.1 Design

Entstanden ist ein teilmodulares Framework, welches auf Erweiterbarkeit und Flexibilität ausgerichtet ist. Die Kernkomponente stellt dabei nur einige Basisfunktionen, wie die Verwaltung der geladenen Bilder und Gruppen, den Zugriff auf die aktuelle Konfiguration sowie das Laden und Speichern derselben, die Ausführung der geladenen Filter für die Vorverarbeitung sowie die Merkmalsextraktion und einige grundlegende Statistiken über die Gruppen. Außerdem ist die Konfiguration der Suchoptionen hier integriert.

Weiterführende Statistiken und Funktionen, wie die Anzeige von einzelnen Bildern oder allen Mitgliedern einer Gruppe, die Suche mit der Ergebnisdarstellung, die Generierung globaler Featurestatistiken oder die Vorschau der Vorverarbeitung sind in einzelnen Komponenten innerhalb des Hauptprogrammes realisiert. Diese werden in der Ausführung als jeweils eigenes Fenster angezeigt, welches seine Darstellung je nach ausgewähltem Motiv bzw. Gruppe anpasst. Es wurde Wert darauf gelegt, dass sich das System relativ einfach um zusätzliche Werkzeuge erweitern lässt. Dies wurde mittels einer gemeinsamen Oberklasse realisiert, welche die Kommunikation mit dem Kern sowie das Eventhandling übernimmt.

Um eine hohe Flexibilität bei der Verwendung von Vorverarbeitungsfiltern und Merkmalsextraktoren zu erreichen, wurde ein Pluginsystem integriert, welches es ermöglicht neue Filter als dll[23] zu kompilieren und durch einfaches Kopieren in das Plugins–Unterverzeichnis in das Programm einzubinden. Hierzu wurden zwei Interfaces erstellt, die jeweils in eine eigene dll kompiliert werden und von den neuen Plugins implementiert werden müssen. Die Entwicklung findet daher ohne Eingriffe in das Hauptprogramm statt.

[22] Mit der kommenden Version 4.0 des .NET Frameworks, sollen deutlich erweiterte Techniken der Parallelisierung von Abläufen integriert werden. Siehe: `http://blogs.msdn.com/pfxteam/archive/2008/10/10/8994927.aspx`

[23] Dynamic Link Library: Programmbibiliothek

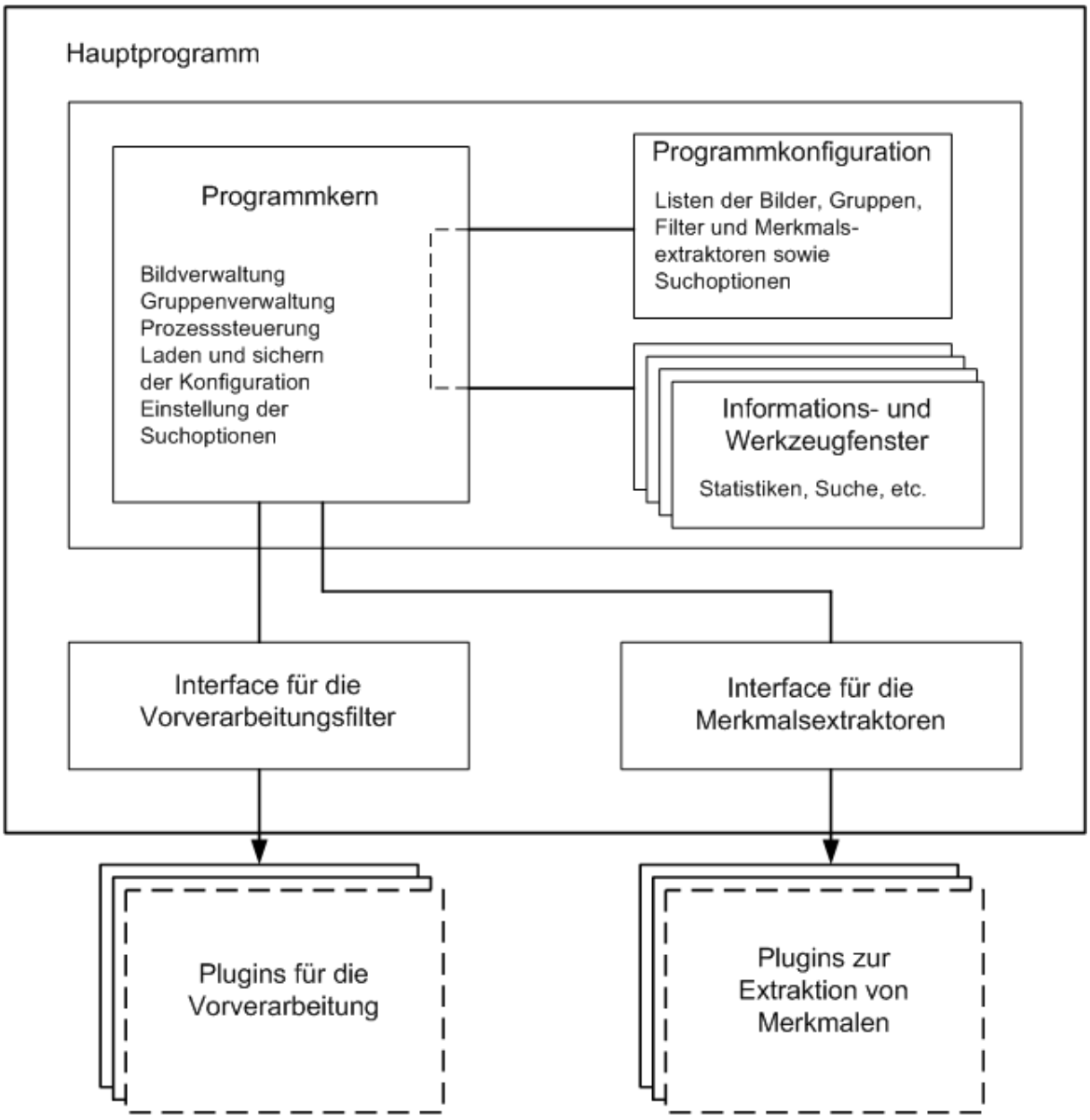


Abbildung 6: Designübersicht der fertigen Software

4.4.2 Technische Details

Damit eine möglichst große Freiheit bei der Entwicklung der Merkmale gegeben ist, besitzt jedes geladene Bild einen Merkmalsvektor, der als Dictionary von *double* Werten[24] realisiert ist, auf die über einen eindeutigen *string* zugegriffen wird. Dies ermöglicht jedem Merkmalsextraktor einen beliebig hochdimensionalen Ergebnisvektor in hoher Präzision zu speichern.

Nachteil daran ist jedoch, dass der Speicherverbrauch bei 8 Byte pro Wert und Deskriptoren mit teils über 100 Dimensionen und mehreren tausend Bildern merklich ansteigt. Da es sich jedoch zunächst nur um eine Testplattform handelt, ist die gewonnene Flexibilität es wert, diese Einschränkung in kauf zu nehmen.

4.4.3 Einschränkungen

Abgesehen von dem erhöhten Speicherbedarf ist besonders das Speichern und Laden von Programmkonfigurationen ein Schwachpunkt. Um den Aufwand überschaubar zu halten, wird die komplette Konfiguration binär serialisiert bzw. deserialisiert, was unter anderem dazu führt, dass der Ladevorgang bei sehr großen Zusammenstellungen[25] unverhältnismäßig lange dauern kann und die extrahierten Merkmale nicht von anderen Programmen ausgelesen werden können.

4.4.4 Umgesetzte Features

Entsprechend den in Kapitel 4.2 gestellten Anforderungen wurden zunächst sämtliche als *„must“* gekennzeichneten Eigenschaften implementiert.

Das Framework bietet die folgenden Basisfunktionalitäten:

- Laden von einzelnen Bildern oder ganzen Ordnerstrukturen
- Laden von Gruppenzuordnungen als CSV-Datei
- Speichern und Laden der aktuellen Konfiguration
- Hinzufügen und Konfigurieren von dynamisch als Plugin geladenen Vorverarbeitungsfiltern und Merkmalsdeskriptoren
- Funktionalität zur Ausführung der Vorverarbeitung und/oder Merkmalsextraktion der geladenen Bilder unter Nutzung beliebig vieler Prozessorkerne

[24] doubles sind in .NET 64 bit Gleitkommazahlen mit bis zu 17 signifikanten Dezimalstellen. Siehe: `http://msdn.microsoft.com/en-us/library/system.double.aspx`

[25] Konfigurationen mit mehreren hundert Megabyte

- Listendarstellung aller geladenen Bilder oder Gruppen inklusive der extrahierten Merkmale (einzeln auswählbar)

Zusätzlich sind Funktionalitäten in den folgenden Kategorien implementiert:

- **Informations- und Statistikanzeigen**
  - Ansicht der verarbeiteten Bilder
  - Vorschau der Bildvorverarbeitung
  - Anzeige aller Bilder einer Gruppe mit Statistiken über den Durchschnitt und die Varianz der Merkmale sowie der durchschnittlichen und maximalen Distanzen innerhalb der Gruppe
  - Berechnung der globalen Merkmalsstatistiken Durchschnitt, Varianz, Minimum und Maximum über alle Bilder und Visualisierung der Merkmalsverteilung als Histogramm[26]
  - Basierend auf den Mittelwerten und Varianzen der Merkmale innerhalb der Gruppen, wird der Mittelwert der Mittelwerte, die Varianz der Mittelwerte und der Mittelwert der Varianzen berechnet, sowie die Verteilung der Varianzen per Histogramm dargestellt
  - Generierung einer Korrelationsmatrix der Merkmale, in der für jede Kombination zweier Merkmale der Korrelationskoeffizient[27] berechnet wird

- **Konfiguration der Suche**
  Hier wird die Kombination der Merkmale und Deskriptoren für die Durchführung der Ähnlichkeitssuchen festgelegt. Für die Vorfilterung der Bildmenge wird eine Auswahl von Merkmalen, deren Reihenfolge der Anwendung, sowie die jeweils zulässigen Toleranzen festgelegt. Entsprechend wird für die anschließende Sortierung der Kandidaten eine Auswahl von Merkmalen und Deskriptoren getroffen, entsprechend ihrer Wertebereiche und Wichtigkeit gewichtet und eine Methode zur Entfernungsmessung[28] festgelegt.

- **Ermittlung globaler Metriken**
  Dieses Werkzeug erlaubt es, für jedes geladene Bild, entsprechend der individuellen Suchkonfiguration, automatisiert eine Suche durchzuführen. Die Ergebnisse der Einzelsuchen werden akkumuliert und sowohl numerisch als auch grafisch, in Form von

[26] Für die Darstellung aller Histogramme wurde *NPlot - A charting library for .NET* von Matt Howlett und anderen verwendet. `http://www.nplot.com/`

[27] Der Korrelationskoeffizent ist ein statistisches Maß über die lineare positive oder negative Abhängigkeit zweier Variable voneinander

[28] L_1 Norm (Manhatten Entfernung), L_2 Norm (euklidische Distanz) oder L_∞ Norm (Maximum–Nurm)

Histogrammen ausgegeben.
Die hier berechneten Metriken sind

- der ANMRR,
- die Goal Success Rate,
- die durchschnittliche Anzahl der Kandidaten nach der Vorfilterung,
- die False Exclusion Rate
- sowie die durchschnittliche Dauer eines Suchvorganges.

5 Bildvorverarbeitung

5.1 Ziele

Die Vorverarbeitung der Bilder verfolgt zwei wesentliche Ziele für das Verfahren:

1. Erhöhung der Geschwindigkeit, indem die Bilder soweit in der Größe reduziert werden, dass einerseits die Extraktion und Weiterverarbeitung weniger Rechenzeit beansprucht, aber gleichzeitig die Qualität des Bildvergleichs nicht nennenswert darunter leidet.

2. Vermeidung externer Einflüsse, die durch das Scannen der Bilder, durch ungenaues Ausschneiden oder allgemein durch Störungen wie Staub oder Bildrauschen entstehen können.

5.2 Größenanpassung

Um einen guten Kompromiss aus der Bildgröße — und damit der Geschwindigkeit — und der Effektivität der Deskriptoren zu finden, wurden für verschiedene Größen zwischen 100 und 800 Pixeln der ANMRR sowie die Goal Success Rate (siehe Kapitel 3.4.1) bestimmt. Bei der Verkleinerung kam jeweils ein bikubisches Verfahren des AForge.NET[29] Frameworks zum Einsatz, wobei die längere Bildseite unter Beibehaltung des Seitenverhältnisses auf die angegebene Pixelzahl skaliert wurde.

Die Ergebnisse (Tabelle 1 und 2 sowie Abbildung 7 und 8) zeigen, dass die Erfolgsrate ab 400 Pixel minimal und ab 200 Pixel deutlich schlechter wird. Da die Deskriptoren oft sehr unterschiedlich reagieren und z.B. farbbasierte Merkmale wesentlich unempfindlicher gegenüber Größenänderungen sind als kantenbasierte, werden alle weiteren Messungen und Versuche, soweit nicht anders angegeben, mit einer Skalierung auf 600 Pixel unter Beibehaltung der Seitenverhältnisse durchgeführt.

[29] AForge.NET ist ein C# Framework, welches weitreichende Funktionen im Bereich der Bildverarbeitung und anderem bietet. `http://code.google.com/p/aforge/`

Bei der Wahl des Verfahrens zur Interpolation standen die 3 Varianten des AForge.NET Frameworks zur Auswahl. Dieses bietet die simple *Nearest Neighbour*, die etwas anspruchsvollere *bilineare* und letztendlich als qualitativ hochwertigstes Verfahren, die *bikubische* Interpolation an. Da die Ausgangsbilder eine Kantenlänge von typischerweise maximal ca. 3500 Pixel[30] haben, welche ein Umrechnungsverhältnis von ca. 6:1 bei einer Zielgröße von 600 Pixel zur Folge hat, kann es sinnvoll sein, die Verkleinerung in mehreren Einzelschritten durchzuführen.

