Seminar SS 2002

Multimediale Informationssysteme

Audio / Video Retrieval

AG Datenbanken und Informationssysteme, Prof. Dr. Härder, Universität Kaiserslautern

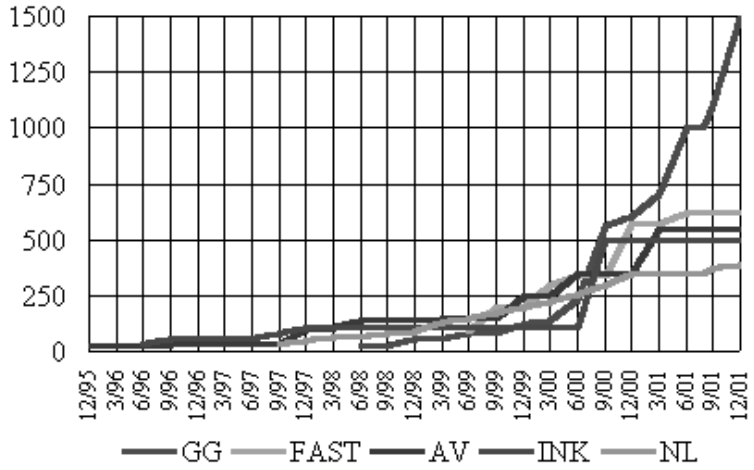
von Jens Körte

jk060674@gmx.de

Inhaltsverzeichnis

1. Einleitung

Die Suche in Video-Datenbanken gestaltet sich zunehmend schwieriger, da immer mehr Möglichkeiten der Anfrage beachtet werden müssen. Die heutigen Suchmaschinen im Internet erfüllen die Anfrage zum Beispiel nach einem Video. Bei spezielleren Anfragen, die zum Beispiel eine bestimmte Sequenz eines Video, oder ein bestimmtes Thema eines Liedes suchen, versagen die meisten Suchmaschinen. Demnach müssen je nach Art der Anfrage die Datenbanken unterschiedlich angelegt werden.
Sicherlich wäre es möglich, alle Anfrageaspekte in eine Datenbank zu stecken, jedoch übersteigt die Datenbank den eigentlichen Nutzen. Eine Datenbank, die alle möglichen Suchanfragen bearbeiten kann, aber nur ein Teil der möglichen Suchanfragen erhält, ist ineffizient. Aus diesem Grund werden Datenbanken speziell für die jeweilige Suchanfragen angelegt.
Im Internet sind die Datenbanken für eine große Gemeinde angelegt, so daß eine spezielle Suche, zum Beispiel nach der Häufigkeit des Erscheinens von Mel Gibson in Braveheart, meist zu keinem befriedigenden Ergebnis führt. Wenn man sich vor Augen führt, daß die zu indexierenden Seiten exponentiell wachsen und somit die Gesamtmenge an Daten eine unüberschaubare Menge annimmt, ist es kein Wunder, daß die Indexierung immer „hinterherhinkt". Abbildung 1 zeigt den Verlauf der indexierten Internetseiten der größten Suchmaschinen vom Dezember 1995 bis zum Januar 2001. Zu erkennen ist, daß die indexierten Internetseiten exponentiell steigen.
Die Internetseitenanzahl wurde im Frühjahr 2001 von Cyveillance [CYV] auf 2,8 Milliarden geschätzt, täglich kommen ca. sieben Millionen Seiten hinzu. Wenn man Google außer acht läßt (Google indexiert z.B. auch PDF Dokumente), haben die größten Suchmaschinen nur ca. 18% der existierenden Seiten indexiert.



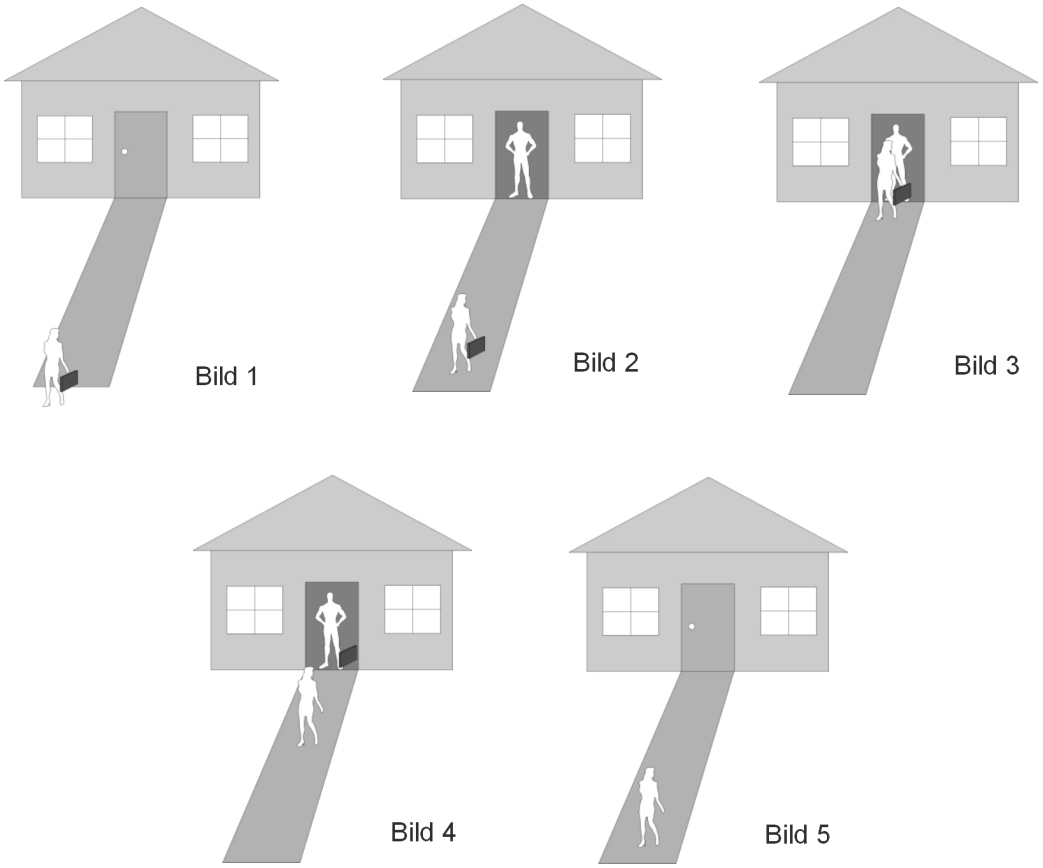
(Abb. 1) Entwicklung der Suchmaschinengröße (in Millionen Seiten)

Der Anstieg der Datenmenge ist auf die gesteigerte Rechenleistung bei gleich bleibendem Preis, den billigen Speichermedien, die Digitalisierung, etc., zurückzuführen
In diesem Seminar wird gezeigt, wie man den Inhalt eines Videos und von Audiodaten optimal in eine Datenbank speichert und Suchanfragen formuliert.

