

Visualisierung der Inhalte von Videos
mit Hilfe von Video-Indexing

Diplomarbeit
zur Erlangung des akademischen Grades
DIPLOM-INGENIEUR
in der Studienrichtung Informatik

Eingereicht von
Simon G. Vogl

28. April 1998

Angefertigt am Institut für techn. Informatik und Telematik
Abteilung Telekooperation
Johannes Kepler Universität Linz
Eingereicht bei: Univ.-Prof. Max Mühlhäuser

Eidesstattliche Erklärung

Ich erkläre an Eides statt, daß ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfaßt, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Linz, am 28. April 1998

(Simon G. Vogl)

Zusammenfassung

Je größer und leistungsfähiger die Informationssysteme werden, werden umso komplexere und umfangreichere Datenquellen zugänglich. Dabei wird das Sichten dieser Materialien zu einem nichttrivialen Problem ob der schiereren Menge an angebotenen Daten. Besonders deutlich wird diese Problematik am Beispiel digitaler Videos, bei denen klassische textuelle Suchverfahren versagen. Hier ist es erforderlich, auf neuen Wegen eine Repräsentation des Inhalts zu bieten, um eine schnelle Einsicht in diese Datenbestände gewähren zu können.

In der vorliegenden Arbeit wird ein System zur Visualisierung der Inhalte von Videos vorgestellt, das mit Hilfe von Video-Indexing den Inhalt von Videos bewertet und eine hierarchische Zusammenfassung generiert. In einer neuartigen Benutzerschnittstelle werden die extrahierten Schlüsselbilder dem Benutzer in einer dreidimensionalen Szene derart präsentiert, daß sowohl der temporale wie auch der räumliche Zusammenhang zwischen aufeinanderfolgenden Bildern gewahrt bleibt. Diese Art der Darstellung ermöglicht einen schnellen Eindruck des Video-Inhalts.

Das System wurde in Java implementiert und benutzt ein VRML-Plugin zur Darstellung der Hierarchien in einer 3D-Umgebung. Es wurden sowohl ein Editor zum lokalen Bearbeiten der Hierarchien implementiert, als auch ein Viewer, der eine Einbindung in das WWW ermöglicht.

Abstract

The rapidly growing amounts of data that is available to the individual user makes the process of reviewing these sources of information extremely time-consuming. This problem is evident in the case of digital video, where classical textual search principles cannot be applied. Therefore new forms of representing video data have to be developed to offer an effective way of browsing its contents.

This paper describes a system to visualize the contents of digital videos. Video indexing techniques are used to rank the frames of a video and generate a hierarchical representation. The extracted key frames are displayed in a new kind of user interface that incorporates a three-dimensional arrangement of the frames to maintain the temporal as well as the spatial order of the visualized data.

The system has been implemented using Java in conjunction with a VRML-plugin which is used to visualize the data in a 3D scene. The system consists of an editor to generate and manipulate those hierarchies, and a viewer for use in HTML-pages on the Web.

Inhalt

1	Einleitung	9
1.1	Aufgabenstellung	10
1.2	Ergebnis der Arbeit	11
1.3	Stand der Forschung	11
1.4	Gliederung der Arbeit	12
2	Video Indexing	13
2.1	Struktur von Videos	14
2.2	Farbräume	15
2.2.1	YUV	16
2.2.2	OPP Farbraum	16
2.3	Videoformate	16
2.3.1	AVI	17
2.3.2	MPEG	17
2.4	Merkmalsextraktion und Bewertung	18
2.4.1	Punktdifferenz	19
2.4.2	Histogramm-basierte Algorithmen	19
2.4.3	Block-orientierte Methoden	21
2.4.4	Farbkorrelogramme	21

2.5	Schnitterkennung	22
2.5.1	Mittelwert-Gruppierung	23
2.5.2	Doppelvergleich	23
2.5.3	Schnitterkennung in MPEG-Filmen	24
2.5.4	Schlüsselbilder	25
2.6	Kamera- und Objektbewegung	25
2.6.1	Kamerabewegung	26
2.6.2	Objektextraktion	27
3	Existierende Systeme	29
3.1	Video Slide Show Interface	30
3.2	VAbstract: Generierung von Film-Trailern	31
3.3	WebSEEk	32
3.4	PanoramaExcerpts	33
3.5	VideoSTAR	34
3.6	SWIM	35
3.7	CVEPS, WebClip	36
3.8	Scene Transition Graphs	36
3.9	VideoSpaceIcon, VideoMap	37
4	Implementierung	39
4.1	Daten- und Dateistrukturen	40
4.1.1	Generierung eines OBVIs	41
4.2	OBVI-Editor	45
4.2.1	Benutzerschnittstelle	45
4.2.2	Architektur des Editors	50

4.3	OBVI-Viewer	52
4.3.1	Hierarchiegenerierung	54
4.3.2	Train mode	54
4.3.3	Datenbank-Schnittstelle	54
5	Zusammenfassung und Ausblick	57
5.1	Bewertung	57
5.1.1	Laufzeiteffizienz	57
5.1.2	Portabilität	57
5.1.3	Anwendungsbereiche	58
5.2	Ausblick	58
A	Dateiformate	61
A.1	Allgemeine Konventionen	61
A.1.1	Definitionen	62
A.2	Movie-Descriptor	62
A.3	Tags: Annotationen	63
A.4	Objekt-Dateien	63
A.5	OBVIs	63
B	URL-Verzeichnis	65
B.1	Video-Indexing Projekte	65
B.2	Forschungsgruppen	66
B.3	Sonstiges	66
C	Aufnahmen der Benutzerschnittstelle	67
D	Literatur	71

Kapitel 1

Einleitung

Digitale Videos bekommen eine immer größere Verbreitung, zum Beispiel als Komponenten in Multimedia-Präsentationen. Sie sind aber auch immer stärker im Internet als Quelle von Information zu finden. Diese Entwicklung, deren Ende noch nicht zu erahnen ist, wurde maßgeblich durch die Vermarktung immer schnellerer Computer mit immer größeren Festplatten getragen.

Während Videos bisher meist digitalisiert werden mußten, um sie auf Computern weiterbearbeiten zu können, befindet sich die digitale Videotechnik von der Kamera bis zum Schnittplatz gerade im Stadium der Markteinführung.

Dies schlägt sich unter anderem in der Tatsache nieder, daß immer mehr Informationsanbieter wie Nachrichtenagenturen ihre technischen Einrichtungen auf Digitalverarbeitung umstellen (unter dem Schlagwort „going digital“).

Diese Entwicklung beschränkt sich allerdings nicht allein auf den professionellen Sektor, diese Technik wird im Moment auch für den Heimanwender erschwinglich. Dies wird beispielsweise am Auftauchen von digitalen Camcordern augenscheinlich, die von einer ganzen Reihe von Firmen auf den Markt gebracht werden. Erweiterungskarten für PCs ermöglichen den direkten Zugriff auf die so aufgezeichneten Informationen, die in Zukunft nur noch rein digital verarbeitet werden.

Um die gigantischen Datenmengen, die bei der digitalen Videobearbeitung anfallen, bewältigen zu können, ist es notwendig, den Benutzer durch geeignete Werkzeuge bei seinen Aufgaben zu unterstützen.

Szenario 1: Nachrichtenagenturen haben das Problem, täglich Stunden von ungeschnittenem Videomaterial sichten zu müssen. Zwar ist bekannt, welches Ereignis in den einzelnen Stücken festgehalten wurde, ob jedoch eine besonders spektakuläre Einstellung darunter ist, läßt sich aus dem Titel nicht entnehmen. Hier ist also nicht so sehr ein Überblick über den gesamten Inhalt des Films gefragt, sondern das Interesse gilt vielmehr der Suche nach bestimmten Details (d.h. Bildinhalten).

In diesem Fall kann man mit Hilfe von Video Indexing eine Bewertung der Videostücke vornehmen und die interessanteren Stücke dem Betrachter zuerst vorlegen. Dadurch bleibt es diesem erspart, das komplette Stück zu sichten.

Szenario 2: Otto Heim, der an das Kabelnetz von *Digital Interactive* angeschlossen ist, sucht sich für einen unterhaltsamen Abend einen Film mit seinem Lieblingsschauspieler aus. Auf der Homepage der Firma, die auf Knopfdruck am Fernseher erscheint, läßt er sich vom Server alle Filme aus der Datenbank auflisten, die seinen Wünschen entsprechen.

In dieser Liste findet Otto ein paar Filme, die ihm bekannt vorkommen, kann sich aber nicht mehr genau erinnern, ob genau die Szene, die sein Zwerchfell so strapaziert nun in „Spiel mir das Lied vom Jod“ oder in „Krankenschwestern beißen nicht“ vorkommt. Also wirft er einen Blick in die visuelle Übersicht der Filme und weiß in Kürze, daß ihm ein paar vergnügliche Stunden bevorstehen. . .

Benutzer werden meist von einem dieser zwei Ziele geleitet, welche Ding et al. [DMT97] als *gist determination* und *object identification* (Erfassung des wesentlichen Inhalts, Objektidentifikation) bezeichnen. Im ersten Fall soll möglichst schnell ein Überblick über das Filmgeschehen gewonnen werden, eine Tätigkeit die im allgemeinen mit „Browsing“ bezeichnet wird. Sie ist eine nicht zielgerichtete Suche, und läßt sich wohl am besten mit dem Ausdruck „Sichten von Daten“ übersetzen. Im Gegensatz dazu stellt die Objektidentifikation eine zielgerichtete Suche dar, bei der das Objekt des Interesses bereits feststeht.

Im allgemeinen dient das Browsing als Filteroperation bei der Suche nach bestimmten Informationen, um schnell zu entscheiden, ob sich eine nähere Beschäftigung mit der aktuellen Filmsequenz lohnt.

Die Suche in Datenbanken ist ein gut erforschtes Gebiet, während *Browsing* ein völlig anderes Benutzerverhalten darstellt, das erst mit dem Erscheinen des WWW als bedeutend erkannt wurde. Deshalb ist die Erforschung der damit zusammenhängenden Verhaltensweisen ein noch junges Forschungsgebiet. Es ist daher nicht verwunderlich, daß die normalen Videoplayer nicht mehr an Funktionalität aufweisen, als ihre physischen Pendanten, die Videorekorder.

1.1 Aufgabenstellung

Es sollte ein Softwaresystem entwickelt werden, welches die Schlüsselbilder einer Videosequenz entsprechend ihrer ursprünglichen zeitlichen oder räumlichen Reihenfolge im Dreidimensionalen anordnet und unter einem verstellbaren Sichtwinkel betrachten läßt. Ein variabler Detaillierungsgrad sollte durch Hierarchien von Ansichten einstellbar sein. Zur Berechnung der Schlüsselbilder war ein existierendes System heranzuziehen.

Das Softwaresystem sollte einen Editor zur Erstellung und Manipulation von Hierarchien, einen Viewer zur Betrachtung derselben umfassen sowie externe Hilfsprogramme, wenn erforderlich.

Der Editor sollte aus Daten, die von einem existierenden Programm zur Bewertung von Videos erzeugt werden, Hierarchien von Schlüsselbildern berechnen und diese darstellen. Die so entstandenen Hierarchien sollten mit verschiedenen Werkzeugen bearbeitet werden, zusätzlich zu den Bilddaten sollten Informationen zu den Bildern selbst sowie zu den einzelnen Ebenen der Hierarchie eingegeben und angezeigt werden.

Der Betrachter sollte die vom Editor erzeugten Daten lesen und diese anzeigen können. Bei der Implementierung sollte insbesondere auf einen künftigen Einsatz im Internet geachtet werden. Die

Anbindung an ein bestehendes Datenbanksystem sollte die dynamische Extraktion der Bilddaten zur Laufzeit ermöglichen.

1.2 Ergebnis der Arbeit

Im Rahmen dieser Diplomarbeit wurde ein Editor implementiert, der es ermöglicht, Bewertungsdaten von Videos zu lesen, aus diesen eine Hierarchie aufzubauen, diese anzuzeigen und zu editieren.

Zur Generierung der Bewertungsdateien wurde das Programm `vamt` verwendet, das verschiedene Bewertungsalgorithmen anbietet. Die resultierenden Daten werden in jeweils einer Datei abgelegt, die vom Editor ausgewertet wird.

Der Editor ist in Java implementiert, läuft in einem Java-fähigen HTML-Browser und verwendet für die Darstellung der 3D-Daten ein VRML 2.0-Plugin. Der Editor bedient sich eines externen Programms, `Extractor`, um die benötigten Einzelbilder zur Laufzeit aus den Videos zu extrahieren. Ein weiteres externes Programm (`VPlayer`) wurde implementiert, um die ausgewählten Videostücke auf Wunsch abzuspielen.

Es wurden eine Anzahl von Videos digitalisiert und bewertet, jeweils mehrere dieser Hierarchien erzeugt und gespeichert. Darüberhinaus wurden textuelle Annotationen an verschiedenen Stellen vorgenommen, nach denen gesucht werden kann.

Ein Viewer wurde auf Grundlage des Editors entwickelt, der die Betrachtung der mit dem Editor erzeugten Hierarchien dient. Zur Verwendung im WWW weist er eine Anbindung an ein existierendes Datenbanksystem auf, welches sowohl die Hierarchiedaten speichert, als auch die Bilddaten zur Verfügung stellt.

1.3 Stand der Forschung

Die Verarbeitung digitaler Videos ist ein Gebiet lebhafter Forschung, auf dem eine große Anzahl verschiedener Lösungsansätze entwickelt wurde.

Als erster Schritt in der Bearbeitung von Videodaten wird mit *Video Indexing* versucht, Videos wieder in Einstellungen zu unterteilen, aus denen sie konstruiert wurden und aus diesen Teilen möglichst genau die semantische Struktur des Inhalts zu rekonstruieren. Dies wird dadurch erreicht, indem aus den Bilddaten Merkmale extrahiert werden, die den Bildinhalt repräsentieren (etwa Farbhistogramme). Die so erzeugten Daten der Bilder werden miteinander verglichen, um anhand der Änderung die Videos in einzelne Einstellungen zu unterteilen. Diese Segmente können durch ein oder mehrere Bilder aus diesem Teil des Videos repräsentiert werden (welche *Schlüsselbilder* genannt werden), womit eine wesentliche Datenreduktion vorgenommen worden ist. Mithilfe der Bildmerkmale können die Segmente zu Hierarchien gruppiert werden, wobei die Gruppierung entweder aufgrund der zeitlichen oder der inhaltlichen Nähe erfolgt. Diese Daten werden meist in einer Datenbank gespeichert, um mit ihrer Hilfe Benutzern komplexe Such- und Browsingmöglichkeiten anzubieten.

Verschiedene Strategien zur Browsingunterstützung wurden bereits untersucht. Einige Systeme verwenden eine dynamische Präsentation von Schlüsselbildern oder Teilen aus Videos zur Wahrung des zeitlichen Zusammenhangs. Andere stellen die gefundenen Einstellungen in tabellarischer Form dar, um einen Überblick über weite Teilvideos zu bieten. Meist jedoch wird versucht, die Menge von Einstellungen durch eine Hierarchisierung zu gliedern, wie schon oben beschrieben. Diese Hierarchien werden meist als Baumstruktur in zwei Dimensionen dargestellt, in einem allgemeineren Ansatz wird von Yeung et al. (siehe Abschnitt 3.8) ein gerichteter Graph aus den Einstellungen erzeugt. Ein Ansatz zur Visualisierung von Bilddaten in mehr als zwei Dimensionen wurde von Tonomura (siehe Abschnitt 3.9) vorgestellt.

1.4 Gliederung der Arbeit

In Kapitel 2 werden die wichtigsten Algorithmen und Vorgehensweisen vorgestellt, um Videos zu indexieren. Dazu wird zuerst Grundlegendes über Filmformate und die damit verbunden Bildformate dargelegt, um dann die verschiedenen Bildmerkmale und Bewertungsmetriken vorzustellen. Ein eigener Abschnitt beschäftigt sich mit der Problematik der Auswertung dieser Daten, um daraus Schnitte im Videomaterial zu finden.

Das Kapitel 3 stellt die wichtigsten Systeme vor, die an verschiedenen Forschungseinrichtungen entstanden sind. Die Reihenfolge lehnt sich an eine 2-dimensionale Klassifizierung an, die auf die Art der Präsentation sowie die Hierarchiebildung eingeht.

Im darauffolgenden Kapitel 4 wird das vom Autor implementierte System vorgestellt, das eine neuartige Benutzerschnittstelle aufweist, in der die Schlüsselbilder in einem 3D-Raum präsentiert werden, womit sowohl die zeitliche Ordnung als auch die räumlichen Zusammenhänge zwischen den Bildern berücksichtigt werden.

In Kapitel 5 wird zusammenfassend das Erreichte erläutert, sowie noch offene Fragen und Erweiterungsmöglichkeiten aufgezeigt.

Kapitel 2

Video Indexing

Will man Filme in eine Datenbank aufnehmen, hat man das Problem, daß sie a priori keine Struktur besitzen, die vom Computer zur Gliederung ihres Inhalts herangezogen werden könnte, da sie nur einen sequentiellen Strom von Einzelbildern darstellen. Aus diesem Grund müssen aus diesen „Rohdaten“ zuerst geeignete Informationen gewonnen werden, die eine sinnvolle Gliederung des Materials erlauben, bevor dieses für den Benutzer in geeigneter Weise aufbereitet werden kann (siehe Abschnitt 2.1).

In Abbildung 2.1 sind die prinzipiellen Schritte zu sehen, die zur Aufbereitung von Videodaten notwendig sind: Zuerst müssen die gespeicherten Videos Bild für Bild betrachtet werden, um aus ihnen Merkmale zu extrahieren die für die weiteren Berechnungen genutzt werden können.

Aus den so gewonnenen Daten wird durch Vergleiche mit den zeitlich benachbarten Einzelbildern eine Bewertung für jedes Bild berechnet, aufgrund derer entschieden wird, ob ein Schnitt aufgetreten ist oder nicht.

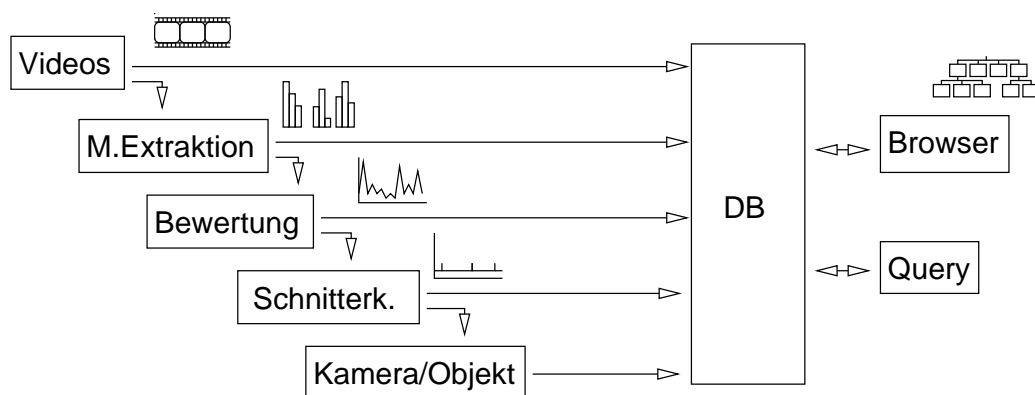


Abbildung 2.1: Prinzip Filmbewertung.

Der letzte Schritt besteht darin, aus den gefundenen Einstellungen weiterführende Erkenntnisse über den Inhalt zu gewinnen, sei es die Kameraführung oder die Identifizierung sich bewegender Objekte im Bild.

All diese Daten können in einer entsprechenden Datenbank dazu genutzt werden, gezielter und schneller zu den Informationen zu kommen, die man sucht.

In den folgenden Abschnitten sollen nun die gängigsten Algorithmen und Datenstrukturen für den vorgestellten Berechnungsschritte präsentiert werden.