Selbst das bikubische Verfahren betrachtet in der Berechnung „nur“ die 4×4 umgebenden Pixel der errechneten Position des Quellbildes, was bei starker Verkleinerung (größer Faktor 4) dazu führt, dass nicht mehr alle Pixel des Ausgangsbildes zum Ergebnis der Verkleinerung beitragen. Daher wurde in den Test auch ein bikubisches 2–Schritt–Verfahren aufgenommen, das das Ausgangsbild zunächst auf 1200 Pixel und im zweiten Durchgang auf die entgültigen 600 Pixel skaliert.

Die Ergebnisse der Interpolationstests sind in Abbildung 9 (S. 30) dargestellt. Deutlich ist der qualitative Vorteil des bikubischen 2–Schritt–Verfahrens sowohl in den weniger gestörten homogenen Bereichen als auch der Darstellung der Schrift erkennbar.

Da die Qualitätsansprüche für die Weiterverarbeitung und Merkmalsextraktion den erhöhten Rechenaufwand für die Verkleinerung überwiegen, werden alle weiteren Versuche mit dem genannten 2–Schritt–Verfahren durchgeführt.

[30] Publikumszeitschriften überscheiten meist die Größe von DIN A4 ($29{,}7 \times 21{,}0cm$) nicht. Bei 300 dpi bzw. ppi (pixep per inch) Scanauflösung ergibt sich somit eine maximale Pixelhöhe von: $\frac{29{,}7cm}{2{,}54\frac{cm}{zoll}} * 300\frac{pixel}{zoll} \approx \underline{3508pixel}$

Deskriptor \ Größe	800px	600px	400px	200px	100px
CEDD	0,132	0,127	0,138	0,371	0,509
FCTH	0,152	0,154	0,156	0,159	0,184
MPEG7 CLD	0,074	0,074	0,074	0,074	0,079
MPEG7 EHD	0,068	0,062	0,077	0,142	0,461
MPEG7 SCD	0,625	0,622	0,602	0,605	0,614
Scalable Value Matrix	0,041	0,045	0,051	0,068	0,118

Tabelle 1: ANMRR in Abhängigkeit der Bildgröße

Deskriptor \ Größe	800px	600px	400px	200px	100px
CEDD	88,34%	88,84%	87,20%	64,23%	49,82%
FCTH	86,37%	86,34%	86,09%	85,87%	83,51%
MPEG7 CLD	94,38%	94,42%	94,24%	94,38%	93,71%
MPEG7 EHD	94,21%	94,71%	93,24%	86,62%	54,54%
MPEG7 SCD	38,52%	39,09%	40,52%	40,09%	39,41%
Scalable Value Matrix	96,57%	96,21%	95,64%	93,96%	88,98%

Tabelle 2: Goal Success Rate in Abhängigkeit der Bildgröße

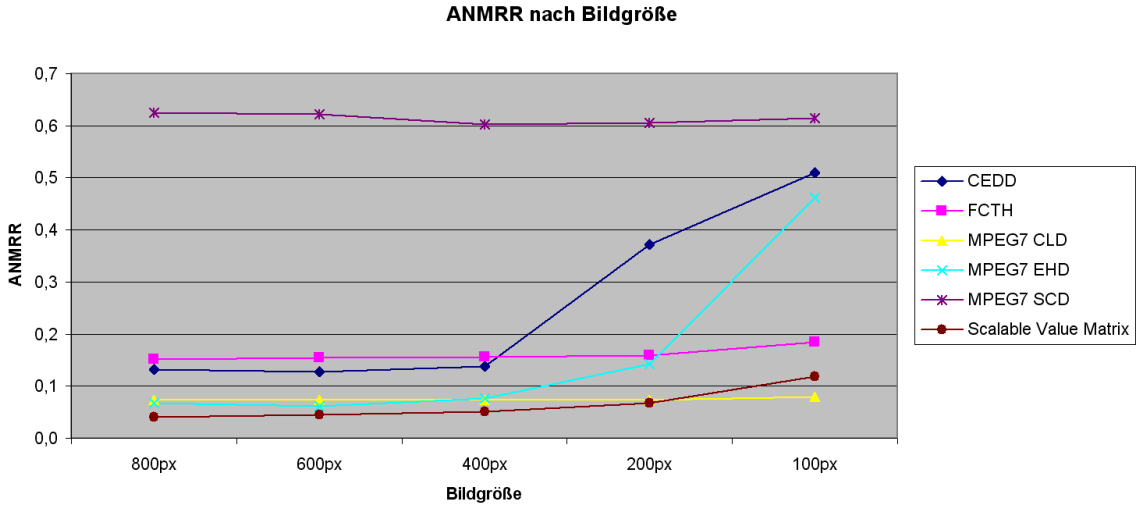


Abbildung 7: ANMRR in Abhängigkeit der Bildgröße

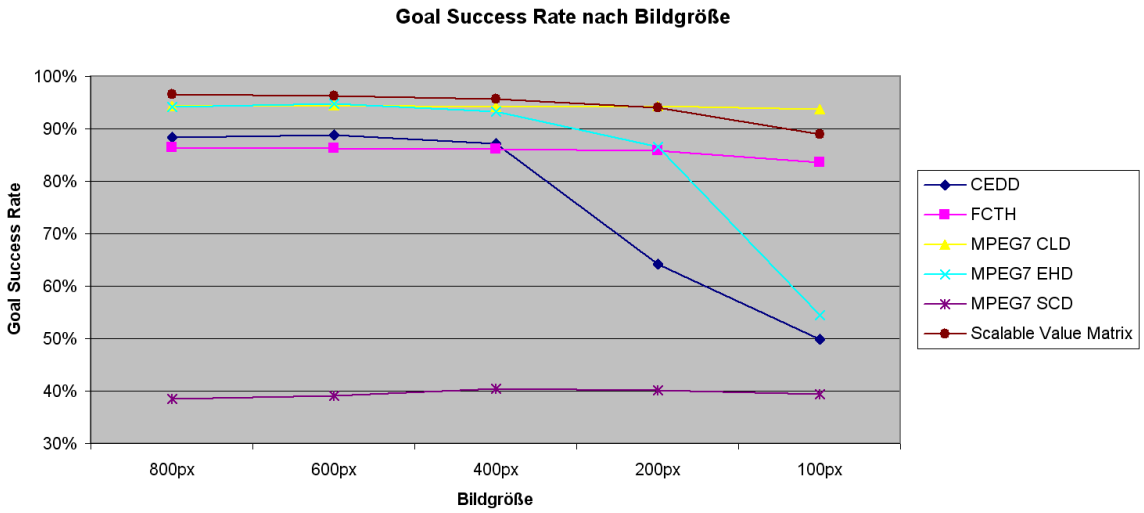


Abbildung 8: Goal Success Rate in Abhängigkeit der Bildgröße

Abbildung 9: Vergleich verschiedener Interpolationsverfahren bei der Bildverkleinerung

5.3 Filter

Eines der größten Probleme für viele Deskriptoren sind die häufig unterschiedlich breiten Ränder der Bilder, die je nach Zeitschrift oder der Person, die das Bild zuschneidet, stark variieren können.

Betroffen sind unter anderem farbbasierte Merkmale, bei denen durch den höheren Anteil an Hintergrundfarbe die Verhältnisse verschoben werden, aber besonders auch solche Merkmale, die die Bilder in Kacheln oder Ausschnitte zerlegen, die dann in Randnähe stark verfälscht werden können.

Ziel ist es, möglichst zuverlässig diejenigen Randbereiche der Bilder zu entfernen, welche keinerlei wichtige Informationen beinhalten und durch ihren variablen Anteil am Bild die Varianz der Deskriptoren innerhalb einer Gruppe erhöhen würden.

Das einfachste Verfahren, den störenden Einfluss von unpassenden Rändern zu beseitigen, ist sicherlich das simple Abschneiden eines bestimmten Anteils des Bildes. Hierzu wurde ein Crop-Filter verwendet, der das Bild horizontal und vertikal jeweils um 5% beschneidet.

Die Ergebnisse der Messungen zeigen zwar eine Verbessung im Vergleich zu den nicht beschnittenen Bildern, es wird jedoch keinerlei Entscheidung getroffen, ob ein Beschnitt überhaupt nötig gewesen wäre, noch in welchem Maß dieser zu erfolgen hätte.

Um nicht unnötigerweise Bildinformationen zu entfernen, die für zukünftige Merkmale wichtig sein könnten, wird dieser Filter im weiteren Verlauf nicht zur Randentfernung eingesetzt.

Ein allgemeiner, kontrastbasierter Ansatz um Randbereiche mit wenig Details zu entfernen, wird in (Chi Wong u. a., 2002) vorgestellt. Es wird vorgeschlagen, für jede Zeile und Spalte die Summe der absoluten Differenzen zweier benachbarter Pixel zu berechnen und in horizontaler und vertikaler Richtung jeweils die Spalten bzw. Zeilen zwischen 5% und 95% der Gesamtkonstrastsumme für die Merkmalsextraktion zu verwenden. Da in dieser Arbeit mit Farbbildern gearbeitet wird, wird für die Kontrastberechnung der Value-Kanal der HSV–Farbdarstellung verwendet.

Dieser simple aber effektive Ansatz ermöglicht zwar sehr gute Ergebnisse in der Effizienzmessung (siehe Tabelle 3), liefert aber subjektiv betrachtet sehr unschön beschnittene Bilder, da keinerlei Rücksicht auf den spezifischen Einsatzbereich genommen wird. So kann es vorkommen, dass verrauschte Ränder erhalten bleiben (Abbildung 10, S. 33) oder eigentlich relevante Bildteile mit geringem Kontrast abgeschnitten werden (Abbildung 11, S. 34).

Als Alternative wurde ein speziell auf Printmedien angepasster, parametrisierbarer Randentfernungsfilter entwickelt, welcher die Besonderheiten der Druckverfahren berück-

sichtigt. Dieser analysiert zunächst die Helligkeitsverteilung anhand des Value-Anteils der HSV-Darstellung innerhalb des Bildes und ermittelt daraus die Spannweite zwischen hellsten und dunkelsten Anteilen des Bildes. Außerdem wird der höchste Helligkeitswert als Helligkeit des Hintergrundes bzw. des Papiers angenommen.[31] Anhand dieser Spannweite und der Helligkeit des Papiers wird für jedes Bild eine individuelle *Deckungsgrenze* definiert. Als *gedeckt* wird ein Pixel dann bezeichnet, wenn seine Helligkeit abzüglich einer gewichteten Farbsättigung unter der gesetzten Grenze liegt.

Die Farbsättigung wird mit einbezogen, da in der HSV–Farbdarstellung im Gegensatz zur HSL–Darstellung ein reines Blau[32] die maximale Helligkeit 255 hat und somit zunächst als nicht *gedeckt* betrachtet werden würde.[33]

Nun wird ausgehend vom oberen Rand des Bildes ein Rechteck, das der vollen Breite des Bildes und einem Prozent der Höhe des Bildes entspricht, schrittweise nach unten geschoben. Sobald der Anteil der gedeckten Pixel innerhalb dieses Rechtecks einen eingestellten Schwellenwert (für die Messungen auf 5% eingestellt) überschreitet, wird die obere Grenze des Rechtecks als neuer oberer Rand des Bildes festgelegt. Entsprechend wird auch für die anderen drei Seiten verfahren.

Damit nicht zuviel von Bildern abgeschnitten wird, die aus Designgründen große, helle Flächen aufweisen, wird der maximale Beschnitt des Bildes auf 10% in jede Richtung begrenzt. Da immer die äußere Grenze der Rechtecke verwendet wird, bleiben dünne graue Ränder bestehen. Es bietet sich daher an, das Bild im Nachhinein noch um ca. 1% in jeder Richtung zu beschneiden (In den Messungen wird ein Beschnitt von absolut festgelegten 7 Pixeln an jedem Rand gewählt).

Probleme hat dieses Verfahren mit Störungen des gescannten Bildes, die die äußersten Randbereiche betreffen. So kann es passieren, dass im Falle eines wenige Pixel breiten, dunklen Randes, das Bild an der entsprechenden Seite nicht beschnitten wird. Um diesem Risiko vorzubeugen, fängt die Erkennnung nicht ganz außen an, sondern um 0,5% der längeren Seite nach innen versetzt. Wird kein Rand festgestellt, wird natürlich das ursprüngliche Bild erhalten.

Die deutliche Verbesserung im optischen, subjektiven Bereich als auch die hohen Erfolgsraten in den Erfolgsmetriken (siehe Tabelle 3), welche sich etwa auf einer Stufe mit denen, des simplen Kontrastbasierten Ansatzes befinden, empfehlen den Einsatz dieses Verfahrens für alle weiteren Messungen.

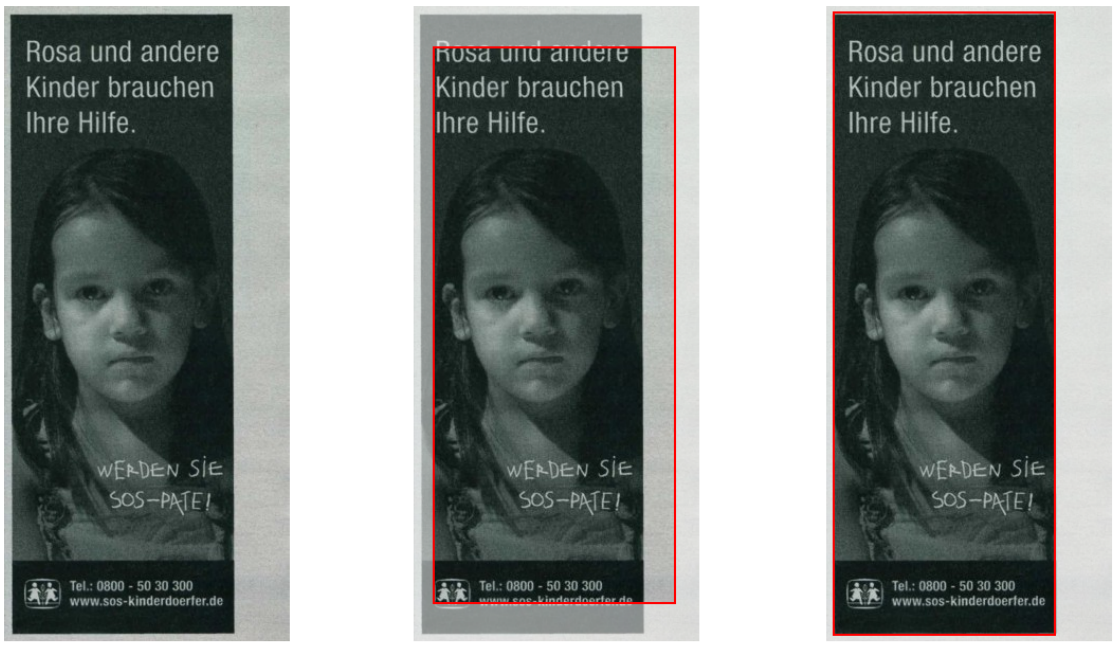
[31] Der Druck von Zeitschriften erfolgt im subtraktiven CMY(K) Verfahren, weshalb es nur möglich ist den Hintergrund abzudunkeln, aber nicht aufzuhellen.