2. Video-Retrieval

2.1 Beispiel

Wir führen ein Beispiel-Video ein.
Die Polizei überwacht ein Haus und nimmt folgende Videosequenz auf (dargestellt sind nur fünf wichtige Ausschnitte):



(Abb. 2) Ausschnitte des Überwachungsvideos

Die Polizei kennt den Bewohner des Hauses. Es ist Jens Dope, ein bekannter Drogendealer. Die Polizei beobachtet, wie ein Mann, Stefan F., mit einem Koffer vor dem Haus erscheint (Bild 1) und über den Weg zum Haus geht (Bild 2). Die Tür öffnet sich und Herr Dope steht in der Tür (Bild 2). An der Tür angekommen übergibt Herr F. Herrn Dope, nach einem kurzen Gespräch, den Koffer (Bild 3). Herr F. verläßt das Grundstück (Bild 4). Herr Dope verschwindet im Haus und schließt die Tür (Bild 5). Herr F. verläßt das Grundstück.
Während dieser Filmsequenz fahren einige unbekannte Autos am Haus vorbei und eine Person läuft mit einem Hund am Haus vorbei.

2.2 Organisation

Bei der Erstellung der Datenbank muß beachtet werden, welche Aspekte für den Anwender wichtig sind und wie man diesen Inhalt am besten speichert.
Wie sieht die Anfrage an die Datenbank aus und kann man Teile des Inhalts durch einen automatischen Prozeß indexieren?

Betrachten wir das Überwachungsvideo. In diesem Fall sind möglicherweise folgende Dinge interessant:
Menschen, wie z.B. Stefan F. der den Koffer überbringt, Jens Dope, der der Empfänger ist.
Aktivitäten, wie z.B. *übergeben* (hier der Koffer), *sprechen* (das Gespräch zwischen Jens Dope und Stefan F.).
Die Aktivität *Übergeben*(Koffer, Stefan F.) bedeutet, daß Stefan F. den Koffer übergibt, oder die Aktivität *Sprechen*(Jens Dope, Stefan F) bedeutet, daß Jens Dope mit Stefan F. spricht.

Die Aktivität *Übergeben* hat, in diesem Fall, zwei Attribute:

Objekt, welches der Koffer ist
Empfänger, welches Stefan F. ist und
Geber, welches Jens Dope ist.

Jedes Video beinhaltet unterschiedliche Objekte und Aktivitäten. Objekte können unterschiedliche Attribute besitzen und diese können sich auch von Bild zu Bild ändern. Um eine Datenbank zu definieren, müssen wir einige Definitionen betrachten.

Definition 1:
Eine Paar (ename,wert) ist eine *Eigenschaft*. ename ist der Name der Eigenschaft und wert ist eine Menge. Eine Instanz einer Eigenschaft lautet ename = v, $v \in$ wert.

Als Beispiel:
(Nummernschild,X), welches ein Nummernschild beschreibt. X ist z.B. eine Menge, beinhaltend $Ort \times Alpha_2 \times Num_4$.
(Größe, R^+), welches die Größe einer Person in positiven realen Zahlen beschreiben kann.

Definition 2:
Eine *Objektbeziehung* ist ein Paar (fd,fi), bei dem gilt:
1. fd ist eine Menge von bildabhängigen (frame-dependent) Eigenschaften
2. fi ist eine Menge von bildunabhängigen (frame-independent) Eigenschaften
3. $fi \neq fd$

Als Beispiel:
Betrachten wir die Eigenschaft *Hemdfarbe*. Diese Eigenschaft kann sich von Bild zu Bild (Sonneneinstrahlung, Farbverfälschung der Kamera, etc.) ändern.

Definition 3:
Eine *Objektinstanz* ist ein Tripel (oid,os,ip). Es gilt:
1. oid ist eine eindeutige Zuweisung für die Objektinstanz (Object-ID).
2. os = (fd,fi) ist eine Objektbeziehung und
3. ip ist eine Menge von Aussagen, bei denen gilt:
   (a) Für jede Eigenschaft (ename,wert) in fi enthält ip mindestens einen Inhalt von (ename,wert) und
   (b) Für jede Eigenschaft (ename,wert) in fd und jedem Bild *b* des Videos enthält ip mindestens einen Inhalt von (ename,wert).

Als Beispiel:
Wir haben ein Objekt (Person,fd_1,fi_1) bei dem *Person* Stefan F. ist, fd enthält die Eigenschaft *haben* (Stefan F. hat den Koffer in Bild 1) und fi enthält die Eigenschaft *Größe*, welches aussagt, wie groß die Person ist (Stefan F. ist 187cm groß).

Die folgende Tabelle zeigt chronologisch die interessanten Objekte des Überwachungsvideos:

Bild	Objekte	Bildabhängige Eigenschaften
1	Stefan F. Dope's Haus Koffer	besitzt(Koffer), wo(Gehweg_anfang) Tür(geschlossen)
2	Stefan F. Jens Dope Dope's Haus Koffer	besitzt(Koffer), wo(Gehweg_mitte) wo(in_Tür) Tür(offen)
3	Stefan F. Jens Dope Dope's Haus Koffer	besitzt(Koffer), wo(Gehweg_ende) wo(in_Tür) Tür(offen)
4	Stefan F. Jens Dope Dope's Haus Koffer	wo(Gehweg_mitte) besitzt(Koffer), wo(in_Tür) Tür(offen)
5	Stefan F. Dope's Haus Koffer	wo(Gehweg_anfang) Tür(geschlossen)

Natürlich gibt es auch bildunabhängige Informationen, die jedes Objekt besitzen kann, die es wert sind, gespeichert zu werden. Eine Tabelle könnte wie folgt aussehen:

Objekte	Bildunabhängige Eigenschaften	Wert
Stefan F.	Alter	27
	Größe	187 cm
Dope's Haus	Adresse	Gottlieb-Daimler-Str. 1 67663 Kaiserslautern
	Art	Holzhaus
	Farbe	braun
Jens Dope	Alter	28
	Größe	189 cm
Koffer	Farbe	Rot
	Breite	40 cm
	Höhe	30 cm

Jetzt haben wir das Überwachungsvideo der Polizei hinreichend beschrieben. Jedoch fehlen noch Angaben über Aktivitäten.

Definition 4:
Eine *Aktivitätsbeziehung* AktBez ist eine endliche Menge, bei der gilt:
Wenn (ename,$wert_1$) und (ename,$wert_2$) in AktBez sind, dann gilt $wert_1 = wert_2$.

Als Beispiel:
Betrachten wir die Aktivität *Übergeben*, welche zwischen Stefan F. und Jens Dope stattfindet, dann hat diese Aktivität eine 3-Paar-Beziehung:
1. (Geber,Person) : Die Eigenschaft Geber ist vom Typ Person (Stefan F.)
2. (Empfänger,Person) : Die Eigenschaft Empfänger ist vom Typ Person (Jens Dope)
3. (Gegenstand,Objekt) : Dieses Paar bezeichnet den Gegenstand der ausgetauscht wird (Koffer).