2.1 Struktur von Videos

Videos erzählen Geschichten. Ähnlich wie bei Büchern werden sie aus Handlungsteilen von unterschiedlichem Gewicht aufgebaut. Angefangen bei einzelnen Erzählsequenzen, Blöcken ununterbrochener Handlung ohne zeitliche, räumliche oder gedankliche Sprünge, können immer größere Blöcke abgegrenzt werden.

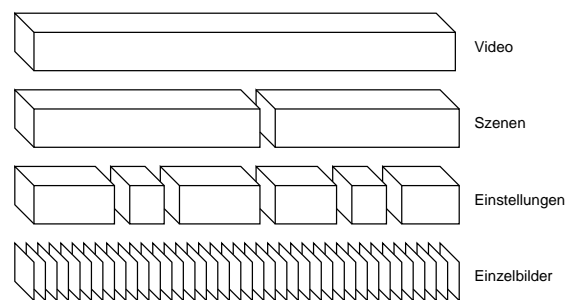


Abbildung 2.2: Video-Schichtstruktur.

Der Mensch nimmt diese Struktur und die Zusammenhänge zwischen den Teilen ganz selbstverständlich auf. Dem Computer hingegen fehlen diese semantischen Zusammenhänge. Dadurch kann diese Struktur nicht vollständig rekonstruiert, aber doch approximiert werden.

Der Aufbau eines Filmstückes ist naturgemäß stark vom Inhalt abhängig: Eine Nachrichtensendung folgt völlig anderen Gesichtspunkten als ein Epos oder ein Musikvideo.

Obwohl jedoch die Natur eines Videos nicht bekannt ist, läßt sich dennoch bis zu einem gewissen Grad eine semantische Struktur aus den Bilddaten ableiten: Die in Abbildung 2.2 dargestellte Schichtstruktur [YY97] enthält nur eine einfache semantische Hierarchiebildung.

Einzelbild: Das Einzelbild (engl. *frame*) ist die kleinste Dateneinheit eines Videos. Auf dieser Dateneinheit beruhen alle bisherigen Videoformate, Kompressionsalgorithmen und Suchstrategien.

Einstellungen: Auch *Take*, *Shot*, Kameraeinstellung oder Aufnahme, umfaßt ohne Unterbrechung alle Bilder zwischen dem Ein- und dem Ausschalten einer Kamera.

Einstellungen werden entweder durch harte Schnitte aneinandergereiht (*hard cut*), oder durch verschiedene Überblendeffekte miteinander verbunden. Gerade diese Effekte stellen für die automatische Schnitterkennung eine große Herausforderung dar.

Die Wahl des Schnittes zwischen zwei Einstellungen hat großen Einfluß darauf, wie die dadurch entstandene Filmsequenz vom Betrachter verstanden wird.

Szenen: Eine Szene stellt eine übergeordnete semantische Einheit dar, die mehrere Einstellungen umfassen kann, beispielsweise eine Dialogsequenz. Eine Hilfe bei der Erkennung zusammenhängender Einstellungen kann die Tonspur bieten: Ändert sich diese nicht signifikant, so kann angenommen werden, daß die folgenden Bilder zur aktuellen Szene gehören [LPE97].

Video: Die Filmdatei, die betrachtet werden soll oder auch ein Teilstück daraus, wird in der Regel als Videosequenz oder schlicht als Video bezeichnet.

Je nach dem Ziel der einzelnen Studien können zwischen der Szenen-Schicht und dem ganzen Video noch verschiedene Schichten definiert werden, um den Handlungsverlauf des Films genauer zu modellieren (beispielsweise könnten mehrere Szenen in einer eigenen Schicht zu größeren Handlungsblöcken, vergleichbar mit Akten in Theaterstücken, zusammengefaßt werden).

2.2 Farbräume

Der Standard in der Farbdarstellung am Computer sind die RGB-Farben. Dies hat technische Gründe, die hauptsächlich in der Monitor- bzw. Bildröhrentechnologie begründet liegen. Diese Darstellung ist zwar für die interne Repräsentation sehr gut geeignet, entspricht jedoch nicht den physiologischen Gegebenheiten des menschlichen Auges, das beispielsweise Blautöne weniger differenziert wahrnimmt. Aus dieser Überlegung heraus wurde eine Reihe von anderen Farbdarstellungen entwickelt, die dieser Tatsache Rechnung tragen. Insbesondere können Farbanteile, für die das Auge weniger empfindlich ist, mit geringerer Auflösung gespeichert werden, ohne einen merkbaren Qualitätsverlust zu erleiden.

Kasturi, Gargi et al. [KSGA96, GOK⁺95] haben gezeigt, daß die Wahl des Farbraumes, in dem Bilder verglichen werden, entscheidenden Einfluß auf die subjektive Bewertung der Ergebnisse von Schnitterkennungsalgorithmen haben: Jene Farbräume schnitten am besten ab, bei denen Farben mit getrennten Luminanz- bzw. Chroma-Werten beschrieben wurden (L^*u^*v , Munsell, OPP, HSV, L^*a^*b , YIQ). Schlechter war die Leistung der Algorithmen im RGB-Raum und bei ausschließlicher Bewertung der Luminanz.

Bemerkenswert ist dabei, daß zum Histogrammvergleich in erstgenannter Gruppe nur die Farbebenen herangezogen wurde, während die Helligkeitsdaten ignoriert wurden.

In [KSGA96] sind Umrechnungsformeln für alle von ihnen untersuchten Farbräume zu finden, hier soll nur eine Auswahl vorgestellt werden.

2.2.1 YUV

Der YUV-Farbraum wird in der Fernsehtechnik zur Übertragung von Farbfernsehen verwendet. Wie in Gleichung 2.1 zu sehen ist, setzen sich die Farben aus einem Wert für Helligkeit (Y) und zwei Chrominanz-Komponenten (U, V).

$$\begin{aligned} Y &= 0,30R + 0,59G + 0,11B \\ U &= (B - Y) * 0,493 \\ V &= (R - Y) * 0,877 \end{aligned} \quad (2.1)$$

Da für den Menschen die Helligkeitsinformation wichtiger ist als der Farbeindruck, werden die beiden Farbwerte auf Fernsehkanälen mit geringerer Bandbreite übertragen.

Dieses Farbsystem wird auch in digitalen Datenformaten häufig verwendet, beispielsweise bei JPEG und MPEG. Dabei werden die beiden Farbebenen mit halb so großer Auflösung wie die Luminanz gespeichert: Für ein Bildteil der Größe 16×16 Pixel werden 4 Y- aber nur je ein U- bzw. V-Block kodiert ($Y:U:V \equiv 4:1:1$).

Im Zusammenhang mit der digitalen Bildtechnik wird für den YUV-Farbraum meist die Bezeichnung $YCrCb$ -Farbraum oder auch *digitales YUV* verwendet.

2.2.2 OPP Farbraum

Der OPP (*OPponent color axes*) Farbraum stellt eine rechnerisch günstige Transformation von RGB-Werten in eine Darstellung dar, bei der Farben durch eine Rot/Grün- und eine Blau/Gelb-Farbachse, sowie einen Helligkeitsanteil beschrieben werden (2.2).

$$\begin{aligned} rg &= R - G \\ by &= 2 * B - R - G \\ wb &= R + G + B \end{aligned} \quad (2.2)$$

Diese in [KSGA96] vorgeschlagene Variante schnitt dort nur unwesentlich schlechter als andere Farbdarstellungen ab, deren Transformation aus dem RGB-Raum jedoch aufwendigere Rechenoperationen erfordert.

2.3 Videoformate

So vielfältig wie die Rechnerplattformen ist auch die Palette an Videoformaten, die dem Benutzer zur Verfügung stehen.

2.3.1 AVI

In der PC-Welt ist das AVI-Dateiformat das am meisten verbreitete Format zur Speicherung von Filmen. Es kapselt eine große Anzahl unterschiedlicher Kodierungsformate, die über entsprechende Softwarekomponenten, sogenannte *Codecs* (Coder/Decoder Einheiten), zugänglich sind.

Diese Kompressoren können nach den unterschiedlichsten Algorithmen arbeiten, daher kann die Forschung hier nur teilweise ansetzen, beziehungsweise können aus anderen Formaten Analogieschlüsse angewandt werden. Hinzu kommt, daß die meisten dieser Software-Bausteine, bzw. teilweise auch die Verfahren, die sie implementieren (vgl. Intel Indeo) rechtlich geschützt sind. Ähnliches gilt auch für Apples Quicktime-Format.

2.3.2 MPEG

Der MPEG Standard (benannt nach der *Moving Pictures Expert Group*) wurde auf der Basis anderer Standards, wie JPEG und H.261, entwickelt. Es werden eine ganze Reihe von Bildgrößen und Transferraten unterstützt.

MPEG definiert sowohl Video- als auch Audio-Datenströme, sowie eine *System*-Schicht, die definiert, wie mehrere solcher Ströme kombiniert werden können.

Zur Erreichung einer höheren Kompressionsrate werden bei MPEG Redundanzen zwischen zeitlich benachbarten Bildern ausgenutzt, da sich oft nur Teile eines Bildes bewegen oder bloß etwas verschoben sind.

Die Ausnutzung solcher temporaler Abhängigkeiten spiegelt sich in einer Anzahl von Bildtypen wider, die in regelmäßigen Mustern im Video angeordnet sind.

I-Bilder (*intra-coded*) zeichnen sich dadurch aus, daß sie keine Abhängigkeiten zu anderen Bildern besitzen.

P-Bilder (*predictive-coded*) benötigen zur Dekodierung das vorherige I-Bild, bzw. alle dazwischenliegenden P-Bilder.

B-Bilder (*bi-directionally predictive-coded*) können den Bildinhalt der beiden umgebenden I- oder P-Bilder benutzen, und sowohl vorwärts- als auch rückwärts vorhergesagte Makroblöcke enthalten.

Das MPEG-Video-Format ist in sechs Schichten definiert, jede kapselt die Informationen der darunterliegenden Schicht durch ein spezielles Startwort, das unkomprimiert im Datenstrom gespeichert wird:

1. *Sequence layer*: In dieser Schicht werden benötigte Ressourcen wie Bildgröße, Datenrate oder Speicherplatzanforderungen für die Dekodierung übermittelt. Hier lassen sich auch Quantisierungsmatrizen abspeichern, die mehrmals im Datenstrom geändert werden können.

2. *Group of pictures*: Eine Bildergruppe besteht aus zumindest einem I-Bild, mit dem es beginnt, sowie einer Zahl von P- u. B-Bildern. Sie stellt die Einheit für den wahlfreien Zugriff dar, und benötigt keine Informationen aus anderen GoPs zur Dekodierung.
3. *Picture layer*: In dieser Schicht werden alle bildspezifischen Daten abgespeichert, wie die Position innerhalb des GoPs, und der Bildtyp.
4. *Slice layer*: Makroblöcke werden hier zu Streifen zusammengefaßt, um im Fehlerfall eine Resynchronisierung zu erleichtern.
5. *Macroblock layer*: Ein Makroblock umfaßt jeweils vier Blöcke für Luminanzdaten und je einen pro Chrominanzebene. Er enthält die alle Daten für ein Bildstück von 16×16 Pixeln.

Es gibt verschiedene Typen von Makroblöcken: Entweder sind direkt Bilddaten enthalten, wie es bei I-Bildern und nicht vorhergesagten Makroblöcken der Fall ist, oder sie enthalten einen Bewegungsvektor zur Identifizierung der ähnlichsten Region im Referenzbild, zusammen mit sechs Blöcken in denen die Differenz (d.h. der Fehler zur Vorraussage) kodiert ist.

6. *Block layer*: Blöcke enthalten jeweils eine 8×8 -Region von Luminanz- oder Chrominanzdaten als Ergebnis einer zweidimensionalen diskreten Kosinus-Transformation (DCT), stellen also das Spektrum dieses Bildbereichs dar. Daher bezieht man sich auf den $(0, 0)$ -Wert eines solchen Blocks oft als „Gleichanteil“ dieses Bereichs, er entspricht damit dem Mittelwert (im Bildbereich), also der Grundfarbe in diesem Bereich.

Der Gleichanteil der einzelnen Blöcke kann dazu benutzt werden, sogenannte *DC-Bilder* zu extrahieren. Ihre Größe beträgt nur ein vierundsechzigstel der Originalbilder, sie können aber sehr schnell generiert werden [MC96].

Der Nachfolger von MPEG ist MPEG-2, der höhere Übertragungsraten von bis zu 40Mbit/s unterstützt und etliche andere Erweiterungen aufweist. Eine detaillierte Beschreibung ist beispielsweise in [SN95, TM96] zu finden.

2.4 Merkmalsextraktion und Bewertung

Der Ausdruck „Merkmalsextraktion“ bezieht sich auf den englischen Terminus *feature extraction*.

Zur Bewertung von Videos ist es notwendig, für jedes Einzelbild eine Beschreibung seines Inhalts zu generieren. Dazu wird aus den Bilddaten ein Satz von Kennwerten berechnet, die dann zur Weiterverarbeitung anstelle des eigentlichen Bildinhalts weiterverwendet werden.

Die verwendeten Algorithmen lassen sich verschieden klassifizieren, wie etwa nach dem Datentyp auf dem sie operieren: Hier wird zwischen der *Pixel-Domäne* und der *komprimierten Domäne* unterschieden, je nachdem ob mit den Bilddaten selbst gearbeitet wird oder mit dem noch komprimierten Datenstrom. Letzteres bedarf noch der Forschung, ist aber sehr vielversprechend, da die Dekompressionskosten und -zeiten wegfallen und etliche Eigenschaften der Kompressionsalgorithmen, wie etwa Bewegungsvektoren, ausgenützt werden können.

2.4.1 Punktdifferenz

Der einfachste Weg, die Unterschiedlichkeit zweier Bilder f_m und f_n festzustellen, ist auszurechnen, wie viele Bildpunkte sich geändert haben.

$$S(f_m, f_n) = \sum_{i,j=1}^{X,Y} D(f_m, f_n, i, j) \quad (2.3)$$

$$D(f_m, f_n, i, j) = \begin{cases} |P(f_m, l, i, j) - P(f_n, l, i, j)| & \text{für Graubilder} \\ \sum_{l=1}^3 |P(f_m, C_l, i, j) - P(f_n, C_l, i, j)| & \text{für Farbbilder} \end{cases}$$

(2.3) beschreibt die absolute Differenz zwischen zwei Bildern. $P(f_n, X, i, j)$ beschreibt dabei die Intensität des Pixels an der Stelle (i, j) für die jeweilige Farbkomponente, bzw. die Helligkeit bei Graubildern.

Ein Schnitt gilt als erkannt, wenn der errechnete Wert einen vorher festgelegten Schwellenwert übersteigt. Eine andere Form des direkten Vergleichs von Bildern ist in (2.4) angegeben: Durch den Schwellenwert δ kann das Rauschverhalten des Algorithmus genauer gesteuert werden, während bei der zuletzt vorgestellten Vergleichsmethode jeder Unterschied in die Berechnung mit einging.

$$\frac{1}{XY} \cdot \sum_{x,y=1}^{X,Y} DP_i(x, y) \quad , \text{ wobei} \quad (2.4)$$

$$DP_i(x, y) = \begin{cases} 1 & \text{wenn } |F_i(x, y) - F_{i+1}(x, y)| > \delta \\ 0 & \text{sonst} \end{cases}$$

Problematisch ist dieses Vergleichskriterium in Hinsicht auf seine Anfälligkeit bezüglich Bildverschiebungen, werden beispielsweise zwei Schachbrettmuster miteinander verglichen, die leicht versetzt sind, können daraus große Unterschiede resultieren, obwohl sich der visuelle Eindruck für den Benutzer nur unwesentlich geändert hat.

2.4.2 Histogramm-basierte Algorithmen

Histogramme stellen eine Möglichkeit dar, den Farbgehalt ein Bild zu messen. Sie beschreiben die Häufigkeit einer Menge von N Farben. Das Histogramm eines Bildes f_n ist daher ein N -dimensionaler Vektor $[H(f_n, i) : i = 1, 2, \dots, N]$. N ist dabei die Anzahl der Farben und $H(f_n, i)$ die Anzahl der Pixel im Bild mit dem Farbwert i .

Die grundlegende Idee an diesem Ansatz besteht darin, daß sich geringe Änderungen im Bildinhalt durch Objekt- oder Kamerabewegung, nur wenig auf das Histogramm auswirken. Allerdings können andere Störungen, wie das Aufblitzen von Reflexionen unerwünschte Auswirkungen haben.

Zur Reduzierung des Platzbedarfs wird die Anzahl der Bildfarben auf eine Menge von Farbklassen abgebildet. Die daraus entstehenden Farbzellen können entweder gleich groß sein oder sich an der Häufigkeit verschiedener Farbgruppen orientieren. Eine Vorgehensweise um auf letztere Art eine genauere Repräsentation des Bildinhalts zu bekommen ist in [ZZC95] beschrieben.

Wie schon im Abschnitt 2.2 erwähnt, können Bilder in verschiedenen Farbräumen dargestellt werden. Darüber hinaus können die Histogramme auch mit verschiedenen Algorithmen miteinander verglichen werden. Für eine genaue Bewertung der Leistungsfähigkeit der unterschiedlichen Methoden sei auf [KSGA96] verwiesen.

Als erstes Differenzmaß sei der Differenzbetrag zweier Histogramme als die Absolutsumme der Differenzen der Farbwerte in (2.5) definiert. Dabei steht N für die Anzahl der Pixel in einem Bild.

$$d_{diff}(h_1, h_2) = \frac{1}{N} \sum_i |H_1[i] - H_2[i]| \quad (2.5)$$

In [KSGA96] lieferte dieses Vergleichsverfahren die besten Ergebnisse, darüber hinaus ist es weniger rechenaufwendig als andere.

Die Schnittmenge zweier Histogramme (2.6) beschreibt ihre Ähnlichkeit. Daraus läßt sich auch ein Maß für die Unterschiedlichkeit ableiten, wie in (2.7) angegeben.

$$Intersec(h_1, h_2) = \sum_i \frac{\min(h_1[i], h_2[i])}{N} \quad (2.6)$$

$$d_{Inter}(h_1, h_2) = 1 - Intersec(h_1, h_2) \quad (2.7)$$

Es lassen sich aber auch statistische Maßzahlen heranziehen, um eine Wertung zu finden. Zhong et. al. ([ZZC95]) verwenden Mittelwert und Varianz der Histogramme als Metrik für die zeitlichen Veränderungen im Film (2.8).

$$\begin{aligned} \mu &= \frac{1}{N} \sum_i d_i \\ \sigma^2 &= \frac{1}{N} \sum_i (d_i - \mu)^2 \quad , \text{ wobei} \\ d_i &= \frac{1}{L} \sum_{j=1}^L (h_i[j] - \bar{h}[j]) \end{aligned} \quad (2.8)$$

L ist die Anzahl der Elemente des Histogramms, \bar{h} ist der Histogrammdurchschnitt des gesamten Videos ($\bar{h}[j] = 1/\text{frames} \sum_{i=1}^{\text{frames}} h_i[j]$).

2.4.3 Block-orientierte Methoden

Alle bisher beschriebenen Methoden beruhen auf globalen Attributen der Bilder. Zur Verringerung der Störanfälligkeit durch Aufblitzen von Reflektionen und Rauschen arbeiten die blockorientierten Methoden mit lokalen Attributen.

Dabei wird jedes Bild in eine Menge von r Blöcken aufgeteilt. Zwei Bilder werden dann verglichen, indem die einzelnen Teilbereiche des Bildes betrachtet werden. Die Ähnlichkeit zweier Bilder f_m und f_n läßt sich allgemein wie in Gleichung 2.9 definieren [IP97].

$$S(f_m, f_n) = \sum_{i=1}^r C_i \cdot S_p(f_m, f_n, i) \quad (2.9)$$

Dabei beschreibt C_i einen skalaren Gewichtungsfaktor, und $S_p(f_m, f_n, i)$ die Ähnlichkeit der Region i bezüglich der Bilder f_m und f_n .

