h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt

- Fachbereich Informatik -

Methoden zur Verkürzung der Umschaltzeiten beim digitalen Fernsehen

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

vorgelegt von

Simon Augustin

Referentin: Elke Hergenröther

Korreferent: Klaus Frank

Ausgabedatum: 14.2.2013

Abgabedatum: 14.8.2013

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 12.8.2013

_____

Inhaltsverzeichnis

Abstract (Deutsch)

Bei allen Vorteilen, die das digitale Fernsehen bietet - mehr Sender, weniger Bildstörungen -, gibt es auch eine Reihe Nachteile, darunter digitale Artefakte wie Blockbildung; aber vor allem sind vielen Menschen die längeren Umschaltzeiten störend aufgefallen. Meine Tests (siehe timings.xls, Diagramm 2) zeigten, dass digitales Fernsehen der ersten Generation durchschnittlich 1,1[1] Sekunden zum Kanalwechsel benötigt, digitales Fernsehen der zweiten Generation sogar über 2 Sekunden. Bei diesen Werten besteht Optimierungsbedarf, insbesondere im Vergleich zu analogem Fernsehen, wo die meisten Geräte im Schnitt eine Viertelsekunde[2] für den Kanalwechsel benötigen. Der entscheidende Grund für die langen Zeiten ist jedoch nicht der Frequenzwechsel (obwohl auch dieser eine Rolle spielt und Messungen zeigen, dass auch hier viel Zeit verloren geht - manche Geräte brauchen eine Sekunde, nur um die Frequenz zu wechseln), sondern die grundlegende Art und Weise, wie digitale Bildsequenzen codiert sind.
In dieser Arbeit werden die Gründe für die langen Umschaltzeiten herausgearbeitet und Wege vorgestellt, diese zu verkürzen. Zwei dieser Wege sind Intra-Frame-Wiederherstellung und Intra-Block-Interpolation. Alleine mit Intra-Frame-Wiederherstellung konnten die Umschaltzeiten deutlich verkürzt werden. Intra-Block-Interpolation soll versuchen, aus Zwischenbildern (welche nur Unterschiede zu Vorgängerbildern enthalten) ästhetisch akzeptable Vollbilder zu erzeugen. Intra-Frame-Wiederherstellung läuft auf Stream-Ebene, also unabhängig vom Decoder.

[1] 1.1 Sekunden ist die halbe Strecke zwischen der Durchschnittszeit zum Umschalten im selben Bouquet und in unterschiedlichen Bouquets zwischen allen getesteten Geräten [timings.xls]. Leider ist darunter der mit Baujahr 2005 schon recht betagte DUAL DVB-T 1000, der die Werte merklich erhöht.

[2] Der tatsächliche Wert ist 0.248s

Abstract (English)

Despite all the advantages of digital television, like more TV programmes and less noise, there are a couple of disadvantages, among them digital artefacts like blocking, but the key issue that many people find annoying is the long channel switch time. Tests showed that first generation digital TV averages at 1.1 seconds for changing the channels, second generation digital (HD)TV averages at 2-3 seconds (depending on the set). These values are unacceptable and need to be optimised, especially when comparing it to analogue TV where most sets do not take longer than a quarter of a second. The main reason for the long switching times is not changing the frequency (although this does contribute, and measurements indicate that changing the frequency does lose time - some receivers took one second for this process alone), but in the basic way a digital TV broadcast signal is coded.
This thesis will work out the reasons for the long switching times as well as show ways to shorten them. The two main ways are Intra-Frame-Restoration and Intra-Block-Interpolation. Tests with Intra-Frame-Restoration showed that this measure alone could significantly reduce the worst-case-timings. Intra-Block-Interpolation shall attempt to extract information from Delta-Frames (containing only differences from previous pictures) in order to create aesthetically acceptable Full-Frames. Intra-Frame-Restoration runs on stream level, so it can be used independently from the decoder.

Konventionen für Zahlendarstellung:

Keine Angabe: In der Regel dezimal (Basis 10). Möglicherweise existieren jedoch Hexadezimalzahlen, die nicht explizit ausgewiesen werden. In der Regel kann man die Basis dieser Zahlen jedoch aus dem Kontext erkennen.
(hex) dahinter oder 0x davor: Hexadezimal (Basis 16)
d dahinter: Explizit Dezimal (Basis 10). Wird in Umgebungen mit dominanten Vorkommen nichtdezimaler Zahlen als Komfortangabe verwendet.
b dahinter oder 0b davor: Binär (Basis 2).

Abkürzungsverzeichnis

Abkürzungen sind in der Regel im selben Absatz aufgeschlüsselt, in dem sie zuerst verwendet werden. Nicht erklärte Abkürzungen sind in der Regel entweder allgemein bekannt oder bekannte Eigennamen.
2D/3D: Zweidimensional/Dreidimensional.
AFC: Automatic Frequency Control. Automatische Feinabstimmung von Sendern im Empfangsgerät.
ATSC: Advanced Television Standard Committee: Komitee zur Festlegung von Standards in HDTV, z.B. welche Codecs verwendet werden, im Vergleich zu MPEG, welches ein Komitee zur Festlegung von Standards in Codecs ist.
AVC: Advanced Video Codec. Alternative Bezeichnung für H.264.
CABAC: Context-Adaptive Binary Arithmetic Coding: Eine angepasste Form der Arithmetischen Codierung bei H.264.
CAT: Conditional Access Table. Für diese Arbeit unwichtig.
CAVLC: Context-Adaptive Variable Length Coding: Eine alternative Form zu CABAC bei H.264, für Bereiche, in denen weniger Prozessorleistung zur Verfügung steht.
CD: Compact Disc.
CPU: Central Processing Unit. Wird oft als das Gehirn eines Computers bezeichnet.
DCT, DC- /AC- Koeffizient: Discrete Cosinus Transform, DC/AC steht üblicherweise für Direct Current, Alternating Current (Elektrizität), hier bedeutet DC die Gesamthelligkeit (Frequenz von 0, wie bei DC-Spannungen) und AC-Koeffizienten für Bildanteile mit einer Frequenz größer als 0.
DSP: Digital Signal Processor. Ein auf bestimmte Datentypen spezialisierter Prozessor. Während eine CPU mit unterschiedlichen Datentypen in etwa gleich gut umgehen kann, kann

ein DSP mit einem bestimmten Datentyp besonders gut, dafür mit allen anderen Datentypen schlecht oder gar nicht umgehen.

DTS/PTS: Decode TimeStamp, Presentation TimeStamp. Zeigt, in welcher Reihenfolge der Decoder die empfangenen Frames auf dem Bildschirm anzeigen soll.

DVB (T/S/C): Digital Video Broadcasting (Terrestrial/Satellite/Cable). Standards zur Übertragung des Digitalfernsehens (über Antenne, Satellit, Kabel).

EPG: Electronic Program Guide. Elektronische Fernsehzeitung.

Exp-Golomb: Exponential-Golomb. Name für Codierungsmethode.

(F)BAS: (Farb-)Bild- Austast- Synchronsignal

GOP: Group Of Pictures: Zusammenhängende Sequenz von (komprimierten) Bildern. Üblicherweise besteht eine GOP aus einem I-Frame, gefolgt von mehreren Nicht-I-Frames. Der nächste I-Frame kennzeichnet den Beginn der nächsten GOP.

HDMI: High Definition Multimedia Interface. Eine Video-Schnittstelle für HD-Fernseher.

hex: hexadezimal.

HSV: Hue Saturation Value. Alternative Ebenenzerlegung von Farbbildern, welche intuitivere Farbbearbeitung erlaubt, als wenn man die Kanäle in Rot, Grün und Blau trennt.

I-/P-/B-/SI-/SP-Frames: Intra/Predictive/Bidirectional Predictive/Switching Intra/Switching Predictive-Frames.

IDR: Instantaneous Decoding Refresh. Zeigt in einem H.264-Videostrom eine Stelle auf, an der mit dem Decodieren begonnen werden kann.

IEC: International Electrotechnical Commission

ISO: Internationale Organisation für Normung

ITU: Internationale Fernmeldeunion. Verweist auf Video Coding Experts Group (VCEG), die sich mit MPEG zusammengeschlossen haben.

JPG/JPEG, BMP, PNG: Bildspeicherformate: JPG/JPEG: Joint Picture Expert Group. BMP: BitMaP. PNG: Portable Network Graphic.

MB: In dieser Arbeit steht MB üblicherweise für Macroblock, entgegen der bekannteren Bedeutung Megabyte.

MPEG: Motion Picture Expert Group. Komitee zur Verabschiedung von Standards für digitale Videokompression.

NAL: Network Abstraction Layer. Ein Format zur Kapselung von H.264-Videoströmen.

NTSC: National Television Standard Committee. Amerikanischer Farbfernsehstandard.

OFDM: Orthogonal Frequency Division Multiplex. Ein Verfahren zur Übertragung von DVB-T Signalen.

PAL: Phase Alternating Line. Analoger Europäischer Farbfernsehstandard.
PAT: Program Association Table. Paket im TS, welches die (Fernseh-)Programmbelegung enthält.
PC: Personal Computer.
PCM: Puls Code Modulation. Standardmethode zur Binarisierung (Digitalisierung) analoger Signale.
PDF: Portable Document Format.
PES: Packetized Elementary Stream. Ein PES-Header kann Frame Headers vorgeschaltet werden und enthält unter anderem DTS/PTS.
PID: Programme ID. Paketadresse in einem TS.
PMT: Program Map Table. Paket in TS, welches die Zusammensetzung des Datenstroms eines (Fernseh-)Programms enthält.
QAM: QuadraturAmplitudenModulation. Methode, mehrere Signale über Hilfsträgerfrequenzen auf ein einziges Signal aufzumodulieren.
SECAM: Sequential Couleur à Memoire. Analoger Französischer Farbfernsehstandard.
SD/HD/HDTV: Standard Definition/High Definition Television. Normales/Hochauflösendes Fernsehen.
TCP/IP: Transport Control Protocol/Internet Protocol. Das im Internet gebräuchliche Datenprotokoll.
TS: Transport Stream. Paketbasierter Datenstrom, der für DVB verwendet wird.
USB: Universal Serial Bus. Eine weit verbreitete Computerschnittstelle.
VLC: Variable Length Code. Allgemeine Bezeichnung für Komprimierungsmethoden, bei denen Datenworte unterschiedlich lang sind. Bei unkomprimierten Daten sind Datenworte gleich lang.
VCO: Voltage Controlled Oscillator. Ein Spannungsgesteuerter Oszillator ist ein Gerät zur Umwandlung von Spannung in Frequenz.
VCR/MAZ: Video Cassette Recorder / Magnetische AufZeichnung. Verfahren zur analogen Speicherung von Videosequenzen auf Magnetband.
VOP: Video Object Plane. Entspricht Frame bei MPEG-4.

Begriffsverzeichnis

Hier stehen nur Begriffe, die im Text nicht erklärt werden, um Redundanzen zu vermeiden.

Analogabschaltung: Abschaltung analoger Fernsehsendungen. Über Antenne wird seit März 2010 nicht mehr analog gesendet, über Satellit seit April 2010, für Kabel ist die Analogabschaltung für 2017 geplant, da noch recht große Anteile der Kabelnetzzuschauer analog empfangen.

Artefakt: Künstlich erstelltes Objekt, in der realen Welt Werke, in dieser Arbeit Programme, Texte; im Computer, insbesondere der Computergrafik bezeichnen Artefakte erkennbare Kompressionsverluste, die z.B. an runden Flächen vor einfarbigen Hintergründen gut erkennbar sind.

Bewegungsvektor: Richtungs-(und Entfernungs-)-angabe einer Bewegung.

Bildsequenz: Eine lückenlos zusammengehörige Menge an Einzelbildern. Weitere Verwendung: Videosequenz. Die Bildsequenz ist hierbei zeitlich (ausreichend) lückenlos.

Bildschirmterminal: Gerät zur Betrachtung und zum Senden elektronischer Nachrichten. Der Vorläufer des Arbeitsplatz-PCs in Büros.

Binärbaum: Datenstruktur, die aus Knoten und Kanten besteht, genau einen Wurzelknoten hat und azyklisch ist. Jeder Knoten hat maximal 2 Kanten.

Bitstring: Sequenz aus Bits mit beliebiger Länge. Ein Bit kann nur die Information 1 oder 0 haben.

Bouquet: Vereinigung mehrerer Fernsehsender auf derselben Frequenz. Die Bild- und Tonsignale der Sender werden so verwoben, dass sie bei der Wiedergabe eindeutig zugeordnet werden können.

Byte: Bitstring mit der Länge 8, gebräuchlichste Speicherzellenlänge in der Informatik.

Byte-aligned: Position in einem Bitstring, die durch 8 teilbar ist.

Bytestream: Zusammenhängende Sequenz aus Bytes, deren Anfang und Ende nicht zwangsläufig vorhanden sein müssen, einen Bytestream mit festem Anfang und Ende bezeichnet man auch als Mitschnitt eines Bytestreams. (Datenstrom)

Capture: Synonym für Mitschnitt eines Bytestreams.

Chroma: Farbinformation, Farbanteil, Farbsignal.

Codec: Akronym aus Coder/Decoder. Ein Gerät, Programm oder Algorithmus, das aus einem Bytestream einen anderen Bytestream mit derselben Information, aber anderen Eigenschaften (z.B. weniger Bytes) erzeugen kann (Coder), sowie den originalen Bytestream aus dem

Erzeugten zurückgewinnen kann (Decoder), letzteres häufig auf räumlich von der Erzeugung getrennter Hardware.

Codewort: Hier: Ein von einem Codec komprimiertes Datenwort ist ein Codewort.

Delta/Delta-Frames: Unterschiede/Bilder, die nur Unterschiede zu vorhergehenden (und nachfolgenden) Bildern enthalten.

Diskrepanz: Unterschied, Größe des Unterschieds.

Entropiecodierung: Komprimierung unter Abhängigkeit der Vorkommenswahrscheinlichkeit eines Zeichens. Z.B. taucht das 'E' in deutschen Texten besonders häufig auf und würde nach einer Entropiecodierung im kürzesten Codewort resultieren.

Epistasie: Eigenschaft, dass Veränderungen am Anfang eines Objekts (real oder virtuell) größere Auswirkungen haben als Veränderungen am Ende. Dies tritt auf, wenn das Ende vom Anfang abhängig ist, z.B. in einer Kausalkette.

Forward-Error-Correction: Vorwärtsfehlerkorrektur. Eigenschaft, Daten zu senden, die später evtl. auftretende Fehler korrigieren können. Der Nachteil ist, dass vorausgesetzt wird, dass Daten einmal bereits korrekt übertragen worden sein müssen.

Frame: Bild, Vollbild.

Frequenzwechsel: Veränderung der Abstimmfrequenz des Tuners, um einen anderen Kanal empfangen zu können.

Gradient: Farbabstufung, Farbübergang.

Halbbild/Vollbild: In einem Vollbild werden alle Bildinformationen linear in der richtigen Reihenfolge übertragen. Beim Zeilensprungverfahren (siehe dort) werden zwei Halbbilder (die halb so hoch sind) miteinander verwoben, um ein Vollbild zu erhalten, pro Halbbild werden die Bildinformationen jeder zweiten Zeile übertragen.

Header: Eigentlich: Kopfzeile. In der Informatik steht Header für Daten, die den eigentlichen Nutzdaten, die man senden will, vorausgehen und benötigt werden, um die Nutzdaten korrekt entschlüsseln zu können, z.B. welche Auflösung ein Bild hat, bevor die eigentlichen Bilddaten kommen.

Initialisierung: Setzen von Variablen bei der Erstbenutzung, wenn der eigentliche Inhalt dieser Variablen noch undefiniert ist. Ein Zähler wird bei der Erstbenutzung üblicherweise mit 0 initialisiert, anstatt mit einer Zufallszahl.

Interlaced: Siehe Zeilensprungverfahren.

Interpolation: Verfahren, um den Inhalt von Lücken in Datenreihen, hier Bildmaterial, zu füllen.

Intra/Inter(-Frame): Intra bedeutet hier, dass das Material in sich geschlossen dargestellt werden kann, während ein Inter-Frame auf mindestens einen weiteren Frame angewiesen ist, um angezeigt werden zu können.

Kathodenstrahl: Wird in einer Bildröhre erzeugt, mit Hochspannung beschleunigt und trifft dann auf eine Phosphorschicht, die mit einer regelbaren Intensität (Helligkeit) zu leuchten beginnt. Wird mit Elektromagneten abgelenkt, um Bilder zu erzeugen.

Koeffizient: Wert, Datenwert.

Konkatenieren: Zusammenhängen.

Komprimierung: Vorgang der Verkleinerung eines Datenstroms (siehe Bytestream), meist mittels eines Codec.

Lesezeiger: Stelle im Bytestream, an der gerade gelesen wird.

Marginal: Rand- (z.B. Marginale Bedeutung: Bedeutung nur am Rande). Sehr gering.

Metadaten: Beschreibungsdaten, also Daten über Daten. Siehe Header.

Obsolet: Veraltet. (Substantiv: Obsoleszenz)

Opak: Undurchsichtig. Gegenteil: Transparent.

Oszillator: Schwinger, Schwingkreis. Apparatur, die Schwingungen, häufig mit einstellbarer Frequenz, erzeugt.

Phasenabgleich: Synchronbringen des Empfängers mit der Phase des Senders, nachdem die Frequenz korrekt eingestellt wurde. Siehe Diagramm rechts.

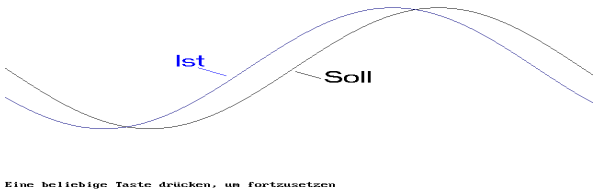


Diagramm 1: Phasenabgleich

Prädiktion: Vorhersage.

Progressive: Gegenteil von Interlaced.

Raw: Roh. In der Informatik bedeutet das, dass Daten unbehandelt (häufig nur durch Darstellung im Hexadezimalformat) dem Betrachter oder verarbeitenden Algorithmus präsentiert werden.

Rechenzeit: Zeit, die eine CPU braucht, um eine Aufgabe zu bearbeiten.

Redundanz: Mehrfachauftreten (z.B. einer Information).

Run (Lauf, hintereinander): Hier: Sequenz von Koeffizienten mit demselben Wert (0).

Schwarzwert: Wert, der einer analogen Repräsentation von Schwarz entspricht. Bei analogen Videosignalen ist 0V Schwarz, 0,7V ist Weiß.

Skalieren: Veränderung der Größe.

Spatial: Räumlich.

Speichermatrix, Array. Sequenz aus Computer-Speicherzellen.

Stream: Strom, Datenstrom. Siehe Bytestream.

Temporal: Zeitlich.

Terminieren: Beenden.

Timestamp: Zeitstempel. Daten über die Uhrzeit, zu der das zugehörige Datenpaket losgeschickt wurde.

Tuner: Gerät, um aus einer Menge frequenzmultiplexter Programme ein einzelnes Programm zu isolieren.

Worst/Average/Best case: Schlimmster/Mittlerer/Bester Fall.

YUV-Farbmodell: Ein Farbmodell, nach dem das Bild in die Ebenen Helligkeit, Rotabweichung und Blauabweichung zerlegt wird.

Zeilensprungverfahren: Verfahren, um bei niedrigen Bandbreiten eine höhere Auflösung (temporal oder spatial) zu ermöglichen. Hierbei werden statt Vollbilder (progressive) Halbbilder gesendet, die bei jedem Bild um eine halbe Zeile versetzt werden, was das wahrnehmbare Flimmern halbiert und die Y-Auflösung verdoppelt.

1. Einleitung

1.1 Motivation

Vergleicht man analoges mit digitalem Fernsehen, bemerkt man, dass die Zeit, die benötigt wird, um den Sender zu wechseln, gestiegen ist. Tests ergaben, dass analoge Geräte oft sehr viel schneller schalten als digitale. Um auch nach der Analogabschaltung die Umschaltzeiten von analogen Fernsehern nachvollziehen zu können, wurde eigens ein Video aufgenommen, das verschiedene Geräte mit einem Baujahr zwischen 1975 und 2000 zeigt. [AUA-2013i]
In der Regel braucht ein analoges Fernsehgerät nicht mehr als eine Viertelsekunde, um den Sender zu wechseln, bei digitalen Geräten ist eine Viertelsekunde oft die untere Grenze, die obere kann durchaus bei über drei Sekunden liegen (siehe Diagramm 2). Diese Werte führen vielerorts zu Verärgerung.[3]
Fernsehzuschauer kann man vom Umschaltverhalten im Interesse dieser Arbeit in zwei Lager teilen: Erstens: Die Planer. Fernsehzuschauer, die ihr Sehverhalten nach der Fernsehzeitung planen und in der Regel nur ein festes Programm schauen. Diese Personen stört eine lange Umschaltzeit in der Regel weniger. Zweitens: Die Zapper. Fernsehzuschauer, die kein festes Programm haben und die bevorzugt durch die Programme "zappen", auf der Suche nach interessantem Inhalt. Ein Zapper verweilt etwa ¼ bis 1 Sekunde auf einem uninteressanten Kanal.[4] Die Umschaltzeit des Fernsehgeräts/Decoders muss auf diesen Wert addiert werden. Selbstverständlich sind beide Lager nicht streng getrennt, so kann z.B. ein Planer während einer Werbepause vorübergehend zum Zapper werden.
Nach der Analogabschaltung berichteten mir mehrere Personen von längeren Umschaltzeiten im Digital-TV, sowie dass sie diese von unangenehm störend bis hin zu inakzeptabel empfinden. Dies - sowie meine bereits vorhandenen Kenntnisse über MPEG-2 Videokomprimierung - war der Ansporn für diese Arbeit.

[3] Eine Websuche nach "Umschaltzeiten bei Digital-TV" zeigt mehr Quellen auf, als ich listen kann, ein paar davon finden sich im folgenden Kapitel
[4] Zu diesem Thema gibt es Studien, die Ergebnisse waren jedoch nur kostenpflichtig abrufbar.

1.1.1 Messwerte

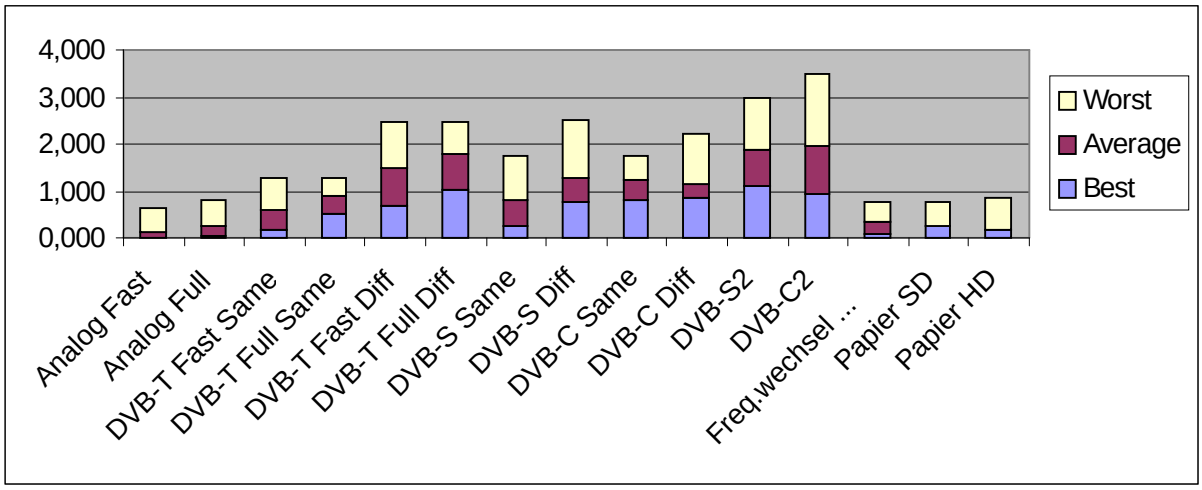


Diagramm 2: Gemessene Umschaltzeiten für verschiedene Medientypen. Y-Achse in Sekunden.

Im Internet findet man viele, auch neuere, Einträge von Benutzern von HDTV, die 2,0-4,0 Sekunden angeben. Ein Benutzer in einem Thread von 2004 gab einen Average Case von 2,4s an, einer von 2010 gab ganze 3 Sekunden bei einem Neugerät an. Er gab ebenso an, eine Versuchsreihe über alle Sender durchgeführt zu haben. [HIF-2013] Weitere Benutzer gaben 3-4 Sekunden an. Diese Werte kann man durchaus als belastbar ansehen.

Die in diesem Diagramm aufgelisteten Werte wurden vom Autor wie folgt ermittelt:

Aufzeichnung mittels einer Videokamera, die nach dem PAL System im Zeilensprungverfahren bei 50Hz arbeitet, also 50 Halbbilder bzw. 25 Vollbilder pro Sekunde aufzeichnet.

Auswertung mittels eines Videorekorders mit Einzelbildfortschaltung. Hierbei ist bekannt, dass der Rekorder Vollbilder weiterschaltet, d.h. 25 Einzelbildfortschaltungen entsprechen einer Sekunde, was durch Abfilmen einer Stoppuhr bestätigt werden konnte. Dieses Verfahren ermöglicht eine Messgenauigkeit von 1/25 Sekunden, was bei interessanten Messperioden von 0.1-5.0 Sekunden ausreichend genau ist.

Aufstellung einer Messreihe nach folgenden Kriterien:

Wichtig: Intra-Bouquet-Messung: Umschaltung der Sender ohne Umschaltung der Frequenz.

Wichtig: Inter-Bouquet-Messung: Umschaltung zwischen Sendern auf unterschiedlichen Frequenzen.

Wichtig, wenn möglich: Umschaltung zwischen 2 HD-Sendern. Diese Umschaltung ist immer Inter-Bouquet, da zur Zeit nur 3 Sender unverschlüsselt in HD ausstrahlen und diese auf unterschiedlichen Frequenzen gesendet werden.

Weniger wichtig: Umschalten von HD auf SD und von SD auf HD.

1.1.2 Ursachen (grob)

Die Hauptursache für die langen Umschaltzeiten ist die Komprimierung des Videosignals, insbesondere die Entfernung von redundanten Bildinhalten, was ja der Hauptbestandteil der Komprimierung ist.
Einige der Geräte brauchen von HD auf SD länger als von SD auf SD, ebenso brauchen manche von SD auf HD länger als von HD auf HD. Diese Diskrepanz ist in der Neuinitialisierung der Videowiedergabehardware zu suchen und kein Bestandteil dieser Arbeit. Nichtsdestotrotz sollte dieser Sachverhalt von Entwicklern von Decodern im Hinterkopf behalten werden, wenn die Umschaltgeschwindigkeit zur Debatte steht.
Beim digitalen Fernsehen werden mehrere Sender auf einer Frequenz übertragen (Bouquet, die Anzahl der Sender im Bouquet ist von Übermittlungsverfahren und evtl. vorhandenen HD-Kanälen, die mehr Bandbreite brauchen, abhängig) [MKB-2013][ASE-2013i], die Umschaltzeit ist häufig abhängig davon, ob beide Sender auf derselben oder auf unterschiedlichen Frequenzen übertragen werden. Je nach Gerät tun sich hier große Unterschiede auf.

1.1.3 Ergebnisse (grob)

Durch Intra-Frame-Wiederherstellung (siehe Kapitel 4) können - wenn mindestens das untere Drittel des Bildes wiederhergestellt werden kann - die Worst-Case Umschaltzeiten um knapp 1/10 Sekunde reduziert werden. Wird weniger wiederhergestellt, kann man sich darüber streiten, ob das Bild als "wiederhergestellt" zu betrachten ist (darstellen werde ich es trotzdem). Die bisherigen Worst-Case-Zeiten, die darüber liegen, werden zum Best-Case. Geht man den gesamten Weg (minimale Bildinformation aus I-Frame), gewinnt man 0,14 Sekunden. Zusätzliche Techniken, die in Kapitel 5 und 7 vorgestellt werden, erlauben weitere Verkürzungen, bis hin zur Dauer eines Einzelbildes.

1.2 Aufbau der Arbeit

Die Arbeit ist in zwei Teile untergliedert, nämlich einem Teil für SD-Digitalfernsehen und einem Teil für HD-Digitalfernsehen. Die einzelnen Teile sind wie folgt unterteilt: Entsprechend dem Lösungsfindungsprozess werden, nach dem Vergleich von analogem und digitalem Fernsehen im ersten Teil, zunächst detailliert die Grundlagen des digitalen Fernsehens, nämlich die Komprimierung und der Aufbau des Videostroms, dargelegt, um die Ansatzpunkte zur Lösung, also zur Verkürzung der Umschaltzeiten, herauszuarbeiten. Im Anschluss daran wird die Lösung vorgestellt. Der HDTV-Teil (Kapitel 6) enthält prinzipiell denselben Aufbau für die Varianten des hochauflösenden Fernsehens in verkleinertem Maßstab, da viele der Grundprinzipien bei manchen Codecs ähnlich sind.

Teil 1

2. Grundlagen - MPEG-2

2.1 Einleitung: Vergleich Analoges/Digitales TV

Bei digitalem Fernsehen werden Bildinformationen völlig anders übertragen als bei analogem Fernsehen. Um zu verstehen, wieso die Umschaltzeiten bei digitalem Fernsehen deutlich länger sind, müssen beide Darstellungsarten vergleichend erklärt werden.

2.1.1 Analoges Fernsehen: Grundlagen

Beim analogen Fernsehen wird traditionell ein Kathodenstrahl zeilenweise von oben nach unten über den Bildschirm bewegt und die Bildinformationen in Echtzeit durch Steuerung der Intensität des Kathodenstrahls dargestellt. Trennung des Signals in Grundfarben würde den Rahmen dieses Dokumentes sprengen und ist dem weiteren Verständnis nicht zuträglich. Synchronisierung wird durch horizontale und vertikale Austastlücken - ein Signalpegel von minus 0,3V im analogen Signal, im Vergleich zum Schwarzwert, der üblicherweise bei 0V liegt - realisiert. [RWV-2009, S.10ff] Bei der europäischen PAL-Fernsehnorm (das französische SECAM unterscheidet sich hier nur in der Beschaffenheit des Farbsystems) beträgt die vertikale Synchronfrequenz 50Hz, durch das Zeilensprungverfahren - ein Verfahren, das eine höhere Bildqualität bei eingeschränkter Videobandbreite erlaubt [WZE-2013] - bedeutet das 50 Halbbilder oder 25 Vollbilder pro Sekunde. Die horizontale Synchronfrequenz beträgt 15,625kHz, was sich bei älteren Geräten in einem hörbaren Fiepen äußert. [RWV-2009 S.15] Die Geschwindigkeit der erstmaligen Synchronisierung beim Einschalten oder beim Übergang von keinem Empfang zu vorhandenem Empfang ist von der Abstimmung des Referenzoszillators (in der Regel ein einfacher analoger Schwingkreis) im Gerät abhängig und kann unterschiedlich lange dauern. Es gibt Geräte, die über eine Sekunde dafür benötigen. Die Umsynchronisierung von einem Kanal auf einen anderen dauert jedoch bei keinem mir bekanntem Gerät (da ich antike Fernsehgeräte sammle, sind das einige) lange genug, um vom Auge bemerkt zu werden, selbst, wenn Offset und Frequenz differieren. Die vertikale Synchronisation bei analogem Fernsehen ist jedoch als recht langsam bekannt. Das

äußert sich darin, dass beim Umschalten das neue Bild "einrollt", d.h. die vertikale Austastlücke ist als schwarzer Balken zu sehen, der den Bildschirm hinaufrollt, was je nach Gerät zwischen 1/25s und 3/4s dauert [AAT-2013i] und bei allen mir bekannten Geräten mit einem Trimmpotentiometer (Poti) beeinflusst werden kann. Für den menschlichen Betrachter ist das störend, aber der Bildinhalt ist dennoch erkennbar. Höhere Werte des Potis bedeuten schnelleres Synchronisieren, aber geringere Stabilität bei schlechtem Empfang oder Magnetbandaufzeichnungen (VCR oder MAZ genannt) - das Bild kann also von alleine beginnen durchzurollen, selbst wenn es bereits stabil war. Niedrigere Werte bedeuten entsprechend langsameres Synchronisieren (das Rollen wird sichtbar oder gar störend), aber bessere Stabilität bei schlechtem Empfang oder VCR-Wiedergabe. [AUA-2013i]

2.1.2 Digitales Fernsehen: Grundlagen

Der erste große Unterschied beim digitalen Fernsehen ist nicht, dass das Signal nun digital ist (was z.B. einer binär codierten Helligkeitsinformation entspräche), sondern, dass die Bildinformationen komprimiert, also nicht mehr direkt in Echtzeit darstellbar gesendet werden. Die Bilddaten müssen jetzt gepuffert, decodiert und dekomprimiert werden, um ein Bild zu erhalten. Die echtzeitnahestmögliche Decodierung würde einem Update des Bildes nach Decodierung eines vollständigen Macroblocks (16*16 Pixel, siehe Kapitel 2.2.1) entsprechen, aber der Zusammenhang Amplitude des Videosignals = Helligkeit des Bildpunkts ist aufgebrochen (es wird bei diesem Vergleich davon ausgegangen, dass das analoge Videosignal als (F)BAS-Signal und das digitale Videosignal als Bitstring vorliegt). Komprimierung wird auf zweierlei Art erreicht: Entfernen von spatialen und temporalen Redundanzen.

2.2 Aufbau des digitalen Bildes

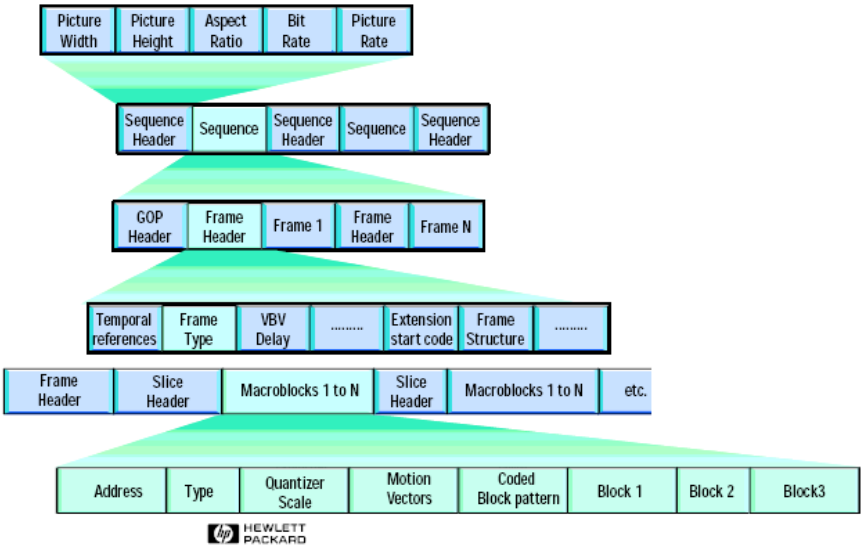


Diagramm 3: Aufbau eines MPEG2-Videostreams [HPM-2006]

Diagramm 4: Aufbau eines MPEG-2 Bildes

2.2.1 Blocks/Macroblocks

Fängt man bei der kleinsten darstellbaren Information an, so stellt man fest, dass Bildpunkte (Pixel) nicht mehr einzeln übertragen werden. Stattdessen werden diese beim Senden in Gruppen von 8*8 aufgeteilt und durch eine Diskrete Cosinus Transformation (DCT) in den Ortsfrequenzbereich transformiert. [EHG-2012, Kapitel Kompression, Folien 33ff] Statt also nun die Helligkeit der Farbkanäle (R,G,B) jedes Bildpunkts zu übertragen, wird der Helligkeitswerteverlauf (Y) und der Farbwerteverlauf (Cb, Cr für Blau- bzw. Rotabweichung) einer Gruppe von Bildpunkten übertragen. So kann man die Abhängigkeit der Helligkeit eines Bildpunkts von dessen Umgebung codieren. Der Vorteil darin ist, dass in Bildregionen mit

wenig Details (hohe spatiale Redundanz) - z.B. ein wolkenloser Himmel - deutlich weniger Informationen übertragen werden müssen. Anstatt pro Block 64-mal zu senden: "Dieses Pixel ist blau", mit kaum wahrnehmbaren Variationen in Helligkeit und Sättigung, wird gesendet: "Dieser Block ist hellblau und wird nach unten hin leicht dunkler". Für diese Information müssen nur zwei bis vier Werte - in der DCT sogenannte Koeffizienten - übertragen werden (Einer für Helligkeit, einer für das „Dunklerwerden“ plus ein oder zwei weitere für Farbe). Liegt das in diesem Beispiel genannte „Dunklerwerden“ unter einem bestimmten Schwellwert, so wird nur noch ein Koeffizient - die Helligkeit – übertragen (und die Farbe). Die Gesamthelligkeit des Blocks wird als DC-Koeffizient, die Verlaufskoeffizienten (also alle anderen) als AC-Koeffizienten bezeichnet.

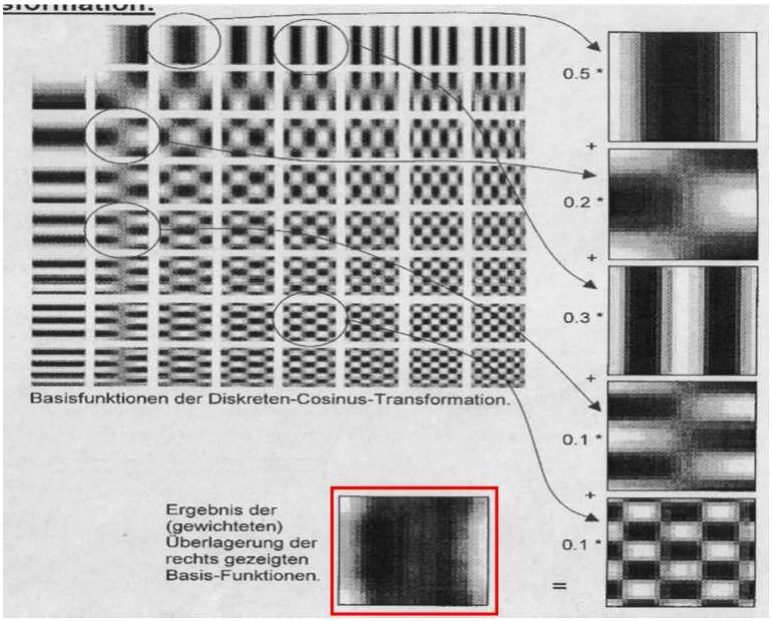


Diagramm 5: Inverse DCT- Funktionstabelle. Aus: [EHG-2012]

Die Farbe und Farbverläufe werden in separaten Blocks übertragen. Durch die beim Menschen im Verhältnis zur Helligkeitswahrnehmung reduzierte Farbdetailwahrnehmung können hier weitere Daten eingespart werden, in der Regel kommen 2 Farb-Blocks auf 4 Helligkeits-Blocks. Immer 4 Helligkeits-Blocks gehören zusammen und werden gefolgt von 2-8 Farb-Blocks, abhängig vom Chroma-Modus. Diese Zusammenstellung nennt man Macroblock. Es gibt 3 Chroma-Modi: 4:2:0, welcher 2 Chroma-Blocks mit halber horizontaler und vertikaler Auflösung enthält und der gängigste Modus für DVB ist. 4:2:2 enthält 4 Chroma-Blocks mit halber horizontaler Auflösung und 4:4:4 enthält 8 Chroma-Blocks mit voller Auflösung.[I32-1995, S.27] Das 4:4:4 Format wird im professionellen Bereich bevorzugt.