Definition 5:
Eine *Aktivität* ist ein Paar, beinhaltend
1. AktID (der Name der Akivitätsbeziehung AktBez) und
2. für jedes paar (ename,wert) $\in$ AktBez gilt: ename = v, v $\in$ wert.

Als Beispiel:
Wir betrachten die Aktivität: *Objektaustausch*.
In diesem Fall hätte die Aktivität drei Paare (ename,wert):
{(Geber,Person),(Empfänger,Person),(Gegenstand,Objekt)}
In Bild 3 des Überwachungsvideos würden folgende Daten eingetragen werden:
Geber = Stefan F.
Empfänger = Jens Dope
Objekt = Koffer

2.3 Datenbankanfragen

Der Inhalt einer Datenbank unterscheidet sich von der Art der Anfrage. Im wesentlichen ist man an den folgenden vier Suchmöglichkeiten interessiert:

1.	**Teil-Anfrage:**	Finde alle Segmente von einem oder mehreren Videos einer Datenbank, die bestimmten Bedingungen entsprechen.
2.	**Objekt-Anfrage:**	Finde bestimmte Objekte in einem Segment[s,e] (Start-Frame, End-Frame) eines Videos. Als Beispiel: „Finde alle Personen, die zwischen Bild 80 und 120 im Überwachungsvideo erscheinen".
3.	**Aktivität-Anfrage:**	Finde alle Aktivitäten in einem Segment[s,e] eines Videos.
4.	**Eigenschaften-Anfrage:**	Hier müssen alle Videos und Segmente gefunden werden, die bestimmten Eigenschaften entsprechen. (Dies ist eine boolsche Kombination der ersten drei Anfragetypen).

Für bestimmte Anfrageformen ist es sinnvoll, die bestehende Datenbanksprache durch eigene Anfragetypen zu erweitern. Laut [SUB] interessieren uns im wesentlichen acht verschiedene Videofunktionen. Eine Videofunktion ist eine Funktion, die eine SQL-Anfrage möglicherweise benutzt. Die Typen der Videofunktionen sind:

1. **FindeVideoMitObjekt(o):**

   Bei der Übergabe eines Objekts o an die Funktion, erhält man ein Tripel der Form:

   (Video-id , StartBild , EndBild)

   Als Beispiel:
   FindeVideoMitObjekt(Koffer) liefert als Ergebnis: (Überwachungsvideo , 0 , 3000)
   Dies bedeutet, daß zwischen den Bildern 0 und 3000 im Überwachungsvideo ein Koffer zu sehen ist.

2. **FindeVideoMitAktivität(a):**

   Hier wird eine Aktivität übergeben und das Resultat hat die gleiche Form, wie bei 1.

3. **FindeVideoMitAktivitätUndEigenschaft(a,e,z):**

   Bei der Übergabe einer Aktivität a, einer Eigenschaft e und dessen Wert z, erhält man als Ergebnis ein Tripel der gleichen Form, wie in 1.

   Als Beispiel:
   FindeVideoMitAktivitätUndEigenschaft(Sprechen , Person , Stefan F.) liefert das Ergebnis (Überwachungsvideo , 2200 , 2500).
   Dies bedeutet, daß Stefan F. zwischen Bild 2200 und 2500 im Überwachungsvideo spricht.

4. **FindeVideoMitObjektUndEigenschaft(o,e,z):**

   Hier wird ein Objekt übergeben und das Resultat hat die gleiche Form, wie bei 3.

5. **FindeObjekteInVideo(v,s,e):**

   Als Übergabe wird ein Video v, ein StartBild s und ein EndBild e angegeben. Das Ergebnis ist eine Menge von Objekten, die in dem angegebenen Video und Segment vorkommen.

   Als Beispiel:
   FindeObjekteInVideo(Überwachungsvideo , 2500 , 3250) liefert als Ergebnis:
   { (Stefan F.) , (Jens Dope) , (Koffer) }
   Beachte, daß Jens Dope und der Koffer im Ergebnis auch erscheinen, obwohl diese Objekte ab dem Bild 3000 nicht mehr zu sehen sind.

6. **FindeAktivitätenInVideo(v,s,e):**

   Identisch zu 5., jedoch wird hier nach Aktivitäten gesucht.

7. **FindeAktivitätenUndEigenschaftenInVideo(v,s,e):**

   Bei der Übergabe eines Videos v und eines Segments [s,e] erhält man eine Menge der Form:

   Aktivitätsname:Eigenschaft1 = Instanz1;
   Eigenschaft2 = Instanz2;
   Eigenschaft3 = Instanz3;
   ...
   Eigenschaftk = Instanzk

   Dies beschreibt alle Aktivitäten und deren Eigenschaft im festgelegten Videosegment.

8. **FindeObjekteUndEigenschaftenInVideo(v,s,e):**

   Identisch zu 7., jedoch wird hier nach Objekten gesucht.

Eine einfache SQL-Anfrage hat die Form:

```
SELECT    Feld1, ... , Feldn
FROM      Relation1 (B1), Relation2 (B2), ... , Relationk (Bk)
WHERE     Bedingung
```

Zu den vorhandenen Parametern der Terme (SELECT,FROM,WHERE) werden wir einige Parameter hinzufügen:

1. SELECT enthält zusätzlich:

   video_ID : [s,e]

   welches die Auswahl eines Videos mit der Kennung video_ID und dem entsprechenden Segment [s,e] dieses Videos bezeichnet.

2. FROM enthält zusätzlich:

   video : <*quelle*><*V*>

   welches bedeutet, daß *V* eine Laufvariable über die Videos, die in *quelle* bezeichnet sind, ist.

3. WHERE enthält zusätzlich:

   Ausdruck IN *Funktionsaufruf*

   wobei
   (a) *Ausdruck* entweder eine Variable, ein Objekt, eine Aktivität oder eine Eigenschaft ist.
   (b) *Funktionsaufruf* ist eine der acht Videofunktionen

Nun können wir einige Anfragen an die Video-Datenbank stellen. Betrachten wir zunächst die Anfrage: „Finde alle Videos und relevanten Segmente aus der Videobibliothek vidbib in denen Jens Dope zu sehen ist". Eine entsprechende Anfrage mit Hilfe von SQL sähe wie folgt aus:

```
SELECT    vid:[s,e]
FROM      video:vidbib
WHERE     (vid,s,e) IN FindeVideoMitObjekt('Jens Dope')
```

Betrachten wir ein weiteres Beispiel. Die Anfrage „Finde alle Videos und relevanten Segmente aus der Videobibliothek vidbib, in denen Jens Dope und Stefan F. zu sehen sind" zeigt folgende SQL-Anfrage:

```
SELECT    vid:[s,e]
FROM      video:vidbib
WHERE     (vid,s,e) IN FindeVideoMitObjekt('Jens Dope') AND
          (vid,s,e) IN FindeVideoMitObjekt('Stefan F.')
```

