Analyse von Multimedia–Datenströmen

Technische Universität Dresden

Großer Beleg

Fakultät Informatik

Mike Hanitsch

Matrikelnummer: 2293977

Dresden, den 15.02.1999

Inhalt

1 Zielstellung und Vorüberlegungen

1.1 Einleitung

In dieser Arbeit werden Multimedia–Datenströme unter dem Gesichtspunkt analysiert, inwieweit diese in sinnvollen Einheiten übertragen werden können. Das Übertragungsmedium selbst spielt dabei eine untergeordnete Rolle und kann beispielsweise ein Datenbus oder ein Netzwerk sein. Dabei sollen spezielle Eigenschaften herausgefunden werden, die sich zwangsläufig aus der Kompression solcher Datenströme ergeben. Das Ergebnis dieser Arbeit soll ein Softwaretool sein, das einige ausgewählte Formate erkennt, untersucht und eine sinnvolle Zerlegung ermittelt. Es soll eine Bibliothek implementiert werden, die sich in Zukunft leicht durch neue Formate ergänzen läßt. Ein in Paper [6] entwickeltes Programm soll die Ergebnisse dieser Arbeit dann für weitere Berechnungen benutzen können. Das Übertragungsmedium selbst steht dabei nicht zur Debatte und ist als solches in einer gesonderten Arbeit zu untersuchen. Da die Datenströme in der Regel in komprimierter Form vorliegen, soll dem Leser auch ein kleiner Überblick über Kompressionsverfahren vermittelt werden. Weiterhin wird untersucht, inwieweit sich die Ergebnisse dieser Arbeit mit den im DROPS–Projekt (siehe 1.5) getroffenen Annahmen decken.

1.2 Multimedia–Datenströme

Multimedia ist gegenwärtig ein häufig gebrauchter Begriff, und trotzdem erhält man auf die Frage nach der Definition dieses Begriffes verschiedene Antworten. Wie ist Multimedia nun eindeutig bestimmt? Diese Frage ist nicht leicht zu beantworten, und in je mehr Büchern man eine Antwort sucht, um so länger und verschiedenartiger wird diese. Die Definition hängt jedoch im wesentlichen von der Sicht des Betrachters bzw. des betrachteten Kontextes ab. Deswegen werde ich hier eine Erklärung geben, die dem Inhalt der Arbeit gerecht wird. Im wesentlichen geht es um mehrere (multi) Medien (media).

Bei den hier untersuchten Datenströmen geht es speziell um zwei Medien, nämlich Video und Audio, die einzeln oder in Kombination auftreten können. In der Regel handelt es sich um beliebig große Datenmengen, die möglichst in einer konstanten Datenrate zum Verbraucher übertragen werden müssen, damit eine sinnvolle

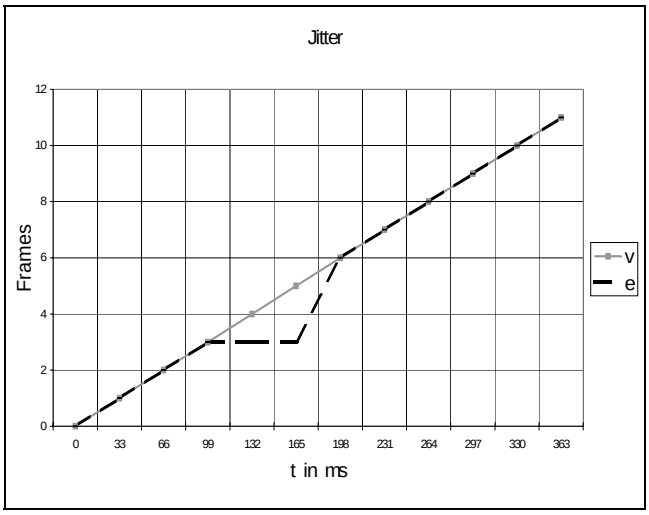
Nutzung gewährleistet ist. In den meisten Fällen unterliegen verschiedenartige Datenströme zusätzlich zeitlich sehr strengen Synchronisationsbedingungen (z. B. Lippensynchronität im Video mit den hörbaren Audiosignalen).
Weiterhin sind diese Daten komprimiert, um die ohnehin schon riesigen Datenmengen zu verringern. Die durch die Kompression entstehenden Datenströme ändern sich dabei entscheidend in ihrer Bedeutung, so daß die sinnvolle Partitionierung ein wesentlicher Bestandteil dieses Beleges ist.

1.3 Übertragung von Multimedia–Datenströmen

Die Übertragung von Multimediadaten unterliegt besonders strengen Anforderungen, die im folgenden vorgestellt und kurz erläutert werden.

- Synchronität:
  Der Datenstrom unterliegt einem fest vorgegebenen zeitlichen Verhalten. Daraus folgt unmittelbar, daß Empfangsrate = Senderate gelten muß.

- Geringer jitter:
  Jitter bezeichnet das zeitliche Verhalten zwischen nachfolgenden Datenpaketen. Werden Pakete z. B. in einem Takt von 20 ms abgesandt, so soll der Abstand der eingehenden Datenpakete beim Empfänger ebenfalls 20 ms betragen. Tritt jetzt jedoch der Effekt des jitter auf, dann kann es durchaus sein, daß für 60 ms keine Pakete mehr beim Empfänger ankommen, während dann quasi die drei ausstehenden Pakete zum gleichen Zeitpunkt unmittelbar hintereinander ohne Verzug ankommen. Ein solches Verhalten ist für Medienströme nicht tolerierbar, denn die Wiedergabe ist dann für 60 ms gestört, da die entsprechenden Datenpakete fehlen. Andererseits sind die verspätet eintreffenden Pakete dann nicht mehr für die Wiedergabe brauchbar, da sie bereits veraltet sind, siehe hierzu auch Abbildung 1-1.

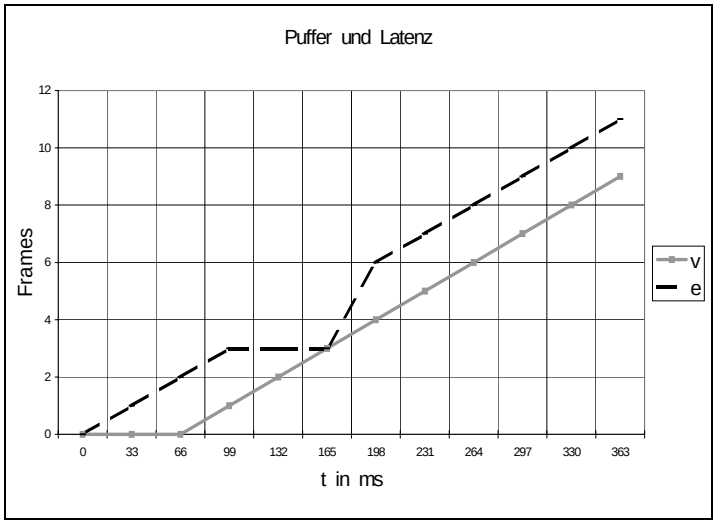


Man kann hier erkennen, daß der Verbraucherprozess kurzzeitig nicht arbeiten kann, da der Erzeugerprozess die nötigen Frames nicht rechtzeitig liefern kann. Im Mittel jedoch gibt es keine Probleme. Die zu spät eingetroffenen Frames müssen verworfen werden.

Abbildung 1-1 Jittereffekt

- Geringe Latenz:

  Durch die Latenz ist das zeitliche Verhalten am Beginn einer Übertragung definiert. Je nach Typ des Medienstrom benötigt ein Empfänger mehr oder weniger viel Information, um aus den empfangenen Daten tatsächlich eine Wiedergabe produzieren zu können. Bei einer Videoübertragung ist der Start erst dann möglich, wenn ein komplettes Anfangsbild übertragen wurde. Ist die zu sendende Datenmenge am Anfang einer Übertragung sehr hoch, so ergibt sich eine sehr große Latenz. Dies ist bei einer statischen Übertragung (z. B. Videofilm) nicht von besonders großer Bedeutung, bei hoch interaktiven Anwendungen (Telefonie und Videotelefonie) kann dies jedoch die Benutzer stören, da die Übertragung teilweise stockt.



Durch das Füllen eines Puffers, um eventuell auftretende Schwankungen auszugleichen, entsteht eine geringe Latenzzeit. Je nach Anwendung ist das Verhältnis zwischen Puffergröße und Latenzzeit abzuwägen.

Abbildung 1-2 Latenzzeit und Puffer

- Kleiner Puffer:
  Das Übertragungsmedium bietet kurzfristig kein deterministisches Verhalten. Das heißt, die Übertragung kann kurzfristig durch Störungen beeinträchtigt werden, die durch z. B. einen parallel angeforderten Datenstrom entstehen können. Ist längerfristig, sozusagen als "Durchschnittswert", eine entsprechende Übertragungsgüte, das heißt Sende-/Empfangsrate und Fehlerrate vorhanden, so können durch einen zeitlichen "Vorsprung" bei der Übertragung, kurzzeitige Störungen ausgeglichen werden. Dies erfolgt durch den Einsatz eines Puffers. Bei der Übertragung wird also zuerst dieser Puffer (z. B. 1 s Übertragung) gefüllt, bevor die Wiedergabe beim Benutzer erfolgt. Treten jetzt während der Übertragung vereinzelte Störungen auf, so kann der Empfänger durch entsprechende Strategien der Fehlerbehebung hierauf reagieren (siehe Abbildung 1-2). Je größer der Puffer ist, um so besser sind zeitweilige Störungen ausgleichbar. Andererseits bedeutet ein großer Puffer eine Steigerung der Latenz, was zuvor als nachteilig dargestellt wurde.

- Fehlerfreiheit, Vollständigkeit:
  Diese Zielvorgaben sind grundsätzlich immer wünschenswert, sofern die unabdingbaren Zielvorgaben nicht verletzt werden, denn durch eine fehlerfreie und vollständige Übertragung ergibt sich in der Regel eine bessere Übertragungsqualität. Somit können mehr Bilder pro Zeiteinheit übertragen werden, da möglicherweise fehlerhafte Daten nicht verworfen werden müssen. Daneben wird die Qualität eines einzelnen Bildes auch besser, da Änderungen des Bildes im zeitlichen Verhalten immer richtig berücksichtigt werden.

Diese Betrachtungen sind allgemein auf jedes Übertragungsmedium anwendbar, egal ob die Übertragung von Festplatte zu Hauptspeicher oder die Übertragung zwischen zwei verschiedenen Rechnern über ein lokales Netzwerk betrachtet wird. Es ist lediglich eine Gewichtung der einzelnen Faktoren vorzunehmen, die wiederum vom Medium selbst abhängt.

1.4 QoS–Quality of Service (Dienstgüteparameter)

Niemand möchte, daß sein Lieblingsfilm im Fernsehen durch ruckelnde Sequenzen heimgesucht wird, oder daß der im Radio gespielte Lieblingssong Lücken aufweist. Doch gerade auf solche Probleme stößt man, wenn Multimedia–Datenströme, z. B.

über das Internet, in Echtzeit übertragen werden sollen. Damit es gerade nicht zu solchen Störungen kommt, müssen Sender und Empfänger vor der Übertragung gewisse Vereinbarungen treffen. Dabei werden erst einmal relevante Informationen ausgetauscht. Je nach Situation beziehen sich die Informationen dabei auf das Übertragungsprotokoll, das Übertragungsmedium und natürlich auf die zu übertragenden Daten. Der Empfänger könnte z. B. Informationen über die Latenzzeit der Daten sowie über das jitter–Verhalten des Übertragungsmediums erhalten. Daraufhin berechnet er für seine Seite die Größe des Puffers, die er für eine störungsfreie Verarbeitung der Daten benötigt. Die erhaltenen Informationen müssen allerdings sicher sein, das heißt der Sender und das Übertragungsprotokoll haben dafür Sorge zu tragen, daß die übermittelten Informationen für die Dauer der Übertragung eingehalten werden.

An dieser Stelle soll auch auf die Arbeit [6] verwiesen werden, die einige mathematische Modelle für die Auswertung der Informationen auf Empfängerseite vorstellt. Bei der Analyse der zu übertragenden Multimedia–Datenströme wird insbesondere darauf Wert gelegt, daß die analysierten Daten so abgelegt werden, daß sie in [6] ausgewertet werden können, um zwischen 2 Partnern eine QoS–Vereinbarung zu treffen.

1.5 Die betrachtete Übertragungsumgebung: DROPS

DROPS steht für Dresden Real Time Operation Systems. Während sich viele Arbeiten bezüglich QoS mit den für Übertragung verwendeten Medien und Protokollen auseinandersetzen, ist das Ziel des DROPS–Projektes die Untersuchung von QoS–Zusagen auf Basis von verteilten Betriebssystemen. Ausgangspunkt ist dabei die Möglichkeit, daß einzelne Komponenten des Systems eine gewisse Dienstgüte zusagen können (siehe hierzu auch [20]).

Unter Verwendung der von den einzelnen Komponenten zur Verfügung gestellten Dienstgüteparameter und den Ergebnissen der Analyse des zu übertragenden Datenstromes werden dann mit dem in Paper [6] vorgestellten Modell weitere Parameter auf Empfängerseite errechnet. Die von dieser Arbeit gelieferten Analyseergebnisse umfassen dabei eine Liste von Paketgrößen, in die der jeweilige Bitstrom für die Übertragung zerlegt wird. Des weiteren werden folgende den Bitstrom allgemein kennzeichnende Parameter benötigt:

- Dgesamt:
  Dieser Wert beschreibt die Gesamtgröße des zu übertragenden Datenstroms und wird in Byte angegeben.

- anz_item:
  Die Variable gibt die Anzahl der zu übertragenden Pakete an, in welche der Bitstrom zerlegt wird.

- InputQ:
  Der Parameter gibt die Größe des Quantums an und wird ebenfalls in Byte angegeben. Er ist abhängig vom verwendeten Übertragungsmedium bzw. den Kenndaten des Absenders und spiegelt genaugenommen die Eingangsrate wieder. Diese Variable wird im folgenden nicht weiter betrachtet.

