Seminar

Digitales Video

Universität Dortmund
Sommersemester 1999

Betreuer:
Prof. Dr. Gisbert Dittrich
Dipl.-Inform. Jörg Westbomke

Teilnehmer:
Markus Bajohr
Dirk Denninghoff
Bernhard Flechtker
Karsten Frommolt
Marc Iwan
Christian Klaas
Mike Ndambuki
Paul Neubach
Helge Schneider
Rüdiger Schwarz

Vorwort

Elektronische Dokumente werden immer häufiger durch zeitbasierte Medienobjekte wie z.B. Videoclips angereichert. Selbst im WWW sind mittlerweile Videos komfortabel durch Streamingtechnologie verfügbar. Rechnergestützte Videoschnittlösungen und Telekonferenzsysteme sind zudem weitere Einsatzfelder für digitale Videosequenzen. Aufgrund beschränkter Ressourcen ist Kompression aber weiterhin ein entscheidender Einsatzfaktor.

Dieses Seminar sollte Studierenden im Hauptstudium des Fachbereichs Informatik einen Überblick über die grundlegenden Ansätze zur Kompression von Videos/Videoclips geben und Einsatzmöglichkeiten in der Praxis aufzeigen.

Dabei sollten die folgenden Themenschwerpunkte behandelt werden:

Nach einer Einführung in die Grundlagen der digitalen Bilddarstellungen, sowie der grundlegenden Bildkompressionsalgorithmen für Einzelbilder wurde die Kompression von Bewegtbildern vorgestellt. Dazu wurde der zentrale Algorithmus MPEG in seinen verschiedenen Spezifikationsversionen erläutert. Ergänzt wurde diese Darstellung um die Betrachtung der Standards für Videokonferenzen. Die Praxisrelevanz solcher Standards wurde durch vergleichende Analyse der Algorithmen und Vorstellung von existierenden Softwareimplementierungen berücksichtigt.

Dortmund, den 7. Juni 1999

G. Dittrich J. Westbomke

Inhaltsverzeichnis

Rüdiger Schwarz::
„Grundlagen der Datenkompression“.....1

Dirk Denninghoff:
„Kompressionstechniken für Einzelbilder“.....21

Mike Ndambuki:
„JPEG Kompression“.....41

Karsten Frommolt:
„MPEG-1 und MPEG-2“.....59

Marc Iwan:
„MPEG-4 und MPEG-7“.....79

Christian Klass:
„Standards für Videokonferenzen“.....97

Paul Neubach / Helge Schneider:
„Vergleichende experimentelle Betrachtungen“.....115

Markus Bajohr:
„DV Formate“.....153

Bernhard Flechtker:
„Quicktime 3/4“.....177

Grundlagen der Datenkompression

Rüdiger Schwarz

FB Informatik, Uni Dortmund
Email-Adresse : schwarz@platane.cs.uni-dortmund.de

Zusammenfassung

Dieses Dokument basiert hauptsächlich auf den Kapiteln 1 bis 3 aus dem Buch „Video Compression Techniques“ (vgl. Literaturangabe [EfSt 98]). Das Thema dieser Arbeit sind die Grundlagen der Datenkompression. Am Anfang wird der Sinn der Datenkompression motiviert an Hand von zahlreichen Beispielen. Insbesondere wird an dieser Stelle dargestellt, wieviel Speicherplatz unterschiedliche unkomprimierte Medien benötigen. Dann erfolgt eine Betrachtung der Schnittstelle zwischen Anwendungen und Kompressionsverfahren. Interessant ist hierbei, welche Anforderungen stellt eine Anwendung, und welche Auswirkungen ergeben sich somit für die Kompressionsverfahren. Im Anschluß daran, wird ein allgemeines Grundwissen über die verschiedenen Kompressionstechniken erläutert. Hierzu gehört die Darstellung der verschiedenen Kategorien der Kompressionstechniken, wie auch eine Veranschaulichung der prinzipiellen Schritte der Datenkompression. Im nächsten Kapitel erfolgt dann eine ausführliche Vorstellung einer Auswahl der wichtigsten Kompressionstechniken. Es beginnt mit der Vorstellung des Verfahrens der Lauflängenkodierung und deren wichtigsten Variationen. Das nächste Unterkapitel beschreibt unter anderem mit Hilfe von einfachen Beispielen Static Pattern Substitution und Diatomic Encoding. Unter dem Gesichtspunkt der Kompression und Dekompression wird danach das Verfahren Lempel-Ziv Encoding untersucht, das ein Dynamic Pattern Substitution Algorithmus darstellt. Weiterhin werden dann die beiden Verfahren Huffman- Kodierung und arithmetische Kodierung analysiert, die die Technik Variable-Lenght Encoding benutzen. Bei der Vektorquantisierung werden die grundlegenden Ideen aufgezeigt. Die verlustbehafteten Verfahren Transformationskodierung wie auch die Subband-Kodierung werden daran anschließend präsentiert. Die Interpolation wird dann mit Hilfe von dem YUV- Farbmodell erklärt. Weiter geht es mit den wichtigen allgemeinen Verfahren der relativen Kodierung und der adaptiven Kompression. Abschließend erfolgt dann noch eine kleine Vorstellung weiterer Verfahren.

1 Einleitung

Unkomprimierte Grafiken, Audio- und Videodaten erfordern eine beträchtliche Speicherkapazität, die im Falle unkomprimierter Videodaten nicht einmal mit der heutigen CD- oder DVD-Technologie erreichbar ist. Ein Datentransfer unkomprimierter Videodaten über digitale Netze erfordert eine sehr hohe Bandbreite für die jeweiligen Punkt-zu-Punkt Verbindungen. Diese Verbindungen währen dann mit erheblichen Kosten verbunden.

1.1 Anforderungen an den Speicherplatz

Bilder haben im allgemeinen beträchtlich höhere Anforderungen an den Speicherplatz als ein normaler Text. Für Audio und Video muß eine große Menge an Speicherplatz zur Verfügung stehen, und für die Übertragung muß eine ausreichende Datenrate gewährleistet werden. In diesem Kapitel soll der quantitative Übergang von einfachen Text zu Videodaten verdeutlicht werden, und hieraus leitet sich dann die Notwendigkeit der Kompression ab. Um die unterschiedlichen unkomprimierten visuellen Medien (Text, Grafik , Bild) vergleichen zu können, basieren die folgenden Angaben auf einer Fenstergröße von 640·480 Bildpunkten.

i.) Text : Zur Darstellung von dem Medium Text hat jedes Zeichen eine Größe von 8·8 Pixel und es werden hierfür 2 Byte verwendet. Somit folgt :

Anzahl der Zeichen pro Bildschirmseite = (640·480) / (8·8) = 4800

Benötigter Speicherplatz pro Bildschirmseite = 4800·2 Byte = 9600 Byte ≈ 9,4 KByte

ii.) Vektorbilder : Zur Darstellung von Vektorbildern sei angenommen, daß ein typisches Bild aus 500 Geraden besteht. Jede Gerade sei durch die Koordinaten in x- und y- Richtung sowie einem 8 Bit großen Attributfeld beschrieben. Die Koordinaten in x- Richtung benötigen 10 Bit (wegen $\log_2(640)$), und in y- Richtung 9 Bit (wegen $\log_2(480)$).

Bits je Gerade = 2 · (9 Bit + 10 Bit) + 8 Bit = 46 Bit

Speicherplatz pro Bildschirmseite = 500 · 46 = 23000 Bit = 2,8 KByte

iii.) Pixelbilder : Einzelne Pixelbilder sollen mit 256 verschiedenen Farben kodiert werden, d.h. ein Pixel benötigt 1 Byte.

Speicherplatz pro Bildschirmseite = 640·480·1 Byte = 300 KByte

Die nächsten Beispiele spezifizieren unkomprimierte kontinuierliche Medien. Hierbei wird die zur Speicherung einer abgespielten Sekunde erforderliche Speichermenge ermittelt.

i.) Sprache : Unkomprimierte Sprache in Telefonqualität wird mit 8 kHz abgetastet und mit 8 Bit quantisiert.
Datenrate : 8 kHz · 8 Bit = 8 KByte/s .

Benötigter Speicherplatz pro Sekunde = 8 KByte

ii.) Stereo- Audiosignal : Ein unkomprimiertes Stereo- Audiosignal in CD-Qualität sei mit 44,1 kHz abgetastet und mit 16 Bit quantisiert.

Datenrate = 2 · 44100 Hz · 16 Bit = 1411200 Bits/s ≈ 172 KByte/s

Benötigter Speicherplatz pro Sekunde ≈ 172 KByte

iii.) CCIR 601 : Nach CCIR 601 (Studiostandard für digitale Videos) wird die Luminanz mit 13,5 MHz und die Chrominanz mit 6,75 MHz abgetastet. Mit einer 8 Bit Kodierung folgt :

Datenrate = (13,5 MHz + 6,75 MHz + 6,75 MHz) · 1 Byte ≈ 25,7 MByte/s

Benötigter Speicherplatz pro Sekunde ≈ 25,7 MByte

iv.) Sequenzen von digitalen Computer Bilder : Sequenzen von digitalen Computer Bildern können eine Größe von 640·480 Pixel haben bei 25 Bildern pro Sekunde. Wenn für die Grundfarben Rot, Grün und Blau jeweils 8 Bit verwendet werden, ergeben sich diese Werte :

Datenrate = 640 · 480 · 25 Hz · 3 Byte ≈ 22 MByte/s

Benötigter Speicherplatz pro Sekunde ≈ 22 MByte

Diese Ausführung zeigt exemplarisch die erhöhten Anforderungen an ein System im bezug auf den erforderlichen Speicherplatz und den Datendurchsatz, wenn Einzelbilder und insbesondere kontinuierliche Medien im Rechner verarbeitet werden sollen. Die Verarbeitung unkomprimierter Datenströme in einem Multimediasystem erfordert für Bewegtbilder Speicherplatz im Gigabyte-Bereich und für Zwischenpuffer im Hauptspeicher im Megabyte- Bereich. Sie bedingt auch weiterhin hohe Datenübertragungsraten innerhalb des Systems. Dies ist aber mit der heute

verfügbaren und in den nächsten Jahren zu erwartenden Technologie nicht kostengünstig realisierbar.

Mit Hilfe von geeigneten Kompressionsverfahren lassen sich diese Werte jedoch erheblich reduzieren. In den letzten Jahren wurden große Fortschritte in der Forschung, Entwicklung und Standardisierung in diesem Gebiet erzielt. Allgemein gilt, um kosteneffektive und realisierbare Lösungen anzubieten, müssen Multimediasysteme komprimierte Video- und Audioströme verarbeiten.

1.2 Anforderungen an die Kompressionsverfahren

Durch den Einsatz von Kompressionsverfahren in Multimediasystemen müssen hohe Anforderungen erfüllt werden. Die Qualität der komprimierten Daten sollte möglichst gut sein bzgl. des Kompressionsgrades, und die anschließend wieder dekomprimierten Daten sollten möglichst gut sein bzgl. der Genauigkeit. Dabei sollte die Komplexität des verwendeten Verfahrens gering sein, um eine schnelle und einfache Realisierung sowohl bei der Kompression als auch bei der Dekompression zu ermöglichen. Die Algorithmen dürfen insbesondere bei der Dekompression und gegebenenfalls auch bei der Kompression bestimmte Zeitschranken nicht überschreiten. Alle modernen Kompressionstechniken bilden einen Kompromiß zwischen diesen Zielen.

Jede Anwendung spezifiziert unterschiedliche Anforderungen. Allgemein gibt es zwei grundlegende Arten der Anforderung und zwar den Dialog- und den Abfrage- Modus. Beispielsweise gehören Videokonferenzen zum Dialog- Modus und audiovisuelle Auskunftssysteme zum Abfrage- Modus. Manche Kompressionsverfahren wie z.B. H.261 sind eher auf Dialog- Anwendungen abgestimmt, wo eine geringe Verzögerung benötigt wird. Dies geschieht meistens durch einen symmetrischen Aufwand bei der Kompression und Dekompression. Wogegen andere Verfahren wie MPEG mit hohen Aufwand bei der Kompression versuchen den Einsatz in Abfrage- Systemen zu optimieren.

Die besonderen Anforderungen an den Dialog- Modus ergeben sich aus den Eigenschaften der menschlichen Wahrnehmung. Eine Anforderung wäre hierfür :

- Die Ende- zu- Ende- Verzögerung für ein im Dialogsystem eingesetztes Verfahren soll für die reine Kompression und Dekompression nicht länger als 150 ms dauern. Werte im Bereich von 50ms sollten angestrebt werden, um einen Dialog in einer natürlichen Weise führen zu können. Dieser Wert von 50ms bezieht sich ausschließlich auf die Verzögerung, die sich durch die Kompression bzw. Dekompression ergibt. Die gesamte Ende- zu- Ende-Verzögerung beinhaltet zusätzlich noch Verzögerungen durch das Netzwerk, die Verarbeitung der im Endsystem benutzten Kommunikationsprotokolle und den Datentransfer von und zu den entsprechenden Ein- und Ausgabegeräten.

Folgende Anforderungen werden an die Kompressionsverfahren mit Anwendungen im Abfrage-Modus gestellt :

- Es soll sowohl ein schneller Vorlauf als auch ein schneller Rücklauf mit gleichzeitigen anzeigen bzw. abspielen der Daten möglich sein. Hiermit können einzelne Stellen gesucht und schnell gefunden werden.
- Ein wahlfreier Zugriff auf Einzelbilder und Audiopassagen im Datenstrom sollte innerhalb eines Intervalls von etwa einer halben Sekunde realisierbar sein. Dieser Zugriff sollte relativ schnell sein, um dem interaktiven Charakter von dem Abfrage- Modus gerecht zu werden.
- Die Dekompression von Einzelbildern, Video- und Audiopassagen sollte ohne Interpretation aller vorherigen Daten möglich sein. Damit lassen sich solche Passagen nach einem wahlfreien Zugriff editieren.

An die Kompressionsverfahren werden sowohl im Anfrage- Modus als auch im Dialog- Modus diese Anforderungen gestellt :

- Damit dieselben Daten unter unterschiedlichen Konfigurationen eingesetzt werden können, sollte ein Datenformat gewählt werden, das zum Beispiel unabhängig von der Bildschirmgröße und der Bildwiederholfrequenz ist.
- Beispielsweise für Audio- und Videodaten werden verschiedene Datenraten mit unterschiedlicher Qualität gewünscht. Die Datenrate sollte den jeweiligen Gegebenheiten angepaßt werden.
- Audio- und Videodaten sollten exakt synchronisierbar sein. Eine Synchronisation sollte auch mit anderen Medien von einem Systemprogramm realisiert werden können.
- Eine Realisierung sollte in Software oder mit wenigen hochintegrierten Bausteinen durchgeführt werden. Dies ermöglicht eine kostengünstige Lösung.
- Das Verfahren sollte eine Kooperation verschiedener Systeme ermöglichen. Die Daten sollten also von unterschiedlichen Systemen benutzt werden können. Insbesondere kann ein Datenaustausch über ein Netz erfolgen. Hierzu muß auf den unterschiedlichen Systemen eine Kompatibilität der Kompressionsverfahren bestehen. Dieser Anspruch läßt sich durch eine Festlegung von einem allgemeingültigen Standard realisieren (z.B. durch ISO, ITU).

Diese Anforderungen müssen in einem unterschiedlichem Ausmaß für die verschiedenen Kompressionsalgorithmen in Betracht gezogen werden.

2 Grundlagen der Kompression

Für ein besseres Verständnis der vielen Variationen von Kompressionstechniken, werden sie bezüglich ihres Verhaltens klassifiziert. Weiterhin ist es auch Sinnvoll einmal die prinzipiellen Schritte von einem Kompressionsverfahren zu betrachten.

2.1 Einteilung der Kompressionstechniken in Kategorien

Kompressionsverfahren lassen sich, wie in Abbildung 2.1 dargestellt, in verschiedene Kategorien einteilen. Es wird zwischen Entropiekodierung, Quellenkodierung und hybride Kodierung unterschieden. Dabei arbeitet die Entropiekodierung verlustfrei und die Quellenkodierung oft verlustbehaftet. Die hybriden Ansätze werden in den meisten Multimediasystemen eingesetzt. Sie sind meist eine Kombination der Quellen- und Entropiekodierung, und bilden somit nicht unbedingt andere Algorithmen.

Kodierungsart	Verfahren	
Entropiekodierung	Lauflängenkodierung	
	Pattern Substitution	Static Pattern Subst.
		Dynamic Pattern Subst.
	Variable- Length Encoding	Huffman- Kodierung
		Arithmetische Kodierung
	Adaptive Kompression	Variante der Huffman-Ko.
Quellenkodierung		ADPCM
	Vektorquantisierung	
	Transformationskodierung	
	Layered Coding	Subband- Kodierung
		Unterabtastung
	Prädiktion	DPCM
		Delta- Modulation
Hybride Kodierung	JPEG	
	MPEG	
	H.263	

Abbildung 2-1: Einige Kodierungs- bzw. Kompressionsverfahren in der Übersicht

Die Verfahren der Entropiekodierung werden auf verschiedene Medien (z.B. Sprache, Bilder) angewendet , ungeachtet deren medienspezifischer Eigenschaften. Diese allgemeinen Medien werden nur als eine Sequenz digitaler Datenwerte angesehen, deren Bedeutung nicht beachtet wird. Die Verlustfreiheit bezieht sich auf den Vergleich der zu kodierenden mit den dekodierten Daten. Das heißt, die Daten sind identisch, und es gehen keine Daten verloren. So kann beispielsweise die Lauflängenkodierung bei der Kompression von Daten beliebiger Natur in einem Dateisystem, bei Texten, Einzelbildern oder als Teil einer Bewegtbild- oder Audiokodierung eingesetzt werden.

Die Quellenkodierung verwendet hingegen die Eigenschaft der zu kodierenden Informationen, und macht sich zum Beispiel die eingeschränkte menschliche Wahrnehmung zum Vorteil. Diese oftmals verlustbehafteten Verfahren sind bezüglich des erreichbaren Kompressionsgrades abhängig vom jeweiligen Medium. Bei einer verlustbehafteten Kodierung werden die zu kodierenden Daten mit den dekodierten Daten in Beziehung gesetzt, wobei diese meist sehr ähnlich aber nicht gleich sind. Die Sprache beispielsweise ist sehr geeignet für eine Transformation des Zeitsignals in den Frequenzbereich, was zu einer erheblichen Reduktion der Datenmenge führt. Bei Einzelbildern hingegen können örtliche Ähnlichkeiten über eine Prädiktion des Inhaltes zur Kompression eingesetzt werden. In einem anderen Verfahren kann eine Transformation von dem Ortsraum in den zweidimensionalen Frequenzraum mit Hilfe der Kosinus- Transformation erfolgen. Tiefe Frequenzen definieren dann die durchschnittliche Farbe, wohingegen die Information hohe Frequenz eine scharfe Kante darstellt. Tiefe Frequenzen sind hier wesentlich wichtiger als die höheren Frequenzen. Diese Eigenschaft kann bei der Kompression genutzt werden.

Die Abbildung 2-1 zeigt nur einen Ausschnitt aller Kodierungs- und Kompressionsverfahren. Der Schwerpunkt liegt hier in den für Multimediasystemen wichtigen Algorithmen und ihren Eigenschaften.

2.2 Prinzipielle Schritte der Datenkompression

Abbildung 2-2 zeigt die allgemeine Vorgehensweise für eine Kompression von Einzelbildern, Video- oder Audiodaten.

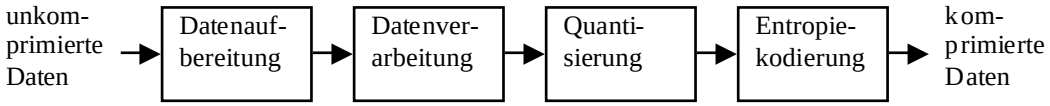


Abbildung 2-2: Wesentliche Schritte der Datenkompression

Als Beispiel wird in den folgenden vier Schritten hauptsächlich von einem Einzelbild als Datenstrom ausgegangen.

i.) Die Datenaufbereitung erzeugt eine geeignete digitale Darstellung der Informationen des zu verarbeitenden Mediums. Hierzu gehört zum Beispiel die Übersetzung analoger Daten in digitale Daten. Ein Bild wird hier beispielsweise zu 8·8 Pixel mit einer festgelegten Anzahl von Bits pro Pixel zerlegt. Im bezug auf Bilder wird dieser Schritt auch Bildaufbereitung genannt.

ii.) Die Datenverarbeitung führt eigentlich den ersten Schritt der Kompression mit Hilfe der unterschiedlichen Verfahren durch. So kann beispielsweise eine Transformation vom Zeitbereich in den Frequenzbereich mit Hilfe der diskreten Kosinus- Transformation (DCT) erfolgen. Bei einer Interframe- Kodierung können hier Bewegungsvektoren bestimmt werden, die jeweils für einen Block mit 8·8 Pixel gelten. Dieser Schritt heißt auch Bildverarbeitung bei Einzelbildern.

iii.) Die Quantisierung erfolgt im Anschluß an die mathematisch exakt ausgeführte Datenverarbeitung. Ein in der vorherigen Stufe ermittelter Wert kann und soll nicht mit seiner vollen Genauigkeit weiterverarbeitet werden. Er wird vielmehr gemäß einer bestimmten Auflösung quantisiert. So können auch in einem transformierten Raum die ermittelten Werte gemäß ihrer Wichtigkeit unterschiedlich behandelt werden, wie z.B. mit einer unterschiedlichen Anzahl an Bits.

iv.) Die anschließende Entropiekodierung geht von einem linearen Datenstrom einzelner Bits und Bytes aus. Hier erfolgt mit verschiedenen Verfahren eine abschließende verlustfreie Kompression. Beispielsweise können häufig auftretende längere Folgen von Nullen mit einer Angabe der Anzahl der aufeinander folgenden gleichen Zeichen und anschließend dem Zeichen selbst komprimiert werden (Lauflängenkodierung).

Die Datenverarbeitung und die Quantisierung können dabei mehrfach durchlaufen werden, wie im Fall der Adaptive Differential Pulse Code Modulation (ADPCM). Hier kann entweder eine Rückkopplung, wie bei der Delta- Modulation erfolgen, oder es können mehrere Verfahren hintereinander auf die Daten angewendet werden, wie z.B. eine Interframe- und eine Intraframe-Kodierung bei MPEG. Im Anschluß an diese vier Kompressionsschritte werden die digitalen Daten in einem bestimmten Format als Datenstrom zusammengefaßt. Dabei werden beispielsweise Bildanfang und Art der Kompression als Teil des Datenstroms integriert. Zusätzlich wäre an dieser Stelle auch ein Fehlerkorrekturcode bzw. Fehlererkennungscode denkbar.

Der Dekompressionsvorgang erfolgt invers zur Kompression, wobei die Realisierung einzelner Kodierer und Dekodierer jedoch sehr unterschiedlich aussehen kann. Eine symmetrische Kodierung zeichnet sich durch einen vergleichbaren Aufwand bei der Kodierung und Dekodierung aus, was insbesondere bei Dialoganwendungen anzustreben ist. Ein asymmetrisches Verfahren ermöglicht eine Dekodierung mit wesentlich geringerem Aufwand als bei der Kodierung. Dies ist für Anwendungen gedacht, bei denen die Kompression einmal erfolgt und die Dekompression sehr häufig und daher schnell erfolgen soll. Die Erstellung einer audiovisuellen

Lerneinheit wird beispielsweise einmal vorgenommen, und viele Lernende werden im Anschluß diese Daten oft dekodieren. Die prinzipielle Anforderung ist in diesem Fall die Dekompression in Echtzeit. Über diesen Mechanismus kann die Qualität der komprimierten Bilder gesteigert werden, da mehr Zeit für die Generierung der Bilder in Anspruch genommen werden kann, und somit auch der Kompressionsgrad verbessert wird.

3 Grundlegende Kompressionstechniken

Die Kompressionsverfahren für Video- und Audiodaten sind meistens hybride Verfahren. Sie enthalten mehrere Kompressionstechniken, wozu meistens auch eine Entropiekodierung oder eine Quellenkodierung gehört. Deswegen werden in diesem Kapitel eine Reihe der grundlegendsten Kompressionstechniken vorgestellt.

3.1 Lauflängenkodierung

Die Lauflängenkodierung nutzt die Eigenschaft, daß viele Daten aus einer Folge identischer Bytes bestehen. Wenn die Anzahl dieser identischen Bytes groß genug ist, kann eine erhebliche Reduktion der Datenmenge erzielt werden. Hierzu erfolgt eine Angabe von dem entsprechenden Byte und von der Anzahl des wiederholten Auftretens. So komprimierte Daten müssen aber gekennzeichnet werden, um sie bei der Dekompression von den nicht komprimierten Daten unterscheiden zu können. Hierzu kann eine beliebige Markierung M benutzt werden, die nicht Bestandteil der Daten ist. Oft ist es günstig eine Markierung in Form eines Bytes zu wählen. Da die nicht komprimierten Daten aber alle möglichen Bytes enthalten können, muß eventuell das Alphabet erweitert werden. Dieses Verfahren heißt auch Bytestopfen oder allgemeiner Datenstopfen.

Zum Beispiel kennzeichnet ein einmal hintereinander auftretendes M- Byte (Markierungsbyte) eine Lauflängenkodierung und ein zweimal hintereinander auftretendes M- Byte ein normales M-Byte in den Daten. Unter der Annahme, daß bei der Angabe der Anzahl der Bytewiederholungen in den Daten ein Byte benutzt wird, lohnt sich eine Kompression von Bytes nur wenn sie mindestens viermal in Folge auftreten, da drei Bytes (M- Byte, Zeichen und Anzahl) zur Kompression angegeben werden müssen. Auf diese Weise lassen sich dann zwischen 4 und 259 gleiche aufeinander folgende Bytes zu drei Bytes zusammenfassen, da z.B. für 4 Bytes der Index 0 benutzt werden kann. Bei der Kodierung wie auch bei der Dekodierung müssen diese Absprachen natürlich bekannt sein.

Ein Beispiel : (etwas vereinfacht dargestellt)

Unkomprimierte Daten : aMbbbccccccccd

Lauflängenkodierung : aMMbbbMc5d

Ein einmal auftretendes M- Byte in den unkomprimierten Daten wird hierbei zu zwei M-Bytes. Zeichenketten der Länge 3 werden nicht komprimiert. Und das Zeichen „c“ tritt neunmal hintereinander auf und wird zu den drei Zeichen „Mc5“ komprimiert.

Die Lauflängenkodierung ist eine Verallgemeinerung der Nullunterdrückung, wo das sich wiederholende Byte nicht angegeben wird. Statt dessen werden immer nur Folgen eines bestimmten Bytes komprimiert. In Texten beispielsweise könnte es das Leerzeichen sein. Unter den Annahmen, daß ein Byte für die Anzahl der Wiederholungen bereitgestellt wird, und wenn das Alphabet nicht wie im Beispiel der Lauflängenkodierung erweitert wurde, können Folgen der Länge 3 bis maximal 258 Bytes so auf zwei Bytes reduziert werden. Einzelne oder paarweise auftretende Zeichen werden nicht komprimiert.

Eine noch weitergehende Spezialisierung in diesem Zusammenhang bilden die Tabulatoren. Hierbei werden unterschiedliche M- Bytes definiert zur Charakterisierung einer unterschiedlichen Anzahl von Bytes. Beispielsweise kann ein M1-Byte für 4 Freistellen (blanks) stehen und ein

M2- Byte für 8 Freistellen. Beide Bytes hintereinander würden dann insgesamt 12 Freistellen einnehmen.

Ein Beispiel : (vereinfachte Darstellung)

Unkomprimierte Daten : ´ for(i=...)´

Komprimierte Daten : ´M2for(i=...)´

Offensichtlich ist die Lauflängenkodierung nur dann Effizient, wenn die Daten lange Folgen von gleichen Bytes beinhalten. Die Konsequenz für zusammengesetzte Kompressionsverfahren ist dann, daß vor der Lauflängenkodierung bestimmte Phasen der Kompression ihre Ausgabe auf diese Eigenschaft optimieren müssen.

3.2 Static Pattern Substitution und Diatomic Encoding

Bei Static Pattern Substitution werden häufig vorkommende Muster durch einzelne Bytes ersetzt. Dieses Verfahren eignet sich insbesondere zur Kompression von Texten. Beispielsweise können hier die Befehle einer höheren Programmiersprache ersetzt werden. Geeignete Befehle für die Ersetzung wären hierbei eventuell „Beginn“ oder auch „While“.

Ein Beispiel :

unkomprimierte Daten : WHILE(...);

komprimierte Daten : M(...);

Codetabelle : M = WHILE

Mit Hilfe einer Erweiterung von dem Alphabet kann auch eine große Anzahl von Wörtern komprimiert werden. Ein M- Byte zum Beispiel könnte auch anzeigen, daß ein kodiertes Wort folgt. Ein daran anschließendes Byte kann als Index zur Repräsentation eines Wortes dienen. Mit Hilfe eines Bytes können dann 256 verschiedene Wörter kodiert werden. Dieses Verfahren kann auch auf Einzelbilder und Audiodaten angewendet werden. Das Problem hierbei ist allerdings geeignete Muster zu finden, die oft genug auftreten. In diesem Fall benutzt man besser eine Approximation, die ein ähnliches Muster ermittelt. Diese Vorgehensweise entspricht dann der Vektorquantisierung.

Eine Variante von Static Pattern Substitution ist Diatomic Encoding, wo immer nur zwei Datenbytes zusammengefaßt werden. Hierzu werden die Bytepaare ermittelt die am häufigsten auftreten. Bei der Untersuchung der englischen Sprache hat sich gezeigt, daß

´E ´,´T ´,´TH´,´ A´,´S ´,´RE´,´IN´ und ´HE´

die häufigsten Bytepaare sind. Werden diese Bytepaare durch spezielle einzelne Bytes ersetzt , die im Text sonst nicht auftreten, führt das bereits zu einer durchschnittlichen Datenreduktion von über 10%.

3.3 Dynamic Pattern Substitution, Lempel-Ziv Encoding

Bei Static Pattern Substitution wird vorher eine Tabelle benötigt in der die einzelnen Muster stehen. Deshalb funktioniert das Verfahren nicht gut, wenn vorher keine Informationen über die Daten bekannt sind. Deshalb ist es besser eine Codetabelle erst zur Laufzeit zu erstellen. Dieses Verfahren heißt dann Dynamic Pattern Substitution. Im allgemeinen ist es aber schwierig immer die besten Muster zu extrahieren, und es erfordert eine beträchtliche Rechenleistung.

Ein Beispiel :

Unkomprimierte Daten : ABCDEABCEEABCEE

Komprimierte Daten : ABCDE11

Tabelleninhalt : 1=ABCEE

Lempel, Ziv und Welch haben einen Dynamic Pattern Substitution Algorithmus entworfen (vgl. Literaturangabe [HeMa 96]) , wo die Codetabelle während der Kompression konstruiert wird. Die Grundidee bei der Lempel-Ziv Kodierung ist es, jede neue Folge von Bytes sofort in der Codetabelle aufzunehmen. Hierzu wird jede neue Sequenz gespeichert als ein neuer Eintrag in der Codetabelle. Alle zukünftigen vorkommen dieser Sequenz werden kodiert mit dem zugehörigen Index. Lempel-Ziv Encoding wird in vielen modernen Programmen zur Kompression eingesetzt.

Der Lempel-Ziv Algorithmus sucht zuerst im Datenstrom nach dem ersten Vorkommen einer Sequenz die noch nicht gesehen wurde. Die Codetabelle (dictionary) wird hierbei sofort erstellt, und sie enthält zu jedem Zeitpunkt alle bisher gesehenen Teilstücke. Sei #i die Bezeichnung für den i-ten Index in der Tabelle und W der Inhalt von einem Fenster, das die Zeichenkette von dem zu komprimierendem Ausdruck angibt, der momentan von dem Verfahren betrachtet wird. Weiterhin sei K das nächste Zeichen aus dem Datenstrom.

Der Algorithmus (Pseudocode) :

i.) Initialisierung; jedes Element von dem Alphabet wird in der Codetabelle sortiert aufgenommen

ii.) Das Fenster wird initialisiert als leer, also : Fenster = [W] mit W = leer

iii.) Falls ein Zeichen K vorhanden ist, wird es am Ende von dem Fenster eingefügt :
Fenster = [WK]

Sonst wird der Index von W ausgegeben, und das Programm beendet

iv.) Dann wird überprüft, ob der momentane Fensterinhalt in der Codetabelle vorkommt.

Wenn ja, dann sei W = WK und springe zu Punkt iii.)

Sonst füge WK als einen neuen Eintrag in die Codetabelle ein, Index von W ausgeben ,

setze W = K und springe zu Punkt iii.)

Ein Beispiel für die Kompression mit dem Alphabet { A,B,C } und dem zu komprimierenden Ausdruck ´ABABAAA´ ist in Abbildung 3-1 dargestellt. Insgesamt ergeben sich die komprimierten Daten ´#1 #2 #4 #1 #7´ in der vorletzten Spalte.

	nächstes Zeichen	Fensterinhalt	bekannt ?	Codetabelle	Ausgabe	neuer Fensterinhalt
Initialisierung		[ ]		#1 = ´A´		
		[ ]		#2 = ´B´		
		[ ]		#3 = ´C´		
Interation	A	[A]	Ja (#1)			
	B	[AB]	Nein	#4 = ´AB´	#1	[B]
	A	[BA]	Nein	#5 = ´BA´	#2	[A]
	B	[AB]	Ja (#2)			
	A	[ABA]	Nein	#6=´ABA´	#4	[A]
	A	[AA]	Nein	#7=´AA´	#1	[A]
	A	[AA]	Ja (#7)			
Ende		[AA]			#7	

Abbildung 3-1: Lempel-Ziv Kodierung von ´ABABAAA´ mit einem Alphabet { A,B,C }

Die Besonderheit von dem Verfahren ist es, daß die Codetabelle dynamisch erzeugt wird, und sie muß nicht explizit übertragen werden. Die Codetabelle wird aber trotzdem irgendwo innerhalb der Daten übertragen, weshalb eine implizite Übertragung der Codetabelle stattfindet.
Die Dekomprimierung verwendet nun im Prinzip das inverse Verfahren der Kompression. Der Dekodierer kann die vollständige Codetabelle immer aus den Daten zurückgewinnen. Anhand von einem Index kann durch die Codetabelle ermittelt werden, welche Zeichensequenz dem Index entspricht. Der initiale Inhalt der Codetabelle entspricht dem sortierten Alphabet. Der Rest der Codetabelle baut sich auf indem außer beim ersten dekomprimierten Index das erste Zeichen von dem neuen dekodierten Codewort herausgenommen wird, und an das Ende des vorherigen dekodierten Codewortes angehängt wird. Dieses neue Codewort wird dann innerhalb der normalen Reihenfolge in der Tabelle aufgenommen.

	nächster Index	Ausgabe	Codetabelle
Initialisierung			#1 = ´A´
			#2 = ´B´
			#3 = ´C´
Interation	#1	´A´	
	#2	´B´	#4 = ´AB´
	#4	´AB´	#5 = ´BA´
	#1	´A´	#6 = ´ABA´
* Sonderfall	#7 (existiert nicht)	´AA´	#7 = ´AA´

Abbildung 3-2: Lempel-Ziv Dekodierung von ´#1#2#4#1#7´ mit einem Alphabet { A,B,C }

Das Problem hierbei ist aber, daß eventuell ein Index benötigt wird, für den es noch keinen Eintrag in der Tabelle gibt. Dieser Fall tritt auch exemplarisch im Beispiel von Abbildung 3-1 bzw. 3-2 auf mit dem Index 7. Die Frage ist hierbei, wie sieht der erste Buchstabe von dem neuen Codewort aus ? Das Problem entsteht durch einen um einen relevanten Schritt verspäteten Aufbau der Codetabelle bei der Dekompression. Denn bei der Kompression wird außer beim letzten Index der ausgegeben wird immer ein (nicht relevantes, da neues) Codewort in der Codetabelle aufgenommen, und bei der Dekompression wird außer beim ersten Index ein Eintrag in der Codetabelle aufgenommen. Ein unbekannter Index bedeutet, daß bei der Kompression der letzte Tabelleneintrag sofort im nächsten Schritt wieder benutzt wurde. Für diesen Fall gilt aber, daß der Anfang des zuletzt dekodierten Codewortes gleich dem Anfang von dem neuen dekodierten Codewort ist. Daraus ergibt sich insgesamt, daß der neue Eintrag in der Codetabelle dem zuletzt dekodierten Codewort plus seinem ersten Zeichen entspricht.

Der Kompressionsgrad verbessert sich normalerweise, wenn die Codetabelle größer wird. Die Größe der Codetabelle bildet ein Kompromiß zwischen der Geschwindigkeit und der maximal möglichen Kompressionsrate. Denn um so größer die Tabelle ist, um so aufwendiger gestaltet sich das Suchen der Daten. Wenn sie aber zu klein gewählt ist, können kaum große Wiederholungen von Zeichensequenzen gefunden werden, die aber für eine erfolgreiche Kompression entscheidend sind. Weiterhin existiert das Problem, daß die Codetabelle extrem schnell anwachsen kann, so daß eventuell zuviel Speicherplatz benötigt wird. Deswegen wird häufig eine Tabelle nur bis zu einer gewissen Größe generiert (z.B. 65536 Einträge), und ab dann muß die Tabelle wieder neu initialisiert werden.

3.4 Variable-Length Encoding, Statistiche Kodierung

Die Idee ist hierbei, daß verschiedene Zeichen nicht grundsätzlich mit der gleichen Anzahl von Bits kodiert werden. Dieser Gedanke liegt auch dem Morsealphabet zugrunde.

Ein Beispiel :

E = •
Z = —••

Hierbei werden häufig auftretende Zeichen wie z.B. das ´E´ mit kurzen und seltener auftretende Zeichen wie z.B. das ´Z´ mit längeren Sequenzen kodiert.

Die statistische Kodierung richtet sich nach der Häufigkeit des Auftretens einzelner Zeichen oder Folgen von Datenbytes. Dabei ist besonders auf eine eindeutige Dekompression zu achten. Es gibt verschiedene Verfahren, die nach derartigen statistischen Maßstäben arbeiten. Die bekanntesten sind die Huffman- Kodierung und die arithmetische Kodierung.

3.4.1 Huffman- Kodierung

Huffman Codes (vgl. Literaturangabe [Schw 97]) sind sogenannte Prefix Codes. Dies sind Codes mit variabler Symbollänge und der Eigenschaft, daß kein Codewort gleichzeitig vorderer Teil (Prefix) eines anderen Codewortes ist. Die variable Symbollänge erlaubt dann auch die Berücksichtigung der unterschiedlichen Häufigkeiten der verschiedenen Symbole.

Beispielsweise in einem deutschen Text kommt der Buchstabe ´e´ wesentlich häufiger als der Buchstabe ´x´ vor. Bei der ASCII Kodierung benötigen aber beide Symbole jeweils 8 Bit. Wahrscheinlich wäre ein Gewinn zu erwarten, falls ´e´ mit 4 Bits und ´x´ mit 12 Bits kodiert wird.

Voraussetzungen für die Anwendung eines Prefix Codes zur effizienten Datenkompression sind :

- die Existenz eines Alphabetes
- eine bekannte Häufigkeitsverteilung aller Symbole
- unterschiedliche Häufigkeiten der einzelnen Symbole

Der Huffman Algorithmus ermittelt dann die Kodierung eines Zeichens mit der optimalen Anzahl an Bits für eine vorgegebene Auftrittswahrscheinlichkeit. Die am häufigsten auftretenden Zeichen erhalten hierbei die kürzesten Codewörter.

Beispielsweise hat ein Alphabet die Mächtigkeit 6, und besteht aus den Buchstaben von ´A´ bis ´F´. Eine dazugehörige Häufigkeitsverteilung wurde auf statistischem Weg ermittelt (Profiling). Somit könnten sich die Werte, wie in Abbildung 3-3 ergeben.

Symbole	A	B	C	D	E	F
Häufigkeit	45	13	12	16	9	5
Kodierung mit fester Länge	000	001	010	011	100	101
Huffman Codes	0	101	100	111	1101	1100

Abbildung 3-3: Beispiel der Huffman- Kodierung

Zum Beispiel die Zeichenfolge ´AAB´ entspricht dann der kodierten Bitsequenz ´00101´. Aufgrundlage der Werte aus der Tabelle kann dann der Kompressionsgewinn berechnet werden, indem eine Summe über alle Zeichen gebildet wird, wo die Anzahl des Auftretens multipliziert wird mit der Anzahl der benötigten Bits für einen Buchstaben. Für das verwendete Beispiel ergeben sich diese Werte :

Kodierung mit fester Länge : (45+13+12+16+9+5) · 3 Bit = 300 Bit

Huffman Codes : 45· 1 Bit + (13+12+16) · 3 Bit + (9+5) · 4 Bit = 224 Bit

Gewinn : 100 % - 224/300 · 100% = 25.3 %

Die Kodierung mit der festen Länge muß natürlich gegeben sein. Zur Ermittlung eines Huffman Codes kann man sukzessive einen binären Baum aufbauen. Die Blätter stellen die zu kodierenden Zeichen dar. Alle Knoten beinhalten die Häufigkeit des Auftretens aller in diesem Unterbaum befindlichen zu kodierenden Zeichen. Die Kanten werden jeweils mit den Bits 0 und 1 versehen. Der Weg von der Wurzel zu einem Blatt entspricht den Bits des entsprechenden Symbols. Für dieses Beispiel ergibt sich der Baum in Abbildung 3-4.

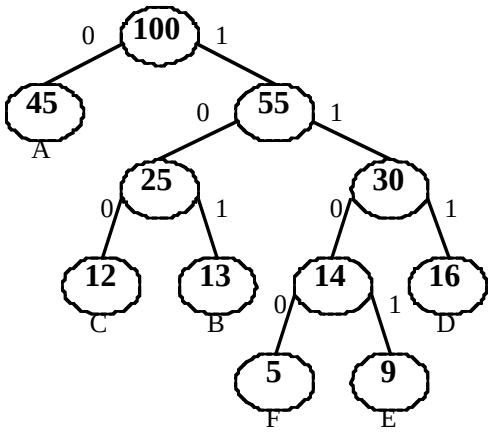


Abbildung 3-4: Beispiel von einem Baum für die Huffman- Kodierung

Der Baum wird konstruiert mit Hilfe der Häufigkeitsangaben. Hierbei werden immer die beiden Knoten mit der niedrigsten Häufigkeit zusammengefaßt zu einem neuen Knoten. Dieser Schritt wird dann solange wiederholt bis nur noch ein Knoten übrig bleibt. Beispielsweise wird in Abbildung 3-4 zuerst der Wert 5 (Häufigkeit vom Symbol ´F´) mit dem Wert 9 (Häufigkeit vom Symbol ´E´) addiert und zusammengefaßt zu einem neuen Knoten mit dem Wert 14.

Bei der Dekodierung einer Nachricht werden die Bits der Nachricht sequentiell abgearbeitet, und das aktuelle Symbol mit allen Symbolen der Tabelle, die die gleiche Symbollänge besitzen, verglichen. Nach einer Übereinstimmung wird mit einem neuen Symbol begonnen. Dieselbe Tabelle muß bei der Kodierung und Dekodierung vorliegen. Das heißt, das ein Teil der Tabelle an die komprimierten Daten angehängt wird, oder zwei Parteien müssen sich auf eine Tabelle vorher einigen. Die Huffman- Kodierung gehört zu den verlustfreien Kompressionstechniken und kann dadurch auf beliebige Datenströme angewendet werden.

3.4.2 Arithmetische Kodierung

Eine Einschränkung der Huffman- Kodierung ist es, daß die Bäume fast nie ausgeglichen sind bzgl. der Unterbäume eines Knotens, wo das Produkt aus Anzahl der Bits und Anzahl des Auftretens fast nie gleich ist. Wenn ein Zeichen eine Wahrscheinlichkeit von z.B. 90% hat, ist es leicht vorstellbar, daß bereits ein Bit, das mindestens bei der Huffman- Kodierung benötigt wird, zuviel ist für eine effiziente Kompression. Dieser Nachteil existiert bei der arithmetischen Kodierung (vgl. Literaturangabe [HeMa 96]) nicht unbedingt.

Die arithmetischen Kodierung ist eine Technik, bei der jedem Zeichen ein Intervall zwischen 0 und 1 zugeordnet wird in Abhängigkeit der Auftrittswahrscheinlichkeit. Eine Zeichenkette besteht aus mehreren Zeichen. Um so mehr Zeichen eine Zeichenkette hat, um so ein kleineres Intervall zwischen 0 und 1 wird dieser Zeichenkette zugeordnet. Eine beliebige Zahl aus dem Intervall repräsentiert dann später die Zeichenkette.

Es muß mit der Bestimmung der Auftrittswahrscheinlichkeiten von jedem Zeichen begonnen werden. Als nächstes erfolgt eine Festlegung der Reihenfolge der Zeichen. Diese Reihenfolge muß dann sowohl bei der Komprimierung als auch bei der Dekomprimierung vorliegen. Im Maße der Auftrittswahrscheinlichkeit wird ein entsprechend großes Intervall jedem Zeichen zugeordnet. Abbildung 3-5 stellt diesen Zusammenhang noch einmal an Hand von einem Beispiel dar. Das ´A´ zum Beispiel mit einer Auftrittswahrscheinlichkeit von 20% bekommt hier ein Intervall der Größe 0,2 zugeordnet.

Zeichen	Häufigkeit	Intervall
A	20 %	[0..0,2 [
B	30 %	[0,2..0,5 [
C	50 %	[0,5..1]

Abbildung 3-5: Beispiel für geeignete Intervallgrenzen

Sei low(i) die untere Intervallgrenze des i-ten Zeichens, und high(i) die entsprechende obere Intervallgrenze. Weiterhin sei L = 0 und H = 1. Um eine Zeichenkette zu komprimieren, werden für alle Zeichen i = {1..n} folgende zwei Gleichungen berechnet :

$$L = L + (H - L) \cdot low(i)$$

$$H = L + (H - L) \cdot high(i)$$ // hier muß mit dem alten L-Wert gerechnet werden

Eine Zahl aus dem Intervall [L..H [kann dann zur Kompression gewählt werden.

Zur Verdeutlichung dient Abbildung 3-6 mit den Werten aus Abbildung 3-5. Hier soll exemplarisch die Zeichenkette ´ACB´ komprimiert werden.

Der Wert des ersten Zeichen von ´ACB´ eingesetzt in den obigen Gleichungen, ergibt die Werte L=0 und H=0,2. Veranschaulicht wird dann das Intervall von dem Zeichen ´A´ graphisch auseinander gezogen, und die alten Intervallgrenzen 0 und 0,2 werden übernommen. Dann wird das neue Intervall unterteilt entsprechend der Einteilung, die auch schon vorher gewählt wurde. Weiterhin kommt insgesamt noch ein neuer zweiter Buchstabe für jedes mögliche Zeichen hinzu. Für das nächste Zeichen ´C´ ergibt sich $L = 0 + (0,2-0) \cdot 0,5 = 0,1$ und $H = 0 + (0,2-0) \cdot 1 = 0,2$, was dann der Zeichenkette ´AC´ entspricht. Jetzt wird wieder das Intervall ´AC´ graphisch auseinander gezogen und unterteilen. Abschließend folgt $L = 0,1 + (0,2-0,1) \cdot 0,2 = 0,12$ und $H = 0,1 + (0,2-0,1) \cdot 0.5 = 0,15$. Ein Wert wie z.B. 0,12 aus dem Intervall [0,12..0,15 [repräsentiert dann die kodierte Zeichensequenz.

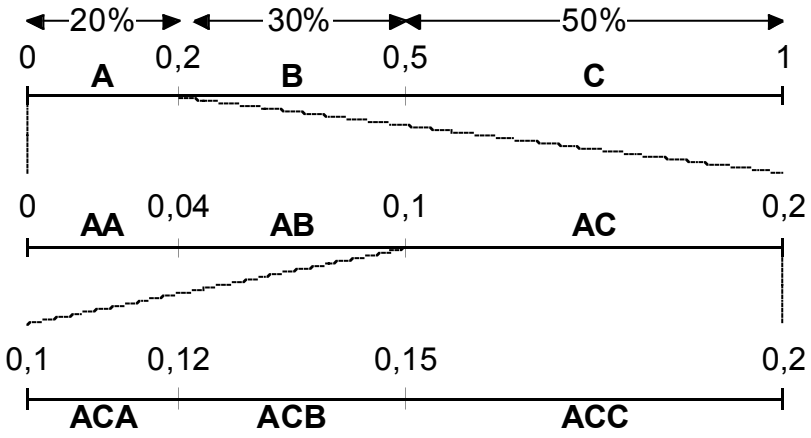


Abbildung 3-6: Ein Beispiel für die arithmetische Kodierung

Um bei der Dekompression zu wissen, wieviel Zeichen kodiert wurden, gibt es zwei Möglichkeiten. Entweder wird die Länge der Zeichenkette vorher übergeben, oder die Kompression benutzt ein Endezeichen. Dieser Mechanismus wird insbesondere auch deshalb benötigt, da Gleitkommazahlen nur eine begrenzte Genauigkeiten haben, und somit nicht beliebig genaue Zahlen übertragen können. Da Zeichenketten die häufiger vorkommen auch größere Intervalle besitzen, können hiervon mehr Zeichen durch eine Gleitkommazahl dargestellt werden, was zu der eigentlichen Kompression führt.

Bei der Dekompression wird zum Beispiel nachgesehen in welchem Intervall der Wert 0,12 fällt. Es ist das Intervall [0..0,2 [, daß für den Buchstaben ´A´ steht. Die untere Intervallgrenze 0 wird dann subtrahiert von der Zahl 0,12 , und die sich ergebene Zahl wird durch die Intervallgröße (0,2-0) geteilt . Das Ergebnis ist 0,6. Diese Zahl gehört zum Intervall [0,5..1], daß heißt sie symbolisiert den Buchstaben ´C´. Wieder kann die nächste Zahl berechneten werden mit (0,6-0,5) / (1-0,5) = 0,2. Diese Zahl liegt dann im Intervall [0,2..0,5 [und entspricht dem ´B´. Insgesamt ergibt sich die richtige Zeichenkette ´ACB´.

Die arithmetischen Kodierung kann auch auf Integer Zahlen basieren, wo z.B. Werte zwischen 0 und 65535 für die Intervalle benutz werden. Huffman- und arithmetische Kodierung können einzelne Symbole nicht direkt dekodieren. Sie müssen immer von Beginn an einen Datenstrom dekodieren, was einen zufälligen Zugriff unterbindet. In der Praxis hat sich gezeigt, daß beide Verfahren im allgemeinen sehr ähnliche Kompressionsraten besitzen.

3.5 Vektorquantisierung

Bei der Vektorquantisierung wird ein Datenstrom in Blöcke aufgeteilt. Diese Blöcke besitzen alle eine feste Größe von n Bytes. In einer gegebenen Tabelle befindet sich zusätzlich eine Menge von Mustern. Zur Kompression wird dann jeder einzelne Block mit den vorgegebenen Mustern verglichen. Hierbei wird das Muster gesucht, das nach einem festgelegten Kriterium dem Block am ähnlichsten ist. Ein Muster in der Tabelle entspricht einem eindeutigen Wert, wodurch jedem Block ein Index zugewiesen werden kann. Diese eindeutigen Werte sind dann die komprimierten Daten. Da eine solche Tabelle mehrere Dimensionen haben kann, entspricht der Index einem Vektor. Der korrespondierende Dekoder verfügt über die gleiche Tabelle, und generiert aus dem Vektor eine Approximation des ursprünglichen Datenstroms. Vektorquantisierung eignet sich insbesondere zur effizienten Kodierung von Sprachsignalen.

Ein einfaches Beispiel für die Vektorquantisierung wären Zahlen, die komprimiert werden sollen. Abbildung 3-7 sei eine gegebene Tabelle mit Mustern für Blöcke der Länge 2. Der Vektor zu einem Tabelleneintrag ergibt sich aus dem Wert der ersten und zweiten Dimension.

Dimension 1	Dimension 2			
	1	**2**	**3**	**4**
1	10	12	15	17
2	20	21	25	27
3	30	32	35	38
4	40	43	45	47
5	50	52	55	59
6	70	72	75	79
7	90	93	95	97

Abbildung 3-7: Ein Beispiel für die Vektorquantisierung

Die Muster ´10´ und ´80´ könnten z.B. hiernach zu den komprimierten Daten (1,1) und (6,4) führen. Die Kompression besteht dabei darin, daß ein Vektor nur noch 5 Bits benötigt, und zwar drei Bits für die erste Dimension und zwei Bits für die zweite Dimension. Die nicht komprimierten Daten hingegen benötigen 7 Bits.

3.6 Transformationskodierung

Ein ganz anderer Ansatz wir bei der Transformationskodierung verfolgt. Hier werden Daten in einem anderen mathematischen Raum transformiert, der sich besser für eine Kompression eignet. Es muß dabei immer eine inverse Transformation möglich sein. Transformationen sind beispielsweise die Fourier-, Hadamard-, Haar- und die Slant- Transformation. Die unterschiedlichen transformierten Daten haben jedoch keine wesentlichen Vorteile im bezug auf eine weitere Kompression. Die effektivsten Datenreduktionen sind die Diskrete Kosinus-Transformation (DCT) und die Schnelle Fourier- Transformation (FFT).

Die Kompression geschieht dadurch, daß die Daten in einem anderen Raum transformiert werden, wo sie weniger Bits für die Darstellung benötigen (siehe auch [Capp 85]). Beispielsweise muß ein Signal, das von einer Sinuswelle gebildet wird, im Zeitbereich durch sehr viele Abtastwerte dargestellt werden, und in einem anderen Bereich eventuell nur durch die drei Werte Frequenz, Phase und Amplitude.

3.7 Subband- Kodierung, selektive Frequenztransformation

Während die Transformationskodierung alle Daten in einem anderen Raum transformiert, wird bei einer selektiven Frequenztransformation nur ein spektrale Anteil des Singales in vorgegebenen Bereichen, wie Frequenzbändern betrachtet. Die Anzahl der Frequenzbänder ist hier eine entscheidendes Qualitätskriterium.

Das Prinzip der Subband- Kodierung wird in Abbildung 3-8 dargestellt. Eine Reihe von Filtern unterteilt den Eingabestrom in bestimmte Frequenzen, und ein Kodierer erzeugt einen digitalen Datenstrom für jedes Subband. Der Multiplexer fügt dann am Ende diese Datenströme zusammen. Die Dekodierung verläuft in einem entsprechenden inversen Prozeß.

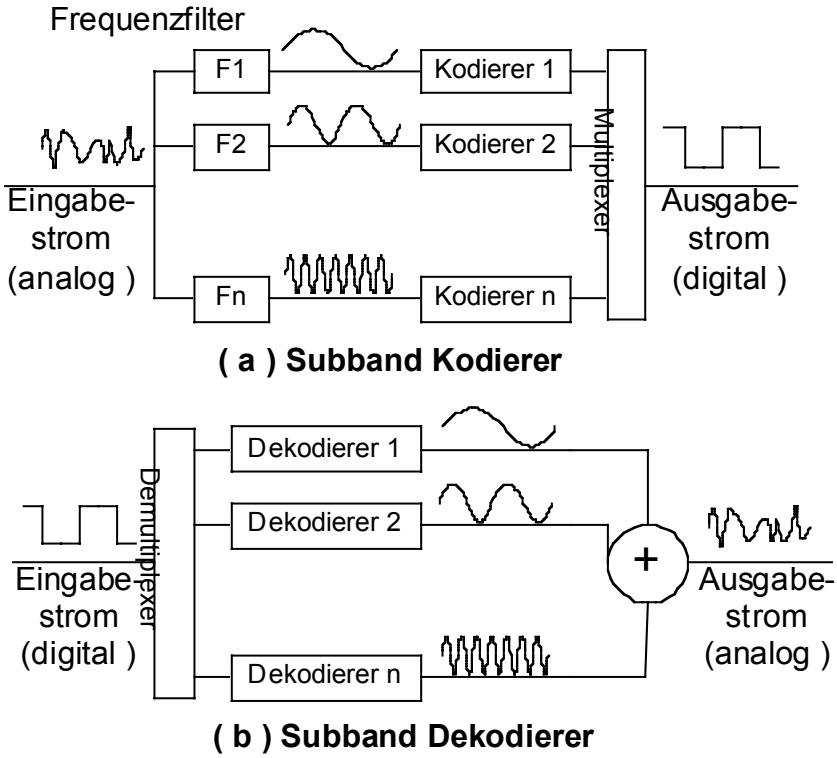


Abbildung 3-8: Die selektive Frequenztransformation

Wenn alle Frequenzen gleich wichtig sind, nützt die selektiven Frequenztransformation nichts. Es werden statt dessen unterschiedliche Intervalle benötigt , die mit einer unterschiedlichen Genauigkeit kodiert werden können. Ein sehr gutes Verständnis des Frequenzspektrums ist somit eine Vorbedingung für eine erfolgreiche Kodierung.

Eine Motivation hinter der Idee der Subband- Kodierung ist die Nutzung der Eigenschaften der menschlichen Wahrnehmung. Beispielsweise kann das menschliche Ohr nur Frequenzen von 40 Hz bis zu 20000 Hz hören. Die niedrigen und die hohen Frequenzen benötigen aber sehr viel Energie um überhaupt hörbar zu sein, und die Präzision bei der Wahrnehmung ist hierbei sehr schlecht. Deswegen können dieses Stellen mit weniger Bits kodiert werden.

3.8 Interpolation, Subsampling

Bei den auf der Interpolation basierenden Verfahren werden die spezifischen Eigenschaften der menschlichen Wahrnehmung genutzt. Dieses bezieht sich insbesondere auf das menschliche Gehöhr und auf die Augen.

Zum Beispiel reagiert das menschliche Auge empfindlicher auf Helligkeits- als auf Farbänderungen. Deswegen ist auch eine Aufteilung der Farben in Rot, Grün und Blau nicht unbedingt sinnvoll. Bei diesem RGB- Modell können aus einer Kombination dieser drei Primärfarben die meisten Farben erzeugt werden. Das YUV- Modell hingegen nutzt die oben aufgeführte Eigenschaft der Wahrnehmung aus. Anstatt Farben zu separieren, werden die Helligkeitsinformationen von den Farbinformationen getrennt. Der Kanal Y enthält die Helligkeitsinformationen (Luminanz), und die beiden Kanäle U und V enthalten die Farbinformationen (Chrominanz). Die Komponenten U und V können dann mit einer geringeren Auflösung an Zeilen und Spalten abgetastet werden, was zu einer Verringerung der gesamten Datenmenge führt. Diesen Vorgang heißt auch Unterabtastung.

Die Farben können aber trotzdem zwischen den beiden Farbmodellen transformiert werden mit diesen Gleichungen :

$Y = 0.30\ R + 0.59\ G + 0.11\ B$

$U = (B\text{-}Y) \cdot 0.493$

$V = (R\text{-}Y) \cdot 0.877$

Aufgrund dieser Gleichungen wird das YUV- Signal manchmal auch als Y,B-Y,R-Y Signal bezeichnet. Die YUV- Kodierung kann auch durch die unterschiedlichen Verhältnisse zwischen Helligkeitsinformationen und Farbinformationen beschrieben werden, wir z.B. als (4:2:2) Signal. Das YUV- Modell wird bei Fernsehsignalen benutzt, wo die Helligkeit aus Kompatibilitätsgründen für den Schwarzweißempfang immer mit übertragen werden muß. Die Benutzung der Farbinformationen hängt dann von den Farbfähigkeiten des Fernsehers ab.

3.9 Prädiktion, relative Kodierung

Anstatt einzelne Bytes oder Bytefolgen zu komprimieren, kann auch eine Differenzkodierung von Bytes und Bytefolgen vorgenommen werden. Dieses Verhalten wird mit Prädikation oder relative Kodierung bezeichnet. Das Verfahren benötigt Folgen ähnlicher Zahlen, damit insgesamt kleine Werte herauskommen können, die dann mit der Nullkompression oder mit weniger Bits kodiert werden können.

Im folgenden sei die Anwendung dieses Verfahrens auf einige Medien kurz beschrieben :

- Bezogen auf ein Bild wirken sich Kanten als große und Flächen mit ähnlicher Luminanz und Chrominanz als kleine Werte aus. Eine homogene Fläche wäre durch eine Vielzahl von Nullen charakterisiert. Diese können dann anschließend mit einer Lauflängenkodierung weiter komprimiert werden.
- Bezüglich einer Differenzbildung über die Zeit würden bei Bewegtbildern immer nur die Unterschiede zum vorherigen Bild kodiert. Eine Nachrichtensendung und die Videotelefonie haben hier einen großen Anteil von Nullbytes, weil sich der Hintergrund aufeinanderfolgender Bilder oft nicht verändert. Hier kann auch eine Bewegunugskompression erfolgen. Dafür werden Bereiche von beispielsweise jeweils 16·16 Pixel von aufeinanderfolgenden Bilder miteinander verglichen. Im Fall eines von links nach rechts fahrenden Autos würde im betrachteten Bildbereich ein weiter links liegender Bereich des vorherigen Bildes am ähnlichsten sein. Diese Bewegung kann dann als ein Vektor kodiert werden.
- In der Audiotechnik wird die Differential Pulse Code Modulation (DPCM) auf eine Folge von Abtastwerten angewendet. Hier muß nicht jeder Abtastwert mit seiner vollen Anzahl von Bits abgelegt werden, da Audio Signale sich nur ziemlich langsam ändern. Es genügt, den ersten kodierten Abtastwert mit vielen Bits festzulegen. Jeder weitere wird als Differenz zum vorherigen Wert mit wenigen Bits kodiert.
- Die Delta- Modulation ist eine Abwandlung der DPCM. Hier existiert die Beschränkung bei der Kodierung der Differenzwerte auf genau ein Bit, das das Steigen bzw. Fallen des Signalverlaufs angibt. Steile Flanken werden so nur ungenau kodiert. Besondere Vorteile hat dieses Verfahren, wenn man bei der Kodierung von Daten nicht auf die 8-Bit-Rasterung (1 Byte) angewiesen ist. Wenn die Differenzen klein sind, so genügt dafür eine weitaus geringere Anzahl an Bits.

Die Differenzbildung ist ein wesentliches Merkmal aller in Multimediasystemen eingesetzter Verfahren.

3.10 Adaptive Kompressionsverfahren

Eine Vielzahl der bisher beschriebenen Kompressionsverfahren basiert auf der Ausnutzung vorab bekannter Eigenschaften der zu komprimierenden Daten, z.B. wie oft Folgen von bestimmten Bytes auftreten. Eine untypische Folge von Zeichen spiegelt sich dann in einer nicht guten Kompression wider. Es existieren jedoch auch adaptive Kompressionsverfahren, die eine Anpassung des Verfahrens an die jeweils zu komprimierenden Daten zulassen. Diese Adaption kann auf sehr unterschiedliche Weise erfolgen.

3.10.1 Eine adaptive Huffman- Kodierung

In einem ersten Verfahren wird eine adaptive Huffman- Kodierung beschrieben. Es existiert hierfür wieder eine Tabelle mit Einträgen, und es wird jedem Zeichen ein Codewort und ein Zähler zugeordnet. Der Zähler entspricht der Anzahl des Auftretens von einem Zeichen, und wird zu Beginn mit Null initialisiert. Die Codewörter sind am Anfang alle gleich lang. Gegeben sei nun das erste zu kodierende Zeichen. Der Kodierer liefert dafür das Codewort gemäß der Tabelle. Zusätzlich erhöht er den zugehörigen Zähler von dem entsprechenden Eintrag um eins. Nach einer bestimmten Anzahl von Zeichen wird ein Algorithmus aufgerufen, der jedem Zeichen ein Codewort variabler Länge zuweist gemäß der Auftrittshäufigkeit. Weiterhin werden dann die Zähler durch eine Konstante geteilt, die in Abhängigkeit von der Anzahl der bisherigen Schritte und der Größe des Alphabets gewählt wird. Dadurch entstehen eine neue Zuordnungen der Codewörter, wobei die lokal am häufigsten auftretenden Zeichen, die höchsten Zählerstände besitzen, und somit immer mit den kürzesten Codewörtern verschlüsselt werden.

Beispielsweise hätte dieses Verfahren erhebliche Vorteile gegenüber der normalen Huffman-Kodierung bei einer Zeichenkette, wo die einzelnen Buchstaben in großen Blöcken vorkommen, und jedes Zeichen insgesamt gleich oft vorhanden ist. Ein weiterer Vorteil ist, daß die Codetabelle nicht mit übertragen werden muß.

3.10.2 ADPCM

Ein zweites beispielhaftes Verfahren für die adaptive Kompression stellt eine Verallgemeinerung der DPCM dar, die Adaptive DPCM (ADPCM). Diese wird der Einfachheit halber oft unter dem Begriff der DPCM subsumiert. Die Differenzwerte werden bei DPCM nur mit wenigen Bits kodiert. Dann könnten entweder nur sehr grobe Übergänge korrekt kodiert werden, wenn die Bits eine hohe Wertigkeit haben, oder es könnten nur sehr kleine Übergänge korrekt kodieren werden. Im ersten Fall wäre die Auflösung bei leisen Audiosignalen unzureichend, im zweiten Fall würde ein Verlust hoher Frequenzen auftreten.

Die ADPCM ermöglicht eine Anpassung dieser Wertigkeit an die auftretenden Datenströme. Der Kodierer dividiert die DPCM- Abtastwerte durch eine geeignete Konstante, und der Dekoder multipliziert die Werte wieder mit dieser Konstanten. Den Wert der Konstanten paßt der Kodierer dem DPCM kodierten Signal automatisch an.

Für ein Signal mit oft auftretenden sehr großen DPCM- Werten, also mit vielen Anteilen an hohen Frequenzen, wird der Kodierer einen großen Wert für die Konstante ermitteln. Der Effekt ist eine sehr grobe Quantisierung des Signals in Passagen mit steilen Flanken. Niederfrequente Anteile innerhalb einer solchen Passage werden kaum berücksichtigt.

Für ein Signal mit ständig relativ niedrigen Werten, also mit wenigen Anteilen an hohen Frequenzen, berechnet der Kodierer eine kleine Konstante. Damit ist eine Auflösung der dann dominanten niederfrequenten Signalanteile gewährleistet. Sollten in einer derartigen Passage aber plötzlich hochfrequente Signalanteile auftreten, dann entsteht eine Signalverzerrung, die einem sogenannten Slope Overhead entspricht. Die größtmögliche Änderung mit der vorhandenen Anzahl an Bits unter Berücksichtigung der derzeit gegebenen Schrittweite ist nicht groß genug, um mit einem ADPCM- Wert den optimalen Wert darzustellen. Ein Sprung im Signal wird verwaschen.

Eine Änderung der adaptiv einzustellenden Konstanten kann beim Kodieren explizit zusätzlich in die komprimierten Daten eingefügt werden. Alternativ würde der Dekoder aus einem ADPCM-kodierten Datenstrom die Konstante selbst berechnen. Dieser Prädiktor ist so zu bemessen, daß für die auftretenden Daten der Fehler minimiert wird. Dabei sei angemerkt, daß hier der Begriff des Fehlers und die damit verbundene Bemessung des Prädiktors abhängig vom Medium ist.

Ein Audiosignal mit sich häufig ändernden Anteilen extrem niedriger und hoher Frequenzen eignet sich nur bedingt für eine ADPCM- Kodierung. Für Anwendungen in der Telefonie hat die ITU eine mit 32 KBit/s arbeitende Version des ADPCM- Verfahrens standardisiert, die 4 Bit pro Abtastwert und eine Abtastfrequenz von 8 kHz verwendet.

3.11 Weitere Verfahren

Neben dem bisher beschrieben Kompressionstechniken sind einige weitere Techniken bekannt wie zum Beispiel :

- In der Videotechnik wird auch eine Reduktion der Datenmenge durch die Verwendung von Farbtabellen erreicht. Hierbei werden beispielsweise nur die benutzen Farben in einer Tabelle geschrieben und durch einen Index adressiert. Wodurch die Anzahl der Bits pro Pixel gesenkt werden kann.
- Eine einfaches Verfahren in der Audiotechnik ist die Stummschaltung. Hier werden nur Daten kodiert, wenn der Lautstärkepegel einen bestimmten Schwellenwert überschreitet.

4 Literatur

[EfSt 98] W. Effelsberg, R. Steinmetz :
„Video Compression Techniques“
dpunkt Verlag 1998

[HeMa 96] G. Held, T. Marshall :
„Data and image compression“
John Wiley & Sons LTD 1996

[Stei 93] R. Steinmetz :
„Multimedia-Technologie“
Springer Verlag 1993

[Stei 98] R. Steinmetz :
„Multimedia-Technologie“
Springer Verlag 1998

[Schw 97] U. Schwiegelsohn :
„Technische Informatik I“
p. 122-125, Universität Dortmund, Lehrstuhl für Datenverarbeitungssysteme 1997

[VaHH 98] P. Vary, U. Heute, W. Hess:
„Digitale Sprachsignalverarbeitung“
B.G. Teubner Stuttgart 1998

[Capp 85] V.Cappelini
„Data compression and error control techniques with applications“
Academic Press Inc. 1985

Kompressionstechniken für Einzelbilder

Dirk Denninghoff

FB Informatik, Uni Dortmund

denninghoff@gmx.de

Zusammenfassung

In diesem Beitrag sollen mehrere Verfahren zur Einzelbildkompression vorgestellt werden.

Im ersten Teil werden eher einfache Techniken beschrieben. Block Truncation Compression und Color Cell Coding zeigen, daß bereits einfache Blocking Algorithmen passable Kompressionsraten ermöglichen. Das vorgestellte Telefax-Komprimierungsverfahren verwendet bereits bekannte Basiskomprinierungsmethoden wie Huffmann- und Run-Length Kodierung und ist ausschließlich für s/w-Bilder vorgesehen.

Im zweiten Teil folgen zwei relativ neue, effizientere Methoden. Diese Methoden benutzen im Gegensatz zu den ersten Dreien intelligentere Methoden.

Die auf Wavelets basierende Komprimierung ist eine Methode, welche die sich durch eine Transformation der Ausgangsdaten in eine Frequenzdarstellung ergebenden Eigenschaften ausnutzt. Nach dieser Transformation liegen die Bilddaten in einer Form vor, die eine gute Komprimierung (lossy) bei akzeptablen Qualitätseinbußen zuläßt. Es werden Komprimierungsraten von 50:1 und höher erreicht.

Bei der Fraktalen Komprimierung handelt es sich um eine völlig neuartige und unkonventionelle Methode. Die Pixelwerte werden hier nicht als (zweidimensionales) Signal angesehen, das beim Dekomprimieren möglichst genau wiederhergestellt werden muß. Vielmehr wird versucht, im Bild sogenannte „Selbstähnlichkeiten“ zu finden. Allein durch die Beschreibung dieser „Selbstähnlichkeiten“ kann das Bild rekonstruiert werden. Auch hier sind Komprimierungsraten von 50:1 bei passablen Ergebnissen möglich.

1 Einleitung

In diesem Beitrag geht es zum einen um elementare, einfache Techniken zur Einzelbildkomprimierung. Hierbei ist Kap. 2 gemeint. Es soll gezeigt werden, daß unter Inkaufnahme eines gewissen Verlustes bezüglich der Bildinformationen schon beachtliche Kompresionsraten zustandekommen. Die folgende Beschreibung des TELEFAX-Komprimierungstandards zeigt die Effizienz von grundlegenden Komprimierungstechniken wenn Verluste unerwünscht sind.

Diese einfachen Techniken sind allerdings in keinster Weise zufriedenstellend. In modernen Kompressionsstandards findet man diese Verfahren höchstens im Ansatz. Moderne Komprimierungstechniken kombinieren zumeist eine verlustbehaftete Transformation des Bildes und kodieren das Ergebnis mit einfachen Methoden, um den Komprimierungseffekt zu erzielen.

Eine solche Technik wird im Kap. 3 beschrieben. Hierbei handelt es sich um ein sehr neues Verfahren, für das noch kein Standard definiert wurde. Es existieren aber bereits erste Implementierungen, die sehr gute Ergebnisse vorweisen.

Im Kap. 4 wird ein weiteres sehr neues Verfahren vorgestellt, das einen alternativen Ansatz wählt. Es wird keine explizite Repräsentation der Bildinformation gespeichert, sondern ein Funktionensystem, das in seiner Anwendung eine Annäherung an das Ursprungsbildmaterial ermöglicht.

Im Folgenden wird davon ausgegangen, daß das Bildmaterial in einer digitalen Form vorliegt. Die Pixel bestehen aus Tupeln von Farbwerten im RGB-Format. Alle folgenden Verfahren (bis auf CCC) beziehen sich allerdings nur auf die Behandlung eines Farbwertes innerhalb dieser Tupeln. Ein Farbbild läßt sich mit diesen Verfahren komprimieren, indem drei Durchläufe für jeweils einen Farbkanal gestartet werden.