Die DCT-Koeffizienten in einem Macroblock werden Huffman-codiert (komprimiert) übertragen (vgl. [WHK-2013]). Bei der Huffman-Codierung wird ein Binärbaum aufgebaut, in dessen Blattknoten die zu codierenden Symbole gespeichert werden, der Pfad zum Symbol entspricht dem gesendeten Codewort. Die Pfadlänge und entsprechend die Anzahl der Bits pro Codewort ist abhängig von der Wahrscheinlichkeit des Symbols. Der zur Codierung und Decodierung benötigte Huffman-Baum ist fest vorgegeben und befindet sich im Anhang B zu [I32-1995], muss also in jedem En- und Decoder fest gespeichert sein.

2.2.2 Slice

Eine Slice setzt sich aus einer beliebigen Menge Macroblocks zusammen [I32-1995, S.25; 6.1.2].
Macroblocks haben ein address_increment-Feld, welches erlaubt, die Macroblocks zu nummerieren, aber auch, Macroblocks zu überspringen. Verändert sich ein Macroblock von einem Bild auf das nächste nicht, muss er nicht erneut übertragen werden; ändert er sich nur geringfügig, werden nur die Änderungen übertragen. So werden temporale Redundanzen entfernt. Ein weiteres Verfahren, um temporale Redundanzen zu entfernen, sind Bewegungsvektoren. Diese erlauben, dass bewegliche Objekte in Folgebildern nicht erneut übertragen werden müssen. Da in der Realität Bewegungen selten so invariant ablaufen wie auf dem Papier, wurde MPEG4 (ISO/IEC 14496-2) mit einer Reihe von affinen Transformationen(Drehung, Skalierung...) ausgestattet. Mehr dazu in Kapitel MPEG4.
MPEG2 muss sich für diese Komplikationen mit Delta-Macroblocks oder prediction-error-Macroblocks aushelfen. Bei H.264 ermöglicht eine feinere Block-Aufteilung genauere Bewegungsprädiktion. Mehr dazu im Kapitel H.264.
Das address_increment-Feld erlaubt des weiteren Rückschlüsse auf die horizontale Position eines Macroblocks. Slices haben einen slice_start_code, der zum Einen einen Synchronisationspunkt bereitstellt (zum Schneiden, für Trick-Modes wie Schnellvorlauf), zum Anderen die vertikale Position der Slice auf dem Bildschirm angibt (dies wird in [I32-1995] Kap.6.2.1 angegeben, widerspricht jedoch der Idee einer Slice wie in Diagrammen 6-8 und 6-9 dort angegeben).
Die Angabe der vertikalen Position (das Nichtvorhandensein von mehr als einer Slice auf einer Zeile) konnte durch Analysen realer DVB-Streams bestätigt werden. Der Slice-Synchronisationscode ist 00 00 01 xx (hexadezimal), wobei xx Werte von 01-AF annehmen kann und, wie ermittelt, die Y-Position der Slice darstellt.

2.2.3 Bild

Ein Bild setzt sich aus einer beliebigen Anzahl Slices zusammen. Bei einem vollständig definierten Bild (I-Frame, siehe unten) mit der SDTV-Auflösung 720*576 sind das 36 Slices zu jeweils 45 Macroblocks.

Bildtypen:
Es gibt 3 Arten von Bildern bei MPEG2: I-Frames, P-Frames und B-Frames.
Bei einem I-Frame (Intra) sollte jeder Macroblock im Bild definiert (in der Übertragung vorhanden) sein. Da dies aber eine große Menge Speicherplatz benötigt, kann in einem bandbreiten- oder speicherplatzbegrenzten Datenstrom nicht jedes Einzelbild als I-Frame codiert werden (ohne Speicherplatzbegrenzung ist es möglich). P- und B-Frames erlauben das Entfernen von temporaler Redundanz, was, wie bereits beschrieben, durch das Auslassen bereits gesendeter identischer Bildinformationen, sowie Bewegungsvektoren, um Objekte, die im nächsten Frame nicht an derselben Stelle sind, aber sich weiter nicht verändert haben, nicht erneut übertragen zu müssen. Geringfügige Veränderungen oder subpixelgenaue Bewegungen [MPEG2 erlaubt nur eine Genauigkeit von ½ Pixel, bei MPEG4 (sowohl ISO/IEC14496-2 als auch ISO/IEC14496-10) ist es ¼ Pixel] werden als prediction-error-Daten übertragen. Der Encoder vergleicht das codierte Bild, das nur Bewegungsdaten vom vorhergehenden Bild enthält, und entscheided anhand der Größe des Unterschieds und der verbleibenden Bandbreite, ob ein Intra-Macroblock oder ein prediction-error-Block plus Bewegungsvektoren gesendet wird.
P-Frames (Prediction) verwenden hierbei Informationen aus vergangenen Bildern ("was hat sich verändert?"), neue Informationen, z.B. von bewegten Objekten aufgedeckte, müssen als Intra-Macroblocks 1:1 übertragen werden.
B-Frames (Bidirectional Prediction) verwenden Informationen sowohl aus vergangenen, als auch aus zukünftigen Bildern, was erlaubt, von bewegten Objekten aufgedeckte Bildinformationen nicht (d.h. erst im nächsten I- oder P-Frame) übertragen zu müssen. [I32-1995, 6.1.1.5]
Dies hat jedoch eine Umordnung der Frames bei der Darstellung zur Folge, genannt "Frame-Reordering".

2.2.4 Frame-Reordering

B-Frames können Bilddaten sowohl aus vergangenen als auch aus zukünftigen Bildern beinhalten. Dies bedingt, dass diese zukünftigen Bilder bereits gesendet wurden, und dass die Reihenfolge der Bilder bei der Darstellung eine andere ist als beim Empfang, da B-Frames den jeweils nächsten gesendeten, nichtbidirektionalen Frame brauchen. [I32-1995, 6.1.1.11] Ein Beispiel für eine solche Umordnung findet sich ebenfalls in [I32-1995, 6.1.1.11]:

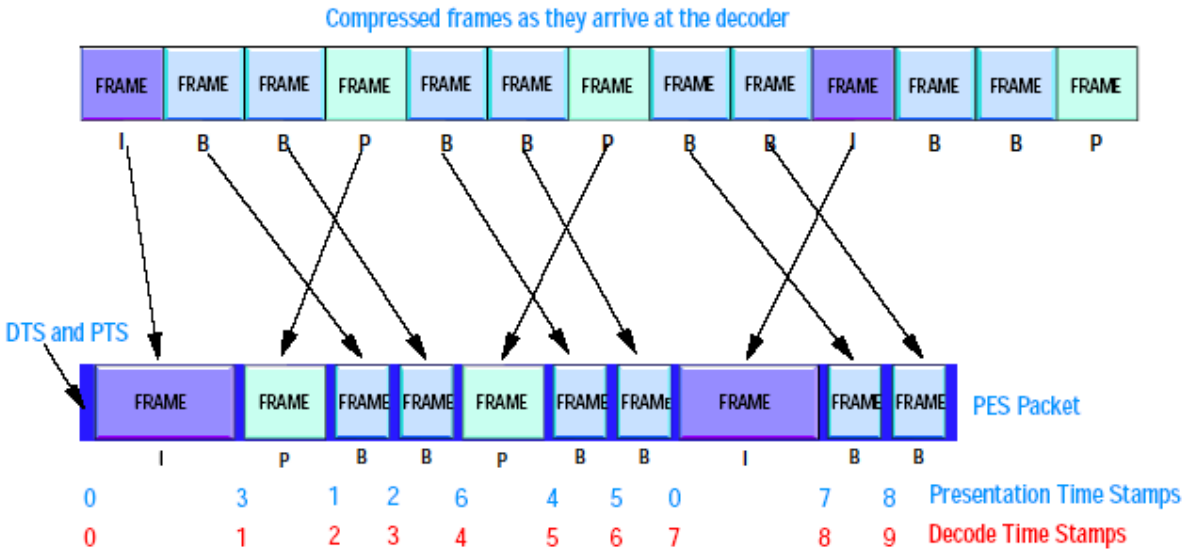


Diagramm 6: Frame-Reordering in Empfangs- (oben) und Wiedergabereihenfolge (unten). Aus [HPM-2006].

(Der Encoder hält sich gewöhnlich an einen gewissen Rhythmus, was die Verteilung von I-, P- und B-Frames angeht - wenn ein Szenenwechsel ansteht und die Bandbreite es zulässt, können I-Frames außerhalb der Reihe eingefügt werden)

Aufgabe des Decoders ist es, die Reihenfolge der Bilder wiederherzustellen. Dies kann über Presentation Time Stamps (PTS) und Decode Time Stamps (DTS) in Packetized Elementary Stream (PES) - Paketen geschehen (was bei DVB der Fall ist), siehe Diagramm 6, [HPM-2006], Folie 11f. Dies ist jedoch kein fester Bestandteil von MPEG-2 und kommt auch nicht in der Referenzdokumentation [I32-1996] vor. Dort stehen ein paar kurze Regeln, welche den auszugebenden Frame von dem gegenwärtig decodierten Frame abhängig machen. [I32-1995, 6.1.1.11] Im Code des Referenzframeworks sind die Regeln besser erklärt. [M2S-1996, getpic.c, Funktion frame_reorder (ab Zeile 754). Die Regeln sind nicht kommentiert, aber beim Code selbst wurde auf gute Lesbarkeit geachtet.]

Der MPEG-2-Referenzdecoder ignoriert PES-Pakete jedenfalls, der H.264-Decoder beendet die Decodierung mit einer Fehlermeldung, wenn PES-Pakete im Strom verbleiben.

2.3 Bild-Synchronisierung

Anders als beim analogen Fernsehen wird beim digitalen kein "Bildfang" benötigt. Jedem Einzelbild geht ein sogenannter Picture_header voraus, der mit dem Synchronisationscode 00 00 01 00 (hexadezimal) beginnt. Von dort an geht der Decoder seinen Weg über die verschiedenen Zusatzinformationen (darunter picture_coding_type, welches angibt, um welchen Typ Frame es sich handelt), zum Slice Header (die erste Slice nach dem Picture_header sollte 01 sein), über weitere Zusatzinformationen (z.B. optionale benutzerdefinierte Quantisierungsmatritzen) zu den Macroblocks.
In einem I-Frame hat die erste Slice tatsächlich 00 00 01 01 (hex) als Synchronisationscode. Der erste Macroblock in jeder Slice (auch in P- und B-Frames) hat die Nummer 1. Diese Nummer ist nicht explizit angegeben, sondern wird implizit über macroblock_address_increment hochgezählt, welches typischerweise 1 ist, was auch aus der kürzesten Huffman-Codierung in [I32-1995, Tabelle B-1] hervorgeht. Macroblock 1 in Slice 1 befindet sich immer in der linken oberen Bildschirmecke. Macroblocks mit aufsteigender Nummerierung in derselben Slice bewegen sich nach rechts, eine höhere Slice-Nummer entspricht einer Bildinformation weiter unten im Bild. Die Position jedes Macroblocks im Bild ist also festgelegt. Lediglich die Bildauflösung muss mit dem Anzeigegerät verhandelt werden. Die Informationen zur Bildauflösung (Höhe, Breite, Seitenverhältnis (4:3, 16:9...)) sind Bestandteil des Sequence Headers (00 00 01 B5 hexadezimal), welcher zumindest bei DVB-Streams jedem I-Frame beigelegt ist.[5] Da es allgemeiner Gebrauch ist, dass der Slice_start_code die absolute Y-Position der Slice enthält, ist es möglich, ohne Kenntnis der Auflösung Bildinhalte unterschiedlicher Auflösung erkennbar darzustellen. Bei analogem Fernsehen resultierte ein solcher Versuch meistens in einem unerkennbaren Bild. (Bild 1)

[5] Bestätigt durch [PDM-2013], in MPEG2-Dateien für die Speicherung auf Datenträgern gibt es nur einen Sequence Header am Anfang der Datei

Bild 1: Inkompatibler Auflösungsmodus bei einem analogen Monitor

Es ist Aufgabe des Ausgabegerätes, die digitalen Bildinformationen auf seinen eigenen Bildschirm (ggf. durch Skalierung) einzupassen. Ist das Ausgabegerät analog, muss das digitale Empfangsgerät ein gültiges analoges Signal generieren und das digitale Bild in die physischen Gegebenheiten des erzeugten analogen Signals einpassen. Synchronisation auf der digitalen Seite geschieht über den Sequence Header. Die Darstellung eines gültigen Videostreams kann somit erst ab einem I-Frame beginnen. [vgl. I32-1995, Kapitel 6.2 für den gesamten Absatz]

2.4 Videostream-Decodierung

Kapitel 6.2 von [I32-1995] enthält eine Definition des Aufbaus eines MPEG-2 Videostroms in Form eines C-ähnlichen Pseudocodes. Für einen gültigen Videostrom gemäß dieser Spezifikation wird zuerst ein sequence_header (00 00 01 B3 hex) benötigt, welcher Informationen wie Höhe und Breite des Bildes in Pixeln, das Seitenverhältnis, Bildwiederholrate und einige weitere Parameter enthält. Anschließend folgt - zumindest bei MPEG2 - die sequence_extension, welche den Startcode 00 00 01 B5 (hex) hat und weitere Informationen über den Bildinhalt enthält, z.B. das chroma_format, oder ob das vorliegende Bildmaterial progressive oder interlaced (Zeilensprungverfahren) ist. In der Regel folgt - wenn der Videostrom mit einem I-Frame beginnt - noch ein group_of_pictures_header (00 00 01 B8 hex).
Die eigentlichen Bilddaten - vom picture_header durch sämtliche Slices und Macroblocks - laufen anschließend in einer Schleife mit folgender Struktur ab:

```
(endlos) //endet mit sequence_end_code 00 00 01 B7 (hex), der aber bei Streams
nicht vorkommt
{
	sequence_header, extension, GOP;
	(Für jedes Einzelbild bis zum nächsten I-Frame)
	{
		picture_header, coding_extensions;
		(Für jede Slice = 0 bis (Y-Auflösung/16))
		{
			Weitere Parameter;
			(Für jeden Macroblock = 0 bis (X-Auflösung/16))
			{
				Delta_Macroblock_Adresse; (wenn >1, werden so viele
Macroblocks übersprungen, wie in diesem Wert stehen)
				(optional) Weitere Parameter;
				(Für jeden Block=1 bis chroma_format:{6,8,12})
				{
					Block: (Für jeden Koeffizient von 0-63) {lies Daten
aus Strom}
				}
			}
		}
	}
}
```
[I32-1995, Kap. 6.2.2ff]

2.4.1 Transport Stream

Ein Transport Stream (TS) kapselt eine Videoübertragung, vergleichbar wie ein TCP/IP-Stream eine Internet-Datenübertragung kapselt. Es ist eine paketbasierte Übertragung, Pakete sind adressiert. Im Gegensatz zum Internet mit seinen 2^32 IP V4 (bzw. 2^48 bei IP V6) Adressen, gibt es bei einem Transport Stream nur 2^13 (8192) verschiedene Adressen. Ein weiterer Unterschied ist die Paketlänge, die bei TS fest auf 188 Byte eingestellt ist, was eine Synchronisationsmöglichkeit auf physischer Ebene darstellt. [ADB-2013]

2.4.1.1 Physische Übertragung und Synchronisierung

Bei der Übertragung der Pakete wird eine Entropiecodierung zwecks Fehlererkennung und -korrektur durchgeführt, dadurch ist ein TS-Paket bei der Übertragung 204 Byte groß (In Europa. Andere Länder verwenden andere Systeme, z.B. ATSC, wo die Pakete 208 Byte groß sind) [WTS-2013]. Entropiedecodierung könnte durchaus zu den langen Umschaltzeiten beitragen, da die Synchronisation mit dem Byte- und dem Paketanfang erschwert wird (Wo fängt ein Byte an, wenn man mitten in eine Übertragung hineinschaltet?), zumal ein digital gesendeter, analog empfangener Wert (möglicherweise durch Störungen verändert) wieder einem digitalen Wert zugewiesen werden muss. Hinzu kommt, dass bei den meisten Verfahren ein solcher Wert nicht 8 Bits, sondern 2, 4 oder 6 Bits codiert (bei 256-QAM

werden 8 Bits pro Wert codiert. Das Verfahren ist aber nur im digitalen Kabelnetz und im Satellitennetz (DVB-S2) gebräuchlich. [MKB-2013]). Weiterführende Informationen zum Empfang, insbesondere dem digitaler Signale, auf der deutschen Wikipediaseite zur Quadraturamplitudenmodulation [WQA-2013], technische Informationen zu gebräuchlichen physikalischen Übertragungsarten von DVB-T/C/S/S2 finden sich in [FFG-2010].
Die physische Übertragung und Entropiedecodierung ist nicht Bestandteil dieser Arbeit, obwohl auch hier Handlungsbedarf zu bestehen scheint. Der DVB-T Leitfaden schreibt auf Seite 7: "Die Synchronisation dauert [unter ungünstigen Empfangsbedingungen] jedoch oft einige Sekunden" [TFD-2007]. Erwähnenswert ist jedoch, dass erstens die Entropiedecodierung im Demodulator, unabhängig vom restlichen Decodierungsprozess stattfindet und zweitens hierbei das Transport-Error-Flag im TS-Paket im Fehlerfalle gesetzt wird. Beim Philips TU/CU 1216 DVB-T/DVB-C Tuner ist dies beispielsweise der Fall [PCU-2004]. Eine Untersuchung ergab einen Wert von durchschnittlich 0,35 Sekunden für die Frequenzumschaltung, bis erneut ein gültiges Signal geliefert wurde.[APT-2013i]

2.4.1.2 Aufbau eines Pakets

Ein Paket beginnt mit einem Synchronisierungsbyte (47 hex). Da dieser Wert auch mitten im Paket vorkommen kann, wird die Synchronisierung weiter erschwert, jedoch darf dieser Wert außerhalb des Sync-Bytes nie mehr als viermal an derselben Stelle stehen [I31-2000, S.149]. Die nächsten 2 Byte beinhalten in ihren unteren 13 Bits die PID, also die Adresse des Pakets, und in ihren oberen 3 Bits gibt es folgende Flags:

- Transport-Error (wird vom Demodulator gesetzt, unabhängig vom Rest des Systems)
- Payload Unit Start Indicator (1 bedeutet, dass in diesem Block der Anfang einer Nutzlast ist, z.B. ein Picture Header, was für den Decoder interessant ist, da er bei der Erstsynchronisation nur Pakete mit diesem Flag auswerten muss.)
- Transport Priority (1 bedeutet: Höhere Prioirtät)

Das 4. Byte enthält 3 Felder: Scrambling Control (für diese Arbeit bedeutungslos), Adaptation Field Control (Zeigt, ob ein Adaptation-Field existiert, dieses enthält für die Videodecodierung keine Daten und muss übersprungen werden, wenn in diesem Feld angegeben ist, dass ein Adaptation Field existiert), und einen 4-Bit Continuity Counter, der z.B. genutzt werden kann, um zu erkennen, ob Pakete fehlen. [ADB-2013, WTS-2013]

2.4.1.3 Arten von Paketen

Die Art eines Pakets wird von dessen PID bestimmt oder vom Adaptation Field Control-Feld, wenn dieses angibt, dass außer eines Adaptation Fields keine weiteren Daten im Paket sind. Die PID ist im Wesentlichen sehr variabel, es gibt meines Wissens nur drei vorgegebene PIDs, nämlich die 0, welche die Program Association Table (PAT, siehe nächster Abschnitt) enthält, die 1, welche die Conditional Access Table (CAT, auf diese wird nicht näher eingegangen) enthält, und die 8191, welche auf ein Stuffing-Paket hinweist. Die restlichen PIDs sind von den Werten in diesen beiden Tabellen abhängig. [HPM-2006, Folie 14]

Program Association Table (PAT):
Die PAT enthält weiterführende Daten über die im Transport Stream enthaltenen Programme und muss daher das erste sein, wonach ein Receiver beim Sendersuchlauf suchen muss. Diese Daten enthalten die Anzahl der Sender sowie die PIDs von deren Program Map Tables (PMT, siehe nächster Abschnitt). Ein Receiver beim Sendersuchlauf muss nun nach Paketen mit diesen PIDs Ausschau halten. [HPM-2006, Folie 14, I31-2000 Table 2-25]

Program Map Table (PMT):
In der PMT steht, aus welchen Datenströmen sich ein Fernsehprogramm zusammensetzt (z.B. Audio und Video), und welche PIDs diese Ströme haben. Mit diesen Informationen ist ein Receiver nun in der Lage, sich die Pakete mit den Bild- und Toninformationen aus dem Strom "herauszupicken". [HPM-2006, Folie 14, I31-2000 Table 2-28]
Dieser Ablauf ist auch in ORDERTS [AOT-2013i] und TS2M2V [ATS-2013i] implementiert.

Datenpaket:
Die PID eines Datenpakets ist abhängig von den Daten der PMT, ebenso der Typ der Payload. Lediglich ein Adaptation Field kann auftreten, der Rest des Pakets ist reine Nutzlast.

Zusammenfassung:

PAT (PID 0) {

PID PMT[1],

PID PMT[2],

...

}, PMT[1] {

PID Video 1,

PID Audio 1a,

PID Audio 1b (z.B. für Fremdsprache),

PID Untertiteldaten...

},

PMT[2]{...} ...

Ein ausführliches Beispiel anhand echter Pakete folgt nun.

2.4.2 Ausführliches Beispiel anhand echter Pakete (test4.ts)

Dieses Beispiel entstammt der Datei test4.ts, welche ich selbst über eine DVB-T Decoder-Karte unter Verwendung des VLC Media Players mit der Einstellung, den Stream ohne Be- oder Verarbeitung zu dumpen, erstellt habe. Um die Ergebnisse nachzuvollziehen, bediene man sich des DVB-Analyzers von Peter Daniel [PDM-2013].

2.4.2.1 Transport-Stream-Seite

1.: Suche nach der PAT (gefunden in Paket 1807).

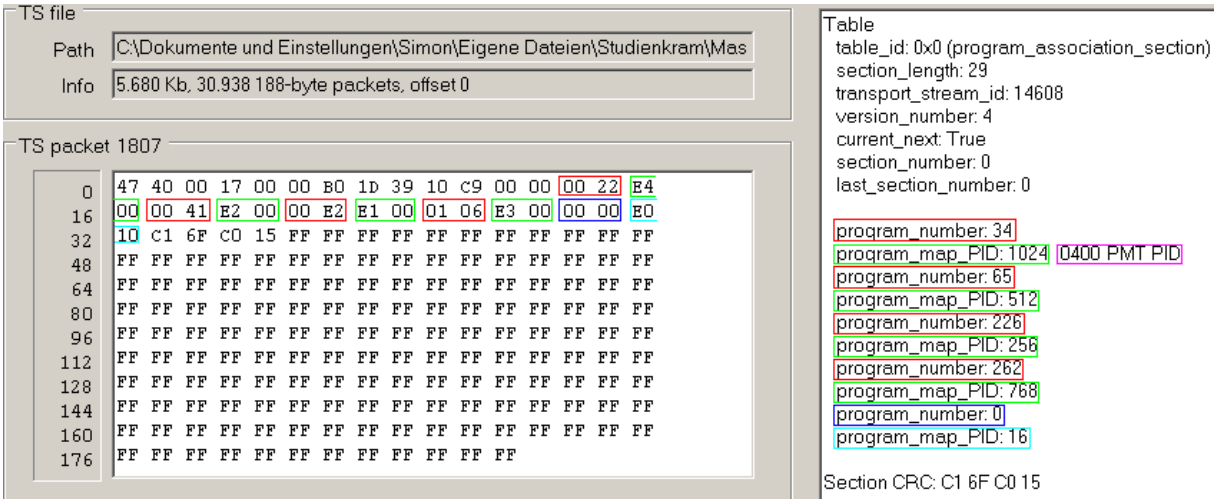


Bild 2: Program Association Table (PAT). Programmnummern wurden farbig markiert

2.: Suche nach der PMT für Programm 1 (PID:1024) (gefunden in Paket 2229 (und 34)).

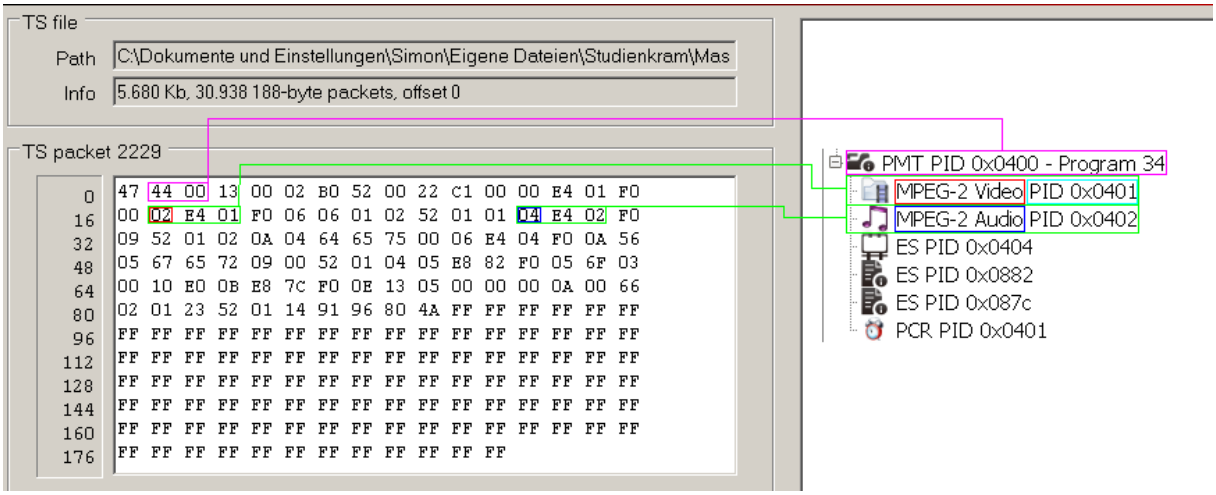


Bild 3: Program Management Table (PMT) von Programm 34. Jedes Programm in der PAT hat eine eigene PMT. (Quellen: [PDM-2013] und [ODA-2009])

Der Aufbau von PAT und PMT findet sich in ISO/IEC 13818-1 in Tabellen 2-25 und 2-28 [I31-2000].

3.: Suche nach dem ersten I-Frame im Videostream von Programm 1 (gefunden in Paket 3327) bzw. erstelle Videostream aus allen Paketen mit der PID dieses Programms.

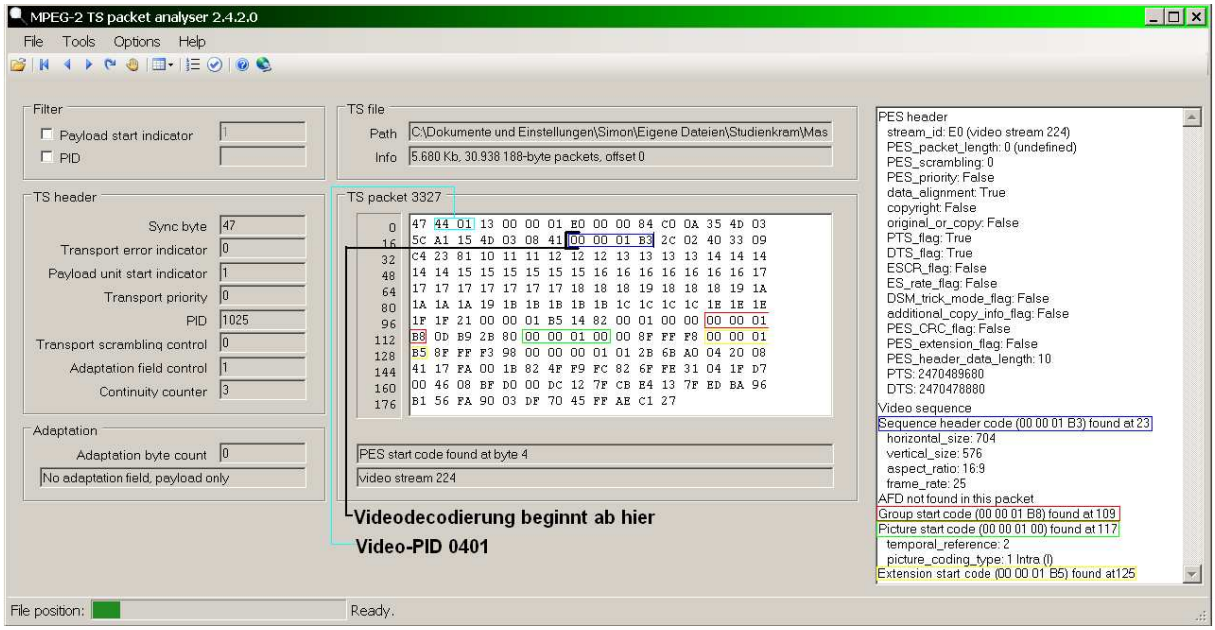


Bild 4: I-Frame-Header in Transport Stream Packet

Schritt 1 und 2 muss der Decoder lediglich bei der ersten Inbetriebnahme oder bei einem größeren Standortwechsel durchgeführt werden. Schritt 2 muss für die Anzahl der Programme in der PAT wiederholt werden.

2.4.2.2 Videostream-Seite

Für diese Analyse wird weiterhin Bild 4 verwendet. Hauptaugenmerk ist hierbei der Raw-Hex-Code in der Bildmitte.
Beginnen wir die Decodierung unmittelbar hinter dem Sequence Header (00 00 01 B3 hex) in Byte 23 und schlagen Kapitel 6.2.2.1 von ISO/IEC 13818-2 [I32-1995] auf:
Lesezeiger: Byte 27.

Feld	Länge (Bit)	Wert	Lesezeiger	Anmerkungen
horizontal_size_value	12	0x2C0 = 704d	28.4	Nachkommastellen des Lesezeigers zeigen Bitnummer an, Zählen beginnt bei 0
vertical_size_value	12	0x240 = 576d	30	
aspect_ratio_information	4	3 = 16:9	30.4	Tabelle 6-3
frame_rate_code	4	3 = 25fps	31	Tabelle 6-4
bit_rate_value	18	0x2710	33.2	0x09C4(2) 0b00001001 11000100 00 = 10000d
marker_bit	1	1	33.3	(immer 1)
vbv_buffer_size_value	10	0x70	34.5	0b0(2) 0x38 0b0(1)
constrained_parameters	1	0	34.6	
load_intra_quantiser_matrix	1	0	34.7	
load_non_intra_quantiser_matrix	1	1	35	Hiernach folgen 64 Byte Non-Intra-Quantiser-Matrix; Lesezeiger jetzt 99

Tabelle 2: Decodierung des Sequence Headers von test4.ts

An der Lesezeigerposition 99 finden wir den Start Code 00 00 01 B5 (hex), also den Extension Start Code, vor. Dies verwundert nicht weiter, da das Fehlen dieses Codes auf einen MPEG-1 Videostrom hinweisen würde.
Die zur Decodierung der Extension notwendigen Daten finden sich in Kapitel 6.2.2.3:

Feld	Länge	Wert	Lesezeiger	Anmerkungen
start_code_identifier	4	1	103.4	Sequence Extension ID, Tabelle 6-2
profile_and_level	8	48 = MP@ML	104.4	Appendix 8, Tabelle 8-2 und 8-3
progressive_sequence	1	0 = interlaced	104.5	
chroma_format	2	1 = 4:2:0	104.7	Tabelle 6-5
horizontal_size_ext.	2	00	105.1	benötigt für Ultra HDTV
vertical_size_ext.	2	00	105.3	
bit_rate_ext.	12	0	106.7	0b0 0000 0000 000
marker_bit	1	1	107	(immer 1)
vbv_buffer_size_ext.	8	0	108	
low_delay	1	0	108.1	Frame-Reordering findet statt
frame_rate_extension_n	2	0	108.3	
frame_rate_extension_d	5	0	109	

Tabelle 3: Decodierung der Sequence Extension

Die letzten 8 Bits sind extension_and_user_data(mode 0), welche wir in Kapitel 6.2.2.2.1 von [I32-1995] nachschlagen, hier wird jedoch ein 32 Bit Startcode erwartet, und der nächste Startcode ist der Group Start Code 00 00 01 B8 (hex), dessen Decodierung anhand von Kapitel 6.2.2.6 der Dokumentation durchgeführt werden kann.

Feld	Länge	Wert	Lesezeiger	Anmerkungen
time_code	25	1A5257	113.1	0b0000 1101 1011 1001 0010 1011 1
closed_gop	1	0	116.2	
broken_link	1	0	116.3	

Tabelle 4: Decodierung des Group of Pictures Header

Die Referenz befiehlt nun, auf den nächsten Startcode vorzuspulen. Dieser Startcode ist ein Picture Header.

Feld	Länge	Wert	Lesezeiger	Anmerkungen
temporal_reference	10	2	122.2	vom Decoder ignoriert
picture_coding_type	3	001 = I-Frame	122.5	Tabelle 6-12
vbv_delay	16	0xFFFF	124.5	vom Decoder ignoriert
extra_bit_picture	1	0	124.6	Wenn 1, folgt extra_information_picture

Tabelle 5: Decodierung des Picture Header

Als nächster Startcode wird eine picture_coding_extension erwartet (00 00 01 B5 hex). Diese findet sich in Kapitel 6.2.3.1. [I32-1995]

Feld	Länge	Wert	Lesezeiger	Anmerkungen
extension_start_code	4	8	129.4	Picture Coding Extension ID, Tabelle 6-2
f_code[0][0]	4	0xF	130	
f_code[0][1]	4	0xF	130.4	
f_code[1][0]	4	0xF	131	
f_code[1][1]	4	0xF	131.4	
intra_dc_precision	2	0 = 8 Bits	131.6	Tabelle 6-13
picture_structure	2	3 = Frame	132	Tabelle 6-14
top_field_first	1	1	132.1	
frame_pred_frame_dct	1	0	132.2	
concealment_motion_vect	1	0	132.3	
q_scale_type	1	1	132.4	
intra_vlc_format	1	1	132.5	
alternate_scan	1	0	132.6	
repeat_first_field	1	0	132.7	
chroma_420_type	1	0	133	

progressive_frame	1	0	133.1	
composite_display_flag	1	0	133.2	

Tabelle 6: Decodierung der Picture Coding Extension

Die restlichen 6 Bit bis zum nächsten Startcode, welcher der Slice Header der ersten Slice ist, enthalten einige weitere Bits, die in der Dokumentation nicht weiter erläutert sind (vermutlich Byte-alignment Stopfbits)
Um nun die Slice decodieren zu können, bedienen wir uns der Informationen aus Kapitel 6.2.4.
Hierzu müssen wir noch wissen, ob scalable_mode=="data partitioning" ist. vertical_size ist in unserem Falle ja kleiner als 2800, daher wissen wir, dass slice_vertical_position_extension nicht vorhanden ist. Scalable_mode bekommen wir aus der Tabelle Sequence_scalable_extension (Kapitel 6.2.2.5, [I32-1995]). Diese Tabelle wurde jedoch nicht übermittelt, also können wir davon ausgehen, dass kein priority_breakpoint vorhanden ist. Eine Konsultation der Profiles and Levels [I32-1995, Anhang 8] gibt ebenfalls Aufschluss über das (nicht)Vorhandensein einer sequence_scalable_extension. Für den Wert 48 (4 für Main Profile und 8 für Main Layer) gibt Tabelle 8-5 an, dass keine sequence_scalable_extension vorhanden ist.

Feld	Länge	Wert	Lesezeiger	Anmerkungen
quantiser_scale_code	5	5	138.5	Nächstes Bit==0 -> keine intra_slice
extra_bit_slice	1	0	138.6	

Tabelle 7: Decodierung der Slice-Extrainformationen

Jetzt können wir den ersten Macroblock decodieren. Dies bedeutet, dass nun variable length codes kommen, d.h. dass die folgenden Werte unterschiedliche Längen annehmen können. Ein gewisses Verständnis von Zustandsautomaten wird für die Decodierung vorausgesetzt. Das nächste Feld ist macroblock_address_increment, der Lesezeiger steht immer noch auf 138.6, das nächste Bit im Datenstrom ist 1, also kann auch kein macroblock_escape vorhanden sein. In Tabelle B-2 steht nun, dass bei einer 1 mit dem Decodieren von macroblock_address_increment aufgehört werden kann und dass der Wert 1 ist, d.h. es werden keine Macroblocks übersprungen. Nun folgt macroblock_modes, wofür wir Kapitel 6.2.5.1 von [I32-1995] brauchen.

Feld	Länge	Wert	Lesezeiger	Anmerkungen
Macroblock_type	1	1 = Intra	139	Tabelle B-2.

Tabelle 8: Beginn der Decodierung von Macroblock modes. Hinweis: macroblock_intra nimmt hierbei automatisch den Wert 1 an.

Die nächsten beiden if-Abfragen nehmen den Wert false an. Die letzte if-Abfrage in macroblock_modes bedarf jedoch gründlicherer Untersuchung:

Feld	Wert	Gefunden in
picture_structure=="Frame picture"	true	picture_coding_extension
frame_pred_frame_dct==0	true	picture_coding_extension
macroblock_intra	true	Tabelle B-2
macroblock_pattern	false	Tabelle B-2

Tabelle 9: Abschluss der Decodierung von Macroblock modes

Da die letzten beiden Werte ODER-verknüpft sind, evaluiert die Abfrage auf wahr. Entsprechend wird dct_type = 0 eingelesen und der Lesezeiger auf 139.1 gesetzt und wir gehen zurück in Kapitel 6.2.5 der Referenzdokumentation. Keine der weiteren if-Abfragen in Macroblock evaluiert auf wahr, also können wir nun mit der Block-Decodierung beginnen, was bedeutet, dass die nun folgenden Daten Huffman-codierte DCT-Koeffizienten darstellen, was bedeutet, dass wir nun, wenn wir wollten, mittels der vorgegebenen Huffman-Tabelle und einem inversen DCT-Algorithmus tatsächlich Bildinformationen erhalten würden.

[ATF-2013i] enthält eine vergleichbare Analyse der Datei sample.ts www.pjdaniel.org.uk/mpeg/downloads/mux1-cp.zip] zwecks Analyse der Wiederherstellbarkeit eines I-Frames mit verpasstem Header.

3. Analyse – MPEG-2

Spezifische Ursachen für die Umschaltzeiten:
Wie in den Grundlagen beschrieben, muss beim Kanalwechsel (selbe oder unterschiedliche Frequenz) ein I-Frame-Header, bei MPEG-2 verbunden mit einem Sequence-Header, bei H.264 (siehe Kapitel 6.3.2) verbunden mit einem NAL-Unit-Delimiter, einem Sequence- und zwei Picture Parameter Sets empfangen werden, bevor mit der Decodierung begonnen werden kann. Des weiteren muss erst der komplette I-Frame decodiert werden, bevor mit der Darstellung begonnen werden kann.

