Universität Innsbruck

Institut für Informatik

Intelligente und Interaktive Systeme

Image Retrieval

Chris Wendler und Matej Stanic

Bachelorarbeit

Betreuer: Univ.-Prof. Dr. Justus Piater, PhD

24. November 2013

Kurzfassung

Diese Arbeit beschäftigt sich mit einer Machbarkeitsstudie eines Content-based Image Retrieval-Systems und präsentiert infolgedessen essentielle Schlüsseltechnologien, welche von der Auswahl eines geeigneten Deskriptors, über die Wirkung verschiedener Metriken und Normalisierungen, bis hin zur Laufzeitoptimierung mittels Indexing und der Verbesserung der Ergebnisqualität mittels Relevance Feedback reichen. Diese Schlüsseltechnologien werden in unser eigenes System integriert und anschließend evaluiert. Das von uns implementierte System erzielt mit einfachen Mitteln akzeptable Ergebnisse und kann als Grundlage für größere Softwareprojekte verwendet werden.

Danksagungen

Ohne die Hilfe der unten angeführten Personen wäre diese Bachelorarbeit nicht möglich gewesen. In diesem Sinne wollen wir uns bei allen Beteiligten bedanken.

Danke an unseren Betreuer Justus Piater, der uns während der gesamten Dauer dieser Bachelorarbeit begleitet und exzellent betreut hat und der außerdem einen massiven Zeitaufwand betrieben hat, um während den vielen Treffen in seinem Büro und auch außerhalb rund um die Uhr für uns da zu sein.

Danke an Antonio José Rodríguez Sánchez, der uns diese Bachelorarbeit vorgeschlagen hat, der an der Betreuung unserer Bachelorarbeit maßgeblich beteiligt war und der mit seiner Anwesenheit bei all unseren Treffen durch seine Expertise geglänzt hat.

Danke an Herrn Wiener von der Firma M-BOX, der uns die Kundendatenbanken für die Evaluierung zur Verfügung gestellt hat.

Danke an Simon Haller, der uns technischen Support beim Einrichten der Versionsverwaltungssoftware geboten hat.

Danke.

Inhaltsverzeichnis

Kurzfassung **i**

Danksagungen **iii**

Inhaltsverzeichnis **v**

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **ix**

Erklärung **xi**

1 Einleitung **1**

2 Deskriptoren **5**
2.1 Typen von Deskriptoren . . . 5
2.2 Kombinierte Deskriptoren . . . 6
2.3 Der Fluch der Dimensionalität . . . 7
2.4 Auswahl von relevanten Deskriptoren für CBIR-Systeme . . . 8
2.4.1 Testkonfiguration und -ergebnisse . . . 8
2.4.2 Schlussfolgerung und Auswahl . . . 9
2.5 Verwendete Deskriptoren . . . 14
2.5.1 Color Histogram . . . 14
2.5.2 MPEG7-EHD . . . 14
2.5.3 CEDD . . . 15

3 Metriken **17**
3.1 Euklidische Metriken . . . 17
3.1.1 Die Basisform . . . 17
3.1.2 Die gewichtete euklidische Metrik . . . 18
3.2 Kosinus-Metriken . . . 18
3.2.1 Kosinus-Ähnlichkeit . . . 18
3.2.2 Gewichtete Kosinus-Ähnlichkeit . . . 19
3.2.3 Kosinus-Distanz . . . 19
3.3 Earth Mover's Distance . . . 20

4 Evaluierung der verwendeten Deskriptoren und Metriken **21**
4.1 Evaluierung des kombinierten Deskriptors . . . 21
4.1.1 Testkonfiguration und -ergebnisse . . . 21
4.2 Automatisierter Test mit annotierten Bildern . . . 24

5 Indexing **27**
5.1 k-d-Bäume . . . 28
5.1.1 Definition . . . 28
5.1.2 Operationen . . . 28
5.2 Fluch der Dimensionalität . . . 31
5.3 Evaluierung . . . 32

6 Relevance Feedback **35**
6.1 Pseudo-Relevance Feedback . . . 36
6.1.1 Gewichtung der einzelnen Features . . . 36
6.1.2 Verschiebung des Query-Punktes . . . 37
6.2 Bayesischer Ansatz . . . 38
6.3 Nearest-Neighbor-Methoden . . . 39
6.3.1 Stabilisierung . . . 40
6.4 Relevance Feedback als Klassifikationsproblem . . . 41
6.5 Analyse bezüglich der Kompatibilität mit Indexing . . . 42
6.5.1 Gewichtung der einzelnen Features . . . 42
6.5.2 Querypunkt Verschiebungen . . . 42
6.5.3 Nearest-Neighbors-RF . . . 42
6.6 Evaluierung . . . 43
6.6.1 Automatische Evaluierung . . . 44
6.6.2 Subjektive Evaluierung . . . 45

7 Implementierung **53**
7.1 Funktionsweise und Details . . . 53

8 Schlussfolgerung **55**

Literaturverzeichnis **57**

A Notation **59**
A.1 Deskriptoren . . . 59
A.2 Metriken . . . 59
A.3 Relevance Feedback . . . 59

B Arbeitsverteilung **61**

Abbildungsverzeichnis

1.1 Illustration eines simplen CBIR-Systems. 2

2.1 Transformation von Bild zu Deskriptor-Vektor. 5
2.2 Querybilder *lentos-drawing*, *lentos-box*, *lentos-cartoon* und *lentos-group* der Datenbank *Lentos*. Anm.: Wegen ungeklärter Lizenzierung wird bei *lentos-drawing* nur eine Karikatur angezeigt, die dem Bild ähnlich schaut. 10
2.3 Querybilder *madias-warrior*, *madias-pinkbg*, *madias-building* und *madias-tree* der Datenbank *Madias* 10
2.4 Querybilder *oemv-knife*, *oemv-woman*, *oemv-sheep* und *oemv-image* der Datenbank *OeMV* 10
2.5 Querybilder *cars-wheel*, *cars-multiple* und *cars-red/cars-side* der Cars-Datenbank 11
2.6 Präzision der getesteten Deskriptoren in % (Testdatenbank Lentos) 11
2.7 Präzision der getesteten Deskriptoren in % (Testdatenbank Madias) 12
2.8 Präzision der getesteten Deskriptoren in % (Testdatenbank OeMV) 12
2.9 Präzision der getesteten Deskriptoren in % (Testdatenbank Cars) 13
2.10 Gesamtpräzision der getesteten Deskriptoren in % (mit Fehlerbalken) 13

3.1 Euklidische Distanz zweier Punkte im 2-dimensionalen Raum. 17
3.2 Gewichtete euklidische Distanz zweier Punkte im 2-dimensionalen Raum. 18
3.3 Kosinus-Ähnlichkeit im 2-dimensionalen Raum. 19

4.1 Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Lentos) 22
4.2 Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Madias) 22
4.3 Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank OeMV) 23
4.4 Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Cars) 23
4.5 Gesamtpräzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (mit Fehlerbalken) 24
4.6 Präzision des kombinierten Deskriptors im Vergleich zu Color Histogram, MPEG7-EHD und CEDD 25

4.7 Durchschnittliche benötigte Zeit für eine Suchiteration mit dem kombinierten Deskriptor im Vergleich zu Color Histogram, MPEG7-EHD und CEDD (Ergebnismengengröße = 10) . . . 26

5.1 k-d-Baum-Partitionierung für die zweidimensionale Punktemenge (8.5,4), (4,9), (13,7), (6,6), (2.5,14), (10.5,5), (12,13). . . . 29
5.2 Der aus Abbildung 5.1 resultierende k-d-Baum. . . . 29
5.3 Illustration des Speedups beim Anwenden des Nearest-Neighbor-Search-Algorithmus für den Punkt (3,11) auf den k-d-Baum aus Abbildung 5.2. . . . 30
5.4 Durchschnittlich benötigte Zeit für eine Suchiteration mit und ohne Indexing in der *Corel*-Datenbank. . . . 33

6.1 Beispiel für „Semantic Gap“: Der Benutzer sucht nach Fahrrädern, das CBIR-System findet Fahrräder und Sonnenuntergänge. Durch RF kann der Benutzer dem System mitteilen, dass er nach Fahrrädern sucht und nicht nach Sonnenuntergängen. . . . 35
6.2 Illustration eines CBIR-Systems mit RF. . . . 36
6.3 Illustration der approximierten decision boundary und der Querypunktverschiebung von $Q^{(i)}$ nach $Q^{(i+1)}$. . . . 39
6.4 (a) Durchschnittliche Präzision, (b) Standardabweichung, (c) durchschnittliche Zeit einer RF-Iteration der verschiedenen RF-Methoden gemessen über 500 Queries mit jeweils 4 RF-Iterationen in der Corel Datenbank. . . . 49
6.5 Beispiel für Bilder mit ähnlichem Inhalt und unterschiedlichen Objektkategorien in der Corel Datenbank (Die Katze links ist in der Klasse *Wildkatze* und die Katze rechts in der Klasse *Hauskatze*) . . . 50
6.6 (a) Durchschnittliche Präzision, (b) Standardabweichung von 16 Queries mit jeweils 4 RF-Iterationen in den 4 M-BOX Datenbanken. . . . 50
6.7 (a) Beispiel für eine Query, bei der kein RF-Mechanismus die Ergebnisse verbessern konnte. (b) Beispielquery, bei der RF sehr gut funktioniert. . . . 51
6.8 (a) Ergebnismenge der Query **cars-red** bzw. **cars-side** ohne RF. (b) Ergebnismenge der Query **cars-red** bzw. **cars-side** nach 4 RF-Iterationen mittels BQS- und NN-RF. . . . 52

7.1 Allgemeines Modell eines CBIR-Systems . . . 54

Tabellenverzeichnis

B.1 Arbeitsverteilung bezüglich der einzelnen Kapitel 61

Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Signiert: ..... Datum: .....

Kapitel 1

Einleitung

Dank des technischen Fortschritts der letzten Jahrzehnte wurde es immer einfacher und kostengünstiger große Bilddatenbanken aufzubauen. Immer mehr Personen, Firmen und Institutionen wie Museen oder Archive besitzen eigene Bilddatenbanken, deren Größen recht unterschiedlich sein können. Bei sehr großen Datenbanken ist es schwierig, sich zurechtzufinden, Millionen von Bilder müssen suchbar gemacht machen. Mit diesem Problem beschäftigt sich *Image Retrieval (IR)*. Das Ziel von IR-Systemen ist es, Bilder in einer gegebenen Datenbank aufzufinden, die ähnlich zu einer vom User spezifizierten Anforderung sind. Als Suchergebnis bekommt der Benutzer eine Liste von Bildern zurück, die nach deren Relevanz bezüglich der Suchkriterien sortiert ist.

Die am weitesten verbreitete Methode für IR ist die textuelle Bildersuche. Jedes Bild besitzt manuell hinzugefügte Metadaten (meistens kurze Stichwörter), welche das Bild beschreiben. Startet der Benutzer eine Suchanfrage, gibt er bestimmte Stichwörter an, nach denen gesucht werden soll. Das IR-System vergleicht die Metadaten der einzelnen Bilder und erstellt ein Ranking bezüglich der Relevanz der Bilder.

Obwohl die textuelle Bildersuche in der Praxis gut zu funktionieren scheint, verbergen sich zahlreiche Probleme dahinter:

- Eine textuelle Beschreibung ist immer mit einer bestimmten Sprache verbunden. Potenziell relevante Bilder können wegfallen, wenn sich die Sprache der Bildmetadaten von der Sprache der Benutzereingabe unterscheidet. Dies gilt auch für Synonyme in gleicher Sprache (z.B Kartoffel/Erdapfel oder Kamin/Schornstein).

- Jedes Bild muss manuell beschrieben werden. Bei sehr großen Datenbanken kommt es daher schnell zu hohen Kosten, da das textuelle Beschreiben durch den Menschen zeitaufwändig ist.

- Jeder Mensch macht Fehler - man muss davon ausgehen, dass Bilder in Datenbanken falsche Metadaten besitzen (z.B. durch Rechtschreibfehler). Somit könnten relevante Bilder bei Suchanfragen wegfallen.

Eine Möglichkeit, die genannten Probleme zu vermeiden, ist den Inhalt der Bilder zu vergleichen. Diese Suchmethode wird als *Content-based Image Retrieval (CBIR)* bezeichnet. Mittels sogenannten *Deskriptoren* lassen sich qualitative Merkmale eines Bildes extrahieren, die in Vektoren gespeichert werden und zum Vergleich von Bildern benutzt werden. Deskriptoren sind aus Performancegründen meist einfach gehalten, d.h. sie fokussieren sich gezielt auf bestimmte

Merkmale (z.B Farbe, Textur, Form). Dadurch entstehen Nachteile gegenüber der textbasierten Suche:

- Da nur bestimmte Features eines Bildes berücksichtigt werden, sind die Ergebnisse einer Suche meistens nicht so gut wie gewünscht. Mittels *Relevance Feedback (RF)* können die Ergebnisse verbessert werden: Nach einer erfolgreichen Suche werden positive und negative Ergebnisse vom User markiert, welche in der nächsten Iteration berücksichtigt werden und zu besseren Ergebnissen führen sollen.

- Deskriptoren extrahieren spezielle Features eines Bildes und speichern sie in einem Vektor. Um die Ähnlichkeit zweier Vektoren vergleichen zu können, bedarf es einer *Metrik*. Eine Metrik ist eine Funktion, die zwei Vektoren eine Distanz zuweist. Je kleiner die Distanz zwischen zwei Vektoren, desto ähnlicher sind die dazugehörigen Bilder bezüglich dem Deskriptor. Aufgrund der häufigen Distanzberechnungen, die mit steigender Vektordimension komplexer werden, kommt es zu einem Performanceverlust. Durch geeignete *Indexstrukturen* kann man versuchen, diesen Performanceverlust wieder auszugleichen.

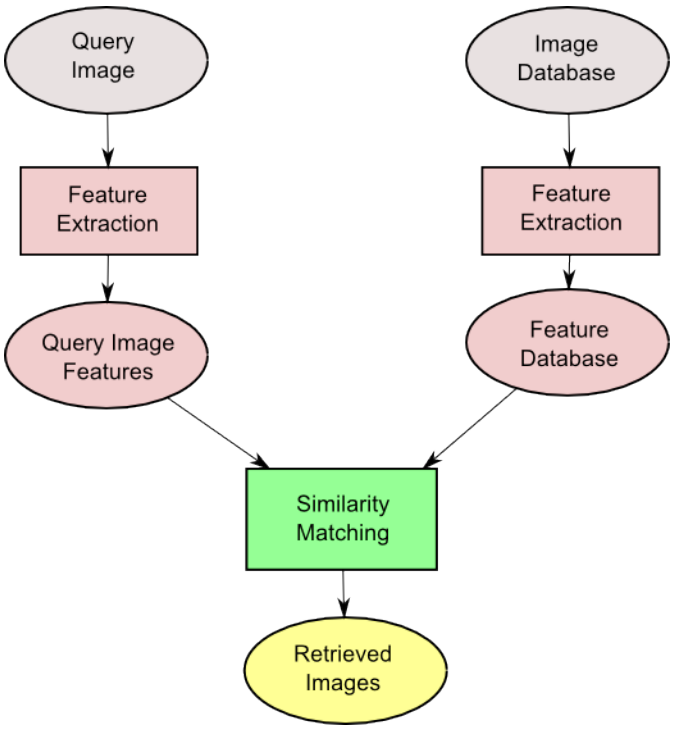


Abbildung 1.1: Illustration eines simplen CBIR-Systems.

Mit der Repräsentation von Bildern als Deskriptoren und einer Metrik, um diese Deskriptoren zu vergleichen, ist der Grundstein eines CBIR-Systems geschaffen. Um in der Praxis ein CBIR-System aufzusetzen, sind also zuerst die Deskriptoren für die gegebene Bilderdatenbank zu extrahieren. Anschließend kann für jedes beliebige Querybild eine Query erstellt werden. Dazu wird der Deskriptor vom Querybild extrahiert, dieser muss vom selben Typ sein wie die in der Datenbank vorhandenen Deskriptoren. Zu beachten ist jedoch, dass ein Querybild allein keine Query definiert. Vielmehr ist es die Vorstellung des Benutzers über gewünschte Ergebnisse, die eine Query ausmachen. Da der Benutzer dem System seine Vorstellungen nicht direkt mitteilen kann, muss er ein Querybild finden, welches seine Vorstellungen am besten approximiert.

Anschließend kann eine Rangliste der Bilder in der Datenbank bezüglich des Query-Deskriptors mittels einer Metrik erstellt werden. Diese Schritte sind in Abbildung 1.1 illustriert. Die naivste Implementierung für eine Rangliste von Bildern ist eine Sortierung aller Bilder der Datenbank bezüglich des Abstandes ihrer jeweiligen Deskriptoren zum Query-Deskriptor. Die Abstände werden mittels einer der vorgeschlagenen Metriken berechnet. Im Idealfall erfasst der gewählte Deskriptortyp die Inhalte der Bilder so gut, dass die Ergebnisse zufriedenstellend sind und die Datenbank ist klein genug, dass die Laufzeit gering ist. Technologien wie Indexing und Relevance Feedback wären dann überflüssig. Die Realität sieht allerdings anders aus. Zufriedenstellende Ergebnisse sind in keinem Fall garantierbar, da die subjektive Bildähnlichkeitswahrnehmung von der Ähnlichkeit der Bilder bezüglich einer Metrik abweicht. Des Weiteren ist das Ausmaß von heutigen Online-Bilddatenbanken wie *Picasa*, *Flickr* oder *Instagram* so gewaltig, dass Indexing zur Laufzeitreduzierung unerlässlich ist.

Bei der Evaluierung stellt man schnell fest, dass die Qualität der Ergebnisse eines solchen CBIR-Systems stark von folgenden Faktoren abhängen:

- **Die Datenbank**
  Logischerweise hängen die Ergebnisse eines CBIR-Systems nicht nur vom System ab, sondern auch von der Bilddatenbank, auf die das System angewandt wird. Wenn für ein Querybild keine relevanten Bilder in der Datenbank vorhanden sind, gibt es keine guten Ergebnisse.

  Anzahl der Bilder

  Ein datenbankspezifischer Faktor, von dem die Qualität der Suchergebnisse stark abhängt, ist die Anzahl von Bildern. Durchschnittlich ist die Qualität von Suchergebnissen in Bilddatenbanken mit größerer Anzahl von Bildern höher. Wenn mehr Bilder in einer Datenbank vorhanden sind, dann ist die Wahrscheinlichkeit für eine bessere Abdeckung (Diversität) aller möglicher Querybilder höher. Zusätzlich ist es wahrscheinlicher, dass von den verschiedenen Bildklassen jeweils mehrere Exemplare vorhanden sind, also eine höhere Bilddichte herrscht.

  Dichte der Bildklassen

  Neben der Anzahl von Bildern spielt die Dichte der einzelnen Bildklassen eine gravierende Rolle. Beispielsweise ist in einer Autosport-Datenbank die Bilddichte von Autos, Fahrern, Reifen oder Lenkrädern sehr groß und die Bilddichte von Hunden und Katzen vermutlich minimal. Folglich wird man in dieser Datenbank mit Querybildern, die Autos enthalten, exzellente Ergebnisse erzielen und mit Querybildern, die Katzen enthalten, miserable.

  Diversität der Bilder

  Die Diversität der Bilder in einer Datenbank bestimmt die Wahrscheinlichkeit für ein zufälliges Querybild gute Ergebnisse zu erzielen. In unserer Autosport-Datenbank ist diese Wahrscheinlichkeit sehr klein, da die Datenbank nur Bilder enthält, die in irgendeiner Weise mit Autos zu tun haben. Im Gegensatz dazu ist sie in der *WANG*-Datenbank[1] sehr groß, weil diese speziell für die CBIR-Evaluierung entwickelt wurde und viele verschiedene Bildkategorien enthält.

  Qualität der Bilder

  Die Qualität der Bilder beeinflusst die Qualität der Deskriptoren. Je niedriger die Bildqualität ist, desto schwieriger ist es Informationen zu extrahieren. Wenn man zum Beispiel

[1] `http://wang.ist.psu.edu/docs/related/`

nur die Bildauflösung als Qualitätsmerkmal betrachtet, stellt man fest, dass Bilder mit geringerer Auflösung deutlich weniger Informationen enthalten.

- **Die verwendeten Deskriptoren**

  Verschiedene Deskriptoren sind für verschiedene Datenbanken geeignet. Zum Beispiel ist der *GIST*-Deskriptor[2] für Internetdatenbanken mit mindestens 100 000 000 Bildern optimiert, im Gegensatz dazu eignen sich *BoF*-Methoden O'Hara and Draper (2011) nur für Datenbanken mit bis zu 10 000 000 Bildern. Des Weiteren hängt es vom Anwendungsfall ab, welche Deskriptoren in Frage kommen. Bei der Analyse von Röntgenbildern für medizinische Zwecke wäre es zum Beispiel sinnlos einen farbbasierten Deskriptor zu verwenden, da alle Röntgenbilder zum Großteil schwarz sind. Ein eckenbasierter Deskriptor hingegen könnte die Informationen, die auf den Röntgenbildern enthalten sind, viel besser erfassen.

- **Die verwendete Metrik**

  Zu guter Letzt beeinflusst die verwendete Metrik die Query-Ergebnisse. Die Metrik wird zur Erstellung der Rangliste verwendet. Die Wirkung unterschiedlicher Metriken lässt sich gut am zusammengesetzten Deskriptor veranschaulichen, wie sich in der Evaluierung zeigen wird.

Das Ziel dieser Bachelorarbeit ist eine Machbarkeitsstudie (Proof of Concept), die auslotet, wie weit man mit einfachen Mitteln kommt und herausarbeitet, wo man ansetzen muss, um die Ergebnisse deutlich zu verbessern. Hierfür werden die zwei vorgeschlagenen Mechanismen, *Relevance Feedback* und *Indexing* (Kapitel 6 und 5), verwendet und evaluiert. Essentiell für ein CBIR-System sind die erwähnten *Deskriptoren*, die in Kapitel 2 ausführlich behandelt werden. Kapitel 3 beschäftigt sich mit den verwendeten *Metriken*, Details zur Implementation werden in Kapitel 7 vorgestellt. Abschließend werden in Kapitel 8 die Erkenntnisse der Evaluierung zusammengefasst und kommentiert.

Diese Bachelorarbeit wurde auf Initiative der Firma *M-BOX*[3] verfasst, welche uns Testdatenbanken zur Verfügung stellte.

[2] `http://people.csail.mit.edu/torralba/code/spatialenvelope/`
[3] `http://www.m-box.at/`

Kapitel 2

Deskriptoren

Die Realisierung einer inhaltsbasierten Ähnlichkeitssuche erfordert das Vergleichen von Bildern. Für die meisten Menschen ist das keine besonders große Herausforderung, da der „Mensch“ über die derzeit beste „Objekterkennungssoftware“ im Visual-Cortex verfügt. Bilder von Objekten aus derselben Objektkategorie werden vom Menschen sofort als ähnlich erkannt, unabhängig von Beleuchtung, Farbe, Rotation oder Transformation. Für einen Computer jedoch besteht ein Bild nur aus einer Matrix von Farbwerten, also Zahlen. Bilder, deren Inhalt ähnlich ist, haben in der Regel verschiedene Farbwerte.