Ein komplexeres Beispiel ist: „Finde alle Videos und relevanten Segmente aus der Videobibliothek vidbib in denen Jens Dope einen Koffer von Stefan F. erhält". Diese Anfrage kann folgendermaßen ausgedrückt werden:

```
SELECT    vid:[s,e]
FROM      video:vidbib
WHERE     (vid,s,e) IN FindeVideoMitObjekt('Jens Dope') AND
          (vid,s,e) IN FindeVideoMitObjekt('Stefan F.') AND
          (vid,s,e) IN FindeVideoMitAktivitätUndEigenschaft
                    (Austausch,Objekt,Koffer) AND
          (vid,s,e) IN FindeVideoMitAktivitätUndEigenschaft
                    (Austausch,Empfänger,'Jens Dope') AND
          (vid,s,e) IN FindeVideoMitAktivitätUndEigenschaft
                    (Austausch,Geber,'Stefan F.')
```

Eine weitere Anfrage könnte sein: „Finde in allen Videos die Personen, die mit Jens Dope zusammen gesehen wurden". Diese Anfrage muß abweichend vom obigen Schema gestellt werden.

```
SELECT    vid:[s,e], Objekt
FROM      video:vidbib
WHERE     (vid,s,e) IN FindeVideoMitObjekt('Jens Dope') AND
          Objekt IN FindeObjekteInVideo(vid,s,e) AND
          Objekt ≠ 'Jens Dope' AND
          type of(Objekt,Person)
```

Interessant an SQL ist, daß SQL ganz einfach erweitert werden kann, um solche Anfragen an eine Videodatenbank zu erledigen. Würde man nur nach den Objekten in obigem Beispiel fragen, so würde man einfach das SELECT-Statement in folgender Weise abändern:

```
SELECT Objekt
```

Diese Anfrage liefert alle Personen, die mit Jens Dope zu sehen sind.

Die bis jetzt vorgestellten Speicherungsmöglichkeiten stoßen schnell an ihre Grenzen. Betrachtet man zum Beispiel ein 90 Minuten langes Video, dann beinhaltet dieses 135.000 Einzelbilder (pro Sekunde werden 25 Bilder benötigt (PAL)). Diese Datenmenge ist vor allem dann zu groß, wenn mehrere Videos in einer Datenbank gespeichert werden sollen.
Dies führt uns zu den *„Frame Segment Trees"* , der dieses Problem weitgehend löst.

2.4 Frame Segment Trees

Betrachten wir das Überwachungsvideo. Offensichtlich braucht Stefan F. länger als drei Bilder um über den Gehweg zu laufen. Nehmen wir an, daß er zehn Sekunden für seinen Weg benötigt, dann bedeutet dies, daß 250 Bilder für diese Aktivität gebraucht werden, also auch 250 Einträge in der Datenbank auftauchen. Dies ist nicht effektiv. Besser wäre es, die Daten abschnittsweise zu speichern.
Überlegen wir uns für unsere Überwachungsvideo, welches 180 Sekunden lang ist (4500 Bilder), das Auftreten einiger Objekte und deren Dauer. In diesem Video haben wir fünf interessante Objekte, die indexiert werden sollen.
Eine folgende Tabelle soll das Auftreten aller Objekte zeigen:

ID	Objekt	Bildsequenz
1	Jens Dope	0 - 125
1	Jens Dope	250 - 3000
2	Stefan F.	0 - 4000
3	Koffer	0 - 3000
4	unbekanntes Auto	500 - 625

ID	Objekt	Bildsequenz
4	unbekanntes Auto	1000 - 1125
4	unbekanntes Auto	1750 - 1875
4	unbekanntes Auto	3000 - 3250
5	Passant	1500 - 2000
5	Passant	2750 - 3250

Dies sind insgesamt 11500 Bilder, die in der alten Darstellung gespeichert werden müssen. In der folgenden Darstellung (Abb. 3) müssen nur 10 Einträge vorgenommen werden.

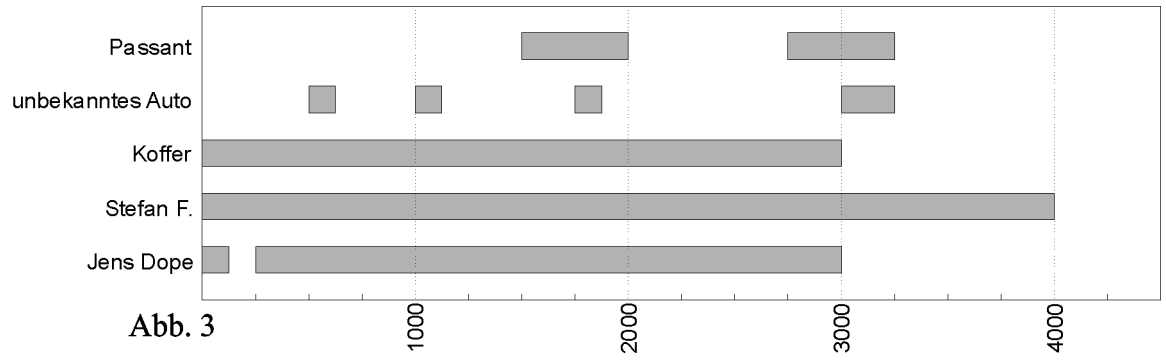


Abb. 3

Trotzdem kann auch diese Darstellung in der Praxis sehr schnell unhandlich werden. Diese Struktur eignet sich nicht um Anfragen zu beantworten, die Segmente manipulieren, wie dies zum Beispiel bei der temporalen Logik der Fall ist. Hier kommt der *Frame Segment Tree* zum Einsatz. Der Grundgedanke ist einfach.
Wir bauen einen binären Baum nach folgenden Regeln auf:

1. Jeder Knoten im Baum repräsentiert ein Segment der gesamten Sequenz.
2. Jedes Blatt befindet sich auf der untersten Ebene. Das linke Blatt bezeichnet die Sequenz $[s_1,s_2)$, das nächste die Sequenz $[s_2,s_3)$, usw. Wenn N ein Knoten mit zwei Kindknoten und den Intervallen $[p_3,p_4)$,$[p_4,p_5)$ ist, dann bezeichnet der Knoten N das Intervall $[p_3,p_5)$.
3. Jede Zahl in den Knoten kann als Adresse angesehen werden.
4. Die Zahl neben den Knoten bezeichnet die eindeutige Nummer der Objekte.

Für die Aktivitäten wird ein zweiter Baum nach gleichem Schema wie oben angelegt.