3.1 Timings

Betrachten wir sample.ts, so stellen wir fest, dass zwischen zwei I-Frames knapp 6000 Pakete liegen und es durchschnittlich 1630 Pakete dauert[6] , bis der I-Frame komplett übertragen wurde [AFT-2013i], bei einer Datenrate von 18 MBit/s (Faktor 1000000, nicht 1048576), was einer Paketlänge von 835,556 Mikrosekunden (µs) entspricht - Stichproben anhand des Abstands zweier I-Frames in Paketen lieferten einen Wert von 864,08 µs, in dieser Arbeit wird jedoch mit 835,556µs gearbeitet, dieser Wert ist genauer - und 0,6375 Sekunden, bevor der Decoder das erste gültige Bild ausgibt. Dies entspricht der Worst-Case Umschaltzeit auf dem Papier. Hierbei ist nicht berücksichtigt, dass evtl. Frame-Reordering stattgefunden haben kann, was bedeutet, dass der Decoder die Transmission weiterer Frames abwarten muss. Wie in Kapitel 3.2 (und [HPM-2006]) beschrieben, wäre IPBBPBB... die optimale Reihenfolge zur Übertragung. Diese traf ich jedoch bei der Analyse [mit ORDERTS.EXE] nicht an, stattdessen die klassische IBBPBBP...PBBI. Ich vermutete zunächst, dass B-Frames keine vorwärtsgerichteten Prädiktionen enthielten, die Trace-Ausgabe des MPEG2-Decoder-Frameworks [MDO-2013i] belehrte mich jedoch eines Besseren. Entsprechend müsste der Decoder bis einschließlich auf den nächsten P-Frame warten, was ungefähr weitere 1093 Blocks oder 0,09 Sekunden dauert. Eine genaue Analyse des Frame-Reorderings bei der Transmission findet sich weiter unten.
Die Berechnung der Paketlängen lässt sich recht zuverlässig nach folgender Formel ermitteln:

[6] 1632 Pakete bis zum nächsten Frame-Header. Ende der Transmission ist 1 Paket vorher, wobei sich Pakete von mehreren anderen Sendern dazwischen befinden können

$$Pakete/Sekunde = \frac{Paketnummer\_letzter\_Frame - Paketnummer\_erster\_Frame}{Anzahl\_Frame\_Header - 1} * Frame\_Rate$$

Formel 1: Ermittlung der Anzahl der Pakete pro Sekunde.

Der Kehrwert dieser Formel entspricht der Länge der Pakete.

Die Anzahl der Frame-Header (=Anzahl der Frames) erhält man empirisch durch Nachzählen, was automatisiert von Orderts.exe [AOT-2013i] erledigt werden kann[7]. Da der letzte Frame nicht vollständig vorliegt, muss man von dieser Zahl eins abziehen. Die Paketnummern kann man mit dem MPEG-2 TS analyzer von Peter Daniel [PDM-2013] ermitteln, indem man "Payload start indicator" setzt und die gewünschte PID einträgt. Der erste Frame wird vom Anfang der Datei aus vorwärts gesucht, der letzte Frame vom Ende der Datei aus rückwärts. Um bei [PDM-2013] ans Ende der Datei zu kommen, wählt man bei "Go to Packet" das letzte oder vorletzte Paket in der Datei. Der MPEG-2 TS packet analyzer zeigt die Anzahl der Pakete an.

Die hier ermittelten Werte entstammen sample.ts und sind beispielhaft, jedoch nicht ohne Aussagekraft. Die Anzahl der Pakete pro Frame kann zwischen unterschiedlichen Programmen differieren, da unterschiedliche Bitraten in einem Datenstrom mit konstanter Bitrate möglich sind. Dies hat jedoch auch einen relativen Unterschied in der Häufigkeit des Auftretens von Paketen der unterschiedlichen Programme als Ergebnis, sodass sich diese Unterschiede aufheben. Gleiches gilt für Datenströme mit insgesamt höherer (oder niedrigerer) Bitrate. Ein Frame wird bei höherer Bitrate mehr Pakete belegen; die Zeit, die ein einzelnes Paket benötigt, reduziert sich jedoch aufgrund der höheren Bandbreite. In allen Fällen muss die konstante Ablaufgeschwindigkeit des Videostroms auf dem Zielgerät gewährleistet bleiben, welche auch das Fallback-Kriterium für Zeitmessung in Streams mit unbekannter Bitrate darstellt. Da die einzelnen Programme in einem Strom round-robin (der Reihe nach) bedient werden und selten 2 Pakete hintereinander zum selben Programm gehören (außer bei HD-Videoströmen) (siehe auch Bild 81 und 82 in Kapitel 9.1.4.1), trifft dies in eingeschränkter Form auch auf HD-Programme und Streams mit mehr als 4 Programmen zu, allerdings nicht auf HD-Programme selbst, da diese in der Regel eine höhere Bildwiederholrate haben.

[7] Vorher prüfen, wie viele Pakete in der Datei sind, da am Ende der Datei nach Analyse Y/N gefragt wird und jede andere Taste als Y als Nein behandelt wird.

3.2 Frame-Reordering

Die tatsächliche Reihenfolge der Frames wurde wie folgt ermittelt:
Frame Nr. steht für die Reihenfolge bei der Wiedergabe, die Adresse entspricht der hexadezimalen Adresse des Framebeginns in der entpackten Videodatei, aus welcher sich die Sendereihenfolge ableitet. Der Empfang startet beim I-Frame.

Frame Nr.	Adresse (hex)	Sendereihenfolge
0 B	0x31371	1
1 B	0x56715	2
2 I	0x0	0
3 B	0xa5fb2	4
4 B	0xd6be2	5
5 P	0x7c0f7	3

Diese Daten entstammen CodecVisa [CV-2013] unter Verwendung von test4.ts, entpackt von PID 1025.

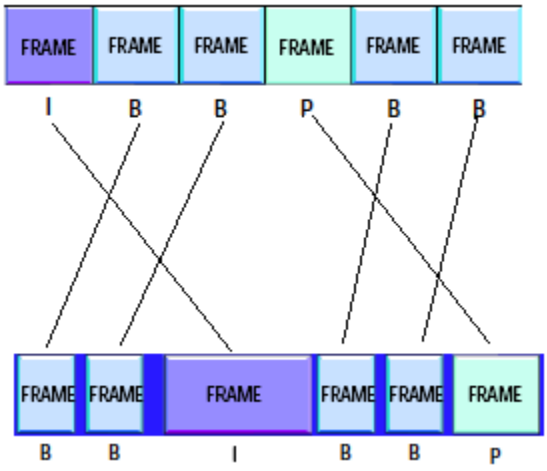


Diagramm 7: Ermittelte Frame-Reihenfolge in MPEG-2 Videoströmen.

Diese Reihenfolge wurde auch in den aus sample.ts entpackten Videoströmen sowie in MPEG-2 komprimierten HD-Videoströmen vorgefunden.
Die für sample.ts (über alle PIDs, je nur 1 I-Frame) ermittelten Durchschnittswerte für die Zeit, die vom Empfang des ersten I-Frames (vom I-Frame-Header) bis zum Ende des 4. empfangenen Frames (der dem frühesten Beginn einer Bewegtwiedergabe entspricht) vergeht, beträgt 3179 Blocks, was 0,265 Sekunden entspricht. Diese Zeit muss also abzüglich der 0,146 Sekunden für die Transmission des I-Frames auf die 0,6375 Sekunden des ermittelten Worst-Case aufaddiert werden, wenn man Bewegtwiedergabe abwartet. Wie einer der getesteten Decoder (Sky DVB-S2) im selben Bouquet die Bestzeit auf Papier unterbieten konnte, ist mir unklar. Dies geschah jedoch nur einmal, könnte also ein Messfehler sein. Der Decoder hat in zwei Fällen 0,36 Sekunden gebraucht, sonst erheblich länger (~1/2s). Eine andere Möglichkeit wäre, dass im empfangenen Bouquet höhere I-Frame-Raten vorliegen, oder der Encoder auf dynamische I-Frames (also I-Frames bei Szenenwechsel anstelle des monotonen IBBPBBPB...) eingestellt ist.

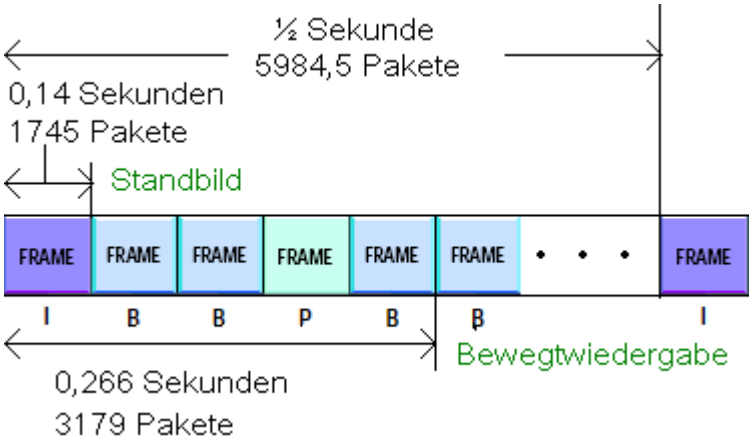


Diagramm 8: Ermittelte Durchschnittszeiten für die Übertragungsdauer relevanter Streambestandteile. Werte aus [AFT-2013i].

3.3 Frequenzwechsel und Neusynchronisation

Beim Wechsel zwischen zwei Programmen, die auf unterschiedlichen Frequenzen liegen, muss der Tuner einen Frequenzwechsel und der Demodulator eine Neusynchronisation durchführen.

Auf beides wird nicht im Detail eingegangen, aber die Vorgänge sollten bekannt sein. Die Zeit, die ein Frequenzwechsel erfordert, ist von der Güte und Reaktionsgeschwindigkeit des Schwingkreises im Tuner abhängig und lässt sich ungefähr in folgende Phasen einteilen:

- Wechsel von Ist-Frequenz auf Soll-Frequenz
- Phasenabgleich mit dem Signal
- Feinabstimmung.

Die letzten beiden Phasen können vertauscht und/oder beliebig oft ausgeführt werden. In der ersten Phase ist die Nähe der gespeicherten Soll-Frequenz zur tatsächlichen Frequenz des Senders von entscheidender Bedeutung. Eine große Nähe bedeutet einen geringen Aufwand bei der Feinabstimmung sowie eine Signalqualität, die bereits ohne Feinabstimmung decodiert werden kann, was aber einen erhöhten Speicheraufwand für die Frequenzspeichertabelle darstellt. Eine geringe Nähe ermöglicht weniger Speicheraufwand, bedeutet jedoch auch längere Feinabstimmungsphasen. Ein guter Vergleich ist der Umstieg von analogen Senderspeichern zu digitalen in Analog-Fernsehern Ende der 1970er Jahre. Tuner funktionierten ab Anfang/Mitte der 70er (manche auch früher) häufig über VCOs (Voltage Controlled Oscillator, spannungsgesteuerter Oszillator). Bei analogen Senderspeichern wird ein Array aus Potentiometern eingesetzt, deren Spannungswerte über einen Demultiplexer an

den Tuner geleitet wurden. Üblich war hier eine Spannung von 32V, die über die Potis von 1-32V geregelt wurde und entsprechend an den VCO im Tuner geleitet wurde.[8] Der Vorteil hier ist, dass der Benutzer die Frequenz exakt einstellen kann, und automatische Feinabstimmungen nicht nötig (und in solchen Geräten oft nicht vorhanden) sind. Der offensichtliche Nachteil ist, dass die Anzahl der speicherbaren Sender von der Anzahl der Potis abhängt - also Platz benötigt, und ein weiterer, weniger offensichtlicher Nachteil sind Verstellungen durch Temperaturdrift. [WTD-2013] Bei diesen Geräten war die Umschaltzeit alleine von der Geschwindigkeit der Frequenzänderung im Tuner abhängig und lag häufig unter einer Zwanzigstelsekunde. [AAT-2013i]

Die ersten Fernsehgeräte mit digitalem Senderspeicher hatten jedoch aufgrund von Hardwarebeschränkungen eine geringe Genauigkeit der Speicher, sodass automatische Feinabstimmung erforderlich wurde. Eine Erhöhung der Umschaltzeiten war die Folge. Da ein guter digitaler Speicher mit einem guten Tuner schneller umschalten konnte als ein analoger Speicher mit einem mittelmäßigen Tuner, war dies hinnehmbar. (Vgl. auch SABA 1978 (Potis) mit Grundig 1989 (I^2C-Bus) in [AAT-2013i].)

Neusynchronisierung z.B. bei DVB-T, welches OFDM (Orthogonal Frequency-Division Multiplexing) verwendet [FFG-2010, Folien 22ff], ist um einiges komplizierter. Analoges Fernsehen verwendet ein frequenzmoduliertes Signal, eine Verschiebung der Referenzfrequenz beim Empfangsgerät hat hier nur eine Erhöhung/Erniedrigung des Signalpegels zur Folge, was ein helleres/dunkleres Bild bedeuten würde, was jedoch durch den festgelegten Schwarzwert in der Austastlücke kompensiert wird. [RWV-2009 S.10ff] "Aus der Ausgangsspannung des FM-Demodulators gewinnt man häufig gleichzeitig eine Regelspannung, mit der man den Oszillator des Empfängers nachführt (Automatic Frequency Control, kurz AFC), um das Signal in der Mitte des Durchlassbereiches der Filter und so die Verzerrung gering zu halten." [WFM-2013] Bei digitalem Fernsehempfang über Antenne (DVB-T) müssen Frequenz und Phase durch Ausprobieren ermittelt werden, was ein zeitraubendes Verfahren ist und die Qualitätsstufen von Digital-Tunern definiert [WOF-2013].

Bei digitalem Kabelempfang wird häufig Quadraturamplitudenmodulation (64QAM, 256QAM) eingesetzt, da hier Phasendrehungen oder -verschiebungen nicht zu erwarten sind und insbesondere 256QAM erlaubt, 8 Bits "auf einmal" (pro Symbol) zu senden, was wiederum die Synchronisationszeit reduziert. DVB-S verwendete 4QAM mit 2 bzw. 8QAM mit 3 Bits pro Symbol. [FFG-2010, Folien 19-21] DVB-S2 verwendet, wie im Kabel, 64QAM

[8] Ich habe genügend antike Fernsehgeräte zerlegt und repariert, die dies bestätigen

und 256QAM, aber die Quellen widersprechen sich teilweise und ich habe keine Möglichkeit, die tatsächlich verwendete Modulation zu ermitteln.
Bei Kabelempfang wurden Umschaltzeiten von 0,08-0,68 Sekunden mit einem Durchschnittswert von 0,35 Sekunden gemessen [APT-2013i], unter Verwendung des Philips CU1216-Tuners [PCU-2004]. Die Messung wurde durchgeführt am Data_Valid Pin des Tuners sowie an der Ausgabe des angeschlossenen Bildschirmterminals. Beide Messmethoden erzielten keine signifikanten Unterschiede, ebensowenig wurden signifikante Unterschiede bei der Abstimmung von QAM64 und QAM256 festgestellt.

4. Konzept zur Frame-Wiederherstellung

In diesem und dem nächsten Kapitel werden zwei Ansätze zur Beschleunigung der Umschaltzeiten vorgestellt. Der in diesem Kapitel vorgestellte Algorithmus sieht eine Erweiterung der Entpackung des Videostroms aus dem Transportstrom vor, kann also auf der CPU der meisten Geräte zum Laufen gebracht werden, was die Möglichkeit erlaubt, den vorgestellten Algorithmus mittels Firmware-Update auf bereits bestehende Geräte zu laden. Die Frame-Wiederherstellungsalgorithmen befinden sich implementiert und getestet im Programm TS2M2V auf der beiliegenden CD. Der im nächsten Kapitel vorgestellte Ansatz arbeitet auf den Videodaten, sollte also bevorzugt auf externen Decoderchips implementiert werden.

4.1 Zusammenfassung der Ansatzpunkte zur Frame-Wiederherstellung

Unter dem Wissen, dass Start-Codes Synchronisationspunkte darstellen, wird klar, dass eine Synchronisation bereits vor einem I-Frame-Header möglich ist. Dass jede Slice einen Start-Code hat, ermöglicht eine Synchronisation mitten im Bild. Entstammt die Slice zudem einem I-Frame, können (mit Einschränkungen) beträchtliche Mengen an Bildinformationen wiederhergestellt werden (siehe Kapitel 4.4).
I-Frames machen jedoch nur etwa 27% einer Übertragung aus, sodass Synchronisation mit Nicht-I-Frames nötig wird. In Nicht-I-Frames gibt es Intra-Macroblocks, die Informationen enthalten, welche direkt unverändert auf dem Bildschirm angezeigt werden können. Diese sind jedoch in der Regel derart spärlich, dass zusätzliche Maßnahmen notwendig sind, um ein betrachtbares Bild zu extrahieren (siehe Kapitel 5: Intra-Block-Interpolierung). Eine vollständige Rekonstruktion ist so gut wie ausgeschlossen, daran wird auch Kapitel 5 wenig ändern.

4.2 Konventionelles Vorgehen eines Decoders

Wie im Kapitel Grundlagen beschrieben, ist der früheste bekannte Synchronisationspunkt zur Darstellung eines gültigen Videostreams ein video_sequence_header (00 00 01 B5 hex).

Diesem folgen alle benötigten Informationen zur korrekten Darstellung der Bildinformation, wie Auflösung, Framerate und Seitenverhältnis, sowie ein vollständig definiertes Bild (I-Frame), das direkt dargestellt werden kann.

Verfügt ein Decoder noch nicht über ein vollständig definiertes Bild, muss er darauf warten, dass ein weiteres derartiges Bild im Strom auftaucht. Bei statischen Videoströmen (wahlfreier Zugriff, z.B. gespeicherte Dateien) ist das kein Problem, der Decoder kann einfach zum nächsten I-Frame "springen", indem er in der Datei nach der Zeichenkette "00 00 01 B5" (hex) sucht. Bei DVB werden die Daten jedoch mit konstanter Bitrate in Echtzeit übertragen, Vorwärtsspringen im Stream ist also nicht möglich, Rückwärtsspringen nur bis zum ersten empfangenen gepufferten Paket. Ein Decoder muss somit auf die Übertragung des nächsten I-Frames warten, und bis zu dessen vollständiger Übertragung ein schwarzes Bild (oder das letzte erfolgreich decodierte) anzeigen.
Die Bitrate einzelner Videoströme fluktuiert selbst bei konstanter Bitrate (siehe Diagramm 9). Encoder legen in der Regel eine Art Pufferzone an, da die gewünschte Anzahl der Pakete pro Bild sich in der Realität nicht erreichen lässt. Die nicht genutzten Pakete in der Pufferzone werden mit Füllpaketen (stuffing packets, PID 8191) aufgefüllt, siehe Diagramm 9 (dunkelgrün, oberer Bildrand). Die Gesamtbitrate jedoch ist konstant, da diese an physische Parameter gebunden ist, siehe Kapitel 3.1.

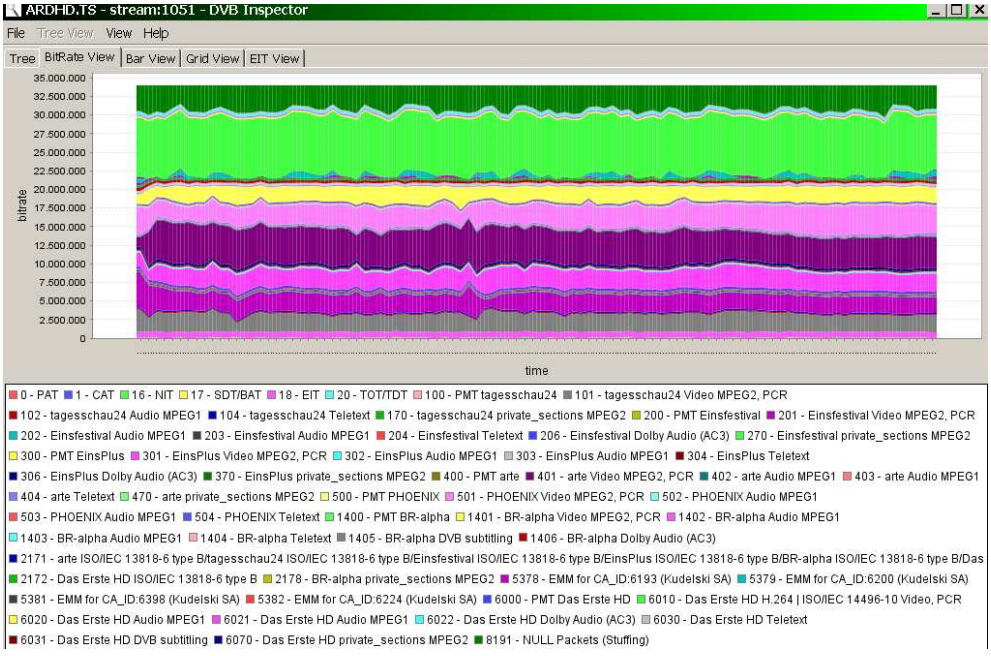


Diagramm 9: Innenleben eines Transport Stream mit 7 Programmen (ardhd.ts) Der dunkelgrüne, obere Teil des Graphen entspricht Stuffing-Paketen.

Zwischenbilder (P- und B-Frames) enthalten zwar auch einen Synchronisationspunkt, jedoch nur wenige oder keine direkt darstellbaren Bilddaten. Eine Methode, aus diesen Daten ein ästhetisch akzeptables Bild zu extrahieren, wird im nächsten Kapitel besprochen.
Prozentual gesehen machen I-Frames etwa 27%, [AFT-2013i] des gesamten Datenaufkommens aus, werden aber nur 8%[9] der Zeit auf dem Bildschirm angezeigt. Entsprechend ist die Chance etwa ¼, beim Umschalten in einen I-Frame zu geraten, der jedoch aufgrund des verpassten Headers für konventionelle Decoder nutzlos ist. Dieses Verhältnis bestätigte sich in meinen Versuchen zur Wiederherstellung, wo von acht Streams drei tatsächlich mit einem unvollständigen I-Frame begannen, was statistisch gesehen durchaus in den Bereich ¼ fallen kann.

Da jedoch, wie bereits erwähnt, bei der Zusammensetzung eines Bildes strengere Gesetzmäßigkeiten herrschen als von [I32-1995, S.26] vorgeschlagen, ist eine Teilrekonstruktion eines I-Frames recht einfach möglich. Jede Slice enthält mit dem Startcode im Slice_header einen Synchronisationspunkt, der eine absolute Y-Koordinate plus die X-Koordinate 0 enthält. Um einen gültigen I-Frame zu erhalten, muss lediglich ein gespeicherter I-Frame-Header verwendet und die fehlenden Bildinformationen bis zur ersten empfangenen Slice mit Dummy-Daten (z.B. grauen oder schwarzen Macroblocks) aufgefüllt werden, damit der Decoder ein gültiges Resultat produziert.

4.3 Frame-Wiederherstellung

In diesem Kapitel wird nun ein Algorithmus vorgestellt, der den ersten im Stream befindlichen Frame in relativ kurzer Zeit identifizieren und anschließend einen gültigen, "vollständigen" Frame daraus produzieren kann, der von einem Decoder korrekt decodiert werden kann. Eine korrekte Darstellung ist jedoch nur im Falle eines I-Frames zu erwarten.

4.3.1 Entwickelter Algorithmus

Das Programm TS2M2V [ATS-2013i] macht zunächst einen Sendersuchlauf, wie in Kapitel 2.4.1 beschrieben und bietet dem Benutzer eine Auswahl an gefundenen Videoströmen an.

[9] Auf Basis des europäischen Digitalfernsehens, bei 25 Bildern/Sekunde und 2 I-Frames/Sekunde = 2/25

Wählt der Benutzer Modus 2 oder 3, wird das Programm den ersten I-Frame des gewählten Videostroms suchen und den sequence_header, sowie sequence_extension (also alle Daten bis zum picture_header) extrahieren.
In Modus 3 wird anschließend versucht, den Typ des ersten unvollständigen Frames zu erkennen, dessen picture_header mit dieser Information ja verpasst wurde.

4.3.2 Strategien zur Frame-Erkennung

Zur Bestimmung des Frame-Typs bei verpasstem/verlorenem Header werden hier 4 Strategien vorgestellt:

Erstens: Blocklänge des gesamten Bildes von einem Picture Header zum anderen.
I-Frames sind, wie gesagt, erheblich größer als andere Frametypen. Im Beispielstrom sample.ts sind I-Frames üblicherweise über 300 Blocks groß, während P-Frames um die 80 und B-Frames um die 40 Blocks groß sind. Es gibt jedoch Ausnahmen, so sind mir I-Frames begegnet, die nur 137 Blocks klein sind, sowie P-Frames mit 255 Blocks und B-Frames mit 111 Blocks. [Quelle: [AOT-2013i] zusammen mit Sample.ts und test4.ts]
Diese Methode hat also zwei gravierende Nachteile: Zum Einen ist die Länge eines vollständigen I-Frames nicht eindeutig, zum Anderen müsste ein I-Frame vollständig empfangen worden sein, um eine Aussage treffen zu können, und genau dies ist nicht der Fall.

Zweitens: Rhythmus.
Im Stream gibt es eine bestimmte Reihenfolge, in der die Bilder auftauchen; diese ist üblicherweise IBBPBBPBBPBBI...usw. Wartet man nun die nächsten beiden Picture_headers ab und bestimmt jeweils den Frametyp, bekommt man einen guten Schluss auf den aktuellen Frametyp, nach einer einfachen Fallunterscheidung: Ist die Reihenfolge BB, kann der aktuelle Frame kein B-Frame sein.
Auch diese Methode hat wieder zwei Nachteile: Zum Einen müssen 2 Folgeframes abgepasst werden, was Zeit braucht (rund 400 Blocks im Falle von 2 B-Frames und 4 Programmen sowie Zusatzinformationen im Stream). Zum Anderen kann auch hier keine zuverlässige Unterscheidung zwischen P- und I-Frames gemacht werden. Selbstverständlich terminiert die Methode, wenn einer der abgepassten Frames ein I-Frame ist.

Ein Trost ist, dass der Decoder sehr schnell (bereits nach wenigen decodierten Macroblocks) Fehlermeldungen produziert, wenn man eine falsche Annahme zum Frametyp macht, sodass man diesen ändern und den Versuch neu starten kann.

Drittens: Länge der Slices.
Anhand des Auftretens der slice_start_codes kann man die Länge einer Slice bestimmen. Dies geht erheblich schneller und erheblich zuverlässiger als die beiden anderen Methoden. In einem B-Frame benötigt eine Slice üblicherweise 1-4 Blocks, um übertragen zu werden, mit einem Durchschnitt, der etwa zwischen 1 und 2 liegt. P-Frames befinden sich üblicherweise im Bereich 1-9 Blocks, mit einem Durchschnitt, der bei 3-4 Blocks liegt, I-Frames hingegen benötigen 1-30 Blocks pro Slice, mit einem Durchschnitt von ungefähr 10. (Werte ermittelt mit [AOT-2013i] in Modus 2: Slice Längen bestimmen.) In der Regel reicht es, die Länge zweier Slices zu messen, da zwei Slices hintereinander unter 4 Blocks bei einem I-Frame selten vorkommen, und auf einen schwarzen oder einfarbigen Bildschirm geschlossen werden kann. Kombiniert man die Methoden bei Unklarheiten, kann man recht zuverlässige Aussagen über den Frametyp treffen.

Eine alternative, leichtere und schnellere Methode (**viertens**) ist jedoch, einfach einen I-Frame anzunehmen und zu prüfen, ob der Decoder Fehlermeldungen produziert. Das intra_slice_flag [I32-1995, Kap.6.2.4] war ausnahmslos bei allen Slices 0, auch in I-Frames, kann also nicht zur Bestimmung des Frametyps herangezogen werden.[10]

4.3.3 Erzeugen der fehlenden Bildinformationen

Nach Erkennung des Frame-Typs wird in der Videodatei nach einem entsprechenden vollständigen Frame gesucht. Dieser wird von dessen picture_header bis zu der Stelle, an der die Übertragung des unvollständigen Frames beginnt, kopiert (identische slice_start_codes.). Anschließend werden die Bilddaten des unvollständigen Frames kopiert. Da einem Receiver keine Bilddaten eines zukünftigen Frames vorliegen können, sollte folgende Strategie gewählt werden:
Stelle einen gespeicherten I-Frame-Header her. Rechne die mittlere Bildfarbe und Helligkeit anhand der übertragenen Daten aus. Fülle die fehlenden Bildstellen mit festen Blocks dieser Farbe aus (oder nimm schwarz). Eine gute Kompromisslösung wäre, die horizontalen AC-

[10] Um dies zu prüfen, habe ich ein weiteres Programm, tsislice, geschrieben, welches eine .m2v Datei nimmt und sowohl Frametyp als auch Wert des intra_slice_flags für jeden Frame ausgibt. Die aus test4.ts entpackten Dateien weisen Unregelmäßigkeiten als Folge von Empfangsstörungen auf.

Koeffizienten der obersten decodierten Macroblock-Reihe nach oben zu wiederholen, also die fehlenden Bildinformationen mit diesen zu füllen, was optisch zu farbigen Balken führt, und sowohl von Aussehen als auch von Rechenzeit einen recht guten Kompromiss darstellt. An dieser Stelle wäre auch darauf hinzuweisen, dass manche Decoder (z.B. der VLC Media Player [www.videolan.org/vlc]) bei unzureichender Rechenleistung oder Datenfehlern im Stream dieses Verhalten zur Fehlerverschleierung anwenden (Siehe auch Bild 12-15).
Die Frame-Wiederherstellung ist auch kompatibel zu dem in Kapitel 7 genannten Verfahren, Slices von I-Frames unmittelbar nach der Decodierung anzuzeigen, anstatt auf die vollständige Übertragung des Frames zu warten. Nach Abschluss der Übertragung kann wie üblich mit der Decodierung fortgefahren werden.

4.4 Analyse der Ergebnisse

Die Wiederherstellung "verpasster" I-Frames ist mit geringem Rechenaufwand (Größenordnung O(N)) verbunden und produziert Ergebnisse, die zwischen akzeptabel und hervorragend liegen, und die Umschaltzeiten im bisherigen Worst-Case auf maximal 0,14 Sekunden reduzieren und den neuen Worst-Case auf 0,37 Sekunden[11] setzen. Diese Werte sind diskutabel und vom Betrachter abhängig. Die Frage "Ab wann ist das Bild als ungenügend zu betrachten?" stellt sich, da das Bild ja nicht vollständig wiederhergestellt werden kann.
Zwei Ergebnisse, bei denen der erste empfangene slice_start_code 00 00 01 06 (hex) ist (zufälligerweise), d.h. die Darstellung des Bildes kann ab Scanzeile 96 (von 576) beginnen, werden hier gezeigt:

Bild 5: Wiederhergestellter I-Frame aus sample.ts

[11] 1630 Pakete*835,5µs; 6000-1630 Pakete*835,5µs

Bild 6: Wiederhergestellter I-Frame aus test4.ts

Der in Slice 6 befindliche grüne Balken ist wahrscheinlich einem Programmierfehler zuzuschreiben. Selbst wenn das nicht der Fall wäre, kann man diesen Streifen durch Interpolation der oberen und unteren Slice ein wenig kaschieren (siehe auch nächstes Kapitel, denn der VLC Media Player tut genau dies), abgesehen davon erscheint er nur für eine halbe Sekunde.

Ein weiterer Videostrom mit verpasstem I-Frame-Header, der ab Startcode 15hex beginnt (von 36, also Scanzeile 240), enthält einige seltsame Artefakte. Zum Beispiel wird deutlich ersichtlich, dass die Decodierung ab der 6. Slice von unten vollständig abgebrochen wurde. Fehlermeldungen des Decoders bei der Decodierung schildern, dass der Decoder an dieser Stelle meinte, das Bild müsse zu Ende sein, und folgende slice_start_codes ignoriert hat.[HER-2013i] Eine Analyse ergab, dass tatsächlich Empfangsstörungen dafür verantwortlich zu machen sind.
Im oberen Teil des Bildes, der aus dem ersten vollständig übertragenen I-Frame stammt, sind Fehler zu sehen, die zwar wie Empfangsstörungen aussehen (was durchaus zu vermuten wäre, da mein DVB-T Empfang zum Zeitpunkt des Capturevorgangs mangelhaft war - was aber jedoch auch ein hervorragender Praxistest meines Wiederherstellungs-Tools ist), jedoch im tatsächlich übertragenen Bild fehlen: Das untere Bild zeigt denselben Frame, der im Modus 1 decodiert wurde - derselbe Frame im Modus 3 würde ebenfalls keine oder nur wenige Störungen zeigen, da Forward-Error-Correction zur Standardausstattung von DVB gehört. Demnach müsste der erste vollständige I-Frame in Modus 3 bereits von vorangegangener Fehlerkorrektur profitiert haben, im Modus 1 ist dies jedoch nicht möglich. Ergo muss ein Programmierfehler dafür verantwortlich sein.

Die einzig denkbare andere Antwort lautet, dass die Idee der I-Frame-Wiederherstellung an sich nicht fehlerfrei durchführbar ist, was jedoch in Anbetracht der großen, fehlerfrei wiederhergestellten Bildanteile und der sehr fehlerempfindlichen Huffman-Codierung der Macroblocks unwahrscheinlich ist; alle Macroblocks in einer Slice sind Huffman-codiert und sequentiell angeordnet, Beginn der Decodierung mitten in einer Slice ist unmöglich, Bitfehler am Anfang einer Slice zerstören die gesamte Datenstruktur der Slice (Epistasie).

Bild 7 (links): Ein weiterer wiederhergestellter I-Frame aus test4.ts Man bedenke, dass die obere Hälfte des Bildes bis zum 16 Pixel großen, grünen Balken aus einem zukünftigen I-Frame stammt.

Bild 8 (rechts): Der I-Frame, aus dem die oberen Bildinformationen von Bild 8 stammen.

Ebenfalls anzumerken ist bei diesem fehlerhaft decodierten Bild, dass es sich bei der Decodierung mit dem Referenzframework bei den Folgebildern "auflöst", also anstatt dass die Qualität sich festigt, sodass bei der weiteren Decodierung kein Anzeichen eines Fehlers mehr zu sehen ist, zerfällt das fehlerhaft wiederhergestellte Bild nach bereits 3 Frames in Wohlgefallen. Dies scheint eine Eigenschaft des Decoders zu sein. Bei der Betrachtung des Videostroms mit dem VLC Media Player fand diese Auflösung hingegen nicht statt, siehe Bild 12. Ebenso fällt auf, dass der VLC Media Player die genannte Interpolierung zum Kaschieren von Fehlern wie beschrieben anwendet, was bei den grünen Streifen im Bild passiert, besonders jedoch am unteren Bildrand bemerkt wird. Ebenso fällt auf, dass die Biene in Folgebildern große Mengen an Artefakten ansammelt, da der obere Teil der Biene aus dem zukünftigen I-Frame-Header stammt und sich entsprechend die Bewegungsvektoren der Zwischenbilder auf das falsche Bild beziehen, was verständlicherweise zu den beobachteten Artefakten führt, siehe Bild 13, 14, 15.

Betrachtet man alle Programme in diesem Strom sowie die letzten, vom Referenzdecoder decodierten Macroblocks in Bild 7, so ist anzunehmen, dass ein Empfangsfehler für den Verlust der unteren Bildinformationen verantwortlich ist. Die Debugausgabe des VLC Media

Players zeigt eine größere Menge von Datenfehlern an sowie, dass diese kaschiert wurden ("concealing") [VDO-2013i]. Mich verwundert dennoch, dass selbst der VLC Media Player, der für Robustheit und Fehlerkorrektur bekannt ist, die letzten 6 Slices ebenso verwirft wie das Referenzframework, da zu vermuten ist, dass diese Slices, deren Header ja bekanntlich im Stream vorhanden sind, noch zumindest teilweise gültige Daten enthalten müssen.

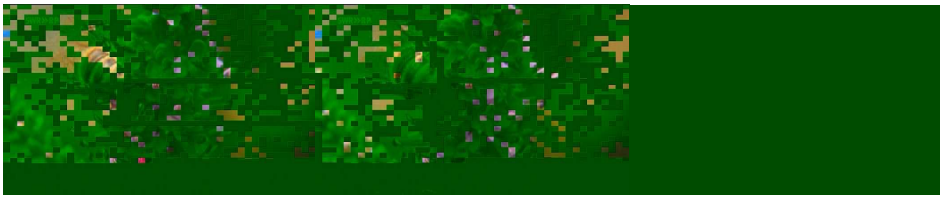


Bild 9,10, 11: Die folgenden 3 Frames von Bild 7

Bild 12, 13, 14, 15: Bild 7 sowie Folgeframes, mit VLC Media Player decodiert.

Betrachtet man das Video-Decoding-Beispiel aus den Grundlagen, fällt auf, dass eine korrekte Decodierung nur dann stattfinden kann, wenn die Intra-Quantiser-Matritzen (enthalten im Sequence Header) stimmen. Dies kann nicht über einen längeren Zeitraum gewährleistet werden. Stimmen die gespeicherten Intra-Quantiser-Matritzen nicht, ist mit verzerrten Helligkeits- und Farbwerten zu rechnen. In test4.ts werden im I-Frame-Header nur eine Non-Intra-Quantiser-Matrix übertragen. Verwendet man den I-Frame-Header eines anderen Videostreams für die Vervollständigung des gewünschten Streams, finden keine wahrnehmbaren Veränderungen statt. In sample.ts haben zwar alle Sender eine Intra-Quantiser-Matrix, diese ist jedoch für alle Sender unabhängig vom Bildinhalt identisch. Weiter ist anzunehmen, dass, wenn Sender diese Matrizen ändern, die Änderungen in geringem Umfang stattfinden und kaum wahrnehmbar sind (z.B. leuchtendere oder mattere Farben, mehr oder weniger Kontrast).

Betrachtet man die Dateien im Verzeichnis firstframes (auf der CD), so fällt auf, dass häufig nicht der tatsächlich erste Frame für diese Arbeit verwendet wird. Der Grund ist, dass die ersten 1-3 Frames nicht korrekt decodiert werden (siehe sämtliche Bilder im Verzeichnis "mode1" auf der beiliegenden CD). Der Grund dafür findet sich im Frame Reordering (siehe

Kapitel 3.1); hier wurde festgestellt, dass die beiden ersten B-Frames nach einem I-Frame in der Darstellungsreihenfolge vor den I-Frame gehören, d.h. anstatt dem Decoder zu helfen, indem früher in der Darstellungsreihenfolge auftretende Frames früher gesendet werden, werden diese später gesendet, sodass der Decoder noch länger mit dem Beginn der Wiedergabe warten muss.

5. Intra-Block-Interpolierung

5.1 Einleitung

Die Reduzierung der bisherigen Worst-Case-Zeit zum Umschalten um 0,14 auf 0,37 Sekunden ist schon ein ganz guter Anfang, jedoch treten zwei Probleme auf: Erstens liegen die neuen Umschaltzeiten immer noch hinter denen eines analogen Fernsehers mit einem für diese Geräteklasse langsamen Umschaltverhalten [AAT-2013i], zweitens sinkt der Anteil der angezeigten Bildinformation graduell, je später in einen verpassten I-Frame hineingeschaltet wurde, da nicht empfangene Informationen definitiv nicht angezeigt werden können.
Für die nun folgende Erklärung, wie durch Sichtbarmachung der Zwischenbilder die Umschaltzeiten weiter reduziert werden können, muss die Psychologie der visuellen Wahrnehmung des Menschen, insbesondere deren Stärken und Schwächen, die sich das Verfahren zunutze macht, näher beleuchtet werden.
Da dieses Verfahren auf DCT-Koeffizienten von Blocks/Macroblocks zugreift, ist es für die Implementierung auf der CPU eines Endgeräts mit externem Decoderchip ungeeignet, weil hierfür Werte decodiert und anschließend recodiert werden müssten. Eine Implementierung auf Decoderchips (üblicherweise DSPs (Digital Signal Processor)) ist zu empfehlen.