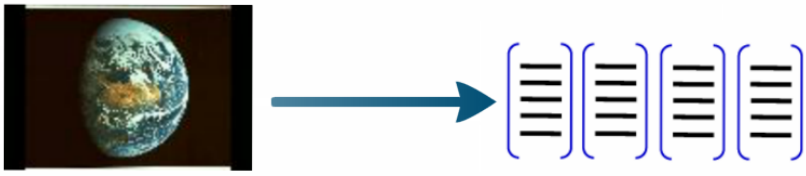


Abbildung 2.1: Transformation von Bild zu Deskriptor-Vektor.

Um Bilder miteinander vergleichen zu können, müssen Bilder mit ähnlichem Inhalt im Bildraum näher beieinander liegen als Bilder mit unterschiedlichem Inhalt. Die Farbwerte müssen in einen Vektorraum transformiert werden, in dem ähnliche Bilder ähnlich sind. Vektoren in diesem Vektorraum heißen *Deskriptoren*. In den Deskriptoren werden meist signifikante Features, die aus den Farbmatrizen der Bilder gewonnen werden, gespeichert. Es gibt vier Arten von Deskriptoren, die auf Farben, Formen, Texturen oder lokalen Features, basieren. Schon die Namensgebung gibt Aufschluss über deren Funktionsweise.

2.1 Typen von Deskriptoren

- **Farbbasierte Deskriptoren:** Als farbbasierte Deskriptoren werden jene bezeichnet, die den Informationsgehalt eines Bildes anhand der vorhandenen Farben extrahieren. Eine einfache Methode einen farbbasierten Deskriptor zu generieren ist, Pixel, die sich in vordefinierten Farbwertintervallen befinden, zu zählen und in einem Vektor zu speichern. Eine andere Möglichkeit ist die räumliche Verteilung der Farben zu extrahieren.
  Beispiele für farbbasierte Deskriptoren sind: *Color Histogram*, *MPEG7-CLD*, *MPEG7-SCD*, *SpCD*.

- **Texturbasierte Deskriptoren:** Ausschlaggebend für Texturen sind unter anderem Kanten. Kanten können zum Beispiel mittels Gaborfiltern oder gradient-basierten Filtern ge-

funden werden. Weil Kanten nichts anderes als Änderungen in der Bildhelligkeit sind, bieten sich gradient-basierte Filter an, diese reagieren nämlich dort, wo die Änderungen in den Farbwerten am stärksten sind.
Beispiele für texturbasierte Deskriptoren sind: *Tamura*, *Global Texture Features*, *Gabor-Histogramm*, *Gabor-Vektor*, *MPEG7-EHD*.

- **Auf lokalen Merkmalen basierende Deskriptoren:** Methoden, die im Bereich der Objekterkennung verwendet werden, können auch für CBIR verwendet werden. Hierbei werden lokale Merkmale eines Bildes extrahiert, die invariant zu perspektivischen Verzerrungen sind. Beispiele dafür sind: *SIFT*, *SURF*.

- **Formbasierte Deskriptoren:** Um translations- und rotationsinvariante Formen zu extrahieren, können verschiedene Filter eingesetzt werden. Formbasierte Deskriptoren sind nicht verbreitet und werden im Kontext mit CBIR nur lokal (SIFT) oder nur sehr grob global eingesetzt.
Beispiele dafür sind: *SIFT*, *MPEG-CLD*.

Ein Überblick über Vertreter der Deskriptorklassen und ein experimenteller Vergleich der einzelnen Deskriptoren wird von Deselaers et al. (2008) präsentiert.

2.2 Kombinierte Deskriptoren

Da sich die meisten Deskriptoren auf einzelne Bildmerkmale beschränken, obwohl in anderen Merkmalen noch zusätzliche Information steckt, ist es naheliegend, verschiedene Deskriptoren zu kombinieren. Im Idealfall verbessern sich die Ergebnisse einer Suchabfrage dahingehend, dass alle guten Resultate der einzelnen Deskriptoren in der Ergebnismenge sind und viele schlechte Resultate im Ranking weiter nach unten wandern. Zu beachten ist jedoch die Tatsache, dass distanzbasierte Systeme bei hochdimensionalen Vektoren an ihre Grenzen stoßen, siehe Kapitel 2.3. Rui and Huang (2000) schlagen zwei verschiedene Möglichkeiten vor, um Deskriptoren miteinander zu kombinieren:

- **Flacher Ansatz:** Beim flachen Ansatz werden die verschiedenen Deskriptoren in einem großen Deskriptorvektor hintereinander abgespeichert. Der somit erhaltene Deskriptor wird wie die anderen Deskriptoren verwendet. Für die Berechnung von Distanzen zwischen zweier so kombinierter Deskriptoren sind keine speziellen Metriken notwendig.

- **Hierarchischer Ansatz:** Beim hierarchischen Ansatz werden die verschiedenen Deskriptoren beibehalten und nur bei der Berechnung der Distanz zweier Bilder mittels einer Kombinationsfunktion kombiniert.

In unserer Arbeit wird der *flache Ansatz* verwendet, da die von uns verwendeten Relevance Feedback- und Indexingmechanismen direkt mit diesem kompatibel sind. Bereits bestehende Beispiele für kombinierte Vektoren sind: *CEDD* und *FCTH*. In Abschnitt 2.4 wird bei der Evaluierung verschiedener Deskriptoren über Kombinationen, die für unsere Arbeit nützlich sind, reflektiert.

Gauß-Normierungen

Da beim *flachen Ansatz* verschiedene Deskriptoren einfach hintereinander abgespeichert werden, fallen große Deskriptoren mehr ins Gewicht als kleine Deskriptoren. Um diesem Problem zu entgehen, muss man den kombinierten Deskriptor normalisieren. Hierfür bietet sich die einfache

Gauß-Normierung an.

In diesem Unterkapitel werden einige Variationen der von Rui et al. (1997) vorgestellten Gauß-Normierung behandelt. Man betrachte die verwendete Deskriptordatenbank als $M \times N$-Matrix $F = f_{ij}$, wobei M die Anzahl von Bildern und N die Länge der Deskriptoren ist. f_{ij} notiert das j-te Feature vom Deskriptor des i-ten Bildes in der Datenbank. Unter der Annahme, dass die Einträge der einzelnen Featurespalten F_j normalverteilt sind, kann der Erwartungswert m_j und die Standardabweichung σ_j für jedes Feature berechnet werden.

- **Normierung auf den Bereich [-1,1]:**

$$f'_{ij} = \frac{f_{ij} - m_j}{3 \cdot \sigma_j} \tag{2.1}$$

  f'_{ij} bezeichnet dabei die normierte Form von f_{ij}. Interessanter als die eigentlichen Werte der jeweiligen Features ist deren Abweichung zum Erwartungswert $f_{ij} - m_j$. Diese wird mit der Division durch die Standardabweichung σ_j der Spalte j normiert. Bei der Division durch die Standardabweichung werden die Unterschiede der Werte in Spalten mit hoher Standardabweichung reduziert und die Unterschiede der Werte in Spalten mit niedriger Standardabweichung erhöht. Laut Rui et al. (1997) werden die Werte der Deskriptoren mit der Normierung nach Formel 2.1 mit 99%-iger Wahrscheinlichkeit auf den Wertebereich [-1,1] normiert.

- **Normierung auf den Bereich [0,1]:**

$$f''_{ij} = \frac{f'_{ij} + 1}{2} \tag{2.2}$$

  Die Normierung auf den Wertebereich $[0, 1]$ erfordert keine groben Veränderungen. Durch die Addition von 1 zur vorherigen Normierung nach Formel 2.1 erhält man einen Wertebereich von $[0, 2]$, welchen man mit der Divison durch 2 auf einen Wertebereich von $[0, 1]$ stauchen kann.

- **Normierung mittels Varianz:**

$$f'_{ij} = \frac{f_{ij} - m_j}{\sigma_j^2} \tag{2.3}$$

  Möchte man sich den Berechnungsaufwand des Wurzelziehens für die Standardabweichung sparen, so kann anstelle der Standardabweichung durch die Varianz σ_j^2 der jeweiligen Spalten dividiert werden. Die Varianz verhält sich ähnlich wie die Standardabweichung, die Werte in den Spalten mit hoher Varianz werden wieder gestaucht und die Werte in den Spalten mit niedriger Varianz gestreckt.

Bei allen Varianten der Gauß-Normierung ist zu beachten, dass die normierten Deskriptorvektoren nach einer gewissen Zeit, wenn einige neue Bilder in die Datenbank eingefügt wurden, aktualisiert werden sollten.

2.3 Der Fluch der Dimensionalität

Wie schon erwähnt stoßen distanzbasierte Verfahren bei zu hoch dimensionalisierten Deskriptoren an ihre Grenzen. Dieses Phänomen wird als der Fluch der Dimensionalität bezeichnet. Der Fluch der Dimensionalität beschreibt negative Effekte, die beim Einsatz von hochdimensionalen

Daten auftreten. Ursache für diese Effekte ist die Tatsache, dass beim Hinzufügen von weiteren Dimensionen das Gesamtvolumen rapide ansteigt und die verfügbaren Daten dadurch extrem zerstreut werden.

Bei steigender Dimension werden die Differenzen der Distanzen von verschiedenen Punkten beliebig klein, so dass die Distanz zwischen zwei Punkten immer weniger Aussagekraft besitzt (Kevin Beyer and Shaft (1999)). Dies lässt sich mit folgender Formel ausdrücken:

$$\lim_{dim \to \infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0 \tag{2.4}$$

Die Differenz zwischen maximaler und minimaler Distanz geteilt durch die minimale Distanz konvergiert bei steigender Dimension gegen Null.

2.4 Auswahl von relevanten Deskriptoren für CBIR-Systeme

Da CBIR-Systeme möglichst gute Ergebnisse liefern sollen, ist es essentiell, Deskriptoren zu verwenden, die gute Suchergebnisse liefern. Da jeder Deskriptor neben seinen Stärken auch Schwächen besitzt, ist es naheliegend, Deskriptoren zu kombinieren, um einen idealen Deskriptor ohne oder mit sehr wenigen Schwächen zu erhalten. Ein wesentlicher Bestandteil bei der Suche von Stärken und Schwächen ist es, die Deskriptoren mittels Experimenten zu evaluieren.

In unseren Experimenten wurden zehn Deskriptoren getestet (CEDD, FCTH, SpCD, MPEG7-CLD, MPEG7-EHD, MPEG7-SCD, Tamura, Color Histogram, Gabor Histogram und Global Texture Features). Es wurde mit vier von *M-BOX* bereitgestellten Datenbanken getestet (Lentos, Madias, OeMV, Cars), aus jeder Datenbank wurden vier Bilder ausgewählt, die als Querybild benutzt wurden. Nach einer Suchiteration mit n Ergebnissen wurden die Bilder subjektiv als „gutes Ergebnis“ oder als „schlechtes Ergebnis“ markiert. Die Präzision der Deskriptoren wurde wie folgt berechnet:

$$\frac{\#guteErgebnisse}{n} * 100 \tag{2.5}$$

Die einzelnen Testkonfigurationen und -ergebnisse werden im Kapitel 2.4.1 behandelt, während die Gesamtergebnisse und die Auswahl der Deskriptoren für das CBIR-System in Kapitel 2.4.2 beleuchtet wird. Die Extraktion der Featurevektoren und die anschließenden Tests wurden mit den Open-Source-Tools *img(Rummager)*[1] und *FIRE*[2] durchgeführt.

2.4.1 Testkonfiguration und -ergebnisse

- Die Datenbank *Lentos* besteht aus Ausstellungstücken eines Kunstmuseums, hauptsächlich aus Zeichnungen und abstrakten Bildern. Verwendet wurden die Querybilder aus Abbildung 2.2 mit den folgenden Suchkriterien:
  - **lentos-drawing:** ähnliche Zeichnungen (schwarz auf weiß)
  - **lentos-box:** ähnliche Boxen oder Holzfiguren

[1] `http://chatzichristofis.info/?page_id=213`
[2] `http://thomas.deselaers.de/fire/`

  - **lentos-cartoon:** Cartoonfiguren und Menschen
  - **lentos-group:** einzelne oder Gruppen von Menschen

  Die Resultate der Tests zeigt Abbildung 2.6.

- Die Datenbank *Madias* beinhaltet größtenteils historische Fotos und Illustrationen und Bilder aus der Natur. Verwendet wurden die Querybilder aus Abbildung 2.3 mit den folgenden Suchkriterien:

  - **madias-building:** ähnliche Bilder von Gebäuden
  - **madias-pinkbg:** ähnliche Illustrationen
  - **madias-tree:** Suchkriterien: Bilder/Illustrationen mit Bäumen und Personen
  - **madias-warrior:** Suchkriterien: ähnliche Krieger bzw. Personen

  Abbildung 2.7 zeigt die Ergebnisse des Tests.

- Die *OeMV*-Datenbank beinhaltet historische Ausstellungsstücke des *Österreichischen Museums für Volkskunde, Wien.*

  Verwendet wurden die Querybilder aus Abbildung 2.4 mit den folgenden Suchkriterien:

  - **oemv-image:** Gegenstände mit ähnlicher Verzierung bzw. Textur
  - **oemv-knife:** Messer mit Klinge
  - **oemv-sheep:** Holzfiguren von Tieren
  - **oemv-woman:** Zeichnungen von Personen

  Abbildung 2.8 zeigt die Auswertung der Suchanfragen.

- Die *Cars*-Datenbank enthält 6756 Bilder, deren Inhalt meistens mit Autosport in Beziehung steht.

  Verwendet wurden die Querybilder aus Abbildung 2.5 mit den folgenden Suchkriterien:

  - **cars-red:** rote Autos
  - **cars-side:** Autos beliebiger Farbe, die von der Seite fotografiert wurden
  - **cars-wheel:** Innenräume von Autos mit Lenkrad.
  - **cars-multiple:** Bilder mit mehr als einem Auto.

  Abbildung 2.9 zeigt die Ergebnisse des Tests.

2.4.2 Schlussfolgerung und Auswahl

Die Gesamtpräzision, also der Mittelwert aller Präzisionen der einzelnen 12 Tests, zeigt einen deutlichen Trend: farbbasierte Deskriptoren sind im Gesamtranking viel weiter oben anzutreffen als der Rest. Dieses Ergebnis überrascht nicht, da dieser Test von der subjektiven Meinung des Benutzers abhängt. Obwohl die Querybilder so gewählt wurden, dass alle Merkmale eines Bildes (Farben, Form, Textur) einigermaßen gleichwertig verteilt sind, reagiert der Mensch besser auf Farben als auf andere Merkmale. Der Fokus auf Farben ist jedoch nicht immer sinnvoll, da die Farben auf Bildern stark von der Beleuchtung und von einem willkürlich gewählten Hintergrund

Abbildung 2.2: Querybilder *lentos-drawing*, *lentos-box*, *lentos-cartoon* und *lentos-group* der Datenbank *Lentos*. Anm.: Wegen ungeklärter Lizenzierung wird bei *lentos-drawing* nur eine Karikatur angezeigt, die dem Bild ähnlich schaut.

Abbildung 2.3: Querybilder *madias-warrior*, *madias-pinkbg*, *madias-building* und *madias-tree* der Datenbank *Madias*

Abbildung 2.4: Querybilder *oemv-knife*, *oemv-woman*, *oemv-sheep* und *oemv-image* der Datenbank *OeMV*

Abbildung 2.5: Querybilder *cars-wheel*, *cars-multiple* und *cars-red/cars-side* der Cars-Datenbank

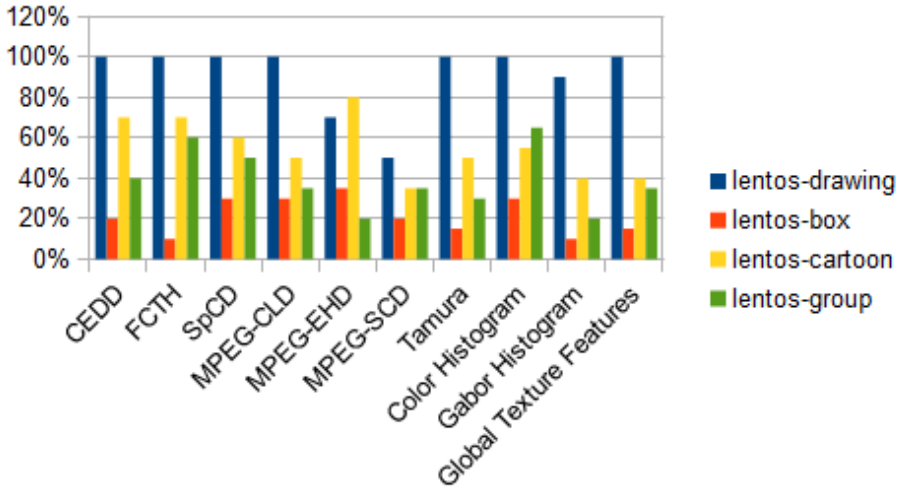


Abbildung 2.6: Präzision der getesteten Deskriptoren in % (Testdatenbank Lentos)

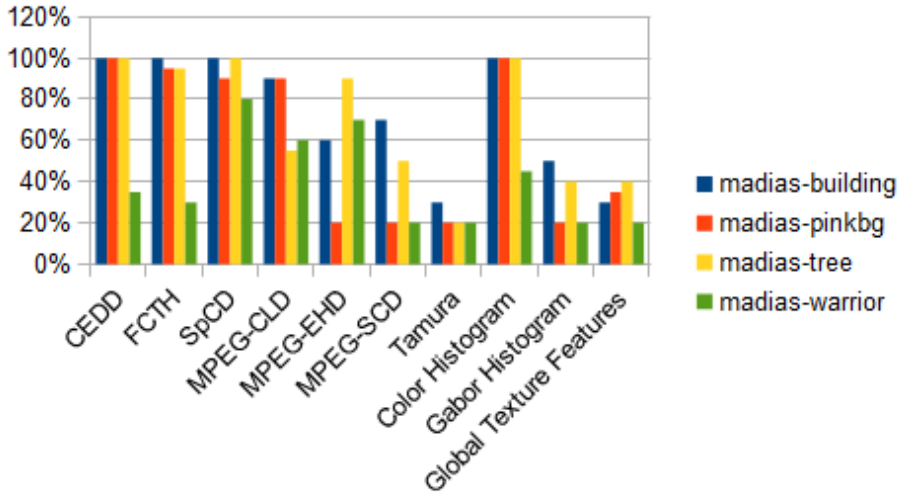


Abbildung 2.7: Präzision der getesteten Deskriptoren in % (Testdatenbank Madias)

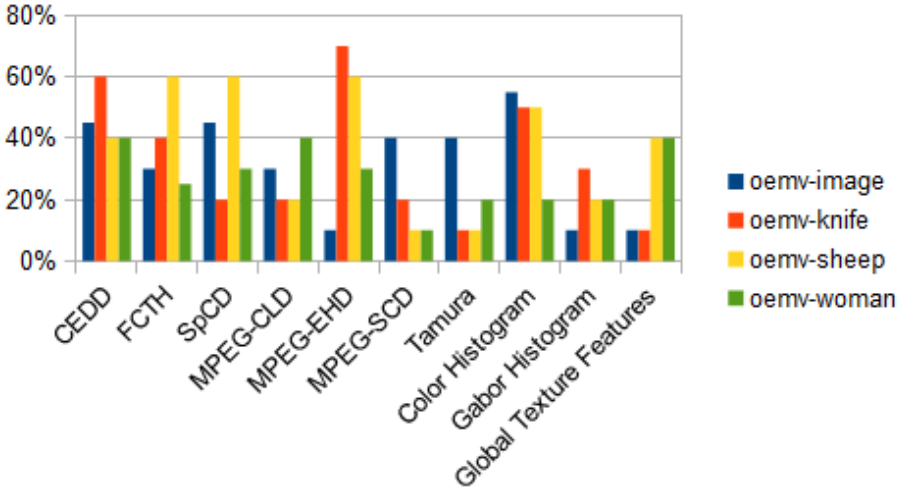


Abbildung 2.8: Präzision der getesteten Deskriptoren in % (Testdatenbank OeMV)

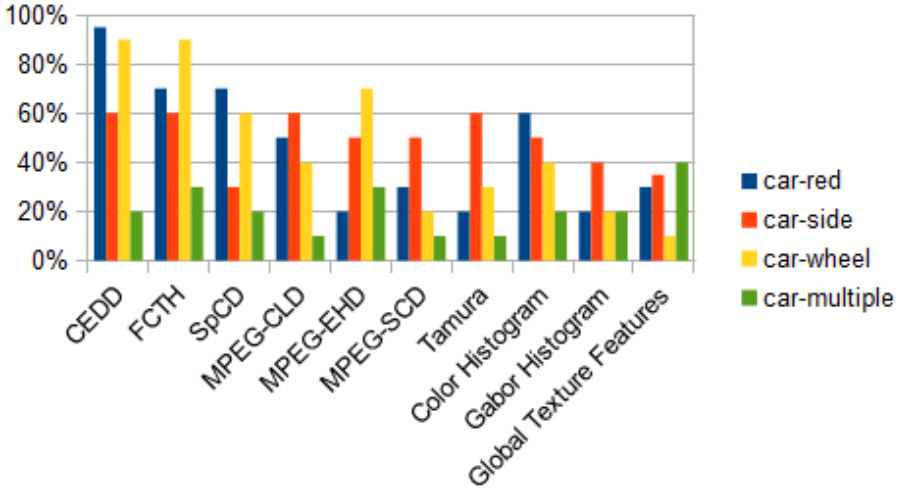


Abbildung 2.9: Präzision der getesteten Deskriptoren in % (Testdatenbank Cars)

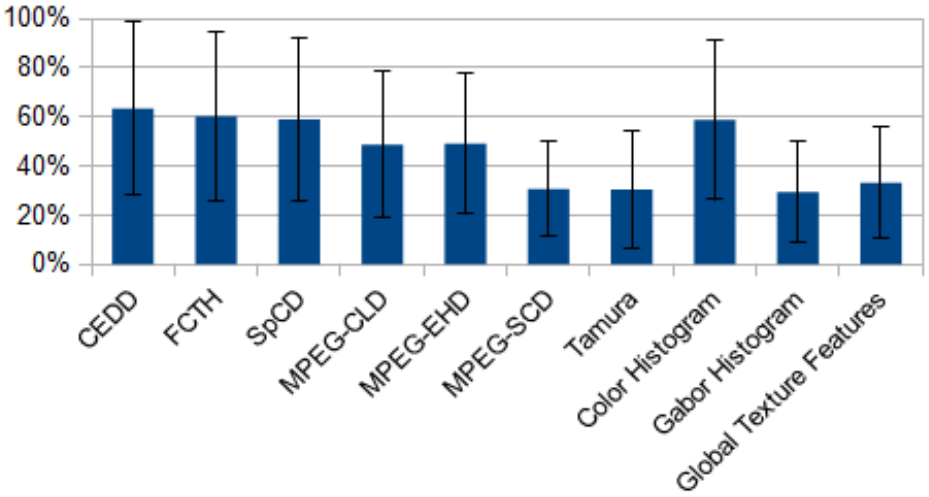


Abbildung 2.10: Gesamtpräzision der getesteten Deskriptoren in % (mit Fehlerbalken)

abhängen.