- InputTB:
  InputTB beschreibt die Periode des Verbraucherprozesses und ist bei den Datenströmen von der Arbeitsweise des Decoders abhängig. In der Regel erfolgt die Angabe in Millisekunden.

Zu weiteren Details soll an dieser Stelle noch mal auf Paper [6] verwiesen werden, welches sich intensiv mit mathematischen Modellen hierzu beschäftigt und darüber hinaus eine Implementierung vorstellt, welche die analytischen Ergebnisse der vorliegenden Arbeit auswertet.
Im einzelnen spielt es dabei keine Rolle, welche konkrete Komponente betrachtet wird, da dieses Verfahren eigentlich generell bei der Übertragung von A nach B angewendet werden soll. Es kann dabei natürlich zu einer längeren Kette kommen (von A über B nach C), die Parameter werden trotzdem immer mit dem konkreten Partner ausgehandelt. Beim derzeitigen Stand der Entwicklung ist jedoch noch nicht eindeutig geklärt, ob einzelne Komponenten auf Grund besonderer Eigenschaften mit diesem Modell harmonieren. Es besteht also noch die Möglichkeit von Speziallösungen, um besonderen Gegebenheiten gerecht zu werden.

2 Kompressionsverfahren

2.1 Einleitung

Im folgenden wird ein kurzer Einblick in heute gängige Kompressionsverfahren gewährt. Da im Vordergrund dieser Arbeit die Analyse von Multimedia–Datenströmen steht, ist es von essentieller Bedeutung, die durch die angewendeten Kompressionsverfahren entstehende semantische Bedeutung des Datenstroms zu verstehen.

2.2 Charakterisierung

Betrachtet man heute übliche Multimedia–Anwendungen, wie z. B. Videokonferenzen über das Internet, dann wird schnell klar, daß die Daten nicht in ihrer ursprünglichen digitalen Form gespeichert oder übertragen werden können. Z. B. benötigt ein Videobild (je nach Fernsehnorm) in etwa zwischen 750 und 920 KByte, daraus ergibt sich ein Datenstrom von 20–22 MByte pro Sekunde Videofilm. Es geht also im wesentlichen darum, den Datenstrom zu komprimieren. Das Ergebnis wird dann gespeichert oder übertragen. Diese so abgelegten oder übertragenen Daten werden dann in irgendeiner Form weiterverarbeitet. Dabei wird versucht, aus den komprimierten Daten je nach verwendeten Kompressionsverfahren eine dem Original ähnliche oder identische Kopie, zu rekonstruieren. Im wesentlichen ergeben sich dabei die folgenden charakteristischen Kriterien für ein Kompressionsverfahren.

- Verlustfreie Kompression:
  Bei diesem Verfahren läßt sich aus den komprimierten Daten eine identische Kopie rekonstruieren.

- Verlustbehaftete Kompression:
  Das Original wird auf eine Art und Weise komprimiert, so daß keine identische, sondern nur eine ähnliche Kopie rekonstruiert werden kann. Bei der Kompression werden Informationen entfernt, die rein subjektiv gesehen, vom menschlichen Wahrnehmungssystem nicht erkannt werden. Da das natürlich von Betrachter zu Betrachter unterschiedlich sein kann, ist keine direkte qualitative Bewertung möglich. Ziel sollte es jedoch sein, daß Original und Kopie für den Betrachter möglichst identisch erscheinen.

- Hybride Kompressionsverfahren:
  Dieses Verfahren stellt eine Mischung von verlustfreien und verlustbehafteten Verfahren dar.

- Kompressionsgrad:
  Es gibt zwei gebräuchliche Varianten, um den Kompressionsgrad zu beschreiben. Zum einen ist es üblich, die Anzahl der Bits pro Pixel im komprimierten Datenstrom anzugeben. Als Beispiel stelle man sich ein Bild mit folgenden Eckdaten vor: 256 x 240 Pixel bei einer Farbtiefe von 8 Bit. Die resultierende Datengröße wäre dann 256 x 240 x 8 = 491520 Bit. Wird das Bild um das vierfache komprimiert ergibt sich eine Datengröße von 122880 Bit. Die resultierende Pixeltiefe ergibt sich dann aus 122880 / 256 / 240 = 2 Bit/Pixel. Man sagt dann, daß das Bild von 8 Bit/Pixel auf 2 Bit/Pixel komprimiert wurde. Die andere und häufiger genutzte Möglichkeit besteht darin, das Verhältnis der Datenmenge zwischen Original und komprimierter Version anzugeben. Im obigen Beispiel würde man dann von einer Kompressionsrate von 4 : 1 sprechen. Letztere Möglichkeit ist natürlich universeller, da man nicht immer in Pixeln rechnen kann. Man denke hierbei an Audiodaten, die man sich, wenn überhaupt, nur äußerst schwer in Pixeln vorstellen kann.

- Symmetrische Kompressionsverfahren:
  Es handelt sich hierbei um Verfahren, bei denen die benötigten Zeiten für Kompression und Dekompression annähernd gleich lang sind. Ein denkbares Szenario wäre beispielsweise eine Videokonferenz, bei der Kompression und Dekompression in Realzeit erfolgen müssen.

- Asymmetrische Verfahren:
  Im Gegenteil zu den symmetrischen Verfahren unterscheidet sich die Zeitdauer von Kompression und Dekompression zum Teil erheblich. In der Regel ist das Ergebnis des hohen Rechenaufwandes ein deutlicher Gewinn an Qualität. Weiterhin sei noch bemerkt, daß die Geschwindigkeit von Kompression und Dekompression eigentlich nur noch bei der Bewegtbildkompression eine Rolle spielt. Standbilder kann man beim derzeitigen Stand der Technik in Realzeit sowohl komprimieren als auch dekomprimieren.

Details hierzu und zu den folgenden Kapiteln sind unter anderem in [2] und [8] zu finden.

2.3 Verlustfreie Kompression

2.3.1 Statistische Kodierung

Hier wird der Sachverhalt genutzt, daß im Datenstrom auftretende Bitfolgen nicht gleich verteilt sind. Das Prinzip beruht darauf, häufiger auftretende Bitfolgen durch kurze Bitfolgen zu ersetzen, während selten vorkommende Bitfolgen durch längere ersetzt werden. Damit eine eindeutige Dekodierung gewährleistet ist, darf kein Kodewort Anfangswort eines anderen sein. Dieses Prinzip ist am bekanntesten unter dem Namen Huffman–Kodierung und ist zu den symmetrischen Verfahren zu zählen.

2.3.2 Arithmetische Kodierung

Auch hier werden statistische Eigenschaften von auftretenden Bitfolgen genutzt. Als Erweiterung zur statistischen Kodierung werden noch arithmetische Beziehungen zwischen den Kodewörtern genutzt. Dies bringt einen höheren Kompressionsgrad, hat aber den Nachteil, daß immer nur Gruppen von Bitfolgen kodiert bzw. dekodiert werden können. Das Verfahren ist ebenfalls als symmetrisch einzuordnen.

2.3.3 Modellbasierende Kodierung

Ein allgemein bekanntes Verfahren ist es, mathematische Formeln durch grafische Modelle zu visualisieren. Auf dieser Idee aufbauend, wird bei der Kodierung versucht, die zu komprimierenden Daten (z. B. Sound, Grafik, Bitfolgen) als mathematisches Modell zu beschreiben. Problematisch ist es jedoch, immer ein passendes Modell zu finden. Gelingt dies jedoch, dann sind die erreichbaren Kompressionsraten natürlich sehr hoch. Es wird auch versucht, Bilder oder Teile von Bildern durch fraktale Beschreibungen zu kodieren. Gerade letzteres sorgt aber auch für sehr hohe Rechenzeiten, weswegen das Verfahren als asymmetrisch zu bezeichnen ist.

Generell ist es weiterhin als verlustfrei einzuordnen, obwohl es schwer ist, immer ein korrektes Modell zu finden. Häufig werden nur annähernde Modell gefunden, die verbleibende Differenz erfährt dann eine Sonderbehandlung. Eine sehr triviale Möglichkeit wäre es, geringe Differenzen zu verwerfen. Dann aber wäre das Verfahren nicht mehr verlustfrei.

2.3.4 Lempel–Ziv–Technik

Wie der Name schon sagt basiert dieses Verfahren auf Algorithmen von Lempel und Ziv. Anwendungsgebiet ist das Packen und Entpacken von Daten auf Festplatte. Der zu speichernde Bitstrom wird dabei auf identische Bitfolgen, sogenannte Phrasen, untersucht. Gefundene Phrasen werden in einer Phrasentabelle gespeichert. Im Bitstrom selbst werden die Originalphrasen durch Zeiger auf die Phrasentabelle ersetzt. Aufgrund der verwendeten Zeigerarithmetik ist diese Technik für Festplatten besonders gut geeignet. Das Verfahren kann als symmetrisch eingeordnet werden.

2.3.5 Lauflängenkodierung

Diese Kompressionstechnik beruht auf dem Sachverhalt, das häufig Pixelfolgen mit gleichen Werten auftreten. Ein sehr anschauliches Beispiel wäre ein Landschaftsbild mit klarem blauen Himmel als Hintergrund. Im Bitstrom bräuchte man nun nach dem ersten Pixel nur noch die Information hinzufügen, wie oft dieser Pixel folgt, und hätte vielleicht schon ein Drittel des Bildes komprimiert. Bei Bildern mit großen gleichfarbigen Flächen ist hier ein sehr hoher Kompressionsgrad zu erreichen. Das Verfahren ist als symmetrisch einzuordnen.

Die bisher vorgestellten Verfahren finden hauptsächlich in der Standbildkompression Anwendung, wurden aber trotzdem mit aufgeführt, um einen kleinen Überblick zu verschaffen. Betrachtet man eine Videosequenz, müßten auf diese Weise ca. 30 Bilder pro Sekunde komprimiert werden. Es ist leicht zu sehen, daß das nicht besonders effektiv ist. Trotzdem werden Verfahren der Standbildkompression immer noch als Grundlage der Bewegtbildkompression eingesetzt.

2.4 Verlustbehaftete Verfahren

2.4.1 Prädiktive Kompressionsverfahren

Es handelt sich hierbei um ein Verfahren, das auf Grundlage von Beobachtungen vorangegangener Werte Voraussagen über die folgenden Werte trifft. Das Anwendungsgebiet ist dabei abstrakt zu sehen, die Voraussagen können auf Pixelebene als auch auf Bildebene getroffen werden. Vergegenwärtigt man sich die in der Regel geringen Unterschiede zwischen den Bildern eines Videos (30 Bilder pro Sekunde!), dann ist die hohe Effektivität leicht einzusehen. Zwei wichtige Vertreter

dieses Verfahrens werden nun genauer vorgestellt, es handelt sich dabei um Motion–Compensation und DPCM.

- Motion–Compensation:
  Grundidee dieser Methode besteht darin zu prüfen, ob sich Teile von Bildern innerhalb einer Bildfolge so wiederholen, daß diese Informationen aus vorangegangen Bildern bezogen werden können. Durch die Angabe eines Verschiebungsvektors (siehe Abbildung 2-1) können so Bildteile aus den kodierten Informationen eines Vorgängerbildes rekonstruiert werden. Jedoch benötigt jede so komprimierte Bildfolge mindestens ein komprimiertes Bild, welches alle Informationen beinhaltet.

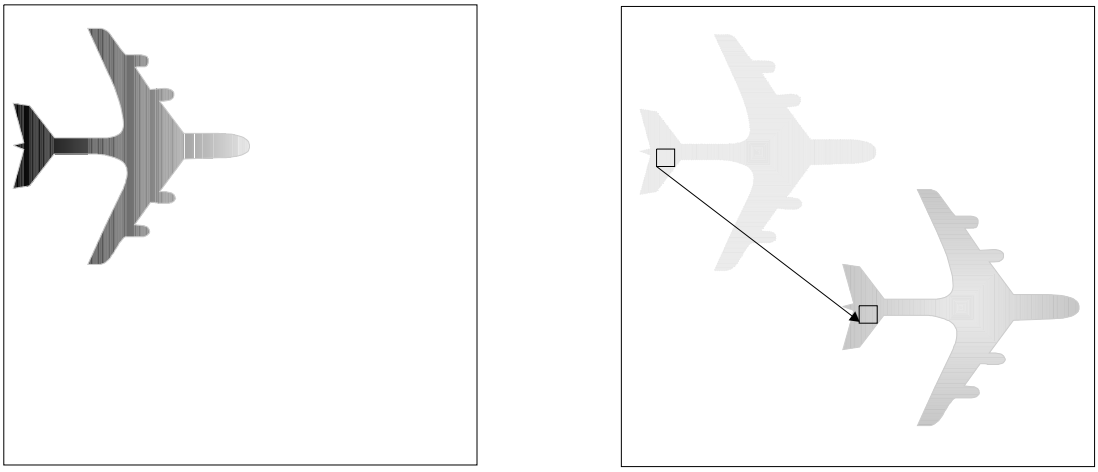


Abbildung 2-1 durch Verschiebungsvektoren gewonnene Bildinformationen

- DPCM (Differential Pulse Code Modulation):
  DPCM wurde für die Digitalisierung und Kompression von Audiosignalen entwickelt. Auch bei der Abtastung von Audiosignalen konnte beobachtet werden, daß es in der Regel effektiver ist, die Differenzen zwischen den abgetasteten Werten anstatt die Werte selbst zu speichern. Probleme bereiten jedoch sprunghafte Wertänderungen. In einer Verfeinerung des Verfahrens, dem ADPCM (Adaptive Differential Pulse Code Modulation), wird daher eine Vorausbetrachtung der Abtastwerte durchgeführt. Sollte es zu sprunghaften Änderungen der abgetasteten Werte kommen, wird die Abtastrate verringert. Dieses Prinzip kann auch für die Kompression von Bilddaten genutzt werden.

2.4.2 Frequenzorientierte Techniken

Diese Techniken nutzen den Umstand, daß die menschliche Wahrnehmung, zumindest was den audiovisuellen Bereich betrifft, gegenüber unterschiedlichen Frequenzen eine unterschiedliche Sensitivität aufweist.