5.1.1 Psychologie des menschlichen Sehens

MPEG-2 und -4 Videokomprimierung, sowie das JPEG Bildkompressionsformat nutzen bereits eine menschliche Wahrnehmungsschwäche aus: Die Unfähigkeit, hochfrequente Farbänderungen zu detektieren, das heißt, dass die Farbauflösung des menschlichen Auges geringer ist als die Helligkeitsauflösung. Zwei weitere Schwächen, die Unfähigkeit, alle Details eines Bildes auf einmal zu erfassen, sowie die begrenzte temporale Auflösungsfähigkeit bei einem radikalen Szenenübergang (von einem Bild zu einem völlig anderen; aber auch von einem schwarzen Bildschirm zu einem Bild) [vgl. z.B. CSM-1993, Kap. 6.9 S.127] zusammen mit der Prioritätsliste der Aufmerksamkeit - worauf das Auge gelenkt wird - können genutzt werden, um auch unvollständige oder (farblich) verzerrte Bilder darstellen zu können, die vom Menschen immer noch ausreichend ausgewertet werden können, um z.B. die Entscheidung zu treffen, den Kanal erneut zu wechseln, oder auf ihm zu

verbleiben (vgl. normales Vorgehen beim Zapping), sowie eine ungefähre Vorstellung vom Bildinhalt zu bekommen.
Da das menschliche Auge nur im Gelben Fleck eine hohe spatiale Auflösung hat [WGE-2013] und somit nicht das gesamte Bild scharf abgebildet werden kann, richtet sich der Blick auf Punkte von Interesse. Für diese hat sich durch die Evolution eine Prioritätenliste gebildet. Diese ist dynamisch vom Kontext abhängig, bewegte Objekte, Gebiete mit hohem Kontrast, Gebiete mit hoher Frequenz (Detailreichtum) und Signalfarben stehen auf dieser Liste sehr weit oben. [vgl. ETR-2009, S.27, NWA-2013, KUG-2013: Aufmerksamkeit, WSW-2013, insbesondere WSW-2013 Kap. 3-4.].
Zusätzlich dazu führt das Gehirn noch eine Reihe von Operationen aus, bis das Bild bewusst wahrgenommen werden kann. Bereits auf physischer Ebene werden unterschiedliche Signale unterschiedlicher Sinneszellen getrennt übertragen, z.B. Trennung zwischen Bildinformation und Bewegungsinformation [CSM-1993, Kap 1.6.2, S.21], auf unterer psychischer Ebene einige recht simple Filter wie Kantendetektion und Kontrastverbesserung [CSM-1993, Kap 9.1 inkl. Unterkapitel], aber auch weitere Verarbeitungsstufen zur Bewegungserkennung, insbesondere bei/zu räumlicher Wahrnehmung [CSM-1993, Kap.1.4.4], weiter oben etwas kompliziertere Operationen wie Segmentierung, also Zerlegung des Bildinhalts in Regionen [MVC-1982], und auf höherer Ebene Abstraktion, Mustererkennung und schließlich das bewusste Wahrnehmen [vgl. z.B. CSM-1993, Kap 1.6.2]. Diese Liste ist bei weitem nicht vollständig.
Diese Operationen benötigen Zeit, was eine begrenzte temporale Auflösung zur Folge hat. [RBV-2013, Folien 7 und 10, sowie KUG-2013: Verarbeitungsgeschwindigkeit] gibt 150ms für die Objekterkennung an, von der die bewusste Wahrnehmung abhängt. Diese Zeit kann ausgenutzt werden, um kurze Zeiten mit unvollständigen oder fehlerhaften Bildinhalten zu überbrücken, funktioniert jedoch nicht, wenn Bildinhalte länger bekannt sind (z.B. ein Ausfall, nachdem bereits längere Zeit (z.B. 1 Sekunde) ein gültiger Videostrom vorhanden war, oder ein Ausfall nach einem erwarteten Szenenwechsel (z.B. Änderung der Kameraposition). In diesem Fall ist die Rekonstruktion des Bildinhaltes nach der Wahrnehmung durch das Gehirn jeder anderen Form der elektronischen Rekonstruktion überlegen.). Bei vorher nicht vorhandenen Bildinformationen kann man jedoch von den Verzögerungszeiten profitieren.

5.1.2 Sichtbarmachung der Zwischenbilder - TS2M2V Modus 2

Die vermeintlichen Schwächen des menschlichen Wahrnehmungsapparates verwandeln sich jedoch in Stärken, wenn es um die Erkennung von Bildern mit fehlenden, fehlerhaften oder verzerrten Inhalten geht. Betrachtet man Zwischenbilder, die ohne vorhergehenden I-Frame decodiert wurden, kann man trotzdem auf einigen, insbesondere bei bewegten, den Bildinhalt erkennen. Dies ist der Mustererkennungsfähigkeit des Gehirns zu verdanken.
Das Gesamtziel ist, aufgrund fehlender (weil verpasster) Bildinformationen unvollständige oder ungültige Bildsequenzen so weit aufzuwerten, dass dem Betrachter nicht sofort auffällt, dass diese fehlerhaft oder unvollständig sind, zumal ja innerhalb einer halben Sekunde fehlerfreie Bildinformationen folgen.
TS2M2V [ATS-2013i] hat, wie beschrieben, 3 Modi für MPEG-2 komprimierte Videoströme, wobei Modus 1 die herkömmliche Methode [Vgl. Kapitel 4.2] und Modus 3 die in Kapitel 3 beschriebene Methode zur Frame-Wiederherstellung ist. Modus 2 beginnt die Decodierung beim ersten vollständig erhaltenen Frame, unabhängig von dessen Typ. Modus 3 enthält die Menge aller Bilder, die Modus 2 entpackt, und zusätzlich das dem ersten picture_header vorangegangene Bild, welches unvollständig ist. Wenn dieses Bild jedoch kein I-Frame ist, ist der Qualitätsgewinn kaum bis nicht vorhanden, weshalb für dieses Kapitel die Ergebnisbilder von Modus 2 verwendet werden. [vgl. auch Bilder von Modus 2 und 3 auf der CD]
Intra-Block-Interpolation ist bei TS2M2V deshalb Modus 2, weil es algorithmisch erheblich leichter umzusetzen ist als die Frame-Wiederherstellung mit Modus 3, entsprechend wurde Modus 2 auch zuerst vollendet.
Betrachtet man die mit Modus 2 oder mit Modus 3 ohne I-Frame decodierten Bilder, sieht man die Deltainformationen in Form von undefinierten, geisterhaften Konturen (diese stammen von prediction errors), und gelegentlich vereinzelten Blöcken, die normal aussehen. Dies sind Intra-Blocks.

5.1.3 Analyse und Erklärung an Beispielbildern

Hier sind nun drei Zwischenbilder aus zwei Fernsehprogrammen. Die ersten beiden zeigen Bilder desselben Programms zu unterschiedlichen Zeiten (Frame 2 und Frame 6).

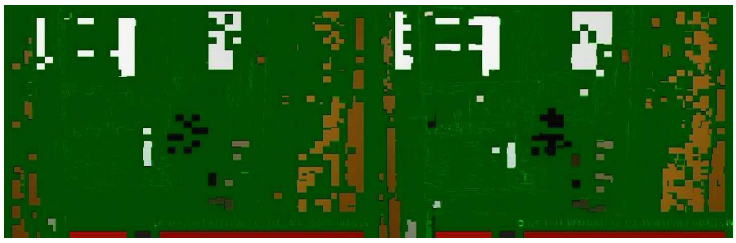


Bild 16, 17: 2 Frames aus einem Stream, dem kein I-Frame voranging

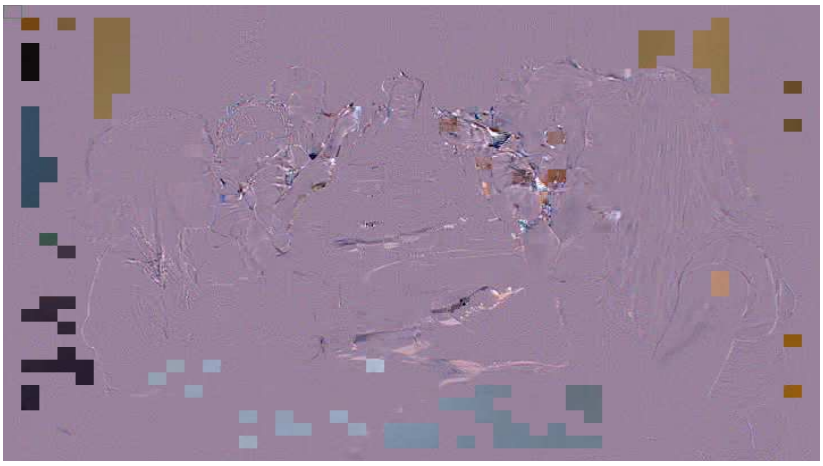


Bild 18: Frame aus dem Stream von Bild 32, ohne Wiederherstellung des I-Frames, aufgenommen mit CodecVisa (Initialisiert mit Pink statt Dunkelgrün). Die Druckerei hat mir eine Vorschau dieses Bilds gedruckt, welche deutlich schlechter zu erkennen ist, daher ist die elektronische Betrachtung dieses Bildes anzuraten.

An diesen Bildern kann man erkennen, dass Delta-Macroblocks seltenst Farb- oder Helligkeitsinformationen enthalten. Bewegt man eine der Bildsequenzen, bemerkt man, dass das Gehirn erstaunliche Arbeit bei der Rekonstruktion der Bildinhalte leistet. Am Standbild jedoch erkennt man wenig bis gar nichts.
Das Ziel ist nun, anhand von Intra-Macroblocks auf die Farbe der nicht übermittelten Blocks zu schließen. Die naheliegendste Idee ist, die Farben von übermittelten Intra-Macroblocks auf das Restbild auszudehnen. Genauer formuliert ergibt sich eine Interpolierung der Farben zwischen zwei übermittelten Intra-Macroblocks. Dies lässt sich am einfachsten auf der DCT-Ebene durchführen, da dort der Bildinhalt in den Ortsfrequenzraum übertragen wird, und dieses Verfahren horizontal und vertikal voneinander unabhängig ist.

5.2 Theoretische Umsetzung der Intra-Block-Interpolation

5.2.1 Grundlagen

Die Idee ist, dass ein Intra-Macroblock, der also im Gegensatz zu anderen Macroblockarten in P- und B-Frames über Helligkeits- und Farbinformationen verfügt, diese an seine Umgebung weitergibt, also sozusagen abfärbt. Ein hellroter Macroblock in einem Gebiet, dessen Farbe und Helligkeit unbekannt ist, lässt vermuten, dass ein gewisser Umkreis um diesen Macroblock ebenfalls hellrot ist. Da aber nicht anzunehmen ist, dass das gesamte Bild hellrot ist (es sei denn, es existiert nur ein Intra-Macroblock im Bild oder alle Intra-Macroblocks sind hellrot), muss sich die Wirkung des Blocks auf seine Umgebung beschränken (Einwirkradius).
Für eine gradiente Macroblockabstufung werden lediglich die ersten 3 Koeffizienten benötigt: Die Gesamthelligkeit, der erste vertikale und der erste horizontale Koeffizient. Diese sind in Diagramm 10 markiert. Durch die Zickzack-Abtastung der DCT, wodurch das zweidimensionale Feld in ein eindimensionales umgewandelt wird, sind das die ersten 3 codierten Werte in besagtem eindimensionalen Feld, wenn alternate_scan nicht verwendet wird. [I32-1995, Figure 7-2, S.81]

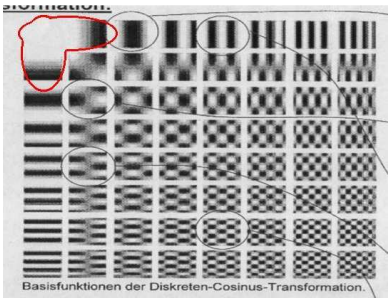


Diagramm 10: Diagramm 5, für Gradienten markiert.

In Richtungen, in denen keine Intra-Macroblocks enthalten sind, würde ich, ebenfalls in Abhängigkeit von der Entfernung zum Bildrand, einen Farbverlauf nach Grau und zusätzlich einen Helligkeitsverlauf nach Schwarz einfügen, um die Aufmerksamkeit des Betrachters auf die in höherer Qualität vorhandenen Bildinformationen zu lenken.
Hierfür und für das weitere Verständnis muss man den Begriff "Richtung" jedoch erst einmal definieren, da es im digitalen Bild nur 4 Richtungen gibt: oben, unten, links und rechts.

Befindet sich ein zweiter Intra-Macroblock 20 Macroblocks rechts und 1 Macroblock unterhalb des ersten, so kann er durch die Richtungsangabe "Rechts" nicht erreicht werden, obwohl er für den menschlichen Betrachter eindeutig rechts liegt.
Der Einwirkradius eines Intra-Macroblocks sollte abhängig von der Gesamtmenge aller Intra-Macroblocks im Bild und deren Entfernung zueinander sein. Sind viele Intra-Macroblocks über das Bild verstreut, sollte der Einwirkradius eines jeden Intra-Macroblocks reduziert werden, um ein schärferes Ergebnisbild zu erhalten. Steht wenig Rechenleistung zur Verfügung, muss die Anzahl genügen oder ein fester Wert für die Abschwächung der Einwirkung verwendet werden. Letzteres ist bei Zusammenballungen größerer Mengen an Intra-Macroblocks empfehlenswert.

5.2.2 Intra-Block-Interpolation: Vorbereitung zur Implementierung

Eine möglichst einfache Grundidee zur Implementierung des Intra-Block-Interpolation stellt sich folgendermaßen dar:
Für jeden Intra-Macroblock: Male einen Kreis, dessen Durchmesser von Anzahl und Verteilung aller Intra-Macroblocks abhängig ist, mit der Farbe und Helligkeit dieses Blocks. Zum Rand hin wird der Kreis zunehmend transparenter.
Dieses Verfahren hat 3 Phasen: Bestimme den Radius, Zeichne die Kreise und addiere die Kreise aller Blocks gewichtet.
Für Phase 2 muss jeder Kreis in einer eigenen Bildmatrix gespeichert werden, was einen erheblich gesteigerten Speicheraufwand hat. Alternativ kann man alle Kreise auf einer Bildmatrix speichern und mit 50% Transparenz beginnen.
Der Nachteil ist, dass hier lokale Informationen global verarbeitet werden müssen. Ein weiterer Nachteil ist, dass lokale Korrekturinformationen (z.B. zwei völlig unterschiedlich farbige Macroblocks in großer Nähe, was auf zwei unterschiedliche Objekte schließen lässt) völlig ignoriert werden.
Entsprechend muss ein computativ günstigerer Ansatz entwickelt werden, der lokale Informationen berücksichtigt. Ein solcher Algorithmus wird in 5.2.3 vorgestellt.
Da der Speicheraufwand von diesem Algorithmus tendenziell ausufern kann, ist zu empfehlen, kleinere Lücken zwischen Intra-Macroblocks zuerst lokal zu schließen, insbesondere 1 Macroblock große Lücken sind trivial zu schließen. Betrachtet man Bild 16, 17 oder 18, merkt man, dass Ansammlungen von Intra-Macroblocks nach Schließen der recht kleinen Lücken als eine einheitliche Form behandelt werden kann, welche dann auf einer

Speicherebene vereinheitlicht werden können. Zum Schließen der Lücken kann man sich auch Methoden aus der Diskreten Relaxation bedienen.

5.2.3 Intra-Block-Interpolation: Algorithmus

Mein Vorschlag wäre nun, den Vorgang der Intra-Macroblock-Interpolation ebenfalls in 3 Phasen einzuteilen. In diesem Vorgang wird der Einfachheit halber von Farbinformationen geschrieben. Gemeint sind hierbei Helligkeits- und Farbinformationen.
Die Erklärungen erfolgen an folgenden Beispielbildern:

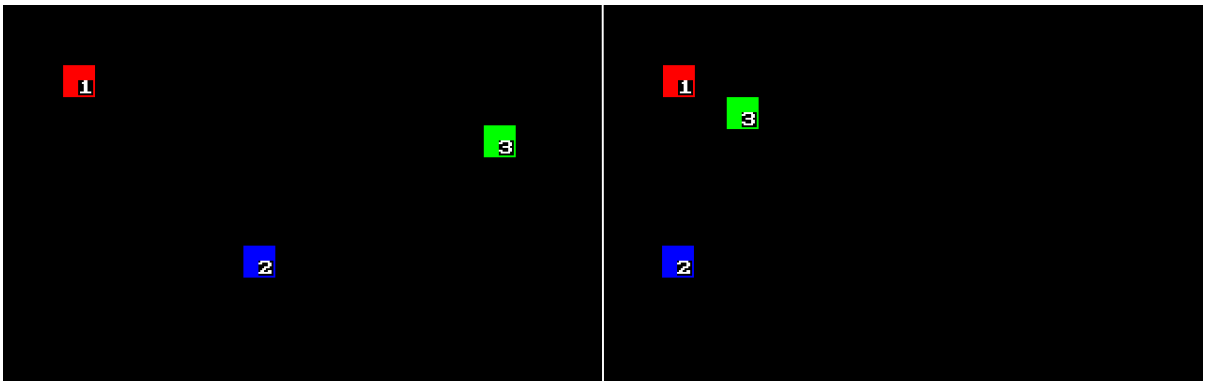


Bild 19, 20: Phase 0 (Ausgangssituation) der Intra-Block-Interpolation

Die bunten, mit Zahlen markierten Quadrate sollen Intra-Macroblocks darstellen, der Rest des Bildes ist schwarz, was hier "don't care", sprich: unbekannter Bildinhalt bedeutet.

Phase 1:
Alle Macroblocks über und unter einem Intra-Macroblock werden in der Farbe und Helligkeit dieses Intra-Macroblocks gefärbt (also alle mit derselben x-Koordinate). Dies geschieht, indem der erste Koeffizient der DCT der 6 (bzw. 8 oder 12, von chroma_format abhängig) Blocks, aus denen der zu verändernde Macroblock besteht, verändert wird. Weitere Bildinformationen werden davon nicht beeinflusst. Im Falle von Farbinformationen sind diese nicht einmal vorhanden. Liegt oberhalb oder unterhalb eines Intra-Macroblocks ein weiterer Intra-Macroblock, so kann hier schon eine Interpolation durchgeführt werden (siehe Interpolationsformel), es sei denn, die Intra-Macroblocks sind benachbart. In jedem Falle ist an dieser Stelle mit dem Prozess aufzuhören und beim nächsten Macroblock rechts des aktuellen Intra-Macroblocks fortzufahren, entsprechend wird in diese Richtung keine Farbinformation aufgebaut. Zusätzlich zur Einfärbung nach oben und unten wird bei der Interpolation jedoch auch in einer zusätzlichen Speichermatrix die Entfernung zum

farbgebenden Intra-Macroblock gespeichert. Im Beispiel ist das Verhältnis der Abstufung Horizontal zu Vertikal 2:1. Ideal wäre ein Verhältnis, das mit dem Seitenverhältnis des Bildes identisch ist, aber um Rechenzeit zu sparen, kann man Abstriche machen.
Die Ausgangsfarbe für die Interpolation wird aus der Kantenfarbe des Intra-Macroblocks gewonnen, siehe Bild 23 (hier wird Schritt 2 vorweggenommen), in zukünftigen Versionen ist es denkbar, das Muster des Intra-Macroblocks (siehe Diagramm 5) auf größere Bereiche des Bildschirms auszudehnen.

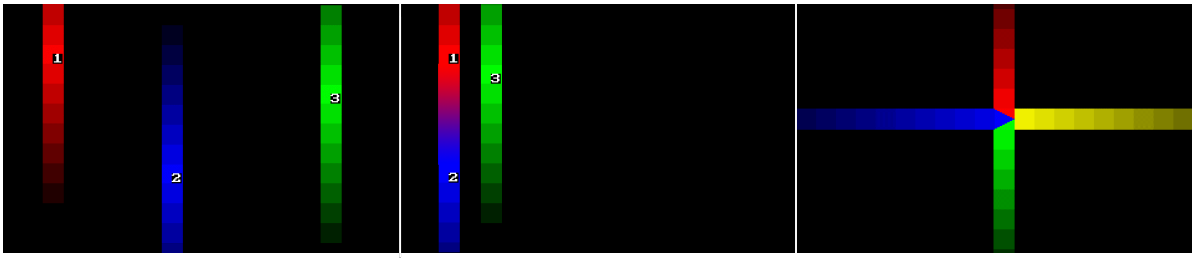


Bild 21, 22, 23: Ergebnis von Schritt 2 der Intra-Block-Interpolation für Bild 19, 20, sowie Veranschaulichung des Vorgehens für Intra-Macroblocks mit AC-Koeffizienten.

$$x' = \frac{(x - x1)}{(x2 - x1)} \quad , \quad f = f1 * x' + f2 * (1 - x')$$

Formel 2: Allgemeine lineare Interpolation.

f = Farbwert an der Stelle x'. f1 = Farbwert von Macroblock 1, f2 = Farbwert von Macroblock 2. DCT-Koeffizienten 2 und 3 werden so gewichtet, dass eine sanfte Abstufung zwischen den Blocks möglich wird.

x=Aktuelle x-Position. x1=x-Position Block 1, x2=x-Position Block 2. f steht für Farbe des Blocks, entsprechend f1 und f2 Farbe von Blocks 1 und 2.

(Allgemeine lineare Interpolationsformel. Für vertikale Interpolation wird entsprechend x mit y substituiert.)

Phase 2:

Von jedem Intra-Macroblock aus wird:

- Wenn nach links oder rechts kein weiterer Intra-Macroblock oder aus Phase 1 markierte, ausreichend nahe Farbinformationen vorhanden sind, wird, in Abhängigkeit zur Entfernung zum Bildschirmrand, eine logarithmische Abstufung zu Grau bzw. Schwarz durchgeführt. Logarithmisch, um die Bildinformationen so weit wie möglich zu erhalten, sodass erst auf die letzten 2-10% des Bildschirms das Bild ausgeblendet wird. Auf diesen Effekt kann man verzichten (was im Beispiel getan wird) und in

Richtung des Bildrandes überhaupt keine Abstufung durchführen, aber meiner Ansicht nach dürfte dies zu künstlich aussehenden Bildern führen. Bei künstlichen Ursprungsbildern (z.B. Zeichentrick, oder eben dieses Beispiel) führt das Weglassen dieses Effekts zu besseren Ergebnissen.

- Befinden sich links oder rechts (eher rechts, da der Algorithmus von Phase 2 entsprechend der klassischen Abtastung vorgeht, die von analogem Fernsehen bekannt ist und nach der auch die Slices im digitalen Fernsehen übertragen werden; links nur bei der Initialisierung, also dem ersten Intra-Macroblock einer Slice) andere Intra-Macroblocks oder ausreichend nahe Informationen aus Phase 1, wird zu diesen eine Interpolation durchgeführt, auch hier vorausgesetzt, sie sind nicht benachbart. "Ausreichend nahe" bedeutet eine Gewichtung der fremden Farbinformation im Verhältnis zur im Zusatzspeicher von Phase 1 angegebenen Entfernungsinformation. Je weiter sich die fremde Farbinformation von ihrem Intra-Macroblock entfernt, desto mehr überwiegt die in Phase 2 erstellte Farbinformation. Dies gilt einschließlich für die optionale Ausblendung zum Bildschirmrand hin. Übersteigt die Entfernung in Phase 2 jedoch deutlich (z.B. Faktor 2) der aus in Phase 1 gespeicherten Entfernung, so überwiegt die Farbinformation aus Phase 1.

Auch bei dieser Phase werden die Entfernungsinformationen in einer Temporärmatrix gespeichert.

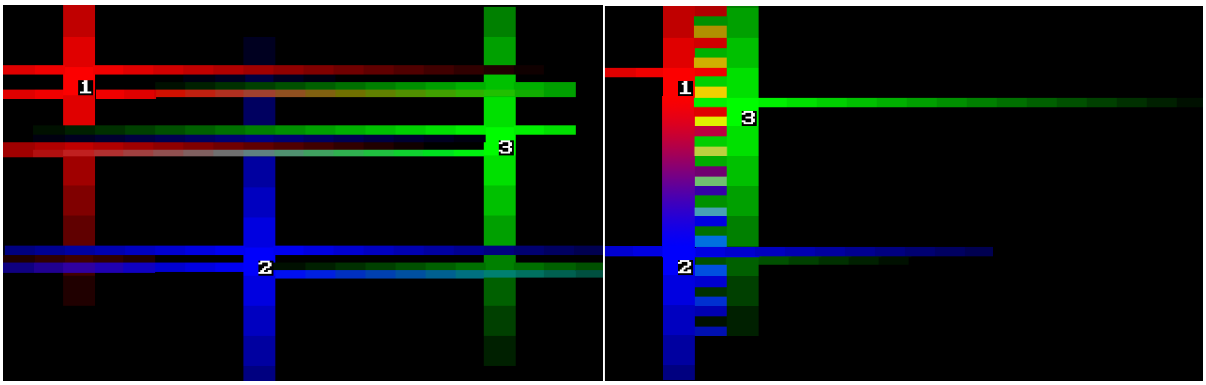


Bild 24, 25: Zwischenergebnis von Schritt 2 für Bild 19, 20.

In diesem Bild setzen sich die horizontalen Balken aus den Entfernungsinformationen der vertikalen Balken in der Reihenfolge des Auftreffens, sowie die resultierende Mischfarbe, zusammen. Ausgegangen wird jeweils vom Urheberblock, also dem übertragenen Intra-Macroblock, entsprechend ist dessen Farbe die oberste im Balken. Von diesem Vorgehen kann bei der Implementierung abgewichen werden. Die Anzahl der horizontalen Balken pro

Slice ist von der Anzahl der vertikalen Balken in Reichweite abhängig. In Bild 26 sieht man nur noch die resultierende Mischfarbe.

Phase 3:

Die Vorstufe von Phase 3 findet sich im folgenden Bild, das nun mindestens einen auf ganzer Breite durchgehenden Farbbalken (hier: 3) enthält:

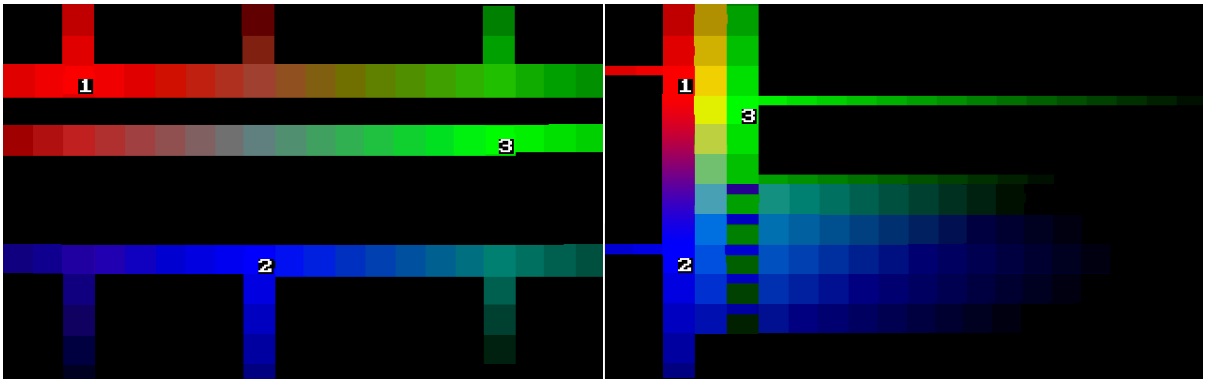


Bild 26,27 : Weiteres Zwischenergebnis von Schritt 2 für Bild 19, 20

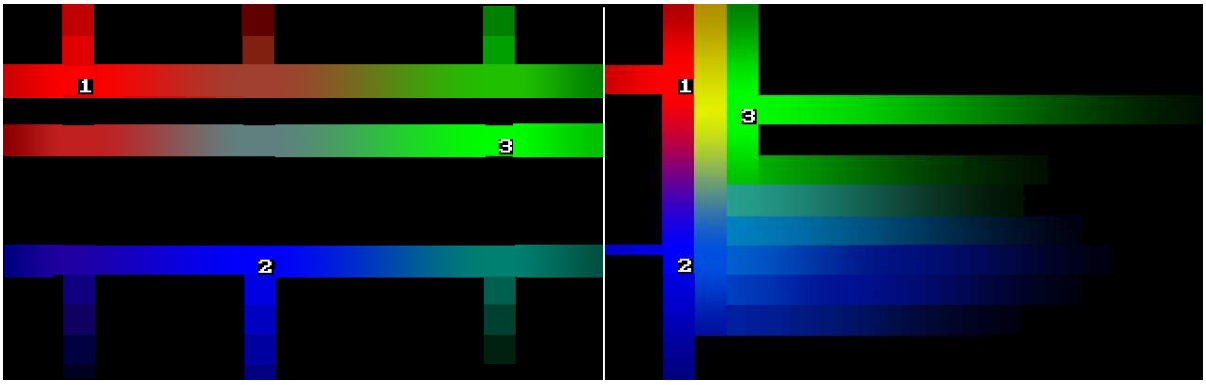


Bild 28,29: Weitere Verfeinerung des Zwischenergebnisses aus Phase 2 für Bild 19, 20.

Somit bleiben nur noch horizontale Farbgradienten. In Phase 3 wird vertikal zwischen diesen Säulen interpoliert, gemäß der allgemeinen linearen Interpolationsformel (x wird hier mit y substituiert):

Bild 30 und 31 verdeutlichen diesen Prozess. Diese Skizzen wurden zumindest teilweise von Hand erstellt. Eine bessere Abstufung hätte einen immens höheren Arbeitsaufwand bedeutet.

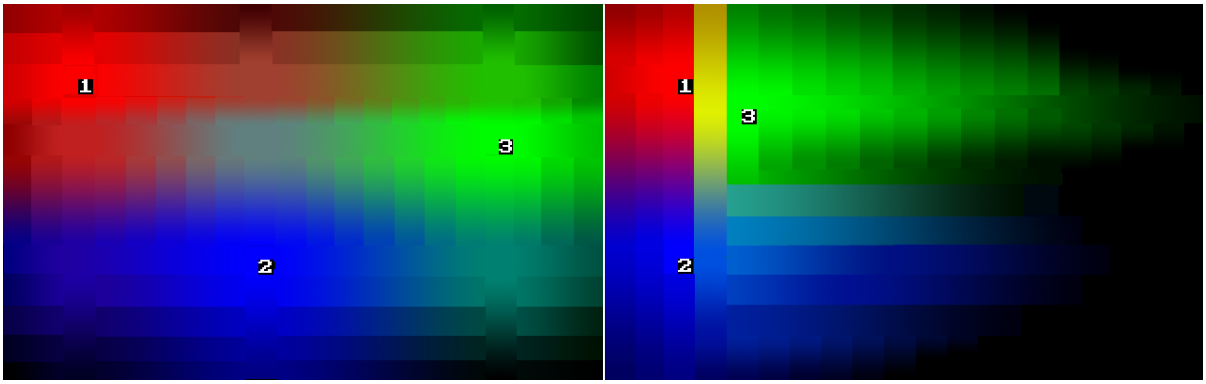


Bild 30, 31: Ergebnis der Intra-Block-Interpolation für Bild 19, 20.

Man vergleiche Bild 18 und Bild 19 mit Bild 32-34 und wende den Algorithmus analog an.

Intra-Macroblocks sollten bei diesem Verfahren über mehr als 3 Bilder hinweg gespeichert werden, da in aufeinanderfolgenden Zwischenbildern die Wahrscheinlichkeit von Intra-Macroblocks an unterschiedlichen Stellen steigt, sodass ein präziserer Farbverlauf über das gesamte Bild erzeugt werden kann.

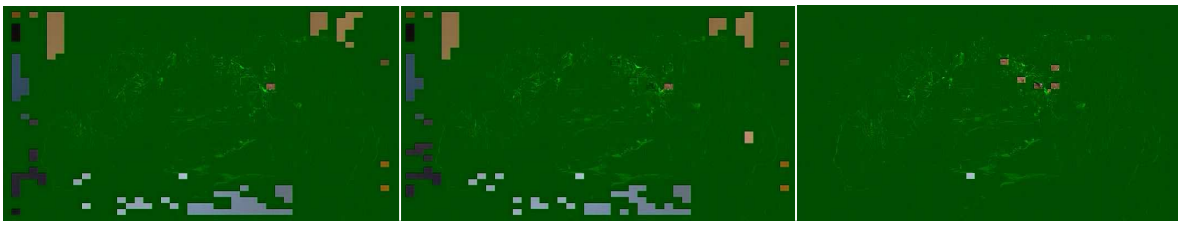


Bild 32, 33, 34: Sequenz aus Bildern aus sample.ts, man beachte die veränderlichen Positionen von Intra-Macroblocks

5.3 Analyse der (voraussichtlichen) Ergebnisse der Intra-Block-Interpolation

Das hier gezeigte Ergebnis ist ein Resultat der Intra-Block-Interpolation, jedoch entstand es von Hand, nicht mit Hilfe eines Computerprogramms. Hierbei war es mir nicht möglich, die Helligkeitsinformationen der Blöcke zu ändern, ohne dass dabei alle höheren Frequenzen, sprich: die bereits sichtbaren Details, verloren gingen (ich verwendete den GIMP Bildeditor, mit Aufteilung des Bildes in HSV-Ebenen (Farbe, Sättigung, Helligkeit)).

Da ich aus Zeitmangel[12] diesen Algorithmus nicht implementieren konnte, muss die Vorstellungskraft des Lesers für die weitere Ergebnisdiskussion herangezogen werden.

Bild 35: Von Hand durchgeführte Interpolation, allerdings nur für die Ebenen Farbe und Sättigung, nicht Helligkeit. Interpolationen außerhalb dieser Ebenen wurden nur in Bildbereichen ohne nennenswerte Informationen gemacht.

Als Erstes erkennt man, dass ich die Wichtigkeit der Helligkeitsinformation unterschätzt habe. Man erkennt, dass ohne Absolutwerte für Farbe und Helligkeit die Bilder in den Konturen geisterhaft bleiben, dafür aber bunter werden. Stellt man sich die Umsetzung der Helligkeitsanpassung der umliegenden Blocks vor, sieht man, dass Farben die Konturen verlassen, in die sie gehören. Die Esstischszene zeigt eine gute Optimierungsmöglichkeit, da Farben von durchgehenden Konturen aufgehalten werden könnten. Ein Konturverfolger würde hier für deutlich bessere Ergebnisse sorgen, hätte aber auch erheblich mehr Rechenaufwand zur Folge.
Selbst ohne Konturverfolger dürfte der Rechenaufwand deutlich steigen. Phase 1 und 2 beider Verfahren sind von der Anzahl der übertragenen Intra-Macroblocks abhängig. Man kann grob von einer Versechsfachung des Rechenaufwands pro Bild ausgehen.

Ein großer Vorteil der Intra-Block-Interpolierung sind Bewegungsvektoren. Diese bewegen nun nicht mehr nur Konturen, sondern Farbgradienten über den Bildschirm, was die Erfassung eines bewegten Objekts als Ganzes erleichtert, und man nicht nur Konturen sieht, die sich in die gleiche Richtung bewegen. Im ungünstigsten Fall war es sogar möglich, dass

[12] Um, wie beschrieben, die ersten 3 Koeffizienten der IDCT eines Macroblocks ändern zu können, müsste ich größere Änderungen im Referenzframework durchführen, was eines gründlicheren Studiums dessen Quellcodes bedurft hätte.

ein bewegtes Objekt keine Konturen erzeugt, z.B. wenn der prediction error vernachlässigbar gering ist.

Es gibt jedoch Situationen, in denen keine Intra-Macroblocks vorkommen, wie z.B. in folgender Szene:

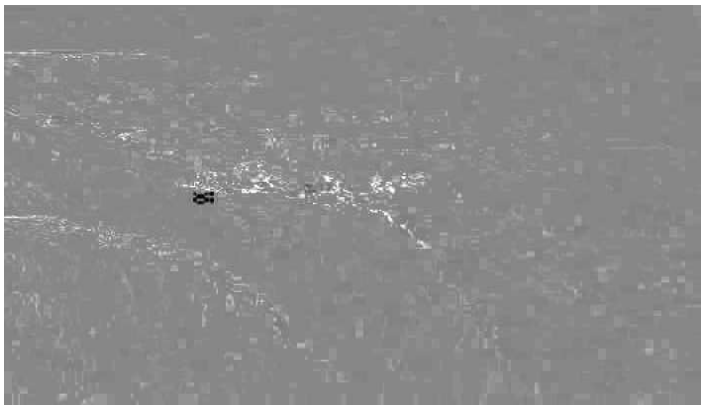


Bild 36: Erstes Bild aus dem hr-Videostream ohne vorhergehenden I-Frame

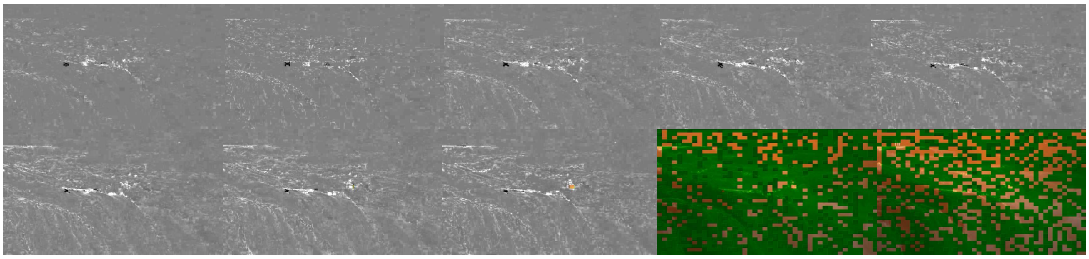


Bild 37-46: Bildsequenz bis vor den ersten I-Frame. Spielt man die Bilder als Video ab, erkennt man den Wasserfall.

Bild 47: Erster I-Frame aus dem hr-Videostream

In diesem Fall schlage ich eine Schwarzweiß-Darstellung vor, was in den obenstehenden Bildern bereits geschehen ist. Es existiert zwar ein Block mit teilweise enthaltenen Farbinformationen, diese sind jedoch knallrot und möglicherweise sogar ein Übertragungsfehler (schlechter Empfang). Eine Ausweitung extremer Farben auf das

Gesamtbild wäre also nicht ratsam; sind diese jedoch die einzige Farbquelle, so sollte eine abgeschwächte Version der Farbe Verwendung finden. Hierbei ist natürlich auch die Quantität der Farbvorkommen von Bedeutung. Existieren ausreichend große Mengen an Intra-Macroblocks, so kann deren Farbinformation bedenkenlos extrahiert werden. Im Videostream von Bild 36-47 taucht erst im 8. decodierten Bild ein einzelner Intra-Macroblock auf, der jedoch 2 Bilder vor dem I-Frame kaum noch Relevanz besitzt.
Betrachtet man ein Einzelbild dieser Sequenz für sich alleine, so erkennt man nicht viel; betrachtet man jedoch die gesamte Sequenz der ohne I-Frame decodierten Bilder, so erkennt man deutlich, dass es sich um einen Wasserfall handelt. Eine Analyse der Streams aus test4.ts mit CodecVisa ergab, dass zumindest der Hessische Rundfunk, der Bayrische Rundfunk und der Südwestfunk generell keine Intra-Macroblocks außerhalb von I-Frames übertragen.