[32] in RGB (0, 0, 255)

[33] HSL ist für diesen Zweck nicht geeignet, da sehr zarte Farben wie ein helles Grün oder Pasteltöne als voll gesättigt angesehen werden. Das Papier auf dem die Anzeigen gedruckt sind, hat jedoch häufig einen leichten Farbstich, welcher sich hier überproportional stark auf das Ergebnis auswirken würde.

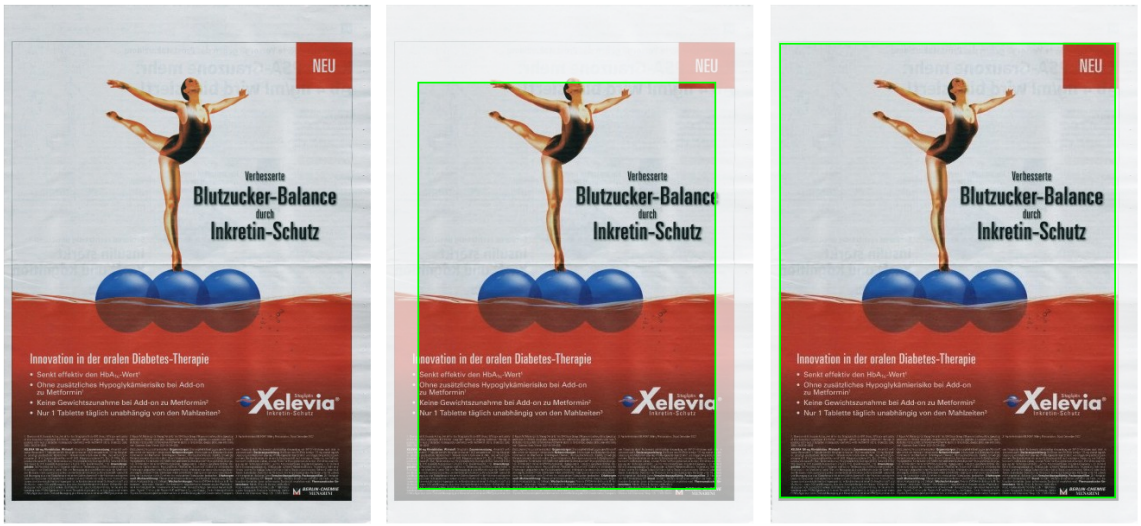
Deskriptor \Verfahren	keins	Crop um 5%	Kontrastbasiert	Deckungsbasiert
CEDD	88,84%	89,70%	89,91%	**90,41%**
FCTH	86,09%	87,55%	**87,91%**	87,55%
MPEG7 CLD	94,49%	95,42%	**96,75%**	96,17%
MPEG7 EHD	94,84%	96,28%	**97,50%**	97,00%
MPEG7 SCD	36,87%	37,80%	39,27%	**39,66%**
Scalable Value Matrix	96,21%	96,96%	97,35%	**97,82%**

Tabelle 3: Goal Success Rate für verschiedene Verfahren der Randentfernung



Der rote Rahmen markiert den Bereich des beschnittenen Bildes

Abbildung 10: Vergleich von Verfahren zur Verringerung von Einflüssen durch Ränder



Original | Kontrastbasiertes Verfahren | Deckungsbasiertes Verfahren

Der grüne Rahmen markiert den Bereich des beschnittenen Bildes

Abbildung 11: Vergleich von Verfahren zur Verringerung von Einflüssen durch Ränder

5.4 Ergebnis

Die Auswertung der Vorverarbeitung zeigt, dass eine Verkleinerung der Bilder auf etwa 600 Pixel die Ergebnisse nicht negativ beeinflusst und die Entfernung von hellen Rändern um die Motive herum eine deutliche positive Wirkung auf die Effizienz der verschiedenen Verfahren hat.

Als Reihenfolge für die Bildvorverarbeitung wird daher festgelegt:

1. Verkleinerung auf 1200 Pixel bikubisch unter Beibehaltung des Seitenverhältnisses
2. Verkleinerung auf 600 Pixel bikubisch unter Beibehaltung des Seitenverhältnisses
3. Entfernung von Bildrändern mittels selbstentwickeltem, deckungsbasiertem Verfahren
4. Leichtes Beschneiden um ca. 1% bzw. 7 Pixel absolut, um letzte Ränder zu entfernen

Auf dem Testsystem, bestehend aus einem Intel Core 2 Quad 4-Kernprozessor mit 2,4 GHz und 4GB Arbeitsspeicher, benötigen diese Schritte im Schnitt 1108ms pro Bild für das erste Testset (Kapitel 3.4.2).

Da die Schritte sequentiell und *single-threaded* ablaufen, lassen sich bis zu vier Bilder parallel verarbeiten.

6 Merkmalsdeskriptoren

6.1 Anforderungen

Das perfekte Merkmal würde alle Eigenschaften eines Bildes in eine Zahl zusammenfassen, die für ähnliche Bilder sehr geringe Schwankungen und für unähnliche Bilder große Differenzen aufweisen würde. Da allein der Begriff der Bildeigenschaften viel zu weit gefasst ist und auch die Ähnlichkeit in vielerlei Hinsicht interpretiert werden kann, ist ein solche Merkmal leider nicht verfügbar und wird es wohl auch nie sein.

Es gilt also, Merkmale und Deskriptoren zu finden, die klar definierte Eigenschaften eines Bildes beschreiben, und dahingehend die Kriterien zu erfüllen, dass die Werte der Merkmale innerhalb einer Gruppe von ähnlichen Bildern dicht beieinander liegen, bzw. die Distanzen der Deskriptoren nahe null sind. Entsprechend sollen die Distanzen für unähnliche Bilder deutlich größer sein, um eine möglichst hohe Selektivität zu erreichen.

Technisch ist es außerdem erstrebenswert, die Deskriptoren und Merkmale möglichst kompakt zu halten und die Distanzberechnung nicht zu aufwendig zu gestalten.

Im Hinblick auf das Bildmaterial dieser Arbeit, ist es zudem erforderlich, die Merkmale und Deskriptoren möglichst invariant bzw. tolerant gegenüber Helligkeitsschwankungen und Farbstichigkeit oder Farbverschiebung zu gestalten.

Da sich die Ansprüche der hohen Selektivität, Kompaktheit und Toleranz gegenseitig ausschließen, müssen für die verschiedenen Bereiche des Verfahrens auch verschiedene Merkmale und Deskriptoren entwickelt werden.

6.1.1 Vorfilterung

Für die Vorfilterung der Kandidatenmenge sollten die Deskriptoren möglichst nur aus einzelnen Merkmalen bestehen, die eine möglichst gleichmäßige Verteilung über die gesamte Bildmenge aufweisen. Die Selektivität muss für diese Merkmale nicht überragend sein. Es reicht bereits, wenn die Menge der Kandidaten mit jedem Schritt der Vorfilterung auf 20–50% reduziert werden kann. Der Fokus liegt hier vor allem auf der möglichst geringen Anzahl der Fehlausschlüsse und auf der guten Kombinierbarkeit der Merkmale.

Mit zunehmender Tiefe der Vorfilterung und damit auch geringerer Anzahl der Kandidaten darf die Berechnung der Distanzen durchaus komplexer werden, was allerdings eine spätere Indizierung wesentlich erschweren würde.

Ziel ist es, die Menge der Kandidaten auf einige Prozent der Gesamtmenge einzuschränken und dabei möglichst wenig *false-negatives* zu generieren.

6.1.2 Ranking der Ergebnisse

Die Deskriptoren für das Ranking der Ergebnisse dürfen (und müssen) wesentlich komplexer sein, da an dieser Stelle auch kleinere Abweichungen zum gesuchten Bild sich in der Distanzberechnung auswirken sollen. Es muss sichergestellt werden, dass verschiedene Eigenschaften der Bilder abgedeckt werden, was sich durch die Kombination mehrerer spezialisierter Deskriptoren realisieren lässt.

Da die Deskriptoren für das Ranking nicht indiziert werden müssen, können diese wesentlich höherdimensional ausfallen als solche für die Vorfilterung.

6.2 Generelle Bildeigenschaften

6.2.1 Seitenverhältnis

Das Seitenverhältnis eines Bildes ist das wohl einfachste Kriterium, da es keinerlei Angaben zum Inhalt des Bildes machen muss. Dies disqualifiziert es allerdings auch für den Einsatz in der Sortierung der Ergebnisse.

Wofür es sich jedoch gut eignet, ist die Vorfilterung der Kandidaten, da gleiche Anzeigen auch ein recht ähnliches Seitenverhältnis haben müssen.

Extraktion Ein primitiver Ansatz, der starr die Breite durch die Höhe teilt, führt zu Werten im Interval $(0:\infty)$ und einer Häufung der Werte um 1 herum. Da keine Rücksicht auf Quer– oder Hochformat genommen wird, kommt es dazu, dass die Verteilung stark asymmetrisch erfolgt. Alle Bilder im Hochformat würden Werte zwischen 0 und 1 erhalten, während querformatige Bilder im nach oben offenen Bereich 1–∞ liegen würden. Zusätzlich wäre der Abstand zwischen zwei Bildern des Seitenverhältnisses $8:1$ und $10:1$ größer, als der Abstand zwischen den Verhältnissen $1:1$ und $2:1$.

Um diesen Effekten entgegen zu wirken, wurde die folgende Berechnung gewählt:

$$S'(I) = \begin{cases} \frac{B(I)}{H(I)} & \text{wenn gilt: } B(I) > H(I) \\ \frac{H(I)}{B(I)} & \text{sonst} \end{cases}$$

$$S(I) = \text{sign}(B(I) - H(I)) * \ln(\ln(S'(I)) + 1)$$

Wobei $B(I)$ der Breite und $H(I)$ der Höhe eines Bildes I entspricht.

Da $S'(I)$ immer positiv und ≥ 1 ist, führt der doppelte Logarithmus zu einem steilen Anstieg der Kurve im Bereich nahe 1 und verflacht zunehmend. Für die Unterscheidung zwischen Quer– und Hochformat wird das Ergebnis mit der signum[34] Funktion der Differenz von Breite und Höhe multipliziert.

Für die bessere Lesbarkeit wird das Ergebnis abschließend auf ein Interval von $[-127 : 127]$ skaliert, wobei ein Seitenverhältnis von $10 : 1$ dem maximalen Wert entspricht. Höhere Seitenverhältnisse werden auf den Maximalwert beschränkt.

Ergebnisse Als einziges Kriterium für die Suche ist das Seitenverhältnis völlig ungeeignet. Die Erfolgsrate liegt bei nur 6,37%.

Besser sieht es bei der Verwendung als Kriterium für die Vorfilterung der Ergebnisse aus. Hier wird durch die Filterung der Kandidaten des ersten Testsets durch das Seitenverhältnis mit einer Toleranz von ± 15 eine Reduktion auf im Schnitt ca. 50% (1411,1 verbleibende Kandidaten von 2795) erreicht. Die Rate der *false-negatives* liegt hier bei 0,79%.

6.3 Farb- und helligkeitsbasierte Merkmale

Der mit Abstand größte Teil der implementierten und getesteten Deskriptoren fällt in diese Kategorie. Dies ist auch nicht weiter ungewöhnlich, besteht doch der erste Eindruck, den man von einem Bild erlangt, zumeist aus den Anteilen und der Verteilung der Farben und Helligkeiten in einem Bild und erst zweitrangig aus den Details der Oberflächenbeschaffenheit.

6.3.1 HSL und HSV Statistiken

Diese beiden Deskriptoren sind im Grunde eher lose Ansammlungen verschiedener Statistiken bezüglich der Luminanz– bzw. Valueverteilung der Bilder.

Extraktion Hierzu werden die entsprechenden Histogramme generiert und daraus die Werte Minimum, Maximum, Mittelwert, Median und die Standardabweichung berechnet. Da das AForge.NET Framework nur HSL unterstützt, wurde für die Berechnung des HSV-Value Histogramms eine eigene Klasse implementiert.[35]

Zusätzlich werden für den HSV–Raum die Anteile des Bildes bestimmt, die eine festgelegte Helligkeit unterschreiten und somit als „gedeckt“ gezählt werden.[36] Um starke Sprünge

[34] Die signum Funktion ist 1 für ein positives Argument und -1 für ein negatives.

[35] Der Code für die Umrechnung von RGB $\rightarrow$ HSV wurde diesem Artikel entnommen: `http://msdn.microsoft.com/en-us/magazine/cc164113.aspx`

[36] Siehe auch Kapitel 5.3. Jedoch ist in diesem Fall die Farbsättigung nicht verfügbar, weshalb nur der Value-Kanal genutzt wird.

der Werte durch leichte Veränderungen der Helligkeit zu vermeiden, wurde unter Verwendung der *Fuzzy Logic Library for Microsoft .Net (fuzzynet)*[37] ein unscharfer (fuzzy) Übergang zwischen gedeckt und ungedeckt gewählt.

