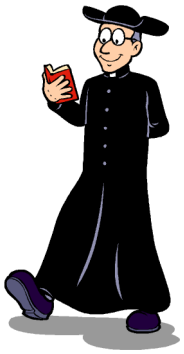

Brother John

Das Encodingwissen

DVD-Backup nach MPEG-4

Revision 35 vom 8. Dezember 2006
http://encodingwissen.brother-john.net

Dieses Dokument unterliegt der Creative-Commons-Lizenz
Namensnennung-NichtKommerziell-KeineBearbeitung 2.0 Deutschland.
http://creativecommons.org/licenses/by-nc-nd/2.0/de/deed.de
Copyright © Brother John 2004–2006

Inhalt

Vorwort und Wegweiser IV

Teil A
Hintergrundwissen 1

A.1 Grundlagen 3
A.1.1 Der Nutzen der Kompression 3
A.1.2 Die Videokompression 5
A.1.2.1 Intraframe-Kompression 5
A.1.2.2 Interframe-Kompression 8
A.1.3 Die Audiokompression 11
A.1.4 Bestandteile eines Films 14
A.1.5 Der Ablauf eines Encodings Schritt für Schritt 17

A.2 Der Formatedschungel 20
A.2.1 Video- und Audioformate 20
A.2.2 Das Format der DVD 25
A.2.3 Die Containerformate 27
A.2.4 Die Videocodecs 31
A.2.5 Die Audiocodecs 33
A.2.6 Die Untertitelformate 36

A.3 Das Videobild 39
A.3.1 Cropping 39
A.3.2 Anamorphes Video 40
A.3.2.1 Anamorphes Quellvideo 40
A.3.2.2 Anamorphes MPEG-4 46
A.3.3 Die passende Zielauflösung 51

A.4 Videocodec-Konfiguration 58
A.4.1 VfW-Interfaces 58
A.4.1.1 Xvid v1.2 VfW-Konfiguration 58
A.4.1.2 DivX 6.4 VfW-Konfiguration 65
A.4.1.3 x264 rev517 VfW-Konfiguration 68
A.4.2 Kommandozeilen-Interfaces 73

A.4.2.1 Xvid Kommandozeilen-Konfiguration 74
A.4.2.2 x264 Kommandozeilen-Konfiguration 75

Teil B
Praxiswissen 77

B.1 Vorarbeiten 79
B.1.1 Die nötige Software 79
B.1.2 DVD-Ripping 82
B.1.2.1 Ripping des Films 83
B.1.2.2 Ripping der Kapitelliste 86
B.1.3 Indexieren der VOBs mit DGIndex 86

B.2 Audio-Transcoding 89
B.2.1 BeSweet einrichten 89
B.2.2 Audio decodieren und bearbeiten 92
B.2.3 AAC-Encoding mit Nero und Winamp 95
B.2.4 Vorbis-Encoding 97
B.2.5 MP3-Encoding mit Lame 98
B.2.6 AC3-Encoding 99

B.3 Untertitel 101
B.3.1 Ins Bild eingebrannte Untertitel 101
B.3.2 Dynamisch einblendbare Untertitel 103

B.4 Die Encoding-Frontends 107
B.4.1 Gordian Knot 107
B.4.1.1 Gordian Knot einrichten 109
B.4.1.2 Kalkulieren der Bitrate 110
B.4.1.3 Vorbereiten des Videos 112
B.4.1.4 Dynamische Untertitel und Kapitel einbinden 115
B.4.1.5 Encoding-Parameter konfigurieren 116
B.4.1.6 Das Encoding starten 119
B.4.1.7 Akapumas Gordian-Knot-Personalisierer 123

B.5 Manuelles Muxing 135
B.5.1 MKVMerge für den Matroska-Container 135
B.5.2 AVI-Mux GUI für den AVI-Container 138

Teil C
Spezialthemen 139

C.1 Manuelles Splitting 141

C.2 DivX 5.2.1 VfW-Konfiguration 145

C.3 Die Bedeutung der ITU-R BT.601 für das PAR 148

Teil D
Anhang 159

Literatur 161
Abkürzungen 164
Changelog 167

Vorwort und Wegweiser

Wenn du deine geliebten DVDs in hoher Qualität sichern willst und dir dafür Xvid, DivX oder x264 vorschwebt, bist du hier genau richtig. Das Encodingwissen ist keine typische Schritt-für-Schritt-Anleitung, sondern konzentriert sich stark auf Hintergrundinfos. Das bedeutet deutlich mehr Lesestoff als das typische Encoding-Tutorial. Dafür bist du hinterher gerüstet, um auch komplizierte Projekte in den Griff zu kriegen.

Dieses Vorwort möchte ich auch als Gelegenheit nutzen, allen Lesern des Encodingwissens Danke zu sagen. Ohne euer Interesse, euer Feedback und eure Weiterempfehlungen wäre aus der ursprünglichen wirren Textdatei sicher nicht ein so umfangreiches und (hoffentlich ;-)) kompetentes Tutorial zum DVD-Backup geworden. Ganz besonderer Dank gilt Naito, der für die neue Ausgabe des gedruckten Encodingwissens den Text aus den HTML-Dateien ins OpenOffice-Dokument übertragen hat.

Das Encodingwissen ist extrem umfangreich. Alles wie einen Roman von vorne bis hinten durchzulesen, dürfte meistens nicht sinnvoll sein. Aber dann wo anfangen? Bei der Entscheidung versucht dieser Abschnitt eine Hilfestellung zu geben, je nachdem, welches Vorwissen du mitbringst.

Anfänger

Du bist neu, hast vielleicht noch nie einen Film encodiert? Dann zählen erst einmal Ergebnisse. Bevor du loslegst, solltest du auf jeden Fall die Hintergründe zu den Bestandteilen eines Films (A.1.4, Seite 14) und zum Encodingablauf (A.1.5, Seite 17) durchlesen, um eine grobe Vorstellung zu erhalten, wie eine Videodatei aufgebaut ist und welche Schritte nötig sind, um von der DVD zum fertigen Encoding zu gelangen. Dann stürze dich auf den Praxisteil (Teil B, Seite 77), wo ein komplettes Encoding Schritt für Schritt im Detail erklärt wird. Ein paar Mal musst du dazu sicherlich zurück zum Hintergrundabschnitt Formatedschungel (A.2, Seite 20) springen, um dich für einen Codec o. ä. zu entscheiden. Einfach den entsprechenden Links in den Praxiskapiteln folgen. Bei der Konfiguration der Audio- und Videocodecs kannst du ruhig erst einmal auf die empfohlenen Einstellungen vertrauen.

Auf- und Umsteiger

Die ersten Schritte sind gemacht und die ersten Encodings erfolgreich beendet? Oder kommst du aus der Ecke von (S)VCD und DVD-Authoring und willst dich auch mit der MPEG-4-Welt vertraut machen? Schwierig, hier eine Empfehlung zu geben. Sicher ist das Praxiswissen (Teil B, Seite 77) zentral wichtig – als Umsteiger auch das Kapitel zum Encodingablauf (A.1.5, Seite 17). Außerdem solltest du es nicht versäumen, dich mit der Videocodec-Konfiguration (A.4, Seite 58) eingehender zu beschäftigen. Der Rest des Hintergrundwissens ist hauptsächlich Geschmackssache. Ich persönlich würde mir den Grundlagen-Teil (A.1, Seite 3) bis zum Schluss aufheben, denn was dort besprochen wird, ist wirklich grundlegend. Fürs tägliche Encoding hat dieser Abschnitt wenig unmittelbaren Nutzen.

Profi

Du bist Profi mit mehrjähriger Encodingerfahrung und mit allen Wassern gewaschen? Meiner Erfahrung nach besonders interessant dürften die Kapitel rund um anamorphes Bild (A.3.2, Seite 40) und ITU-R BT.601 (C.3, Seite 148) sein, außerdem möglicherweise Akapumas Gordian-Knot-Personalisierer (B.4.1.7, Seite 123). Ansonsten solltest du das Inhaltsverzeichnis als erste Anlaufstelle benutzen und das Encodingwissen als Nachschlagewerk ansehen. Schließlich kann man nicht absolut jedes Detail immer im Kopf behalten.

Nachdem nun das Vorgeplänkel erledigt ist, bleibt mir nur noch, viel Spaß beim Lesen und Encodieren zu wünschen. Steigen wir ein in die Grundlagen!

Brother John

Teil A
Hintergrundwissen

Einleitung

In diesem ersten großen Abschnitt beschäftigen wir uns mit der hoffentlich nicht gar so grauen Theorie. Warum komprimieren wir Mediendaten überhaupt? Wie funktioniert so eine Kompression? Wie ist eine DVD grundlegend aufgebaut und welche Formate stehen für Video, Audio usw. zur Auswahl.

Etwas praktischer sind die Kapitel weiter hinten im Abschnitt. Dort werfen wir einen Blick darauf, in welcher Situation die eine oder andere Kombination von Formaten besser geeignet ist, wie ein DVD-Backup Schritt für Schritt abläuft, wie wir das Quellvideo passend fürs Encoding vorbereiten und wie wir unseren gewünschten Videocodec konfigurieren.

Das alles gehört zum nötigen Rüstzeug, um uns anschließend auf die praktische Arbeit stürzen zu können. Wer lieber gleich loslegen will, um dann zwischendurch die Hintergründe nachzulesen, kann mit dem Praxisabschnitt (Teil B) anfangen und von dort den Links zurück zu den zum Thema passenden Hintergrundkapiteln folgen.

A.1 Grundlagen

A.1.1 Der Nutzen der Kompression

Bevor wir anfangen, uns äußerst ausführlich mit der praktischen Seite von Audio- und Videokompression zu beschäftigen, sollten wir uns erst einmal darüber klar werden, warum das Schrumpfen überhaupt nötig ist. Ginge es nicht auch ohne Kompression?

Betrachten wir dazu einmal, wie groß die Datenmenge ist, die für einen durchschnittlichen Kinofilm anfällt. Dabei lassen wir unberücksichtigt, wie ein Film ursprünglich gedreht wird, sondern stellen uns vor, wir hätten ihn schon in einem passenden Format für die DVD – nur ohne Kompression. Zu berücksichtigen sind also eine Videospur und mindestens eine Audiospur. Untertitel und der Overhead des DVD-Formats gehören zwar genau genommen auch dazu, fallen aber größenmäßig nicht weiter ins Gewicht.

Platzbedarf unkomprimierter Videodaten

Ein einzelnes Bild des Videos besteht aus in Zeilen und Spalten angeordneten Pixeln, so dass eine rechteckige Fläche entsteht. Jedem einzelnen Pixel ist ein Farbwert zugeordnet, wobei sämtliche Farbtöne aus drei Grundfarben zusammengesetzt werden: Rot, Grün und Blau. Um der Genauigkeit des menschlichen Auges gerecht zu werden, benötigt jede der Grundfarben 1 Byte Speicherplatz, was insgesamt rund 16,7 Millionen verschiedene Farbtöne ermöglicht. Ein einzelnes Pixel verbraucht also 3 Byte Platz. Das gesamte Bild besteht aus 576 Zeilen mit jeweils 720 Pixeln, macht insgesamt 414 720 Pixel pro Bild. Das bedeutet einen Platzbedarf von

414 720 Pixel/Bild × 3 Byte/Pixel = 1 244 160 Byte/Bild (ca. 1,2 MByte/Bild).

Mit einem einzelnen Bild würden wir also schon fast eine komplette Diskette füllen. Nun hat ein Film aber mehrere Bilder, und zwar 25 Stück pro Sekunde. Das macht für einen 100-Minuten-Film (6 000 Sekunden) 150 000 Bilder. Der Platzbedarf beträgt dann

1 244 160 Byte/Bild × 150 000 Bilder = 186 624 000 000 Byte (ca. 174 GByte).

Allein das unkomprimierte Video füllt also eine handelsübliche Festplatte. Oder – um einen Vergleich zu bemühen, mit dem die Online-Generation vielleicht mehr

anfangen kann: Mit einem handelsüblichen DSL-2000-Breitbandanschluss und durchgehend voller Geschwindigkeit braucht man über eine Woche, um diese Datenmenge herunterladen.

Platzbedarf unkomprimierter Audiodaten

Der Ton braucht deutlich weniger Platz, verschärft die Situation aber doch noch ein Stück. Nehmen wir eine übliche Audiospur mit 6 Kanälen, 48 000 Abtastungen pro Sekunde (Hz) und 16 Bit (2 Byte) Auflösung. Das ergibt pro Sekunde

$$2\ \text{Byte} \times 48\,000\ \text{Hz} \times 6\ \text{Kanäle} = 576\,000\ \text{Byte/Sekunde}.$$

Eine Sekunde Audio füllt damit immer noch knapp eine halbe Diskette. Das Ganze auf unseren 100-Minuten-Film hochgerechnet ergibt

$$576\,000\ \text{Byte/Sekunde} \times 6\,000\ \text{Sekunden} = 3\,456\,000\,000\ \text{Byte (ca. 3,2 GByte)}.$$

Den Wert können wir noch mindestens verdoppeln, da kaum eine DVD nur eine einzelne Audiospur enthält. Ein kompletter unkomprimierter Film würde also grob 180 bis 200 GByte Speicherplatz beanspruchen. Dagegen wirkt die DVD mit ihrer Kapazität von 8 GByte geradezu winzig.

Verlustlose und verlustbehaftete Kompression

Um Filme zu einem vernünftigen Preis speichern zu können, müssen sie also komprimiert werden. Grundsätzlich stehen dafür zwei Möglichkeiten zur Verfügung.

Verlustlos (lossless)

Die Daten werden zusammengepresst, ohne dass dabei irgendeine Information verloren geht. Das dekomprimierte Video entspricht also exakt dem Original. Das Verfahren kennen wir von Packprogrammen wie Zip oder Rar. Technisch funktioniert es so, dass im Original nach sich wiederholenden Zeichenfolgen gesucht wird. Diesen Zeichenfolgen wird ein kürzerer Code zugeordnet und für jedes weitere Auftauchen nicht mehr die komplette Zeichenfolge, sondern nur noch der kurze Code gespeichert.

Verlustbehaftet (lossy)

In unkomprimierter Form enthält ein Film viel mehr Daten, als die menschlichen Sinnesorgane verarbeiten können. Das heißt, dass man einen gewaltigen Batzen an Informationen einfach wegwerfen kann, ohne dass es beim Anschauen und Zuhören auffällt. Die komprimierte Datei wird dadurch extrem klein – um Klassen kleiner als die verlustlos komprimierte Variante – stellt aber kein exaktes Abbild des Originals mehr dar. Das heißt, aus einer verlustbehaftet komprimierten Datei lässt sich das Original nie mehr exakt rekonstruieren. Ein Teil der Informationen ist endgültig verloren gegangen. Das stört aber nicht, solange nur die Informationen fehlen, die man sowieso nicht wahrnehmen würde.