Zusammenfassend kann man sagen, eine vollständige Wiederherstellung ohne einen vorangegangenen I-Frame ist unmöglich, es sei denn, das Bild enthält nur sehr wenige Details (z.B. schwarzer, einfarbiger oder mit einem Gradienten abgestufter Bildschirm) und der Datenstrom hat eine konstante Bitrate (was bei DVB gegeben ist). Hierbei muss der Encoder jedoch so eingestellt sein, dass er, wenn er beim Encoding unter die Sollbitrate fällt, Redundanzen produziert und überträgt. Häufig genug erzeugen Encoder stattdessen Stuffing-Pakete, also Pakete, die z.B. nur FF(hex) (also keine Bildinformationen) enthalten, um die Datenrate aufrechtzuerhalten.
Ein Beispiel, das selbst unbearbeitet einem fast vollständigen Bild entspricht, findet sich in Bild 48:

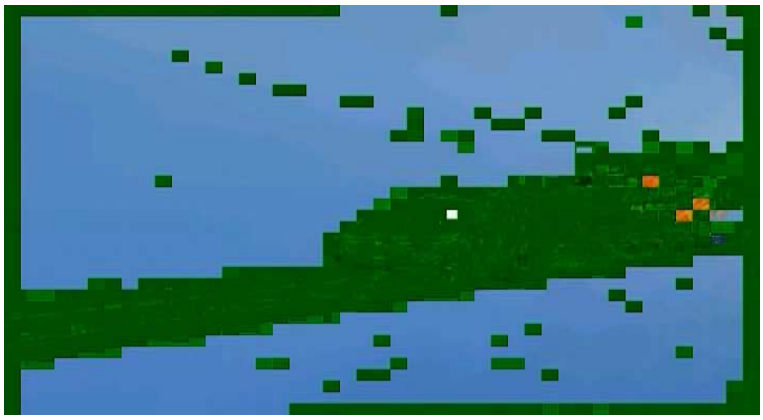


Bild 48: B-Frame ohne vorhergehenden I-Frame aus einem der Videoströme aus sample.ts.

Hier sehen wir eine Umkehrung der Bildqualität interessanter Punkte: Der Hintergrund - ein computererzeugter Farbgradient - ist fast vollständig redundant übertragen worden, während

der interessante Vordergrund - der Raketenschlitten - im I-Frame übertragen wurde und sich mangels Intra-Macroblocks in diesem Bereich nicht wiederherstellen lässt.
Eine Methode, welche die Intra-Block-Interpolation unterstützen würde, findet sich unter den aktiven Methoden in Kapitel 9: Ausblick.

Teil 2

6. HDTV

Für hochauflösendes Digitalfernsehen wurden 3 Codecs in Betracht gezogen: MPEG-2, MPEG-4 und H.264, auch bekannt als MPEG-4/AVC. In der Praxis werden jedoch nur MPEG-2 und H.264 verwendet. In dieser Arbeit erhält jeder dieser Codecs ein eigenes Kapitel, das im Wesentlichen eine verkleinerte Version des ersten Teils dieser Arbeit darstellt.

6.1 HDTV mit MPEG2 (ISO/IEC 13818-2)

MPEG2 ist die in den USA gängige Komprimierung für HD-Videoströme. Seit 2008 wird von der ATSC auch H.264 unterstützt, es steht stark zu vermuten, dass zur Zeit beide Codecs parallel verwendet werden [WAT-2013].
Eine Analyse eines Beispielstroms [www.dododge.net/roku/ts-samples.html, football.ts] mit CodecVisa ergab, dass der Strom dem Main Profile @ High Layer zugeordnet ist, laut Tabelle E-25 der Referenzdokumentation 13818-2 kommen demnach keine weiteren Eigenschaften von MPEG2 hinzu, wie Spatial oder Temporal Scalability (weitere Informationen, siehe [I32-1995], Kap 7.7 und 7.9), die hier erläutert werden müssten. Dementsprechend sind HD-MPEG2-Videoströme von den Grundlagen her identisch mit SD-MPEG2-Videoströmen, lediglich mit anderen Parametern für Auflösung und Bildrate. Es wäre jedoch auch denkbar gewesen, dass der HD-Layer spatial und temporal von einem SD-Layer hochskaliert ist, um Bandbreite zu sparen. Im Beispiel-Transportstrom existiert jedoch ein SD-Simulcast des HD-Videostroms, was ebenfalls gegen eine Zweiteilung des Stroms in Base und Extended Layer spricht. ([WAT-2013] bestätigt MP@HL als einzig erlaubtes Profil)

6.1.1 Analyse: Unterschiede

In Sachen Parameter des Videostroms[13] kann man zusammenfassend schreiben:

[13] Der heruntergeladene Transportstrom ist nicht synchron, d.h. er beginnt nicht mit einer 0x47. Er hat einen Offset von 24(dezimal) Byte, der erst beseitigt werden muss. Dies kann man z.B. mit dem beigelegten Programm rmvoffs.exe tun.

Auflösung 1280*720, 59,94Hz (NTSC). (Quelle: [CV-2013], [PDM-2013], kann aber auch von Hand ermittelt werden (siehe das Beispiel in Kapitel 2.4.2))
Weitere Analysen, beispielsweise mit Orderts [AOT-2013i] oder auch CodecVisa [CV-2013], ergeben, dass alle 15 Bilder ein I-Frame kommt; rundet man die Framerate auf 60Hz auf, ergeben sich also 4 I-Frames pro Sekunde.
Die von CodecVisa ermittelte Frame-Reordering-Reihenfolge für MPEG-2 trifft auch hier zu, entsprechend müsste der Decoder mindestens bis einschließlich zum ersten B-Frame nach dem ersten P-Frame (also 4 weitere Frames nach Erhalt des I-Frames) abwarten, bis er mit der Wiedergabe beginnen kann - es sei denn, er zeigt so lange ein Standbild des I-Frames (da dieses Verhalten bei manchen SD-Decodern beobachtet wurde, liegt nahe, dass auch bei manchen HD-Decodern dieses Verhalten implementiert ist).
In [AFT-2013i], also mit einer Kombination aus Orderts und MPEG-2 TS packet analyzer, habe ich ermittelt, dass eine Sekunde ziemlich genau (±1%) 10088 Pakete "dauert" (da noch andere Videoströme vorhanden sind) und ein Worst Case-Timing von 4756 Paketen (bzw. streng genommen eins weniger), d.h. eine Worst-Case Umschaltzeit von 0,47 Sekunden mit (bzw. 0,39 Sekunden ohne) Einbeziehung von Frame-Reordering, also wenn der Decoder ein Standbild des I-Frame zeigt.
Eine Analyse der Slicelängen und Bildgrößen mit Orderts ergibt, dass I-Frames nur geringfügig mehr Platz im Stream wegnehmen als P- und B- Frames, wobei der Abstand zwischen I- und B- Frames immer noch ausreichend für eine Trennung nach Schwellwert ist. Um die Slicelängen mit einem Schwellwertverfahren zuverlässig einem bestimmten Bildtyp zuzuordnen, sind mehr Werte nötig als bei SD-Video, wo 2-3 Werte genügten. Für HDMPEG2.TS empfehle ich 4-6 Werte, oder die Annahme eines I-Frames und prüfen, ob der Decoder Fehlermeldungen liefert.
Für MPEG-2 HDTV Slices gelten dieselben Konventionen wie bei SDTV, nämlich dass eine Slice am linken Bildrand anfängt und am rechten Bildrand endet, sowie dass 2 Macroblocks untereinander immer zu unterschiedlichen Slices gehören.
Eine Analyse der Bilder mit CodecVisa zeigt, dass das in HDMPEG2.TS verwendete Bildmaterial eine recht große Menge an Intra-Macroblocks in P- und B- Frames (insbesondere in P-Frames) aufweist, was eine genauere Intra-Macroblock-Interpolation möglich macht, allerdings auch für die größeren Bilder/Slices verantwortlich ist.

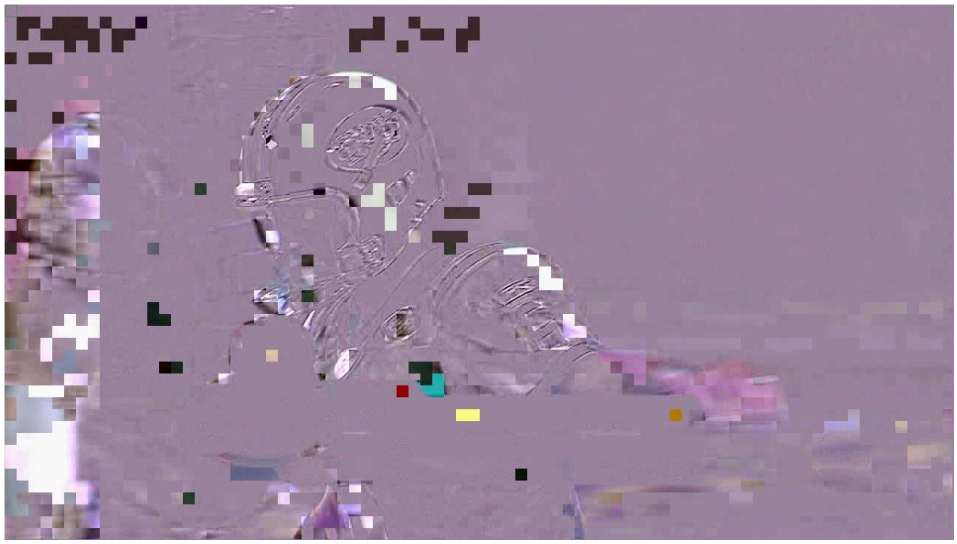


Bild 49: Delta-Frame aus dem HD-Videostrom von hdmpeg2.ts ohne vorangegangenen I-Frame. Es ist der dritte empfangene Frame.

6.1.2 Anpassung der Algorithmen

Durch die Geringfügigkeit der Unterschiede im Aufbau des Videostroms müssen keine grundlegenden Veränderungen an den Algorithmen gemacht werden. Aufgrund der Tatsache, dass die Größenverhältnisse zwischen I-/P-/B-Frames zusammenrücken, müssen die Schwellwerte für die Erkennung des Frametyps für die Frame-Wiederherstellung angepasst werden, sowie die Menge der Slices, deren Größe gemessen wird, zu erhöhen. Wie bereits geschrieben, empfehle ich, statt 2-3 Slices nun 4-6 Slices abzuwarten. Auch zu empfehlen wäre, dass die Schwellwerte nicht fest sind, sondern vom Decoder im Betrieb angepasst werden können. Die Frame-Wiederherstellung selbst kann 1:1 ablaufen. Für die Intra-Block-Interpolation können niedrigere Werte für den Einwirkradius verwendet werden, da die Erhöhung der Menge an Intra-Macroblocks die erwartete Vergrößerung der Einwirkradien aufgrund der erhöhten Bildauflösung weit übersteigt und somit die Einwirkradien verkleinert werden sollten. Da mir kein anderes Bildmaterial zur Verfügung steht, bleibt dies eine Empfehlung, keine Vorschrift.
Die Skalierung der Einwirkradien in Abhängigkeit von der Auflösung und der Menge der Intra-Macroblocks sollte sowieso in langen Testreihen bestimmt werden.

6.1.3 Analyse der Ergebnisse

Die Resultate (siehe Bild 50-57) sind vergleichbar mit denen aus Kapitel 5. Jedoch sind aufgrund des erhöhten Datenaufkommens mehr Intra-Macroblocks vorhanden und die Konturen in Inter-Macroblocks sind schärfer und deutlicher zu erkennen. Man braucht deutlich weniger Fantasie, um zu erkennen, was das Bild zeigt. Lediglich der etwas unscharfe Hintergrund ist schwer zu erkennen.

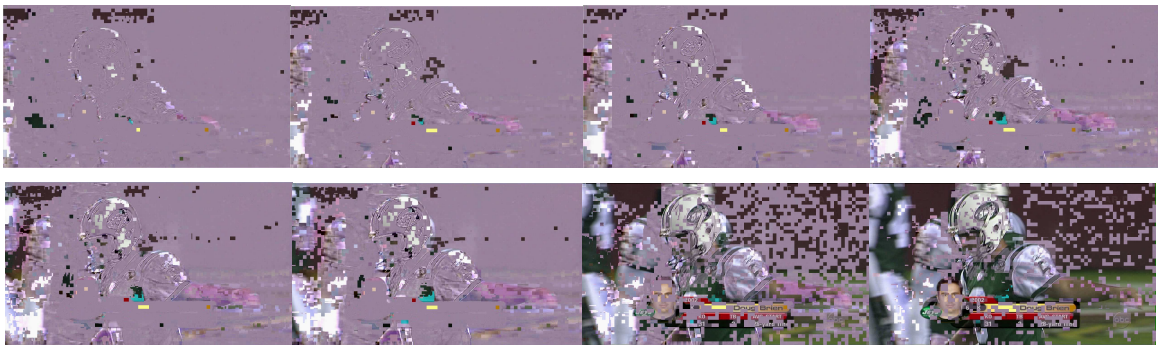


Bild 50-57: Bildsequenz aus dem HD-Videostrom aus hdmpeg2.ts vor dem ersten I-Frame.

Bild 58: Erster I-Frame des HD-Videostroms aus hdmpeg2.ts.

In Bild 50-57 erkennt man auch deutlich das Konzept der Aufdeckung nicht vorhandenen Bildinhalts: Die Kamera bewegt sich nach links, entsprechend bewegt sich der Bildinhalt nach rechts. Von links kommt also neuer, unbekannter Bildinhalt hinzu, der nur schwer als Deltainformation zu übertragen ist. Encoder gehen zwar nach dem Prinzip der Ähnlichkeit auf

dem gesamten Bild vor, was man in den unscharfen Bereichen oben und unten links, die als Inter-Macroblocks übertragen werden, erkennen kann, aber die neuen Vordergrundinformationen muss der Decoder intra codieren, sodass vor dem I-Frame schon ein verhältnismäßig gutes Bild vorliegt.

6.2 HDTV mit MPEG-4 (ISO/IEC 14496-2)

MPEG-4 hat im HD-Broadcasting-Bereich nur marginale Bedeutung. Liest man dort MPEG-4, wird meistens MPEG-4/AVC = H.264 gemeint, was sich häufig durch fehlende Expertise zur Bezeichnung MPEG-4 reduziert. Auf der Wikipedia Seite „Geschichte des hochauflösenden Fernsehens“ wird MPEG-4 nicht erwähnt. Da H.264 eine mindestens doppelt so hohe Komprimierungseffizienz wie MPEG-4 hat [WGH-2013], ist zu erwarten, dass Sender, die noch nicht in HD senden, stattdessen H.264 verwenden werden.
Aufgrund dieser geringen Bedeutung wird dieses Kapitel nur sehr oberflächlich ausfallen.

6.2.1 Grundlagen

MPEG-4 ist meines Erachtens MPEG-2 ähnlicher als H.264 zu MPEG-4. Dies liegt, so vermute ich, daran, dass MPEG-4 auf MPEG-2 basiert, H.264 aber neu entwickelt wurde (von der ITU) und erst anschließend in die ISO/IEC 14496-Familie aufgenommen wurde. [WH2-2013]
Einige wesentliche Neuerungen in MPEG-4 umfassen, dass anstelle von Frames Video Object Planes (VOPs) verwendet werden [I42-2001, 6.1.1-6.1.3, insbesondere 6.1.3.3], welche eine Form und eine ID haben, sowie durch Überspringen von Macroblocks wahlweise den gesamten Bildschirm oder nur einzelne Teile desselben abdecken können [I42-2001, 6.2.3]. Die Form ist in der Regel "rectangular", also rechteckig, oder "Shape". In letzterem Falle wird eine binäre Transparenzmaske übertragen (Pixel oder Macroblock ist transparent oder opak). Ebenso kann sich ein Bild aus mehreren VOPs zusammensetzen. Zusätzlich dazu gibt es Sprites. Ein Sprite ist ein Videoobjekt, das zusätzlich über Koordinaten verfügt.[I42-2001, Kapitel 8.2, weitere Informationen in Kap. 7.8-7.8.7] Dies erlaubt, bewegte Objekte unabhängig von deren Hintergrund codieren und übertragen zu können. Abgesehen davon sind sämtliche Videoobjekte unabhängig voneinander.

MPEG-4 hat darüber hinaus eine Reihe von dreidimensional basierten Bildmorphing-Operationen, wie z.B. ein 2D- und 3D-Mesh (also eine 2D-Textur, die anhand von 3D-Stützkoordinaten "gewarped", also z.B. zerknittert, gedreht oder auch einfach nur skaliert werden kann) [I42-2001, Kap. 6.1.4, 6.1.6], das 3D-Mesh kann verwendet werden, um höhere dreidimensionale Geometrien (z.B. Kugeln) zu erzeugen. MPEG-4 besitzt darüber hinaus ein eingebautes dreidimensionales Gesichtsmodell und ein eingebautes dreidimensionales Körpermodell [I42-2001, Kap. 6.1.5]. Von diesen Erweiterungen sind aber höchstens die Meshes interessant, da diese affine Transformationen erlauben, wenn beispielsweise eine Tür geschlossen wird, muss deren Textur für beide Seiten nur einmal übertragen werden, der Bewegungsvektor kann dann dreidimensional übertragen werden.

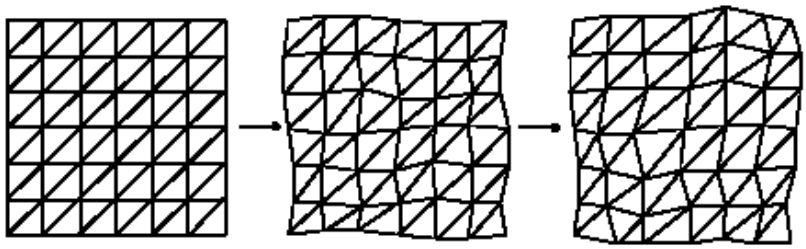


Bild 59: Mesh. [I42-2001]

Bilder bzw. VOPs werden nicht mehr in Slices, sondern als Ganzes übertragen, die VLCs für die Bilddaten selbst sowie die Metadaten - wie Macroblock-Typ und -Adresse - sind derart gewählt, dass es möglich ist, sie rückwärts auszulesen. Im Falle eines Datenfehlers im Strom kann also vom Ende des Bildes bis zum Fehler dieses rückwärts gelesen und decodiert werden. [I42-2001, E.1.3]

Um das Fehlen von Slices auszugleichen, können im Bild resync_marker angebracht werden. Resync-marker erlauben die Synchronisation im Bild und sehen aus wie verkürzte Startcodes. Eine der Informationen, die einem resync_marker folgt, ist die absolute Adresse des nächsten Macroblocks [I42-2001, Kap. 6.2.5->6.2.5.2, video_packet_header], sodass man dessen Position und die der folgenden Macroblocks nach folgender, einfacher Formel ausrechnen kann, wenn die Bildauflösung bekannt ist.

```
x=mb_number%(16*VOP_width)
y=mb_number//(16*VOP_width)
```

Formel 3: Ermitteln der Position eines Macroblocks anhand seiner Nummer (mb_number). % ist der Modulo-Operator, der den Rest einer Ganzzahlteilung liefert, // ist hier das Symbol für eine Ganzzahlteilung.

6.2.2 Analyse und Konzeption

Da keine Videoströme vorliegen, enthält dieses Kapitel eine Auflistung der am wahrscheinlichsten verwendeten Möglichkeiten von MPEG-4 sowie Ideen, diese für die Maximierung der Umschaltgeschwindigkeit einzusetzen.
Für den TV-Regelbetrieb wird sehr wahrscheinlich eine einzelne VOP pro Bild, die dieses vollständig ausfüllt, übertragen, und die klassische IBBPBBPB...-Reihenfolge oder eine Abart derselben verwendet.
Eine Alternative bieten mehrere VOPs, dies muss der Encoder jedoch unterstützen. Der Vorteil davon wäre, bestimmte Teile des Bildes zu unterschiedlichen Zeiten intra zu codieren, sodass sich das Bild beim Umschalten nach und nach mit Objekten füllt, aber bereits sehr schnell etwas zu sehen und vor allem zu erkennen ist, da intra-codierte Objekte eben 1:1 übertragen werden.
Die Fähigkeit, Bilder rückwärts auszulesen, ermöglicht die Wiederherstellung von nur teilweise übertragenen Frames, eine Frame-Wiederherstellung, wie in Kapitel 4 beschrieben, wäre also von Anfang an Teil des Videocodecs. Der Nachteil hierbei ist, dass eine Beschleunigung des Bildaufbaus, wie in Kapitel 7 beschrieben, nicht durchgeführt werden kann, weil die Rückwärtsdecodierung erst mit dem Abschluss der Übertragung des Frames begonnen werden kann.

6.3 HDTV mit H.264 (ISO/IEC 14496-10)

In Deutschland und anderen europäischen Ländern ist H.264 der gebräuchliche Codec für die Übertragung von HDTV: "Bei allen [(DVB-S2 und DVB-C)] kommt die effektive MPEG-4/AVC-Videokomprimierung zum Einsatz.", "H.264 erreicht typischerweise eine etwa dreimal so hohe Codiereffizienz wie H.262 (MPEG-2) und ist auch für hoch aufgelöste Bilddaten (z.B. HDTV) ausgelegt. [...] Allerdings ist der Rechenaufwand auch um den Faktor 2 bis 3 höher." [WH2-2013].[14]
Die subjektive Bildqualität steigt ebenfalls.

[14] In einigen Ländern, in denen sich analoges Antennenfernsehen länger gehalten hat, wird H.264 auch für SDTV (DVB-T) verwendet. Die nötigen Belege kann ich jedoch beim besten Willen nicht mehr finden, das Thema ist auch nicht von ausreichender Wichtigkeit für einen zwingenden Beleg.

6.3.1 Grundlagen: Aufbau des digitalen Bildes

Da sich H.264-Bilder im Aufbau und der Übertragung teils stark von MPEG-2 Bildern unterscheiden, müssen die Grundlagen von H.264-Bildern dargelegt werden, vergleichbar mit Kapitel 2, jedoch weniger detailliert. Auf eine Beispieldecodierung wie in Kapitel 2.4.2 wird verzichtet. Zur Veranschaulichung des Aufbaus eines H.264-codierten Videostroms kann das von mir zu diesem Zweck entwickelte Programm h264dec.exe [AH2-2013i] auf der beiliegenden CD verwendet werden. Man bedenke, dass es auf bereits entpackte Videodateien, nicht auf Transport Streams angewendet werden muss. Für letzteres steht nach wie vor das Programm Orderts.exe [AOT-2013i] zur Verfügung, welchem ich Erweiterungen eingebaut habe, die die Analyse von H.264 Dateien auf ähnlichem Level erlauben wie bei MPEG-2, mit Ausnahme der Erkennung der Slice-Längen. Warum dies so ist, wird später (in Kapitel 6.3.3) deutlich.
Die Unterschiede zu MPEG-2 im Aufbau des Bildes selbst sind nicht so gravierend, es handelt sich hier mehr um Erweiterungen und Verbesserungen als um Änderungen. Die Transportmodalitäten, d.h. wie der Stream zusammengesetzt ist und die Bilddaten aus dem Stream gewonnen werden, haben sich jedoch stark geändert.

6.3.1.1 Blocks/Macroblocks

Wie schon bei MPEG-2 setzt sich das Bild wieder aus Blocks anstelle von Pixeln zusammen. Blocks werden in Macroblocks zusammengefasst, die, wie bei MPEG-2 immer 16x16 Pixel umfassen, ein Block kann bei H.264 jedoch verschiedene Größen haben, nämlich 16*16 (1 Block pro Macroblock (MB)), 8*8 (4 Blocks pro MB) und 4*4 (16 Blocks pro MB). In P- und B- Frames sind sogar Mischungen erlaubt, z.B. 8*16 (2 Blocks pro MB) oder ein 8*16 Block und zwei 4*4 Blocks pro MB. Tabellen 9-26, 9-27 und 9-28 in [I4A-2004] enthalten eine Liste[15] der verschiedenen Macroblock-Typen mit deren Codierung.
In I-Frames werden jedoch nur symmetrische (16*16, 8*8 und 4*4) Macroblock-Typen verwendet. (Quelle: CodecVisa mit hd_test5.m4v, siehe Bild 60) Die genauere Abstufung und Zerlegung von Macroblocks in Blocks unterschiedlichen Typs wird verwendet, um eine genauere Bewegungsprädiktion zu erlauben. In I-Frames möchte man nur die Anzahl der Null-Koeffizienten in einem Run maximieren; hierbei hilft die Zerlegung in einen 16*16-

[15] unvollständig, da veraltet - z.B. ist der 8*8 Modus in I-Frames nicht gelistet, obwohl er existiert und häufig verwendet wird

Block für Bildregionen mit wenigen Details (niedrige Frequenzen im Ortsfrequenzraum), bzw. in 16 4*4-Blöcke für Bildregionen mit vielen Details.
Eine Erweiterung bieten P-Macroblocks in I-Frames, also Macroblocks, deren Bildwerte von anderen Bildwerten aus demselben Frame abgeleitet werden, d.h. nicht alle Macroblocks im I-Frame werden direkt übertragen; Bildinformationen können jedoch von unterschiedlichen Stellen aus demselben Bild abgeleitet werden, ohne diese erneut übertragen zu müssen. Der I-Frame bleibt also trotz P-Macroblocks vollständig definiert. [I4A-2004, Kap. 0.5, 8.3]
Leider zeigt CodecVisa P-Macroblocks in I-Frames nicht gesondert an (Bild 60).

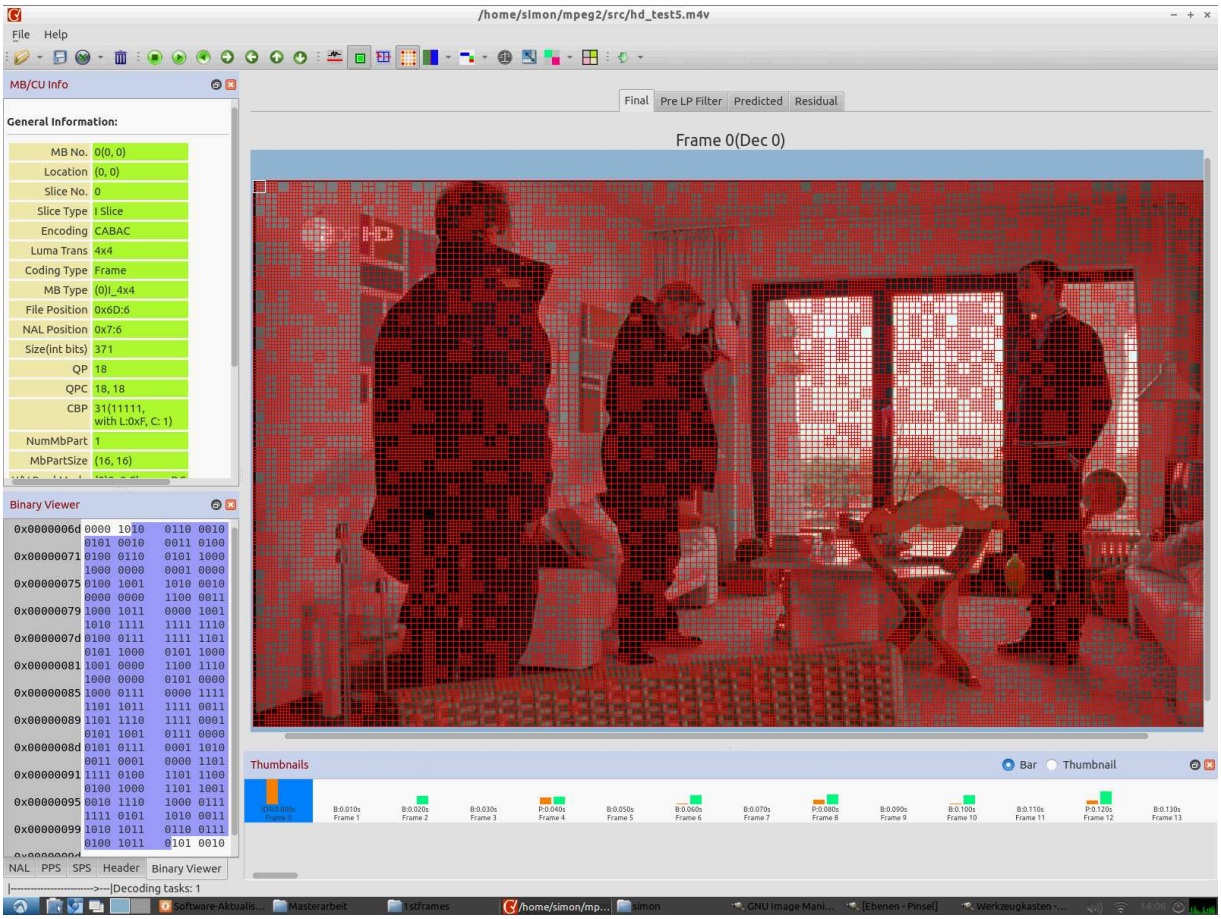


Bild 60: Screenshot von CodecVisa mit aktiven Macroblock-Typmarkierungen. P-Macroblocks in I-Frames werden leider nicht markiert, aber man erkennt, dass nur symmetrische Macroblocks vorkommen.

Eine weitere Neuerung sind I-PCM Macroblocks, welche nur in I-Frames vorkommen können. Diese werden unkomprimiert in 8 Bit übertragen, d.h. das gelesene Byte aus dem Videostrom enthält tatsächlich die Helligkeit des Pixels (bzw. dessen Farbe im Chroma-Teil des I-PCM-Macroblocks). I-PCM-Macroblocks können verwendet werden, um extrem

hochauflösende, extrem detaillierte Bildregionen verlustfrei übertragen zu können. [I4A-2004, Kap. 8.3.4, geht aus Formel 8-105 hervor]

Anstelle der Diskreten Cosinus Transformation (DCT) wird eine Integertransformation auf stets 4*4 Pixel großen Blöcken durchgeführt, welche im Wesentlichen eine verfeinerte und auf CPU-Performance optimierte DCT darstellt - möglichst wenige Prozessorzyklen, z.B. durch Shift und Addition statt Multiplikation und Division, unter Inkaufnahme von Ungenauigkeiten -, welche über Matrizenmultiplikation durchgeführt wird. Die Werte in den Matrizen sind so gewählt, dass sie wie beschrieben durch alternative und Rechenzeit sparende Instruktionen verrechnet werden können. [RTQ-2009]

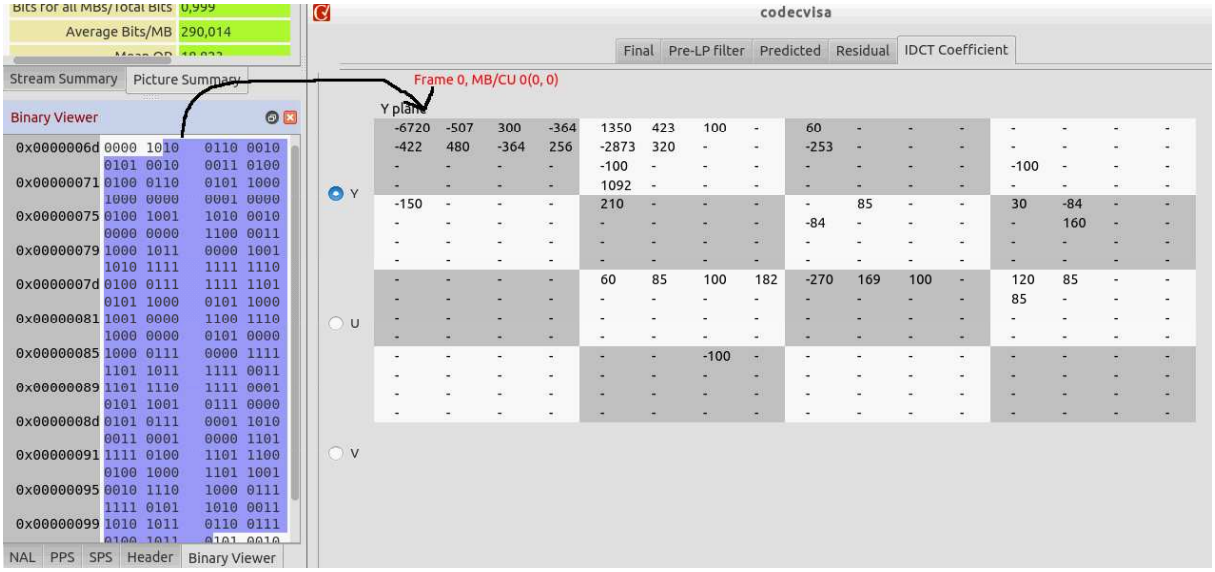


Bild 61: Screenshot von CodecVisa, mit den Koeffizienten eines Macroblocks (rechts) und dem zugehörigen Bytestream (links)

Ein wesentlicher Unterschied zu MPEG-2 auf Macroblock-Ebene ist, dass Intra-Macroblocks in Non-Intra-Frames nicht 1:1 dargestellt werden können, also die Bezeichnung "Intra" eigentlich nicht verdient haben. Zumindest hat keiner der verwendeten Videodecoder (VLC Media Player, Gnome Player, CodecVisa, Referenzframework) Intra-Macroblocks betrachtbar angezeigt. Sie sahen aus wie alle anderen Non-Intra-Macroblocks auch. Betrachtet man beispielsweise Bild 62 und 63, dann 66 und folgende, kann man diesen Sachverhalt erkennen.

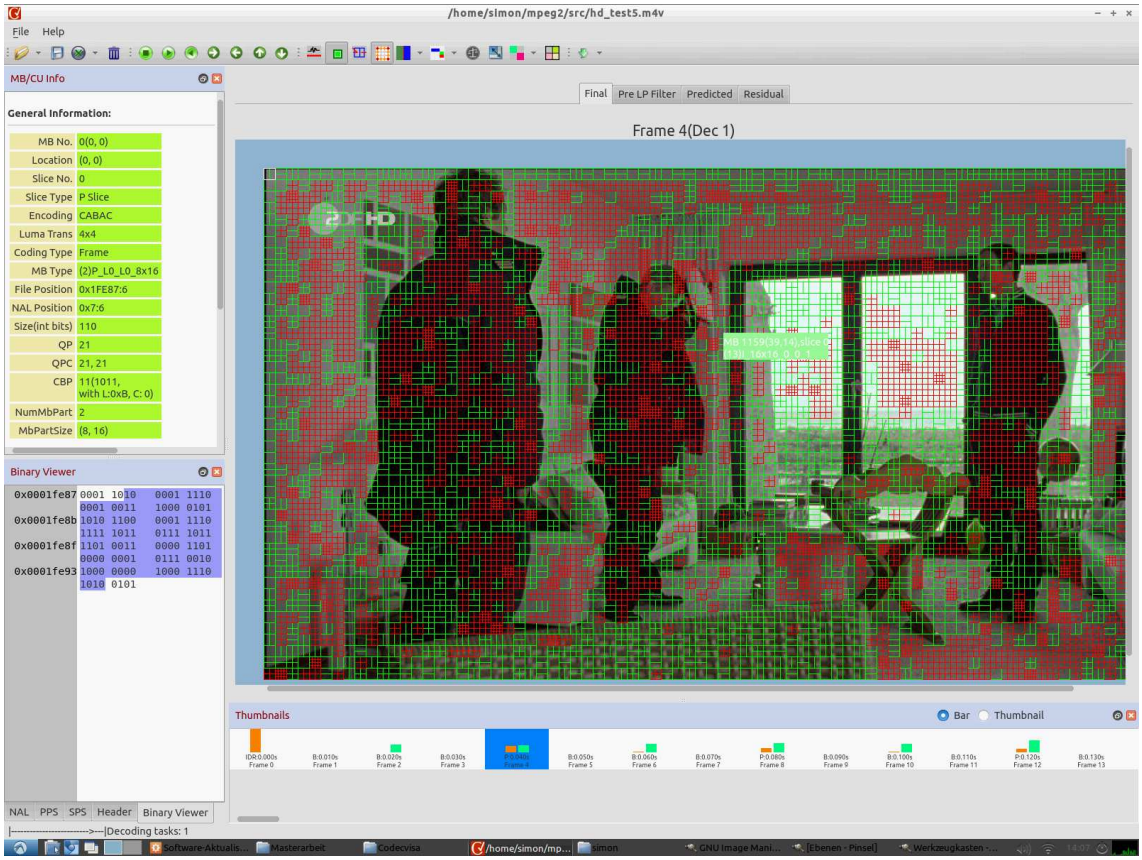


Bild 62: Macroblocktypen in P-Frames (Rot: I-Macroblocks, Grün: P-Macroblocks)

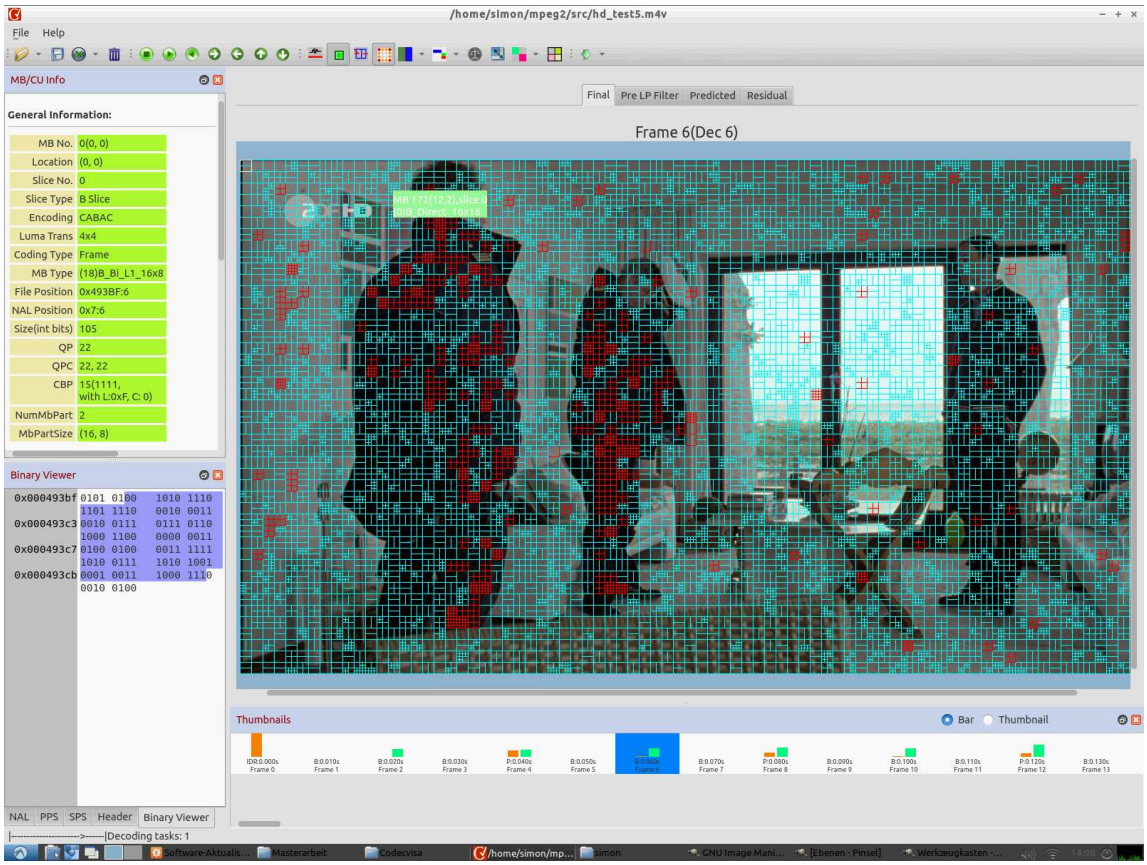


Bild 63: Macroblocktypen in B-Frame (Rot: I-Macroblocks, Cyan: B-Macroblocks)

6.3.1.2 Codierung

Ein weiterer wesentlicher Unterschied liegt in der Codierung der Bilddaten selbst. Hierfür wurden bei MPEG-2 VLCs (Variable Length Code) verwendet, mit einer Tabelle der gültigen Codes und deren jeweiligem Wert, wobei die Länge des Codes von dessen Wahrscheinlichkeit abhängig ist. [I32-1995, Table B-14 und B-15] Bei H.264 werden die Werte CABAC codiert (CAVLC ist eine Option, wird aber in der Praxis nicht verwendet). Bei CABAC handelt es sich um eine arithmetische Codierung, d.h. eine Reihe von Zahlen (hier: Koeffizienten) wird als eine einzige Zahl zwischen 0 und 1 codiert, dieser Bereich wird in Intervalle aufgeteilt. Die Intervalle entsprechen den codierten Werten und die Größe der Intervalle ist abhängig von deren Wahrscheinlichkeit.
Hier eine stark vereinfachte (und entsprechend ungenaue) Erläuterung von arithmetischer Codierung (eine verständliche und genaue Erläuterung (67 Seiten) findet sich in [HPA-2004]):
Es wird eine festgelegte Anzahl an Bits eingelesen, diese wird als Festkommazahl zwischen 0 und 1 interpretiert. Es wird geprüft, in welches Intervall - das ist der decodierte Wert - die Zahl fällt, dessen Basis wird anschließend von der Zahl abgezogen und der Prozess mit einem skalierten Intervall wiederholt. Das Intervall wird um den Kehrwert einer Zweierpotenz skaliert, wenn die codierte Zahl minus die Basis kleiner als das nächstkleinere Intervall ist. Anschließend wird der Prozess wiederholt. Dies kann unbegrenzt fortgesetzt werden und erlaubt, mehrere Zahlen mit nur einem Bit zu codieren. Die Anzahl der Iterationen muss ebenso wie die Bitlänge festgelegt werden.
Adaptive arithmetische Codierung (z.B. CABAC) ändert die Größen der Intervalle abhängig von der decodierten Zahl. Dem geht der Gedanke der Kontextabhängigkeit voraus. Als Beispiel wird angeführt, dass die Wahrscheinlichkeit für ein "u" in einem Wort enorm steigt, wenn diesem ein "q" oder ein "Q" vorausgeht.
Die Intervallbasen müssen bei CABAC auf En- und Decoderseite identisch sein, d.h. wie die Tabellen mit den VLCs bei MPEG-2 in die Hardware eingebaut sein. [Quellen für den gesamten Absatz: HPA-2004 (Kap. 2.2 und Unterkapitel, insbesondere 2.2.3) und WAC-2013.]