S_p liefert als Ergebnis entweder 0 oder 1, wenn ein Schwellwert überschritten wurde. Der Regionenvergleich kann, je nach Ausgangsdaten, verschieden definiert werden.

$$d_{erw}(f_m, f_n, i) = \frac{\left(\frac{\mu_{m,i} + \mu_{n,i}}{2} + \frac{\sigma_{m,i}^2 - \sigma_{n,i}^2}{2} \right)^2}{\sigma_{m,i}^2 \sigma_{n,i}^2} \quad (2.10)$$

$$S_{p,erw}(f_m, f_n, i) = \begin{cases} 1 & d_{erw}(f_m, f_n, i) > \tau \\ 0 & \text{sonst} \end{cases} \quad (2.11)$$

(2.10) definiert den Erwartungswert als Ähnlichkeitskriterium auf der Basis von statistischen Maßzahlen: Mittelwert ($\mu_{m,i}$) und Varianz ($\sigma_{m,i}^2$) werden zur Berechnung verwendet. In (2.11) wird die Zahl der geänderten Blöcke als Ähnlichkeitsmaß für (2.9) definiert (wobei $C_i = 1$).

Der blockorientierte Vergleich läßt sich aber auch auf Histogramme anwenden, wobei jede Region durch ein eigenes Histogramm repräsentiert wird (vgl. (2.12), $C_i = 1/r$).

$$S_p(f_m, f_n, i) = \sum_{c \in R, G, B} \sum_{j=0}^{255} \frac{(H_c(f_m, i) - H_c(f_n, i))^2}{H_c(f_m, i) + H_c(f_n, i)} \quad (2.12)$$

2.4.4 Farbkorrelogramme

Farbkorrelogramme stellen eine relativ neue Bildbewertungsmethode dar, die nach [HKM⁺97] besser als die gängigen Histogrammalgorithmen abschneidet.

Analog zur Korrelation geben Farbkorrelogramme die Nachbarschaftsbeziehungen zu anderen Farbwerten in Abhängigkeit zu ihrer Distanz wieder.

$$L_\infty - Norm : |p_1 - p_2| := \max(|x_1 - x_2|, |y_1 - y_2|) \quad (2.13)$$

$$I_c(p) = \{p | I(p) = c\} \quad (2.14)$$

$$\gamma_{c_i, c_j}^{(k)}(I) = |\{p_1 \in I_{c_i}, p_2 \in I_{c_j} : k = |p_1 - p_2|\}| \quad (2.15)$$

$$\alpha_c^{(k)}(I) = \gamma_{c,c}^{(k)}(I) \quad (2.16)$$

Zur Erreichung einer effizienten Implementierung betrachten [HKM⁺97] quadratische Regionen um den betrachteten Punkt als von diesem konstant weit entfernt (2.13), die sogenannte $L_\infty - Norm$.

$I_c(p)$ stellt ein rechnerisches Hilfsmittel dar, es beschreibt ein Feld von Binärwerten, die genau dann eins sind, wenn der Bildpunkt p die Farbe c trägt.

Die Korrelation zweier Farben c_i, c_j für einen festen Abstand k läßt sich als die Mächtigkeit der Menge aller Punkt-Paare p_1, p_2 beschreiben, die k Einheiten voneinander entfernt sind (2.15).

Die Korrelation zweier Bilder für eine bestimmte Distanz k wäre eine Matrix der Größe $N \times N$ (bei N Farben). Wegen des großen Rechen- und Speicheraufwands, der für die Erstellung dieser Matrix erforderlich wäre, wird in der Regel die Autokorrelation eingesetzt (2.16).

Die Wahl der Distanzen k , für die die Korrelogramme berechnet werden, unterliegt einem Kompromiß zwischen Rechengenauigkeit und Speicheraufwand. [HKM⁺97] verwendeten eine Distanzmenge von $D = \{1, 3, 5, 7\}$ für ihre Testfälle.

2.5 Schnitterkennung

Die Struktur von Videos ergibt sich aus der Tatsache, daß Videos durch das Zusammenfügen verschiedener Segmente erzeugt werden. Ein solches Segment, auch Einstellung genannt, besteht aus einer Folge von Bildern, die von einer Kamera ohne Unterbrechung aufgenommen worden sind.

Diese filmerischen Grundbestandteile können entweder durch abrupte Übergänge aneinandergereiht werden oder durch eine Fülle von verschiedenen Effekten, die einen graduellen Übergang zwischen zwei Filmstücken erzeugen (als Beispiele seien hier Ein-, Ausblenden und Wischen genannt).

Der Sinn der Schnitterkennung liegt nun darin, ein Stück Video in sinnvolle Stücke zu unterteilen, die als Basisblöcke für die Indexierung dienen können.¹

Wie im vorigen Abschnitt schon angesprochen, muß bei den meisten Algorithmen ein Schwellwert festgelegt werden, dessen Überschreiten einen Schnittpunkt im Videostrom angibt.

¹An dieser Stelle sei darauf hingewiesen, daß der im englischen Sprachraum für Schnitterkennung gebrauchte Terminus *scene change detection* eigentlich etwas anderes beschreibt (nämlich die Erkennung eines Szenenwechsels, was semantisch eine Stufe höher angesiedelt ist). Besser sollte man hier den Ausdruck *shot boundary detection* verwenden.

Da ein geeigneter Wert stark vom untersuchten Videomaterial abhängt, ist es unmöglich, einen festen Wert vorzugeben. Aus diesem Grund wurden verschiedene Algorithmen entwickelt, die mögliche Schwellwerte aus den Differenzdaten liefern.

2.5.1 Mittelwert-Gruppierung

In [KSGA96] und anderen wird der *k-means* Algorithmus verwendet, um eine Wertemenge k Gruppen zu unterteilen. Bei [KSGA96] wird dieser Algorithmus dazu eingesetzt, die Menge der Differenzen in eine signifikante und eine insignifikante Teilmenge zu unterteilen:

Die beiden Mengen werden mit der höchsten und der niedrigsten Distanz initialisiert, sie entsprechen im ersten Schritt auch dem jeweiligen Mittelwert.

Daraufhin werden alle Differenzen der Menge einverleibt, deren Mittelwert näher liegt. Der neue Mittelwert wird aus den Elementen der Klassen berechnet, und der Algorithmus solange iteriert, bis die Änderungen unter einen Konvergenzwert fallen.

2.5.2 Doppelvergleich

Eine Schwäche aller bisher vorgestellten Algorithmen besteht darin, daß graduelle Übergänge entweder schlecht erkannt werden, oder, je nach Wahl des Schwellwertes, zu viele Scheintreffer geliefert werden.

Beim Doppelvergleich (engl. *twin-comparison*) werden zwei Schwellwerte zur Findung von Schnitten herangezogen. In einem ersten Durchlauf werden zuerst harte Schnitte mit einem hohen Schwellwert (T_c) erkannt (siehe Abbildung (2.3)).

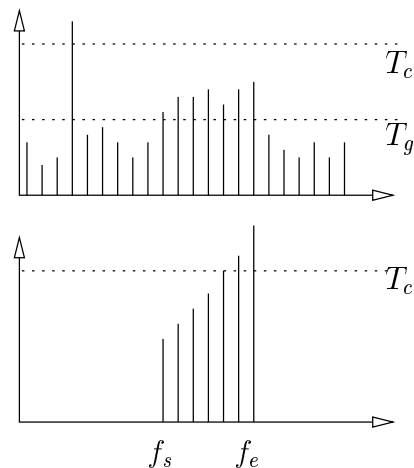


Abbildung 2.3: Doppelvergleich.

In einem zweiten Schritt wird mit einem niedrigerem Wert T_g ein mögliches Startbild f_s für einen graduellen Übergang gesucht. Von diesem ausgehend wird dann die kumulierte Änderung berechnet,

bis die Differenzwerte wieder unter T_g absinken. Wurde dabei T_c überschritten, so ist ein Schnitt gefunden worden, wenn nicht kann das Ergebnis verworfen, und beim nächsten weitergesucht werden.

Obwohl das Verfahren zum Auffinden gradueller Übergänge gut geeignet ist, liefert es jedoch keinerlei Aussage über die Art des aufgetretenen Schnitts.

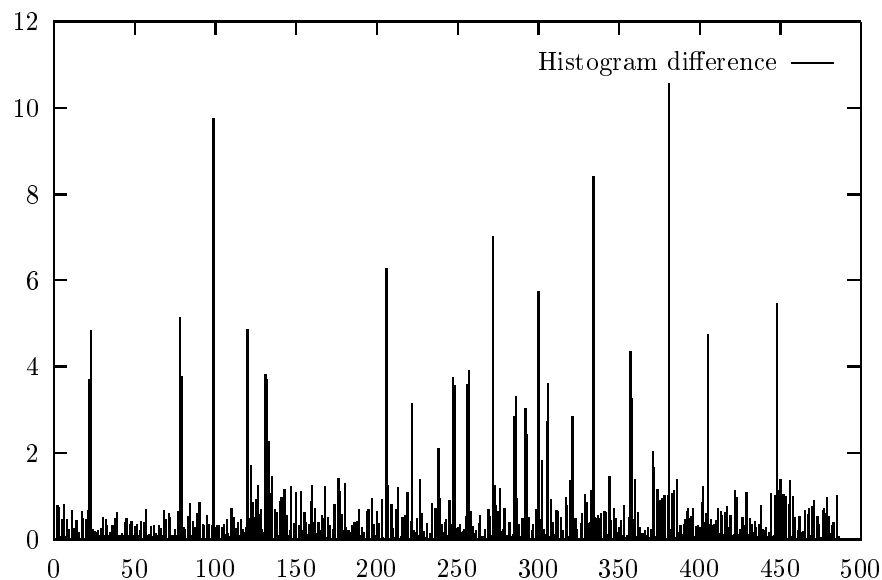


Abbildung 2.4: Histogramm-Differenzen.

2.5.3 Schnitterkennung in MPEG-Filmen

Die Analyse komprimierter Videodaten kann oft auch direkt zur Schnitterkennung herangezogen werden: In MPEG-Filmen kann die Art der Makroblöcke, insbesondere die Verhältnisse zwischen vorhergesagten zu nicht vorhergesagten, sowie ihre Richtung, als Entscheidungsmittel verwendet werden.

Da diese Verfahren weitaus weniger Rechenzeit in Anspruch nehmen, als das Rechnen auf den Bilddaten selbst, bringen sie das Indizieren von Videos in Echtzeit in greifbare Nähe.

Vikrant Kobla et al. [KDR96, KDL96] benutzen die Typinformation der Makroblöcke, um Schnitte zu erkennen: Aus der Zahl der vorhergesagten Blöcke, sowie derer, die nicht vorhergesagt werden konnten, berechnen sie ein Ähnlichkeitsmaß für jedes Bild. Wenn die Information der Makroblöcke nicht ausreicht, beispielsweise bei aufeinanderfolgenden I-Bildern, wird eine (zeitaufwendigere) Analyse der DC-Bilder (oder Gleichanteilsbilder) vorgenommen.

Die Extraktion von DC-Bildern wird neben oben angeführten Methoden auch von [MC96] eingesetzt: Wie schon in Abschnitt 2.3.2 erläutert, werden bei MPEG Einzelbilder in Blöcke unterteilt und mittels der diskreten Kosinus-Transformation (DCT) in eine Spektraldarstellung transformiert. Aus den komprimierten DCT-Daten lassen sich die Gleichanteile dieser Blöcke direkt auslesen. Diese entspre-

chen dem Mittelwert für diesen Bereich und können als Approximation für den gesamten Bildinhalt verwendet werden.

Dies funktioniert für I-Bilder direkt, für P- und B-Bilder müssen die Werte aufgrund der vorhergesagten Bildteile teilweise approximiert werden. Entsprechende Algorithmen sind beispielsweise in [YL95a] zu finden.

Auf diesen sogenannten DC-Bildern sind die bisher vorgestellten Indexing-Verfahren anwendbar.

2.5.4 Schlüsselbilder

Nach erfolgreicher Durchführung der Schnitterkennung liegt der zu untersuchende Film in einzelne Einstellungen geteilt vor. Nun gilt es, eine geeignete Zusammenfassung für das Geschehen innerhalb einer solchen Bilderfolge zu finden, die in weiterer Folge in Browsern als Repräsentation dieses Filmstücks Verwendung findet. Meist wird dafür ein einzelnes Bild aus dieser Menge ausgewählt, das in weiterer Folge *Schlüsselbild* genannt wird.

Je nachdem, welches Videomaterial untersucht wurde, kamen verschiedene Forschungsgruppen zu unterschiedlichen Ansätzen.

Die einfachste Variante ist das erste Bild der Sequenz zu verwenden. Damit wird der Beginn der Szene als Inhaltsangabe für das weitere Geschehen ausgewählt. Dies ist ein nicht unübliches Verfahren und wird auch im hier implementierten System verwendet. [ADHC94] hingegen verwenden das zehnte Bild als Repräsentant, wodurch störende Einblendeffekte, die am Anfang einer Bildfolge auftreten können, vermieden werden.

[MTCM96] untersuchte die Schnitterkennung mit Zeichentrickfilmen, wo die Aufnahmen meist durch harte Schnitte getrennt sind. Zur Findung eines geeigneten Schlüsselbildes werden die Histogramme der Einzelbilder herangezogen, ein Mittelwert μ für die gesamte Sequenz berechnet und das Bild mit der größten Abweichung von μ ausgewählt.

Für Einstellungen mit starker Aktivität oder langen Schwenks ist es oft nicht ausreichend, nur ein Schlüsselbild auszuwählen. [YL95b] beschreiben jede Einstellung durch eine Menge von Schlüsselbildern, die nach einem nichtlinearen Verfahren ausgewählt werden.

Ein anderes Verfahren zur Gewinnung von Schlüsselbildern wird in Abschnitt 3.4 des folgenden Kapitels vorgestellt: Die Generierung von Panorama-Ansichten als Inhaltsangabe.

2.6 Kamera- und Objektbewegung

Das Interesse des Benutzers gilt oft dem Geschehen zwischen den im vorigen Abschnitt erkannten Schnitten. Deshalb wurden etliche Verfahren entwickelt, um mehr Information aus den einzelnen Einstellungen zu gewinnen.

2.6.1 Kamerabewegung

Die Erkennung von Zooms und anderen Kamerabewegungen hat große semantische Bedeutung, da sie bestimmte Absichten des Regisseurs widerspiegeln. Sie spielen aber auch in technischen Belangen eine große Rolle, und zwar können sie die für Wahl des repräsentativen Bildes einer Sequenz eingesetzt werden: Oft wird eine Einstellung durch ihr Anfang oder Ende nur unzureichend repräsentiert, die Hinzunahme weiterer Bilder erhöht hier die Genauigkeit.

In [TAT97] wird ein sehr einfaches Translationsmodell (2.17) verwendet, das sich auf Verschiebungen des Blickfelds beschränkt und für große Fokallängen benutzt werden kann.

$$(x', y') = (x, y) + (d_x, d_y) \quad (2.17)$$

Dabei gibt $d = (d_x, d_y)$ den Kamera-Parameter an, der zu schätzen ist. Dies wird durch die Minimierung folgender Summenbildung erreicht:

$$MSE(d) = \frac{1}{N} \sum_{(x,y)} (f(x, y) - f'(x', y'))^2 \quad (2.18)$$

Umfangreicher ist die Bewegungskompensation die von Meng et al. ([MC96]) verwendet wird. Hierbei wird ein Bewegungsmodell mit sechs Parametern verwendet und sowohl Kamerabewegungen als auch Zooms erkannt.

Eine noch komplexere Art der Modellierung wird in [SAG95] eingesetzt, um nicht nur solche einfache Kameramanipulationen auszugleichen, sondern auch die räumliche Tiefe der Szene zu berücksichtigen.

X-Ray Bilder

Ein völlig anderer Ansatz zur Gewinnung von Kameraparametern wird in [TAOS93] angewandt: Mit Hilfe sogenannter *X-Ray Bilder* (Röntgenbilder) können Kameraschwenks, -fahrten und Zooms erkannt werden.

X-Ray-Bilder werden für eine Menge von Bildern $f \in F$ der Größe $M \times N$ auf die folgende Weise berechnet:

1. Jedes Bild durchläuft einen Kantendetektor $\bar{f}_i = edge(f_i)$.
2. Für jedes Kantenbild werden sowohl die Zeilen- ($s_r(f)$) als auch die Spaltensumme ($s_c(f)$) gebildet.
3. Das Aufsicht-Röntgenbild ergibt sich durch die Aneinanderreihung der Spaltensummen:

$$R_{top}(x) = s_c(\bar{f}_x)$$

4. Desgleichen wird die Seitenansicht aus den Zeilensummen errechnet:

$$R_{side}(x) = s_r(\bar{f}_x)$$

Anhand des Streifenbildes, das sich ergibt, lassen sich die entsprechenden Kameraparameter schätzen.

2.6.2 Objektextraktion

Durch die Errechnung der räumlichen Tiefe für jeden Bildpunkt ist es möglich, Bildelemente in verschiedene Entfernungsschichten zu unterteilen. Dadurch können Vordergrundobjekte aus dem Hintergrund gelöst und so eine Trennung zwischen beiden vorgenommen werden. Diese Vorgehensweise erlaubt einerseits das „Ausschneiden“ von Objekten aus dem Hintergrund, um so letzteren zu regenerieren, andererseits können dadurch Objekte ohne ihren Bildkontext betrachtet werden.

Die in [SAG95] angeführten Algorithmen arbeiten alle auf den unkomprimierten Bilddaten, und sind daher sehr ressourcenintensiv. Einen anderen Weg zur Objektextraktion wird von [MC96] besprochen: Die Bewegungsvektoren der komprimierten Videodaten werden analysiert.

[MC96] verwenden die errechneten Kameraparameter zur Bewegungskompensation der Bewegungsvektoren der MPEG-Makroblöcke. Durch diesen Rechenschritt wird die Länge der Vektoren für Blöcke des Hintergrunds fast gleich Null, während sich bewegende Objekte durch große Bewegungsvektoren auszeichnen. Diese können dann einfach vom Hintergrund gelöst werden, indem die Länge der Vektoren als Kriterium herangezogen wird.

Kapitel 3

Existierende Systeme

Die verschiedenen Forschungsgruppen, die auf dem Gebiet des Video Indexing arbeiten, haben eine Reihe von Systemen entwickelt, um die enorme Datenflut die bei der Archivierung, der Auswahl und dem Sichten von digitalen Filmen anfällt, beherrschbar zu machen.

Die Forschungsschwerpunkte liegen dabei auf den unterschiedlichsten Problemkreisen, sei es die Sicherstellung einer genügend hohen Bandbreite für die Übertragung bei Video-on-Demand Systemen, die geeignetste Architektur der Datenbanken um schnell Ergebnisse liefern zu können, oder die Untersuchung der Benutzerschnittstellen, um den Benutzer bei seiner Suche nach Daten möglichst effektiv zu unterstützen.

Der Aufbau und Funktionsumfang der eingesetzten Server hängt stark davon ab, wie sehr auf textuelle Informationen zurückgegriffen wird, die meist manuell eingegeben werden müssen. Besonders deutlich ist dies bei VideoSTAR (siehe Abschnitt 3.5) zu sehen, wo das Videomaterial von Hand strukturiert wird.

Die meisten unten vorgestellten Systeme gehen davon aus, daß zum Zeitpunkt des Indexing keine textuelle Information über die Struktur des Videos mehr vorliegt, wie sie aus Drehbüchern und Schnittlisten gewonnen werden könnte. Diese Projekte verwenden sehr unterschiedliche Lösungen um die resultierenden Datenmengen zu strukturieren.

Aufgrund ihrer Vielfalt ist es nicht einfach, die verschiedenen Mensch-Maschine-Schnittstellen zu klassifizieren. [DM97] beispielsweise unterteilt die existierenden Systeme nach der Art der Schlüsselbildanzeige in *statische* und *dynamische*.