Ergebnisse Die meisten extrahierten Merkmale haben eher informellen Wert und eignen sich nicht für die Verwendung im Suchprozess, da sie wenig Aussagekraft haben und zu starken Schwankungen unterworfen sind.

Dessen ungeachtet lassen sich die Standardabweichung und der berechnete Fuzzywert für die Deckung relativ gut für die Vorfilterung einsetzen. In Kombination lässt sich die Kandidatenmenge — bei 0,86% *false-negatives* — auf etwa 22% reduzieren.

6.3.2 Fuzzy–Farbhistogramm

Als Vertreter der großen Gruppe der Farbhistogramme wurde nach Vorbild von (Chatzichristofis und Boutalis, 2008a) ein 10–Bin fuzzy Farbhistogramm implementiert. Wie alle Farbhistogramme enthält es keinerlei Informationen über die räumliche Verteilung der Farben, sondern teilt die Farben des Bildes in die Kategorien Schwarz, Grau, Weiß, Rot, Orange, Gelb, Grün, Cyan, Blau und Magenta ein. Damit leichte Farb- und Helligkeitsabweichungen nicht sofort zu einer komplett anderen Klassifikation führen, wurden alle Übergänge zwischen den Klassen mittels fuzzy Zugehörigkeitsfunktionen fließend gestaltet.

Extraktion Zunächst wird wieder die „Helligkeitsspannweite“ (siehe Kapitel 5.3) berechnet, um die Grenzen für die Zuordnung zu Weiß, Grau und Schwarz dem Dynamikumfang des Bildes anzupassen. Es werden 13 fuzzy Regeln aufgestellt, die die Helligkeit (3 Regeln), die Farbsättigung (2 Regeln) sowie die Farbzugehörigkeit (8 Regeln) beschreiben.

Anhand der Regeln wird nun jeder Bildpunkt einem oder mehreren Feldern des Histogramms zugeordnet, wobei die Summe der Zugehörigkeiten immer gleich 1 ist. Abschließend wird das gefüllte Histogramm noch auf eine Summe von 255 normiert, um den Deskriptor invariant in Bezug auf die Bildgröße und damit die Gesamtpixelzahl zu machen.

Ergebnisse Selbst durch eine starke Verbreiterung der fuzzy Übergänge zwischen den Farben und eine Anpassung der Distanzberechnung, so dass auch benachbarte Farben mit einbezogen werden, lassen sich die Varianzen innerhalb der Gruppen nicht auf eine annehmbare Größe bringen. Bedingt ist dies durch die teilweise recht großen farblichen Verschiebungen, besonders bei den Übergängen zwischen Schwarz und sehr dunklen Farben. Zwei typische Problemfälle sind in Abbildung 12 zu sehen.

[37] fuzzynet ist eine quelloffene Bibliothek, welche Fuzzy–Funktionalitäten bereitstellt. `http://fuzzynet.sourceforge.net/`

Aber auch Schwankungen zwischen sehr hellen, zarten Farben (Abbildung 13) können zu starken Abweichungen der Histogramme führen.

Maximal lässt sich eine Goal Success Rate von 65,7% bei einer ANMRR von 0,369 erreichen. Die Sortierung der 2795 Kandidaten dauert dabei im Schnitt 10ms.

15193_00001194 15247_00000113 15224_00000089 15242_00000441

Abbildung 12: Typische Problemfälle für ein Farbhistogramm

5460_00000169 5462_00000002

Abbildung 13: Problematische Farbabweichungen in hellen und blassen Bereichen

6.3.3 MPEG–7 Scalable Color Descriptor

Wie der Name bereits andeutet, handelt es sich um ein skalierbares Farbhistogramm in HSV–Darstellung. Anpassbar ist sowohl die Größe des Histogramms (16, 32, 64, 128 und 256 Bins) als auch die Anzahl der Bits, die für die Speicherung der Werte verwendet werden. Der besondere Vorteil ist, dass sich ein einmal extrahierter Merkmalsvektor auch nachträglich noch auf eine geringere Größe herunterrechnen lässt, was auch Vergleiche zwischen unterschiedlich detailierten SCD–Histogrammen ermöglicht (Salembier und Sikora, 2002).

Extraktion Die Implementierung dieses und der anderen in dieser Arbeit verwendeten MPEG–7 Deskriptoren entstammt ursprünglich dem LIRe–Projekt[38] (Mathias Lux, 2008) und ist als C# Bibliothek verfügbar[39].

Ergebnisse Da der Deskriptor die Verwendung verschiedener Histogrammgrößen zulässt, wurden die Ergebnisse für die fünf oben genannten Größen ermittelt und in Tabelle 4 dargestellt. Wie zu erwarten, steigt die Rechenzeit für die Distanzberechnung linear mit der Anzahl der Histogrammfelder, während die Steigerung der Erfolgsraten nach den großen Sprüngen zu Beginn merklich abflacht.

Insgesamt hat der Scalable Color Descriptor dieselben Probleme wie das Fuzzy Color Histogramm (Kapitel 6.3.2), erreicht aber erst ab einer Größe von 128 Bins ähnliche Erfolgsraten und braucht dafür im Schnitt auch gut 8 mal so viel Zeit für die Distanzberechnung und Sortierung.

Histogrammgröße	16 Bins	32 Bins	64 Bins	128 Bins	256 Bins
ANMRR	0,906	0,777	0,612	0,325	0,226
Goal Success Rate	8,73%	22,42%	39,66%	69,16%	78,22%
Ø Suchdauer	14,4ms	24,1ms	43,9ms	83,6ms	160,2ms

Tabelle 4: MPEG–7 Scalable Color Descriptor Ergebnisse

6.3.4 MPEG–7 Color Layout Descriptor

Der MPEG–7 Color Layout Descriptor (CLD) bietet eine sehr kompakte Repräsentation der Helligkeits- und Farbverteilung innerhalb eines Bildes. In der gebräuchlichen Form mit 12 Koeffizienten benötigt der Deskriptor nur 63 Bit zuzüglich eventuell nötiger Metainformationen, wenn innerhalb des Systems verschiedene Größen des CLD verwendet werden. Dies führt zu einer einfachen und extrem schnellen Distanzberechnung[40], die sich hervorragend für die Arbeit mit großen Datenmenge eignet (Salembier und Sikora, 2002).

Extraktion Wie bereits beim SCD (Kapitel 6.3.3) wird auch hier die C# Bibliothek auf Basis des LIRe–Projekts benutzt. Im ersten Schritt erfolgt eine Umwandlung des Bildes in den YCbCr–Farbraum und eine Verkleinerung auf 8×8 Pixel, indem die Durchschnittswerte der

[38] Website: `http://www.semanticmetadata.net/lire/`

[39] Die Bibliotheken mit Beispielen für die Verwendung können sind hier zu finden: `http://savvash.blogspot.com/2007/10/here-acm-multimedia-2007.html`

[40] Laut (Salembier und Sikora, 2002) werden für eine Distanzberechnung unter Verwendung von Intels SSE Instruktionen nur 110 CPU-Takte benötigt

drei Kanäle Y, Cb und Cr gebildet werden. Für jeden Kanal des 8×8 Bildes wird nun eine diskrete Kosinustransformation (DCT) durchgeführt, wodurch pro Kanal 64 Koeffizienten herauskommen. Je nach Einstellung des Deskriptors bezüglich der Koeffizienten pro Kanal werden die ersten $x \in \{3, 6, 10, 15, 21, 28, 64\}$ per ZickZack-Muster ausgehend von der oberen, linken Ecke ausgewählt.[41] Abschließend werden die Koeffizienten noch auf 6 (für den ersten) bzw. 5 (alle anderen) Bit quantisiert, um den Speicherbedarf gering zu halten.

Typischerweise werden für den Y–Kanal (Luminanz–Anteil) 6 Koeffizienten und für die beiden Cb– und Cr–Kanäle (Chroma–Anteil) jeweils 3 Koeffizieten extrahiert. Für gesteigerte Genauigkeit kann die Zahl der Koeffizienten weiter erhöht werden.

Ergebnisse Der CLD erreicht bereits in der Standardkonfiguration sehr gute Ergebnisse mit über 96% Erfolgsrate bei einer Suchdauer von nur etwa 9ms (Tabelle 5). Auch für die Vorfilterung der Kandidaten ist dieser Filter gut einsetzbar und reduziert die Menge der Kandidaten auf nur knapp 2,7% der Ausgangsgröße.

Dadurch, dass in der Distanzberechnung der jeweils erste Koeffizient eines Kanals (der Gleichanteil der DCT) höher gewichtet wird als die folgenden Koeffizienten, die die räumliche Verteilung angeben, ist dieser Deskriptor empfindlich gegenüber Helligkeitsschwankungen oder Farbstichigkeiten. (Abbildung 14)

15230_00000759 15265_00000085

Abbildung 14: Problematische Helligkeitsunterschiede für den CLD

In dieser Bündelung der Gleichanteile in jeweils einen Koeffizienten pro Kanal, liegt aber auch einer der großen Vorteile dieses Verfahrens. Durch simples Weglassen des jeweils ersten Koeffizienten kann der Einfluss der Grundhelligkeit bzw. eines Farbstiches fast vollständig eliminiert werden, weshalb eine entsprechende Version der Distanzberechnung implementiert wurde, die die Gleichanteile mit 0 gewichtet.

Die Ergebnisse dieser Anpassung (Tabelle 5) zeigen, dass sich die Erfolgsrate hierdurch noch weiter steigert, obwohl nur noch 9 der 12 extrahierten Koeffizienten benutzt werden.

[41] Die Informationen über die niedrigen Frequenzanteile finden sich nahe des Ursprungs (oben, links). Diese sind für den Bildeindruck zunächst wichtiger als die höherfrequenten, welche sich weiter unten, rechts befinden.

Leider wird zeitgleich die Eignung für die Vorfilterung eingeschränkt, da mit der Grundhelligkeit und den Grundfarbanteilen drei sehr selektive Merkmale weggefallen sind. Die Zeitunterschiede lassen sich durch die flexiblere Implementierung des individuell gewichteten Deskriptors erklären. Der Rechenaufwand würde in einer optimierten Umgebung sogar sinken.

Eine Erhöhung der Zahl der Koeffizienten bringt eine weitere deutliche Verbesserung der Ergebnisse bei nur leicht erhöhtem Rechenaufwand mit sich. Auch die Tauglichkeit für die Vorfilterung steigt mit zunehmender Zahl der Koeffizienten wieder an und erreicht bereits in der $6:6:6$ (Y:Cb:Cr) Konfiguration fast das Niveau des original CLD und überholt ihn ab $10:6:6$ Koeffizienten (Tabelle 5). Da mit zunehmender Koeffizientenzahl jedoch auch die Toleranz der Filterung steigen muss, ist eine beliebig hohe Anzahl Koeffizienten für die Vorfilterung nicht immer zuträglich (siehe Ergebnisse für die $15:10:10$ Konfiguration).

Koeff. Konfiguration	6:3:3	6:3:3 CW	6:6:6 CW	10:6:6 CW	15:10:10 CW
ANMRR	0,054	0,044	0,019	0,009	0,005
Goal Success Rate	96,17%	97,25%	98,96%	99,46%	99,79%
Ø Suchdauer	9ms	16,3ms	20,6ms	23,5ms	33,1ms
Vorfilterratio	2,66%	4,29%	3,10%	2,26%	2,93%
False Exclusion Rate	0,57%	0,64%	0,50%	0,50%	0,36%

Tabelle 5: MPEG–7 Color Layout Descriptor Ergebnisse (CW = Custom Weights)

6.3.5 Geometrische Momente

Die geometrischen Momente werden eigentlich eher den formbasierten Merkmalen zugeordnet. Da jedoch keine Segmentierung oder Extraktion einzelner Formen vorgenommen wird, beschreiben die Momente in dieser Arbeit die Helligkeitsverteilung des Bildes.

Extraktion Es wurden die normalisierten Zentralmomente und darauf basierend die Hu–Momente (Hu, 1962) nach den Rechenvorschriften in (Gonzalez und Woods, 2002, 673–675) implementiert.

Es werden zunächst alle Momente bis zur dritten Ordnung berechnet, anschließend daraus die Zentralmomente gebildet, wodurch Translationsinvarianz erreicht wird. Um zusätzlich Invarianz bezüglich der Skalierung zu erhalten, werden die Zentralmomente mit der Summe aller Pixel unter Einbeziehung der Ordnung normalisiert. Im Fall der Hu–Momente werden durch Kombination der normalisierten Zentralmomente 7 rotationsinvariante Momente erzeugt.

Ergebnisse Mit Werten von ca. 35% GSR für die Hu–Momente und 58,4% für die normalisierten Zentralmomente eignen sich beide Verfahren nicht sonderlich gut für eine Sortierung der Ergebnisse.

Da die Werte der einzelnen Momente zudem stark miteinander korrelieren, wie Abbildung 15 zu entnehmen ist, sind sie zudem nur eingeschränkt für die Vorfilterung einsetzbar. Unter Verwendung der zwei normalisierten Zentralmomente η_{20} und η_{02} lässt sich zumindest eine Reduktion auf im Schnitt 22,8% bei 0,57% False Exclusion Rate erreichen.