Die Konstruktion erfolgt in zwei Schritten. Als Erstes müssen von allen Segmenten die höchsten und niedrigsten Bildnummern gefunden werden. Diese Werte werden sortiert und Duplikate entfernt. Diese Liste sähe im Fall des Überwachungsvideos folgendermaßen aus:

0,125,250,500,625,1000,1125,1500,1750,1875,2000,2750,3000,3125,3250,4000

Diese Liste hat 16 Elemente, welche uns zu folgendem *Frame Segment Tree* (Abb. 4) führt. Dieser Baum hat neben jedem Knoten die Bildsequenz stehen:

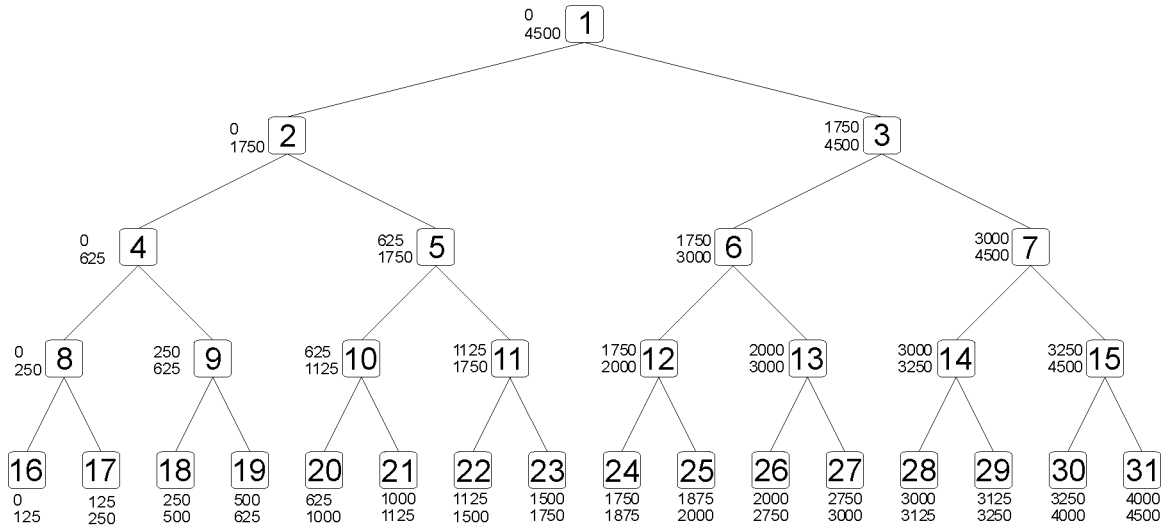


Abb. 4

Im folgenden *Frame Segment Tree* (Abb. 5) unseres Beispiel stehen die Objekte neben den Knoten, wenn diese in dem entsprechenden Intervall auftreten. Beachte, daß dies der selbe *Frame Segment Tree* ist.

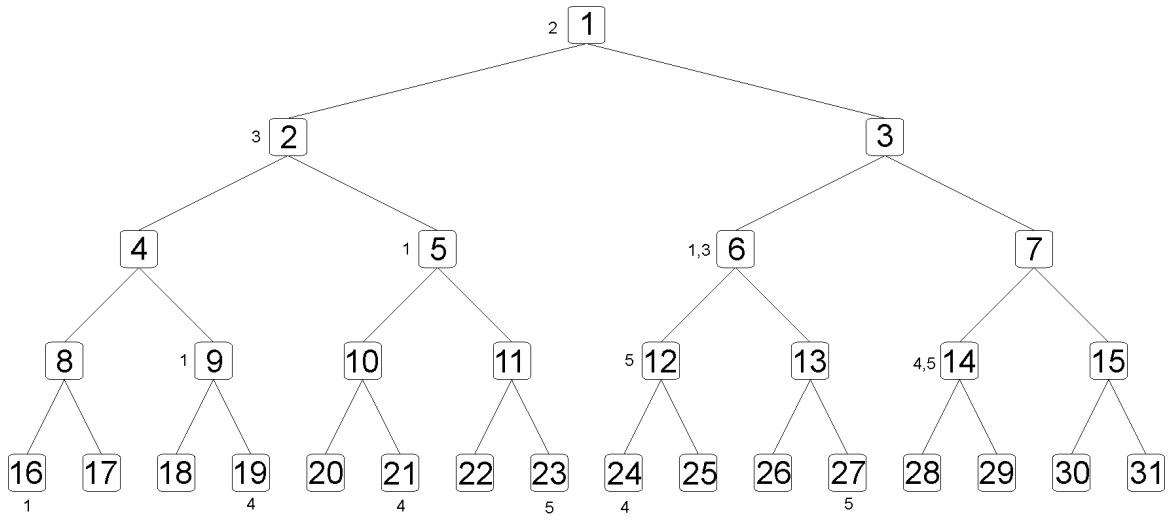


Abb. 5

Das Einfügen, Löschen, Ändern und Suchen in einem binären Baum werden als bekannt vorausgesetzt und hier nicht weiter besprochen.

Es gibt zwei Möglichkeiten, die Daten im Frame Segment Tree zu speichern. Die erste Methode wäre die Daten abhängig vom jeweiligen Knoten zu speichern. Dann hätten wir eine Liste in folgender Form. Abbildung 6 zeigt den Inhalt des Knoten 6.

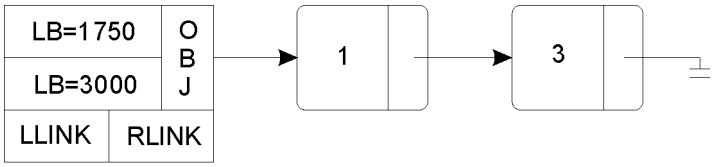


Abb. 6

Oder man speichert die Daten abhängig von den Objekten, dann sähe die Liste wie folgt aus. Abbildung 7 zeigt den Inhalt der Objekte ausschnittsweise für Objekt 1.

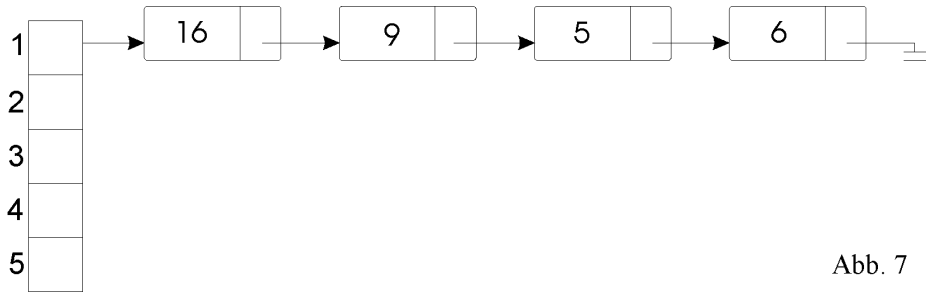


Abb. 7

Abhängig von der Art der Anfrage muß man sich im Klaren darüber sein, welche Speicherungsmethode am geeignetsten ist. Möchte man hauptsächlich von der Benutzerseite Objekte in Videos suchen, so eignet sich die Darstellung aus Abbildung 7, jedoch ist die Speicherungsmethode nach Abbildung 6 geeigneter für Anfragen, die Objekte in bestimmten Segmenten suchen.
Der RS-Baum baut auf dem Konzept des R-Baums auf. Mehrere Sequenzen werden in Gruppen zusammengefasst und daraus ein Baum aufgebaut. Diese Baumart eignet sich für Festplattenanfragen, da hier nicht eine Sequenz zurückgeliefert wird, sondern ein Menge (Gruppe) von Segmenten.