Zhang et al. [ZLSZ96] unterteilen hierarchische Browser nach der Art der Hierarchiebildung in *clips*- (d.h. basierend auf Stücken von Videos, wobei Hierarchien eine Verfeinerung über die Zeit darstellen) und in *class*-basierte (Gruppierung aufgrund von Bildähnlichkeit, ähnlich der Suche in Bilddatenbanken). Letztgenannte Gruppierungsmethode wird häufig in Editoren wie WebClip (Abschnitt 3.7) eingesetzt, um schnell aus ähnlichen Einstellungen wählen zu können. Desgleichen eignet sie sich zur Zusammenfassung von Ergebnissen aus Datenbank-Anfragen.

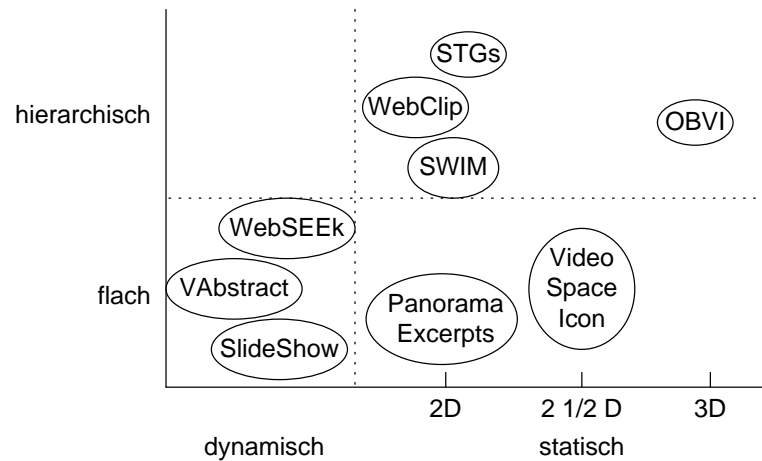


Abbildung 3.1: Überblick über existierende Systeme.

Abbildung 3.1 zeigt einen Überblick über die existierenden Systeme, die im folgenden vorgestellt werden.

Die Systeme wurden danach sortiert, ob sie Schlüsselbilder in Hierarchien darstellen oder nicht, sowie danach, ob die extrahierten Inhalte statisch oder dynamisch präsentiert werden.

3.1 Video Slide Show Interface

Ding et al. [DMT97] haben einen einfachen dynamischen Browser implementiert, um die Leistungsfähigkeit des humanen visuellen Systems zu testen. Dabei werden dem Benutzer die Schlüsselbilder verschiedener Videos mit unterschiedlichen Geschwindigkeiten angezeigt, ähnlich einem Zeitrafferfilm.

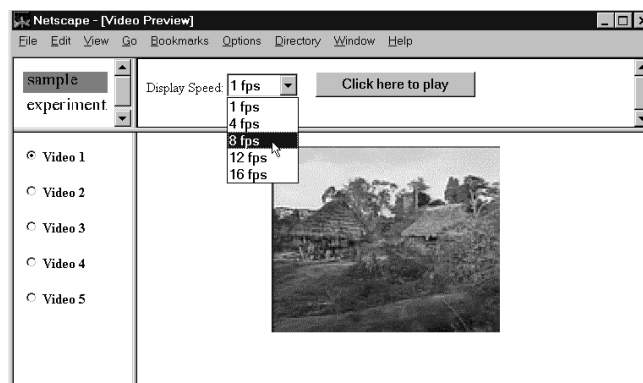


Abbildung 3.2: Video Slide Show Interface.

Die Zahl der in Erinnerung gebliebenen Bildmerkmale wurde mit der Anzahl der vergessenen für verschiedene Geschwindigkeiten verglichen. Die Testfragen, die den Probanden gestellt wurden, wurden

anhand zweier Kriterien ausgewertet: Einerseits wurde die Zahl der identifizierten Objekte getestet, andererseits die Genauigkeit, mit der sich der Inhalt der Filme gemerkt wurde.

Während bei der Objektidentifikation ein Leistungsknick zwischen 8 und 12 Schlüsselbildern pro Sekunde registriert wurde, konnte bei der zweiten Aufgabe kein ähnlicher Leistungseinbruch nachgewiesen werden.

3.2 VAbstract: Generierung von Film-Trailern

Im Rahmen des MoCA-Projektes (*Movie Content Analysis*) wurde an der Universität Mannheim eine Arbeitsumgebung zur Entwicklung und Analyse verschiedener Audio- und Videoanalyse-Algorithmen entwickelt [LPE97].

Im Zuge dieses Projektes wurde mit VAbstract ein System entwickelt, das automatisch aus einer längeren Filmsequenz eine Inhaltsangabe generiert, die den Trailern in der Filmbranche entspricht.

Die Erstellung einer solchen Inhaltsangabe vollzieht sich in drei Schritten:

1. *Video Segmentierung und Analyse.* Zuerst werden die einzelnen Aufnahmen ermittelt. Diese Abschnitte werden durch folgende Heuristiken zu Szenen als übergeordnete Einheit zusammengefaßt. Um die Grenzen dieser semantischen Blöcke zu finden, werden folgende Heuristiken eingesetzt:
 - Aufeinanderfolgende Aufnahmen mit ähnlichen Farbeigenschaften stellen meist den selben Ort aus verschiedenen Blickwinkeln dar.
 - Die Tonspur ändert sich normalerweise stark von Szene zu Szene. Um einen solchen Audio-Schnitt zu finden, wird in der Umgebung eines Schnitt-Kandidaten die Spektralverteilung der Tonspur berechnet, und interpoliert. Die Abweichung zwischen vorhergesagtem und tatsächlichem Wert nach dem Schnitt dient dafür als Indikator.
 - Aufeinanderfolgende Einstellungen werden auch gruppiert, wenn sie als zu einem Dialog gehörig identifiziert werden (Szenenfolge -A-B-A-B-).

Danach werden mittels verschiedener Algorithmen semantisch reichhaltige Szenen gesucht, im speziellen nach Großaufnahmen von Darstellern, Dialogen und Action-Szenen, die sich durch Explosionen und dergleichen auszeichnen.

2. *Clip Selection.* In der zweiten Phase werden aus der Menge der Szenen diejenigen ausgewählt, die in die Inhaltsangabe aufgenommen werden. Aus einer Menge von Kandidaten (das sind die Szenen der ersten 80% des Films, um nicht das Finale zu zeigen) wird zuerst ein Prozentsatz von Action- und Dialogszenen ausgewählt, die im vorherigen Schritt identifiziert wurden. Anschließend wird der Trailer mit Füllszenen aus der übrigen Menge solange angereichert, bis die gewünschte Länge erreicht ist.

Aus den Szenen werden einzelne Einstellungen als Repräsentant ausgewählt.

In die Menge der extrahierten Szenen wird auch immer die Textsequenz vom Anfang des Films enthalten, die den Titel und eine Aufzählung der Hauptdarsteller enthält.

3. *Clip Assembly.* Die ausgewählten Einstellungen werden im letzten Arbeitsschritt zur endgültigen Zusammenfassung vereinigt, wobei je nach Übergang verschiedene Misch-Effekte angewandt werden. Besonderes Augenmerk wurde hierbei auf die Zusammenstellung der Tonspur gelegt, da Schnitte in Dialogsequenzen fatale klangliche Konsequenzen nach sich ziehen würden.

Vor der Extraktion kann der Gehalt an Szene-Typen (wie etwa Großaufnahmen, Action-Sequenzen oder Dialogen) mit Parametern vorgegeben werden, um die Inhaltsangabe an verschiedene Zielgruppen anzupassen.

In [LPE97] schlagen die Autoren diese visuellen Inhaltsangaben als Ersatz für ihre textuellen Pendant vor, da sie zum Browsen mehrerer Dokumente eine wesentlich bessere Beschreibung bieten.

3.3 WebSEEk

Das WebSEEk Projekt wurde an der University von Columbia entwickelt [SC96]. Es stellt ein visuelles Informationssystem dar, das die Suche von Bildern und Videos im WWW erlaubt. Dies geschieht in einer Kombination von textuellen und inhaltsbezogenen Anfragen an die Suchmaschine.



Abbildung 3.3: WebSEEk Benutzerschnittstelle.

Die Datenbank, aus der die Benutzeranfragen beantwortet werden, wird von mehreren autonomen Agenten aufgebaut:

1. Der erste lädt eine Liste von HTML-Seiten vom Netz und reicht diese an den zweiten weiter.
2. Aus den HTML-Seiten werden die Links extrahiert, und nach Typen sortiert.
Verweise auf andere Seiten werden an den ersten Agenten zurückgereicht, während Verweise auf Videos und Bilder an den dritten geschickt werden. Verweise auf Audiodateien, Applets und auf andere Dateitypen werden verworfen.
3. Der dritte Agent lädt dann die jeweiligen Bilder oder Videos auf den Rechner um diese zu analysieren.

Nach dem Herunterladen werden die Daten in ein einheitliches Format konvertiert. Daraus werden dann verschiedene Nutzdaten gewonnen:

- Größe
- Farbhistogramm
- Icon. Die Bilder werden in einem kleineren Format komprimiert abgespeichert. Filme werden hierbei nicht nur räumlich, sondern auch zeitlich komprimiert: Um Rechenzeit zu sparen, werden zuerst nur je ein Bild pro Sekunde aus dem Film extrahiert. Mithilfe eines Schnittdetektors werden die Grenzen der Einstellungen erkannt und pro Einstellung nur jeweils ein Bild für die Zusammenfassung beibehalten. Dann werden die verbliebenen Bilder wieder zu einem Video vereint.

Aus der URL der Bilder werden die textuellen Attribute der Bilder gewonnen, indem Dateiname und Verzeichnis als Schlüsselwort für den Bildinhalt und einer Klassifizierung gewertet werden.

Dieser Schritt geschieht halbautomatisch: Die extrahierten Eigenschaften werden nach Häufigkeit sortiert dem Datenbankadministrator vorgelegt, der solche Begriffe entfernt, die nondeskriptiver Natur oder mehrdeutig sind (etwa *pictures*, *graphic*; *rock* könnte sich auf eine Musikrichtung, Steine oder Kleidung beziehen).

Das gleiche geschieht mit den Bildklassen, die aus den Verzeichnisnamen gewonnen wurden. Aus diesen wird dann eine hierarchische Struktur der Datenbank aufgebaut, durch die man themenbezogen navigieren kann.

Der Benutzer kann die Datenbank auf verschiedene Arten durchforsten, entweder indem er über eine Suche nach Schlüsselwörtern oder durch Auswahl eines Themas eine Tabelle mit gefundenen Bildern erhält.

Die Ergebnisse werden dem Benutzer in tabellarischer Anordnung präsentiert (Abbildung 3.3).

Basierend auf dieser Auswahl kann er anschließend nach Bildern mit ähnlichem Farbgehalt durchsuchen. Dabei besteht auch die Möglichkeit, das Histogramm zu manipulieren.

3.4 PanoramaExcerpts

PanoramaExcerpts [TAT97] stellen die aus den Filmen gewonnenen Bilddaten ebenfalls in einer Ebene ohne Einführung einer Hierarchie dar. Dazu werden aus den Einstellungen Panoramen generiert. Durch diese unterschiedlich großen Bilder werden komplexere Layoutalgorithmen notwendig.

Wie schon in Abschnitt 2.6.1 erläutert, wird von [TAT97] ein einfaches Projektionsmodell verwendet, um die Kamerabewegung aus den Bildfolgen zu rekonstruieren. Die so gewonnenen Daten über die Bewegung werden dazu benutzt, die Einzelbilder zu einem Panorama zu verbinden.

Dies wird erreicht, indem die Bilder um den jeweils errechneten Wert der Bewegung transformiert, übereinandergelegt und die sich überlappenden Pixel verschmolzen werden. Dazu wird für jeden



Abbildung 3.4: PanoramaExcerpts: Ausschnitt aus einem Griechenland-Video.

Punkt des Panoramas der Median oder der Mittelwert über die darüberliegenden Pixel der Ausgangsbilder gebildet.

Mittels Heuristiken werden die entstandenen Panoramen derart auf einer Übersichtsseite plaziert, daß einerseits für den Benutzer der zeitliche Fluß gewahrt bleibt, aber andererseits nicht zu viel Platz durch Leerräume verschenkt wird.

Der Vorteil dieser Methode besteht darin, daß die Wahl des Schlüsselbilds für jedes Stück Film wegfällt, und damit einhergehende Probleme vermieden werden. So können etwa Anfang oder Ende von langen Schwenks nicht aussagekräftig genug für die dazwischenliegende Strecke sein.

3.5 VideoSTAR

Am norwegischen Institut für Technologie wurde von Hjelsvold, Langørgen, Midtstraum und Sandstå eine Video-Datenbank namens VideoSTAR (*Video Storage And Retrieval*) implementiert. Sie stellt eine Forschungsplattform zum Indexieren, Durchsuchen, Browsen und Abspielen von digitalen Filmen dar [HLMS95, HMS95].

Die Browsing-Unterstützung stützt sich im Wesentlichen auf textuelle Metadaten des Films, das heißt auf Annotationen und Strukturinformation die in einem eigenen Arbeitsschritt von Hand eingegeben werden muß:

- Annotationen müssen von Benutzern eingefügt werden; es werden drei Typen unterschieden: Personen-, Orts- und Ereignisannotationen. Sie werden auf jeweils einem Zeitabschnitt des Videos definiert und können später über eine Abfragemaske gesucht werden.
- Die semantische Struktur des Films wird in drei Schichten modelliert: In Aufnahmen, Szenen und Sequenzen. Szenen umfassen hierbei logisch zusammengehörige Einstellungen, etwa verschiedene Ansichten eines Sprechers oder einen Nachrichtenartikel.

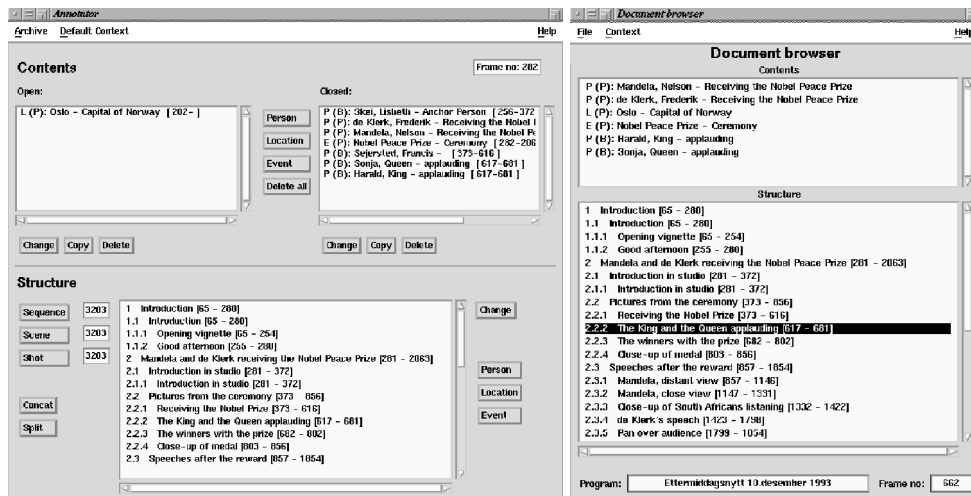


Abbildung 3.5: VideoSTAR. Video-Annotator, Document Browser.

Diese Information wird dem Benutzer im *Document Browser* präsentiert und soll den Überblick über des Geschehens erleichtern, wenn man durch eine Abfrage auf ein kurzes Videostück trifft.

3.6 SWIM

An der Universität von Singapur wurde ein System zur Verwaltung von Videodaten namens SWIM (Show What I Mean) geschaffen, das eine Vielzahl von Interaktionsmöglichkeiten gestattet, um die gesuchte Sequenz schnell zu finden.

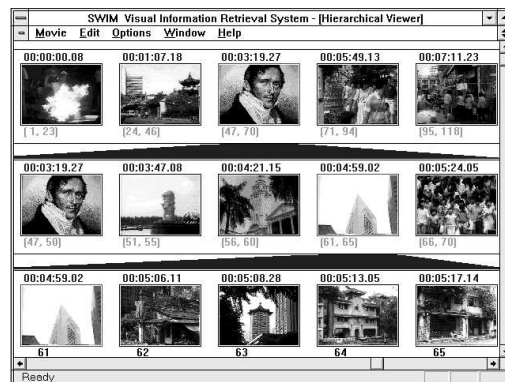


Abbildung 3.6: SWIM.

Der Schlüsselbild-Browser (Abbildung 3.6) unterstützt sowohl eine Ordnung der Schlüsselbilder nach Bildklassen, als auch die zeitliche Segmentierung.

Darüber hinaus unterstützt das System die Auswahl von Sequenzen durch inhaltsbezogene Suche: Jedes Schlüsselbild ist in 3×3 Regionen unterteilt, deren Histogramme in der Datenbank gespeichert sind. Dadurch lassen sich Szenen mit bestimmten Farbmerkmalen in verschiedenen Bildregionen suchen und anzeigen.

3.7 CVEPS, WebClip

Das CVEPS (Compressed Video Editing and Parsing System) wurde, wie genauso wie WebSEEK (siehe Abschnitt 3.3), ebenfalls an der Universität von Columbia entwickelt [MC96]. Es ist ein Filmeditor, der rein auf komprimierten MPEG-Daten arbeitet. Dabei wird nur ein Minimum der Daten dekomprimiert, wodurch ein Maximum an Leistungsfähigkeit erreicht wird. Insbesondere stehen etliche Spezialeffekte zur Verfügung, die direkt auf die komprimierten Bilddaten angewandt werden können.

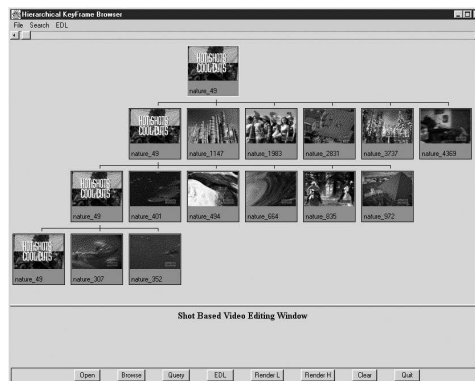


Abbildung 3.7: WebClip.

WebClip ist eine Neuentwicklung, die mit den Prinzipien von CVEPS arbeitet, aber speziell für die Filmbearbeitung über das Internet ausgerichtet ist (Abbildung 3.7).

CVEPS besteht aus 3 Modulen, die für die Schnitterkennung, Visualisierung und das Editieren von MPEG-Strömen verantwortlich sind.

Zur Schnitterkennung werden DC-Bilder aus den Filmen extrahiert, sowie die Zahl der Bewegungsvektoren (mit den in Abschnitt 2.5.3 vorgestellten Verfahren). Beide werden zur Schnitterkennung verwendet.

3.8 Scene Transition Graphs

Scene Transition Graphs (Szenen-Übergangs-Graphen) sind gerichtete Graphen, die Inhalt und Struktur von Videosequenzen zusammenfassen.

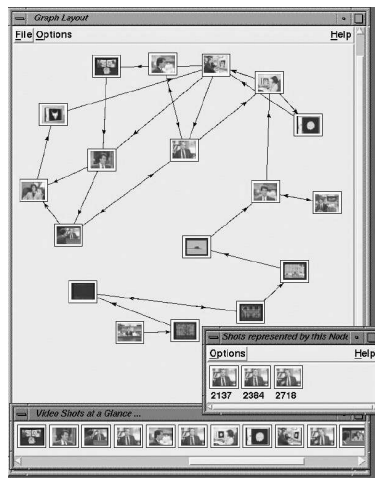


Abbildung 3.8: Scene Transition Graphs.

Zum Aufbau des Graphen wird zuerst das Video in seine Einstellungen zerlegt. Diese werden nach einem Verfahren, das in [YL95b, YYL96] vorgestellt wird, zu Klassen ähnlicher Einstellungen zusammengefaßt.

Die Transitionen zwischen den Einstellungen werden daraufhin auf die korrespondierenden Klassen abgebildet, woraus sich die Graphenstruktur ergibt.

Für lange Videosequenzen wird das oben vorgestellte Verfahren mit einer Zeitschranke versehen, wodurch immer nur ein Teil des Videomaterials begutachtet wird. Dadurch werden nicht zu weit auseinanderliegende Einstellungen verschmolzen, woraus zu komplexe Graphen entstehen würden.