Um einen Film auf eine brauchbare Größe einzudampfen, führt an der verlustbehafteten Kompression kein Weg vorbei. Praktisch wenden die Codecs aber beide Verfahren an: erst werden unnötige Informationen entfernt und das Ergebnis dann zusätzlich mit einem verlustlosen Verfahren komprimiert.

A.1.2 Die Videokompression

In diesem Kapitel betrachten wir etwas genauer, wie ein Codec einen Film bearbeitet. Keine Angst, es wird nicht mathematisch. Davon habe ich selbst keine Ahnung ;-). Grundsätzlich setzt sich ein Codec mit dem Film auf zwei Ebenen auseinander: einmal werden die Einzelbilder unabhängig voneinander bearbeitet, dann wird das Kompressionspotenzial aus der zeitlichen Abfolge der Bilder ausgeschöpft. Im Zusammenspiel mit einem verlustlosen Packalgorithmus entsteht so der komprimierte Film.

A.1.2.1 Intraframe-Kompression

Wie der Name schon sagt, beschäftigt sich die Intraframe-Kompression damit, das Schrumpfungspotenzial innerhalb eines einzelnen Bildes auszuschöpfen. Der Codec tut hier das, was JPEG mit einem einzelnen digitalen Foto macht. Und tatsächlich zeigen sich zwischen den Kompressionsverfahren der MPEG-Codecs und der JPEG-Methode deutliche Parallelen. Im Wesentlichen basiert die Intraframe-Kompression auf zwei Verfahren, die wir uns jetzt näher ansehen wollen.

Farbräume

Wie schon auf Seite 3 erwähnt, kombiniert ein Computermonitor sämtliche Farbtöne aus den drei Grundfarben Rot, Grün und Blau, wobei ein Pixel 3 Byte = 24 Bit Speicherplatz benötigt. An dieser Stelle setzt die erste Komprimierung an, die eine Besonderheit des menschlichen Auges ausnutzt. Für Farben sind wir nämlich weit unempfindlicher als für Kontraste (Helligkeitsunterschiede). Deshalb wird eine Farbe in einen Helligkeitsanteil (Luminanz) und einen Farbanteil (Chrominanz) zerlegt. Der weniger wichtige Chroma-Anteil wird mit geringerer Genauigkeit gespeichert und dadurch Platz gespart.

Es gibt eine ganze Reihe von Farbräumen, die nach diesem Prinzip arbeiten. Für die DVD wird YV12 verwendet, der mit 12 Bit Speicherplatz pro Pixel auskommt – das ist die Hälfte des Originals. Natürlich handelt es sich um eine verlustbehaftete Kompression, die sich allerdings nur manchmal bei Rot-Tönen bemerkbar macht. Die Farbunempfindlichkeit gilt offenbar für Rot weniger als für Grün und Blau, was bei der Entwicklung der MPEG-Standards entweder noch nicht bekannt war oder übersehen wurde. Deshalb neigen alle MPEG-Codecs bei Szenen mit hohem Rotanteil schon früh zur Bildung von Artefakten.

Makroblocks und DCT

Nach dem Wechsel des Farbraums können wir uns nun mit dem Inhalt des Bildes beschäftigen. Alle gängigen Codecs zerlegen ein Bild nicht einzelne Pixel, sondern in so genannte Makroblocks, die typischerweise 16 × 16 Pixel groß sind. Je mehr Bitrate (Speicherplatz) wir unserem Film gönnen, desto mehr bleibt pro Makroblock übrig und desto mehr Details können im Block erhalten bleiben. Sinkt die Bitrate zu weit, besteht ein Block im Extremfall nur noch aus einer einzelnen Farbe. Spätestens dann werden im Bild die »Riesenpixel« sichtbar, die man von schlechten Encodings kennt.

Die Entscheidung darüber, welche Details erhalten bleiben und welche verworfen werden, trifft eine mathematische Methode, die sich zwischen den Codec-Familien unterscheidet, jedoch immer nach dem selben Prinzip arbeitet. Wir betrachten die Kompression hier am Beispiel der MPEG-4-ASP-Codecs (Xvid, DivX), welche die diskrete Kosinustransformation (DCT) verwenden.

An dieser Stelle kommen die berühmt-berüchtigten Quantisierungsmatrizen und der Quantizer ins Spiel. Die Matrix ist der zentrale Baustein zur Durchführung der DCT. Sie besteht aus 64 Werten, angeordnet in 8 Zeilen zu je 8 Spalten. Je höher die enthaltenen Werte, desto stärker wird komprimiert und desto größer

ist der Informationsverlust. Der Quantizer dient zusätzlich als Multiplikator. Man könnte ihn als eine Art Kompressionsfaktor bezeichnen. Genau wie bei der Matrix gilt: höhere Werte vernichten mehr Details.

Wer tiefer in die Materie einsteigen will, sollte zuerst einen Blick auf den Artikel von Ethanolix und Videostation werfen, der sich im Anhang von Selurs »*Wissenswertes rund um Xvid*« findet. Nötig ist Detailwissen aber nur, wenn wir eigene Matrizen entwerfen wollen.

Unser Einzelbild besteht nun also aus einer Menge mehr oder weniger stark komprimierter Makroblocks, wodurch der Platzbedarf deutlich geschrumpft ist. Um das Bild später wieder anzeigen zu können, muss der Kompressionsvorgang rückgängig gemacht werden, wofür bei den ASP-Codecs die inverse diskrete Kosinustransformation (iDCT) zuständig ist. Da wir mit einer verlustbehafteten Kompression arbeiten, entspricht der wiederhergestellte Makroblock nicht exakt dem Original. Ob der Unterschied sichtbar ist – d. h. wie gut sich das Ergebnis ans Original annähert – hängt im Wesentlichen von der Höhe des Quantizers und vom Design der Quantisierungsmatrix ab. Weitere Einflüsse von außerhalb der eigentlichen DCT kommen natürlich dazu: verfügbare Bitrate, andere Codeceinstellungen, Art des Films usw.

Betrachten wir zum Schluss anhand Abbildung 1 das Ergebnis einer DCT-Codierung an einem 8 × 8 Pixel großen Block. Ganz links sehen wir stark vergrößert den ursprünglichen Block, rechts daneben die per JPEG komprimierte Version. Da JPEG auch auf der DCT aufbaut, kommt das Ergebnis einer MPEG-4-ASP-Kompression recht nahe. Besonders in der linken unteren Ecke sind deutlich die Unterschiede zum Original zu erkennen, allerdings nur in der Vergrößerung. Die Blocks in Originalgröße (rechts) sehen sich schon zum verwechseln ähnlich.

Abbildung 1: Ursprüngliche und DCT-komprimierte Blöcke im Vergleich.

Wir können jetzt also einzelne Bilder komprimieren und wiederherstellen, was für ein Video prinzipiell schon ausreicht. Nichts hindert uns daran, die JPEG-artigen Bilder in einer rasanten Diashow hintereinander zu hängen. Tatsächlich gibt es einen Codec namens Motion-JPEG, der genau das tut. Allerdings lässt sich die Dateigröße mit zusätzlichen Methoden noch immer deutlich schrumpfen.

A.1.2.2 Interframe-Kompression

Bisher haben wir uns nur damit beschäftigt, wie ein einzelnes Bild des Videos encodiert wird, ohne gleichzeitig auf die restlichen Bilder zu achten. Das tun wir jetzt bei der Interframe-Kompression, die identischen Inhalt über mehrere Bilder hinweg sucht und dadurch weiter Dateigröße einspart. MPEG-Codecs kennen dafür drei verschiedene Arten von Bildern, die wir uns nacheinander ansehen.

I-Frames

Codieren wir ein Bild rein mit den Methoden aus dem letzten Abschnitt, handelt es sich um ein Intraframe (kurz I-Frame, auch Keyframe genannt), ein vollständiges Einzelbild. Wie ein JPEG-Foto kann ein Intraframe für sich alleine existieren, d. h. um es zu decodieren, werden keine Informationen aus anderen Bildern benötigt. Man kann also sagen, dass für I-Frames keine Interframe-Kompression stattfindet.

Entsprechend belegen sie von allen Frametypen am meisten Platz, enthalten aber auch das qualitativ hochwertigste Bild. Außerdem sind I-Frames zum Spulen und Schneiden des Films wichtig. Dazu aber mehr im Praxisteil des Encodingwissens.

P-Frames

Klassisches Beispiel für die Interframe-Kompression sind die die *Predicted Frames* (kurz P-Frames). Stellen wir uns einen Nachrichtensprecher vor, der vor einem statischen Hintergrundbild seinen Text vorliest. Der Großteil dieser Szene bleibt über einen längeren Zeitraum unverändert. Die größte Bewegung geht von den Lippen des Sprechers aus.

Gäbe es nichts anderes als I-Frames, müssten wir in jedem einzelnen Bild immer wieder all die Informationen abspeichern, die sich überhaupt nicht ändern – eine riesige Platzverschwendung. Deswegen speichert ein P-Frame diese Infos nicht mehr, sondern verweist einfach auf das vorangehende Bild. Das war aber noch nicht alles. Schließlich könnte ein Teil des Bildes auch gleich bleiben, sich aber an eine andere Position bewegen. Stellen wir uns ein Auto vor, das von links nach rechts durchs Bild fährt. Auch solche »bewegten Gleichheiten« werden von P-Frames erfasst.

Da der Codec weiterhin auf der Basis von Makroblocks arbeitet, stellt sich die Situation so dar: »Makroblock A ist sowohl in Bild 1 als auch 2 unverändert vorhanden, nur wandert er von Position X in Bild 1 auf Position Y in Bild 2«.

Wir vermeiden Verschwendung, indem wir den Block nicht ein zweites Mal in Bild 2 speichern. Was wir allerdings speichern müssen, ist der Bewegungsvektor, also die Wanderbewegung. Beim Abspielen kann der Decoder den Block aus dem alten Bild holen und mit Hilfe des Vektors an die richtige neue Position setzen.

Ein auf diese Weise zusammengebautes P-Frame ist kein vollständiges Einzelbild mehr, sondern es speichert grob gesagt nur den Unterschied zum vorangehenden Bild. Welche Konsequenzen das besonders beim Abspielen des Films hat, sehen wir anhand der kurzen Bildsequenz aus einem I-Frame und drei P-Frames in Abbildung 2.

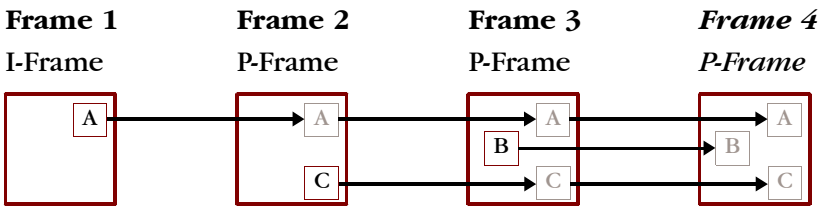

Abbildung 2: Rückbeziehung auf Blocks aus früheren Bildern bei P-Frames.

Wir laden den Film und möchten sofort Frame 4 am Bildschirm anzeigen. Kein Problem: Ein paar neue Makroblocks sind dort sowieso gespeichert und werden direkt aus Frame 4 decodiert. Makroblock A ist nur gewandert. Dessen Daten stehen also in Frame 3. Denkste! Nummer 3 ist auch ein P-Frame und enthält den lapidaren Hinweis: »Makroblock A ist unverändert aus Frame 2 übernommen«. Auch Frame 2 enthält einen analogen Hinweis, so dass wir den Makroblock A schließlich in Frame 1 finden. Beim anschließenden Makroblock B haben wir mehr Glück. Der findet sich schon in Frame 3. Makroblock C könnte seinen Ursprung in Frame 2 haben.

Weiter als bis zu Frame 1 müssen wir mit Sicherheit nie zurückspringen, denn dabei handelt es sich um ein I-Frame, ein vollständiges Bild, das keine Verweise auf frühere Bilder enthält.

Damit wird klar: Wenn wir an einer beliebigen Stelle in den Film hineinspringen und P-Frame 4 erwischen, lässt sich das nur dann mit Sicherheit vollständig decodieren, wenn sämtliche Frames bis zum vorangehenden I-Frame vorhanden sind. Dem Codec bleibt dann nichts anderes übrig als zu Frame 1 zurückzugehen und von dort aus alle Frames zu decodieren, bis er beim gewünschten Bild ange-

kommen ist. Das Splitting-Kapitel C.1 ab Seite 141 im letzten Abschnitt des Encodingwissens ist ein guter Punkt, um sich an diese Tatsache wieder zu erinnern.

Wenn der Film ganz gewöhnlich von vorne bis hinten durchläuft, existiert dieses Problem nicht. Bevor Frame 4 an die Reihe kommt, wurde schließlich Frame 3 am Bildschirm angezeigt und musste dafür vollständig decodiert werden. Auf diese vollständige Version kann der Codec jetzt für Frame 4 zurückgreifen. Ein weiteres Zurückspringen wird dadurch unnötig.

B-Frames

Wenden wir uns dem dritten Typ Frame zu, dem *Bidirectional Frame* (kurz B-Frame). Dabei handelt es sich prinzipiell um ein erweitertes P-Frame, das nicht nur Verweise auf vorangehende Bilder enthalten kann, sondern auch Verweise auf nachfolgende. An der Bildsequenz in Abbildung 3 wird das deutlicher.

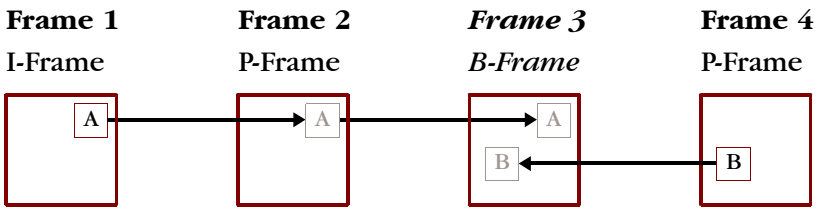

Abbildung 3:Bidirektionale Rück- und Vorwärtsbeziehung bei B-Frames.

Wir wollen direkt Das B-Frame 3 decodieren und finden für Makroblock A den Eintrag: »Aus Frame 2 übernommen«. Das kennen wir. So machen das normale P-Frames auch. Der Eintrag für Makroblock B lautet: »Aus Frame 4 übernommen«. Das ist das bidirektionale am B-Frame. Die Bidirektionalität ist auch der Grund dafür, dass B-Frames in der Regel höher komprimiert werden können als einfache P-Frames. Gut für uns, denn so belegt die gleiche Bildqualität weniger Speicherplatz.