2.4 Automatische Indexierung

Jeder Film kann in mehrere Teile unterteilt werden. Jede Sequenz hat mehrere Szenen, jede Szene mehrere Shots und jeder Shot mehrere Bilder (Frames). Um nun einen automatischen Indexierungsprozeß zu implementieren, muß man die drei „Hauptschnitte" kennen.

(a) Shot concatenation:	Dies ist die einfachste Methode zwei Shots zusammenzufügen. Die Shots werden einfach „aneinandergeklebt".
(b) Spatial composition:	Dies ist die häufigste Methode zwei Shots zusammenzufügen. Die Shots werden ineinander übergeblendet.
(c) Chromatic composition:	Hier gibt es zwei Methoden. Einmal das Ausblenden und natürlich das Einblenden einer Szene.

Alle weiteren Schnittmethoden sind Abwandlungen der oben Aufgeführten.

Die Schnitte nach Methode (a) können leicht entdeckt werden, da sich hier die kompletten Bildmerkmale von einem auf das andere Bild drastisch ändern. Es wird in diesem Fall ein pixelweiser Vergleich vorgenommen. Bei Szenen, die eine schnelle Kamerabewegung beinhalten ist es problematisch einen Schnitt zu erkennen, da sich hier die Bildinformation auch drastisch ändert. Um in diesem Falle keine falschen Schnitte zu erkennen, muß ein Schwellwert eingeführt werden. Dieser Schwellwert überprüft zum Beispiel die Helligkeitsverteilung. Die zugrunde liegende Annahme ist: die Helligkeitsänderung zwischen Bildern derselben Sequenz ist klein, jene zwischen Bildern unterschiedlicher Sequenzen in der Regel groß.
Bei Schnitten nach Methode (b) oder (c) hilft ein einfacher Schwellwert zur Findung eines Schnittes nicht, da die Änderungen zwischen den Bildern sehr gering sind. Dies führt zu einer einfachen Idee:
Es wird zum ersten Schnitterkennungsschwellwert ein zweiter Schwellwert eingeführt. Dieser überprüft anhand eines Referenzbildes die folgenden Bilder und bestimmt die Differenz. Falls die Differenz über den Schwellwert steigt, so hat ein „weicher" Schnitt stattgefunden. Falls die Differenzen für längere Zeit unterhalb des Schwellwerts bleiben, war das Referenzbild nicht der Anfang eines Übergangs und das Verfahren wird an entsprechender Stelle fortgesetzt.
Eine andere Möglichkeit Schnitte nach Methode (b) oder (c) zu erkennen, ist ein Helligkeitshistogramm zu erstellen. Dieses Histogramm eignet sich fade-in, bzw. fade-out Schnitte zu finden. Wenn das Helligkeitshistogramm gestaucht wird, eine Zeit lang nur dunkel enthält und wieder gestreckt wird, so hat ein Schnitt stattgefunden.

Die automatischen Prozesse können auf der Bildebene die einzelnen Bilder nach Merkmalen durchsuchen (Umrisse, Farben, Personen, etc.) und diese in eine Datenbank eintragen.

Die automatischen Prozesse können den Einsatz des Menschen nicht ersetzen, da viele wichtige Daten, wie z.B. Personennamen, nicht erkannt werden. Die Prozesse können aber einen großen Teil der Arbeit übernehmen.

3 Audio-Retrieval

Audio-Retrieval kann grob in vier Kategorien eingeteilt werden:

- Suche auf Metadaten
- Suche mit akustischen Merkmalen
- Suche mit Noten, resp. Tonintervallen
- Suche in gesprochenem Text

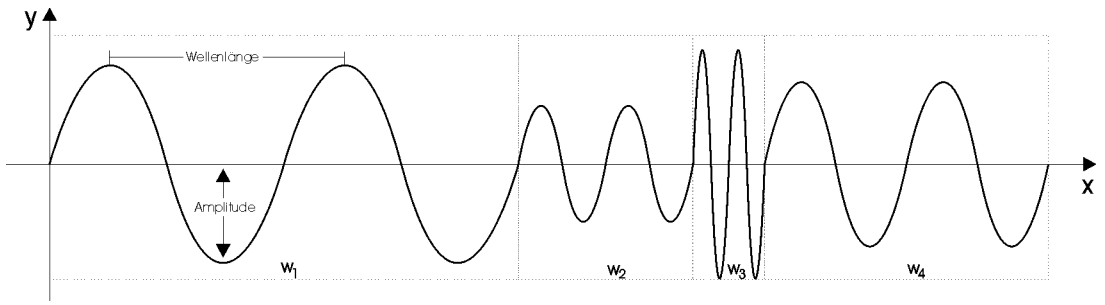
3.1 Suche auf Metadaten

Die Suche auf Metadaten erfolgt auf der Text-Retrieval-Basis, da hier nur über die textuelle Zusatzinformation gesucht wird. Die Audio-Daten werden nach dem gleichen Schema indexiert und gespeichert, wie das bei den Video-Daten geschehen ist. Jede Aktivität besitzt eine Menge von zugehörigen Daten. Als Beispiel sei eine Oper (Lohengrin) aufgeführt:

- *Sänger:* Dies könnte eine Menge mit den Inhalten *Rolle, Tonlage* und *Person* sein. (Lohengrin, Tenor, Rene Kollo)
- *Verweis:* Dies könnte ein Verweis auf weiterführende Literatur zu der gerade gehörten Szene sein.
- *Inhalt:* Dies könnte ein Verweis auf den Text der gerade gehörten Arie sein.

3.2 Suche mit akustischen Merkmalen

Das Erstellen einer Audio-Datenbank ist sehr einfach, wenn man das jeweilige Objekt oder die Aktivität benennen kann. Schwieriger wird dies, wenn zum Beispiel ein Überwachungsband eines Telefonats in eine Datenbank aufgenommen werden soll. Der Partner an der anderen Seite der Telefonleitung bleibt vielleicht unbekannt.
In solchen Fällen muß man auf die Rohdaten zurückgreifen. Hier wird das Signal indexiert, signifikante Eigenschaften extrahiert und gespeichert.



(Abb. 8) Wellenform eines Audiosignals

Dazu muß man den Aufbau eines Audiosignals kennen. Wir betrachten Abbildung 8.

- Wellenlänge: w
- Frequenz: $f = \frac{1}{T}$ T ist die Zeit, die ein Signal entlang einer Wellenlänge braucht.
- Geschwindigkeit: $v = \frac{w}{T} = w \cdot f$

Das Signal in einzelne Zeitbereiche zu unterteilen, ist die beste Möglichkeit Audiodaten zu indexieren. In der Regel findet man aber in solchen Fenstern (window $w_1,...,w_n$) keine homogenen Signale. Wenn die Breite der Fenster jedoch klein gewählt wird, kann man diese als homogen annehmen.