3.9 VideoSpaceIcon, VideoMap

In [TAOS93] werden von Tonomura et al. zwei verschiedene Werkzeuge zur visuellen Zusammenfassung von Videosequenzen vorgestellt. Als Basis für die Analyse der Videos dienen eine Reihe verschiedener Bildmerkmale wie Intensitäts- und Farbhistogramme sowie X-Ray-Bilder (siehe Abschnitt 2.6.1).

VideoMap stellt für eine Folge von Einstellungen alle extrahierten Merkmale der Bilder dar, und bietet so eine kompakte Übersicht über den Filminhalt.

VideoSpaceIcon ist eine $2\frac{1}{2}$ D-Darstellung der Kamerabewegung in einer Einstellung, bei der die Kamerabewegung dazu benutzt wird, eine räumliche Darstellung der Szene zu generieren (Abbildung 3.9).

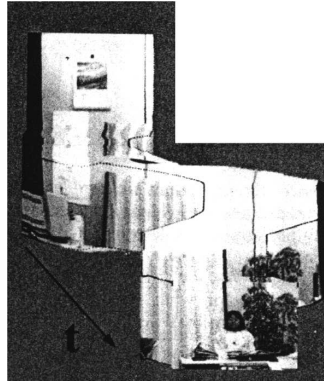


Abbildung 3.9: VideoSpacelcon.

Kapitel 4

Implementierung

Im Rahmen des praktischen Teils der Diplomarbeit entwickelte ich ein System zur Video-Indexierung, das insbesondere die Visualisierung und Manipulation von hierarchischen Repräsentationen von Videos ermöglicht.

Ein Hauptaugenmerk lag dabei auf der Eignung des erstellten Programms zur Verwendung in WWW-Seiten, woraus sich Implikationen bezüglich der Implementierung ergaben. Es fiel daher die Wahl auf Java zur Implementierung der Programme und VRML für die Visualisierung, da beides für viele Plattformen verfügbar ist.

OBVI (Object Based Video Indexing) stellt unseren Ansatz zur Bewertung und Visualisierung von Videos dar. Dabei werden die extrahierten Schlüsselbilder in einem 3D-Raum derart angeordnet, daß zwischen ihnen sowohl der temporale wie auch der spatiale Zusammenhang gewahrt bleibt. Der Zusatz „objektbasiert“ bezieht sich auf die Möglichkeit, Objekte, wie etwa Schauspieler, zu definieren die in Teilsequenzen vorkommen und diese ohne den umgebenden Hintergrund darzustellen (s.u.).

Das System teilt sich in zwei Hauptprogramme, einen Editor zur Erstellung von OBVIs und einen Viewer, der nur das Betrachten der erzeugten Hierarchien erlaubt und für den Internet-Einsatz gedacht ist.

Der nächste Abschnitt beschreibt die Zusammenhänge zwischen dem digitalen Film und den dazu generierten Informationsdateien, sowie den Aufbau der hierarchischen Repräsentation. Im folgenden Teil möchte ich den Editor in Funktionsweise und Aufbau beschreiben, um danach auf die Eigenschaften des Viewers einzugehen, der im wesentlichen nur eine Teilmenge der Funktionalität umfaßt. Schließlich möchte ich noch die Arbeitsschritte vorstellen, die notwendig sind, um aus einem Film ein OBVI zu erzeugen.

4.1 Daten- und Dateistrukturen

Zur Erzeugung von OBVIs werden eine Menge von Metadaten über die eigentlichen Videodaten benötigt. Darüberhinaus fallen bei ihrer Entstehung und Manipulation noch eine Reihe von anderen Daten an, die in Beziehung zum Film oder seinen Teilen stehen.

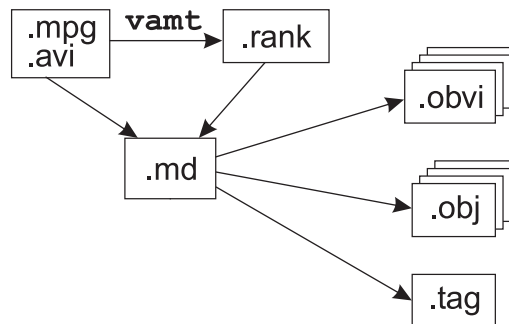


Abbildung 4.1: Datei-Struktur.

In Abbildung 4.1 ist der Zusammenhang zwischen den einzelnen Dateien, bzw. Datenklassen, und dem eigentlichen Film zu sehen. Die folgenden Absätze geben eine Erklärung über den Inhalt der jeweiligen Dateien und ihren Zusammenhang. Eine formale Spezifikation der Daten findet sich in Anhang A.

Movie-Descriptor (.md): Dies ist die zentrale Definitionsdatei, die vom Editor zuerst gelesen wird. Sie enthält alle relevanten Daten über den Film, wie seine Länge und die Anzahl der erzeugten OBVIs und Objekte.

Er beinhaltet Referenzen auf alle benötigten Dateien, die im folgenden aufgeführt sind.

Ranking (.rank): Die Bewertungsdatei, auch *ranking*-Datei genannt, enthält einen Wert für jedes Bild des Films, sodaß sich eine Hierarchie aufbauen läßt.

Diese Datei und die Filmdatei selbst sind notwendig, um den Movie-Descriptor erzeugen zu können. Die genaue Vorgehensweise wird in Abschnitt 4.1.1 erläutert.

Tags (.tag): In dieser Datei werden Annotationen zu den einzelnen Bildern gespeichert, die für alle OBVIs des Films global definiert sind.

Im momentan implementierten System ist nur vorgesehen, daß textuelle Beschreibungen eingegeben werden können. Wie aus Anhang A.3 zu ersehen ist, ist eine Erweiterung um andere Typen, im speziellen um Links im URL-Format, bereits vorgesehen.

OBVI (.obvi): Für jeden Film kann eine beliebige Anzahl von OBVIs erzeugt werden, die in jeweils einer eigenen Datei gespeichert werden. Der Aufbau der Hierarchie wird im nächsten Abschnitt beschrieben.

Objekte (.obj): Desgleichen kann eine Reihe von Objekten definiert werden die im Film auftreten. Dazu können zu einer Menge von Bildnummern die dazugehörigen Bilddateien angegeben wer-

den, die das extrahierte Objekt (d.h. das Objekt umgeben von einem transparenten Hintergrund) enthalten (Abbildung 4.2).

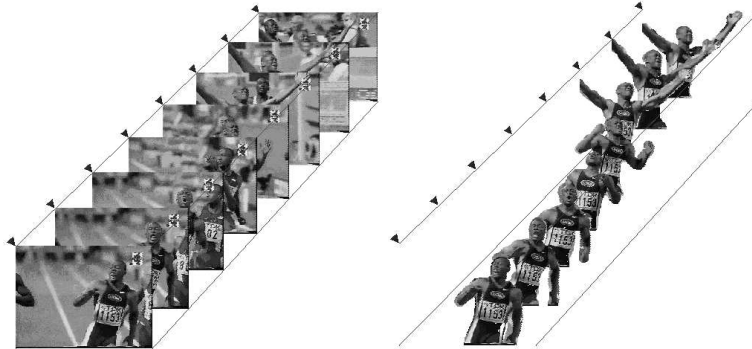


Abbildung 4.2: Läufer, Originalsequenz und extrahiertes Objekt.

Diese Objekte können auf Wunsch anstelle der normalen Bilder angezeigt werden, um so Bewegungsabläufe und dergleichen deutlicher analysieren zu können. Im unten beschriebenen System ist noch keine Möglichkeit enthalten, solche Objektinformationen zu editieren, eine Erweiterung um diese Möglichkeit ist allerdings vorgesehen.

4.1.1 Generierung eines OBVIs

Wie schon in Kapitel 2 angesprochen, sind etliche Schritte notwendig, um von gespeicherten Videodaten genügend Informationen zu extrahieren, sodaß man Browsing in geeigneter Weise unterstützen kann.

Filmbewertung

Die Bewertung der Einzelbilder wurde mit einem externen Programm namens **vamt** (Video Activity Measurement Tool) generiert. Dieses Programm arbeitet mit zwei verschiedenen AVI-Filmformaten (Intel Indeo 3.2, 24bit sowie Radius Cinepak), die es selbst dekomprimiert. Diese Einschränkung macht es notwendig, Filmdateien erst in das richtige Format zu bringen, bevor sie zur Bewertung herangezogen werden können.

Von **vamt** werden insgesamt sieben verschiedene Bildbewertungsalgorithmen zur Verfügung gestellt, die sich in ihrer Leistungsfähigkeit stark unterscheiden:

Algorithmus 1: Absolut-Differenz von Grauwerten: Bilder werden in ein Feld von $M \times N$ von Blöcken geteilt, die jeweils durch den mittleren Grauwert der enthaltenen Pixel repräsentiert werden, bezeichnet als $B_t(i, j)$.

Die Bewertung für jeden Block ergibt sich als

$$J_t(i, j) = \begin{cases} 1 & |B_{t+1}(i, j) - B_t(i, j)| > \tau \\ 0 & \text{sonst.} \end{cases} \quad (4.1)$$

Das gesamte Bild wird mit der Summe über alle J_t bewertet:

$$\text{rank}_1(t) = \sum_i^M \sum_j^N J_t(i, j) \quad (4.2)$$

Algorithmus 2: Dieser Algorithmus stellt eine Variante des oben vorgestellten dar: Die Bewertung ergibt sich aus der Summe der Differenzen selbst, wodurch der Schwellwert für die Änderung je Block wegfällt:

$$\text{rank}_2(t) = \sum_i^M \sum_j^N |B_{t+1}(i, j) - B_t(i, j)| \quad (4.3)$$

Algorithmus 3: Eine weitere Variante stellt der dritte Algorithmus dar: Als Bewertung wird die minimale Änderung herangezogen:

$$\text{rank}_3(t) = \min(|B_{t+1}(i, j) - B_t(i, j)|) \quad (4.4)$$

Algorithmus 4: Der Erwartungswert stellt die Bewertungsgrundlage der vierten Bewertungsmethode dar:

$$l_t(i, j) = \left(\frac{\frac{\sigma_t^2(i, j) + \sigma_{t+1}^2(i, j)}{2} + \frac{(\mu_t(i, j) - \mu_{t+1}(i, j))^2}{2}}{\sigma_t(i, j)\sigma_{t+1}(i, j)} \right)^2 \quad (4.5)$$

$$J_t(i, j) = \begin{cases} 1 & J_t(i, j) > \tau \\ 0 & \text{sonst.} \end{cases} \quad (4.6)$$

$$\text{rank}_4(t) = \sum_i^M \sum_j^N J_t(i, j) \quad (4.7)$$

Algorithmus 5: Als erster von zwei histogrammbasierten Algorithmen wird die Summe der Absolutdifferenzen zur Bewertung herangezogen.

$$H_t(g) = \text{Zahl der Pixel mit Grauwert } g \quad (4.8)$$

$$\text{rank}_5(t) = \sum_{g=1}^G |H_{t+1}(g) - H_t(g)| \quad (4.9)$$

Algorithmus 6: Der sechste Algorithmus stellt eine Variante des eben besprochenen dar. Als Bewertung dient die maximale Abweichung anstatt der Differenzsumme:

$$\text{rank}_6(t) = \max(|H_{t+1}(g) - H_t(g)|) \quad (4.10)$$

Algorithmus 7: Die letzte Art der Bildbewertung basiert auf der Analyse des optischen Flusses, der als Summe über die Länge der Bewegungsvektoren ($mv_t(x, y)$) im Bild berechnet wird:

$$rank_7(t) = \sum_{x,y} mv_t(x, y) \quad (4.11)$$

Im Laufe unserer Tests hat sich Algorithmus 5 als der brauchbarste herausgestellt, und wurde zur Erzeugung der Ausgangshierarchien verwendet. Diese Wertung beruht allerdings auf dem subjektiven Eindruck der am Projekt Beteiligten, es wurden keine numerischen Analysen durchgeführt, da die dafür notwendigen Ressourcen nicht vorhanden waren (ein Durchlauf des Java-Programms benötigte für rund 500 Bilder einen Zeitraum von 5–10 Stunden, längere Sequenzen konnten gar nicht analysiert werden).

Das Resultat der Bewertung eines Filmes mit oben angeführtem Programm ist ein Fließkommawert für jedes Bild des Films. Dieses steht zur späteren Verwendung in einer Bewertungsdatei zur Verfügung, von der im nächsten Abschnitt Gebrauch gemacht wird.

Da *rank* nur auf Grauwerten arbeitet, sind die Algorithmen stark auf Helligkeitsschwankungen anfällig. Die Auswertung von Farbinformation, etwa durch Farbhistogramme oder -korrelogramme, sollte entscheidende Verbesserungen bezüglich Qualität und Geschwindigkeit der Bewertung bringen.

Erzeugung von OBVIs

Wie oben schon beschrieben, charakterisiert die Bewertung (auch *ranking* genannt) die Wichtigkeit der einzelnen Bilder des Films. Diese wird benötigt, um eine Hierarchie von Schlüsselbildern aufzubauen.

Zur Erzeugung eines OBVIs für einen neuen Film wählt man im Editor Film und zugehörige Bewertungsdatei aus, um eine neue Definitionsdatei, einen *Movie-Descriptor* (siehe Abschnitt 4.1), zu erzeugen.

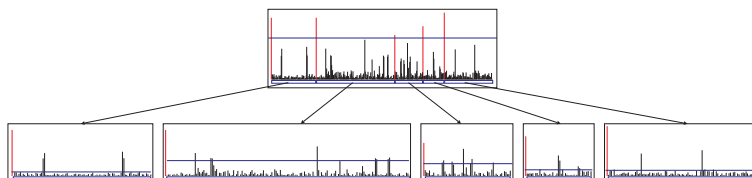


Abbildung 4.3: Hierarchiegenerierung.

Aus den Bewertungsdaten wird in weiterer Folge ein Baum aufgebaut, bei dem jeder Knoten bis zu k Elemente enthält. Der Wert für k kann vom Benutzer in einem Dialog für verschiedene Ebenen der Hierarchie festgelegt werden, bevor der Baum erzeugt wird.

Jeder Knoten des Baumes enthält eine Fülle von Daten, die teils bei seinem Aufbau, teils durch Editieren der entstandenen Struktur bestimmt werden können:

- Er enthält bis zu k Elemente, bestehend aus jeweils einem Schlüsselbild und jenem Teilbaum, den dieses Bild repräsentiert. Dieser Sohn kann natürlich auch leer sein, wenn das Element ein Blatt ist.
- Die Zeitspanne, die der Knoten insgesamt repräsentiert. Für den Wurzelknoten entspricht dies der Länge des gesamten Films.
- Weitere Daten, um die der Knoten angereichert werden kann. Dazu zählen ein Textfeld, in dem Informationen über den Knoten abgelegt werden können, oder ein Verweis auf eine Audiodatei, die auf Wunsch abgespielt werden kann.

Zur Erzeugung des Wurzelknotens werden jene k Bilder ausgewählt, die die höchste Bewertung besitzen. Dadurch wird die Zeitspanne des Films in k Teilstücke gespalten, die von diesen Bildern repräsentiert werden. Sinnbildlich entspricht dies dem Senken eines Schwellwertes über dem Ranking, bis k Werte über diesen hinausragen (siehe Abbildung 4.3).

Dieser Teilungsalgorithmus wird dann rekursiv auf die einzelnen Teilsequenzen angewandt, die dadurch ihrerseits wieder in Knoten gespalten werden. Dieser Prozess wird solange fortgesetzt, bis die Knoten nur noch Blätter enthalten, sprich aus einer Reihe von aufeinanderfolgenden Einzelbildern bestehen.

Das erste Bild des Films wird, seine Bewertung ignorierend, automatisch in die höchste Ebene aufgenommen, da der Anfang eines Filmstückes prinzipiell von Interesse ist.

Ein weiterer diskussionswürdiger Punkt ist die Wahl des repräsentierten Intervalls: Die Grenzen für einen bestimmten Teilabschnitt, der durch ein Schlüsselbild zusammengefaßt wird, beginnt bei ebendiesem und endet ein Bild vor dem nächsten Schlüsselbild des Knotens (beziehungsweise endet er, im Falle des letzten Elements, mit dem Intervallende des Knotens).

Durch diese Wahl der Grenzen befindet sich das Schlüsselbild innerhalb des Intervalls, das es repräsentiert: Es ist das erste Bild des repräsentierten Zeitraums. Dies hat zur Folge daß das erste Element auf der nächsten Hierarchie-Ebene wieder durch ebendieses Schlüsselbild repräsentiert wird (ähnlich wie bei SWIM, siehe Abschnitt 3.6).

Dies gewährleistet eine optische Konsistenz beim Betreten einer Hierarchie, da man das gewählte Bild als das erste des neu angezeigten Knotens präsentiert bekommt.

Dies stellt allerdings nur eine von vielen möglichen Varianten dar. Stehen beispielsweise die Schnittübergänge im Mittelpunkt des Interesses, erscheint es sinnvoll, die repräsentierte Zeitspanne anders zu wählen, etwa indem das Schlüsselbild jeweils den (zeitlichen) Mittelpunkt seines Intervalls darstellt. Dadurch erscheinen sowohl Bilder vor als auch nach dem Schnitt in einem Knoten und können genauer analysiert werden.

4.2 OBVI-Editor

Der Editor dient der Erstellung und der Manipulation von OBVIs, bevor diese in eine Datenbank auf einem Server eingebunden werden.

Um die Verwendung des Systems auch ohne Datenbank-System zu ermöglichen, wurde das Programm zuerst so konzipiert, daß alle Metainformationen zu einem Film in Dateien abgelegt werden, wie schon im vorigen Abschnitt besprochen.

4.2.1 Benutzerschnittstelle

Der Editor ist, wie in Abbildung 4.2.1 zu sehen, in eine WWW-Seite eingebettet und in Java und VRML implementiert. Wie bereits oben angesprochen war eine der Zielsetzungen die Entwicklung eines Viewers für das WWW, daher wurde dieser Weg beschritten.

Da Java eine interpretierte Sprache ist, wird größtmögliche Portabilität gewährleistet. Allerdings ergeben sich aus genau dieser Tatsache erhebliche Leistungseinbußen bei der Programmausführung, die es unmöglich machen, die Visualisierung in dieser Sprache umzusetzen.

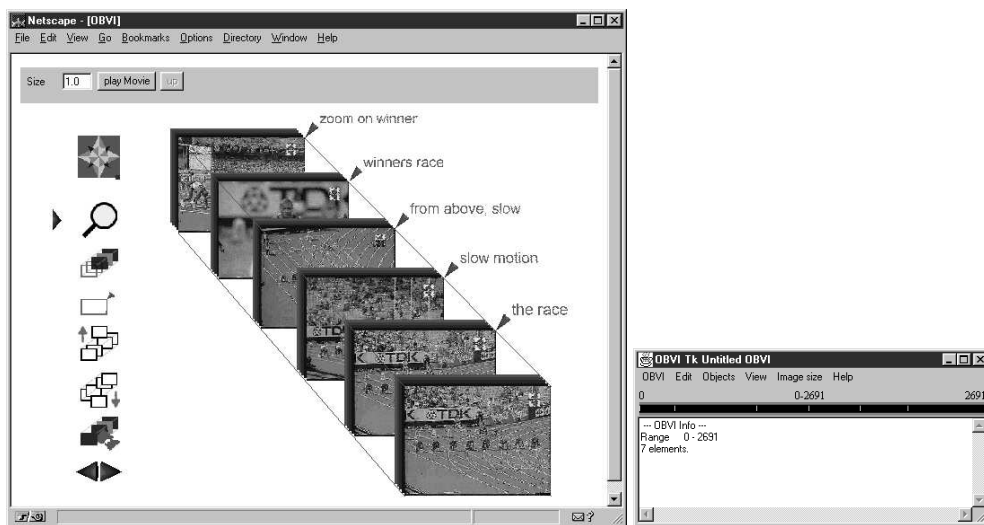


Abbildung 4.4: OBVI Editor.