Abbildung 2.10 zeigt das genaue Ergebnis der empirischen Tests. In den Tests schnitten CEDD (63%), FCTH (60%), Color Histogram (59%) und SpCD (59%) am besten ab. Aufgrund der Tatsache, dass Color Histogram einfacher gestaltet ist als die anderen Deskriptoren (SpCD berücksichtigt neben den Farben auch die räumliche Verteilung, CEDD und FCTH sind auch texturbasiert), entschieden wir uns, Color Histogram in unserem CBIR-System zu verwenden. Obwohl Color Histogram bei den meisten Queries gute Ergebnisse liefert, zeigt der Deskriptor bei einigen Bildern Schwächen. Die schwachen Ergebnisse resultieren meistens aus einem unnatürlichem und einheitlichem Hintergrund, welcher das Histogramm negativ beeinflusst. Ein Beispiel: Die Query *oemv-knife* (Abbildung 2.4) besitzt einen grauen Hintergrund, was zur Folge hat, dass Messer auf weißem Hintergrund (welche in der Datenbank existent sind) nicht gefunden werden.

Um diesem Problem zu entgehen, bietet sich der Deskriptor MPEG7-EHD als Kombination mit Color Histogram an, da er in unseren Tests immer dort gut abgeschnitten hatte, wo Color Histogram nicht überzeugen konnte. MPEG7-EHD ist ein texturbasierter Deskriptor, welcher die räumliche Verteilung von Kanten darstellt. Unnatürliche und einheitliche Hintergründe besitzen keine Kanten, was gute Ergebnisse zur Folge hat, da wirklich nur die Kanten des gewünschten Objekts berücksichtigt werden.

Color Histogram und MPEG7-EHD liefern zusammen betrachtet bei jeder getesteter Query sehr gute Ergebnisse. Kombiniert liefern sie sogar noch bessere Ergebnisse, wie Kapitel 4.1 zeigt. Interessant ist auch ein Vergleich zum kombinierten Deskriptor CEDD, welcher für den texturbasierten Teil auch MPEG7-EHD verwendet. Aus diesem Grund wird der Deskriptor auch in unserem CBIR-System berücksichtigt.

Color Histogram, MPEG7-EHD und CEDD werden im folgenden Kapitel im Detail vorgestellt.

2.5 Verwendete Deskriptoren

In dieser Sektion werden die Deskriptoren vorgestellt, die für unser CBIR-System verwendet werden (Color Histogram, MPEG7-EHD und CEDD). Open-Source-Implementierungen der Deskriptoren bietet das auf Java basierende *LIRE*-Projekt (Lux and Chatzichristofis (2008))[3].

2.5.1 Color Histogram

Color Histogram erfasst die Farbverteilung eines Bildes. Die Anzahl der Vorkommen verschiedener Farbwerte werden in einem Histogramm-Vektor, dessen verschiedene Stellen für verschiedene Farbwertebereiche stehen, erfasst. Obwohl bei der Verwendung von Color Histogram als Deskriptor jegliche Informationen über Ecken und Formen ignoriert werden, wurden auf den von uns getesteten Datenbanken mit dem Color Histogram gute, nachvollziehbare Ergebnisse erzielt.

2.5.2 MPEG7-EHD

EHD (Edge Histogram Descriptor) ist ein texturbasierter Deskriptor, der auf dem MPEG7-Standard (ISO/IEC 15938) aufbaut. Der Deskriptor stellt die räumliche Verteilung von 5 Arten

[3] `http://www.semanticmetadata.net/lire/`

von Kanten dar: vertikale, horizontale, 45-Grad-gerichtete, 135-Grad-gerichtete und ungerichtete Kanten. Das Bild wird hierbei in kleinere Teilstücke unterteilt, aus denen Histogramme beruhend auf den 5 Kantenarten aufgebaut werden. Shoukry et al. (2010) zeigen eine genaue Beschreibung des Extraktionsalgorithmus und die Visualisierung einer Extraktion.

2.5.3 CEDD

CEDD (Color and Edge Directivity Descriptor) von Chatzichristofis and Boutalis (2008) ist ein Deskriptor, der Farben- und Texturinformationen von Bildern in einem 144-bin Histogramm speichert. Die Farbinformation wird durch zweimalige Anwendung von verschiedenen Fuzzy-Regeln im HSV-Farbraum extrahiert und mit der Texturinformation kombiniert, die mittels MPEG7-EHD extrahiert wird. Chatzichristofis and Boutalis (2008) liefern nähere Informationen zur Extraktion und zur Implementierungsstrategie von CEDD.

Kapitel 3

Metriken

Im vorherigen Kapitel wurde das Konzept von Deskriptoren vorgestellt. Deskriptoren werden nur aus einem Grund verwendet, nämlich um Bilder vergleichbar zu machen. Um die Ähnlichkeit zweier Deskriptoren festzustellen, kann eine beliebige Distanzmetrik verwendet werden.

3.1 Euklidische Metriken

Eine gute Übersicht der Variationen der euklidischen Metriken wird von Rui and Huang (2000) vorgestellt.

3.1.1 Die Basisform

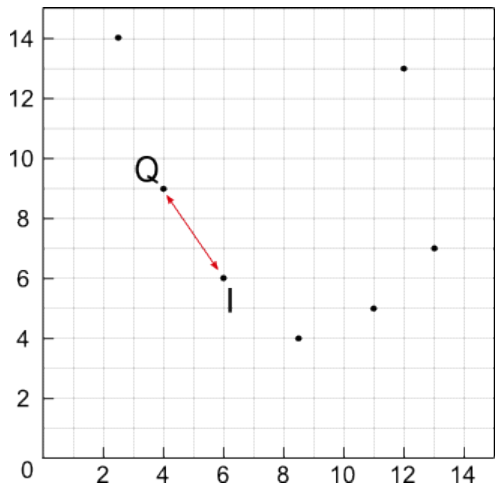


Abbildung 3.1: Euklidische Distanz zweier Punkte im 2-dimensionalen Raum.

Die einfachste Möglichkeit eine Rangliste von Bildern bezüglich eines Querybildes zu erstellen ist es für jedes Bild in der Datenbank die euklidische Distanz zum Querybild zu berechnen. Die euklidische Distanz zweier Bilder entspricht der Vektordistanz im euklidischem Vektorraum.

$$d(\mathbf{Q}, \mathbf{I}) = \sqrt{(\mathbf{Q} - \mathbf{I})^T (\mathbf{Q} - \mathbf{I})} \tag{3.1}$$

$d(\mathbf{Q}, \mathbf{I})$ notiert die Distanz zwischen dem Query-Deskriptorvektor $\mathbf{Q}$ und dem Deskriptorvektor $\mathbf{I}$ eines beliebigen Bildes aus der Datenbank.

3.1.2 Die gewichtete euklidische Metrik

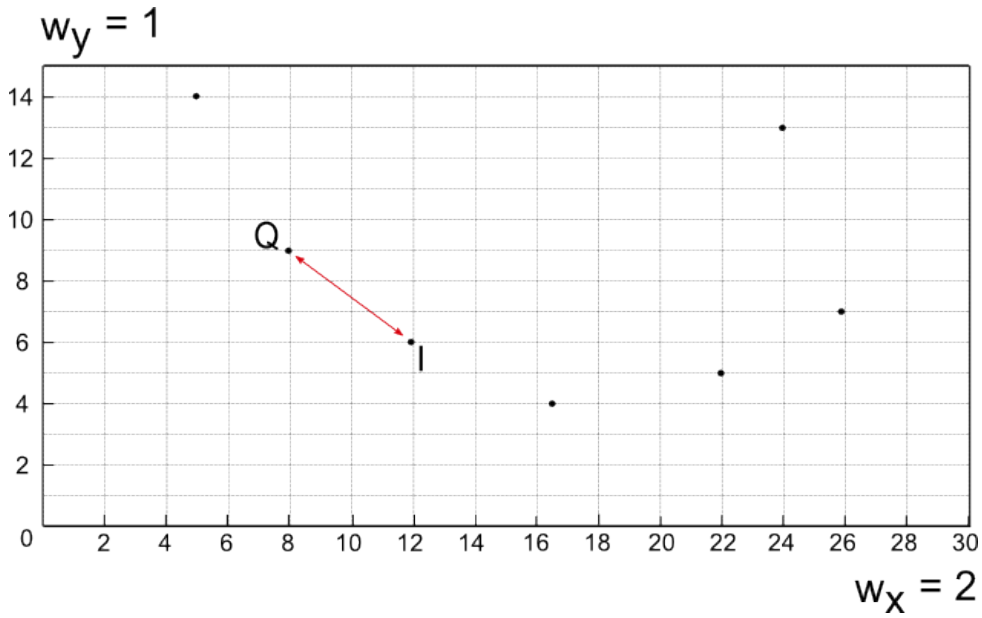


Abbildung 3.2: Gewichtete euklidische Distanz zweier Punkte im 2-dimensionalen Raum.

Die gewichtete euklidische Metrik ist eine Erweiterung der euklidischen Metrik, die es erlaubt jede Achse des Koordinatensystems beliebig zu skalieren. Die Wichtigkeit einzelner Features kann durch die Gewichtung deren Koordinatenachse reguliert werden.

$$d(\mathbf{Q}, \mathbf{I}) = \sqrt{(\mathbf{Q} - \mathbf{I})^T W (\mathbf{Q} - \mathbf{I})} \tag{3.2}$$

$d(\mathbf{Q}, \mathbf{I})$ notiert die Distanz zwischen dem Query-Deskriptorvektor $\mathbf{Q}$ und dem Deskriptorvektor $\mathbf{I}$ eines beliebigen Bildes aus der Datenbank. W ist eine Diagonalmatrix, deren Einträge in der Diagonale den Gewichten der jeweiligen Features entsprechen. W_{ii} ist das Gewicht für das i-te Feature von $\mathbf{Q}$ und $\mathbf{I}$.

3.2 Kosinus-Metriken

3.2.1 Kosinus-Ähnlichkeit

Im Gegensatz zu den euklidischen Metriken handelt es sich hierbei um eine Ähnlichkeitsfunktion und nicht um eine Distanzfunktion. Wie der Name der Kosinus-Metrik bereits aufzeigt, hat die Berechnung der Ähnlichkeit zwischen den Deskriptorvektoren der zu vergleichenden Bildern etwas mit dem Kosinus zu tun. Zwei Bilder sind genau dann ähnlich, wenn der Winkel zwischen den beiden Vektoren klein ist, also wenn der Kosinus des Winkels groß ist.

$$sim(\mathbf{Q}, \mathbf{I}) = cos(\mathbf{Q}, \mathbf{I}) = \frac{\mathbf{Q} \cdot \mathbf{I}}{||\mathbf{Q}|| \cdot ||\mathbf{I}||} \tag{3.3}$$

$sim(\mathbf{Q}, \mathbf{I})$ notiert die Ähnlichkeit zwischen dem Deskriptorvektor des Querybildes $\mathbf{Q}$ und dem Deskriptorvektor eines Bildes aus der Datenbank $\mathbf{I}$.

Unter der Voraussetzung, dass alle Deskriptoren nur positive Features enthalten, liefert diese Ähnlichkeitsfunktion Werte im Intervall $[0, 1]$. Diese Eigenschaft ist für die Verwendung dieser

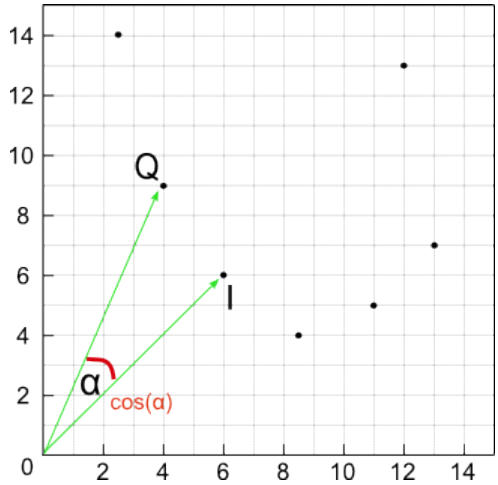


Abbildung 3.3: Kosinus-Ähnlichkeit im 2-dimensionalen Raum.

Metrik in CBIR-Frameworks von Nutzen, die zur Erstellung einer Rangliste Relevanzwahrscheinlichkeiten anstatt Distanzfunktionen verwenden.

3.2.2 Gewichtete Kosinus-Ähnlichkeit

Als gewichtete Kosinus-Metrik bezeichnen wir eine kleine Abwandlung der Kosinus-Metrik bei der ähnlich wie bei der gewichteten euklidischen Metrik, die einzelnen Features mit Gewichten versehen werden.

$$sim(\mathbf{Q}, \mathbf{I}) = \frac{W\mathbf{Q} \cdot \mathbf{I}}{||W\mathbf{Q}|| \cdot ||W\mathbf{I}||} \tag{3.4}$$

$sim(\mathbf{Q}, \mathbf{I})$ notiert wieder die Ähnlichkeit zwischen dem Deskriptorvektor des Querybildes $\mathbf{Q}$ und dem Deskriptorvektor eines Bildes aus der Datenbank $\mathbf{I}$. W ist eine Diagonalmatrix, deren Einträge in der Diagonale den Gewichten der jeweiligen Features entsprechen. W_{ii} ist das Gewicht für das i-te Feature von $\mathbf{Q}$ und $\mathbf{I}$.

3.2.3 Kosinus-Distanz

Um eine Ähnlichkeitsfunktion zu einer Distanzfunktion umzuwandeln, gibt es mehrere Möglichkeiten. Die Distanzfunktion soll informell folgende Eigenschaften erfüllen:

- Ist die Ähnlichkeit zwischen zwei Bildern groß, so ist die Distanz zwischen den gleichen Bildern klein.
- Ist die Ähnlichkeit zwischen zwei Bildern klein, so ist die Distanz zwischen den gleichen Bildern groß.

Die Inverse der Ähnlichkeitsfunktion erfüllt diese Eigenschaften per Definition:

$$d(\mathbf{Q}, \mathbf{I}) = \frac{1}{sim(\mathbf{Q}, \mathbf{I}) + \epsilon} \tag{3.5}$$

Die Distanz der Deskriptorvektoren $\mathbf{Q}$ und $\mathbf{I}$ ist die Inverse derer Ähnlichkeit. Da zwei Bilder auch einen Ähnlichkeitswert von Null haben können, benötigt man einen kleinen Offset ϵ, um eine Division durch Null zu vermeiden. Wir verwenden für unser ϵ zum Beispiel 10^{-7}.

Im Falle der Kosinus-Ähnlichkeit gibt es aber noch eine andere praktische Variante einer Distanzfunktion.

$$d(\mathbf{Q}, \mathbf{I}) = 1 - sim(\mathbf{Q}, \mathbf{I}) \tag{3.6}$$

Damit diese Version der Kosinus-Distanzfunktion gültig ist, dürfen alle verwendeten Deskriptoren nur positive Einträge enthalten. Bei positiven Deskriptoren liegen alle Ähnlichkeitswerte im Intervall $[0, 1]$. Zieht man also den Ähnlichkeitswert von 1 ab, so erhält man wiederum eine Funktion, die die beiden Eigenschaften der Distanzfunktion erfüllt. Diese Variante ist praktisch, weil die Distanzfunktion wieder Werte im Bereich $[0, 1]$ liefert.

3.3 Earth Mover's Distance

Die *Earth Mover's Distance* ist eine sehr populäre Metrik für CBIR, die im Gegensatz zu den erwähnten Metriken mehrdimensionale Signaturen vergleicht. Bei der Berechnung dieser Distanz werden die zu vergleichenden Signaturen als Größen eines Erdbewegungsvorganges mit Hügeln und Löchern angesehen. Ziel ist es, die Hügel auf die Löcher mit möglichst wenig Aufwand so zu verteilen, so dass eine Ebene entsteht. Die Distanz berechnet sich schließlich aus der beförderten Masse pro zurückgelegter Distanz. Rubner et al. (2000) beleuchten die *Earth Mover's Distance* für IR im Detail.

Da unser CBIR-System eindimensionale Histogramme verwendet, ist die *Earth Mover's Distance* in unserem Fall mit der *Manhattan-Metrik* gleichzustellen (Rubner et al. (2000)), die die Differenz der einzelnen Bins der Histogramme aufaddiert:

$$d(\mathbf{Q}, \mathbf{I}) = ||\mathbf{Q} - \mathbf{I}|| \tag{3.7}$$

Die Manhattan-Metrik wird auch *L1-Distanz* genannt.

Kapitel 4

Evaluierung der verwendeten Deskriptoren und Metriken

4.1 Evaluierung des kombinierten Deskriptors

Wie in Kapitel 2.4.2 festgestellt wurde, liefern Color Histogram und MPEG7-EHD zusammen betrachtet gute Ergebnisse bei jeder getesteter Query. Im Folgenden wurden die zwei Deskriptoren zu einem gemeinsamen Deskriptor kombiniert und getestet. Test 4.1.1 bezieht sich auf die Datenbanken und Queries aus Kapitel 2.4, während in Kapitel 4.2 ein automatischer Test mit einer annotierten Datenbank durchgeführt wird.

4.1.1 Testkonfiguration und -ergebnisse

In diesem Test wurde der kombinierte Deskriptor mit den 16 Querybildern und den jeweiligen Bewertungskriterien aus 2.4 getestet. Hierfür wurden je fünf Metriken (euklidisch, gewichtet euklidisch, Kosinus, gewichteter Kosinus und Manhattan) in Kombination mit zwei Normalisierungen (Gaußnormalisierung und Varianznormalisierung, Formeln 2.1 bzw. 2.3), die in Kapitel 3 im Detail vorgestellt werden, verwendet. Die Tabellen 4.1, 4.2 und 4.3 spiegeln die Testergebnisse im Vergleich zu Color Histogram, MPEG7-EHD und CEDD wider.

Der Gesamteindruck des kombinierten Deskriptors im Vergleich zu den anderen ist sehr positiv. Beim Vergleich der Normalisierungen setzte sich die Gaußnormalisierung klar gegen die Varianznormalisierung durch, während bei den Metriken die beiden gewichteten Varianten am besten abschnitten, siehe Abbildung 4.5.

Bei vielen Queries werden die Schwächen der einzelnen Deskriptoren durch die Stärken ausgeglichen, was sich auf die Suchergebnisse auf unterschiedliche Weise auswirkt. Einerseits kann es zu einer Verbesserung der Ergebnisse führen, wie z.B bei der Query *oemv-woman*, wo die Präzision von den anfänglichen 20% (Color Histogram) bzw. 30% (MPEG7-EHD) stark gestiegen ist (80%). Andererseits kann es auch zu einer Verschlechterung der Ergebnisse kommen, bei der Query *madias-pinkbg* sinkt die Präzision von 100% (Color Histogram) auf 80% aufgrund des sehr schlechten Ergebnisses von MPEG7-EHD (20%).
Insgesamt überwiegen jedoch die Verbesserungen, sodass der kombinierte Deskriptor auf eine Gesamtpräzision von 67% kommt. Im Vergleich ist der Deskriptor besser als die bisherigen bestplazierten Deskriptoren. Abbildung 4.5 zeigt die Ergebnisse im Detail.

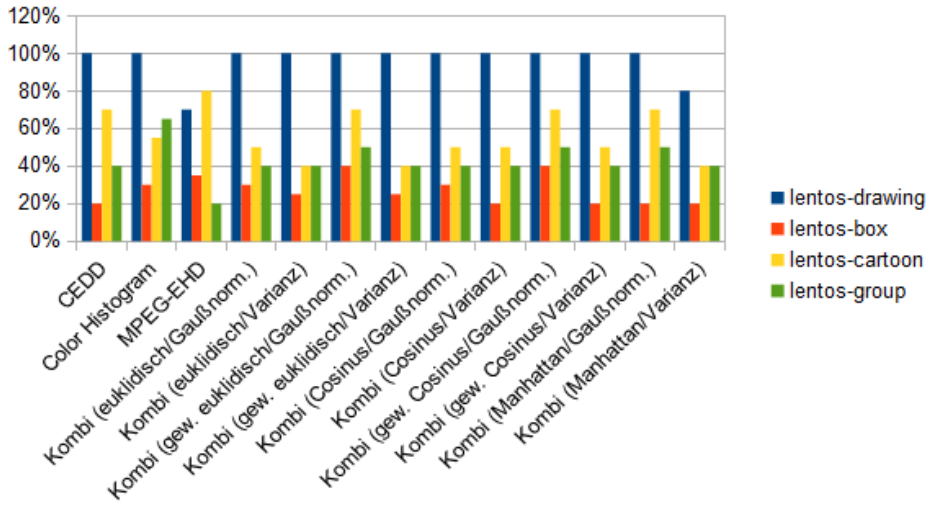


Abbildung 4.1: Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Lentos)

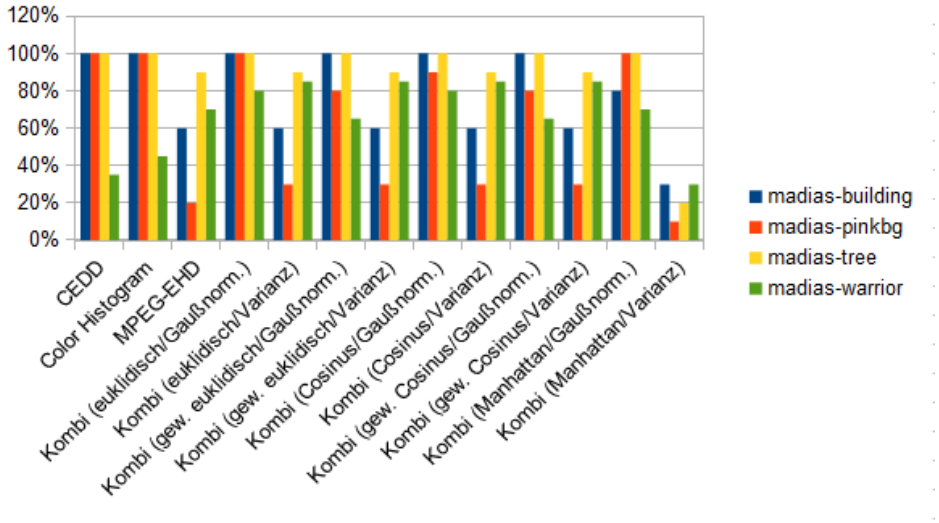


Abbildung 4.2: Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Madias)

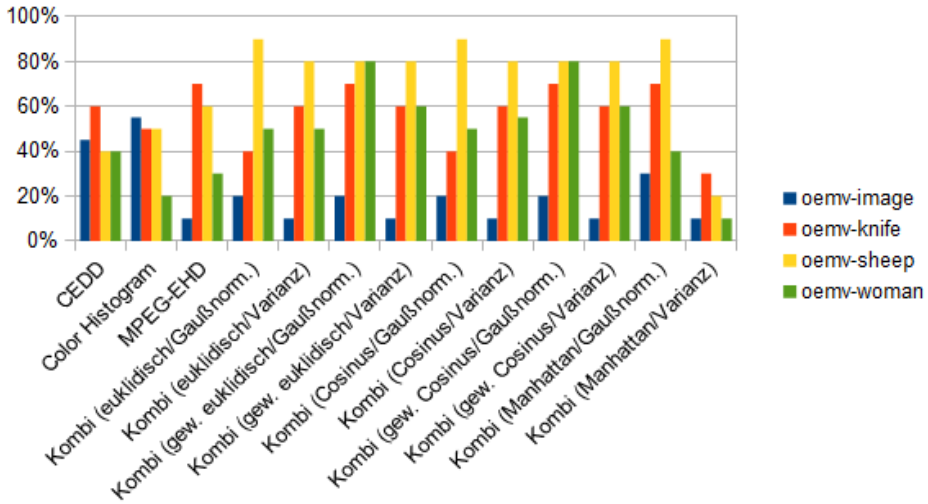


Abbildung 4.3: Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank OeMV)