- Subband–Kodierung:
  Eine Technik besteht darin, das wahrzunehmende Frequenzband in Subbands zu zerlegen. Je nach der menschlichen Sensitivität werden einzelne Subbands genauer kodiert als andere.

- DCT–basierende Kodierung (Discrete Cosine Transform):
  Dieses Verfahren ist sehr bedeutungsvoll für die Bilddatenkompression. Hierbei handelt es sich um eine spezielle Form der Fourier–Transformation. Bei diesem System werden Pixel in Blöcken gruppiert. Die Blöcke beinhalten meist 8 x 8 oder 16 x 16 Pixel. Die entstehenden Blöcke werden wiederum in Koeffizienten transformiert, welche wieder kodiert und übertragen werden. Der Algorithmus filtert dabei nur `nützliche‘ Informationen heraus, so daß am Ende die Informationen eines Blockes in wenigen Koeffizienten konzentriert vorliegen. Der hierbei effektivste Algorithmus verwendet dabei die Karhunen–Loeve–Transformation (KLT), welche allerdings sehr rechenintensiv ist. DCT ist wesentlich weniger rechenintensiv sowie nahezu optimal und findet aus diesem Grund eine weit verbreitete Anwendung.

2.4.3 Bedeutungsorientierte Techniken

Verfahren dieser Kategorie sind gegenwärtig noch Gegenstand umfangreicher theoretischer und psychologischer Untersuchungen. Sie finden bei der Kompression von Bilddaten Anwendung und beruhen auf der Einsparung unwichtiger Informationen. Die Forschungen auf dem Gebiet beziehen sich dabei auf die Analyse und Einstufung der Bildinformationen als wichtig oder unwichtig. Da hierbei immer Informationen des Originals verworfen werden, sind Verfahren dieser Kategorie generell als verlustbehaftet einzustufen. Weiterhin ist der Rechenaufwand für das Auffinden unwichtiger Informationen in der Regel sehr hoch, so daß es sich weiterhin um asymmetrische Techniken handelt.

- Filterung:
  Ähnlich wie bei der Subband–Kodierung werden Signale weggefiltert, die vom menschlichen Wahrnehmungssystem nicht oder nur kaum wahrgenommen werden, da nicht wahrnehmbare Informationen eindeutig als unwichtig interpretiert werden können.

- Bitzuweisung:
  Hier werden im Prinzip wichtige Bildbestandteile mit mehr Bits kodiert als weniger wichtige. Momentan werden insbesondere Kanten und Konturen als besonders wichtig eingestuft und demzufolge mit einer großen Genauigkeit kodiert. Die Kompression wird über die geringere Kodierung der dazwischenliegenden Flächen erreicht, da diese ja in der Regel auch den größeren Bildanteil ausmachen.

- Subsampling:
  Es handelt sich hierbei um ein Verfahren, das Bestandteil der meisten hybriden Kompressionsverfahren für Bilddaten ist. Subsampling bezieht sich auf die Art und Weise der Abtastung. Bei Signalen wird im wesentlichen zwischen wichtigen und unwichtigen Signalen unterschieden. Unbedeutende werden dann in größeren Schritten abgetastet, währenddessen wichtige Informationen mit kleinen Schritten abgetastet werden. Beispielsweise reagiert das menschliche visuelle Wahrnehmungssystem sehr empfindlich auf Helligkeitsunterschiede, im Vergleich dazu fällt die Reaktion auf Farbunterschiede wesentlich schwächer aus. Dementsprechend würde ein Subsampling–Verfahren die Helligkeit in relativ kleinen und die Farbinformationen in wesentlich größeren Schritten abtasten.

- Quantisierung:
  Durch Rundung abgetasteter auf vorgegebene Werte wird praktisch die Anzahl möglicher Werte reduziert. Indem man die Zahl der möglichen Werte beeinflußt, kann man auf die entstehende Datenmenge Einfluß nehmen, was bei einigen Verfahren auch in Echtzeit praktiziert wird. Beim Digitalisieren kommt es zwangsläufig zu einer Quantisierung, da die analogen Werte nur mit endlich vielen Bits kodiert werden können. Weitere Reduktion erreicht man durch das Abschneiden niederwertiger Bits oder die Verwendung von Schwellenwerten.

2.5 Hybride Kompression

Die meisten in der Praxis eingesetzten Verfahren fallen in diese Kategorie. Es handelt sich in der Regel um eine Kombination der bisher erläuterten Techniken. Einige bekannte Vertreter sind dabei:

- JPEG,
- H.261 / H.263,
- MPEG,
- DVI,
- CD–I.

Einige Vertreter werden in späteren Kapiteln konkreter vorgestellt. Dabei wird auf die verwendeten Kompressionsverfahren hingewiesen. Des weiteren werden diese Verfahren konkret auf ihre Partitionierbarkeit hin untersucht, welche ja eng mit der verwendeten Kompressionstechnik zusammenhängt.

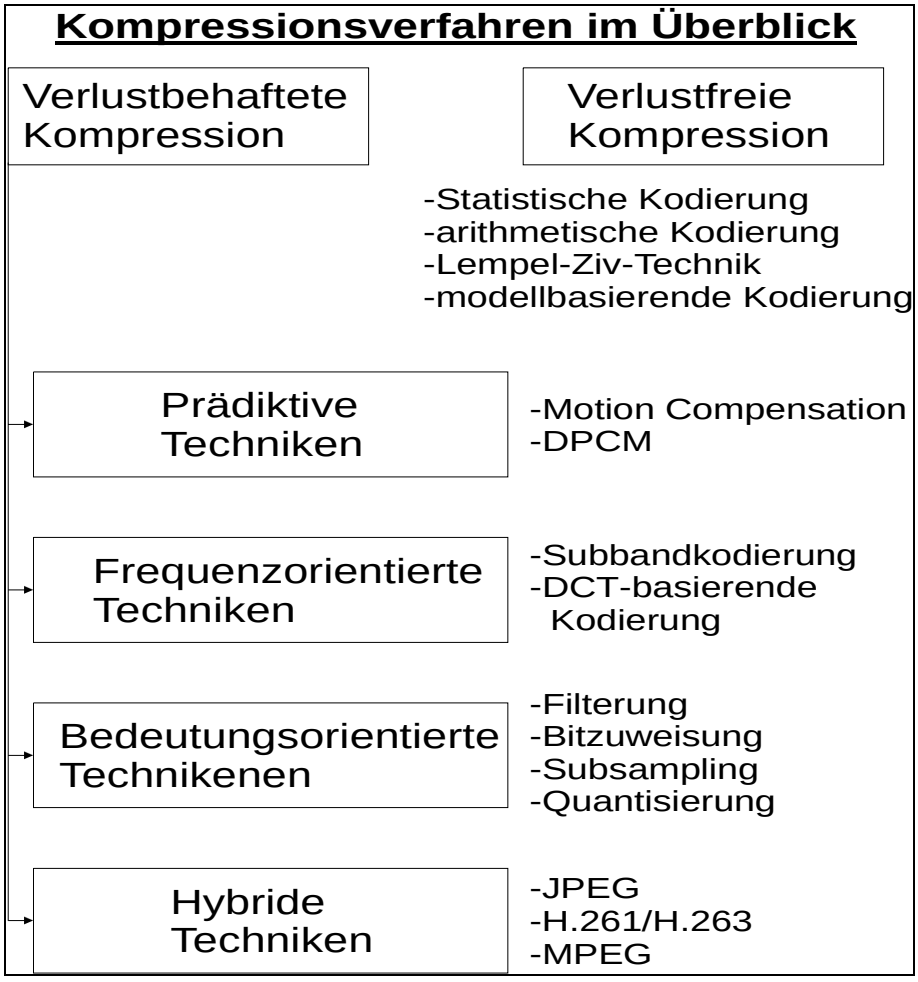


Abbildung 2-2 Überblick über Kompressionsverfahren

2.6 Standbildkompression

Hohe Kompressionsraten werden derzeit nur durch hybride Verfahren erreicht. Dabei werden redundante Informationen entfernt. Es wird dabei zwischen räumlicher und zeitlicher Redundanz unterschieden. Verfahren der ersten Kategorie werden unter dem Begriff Intraframekompression, oder auch Standbildkompression zusammengefaßt, und werden immer auf einzelne Bilder angewendet. Typische angewendete Techniken sind dabei DCT–basierende Verfahren, die Huffmann–Kodierung als auch prädiktive Verfahren.

2.7 Bewegtbildkompression

Bei Videoströmen handelt es sich vom Prinzip her um eine Folge von Einzelbildern, die allerdings in sehr kurzen Abständen aufeinanderfolgen. In der Regel handelt es sich um ca. 30 Bilder pro Sekunde. Die daraus resultierende Datenmenge ist sehr groß und muß daher komprimiert werden. Um die Kompressionsraten der Standbildkompression zu erhöhen, wird die Intraframekompression durch die Interframekompression ergänzt. Durch Untersuchung von Bildfolgen werden bei dieser Technik zeitlich redundante Informationen reduziert. Da gerade bei Videoströmen, die Bildfolgen von 30 Bildern pro Sekunde beinhalten, keine großen Änderungen zwischen 2 Bildern auftreten, ist dieses Verfahren sehr effektiv. Es wird auch Bewegtbildkompression genannt. Dabei wird die Bildfolge in Gruppen aufgeteilt. Nachteilig ist, daß innerhalb einer Gruppe direkt nur auf das erste Bild zugegriffen werden kann. Es hat sich als sinnvoll erwiesen, daß der direkte Zugriff auf 3 Bilder pro Sekunde möglich sein sollte, daher ergibt sich eine Gruppengröße von ca. 12 Bildern. Eine Gruppe kann bis zu 3 verschiedene Bildtypen beinhalten, diese werden im folgenden erläutert.

- Intrapicture:
  Dieses Bild, auch I–Bild genannt, wird mit einem Verfahren der Standbildkompression reduziert und erlaubt einen direkten Zugriff auf einzelne Elemente einer Folge von Bildern. Darüber hinaus fungiert es innerhalb einer Gruppe als erstes und wichtigstes Referenzbild für die sogenannten Predicted Pictures als auch für Bidirectional Pictures.

- Predicted Pictures:

  Diese Bilder, kurz P–Bilder, werden in bezug auf vorhergehende Bilder komprimiert, welche dann auch als Referenzbilder bezeichnet werden. Als Referenzbilder können I–Bilder aber auch P–Bilder genutzt werden. Die Grundidee besteht darin, daß sich die Änderungen zwischen zwei aufeinanderfolgenden Bildern durch sogenannte Verschiebungsvektoren beschreiben lassen. In einem P–Bild findet man also nur noch Teile des Originalbildes wieder, die fehlenden Teile müssen mit Hilfe der Vektoren aus dem Referenzbild gewonnen werden.

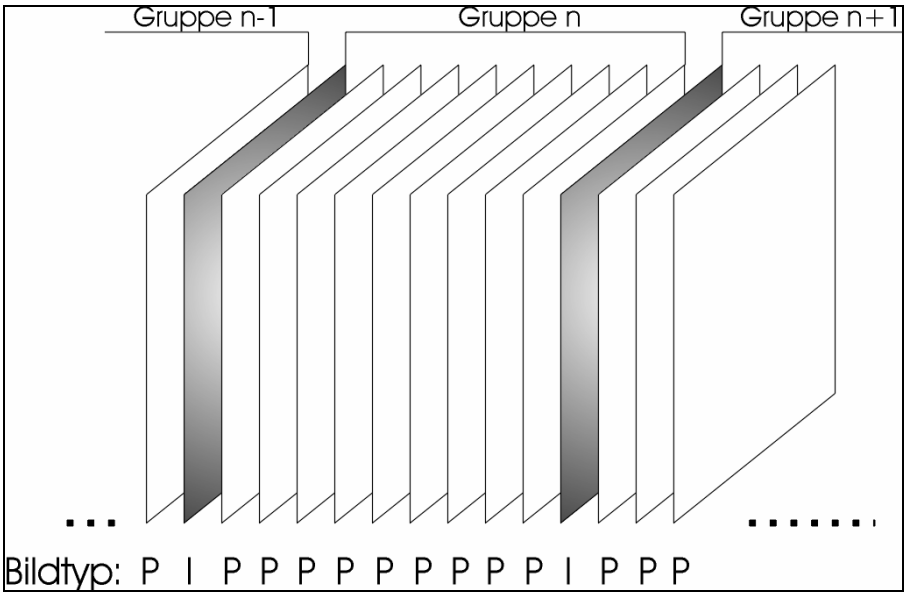


Abbildung 2-3 Bildstrom bestehend aus I- und P–Bildern

- Bidirectional Pictures:

  Bei den sogenannten B–Bildern handelt es sich praktisch um eine Erweiterung der P–Bilder. Als Referenzbilder dienen zusätzlich später folgende I–Bilder oder P–Bilder, aus welchen man in der Regel Informationen über den freigewordenen Bildinhalt gewinnt (siehe Abbildung 2-4).