Die breite Unterstützung für den Standard VRML 2.0 machte diesen zur idealen Lösung. VRML Plugins sind mittlerweile für fast jedes Betriebssystem erhältlich, wodurch sich die Portabilität nicht einschränkt. Alternativ wäre eine eigene, nicht portable Anbindung an eine andere Graphikbibliothek wie OpenGL oder Direct3D zu implementieren gewesen.

Wie aus obiger Abbildung ersichtlich, teilt sich der Editor in mehrere Hauptelemente:

- Der größte Bereich umfaßt die Visualisierung des momentan aktiven Knotens. Dabei werden die Bildinformationen der Schlüsselbilder, die er enthält, in chronologischer Reihenfolge von vorne nach hinten auf Rechtecken dargestellt.

Die Schlüsselbilder sind chronologisch geordnet, das vorderste Bild hat die niedrigste Bildnummer im Film. Sie lassen sich so verschieben, daß ein Einblick von verschiedenen Seiten möglich wird.

Neben den Bilddaten selbst werden noch andere Informationen angezeigt: Ein blauer Wimpel an der linken oberen Ecke eines Bildes zeigt an, daß dieses Schlüsselbild einen (nicht leeren) Teilbaum repräsentiert.

Textuelle Annotationen (sog. *Tags*), die für jedes Bild definiert werden können, werden an der rechten oberen Ecke eingeblendet.

- In einer Spalte am linken Rand finden sich eine Reihe von Werkzeugen, die zur Navigation und zur Manipulation im Baum benötigt werden. Der aktuelle Modus wird durch einen roten Pfeil angezeigt, der sich links neben dem eigentlichen Icon befindet.



Auf der „Kompaßrose“ befindet sich ein kleiner roter Kubus. Dieser erlaubt die Modifikation der aktuellen Ansicht, indem das erste Bild analog zur Position dieses Knopfes in Relation zum Mittelpunkt des Kompaßes parallelverschoben wird. Die anderen Bilder verhalten sich ähnlich, doch wird die Bewegung mit einem Faktor so skaliert, daß die Mitte des Bilderstapels an der selben Stelle konstant stehen bleibt, und sich die Bilder die dahinterliegen gegengleich zur Mausbewegung verschieben.



Die Lupe stellt den Standard-Betrachtungsmodus dar. In diesem Modus können Sub-Hierarchien des Baumes betreten werden, indem auf das gewünschte Schlüsselbild geklickt wird. Daraufhin wird die aktuelle Szene durch die neuen Bildinformationen ersetzt.

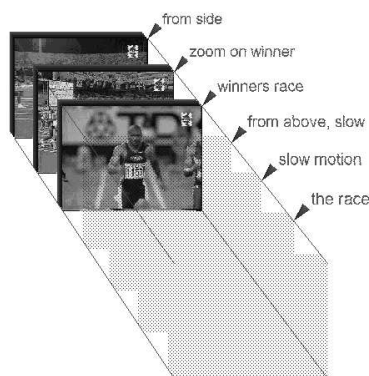


Abbildung 4.5: Transparenz.



Das Transparenzhilfsmittel ermöglicht es, ein weiter hinten liegendes Bild komplett zu sehen, indem die davorliegenden bis auf ein Minimum ausgeblendet werden. Dadurch lassen sich auch räumliche Verhältnisse zwischen Bildern analysieren, selbst wenn diese fast direkt übereinander liegen.



Dies ist der erste einer Reihe von Befehlen, die die Manipulation von Hierarchien erlauben. Durch Auswahl eines Schlüsselbildes wird es selbst und der Teilbaum, den es repräsentiert, eine Ebene tiefer einsortiert.

Um die chronologische Ordnung aufrechtzuerhalten, wird der Teilbaum an das Ende des Sohns seines Vorgängers gestellt (siehe Abbildung 4.6).

Der erste Knoten stellt dabei einen Sonderfall dar, dieser wird an den Anfang des Sohns seines Nachfolgers plaziert.

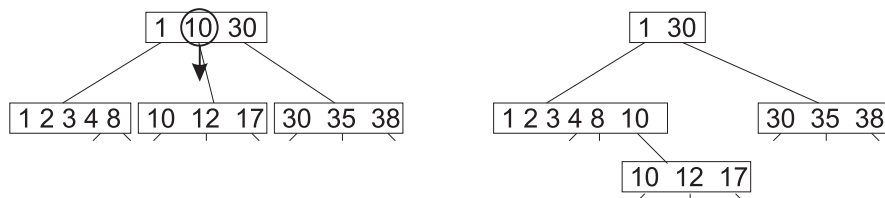


Abbildung 4.6: Absenken des mittleren Teilbaumes.

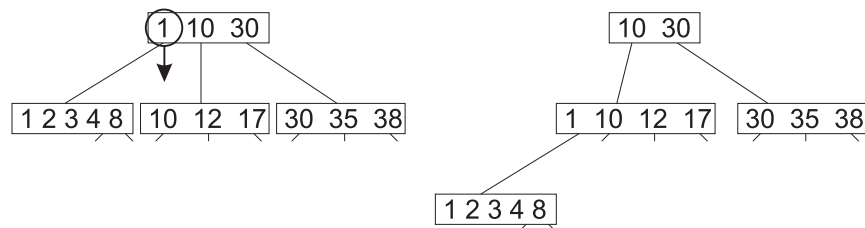


Abbildung 4.7: Absenken des ersten Knotens.



Im Gegensatz zum obigen Befehl dient dieser dazu, einen Teilbaum in die nächsthöhere Hierarchie anzuheben. Dieser wird so einsortiert, daß die temporale Ordnung in der Hierarchie gewahrt bleibt.



In diesem Modus lassen sich Teilbäume aus der Hierarchie entfernen; diese gehen unwiderruflich verloren.

- Das Java-Applet im oberen Teil des Bildes dient dazu, Befehle auszuführen, die den Bedienungsmodus nicht ändern. Hier finden sich Befehle, die die Navigation in und zwischen den Knoten unterstützen.



Dieser Befehl verändert die Anzahl der zugleich angezeigten Bilder, es wird eines aus der aktuellen Szene entfernt.

Es wird dabei nicht nur die Visualisierung verändert, sondern zugleich auch der Teilbaum für dieses Zeitstück neu generiert:

Der Algorithmus aus Abschnitt 4.1.1 wird dazu benutzt, das vom Knoten repräsentierte Stück Video neu zu strukturieren. Dazu wird für den Wert k , der die maximale Anzahl an Elementen pro Knoten angibt, die um eins reduzierte Zahl der Elemente des aktiven Knotens verwendet.

Anders ausgedrückt wird das jeweils unwichtigste Schlüsselbild aus der aktuellen Ansicht entfernt, wodurch Redundanzen verringert werden.



Dies stellt das Pendant zu letztgenanntem Befehl dar, indem ein Element zusätzlich in die Anzeige mit aufgenommen wird. Dazu wird, ganz analog, das höchstbewertete Bild aus den nicht sichtbaren ausgewählt.



Mittels eines externen Filmbetrachters (VPlayer, siehe Abschnitt 4.2.2) läßt sich das momentan angezeigte Teilstück des Videos abspielen.



Der Befehl *up* erlaubt es, aus tiefer gelegenen Hierarchieebenen zur jeweils nächsthöheren zu gelangen. Am Wurzelknoten angelangt ist dieser Menüpunkt inaktiv.



In diesem Eingabefeld ist es möglich einen Skalierungsfaktor anzugeben, der sich auf die Größe der Schlüsselbilder auswirkt. Dieser Faktor geht multiplikativ in die Berechnung ein, 1,0 ist daher der Standardwert.

In unseren Beispielen haben sich Werte zwischen 0,8 und 1,2 als dienlich erwiesen, je nachdem ob das Interesse an der Änderung zwischen den Bildern oder das am Bildinhalt selbst überwog.



Das einzige rein passive Element dient dazu, einen besseren Eindruck von der temporalen Ordnung der Elemente zu vermitteln. Darüberhinaus wird die repräsentierte Zeitspanne im Vergleich zur Gesamtlänge des Videostücks angezeigt.

Oberhalb der graphischen Präsentation dieser Werte werden diese auch numerisch angezeigt, um exakte Daten zu erhalten.



Abbildung 4.8: Zeitskala.

- Die dritte Komponente stellt ein externes Fenster dar (siehe Abbildung 4.9), das zusätzliche Editierfunktionen enthält und *Toolkit*-Fenster genannt wurde. Unterhalb des Menüs, das weiter unten beschrieben wird, befindet sich nochmals eine Darstellung der Zeitachse, wie sie schon weiter oben im Applet vorgestellt wurde.

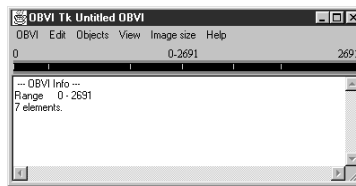


Abbildung 4.9: Toolkit-Fenster.

Dadurch bleibt ein Eindruck des zeitlichen Aufbaus des momentan dargestellten Knotens erhalten, selbst wenn das Fenster die Anzeige verdeckt.

Darunter findet sich ein Status-Fenster, in dem Informationen über den aktuellen Knoten sowie eventuelle Fehlermeldungen angezeigt werden.

Das Menü bietet folgende Befehle:

File: Dieser Menüpunkt dient dazu, OBVIs zu erzeugen, zu öffnen und zu speichern.

Mit dem Punkt *New* kann mittels eines Datei-Öffnen-Dialogs eine Film-Datei ausgewählt werden. Daraufhin wird für diese Datei ein neuer Movie-Descriptor erzeugt, und der Benutzer nach der Position der Bewertungsdatei gefragt. Nachdem diese eingelesen wurde, wird durch die Eingabe der gewünschten Elemente pro Knoten ein neues OBVI erzeugt und angezeigt.

Edit: Dieser Punkt erlaubt einerseits die Erzeugung neuer OBVIs, andererseits gestattet er die Manipulation von Zusatzinformationen.

New OBVI: Ein neues OBVI wird für das aktuelle Video erzeugt. Dazu bekommt der Benutzer einen Dialog präsentiert, in dem er die Zahl der Elemente pro Knoten einstellen kann.

Search: Mit *Search* können die Annotationen der Bilder nach Stichworten (über eine Teilstring-Suche) durchsucht werden. Die Menge der gefundenen Treffer wird in einer Liste angezeigt. Die Auswahl eines der Treffer läßt die Anzeige direkt zu diesem Element springen.

Change OBVI Title: Jedes OBVI kann mit einem Titel versehen werden, der in der Titelleiste des Toolkit-Fensters ausgegeben wird.

Change Level Info: Zu jedem Knoten kann zusätzlich ein Stück Text gespeichert werden, das Informationen über seinen Inhalt beherbergen kann. Diese Information wird im Textfeld des Fensters ausgegeben.

Change Level Audio: Einem Knoten kann eine Tonsequenz zugewiesen werden, die über einen Play-Knopf abspielbar ist.

Objects: Das Objektmenü ermöglicht die Einbindung eines Objekt-Extraktors, der momentan noch nicht verfügbar ist. Entsprechende Daten müssen daher im Moment noch manuell erzeugt und in die entsprechenden Dateien eingetragen werden.

Alle momentan sichtbaren Objekte werden in diesem Menü angezeigt. Wählt man eines aus diesen aus, so wird an den Stellen, an denen dieses Objekt auftritt, ein alternatives Bild anstelle des normalen gezeigt: Das Objekt wird alleine gezeichnet, indem der umliegende Hintergrund transparent dargestellt wird.

View: Im Menü *View* werden alle momentan sichtbaren Bilder angezeigt, entweder mit ihrer Bildnummer oder, sofern vorhanden, durch den Text ihrer Annotation.

Die Auswahl eines dieser Bilder öffnet ein neues Browser-Fenster, in dem das Bild groß angezeigt wird.

Help: Bei Auswahl des Punktes *About* öffnet einen Dialog mit Informationen über Autor und Version des Programms.

Über dieses Menü kann in Zukunft ein Hilfesystem angesprochen werden.

4.2.2 Architektur des Editors

Wie oben schon beschrieben, gliedert sich der Editor in mehrere, mehr oder weniger autark agierende Programmteile. Neben einem Applet in der HTML-Seite läuft in einem anderen Java-Interpreter der Code der VRML-Welt. Zusätzlich existieren noch einige externe Hilfsprogramme, die bei Bedarf aufgerufen werden.

Applet

Die Einbindung eines Java-Applets ermöglicht es, das Programm mit verschiedenen Optionen zu starten, indem diese in der HTML-Seite als Parameter angegeben werden:

movieurl: Spezifiziert die Filmdatei, bzw. den zugehörigen Movie-Descriptor, der geladen werden soll.

tempdir: Gibt das temporäre Verzeichnis an, in dem extrahierte Bilddaten abgelegt werden sollen.

obvnr: Nummer des OBVIs, das geladen werden soll.

autoload: Dieser Parameter gibt an, ob das OBVI sofort nach dem Laden aller Programmdateien eingelesen und aufgebaut werden soll.

Die Kommunikation zwischen dem Applet und dem Programm, das vom Java-Interpreter im VRML-Fenster ausgeführt wird, erfolgt ausschließlich über Ereignisse. Um hier möglichst große Flexibilität zu gewährleisten, wurde ein Typ gewählt, der eine beliebige Anzahl von Strings übermitteln kann, in denen oben angeführte Parameter versandt werden.

Normalerweise werden die Einzelbilder erst zur Laufzeit extrahiert. Der dadurch entstehende Rechenaufwand kann allerdings vermieden werden, wenn die Dateien schon vorher generiert wurden. Diese können beispielsweise mit der Filmdatei auf einer CD-ROM abgelegt worden sein. In diesem Fall ist es nur notwendig, den Parameter *tempdir* auf dieses Verzeichnis zu setzen, damit das Programm die Bilder findet.

Externe Programme

Der Einsatz externer Hilfsprogramme schränkt die Mobilität des Programmes stark ein: Um Programme aufrufen zu können, aber auch um auf Dateien schreiben zu können, muß der Java-Code im CLASSPATH eingetragen sein, ansonsten gilt er als *unsecure*, also als unsicher, und wird in seinen Manipulationsmöglichkeiten stark eingeschränkt. Das gleiche gilt auch für die Kapselung der benötigten Funktionen in eine Windows Bibliothek über die entsprechende Java-Schnittstelle.

VPlayer: Der VPlayer (Video Player, Abbildung 4.10) wurde im Rahmen dieser Diplomarbeit entwickelt. Er wird vom Editor benutzt, um das Teilstück des Videos, das vom momentan aktiven Knoten repräsentiert wird, abzuspielen. Diese Entwicklung wurde notwendig, da es nicht möglich ist, den normalen Multimedia-Betrachter von Windows auf ein Teilstück eines Videos zu begrenzen.

Die benötigten Parameter werden als Kommandozeilenparameter übergeben:

```
vplayer [-f <from>] [-t <to>] MovieToPlay.avi.
```



Abbildung 4.10: VPlayer.

Die Interaktionsmöglichkeiten des VPlayer bestehen, wie aus Abbildung 4.10 zu ersehen ist, aus der Möglichkeit an den Anfang und an das Ende des Filmstückes zu springen, sowie den Film anzuhalten bzw. wieder fortzufahren.

Extractor: Der Extractor wurde an der Abteilung im Rahmen eines Programmierprojektes entwickelt und dient zur Extraktion von Einzelbildern aus Filmdateien.

Dazu können alle Parameter entweder interaktiv über Dialoge gesetzt werden oder als Kommandozeilenparameter angegeben werden. Letztere Eigenschaft macht sich der Editor zunutze, der dieses Programm so aufruft, daß es unsichtbar bleibt und die benötigten Bilddateien im Hintergrund erzeugt.

Ebendiese Eigenschaft jedoch macht eine Synchronisierung zwischen diesem Hilfsprogramm und dem Editor schwierig: Der Extractor arbeitet nach dem Aufruf im Hintergrund als ein anderer Prozess weiter, über den das aufrufende Programm keine Kontrolle hat.

Dadurch mußte eine Heuristik eingesetzt werden, um sicherzustellen, daß die Bilddaten komplett geschrieben wurden: Wurde das letzte Bild angelegt, kann mit dem Aufbau der Geometrie begonnen werden, da das Einlesen, Dekodieren und Anzeigen der Bilder wesentlich länger dauert als das Speichern eines einzelnen Bildes.

VRML & Co.

Die Interaktion zwischen einem Java-Programm und einer VRML Welt kann auf zwei Arten erfolgen, entweder von außen durch ein Java-Applet, oder man kann den Java-Code direkt in die 3D-Szene als Skript-Knoten einbetten.

Zwischen dem VRML-Plugin und dem umliegenden HTML-Dokument erfolgt die Kommunikation über das sogenannte EAI, das *External Authoring Interface*. Das ist eine Reihe von Java Klassen, die es ermöglicht, die VRML Welt zu manipulieren, erlaubt aber auch das Empfangen von Ereignissen, die innerhalb dieser 3D-Welt generiert wurden. Im allgemeinen besteht jedoch eine schlechte zeitliche Synchronisation zwischen Eintreten des Ereignisses und Empfang durch das externe Programm.

VRML Welten lassen sich durch *Script-Knoten* in ihrer Funktionalität beliebig erweitern. Diese Script-Knoten enthalten dazu frei definierbare *Ein- und Ausgangsereignisse* und *Datenfelder*, die durch das Programm manipuliert werden können. Es existiert keine verbindlich festgelegte Sprache, in der Skript-Knoten implementiert werden müssen, Java und ECMASkript (eine Abart von JavaScript) haben sich aber zum de-facto Standard herauskristallisiert, den fast alle Browser unterstützen.

Ereignisse haben immer einen bestimmten Datentyp, der zur Übermittlung von Nutzdaten genutzt wird. Um komplexere als die angebotenen Grunddaten zu übermitteln, besteht die Möglichkeit, Strukturen auf VRML-Prototypen abzubilden, die lesbare Felder mit den benötigten Daten enthalten. Diese können dann einem Ereignis des Typs SFNode überreicht werden. Das Skript hat dann Zugriff auf die einzelnen Datenfelder mit den üblichen `getExposedField()`-Aufrufen.

Wird ein Ereignis erzeugt und an einen Skript-Knoten weitergeleitet, wird in diesem die Methode `doEvent()` aufgerufen, die als Parameter das Ereignis sowie dessen Zeitstempel übergibt. Darin liegt momentan die größte Einschränkung des verwendeten Ereignismodells: Der Erzeuger des Ereignisses ist dem Empfänger nicht bekannt.

Oft ist es jedoch notwendig den Erzeuger zu kennen, besonders wenn eine unbekannte Anzahl mit einem Ereignis-Eingang verknüpft ist. Es ist nicht möglich, die Schnittstelle eines Skript-Knotens zur Laufzeit zu ändern, um so für jedes erzeugte Element genau einen Eingang zu generieren.

4.3 OBVI-Viewer

Der Viewer ist im wesentlichen eine funktional reduzierte Version des Editors, die für den Einsatz im Internet gedacht ist. Ihr fehlen alle Editiermöglichkeiten, die ein Abspeichern in Dateien nötig machen würden.