Das offensichtliche Problem ist für den Zuschauer unsichtbar: Frame 3 kann sich schlecht auf Frame 4 beziehen, wenn Frame 4 noch überhaupt nicht vorhanden ist. Ein Film mit B-Frames kann also nicht strikt sequenziell bearbeitet werden. Unsere kleine Bildfolge müsste der Codec in der Reihenfolge 1, 2, 4, 3 encodieren und in dieser Reihenfolge auch wieder decodieren. Für die Anzeige am Bildschirm muss diese Vertauschung rückgängig gemacht werden. Der Decoder bearbeitet dafür nach dem Anzeigen von Nummer 2 erst Frame 4 und dann mit dessen Hilfe Frame 3. Solange Frame 3 angezeigt wird, parkt Frame 4 im Arbeitsspeicher.

Auswahl der Frametypen

Für welches Bild welcher Frametyp am günstigsten ist, müssen wir zum Glück nicht selbst entscheiden. Das erledigt der Codec automatisch und heutzutage auch zuverlässig. Einstellen müssen wir in der Regel nur, wie oft ein I-Frame erzwungen wird und ob überhaupt B-Frames verwendet werden sollen. Die Auswahl des Codecs richtet sich dann nach zwei Grundregeln:

- I-Frames stehen – außer nach dem vom Benutzer festgelegten zwingenden Intervall – nur, wenn zwei hintereinander folgende Bilder komplett verschieden sind und Gemeinsamkeiten nur zufällig wären. Bestes Beispiel dafür ist ein Szenenwechsel.
- B-Frames eignen sich besonders für ruhige Szenen mit wenig Bewegung, denn dann bestehen zwischen einer ganzen Reihe von Bildern kaum Unterschiede und die Vorteile der Bidirektionalität kommen besonders gut zum Tragen.

Und damit haben wir eine vollständig codierte Videospur. Wir können uns also dem Ton zuwenden. Aber keine Angst, der lässt sich schneller abhandeln als das Bild.

A.1.3 Die Audiokompression

Genauso wie fürs Bild existieren auch für den Ton sowohl verlustlose als auch verlustbehaftete Kompressionsverfahren. Ebenso verwenden die gängigen Codecs beide Methoden, um die maximal mögliche Kompression zu erzielen.

Da die Details genauso schnell wie beim Bild in die höheren Gefilde von Technik und Mathematik abdriften, soll hier eine kurze Beschreibung der prinzipiellen Methoden genügen. Schließlich wollen wir irgendwann ja auch tatsächlich dem ersten Film an die Gurgel gehen, oder? :-)

Maskierung

Ansatzpunkt sind wieder einmal die beschränkten Fähigkeiten der menschlichen Sinnesorgane. Im Fall der Maskierung geht es darum, dass das Ohr manche Töne nicht wahrnimmt, weil sie von einem ähnlich klingenden und/oder lauteren Ton

überlagert werden. Der Audiocodec versucht anhand eines psychoakustischen Modells solche Überlagerungen zu erkennen und nur die Töne zu speichern, die tatsächlich hörbar sind.

Leise Töne

Jedes Audioformat – egal ob analog oder digital – enthält einen gewissen Anteil an Rauschen. Je leiser ein Ton ist, desto geringer unterscheidet er sich von diesem Grundrauschen, bis er schließlich völlig darin untergeht und unhörbar wird. Solche Töne kann der Codec natürlich weglassen, ohne dass es zu hörbaren Qualitätseinbußen kommt.

Hohe Frequenzen

Die auf der DVD übliche Samplingrate erlaubt es, Tonhöhen bis zu ca. 24 kHz zu speichern, was die Hörfähigkeit der meisten Menschen deutlich übersteigt. Kinder hören in der Regel sehr gut (grob bis 20, vielleicht auch 22 kHz). Bis ins Erwachsenenalter sinkt dieser Wert deutlich bis in die Region um 15 – 17 kHz und kann noch deutlich weiter zurückgehen, je näher die Rente rückt. Deswegen können meine Eltern seelenruhig vor ihrem uralten Fernseher sitzen, während ich das Teil durch die geschlossene Tür bis auf den Flur grauenhaft pfeifen höre.

Dazu kommt, dass ein isolierter hoher Ton viel einfacher auszumachen ist als einer, der sich in den vielen anderen Tönen einer Filmtonspur versteckt. Auch hier besteht also Einsparpotenzial.

Kanalgemeinsamkeiten

Eine Tonspur besteht nicht aus einem einzelnen Kanal, sondern in den meisten Fällen entweder aus zwei (Stereo), sechs (5.1) oder sieben (6.1). Zwischen den Kanälen bestehen dabei mehr oder weniger starke Gemeinsamkeiten, die sich für die Kompression ausnutzen lassen (Channel Coupling).

MP3 z. B. tut das unter dem Begriff Joint Stereo. Dabei werden die Daten der Kanäle aufgeteilt in eine gemeinsame und eine unterschiedliche Komponente. Die Gemeinsamkeiten speichert der Codec nur einmal für alle Kanäle, den unterschiedlichen Anteil separat für jeden Kanal. Das Ergebnis ist eine kleinere Datei.

Gerade Joint Stereo ist als Qualitätskiller in Verruf geraten, was weniger an der

Methode an sich als an der schlechten Implementierung mancher Codecs liegt. Modernes und anständig programmiertes Channel Coupling, wie es z. B. Lame und AC3 verwenden, arbeitet nahezu oder gar komplett verlustlos.

Neben diesen Standardverfahren verwenden verschiedene Codecs noch andere Methoden, um weiter zu komprimieren. Und natürlich ist dem verlustbehafteten Durchlauf immer zusätzlich ein verlustloser Packer nachgelagert, der die Datei noch eine Ecke weit schrumpft.

Variable und konstante Bitrate

Wie brutal der Audiocodec Details entfernen muss, hängt neben der Komplexität des Audiomaterials entscheidend davon ab, wie viel Speicherplatz wir der Datei gönnen. Üblicherweise wird der als Bitrate in *Kilobit pro Sekunde* (kbit/s) angegeben. Dabei gibt es drei verschiedene Modi.

Constant Bitrate

CBR verteilt die Bitrate ohne Schwankungen gleichmäßig über die Datei. Bei 128 kbit/s erhält jede Sekunde auch diese 128 kbit, egal ob mehr nötig wären oder weniger ausreichen würden. Deswegen hat CBR mit voller Berechtigung mit dem Vorwurf zu kämpfen, ineffizient zu arbeiten. Allerdings ist das Verfahren sehr simpel und war deswegen besonders in der Anfangszeit der digitalen Kompression beliebt. Aktuelle Codecs setzen allerdings aus gutem Grund stark auf VBR.

Variable Bitrate

VBR hat als Ziel nicht eine gleichmäßige Datenrate, sondern konstante Qualität. Je nach Codectechnologie darf die Bitrate im Verlauf des Tracks beliebig schwanken. So können schwierig zu komprimierenden Stellen viele Bits zugeteilt werden und weniger komplexen Stellen wenig Bits; im Idealfall immer genau so viel, um die gewünschte Qualität zu erhalten. Nachteil dieser Methode ist, dass man die genaue Dateigröße vor dem Encoding nicht kennt. Entsprechend kann man einem VBR-Codec auch keine Bitrate angeben, sondern ein Qualitätslevel. Die Bitrate einer VBR-Datei ist lediglich das Rechenergebnis aus der Dateigröße geteilt durch die Spielzeit.

Average Bitrate

ABR versucht einen Kompromiss zwischen VBR und CBR. Zwar ist die Bitrate nicht mehr exakt festgelegt wie bei CBR, sondern darf schwanken. Allerdings achtet der Codec darauf, im gesamten Durchschnitt die angegebene Datenrate zu

erreichen. Ergebnis ist eine Datei, die die gewünschte Dateigröße recht genau trifft, deren Datenrate aber weniger freizügig als bei echtem VBR schwankt, also weniger effizient komprimiert ist.

Im IT-Bereich bereitet die Umrechnung zwischen Kilo, Mega, Giga usw. allgemeines Kopfzerbrechen. Das liegt daran, dass es keinen festgeschriebenen Standard gibt, also prinzipiell jeder rechnen kann, wie er will. Allgemein üblich und weitläufig als richtig akzeptiert ist diese Rechenregel:
Wenn es um *Bit* geht, dann gilt 1 Megabit = 1000 Kilobit = 1000 Bit. Also immer der Faktor 1000. Wenn es um *Byte* geht, dann gilt 1 Megabyte = 1024 Kilobyte = 1024 Byte. Also immer der Faktor 1024.
Eine wichtige Ausnahme sollte man dabei immer im Kopf haben. Festplatten- und Rohlinghersteller verwenden für ihre Kapazitätsangaben abweichend vom Üblichen die 1000er-Regel. Auf diese Weise kann man einen DVD-Rohling mit einer Kapazität von 4,7 Gigabyte auszeichnen, obwohl jedes übliche Betriebssystem die Disc nach der 1024er-Regel mit 4,37 Gigabyte erkennt.

A.1.4 Bestandteile eines Films

Ein digitaler Film ist kein einziger großer Batzen Daten, sondern ein gut geschnürtes Paket einzelner Datenhäppchen. Zu verstehen, was diese einzelnen Happen bedeuten und wie sie zusammenspielen, ist einer der wichtigsten Punkt an der ganzen Thematik des digitalen Videos.

Datenspuren

Die wichtigsten Elemente eines Films sind die Datenspuren. Jede Spur ist ein abgeschlossenes Ganzes und besitzt einen genau definierten Typ. Auf der allgemeinsten Ebene fällt der in eine der drei Kategorien Video, Audio oder Untertitel. Diese Anforderung an die Eindeutigkeit setzt sich auch im Detail fort. Z. B. kann eine Audiospur kein Mix aus MP3 und Vorbis sein, sondern muss sich auf ein Format beschränken.

Unser typischer Film am Ende des Encodings besteht aus einer Videospur im MPEG-4-Format, ein bis zwei Audiospuren und vielleicht ein bis zwei Untertitelspuren, die alle entweder parallel abgespielt oder übersprungen werden. Diese einfache Struktur lässt sich schnell verkomplizieren. Dazu brauchen wir nur an die Menüs der DVD denken. Dort müssen die zum Menü gehörenden Audio- und

Videoelemente zum einen miteinander verknüpft werden. Außerdem muss ein System existieren, das Interaktionen definiert. Sonst wüsste der Player nicht, welcher Teil des Bilds ein Button ist, auf den man drücken kann.

Metadaten

Ganz allgemein sind Metadaten »Daten über Daten«, also Daten, die andere Daten näher beschreiben. Der typischste Fall für einen digitalen Film ist die Kapitelliste der DVD. Diese bietet ja Zusatzinformationen zur Videospur. Ähnlich typisch ist die Angabe der Sprache einer Spur. Moderne Container bieten darüber hinaus ein umfangreiches System an Metadaten an, das weite Anwendungsbereiche abdeckt. Da man einen Film aber nach wie vor in erster Linie startet, ansieht und die Datei wieder schließt, werden diese Möglichkeiten wenig genutzt und entsprechend von den Softwaretools nicht allzu enthusiastisch unterstützt.

Container und Interleaving

Jeder Anfänger stolpert recht bald über den Begriff *Container*. Die meisten dürften keine genau Vorstellung haben, was damit gemeint ist. Dass ein Film aus einzelnen Spuren und zusätzlichen Metadaten besteht, haben wir gerade gesehen. Wäre das alles, müssten wir uns pro Spur mit einer eigenen Datei herumschlagen. Und obwohl man das manchmal bei AVI und Untertiteln beobachten kann, ist es unschön, unpraktisch und lästig. Deswegen verpackt man die Spuren in einen Container.

Bildlich gesprochen stellt der Container die Schuhschachtel dar, in der Filmrolle und Tonband aufbewahrt werden, also die Verpackung um Bild und Ton außen herum. Das Format des Containers lässt noch nicht unbedingt auf das Format des Inhalts schließen. Begriffe wie *AVI-Video* sind streng genommen nicht richtig und auch nicht besonders aussagekräftig: AVI ist kein Video-, sondern ein Container-Format. Innerhalb der AVI können Bild- und Tonspuren verschiedenster Formate liegen. Möglichkeiten für das Bild sind natürlich Xvid, DivX und x264, aber auch Huffyuv oder Theora und viele andere. Analog gilt die Vielfalt für den Ton. Von MP3 bis AC3 ist eine ganze Reihe von Formaten denkbar. Exakt müsste man also beispielsweise

Abbildung 4: Schubkarton-Vergleich

von *Xvid und MP3 in AVI* sprechen. Da das heftig umständlich ist, werden im Alltag die Begriffe wild durcheinander geworfen. Man sollte sich trotzdem immer im Klaren darüber sein, was eigentlich dahinter steckt.

So nett der Schuhkarton-Vergleich auch ist, er beinhaltet die Gefahr, dass man sich die Spuren parallel nebeneinander in einer Art Schachtel liegend vorstellt. Etwa so wie oben in Abbildung 4. Das ist in der Praxis unmöglich. Ein Filmcontainer ist nichts anderes als eine ganz normale Computerdatei, im Wesentlichen also eine schier endlose Abfolge von Nullen und Einsen. Von Parallelität keine Spur. Daten können in einer Datei eben nur *hintereinander* gespeichert werden. Korrigieren wir also das Bild durch das kleine Beispiel eines Films mit einer Videospur (100 MB) und zwei Audiospuren (20 und 15 MB). Dass die Containerstruktur selbst Speicherplatz braucht, wollen wir vernachlässigen. Die einfachste denkbare Art, einen solchen Film zu verpacken, wäre die folgende: die Datei beginnt mit den Videodaten, dann folgt die erste und schließlich die zweite Audiospur, alles brav hintereinander wie in Abbildung 5.

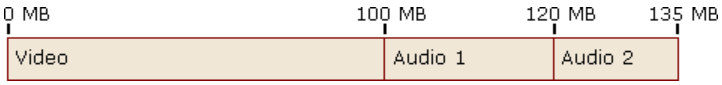

Abbildung 5: Physisches Containerlayout ohne Interleaving.

Es ist praktisch möglich, einen Film so zu speichern. Nur hat diese Methode einen entscheidenden Nachteil. Stellen wir uns vor, wir spielen den obigen Film von einer CD ab. Was würde passieren?

Der Player fängt an das Video zu spielen, decodiert die ersten Frames, und das Laufwerk liest von der Disc die nötigen Daten ein, die ganz am Anfang der Datei, also auch am Anfang der Disc liegen. Gleichzeitig soll die Tonspur anlaufen. Das Laufwerk muss nun seinen Lesekopf 100 MB weit nach hinten positionieren, Audiodaten lesen, flott wieder zurückspringen, um die nächsten Videodaten nicht zu verpassen, genauso flott zurück zur Audiospur und immer so weiter.
Auch wenn das Laufwerk wahrscheinlich schnell genug ist, um diese Tortur mitzumachen, ist es weder materialschonend noch effizient. Von der Geräuschentwicklung ganz abgesehen. Liegt der Film auf der Festplatte, passiert genau das Gleiche, nur fällt es dort weniger auf, weil eine Festplatte deutlich schneller und in der Regel leiser arbeitet als ein CD-Laufwerk.