	NCM_03	NCM_30	NCM_12	NCM_21	NCM_02	NCM_20	NCM_11	HuMoment_7	HuMoment_6	HuMoment_5	HuMoment_4	HuMoment_3	HuMoment_2
HuMoment_1	0,513	0,253	0,49	0,408	0,672	0,29	0,355	0,705	0,784	0,735	0,73	0,748	0,825
HuMoment_2	0,385	-0,023	0,287	0,129	0,585	-0,068	0,152	0,477	0,717	0,594	0,577	0,599	
HuMoment_3	0,711	0,251	0,528	0,389	0,561	0,186	0,391	0,876	0,864	0,922	0,881		
HuMoment_4	0,779	0,274	0,489	0,375	0,551	0,181	0,363	0,943	0,962	0,983			
HuMoment_5	0,774	0,23	0,481	0,363	0,559	0,159	0,354	0,919	0,959				
HuMoment_6	0,736	0,167	0,434	0,318	0,59	0,1	0,306	0,874					
HuMoment_7	0,756	0,457	0,586	0,483	0,527	0,326	0,47						
NCM_11	0,256	0,447	0,467	0,43	0,235	0,377							
NCM_20	-0,118	0,723	0,269	0,544	-0,298								
NCM_02	0,707	-0,141	0,44	0,171									
NCM_21	0,226	0,469	0,372										
NCM_12	0,397	0,376											
NCM_30	-0,03												

Abbildung 15: Korrelationsmatrix der invarianten Momente

6.3.6 Scalable Value Matrix

Der Scalable–Value–Matrix–Deskriptor ist ein von mir selbstentwickeltes, mehrstufiges Verfahren, um die Verteilung von hellen und dunklen Bereichen eines Bildes zu erfassen. Die Darstellung ist extrem kompakt, da die Grundhelligkeit einmal in 3 Bit quantisiert wird und alle Abweichungen mit nur drei Zuständen und somit maximal 2 Bit beschrieben werden. Der Deskriptor für eine Zerlegung in 64 Einzelblöcke inklusive der darüber liegenden Ebenen, benötigt somit maximal 22 Byte Speicher und bietet ein hohes Maß an Details.

Extraktion Zunächst wird die durchschnittliche Helligkeit und die Helligkeitsspannweite des Gesamtbildes bestimmt. Dieses stellt zugleich die Ebene 0 des Verfahrens dar. Auf Ebene 1 wird jeder Block der vorherigen Ebene 0 (für die Ebene 0 ist das nur einer) in 4 Teilblöcke zerlegt und für jeden die durchschnittliche Helligkeit berechnet. Anhand der Differenz zwischen Helligkeit des Teilblocks zum Block der darüberliegenden Ebene, wird unter Berücksichtigung einer Toleranz t (hier $t = 8$) festgelegt, ob der Teilblock heller ($= 1$), gleich hell ($= 0$) oder dunkler ($= -1$) ist.

Durch die weitere Zerlegung der so entstandenen neuen Blöcke entstehen weitere Ebenen, von denen jede 4 mal soviele Blöcke wie die darüberliegende hat (Abbildung 16).

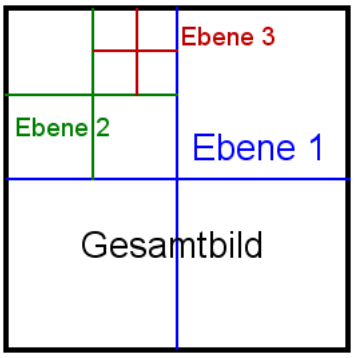


Abbildung 16: Zerlegung des Bildes in die Blöcke der Ebenen

Ergebnisse Für die Ermittlung der Ergebnisse wird eine angepasste Variante eingesetzt, die jeweils die Kombination von 4 Werten (-1, 0, 1) zu einem Wert (0–80) zusammenfasst und für die Distanzberechnung auf vorgenerierte Indextabellen zurückgreift, die für jede Kombination zweier Werte die Differenz bereithalten.

An den Ergebnissen in Tabelle 6 ist deutlich zu sehen, dass nur 5 Werte, die in diesem Fall aus 1×3 Bit und 4×2 Bit bestehen, nicht ausreichen können[42], um ein Bild hinreichend genau zu beschreiben und tolerant gegenüber kleinen Schwankungen in der Helligkeit zu sein. Ab der zweiten und dritten Ebene, und damit insgesamt 21 bzw. 85 Parametern, steigt die Genauigkeit deutlich an, und auch die Geschwindigkeit hält sich in einem guten Bereich.

Eine Besonderheit dieses Verfahrens ist, dass es theoretisch eine hierarchische Distanzberechnung unterstützt. Hierbei kann bereits beim (schnellen) Vergleich der oberen Ebenen mit wenigen Parametern entschieden werden, dass die Unterschiede zu groß sind und eine weitere Berechnung der detaillierteren, tiefen Ebenen nicht mehr nötig ist. Für diese Arbeit ist ein solches Verhalten jedoch nicht implementiert worden.

Probleme Durch die starre Aufteilung des Bildes in Blöcke und deren festen Bezug zu dem in der höheren Ebene darüber liegenden Block, reagiert der Deskriptor sehr empfindlich auf die Verschiebung von einzelnen Teilen des Bildes (Abbildung 17). Dieses Verhalten lässt sich auch nicht einfach unter Kontrolle bringen, sondern nur in der Gewichtung der Distanzen auf den verschiedenen Detailebenen berücksichtigen.

[42] Bei diese 5 Werten kann es überhaupt nur $8 * 3^4 = \underline{648}$ Kombinationen geben, von denen sich viele kaum unterscheiden.

Grad der Zerlegung	1 Ebene	2 Ebenen	3 Ebenen
ANMRR	0,774	0,138	0,028
Goal Success Rate	22,68%	88,70%	97,82%
Ø Suchdauer	5,4ms	8,5ms	25,4ms
Vorfilterratio	24,46%	7,30%	3,57%
Avg False Exclusion Rate	0,72%	0,72%	0,57%

Tabelle 6: Scalable Value Matrix Ergebnisse)

15222_00000695 15304_00000145 15256_00000822 15292_00000099

Abbildung 17: Probleme durch verschobene Elemente

6.4 Texturbasierte Merkmale

6.4.1 Tamura Textur

Die von Tamura in (Tamura u. a., 1978) vorgestellten Merkmale ermöglichen es, bestimmte Eigenschaften von Oberflächen oder Mustern in Zahlen auszudrücken. Wie in Kapitel 2.3.2 erwähnt, werden besonders die ersten drei Features (*coarseness*, *contrast* und *directionality*) in CBIR–Systemen eingesetzt.

Extraktion Es wurde auf eine von Java nach C# portierte Klasse aus dem LIRe–Projekt zurückgegriffen, die jedoch offenbar einige Fehler in der Implementierung des *coarseness*–Merkmals aufwies. Diese wurden entsprechend der Beschreibung aus (Tamura u. a., 1978) von mir korrigiert.

Ergebnisse Die Goal Success Raten für die *coarseness* und den *contrast* befinden sich im unteren, einstelligen Prozentbereich, weshalb diese für die Verwendung in der Sortierung nicht geeignet sind.

Auch die Suche mittels des *directionality*–Histogramms führt in nur knapp 48% der Fälle zum gewünschten Ergebnis und rangiert damit im hinteren Mittelfeld.

Erschwerend kommt noch hinzu, dass die Berechnung des Deskriptors, bedingt durch die *coarseness*, sehr aufwendig ist und selbst auf einem stark verkleinerten Bild[43] noch über 600ms benötigt.

6.4.2 MPEG–7 Edge Histrogram Descriptor

Das MPEG–7 Edge Histogram (EHD) ist ein 80–Bin Histogramm, welches die lokale Verteilung von Kanten, mit und ohne dominante Richtung, innerhalb eines Bildes darstellt. Um Speicherplatz zu sparen und eine kompakte Darstellung zu ermöglichen, werden sämtliche Werte des Histogramms in 3 Bit dargestellt, was zu einer Gesamtgröße von $80 \times 3\text{Bit} = 240\text{Bit}$ führt.

Extraktion Wie die anderen verwendeten MPEG–7 Deskriptoren, ist auch dieser eine C# Implementierung auf Basis des LIRe–Projekts.

Das Bild wird zunächst in 4×4 Teilbilder zerlegt, für die jeweils ein Histogramm mit 5 Bins für die verschiedenen Kantenrichtungen erstellt wird. Für die Berechnung der Kantenrichtungen wird das gesamte Bild in ca. 1100 Blöcke eingeteilt und für jeden Block getestet, ob er eine Kante beinhaltet und wenn ja, ob diese eine dominante Richtung aufweist. Unterschieden werden: Kanten ohne bestimmte Richtung, horizontale Kanten, vertikale Kanten, 45° diagonale Kanten und 135° diagonale Kanten. Entsprechend der Aufteilung der Blöcke auf die Teilbilder, werden die Histogramme der Teilbilder mit der Anzahl der dem jeweiligen Kantentyp entsprechenden Blöcke gefüllt und mit der Gesamtzahl an Blöcken innerhalb des Teilbildes normalisiert. Die resultierenden 5er Histogramme haben daher immer eine Summe ≤ 1. Eine Summe die kleiner als 1 ist, zeigt implizit, dass es homogene Blöcke ohne Kanten innerhalb des Teilbildes gibt.

Abschließend werden die Histogrammwerte noch nichtlinear[44] auf 3 Bit quantisiert. Für das 80–Bin Ergebnishistogramm werden die 16 Histogramme der Teilbilder konkateniert.

Ergebnisse Mit 98,14% Goal Success Rate ist der EHD ein sehr guter Deskriptor für die Sortierung der Kandidaten. Für die Vorfilterung ist er kaum geeignet, da er als 80 dimensionaler Merkmalsvektor relativ aufwendig in der Distanzberechnung ist, und vor allem die Distanzen sehr stark von der Homogenität des jeweiligen Bildes abhängen.

Probleme gibt es durch das festgelegte 4×4 Raster, wie auch bei der Scalable Value

[43] Die Bilder werden für die Tamura Merkmalsextraktion auf 128 Pixel in der längeren Seite verkleinert.

[44] Die Werte liegen üblicherweise unter $0,3$, weshalb man bei einer linearen Quantisierung, im Bereich unter $0,5$ auf Präzision verzichten würde.

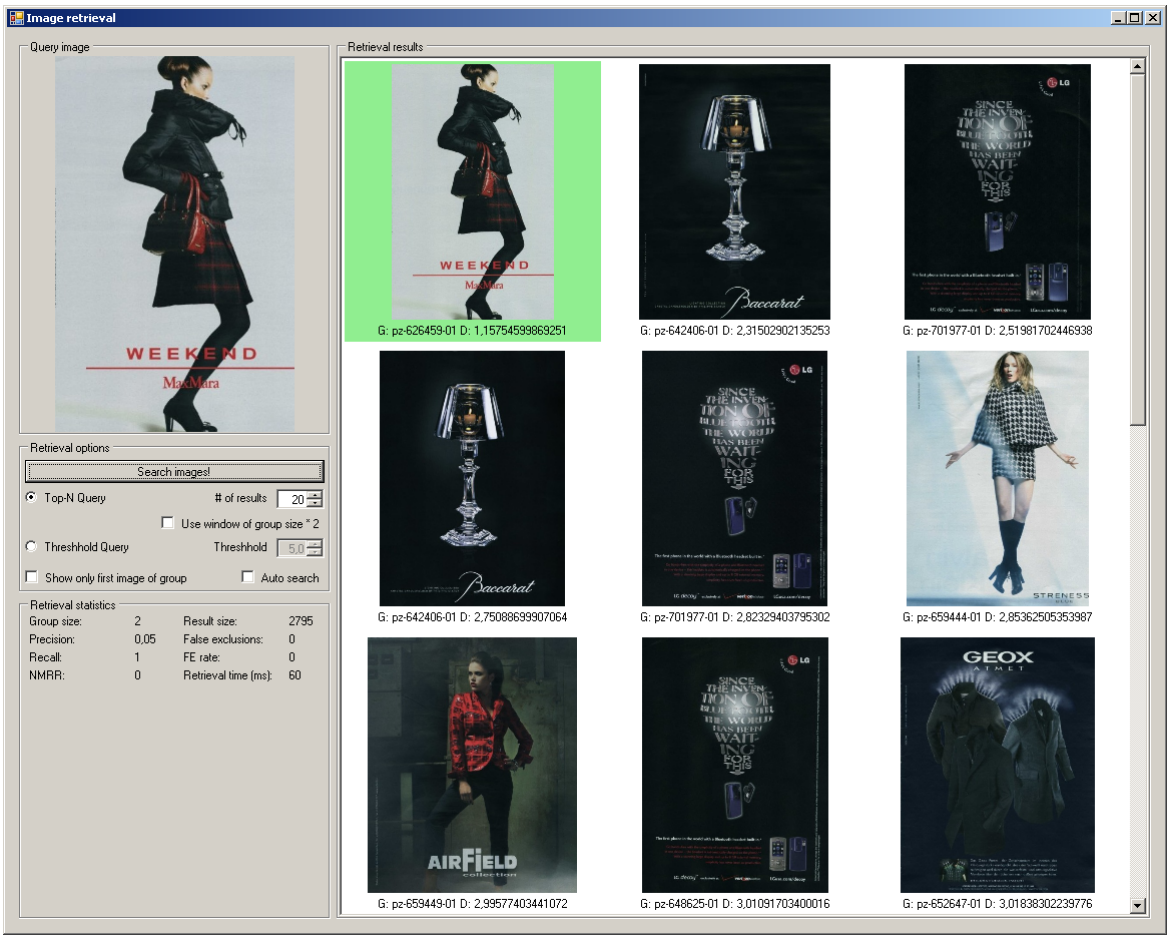


Abbildung 18: Ergebnis einer Suche mittels MPEG–7 EHD

Matrix (Kapitel 6.3.6), besonders bei verschobenen Objekten innerhalb des Bildes, welche dadurch in einem anderen Teilbild liegen.

Einen Eindruck von der Arbeitsweise des Deskriptors erhält man in Abbildung 18, wo man gut die Anordnung von Bereichen mit vielen, gerichteten Kanten und sehr homogenen Teilbildern der 4×4 Einteilung der Bilder erkennen kann.

Anmerkung: Durch einen Fehler in der Referenzimplementierung der Distanzberechnung des Deskriptors aus dem LIRe–Projekt, der mir erst zu diesem Zeitpunkt auffiel, fällt das Ergebnis hier besser aus als bei der Evaluierung der Vorverarbeitungsschritte!