Nebenbei sei an dieser Stelle angemerkt, daß bei den folgend beschriebenen Verfahren durch eine vorangestellte Farbraumtransformation in einen alternativen Farbraum (z.B. YUV) und eine an das menschliche Sehverhalten angepaßte Behandlung der einzelnen Kanäle möglicherweise andere, bessere Ergebnisse erzielt werden können. Da hier aber nur die Grundtechniken der einzelnen Verfahren vorgestellt werden sollen, wird auf diese Tatsache in den einzelnen Kapiteln nur teilweise eingegangen.

2 Einfache Techniken

Bevor moderne, effiziente Kompressionstechniken für Multimediaapplikationen entwickelt wurden, waren viel einfachere, block-orientierte Kompressionsschemata sehr populär. Im Gegensatz zu MPEG oder JPEG benutzen sie keine Transformation (z.B. DCT o.ä.), sondern behalten den zweidimensionalen Bildraum bei. Da der rechenintensivste Schritt der Kompression hier ausgelassen wird, laufen diese Algorithmen in der Regel sehr viel schneller. Allerdings wird hier der Kompromiß der verlustbehafteten Kompression auf sehr radikale Art und Weise erreicht und somit liefern diese Techniken aus heutiger Sicht keine zufriedenstellenden Ergebnisse.

2.1 Block Truncation Coding (BTC)

Beim Block Truncation Coding handelt es sich um ein sehr einfaches Komprimierungsverfahren. Das Ausgangsbild wird blockweise behandelt, d.h. es findet eine Aufteilung des Gesamtbildes in $n \times m$-Blöcke statt. Diese Blöcke werden völlig unabhängig voneinander betrachtet.

Die Grundidee ist, für die einzelnen Blöcke eine Farbreduktion auf nur zwei für jeden einzelnen Block repräsentative Farben vorzunehmen. Dieser Verlust an Bildinformation ist aus heutiger Sicht zu groß und nur in wenigen Fällen erträglich. Durch diese Reduktion läßt sich allerdings eine sehr schnelle Berechnung der komprimierten Version erreichen.

Zur Berechnung der zwei Farbwerte und deren Verteilung innerhalb des Blocks ist es nötig den Durchschnitt der vorkommenden Farbwerte und die Standardabweichung zu berechnen.

$$\mu = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{i,j}$$

$$\sigma = \sqrt{\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{i,j}^2 - \mu^2}$$

mit $P_{i,j}$ = Farbwert des Pixels an der Position i,j

Anhand des Durchschnitts μ wird entschieden, ob der Farbton an der Stelle i,j heller oder dunkler ist. Das Ergebnis für alle Pixel wird in einer Bitmatrix abgelegt.

Für die Rekonstruktion des Blocks werden zusätzlich zwei Farbwerte a und b benötigt.

$$a = \mu + \sigma\sqrt{p/q}\ , \qquad b = \mu - \sigma\sqrt{q/p}$$

mit p = # helle Pixel, q = # dunkle Pixel

Bei der Wiederherstellung wird nun ein Bild Block für Block wiederaufgebaut, indem anhand der Bitmatrix entschieden wird ob ein Pixel den Farbwert a oder b trägt.

2.2 Color Cell Compression (CCC)

Das Verfahren der Color Cell Compression (CCC) geschieht ganz ähnlich dem Block Truncation Coding. Bei diesem Verfahren werden die Farbkanäle nun nicht mehr getrennt betrachtet. Es wird versucht durch das Zusammenfassen der drei Farbkanäle den Beschreibungsaufwand für jeden Block noch weiter herunterzudrücken. Man berechnet nun für jeden Block nur noch eine Bitmatrix (BTC bei Farbbildern: drei). Als Durchschnitt muß nun ein Wert herangezogen werden, der alle Farbkanäle mit einbezieht. Man berechnet hierfür einen Luminanzwert:

$$Y = 0,3P_{rot} + 0,59P_{gruen} + 0,11P_{blau}$$

Dieser Luminanzwert ist wahrnehmungsorientiert, d.h. die Gewichtung der drei Farbkomponenten entspricht am ehesten dem subjektive Helligkeitsempfinden des Menschen. Dieser Wert kommt z.B. auch beim PAL-Fernsehstandard zum Einsatz und stellt so die Kompatibilität zu S/W-Geräten her. Der μ-Wert wird also wie folgt berechnet:

$$\mu = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} Y_{i,j}$$

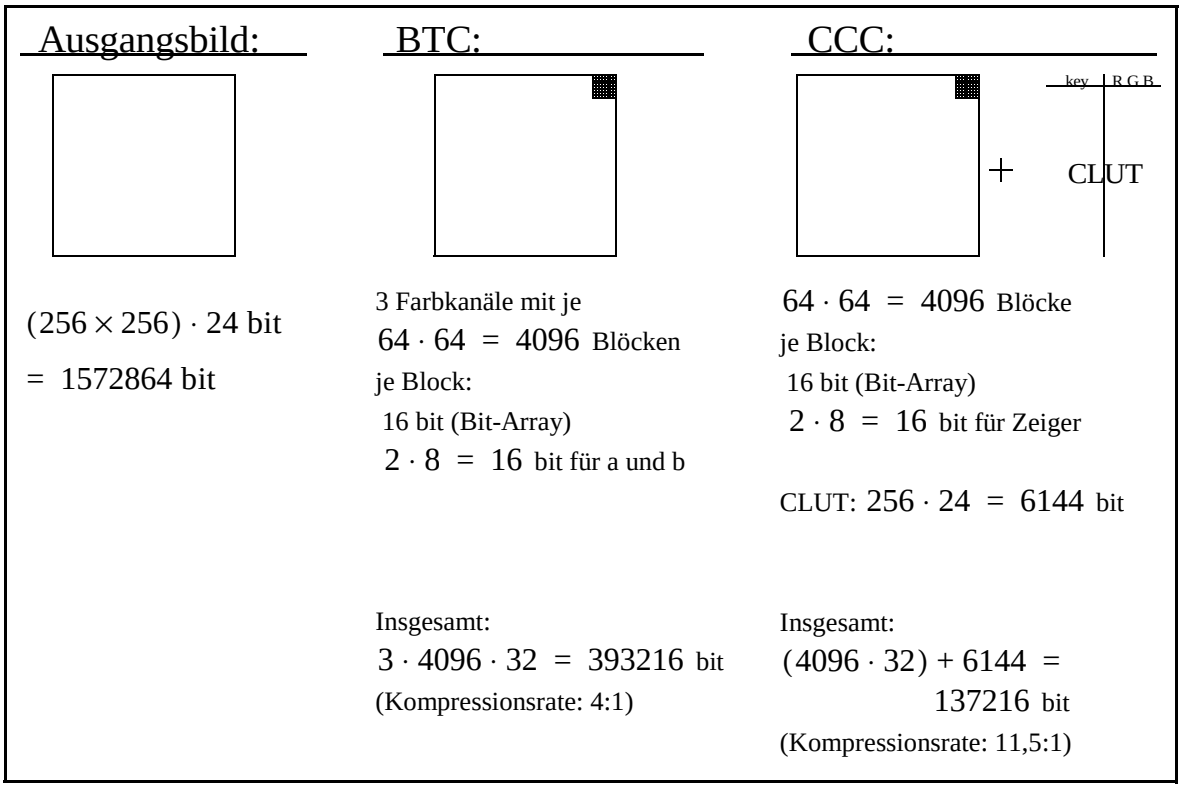
Für die Berechnung der Farben a_{Farbe} und b_{Farbe} benötigt man nun keinen Abweichungswert, sondern bildet nun einfach den Durchschnitt aus den Farbkomponenten für die hellen (a) bzw. dunklen (b) Töne.

$$a_{Farbe} = \frac{1}{q}\sum_{Y_{i,j} \geq \mu} P_{Farbe,i,j}\ , \qquad b_{Farbe} = \frac{1}{p}\sum_{Y_{i,j} < \mu} P_{Farbe,i,j}$$

Die Farbwerte a und b werden nicht mehr direkt mit jedem Block gespeichert, sondern nur noch als Zeiger in eine Fartabelle (Color Lookup Table = CLUT). Diese Farbtabelle trägt als Inhalt einen geeigneten Ausschnitt aus dem gesamten Farbraum. Dieser Ausschnitt ist in der Regel sehr viel kleiner als der gesamte darstellbare Farbraum. Auf diese Weise verschwinden ganz feine Farbunterschiede, die vom menschlichen Auge sowieso kaum wahrgenommen werden können zugunsten eines erheblich kleineren Beschreibungsaufwands für die Zeiger auf diese Tabelle. Bei der Berechnung der Zeiger ist eine Quantisierungsfunktion (Rundung) zum Treffen der gewünschten Tabelleneinträge nötig.

2.3 Vergleich BTC vs. CCC

Es folgt ein konkretes Beispiel, an dem die Komprimierungsraten der soeben vorgestellten Komprimierungstechniken nachvollzogen werden können.



2.4 Telefax Komprimierung

Ein weiterer einfacher und unkomplizierter Ansatz für die Einzelbildkomprimierung ist die Verwendung der Lauflängenkodierung und der Huffmannkodierung. Diese Kompressionsmethoden können beispielsweise zeilenweise auf das Bildmaterial angewendet werden.

Für photografisches Bildmaterial ist diese einfache Art der Komprimierung allerdings denkbar schlecht. Längere Lauflängen von ein und demselben Farbwert sind eher selten, aus diesem Grunde ließe sich so keine zufriedenstellende Reduktion der Datenmenge erreichen. Für Schwarzweißbilder oder Graustufenbilder mit verhältnismäßig wenig verschiedenen Farben kann diese Methode aber schon gute Ergebnisse liefern. In den ITU-T Standards für Telefax Gruppe 3 und 4 werden ausschließlich solche einfachen Methoden benutzt.

Hier wird der Kompressionsstandard der Gruppe 3 beschrieben. Er wurde zwischen 1980 und 1988 basierend auf den folgenden Parametern entwickelt:

- Schwarzweißbilder von der Größe A4
- Auflösung 100 Punkt pro Zoll
- 1728 Samples pro Zeile
- Übertragungsgeschwindigkeit von 4800 bit/s über das Telefonnetz

Hier wird zur besseren Veranschaulichung ein etwas vereinfachter Algorithmus behandelt. Offensichtlich ergeben sich bei den oben angegebenen Auflösungen Lauflängen, die viel größer als eins sind. Aus

diesem Grund ist Lauflängenkodierung hier effizient. Da außerdem ja nur die Farben schwarz und weiß definiert sind, müssen die Farben nicht explizit kodiert werden. Eine Zeile besteht somit nur aus verschiedenen Lauflängen wechselnder Bits.

Statistische Untersuchungen haben ergeben, daß die Verteilung der verschiedenen Lauflängen nicht gleichmäßig ist. Aus diesem Grund ist eine für diese Gegebenheit optimierte Huffmankodierung, welche Kodewörter unterschiedlicher Länge benutzt angebracht. Die folgende Tabelle zeigt als Beispiel die Reihen 0 bis 20 der Kodetabelle. Kodewörter für Lauflängen, die mit hoher Wahrscheinlichkeit auftreten sind, sind besonders kurz [EfSt 98].

weiße Lauflänge	Kodewort	schwarze Lauflänge	Kodewort
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	000001000
19	0001100	19	00001100111
20	0001000	20	00001101000

Einzelne Bitfehler bei der Übertragung würden das gesamte Bild beschädigen. Dagegen definiert der Standard spezielle Kodewörter für die resynchronisierung, wie ein EOL (end-of-line) Kodewort.

Für typische Briefe (z.B. Schreibmaschine) ist dieses simple Kompressionsschema sehr effizient.

3 Komprimierung mit Wavelets

Auch Wavelet-basierte Bildkompressionstechniken sind vom Typ „lossy“. Die erreichbaren Kompressionsraten bei akzeptablen Rekonstruktionen sind verhältnismäßig hoch. Bei fotografischem Ausgangsmaterial sind Kompressionsraten von ca. 50:1 und darüber erreichbar. Es wurde bisher allerdings noch kein Standard festgeschrieben wie beispielsweise bei JPEG. Aus diesem Grund finden solche Verfahren bis heute noch keine große Verbreitung und sind noch sehr speziellen Anwendungen (z.B. zur Speicherung von Fingerabdrücken beim FBI) vorbehalten. Erste Versuche von kommerziellen Firmen ein Wavelet-basiertes Bilddatenformat als Quasi-Standard zu etablieren wurden bereits gestartet.

Auf Wavelets basierende Verfahren ähneln in ihrem Kernstück (der Wavelet-Transformation) dem

JPEG-Verfahren. Ähnlich der DCT (JPEG) wird bei der Wavelet-basierten Komprimierung versucht durch Transformation des Signals (Bilddaten) aus der Zeit-/Amplitudendarstellung in einen anderen Darstellungsraum eine für Komprimierungszwecke günstige Repräsentation zu finden.

Lokale Verdichtung ist ein solches Kriterium. Ist das zu komprimierende Signal an den meisten Stellen gleich oder sehr nahe Null, so läßt es sich durch geeignete Quantisierung und eine anschließende run-length-Kodierung sehr gut komprimieren.

In Farbbildern (speziell Fotos) kommen Bereiche „genügend“ gleicher Farbwerte nur sehr selten vor. Gleichmäßige Farbverläufe und Übergänge, an denen keine scharfen Kanten existieren, sind jedoch relativ häufig vorzufinden.

Die Strategie der vorzustellenden Komprimierungsmethode ist es nun Kanten, d.h. „scharfe“ Übergänge eines Farbwertes, herauszufiltern, und diese getrennt von einer „geglätteten“ Version des Ursprungsbildes in geeigneter, platzsparender Kodierung abzulegen.

Die Wavelet-basierte Kompression verfolgt keine Blocking-Strategie. Das Signal wird als eindimensionales Signal angesehen. In einem zweistufigen Verfahren werden die extremen Amplitudensprünge zuerst in horizontaler Richtung und dann in vertikaler Richtung „herausgefiltert“.

3.1 Von der Fourier- zur Wavelet-Transformation

Dieser Abschnitt soll ein intuitives Verständnis dafür erzeugen, was Wavelets sind und warum sie gerade für die Bildkompression geeignet sind.

Eine der gebräuchlichsten Arten ein Signal $f(x)$ zu analysieren ist, es als gewichtete Summe einfacher Bausteine (Basisblocks) darzustellen:

$$f(x) = \sum_i c_i \psi_i(x)$$

mit $\Psi_i(x)$ = Basisfunktionen und c_i = Koeffizienten (oder Gewichte)

Da die Basisfunktionen fest sind, steckt die ganze Information über das Signal in den Koeffizienten.

Die einfachste einer solchen Repräsentation ist die, die lediglich die Impulsfunktion benutzt. Auf diese Weise erhält man eine Darstellung, die nur Informationen über das Verhalten des Signals im zeitlichen Verlauf enthält.

Verwendet man hingegen Sinussignale als Basisfunktionen, erhält man eine Fourier-Darstellung. Diese Darstellungsweise zeigt das Frequenzverhalten des Signals (siehe Kapitel über JPEG/DCT).

Keine der beiden soeben genannten Darstellungsmöglichkeiten ist für die Komprimierung von Bildinformationen ideal, da man eine Repräsentation benötigt, die Informationen sowohl über das zeitliche Verhalten als auch über das Frequenzverhalten des Signals in sich trägt. Man braucht Informationen über das Frequenzverhalten des Signals zu einem bestimmten Zeitpunkt um nach inverser Transformation eine möglichst exakte Reproduktion des Ausgangssignals zu erhalten.

Glücklicherweise treten in fotographischem Bildmaterial Bereiche mit einem Anteil hoher Frequenzen („Kanten“) räumlich sehr konzentriert auf, während niedrige Frequenzen meist größere Flächen betreffen. Diese Tatsache legt nahe, die hochfrequenten Teile des Bildes „herauszufiltern“ und sie getrennt von den restlichen Informationen in einer geeigneten Weise darzustellen. Mit geeigneten Bandpassfiltern ließe sich so die Bandbreite des Signals stufenweise halbieren.

Aber wie lassen sich nun geeignete Basisfunktionen zusammenstellen?

Die Impulsfunktionen können keine Informationen über das Frequenzverhalten liefern, weil der Bereich

über dem sie nicht Null sind unendlich klein ist. Das zweite Extrem sind die Sinusfunktionen. Sie liefern keine Information über das zeitliche Verhalten, da sie einen unendlichen Definitionsbereich haben. Ein Kompromiß zwischen diesen beiden Extremen scheint ideal: eine Menge von endlich definierten Basisfunktionen $\{\Psi_i\}$, jede mit einem Definitionsbereich unterschiedlicher „Breite".

Um die Sache zu vereinfachen kann angenommen werden, daß alle Basisfunktionen $\{\Psi_i\}$ skalierte und verschobene Versionen ein und derselben Prototypfunktion Ψ, dem Scaling-Funktion oder Mutterwavelet sind. Die passende Skalierung wird durch die Multiplikation von x mit einem Vielfachen von 2^v erreicht. Weil Ψ endlich ist, muß die Funktion um einen Wert k verschoben werden, um das ganze Signal zu erfassen.

$$\Psi(2^v x - k) \quad k \in Z$$

Man erhält die vollständige Wavelet Dekomposition des Signals:

$$f(x) = \sum_{v \text{ endl.}} \sum_{k \text{ endl.}} c_{vk} \Psi_{vk}(x)$$

Bisher wurde noch nichts über die Berechnung der Koeffizienten c_{vk} gesagt. Diese werden in der Wavelet-Transformation berechnet und sind das innere Produkt des Signals $f(x)$ und den Basisfunktionen $\Psi_{vk}(x)$. Wavelet-Transformationen können tatsächlich als octave Bandpassfilter angesehen und implementiert werden, aus diesem Grunde werden sie im folgenden auch so behandelt [HiJaSe 94].

Für den Transformationsschritt innerhalb der Bildkompression benötigt man ein System von orthogonalen Funktionen. Erst ein Paar von orthogonalen Funktionen ermöglicht in seiner inversen Anwendung, das Eingangssignal verlustfrei wiederherzustellen. So ein Filterpaar wird als „Quadrature Mirror Filter" (QMF) bezeichnet. Diese „Hoch- und Tiefpaßfilter" ließen sich rekursiv wie folgt definieren:

$$T(x) = \sum_{k=0}^{M-1} c_k T(2x - k) \quad \text{und} \quad H(x) = \sum_{k=0}^{M-1} (-1)^k c_{1-k} T(2x - k)$$

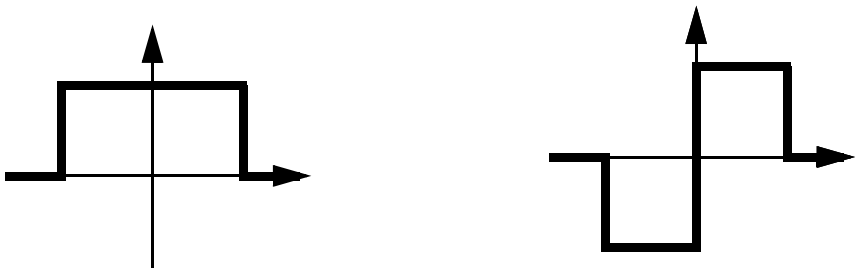
mit $T(x)$ = Scaling-Funktion (oder Mutter-Wavelet)

c_k = Filterkoeffizienten

Die Anzahl der Filterkoeffizienten bestimmt die Ordnung des Wavelets und somit die Anzahl der Verschiebungen über die x-Achse. Die Ordnung legt fest wieviele Samples in der Umgebung des aktuell zu beschreibenden Punktes betrachtet werden [Ed 91].

3.1.1 Beispiel: Haar-Wavelet

Als sehr einfaches Beispiel soll hier das sogennannte Haar-Wavelet dienen. Es hat in der Praxis der Bilddatenkompression keine besonders große Bedeutung. Mit komplexeren Wavelets lassen sich Effekte erzielen, anhand derer eine effizientere Quantisierung und damit eine entsprechend höhere Kompression erreichbar ist. Zu Zwecken der Erläuterung ist dieses Beispiel jedoch ideal, da es in dieser Form ein Interpolationsschema repräsentiert, das geläufig ist.



Haar-Scaling Funktion Haar-Wavelet

Abbildung 3-1: Haar-Scaling-Funktion und Haar-Wavelet

Im folgenden wird ein Haar-Wavelet der Ordnung zwei zugrundegelegt, d.h. bei der Faltung während der Berechnung der Koeffizienten werden jeweils nur zwei benachbarte Samples des Eingangssignals betrachtet. Hier läßt dich anschaulich das Interpolationsschema erkennen. Das Signal wird zum einen in ein Duchschnittssignal (hier: Durchschnitt je zweier benachbarter Werte) und zum anderen in ein Differenzsignal (hier: Differenz je zweier benachbarter Werte) gewandelt. Die zwei Ausgangssignale besitzen jeweils nur halb so viele Koeffizienten wie das Eingangssignal Samples enthielt.

Ein Sinussignal, gesampled über 16 Punkte, soll im folgenden Beispiel gefiltert werden (siehe Abb.3-2). Nachdem die Daten durch die Filterfunktionen gelaufen sind, entsteht, wie oben angedeutet, ein Tiefpaßgefilterter Anteil (links), dem Durchschnitt jeweils zweier Samples, und einem Hochpaßgefiltertem Anteil (rechts), der Differenz je zweier Samples. Das Ergebnis des Hochpaßfilers enthält offensichtlich weniger Informationen als das tiefpaßgefilterte Signal.

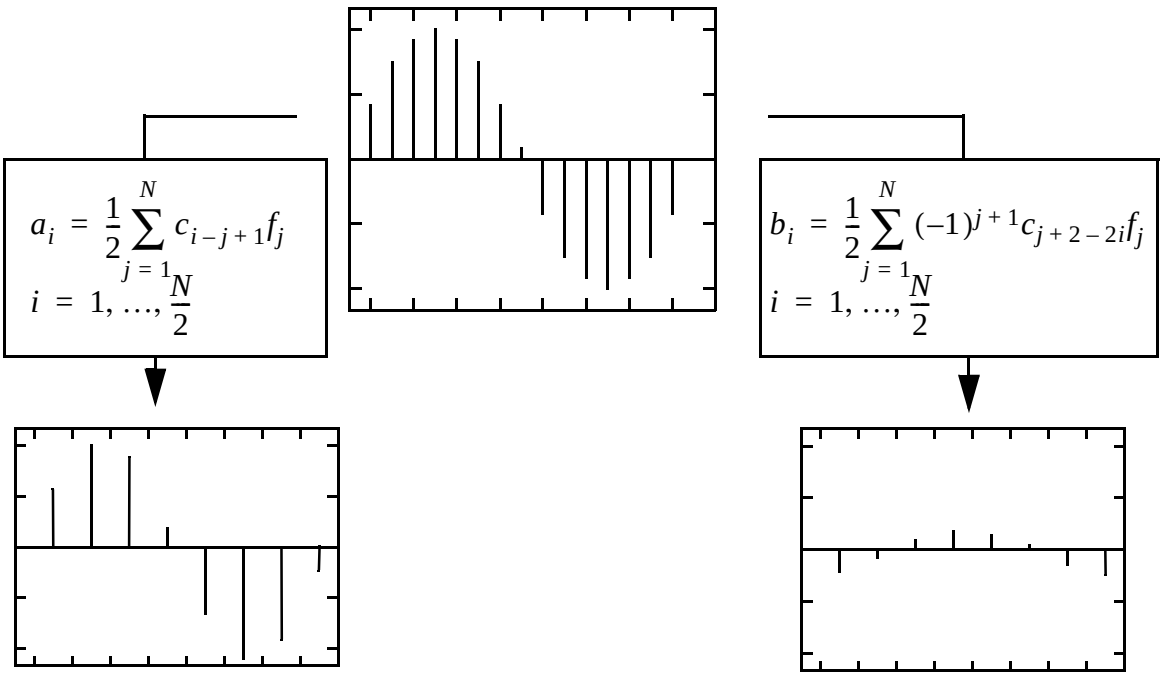


Abbildung 3-2: Vorwärts-Wavelet-Transformation mit Haar-Wavelet zweiter Ordnung

Wenn das Signal allein durch den inversen Tiefpaßfilter rekonstruiert wird, besteht das Ergebnis aus Duplikaten jedes Koeffizienten der Vorwärtsfilterung (links). Das entspricht einer Wavelet-Kompression der Rate 2:1.

Eine perfekte Rekonstruktion des Eingangssignals ist erst durch die Summe der invers gefilterten Signale zu erreichen.

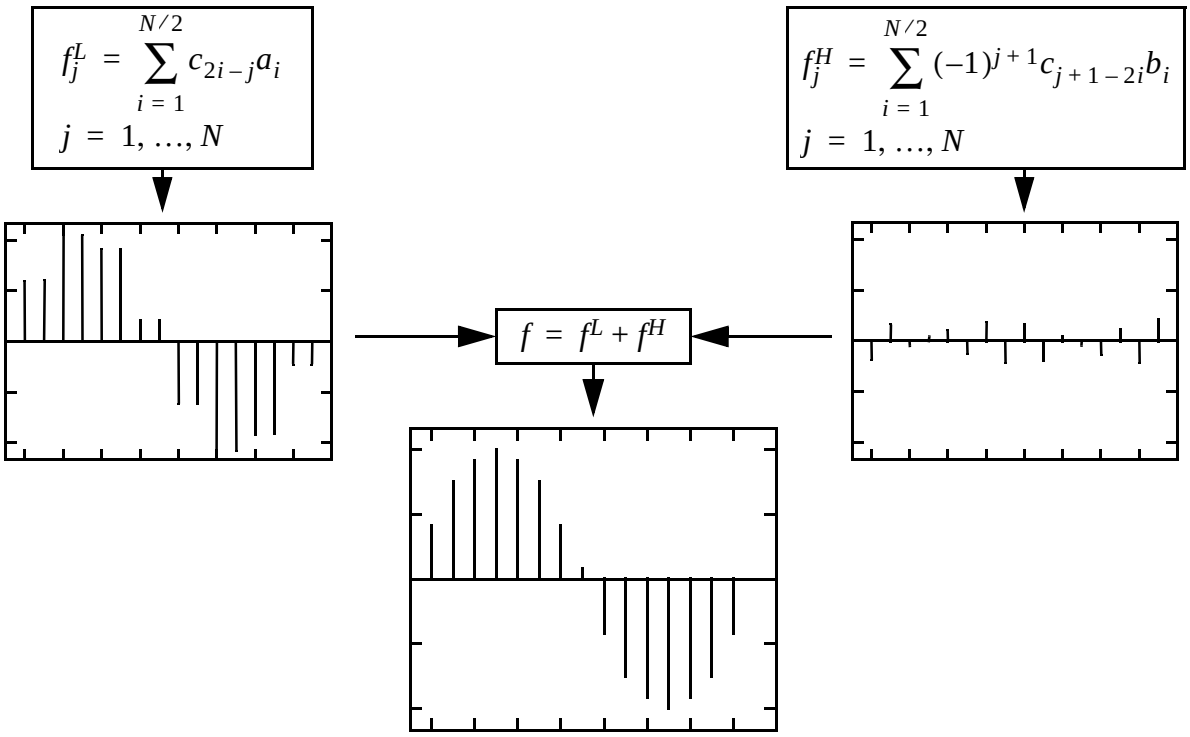


Abbildung 3-3: Inverse Wavelet Transformation

Werden andere Koeffizienten und Wavelets anderer Ordnung benutzt, sind die Ergebnisse immer noch mit denen dieses einfachen Beispiels vergleichbar. Die Ausgangssignale sind dann allerdings nicht mehr genau die Durchschnitts- und Differenzsignale, sondern können den Bedürfnissen der Bildkompression angepaßt werden. Ein wichtiger Schritt ist das Finden von Wavelet-Funktionen, die beim Hochpaßgefilterten Signal viele sehr kleine Werte erzeugen (wichtig für den anschließenden Quantisierungsschritt (siehe Kap. 3.2) [Ed 91].

3.1.2 Weitere Beispiele

Normalerweise wird die Transformation mit anderen Funktionspaaren durchgeführt. Diese spezielleren Funktionen, wie Daubechies Wavelets oder biorthogonale Wavelets, führen erst zu gegenüber JPEG überlegenen Resultaten.

Es folgt eine Aufstellung einiger Filterklassen, die in der Bilddatenkompression größere Bedeutung haben. Diese Aufstellung soll einen Eindruck über die vielfältigen Gestaltungsmöglichkeiten beim Design der Wavelets vermitteln. Die Wavelets sind oft sehr speziell und werden durch experimentelle Betrachtungen an bestimmte Quantisierungsfunktionen angepaßt.

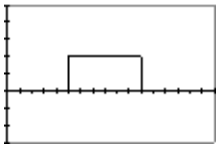
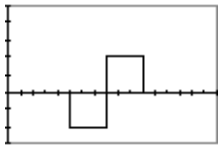
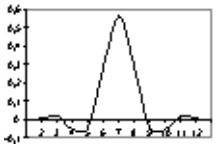
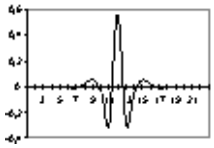
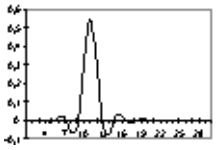
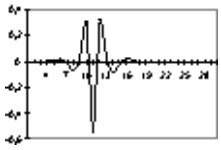
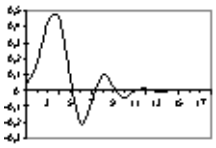
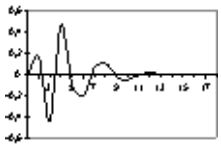
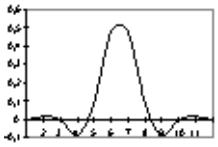
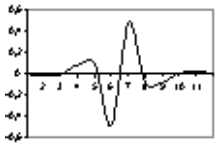
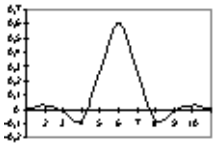
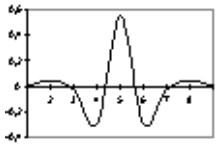
Filterklasse	Scaling-Funktion	Wavelet	Beschreibung
Haar			Einfachste Filterklasse, Mittelwert- und Differenzfilter
CloseToCoiflet			
Coiflet			Frühe Waveletklasse, entwickelt von R. Coifman
Daubechies			nicht symetrisch, streng orthogonal, strenger „compact support“ selbstähnlich
Johnston-Barnard			
Biorthogonal-Spline			Symetrisch, werden aus Binomialkoeffizienten berechnet

Abbildung 3-4: Filterfunktionen [LuRa 99]

3.2 Die drei Phasen der Kompression

Es gibt eine Vielfalt verschiedener Kompressionsschemata, die die Wavelet-Transformation benutzen. Alle diese Schemata können jedoch in einem recht allgemeinen Rahmen gemeinsam beschrieben werden. Die Komprimierung wird durch Anwendung der Wavelet-Transformation, Quantisierung der resultierenden Koeffizienten und schließlich der Kodierung des Ergebnisses erreicht. Die Rekonstuktion des Ausgangsbildes ist einfach durch entsprechende inverse Operationen zu erreichen. Allein der Quantisierungsschritt läßt sich nicht verlustfrei umkehren. An dieser Stelle lassen sich während der Kompression

also durch bewußtes Weglassen „unwichtiger“ Informationen höhere Kompressionsraten erzielen. In den folgenden Abschnitten werden die einzelnen Kästchen des folgenden Diagramms ausführlich beschrieben.

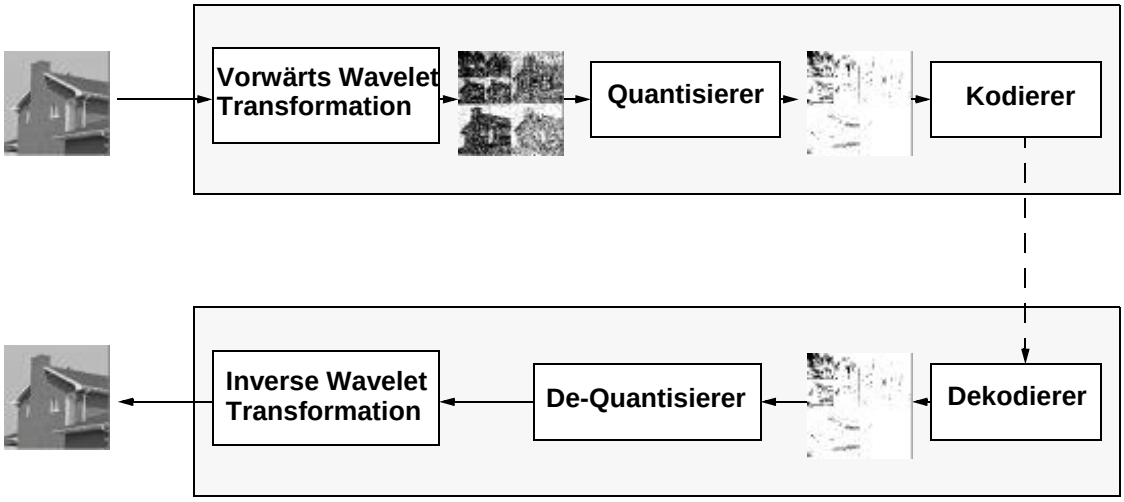


Abbildung 3-5: Block-Diagramm eines Wavelet-basierten Kodierers

3.2.1 Dekorrelation der Bilddaten mittels Wavelet-Transformation

Die Wavelet-Transformation zerlegt das Bild in hoch- und tiefpaßgefilterte Anteile. Es ergeben sich vier verschiedene „Versionen“ des Gesamtbildes: drei verschiedene „Detailsignale“ und ein „Durchschnittssignal“, das für eine weitere Iteration der Transformation verwendet wird.

Die Transformation läßt sich in vier Schritten beschreiben:

1. Zeilenweise Transformation mittels geeignetem (QMF)-Paar; durch Faltung der Filterkoeffizienten c_i mit dem Eingangssignal $f(x)$ ergibt sich wieder eine Amplitudendarstellung.

$$f'(x) = \sum_i c_i f(x-i)$$

2. Da die Bandbreite bezüglich der Zeitachse nur noch halb so groß ist, kann jede zweite Spalte verlustfrei entfernt werden. Es entsteht ein Bild mit den “Kanten” und ein Bild mit den “Flächen” des Ausgangsbildes.
3. Spaltenweise Transformation der beiden Bilder aus (1) mittels QMF-Paar:

$$f'(y) = \sum_i c_i f(y-i)$$

4. Löschen jeder zweiten Zeile. Es entstehen vier Bilder; ein Bild (Tiefpaß/Tiefpaß) ist das Hauptbild (Durchschittssignal), das für die nächste Iteration verwendet wird, die anderen drei enthalten Detailsignale (siehe Abb. 3-6).

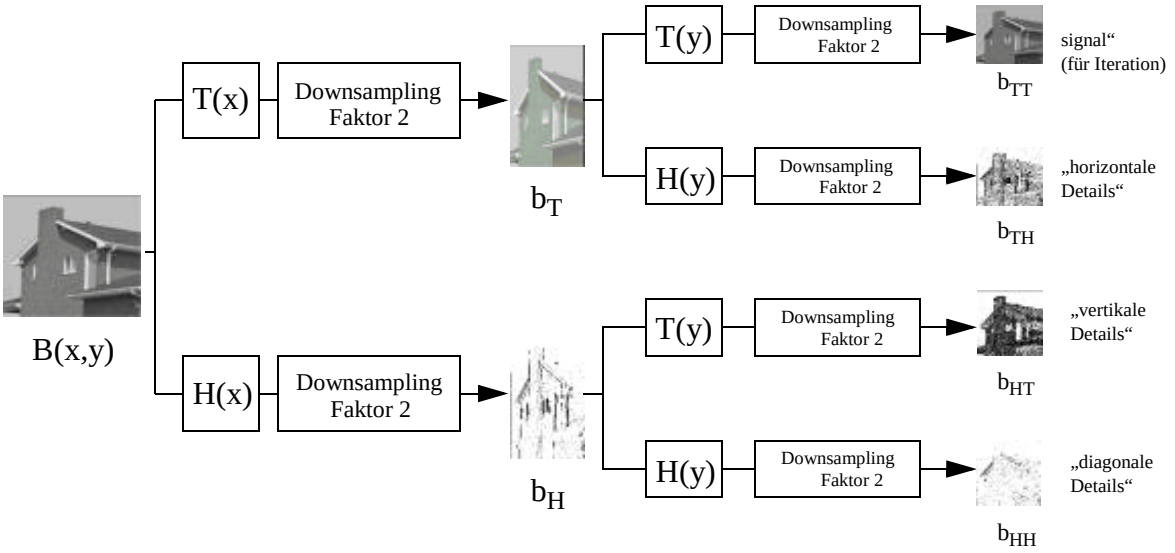


Abbildung 3-6: Transformationsschritt

Die zuvor beschriebenen vier Schritte können iteriert werden, um weitere Details herauszufiltern. Hierzu wird das Ergebnis nach zweimaligem Tiefpaßfilter als Eingangssignal benutzt. Bereits an dieser Stelle kann die Kompressionsrate durch die Wahl der Anzahl der durchzuführenden Iterationen beeinflußt werden. Theoretisch kann dieses Verfahren solange iteriert werden bis das Signal b_{TT} nur noch aus einem Sample besteht.

3.2.2 Quantisierung der erhaltenen Koeffizienten

Die Wavelet-Transformation dekorreliert die Pixelwerte des Originalbildes und konzentriert die Bildinformation in einer relativ kleinen Anzahl von Koeffizienten. Ein erster „Komprimierungseffekt“ der Wavelet-Transformation ist durch die kleine Anzahl von Koeffizienten mit großen Werten ungleich null ersichtlich. Diese Tatsache bildet eine gute Vorausstzung für eine Quantisierung dieser Werte.

An dieser Stelle findet die eigentliche Kompression statt. Die Qunatisierungsfunktionen nehmen direkt Einfluß auf die Kompressionsrate und die Qualität des rekonstruierten Bildes. Die Quantisierungsfunktionen bestehen in der Regel einerseits aus einer Schwellenfunktion, die Koeffizienten nahe null einfach auf null setzen und zum anderen aus der eigentlichen Quantisierung durch Runden der restlichen Werte.

Die Quantisierungsfunktionen werden zusammen mit der Wahl eines bestimmten Wavelets festgelegt. Bei der Filterung unterschiedlicher Frequenzanteile haben sich verschiedene Quantisierungsfunktionen als nützlich erwiesen. So ist es üblich für die Ergebnisse der unterschiedlichen Iterationen der Transformation auch unterschiedliche Quantisierungsfunktionen zu benutzen [HiJaSe 94].

3.2.3 Codierung der quantisierten Werte

Der nach dem Quantisierungsschritt vorhandene Bitstrom kann nun mit den bereits bekannten verlustfreien Kompressionsalgorithmen behandelt werden. Es ist üblich Lauflängenkodierung sowie einen Huffmannkode einzusetzen. Es gibt allerdings auch neuere Algorithmen, die speziell auf die Eigenschaften des Bitstroms zugeschnitten sind (z.B. Shapiro’s Zero Tree Encoding). Solche Algotithmen sind allerdings bezüglich Rechen- und Speicherbedarf sehr anspruchsvoll [HiJaSe 94].

3.3 Rekonstruktion des Ausgangsbildes

Die Rekonstruktion wird durch zu den einzelnen Kompressionsschritten inverse Operationen realisiert. Allein der Quantisierungsschritt läßt sich nicht verlustfrei umkehren. Die inverse Wavelet-Transformation ist in Abb. 3-7 schematisch dargestellt. $T^i(x)$ und $H^i(x)$ sind die inversen Tief- bzw. Hochpaßfilter. Das Upsampling wird einfach durch das Einfügen von Nullwerten zwischen je zwei Koeffizienten erreicht.

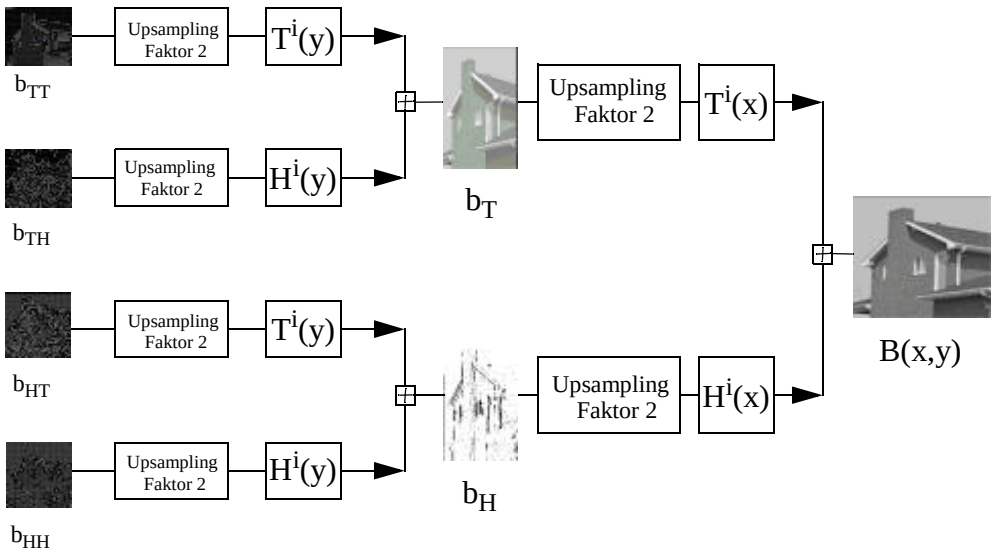


Abbildung 3-7: Inverser Transfortmationsschritt

4 Fraktale Komprimierung

Im Gegensatz zu anderen Verfahren ist die fraktale Kompression das Ergebnis der Arbeit von lediglich zwei Autoren: 1988 wiesen Michael Barnsley und Alan Sloan die Möglichkeit einer solchen Bildkompressionsmethode nach. Die erste Implementierung legte Arnaud Jaquin 1989 vor. Patentinhaber sind allerdings Barnsley und die Firma Iterated Systems Inc. von Barnsley und Sloan. Die folgende Beschreibung des Kompressions- und des Dekompressionsvefahrens beruht auf dem Algorithmus von Barnsley sowie einiger Erweiterungen der Skalierungen, Transformationen und Suchstrategien.

4.1 Grundüberlegung / IFSs

Fraktale Bildkompression, wie die Wavelet-Kompression auch vom Typ „lossy“, basiert auf der fraktalen Geometrie. Grundlage des Verfahrens ist die Beobachtung, daß in dem meisten Bildmaterial sogenannte Selbstähnlichkeiten auftreten. D.h. ob nun Zeichnungen, die vorwiegend mit Elementen der klassischen Geometrie arbeiten oder fotographischen Material, es gibt Bereiche im Bild, die in ähnlicher Form an einer anderen Stelle noch einmal oder ähnlich vorkommen.

Die Strategie der fraktalen Bilddatenkompression ist es solche Selbstähnlichkeiten innerhalb des Bildes zu finden und zu beschreiben. Wenn die Beschreibung vollständig ist, d.h. das komplette Bild wurde durch andere, ähnliche Bildbereiche beschrieben, läßt sich allein aus dieser Information das Ausgangsbild rekonstruieren. Die Beschreibung der einzelnen Bildbereiche durch andere, größere Bildbereiche wird als Funktion angesehen. Das resultierende Funktionensystem kann nun iteriert auf ein beliebiges

Bild von derselben Ausgangsgröße angewendet werden. Es entsteht eine stufenweise Annäherung an das Ausgangsbild.

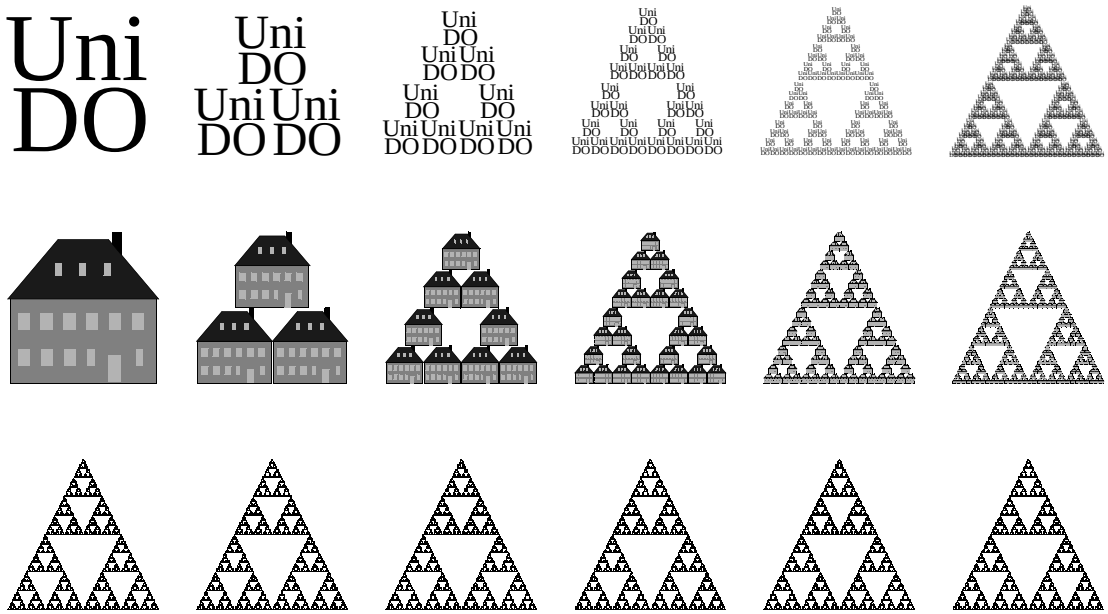


Abbildung 4-1: Iterierte Anwendung einer Funktion auf verschiedene Ausgangsbilder

In Abb. 4-1 läßt sich erkennen, daß das Resultat einer iterierten Anwendung einer Funktion (hier: Skaliere das Ausgangsbild um 50% und kopiere es dreimal in den neuen Bildbereich) nicht vom Ausgangsbild abhängt, sondern allein von der Funktion. Bereits nach wenigen Iterationen stellt sich eine gute Annäherung ein, die einen ersten Eindruck vom Fixpunkt macht, der schließlich angenähert wird.

So einfach die Idee dieses Verfahrens ist, so kompliziert ist die Realisierung. Das Finden von Selbstähnlichkeiten ist kein einfacher Prozeß. Es gibt unendlich viele Möglichkeiten einen endlichen Bildbereich durch andere Bildbereiche zu beschreiben. Da das Verfahren aber als Ziel hat den Beschreibungsaufwand zu verringern, beschränkt man von vornherein die Menge der erlaubten affinen Transformationen. Die entstehenden Funktionen sollen in ihrem Beschreibungsaufwand deutlich unter dem Beschreibungsaufwand für das Bild in Pixeldarstellung liegen. Darüberhinaus läßt sich erst so eine vertretbare Laufzeit realisieren.

Der Algorithmus von Barnsley und Sloan sieht vor, das Bild gleichmäßig und ohne Überschneidungen in quadratische Blöcke aufzuteilen, die „range-blocks“. Diese Blöcke sollen nun durch andere Blöcke doppelter Größe beschrieben werden. (Durch das Festsetzen der Blockgröße findet hier eine erste Reduktion der Beschreibung statt. Skalierungen sind nun fest und müssen nicht mehr beschrieben werden.)

Die größeren Blöcke („domain-blocks“) dürfen sich überschneiden, so erhöht sich die wahrscheinlichkeit gut zueinander passende Blöcke zu finden (siehe Abb 4-2).

Die Ähnlichkeit zwischen den domain- und range-blocks muß nicht offensichtlich sein und stellt sich oft erst nach Transformationen wie Spiegelung oder Rotation ein. Insgesamt gibt es acht Typen dieser affinen Transformationen. Jeder range-block wird unter Anwendung aller möglichen Transformationen mit jedem domain-block verglichen.

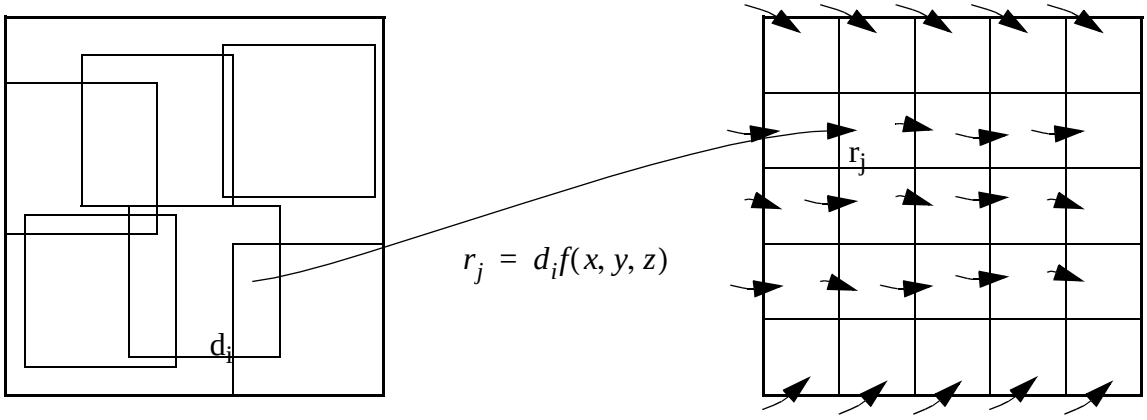


Abbildung 4-2: Repräsentation eines range Blocks r_j durch ein domain Block d_i

Zum Vergleich wird eine Distanzfunktion herangezogen. Als Distanzfunktion für zwei Bilder f, g kann prinzipiell jede Funktion verwendet werden, die die Bedingungen einer Metrik erfüllt, z.B.:

- Pixelweise Maximumsfunktion: $\delta(f, g) = \sup_{(x, y) \in I^2} |f(x, y) - g(x, y)|$

- Summe der Beträge aller Pixelabstände: $\delta(f, g) = \sum_{(x, y) \in I^2} \frac{|f(x, y) - g(x, y)|}{I^2}$

Zwei Blöcke passen dann zueinander, wenn die Distanzwerte zwischen ihnen am geringsten sind. Am Ende dieses Prozesses steht ein Bündel mathematischer Gleichungen, die das Bild repräsentieren.

4.2 Alternatives Mapping

Die oben vorgestellte Methode das Bild in Blöcke aufzuteilen ist eine Einschränkung bezüglich aller Möglichkeiten. Es sind undendlich viele andere Verfahren denkbar, die aus Effizienzgründen aber nicht betrachtet wurden.

Um in detailreichen Bildbereichen eine bessere Annäherung durch domain Blocks zu erreichen gibt es Verfahren, die in solchen Bereichen eine feinere Aufteilung vorsehen. In den folgenden Abschnitten werden vier unterschiedliche Methoden geegnübergestellt [Qua 94].

4.2.1 Quadrate fester Größe

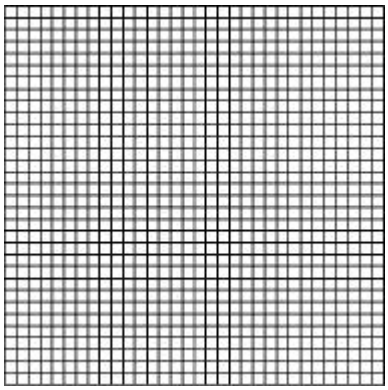


Abbildung 4-3: Partitionierung

Die Partitionierung des Bildes in Quadrate fester Größe stellt die einfachste Art der Partionierung dar. Da Verfahren entspricht dem, das im vorhergehenden Abschnitt beschrieben wurde. Vorteil: Durch die starke Einschränkung bezüglich der möglichen Transformationen ist eine relativ schnelle Berechnung gewährleistet. Nachteil: Nutzt in keinster Weise Redundanzen innerhalb des Bildes aus.

4.2.2 Quadtree

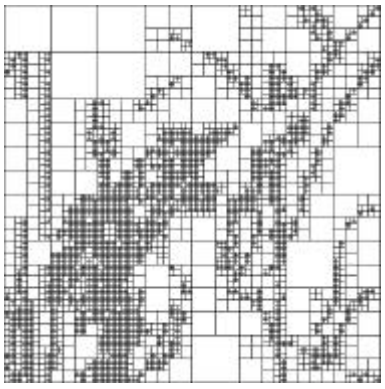


Abbildung 4-4: Partitionierung

Das Quad-Tree-Verfahren teilt das Bild auf in große quadratische Blöcke. Wenn zu den range-blocks kein genügend passender (Schwellenfunktion) domain-block existiert, so findet eine weitere Aufspaltung eines Quadratblocks in vier gleich große quadratische Unterblöcke statt. Dieser Prozeß wiederholt sich, bis zueinander passende Blöcke gefunden werden. Vorteil dieser Methode: Die Zuordnung größerer Flächen mit wenig Detailinformationen passiert schneller bei geringerem Speicherverbrauch. Detailreiche Bilder können aber entsprechend längere Zeiten erfordern, da sehr viele Vergleiche gemacht werden müssen.

4.2.3 HV

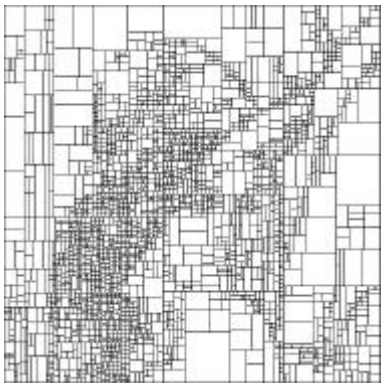


Abbildung 4-5: Partitionierung

Die HV-Methode (horizontal/vertikal) beginnt damit, das gesamte Bild als einen einzigen Block zu betrachten. Domain- und range-blocks bestehen aus der gleichen Menge und werden durch die Aufspaltung vorhandener Blöcke entlang von Kanten entweder in horizontaler oder vertikaler Richtung erzeugt. Die Aufspaltung wird solange fortgesetzt bis zueinander passende Blöcke gefunden werden. Der Vorteil der besser zueinander passenden Blöcke wird durch den zusätzlichen Rechen- und Beschreibungsaufwand (z.B. durch Streckungen und unterschiedliche Skalierungen) relativiert. Das rekonstruierte Bild ist allerdings im Vergleich zu Quadtree meist von höherer Qualität.

4.2.4 Triangular

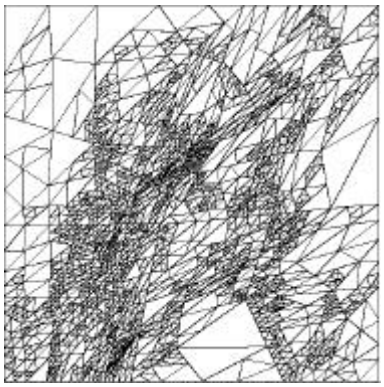


Abbildung 4-6: Partitionierung

Die triangulare Aufteilung geschieht ganz ähnlich dem HV-Verfahren. Es werden hier Dreiecke benutzt, die so zusätzlich Kanten erfassen können, die nicht genau horizontal bzw. vertikal verlaufen. Hier werden in der Regel die besten Ergebnisse erzielt, der benötigte Rechenaufwand ist jedoch enorm groß.

4.3 Rekonstruktion des Ausgangsbildes

Die fraktale Bildkompression ist ein asymmetrischer Prozeß. Anfangs dauerte ein fraktaler Kompressionsprozeß Stunden bis Tage, mittlerweile schaffen das clevere Algorithmen in Minuten. Die Dekompression ist dagegen rasend schnell. In weniger als 16 Iterationen ist die Bildrekonstruktion abgeschlossen. Die Wiederherstellung beginnt gewöhnlich mit einem grauen Anfangsbild, das lediglich die gleiche Größe wie das Originalbild besitzen muß. Die affinen Transformationen werden nun in umgekehrter Richtung abgearbeitet, in dem die vorhandenen Gleichungen immer wieder auf das Ergebnis der vorhergehenden Iteration angewendet werden. Es stellt sich allmählich das Verhältnis von domain-

zu range-blocks wieder ein. Dieses Verfahren wird so lange wiederholt, bis das nach einigen Iterationen erzeugte Bild sich nicht mehr vom Ergebnis der nächsten Iteration unterscheidet.

4.4 Anwendung der fraktalen Kompression

4.4.1 Ein einfaches Beispiel

Es folgt eine Beschreibung eines Komprimierungsvorgangs, der ein Bild mit 256 Farben und einer Größe von 256×256 Punkten bearbeitet. Es wird das einfachste Partitionierungsverfahren benutzt, die das Bild in Quadrate fester Größe aufteilt, wobei die Größe der Quadrate 8×8 Punkte sei. In der Summe ergibt das 1024 kleine Quadrate als range-blocks. Die domain-blocks sollen eine Größe von 16×16 Punkten besitzen; es werden alle möglichen Quadrate (mit Überschneidungen) betrachtet. Das ergibt $241 \cdot 241 = 58081$ domain-blocks.

Durch die Partitionierung in Quadrate sind nur noch Drehungen um 90° und Spiegelungen an der X- oder Y-Achse erlaubt. Außerdem ist der Skalierungsfaktor mit 25% fest vorgegeben. Es gibt also nur noch 8 mögliche Transformationen.

Es wird zunächst jeder range-block auf die Größe eines domain-blocks gebracht, jeder Pixel wird also vervierfacht. Es werden die Faktoren für die Helligkeits- und Kontrastanpassungen berechnet und angewendet. Anschließend wird für jede der 8 möglichen Transformationen der Abstand berechnet. Nach einem Vergleich der Abstände wird die Transformation mit geringstem Abstand gespeichert.

Insgesamt ergeben sich $1024 \cdot 58081 \cdot 8 = 475799552$ Transformationen und Bildvergleiche, zusätzlich wurden $1024 \cdot 58081 = 59474944$ Helligkeits- und Kontrastanpassungen vorgenommen. Der Aufwand ist also trotz der vereinfachten Voraussetzungen (relativ kleines Graustufenbild, Verwendung des einfachsten Partionierungsschemas, starke Einschränkung des erlaubten Quellbereichs) erheblich.

Zur Kompressionsrate: Die Quelldatei hat eine Größe von $256 \cdot 256 = 65536$ Byte, die komprimierte Datei besteht aus 1024 Transformationen mit jeweils 16 Bit für den Quellbereich, 3 Bit für die Transformation (da nur 8 verschiedene erlaubt sind) und 12 Bit für die Helligkeits- und Kontrastanpassung (gute Erfahrungswerte), also insgesamt 31 Bit. Sie hat eine Dateigröße von 3968 Byte, was einer Komprimierungsrate von 16,5:1 entspricht.

5 Vergleich verschiedener Kompressionsverfahren

	JPEG	Fraktal	Wavelet
verlustfrei?	nein	nein	nein (meist)
Strategie	Blocking	Blocking	Non-Blocking, Blokking möglich
Methode	DCT	IFS (iterierte Funktionensysteme)	versch. Basisfunktionen und Filtertypen
bevorzugtes Bildmaterial	Realbilder in Echtfarben	Realbilder	alle
weniger bevorzugtes Bildmaterial	Grauwertbilder, Binärbilder	abstrakte Bilder	abh. vom verwendeten Wavelet-Typ, teilweise Probleme mit abstrakten Bildern
abstrakte Bilder	schlecht bis mittel	gut bei wenigen Texturen	gut bis mittel (abh. vom verwendeten Wavelet-Typ)
Realbilder, niedrige Kompression	sehr gut	gut	sehr gut (abh. vom Typ)
Realbilder, hohe Kompression	ab 25:1 deutliche Artefakte	sehr gut	sehr gut (abh. vom Typ)
Zeitverhalten	symmetrisch; im Sekundenbereich	asymmetrisch: Kompression langsam, Dekompression sehr schnell	symmetrisch: im Sekundenbereich
Standards	JPEG; zwei Formate (JFIF, JPEG/TIFF)	-	-

Die hier beschriebenen Ergebnisse sind zum Teil stark von der Implementation abhängig, zeigen aber die wichtigsten Tendenzen [SiBe 96].

6 Literatur

[Ed 91] Edwards, T.: Discrete Wavelet Transforms: Theory and Implementation, Stanford University, September 1991

[EfSt 98] Effelsberg, W., Steinmatz,R.: Video Kompression Techniques (inkl. CD-ROM), dPunkt Verlag, 1998

[HiJaSe 94] Hilton, M.L., Jawerth, B.D., Sengupta, A.: Compressing still and moving Images with Wavelets, Multimedia Systems, Vol.2, No.3, 1994; paper supported by Summus, Ltd.

[LuRa 99] LuRaTech Homepage, Informationen zur Waveletkomprimierung: http://www.luratech.com/knowhow/knowhow01_g.html (Stand: 28.05.99)

[Qua 94] Quante, J.: Ftaktale Kompression, Beitrag zm Seminar „Multimedia Datenformate“, Universität Karlsruhe, Institut für Betriebs- und Dialogsysteme, WS 94/95

[SiBe 96] Simon, U., Berdtge, M.: Handlich Bunt - Kompressionstechniken für Bild- und Videodateien im Vergleich, c’t 11/1996, Heise Verlag, S. 222-233

[Stei 94] Steinwandt, R.: Wavelet-Kompression, Beitrag zm Seminar „Multimedia Datenformate“, Universität Karlsruhe, Institut für Betriebs- und Dialogsysteme, WS 94/95

[Su 99] Summus - Wavelet Theory, http://www. summus.com/publish/wavelets/ (Stand: 28.05.99)

JPEG Kompression

Mike Ndambuki

FB Informatik, Uni Dortmund
Email-Ndambuki@ls1.informatik.uni-dortmund.de

Zusammenfassung

JPEG ist ein Werkzeugkasten zur Komprimierung von digitalisierten Standbildern in Echtfarb-Qualität. Die in JPEG enthalten Algorithmen ermöglicht Entwicklern unterschiedliche Anwendungen zu programmieren. Die benötigten Teile werden herausgenommen, um in ihrer Software eingesetzt zu werden.

Als solches ist JPEG auch als ein Werkzeug zu interpretieren, da es anhand von Eingabe unterschiedlicher Parameter unterschiedlichen Bildkompressionen ermöglicht, z.B. eine Kompressionsrate. JPEG ist das meist verbreitete Standardkompressionsverfahren, insbesondere der verlustbehaftete Modus, der auf der DCT-Transformation basiert ist.

Durch die Anwendung der DCT-Transformation hat JPEG eine besondere Brisanz im Vergleich zu anderen Kompressionsverfahren. JPEG wird mittlerweile nicht nur für Stand- bilder benutzt, sondern auch für bewegte Bilder. Dieses Referat beschäftigt sich mit JPEG-Komprimierungsverfahren als solches und insbesondere als Schwerpunkt des verlustberhafteten Kodierungsverfahrens.

1 Einführung in JPEG

JPEG steht für Joint Photografic Expert Groups. Das Wort Joint wurde gewählt, weil es aus einer gemeinsamen Aktivität der ISO/IEC, JJC1/SC2/WG10 und der Kommission Q.16 der CCITTSGVII. resultiert. Die Entwicklung von JPEG begann 1982 in einer Arbeitsgruppe(Working Group 8) der Firma ISO, als es darum ging, einen Standard zur Kompression und Dekompression von Standbildern zu erstellen. 1987 setzte sich JPEG von anderen zehn Kompressionsverfahren durch und ist seit 1992 ein ISO-Internationalstandard (IS)

JPEG ist ein Kompressions- und Dekompressionsverfahren für Standbilder und zwar für Farb und grauskalierte Standbilder.

Durch eine schnelle Kodierung und Dekodierung werden Einzelbilder zu bewegten Bildsequenzen bearbeitet. Dies nennt man Motion JPEG, weiter Entwicklungen von JPEG Kompressionsverfahren für bewegte Bilder MPEG-1 und MPEG-2.

JPEG wird teilweise als Teilsoftware oder mit spezieller Hardware vermarktet, wobei diese Systeme nur den notwendigen Algorithmus unterstützen (Basis Mode). Dies hat die Auswirkung, dass nur ein Teil von JPEG kommerziell erhältlich ist.

Um die Standardbasis von JPEG zu ermöglichen, sowie eine weite Verbreitung sowie Anwendung zu erreichen, sind folgende Anforderungen [Wall 91] erfüllt worden:

I.) Unabhängigkeit in folgenden Kriterien:

- Bildgröße d.h.. es muss möglich sein, unterschiedliche Bildgrößen zu komprimieren und zu dekomprimieren.

- Farbraum und Farbvielfältigkeit, d.h. der verwendete Farbraum muss unabhängig von der Farbvielfalt sein.

II.) Unabhängigkeit der Komplexität:

- Kompressionsgrad und erreichbare Bildqualität müssen dem momentanen Standard der Technik entsprechend sein.

- Eine Softwarelösung muss auf eine große Anzahl verfügbarer Standardprozessoren zugeschnitten sein und soll mit Spezialhardware verringert werden.

III.) Mit Hilfe von Parameterangaben soll der Komprimierungsprozeß beeinflussen werden und zwar z.B. Dauer der Kompression und die Größe des zu komprimierenden Bildes.

1.1 Kodierung und Dekodierung von Einzelbildern in JPEG

In vielen Anwendungen ist nicht festgelegt, daß Kompressionsverfahren unbedingt mit Kodierungs- bzw. Dekodierungsschritten erfolgen muß. Der JPEG-Baselinecode setzt aber voraus, daß die beiden Kompressionsschritte erfolgen müssen. Zuerst werden die Daten kodiert und danach dekodiert.

In der Kodierungsphase wird ein Bild in einem bestimmten Format zusammengesetzt und durch eine Transformation (DCT) werden die Daten in Signale umgewandelt zur Koeffizienten und anschließend quantisiert. Dann erfolgt die Kodierung anhand unterschiedlicher Algorithmen.

Danach wird der Prozeß der Dekodierung fortgesetzt, wo es zur erst einer Kodierung stattfindet gefolgt von einer Dequantisierungsprozeß zum Schluß folgt einer Transformation der Daten anhand von IDCT (Umkehr Transformation von DCT).angewendet.

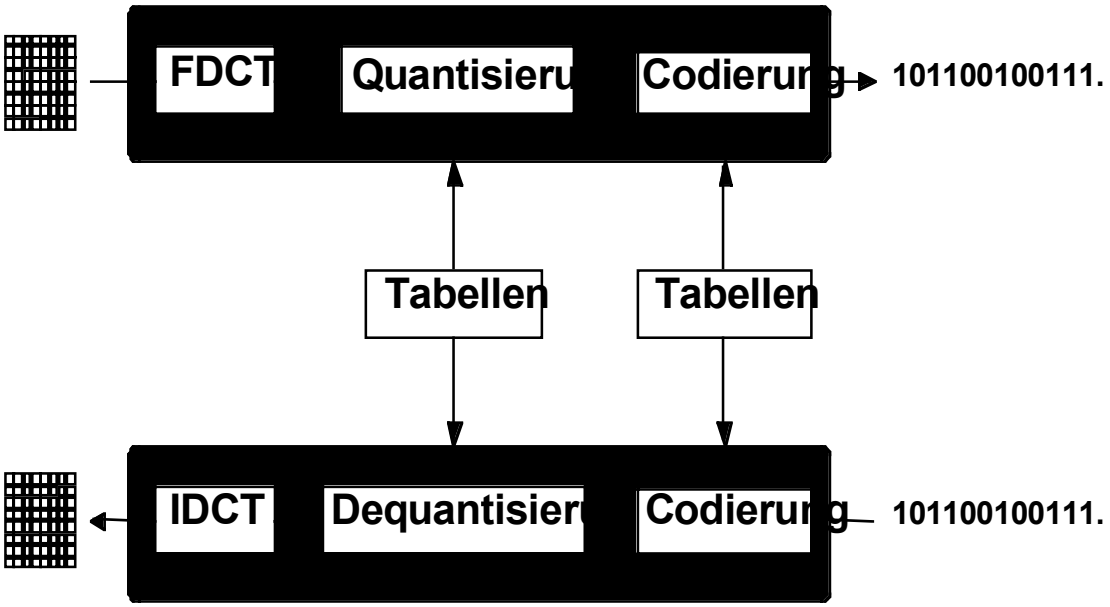


Abbildung 1-1: Kodierung und Dekodierung beim Baseline Codec [Bau 94]

Bei Kompressionsverfahren unterscheidet man zwischen

I.) Verlustbehaftete Kompression

II.) Verlustfreie Kompression

1.1.1 Verlustbehaftete Kompressionen

Unter verlustbehafteten Kompressionen versteht man Kompressionen, die einen Informationsverlust enthalten, d.h. die dekodierten Bilddaten entsprechen nicht mehr bitgenau den originalen Bilddaten. Alle verlustbehafteten Kompressionen in JPEG basieren auf der DCT-Transformation (Discrete Cosinus Transformation).

Der DCT zählt zu den Fouriertransformationen, die unterschiedliche Bildsignale in unterschiedliche Amplituden und Frequenzen darstellen können.

In JPEG beträgt die Auflösung in verlustbehafteter Anwendung entweder 8 oder 12 Bit pro Pixel und Farbkanal.

Anhand von Eingaben eines Parameters (Kompressionsfaktor) unterschiedlicher Größe kann der Benutzer die Qualität, sowie auch die Größe der Datenmenge beeinflussen.

Bei einer Kompressionsrate von 20:1 sind kaum wahrnehmbare Unterschiede zum Originalbild zu sehen, obwohl das Bildmaterial einen Informationsverlust enthält.

Verlustbehaftetete Kompressionsverfahren werden hauptsächlich in Bildbearbeitungssystemen sowie im Internet und der Videobearbeitung angewendet.

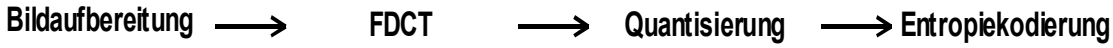


Abbildung:1-2: Kodierungsschritte in verlustbehafteter JPEG Kompression

1.1.2 Verlustfreie Kompression

Die Bilddaten werden so komprimiert, daß nach der Dekompression die Bilddaten bitgenau den Originaldaten entsprechen.

In JPEG nennt man dies „Lossles Mode“. Die anfallenden Daten werden zuerst in eine Menge von Deskriptoren transformiert. Für diese Deskriptoren wird ein statisches Modell (Verschlüsselungstabelle) erstellt, und danach werden die Deskriptoren anhand des Modells kodiert.

Für die Kodierung verwendet man die „Huffman-Codierung“ und die binäre arithmetische Algorithmen

Kodierung. Das verlustfreie Komprimierungsverfahren findet Anwendung, wo kaum Bit-Verluste von Daten tolerierbar sind, wie in der Medizin, der Textverarbeitung und maschinellen Bildauswertungssystemen.

Im Vergleich mit verlustbehafteten Kompressionen ist hier der Kompressionsrate nicht hoch, eine Kompressionsrate von 2:1 ist üblich. Die Auflösung beträgt entweder minimal 2 oder maximal 12 Bit pro Pixel.

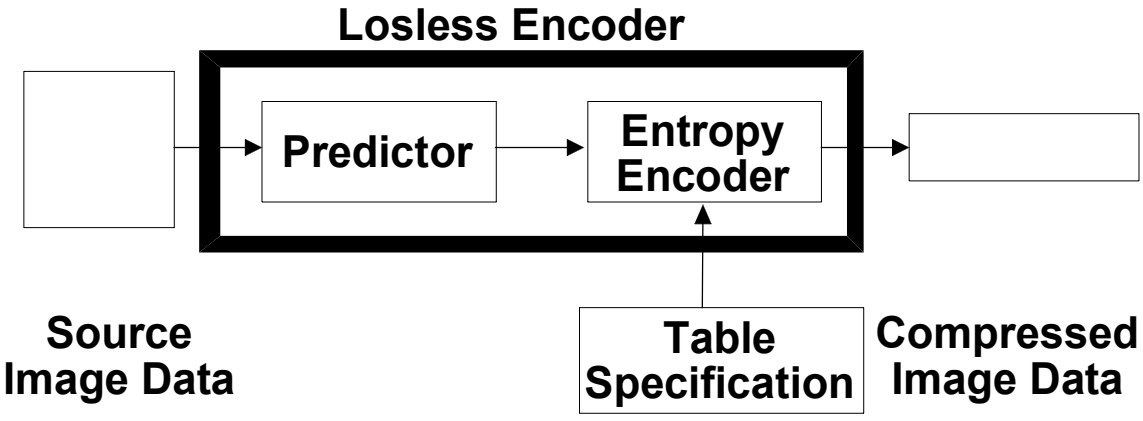


Abbildung 1-3: Verlustlose Kodierung [Wal 91]

2 JPEG Übersicht

In diesen Kapital werden die unterschiedlichen JPEG-Modi kurz beschrieben. Danach wird Ausführlich die verlustbehaftetet Kompression in JPEG beschrieben.

2.1 Die Unterschiedlichen JPEG-Modi

Es gibt in JPEG drei unterschiedliche Bildverarbeitungsmechanismen (Modi), die alle für unterschiedliche Anwendungen geeignet sind. Sie können sowohl in verlustbehafteter oder verlustloser Komprimierung von Daten eingesetzt werden. Es handelt sich um:

I.) Sequentiellen Bildverarbeitungsmechanismus

II.) Progressiven Bildverarbeitungsmechanismus

III.) Hierarchischen Bildverarbeitungsmechanismus

2.1.1 Sequentielle Bildbearbeitung in JPEG

Der sequentielle Bildbearbeitungsmechanismus ist das am meisten verbreitete Komprimierungsverfahren bei der Einzelbilder-Komprimierung. Die zu komprimierenden Daten werden Zeile für Zeile von links oben nach rechts unten aufgearbeitet. Das Bild wird dadurch Schritt für Schritt aufgebaut.