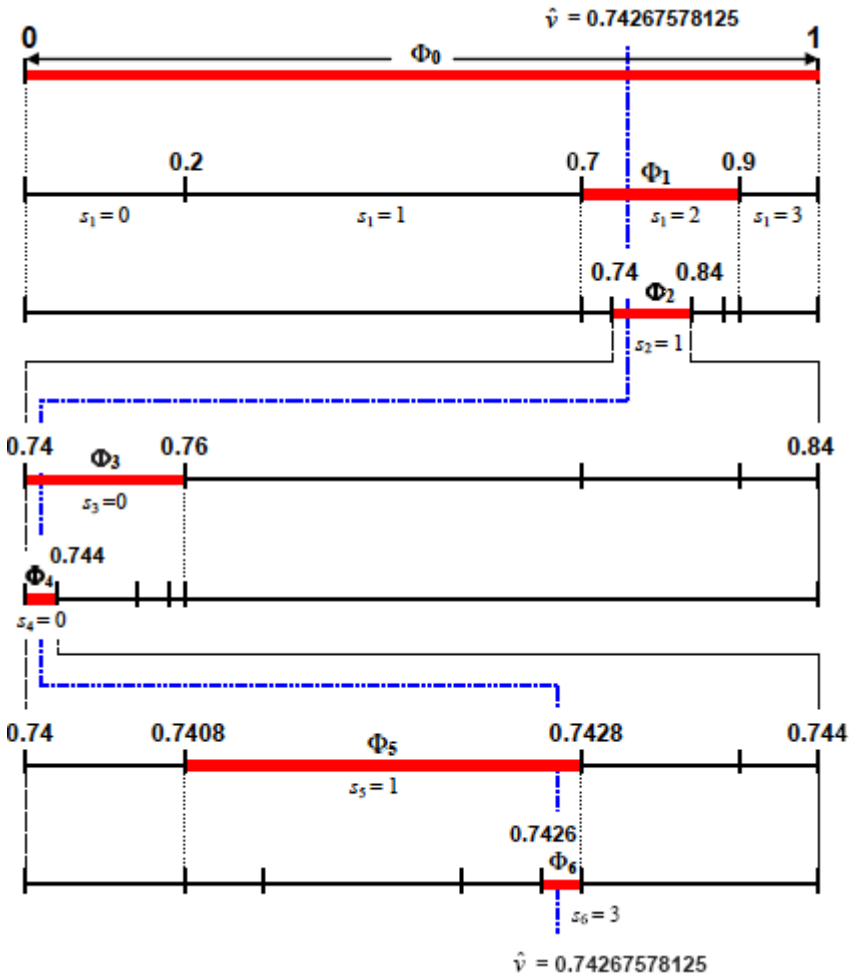


Diagramm 11: Beispiel für Arithmetische Codierung: Im Beispiel werden pro Iteration 2 Bit codiert, zur Codierung wird jedoch oft weniger als 1 Bit benötigt. Der codierte Wert ist in diesem Beispiel 210013 (Basis 4). Aus [HPA-2004].

6.3.1.3 Slice

Die formale Definition einer Slice hat sich im Vergleich zu MPEG-2 nicht geändert. Weiterhin setzt sich eine Slice aus einer beliebigen Menge an Macroblocks zusammen und kann beliebig große Teile des Bildes umschließen. Macroblocks haben keine Adresse mehr (auch nicht implizit über address_increment, wie bei MPEG-2). Jedenfalls konnte ich in Kapitel 7 von [I4A-2004] in keiner Tabelle etwas Entsprechendes finden, lediglich die Adresse des ersten Macroblocks in einer Slice wird im Slice-Header [I4A-2004, Kap. 7.3.3] übertragen, für anschließende Macroblocks gibt es ein mb_skip_flag und einen mb_skip_run für übersprungene Macroblocks. Im Gegensatz zu MPEG-2 gibt es bei Slices in H.264 keine Startcodes mehr, jedoch wird sichergestellt, dass der Anfang byte-aligned ist, sowie am Ende

ein Nullwort (2 Byte 0x00) steht, das ebenfalls byte-aligned ist. [I4A-2004, Kap. 7.3.2.10->7.4.2.10, 7.3.2.8, 7.3.2.9]
Ein weiterer Unterschied zu MPEG-2, wo es Gebrauch war, dass eine Slice so breit wie das Bild ist, ist, dass in H.264, zumindest bei ARD und ZDF, das Bild nur noch aus einer einzigen Slice besteht. (Quelle: h264dec [AH2-2013i] mit ardhd.m4v und hd_test5.m4v.)

6.3.1.4 Bildtypen

Bei H.264 kommen zusätzlich zu den bekannten I-/P-/B- Frames noch SI-/SP- Frames hinzu, welche die Eigenschaft haben, I-Frames ersetzen zu können und dennoch kompakter komprimiert werden können, für TV-Broadcasting aber eher uninteressant sind. (Weiterführende Literatur:[KSP-2003])
Darüber hinaus können B-Frames als Referenz verwendet werden, was erlaubt, mehr B-Frames im Verhältnis zu P-Frames zu verwenden und somit mehr Platz zu sparen. Ebenso ermöglicht H.264 die Verwendung von mehr als 2 Referenzbildern [I4A-2004, Kap.0.5.3, 7.3.3.1, 7.3.3.3, 7.4.3.1 und weitere], was die Komprimierung von schnellen Szenenwechsel auf einem wahlfreien Zugriffsmedium wie einer Festplatte verbessert, für TV-Broadcasting jedoch ungeeignet ist.
Frame-Reordering tritt bei H.264 ebenso wie bei MPEG-2 auf. Unterschiede bei der praktischen Ausführung finden sich im Analyse-Kapitel.

6.3.2 Videostream-Decodierung

Im Vergleich zu MPEG-2 befindet sich der Videostrom bei H.264 in einer eigenen Kapselung, genannt NAL (Network Abstraction Layer). Das bedeutet, dass der Videostrom insgesamt doppelt gekapselt ist: Zum Einen im Transport-Stream, zum Anderen im NAL. NAL-Pakete beginnen mit Start-Codes, ähnlich wie bei MPEG-2, aber mit einem zusätzlichen Null-Byte und es sind weniger Startcodes definiert (nur 12 verschiedene). Nummerierung von Slices ist nicht mehr vorhanden [I4A-2004, Tabelle 7-1], die obersten 3 Bits des Startcode-Wertes sind reserviert bzw. für andere Zwecke verwendet [I4A-2004, Kap. 7.3.1].

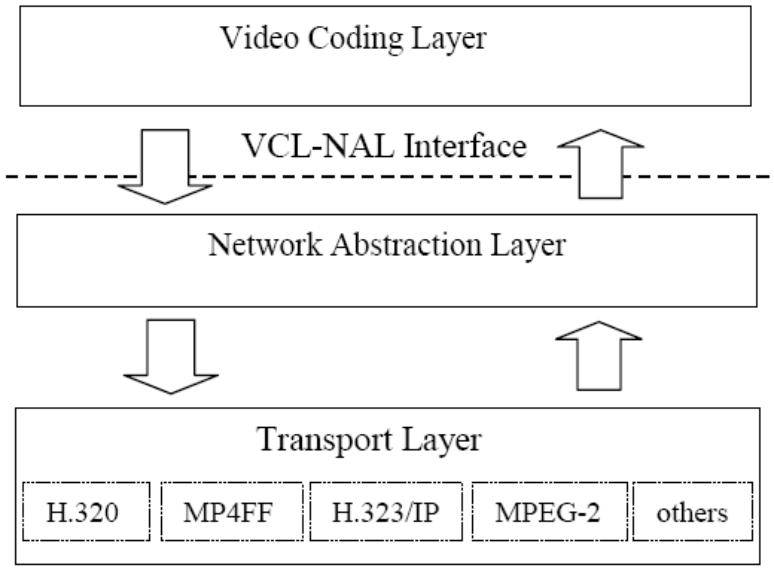


Diagramm 12: Kapselung des H.264-Videostroms. Das Transport Layer (unten) nimmt hier den Wert 13818-1 Transport Stream an. [KER-2006]

Während bei MPEG-2 die Art und Reihenfolge der Daten vor den eigentlichen Bilddaten in der Referenzanleitung klar definiert ist, wird in 14496-10 [I4A-2004] nur die Art und Reihenfolge der Daten auf Slice-Ebene definiert, der Zusammenhang aller übergeordneten Daten ist nicht definiert. Es bleibt also nur, diesen durch Parsen des Aufbaus eines realen Videostroms zu ermitteln. Hierzu dient h264dec.exe [AH2-2013i]. Die folgende Passage wurde mit diesem Programm sowie durch trial-and-error ermittelt (welche der Daten in welcher Reihenfolge müssen in den Videostrom übernommen werden, damit der Decoder diesen abspielt).

Bevor der Decoder mit der Wiedergabe beginnen kann, werden folgende Datenpakete benötigt:

Ein NAL-Unit-delimiter (00 00 00 01 09 hex), ein Sequence-Parameter-Set (00 00 00 01 07 hex), dieses enthält vergleichbare Informationen wie ein sequence_header bei MPEG-2, ein oder mehrere Picture-Parameter-Sets (00 00 00 01 08 hex), diese enthalten zusätzliche, für den Menschen weniger wichtige Daten. Anschließend kommt eine IDR-NAL (00 00 00 01 05 hex). In dieser befinden sich die reinen Bilddaten. Innerhalb dieser existieren keine weiteren Startcodes, die es erlauben, innerhalb eines Bildes zu synchronisieren. Für andere Bildtypen kommt eine Non-IDR-NAL (00 00 00 01 01 hex). Eine IDR-NAL dient der Kennzeichnung für einen Decoder, dass die Decodierung hier beginnen kann.

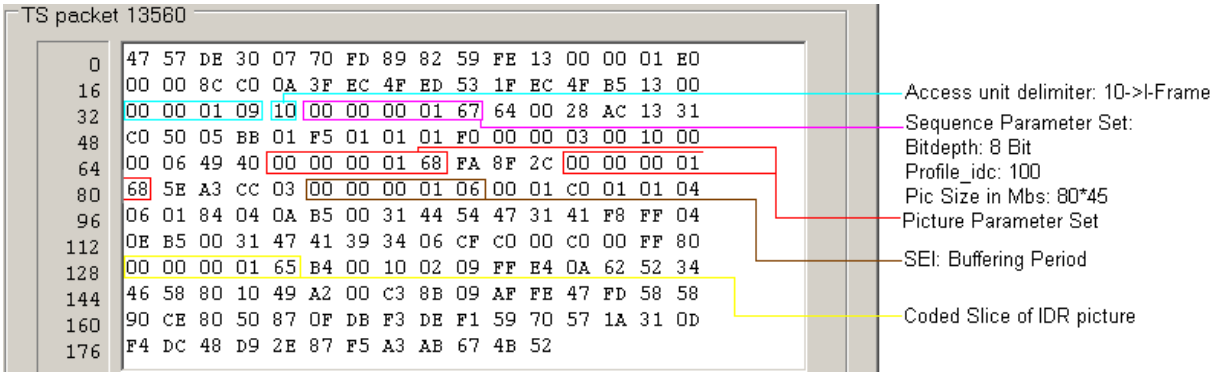


Bild 64: [PDM-2013] mit einer IDR-NAL (ZDF).

6.3.3 Analyse

Die Analyse geschieht beispielhaft anhand von Videoströmen von ARD und ZDF. Verschlüsselte Privatsender sind für diese Arbeit uninteressant.

6.3.3.1 Unterschiede

Auf Einzelbildebene fällt auf, dass Bilder nur aus einer einzigen Slice bestehen. I-PCM-Macroblocks werden nicht verwendet, SI- und SP-Frames sowie Data Partitioning ebenfalls nicht. Keine der in [KER-2006] verwendeten Fehlerkorrekturmaßnahmen wird von den Sendern umgesetzt.
Auf Stromebene verwendet ARD nicht einmal den IDR-NAL Startcode für I-Frames, stattdessen wird ein I-Frame implizit über einen NAL-Unit Delimiter mit primary_pic_type=I (wie auch bei ZDF), gefolgt von einem Non-IDR-NAL-Paket, angezeigt. Dies entspricht nicht dem Standard und ist für manche Decoder verwirrend. In [ADT-2013i] kann man erkennen, dass manche Geräte zum Umschalten von ZDF auf ARD erheblich (~40%) länger brauchen als von ARD auf ZDF, obwohl beide die gleiche I-Frame-Rate haben. Betrachtet man Bild 64 und 65, ist zu beachten, dass die oberen 3 Bits des ersten Bytes nach dem Picture Start Code nicht zur Codierung des NAL-Typs gehören.

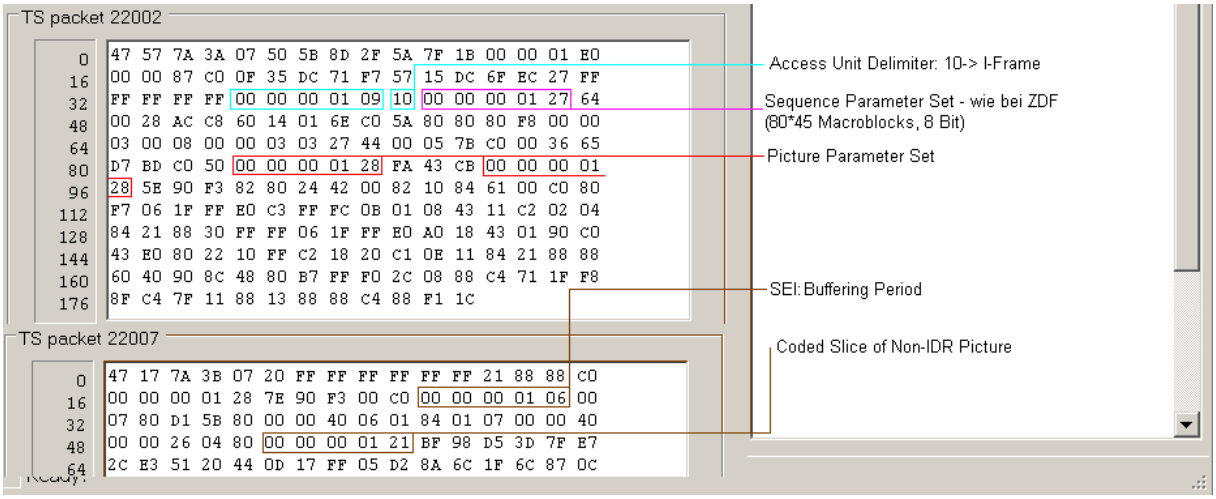
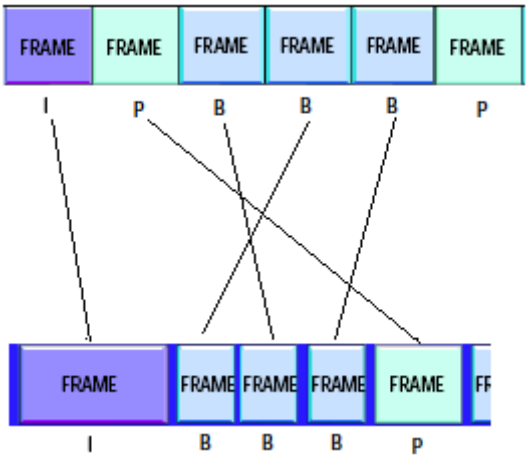


Bild 65: I-Frame von ARD, der als Non-IDR-NAL codiert ist.

Die Frame-Reordering-Reihenfolge ist wie folgt: (hd_test5.m4v als Beispieldatei, Quelle: CodecVisa)

Frame	Adresse	Position im Stream
0 I	0	0
1 B	0x38CB2	3
2 B	0x2F3A4	2
3 B	0x38D0F	4
4 P	0x1FE4D	1



Wenn also der Lesezeiger 0x38D0F erreicht hat, kann erst mit der Wiedergabe begonnen werden, obwohl der I-Frame schon bei 0x1FE4D (als Standbild) dargestellt werden könnte.

Diagramm 13: Ermittelte Frame-Reihenfolge für H-264-Videoströme von ZDF.

ARD verwendet erheblich weniger P-Frames als ARD, nämlich nur einen in der Mitte zwischen zwei I-Frames [AH2-2013i mit dem aus ardhd.ts entpackten HD-Videostrom (PID 6010)], P-Frames kommen damit genauso häufig vor wie I-Frames. Da zum Zeitpunkt dieser Feststellung meine CodecVisa-Lizenz bereits abgelaufen war, kann ich die Auswirkungen dieser Gegebenheit auf das Frame-Reordering nicht mehr ermitteln, aber wenn bereits die ersten B-Frames nach dem I-Frame auf das Vorhandensein eines P-Frames angewiesen wären (was ich nicht vermute, da genannt wurde, dass in H.264 B-Frames als Referenzbilder

verwendet werden können), würde das die deutlich erhöhte Umschaltzeit von ZDF auf ARD auch erklären.[16]

6.3.3.2 Timings

Sowohl ARD und ZDF haben eine I-Frame Rate von einem I-Frame alle 33 Bilder, also einen alle 2/3 Sekunde, was also seltener ist als bei MPEG-2, wo 2 I-Frames pro Sekunde Standard sind. Da bei verpasstem NAL-Unit Delimiter inkl. folgenden Sequence- und Picture Parameter Sets eine Decodierung nicht möglich ist, erhöht sich die Umschaltgeschwindigkeit erheblich (siehe auch Diagramm 2, HDTV). Allerdings wird der Videostrom auf Transportstromseite üblicherweise so formatiert, dass Pakete mit einem I-Frame mit einem NAL-Unit Delimiter beginnen. Dieser sowie die Sequence- und Picture Parameter-Sets und der Startcode der IDR-NAL passen somit in ein einziges Transport Paket (188 Byte). Bei ARD war dies allerdings nicht der Fall, die SEI-Message und der (Non-!)IDR-NAL Startcode fanden sich im nächsten Paket. Dies führt dazu, dass Orderts [AOT-2013i] I-Frames in diesem Strom nicht erkennt.
In [AFT-2013i] für HDSMPCOR.TS habe ich eine Rate von etwa 33772,14 Paketen pro Sekunde ermittelt. Der durchschnittliche Abstand zwischen 2 I-Frames beträgt 22515 Pakete. Ein I-Frame braucht durchschnittlich 3140 Pakete für die Übermittlung, was knapp 14%[17] der Übertragung ausmacht. Dank Frame-Reordering muss der Decoder insgesamt durchschnittlich 5707 Pakete warten, bis er mit der Decodierung beginnen kann. Addiert man diese Zahlen auf das I-Frame-Spacing, erhält man die Worst-Case-Zeiten von 25655 bzw. 28222 Paketen, d.h. etwa 0,76 bzw. 0,836 Sekunden Worst-Case-Umschaltzeit sowie einer Best-Case-Umschaltzeit von 0,17 Sekunden (mit Frame-Reordering). Diesen Timings muss in den meisten Fällen noch die physische Umschaltzeit des Tuners, was bei Kabelempfang mit durchschnittlich 0,35 Sekunden ermittelt wurde, siehe Timings in Analyse bei MPEG-2 (Kapitel 3.1), hinzuaddiert werden. Dass der Best Case von HDTV besser ist als der Best Case von SDTV (0,265s) liegt daran, dass erstens das Frame-Reordering um einen Frame günstiger ist, d.h. ein Frame weniger lang gewartet werden muss, bis mit der Bewegtwiedergabe begonnen werden kann, und zweitens, dass sich die Wartezeiten durch die doppelte Framerate halbieren.

---

[16] Der Betrachter könnte vermuten, dass die technischen Ausführenden bei ARD und ZDF über einen Mangel an Fachkenntnissen verfügen

[17] (3*3140)/(2*33772,14). 1 I-Frame alle 2/3 Sekunde macht 3 I-Frames alle 2 Sekunden.

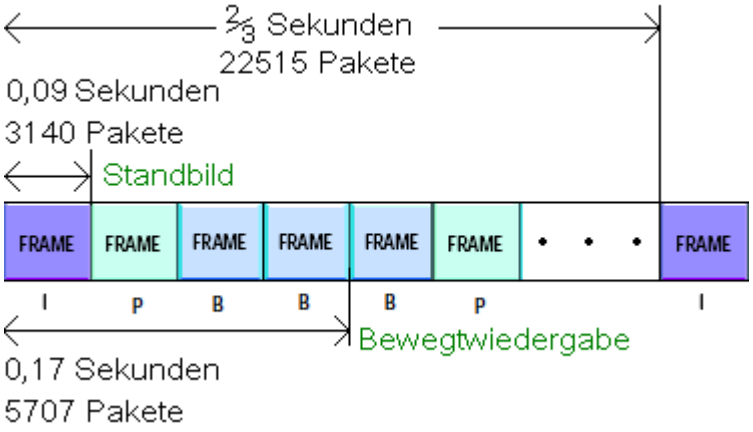


Diagramm 14: Ermittelte Durchschnittszeiten für die Übertragungsdauer relevanter Streambestandteile bei HDTV (ZDF). Werte aus [AFT-2013i].

6.3.4 Konzept - Anpassung

Aufgrund von Schwierigkeiten in der Implementierung der Frame-Wiederherstellung wird in diesem Kapitel die Sichtbarmachung der Zwischenbilder zuerst besprochen.

6.3.4.1 Sichtbarmachung der Zwischenbilder - TS2M2V Mode 6

Damit ein Decoder eine Bildsequenz zu decodieren beginnt, muss, wie in Kapitel 6.3.2 besprochen, ein NAL-Unit Delimiter, gefolgt von einem Sequence Parameter Set und zwei Picture Parameter Sets einer IDR-NAL (den meisten Decodern genügt auch eine Non-IDR NAL) vorausgehen. Das Vorhandensein von Sequence- und Picture Parameter Sets scheint zu genügen, um den Decoder dazu zu bringen, die Sequenz ab diesem Bild (unabhängig vom Typ) zu decodieren.
Entsprechend geht TS2M2V wie folgt vor: Feststellung des Bildtyps des ersten (vollständigen) Bilds im Strom, Kopieren eines I-Frame-Headers vom NAL-Unit Delimiter (der anhand des Bildtyps angepasst wird) bis zum, aber nicht einschließlich des IDR-NAL Startcodes. Anschließend wird zum Non-IDR-NAL Startcode zurückgesprungen und das Entpacken der Bilddaten von dort an fortgesetzt.

6.3.4.2 Analyse und Erklärung an Beispielbildern

Die Ergebnisse sind durchaus ähnlich wie bei Mode 2 (MPEG-2), siehe Kapitel Intra-Block-Interpolation, jedoch durch die Nichtdarstellbarkeit von Intra-Macroblocks in Non-Intra-Frames ist es unmöglich, Farben und Helligkeiten wiederherzustellen; man sieht also weiterhin nur Konturen und Bewegungen, d.h. die Ergebnisse sind von niedrigerer Qualität als bei MPEG-2.
Unterschiedliche Decoder produzieren jedoch unterschiedliche Ergebnisse. Der Referenzdecoder schiebt leider alle decodierten Bilder in ein einziges riesiges YUV-Bild und es besteht für mich keine Möglichkeit, die Einzelbilder betrachtbar auszulesen. Zudem verweigert der Referenzdecoder und auch der VLC Media Player die Wiedergabe von mit Mode 6 entpackten Videoströmen.
CodecVisa produziert ähnliche Bilder wie der MPEG-2 Referenzdecoder, nämlich dunkelgrüne Bilder mit hellgrünen Konturen.

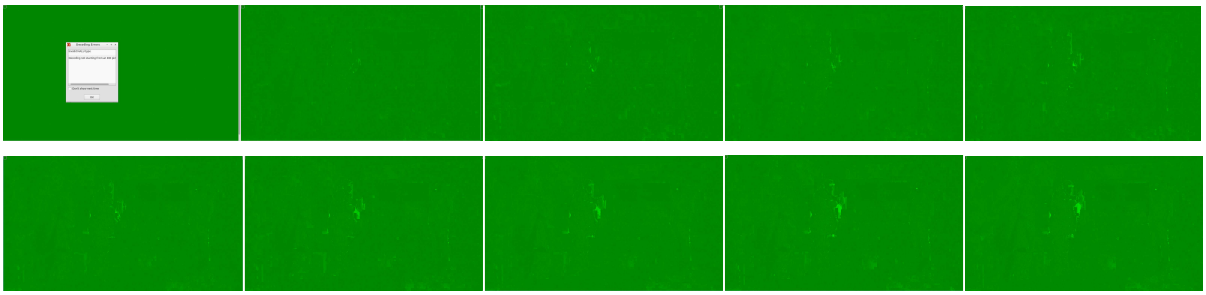


Bild 66-76: Bildsequenz aus hd_test5.m4v (hdsmpcor.ts nach PID 6110 entpackt). Die Fehlermeldung von CodecVisa in Bild 66 („Decoding not starting from an IDR-picture“) habe ich absichtlich im Bild verbleiben lassen. Die Meldung erscheint auch beim Versuch, einen sauber entpackten Stream von ARD zu decodieren.

Der Gnome-Mediaplayer zeigt, welche qualitativen Sprünge lediglich durch unterschiedliche Initialisierung möglich sind, d.h. ein nicht vorhandenes Referenzbild mit Grau statt mit Dunkelgrün zu initialisieren (siehe Bild 77). Da der Gnome Media Player ein Overlay für Videodateien verwendet (Videoströme werden direkt zur Grafikkarte, also am Bildschirmspeicher vorbei, geschrieben), sodass ein Screenshot lediglich ein Rechteck mit voll transparenten Pixeln zurückliefert. Der Gnome Media Player verfügt zudem nicht über eine eingebaute Screenshot-Taste, die es erlaubt, das momentan betrachtete Bild in einem Standard Bildcontainer (z.B. JPG, BMP, PNG) zu speichern. Der VLC Media Player hat eine

solche Funktion. Entsprechend mussten die Bilder vom Gnome Media Player mit einer gewöhnlichen Digitalkamera vom Bildschirm abfotografiert werden.

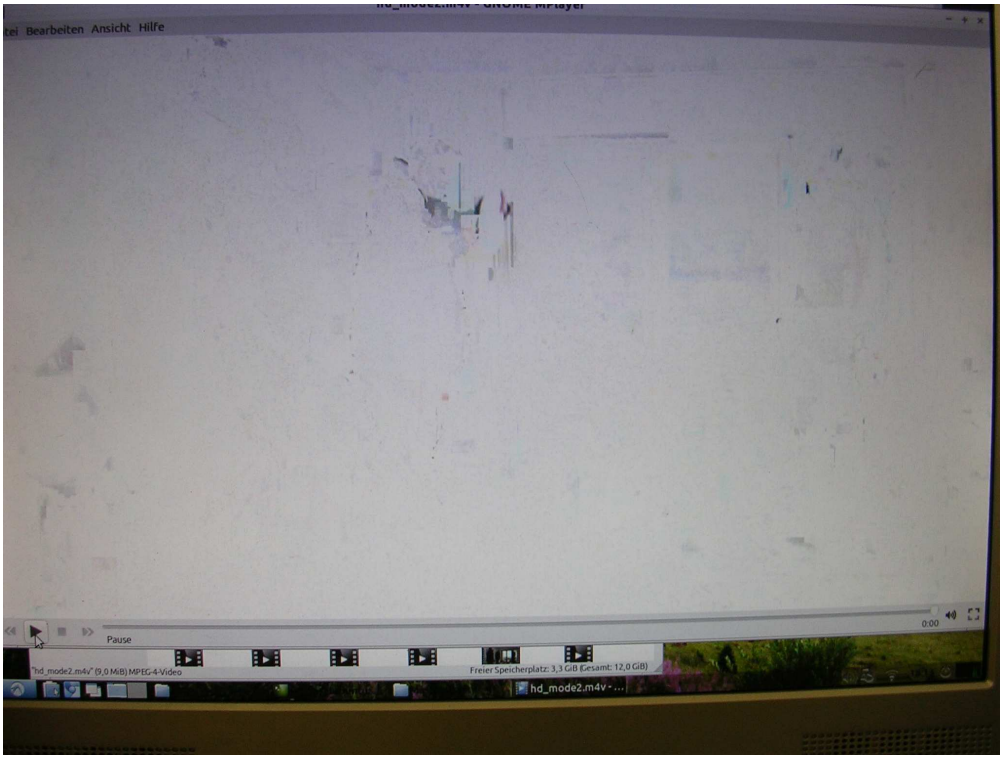


Bild 77: Derselbe Videostrom, mit Gnome Media Player. Da kein Screenshot möglich war, habe ich den Bildschirm mit einer Digitalkamera fotografiert.

Auf diesem Bild erkennt man deutlich mehr, sogar Farben sind vorhanden, so sind Gesichter tatsächlich rosa, Konturen der Gesichter (Mund, Nase, Augen, Haare) sind erkennbar, sodass das menschliche Gehirn seine Stärke bei der Gesichtserkennung spielen lassen kann (siehe auch Nahaufnahme in Bild 78).

Bild 78: Nahaufnahme von Bild 77.

Bild 79: Erster I-Frame im verwendeten Videostrom

6.3.4.3 I-Frame-Wiederherstellung

Um kurz das Ergebnis vorweg zu nehmen: Eine I-Frame-Wiederherstellung bei H.264 ist bei der derzeitigen Sendesituation nicht möglich.
Wie in Kapitel 6.3.1: Grundlagen bereits geschildert, setzen weder ARD noch ZDF irgendwelche der optionalen Fehlerkorrekturmaßnahmen von H.264 ein, sondern verlassen sich ausschließlich auf die fest eingebaute Forward-Error-Correction. Der Videostrom der

ARD ist, aufgrund der fehlerhaft markierten I-Frames, nicht einmal standardkonform (Quelle: [AH2-2013i] mit ardhd.m4v). Im Feldversuch (Antennenkabel leicht herausgezogen, Gerät: PC mit DVB-C Karte) konnte erkannt werden, dass bei auftretenden Datenfehlern Bildfehler recht gut verschleiert werden konnten. Man konnte zwar erkennen, dass Bildfehler auftraten, aber die Wiedergabe wurde bei einer relativ geringen Menge (schätzungsweise etwa hundert Bit pro Bild) von Fehlern normal fortgesetzt. Erst bei deutlich mehr Datenfehlern wurde die Wiedergabe abgebrochen. Es konnte jedoch beobachtet werden, dass die Wiedergabe erst beginnen konnte, wenn ein I-Frame fehlerfrei übertragen wurde. Bei einem DVB-C HD-Fernseher war das nicht der Fall, dieser konnte auch stark gestörte Videoströme wiedergeben. Allerdings war darauf nicht viel zu erkennen.
Da H.264 codierte Bilder nur aus einer einzelnen Slice bestehen, ist das Konzept aus Kapitel 4 hier nicht brauchbar. Der Versuch, innerhalb einer Slice mit dem Decodieren zu beginnen, wurde bei MPEG-2 als unverhältnismäßig verworfen, da im besten Falle nur ein um 16 Pixel höheres Bild gewonnen werden könnte.
Entsprechend musste ein Konzept ersonnen werden, das eine Synchronisation innerhalb einer Slice ermöglicht.
Diesem förderlich wäre das Wissen, dass Daten byte-aligned sind, da der Decodierung-durch-Trial-and-Error-Aufwand dadurch geachtelt würde.
Byte-aligned sind Videoströme am Anfang einer Slice. Am Ende einer Slice stehen zudem zwei Nullbytes, sodass es zumindest eine gewisse Möglichkeit gibt, Slices zu erkennen und mitten im Bild – an einer Slicegrenze – mit der Decodierung zu beginnen. Da ein Bild nur eine Slice hat, besteht diese Möglichkeit nicht. Eine andere Möglichkeit, Byte-Alignment zu erzwingen, sind Start Code Emulation Prevention Codes. Hier wird geprüft, ob der Encoder-Output einen künstlichen Startcode erzeugt, und wenn ja, wird die Byte-Sequenz 00 00 03 (hex) eingefügt. Der Decoder erkennt und entfernt diese Sequenz. Jedoch konnten im Inneren eines I-Frames keine solchen Codes gefunden werden (10 I-Frames getestet), siehe auch [AF2-2013i]. Entsprechend ist diese Methode nicht anwendbar, bzw. unverhältnismäßig, da derartige Ereignisse zu selten auftreten.

Der zweite Versuch betraf das Rückwärtsauslesen von Exp-Golomb-Codes. Exp-Golomb-Codes werden in H.264 verwendet, um Parameter mit variabler Länge zu codieren. Sie bestehen aus einem unären Teil und einem Datenteil. Im unären Teil wird eine Folge aus Nullbits, gefolgt von einem Einsbit, gesendet. Die Anzahl der Nullbits entspricht der Länge der nachfolgenden Datenbits. Es wurde sogar ein (Pseudocode) Algorithmus zum

Rückwärtsauslesen von Exp-Golomb-Codes entwickelt [reverse_expgolomb.cpp], der auch Rücksicht darauf nimmt, dass Daten im Datenteil von Exp-Golomb-Codes ebenfalls als vollständige Exp-Golomb-Codes ausgelesen werden können.
Da jedoch leider die Interpretation der codierten Werte von dem Wert vorhergehender Werte abhängig ist, wurde die Methode verworfen.

Der dritte Versuch ist eng mit dem zweiten verwandt. Diesmal wurde versucht, Exp-Golomb-Codes "aufs Geratewohl" zu lesen. Leider gibt es keine ungültigen Exp-Golomb-Codes. Dass etwas schiefgegangen ist, merkt man erst, wenn man am Ende der Datei, aber nicht am Ende des gegenwärtigen Codeworts ist.
Es wurde ein Programm geschrieben, das an einer beliebigen Adresse an einem beliebigen Bit-Offset eine Datei decodieren kann, sowie eine Serie durchgeführt, bei denen alle Offsets von 0-7 Bit an derselben Adresse verwendet werden.
Das Ergebnis steht in [ATE-2013i].
Man erkennt, dass Exp-Golomb sich erstaunlich schnell, nämlich innerhalb weniger Codes, mit dem Bitstream synchronisiert. Wenige, längere Codes, z.B. {9, 2, 8}, werden in mehrere kürzere, z.B. {4, 5, 3, 0}, zerteilt, danach ist der Bitstrom synchronisiert.
Eine Synchronisation auf Bitstreamebene kann man eigentlich nur verhindern, indem man viele große Zahlen sendet, die eine Zweierpotenz minus eins sind {z.B. 31, 63, 127...}.
Das gewöhnliche, positive Exp-Golomb ist durch folgende Formel definiert:

$$codeNum = 2^{leadingZeroBits} - 1 + read\_bits(leadingZeroBits)$$

Formel 4: Decodierung des Wertes eines Exp-Golomb-Codes. Aus [I4A-2004], Kapitel 9.1.

Eine Zahl, die einer Zweierpotenz minus eins entspricht, wird entsprechend 0...010...0 codiert. Befindet sich der Lesezeiger auf oder hinter der mittleren 1, wird er für den nächsten Code eine größere Zahl lesen, als tatsächlich codiert ist, der Lesezeiger wird hierbei also näher an die mittlere 1 des übernächsten Codes herangeschoben, wird dann also zwangsweise weniger Bits einlesen, als codiert sind. Beim anschließenden Code werden also wieder mehr Bits eingelesen und so beginnt das Spielchen von vorne.
Bei Videokompression sind jedoch folgende Einschränkungen gegeben:

1. Der Encoder ist optimiert, möglichst viele kleine Zahlen zu produzieren.
2. Statistisch gesehen haben Bilddaten eher Zufallscharakter.

Aus dem beobachteten Verhalten (siehe Datenreihen in [ATE-2013i]) lässt sich eine Obergrenze für die Synchronisation auf Bitstream-Ebene innerhalb eines Macroblocks setzen. Das Problem hierbei ist jedoch, dass die eigentlichen Bildinformationen CABAC codiert sind, und CABAC nicht auf Exp-Golomb basiert. In [I4A-2004, Kap. 7.3.4 sowie 7.3.5] steht zudem, dass CABAC nicht nur für die Bilddaten selbst, sondern auch für die anderen Parameter auf Slice-Ebene verwendet werden kann. Die schnelle Selbstsynchronisationsfähigkeit von Exp-Golomb sowie die Fähigkeit, diese Codes rückwärts auszulesen, stellt somit keine zuverlässige Möglichkeit mehr dar, die Decodierung mitten im Bild starten zu können.
Da über die innere Arbeitsweise von CABAC nur wenig offen liegt, das sich leicht erschließt (nach nur mehrtägigem Studium), können über den Aufbau der Bilddaten in H.264 nur Mutmaßungen angestellt werden. [HPA-2004] legt nahe, dass die Anzahl der gelesenen Bits fest und die Anzahl der codierten Werte pro fester Bitlänge in jeden CABAC-Codewort unkomprimiert übermittelt wird. Denkbar ist selbstverständlich auch, dass die Anzahl der codierten Werte fest ist und die Anzahl der zu lesenden Bits unkomprimiert übermittelt wird, oder dass beides unkomprimiert übermittelt wird. Bestätigt werden kann keine dieser Theorien.
Einzige Hilfe ist, dass [HPA-2004, Kap.1.6.6, Fig. 1.7] undefinierte Bereiche im Intervall vorschlägt, sodass man am Auftreten einer Zahl, die in ein solches Intervall fällt, einen Fehler im Datenstrom erkennen kann. Eine Synchronisation via Trial-and-Error halte ich für zu aufwändig.