6.5 Kombinierte Deskriptoren

6.5.1 FCTH und CEDD

Das Fuzzy Color and Texture Histogram (FCTH) (Chatzichristofis und Boutalis, 2008a) und der Color and Edge Directivity Descriptor (CEDD) (Chatzichristofis und Boutalis, 2008b) sind zwei Deskriptoren, welche Farbinformationen mit Texturbeschreibungen kombinieren.

Die Farbinformationen werden in beiden Deskriptoren durch ein 24–Bin fuzzy Farb Histogramm dargestellt. Die Texturinformationen des FCTH basieren auf einer Haar–Wavelet Transformation, während der CEDD für die Kantenklassifikation auf den MPEG–7 EHD (Kapitel 6.4.2) zurückgreift. Aufgrund der Kombination ergeben sich relativ große Histogramme mit 192 Feldern für den FCTH und 144 für den CEDD.

Extraktion Beide Deskriptoren beginnen mit der Zerlegung des Bildes in 1600 Blöcke. Zunächst wird für jeden Block anhand eines zweistufigen fuzzy Systems die Farbeinordnung in das 24–Bin Farbhistogramm bestimmt.

Das FCTH zerlegt jeden Block mittels einer Haar-Wavelet Transformation in 4 Frequenzbänder. Mittels Momenten zweiter Ordnung der Koeffizienten dieser Frequenzbänder wird der Block von einem dritten fuzzy System in eine von 8 Klassen eingeordnet. Abschließend wird der Block in das Ergebnishistogramm an der Stelle S eingeordnet, wobei $S = 24 * (\text{Index der Texturklassifikation}) + (\text{Index des Farbhistogramms})$.

Der CEDD benutzt für jeden Block das Verfahren des MPEG–7 EHD, mit dem Unterschied, dass zusätzlich zu den 5 Kantenarten auch noch der Fall „keine Kante vorhanden“ explizit erfasst wird, wodurch ein 6–Bin Histogramm gebildet wird. Ein Block kann dabei auch zu mehreren Feldern des Histogramms beitragen, wenn die Kantenausprägung in mehrere Richtungen bestimmte Schwellwerte überschreitet. Unter Verwendung des gleichen Verfahrens wie beim FCTH wird der Block entsprechend seiner Zuordnung(en) in Farbe und Textur auf ein oder mehrere Felder des Ergebnishistogramms abgebildet.

Ergebnisse Die Ergebnisse liegen mit 87,4% für das FCTH und 88,5% für den CEDD im guten Bereich. Die Suche ist mit durchschnittlich 300ms (CEDD) bis 390ms (FCTH) jedoch sehr rechenaufwendig.

Da das Farbhistogramm, das die beiden Verfahren nutzen, auf dem 10–Bin fuzzy Histogramm basiert, welches auch Vorlage für das Fuzzy Farb Histogramm (Kapitel 6.3.2) war, sind die Probleme mit verschiedenen Helligkeiten und Farbverschiebungen dieselben.

6.6 Bild-zu-Bild Vergleichsverfahren

6.6.1 Image Distortion Model

Das Image Distortion Model (IDM) ist kein Deskriptor, sondern vielmehr ein Verfahren, das es ermöglicht, zwei Bilder direkt miteinander zu vergleichen und dabei leichte Pixelverschiebungen zu tolerieren (Keysers u. a., 2004).

Extraktion Das Bild wird zunächst in mehreren Schritten auf 32×32 Pixel verkleinert, bevor die Durchschnittshelligkeit des Bildes und für jeden Pixel der HSV–Value Wert bestimmt wird. Um invariant gegenüber unterschiedlichen Grundhelligkeiten zu sein, wird in dem 32×32 Byte–Array nur die Abweichung zwischen Durchschnitt und Pixelwert plus 127 gespeichert[45].

Vergleich Um die Distanz zweier Bilder unter Verwendung ihrer jeweiligen 32×32 Byte–Vektoren zu bestimmen, wird über alle Elemente des Quellbildes die Summe der minimalen Differenzen berechnet. Die minimale Differenz für einen Punkt wird ermittelt, indem der korrespondierende Punkt im Zielbild bestimmt wird[46] und für ihn sowie für seine 3×3 Nachbarschaft die Differenzen zum aktuellen Pixel des Quellbildes gebildet werden. Die kleinste absolute Differenz hiervon wird quadriert und auf die Differenzsumme des Bildes addiert. Das Ergebnis ist dann die Quadratwurzel der Gesamtdifferenzsumme.

Zu beachten ist, dass die Ergebnisse nicht symmetrisch sein müssen! Der Vergleich von Bild A zu B kann eine andere Distanz liefern als der Vergleich von B zu A.

Ergebnisse Mit einer Erfolgsrate von 99,39% ist das IDM das zweiteffektivste Verfahren nach der angepassten Variante des MPEG–7 CLD (Kapitel 6.3.4). Mit im Schnitt ca. 250ms pro Suche zählt es jedoch auch zu den langsameren Vertretern. Höhere Raten ließen sich vermutlich durch eine Vergrößerung der betrachteten Nachbarschaft auf 5×5 oder gar 7×7 und Einführung einer Gewichtung der Differenzen nach Entfernung zum Ursprungspixel erzielen. Auch eine Erhöhung der Auflösung auf 64×64 Pixel könnte eine Verbesserung bewirken. Leider steigt der Rechenaufwand mit jeder dieser Anpassungen quadratisch an, was das ohnehin schon aufwendige Verfahren unpraktikabel machen würde.

[45] Da Byte–Werte vorzeichenlos sind, muss der Nullpunkt der Verteilung auf die Mitte des Wertebereichs verschoben werden

[46] Im Falle gleich großer Bilder können dieselben Indizes benutzt werden. Falls das Seitenverhältnis beibehalten wurde, muss hier interpoliert werden.

6.7 Übersicht der Suchergebnisse

In Tabelle 7 werden alle Ergebnisse der Deskriptoren zusammengefasst, die auf dem ersten Testset (Siehe 3.4.2) eine Goal Success Rate von mindestens 75% erreicht haben.

	MPEG–7			
Deskriptor	SCD (256 Bin)	CLD	CLD (CW) (15:10:10)	EHD
ANMRR	0,229	0,054	0,005	0,024
Goal Success Rate	78,22%	96,17%	99,79%	98,14%
Ø Suchdauer	160,2ms	9ms	33,1ms	58,5ms

	Composite Descriptors			
Deskriptor	FCTH	CEDD	SVM (3 Ebenen)	IDM
ANMRR	0,142	0,127	0,028	0,009
Goal Success Rate	87,37%	88,52%	97,82%	99,39%
Ø Suchdauer	390,5ms	306,1ms	25,4ms	250,4ms

Tabelle 7: Übersicht der Ergebnisse aller Deskriptoren über 75%

7 Verfahren

7.1 Auswahl der Deskriptoren

Grundsätzlich wurden in Kapitel 6 alle Deskriptoren und Merkmale auf ihre Eignung sowohl für die Sortierung als auch die Vorfilterung hin untersucht. Es gilt also eine möglichst gute Kombination von untereinander unabhängigen Merkmalen zu finden, so dass das Verfahren tolerant gegenüber Schwankungen in der Bildqualität, aber auch selektiv im Hinblick auf die Filterung und die abschließende Sortierung ist.

7.1.1 Vorfilterung

Für die Vorfilterung sind besonders die einzelnen Merkmale innerhalb der Deskriptoren interessant, da eine Filterung mittels einzelner numerischer Werte die höchste Geschwindigkeit bietet. Zur Bewertung der Tauglichkeit wurden zunächst alle überhaupt in Frage kommenden Merkmale extrahiert und hinsichtlich ihrer Verteilung über die Gesamtmenge, die Varianz innerhalb der Gruppen und ihre Korrelation untereinander untersucht.

Die Ergebnisse der Verteilungs- und Varianzuntersuchung und der Effizienzmessungen können der Tabelle 8 entnommen werden. Das Filterratio besagt, wieviel Prozent der Ausgangsmenge nach der Filterung mit dem Merkmal unter der beschriebenen Toleranz erhalten bleiben. Ein niedriges Ratio ist also besser.

		Varianz		Filterratio	Nötige
Merkmal	Wertebereich	Gesamt	Gruppen	bei 1% FER	Toleranz
Seitenverhältnis	-111 – 127	36,01	1,38	50,5%	$\pm$ 15
HSV Fuzzy Covering	17,4 – 99,2	17,67	1,21	43,7%	$\pm$ 15
HSV Value Std. Dev.	26,7 – 109,9	14,12	1,15	34,3%	$\pm$ 9
1. Hu Moment	3,11 – 7,19	0,47	0,03	38,5%	$\pm$ 0,3
2. Hu Moment	6,23 – 21,92	1,65	0,14	51%	$\pm$ 1,5
NCM η_{20}	3,11 – 9,49	0,66	0,034	28,1%	$\pm$ 0,3
NCM η_{02}	4,25 – 11,04	0,66	0,035	29,2%	$\pm$ 0,31
Tamura *coarseness*	0 – 7,3	0,71	0,05	40,1%	$\pm$ 0,5
Tamura *contrast*	10 – 93,7	13,36	1,70	53,8%	$\pm$ 0,5
MPEG–7 CLD Y2 Koeff.	0 – 30	4	0,34	51,4%	$\pm$ 3
MPEG–7 CLD Y3 Koeff.	0 – 31	6,48	0,35	29%	$\pm$ 3

Tabelle 8: Übersicht potentieller Merkmale für die Vorfilterung

Um die Unabhängigkeit der Merkmale voneinander zu bestimmen, wird die Matrix der Korrelationskoeffizienten generiert (Abbildung 19), die die positive oder negative lineare Korrelation jeweils zweier Merkmale darstellt. Zur Auswahl der geeigneten Merkmale wurden Kombinationen möglicher Kandidaten gebildet und sämtliche Kreuzungspunkte auf eine Korrelation hin geprüft.

Eine gute Kombination, die keine signifikante lineare Korrelation untereinander aufweist, wird in Abbildung 20 vorgestellt und setzt sich aus dem Seitenverhältnis, zwei Merkmalen über die Helligkeitsanteile und zwei niedrigfrequenten DCT–Koeffizienten des Y–Kanals des MPEG–7 CLD (Kapitel 6.3.4) zusammen.

	TamTex_01	TamTex_00	NCM_02	NCM_20	M7_CLD_Y3	M7_CLD_Y2	HuMoment_2	HuMoment_1	HSVValDistStdDev	HSVValDistFuzzyCovering
AspectRatioValue	-0,039	0,108	0,776	-0,786	0,07	-0,051	0,342	0,222	-0,016	0,003
HSVValDistFuzzyCovering	-0,211	0,079	-0,501	-0,508	0,096	0,021	-0,391	-0,706	-0,047	
HSVValDistStdDev	0,842	0,196	0,015	0,011	-0,064	-0,021	0,075	0,044		
HuMoment_1	0,23	0,129	0,672	0,29	-0,044	-0,044	0,825			
HuMoment_2	0,162	0,262	0,585	-0,068	0,046	-0,046				
M7_CLD_Y2	-0,019	-0,047	-0,05	0,018	-0,041					
M7_CLD_Y3	-0,128	0,072	0,008	-0,143						
NCM_20	0,166	-0,163	-0,298							
NCM_02	0,12	-0,026								
TamTex_00	0,179									

Abbildung 19: Korrelationsmatrix der potentiellen Merkmale

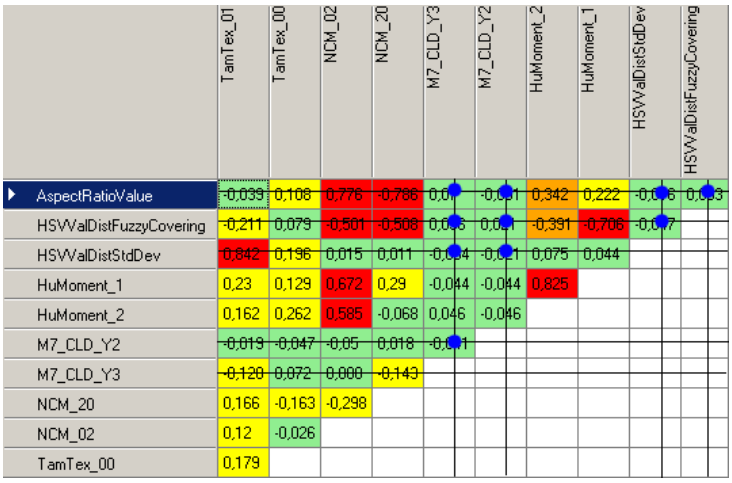


Abbildung 20: Eine mögliche Konfiguration die untereinander nicht Korreliert

In der vorgeschlagenen Kombination ergibt sich eine Gesamtreduktion der Kandidatenmenge auf nur noch 1,33% bzw. im Schnitt bleiben nur 37,4 von 2.795 Kandidaten für die aufwendige Filterung erhalten. Im Gegenzug erhöht sich natürlich auch die False Exclusion Rate auf 4,01%. Um dem entgegen zu wirken, müssen die Toleranzen wieder etwas angehoben werden.

Nach der Anpassung der Toleranzen und der aufsteigenden Sortierung nach dem Filterratio[47] ergibt sich die in Tabelle 9 dargestellte Konfiguration und Anordnung.

Bei genau einem Prozent False Exclusion Rate bleiben noch 3,84% der Kandidaten erhalten, die im Folgenden durch komplexere Filter bewertet und sortiert werden müssen. Abbildung 21 zeigt das Ergebnis einer Vorfilterung des ersten Testsets mit der vorgestellten Konfiguration.

Merkmal	Toleranz	Filterratio
MPEG–7 CLD Y3 Koeff.	± 5	$43,6\%$
HSV Value Std. Dev.	± 12	$44,7\%$
Seitenverhältnis	± 20	$52,7\%$
HSV Fuzzy Covering	± 20	$55,7\%$
MPEG–7 CLD Y2 Koeff.	± 4	$61,4\%$

Tabelle 9: Resultierende Konfiguration der Vorfilterung

7.1.2 Ergebnisranking

Zur Bewertung der Deskriptoren im Bezug auf die Sortierung der Ergebnisse werden zunächst die Ergebnisse unter Verwendung der in Kapitel 7.1.1 festgelegten Vorfilterung aktualisiert.