Das Indexieren von Audiodaten kann nach folgendem Schema erfolgen:

1. *Fensterbildung:* Teile das Signal in kleine Bereiche auf. Dies kann a priori geschehen, in dem der Benutzer eine bestimmte Breite w vorgibt. Die andere Möglichkeit ist, die Bereiche nach einem Schema, wie bei den Videodaten, aufzubauen. Hier wird automatisch nach bestimmten Merkmalen gesucht und dort ein Schnitt vorgenommen.

2. *Inhaltsextraktion:* Danach wird jedes Fenster einzeln behandelt und die interessanten Aspekte extrahiert. Die bekanntesten Merkmale sind:

(a) *Intensität:* Die Intensität einer Welle ist definiert als
$I = 2 \cdot \pi^2 \cdot f^2 \cdot \mu \cdot a^2 \cdot v$
wobei gilt: f ist die Frequenz, μ ist die Dichte des Materials, a ist die Amplitude und v die Geschwindigkeit. Die Intensität hat die Einheit $\frac{W}{m^2}$

(b) *Lautstärke:* Wellen mit einer höheren Intensität werden vom menschlichen Ohr lauter empfunden, als Wellen mit einer niedrigeren Intensität. Die Lautstärke steigt nicht linear mit der Intensität. Beachte, daß zwei Wellen außerhalb der hörbaren Frequenz die Lautstärke Null haben, obwohl die Eine einen signifikanten Unterschied zur Anderen hat. Nehmen wir an L_0 bezeichnet die unterste Frequenz die ein Mensch hören kann, (ca. 15Hz) dann ist die Lautstärke definiert als:
$L = 10 \cdot \log\left(\frac{I}{L_0}\right)$

(c) *Tonhöhe:* Die Tonhöhe p(f,a) wird berechnet aus der Amplitude a und der Frequenz f.

(d) *Helligkeit:* Die Helligkeit ist ein Maß, wie klar der Ton zu hören ist und beschreibt das Auftreten hoher Frequenzen im Signal.

Es existieren weitere Merkmale, wie die Varianz, Korrelation, usw.. Im allgemeinen kann gesagt werden, daß ein Audiosignal aus einer Menge von K-Tupeln, welche die Eigenschaften enthalten, besteht.

Suchanfragen auf akustischen Merkmalen könnten lauten: „Finde alle Sequenzen des Liedes X, die der Mensch nicht hören kann". Dies könnte für Tierliebhaber interessant sein, die testen wollen, ob ihren Musik einen besonderen Reiz auf ihre Tiere ausübt. Hierbei würde der Wert der Lautstärke einen wesentlichen Einfluß auf das Suchergebnis ausüben.
Eine andere Anfrage könnte lauten: „Finde alle Lieder, die unter Wasser aufgenommen wurden". Hier kommt die Intensität zum tragen, da hier die Dichte des Übertragungsmaterials eine große Rolle spielt.
Die Anfrage „Finde alle Lieder, bei denen hauptsächlich Geige gespielt wird" fragt nach der Helligkeit. Lieder, die viele helle, hohe Töne enthalten führen zu einem Ergebnis.

3.3 Suche mit Noten

Bei der Suche in Musikstücken machen akustische Merkmale wenig Sinn. Es sollte vielmehr die Ähnlichkeit über Takt, Tempo oder Noten definiert werden. Bei der Suche gilt es jedoch zwei Probleme zu lösen:

- **Erkennung der Noten aus dem Audiosignal:** Die größten Probleme liegen an der Vielzahl von Instrumenten, der Überlagerung und dem Gesang. Die meisten Ansätze verwenden deshalb Audiodaten im MIDI-Format.

- **Definition von Melodie:** Eine Melodie ist grundsätzlich über eine Folge von Noten definiert. Jedoch kann die selbe Melodie in verschiedenen Tonhöhen abgespielt werden, daher werden nicht die einzelnen Noten gespeichert, sondern die Änderung von einer auf die andere Note.

Bei letzterem Ansatz (Search by Humming) werden die Lieder als Folge von „D"(Down), „U"(Up) und „S"(Same) gespeichert. Die Anfrage kann entweder „gesummt" werden, da die Umwandlung in Noten in diesem Fall recht einfach ist. Heraus kommt eine Folge von „S", „U" und „D".
Die andere Methode ist es, die Noten direkt einzugeben.

3.4 Suche in gesprochenem Text

Das Kapitel „Spracherkennung" ist ein weites Feld. Hier wird nur grob gezeigt, wie der aktuelle Forschungsstand aussieht.
Als erster Schritt muß das Audio-Signal von allen uninteressanten Merkmalen befreit werden. Die Tonhöhe, das Tempo und die Lautstärke sollten keine Rolle, bei der Erkennung von gesprochenem Text, spielen. Zu diesem Zweck wird eine MFCC (mel frequency central coefficient) Analyse durchgeführt. Phoneme können mittels eines HMM (Hidden Markov Modell) basierend auf den quantisierten MFCC erkannt werden.
Ein HMM beschreibt einen stochastischen Prozeß, der aus mehreren Schritten besteht. Beim Erreichen eine Zustands s wird ein Ausgabesymbol k mit einer bestimmten Wahrscheinlichkeit erzeugt.

Für jedes zu erkennende Phonem wird ein solches Modell erstellt. Bei der Erkennung ermittelt man das Phonemmodell, welches den Ausgabestrom erzeugt haben könnte. Das Audiosignal wird letztlich in einen Strom von Phonemen umgewandelt.
Die Suche kann nun auf dem Phonemen erfolgen, oder man versucht die Phoneme in Wörter umzuwandeln, bzw. zu vereinen. Wortgrenzen zu finden ist dabei schwierig. Beim Speichern in die Datenbank muß beachtet werden, daß die erkannten Phoneme eventuell falsch erkannt wurden. Entweder wurden die Wörter falsch erkannt, zwei nacheinander folgende Wörter als ein Wort interpretiert, oder ein Wort wurde überhaupt nicht erkannt. Die ersten beiden Fehler kann ein kontrolliertes Wörterbuch größtenteils beheben.
Die Suche auf Phonemen ist laut [Schäuble] besser, da aktuelle Worterkennungssysteme nur höchstens 64.000 Wörter beinhalten und die deutsche Sprache aufgrund von Wortzusammensetzungen und unterschiedlich gesprochenen Formen für die Suche auf Wörtern nicht geeignet ist. Die Anzahl der zu trainierenden Worte wäre zu groß. Außerdem können Eigennamen, Firmen oder Länder kaum erkannt werden.
Die Anfrage sollte auf folgende zwei Arten möglich sein. Der zu suchende Text wird entweder gesprochen, oder eingetippt. Bei beiden Anfrageformen muß jeweils eine Anfrage in den anderen Anfragetyp umgewandelt werden. Spricht man den zu suchenden Text ein und die Datenbank enthält aber nur Wörter, so muß der gesprochene Text in einzelne Wörter umgewandelt werden. Dies gilt auch für Phoneme, wenn der Text getippt wird.