Bei etwas langsamen Rechnern kann der Anwender diesen Prozeß verfolgen. Bei Texten beobachte man, daß die Text von links nach rechts aufgearbeitet wird und zwar Zeile für Zeile. Bei Bildern beobachtetet man den gleichen Prozeß.

Besteht ein Objekt aus mehreren Komponenten, werden die Komponenten überlappt behandelt und nicht einzeln. Die Überlappung benötigt nur eine kleine Speicherkapazität (Puffer). Ein wichtiger Vorteil des sequentiellen Bildbearbeitungsmechanismus besteht darin daß ein Teil der Ergebnisse weitergegeben werden können, ohne unbedingt darauf zu warten, bis alle Komponenten aufgearbeitet worden sind. Sie liefert die beste Kompressionsrate und ist am einfachsten zu implementieren.

Die Bearbeitungsmechanismen gelten sowohl für verlustfreie als auch für verlustbehaftete Anwendungen, siehe Abb. 2-4:

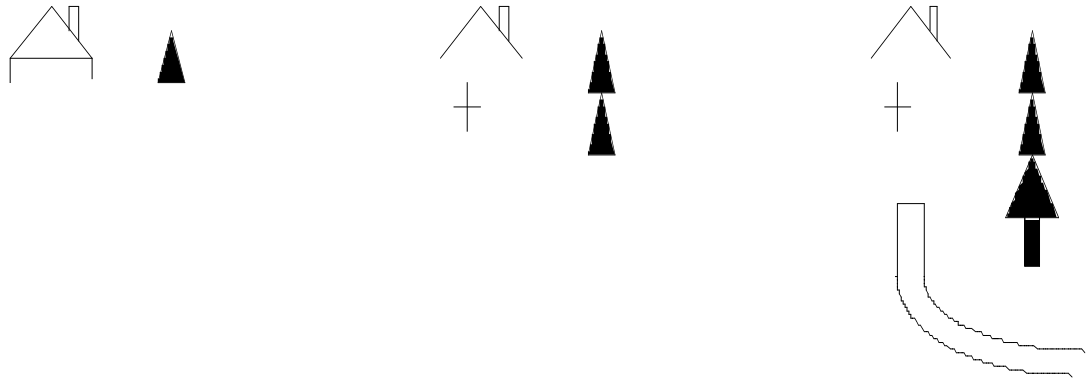


Abbildung 2-4: Dekodierung im sequentiellen Modus [ES 98]

2.1.2 Progressive Bildbearbeitung

Die zu bearbeitenden Daten werden mehrere Male durchlaufen, d.h. es wird pro Schritt nur ein Teil der Koeffizienten kodiert und zwar von niedrigen zu hohen Frequenzen. Es gibt zwei grundlegende Arten von progressiven Berarbeitungsmechanismen und zwar werden zum einen die unterschiedlichen Koeffizienten in Frequenzbändern zusammengefaßt (bei der Kodierung werden die niedrigen Frequenzen kodiert), zum anderen werden die hohen Frequenzen nacheinander kodiert. JPEG ermöglicht auch die Kombination der beiden Möglichkeiten um bessere Ergebnisse zu erzielen.

Bei einer Bildbetrachtung erscheint zuerst das Bild unscharf und wird im Laufe der Zeit immer schärfer. Dies bedeutet, daß bei der Bildbearbeitung immer die hohe Frequenzen kodiert werden, man spricht von einer niedrigen zu einer hohen Auflösung.

Progressive Bildbearbeitung ist sehr geeignet für Bildbearbeitung wo man sehr schnell einen Überblick über das übertragene Bild bekommt und je nach Qualitätswunsch den Prozess abbrechen kann. Progressive Bildbearbeitung hat sich im Internet durchgesetzt.

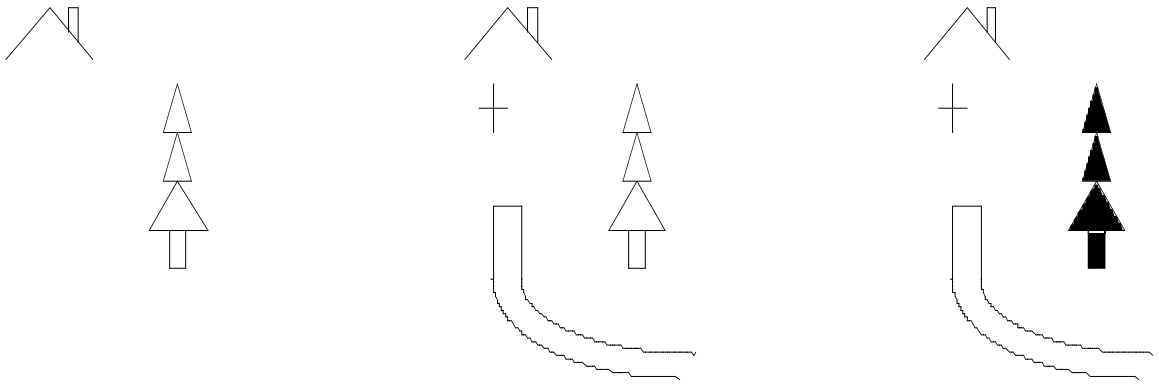


Abbildung 2-5: Dekodierung im progressiven Modus [ES 98]

2.1.3 Hierarchische Bildbearbeitung

In einem hierarchischen Verarbeitungsmechanismus wird eine Menge von Daten mit gröberer Auflösung verwendet, die durch „downsampling“ bewertetet werden, d.h. Filter von niedrigen zu hohen Frequenzen. Am Anfang werden die Daten mit den niedrigen Frequenzen kodiert. Diese Daten werden als Basis für die nächste Kodierung benutzt. Dies wird wiederholt, bis alle Frequenzen kodiert sind. Eine andere Interpretation ist die durch Auflösung, d.h. die Kodierung erfolgt von einer niedrigen Auflösung zu einer hohen Auflösung.

Die hierarchische Bildverarbeitung findet hauptsächlich in Anwendungen statt, wo zuerst der Benutzer eine grobe Darstellung der Daten benötigt, z.B. in den Bilddatenbanken, wo zuerst der Benutzer einen groben Überblick der Inhaltsverzeichnisse benötigt und danach bei tatsächlichem Bedarf das gewünschte Bild dennoch in einer wesentlich besseren Bildqualität zu bekommen.

2.2 Kodierungsschritte im JPEG-Kompressionensverfahren

Trotz der unterschiedlichen Bildverarbeitungsmechanismen stellt JPEG ein einheitliches Modell vor, das möglichst viele allgemein verwendeten Arten von Standbildern bearbeitet. Das Modell beschreibt die folgenden Kodierungsschritte:

I.) Bildaufbereitung

II.) Bildverarbeitung

III.) Quantisierung

IV.) Entropiekodierung

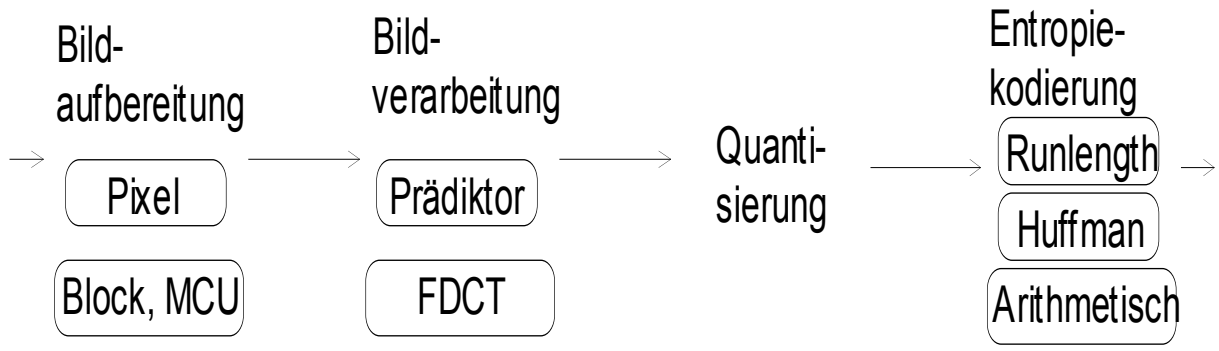


Abbildung 2-6: Schritte im JPEG Kompressionsverfahren [Stei 98]

2.2.1 Bildaufbereitung in JPEG

Um die im Kapitel 1 aufgeführten Anförderungen in JPEG zu garantieren, betrachtet man nicht explizit drei Bildebenen mit einer 9-Bit YUV Kodierung, sondern eine feste Anzahl von Zeilen und Spalten von Ebenen. Die Farbwerte werden nicht auf die tatsächlich dargestellten Farben mitkodiert.

JPEG ist ein Blockverfahren, d.h., das Bild wird in unterschiedliche N- Blöcke (Ebenen) verteilt. Diese Ebenen stellen die einzelnen Farben wie rot, gelb, blau (RGB) ,YIQ oder YUV Signale dar.

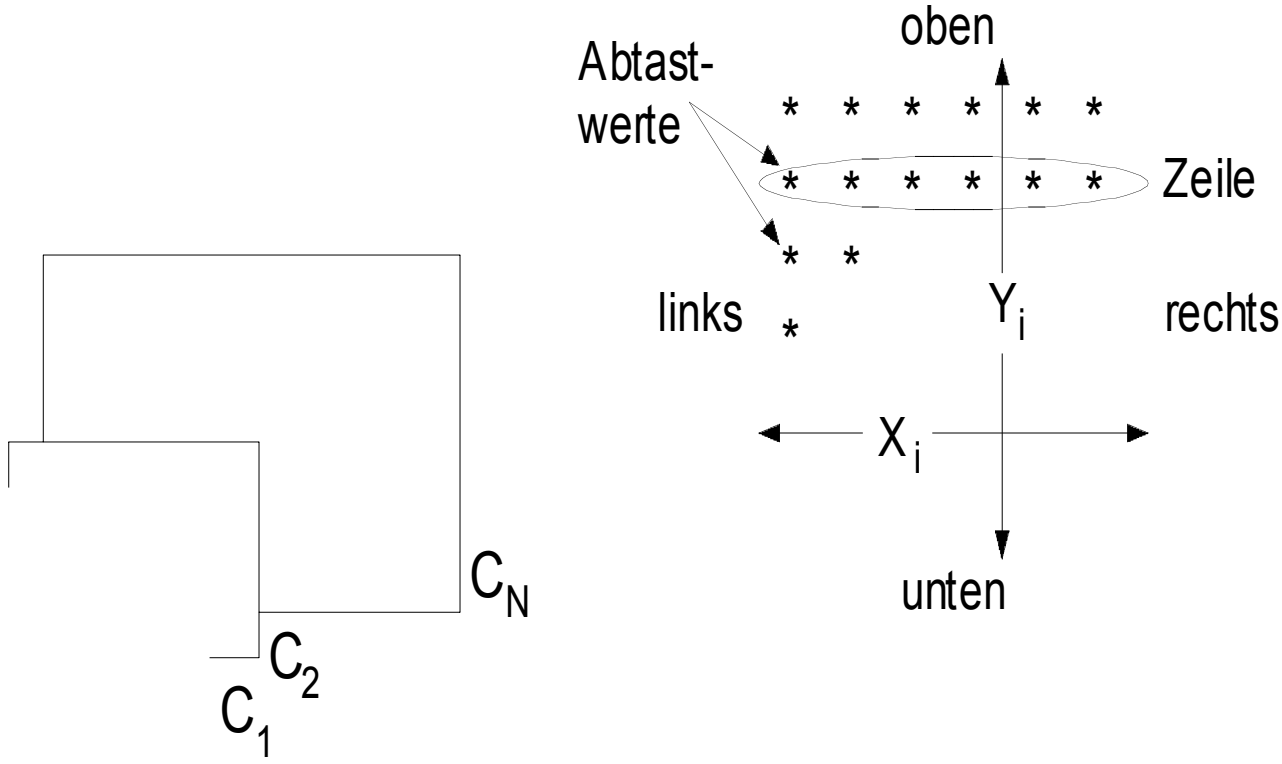


Abbildung 2-7: Digitales unkomprimiertes Einzelbild [Wall 92]

Eine Ebene besteht aus den unterschiedlichen Abtastwerten, den X- und Y-Koordinaten, d.h. Zeilen und Spalten in einem 2-dimensionalen Raum. Jede Ebene setzt sich aus einer rechtwinkligen Anordnung von X_i mal Y_i Pixel zusammen.

Bei verlustbehafteter Komprimierung, wo eine DCT-Transformation angewendet wird, werden die Blöcke jeweils in 8 mal 8 Pixel-Blöcke aufgeteilt.

Die Aufteilung der Pixelwerte geben die Auflösung der Ebenen an. Man unterscheidet Ebenen gleicher und ungleicher Auflösung.

2.2.1.1 Ebenen gleicher Auflösungen

Ebenen gleicher Auflösung bedeutet, daß jeweils in jeder Ebene die gleiche Anzahl von Pixeln vorhanden sind, d.h. in jeder Ebene die gleiche Anzahl von Zeilen und Spalten vorhanden sind. Zum Beispiel besteht ein RGB-Farbbild aus drei Ebenen mit jeweils der gleiche Auflösung (gleiche Anzahl von Zeilen $Y_R = Y_G = Y_B$ und derselben Anzahl von Spalten $X_R = X_G = X_B$)

Beispiele von jeweils 4 Zeilen und 5 Spalten

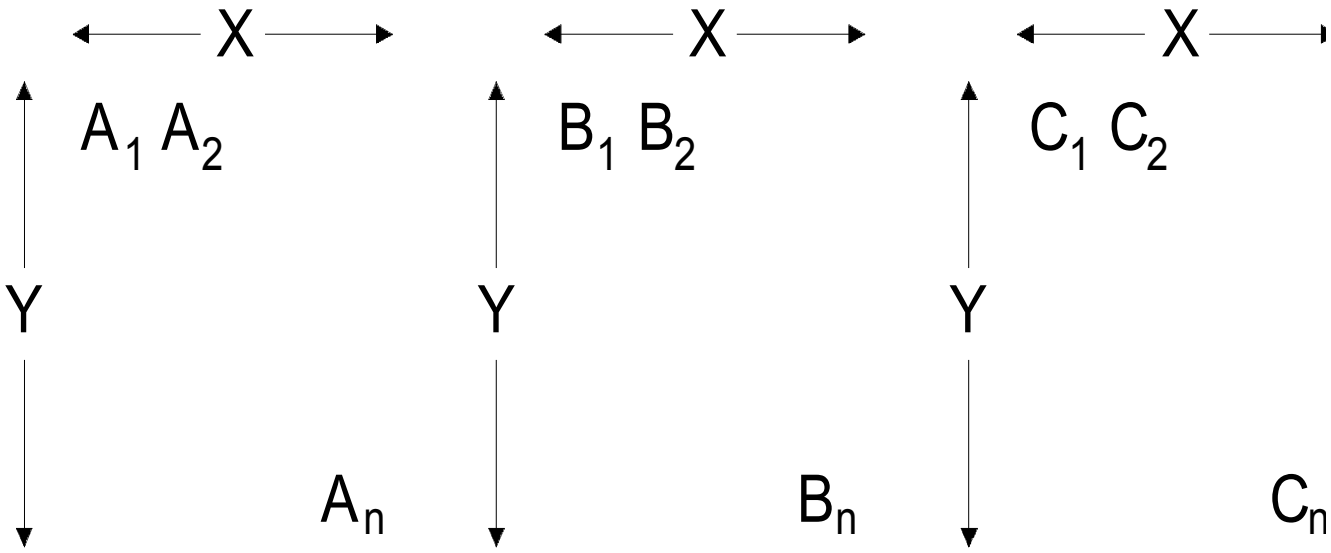


Abbildung 2-8: JPEG Bildaufarbeitung mit drei Ebenen und gleicher Auflösung (Stei 98)

2.2.1.2 Ebenen ungleicher Auflösungen

Eine Ebene mit ungleicher Auflösung bedeutet, daß jeweils in jeder Ebene eine ungleiche Anzahl von Pixeln vorhanden sind, das heißt, die Anzahlen der Spalten oder Zeilen in jeder Ebene unterschiedlich sind. Beispiel mit Ebenen ungleicher Auflösung:

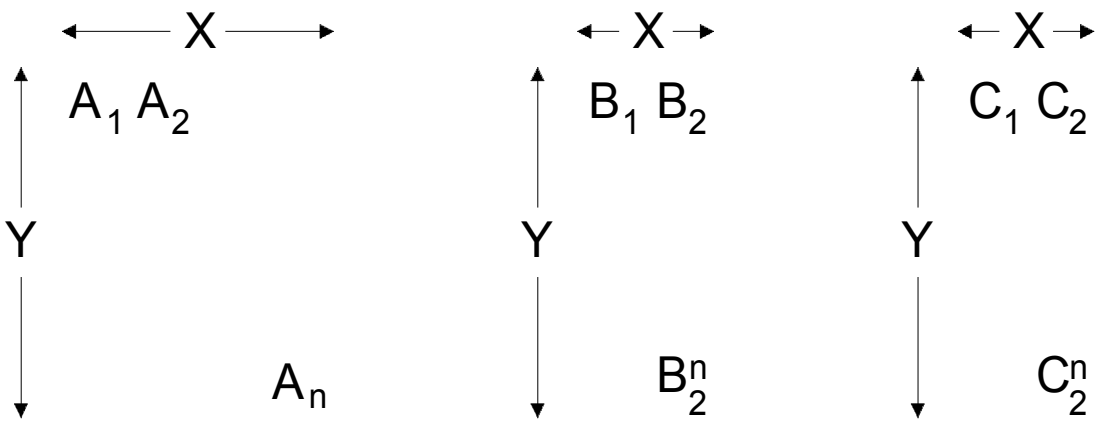


Abbildung 2-9: JPEG Bildaufarbeitung mit 3 Ebenen unterschiedlicher Auflösung (Stei 98)

Die erste Ebene (oben links) hat 5 Zeilen und 4 Spalten, die zweite Ebene hat 5 Zeilen und 2 Spalten und die dritte Ebene hat 5 Zeilen und 2 Spalten.

Bei der Komprimierung werden die Werte X und Y (Maximum alle X_i und Y_i), sowie H_i und V_i berechnet. H_i und V_i sind Ganzzahlen mit Werten zwischen 1 und 4. Sie stellen für jede Ebene die relative Auflösung in Bezug auf die minimale Auflösung dar.

In JPEG beträgt die Auflösung in verlustbehaftetem Modus entweder 8 oder 12 Bit pro Pixel und in verlustfreiem Modus zwischen minimal 2 und maximal 12 Bits pro Pixel.

Die Aufteilung der Pixel in unterschiedliche Ebenen bestimmen die Bildaufbereitungsmechanismen.

2.2.2 Bildaufbereitungsmechanismen.

Bei bei der Bildaufbereitung erfolgt eine Zerlegung des Bildes in Dateneinheiten. Unter verlustfreiem Modus versteht man unter einer Dateneinheit ganz genau 1 Pixel. Unter verlustbehaftetem (DCT) Modus versteht man eine Dateneinheit als einen Block von jeweils 8 mal 8 Pixel, dies liegt daran, daß die verwendete DCT-Transformation nur zusammenhängende Blöcke transformiert. Die Dateneinheiten werden Ebene für Ebene aufgearbeitet und zwar von links oben nach rechts unten.

Es gibt zwei Arten von Bildaufbereitungsmechanismen:

Die Aufteilung der Pixel in den unterschiedlichen Spalten einer Ebene bestimmt die Bildaufbereitung. Bestehen Ebenen aus Pixeln unterschiedlicher Spalten, so wird eine verschachtelte Bildaufbereitung benötigt, ansonsten die nichtverschachtelte Bildaufbereitung.

2.2.2.1 Nichtverschachtelte Bildaufbereitung (Non-Interleaved Data).

Die nichtverschachtelte Bildaufbereitung erfolgt, wenn die Pixelverteilung in einer Ebene nur in einer Zeile vorhanden ist. Die Dateneinheiten werden Pixel für Pixel von links oben nach rechts unten aufgearbeitet.

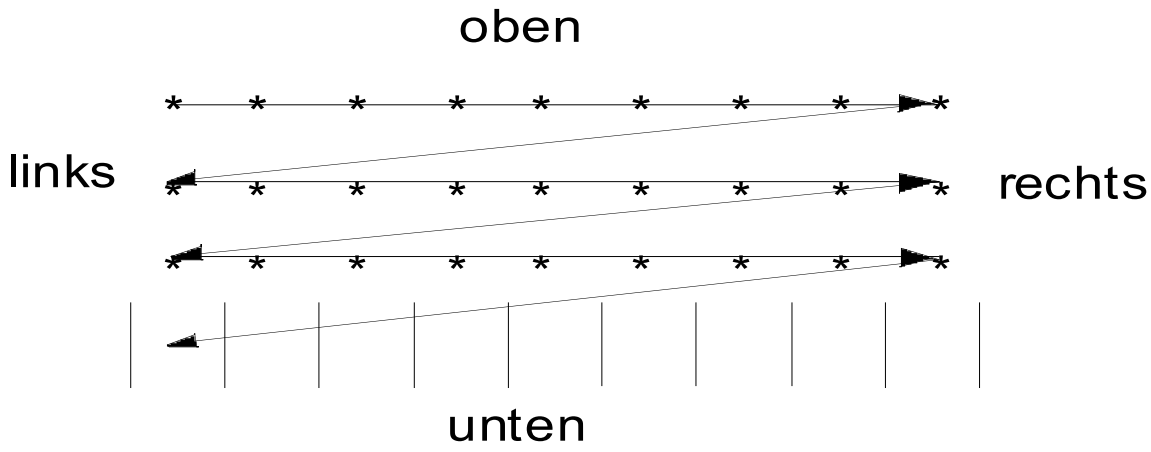


Abbildung 2-10: Nicht verschachtelte Reihenfolge der Bildverarbeitung [Wall 92]

2.2.2.2 Verschachtelte Bilder (Interleaved Data)

Verschachtelte Bildverarbeitung erfolgt, wenn ein oder mehrere Pixel in unterschiedlichen Zeilen unterschiedlicher Ebenen angeordnet sind. Hier faßt man die Dateneinheiten in verschiedenen Ebenen, im Gegensatz zu nichtverschachtelten zusammen, den sogenannten MCU´s (Minimum Coded Units).

Bei unterschiedlichen Auflösungen der verschiedenen Komponenten wird die gleiche Anzahl von Regionen mit unterschiedlicher Anzahl von Pixeln gebildet.

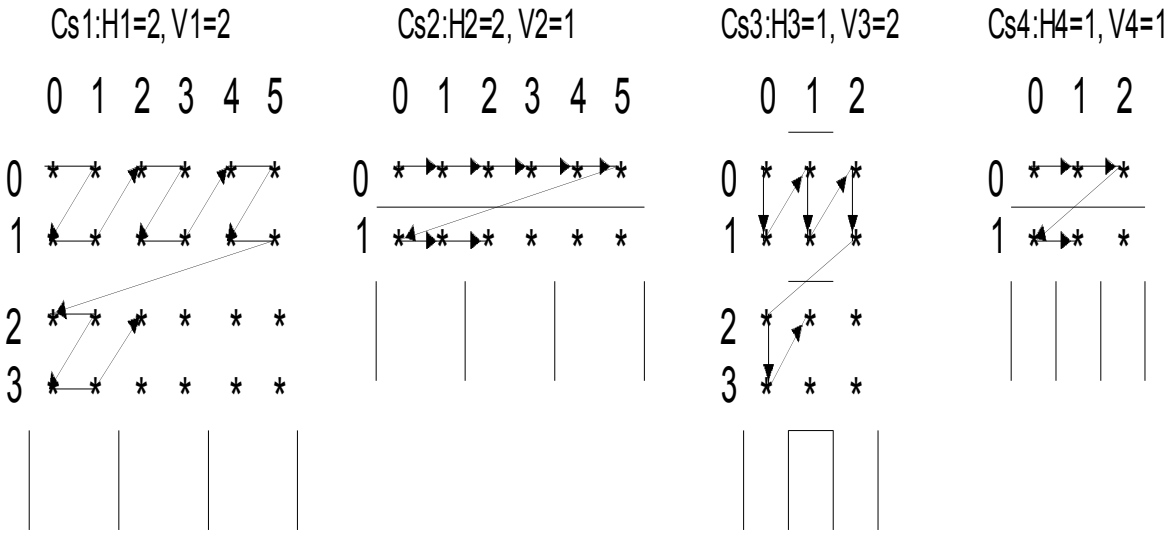


Abbildung 2-11: Verschachtelte Reihenfolge der Bearbeitung von Dateneinheiten (Wall 91)

Bei dem oben vorgestellten Beispiel hat die erste Ebene (oben links) die höchste Auflösung in beiden Richtungen (H1= 2 und V1 = 2) die vierte Ebene die niedrigste Auflösung in beiden Richtungen

(H4 = 1 und V4 = 1).

Eine MCU für dieses Beispiel setzt sich zusammen aus den unterschiedlichen Regionen der einzelnen Komponenten

MCU_1 =

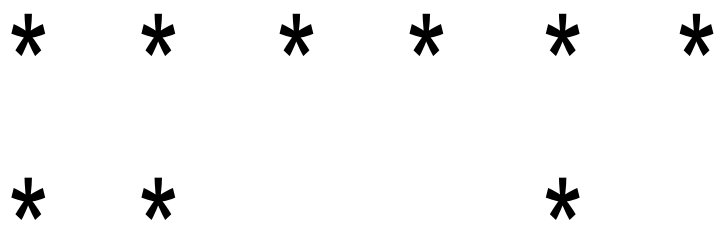


Abbildung 2-12: Eine MCU

Die anderen resultierenden MCU's werden wie folgt berechnet:

MCU_2 = a02 a03 a12 a13 - b02 b03 - c01 c11 - d01
MCU_3 = a04 a05 a14 a15 - b04 b05 - c02 c12 - d02
MCU_4 = a20 a21 a30 a31 - b10 b11 - c20 c30 - d10

In JPEG können nur maximal vier verschachtelte Komponenten kodiert werden, welches nicht erheblich ist, da bei Farbbildern nur drei Komponenten benötigt werden. Eine MCU darf maximal 10 Dateneinheiten beinhalten. Es können jedoch unterschiedliche Verschachtelungen kodiert werden.

2.2.3 Bildverarbeitung anhand von DCT-Transformation

In der Bildverarbeitung findet eine Transformationskodierung anhand von DCT statt. DCT gehört zu der Klasse der Fouriertransformation, die Fähigkeit besitzt, Signale in unterschiedliche Amplituden und Frequenzen umzuwandeln und darzustellen.

2.2.3.1 Kodierprozeß / Vorwärts -Transformation

Die unkomprimierten Daten werden jeweils in 8 mal 8 Pixel Blöcke aufgeteilt und werden durch die Reihenfolge der MCU's vorgegeben.

Abbildung 2-13: Darstellung der Frequenz-Transformation

In einer Frequenzdarstellung werden die Pixelwerte der verschiedenen Dateneinheiten mit S_{yx} bezeichnet. Die Werte X und Y liegen zwischen 0 und 7.

Jeder transformierter Pixelwert wird nach der folgenden Formel berechnet:

$$S_{vu} = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

$$\text{Es gilt: } c_u, c_v = \frac{1}{\sqrt{2}} \text{ für } u, v = 0; \text{ bzw. sonst } c_u, c_v = 1$$

Insgesamt werden 64 Transformationen pro Dateneinheit ausgeführt. Es werden 64 Koeffizienten S_{vu} berechnet. Die Koeffizienten werden als Frequenzen in einem zweidimensionalen Raum betrachtet. Ein Koeffizient repräsentiert einer senkrechten und einer waagerechten Richtung.

Bedeutung der Koeffizienten:

Die Koeffizienten repräsentieren eine Bilddarstellung. Einer hohe Frequenz bedeutet eine scharfe Kante oder eine hohe Auflösung. Einer niedrige Frequenz das Umgekehrte.

Zum Beispiel der Koeffizient S_{00} entspricht dem Anteil der Frequenz Null in beiden Achsen. Dies nennt man auch den DC- Koeffizient. Er dient zur Darstellung der Grundfarbtöne für die gesamte Dateneinheit. Die übrigen Koeffizienten werden AC-Koeffizienten genannt, z.B. S_{70} ist die höchste Frequenz, die in waagerechter Richtung (höchste Auflösung) auftritt. Desgleichen ist S_{07} die höchste Frequenz in senkrechter Richtung. Die Frequenz für die waagerechte Richtung ist für die senkrechten Streifen zuständig, und die senkrechte Richtung für die waagerechten Streifen. S_{77} entspricht dann der höchsten Frequenz für beide Richtungen. Beispiel koeffizienten-Darstellung für einem Schachbrett siehe Abbildung 2-14

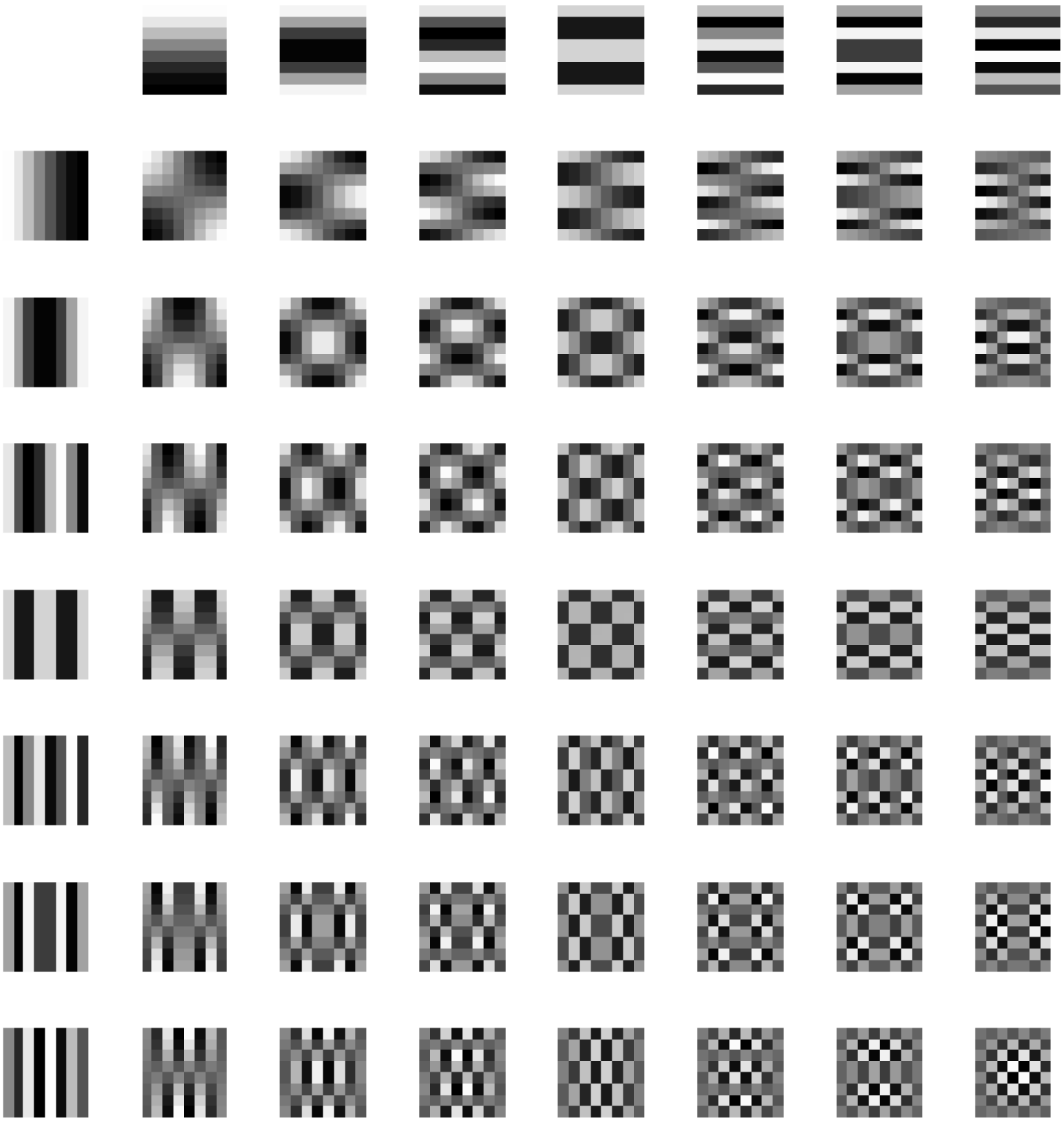


Abbildung 2-14: Basisblöcke für diskrete Kosinustransformation [Roqu 94]

2 2.3.2 Der Dekodierprozeß / Rückwärtstransformation

Der Dekodierungsprozeß ist der Umkehrprozeß von Kodierungsprozeß. Es werden anhand von IDCT-Transformation Frequenzsignale in Dateneinheiten umgewandelt(MCU's).

Man verwendet dafür folgende Gleichung:

$$S_{xy} = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{vu} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

Es gilt: $c_u, c_v = \frac{1}{\sqrt{2}}$ **für** $u, v = 0$; **bzw. sonst** $c_u, c_v = 1$

Anmerkungen zu FDCT und IDCT

Der FDCT und IDCT sind arithmetisch nur schwer genau berechenbar. Sie setzen einen hohen Rechenaufwand voraus. Es gibt 64 Additionsoperationen und 63 Multiplikationsoperation pro Block.

Der Bildinhalt vieler Bilder besteht nur zu einem geringen Teil aus scharfen Kanten. Es handelt sich meistens um eine grobe Fläche, was sich in DCT als niedrige Frequenz ausdrückt.

Bilder durchschnittlicher Komplexität besitzen viele AC-Koeffizienten die den Wert Null, oder fast Null haben.

2.2.4 Quantisierung

Der Quantisierungsprozess ist verlustbehaftet, d.h., er führt zu deutlichem Verlust (Verringerung der Datenmenge). von nicht wesentlichen Daten. In diesem Schritt wird eine Liste mit 64 Einträgen bereitgestellt. Jeder Eintrag wird zur Quantisierung einer der 64 DC-Koeffizienten verwendet. Damit ist garantiert, dafl jeder der 64 Koeffizienten separat abgearbeitet wird. So lassen sich bestimmte Frequenzen höher bewerten als andere. Dadurch kann man eine bestimmte Bildqualität beeinflussen.

2.2.4.1 Quantisierungsprozeß

Die Tabelleneinträge sind 8-Bit ganzzählige Werte. Dies bezeichnet man als Q_{vu}.

Die Quantisierungsformel lautet:

$$Sq_{vu} = \frac{S_{vu}}{Q_{vu}}$$

Je größer der Tabelleneintrag, desto größer ist die Quantisierung.

2.2.4.2 Dequantisierungsprozeß

Dies ist der Umkehrprozeß der Quantisierung und findet statt in der Dekodierungsphase.

Die Dequantisierungsformel (IDCT) ist:

$$R_{vu} = Sq_{vu} \times Q_{vu}$$

Man nutzt die gleichen Tabelleneinträge sowohl im Quantisierungs- als auch im Dequantisierungsprozeß.

1.1.5 Entropiekodeirung

Nach dem Quatisierungsprozefl werden die DC- und die AC-Koeffizienten in einen sequentiellen Bitstrom von 64 Integers eingeordnet. Zuerst der DC-Koeffizient der einen Wert gleich Null hat, er ist

die Differenz des ersten und zweiten Blockes. Der Rest sind die 63 AC-Koeffizienten die einen Wert Größe Null haben. Die Koeffizienten werden anhand einer Zick-Zack Kurve von einer niedrigen Frequenz zu einer höheren eingeordnet.

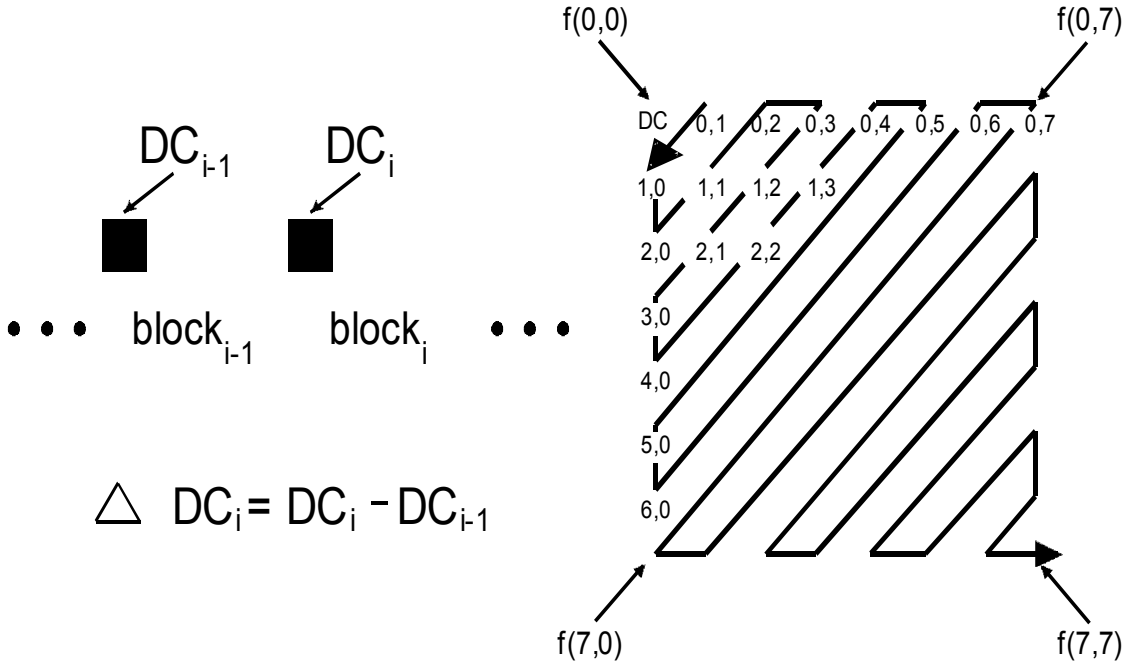


Abbildung 2-15: Zick-Zack Kurve [Wall 91]

Nachdem man eine Reihe von geordneten Integers (Koeffizienten) bekommen hat, erfolgt nun die Kompressionsphase. Hier werden unterschiedliche Algorithmen angewendet, wie Runlength, Huffman und arithmetische.

3 Literaturverzeichnis

[EfSt 98] Wolfgang Effelberg, Ralf Steinmetz
"Video Compression Technique"
1.Auflage 1998, Verlag für digitale Technologie, Hüthing GmbH

[Baum 92] Dietmar Baumstark
JPEG
Seminar:"Video Kompression"
Universität Karlsruhe 1991

[Roqu 94] C. von Roques
JPEG / MPEG Bilddatenkompression
Seminar: "Innovative Programmiermethoden für graphische Systeme"
Universität Karlsruhe, Institut für Betriebs- und Dialogsysteme, 1994

[Wall 91] Gregory K. Wallace
"The JPEG Still Picture compression Standard"
ACM Vol. 34, Number 4, p. 33-45,April 1991

MPEG-1 und MPEG-2

Karsten Frommolt

FB Informatik, Uni Dortmund
k.frommolt@cityweb.de

Zusammenfassung

MPEG-1 und MPEG-2 sind zwei Standards für digitales Video und Audio. Die Technik der beiden Verfahren beruht auf der Komprimierung einzelner Bilder im JPEG-Format, wobei ein digitalisiertes Bild über eine Farbraumtransformation und mittels der diskreten Kosinustransformation über Entropie- und Huffmankomprimierung seicherplatzoptimiert wird. Die aufeinanderfolgenden Bilder werden in intracodierte I-Frames und in intercodierte P- und B-Frames gewandelt und in der Videospur abgelegt. Der Hauptzweck von MPEG-1 war die Video-CD, CD interaktiv und die Spielekonsolen, wie z.B. AMIGA32. MPEG-2 ist für den profesionellen Einsatz in der Fernseh- und Satellitentechnik entwickelt worden. Insbesondere die HDTV Kompatibilität, die Interaktivität und die Skalierbarkeit sind die herausragendsten Eigenschaften von MPEG-2.

1 Motivation

1.1 Warum „Digitales Video“?

Ende der achtziger Jahre und Anfang der neunziger Jahre waren es gerade die Spielekonsolen und Entertainmentunternehmen, die besonderes Interesse darin hatten, bewegte Bilder in Form von digitalen Videos dem Privatmann näher zu bringen. Die technischen Voraussetzungen waren denkbar schlecht, denn sowohl die Rechnergeschwindigkeit beim Verarbeiten, als auch die Übertragungsgeschwindigkeit der damalig „neuen“ CD-ROM Laufwerke waren sehr gering. Mit einem Single-Speed-Laufwerk sollte man Filme ansehen können, was sich schnell als eine enorme Aufgabe für die Entwickler dieser Zeit herausstellen sollte.

1.2 Problemstellung

Bei einem PAL-Fernsehbild mit einer Auflösung von 768 x 576 Bildpunkten in einer Farbtiefe von 24 Bit pro Pixel ergeben sich schon bei kurzer Betrachtung die Probleme, vor denen die Informatiker und Mathematiker standen:

1 Bild benötigt also 768 x 576 x 3 x 8 Bit ≈ 1,327 MByte

1 Sekunde Video mit 25 Bildern pro Sekunde benötigt daher 1,327 x 25 = 33,18 MByte

Für 1 Stunde Video ergibt sich also eine Datenmenge von ca. 119,4 GByte

Das würde bedeuten, auf eine CD-Rom passen gerade einmal 183 Sekunden Video.

1.3 Idee

Wie reduziert man nun diesen Datenstrom? Die Idee besteht zum einen darin, möglichst viele hintereinanderliegende kleine Werte oder Nullen durch Umformung zu bekommen und zum anderen die Bilder so zu bearbeiten, daß man hinreichend wenig komplette Bilder abspeichert und für die Bewegungsdarstellung nur die Änderungen der Einzelbilder benutzt.

2 MPEG-1

Im Jahre 1988 enstand die Motion Picture Expert Group. Diese Gruppe entwickelte einen Standard für digitales Video und Audio. Dieser Standard definiert eine Systemspur, eine Videospur und eine Audiospur innerhalb des Datenstroms. Der Standard ist auf die damaligen technischen Voraussetzungen angepaßt und sollte mit einem Datenstrom von max. 1,86 MBit / sec auskommen. Das entspricht in etwa einer Datenübertragungsrate von 150 KByte / sec, einem Single-Speed-CD-ROM.

Den größten Anteil innerhalb des Datenstroms hat die Videospur mit ca. 1,1 MBit / sec, die Audiospur erhält 128 KBit/sec und die Systemspur, deren Aufgabe die Synchronisation ist, bekommt 300 KBit/sec. Wenn man eine Videosequenz mit einem Framegrabber abspeichert, so liegt das Video im M-JPEG-Format vor. Dieses ist eine Aneinanderreihung von im JPEG Format komprimierten Bildern.

Auch MPEG-1 benutzt den JPEG-Standard um Teile der Bilder zu komprimieren. Wie bereits eingangs erwähnt lag das Einsatzgebiet von MPEG-1 besonders bei den Spielkonsolen, dem von Phillips entwickelten CD-I und natürlich im PC-Bereich bei den CD-Rom. Der Standard definiert weiterhin eine maximale Auflösung der einzelnen Bilder von 720 x 576 Bildpunkten, was einem PAL-Fernsehbild entspricht. Allerdings ist die normale Wiedergabegröße eines MPEG-1 Videos durch den Standard durch 352 x 288 Pixeln bei 50 Hz und 352 x 240 Pixeln bei 60 Hz definiert. Wie auch bei MPEG-2, so ist die Codiergeschwindigkeit auf Grund der rechenintensiven Schritte sehr gering. Die Decodiergeschwindigkeit hingegen, ist so gut, daß sich der Einsatz dieses Verfahren lohnte. Man betrachtet bei einem MPEG-Datenstrom folgende verschiedene Schichten:

- Video Sequenz
- Group of pictures
- Picture
- Slice
- Macroblock
- Block

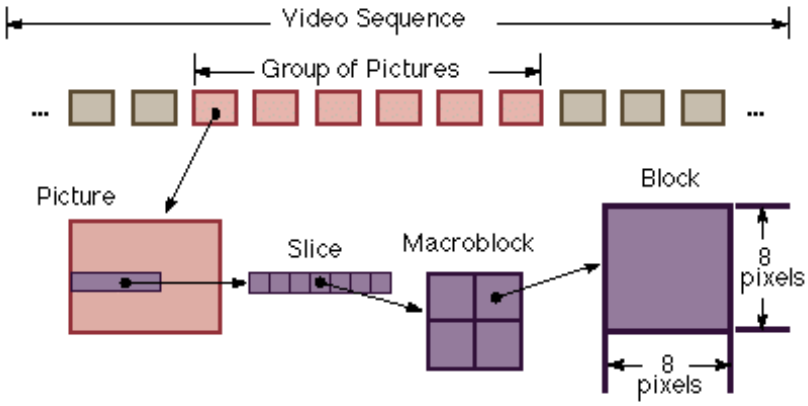


Abbildung 2-1: Schichten von MPEG-1

Die Sequenz Layer beinhaltet zu Beginn u.a. folgende 2 Einträge: Die für die Sequenz konstante Bitrate und den zur Decodierung minimal notwendigen Speicherplatz, wobei der Standard eine maximale Buffergröße von 376832 Bit vorgibt. Die Group of Pictures muß mindestens ein sogenanntes I-Frame enthalten, die Picture Layer beinhaltet ein Bild und die Slices bestehen aus einer Anzahl von Macroblöcken, die im folgenden Kapitel erläutert werden. Die folgende Abbildung zeigt einen Ablaufplan zur Generierung eines MPEG-1 Datenstroms.

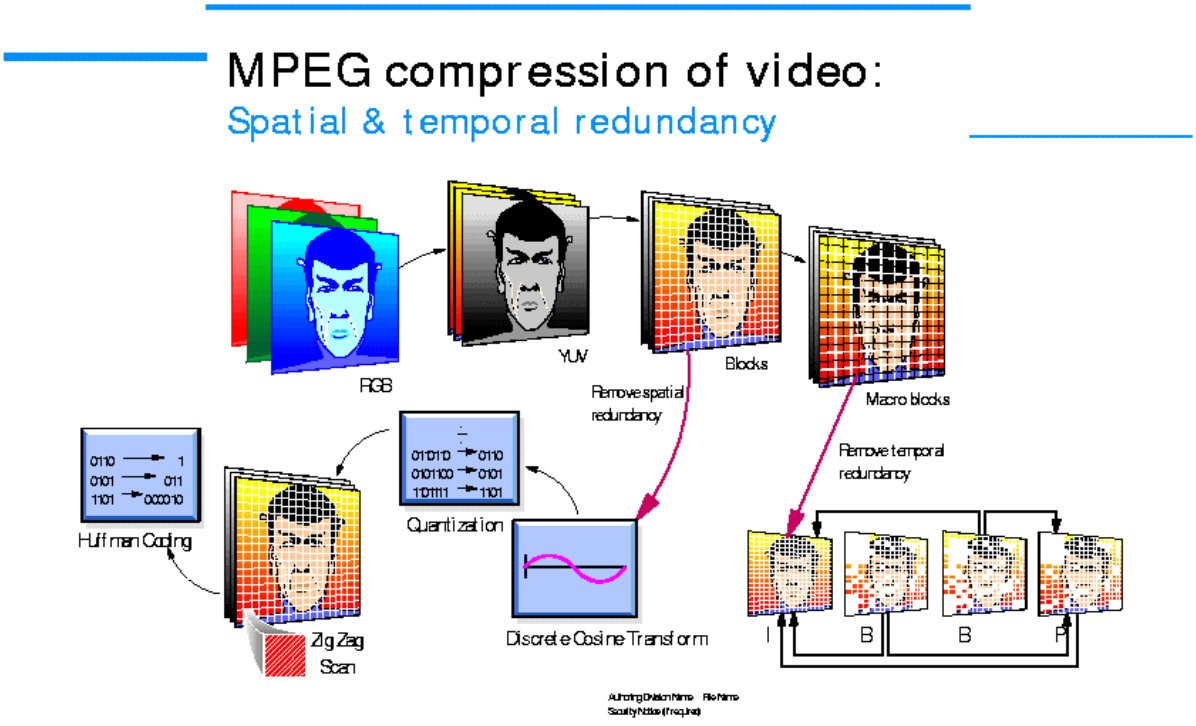


Abbildung 2-1: Ablauf einer MPEG-Codierung

3 Bildaufbereitung

3.1 Digitalisiertes Bild

Schon beim ersten Schritt des Ablaufplans, muß man einige Qualitätsverluste in Kauf nehmen, denn es ist zum Beispiel unmöglich ein Analogsignal, etwa das einer Kamera oder eines Videorekorders ohne Verluste zu digitalisieren. Um ein Bild am Rechner zu digitalisieren, benutzt man einen sogenannten Framegrabber.

Diese Zusatzkarte hat einen Videoeingang und ist in der Lage 25 – 30 Bilder pro Sekunde zu digitalisieren und auf einem Medium (meist der Festplatte) abzuspeichern. Die Preise für solche Karten liegen bereits in einem, für den Privatmenschen erschwinglichen Rahmen. Hat man eine digitale Kamera, so ist der Qualitätsverlust gleich Null, wobei natürlich die Abbildung der Realität nie perfekt sein kann.

3.2 Farbraumtransformation

Ein Bild des Framegrabbers wird meist im üblichen RGB-Format (Rot, Grün, Blau) abgelegt, das bedeutet, daß jeder Pixel des Bildes einen Farbwert als Kombination des Rotanteils, des Grünanteils und des Blauanteils erhält. Besteht ein Video aus RGB-Format Bildern, so nennt man es 24 Bit Video.

Aus der Nachrichtentechnik ist schon lange ein anderes Bildformat bekannt: Das YUV-Format

Beim YUV-Format erhält jeder Pixel einen Helligkeitswert und zwei Farbwerte. Dabei ergibt sich eine Umrechnungsformel vom RGB- ins YUV-Format:

$$Y = 0{,}299R + 0{,}587G + 0{,}114B$$

$$U = -0{,}146R - 0{,}288G + 0{,}434B = 0{,}496(B - Y)$$

$$V = 0{,}617R - 0{,}17G - 0{,}1B = 0{,}877(R - Y)$$

Der Helligkeitswert Y wird der Luminanzanteil (Y) und die Farbwerte(Cb, Cr) Chrominanzanteil genannt. Zur weiteren Verarbeitung wird der Wertebereich [0,255] aller Abtastwerte in das symetrische Intervall um die Null verschoben [-127,128].

3.3 Subsampling

Es stellt sich nun die Frage, warum man nun das RGB-Format in das YUV-Format umwandeln sollte, denn in der unveränderten Form des YUV-Formats, dem sogenannten 4:4:4 gibt es keinen ersichtlichen Grund dieser Transformation. Dennoch gibt es einen entscheidenden Vorteil. Auf Grund der Physiologie des Auges, ist bei uns Menschen die Unterscheidung von Helligkeitswerten sehr viel hochauflösender, als die Unterscheidung von Farben. Daher kann man beim einem YUV-Bild sogenanntes Subsampling anwenden. Subsampling bedeutet, daß zwei oder mehr nebeneinander liegende Bildpunkte den gleichen Farbwert erhalten. Da das Auge nicht so hochauflösend reagiert, fällt dieser Qualitätsverlust kaum auf.

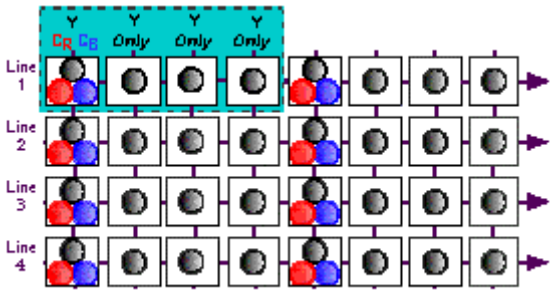


Abbildung 3-1: 4:1:1-Subsampling

Beim sogenannten 4:1:1-Subsampling erhalten vier nebeneinander liegende Bildpunkte den gleichen Farbwert. Es gibt noch andere Subsamplingverfahren, wie zum Beispiel 4:2:0 und diverse co-sited Ansätze, die hier nur der Vollständigkeit halber erwähnt werden sollen.

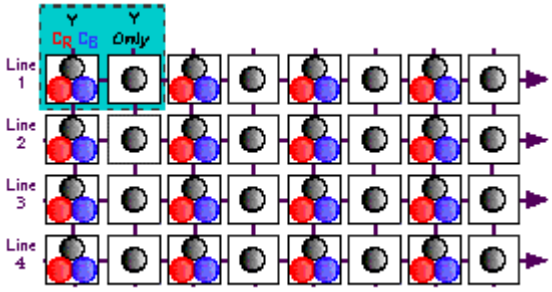


Abbildung 3-2: 4:2:2-Subsampling

Das sogenannte 4:2:2-Subsampling wird beim MPEG-2 Standart eingesetzt. Es soll aber im Kontext erwähnt werden. Wie bereits erwähnt, lehnt sich MPEG bei der Einzelbildcodierung an den JPEG-Standard. Das Bild wird in sogenannte Macroblöcke zu je 16 x 16 Pixeln aufgeteilt. Die Pixel innerhalb eines solchen Macroblocks werden durch 4 Blöcke mit Luminanz und je 2 Blöcke Chrominanz aufgebaut (Siehe nächste Abbildung).

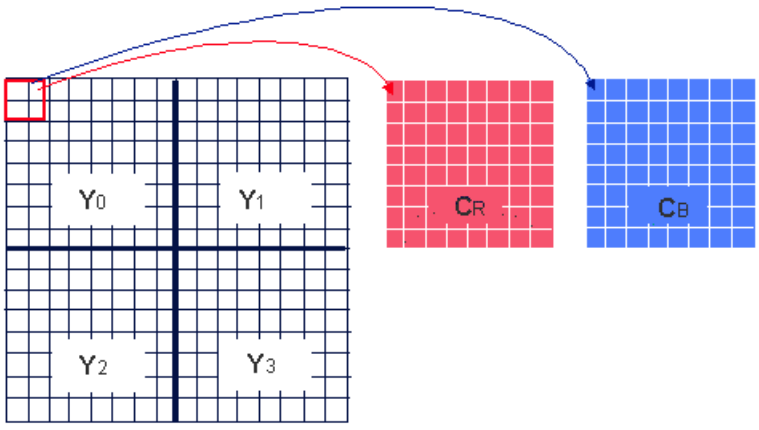


Abbildung 3-3: Codierung der Pixel durch Macroblöcke

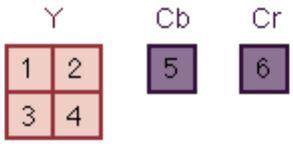


Abbildung 3-4: YUV-Komponenten in den Makroblöcke

3.4 Diskrete Cosinus Transformation

Mittels diskreter Tranformation ist es möglich, eine begrenzte Anzahl von Abtastwerten eines Signals in eine diskrete Frequenzbereichs-Repräsentation umzuwandeln. Die Anzahl der Abtastwerte bleibt durch die Transformation unverändert, d.h. man erhält ebenso viele spektrale Stützstellen, wie der ursprüngliche Signalabschnitt Abtastwerte enthielt.

Es handelt sich also um eine umkehrbare, verlustfreie (vernachlässigbare Ungenauigkeiten durch Gleitkommazahlarithmetik des Rechners) Transformation eines Bildes in den Freuquenzbereich des Bildes. Die DCT wird auf jedem Block angewendet. Dadurch erhält man innerhalb eines 8 x 8 Pixel großen Blocks eine Aufteilung der Frequenzen. Der Koeffizient in der linken oberen Ecke eines jeden Blocks nennt man DC-Koeffizient. Er spiegelt die niedrigste Frequenz wieder. Werte in den Zeilen stehen für Horizontalfrequenzen, die in den Spalten, für Vertikalfrequenzen. Somit hat der Pixel an den Koordinaten (8,8) innerhalb der Matrix die höchsten Frequenzen.

Durchschnittliche Bilder haben viele Flächen und wenig Kanten (also mehr nieder- als hochfrequente Anteile), was bedeutet, das viele AC-Koeffizienten (alle anderen außer dem DC-Koeffizienten) fast Null, oder gleich Null werden.

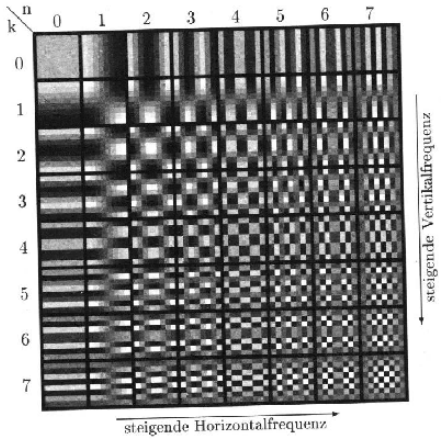


Abbildung 3-5: Auswirkungen der DCT auf Frequenzen

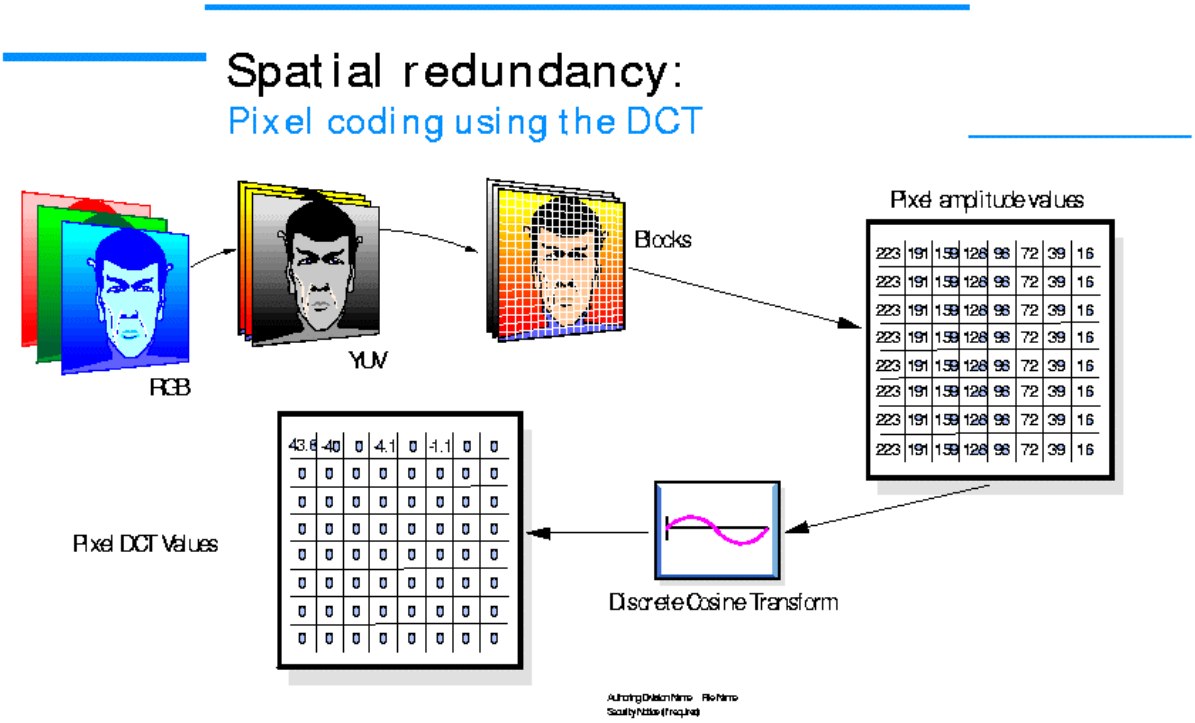


Abbildung 3-6: Auswirkungen der DCT

3.5 Quantisierung

Jeder der enstandenen Koeffizienten (S) der Makroblöcke, wird mit einem entsprechendem Eintrag in einer Quantisierungstabelle (Q) quantisiert, wobei gilt.

$$Sq = round\,\frac{S}{Q} \quad \text{mit S = Koeffizient und Q =Eintrag in der Quantisierungstabelle}$$

Die Werte in der Quantisierungstabelle nehmen mit höher werdenden Koordinaten zu, damit die hoch-frequenten Anteile des Blocks zu Null werden. Das ist natürlich mit einem Qualitätsverlust behaftet.

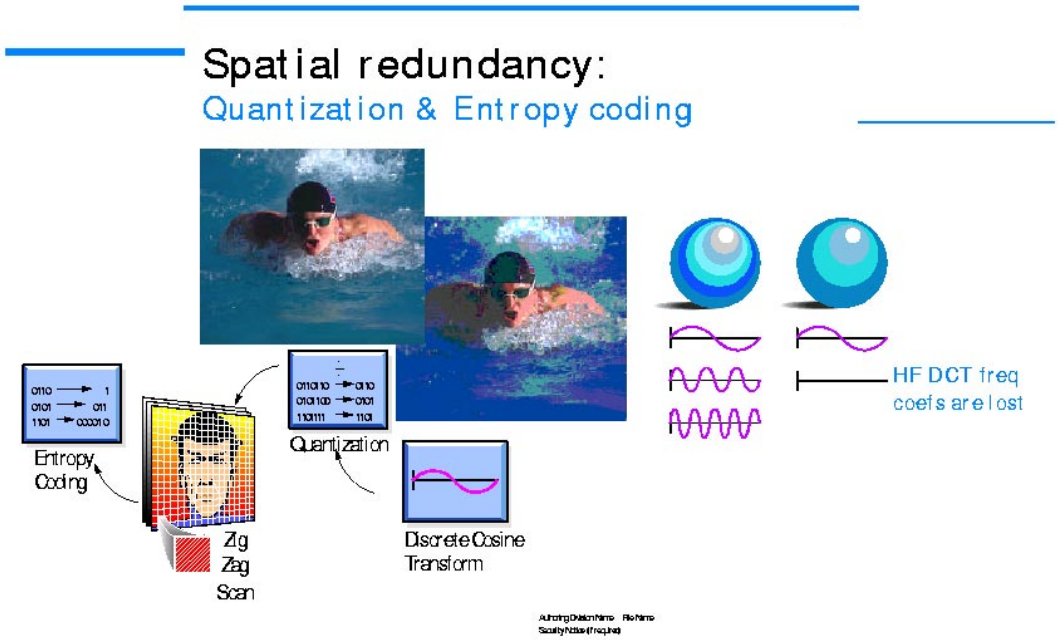


Abbildung: Auswirkungen der Quantisierung auf das Bild

3.6 Entropie und Huffman

Hat man nun eine solche Matrix quantisiert, bedient man sich gängiger Kompressionsverfahren, wie der Entropiekomprimierung und der Huffmankomprimierung. Dabei durchläuft man die Matrix in einem sogenannten Zick-Zack-Scan, das in der nächsten Abbildung dargestellt ist.

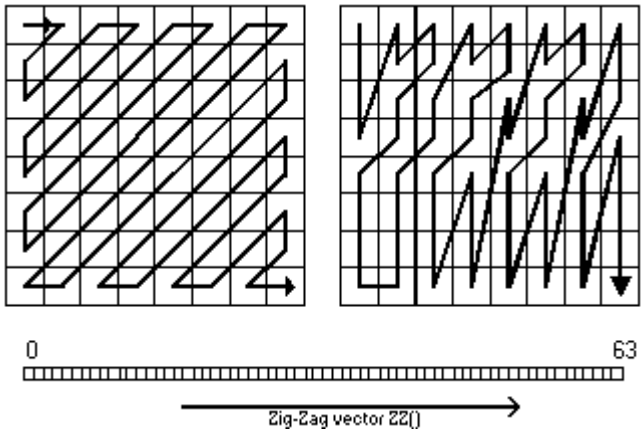


Abbildung 3-4: Zick Zack Scan Verfahren

Links MPEG-1, rechts Sonderfall bei MPEG-2

Die rechte Seite der Abbildung zeigt einen Durchlauf, wie er bei bestimmten Voraussetzungen beim MPEG-2 Standard benutzt wird.

Bei der Entropiekomprimierung (oder auch Lauflängenkompromierung) werden aufeinanderfolgende gleichartige Bits nur durch Ihre Anzahl repräsentiert. Bei der Huffmankomprimierung werden häufige Bytefolgen werden durch bestimmte Bitsequenzen ersetzt, die in einer Tabelle gespeichert werden.

4 Frames

4.1 I-Frames

Wodurch ist nun die enorme Datenreduzierung zu begründen? Wir haben zwar durch die oben beschriebenen Verfahren extrem viel komprimiert, doch sind die einzelnen Bild momentan eigentlich nur im JPEG-Format gespeichert und man hätte nichts anderes als ein Motion JPEG Video. MPEG definiert verschiedene Frames, wobei die intracodierten Frames (I-Frames) genau der JPEG Standbildkompression entsprechen. Diese Frames dienen als Ankerpunkte, für die nun folgenden Frametypen.

MPEG gibt keine Anweisungen, wie oft ein I-Frame pro Sekunde im Strom enthalten sein muß, aber praktisch hat sich eine Wiederholrate von 15 Frames als gut erwiesen, also ungefähr zwei I-Frames pro Sekunde. Gegenüber den anderen Frametypen, haben die I-Frames eine schlechte Komprimierungsrate.

4.2 P-Frames

Wie bei der Intraframekodierung wird auch hier das Bild in Makroblöcke aufgeteilt. Nun wird für jeden Makroblock der möglichst ähnliche Makroblock im vorangegangenen I-Frame oder P-Frame gesucht. Dafür wird eine einfache Differenzbildung aller Absolutwerte der Luminanzkomponenten vorgenommen. Der minimale Betrag der Summe aller Differenzen zeichnet den ähnlichsten Macroblock aus. MPEG schreibt keinen Algorithmus zur Bewegungsschätzung vor, sondern spezifiziert nur die Codierung des Ergebnisses. Zu codieren ist dann nur noch der Bewegungsvektor (motion vector) und die geringe Differenz (prediction error) zwischen den Makroblöcken. Durch diese Angaben läßt sich ein neuer Makroblock rekonstruieren.

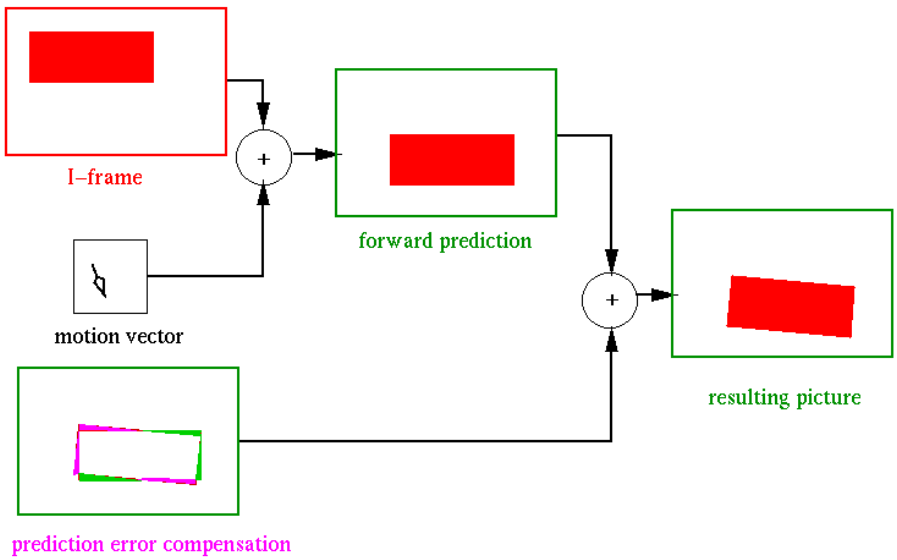


Abbildung 4-1: Prädiktive Codierung

In P-Bildern wird also nur die Differenz zweier Makroblöcke codiert und gespeichert.

Um einen motion vector zu finden wird ein Makroblock in einem Suchbereich von 48 x 48 Pixeln verschoben. Die nächste Abbildung verdeutlicht den Vorgang.

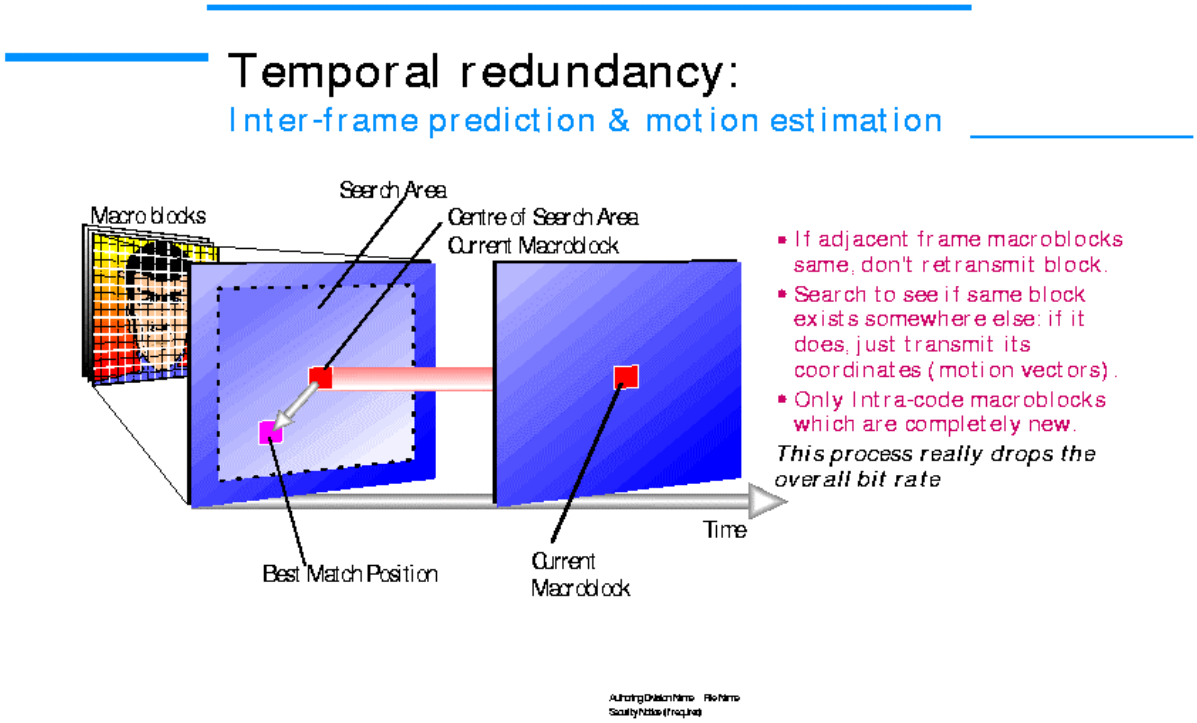


Abbildung 4-2: Suchbereich des ähnlichsten Makroblocks

4.3 B-Frames

In einer Bildsequenz bewegt sich ein Ball von rechts nach links vor einem statischen Hintergrund. Im rechten Bereich des Bildes erscheinen nach und nach Teile, die in den vorhergehenden Einzelbildern noch von diesem Ball abgedeckt waren. Die Prädiktion für diese Bildbereiche werden dann günstigerweise nicht aus einem vorhergehenden, sondern aus einem noch folgenden Einzelbild abgeleitet. Das bedeutet, daß B-Bilder einen Mittelwert aus den im aktuellen Makroblock enthaltenen Informationen und der eines vorausgegangenem und eines nachfolgenden Makroblocks berechnen. Dieses Verfahren nennt man „motion compensated interpolation“. Sie beziehen sich auf P-Frames und I-Frames.

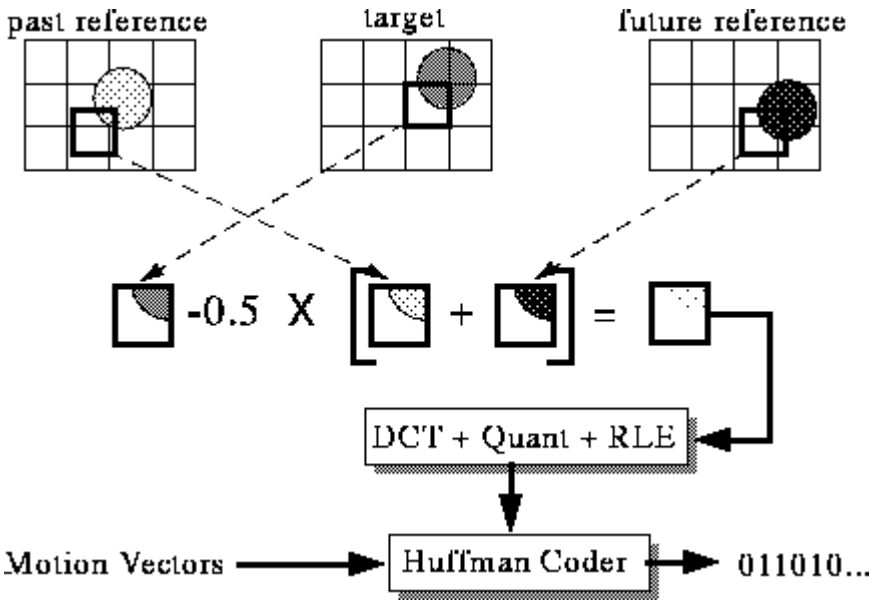


Abbildung 4-3: Bidirektionale Codierung

4.4 DC-Frames

Die sogenannten DC-Frames sollen hier nur der Vollständigkeit halber erwähnt werden, da sie nur beim schnellen Vorlauf innerhalb des Videos eine Rolle spielen.