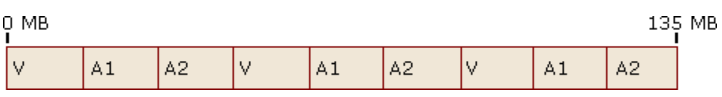

Abbildung 6: Physisches Containerlayout mit Interleaving.

Die Lösung des Problems ist reichlich simpel: Wir zerlegen alle Spuren in kleine Happen und hängen die abwechselnd aneinander. Das Verfahren nennt sich Interleaving (vgl. Abbildung 6).

In Wirklichkeit sind die Happen deutlich kleiner als im Bild. Eine halbe Sekunde wäre eine übliche Länge. Läuft alles ideal, liest das Laufwerk die ersten Videodaten und erreicht genau dann mit dem Lesekopf die Grenze zum ersten Audiohappen, wenn der Player die ersten Audiodaten verlangt. Der Kopf muss nicht neu positioniert werden. Natürlich wird dieser Idealfall in der Praxis nicht erreicht. Kleinere Repositionierungen sind immer nötig. Auf solche Kleinigkeiten sind die Laufwerke aber ausgelegt.

A.1.5 Der Ablauf eines Encodings Schritt für Schritt

Bei einem so komplexen Prozess wie dem DVD-Backup ist es nicht immer ganz einfach, den Überblick zu behalten. Deswegen betrachten wir in diesem Kapitel die Schritte des ganzen Prozesses in aller Kürze.

DVD-Ripping

Bevor wir mit dem Film irgend etwas anstellen können, müssen wir ihn erst einmal von der DVD auf die Festplatte übertragen. »Ripping« nennt sich der Vorgang deshalb, weil wir in aller Regel nicht die komplette DVD kopieren, sondern weglassen, was wir zum Encoding nicht brauchen. Menüs und Extras gehören nahezu immer dazu. Auch Kopierschutzmaßnahmen werden beim Ripping oft entfernt. Da sich die Politik leider nur allzu bereitwillig von der milliardenschweren Medienindustrie bestechen lässt, ist das inzwischen leider in einer ganzen Reihe von Ländern – einschließlich Deutschland – illegal.

Das Programm, das wir im Encodingwissen zum Rippen verwenden, heißt vStrip. Andere beliebte Ripper sind der DVDFab Decrypter, der inzwischen tote DVD Decrypter (den Macrovision auf dem Gewissen hat) und der schon recht betagte SmartRipper.

Als »Ripping« wird oft der komplette Prozess von der Quell-DVD bis zum fertigen MPEG-4-Film bezeichnet, der dann »DVD-Rip« heißt. Das ist zwar weit verbreitet, allerdings (um mal ordentlich Haare zu spalten) nicht ganz richtig, weil *Ripping* lediglich das Weglassen unnötiger Teile der DVD bezeichnet.

Indexing und Audio-Demuxing

Video, Audio und Untertitel der DVD stecken nach dem Ripping wie auf der DVD in VOB-Containerdateien. Wir verwenden DGIndex aus dem DGMPGDec-Paket, um die Audiospuren zu extrahieren und einen Index der Videospur anzulegen, der dann von AviSynth weiterverwendet werden kann.

Audio-Transcoding

Sind die originalen AC3-Dateien der DVD zu groß für die gewünschte Zielgröße, müssen wir sie verkleinern. Fürs manuelle Transcoding ist BeSweet zuständig, und der Einfachheit halber verwenden wir BeLight als grafische Oberfläche. Je nach Encoding-Frontend kann der Audioschritt auch entfallen, wenn die Audiounterstützung des Frontends unseren Ansprüchen genügt.

Untertitel-Vorbereitung

Bisher stecken die Untertitel noch in den VOB-Containerdateien. Um sie im fertigen Encoding einzubinden extrahieren wir sie in diesem Schritt mit Vobsub oder VSRip. Eventuell steht auch eine Umwandlung vom Vobsub-Bildformat in Text an, was SubRip erledigt.

Videocodec-Konfiguration

Extrem wichtig ist ein gut konfigurierter Videocodec, denn von diesem hängt die endgültige Bildqualität entscheidend ab. Welche Form der Konfiguration auf uns zukommt, hängt vom Encoding-Frontend ab.

Video: AviSynth-Skript

Das AviSynth-Skript fasst mehrere Arbeitsschritte zusammen. Insgesamt geht es um alles, was mit dem Bearbeiten des Bilds zu tun hat. Dazu gehört, die schwarzen Balken wegzuschneiden und auf die richtige Zielauflösung zu skalieren. Außerdem kümmert sich AviSynth um alle Filter, die z. B. Bildrauschen beseitigen oder interlaced Frames in ein gut encodierbares progressives Bild umwandeln.

Encoding

Das Encoding ist der Vorgang, der unser Zielvideo erstellt. Dabei rechnet der Videocodec das Quellmaterial der DVD ins MPEG-4-Format um und trimmt es auf die gewünschte Größe. Um optimale Qualität zu erhalten und Gleichzeit die Zielgröße nicht zu verpassen, findet das Encoding in zwei Durchläufen statt. Im ersten Durchgang (1st Pass) analysiert der Codec Bild für Bild die Komplexität des kompletten Films. Eine Videodatei wird erst im zweiten Durchgang (2nd Pass) erstellt, in dem anhand der gesammelten Daten die Bitrate verteilt wird. Manche

Codecs unterstützen auch mehr als zwei Durchgänge, die dann *N^th Passes* genannt werden. Auch hier erhalten wir ab dem zweiten Durchlauf ein funktionsfähiges Video. Allerdings werden im 2nd Pass wieder Daten gesammelt, mit denen ein dritter, nochmals optimierter Durchgang erfolgen kann.

Muxing

An dieser Stelle haben wir alle Einzelteile des Films in fertiger Form vorliegen. Was bleibt, ist sämtliche Spuren in einen Container zu verpacken und mit Metadaten zu versehen. Der Vorgang nennt sich *Multiplexing* oder kurz *Muxing*. Je nach Encoding-Frontend und den Formaten der Datenspuren müssen wir beim Muxing selbst Hand anlegen oder können uns auf die Automatik des Frontends verlassen.

Splitting

Dieser letzte Schritt stirbt dank des Siegeszugs der DVD-Brenner langsam aus. Wenn wir traditionell auf CD brennen und als Zielgröße mehrere CDs gewählt haben (z. B. 1400 MB für zwei CDs), müssen wir den fertigen Film in Happen aufteilen, die auf eine CD passen. Normalerweise erledigen wir das zusammen mit dem Muxing.

A.2 Der Formatedschungel

Wer neu in die digitale Medienwelt einsteigt, wird erst einmal erschlagen. Für Video, Audio, Untertitel und Container gibt es jeweils eine ganze Latte an Standards und Formaten, die teilweise zueinander inkompatibel sind und sich für die eine oder andere Art der Wiedergabe besser oder schlechter eignen. Um dem ganzen die Krone aufzusetzen, wird oft ein Standard von mehreren Codecs bedient. So erzeugen sowohl Xvid als auch DivX MPEG-4-ASP-Video. Wer soll sich da noch auskennen?

Um ein wenig Licht ins Dunkel zu bringen, werfen wir in diesem Kapitel einen intensiven Blick auf das ganze Chaos, das nur auf Anhieb dermaßen chaotisch wirkt. Wenn der erste Schock überwunden ist, erkennt man schnell, dass die Formatevielfalt recht gut strukturiert ist. Die meiste Überlegung ist immernoch nötig, wenn wir die Kombination einiger Formate aussuchen, die im endgültigen Film verwendet werden soll.

A.2.1 Video- und Audioformate

Dieses Kapitel beschäftigt sich nicht mehr allgemein mit Kompressionstechniken, sondern mit konkreten digitalen Formaten für Audio und Video. Allerdings betrachten wir nur die wichtigsten, denn eine komplette Liste aller aktueller Audio- und Videoformate würde für sich ein ganzes Buch füllen.

Die MPEG-Videostandards

MPEG ist eine Abkürzung und steht für *Moving Picture Experts Group*, (nicht Motion Picture Experts Group, wie oft zu lesen ist) ein Gremium, das verschiedene Standards zur Codierung von digitalem Video, Audio und den dazugehörigen Ergänzungen (z. B. Containerformate, Interaktivität) erarbeitet.

MPEG-1

MPEG-1 (ISO/IEC 11172) ist der älteste Standard, in der ersten Version 1993 verabschiedet. Der Videoteil ist im *Part 3* des Standards definiert. Verwendung findet MPEG-1 für die Video-CD und bei vielen im Internet angebotenen Videos, die größtmögliche Kompatibilität erreichen wollen. MPEG-1 ist nämlich erstens so weit verbreitet, dass ihn nahezu jeder Computer abspielen kann, und stellt zweitens so geringe Anforderungen an die Rechenleistung, dass die Videos auch auf veralteten Computern gut laufen. Nachteil ist, dass mit steigender Kompression die Qualität schnell deutlich abnimmt.

MPEG-2

Seit 1994 existiert MPEG-2 (ISO/IEC 13818). Der Video-Teil ist im *Part 2* definiert. Was die Verbreitung angeht, steht MPEG-2 seinem Vorgänger MPEG-1 nicht nach. Schließlich sind DVDs in diesem Format codiert. MPEG-2 erreicht eine ordentliche Kompression. Die mehr als 8 GB Speicherplatz der DVD tun dann ihr Übriges, um für erstklassige Qualität zu sorgen.

MPEG-4

Der Untertitel dieses Guides sagt es schon: MPEG-4-Video ist unser Zielformat. Der Standard wurde 1998 als ISO/IEC 14496 verabschiedet. Der Video-Teil ist hier in zwei Bereiche aufgeteilt:

- **MPEG-4 Part 2** enthält mehrere Profile, von denen das bekannteste sicher das Advanced Simple Profile (ASP) ist. Xvid und DivX benutzen diesen Teil des Standards.
- **MPEG-4 Part 10** ist besser bekannt als Advanced Video Coding (AVC) oder unter seiner Dokumentnummer der ITU-Organisation: H.264. Verwendet wird AVC z. B. von Nero Digital und x264.

Gegenüber MPEG-2 lässt sich in der gleichen Dateigröße noch einmal deutlich mehr Qualität unterbringen, wobei AVC wiederum tendenziell bessere Ergebnisse liefert als ASP. Deshalb können wir einen Film meistens ohne extreme Verluste auf ⅓ bis ½ DVD-5 (normaler Rohling mit 4,37 GB Kapazität) eindampfen.

MPEG-4 bitte nicht mit MP4 verwechseln. Das erste ist die Bezeichnung des kompletten Standards, das zweite die Dateiendung des MPEG-4-Containerformats (MPEG-4 Part 14).

Implementierungen der Standards

Die MPEG-Standards allein bringen uns dem codierten Video noch nicht näher. Das MPEG-Gremium programmiert keine Codecs, sondern definiert nur, wie ein gültiger Videostream der entsprechenden MPEG-Version auszusehen hat. Daraus ergibt sich auch grundsätzlich, welche Methoden beim Encoding angewendet werden können und welche nicht. Die Details der Codierung bleiben dann der Phantasie der Codec-Programmierer überlassen. Alle Tricks sind erlaubt, solange das Endergebnis den Vorgaben der verwendeten MPEG-Version entspricht.

Diese Tatsache führt dazu, dass z. B. Xvid und DivX zwei unabhängige und verschiedene Codecs sind, die aber beide Videos nach dem MPEG-4 Advanced Simple Profile erzeugen. Deshalb interessiert sich ein guter Decoder auch nicht dafür, welcher Codec nun das Video erstellt hat (inzwischen funktioniert das auch praktisch in dieser Form). Denn das Format des Bitstroms ist bei beiden exakt das selbe. Probleme können höchstens Decoder bereiten, die MPEG-4 ASP nicht vollständig unterstützen; z. B. mehrere B-Frames hintereinander nicht korrekt erkennen. Solche Einschränkungen existieren am Computer kaum, sind allerdings bei Wohnzimmerplayern recht wichtig.

Audioformate

Dolby Digital

Das wichtigste Audioformat der DVD wird von der Firma Dolby entwickelt und ist unter vielen Namen bekannt. Am meisten verbreitet sind die Bezeichnungen *Dolby Digital* (der von Dolby verwendete Marketingname) und AC3 (Abkürzung für *Adaptive Transform Coder 3*, der Name des verwendeten Bitstream-Formats). Kaum jemand kennt dagegen den offiziellen Namen, den das ATSC-Gremium bei der Standardisierung vergeben hat: *ATSC A/52*.

Dolby Digital ist ein verlustbehaftetes Format, das mit konstanter Bitrate arbeitet. In der Regel werden für Stereo-Tonspuren 192 kbit/s verwendet, für Mehrkanalton 384 bzw. 448 kbit/s. AC3 unterstützt bis zu sechs Kanäle, wobei der sechste

als zusätzlicher Basskanal ausgelegt ist. Soll heißen, die normalen Bässe stecken in den fünf vollständigen Kanälen, der sechste enthält die zusätzlichen Basseffekte, die die Wände wackeln lassen. Da er eigentlich kein vollständiger Kanal ist, hat es sich eingebürgert, ihn extra anzugeben. Daher kommt die Schreibweise »5.1«, d. h. fünf vollständige Kanäle (vorne links und rechts, hinten links und rechts, vorne Mitte) und ein zusätzlicher Basskanal (LFE: Low Frequency Effects).

Digital Theater Systems

Beliebt ist auf der DVD auch DTS, das genauso wie AC3 ein verlustbehaftetes Format ist, allerdings mit deutlich höheren Bitraten arbeitet und noch mehr Kanäle unterstützt (bis zu 6.1).

MPEG

Die MPEG entwickelt, wie oben schon erwähnt, nicht nur Video-, sondern auch Audiostandards. In jeder Version ist zum Videoteil auch ein passendes Audioformat definiert. Hier sind die drei wichtigsten, alles verlustbehaftete Formate:

MPEG-1 Part 3 Layer 2

Besser bekannt als MP2. NTSC-DVDs dürfen Tonspuren in diesem Format nicht enthalten, PAL-Discs dagegen schon (was aber nur selten vorkommt). Zwingend wird MP2 auf der (S)VCD eingesetzt, besitzt also nach wie vor eine nicht geringe Bedeutung. Lediglich für unsere Zwecke im Encodingwissen können wir das Format vernachlässigen.

MPEG-1 Part 3 Layer 3

Dieses Format kennt wohl jeder als MP3 (Hat nichts mit MPEG-3 zu tun!). Unterstützt wird Mono- oder Stereoton. Zusätzlich existiert seit Herbst 2004 auch eine Spezifikation für Surround-Ton. Ob sich dieser Zusatz durchsetzen wird, ist allerdings fraglich, da für Multikanalton bessere und breiter unterstützte Formate existieren.