3.5 Die diskreten Transformationen

Ein 10-minütiges Audiosignal kann mehr als 100.000 Fenster besitzen. Wie bei den Videodaten versucht man auch hier die Daten zu komprimieren, um im Endeffekt weniger Fenster speichern zu müssen. Dies wird durch die diskrete Fourier Transformation (DFT), oder der diskreten Cosinus Transformation (DCT) erreicht.
Bei der DCT [DCT] werden ähnlich, wie bei der DFT die Ausgangswerte vom Zeit- in den Frequenzbereich umgewandelt. In diesem Frequenzbereich werden anhand komplizierter mathematischer Formeln uninteressante Merkmale entfernt oder weniger stark betrachtet. Die Komprimierung nach dem MP3-Standard [TUB] benutzt mehrere Methoden. Bei der Maskierung wird erkannt, daß ein lautes Signal ein Leises überdeckt und somit nicht mehr mitgespeichert wird. Beim Sub-Band Coding werden, auf Grund der nicht konstanten Bitrate, einzelne unwichtige Signale (z.B.: monoton lauter Bass und leise Höhen) nur mit einer geringen Bitrate übertragen. Das Stereo-Signal unterscheidet sich nur unwesentlich von einem Mono-Signal. Unterschiede sind nur zu hören, weil bei der Aufnahme zum Beispiel das Schlagzeug links und die Gitarre rechts stand. Bei MP3 sind der rechte und der linke Kanal gleich. Bei der Kompression werden die Frequenzbänder des Schlagzeugs links relativ lauter, die Frequenzbänder der Gitarre rechts relativ lauter zur Originallautstärke gespeichert. Die Datenmenge wird somit wesentlich kleiner (Kompressionrate 1:11).
Die Suche auf MP3-Daten unterscheidet sich in keiner Weise von den vorangegangen Suchmechanismen. Es muß lediglich vorher eine Dekomprimierung stattfinden. Die Suche nach unhörbaren Tönen könnte aber zu keinem Ergebnis, da diese Töne eventuell eliminiert wurden.

4 Zusammenfassung und Ausblick

In diesem Seminar wurde gezeigt, wie man den Inhalt eines Videos extrahiert und Schritt für Schritt die Daten in eine geeignete Struktur speichert werden, um so differenzierte Suchanfragen stellen zu können. Das Ergebnis war der Frame Segment Tree, der eine Suchanfrage einfach behandeln kann und bei dem die Datenmenge überschaubar bleibt.
Bei den Audiodaten verhielt es sich nicht anders. Die Erkennung von Audiomerkmalen stand im Vordergrund. Je nach Anwendungsgebiet konnte man die Inhalte eines Audiosignals unterschiedlich extrahieren und speichern. Zu nennen sei hier die Suche auf Metadaten, die Suche auf Basis der Noten, die Suche auf akustischen Merkmalen und die Suche auf gesprochenem Text.

Die Situation bei der Suche nach Video-, oder Audiodaten wird sich in den nächsten Jahren nicht wesentlich bessern. Die täglich neu anfallenden Daten können nicht in der gleichen Zeit in der sie anfallen, indexiert werden.
Abhilfe scheint für Videodaten der Dublin-Core [DUB] zu schaffen. Der erste Metadaten Workshop wurde im März 1995 vom Online Computer Library Center (OCLC) und dem National Center for Supercomputing (NCSA) in Dublin (Ohio) abgehalten und gibt seitdem der Dublin Core Metadaten Initiative den Namen. In dieser Konferenz wurde ein Kernsatz (Core) von 15 Elementen erarbeitet. Die Inhalte der 15 Elemente werden vom Autor selbst vergeben. Die Dublin Core Metadaten dienen dazu elektronische Textobjekte („document-like objects" DLOs) im WWW schneller und leichter zu finden. Viele Ansätze erweitern die Dublin Core, um diese auch für Video-, bzw. Audiodaten zu verwenden.

Ein Ansatz für Audiodaten ist MPEG-7. Dies ist jedoch kein neues Kompressionsverfahren. Vielmehr sind die Ziele von MPEG-7 wie folgt definiert:

- Beschreibung multimedialer Information aller Art
- Beschreibung der möglichen Merkmale und deren Beziehung zueinander.
- Eine Sprache für die Definition der Merkmale
- Eine oder mehrere Möglichkeiten, Merkmale zu kodieren und effizient zu durchsuchen.

Das Frauenhofer Institut für integrierte Schaltungen [FRA] entwickelt gerade einen „Metadaten-Steckbrief für Multimediale Inhalte". Die ersten Entwicklungen in diesem Gebiet beruhen auf der Äquivalenzprüfung von Audioinhalten. Eingegebene Daten - auch über ein Mikrofon - werden mit den Inhalten einer Datenbank auf Ähnlichkeiten überprüft. Markante Merkmale, wie Rhythmus, Takt, Tempo, Geschlecht des Sängers, Art und Menge der Musikinstrumente und Art der Melodieführung mit den zugehörigen Harmonien, werden verglichen. Diese Charakteristika werden gespeichert damit jede Datei ihren eigenen „Steckbrief" erhält.
Die Suche auf Video-, oder Audiodaten wird weiterhin schwer bleiben. Die Standards für das Internet werden wahrscheinlich durch MPEG-7 erweitert, jedoch wird auch hier eine in die Tiefe gehende Anfrage zu keinem befriedigenden Ergebnis führen.

5. Literaturangabe

[CYV] http://www.cyveillance.com

[DCT] Die diskrete Cosinustransformation (DCT), Joachim Schwarz und Guido Soermann

[DUB] dublin core: die renaissance der bibliothekare http://www.informatik.hu-berlin.de/~mayr/d_c.htm

[ETH] 37-342 Multimedia Retrieval, ETH, Eidgenössische Technische Hochschule, Zürich, Dozent: Dr. Roger Weber, 2001

[FRA] http://www.emt.iis.fhg.de/metadaten_mehr.html

[SCH] Peter Schäuble, Multimedia Information Retrieval, Content-based Information Retrieval from Large Text and Audio Databases, Kluwer Academic Publishers, Zurich, Switzeland, 1997

[SUB] Principles of Multimedia Database Systems, V.S. Subrahmanian Morgan Kaufman Publisher, San Francisco, California, 1998

[TUB] http://www.awb.tu-berlin.de/Studium/ Lehrunterlagen_und_Pruefungsleistungen/ Daten-_und_Info-Verarbeitung/3._Infouebertragung/ 3.3_Datenformate/_uebersicht.html

Abbildungsnachweis

[Abb. 1] http://www.searchenginewatch.com

[Abb. 2], [Abb. 6], [Abb. 7] Principles of Multimedia Database Systems, V.S. Subrahmanian, Morgan Kaufman Publisher, San Francisco, California,