Bei diesen Frames wird immer nur der DC-Anteil jedes Makroblocks codiert.

4.5 Reihenfolge der Frames

Bedingt durch die Referenz auf nachfolgende Bilder, müssen die B-Frames nach den zu ihnen gehörigen Referenzbildern übertragen werden.

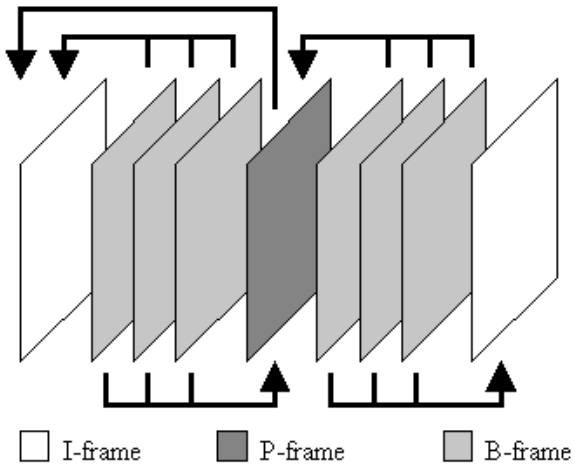


Abbildung 4-4: Schachtelung der einzelnen Frames

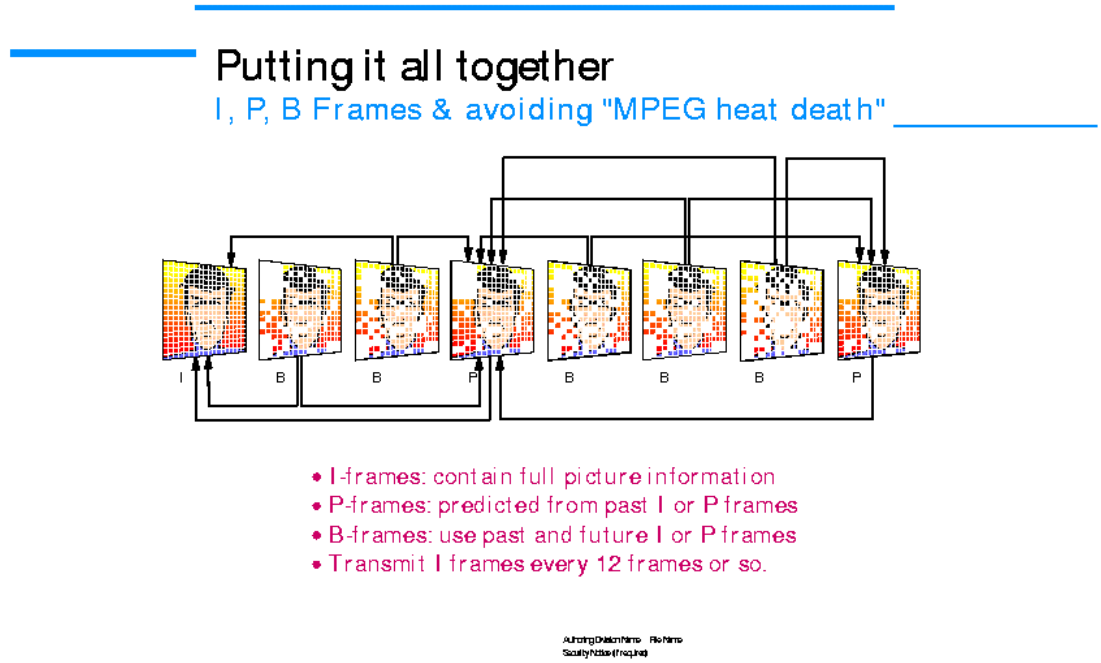


Abbildung 4-5: Die Interaktion der verschiedenen Frametypen

4.6 Der Videostrom bei MPEG-1

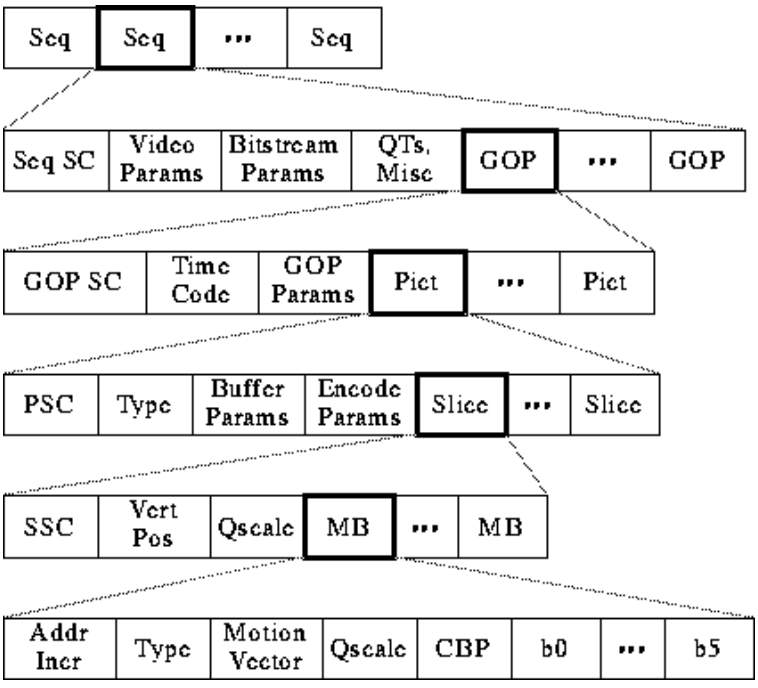


Abbildung 4-6: Schichten des Videostroms

Die Aufteilung des Videostroms erfolgt in der Reihenfolge der einzelnen Schichten, somit wird die Sequenz in die Gruppe der Bilder gesplittet und darunter stehen dann das Bild, die Slices und die Makroblöcke.

5 MPEG-2

Die Motion Picture Expert Group setzte sich noch einmal in den neunziger Jahren zusammen um einen neuen Standard zu definieren. Dieser ist notwendig geworden, da sich das Einsatzgebiet von digitalem Video, insbesondere aber auch die technischem Voraussetzungen entscheidend geändert haben.

Die Hauptanwendungen, für die MPEG-2 entwickelt worden sind, liegen besonders in der weltweiten Fernsehübertragung in digitaler Qualität. Der Einsatz von Satelliten und Glasfaserkabeln ermöglichen heutzutage eine viel höhere Datenübertragungsrate, als noch zur Zeit von MPEG-1. MPEG-2 legt eine Datenübertragungsrate von bis zu 100 MBit / sec fest. Obgleich MPEG-2 eine Weiterentwicklung von MPEG-1 ist, so ist es abwärtskompatibel. Die entscheidenden Verbesserungen gegenüber MPEG-1 sind vor allen Dingen eine bessere Wiedergabequalität, viele Trickeffekte wie Überblendungen, Fading, Zeitlupe, Vorspulen und Zurückspulen. Desweiteren ist für MPEG-2 eine weitaus bessere Audiospur definiert, die Stereo- und 2-Kanal Daten enthalten kann. MPEG-2 ist interaktiv, d.h. der Nutzer kann zu jedem Zeitpunkt Veränderungen an der Qualität, oder der Darstellungsgröße des Videos vollziehen. Um zukunftstauglich zu sein, ist MPEG-2 HDTV (High Definition Television) fähig und bietet die Möglichkeit des Interlaced Video.

5.1 Neue Codierungsmethoden

5.1.1 Profile und Level

Die neuen Profile und Level schränken die zur Verfügung stehenden Parameter der Codierung ein und standardisieren gleichzeitig die Kodierungsparameter. Die folgende Tabelle veranschaulicht die Level von MPEG-2:

Level	Max. Frame sample Größe	Pixel / sec	Maximale Bitrate
Low	352 x 288	3,05 M	4 MBit / s
Main	720 x 480	10,40 M	15 MBit / s
High1440	1440 x 1152	47,00 M	60 MBit / s
High	1920 x 1080	62,70 M	80 MBit / s

Abbildung 5-1: Level bei MPEG-2

Die Profile bestimmen die verschiedenen neuen Layer von MPEG-2. Es gibt über den normalen Schichten noch weitere wie zum Beispiel die Enhancement Layer:

	Simple profile No B frames Not scalable	Main profile B frames Not scalable	SNR scalable profile B frames SNR scalable	Spatially scalable profile B frames SNR scalable	High profile B frames Spatial or SNR scalable
High level **1920 x 1152 x 60**		≤ 80 MBit / s			≤ 100 MBit / s
High 1440-level **1440 x 1152 x 60**		≤ 60 MBit / s		≤ 60 MBit / s	≤ 80 MBit / s
Main level 720 x 576 x 30	≤ 15 MBit / s	≤ 15 MBit / s	≤ 15 MBit / s		≤ 20 MBit / s
Low Level 352 x 288 x 30		≤ 4 MBit / s	≤ 4 MBit / s		

Abbildung 5-2: Profile von MPEG-2

5.1.2 Skalierbarkeit

Zitat von Leonardo Chiariglione, Chairman der MPEG-2 Gruppe:

„Scalability ist die Möglichkeit des Decoders, Teile eines Datenstroms zu ignorieren und doch sinnvolle und angepaßte Video- und Audiodaten zu erzeugen."

Die Skalierbarkeit erstreckt sich beim MPEG-2 Standard auf eine temporale Skalierbarkeit, eine sogenannte Signal-to-Noise-Ratio scalability (SNR scalability), der spatialen Skalierbarkeit und einer Datenpartionierung. Die temporale Skalierbarkeit, bietet dem Anwender dir Möglichkeit, sein Video in einer unterschiedlichen Anzahl an Frames pro Sekunde zu decodieren, dabei enthält die Base Layer die Sequenz in einer geringen Frame Codierung und die sogenannte Enhancement Layer die restlichen Frames, um die Sequenz in ihrer besten Qualität darstellen zu können.

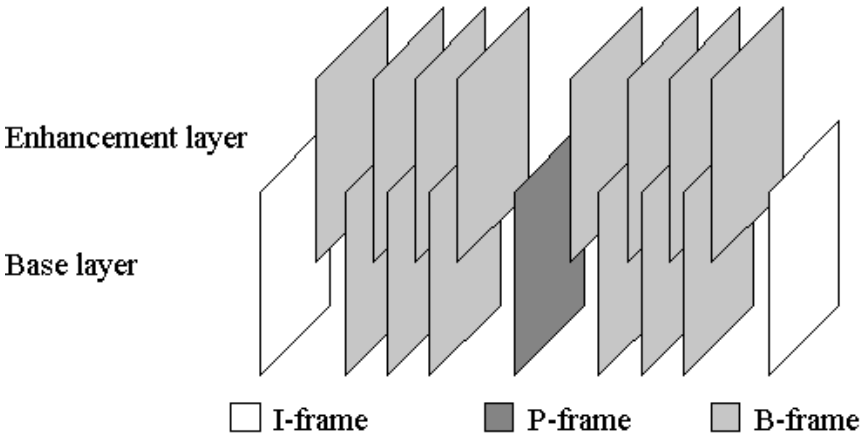


Abbildung 5-3: Temporale Skalierbarkeit

Die Signal-to-Noise-Ratio Skalierbarkeit ermöglicht die zu decodierende Sequenz in einer normalen, niedrigen Qualität und in einer sehr guten Qualität abzuspielen. Dabei werden B-Frames in P-Frames getauscht. Auch hierbei sind die Zusatzinformationen innerhalb der Enhancement Layer abgelegt.

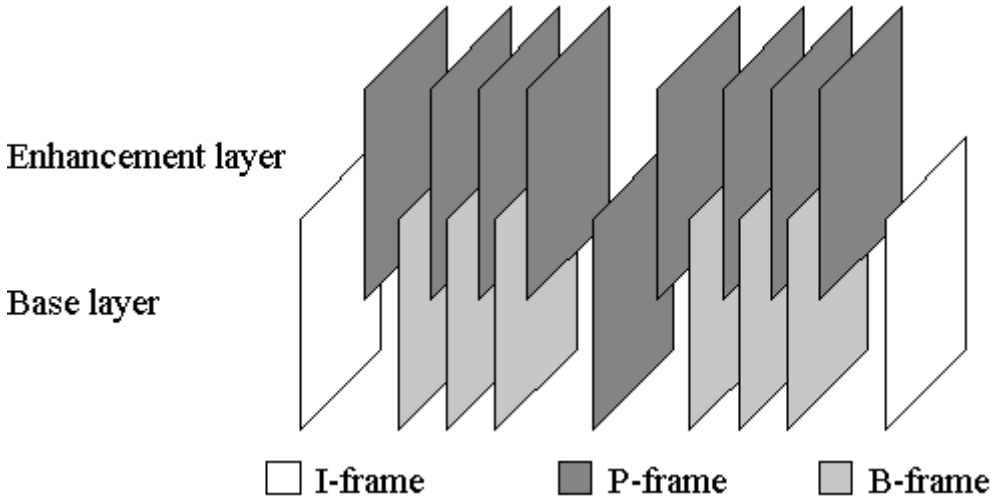


Abbildung 5-4: SNR Skalierbarkeit

Das Data Partitioning definiert innerhalb des Datenstroms High- und Low-priority-Streams. Dies hat zur Folge, daß zum Beispiel bei einem Videofile über die Rede Kennedys auch bei einer schlechten Datenübertragungsrate die Audiospur auf Grund ihrer höheren Priorität in einer sehr guten Qualität decodiert wird und die Videosequenz an sich in einer geringeren Qualität.

Als dritte Scalability, die in diesem Zusammenhang erwähnt werden sollte, ist die sogenannte Pan-Scan-Scalability, die es dem Anwender ermöglicht, Ausschnitte im aktuellen Bild zu definieren.

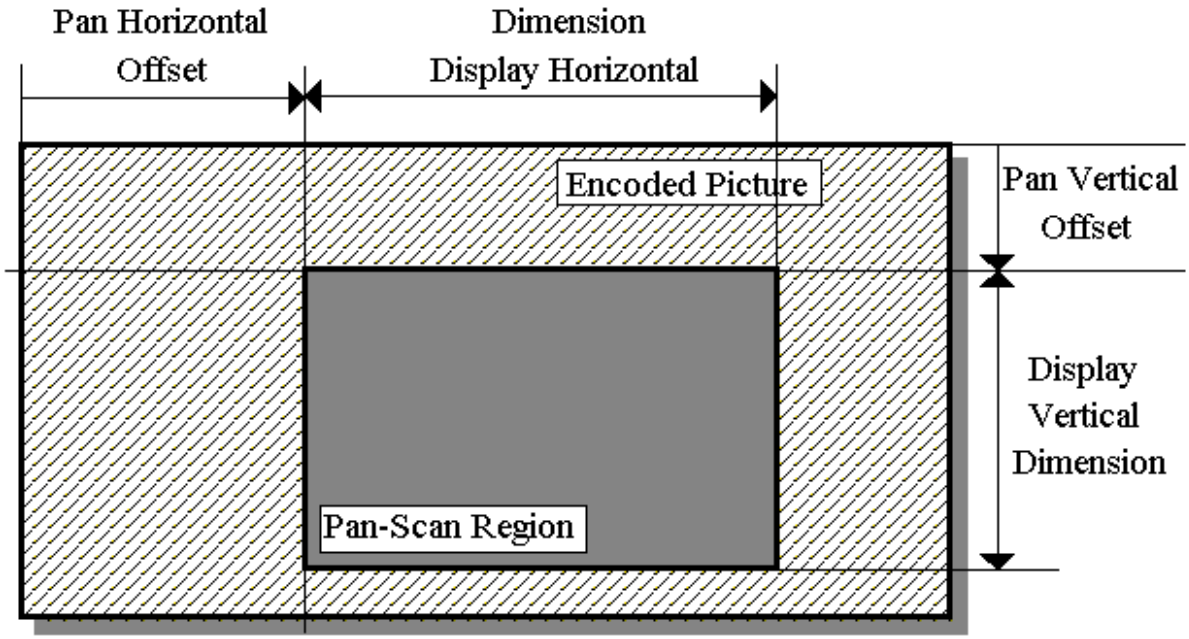


Abbildung 5-5: Pan-Scan-Scalability

5.1.3 Security

MPEG-2 ist weiterhin der erste der Standards und Formatspezifikationen im Bereich Multimedia und Video, in dem auch der Bereich der Security (Vertraulichkeit und Integrität) mit konzipiert worden ist.

MPEG-2 ist robust gegenüber Bitfehlern und Zellenverlusten. Ein spezieller für MPEG-2 optimierter, schneller Chiffrier-Algorithmus ist spezifiziert, der eine Verschlüsselung und Überprüfung des kompletten Datenstroms ermöglicht.

Diese Verschlüsselung kann insbesondere für die Satellitenübertragung (z.B. von einem Studio ins andere) und für Pay-TV genutzt werden. Durch Verwendung von Checksummen, kann zu jedem Zeitpunkt der Übertragung ein Fehler im Datenstrom erkannt werden, der ansonsten eventuell den Abspielvorgang unterbrechen oder blockieren könnte. Auch ist damit die Veränderung der Daten durch einen Nutzer erkennbar.

Der dafür zuständige Teilbereich ist der sogenannte Transport Stream, der aus Paketen von 288 Bytes Länge besteht und in dem die Fehlerkorrektur abgelegt wird.

Mit MPEG-2 ist es möglich innerhalb eines Datenstroms bis zu dreißig verschiedene Videokanäle und bis zu 5 + 1 Audiokanäle zu verwalten. (Der letztgenannte ist für einen Subwoofer, der beim Dolby Surround eingesetzt wird). Auch ein 2-Kanalton ist integriert, so daß mit MPEG-2 digitales Fernsehen in noch nie dagewesener Qualität bezüglich Video, Audio und Interaktivität realisiert werden kann.

6 Streams

6.1 PTS und DTS

Die in einnem Datenstrom enthaltenen B-Frames müssen bezüglich der referenzierenden I- und P-Frames später vorkommen, aber zwischen den beiden abgespielt werden. Darum enthält der Datenstrom sogenannte Stamps, die als Indikatoren für das zeitliche Auftreten im Stream und die Darstellung als Frame dienen. Man unterscheidet im MPEG-2 Datenstrom sogennate Decode Time Stamps (DTS) und Presentation Time Stamps (PTS) beide sind Bestandteil des im nächsten Abschnitt erklären Packet Elementary Streams

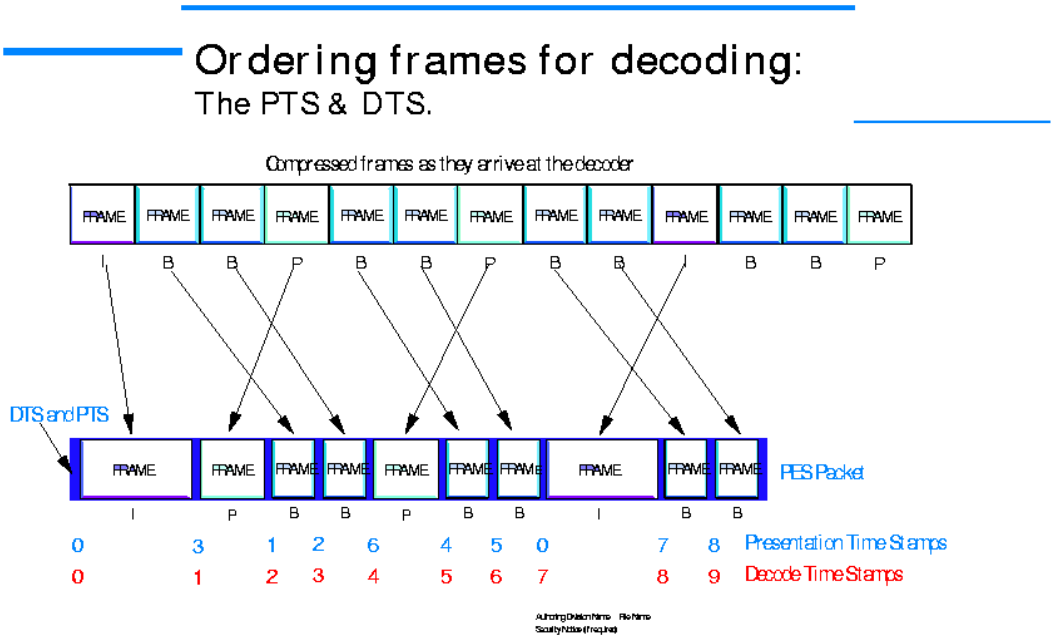


Abbildung 6-1: DTS und PTS

6.2 Packet Elementary Stream

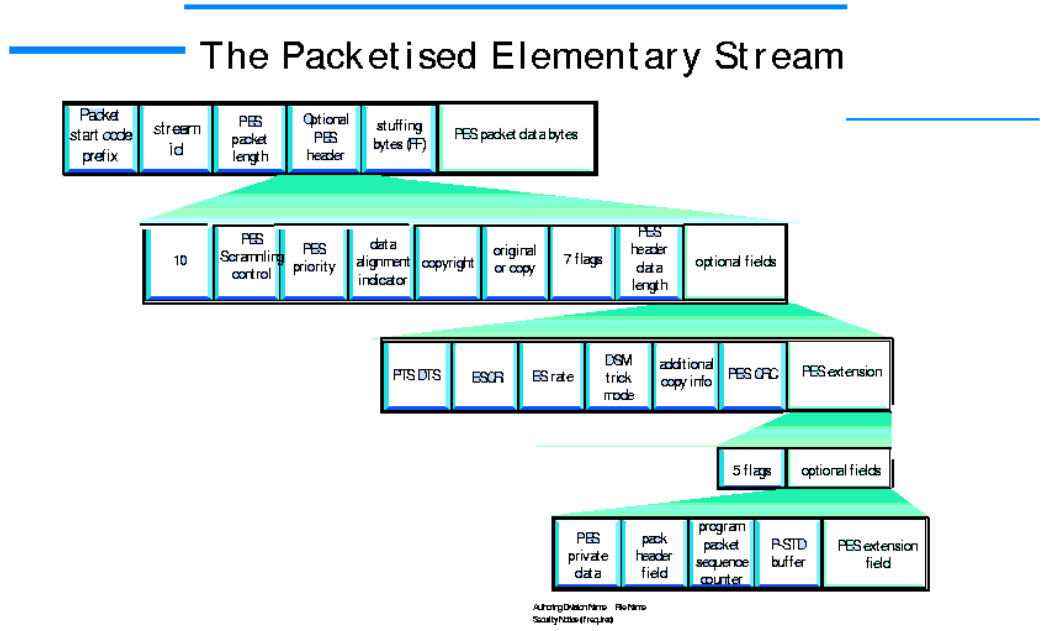


Abbildung 6-2: Packet Elementary Stream

Der Header des PES enthält Informationen über den Inhalt der PES Daten Bytes. Die Packetlänge der PES ist variabel, meist 64 Kbytes, aber sie können auch länger sein. Falls Informationen, die im Header liegen, fehlerhaft sind, geht das gesamte Paket verloren.

6.3 MPEG-2 Transport Stream

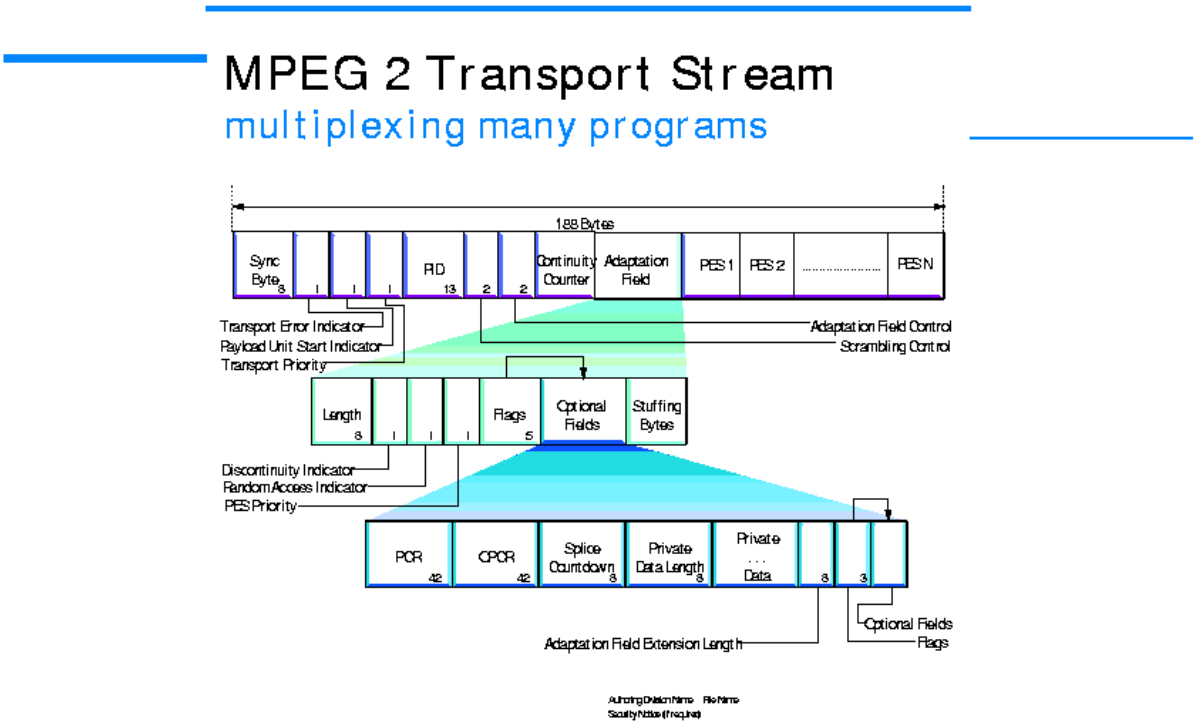


Abbildung 6-3: Der MPEG-2 Transport Stream

Der Transport Stream packt die Pakete des PES mit variabler Länge in Pakete fester Länge, was für die meisten Datenübertragungssysteme notwendig ist. Der Transport Stream erlaubt das Multiplexing von bis zu 30 Kanälen.

Das Sync Byte setzt den Beginn eines TS Pakets und dient zur Synchronisation. Der Transport error indicator gibt mittels eines block error testing an, ob ein Paket fehlerhaft ist. Der PID (Packet Identifier) beinhaltet alle notwendigen Informationen zum Finden, Identifizieren und Rekonstruieren eines jeden Kanals.

7 Literatur

[Stei 98] Steinmetz, Effelsberg:
„Video Compression Techniques“
dpunkt.verlag 1998

[Chi1 96] Leonardo Chiariglione:
http://drogo.cselt.stet.it/mpeg/mpeg2[Stand:Juli 96]

[Chi2 96] Leonardo Chiariglione:
http://drogo.cselt.stet.it_mpeg_Standarts_mpeg1/mpeg-1.htm[Stand Juni96]

[Anso 97] Ansorg:
http://www.fh-jena.de/contrib/fb/et/personal/ftp/kodierung

[Gühr ??] Gabriele Gühring:
http://michelangelo.mathematik.uni-tuebingen.de_people_gagu_ref/ref.htm

[Grub 97] Jürgen Gruber:
MPEG-Seminar[11.06.1997]

[Chi3 96] Leonardo Chiariglione:
MPEG and multimedia communications
http://www.isce96.it

[Gowa 99] John McGowan:
http://www.rahul.net/jfm/dct.htm[Stand:05.1999]

[Unkn 99] Unbekannt:
http://rnvs.informatik.tu-chemnitz.de.mpeg_tech/mpeg_tech.htm
[Stand 05.1999]

[Gade 95] Gadegast, Schmidt:
Ein packendes Format
www.mpeg1.de.mpuo12/mpuo12.htm

[Gade 96] Gadegast, Schmidt:
Ein breites Spektrum
www.mpeg1.de.mpuo5/mpuo05.htm

Weitere interessante Informationen zu MPEG im Internet:

http:// www.iso.ch/welcome.html

http://www.mpeg1.de

http://www.-plateau.cs.berklex.edu/mpegfaq/MPEG-s-FAQ.html

http://www.creative.net/~tristan/MPEG.html

ftp://ftp.tnt.uni-hannover.de/pub/MPEG/Audio/mpeg2/

ftp//ftp.mpeg.cablelabs.com/pub/bitstreams/working/video

MPEG-4 und MPEG-7

Marc Iwan

FB Informatik, Uni Dortmund
Marc.Iwan@ruhr-uni-bochum.de

Zusammenfassung

Fortschritte in den Bereichen Computertechnik, Telekommunikation und Unterhaltungselektronik haben die Grundlagen für interaktive Multimediaanwendungen geschaffen. Der separate Zugriff und die Übertragung ausgewählter, semantische zusammenhängender Bildelemente setzen geeignete Strukturen und Organisation der Bilddaten vorraus. MPEG-4 stellt den Standard, der Lösungen für solche Probleme bietet und erforderliche Funktionalitäten zusammenstellt. Es treten außerdem immer mehr audiovisuelle Daten in digitaler Form auf. Diese Daten sind meist irgendwie über Netze zugreifbar, aber voher will potentiell relevantes Material gefunden werden. Es existiert bisher keine Lösung, die die Suche nach audiovisuellen Daten ermöglicht, da noch keine Möglichkeit existiert solche Daten zu beschreiben. MPEG-7 soll ein Standard werden, der diese Fragen nach der Suche von AV-Material klären will. Dieser Seminarbeitrag gibt einen Überblick über MPEG-4 und MPEG-7.

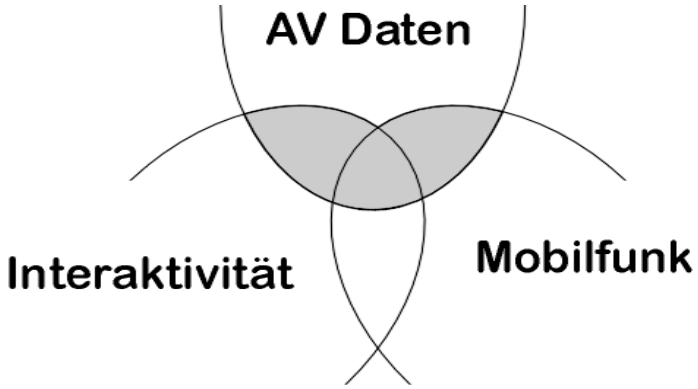
1 MPEG-4

1.1 Überblick und Motivation

Seit einigen Jahren gibt es Veränderungen in den Bereichen Computer, Telekommunikation, TV/Kino. Diese Bereiche wachsen mehr und mehr zusammen. Im Computer können Audio- und Videodaten verarbeitet werden. Computer kommunizieren über heterogene Netze, die aus Fest- und Mobilfunknetz bestehen. Im Bereich der Telekommunikation existieren Videodaten und Interaktivität und auch im Bereich TV/Kino, z.B. *Video on Demand*, verlangt es nach Interaktivität. Es besteht Bedarf nach der Möglichkeit Bilder und Inhalte zu manipulieren oder auszuwählen. Um dieses Ziel zu erreichen, müßen geeignete Strukturen und Operationen für die Bilddaten bereitgestellt werden, um Bildinhalte zu verändern. MPEG-4 ist der Standard, der diese Mittel zur Verfügung stellt. Abbildung 1-1 zeigt die drei sich verschmelzenden Bereiche „Computer“, „Telekommunikation“ und „TV/Kino“. Die schraffierte Fläche stellt die Anwendungsgebiete für MPEG-4 dar.

Es wird der Informationsaustausch, die Manipulation und Speicherung von Datenmaterial immer wichtiger werden („Informationsgesellschaft“). Das Material an sich ändert sich, es wird nicht nur mit Kamera und Mikrophon 'aufgenommen' und in den Computer überspielt. Viel mehr AV-Material wird synthetisch im Computer erzeugt. Dies findet in breiten Teilen Anwendung, so z.B. in der Architektur, wenn 3D Ansichten eines zu bauenden Objekts zur Präsentation benötigt werden. Die Übertragung von Daten ist vielfältiger. Informationen werden über Fest- und Mobilfunknetze zum Benutzer

transportiert. Dabei ist zu beachten, daß dort jeweils andere Übertragungsbedingungen auftauchen. Der Konsum ändert sich ebenso. Informationen werden nicht mehr nur textbasiert, sondern audiovisuell dargestellt. Das Material wird interaktiv gehandhabt, wie dies z.B. im WorldWideWeb geschieht. Die Wiederverwendbarkeit von AV-Material steigt erheblich durch die Digitalisierung. So entstehen Multimediaanwendungen, und der Bedarf nach Manipulation des Datenmaterials wächst. Man möchte auf separate Inhalte von AV-Material auf intuitive Weise zugreifen können und z.B. ein Video nicht einfach nur vor- und zurückspulen können. Die Linearität, die man aus den bisherigen Standards MPEG-1 und MPEG-2 oder auch z.B. vom Videorecorder her kennt, entfällt nun völlig. Der Zugriff auf die Daten soll inhaltsbasiert erfolgen. Gleichzeitig soll ein hoher Kompressionsgrad der Daten erreicht werden. Der MPEG-4 Standard gewährleistet dies alles unter Beachtung von Flexibilität und Erweiterbarkeit. Die Anforderungen an MPEG-4 sollen nun im folgenden dargestellt werden.



Einsatzgebiet von MPEG-4

Abbildung 1-1: Einsatzgebiet von MPEG-4

1.2 Funktionalitäten in MPEG-4

Das Besondere und Neue an MPEG-4 ist, daß es die Bildinformationen anders als bisher handhabt. Es wird nicht mehr das gesamte (rechteckige) Bild als Basiselement betrachtet wie es noch bei MPEG-2 der Fall ist (I-, P-, B-Frames), sondern ein einzelnes Objekt, welches inhaltliche Information verkörpert. Diese Objekte müssen natürlich identifiziert, manipuliert, gespeichert und übertragen werden. Es ist nicht Ziel von MPEG-4 einen Standard zu beschreiben, der für die Analyse des Materials zuständig ist. Dabei ist natürlich klar, daß die Analyse des Bildinhaltes sehr viel Aufwand erfordern kann. Bei einer automatischen Segmentierung kann der Computer wohl sehr leicht z.B. farbige Flächen identifizieren, aber was ist, wenn es sich um viel komplexere Dinge handelt? In

MPEG-4 geht es vielmehr darum, Multimediadaten zu komprimieren, deren Szene eine Zusammenstellung aus Objekten ist. Bei MPEG-4 ist also das semantische Objekt der Mittelpunkt um den es geht. Jetzt wird vielleicht auch ein bißchen klarer, daß der Rechner alleine, also ohne interaktive, menschliche Hilfe kaum alle möglichen Bildinhalte erkennen und differenzieren kann.

Derzeit existiert kein anderer Standard, der diese Art der Betrachtung unterstützt. Mit so einer Struktur ist ein höherer Grad an Interaktivität möglich, als es bisher der Fall war. MPEG-4 möchte dem Benutzer Zugriff auf einzelne sogenannte audiovisuelle Objekte (AVOs) verschaffen. Dort setzen die Funktionalitäten von MPEG-4 an.

1.2.1 Inhaltsbezogene Interaktivität

In diesem Bereich geht es um Funktionalitäten, die sich auf inhaltliche Interaktivität beziehen.

Manipulation und Editieren von Datenströmen

MPEG-4 soll geeignete Syntax und Kodierungstechniken bereitstellen, damit man Inhalte manipulieren und editieren kann ohne die Objekte erst dekodieren und dann wieder kodieren zu müssen. Zum Beispiel können im Heimfilmstudio Inhalte zu Szenen hinzugefügt, verändert oder entfernt werden. Ebenso könnte man nachträglich Filme mit Zeichensprache oder Untertiteln unterlegen.

Inhaltsbasierter Zugriff

MPEG-4 soll durch effiziente Organisation der audiovisuellen Daten effiziente Zugriffsoperationen wie Indizierung, Hyperlinks, Abfragen, Browsing, Up- und Download und Löschen liefern. Zum Beispiel können Informationen inhaltsbasiert aus online-Bibliotheken oder Reiseauskunftsystemen wiedergewonnen werden.

Kodierung von Daten mit natürlichem und synthetischem Ursprung

MPEG-4 soll Methoden liefern, natürliche mit synthetischen Szenen zu kombinieren. Dabei sollen die einzelnen Elemente separat kodiert werden und auch separat manipulierbar sein, um dem Benutzer die Möglichkeit zu geben, die Mischung dieser Daten interaktiv zu steuern. Zum Beispiel können Virtual Reality Anwendungen, Animationen und synthetische Audio-Sequenzen (z.B. MIDI)mit herkömmlichen Audio- und Videosequenzen für ein Videospiel kombiniert werden. Graphiken lassen sich so kodieren, daß die Perspektive des Betrachters oder eine wahlfreie andere Perspektive bei der Wiedergabe darstellbar ist.

1.2.2 Kodierung und Kompression

Kodierung und Synchronisation mehrerer nebenläufiger Datenströme

MPEG-4 soll die Ansichten aus mehreren Perspektiven und mehrere Tonspuren einer Szene sowie die Synchronisation der resultierenden Datenströme kodieren. Dabei soll die Redundanz, die zwangsläufig aufgrund des mehrfachen Bildmaterials auftritt zur erhöhten Kompressionsleistung genutzt werden. Außerdem soll die Kompatibilität mit anderen herkömmlichen Videoformaten dabei gewährleistet sein. Mit dieser Funktionalität können natürliche 3D Objekte dargestellt werden, allerdings ist dabei eine komplizierte Bildanalyse erforderlich. Diese Funktionalität ist zum Beispiel interessant für Multimedia-Entertainment-Anwendungen, wie z.B. Virtual-Reality-Spielen, 3D-Filmen oder Flugsimulatoren, sowie für multimediale Repräsentation und Ausbildung nützlich. Letzteres ist gerade im Bereich der Medizin von Relevanz, allein wenn es um die Darstellung des menschlichen Körper bzw. der einzelnen Organe bis hin zu kleinsten Gefäßen geht. Mit dieser Art der Vermittlung wird ein sehr hoher Grad an Flexibilität erreicht, der mit realen, künstlichen Modellen so nicht zu erreichen wäre.

Kodierleistung erhöhen

MPEG-4 soll einen höheren Kompressionsgrad liefern, der gerade bei Übertragung über die Mobilfunknetze von Interesse ist. Daher soll MPEG-4 bei gleicher Bitrate eine subjektiv bessere audiovisuelle Qualität erbringen. Andere Funktionalitäten können zu diesem Ziel in Konflikt stehen. Zum Beispiel können audiovisuelle Daten über Wege mit niedriger Bandbreite übertragen werden. Es können audiovisuelle Daten effizient auf Medien mit niedriger Kapazität gespeichert werden (Chipkarte).

1.2.3 Skalierbarkeit

räumliche und zeitliche Auflösung

MPEG-4 soll bei differenzierte Darstellung die Skalierbarkeit der zeitlichen und räumlichen Auflösung gewährleisten. Der Benutzer kann für die Objekte getrennt die gewünschte Qualität der Dekodierung festlegen. Bestände von Datenbanken können bei unterschiedlicher Skalierung und Auflösung betrachtet werden, um z.B. die Suche schneller zu gestalten.

Qualität (Bildauflösung)

Ebenso soll der Standard für die Skalierbarkeit der Qualität im Sinne von Bildauflösung sorgen.

Komplexität

Auch bei der Komplexität muß die Skalierbarkeit garantiert sein.

1.3 Objektrepräsentation

MPEG-4 erlaubt es auf Bildinhalte zuzugreifen und diese zu manipulieren, daher wird als Zugriffseinheit nicht wie bei MPEG-1 oder MPEG-2 das komplette Bild (z.B. I-Frame), sondern ein semantischer Teil des Gesamten als Objekt definiert. Ein Objekt ist ein Repräsentant von Bildinhalt und steht im Mittelpunkt von MPEG-4. Für Audioinhalte gilt dies in analoger Form, daher spricht man vom audiovisuellen Objekt (AVO). Die AVOs können natürlichen, d.h. mit Kamera und Mikrophon aufgezeichnet, oder synthetischen Ursprungs, also Computer generiert sein. Die Szenen, aus denen ein Film besteht, werden also aus AVOs komponiert.

Eine (einfache) audiovisuelle Szene kann zum Beispiel die Wettervorhersage im Fernsehen sein. Die Wetterkarte ist als Hintergrundbild kodiert und zudem noch computergeneriert. Im Vordergrund steht der Nachrichtensprecher, der die Karte erklärt. Seine Worte, d.h. Sprache ist als eigenständiges audiovisuelles Objekt zu betrachten, die mit dem Sprecher später bei der Wiedergabe synchronisiert werden muß. Als weiteres Objekt könnte man sich ein Nachrichtenlaufband am unteren Bildschirmrand vorstellen, daß die letzten Meldungen wiedergibt. Diese Informationen werden textuell zum Clientrechner übertragen und erst dort mit entsprechenden mitgelieferten Informationen wie Schriftart oder Schriftgröße auf dem Bildschirm des Betrachters dargestellt. D.h. es wird nicht der Text als Bitmap-Objekt über den Datenkanal verschickt, sondern die reine (Text-)Information zuzüglich Darstellungsinformation.

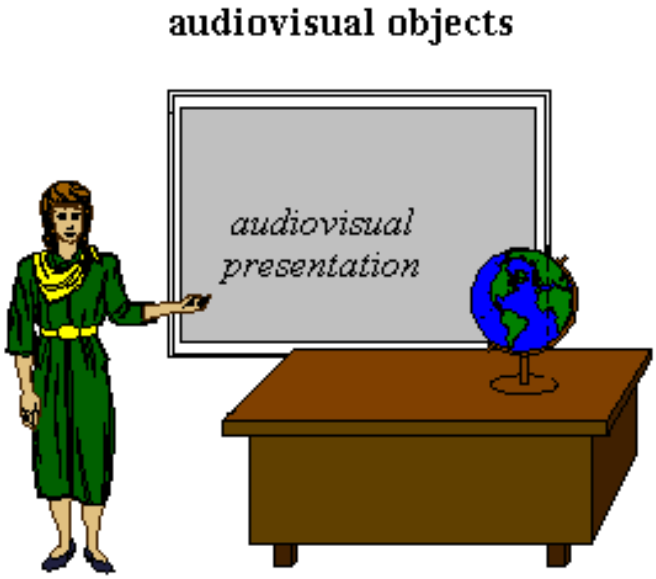


Abbildung 1-2: MPEG-4 Szene

Eine Szene unter MPEG-4 hat eine hierarchische Struktur und läßt sich als zyklenfreier, gerichteter Graph beschreiben. Die Wurzel entspricht der Szene selbst, während innere Knoten zusammengesetzte AVOs sind, anders ausgedrückt, eine Kante hat die Bedeutung „besteht aus“. Die Blätter entsprechen primitiven AVOs, also z.B. einem 2D Hintergrund, das Bild einer sprechenden Person (ohne Hintergrund) oder die Stimme der Person. MPEG-4 standardisiert eine Anzahl solcher primitiven AVOs, die in ihrer kodierten Repräsentation so effizient wie möglich gespeichert werden. Der Graph braucht nicht statisch zu sein und ist daher veränderbar, so kann der Benutzer Knoten hinzufügen, löschen oder ersetzen. Die Knoten besitzen Attribute, die ebenfalls geändert werden können.

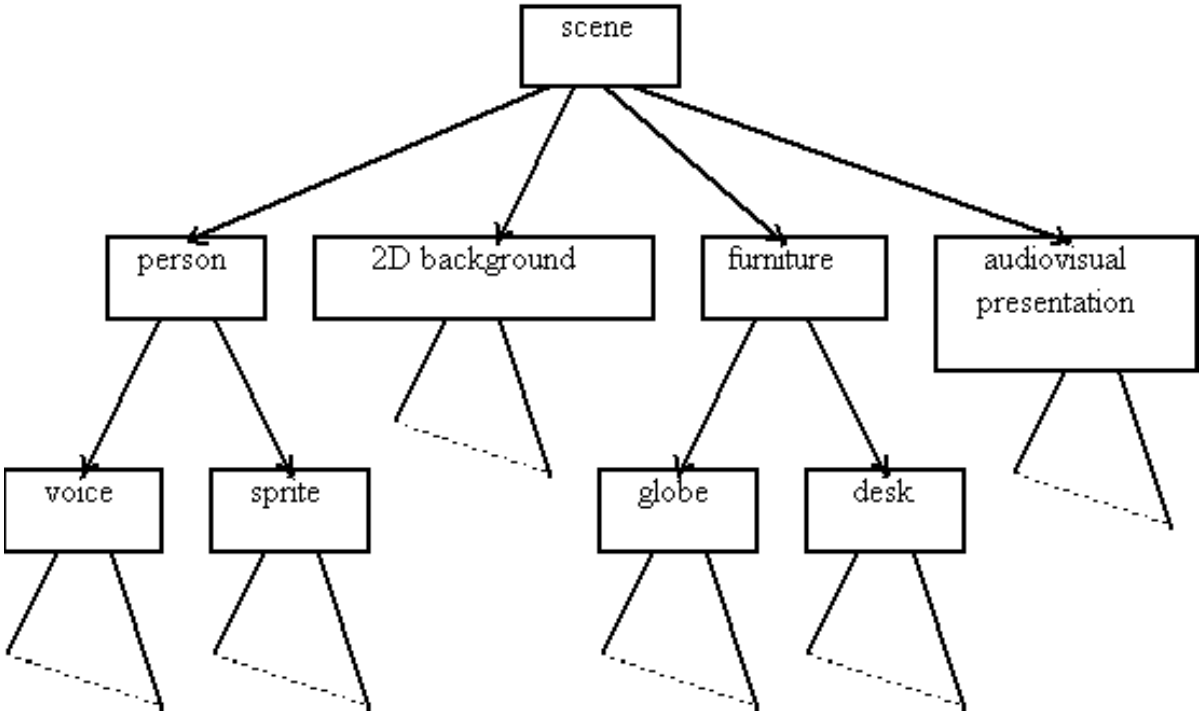


Abbildung 1-3: MPEG-4 Szenenbaum

AVOs haben zeitliche und räumliche Dimension, ebenso ein Koordinatensystem, welche die Grundlage für die räumliche und zeitliche Manipulation bieten. Das lokale Koordinatensystem der audiovisuellen Objekte bestimmt die Objekte in räumlicher und zeitlicher Position und Skalierung. AVOs werden durch Koordinatentransformation des lokalen Koordinatensystems des Objektes in ein globales Koordinatensystem des übergeordneten Knotens der Szenenbeschreibung in der Szene positioniert. Die Kompositionsebene besitzt ebenfalls Parameter mit denen der Benutzer den Ablauf der Szene bzw. das Verhalten einzelner AVOs steuern kann.

1.4 Multiplexing

Jedes AVO ist separat kodiert und kommt als Elementarstrom über das Netz. Daneben werden die Kompositionsinformationen der Szene, die Parameter der AVOs beschreiben, mitübertragen. Der Dekoder muß die Elementarströme und die Kontrollinformationen aus dem eintreffenden Datenstrom mit dem Demultiplexer zurückgewinnen. Die AVOs werden dekodiert und gemäß den Vorgaben der Kompositionsinformation angeordnet und bewegt. Es werden zwei Multiplexer unterschieden. Die FlexMux Schicht (Flexible Multplexing) ist durch MPEG-4 vollständig spezifiziert. Diese Schicht enthält ein Multiplexer Werkzeug, das die Elementarströme mit wenig Overhead gruppiert. Die TransMux Schicht (Transport Multiplexing) modelliert die Ebene, die Transportdienste anbietet, die mit dem geforderten Qualitätsbedarf übereinstimmt. Nur die Schnittstelle zu dieser Schicht ist von MPEG-4 spezifiziert. Es bleibt dem Benutzer oder dem Dienstanbieter überlassen, welcher passende, existierende Transportprotokollstapel eine spezifische TransMux Instanz wird.

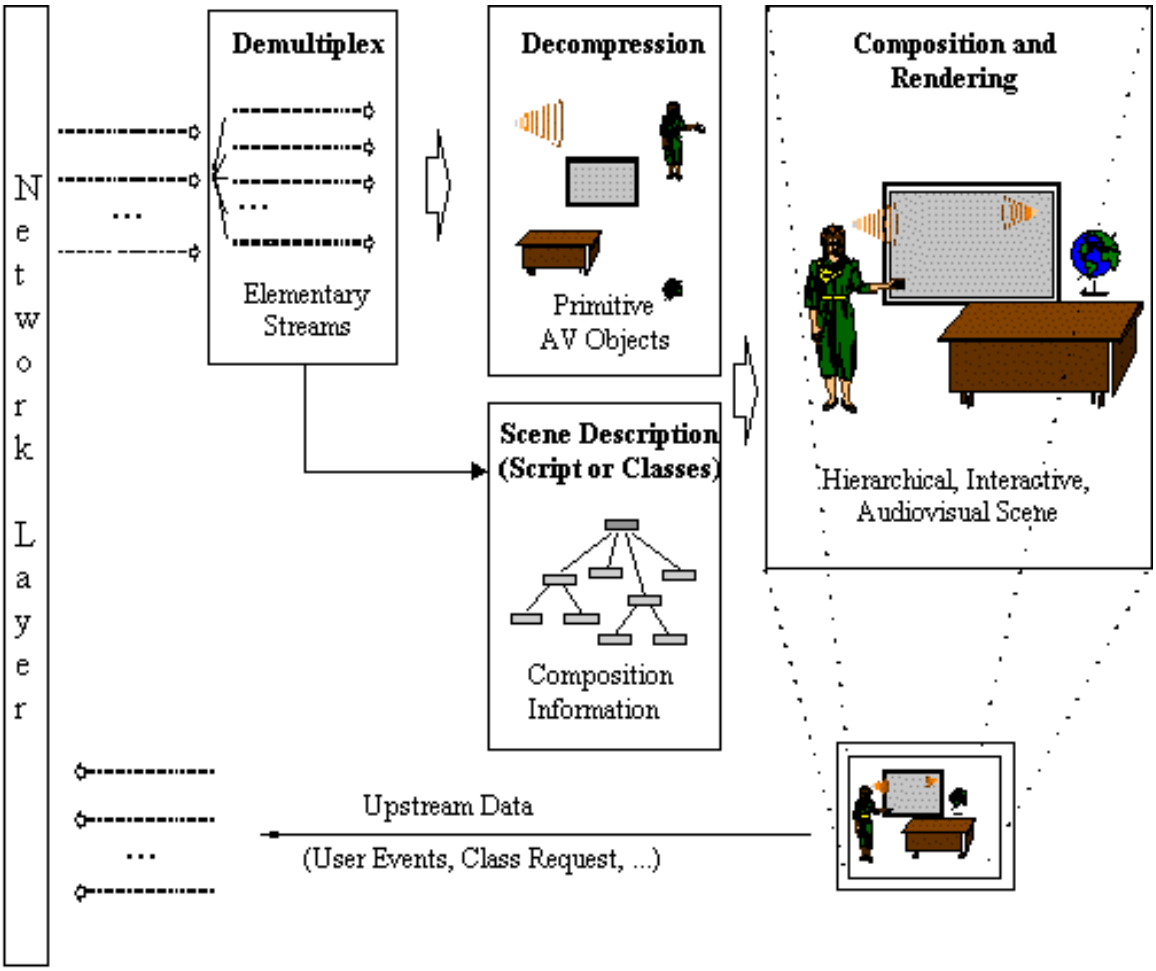


Abbildung 1-4: MPEG-4 Multiplexing

Über einen Rückkanal kann der Benutzer interaktiv Eingaben vornehmen und den Verlauf der Szene beeinflussen.

1.5 Benutzerinteraktivität

Die Kompositionsinformation einer Szene ist von den Daten der beteiligten AVOs strikt getrennt. Daher kann der Benutzer das Kompositionsskript interaktiv beeinflussen und manipulieren. Auf diese Weise werden die Abläufe der Szenen geändert. Die Änderung des Skriptes kann entweder auf Client- oder auf der Serverseite geschehen. Arbeitet der Benutzer auf dem Client, so kann er Manipulationen lokal durchführen. Dies hat den Nachteil (gerade bei schmalbandigen Netzen), daß alle AVOs und das Skript vollständig vom Server auf den Client übertragen werden müssen. Daher gibt es die Möglichkeit Änderungen des Ablaufs einer Szene auf der Serverseite vorzunehmen. Dann können auch nur die AVO Repräsentationen übertragen werden, die der Benutzer haben möchte. Diese Variante ist technisch aufwendiger. Die Einstellungen des Benutzers müssen von der Clientseite auf den Server übertragen werden, damit dort die AVOs ausgewählt werden können und das Kompositionsskript manipuliert werden kann, bevor die Daten an den Client gesendet werden.

Der Benutzer hat mehrere Möglichkeiten der Interaktivität. So kann er den Blickwinkel einer Szene ändern, ebenso wie die Sprache sofern dies den in dem AV-Material vorgesehen ist. Der Benutzer kann Objekte verschieben, löschen oder hinzufügen. Der Benutzer kann ein Ereignis triggern indem er auf ein bestimmtes Objekt clickt und so zum Beispiel einen Videostrom startet oder stoppt. Allerdings kann der Benutzer nur insoweit die Szene verändern, als daß es vom Autor der Szene vorgesehen ist. Abhängig vom Freiheitsgrad, den der Autor vorgibt, kann er die Szene verändern und manipulieren. Die Datenströme, die aus dem Netzwerk kommen sind als TransMux Ströme kodiert und werden in FlexMux Ströme demultiplext, die von einem passenden FlexMux Demultiplexer in Elementarströme aufgegliedert werden. Die elementaren Datenströme werden geparst und zu passenden Dekodern weitergeleitet. Der Dekoder stellt die Daten des AV-Objekts wieder her und führt alle notwendigen Operationen durch, um die originalen AV-Objekte zu rekonstruieren und bereitzustellen für die Wiedergabe auf dem Ausgabegerät.

1.6 Kodierung von visuellen Objekten

MPEG-4 stellt verschiedene Möglichkeiten bereit Bildmaterial effizient zu kodieren, zu manipulieren und zu übertragen. Die grundlegende (atomare) Einheit ist das sogenannte *video object* (VO). Das kann z.B. eine Textur, ein Bild oder eine sprechende Person (ohne Hintergrund) sein. MPEG-4 erlaubt es diese VOs zu dekodieren und darzustellen. Um dieses Ziel zu erreichen bietet der MPEG-4 Standard Lösungsmöglichkeiten in Form von Werkzeugen und Algorithmen an:

- effiziente Kompression von Bildern und Video
- effiziente Kompression von Texturen für *texture mapping* auf 2D oder 3D Gittern
- effiziente Kompression von impliziten 2D Gittern
- effiziente Kompression von sich zeitlich verändernden geometrischen Strömen, die Gitter animieren
- effizienter zufälliger Zugriff zu allen Typen von visuellen Objekten

- erweiterte Funktionalität in Manipulation von Bildern und Videosequenzen
- inhaltsbasierte Kodierung von Bildern und Video
- inhaltsbasierte Skalierung von Texturen, Bildern und Video
- zeitliche, räumliche und qualitative Skalierung
- Fehlerrobustheit in zu Fehler neigenden Umgebungen

Für synthetische Objekte, die eine Untermenge der größeren Klasse von Computergraphiken bilden, stellt MPEG-4 einen Standard, der zum Beispiel folgende Objekte kodiert:

- parametrische Beschreibung der synthetischen Repräsentation des menschlichen Gesichtes und Körpers, sowie Animationsströme von Gesicht und Körper
- statische und dynamische Gitterkodierung mit texture mapping
- Texturkodierung für blickpunktabhängige Anwendungen

Als Beispiel sei die Animation eines Gesichtes betrachtet. Das Gesicht ist ein Objekt mit der Geometrie eines menschlichen Gesichtes, bereit dargestellt und animiert zu werden. Die Form, Textur und der Ausdruck des Gesichtes wird normalerweise von dem Bitstrom kontrolliert, der Instanzen des *Facial Definition Parameter (FDP) set* und/oder *Facial Animation Parameter (FAP) set* enthält. Bei der Konstruktion enthält das Gesichtsobjekt ein generisches Gesicht mit neutralem Gesichtsausdruck. Das Gesicht kann nun dargestellt werden. Es kann sofort FAPs vom Bitstrom empfangen, die dem Gesicht Animation, also Gesichtsausdruck, Sprache oder ähnliches verleihen.

In Abbildung 1-5 ist die Kodierung von Video schematisch dargestellt. Die illustrierten Algorithmen kodieren rechteckige sowie willkürlich geformte Bildsequenzen. Die grundlegenden Kodierverfahren umfassen *shape coding* für willkürlich geformte Objekte, *motion compensation* sowie DCT basierte Kodierung. MPEG-4 kann die Kodierleistung für inhaltbasierte Kodierung dadurch verbessern, daß es jedes AV-Objekt einer Szene separat mit einem zu dem Objekt passendem Verfahren (*motion prediction*) kodiert.

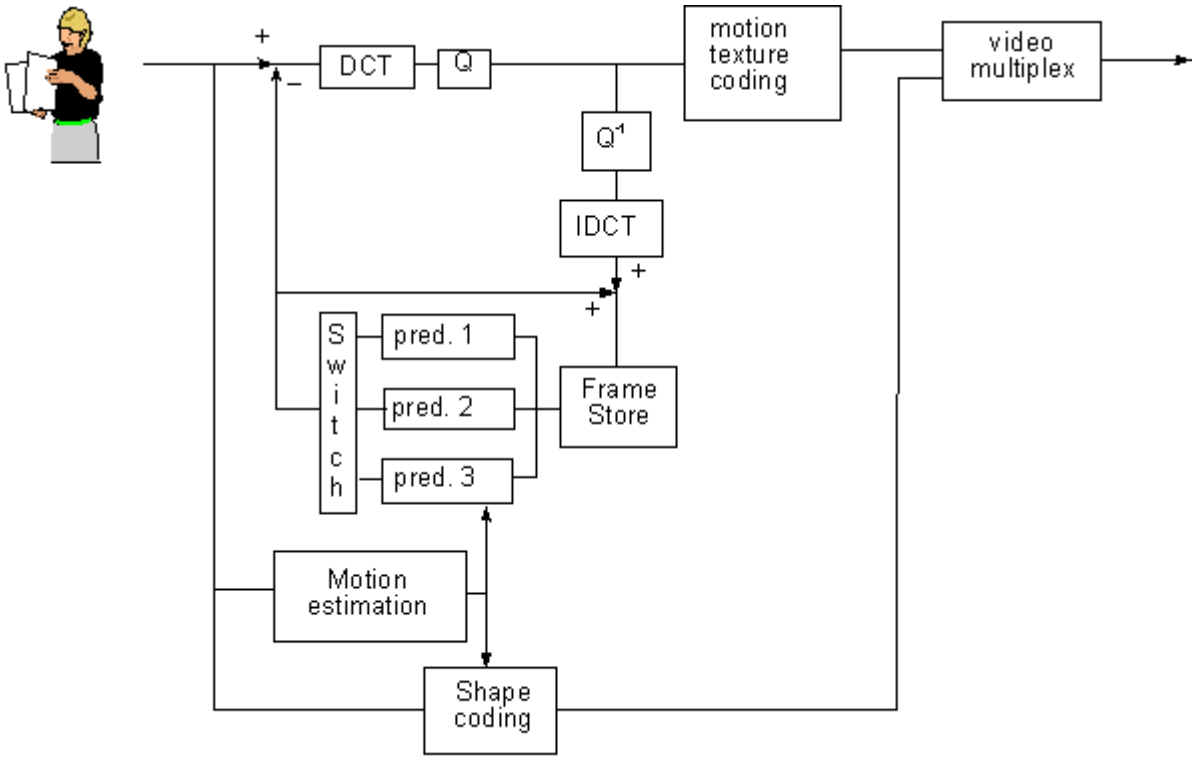


Abbildung 1-5: MPEG-4 VideoKodierung

Folgende *motion prediction* Techniken können benutzt werden, um effiziente Kodierung und flexible Präsentation der Objekte zu erreichen:

- Standard 8x8 oder 16x16 Pixel blockbasierte *motion estimation* und *compensation*
- Globale motion compensation, die acht Bewegungsparameter benutzt, die eine affine Transformation beschreiben
- Globale motion compensation auf statischen *Sprites* basierend. Ein Sprite ist möglicherweise ein großes (unbewegtes) Bild, welches ein Panorama beschreibt.
- Globale motion compensation auf dynamischen Sprites basierend.

Effiziente Kodierung von visuellen Texturen und (unbewegten) Bildern wird von dem Visuellen Textur Modus von MPEG-4 unterstützt. Dieser Modus beruht auf einem speziellen *Wavelet* Algorithmus, der sehr hohe Kodierleistung bei einer großen Variation an Bitraten anbietet.

2 MPEG-7

MPEG-7 wird eine standardisierte Beschreibung von multimedialen Informationen sein. Diese multimedialen Daten müssen keinem bestimmten Format entsprechen. Die Beschreibung der Daten wird mit dem Inhalt assoziiert sein. Dadurch soll dem Nutzer schnelles und effizientes Suchen nach Daten mit bestimmten gewünschten multimedialen Inhalt gewährleistet werden. MPEG-7 befindet sich momentan noch in der Entwicklungsphase und ist daher zu diesem Zeitpunkt nicht als Standard zu betrachten.

2.1 Motivation

Es treten immer häufiger audiovisuelle Daten in digitaler Form auf, die durch die globale Vernetzung auch eigentlich jedermann zugänglich sind bzw. gemacht werden kann. Aber bevor man das Material nutzen kann, so muß man es erst einmal gefunden haben. Die Suche wird durch die steigende Verfügbarkeit des potentiell interessanten Materials schwieriger. Momentan existieren Lösungen, die dem Nutzer das Auffinden von textbasiertem Material ermöglichen (Suchmaschinen). Es zeigt sich, daß der Bedarf nach Informationen und deren Lokalisierung sehr groß ist. Momentan ist es nicht möglich nach Informationen audiovisueller Art zu suchen, da bisher keine generelle erkennbare Beschreibung dieser Materialien existieren. Man kann z.B. noch nicht nach dem Motorrad aus dem Film „Terminator II“ suchen. In speziellen Fällen existieren Teillösungen, welche die Suche nach Farbe oder Textur von Objekten anbieten. Es geht bei MPEG-7 also um die Frage nach dem Auffinden von multimedialen Inhalten. Dies ist zum Beispiel interessant, wenn es um eine Auswahl an digitalen Fernseh- oder Radiokanälen geht, die heute in einem immer größer werdenden Angebot konkurrieren. Die Suche ist also nicht nur auf Datenbanken oder Archive beschränkt.

2.2 Beschreibung des Standard

MPEG hat sich vorgenommen einen Standard zu entwickeln mit dem es möglich sein soll, multimediale Daten so zu beschreiben, daß der Benutzer effizient nach ihnen suchen kann. Die Möglichkeiten der inhaltlichen Beschreibungen soll also aufgeweitet werden. MPEG-7 („Multimedia Content Description Interface“) wird einen Standard an Deskriptoren entwickeln, die benutzt werden sollen, um die verschiedenen Typen multimedialer Information zu beschreiben. Außerdem sollen auch andere Deskriptoren spezifiziert werden können, ebenso wie ganze Beschreibungsschemata (Description Scheme) für die Deskriptoren und ihre Beziehungen untereinander. Diese Beschreibung, also die Deskriptoren und die Beschreibungsschemata, sollen mit dem Inhalt selbst verknüpft sein, und so dem Benutzer eine schnelle und effiziente Suche ermöglichen. Dazu möchte MPEG-7 eine Sprache entwickeln, welche die Beschreibungsschemata spezifiziert, die sogenannte „Description Definition Language“. Audiovisuelles Material, mit dem MPEG-7 Daten assoziiert sind, kann indexiert und gesucht werden. Solche Materialien können sein: Photos, Graphiken, 3D Modelle, Audio, Sprache oder Video. Spezielle Fälle dieser Informationen können auch z.B. Gesichtsausdrücke sein. MPEG-7 wird also eine standardisierte Repräsentation von audiovisuellem Material sein (ebenso wie MPEG1,

2, 4). Die MPEG-7 Deskriptoren sind unabhängig von der Art und Weise der Speicherung bzw. Kodierung des zu assoziierenden Materials.

Um ein besseres Verständnis der MPEG-7 Terminologie Deskriptor (D), Description Scheme (DS) und Description Definition Language (DDL) zu geben sind die Abbildungen 2-1 und 2-2 von Interesse. Die Kästchen in den Abbildungen umfassen die normativen Elemente des MPEG-7 Standards, diese Elemente müssen nicht unbedingt in allen MPEG-7 Applikationen erscheinen.

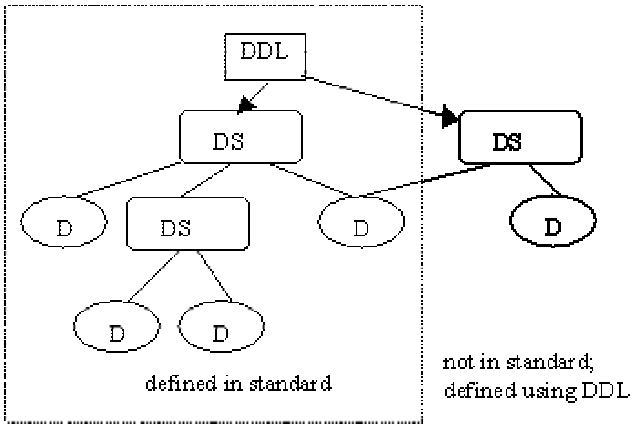


Abbildung 2-1: mögliche Beziehungen zwischen Deskriptoren(D) und Description Schemes(DS)

Abbildung 2-1 zeigt die Erweiterbarkeit der oben erwähnten Konzepte. Die Description Schemes werden durch den Gebrauch der Description Definition Language generiert. Es ist ebenso möglich ein neues Description Scheme durch die Verwendung eines existierenden Description Scheme zu erstellen.

Abbildung 2-2 stellt den Mechanismus dar, ein Beschreibungs Schema mit der Description Definition Language zu erstellen, das die Grundlage für die Generierung einer Beschreibung bildet.

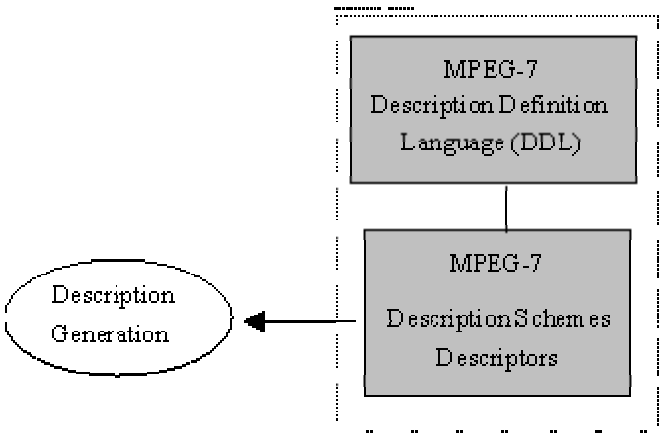


Abbildung 2-2: Generierung von MPEG-7 Beschreibungen

2.3 Umfang des Standard

MPEG-7 adressiert Anwendungen, die gespeichert (on- oder offline) oder gestreamt (Fernsehübertragungen) werden, ebenso Echzeit- und nicht Echtzeitumgebungen.

Abbildung 2-3: Umfang von MPEG-7

Abbildung 2-3 soll den Umfang von MPEG-7 anhand einer Prozeßkette verdeutlichen. Diese Kette beinhaltet die Merkmalsanalyse, die Beschreibung selbst und die Suchmaschine (Anwendung). Die automatische Erkennung von Merkmalen des audiovisuellen Materials gehört nicht zum Umfang von MPEG-7. Dazu werden keine Algorithmen bereitgestellt. Auch die Suchmaschinen werden nicht zum Umfang des Standard gehören. Dies ist beides einfach nicht notwendig, und hat auch den Grund den Raum für Wettbewerb in diesen Kategorien nicht einzuengen und die Möglichkeit für Verbesserungen zu geben. Der Hauptaugenmerk liegt auf der Bereitstellung von (neuen) Lösungsansätzen für die Beschreibung audiovisueller Inhalte. Die Adressierung rein textbasierter Dokumente soll nicht zum Ziel von MPEG-7 gehören. Dennoch kann AV-Material zusätzlich auf Textdokumente verweisen. Die Art und Weise wie Nutzeranfragen mit MPEG-7 Beschreibungen beantwortet werden liegt nicht im Zugriffsbereich von MPEG-7. Prinzipiell kann jedes AV-Material, das mit MPEG-7 Metadaten verknüpft ist, gefunden werden. Videomaterial kann z.B. mit Sprache, Video oder Musik gefunden werden. Es ist Aufgabe der Suchmaschine die Anfrage und die MPEG-7 Beschreibungen zur Deckung zu bringen.

Eine genauere Übersicht über das Zusammenspiel von Merkmalsextraktion, MPEG-7 Deskriptionen und Suchmaschinen gibt Abbildung 2-4.

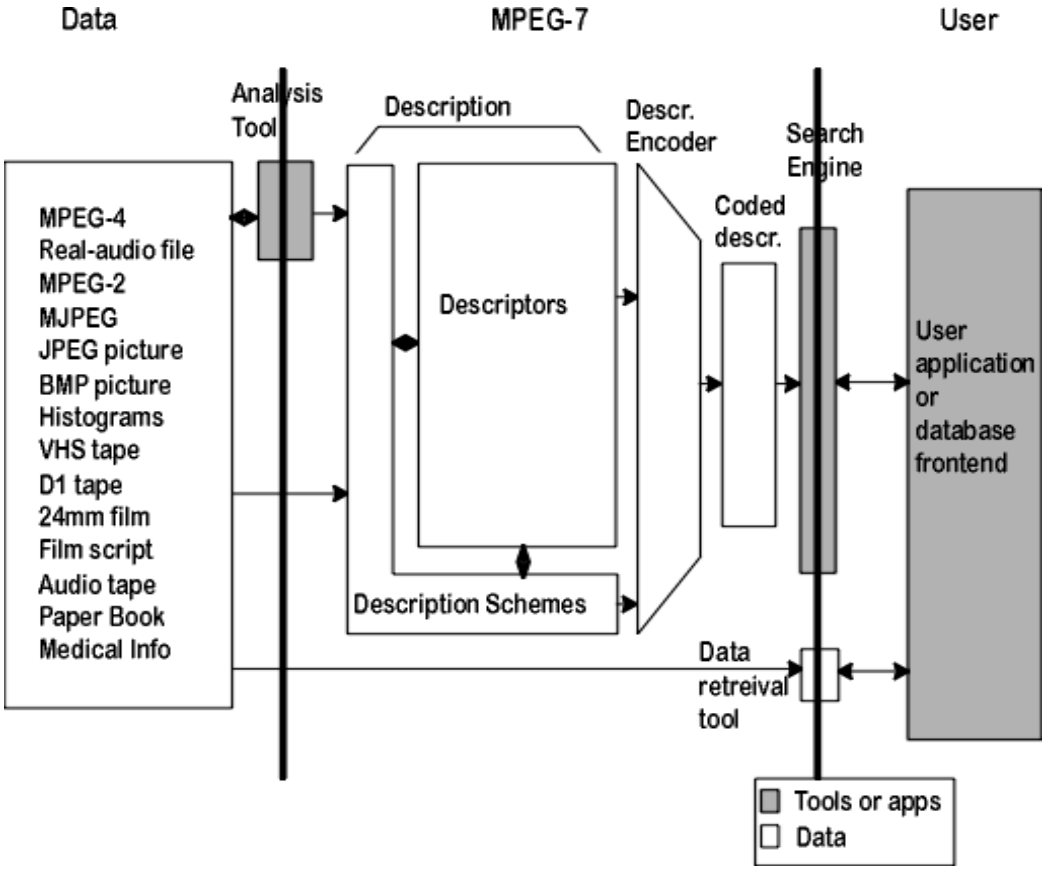


Abbildung 2-4: MPEG-7 Rolle

2.4 Art der Beschreibung

MPEG-7 ermöglicht verschiedene Abstufungen innerhalb der Beschreibung. Dies gibt einem die Möglichkeit nach verschiedenen Unterscheidungsstufen. Die beschreibenden Merkmale müssen innerhalb des Kontextes aussagekräftig sein. Daher unterscheiden sie sich für unterschiedliche Nutzerbereiche, wie auch für unterschiedliche Anwendungen. Das bedeutet, daß das selbe Material unter Zuhilfenahme unterschiedlicher Merkmalstypen beschrieben werden kann, je nach Einsatzgebiet der Anwendung.

Einfache Abstraktionsmaße für visuelle Materialien sind z.B. Oberflächenbeschaffenheit, Größe, Textur, Bewegung oder Position, Tonhöhe, Tempo, Tempoveränderung, Position und Raum. Komplexe Abstraktionsmaße geben dann semantische Informationen wieder: „Dies ist eine Szene mit einem braunen bellenden Hund im Vordergrund links, und einem blauen herabfallenden Ball auf der rechten Seite, mit dem Hintergrundgeräusch fahrender Autos.“ Es wird Zwischenstufen dieser Abstraktionsmaße geben. Diese Beschreibungen werden im Hinblick auf die Suche in einer effizienten Art und Weise kodiert. Der Grad der Abstraktion hängt von der Art und Weise der

Merkmalserkennung ab, viele einfache Merkmale können vollautomatisch erkannt werden, sehr komplexe Merkmale bedürfen der menschlichen Interaktion.

