Untersuchungen zu semantischem Retrieval von Bildern mit Hilfe von MPEG-7 und Implementation einer performanceoptimierten Beispielapplikation

- Diplomarbeit -
an der
Professur Medieninformatik

vorgelegt von

Daniel Pötzinger[1]

28. September 2009

Erstgutachter und Betreuer : Prof. Dr. Maximilian Eibl
TU-Chemnitz
Fakultät für Informatik
Professur Medieninformatik
D-09107 Chemnitz

[1] E-Mail: poetzinger@aoemedia.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In der heutigen Welt spielen multimediale Inhalte eine immer größere Rolle: moderne Kommunikations- und Produktionstechnologien haben dazu beigetragen, dass wir eine immer größer werdende Menge von multimedialen Inhalten haben. Diese wachsenden Datenmengen sinnvoll speichern und erschließen zu können, stellt eine wichtige Herausforderung dar.

Ist man auf der Suche nach bestimmten Inhalten, so ist zurzeit immer noch text- bzw. schlagwortbasierte Suche die häufigste Wahl. Um jedoch auch in Zukunft die Fülle der Daten möglichst optimal nutzen und erschließen zu können, ist es wichtig auch semantisch vielfältigere Suchanfragen nutzen zu können.

Denkbar sind beispielsweise Bildsuchen anhand von Skizzen und Beispielbildern, oder das Vorsummen einer gesuchten Melodie. Betrachtet man zum Beispiel „Flickr“, eine der bekanntesten Fotocommunities im Internet mit über 3 Milliarden Bildern[1]: zurzeit hat man keine Möglichkeit Bilder anhand von Farben oder Objekten zu finden und eine textbasierte Suche führt nicht immer zum Ziel.

Es gibt bereits viele Untersuchungen und einige Anwendungen in diesem Bereich. Ein viel versprechender Standard zur Beschreibung verschiedenster Aspekte multimedialer Inhalte ist MPEG-7. Die verschiedenen in einer MPEG-7 Beschreibung verfügbaren Informationen können genutzt werden, um unterschiedlichste semantische Abfragen durchführen zu können und somit deutlich helfen, gesuchte Daten zu finden.

In dieser Arbeit wird der Standard MPEG-7 untersucht und verwendet, um am Beispiel von visuellen Eigenschaften – wie Farbe und Farbverteilung – ähnlichkeitsbasierte Suchen durchzuführen. Da MPEG-7 Beschreibungen im XML Format vorliegen, wird auf verschiedene Speichermöglichkeiten von XML in Datenbanken sowie auch auf die Nutzung und Performance von XQuery eingegangen.

Im Rahmen der Arbeit wurde eine Beispielapplikation zur semantischen Bildsuche implementiert, mit der es unter anderem möglich ist, anhand eines gegebenen Bildes ähnlichkeitsbasierte Suchanfragen durchzuführen. Die Applikation sowie die zu Grunde liegenden Konzepte und Funktionsweisen werden in Kapitel 5 beschrieben.

[1] Blogeintrag von Flickr Ende 2008: http://blog.flickr.net/en/2008/11/03/3-billion/ (Abruf am 10. August 2009)

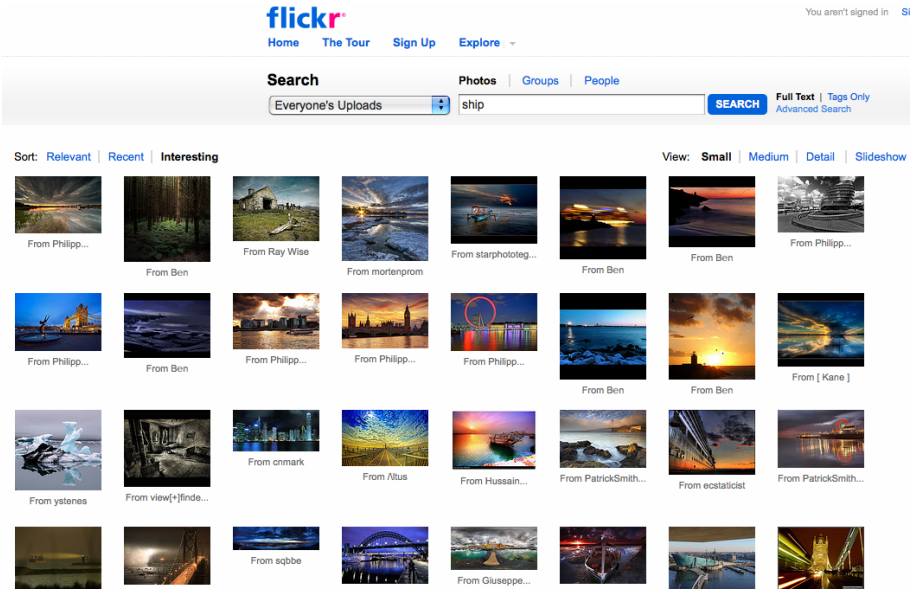

Abbildung 1.1: Suche nach „ship“ auf flickr bringt nicht unbedingt die gesuchten Ergebnisse.

Neben der Möglichkeit semantische Suchen zu definieren und durchzuführen, ist Performanceoptimierung ein wichtiges Thema der Arbeit. Es werden mögliche Optimierungsmaßnahmen diskutiert und der durch Clustering erfolgreich erreichte Performanceschub beschrieben.

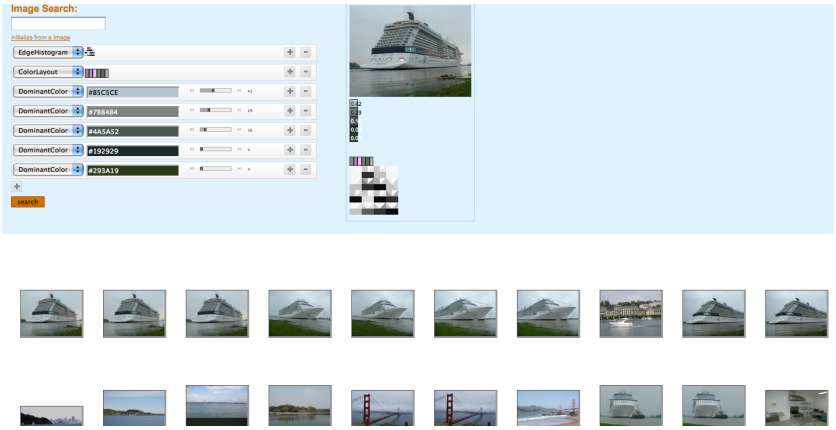

Abbildung 1.2: Suche nach einem Schiff in der Beispielapplikation durch „Query by Example“

2 MPEG-7

2.1 Einführung

Multimediale Inhalte zu produzieren ist einfacher als je zuvor und die Anzahl an produzierten multimedialen Inhalten - wie beispielsweise Bilder, Videos und Musik - wächst unaufhörlich. Doch die Fülle solcher Inhalte hat nur einen Wert, wenn man sie auch entdecken und nutzen kann. Dies ist wohl eine der Gründe, die die Moving Picture Experts Group dazu bewegte den Standard MPEG-7 zu verfassen.

Formal wird der Standard auch "Multimedia Content Description Interface" (ISO/IEC 15938) genannt und er behandelt, im Unterschied zu den vorangegangenen Standards, nicht die Komprimierung oder technischen Auslieferung multimedialer Daten. MPEG-7 standardisiert die Beschreibung verschiedener Typen multimedialer Inhalte unabhängig von deren Repräsentation oder Speicherung. Eine MPEG-7 Beschreibung stellt also eine Repräsentation der Informationen eines multimedialen Inhaltes dar. Die Beschreibung reicht dabei von low-level Beschreibungen, wie beispielsweise die Angabe der Farbverteilung, bis hin zu high-level Beschreibungen, wie beispielsweise die Angaben von Ort, Zeit und Personen.

Der Standard selbst ist in acht Teile aufgeteilt, die sich jeweils mit unterschiedlichen Teilaspekten beschäftigen. Offiziell startete die Spezifikation des Standards 1997 und lief in verschiedenen Phasen ab. Zu Beginn wurde eine Reihe von Zielgrundlagen definiert. Auszugsweise genannt seien folgende:

- Unterstützung für eine große Anzahl verschiedener Anwendungen
- Große Anzahl unterstützter Inhaltstypen
- Getrennte oder kombinierte Speicherung vom beschriebenem Inhalt
- Erweiterbar für zukünftige Anforderungen oder spezielle Anwendungen

[1, S. 3-11]

In diesem Kapitel werden die meisten Konzepte und Elemente von MPEG-7 vorgestellt. Zusammen mit konkreten Beispielausschnitten einer XML MPEG-7 Beschreibung und den, im Anhang zu findenden, Diagrammen zum Aufbau von MPEG-7 Beschreibungen soll ein Gesamtüberblick vermittelt werden.

2.2 Tools

In MPEG-7 werden verschiedene normative Elemente oder auch Tools unterschieden, die, falls implementiert, standardkonform implementiert sein müssen um die Interoperabilität zu gewährleisten. Mit diesen Tools kann man eine MPEG-7 Beschreibung erzeugen. Diese besteht aus einen oder mehreren Description Schemes Instanzen, die jeweils voll oder teilweise instanziert sein können, d.h. das allen oder nur einigen Descriptoren Attribute zugewiesen sind.

- **Descriptor (D)**
  - Ist eine Repräsentation eines Merkmals (im Sprachgebrauch Feature). Wobei Feature als eine bestimmte Charakteristik der Daten[1] verstanden wird.
  - Man unterscheidet low-level Merkmale (wie beispielsweise Farbe, Textur, Bewegung, Tonenergie) und high-level Merkmale semantischer Objekte (wie Ereignis, Inhaltsmanagement oder Informationen zur Speicherung). Es wird erwartet, dass low-level Descriptoren automatisch erzeugt werden können und für high-level Descriptoren menschliche Hilfe erforderlich ist.
  - Ein Descriptor definiert die Syntax und die Semantik der Featurerepräsentation.
  - Er erlaubt eine Bewertung des korrespondierenden Merkmals durch seinen Wert. Es ist möglich mehrere Descriptoren zu haben, die sich auf ein Merkmal beziehen, zum Beispiel um verschiedenen Anforderungen zu genügen.
  - Korrespondiert im MPEG-7 audiovisuellem konzeptualem Modell mit Attributen.
  - Es gibt keine many-to-one Beziehungen zu anderen Descriptoren
- **Description Schemes (DS)**
  - Description Schemes bauen auf Descriptors auf, kombinieren individuelle Descriptors und Description Schemes und beschreiben die Struktur und Semantik der Beziehung zwischen diesen.
  - Bietet die Möglichkeit die Struktur und Semantik multimedialer Informationen zu modellieren.

[1] Daten sind in diesem Zusammenhang audiovisuelle Informationen die durch MPEG-7 beschrieben werden

  - Ein einfaches Beispiel wäre ein Film mit Szenen und Schnappschüssen, der einige textuelle Descriptoren auf der Szenenebene beinhaltet und Farb- sowie Bewegungs- Descriptoren auf der Schnappschussebene.

- **Description Definition Language**

  - Erlaubt die Erzeugung neuer sowie die Veränderung vorhandener Description Schemes und Descriptoren.
  - Ist eine Erweiterung von XML Schema. Für die Definition von D und DS wurden Namenskonventionen spezifiziert:
    * Wenn der Name aus verschiedenen Wörtern zusammengesetzt ist, wird der erste Buchstabe jedes Wortes groß geschrieben.
    * Elementnamen beginnen mit Großbuchstaben.
    * Attributnamen beginnen mit kleinen Buchstaben.
    * Namen komplexer Typen (complexType) beginnen mit einem Großbuchstaben und enden mit dem Suffix „Type“
    * Namen simpler Typen (simpleType) beginnen mit einem Kleinbuchstaben und dürfen mit dem Suffix „Type“ enden.

- **System Tools**

  - Tools für Binärdarstellung, Synchronisation, Transport und Speicherung sowie für das Management und für die Sicherung intellektuellen Eigentums.

2.3 Multimedia Description Schemes (MDS)

Die Description Schemes, die unter der Bezeichnung Multimedia Description Schemes zusammengefasst sind, erlauben die Beschreibung und Annotation von audiovisuellen Inhalten. Sie ermöglichen die standardisierte Beschreibung von audiovisuellen Inhalten und des Inhaltsmanagements und erleichtern somit das Suchen, Filtern und den Zugriff auf audiovisuelle Inhalte. Sie wurden von der MPEG als Teil des MPEG-7 Standard entwickelt.

Die MDS werden üblicher Weise, aufgrund Ihrer Funktionalität und Aufgaben, in die sechs verschiedenen, in Abbildung 2.1 gekennzeichneten, Bereiche gegliedert. Die einzelnen Bereiche sind Basic Elements, Content Description, Content Management, Content Organization, Navigation and Access und User Interaction gegliedert.[2] Die einzelnen Bereiche werden in den folgenden Kapiteln näher betrachtet.

[2]Da in der Literatur üblich, wurden die englischen Begriffe verwendet. Eine entsprechende Übersetzung

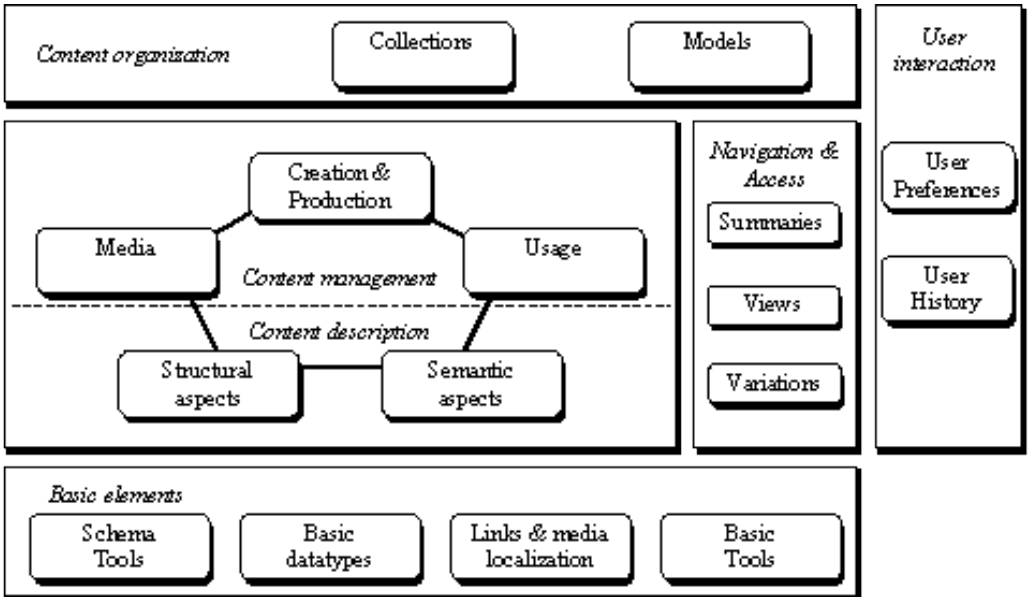

Abbildung 2.1: Überblick über die MDSs

2.4 Basic Elements

Die Spezifikation der MDS definiert eine Reihe grundlegender Elemente, die in anderen MPEG-7 Descriptoren wiederholt genutzt werden, diese werden unter Basic Elements zusammengefasst und stellen das Herz der MDS dar. Dazu gehören Schema-Tools, grundlegende Datentypen, Verweise und Lokalisierung sowie grundlegende Beschreibungs-Werkzeuge.

2.4.1 Schema Tools

Werden genutzt um valide Beschreibungen zu erzeugen und zu verwalten. Das Rootelement *Mpeg7* und die definierten top-level Elemente sind spezielle Elemente um eine valide MPEG-7 Beschreibung zu erzeugen.

Es werden zwei Arten valider Beschreibungen unterschieden: komplette Beschreibungen und partielle Beschreibungen, die nur bestimmte Description Units[3] für eine Applikation beinhalten. Komplette Beschreibungen enthalten eine semantisch komplette Beschreibung für eine bestimmte Anwendung. Die Top-Level-Elemente einer kompletten Beschreibung fungieren als Hülle für Tools, die für eine bestimmte Beschreibungsaufgabe passend

für die einzelnen Bereiche wäre: Basis Elemente, Inhalts-Beschreibung, Inhalts-Management, Inhalts-Organisation, Navigation und Zugriff sowie Nutzer-Interaktion.

[3]Eine Description Unit kann alle MPEG-7 Descriptions oder Description Schemes enthalten.

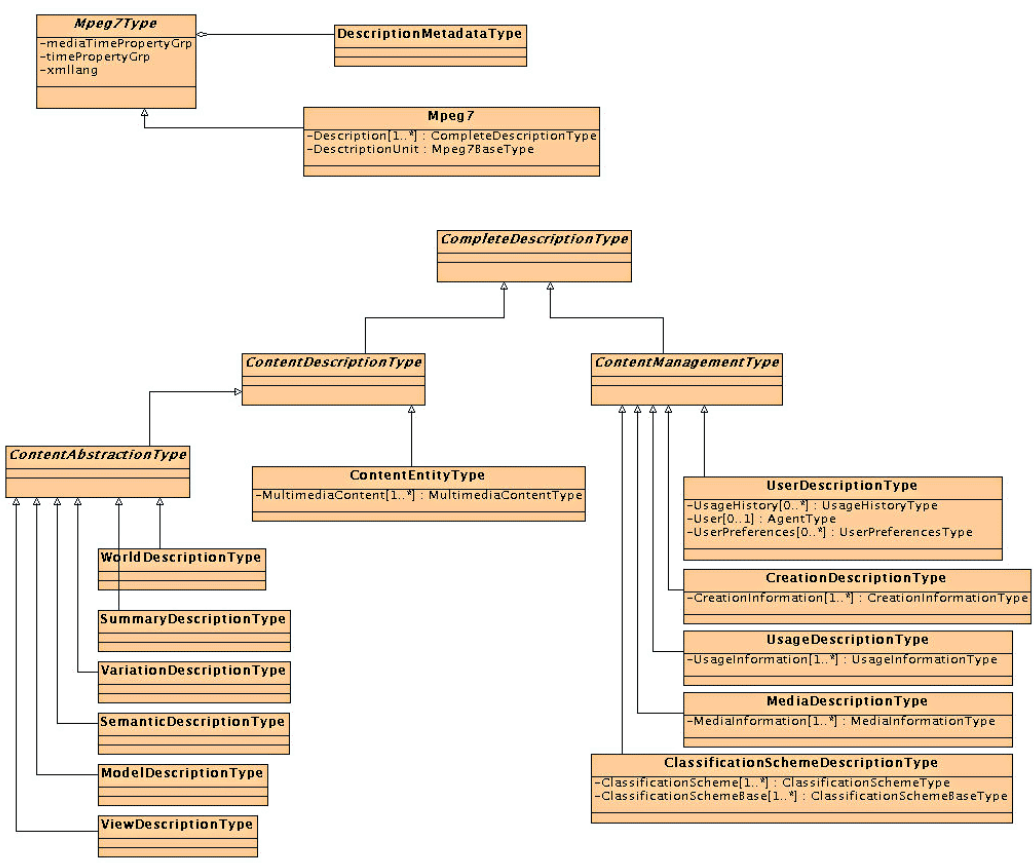

Abbildung 2.2: Die top-level Elemente eines validen MPEG-7 Dokumentes [4]

sind. Es werden drei Haupt-Beschreibungs-Aufgaben unterschieden: Content Entity Description, Content-Abstraction Description und Content-Management Description. Eine grafische Übersicht über die Organisation von MPEG-7 findet man in Abbildung 2.2. Die MPEG-7 Toplevel Elemente werden also nicht nur genutzt um valide MPEG-7 Beschreibungen zu erhalten, sondern repräsentieren auch eine Auswahl verschiedener Descriptors und Description Schemes, um die gebräuchlichsten Beschreibungs- und Verwaltungsaufgaben zu verrichten.

- **Content Entity Type:** Diese Elemente stellen ein Modell für die Beschreibung verschiedener Arten multimedialer Inhalte wie Bilder, Video oder Audio zur Verfügung. Insbesondere wichtig für das später behandelte Retrival von multimedialen Inhalten aufgrund ihrer Eigenschaften sind die Elemente Image (Typ StillRegion), Video (Typ VideoSegment) und Multimedia (Typ MultimediaSegment).
- **Content Abstraction Type:** Diese Elemente erlauben die Beschreibung von Zusammenfassungen des Inhalts, verschiedenen Sichten auf den Inhalt, Variationen

des Inhalts. Und sie beinhalten Real-Welt Modelle um, die zum multimedialem Inhalt gehörende Semantik zu beschreiben. Möchte man Inhalte nach semantischen Kriterien abfragen, sind diese Elemente relevant.

- **Content Management Type:** Diese Elemente stellen eine Menge von Tools für grundlegende Verwaltung von multimedialen Inhalten zur Verfügung. Darunter die Beschreibung der Erzeugung, der Klassifikation, der Verwendung und des Nutzers der Inhalte. Auch diese Elemente kann man für semantische Abfragen nutzen.

Unter die Schema Tools fällt auch die Beschreibung der verschiedenen Typen multimedialer Inhalte, dazu werden die so genannten MultimediaContent Entities definiert, eine Übersicht dieser findet man in Abbildung 2.3.

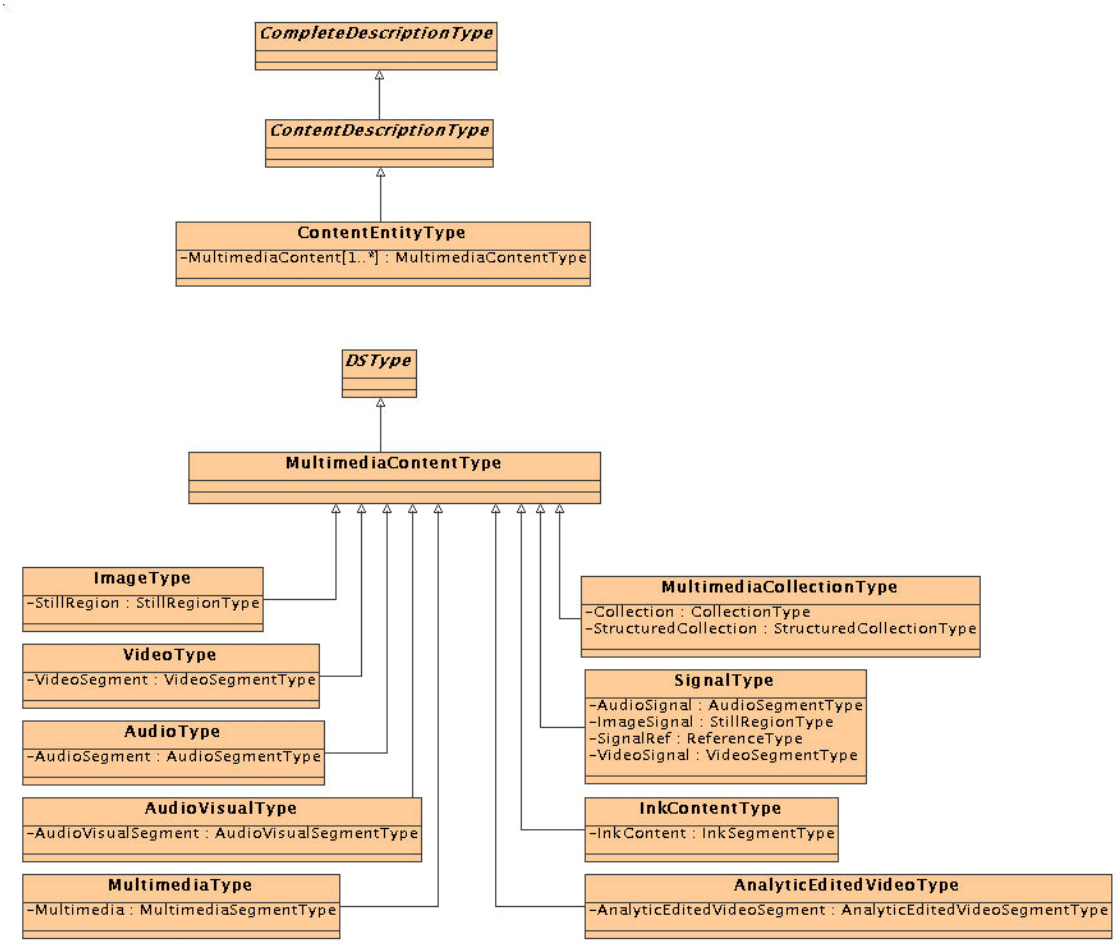

Abbildung 2.3: Die möglichen ContentEntityTypes einer MPEG-7 Beschreibung [4]

Des weiteren wird die Paketierung von Descriptors oder Description Schemes für Anwen-

dungen beschrieben und das DescriptionMetadata Ds, zur Beschreibung von Metadaten einer MPEG-7 Instanz, definiert.

2.4.2 Basic Data Types

Obwohl die DDL in MPEG-7 schon verschiedene allgemeine Datentypen einführt, benötigen Beschreibungen für multimediale Inhalte zusätzliche mathematische Konstrukte. Demnach gibt es in MPEG-7 die Datentypen Integer und Reals, Vektoren und Matrizen, Wahrscheinlichkeits-Vektoren und -Matrizen sowie Zeichenketten (Strings).

2.4.3 Linking und Lokalisierungs Tools

Hierunter zählen Tools für die Verlinkung und das Lokalisieren, die genutzt werden um Beschreibungen zu referenzieren und diese mit dem multimedialem Inhalt zu verlinken, und zum Darstellen von Zeit in Inhaltsbeschreibungen.

Referenzierung:
MPEG-7 stellt ein Tool zur Verfügung um auf ein beliebiges Beschreibungselement (ein Descriptor, Description Scheme oder ein Unterelement) zu verweisen. Dafür definiert man den Typ *ReferenceType* der die drei Referenziermethoden, *idref* ,*xpath* und *href* zur Verfügung stellt. Folgender Auschnitt zeigt die relevante Definition des ReferenceType Datentyps im XML Schema.

```
<!-- Definition of Reference datatype -->
<complexType name="ReferenceType">
  <attributeGroup ref="MPEG-7:referenceGrp"/>
</complexType>
<!-- Definition of referenceGrp attribute group -->
<attributeGroup name="referenceGrp">
  <attribute name="idref" type="IDREF" use="optional"/>
  <attribute name="xpath" type="MPEG-7:xPathRefType" use="optional"/>
  <attribute name="href" type="anyURI" use="optional"/>
</attributeGroup>
<!-- Definition of xPathRef datatype -->
<simpleType name="xPathRefType">
  <restriction base="MPEG-7:xPathType">
    <pattern value="/?((((child::)<... gekürzt ...>)"/>
  </restriction>
</simpleType>
```

Der ReferenceType kann in MPEG-7 an vielen Stellen benutzt werden, so stehen zu vielen Elementen entsprechende Elemente vom Typ ReferenceType zur Verfügung, um auf die entsprechende Beschreibung zu verweisen. Entsprechende Elemente enden meistens mit *Ref*, beispielsweise kann man für das Element *Event* auch *EventRef* verwenden. Das macht es später bei Abfragen gegebenenfalls nötig Referenzen aufzulösen.