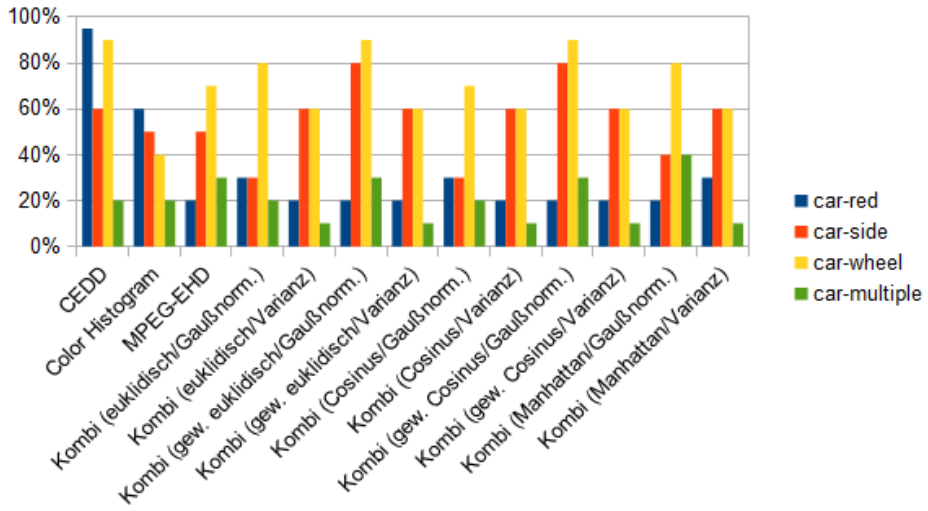


Abbildung 4.4: Präzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (Testdatenbank Cars)

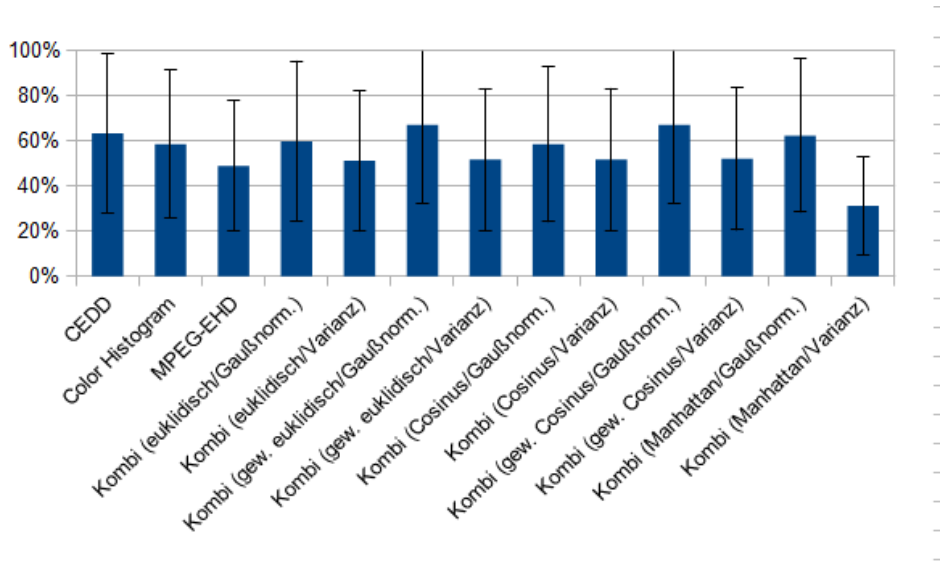


Abbildung 4.5: Gesamtpräzision des kombinierten Deskriptors mit allen Metrik/Normalisierung-Varianten in % im Vergleich zu CEDD, MPEG7-EHD und Color Histogram (mit Fehlerbalken)

4.2 Automatisierter Test mit annotierten Bildern

Die bisherigen Tests waren auf die subjektive Meinung des Benutzers angewiesen - verschiedene Benutzer können dieselben Suchergebnisse einer Query unterschiedlich bewerten, was die Auswertung der Tests beeinflusst. Um diese Subjektivität in den Tests zu minimieren, müssen also eine vollautomatische Suche und Bewertung durchgeführt werden. Da *image retrieval* in vielen Fällen auch als ein Klassifizierungsproblem angesehen werden kann, bietet sich eine klassenbasierte Datenbank als Testdatenbank für einen automatischen Test an. Jedes Bild einer solchen Datenbank besitzt als Annotation die Klasse, der sie angehört. Bilder, die bei einer Suche gefunden werden, werden nur dann als positiv bewertet, wenn sie der gleichen Klasse wie das Querybild angehören. Diese Vorgehensweise erlaubt vollautomatische Tests, welche aufgrund des höheren Umfangs und dem Ausschluss der Subjektivität eine objektivere Alternative zu subjektiven Tests darstellt. Andererseits können vordefinierte Klassen nicht mit individuellen Benutzersuchkriterien gleichgesetzt werden, sodass diese Tests auch nur eine eingeschränkte Aussagekraft besitzen.

Im folgenden Test wurde eine auf CBIR angepasste *Corel*-Datenbank[1] verwendet. Sie besteht aus insgesamt 10 800 Bildern, die wiederum in 80 Klassen eingeteilt wurden. Jede Klasse besitzt mindestens 100 Bilder. Jeder Deskriptor durchlief fünf Testdurchgänge mit 500 zufällig gewählten Querybildern, woraus dann eine durchschnittliche Präzision ermittelt wurde. Außerdem wurden verschiedene Größen der Ergebnismenge getestet, die sich erheblich auf die Präzision auswirken. Abbildung 4.6 zeigt die Ergebisse der Tests für die Ergebnismengengrößen 10, 25 und 50. Der kombinierte Deskriptor wurde mit der euklidischen Metrik und der gewichteten Gauß-Normierung getestet.

Die Testergebnisse zeigen den erwarteten Effekt: Die Kombination von zwei Deskriptoren

[1] `https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval`

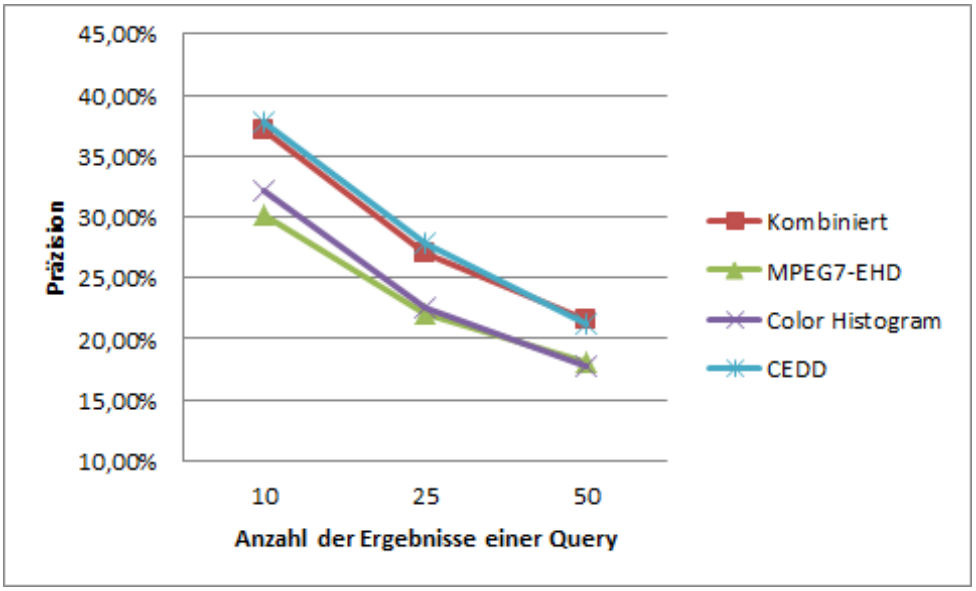


Abbildung 4.6: Präzision des kombinierten Deskriptors im Vergleich zu Color Histogram, MPEG7-EHD und CEDD

verbessert die Präzision im Gegensatz zu den nicht kombinierten. Die Kombination von Color Histogram und MPEG7-EHD erreicht um 4% bis 7% bessere Ergebnisse als die Deskriptoren im Einzelnen. Interessant ist auch der Vergleich zum ähnlich kombinierten Deskriptor CEDD. Trotz der einfachen, flachen Kombinationsmethode erzielt der kombinierte Deskriptor ähnlich gute Ergebnisse wie CEDD. Bei 10 Ergebnissen kommt der kombinierte Deskriptor auf eine durchschnittliche Präzision von 37,1%, während CEDD auf 37,7% kommt.

Die große Schwäche des flach kombinierten Deskriptors verbirgt sich in der hohen Berechnungszeit. Abbildung 4.7 zeigt die durchschnittlich benötigte Zeit für eine Query[2] mit den jeweiligen Deskriptoren. Der kombinierte Deskriptor benötigte in den Tests im Durchschnitt 257 ms für eine Query mit 10 Ergebnissen, ungefähr gleich lang wie Color Histogram und MPEG7-EHD zusammen (217 bzw. 44 ms). CEDD schnitt mit 65 ms viel besser ab als der kombinierte Deskriptor.

Die Testergebnisse zeigen, dass eine einfache Kombination von Deskriptoren mittels der flachen Methode eine Präzisionsverbesserung mit sich bringt, jedoch auf Kosten einer erhöhten Berechnungszeit geht. Speziell entwickelte kombinierte Deskriptoren wie CEDD sollten bevorzugt werden, da sie aufgrund der geringeren Anzahl von Stellen schneller sind und dennoch ähnlich gute Ergebnisse liefern.

[2]getestet mit einem Prozessor mit 2,53 GHz

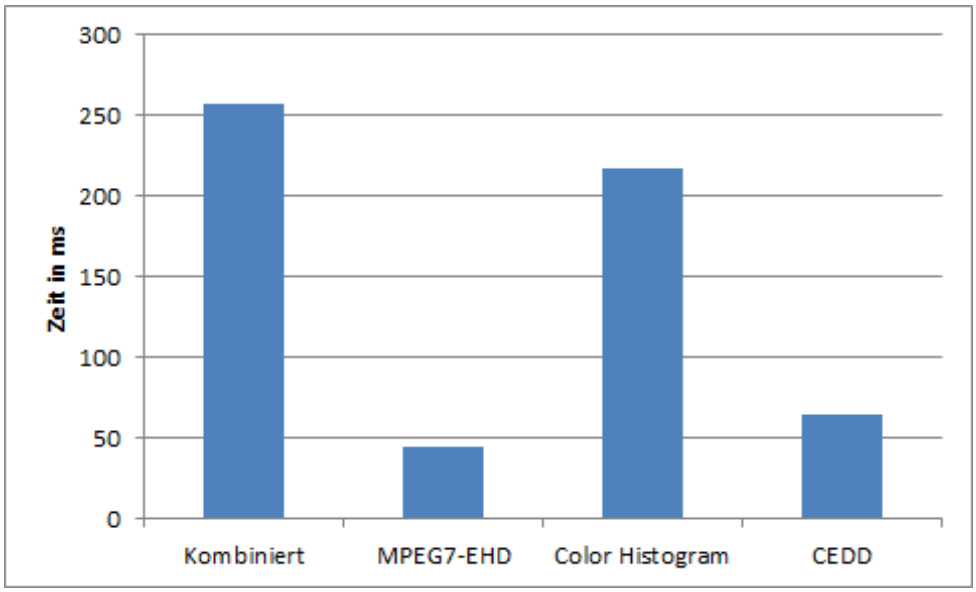


Abbildung 4.7: Durchschnittliche benötigte Zeit für eine Suchiteration mit dem kombinierten Deskriptor im Vergleich zu Color Histogram, MPEG7-EHD und CEDD (Ergebnismengengröße = 10)

Kapitel 5

Indexing

CBIR-Systeme müssen nicht nur gute Ergebnisse liefern, sondern sie müssen auch schnell arbeiten. Es bedarf keiner Laufzeitanalyse um mögliche Schwachstellen im System aufzufinden. Vielmehr genügt es den Anwendungsfall einer Suchiteration unter die Lupe zu nehmen.

Bevor überhaupt eine Suchiteration gestartet werden kann, müssen die Features der Datenbankbilder zunächst extrahiert und in Vektoren gespeichert werden. Die Performance ist hier einerseits von der Datenbankgröße und andererseits von der Komplexität des Deskriptors abhängig. Der einzige Lösungsansatz zur Verbesserung der Performance ist einen einfacheren Deskriptor zu verwenden, was oft den Nebeneffekt hat, dass die Ergebnisse schlechter werden. Aufgrund dieser Eigenschaft und aufgrund der Tatsache, dass die Extraktion der Features für jedes Bild nur einmal gemacht werden muss, kann die Performance hier vernachlässigt werden.

Die eigentliche Suchiteration besteht aus drei Teilen: zunächst muss der Deskriptorvektor aus dem Querybild extrahiert werden, wenn er noch nicht vorhanden ist. Danach folgt die Suche nach den nächsten Nachbarn und schließlich gibt der Benutzer mittels RF Rückmeldung. Die Dauer der Extraktion und des RF können auch hier vernachlässigt werden, da keine Verbesserungsmöglichkeiten vorhanden sind. Genauer muss man jedoch die Suche nach den nächsten Nachbarn betrachten: Bei einer Suchiteration werden alle Datenbankbilder sequentiell durchlaufen und die Distanz zum Querybild wird berechnet. Die Bilder mit den kürzesten Distanzen sind die nächsten Nachbarn. Hier gibt es Verbesserungsbedarf: Eine passende *Indexstruktur* kann verhindern, dass der gesamte Datenbestand durchsucht wird und trägt somit zu einem Performancegewinn bei. Indexstrukturen sind nichts anderes als Datenstrukturen (z.B. Bäume, Hashtabellen), die die Daten so repräsentieren, dass eine Suche effizient realisierbar ist. In unserem Fall bedeutet das, dass ähnliche Bilder in der Indexstruktur nahe beisammen sind und somit nur ein kleiner Bestandteil der gesamten Datenbank durchsucht werden muss. Nicht außer Acht zu lassen ist der Overhead, der beim Erzeugen der Indexstruktur entsteht. Wenn sich die Bilddatenbank häufig ändert, z.B. durch Hinzufügen von Bildern oder durch Manipulation der vorhandenen Bilder, muss die Indexstruktur neu aufgebaut werden.

Die Wahl einer geeigneten Indexstruktur fällt aufgrund der hohen Dimensionalität der Vektoren nicht leicht. Die Anforderungen sind klar, die Indexstruktur muss hochdimensionale Punkte verwalten können und eine effiziente Nearest-Neighbor-Suche anbieten. Eine intuitive Indexstruktur, die diese Anforderungen erfüllt, ist der *homogene* k-*d-Baum* von Bentley (1975), der

in den folgenden Kapiteln vorgestellt und evaluiert wird.

5.1 *k*-d-Bäume

5.1.1 Definition

Ein homogener k-d-Baum (Abk. für k-dimensionaler-Baum) ist ein Binärbaum, der durch Partitionierung eines k-dimensionalen Raumes ($k \in \mathbb{N}$) durch eine Menge von Hyperebenen entsteht. Jeder Knoten des Baumes entspricht einem Punkt aus dem k-dimensionalem Raum, wobei jeder innere Knoten zusätzlich eine Teilungsebene darstellt, welcher den Raum in zwei Halbräume teilt. Die linken bzw. rechten Teilbäume eines jeden inneren Knotens repräsentieren alle Punkte, die sich links bzw. rechts von der teilenden Hyperebene befinden. Das Besondere an diesen teilenden Hyperebenen ist ihre Achsenparallelität, d.h. sie sind immer parallel zu den Achsen des Raumkoordinatensystems ausgerichtet. Die Wahl der Hyperebene erfolgt im Round-Robin-Schema, welches im nächsten Teil des Kapitels im Detail behandelt wird.

Zusätzlich zu den homogenen k-d-Bäumen existiert eine inhomogene Variante, bei der die inneren Knoten nur Zeiger auf den linken und rechten Teilbaum speichern. Die Punkte des Raumes werden nur in den Blattknoten gespeichert. In den folgenden Kapiteln werden homogene k-d-Bäume behandelt. Eine genaue Definition des k-d-Baumes wurde von Bentley (1975) präsentiert.

5.1.2 Operationen

Ein k-d-Baum bietet neben den für Binärbäumen üblichen Operationen *Konstruktion*, *Einfügen* und *Löschen eines Elementes* auch eine *k-Nearest-Neighbor-Search* an, die für das CBIR-System mit einer Suchiteration gleichzustellen ist. Weiters bietet der Baum Bereichsabfragen an, die für unser CBIR-System jedoch uninteressant sind.

Konstruktion

Die Konstruktion des Baumes hängt einzig und allein von der Auswahl der teilenden Hyperebenen ab. Typisch für den k-d-Baum ist die im Round-Robin-Schema wechselnde Auswahl. Beginnend bei der ersten Dimension werden mit steigender Tiefe alle Dimensionen zyklisch durchlaufen bis alle Punkte gesetzt werden. Die Auswahl der aktuellen Teilungsdimension beeinflusst dabei die Achsenausrichtung der teilenden Hyperebene in der aktuellen Tiefe.

Zum Beispiel hätte in einem dreidimensionalen k-d-Baum die Wurzel des Baumes eine Teilungsebene in x-Ausrichtung, die Kinder der Wurzel eine Ebene in y-Ausrichtung und die Enkelkinder der Wurzel eine Ebene in z-Ausrichtung. Deren Kinder hätten dann wieder eine Ebene in x-Ausrichtung.

Neben der Teilungsdimension ist auch die Auswahl des Teilungswertes wichtig. Hierfür werden alle nicht behandelten Punkte in Bezug auf die aktuelle Teilungsdimension sortiert. Der Median wird als Wurzel des Teilbaumes gespeichert und bestimmt den Teilungswert. Alle Punkte, deren Wert kleiner ist als der Teilungswert, wandern in den linken Teilbaum, alle Punkte mit größerem oder gleichem Wert in den rechten Teilbaum, wobei das Prozedere so lange wiederholt wird, bis alle Punkte eingeordnet wurden. Diese Methode führt zu einem balancierten Baum (Bentley (1975)).

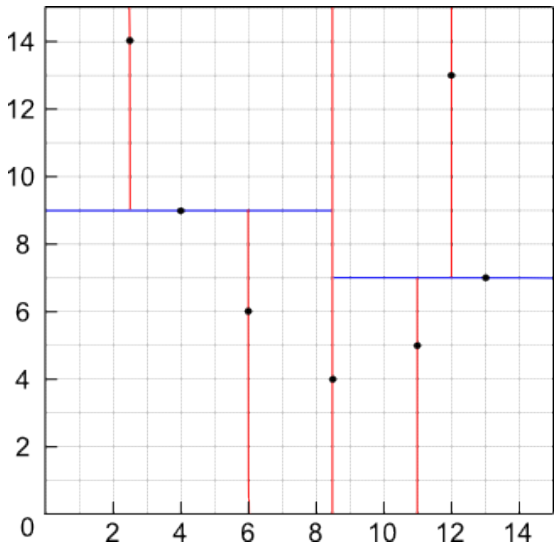


Abbildung 5.1: k-d-Baum-Partitionierung für die zweidimensionale Punktemenge (8.5,4), (4,9), (13,7), (6,6), (2.5,14), (10.5,5), (12,13).

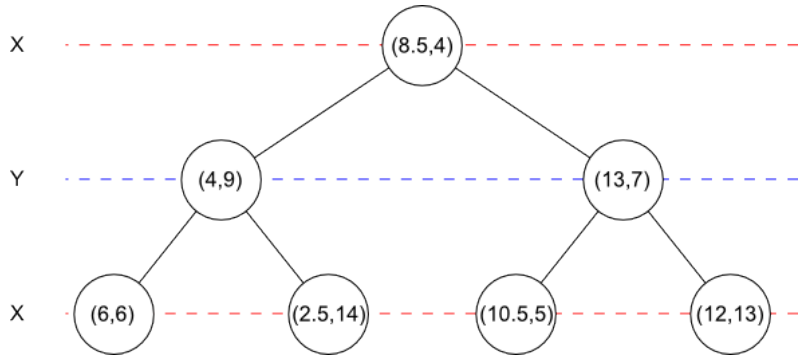


Abbildung 5.2: Der aus Abbildung 5.1 resultierende k-d-Baum.

Abbildung 5.1 zeigt ein Beispiel für die Partitionierung einer Punktemenge im zweidimensionalen Raum. Die roten bzw. blauen Geraden, also Hyperebenen im zweidimensionalen Raum, teilen die Räume in x- bzw. y-Ausrichtung. Abbildung 5.2 zeigt den resultierenden Baum. Dieser Konstruktionsalgorithmus wurde eins zu eins in das CBIR-System übernommen (Konstruktor der Klasse *KDTree.java*).

Nearest-Neighbor-Search

Der Nearest-Neighbor-Search-Algorithmus findet den Punkt mit der niedrigsten Distanz zu einem vom Benutzer spezifizierten Eingabepunkt. Dabei werden die Eigenschaften des k-d-Baumes vorteilhaft genutzt um Teilräume aufgrund ihrer großen Distanz zum Eingabepunkt auszuschließen. Dies hat einen Speedup zum Effekt, weil diese Teilräume (und dadurch dessen Punkte) nicht untersucht werden müssen. Der Nearest-Neighbor-Search-Algorithmus besitzt im durchschnittlichen Fall eine Komplexität von $O(log\ n)$, im schlechtesten Fall jedoch $O(n)$.

Der Algorithmus besteht aus zwei Teilen: beginnend von der Wurzel wird der Baum nach unten traversiert, mit dem Ziel eine gute Annäherung zum *nearest neighbor* zu finden. Dabei

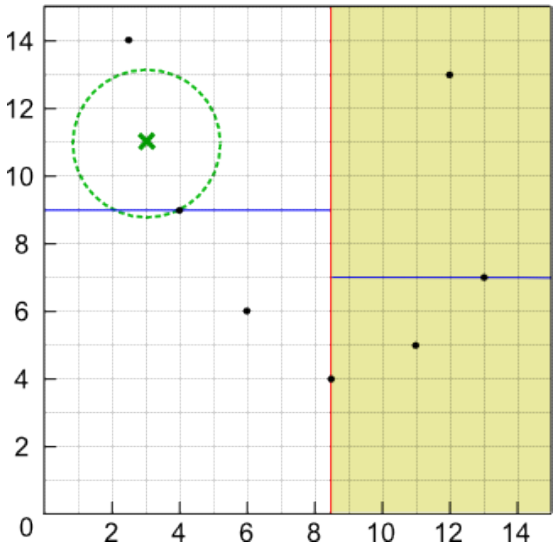


Abbildung 5.3: Illustration des Speedups beim Anwenden des Nearest-Neighbor-Search-Algorithmus für den Punkt (3,11) auf den k-d-Baum aus Abbildung 5.2.

geht man wie beim Einfügen eines Punktes vor: Der Eingabepunkt wandert abhängig vom Wert der aktuellen Teilungsdimension entweder nach links oder rechts, bis er zu einem Blatt kommt. Der im Blatt gespeicherte Punkt wird als *current best* gespeichert. Nach der Ermittlung des *current best* überprüft der Algorithmus rekursiv, ob es Punkte gibt, die eine kleinere Distanz besitzen. Bei jedem Rekursionsschritt geschieht folgendes:

1. Falls der aktuelle Knoten näher ist als der *current best*, so wird dieser als *current best* gespeichert.

2. Der Algorithmus überprüft, ob es nähere Punkte auf der anderen Seite der Teilungsebene geben kann, indem die Teilungsebene mit einer Hypersphäre um den Eingabepunkt geschnitten wird, die als Radius die Distanz zum *current best* besitzt. Da beim k-d-Baum alle Teilungsebenen achsengerichtet sind, genügt ein einfacher Vergleich, ob die Distanz bezüglich der Teilungsdimension zwischen dem aktuellen Punkt und dem Eingabepunkt kleiner ist als die gesamte Distanz zwischen dem Eingabepunkt und dem *current best*.