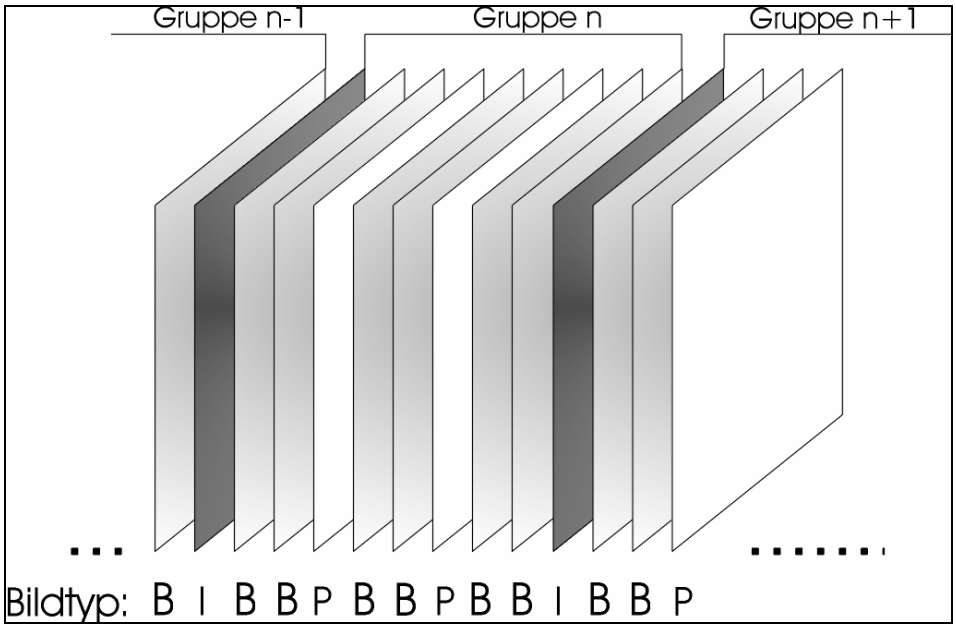


Abbildung 2-4 Bildstrom bestehen aus I-, P- und B–Bildern

Als Besonderheit einer solchen Gruppe von Bildern ist noch zu bemerken, daß die Reihenfolge von Dekompression und Darstellung der Bilder nicht identisch ist (siehe Abbildung 2-5). Dieser Effekt entsteht durch die B–Bilder, für welche man die dekomprimierten Informationen eines später folgenden Bildes benötigt. Der erreichbare Kompressionsgrad ist bei den B–Bildern jedoch am höchsten, gefolgt von den P–Bildern und dann den I–Bildern. Mit dem nötigen Rechenaufwand verhält es sich ähnlich, die Referenzkodierung benötigt komplizierte und aufwendige Such- und Vergleichsoperationen. Trotzdem gibt es mittlerweile spezialisierte Systeme, die das in Echtzeit bewältigen können.

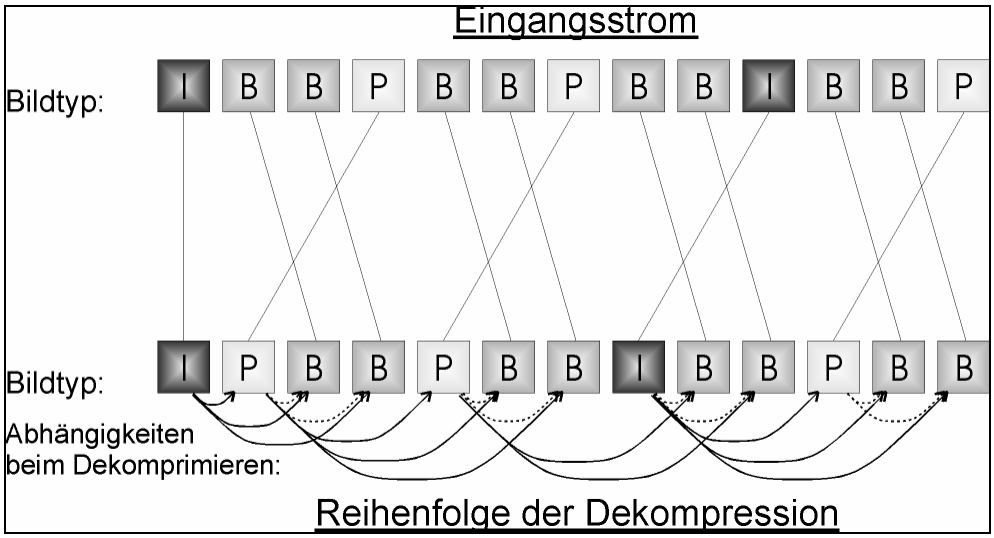


Abbildung 2-5 Unterschiede in der Reihenfolge zwischen Eingang und Dekompression der Bilder

3 Multimedia–Dateiformate

3.1 AVI–Audio Video Interleaf

3.1.1 Hintergrundinformationen

Das zur Zeit am PC wohl noch verbreitetste Dateiformat AVI ist ein von Microsoft geschaffener proprietärer Standard. Eigentlich verbirgt sich dahinter das Containerformat RIFF (Resource Interchange File Format), das damals noch gemeinsam von IBM und Microsoft definiert wurde. Ursprünglich läßt sich das RIFF–Format sogar auf das IFF–Format zurückführen, welches Electronic Arts 1984 als Grafikformat für DeluxePaint auf dem Amiga entwickelte (siehe auch [4]). AVI ist demnach eine spezialisierte Version des RIFF–Formates.

3.1.2 Technische Informationen

Das Format unterstützt komprimierte als auch unkomprimierte Videostreams. AVI ist eine reine Softwarelösung, und benötigt daher auch keinerlei besondere Hardware. Trotzdem ist es möglich, zusätzliche Hardwareunterstützung zu nutzen, um die (De)–Kompression zu beschleunigen. Das anfängliche Konzept erlaubte Videos in Fenstern der Größe 160 x 120 Pixel, mit 15 Bildern pro Sekunde und 8–bit Sound (siehe [5]). Als großer Vorteil von AVI ist die Skalierbarkeit zu erwähnen. Der Benutzer kann beispielsweise völlig frei wählen, welche Fenstergröße und Framerate er haben möchte, sowie welcher Dekompressionsalgorithmus verwendet werden soll, um somit die Wiedergabe an die verfügbare Hardware anzupassen.
Eine vollständige Analyse des Formates würde den Umfang der Arbeit bei weitem sprengen. Allein die Anzahl der verfügbaren Codecs für jeweils Video und Audio ist in der letzten Zeit enorm gestiegen, um sich der immer rasanteren Hardware–Entwicklung anzupassen. Auch die ewige Abwärtskompatibilität ist natürlich ein Grund für die aktuelle Situation.

3.1.3 Formatinformationen

Das Grundprinzip dieses Formates ist die Anordnung der Daten in sogenannten Chunks. Jeder Chunk ist eindeutig durch die in Tabelle 3-1 angegebene Struktur bestimmt.

Offset	Bytes	Erklärung
0x00	4	Signatur, Identifikator
0x04	4	n: Größe des folgenden Datenabschnittes
0x08	n	Datenabschnitt

Tabelle 3-1

Der oberste Chunk ist eine Art Dateikopf zur schnellen Identifikation des vorliegenden Dateityps. Die Signatur dafür ist 'RIFF', der Datenabschnitt beinhaltet als Identifikator die Zeichenkette 'AVI ' und die angegebene Größe bezieht sich auf die Dateilänge. Die nachfolgenden Chunks halten sich an die in Tabelle 3-1 gegebene Syntax. Bitströme vom Typ 'AVI ' haben im wesentlichen den in Tabelle 3-2 dargestellten Aufbau.

Chunktyp; Länge; Signatur	Erläuterung
'RIFF'; Dateilänge; 'AVI '	Dateikopf zur Bestimmung des Typs
'LIST'; Listenlänge; 'hdrl'	Kennzeichnung für eine Headerliste
'avih'; Chunkgröße; Daten	AVI–Header mit spezifischen Daten zum Format
'LIST'; Listenlänge; 'strl'	Kennzeichnung für eine Streamliste
'strh'; Chunkgröße; Daten	Stream–Header für jeden Video- und Audiostream
'strf'; Chunkgröße; Daten	Stream–Format, Informationen über das Format
'strd'; Chunkgröße; Daten	Stream–Daten, optional, weiterleiten an Dekomprimierer
'JUNK'; Chunkgröße; Daten	Optional, Auffüllen der Daten bis zur 2048–Grenze (Optimierung für bestimmte Quellmedien wie CD–ROM)
'LIST'; Listenlänge; 'movi'	Kennzeichnung für die Liste der Video- und Audiodaten
'LIST'; Listenlänge; 'rec '	Kennzeichnung für eine Liste von Records
'##wb'; Chunkgröße; Daten	Chunk mit Audiodaten
'##dc'; Chunkgröße; Daten	Chunk mit komprimierten Videodaten
'##db'; Chunkgröße; Daten	Chunk mit unkomprimierten Videodaten
'idx1'; Chunkgröße; Daten	Liste mit Positionsdaten über die einzelnen Frames des Datenstromes, um schnellen Zugriff zu gewähren

Tabelle 3-2

'##' ist dabei ein Platzhalter für eine Zahl, die den Bezug zu einem der Streamheader herstellt, in welchem formatspezifische Informationen gespeichert sind. '00dc' würde sich dabei auf den ersten Streamheader beziehen, bei dem es sich um komprimierte Videodaten handelt. Detailliertere Informationen, was die Inhalte der einzelnen Header betrifft sind in [11] und [14] zu finden.
Nun noch einige zusätzliche Anmerkungen zu diesem Format. Eine konkrete Richtlinie, was die Anordnung der eigentlichen Audio- und Videodaten betrifft, konnte leider nicht entdeckt werden. Die 'movi'–Liste kann verschiedene Formate annehmen, immer in Abhängigkeit von dem Vorhandensein der 'rec '–Liste. Diese ist eigentlich nur für Datenströme gedacht, die sowohl Audio als auch Video enthalten. Trotzdem gibt es bereits Formate, die beides enthalten, aber trotzdem keine 'rec '–Liste verwenden. AVI enthält keine Zeitstempel oder Framenummern zum zeitlichen Synchronisieren der beiden Datenströme. Im Normalfall sollten jeweils Audio- und Videodaten in chronologischer Reihenfolge gespeichert sein, aber auch hier gibt es Ausnahmen. Das Fehlen solcher Mechanismen wurde bereits als Schwachpunkt erkannt. Zukünftig wird AVI durch Microsoft's ASF (Advanced oder Active Streaming Format) ersetzt, welches Zeitstempelobjekte beherrscht.

3.1.4 Partitionierung von AVI

Wie bereits erläutert ist dieses Format sehr reich an verschiedenen Varianten. Der wichtigste Punkt ist dabei die Möglichkeit, einen beliebigen Treiber zur Dekompression zu benutzen. Um den Datenstrom auf dieser Ebene zu partitionieren, müßte man sämtliche Codecs für Video und Audio gezielt untersuchen. Die Zahl der zur Zeit vorhandenen Codecs dürfte weit über 30 liegen, eine recht umfangreiche Liste ist in [11] zu finden. Aus diesem Grund erscheint es wesentlich sinnvoller und effektiver, die Partitionierung auf Chunk–Ebene vorzunehmen. Die drei in Kapitel 1.5 vorgestellten Parameter ergeben sich wie folgt:

- Dgesamt:
  Dieser Wert kann direkt aus dem RIFF–Header entnommen werden.

- anz_item:
  Der Parameter wird am Ende der Analyse festgelegt und entspricht im wesentlichen der Anzahl der enthaltenen Chunks.

- InputTB:
  Dieser Wert wird dem Streamheader des Videostreams entnommen, der vom Prinzip her die Framerate widerspiegelt. Die Angabe bezieht sich dabei auf ein Frame und erfolgt in Mikrosekunden, wird aber noch in Millisekunden umgerechnet.

3.2 WAVE

3.2.1 Hintergrundinformationen

Dieses Format wird in Microsoft Windows Systemen genutzt. Es beschreibt Audiodaten, die in digitalisierten Samples vorliegen, Details sind in [18] zu finden.

3.2.2 Technische Informationen

Die Daten können sowohl unkomprimiert im PCM (Pulse Code Modulated) als auch komprimiert vorliegen. Zur Kompression können, ähnlich wie bei AVI, verschiedene Codecs genutzt werden. Zur Kodierung wird in der Regel ein DPCM oder ADPCM–Verfahren genutzt (siehe 2.4.1).

3.2.3 Formatinformationen

Auch WAVE ist eine spezielle Variante des RIFF–Fileformats. Im Gegensatz zu AVI jedoch ist es recht einfach und klar gegliedert. Die Beschreibung der Gliederung des Bitstroms ist in Tabelle 3-3 dargestellt.

Typ	Länge in Byte	Beschreibung
RIFF–Header	12	Header zur Bestimmung des Dateitypes
Format–Chunk	24	Formatinformationen
Daten–Chunk	n	Samples

Tabelle 3-3

Aus dem Format–Chunk können die in Tabelle 3-4 aufgelisteten Informationen gewonnen werden.

Information	Bemerkung
Format–Typ	0 = Mono; 1 = Stereo
Kanalzahl	Anzahl der verwendeten Kanäle
Samplerate in Hz	z. B. 22050 Hz
Byte pro Sekunde	Ergibt sich aus Samplerate und Bytes pro Sample
Byte pro Sample	1 = 8 Bit Mono; 2 = 8 Bit Stereo / 16 Bit Mono; 4 = 16 Bit Stereo
Bit pro Sample	8, 12 oder 16

Tabelle 3-4

Die Audiodaten sind im Daten–Chunk sequentiell abgelegt, bei Stereo–Signalen werden die einzelnen Kanäle wechselseitig angeordnet. Die exakten Strukturdefinitionen sind in [18] zu finden.

3.2.4 Partitionierung von WAVE

Wie bereits erwähnt sind bei WAVE–Strömen eine große Anzahl Codecs nutzbar und verfügbar. Daher wird auch eine einfache sequentielle Partitionierung durchgeführt, ohne weiter auf das Format innerhalb des Daten–Chunks einzugehen. Zuerst wird der Format–Chunk als ein Paket übertragen, dann werden die Samples in Pakete zerlegt. Dabei werden die Informationen des Format–Chunks berücksichtigt, um die Paketgröße zu bestimmen. Die Pakete sind dann, bis auf das letzte Paket, immer gleich groß. Die zusätzlichen Parameter werden folgendermaßen bestimmt:

- Dgesamt:
  Dieser Wert kann direkt aus dem RIFF–Header entnommen werden.

- anz_item:
  Der Parameter wird am Ende der Analyse festgelegt und entspricht im wesentlichen der Anzahl der enthaltenen Chunks.

- InputTB:
  Dieser Wert wird in Mikrosekunden angegeben und willkürlich gewählt. Vom ihm ist die resultierende Paketgröße abhängig, die dem Format–Chunk entnommen wird.