MP3 erreicht bei Musik transparente (also vom Original nicht mehr zu unterscheidende) Qualität in der Region um 150 kbit/s, einen sinnvoll konfigurierten und qualitativ hochwertigen Encoder vorausgesetzt. Das oft gehörte »128 Kilobit sind CD-Qualität« war lange Zeit ein Märchen. Wenn man sich die Ergebnisse von Sebastian Mares' *128-kbit-Hörtest* ansieht, darf man über die Aussage inzwischen nicht mehr ganz so laut lachen. Die meisten modernen Encoder (ob MP3 oder nicht) kommen bei den berühmten 128 kbit/s der Transparenz erstaunlich nahe (5,0 in den Hörtest-Diagrammen bezeichnet Transparenz). Aber Vorsicht: Hier gilt

ganz besonders die Voraussetzung eines guten Encoders und passender Konfiguration. Wenn wir von MP3 sprechen sollten wir ganz selbstverständlich auch vom Lame-Encoder sprechen. Es gibt keinen weiter entwickelten MP3-Codec und kaum einen Grund, Lame nicht zu verwenden.

Da wir uns mit Filmtonspuren beschäftigen, die wegen ihrer hohen Dynamik vergleichsweise einfach zu codieren sind, eignet sich eine 128-kbit-MP3 hervorragend für hochwertigen Stereosound.

MPEG-4 Part 3

Wird oft als offizielles MP3-Nachfolgeformat bezeichnet und dürfte vielen unter dem Namen Advanced Audio Coding (AAC) bekannt sein. Im Gegensatz zu MP3 unterstützt AAC auch Multikanalton. Die transparenten Bitraten für Filmton liegen ein wenig unter denen von MP3. Multikanalton lässt sich in sehr guter Qualität bis auf 160 kbit/s schrumpfen.

Auch für AAC existieren mehrere Encoder, wobei aus Lizenzierungsgründen nur die kommerziellen Varianten den für Filmtonspuren besonders interessanten HE-Modus unterstützen. Nero ist hier sicherlich der beliebteste Vertreter. Als hervorragender HE-Encoder macht in letzter Zeit Winamp von sich reden. Auch wenn er nur konstante Bitrate beherrscht, ziehe ich ihn Nero inzwischen vor.

Vorbis

Xiph.org entwickelt Vorbis als alternatives Audioformat, das komplett frei von patentierten Technologien sein soll und so eventuelle rechtliche Probleme mit den Patentinhabern vermeidet.

Als allein stehende Audiodatei ist Vorbis grundsätzlich in den Ogg-Container verpackt (daher der Doppelname Ogg Vorbis), als Sound eines Videos liegt die Vorbis-Tonspur ohne Ogg-Hülle im Container des gesamten Films.

Vorbis unterstützt natürlich Mono- und Stereoton. Auch ein Multikanal-Modus existiert, der allerdings noch wenig ausgereift ist und für gute Qualität AC3-ähnliche Bitraten benötigt. Davon abgesehen bietet Vorbis gerade bei geringen Bitraten um 80 kbit/s und weniger deutlich bessere Qualität als MP3. Bei den transparenten Musik-Bitraten ist der Unterschied dagegen nicht so spektakulär.

PCM

Dieses Format kennt der Windowsbenutzer als Wave. Es handelt sich im Gegensatz zu allen anderen um ein nicht komprimiertes Format, das entsprechend Platz belegt. PCM erlaubt bis zu 8 Kanäle.

A.2.2 Das Format der DVD

Die Eckdaten der DVD

Video	MPEG-2 720 × 576 Pixel PAL 720 × 480 Pixel NTSC
Audio	AC3, DTS, MPEG oder PCM maximal 8 Audiospuren
Untertitel	gespeichert als Pixelbilder Forced-Subs-Funktion maximal 32 Untertitelspuren
Datenrate	insgesamt maximal 10,08 Mbit/s

Tabelle 1: Technische Daten der Video-DVD.

Bildformate

Die Welt ist sich uneins, das gilt auch für digitales Video. Europa setzt auf den PAL-Standard, Amerika auf NTSC. Entsprechend gibt es verschiedene DVDs. Wer einen Film aus Amerika bestellt und auf einem europäischen Player/Fernseher abspielen will, hat keine besonders große Freude daran, es sei denn, seine Geräte verstehen außer dem üblichen PAL auch ausdrücklich das NTSC-Format.

Wer nachrechnet wird feststellen, dass der heimische Fernseher entweder ein Seitenverhältnis (AR: Aspect Ratio) von 4:3 (entspricht 1,33:1) oder 16:9 (1,78:1) hat. Das Bild einer PAL-DVD dagegen besitzt ein AR von $^{720}/_{576} = 1{,}25$. Das passt doch nicht zusammen? Doch, tut es. Die DVD enthält das Bild nämlich in verzerrter Form. Erst beim Abspielen wird dafür gesorgt, dass es mit dem richtigen Seitenverhältnis am Monitor ankommt.

Die Tiefen der AR-Thematik sind nicht nur für die DVD wichtig, sondern unter Umständen auch für das encodierte Video. Deswegen beschäftigen wir uns damit ausführlich im Abschnitt A.3.2 Anamorphes Video ab Seite 40.

Video und Audio

Das Bild einer DVD ist normalerweise im MPEG-2-Format komprimiert, obwohl auch MPEG-1 zulässig wäre. Je nach Codec und Bildformat sind folgende Auflösungen möglich:

	PAL	NTSC
MPEG-1	352 × 288	352 × 240
MPEG-2	720 × 576 (Standard)	720 × 480 (Standard)
	704 × 576	704 × 480
	352 × 576	352 × 480
	352 × 288	352 × 240

Tabelle 2: Erlaubte Auflösungen der Video-DVD.

Auch die möglichen Audiospuren unterscheiden sich zwischen NTSC und PAL. Maximal dürfen es acht Stück sein, wobei für PAL mindestens eine Spur im PCM-, MPEG-, oder AC3-Format Pflicht ist. Für NTSC-Discs dagegen ist MPEG als einzige Audiospur verboten.

PCM darf bis zu 8 Kanäle enthalten, AC3 bis zu 6 Kanäle. MPEG-Ton gibt es in zwei Varianten: Als MPEG-1 Audio Layer 2 (MP2) darf maximal Stereo und 384 kbit/s verwendet werden. Als MPEG-2 sind bis zu 7.1 Kanäle bei maximal 912 kbit/s erlaubt. Alternative Formate wie DTS dürfen auf jeder DVD vorhanden sein, allerdings nicht als einzige Tonspur. Eine Spur in einem der obigen Standardformate ist zwingend.

Die typische DVD

Eine typische deutschsprachige DVD dürfte aktuell in etwa so ausgestattet sein:

- PAL, 720 × 576 Pixel, MPEG-2-Video.
- Audiospuren in AC3 5.1, Deutsch und Originalsprache (also oft Englisch).
- Eventuell eine deutsche 6.1-DTS-Audiospur.
- Manchmal zusätzliche Audiospuren mit Director's Comments o. ä.
- Eine Reihe von Untertiteln in verschiedenen Sprachen.

Es besteht also kein Grund, angesichts der Vielfalt der möglichen Formate in Panik auszubrechen, denn die Wirklichkeit sieht recht übersichtlich aus. Wer sich tiefer einlesen möchte, findet auf *dvd-tipps-tricks.de* massig Informationen.

A.2.3 Die Containerformate

Nachdem wir im Grundlagen-Abschnitt schon erklärt haben, was ein Container eigentlich ist, sehen wir jetzt die wichtigsten etwas näher an. Eine Übersicht der Funktionalität findet sich auf Seite 30.

Audio/Video Interleave (AVI)

AVI stammt aus der Zeit, in der ein kurzer Videoclip im Briefmarkenformat das höchste der Gefühle darstellte. Komplette Filme im Vollbild, codiert mit modernen Encodern, hatte bei der Entwicklung des Containers niemand im Sinn. Für diese Voraussetzungen hält sich AVI zwar erstaunlich gut, allerdings zeigt er spätestens seit dem Erfolg der echten MPEG-4-Codecs (was in die Zeit von DivX 4 zurückreicht) Schwächen. Denn um MPEG-4-Video zuverlässig in den AVI-Container zu packen, musste der Stream auf eine Art verändert werden, die dem MPEG-4-Standard entgegenläuft. Außerdem hat AVI mit einigen Einschränkungen und Problemen zu kämpfen, die oft nicht am Containerformat selbst, sondern an alten und fehlerhaften Tools und Filtern liegen. MP3 mit variabler Bitrate funktionierte lange Zeit nicht, genauso wie AAC. Und nach wie vor ist es nicht möglich, Vorbis-Audio zu verwenden. Metadaten, wie Informationen zu den Streams oder eine Kapitelliste, werden nicht unterstützt, genauso wenig wie Menüs.

Trotzdem hat sich AVI bis heute als verbreitetster Container für MPEG-4-Video behauptet. Dank Matroska und vor allem MP4 zeichnet sich inzwischen aber eine dringend nötige Änderung ab.

Matroska (MKV)

Das ambitionierte freie Containerprojekt von *matroska.org* ist mit dem Anspruch angetreten, AVI zu ersetzen. Das Potenzial dazu hat der Container allemal: Er unterstützt eine äußerst breite Palette von Video-, Audio und Untertitelformaten, enthält ein umfangreiches Metadaten-System (einschließlich Kapiteln) und bietet Spezialitäten wie Attachments (ähnlich E-Mail-Anhängen) oder die Fähigkeit, mehrere Dateien aneinander zu linken. Einen passenden Filter vorausgesetzt, können die dann als ein langer Film abgespielt werden. Auch DVD-ähnliche Menüunterstützung ist vorgesehen, allerdings noch nicht funktionsfähig.

Was fehlt, ist die breite Unterstützung außerhalb des PCs. Aufgrund der nahe-

zu endlosen Fähigkeiten stellt Matroska für einen Standalone-Hersteller keine attraktive Lösung dar. Ein Container mit einem enger definierten Feature-Set ist vorteilhafter, weil er einfacher implementiert werden kann und weniger potenzielle Problemquellen aufweist. Am Computer bietet Matroska allerdings klar die umfangreichsten Freiheiten.

MP4

Der MPEG-4-Standard definiert in Part 14 sein eigenes Containerformat mit dem Namen MP4, was gleichzeitig als Dateiendung verwendet wird. Er unterstützt Menüs und eine recht eingeschränkte Auswahl an Video-, Audio- und Untertitelformaten. Gerade deswegen und als Teil eines Industriestandards ist ihm mittelfristig die breite Unterstützung der Hardwarehersteller sicher.

Am Computer kann der eingeschränkte Funktionsumfang eher störend wirken. Aber um es ganz deutlich zu sagen: MP4 ist für die Welt der Hardwareplayer gemacht und entsprechend sollte man ihn behandeln. Standardkonforme Streams sind Pflicht. Die Einschränkungen der erlaubten Streamformate haben ihren Sinn und sollten respektiert werden. Wozu auch MP4 aufbohren? Hardwareplayer werden sich eher an den Standard halten und mit solchen Zusatzfeatures nichts anzufangen wissen. Und am Computer spricht nichts dagegen, Matroska zu verwenden, wenn man auf maximale Freiheit Wert legt.

Man kann es nicht oft genug sagen, deshalb an dieser Stelle noch einmal: MPEG-4 und MP4 sind nicht das selbe! MPEG-4 ist der Name des kompletten Standards, MP4 ist der Name des in diesem Standard definierten Containerformats.

OggMedia (OGM)

Da Vorbis in AVI unmöglich ist, warum sollte nicht der umgekehrte Weg funktionieren: also eine AVI in den Ogg-Container zu packen? Das Ergebnis heißt OggMedia und unterstützt natürlich Vorbis, genauso wie MP3, AC3, DTS und AAC. Untertitel und Kapitel sind auch kein Problem. Allerdings ist OGM ein aus der Not geborenes Format, um ein spezielles Problem – nämlich Vorbis als Filmtonspur – zu lösen. Es existiert keine Dokumentation und die Entwicklung steht seit längerer Zeit still. OggMedia kann nichts, was Matroska oder MP4 nicht genauso kön-

nen. Wie alle Notlösungen, so sollte man auch OGM in Ruhe sterben lassen, wenn eine bessere Alternative verfügbar ist.

Die Fähigkeiten der Container im Vergleich

Tabelle 3 auf Seite 30 bietet einen Überblick über die Fähigkeiten der Container. Hier folgen die Erklärungen zur Tabelle.

a *VfW MPEG-4* ist der für AVI aufbereitete, aber nicht standardkonforme, Bitstream. Einen solchen Stream in einem modernen Container zu verwenden, der mit standardkonformem *native MPEG-4* problemlos arbeiten kann, ist sinnlos. Deswegen sollte außerhalb von AVI wo immer möglich dem nativen MPEG-4 der Vorzug gegeben werden.
b Damit sind Angaben wie Sprache oder Trackname gemeint.

1 Von der technischen Seite betrachtet besteht kein Unterschied zwischen AAC in AVI und VBR MP3 in AVI. Beides ist laut AVI-Spezifikation erlaubt. Der Unterschied besteht darin, dass VBR MP3 sowohl beim Muxing als auch bei der Wiedergabe weitläufig unterstützt wird. AAC kann dagegen nur von AVI-Mux GUI gemuxt werden. Die Wiedergabe am PC stellt mit einem sinnvollen Filter allerdings kein Problem dar.
2 Der MPEG-4-Standard sieht keine Vobsub-Unterstützung vor. Allerdings existieren in MP4 so genannte private Streams, in denen prinzipiell alles gespeichert sein darf. Nero nutzt dies aus, um Vosubs in MP4 zu packen. Der Standard erlaubt das einwandfrei. Allerdings kann man sich nicht darauf verlassen, dass ein privater Stream überall wiedergegeben wird.
3 Im Standard vorhanden, aber noch ohne breite Unterstützung.
4 Die so genannten DivX- oder MPEG4-fähigen Hardwareplayer verwenden in der Regel AVI als Container.
5 Noch wenig verbreitet, ist aber v. a. dank Nero im Kommen.

	AVI	Matroska	MP4	OggMedia
	Video[a]			
Native MPEG-4	Nein	Ja	Ja	Nein
VfW MPEG-4	Ja	Ja	Nein	Ja
	Audio			
MP3	Ja [1]	Ja	Ja	Ja
Vorbis	Nein	Ja	Nein	Ja
AAC	teils [1]	Ja	Ja	Ja
AC3, DTS	Ja	Ja	Nein	Ja
	Untertitel			
SubRip, SSA	Ja	Ja	Ja	Ja
Vobsub	Nein	Ja	teils [2]	Nein
	Metadaten			
Stream-Info[b]	Nein	Ja	Ja	teils
Kapitel	Nein	Ja	Ja	Ja
Menüs	Nein	teils [3]	teils [3]	Nein
	Player			
Computer	Ja	Ja	Ja	Ja
Hardware	teils [4]	Nein	teils [5]	Nein

Tabelle 3: Funktionsumfang von Multimedia-Containern.