   (a) Wenn die Teilungsebene die Hypersphäre schneidet, wandert der Algorithmus in den noch nicht betrachteten Teilbaum und sucht nach näheren Punkten, indem er den Algorithmus rekursiv startet, also bei *1.* wieder anfängt.

   (b) Wenn die Teilungsebene die Hypersphäre **nicht** schneidet, dann wandert der Algorithmus weiter nach oben und der nicht betrachtete Teilbaum kann von der Suche ausgeschlossen werden, da er keine näheren Punkte besitzen kann. Der Algorithmus wird rekursiv fortgesetzt, er startet wieder bei *1.*.

3. Der Algorithmus ist beendet, nachdem der Wurzelknoten betrachtet wurde. Der gesuchte Punkt ist als *current best* gespeichert.

Abbildung 5.3 zeigt ein Beispiel für den Nearest-Neighbor-Search-Algorithmus. Ausgehend vom Baum aus Abbildung 5.2 wird der nächste Nachbar für den Punkt (3,11) ermittelt (als grünes X dargestellt). Zunächst wird eine Annäherung mittels Traversierung nach unten gesucht. Man traversiert zuerst nach links (Vergleich der x-Koordinaten: 3 ist kleiner-gleich 8.5) und dann nach rechts (Vergleich der y-Koordinaten: 11 ist größer als 9). Der erreichte Blattknoten besitzt

den Punkt (2.5,14), welcher als *current best* gespeichert wird.

Schließlich wandert der Algorithmus wieder nach oben um bessere Ergebnisse zu finden. Dabei wird einen Schritt nach oben traversiert, der aktuelle Knoten ist somit (4,9). Hier trifft Punkt 1 des Algorithmus zu: (4,9) liegt näher zu (3,11) als der vorherige Punkt (Distanz: $\sqrt{5} < \sqrt{9.25}$), er wird als neuer *current best* gespeichert. Punkt 2 des Algorithmus überprüft, ob es notwendig ist, den noch nicht betrachteten linken Teilbaum zu überprüfen. Dabei wird wie beschrieben die Hypersphäre, im zweidimensionalen Fall ein Kreis (in grüner Farbe), mit der Teilungsebene (blau) geschnitten. Man erkennt sofort, dass sie sich schneiden und dass es nähere Punkte geben kann - somit muss der rechte Teilbaum betrachtet werden, der jedoch beim Untersuchen keine näheren Punkte aufweist.

Das ganze Prozedere wird für den Knoten auf der nächsthöheren Tiefe wiederholt, also für den Wurzelknoten. Punkt 1 des Algorithmus trifft hier nicht zu, bei Punkt 2 schneiden sich Hypersphäre und Teilungsebene (diesmal in Rot) nicht. Der ganze rechte Teilbaum des Wurzelknotens muss nicht betrachtet werden, da sich dort keine besseren Ergebnisse befinden können (dieser Bereich ist in gelber Farbe dargestellt). Insgesamt müssen also statt 7 nur 4 Knoten betrachtet werden um den nächsten Nachbarn zu finden.

Bei CBIR-Systemen muss der Nearest-Neighbor-Search-Algorithmus modifiziert werden, denn eine Bildersuche soll mehr als nur ein Suchergebnis liefern. Die Lösung ist sehr einfach: Statt dem *current best* benutzt man eine Liste, in der man k *current best*s speichert. Wenn nun während dem Ausführen des Algorithmus ein besserer Wert gefunden wird, ersetzt dieser den schlechtesten Wert in der Liste. Diese Modifizierung bezeichnet man als k-Nearest-Neighbor-Search.

Obwohl der k-d-Baum mit Nearest-Neighbor-Search eine optimale Lösung für die Bildersuche bei CBIR-Systemen zu sein scheint, erweist er sich als schlecht (siehe Kapitel 5.3). Grund dafür ist der sogenannte *Fluch der Dimensionalität*, der im folgenden Unterkapitel behandelt wird.

5.2 Fluch der Dimensionalität

Neben des genannten Effekts auf die Distanzfunktion, der auch den k-d-Baum negativ beeinflusst (Kapitel 2.3), gibt es einen Effekt auf den Partitionierungsalgorithmus des Baumes. Bei der Partitionierung von Bildern aus kleinen Datenbanken kommt es dazu, dass viele Dimensionen nicht als Teilungsdimension berücksichtigt werden, da zu wenig Bilder vorhanden sind, um den k-d-Baum in die Tiefe aufzufüllen. Die schlechte Partitionierung und die stark sinkende Bedeutung der Distanzfunktion bei steigender Dimension haben zur Folge, dass beim Nearest-Neighbor-Search-Algorithmus keine Bereiche ausgeschlossen werden können und somit alle Knoten betrachtet werden müssen. Bei so einem Szenario ist der Algorithmus schlechter als eine lineare Suche ohne Indexstruktur. Um eine gute Partitionierung und somit eine bessere Laufzeitreduzierung zu erreichen muss der Baum so gefüllt sein, dass jede Dimension zumindest einmal als Teilungsdimension verwendet wird. Als generelle Regel gilt, dass $N >> 2^k$ ist, wobei N die Anzahl aller Punkte im Raum und k die Dimensionalität darstellt.

Es gibt viele Lösungsansätze für CBIR-Systeme, die versuchen den Effekten des Fluchs der Dimensionalität zu entgehen:

- Die einfachste Lösung wäre, einen einfachen Deskriptor zu benutzen, der sehr wenige Dimensionen verwendet. Der große Nachteil dabei ist jedoch, dass die Suche höchstwahrscheinlich schlechtere Ergebnisse liefern würde als bei komplizierteren Deskriptoren.
- Eine zweite einfache Lösung ist die Modifizierung des Nearest-Neighbor-Search-Algorithmus zu einer Approximation. Darunter versteht man das vorzeitige Abbrechen des Algorithmus, z.B. wenn eine bestimmte Anzahl an Knoten ausgewertet wurde oder eine bestimmte Laufzeit überschritten wurde. Ansätze zur Approximation des Nearest-Neighbor-Search-Algorithmus werden von Indyk (2004) vorgestellt und ausgewertet.
  Neben einem Speedup hat die Approximation zur Folge, dass Suchergebnisse möglicherweise nicht präzise sind, also für die Evaluierung in dieser Arbeit irrelevant sind, da korrekte Ergebnisse Voraussetzung sind.
- Es wird laufend an speziell für hochdimensionale Daten angepasste Indexstrukturen geforscht, die bessere Resultate versprechen. Ramaswamy and Rose (2009) schlagen eine Cluster-basierte Indexstruktur vor, um dem Problem des Fluches zu entgehen. *Locality-sensitive hashing* von Wang and Wang (2009) ist eine effiziente Indexstruktur, die auf einer aufgelockerten Hashtabelle basiert. Aufgelockert deshalb, weil im Gegensatz zu traditionellem Hashing die Eindeutigkeit der Schlüssel nicht angestrebt wird, sondern mehrere ähnliche Deskriptoren demselben Schlüssel zugewiesen werden.

5.3 Evaluierung

Die negative Auswirkung des Fluches der Dimensionalität auf die Performance zeigt sich auch im folgenden Test. Es wurden der kombinierte Deskriptor und die drei Deskriptoren Color Histogram, MPEG7-EHD und CEDD jeweils mit und ohne k-d-Baum getestet. Die verwendete Datenbank ist die *Corel*-Datenbank aus Kapitel 4.2, welche aus 10 800 Bildern besteht. Abbildung 5.4 zeigt die durchschnittlich benötigte Zeit einer Query für die jeweiligen Szenarien.

Wie in 5.4 deutlich zu erkennen ist, ist der k-d-Baum als Indexstruktur für diese Deskriptoren und die kleine Datenbank unbrauchbar. Die durchschnittliche Query-Zeit mit dem k-d-Baum ist für jeden getesteten Deskriptor viel größer als ohne Indexstruktur, eine Suchiteration dauert 50-70% länger. Die Ursache liegt am Fluch der Dimensionalität, es kann keine gute Partitionierung erreicht werden, die Bedingung $N >> 2^k$ wird nicht erfüllt. Es wird entweder eine extrem große Datenbank oder ein kleiner Deskriptor benötigt, um dem Fluch zu entgehen, beides ist jedoch sehr schwer zu realisieren. Ein kleiner Deskriptor besitzt nicht genügend Aussagekraft über ein Bild, die Ergebnisse bei Suchiterationen wären sehr schlecht. Andererseits ist es schon bei einer sehr kleinen Deskriptorgröße von $k = 20$ bezüglich der Datenbankgröße problematisch, sie müsste viel größer sein als 2^k, also viel größer als 1 000 000. Bei $k = 25$ steigt diese Zahl schon auf über 33 Millionen.

Insgesamt ist der k-d-Baum ungeeignet für kleine Datenbanken, es muss eine andere Lösung gefunden werden (Kapitel 5.2). In unserem Fall ist es jedoch ausreichend ohne Indexing zu arbeiten, da die Suche nahezu in Echtzeit erfolgt. Für große Datenbanken hingegen kann ein k-d-Baum eine gute Wahl sein, denn je größer die Datenbank ist, desto besser ist die Partitionierung und damit die Performance. Interessant wäre eine k-d-Baum-Evaluierung mit einer großen Datenbank (> 1 000 000 Bilder), die im Rahmen dieser Arbeit nicht möglich ist. Ab einer gewissen Datenbankgröße ist der k-d-Baum schneller als die lineare Suche.

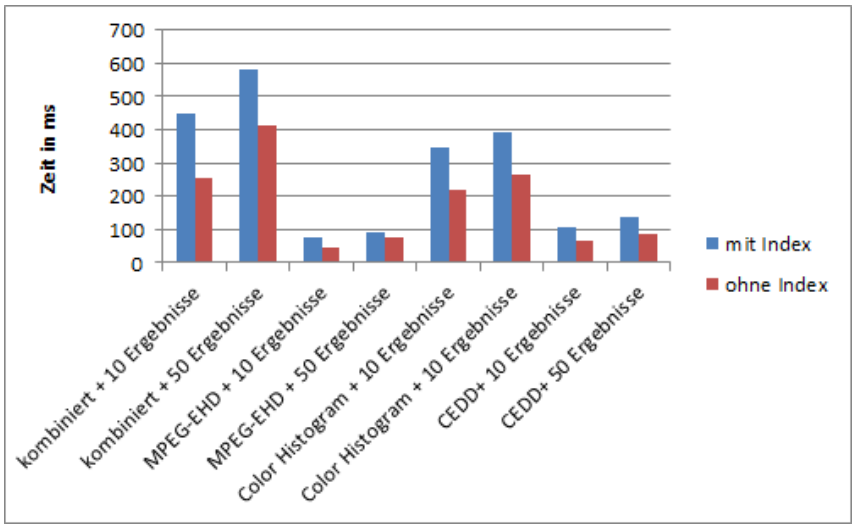


Abbildung 5.4: Durchschnittlich benötigte Zeit für eine Suchiteration mit und ohne Indexing in der *Corel*-Datenbank.

Kapitel 6

Relevance Feedback

Die Ergebnisse bei der Ähnlichkeitssuche (CBIR) von Bildern stimmen meistens nicht mit den Erwartungen des Benutzers überein. Das liegt daran, dass bei CBIR elementare Bildmerkmale miteinander verglichen werden. Benutzer suchen allerdings in der Regel nach Kategorien. Für unser CBIR-System ist ein Bild ein gutes Ergebnis, wenn es viele Merkmale mit dem Eingabebild gemeinsam hat, für den Benutzer hingegen nur dann, wenn es in einer ähnlichen Objektkategorie liegt. Auch an welchen Aspekten des Querybildes der Benutzer interessiert ist, kann der Computer nicht wissen. Um dieses Verständnisproblem zwischen Computer und Menschen zu beheben, wird ein Mechanismus benötigt. Dieser Mechanismus wird als *Relevance Feedback (RF)* bezeichnet.

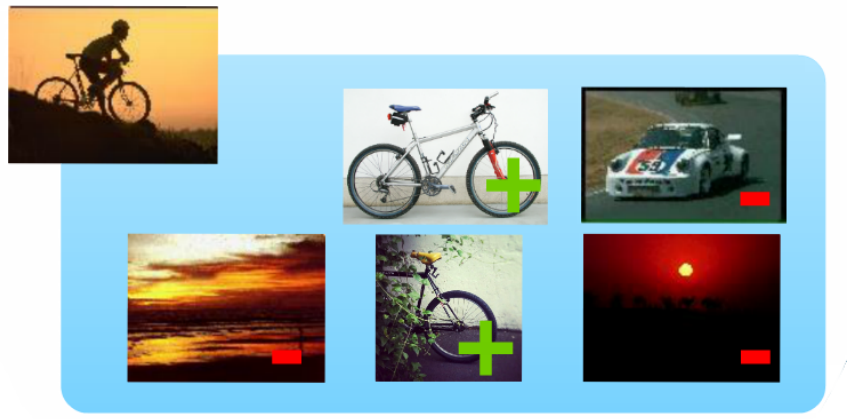


Abbildung 6.1: Beispiel für „Semantic Gap“: Der Benutzer sucht nach Fahrrädern, das CBIR-System findet Fahrräder und Sonnenuntergänge. Durch RF kann der Benutzer dem System mitteilen, dass er nach Fahrrädern sucht und nicht nach Sonnenuntergängen.

RF ermöglicht dem Benutzer, dem System seine Vorstellung von einem relevanten Ergebnis mitzuteilen. Zum Beispiel kann der Benutzer nach jedem Suchdurchlauf Bilder aus der Ergebnismenge als relevant oder als nicht relevant markieren. Mit der so gewonnen Information werden die Suchergebnisse der nächsten Iterationen mittels verschiedenen Algorithmen an die Vorstellungen des Benutzers angeglichen. Im Gegensatz zu den bekannten Machine-Learning-Algorithmen besteht dabei die größte Schwierigkeit darin, Voraussagen über die relevante Klasse von Bildern zu treffen, ohne dabei über eine ausreichende Menge an klassifizierten Bildern zu verfügen. Dieses Problem liegt dem RF per Definition zu Grunde, da die Informationen über die

Klasse von relevanten Bildern von den Bildern mit der geringsten Distanz zum Querybild abgeleitet werden und nicht von einer zufällig verteilten Menge von Bildern. In der Regel markiert der Benutzer nicht mehr als die ersten zwanzig Bilder, unter denen sich meist mehr irrelevante als relevante Bilder befinden.

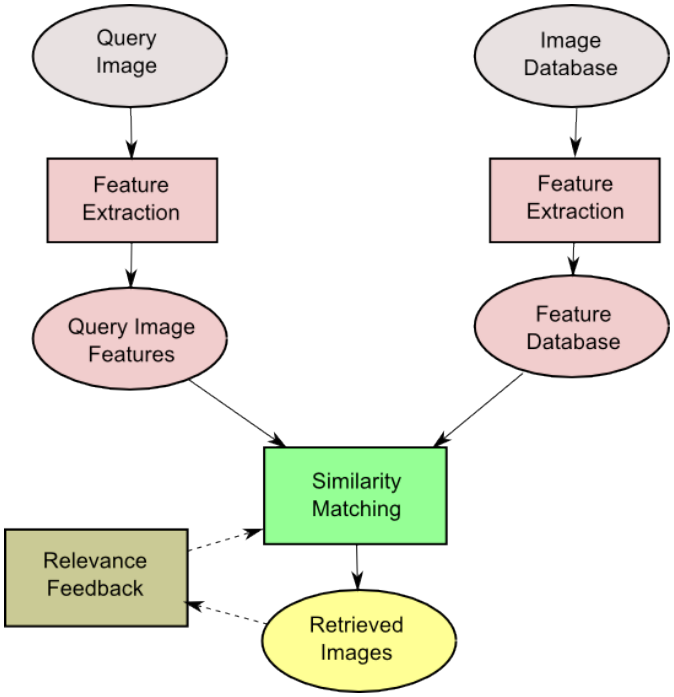


Abbildung 6.2: Illustration eines CBIR-Systems mit RF.

Eine weitere Herausforderung, die es in einem CBIR-System zu bewältigen gilt, stellt die Auswahl eines RF-Algorithmus dar, der mit einer Indexstruktur kombinierbar ist. Dies ist bei großen Datenmengen aus Performancegründen wichtig, weil für ein gutes Benutzererlebnis Suchabfragen in Echtzeit (max. 2-3 Sekunden) notwendig sind.

Im folgenden Teil des Kapitels werden verschiedene RF-Strategien vorgestellt und auf die Kompatibilität mit Indexstrukturen, wie k-d-Bäumen, untersucht.

6.1 Pseudo-Relevance Feedback

In den Anfängen von RF für CBIR wurden RF-Technologien, die für Text-Retrieval (TR) entwickelt wurden, übernommen und für CBIR-Zwecke angepasst. Das Konzept von RF ist bei CBIR dasselbe wie bei TR: Positive Ergebnisse werden markiert, die gewonnenen Informationen verarbeitet und die Qualität der zukünftigen RF-Iterationen wird verbessert. Statt Text-Deskriptoren werden nun Bild-Deskriptoren verwendet. In dieser Arbeit werden diese RF-Methoden als Pseudo-RF-Methoden bezeichnet, da sie auf heuristischen Annahmen über die Beschaffenheit der Deskriptoren von ähnlichen Bildern basieren.

6.1.1 Gewichtung der einzelnen Features

Der wohl intuitivste Ansatz für ein RF-System ist es, eine gewichtete Distanzmetrik zu verwenden und die Gewichte jedes einzelnen Features im Laufe der RF-Iterationen zu ermitteln

(Rui et al. (1997)). Unter der Annahme, dass bei ähnlichen Bildern auch die wichtigen Features ähnlich sind, weist man diesen ein hohes Gewicht zu. Als Metrik kann eine beliebige gewichtete Metrik verwendet werden. Wir demonstrieren den Ansatz für die *gewichtete euklidische Distanz* (Formel 3.2):

$$d(\mathbf{Q}, \mathbf{I}) = \sqrt{(\mathbf{Q} - \mathbf{I})^T W (\mathbf{Q} - \mathbf{I})} \tag{6.1}$$

$\mathbf{Q}$ bezeichnet den Deskriptor des Querybildes, $\mathbf{I}$ ist der Deskriptor eines Bildes aus der Datenbank und W eine Diagonal-Matrix, die in der Diagonale die Gewichte für die jeweiligen Features enthält. Nun bleibt nur noch die Frage, wie die Einträge der Gewichtsmatrix W berechnet werden. Um die Wichtigkeit eines einzelnen Features zu bestimmen, bietet sich die Berechnung der Standardabweichung bezüglich des Features in den als relevant gekennzeichneten Deskriptoren an. Ein Feature ist genau dann wichtig, wenn es in allen relevanten Bildern ähnlich ist, also wenn die Standardabweichung gering ist. Ein Feature ist genau dann unwichtig, wenn es unter den relevanten Bildern variiert, also die Standardabweichung groß ist. Für die Gewichtung eines Features an der Stelle i bietet sich also folgende Formel an (Rui et al. (1997)):

$$W_{ii} = \frac{1}{\sigma_i} \tag{6.2}$$

Um eine initial faire Gewichtung zu erhalten, empfiehlt es sich alle Deskriptoren zu normalisieren. Eine Möglichkeit dafür ist die Gauß-Normierung, siehe Kapitel 2.2. Die gewählte Normierung kann die Qualität der Ergebnisse beeinflussen. Um ein zufriedenstellendes Ergebnis zu bekommen, sollte man daher einige Normierungen ausprobieren. Bei der Verwendung zusammengesetzter Deskriptoren, zum Beispiel bei einem kombinierten Deskriptor, ist es wichtig zu normalisieren, weil die Wertebereiche der verschiedenen Deskriptoren variieren können. Im von uns getesteten kombinierten Deskriptor reichen die Werte der einzelnen Features des Color-Histograms von 0 bis zur Anzahl der Pixel im Bild und beim MPEG7-EHD-Deskriptor von 0 bis 9.

6.1.2 Verschiebung des Query-Punktes

Eine ähnlich simple Methode, die Suchergebnisse zu verbessern, stellt die Verschiebung des Query-Punktes in Richtung der relevanten Ergebnisse dar. Eine Möglichkeit, dies zu erreichen, ist nach Rui et al. (1997) und Moschitti (2003) die Approximation des neuen Query-Punktes mittels der Rocchio-Gleichung:

$$Q^{(i+1)} = \alpha Q^{(i)} + \beta \left(\frac{1}{N_R} \sum_{\mathbf{D} \in M_R(\mathbf{Q^{(i)}})} \mathbf{D} \right) - \gamma \left(\frac{1}{N_N} \sum_{\mathbf{D} \in M_N(\mathbf{Q^{(i)}})} \mathbf{D} \right) \tag{6.3}$$

Der Querypunkt $Q^{(i)}$ soll näher zu den positiven Ergebnissen beziehungsweise weiter weg von den negativen Ergebnissen verschoben werden. Realisiert wird dies durch die Addition vom Mittelwert der positiven Ergebnisse $\frac{1}{N_R} \sum_{\mathbf{D} \in M_R(\mathbf{Q^{(i)}})} \mathbf{D}$ zum altem Querypunkt $Q^{(i)}$ und der anschließenden Subtraktion des Mittelwerts der negativen Ergebnisse $\frac{1}{N_N} \sum_{\mathbf{D} \in M_N(\mathbf{Q^{(i)}})} \mathbf{D}$. $M_R(\mathbf{Q^{(i)}})$ bezeichnet die relevanten Dokumente und $M_N(\mathbf{Q^{(i)}})$ die nicht relevanten. N_R und N_N steht für die Anzahl der relevanten bzw. nicht relevanten Dokumente. Folglich hängt die Rocchio-Gleichung von drei Parametern ab:

- α bestimmt die Gewichtungen des Queryvektors.

- β bestimmt die Gewichtung der als relevant markierten Ergebnisse.
- γ bestimmt die Gewichtung der als irrelevant markierten Ergebnisse.

Für die Verwendung der Rocchio-Querypunktverschiebung empfiehlt Rui et al. (1997) die Umwandlung von Deskriptoren zu Gewichtsvektoren. Hier erfolgt die Gewichtung im Gegensatz zu Kapitel 6.1.1 implizit. Für das Vergleichen zweier Gewichtsvektoren wird die Kosinus-Distanz, siehe Kapitel 3.2.3, vorgeschlagen.

Die Qualität der Ergebnisse hängt stark von der Wahl der Parametern α, β und γ ab. Die optimalen Werte der Parameter sind abhängig von der jeweiligen Datenbank und des Anwendungsfalles. Wenn man zum Beispiel die negativen Resultate ignorieren will, setzt man γ auf Null. Die Wahl dieser Parameter ist aufwändig, da sie nicht nur für jede Datenbank ermittelt werden müssen, sondern sich auch mit der Zeit ändern sollten, wenn sich die Datenbank verändert.