Außer der inhaltlichen Beschreibung können auch andere Informationstypen über die multimedialen Daten herangezogen werden.

- **die Form** dies kann z.B. das Kodierschema (JPEG) oder auch die Größe der Datei sein. Damit kann der Nutzer entscheiden, ob er das Material überhaupt lesen kann.
- **Zugriffsbedingungen** dies könnte die Copyright Information sowie den Preis beinhalten
- **Klasse/Kategorie** dies könnte z.B. eine FSK-Einteilung beinhalten, oder einen Verweis auf eine schon definierte Kategorie
- **Links** diese Information kann verwendet werden, um auf andere Quellen zu verweisen
- **Inhalt, Autor, Aufnahmedatum** Informationen zum Inhalt des Materials, zum Ersteller, etc.

2.5 Anwendungen und Anwendungsbereiche

Einige Beispiele zu Anwendungen:

- digitale Bibliotheken (Bilderkatalog, musikalisches Wörterbuch)
- Multimedia Verzeichnis (Gelbe Seiten)
- Auswahl an Mediensendungen (Fernseh- und Radiokanäle)
- Multimedia Editierung (personalisierter elektronischer News Service)

und Beispiele zu Anwendungsbereichen:

- Bildung und Erziehung
- Journalismus
- Touristeninformation
- Entertainment
- Geographische Informationssysteme
- Medizin
- Shopping
- Architektur
- Sozialwesen
- Film-, Video-, Radioarchive

Der MPEG-7 Standard wird nicht die Frage klären wie Benutzeranfragen beantwortet werden. Im Prinzip könnte jedes audiovisuelle Datenmaterial mit jeder Art von Anfrage gefunden werden.

Beispielsweise kann Videomaterial durch Sprach-, Video- oder Musikdaten gefunden werden. Es liegt an der Suchmaschine die Anfrage bzw. die Inhalte des Anfragematerials mit der vorhandenen MPEG-7 Beschreibung abzugleichen.

Als Beispiele sind zu nennen:

i. Musik
   Man könnte ein paar Noten mit dem Keyboard (o.ä.) eingeben und die Suche danach starten. Abhängig von der Beschreibung des Materials im Suchbereich werden die Ergebnisse mit der Anfrage mehr oder weniger stark in Verbindung stehen.
ii. Grafik
   Zeichnet man ein paar Linien auf dem Bildschirm, so erhält man eine Auswahl an Bildern, Graphiken, Logos etc.
iii. Bilder
   Man kann nach Bildern mit ähnlichen Eigenschaften wie die des Ausgangsobjektes suchen. Eigenschaften könnten zum Beispiel Farbe, Textur oder Form sein.
iv. Bewegung
   Man kann Bewegungen vielleicht an einer Auswahl von Objekten beschreiben. Als Ergebnis erhält man Animationen, die den vorgegebenen Bewegungsmustern entsprechen.
v. Stimme
   Benutzt man eine Stimme zur Suchanfrage, so erhält man eine Liste von CDs, Videos, Reden, Interviews, die diese Stimme enthalten.

2.6 Arbeitsplan

Der vorläufige Arbeitsplan bis zum Abschluß von MPEG-7 als ISO-Standard ist folgender:

call for proposals	october 1998
working draft	december 1999
committee draft	october 2000
final committee draft	february 2001
draft international standard	july 2001
international standard	september 2001

MPEG-7 ist der erste Versuch die Beschreibung aller Arten von multimedialer Information in einem Standard zusammenzufassen. MPEG-7 wird sehr hilfreich sein bei der Suche nach jeglichem audiovisuellen Material und darüberhinaus die Suche (auch nach weiterführendem Material) stark vereinfachen. MPEG-7 ist aber nur dann für jedermann interessant, wenn auch die Anbindung dieses

Standards an die Umgebung z.B. die Suchmaschinen stimmt. Auch der Bedarf nach Merkmalsextraktion sollte nicht vernachlässigt und unterschätzt werden.

3 Literatur

[Dbox 99] MPEG Grundlagen
http://www.d-box.to/technik/mpeg1.html
[Stand: April 1999]

[Effe 98] Wolfgang Effelsberg, Ralf Steinmetz:
„Video compression techniques“
dpunkt-Verlag, 1998

[Leon 96] Leonardo
http://www.cselt.stet.it/ufv/leonardo/paper/isce96.htm

[Meyr 98] Jan Meyer
Hausarbeit MPEG-7
http://o2f.fh-potsdam.de/jan/mpeg7/index.html
[Stand: November 1998]

[MPEG 99a] MPEG Home Page
http://drogo.cselt.stet.it/mpeg
[Stand: April 1999]

[MPEG 99b] MPEG Home Page
http://www.mpeg.org

[Koen 97a] Rob Koenen
MPEG-4 Overview - (Fribourg Version) (N1909)
International Organisation for Standardisation ISO/IEC JTC1/SC29/WG11
http://wwwam.hhi.de/mpeg-video/standards/mpeg-4.htm
[Stand: Oktober 1997]

[Koen 97b] Rob Koenen
MPEG-7: Context and Objectives (v.5 - Fribourg) (N1920)
International Organisation for Standardisation ISO/IEC JTC1/SC29/WG11
http://wwwam.hhi.de/mpeg-video/standards/mpeg-7.htm
[Stand: November 1998]

[Raut 99] Mathias Rautenberg
„MPEG-4: Ein neuer Standard für mehr Funktionalität in Multimedia Anwendungen“
Band 22, Heft 2, Informatik Spektrum, April 1999

Digitales Video

Kompression, Datenformate und Anwendungen

Christian Klaas

FB Informatik, Uni Dortmund
chris@knipp.de

Zusammenfassung

Diese Seminarausarbeitung beschäftigt sich mit verschiedenen Datenformate und Standards und deren Einsatz in Videokonferenzen. Am Beispiel von H.263 wird die Vorgehensweise einer Videokompression erklärt.

1 Einführung

Der Seminarvortrag behandelt die verschiedenen Videoformate und deren Einsatz im täglichen Bereich. Videoformate werden in Videokonferenzsystemen eingesetzt und sind, wie alle anderen Telekommunikationstechnologien auch, erst dann von wirklich praktischem Interesse, wenn die von ihnen verwendete Protokolle international standardisiert sind. Somit sind proprietäre Lösungen mit dem Nachteil versehen, daß alle an der Videokonferenz beteiligten die selbe Software einsetzen müssen. Das erhöht den Verwaltungsaufwand und verhindert Spontaneität.

Internationale Standards im Bereich der Telekommunikation sind herstellerunabhängig und werden von der ITU (International Telecommunications Union, die frühere CCITT) erlassen. Neben der ITU arbeitet auch die IETF (Internet Engineering Task Force) an entsprechenden Standardisierungsvorschlägen: sie hat das RTP (Real Time Transport Protocol) und das RSVP (Resource Reservation Protocol) erarbeitet. Die ITU wurde am 17. Mai 1865 in Paris gegründet und beschäftigte sich von Anfang an mit der Entwicklung von Standards. Mittlerweile ist die ITU in Genf ansässig, eine UN-Behörde geworden und beschäftigt 700 Mitarbeiter aus 71 Ländern. Der ITU gehören 190 Staaten an und die Mitgliedschaft steht allen souveränen Staaten offen [ITU 99]. Die ITU nimmt keine hoheitlichen Aufgaben wahr und vertritt keine finanziellen Interessen, aber die Empfehlungen werden im allgemeinen von den Mitgliedsländern verbindlich übernommen. Die schnelle Entwicklung des Internets und die Entwicklung von Standards durch Firmen führte in den letzten Jahren zu Problemen. Die größten Schwierigkeiten gibt es aber, seit der weltweiten Deregulierung der Telekommunikation. Da die Nationen nicht mehr die relevanten Partner sind, muß sich die ITU mit Regierungsbevollmächtigten auseinander setzen, die wieder die globalen Netzbetreiber vertreten sollen. Und die Netzbetreiber fühlen sich nicht richtig vertreten, da sie Ihre wirtschaftlichen Interessen nicht so einfach durchsetzen können. Inwieweit die ITU die unterschiedlichen Interessen unter einen Hut bekommt, wird die Zeit zeigen.

Die in den weiteren Kapiteln vorgestellten Protokolle sind für Videokonferenzen gedacht und haben sich in der Praxis bewährt. Sie sind über Festverbindungen genauso wie im Internet einsetzbar und unterteilen sich in verschiedene Kategorien. Die Protokolle für Videokonferenzen unterteilen sich zum einen in Protokolle für verschiedene Übertragungskapazitäten und zum anderen in Übertragungsprotokolle für die verschiedenen Übertragungsarten. Die Abb.1-1 zeigt die verschiedenen Standards ein-

geordnet anhand der Übertragungskapazität. Die erste Zeile zeigt die übergeordneten Videokonferenzprotokolle, welche die untergeordneten Protokolle unterstützen und die Rahmenstandards für die verschiedenen Übertragungsmedien sind. Sie vermitteln eine Übersicht, verweisen viel auf die anderen Empfehlungen, erläutern die Zusammenhänge zwischen denselben und lassen sich in folgende Kategorien unterteilen:

- H.320 Narrowband, Vielfaches der ISDN Kapazität von 64 kBit/s
- H.324 Low-Bitrate, Modemleitung 28k - 33 kBit/s
- H.322 Iso-Ethernet, Ethernet mit einer garantierten Übertragungsrate von 6 MBit/s
- H.323 Ethernet, normales Ethernet mir einer Übertragungsrate von bis zu 10 MBit/s
- H.321 ATM, ermöglicht eine Übertragung von bis zu 155 MBit/s
- H.310 High Res, Übertragungsrate von mehr als 655 MBit/s möglich

	H.320 Narrowband	**H.324 Low-Bitrate**	**H.322 Iso-Ethernet**	**H.323 Ethernet**	**H.321 ATM**	**H.310 High Res**
Video	H.261	H.261 H.263	H.261	H.261 H.263	H.261	MPEG-2 H.261
Audio	G.711 G.722 G.728	G.723	G.711 G.722 G.723 G.728	G.711 G.722 G.723 G.728 G.729	G.711 G.722 G.728	MPEG-1 MPEG-2 G.7xx
Data	T.120	T.120 T.434 T.84	T.120	T.120	T.120 H.281 (H.224)	T.120
Multi.	H.221	H.223	H.221	H.22z	H.221	H.222.1 H.221
Signal	H.230 H.242	H.245	H.230 H.242	H.230 H.245	H.230 H.242	H.245
Multi-point	H.243	N/A	H.243	N/A	H.243	N/A

Abb. 1-1: Aufbau der Protokolle

Bei den Videokonferenzprotokollen sind H.261 und H.263 die beiden meist genutzten und eingesetzten Protokolle für Videoübertragungen. Bei Audioübertragungen spielt die Sprachqualität eine entscheidende Rolle, so daß die Reduktion der Daten durch das Weglassen von leisen Tönen und das Überlagern des Sprachbandes zu einer Verringerung der Datenmenge führt. Hierbei sind vor allem die G.7xx Protokolle im Einsatz. T.120 bietet die Standardisierung des Datenaustausches (Datenübertragungen bei Videokonferenzen sind Standbilder, Whiteboards oder Dokumente, etc.) T.120 ist hierbei als übergeordneter Standard zu sehen, welcher verschiedene T.1xx Standards beinhalten kann und die Rahmenbedingungen für die Übertragung setzt. Für Multisession bietet sich das H.221 Protokoll an, welches mit H.223 und H.225 die Transport-, Multiplex- und Synchronisationsmechanismen für die verschiedenen Datentypen spezifiziert. Zur Signalverarbeitung ist das Protokoll H.230 und die beiden Weiterentwicklungen H.242 und H.245 vorgesehen, welche mit H.243 zusammen für die Steuerung (Verbindungsauf und -abbau, Vereinbarung der Betriebsparameter etc.) zuständig sind.

2 H.261

1984 begann eine Arbeitsgruppe vom ITU sich ein Verfahren zu überlegen, wie man Bewegtbilder kodieren kann um die Datenmenge zu verringern. Ein Final Draft wurde 1990 veröffentlicht und die Verabschiedung von H.261 war 1993. Dabei definiert H.261 eine Methode zur Datenkompression der digitalisierten Bildinformation für Videokonferenzsysteme über ISDN. Der Vorteil dieses Verfahrens gegenüber MPEG-1 liegt in der Skalierbarkeit. Aufgrund der abzusehenden Verbreitung von ISDN hat man sich für p*64 kBit/s entschieden. Dabei H.261 übernimmt eine ähnliche Rolle wie G.711 bei der Toncodierung; jedes Gerät muß H.261 beherrschen, um die Interoperatibilität sicherzustellen.

	Auflösung		Unterstütztes Übertragungs-		Bitrate (Mbit/s) (unkomprimiert)			
Bildformat	x	y	Protokoll		10 Bilder/s		30 Bilder/s	
					s/w 8Bit	Farbe 12Bit	s/w 8Bit	Farbe 12Bit
SQCIF	128	96	-	H.263	1.0	1.5	3.0	4.4
QCIF	176	144	H.261	H.263	2.0	3.0	6.1	9.1
CIF	352	288	H.261	H.263	8.1	12.2	24.3	36.5
4CIF	704	576	-	H.263	32.4	48.7	97.3	146.0
16CIF	1408	1152	-	H.263	129.8	194.6	389.3	583.9

Abb.2-1: Die CIF-Formate (Common Intermediate Format)

Im Standard wird neben dem Vollbild (CIF) auch eine minimale Bildgröße definiert (QCIF: quarter-screen video image), die ein H.261-kompatibles System unterstützen muß. Das Image Format liegt im CIF Format bei 352x288 und QCIF Format bei 176x144 Pixeln.

Der Algorithmus ist ähnlich dem Protokoll von H.263, welches in dem nächsten Kapitel genau erklärt wird. H.261 hat sich mittlerweile als ein Internet-Standard etabliert und ist durch die Skalierbarkeit anhand von ISDN Kanälen auch für den Privat Anwender sinnvoll einsetzbar. Durch den Einsatz von Internet-Protokollen, wie RTP und RTCP über Multicast oder Unicast hat sich H.261 durchgesetzt. Im Gegensatz zu MPEG-1 hat H.261 die Möglichkeit die Qualität der Übertragung und somit die Menge der Daten den Gegebenheiten an zu passen. Der Encoding Einheit ist dabei das entscheidende Glied um die Datenmenge zu regulieren. In der Praxis kommen dann noch ein paar Faktoren hinzu, die das Bild, welches der Anwender sieht, beeinflussen. Die Encoding Einheit, im weiteren Verlauf Coding Control genannt, kann mit intelligenten Strategien das Bild für den Anwender verbessern. Die Einflußfaktoren und Strategien im folgenden:

- Rechenpower, in der heutigen Zeit weniger ein Einflußfaktor, aber zur Zeit als dieser Standard entwickelt wurde war die Leistung der meisten Rechner noch nicht geeignet die maximale Framegröße zu übertragen. Frameverluste und schlechtere Bildqualität wurden erst durch mehr Rechenpower weniger.
- Im Internet ist die Bandbreite nicht überall gleich groß verfügbar, so daß sich die Coding Control auf die Bandbreite einstellen kann. Dabei kann zum einen die Bildqualität gesenkt werden oder zum anderen die Framerate sinken.

- Da die Übertragung der Daten im Internet über das UDP Protokoll erfolgt, gibt es keine Bestätigung für angekommene Pakete. Die Coding Control muß in einer gewissen Zeitspanne ein Komplettbild übertragen.
- Die Coding Control Einheit sollte sich intelligent verhalten und die zu übertragenden Makroblöcke nicht chronologisch übertragen, sondern gezielt solche Makroblöcke übertragen, welche sich gegenüber den vorangegangenen verändert haben.

Der Nachfolger von H.261 das Protokoll H.263 hat dagegen viele Verbesserungen erhalten, die eine optimiertere und qualitativ besser Bildqualität bei gleicher Übertragungsrate bietet.

3 H.263

H.263 wurde 1996 verabschiedet und das Einsatzgebiet sollte vor allem die Videokodierung für niedrige Bitraten sein. Aber wie sich herausstellte, waren die Umsetzungen des H.263 Protokolls auch bei höheren Bitraten besser als H.261. Das liegt zum einen an der besseren Bildqualität und zum anderen an der besseren Bewegungsvektorsuche. Außerdem sind viele Teile der hierarchischen Struktur des Datenstroms optional, so daß eine bessere Fehlerkorrektur eingebaut werden kann.

Das Farbmodell, welches auch H.261 benutzt, ist das YCC oder auch YUV[1] genannt. YCbCr im Verhältnis 4:2:2, wobei das Bild in Makroblöcke von 16x16 zerlegt wird. 4 Blöcke mit der Größe 8x8 als Luminanzmatrix und 2 Crominanzmatrizen der Größe 16x16.

Bei vielen verlustbehafteten Bildformaten und Bewegtbildcodecs wird der YUV-Farbraum genutzt. Das geschieht einerseits aus Kompatibilitätsgründen zur Fernsehtechnik, und andererseits, weil dieser Farbraum Chroma-Subsampling zuläßt. Beim Chroma Subsambsampling werden Blau und Rotdifferenz mit einer geringeren räumlichen Auflösung gespeichert als die Helligkeit. Sichtbar ist dieser (mathematische) Informationsverlust in der Regel nicht, da das menschliche Auge Helligkeitsunterschiede wesentlich feiner auflösen kann als Farbunterschiede. Die Arbeitsweise des Encoders zum Erstellen des Datenstroms wird im nächsten Kapitel erklärt.

3.1 Arbeitsweise eines H.263-Encoders

Wie schon gesagt wird das Bild in 16x16 Pixel große Makroblöcke zerlegt. Für jeden dieser Makroblöcke muß die CC (Coding Control) des H.263 Encoders eine der folgenden Alternativen auswählen:

- Der Makroblock kann im INTRA-Mode[2] übertragen werden. Die Daten des Makroblocks werden mittels DCT[3] transformiert (T) und quantisiert (Q). Der Makroblock kann auf der Empfängerseite ohne weitere Daten komplett decodiert werden.
- Der Makroblock kann im INTER-Mode[4] übertragen werden. Für alle Pixel des Makroblocks wird die Differenz zwischen dem aktuellen und dem zuletzt kodierten Bild berechnen. Diese Differenzen werden DCT -transformiert und quantisiert. Der Decoder muß für die Dekodierung des Blocks das zuvor gesendete Bild hinzuziehen (Inter-Picture-Prediction)

[1] YUV ist ein Farbraum bei dem nicht die drei Grundfarben gemischt werden, sondern man verwendet stattdessen die Luminanz (Y- die Helligkeit) sowie Chrominanz (U und V-Farbinformation als Blau und Rot Differenz).

[2] INTRA-Mode bedeutet, daß das Bild nichts als Differenzbild, sondern als Originalbild übertragen wird.

[3] Diskrete Kosinus Transformation, invertierbare, diskret orthogonale Transformation.

[4] INTER-Mode ist im Gegensatz zum INTRA-Mode die Übertragung mit Differenzbildern.

- Es kann ein Bewegungsvektor übertragen werden (Motion Compensation), um dem Decoder mitzuteilen, daß sich der Makroblock verschoben hat.
- Die beiden letztgenannten Möglichkeiten sind kombinierbar: Der Makroblock kann mit Bewegungsvektor im INTER-Mode gesendet werden.
- Auf das Übertragen des Makroblocks kann verzichtet werden.

Der Encoder führt für jeden gesendeten INTRA- oder INTER-kodierten Block zusätzlich eine Dequantisierung und eine inverse DCT durch. Das so rekonstruierte Bild wird im Picture-Memory gespeichert. Für die Berechnung der Pixel-Differenz bei der Übertragung eines Makroblocks im INTER-Mode wird das rekonstruierte Bild herangezogen, um eine Aufsummierung der bei der Quantisierung zwangsläufig entstehenden Ungenauigkeiten zu vermeiden.

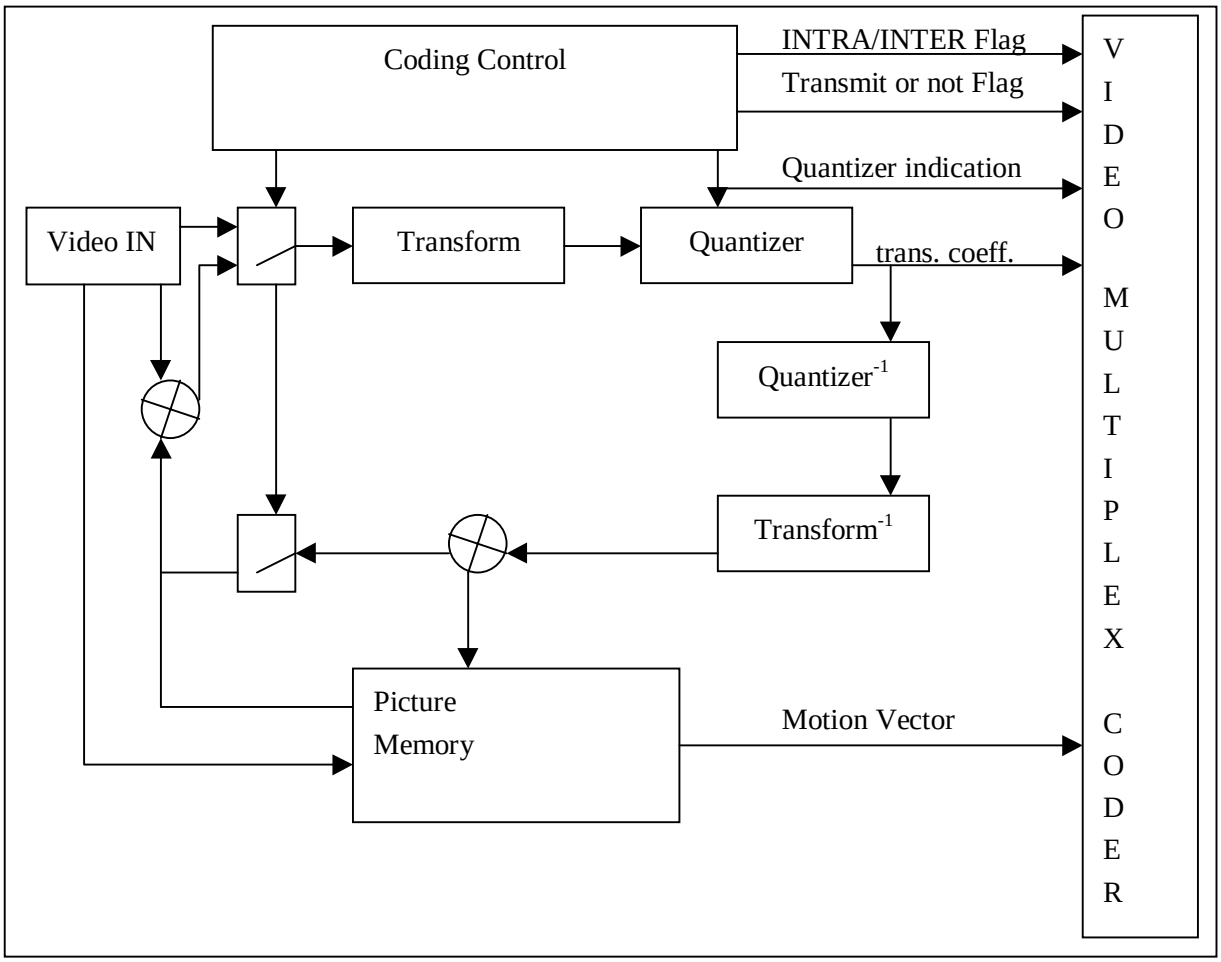


Abb 3-1: Die Coding Control

Abb. 3-1 zeigt den Ablauf schematisch. Der Video Multiplexer hingegen nimmt die Informationen auf und erstellt daraus einen Bitstream. Der Füllgrad des Transmission Buffers im Video Multiplexer dient als Anhalt, die Datenrate (und damit die Qualtität des Bildes) zu erhöhen oder zu vermindern. Auch könnte der Video Multiplexer eine Rückmeldung an die Coding Control bzgl. der verfügbaren Bandbreite liefern, welche dann die Coding Control veranlaßt die Qualität des Bildes zu verbessern oder zu verschlechtern.

3.1.1 Die Coding Control Einheit

Die Coding Control Einheit ist das Herzstück eines H.263 Encoders Sie entscheidet, ob ein Makroblock im INTER-Mode, im INTRA-Mode oder gar nicht übertragen wird. Sie entscheidet, wieviel Re-

chenzeit für die Bewegungsvektorsuche aufgewendet wird. Sie hat darüber zu wachen, daß die sonstigen Rahmenbedingungen, wie z.B. eine fest vorgegebene Bandbreite für den Bitstrom, auch eingehalten werden. Die Qualität des entstehenden Videobildes hängt damit offentsichtlich von der Güte der in der Coding Control Einheit verwendeten Algorithmen ab.

Die Empfehlung H.263 beschreibt präzise, wie der kodierte H.263-Bitstrom auszusehen hat und wie der Decoder diesen interpretieren muß. Letzteres trifft auch auf den Bildrekonstruktionsteil des Encoders zu. Darüber hinaus sagt die Empfehlung zum Vorgehen des Encoders und der Gestaltung des Coding Control wenig aus:

„Several parameters may be varied to control the rate of generation of coded video data. These includes processing prior to the source coder, the quantizer, block significance criterion and temporal subsampling. The proportions of such measures in the overall control strategy are not subject to recommendation“ [ITU 95, Kap.4.3]

Es werden lediglich einige Rahmenvorgaben gegeben. Beispielsweise soll der Encoder die Makroblöcke nicht ausschließlich im INTER-Mode senden, sondern ab und zu auch als INTRA-Block. Sinn dieser Vorgabe ist es, ein zu starkes Aufsummieren von Rundungsfehlern durch unterschiedliche iDCT-Implementierungen zu vermeiden.

3.2 Aufbau eines H.263 Bitstreams

Ein H.263 Bitstream besteht aus mehreren Schichten:

- Picture-Layer
- Group-of-Blocks Layer
- Macroblock-Layer
- Block-Layer

Nun der Aufbau im Einzelnen:

3.2.1 Picture-Layer

Der Picture-Layer ist auf Bytegrenzen ausgerichtet und besteht aus einem Header, den zum bildgehörenden Group-of-Blocks und gegebenfalls bis zu sieben Füllbits. Der Header beginnt mit dem Picture-Start-Code: 0000 0000 0000 1 00000. Danach folgen die Informationen, die zur Dekodierung des Bil-Bildes notwendig sind: Eine laufende Nummer, das Bildformat, die Angabe, welche der optionalen Erweiterungen verwendet werden, Quantisierungsfaktor etc.

Die laufende Nummer bezieht sich dabei auf alle Bilder der Bildsequenz, d.h. auch Bilder, die der Encoder nicht kodiert, werden mitgezählt. So kann der Decoder erkennen, ob und wie viele Bilder der Encoder weggelassen hat.

3.2.2 Group-of-Blocks

Eine Group-of-Blocks (GOB) faßt jeweils eine Zeile, bei 4CIF und bei 16CIF auch mehrere Zeilen, von Makroblöcken zusammen. Jede Group-of-Blocks besteht aus einem Header und den einzelnen Makroblöcken. Der Header wird bei der ersten GOB des Bildes weggelassen, bei den anderen GOBs ist er optional. Er kann, muß aber nicht, auf Bytegrenzen ausgerichtet sein. Wesentliche Bestandteile sind Startcode, GOB-Nummer und Quantisierungsfaktor. Innerhalb eines Bildes sind die GOBs von oben nach unten sortiert.

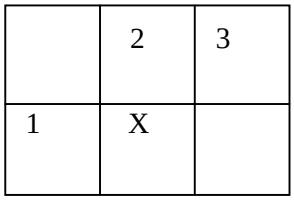
Die Start-Codes von Picture- und Group-of Block Layer sind so gewählt, daß sie zum Aufsynchronisieren auf den Datenstrom genutzt werden können. 16 aufeinanderfolgende Nullbits kommen nur in

diesen Startcodes vor. Sichergestellt wird das durch eine entsprechende Zusammenstellung der VLC-Tabellen (bei Huffman) bzw. durch Bit-Stuffin (bei Arithmetic Coding).

3.2.3 Makroblock

Ein Makroblock umfaßt einen 16x16 Pixel großen Bildausschnitt. Auch er enthält einen Header sowie die Elemente der untersten Schicht, die Blöcke. Ein Block ist 8x8 Pixel groß und ein Makroblock besteht aus sechs dieser Blöcke. Vier der Blöcke sind für die Speicherung der Luminanzsamples vorgesehen, die beiden anderen enthalten, mit der halben räumlichen Auflösung, die Chrominanzsmples, d.h. je einen Block für die Rot- und Blaudifferenz. Im Header des Makroblocks stehen die Informationen wie der Übertragungsmode (INTRA und INTER), der Bewegungsvektor, DQUANT (Differenz für Quantisierungsfaktor), welche der sechs Blöcke übertragen werden und einige weitere. Ein Teil wirdb für die Codierung der optionalen Erweiterungen benötigt und diese sind nur bei Verwendung der Erweiterungen auch vorhanden.

H.263 nutzt für die Motion Compensation Bewegungsvektoren mit einer Schrittweite von einem halben Pixel. Das zulässige Intervall [-16;15,5], sowohl für vertikale als auch für horizontale Vektoren. Es ist nicht erlaubt, Pixel außerhalb des Bildes zu referenzieren: Der Bildausschnitt, auf den ein Bewegungsvektor verweist, muß vollständig innerhalb des Bildes liegen.



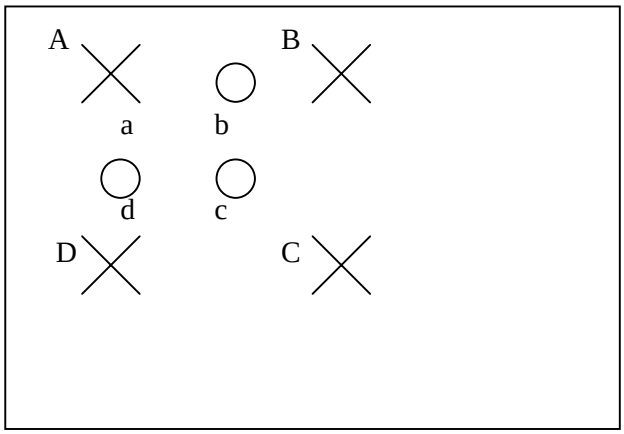
X aktueller Makroblock

1,2,3 für die Ermittlung des Prediction Vektors benutzte Makroblöcke

Abb 3-2: Ermittlung des Predictors für Bewegungsvektoren [ITU 95] Kap. 6.1.1

Die Bewegungsvektoren werden nicht direkt, sondern als Differenz zum Prediction Vektor kodiert. Der Prediction Vektor wird aus den Bewegungsvektoren von drei angrenzenden Makroblöcken (Abb 3-2) bestimmt, indem man - getrennt für vertikal und horizontal - den mittleren der drei Werte ermittelt. Dadurch ergeben sich im Mittel kleinere und besser komprimierte Werte als bei der direkten Kodierung.

Für die Chrominanzanteile werden die Bewegungsvektoren jeweils halbiert. Dabei eventuell entstehende Viertel-Pixel werden auf das nächstegelegende halbe Pixel gerundet, d.h. sowohl ¼ als auch ¾ wird zu ½.



Ganze Pixel

Halbe Pixel

a = A

b = (A+B+1) / 2

c = (A+B+C+D+2)/4

d = (A+D+1)/2

Abb 3-3: Bilineare Interpolation [ITU 95] Kap. 6.1.2

Werden Bewegunsvektoren mit halben Pixel verwendet, so werden die betroffenen Werte wie in Abb. 3-3 dargestellt bilinear interpoliert.

3.2.4 Blöcke

Die Blöcke enthalten die Bilddaten selbst. Die DCT-Transformation und die Quantisierung wird für jeden Block durchgeführt. Für die Quantisierung wird der im letzten Picture- oder GOB-Header angegebene Quantisierungsfaktor zuzüglich der im Makroblockheader angegebenen Änderungen (Dquant verwendet. Mit dem DQUANT-Wert kann der Quantisierungsfaktor bei jedem Makroblock um einen oder zwei Schritte nach oben oder unten korrigiert werden. Im Picture- oder GOB-Header kann der volle Wertebereich von 1 bis 31 direkt angegeben werden. Es wird stets der gleiche Quantisierungsfaktor für alle Koeffizienten verwendet, einzige Ausnahme ist der DC-Wert von INTRA-Blöcken: dieser wird immer mit 8 quantisiert.

Nach der Quantisierung werden die Koeffizienten des Blocks im Zickzack angeordnet und lossless komprimiert.

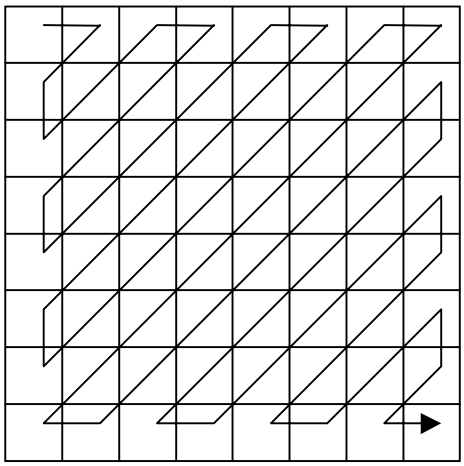


Abb. 3-4 Zickzack Anordnung der DCT-Werte

Da der weitaus häufigste Wert die Null ist, wird die Anzahl der Nullen mittels RLE kodiert. Die Anzahl der Nullen und der Wert des nächsten Koeffizienten werden zu einem Kodewort zusammengefaßt. Dieses Kodewort wird dann VLC kodiert. Die entsprechenden VLC-Tabellen sind in der Empfehlung H.263 festgeschrieben und können nicht verändert werden. Das gleiche trifft auch auf die VLC-Tabellen (für Makroblock-Typ, Bewegungsvektor, etc.) zu.

3.3 Video-Muiltiplex Coder

Der Video-Multiplex-Coder formt aus den einzelnen Makroblöcken einen hierarchischen Datenstrom. Diese Hierarchie ist in vier Ebenen geteilt (s. Abb. 3-5):

- Bild (picture)
- Blockgruppen group of blocks, GOB,
- Makroblock MB.
- Blöcke.

Das Bild beteht aus einem 20-Bit Bild-Start-Code, der Bildnummer, Informationen über den Bildstatus und den GOP-Daten. Der Bildstatus enthält Flags für aktives Screen-Splitting, für den Betrieb der Dokumentenkamera, das freeze picture release Kommando und das Videoformat (QCIF oder CIF).

Jedes Bild ist in Blockgruppen unterteilt. Dabei repräsentiert ein GOP eines CIF-Bildes oder eines QCIF-Bildes. Abb. 3-5 zeigt, wie die GOPs in einem Bild angeordnet sind. Der GOP besteht aus dem 16-Bit GOB-Start-Code, der Gruppennummer, der Quantisierungsschrittweite und den folgenden Macroblock-Daten.

Die Quantisierungsschrittweite gilt für alle Makroblöcke der Gruppe, wenn sie nicht durch eine Quantisierungsangabe im Makroblock überschrieben wird.

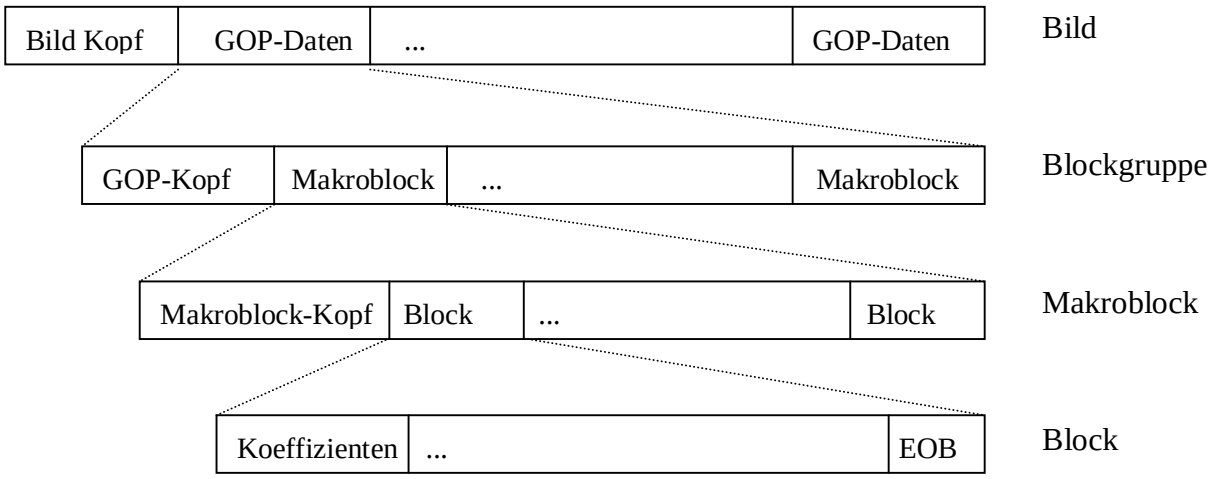


Abb. 3-5: Aufbau des Bitstreams

Ein GOP besteht aus 33 Makroblocks, die zeilenweise im GOP angeordnet sind. Makroblöcke enthalten neben den Koeffizienten der Luninanz- und Crominanzblöcke Informationen über den Blocktyp und ggf. eine blockeigene Quantisierungsschrittweite, den Bewegungsvektor und einen Bitvektor, der angibt, welche Blöcke des Makroblocks überhaupt übertragen werden. Makroblöcke, die keine Information beinhalten (bei denen also kein Block enthalten wäre), werden ausgelassen.

Die Zusammensetzung der Blöcke zu Makroblöcken wurde bereits in Abbildung auf Seite dargestellt. Jeder Block besteht aus den Fourier-Koeffizienten und einem Blockende-Kennzeichen (EOB). Die Fourier-Koeffizienten werden dabei mit aufsteigender Frequenz übertragen, um eine vorteilhafte Entropiekodierung zu erzielen.

Die Übertragung des Datenstroms wird in Gruppen zu je 492 Bit vorgenommen. Sollten keine zu übertragenden Daten anliegen, wird ein Füllblock versandt, den der Empfänger verwirft. Dieser Datenblock wird durch ein Synchronbit eingeleitet und mit einer 18-Bit langen BCH-Prüfsumme versehen. Diese Prüfsumme ermöglicht die Korrektur von Bitfehlern (forward error correction). Das Verwenden dieser Prüfsumme im Decoder ist optional.

Die Bildwechselfrequenz des Eingangssignals beträgt ca. 30 Bilder/s. Durch Auslassen von je Bildern kann die Bildwiederholfrequenz auch auf Werte von 15, 10 oder 7,5 Bildern/s eingestellt werden. Die Wahl der Bildwechselfrequenz nimmt der Benutzer vor.

3.4 Bewegungsvektorsuche

Um die Möglichkeit, die Motion Compensation bietet, auszunutzen, müssen die Bewegungsvektoren vom Encoder ermittelt werden. Die Suche nach den Bewegungsvektoren (Motion Estimation) ist sehr rechenintensiv.

Zur Ermittlung der Bewegungsvektoren sucht man mittels Blockvergleich nach Übereinstimmungen zwischen dem zuletzt gesendeten und dem zu kodierenden Bild. Als Kriterium für die Übereinstimmung wird meist die Durchschnittliche Pixeldifferenz (MAD-mean absolute distortion) benutzt.

Bei einer vollständigen Suche wird für jeden möglichen Bewegungsvektor der Blockvergleich durchgeführt. Bei einem Wertebereich von [-15; 16] muß die Berechnung immerhin 1024 mal pro Makroblock durchgeführt werden (ohne Berücksichtigung der Halbpixelbewegungsvektoren). Um den Rechenaufwand zu reduzieren, gibt es verschiedene Möglichkeiten:

Man braucht nicht für jeden möglichen Bewegungsvektor die durchschnittliche Pixeldifferenz. Gesucht wird der Bewegungsvektor, für den dieser Wert minimal ist. Daher kann man den Blockvergleich frühzeitig abbrechen, wenn der Minimalwert der bereits durchgeführten Vergleiche überschritten wurde.

Im Zusammenspiel mit dem obigen Punkt ist es sinnvoll, die Blockvergleiche so zu sortieren, daß die wahrscheinlicheren Fälle zuerst überprüft werden. Man könnte also in der Mitte des Suchraums anfangen und sich dann spiralförmig von innen nach außen bewegen.

Die Suche kann in 2 Phasen aufgeteilt werden, indem man zuerst eine Suche mit einem groben Raster durchführt und anschließend um das gefundene Minimum herum genau sucht. Es besteht dabei jedoch die Gefahr, daß man bei der Grobsuche ein lokales Minimum findet und dann die Feinsuche an der falschen Stelle ansetzt.

Man kann beim Blockvergleich statt aller Pixel nur einige ausgewählte Pixel eines Makroblocks in die Berechnung der durchschnittlichen Pixeldifferenz einbeziehen. Auch ist es möglich, daß der optimale Bewegungsvektor verfehlt wird. Die Ggefahr ist um so größer, je weniger Pixel für den Vergleich herangezogen werden.

Die beiden ersten Verfahren haben keinen Einfluß auf den gefundenen Bewegungsvektor, da sie nur unnötige Rechenoperationen vermeiden. Die beiden letzten Verfahren können die Wahl des Bewegungsvektors beeinflussen. Bewegungsvektoren sind gut für Kameraschwenks und linear bewegten Objekten und leider schlecht für Zoom und rotierende Objekte.

3.5 Performance Verbesserung von H.261 nach H.263

Gegenüber H.261 hat H.263 eine Performance Verbesserung erhalten, so daß man davon ausgehen kann, daß das Einsatzgebiet von H.263 nicht nur bei niedrigen Bitraten liegt, sondern auch bei höherer Übertragungskapazitäten H.261 ablösen wird.

3.5.1 Uneingeschränkte Bewegungsvektoren (unrestricted motion vectors)

In diesem Modus dürfen die Bewegungsvektoren aus dem Bild heraus zeigen. Die Randpixel werden als Prädiktion für „nicht existente„ Pixel benutzt. Mit diesem Modus wird eine signifikante Verbesserung erzielt, falls eine Bewegung entlang des Randes eines Bildes auftritt, besonders bei kleinen Bildformaten. Zusätzlich setzt dieser Modu einen größeren Bereich für die Vektor-Suche fest, so daß größere Motion Vektoren eingesetzt werden können. Das ist besonders nützlich bei bewegter Kameraführung.

3.5.2 Erweiterte Prädiktion (advanced prediction)

Diese Option erlaubt eine überlappende Bewegungskompensation bei P-Bildern. Vier 8x8 Blöcke anstatt eines 16x16 Blocks werden für einige der Makroblocks benutzt und Bewegungsvektoren dürfen (wie unter 1.) außerhalb des Bildes zeigen. Der Encoder muß entscheiden, welcher Typ von Vektor zu wählen ist. Vier Vektoren verbrauchen mehr Bits, ergeben aber eine bessere Prädiktion. Das führt zu weniger Block-Artefakten.

3.5.3 Syntax-basierte Arithmetische Kodierung (syntax-based Arithmetic coding)

Als weiterer Vertreter der Entropie-Kodierung wird anstatt Huffman die Arithmetische Kodierung verwendet, welche im Mittel weniger Bits liefert.

3.5.4 PB-Bilder (PB-frames)

Ein PB-Bild besteht aus zwei Bildern, die als eine Einheit kodiert werden.. Der Name entstammt den in MPEG benutzten P(redicted)- und B(idirectional)-Pictures. Ein PB-Bild besteht aus einem P-Bild, das durch Prädiktion aus dem vorhergegangenen P-Bild entstand und einem B-Bild aus dem vorhergegangenem und dem gerade kodierten P-Bild. Für einfache Sequenzen kann der PB-Mode die Bildrate verdoppeln, wobe die Bitrate nicht sehr erhöht wird. Die Güte des PB-Modus ist gegenüber den MPEG B-Bildern zwar unterlegen, er produziert aber weniger Overhead und bietet sich daher für niederratige Verbindungen, wie Bildtelefon, an.

4 Aufbau einer Videokonferenzverbindung

Bei einer Videokonferenzverbindung gibt es verschiedene Szenerien, die zu berücksichtigen sind. Wie schon in der Einführung erläutert unterscheiden sich die Protokolle anhand der Übertragungskapazität.

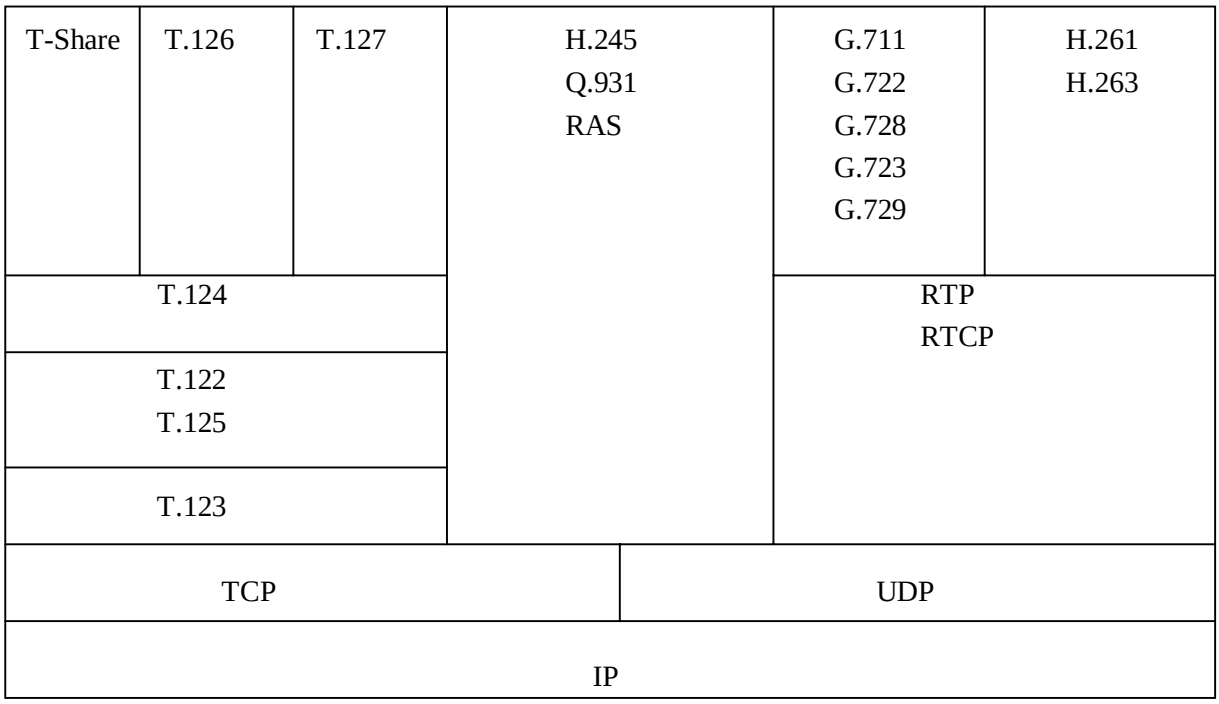


Abb. 4-1 Aufbau einer Videokonferenzverbindung [CT 9/99]

Aber nicht nur die Übertragungskapazität ist ein Unterscheidungskriterium, sondern auch die Daten, welche übertragen werden stellen ein Kriterium da. Die Abb. 4-1 verdeutlicht, daß nebenher der verschiedenen Datenarten und Steuerungsprotokollen. In den weiteren Abschnitten wird auf die einzelnen Protokolle näher eingegangen.

4.1 ISDN

H.320 ist der Rahmenstandard für ISDN-basierte Systeme. ISDN (Integrated Services Digital Network) ist ein digitales, verbindungsorientiertes Telekommunikationsnetz. Ein ISDN-Basisanschluß stellt zwei Datenkanäle (B-Kanäle) mit je 64 kbit/s Bandbreite sowie einen Steuerkanal (D-Kanal) zur Verfügung. H.320 ist die älteste der Rahmenempfehlungen, die erste Revision stammt aus dem Jahre 1990 und beschreibt den prinzipiellen Aufbau eines ISDN-Bildtelefons. Die aktuell verabschiedete Fassung stammt vom März 1996. Als Videokodierung ist H.261 festgelegt, für Audio kommt G.711 (Pflicht) oder G.722 bzw. G.728 (optional) zum Einsatz. Modernere Kodierungen wie H.263, G.729 und G.723.1 werden demnächst integriert, sie sind im aktuellen Entwurf bereits enthalten.

H.221 beschreibt einen Multiplexmechanismus für die verschiedenen Datenarten (Audio, Video, Daten) für ISDN-Verbindungen. H.221 kann mehrere ISDN-B-Kanäle synchronisieren und zu einer logischen Verbindung zusammenfassen. Die zur Verfügung stehende Übertragungskapazität wird in Unterkanäle mit je 8 kbit/s Bandbreite unterteilt. Diese Unterkanäle werden gebündelt und den verschiedenen Datenarten zugeteilt. Jeder Datenart steht ein gebündelter Datenkanal zur Verfügung. Die Zuordnung der Unterkanäle (und damit die Bandbreite der gebündelten Datenkanäle) kann dynamisch geändert werden.

H.242 beschreibt die Verbindungsauf- und -abbauprozedur für Punkt-zu-Punkt Verbindungen. Dazu gehören der Austausch der Fähigkeiten der Telefone und die Vereinbarung der Betriebsparameter, etwa welcher Audiocodec verwendet werden soll. Die Betriebsparameter können auch während der Verbindung geändert werden. Der Verbindungsauf- und -abbau für Mehrpunktverbinungen ist analog in H.243 festgehalten; die Spezifikation der zugehörigen Zentrale - der Multipoint Control Unit (MCU) - steht in der Empfehlung H.231. Eine MCU ist aufgrund des verbindungsorientierten Charakters des ISDN-Netzes notwendig. Sie empfängt die Daten der teilnehmenden Endgeräte und hat die Aufgabe, diese gegebenenfalls zu bearbeiten (beispielsweise Audio mischen, ein Videobild auswählen) und an die anderen Teilnehmer weiterzuleiten.

4.2 GSTN

H.324 ist die Rahmenempfehlung für GSTN (General Switched Telephone Network). Als GSTNs werden verbindungsorientierte Telefonnetze allgemein bezeichnet. Das Versenden der digitalen Daten erfolgt mit Hilfe von Modems. H.324 ist wesentlich jünger als die äquivalente ISDN-Empfehlung. Erst seit kurzer Zeit lassen die Kompressionstechniken für Audio und Video Desktop Conferencing über Modemverbindungen überhaupt zu. Für die Audiokomprimierung kommt G.723.1 zum Einsatz, für Video H.261 oder H.263. Auf der Modemleitung kommt V.34 (bis zu 28.8 kbit/s) zum Einsatz. Die Steuerung (Verbindungsauf- und -abbau, Austausch der Fähigkeiten etc.) ist in der Empfehlung H.245 spezifiert. H.245 stellt auch logische Kanäle für die verschiedenen Datenarten zur Verfügung. Das Multiplexing dieser Kanäle über die Modemverbindung übernimmt H.223. Mehrpunktkommunikation unter Zuhilfenahme einer MCU ist ebenfalls vorgesehen.

4.3 LAN

Die jüngste, erst im November 1996 verabschiedete Rahmenempfehlung ist H.323. Darin sind die entsprechenden Spezifikationen für LANs enthalten. Unter einem LAN (Local Area Network) versteht man ein lokales Computernetz. Ein LAN arbeitet im Gegensatz zu ISDN und GSTN nicht verbindungs- sondern paketorientiert. Es steht wesentlich mehr Bandbreite zur Verfügung, die jedoch nicht exklusiv genutzt werden kann. Bei den klassischen und auch heute noch stark verbreiteten Systemen Ethernet (auch Fast Ethernet) und Token Ring gibt es weder Mechanismen zur Bandbreitenreservierung noch Garantien dafür, daß ein Paket innerhalb einer bestimmten Zeit ankommt. Pakete können auch ganz verloren gehen. ISDN und GSTN dagegen bieten exklusiv nutzbare Bandbreite mit fester Latenzzeit.

H.323 ist weitaus komplexer und umfangreicher als H.320; neben den LAN-basierten Desktop Conferencing Systemen selbst werden auch Gateways zu den anderen Übertragungsmedien betrachtet. So sollen mit einer als Gateway agierenden MCU auch gemischte LAN/ISDN/GSTN Konferenzen möglich sein. Bei reinen LAN-Konferenzen ist eine MCU nicht zwingend notwendig.

Die Daten können auch mit Multicast verschickt werden, wenn das darunterliegende LAN das erlaubt. Die Entwicklung hier steckt jedoch noch in den Kinderschuhen.

Die Audiokodierung G.711 muß unterstützt werden, optional können auch G.722, G.728, G.729, G.723.1 und MPEG-1-Audio verwendet werden. Videounterstützung ist optional. Wenn das Endgerät Video beherrscht, muß H.261 und kann H.263 unterstützt werden.

Für die Steuerung wird auch hier die Empfehlung H.245 verwendet. Wie der Transport der Daten erfolgt, ist in H.225 spezifiziert.H.225 wiederum stützt sich auf die Internet-Transportprotokolle, insbesondere RTP[RTP 96]. Die Nutzung von IP Multicast ist ebenfalls vorgesehen.

5 Tonkodierungen

G.711 ist die PCM-Audiokodierung für normale ISDN-Telefone. Es wird die volle Bandbreite eines ISDN B-Kanals (64 kbit/s bzw. 56 kbit/s in Amerika) benutzt. G.711 muß von allen Geräten beherrscht werden, damit die Geräte interoperabel sind.

Darüber hinaus gibt es weitere (optionale) Audiokomprimierungen, die geringere Anforderungen an die Bandbreite stellen bzw. bessere Tonqualität bieten.

G.722 ist eine auf ADPCM basierende Audiokodierung. G.722 kann mit Bandbreiten zwischen 64 kbit/s und 22 kbit/s arbeiten. Bei Verwendung 64 kbit/s erreicht man eine Audiobandbreite von 100 Hz bis 7 kHz; das ist mehr als mit G.711 (ca. 100 Hz bis 3,5 kHz) möglich ist.

G.728 benötigt nur 16 kbit/s Bandbreite und ist bei ISDN-basierten Systemen de-facto Standard, weil damit die Übertragung von Ton und Bewegtbild über einen B-Kanal überhaupt erst möglich wird und auch bei zwei B-Kanälen wesentlich mehr Bandbreite für das Bewegtbild zur Verfügung steht. G.728 ist eine Sprachkodierung, andere Daten - etwas Musik - lassen sich nur sehr schlecht oder gar nicht kodieren.

Noch bessere Kompressionsraten lassen sich mit G.729 (8 kbit/s) und G.723.1 (5.3 kbit/s und 6.3 kbit/s) erreichen. Auch diese Kodierungen sind wie G.728 Sprachkodierungen.

6 T.120

Für Übertragung von Nutzdaten werden in der Regel die Empfehlungen der T.120-Serie verwendet. Es gibt noch weitere Standards zur Übertragung von Nutzdaten, auf die aber wegen der geringen praktischen Bedeutung nicht weiter eingegangen wird.

Die Empfehlung T.120 selbst übernimmt die Rolle des Rahmenstandards, der die allgemeinen Zusammenhänge erklärt.

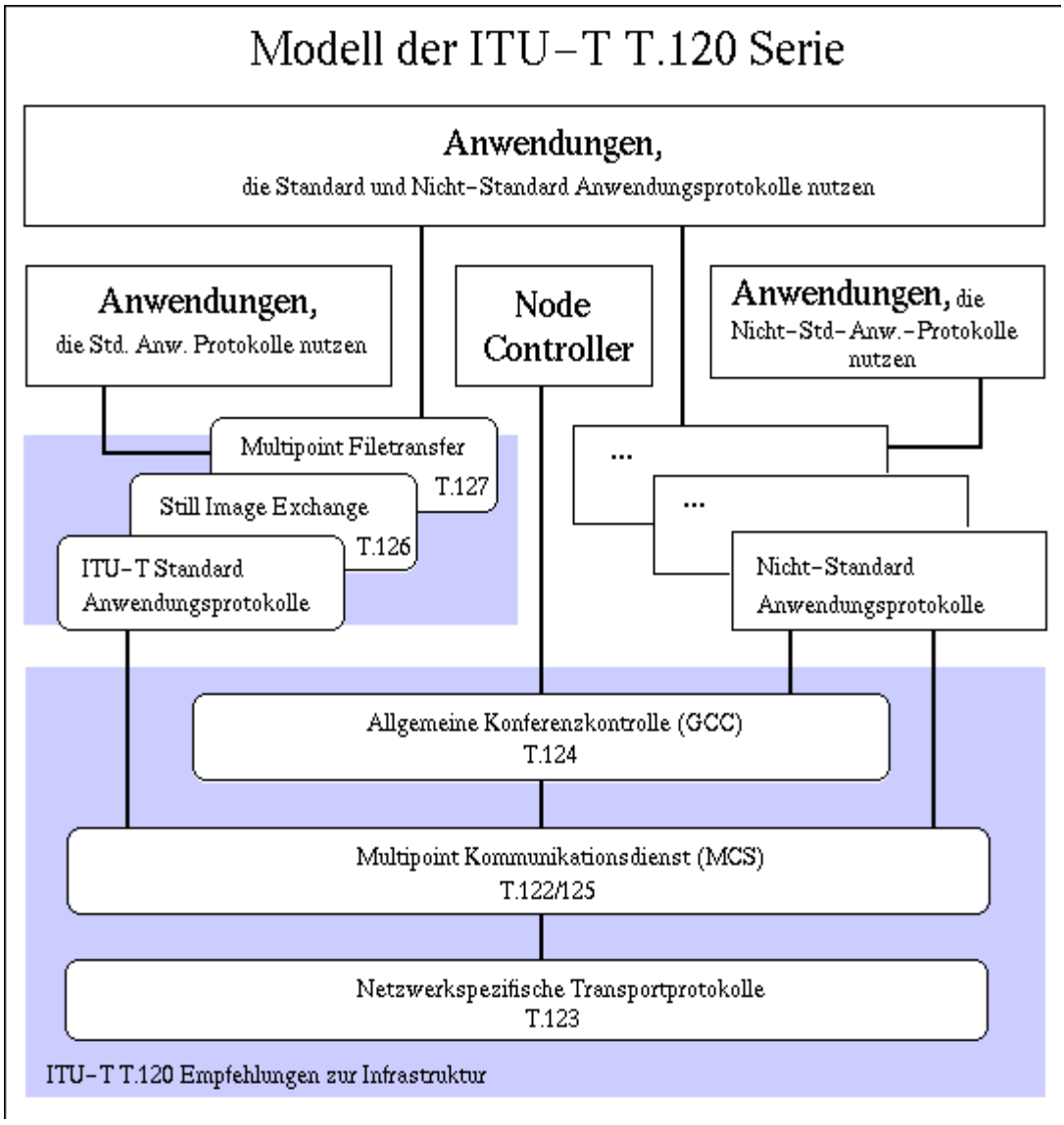


Abb. 6-1: ITU T.120 Empfehlung zur Infrastruktur

T.123 spezifiziert für die verschiedenen Netze, welche Protokolle für Datenübertragung und Fehlersicherung benutzt werden. Je nach Art und Güte des Netzes sehen diese sehr unterschiedlich aus, nach oben wird jedoch eine einheitliche, netzunabhänige Dienstschnittstelle zur Verfügung gestellt.

In T.122 und T.125 ist der Multipoint Communication Service (MCS) spezifiziert. Er benutzt die von T.123 bereitgestellten Punkt-zu-Punkt-Verbindungen, um darauf aufbauend Mechanismen zur Mehrpunkt-Datenkommunikation bereitzustellen. Die einzelnen Teilnehmergeräte werden dabei baumartig vernetzt.

Der MCS stellt mehrpunktfähige Kommunikationskanäle (Channels genannt) zur Verfügung. Die Daten können (müssen aber nicht) als sequenced Data verschickt werden. Als sequenced Data gesendete Daten werden immer über die Wurzel des Kommunikationsbaumes verschickt, um sicherzustellen, daß die Reihenfolge des Eintreffens bei allen Teilnehmern gleich ist. Der MCS stellt auch Token zur Verfügung, mit deren Hilfe die auf dem MCS aufbauenden Anwendungen

beispielsweise Zugriffskontrollmechanismen implementieren können. Für den MCS selbst haben die Channels und Tokens keinerlei Bedeutung, er stellt nur die Basismechanismen zur Verfügung.

Die vom MCS bereitgestellten Dienste nutzen dann die Anwendungen und die Generic Conference Control (GCC). Die GCC ist in der Empfehlung T.124 spezifiziert. Die GCC ist unter anderem für den Auf- und Abbau der (Daten-) Konferenz, Konfigurationsänderungen und den Austausch der Fähigkeiten der Teilnehmergeräte (bezüglich der T.120 Anwendungen) zuständig.

An Anwendungsprotokollen ist bisher spezifiziert:

- T.126 / T.SI

  Still Image Transfer and Annotation - Protokoll zum Austausch von grafischen Informationen. Whiteboard-Funktionalität ist damit beispielsweise realisierbar.

- T.127 / T.MBTF

  Multipoint Binary File Transfer - Protokoll zum Austausch von Binärdaten, beispielsweise von Dokumenten.

- T.128 / T.Share

  Application Sharing - Protokoll für die gemeinsame Benutzung von Programmen. Die Realisierung erfolgt über das Umleiten der Bildschirmein- und -ausgaben, vergleichbar mit der Funktionsweise von Remote Control Software. Das betreffende Programm läuft auf einem der an der Konferenz beteiligten Computer, die Bildschirmausgaben können an die anderen an der Konferenz teilnehmenden Rechner weitergeleitet werden.

Darüber hinaus können auch andere, nichtstandardisierte Anwendungen auf die Dienste des MCS zurückgreifen. In T.121 (Generic Application Template) ist allgemein beschrieben, wie die Kommunikation zwischen zwei Anwendungen ablaufen soll.

Allerdings funktionieren auf diesem Weg implementierte Features üblicherweise nur zwischen Teilnehmergeräten der gleichen Firma.

7 Ausblick H.263+

Im Zuge der technischen Entwicklung finden Neuerungen nach und nach Eingang in die Standards. Zur Zeit befindet sich neben vielen anderen Desktop Conferencing Standards der unter dem Arbeitstitel H.263+ bekannte Nachfolger der aktuellen H.263-Revision in der Entwicklung. Die nachfolgenden Auszüge stammen aus dem Draft 9 [GR 9/97] für H.263+. Die Verabschiedung der neuen H.263 Revision war noch im letzten Jahr vorgesehen.

Im Baseline sind wenige Veränderungen vorgenommen worden; nahezu alle Neuerungen sind als optionale Erweiterungen in den neu hinzugekommenen Annexen I-T zu finden. Im Baseline neu eingeführt wurde lediglich die Möglichkeit, neben den in Abb. 2-1 aufgeführten Bildgrößen auch eigene Formate zu benutzen. Daneben wurde wegen der zahlreichen neuen Erweiterungen die Einführung eines neuen Feldes im Picture Header notwendig, um die Benutzung oder Nichbenutzung der Erweiterungen signalisieren zu können.

Einige der Neuerungen stehen im Zusammenhang mit der zunehmenden Bedeutung von paketorientierten Netzen für die Übertragung von Bewegtbildern. Damit sollen H.263+ Encoder und Decoder die typischen Problemen paketorientierter Netze - Pakete können verlorengehen, die Sendereihenfolge muß nicht mit der Empfangsreihenfolge übereinstimmen - besser bewältigen.

Im Advanced Intra Coding Mode (Annex I) können INTRA-Blöcke platzsparender kodiert werden. Er führt Prediction auch für INTRA-Blocks ein. Dafür wird jeweils der linke bzw. obere Makroblock des gleichen Bildes herangezogen. Es wird eine spezielle VLC-Tabelle für INTRA-Blöcke verwendet.

Der Slice Structed Mode (Annex K) ersetzt die Group-of-Blocks durch Slices. Slices können rechteckige Bildausschnitte oder eine bestimme Zahl aufeinanderfolgender Makroblöcke umfassen. Die Slices sind relativ unabhänig voneinander; es werden keine Daten aus anderen Slices desselben Bildes für Predicion herangezogen. Das erlaubt die Dekodierung der Slices eines Bildes in einer beliebigen Reihenfolge. Es ist sinnvoll, die Größe der Slices passend zur Paketgröße des verwendeten LANs zu wählen.

Der Independently Segmented Decoding Mode (Annex R) geht noch einen Schritt weiter. Slicegrenzen (oder GOB-Grenzen) werden wie Bildgrenzen behandelt. Das heißt insbesondere, daß Bewegungsvektoren nicht aus Slices herauszeigen dürfen. Wird gleichzeitig der Unrestricted Motion Vector Mode verwendet, dürfen die Bewegungsvektoren zwar herauszeigen, jedoch werden auch hier Slicegrenzen wie Bildgrenzen behandelt. Es werden also die Randpixel des Slices und nicht die Pixel des benachbarten Slices herangezogen. Damit haben verlorengegangene Pakete keine negativen Auswirkungen auf benachbarte Slices.

Im Reference Picture Selection Mode (Annex N) wird nicht das letzte gesendete Bild als Referenzbild für Inter-Picture-Prediction verwendet, sondern der Encoder kann dem Decoder mitteilen, welches Bild er als Referenzbild nutzen soll. Der Decoder muß dafür mehrere decodierte Bilder speichern. Optional kann ein Rückkanal verwendet werden, über den der Decoder den Empfang der Bilder positiv (ACK) und/oder negativ (NACK) bestätigt. Damit kann man man vermeiden, im Falle von Paketverlusten das Bild oder den betroffenen Bildausschnitt komplett neu INTRA-kodiert senden zu müssen.

Der Temporal, SNR, and Spatial Scalability Mode (Annex O) erlaubt die Kodierung von unterschiedlichen Qualitätsstufen in einem Datenstrom. Die Bilddaten werden dabei in Basisdaten und darauf aufbauende Zusatzdaten eingeteilt. Sinnvoll ist das zum Beispiel bei Konferenzen, die über zwei oder mehr verschiedene Netze mit unterschiedlichen Bandbreiten geführt werden. Auf dem breitbandigeren Netz wird mit hoher Qualität gearbeitet, für das schmalbandigere Netz können die Zusatzinformationen einfach weggelassen werden, ohne daß dafür Umkodierungen (die zusätzliche Latenzzeiten verursachen würden) notwendig sind. Denkbar ist auch, Fehlersicherung und Bandbreitenreservierungen nur für die Basisdaten vorzunehmen. Die Zusatzdaten können sowohl die zeitliche als auch die räumliche Auflösung verbessern. Die Verbesserung der zeitlichen Auflösung erreicht man mit B-Frames, die im Gegensatz zum PB-Frames Mode nicht verschachtelt, sondern komplett extra kodiert werden. Zur Verbesserung der räumlichen Redundanz werden feiner quantisierte Differenzinformationen zum Basisbild kodiert. Die Differenzen können auch in einem größeren Bildformat gesendet werden, in diesem Fall wird das Basisbild entsprechend interpoliert. Es sind auch mehrere Schichten Zusatzinformationen zulässig.

Der Deblocking Filter Mode (Annex J) ist eine Alternative zum Advanced Prediction Mode. Es kommen bis auf Overlapped Block Motion Compensation die gleichen Mechanismen zum Einsatz wie im Advanced Prediction Mode. Statt OBMC kommt ein Filter für die Blockgrenzen zum Einsatz. Der gewünschte Effekt - Verwischen der Blockgrenzen - ist der gleiche, allerdings ist das Verfahren weniger aufwendig.

Der Reference Picture Resampling Mode (Annex P) erlaubt die Übertragung von vier Vektoren pro Bild. Diese vier Vektoren geben die neue Lage der Bildecken an, das Referenzbild im Decoder wird vor der der Benutzung für Inter-Picture-Prediction passend transformiert. Bildglobale Bewegungen, verursacht durch Kamerabewegungen und/oder -zoom, lassen sich damit ausgezeichnet kodieren. Darüber hinaus kann man diese Erweiterung auch für diverse Spezialeffekte benutzen. Die Transformation ist allerdings sehr rechenaufwendig.

Der Reduced-Resolution Update Mode (Annex Q) erlaubt es, bei bewegten Szenen die Bilder vorübergehend mit geringerer Auflösung zu übermitteln. Es wird ein Makroblock pro 32x32-Bildausschnitt gesendet. Der übertragene Makroblock ist nach wie vor nur 16x16 Pixel groß, das Bild wird entsprechend hochinterpoliert.

Mit der Supplemental Enhancement Information Specification (Annex L) können Zusatzinformationen für den Decoder in den H.263-Bitstrom kodiert werden, beispielsweise daß der Decoder das Bild oder einen Bildausschnitt unverändert stehen lassen soll (Picture Freeze).

Der Improved PB-Frames Mode (Annex M) ist eine Verbesserung des in Annex G spezifizierten PB-Frame Mode. Für Makroblöcke des B-Frames ist zusätzlich forward Prediction erlaubt, Annex G verwendet immer bidirectional Prediction.

Im Alternative Inter VLC Mode (Annex S) kann eine andere VLC-Tabelle für die Kodierung von INTER-Blöcken herangezogen werden.

Der Modified Quantization Mode (Annex T) ändert die Semantik des DQUANT-Feldes im Makroblock-Header. Die Schrittweite für die Änderung des Quantisierungsfaktors hängt vom Wert des Quantisierungefaktors ab, bei großen Quantisierungsfaktoren ist auch die Schrittweite groß. Darüber hinaus ist es auch möglich, jeden möglichen Quantisierungsfaktor direkt im DQUANT-Feld anzugeben. Die Chroma-Blöcke werden mit einem kleineren Quantisierungsfaktor quantisiert, was zu einer besseren Bildqualität bei großen Quantisierungsfaktoren führt.

8 Literatur

[CT 4/97] Harald Bögeholz, Andreas Stiller:
„Durchs Fenster betrachtet“
CT Magazin für Computer Technik, Heise Verlag, p. 238 - 244, 4/1997

[CT 9/99] Jürgen Kuri:
„Nummernspiele“
CT Magazin für Computer Technik, Heise Verlag, p. 180 - 191, 9/1999

[GR 9/97] Gary Sullivan:
Rapporteur: Draft Text of Recommendation H.263 version 2 ("H.263+") for decision
http://www.itu.int//itudoc/itu-t/com16/contr/026.html [Stand: 18:4. 1999]

[ITU 99] International Telecommunication Union
http://www.itu.int [Stand: 10.5.1999]

[ITU 95] ITU-T Draft H.263, Study Group 15
ITU-T Recommendation H.263
http://www.nta.no/brukere/DVC/h263_wht/h263wht.htm [Stand: 9.4.1999]

[RTP 96] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson:
„RTP: A Transport Protocol for Real-time“
Applications (RFC 1889), 1996

[WERS 99] Wolfgang Effelsberg, Ralf Steinmetz
„Video compression techniques“

Vergleichende experimentelle Betrachtungen

Paulus Neubach und Helge Schneider

Fachbereich Informatik Universität Dortmund

Zusammenfassung

Die vorliegende Ausarbeitung bezieht sich auf den experimentellen Vergleich von ausgewählten Kompressionsverfahren und beruht, was Teil I anbelangt, im wesentlichen auf Kapitel 6 des Buches „Video Compression Techniques“ von *Wolfgang Effelsberg und Ralf Steinmetz*.

In Teil I der Ausarbeitung wird im Besonderen auf die Messung und den Vergleich von Qualität und Performance, bezogen auf Still Images und Videoclips eingegangen. Dabei werden die Kompressionsverfahren JPEG, MPEG und H.261 genauer betrachtet. Die genannten Verfahren werden als Bekannt vorausgesetzt..

Teil II der Ausarbeitung bezieht sich auf ausgewählte Anwendungsbeispiele für Kompressionsverfahren. Hierbei wird zunächst der DVD (Digital Versatile Disc) Standard vorgestellt. Anschließend werden die einzelnen Schritte bei der Erstellung eines MPEG-1 und MPEG-2 Videoclips mit einem rudimentären Vergleich dargelegt.

Kompressionsalgorithmen für Still Images und Videoclips werden von einer großen Anzahl von Parametern beeinflußt. Je nach verwendeter Applikation besteht für den Anwender die Möglichkeit, diese Parameter mehr oder weniger stark zu beeinflussen. Die Einstellung der Kompressionsparameter hat aber gerade eine große Auswirkungen auf die Kompressions- und Dekompressionszeit und auf die Qualität. Leider gibt es aber keine generellen Regeln, nach denen ein Anwender diese Parameter einstellen kann. Die optimalen Einstellung der Parameter hängt vom Material ab, das zu bearbeiten ist. Aus diesem Grund soll in dieser Ausarbeitung auch der Einfluß verschiedener Parameter auf die Qualität und Performance vergleichend dargestellt werden.