3.3 MPEG–Motion Picture Expert Group

3.3.1 Hintergrundinformationen

MPEG ist eine Arbeitsgruppe der International Standards Organization (ISO), die verantwortlich für die Entwicklung von Standards für Video- und Audiokompression ist. Die erste erarbeitete Spezifikation war ISO 11172, allgemein als MPEG–1 bekannt, welche sich auf Datenströme bezieht, die komprimierte Video- und Audioinformationen beinhalten. MPEG–1 beinhaltet 3 Schlüsselgebiete. Teil 1 (Systems) befaßt sich mit der Synchronisation und dem Multiplexen von mehreren komprimierten Audio- und Videodatenströmen. Teil 2 (Video) bezieht sich auf die Videokodierung und Teil 3 (Audio) auf die Komprimierung digitaler Audiodaten (siehe auch [4]). Als Haupteinsatzgebiet war anfangs die Speicherung von Video- und Audiodaten auf Compact Discs (CD) und Digital Audio Tapes (DAT) gedacht. Mittlerweile wurden jedoch mehrere Standards kreiert. Es handelt sich hierbei um MPEG–1, MPEG–2, MPEG–4 sowie MPEG–7, welcher sich allerdings erst in der Planungsphase befindet. Die neueren Standards sind natürlich auch für andere Anwendungsgebiete gedacht, so soll MPEG–2 in der digitalen Fernsehtechnik als auch auf DVD–Basis Verwendung finden.
Bis 1999 wird MPEG–1 der dominierende Standard für Videokodierung bleiben, danach wird MPEG–1 in vielen Anwendungen nach und nach durch MPEG–2 verdrängt werden.

3.3.2 Technische Informationen

MPEG basiert auf dem in CCIR–601 (Comité Consultatif International des Radiocommunications) spezifizierten Standard für digitales Fernsehen, der in den USA Anwendung findet. Dieser Standard definiert eine Auflösung von 720 x 576 Pixeln. Das ergibt eine Datenrate von 166 Mbit/s für ein vollständiges Fernsehbild. Da der komprimierte Datenstrom die maximale Datenrate von 1,86 Mbit/s von MPEG-1 bei weiten überschreitet, wird das oben definierte Bild halbiert, so daß eine Auflösung von 352 x 288 Pixeln entsteht. Das so entstandene Format SIF hat „nur“ noch eine unkomprimierte Datenrate von 30 Mbit/s. Weiterhin unterscheiden sich die Datenraten bereits durch verschiedene länderbezogenene Standards. So wird in den USA eine Rate von 30 frames/s benutzt, während in Europa 25 frames/s Standard sind. Diese Angaben verdeutlichen bereits, daß MPEG eine große Vielzahl von

Variationen im Standard unterstützen muß, um eine möglichst breite und internationale Anwendung zu ermöglichen.
MPEG bietet ausschließlich komprimierte Videostreams. Ein wesentliches Merkmal ist die zeitlich deutliche Diskrepanz zwischen Kompression und Dekompression. Während man mit heute üblichen Systemen MPEG–Daten in Echtzeit auf Softwarebasis dekomprimieren und darstellen kann, bedarf es für die Echtzeitkomprimierung spezieller Hardware.

3.3.2.1 MPEG–1

MPEG–1 definiert kein spezielles Design für einen Videokodierer, sondern die Syntax für einen kodierten Bitstrom sowie ein Modell für einen Dekodierer. Einzelne Implementierungsdetails bleiben dem Entwickler überlassen. Einzige Bedingung ist, daß der Bitstrom der Syntaxdefinition entsprechen muß und mit dem Modell–Dekodierer entschlüsselt werden kann.
Einzelne Frames werden als einzelne Bilder in fortschreitender Ordnung kodiert. Die Auflösung der Videoquelle spielt zwar keine Rolle, aber bei 352 x 240 Pixeln und 30 Frames/s ergibt sich ein Datenstrom von ungefähr 1,2 Mbit/s. Dieser Strom würde ungefähr der Qualität von VHS–Systemen entsprechen (siehe [2]).

3.3.2.2 MPEG–2

Da sich herauskristallisierte, daß sich MPEG–1 nicht für Anwendungen im Fernsehbereich eignete, wurde MPEG–2 entwickelt. Der bei „Fernsehqualität“ entstehende kodierte Datenstrom ist wesentlich größer als die Spezifikation von MPEG–1 erlaubt. MPEG–2 ist wie MPEG–1 in drei wesentliche Teile gegliedert, welche System, Video und Audio umfassen. Der Standard ist gleichermaßen durch die ITU-T als auch durch ISO definiert, hier hat also eine enge Kooperation bei der Entwicklung von MPEG–2 stattgefunden. Im wesentlichen handelt es sich um eine Erweiterung der Funktionalität von MPEG–1, um effektivere Video- und Audiokomprimierung bei einem großen Spektrum von unterstützten Auflösungen als auch Bandbreiten zu gewährleisten. Die Erweiterungen umfassen beispielsweise:

- Unterstützung von interlaced als auch noninterlaced Videos
- bessere Farbauflösung für bessere Videoqualität
- Skalierbarkeit bei der Kodierung/Dekodierung (räumlich, Daten, SNR, zeitlich).

Eine exakte Liste der Erweiterungen ist in [1] zu finden.

3.3.3 Formatinformationen

3.3.3.1 MPEG–1

Es gibt drei Typen von Bildern, dabei handelt es sich um I–Bilder (Intrapicture), P–Bilder (Forward Predicted Pictures) und B–Bilder (Bidirectionally Predicted Pictures). Diese drei Bildtypen werden im Datenstrom zu einer Gruppe zusammengefaßt. Eine Gruppe umfaßt dabei immer ein I–Bild, gefolgt von einer variablen Anzahl von P– und B–Bildern (siehe Abbildung 2-4). Die Größe der Gruppe ist nicht definiert und kann an die jeweilige Anwendung angepaßt werden. Allgemein läßt sich sagen, daß die Kompressionsrate mit zunehmender Gruppengröße steigt. Andererseits ist es oft nützlich und gewünscht, ein Video schnell vorzuspulen, dabei sind dann die I–Bilder wichtig, da nur diese ohne zusätzliche Informationen eine Dekodierung ermöglichen und somit eine Art direkten Zugriffspunkt bilden.

Leider sind Quellen, die sich genauer mit dem Format beschäftigen, nur schwer auffindbar. Es existieren zwar Unmengen von Arbeiten, die sich mit den technischen Details auseinandersetzen, aber es waren keine Informationen zu finden, die konkrete Angaben über sogenannte Startcodes machen. Die hier gemachten Angaben wurden durch die Analyse eines frei verfügbaren MPEG–Programmes [10] gewonnen. Startcodes sind Bitfolgen, die wesentliche Teile, des MPEG–Streams kennzeichnen. Der Bitstrom ist hierarchisch in 6 Schichten gegliedert, was schematisch in Abbildung 3-1 zu sehen ist. Der Header jeder Schicht kann anhand eines klar definierten Startcodes identifiziert werden.

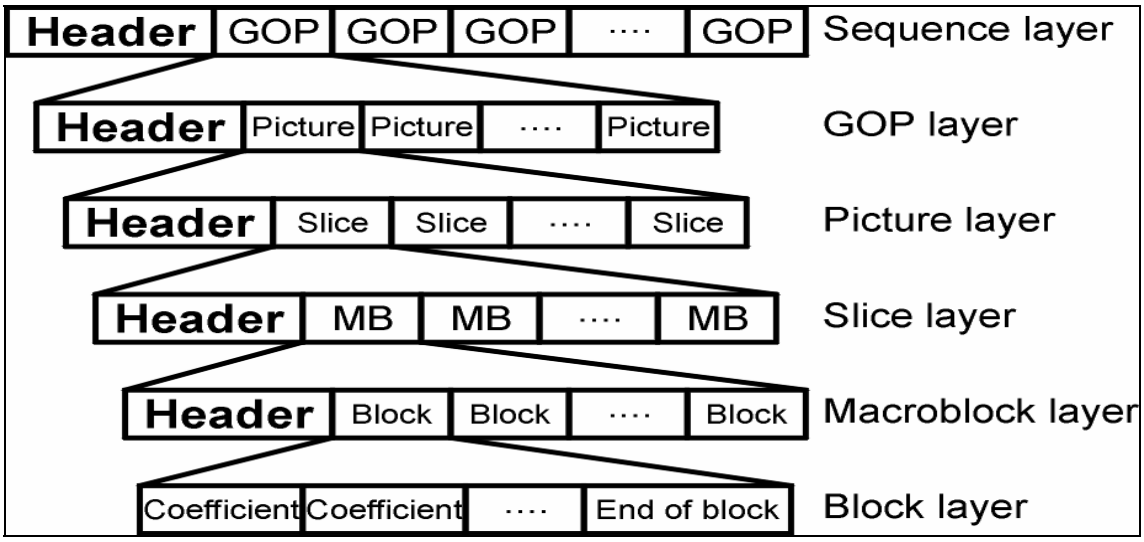


Abbildung 3-1 Struktur eines MPEG–Streams

3.3.3.2 MPEG–2

Der Datenstrom entspricht im wesentlichen dem durch MPEG–1 definierten Standard, damit Abwärtskompatibilität gewährleistet wird.

Die Unterscheidung zwischen MPEG–1 und MPEG–2 wird durch einen zusätzlichen Headercode ermöglicht, der direkt hinter dem Sequenz–Headercode folgen muß. Alle notwendigen MPEG–2 Informationen werden in Extension–Codes abgelegt, die dann zusätzlich ausgewertet werden müssen. Der kodierte Bitstrom hat dabei das in Abbildung 3-2 dargestellte Format. Wie man aus der Abbildung entnehmen kann, besteht ein MPEG–Stream unter Umständen aus mehreren Sequenzen. Für die Analyse–Bibliothek wird die Annahme getroffen, daß immer nur eine Sequenz vorliegt. Diese Annahme ist notwendig und die Begründung in 3.3.4.2 zu finden.

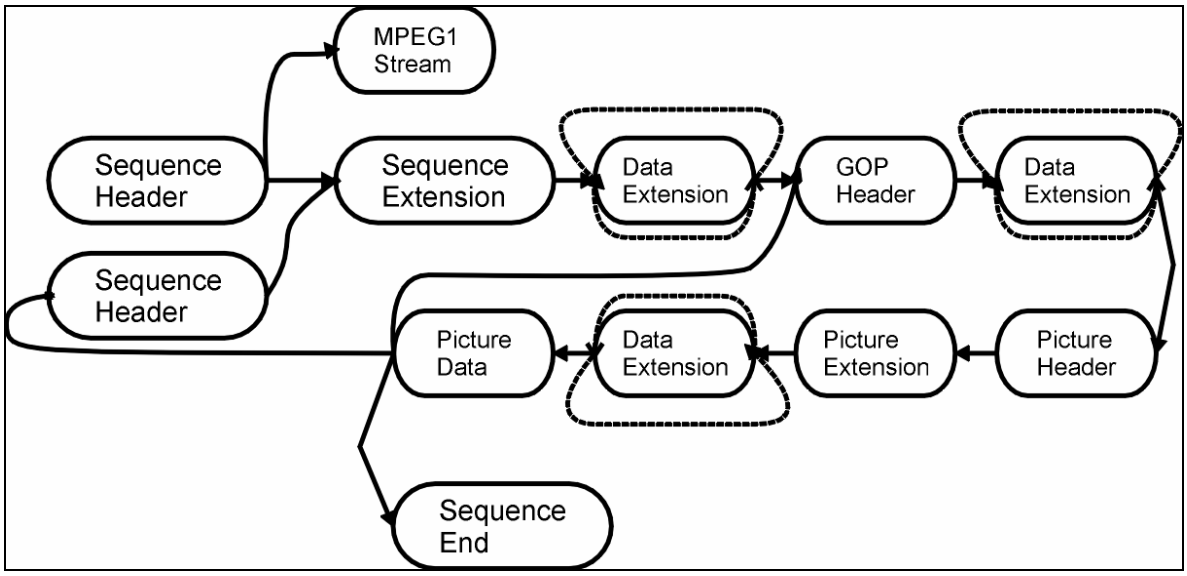


Abbildung 3-2 vereinfachte Struktur des MPEG–2 Formates

Die Erweiterung des MPEG–1 Formates führt, was in Abbildung 3-2 sehr deutlich wird, zu einer hohen Komplexität. Detailliertere Informationen sind in [16] zu finden.

3.3.4 Partitionierung von MPEG

3.3.4.1 MPEG–1

Die günstigste Variante ist die Partitionierung auf Bildebene, analog zu H.26x. Betrachtet man das ganze eine Ebene höher, so wären die Paketgrößen sehr hoch, eine Ebene tiefer würde die Paketzahl sehr hoch ausfallen. Ein weiterer Vorteil ergibt sich unmittelbar aus der Syntax. Wenn man auf Bildebene partitioniert, ist der

Bitstrom sehr einfach zu durchlaufen. Einzelne Header fallen praktisch weg, da die Größe der Header im Vergleich zur Größe der Bilddaten vernachläßigbar klein ist.
Effektiv wird praktisch nur der Header zur Kennzeichnung eines MPEG–1 Streams geprüft, danach werden nur noch Picture–Startcodes gesucht. Somit ist es nicht notwendig, alle möglichen Header–Kombinationen zu untersuchen und abzuarbeiten.
Die allgemeinen Parameter ergeben sich folgendermaßen:

- Dgesamt:
  Dieser Wert ergibt sich am Ende bei Erreichen des Sequence Endcodes.

- anz_item:
  Der Parameter wird am Ende der Analyse festgelegt und entspricht der Anzahl der enthaltenen Frames plus einem Paket für den Sequence Header.

- InputTB:
  Im Sequence Header kodiert ist die Framerate angegeben. Aus der Framerate läßt sich direkt errechnen, aller wieviel Millisekunden ein Bild aus dem Puffer geholt und dekodiert werden muß. Der errechnete Wert entspricht dann InputTB.