Als vierte Möglichkeit habe ich den Macroblock Type (mb_type) gehandelt, da für mb_type fest vorgegebene Codemuster (14496-10, Tabelle 9-26) vorhanden sind, auch wenn diese Tabelle aufgrund des Alters des Dokuments unvollständig ist. CodecVisa zeigt den Scope (Start- und Endadresse) eines Macroblocks in der Binärdatei bitgenau an, d.h. man kann sehen, wo in der Datei selbst ein Macroblock beginnt und endet, und dann beispielsweise mit einem Hex-Editor Bits ändern. Ich habe in diesem Datenstrom für mehrere Macroblocks nach dem Codemuster gesucht, welches dessen mb_type entsprach, bin aber an Stellen fündig geworden, an denen der Wert nicht vorkommen sollte. Bei einem Macroblock trat die Bitsequenz zweimal auf, und als ich dann einen Macroblock fand, bei dem die Sequenz nicht auftrat, wusste ich, dass mb_type anders codiert ist, als in der Tabelle angegeben. Ein Versuch, die Daten mehrerer Macroblocks zu korrelieren, im Wissen, dass mb_type der erste Wert im Macroblock ist, scheiterte auch. Die naheliegendste Erklärung ist, dass mb_type

zusammen mit einigen anderen Werten CABAC-codiert ist. Eine andere Möglichkeit wäre, dass die von CodecVisa ermittelten Grenzen im Bitstrom nicht stimmen. Hierfür würde sprechen, dass beobachtet wurde, dass sich zwei Macroblocks an einer Stelle 6 Bits teilen.

Als letzten Versuch habe ich tatsächlich Bitfehler in die Datei eingebaut, um festzustellen, ob CABAC ähnliche Einschwingeigenschaften hat wie Exp-Golomb, d.h. wie lange es dauert, bis der Decoder wieder mit dem Videostrom synchron ist. Ich habe hierzu einmal einige Bitfehler in den ersten Macroblock eingebaut. Das Ergebnis war, dass CodecVisa und der Gnome Media Player die komplette Group Of Pictures (alle 33 Bilder bis zum nächsten I-Frame) verworfen haben. Dann habe ich ein Bit in einem Macroblock in der Bildmitte gekippt. Das Ergebnis war bei CodecVisa dasselbe, Gnome Media Player spielte die Bildsequenz zwar ab, jedoch waren sämtliche Macroblocks nach dem Bitfehler derart fehlerhaft, dass in der unteren Bildhälfte kein erkennbarer Bildinhalt mehr zu finden war. Das Ergebnis befindet sich in Bild 80.

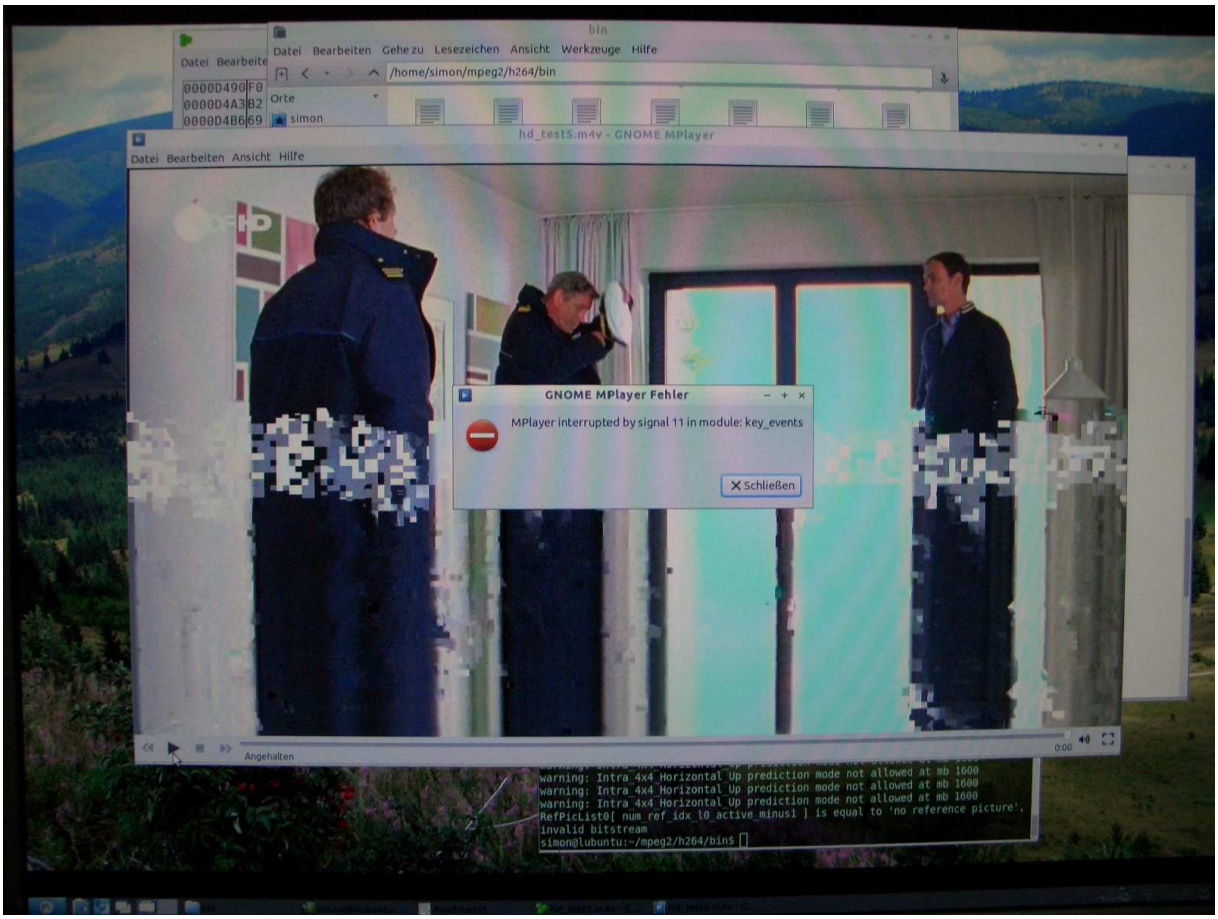


Bild 80: I-Frame aus hd_test5.m4v, mit einem gekippten Bit in Macroblock Nummer 1600. Man sieht auch, dass der Gnome Media Player dieselbe Verschleierungsstrategie wie der VLC Media Player (siehe Bild 12) verwendet.

7. Weitere Optimierungsmöglichkeiten

Die im Hauptteil der Arbeit vorgestellten Methoden zur Beschleunigung des Programmwechsels stellen nicht die einzige Möglichkeit dazu dar.
In diesem Kapitel werden nun einige weitere Methoden vorgestellt, die auch unabhängig von den in Kapitel 3 und 4 vorgestellten Methoden, oder als Ergänzung verwendet werden können. Dieses Unterkapitel ist in zwei Teile unterteilt, nämlich passive und aktive Methoden. Manche der passiven Methoden benötigen eine Umstellung (zukünftiger) Decoderchips, die in Geräten verbaut werden, bzw. im Falle der Frame-Wiederherstellung genügt bei ausreichend leistungsfähigen Geräten ein Firmware-Update. Aktive Methoden bedürfen einer Änderung des Videostroms seitens des Senders, hier ist also deutlich mehr Überzeugungsarbeit zu leisten, da Änderungen nicht alleine die Käufer von Neugeräten, sondern sämtliche Fernsehzuschauer betreffen, selbst, wenn anzunehmen ist, dass diese bzw. deren Hardware keinen, zumindest keinen negativen, Unterschied bemerken.

7.1 Passiv

Die hier vorgestellten Möglichkeiten können sowohl auf MPEG-2 als auch auf H.264 angewendet werden. Die ersten beiden betreffen die Empfangssituation, eine Änderung des Betriebsverhaltens ist seitens des Systemprozessors der Empfangsgeräte zu erzielen, die dritte Möglichkeit muss im (häufig externen) Decoderchip erledigt werden. Hierzu ist zu erwähnen, dass die Decodierung/Dekomprimierung von Videoströmen häufig nicht vom Systemprozessor, sondern von externen digitalen Signalprozessoren erledigt wird.

7.1.1 Intra-Bouquet

Wie in den Grundlagen geschildert, befinden sich üblicherweise mehrere Sender in einem Bouquet, abhängig von der Empfangsart (bei DVB-T üblicherweise weniger als bei -S und -C) und den Begleitumständen (befindet sich ein HDTV-Sender im selben Strom?). Beim Umschalten zwischen zwei Programmen in einem Stream bemerkt man schon im Normalfall eine Reduzierung der Umschaltzeiten (siehe Diagramm 2, [timings.xls]). Dennoch ignorieren Decoder Pakete aus anderen Videostreams gänzlich, andernfalls würde beim Umschalten

zwischen zwei Programmen in einem Transport Stream kaum eine wahrnehmbare Umschaltzeit stattfinden.
Dies ist die erste Optimierungsmöglichkeit. Rapide steigende Rechenleistungen im embedded-Bereich, insbesondere die bereits vorhandene, höhere Rechenleistung von HD-Digitalreceivern, die ja auch SD-Inhalte decodieren können, ermöglichen die parallele Decodierung von mindestens einem weiteren Programm im Stream.
Diese Vorgehensweise eignet sich nicht, wenn der Benutzer die Programme so angeordnet hat, dass nie zwei Programme im selben Transport Stream aufeinander folgen. Auch die im nächsten Unterkapitel vorgestellte Methode ist nutzlos, wenn, statt zu "zappen", Programmnummern direkt eingegeben werden. Dies ist jedoch ein Verhalten, das von Nicht-"Zappern" weniger an den Tag gelegt wird. (wenn ein "Zapper" zwei interessante Programme entdeckt hat, die auf unterschiedlichen Frequenzen senden, oder ein Programm verfolgt, auf dem zur Zeit Werbung läuft, findet dieses Verhalten statt. Letzteres auch oft bei Nicht-"Zappern".)
Ein Flussdiagramm für dieses optimierte Verhalten findet sich in Diagramm 15.

Eine Anpassungsmöglichkeit, wenn nicht ausreichend Rechenleistung zur Verfügung steht, wäre, nur die I-Frames fremder Programme zu decodieren und beim Umschalten beispielsweise eine Überblende zwischen beiden Programmen zu realisieren.
Der nach dem Umschalten zur Verfügung stehenden Zeit für die Decodierung des nächsten Senders sind durch die Umschaltgeschwindigkeit des Betrachters Grenzen auferlegt.
Die Schwarzzeiten (Zeit ohne Bild) zwischen den Programmwechseln können mit diesem Verfahren um die Verweildauer auf dem gegenwärtig laufenden Programm reduziert werden, insbesondere, wenn mehr als ein Tuner zur Verfügung steht (siehe nächstes Unterkapitel). Geht man von einer minimalen Verweildauer von 0.2 Sekunden, was der eines ungeduldigen "Zappers" entspricht (und sich mit den 0.15 Sekunden für die reine Objektwahrnehmung von Dr. Dr. Reiner Beck [RBV-2013] decken, wenn 0.05 Sekunden (massiv parallele) Verarbeitung stattgefunden hat), und einer durchschnittlichen Umschaltzeit von 0.5 Sekunden (bei 2 I-Frames pro Sekunde) aus, sinkt die Schwarzzeit auf 0.3 Sekunden.

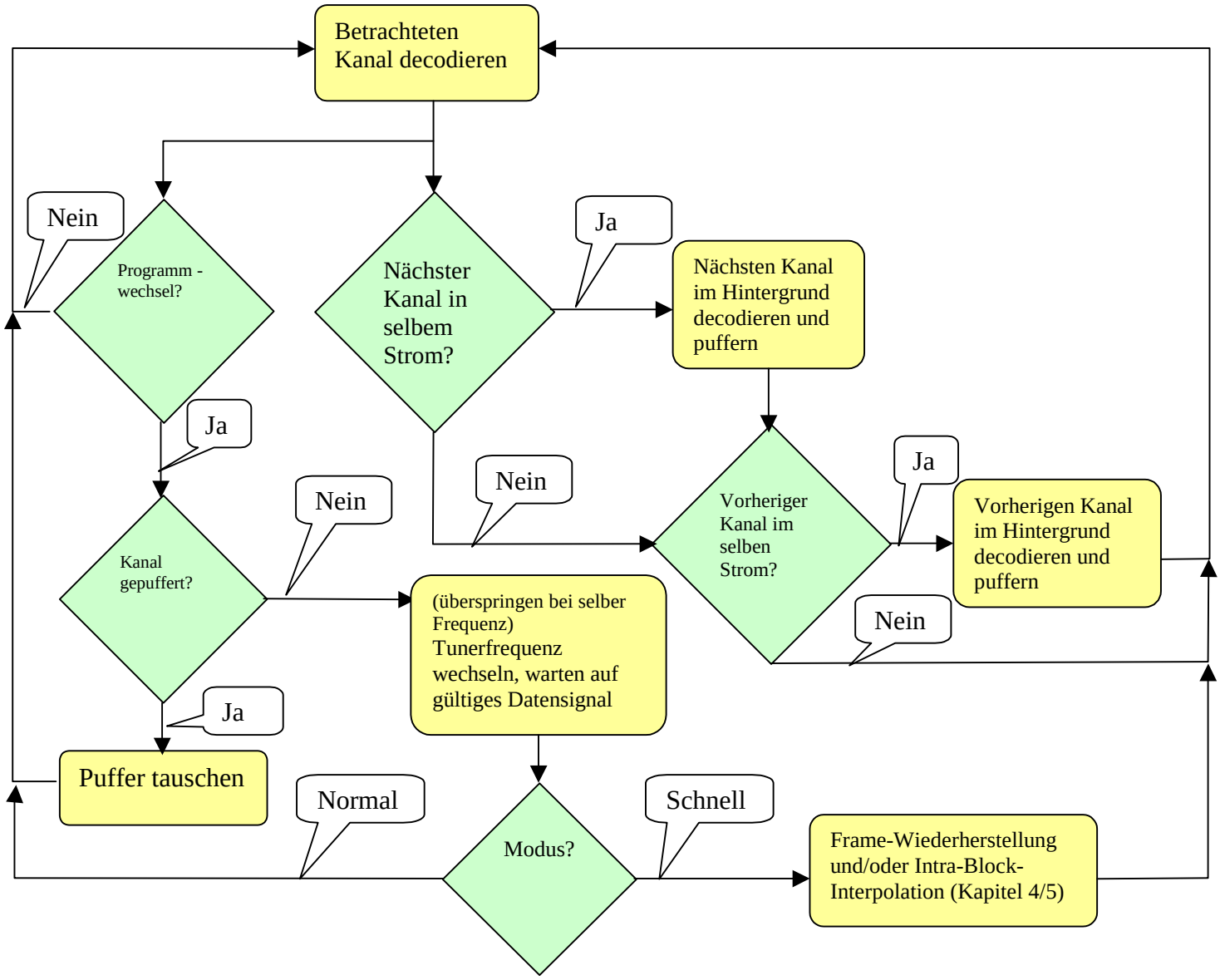


Diagramm 15: Flussdiagramm für optimiertes Umschalten im selben Bouquet.

Durch Verbindung dieser Methode mit der Frame-Wiederherstellung oder der Intra-Block-Interpolierung kann dieser Wert bis auf Null reduziert werden, allerdings muss man davon ausgehen, dass die Intra-Block-Interpolierung große Mengen von Rechenzeit verbraucht, und dass die erzeugten Ergebnisse keine vollwertigen Bilder darstellen. Dennoch kann die Methode des Decodierens anderer Programme im Hintergrund die Umschaltzeit weiter reduzieren helfen und außerdem der Intra-Block-Interpolierung mehr Zeit geben, um bessere Ergebnisse zu produzieren.

7.1.2 Multi-Tuner

Eine Erweiterung des eben vorgestellten Verfahrens stellen Geräte mit Twin- oder Multi-Tuner dar. Bei Fernsehgeräten sind das häufig Multi-Wege-Tuner, also z.B. ein Tuner für Satellitenempfang, einer für terrestrischen und einer für Kabelempfang. Dennoch gibt es Geräte mit mehreren Tunern für dasselbe Empfangssystem, z.B. häufig in Festplattenrekordern, die erlauben, ein Programm aufzunehmen und gleichzeitig ein zweites zu betrachten. In Fernsehgeräten sind diese für Bild-in-Bild-Operation oder für das Aufnehmen auf USB-Datenträger gedacht. Wenn jedoch keine der gedachten Verwendungsmöglichkeiten zur Zeit genutzt wird, können zusätzliche Tuner genutzt werden, um die im Senderspeicher umliegenden Programme im Hintergrund zu empfangen und zu decodieren.
Eine beispielhafte Vorgehensweise findet sich leicht in einer nur geringfügigen Veränderung des Flussdiagramms aus dem vorangegangenen Unterkapitel. Ein Gespräch mit einem Unitymedia-Mitarbeiter am 7.8.2013 brachte hervor, dass diese Methode bei Unitymedia-Receivern bereits eingesetzt wird.

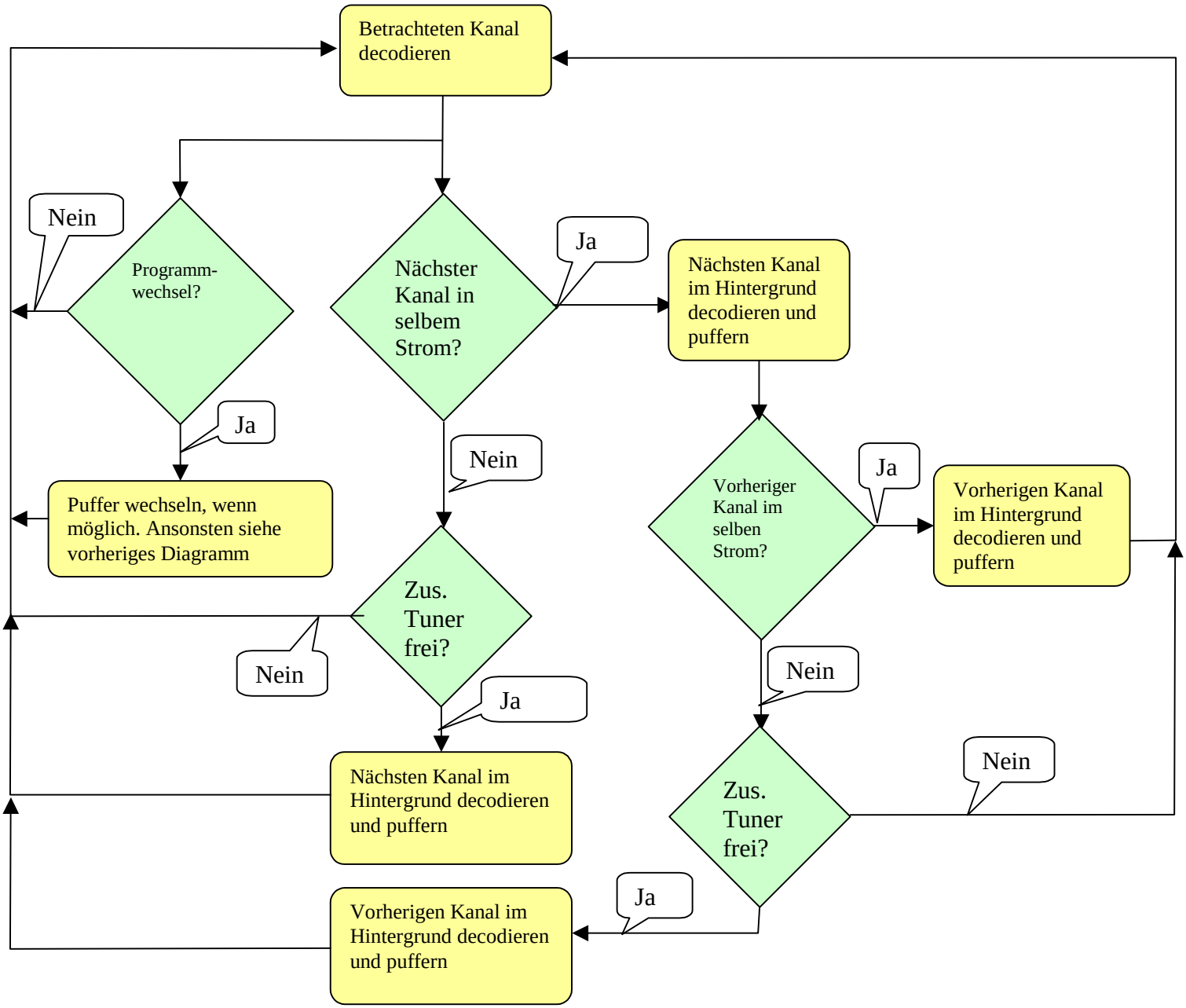


Diagramm 16: Flussdiagramm für optimiertes Umschalten in unterschiedlichen Bouquets bei Geräten mit Multi-Tuner.

Eine Ergänzung für Multi-Wege-Tuner ist der Mehrfachempfang desselben Senders über verschiedene Wege. Dies ermöglicht, zur Decodierung im Hintergrund die Kopie des Senders, die über den momentan nicht benutzten Tuner empfangen werden kann, heranzuziehen, erfordert jedoch auch, dass dieser Tuner überhaupt ans Empfangsnetz angeschlossen ist. Ein gutes Beispiel ist der Kabelempfang. Im Unitymedia-Netz (getestet in Griesheim) ist ein Angebot aus teilweise identischen Sendern (ARD, ZDF, eine Reihe Dritter Programme sowie die bekannteren öffentlich- rechtlichen Sender) über 3 Wege über verschiedene Tuner zu empfangen: In HD (nur ARD, ZDF und ARTE senden unverschlüsselt) und SD über DVB-C (unterschiedliche Frequenzen) und analog. Kommt nun noch DVB-T dazu, sind das 4 Wege. Bei DVB-T sowie DVB-S alleine gibt es jedoch nur eine Empfangsmöglichkeit, wenn nur ein entsprechender Tuner eingebaut ist.
Da bei externen Decoderchips anzunehmen ist, dass sie nicht mehrere Streams gleichzeitig decodieren können (außer bei Geräten, die eigens dafür gebaut wurden), wird sich das Decodieren anderer Sender im Hintergrund nicht durchführen lassen. Wenn ein Signal von einem Sender jedoch nicht nur auf unterschiedlichen Kanälen, sondern auch mit unterschiedlichen Codecs ausgestrahlt wird, wäre denkbar, dass eine Decodierung im Hintergrund selbst bei Decoderchips, die nur für einen Kanal vorgesehen sind, ermöglicht werden kann.[18]
Ein Ersatz, den man bei Einkanalempfang verwenden kann, wäre das Puffern von I-Frames im Arbeitsspeicher, da Decoder heutzutage immer mehr Leistung und Speicher zur Verfügung haben. Dieses Verfahren benötigt nur Speicher und relativ wenig CPU-Leistung, da der Transport Stream nur auf Pakete mit einer weiteren PID abgesucht werden muss, diese entpackt und gepuffert werden müssen, was ein Prozess ist, der in Linearzeit (O(N)) abläuft.

7.1.3 Beschleunigung der Anzeige

In den Grundlagen wurde angedeutet, dass die früheste Bild-Updatemöglichkeit nach der Decodierung eines vollständigen Macroblocks möglich ist. Die Vorgehensweise eines konventionellen Decoders gibt ein Bild auf dem Bildschirm aus, wenn der letzte Macroblock empfangen und decodiert wurde. Da I-Frames 27% der Zeit zur Übertragung brauchen, aber nur 8% Bildschirmzeit haben, vergehen nach konventioneller Decodierung selbst im besten Fall, selbst bei Frame-Wiederherstellung, 0,14 Sekunden, bis der Bildschirm nicht mehr

[18] Hacker und Homebrew-Entwickler haben schon so manche ungeahnten (undokumentierten) Fähigkeiten aus so mancher unterschätzten Hardware herausgeholt

schwarz ist. Wenn nun der Bildschirminhalt nach jedem empfangenen und decodierten Macroblock (ein vollständig empfangener Macroblock kann sofort und unabhängig von folgenden Macroblocks decodiert werden) oder zumindest nach jeder empfangenen und decodierten Slice erneuert wird, kann der Benutzer früher mit dem Betrachten des Bildes beginnen. Aufgrund des unterschiedlichen Zeitverhältnisses von Übertragungs- und Anzeigedauer wird ein sich von oben nach unten aufbauender Bildschirm als Ergebnis wahrgenommen. Diese Technik eignet sich auch für I-Frames, deren Header verpasst wurde (Kapitel 4:Frame-Wiederherstellung).

7.2 Aktiv

Die hier genannten Änderungen müssen am Encoder auf Seiten der Sender durchgeführt werden. Da eine Änderung einer Encodersoftware eine Unterbrechung im Videostrom darstellt sowie die Auswirkungen der Änderungen ein größeres Gebiet (ein ganzes Land oder mehrere Länder) betreffen, ist die Hemmschwelle für solche Änderungen seitens der Betreiber generell sehr hoch. In jedem Fall muss der Videostrom nach der Änderung immer noch (bzw. wieder - siehe ARD) standardkonform sein.
Die hier vorgestellten Methoden sind nicht auf unterschiedliche Codecs übertragbar, daher gibt es für jeden Codec ein Unterkapitel (außer MPEG-4, da wurde im Hauptkapitel bereits eine Reihe von Möglichkeiten auf Seiten der Sender angegeben).

7.2.1 MPEG-2

Um zu gewährleisten, dass die Frame-Wiederherstellung stets optimale Ergebnisse erzielt, dürfen Intra-Quantiser-Matrizen nicht oder nur geringfügig geändert werden. Ich habe nicht geprüft, ob dies überhaupt der Fall ist. Wenn unterschiedliche Sender unabhängig vom Bildmaterial identische Intra-Quantiser-Matrizen verwenden, ist auch anzunehmen, dass sich diese auch auf temporaler Ebene nicht oder nur wenig verändern.
Bei der Intra-Block-Interpolation besteht jedoch Bedarf an Unterstützung seitens des Encoders. Diagramme von Transportströmen zeigen bis zu 5% und selten weniger als 1% an Stuffing-Paketen, also sinnlosen Paketen, die nur gebraucht werden, um eine konstante Datenrate zu gewährleisten, siehe Diagramm 9 (Kapitel 4.2).

Diese könnten für mehr Intra-codierte Macroblocks in Nicht-Intra-Frames verwendet werden. Meine Idee hier wäre, dass der Encoder bei ausreichenden Reserven im Datenstrom gezielt bereits codierte Macroblocks verwirft und diese stattdessen intra-codiert. Die beste Strategie hierzu wäre, Intra-Macroblocks an Objektgrenzen einzusetzen. Dies würde die Intra-Block-Interpolation optimieren, da zwischen zwei Macroblocks auf gleicher Höhe im Wesentlichen nur eine Farbe (bzw. zwei sehr ähnliche Farben) zum Einsatz kommt und nicht angenommen werden muss, dass sich dazwischen noch ein Objekt einer dritten Farbe befindet, es sei denn, es unterschreitet eine gewisse Größe, welche selbstverständlich von der Restkapazität des Transportstroms abhängig ist.
Eine solche Methode benötigt natürlich eine gewisse Menge an zusätzlicher Rechenleistung. Ist diese nicht gegeben, können Macroblocks auch regelmäßig über das Bild verteilt werden. Ein Beispiel wäre, dass alle 4*4 oder 8*8 Macroblocks einer intra-codiert wird. In diesem Falle sollte aber bei jedem Frame mit regelmäßigen Intra-Macroblocks diese versetzt zum vorherigen übertragen werden, um eine erhöhte Bildqualität zu erzielen. In [KER-2006, Kap IIIc] wurden Macroblocks im Schachbrettmuster angeordnet und die Auswirkung von Datenfehlern auf die Bildqualität getestet, im Vergleich zu normaler Anordnung. Das Ergebnis war, dass Fehler im Schachbrett-Bild viel besser verschleiert werden konnten als bei normaler Anordnung. Durch den Versatz der Intra-Macroblocks wird eine ähnliche Qualitätssteigerung ermöglicht.
Um weiter Platz zu sparen, braucht dies nur in P-Frames durchgeführt werden.

7.2.2 H.264

Hier besteht viel Nachbesserungsbedarf. Nicht nur, dass keine der in [KER-2006] erwähnten Fehlerkorrekturmaßnahmen in der Praxis verwendet wird, der Videostrom von ARD ist nicht einmal standardkonform. Die Fehlerkorrekturmaßnahmen sind allerdings auch von Profiles und Levels abhängig, z.B. sind einige dieser Maßnahmen, wie Flexible Macroblock Ordering, im normalen HDTV-Layer nicht erlaubt.
Eine Maßnahme zur Verbesserung der Fehlertoleranz, welche noch nebenbei die Frame-Wiederherstellung bei H.264 ermöglicht, wäre, das Bild in Slices einzuteilen. Ein Overhead von 45*5 Byte = 225 Byte pro Bild würde entstehen, also nicht mehr als 2 Transportpakete. Ein Rechenzeit-Mehraufwand entstünde nicht, jedenfalls kein nennenswerter. Dennoch könnte, aufgrund des Byte-alignment, gefolgt von einem Nullwort, eine Neusynchronisation im Bild stattfinden, was durchaus nötig ist, wenn bereits ein einziges fehlerhaftes Bit eine

gesamte Group Of Pictures zerstört (siehe 6.3.3). Der Strom bleibt standardkonform, aber die Qualität steigt und Frame-Wiederherstellung würde ermöglicht. Durch die lose Koppelung von Startcodes und zugehörigen Daten ist jedoch anzunehmen, dass jede Slice einen eigenen Startcode bekommen kann.
Weitere Maßnahmen zur Erhöhung der Fehlertoleranz und in manchen Fällen auch zur Verkürzung der Umschaltzeiten, wären Data Partitioning und redundante Slices. Ersteres kostet jedoch Rechenzeit und macht den Datenstrom komplexer (und vor allem für Menschen schwerer zu verstehen), letzteres kostet wertvolle Übertragungsbandbreite.
Bei Data Partitioning wird das Bild in 3 unterschiedlich wichtige Kategorien eingeteilt und diese getrennt voneinander gesendet. Gehen die Daten in der wichtigsten Kategorie verloren, geht das Bild verloren. Gehen die Daten in den beiden anderen Kategorien verloren, wird das Bild unterschiedlich stark beschädigt, bleibt aber decodier- und betrachtbar.
Bei der Übertragung redundanter Slices kann der Encoder dieselbe Slice mehrmals (auch in unterschiedlichen Qualitätsstufen) übertragen, sodass der Decoder bei einem Übertragungsfehler auf eine Ersatzrepräsentation der Slice zurückgreifen kann. Dies kostet aber, wie man sich denken kann, nicht wenig Platz im Datenstrom. [KER-2006]

8. Fazit

Die hier vorgestellten Verfahren können die Umschaltgeschwindigkeit auf dem Papier teils deutlich verkürzen. Dies geht allerdings je nach Verfahren mit dem Preis eines mehr oder weniger "beschädigten" Bildes und/oder einer spürbaren Mehrbelastung der CPU einher. Die Ausnahme ist hierbei die Wiederherstellung eines I-Frames.

8.1 I-Frame-Wiederherstellung

Aufgrund der hohen Qualität der Ergebnisbilder, denen ja oft nur ein Streifen am oberen Bildrand fehlt, trifft das Verfahren auch entsprechend auf mehr Akzeptanz beim Endkunden (siehe Umfrageergebnisse Kapitel 8.3). Da sich das Verfahren mit geringem Rechenaufwand implementieren lässt und es zwischen Tuner und externem Decoder gesetzt werden kann, ist anzunehmen, dass es auf vielen bereits vorhandenen Geräten via Firmware-Update aufgespielt werden kann. Da anzunehmen ist, dass konventionelle Decoder den Transportstream auf der CPU entpacken, - die Vorgehensweise ist in etwa die: Lies 188 Bytes aus dem Puffer, prüfe: Sind Bytes 1 und 2 (Byte 0 = 0x47) verUNDet mit 1FFF gleich der PID des aktuellen Programms, wenn ja, sende die restlichen 185 Bytes[19] an den Decoder - muss dieser Vorgang nur geringfügig angepasst werden (siehe Kapitel 4.3):
Kopiere gespeicherte I-Frame-Headerinformationen.
Für (0 bis aktuelle Slice): Erzeuge Slices mit leeren Macroblocks (z.b. durch address_increment, oder Zero_Run, oder Macroblocks mit nur einem DC-Koeffizienten, z.B. Schwarz oder Mittelgrau, oder wiederhole die erste empfangene Slice).
Ab der aktuellen Slice: Schalte in den Normalmodus.

8.2 Intra-Block-Interpolation / Sichtbarmachung der Zwischenbilder

Diese Methode stellt bisher das größte Defizit dar. Eine Implementierung fehlt, die Ergebnisse sind der Vorstellungskraft des Lesers überlassen, eine reine Sichtbarmachung der Zwischenbilder traf auf relativ wenig Akzeptanz. Im Gespräch mit Umfrageteilnehmern wurde vielfach genannt, dass eine komplette Entfärbung bei Bildern ohne Intra-Macroblocks (z.B. der Wasserfall im hr-Fernsehen – Bild 36-47) als angenehmer zu betrachten seien als ein

[19] Abhängig von Vorhandensein von Adaptation Field oder PES-Paket

Bild, das sich teilweise aus Intra-Macroblocks und teilweise aus dunkelgrünen Flächen zusammensetzt.
Die Qualität des Ergebnisses ist ohnehin von den Initialwerten des Decoders abhängig. CodecVisa zum Beispiel zeigte, dass ein helles, schwach gesättigtes Pink erheblich bessere Ergebnisse erzielte als das gewöhnliche Dunkelgrün (siehe Bild 18). Der Gnome Media Player zeigte für H.264, dass mit Mittelgrau als Initialwert überraschende Ergebnisse erzielt werden konnten, man kann auf Bild 78 sogar erkennen, dass Gesicht und Hände des mittleren Polizisten rosa sind.
Bei der Erzeugung des gezeigten Demovideos wurde wie folgt verfahren: Von jedem Bild wurde ausschließlich der Grün-Kanal genommen und eine Kontrastoptimierung auf das zweite Fünftel im Histogramm vorgenommen.

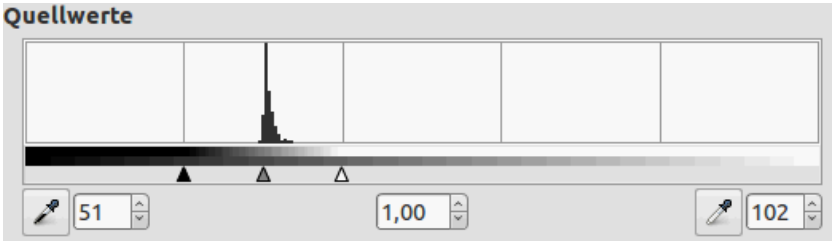


Diagramm 17: Histogramm vom Ursprungsbild zu Bild 42 (dunkelgrün). Bild 42 entstand durch Strecken des zweiten Fünftels auf den Gesamtwerteverlauf im Grünkanal, und die Verwertung dessen als Schwarzweißbild.

Laut dem Histogramm entspricht das nicht dem optimalen Kontrast. Dieser hätte jedoch ein gänzlich unrealistisches Bild erzeugt, welches auch den Übergang zum normalen Bild nach dem ersten I-Frame erschwert hätte. Um die Vorstellung der Ergebnisse der Intra-Block-Interpolation zu erleichtern, kann man ein derart bearbeitetes Schwarzweißbild mit den Farb- und Helligkeitsinformationen aus den Intra-Macroblocks versehen, die vorher homogenisiert, also unscharf vereinheitlicht wurden.
Die Anzahl der Rechenschritte könnte geschätzt wie folgt aussehen:

- Phase 1: Anzahl Intra-Macroblocks*K mit K=eine aus einer nicht festgelegten Formel zu ermittelnde Konstante, die maximal gleich der Bildbreite in Macroblocks ist und mit zunehmender Zahl von Intra-Macroblocks abnimmt.
- Phase 2: Anzahl aus Phase 1 entstandener vertikaler Balken*Anzahl Intra-Macroblocks*K
- Phase 3: Anzahl aus Phase 2 entstandener horizontaler Balken*Bildbreite in Macroblocks*L mit L=Anzahl Slices ohne horizontale Balken.

Der Speicherbedarf kann jedoch in Abhängigkeit von der Verteilung und Entfernung sowie Anzahl der Intra-Macroblocks ausufern. Ob das Verfahren auf der CPU des Receivers eingesetzt werden kann, hängt davon ab, ob der Decoderchip über einen geteilten Arbeitsspeicherbereich verfügt, d.h. ob sich CPU und Decoder einen Bereich im Arbeitsspeicher teilen und ob sich in diesem Speicherbereich die DCT-Koeffizienten befinden. Andernfalls muss das Verfahren auf dem Decoderchip implementiert werden, was eine Nachrüstbarkeit älterer Geräte unmöglich macht.

8.3 Umfragewerte

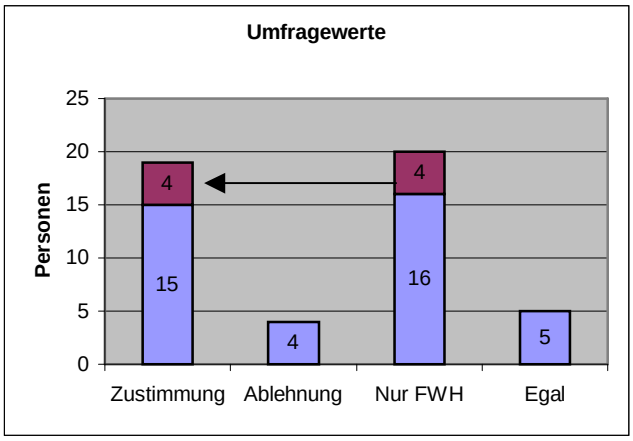


Diagramm 18: Umfrageergebnisse. Die lila Balken enthalten Befürworter von „Nur FWH“ (nur Framewiederherstellung), welche tendenziell auch Intra-Block-Interpolation zustimmen würden, sobald diese einsatzbereit ist. Es wurden 44 Personen befragt.

Bislang überwiegen die Befürworter die Gegner deutlich, wobei rund die Hälfte der Befürworter nur die Frame-Wiederherstellung unterstützt, die Sichtbarmachung der Zwischenbilder aber ablehnt. Ich erwarte, dass die Diskrepanz zwischen Befürwortern und Gegnern der Intra-Block-Interpolation sinkt, wenn diese implementiert und deren tatsächliche Resultate gezeigt werden. Sehr wahrscheinlich wird es aber weiterhin Personen geben, welche die Intra-Block-Interpolation ablehnen, die Frame-Wiederherstellung aber befürworten.
In jedem Fall sind die genannten Verfahren - mit Ausnahme derer, welche die Bildqualität nicht beeinträchtigen - in zukünftigen Fernsehgeräten als Option zuzuschalten, d.h. der Benutzer kann wählen, ob er die volle Beschleunigung, eine Teilbeschleunigung nur mit Frame-Wiederherstellung oder die konventionelle Methode haben möchte.