I. Experimenteller Vergleich von ausgewählten Kompressionsverfahren bezogen auf Qualität und Performance

1 Messung von Qualität und Performance

Um Möglichkeiten der Messung von Qualität und Performance darzustellen, sollte ersteinmal klar sein, was unter Qualität und Performance zu verstehen ist. Unter Qualität soll im Folgenden der Unterschied zwischen Originalmaterial, und mittels Kompressionsverfahren bearbeiteten Material verstanden werden. Bei Bezügen zur Performance soll der zeitliche Aspekt der Kompression und Dekompression der Verfahren gemeint sein.

Für die Messung von Qualität existieren bisher keine geeigneten, objektiven Methoden. Die Bestimmung von zeitlicher Performance ist viel einfacher. Hier können Kompressions- und Dekompressionszeit physikalisch gemessen werden. Qualität hingegen ist aber in hohem Maße subjektiv und hängt von Auge des Betrachters und dessen Schulung ab. Das menschliche Auge ist ein komplexes System und die Instrumentalisierung des Sehens und Erfahrens (mit Einfluß von Erfahrungswerten) bisher nicht durchführbar. Aus diesem Grund wird versucht eine objektive Messung von Qualität zu approximieren.

1.1 Still Images: Messung von Qualität

Im Folgenden wird auf die Messung von Qualität bei Still Images, also einfache Bilder, die mit einer verlustbehafteten Kompression bearbeitet werden (JPEG) eingegangen. Hierbei wird im wesentlichen auf eine automatisierbare Methode (SNR) eingegangen, da eine manuelle Methode zu fehleranfällig und zu kostspielig ist.

1.1.1 Manuelle Methode

Eine Möglichkeit, die Qualität von Still Images oder auch Videoclips zu messen, besteht darin, diese einer Testgruppe von Menschen zu zeigen. Die subjektiven Eindrücke dieser Testgruppe müssen hierbei mittels Fragebogen festgehalten werden. Durch Skalierung der einzelnen Fragen können die Fragebögen dann statistisch ausgewertet werden. So lassen sich dann Aussagen über das Referenzmaterial machen. Skalierung bedeutet hierbei, daß die Testpersonen eine Eigenschaft des Testmaterials auf einer Skala markieren sollen. Beispielsweise könnte eine mögliche Skala zur Frage „Wie würden Sie die Struktur des Bildes beurteilen?“ folgendermaßen aussehen: 1. Sehr gut zu erkennen, 2. gut zu erkennen, 3. geht so, 4. schlecht zu erkennen, 5. sehr schlecht zu erkennen.

Problematisch ist hierbei die Erstellung der Fragebögen, aber auch eine Fehlinterpretation der informellen Fragen durch die Testpersonen. Dies verzerrt die erarbeiteten Ergebnisse und führt zu falschen Aussagen. Desweiteren ist die Auswertung zu zeit- und damit auch kostenintensiv, weshalb auf eine automatische Methode zurückgegriffen werden sollte.

1.1.2 Automatische Methode

Um die Qualität der Bilder automatisch zu messen, wurde von *Effelsberg und Steinmetz* zu jedem komprimierten Bild eine Signal-to-Noise Ratio (SNR) berechnet. Dieser Wert wird in der analogen

Elektronik verwendet, um die Qualität einer Signalverarbeitung zu beschreiben. In der Praxis wird ein Signal durch eine Verarbeitung (fast) immer verfälscht, sodass Eingangs- und Ausgangssignal nicht übereinstimmen. Der Unterschied der beiden Signale wird als Rauschen bezeichnet.

Für die folgenden Experimente wurde der Begriff Rauschen auf die Bildverarbeitung übertragen und beschreibt nun den Unterschied des originalen Bildes B_{in} zu dem komprimierten und wieder dekomprimierten Bild B_{out}.

Für ein Graustufenbild ist der SNR-Wert wie folgt definiert:

Signal-to-Noise Ratio (SNR) :

$$SNR = 10 \log_{10} \left(\frac{\sum_{j=1}^{M} \sum_{k=1}^{M} \left(B_{in}(j,k)\right)^2}{\sum_{j=1}^{M} \sum_{k=1}^{M} \left(B_{in}(j,k) - B_{out}(j,k)\right)^2} \right)$$

$B_{in}(j,k)$: Y-Komponente des Pixel (j,k) im Bild B_{in}.

$B_{out}(j,k)$: Y-Komponente des Pixel (j,k) im Bild B_{out}.

Das Verhältnis von Signal und Rauschen wird in der Formel durch den Bruch dargestellt. Dabei steht im Nenner des Bruches das Rauschen im Bild B_{out}. Damit sich positive und negative Unterschiede zwischen den Bildern nicht ausgleichen und zu Null summieren, wird jede Differenz quadriert. Das Signal B_{in} steht im Zähler und wird auch quadriert, um die Wurzel im Nenner zu sparen. Eigentlich wird so ein SNR^2 berechnet, was aber die Unterschiede zwischen den Bildern nur deutlicher macht.

Um den SNR-Wert in Dezibel anzugeben, wird wegen

$$P_{[db]}=10\log_{10}(P)$$

die Ergänzung vorgenommen. Ein kleiner SNR-Wert läßt also auf einen großen Unterschied zwischen B_{in} und B_{out} schließen.

1.2 Still Images: Messung von Performance

Um die Leistung einer Komprimierung zu bewerten, sind folgende Werte zu beachten:

Kompressionsrate	R	= Dateigröße(B_{out})/ Dateigröße(B_{in})
Kompressionszeit	t_{comp}	= Zeit fürs Komprimieren
Dekompressionszeit	t_{decomp}	= Zeit fürs Dekomprimieren

Die Komprimierungs- und Dekomprimierungszeiten wurden mit Hilfe eines Profilingtool ermittelt. Beim Profiling wurden die Maschineninstruktionen während der Ausführung der Komprimierungs- und Dekomprimierungsroutine ermittelt, und diese Zahl durch die Taktfrequenz der CPU dividiert. Dies führt zu einem CPU abhängigem Wert für den Rechner, der in den Experimenten benutzt wurde.

Bei den Kompressionsraten ist zu beachten, daß ein kleiner Wert für R eine starke Komprimierung beschreibt.

1.3 Videoclips: Messung von Qualität

Für die Messung der Qualität von Videoclips gelten die gleichen Aussagen, wie für Still Images. Hier wird die Messung aber nicht mit der SNR-Methode durchgeführt, sondern mit einem Qualitätswert

Q_v. Dieser kombinierte, gewichtete Wert berücksichtigt zusätzlich den Einfluß von Bewegung. Neben der Messung von Qualität mit dem Qualitätswert Q_v existieren noch anderen Verfahren, auf dieses soll hier aber nicht weiter eingegangen werden.

1.3.1 Messwert Q_v

Wie bereits erwähnt, ist Q_v ein kombinierter Meßwert. Er setzt sich aus drei unterschiedlichen Meßwerten m_1, m_2 und m_3 zusammen, die ihrerseits gewichtet werden. Qv ist nun folgendermaßen definiert:

$$Q_v = 4.77 - 0.992*m_1 - 0.272*m_2 - 0.356*m_3$$

Hier bei steht

m_1 - für den Informationsverlust bei räumlichen Details (Kanten) im Videoclip

m_2 - für den Informationsverlust bei Bewegungen im Videoclip

m_3 - für den Informationsverlust bei Fehlern (Flackern, Rauschen) im Videoclip

Im Folgenden soll genauer auf diese drei Meßwerte eingegangen werden.

m_1

Für den Meßwert m_1 wird die räumliche Information $SI(F_n)$ (**S**partial **I**nformation) für jeden Frame F_n eines Videoclips berechnet. Der dazu verwendete Operator ist der Sobelfilter, der Kanten in einem Frame herausfiltert. Kanten werden hierbei durch Helligkeitsunterschiede ausgemacht. $SI(F_n)$ berechnet sich dann folgendermaßen:

$$SI(F_n) = STD_{space}(Sobel(F_n))$$

Hierbei ist STD_{space} die gemittelte Standardabweichung über die horizontale und vertikale Dimension in einem Frame F_n des Videoclips. Große Werte für SI kennzeichnen mehr räumliche Details. Das bedeutet, es existieren mehr Kanten im Frame F_n.

Ein gewichteter, repräsentativer Wert für den Informationsverlust von räumlichen Details eines Frames n des komprimierten Videoclips berechnet sich dann wie folgt:

$$R_n = 5.81 * ((SI(O_n)-SI(K_n)) / SI(O_n))$$

wobei $\mathbf{O_n}$ Frame n des Originalvideoclips und $\mathbf{K_n}$ Frame n des Kompressionsvideoclips kennzeichnet.

Der Meßwert m_1 berechnet sich nun über:

$$m_1 = RMS(R_n)$$

RMS bezeichnet die Wurzel aus der gemittelten Summe der einzelnen R_n Werte zum Quadrat. Hierbei bezieht sich der Wert m_1 also auf alle Frames, d.h. die Anzahl der Frames des gesamten Videoclips wird mit einbezogen.

m2 und m3

Die Meßwerte m_2 und m_3 berücksichtigen Veränderungen von Frames, bezogen auf die zeitliche Abfolge. Hierzu werden Bewegungsdifferenzbilder ΔF_n gebildet. Diese werden durch Pixelunterschiede von zwei aufeinanderfolgenden Frames berechnet:

$$\Delta F_n = F_n - F_{n-1}$$

Die zeitliche Information $TI(F_n)$ (**T**ime **I**nformation) für jeden Frame F_n eines Videoclips läßt sich dann folgendermaßen berechnen:

$$TI(F_n) = STD_{space}(\Delta Fn)$$

Je mehr Bewegung zwischen zwei Frames existiert, desto größer werden die Werte für TI.

m2 im Detail

Ein gewichteter, repräsentativer Wert für die zeitliche Information eines Frames n des komprimierten Videoclips berechnet sich dann wie folgt:

$$Z_n = 0.108 * MAX(TI(O_n)-TI(K_n) , 0)$$

wobei $\mathbf{O_n}$ wieder Frame n des Originalvideoclips und $\mathbf{K_n}$ Frame n des Kompressionsvideoclips kennzeichnet und sich Z_n auf den Informationsverlust von Frame n bezieht.

Der Meßwert m_2 für den kompletten, komprimierten Videoclip berechnet sich nun als Standardabweichung der Informationsverluste aller Frames:

$$m_2 = STD_{time} (Z_n)$$

wobei zuvor noch ein Hochpassfilter, der Veränderungen in der Bewegung stärker herausarbeitet, auf jeden Frame n angewendet wird. Auf eine genauere Beschreibung der Arbeitsweise des Hochpassfilters soll hier nicht weiter eingegangen werden.

m3 im Detail

Der Meßwert m_3 wird dazu benutzt, Flackern im Bildmaterial, periodisches Rauschen oder Fehler, die bei der Übertragung entstehen können, zu berücksichtigen:

$$m_3 = MAX_{time}(4.23 * LOG_{10} (TI(K_n)/TI(O_n)))$$

wobei MAX_{time} den größten Wert aller Frames zurückliefert. Auf eine ausführliche, Beschreibung der theoretischen Grundlagen für den Meßwert m_3 soll hier verzichtet werden, da dies den Rahmen der Ausarbeitung sprengen würde.

Zusammenfassung

Die Beschreibung der Berechnung des Meßwertes Q_v sollte zeigen, wie aufwendig es ist, eine wissenschaftliche Definition für den Begriff Qualität und ihre Messung zu finden, die dann in der Anwendung auswertbare Ergebnisse liefert. Auf die Beschreibung alternativer Methoden zur Bestimmung von Qualität soll hier verzichtet werden. Hierzu eignet sich das Studium der angegebenen Literatur im Literaturverzeichnis.

Zur Berechnung von Q_v nach oben beschriebener Methode, wurde von *Effelsberg und Steinmetz* die verwendete MPEG Software erweitert um die Berechnung von Q_v effizienter zu machen, da die relevanten Daten während der MPEG Kompression vorhanden sind.

1.4 Videoclips: Messung von Performance

Die Messung von Performance bei Videoclips wird in gleicher Weise durchgeführt wie bei den Still Images. Hierbei wird der gesamte Videoclip als Einheit betrachtet, so wie zuvor ein Bild als Einheit betrachtet wurde.

2 Still Images : Experimenteller Vergleich

Für den experimentellen Vergleich von Still Images wird das standardisierte Kompressionsverfahren JPEG verwendet. Verglichen werden zwei verschiedene Bildmotive. Der Vergleich bezieht sich auf das Originalbild B_{in}, das mit dem komprimierten und wieder dekomprimierten Bild B_{out} verglichen wird.

2.1 Experimentelles Setup

Als Bildquelle dienen zwei, von *Effelsberg und Steinmetz* aufgezeichnete Videosequenzen. Diese Videosequenzen wurden auf einem PC mit einem miroVIDEO DC20 Framegrabber digitalisiert. Der Prozeß der Digitalisierung lieferte zwei AVI Datei mit Still Images einer Auflösung von 360x270 Pixel. Aus diesen Dateien wurden dann jeweils zwei Einzelbilder ausgewählt, wobei die Auflösung dieser Einzelbilder auf 352x288 Pixel skaliert wurde. Diese Einzelbilder dienen im Folgenden als Originalbilder (Bildquellen).

Für die Experimente mit Still Images wurde von *Effelsberg und Steinmetz* die Kodiersoftware der PVRG (Portable Video Research Group at Stanford University) benutzt. Diese Software erlaubt es für jeden 8x8 großen DCT Block zwei Quantisierungstabellen (für den jeweiligen Luminance- und Chromincance Anteil) festzulegen. Desweiteren erlaubt diese Software die Festlegung eines globalen Quantisierungsfaktors (im Folgenden mit Q-Faktor bezeichnet), der sich auf alle DCT-Koeffizienten bezieht, indem er mit jedem Wert der zwei Quantisierungstabellen multipliziert wird, und diese Werte somit vergrößert. Für die Experimente wurden die Standardeinstellung der Quantisierungstabellen der Software beibehalten, dafür aber der Q-Faktor variiert.

2.1.1 Referenzmaterial

Als Bildmotive dienen das Schloß und der Park der Stadt Mannheim. Das Schloßmotiv stellt einen Teil der Außenfassade dar. Hierbei wurde darauf geachtet, daß das Motiv eine klare Struktur, klare Linien und Kanten besitzt. Dies ergibt sich aus der Anordnung und Gestaltung der Fenster (schachbrettartig), aber auch aus der Verputzung der Außenwände. Helle und dunkle Übergänge wechseln sich horizontal und vertikal ab. Beim Parkmotiv hingegen wurde darauf geachtet, daß keine klaren Strukturen und Kanten vorhanden sind, sondern eine Vielfalt von Farben und Strukturvariationen. Diese machen sich in unterschiedlich großen Blättern und Stilen mit unterschiedlichen Färbungen deutlich.

Die folgenden Abbildungen zeigen das Park- und Schloßmotiv in unkomprimierter Form.

Abbildung 2-1: Parkmotiv

Die komprimierten Bilder der Versuchsreihen, die mit verschiedenen Q-Faktoren bearbeitet wurden, sind im Anhang zusammengefaßt.

Abbildung 2-2: Schloßmotiv

2.1.2 Kompressionparameter

Die Quantisierungstabellen der verwendeten Kodiersoftware für die Luminance- und Chrominance Anteile sehen wie folgt aus.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tabelle 2-1: Luminance Quantisierungstabelle für 8x8 DCT-Block

Diese Tabellen sind das Ergebnis von Qualitätsuntersuchungen verschiedener Bildmaterialien, die durch die Softwareentwickler durchgeführt wurden.

Für die globalen Quantisierungsfaktoren werden die Werte 1, 6, 12 und 20 verwendet und die entsprechenden Resultate betrachtet.

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Tabelle 2-2: Chrominance Quantisierungstabelle für 8x8 DCT-Block

2.2 Experimenteller Vergleich : Schloßmotiv

Die folgende Tabelle stellt Kompressionsraten, Kompressions- und Dekompressionszeiten und Qualitätswerte für die vier verschiedenen globalen Quatisierungsfaktoren dar.

Q-Faktor	SNR[db]	R[%]	T_{Komp}[s]	T_{Dekomp}[s]
1	21.07	10.78	0.283	0.283
6	19.24	4.04	0.267	0.267
12	17.68	2.76	0.267	0.233
20	16.73	2.27	0.250	0.233

Tabelle 2-3: Ergebnisse Schloßmotiv (Still Image)

Die Qualität der ersten beiden Bilder (Q-Faktor 1 und 6) unterscheidet sich nur durch 1.83db und dies ist für den Betrachter nicht sichtbar. Auch die Kompressionsraten sind hierbei sehr niedrig (10.78% und 4.04%). Erst für die Q-Faktoren 12 und 20 sind Qualitätseinbußen sichtbar. Diese Einbußen äußern sich in Artefakten, die die 8x8 JPEG Quantisierungsblockgröße sichtbar machen.

Dies läßt sich folgendermaßen interpretieren und ist auch als ein Charakteristikum der JPEG Kompression bekannt. Wenn sich die Farben innerhalb eines 8x8 großen Blocks nicht wesentlich unterscheiden, dann sind die AC Werte nach der DCT Transformation dem DC (Gleichanteil für alle Werte des Blocks) Wert ähnlich. Dies führt dazu, daß sie nach der Quantisierung den gleichen Wert annehmen, also wird hier nur der DC Wert der Quantisierungstabelle kodiert. Existieren viele solcher Blöcke, erhöht sich zwar die Kompressionsrate, aber beim Dekomprimieren sind monochromen Blöcke (Artefakte) sichtbar.

Unterscheiden sich die Farben innerhalb eines 8x8 großen Blocks aber wesentlich, so daß die AC Komponenten nach der Quantisierung verschiedene Werte aufweisen, so wird der später dekomprimierte Block mehr Details (Farbabstufungen) und somit schärfere Kanten enthalten. Die Kompressionsrate steigt. Je größer also der Quantisierungsfaktor gewählt wird, desto mehr monochrome Blöcke können beim dekomprimierten Bild entdeckt werden und desto kleiner wird die Kompressionsrate.

Da nun das Schloßmotiv viele klare Linien und Kanten enthält, führt die Vergrößerung des globalen Quantisierungsfaktors zu leicht sichtbaren Artefakten (Übergang von den Fenstern zur Außenwand). Durch die Quantisierung gehen die Kanteninformationen verloren, denn die Kanten werden durch große Werte (hohe Frequenzen) in den DCT Blöcken repräsentiert. Durch einen großen globalen Quantisierungsfaktor, werden diese hohen Frequenzen mit den mittleren zusammengefaßt, wodurch die Kantenübergänge nicht mehr klar zu erkennen sind.

2.3 Experimenteller Vergleich : Parkmotiv

Im Gegensatz zum Schloßmotiv, weist das Parkmotiv keine klaren Kanten auf. Nach dem ersten Experiment kann für diesen Versuch vermutet werden, daß:

- sich Artefakte erst bei größeren Quantisierungsfaktoren zeigen (durch das Fehlen scharfer Kanten)
- das Bild durch geringe Farbschwankungen in den Grün-, Orange- und Gelbwerten, mit zunehmendem Quantisierungsfaktor unscharf wird

Die folgende Tabelle stellt Kompressionsraten, Kompressions- und Dekompressionszeiten und Qualitätswerte für die vier verschiedenen globalen Quatisierungsfaktoren dar.

Q-Faktor	SNR[db]	R[%]	T_{Komp}[s]	T_{Dekomp}[s]
1	22.92	17.80	0.317	0.317
6	17.25	5.70	0.283	0.267
12	15.22	3.33	0.267	0.250
20	14.18	2.47	0.250	0.250

Tabelle 2-4: Ergebnisse Parkmotiv (Still Image)

Wie bereits beim Schloßmotiv, führt ein Q-Faktor von 1 zu keinem bemerkbaren Einfluß auf die Qualität. Bei einem Q-Faktor von 6 lassen sich im Bild Blockartefakten finden, welche aber nicht als einfarbige Flächen punktuell das Bild stören, sondern das Bild stark verschwimmen lassen. Ab einem Q-Faktor von 12 ist das Bild quasi zerstört. Alle gelben Blühten sind zu einer braungelben Fläche verschmolzen. Es sind nur noch die größeren Konturen im Vordergrund zu erkennen.

In folgenden Abbildungen sind die Qualitäten und Kompressionsraten des Park- und Schloßmotives vergleichend gegenüber gestellt.

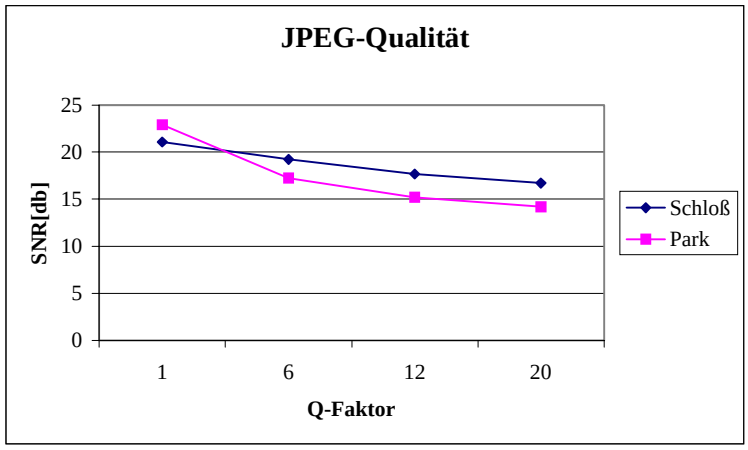


Abbildung 2-3: JPEG Qualitäten im Vergleich (Still Images)

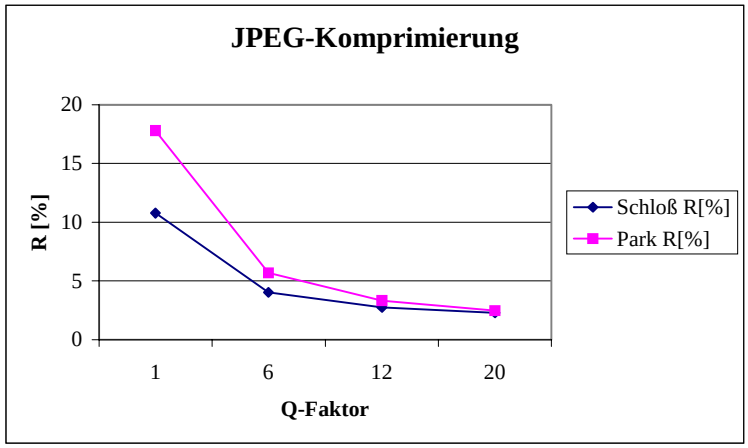


Abbildung 2-4: JPEG Komprimierung im Vergleich (Still Images)

Im Vergleich zum ersten Versuch ist die Qualität des Parkmotives bei einem Q-Faktor von 1 um 1.85db besser als die Qualität des Schloßmotives. Dabei ist aber die Kompressionsrate des Parkmotives um 7.02 % schlechter. Beim zweiten Versuch (Q-Faktor = 6) fällt die Qualität des Parkmotives um 1.99db unter die Qualität des Schloßmotives, dabei ist die Kompressionsrate des Parkmotives nur noch um 1.66 % schlechter. Erst in den Versuchen mit großen Q-Faktoren nähern sich die Kompressionsraten an, welches sich aber auch in den einfarbigen 8x8 Blöcken widerspiegelt.

Die Quantisierungstabelle mit ihren hohen Werten bewirkt eine starke Glättung der Farbverläufe innerhalb der 8x8 Blöcke. Diese Glättung wirkt sich bei den feinen Strukturen im Parkmotiv aber ungünstiger aus als bei den gröberen Strukturen im Schloßmotiv.

3 Videoclips: Experimenteller Vergleich

In diesem Kapitel wird eine Reihe von Experimenten mit Videoclips im MPEG-1 Format vorgestellt.

3.1 Experimentelles Setup

Als Videoquellen dienten zwei, von *Effelsberg und Steinmetz* mit einer PAL Kamera aufgezeichnete Videosequenzen. Diese wurden auf einem PC mit einem miroVIDEO DC20 Framegrabber und zum

Vergleich auf einer Silicon Graphics Indigo II mit dem Galileo Video Digitizer Board und dem Cosmo Compression Board digitalisiert.

Da es mit keiner der genannten Hardware möglich war die Videosequenzen bei einer Bildrate von 25 Bilder pro Sekunde unkomprimiert zu speichern, mußten die „original“ Videosequenzen mittels einer Hardware JPEG-Komprimierung (MJPEG) gewonnen werden. Daß dabei schon während der Digitalisierung Artefakte in die Sequenzen eingebracht wurden, mußte in Kauf genommen werde. Um diesen Effekt so gering wie möglich zu halten, wurde die höchst mögliche MJPEG-Qualität gewählt.

3.1.1 Referenzmaterial

Bei der ersten Videosequenz wurde die Kamera starr auf einen Gemüsestand des Mannheimer Wochenmarktes gerichtet. Dabei entstanden durch Marktbesucher lokale Bewegungen in der Szene. Durch die verschiedenen Gemüsearten wurden feine Strukturen in verschiedenen Farben in die Sequenz aufgenommen.

Die zweite Videosequenz wurde im Park von Mannheim aufgenommen. Es zeigt einen Kameraschwenk über ein Blumenbeet mit angrenzender Rasenfläche.

3.1.2 Parameter

Die hier vorgestellten Experimente wurden mit dem MPEG-1 Sofwarekodierer der PVRG durchgeführt. Bei diesem Experiment wurden pro Videoclip die drei Parameter

- quantization factor (global scaling factor)
- motion vector search range
- picture pattern

untersucht. Dabei wurden Experimentalreihen benutzt, bei denen sich jeweils nur ein Parameter verändert und die Anderen jeweils konstant gehalten wurden.

3.1.3 Quantisierung

Genau wie in den JPEG-Experimenten wurden für alle Versuche die gleichen Quantisierungstabellen benutzt, welche durch einen globalen Quantisierungsfaktor von Versuch zu Versuch modifiziert wurden. Es wurden die Tabelle 5 und 6 benutzt, welche jedem DCT-Wert einen Quantisierungswert zuweisen.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Tabelle 3-1: Quantisierungstabelle MPEG-1 (Intrakodierung)

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Tabelle 3-2: Quantisierungstabelle MPEG-1 (Interkodierung)

Zusätzlich wurden die globalen Quantisierungsfaktoren 1, 15 und 31 benutzt.

3.1.4 Motion Vector Search Range

Bei der benutzten Software wurde die Motion Vector Search Range als Durchmesser angegeben. Wenn die Motion Vector Search Range also mit einem Wert von 8 angegeben ist bedeutet dies, daß ein Bereich von ±4 Pixel nach Ähnlichen Blöcken durchsucht wurde.

3.1.5 Picture Pattern

Für die verschiedenen Bildreihenfolgen werden folgende Abkürzungen verwendet:

Intra = IIII
IP = IPPP
IB = IBBB
BP = IBBPBB

3.2 Einfluß Quantisierungsfaktor

In der ersten Versuchsreihen wurde der Einfluß des Quantisierungsfaktors auf die Qualität, die Kompressionsrate, die Komprimier- und Dekomprimierungszeit genauer untersucht. Dazu wurde zunächst das Picture Pattern Intra benutzt.

3.2.1 Intra

Die Grafiken *Park INTRA* und *Markt INTRA* zeigen die Ergebnisse des ersten Versuchs. Dabei ist zu beachten, daß übersichtshalber die Werte des Parameter *CodeTime* durch Zehn geteilt wurden.

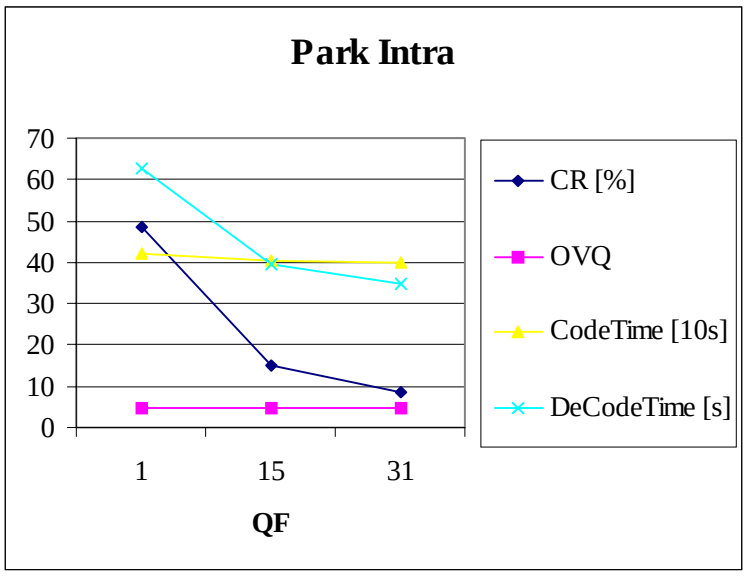


Abbildung 3-1: Ergebnisse Parkmotiv (MPEG-1 Intra)

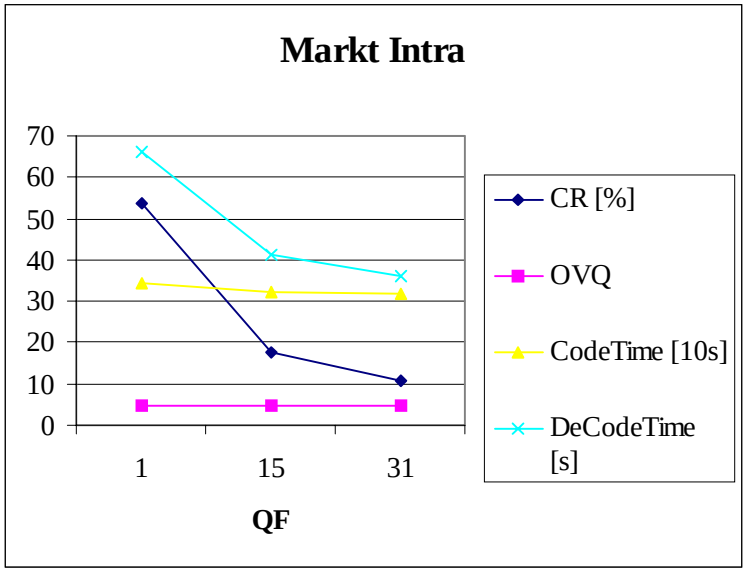


Abbildung 3-2: Ergebnisse Marktmotiv (MPEG-1 Intra)

Bei diesem Versuch zeigen die Ergebnisse deutlich, daß der Quantisierungsfaktor (QF) einen großen Einfluß auf die Kompressionsrate (CR) und die Dekodierzeit (DeCodeTime) hat. Wobei die Qualität (OVQ) und die Komprimierzeit (CodeTime) nur gering beeinflußt werden.

Daß mit steigendem Quantisierungsfaktor auch die Kompressionsrate abnimmt, ist nach den JPEG-Versuchen nicht mehr verwunderlich. Die DCT-Werte werden durch große Werte in der durch den Quantisierungsfaktor modifizierten Quantisierungstabelle geteilt, was zu vielen Nullwerten führt. Die Verringerung der Dekodierzeit ist durch die starke Abnahme des Datenaufkommens zu erklären. Daß dabei die Qualität der Videoclips nicht gravierend schlechter wird, ist auf eine ausgewogenen Quantisierungstabelle zurückzuführen.

Um den Einfluß des Quantisierungsfaktors auf die beiden unterschiedlichen Videos genauer zu analysieren, wurden die Koeffizienten, welchen nach der Quantisierung nicht gleich Null sind gezählt und in folgender Grafik aufgetragen.

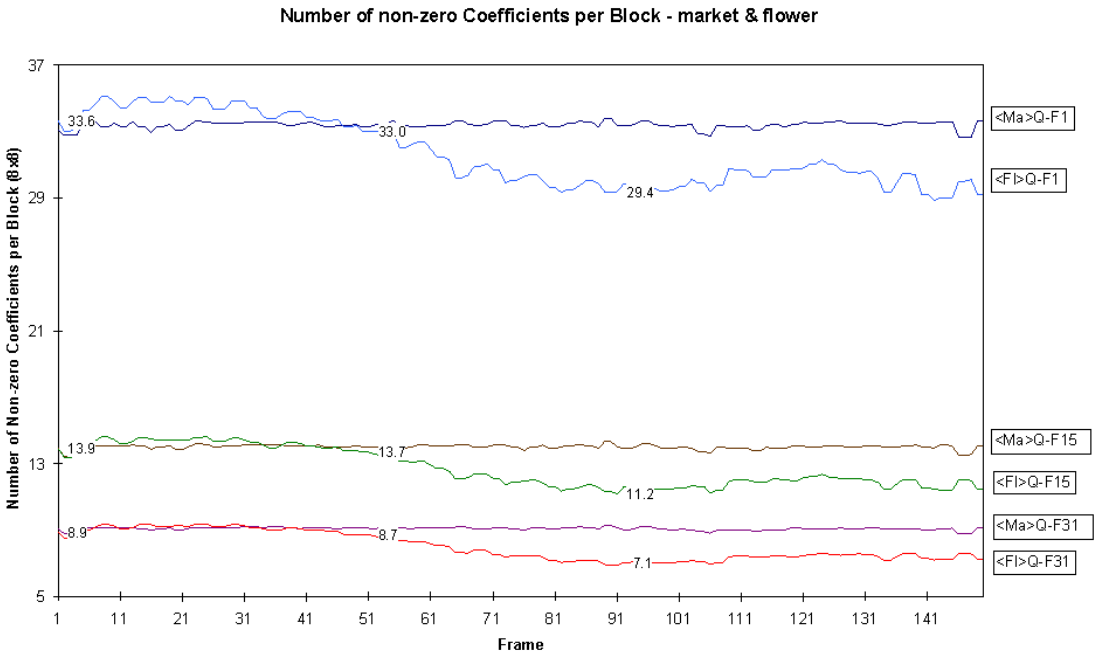


Abbildung 3-3: Non-Zero Koeffizienten

Dabei fällt auf, daß die Werte für die Marktszene (<Ma>) relativ gleichmäßig um einen Durchschnittswert verteilt sind. Im Gegensatz dazu verhalten sich die Werte für das Parkmotiv (<Fl>), welche in zwei Bereiche unterteilt werden können. Im ersten Teil, von Bild 0 bis Bild 50, entstehen größere Werte als in den folgenden Bildern des Videoclips. Dies ist durch den Kameraschwenk und die damit veränderte Motivsituation zu erklären. Wärend im ersten Teil der Parkvideosequenz die Blumen mit feinen Strukturen überwiegen und der Rasen mit flächigeren Strukturen eher eine untergeordnete Rolle spielt, ist es im zweiten Teil umgekehrt.

Durch die starre Kamera in der Marktszene unterscheiden sich die einzelnen Bilder nur durch die Bewegung der Marktbesucher. Da diese lokalen Bewegungen relativ kleiner Flächen ändern aber nicht das gesamt Motiv, führen sie nur zu lokalen Schwankungen in obiger Kurve.

3.2.2 Inter (IP)

Bei den Versuchen mit dem Picture Pattern IP wurde eine Motion Vector Search Range von eins eingesetzt.

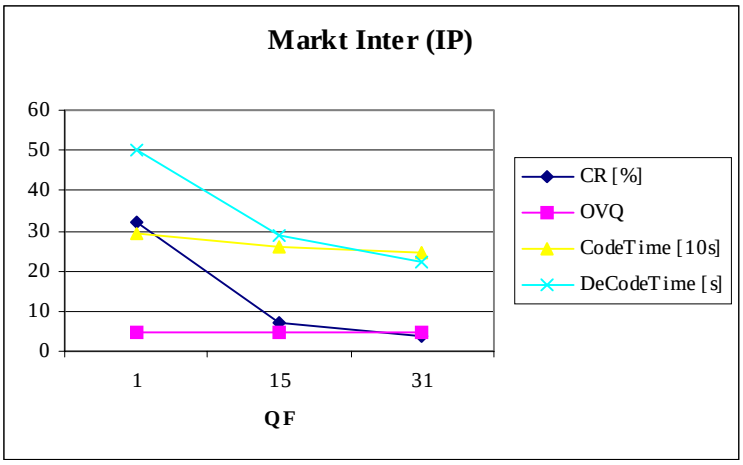


Abbildung 3-4: Ergebnisse Marktmotiv (MPEG-1 IP)

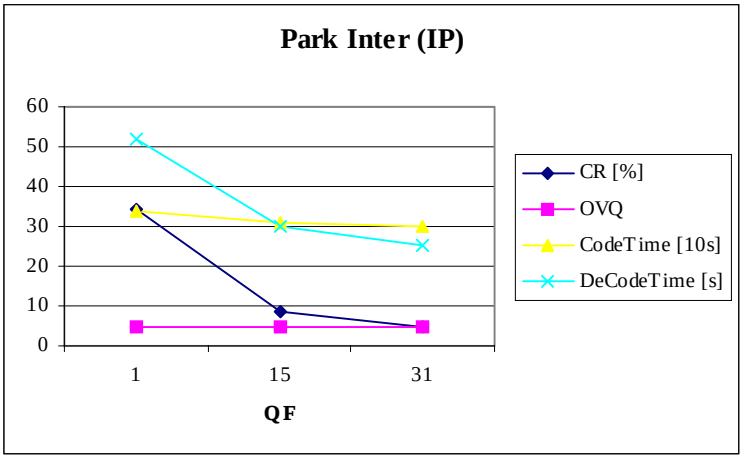


Abbildung 3-5: Ergebnisse Parkmotiv (MPEG-1 IP)

Die Ergebnisse der zweiten Versuchsreihe zeigen denselben Trend wie die ersten Versuche. Mit zunehmenden Quantisierungsfaktor verändern sich die Kompressionsrate und die Dekodierzeit drastisch, während Kodierzeit und Qualität nur gering schwanken.

Im direkten Vergleich zur Intrakodierung zeigen sich aber Verbesserungen in der Kompressionsrate, Kodier- und Dekodierzeit. (vgl. 3.2.5)

In beiden Videosequenzen sind langsame Bewegungen festgehalten, wodurch nur geringe Veränderungen der Makroblöcke zwischen zwei benachbarten Frames entstehen. Dies führt zu einer effektiven Motion Compensation während der Komprimierung und damit zu einer niedrigeren Kompressionsrate. Durch die kleine Motion Vector Search Range von 1 kann der Makroblockvergleich während der Komprimierung schnell ausgeführt werden, was zu einer Verkürzung der Kompressionszeit führt. Die Einsparungen bei der Dekodierzeit lassen sich zusätzlich durch ein vermindertes Datenaufkommen erklären.

3.2.3 Inter (IB)

Die nächste Versuchsreihe beschäftigte sich mit den bidirektionalen Bildern.

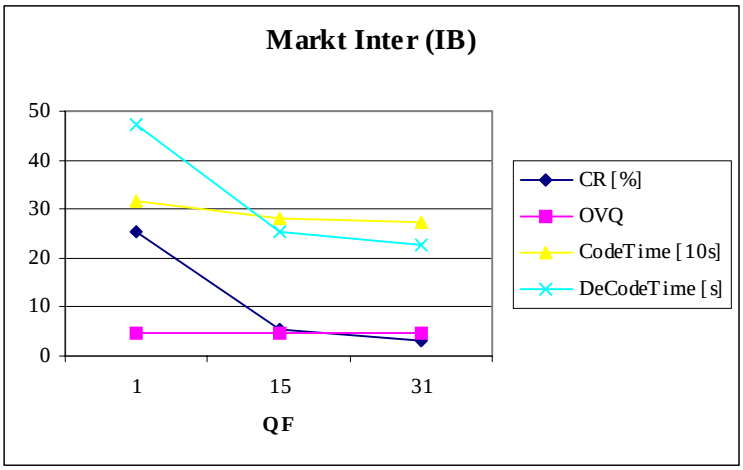


Abbildung 3-6: Ergebnisse Marktmotiv (MPEG-1 IB)

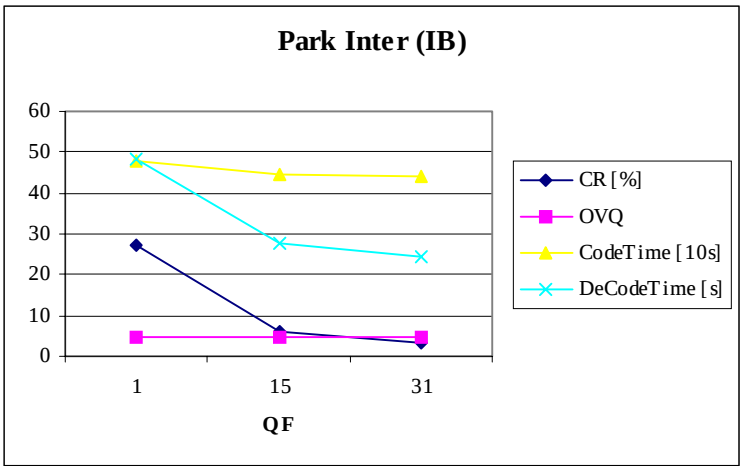


Abbildung 3-7: Ergebnisse Parkmotiv (MPEG-1 IB)

Da sich B-Frames auf das folgende, das vorherige, das folgende und das vorherige oder auf gar kein Bild beziehen können, muß der Kodieralogrithmus die Entscheidung fällen, welcher Bezug gewählt wird. Dazu benötigt er Zeit, welches sich in den Ergebnissen zeigt. Im Vergleich zur IP-Bildfolge (vgl. 3.2.5) zeigt sich, daß das Kodieren einer IB-Bildfolge langsamer ist. Im Gegensatz dazu verhalten sich die Kompressionsraten, welche in diesen Versuchen immer unter den Raten der IP- und intrakodierten Videos lagen.

3.2.4 Inter (BP)

In dieser letzten Versuchsreihe wurde die Kombination von P- und B-Frames genauer untersucht.

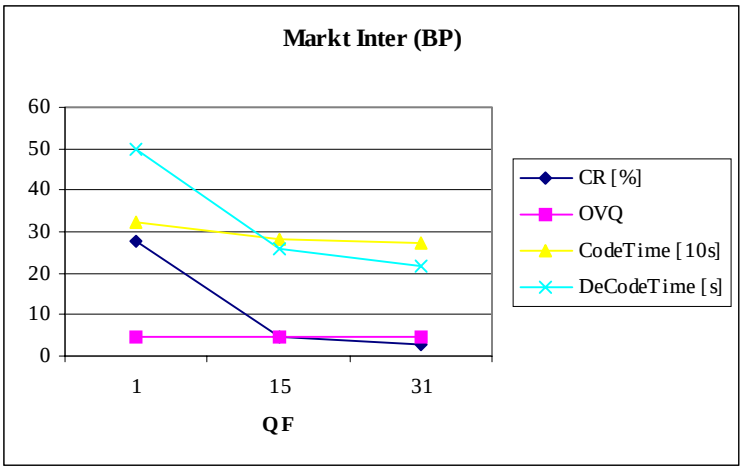


Abbildung 3-8: Ergebnisse Marktmotiv (MPEG-1 BP)

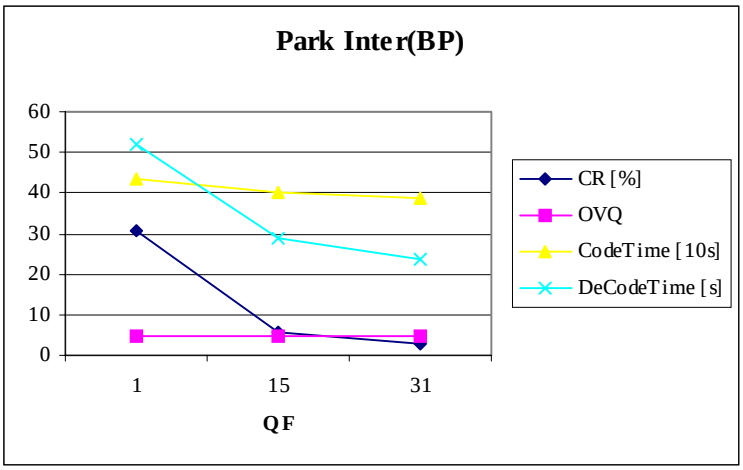


Abbildung 3-9: Ergebnisse Parkmotiv (MPEG-1 BP)

Dabei zeigen die Ergebnisse, daß sich die Kompressionsraten durchaus vergleichbar sind.

3.2.5 Zusammenfassung

Um die Versuchsergebnisse im direkten Vergleich zu betrachten, sind im Folgenden die Ergebnisse aller Versuchsreihen pro Parameter zusammengefaßt.

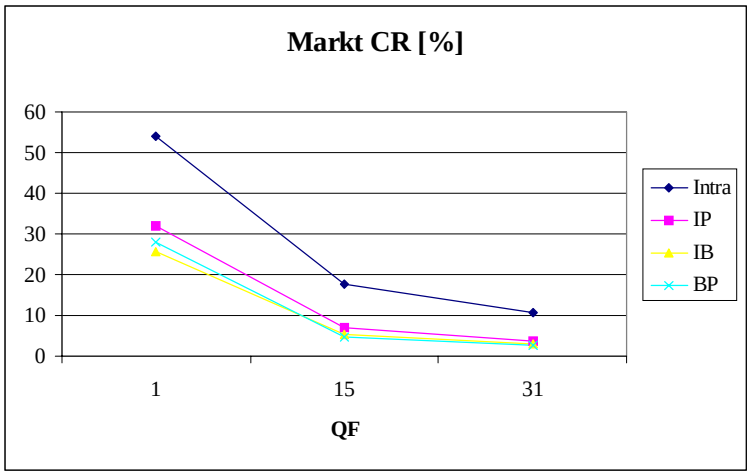


Abbildung 3-10: Compression Ratio Markt (Vergleich)

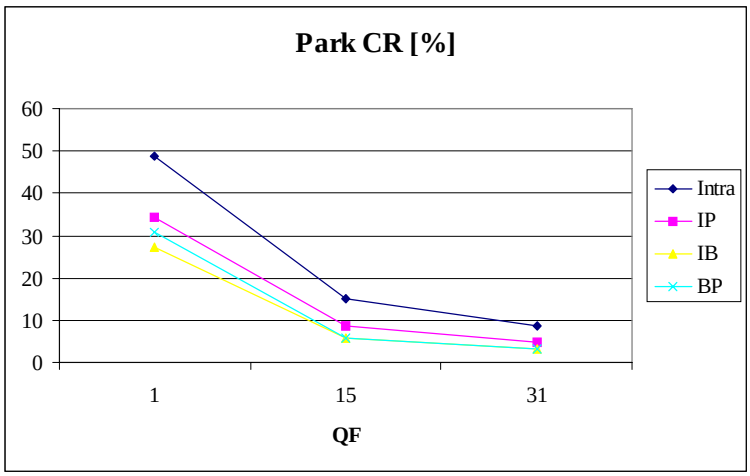


Abbildung 3-11:Compression Ratio Park (Vergleich)

Bei den Kompressionsraten fällt auf, daß die Ergebnisse für die picture pattern IP, IB und BP relativ nah zusammen liegen. Dies liegt wohl an dem Verfahren der Motion Compensation, welche das Speichern aller Frames in vollem Umfang unnötig macht.

Außerdem zeigen die Ergebnisse ein näheres zusammenrücken der Werte je höher der Quantisierungsfaktor ist.

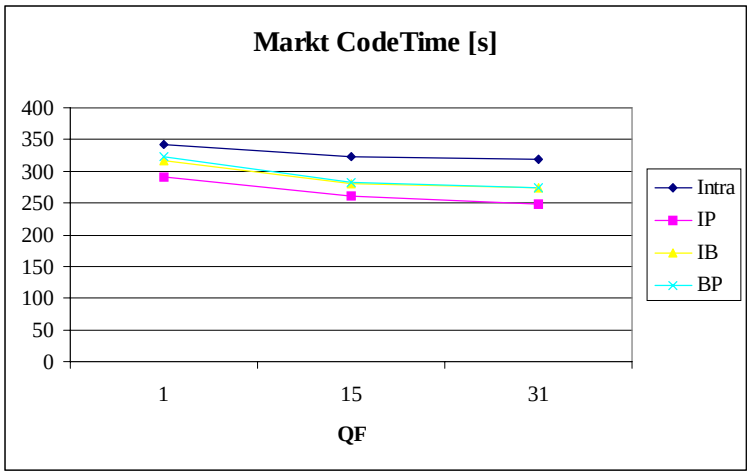


Abbildung 3-12: Code Time Markt (Vergleich)

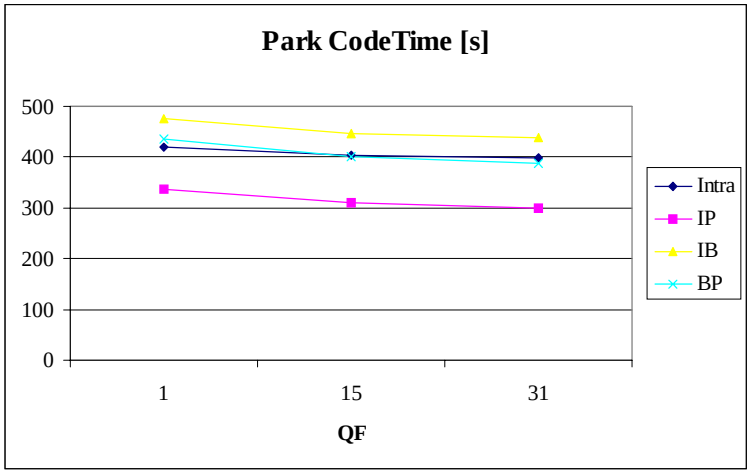


Abbildung 3-13: Code Time Park (Vergleich)

Bei den Kompressionszeiten zeigt sich deutlich, daß diese sowohl vom picture pattern als auch vom Motiv abhängig sind.

Besonders zeigt sich dies am picture pattern IB, welches beim Marktmotiv noch im Mittelfeld der Kompressionszeiten liegt, beim Parkmotiv aber die langsamste Komprimiermethode darstellt.

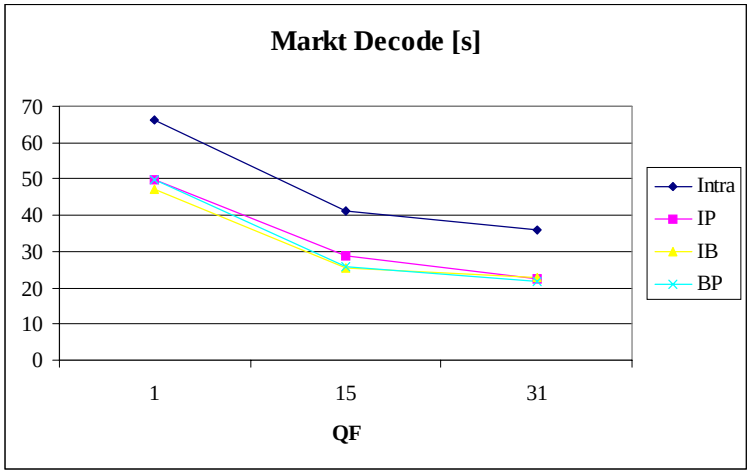


Abbildung 3-14: Decode Time Markt (Vergleich)

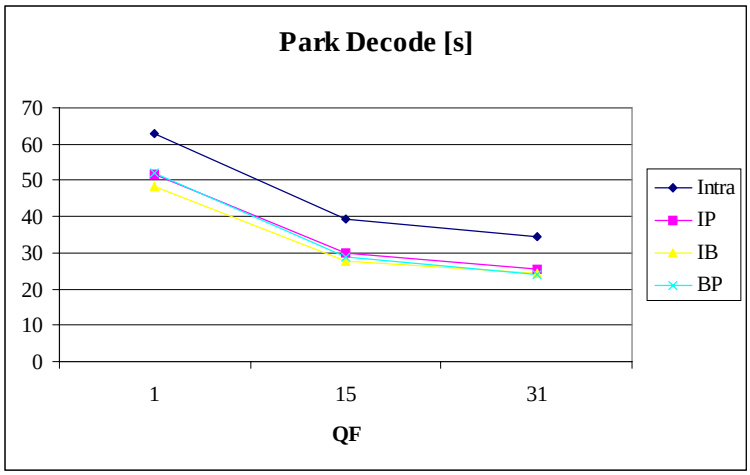


Abbildung 3-15: Decode Time Park (Vergleich)

Bei den Dekodierzeiten verhalten sich IP-, IB- und BP-Muster mit nahezu identischen Werten gleich. Die Intrakodierung zeigt auch hier die höchsten Werte.

3.3 Einfluß Motion Vector Search Range

Im Folgenden soll aufgezeigt werden, wie die Wahl der Motion Vector Search Range die Qualität, das Kompressionsverhältnis und die Kompressions- und Dekompressionszeit beeinflußt. Die Experimente beziehen sich auf das Parkmotiv, wobei zwischen den drei Picture Patterns IP, IB und BP unterschieden wird. Der globale Quantisierungsfaktor erhält in allen drei Fallen den konstanten Wert 15 und der Motion Vector variiert zwischen den Werten 1, 8 und 15.

3.3.1 Picture Pattern IP

Die folgende Tabelle zeigt die Ergebnisse des ersten Experiments mit dem Picture Pattern IP.

Motion Vector	Q_v	R[%]	T_{comp}[s]	T_{decomp}[s]
1	4.741	8.67	309.3	30.1
8	4.739	8.17	352.9	29.6
15	4.738	8.09	381.7	29.5

Tabelle 3-3: Ergebnisse Parkmotiv (MPEG-1 IP)

Hierbei fällt erst einmal auf, daß sich die Kompressionsraten nicht wesentlich unterscheiden. Auch die Dekompressionszeiten liegen nah beieinander. Dies spiegelt die Tatsache wieder, daß die Dekompressionszeit wesentlich von der zuvor erreichten Kompressionsrate abhängt. Je mehr Daten komprimiert wurden, desto länger dauert die Dekompression. Da nun aber die Kompressionsraten fast gleich sind, spiegelt sich dies in fast gleichen Dekompressionszeiten wieder.

Desweiteren spiegelt sich die Tatsache, daß die Kompressionsraten nicht wesentlich auseinander liegen, auch in ähnlichen Qualitätswerten Q_v wieder, denn wenn die Kompression in allen drei Fällen gleich stark ist, so ist auch der Informationsverlust in den drei Fällen gleich groß, was zu gleichen Q_v Werten führt, da sich das bearbeitete Material nicht wesentlich voneinander unterscheidet.

Letztlich fällt auf, daß die Kompressionszeit wächst, je größer der Motion Vector Wert gewählt ist. Dieser Effekt beruht natürlich auf dem größeren Rechenaufwand, der betrieben werden muß, um

möglichst übereinstimmende Blöcke in aufeinanderfolgenden Bildern (für die Motion Compensation) zu finden.

Die Analyse der statistischen Auswertung des Kodiervorganges (von der Software bereitgestellt) ergab, daß zwei aufeinanderfolgende Frames meist interkodiert wurden und kein Motion Vector (für eine Motion Compensation) benutzt wurde, was sich, trotz Vergrößerung des Motion Vector Bereichs, in gleichen Kompressionsraten (keine Verwendung von Motion Compensation) widerspiegelt.

3.3.2 Picture Pattern IB

Die Ergebnisse des Versuchs mit dem Picture Pattern IB unter Verwendung einer bidirektionalen Interpolation werden in der folgenden Tabelle zusammengefaßt.

Motion Vector	Q_v	R[%]	T_{comp}[s]	T_{decomp}[s]
1	4.710	5.88	445.9	27.7
8	4.703	5.18	624.4	26.6
15	4.703	5.00	781.8	26.2

Tabelle 3-4: Ergebnisse Parkmotiv (MPEG-1 IB)

Wie schon zuvor liegen die Qualitätswerte Q_v und die Kompressionsraten nah beieinander. Dementsprechend verhalten sich die Dekompressionszeiten. Besonders auffällig gegenüber den Versuchen mit dem Picture Pattern IP ist, daß die Kompressionszeiten bei den unterschiedlichen Motion Vector Werten extrem ansteigen. Dies hängt mit den gesteigerten Berechnungen zusammen, die für die bidirektionale Interpolation bei der Suche nach gleichen Blöcken verwendet werden müssen. Je größer der Motion Vector gewählt wird, desto größere Blöcke müssen verglichen werden, und dementsprechend steigt die Berechnungszeit.

3.3.3 Picture Pattern BP

Die Ergebnisse des letzten Versuchs (Picture Pattern BP) sind in der nachfolgenden Tabelle zusammengefaßt.

Motion Vector	Q_v	R[%]	T_{comp}[s]	T_{decomp}[s]
1	4.713	5.82	400.3	28.7
8	4.708	4.79	527.4	26.8
15	4.707	4.48	641.2	26.4

Tabelle 3-5: Ergebnisse Parkmotiv (MPEG-1 BP)

Die zuvor genannten Gründe für die Beziehungen zwischen den Qualitätswerten, den Kompressionsraten und den Dekompressionszeiten, gelten auch hier. Nur die Kompressionszeiten sind im Gegensatz zum Picture Pattern IB geringer, da der Berechnungsaufwand für die Interpolation wegfällt.

Zusammenfassend läßt sich sagen, daß in den drei Versuchsreihen die Verwendung der Motion Compensation nur wesentlich zum Anstieg der Kompressionszeit führte. Qualität und Kompressionsraten blieben in den Versuchen weitestgehend gleich.

3.4 Einfluß Picture Pattern

In einer letzten Versuchsreihe wurde der Einfluß des Picture Pattern auf die Kompressionsrate, die Komprimierungs- und Dekomprimierungszeit genauer untersucht. Dazu wurde ein Quantisierungsfaktor 15 und ein Motion Vector von 8 konstant gewählt.

Da beim IP-Muster im Vergleich mit dem IB-Muster nur P- und B-Fames ausgetauscht sind, spiegeln die Ergebnisse für beide Muster einen Vergleich der Frametypen wieder.

Bei den Kodierzeiten zeigt sich, daß die P-Frames schneller zu berechnen sind. Dies ist auch offensichtlich, da sie nur in eine „Richtung“ (Referenzframe in der Vergangenheit beschreibt die Zukunft) wirken. Bei den B-Frames ist der Berechnungsaufwand größer, da der Referenzframe sowohl in der Vergangenheit als auch in der Zukunft liegen kann.

Beim BP-Muster überwiegen die B-Frames, da durch liegen die Ergebnisse für IB- und BP-Muster auch sehr nah zusammen.

4 H.261 Videokompression

Die Experimente mit dem Standard H.261 wurden von *Effelsberg und Steinmetz* in der gleichen Weise durchgeführt, wie die Experimente mit dem MPEG-1 Verfahren. Es wird die Qualität, die Kompressionsrate und die Kompressions- und Dekompressionszeit betrachtet. Weil der H.261 Standard nicht die Festlegung eines Picture Patterns vorsieht, bezieht sich die anschließende Beschreibung der Versuchsergebnisse auf die Variation des Quantisierungsfaktors und der Motion Vector Search Range.

4.1 Experimentelles Setup

Wird bei H.261 von einem Quantisierungsfaktor (Q-Faktor) gesprochen, so bezieht sich dieser nicht, wie bei den MPEG-1 Versuchen, auf eine Quantisierungsmatrix, wobei die Einträge der Matrix vor der eigentlichen Quantisierung mit dem Q-Faktor multipliziert werden. Bei H.261 wird allein der Q-Faktor als globaler Quantisierungsfaktor benutzt, eine Quantisierungsmatrix ist im festgelegten Standard nicht vorgesehen. Für die folgenden Versuche variiert dieser Q-Faktor zwischen 1, 15 und 31.

Für die Versuche mit den unterschiedlichen Motion Vektoren, variieren diese zwischen 1, 15 und 31, wobei der Q-Faktor konstant den Wert 15 behält.

Das zugrundeliegende Referenzmaterial ist das Parkmotiv.

4.2 Einfluß Quantisierungsfaktor

Um den Einfluß des Quantisierungsfaktors darzustellen, werden die Ergebnisse von zwei Experimentalreihen vorgestellt. Bei der ersten Reihe wird eine Intrakodierung der einzelnen Frames und bei der zweiten die Interkodierung genauer betrachtet. Dabei wird im zweiten Teil ein konstanter Motion Vector von eins verwendet.

Den Erwartungen zufolge hat der Quantisierungsfaktor (QF) einen großen Einfluß auf die Qualität des Videoclips. Da der selbe QF auf alle DCT-Koeffizienten angewendet wird, führt dies bei steigendem QF schnell zu klar sichtbaren Blockartefakten.

4.2.1 Intra Coding Mode

Die Ergebnisse (vgl. Tabelle 4-1) zeigen deutlich, wie die Kompressionsrate mit steigendem QF abfällt, die Kompression also größer wird. Erstaunlich ist, daß die Kompressionszeit scheinbar nicht im direkten Zusammenhang zum QF steht, da sie mit seinem steigenden erst kleiner dann größer wird.

Coding mode	Intra		
Q-Faktor	1	15	31
Q_v	4.758	4.717	4.559
R [%]	66.37	7.43	3.65
T_{comp} [s]	416.1	409.0	417.6
T_{decomp} [s]	57.0	22.6	20.3

Tabelle 4-1: Ergebnisse Parkmotiv (H.261 Intra)

4.2.2 Inter Coding Mode

In diesem Versuch zeigt sich, daß die Inter Kodierung gegenüber der Intra Kodierung eine wesentlich bessere Kompression bietet. Diese Verbesserung ist auf die Mechanismen Prediction Error und Motion Vector, welche schon bei den MPEG Versuchen genauer betrachtet wurden, zurückzuführen.

Interessanterweise verhält sich die Kompressionszeit in diesem Versuch in Abhängigkeit von QF. Mit steigendem QF sinkt die Kompressionszeit.

Coding mode	Inter		
Motion vector search range	1		
Q-Faktor	1	15	31
Q_v	4.759	4.715	4.557
R [%]	48.78	4.94	2.33
T_{comp} [s]	267.6	226.8	222.5
T_{decomp} [s]	47.8	18.3	16.2

Tabelle 4-2: Ergebnisse Parkmotiv (H.261 Inter)

4.3 Einfluß Motion Vector Search Range

Die folgende Tabelle enthält die Ergebnisse der letzten Versuchsreihe, in der der Wert des Motion Vector zwischen 1, 15 und 31 variiert.

Motion Vector	Q_v	R[%]	T_{comp}	T_{decomp}
1	4.715	4.94	226.8	18.3
15	4.718	4.89	277.7	19.2

31	4.702	3.72	389.9	20.3

Tabelle 4-3: Ergebnisse H.261

Von Interesse ist hierbei, daß die Kompressionszeit auffällig stark ansteigt, je größer der Motion Vector gewählt wird. Dies hängt wiederum mit dem gesteigerten Rechenaufwand für die Motion Compensation zusammen. Dieser Rechenaufwand resultiert dann aber in einem besseren Kompressionsverhältnis, trotz bewegungsintensivem Bildmaterial.

Die Dekompressionszeiten sind fast konstant, d.h. der Motion Vector hat keinen (auffälligen) Einfluß auf die Dekompression, denn für die Reproduktion eines Makroblocks spielt es keine Rolle, wie weit er von der ursprünglichen Position entfernt ist, im Gegensatz zum Auffinden eines ähnlichen Blocks.

Eine weitere Auffälligkeit, die sich nicht direkt aus der oben angegebenen Tabelle ableiten läßt, die aber bei den Versuchen von *Effelsberg und Steinmetz* entdeckt wurde, ist folgende: Die Kompressionszeiten für interkodierte Videoclips waren besser (kleiner) als die Kompressionszeiten für die intrakodierten, obwohl anzunehmen wäre, daß gerade die Kodierung von Differenzbilder (bei der Interkodierung) aufwendiger ist, als das Kodieren von ganzen Frames (bei der Intrakodierung). Erklärt werden kann dies damit, daß die geringer Menge an Daten (interkodiert) bei der Übertragung die zeitaufwendige Kodierung kompensiert. Bei Intrakodierung ist der Zeitaufwand für die Übertragung (große Datenmenge aufgrund von ganzen Frames) sehr viel größer.

Die beschriebene Auffälligkeit spiegelt eine Stärke von H.261 wieder. Da H.261 keine bidirektionale Interpolation für Differenzbilder benutzt, wird nur auf schon vorhandene Frames zurückgegriffen. Bei der Kodierung muß nicht auf bestimmte Frames, die z.B. für eine Interpolationsberechnung benötigt würden, gewartet werden. Dies spiegelt sich in schnelleren Kompressionszeiten wieder.

Der gleiche Effekt tritt auch bei MPEG-1 (Picture Pattern IP) auf, da hier auch nur auf schon vorhandene Frames zurückgegriffen wird.

II. Ausgewählte Anwendungsbeispiele

Im folgenden zweiten Teil der Ausarbeitung wird zunächst auf das DVD (Digital Versatile Disc) Standard eingegangen. Anschließend werden die einzelnen Schritte bei der Erstellung eines MPEG-1- und MPEG-2 Videoclip dargelegt.

5 DVD - Digital Versatile Disc

Seit mehr als 15 Jahren existiert nun schon die Compact Disc. In dieser Zeit haben natürlich Fabrikations-, Kodierungs- und Kompressionstechniken enorme Fortschritte gemacht. Diese Fortschritte finden sich in der neuen Generation der optischen Speichermedien wieder. Ein Vertreter dieser Medien ist die DVD bzw. der DVD Standard. Er wurde von den Firmen Matsushita, Toshiba, Philips, Sony und v.a. ins Leben gerufen. Im Folgenden soll in kurzer Form auf die technischen Details der DVD eingegangen werden. Vertiefende Informationen finden sich in der angegebenen Literatur des Literaturverzeichnisses.

5.1 Speicherkapazität

Die auf den gepreßten DVDs eingeprägten Pits sind im Vergleich zur normalen CD bedeutend kleiner und enger angeordnet. Dadurch wird der Spurabstand geringer, was zu einer erhöhten Speicherkapazität gegenüber der normalen CD führt. Hierbei kann sich eine Seite in zwei Informationsebenen unterteilen, die durch unterschiedliche Fokussierung des Lasers voneinander unabhängig gelesen werden können, was die Speicherkapazität nahezu verdoppelt. Desweiteren kann jede DVD grundsätzlich aus zwei Seiten zusammengesetzt werden kann, die Rücken an Rücken miteinander verklebt werden, was zu einer weiteren Verdoppelung der Speicherkapazität führt. Die folgende Tabelle listet, je nach Art der Produktion und Spezifikation (single sided, double sided; single layer, dual layer), die Speicherkapazitäten auf.

Spezifikation	Seiten	Ebenen	Kapazität [GB]	Videospielzeit [h]
DVD-5	1	1	4.7	> 2
DVD-9	1	2	8.5	≈ 4
DVD-10	2	1	9.4	≈ 4.5
DVD-18	2	2	17	> 8

Tabelle 5-1: DVD Speicherkapazitäten

Um das Prinzip der Informationsebenen, auch Layer genannt, genauer zu beschreiben, dienen folgende Abbildungen.

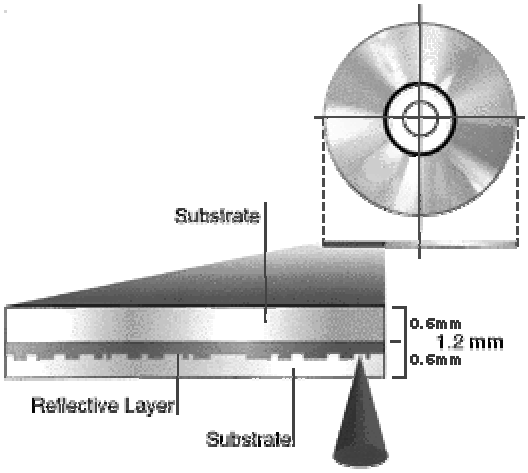


Abbildung 5-1: DVD Single Layer

Beim Single Layer Prinzip wird der Laserstrahl so fokussiert, daß er die Informationsschicht genau so ausliest, wie dies bei der herkömmlichen CD geschieht. Im Bild ist dies durch den Kegel an der Unterseite der DVD verdeutlicht.

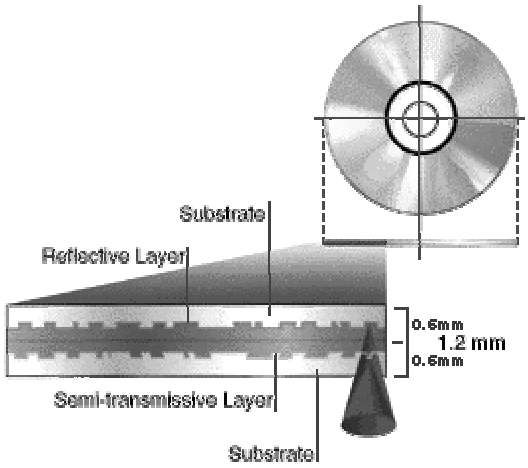


Abbildung 5-2: DVD Dual Layer

Beim Dual Layer Prinzip ist die erste Informationsschicht lichtdurchlässig, sodass der Laserstrahl durch eine andere Fokussierung durch diese Schicht hindurch dringt und somit die zweite Informationsschicht abtasten kann. Auch hier ist dies im Bild durch den Kegel an der Unterseite der DVD verdeutlicht.

Die folgende Abbildung zeigt vergleichend die Abstände der Pits bei einer normalen Compact Disc und bei einer DVD.

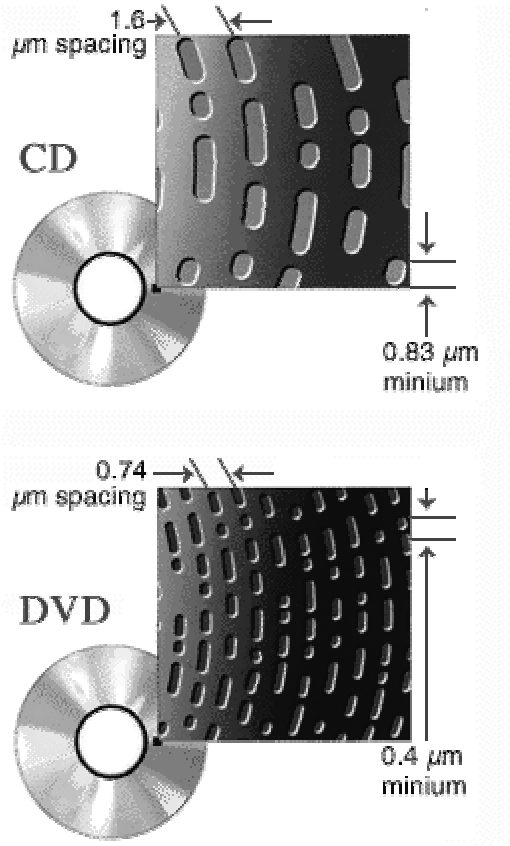


Abbildung 5-3: Abstände der Pits bei CD und DVD

Im Folgenden sollen nun die wesentlichen technischen Informationen zum DVD-Video Standard aufgeführt werden, eine Anwendung von DVD-ROM Prinzipien. Auf Anwendungen im DVD-RAM Bereich sollen hier nicht weiter eingegangen werden.

5.2 Technische Informationen zu DVD-Video

Die technischen Details von DVD-Video können in den Bild- und Tonanteil aufgegliedert werden.

5.2.1 Bild

Ein normales, digitalisiertes Videobild in hoher Auflösung umfaßt eine Datenmenge von etwa 270 Mbit pro Sekunde. Bei einem solchen Datenaufkommen, würde selbst eine DVD nach wenigen Minuten an ihre Grenzen gelangen. Aus diesem Grund wird bei DVD-Video das MPEG-2 Kompressionsverfahren angewendet, welches ein Kompressionsverhältnis von 36:1 ermöglicht. Das bedeutet, daß bei einem Datendurchsatz von durchschnittlich 4.7 Mbit pro Sekunde (für Videosignale ausreichend), 135 Minuten auf einer DVD abgespeichert werden können, was für einen 'Durchschnittsfilm' völlig ausreicht. Hierbei erreicht die DVD eine Horizontalauflösung von nicht weniger als 540 Linien (in der Praxis knapp 500 Linien), das ist sogar noch deutlich mehr als die 425 Linien, die maximal von einer Laserdisc (Bildplatte) zu bekommen sind. Das analoge VHS Format liegt im Vergleich dazu bei 230 Linien.

5.2.2 Ton

Das DVD-Videoformat kann bis zu acht Audiodatenströme enthalten. Jeder davon kann in einem der folgenden Tonformate gehalten sein:

- Lineare Pulse-Code Modulation (PCM) mit 1-8 Audiokanäle bei einer Samplingrate von 48-96kHz und maximal 24bit
- Dolby Digital (AC-3) mit (1-5) +1 Kanäle
- MPEG 2 Audio: 1 - 5+1 oder 7+1 Kanäle

Auch andere Tonformate wie zum Beispiel DTS oder DSD sind erlaubt, allerdings nur in Ergänzung zu mindestens einem der erwähnten Systeme. Auf diese soll hier aber nicht weiter eingegangen werden.

Um die Angaben zu den Tonkanälen genauer zu verstehen, dient folgende Abbildung, die eine 5+1 Anordnung beschreibt.