6.2 Bayesischer Ansatz

Eine elegantere RF-Methode stellt das *Bayesian Query Shifting* (BQS) von Vasconcelos and Lippman (2000) dar, bei der die Verschiebung des Queryvektors in Richtung der relevanten Bilder bezüglich einer Annäherung der Entscheidungsgrenze (*decision-boundary*) zwischen relevanten und nicht relevanten Bildern mittels bayesischer Entscheidungstheorie erfolgt. Als Entscheidungsgrenze wird eine Hyperebene bezeichnet, die zwei Klassen voneinander abgrenzt. Im RF-Fall ist das die Grenze zwischen den relevanten und irrelevanten Ergebnissen. Um die Entscheidungsgrenze zu approximieren, werden die Klassen der relevanten und nicht relevanten Bilder als normalverteilt angenommen. Die Parameter μ und σ^2 der Normalverteilungen werden nach Vasconcelos and Lippman (2000) wie folgt approximiert:

$$\mu_{\mathbf{R}} = \frac{1}{N_R} \sum_{\mathbf{I} \in M_R(\mathbf{Q})} \mathbf{I} \tag{6.4}$$

$$\mu_{\mathbf{N}} = \frac{1}{N_N} \sum_{\mathbf{I} \in M_N(\mathbf{Q})} \mathbf{I} \tag{6.5}$$

Als Näherungen für $\mu_{\mathbf{R}}$ und $\mu_{\mathbf{N}}$ werden die Erwartungswerte der relevanten $M_R(\mathbf{Q})$, beziehungsweise nicht relevanten Bilder $M_N(\mathbf{Q})$ verwendet. N_R bezeichnet dabei die Anzahl aller als relevant markierten Bilder und N_N die Anzahl aller als irrelevant markierten Bilder.

$$\sigma^2 = S_W \cdot S_B \tag{6.6}$$

$$S_W = \frac{N_R}{N} \cdot \frac{\sum_{\mathbf{I} \in M_R(\mathbf{Q})} (\mathbf{I} - \mu_{\mathbf{R}})^2}{N_R - 1} + \frac{N_N}{N} \cdot \frac{\sum_{\mathbf{I} \in M_N(\mathbf{Q})} (\mathbf{I} - \mu_{\mathbf{N}})^2}{N_N - 1} \tag{6.7}$$

$$S_B^2 = (\mu_{\mathbf{R}} - \mu_{\mathbf{N}})^2 \tag{6.8}$$

σ^2 wird als Produkt von den beiden Skalaren, Scatter-Between S_B und Scatter-Within S_W, approximiert. Als Scatter-Between S_B wird der Abstand zwischen den beiden Erwartungswerten $\mu_{\mathbf{R}}$ und $\mu_{\mathbf{N}}$ bezeichnet und unter Scatter-Within S_W versteht man die durchschnittliche Abweichung aller markierten Deskriptoren zum Erwartungswert der jeweiligen Bildklasse (Bollwerk, 2005, Seite 84), wobei N die Summe von N_R und N_N ist. Die so erhaltene Approximation der

Varianz σ^2 wird für beide Normalverteilungen verwendet. Anschließend wird eine approximierte Decision-Boundary mit der Verbindungslinie der Erwartungswerte $\mu_{\mathbf{R}}$ und $\mu_{\mathbf{N}}$ beider Klassen geschnitten, um einen Bezugspunkt **x0** für die Queryverschiebung aufzustellen. Nun muss nur noch ermittelt werden, wie weit der Querypunkt vom berechneten Hilfspunkt **x0** aus in Richtung $\mu_{\mathbf{R}} - \mu_{\mathbf{N}}$ verschoben werden muss. Die optimale Position des neuen Querypunktes ist die Position des echten Erwartungswerts der relevanten Bilder. Dieser kann durch den Erwartungswert der relevanten Bilder der nächsten Iteration ${\mu_{\mathbf{R}}}^{(i+1)}$ approximiert werden. Für die Berechnung von ${\mu_{\mathbf{R}}}^{(i+1)}$ sind noch einige Modellannahmen notwendig. Details zur Berechnung sind in Vasconcelos and Lippman (2000) (Seite 1503) zu finden. Als Resultat für die neue Position des Querypunktes bekommt man dann:

$$ {\mu_{\mathbf{R}}}^{(i+1)} = {\mu_{\mathbf{R}}}^{(i)} + \frac{\sigma^2}{||{\mu_{\mathbf{R}}}^{(i)} - {\mu_{\mathbf{N}}}^{(i)}||^2} \cdot \left(1 - \frac{N_R^{(i)} - N_N^{(i)}}{\max\left(N_R^{(i)}, N_N^{(i)}\right)}\right) ({\mu_{\mathbf{R}}}^{(i)} - {\mu_{\mathbf{N}}}^{(i)}) \tag{6.9} $$

Die in Klammern hochgestellten Zahlen bezeichnen die jeweilige RF-Iteration, wobei (i) die aktuelle und $(i+1)$ die zukünftige Iteration bezeichnen. Obwohl viele realitätsferne Annahmen für die Berechnung des neuen Querypunktes verwendet werden, liefert die Methode sehr gute Ergebnisse.

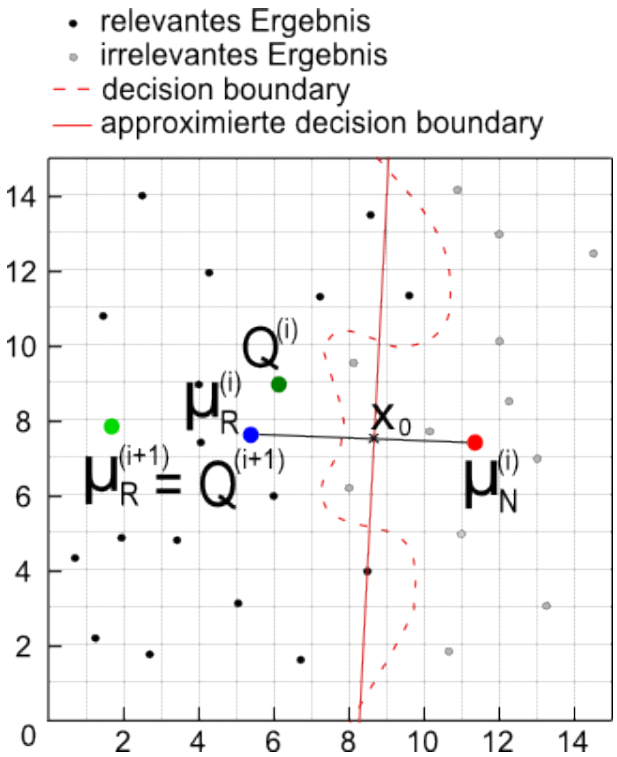


Abbildung 6.3: Illustration der approximierten decision boundary und der Querypunktverschiebung von $Q^{(i)}$ nach $Q^{(i+1)}$.

6.3 Nearest-Neighbor-Methoden

Eine völlig andere RF-Strategie liegt den Nearest Neighbor (NN) Methoden zu Grunde. Anstatt den Querypunkt zu verschieben oder die Features neu zu gewichten, zielen diese Methoden darauf ab, die Wahrscheinlichkeit für die Relevanz jedes Bildes in der Datenbank zu definieren.

Diese Wahrscheinlichkeit hängt, wie der Name "Nearest-Neighbor-Methoden" nahelegt, von dem nächsten relevanten und dem nächsten nicht relevanten Bild ab. Das folgende Kapitel hält sich großteils an Giacinto (2007) und Arevalillo-Herraez and Ferri (2013). Giacinto (2007) leitet folgende Gleichung für die Relevanzwahrscheinlichkeit her:

$$relevance_{NN} = P(relevant|\mathbf{I}) = \frac{d(\mathbf{I}, NN_N(\mathbf{I}))}{d(\mathbf{I}, NN_R(\mathbf{I})) + d(\mathbf{I}, NN_N(\mathbf{I}))} \tag{6.10}$$

Die Wahrscheinlichkeit für die Relevanz eines Bildes mit dem Deskriptor $\mathbf{I}$ hängt nun konkret von dessen Abstand zum Deskriptor des nächsten als relevant markierten Bildes $NN_R(\mathbf{I})$ und zum Deskriptor des nächsten als irrelevant markierten Bildes $NN_N(\mathbf{I})$ ab. Für relevante Bilder $\mathbf{Q_R} \in M_R(\mathbf{Q})$ ist $d(\mathbf{Q_R}, NN_R(\mathbf{Q_R}))$ gleich Null und somit $P(relevant|\mathbf{Q_R})$ gleich $\frac{d(\mathbf{Q_R}, NN_N(\mathbf{Q_R}))}{d(\mathbf{Q_R}, NN_N(\mathbf{Q_R}))}$ also gleich Eins, was der maximalen Wahrscheinlichkeit von 100% entspricht. Analog dazu gilt für irrelevante Bilder $Q_N \in M_N(\mathbf{Q})$, dass $P(relevant|\mathbf{Q_N})$ gleich Null ist, was dem Minimalwert entspricht. Für neue Bilder kommt man mit dieser Gleichung also auf Werte im Intervall $[0, 1]$. Anders als bei den vorigen Methoden wird somit die Suche nach relevanten Bildern in mehreren Bereichen der Datenbank fortgesetzt. Die Query wird durch die relevanten und irrelevanten Bilder erweitert.

6.3.1 Stabilisierung

Im Falle einer kleinen Menge von positiven Ergebnissen ist die Berechnung der Relevanzwahrscheinlichkeit nach Gleichung 6.10 ungeeignet, weil sie dann nur für jene Bilder hohe Werte ergibt, die entweder in unmittelbarer Nähe von den wenigen relevanten Bildern oder sehr weit entfernt von den irrelevanten Bildern liegen. Dieser Fall kommt in der Realität öfters vor, da in den meisten CBIR-Systemen überschaubare Ergebnismengen zwischen 10 und 30 Bildern verwendet werden. Die Qualität der so erhaltenen Ergebnisse variiert also stark im gesamten Deskriptor-Vektorraum. Gibt es für eine Query unter den ersten Ergebnissen ohne RF genügend relevante Ergebnisse, so ist die Qualität sehr gut, andernfalls sehr schlecht. Um diesem Problem entgegen zu wirken, gibt es mehrere Möglichkeiten:

Nearest-Neighbors-RF kombiniert mit Bayesian Query Shifting

Giacinto (2007) schlägt vor, die Relevanzberechnung um einen Term zu erweitern, der den Queryvektor mittels bayesischem Query-Shifting (siehe Kapitel 6.2) an eine bessere Position verschiebt, falls die ursprüngliche Position nur wenige relevante Ergebnisse liefert. Realisiert wird dies mittels einer gewichteten Summe der Relevanzwahrscheinlichkeiten vom NN-Ansatz und vom BQS-Ansatz. Für die Berechnung der BQS-Relevanzwahrscheinlichkeit transformiert Giacinto die Distanz vom zu bewertenden Bild I zum verschobenen Querypunkt Q_{BQS} in eine Punktezahl im Intervall $[0, 1]$ mittels folgender Gleichung:

$$relevance_{BQS}(\mathbf{I}) = \frac{1 - exp(1 - d(\mathbf{Q_{BQS}}, \mathbf{I}) / \max_{\mathbf{I}}(d(\mathbf{Q_{BQS}}, \mathbf{I})))}{1 - e} \tag{6.11}$$

$d(\mathbf{Q_{BQS}}, \mathbf{I})$ notiert die Distanz vom zu bewertenden Bild-Deskriptorvektor $\mathbf{I}$ zum mittels BQS erhaltenen neuen Queryvektor $\mathbf{Q_{BQS}}$. Diese Distanz wird zunächst mit der Division durch die in der Deskriptor Datenbank vorkommende Maximaldistanz zum neuen Queryvektor $\max_{\mathbf{I}}(d(\mathbf{Q_{BQS}}, \mathbf{I}))$ auf den Wertebereich $[0, 1]$ normiert und anschließend zu einer Punktzahl konvertiert $1 - \frac{d(\mathbf{Q_{BQS}}, \mathbf{I})}{\max_{\mathbf{I}}(d(\mathbf{Q_{BQS}}, \mathbf{I}))}$. Zu guter Letzt wird die so erhaltene Punktzahl zum Exponenten eines Gauß'schen Modells. Die kombinierte Relevanz-Punktzahl ergibt sich nun als gewichtete Summe der beiden Relevanzwahrscheinlichkeiten.

$$relevance_{NN+BQS}(\mathbf{I}) = w_{NN} \cdot relevance_{NN} + w_{BQS} \cdot relevance_{BQS} \tag{6.12}$$

w_{NN} bezeichnet das Gewicht der NN-Relevanzwahrscheinlichkeit und w_{BQS} das Gewicht der BQS-Relevanzwahrscheinlichkeit. Als Gewichte w_{NN} und w_{BQS} werden $\frac{1}{n/k+1}$ und $\frac{n/k}{n/k+1}$ verwendet, wobei n für die Anzahl der irrelevanten Bilder steht und k für die Gesamtanzahl von markierten Bildern. Falls keine Ergebnisse als relevant markiert wurden, sind beide Gewichte 1/2. Mit steigender Anzahl von relevanten Markierungen geht w_{BQS} gegen Null und w_{NN} gegen Eins.

Einführung einer Zufallsvariable für die Zuverlässigkeit

Ein Ansatz, der bessere Ergebnisse liefern soll als der im vorigen Abschnitt vorgestellte, wird von Arevalillo-Herraez and Ferri (2013) präsentiert. Grundidee dabei ist, eine Zufallsvariable $reliable$ mit den möglichen Werten $true$ und $false$ einzuführen, die vom zu bewertenden Bild I abhängt und angibt, ob die mittels der Formel 6.10 berechnete Relevanzwahrscheinlichkeit zuverlässig ist. Wenn sie nicht zuverlässig ist, wird die a-priori-Wahrscheinlichkeit $P(relevant)$ der Relevanz verwendet und, wenn sie zuverlässig ist, die a-posteriori-Wahrscheinlichkeit $P(relevant|I)$ der Relevanz bezüglich I. Unter der Annahme, dass Relevanz und Zuverlässigkeit unabhängig sind, kann folgende Gleichung für die Wahrscheinlichkeit der Relevanz des Bildes I verwendet werden:

$$P'(relevant|\mathbf{I}) = P(reliable|\mathbf{I}) \cdot P(relevant|\mathbf{I}) + (1 - P(reliable|\mathbf{I})) \cdot P(relevant) \tag{6.13}$$

Die Wahrscheinlichkeit der Zuverlässigkeit in Abhängigkeit vom Bild $P(reliable|\mathbf{I})$ bestimmt die Gewichtung der einzelnen Komponenten der Gleichung. Je zuverlässiger der Wert von $P(relevant|\mathbf{I})$ ist, desto höher soll $P(reliable|\mathbf{I})$ sein und umgekehrt. Bei der Verwendung von Distanzfunktionen, deren Eingabewerte Deskriptoren mit Einträgen in $\mathbb{R}^{\geq 0}$ sind und deren Wertebereich $[0, 1]$ ist, wird für $P(reliable|\mathbf{I})$ folgende Approximation vorgeschlagen:

$$P(reliable|\mathbf{I}) = 1 - \min(d(\mathbf{I}, NN_N(\mathbf{I})), d(\mathbf{I}, NN_R(\mathbf{I}))) \tag{6.14}$$

Eine geeignete Distanzfunktion wäre zum Beispiel die Kosinus-Distanz. $NN_N(\mathbf{I})$ notiert wieder den nächsten irrelevanten und $NN_R(\mathbf{I})$ den nächsten relevanten Deskriptor. Als konkrete Formel für die stabilisierte Relevanzwarscheinlichkeit ergibt sich nach dem Einsetzen von Gleichung 6.10 und 6.14 in die Gleichung 6.13 und der Vereinfachung der resultierenden Gleichung:

$$P_{stabil} = (1 - \min(d(\mathbf{I}, NN_N(\mathbf{I})), d(\mathbf{I}, NN_R(\mathbf{I})))) \cdot \frac{d(\mathbf{I}, NN_N(\mathbf{I}))}{d(\mathbf{I}, NN_R(\mathbf{I})) + d(\mathbf{I}, NN_N(\mathbf{I}))} \tag{6.15}$$

6.4 Relevance Feedback als Klassifikationsproblem

Im Trend liegen RF-Methoden, die das RF-Problem als Klassifizierungsproblem auffassen. Hierbei gibt es zwei Richtungen, bei der einen wird das RF-Problem als Zwei-Klassen-Problem aufgefasst, bei der anderen als (1 + x)-Klassen-Problem. Beim Zwei-Klassen-Problem gilt es zwischen der relevanten und der nicht relevanten Bildklasse zu unterscheiden und beim (1 + x)-Klassen-Problem wird die Tatsache berücksichtigt, dass nicht alle irrelevanten Bilder zur selben Bildklasse gehören und dass die Anzahl der nicht relevanten Bildklassen in der Datenbank unbekannt ist. Um Klassifizierungsprobleme zu lösen, können beliebige Klassifizierer eingesetzt werden, am häufigsten werden im RF-Fall "Support Vektor Maschinen (SVM)" und "Diskriminanten Analysen (DA)" verwendet. In unserer Arbeit werden wir auf diese Methoden nicht

eingehen, wir verweisen daher auf die bestehende Literatur. In Tao et al. (2006a) wird eine RF-Strategie basierend auf Diskriminanten Analyse vorgestellt und in Tao et al. (2006b) wird eine RF-Strategie basierend auf Support Vektor Maschinen vorgestellt.

6.5 Analyse bezüglich der Kompatibilität mit Indexing

6.5.1 Gewichtung der einzelnen Features

Beim im Kapitel 6.1.1 vorgestellten Pseudo-RF ist schon am Ansatz zu erkennen, dass diese naive RF-Methode inkompatibel mit den meisten Indexstrukturen ist. Indexstrukturen, wie zum Beispiel k-d-Bäume, basieren darauf, die Bilder aus der Datenbank anhand einer gewissen Distanzmetrik in einer Datenstruktur anzuordnen, die eine schnelle Suche ermöglicht. Bei dieser RF-Methode verändert sich die Distanzmetrik durch die Neugewichtung der Features nach jeder RF-Iteration. Folglich müsste für jede mögliche Gewichtsmatrix eine eigene Indexstruktur berechnet werden. Das ist in der Praxis nicht machbar, in unserem Fall hätten wir zum Beispiel 580 Gewichte, die theoretisch alle Werte im Bereich der positiven reellen Zahlen annehmen können, also unendlich viele Möglichkeiten.

6.5.2 Querypunkt Verschiebungen

Bei den in Kapiteln 6.1.2 und 6.2 vorgestellten RF-Methoden handelt es sich in beiden Fällen um Verschiebungen des Query-Punktes in eine günstigere Region. Folglich verhalten sie sich im Bezug auf Indexstrukturen äquivalent. Querypunkt Verschiebungen sind kompatibel mit den meisten Indexstrukturen, da die Distanzfunktion, welche Grundstein für die Generierung von Indexstrukturen ist, bei jeder RF-Iteration unverändert bleibt.

6.5.3 Nearest-Neighbors-RF

Der im Kapitel 6.3 vorgestellte RF-Ansatz ist deutlich berechnungsintensiver als die vorherigen RF-Methoden, umso wichtiger ist die Frage der Kompatibilität mit Indexing. Das NN-RF basiert auf einer Neuerstellung der Bild-Rangliste bezüglich der Relevanzwahrscheinlichkeiten der einzelnen Bilder. Die Berechnung der Relevanzwahrscheinlichkeit hängt für jedes Bild von der Distanz zum nächsten als relevant beziehungsweise als irrelevant markierten Bildes ab. Auf den ersten Blick scheint es kompliziert, diese immer wiederkehrende Neuerstellung der Bild-Rangliste bezüglich der aktuellen Relevanzwahrscheinlichkeit mittels Indizierung zu beschleunigen, da sich die nächsten relevanten und irrelevanten Bilder bezüglich des zu bewertenden Bildes nach jeder RF-Iteration verändern können. Eine Indexstruktur mittels der Relevanzwahrscheinlichkeiten zu generieren, kann folglich nicht funktionieren. Allerdings kann die Berechnung der Relevanzwahrscheinlichkeit beschleunigt werden, wenn eine Indexstruktur für die verwendete Distanzfunktion erstellt wird. Das NN-RF verwendet nämlich immer wieder dieselbe Distanzfunktion, ohne diese zu verändern. Beim ersten Suchdurchlauf werden die besten k Ergebnisse ohnehin allein durch die Verwendung der Distanzfunktion ermittelt und für die Berechnung der Relevanzwahrscheinlichkeit werden nur Distanzen zwischen Bildern, die in der Datenbank sind, also für Bilder, deren räumliche Anordnung in der Indexstruktur erfasst wird, berechnet. Der Speedup, der alleine durch die Verwendung einer Indexstruktur resultiert, könnte bei dieser RF-Methode nicht ausreichend sein, um ein Benutzererlebnis in Echtzeit zu garantieren. Ab der ersten RF-Iteration wird lediglich die Berechnung der Relevanzwahrscheinlichkeit beschleunigt, diese muss aber dennoch für alle Bilder berechnet werden. Weitere Optimierungen, wie zum Beispiel die Parallelisierung der Punktezahlberechnung, sind daher empfehlenswert.

NN mit Zuverlässigkeits-Zufallsvariable

Bei dieser Erweiterung der NN-RF-Methode verändert sich die Berechnung der Relevanzwahrscheinlichkeit lediglich um einen zusätzlichen Regulierungsterm (Formel 6.15), der wiederum die Distanzfunktion verwendet, aber nicht modifiziert. Für die Kompatibilität mit Indexstrukturen gelten also dieselben Limitierungen wie beim NN-RF.

NN mit BQS

Die bayesische Query-Verschiebung ist völlig kompatibel mit beliebigen Indexstrukturen. Für das NN-RF stellt Indexing auch kein großes Problem dar. Folglich ist auch die Kombination beider Methoden kompatibel mit Indexstrukturen.

6.6 Evaluierung

Um festzustellen inwiefern die vorgestellten RF-Mechanismen die Query-Ergebnisse verbessern, testen wir diese auf verschiedenen Datenbanken. In diesem Kapitel wird zuerst eine „objektive“ vollautomatische Evaluierung durchgeführt und anschließend eine subjektive.

Getestet werden folgende RF-Mechanismen:

- **Gewichte**
  RF-Mechanismus: 6.1.1 Gewichtung der einzelnen Features
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus Metrik

- **Rocchio**
  RF-Mechanismus: 6.1.2 Verschiebung des Query-Punktes mittels Rocchio Gleichung
  Parameter: $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.15$
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus-Metrik

- **BQS**
  RF-Mechanismus: 6.2 Bayesische Query Verschiebung
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus-Metrik

- **NN**
  RF-Mechanismus: 6.3 Nearest Neighbors RF
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus-Metrik

- **NN & BQS**
  RF-Mechanismus: 6.3.1 NN-RF mit BQS als Stabilisierung
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus-Metrik

- **NN & reliable**
  RF-Mechanismus: 6.3.1 NN-RF mit Zuverlässigkeitszufallsvariable als Stabilisierung
  Normierung: 2.2 Gauß-Normierung auf [0,1]
  Metrik: 3.2.2 Gewichtete Kosinus-Metrik

Im nächsten Teil der Arbeit werden nur noch die fettgedruckten Abkürzungen verwendet.