8.4 Passive Hilfsmittel

Diese Verfahren ermöglichen mit einfachen und recht naheliegenden Mitteln den besten Kompromiss zwischen Umschaltgeschwindigkeit und Qualität. Die Nachteile sind erhöhter Bedarf an Speicher und Rechenzeit. Nachrüstbarkeit von Geräten mit Twin-Tuner sollte gegeben sein, das einzige verbleibende Problem ist das der Unvorhersagbarkeit des Umschaltverhaltens des Benutzers: Es können immer nur so viele verschiedene Streams empfangen werden, wie Tuner vorhanden sind. Wechselt der Benutzer auf eine Frequenz, die gerade nicht im Hintergrund decodiert wird, sind die Wartezeiten wieder da. In den meisten Fällen wird die Umschaltzeit jedoch drastisch reduziert werden; selbst bei Geräten mit nur einem Tuner sinkt die mittlere Umschaltzeit um etwa 3/4 (ausgehend von 4 Sendern pro Frequenz, also alle 4 Kanalwechsel = 1 Frequenzwechsel). Hersteller von Geräten, die diese Verfahren einsetzen, müssen nur das hardwareseitige Umschalten optimieren, d.h. z.B. zu vermeiden, unnötige Hardware-Neuinitialisierungen durchzuführen. Eine Lösungsmöglichkeit wäre, zwei Decoder unabhängig voneinander laufen zu lassen und den Videostrom über einen Demultiplexer umzuschalten, was die Umschaltzeit auf maximal einen Frame reduziert, aber die Hardwarekosten erhöht.
Die inneren Vorgänge von HDMI sind mir nicht bekannt. Wird das Bild unkomprimiert übertragen, ist anzunehmen, dass die Umschaltzeit von einem auf den nächsten Pixel erfolgen kann. Wird das Bild komprimiert übertragen, muss wahrscheinlich auf den nächsten Frame gewartet werden. Im Höchstfall ist das also 1/25 Sekunde, also deutlich schneller als die meisten Analogfernseher.

8.5 Aktive Hilfsmittel

Die hier dargestellten Möglichkeiten werden eine moderate Qualitätssteigerung bei vermutlich eher geringer Akzeptanz seitens der Sender ermöglichen. Eine wahrnehmbare Qualitätssteigerung wird nur bei Geräten, auf denen Frame-Wiederherstellung und Intra-Macroblock-Interpolation implementiert sind, stattfinden, wobei durch die Einteilung des Bildes in Slices bei H.264 selbiges etwas robuster gegen Empfangsstörungen wird, auch auf Geräten, die nicht mit den in dieser Arbeit beschriebenen Methoden ausgestattet sind. Die Kluft zwischen einem guten digitalen Bild und einem durch Empfangsstörungen komplett zerstörten Bild ist bei QAM256 jedoch sehr gering [WQA-2013].

Für die Betreiber von Sendern wird, wie bereits genannt, die Hemmschwelle für die Veränderung des Encodings sehr hoch sein, da eine Veränderung oftmals eine Unterbrechung der Sendung sowie eine Ungewissheit in den Auswirkungen der Veränderung bedeutet, insbesondere, wenn diese Veränderungen nur in (anfangs) sehr wenigen Geräten zu wahrnehmbaren Verbesserungen führt. Kurz gefasst: Großer Aufwand, geringe Auswirkungen.[20]

8.6 Zusammenfassung

Die in 1.1.1, 3.1, 6.1.1 und 6.3.3.2 ermittelten Werte für den Ist-Zustand der Umschaltzeiten beim digitalen Fernsehen bleiben weit hinter den für das analoge Fernsehen ermittelten Werten zurück. In dieser Arbeit wurden Methoden vorgestellt, sowohl experimentelle mit fragwürdigem Ergebnis als auch praktisch anwendbare mit großem Nutzen bei geringem Aufwand und recht akzeptablen Ergebnissen. Alles in allem können die Umschaltzeiten beim digitalen Fernsehen auf Werte unterhalb denen des analogen Fernsehens reduziert werden - es wird zwar weiterhin Ausnahmen geben, welche auch (vergleichsweise selten) auftreten werden, aber insgesamt ist das Ergebnis recht beachtlich.
Bei MPEG-2 (SDTV, 576i, 25 Bilder/Sekunde) konnte die Umschaltzeit im Falle eines I-Frames auf etwa 0,14 Sekunden, bei einem B-Frame etwa 0,022 Sekunden, reduziert werden. Die ästhetische Akzeptanz des Bildes im Falle eines B-Frames ist zwar geringer, aber mancher zieht es einem schwarzen Bildschirm vor (siehe Kapitel 8.3). Bei H.264 beträgt die minimale Umschaltzeit bei einem B-Frame 0,024 Sekunden.[21]
Ein erstaunliches Ergebnis ist, dass die einfachsten und naheliegendsten Methoden (Kapitel 4.3, alle Unterkapitel von 7.1) die besten Resultate versprechen, in der Praxis aber von den meisten Herstellern nicht angewandt zu werden scheinen (ein Unitymedia-Mitarbeiter schilderte mir, dass Unitymedia das in 7.1.2 genannte Multi-Tuner-Verfahren einsetzt).

[20] Pareto (80-20)-Prinzip: 80% des Erfolgs mit 20% des Aufwands und die restlichen 20% des Erfolgs benötigen 80% des Aufwands.
[21] Ermittelt in HDSMPCOR.TS. Parameter: ZDF HD, Kabelnetz Unitymedia, [AFT-2013i]

8.7 Ausblick

Das Medium Digitalfernsehen ist einem stärkeren Wandel unterzogen als das analoge Fernsehen. Dies gilt generell für alle digitalen Geräte, man nehme zum Beispiel Mobiltelefone, welche im Jahresrhythmus veralten[22]. Das analoge Fernsehen in der für Europa bekannten Norm CCIR-625 ist seit 1950 in Betrieb (die 625-Zeilen-Norm für das europäische Fernsehen wurde am 24.7.1950 festgelegt [WWG-2013]), in Farbe (PAL) seit 1967 [WGF-2013], während das erste massentaugliche Digitalfernsehen in MPEG-2 sich nach nicht einmal 2 Jahrzehnten bereits auf dem Rückzug befindet, da H.264 ein deutlich effizienterer Codec ist, und sich bereits mit H.265 (auch bekannt als MPEG-7/HEVC) [WH5-2013] ein noch effizienterer Codec ankündigt. Ebenso wie Codecs ändern sich Auflösungsformate: Während PAL und das erste DVB nur 720*576 Pixel Auflösung haben, wird HDTV in 1280*720 oder 1920*1080 Pixeln Auflösung übertragen, und 4kTV (4096*2304) ist der nächste Schritt.
Diese Entwicklungsgeschwindigkeit und die Aufwärtsinkompatibilität (ältere Geräte können Sendungen für neuere Geräte nicht abspielen) wird für eine relativ schnelle Obsoleszenz einiger der hier beschriebenen Methoden sorgen, ebenso ist sehr wahrscheinlich, dass nach Ablauf derselben Zeitspanne, die das analoge Fernsehen überdauert hat, Codecs wie MPEG-2 oder H.264 gänzlich in Vergessenheit geraten könnten. Nichtsdestotrotz können die hier vorgestellten Methoden das digitale Fernsehen für die nächsten Jahre verbessern.

[22] Nach der Vorstellung unserer Wegwerfgesellschaft vom Begriff "Veralten".

9. Anhang

Dies ist kein integraler Bestandteil dieser Arbeit. Es handelt sich hauptsächlich um Bedienungsanleitungen von Programmen, welche ich im Verlauf dieser Arbeit geschrieben habe, heruntergeladene Programme und den Inhalt der beigelegten CD. Die Bedienungsanleitungen braucht man, um viele der in dieser Arbeit ermittelten Ergebnisse nachzuvollziehen.

9.1: ORDERTS

9.1.1 Orderts - Übersetzung

Der beigefügte Quellcode ist mittels Borland Turbo C++ 2.0 oder höher in der MS-DOS Version zu übersetzen. Der Quellcode verwendet proprietäre Konsolenausgabe, die Befehle textcolor, textbackground, cprintf, gotoxy, wherey (und unter Linux auch getch) kommen in der conio-Includedatei heute gebräuchlicher C/C++ Compiler nicht vor, sind aber von essentieller Bedeutung für das Funktionieren des Programmes bzw. für das Verständnis von dessen Output.
Die beigelegte ORDERTS.EXE wurde mit Borland Turbo C++ 3.1 übersetzt und läuft als Standalone, d.h. jeder Computer mit MS-DOS kann dieses Programm starten, ohne dass weitere Dateien nötig wären. Eine Übersetzung unter Linux wird nicht ohne weiteres möglich sein.

9.1.2 Orderts - Systemanforderungen

ORDERTS.EXE wurde im Modus für 8086/8088 PCs übersetzt, würde also auch auf den Museumsstücken in der Vitrine laufen. Es arbeitet im Wesentlichen lokal auf der Datei und speichert nur einige hundert Byte an Zusatzinformationen, ist also nicht allzu speicherhungrig und würde - wie gesagt - auch auf den IBM-Computern (5100 und höher) in der Vitrine laufen (das Speichern eines über 20 MB großen TS-Mitschnitts wird jedoch weniger möglich sein). Ich glaube jedoch, dass MS-DOS Version 3.3 oder höher benötigt wird, außerdem ist

eine Farbgrafikkarte (CGA oder höher) dringend anzuraten, um aus der Ausgabe schlau zu werden.

9.1.3 Orderts - Kurzbeschreibung

Dieses Programm stellt das Innenleben eines Transport Streams gemäß ISO/IEC 13818-1 [I31-2000] dar. Es wird ein Sendersuchlauf gemacht (in einem Transport Stream gibt es üblicherweise mehr als einen Sender), die PIDs der Datenpakete sowie deren Bedeutung angezeigt, im Hauptteil des Programms wird jedes Paket im Strom, das zu einem Sender gehört, farbig dargestellt, sowie, ob es ein Audio- oder ein Video-Paket ist, und wenn es ein Videopaket ist, welcher Frametyp es ist (I/P/B, sofern feststellbar). Jeder Frame-Header wird hell und mit farbigem Hintergrund (Ausnahme: Dunkelgrau) hervorgehoben, sowie die Anzahl der Pakete, die der vorausgehende Frame dauerte, diese Zahl ist absolut, Pakete von anderen Programmen im Strom werden nicht mitgezählt. Um die relativen Werte zu erhalten, kann man [PDM-2013] verwenden. Alternativ kann Orderts auch die Längen der Slices eines Bildes anzeigen (nur für MPEG-2).

9.1.4 Orderts - Bedienungsanleitung

Aufgerufen wird das Programm über die Kommandozeile. Alternativ kann man eine .TS-Datei auf das Programm ziehen oder mit diesem verknüpfen (Windows). Empfohlen wird jedoch, eine MS-DOS-Konsole zu öffnen und "orderts [DATEI]" einzugeben. Beispiel: "orderts sample.ts".
Anschließend wird man gefragt, ob man die Programmzugehörigkeit der Pakete (1) oder die Slicelängen (2) sehen will. Als nächstes wird man gefragt, ob die PAT gesucht werden soll. Wenn eine andere Datei als sample.ts untersucht werden soll, muss Y gedrückt werden, andernfalls wird das Programm die Datei erfolglos durchsuchen und nichts anzeigen.
Das Programm zeigt anschließend, in welchem Block die PAT gefunden wurde, sowie deren Inhalt, also welche Programme gefunden wurden. Anschließend wartet das Programm auf einen Tastendruck, da die folgende Ausgabe die vorherige aus dem Bildschirm scrollen kann, bevor man sie lesen kann (auf neueren Maschinen kann man hochscrollen).
Hiernach wird die Datei nach den Programmen (PMT) aus der PAT durchsucht, nach jedem gefundenen Programm hält die Bildschirmausgabe erneut und wartet auf einen Tastendruck.

Bestandteile der gefundenen Programme werden auf dem Bildschirm ausgegeben, also Videospuren (MPEG2/H.264), Audiospuren, ES (Zusatzinformationen).
Als letztes wird die Belegung des Video-PID Arrays ausgegeben, was sowohl für Debug-Zwecke interessant ist, als auch nützlich ist, um die Farben den PIDs zuordnen zu können. Jede Farbe steht für ein Programm.
Nach Ende des Hauptteils des Programmes, dessen 2 Modi in A.4.1 und A.4.2 beschrieben sind, fragt das Programm, ob Statistiken erwünscht sind. Der Hauptteil wird beendet, wenn entweder Escape gedrückt wird oder die Datei zuende ist. Das Programm schreibt die Anzahl der geparsten Pakete auf den Bildschirm, was wichtig ist, um z.B. zu berechnen, wieviele Pakete eine Sekunde lang ist. Anschließend wird der Anteil, den jedes Programm am Strom hat, errechnet und angezeigt, wieviele Frames bis zum Programmende vergangen sind (für jedes Fernsehprogramm einzeln). Abschließend wird geprüft, wie regelmäßig die PMTs sowie die PAT vorkommen. Dies kann z.B. verwendet werden, um zu berechnen, wie lange der Sendersuchlauf dauert.

9.1.4.1 Orderts - Modus 1: Blockzuordnung

Bild 81: Beispielausgabe von Orderts, Hauptprogramm, Mode 1.

Auf dem Bildschirm erscheinen viele Buchstaben und einige Zahlen. Legende:
V: Video, unbekannter Frametyp
4: Video, H.264 codiert.
I: Video, I-Frame.
P: Video, P-Frame.
B: Video, B-Frame.
A: Audio.

Gelegentlich erscheinen Zahlen in eckigen Klammern, Hintergrundfarbe und heller Textfarbe, gefolgt von einem Buchstaben, der den Frametyp angibt. Die Zahl gibt, wie beschrieben, die Anzahl der Blocks, die der letzte Frame gedauert hat, an, der Buchstabe gibt den Frametyp des nächsten Frames an. Sieht man in einer Farbe (beispielsweise Grün) den Wert [10]B, und sieht man als nächsten grün gehighlighteten Wert den Wert [58]I, dann weiß man, dass der B-Frame 58 Blocks groß war. Um herauszufinden, wie lange der I-Frame dauert, muss man zum nächsten grünen Highlight-Block gehen, und den Wert dort ablesen (hier: [306]B). Die Zahl im ersten Vorkommen eines Highlighters (hier: [10]B) ist bedeutungslos, der Frametyp jedoch nicht.
Bei H.264-codierten Videoströmen wird der Frametyp zwar in jedem Highlighter angezeigt, Einzelpakete werden jedoch als '4' ausgewiesen, um kenntlich zu machen, dass dieser Videostrom H.264-codiert ist.

9.1.4.2 Orderts - Modus 2: Slice Längen

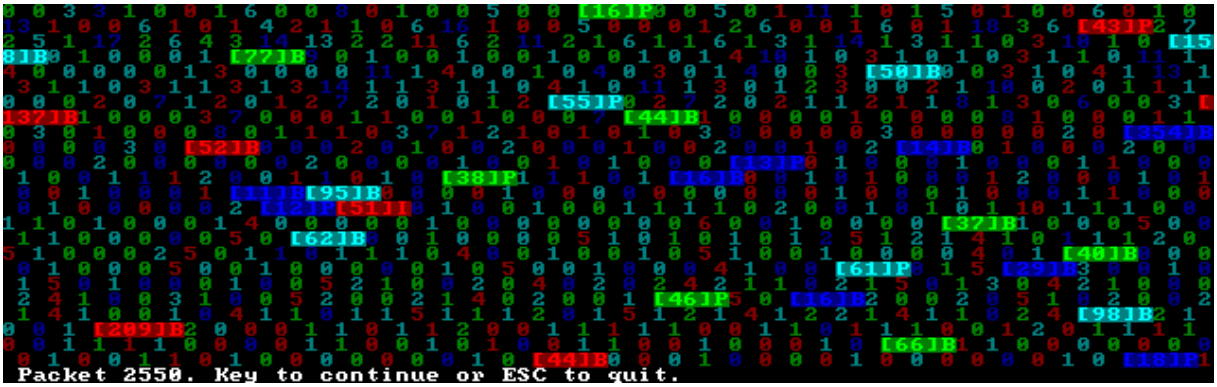


Bild 82: Beispielausgabe von Orderts, Hauptprogramm, Mode 2.

Es wird davon abgeraten, diesen Modus mit Transport Streams, die H.264-codierte Videoströme beinhalten, zu verwenden, da die Ausgabe dann etwas unschön ist. Wenn man jedoch alle Ausgaben in der Farbe des Programms mit dem H.264-Videostrom ignoriert, geht es.
Die Highlighter mit dem Frametyp und den Längen der Frames in Blocks, wie aus A.4.1 bekannt, wird weiterhin angezeigt. Zwischen Frame-Headern werden jedoch Zahlen in der Farbe des zugehörigen Programms angezeigt. Jede Zahl steht für die Anzahl an Transport-Stream-Paketen, welche benötigt wurden, um diese Slice zu codieren, minus 1, also das erste Paket wird nicht mitgezählt. Slicelängen erlauben Mutmaßungen über den Frametyp, wenn dieser nicht bekannt ist.

9.2 TS2M2V

9.2.1 Ts2m2v - Übersetzung

Im Gegensatz zu Orderts ist bei ts2m2v farbige Ausgabe nicht wichtig. Der Quellcode ist für Linux vorbereitet, da Linux nicht über die getch-Funktion verfügt. Um das Programm unter Windows zu kompilieren, muss die Zeile #include <termios.h> sowie die Funktion int getch() gelöscht werden. Anschließend muss eine Zeile #include <conio.h> eingefügt werden. Abgesehen davon ist das Programm ANSI-C kompatibel, benötigt also keinen exotischen Compiler wie Borland Turbo C++. Die Systemanforderungen sind abhängig vom kompilierenden System. Auf dem System, auf dem das Programm übersetzt wird, wird es also auch laufen.
Auf der CD ist eine mit Borland Turbo C++ 3.1 übersetzte Version des Programms beigelegt.

9.2.2 Ts2m2v - Kurzbeschreibung

Dieses Programm entpackt Transportströme. Es ist gedacht, um Raw-Videostreams aus Transport Streams zu entpacken, kann jedoch verwendet werden, um beliebige Datenstreams aus Transport Streams zu entpacken. Es kann mit MPEG-2 und H.264-codierten Videoströmen umgehen und beinhaltet Sichtbarmachung der Zwischenbilder für MPEG-2 und H.264, sowie Frame-Wiederherstellung von MPEG-2.
Die von ts2m2v erzeugten Videoströme können dann mit einigen Videowiedergabeprogrammen, z.B. dem VLC Media Player, abgespielt werden.

9.2.3 Ts2m2v - Bedienungsanleitung

Das Programm erwartet 2 Parameter: Die Eingabedatei (ein Transportstrom) und die Ausgabedatei (Name beliebig). Damit kann es nur noch von einer Konsole aus gestartet werden.
Nach erfolgreichem Aufruf zeigt das Programm zunächst, welche Videoströme es in der Transportstreamdatei gefunden hat. Wie schon bei orderts, steht -1 für ein nicht vorhandenes Programm, also ein NULL-Wert.
Nun drückt man die Taste des Programms, das man extrahiert haben möchte.

Der weitere Prozess ist abhängig vom Typ des Videostroms (MPEG-2 oder H.264).

Der MPEG-2 Referenzdecoder wird mit
./mpeg2decode –b [Name der Videodatei] –o3 [Name der Ausgabedatei, beliebig]
aufgerufen. Gibt man das Flag –t hinzu, findet eine Trace-Ausgabe statt, die mehrere Megabyte pro Bild groß ist und den Decodiervorgang bis ins letzte Detail aufschlüsselt.

9.2.4 Ts2m2v - Implementierte Modi

Es sind 6 Modi implementiert:
1.) Clean (Sauber): Das Ergebnis ist eine mustergültige m2v-Datei. Das Verfahren entspricht dem in Kapitel 4.2 beschriebenen.
2.) Dirty (Schmutzig): Die Zwischenbilder werden sichtbar gemacht, die Datei beginnt mit dem ersten gefundenen Frame-Header, alles davor wird abgeschnitten.
3.) Ugly (Hässlich): Eine Frame-Wiederherstellung wird durchgeführt. Hierbei wird der Typ des ersten, unvollständige Frame ermittelt, der Benutzer hat diesen zu verifizieren (Is that correct (Y/N)), wenn dies nicht korrekt ist, muss der korrekte Frametyp (1=I, 2=P, 3=B) angegeben werden. Das Programm füllt die fehlenden Informationen anhand eines zukünftigen Frames dieses Typs auf und erzeugt eine mustergültige m2v-Datei.
4.) Just unpack (Nur entpacken). Ein Rawdump des Streams wird erstellt, d.h. Pakete werden ausgepackt und deren Inhalt konkateniert, unabhängig ihres Typs. Damit kann man z.B. Audioinformationen oder EPG-Informationen entpacken. Ebenso kann man damit Videoströme, die mit zukünftigen Videocodecs komprimiert sein werden, entpacken, man wird jedoch keine mustergültige Datei erhalten.
5.) Clean (Sauber): Nur für H.264-codierte Videoströme. Das Ergebnis ist eine mustergültige .264-Datei, das Verfahren verwendet die in Kapitel 6.3.2 festgestellten Reihenfolge.
6.) Dirty (Schmutzig): Nur für H.264-codierte Videoströme. Das Ergebnis entspricht dem von Modus 2, jedoch für H.264-codierte Videoströme. Das Verfahren nimmt einen Frame-Header aus einem I-Frame, verändert jedoch den NAL-Unit-Delimiter. Dies wird benötigt, da Frame-Header für Nicht-I-Frames (Non-IDR-NAL) minimalistisch sind und die notwendigen Informationen, damit Decoder funktionieren können, nicht enthalten. Mit dieser Methode entpackte Videoströme können nur mit wenigen Videowiedergabeprogrammen (z.B. CodecVisa und Gnome Media Player) abgespielt werden. Das Referenzframework und der VLC Media Player streikten.

9.3: H264DEC - Übersetzung und Bedienungsanleitung

Das Programm benötigt zur Übersetzung - ebenso wie Orderts - Borland Turbo C++ Version 2 oder höher. Eine übersetzte, ausführbare Datei ist auf der CD.
Das Programm wird, wie Orderts, mit dem Dateinamen der zu decodierenden Datei aufgerufen.
H264dec sucht im Wesentlichen nach H.264-spezifischen Startcodes und zeigt, wenn es welche findet, einige für diesen Startcode spezifische Informationen an, sowie den Dateizeiger (dezimal), an dem der Startcode zu finden ist.
Das Programm hält, wie der Hauptteil von Orderts, nach jeder vollen Bildschirmseite.
Das Programm ist sehr simpel gehalten und führt keine Fehlererkennung/Sanity Check aus. (Das Programm kann man mit jeder beliebigen Datei (spaßeshalber auch dem Programm selbst) ausführen)

9.4: EXPGOL - Übersetzung und Bedienungsanleitung

Das Programm benötigt zur Übersetzung - ebenso wie Orderts - Borland Turbo C++ Version 2 oder höher. Eine übersetzte, ausführbare Datei ist auf der CD.
Das Programm wird wie Orderts mit dem Dateinamen der zu decodierenden Datei aufgerufen.
Beim Start von Expgol kann man einen Offset einstellen. Dieser wird hexadezimal eingegeben, indem man mit den Tasten U/J die 16er-Stellen in/dekrementiert, mit I/K die 1er-Stellen in/dekrementiert und mit O/L die Bitposition im Byte einstellt. Zum Abschluss drückt man Enter.
Das Programm zeigt die decodierten Exp-Golomb Codes in einer Tabelle an, wobei links der bytegenaue Lesezeiger steht. Ungewöhnliche Werte werden mit einem gelben "WARNING", illegale Werte mit einem pinken "ERROR" versehen.
Der Zweck des Programms ist es, das Einschwingverhalten von Exp-Golomb-Codes zu studieren.

9.5 TSISLICE - Übersetzung und Bedienungsanleitung

Dieses Programm ist in der Bedienung und Übersetzung ähnlich zu H264DEC und EXPGOL. Die Ausgabe beschränkt sich auf Frametyp und Wert des intra_slice_flags. Format der Eingabedatei ist ein MPEG-2 Video, es könnte jedoch auch auf Transport Stream Dateien funktionieren, vermutlich wird die Ausgabe ein wenig durcheinander sein. Zweck war, zu prüfen, ob dieser Flag in I-Frames den Wert 1 annimmt, was nicht der Fall war.

9.6 Externe Programme

Diese Programme wurden aus dem Internet geladen und halfen bei der Analyse von Transport Streams sowie der Videoströme selbst. Anleitungen zu den Programmen sollten sich auf den zugehörigen Webseiten befinden, die Programme selbst sind recht intuitiv. Dennoch werden hier Kurzanleitungen gegeben.

9.6.1 [PDM-2013] MPEG-2 TS packet analyser 2.4.2.0 von Peter Daniel

In der Symbolleiste sind die 5 Symbole von links wichtig: Datei öffnen, an den Anfang der Datei springen, vorheriges Paket, nächstes Paket, gehe zu Paket. Die Pfeile (vorheriges Paket, nächstes Paket) sind abhängig von den Filtern (links oben).
Filter: Payload start indicator: Setzt man dieses Flag, zeigt der TS packet analyser beim Benutzen der Pfeile in der Symbolleiste nur noch Pakete an, deren Payload start indicator gesetzt ist. In diesen Paketen ist häufig ein Frame-Header zu finden. PID: Hier kann man eine beliebige PID eingeben, z.B. die eines Videostroms. Setzt man das Flag und trägt eine PID ein, zeigt der TS packet analyser beim Benutzen der Pfeile nur noch Pakete mit dieser PID an. Wenn man beispielsweise sample.ts öffnet, bei PID 600 einträgt und Payload start indicator aktiviert, werden nur noch Frame Header angezeigt, in Verbindung mit der Paketnummer (Rahmen des mittleren Fensters (Hex-Viewer)) sehr nützlich, um die Abstände zwischen Frames zu ermitteln.
Im linken mittleren Fenster werden die ersten 4 Byte des Pakets aufgeschlüsselt, das mittlere ist ein Hex-Viewer, der einen direkten Blick auf den Datenstrom erlaubt, das Fenster unten Mitte zeigt gelegentlich Daten über den Typ des Pakets, das Fenster ganz rechts zeigt gelegentlich Daten zum Videostrom (z.B. Frame Header) an.

9.6.2 [ODA-2009] OpenEye DVBAnalyzer

Eine Datei wird mit dem Menü File->Select Source geöffnet. Im linken Fenster werden PAT und PMTs sowie deren Inhalt angezeigt, also auch Typ der Audio- und Videoströme. Selbst H.264 wird erkannt. Das Programm analysiert jedoch die gesamte TS-Datei, was eine Weile dauert, und leider ist es nicht sonderlich robust. Bei ardhd.ts stürzt es ab, bevor es mit Decodieren fertig ist. Auf der rechten Seite werden (meines Erachtens) ziemlich unnütze Informationen über die einzelnen Pakete angezeigt.

9.6.3 [SDI-2013] DVB Inspector 0.0.9

Dateien werden über File->Open geöffnet. Im Tree-Tab werden Informationen vergleichbar mit denen im linken Fenster vom [ODA-2009] angezeigt, jedoch mit mehr Text, dafür ohne Symbole (nur ein Ordner-Symbol).
Die anderen Tabs stellen einige interessante und nützliche Diagramme über den Aufbau des Datenstroms bereit, wie Bitrate, prozentualer Anteil der PIDs, und EPG-Informationen. Grid View stellt im Wesentlichen eine Variante meines Orderts-Programmes dar, jedoch ohne spezifische Informationen. Diese erscheinen zwar, wenn man die Maus darüber hält, aber Frametyp, Länge des Frames in Paketen und Länge der Slices fehlen.

9.7 Inhalt der CD

Die CD enthält außer einer Kopie dieser Arbeit in Word2000- und PDF-Format, Quellen, erstellte Artefakte (darunter interne Software, das Videodokument zu den analogen Umschaltzeiten und die im Literaturverzeichnis referenzierten internen Quellen), einige der benutzten Programme, weitere Bilder (z.B. decodierte Frames aus Videodateien), das Demovideo (Frame-Wiederherstellung und Sichtbarmachung der Zwischenbilder). Des Weiteren sind nicht verwendete Quellen, die jedoch als weiterführende Literatur genutzt werden können, vorhanden. Einige dieser unbenutzten Quellen haben den Entscheidungsfindungsprozess während der experimentellen Phase der Arbeit beeinflusst.

10. Literaturverzeichnis

10.1 Interne Dokumente

Diese Dokumente wurden von mir, Simon Augustin, als Nebenprodukte der Masterarbeit erstellt. Viele enthalten nützliche, weiterführende Informationen, andere enthalten Mengen an Detailinformationen (auf denen z.B. die erstellten verwendeten Diagramme basieren), drei davon ([ASE-2013i], [VDO-2013i] und [MDO-2013i]) wurden maschinell erstellt und enthalten Debugausgaben bzw. Detailinformationen vom Decodierungsvorgang.
Zur besseren Unterscheidbarkeit enthalten die Referenzen das Suffix 'i' für 'intern'.

[AAT-2013i] analog_timings.txt
[ADT-2013i] digital_timings.txt
[AF2-2013i] Framewiederherstellung_h264.txt
[AFT-2013i] frametimings.txt
[APT-2013i] physical_timings.txt
[ASE-2013i] sender.txt
[ATE-2013i] tracing_expgolomb.txt
[ATF-2013i] tracing_flags.txt
[AUA-2013i] umschaltzeiten_analog.avi
[HER-2013i] hr-error.txt
[MDO-2013i] traceout.txt
[VDO-2013i] vlcdebug_test4s2.txt

10.2 Interne Software (Autor / Programmierer: Simon Augustin)

[AEG-2013i] expgol.exe. Exp-Golomb-Decoder.
[AH2-2013i] h264dec.exe. Zeigt Aufbau und Struktur von H.264-Dateien an.
[AOT-2013i] Orderts.exe. Zeigt viele Details über das Innenleben von Transport Streams an.
[ATS-2013i] ts2m2v.exe. Entpackt Transport Stream Dateien in einzelne Streams.

10.3 Externe Dokumente

[ADB-2013] Andrew Duncan: MPEG-1 Data Structures: The Big Picture. www.andrewduncan.ws/MPEG/MPEG-2_Picts.html, zuletzt besucht am 7. März 2013

[CSM-1993] Campenhausen, Christoph von: Die Sinne des Menschen: Einführung in die Psychophysik der Wahrnehmung. Stuttgart: Georg Thieme Verlag, 2. Auflage, 1993

[EHG-2012] Hergenröther, Elke: Grafische Datenverarbeitung: Bildkompression & Dateiformate. Hochschule Darmstadt, 2012

[ETR-2009] Unbekannt: Eyetracking. FH-Köln, 2009. Autor ließ sich nicht ermitteln. ebookbrowse.com/1-eyetracking-introduction-sgmci-ss09-pdf-d372265142

[FFG-2010] Frank Fuhrmann: Seminartour 2010 Aufbereitungen. Grundig SAT Systems, 2010.

[HPA-2004] Said, Amir: Introduction to Arithmetic Coding - Theory and Practice. Hewlett-Packard, April 2004.

[HPM-2006] Hewlett-Packard: MPEG-2: The basics of how it works. Hewlett-Packard, 2006. Autor unbekannt. Datei enthält Vortragsfolien.

[I31-2000]* ISO/IEC: 13818-1: Information Technology - Generic coding of moving pictures and associated audio information: Systems. ISO/IEC, Second Edition, Dezember 2000

[I32-1995]* ISO/IEC: 13818-2: Information Technology - Generic coding of moving pictures and associated audio information: Video. ISO/IEC, 1995

[I42-2001]* ISO/IEC: 14496-2: Information technology - Coding of audio-visual objects: Part 2: Visual. ISO/IEC, Second Edition, Dezember 2001

[I4A-2004]*[*] ISO/IEC: 14496-10: Information technology - Coding of audio-visual objects: Part 10: Advanced Video Coding. ISO/IEC, Second Edition, Oktober 2004

[KER-2006] Kumar, Sunil et. al: Error Resiliency Schemes in H.264/AVC Standard. Elsevier J. of Visual Communication & Image Representation (Special issue on Emerging H.264/AVC Video Coding Standard), Vol. 17(2), April 2006.

[KSP-2003] Karczewicz, Marta; Kurceren, Ragip: The SP- and SI-Frames Design for H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology, Vol.13, No.7, Juli 2003.

[*]) Kostenpflichtiges Dokument, nicht auf der CD enthalten.

[MVC-1982] D.Marr: Vision: a computational investigation into the human representation and processing of visual information, zitiert über [CSM-1993], Kap 11.2.1

[PCU-2004] Philips: Datasheet: All Bands DVB-C Receiver Unit CU1216 family. Philips RF Solutions, 2004

[RBV-2013] Dr. Dr. Beck, Reiner: Visuelle Wahrnehmung. Datum unbekannt.

[RTQ-2009] Iain Richardson, 4x4 Transform and Quantization in H.264/AVC, VCodex Ltd White Paper, April 2009, http://www.vcodex.com/

[RWV-2009] Rüdiger Wölfelscheider: Vorlesungsskript zur digitalen Videotechnik: Teil 2. Hochschule Darmstadt, 2008

[TFD-2007] ARD, ZDF: DVB-T Leitfaden. Task Force DVB-T Deutschland von ARD und ZDF. Oktober 2007.

[WSW-2013] Schneider, Werner et. al: Wettbewerb und Prioritätskontrolle in Geist und Gehirn: Neue Perspektiven aus der Forschung zu Aufmerksamkeit und Sehen. In: ZiF-Mitteilungen Februar 2013.

10.4 Verwendete externe Software

[CV-2013] CodecVisa. www.codecian.org, erstellt 2013, Seite ist nicht datiert. Zuletzt besucht am 11.7.2013

[M2S-1996] MPEG Software Simulation Group: MPEG-2 Encoder / Decoder. MSSG: www.mpeg.org/MSSG/, Version 1.2, Juli 1996. (Quellcode MPEG-2 Framework)

[M4S-2013] Sühring, Karsten: H.264/AVC Reference Software. iphome.hhi.de/suehring/tml/download/, Mai 2013. (Quellcode H.264 Framework)

[ODA-2009] OpenEye DVBAnalyzer http://dvbanalyzer.sourceforge.net. Erstellt 2009, Seite ist nicht datiert. Zuletzt besucht am 11.7.2013

[PDM-2013] Peter Daniel: MPEG-2 TS packet analyzer, www.pjdaniel.org.uk/mpeg, Seite ist nicht datiert. Zuletzt besucht am 11.7.2013

[SDI-2013] DVB Inspector 0.0.9 sourceforge.net/projects/dvbinspector, erstellt (geändert) am 21.6.2013, zuletzt besucht am 11.7.2013

10.5 Internetquellen

[HIF-2010] HIFI-FORUM: Lange Umschaltzeiten - LC-46LE705S. www.hifi-forum.de/viewthread-140-2209.html, erstellt am 17.5.2010, zuletzt besucht am 11.7.2013

[KUG-2013] Gegenfurtner, Karl et. al: Visuelle Informationsverarbeitung im Gehirn. www.allpsych.uni-giessen.de/karl/teach/aka.htm, zuletzt besucht am 11.7.2013

[MKB-2013] Unbekannt: Kanalbelegung digitales Kabelfernsehen (DVB-C) ausgebaute Unitymedia-Kabelnetze (ehem ish und iesy), primär Bonn. http://www.mischobo.de/Kabel/kanalbelegung.html, erstellt am 5.7.2013, zuletzt besucht am 11.7.2013

[NWA-2008] N-TV: Aufmerksamkeit entscheidet: Verarbeitung visueller Reize. www.n-tv.de/wissen/Aufmerksamkeit-entscheidet-article31817.html, erstellt am 30.10.2008, zuletzt besucht am 11.7.2013

[WAC-2013] Wikipedia: Arithmetic coding. en.wikipedia.org/wiki/Arithmetic_coding, erstellt am 2.7.2013, zuletzt besucht am 11.7.2013

[WAT-2013] Wikipedia: Advanced Television Systems Committee standards. en.wikipedia.org/wiki/Advanced_Television_Systems_Committee_standards, erstellt am 30.6.2013, zuletzt besucht am 11.7.2013

[WFM-2013] Wikipedia: Frequenzmodulation. de.wikipedia.org/wiki/Frequenzmodulation, erstellt am 1.7.2013, zuletzt besucht am 11.7.2013

[WGE-2013] Wikipedia: Gelber Fleck (Auge). http://de.wikipedia.org/wiki/Gelber_Fleck(Auge), erstellt am 30.3.2013, zuletzt besucht am 11.7.2013

[WGF-2013] Wikipedia: Geschichte des Fernsehens. de.wikipedia.org/wiki/Geschichte_des_Fernsehens#Entwicklung_zum_Massenmedium_ab_1950, erstellt am 28.6.2013, zuletzt besucht am 16.7.2013

[WGH-2013] Wikipedia: Geschichte des hochauflösenden Fernsehens. de.wikipedia.org/wiki/Geschichte_des_hochauflösenden_Fernsehens, erstellt am 10.5.2013, zuletzt besucht am 11.7.2013

[WH2-2013] Wikipedia: H.264. de.wikipedia.org/wiki/H.264, erstellt am 5.6.2013, zuletzt besucht am 11.7.2013

[WH5-2013] Wikipedia: High Efficiency Video Coding. de.wikipedia.org/wiki/High_Efficiency_Video_Coding, erstellt am 13.6.2013, zuletzt besucht am 16.7.2013

[WHK-2013] Wikipedia: Huffman-Codierung. http://de.wikipedia.org/wiki/Huffman-Kodierung, erstellt am 3.7.2013, zuletzt besucht am 11.7.2013

[WOF-2013] Wikipedia: Orthogonales Frequenzmultiplexverfahren. de.wikipedia.org/wiki/Orthogonales_Frequenzmultiplexverfahren, erstellt am 4.6.2013, zuletzt besucht am 11.7.2013

[WQA-2013] Wikipedia: Quadraturamplitudenmodulation. de.wikipedia.org/wiki/QAM, erstellt am 11.5.2013, zuletzt besucht am 11.7.2013

[WTD-2013] Wikipedia: Drift (Nachrichtentechnik). de.wikipedia.org/wiki/Temperaturdrift, erstellt am 5.4.2013, zuletzt besucht am 11.7.2013

[WTS-2013] Wikipedia: MPEG Transport stream. en.wikipedia.org/wiki/MPEG_transport_stream, erstellt am 20.6.2013, zuletzt besucht am 11.7.2013

[WWG-2013] Wikipedia: Walter Gerber. de.wikipedia.org/wiki/Walter_Gerber, erstellt am 5.7.2013, zuletzt besucht am 16.7.2013

[WZE-2013] Wikipedia: Zeilensprungverfahren. de.wikipedia.org/wiki/Zeilensprungverfahren, erstellt am 19.4.2013, zuletzt besucht am 11.7.2013