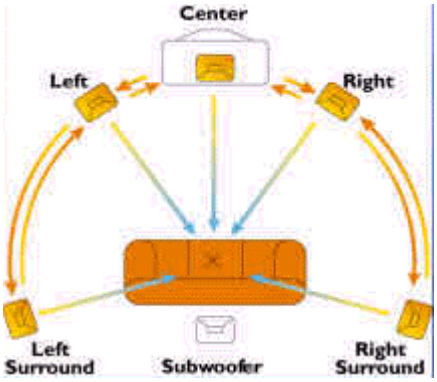


Abbildung 5-4: Kanalverteilung

Hierbei deutet das +1 darauf hin, daß zu den 5 Kanälen ein weiter Kanal für die Ansteuerung eines Subwoofers hinzukommt, der für besonders tiefe Frequenzen im Audiospektrum ausgelegt ist.

Bei einer 7+1 Anordnung kommen noch zwei weitere Kanäle, rechts und links vom Zentrum (Center) hinzu.

Linear Pulse Code Modulation

Die Lineare Puls-Code Modulation funktioniert im Prinzip nicht anders als auf der standard Compact Disc. Beim DVD-Standard sind allerdings verschiedene Qualitätsstufen der PCM-Technik erlaubt. Dies sind Abtastfrequenzen von 48 kHz oder 96 kHz und Wortbreiten von 16, 20 oder 24 Bit. Die bei der standard Compact Disc verwendete Abtastfrequenz von 44,1 kHz wird nicht unterstützt. Aufgrund der Beschränkung der Datenkapazität für den Audioteil einer DVD von 6.144 Mbit pro Sekunde, ergeben sich folgende Möglichkeiten der Kombinationen:

Frequenz [kHz]	Wortbreite [Bit]	Unterstützung von 2-Kanälen	Unterstützung von 5-Kanälen	Unterstützung von 8-Kanälen
48	16	ja	ja	ja
48	20	ja	ja	nein
48	24	ja	ja	nein
96	16	ja	nein	nein
96	20	ja	nein	nein
96	24	ja	nein	nein

Tabelle 5-2: PCM Frequenzkombinationen bei DVD

Dolby Digital (AC-3)

Das AC-3 Verfahren, von den Dolby Laboratories entwickelte, wird bereits seid längerer Zeit in Kinos, auf Laserdiscs (Bildplatten) und im amerikanischen Satellitenfernsehen verwendet. Es handelt sich um ein Mehrkanalverfahren, das das originale PCM komprimiert auf der DVD unterbringt. Dadurch wird eine, je nach Komplexität des Signals, niedrige Datenrate erreicht. Diese liegt zwischen 64 und 448 Kbit pro Sekunde. AC-3 liefert unterstützt den vollen Frequenzumfang.

MPEG-2 Audio

Auch das MPEG-2 Audio Verfahren ist ein digitales Mehrkanalverfahren, das mit Datenkompression arbeitet. Es wurde vom Münchener Institut für Rundfunktechnik und Philips entwickelt und standardisiert. Die Datenrate reicht von 32 bis 912 Kbit pro Sekunde mit einem Durchschnitt von 384 Kbit pro Sekunde. MPEG-2 bietet Rückwärtskompatibilität mit dem MPEG-1 System.

5.2.3 Zusätzliches

Die hier erwähnten zusätzlichen Optionen von DVD-Videos sind Länderkodes, Untertitel und Kopierschutzverfahren.

Länderkodes

Bei der Entwicklung des DVD Standards wurden, aufgrund von Einwänden der amerikanischen Filmindustrie, Länderkodes eingeführt. Diese unterteilen den weltweiten Markt in sechs Regionen, mit dem Ziel, einen internationalen Austausch von DVD-Videos zu unterbinden und unterschiedliche Versionen für z.B. den europäischen und amerikanischen Markt (Schnitt, Ton und Untertitel) zu unterstützen. DVD-Videos müssen diese Länderkodes aber nicht enthalten.

Die nachfolgende Abbildung zeigt die Einteilung der Weltkarte in die unterschiedlichen Regionen, auch Zonen genannt.

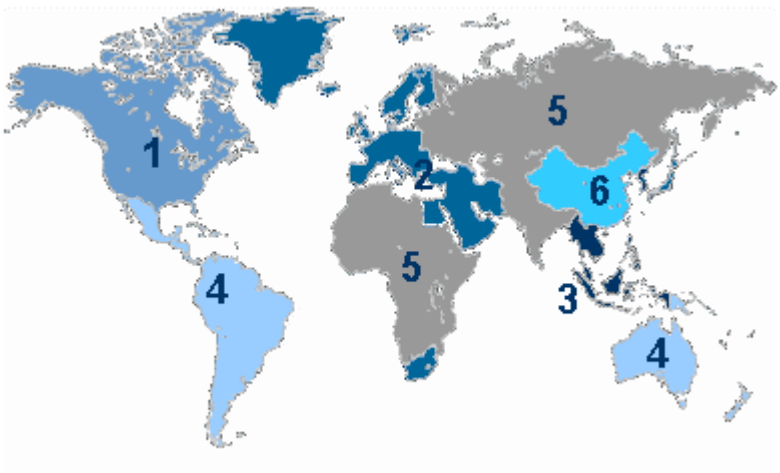


Abbildung 5-5: Länderkodes

Hierbei gilt:

Zone 1: Kanada, USA

Zone 2: Europa, Japan, Naher Osten inklusive Ägypten und Südafrika

Zone 3: Ostasien inklusive Hongkong und Südostasien

Zone 4: Australien, Karibik, Mittel- und Südamerika, Neuseeland

Zone 5: Afrika, Indien, Mongolei, Pakistan, Nordkorea, ehemalige UdSSR

Zone 6: Volksrepublik China

Untertitel

Wie bereits erwähnt, können neben dem Bild und Tonmaterial weitere 32 Datenströme, die insgesamt 3.36 Mbit Daten pro Sekunde enthalten könne, für unterschiedliche Untertitel verwendet werden. Dies bedeutet, daß maximal 53220 Bytes pro Frame zusätzlich bereitstehen.

Kopierschutz

Weil es in der Filmindustrie, der hauptsächliche Nutzer von DVD-Videos, um sehr viel Geld geht, wurde bei der Spezifikation von Anfang an die Unterstützung von Kopierschutzsystemen mit einbezogen, um das Kopieren von DVD-Videos nahezu unmöglich zu machen.

Zum einen findet das schon länger bekannte, analoge Macrovision System Verwendung. Dieses sorgt dafür, daß beim Kopieren von DVD-Videos auf VHS Systeme die automatische Aussteuerung des VHS-Rekorders systematisch gestört wird, sodass die Aufnahme größtenteils nur Flimmern und Flackern zeigt.

Desweiteren wollten die Entwickler des Standards auch verhindern, daß DVD-Videos auf digitalem Wege kopiert werden, z.B. durch ‘Grabben’ eines Videos mit einem Computer (auf die Harddisk) mit anschließendem Kopieren auf eine DVD-R oder DVD-RAM. Es gibt daher im, vom DVD-Player ausgegebenen Signal, auch ein ‘Serial Copy Generation Management System’ (CGMS), welches das Kopieren auf ganz ähnliche Weise verhindert, wie es beim SCMS von DAT-Rekordern der Fall ist.

Schließlich werden die Daten auf der DVD verschlüsselt gespeichert. Das verwendete System heißt ‘Content Scrambling System’ (CSS). Erst für die Wiedergabe wird die Entschlüsselung durchgeführt, nach einem Code, den die Gerätehersteller beim Lizenzgeber beziehen müssen. Auf eine genaue Beschreibung dieses Verfahrens soll hier verzichtet werden.

6 Digitale Videos in der Praxis

Nach der ausführlichen Betrachtung des MPEG-1 Standards und seiner Möglichkeiten soll nun die praktische Ebene der Erstellung digitaler Videos im Vordergrund stehen. Dabei werden zunächst die einzelnen Schritte auf dem Weg vom Videosignal zum digitalem Video betrachtet. Später wird dann als Beispiel ein Programm vorgestellt, mit dem der Schritt der Softwarekomprimierung durchgeführt werden kann.

Im Vortrag sollte der Einsatz dieser Software exemplarisch an einem Videoclip gezeigt werden. Dies war leider aus technischen Gründen unmöglich, sodass die Möglichkeiten des Programmes anhand von Screenshots vorgestellt wurden.

6.1 Digitalisierung eines Videosignals

Der Weg vom Videosignal zum digitalem Video führt in der Regel über die in Abbildung 6-1 skizzierten Stationen.

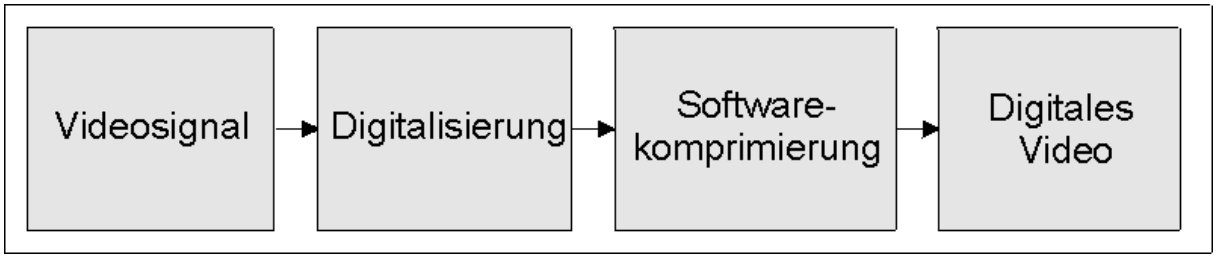


Abbildung 6-1 Verarbeitungskette

Dabei können bei jedem Übergang von Station zu Station, hinsichtlich der Qualität (Artefakte), Auflösung der Einzelbilder und Bildfrequenz Verluste in Videoclip auftreten. Da entstandene Verluste im Allgemeinen nicht wieder ausgeglichen werden können, ist die Summe der einzelnen Verluste gleich dem gesamten Verlust.

Das Endprodukt ist also in der Regel qualitativ etwas schlechter als das schwächste Glied dieser Verarbeitungskette.

Beispiel:

> Wenn der Digitalisierungsschritt im VHS-Standard arbeitet, kann auch ein S-VHS-Videosignal das Endprodukt nicht verbessern.

6.1.1 Videosignal

Das Videosignal liegt im privaten bzw. low-cost Bereich meist analog vor. Mit zunehmendem Preisverfall kommen hier aber auch Geräte im digitalem Standart zum Einsatz. Bei diesen Geräten muß das Videosignal nicht im konventionellem Sinn übertragen werden, da das Video schon in digitaler Form vorliegt. Dennoch muß auch bei diesen Geräten eine Verbindung zur Datenübertragung aufgebaut werden.

Im Bereich Videosignal können alle Geräte zum Einsatz kommen, welche in der Lage sind, ein zur Digitalisierungshardware kompatibles Signal zu liefern. Dabei sind im Besonderen Aufnahmegeräte (z.B. Videorekorder), aber auch Geräte mit einem Live-Signal (z.B. TV-Tuner) zu nennen.

6.1.2 Digitalisierung

In der in Abbildung 6-1 vorgestellten Verarbeitungskette umfaßt die Station *Digitalisierung* nicht nur die reine Digitalisierung, die Wandlung analoger Signale in digitale Daten. Vielmehr sind hier die Arbeitsschritte

- Akquirierung digitaler Daten und
- Speicherung dieser Daten

zusammengefaßt.

Bei der Akquirierung digitaler Daten unterscheiden sich die Arbeitsschritte je nachdem, ob das Videosignal analog oder digital vorliegt.

Bei einem digitalem *Videosignal* werden die Daten nur übertragen (vgl. 6.1.1). Bei einem analogem Videosignal muß dieses in Echtzeit abgetastet werden.

Da die anfallende Datenmengen heute übliche Rechnersysteme bei der Speicherung überfordern würde, wird von der beteiligten Hardware üblicherweise eine Komprimierung vorgenommen. Dabei handelt es sich in den meisten Fällen um eine MJPEG-Komprimierung.

6.1.3 Softwarekodierung

Da die von der Hardware vorgenommene MJPEG-Komprimierung in der Regel nur mit der gleichen Hardware dekomprimiert werden kann, liegt nach dem Digitalisierungsschritt ein Hardware abhängiger Videoclip vor. Um diesen Videoclip Hardware unabhängig zu machen folgt nun eine neu Komprimierung in eine Softwarekomprimierung. Die Softwarekomprimierung läuft im allgemeinen nicht in Echtzeit, sondern benötigt je nach Rechner und Komprimierungscode ein Vielfaches der Abspielzeit.

6.1.4 Digitales Video

Nach der Softwarekodierung liegt nun ein digitales Video vor, welches ohne die Komprimierungs-hardware abgespielt werden kann.

6.2 Praxis

Während des Vortrages wurde das Programm DVMPEG (Demoversion) von der Firma Darim Vision CO., Ltd. vorgestellt [DVMP 99]. Dieses Programm wurde ausgewählt, weil es eigentlich auf dem Vortragsrechner (Pentium II 266MHz + Windows NT) laufen sollte. Das Programm wurde von den Autoren dieser Ausarbeitung unter den Betriebssystemen Windows NT und Windows 95 (beide von der Firma Microsoft) getestet und ist durchaus lauffähig. Das eine Vorführung während des Vortrages nicht zustande kam lag eindeutig an der Konfiguration des Vorführrechners.

Der Vollständigkeit halber sei hier noch erwähnt, daß es eine große Auswahl an Softwareprodukten am Markt gibt, welche die hier beschriebenen Aufgaben lösen können. Besondere Aufmerksamkeit verdienen sich dabei die Produkte der MSSG [MSSG 99] welche zum einen frei erhältlich sind und zum anderen direkten Zugriff auf eine Fülle von Parametern erlauben.

6.2.1 Setup

Mit einem Digitalisierungsprogramm (hier Adobe Premiere) wurde eine Videosequenz digitalisiert und unter dem Namen *mjpeg.avi* abgespeichert. Diese Datei ist hardwareabhängig und kann nur mit der Digitalisierungshardware (hier miroDC 1) abgespielt werden. Um den Videoclip *mjpeg.avi* in eine hardwareunabhängiges Format (hier MPEG-1) zu wandeln wurde das Programmpaket DVMPEG

installiert. Dieses Programmpaket besteht aus vielen Programmen, von denen im Folgenden nur ein Teil betrachtet wird. Der beschriebene Programmteil der Software installiert sich als Komprimiermethode des Betriebssystems.

Im Prinzip wird die Datei *mjpeg.avi* in das Programm Adobe Premiere eingeladen und als *mpeg.avi* wider gespeichert. Während des Speichers legt DVMPEG eine *mpeg.avi* und eine *default* Datei an. Die Defaultdatei beinhaltet das konvertierte Video und kann nach Beendigung der Konvertierung umbenannt werden. Durch diese scheinbar recht umständliche Methode erhält der Benutzer die Möglichkeit bestehende Videobearbeitungsprogramme, welche nur mit *.avi-Dateien arbeiten zu nutzen.

6.2.2 Softwarekomprimierung

Bei der Speicherung der Dabei kann die Komprimierungsart für Video und Audio eingestellt werden. Bei der Einstellung der Komprimierungsmethode zeigt Adobe Premiere ein Fenster (vgl. Abbildung 6-2) in dem alle installierten Komprimierungsmethoden in einer Auswahlliste erscheinen. Nach Auswahl der Methode DVMPEG kann über den Knopf *Einstellen...* das Programm DVMEPG konfiguriert werden.

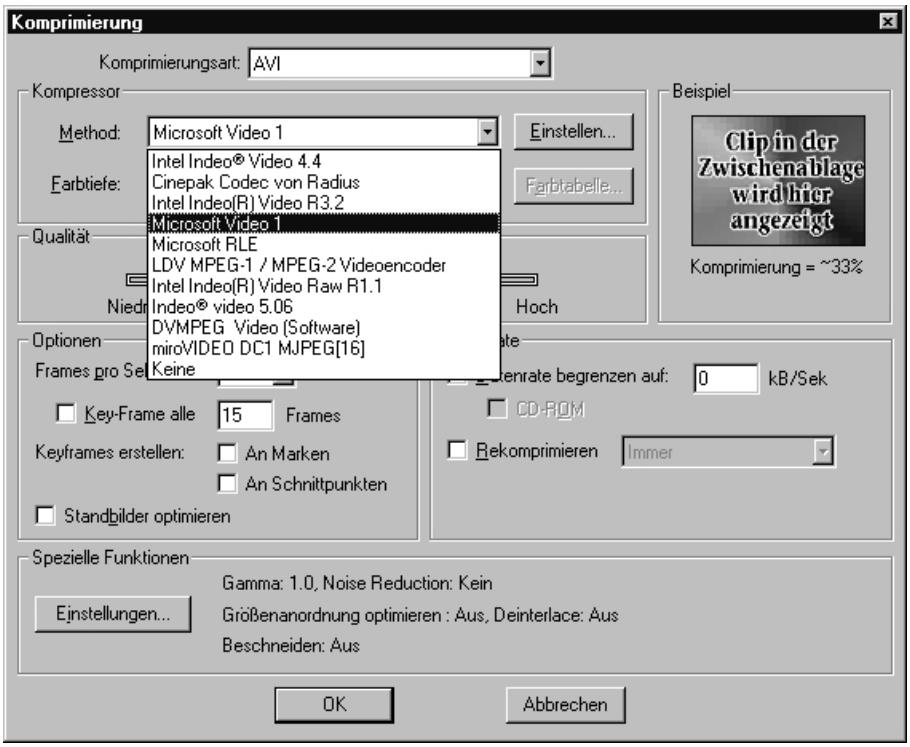


Abbildung 6-2: Komprimierungsmethoden

Die Konfiguration ist in die Abschnitte MPEG Base, MPEG Adv, MPEG Audio und Template unterteilt.

Im Abschnitt MPEG Base (vgl. Abbildung 6-3) kann der Streamtype, die Bitrate und der Standart festgelegt werden. Als Streamtype sind hier MPEG-1 und MPEG-2 besonders zu erwähnen.

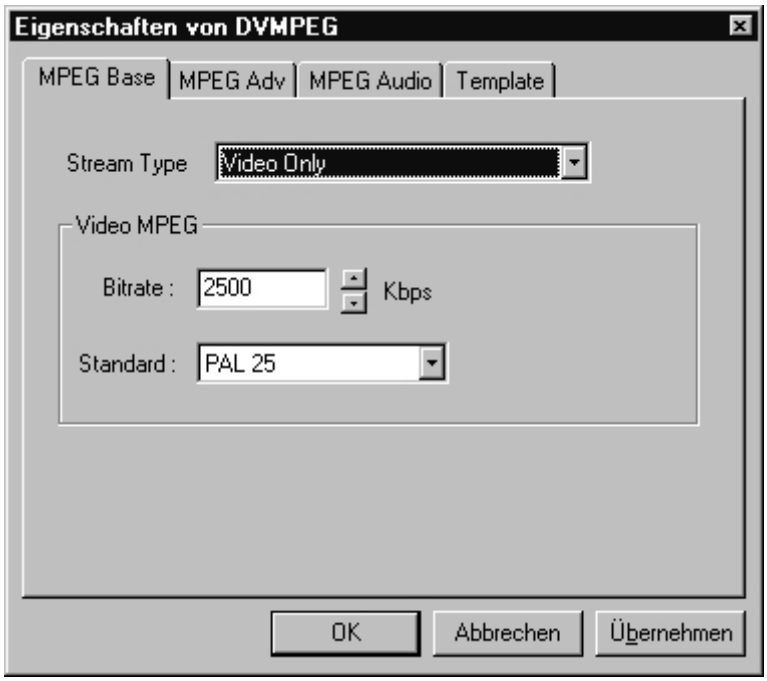


Abbildung 6-3: DVMPEG MPEG Base

Im Abschnitt MPEG Adv (vgl. Abbildung 6-4 und Abbildung 6-5) stehen dann je nach gewähltem Streamtype die Parameter des Streamtyps.

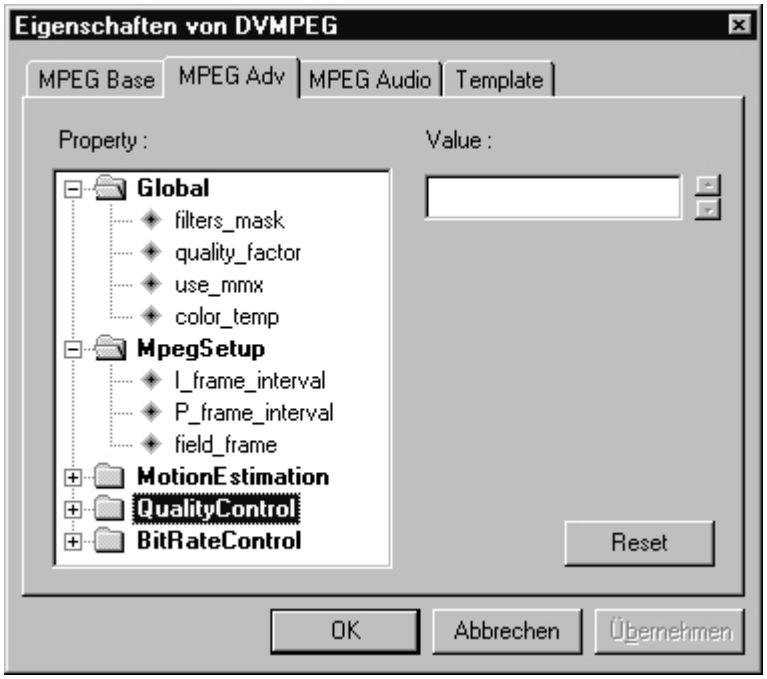


Abbildung 6-4: DVMPEG MPEG Adv 1

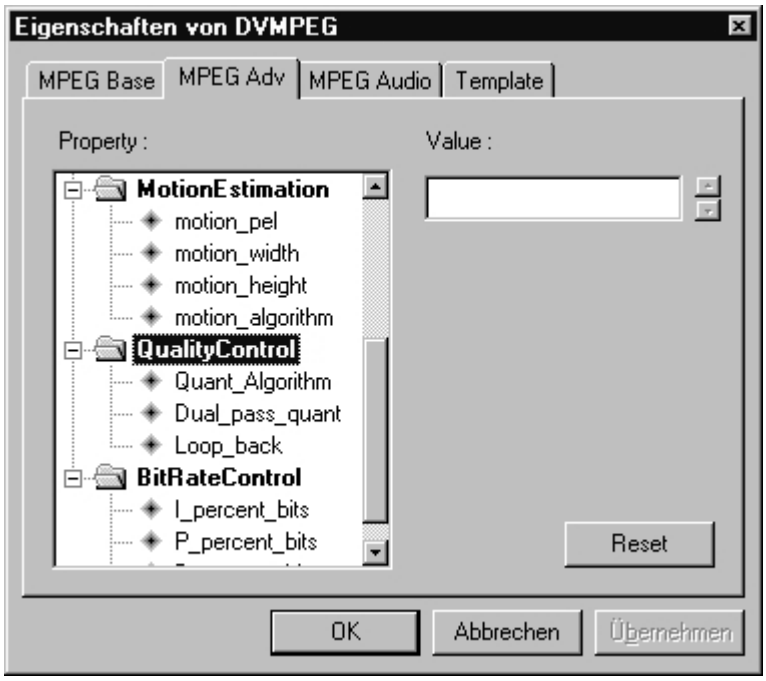


Abbildung 6-5: DVMPEG Adv 2

Im vierten Abschnitt (vgl. Abbildung 6-6) ist die Template-Verwaltung untergebracht. Hier können die Einstellungen der Parameter unter einem Namen dem Template abgespeichert werden.

Abbildung 6-6: DVMPEG Template

Anhang A

Abbildung A-1: Schloßmotiv QF=1

Abbildung A-5: Parkmotiv QF=1

Abbildung A-2: Schloß QF=6

Abbildung A-6: Parkmotiv QF=6

Abbildung A-3: Schloßmotiv QF=12

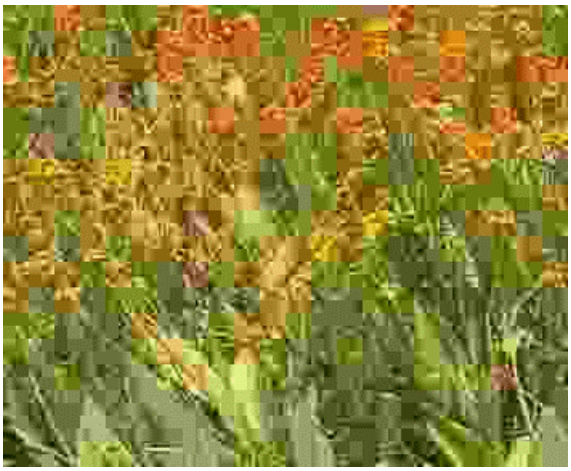


Abbildung A-7: Parkmotiv QF=12

Abbildung A-4: Schloßmotiv QF=20

Abbildung A-8: Parkmotiv QF=20

Literatur

[MuRy 96] James D. Murray, William Van Ryper:
"Encyclopedia of Graphics File Format"
Second Edition, O'Reilly & Associated INC., 1996

[WJPV 93] A.A. Webster, Coleen T. Jones, Margearet H. Pinson, Stephen D. Voran, Stephen Wolf:
"An objective video assessment system based on human perception."
Vol. 1913, Society of Photo-Optical Instrumentation Engineers, 1993

[Goar 99] Bret M. Goar:
http://www.cs.vwf.edu/~bgoar/bgcap4401/Program02/assignment02.htm [Stand:14.05.1999]

[DVDF 98] DVD-Forum Deutschland:
http://home.t-online.de/home/bieser/dvd.htm [Stand: 14.05.1999]

[MSSG 99] MPEG Software Simulation Grupe
http://www.mpeg1.de [Stand: 01.04.1999]
http://www.mpeg2.de [Stand: 01.04.1999]

[DVMP 99] Darim Vision CO.,Ltd.
http://www.darvision.com [Stand: 10.05.1999]

DV Formate

Markus Bajohr

FB Informatik, Uni Dortmund
bajohr@emm-ell.net

Zusammenfassung

Video begann ab 1990 digital zu werden. Zu Anfang waren digitale Videosysteme für den Privatanwender noch unbezahlbar, jedoch brachte Sony 1995 die DCR-VX700, eine DV Kamera, für den Videoamateur auf den Markt. Weitere Firmen zogen nach und es begann ein Gerangel um die verschiedenen Aufnahme Formate. Ich möchte mit diesem „Paper“ die verschiedenen DV Formate, sowie die Funktionsweise von DV näher beleuchten.

1 Was ist DV?

DV ist ein internationaler Standard, der von 10 großen Industrien gegründet wurde.
Diese sind:

Matsushita Electric Industrial Corp (Panasonic)
Sony Corp
Victor Corporation of Japan (JVC)
Philips Electronics, N.V.
Sanyo Electric Co. Ltd
Hitachi, Ltd.
Sharp Corporation
Thompson Multimedia
Mitsubishi Electric Corporation
Toshiba Corporation

Im September ’95 begann die Videotechnik für den Privatanwender digital zu werden. Sony baute 2 digitale Camcorder, den DCR-VX1000 als 3 Chip Kamera und den DCR-VX700 als Einchip Variante. Eine direkte digitale Datenübertragung konnte mit Hilfe des FireWire IEEE-1394 bewerkstelligt werden.

Wenige Wochen später erschien auch Panasonic auf dem DV-Camcorder Markt. Jedoch hatte Panasonic versäumt eine direkte Datenübertragung via IEEE-1394 einzubauen.

JVC und Sony bauten 1996 zwei relativ kleine digitale Camcorder. Sonys Gerät, der DCR-PC7, war mit einem FireWire Anschluß und einem ausklappbaren 2.5‘‘ LCD versehen. 1997 brachte Panasonic 2 neue Camcorder auf dem Markt, die jetzt auch über einen FireWire Anschluß verfügten. Andere Hersteller, wie JVC und Sharp, boten das FireWire Interface noch nicht an.

Es haben sich schließlich 3 professionelle Videoformate in dem Wust von Formaten durchgesetzt. Pnansonic hat das DVCPRO Format entworfen, welches die gleiche Audio und Video Kodierung wie das consumer DV verwendet. Sonys Kodierungsverfahren, DVCAM, lehnt sich ebenfalls an das DV Format an. Jedoch sind diese 3 Formate nicht untereinander kompatibel.

DV wurde ursprünglich durch das Kassettenformat DVC (Digital Video Cassette) bekannt. Um die hohe Qualität von DV zu erreichen, wird ein 1/4 inch (6.35mm) Metallband verwendet. Das Video wird in einer Rate, ähnlich D-1, D-5 oder Digital Betacam Video gesampled. Jedoch wird die Farbinformation nur in der halben Rate von D-1 gesampled. Dies sind dann 4:1:1 bei 525 Zeilen (NTSC) und 4:2:0 bei 625 Zeilen (PAL). In einer Scanline befinden sich 720 Pixel.

Nach dem Digitalisieren wird das Video mit Hilfe der Diskreten Cosinus Transformation (DCT) und einer anschließenden Gewichtung komprimiert. DCT erlaubt eine lokale Optimierung innerhalb eines Bildes. Dieses Verfahren wird auch MJPEG genannt.

DV benutzt die Intraframe Kompression. Jedes Bild steht für sich allein und benötigt keine weiteren Informationen der vorherigen Bilder. Falls der Kompressor jedoch nur geringe Differenzen zwischen den beiden Halbbildern entdeckt, werden diese zusammen komprimiert. Dieses wird auch Interfield Compression gennant. Theoretisch ergibt sich dadurch eine saubere Digitalisierung für Statische Objekte als Bewegte.

DV Video Daten werden in einem 25 Mbps Datenstrom transportiert. Zudem muß noch Audio, subcode (including timecode), Insert and Track Information (ITI), und error correction hinzugefügt werden, so daß der Datenstrom auf 36 Mbps anwächst.

2 PIX Sampling

Es gibt verschiedene Notationen für die Sampling Struktur von DV. Es werden neben SDTV (standard definition TV), festgelegt nach ITU-R BT.601 mit 13.5 MHz Sampling Frequenz und 720 Pixel pro Linie, auch Definitionen nach CIF und QSIF verwendet. Ich werde mich im Folgenden auf den SDTV Standard beziehen.

Die Sampling Angaben erfolgen immer in einem Tripel von Zahlen, wovon die erste Zahl die Helligkeitsabtastungen und die anderen beiden die Farbabtastungen für Rot und Blau angeben.

Der Speicherbedarf pro Pixel läßt sich folgendermaßen bestimmen:

$$\frac{bits}{Pixel} = Farbtiefe * (1 + \frac{Wert2}{Wert1} + \frac{Wert3}{Wert1})$$

wobei *Farbtiefe:* Anzahl Bits für die Helligkeit (nach ITU-R BT.601: 8 Bit)

Zum Vergleich benötigt ein Bild, welches mit 4:4:4 720 * 480 Pixel gecodet wird, den folgenden Speicherbedarf:

$$8\,bit * (1 + \frac{4}{4} + \frac{4}{4}) * 720\,Pixel * 480\,Pixel = 1.036.800\,byte \approx 1\,MByte$$

Bei diesem Verfahren handelt es sich um ein unkomprimiertes Bild, welches einem RGB Bild in TrueColor (=24 Bit) entspricht.

2.1 4:2:2 Format

In 4:2:2 System, wie D-1, D-5, DigiBeta, BetaSX, Digital-S, DVCPRO50, wird die Farbe in der halben Rate des Helligkeitssignals gesampled. Die Rot und Blau Farbinformation wird getrennt gespeichert. Bei einer Scanline von 720 Pixels ergeben sich somit 360 Farbabtastungen für jeweils Blau und Rot.

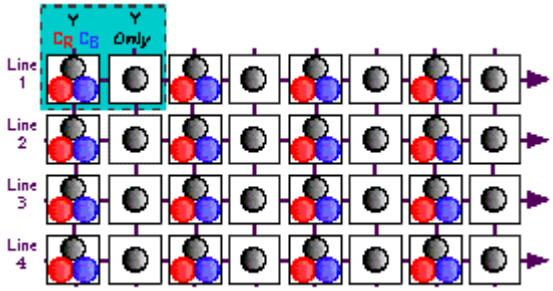


Abb. 2.1: 4:2:2 Format

2.2 4:1:1 Format

Systeme die nach der 4:1:1 Norm arbeiten, samplen die Farbdaten in der halben Rate des 4:2:2 Verfahrens. Es ergeben sich somit 180 Farbabtastungen pro Scanline für die Farbe Rot und Blau. Die Rot- und Blauwerte werden zusammen an jeder vierten Position des Helligkeitssignals abgetastet. Diese geringe Abtastrate reicht aus, da die Farbinformation ca. 1.5 MHz Bandbreite benötigt. Die NTSC DV, DVCAM und DVCPRO arbeiten nach dem 4:1:1 Verfahren.

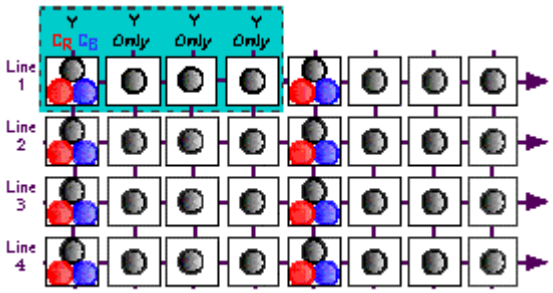


Abb. 2.2: 4:1:1 Format

2.3 4:2:0 Format

Neben diesen Normen gibt es noch das 4:2:0 Verfahren, welches von PAL DV, DVD, main-profile MPEG-2 angewandt wird. Sicherlich stellt sich nun die Frage, ob bei diesem Sampling eine Farbe gar nicht abgetastet wird, da eine Zahl des Tripels Null ist. Jedoch steht diese Bezeichnung dafür, daß die Farbinformation in der halben Rate des Helligkeitssignals in horizontaler und vertikaler Richtung abgetastet wird. Technisch ist dies durch eine Art Zeilensprung gelöst. Es wird in jeder zweiten Scanline 360 Farbinformationen für jeweils Rot und Blau gespeichert. In den ungeraden Zeilen wird keine Farbinformation gesampled. Eine weitere Verbesserung läßt sich erreichen, wenn in den geraden Zeilen jeweils 360 Samples für den Blauwert und in den Ungeraden 360 Samples für den Rotwert abgelegt werden. Dieses entspricht aber nicht der Norm und wird co-sited genannt!

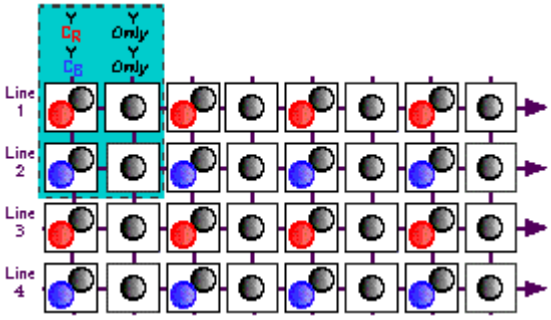


Abb. 2.3: 4:2:0 Format (co-sited)

3 PIX Artefakte

Bei dem PIX Sampling gibt es 3 verschieden Probleme:

a) Mosquito Noise
b) Quilting
c) Motion Blocking

Das Mosquito Rauschen entsteht durch Hochfrequente Bildanteile, die durch die Komprimierung/Dekomprimierung verloren gehen. Sichtbar ist dieses nur innerhalb einer 8 Pixel Region an „harten Rändern“ des Bildinhaltes, wie z.B. Schrift. Dieser Effekt tritt in allen DCT orientierten Kompressionsverfahren, wie DV, JPEG oder MJPEG auf. Als Beispiel dient ein ungefiltertes Bild, welches mit dem Titelgenerator von Premiere 4.2 erstellt wurde. In der Vergrößerung, 72 * 48 Pixel, erkennt man die dunkleren Flecken auf der hellen Schrift und den zerstörten Hintergrundschatten.

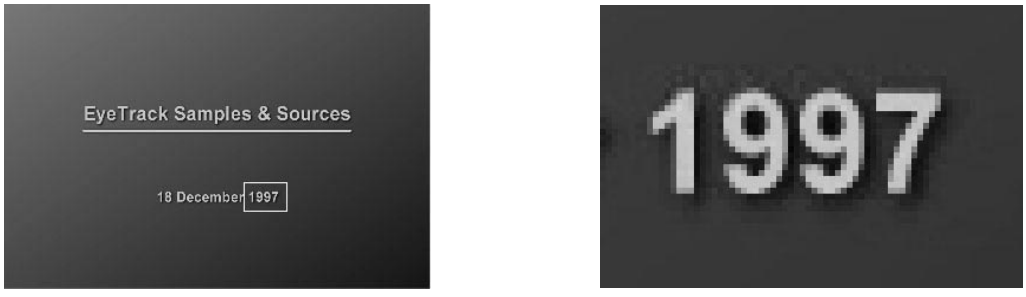


Abb. 3.1: Mosquito Rauschen, rechts Detailaufnahme

Quilting ist die Diskontinuität benachbarter DCT Blöcke innerhalb eines Bildes. Am deutlichsten wird dies bei leicht diagonalen Linien, da diese durch die Auflösung der Kamera als Treppeneffekt dargestellt werden. Bei DV Kameras lassen diese Effekte bei langsamen Kamerachwenks zeigen. Während des Schwenks sind dann Regionen innerhalb eines Bildes verzerrt oder versetzt zum vorherigen Bild. Durch die Veränderung der Schärfe an dem Monitor lassen sich diese Artefakte noch weiter verdeutlichen.

In diesem Beispiel kann man das Quilting an der Ecke des Tisches in der Vergrößerung, 72 * 48 Pixel, erkennen. Die kleinen Pfeile am unteren Bildrand zeigen die 8 Pixel Regionen innerhalb des Bildes an.

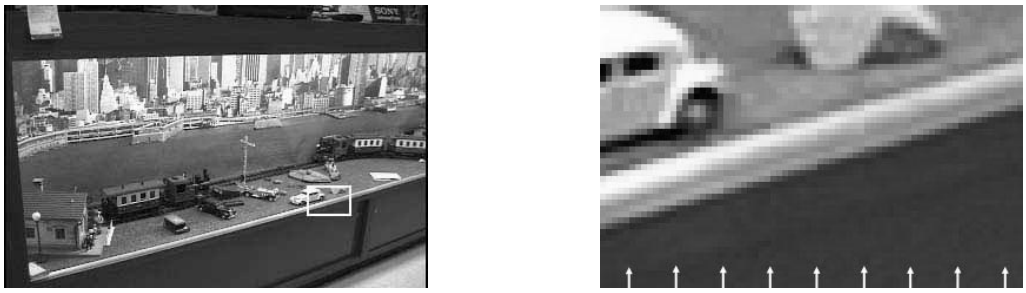


Abb. 3.2: Quilting, rechts Nahaufnahme

Motion Blocking verzerrt kleinere Details eines Bildes bei schnellen Bewegungen. Dieses Artefakt ist auf den DV codecs, die 2 Felder getrennt kodieren, zurückzuführen. In diesem Bild kann man den Detailverlust in der Mitte der Straße und an den Autos erkennen. Interessant ist allerdings, daß nach dem Vorbeifahren der Autos, die Straßenmitte, sowie das hintere Efeu „sauberer“ dargestellt werden.

Abb. 3.3: Motion Blocking

4 PIX Sampling in der Praxis

Als Beispiel dient ein Bild (Abb. 4.1), welches nach 4:1:1 mit einer VX1000 Kamera im 525/29,94 NTSC DV Format aufgenommen wurde. Die 72 * 48 Pixel Region innerhalb des Rechtecks zeigt die grobkörnige Farbabtastung in horizontaler Richtung, da nach jeder vierten Helligkeitsabtastung zwei Farbwerte folgen. Somit haben 4 Pixel immer die gleichen Farbwerte, wie das erste Pixel einer 4 Pixelgruppe. 4 Pixel unterscheiden sich nur in der Helligkeit. Da dieses Bild aber mit der 4:1:1 co-sited Norm abgetastet wurde, ergibt sich ein Farbunterschied in jeweils 2 Pixeln. Eine co-sited Norm besagt, daß die beiden Farbwerte YCr und YCb nicht zusammen bei einem Pixel abgetastet werden, sondern getrennt. Somit wird bei jedem zweiten Pixel ein Farbwert im Wechsel abgetastet.

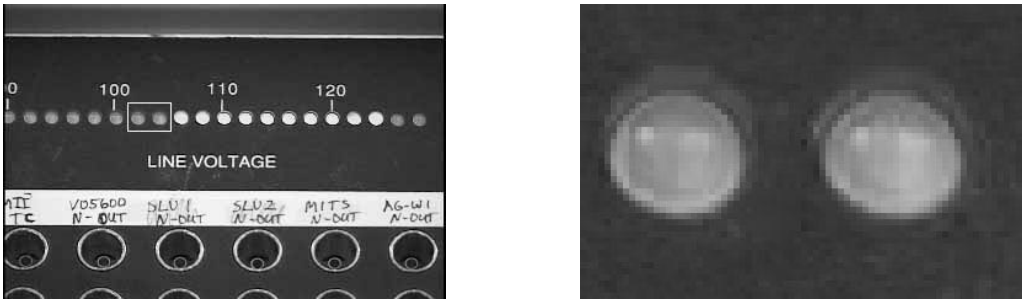


Abb. 4.1: 4:1:1 digitalisiertes Bild, rechts Detailaufnahme

Interessant ist, daß dieser Effekt bei den gelben oder grünen LED nicht so markant ist, wie bei den roten LEDs. Zudem muß man berücksichtigen, daß diese Effekte ohne Vergrößerung kaum ins Auge fallen und die Farbschärfe vergleichbar mit BetacamSP ist. Eine Verbesserung der Farbschärfe wird durch das 4:2:2 Format, welches in der Digital-S oder DVCPRO50 verwendet wird, erreicht.

In einem weiteren Bild von Karel Holubièka (Abb. 4.2), welches dem 4:2:0 Format gehorcht, kann man eine wesentliche Verschlechterung in der Farbdigitalisierung feststellen. Dieses Bild zeigt 2 rote LEDs in einer 45 * 30 Pixel Region, aufgenommen mit einer DCR-VX1000 im 625/50 Modus.

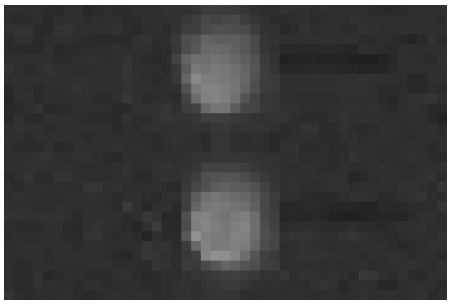


Abb. 4.2: 4:2:0 digitalisiertes Bild

5 Allgemeines DV Format

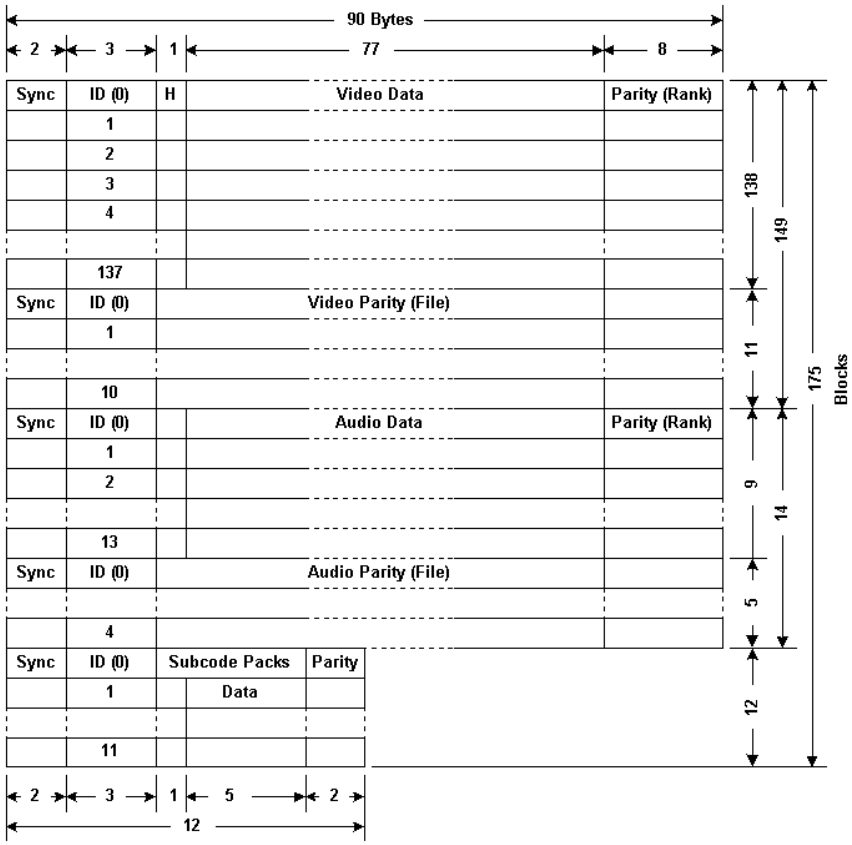


Abb. 5.1 : Video, Audio und Subcode Aufnahme Format

Abbildung 5.1 zeigt das Aufnahmeformat eines Bildes im NTSC-Format mit video, audio, und subcode data. Ein komprimiertes Bild besteht aus 10 tracks mit jeweils 138 data blocks. Jeder Data block umfaßt 76 bytes an Video Nutzdaten und einen 1 byte Header.

Die Datenrate läßt sich somit folgendermaßen berechnen:

$$10\frac{tracks}{frame} * 138\,blocks * 76\,bytes * 8\frac{bits}{byte} * 29{,}97\frac{frames}{\sec} = 25{,}146\,Mbps$$

Es werden statt 30 *frames/sec* immer nur 29,97 *frames/sec* berücksichtigt, da bei 1000 *frames* ein Bild nicht verwendet werden kann.

$$\frac{999\ frames}{1000\ frames} * 30\frac{frames}{\sec} = 29{,}97\frac{frames}{\sec}$$

Die theoretische Datenrate nach 8-bit ITU-R BT.601 mit non-standard 4:1:1 Kodierung (720 Bildpunkte pro Zeile in 8 bit Helligkeit und 4 bit Farbe, 485 Zeilen mit 29,97 frames/sec) beträgt 125,6 Mbps. Dieses läßt sich, wie folgt berechnen:

$$8\,bits * (1 + \frac{1}{4} + \frac{1}{4}) * 720\ Pixel * 485\ Pixel * 29{,}97\frac{frames}{\sec} = 125{,}586\,Mbps$$

Das Kompressionsverhältnis entspricht:

$$\frac{125,6}{25,146} = 5:1$$

Wird das Kompressionsverhältnis nach BT.601's standard 4:2:2 berechnet, so ergibt sich:

$$\frac{167,5}{25,146} = 6,6:1$$

Die Audioinformationen werden in 9 blocks zu 76 bytes gespeichert. Die max. Audiodatenrate entspricht somit:

$$10\frac{tracks}{frame} * 9\frac{blocks}{track} * 76\frac{bytes}{block} * 8\frac{bits}{byte} * 29,97\frac{frames}{\sec} = 1,64\ Mbps$$

Im Vergleich ergibt sich für 4 Spuren in 32 kHz zu 12-bit, die max. Datenrate:

$$4\ channels * \frac{32}{1000} MHz * 12\ bits = 1,536\ Mbps$$

Wird hingegen mit einer 2 Spur (Stereo) Aufnahme gearbeitet, so läßt sich bei gleicher Datenrate errechnen:

$$2\ channels * \frac{48}{1000} MHz * 16\ bits = 1,536\ Mbps$$

Die Gesamtdatenrate (Audio & Video), inklusiv Parity, aber ohne ITI sector beträgt:

$$10\frac{tracks}{frame} * ((90\frac{bytes}{block} * 163\ blocks) + (12\frac{bytes}{block} * 12 blocks)) * 8\frac{bits}{byte} * 29,97\frac{frames}{\sec} = 35,5\ Mbps$$

Der enstehende Overhead läßt sich bestimmen nach:

$$35,5\ Mbps - 25,15\ Mbps - 1,64\ Mbps = 8,7\ Mbps$$

$$\frac{8,7}{35,5} * 100 = 25\%$$

25 % der aufgezeichneten Daten ergibt sich für subcode data, error detection, und error correction!

5.1 Fehlererkennung, Fehlerkorrektur

Die Video und Audio Daten werden mit einer Kreuzparität versehen, d.h. es wird eine Parität in einer Zeile und einer Spalte gebildet. Das Paritätsbit wird auf 1 gesetzt, sobald eine ungerade Zahl an 1 bits in einer Zeile oder Spalte auftritt. Es handelt sich also um eine gerade Paritätsbildung.

Falls es zu einem „1 bit-Fehler“ während der Aufnahme oder Wiedergabe kommt, läßt sich dieser korrigieren, siehe dazu Abb. 5.2. Eine Korrektur ist möglich, da die innere und äußere Parität gebildet werden. Das fehlerhafte Bit liegt auf dem Schnittpunkt der fehlerhaften Zeile und Spalte.

Jedoch kann dieses Verfahren nur „1 bit-Fehler“ zuverlässig korrigieren. Verbesserte Verfahren arbeiten mit polynomellen Erweiterungen für größere Datenblöcke.

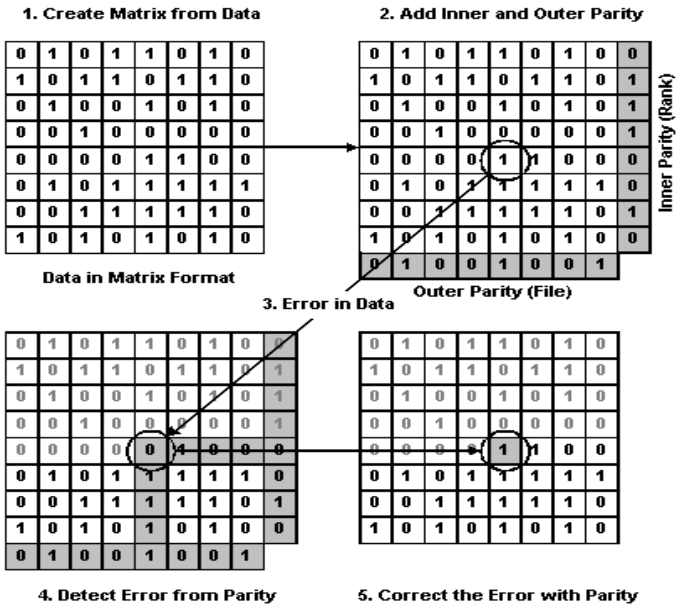


Abb. 5.2 : Vereinfachte Darstellung einer Error Detection mit Parity bits

DV nutzt den Reed-Solomon (RS) error detection / correction Code. RS kann lokale Fehler korrigieren, aber nur selten Daten eines Dropouts rekonstruieren. Bei burst errors versagt allerdings dieses Verfahren.

Weitere Techniken, wie das error concealment, sind nicht von der Spezifikation vorgegeben. Error concealment arbeitet mit einer Schätzung der verlorenen Daten anhand der vergangenen und folgenden Bildern. Meist werden proprietäre Methoden angewandt, die in den DV Geräten implementiert sind. Damit diese Techniken zufriedenstellend funktionieren, wird eine große Menge an Arbeitsspeicher (RAM) benötigt, um die Informationen zwischenzuspeichern.

6 Digital Interface (DIF) Format

Die Anordnung der digitalen Daten weicht von dem Aufzeichnungsformat aus Abb. 5.1 ab, da dieses eine error correction umfaßt. Zur Datenübertragung wird keine error correction, sondern eine error detection benötigt, so daß im Fehlerfall ein Datenpaket erneut gesendet wird. Der Vorteil von error detection liegt in der einfacheren Implementierung.

In Abb. 6.1 sieht man das Datenformat zur Übertragung via IEEE 1394-1995. Daten werden in gleich lange **d**ata **i**n **f**rame Sequenzen (DIF) gepackt. Ein NTSC Bild mit 720 x 480 Bildpunkten wird in 10 DIF Sequenzen, mit jeweils 12000 bytes Inhalt, aufgeteilt. Eine DIF Sequenz enthält 5 Super blocks für die Video pixel data.

DIF Sequenzen bestehen aus 150 DIF blocks zu je 80 bytes (20 quadlets). Dieses ergibt 12000 bytes pro DIF Sequenz (150 DIF blocks * 80 bytes).

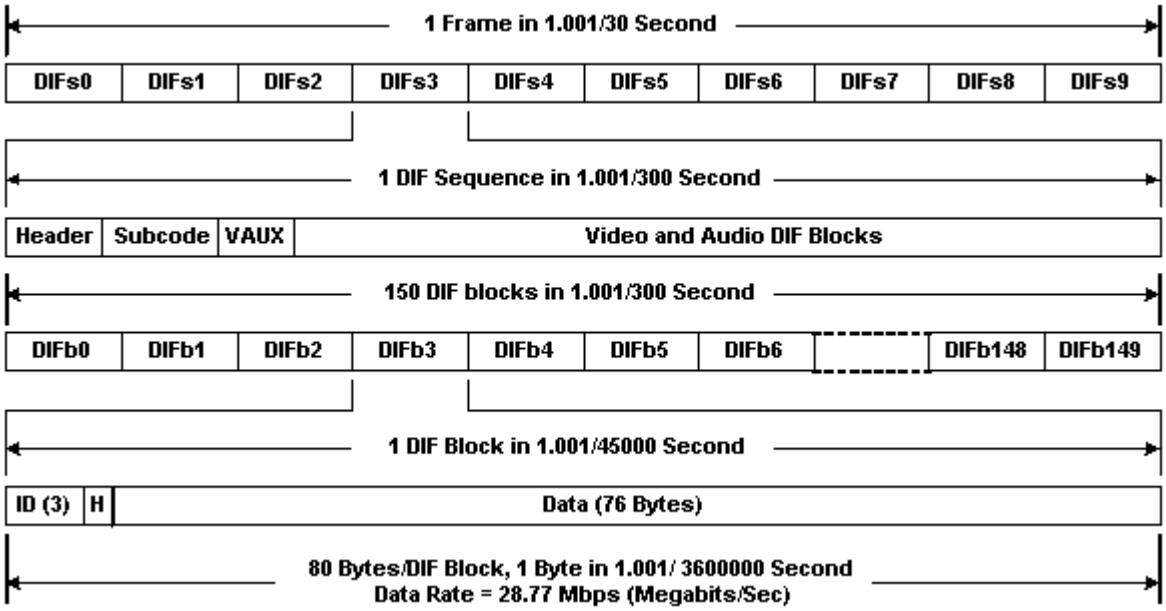


Abb. 6.1: Anordnung von DIF Sequenzen

Eine DIF Sequenz ist folgendermaßen aufgebaut:

- 135 DIF blocks für Video Daten
- 9 DIF blocks für Audio Daten
- 6 DIF blocks für Header, Subcode und Video Auxiliary (VAUX)

Die Audio und Videodaten werden nach Abb. 6.1 in den DIF blocks organisiert. Die damit entstehende DIF Datenrate ist:

$$10\frac{Sequenzen}{Bild} * 150\frac{blocks}{Sequenz} * 80\frac{bytes}{block} * 8\frac{bits}{byte} * 29{,}97\frac{Bilder}{Sekunde} = 28{,}77 Mbps \approx 3{,}6\frac{Mbyte}{Sekunde}$$

Abb. 6.2 zeigt die Anordnung der DIF blocks innerhalb einer DIF Sequenz. 9 Audio DIF blocks werden interleaved mit 135 Video blocks in einer 9- zu 14- block Matrix vermischt. Zu Beginn wird ein 6 Block langer Header mit Subcode und VAUX übertragen.

Komprimierte Makroblocks werden während des Aufnahmeprozesses in den Datenstrom eingemischt, um eine Fehlerfortpflanzung in den nachfolgenden Bildern zu vermeiden. Durch diese Methode wird ein error concealment effektiv unterstützt

Die Audiodaten werden ebenfalls in den Datenstrom eingemischt. Bei der Wiedergabe werden die Daten in der Aufnahmereihenfolge wiedergegeben.

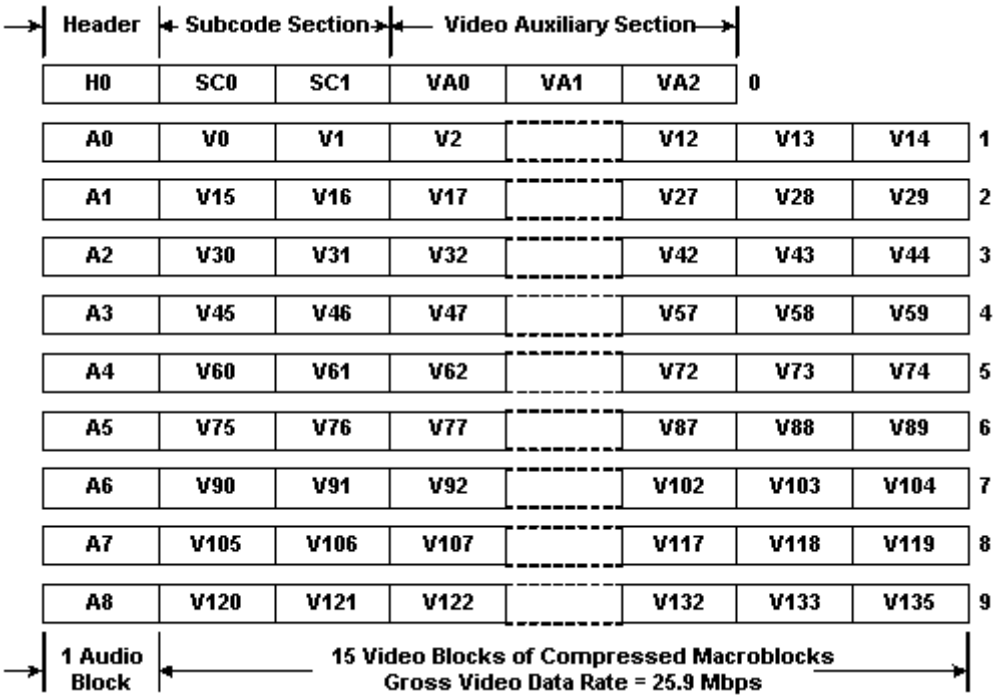


Abb. 6.2: Übertragungssequenz von Data, Video und Audio DIF Blocks

DV Video frames sind in 270 individuellen Video Segmenten zusammengefaßt. Pro DIF Sequenz ergibt dies 27 Video Segmente, da ein frame aus 10 DIF Sequenzen besteht. Ein Video Segment ist aus 5 komprimierten Makroblocks zusammengesetzt, die jeweils 80 bytes lang sind. Video Segmente beinhalten nur die Videodaten und nicht die Audio bzw. VAUX Informationen. Innerhalb einer DIF Sequenz werden 135 Blocks für die Videodaten benötigt (5 Makroblocks * 27 Video Segmente).

Ein Makroblock enthält die folgenden Informationen:

- 3 bytes für die DIF block ID information
- 14 bytes jeweils für die Helligkeitsinformationen Y0, Y1, Y2, Y3 (= 56 bytes)
- 10 bytes jweils für die Farbinformation Cr und Cb (= 20 bytes)
- 1 byte als quantization number (QNO) und block status (STA)

Jeder Makroblock beinhaltet eine 32 x 8 Pixel Region, die von jeweils 5 Spalten eines Video frames erzeugt wird.

Neben den Makroblocks gibt es noch Superblocks. Ein Superblock ist eine logische Einheit von 27 Makroblocks. In einem NTSC DV Video frame gibt es 50 Superblocks.

$$27\frac{Makroblocks}{Superblock}*50\ Superblocks = 27\ Videosequenzen*5\frac{Makroblocks}{Videosequenz}*10\frac{frames}{Sequenz} = 1350\ Videoblocks$$

Eine Gruppe von 5 Superblocks, einer von jeder Superblock Spalte, ergibt eine DIF Sequenz.

6.1 Dekodierung eines Video Segments

Ein Video Segment besteht aus 5 komprimierten Makroblocks. Die Dekodierung der AC Koeffizienten erfolgt mit Hilfe eines 3 schrittigen variable length decoding Algorithmus.

Schritt 1:

- dekodiere VLC AC Koeffizienten für Y0, Y1, Y2, Y3, Cr und Cb innerhalb eines Makroblocks

Schritt 2:

- dekodiere übergelaufene VLC AC Koeffizienten innerhalb eines Makroblocks

Schritt 3:

- dekodiere übergelaufene VLC AC Koeffizienten innerhalb eines Video Segments

Nachdem nun die AC Koeffizienten bestimmt wurden, sind folgende Schritte durchzuführen:

- Inverse quantization
- „Zigzag“ Koeffizienten Sortierung
- Inverse Gewichtung
- Inverse diskrete Cosinus Transformation (DCT) nach 8-8 oder 2-4-8 (2-4-8 wird benötigt, wenn sich viele Details im Bild befinden.)
- Speichere die Pixeldaten in der richtigen Reihenfolge innerhalb des Video frames

Des weiteren müssen folgende Randbedingungen eingehalten werden:

- Alle 3 Video Segmente muß der Audio DIF Block übersprungen werden.
- Nach jeweils 27 Video Segmenten muß der Header, Subcode und VAUX übersprungen werden. Dieses sind 6 DIF blocks.
- Dieses wird nun 10 Mal pro Bild durchgeführt. Als Ergebnis erhält man ein YUV kodiertes 4:1:1 Bild mit einer Auflösung von 720 x 480 Bildpunkten.

7 DV Kassetten und Aufzeichnungsformat

Um die digitalen Videodaten aufzuzeichnen haben sich 2 Kassettenformate durchgesetzt. Dies ist zum einen die Standard DV Kassette mit den Abmessungen 125 mm x 78 mm x 14,6 mm und einer Aufnahmekapazität von bis zu 4,5 Stunden. Zum anderen wurde eine verkleinerte Version der Standard Kassette, mit den Abmessungen 56 mm x 49 mm x 12,2 mm, entworfen. Sie findet ihren Einsatz hauptsächlich in DV Camcordern und bietet eine Aufnahmekapazität von max. 1 Sunde.

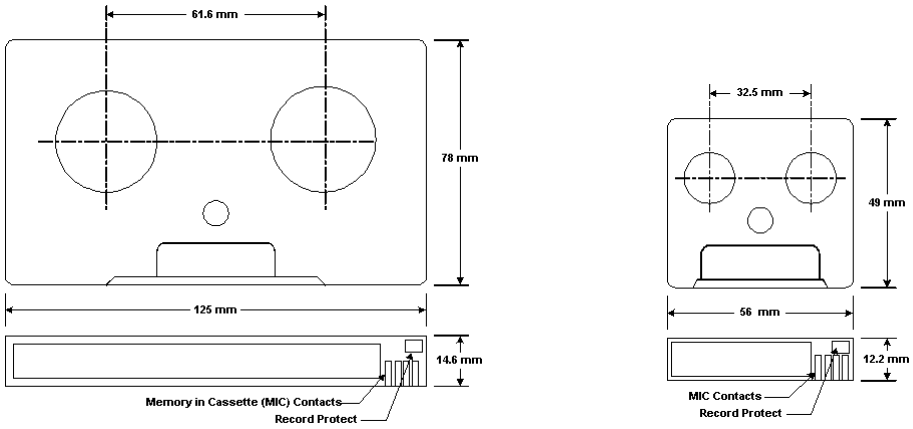


Abb. 7.1 Standard DV (links) und Mini DV (rechts)

Das Videoband hat eine Breite von 6,35 mm (1/4 inch) und wird mit einer Geschwindigkeit von 18,81 mm/sec im Normal Aufnahme/Wiedergabe Modus fortbewegt. Eine MiniDV Kassette (1 h) hat eine Bandlänge von 65 m und eine Standard DV Kassette beherbergt 250 m Band. Die MiniDV Kassette kann ähnlich wie beim VHS-C mit einem Adapter auf einem Standard DV Videorecorder abgespielt werden.

Interessant ist der interne Speicher, MIC (Memory in Cassette), der in jeder Kassette integriert ist und von außen (unten rechts) kontaktiert wird. Bei diesem Speicher handelt es sich um einen nichtflüchtigen Speicher, so daß auch nach dem Herausnehmen der Kassette der Inhalt erhalten bleibt. Laut Spezifikation darf der Speicher eine Kapazität von max. 16 Mbyte haben. Sony DV Kassetten haben ein 512 Byte großes MIC, das Daten über die Kassette, sowie Datum und Uhrzeit von verschiedenen Aufnahmeszenen speichert. Die MIC Daten werden von den Geräten intern genutzt und auch über das digitale Interface übertragen. Zur Zeit nutzt nur Sony den MIC Speicher, der für die Kassettenkompatiblität nicht erforderlich ist.

Sony DV Kassetten bestehen aus 5 Schichten:

- Back coating, um die Reibung des Bandes an den Führungsrollen zu minimieren
- Base film, welches als Trägerschicht dient
- Double metal-evaporated Magnetschicht
- Evaporated carbon overcoat Schicht, die die magnetischen Informationen schützt
- Surface preparation Schicht, um die Reibung des Bandes an der Kopftrommel zu reduzieren

Hi8 Metallbänder haben im Gegensatz zu den DV Bändern große Probleme mit Dropouts, sowohl neue als auch ältere Kassetten. Selbst häufig abgespielte DV Kassetten haben die gleiche hohe Datensicherheit, wie zu Anfang.

7.1 Geometrische Anordnung der Spuren

DV Aufnahmesysteme nutzen ein azimuth recording Verfahren (+/- 10 Grad), wofür mindestens 2 rotierende Köpfe, die auf einer Kopftrommel angeordnet sind, erforderlich sind. Die Rotationsgeschwindigkeit für PAL, NTSC oder SECAM beträgt 9000 rpm. Die Spurbreite ist nur 10 microns (Millionstel eines Meters) breit. Das Hi8 System hat eine Spurbreite von 20,5 microns und VHS sogar 58 microns! Zum Vergleich hat ein Menschenhaar eine Dicke von 100 microns. Die Spuren sind um 9 Grad geneigt, so daß sich eine Spurlänge von 35 mm ergibt.

Die genutzte Länge liegt bei 33 mm, da der Randbereich des Bandes zu Fehlern neigt. DV benötigt keine Kontrollspur, wie beim VHS System, sondern mischt Pilottöne zur Bandgeschwindigkeitsmessung in den Datenbereich. Diese werden mit einem Sollwert verglichen, so daß die Bandgeschwindigkeit laufend angeglichen werden kann.

Des weiteren sieht die Spezifikation 2 horizontale Spuren an beiden Seiten des Bandes vor. Diese können Herstellerspezifisch genutzt werden. Bei der DVCPRO werden diese Spuren für den Timecode (LTC) und ein audio cuing während des Vor oder Rückspulens verwendet. Neben dem LTC wird auch noch ein vertikaler Timecode, VITC, zur Abwärtskompatiblität geschrieben.

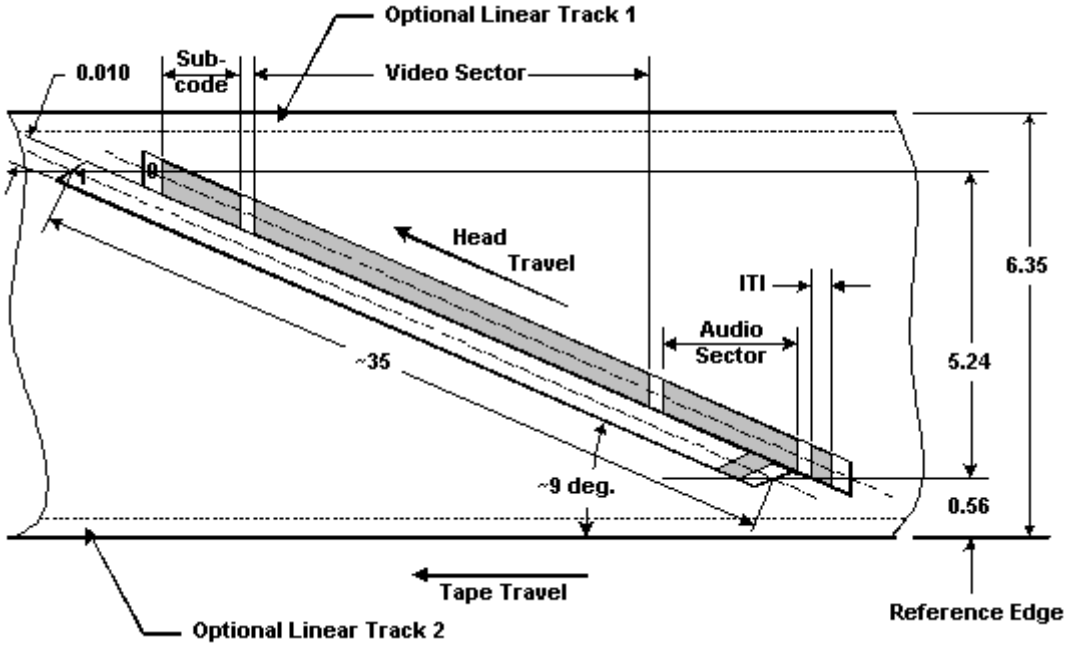


Abb. 7.2 : Geometrische Spuranordnung auf dem DV Band

Um nun den beiden bedeutsamen Fernsehnormen, NTSC und PAL, gerecht zu werden, kann die Anzahl an Tracks pro Bild durch das DV Format geändert werden. NTSC, 525/60, benötigt 10 Tracks pro Bild, wohingegen beim PAL, 625/50, 12 Tracks pro Bild erforderlich sind. Dieses erklärt auch die Anzahl der Störstreifen bei DV Aussetzern, siehe dazu Kapitel 8. In der Abbildung 7.3 sieht man die Aufzeichnung eines Bildes auf dem Band.

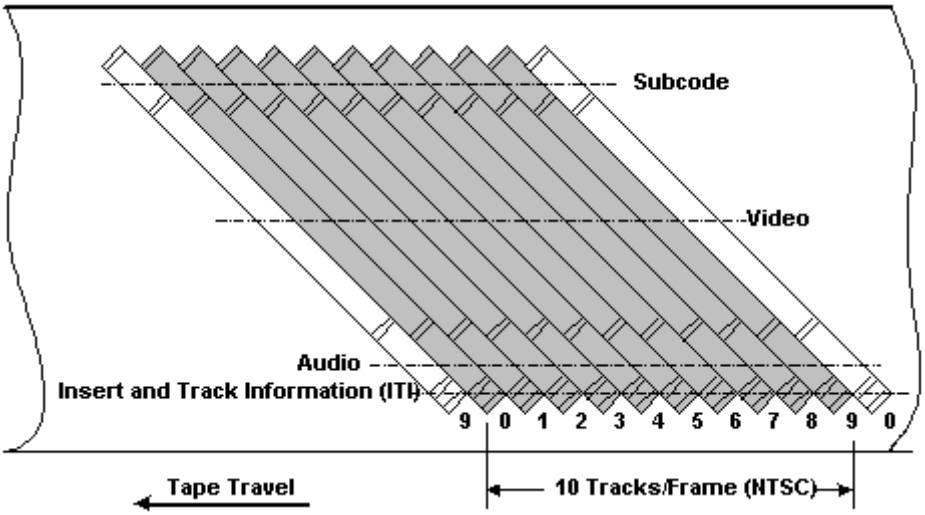


Abb. 7.3 : Aufnahme eines Bildes nach NTSC Norm

Um nun weitere Funktionen, wie Video over Sound (VOS), insert editing, audio dubbing und Timecodeanzeige während des Spulens zu ermöglichen, wird die Schrägspur in 4 horizontale Sektoren aufgeteilt, siehe dazu Abb. 7.3:

- *Insert and Track Information (ITI) Sektor*: Dieser Bereich enthält Informationen über den Track status und dient als Kontrollspur während eines Insert Schnittes, da die Pilottöne bei der Aufnahme nicht gelesen werden können.
- *Audio Sektor*: Die Audiospur enthält Audio und Audio Auxiliary (AAUX) Daten. Die DV Spezifikation sieht entweder zwei 32 kHz 12 Bit abgetastete Stereo Kanäle (=4 Kanäle) oder einen Stereo Kanal in 16 Bit mit einer Abtastrate von 48 kHz, 44.1 kHz oder 32 kHz vor.
- *Video Sektor*: Der Videobereich nimmt den größten Platz des Bandes für Video und Auxiliary Video (VAUX) Daten in Anspruch. Video Bilder werden mit Hilfe der DCT und einer Huffman Kodierung auf 5:1 komprimiert gespeichert. 27 Gruppen von komprimierten 8 * 8 Pixel Regionen sind zu einem Superblock zusammen gepackt. In den VAUX Daten werden Datum und Uhrzeit, Zoom und Linseneinstellung, shutter speed, Farbbalance, sowie andere Kamera Einstellungen gespeichert.
- *Subcode Sektor*: In diesem Bereich wird eine Vielzahl von Informationen geschrieben. Der Bedeutsamste Teil ist der Timecode. Damit der Timecode schnell und kontinuierlich ausgelesen werden kann, müssen die Daten in sehr kleinen Blöcken, genannt packs, auf dem Band abgelegt sein.

8 DV Aussetzer und Fehler

Die meisten Fehler in einem DV System beruhen auf Probleme mit dem Bandmaterial. Da eine Störung auf dem Band nicht beseitigt wird, da nur das error detection Verfahren angewandt wird, muß die fehlerhafte Stelle ausgelassen werden. Man spricht hier auch von einem Dropout.