3.3.4.2 MPEG–2

Auch MPEG–2 wird aus den im vorangegangen Kapitel erläuterten Gründen auf Bildebene partitioniert. Aufgrund der hohen Komplexität der erweiterten Syntax macht sich diese Strategie erst recht bezahlt. Die zahlreichen Data–Extensions sind, betrachtet man die Größe dieser Strukturen, vernachläßigbar klein und werden zu den Bildpaketen hinzugezählt. Anhand des Extension Startcodes ist die Unterscheidung von MPEG–1 möglich. Die zusätzlichen Variablen ergeben sich analog zu MPEG–1.
Die in Kapitel 3.3.3.2 gemachte Annahme beruht auf die Angaben des Verbraucherprozesses, den Dekodierer. Die in Arbeit [6] durchgeführten Berechnungen sind von der Charakteristik des Datenstromes abhängig. Der Parameter „InputTB“ spiegelt als solcher die Verbraucherrate wieder, und ist in diesem Fall vom Inhalt des Sequence Headers abhängig. Bei mehreren Sequenzen kann sich dieser Wert allerdings ändern, sodaß die Berechnungen für die folgenden Sequenzen nicht mehr den tatsächlichen Bedingungen genügen. Es muß also für jede Sequenz eine separate Analyse und Auswertung durchgeführt werden.

3.4 H.261: Motion Video Coding für Videokonferenzen

3.4.1 Hintergrundinformationen

H.261 ist ein Teil der Spezifikationen, die unter H.320 zusammengefaßt sind. H.320 beinhaltet die Spezifikationen G.711, G.722, G.728 (audio CODEC), H.230 (audiovisual control and synchronization), H.221 (frame structure for ISDN channel) sowie die oben genannte Spezifikation H.261 (video codec). Aufgrund erhöhter Nachfrage und Verbreitung von ISDN–Verbindungen wurde H.261 mit dem Ziel entwickelt, Videokonferenzen sowie Videotelefonie über ISDN-Verbindungen zu ermöglichen. Die damit zur Verfügung stehende Datenrate ist somit ein vielfaches von 64 Kbit/s (p x 64 Kbit/s, p=1,2,...,30).
Weiterhin machte sich ein internationaler Standard notwendig, da weltweit sehr viele verschiedene Telefonstandards existieren. H.261 beschäftigt sich ausschließlich mit der (De–) Kompression der visuellen Komponente und ist für uni– als auch bidirektionale Kommunikation geeignet. In bezug auf das Einsatzgebiet handelt es sich um symmetrische Verfahren, die in Realzeit durchgeführt werden müssen.

3.4.2 Technische Informationen

H.261 unterstützt genau 2 Auflösungen. Zum einen 352 x 288 Pixel, genannt CIF (common intermediate format), und zum anderen 176 x 144 Pixel, genannt QCIF (quarter CIF). Wie der Name schon verrät, ist die entstehende Datenrate bei QCIF nur ein viertel der entstehenden Datenrate von CIF, QCIF ist also für geringere zur Verfügung stehende Bandbreiten geeignet. QCIF muß dabei von jedem Decoder unterstützt werden, während CIF optional ist. Beide Formate erlauben eine maximale Bildwiederholrate von 29,97 frames/s. Es ist jedoch möglich, 0 bis maximal 3 Bilder bei der Übertragung zu überspringen, um die Datenrate zu verringern. Die Empfehlung für die entstehende Bitrate liegt dabei zwischen 40 Kbit/s und 2 Mbit/s und kann an die vorherrschenden Bedingungen angepaßt werden.
Zur Kompression wird ein hybrides Verfahren, also eine Kombination von verlustfreien und verlustbehafteten Techniken angewendet. Details hierzu werden im folgenden Kapitel angegeben.

3.4.3 Formatinformationen

Bei H.261 finden Intraframe- als auch Interframekompression Anwendung. Erstere findet in den folgenden Schritten statt. Zuerst werden die Bildkomponenten in 8 x 8 Blöcke zerlegt. Diese werden dann jeweils separat einer DCT unterzogen. Die entstandenen Matrizen werden quantisiert und auf ganze Zahlen gerundet. Abschließend wird noch die Huffmann–Kodierung angewendet. Die Interframekompression arbeitet ohne B–Bilder, das heißt es gibt nur Referenzen auf das vorhergehende Bild. Einfachere Realisierungen arbeiten nur mit Nullvektoren um die rechenintensive Bestimmung der Bewegungsvektoren zu umgehen.
Bei der Übertragung wird das höherwertige Bit zuerst übertragen, die Bitanordnung erfolgt dabei nach Big Endian. Der entstehende Bitstrom ist in 4 hierarchische Schichten aufgeteilt, wobei jede Schicht durch einen Startcode, präsentiert durch eine exakt definierte Bitfolge, identifiziert werden kann. Die Anordnung der Schichten ist in Abbildung 3-3 dargestellt, detailliertere Informationen sind in [15] zu finden.

3.4.4 Partitionierung von H.261

Video–Coder und Video–Decoder verwalten beide einen eigenen Buffer, um Übertragungsschwankungen auszugleichen. Die Berechnung entsprechender Parameter in der DROPS–Umgebung spielt also eine untergeordnete Rolle. Weiterhin ist es für Echtzeitanwendungen nahezu unmöglich, gezielte Berechnungen durchzuführen, da ja das Verhalten und die Entwicklung des Bitstromes nicht vorhersagbar sind. In diesem Fall müßten dann vorher konkrete Angaben und Zusagen gemacht werden, auf deren Basis eine Berechnung erfolgt. Der Buffer des Decoders wird alle 33 ms überprüft, bei der höchsten Framerate also bei jedem einzelnen Bild. Der Decoder kann auf die Framerate Einfluß nehmen, indem er beim Coder eine Erhöhung des Quantisierungsfaktors veranlaßt, um die Datenmenge zu reduzieren (siehe 2.4.3). Unter der Annahme, daß drei I–Bilder pro Sekunde für einen Direktzugriff übertragen werden, könnte eine sinnvolle Partitionierung gruppenweise durchgeführt werden. Bei größeren zugrundeliegenden Bandbreiten wäre die Pufferdimensionierung dabei vielleicht schon zu groß. Deshalb ist es sinnvoller, den Bitstrom auf Bildebene zu partitionieren.
Innerhalb der Arbeit wird davon ausgegangen, daß ein H.261–Videostrom im gesamten vorliegt und analysiert werden kann. Weiterhin wird die Annahme getroffen, daß die Framerate beim Maximum von 29,97 Hz liegt. Die Partitionierung wird, wie bereits erläutert, auf Bildebene durchgeführt, der Inhalt der

darunterliegenden Schichten spielt daher keine Rolle. Der gesamte Bitstrom hat die in Abbildung 3-3 dargestellte Syntax.

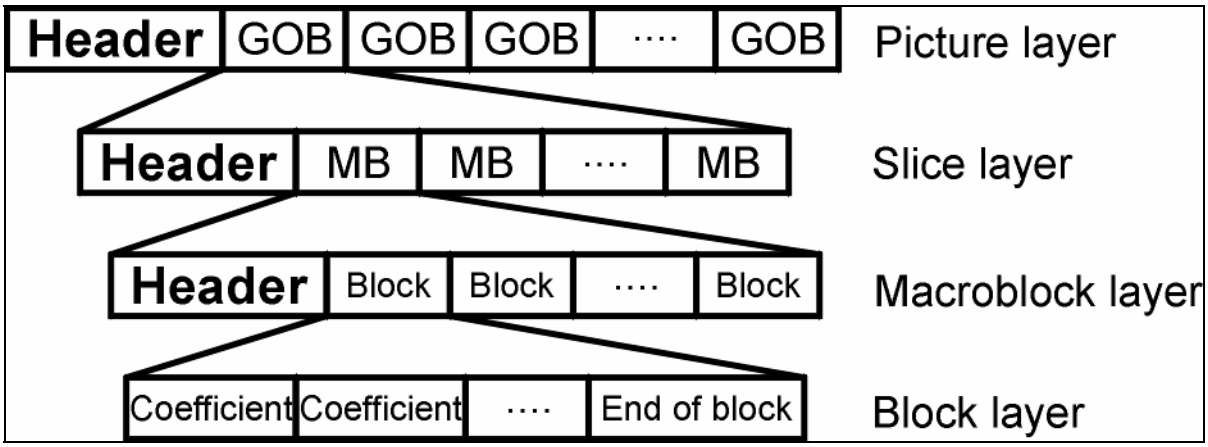


Abbildung 3-3 Struktur eines H.261–Streams

Für die Partitionierung von Bedeutung ist eigentlich nur die ‚Picture layer'. Jedes einzelne Paket präsentiert dabei ein Frame. Die zusätzlichen benötigten Parameter (siehe 1.5) werden folgendermaßen belegt:

- Dgesamt:
  Dieser Wert repräsentiert die Gesamtlänge des vorliegenden Bitstromes.

- anz_item:
  Der Parameter wird am Ende der Analyse festgelegt und ergibt sich aus der Anzahl der Pakete, in die der Datenstrom zerlegt wird.

- InputTB:
  Dieser Wert ergibt sich trivialerweise aus der Arbeitsweise des Decoders, der seinen Puffer aller 33 Millisekunden überprüft.

3.5 H.263–low bit rate video coding

3.5.1 Hintergrundinformationen

H.263 basiert auf H.261, mit dem Ziel bessere Qualität bei noch geringeren Datenraten zu ermöglichen. Da heutzutage bei einem Großteil der Anwender Modems Verwendung finden, die eine effektive Datenrate von max. 30 Kbit/s erlauben, ist ein Standard für diesen Bereich mehr als notwendig.

3.5.2 Technische Informationen

H.263 wurde überhaupt erst ins Leben gerufen, um fundamental neue Kompressionstechniken in bisherige Standards (H.261, siehe hierzu [7] und [8]) einzuflechten. Im wesentlichen bleibt es ein hybrides Verfahren mit folgenden Verbesserungen:

- Verbesserte Motion–Compensation Verfahren
- Arithmetische Kodierung anstatt Huffmann–Kodierung
- PB–pictures, 2 Bilder werden zu einer Einheit zusammengefaßt

Durch den Einsatz dieser Erweiterungen kann definitiv eine deutliche Verbesserung der Qualität, sowie eine Reduzierung der Datenmange erreicht werden. Einsatzgebiet ist die visuelle Kommunikation, wobei bidirektionale als auch unidirektionale Verbindungen gebräuchlich sind. Als Erweiterung zu H.261 wird zu CIF und QCIF nun auch sub–QCIF, 4CIF und 16CIF unterstützt (siehe Tabelle 3-5).

Format	X–Ausdehnung	Y–Ausdehnung	BPPmaxKb
Sub–QCIF	128	96	64
QCIF	176	144	64
CIF	352	288	256
4CIF	704	576	512
16CIF	1408	1152	1024

Tabelle 3-5

Der Decoder besitzt einen eigenen Puffer, dessen Mindestgröße sich aus folgender Formel berechnet:

(B + BPPmaxKb * 1024 bit); mit B = 4 * Rmax / 29,97.

BPPmaxKb ist in Tabelle 3-5 zu finden, Rmax ist abhängig vom Übertragungsmedium und könnte die maximal unterstützte Bitrate sein. Der Puffer wird alle 33 Millisekunden überprüft. Wenn mindestens ein komplettes Bild in kodierter Form im Puffer vorliegt wird das älteste im Puffer befindliche vollständig gelöscht.

3.5.3 Formatinformationen

Da es sich um eine Erweiterung von H.261 handelt, gelten auch die dort angegebenen Details. Veränderungen sind, was das Format betrifft, nur im Detail zu erkennen. Auch H.263 ist in einer hierarchischen Struktur mit 4 Schichten (siehe auch Abbildung 3-3) angeordnet. Die einzelnen Schichten sind:

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

Jede Schicht kann man an einem Startcode erkennen. Der Anfang eines Startcodes für ein einzelnes Bild muß dabei immer auf einer Bytegrenze liegen. Die Picture Layer ist praktisch eine Reihe von Einzelbildern. Jedes Bild ist in den darunterliegenden Schichten kodiert gespeichert. Details zur Struktur der Schichten und den einzelnen Startcodes sind in [17] zu finden.

3.5.4 Partitionierung von H.263

Für das in dieser Arbeit zu untersuchende Modell ist H.263 in seiner gebräuchlichen Form ebenso ungeeignet wie H.261. Ein Problem stellt dabei der vom Decoder selbst verwaltete Puffer dar, ein weiteres ergibt sich aus der dynamischen Bitrate.
Die Pufferverwaltung ließe sich eventuell durch eine eigene Decoder–Implementierung an das in Paper [6] vorgestellte Modell anpassen. Wesentlich komplizierter gestaltet sich die variable Bitrate. Diese müßte in einer selbständigen Arbeit untersucht werden, um die Zusammenhänge und Abhängigkeiten zwischen Coder, Decoder, Übertragungsmedium und Bitrate näher zu beleuchten.
Es wird also wieder die Annahme getroffen, daß ein mitgeschnittener H.263–Bitstrom im gesamten vorliegt und untersucht werden kann. Die Differenzen zwischen H.261 und H.263 machen sich im wesentlichen nur in den unteren Schichten bemerkbar. Da die Partitionierung auf Bildebene erfolgt, und somit nur die oberste Schicht untersucht werden muß, gelten für die Partitionierung eines solchen Bitstromes die in Kapitel 3.4.4 gemachten Aussagen.

4 Zusammenfassung

Multimedia als Teilfachgebiet der Informatik ist noch ein relativ junges Gebiet. Dementsprechend dynamisch ist auch die Entwicklung. Es existieren viele verschiedene Ansätze zur Datenkompression. Die meisten haben ihre Stärken auf speziellen Gebieten und sind deshalb sehr anwendungsorientiert. Einige befinden sich erst am Anfang der Entwicklung, und könnten in Zukunft noch großen Einfluß ausüben. Trotzdem haben sich einige Standards herauskristallisiert, die bereits weit verbreitet sind. Man sollte sich jedoch vor Augen führen, daß sich jeder dieser Standards bereits in der Weiterentwicklung befindet. Dieser Vorgang wird natürlich durch das immense Wachstum der Hardwarebranche gefördert. Keiner kann mit Sicherheit voraussagen, wie die Situation in 10 Jahren aussehen wird, und welche Faktoren zunehmend an Bedeutung gewinnen oder verlieren. Man muß sich jedoch die Eigendynamik dieses Fachgebietes stets vor Augen führen und erkennen, daß statische Lösungen auf Dauer ungeeignet sind.