6.6.1 Automatische Evaluierung

Test Setup

Getestet wurde wieder auf der *Corel* Datenbank, diese beinhaltet in etwa 10.000 Bilder aus 80 verschiedenen Objektkategorien. Die genaue Spezifikation der *Corel* Datenbank und die Vorgehensweise beim automatisierten Test ist im Kapitel 4.2 zu finden. Für die RF-Evaluierung wurden einmalig 500 Querybilder zufällig gewählt und für jede Query wurden vier RF-Iterationen automatisch durchgeführt. Dabei wurden die durchschnittliche Präzision, die Standardabweichung und die durchschnittliche RF-Zeit ermittelt.

Ergebnisse und Schlussfolgerungen

An Abbildung 6.4 (a) ist zunächst auffällig, dass die Qualität der Ergebnisse bei allen Methoden mit der Anzahl von RF-Iterationen steigt, wobei die Präzision aller RF-Methoden, außer **Gewichte**, tendenziell steigend ist und den Anschein macht auch in weiteren Iterationen zu steigen.

Beim Gewichte-RF scheint die Ergebnisqualität nach der zweiten RF-Iteration zu stagnieren. Grund dafür ist vermutlich die Tatsache, dass **Gewichte** zur Ermittlung der neuen Ergebnisse ausschließlich die als positiv markierten Bilder berücksichtigt. Sollte eine RF-Iteration keine neuen positiven Ergebnisse liefern, so stagniert diese Methode. Dennoch können die damit erhaltenen Ergebnisse in den ersten vier RF-Iterationen mit **Rocchio** mithalten. Die intuitive Querypunkt-Verschiebung mittels Rocchio-Gleichung verbessert die Ergebnisse zwar etwas langsamer als **Gewichte**, aber die Ergebnisse tendieren dahin selbst nach der vierten RF-Iteration zu steigen. Diese Tendenz ist mit der Berücksichtigung der als negativ markierten Bilder bei der Querypunkt-Verschiebung zu begründen. Allerdings schneiden die Pseudo-RF-Methoden im Vergleich zu den neueren RF-Methoden eher bescheiden ab. Zu **BQS** und **NN & BQS** beträgt der Unterschied nach der ersten RF-Iteration über 8.7%, was bei einer Ergebnismenge von 20 Bildern ca. zwei relevanten Bildern entspricht. Zu **NN** und **NN & reliable** beträgt der Unterschied nach der zweiten Iteration über bzw. 5.2%, was ca. einem relevanten Bild entspricht.

Somit scheiden die Pseudo-RF Methoden vorerst aus. **BQS** und **NN** verdoppeln die Anzahl der relevanten Ergebnisse annähernd nach drei beziehungsweise vier RF-Iterationen, wobei auffällig ist, dass sich die mittels RF dazugewonnenen Ergebnisse zwischen den beiden Methoden großteils unterscheiden. Der signifikante Präzisionsgewinn durch die Kombination beider Methoden in **NN & BQS** ist daher nicht überraschend. Mit **BQS** wird der Querypunkt näher zu den relevanten Deskriptoren verschoben und bei **NN** werden die Bereiche um den als relevant markierten Deskriptoren erkundet. **NN & reliable** schlägt die Präzision von **NN & BQS** in diesem Fall nicht wie von Arevalillo-Herraez and Ferri (2013) versprochen. Die exakte Ursache dafür ist schleierhaft, von der Beschaffenheit der Datenbank zur verwendeten Relevanzwahrscheinlichkeitsfunktion bis hin zur verwendeten Distanzfunktion für die Ermittlung der Zuverlässigkeitwahrscheinlichkeit ist nichts auszuschließen. Abbildung 6.4 (a) legt also Nahe, dass **NN & BQS** mit Abstand die beste der getesteten RF-Methoden ist.

Aus Abbildung 6.4 (b) lassen sich Aussagen über die Ergebnisstabilität der einzelnen RF-Methoden treffen, wobei eine Methode als stabil angesehen wird, wenn die Standardabweichung ihrer Ergebnisse gering ist. Als Erstes ist auffällig, dass die Ergebnisse mit RF stärker variieren als ohne. Dieser Effekt lässt sich davon ableiten, dass durch die Verwendung von RF ein weiterer Faktor eingeführt wird, der neben verwendeten Deskriptor, Metrik und Normalisierung die Ergebnisse beeinflusst.

Bei **Gewichte** zeigt die Standardabweichungskurve einen ähnlichen Trend wie die Präzisionskurve, ab der zweiten RF-Iteration scheinen die Ergebnisse großteils unverändert zu bleiben. **Rocchio** zeigt mit zunehmender Anzahl von RF-Iterationen eine steigende Standardabweichung. Die geringste Ergebnisstabilität zeigt **BQS**, dafür bleibt diese in allen Iterationen ungefähr gleich. Die Ergebnisse ändern sich mit **BQS** also am markantesten.

Bei den **NN**-Stabilisierungen tendiert die Standardabweichung der Ergebnisse mit steigender RF-Iterationen Anzahl zu sinken. Bei **NN** ist dieser Trend nicht zu erkennen. Die Stabilisierungen **NN & reliable** und **NN & BQS** scheinen also tatsächlich zu wirken. Am stärksten ist dieser Effekt bei **NN & BQS** ausgeprägt, wo die Standardabweichung ab der zweiten RF-Iteration deutlich sinkt und nach der vierten RF-Iteration geringer als alle anderen ist. Das Sinken der Standardabweichung mit zunehmender Anzahl an RF-Iterationen könnte ein Indiz für die Konvergenz hin zur optimalen Ergebnismenge sein. Deshalb ist **NN & BQS** weiterhin klarer Favorit. Weitere Tests in diese Richtung sollten durchgeführt werden, um diese Feststellung eventuell zu untermauern.

Wird zusätzlich die durchschnittlich beanspruchte Zeit für eine RF-Iteration berücksichtigt, siehe Abbildung 6.4 (c), so wird auf einen Blick klar, dass die beeindruckenden Ergebnisse der **NN**-basierenden Methoden mit Vorsicht zu genießen sind. Bei der Neuerstellung der Bildrangliste explodiert die Rechenzeit aufgrund des hohen Berechnungsaufwandes der Relevanzwahrscheinlichkeit, die für alle Bilder berechnet werden muss. Im Gegensatz zu **Gewichte**, **Rocchio** und **BQS**, bei denen eine RF-Iteration durchschnittlich weniger als eine halbe Sekunde dauert, liegt die Dauer einer RF-Iteration bei den **NN**-basierenden Methoden ohne jegliche Optimierungen jeweils bei ca. einer Minute. Des Weiteren steigt der Rechenaufwand für die Relevanzwahrscheinlichkeit mit der Anzahl der gesamt markierten Bilder, also mit der Anzahl der RF-Iterationen. Folglich ist **BQS** für die Verwendung in einem CBIR-System, das ein gutes Benutzererlebnis bieten soll, nach der automatischen Evaluierung ein guter Kompromiss.

6.6.2 Subjektive Evaluierung

Die subjektive Evaluierung bleibt im Bereich von CBIR trotzdem notwendig, da die vollautomatische Evaluierung streng genommen falsch ist. Der Namensgebung "Content Based Image Retrieval" ist zu entnehmen, dass es sich dabei um eine Suche von Bildern bezüglich derer inhaltlichen Ähnlichkeit handelt. Was genau inhaltliche Ähnlichkeit ist, ist dabei nicht festgelegt. Die Ähnlichkeit von Bildern an deren Objektkategorien festzumachen ist daher nicht immer gerechtfertigt.

Ein Gegenbeispiel:

Obwohl die beiden Bilder in Abbildung 6.5 für die meisten Menschen zweifellos ähnlich er-

scheinen, da sie beide Katzen enthalten, wird das zweite Bild bei der automatischen Evaluierung nicht als relevantes Bild für die Query, die durch das erste Bild formuliert wird, gewertet, weil das erste Bild als Wildkatze und das zweite als Hauskatze gelabelt wurden. Dieses Versagen der vollautomatischen Methode kann zwar auf den Labelprozess zurückgeführt werden, ist aber dennoch unumgänglich, da für die Automatisierung des Tests ein manueller Labelprozess notwendig ist. Die Labels für Bilder können nie nach objektiven Kriterien vergeben werden, da es bei Bildern immer darauf ankommt, was der Nutzer sucht.

Sollte dieses Gegenbeispiel nicht ausreichend überzeugend sein, gibt es noch genügend weitere Schwachstellen. Bilder können zum Beispiel mehrere verschiedene Objekte enthalten und es kann durchaus möglich sein, dass der Benutzer des Systems nur nach einem dieser Objekte Ausschau hält. Egal wie die Labels verteilt werden, dieser Fall kann nicht ohne zusätzlichen Aufwand von einem automatisierten Evaluierungsverfahren erfasst werden. Angenommen das Bild erhält die Labels beider Kategorien, dann werden vom automatischen Evaluierungsverfahren vermutlich Bilder als relevant gewertet, die mindestens eines der beiden Labels besitzen. Sobald das automatische Verfahren mit einem Mechanismus erweitert wird, der bei mehreren vorhandenen Labels irgendwelche speziellen Entscheidungsverfahren für eine relevante Teilmenge von Labels verwendet, sinkt die Vergleichbarkeit mit anderen Systemen.

Um dieser Argumentation für eine subjektive Evaluierung trotz der vorhandenen automatisierten Evaluierung ein Ende zu setzen, lässt sich das Problem auf einen Satz reduzieren: Der Benutzer eines Systems kann immer eine Suchanfrage formulieren, deren Abbildung auf Bildlabels nicht möglich oder nicht vorhanden ist.

Testsetup

Getestet wird auf den vier Datenbanken, die uns von der Firma M-BOX zur Verfügung gestellt wurden, genauere Spezifikationen über die Anzahl der Bilder und deren Inhalt sind im Kapitel 2.4 zu finden.

Bewertungskriterien

Als Formel für die Präzision verwenden wir wieder $\frac{\#guteErgebnisse}{n} * 100$. Damit nachvollziehbar ist, welche Bilder als relevant oder als nicht relevant markiert werden, legen wir nun Bewertungskriterien fest. Die Bewertungskriterien in Kapitel 2.4 wurden zu einem Zeitpunkt erstellt, an dem noch kein RF-Mechanismus in unserem System implementiert war, deshalb sind sie etwas zu locker, um den Effekt vom RF zu unterstreichen. Daher wird eine verschärfte Version der Bewertungskriterien eingeführt:

- **lentos-drawing:** Schwarz-weiße Zeichnungen, die auf der linken Bildhälfte eine stehende Person enthalten, werden als positiv markiert, andere als negativ.
- **lentos-box:** Bilder von Boxen werden als positiv bewertet, andere als negativ.
- **lentos-cartoon:** Bilder von weiblichen Cartoonfiguren und Menschen werden als positiv bewertet und andere als negativ.
- **lentos-group:** Gruppenfotos werden als positiv bewertet, andere als negativ.
- **madias-building:** Bilder von Gebäuden auf einem Hügel werden als positiv bewertet, andere als negativ.

- **madias-pinkbg:** Bilder, die Illustrationen in einem ähnlichen Zeichenstil enthalten, werden als positiv bewertet, andere als negativ.
- **madias-tree:** Bilder/Illustrationen mit Bäumen werden als positiv bewertet, andere als negativ.
- **madias-warrior:** Zeichnungen von Kriegern und anderen Personen im ähnlichen Zeichenstil werden als positiv, andere als negativ bewertet.
- **oemv-image**[1]**:** Bilder, die ein ähnliches Blumenmuster enthalten, werden als positiv bewertet, andere als negativ.
- **oemv-knife**[1]**:** Bilder von Messern mit sichtbarer Klinge werden als positiv bewertet, andere als negativ.
- **oemv-sheep**[1]**:** Holzfiguren von Tieren werden als positiv bewertet, andere als negativ.
- **oemv-woman**[1]**:** Zeichnungen von Personen werden als positiv bewertet, andere als negativ.
- **cars-red:** Bei dieser Query sucht der Benutzer nach roten Autos, folglich werden alle Bilder, die im Vordergrund ein rotes Auto enthalten, als positiv markiert. Alle anderen Bilder werden als negativ markiert.
- **cars-side:** Das Querybild ist dasselbe wie bei *cars-red*, allerdings sind die Bewertungskriterien diesmal anders. Alle Seitenaufnahmen von Autos werden positiv bewertet, andere als negativ.
- **cars-wheel:** Nahaufnahmen von Lenkrädern werden als positiv markiert, Aufnahmen, auf denen Lenkräder sichtbar sind aber nicht im Vordergrund stehen, werden ignoriert und alle anderen werden als negativ markiert.
- **cars-multiple:** Bilder, die mehr als zwei Autos enthalten, werden also positiv und andere als negativ bewertet.

Das subjektive RF ist sehr fehleranfällig, da sehr viele ähnliche Bilder immer wieder gleich bewertet werden müssen. Das größte Problem dabei ist, dass Inkonsistenzen in der Bewertung dazu führen, dass die verschiedenen RF-Mechanismen unfair beurteilt werden. Um diesem Problem zu entgehen, verwenden wir bei der Evaluierung eine zusätzliche Komponente, nämlich einen RF-Assistenten. Der RF-Assistent bessert jegliche Inkonsistenzen in der Bewertung aus. Realisiert wird dies dadurch, dass zu jeder Query jeweils alle als positiv oder negativ markierten Bilder gespeichert werden. Anschließend werden bei der Bewertung einer neuen Query-Ergebnismenge alle bereits bekannten Ergebnisse automatisch bewertet.

Ergebnisse und Schlussfolgerungen

Abbildung 6.6 (a) ist großteils ähnlich zu Abbildung 6.4 (a), allerdings gibt es ein paar feine Unterschiede. **Gewichte** verhält sich genau gleich wie beim automatischen Test. **Rocchio** steigt etwas schneller an als vorher, beziehungsweise schlägt es diesmal **Gewichte**, das bei fast jeder Query sehr früh stagnierte. Außerdem ist nun **NN** ab der zweiten RF-Iteration gleich gut wie **BQS**. Beim subjektivem Test war besonders gut zu beobachten, wie unterschiedlich

[1] Die *OeMV*-Datenbank beinhaltet historische Ausstellungsstücke des *Österreichischen Museums für Volkskunde, Wien.*

die Ergebnisse dieser beiden Methoden sind. Veranschaulicht wird dies in Abbildung 6.8 (b), wo die Ergebnisse von den beiden Queries **cars-red** und **cars-side** nach vier RF-Iterationen mittels den jeweiligen RF-Mechanismen gegenübergestellt werden. Die Kombination der beiden Methoden **NN & BQS** schneidet wiederum am besten ab. Überraschend ist noch, dass **NN** ein bisschen besser abschneidet als **NN & reliable**, da **NN & reliable** eine Verbesserung von **NN** sein soll. Der Unterschied ist allerdings erst nach der vierten RF-Iteration sichtbar und sehr klein, also vermutlich auf die geringe Anzahl von Queries zurückzuführen.

Die Standardabweichungsgraphen der Präzision in den verschiedenen RF-Iterationen sind in Abbildung 6.6 (b) nicht so aussagekräftig wie in Abbildung 6.4 (b). Grundsätzlich sind alle Standardabweichungswerte um ein Vielfaches höher, da die Stichprobe sehr klein ist und die Bilder mit Absicht so gewählt wurden, dass sie ein großes Spektrum von Query-Typen abdecken. Aufgrund der geringen Anzahl von Queries lassen sich nur schwer weitere Aussagen über die einzelnen Methoden anhand derer Standardabweichungsgraphen treffen. Auffällig ist noch, dass **BQS**, welches beim automatischen Test die höchsten Standardabweichungswerte aufweist, nun ab der zweiten RF-Iteration die geringsten Standardabweichungswerte produziert.

Die Ergebnisse des subjektiven Tests bekräftigen die Wahl von **BQS** als RF-Mechanismus für unser System. Die RF-Zeit für die **NN** Methoden war bei den kleinen Datenbanken *Lentos* und *Cars* unangenehm und bei den größeren Datenbanken *Madias* und *OeMV* fast unerträglich.

Bei der Analyse nach den durchschnittlichen Präzisionswerten sind die in Abbildung 6.7 (a) und (b) dargestellten Extremfälle von Queries nicht zu vergessen. In manchen Fällen, wie zum Beispiel bei der Query **lentos-box**, versagen alle RF-Mechanismen, weil entweder zu wenige Bilder von Boxen in der Datenbank sind oder der verwendete Deskriptor über zu geringe Ausdrucksstärke verfügt und es daher leicht zu Verwechslungen mit Bildern, in deren Zentrum viereckige Objekte vorhanden sind, kommt. In anderen Fällen ist die Query, wie zum Beispiel bei **oemv-knife**, sehr einfach und nahezu alle RF-Mechanismen verbessern die Ergebnisse enorm.

Zu guter Letzt lässt sich die Macht von RF an Abbildung 6.8 (a) und (b) demonstrieren. Abbildung 6.8 (a) zeigt die Ergebnisse der beiden Queries **cars-red** und **cars-side** ohne RF, welche beide vom selben Querybild (rot umrandet) starten. Durch die Verwendung von RF kann der Benutzer die Query in seine gewünschte Richtung lenken. Sucht er/sie, wie zum Beispiel bei **cars-red** nach roten Autos, so markiert er/sie stets Bilder, die rote Autos enthalten als positiv und andere als negativ und landet nach vier RF-Iterationen bei den Ergebnissen, die in Abbildung 6.8 (b) auf der linken Seite zu sehen sind. Sucht er/sie aber nach Seitenaufnahmen, so markiert er/sie einfach stets alle Seitenaufnahmen von Autos als positiv und andere als negativ und landet nach vier RF-Iterationen bei den Ergebnissen, die in Abbildung 6.8 (b) auf der rechten Seite zu sehen sind.

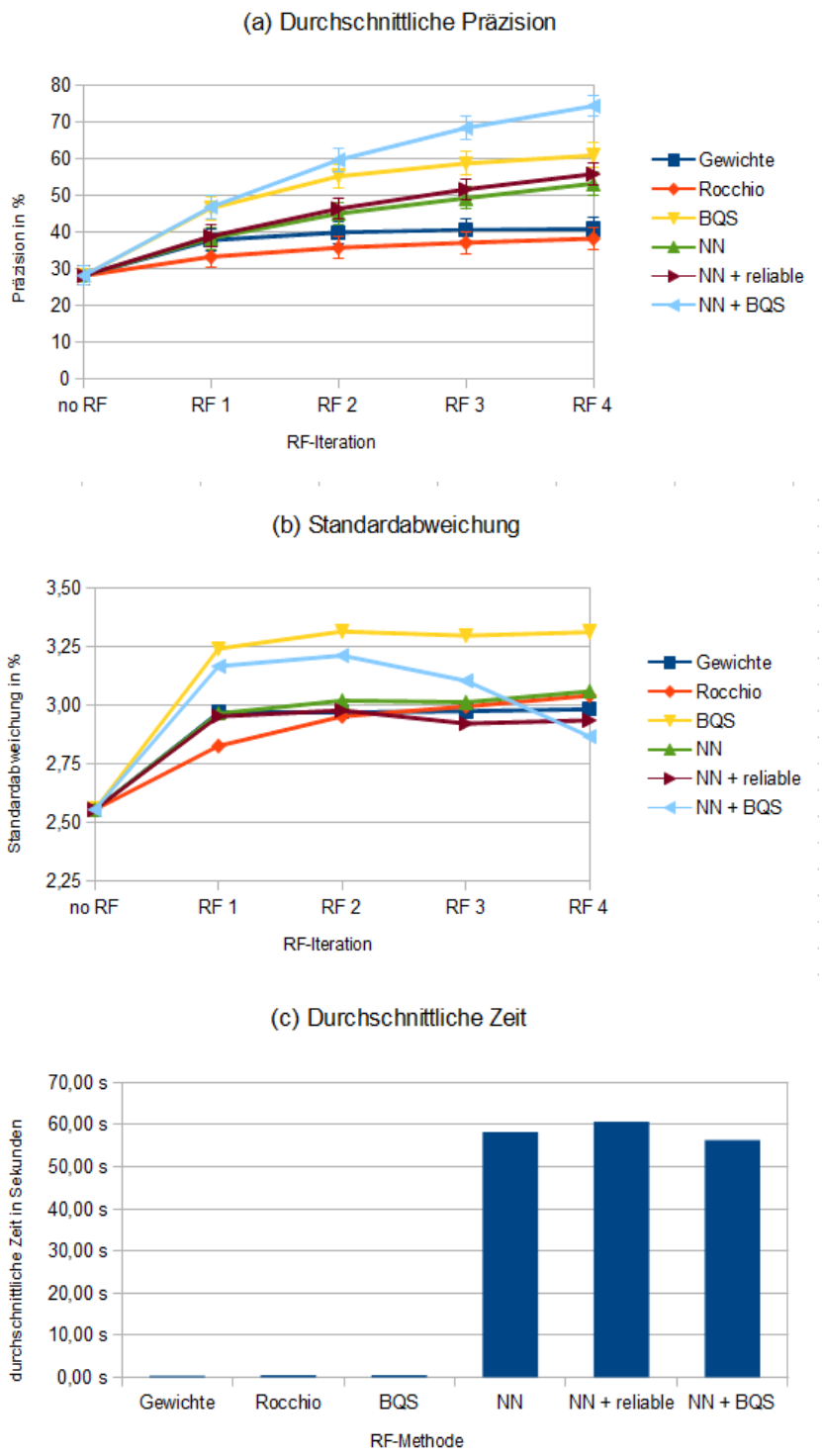


Abbildung 6.4: (a) Durchschnittliche Präzision, (b) Standardabweichung, (c) durchschnittliche Zeit einer RF-Iteration der verschiedenen RF-Methoden gemessen über 500 Queries mit jeweils 4 RF-Iterationen in der Corel Datenbank.