Medien identifizieren und lokalisieren:
Um zu Medien zu verweisen, gibt es in MPEG-7 zwei Möglichkeiten:

- Mit dem UniqueID Datentyp kann man implizit auf ein Medium verweisen. (Identifikation)
- Mit den media locator (Medien Lokalisierungs) Datentypen kann man explicit auf ein Medium verweisen. Es gibt drei Arten von media locators:
  - allgemeiner MediaLocator, mit der Möglichkeit über einen URI auf multimediale Inhalte zu verweisen oder den Inhalt direkt einzubinden.
  - TemporalSegmentLocator um ein Segment in einem zeitlichem Medium zu lokalisieren.
  - ImageLocator um ein Bild oder einen einzelnen Frame in einem Video zu lokalisieren.

Zeit:
In MPEG-7 kann man zwei Arten von Zeit repräsentieren: media time, also die Zeit die für ein Medium gemessen wurde und world time, also die Zeit in der Welt. Zeitperioden werden dabei durch einen Startpunkt (mediaTimePoint Datentyp) und eine Dauer (mediaDuration Datentyp) bestimmt und basieren auf dem Standard ISO 8601.

2.4.4 Basic Tools

Unter Basic Tools kann man eine Art Bibliothek von Descriptions und Description Schemes verstehen, die als primitive Komponenten in anderen komplexeren und funktionsspezifischeren MPEG-7 Tools verwendet werden. Auf die einzelnen Elemente soll im folgenden näher eingegangen werden.

Relationen:
Mit einer Relation kann man die Beziehung zwischen einer oder mehrerer Description Scheme Instanzen beschreiben. In MPEG-7 sind alle Relationen gerichtet und unterteilt in Quell- und Ziel-Argumente. Der Typ einer Relation wird über ein Klassifikationsschema festgelegt. Es wird eine externe Form und eine interne Form der Relationsbeschreibung unterstützt.

Graphen:
Ein Graph kann mit Hilfe von Knoten (MPEG-7 Node Elemente) und Relationen beschrieben werden und wird für strukturelle und semantische Beschreibungen genutzt.

Text Annotation:
Für die textuelle Annotation multimedialer Inhalte steht der Datentyp TextAnnotation zur Verfügung, der verschieden strukturierte Anmerkungen erlaubt: FreeTextAnnotation, KeywordAnnotation, StructuredAnnotation und DependencyStructure. Im Retrival können die Anmerkungen für textbasierende Suchen ausgewertet werden, dabei eröffnen die verschiedenen Anmerkungstypen unterschiedliche Möglichkeiten, beispielsweise erlaubt die StructuredAnnotation eine für Applikationen besser zugängliche Semantik, durch die Aufteilung nach bestimmten Fragewörtern, als die FreeTextAnnotation, bei der lediglich ein frei wählbarer Text eingefügt werden kann.

Classification Schemes und Terms:
Es ist insbesondere für die Interoperabilität wichtig, dass man ein standardisiertes Vokabular in MPEG-7 Beschreibungen hat. Außerdem sollte es für Anwendungen möglich sein eigenes Vokabular hinzuzufügen. Dafür stellt MPEG-7 die Classification Schemes zur Verfügung, mit denen man mit Hilfe von Terms das Vokabular einer bestimmten Domäne beschreiben kann. MPEG-7 hat bereits verschiedene CS vordefiniert, wie zum Beispiel das TemporalRelation CS und das SpatialRelation CS um Zeiten und Orte zu beschreiben.

```
<!-- ################################################ -->
<!--  Definition of ClassificationScheme DS (7.3.2)  -->
<!-- ################################################ -->
<!-- Definition of ClassificationSchemeBase DS  -->
<complexType name="ClassificationSchemeBaseType" abstract="true">
  <complexContent>
    <extension base="MPEG-7:DSType">
      <sequence>
        <element name="Import" type="MPEG-7:ReferenceType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
      <attribute name="uri" type="anyURI" use="required"/>
      <attribute name="domain" use="optional">
        <simpleType>
          <list itemType="MPEG-7:xPathAbsoluteSelectorType"/>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
<!-- Definition of ClassificationScheme DS  -->
```

```
<complexType name="ClassificationSchemeType">
  <complexContent>
    <extension base="MPEG-7:ClassificationSchemeBaseType">
      <sequence>
        <element name="Term" type="MPEG-7:TermDefinitionType" minOccurs="1"
          maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<!-- Definition of ClassificationSchemeDescription Top-level Type -->
<complexType name="ClassificationSchemeDescriptionType">
  <complexContent>
    <extension base="MPEG-7:ContentManagementType">
      <choice>
       <element name="ClassificationScheme" type="MPEG-7:ClassificationSchemeType"
         minOccurs="1" maxOccurs="unbounded"/>
       <element name="ClassificationSchemeBase"
        type="MPEG-7:ClassificationSchemeBaseType" minOccurs="1"
        maxOccurs="unbounded"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

Wie man in der XML Schema Definition der ClassificationScheme erkennt, wird eine ClassificationScheme Instanz durch einen eindeutigen URI identifiziert, und kann beliebig viele Term Elemente beinhalten. Die definierten Terme eines ClassificationScheme kann man in der MPEG-7 Beschreibung auf verschiedene Weisen nutzen: über Verweis auf den Term, oder wie im folgendem Beispiel über einen Verweis mit zusätzlicher Angabe des Terms inline.

```
<TermUse href="urn:examples:cs:SportsCS:soccer">
  <Name xml:lang="en">Soccer</Name>
</TermUse>
```

Außerdem kann man einen Term auch ad-hoc frei in einer Beschreibung definieren, ohne dass er auf ein ClassificationScheme verweist.

Mit GraphicalClassificationScheme werden beispielsweise Klassifizierungen für Graphen möglich, dies kann man beispielsweise für das Speichern von Beschreibungsvorlagen, die im MPEG-7 Dokument instanziert werden können, oder als Mechanismus für Graphregeln nutzen.

```
<!-- Definition of GraphicalClassificationScheme DS  -->
<complexType name="GraphicalClassificationSchemeType">
  <complexContent>
    <extension base="MPEG-7:ClassificationSchemeBaseType">
      <sequence>
        <element name="GraphicalTerm" type="MPEG-7:GraphicalTermDefinitionType"
         minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="strict" type="boolean" use="optional" default="false"/>
    </extension>
  </complexContent>
</complexType>
```

People and Places:
Zu den Basic Tolls der MDS gehören auch DescriptionSchemes zur Beschreibung von Agenten[4] sowie zur Beschreibung von Orten. Mit den, vom AgentType abgeleiteten, DescriptionSchemes für Personen, Personengruppen und Organisationen kann man also reale oder fiktionale Agenten beschreiben.

Mit dem Place DS kann man existierende, historische und fiktive Plätze über Elemente für Name, Rolle, geographische Position, Land, Region und Adresse beschreiben.

Affective Response:
Mit dem Affective DS kann man den Affekt[5] eines Nutzers (Betrachters) in Bezug zu multimedialen Inhalten Beschreiben.

Ordering Descriptions:
Um den große Datenmengen zu organisieren wird diese gern nach bestimmten Schlüsseln sortiert. Mit dem OrderingKeyType DS kann man Hinweise zum Sortieren von Beschreibungen für die Präsentation geben.

```
<!-- Definition of OrderingKey DS -->
<complexType name="OrderingKeyType">
  <complexContent>
    <extension base="MPEG-7:HeaderType">
      <sequence>
        <element name="Selector">
          <complexType>
            <attribute name="xpath" type="MPEG-7:xPathSelectorType"/>
          </complexType>
```

[4] entspricht in MPEG-7 dem abstaktem DS-Typ AgentType, von dem jeweils DS für Personen, Personen-Gruppen und Organisationen abgeleitet sind.
[5] Bezeichnung für ein Gefühl besonderer Stärke beispielsweise Wut, Trauer, Entsetzen. Auslöser kann ein bestimmtes Ereignis sein.

```
        </element>
        <element name="Field" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <attribute name="xpath" type="MPEG-7:xPathFieldType"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="name" type="string" use="optional"/>
      <attribute name="semantics" type="string" use="optional"/>
      <attribute name="direction" use="optional" default="descending">
        <simpleType>
          <restriction base="NMTOKEN">
            <enumeration value="descending"/>
            <enumeration value="ascending"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

Phonetic Description:
Mit dem definiertem PhoneticTranscriptionLexicon DS besteht die Möglichkeit die Aussprache von Wörtern zu beschreiben.

2.5 Content Management

Die Tools, welche unter den Bereich Content Management fallen, erlauben die Beschreibung des Lebenszyklus multimedialer Inhalte. Sie sind untergliedert in Tools für die Beschreibung des Mediums, des Erzeugungsprozesses und der Nutzung.

2.5.1 Media Information

Das MediaInformation DS besteht aus einem optionalem MediaIdentification Descriptor, um ein Medium zu identifizieren und optionale Informationen zur Domäne zu beschreiben, und mindestens einem MediaProfile DS, mit denen man unter anderem die verschiedenen verfügbaren Instanzen eines Mediums lokalisieren kann.[6]

[6] Zur Lokalisierung wird dabei ein Element MediaLocatorType verwendet, der zu den Basic Elements gehört.

2.5.2 Content Creation

Mit den zu Content Creation zugehörigen Tools kann Informationen zum Erzeugungs- bzw. Produktionsprozess eines Mediums beschreiben.

```
<!-- Definition of CreationInformation DS  -->
<complexType name="CreationInformationType">
  <complexContent>
    <extension base="MPEG-7:DSType">
      <sequence>
        <element name="Creation" type="MPEG-7:CreationType"/>
        <element name="Classification" type="MPEG-7:ClassificationType"
         minOccurs="0"/>
        <element name="RelatedMaterial" type="MPEG-7:RelatedMaterialType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Die Beschreibung dieser Informationen wird in das CreationInformation DS gepackt und besteht, wie auch der Auschnitt mit der Typdefinition aus dem zugehörigen XML-Schema zeigt, aus folgenden DS:

- Einem Creation DS, der Informationen zu Ort, Zeit, Aktion, Material und Organisation beinhaltet.
- Einer optionalen Klassifizierung, die genutzt werden kann um nutzerbasierende Klassifikationen wie Sprache, Genre oder Kategorie als auch serviceorientierte Klassifikationen zu beschreiben.
- Einem optionalem RelatedMaterial DS, mit dem man zusätzliche Informationen zu verwandten Inhalten beschreiben kann.

2.5.3 Content Usage

Die Beschreibung der Informationen zur Nutzung des multimedialen Inhalts geschieht über das UsageInformation DS und beinhaltet Elemente für die Beschreibung der Rechte, der Finanzen, der Verfügbarkeit und der Nutzung. Die Informationen werden gegebenenfalls bei einer Nutzung aktualisiert.

2.6 Content Description - Content Structure

Die Content Structure Tools beschreiben die räumliche und zeitliche Struktur sowie die Struktur des Mediums mit Hilfe einer Menge von, in Verbindung stehenden, Segmenten.

2.6.1 Segment Entities

Mit den MPEG-7 Segment Entity Tools kann man Segmente des multimedialen Inhalts Beschreiben. Dazu gehören räumliche Segmente, zeitliche Segmente und Segmente in der Medienquelle. Die Tools sind jeweils abgeleitet vom Segment DS, welches ein abstrakten Datentyp darstellt, der beliebige Teile multimedialer Inhalte repräsentiert.

Mit den Tools in einem Segment DS kann man allgemeine Eigenschaften, wie Erzeugungsinformationen, Nutzungsinformationen (usage), Medieninformationen, Text-Annotationen, Informationen zur Wichtigkeit (MatchingHint D und PointOfView D), beschreiben. Die abgeleiteten DS beschreiben jeweils allgemeine oder spezielle Segmenttypen. Beipspielsweise erweitert das Still Region DS das abstrakte Segment DS um Elemente zur Beschreibung räumlicher, zeitlicher und visueller Aspekte und kann das Element SpatialDecomposition zur Beschreibung räumlicher Dekompositionen beinhalten. In der Abbildung A.2 im Anhang findet man eine graphische Darstellung über den StillRegionType sowie seines Elementes SpatialMask. Im folgenden ein Überblick über die in MPEG-7 unterschiedenen Segmente und Ihren Aufgabenbereich.

Rein visuelle Segment Entity Tools:

- StillRegion, zur Beschreibung räumlicher Segmente in 2-D Bildern oder Video-Frames.
- VideoSegment DS, welches eine Gruppe von Video-Frames repräsentieren kann.
- MovingRegion DS, welches eine Gruppe von Pixeln (eine Region) in einer Gruppe von Video-Frames repräsentiert.

Audio und Audiovisuelle Segment Entity Tools:

- AudioSegment DS repräsentiert ein zeitliches Intervall in einer Audio-Sequenz, also eine Gruppe von Samples.
- AudioVisualSegment DS beschreibt Ton und Bild in audiovisuellen Daten und repräsentiert ein zeitliches Intervall in diesen.

- AudioVisualRegion DS beschreibt auch Ton und Bild in audiovisuellen Daten. Dabei kann man ein willkürliches audiovisuelles Segment beschreiben, welches sich auf ein beliebiges Zeitintervall in den Audiodaten und eine beliebige Region in den Videodaten bezieht. Für die Beschreibung der Region wird im übrigen, genau wie bei dem MovingRegion DS, der SpatioTemporalLocator Descriptor verwendet.

MPEG-7 Segmente spezieller Medien:

- Mosaic (Beschreibung von Panoramas aus verschiedenen Video-Segmenten)
- 3-D
- Image oder Videotext
- Tinte (Beispielsweise zur Beschreibung von elektronischen Whiteboardeinträgen)
- Multimedia (Beschreibt die Komposition aus verschiedenen Inhalten beispielsweise für MPEG-4 oder eine Webseite.)
- Videobearbeitung (Beschreibung verschiedener Ansichten oder Übergänge)

Mit den Segment Entity Tools kann man auch Segmente beschreiben, die nicht direkt verbunden sind sich aber aus separaten Komponenten zusammensetzen. Dafür stehen Mask Descriptoren zur Verfügung.

2.6.2 Segment Eigenschaften

Hier sollen noch einmal die Eigenschaften und Beschreibungsmöglichkeiten von Segmenten in MPEG-7 zusammengefasst werden, wie sie auch in Abbildung 2.4 zu sehen sind.

Da die verschiedenen Segment DS wie erwähnt jeweils von dem abstrakten Segment DS abgeleitet sind, stehen folgende allgemeinen Eigenschaften, welche jeweils optional sind, für Segmente zur Verfügung:

- allgemeinen Medieninformationen (vgl. MediaInformation DS im Kapitel 5.5.1) oder ein MedienLocator D
- Erzeugungsinformationen (vgl. CreationInformation DS Kapitel 5.5.2)
- Nutzungsinformationen (vgl. UsageInformation DS im Kapitel 5.5.3)
- Textannotationen
- Sematische Beschreibung mit Hilfe des Semantic DS
- Matching Informationen zur Beschreibung der relativen Relevanz mit dem MatchingHint D.

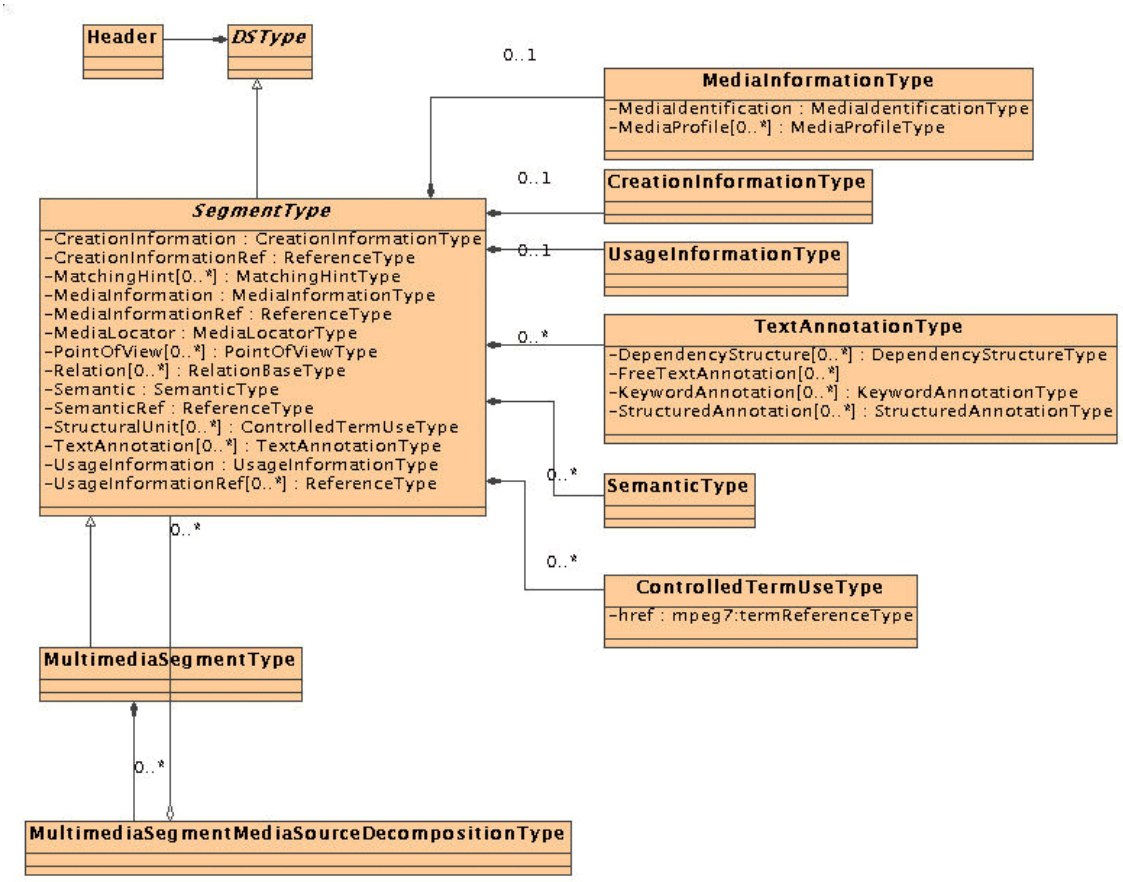

Abbildung 2.4: MultimediaSegment [4]

- Relative Relevanzinformationen in Bezug auf einen bestimmten Gesichtspunkt, beispielsweise Spannung oder Kamera Zoom, mit dem PointOfView D.
- Beziehungen zu anderen Elementen (vgl. Relation DS im siehe Kapitel 5.4.4)

Desweiteren steht mit dem Mask Descriptor für verschiedene Segmente die erwähnte Möglichkeit zur Repräsentation von Segmenten aus verschiedenen verbundenen Komponenten zur Verfügung. Der Mask D ist ein abstrakter D, von dem die D SpatialMask, TemporalMask und SpatioTemporalMask abgeleitet sind. Mit diesen kann man also Segmente beschreiben, die aus - räumlich oder zeitlich separaten - verbundenen Komponenten bestehen. Dabei ist klar, dass andere Beschreibungen im Segment sich auf alle dadurch verbundenen Komponenten beziehen. Möchte man stattdessen eine separate Beschreibung der Komponenten, muss man Sie in Segmente mit Hilfe der Segment Decomposition aufteilen.

Andere Tools von Segmenten beschreiben, die für das Segment relevanten, medienspezi-

fische Eigenschaften. So zum Beispiel Informationen zur Handschrifterkennung.

Schließlich können natürlich, abhängig von der Art des Segments, verschiedene Descriptoren und Description Schemes innerhalb des Segmentes verwendet werden, um etwa visuelle Eigenschaften des Segmentes zu beschreiben.

2.6.3 Segment Decomposition

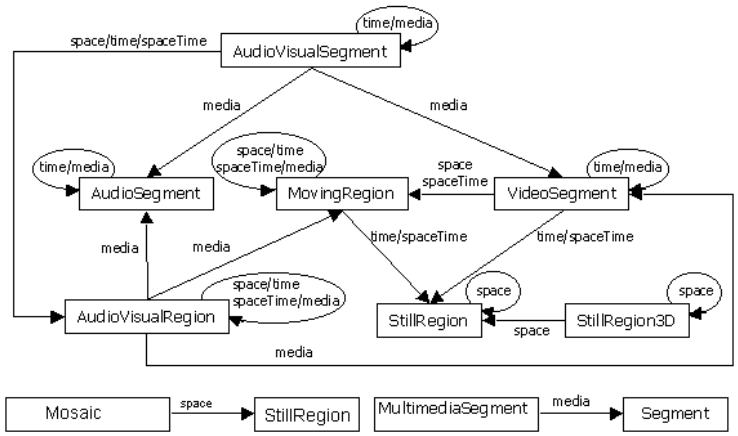

Abbildung 2.5: Gültige Dekompositionen zwischen MPEG-7 Segmenten [1, S. 123]

Mit den Segment Decomposition Tools kan man die Dekomposition oder Aufteilung von Segmenten in Untersegmente beschreiben. Sie Unterstützen die Erzeugung von Hierarchien. Dekompositionen werden durch das SegmentDecomposition DS beschrieben. Dies ist ein abstraktes DS, mit dem man allgemeine Informationen zu einer Dekomposition repräsentieren kann. Dabei schreibt MPEG-7 nicht vor wie eine Aufteilung in Untersegmente geschehen soll, aber es kann die Kriterien und Eigenschaften einer Dekomposition beschreiben.

```
<complexType name="SegmentDecompositionType" abstract="true">
<complexContent>
<extension base="MPEG-7:DSType">
<attribute name="criteria" type="string" use="optional"/>
<attribute name="overlap" type="boolean" use="optional" default="false"/>
<attribute name="gap" type="boolean" use="optional" default="false"/>
</extension>
</complexContent>
</complexType>
```

Folgende Basis Segment Decomposition DS erben direkt ohne irgendwelche Erweiterungen von dem SegmentDecomposition DS und sind wie dieses ebenfalls abstrakt: SpatialSegmentDecomposition DS, TemporalSegmentDecomposition DS, SpatioTemporalSegmentDecomposition DS und MediaSourceSegmentDecomposition DS. Damit haben Instanzen dieser DS die, im SegmentDecomposition DS definierten, optionalen Attribute criteria, gap und overlap. Sie stehen jedoch für die verschiedenen möglichen Dekompositionsarten:

- räumlich (engl. space)
- zeitlich (engl. time)
- räumlich-zeitlich (engl. spacetime)
- Dekompostionen im Medium, wie beispielsweise die Aufteilung nach Bestandteilen wie Audiotracks oder verschiedenen Kameras.

Segmente können nach den verschiedenen Dekompositionsarten aufgeteilt werden, dabei ergibt nicht jede Art für jeden Segmenttyp eine sinnvolle bzw. gültige Aussage. Abbildung 2.5 veranschaulicht die verschiedenen möglichen Aufteilungen von Segmenten in Untersegmente.

Die hierarische Komposition kann auch für das Retrival verwendet werden, denn so ist es beispielsweise mögliche nur bestimmte Bildteile zu finden. Des weiteren ermöglicht es verschiedene Skalierungen der Beschreibung.

Trotzdem kann die hierarische Struktur für bestimmte Applikationen nicht ausreichend genug sein. Aus diesem Grund wurden Strukturelle Relationen (Structural Relations) definiert, um allgemeinere Segmentstrukturen zu definieren. Dazu gehören graph, base und structural relation Tools. Um strukturelle Beziehungen beschreiben zu können werden in MPEG-7 verschiedene alllgeemeine Beziehungen wie „links„ oder „rechts„ mit Hilfe bestimmter Classification Schemes standardisiert. Ein Graph kann beispielsweise im Sematic Descriptor oder Relationship D vorkommen. Zur Veranschaulichung dient folgender kleiner Beispielausschnitt, indem beschrieben wird, dass ein Spieler links vom Torwart steht:

```
<Graph>
<Relation type="urn:mpeg:MPEG-7:cs:TemporalRelationCS:2001:left"
source="#Player_2" target="#GoalKeeper_2"/>
...
</Graph>
```

Es ist klar, dass MPEG-7 nur die Beschreibungsmöglichkeiten definiert - eine Anwendung muss in der Lage sein diese geeignet zu interpretieren. Strukturelle Relationen sind sehr flexibel aber dafür schwieriger für Retrivalzwecke nutzbar.

Erzeugung einer Segmenthierarchie:
Wie erwähnt ist es nicht Bestandteil der MPEG-7 Spezifikation den Segmentierungsprozess zu beschreiben. Er stellt lediglich die Beschreibungsmöglichkeiten bereit. Dennoch soll hier kurz eine mögliche Vorgehensweise bei der hierarischen Segmentierung, die sich insbesondere für StillRegion Segmente eignet, vorgestellt werden. In der Literatur ist dies auch unter dem Problem „partition tree creation“ (Erzeugung eines Partitionsbaums) bekannt und besteht im Wesentlichen aus drei Schritten:[2, S. 46-48]

- Segmentierung
  Ziel des Schrittes ist es eine initiale Segmentierung in einzelne Partitionen, mit eher hohem Detailgrad, vorzunehmen.

- Binären Partitionsbaum zusammensetzen:
  Aus den einzelnen Partitionen werden schrittweise neue Partitionen zusammengesetzt, so das ein binärer Baum entsteht, an dessen Wurzeln die initialen Segmente zu finden sind. Die restlichen Knoten repräsentieren Segmente, die aus ihren Kinder-Segmenten zusammengesetzt sind. In diesem Schritt sollten jeweils zwei, nach einer bestimmten Bewertung (beispielsweise Farbwerte in einem Bild; oder je nach Schritt verschiedene Bewertungskriterien) ähnlichsten Segmente, pro Schritt zusammengefasst werden. In diesem Schritt können auch Algorithmen zur Erkennung von Hinter- und Vordergrundobjekten oder Gesichtern eine Rolle spielen.

- Konstruktion eines beliebigen Baumes
  Ziel des dritten Schrittes ist eine Rekonstruktion des Binärbaums in einen beliebigen Baum, der die Struktur besser repräsentiert. Dies beinhaltet auch die Entfernung von nicht benötigten Knoten. In diesem Schritt ist es beispielsweise auch denkbar Nutzereingriffe zuzulassen.

2.6.4 Weitere MPEG-7 Tools

Weitere MPEG-7 Tools, die jedoch im Rahmen der Diplomarbeit weniger relevant sind, seien der Vollständigkeit wegen kurz erwähnt:

Content Semantic In MPEG-7 werden eine Reihe von Tools definiert um semantische Beschreibungen eines multimedialen Inhalts vornehmen zu können. Dazu gehört die Definition verschiedener Abstraktionsmodelle sowie Tools zur Beschreibung von semantischen Entities[7], Attributen und Relationen.

Navigation und Zugriff Tools die unter diesen Bereich fallen behandeln u.a. die Beschreibung von Zusammenfassungen, Sichten und Variationen. Beispielsweise um das Durchblättern von Inhalten für den Benutzer zu vereinfachen.

Content Organisation Beinhaltet Tools zur Beschreibung von Modellen und Sammlungen.

[7] In diesem Zusammenhang die beschriebenen Realweltobjekte

User Interaction Dazu gehören Tools zur Beschreibung der Nutzungsgeschichte und der Nutzervorzüge. Dies kann man beispielsweise nutzen, um aus bisherigen Nutzeraktionen zu lernen und entsprechend den Präferenzen eines Nutzers zu filtern.

2.7 Visuelle Descriptoren

Ein Hauptziel der visuellen Descriptoren in MPEG-7 ist die standardisierte Beschreibung von visuellen (low-level) Eigenschaften. Diese Low-Level Beschreibungen können genutzt werden, um Bilder oder Videos nur auf der Basis nicht textueller Beschreibungen des Inhalts zu vergleichen zu filtern und zu durchblättern.

Man kann die visuellen Descriptoren in allgemeine visuelle D und anwendungsspezifische (engl. domain-specific) D unterteilen. Zu Ersteren zählen D zur Beschreibung von Farbe, Texture, Kontur (Shape) und Bewegung. Die einzelnen D werden jeweils in eigenen Unterkapiteln näher beschrieben. Zu anwendungsspezifischen D gehört der Face D, zur Unterstützung von Gesichtserkennung.

2.7.1 Color Descriptors

In diesem Kapitel sollen die in MPEG-7 standardisierten Descriptoren zur Beschreibung von Farbeigenschaften näher untersucht werden. Folgende Descriptoren wurden definiert:

- Dominant Color Descriptor erlaubt die Angabe einer kleinen Anzahl dominanter Farbwerte mit ihren statistischen Häufigkeiten, Verteilungen und Varianz. Er bietet damit eine kompakte und intuitive Repräsentation von Farben in einer Region oder einem Bild. Der D nutzt die D ColorSpace und ColorQuantization um den verwendeten Farbraum zu beschreiben. Details zum Dominat Color Descriptor finden sich in Kapitel 3.2.
- Scalable Color Descriptor wird aus einen Farbhistogramm im HSV Farbmodell gewonnen.
- GoFGoPColorType Descriptor erweitert die Nutzung des Scalable Color D auf eine Menge von Frames in einem Video bzw. eine Sammlung von Bildern.
- Color Structure Descriptor basiert auch auf einem Farbhistogramm aber zielt auf die Identifizierung lokalisierter Farbverteilungen indem es ein schmales strukturierendes Fenster verwendet. Bezieht sich auf den HMMD Farbraum.
- Color Layout Descriptor beschreibt das Layout der repräsentierenden Farbe auf einem, über eine Region oder Bild gelegten, Netz. Die Repräsentation basiert auf Koeffizienten der DCT. Es ist ein sehr kompakter und für Such und Filteraufgaben sehr effizienter Descriptor. Weitere Detail im Kapitel 3.3

Außer im Dominant Color Descriptor wurden die verwendeten Farbräume festgelegt, um Abfragen nicht unnötig komplex zu machen.

2.7.2 Texture Descriptoren

Texturen beschreiben das Muster einer Oberfläche durch Eigenschaften wie Homogenität, Struktur oder Beziehung zur Umgebung. Sie stellen eine wichtige Beschreibungsmöglichkeit primitiver Bildeigenschaften dar, da viele Objekte, wie bespielsweise Wasser oder Holzoberflächen, schon anhand ihrer Texture erkannt werden können. Die Texture D in MPEG-7 beziehen sich jeweils auf eine Region und ermöglichen verschiedene Retrivaloperationen. Folgende D, die ebenfalls während des Standardisierungsprozesses eingehend getestet wurden, sind verfügbar:

- Homogeneous Texture Descriptor: Wie der Name schon sagt, ist dieser Descriptor effektiv für die Beschreibung homogener Texturen.
- Texture Browsing Descriptor: Mit diesem Descriptor kann man Texturen durch die Eigenschaften Regelmäßigkeit (regularity), Unebenheit (coarseness) und Ausrichtung (direction) beschreiben. Er ermöglicht also eine wahrnehmungsorientierte Beschreibung und kann für grobe Suchen nach Texturen verwendet werden.
- Edge Histogram Descriptor: Der EHD repräsentiert die räumliche Verteilung von Ecken in einem Bild und stellt damit eine Beschreibung zur Verfügung, welche insbesondere bei unregelmäßigen und unhomogenen Texturen effizient ist. Details siehe 3.4.

2.7.3 Shape Descriptoren

Der Umriss von Objekten stellt eine mächtige Eigenschaft für Retrivaloperationen dar, MPEG-7 stellt eine Reihe von Descriptoren für die Beschreibung von Umrissen zur Verfügung. Oft ist der Umriss eines Objektes mit seiner Funktion bzw. seiner Bedeutung verbunden, so können Menschen charakteristische Objekte allein durch dessen Umriss erkennen. Diese Eigenschaft unterscheidet die Shape D von anderen elementaren visuellen Beschreibungen. Insbesondere sind effektive Beschreibungen von Umrissen für Anwendungen mit Bezug auf Objekterkennung und Objektretrival wichtig. Im MPEG-7 Standard werden drei Descriptoren dafür definiert:

- Region-Based Shape D
- Contour-Based Shape D
- 3-D Shape D

2.7.4 Motion Descriptoren

Mit der Beschreibung von Bewegung in einem Bild hat man einen weiteren wichtigen Anhaltspunkt, den man für Multimedia-Retrival verwenden kann.

2.7.5 Face Descriptor

Die Beschreibung von Gesichtsmerkmalen in Bildern eröffnet ein breites Anwendungsfeld für verschiedene Retrivalaufgaben und es gäbe viele Anwendungen, in denen eine automatische Gesichtserkennung hilfreich wäre. In den letzten Jahren wurden viele Anstrengungen in diesem Bereich unternommen und verschiedene Methoden entwickelt. Eine davon ist Principal Component Analysis (PCA), auf der auch der in MPEG-7 standardisierte Descriptor für Gesichtsbeschreibung aufbaut.

3 Visuelles MPEG-7 Retrieval

Eines der wichtigsten Anwendungsfelder von MPEG-7 ist die Nutzung der vorliegenden Beschreibungen für das Retrieval. Am Beispiel des visuellen Retrievals wird in diesem Kapitel gezeigt, wie man die verschiedenen Descriptoren von MPEG-7 Dokumenten für semantische Abfragen nutzen kann. Es wird im Speziellen auf die Suche von Bildern mit Hilfe bestimmter visueller Merkmale, wie etwa Farbe oder Farbverteilung, eingegangen.

Zunächst soll der allgemeine Ansatz sowie die nötigen Schritte für die Auswertung einer solchen Suchanfrage erläutert werden. Anschließend werden verschiedene visuelle Descriptoren im Detail vorgestellt und mögliche Ähnlichkeitsmaße erläutert.

3.1 Allgemeiner Ansatz

Wie im Information Retrival üblich, kommt das Vektorraummodell zum Tragen. Dabei beschreibt man eine Suche allgemein als eine Anfrage über eine bestimmte Menge von Dokumenten. Die Dokumente haben bestimmte Eigenschaften beziehungsweise Merkmale, die mathematisch als Merkmalsvektor verstanden werden können. Auch die Anfrage selbst wird als Merkmalsvektor betrachtet. Folglich sind die Dokumente relevanter, deren Merkmalsvektor ähnlicher dem anfragenden Vektor ist. Für die Berechnung der Ähnlichkeit zweier Merkmalsvektoren kommt ein Ähnlichkeitsmaß[1] zum Einsatz.[9, Seite 334]

Um relevante Ergebnisse zu einer Anfrage zu erhalten, wird ein Abstandsmaß benötigt, welches möglichst genau dem Ähnlichkeitsempfinden des Menschen entspricht.

So ist es auch bei der Suche nach MPEG-7 Dokumenten. Wie im vorangegangenen Kapitel beschrieben, kann eine MPEG-7 Beschreibung zahlreiche unterschiedliche Descriptoren enthalten, die ein bestimmtes Merkmal beschreiben. Also besteht eine Aufgabe darin, ein möglichst gutes Abstandsmaß zwischen Anfrage und den MPEG-7 Dokumenten zu berechnen.

Während des Standardisierungsverfahrens hat das MPEG-7 Konsortium jeweils schon Vorschläge für Ähnlichkeitsmaße von bestimmten Descriptoren ermittelt. Die Qualität der Ähnlichkeitsmaße wurde mit Hilfe eines definierten Verfahrens gemessen. Dabei wurden die Retrivalergebnisse gegen eine Menge erwarteter Ergebnisse verglichen und ein Qualitätsmaß ANMRR[2] berechnet.

[1] In der Literatur auch Distanzmaß
[2] Average Normalized Modified Retrieval Rate.

Grundsätzlich haben die vorangegangenen Kapitel die Beschreibungsvielfalt von MPEG-7 gezeigt. Ziel einer Suche ist, die zu einer Suchanfrage am besten passenden Dokumente zu erhalten. Welche Beschreibungselemente und Descriptoren eines MPGE7 Dokumentes dabei für eine Suchanfrage relevant sind, ist natürlich abhängig von der Art der Suchanfrage. Demnach kann man den Ablauf für das Retrieval grob in folgende Teilschritte zerlegen:

1. Interpretation und Vorbereitung der Anfrage: Je nach Ziel bzw. Intention der Anfrage sind verschiedene Descriptoren geeignet die Anfrage zu beantworten. Es müssen also in einem ersten Schritt die relevanten Descriptoren bestimmt werden, welche für die Beantwortung der Anfrage geeignet sind. Dafür gibt es verschiedene Ansätze: entweder nutzt man das Wissen um die kodierte Eigenschaft des multimedialen Dokumentes in den verschiedenen Descriptoren, oder man ermittelt die relevanten Descriptoren durch statistische Evaluierungen. Für das Beispiel des visuellen Retrievals sind dies hauptsächlich die verschiedenen visuellen Descriptoren. Demnach ist das Ziel unserer Bildsuche, möglichst Bilder zu finden, deren visuelle Descriptoren möglichst ähnlich zu der Suchanfrage sind.

2. Berechnung des Abstands und Durchführen der Anfrage: Sind die Descriptoren, welche für die Anfrage relevant sind ermittelt, so kann man dies nutzen um das Abstandsmaß für die Anfrage zu ermitteln. Um die Ähnlichkeitsmaße für Descriptoren in der Berechnung des Abstandsmaßes zu nutzen, wird jeweils der anfragende Descriptor in der Suchanfrage ermittelt. Um schließlich den Abstand zwischen einer Anfrage und einem MPEG-7 Dokument zu ermitteln, kann man durch geeignete Kombination der jeweiligen Ähnlichkeitsmaße zwischen anfragendem Descriptor und dem Descriptor des MPEG-7 Dokumentes ein Abstandsmaß berechnen. Für das Retrievals ist es also allgemein wichtig, geeignete Ähnlichkeitsmaße für die unterschiedlichen Descriptoren zu finden und zu nutzen. Wie so etwas im Detail aussehen kann wird im Kapitel 5 anhand einer Beispielapplikation erläutert.

3.1.1 Evaluierung visueller Deskriptoren

Während des Standardisierungsprozesses von MPEG-7 wurden Experimente durchgeführt um verschiedene Deskriptoren und Algorithmen bewerten zu können. Für visuelle D kam dabei der Ansatz, vom Standpunkt einer Retrivalapplikation zu bewerten, zum Einsatz. Die Qualität des Abfrageergebnisse bewertete dabei die Ausdrucksstärke eines Descriptors. Für die Messungen wurde eine Datenmenge, eine Abfragemenge[3] und die, zu den jeweiligen Abfragen korrespondierenden ground-truth Daten definiert. Unter ground-truth Daten versteht man die erwartete Menge ähnlicher Bilder zur jeweiligen Abfrage. Um auf Basis der spezifizierten Abfragen und ground-truth Daten ein möglichst objektives Maß zu definieren werden folgende Faktoren beachtet:

- Normalisierung bezüglich der Anzahl von Daten in den ground-truth Mengen.

[3]Common Color Queries (CCQ) genannt.

- Bessere Bewertung für Algorithmen, welche die ground-truth Daten weiter oben im Suchergebnis enthalten.
- Strafe für jedes nicht gefundene ground-truth Element. Auch Treffer nach einem bestimmtem Limit sollen als nicht gefunden gewertet werden.

Schließlich definiert man zu diesem Zweck das Maß ANMRR[4] , ein Wert zwischen null (alle ground-truth Daten gefunden) und eins (keine ground-truth Daten gefunden), an welchem man die Qualität des Retrievals über alle Abfragen ablesen kann.

3.2 DominantColor D

3.2.1 Aufbau

Wie erwähnt, beschreibt man mit diesem D Farbeigenschaften auf Grund einer kleinen Anzahl dominanter Farbwerte. Er eignet sich für inhaltsbasierte Retrievals nach Farbe - entweder eines ganzen Bildes oder auch nur einer beliebigen Region (rechteckig oder irregulär).

Allgemeiner Aufbau:
Ein DominantColor D kann, wie bereits erwähnt, die optionalen Elemente *ColorSpace* und *ColorQuantization* beinhalten. Außerdem enthält er ein Element *SpatialCoherency* und ein bis acht Elemente *Value*. Wobei es sich gezeigt hat, dass meistens drei bis fünf Farbwerte für eine gute Charakterisierung ausreichen.

Das Element *SpatialCoherency* repräsentiert die gewichtete Summe der räumlichen Kohärenz der einzelnen dominanten Farbwerte.

Die *Value* Elemente beinhalten jeweils ein Element *Percentage* (zur Angabe des prozentualen Anteils) ein Element *Index* (zur Angabe der Farbe) und ein optionales Element *ColorVariance* (zur Angabe der Streuung). Das Element *Percentage* spezifiziert den Anteil der Pixel die dem Farbwert entsprechen. Der Wert wird liniear auf 5 Bit quantisiert, hat also einen Wertebereich von 0 bis 31. Das Element *Index* spezifiziert die Farbe im definierten Farbraum als Liste von Zahlen (jeweils die Dimensionen bzw. Komponenten im Farbraum). Die Anzahl an Bits für jede Komponente wird durch das Element *ColorQuantization* bestimmt.

Der Standard-Farbraum ist RGB. Die Quantisierung muss nicht unbedingt mit jedem Descriptor angegeben werden, sondern kann auch für die Datenbank definiert sein.

Der DominantColor D im MPEG-7 Dokument:
Prinzipiell kann ein DominantColor D in jedem Element mit dem abstrakten Typ Visual

[4] Average Normalized Modified Retrieval Rate.

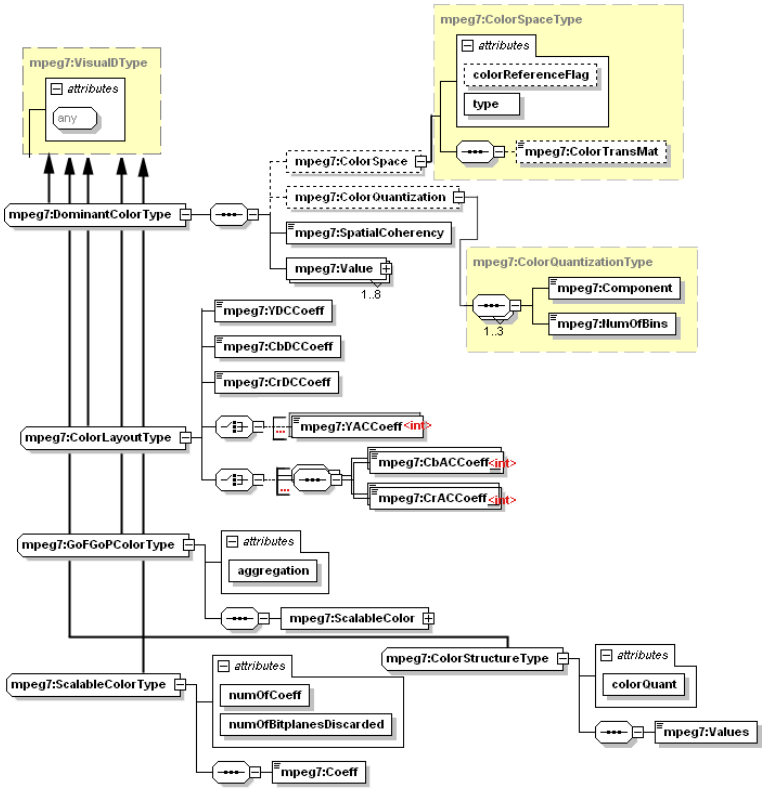

Abbildung 3.1: Überblick des Aufbaus verschiedener visueller Deskriptoren

D (Im XML-Schema der komplexe Typ VisualDType) instanziert werden. So kann beispielsweise ein StillRegion Segment einen Visual D im Element VisualDescriptor beinhalten. Wobei auch ein StillRegion Segment an vielen verschiedenen Stellen in einem MPEG-7 Dokument instanziert werden kann. Ein Beispiel findet man in folgendem Ausschnitt aus einer MPEG-7 Beschreibung:

```
....
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="ImageType">
    ....
      <Image>
          <VisualDescriptor xsi:type="DominantColorType">
          <SpatialCoherency>0</SpatialCoherency>
          <Value>
            <Percentage>0</Percentage>
            <Index>21 2 14</Index>
          </Value>
```

```
        ....
      </VisualDescriptor>
....
```

Hier sieht man eine typische Instanzierung des DominantColor D in einer MPEG-7 Beschreibung. Das Element Image ist ein StillRegion DS, in dem das Element VisualDescriptor eine Instanz eines beliebigen Visual D repräsentieren kann - in diesem Fall einen DominantColor D. Die visuelle low-level Beschreibung eines StillRegion Segmentes kann aber auch erst innerhalb einer Dekomposition zu finden sein.

3.2.2 Vorgeschlagenes Ähnlichkeitsmaß

Auch wenn die Berechnung von Ähnlichkeitsmaßen und die Verwendung der Deskriptoren nicht zum MPEG-7 Standard gehört, gibt es natürlich Vorschläge und Empfehlungen dafür. Im Folgenden sollen Ähnlichkeitsmaße, die für das Retrival verwendet werden können, eingeführt werden:

Definieren wir einen DCD formal wie folgt:

$$F = \{(c_i, p_i, v_i), s\}(i = 1, 2, ..., N) \tag{3.1}$$

Wobei c_i ein Vektor des korrespondierenden Farbraums, p_i der zwischen 0 und 1 normalisierte prozentuale Anteil, v_i die Farbvarianz und N die Anzahl dominanter Farben angibt. Wenn man die Varianz und die räumliche Koheränz ignoriert, kann man die Unähnlichkeit D zweier DominantColor D F_1 und F_2 wie folgt berechnen:

$$D^2(F_1, F_2) = \sum_{1}^{N_1} p_{1i}^2 + \sum_{1}^{N_2} p_{2i}^2 - \sum_{1}^{N_1}\sum_{1}^{N_2} 2a_{1i,2j}p_{1i}p_{2j} \tag{3.2}$$

Wobei $a_{k,l}$ der Gleichheitskoeffizent zweier Farben c_k und c_l ist.

$$a_{k,l} = \begin{cases} 1 - d_{k,l}/d_{max} & d_{k,l} \leq T_d \\ 0 & d_{k,l} > T_d \end{cases} \tag{3.3}$$

Wobei $d_{k,l}$ die euklidische Distanz der beiden Farbvektoren, T_d der maximale Abstand zweier als ähnlich zu betrachtenden Farben und $d_{max} = \alpha T_d$ ist. Empfohlenen Werte für T_d liegen bei 30 im RGB Farbraum und für α zwischen 1,0 und 1,5.

Neben dem angegebenem Ähnlichkeitsmaß D werden auch noch die Maße unter Berücksichtigung der Streuung D_S und der Farbvarianz D_V definiert.

$$D_S = w_1 abs(s_1, s_2) + w_2 D \tag{3.4}$$

Wobei w_1 und w_2 festgelegte Werte zur Gewichtung sind, die Empfehlung liegt bei 0,3 und 0,7. Das Maß D_V zu berechnen ist am kostenintensivsten und soll hier nicht näher

betrachtet werden. In den Experimenten zur Evaluierung während der Standardisierung hat sich gezeigt, dass die Verwendung von D_S mit fünf Farbwerten einen ANMRR Wert von 0,21 erreicht und einen akzeptablen Kompromiss zwischen Performance und Retrivalrate darstellt.

3.2.3 Probleme des Ähnlichkeitsmaßes

Im Rahmen der Untersuchungen in der Diplomarbeit zeigten sich bestimmte Probleme bei der Verwendung des vorgeschlagenen Ähnlichkeitsmaßes. Zum einen wird in der Berechnung der Doppelsumme lediglich die Ähnlichkeit der Farben, nicht aber die Ähnlichkeit der prozentualen Anteile beachtet, was zu unerwarteten Ergebnissen führen kann.

Um hier eine zusätzliche Gewichtung zu erhalten wurde folgendes Ähnlichkeitsmaß definiert:

$$D_P^2(F_1, F_1) = \sum_1^{N_1} p_{1i}^2 + \sum_1^{N_2} p_{2i}^2 - \sum_1^{N_1} \sum_1^{N_2} 2a_{1i,2j} p_{1i} p_{2j} (1 - (abs(p_{1i} - p_{2j})/\beta) \qquad (3.5)$$

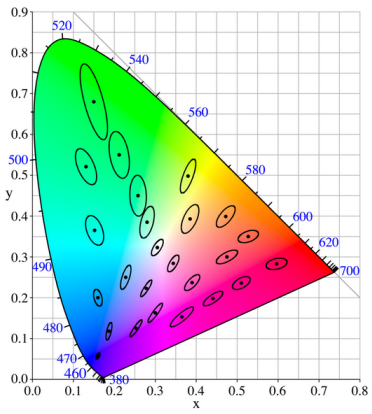

Abbildung 3.2: die MacAdams Ellipsen zeigen Bereiche der empfindungsgemäß gleichen Farben.

Zum anderen ist die Ähnlichkeitsberechnung zweier Farben basierend auf dem euklidischem Abstand im RGB Farbraum ungeeignet. Für den Abstand zwischen den Farben ist für optimale Ergebnisse der Farbraum relevant. Da die meisten Farbräume eigentlich nicht für die Beurteilung menschlicher Farbähnlichkeitswahrnehmungen geeignet sind, weil der geometrische Farbabstand nicht mit dem empfindungsgemäßem Abstand korrespondiert, empfiehlt sich für Abstandsberechnungen die Farbräume CIE-LUV[5] oder CIE-Lab zu verwenden. Diese sind darauf ausgelegt, dass Abstände zweier Farben gut

[5] Von Commission Internationale de l'Éclairage standardisierter Farbraum

mit der menschlichen Farbähnlichkeitswahrnehmung korrespondieren. Die Umrechnung der Farben aus anderen Farbräumen in den CIE-LUV Farbraum gehen aber zu Lasten der Performance. Falls man bei einem Retrieval die Vorteile eines solchen Farbraumes nutzen möchte, empfiehlt sich gegebenenfalls eine Implementierung über eine Umrechnungstabelle.

3.2.4 Definition eines scorebasierten Ähnlichkeitsmaßes

Während der Tests im Rahmen der Diplomarbeit kam auch noch ein weiteres abgeändertes Maß zum Einsatz. Dabei war das Ziel, einen möglichst einfachen Score zu berechnen, bei dem für jede Übereinstimmung Punkte vergeben werden. Für zwei zu vergleichende Farbwerte eines Dominant Color Descriptors soll der erreichbare Punktwert von dem prozentualen Anteil der Farben abhängen.

$$D_P^2(F_1, F_1) = \sum_{1}^{N_1} \sum_{1}^{N_2} a_{1i,2j} min(p_{1i}, p_{2j}) \beta \tag{3.6}$$

3.3 Color Layout Descriptor

Der Color Layout Descriptor (CLD) ist eine sehr kompakte und auflösungsunabhängige Repräsentation der Farbeigenschaften eines Bildes und geeignet für schnelle Abfragen.

Berechnung:

- Der ColorLayout D nutzt den YCbCr Farbraum, auf den sich auch die Berechnung des Descriptors bezieht. Dieser Farbraum kennt einen Luminanz-Koeffizient (Y) und zwei Chrominanz-Koeffizienten (Cb und Cr). Wobei der Luminanz-Koeffizient für die menschliche Wahrnehmung wichtiger ist als die beiden Chromianz-Koeffizienten.
- Schritt 1: Unterteilung des Bildes in 8x8 (64) Blöcke um die Auflösungsunabhängigkeit zu gewährleisten.
- Schritt 2: Ermittlung einer repräsentativen Farbe pro Block. Ergebnis ist also eine 8x8 Pixel Repräsentation des Bildes
- Schritt 3: Im Farbraum YCbCr wird für jede Farbkomponente eine 8x8 DCT[6] durchgeführt. Man erhält also drei mal 64 Koeffizienten.

[6] Diskrete Cosinus Tranformation

- Schritt 4: Diese werden Zick-Zack gescannt und die ersten n Koeffizienten werden nichtlinear[7] quantifiziert.

Der Descriptor kann dabei eine verschiedene Anzahl von Koeffizienten darstellen. Unterstützt werden $3, 6, 10, 15, 21, 28$ und 64 Koeffizienten für die Luminanz- bzw. Chrominanz-Komponenten, empfohlen werden insgesamt 12 (6 für Luminanz und je 3 für Chrominanz) oder 18 (jeweils 6 pro Farbkomponente). In der XML Repräsentation wird jeweils der erste Koeffizient (DC Wert) in einem separatem Element mit 6 Bit codiert dargestellt, und die folgenden Koeffizienten (AC Werte) mit 5 Bit codiert jeweils als Liste in einem weiteren Element. Eine Beispielinstanz des Descriptors sieht demnach wie folgt aus:

```
<VisualDescriptor xsi:type="ColorLayoutType">
  <YDCCoeff>15</YDCCoeff>
  <CbDCCoeff>21</CbDCCoeff>
  <CrDCCoeff>43</CrDCCoeff>
  <YACCoeff5>28 16 16 16 19</YACCoeff5>
  <CbACCoeff2>0 16</CbACCoeff2>
  <CrACCoeff2>1 16</CrACCoeff2>
</VisualDescriptor>
```

Bei den ANMRR Test erhielt man je nach Koeffizientenanzahl Werte im Bereich von 0,2 bis 0,4.

3.3.1 Ähnlichkeitsmaße

Folgendes Ähnlichkeitsmaß wurde vorgeschlagen: [1, Seite 210]

$$D = \sqrt[2]{\sum_i w_{yi}(DY_i - DY'_i)^2} + \sqrt[2]{\sum_i w_{bi}(DCb_i - DCb'_i)^2} + \sqrt[2]{\sum_i w_{ri}(DCr_i - DCr'_i)^2} \tag{3.7}$$

3.4 Edge Histogram Descriptor

Die räumliche Verteilung von Ecken in einem Dokument ist ein hilfreiches Merkmal für das Retrieval. Der Edge Histogram Descriptor gehört zu Texture beschreibenden Descriptoren und repräsentiert die Verteilung von Ecken in einem Bild.[3]

Dazu wird das Bild in 16 Unterbilder mit einem 4x4 Raster unterteilt. Zu jedem Unterbild werden 5 Werte gespeichert, die jeweils die Stärke eines bestimmten Ecktypen im betreffenden Unterbild bestimmen. Dabei werden folgende Ecktypen unterschieden:

[7] Die zu verwendenden Quantifizierungsfunktionen sowie die Zick-Zack Scanreihenfolge sind in Part3 des MPEG-7 Standards angegeben

- vertikale Ecke des Unterbildes
- horizontale Ecke des Unterbildes
- 45-Grad Ecke des Unterbildes
- 135-Grad Ecke des Unterbildes
- Nicht lineare Ecke des Unterbildes

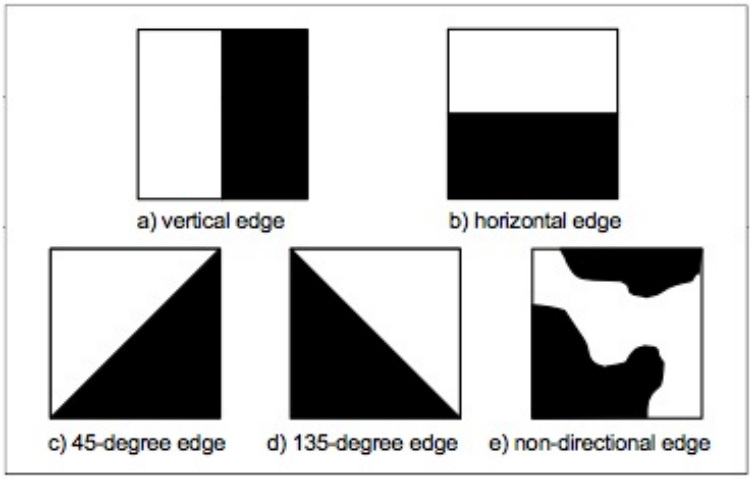

Abbildung 3.3: Veranschaulichung der 5 verschiedenen Ecktypen des Edge Histogram Descriptors [3]

So benötigt der Edge Histogram Descriptor also insgesamt 16 x 5 = 80 Werte.

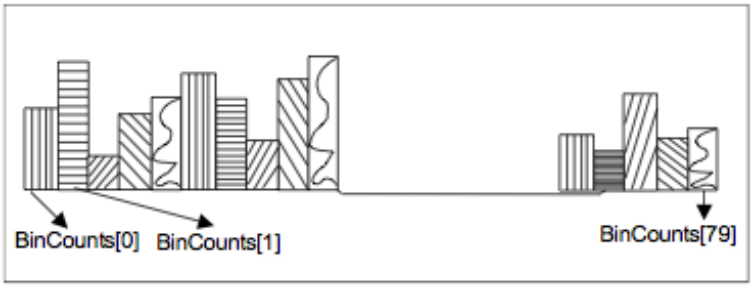

Abbildung 3.4: Veranschaulichung des Edge Histogramms mit seinen 80 Werten [3]

Eine Beispielinstanz eines Edge Historgam Descriptors sieht wie folgt aus:

```
<VisualDescriptor xsi:type="EdgeHistogramType">
```

```
        <BinCounts>1 2 5 6 ... 7 2 3 4 5 7</BinCounts>
</VisualDescriptor>
```

3.4.1 Ähnlichkeitsmaß

Für die Berechnung des Ähnlichkeitsmaßes empfiehlt es sich nicht nur die Eckenverteilung der 16 einzelnen Unterbilder zu betrachten, sondern auch die Eckenverteilung der möglichen Kombinationen der Unterbilder (auch SemiGlobal Unterbilder, vgl. Abbildung 3.5) genannt, sowie die Eckenverteilung im Gesamtbild (Global) selbst. Demnach ist ein definiertes Ähnlichkeitsmaß zweier Edge Histogram Descriptors A und B wie folgt definiert:[3]

$$\begin{aligned} D(A,B) = \sum_{0}^{79} |Local_A(i) - Local_B(i)| + 5 \sum_{0}^{4} |Global_A(i) - Global_B(i)| \\ + \sum_{0}^{64} |SemiGlobal_A(i) - SemiGlobal_B(i)| \end{aligned} \tag{3.8}$$

Es müssen also insgesamt 150 Werte in die Berechnung des Ähnlichkeitsmaßes einfließen. Die Werte für die Histogramme der SemiGlobal Unterbilder sowie des Gesamtbildes lassen sich einfach aus den jeweiligen Durchschnitten der gespeicherten Werte der Unterbilder berechnen.

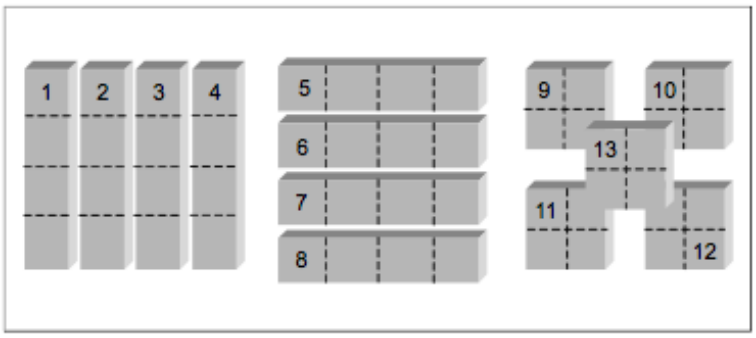

Abbildung 3.5: Veranschaulichung der 12 relevanten SemiGlobal Unterbilder [3]

3.5 Überlegungen zu möglichen Anwendungen und Problemen

Allgemein überwiegt zurzeit das Textretrieval in der Praxis und die meisten Anwender sind vertraut mit textbasierter Suche.

Mit MPEG-7 wird nun semantische Suche möglich, wie die Suche nach konkreten Bildeigenschaften. In der Praxis ist dies zurzeit oft nur in Nischenanwendungen relevant: Oft finden derartige Suchen ihren Anwendungsfall bei der Klassifizierung von Bildern, wie beispielsweise die Klassifizierung von medizinischen Bildern oder Pflanzen. Die Relevanz für diese Nischenanwendungen kommt daher, dass man möglichst viel und möglichst konkrete Dinge über eine konkrete Anwendungsdomäne weiß und somit die Anwendung darauf optimieren kann.

Für eine allgemeine Fotosuche etwa, ohne eingegrenzte Domäne und der großen Heterogenität der Daten und Anwendungsziele, ist es noch schwierig akzeptable Ergebnisse zu erzielen. Darum scheint es sinnvoll diese Art von Suchen zum Beispiel nur als Zusatz zu einer textbasierten Suche einzusetzen.

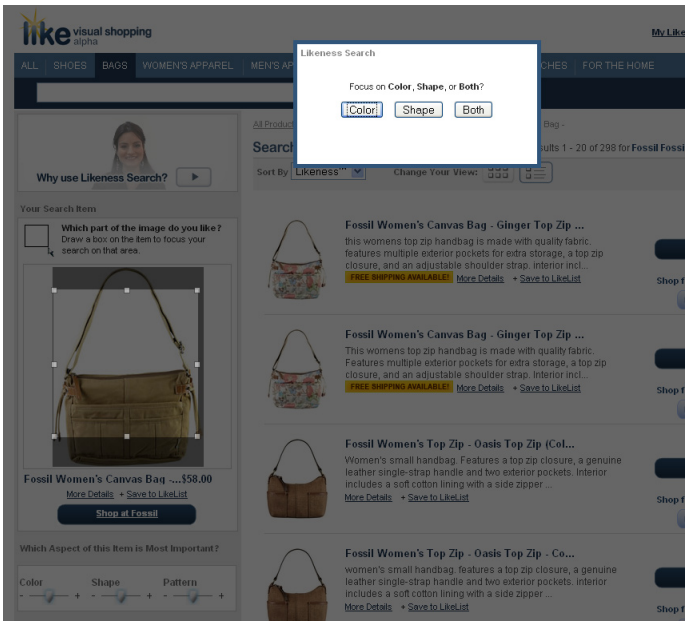

Abbildung 3.6: Screenshot der visuellen Suche nach ähnlichen Produkten bei like.com.

Wenn wir bei der Domäne einer allgemeinen semantischen Bildsuche bleiben, so gibt es zahlreiche verschiedene Ansprüche und Erwartungen, die ein Nutzer an eine Suche stellt. So gibt es verschiedene Möglichkeiten eine Suche zu definieren, und je nach Ziel des Benutzers spielen ganz verschiedene Eigenschaften eine Rolle, wie die folgende kurze Auflistung schon zeigt:

- „Query by Example“ : Der Nutzer möchte Bilder finden, die ähnlich zu einem vorgegebenem Bild (bzw. Video) sind. Je nach Ziel des Benutzers können ganz unterschiedliche Aspekte eine Rolle spielen:

  - Ähnlichkeit nach Farbe: Unterschiedliche Relevanz der verschiedenen Farbeigenschaften (Verteilung, Gesamteindruck, exakte Übereinstimmung bestimmter Farben, Helligkeit usw.)
  - Ähnlichkeit nach Motiven auf dem Bild: Ist die Form, Farbe oder Struktur der Objekte relevant? Ist die Anordnung der Motive wichtig? Die Objekte müssen ggf. auf dem Bild erkannt und gewichtet werden. Spielt die Drehung der Objekte eine Rolle?
  - Ähnlichkeit nach Semantik: Ist der Ort, die Personen oder die dargestellte Aktion relevant?

- „Query by Sketch“ , d.h. das Finden von Inhalten durch das Zeichnen von Motiven und Objekten.
- Suchen nach konkreten Objektbeschreibungen, wie beispielsweise die Angabe von Texturen.

3.5.1 Teilprobleme einer MPEG-7 basierten Suche

Wie erläutert, haben also verschiedene potentielle Nutzer mit ihrer Suchanfrage verschiedene Ziele. Die Kunst liegt darin, die richtige Kombination und Relevanzbewertung in einer Abfrage umzusetzen. Im Folgenden sollen die verschiedenen Aspekte und Schwierigkeiten, die bei einer Suche auf Grundlage von MPEG-7 Dokumenten relevant sein können, aufgezeigt werden.

Eine wichtige Herausforderung liegt in der komplexen Struktur einer MPEG-7 Beschreibung: Unterschiedliche relevante Eigenschaften findet man an verschiedenen Stellen in einer MPEG-7 Instanz. Ein Beispiel dafür stellen die verschiedenen DS dar, welche als komplexe Datentypen im XML-Schema definiert sind und jeweils in verschiedenen Elementen genutzt werden.

Die mit Hilfe eines Descriptors bzw. Description Schemes beschriebenen Eigenschaften haben, je nach dem in welchem Zusammenhang sie in einer Beschreibung verwendet werden, verschiedene Relevanzen für eine Abfrage. Verkomplizierend kommt dazu, dass die Beschreibungen entweder konkrete Instanzen eines D bzw. DS beinhalten können oder alternativ lediglich eine Referenz, die gegebenenfalls natürlich aufgelöst werden muss.

Zusammenfassend kann man folgende Punkte nennen, die bei einer allgemeinen Suche über MPEG-7 Dokumente wichtig sind:

- Auswahl der richtigen Deskriptoren für die Auswertung einer gegebenen Suchanfrage. Die Descriptoren können wie gezeigt an unterschiedlichen Stellen einer Beschreibung auftreten.
- Nutzen des geeignetsten Abstandsmaßes für die jeweiligen Descriptoren.

- Gewichtung von Alternativen und unterschiedlichen Ergebnissen, sowie sinnvolles Kombinieren und Gewichten einzelner relevanter Beschreibungsmerkmale.
- Unterstützung möglichst vieler verschiedener MPEG-7 Instanzen, sowie robustes Umgehen mit eventuell nicht vorhandenen Beschreibungselementen. Gut wären Auswertungen, die möglichst viele in der jeweiligen Beschreibung verfügbare Informationen auswerten. Diese sollten auch gleichzeitig stabil gegenüber verschieden vollständigen bzw. verschieden aufgebauten MPEG-7 Beschreibungen sein.
- Nicht nur konkrete Instanzen eines Descriptors, sondern auch die Struktur und Position eines Beschreibungselementes ist relevant. Abfragen müssen also nicht nur den Inhalt eines Beschreibungselementes auswerten, sondern auch die Struktur (Ort des Beschreibungselementes in einer MPEG-7 Instanz) und die Beziehung zwischen Beschreibungselementen. Die Struktur kann beispielsweise die Relevanz eines Beschreibungselementes verändern, aber auch in der Struktur selbst sind für die Abfrage relevante Informationen kodiert, beispielsweise in einem Dekompositions-Element.
- Optimierte Laufzeit und Performance bei einer Anfrage.
- Die Ergebnisse einer Anfrage sollten sich an den erwarteten Ergebnissen orientieren. Dabei ist es schwierig, die konkreten Erwartungen eines Nutzers an das Suchergebnis zu kennen. Hier ist es beispielsweise denkbar, Lernalgorithmen zusammen mit Relevanzfeedback vom Nutzer einzusetzen.

In konkreten Anwendungen kann man vereinfachend eine konkrete Untermenge von akzeptierten beziehungsweise erwarteten MPEG-7 Beschreibungen definieren. Dies ist bei vielen Anwendungen, wie beispielsweise „Caliph und Emir“ [8], der Fall. Auch die im Kapitel 5 beschriebene Beispielapplikation zur Bildsuche unterstützt zunächst nur einige, von den in einer MPGE-7 Beschreibung vorkommenden Descriptoren.

[8] Caliph & Emir sind javabasierte Applikationen um Bilder mit MPEG-7 zu beschreiben, sowie Retrivaloperationen auszuführen. vgl. B.1.1

4 XML und Datenbanken

Wenn wir die in dieser Arbeit zu untersuchende Thematik näher betrachten, so ist es klar, dass es erforderlich ist mit großen Mengen von XML-Dokumenten zu arbeiten. Aus diesem Grund möchte ich einen Überblick über mögliche Lösungen und Datenbanken für XML geben. Dabei gibt es einerseits zahlreiche verschiedenste Bemühungen relationale Datenbanken mit XML zu verbinden. Und andererseits existieren so genannte native XML Datenbanken, die als reine Datenbanken für XML Dokumente verstanden werden können.

4.1 XQuery

XQuery ist eine Abfragesprache, die speziell für Abfragen auf XML Daten entwickelt wurde. Aus diesem Grund sei diese hier kurz vorgestellt.

XQuery ist als Resultat eines vom W3C im Dezember 1998 organisierten Workshops „QL´98“ in Boston entstanden. Aus diesem Workshop ging eine Arbeitsgruppe für XML Abfragen hervor, mit dem initialen Ziel ein XML Datenmodel und eine XML Abfragesprache unter Beachtung existierender Standards zu spezifizieren. Großen Einfluss auf das Design von XQuery hatten die Standards XPath, XML Schema sowie die Abfragesprache Quilt[1] .

Eine der ersten Ergebnisse war die Veröffentlichung der Anforderungen (Requirements for an XML-Query Language) gefolgt von den XQuery Use-Cases. Letztlich definierte die Arbeitsgruppe eine Abfragesprache mit zwei verschiedenen Syntaxen. Eine für die Lesbarkeit optimierte, schlüsselwortbasierende Syntax: XQuery, sowie eine für Maschinengenerierung optimierte Syntax: XQueryX. Die entstandene Sprache XQuery ist nicht nur eine Abfragesprache sondern eignet sich auch für allgemeine Bearbeitung von XML Dokumenten. Es wurde auch früh entschieden die existierende Sprache „XPath“ als Untermenge von XQuery aufzunehmen.

XQuery ermöglicht im Gegensatz zu SQL die Spezifikation von Abfragen mit komplexeren semantischen Definitionen.

[1] Quilt wird als der direkte Vorgänger von XQuery bezeichnet. Es ist genau wie XQuery eine funktionale Sprache zum Abfragen von XML Dokumenten mit verschiedenen kombinierbaren Ausdrücken.

4.1.1 FLWOR Ausdrücke

Ein FLWOR[2] Ausdruck ist einer der mächtigsten Ausdrücke in XQuery und vergleichbar mit den SELECT-FROM-WHERE Anweisungen bei SQL. Ein FLWOR Ausdruck bindet Variablen mit `for` und `let` Ausdrücken und nutzt diese Bindung um neue Ergebnisse zu erzeugen.

Ein `for` Ausdruck bindet Variablen jeweils an ein Item aus einer Sequenz. Ein `let` Ausdruck bindet eine Variable als Ganzes an das Ergebnis eines Ausdrucks. Folgendes Beispiel verdeutlicht dies:

```
for $i in (1,2,3)
let $j := (1,2,3)
return
  <tupel><i>{ $i }</i><j>{ $j }</j></tupel>
```

Führt zur Ausgabe von:

```
<tupel><i>1</i><j> 1 2 3 </j></tupel>
<tupel><i>2</i><j> 1 2 3 </j></tupel>
<tupel><i>3</i><j> 1 2 3 </j></tupel>
```

Mit der where Anweisung kann man Tupel eliminieren, die einer bestimmten Bedingung nicht genügen. Dabei sind alle Ausdrücke erlaubt, die einen Wahrheitswert zurückgeben. Im Gegensatz zu SQL kann man also auch über mehrere Tupel Bedingungen abprüfen, wie folgendes Beispiel zeigt:

```
declare namespace mpeg7="urn:mpeg:mpeg7:schema:2001" ;
for $descriptions in collection("output")/mpeg7:Mpeg7//mpeg7:Description
let $videosegments:=$descriptions//mpeg7:VideoSegment
where count($videosegment/mpeg7:PointOfView) > 2
return
 <result>
   <gefunden>{ count($videosegments)}</gefunden>
   <VideoSegmente>{ $videosegments }</VideoSegmente>
 </result>
```

Falls man sich für alle Elemente *VideoSegment* unterhalb irgendeines Description Elementes mit mindestens zwei Kindelementen *PointOfView* interessiert, liefert das Beispiel das entsprechende Resultat. Man kann auch erkennen, dass die gebundene Variable `\$videosegments` in der `let` Anweisung mehrfach verwendet wird. Dies stellt eine Möglichkeit dar Ausführungszeit zu sparen.

[2] Ausgesprochen „flower“ . Die Buchstaben stehen für die fünf Bedingungen die in einem FLWR Ausdruck vorkommen können: `for, let, where, order by` und `return` .

4.2 XML und RDBMS

Im Folgenden soll ein kurzer Überblick über die allgemeinen Ansätze, relationale Datenbank Management Systeme mit XML zu verbinden, gegeben werden. Anfang 1999 gab es in der Datenbankforschung zahlreiche Publikationen mit dem Ziel, RDBMS für das Speichern oder Abfragen von XML Daten zu verwenden.[6] In den Ansätzen gibt es große Unterschiede, was die Unterstützung der Abfragesprachen, die Konstrukte zum Mappen zwischen XML-Schema und relationalem Schema oder die Aufgaben der Middleware[3] anbelangt. Viele Anbieter relationaler Datenbanken haben bereits verschiedenartige Unterstützung für XML in Ihren Produkten eingebaut beziehungsweise angekündigt. Die Unterstützung reicht dabei vom einfachen Zur-Verfügung-Stellen von XML Sichten bis hin zu eingebauten XQuery oder SQL/XML Implementationen.[6]

Ich beziehe mich im Folgenden insbesondere auf Zusammenfassungen der Publikationen des Forums „First International XML Database Symposium, XSym 2003“ , in der die in Abbildung 4.2 ersichtliche Taxonomie vorgeschlagen wurde.[6, Seite3]

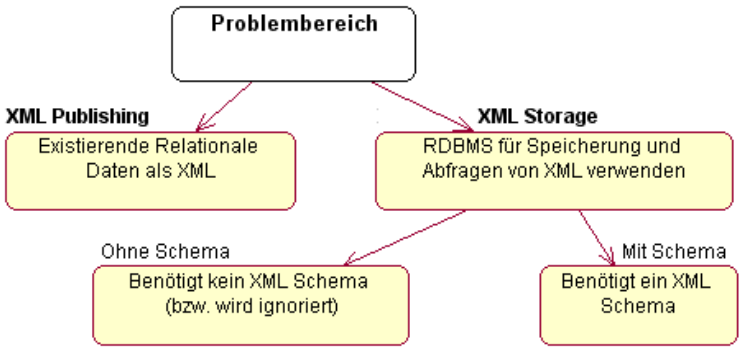

Abbildung 4.1: Mögliche Klassifikation des Problembereiches XML und RDBMS

4.2.1 XML Publishing

Beim XML Publishing geht es darum, existierende relationale Daten als XML zur Verfügung zu stellen. Teilprobleme dabei sind:

- **XML View (XML Sicht) der Daten definieren**
  Wird in der Regel durch eine „View Definition Language“ des DBMS definiert.

[3] In diesem Zusammenhang eine Software zwischen dem RDBMS und der XML Anwendung, die beispielsweise Mapping und Anfrageübersetzungen bearbeitet.

  - In XPeranto[4] zum Beispiel durch eine Baumansicht auf die relationalen Daten. [13]
  - In Oracle XML DB oder Mircosoft SQL Server 2000 SQLXML wird ein annotiertes XSD XML Schema verwendet um die XML Ansicht zu definieren.

- **Materialisieren des XML Views**
  Hier gibt es den Ansatz, einen einzelnen OUTER UNION Join[5] Select, wie bei XPeranto, an die Datenbank zu stellen. Oder es werden von einer Middleware[6] mehrere Selects an die DB geschickt. In Anbetracht, dass SQL nur unzureichende Rekursionsunterstützung bietet, scheint es mit der zweiten Variante einfacher zu sein, Unterstützung für rekursive Views zu bieten.

- **Abfragen auf den XML Daten durchzuführen**

4.2.2 XML Speicherung

Relationale Datenbank Managementsysteme haben den Vorteil, dass sie weit anerkannt und erprobt sind. Aus diesem Grund ist es verständlich, dass die meisten Anbieter von RDBMS eifrig versuchen einen Weg zu finden, XML in relationalen Datenbanken zu speichern. Im allgemeinen gibt es zwei prinzipielle Ansätze dafür: flaches Speichern und zerstückeltes Speichern. Beim flachen Speichern wird ein XML Dokument in eine feste Anzahl von Tabellen gespeichert. Beim zerstückelten Speichern wird das XML Dokument in zahlreiche Tabellen, Zeilen und Spalten normalisiert.[20]

4.2.2.1 XML Speicherung ohne Schema

Das Problem XML Daten ohne ein Schema mit Hilfe eines RDBMS speichern zu wollen, zielt hauptsächlich darauf ab, ein relationales Schema zu finden, mit dem es möglich ist XML Dokumente unabhängig von der Anwesenheit eines Schemas zu speichern. Die Teilprobleme in diesem Bereich sind:

- Design des relationalen Schemas, also welches allgemeine relationale Schema soll für XML Daten benutzt werden.

[4] Eine Middleware die den Ansatz verfolgt eine XML Sicht auf relationale Daten zu bieten. Abkürzung für: Xml Publishing of Entities, Relationships, And Typed Objects

[5] Outer Union vereinigt nicht Union-kompatible Relationen, indem es die nicht gemeinsamen Attribute ergänzt und sie mit Null-Werten auffüllt. [23]

[6] Middleware bezeichnet in der Informatik anwendungsunabhängige Technologien, die Dienstleistungen zur Vermittlung zwischen Anwendungen anbieten, so dass die Komplexität der zugrundeliegenden Applikationen und Infrastruktur verborgen wird. *nach: W. Ruh u. a.: Enterprise Application Integration. Wiley, 2001.*

- Anfrageübersetzungsalgorithmen zu definieren, also wie man gegebene XML Anfragen in SQL Anfragen übersetzt.

4.2.2.2 Design des relationalen Schemas

Hier seien kurz drei Ansätze für mögliche relationale Schemata vorgestellt, in denen man beliebige XML Dokumente speichern kann.

Kantenansatz:
Entsprechend dem Kantenansatz [8] wird ein XML Dokument als Graph betrachtet und jede Kante des Graphen wird als Tupel in einer einzigen Relation gespeichert. Für die Kantentabelle reichen die fünf Spalten *quelle,ziel,name,ordnung und wert* . Jedes Tupel der Relation repräsentiert also eine Kante und damit ein Kindelement oder ein Attribut. Wie aus den Spaltenbezeichnungen ersichtlich, werden zu einer Kante die Ids der Quell- und Zielobjekte (Elemente oder Attribute), der Name der Kante, die Geschwister Position (sibling position) und, falls es sich um einen einfachen Typ handelt, der Wert des Zielobjektes, gespeichert.

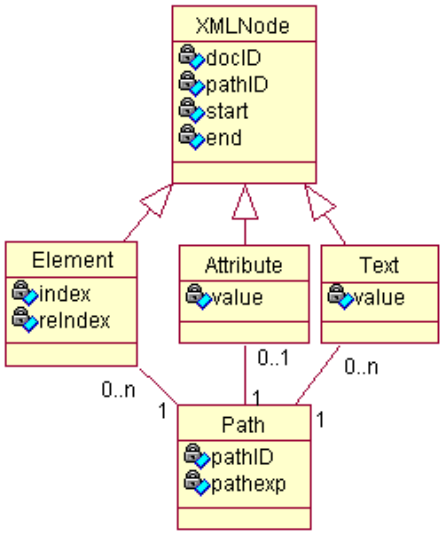

Abbildung 4.2: UML Diagramm des XRel Ansatztes zur Speicherung von XML Daten

XRel:
Als weiteres Beispiel soll hier der XRel-Ansatz [7] kurz vorgestellt werden. Dieser verfolgt die Idee zu jedem Element die Pfad-Id, die mit dem entsprechendem Wurzel-Zu-Knoten Pfad korrespondiert, und das Interval, welches von dem Element abgedeckt wird, zu speichern. Das heißt, es werden alle möglichen Pfade von der Wurzel zu jedem Knoten im XML Baum aufgezählt und als Ausdruck an sich in einem relationalen Attribut gespeichert. Das Intervall wird in XRel einfach als Nummernpaar repräsentiert, welches

für die Anfangs- und Endposition im XML-Dokument steht. Ein dafür vorgeschlagenes relationales Schema kann aus dem UML Diagramm in Abbildung 3.2 entnommen werden. Es besteht aus den vier Entitys: Element, Attribute, Text und Path und einer optionalen Entität Document. In der Entity Path wird der entsprechende Pfad von der Wurzel bis zu einem Knoten direkt als Zeichenkette gespeichert, also beispielsweise „./Mpeg7./Description./MultimediaContent./AudioVisual“ .

In zusätzlichen Überlegungen wird versucht, ordnungsbasierte Anfragen durch das Speichern eines zusätzlichen Ordnungsfeldes effizienter zu unterstützen.

STORED:
In STORED wird noch ein anderer - nicht gewöhnlicher - Ansatz verfolgt: Auf Basis eines teilweise strukturierten relationalen Schemas wird mit Hilfe von Data-Mining Techniken automatisch eine Abbildung generiert. Ein XML Dokument wird dann mit Hilfe der generierten Abbildung in das relationale Schema gespeichert. Dieser Abbildung nicht entsprechende Elemente werden in einem Überlaufgraphen gespeichert.

Andere Strategien sind die Erweiterung der relationalen Maschine um neue Datentypen und/oder neue Algorithmen. In den bekannten Datenbanksystemen Oracle XML DB und IBM DB2 XML Extender wird das Speichern von XML-Dokumenten ohne Schema beispielsweise durch den Datentyp CLOB unterstützt. X-Query Anfragen über diesen Datentyp auszuwerten, fällt dann in den Bereich einer echten XML Datenbank.

4.2.2.3 Anfrageübersetzung

Das Teilproblem der Anfrageübersetzung befasst sich mit der Aufgabe, eine gegebene XQuery Anfrage in eine beziehungsweise mehrere entsprechende SQL Anweisungen zu übersetzen. Die jeweiligen Ansätze dafür sind also direkt an das, zur Speicherung der XML Daten verwendete, relationale Schema gebunden. Ziel ist, neben einer möglichst breiten Unterstützung der XQuery Ausdrücke, eine effektive Anfragebearbeitung zu erreichen.

Im vorherigen Kapitel wurde der Ansatz XRel vorgestellt, dieser kann besonders bei Abfragen mit einfachen XPath Ausdrücken punkten, da eine eigene Relation für die Pfade vorhanden ist. Für Anfrageübersetzungen wurde bei XRel [7] ein Kern von XPath, hier XPathCore genannt, identifiziert und ein detaillierter Algorithmus, um solche Anfragen in SQL zu übersetzen, bereitgestellt. Da zu jedem Element eine Pfad-Id des korrespondierenden Wurzel-Zu-Blatt Pfades gespeichert ist, kann ein simpler XPath Ausdruck wie „buch/kapitel/titel“ einfach ausgewertet werden. Anstatt einen Join für jeden Pfadschritt auswerten zu müssen, reicht es, alle Elemente mit übereinstimmender Pfad-Id zu extrahieren. Folgende XPath-Ausdrücke kann man beispielsweise mit diesem Ansatz effektiv übersetzen:

XPath Ausdruck	XRel Ansatz
/Mpeg7/Description/MultimediaContent	Einfacher Pfadausdruck. Nutzt einfachen Stringvergleich.
//Description//AudioVisual	// kann durch Wildcard Abfragen mit % abgebildet werden.
/Mpeg7/Description[2]	Prädikatabfragen werden durch die Attribute index und reIndex in der Relation „Elements“ effektiv unterstützt.

Es gibt auch detaillierte Überlegungen komplette FLWR Ausdrücke zu übersetzen. Dabei werden verschiedene neue Operatoren vorgeschlagen, um die generierten SQL Anfragen effizient im RDBMS ausführen zu können. Diese sind hochspezialisiert und ähnlich zu Operatoren, welche in echten XML Datenbanken zum Einsatz kommen.

4.2.2.4 Zusammenfassung und offene Probleme

Vorangehend wurden verschiedene Ansätze vorgestellt, die jeweils zum Ziel haben, relationale Datenbanken für die Speicherung und Abfrage von allgemeinen XML Daten zu verwenden.

Eine mögliche Kategorisierung der dabei in der Forschung und Industrie verfolgten Ansätze ist beispielsweise die Folgende:

1. **Id-basiert:** Jedes Element erhält einen eindeutigen Schlüssel und die Baumstruktur des Documents wird durch Fremdschlüssel zum Elternelement gespeichert.

2. **Intervallbasiert:** Jedes Element wird mit einer Region assoziiert, welche den Teilbaum unter diesem repräsentiert.

3. **Pfadbasiert:** Als Ergänzung zum Id-basierten oder intervallbasierten Ansatz, wird jedes Element zusätzlich mit einer Pfad-Id assoziiert, welche den entsprechenden „Wurzel zu Blatt“ Pfad repräsentiert.

Trotzdem bleiben viele Probleme, insbesondere was die Breite der unterstützten Abfragen und die Performance betrifft. Deshalb lohnt es sich, einen Blick auf die so genannten nativen XML Datenbanken zu werfen.

4.3 Native XML Datenbanken

Wie man im vorangehenden Kapitel lesen konnte, gilt es verschiedene Probleme bei der Nutzung von RDBMS für das Speichern und Abfragen von XML-Dokumenten zu lösen. Dies ist insbesondere auf den Fakt zurückzuführen, dass XML Daten - im Gegensatz zu relationalen Daten - semistrukturiert sind.

Es liegt also nahe, XML Daten in einer echten XML Datenbank zu speichern. Der Begriff native XML Datenbank erlangte Bekanntheit in der Marketingkampagne für Tamino[7] ohne eine formale Definition. Dennoch wird der Begriff native XML Datenbank allgemein für Datenbanken verwendet, welche ein logisches Modell für XML definieren und ein XML Dokument als grundlegende Speichereinheit haben. Hier soll unter einer nativen XML Datenbank also eine Datenbank, welche explizit für das Abspeichern und Abfragen von XML Dokumenten implementiert wurde, verstanden werden. Wichtige und nützliche Eigenschaften sind:[19]

- Die Terminologie Collection wird für eine Menge gespeicherter XML Dokumente verwendet.
- Ein oder mehrere XML Abfragesprachen wie XQuery oder XPath werden unterstützt.
- Verschiedene Strategien zum Verändern und Löschen, wie beispielsweise die Implementation von XUpdate[8].
- Unterstützung von Transaktionen, Sperren und Nebenläufigkeit. Wünschenswert dabei ist die Möglichkeit zum Sperren auf Knotenebene.
- Anbieten einer allgemeinen API. Vorkommende Standards sind XML: DB API[9] und XQuery API for Java (XQJ) [10]
- Round-Tripping: Darunter versteht man die Eigenschaft, das selbe Dokument, welches man gespeichert hat, auch wieder zu erhalten.
- Indexierung, als Möglichkeit die Abfrageperformance zu steigern.
- Normalisierung, referenzielle Integrität und Skalierbarkeit.

Wie wir wissen, ist das Herz einer Datenbank das eigentliche Speichern, Indexieren und Abfragen von Informationen. Wie dies im Detail implementiert wird, ist also von enormer

[7] eine native XML Datenbank der Software AG
[8] Standard für das modifizieren von XML Daten in XML Dokumenten. http://xmldb-org.sourceforge.net/xupdate/
[9] Application Programming Interface for XML Databases http://xmldb-org.sourceforge.net/xapi/
[10] Java Specification Requests (JSR) 225: Allgemeine API für XQuery anfragen http://jcp.org/en/jsr/detail?id=225

Wichtigkeit für die Performance und Skalierbarkeit einer Datenbank. In einer XML Datenbank wird ein logisches Modell für das Speichern und Abfragen von XML Dokumenten herausgestellt, welches nicht notwendigerweise äquivalent zum Dokument sein muss. Dabei ist die Indexierung der Daten eines der wichtigsten Eigenschaften und maßgeblich für die Performance bestimmter Abfragen. So findet man zum Beispiel in der Berkeley XML DB Indextypen für die Dokumentenstruktur sowie Indizes für Werte bestimmter Elemente. [17]

Im Rahmen dieser Arbeit wurde die Open Source XML Datenbank eXist verwendet, um XQuery Abfragen über eine Collection mit MPEG7 XML Dokumenten auszuführen.[15] Details dazu sind in Kapitel 5.5 zu finden.

5 Grundlagen der Beispielapplikation Imagerr

Aktuell werden monatlich 100 Milliarden Suchanfragen weltweit [1] getätigt. Allein die Suchmaschine Google hat nach eigenen Angaben etwa 8 Milliarden Webseiten und 1 Milliarde Bilder[2] indexiert. Diese Zahlen verdeutlichen, was mit textbasierten Suchmaschinen zurzeit möglich ist. Doch gerade bei der Suche nach Bildern und anderen multimedialen Daten stellt textbasierte Suche nicht das optimale Werkzeug dar.

Im Rahmen der vorliegenden Diplomarbeit wurde eine Beispielapplikation entwickelt, mit der man auf Basis von visuellen Eigenschaften wie Farbe oder Farbverteilung nach multimedialen Dokumenten suchen kann. Weiterhin erlaubt die Applikation durch ein generisches Modell die Verknüpfung von verschiedenen Suchmethoden.

Die Beispielapplikation bietet folgende Features:

- Webbasiertes Userinterface zur granularen Definition einer semantischen Suchanfrage. Dabei wird auch „Query by Image“ unterstützt.
- Unterstützung verschiedener Indexmethoden, so dass ein Vergleich zwischen SQL Index und XML DB Index erfolgen kann.
- Unterstützung von verschiedenen Ähnlichkeitsmaßen.
- Umsetzung eines generellen Suchansatzes, bei welchem flexibel verschiedene Indexe mit verschiedenen Strategien durchsucht werden können.
- Architektur nach Domain Driven Design[3] und MVC Pattern[4] auf Basis des PHP5 Frameworks FLOW3[5]. Die Implementation der Applikation erfolgte als FLOW3 Paket.
- Optimierte Auswertung von Suchanfragen nach dominanten Farben durch Implementation eines auf den Dominant Color Descriptor basierenden Clustering.

[1] Nach einer Erhebung von comScore (Vgl. Meldung unter http://www.computerwoche.de/netzwerke/web/1904659/ vom 3.9.2009)
[2] Nach einer Newsmeldung von Google. Vgl. http://www.google.de/intl/de/corporate/
[3] Ein Ansatz zum Entwurf eines Klassenmodells einer objektorientierten Applikation. Vgl. 5.2
[4] =Model-View-Controller; ein Quasistandard für die Implementierung moderner Applikationen, bei dem die Funktionalität einer Applikation prinzipiell in 3 seperate Bereiche aufgeteilt ist.
[5] Neues PHP Applikationsframework. Details unter 5.2

5.1 Vorstellen der Applikation

Die Beispielapplikation wird webbasiert bedient und stellt dem Nutzer ein User-Interface zur Verfügung, mit dessen Hilfe er seine Bildsuche definieren kann. Eine Bildsuche besteht dabei aus verschiedenen Suchfiltern, die flexibel definiert und kombiniert werden können. Darunter befindet sich auch die bekannte Textsuche, aber darüber hinaus beliebige weitere Filter. Die verschiedenen Filter einer Suche können vom Nutzer in Ihrer Wertigkeit für die Suche noch gewichtet werden. Insgesamt stehen für eine Suche folgende Möglichkeiten zur Verfügung:

- Einfache Textsuche ohne weitere Filter - zum Beispiel eine Suche nach „rote Sonne“
- Hochladen eines eigenen Bildes, auf Basis dessen eine Suche mit bestimmten Filtern automatisch vorkonfiguriert wird.
- Durch Ergänzen, Löschen und Modifizieren von Suchfiltern kann der Nutzer seine Anfrage verfeinern. Wie in der Abbildung 5.1 zu erkennen, können die Suchfilter einfach mit den Buttons [+] und [-] ergänzt bzw. gelöscht werden. Jeder Filter verfügt selbst über spezifische einstellbare Parameter. Ein Suchfilter zur Bestimmung der dominanten Farbe in einem Bild erlaubt beispielsweise das Auswählen der Farbe und der erwarteten Häufigkeit im Bild.

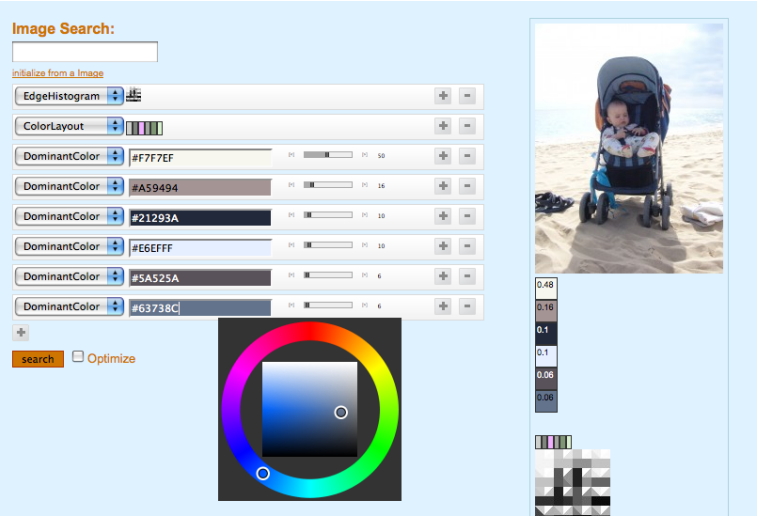

Abbildung 5.1: Screenshot der Suchdefinition in der Beispielapplikation

Nachdem eine Suche ausgelöst wurde, wird diese von der Applikation ausgewertet. Die Funktionsweise und Schrittfolge in der Auswertung wird später näher erläutert. Der Nutzer erhält schließlich als Resultat seiner Anfrage eine Liste von gefundenen Bildern, sortiert nach der berechneten Relevanz.

5.2 FLOW3

Wie fast alles in der IT Welt entwickeln sich auch Denkweisen und Vorgehensmodelle bei der Entwicklung von Software schnell weiter. In der Softwareentwicklung trifft man immer häufiger auf agile Methoden[6] und damit verbundene Ansätze wie Test Driven Development, Domain Driven Design, Refactoring[7] oder XP[8].

FLOW3 ist der Name eines neuen Open Source PHP Frameworks, welches derzeit in Entwicklung ist. Es bringt neue Ansätze in die PHP Welt und unterstützt die Ideen des agilen Paradigmas. FLOW3 wird die Basis des bekannten Enterprise Content Management System TYPO3 in der Version 5 und ist aktuell noch in der Entwicklung. Im Folgenden werden wichtige Features von FLOW3 kurz vorgestellt.

Domain Driven Design (DDD):
Ist ein Ansatz, der sich damit befasst, wie man die Kernlogik einer Anwendung modelliert. Diese Kernlogik, als das Herzstück der Software wird dabei Domain Model genannt. Die Grundidee dabei ist, dass das Design der Software direkt die Domain - also den Problem- bzw. Einsatzbereich wiederspiegelt. Identifiziert man in der Domain beispielsweise ein Objekt „Kunde“ , dann gibt es dieses auch im Code der Software. Diese Domainobjekte sollten möglichst frei von systemnaher Logik sein und die eigentliche Business Logik implementieren.

Domain Driven Design wurde dabei hauptsächlich von Eric Evans mit seinem gleichnamigem Buch „Domain Driven Design“ eingeführt. Dort werden allgemeine Ansätze und Pattern beschrieben um ein wartbares Domain Model zu erhalten.

FLOW3 selbst wird nach diesem Ansatz entwickelt und soll bei der Entwicklung eigener DDD basierter Anwendungen möglichst gut unterstützen. Dies geschieht beispielsweise durch eine vollkommen transparente Persitenzschicht für Objekte und die Unterstützung von DDD Bausteinen wie Entities, Values und Repositories durch das Framework selbst.

Aspect Oriented Programming (AOP):
Das Framework unterstützt als erstes PHP Framework in dieser Art AOP. AOP ermöglicht es, zusätzliches Verhalten vor oder nach bestimmten Methoden hinzuzufügen, wobei die Funktionalitäten der Methoden in verschiedenen Klassen liegen. Dies ist besonders hilfreich um sogenannte querschnittliche Belange (engl. crosscutting concerns) einer Applikation separat implementieren zu können. Damit ist es beispielsweise möglich, Funktionalitäten wie Logging oder Sicherheitschecks außerhalb des Domain Layers als Aspekt zu implementieren. Damit kann der Kern der Applikation beim Wesentlichen bleiben und sich auf die Buissinesslogik konzentrieren.

[6] Grundsätze agiler Methiden sind im agilem Manifest festgehalten. Siehe http://agilemanifesto.org/principles.html

[7] Disziplinierte Technik existierenden Code zu restrukturieren ohne dabei das äußere Verhalten zu ändern.

[8] extreme programming ist ein Ansatz in der Softwareentwicklung. Siehe http://www.xprogramming.com/xpmag/whatisxp.htm

Dependency Injection:
Dependency Injection ist ein Entwurfsmuster und dient in einem objektorientierten System dazu, die Abhängigkeiten zwischen Komponenten oder Objekten zu minimieren. Zwischen den Klassen in einer Applikation bestehen verschiedene Abhängigkeiten. So benötigt eine bestimmte Klasse andere Klassen zur Erfüllung ihrer Aufgaben. Während der Instanzierung einer Klasse in FLOW3 werden diese Abhängigkeiten sozusagen von außen gesetzt (injected), so dass die Klasse selbst sich nicht darum kümmern muss woher Sie Instanzen der benötigen Klassen bekommt. FLOW3 unterstützt dabei Constructor Injection, also das Auflösen von Abhängigkeiten, welche im Constructor der Klasse benötigt werden, und spezielle Injection Methoden.

Weiterhin enthält FLOW3 natürlich ein MVC Framework, einen Paketmanager um neue Pakete zu verwalten und zahlreiche Basisklassen, wie beispielsweise für Caching, Validierung, Sicherheit und Resourcenmanagement. Aus diesen Gründen fiel die Wahl eines Frameworks für die Beispielapplikation auf FLOW3.

5.3 Aufbau und Domain Model

Im Wesentlichen unterstützt die Applikation Indexierung und Retrieval multimedialer Dokumente. Diese zwei Bereiche sollen im Folgenden Indexer und Searcher genannt werden. Im Kern der Applikation können sich verschiedene Indexe befinden, deren wichtige Aufgabe darin besteht, Multimedia-Dokumente (hier Objekte der Klasse MMDoc) zu indexieren, sowie Suchanfragen mit passenden Ergebnisdokumenten zu beantworten.

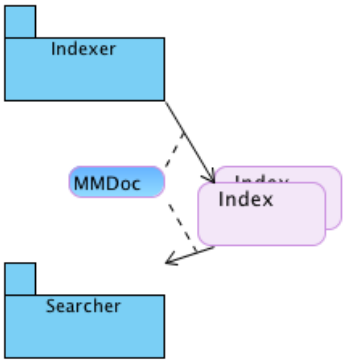

Abbildung 5.2: High Level Aufbau der Beispielapplikation

Man kann die Applikation, wie auch in Abbildung 5.2 zu sehen, in drei Bereiche untergliedern, die in den folgenden Kapiteln näher beschrieben werden:

- „Index“ Klassen bilden den Kern für das Indexieren und Auffinden von Dokumenten.

- Klassen, die für das Indexieren von Dokumenten benötigt werden.
- Sowie das Paket „Searcher“, in dem sich Klassen zur Auswertung einer Suche befinden.

5.3.1 Indexieren von Bildern

Das Indexieren von multimedialen Dokumenten - hier im Speziellen Bilder - erfolgt in einem getrenntem Prozess. Ein solches Multimedia Dokument wird durch eine Instanz der Klasse MMDoc repräsentiert, wobei die Basis dieses Objektes die MPEG-7 Beschreibung des Multimedia Dokumentes ist. Für die Indexierung könnten also einfach vorhandene MPEG-7 Beschreibungen verwendet werden.

Da die MPEG-7 Beschreibung für die Bilder im Anwendungsbereich der Beispielapplikation meist nicht vorliegen, wird auch das Indexieren von Bildern an sich unterstützt. Dabei wird eine Liste von Bildern eingelesen, wobei die Quelle zum Beispiel ein Ordner im Dateisystem oder ein RSS Feed sein kann. Jedes Bild wird analysiert und eine Instanz der Klasse MMDoc erzeugt.

Dieser Schritt erfolgt in der Klasse MMDocFactory. Die Aufgabe dieser Factory[9] ist es, MMDoc Objekte aus rohen Bilddaten zu erzeugen. Dafür wird ein MPEG-7 Dokument des Bildes generiert, welches alle relevanten Descriptoren enthält. Zur Generierung der MPEG-7 Beschreibung zu einem Bild, werden Klassen der Open Source Software Caliph und Emir[10] verwendet, die die benötigten low-level Descriptoren für das gegebene Bild generieren.

Dieses MPEG-7 Dokument stellt die Informationsbasis für das MMDoc Objekt dar. Ein MMDoc Objekt in dieser Applikation stellt also die Repräsentation eines MPEG-7 Dokumentes dar, und sein Interface erlaubt es, die benötigten Descriptoren und Eigenschaften abzufragen.

Die erzeugten MMDoc Objekte werden im Indexierungsvorgang an die Indexobjekte übergeben, welche diese Objekte speichern und indexieren müssen.

5.3.2 Searcher Paket

In diesem Kapitel soll die Funktionsweise einer Suche beschrieben werden. Dabei war es das Ziel, ein möglichst allgemeines Modell einer Suche umzusetzen, welches neben einer flexiblen Definition einer Suchanfrage auch erlaubt in verschiedenen Indexen zu suchen und die Ergebnisse zu kombinieren.

[9] Das Factory Pattern kommt im Domain Driven Design zum Einsatz um komplexe Objekte zu erzeugen
[10] Siehe B.1.1

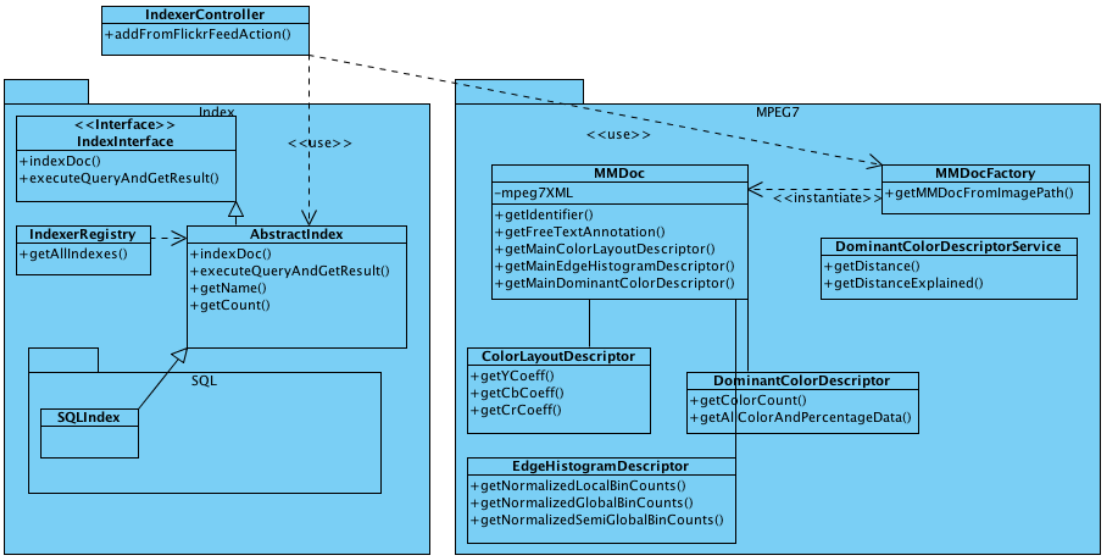

Abbildung 5.3: Überblick der Klassen für die Indexierung von Dokumenten.

Gerade in Anbetracht dessen, dass textbasierte Suchen eine hohe Akzeptanz beim Nutzer haben und es im Bereich der Verwendung und Optimierung textbasierten Retrievals einen großen Erfahrungsvorsprung gibt, liegt es beispielsweise nahe, vorhandene textbasierte Suchmaschinen zu verwenden und die Ergebnisse mit Ergebnissen aus einer semantischen Suche zu kombinieren.

Dazu soll folgende allgemeine Sicht auf eine Suche zugrunde gelegt werden, welche Basis für die im Folgenden vorgestellte Architektur ist: Eine Suche besteht aus einer Suchanfrage (*Query*), welche nach verschiedenen Suchstrategien (*Searchstrategie*) an verschiedene Indexe weitergereicht wird. Jeder Index gibt eine Menge (*Resultset*) von Treffern (*Hit*) zurück, welche mit einer bestimmten Strategie wieder kombiniert werden können. Der Nutzer bekommt letztendlich wieder eine Menge von Treffern zurück.

Demnach ergibt sich das Model aus Abbildung 5.4. Das Searchstrategie Objekt ist verantwortlich, die für die Suche relevanten Indexe zu instanzieren und das Query-Objekt an die Index Objekte weiterzureichen sowie die Resultsets mit Hilfe eines bestimmten ResultsetMerger Objektes zu kombinieren. Ein Searchstrategie Objekt kann dabei das Query für die verschiedenen Index Objekte vorher noch modifizieren. Über verschiedene Suchstrategien ist es also zum Beispiel möglich, die Suchanfrage parallel oder sequentiell an mehrere verschiedene Indexe zur Auswertung zu übergeben und die zurückgelieferten Ergebnisse zu kombinieren. Es ist Aufgabe des Controllers das Query Objekt an Hand der vom Nutzer spezifizierten Suche zu initialisieren, sowie eine Suchstrategie für die Auswertung der Suche zu wählen.

Eine Suchanfrage wird über das Query Objekt abgebildet und besteht aus verschiedenen

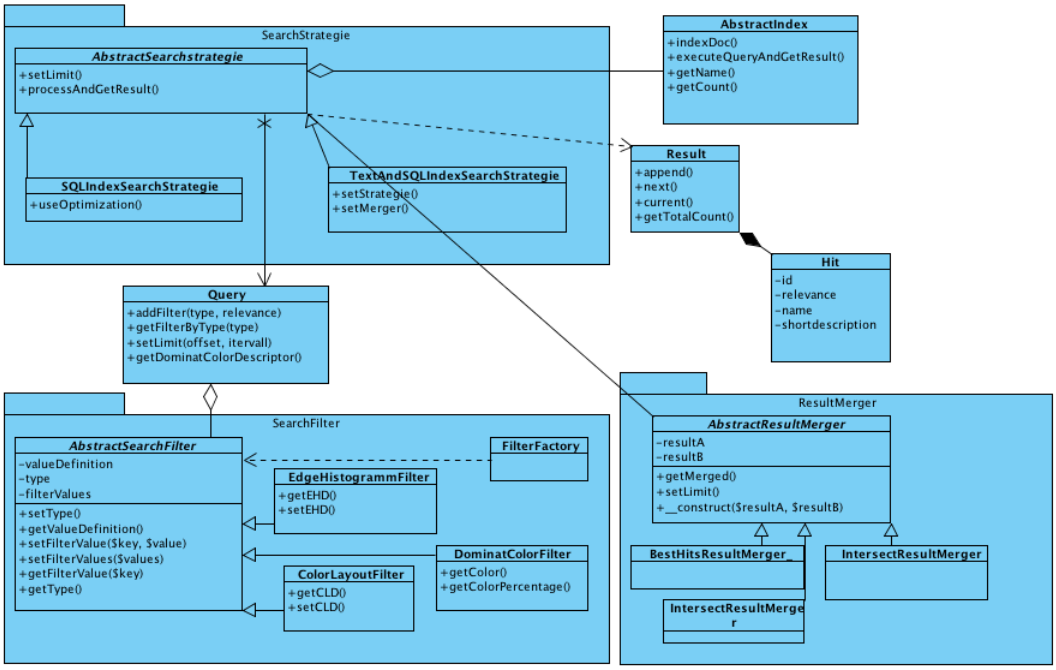

Abbildung 5.4: Klassendiagramm der Beispielapplikation für die Suche

Filtern, die die eigentliche Suchanfrage definieren. Ein Filter kann dabei vollkommen unterschiedliche Merkmale beschreiben, so zum Beispiel eine vorkommende Farbe, einen Ort oder einen Textausdruck.

Das Sequenzdiagramm in Abbildung 5.5 veranschaulicht den prinzipiellen Ablauf nach Senden einer Suchanfrage durch einen Nutzer.

5.4 Details zu Indexen

Wie erwähnt sieht das Domain Model der Applikation theoretisch beliebige Indexe vor. Ein Index ist dabei verantwortlich für das Indexieren von Multimedia Dokumenten und natürlich für das Beantworten einer Suchanfrage (Query).

Technisch muss eine konkrete Implementierung eines Indexes das dafür spezifizierte Interface implementieren. Wie es die Aufgaben erledigt und welche Technologie genutzt wird, ist dabei in der Verantwortung des entsprechenden Indexes. Ein Index muss auch nicht alle Filter eines Querys unterstützen. Durch die Möglichkeit die Rückgabe-Ergebnisse von mehreren Indexen im Rahmen einer Suchstrategie zu verknüpfen stehen die Möglichkeiten zur Verfügung, die Stärken verschiedener Indexe miteinander zu verknüpfen.

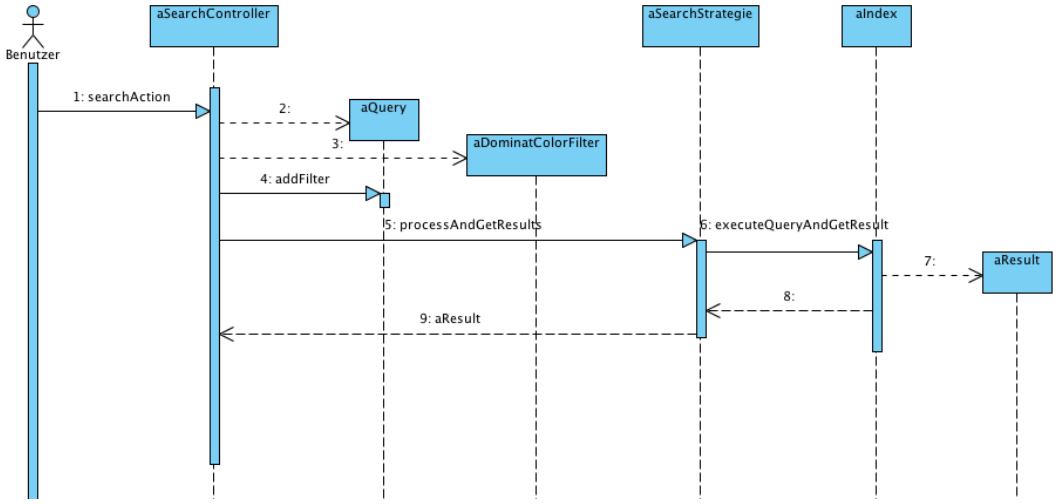

Abbildung 5.5: Sequenzdiagramm einer einfachen Suche

Um konkreter zu werden, stelle ich drei Beispiele möglicher Index Implementierungen vor:

- Index, welcher sich verstärkt auf die Auswertung textbasierter Querys fokussiert. Als Technologie kommt beispielsweise die Anbindung einer Lucene Suche in Frage.
- Index um semantische Filter auszuwerten. Dazu zählen Anfragen, welche sich auf bestimmte Features einer MPEG-7 Beschreibung beziehen, wie beispielsweise die Suche nach dominanten Farben.
  - Solch ein Index kann zum Beispiel MPEG-7 Beschreibungen in einer XML Datenbank speichern, und mit Hilfe von XQuery Abfragen relevante Dokumente finden. Da man in XQuery Anfragen auch komplexe mathematische Ausdrücke berechnen kann, ist es beispielsweise möglich, für die Berechnung der Ähnlichkeitsmaße zwischen visuellen Descriptoren XQuery zu benutzen.
  - Eine andere Möglichkeit ist es, die Werte der relevanten MPEG-7 Descriptoren beim Indexierungsvorgang optimiert zu speichern und die indexierten Werte bei der Auswertung einer Anfrage entsprechend zu nutzen.

5.5 XQuery Index

Wie beschrieben obliegt es dem eigentlichen Index wie er die Suchanfrage beantwortet. Dazu wurde auch die Möglichkeit untersucht, die Suchanfrage mit Hilfe von XQuery auszuwerten. So wurden die MPEG-7 XML Dokumente in einer XML Datenbank abgelegt.

Die Suchanfrage wurde dann in eine XQuery Anfrage übersetzt. An Hand der Suche nach dominanten Farben sei das prinzipielle Vorgehen dazu im Folgenden erläutert.

Ziel ist es, die Dokumente mit dem geringsten Abstand zum anfragenden DCD zu ermitteln. Eine einfache Abfrage über alle MPEG-7 Dokumente in einer XML Datenbank könnte demnach als XQuery wie folgt aussehen:

```
xquery version "1.0";
declare namespace MPEG-7="urn:mpeg:MPEG-7:schema:2001";

(: DCD Descriptor der Suchanfrage:)
declare variable $DCDREF as item() {
<MPEG-7:VisualDescriptor>
....
</MPEG-7:VisualDescriptor>
};

(: Namespaces für Hilfsfunktionen :)
declare namespace mpegquery="http://typo3-media.com/MPEG-7XQuery/1/";
declare namespace mpegqueryhelp="http://typo3-media.com/MPEG-7XQuery/help/";

....


for $MPEG-7Description in collection("MPEG-7emir")/MPEG-7:MPEG-7/MPEG-7:Description
let $DCDDescriptor in $MPEG-7Description/MPEG-7:MultimediaContent/MPEG-7:Image/
       MPEG-7:VisualDescriptor[@xsi:type="DominantColorType"][1],
       $DCDDist:=mpegquery:calculateDCDDistance($DCDREF,$DCDDescriptor)
order by $DCDDist
return
    <resultitem>
      <DCDDist>{$DCDDist}</DCDDist>
      <Media>{$MPEG-7Description//MPEG-7:MediaUri[1]}</Media>
    </resultitem>
```

In dieser Abfrage wird von der Instanzierung eines DCD in einem StillImage D ausgegangen. Dabei wird jedes Description Element der MPEG-7 Dokumente in der entsprechenden Datenbank-Kollektion an die Variable $\$MPEG-7Description$ gebunden, und für den gefundenen DCD wird das Ähnlichkeitsmaß mit dem Vergleichs-DCD berechnet und an die Variable $\$DCDDist$ gebunden. Das Ergebnis wird dann sortiert nach diesem Maß ausgegeben.

Für die Berechnung des Ähnlichkeitsmaßes $\$DCDDist$ wurden entsprechende Funktionen im XQuery Ausdruck definiert. Interessant sind die erreichten Ergebnisse mit der

XML Datenbank „eXist“ [11] .

Testergebnisse:

- 10 Dokumente: 0,8 Sekunden Abfragedauer
- 221 Dokumente: 11 Sekunden Abfragedauer

Der Ansatz XQuery zu verwenden und die MPEG-7 Dokumente einfach in einer XML Datenbank abzulegen, hat seine Vorteile, aber da bereits bei rund 200 Dokumenten die Auswertung der Suche über 10 Sekunden in Anspruch genommen hat, wurde dieser Ansatz nicht weiter verfolgt.

Stattdessen wurde eine optimierte Indexierung der relevanten Eigenschaften des MPEG-7 Dokumentes näher untersucht, welche in den nächsten Kapiteln beschrieben wird.

5.6 Funktionsweise des SQL Index

Für die Beispielapplikation kommt ein Index zum Einsatz, der die relevanten Eigenschaften eines Multimedia-Dokumentes in einer relationalen Datenbank speichert und für das Retrival SQL nutzt. Dieser Index soll nun genauer vorgestellt werden, um im nächsten Kapitel Performanceoptimierungen am Beispiel dieses SQL Indexes zu betrachten.

5.6.1 Aufbau der Indextabelle

Ziel ist es, die Werte der Descriptoren, die für die Auswertung einer Suche benötigt werden, in einer Relation zu speichern. Zum Einsatz kommt dabei genau eine Hauptrelation, die im weiteren Verlauf Indextabelle genannt wird, in der alle benötigten Werte eines Multimedia-Dokumentes gespeichert werden. Dabei gibt es für ein Multimedia-Dokument genau einen Datensatz in der Indextabelle.

Neben Feldern wie Name und Url des Dokumentes gibt es zahlreiche Felder, die für das Speichern der benötigten Descriptoren verwendet werden. Welche Felder das im einzelnen für die jeweiligen Descriptoren sind, wird in den folgenden Abschnitten beschrieben.

5.6.2 Indexierung

5.6.2.1 Indexieren des Dominant Color Descriptors

Wie in Kapitel 3.2 vorgestellt, besteht der Dominant Color Descriptor im Wesentlichen aus n Farbwerten im RGB Farbraum und den zugehörigen Anteilen im Bild. Genau diese

[11] eXist ist ein Open Source XML-Datenbanksystem

Werte werden in der Indextabelle zu einem Multimedia-Dokument gespeichert. Die für die Indexierung benötigten Felder für eine Farbe $C[i]$ sind dabei wie folgt benannt definiert: $dcd[i]p$ für den Anteil in Prozent und $dcd[i]r$, $dcd[i]g$, $dcd[i]b$ für den Rot-, Gelb- bzw. Blauanteil der Farbe, wobei $[i]$ für die Nummerierung der Farbwerte im Descriptor steht. Es werden jeweils nur die sieben am häufigsten vorkommenden Farben des Dominant Color Descriptors gespeichert. Die Nummerierung der Farbwerte erfolgt absteigend nach der Häufigkeit des Farbwertes, so dass die relevanteren Farben im unteren Nummerierungsbereich liegen. Sollen also weniger Farben in der Abstandsberechnung beachtet werden, so kann man zunächst die Farbwerte mit geringer Häufigkeit ausschließen.

5.6.2.2 Indexieren des Color Layout Descriptors

Wie in Kapitel 3.3 vorgestellt, besteht der Color Layout Descriptor aus den Koeffizenten einer DCT[12] über 64 Farben im YCbCr Farbraum. Diese Koeffizenten werden auch direkt in der Indextabelle abgespeichert. Die Felder sind folgendermaßen benannt: $cldy[i]$ für die Koeffizenten des Y Anteils und $cldcb[i]$ bzw. $cldcr[i]$ für den Cr bzw Cb Anteil, wobei $[i]$ jeweils für die Position des Koeffizenten steht. Es werden jeweils sechs Koeffizienten, also insgesamt 18 Werte für den Color Layout Descriptor gespeichert .

5.6.2.3 Indexieren des Edge Histogramm Descriptors

Wie in Kapitel 3.4 vorgestellt, besteht der Edge Histogramm Descriptor aus 80 so genannten Bin Werten für das Histogramm. Da für das Abstandsmaß auch die Histogrammwerte des Gesamtbildes und der so genannten SemiGlobal Unterbilder benötigt werden, werden auch diese Werte bereits berechnet und in der Indextabelle gespeichert, so dass diese Werte bei der Auswertung einer Suche nicht extra berechnet werden müssen.

Demnach gibt es folgende Felder für den Descriptor in der Indextabelle:

- Für die Histogrammwerte der Unterbilder $eh\_lb\_[i]$, mit Werten von 0 bis 79 für $[i]$
- Für die Histogrammwerte des Gesamtbildes $eh\_gb\_[i]$, mit Werten von 0 bis 4 für $[i]$
- Für die Histogrammwerte der SemiGlobal Unterbilder $eh\_sgb\_[i]$, mit Werten von 0 bis 63 für $[i]$

[12] Diskreten Kosinus Transformation

5.6.3 Abfragen

Wie bereits weiter oben vorgestellt, ist das grundlegende Prinzip ein geeignetes Ähnlichkeits- oder auch Abstandsmaß zwischen den anfragenden Descriptoren und den gespeicherten Descriptoren zu berechen. Aus der Suchanfrage selbst werden die entsprechenden Descriptoren ermittelt, die den definierten Suchfiltern entsprechen, so dass man die unter Kapitel 3 beschriebenen Ähnlichkeitsmaße verwenden kann.

Da mit SQL auch mathematische Ausdrücke auswertet werden können, wird die Berechnung des Abstandes direkt mit SQL durchgeführt. Die Ergebnissmenge wird dann, mit Hilfe des ORDER BY Ausdruckes einer SQL Abfrage, nach dem berechnetem Abstand sortiert, wobei der kleinste Abstand die größte Ähnlichkeit bedeutet.

Sind verschiedene Descriptoren in der Suchanfrage enthalten, so werden die einzelnen Ähnlichkeitsmaße entsprechend kombiniert. Der SQL Index in der Beispielapplikation unterstützt die drei Descriptoren: Color Layout, Dominant Color und Edge Histogramm. Demnach berechnet sich das Ähnlichkeitsmaß zwischen einem Datensatz *row* in der Indextabelle und den anfragenden Descriptoren nach folgender Formel, wobei D_{dcd}, D_{ehd} und D_{cld} die im Kapitel 3 vorgestellten Ähnlichkeitsmaße als SQL Ausdruck berechnen. Mit den Parametern w_{dcd}, w_{ehd} und w_{cld} können die einzelnen Abstandsmaße entsprechend gewichtet werden.

$$\begin{aligned} D_{all}(DCD, EHD, CLD, row) = w_{dcd}D_{dcd}(DCD, row) + w_{ehd}D_{ehd}(EHD, row) \\ +w_{cld}D_{cld}(CLD, row) \end{aligned} \tag{5.1}$$

Somit sieht eine Anfrage gegen die Datenbank wie folgt aus:

```
SELECT *, (<Abstandsberechnung>) as distance
  ORDER BY distance
```

5.6.3.1 Berechnung des Abstandes für einen Dominant Color Descriptoren

Als ein Beispiel soll ein Ausschnitt aus der Berechnung des Abstandsmaßes für einen gegebenen Dominant Color Descriptor mit einem Farbwert, der die relative Häufigkeit von 80% hat, dienen. Der Farbwert soll die Werte 242, 183 und 33 für Rot, Grün und Blau haben. Entsprechend dem definiertem Ähnlichkeitsmaß (siehe 3.2.4) kann eine Berechnung in einem SQL Ausdruck wie folgt aussehen:

```
SELECT * ,  (<gewichtung>* (
GREATEST(
 (1- ( SQRT( POW((242-dcd1r),2)+POW((183-dcd1g),2)+POW((33-dcd1b),2)) / 64 ) )
 ,0)   * LEAST(dcd1p,0.8)+

GREATEST(
  (1- ( SQRT( POW((242-dcd2r),2)+POW((183-dcd2g),2)+POW((33-dcd2b),2)) / 64 ) )
  ,0) * LEAST(dcd2p,1)+

... (usw. weiter für die übrigen indexierten Farbwerte)

))  as distance
FROM index
ORDER BY distance
```

Für die anderen Descriptoren lassen sich die Ähnlichkeitsmaße analog mit Hilfe eines SQL Ausdruckes berechnen. Es wird deutlich, dass eine Berechnung mit mehreren Descriptoren eine große Menge an zu berechnenden Ausdrücken enthält, die natürlich ihre Ausführungszeit in der Datenbank benötigt. Das bringt uns zum Thema des folgenden Kapitels, in dem mögliche Performanceoptimierungen beschrieben und untersucht werden.

6 Performanceoptimierungen des SQL Index

Das Grundproblem einer Abfrage, wie im vorangegangenen Kapitel beschrieben, wird recht schnell deutlich: Um die Dokumente mit dem geringsten Abstand zu ermitteln, werden die Abstände zu jedem indexierten Dokument in der Indextabelle berechnet. Hier helfen auch keine zusätzlichen Datenbankindexierungen, da die Berechnung des Abstandes mit Hilfe der Werte des anfragenden Descriptors passieren muss.

Mit diesem Vorgehen steigt der Aufwand mindestens linear zur Anzahl der indexierten Dokumente und es ist klar, dass das für riesige Datenmengen nicht praktikabel ist. Also müssen Möglichkeiten gefunden werden, die Auswertung einer Suchanfrage zu optimieren und insbesondere das Skalierungsverhalten mit steigender Anzahl von zu durchsuchenden Dokumenten zu verbessern.

6.1 Optimierungsansätze

Prinzipiell sind folgende Ansätze für Laufzeitoptimierungen denkbar:

- Caching: Gleiche Anfragen können ggf. aus einem Cache beantwortet werden. Dies ist jedoch für die meisten Use-Cases nicht praktikabel, da eine hohe Anzahl verschiedener Anfragen zu erwarten ist. Laut Google sind etwa 25 Prozent aller aktuellen Anfragen vorher noch nie da gewesen[1].
- Günstige Algorithmen für die Berechnung des Abstandsmaßes verwenden. Dies alleine reduziert zwar nicht die Skalierung in Bezug auf die Anzahl von zu durchsuchenden Dokumenten, kann aber erheblich zu einer schnelleren Berechnung der Ergebnisse beitragen.
- Vorberechnung der Abstandsmaße oder Teile der Abstandsmaße. Dies gestaltet sich aber in den meisten Fällen schwierig, da immer die Werte der Suchanfrage für die Berechnung benötigt werden, und man dementsprechend nur in geringem Maße vorberechnete Werte nutzen kann.

[1] Aussage von Udi Manber, Google's VP of Engineering. Flgl. http://www.readwriteweb.com/archives/udi_manber_search_is_a_hard_problem.php

- Skalierung über zusätzliche Hardwareressource. Dabei ist das Ziel, die Auswertung einer Suche auf verschiedene Systeme zu verteilen, um die Antwortzeit zu verringern. Dieser Ansatz soll hier nicht weiter vertieft werden.

- Kostengünstige Einschränkung der Menge auszuwertender Dokumente. Das heißt, dass vorab eine Teil-Menge aller indexierten Dokumente gebildet wird, über die die eigentliche Anfrage ausgeführt wird. Dieser Ansatz ist am vielversprechendsten, da somit ein deutlich besseres Skalierungsverhalten möglich ist.

Im Rahmen der Arbeit wird der letzte Punkt weiterverfolgt und untersucht. Demnach kann man die Auswertung einer Suchanfrage in zwei Schritte teilen:

- Bilden einer geeigneten Untermenge. Ziel dieses Schrittes ist es, eine möglichst kleine Untermenge zu bilden, in der die Ergebnisse der Suche vermutet werden. Diese Untermenge wird im Folgenden Vorabselektion genannt. Der Aufwand dafür sollte kleiner als $O(n)$[2] sein.

- Durchführen der Abstandsberechnung auf der gebildeten Untermenge. Unter der Annahme, dass die im Schritt 1 gebildete Untermenge eine gewisse Größe nicht überschreitet, kann der Aufwand dieses Schrittes als konstant angesehen werden.

6.1.1 Bilden einer Vorabselektion

Die hier weiterverfolgte Grundidee zur Optimierung der Laufzeit einer Anfrage besteht demnach darin, die Untermenge der Dokumente, für die die Berechnung des Abstands erfolgt, vorab möglichst schnell und günstig zu bestimmen. Dabei ist das Ziel bei dieser Eingrenzung möglichst keine potentiellen Treffer auszuschließen, andererseits aber die Menge soweit einzuschränken, dass möglichst wenig Dokumente für die eigentliche Auswertung übrig bleiben.

Das im Rahmen der Beispielapplikation genutzte Prinzip soll zunächst anhand eines einfachen Beispiels erklärt werden.

6.1.2 Beispiel einer dynamischen Umquadratsuche

Im Folgenden soll ein einfaches Beispiel helfen das grundlegende Konzept einer solchen Optimierung zu erläutern. Jeder kennt vermutlich die typischen Suchen nach Orten im Umkreis bestimmter Koordinaten, wie Sie beispielsweise mit Google Maps[3] möglich ist.

[2] Die so genannte „groß O Notation“ beschreibt den Aufwand eines Algorithmus in einer einfachen Funktion in Bezug auf die gegen Unendlich wachsende Größe bzw. Menge der Argumente. O(n) bedeutet dabei das der Aufwand linear wächst.

[3] Online Kartendienst mit der Möglichkeit nach Orten zu suchen. Flgl. http://maps.google.com

Stellen wir uns eine kleine Beispielapplikation vor, in der zahlreiche Orte in einer Indextabelle liegen, zu jedem Ort sind die Koordinaten (x,y) gespeichert. Aufgabe der Applikation ist es, zu gegebenen Koordinaten x und y die 100 nächstgelegenen Orte zu finden. Wie bei dem SQL Index, soll die Anfrage auch hier mit SQL beantwortet werden. Die Berechnung des Abstandes erfolgt hier einfach über den Satz des Pythagoras. Demnach kann eine SQL Abfrage wie folgt aussehen:

```
select *, SQRT(POW(x-2000,2)+POW(y-2500,2)) as distance from test
order by distance limit 0,100
```

Das RDBMS muss für die Auswertung dieses Querys den Abstand zu jedem Ort in der Indextabelle berechnen. Um die späteren Optimierungsmaßnahmen besser zu bewerten, wurde die o.g. Anfrage auf einem Testsystem über einer Datenbank Relation mit 1,4 Millionen Datensätzen ausgeführt. Die durchschnittliche Antwortzeit Betrug 1,6 Sekunden.

Eine Eingrenzung ist hier relativ einfach möglich, indem wir ein so genanntes Umquadrat um die Koordinaten x und y aufspannen, in dem wir die Ergebnismenge vermuten. Eine SQL Anfrage kann dann so aussehen:

```
select *, SQRT(POW(x-2000,2)+POW(y-2500,2)) as distance from test
where x<3000 AND x>1000 AND y < 3500 AND y > 1500 order by distance limit 0,100
```

Damit wird die Anzahl der Datensätze, für die die Berechnung erfolgen muss, deutlich verringert. So erreicht die Anfrage auf der Testumgebung nunmehr eine Antwortzeit von durchschnittlich 0,38 Sekunden.

Jetzt bleibt noch die Aufgabe das Umquadrat passend zur Anfrage möglichst optimal aufzuspannen. Mit einem einfachen Algorithmus kann man sich dabei an eine geeignete Dimensionierung annähern: Man startet zunächst mit einer durchschnittlichen Größe des Umquadrates und verdoppelt oder halbiert dieses, wenn zu wenig oder zu viele Orte im Umquadrat lokalisiert sind.