Ja	Vollständige, problemlose Unterstützung.
teils	Teilweise oder nicht standardkonforme Unterstützung.
Nein	Keine Unterstützung.

Empfehlung

Für den Player im Wohnzimmer bleibt oft nur AVI, welches die DivX-fähigen bzw. »MPEG4-fähigen« Hardwareplayer verwenden. Man sollte allerdings mit verschiedenen Einschränkungen rechnen. Zum Glück steht MP4 als zukunftssicherer Ersatz in den Startlöchern. Wer heute einen Player fürs Wohnzimmer kauft, sollte sich von AVI verabschieden und auf MP4-Unterstützung (meist in Form einer Nero-Zertifizierung) Wert legen.

Am Computer brauchen wir uns mit solchen Überlegungen nicht herumschlagen. Die Wahl hängt rein von den benötigten Features und dem persönlichen Geschmack ab. – Ach, was sag ich! Matroska wird sie alle in die Tasche stecken! :-) Soll heißen, ich habe meine Wahl getroffen.

A.2.4 Die Videocodecs

Mit sämtlichen Videocodecs werden wir uns im Encodingwissen nicht beschäftigen. Viele sind zu unwichtig und für viele fehlt mir die Erfahrung. Deshalb konzentrieren wir uns im Praxisteil auf »die großen drei«: Xvid, DivX und x264.

DivX ;-) 3.11

So heißt der gehackte MS MPEG4v3-Codec. Der Hack bestand darin, den Codec von seiner Bindung an den nachteiligen ASF-Container zu befreien. Im Gespann mit Nandub (was dann unter der Bezeichnung SBC läuft) hat DivX 3 die Massenbewegung um digitales Video ausgelöst. Das SBC-Encoding ist gerade für Anfänger unübersehbar kompliziert und bietet schon lange keine optimale Qualität mehr. Wegen seines Charakters als Hack einer Microsoft-Software steht der Codec zusätzlich auf wackligem rechtlichen Terrain.

Aus historischen Gründen ist es angebracht, DivX ;-) zu erwähnen. Für den praktischen Einsatz ist er heutzutage unbrauchbar. Deswegen werden wir uns im Encodingwissen nicht weiter damit beschäftigen.

DivX

Der offizielle und legale Nachfolger von DivX ;-) hat seine Wurzeln im DivX 3.11, ist allerdings von Grund auf neu programmiert und wird kommerziell von *DivX Inc.* vertrieben (aktuell in der 6er-Generation). Seine Stärke liegt in der einfachen Konfiguration, doch auch in Sachen Geschwindigkeit und Qualität hat er sich inzwischen auf das Level seines ewigen Konkurrenten Xvid hochgearbeitet.

Mit zertifizierten Profilen und Partnerschaften zu Hardwareherstellern hat DivX Inc. klar den Wohnzimmerplayer und den weniger erfahrenen User als Zielgruppe im Auge.

Xvid

In Zeiten von DivX 4 hat sich Xvid als eigenständiges Projekt abgespalten. Der Codec ist nichtkommerziell und sein Quellcode auf *Xvid.org* frei zugänglich. Dank eines Teams von enthusiastischen und verdammt fähigen Programmierern hatte er sich längere Zeit deutlich von seinem Kollegen DivX abgesetzt, was heute so klar nicht mehr gilt. Xvids Vorteil liegt in der Fülle der Optionen, die eine äußerst detailgenaue Anpassung der Konfiguration ermöglichen. Allerdings kann sich gerade der Anfänger leicht zwischen den vielen Einstellungen verirren.

Früher hieß der Codec *XviD* (mit großem D). Die neue Schreibweise *Xvid* gilt seit dem Redesign der Xvid.org-Website Anfang November 2006. Zur gleichen Zeit wurde auch das Bugfix-Release Xvid 1.1.2 veröffentlicht. Wir können die Namensänderung also genauso an der Versionsnummer festmachen.

---

x264

X264 verwendet genauso wie Xvid und DivX den MPEG-4-Standard, allerdings einen anderen Teil: Part 10 (AVC). AVC ist zur Zeit das Beste, was MPEG-4 in Sachen Videocodierung zu bieten hat. Wie viel Potenzial darin steckt, sieht man am jüngsten *Doom9-Codecvergleichstest*. Neros AVC-Encoder konnte sich an die Spitze setzen, noch vor den weiter entwickelten und besser optimierten ASP-Codecs.

Wer keine Lust hat, für Nero-AVC Geld auszugeben, kann zu x264 greifen, der ähnlich wie Xvid als Open-Source-Projekt entwickelt wird und im Doom9-Test ganz vorne mitspielt. Da die x264-Entwicklung zeitweise rasant voranschreitet, sollte man sich regelmäßig um eine aktuelle Version kümmern.

Empfehlung

Wer sich wenig Gedanken machen will – zwei oder drei Optionen einstellen und loslegen – der ist mit DivX gut beraten, sollte allerdings etwas Geduld mitbringen, da DivX in Sachen Geschwindigkeit nicht besonders glänzt. Wer maximale Bildqualität herausholen will und bereit ist, sich dafür auch mit den Hintergründen der ganzen Einstellungen zu beschäftigten, dürfte an Xvid mehr Freude haben. Als Bonus kommt die spürbar höhere Geschwindigkeit des freien Codecs dazu.

Wer den Hype liebt und vorne dabei sein will, für den führt kein Weg an AVC (wahrscheinlich in Form von x264) vorbei. Gerade bei niedrigen Bitraten lässt sich der Vorteil den ASP-Codecs gegenüber auch schwer verleugnen. Je höher die Bitrate, desto geringer wird AVCs Vorsprung und die Nachteile erhalten mehr Gewicht. Die neue Technologie braucht zum Encoding spürbar länger und verlangt beim Abspielen genauso mehr Prozessorleistung.

Einen weiteren Punkt sollte man beachten. Der Trend zeigt fort von AVI und dem damit verbundenen VfW-Encoding-Framework. Leider bietet DivX ausschließlich einen VfW-Codec an, der entsprechend verbogene Videostreams erzeugt (vgl. Kapitel A.2.3). Und obwohl sich ein solcher Stream wieder gerade richten lässt, ist DivX doch deutlich auf AVI/VfW fixiert. Xvid und x264 dagegen bieten sowohl einen VfW-Encoder als auch eine Version für die Kommandozeile, die native Streams für moderne Container erzeugt.

A.2.5 Die Audiocodecs

Die Auswahl des Audiocodecs ist im Lauf der Zeit deutlich komplizierter geworden. Die einfache alte Formel »MP3 oder bei genügend Platz AC3« gilt längst nicht mehr. Werfen wir deshalb einen Blick auf die verschiedenen Möglichkeiten.

MP3

Der Klassiker. Mit MP3 brauchen wir uns um Abspielprobleme keine Sorgen zu machen: läuft überall. Dank Lame steht ein freier und hervorragender Encoder bereit, der in der 128-kbit-Region exzellente Ergebnisse erzielt – mit Tonspuren noch mehr als mit Musik. Niedrigere Bitraten sollten wir möglichst vermeiden, da

MP3 dafür schlicht und einfach nicht gemacht ist. Davon abgesehen bleibt der größte Nachteil die Beschränkung auf Stereo. Den 6-Kanal-Ton der DVD können wir mit MP3 nicht beibehalten.

Vorbis

Für Stereoton ist Vorbis *der* Konkurrent zu MP3, besonders bei geringer Datenrate. 80 kbit/s sind ohne Bedenken machbar, und die derzeitige Entwicklung arbeitet stark daran, Vorbis für noch niedrigere Bitraten zu optimieren.

Düsterer sind die Aussichten für Mehrkanalton. Zwar besitzt Vorbis einen Multichannel-Modus, allerdings ist der weder intensiv getestet noch wirklich ausgereift. Da die aktiven Entwickler für Verbesserungen hauptsächlich Musik und kaum Tonspuren im Hinterkopf haben, bleibt die Mehrkanalfähigkeit unbeachtet. Um qualitativ auf der sicheren Seite zu stehen, müssen wir mit AC3-ähnlichen Bitraten kalkulieren, und damit bleibt Vorbis praktisch auf Stereo beschränkt.

AAC

Advanced Audio Coding ist das »MP3 der Zukunft« und offizielles Audioformat des MPEG-4-Standards. V. a. im HE-Modus ist die Komprimierungsleistung beachtlich: 70 kbit/s für Stereo sind kein Problem und 6-Kanal-Ton gibt sich mit 160 kbit/s zufrieden. Nachteil: Gerade der HE-Modus braucht beim Abspielen einiges an Rechenleistung. Für ältere Rechner unter 500 MHz schrumpfen AACs Vorteile damit kräftig zusammen.

Der bekannteste Encoder ist sicherlich Nero, doch in jüngster Zeit macht Winamp mit einem Encoder von sich reden, der trotz Beschränkung auf CBR beste Ergebnisse liefert. Beide Encoder sind kommerziell und in Nero bzw. Winamp eingebunden. Zum Glück haben wir trotzdem die Möglichkeit, sie auch bequem und universell mit BeSweet zu verwenden.

Im Lager der freien Software sieht es in Sachen AAC-Encoder düster aus. Sehr viel mehr als FAAC existiert nicht, und der unterstützt nur LC, nicht das interessante HE, und kann qualitativ mit keinem der kommerziellen Encoder mithalten.

AC3

Im fertigen Encoding AC3 einzusetzen ist eine einfache Lösung, weil wir die Tonspur nur unverändert von der DVD übernehmen müssen. Es entfällt sowohl die Arbeit als auch der Qualitätsverlust des Transcodings, und natürlich bleibt der Mehrkanal-Ton erhalten. Nachteil: 5.1-AC3 ist mit 384 kbit/s oder 448 kbit/s recht groß, benötigt also eine entsprechend hohe Zielgröße, damit die Videoqualität nicht leidet. Dank DVD-Brenner ist das heute kein großes Problem mehr, denn bei ½ DVD-5 als Zielgröße ist nahezu immer Platz für mindestens eine AC3-Spur. Für das urklassische hochkomprimierte 1-CD-Encoding können wir AC3 dagegen vergessen. Allerdings muss AC3 nicht immer 6 Kanäle enthalten, auch 192-kbit-Stereo (z. B. die deutschen Tonspuren der Indiana-Jones-DVDs) oder sogar Mono (z. B. Audiokommentare) sind möglich.

Selber AC3 in guter Qualität zu encodieren ist ohne teuren kommerziellen Encoder schwer möglich. Zwar enthält BeSweet einen Encoder namens ac3enc, dessen Qualität allerdings mehr als zweifelhaft ist.

DTS

DTS (Digital Theater Systems) ist inzwischen auf DVDs recht weit verbreitet. Wie AC3 arbeitet DTS mit konstanter Bitrate, verwendet aber höhere Bitraten (768 oder 1536 kbit/s) und unterstützt einen Kanal mehr (6.1). Ohne für einen professionellen Encoder tief in den Geldbeutel zu greifen, können wir DTS nicht selbst erzeugen. Also bleibt nur, die Originalspur in den fertigen Film zu übernehmen. Die Container, die AC3 unterstützen, kommen auch mit DTS zurecht. Persönlich finde ich es allerdings blödsinnig, für eine Audiospur so extrem viel Platz zu opfern.

Manchmal taucht die Empfehlung auf, als Quelle die DTS-Spur anstelle der AC3 zu verwenden, da DTS doch eine höhere Bitrate und damit höhere Qualität hätte. So wären auch in der Zieldatei weniger Artefakte vorhanden. Das ist zwar theoretisch richtig, nur sprechen praktische Erwägungen dafür, doch AC3 zu verwenden. Der Qualitätsunterschied dürfte marginal und auf alle Fälle unhörbar ausfallen. Dazu kommt zum einen, dass sich DTS nicht so bequem wie AC3 per BeSweet umwandeln lässt. Zum anderen können wir gerade das AC3-Decoding dank Azid sehr detailliert konfigurieren, was mit DTS nicht möglich ist. Aus diesen Gründen werden wir uns um DTS als Transcodingquelle auch nicht weiter kümmern.

Empfehlung

Der passende Audiocodec hängt von einigen Faktoren ab, von denen einer sicher immer der persönliche Geschmack ist. Wichtiger ist aber erst einmal der gewünschte Film-Container (vgl. Kapitel A.2.3). Entscheiden wir uns für AVI, müssen wir uns gleichzeitig von Vorbis verabschieden, da Vorbis nicht in AVI gemuxt werden kann. Einige Hardwareplayer haben Probleme mit VBR MP3, und AAC in AVI unterstützt soweit ich weiß keiner. Auch MP4 hat einige Einschränkungen: Vorbis und AC3 sind nicht möglich. Am unproblematischsten ist Matroska, da dieser Container ganz bequem alle angesprochenen Audioformate unterstützt.

Die nächste Frage gilt der gewünschten Anzahl Kanäle. Wollen wir, falls vorhanden, den originalen 6-Kanal-Ton beibehalten, bleibt uns praktisch nur die ursprüngliche AC3 oder AAC. Und ich gebe die Hoffnung nicht auf, dass sich irgendwann doch einmal jemand um den Multichannel-Modus von Vorbis kümmert.

Als letztes sollten wir auch Qualität und Kompatibilität der einzelnen Codecs betrachten. Wer seine Encodings auf dem Standalone-Player abspielen will, wird sich wohl oder übel auf MP3 (und evtl. AC3) beschränken müssen. Davon abgesehen bietet sich MP3 nicht mehr an, denn Vorbis und AAC erreichen die gleiche Qualität bei geringeren Bitraten. Als offizielles MPEG-4-Audioformat dürfte AAC langfristig die besten Chancen haben sich durchzusetzen. Nachteil im Moment ist das Fehlen eines freien Encoders, der alle Features unterstützt. Zusätzlich stellt sich im Zeitalter der DVD±R die Frage: Warum nicht gleich AC3 behalten, wenn auf der Scheibe eh massig Platz ist? Das gilt natürlich weniger für das klassische Encoding auf ein bis zwei CDs.

Eine klare Empfehlung ist bei der heutigen Vielfalt an Formaten nicht mehr möglich. Meine persönlichen Favoriten sind HE-AAC für 5.1-Ton und Vorbis für Stereo. Seit ich den DVD-Brenner habe, schafft es aber dank genügend Platz auch immer häufiger die originale AC3 ins Encoding.

A.2.6 Die Untertitelformate

Konkrete Untertitelformate gibt es viele, wobei alle in zwei Kategorien fallen. Entweder handelt es sich beim eigentlichen Untertiteltext tatsächlich um Text, der dann flexibel in Größe und Schriftart angezeigt wird; oder der Text ist als Bild gespeichert, das übers Video geblendet wird.