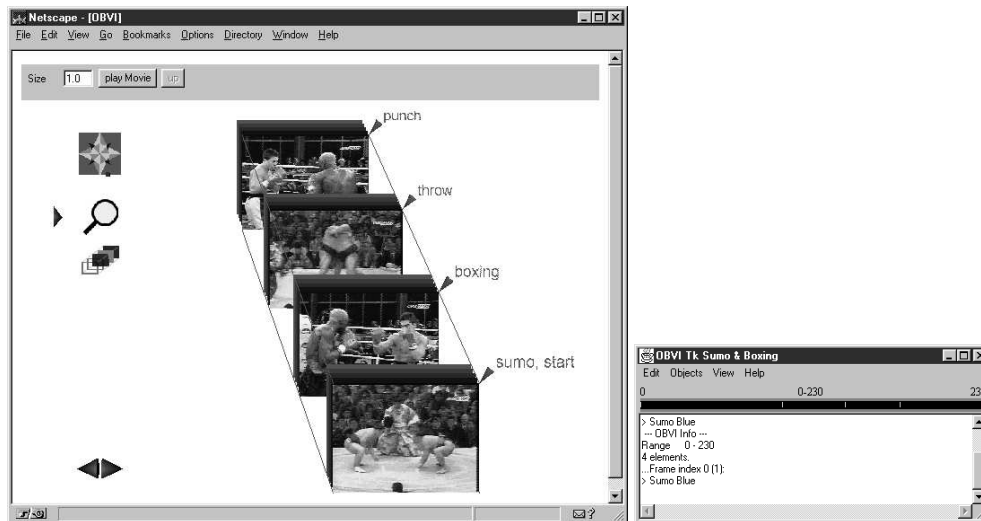


Abbildung 4.11: Viewer Benutzerschnittstelle.

In Abbildung 4.11 ist die Benutzerschnittstelle des Viewers zu sehen. Augenfällig ist dabei der Wegfall der Editierfunktionalität, kenntlich durch die deutlich kleinere Werkzeugpalette am linken Rand.

Beim Editor wurde sichtbar, daß oft der Überblick über den Inhalt von mehr als einem Knoten gleichzeitig von Interesse ist. Aus diesem Grunde wurde die Navigationsstrategie für den Viewer leicht abgewandelt:

Spezielle Knotenelemente, sogenannte *Chunks* (engl. für Klumpen, Brocken), fassen größere Sub-Hierarchien zusammen. Sie sind, wie in Abbildung 4.11 zu sehen, mit mehreren grauen Rechtecken hinterlegt, um auf die semantisch höhere Gewichtung hinzuweisen.

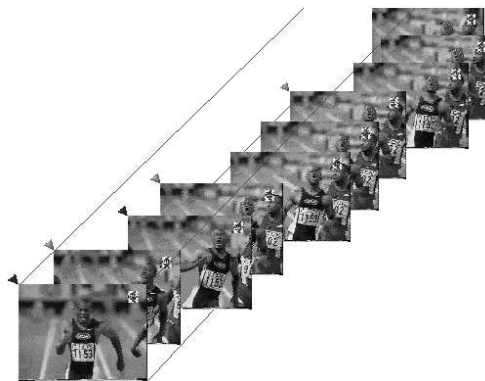


Abbildung 4.12: Viewer mit eingeblendeten Teilbäumen, die geöffneten Schlüsselbilder sind gelb markiert.

Wird einer dieser Chunks ausgewählt, so betritt man den jeweiligen Knoten, analog zum Verhalten im Editor. Die Auswahl normaler Elemente jedoch bewirkt eine andere Reaktion: Anstatt den je-

weiligen Knoten zu betreten, werden seine Elemente in die aktuelle Ebene eingeblendet und so die Informationsdichte erhöht.

4.3.1 Hierarchiegenerierung

Die neue Navigationsstrategie schlägt sich in einem modifizierten Algorithmus zur Hierarchieerzeugung nieder: Anders als in Abschnitt 4.1.1 besprochen, werden Schlüsselbilder aus höheren Knoten in ihren Söhnen nicht mehr verwendet; jedes Bild tritt somit nur einmal im Baum auf.

Um dies zu erreichen wurde der Algorithmus im Bereich der Grenzenberechnung verändert: Jedes Schlüsselbild repräsentiert nun den Bereich, der zwischen ihm selbst und dem nächsten Schlüsselbild seiner Ebene.

4.3.2 Train mode

Die Möglichkeit, Bilder aus unteren Ebenen einzublenden birgt die Gefahr in sich, daß schnell so viele Bilder gleichzeitig sichtbar sind, daß sie zu dicht aufeinander folgen, um den Inhalt noch erkennen zu können.

Um dies zu umgehen kann im Einstellen-Dialog (s.u.) festgelegt werden, wie viele Bilder gleichzeitig angezeigt werden sollen.



Mit dem *Train*-Werkzeug kann der momentan sichtbare Ausschnitt der aktiven Ebene nach vor oder zurück verschoben werden. Die Bilder gleiten dabei solange in die entsprechende Richtung, bis man den gewünschten Ausschnitt erreicht hat.

4.3.3 Datenbank-Schnittstelle

Wie weiter oben schon ausgeführt, wurde das System von vornherein dahingehend konzipiert, eine Videodatenbank für die Datenhaltung einzusetzen.

Die Datenbank besteht im wesentlichen aus einem relationalen Datenbanksystem zur Verwaltung der Metadaten für die Videos sowie einigen Erweiterungen zur Integration der Video-Funktionalität (Abbildung 4.13).

Zur Übertragung der Anfragen sowie der Resultate zwischen Server und Viewer wurde das HTTP-Protokoll gewählt, um die Daten entsprechend zu kapseln. Ein Grund dafür war die Ausnutzung bestehender Technik; so können beispielsweise *Proxy*-Server als Cache für die Bilddaten dienen. Daneben war auch noch ein technischer Grund ausschlaggebend: Die Bilder werden in der VRML-Welt als Verweis auf eine Bild-URL als Textur erzeugt.

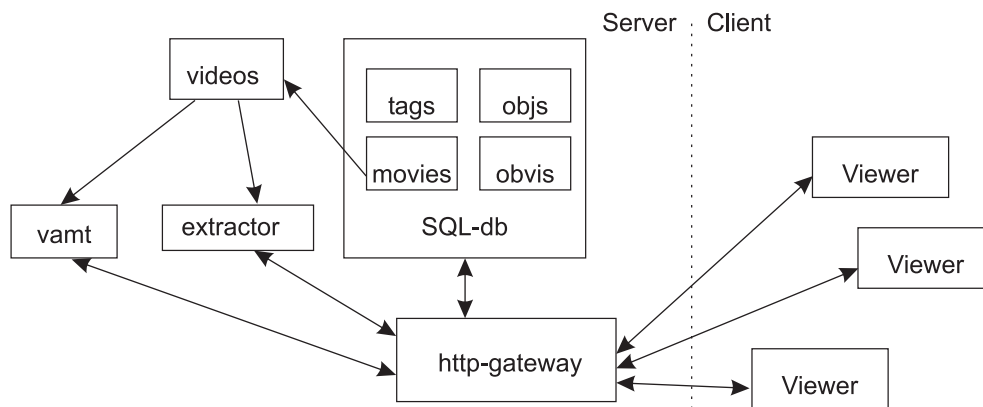


Abbildung 4.13: Aufbau des Video-Servers.

Neben speziellen Abfragen zur Requirierung von Bildern, extrahierten Objekten und Bewertungsdaten kann die Datenbank auch direkt angesprochen werden, indem SQL-Anfragen übermittelt werden.

Der Funktionsumfang der Module „extract“ und „vamt“ entspricht dem der externen Programme **Extractor** und **vamt** des Editors, allerdings existieren diese nicht mehr als selbständige Programme, sondern sind vollständig in den Server integriert.

Datenbank-Editor

Durch die Integration dieser Funktionen war es möglich, den Editor an die Datenbank-Schnittstelle anzupassen, wodurch die Modifikation von OBVIs nicht mehr nur lokal, sondern auch über das Internet hinweg erfolgen kann.

Um neue Videos bearbeiten zu können ist es notwendig, zuerst die entsprechenden Einträge in der Datenbank vorzunehmen, danach können dafür OBVIs erzeugt werden.

Kapitel 5

Zusammenfassung und Ausblick

In Kapitel 4 wurde ein System vorgestellt, das fähig ist, digitale Videos zu indexieren und die daraus gewonnenen Informationen dem Benutzer mit einer neuartigen Schnittstelle zugänglich zu machen.

5.1 Bewertung

Das System eignet sich sowohl für den lokalen Einsatz wie auch zur Präsentation von Videos über das Internet, was durch die Verwendung von Java als Implementierungssprache und durch den Einsatz eines offenen 3D-Standards erreicht wurde.

5.1.1 Laufzeiteffizienz

Die Geschwindigkeit der graphische Darstellung hängt sehr stark von der eingesetzten Hardware ab, die Reaktionszeit und Interaktivität des Systems wird besonders von 3D-Grafik-Hardware beeinflusst. Durch den sparsamen Einsatz von 3D-Elementen ist jedoch auch ohne Spezialausrüstung eine akzeptable Leistung gegeben (in der Größenordnung von 5–10 Bildern pro Sekunde beim Bewegen der Bilder auf einem Pentium mit 166 MHz).

Den größten Zeitanteil benötigt die Generierung der Filmbewertungen, wie schon in Abschnitt 4.1.1 angesprochen wurde. Hierin ist das größte Hindernis für einen künftigen Praxiseinsatz zu sehen.

5.1.2 Portabilität

Dank der Implementierungssprache Java sind die Programme prinzipiell auf allen Rechnersystemen lauffähig, die diese Sprache unterstützen. Neben Java muß der HTML-Browser, in dem die Programme ausgeführt werden, ein VRML-Plugin installiert haben. Dieses muß sowohl die Java-EAI-

Schnittstelle beherrschen, über das das Applet Befehle an die 3D-Szene weiterleitet, als auch Java in VRML-Skript-Knoten unterstützen.

Zur Manipulation der VRML-Geometrie wurden ausschließlich die in der Spezifikation definierten Felder und Methoden verwendet, es wurden keine Hersteller-spezifischen Erweiterungen benutzt. Jedoch muß darauf hingewiesen werden, daß etliche VRML-Plugins die notwendigen Schnittstellen nicht ausreichend unterstützen.

Die einzigen plattformabhängigen Teile sind die Programme `Extractor` und `VPlayer`, die in C++ unter Windows implementiert wurden. Sie können allerdings ohne größeren Aufwand durch andere Programme ersetzt werden (auf Unix-Plattformen ließe sich beispielsweise `mpeg_play` als Filmbeobachter und `mpeg_lib` zum Extrahieren der Einzelbilder einsetzen).

Eine weitere betriebsystemabhängige Komponente ist `vamt`, das durch die Verwendung der Multimedia-Programmierschnittstellen ebenfalls auf Windows angewiesen ist.

5.1.3 Anwendungsbereiche

Der Editor dient zur Erzeugung und Manipulation von OBVIs und der damit verbundenen Daten. Sein Einsatz erfolgt in der Regel lokal und wird zur Vorbereitung von Videos zur Aufnahme in die Datenbank benutzt.

Der Viewer kann dazu benutzt werden, OBVIs in HTML-Seiten einzubinden, und bietet damit eine Anbindung an das WWW. Dadurch können Zusammenfassungen von Videos betrachtet werden, bevor man sich entschließt, das gesichtete Videomaterial auf den eigenen Rechner zu laden. Dies können entweder ganze Filme sein oder auch nur die Teilstücke, für die man sich wirklich interessiert. Dadurch wird ein wesentlich flexiblerer Umgang mit digitalen Videodaten möglich.

Durch die optionale Serveranbindung kann das System sowohl alleine betrieben werden als auch mit einer Datenbank gemeinsam zur Visualisierung größerer Datenbestände genutzt werden. Dies ermöglicht ein großes Spektrum von Einsatzmöglichkeiten, sei es als Vorschau für Video-on-Demand-Systeme, zur Filmsuche beim interaktiven Fernsehen, zum Sichten von Filmmaterial bei verschiedensten Informationsanbietern und Nachrichtenagenturen, als neue Art von Filmtrailern für Kinofilme oder zur Visualisierung von Videos für Medienarchive und -museen.

Die dateibasierte Variante kann dazu eingesetzt werden, auf Datenträgern wie CD-ROMs, DVDs oder anderen eine Vorschau der abgespeicherten Videos zu bieten. Dadurch lassen sich größere Mengen an Videodaten bequem archivieren und später sichten, ohne sich die jeweils die kompletten Videos ansehen zu müssen.

5.2 Ausblick

Das im Rahmen dieser Diplomarbeit implementierte System zum Erstellen, Editieren und Betrachten von OBVIs stellt einen Kern für zukünftige Erweiterungen dar.

Die Implementierung eines eigenen Bewertungsprogramms ist ein vorrangiges Ziel: Der Großteil der Zeit, der für die Vorbereitung von Videos aufgewendet werden mußte, wurde für die Berechnung des Rankings verbraucht. Vorrangig sollte es daher sein, neue Bewertungsalgorithmen in Verbindung mit einer effizienten Implementierung zu testen, um das System auch für längere Videosequenzen nutzen zu können. Um die Plattformunabhängigkeit zu fördern erscheint es sinnvoll, dabei auf ein anderes Videoformat als es `vamt` benutzt umzusteigen: MPEG bietet sich als Lösung an, da es genormt und daher plattformunabhängig ist. Als offener Standard ist es darüberhinaus eines der am besten erforschten Videoformate, wodurch eine Vielfalt von effektiven Bewertungsalgorithmen zur Verfügung steht. Insbesondere lassen sich jene Verfahren einsetzen, die direkt mit den komprimierten Videodaten arbeiten.

Die Erweiterung der Tag-Informationen um einen URL-Link-Typ sollte in einer noch engeren Verknüpfung von OBVI und WWW resultieren. Dadurch lassen sich beliebige im Internet verfügbare Datenquellen als Zusatzinformation nutzen.

Die Extraktion von Objekten ist im System zwar vorgesehen, bisher existiert allerdings noch keine Möglichkeit, interaktiv freigestellte Objekte zu generieren und ins OBVI zu integrieren. Ein Objekt-Extraktor der den Autor bei der Erstellung dieser Objekte unterstützt ist ein notwendiges Erweiterungsmodul. Die Bewegungsinformation in Videos könnte hierbei dazu eingesetzt werden, Voraussagen für die Objektposition in künftigen Bildern abzuschätzen.

Ein allgemeiner Ansatz zur Speicherung beliebiger Zusatzinformationen zu den Schlüsselbildern wäre denkbar. Dies ließe sich beispielsweise durch Server-seitig gespeicherte Java-Klassen realisieren, die über Beschreibungstabellen geladen werden könnten, und durch einen Zugang zur VRML-Geometrie zusätzliche Interaktionsmöglichkeiten bieten könnten.

Um das System für den Heimanwender sinnvoll nutzbar zu machen, ist es notwendig die einzelnen Arbeitsschritte so weit wie möglich miteinander zu verschmelzen und zu automatisieren. Besonders die Einbindung der Bewertungs-Berechnung in den Editor sollte eine wesentliche Vereinfachung bringen.

Anhang A

Dateiformate

In diesem Abschnitt finden sich die formalen Spezifikationen für das verwendete Dateiformat, mit dem die Film-Metainformationen auf Datenträger abgelegt sind.

Die Daten werden textuell gespeichert, woraus sich zwar Performance-Einbußen ergeben, diese stellen jedoch einen vertretbaren Nachteil dar, da sich dadurch eine Plattform- und insbesondere Hardware-unabhängigkeit ergibt.

Eine Erweiterung um ein Binärformat, bei dem die Serialisierbarkeit von Java-Klassen ausgenutzt wird, ist denkbar und über Ausnutzung der Versionsnummer (Abschnitt A.1) der Dateien einfach zu realisieren.

A.1 Allgemeine Konventionen

Alle Dateien, die vom Editor erzeugt wurden, beginnen mit einer Kopfzeile ähnlich derer von Shell-Skripten, und anderen. Sie dient neben der besseren Identifizierbarkeit der Daten der Speicherung von Zusatzinformationen.

```
#OBVI MOVIE 1.0
```

Die Kopfzeile hat das Format „#OBVI <Dateityp> <Versionsnummer>“. Dadurch können die unterschiedlichen Daten sofort an der Kopfzeile erkannt werden, was für Mensch und Maschine die Verifizierung erleichtert.

Durch die Versionierung der Dateien ist es möglich, sich die Option auf spätere Änderungen im Dateiformat freizuhalten, man muß lediglich die Versionsnummer ändern und im Code entsprechend darauf reagieren.

Die verwendeten Sprachelemente sind lose an das Format von VRML-Dateien angelehnt, so werden beispielsweise Array-Elemente in eckigen Klammern ("[","]") zusammengefaßt.

Als Kommentarzeichen dient die Raute („#“), die einen Kommentar bis zum Zeilenende beginnt. Beim Speichern von Daten mit dem Editor gehen von Hand eingefügte Kommentare allerdings verloren, daher sollte wenn möglich darauf verzichtet werden, von Hand Kommentare einzufügen.

Die einzelnen Datenelemente müssen voneinander durch Leerräume getrennt sein, die aus einer beliebigen Menge von Leerzeichen, Zeilenumbrüchen und Kommas bestehen können.

Die Schlüsselwörter können beliebig groß oder klein geschrieben werden, sie werden allerdings vom Programm beim Speichern groß geschrieben.

A.1.1 Definitionen

In den folgenden Kapiteln werden die Dateistrukturen in EBNF vorgestellt. Dabei werden Terminalsymbole in einer nichtproportionalen Schrift und in Hochkommas dargestellt (z.B. 'MOVIE').

Vordefinierte Datentypen werden fett gedruckt (**string**).

A.2 Movie-Descriptor

Wie schon in Abschnitt 4.1 ausgeführt, stellt der *Movie-Descriptor* das Herz der Datenhaltung dar. Alle erforderlichen Informationen sind in ihm enthalten.

```

MovieDesc = Movie [ Objects ] [ OBVIs ]
Movie     = 'MOVIE' '{ { Film | Ranks | Tags | Frames } }'
Film      = 'FILM' url
Ranks     = 'RANKS' url
Tags      = 'TAGS' url
Frames    = 'FRAMES' number
Objects   = 'OBJECTS' '[' { string } ']'
OBVIs     = 'OBVIS' '[' { string } ']'

```

Im *Movie* sind alle Informationen gespeichert, die global für den gesamten Film gelten, dieser Block muß daher immer vorhanden sein. Die beiden anderen dagegen sind optional, wenn auch der Teil mit den OBVIs fast immer zu finden sein wird.

In den Sektionen Objects und OBVIs sind Verweise auf die jeweiligen Definitionsdateien gespeichert, die in den folgenden Abschnitten erklärt werden.

A.3 Tags: Annotationen

```

ObviDesc = { TagItem }
TagItem  = number( TagName | TagLink )
TagName  = 'NAME' string
TagType  = 'LINK' url

```

A.4 Objekt-Dateien

```

ObjDesc  = { Title | Info } Frames
Title    = 'TITLE' string
Info     = 'INFO' string
Frames   = '[' { FrameData } ']'
FrameData = number url

```

A.5 OBVIS

```

ObviDesc = { Title | Info | Frame } ObviNode
Title    = 'TITLE' string
Info     = 'INFO' string
Frame    = 'FRAME' number
ObviNode = '{' { NodeTitle | NodeInfo | Audio | Range } '}'
NodeTitle = 'TITLE' string
NodeInfo  = 'INFO' string
Audio     = 'AUDIO' url
Range     = 'RANGE' number number
Frames    = 'FRAMES' '[' { number } ']'
Children  = 'CHILDREN' '[' { ObviNode } ']'

```

Die Baumhierarchie des OBVIS findet sich hier im ObviNode wieder. Zur Gewährleistung eines korrekten Baumes gelten folgende semantische Beziehungen:

1. Leere Knoten terminieren die Hierarchie ('{ }').
2. Die Zahl und Identität der Elemente ist bekannt (d.h. Frames wurden gelesen), bevor die Kinder bearbeitet werden.
3. Das Kind steht an der selben Index-Position wie sein zugehöriges Schlüsselbild.
4. Leere Knoten am Ende können weggelassen werden.

Anhang B

URL-Verzeichnis

Im folgenden sind die wichtigsten Quellen als URLs im Internet zu den einzelnen Themengebieten angegeben.

Aufgrund der Lebendigkeit des Netzes kann für die Richtigkeit der Daten zum Lese-Zeitpunkt nicht garantiert werden, es wurden jedoch alle im April 1998 geprüft.

Sollte trotzdem ein Link nicht mehr aktiv sein, so sei auf die einschlägigen Suchmaschinen verwiesen, die hier gute Dienste leisten.