Bei den Ergebnisse in Tabelle 10 ist zu beachten, dass durch die Vorfilterung bereits 1% der Kandidaten fälschlicherweise ausgeschlossen wird, weshalb das höchstmögliche Ergebnis bei 99% liegt. Fast alle der komplexeren Filter liegen extrem dicht darunter. Auch die Geschwindigkeit der Suchvorgänge ist durch die Vorfilterung deutlich angestiegen. So benötigt eine Suche mit dem aufwendigen Image Distortion Model (Kapitel 6.6.1) statt ca. 250ms (siehe Tabelle 7, Kapitel 6.7) nur noch gut 11ms.

Da für die Vorfilterung neben einfachen Merkmalen auch komplexe Deskriptoren eingesetzt werden können und sich bei der Analyse des MPEG–7 CLD gezeigt hat (Kapitel 6.3.4), dass dieser auch sehr gut in diesem Bereich einsetzbar ist, wird die Vorfilterung um die CLD–Variante mit $10 : 6 : 6$ Koeffizienten erweitert, da diese in Tabelle 5 das beste Ergebnis für die Vorfilterung erzielt hat.

[47] Die Filterratios wurden nach der Anpassung der Toleranzen neu berechnet

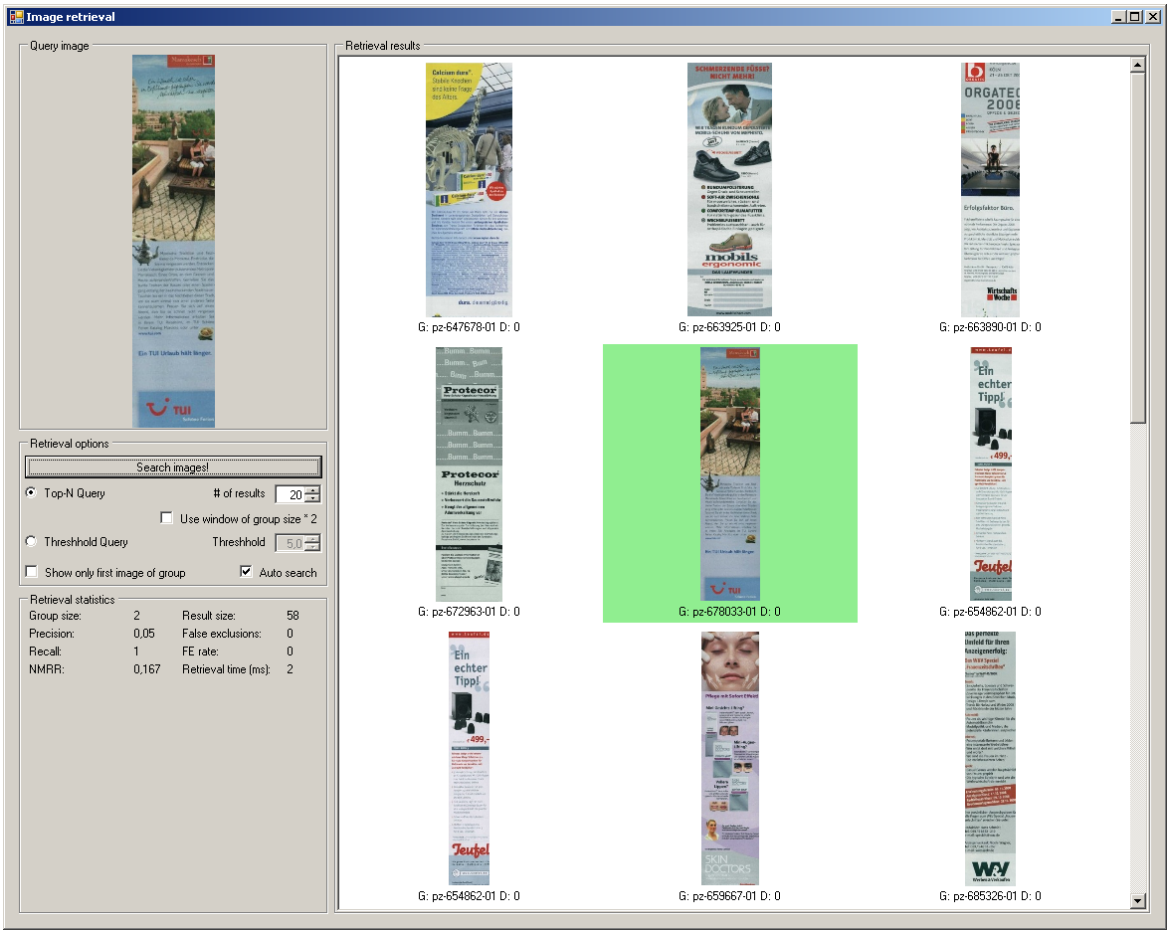


Abbildung 21: Kandidaten nach der Vorfilterung

Deskriptor	CLD	EHD	CLD (CW) (10:6:6))	SVM (3 Ebenen)	IDM
ANMRR	0,032	0,015	0,013	0,017	0,014
Goal Success Rate	97,78%	98,86%	98,93%	98,78%	98,86%
Ø Suchdauer	1,5ms	3,6ms	2,2ms	2ms	11,1ms

Tabelle 10: Ergebnisse der Deskriptoren nach Vorfilterung auf dem Testset 1

Durch die Erweiterung um den MPEG–7 CLD, ist die False Exclusion Rate nicht weiter angestiegen und bei 1,0% geblieben. Die Anzahl der verbleibenden Kandidaten ist jedoch im Schnitt von 107,2 auf 22,9 gesunken, was einer Filterung auf 0,82% der Ursprungsmenge entspricht.

Diese zusätzliche Einschränkung der Kandidatenmenge führt dazu, dass bereits drei Deskriptoren (der MPEG–7 EHD, die SVM mit 3 Ebenen und das IDM) eine perfekte Goal Success Rate von 99% für das erste Testset erreichen (siehe Tabelle 11).

Da die Grenzen des ersten Testsets mit seinen 2.796 Bildern erreicht sind, muss eine weitere Untersuchung der drei Kombinationen, welche jeweils die 99% erreichten, auf der Gesamtmenge mit 31.411 Bildern (Kapitel 3.4.2) durchgeführt werden.

Deskriptor	CLD	EHD	CLD (CW) (10:6:6))	SVM (3 Ebenen)	IDM
ANMRR	0,026	0,010	0,010	0,011	0,009
Goal Success Rate	98,14%	99,00%	98,93%	99,00%	99,00%
Ø Suchdauer	3,8ms	4,3ms	3,9ms	3,9ms	5,9ms

Tabelle 11: Ergebnisse nach Erweiterung der Vorfilterung um den MPEG–7 CLD

Die Ergebnisse dieser Untersuchung sind in Tabelle 12 dargestellt und zeigen das Image Distortion Model ganz leicht an der Spitze, mit nur 0,05% Vorsprung vor dem MPEG–7 Edge Histogramm. Der selbstentwickelte Scalable–Value–Matrix–Deskiptor liegt mit knapp einem Prozent Abstand dahinter, benötigt jedoch die wenigste Rechenzeit.

Zu beachten ist, dass bei dieser großen Menge an Daten mehr Zeit für die Vorfilterung als die Sortierung des Ergebnisses benötigt wird. So beträgt der Anteil der Vorfilterung an dem gesamten Suchvorgang ca. 60ms.

Da die größere Vielfalt der Bilder auch höhere Anforderungen an die Vorfilterung stellt, steigt die False Exclusion Rate für den Gesamtdatenbestand auf 2,99% an. Durch die teilweise größeren Gruppen führt ein einzelner, fehlerhafter Ausschluss jedoch nicht mehr direkt dazu, dass kein passendes Bild mehr gefunden werde kann.

Deskriptor	EHD	SVM (3 Ebenen)	IDM
ANMRR	0,056	0,066	0,048
Goal Success Rate	98,43%	97,60%	98,48%
Ø Suchdauer	65,1ms	61,3ms	82,8ms

Tabelle 12: Ergebnisse der drei besten Kombinationen auf der Gesamtmenge von 31.411 Bildern

7.2 Das Verfahren

Abschließend wird noch einmal das gesamte Verfahren in seinen Einzelschritten dargestellt.

7.2.1 Bildvorverarbeitung

Bevor die Merkmale extrahiert werden können, durchläuft jedes Bild die folgenden Schritte (Kapitel 5.4):

1. Verkleinerung auf 1200 Pixel bikubisch unter Beibehaltung des Seitenverhältnisses
2. Verkleinerung auf 600 Pixel bikubisch unter Beibehaltung des Seitenverhältnisses
3. Entfernung von Bildrändern mittels selbstentwickeltem, deckungsbasiertem Verfahren
4. Leichtes Beschneiden um 7 Pixel absolut, um letzte Ränder zu entfernen

Dieser Prozess dauert auf dem Testrechner ca. 1,1 Sekunden pro Bild.

7.2.2 Merkmalsextrakion

Aufgrund des leicht besseren Ergebnisses im Test (Tabelle 12) auf dem Gesamtdatenbestand und der Möglichkeit der zukünftigen Erweiterung und Verfeinerung, fällt die Wahl des Deskriptors für das Ranking der Ergebnisse auf das Image Distortion Model.

Es müssen also für jedes Bild die folgenden Merkmale extrahiert werden:

- Das Seitenverhältnis (Kapitel 6.2.1)
- Das HSV fuzzy Covering und die HSV Value Standardabweichung (Kapitel 6.3.1)
- Der MPEG–7 Color Layout Descriptor in der $10 : 6 : 6$ Konfiguration (Kapitel 6.3.4)
- Das 32×32 Byte–Array für das Image Distortion Model (Kapitel 6.6.1)

Auf dem Testrechner dauert die Extraktion dieser vier Deskriptoren im Schnitt 100ms pro Bild und lässt sich sehr gut parallelisieren. Der Speicherbedarf der einzelnen Merkmale kann in Tabelle 13 angesehen werden und beträgt pro Bild insgesamt 8.329 Bit, was aufgerundet 1.042 Byte entspricht.

Würde das MPEG–7 Edge Histogramm anstelle des IDM verwendet, so wäre der Featurevektor noch $1042 - 1024 + 30 = \underline{58}$ Byte groß.

Deskriptor	Anzahl Werte	Bit/Wert	Bit gesamt
Seitenverhältnis	1	8	8
HSV Statistiken	2	8	16
CLD Gleichanteile	3	6	18
CLD Frequenzanteile	19	5	95
Image Distortion Model	1024	8	8192
Summe Bits:			8329

Tabelle 13: Speicherbedarf der einzelnen Deskriptoren

7.2.3 Bildsuche

Die Suche nach einem Bild in der Datenbank erfolgt in drei Schritten:

1. Vorfilterung des Gesamtdatenbestand mit Hilfe der in Kapitel 7.1.1 beschriebenen und um den MPEG–7 CLD erweiterten Merkmalskonfiguration (für die Gesamtkonfiguration siehe Tabelle 14)

2. Berechnung der Distanzen und aufsteigende Sortierung der verbliebenen Kandidaten mittels Image Distortion Model

3. Ausgabe der ersten n Bilder an den Benutzer des Systems

Merkmal	Reihenfolge	Toleranz
MPEG–7 CLD Y3 Koeff.	1	± 5
HSV Value Std. Dev.	2	± 12
Seitenverhältnis	3	± 20
HSV Fuzzy Covering	4	± 20
MPEG–7 CLD Y2 Koeff.	5	± 4
MPEG–7 CLD $(10:6:6)$	6	± 20

Tabelle 14: Konfiguration der Vorfilterung inklusive MPEG–7 CLD

8 Fazit

8.1 Was wurde erreicht

In dieser Arbeit wurde ein Verfahren entwickelt und vorgestellt, welches eine effiziente und präzise Suche nach ähnlichen Bildern innerhalb einer Bilddatenbank ermöglicht und einen Großteil der in Kapitel 3.3 aufgestellten Anforderungen erfüllt.

Dazu wurde in Kapitel 3 ein Konzept erarbeitet, welches einen hierarchischen Prozess der Reduzierung der Bildmenge und der Sortierung vorsieht.

Für die Umsetzung dieses Konzepts ist in Kapitel 4 ein flexibles Software–Framework erstellt worden, das es ermöglicht, die Einflüsse von Vorverarbeitungstechniken zu bewerten (Kapitel 5) und die Eignung verschiedener Bildeigenschaften und Deskriptoren hinsichtlich der Ähnlichkeitsbeurteilung von Bildern zu untersuchen (Kapitel 6).

Auf Basis der ermittelten Kennzahlen wurde in Kapitel 7 eine hierarchische Anordnung von Merkmalen festgelegt. Diese erlaubt eine effektive Filterung der Bildmenge auf wenige Kandidaten, die mittels eines komplexeren Deskriptors oder Verfahrens sortiert und dem Anwender präsentiert werden.

Eine Bewertung des Verfahrens hinsichtlich der Erfüllung der gestellten Anforderungen kann der Tabelle 15 entnommen werden.