Die untersuchten Formate haben auf jeden Fall offenbart, daß ein statisches aufgesetztes Modell zur Berechnung und Vereinbarung von QoS–Parametern bis auf wenige Ausnahmen eher ungünstig ist. Denkbar wäre da schon eher, daß bestimmte Multimedia–Applikationen auf Basis der in Arbeit [6] vorgestellten Modelle implementiert werden, um die Pufferverwaltung direkt zu integrieren.

Aber auch dann ergeben sich noch Probleme bei Anwendungsgebieten, die sich mit audiovisueller Kommunikation in Echtzeit beschäftigen. Hier ist ein Modell, daß von vorliegenden Datenströmen ausgeht, völlig ungeeignet. Ein denkbarer Lösungsansatz wäre eine statistische Untersuchung der Paketzusammensetzung von Datenströmen in Abhängigkeit des verwendeten Kompressionsverfahrens. Diese Daten könnten dann sozusagen als Basis dienen. Trotzdem sollte das Modell fähig sein, den eingehenden Strom ständig zu untersuchen, und die Parameter zu aktualisieren. Zusätzlich müßte noch die Möglichkeit bestehen, die QoS–Vereinbarungen in Echtzeit anzupassen.

5 Implementierung der Analysebibliothek

5.1 Vorüberlegungen

Aufgabe der Bibliothek ist es, einen Datenstrom zu analysieren und einige ausgewählte Parameter in einen Ergebnisstrom zu schreiben. Die Bibliothek soll in

der Programmiersprache C implementiert werden. Da die Bibliothek auf verschiedenen Plattformen lauffähig sein soll (Windows, Linux), ist es notwendig, nur ANSI–konforme Bibliotheksfunktionen zu verwenden. Zum Testen wird dabei der Sourcecode jeweils auf einer Linux– und Windows–Maschine kompiliert und ausgeführt. Die Ergebnisse sollen dabei bei gleichen Eingangsparametern immer identisch sein.
Im Hinblick auf die weitere Verwendung im DROPS–Projekt sollen die Ein– und Ausgabefunktionen so programmiert werden, daß eine leichte Anpassung an andere Umgebungen möglich ist.

5.2 Annahmen und Lösungsansätze

Um die gestellten Anforderungen zu erfüllen müssen einige grundsätzliche Designentscheidungen getroffen werden.
Zum Zugriff auf einen Datenstrom wird immer ein Zeiger verwendet. Unabhängig von den verwendeten Geräten muß die Möglichkeit bestehen, diesen Zeiger an eine beliebige gültige Position innerhalb des Datenstroms zu setzen.
Wie bereits erwähnt, sollen nur ANSI–C– konforme Funktionen verwendet werden.
Um die Bibliothek leicht erweiterbar zu halten, sollen sinngemäß zusammengehörige Funktionen in separaten Modulen zusammengefaßt werden.
Die gewählten Module sowie eine kurze inhaltliche Beschreibung sind im folgenden aufgeführt:

- Errors.c:
  In diesem Modul werden alle Funktionen, Ausgabetexte sowie Fehlercodes zusammengefaßt. Anhand der Ausgaben im Fehlerfall soll eine möglichst einfache Fehlerkorrektur von Seiten des Benutzers ermöglicht werden.

- Parse.c:
  Hier werden alle Funktionen abgelegt, die zum Analysieren der Datenströme benötigt werden. Beim Erweitern der Bibliothek durch neue Formate muß hier der Hauptteil der Arbeit geleistet werden.

- Ident_type.c:
  Da die Bibliothek das Format des Datenstroms automatisch erkennen soll, wurde ein Modul eingeführt, daß sich speziell dieser Aufgabe widmet. Hier werden in der

Regel nur einige Bytes vom Anfang des Datenstroms eingelesen, da sich dort die Formatinformationen befinden. Für jedes Format sollte eine extra Funktion existieren. Das Programm geht dabei so vor, daß der Datenstrom der Reihe nach auf bekannte Formate geprüft wird, bis eine eindeutige Identifizierung erfolgt ist. Sollte keine Identifizierung möglich sein, bricht das Programm mit einer entsprechenden Fehlermeldung ab.

- Input_output.c:
  Innerhalb des Programmes erfolgen nur byteweise Lese– und Schreiboperationen. Diese Operationen werden dann auf das jeweilige verwendete Medium abgebildet. Die Identifizierung des Mediums wird über Kommandozeilenparameter geregelt. Der Datenstrom wird eindeutig gekennzeichnet durch den Typ des Mediums und einen Zeiger auf den Datenstrom. Innerhalb der Arbeit wurde eine Implementierung am Beispielmedium Festplatte realisiert. Ein– und Ausgaben erfolgen also über Dateien.

- Tools.c:
  Hier werden alle Funktionen zusammengefaßt, die häufig benötigt werden aber zu keinen der oben genannten Module zugeordnet werden konnten.

- Analyser.c:
  Das ist das eigentliche Programm. Hier werden die Kommandozeilenparameter ausgewertet. Gleichzeitig werden die Eingaben auf ihre Gültigkeit hin überprüft.

5.3 Probleme

Einen Problempunkt stellt sicherlich das vordefinierte Ausgabeformat dar, welches im Textformat vorliegt. Alle ausgegeben Werte müssen also vorher einer Umwandlung in eine Zeichenkette unterworfen werden.
Das Problem ist, daß einige Parameter, die bereits am Anfang bekannt sein müssen (anz_item, siehe Kapitel 1.5) erst am Ende der Analyse vorliegen.
Eine mögliche Lösung wäre die Implementierung einer Liste, um dynamisch anfallende Daten aufzunehmen. Diese Lösung ist allerdings verhältnismäßig aufwendig.

Ein zweiter Ansatz wäre die Verwendung eines Arrays. Dieser Ansatz ist sehr leicht zu implementieren, hat aber den Nachteil, daß die Anzahl der Werte nach oben hin begrenzt ist. Man hätte einen Kompromiß zwischen Speicherver(sch)wendung und Maximalwert zu lösen.
Der einfachste Ansatz wäre das sofortige Schreiben jedes analysierten Wertes. Die unbekannten Header–Informationen müßten dann am Ende der Analyse geschrieben werden. Da in der Regel davon ausgegangen wird, daß kein Einfügen, sondern nur das Überschreiben von Daten möglich ist, müssen am Anfang genügend Platzhalter eingefügt werden. Dieser Ansatz wird in der aktuellen Implementierung genutzt.
Prinzipiell sollte dieser Programmteil bei einer echten Anwendung im DROPS–Projekt völlig neu gestaltet werden.

6 Anwendung und Ergebnisse der Analysebibliothek

6.1 Die Umgebung

Die Bibliothek wurde jeweils unter Windows und Linux kompiliert und getestet. Zum Testen wurden dabei jeweils die gleichen Eingangsparameter hergezogen, die Ergebnisse sind, wie erwartet, identisch. Die Tests wurden dabei an jeweils mindestens einem Vertreter jedes Formates durchgeführt.
Während für AVI, WAVE und MPEG–1 auf vorhandene Datenströme zurückgegriffen werden konnte, mußten für H.261 und H.263 selbst Datenströme kreiert werden. Hauptaugenmerk lag dabei auf dem Erkennen der jeweiligen Startcodes innerhalb des Datenstroms.
Das Programm wird von der Kommandozeile aus gestartet. Dabei müssen beim Programmstart einige Parameter übergeben werden. Es handelt sich dabei um:

- -if: Name des Inputfiles, komplett mit Pfad falls nicht im aktuellen Verzeichnis
- -of: Name des Outputfiles, komplett mit Pfad falls nicht im aktuellen Verzeichnis
- -iq: Parameter, der die Eingangsrate beim Empfänger kennzeichnet, ist vom Sender und Übertragunsmedium abhängig.

Sollte einer der Parameter nicht korrekt sein, bricht das Programm mit einer entsprechenden Fehlermeldung ab.
Alle zum Testen herangezogenen Datenströme befinden sich im übrigen auf dem der Arbeit beigelegten Datenträger.
Da AVI durchaus als das komplexeste hier vorgestellte Format zu betrachten ist, wurden zum Testen sogar drei sich vom Format her unterscheidende Datenströme zum Analysieren ausgesucht.

6.2 Ergebnisse der Analyse

Wie bereits erwähnt, konnten nur für drei der sechs untersuchten Formate relevante Tests durchgeführt werden.
WAVE wird im folgenden nicht weiter betrachtet, da sich durch die feststehende Samplerate ein linearer Datenstrom mit gleich großen Paketen ergibt.
Zuerst sollen die Ergebnisse der Analyse zweier MPEG–1 Datenströme vorgestellt werden. Man kann in Abbildung 6-1 eine grafische Darstellung der Paketgrößen sehen.

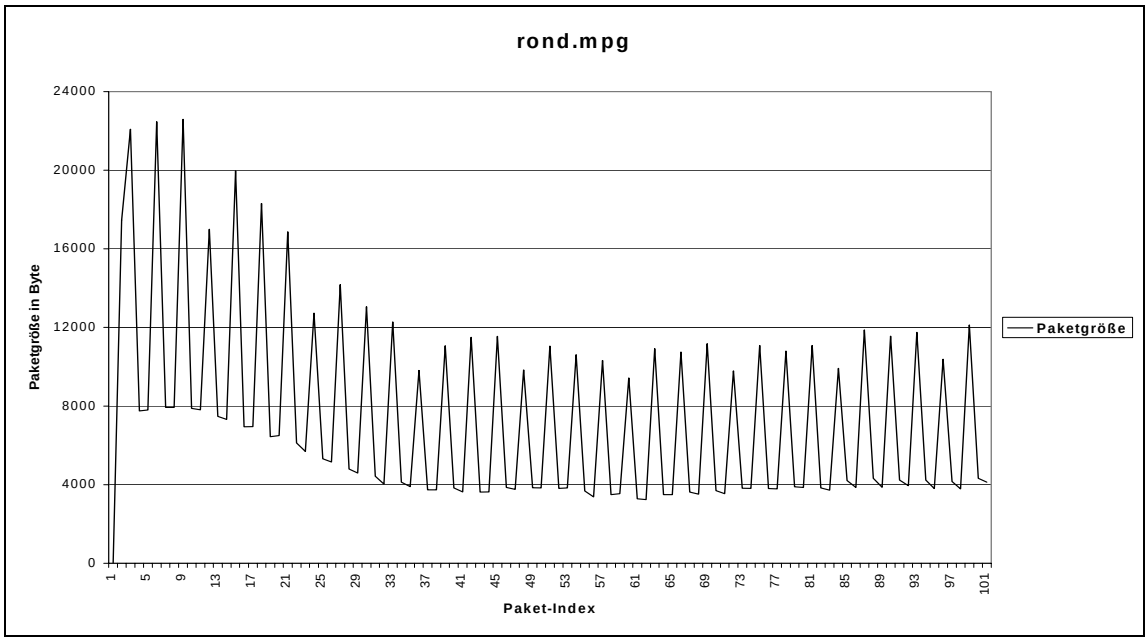


Abbildung 6-1 Beispiel1: MPEG–1 Datenstrom

Dabei ist deutlich zu erkennen, daß zwischen zwei Bildern jeweils zwei weitere angeordnet sind, die jeweils nur halb so groß sind. Am Anfang ist ein sehr kleines Paket auffallend, welches den Streamheader repräsentiert. Weiterhin ist eine sehr

große Streuung der Paketgrößen zu bemerken. Diese Verhalten kann als typisch bezeichnet werden, da die Bildgrößen sehr stark abhängig von der Komplexität der Bilder selbst und dem Änderungsverhalten der Bilder untereinander sind.

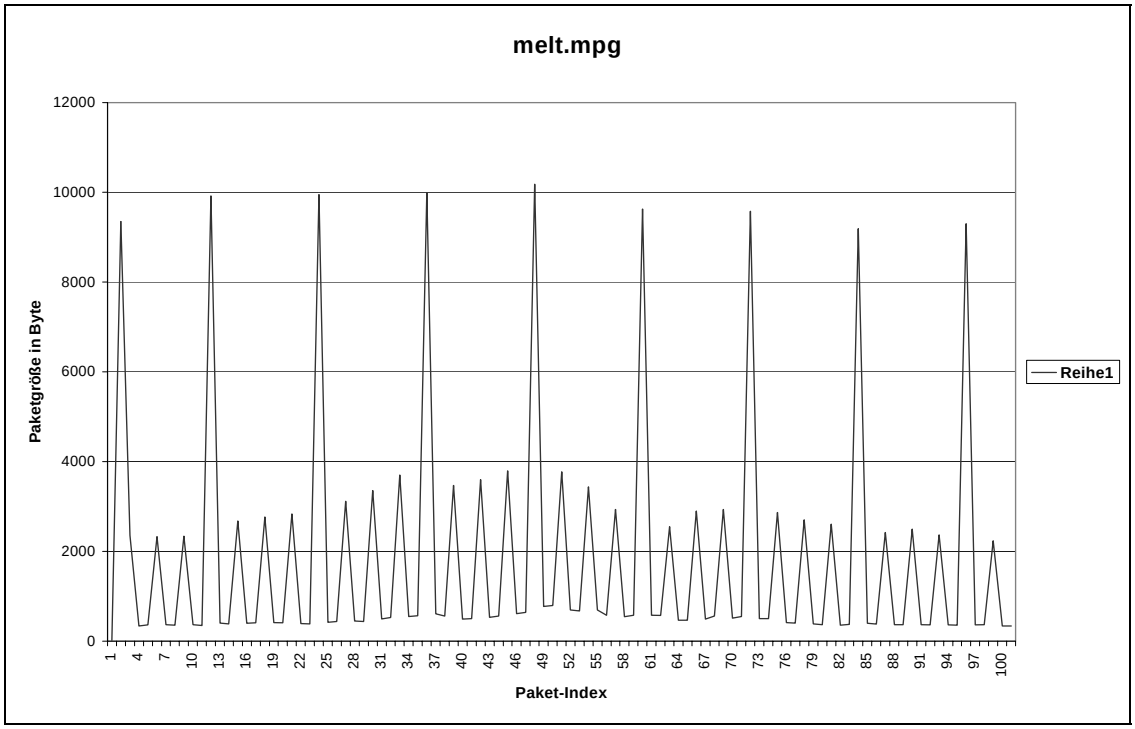


Abbildung 6-2 Beispiel2: MPEG–1 Datenstrom

In Abbildung 6-2 sind in der Grafik alle drei Bildtypen (siehe 3.3.3.1) deutlich zu erkennen. Interessant ist dabei das Größenverhältnis untereinander. Man kann sehr gut die Kompressionsrate ablesen, die hier zwischen I–Bild und P–Bild ca. 5:1 und zwischen I–Bild und B–Bild ca. 10:1 beträgt.