B.1 Video-Indexing Projekte

WebSEEK <http://disney.ctr.columbia.edu/WebSEEk/WebSEEk.html>
Bild- und Videoverzeichnis des WWW – siehe Abschnitt 3.3.

SaFe <http://disney.ctr.columbia.edu/SaFE/SaFE.html>
SaFe ist ein Werkzeug zur Bildsuche nach räumlichen oder Merkmalskriterien wie Farbe u.a.

WebClip <http://disney.ctr.columbia.edu/cveps>
Der auf Basis von CVEPS entwickelte MPEG-Editor.

VideoQ <http://www.ctr.columbia.edu/videoq>
Video Query ermöglicht die Suche in einer Videodatenbank.

SWIM <http://www.iss.nus.sg/RND/MS/Projects/vc/project1.html>
Show What I Mean - Der Browser der University of Singapore.

MoCA <http://www.informatik.uni-mannheim.de/informatik/pi4/projects/MoCA/>
Informationen zum *MoCA*-Projekt (Movie Content Analysis), ein Teilprojekt ist *VAbstract* (Abschnitt 3.2).

QBIC <http://www.qbic.almaden.ibm.com/>
Die Bilddatenbank von IBM (Query By Image Content).

B.2 Forschungsgruppen

Advent <http://www.ctr.columbia.edu/advent/>

Forschungsgruppe der Univ. of Columbia, die mit einer Anzahl an Online-Demos glänzt (s.o.).

VCCL <http://marge.genie.uottawa.ca/vccl/introduction.html>

Das *Visual Computing and Communications Laboratory* der Univ. Ottawa.

VisLab <http://calvin.cse.psu.edu/>

Forschungsgruppe an der Pennsylvania State University.

Almaden http://www-i.almaden.ibm.com/cs/video/video_anno_ext.html

Die Video-Analyse-Gruppe von IBM, Berechnung von Panorama-Aufnahmen aus Kamera-schwenks.

Interactive Cinema <http://highland.media.mit.edu/users/cerebus/ICWWW/>

Interactive Cinema Group, eine Forschungsgruppe am MIT Medialab.

ICV <http://www.icv.ac.il/>

Die Israelische Computer Vision Homepage bietet Links zu allen wichtigen israelischen Forschungsgruppen dieses Themas sowie zu anderen verwandten Ressourcen.

Computer Vision <http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html>

Die Computer Vision Seite der Carnegie Mellon University.

Video Classification <http://www.iss.nus.sg/RND/MS/Projects/vc/>

Die Video Classification Group an der University of Singapore.

B.3 Sonstiges

Olive <http://www.otal.umd.edu/Olive/>

Online-Bibliothek der Informations-Visualisierung.

CS Bibliographie <http://liinwww.ira.uka.de/bibliography/index.html>

The Collection of Computer Science Bibliographies, eine umfassende Literaturdatenbank.

Bibliographie <http://www.icv.ac.il/DataBases/biblio/bibliography/contents.html>

Eine umfassende Bibliographie zum Thema Computer Vision.

Anhang C

Aufnahmen der Benutzerschnittstelle

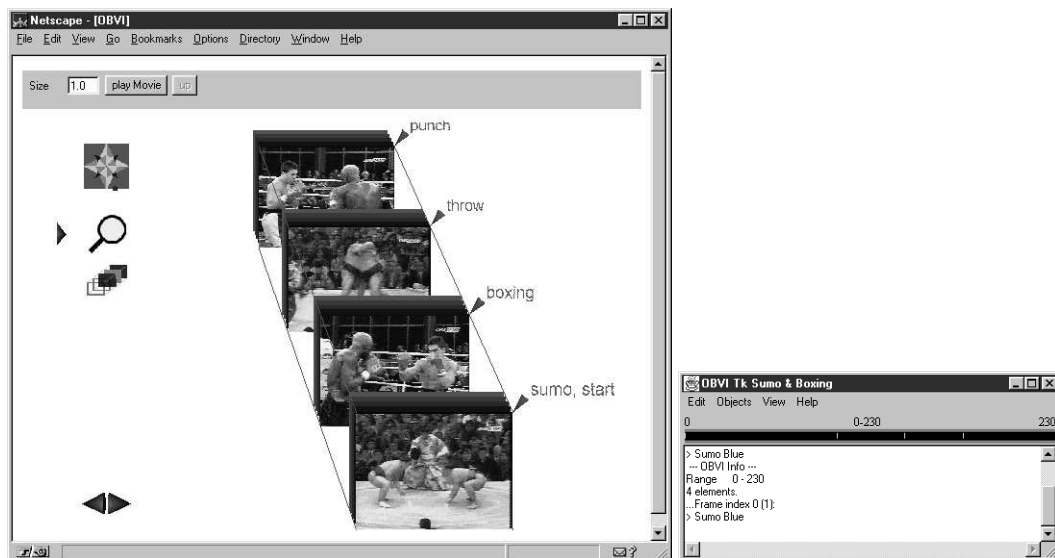


Abbildung C.1: Video „mix“, Hauptübersicht.

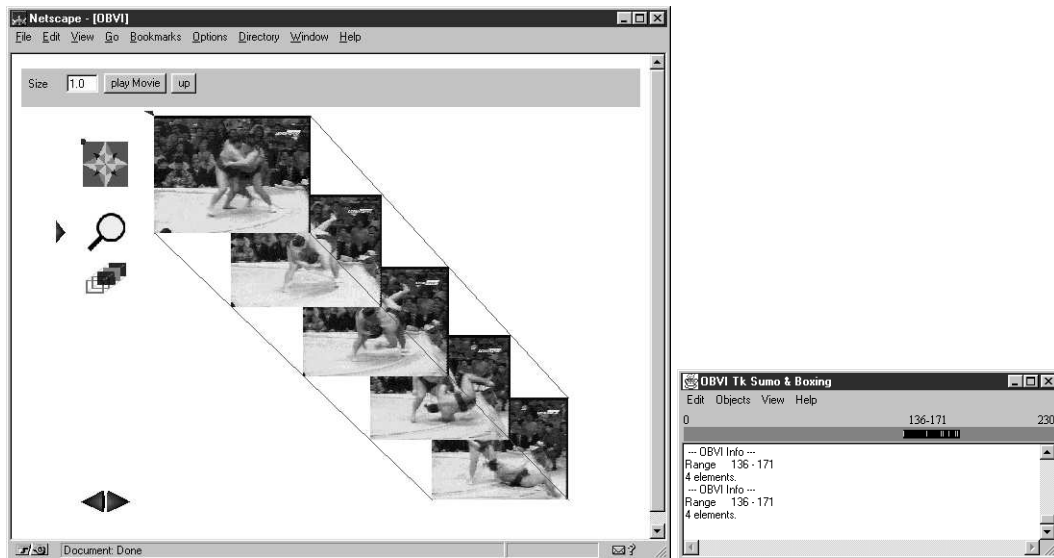


Abbildung C.2: Video „mix“, die zweite Sumo-Sequenz mit Wurf.

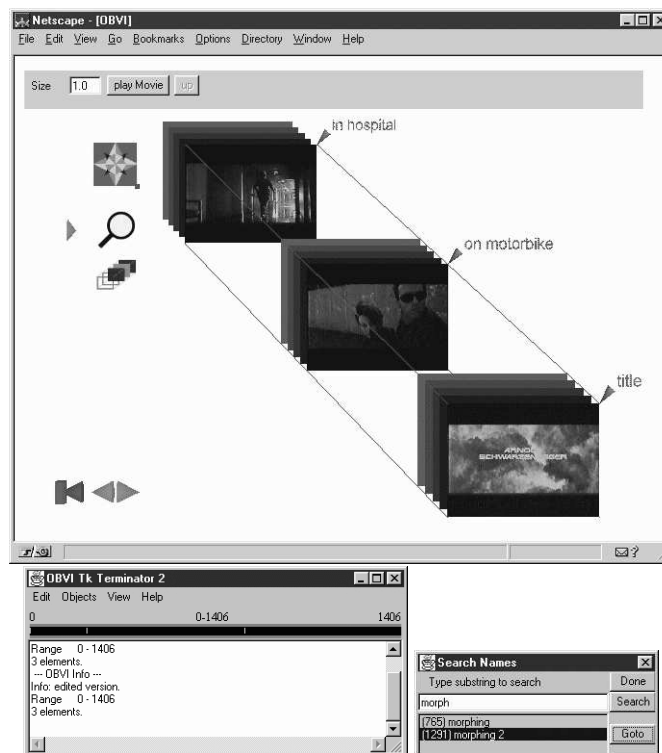


Abbildung C.3: Video „Terminator2“, Hauptübersicht, Stichwortsuche.

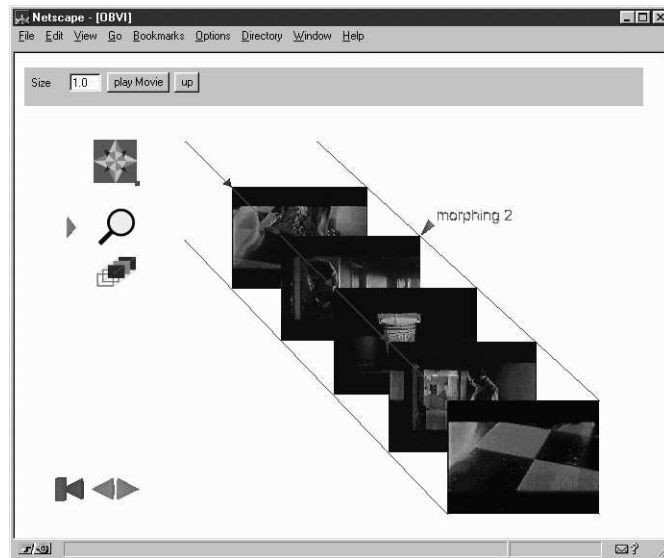


Abbildung C.4: Video „Terminator2“, Eingang zur Morphingsequenz.

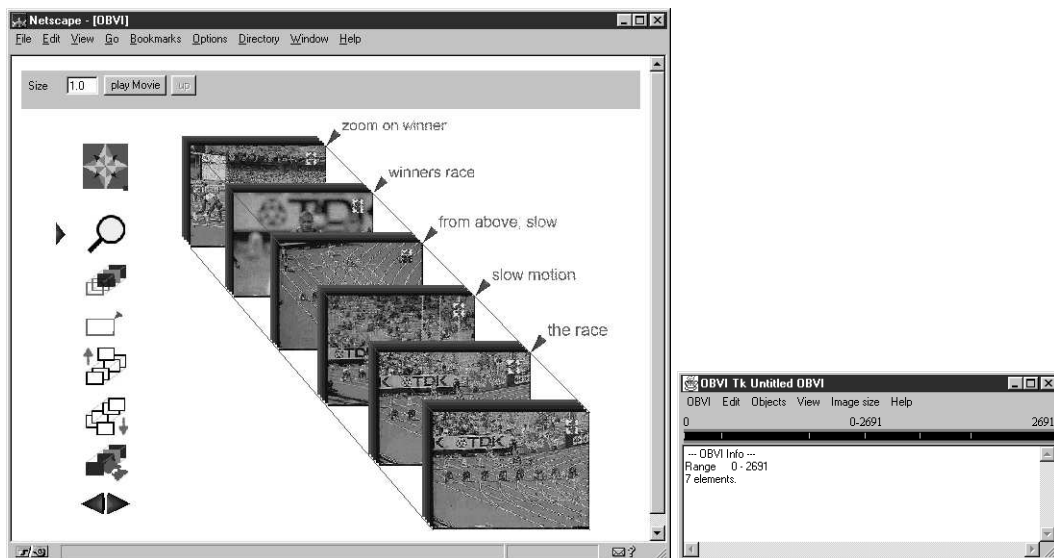


Abbildung C.5: Video „Hundred“, Hauptübersicht.

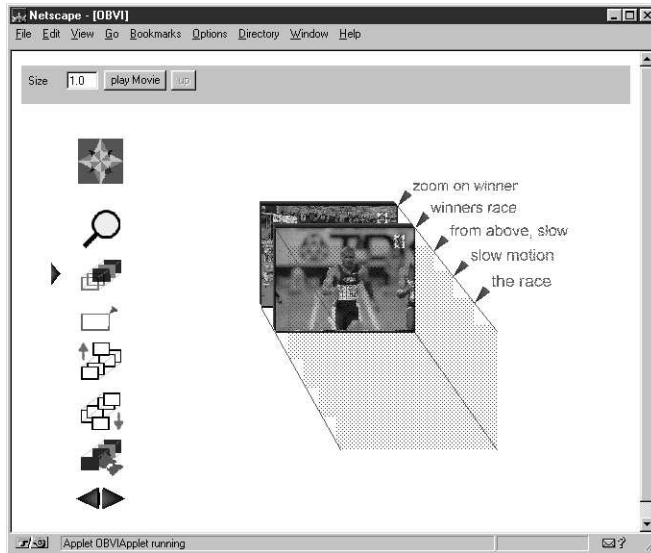


Abbildung C.6: Video „Hundred“, Transparenz.

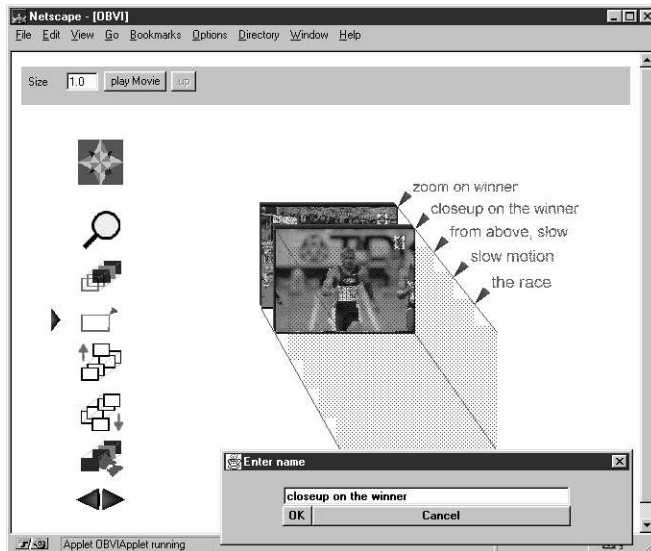


Abbildung C.7: Video „Hundred“, Änderung der Annotation.

Anhang D

Literatur

- [ADHC94] F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu. **Content-Based Browsing of Video Sequences**. In Proceedings of the Second ACM International Conference on Multimedia (MULTIMEDIA '94), pages 97–104, New York, October 1994. ACM Press.
- [DM97] W. Ding and G. Marchionini. **A Study on Video Browsing Strategies**. Technical Report CS-TR-3790, CLIS-TR-97-06, University of Maryland, College Park, August 1997.
- [DMT97] W. Ding, G. Marchionini, and T. Tse. **Previewing Video Data: Browsing Key Frames at High Rates Using a Video Slide Show Interface**. In Proceedings of the International Symposium on Research, Development & Practice in Digital Libraries (ISDL'97), pages 425–426, nov 1997.
- [GOK⁺95] U. Gargi, S. Oswald, D. Kosiba, S. Devadiga, and R. Kasturi. **Evaluation of video sequence indexing and hierarchical video indexing**. In Proc. SPIE: Storage Retrieval image Video Databases III, 1995.
- [HKM⁺97] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. **Image Indexing Using Color Correlograms**. In IEEE Computer Vision and Pattern Recognition Conference, page B7: Video and image database indexing, 1997.
- [HLMS95] R. Hjelsvold, S. Landorgen, R. Midtstraum, and O. Sandstaa. **Integrated Video Archive Tools**. In Proceedings of the ACM Multimedia'95, pages 283–293, San Francisco, California, November 1995.
- [HMS95] R. Hjelsvold, R. Midtstraum, and O. Sandstaa. **A Temporal Foundation of Video Databases**. In J. Clifford and A. Tuzhilin, editors, Recent Advances in Temporal Databases. Proceedings of the International Workshop on Temporal Databases, Zurich, Switzerland, September 1995. Springer Verlag.
- [IP97] F. Idris and S. Panchanathan. **Review of Image and Video Indexing Techniques**. Journal of Visual Communication and Image Representation, 8(2):146–166, Jun 1997.
- [KDL96] V. Kobla, D. Doermann, and K.-I. Lin. **Archiving, indexing, and retrieval of video in the compressed domain**. In SPIE Conference on Multimedia Storage and Archiving Systems, volume 2916, pages 78–89, 1996.

- [KDR96] V. Kobla, D. Doermann, and A. Rosenfeld. **Compressed Domain Video Segmentation**. Technical Report CAR-TR-839, CS-TR-2688, Center for Automation Research, University of Maryland, 1996.
- [KSGA96] R. Kasturi, S. H. Strayer, U. Gargi, and S. Antani. **An Evaluation of Color Histogram Based Methods in Video Indexing**. Technical Report CSE-96-053, Dept. of Computer Science and Engineering, The Pennsylvania State University, 1996.
- [LPE97] R. Lienhart, S. Pfeiffer, and W. Effelsberg. **Video abstracting**. *Communications of the ACM*, 40(12):54–62, December 1997.
- [MC96] J. Meng and S.-F. Chang. **CVEPS—A Compressed Video Editing and Parsing System**. In *Proceedings of ACM Multimedia 96*, pages 43–53, 1996.
- [MTCM96] P. J. Macer, P. J. Thomas, N. Chalab, and J. F. Meech. **Finding the cut of the wrong trousers: Fast video search using automatic storyboard generation**. In *Proceedings of the CHI'96*, 1996.
- [SAG95] H. S. Sawhney, S. Ayer, and M. Gorkani. **Model-based 2D&3D Dominant Motion Estimation for Mosaicing and Video Representation**. Technical report, IBM Almaden Research Center, 1995.
- [SC96] J. R. Smith and S.-F. Chang. **Searching for Images and Videos on the World-Wide Web**. Technical Report 459-96-25, Center for Telecommunication Research, Columbia University NY, 1996.
- [SN95] R. Steinmetz and K. Nehrstedt. *Multimedia: computing, communications, and applications*. Prentice Hall, 1995.
- [TAOS93] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata. **VideoMAP and VideoSpaceIcon: Tools for Anatomizing Video Content**. In *Proceedings of the InterCHI'93*, pages 131–136, 1993.
- [TAT97] Y. Taniguchi, A. Akutsu, and Y. Tonomura. **PanoramaExcerpts: Extracting and Packing Panoramas for Video Browsing**. In *Proceedings, IS&T/SPIE Multimedia Computing and Networking*, pages 427–436, 1997.
- [TM96] L. M. L. Teixeira and M.I. Martins. **Video compression: The MPEG Standards**. In *Proceedings of the ECMAST 96 (European Conference on Multimedia Applications, Services and Techniques)*, pt. II, page 615, 1996.
- [YL95a] B.-L. Yeo and B. Liu. **On the Extraction of DC Sequence from MPEG Compressed Video**. In *Proceedings of the International Conference on Image Processing*, 1995.
- [YL95b] M. Yeung and B. Liu. **Efficient Matching and Clustering of Video Shots**. In *Proceedings of the International Conference on Image Processing*, pages 338–341, 1995.
- [YY97] B.-L. Yeo and M. M. Yeung. **Retrieving and visualizing video**. *Communications of the ACM*, 40(12):43–52, December 1997.
- [YYL96] M. Yeung, B.-L. Yeo, and B. Liu. **Extracting Story Units from Long Programs for Video Browsing and Navigation**. In *Proceedings of the International Conference on Multimedia Computing and Systems*, 1996.

-
- [ZLSZ96] H. J. Zhang, C. Y. Low, S. W. Smoliar, and D. Zhong. **Video parsing, Retrieval and Browsing: An Integrated and Content-Based Solution.** In The Third ACM International Multimedia Conference and Exhibition (MULTIMEDIA '95), pages 15–24, New York, November 1996. ACM Press.
- [ZZC95] D. Zhong, H. Zhang, and S.-F. Chang. **Clustering Methods for Video Browsing and Annotation.** Technical report, Institute of System Science, National Univ. of Singapore, 1995.