Anforderung	Ergebnis	Ziel erreicht
Vorverarbeitung und Merkmalsextraktion innerhalb 5 Sekunden	In Summe dauert dieser Vorgang ca. $1,1 + 0,1 = 1,2$s	Ja
Sofortige Findbarkeit des Bildes	Bei jedem Suchvorgang wird die komplette Datenbank durchgegangen	Ja
Geringer Speicherbedarf durch kompakte Deskriptoren	Das IDM führt zu einem relativ großen Merkmalsvektor. In kritischen Bereichen mit wenig Speicher könnte jedoch auf das MPEG–7 EHD ausgewichen werden.	Teilweise
Die Struktur der Deskriptoren soll effiziente Indizierung erlauben	Die ersten 5 Merkmale der Vorfilterung lassen sich gut indizieren, und verkleinern das Ergebnis auf wenige Prozent	Ja
Minimierung der *false-negatives*	Die Rate beträgt 1% auf Testset 1 sowie ca. 3% auf der Gesamtmenge	Ja
Die Dauer des gesamten Suchvorgangs soll unter einer Sekunde liegen	Auf 31.411 Bildern dauert der Suchvorgang im Schnitt 83ms. Linear auf 800.000 skaliert, würde eine Suche etwa 2,1s benötigen	Nein
Vorhandene Treffer sollen innerhalb der ersten 5 Positionen erscheinen	Diese Anforderung entspricht genau der Goal Success Rate, die für das vorgestellte Verfahren 98,48% beträgt	Ja

Tabelle 15: Erfüllung der Anforderungen aus Kapitel 3.3

8.2 Ausblick

8.2.1 Offene Punkte

Die wichtigsten offenen Punkte für einen produktiven Einsatz des vorgestellten Verfahrens sind:

- Erhöhung der Geschwindigkeit der Suchanfragen, da diese durch die Implementierung des Frameworks, das auf Flexibilität ausgelegt wurde, negativ beeinflusst wird
- Die Indizierung der Merkmale zur Vorfilterung, da aktuell für jeden Suchvorgang auf alle Merkmalsvektoren mindestens einmal zugegriffen werden muss
- Die effiziente Speicherung der extrahierten Merkmale

Denkbar ist auch die Anbindung an eine zentrale Datenbank, so dass das Verfahren dezentral an verschiedenen Stellen nutzbar ist.

8.2.2 Optimierung

Ein äußerst interessanter und zunehmend populärer Bereich ist die hochparallele Bildverarbeitung mittels Graphics Processing Unit (GPU). NVIDIA[48] hat 2006 mit CUDA[49] eine Architektur vorgestellt, die inzwischen in der Version 2.1 verfügbar ist und die Programmierung von GPUs in einem leicht angepassten C ermöglicht. Moderne High–end Grafikkarten wie die NVIDIA GeForce GTX 285[50] bieten bis zu 240 parallele Rechenkerne, die bei 1476MHz pro Kern eine Rechenleistung von maximal 1062 GFLOPS[51] Spitzenleistung erreichen[52].

Der Teil des Verfahrens, der vermutlich am meisten von einer solch massiven Parallelisierung profitieren würde, ist das Image Distortion Model. Hier gibt es fast ausschließlich lesende Operationen und keinen konkurrierenden Zugriff auf den Speicher. So könnte in einer hochparallelen Umgebung jeder Pixel des Quellbildes in einem eigenen Thread mit dem korrespondierenden Pixel des Zielbildes und seiner Nachbarschaft verglichen und die minimale Distanz berechnet werden.

Durch die deutlich höhere verfügbare Rechenleistung könnten auch die in Kapitel 6.6.1 vorgeschlagenen Optimierungen, wie die Vergrößerung der Bilder auf 64×64 Pixel oder die Erweiterung der Nachbarschaftsdefinition auf 5×5 Pixel, für eine sensiblere Distanzberechnung realisiert werden.

Gemeinsam mit einer effizienten Implementierung der Distanzberechnung für den MPEG–7 Color Layout Descriptor und der Indizierung der einfachen Merkmale der Vorfilterung, würde das Verfahren vermutlich um ein Vielfaches beschleunigt werden, was letztendlich auch zur Erfüllung der letzten offenen Anforderung aus Tabelle 15 führen sollte.

[48] `http://www.nvidia.com/`

[49] `http://www.nvidia.com/object/cuda_home.html`

[50] Produktlink: `http://www.nvidia.com/object/product_geforce_gtx_285_us.html`

[51] Floating point operations per second

[52] Der verwendete Prozessor für diese Arbeit (Intel Core 2 Quad 2,4GHz) erreicht eine Leistung von knapp 40 GFLOPS. `http://www.intel.com/support/processors/sb/cs-023143.htm`

Literatur

[Batko u. a. 2004] BATKO, Michal ; GENNARO, Claudio ; SAVINO, Pasquale ; ZEZULA, Pavel: Scalable similarity search in metric spaces. In: *In: Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation, D1.1.1 Final 30/39 Supprimé : : of the 6 th Thematic Workshop of the EU Network of Excellence DELOS, S. Margherita di*, 2004, S. 213–224

[Bauer u. a.] BAUER, Johannes ; SÜNDERHAUF, Niko ; PROTZEL, Peter: COMPARING SEVERAL IMPLEMENTATIONS OF TWO RECENTLY PUBLISHED FEATURE DETECTORS.

[Bay u. a. 2006] BAY, Herbert ; TUYTELAARS, Tinne ; VAN GOOL, L.: SURF: Speeded Up Robust Features. In: *9th European Conference on Computer Vision*. Graz Austria, May 2006

[Birgale u. a. 2006] BIRGALE, L. ; KOKARE, M. ; DOYE, D.: Colour and Texture Features for Content Based Image Retrieval. In: *Computer Graphics, Imaging and Visualisation, 2006 International Conference on* (2006), July, S. 146–149

[Chatzichristofis und Boutalis 2008a] CHATZICHRISTOFIS, S.A. ; BOUTALIS, Y.S.: FCTH: Fuzzy Color and Texture Histogram - A Low Level Feature for Accurate Image Retrieval. In: *Image Analysis for Multimedia Interactive Services, 2008. WIAMIS '08. Ninth International Workshop on* (2008), May, S. 191–196

[Chatzichristofis und Boutalis 2008b] CHATZICHRISTOFIS, Savvas A. ; BOUTALIS, Yiannis S.: CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval. In: GASTERATOS, Antonios (Hrsg.) ; VINCZE, Markus (Hrsg.) ; TSOTSOS, John K. (Hrsg.): *ICVS* Bd. 5008, Springer, 2008, S. 312–322. – ISBN 978-3-540-79546-9

[Chavez und Navarro 2000] CHAVEZ, E. ; NAVARRO, G.: An effective clustering algorithm to index high dimensional metric spaces. In: *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on* (2000), S. 75–86

[Chen und Stentiford 2006] CHEN, Li ; STENTIFORD, F. W.: Comparison of Near-Duplicate Image Matching. In: *Visual Media Production, 2006. CVMP 2006. 3rd European Conference on*, 2006, S. 38–42. – ISBN 978-0-86341-729-0

[Chi Wong u. a. 2002] CHI WONG, H. ; BERN, M. ; GOLDBERG, D.: An image signature for any kind of image. In: *Image Processing. 2002. Proceedings. 2002 International Conference on* 1 (2002), S. I–409–I–412 vol.1. – ISSN 1522-4880

[Datta u. a. 2008] DATTA, Ritendra ; JOSHI, Dhiraj ; LI, Jia ; WANG, James Z.: Image retrieval: Ideas, influences, and trends of the new age. In: *ACM Comput. Surv.* 40 (2008), Nr. 2, S. 1–60. – ISSN 0360-0300

[Deselaers u. a. 2005] DESELAERS, Thomas ; KEYSERS, Daniel ; NEY, Hermann: FIRE – Flexible Image Retrieval Engine: ImageCLEF 2004 Evaluation. In: *CLEF Workshop 2004* Bd. 3491. Bath, UK : Springer, 15/09/2004 2005, S. 688–698. – A very similar (unreviewed) paper appeared in the Working Notes of the CLEF Workshop 2004.

[Deselaers u. a. 2008] DESELAERS, Thomas ; KEYSERS, Daniel ; NEY, Hermann: Features for image retrieval: an experimental comparison. In: *Inf. Retr.* 11 (2008), Nr. 2, S. 77–107. – ISSN 1386-4564

[Faloutsos u. a. 1994] FALOUTSOS, C. ; EQUITZ, W. ; FLICKNER, M. ; NIBLACK, W. ; PETKOVIC, D. ; BARBER, R.: Efficient and effective querying by image content. In: *Journal of Intelligent Information Systems* 3 (1994), S. 231–262

[Flickner u. a. 1995] FLICKNER, M. ; SAWHNEY, H. ; NIBLACK, W. ; ASHLEY, J. ; HUANG, Qian ; DOM, B. ; GORKANI, M. ; HAFNER, J. ; LEE, D. ; PETKOVIC, D. ; STEELE, D. ; YANKER, P.: Query by image and video content: the QBIC system. In: *Computer* 28 (1995), Sep, Nr. 9, S. 23–32. – ISSN 0018-9162

[Foo und Sinha 2007] FOO, Jun J. ; SINHA, Ranjan: Pruning SIFT for scalable near-duplicate image matching. In: *ADC '07: Proceedings of the eighteenth conference on Australasian database*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2007, S. 63–71. – ISBN 1-920-68244-9

[Gonzalez und Woods 2002] GONZALEZ, Rafael C. ; WOODS, Richard E.: *Digital Image Processing (2nd Edition)*. Prentice Hall, January 2002. – URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0201180758`. – ISBN 0201180758

[Hafner u. a. 1995] HAFNER, J. ; SAWHNEY, H.S. ; EQUITZ, W. ; FLICKNER, M. ; NIBLACK, W.: Efficient color histogram indexing for quadratic form distance functions. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17 (1995), Jul, Nr. 7, S. 729–736. – ISSN 0162-8828

[Hjaltason und Samet 2003] HJALTASON, Gisli R. ; SAMET, Hanan: Index-driven similarity search in metric spaces (Survey Article). In: *ACM Trans. Database Syst.* 28 (2003), Nr. 4, S. 517–580. – ISSN 0362-5915

[Hu 1962] HU, Ming-Kuei: Visual pattern recognition by moment invariants. In: *Information Theory, IRE Transactions on* 8 (1962), February, Nr. 2, S. 179–187. – ISSN 0096-1000

[Huang u. a. 1997] HUANG, Jing ; KUMAR, S. R. ; MITRA, Mandar ; ZHU, Wei-Jing ; ZABIH, Ramin: Image Indexing Using Color Correlograms. In: *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*. Washington, DC, USA : IEEE Computer Society, 1997, S. 762. – ISBN 0-8186-7822-4

[Keysers u. a. 2004] KEYSERS, Daniel ; GOLLAN, Christian ; NEY, Hermann: Classification of medical images using non-linear distortion models. In: *In Bildverarbeitung für die Medizin*, Springer-Verlag, 2004, S. 366–370

[Lowe 1999] LOWE, David G.: Object recognition from local scale-invariant features, 1999, S. 1150–1157

[Manjunath u. a. 2001] MANJUNATH, B.S. ; OHM, J.-R. ; VASUDEVAN, V.V. ; YAMADA, A.: Color and texture descriptors. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11 (2001), Jun, Nr. 6, S. 703–715. – ISSN 1051-8215

[Mathias Lux 2008] MATHIAS LUX, S. A. C.: LIRe: Lucene Image Retrieval - An Extensible Java CBIR Library. In: *ACM International Conference on Multimedia 2008*, 2008

[Niblack u. a. 1993] NIBLACK, W. ; BARBER, R. ; EQUITZ, W. ; FLICKNER, M. D. ; GLASMAN, E. H. ; PETKOVIC, D. ; YANKER, P. ; FALOUTSOS, C. ; TAUBIN, G.: QBIC project: querying images by content, using color, texture, and shape. In: NIBLACK, W. (Hrsg.): *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* Bd. 1908, April 1993, S. 173–187

[Patino-Escarcina und Costa 2007] PATINO-ESCARCINA, R.E. ; COSTA, J.A.F.: Content Based Image Retrieval using a Descriptors Hierarchy. In: *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on* (2007), Sept., S. 228–233

[Rui u. a. 1999] RUI, Yong ; HUANG, Thomas S. ; CHANG, Shih fu: Image retrieval: Current techniques, promising directions and open issues. In: *Journal of Visual Communication and Image Representation* 10 (1999), S. 39–62

[Salembier und Sikora 2002] SALEMBIER, Phillipe ; SIKORA, Thomas ; MANJUNATH, B.S. (Hrsg.): *Introduction to MPEG-7: Multimedia Content Description Interface*. New York, NY, USA : John Wiley & Sons, Inc., 2002. – ISBN 0471486787

[Schaefer und Stich 2004] SCHAEFER, Gerald ; STICH, Michal: UCID - An Uncompressed Colour Image Database. In: *In Storage and Retrieval Methods and Applications for Multimedia 2004, volume 5307 of Proceedings of SPIE*, 2004, S. 472–480

[Sebe und Lew 2002] SEBE, N. ; LEW, M.: Robust Shape Matching. In: *In Proceedings of the International Conference on Image and Video Retrieval*, SpringerVerlag, 2002, S. 17–28

[Smith und fu Chang 1996] SMITH, John R. ; CHANG, Shih fu: Visualseek: a fully automated content-based image query system, 1996, S. 87–98

[Tamura u. a. 1978] TAMURA, Hideyuki ; MORI, Shunji ; YAMAWAKI, Takashi: Textural Features Corresponding to Visual Perception. In: *Systems, Man and Cybernetics, IEEE Transactions on* 8 (1978), June, Nr. 6, S. 460–473. – ISSN 0018-9472

[Torres und Falcão 2006] TORRES, Ricardo Da S. ; FALCÃO, Alexandre X.: Content-Based Image Retrieval: Theory and Applications. In: *Revista de Informática Teórica e Aplicada* 13 (2006), S. 161–185

[Vasconcelos 2007] VASCONCELOS, Nuno: From Pixels to Semantic Spaces: Advances in Content-Based Image Retrieval. In: *Computer* 40 (2007), Nr. 7, S. 20–26. – ISSN 0018-9162

[Veltkamp und Tanase 2002] VELTKAMP, Remco C. ; TANASE, Mirela: Content-based image retrieval systems: A survey, revised and extended edition / Department of Computing Science, Utrecht University. 2002. – Forschungsbericht

[White und Jain 1996] WHITE, David A. ; JAIN, Ramesh: Similarity indexing: Algorithms and performance. In: *In Storage and Retrieval for Image and Video Databases (SPIE*, 1996, S. 62–73

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 27. Februar 2009 Ben Struss