```
//CALL function to get startvalue for delta
delta = getStartDelta(x,y)

WHILE ( NOT foundOptimalDelta)
  actualCount = checkResultCount(delta,x,y)

  IF ( actualCount > resultCountMin AND actualCount < resultCountMax)
```

```
      //found optimal $delta
       foundOptimalDelta=1
  ENDIF

   IF ( actualCount < resultCountMin)
       //CALL optimize function to find new delta
      delta= increaseDelta(delta,lastCount,actualCount)
  ENDIF

  IF ( actualCount > resultCountMax)
       //CALL optimize function to find new delta
      delta= decreaseDelta(delta,lastCount,actualCount)
  ENDIF

  lastCount=actualCount

ENDWHILE
```

Der Pseudocode soll lediglich die prinzipiellen Schritte verdeutlichen. Dabei wird das Umquadrat solange vergrößert, bis mindestens eine bestimmte Anzahl (*resultCountMin*) Orte darin enthalten sind. Dabei ist wichtig, dass sich die Vergrößerungs- bzw. Verkleinerungsschritte in den Funktionen *increaseDelta* und *decreaseDelta* nicht linear Verhalten, sondern sich etwa mit jedem Durchlauf verdoppeln. Sind zu viele Orte in dem Umquadrat, wird das Umquadrat wieder verkleinert. Die Kosten der Optimierung des Umquadrates sind somit O(log n).

Bei der gezeigten Optimierung der Anfrage, über das Einschränken der Ergebnismenge mit Hilfe von Vergleichsoperatoren in dem Where Bereich der SQL Anfrage, ist das Ziel möglichst wenig Datensätze in die Berechnung des Abstandsmaßes einzubeziehen.

Noch effektiver kann man diese Einschränkung mit Clustering der Datensätze erreichen. Das bedeutet, dass man die Datensätze vorab bestimmten Clustern zuordnet. Bei unserem Beispiel bedeutet das, dass man beispielsweise die gesamte Fläche in sinnvolle Quadrate aufteilt, und jeden Datensatz einem der so gebildeten Cluster zuordnet. Bleiben wir bei unserem Beispiel und gehen davon aus, dass wir die Datensätze, die für das Ergebniss in Frage kommen, in den Clustern mit der id 100,101,200,201 vermuten. Die SQL Abfrage kann dann so aussehen:

```
select *, SQRT(POW(x-2000,2)+POW(y-2500,2)) as distance from test
where cluster IN LIST (100,101,200,201) order by distance limit 0,100
```

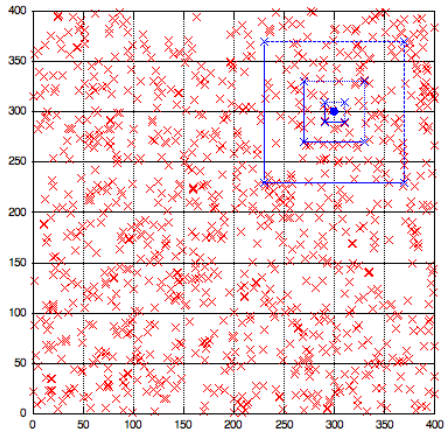

Abbildung 6.1: Visualisierung der Suche nach einem optimalem Umquadrat

Die Cluster beinhalten in diesem Beispiel dabei genau die gleiche Menge von Datensätzen, also alle Datensätze im Bereich „x<3000 AND x>1000 AND y < 3500 AND y > 1500“. Da auf dem Attribut *cluster* ein Datenbankindex angelegt wurde, kann diese Anfrage auf dem Testsystem im Durchschnitt mit 0,018 Sekunden beantwortet werden. So hat man bereits eine Steigerung der Antwortzeit um einen Faktor von 90. Ein weiterer wichtiger Vorteil des Clustering ist das Skalierungsverhalten in der Datenbank selbst.

Geht man davon aus, eine optimale Clusterung zu haben, so setzt sich der Aufwand für die Berechnung des Abstandsmaßes mit dem gezeigtem SQL Ausdruck zusammen aus (1) dem Aufwand für die Berechnung des Abstandsmaßes und (2) dem Aufwand zum Selektieren der Vorabselektion. Wobei der Aufwand (1) selbst als nahezu linear angesehen werden kann, wenn man davon ausgeht in der Vorabselektion nie mehr als eine bestimmte Anzahl von Dokumenten zu erhalten. Der Aufwand zur Ermittlung der Vorabselektion durch das RDBMS[4] ist O(log n), da das Feld *cluster* indexiert[5] ist.

Die Clusterung kann regelmäßig asynchron erfolgen, denn Sie muss optimalerweise an die Anzahl der Dokumente in der Indextabelle angepasst werden, um möglichst schnell eine optimale Vorabselektion zu finden.

[4] Relationale Datenbank Management System

[5] RDBMS nutzen beispielsweise ausgeglichene B-Bäume zum Auffinden indexierter Werte um eine Skalierung von O(log n) zu erhalten.

6.2 Umsetzung eines Clustering anhand des Dominant Color Descriptors

Wie in 3.2 erläutert beschreibt der Dominant Color Descriptor die dominanten Farben in einem Bild. Für das weitere Verständnis hilft folgende Visualisierung des Descriptors im dreidimensionalen Raum: Jede Farbe eines Dominant Color Descriptors wird durch Punkte im Raum dargestellt. Es gehören also zu einem Dominant Color Descriptor genau n Punkte, wobei n der Anzahl der Farbwerte im DCD entspricht.

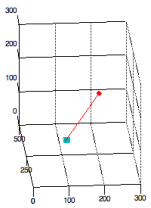

Abbildung 6.2: Visualisierung eines DCD mit zwei Farben

Wenn wir analog des gezeigten Ansatzes nach einer Möglichkeit suchen, die Menge der in Frage kommenden Dokumente für einen anfragenden DCD optimal einzuschränken, so müssen wir zunächst einen Blick auf das zum Einsatz kommende Abstandsmaß aus 3.2.4 werfen. In diesem Fall bedeutet ein höherer Wert des Abstandsmaßes eine höhere Ähnlichkeit der Dokumente.

$$D_P^2(F_1, F_1) = \sum_{1}^{N_1} \sum_{1}^{N_2} a_{1i,2j} min(p_{1i}, p_{2j}) \beta \tag{6.1}$$

Der Gleichheitskoeffizient ist wie folgt definiert

$$a_{k,l} = \begin{cases} 1 - d_{k,l}/d_{max} & d_{k,l} \leq T_d \\ 0 & d_{k,l} > T_d \end{cases} \tag{6.2}$$

Bei der Berechnung des Abstands zwischen zwei DCD werden alle Farben miteinander verglichen und der Gleichheitskoeffizent der Farben mit der geringeren Häufigkeit der Farbe multipliziert. Demnach bedeutet ein größerer Wert des Abstandsmaßes eine höhere Ähnlichkeit der DCD. Der Gleichheitskoeffizent nimmt dabei den Wert null an, wenn der euklidische Abstand zwischen zwei Farben einen bestimmten Wert T_d überschreitet. Nehmen wir als Beispiel ein Bild mit den dominanten Farben $p1$ 70% Grün und $p2$ 30% Blau. Die zu erwartenden Ergebnisdokumente müssen also Farbwerte innerhalb des, von

den Farbwerten des anfragenden DCD aufgespannten, Raumes mit dem euklidischem Abstand T_d haben. Dabei erzielen Dokumente, welche mindestens die gleiche relative Häufigkeit der entsprechenden Farbe haben, eine höhere Ähnlichkeit. Außerdem geht die Farbe Grün in unserem Beispiel mit einer höheren Gewichtung in das Abstandsmaß ein.

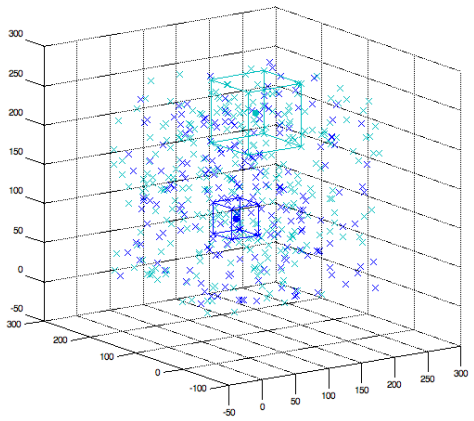

Abbildung 6.3: Visualisierung der in Frage kommenden Dokumente bei einer Suche mit zwei dominanten Farben unterschiedlicher Häufigkeit

Anhand dieser Überlegungen können folgende Aussagen getroffen werden:

- Alle in der Ergebnismenge erwarteten Dokumente haben Farbwerte innerhalb eines bestimmten Raumes um die anfragenden Farbwerte, denn Dokumente außerhalb werden als nicht ähnlich angesehen.
- Dokumente mit mindestens der relativen Häufigkeit der relevanten Farben im anfragendem DCD ergeben einen höheren Wert für das definierte Abstandsmaß.
- Der eingrenzende Raum um Farben mit einer höheren relativen Häufigkeit kann im Vergleich zum Raum um eine Farbe mit geringerer relativen Häufigkeit größer sein, da diese Farben eine höhere Gewichtung im Abstandsmaß haben. In unserem Beispiel kann eine Farbe mit dem Gleichheitskoeffizient von 1 zum blauen Farbwert nur mit 0.3 in das Abstandsmaß eingehen. Wobei eine Farbe mit dem gleichen Gleichheitskoeffizient zur grünen Farbe mit 0.7 eingehen kann. Der Raum um die Farbe Grün kann entsprechend soweit größer sein, so dass Farben mit einem Gleichheitskoeffizient von 3/7 noch inbegriffen sind, um mindestens auch mit einem Ergebnis von 0.3 in das Abstandsmaß einzugehen.

- Dokumente mit Farben in beiden einschränkenden Räumen ergeben ggf. ein höheres Abstandsmaß als Dokumente, die nur eine der Farben haben.

6.2.1 Annäherung an eine optimale Einschränkung

Hier soll nun der prinzipielle Algorithmus zur Bestimmung einer optimalen Einschränkung beschrieben werden. Ziel ist es, mit möglichst wenig Aufwand Unterräume zu finden, in denen nicht mehr als $count\_max$ und nicht weniger als $count\_min$ DCD zugeordnet sind. Analog dem unter 6.1.2 beschriebenem Algorithmus wird dafür folgendes Vorgehen genutzt:

```
LOOP
 SELECTION=bildeSelektion(DCD, SELEKTIONSPARAMETER);
 if (ermittleAnzahlDokumente(SELECTION) > count_max ) {
    if (verkleinernDerSelektionNochMöglich(SELEKTIONSPARAMETER)) {
     SELEKTIONSPARAMETER = verkleinereSelektionParameter(SELEKTIONSPARAMETER);
    }
    else {
     // zufällige auswahl der ersten count_max treffer in der selektion
     SELECTION=slice(SELECTION,count_max)
     foundOptimalSelection(SELECTION)
    }
 }
 else if (ermittleAnzahlDokumente(SELECTION) < count_min) {
    if (vergroessernDerSelektionNochMöglich(SELEKTIONSPARAMETER)) {
     SELEKTIONSPARAMETER = vergroessereSelektionParameter(SELEKTIONSPARAMETER);
    }
    else {
     //keine Optimierung möglich, da selbst bei größter Selektion
// nicht genug Dokumente selektiert werden.
// (nicht genug ähnliche Dokumente vorhanden)
    }
 }
 else {
   foundOptimalSelection(SELECTION)
 }
```

Im Schritt *bildeSelektion* wird dabei mit Hilfe des DCD und der Steuerungsparameter die Vorabselektion gebildet. Eine Selektion wird dabei aus den aufgespannten Räumen um die Farbwerte des DCD gebildet. Über die Steuerungsparameter wird bestimmt, wie groß der aufgespannte Raum um jeden Farbwert ist und welche Farbwerte überhaupt für die Selektion verwendet werden sollen. Im optimalen Fall ist nach dem ersten Schritt die

Selektion bereits optimal. Andernfalls werden die Selektionsparameter mit wachsender Schrittweite angepasst:

1. Vergrößert man den Raum um die Farbwerte, so werden mehr Dokumente selektiert.
2. Verkleinert man den Raum um die Farbwerte, so werden weniger Dokumente selektiert. Bei der Vergrößerung und der Verkleinerung sollte man die relative Häufigkeit der anfragenden Farbwerte beachten.
3. Verringert man die Anzahl der Farbwerte, so werden mehr Dokumente selektiert. Erhöht man die Anzahl werden entsprechend weniger Dokumente selektiert.

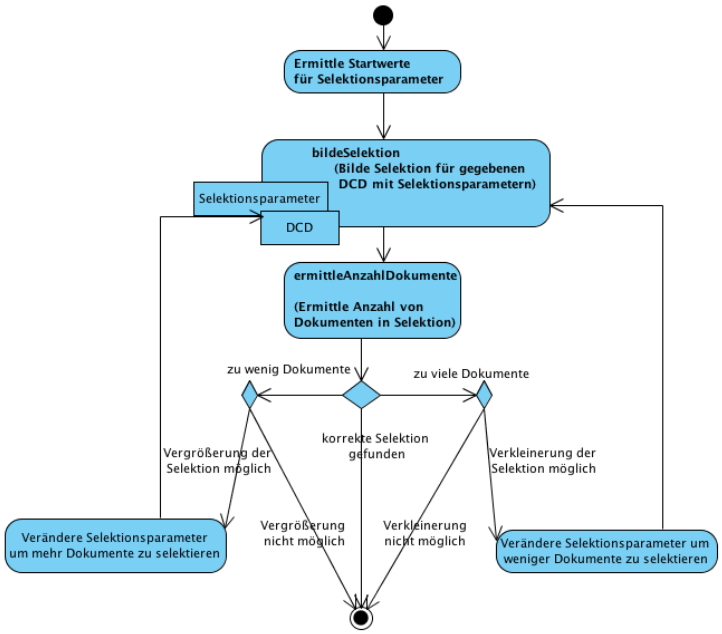

Abbildung 6.4: Aktivitätsdiagramm zum Finden der optimalen Vorabselektion

Der Ablauf ist auch im Aktivitätsdiagramm in Abbildung 6.4 ersichtlich. Wie auch in dem einfachen Beispiel der Umquadratsuche (6.1.2) ist es entscheidend, wie dieser Algorithmus skaliert. Dabei sind zwei Faktoren am wichtigsten: der Aufwand der Funktion *ermittleAnzahlDokumente* und die Anzahl der Wiederholung der Schritte, bis eine optimale Selektion gefunden wurde.

Die Funktion *ermittleAnzahlDokumente* muss für die bestimmten Cluster in der Selektion die Anzahl der Dokumente ermitteln, welche Punkte in jedem der Räume haben, denn nur dann weist das Dokument die entsprechend ähnlichen Farbwerte auf. Der Aufwand dafür sollte in der Größenordnung von O(log n) liegen, um ein akzeptables Skalierungsverhalten mit wachsender Dokumentenanzahl zu haben.

Die Anzahl der Wiederholungen der Schritte kann durch günstiges Setzen der Startwerte für die Selektionsparameter sowie optimales Anpassen der Selektionsparameter optimiert werden. Dadurch, dass die Selektionsparameter mit größer werdenden Schritten angepasst werden, kann erreicht werden, dass die Anzahl nötiger Schritte auch in der Größenordnung von O(log n) skaliert.

6.2.2 Implementation durch Clustering in der Datenbank

Wie bereits beschrieben, ist es das Ziel möglichst effektiv alle Dokumente zu ermitteln, die dominante Farben in der Nähe eines oder mehrerer gegebener Farben haben. Dazu wird der dreidimensionale RGB Farbraum in Unterräume - im Folgenden Cluster genannt - unterteilt und Dokumente mit Farben in einem bestimmten Cluster werden diesem Cluster zugeordnet.

Die Größe der Cluster muss so gewählt sein, das die Einschränkung der Dokumente zu einer Anfrage möglichst effektiv erfolgen kann, d.h. die Cluster dürfen nicht unnötig viele Dokumente beinhalten, da sonst nicht genug eingeschränkt werden kann. Gleichzeitig dürfen die Cluster nicht zu wenig Dokumente beinhalten, da sonst zu viele Cluster zusammengefasst werden müssen um ausreichend viele Dokumente zurückzuliefern. Da dies abhängig von der Anzahl der indexierten Dokumente ist, wurden die Identifier[6] so ausgelegt, dass sich die Cluster später einfach zusammenfassen oder teilen lassen. Dazu wird der verfügbare Raum in jeweils acht gleich große Cluster unterteilt, diese werden durchnummeriert von 0 bis 7, d.h. es werden genau 3 Bit für die Identifikation eines Clusters benötigt.

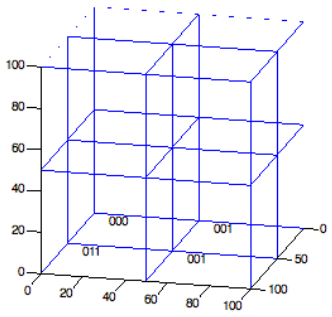

Abbildung 6.5: Visualisierung der Clusterung mit einer Clustertiefe von Eins

[6] der Identifier dient der eindeutigen Identifikation eines Clusters

Jeder Cluster selbst kann wieder unterteilt werden. Die Anzahl der wiederholten Unterteilung von Clustern soll Clustertiefe genannt werden. Entsprechend wird zur Identifizierung eines Clusters je 3 Bit pro Clustertiefe benötigt.

Wie bei der Betrachtung des Abstandsmaßes beschrieben, werden Farben, welche einen größeren euklidischen Abstand als T_d haben, als nicht ähnlich definiert. Da ein typischer Wert für T_d bei 30 bis 35 liegt, sollte die Clustergröße nicht mehr als 16x16x16 betragen. Das heißt die Clustertiefe muss mindestens 4 sein. Wenn die Anzahl indexierter Dokumente im Index steigt, kann einfach die Clustertiefe erhöht werden und die Dokumente entsprechend neu den Clustern zugeordnet werden. Es ist sogar denkbar, verschiedene Clustertiefen in verschiedenen Bereichen des RGB Farbraumes zu verwenden.

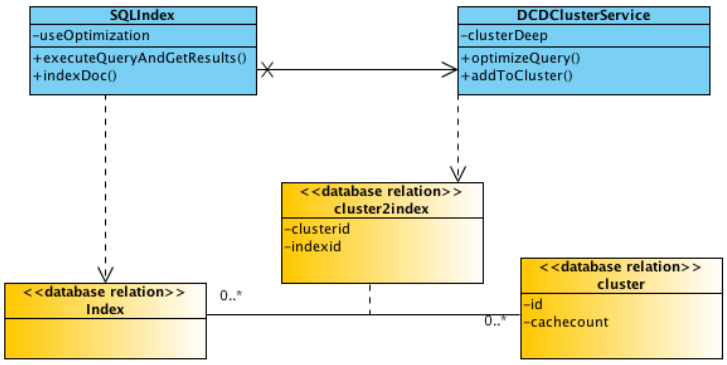

Abbildung 6.6: n:m Relation von Cluster zu Dokument

Abbildung 6.6 verdeutlicht die relevanten Relationen in der Datenbank (gelb), sowie die im Domain Model relevanten Klassen (blau):

1. Die für das Clustering wichtigste Relation in der Datenbank ist *cluster2index*. In dieser ist die n:m Beziehung zwischen Dokumenten und Clustern gespeichert. Beide Felder in der Relation haben einen Datenbankindex gesetzt.

2. Die Relation *cluster* hält die Anzahl zugehöriger Dokumente in einem Cluster, dies kann zum Beispiel verwendet werden, um zu entscheiden, wann das Clustering optimiert werden muss.

3. Im Model der Applikation ist die Optimierung in der Klasse *ClusteringService* gekapselt. Die Methode *optimizeQuery* implementiert den unter 6.2.1 beschriebenen Algorithmus zum Finden einer optimalen Vorabselektion.

Kommen wir auf den unter 6.2.1 beschriebenen Algorithmus zum Finden einer optimalen Vorabselektion zurück. Diese Ermittlung der Anzahl von selektierten Dokumenten in der Funktion *ermittleAnzahl* erfolgt über eine SQL Anfrage auf die Tabelle *cluster2index*,

wobei die Zeilen mit den zur Selektion gehörenden *clusterid* Werten selektiert werden. Diese Anfrage kann ein Limit von $max + 1$ haben, da ja nur interessant ist, ob ggf. mehr als das erlaubte Maximum an Dokumenten selektiert wurde. Da auf dem Feld *clusterid* ein Datenbankindex liegt, können diese Anfragen somit mit einem Aufwand von O(log n) verarbeitet werden.

Die gefundene Vorabselektion wird bei der eigentlichen Berechnung des Abstandsmaßes einfach als zusätzliche Bedingung im *Where* Abschnitt der SQL Anfrage beachtet. Dies kann etwa durch einen Ausdruck wie *index.clusterid IN* $(12, 13, 14)$ abgebildet werden. Auch hier liegt ein Index auf dem entsprechendem Datenbankfeld um die Ausführungszeit des Ausdruckes zu optimieren.

6.3 Validierung von Optimierungsmaßnahmen

Ohne eine ausreichende Testbarkeit von Optimierungsmaßnahmen lassen sich keine gültigen Aussagen ableiten. Aus diesem Grunde soll an dieser Stelle zunächst vorgestellt werden, wie getroffene Maßnahmen überprüft und gemessen werden.

Für die Tests wurden zunächst getrennte Indextabellen mit unterschiedlicher Anzahl von Dokumenten gefüllt. Dazu wurde der RSS Feed von Flickr[7] eingelesen und die entsprechenden Bilder indexiert. Konkret wurden so sieben Indextabellen mit einer Anzahl von 6.000, 25.700, 51.440, 102.800, 205.700, 411.500 und 823.000 Dokumenten erzeugt. Somit konnten verschiedene Optimierungen bezüglich ihres Skalierungsverhaltens untersucht werden.

Für das Ermitteln der Ergebnisse kommt ein eigener Controller zum Einsatz, welcher jeweils zufällige Suchanfragen erstellt und dem zu testenden Index zur Auswertung übergibt. Dabei ist die Anzahl der Wiederholungen, die Anzahl der unterschiedlichen Suchanfragen, sowie die Parameter für die Generierung einer Suchanfrage konfigurierbar. Wie in Abbildung 6.7 zu erkennen wird für jede Art von Suchanfrage eine durchschnittliche Antwortzeit berechnet. Dabei werden die Werte mit und ohne Optimierung nebeneinander angezeigt.

6.4 Auswertung der Optimierung des DCD

In Abbildung 6.8 sind die durchschnittlichen Antwortzeiten visualisiert. Dabei ist auf der x-Achse die Anzahl der indexierten Dokumente und auf der y-Achse die durchschnittliche Antwortzeit eingetragen. Wie zu erwarten, skaliert die Antwortzeit ohne Optimierung linear mit der Anzahl der indexierten Dokumente. Dazu sieht man die Antwortzeiten mit Optimierung, die sich in einem Bereich unter zwei Sekunden bewegen.

[7] eine Online Fotocommunity

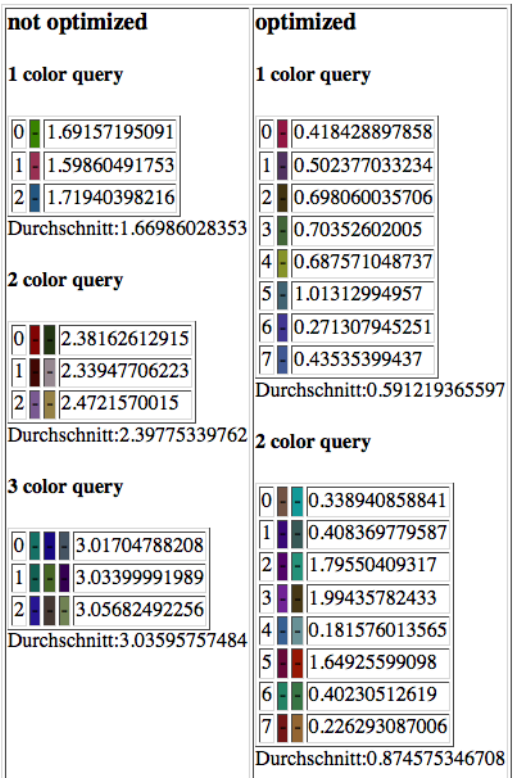

Abbildung 6.7: Screenshot der Ausgabe des Controllers zur Performanceanalyse

In der Auswertung der Optimierung in der Beispielapplikation konnte im Durchschnitt bei 823.000 Dokumenten im Index bereits mit 4 Schritten eine optimale Selektion gefunden werden. Wie bereits beschrieben, ist die Effektivität für das Finden einer optimalen Vorabselektion, neben geeigneten Startwerten für die Selektionsparameter und dem optimalem Anpassen der Selektionsparameter in jedem Schritt auch von der Clustertiefe abhängig.

In Abbildung 6.9 sind die Ergebnisse von Testläufen mit verschiedener Clustertiefe verdeutlicht. Eine höhere Clustertiefe bedeutet, dass im Durchschnitt weniger Dokumente pro Cluster selektiert werden. Damit führt eine hohe Clustertiefe bei wenigen Dokumenten im Index zu unnötig vielen Cluster, die für das Bilden der Vorabselektion zusammengefasst werden müssen. Eine geringe Clustertiefe im Gegensatz zu einer hohen Anzahl von Dokumenten im Index führt dagegen dazu, dass ein Cluster gegebenenfalls schon zu viele Dokumente selektiert und damit keine optimale Selektion erreicht werden kann. Diesen Zusammenhang erkennt man auch in der Abbildung 6.9: Eine Clustertiefe von 8 ist erst bei etwa 800.000 Dokumenten im Index optimaler als eine Clustertiefe von 5.

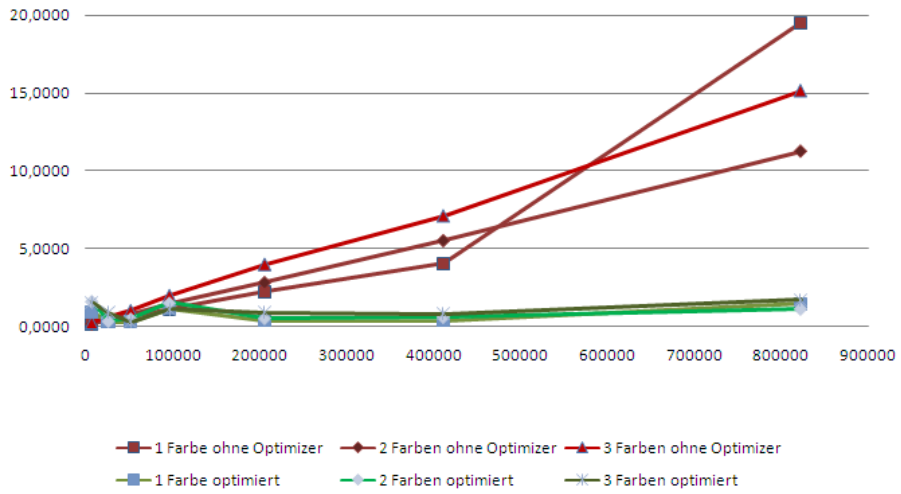

Abbildung 6.8: Visualisierung der Performanceauswertung

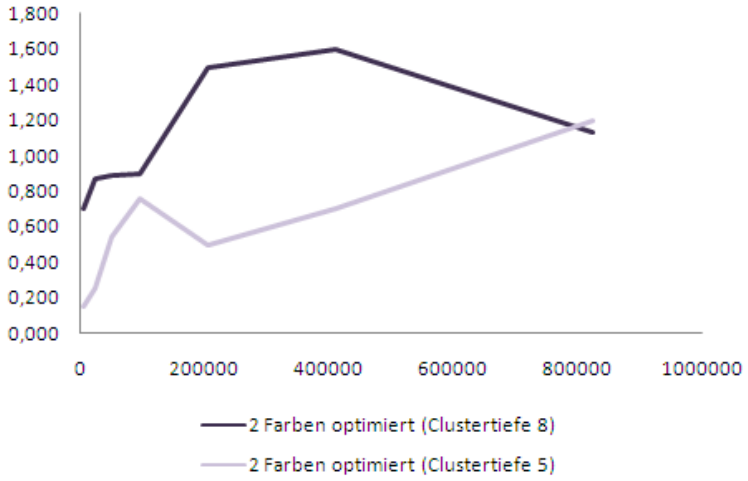

Abbildung 6.9: Antwortzeiten mit unterschiedlicher Clustertiefe

7 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde der Standard MPEG-7 näher untersucht. Mit dem Fokus auf die visuellen Beschreibungselemente im MPEG-7 Standard wurde untersucht, welche Möglichkeiten man hat, diese Beschreibungselemente für semantisches Retrieval zu nutzen.

Im Zuge dessen wurde eine Beispielapplikation mit einem allgemeinem Domain-Model vorgestellt, mit der man semantische Bildsuchen definieren und durchführen kann. Neben der Möglichkeit XQuery für Suchanfragen zu nutzen, wurde insbesondere untersucht, die semantische Suchanfragen mit Hilfe einer relationalen Datenbank auszuwerten.

Dabei lag ein Fokus darauf, Suchanfragen über eine große Menge von Dokumenten zu optimieren. Die getroffenen Überlegungen wurden in der Beispielapplikation implementiert und ermöglichten eine Analyse der Optimierungen. Durch Clustering auf Basis des Dominant Color Descriptors konnte ein deutlich besseres Skalierungsverhalten erreicht werden. So konnte für die Auswertung einer semantischen Suche nach Bildern ein Skalierungsverhalten von O(log n) erreicht und eine Beispielanfrage über etwa 800.000 Dokumente 14 mal schneller beantwortet werden als ohne Optimierung.

Anwendungen, welche die Beschreibungsvielfalt von MPEG-7 Dokumenten nutzen, wären für eine Vielzahl unterschiedlicher Bereiche und Aufgaben relevant. Eine anwenderfreundliche Software sollte möglichst intuitiv sein und dem Nutzer so viele Entscheidungen wie möglich abnehmen, da der durchschnittliche Anwender mit eventuellen Sucheinstellungen und Justierungen überfordert sein könnte. Führt man die gezeigten Überlegungen weiter, so können die in dieser Arbeit gezeigten Konzepte beispielsweise genutzt werden, um neben dem textbasierten Retrieval weitere Zugriffswege auf die ständig wachsende Flut multimedialer Inhalte zu haben.

Der Bedarf solcher Anwendungen ist, wie eingangs erwähnt, bei der immer größer werdenden Anzahl multimedialer Daten groß. Die praktische Anwendung solcher Technologien hat sich bisher aber noch nicht durchgesetzt. Es bleibt abzuwarten, wie beispielsweise eine Google Bildsuche im Jahr 2020 aussieht – und ob sich dort Aspekte der gezeigten Beispielapplikation finden.

Glossar

API Application Programming Interface, englisch für Programmierschnittstelle in der Informatik

Descriptor MPEG-7 Tool. Beschreibt Syntax und Semantik einer Merkmalsrepräsentation

Description Schemes Description Schemes bauen auf Descriptors auf und kombinieren individuel- le Descriptors und Description Schemes, und beschreiben die Struktur und Semantik der Beziehung zwischen diesen.

Domain-Model Begriff aus dem Domain Driven Design. Bezeichnet die Klassen und Methoden im Kern der Applikation

FLOW3 FLOW3 ist der Name eines neuen Open Source PHP Applikations Frameworks. Es bringt neue Ansätze in die PHP Welt und unterstützt die Ideen des agilen Paradigmas. FLOW3 wird die Basis des bekannten Open Source Enterprise Content Management System TYPO3 in der Version 5 und ist aktuell noch in der Entwicklung.

Klasse Klasse ist in der Objektorientierung ein abstrakter Oberbegriff für die Beschreibung der gemeinsamen Struktur und des gemeinsamen Verhaltens von Objekten

Pattern =Entwurfsmuster (engl. design pattern) sind bewährte Lösungs-Schablonen für wiederkehrende Entwurfsprobleme der Softwarearchitektur und Softwareentwicklung

Query by Example Üblicher Begriff für den Anwendungsfall, eine Suche an Hand eines gegeben Beispiels durchzuführen.

REST Representational State Transfer (REST) eine Architektur für verteilte Hypermediasysteme wie beispielsweise das world wide web. Wird im sprachgebrauch eingesetzt um ein einfaches webbasiertes Interface zu beschreiben, welches XML und HTTP ohne zusätzliche Abstraktion wie in SOAP einsetzt. [22]

SOAP Steht für „Simple Object Access Protocol“und ist ein Protokoll für den Austausch von Informationen in verteilten, denzentralen Umgebungen. Es besteht aus drei

Teilen und ermöglicht auch entfernte Prozeduraufrufe (RPC), die Spezifikation ist deutlich komplexer als bei XML-RPC und ermöglicht im Gegensatz zu diesem zum Beispiel die Definition eigener Datentypen.

Use-Case = Ein Anwendungsfall (engl. Use Case) definiert möglichen Szenarien, die eintreten können, wenn einer oder mehrere Akteure mithilfe des betrachteten Systems ein bestimmtes fachliches Ziel (engl. business goal) zu erreichen

WEBDav WebDAV steht für „Web-based Distributed Authoring and Versioning"und ist der Name für ein Protokoll, welches HTTP um Methoden und Eigenschaften erweitert. Es ist eins der ersten Protokolle welches auf XML basiert, und ermöglicht Nutzern das editieren und verwalten von Dateien auf Webservern.

XML-RPC Steht für „XML Remote Procedur Call "und ist ein Standart, der die Kommunikation zwischen verschiedenen Anwendungen in heterogenen Systemen ermöglicht. Als Übertragungsprotokoll wird HTTP verwendet und für die Kodierung XML. Typische Anwendung ist das Aufrufen von entfernten Prozeduren, zum Beipiel über das Internet.

XML Information Set Die XML Information Set (Infoset) Spezifikation ist eine W3C Empfehlung und definiert eine abstrakte Datenmenge mit dem Ziel eine konsistente Definition für andere Spezifikationen zu haben. Das Information Set eines XML Dokumentes besteht aus einer Reihe von so genannten Information Items, welche eine abstrakte Repräsentation eines Teils des XML Dokumentes mit den entsprechenden Eigenschaften darstellen.[12]

XML Namespaces Mit der W3C Empfehlung für Namespaces werden qualifizierte Namen (QNames) für XML Dokumente möglich. Dies hat unter anderem den Vorteil, XML-Dokumente mit bestimmter Struktur oder Elementnamen wieder verwenden zu können, ohne Nameskollisionen befürchten zu müssen. Verwendete Namespaces können im Prolog eines XML Dokumentes oder direkt im Element durch die Angabe von „xmlns:" angegeben werden. Qualifizierte Namen bestehen aus einem optionalem Namespace Prefix und einem lokalem Teil, getrennt durch einen Doppelpunkt. Ist ein Namespaceprefix vorhanden muss dieser an eine IRI[1] gebunden sein. [21]

[1] Internationalized Resource Indentifier, kann in eine URI umgewandelt werden.

Abkürzungen:

APS: Advanced Photo System
AV: Audio-visual
CIF: Common Intermediate Format
CS: Classification Scheme
D: Descriptor
Ds: Descriptors
DCT: Discrete Cosine Transform
DDD: Domain Driven Design
DDL: Description Definition Language
DS: Description Scheme
DSs: Description Schemes
JPEG: Joint Photographic Experts Group
MDS: Multimedia Description Scheme
MPEG: Moving Picture Experts Group
MVC: Model View Controller
RDBMS: Relationale Datenbank Management System
TZ: Time Zone
TZD: Time Zone Difference
URI: Uniform Resource Identifier (IETF Standard RFC 2396)
URL: Uniform Resource Locator (IETF Standard RFC 2396)
XM: eXperimentation Model
XML: Extensible Markup Language

Literaturverzeichnis

[1] B.S. Manjunath; Philippe Salembier; Thomas Sikora: *Introduction to MPEG-7* John Wiley & Sons LTD, 2003

[2] Peter van Beek; Ana B. Benitez; u.a.: *MPEG-7 Multimedia Description Schemes XM* (Version 7.0) ISO/IEC JTC 1/SC 29/WG 11/N3964, March 2001, Singapore URL: http://www.w3.org/2001/05/mpeg7/W3964_mds_xm.doc

[3] Chee Sun Won; Dong Kwon Park; u.a.: *Efficient Use of MPEG-7 Edge Histogram Descriptor* IETRI Journal, vol.24, no.1, Feb. 2002, pp.23-30. URL: http://etrij.etri.re.kr/Cyber/servlet/GetFile?fileid=SPF-1041924741673

[4] Opendrama Project @ MTG-UPF URL: http://www.iua.upf.es/mtg/opendrama/

[5] Howard Katz: *XQuery from the Experts.* Addison-Wesley, 2003

[6] Zohra Bellahsène: *Database and XML technologies: first International XML Database Symposium, XSym 2003, Berlin, Germany, September 8, 2003* Springer, 2003

[7] Masaroshi Yoshikawa; Toshiyuki Amagasa: *XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases.* In: ACM Transactions on Internet Technology, Vol. 1, No. 1, August 2001, Pages 110-141. URL: http://delivery.acm.org/10.1145/390000/383038/p110-yoshikawa.pdf

[8] Rajasekar Krishnamurthy; Jeffrey F. Naughton; Jayavel Shanmugasundaram: *Dealing with (un)structuredness in XML Data and Queries Using Relational Databases.* URL: http://www-db.cs.wisc.edu/dbseminar/spring01/talks/sekar.pdf

[9] Wolfgang G. Stock *Information Retrieval: Informationen suchen und finden* Oldenbourg Wissenschaftsverlag, 2006

[10] Steve DeRose; Eve Maler; David Orchard: *XML Linking Language (XLink) Version 1.0.* W3C Recommendation, 2001, URL: http://www.w3.org/TR/xlink/

[11] Henry S. Thompson; David Beech; Murray Maloney; et.al: *XML Schema Part 1: Structures Second Edition.* W3C Recommendation, 2004, URL: http://www.w3.org/TR/xmlschema-1/

[12] John Cowan; Richard Tobin: *XML Information Set (Second Edition).* W3C Recommendation, 2004, URL: http://www.w3.org/TR/xml-infoset/

[13] Michael Carey; Daniela Florescu; Zachary Ives; et.al: *XPERANTO: Publishing Object-Relational Data as XML* IBM Almaden Research Center, URL: http://xml.coverpages.org/careyXperantoOverview.pdf

[14] Nastaran Fatemi; Omar Abou Khaled; Giovanni Coray: *An XQuery Adaptation for MPEG-7 Documents Retrieval* URL: http://www.idealliance.org/papers/dx_xml03/papers/05-03-01/05-03-01.html

[15] Dokumentation zu eXist: XQuery Support. URL: http://exist.sourceforge.net/xquery.html

[16] Berkeley DB XML Überblick. URL: http://sleepycat2.inetu.net/products/bdbxml.html und http://www.oracle.com/technology/products/berkeley-db/xml/

[17] Anatomy of an XML Database: Oracle Berkeley DB XML An Oracle White Paper September 2006

[18] Sleepycat Software: Introduction to Berkeley DB XML 21.10.2005

[19] Ronald P. Bourret: *XML and Databases* Last updated September, 2005 URL: http://www.rpbourret.com/xml/XMLAndDatabases.htm

[20] Robin Cover: *XML and Databases* URL: http://xml.coverpages.org/xmlAndDatabases.html

[21] Tim Bray; Dave Hollander; Andrew Layman; et.al : *Namespaces in XML 1.1.* W3C Recommendation, 2004, URL: http://www.w3.org/TR/xml-names11/

[22] Fielding, Roy Thomas: *Architectural Styles and the Design of Network-based Software Architectures.* Doctoral dissertation, University of California, Irvine, 2000, URL: http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_ arch_ style.htm

[23] C. Galindo-Legaria. Outerjoins as disjunctions. In Proc. of Int. Conf. on Management of Data (SIGMOD), pages 348-358,1994

A Diagramme zur MPEG-7 Struktur

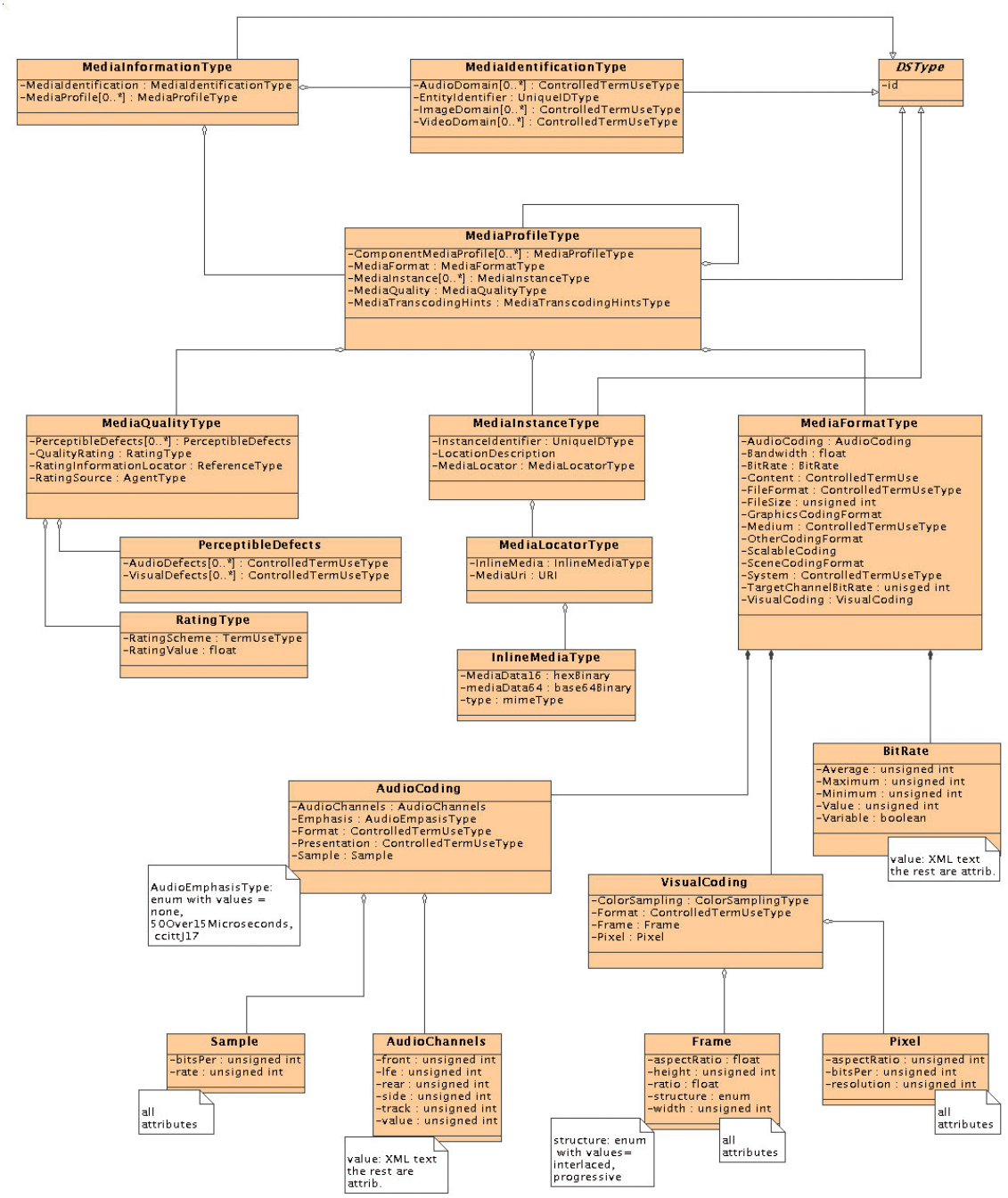

Abbildung A.1: MediaInformation [4]

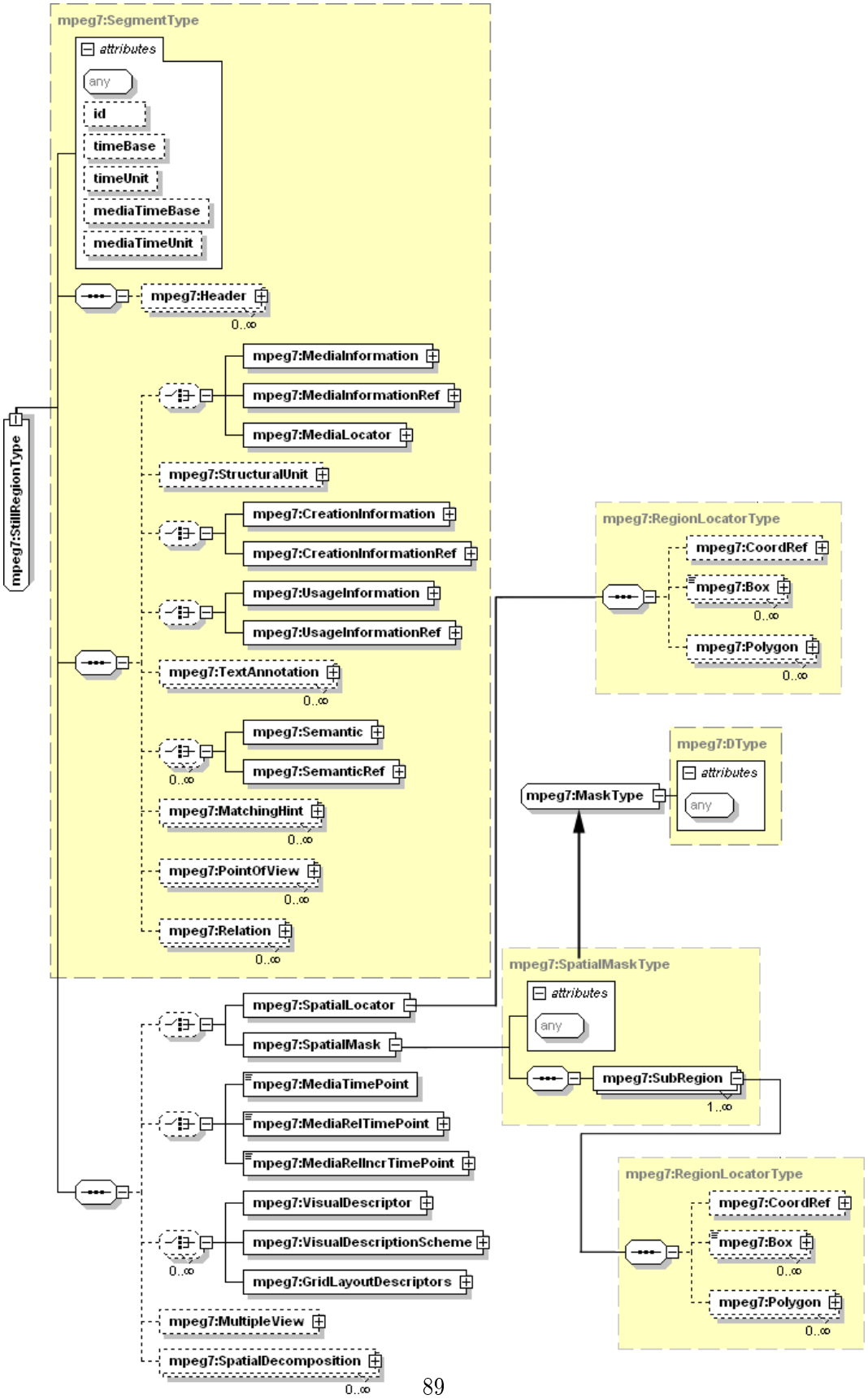

Abbildung A.2: StillRegionType im Zusammenhang mit SegmentType und RegionLocatorType

(eigene Grafik auf Grundlage der XMLSchema-Visualisierung in XMLSpy)

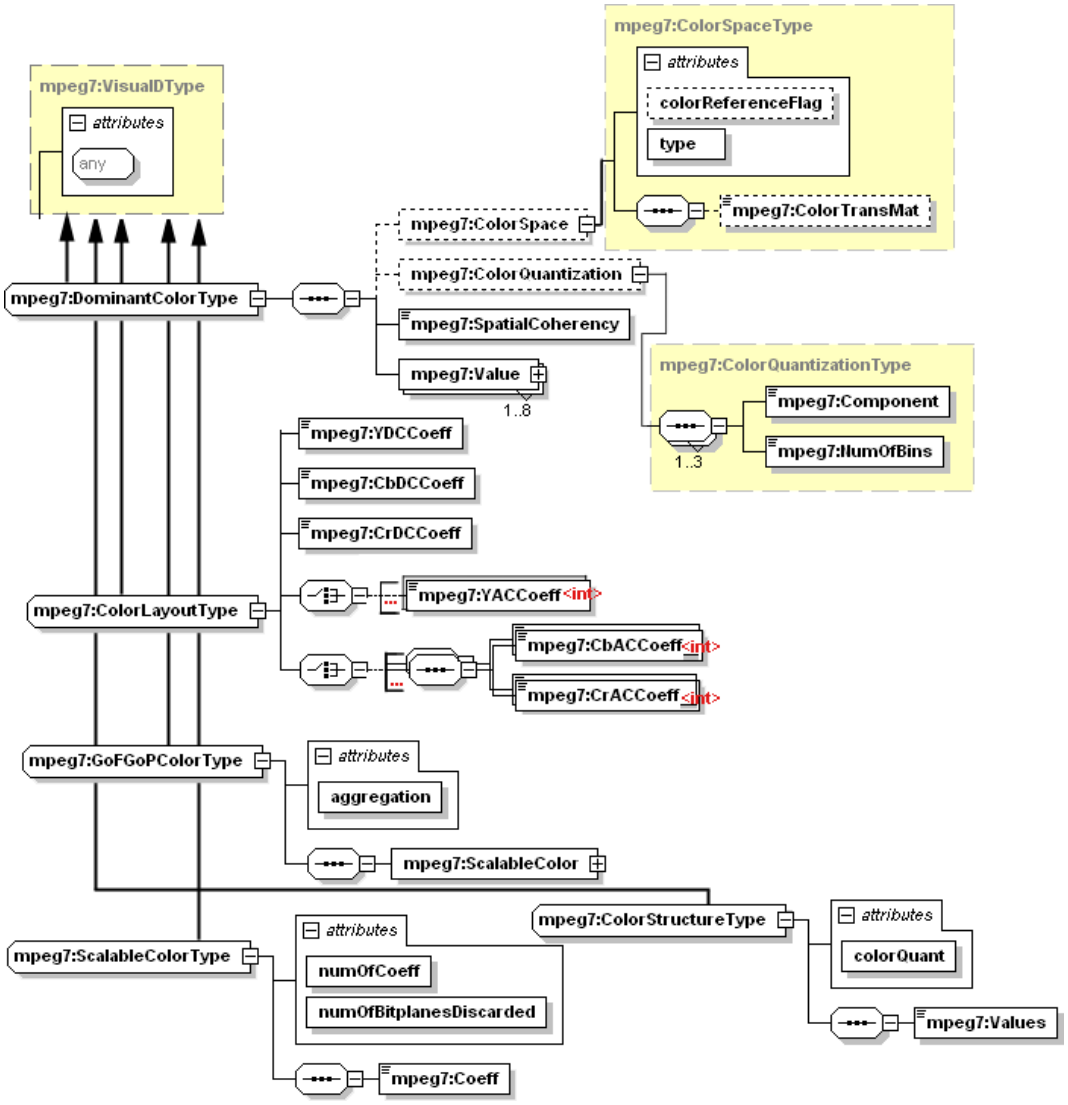

Abbildung A.3: Überblick über die Descriptoren zur Farbbeschreibung
(eigene Grafik auf Grundlage der XMLSchema-Visualisierung in XMLSpy)

B Beispiel Anwendungen und Links

B.1 MPEG7 Anwendungen

B.1.1 Caliph und Emir

Caliph & Emir sind javabasierte Applikationen um Bilder mit MPEG-7 zu beschreiben, sowie Retrivaloperationen auszuführen. Die Applikationen sind unter GPL[1] veröffentlicht und auf Sourceforge oder der Webseite www.semantic-metadata.net zu finden.

B.1.2 MIRROR

MIRROR steht für „MPEG-7 Image Retrieval Refinement based On Relevance feedback“ , ist also eine Anwendung für Bildretrival mit der Möglichkeit die Suchergebnisse durch eine Relevanz-Rückmeldung zu verfeinern. Entwickelt wurde die Anwendung von einem Team unter Dr Lai-Man Po an der City University of Hong Kong[2]

Die Anwendung kann kann über ein Webinterface[3] bedient werden.

B.1.3 Liste weiterer MPEG7 Anwendungen

Weitere Anwendungen seien nur kurz genannt:

- *MPEG-7-XM* (Referenzimplementierungen)
- *Software der MPEG7/21 Community:*[4]
  - *MECCA* An MPEG-7 based tool for multimedia capturing of collaborative scientific discourses about movies

[1] General Pulblic License
[2] http://www.ee.cityu.edu.hk/
[3] Zum Zeitpunkt der Arbeit unter http://abacus.ee.cityu.edu.hk/ mpeg7/index.html
[4] http://www.multimedia-metadata.info/Software and Tools/

  - *MEDINA* A Semi-automatic Dublin Core to MPEG-7 Converter for Collaboration and Knowledge Management in Multimedia Repositories
  - *ACIS* MPEG-7 in a Geographic Information System for Cultural Heritage Management in Afghanistan
  - *MARS* MPEG-7 based annotation of electroacoustic music
  - *A Framework for Visual Information Retrieval* Website of the „VizIR - A Framework for Visual Information Retrieval“ project (funded by the Austrian Scientific Research Funds FWF under grant number P16111-N05).
  - *LIRE - Lucene Image Retrieval* The LIRE (Lucene Image REtrieval) library a simple way to create a Lucene index of image features for content based image retrieval (CBIR). The used features are taken from the MPEG-7 Standard: ScalableColor, ColorLayout and EdgeHistogram. Furthermore methods for searching the index are provided. The LIRE library is part of the Caliph & Emir project and aims to provide the CBIR features of Caliph & Emir to other Java projects in an easy and light weight way.

- *FRAPESA* (T-Systems)
- *LAS* (SOAP MPEG-7 Services)
- *BIM* Codiert XML zu Binary
- *MPEG-7 Audio Low Level Descriptors calculator* (University of Wollongong, Australia)
- *MPEG-7 Speech Recognition engine* (Canon)
- *MPEG-7 Annotation Tool* (IBM)
- *MPEG-7 MovieTool* (Ricoh)
- *IBM VideoAnnEx Annotation Tool* http://www.research.ibm.com/VideoAnnEx/
- *Online Validierung* http://m7itb.nist.gov/M7Validation.html

Andere verwandte Applikationen und Links:

- *Fast Multiresolution Image Querying*
  Weitere Infos: http://grail.cs.washington.edu/projects/query/
  In Anwendung online in einer Suche über flickr Bilder:
  http://labs.systemone.at/retrievr
  oder unter: http://www.imgseek.net/
- *Idee Inc.* http://ideeinc.com/

- *Face Tagging in Photoshop Elements 4 Organizer*
  http://graphicssoft.about.com/od/pselements/ss/pse4whatsnew.htm
- *http://www.riya.com/* Eine neue Suchmaschine für visuelles Retrival, mit Gesichtserkennung
- *http://www.like.com/* Visuelles Produktretrival