Dieses Bild zeigt einen Einzelbild Dropout, der aus einer 32 x 8 Pixel Region besteht. Aufgezeichnet wurde er mit der VX1000 auf einem Sony miniDV Band. Ähnliches tritt auch bei dem DHR1000 auf. Wird dieses Band allerdings in einer JVC GR-DV1u abgespielt, so ergibt sich kein Fehler, da dieses System das vorherige Bild darstellt und das Defekte damit ignoriert. Jedoch tritt der Fehler nur an einer Stelle auf, ohne die nachfolgenden Pixelinformationen zu verfälschen.

Abb. 8.1: Sony Dropout

Dieser Dropout wurde mit einer VX1000 auf einer miniDV Kassette von Panasonic erzeugt. Es betrifft eine 32 x 20 Pixel Region von denen die untere rechte Ecke, 16 x 4 Pixel, korrekt dargestellt werden. Dieser Dropout ist äußerst selten und konnte durch erneutes Wiedergeben des selben Bandes nicht verifiziert werden.

Abb. 8.2: Panasonic Dropout

Bei diesem Bild handelt es sich um eine Vielzahl von Fehlern. Solche Fehler sind auf verschmutzte oder defekte Videoköpfe zurückzuführen, da einige korrekte Daten vom Band gelesen bzw. geschrieben wurden.

Des weiteren läßt sich ein typischer Bandeffekt feststellen, der durch die interne Pufferung der Bilder entsteht

Abb. 8.3: Multiple Dropout

Bandeffekte treten dann auf, wenn einer der beiden Videoköpfe verschmutzt oder ausfällt. In diesem Beispiel spielt der VTR ein defektes verknittertes Band ab. Einer der Videoköpfe konnte jedoch noch ein fehlerfreies Signal erzeugen. Da der andere Kopf kein reproduzierbares Signal lieferte, wurde das letzte Halbbild aus dem internen Puffer des VTRs wiedergegeben. Dieses Bild wurde in 525/59.94 NTSC aufgenommen, da 10 Bänder zu erkennen sind. Bei 625/50 PAL entstehen 12 Bänder.

Abb. 8.4: 525/59.94 Bandeffekt

Diese Sorte von Bandeffekten entsteht bei einer kurzzeitigen Verschmutzung des Videoköpfe, welches durch einen Abrieb der ME Schicht des Videobandes passiert. Nach einigen Sekunden verflüchtigen sich diese Partikel.

Bei diesem Beispiel entstanden 5 eingefrorene Streifen für ca. 3 Sekunden innerhalb einer Szene.

Abb. 8.5: Bandeffekt während der Aufnahme

9 Steckverbindungen: 1394/FireWire

Mit Hilfe dieser neuen digitalen Steckverbindung wird eine Kommunikation zwischen 2 Geräten hergestellt. Firewire ist ein Protokoll, ähnlich TCP/IP für Internetverbindungen, welches eine Hochgeschwindigkeitskommunikation (max 400Mbps) auf einer seriellen Leitung ermöglicht. Der volle Name ist 1394-1995 IEEE Standard for a High Performance Serial Bus. 1394 behandelt nicht die Kodierung und Dekodierung der Daten, sondern nur die reine Kommunikation. Die Vorteile dieser Steckverbindung sind:

- Man kann digitale Kopien zwischen 2 Camcordern oder VTRs mit 1394 I/O Interface ohne Qualitätsverlust durchführen.
- Es ist ein linear editing ohne Qualitätsverlust möglich. Es entstehen keine Artifakte durch mehrfaches Schneiden und Kopieren!
- Digitale Daten können von einem VTR oder einem Camcorder mit 1394 I/O Interface direkt in einen Computer übertragen werden. Der Anschluß erfolgt über eine Steckkarte mit 1394 I/O Anschluß. Diese Karten sind von Adaptec oder FAST erhältlich. Innerhalb des Computers können die Audio/Video Informationen direkt auf die Festplatte gespeichert werden. Es ist keine Digitalisierung erforderlich. Die Festplatte sollte aber wenigstens über eine kontinuierliche Datenrate von 3.6 Mbytes/sec verfügen. Dieses kann mit SCSI II- AV Platten erreicht werden
- Günstiger Preis. Diese Schnittstelle wird in bereits in vielen low-end DV-Kameras eingebaut und kostet wesentlich weniger als die professionelle Lösung, die mit einer SMPTE 259M SDI (serial digital interface) Schnittstelle arbeitet.

Stellt man die Kosten mit der Leistung gegenüber, so ergibt sich die folgende Tabelle, bei der 10 für eine perfekte digitale Kopie steht:

IEEE-1394	10
SDI	9.8
Analog Component (Y, R-Y, B-Y)	9
Y/C ("S-video")	8
Analog Composite	5
Aufnahme mit einer Kamera vom Bildschirm	1

10 DV Formate

System	*DV*	*DVCam*	*DVCPro*	*Betacam SX*	*Digital Betacam*	*Digital-S*
Entwickler	56 Firmen	Sony	Panasonic	Sony	Sony	JVC
Bandgeschwindigkeit	18,8 mm/s	28,2 mm/s	33,8 mm/s	59,6 mm/s	96,7 mm/s	57,8 mm/s
Spurbreite	10 µm	15 µm	18 µm	32 µm	24 µm	20 µm
Quantisierung	8 Bit	8 Bit	8 Bit	8 Bit	10 Bit	8 Bit
Kompressionsfaktor	5:1	5:1	5:1	10:1	2:1	3,3:1
Signalverarbeitung	4:2:0	4:2:0	4:1:1 (4:2:2)	4:2:2	4:2:2	4:2:2
Datenrate Video/ges.	25/41,85 Mbps	25/41,85 Mbit/s	25/41,85 Mbit/s	18/40 Mbit/s	84/125,68 Mbit/s	50/99 Mbit/s
Digitale Schnittstelle	IEEE 1394	QSDI,SDI	SDI,CSDI	SDDi,SDI	SDI	SDI
Audio Sampling	12/16 Bit	16 Bit	16 Bit	16 Bit	16 Bit	16 Bit
Audio Quantisierung	32/48 KHz	32/48 KHz	48 KHz	48 KHz	48 KHz	48 KHz
Audio Kanäle	4/2	4/2	2	4	4	4
Bandmaterial	MiniDV/DV, ME-Band	MiniDV/DV, ME-Band	MiniDV/DV, ME-Band	Betacam	Betacam	VHS
Spielzeiten/ min	60/270	40/184	63/123	60/240	bis 125	105

Um einen groben Überblick über die verschiedenen digitalen Camcorder-Formate zu geben, sind im Folgenden die wichtigsten Systeme, sowie deren Entwickler aufgeführt:

System	*Zielgruppe*
DV (56 Firmen)	Consumer, Amateur
DVCAM (Sony)	Industriefilm,teilweise Broadcast
DVCPRO (Panasonic)	Consumer, Broadcast
Betacam SX (Sony)	Broadcast
Digital Betacam (Sony)	Broadcast
Digital-S (JVC)	Broadcast, Studiobereich

10.1 DVCPRO

DVCPRO ist ein proprietäres Format von Panasonic und Television Systems Co. (PB&TSC), welches auf einem metal-particle Band mit einer 18 micron Spurbreite arbeitet. Die Bandgeschwindigkeit beträgt 33,82 mm/sec. DVCPRO benutzt die beiden Längsspuren auf dem Kassettenband zur Steigerung der Performance bei der Videoschnittbearbeitung. DVCPRO Videorekorder ermöglichen das Kopieren des Bandmaterials in vierfacher Geschwindigkeit. Zudem können auch MiniDV Kassetten, mit einem Adapter, und DVCAM Kassetten wiedergegeben werden. Des weiteren ist das SMPTE Timecode Format an das standardisierte Format D-7 angelehnt.

Als Geräte sind der AJ-D230 VTR und die AJ-D200 Kamera verfügbar. Diese Geräte sind mit dem IEEE 1394 FireWire Anschluß versehen.

10.2 DVCPRO50

Dies ist eine Weiterentwicklung der DVCPRO von PB&TSC. Erreicht wird eine Kompressionsrate von 3,3:1 bei einem 4:2:2 Sampling auf einem 1/4 inch Band. Es wird ein ähnliches Format wie bei der DVCPRO verwendet, aber die Trackgeschwindigkeit auf 67,64 mm/sec erhöht.. Allerdings nutzt die 4:2:2 DVCPRO nicht das allgemeine DV Format! Zudem werden statt 10 tracks pro frame (NTSC) 20 tracks pro frame aufgezeichnet (bei PAL 24 tracks pro frame). Erreicht wird dieses durch 2 weitere Videoköpfe und 2 parallel arbeitenden Codecs Die Datenrate wächst auf 50 Mbps an. Die Aufnahmekapazität einer DVCPRO Kassette verringert sich mit dem DVCPRO50 Format auf ca. 61 min. Eine längere Aufnahmedauer von 93 min wird durch eine spezielle DVCPRO50 Kassette erreicht. Das neue DVCPRO50 Format ist aber abwärtskompatibel zu dem alten 4:1.1 DVCPRO Format.

Die 900-er Serie der DVCPRO50 Geräte zeichnen in beiden Verfahren auf. Es steht ein 4:3 und 16:9 Bildformat zur Verfügung. Der AJ-D950 VTR ist zwischen NTSC 525/59.94 und PAL 625/50 umschaltbar. Jedoch kann dieser VTR die miniDV Kassetten, auch nicht mit dem Adapter, lesen.

Panasonic und JVC arbeiten gemeinsam an einem 100 Mbps DV basierten Aufzeichnungssystem, welches dann auch HDTV Signale verarbeiten kann. Panasonic kündigt dieses Gerät mit der Bezeichnung: DVCPROHD100 an und verwendet eine ähnliche Technologie der 50 Mbps Produkte.

10.3 DVCAM

DVCAM wurde von Sony entwickelt und ist für die industrielle Videoproduktion vorgesehen. Sony verwendet ein ME Band, welches nur mit der Spurbreite von 15 microns und einer Bandgeschwindigkeit von 28,22 mm/sec von dem Standard DV Format abweicht. Man wählte diese breitere Spur, da eine höhere Bildstabilität für linear editing sichergestellt wird. Allerdings verringert sich dadurch die Aufnahmekapazität einer 4,5h DV Kassette auf ca. 3h. DVCAM Kassetten besitzen ein 2MBit MIC, welches bis zu 198 Szenendaten speichern kann. Es stehen zwei Audiospuren mit 16Bit und 48kHz zur Verfügung. Als Profigeräte bietet Sony den DSR-85 DVCAM VTR, die DSR-130 Kamera und den ES-7 Schnittcomputer an. Der low-end Camcorder DSR-200 ist eine DVCAM Version des DCR-VX9000. DVCAM kopiert die Daten mit vierfacher Geschwindigkeit zwischen den Geräten. Zudem können DVCAM VTR`s auch Standard DV Kassetten, aber keine DVCPRO Kassetten, lesen.

10.4 SDL

SDL ist ein long-play consumer DV Format, welches von Sony in der DCR-PC7, ein miniature Camcorder, eingebaut wurde. Die Aufnahmekapazität einer MiniDV Kassette wird um das Doppelte verlängert, jedoch verschlechtert sich dabei aber die Qualität. Erreicht wird dies durch eine langsamere Trackgeschwindigkeit und eine schmalere Spurbreite.

10.5 Digital-S

Dieses Videoformat wurde von der Japan Victor Corp`s (JVC) entwickelt. Es arbeitet im Gegensatz zu den anderen Formaten auf einer herkömmlichen 1/2 inch VHS Videokassette mit einer speziellen Beschichtung. Die Qualität ist vergleichbar mit den wesentlich teueren professionellen Formaten DVCPRO50 oder Betacam SP. Die Kompressionsrate liegt bei 3,3:1 bei einem 4:2:2 Sampling. Digital-S bietet 2 linear audio cue tracks und einen linear control tack. Zudem werden 2 Stereo Audiospuren mit 16Bit und 48kHz aufgezeichnet, von denen aber die JVC Produktlinie nur eine Stereospur unterstützt. Die Aufnahmekapazität liegt bei max. 104 min auf Standardbändern. Digital-S VTRs können neben Digial-S Aufnahmen auch analoge S-VHS Kassetten wiedergeben. Durch das breitere Band wird zudem noch eine geringere Dropoutrate erzielt. Da JVC die konventionelle Mechanik der S-VHS Videorekorder benutzt, konnten die Entwicklungskosten drastisch gesenkt werden.

10.6 Betacam SX

Sony entwickelte dieses 1/2 inch digitale Format für ENG Applikationen, wie Broadcasting. Betacam SX nutzt die MPEG-2 Kompressionstechnik mit 4:2:2 Sampling. Die Auflösung liegt bei 720 x 512 Bildpunkten, im Gegensatz zu DV mit 480 aktiven Zeilen. Es ergibt sich eine max. Datenrate von 50 Mbps, welches jedoch laut Sony auf 18 Mbps reduziert wird, da ein Satellitenkanal nur 9 Mbps Nutzbandbreite hat.

10.7 DV-Formate im Vergleich: Qualität

DV Formate haben eine gleichwertige Qualität wie Betacam SP oder MII. Jedoch zeigt DV bei wiederholten Bildfolgen eine deutliche Verbesserung gegenüber Betacam SP, da dieses zu dropped out frames neigt.

Die verschiedenen Systeme lassen sich folgendermaßen differenzieren (1 = Normales Video, 10 = Studioqualität):

D-5 (10-bit uncompressed digital)	10
D-1 (8-bit uncompressed digital)	9.9
Digital Betacam, Ampex DCT	9.7
Digital-S, DVCPRO50	9.6
DV, DVCAM, DVCPRO	9.2
MII, Betacam SP	9.1
D-3, D-2 (composite digital)	9
1" Type C	8.9
3/4" SP	6.5
3/4", Hi8, SVHS	5
Video 8, Betamax	4
VHS	3
EIAJ Type 1, Fisher-Price Pixelvision	1

10.8 Formatumwandlung: DV - MJPEG

Eine Formatumwandlung ist ohne Qualitätsverlust durchführbar. Jedoch sollte man beachten, daß MJPEG mit JPEG Einzelbildern arbeitet, die bei gleichem Kompressionsverhältnis, wie DV ein schlechteres Ergebnis zeigen. Deshalb ist das Kompressionsverhältnis eines JPEG Bildes auf ca. 3:1 zu erhöhen, um die gleiche Qualität eines DV kodierten Bildes, welches in 5:1 komprimiert ist, zu erhalten.

Beide Verfahren arbeiten nach der diskreten Cosinus Transformation (DCT), so daß die gleichen Artefakte und Schwächen innerhalb eines Bildes entstehen. Allerdings nutzt DV eine blockorientierte Optimierung der Quantisierungstabellen, wohingegen JPEG mit festen Quantisierungstabellen innerhalb eines Bildes arbeitet.

Die besten Resultate für eine Formatumwandlung erreicht man mit Hilfe von transcoding. Ein vorheriges Dekomprimieren und Neukomprimieren in das andere Format führt zu erhöhten Degradierungen innerhalb eines Bildes.

Auf der NAB ’96 hatte Panasonic folgendes Experiment durchgeführt:

Ein digitales unkomprimiertes Signal nach D-5 (ITU-R-601) wurde an Eingang 1 eines digitalen Switches gegeben. Ebenfalls wurde diese Signal von einer DVCPRO komprimiert und anschließend unkomprimiert an den Eingang 2 des Switches geschaltet. Nun wurde das bearbeitete Signal an einen Tektronix ProFile DDR, der nach dem JPEG-Verfahren mit einer Kompression 2.5-3:1 arbeitet, weitergeleitet. Das Ausgangssignal dieses Gerätes wurde an den Eingang 3 des Switches geschaltet und wiederum von einer anderen DVCPRO komprimiert / dekomprimiert an den Eingang 4 gegeben.

Mit Hilfe eines Wipe Screens konnte man nun die Vergleiche zwischen Quelle 1 und den übrigen sehen. Das D-5 Signal war besser als die DVCPRO komprimierten Bilder, welche die typischen Artefakte aufwiesen. Diese waren aber nur bei näheren Hinsehen sichtbar.

Alle nachfolgenden Signale zeigten keine sichtbaren Unterschiede zu dem Signal an Quelle 2. Die erste DV Kompression hatte nämlich die schwierigen Details innerhalb der Bilder entfernt. Der DV Codec hinterließ ein gut zu verarbeitendes DCT Bild, welches ein wenig durch die nachfolgende Kompression des Tektronix ProFile DDR litt. Die erneute Kompression der DVCPRO verlief ohne sichtbaren Qualitätsverlust.

Diese geringen Verluste durch die nachfolgende JPEG Kompression kann bei geringen Kompressionsverhältnissen, wie 3:1 oder weniger vernachlässigt werden. Eine stärkere Degradierung der Bildsignale tritt bei der analogen Übertragung, oder bei höheren Kompressionsverhältnissen (4:1 oder mehr) auf.

Welches Verfahren ist nun das Bessere?

Diese Frage kann nicht pauschal beantwortet werden, da dies vom Verwendungszweck abhängt. DV ist hervorragend, wenn die Signalverarbeitung komplett in DV Codecs durchgeführt wird. Des weiteren werden bei den höheren Kompressionsverhältnissen gegenüber einer gleichwertigen JPEG Komprimierung keine speziellen Festplatten, oder RAID Arrays benötigt.

Der Nachteil bei DV ist allerdings das festgesetzte Kompressionsverhältnis von 5:1. Eine Komprimierung in einem höheren Verhältnis für nicht sehr detailgetreue Aufnahmen ist somit nicht möglich. Dieses ist interessant zur Erstellung von Previews, um den Festplattenplatz zu reduzieren.

M-JPEG ist die ausgefeiltere Technik, die vor allem von High end Geräten (Avid, SciTex, D-Vision, etc.) genutzt wird. Der größte Vorteil liegt in der Wahl des Kompressionsverhältnisses. Bei geringen Kompressionsverhältnissen ergibt sich ein besseres Bild als bei DV. Allerdings stellt M-JPEG, bei gleicher Qualität wie DV, höhere Anforderungen an das Festplattensystem, da eine größere Menge an Daten entsteht. In der Regel wird ein RAID Array benötigt.

11 LSI Implementierung eines DV Aufnahme Systems

Matsushita Electric Ind. Co.,Ltd. beschreibt ein LSI Entwurf der Schaltkreise für das consumer DV. Hier nun ein Auszug aus dem Paper, „The Development of Audio and Video Signal Processing LSI for SD-DVC“:

Der LSI Chipsatz wird in der DVCPRO von Panasonic eingesetzt. Die interessanteste Eigenschaft dieses Chipsatzes besteht in der Fähigkeit ein DV 4:1:1 YCrCb digital component Datenstrom in/von einen Standard 8 bit parallel BT.601 4:2:2 Datenstrom zu interpolieren.

Das Audio I/O wird mit Hilfe des kommerziell verfügbaren I2C-Interface von Philips realisiert. Dieses IC erzeugt ein serial D-1 (SMPTE 259M) video und AES/EBU serial audio Signal zur Versorgung der digitalen Komponenten der DVCPRO.

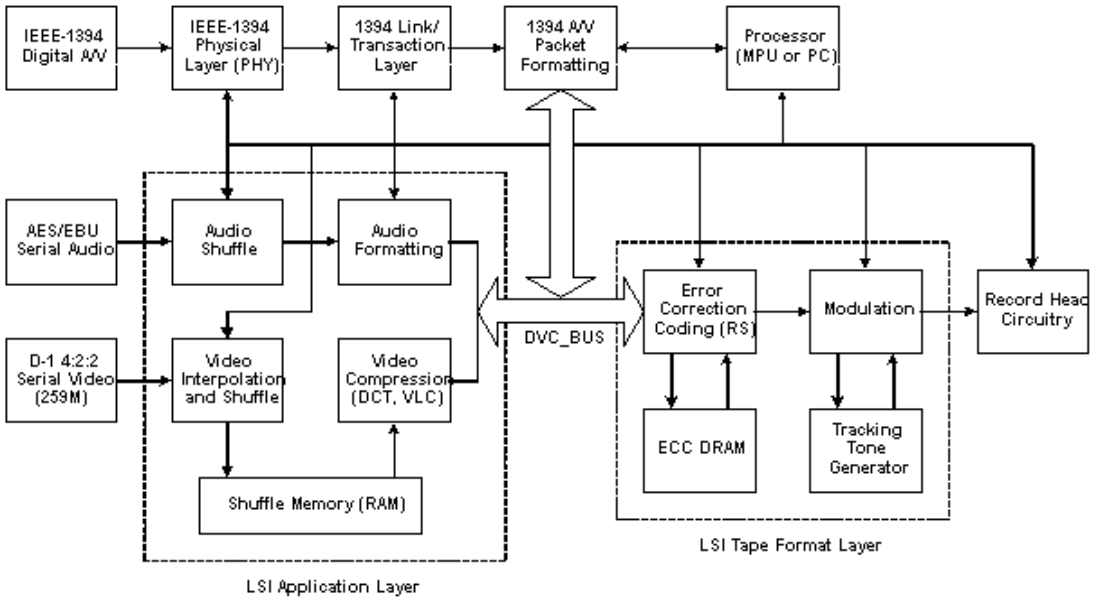


Abb. 11.1 : LSI Implementierung

Jedoch besitzt keine DVCPRO oder consumer Camcorder die IEEE-1394 Schnittstelle, in der Grundausstattung.

Firmen, wie Matrox und Truevision haben bereits PC-Karten mit dem IEEE-1394 Anschluß entwikkelt, die den Chipsatz von Panasonic als Hardware DV-Codec verwenden. Die DV-Master von FAST Multimedia nutzt den Sony DV-Codec.

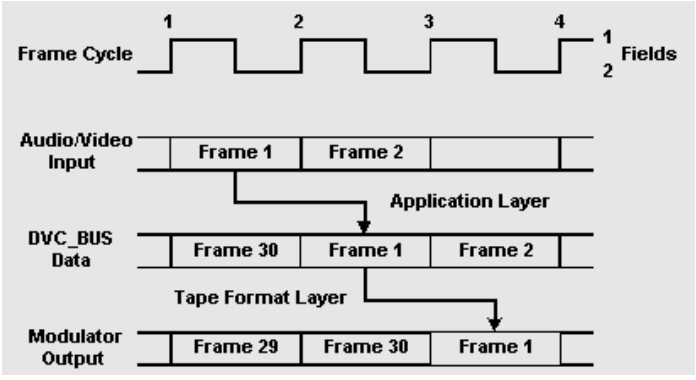


Abb. 11.2 : Timing Diagramm

In der Abbildung 11.1 erkennt man den Aufbau der DVCPRO. Der Application Layer behandelt shuffle/deshuffle, sowie compression/decompression der DV-Daten. Der Tape Format Layer kümmert sich hauptsächlich um die Erzeugung der Reed-Solomon-Codes. Desweiteren wird die Modulation/Demodulation für die Aufnahme/Wiedergabeköpfe erzeugt. Die beiden Einheiten sind mit einem

proprietären Bus, den DVC_BUS, verbunden. Dieser arbeitet mit 8 Datenleitungen und 3 Kontrollsignalen:

- BDEN (data enable)
- BDCK (data clock)
- BQUIET (data start)

Bei der Aufnahme und Wiedergabe des Audio/Videostroms entsteht eine Verzögerung von 2 Bildern, siehe Abb. 11.2.

12 Literatur

[Jenn 98] Roger Jennings: „Consumer and Professional Digital Video Recording and Data Formats“ http://www.adaptec.com/technology/standards/1394formats.html [Stand: 03.08.1998]

[Jenn 99] Roger Jennings: „Video, Audio, and Data Recording Formats“ http://www.adaptec.com/technology/standards/1394formats1.html [Stand: 09.02.1999]

[Tew 99] Thomas Tewell: „DV Coding: How it Works with IEEE-1394“ http://desktopvideo.miningco.com/library/weekly/aa032698.htm [Stand: 06.04.1999]

[Wilt 98] Adam J. Wilt: „DV Pix - Image Defects due to Tape Problems“ http://www.adamwilt.com/pix-defects.html [Stand: 16.08.1998]

[Wilt 98] Adam J. Wilt: „DV Pix - Image Artifacts“ http://www.adamwilt.com/pix-artifacts.html [Stand: 19.07.1998]

[Wilt 98] Adam J. Wilt: „DV Pix - Sampling Methods“ http://www.adamwilt.com/pix-sampling.html [Stand: 16.08.1998]

[Wilt 99] Adam J. Wilt: „The DV, DVCAM, & DVCPRO Formats“ http://www.adamwilt.com/DV.html [Stand: 04.02.1999]

QuickTime 3 / 4

Bernhard Flechtker

FB Informatik, Uni Dortmund

flechtker@microtherapy.de

Zusammenfassung

QuickTime ist eine Entwicklung der Firma Apple um zeitbasierte Daten zu verarbeiten und darzustellen. Es können sowohl typische Multimediadaten wie Audio und Videosequenzen oder Animationen verarbeitet werden, aber auch andere zeitbasierte Daten wie zum Beispiel Daten aus einer Meßwerterfassung.

QuickTime bietet für den Benutzer drei Applikationen als Front-End. Der QuickTime Player verarbeitet Audio und Videosequenzen, die in der Autorenversion auch bearbeitet werden können, der QuickTime Viewer stellt einzelne Bilder dar und der QuickTime VR Player ermöglicht die Betrachtung interaktiver 3D Movies.

Des weiteren beinhaltet QuickTime Routinen, die es dem Programmierer ermöglichen, Multimediananwendungen zu entwickeln ohne selbst grundlegende Funktionen wie zum Beispiel die Wiedergabe oder Aufzeichnung eines Films programmieren zu müssen. Diese hohe Abstraktionsebene sorgt auch dafür, das für den Benutzer ein einheitliches Erscheinungsbild gewährleistet wird. Neuentwicklungen, wie neue Kompressionsalgorithmen oder spezielle Treiber lassen sich einbinden und erweitern dann die Standardroutinen, auf die anschließend andere Applikationen zugreifen können.

Leider bleiben technische Einzelheiten über QuickTime dem Programmierer verborgen, so daß keine genauen Aussagen über die Funktionsweise und Details gemacht werden können. Vor allem die Bild- und Videokompressoren, sind fast ausschließlich Eigenentwicklungen der Firma Apple und nicht weiter Dokumentiert .

1 Überblick und Bestandteile

Apple liefert QuickTime in zwei Versionen aus, einer frei verfügbaren Player Version und einer kostenpflichtigen QuickTime Pro Version. Die Pro Version bietet über das reine Abspielen und Darstellen der Medien hinaus noch Möglichkeiten, um die Medien zu bearbeiten. QuickTime ist

Bestandteil des Macintosh Betriebsystems wird aber auch für alle gängigen Windowsversionen wie Windows NT oder Windows 95/98 angeboten, sowie für Silicon Graphics Rechner.

QuickTime läßt sich in zwei Ebenen unterteilen, einer Benutzerebene und einer Entwicklungsebene. Auf der Benutzerebene stehen drei Applikationen als Front-End zur Verfügung, der QuickTime Viewer, der QuickTime Player und der QuickTime VR Player. Der QuickTime Viewer ermöglicht es dem Benutzer Standbilder darzustellen und in der Pro Version lassen sich diese Bilder auch in die gängisten Bilddateiformate konvertieren, wie zum Beispiel TIFF, PICT, JPEG. Der QuickTime Player kann Audio und Videosequenzen darstellen und in der Pro Version auch bearbeiten. So können zum Beispiel mehrere Audio und Videosequenzen miteinander geschnitten werden und mit diversen Effekten versehen werden. Mit Hilfe des QuicktTime VR Players kann der Benutzer dreidimensionale QuickTime VR Movies darstellen und durch spezielle Eingaben interaktiv steuern.

Abbildung 1-1: QuickTime Player stellt ein Movie dar.

Auf der Entwicklungsebene stehen spezielle Manager zur Verfügung, die grundlegende Routinen beinhalten. Diese Routinenesammlungen werden vom Programmierer angesprochen und ermöglichen es auf einer hohen Abstraktionsebene zu programmieren. So braucht sich zum Beispiel der Programmierer keine Gedanken um die Hardware zu machen, da lediglich die Managerroutinen ansprechen werden und nicht direkt die Hardware. Zu diesen Routinen gehören unter anderem auch Kompressions- und Dekompressionsalgorithmen sowie Routinen zum Anfertigen, Editieren und Abspielen von zeitbasierten Daten. Die Routinen werden von einem Manager, dem Component Manager, registriert und dadurch anderen Managern zur Verfügung gestellt.

2 Dateiformate und Codecs

QuickTime unterstützt alle gängigen Dateiformate für Standbilder, Digitales Video und Audio. Abbildung 2-1 zeigt eine Auflistung dieser Dateiformate.

Darüber hinaus bietet QuickTime auch die gebräuchlichsten Codecs als Software-Implementation, die

durch Drittanbieter in Form von Hard- und Software-Lösungen erweitert werden können. QuickTime integrieren diese Erweiterungen und stellt sie allen Applikationen zur Verfügung. So kann zum Beispiel eine Hardwareerweiterung die das Komprimieren und Dekomprimieren von JPEG Dateien beschleunigt, von allen Applikationen benutzt werden, ohne daß diese spezielle Treiber benötigen.

Import file formats

- 3DMF
- AIFF
- AU
- Audio CD Data (Macintosh)
- AVI
- BMP
- DV
- FlashPix*
- GIF
- JPEG/JFIF
- Karaoke
- MacPaint
- Macromedia Flash
- MIDI
- MPEG 1 (Macintosh)
- MPEG 1,Layer 3 (MP3)
- Photoshop*
- PICS
- PICT
- Pictures
- PNG
- QuickTime Image File
- QuickTime Movie
- SGI
- Sound
- TARGA
- Text
- TIFF*
- Virtual Reality (VR)
- Wave

Export formats

- AIFF
- AU
- AVI
- BMP
- DV Stream
- FLC
- Image Sequence movie exporters
- JPEG/JFIF
- MacPaint
- MIDI
- Photoshop
- PICT
- Picture
- PNG
- QuickTime Image
- QuickTime Movie
- SGI
- System 7 Sound
- TARGA
- Text
- TIFF
- WAV

Abbildung 2-1: Unterstützte Dateiformate (mit * markierte Formate werden erst ab QuickTime 4.0 unterstützt.)

2.1 Video Codecs

QuickTime unterstützt 15 Codecs für die Komprimierung von Stand- und Bewegtbildern. Eine besondere Bedeutung kommt den Codecs Cinepak, Sorenson Video, und Apple Animation zu, da sie einen hohen Verbreitungsgrad haben und in allen wichtigen Videobearbeitungsprogrammen implementiert sind.

2.1.1 Cinepak

Cinepak ist ein alter Codec der schon 1991 mit QuickTime 1.0 ausgeliefert wurde. Er eignet sich besonders für die Komprimierung von 24-Bit-Videos, die anschließend auf einer CD-ROM abgespeichert werden sollen, oder für herunterladbare Web-Videodateien, da er auch auf älteren Rechnern wie zum Beispiel ´486 PCs bzw. 68xx Macs noch ausreichend schnell arbeitet und recht verbreitet ist. Die Bildqualität ist akzeptabel aber jedoch merklich schlechter als die des Sorenson Video Codecs. Cinepak erzeugt einen Datenstrom von 300 KByte/s bei einer Framegröße von 320x240 Pixel und einer Bildwiederholungsfrequenz von 15fps. Dadurch erzielt der Codec ein höheres Komprimierungsverhältnis und eine schnellere Wiedergabegeschwindigkeit als der Video-Codec, der sich eher zu Testzwecken eignet. Die Datenrate für die Wiedergabe kann verändert werden, wobei die Bildqualität bei Datenraten unter 30 KB pro Sekunde merklich abnimmt. Cinepak arbeitet asymmetrisch: Er dekomprimiert zwar schnell, die Komprimierung ist jedoch recht zeitaufwendig.

2.1.2 Sorenson Video Codec

Der Sorenson Video Codec eignet sich zur Komprimierung von 24-Bit-Videos, die anschließend auf einer CD-ROM abgespeichert oder auf einem Internetserver abgelegt werden sollen, sowie für Streaming Video Anwendungen. Ähnlich wie Cinepak ist dieser neuere Codec für hohe Qualität bei Datenraten unterhalb von 200 KB pro Sekunde entwickelt worden. Dieser Codec liefert eine bessere Bildqualität und kleinere Dateien als Cinepak. Sorenson arbeitet ebenfalls asymmetrisch benötigt allerdings mehr Komprimierungszeit als Cinepak. Er unterstützt temporäre Skalierbarkeit, die es ermöglicht, daß ein für einen High-End-Computer exportierter Film problemlos auf einem Low-End-Computer abgespielt werden kann. Dadurch eignet er sich auch hervorragend für Streaming Video Anwendungen. Sorenson erzeugt einen Datenstrom von 72 KByte/s bei einer Framegröße von 320x240 Pixel und einer Bildwiederholungsfrequenz von 15fps.

2.1.3 Animation Codec

Der Codec Animation ist entwickelt worden um 2-Dimensionale Animationen zu komprimieren. Diese Clips enthalten großen Flächen in Volltonfarben, wie man sie zum Beispiel von Zeichentrickfilme her kennt., Es kann ein Verlustgrad angegeben werden, wobei "100%ige Qualität" eine verlustfreie Komprimierung bedeutet. Der Animations-Codec verwendet einen auf dem Run-Length-Encoding basierenden Apple-Komprimierungs-Algorithmus.

2.1.4 Weiter Video Codecs

QuickTime untersützten eine Reihe weiterer Video Codecs, die hier allerdings nur namentlich erwähnt werden. Abbildung 2-2 zeigt eine Aulistung dieser Komprimieralgorithmen .

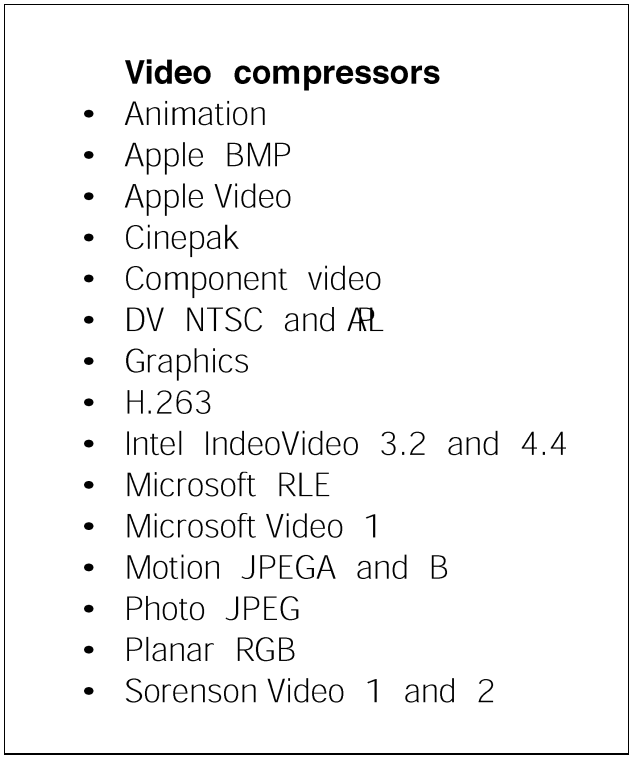


Abbildung 2-2: Von QuickTime unterstützte Video Codecs.

2.2 Audio Codecs

QuickTime unterstützt zwölf Audio Komprimierungsalgorithmen. QDesign Music, Qualcomm PureVoice und MPEG Layer 3 sind wohl die bedeutendsten Codecs.

2.2.1 QDesign Music Codec

Der QDesign Music Codec eignet sich für die Komprimierung qualitativ hochwertiger Musik, die über das Internet verbreitet werden soll. QDesign Music kann Audio in CD-Qualität (16 Bit; 44,1 KHz) über eine Leitung von 28,8 KB pro Sekunde liefern.

2.2.2 Qualcomm Pure Voice Codec

Für die Komprimierung von Sprache wurde Qualcomm PureVoice entwickelt. Dieser Codec funktioniert bei 8 KHz optimal. Pure Voice basiert auf dem Code Division Multiple Access (CDMA)-Technologiestandard für das zellulare Fernsprechwesen.

2.2.3 µLaw

µLaw 2:1 ermöglicht den Audioaustausch mit Anwendungen auf Plattformen geeignet, bei denen µLaw ein Standard-Audioformat ist (wie beispielsweise viele UNIX-Workstations). In Nordamerika und Japan wird µLaw im digitalen Fernsprechwesen eingesetzt.

2.2.4 Weiter Audio Codecs

QuickTime untersützten eine Reihe weiterer Audio Codecs, die hier allerdings nur namentlich erwähnt werden. Abbildung 2-3 zeigt eine Aulistung aller Komprimieralgorithmen unterstützter Codecs.

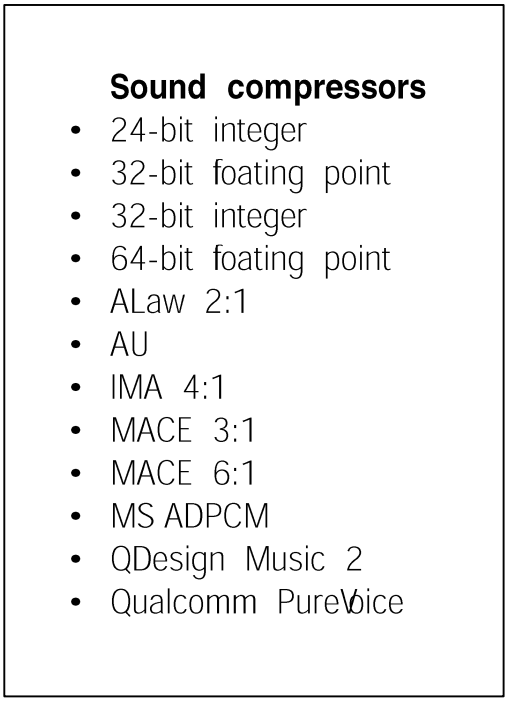


Abbildung 2-3: Audio Codecs die von QuickTime unterstüzt werden.

3 Manager

3.1 Allgemeines zu Manager

Die Macintosh-Systemarchitektur ist so aufgebaut, daß nie direkt auf das Betriebssystem zugegriffen wird sondern immer indirekt über die Toolbox-Ebene auf die Betriebsystemebene. Die Toolbox-Ebene beinhaltet eine große Anzahl an Routinen. Dabei sind die Routinen unter einem Themengebiet zusammengefaßt, den sogenannten Managern bzw. Toolboxen. Es gibt zum Beispiel Manager für die Implementierung von Fenstern, Menüs, Dialogboxen, Buttons, Scrollbars etc.

Die für den QuickTime Entwickler bedeutendsten Manager sind der Component Manager, der Image Compression Manager und die Movie Toolbox.

3.2 Der Component Manager

Durch den Component Manager werden die Components der einzelnen Manager in einer Art Datenbank verwaltet. Der Component Manager bietet dabei Dienste zum Zugriff auf die Components der einzelnen Manager. Ferner gibt es Routinen zum Erzeugen von neuen Components oder zum Registrieren von neuen Components. Dies ist wichtig, da es für die einzelnen Applikationen sonst keine Möglichkeit gibt herauszufinden, welche Components welche Routinen beinhalten. Wenn zum Beispiel eine Hardwareerweiterung neue Codecs enthält, können diese Codecs auch von anderen Applikationen genutzt werden, da sie den Component Manager benutzen um auf eine Routine zuzugreifen, die diese Codecs realisiert.

3.3 Der Image Compression Manager

Der Image Compression Manager beinhaltet Prozeduren zum Komprimieren und Dekomprimieren von Standbildern, bewegten Bildern und Sound. Über die Vor- und Nachteile der einzelnen Codecs wurde bereits in Kapitel 1 berichtet. Er stellt im Prinzip eine universelle Schnittstelle zwischen den Daten und den Applikationen dar. Der Image Compression Manager erkennt, um welchen Komprimierungsalgorithmus es sich handelt und verarbeitet die Daten entsprechend.

3.4 Die Movie Toolbox

Durch die Movie Toolbox werden dem Anwender bzw. Entwickler standardisierte Werkzeuge, Fenster und Bedienelemente zum Erstellen, Bearbeiten und Abspielen von QuickTime Movies zur Verfügung gestellt. So beinhaltet die Movie Toolbox zum Beispiel alle wichtigen Routinen zum Editieren und Abspielen von Daten eines Movie. Der Entwickler braucht sich also keine Gedanken zu machen, welche Hardware benutzt wird oder auf welchem Datenmedium sich die einzelnen Daten befinden.

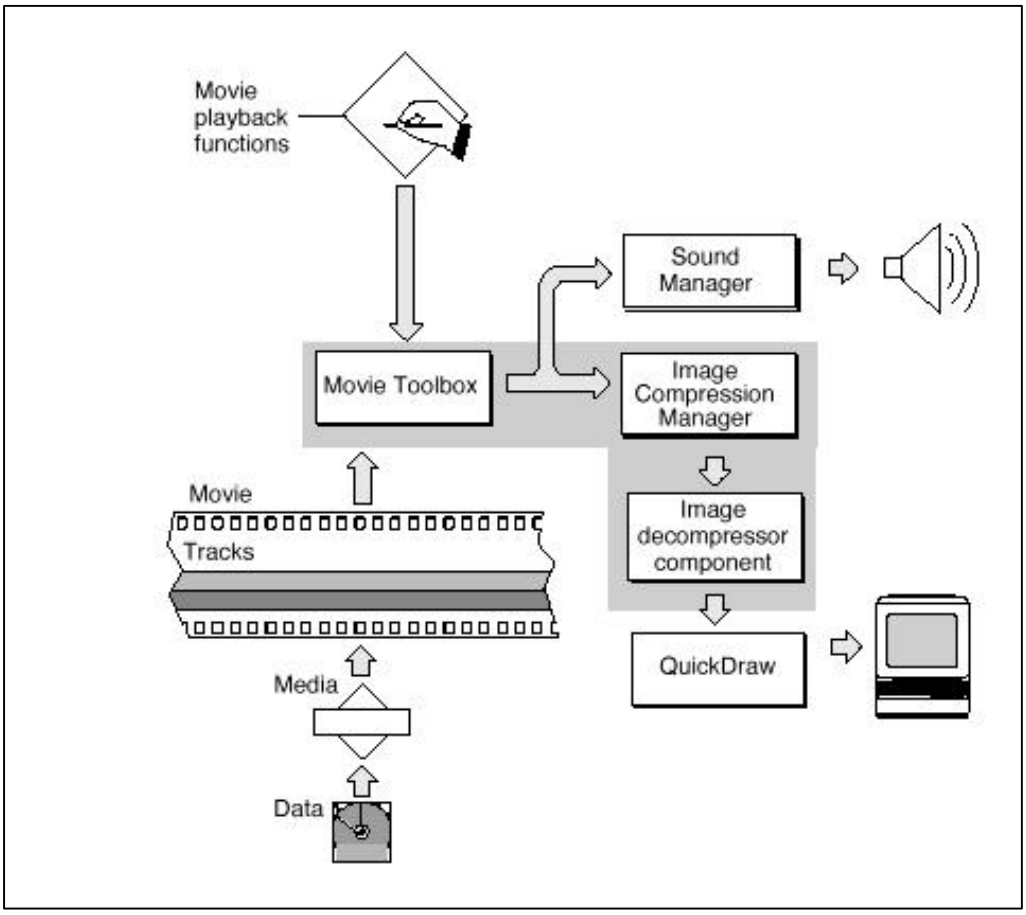


Abbildung 3-1: Darstellen eines QuickTime Movie unter Zuhilfenahme der Manager.

4 Organisation eines Movie

Movie ist die Datenstruktur, in der QuickTime Audio- und Videosequenzen zusammenfaßt. Ein Movie setzt sich aus einem oder mehreren Tracks zusammen, die wiederum jeweils durch Zeiger auf ihr Media (Medium) verweisen. Das Media verweist wiederum auf die eigentlichen Daten des Tracks.

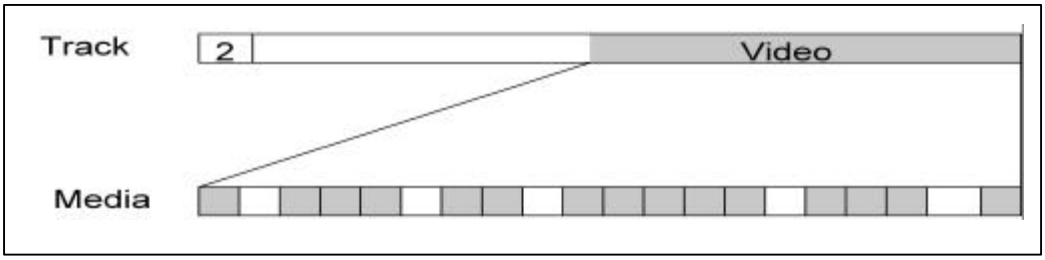


Abbildung 4-1: Track mit entsprechendem Media.

Ein Movie darf beliebig viele Tracks enthalten, die jeweils einen individuellen Startpunkt haben können und unterschiedlich lang sein dürfen. Zusätzlich kann ein Movie auch eine Movie Preview und ein Movie Poster enthalten. Das Movie Poster ist ein einzelnes Bild aus einem Movie, welches zum Beispiel zu Beginn im Movie Player angezeigt wird. Eine Movie Preview ist ein Teilbereich des Movie, der separat angezeigt werden kann (siehe Abb. 4-2).

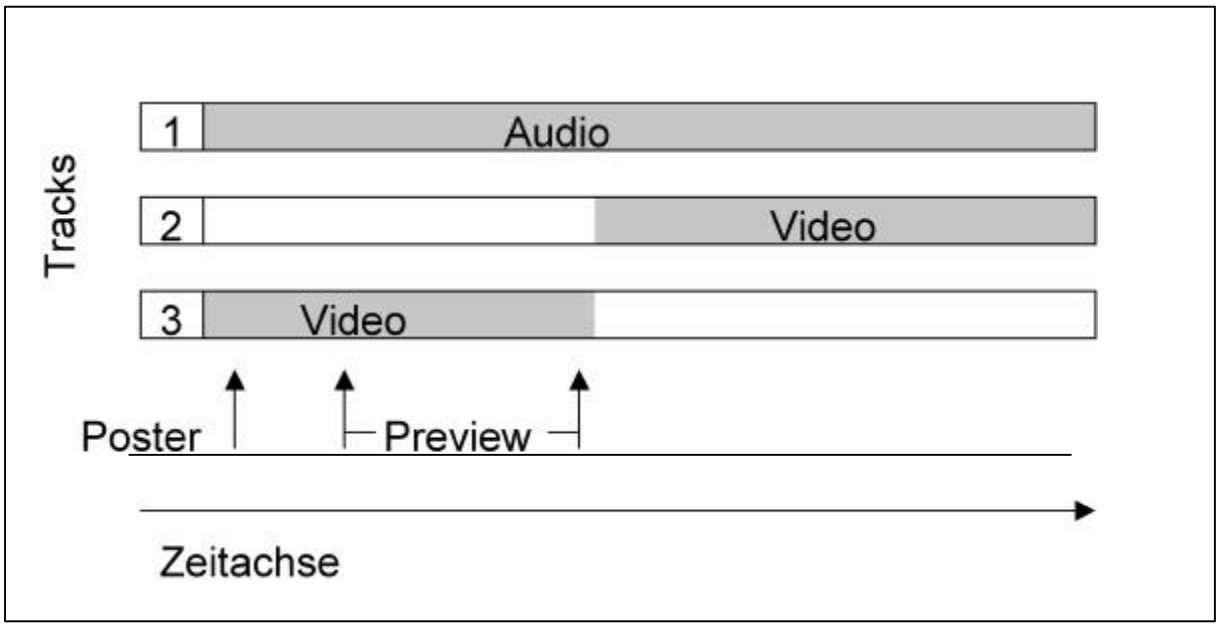


Abbildung 4-2: Tracks eines Movie mit individuellem Offset, Poster und Preview.

4.1 Realisierung eines Movie durch Atome

Ein Movie besitzt eine Baumstruktur und wird durch Atome realisiert. Atome enthalten jeweils einen Header, der Informationen über Länge Art und Typ des Atoms enthält. Atome sind rekursiv, können also andere Atome unterschiedlichen Typs enthalten. Ein Movie Atom enthält zum Beispiel ein Track Atom für jeden Track.

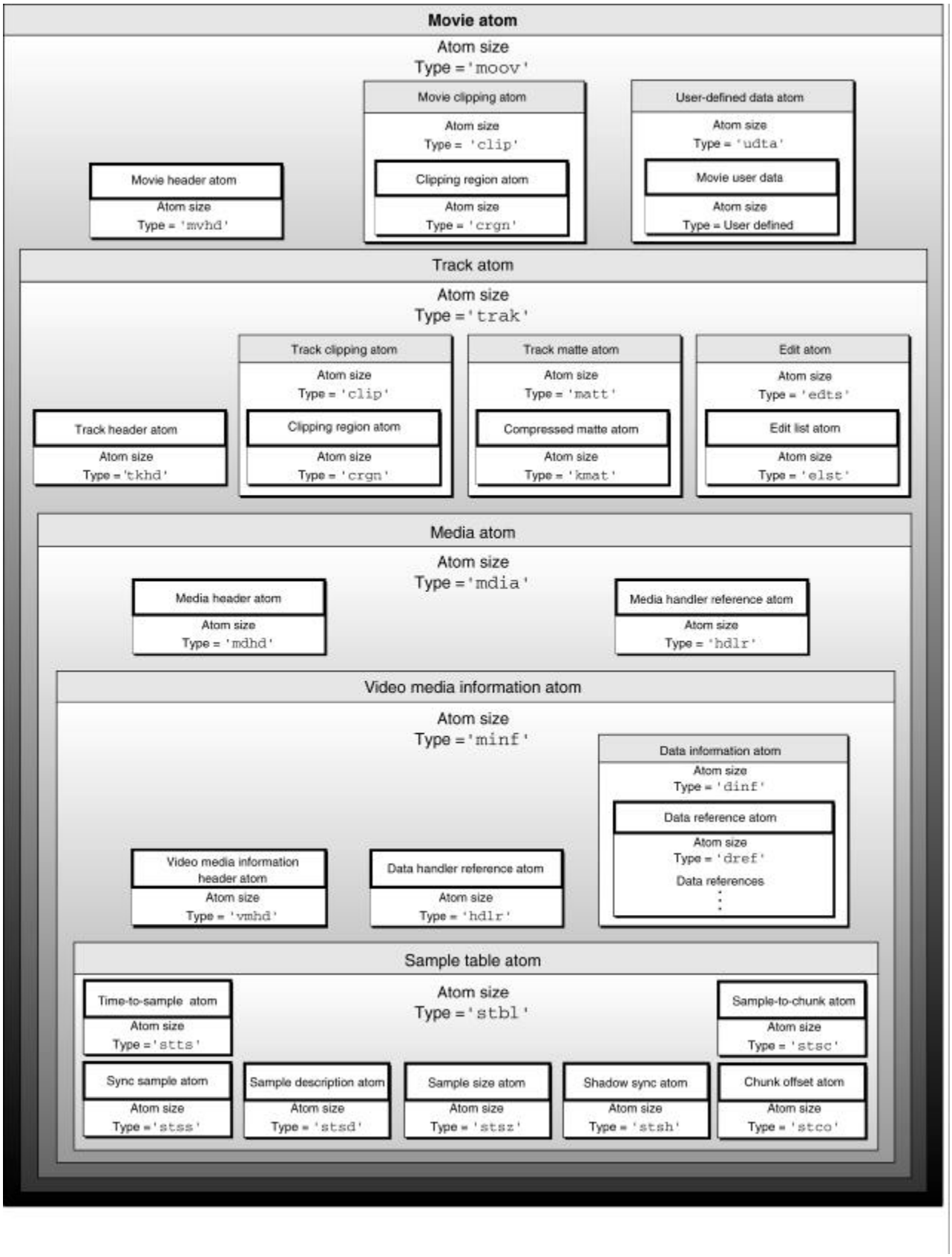


Abbildung 4-2: Aufbau eines Movie Atoms.

4.1.1 Movie Atom

Ein Movie Atom setzt sich aus folgenden Atomen zusammen:

Movie Header Atom: enthält Angaben zur Größe, Datum, Zeitkoordinatensystem, Dauer eines Movie und Informationen zu Movie Preview und Movie Poster.

Movie Clipping Atom: ein Clipping Atom spezifiziert die Fläche, die von einem Movie dargestellt werden soll. Diese Fläche wird durch Grafikbefehle repräsentiert, die die Grafikschnittstelle QuickDraw verarbeiten kann.

User Defined Data Atom: hier können anwendungsspezifische Daten abgelegt werden.

Track Atom: definiert einen einzelnen Track eines Movie, zum Beispiel ein Audio Track.

4.1.2 Track Atom

Ein Track Atom besteht aus:

Track Header Atom: enthält Angaben zur Größe, Datum, Zeitkoordinatensystem eines Tracks sowie Informationen zur Lage eines Tracks innerhalb eines Movie. Weiterhin lassen sich Gruppen von Tracks spezifizieren, von denen je einer alternativ abgespielt werden kann. Dies können zum Beispiel unterschiedliche Audiosequenzen in verschiedenen Sprachen sein oder unterschiedliche Repräsentationen des gleichen Inhalts. So kann zum Beispiel realisiert werden, daß eine Videosequenz mit geringerer Farbtiefe, Auflösung oder Bildgröße wiedergegeben wird, falls die Wiedergabegeschwindigkeit auf einem langsamen Rechner zu gering ist.

Track Clipping Atom: Mit dem Track Clipping Atom wird die Fläche eines Tracks spezifiziert, die in einem Movie dargestellt werden soll. Jeder Track hat ein eigenes Track Clipping Atom, so daß sich ein Movie aus verschiedenen Flächen aus unterschiedlichen Tracks zusammensetzen kann.

Track Matte Atom: gibt für jedes Pixel an, wie stark das gewichtete Mittel aus dem Pixel des Tracks und dem entsprechendem Hintergrundpixel ist.

Edit Atom: enthält eine Liste von Einträgen die angeben wie sich der jeweilige Track aus den Daten seines Media zusammensetzt. Jeder Eintrag gibt Anfangszeit,Dauer und Wiedergabegeschwindigkeit für ein Segment des Tracks an.

4.1.3 Media Atom

Das Media Atom enthält Informationen darüber, mit welchem Algorithmen dieser Track abgespielt werden kann. Ferner enthält es Informationen, wo die Daten abgelegt sind. Dabei werden folgende Atome benutzt:

Handler Reference Atom: identifiziert den Media Handler der aufgerufen wird, um die Daten des

Tracks zu verarbeiten.

Media Information Atom: enthält Informationen über die Daten die in diesem Track enthalten sind. Da es eine Vielzahl dieser Atome gibt, die alle ähnlich organisiert sind, wird hier exemplarisch das Video Media Information Atom vorgestellt.

4.1.4 Video Media Information Atom

Dieses Atom repräsentiert die Videodaten eines Tracks. Es enthält neben Parametern zur Wiedergabe auch eine Tabelle mit Verweisen auf die eigentlichen Daten. Ein Video Information Atom besteht aus:

Video Information Header Atom: enthält allgemeine Daten über Größe, Version und Flags.

Handler Reference Atom: spezifiziert den Media Handler, der benutzt wird, um auf die Daten dieses Media zuzugreifen. Zusätzlich enthält dieses Atom eine Tabelle, in der Verweise und Informationen über die gespeicherten Daten enthalten sind.

Sample Table Atom: ist ein einzelnes Element einer zeitlich geordneten Folge von Daten. Das Sample Table Atom dient dazu, ausgehend von der Zeit im Movie, den Index des Sample und dann die Stelle an der es abgespeichert ist, zu berechnen.

4.2 Darstellung eines Movie

Bei der Darstellung eines Movie kann durch die Einschränkung der anzuzeigenden Fläche, der Angabe der Transparenz, der Verschiebung der Lage im Koordinatensystem sowie der Skalierung, Einfluß auf die Darstellung eines Tracks genommen werden.

Für jeden Track gibt die Track Boundery Reagion an, wie groß die anzuzeigende Fläche eines Tracks sein soll. Dabei setzt sich die Boundary Region aus der Track Clipping Region und dem Track Mate für jedes Pixel zusammen.

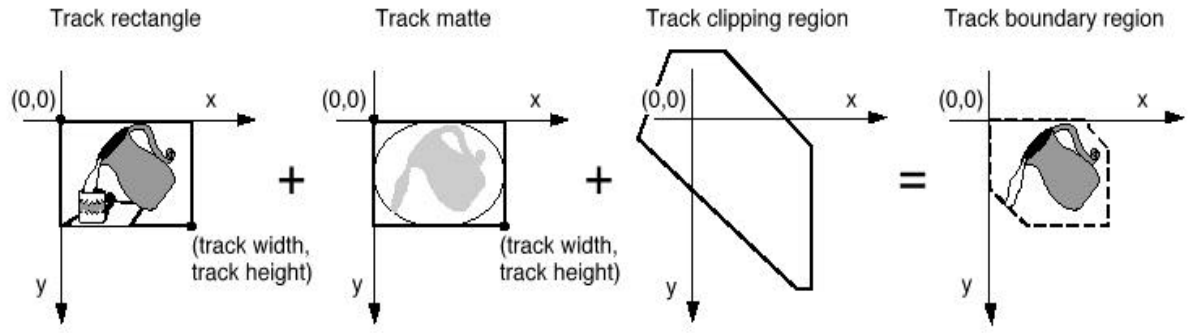


Abbildung 4-3: Aufbau der Track Boundery Region für einen Track.

Anschließend wird diese Track Boundary Region vom Track Koordinatensystem zum Movie Koordinatensystem überführt. Dazu wird eine Matrix, wie sie in Abbildung 3-4 dargestellt ist auf jedes Pixel (x,y) angewendet.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} a & b & u \\ c & d & v \\ t_x & t_y & w \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Abbildung 4-4: Matrix für die Pixeltransformation.

Hierbei sind u und v immer 0 und w ist immer 1. Verschiebungen bezüglich des Nullpunktes werden durch die Konstanten tx und ty ausgedrückt und Skalierungen und Rotationen weden durch übliche Matizenoperationen realisiert.

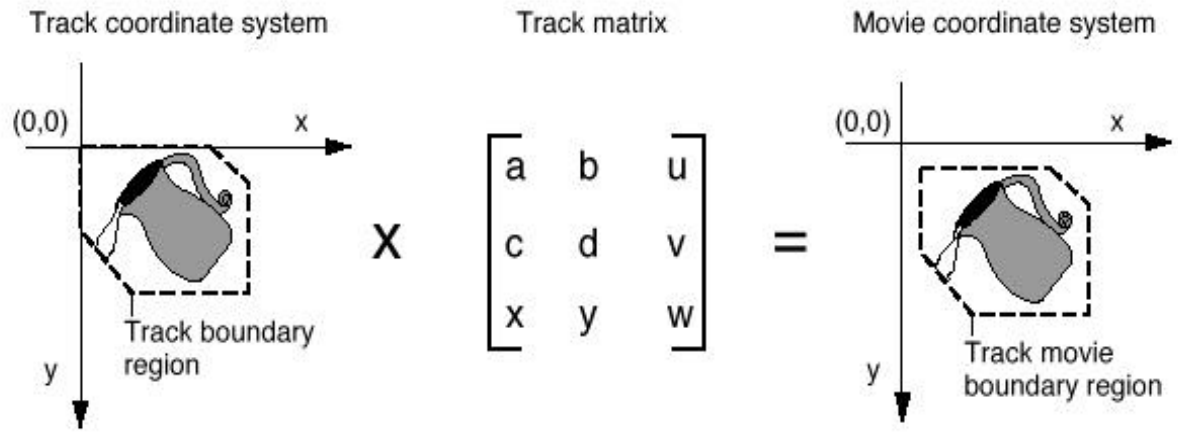


Abbildung 4-5: Umrechnung vom Track Koordinatensystem zum Movie Koordinatensystem.

Nachdem die einzelnen Tracks in das Movie Koordinatesystem überführt wurden, wird durch die Movie Clipping Region die Fläche angegeben, die als Movie dargestellt werden soll.

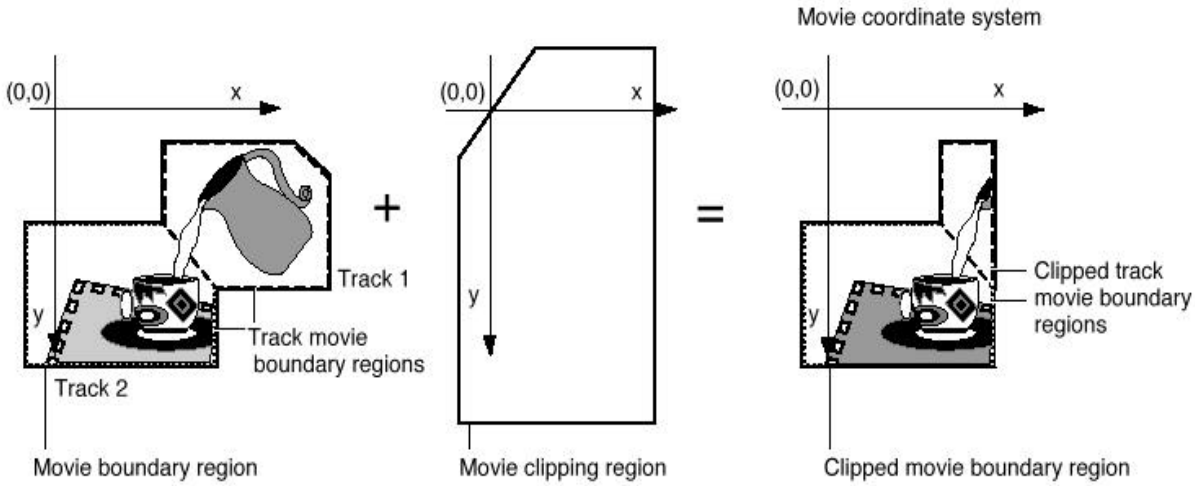


Abbildung 4-6: Clipped Movie Boundary Region.

Zum Schluß wird durch erneute Matrizenoperationen mit der schon bekannten Matrix das Movie in das Koordinatensystem der Bildschirmanzeige überführt. QuickDraw stellt dann das Movie auf dem Bildschirm dar.

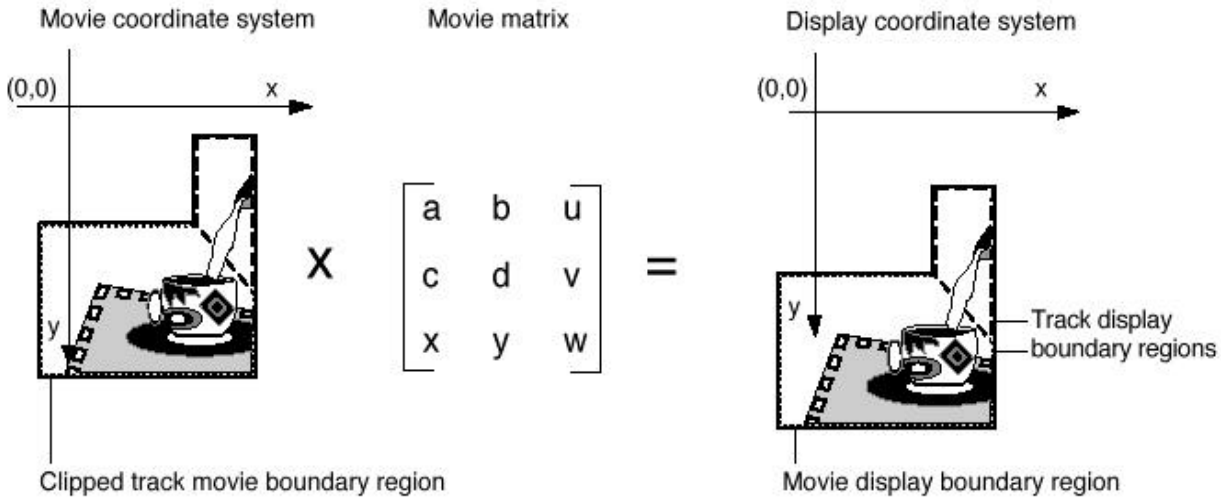


Abbildung 4-7: Überführung des Movie in das Bildschirmkoordinatensystem.

5 QuickTime 4.0

Im April 1999 brachte Apple QuickTime 4.0 als Public-Beta-Version heraus. Die Vorteile gegenüber QuickTime 3.0 sind im Wesentlichen die volle Implementierung der Steamingprotokolle RTP (Realtime Transport Protokoll) und RTSP (RealTime Streaming Protokoll) der Internet Engineering Task Force (IETF) und die Untrstützung des MPEG Audio Layer 3 MP3).

Durch die Implementierung der RTP und RTSP Protokolle ist nun echtes Streaming möglich. In QuickTime 3.0 war nur Progressives Streaming bzw. HTTP Streaming möglich. Beim HTTP Streaming werden die Moviedaten an den Betrachter geschickt und nach einer gewissen Zeit, wenn genügend Daten übertragen sind, wird mit dem Abspielen des Movie begonnen. Die Daten werden also lediglich gepuffert. Beim echten Streaming in QuickTime 4.0 werden erst Daten über das Movie geschickt so daß der Betrachter nur die Moviesequenzen bekommt, die er ausgewählt hat. Das hat den Vorteil, daß der Betrachter beim Darstellen des Movie springen kann und sich zum Beispiel erst das Ende des Movie anschauen kann und dann den Rest. Realisiert wird dieses durch das Hinzufügen von sogenannten Hint Tracks für jeden Medien Track. Der Hint Track enthält Informationen darüber, wie der Server die Daten senden soll.

Als weiterer Vorteil ist auch die Möglichkeit des Multicasting zu sehen, welches Live-Übertragungen von Video und Audiodaten ermöglicht. Ein Apple QuickTime Server kann nun bis zu 250 Streaming Dateien verwalten und 1000 Audio und Videoströme simultan anbieten.

6 QuickTime VR

QuicktTime VR ermöglicht es Dreidimensionale Quicktime Movies darzustellen und durch spezielle Eingaben interaktiv zu steuern. Es gibt zwei Formen von QuickTime VR Movies Panoramen und Objekte.

QuickTime Panoramen, stellen 360 Grad Rundumsichten von einem festen Standpunkt aus dar und können durch sogenannte Hot Spots verknüpft werden, so daß man von einem Panorama in das nächste gelangen kann. Innerhalb dieser Panoramen kann man sich zu allen Seiten hin umschauen und hinein- und herauszoomen.

QuickTime Objekte sind 3D Objekte, die man sich von allen Seiten aus betrachten kann, so als ob man das Objekt in der Hand halten würde. Panoramen können Objekte beinhalten und umgekehrt. Es ist also zum Beispiel möglich, daß man sich durch ein Panorama bewegt und von diesem Panorama zu einem 3D Objekt gelangt, auf dem ein QuickTime Movie abgespielt wird.

Ein QuickTime VR Movie wird aus Einzelbildern aufgebaut, die durch ein Autorensystem zu einem Movie zusammengefaßt werden. Dieses Erzeugen eines VR Movies wird in zwei Schritten vollzogen. Erst werden die Einzelbilder durch einen sogenanntes Stitching zusammengefaßt und anschließend wird duch einen sogenannten Dice Schritt ein QuickTime VR Movie erzeugt.

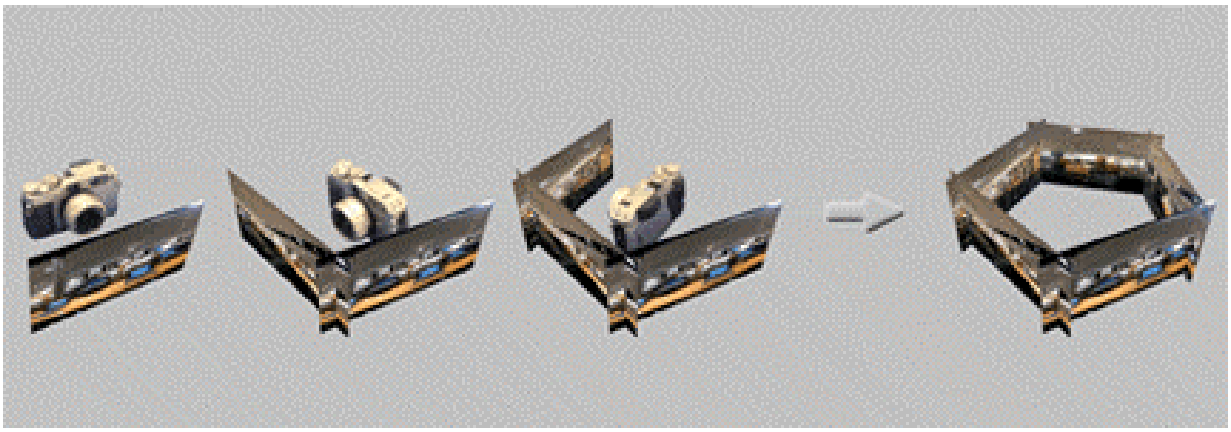


Abbildung 6-1: Aufnahme der Bilder für ein Panorama mit Hilfe einer Kleinbilkamera.

Abbildung 6-2: Einzelbilder mit 50% Überlagerung als Grundlage für das Stiching.

Durch das Autorensystem werden die einzelnen Bilder zu einem Panorama zusammengefaßt und auf eine Trommel projeziert, so daß die Übergänge zu den einzelnen Bildern fließend werden und Perspektive entsprechen. Der QuickTime VR Player bietet zusätzliche Korrekturoptionen die vom Benutzer ausgewählt werden können.

Abbildung 6-3: Panorama nach der Projektion auf eine Trommel.

In der Abbildung 6-3 kann man deutlich die Verzerrungen erkennen, die durch das Stitching entstanden sind. Die Tastatur im Vordergrund zum Beispiel ist stark nach hinten gewölbt. Als Vergleich dazu kann man in der Abbildung 6-4 die Veränderungen anhand des Ausschnitts in der Mitte sehr gut erkennen.

Abbildung 6-4: Fertiges Panorama mit Ausschnitt des Originalbildes zum Vergleich.

Nach dem Zusammenfassen der Einzelbilder zu einem Panoramabild wird nun in einem Dice Schritt ein VR Movie erzeugt. Hierbei fügt das Autorentool spezielle Tracks ein, die es dem VR Player ermöglichen, den passenden Ausschnitt darzustellen.

Die Erstellung eines VR Objektes geschieht im Prinzip genauso, allerdings werden die Einzelbilder nicht auf eine Trommel projeziert sondern schichtweise auf eine Kugel. Die Abbildung 6-5 zeigt eine solche Projektion.

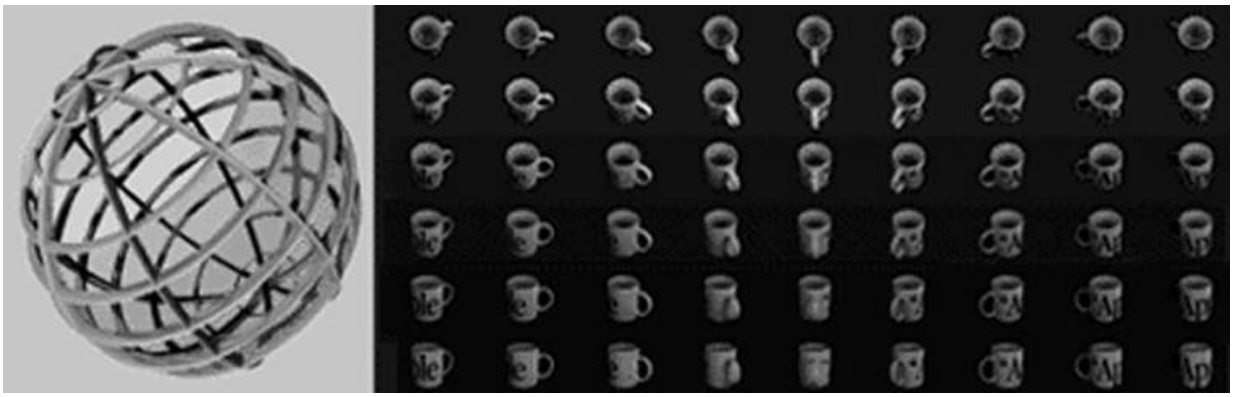


Abbildung 6-5: Schichtweise Projektion der Einzelbilder auf eine Kugel

Die Bildqualität eines VR Movie ist bedingt durch die Verwendung von Einzelbildern als Basismaterial recht hoch. Zusätzlich kommt auch die Art der Darstellung der Qualität zugute, da bei der Betrachtung des VR Movie immer nur ein Ausschnitt des Panoramas dargetellt wird und nicht eine komplette Videosequenz. So ist es auch möglich, recht kleine Panoramen zu erstellen, die mitunter nur 1MB an Speicherplatz benötigen.

7 Literatur

[Appl 99a] Apple Computer Inc.:
http://www.apple.com/qiucktime [Stand: Mai 1999]

[Appl 99b] Apple Computer Inc.:
http://developer.apple.com/techpubs/quicktime/qtdevdocs/ [Stand: Mai 1999]

[Appl 93c] Apple Computer Inc.:
Inside Macintosh: QuickTime, Addison-Wesley, 1993

[Appl 93d] Apple Computer Inc.:
Inside Macintosh: QuickTime Components, Addison-Wesley, 1993