Abbildung 6.5: Beispiel für Bilder mit ähnlichem Inhalt und unterschiedlichen Objektkategorien in der Corel Datenbank (Die Katze links ist in der Klasse *Wildkatze* und die Katze rechts in der Klasse *Hauskatze*)

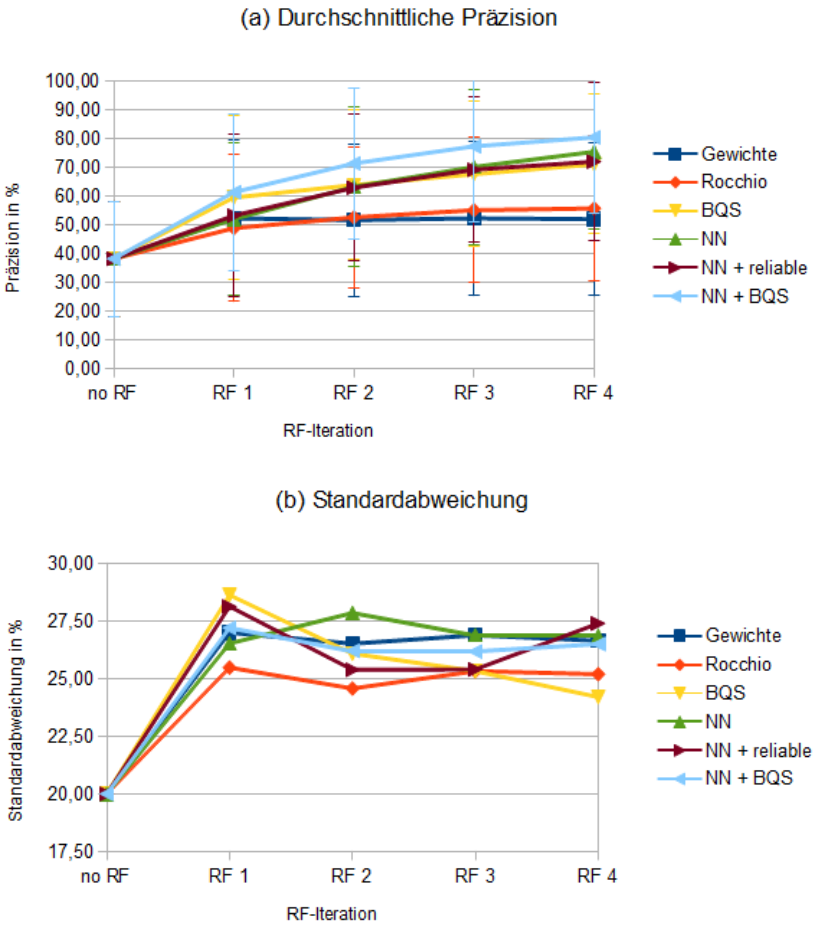
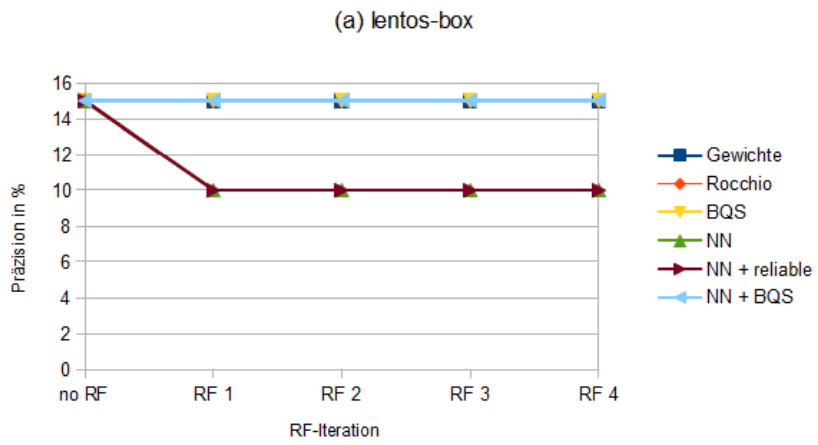


Abbildung 6.6: (a) Durchschnittliche Präzision, (b) Standardabweichung von 16 Queries mit jeweils 4 RF-Iterationen in den 4 M-BOX Datenbanken.



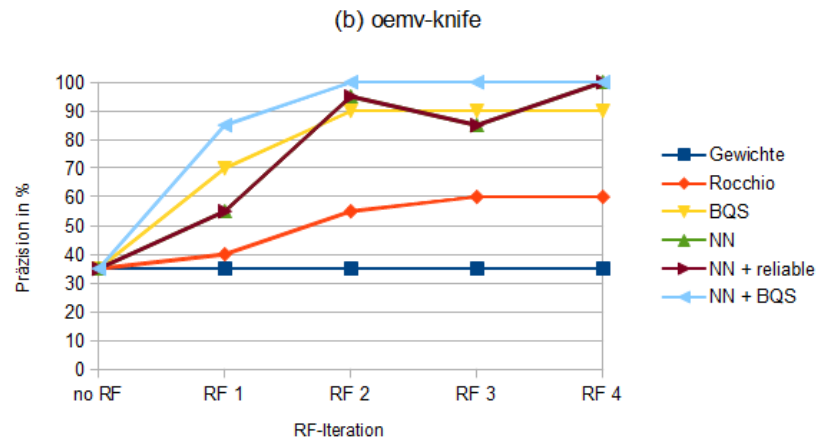


Abbildung 6.7: (a) Beispiel für eine Query, bei der kein RF-Mechanismus die Ergebnisse verbessern konnte. (b) Beispielquery, bei der RF sehr gut funktioniert.

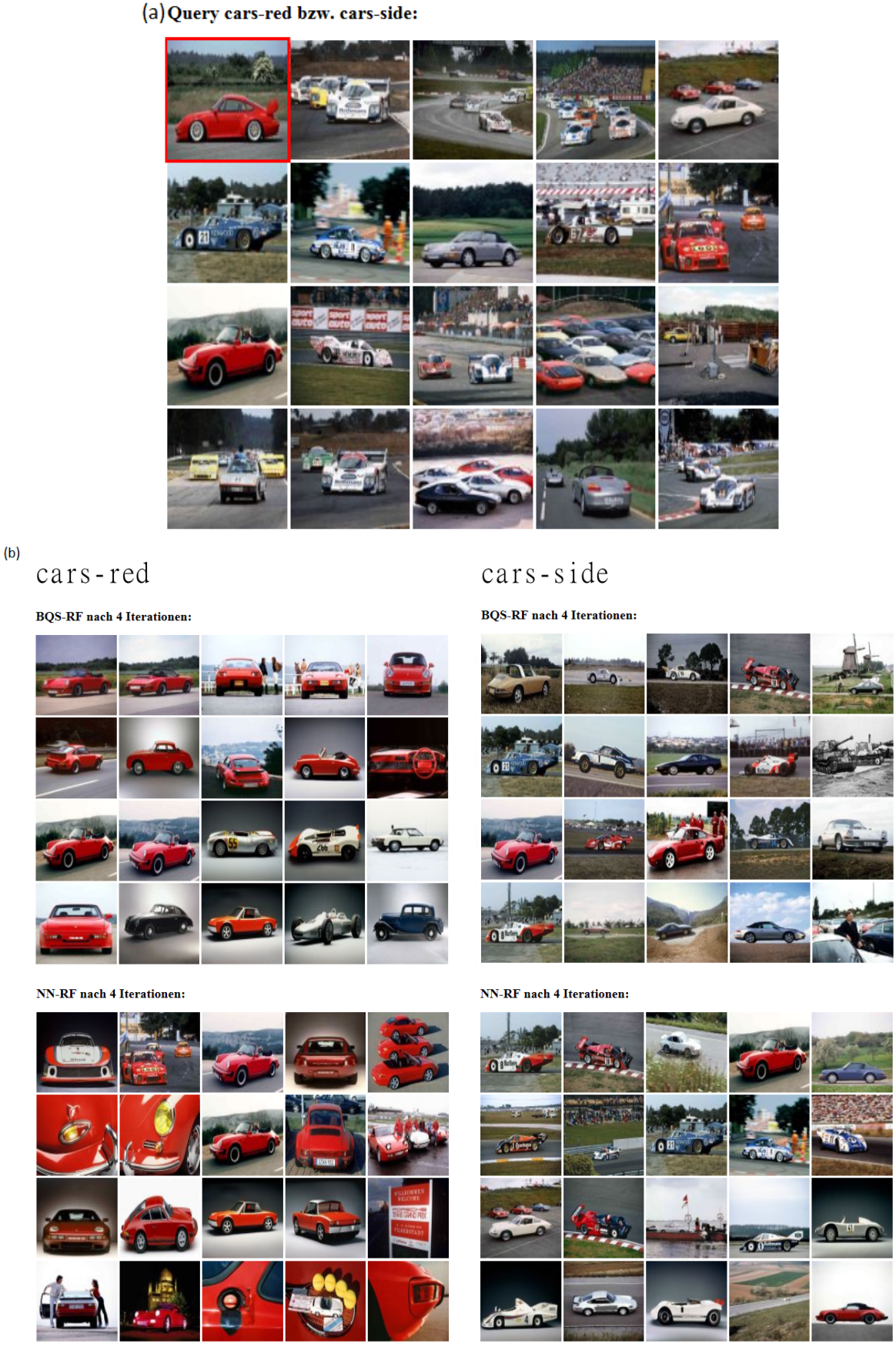


Abbildung 6.8: (a) Ergebnismenge der Query **cars-red** bzw. **cars-side** ohne RF. (b) Ergebnismenge der Query **cars-red** bzw. **cars-side** nach 4 RF-Iterationen mittels BQS- und NN-RF.

Kapitel 7

Implementierung

Die grundlegende Architektur unseres CBIR-Systems ist an das Modell aus Abbildung 7.1 angelehnt, wobei nicht alle Schritte neu implementiert wurden, sondern von anderen Tools übernommen werden. Details zu den einzelnen Modulen des Programmes werden im folgenden Teil beleuchtet.

7.1 Funktionsweise und Details

Bei der Implementierung des CBIR-Systems wurde besonderen Wert auf Modularität und Austauschbarkeit gelegt. Für die einzelnen Module (Deskriptoren, Metriken, Relevance Feedback und Indexing) wurden Interfaces definiert, die die Integration in andere Programme erleichtern soll.
Das Programm wurde in Java 7 geschrieben, der Quellcode ist unter folgender Adresse verfügbar: `http://code.google.com/p/simple-cbir/`.

Die Funktionsweise des CBIR-Systems sieht im Detail wie folgt aus (parallel zu den Schritten aus Abbildung 7.1):

- **Bilddatenbanken:** Grundsätzlich können jegliche Arten von Bilddatenbanken benutzt werden, jedoch ist eine Normalisierung der Bilddimensionen zu empfehlen, da möglicherweise die Ergebnisse verfälscht werden (deskriptorabhängig). Hier ist das Freeware-Programm *Irfanview*[1] zu empfehlen, welches eine gute *resize*-Funktion und Stapelverarbeitung bietet, alternativ kann auch *imagemagick* [2] verwendet werden.

- **Deskriptoren und Extraktion von Features:** Auch die Extraktion der Features ist vom CBIR-Programm vorausgesetzt und daher nicht implementiert. Verwendete Tools zum Extrahieren von Features sind *img(Rummager)*[3] und *FIRE*[4], die die Featuredatenbanken in verschiedenen Formaten speichern (eigene .gz- bzw. XML-Dateien).
  Folglich ist der erste Schritt unseres CBIR-Systems, diese Featuredatenbanken mittels Parsern einzulesen. Dabei wurde neben einem selbst implementierten .gz-Parser ein Open-Source-XML-Parser, *dom4j*[5], verwendet (Paket *cbir.reader*).

[1] `http://irfanview.tuwien.ac.at/`
[2] `http://www.imagemagick.org/script/index.php`
[3] `http://chatzichristofis.info/?page_id=213`
[4] `http://thomas.deselaers.de/fire/`
[5] `http://dom4j.sourceforge.net/`

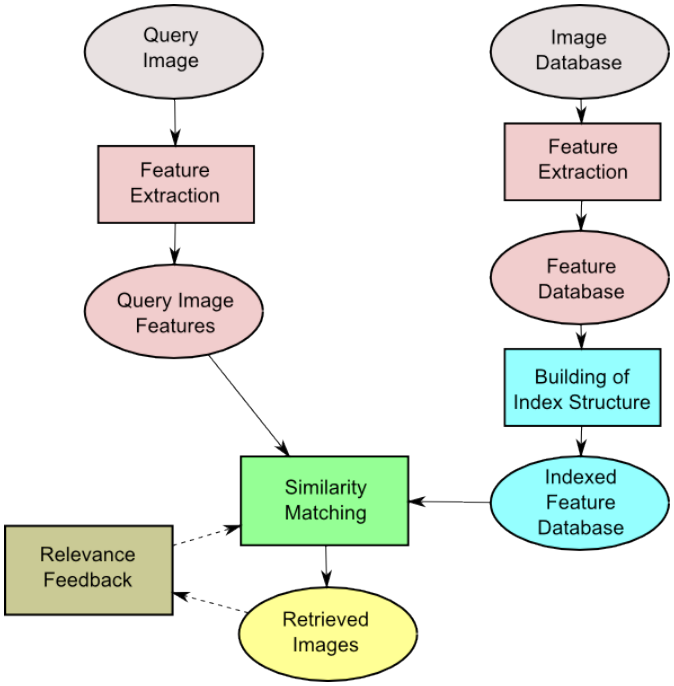


Abbildung 7.1: Allgemeines Modell eines CBIR-Systems

- **Indexing:** Nach dem Einlesen der Featuredatenbanken wird der k-d-Baum aufgebaut. Die Implementierung des k-d-Baumes ist an eine Open-Source-Version[6] angelehnt (Paket *ind.kdtree*).
- **Suche:** Die Featuredatenbank ist nun bereit für die Suche, durch die Spezifikation einer Query und einer Distanzmetrik (im Vorhinein, Pakete *cbir* bzw. *cbir.metric*) startet die Suchiteration und die Resultate werden ausgegeben.
- **Relevance Feedback:** Nach der Ausgabe der Resultate kann beliebig oft RF eingesetzt werden, um die Resultate zu verbessern (in manchen Fällen können sie auch schlechter werden). Implementiert wurden die vorgestellten RF-Methoden *Gewichte*, *Rocchio*, *BQS* und die *NN*-basierten Methoden (Pakete *rf.mars*, *rf.bayesian* bzw. *rf.nn*).

[6] `https://code.google.com/p/java-algorithms-implementation/source/browse/trunk/src/com/jwetherell/algorithms/data_structures/KdTree.java?r=368`

Kapitel 8

Schlussfolgerung

Diese Bachelorarbeit zeigt, dass es möglich ist, mit simplen Technologien ein CBIR-System zu erstellen, welches in angemessener Zeit brauchbare Ergebnisse liefert. Für die Entwicklung unseres Systems wurden verschiedene Deskriptoren, Metriken, RF-Verfahren und Indexstrukturen analysiert bzw. getestet. Dabei war folgendes auffällig:

- Für unsere Datenbanken stellte sich heraus, dass von den getesteten Deskriptoren die farbbasierten Deskriptoren und der MPEG7-EHD-Deskriptor am besten abschnitten. Außerdem konnte gezeigt werden, dass das Kombinieren mehrerer Deskriptoren, um all deren Stärken auszunutzen, funktioniert. Allerdings geschieht das Verbessern der Ergebnisse auf Kosten der Performance, wenn der flache Ansatz ohne weitere Optimierungen verwendet wird.

- Des Weiteren konnte beobachtet werden, dass die Wahl der Distanzmetrik Einfluss auf die Ergebnisse hat. Die gewichtete Euklidische Distanz stellte sich als beste Variante für die von uns getesteten Datenbanken heraus.

- Die größte Herausforderung der Arbeit war das Auffinden einer geeigneten Indexstruktur, die einfach und effizient ist. Der getestete k-d-Baum scheiterte bei kleinen Datenbanken am Fluch der Dimensionalität. Es konnte gezeigt werden, dass die lineare Suche in diesen Fällen schneller ist. Zusätzliche Forschung scheint hier erforderlich, ein vielversprechender Ansatz ist das *Locality-based Hashing* von Wang and Wang (2009).

- Grundsätzlich verbesserten alle von uns untersuchten RF-Verfahren die Ergebnisse. Mit steigender Komplexität der RF-Methoden (**NN**-Methoden) explodierte jedoch die Rechenzeit, weshalb bei der Auswahl der endgültigen RF-Methode für das System ein Kompromiss zwischen Rechenzeit und Präzision geschlossen werden musste. Die bayesische Querypunktverschiebung vereint exzellente Query-Ergebnisse mit sehr geringem Berechnungsaufwand und wird deshalb in der Endversion unseres CBIR-Systemes verwendet. Empfehlenswert wäre es, die neuesten RF-Methoden, welche RF als Klassifikationsproblem auffassen, bezüglich derselben Kriterien (Präzision/Zeit) zu analysieren, wobei die von Tao et al. (2006a) und Tao et al. (2006b) vorgestellten Verfahren als State-of-the-Art-Technologien angesehen werden.

Abschließend empfehlen wir weitere Forschungen zum Thema effizienter Indexstrukturen für CBIR-Systeme und zum Thema RF als Klassifikationsproblem, die bei Kapitel 5.2 und Kapitel 6.4 ansetzen. Außerdem wäre die Analyse weiterer Deskriptoren bezüglich ihrer Präzision und Kombinierbarkeit und auch die Analyse der verschiedenen RF-Methoden mit einer höheren Anzahl von RF-Iterationen interessant.

Literaturverzeichnis

Miguel Arevalillo-Herraez and Francesc J. Ferri. An improved distance-based relevance feedback strategy for image retrieval. *Image and Vision Computing*, 31(10):704 – 713, 2013. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/j.imavis.2013.07.004. URL `http://www.sciencedirect.com/science/article/pii/S026288561300108X`.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL `http://doi.acm.org/10.1145/361002.361007`.

Markus Bollwerk. Relevance feedback for content-based image retrieval. Technical report, RWTH-Aachen, 2005.

Savvas A. Chatzichristofis and Yiannis S. Boutalis. Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *Proceedings of the 6th international conference on Computer vision systems*, ICVS'08, pages 312–322, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-79546-4, 978-3-540-79546-9. URL `http://dl.acm.org/citation.cfm?id=1788524.1788559`.

Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Inf. Retr.*, 11(2):77–107, April 2008. ISSN 1386-4564. doi: 10.1007/s10791-007-9039-3. URL `http://dx.doi.org/10.1007/s10791-007-9039-3`.

Giorgio Giacinto. A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 456–463, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-733-9. doi: 10.1145/1282280.1282347. URL `http://doi.acm.org/10.1145/1282280.1282347`.

Piotr Indyk. Nearest neighbors in high-dimensional spaces, 2004.

Raghu Ramakrishnan Kevin Beyer, Jonathan Goldstein and Uri Shaft. When is "nearest neighbor"meaningful? In *In Int. Conf. on Database Theory*, pages 217–235, 1999.

Mathias Lux and Savvas A. Chatzichristofis. Lire: lucene image retrieval: an extensible java cbir library. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 1085–1088, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-303-7. doi: 10.1145/1459359.1459577. URL `http://doi.acm.org/10.1145/1459359.1459577`.

Alessandro Moschitti. A study on optimal parameter tuning for rocchio text classifier. In *Proceedings of the 25th European conference on IR research*, ECIR'03, pages 420–435, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-01274-5. URL `http://dl.acm.org/citation.cfm?id=1757788.1757828`.

Stephen O'Hara and Bruce A. Draper. Introduction to the bag of features paradigm for image classification and retrieval. *CoRR*, abs/1101.3354, 2011.

Sharadh Ramaswamy and Kenneth Rose. Towards optimal indexing for relevance feedback in large image databases +, 2009.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000. ISSN 0920-5691. doi: 10.1023/A:1026543900054. URL `http://dx.doi.org/10.1023/A:1026543900054`.

Yong Rui and T. Huang. Optimizing learning in image retrieval. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 236–243 vol.1, 2000. doi: 10.1109/CVPR.2000.855825.

Yong Rui, T.S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 815–818 vol.2, 1997. doi: 10.1109/ICIP.1997.638621.

L. Shoukry, S.Klenk, and G. Heidemann. Mpeg-7 feature visualization for cbir systems. In *Proc. Int. Conf. on Computer Theory and Applications (ICCTA 2010)*, pages 86–90, Alexandria, Egypt, 2010.

Dacheng Tao, Xiaoou Tang, Xuelong Li, and Yong Rui. Direct kernel biased discriminant analysis: a new content-based image retrieval relevance feedback algorithm. *Multimedia, IEEE Transactions on*, 8(4):716–727, 2006a. ISSN 1520-9210. doi: 10.1109/TMM.2005.861375.

Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1088–1099, 2006b. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.134.

N. Vasconcelos and A. Lippman. Bayesian relevance feedback for content-based image retrieval. In *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, pages 63–67, 2000. doi: 10.1109/IVL.2000.853841.

Weihong Wang and Song Wang. A method using locality-sensitive hashing for large-scale content-based image retrieval. In *Proceedings of the 21st annual international conference on Chinese control and decision conference*, CCDC'09, pages 1871–1875, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2722-2. URL `http://dl.acm.org/citation.cfm?id=1715350.1715375`.

Anhang A

Notation

In diesem Anhang befindet sich eine kurze Übersicht über die am öftesten verwendeten Bezeichner in den präsentierten Formeln.

A.1 Deskriptoren

- I bezeichnet ein Bild aus der Datenbank.
- $\mathbf{I}$ bezeichnet den zugehörigen Deskriptor für ein Bild aus der Datenbank, der Werte mehrerer Features enthält.
- Q bezeichnet ein Querybild.
- $\mathbf{Q}$ bezeichnet den Deskriptor eines Querybildes.

A.2 Metriken

- $d(\mathbf{Q}, \mathbf{I})$ bezeichnet die Distanz zwischen den Deskriptoren $\mathbf{Q}$ und $\mathbf{I}$.
- $sim(\mathbf{Q}, \mathbf{I})$ bezeichnet die Ähnlichkeit zwischen den Deskriptoren $\mathbf{Q}$ und $\mathbf{I}$.
- W bezeichnet eine Diagonale Gewichtsmatrix.

A.3 Relevance Feedback

- $\mathbf{Q}^{(i)}$ bezeichnet die i-te Version einer Query.
- $\mathbf{D}$ bezeichnet einen Gewichtsvektor, der aus einem Deskriptor gewonnen wird. Die Gewichtsvektoren werden ähnlich wie ein Deskriptor zum vergleichen von Bildern verwendet.
- $M_R(\mathbf{Q})$ bezeichnet die Menge von Bildern der Ergebnismenge für die Query $\mathbf{Q}$, die als relevant markiert wurden.
- $M_N(\mathbf{Q})$ bezeichnet die Menge von Bildern der Ergebnismenge für die Query $\mathbf{Q}$, die als irrelevant markiert wurden.
- N_R bezeichnet die Anzahl der Bilder in $M_R(\mathbf{Q})$.
- N_N bezeichnet die Anzahl der Bilder in $M_N(\mathbf{Q})$.

- N bezeichnet die Anzahl der vom Benutzer markierten Bilder, es gilt $N = N_R + N_N$.
- $NN_R(\mathbf{I})$ bezeichnet den nächsten als relevant markierten Deskripor zum Bild I.
- $NN_N(\mathbf{I})$ bezeichnet den nächsten als irrelevant markierten Deskripor zum Bild I.
- N bezeichnet die Anzahl der vom Benutzer markierten Bilder, es gilt $N = N_R + N_N$.

Anhang B

Arbeitsverteilung

Kapitel	**Autor**
Kurzfassung	Chris Wendler & Matej Stanic
Danksagungen	Chris Wendler & Matej Stanic
1. Einleitung	Chris Wendler & Matej Stanic
2. Deskriptoren	Chris Wendler
2.1 Typen von Deskriptoren	Chris Wendler & Matej Stanic
2.2 Kombinierte Deskriptoren	Chris Wendler
2.3 Der Fluch der Dimensionalität	Matej Stanic
2.4 Auswahl von relevanten Deskriptoren für CBIR-Systeme	Matej Stanic
Die Evaluierungsergebnisse von 2.4	Chris Wendler & Matej Stanic
2.5 Verwendete Deskriptoren	Matej Stanic
2.5.1 Color Histogram	Chris Wendler
2.5.2 MPEG7-EHD	Matej Stanic
2.5.3 CEDD	Matej Stanic
3. Metriken	Chris Wendler
3.3 Earth Mover's Distance	Matej Stanic
4. Evaluierung der verwendeten Deskriptoren und Metriken	Matej Stanic
5. Indexing	Matej Stanic
6. Relevance Feedback	Chris Wendler
7. Implementierung	Chris Wendler & Matej Stanic
8. Schlussfolgerungen	Chris Wendler & Matej Stanic
Notation	Chris Wendler

Tabelle B.1: Arbeitsverteilung bezüglich der einzelnen Kapitel