Nun folgt die Betrachtung der Analyse einiger AVI–Datenströme. Sehr auffällig sind die herausragenden Spitzen in Abbildung 6-3. Da es sich hier um einen Datenstrom aus Audio– und Videodaten bestehend handelt, könnte man diese Spitzen als Audiodaten interpretieren. Diese sind ja, wie in 3.1.3 beschrieben, sequentiell in der Reihenfolge der Wiedergabe gespeichert.

Nun wird zusätzlich noch ein AVI–Strom betrachtet, der nur Videoinformationen beinhaltet. Anhand des Verlaufs der Paketgrößen in Abbildung 6-4 könnte man auf ein Verfahren schließen, welches auf Motion–Compensation (siehe 2.4.1) basiert.

Eine genauere Analyse setzt jedoch eine Untersuchung des hier verwendeten Codecs voraus.

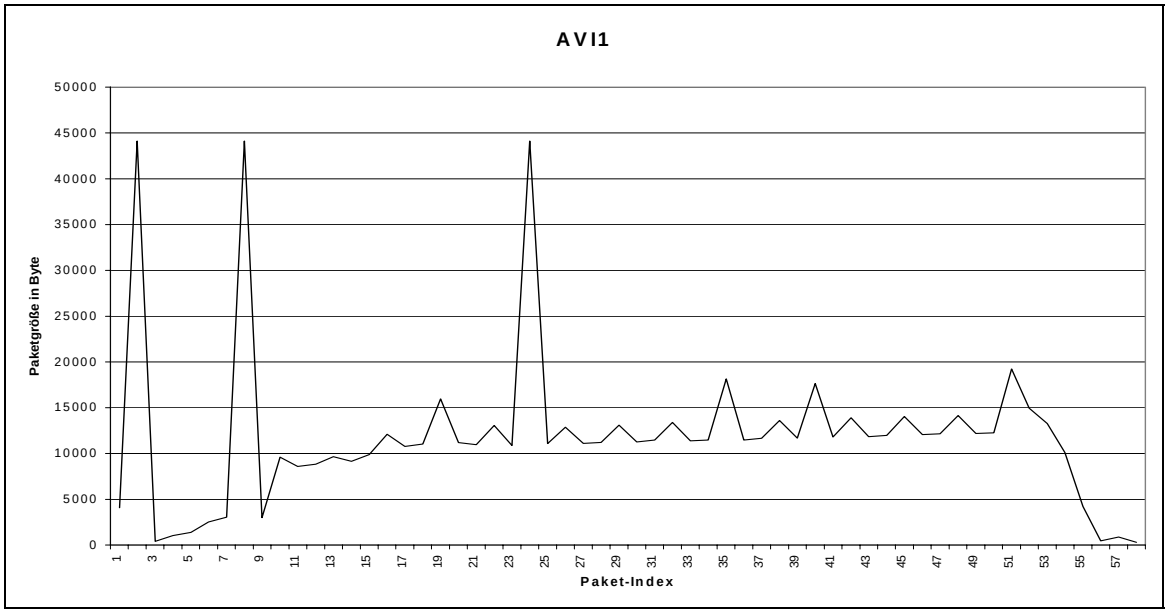


Abbildung 6-3 Beispiel3: AVI–Datenstrom (Audio und Video)

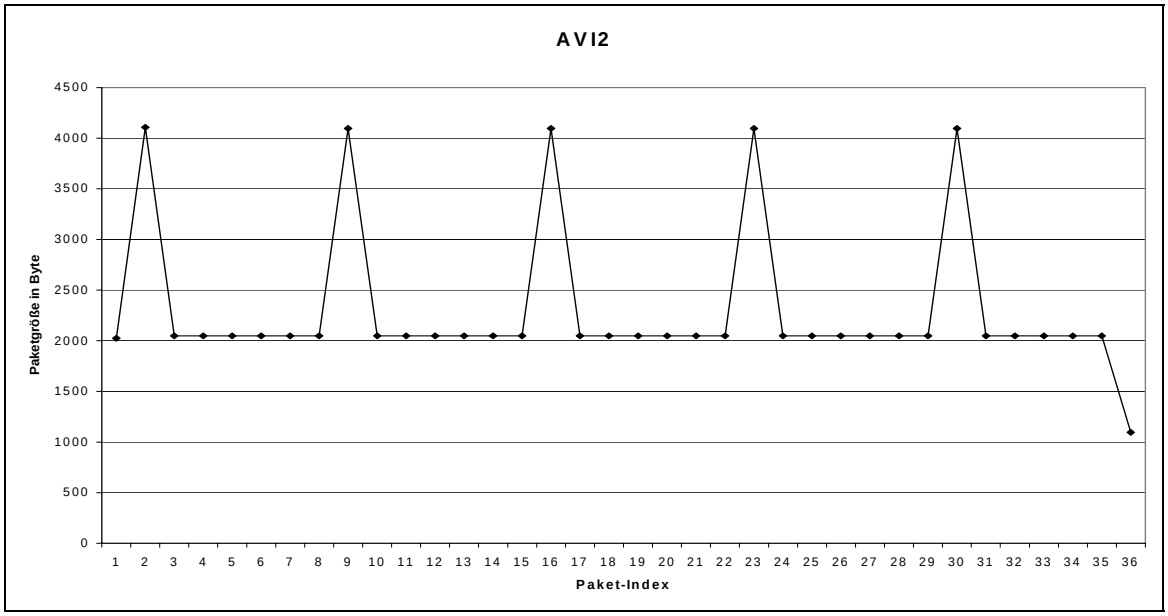


Abbildung 6-4 Beispiel4: AVI–Datenstrom (nur Video)

Obwohl die letzten zwei Analysen auf demselben Format beruhen, sind die analysierten Ergebnisse sehr unterschiedlich ausgefallen. Im Vergleich dazu sind sich die beiden MPEG–1 Datenströme von der Charakteristik her sehr ähnlich, sie unterscheiden sich im wesentlichen nur durch Bildtyp und Kompressionsgrad.

6.3 Experimentelle Ergebnisse

Um die Ergebnisse dieser Arbeit zu verdeutlichen, soll nun noch ein experimenteller Vergleich durchgeführt werden. Dabei wird die Partitionierung eines MPEG–1 Datenstroms einmal, wie in 3.3.4.1 erläutert, auf Basis der einzelnen Bilder durchgeführt. Zum Vergleich wird der gleiche Datenstrom noch einmal auf Basis der GOP partitioniert. Die Ergebnisse der Analysebibliothek werden dann hergenommen und an das in Arbeit [6] entwickelte Programm zur Berechnung übergeben. Die Berechnungsergebnisse sind in Tabelle 6-1 und Tabelle 6-2 dargestellt.

Wert	Pictures	GOP
Periode von A in ms	11.4	11.2
Pmin in Byte	18431	63906
Vorlauf in ms	290	554
Anfangsfüllstand in Byte	13312	25600
Vorperiode in ms	7.4	7.2

Tabelle 6-1 Ergebnisse mit Methode 1, inputTB = 42 ms

Wert	Pictures	GOP
Pmin in Byte	18431	63906
Vorlauf in ms	290	554

Tabelle 6-2 Ergebnisse mit Methode 2, inputTB = 420 ms

Ohne die Werte zu interpretieren ist doch deutlich die Abhängigkeit des Puffers Pmin und der Vorlaufzeit von der Paketgröße und der Verbraucherperiode zu erkennen. Wie bereits in Kapitel 1.3 erläutert wurde, sollten diese Parameter nicht beliebig gewählt, sondern auf die jeweilige Anwendung abgestimmt werden.

In diesem Fall ist die Partitionierung auf Bildebene als günstiger zu betrachten, da wesentlich weniger Ressourcen verwendet werden. Die Bildabhängigkeiten innerhalb eines solchen Datenstroms könnten zwar zu größeren Gruppen führen, jedoch gibt es immer einen Konflikt bei der Zugehörigkeit eines Bildes zu einer Gruppe. Gerade I– und P–Bilder werden in der Regel als Referenzen für vorhergehende und nachfolgende Bilder genutzt, was in Abbildung 2-5 verdeutlicht wurde.

7 Anhang

7.1 Quellenverzeichnis

Das Verzeichnis enthält neben den Angaben über die diesem Beleg zugrunde liegenden Quellen auch weiterführende und aktuelle Werke zu den behandelten Themenfeldern

[1] Riley, Martyn J, Richardson Iain E. G.: *Digital Video Communications*. AZTECH HOUSE, INC., Norwood 1997, S.15-50, S.90-100

[2] Meissner H.: *Digitale Multimediasysteme*. Verlag Technik Berlin, Berlin 1994, S.19-47, S. 56-63

[3] Minoli D.: *Video Dialtone Technolgy*. McGraw–Hill, New York 1995, S.185-226

[4] Andleigh P., Thakrar K.: *Multimedia Systems Design*. Prentice Hall, New Jersey 1995, S. 36-44, S. 107-120, S. 139-160, S.170

[5] Murray J., vanRyper W.: *Graphics File Formats*. O´Reilly & Associates, Inc., Sebastopol 1996

[6] Unger ,F.: Pufferdimensionierung für schwankungsbeschränkte Ströme, 1998

[7] „Kompressionsverfahren für Audio und Video“, 1997/98 www.aifb.uni-Karlsruhe.de\Lehrangebot\Winter199798\Teleseminar\teleseminar\THEMA11\index.html

[8] „Chapter 4. Video and Audio Compression“, 1996 www.as.sfu.ca\cs\undergrad\CourseMaterials\CMPT365\materials\notes\Chap4\Chap4.html

[9] „CCITT/ISO standards“ cuiwww.unige.ch\OSG\info\MultimediaInfo\mmsurvey\standards.html

[10] Lougher; P.: MpegUtil, an analysis and edit program for MPEG–1 video streams., März 1995

[11] McGowan, John F.: AVI Overview. http://www.rahul.net/jfm/, 1998 http://www.rahul.net/jfm/avi.html

[12] „Work Related links“, 1998 http://www-mobile.ecs.soton.ac.uk/peter/links/work.html

[13] „MPEG Pointers and Resources“ http://www.mpeg.org/index.html/index.htm

[14] „Video for Windows Programmer's Guide“, in avi.doc

[15] File Format Specification H.261: ITU-T Recommendation H.261, in h261.doc

[16] File Format Specification MPEG2 (H.262): ITU-T Recommendation H.262, in h262.doc

[17] File Format Specification H.263: ITU-T Recommendation H.263, in h263.doc

[18] Meißner, K.: Vorlesung WS98/99, Kapitel 2: Audio-Technik, -Formate und Schnittsysteme, Kapitel 5: Video-Formate, -Kompression und Scnittsysteme, 1998

[19] Developer Press, Apple Computer, Inc.: Quick Time File Format Specification, May 1996, in mov.zip

[20] DROPS–Project.: http://os.inf.tu-dresden.de/project/index.html

7.2 Sachwortregister

Audiostream: Audiostrom, Datenstrom, der nur Audiodaten beinhaltet

Bewegtbild: steht hier für ein Bild innerhalb einer Sequenz von Bildern, die im gesamten betrachtet einen Film oder eine Animation darstellen

Bitstrom: wie Datenstrom, die Verwendung dieses Begriffs deutet darauf hin, daß der Datenstrom auf Bitebene betrachtet wird, um bestimmte Startcodes zu identifizieren

Chunk: Basis des RIFF–Fileformats, stellt einen Teil eines Formats dar, der durch den Inhalt des Chunk–Headers definiert ist, der Header beinhaltet Typ und Größe des Chunks

Codec: Coder/Decoder, definiert einen Standard zum Komprimieren und Dekomprimieren von Datenströmen
Coder: Kodierer, Einheit eines Systems, welches einen Datenstrom komprimiert und in einem klar definierten Format ablegt, der Coder und das Format des Datenstromes werden durch einen Codec definiert

Datenstrom: Folge von Bytes, bleibt inhaltlich uninterpretiert, wird hier in der Regel für große Datenmengen benutzt

Decoder: siehe Coder

DROPS: Dresden Realtime Operation System, Untersuchung von QoS–Zusagen auf Basis von verteilten Betriebssystemen

Frame: Einzelbild, meist in Verbindung mit einer Zeitangabe (z. B. 25 Frames/s), übliche Charakteristik eines Videostreams

Header: Kopf eines Datenstroms, oder auch eines Teils eines Datenstroms, beinhaltete Informationen über den folgenden Datenstrom

MPEG: Motion Picture Expert Group, Arbeitsgruppe der International Standards Organization (ISO)

QoS: Quality of Service, Dienstgüteparameter, werden zwischen Partnern ausgehandelt, die für ihre Kommunikation klar definierte Mindestanforderungen benötigen

Realzeit: Echtzeit, bedeutet hier, daß zwischen Beginn und Ende von Rechenoperationen nur eine geringe zeitliche Verzögerung vorhanden ist

Standbild: steht hier für einfache separate Einzelbilder (z. B. ein Foto)

Startcode: eine definierte Folge von Bits innerhalb eines Bitstromes, dient dem Auffinden von Teilen eines Bitstromes, die durch den Startcode identifiziert werden

Stream: Strom, hier als Datenstrom zu betrachten

Videostream: Videostrom, Datenstrom, der nur Videodaten beinhaltet