Beim DVD-Backup haben wir es als Quelle immer mit Bild-Untertiteln zu tun, die *Vobsubs* genannt werden. Im fertigen Film können sowohl Vobsubs als auch echte Textuntertitel vorhanden sein.

Untertitel auf der DVD

Die einfachste Art Untertitel sind die, die im technischen Sinn gar keine Untertitel darstellen. Die sind auch auf der DVD schon fest ins Video »eingebrannt«, d. h. sie liegen nicht getrennt von der Videospur vor. Solche Untertitel müssen wir so wie sie sind hinnehmen. Die Möglichkeiten, an denen etwas zu ändern, sind so gut wie nicht vorhanden.

Interessant für uns sind echte Untertitel, die in einer separaten Untertitel-Spur auf der DVD liegen. Einer Bearbeitung steht dadurch nichts im Weg. Echte Untertitel teilen sich in drei Arten ein.

- **Dialog-Untertitel** bilden die kompletten Dialoge des Films ab und enthalten auch alle zusätzlichen Untertitelungen (z. B. die Zeitangaben in *Spy Game*). Das ist die typische All-Inclusive-Untertitelspur.
- **Szene-Untertitel,** liefern nur die Zusatzinformationen außerhalb der Dialoge und Übersetzungen fremder Sprachen. Wer wird da an *Herr der Ringe* denken … Auch Szene-Untertitel liegen in einer separaten Spur.
- **Forced Subs** verpacken die Szene-Untertitel in der Spur der Dialog-Untertitel. Soll heißen, wir haben eine Dialog-Untertitel-Spur. Für einige der Untertitel ist ein Forced-Flag gesetzt, so dass die (und nur die) auch dann angezeigt werden, wenn beim Anschauen die Untertitel abgeschaltet sind. Jeder, der kein Russisch kann, wäre sonst bei *Der Anschlag* reichlich aufgeschmissen.

Für die Verarbeitung müssen wir erst einmal wissen oder herausfinden, auf welche Weise die Untertitel auf der DVD gespeichert sind. Am häufigsten sind All-Inclusive-Untertitelspuren mit einigen Forced Subs. Verlassen können wir uns darauf aber nicht. Leider lässt sich vor dem Rippen oft nicht erkennen, welche Untertitelspur exakt was enthält, so dass uns ein wenig Detektivarbeit nicht erspart bleibt. Ich habe mir deswegen angewöhnt, erst einmal sämtliche Untertitel auf die Platte zu ziehen und hinterher auszusortieren.

Untertitel im Encoding

Nun stehen wir vor der Wahl, welche Untertitel im endgültigen Film vorhanden sein sollen. Eines ist klar: Szene-Untertitel bzw. Forced Subs sollten wir nie weglassen, weil die für das Verständnis des Films wichtig sein können. Dialog-Untertitel dagegen sind hauptsächlich für Dinge wie *Originalton mit Untertiteln* interessant. Im Normalfall können wir auf die auch ganz gut verzichten.

Die zweite Entscheidung betrifft das Format der Untertitel im fertigen Film. Auf der DVD sind sie als Bilder gespeichert, die über den Film geblendet werden. Für unseren fertigen Film gibt es drei Möglichkeit, Untertitel zu übernehmen.

- Bilduntertitel der DVD (Vobsubs) fest ins Video einbrennen (sind dann nicht mehr ausblendbar).
- Vobsubs als dynamische (ausblendbare) Untertitel übernehmen.
- Bilduntertitel in Text umwandeln und in dieser dynamischen Form übernehmen.

Wann ist denn was sinnvoll? Das kommt sehr darauf an, wie unser fertiger Film aussehen soll. Wenn wir uns auf eine Tonspur beschränken, können wir Szene/Forced-Untertitel (wenn der Film überhaupt solche hat) der Einfachheit halber fest ins Bild einbrennen. Bei mehreren Tonspuren in verschiedenen Sprachen bietet sich auch hier das dynamische Einbinden an, um später die Untertitel passend zur jeweiligen Sprache vorrätig zu haben. Für komplette Dialog-Spuren ist die dynamische Methode dagegen immer erste Wahl, um den Film auch einmal ohne Untertitel anschauen zu können.

Ich mag eingebrannte Szene-Untertitel auch bei einsprachigen Filmen nicht, da die Schrift meistens doch spürbar unter dem Hochzoomen aufs Vollbild leidet. Ein Textuntertitel, der mit einer TrueType-Schrift in passender Größe dargestellt wird, sieht deutlich sauberer aus. Zweiter Vorteil: Dynamische Untertitel müssen nicht direkt im Bild erscheinen, sondern können in den schwarzen Balken über/unter dem Video eingeblendet werden. Der gleichmäßige Hintergrund verbessert die Lesbarkeit noch einmal.

A.3 Das Videobild

In diesem Kapitel beschäftigen wir uns mit dem Video. Genauer damit, wie wir das Quellvideo von der DVD aufbereiten und verändern können und müssen, um es schließlich an den Encoder zu verfüttern. Die Encoder selbst sind Thema im nächsten Kapitel.

A.3.1 Cropping

Mit *Cropping* bezeichnet man das Wegschneiden der schwarzen Balken um das Bild. Am linken und rechten Rand sind das – wenn überhaupt – nur wenige Pixel. Oben und unten dagegen haben wir es oft mit dicken Balken zu tun. Da die DVD nur 4:3 und 16:9 kennt, müssen alle schmäleren Formate den restlichen vertikalen Platz mit schwarzen Balken auffüllen. Warum entfernen wir die Balken eigentlich? Aus zwei Gründen:

- Sie tragen keine Information.
- Sie verbrauchen unnötig Bitrate.

Der zweite Punkt verdient eine genauere Erklärung. Die Balken an sich sind nicht das Problem. Da es sich um große, einfarbige Flächen handelt, die sich im Lauf des Films nie verändern, lassen sie sich hervorragend komprimieren und verbrauchen nur wenig Bitrate.

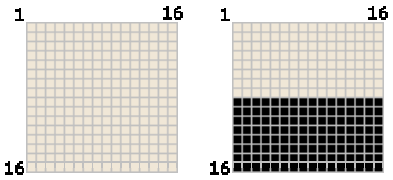

Abbildung 7:
Zwei Makroblocks im Vergleich.

Der große Nachteil der Balken ist die extrem harte Kante zum eigentlichen Bild. Nahezu alle Videocodecs sind darauf optimiert, Bildmaterial aus der realen Welt zu komprimieren. In einer solchen Umgebung existieren kaum harte Kanten. Ein kleiner Übergangsbereich bleibt immer übrig. Entsprechend haben die Codecs mit unnatürlich harten Kanten Probleme und benötigen extrem viele Bits, um sie exakt zu encodieren. Der technische Grund liegt in den Makroblocks. Sehen wir uns die zwei Blöcke in Abbildung 7 an (die grauen Linien verdeutlichen die einzelnen Pixel).

Der linke Block besteht durchgängig aus ein und derselben Farbe. Er ist

extrem einfach zu encodieren, denn der Codec braucht lediglich den einen Farbwert speichern.

Anders der rechte Block. Der liegt am unteren Rand des eigentlichen Bildes und enthält die Grenze zum schwarzen Balken. Um ihn exakt abzubilden, muss der Codec deutlich mehr speichern: beide Farbwerte und Informationen darüber, wie die Grenze verläuft.

Vermeiden lässt sich das Problem nur, wenn die Balkengrenze genau auf eine Makroblockgrenze fällt, da Blocks sowieso unabhängig voneinander betrachtet werden. Ob sie die gleiche oder unterschiedliche Farben haben, spielt keine Rolle. Da so gut wie nie sämtliche Balkengrenzen genau auf Blockgrenzen fallen, gehört das Cropping zu den elementaren Aufgaben beim Bearbeiten des Bildes.

A.3.2 Anamorphes Video

Manch einer nimmt schreiend die Beine in die Hand und rennt so schnell er kann, wenn der Begriff »anamorph« fällt. Verständlich. Das Thema des richtigen Seitenverhältnisses ist genauso komplex wie kontrovers. Die nächsten beiden Kapitel versuchen, etwas Licht ins Dunkel zu bringen und die teils äußerst akademische Diskussion auf ein pragmatisches Niveau zurückzuholen. Schließlich wollen wir Filme encodieren und uns nicht im Streit über die zehnte Kommastelle verlieren. ;-) Um deshalb das ganze Thema in einem Satz zusammenzufassen: Die ganze Verwirrung liegt daran, dass das Bild auf der DVD verzerrt (horizontal gestaucht) gespeichert ist. Das müssen wir beim Encoding berücksichtigen.

A.3.2.1 Anamorphes Quellvideo

In diesem Kapitel beschäftigen wir uns mit ein bisschen Theorie und hauptsächlich mit dem Bild der DVD, das uns als Quelle dient. Zu wissen, wie genau die DVD Video speichert, ist zentral, um nicht zum Schluss mit Eierköpfen im Encoding dazustehen.

Begriffsdefinitionen

So staubig und lästig es sein mag: Ein zentraler Punkt, um nicht rasant in größte Verwirrung zu stürzen, ist, sich erst einmal über eine Reihe von Begriffen klar zu werden.

Seitenverhältnis

Im englischen *Aspect Ratio*, woher die typische Abkürzung *AR* stammt. Beschreibt die Form eines Videobilds, die entweder in der Notation *x:y* oder *x,y:1* angegeben wird (z. B. 16:9 oder 1,78:1). Oft sieht man auch die reine Kommazahl ohne den Zusatz *:1*. Es gibt mehrere Arten von Seitenverhältnissen.

Display Aspect Ratio (DAR)

Das DAR beschreibt das Seitenverhältnis des kompletten Bildes, stellt also das Verhältnis von Breite zu Höhe dar. Ein Bild mit der Auflösung 640 × 480 Pixel hat (im nicht-anamorphen Fall) ein DAR von 640:480, was gekürzt 4:3 entspricht oder als Kommazahl 1,33:1.

Die Xvid-VfW-Oberfläche hat das DAR bis vor kurzem »Picture Aspect Ratio« genannt (ist inzwischen korrigiert). Bitte davon nicht verwirren lassen.

Pixel Aspect Ratio (PAR)

Das PAR beschreibt das Seitenverhältnis eines einzelnen Pixels. Im Gegensatz dazu, was einem der gesunde Menschenverstand auf Anhieb einredet, ist dieser Wert *nicht* identisch mit dem DAR. Unser 640 × 480-Beispiel von oben hat ein PAR von 1:1 – die Pixel sind exakt quadratisch.

Vom x264-Encoder wird das PAR als »Sample Aspect Ratio« (SAR) bezeichnet. Bitte auch hiervon nicht verwirren lassen.

Anamorphes Video

Bezeichnet ein Videobild mit nicht-quadratischen Pixeln, also einem PAR ungleich 1:1. In der professionellen, v. a. analogen, Videotechnik ist exakt festgelegt, welche Pixelform für PAL und NTSC anamorph heißt. Alle anderen Verzerrungen werden nicht als anamorph bezeichnet.

Da dieser enge Rahmen für die komplett digitale MPEG-4-Welt kaum von Bedeutung ist, verwenden wir im Encodingwissen eine weiter gefasst Definition **für das encodierte Video**. Anamorph bedeutet hier tatsächlich lediglich *PAR ist nicht 1;* weniger technisch ausgedrückt: *Das Video ist nicht im korrekten Wiedergabe-Seitenverhältnis, also verzerrt, gespeichert.* Wie die Verzerrung genau aussieht, bleibt offen und deshalb frei wählbar.

Bei DVDs spreche ich nicht von anamorphen und nicht anamorphen Filmen, sondern immer von 4:3 und 16:9. So lässt sich die Begriffsverwirrung hoffentlich in Grenzen halten. In freier Wildbahn sollten wir uns beim Begriff »anamorph« immer zuerst darüber klar werden, was in dem Zusammenhang genau gemeint ist.

Sinn des verzerrten Bildes

Wenn es so viele Probleme verursacht, warum wird das Video dann überhaupt verzerrt gespeichert? Das liegt hauptsächlich an zwei Dingen.

- Die Auflösung der DVD ist fest vorgegeben, egal welches Seitenverhältnis der Film tatsächlich hat.
- Das menschliche Auge ist für vertikale Auflösung empfindlicher als für horizontale.

Die Auflösung der PAL-DVD hat ein DAR von 1,25. Für die meisten Kinofilme, die in der Region zwischen 1,78 und 2,35 liegen, ist das ein extrem ungünstiger Wert. Bei unverzerrter Speicherung könnte im schlechtesten Fall nur etwa die Hälfte der verfügbaren vertikalen Auflösung ausgenutzt werden, um nicht links und rechts einen Teil des Bildes abschneiden zu müssen.

Das Bild verzerrt zu speichern, bietet die Möglichkeit, eine große Vielfalt von Seitenverhältnissen in der DVD-Auflösung unterzubringen und trotzdem so wenig wie möglich wertvolle vertikale Auflösung zu opfern.

Das Bild der DVD

Das Bild einer PAL-DVD ist mit einer Auflösung von 720 × 576 gespeichert ist. Für NTSC gelten 720 × 480. Die Auflösung ist fix, egal welchen Film wir darin verpacken. Wir können nur zwischen zwei Arten der Verzerrung wählen: 4:3 (wenig verzerrt) und 16:9 (stark verzerrt, entsprechend der professionellen Definition von »anamorph«) Daraus ergibt sich: Mit einer DVD als Quelle müssen wir immer eine Verzerrung berücksichtigen, denn das Bild hat nie das richtige Wiedergabe-Seitenverhältnis.

Betrachten wir das Bild der DVD etwas mehr im Detail. Als Beispiel nehmen wir *Die fabelhafte Welt der Amélie*, die praktischerweise nicht kopiergeschützt ist. Der Film ist mit einem Seitenverhältnis von 2,35 gedreht, muss aber mit dem vor-

gegebenen 1,78 einer 16:9-DVD auskommen. Die ungenutzte Auflösung wird einfach mit schwarzen Balken aufgefüllt, so dass Frame Nummer 27 661 so aussieht wie in Abbildung 8 auf Seite 44.

Möglich wäre es auch, das gleiche Bild in einer 4:3-DVD unterzubringen. Das würde weniger Verzerrung, größere Balken und weniger wertvolle vertikale Auflösung bedeuten. In der Anfangszeit der DVD wurden solche Sünden häufiger begangen (die erste Ausgabe von *Titanic* z. B.). Heutzutage begegnet man so etwas auf professionellen DVDs zum Glück nicht mehr.

Beim Abspielen muss nun das Video entzerrt werden. Dazu streckt der Decoder das Bild horizontal so weit, bis das Seitenverhältnis passt. Für eine 16:9-PAL-DVD ergeben sich 1047 Pixel in der Horizontalen. Unser Amélie-Beispiel sieht dann aus wie in Abbildung 9 auf Seite 44.

Für Filme, die nicht im Widescreen-Format gedreht sind, sieht die Sache ähnlich aus. Nehmen wir die Simpsons als Beispiel, die passend für den traditionellen Fernseher mit einem AR von 1,33 produziert werden. Dieses Verhältnis kommt der Auflösung der DVD (AR 1,25) recht nahe und wird deshalb mit der 4:3-Verzerrungsvariante gespeichert. Das sieht dann aus wie in Abbildung 10 auf Seite 44. Schwarze Balken gibt es bei 4:3-Material, wenn überhaupt, nur minimal. Im Beispiel sind nur die Ränder des Bildes ein wenig unsauber.

Falls jemand ins Grübeln kommt. Das Bild stammt aus Season 4: Maggies Mission-Impossible-Einlage in *A Streetcar Named Marge*. Welche Framenummer, weiß ich nicht. Die DVD ist kopiergeschützt, deshalb habe ich den Screenshot beim normalen Abspielen gemacht, nicht mit VirtualDub.

Bei der Wiedergabe passiert das gleiche wie im 16:9-Fall. Der Decoder streckt das Bild auf die korrekte Breite (vgl. Abbildung 11 auf Seite 44). Im Vergleich zur *Amélie* muss man die beiden Maggie-Versionen schon genauer unter die Lupe nehmen, um den Unterschied zu sehen. Am deutlichsten wird die Verzerrung an den Augen, die simpsonstypisch kreisrund sein müssen.

Zentrale Bedeutung des PAR

Jetzt stellt sich sicherlich die Frage, wie man die zur Wiedergabe nötige horizontale Auflösung ermittelt. Dafür ist das Pixel Aspect Ratio zuständig. Erinnern wir uns: das PAR beschreibt die korrekte Form eines einzelnen Pixels zum Zeitpunkt der Wiedergabe. Man könnte auch sagen, das PAR beschreibt, wie rechteckig die Pixel sein müssen. Es gibt nur vier Werte. Für PAL und NTSC je einen für 16:9 und 4:3.

Abbildung 8: Unverändertes 16:9-Frame.

Abbildung 9: Entzerrtes 16:9-Frame.

Abbildung 10: Unverändertes 4:3-Frame.

Abbildung 11: Entzerrtes 4:3-Frame.

	PAL	NTSC
4:3	12 / 11	10 / 11
16:9	16 / 11	40 / 33

Tabelle 4: PAR nach ITU-R BT.601.

Mit dem richtigen Wert aus dieser Tabelle lassen sich die Pixel des Bildes so weit in die Breite ziehen, dass das Seitenverhältnis passt. Allerdings kann ein Computermonitor ausschließlich quadratische Pixel darstellen. Anstatt also die Form der Pixel zu verändern, erhöhen wir in der Horizontalen die Anzahl der Pixel, um den gleichen Entzerrungseffekt zu erhalten.

Dass ein Computermonitor ausschließlich quadratische Pixel darstellen kann, gilt genau genommen nur für TFTs. Röhrenmonitore (CRT) sind theoretisch zu jeder beliebigen Auflösung und Pixelform fähig. Nur arbeiten alle Standard-Desktop-auflösungen mit quadratischen Pixeln.
Fast alle. Eine populäre Ausnahme ist die 1280 × 1024 auf einem CRT. Die Bildröhre des Monitors hat ein Seitenverhältnis von 1,33. Die 1280er Auflösung kommt auf ein Seitenverhältnis von 1,25. Die Abmessungen der Röhre und die Auflösung passen nicht zusammen, was zu rechteckigen Pixeln führt.
Heißt das nicht, dass man eigentlich alles leicht gequetscht sieht, wenn man am CRT mit 1280 × 1024 arbeitet? Ja. Die richtige, nicht verzerrte (quadratische), Auflösung wäre 1280 × 960. Das gilt nicht bei TFTs. Für die Modelle mit 1280 × 1024 als nativer Auflösung hat man die Abmessungen des Panels angepasst, so dass wieder quadratische Pixel entstehen.

Ok, zurück zu unserem Film. Die passende horizontale Auflösung auszurechnen, ist anhand der Tabelle sehr einfach. Wir müssen nur den zur DVD passenden Wert ablesen und in diese Formel einsetzen:

$$\text{hor. Wiedergabeauflösung} = \text{hor. DVD-Auflösung} \times \text{PAR}$$

Für die Amélie müsste die Rechnung dem Beispiel oben zufolge 1047 Pixel ergeben. Und das tut sie auch.

$$\text{hor. Wiedergabeauflösung} = 720 \times {}^{16}\!/_{11} = 1047{,}27$$

Wer aufmerksam mitrechnet, dürfte spätestens jetzt die Stirn runzeln. Unsere Zielauflösung enthält 1047 × 436 Pixel echtes Bild (ohne Balken), was einem Seitenverhältnis (hier das Display Aspect Ratio, DAR) von 2,40 entspricht. Trotzdem heißt es oben, der Film wäre in 2,35 gedreht?

Das stimmt auch. Der Unterschied entsteht, weil wir mit den Werten in der PAR-Tabelle dem Standard ITU-R BT.601 folgen, der ein etwas breiteres Bild als die weithin bekannten Werte fordert. Tabelle 5 auf Seite 46 bietet einen Vergleich von »Standard-DAR« und ITU-DAR.

Um keine Zweifel aufkommen zu lassen: Diese Art der Entzerrung ist korrekt, und zwar unabhängig vom Wiedergabegerät. ITU-R BT.601 zu ignorieren führt in der Mehrzahl aller Fälle zu einem falschen Seitenverhältnis im encodierten Video. Die ITU-Empfehlung zu ignorieren ist nur dann sinnvoll, wenn wir genau wissen, dass unsere DVD nicht nach dem Standard gemastert wurde.

bekanntes DAR	DAR nach ITU
1,33 (4:3)	1,36
1,78 (16:9)	1,82
1,85	1,89
2,35	2,40

Tabelle 5: Vergleich von generischem und ITU-DAR.

Wie genau die einzelnen PARs zustande kommen und welche Bedeutung das generische nicht-ITU PAR tatsächlich hat, damit beschäftigt sich das Spezialkapitel C.3 zur ITU-R BT.601 ausführlich. Für das DVD-Backup bringen uns diese Details keinen unmittelbaren Nutzen, weshalb ich das Kapitel eher als Bettlektüre hinterher vorschlage.

Kaffeepause

Die eine Hälfte des Anamorph-Themas ist geschafft. Im nächsten Kapitel beschäftigen wir uns damit, ob und wie wir das verzerrte Bild im encodierten Film beibehalten. Und obwohl das etwas leichtere Kost ist als dieses Kapitel, ist eine Kaffeepause an dieser Stelle eine gute Idee. Immerhin gehört die Anamorph-Thematik zum schwierigsten, was digitales Video zu bieten hat.

A.3.2.2 Anamorphes MPEG-4

Nicht nur auf der DVD, sondern auch im fertigen MPEG-4-Film wird anamorphe Speicherung immer beliebter. Da die Methode noch relativ neu ist, und um entscheiden zu können, wann ein anamorphes MPEG-4-Bild überhaupt sinnvoll ist, brauchen wir vor dem Praxisteil ein paar Hintergrundinfos.

Was nun an Zahlen folgt, kann ein wenig von den idealen Werten aus dem letzten Kapitel abweichen. Schuld daran ist eine Eigenart des Encodingverfahrens. MPEG-4-Codecs arbeiten nicht mit einzelnen Pixeln, sondern mit 16×16 Pixel großen Makroblocks. Auflösungen, die sich nicht komplett in vollständige Makroblocks aufteilen lassen, werfen zwar keinen modernen Codec aus der Bahn, senken aber die Effizienz der Kompression und damit die Qualität. Deswegen sollte die Zielauflösung immer sowohl horizontal als auch vertikal glatt durch 16 teilbar sein (Mod16-Kriterium).

Der Vorteil anamorphen MPEG-4-Videos

Dass anamorphes MPEG-4 immer beliebter wird, obwohl es einige Tücken birgt, hat einen einfachen Grund: Bildqualität. Als Beispiel taugt *Die wunderbare Welt der Amélie* wieder bestens, deren korrektes Wiedergabe-DAR abzüglich der Balken in der 2,35er-Region liegt. Croppen wir die Balken unter Berücksichtigung des Mod16-Kriteriums, bleiben von den 720 × 576 Pixeln der DVD noch 704 × 432 übrig, was einem Seitenverhältnis von 1,63 entspricht und heftige Eierköpfe beim Anschauen zur Folge hat. Um das zu beheben, sind zwei Möglichkeiten denkbar.

Horizontal Strecken

Das ist die gleiche Methode, die auch der DVD-Player anwendet. Um korrekt zu entzerren, müssen wir mit Hilfe des ITU-PARs das Bild auf 1024 × 432 Pixel in die Breite ziehen, womit wir allerdings eine beachtliche Anzahl Pixel pro Bild encodieren, die im Originalbild gar nicht vorhanden waren und deshalb keine echten Informationen tragen. Die Verschwendung ist beachtlich. Das gecroppte Originalbild enthält

$$704 \times 432 = 304\,128 \text{ Pixel.}$$

Das entzerrte Bild enthält

$$(1024 - 704) \times 432 = 138\,240 \text{ zusätzliche Pixel.}$$

Das heißt, wir encodieren 45 % mehr als eigentlich nötig, was an sich noch nicht extrem schlimm wäre. Allerdings enthalten diese 45 % zusätzliches Bild genau 0 % zusätzliche Details. Dazu kommt, dass das sehr große Bild die CPU beim Abspielen nicht gerade wenig beansprucht. Ältere Rechner stoßen da schnell an ihre Grenzen. Zum anderen sind diese Abmessungen selbst bei einem ½-DVD-Encoding kritisch, wenn wir gute Qualität erreichen wollen.

Vertikal stauchen

Das ist die klassische Methode, die seit DivX 3.11 überwiegend verwendet wird. Wir schrumpfen also die Auflösung auf 704 × 288. Das passt schon eher für ein 2-CD-Encoding, und kein Problem bei ½ DVD, allerdings um den Preis einer um ein Drittel niedrigeren vertikalen Auflösung. Dummerweise ist das menschliche Auge gerade für die vertikale Auflösung empfindlicher als für die horizontale. Die traditionelle Methode beschneidet das Bild also ausgerechnet in der wichtigeren Dimension.

Als Lösung bietet sich das anamorphe Bild an. Es verwirft keine wichtigen vertikalen Informationen und hält die Bildgröße in einem akzeptablen Rahmen. Inzwischen ist auch die Unterstützung von Encoder- und Decoderseite gut genug, so dass man anamorphe Encodings als alltagstauglich ansehen kann. Lediglich Standalone-Player verstehen anamorphes MPEG-4 nicht.

Als Haupteinsatzgebiet bietet sich klar das hochqualitative Encoding an, das die volle Auflösung (bis auf die schwarzen Balken) einer DVD beibehält. Stark komprimierte 1-CD-Filme profitieren weniger, da sie sowieso Details in Form von Auflösung opfern müssen, um weit genug geschrumpft werden zu können. Mein erster Eindruck ist, dass in diesem Bereich der Unterschied zwischen anamorph und nicht-anamorph nur gering ausfällt. Ein paar intensivere Tests könnten allerdings nicht schaden, um das zu bestätigen.

Anamorphe Varianten

Ein anamorphes MPEG-4-Video kann grundsätzlich auf drei verschiedene Arten erzeugt werden, immer mit der DVD als Quelle im Hinterkopf.

Originalbild behalten

Die einfachste Möglichkeit übernimmt das komplette Bild der DVD einschließlich der schwarzen Balken. Wenn uns keine Standalone-Zwänge die anderen Methoden verbieten, empfiehlt sich dieses Vorgehen nicht, denn die schwarzen Balken beanspruchen Bitrate. Und die können wir sinnvoller für das eigentliche Bild verwenden.

Nur Cropping

Das ist die bevorzugte Methode. Wir behalten die Auflösung der DVD grundsätzlich bei, schneiden aber die schwarzen Balken weg. Damit entfällt das Resizing, was uns zusätzliche Vorteile einbringt. Jeder Resizer sorgt für Bildrauschen vor allem auf der Zeitachse, was die Komprimierbarkeit senkt. Ohne Resizer können wir im günstigen Fall bei einer nur 20 % höheren Bitrate ein 40 % größeres Bild verwenden. Umgekehrt heißt das, die negativen Auswirkungen auf die Qualität durch das größere anamorphe Bild halten sich in Grenzen. Die Größe des Vorteils kann allerdings von Film zu Film stark schwanken.

Cropping und Resizing

Diese Methode ähnelt dem klassischen nicht-anamorphen Vorgehen. Wir schneiden die schwarzen Balken weg und verkleinern dann die Auflösung, allerdings

ohne die Verzerrung zu korrigieren. Auf diese Weise erhalten wir ein kleineres Bild, das aber mehr vertikale Auflösung beibehält als gewohnt. Tests zeigen allerdings, dass diese Methode kaum Qualitätsvorteile gegenüber einem entsprechenden nicht-anamorphen Video bietet.

Klar die sinnvollste Möglichkeit ist Cropping ohne Resizing, worauf wir uns im Rest des Encodingwissens auch konzentrieren. Die Frage stellt sich nun, wie wir dem Decoder mitteilen, dass er es mit einem anamorphen Bild zu tun hat.

Display Aspect Ratio und Pixel Aspect Ratio

Was der Decoder für eine korrekte Entzerrung benötigt ist die Angabe des passenden Wiedergabe-Seitenverhältnisses. Zur Erinnerung: Uns stehen zwei – im Ergebnis grundsätzlich identische – Möglichkeiten zur Verfügung, um ein Seitenverhältnis anzugeben.

- **Display Aspect Ratio (DAR).** Beschreibt das Seitenverhältnis des kompletten Bildes zum Zeitpunkt der Wiedergabe, stellt also nichts anderes als das Verhältnis von Breite zu Höhe dar.
- **Pixel Aspect Ratio (PAR).** Bezieht sich nicht auf das ganze Bild, sondern beschreibt die Form eines einzelnen Pixels zum Zeitpunkt der Wiedergabe.

Für DVD-Quellen ist das PAR interessanter als das DAR. Der Nachteil des DAR liegt darin, dass es sich ändert, je nachdem, wie viel schwarze Balken wir wegschneiden müssen. Das heißt, das DAR kann für jeden Film unterschiedlich sein und muss natürlich jedes Mal neu berechnet werden. Praktisch liegen die Werte zwar eng beieinander, nur sollten wir uns darauf nicht blind verlassen.

Im Gegensatz dazu gibt es für das PAR nur vier mögliche Werte (vgl. Seite 44, Tabelle 4), und zwar 16:9 und 4:3 jeweils für PAL und NTSC. Da uns DGIndex alle nötigen Informationen liefert, brauchen wir an der passenden Stelle nur diese Infos einsetzen. Das Cropping allein ändert nichts an der Form der einzelnen Pixel. Die sehen also im encodierten Video genauso aus wie auf der Quell-DVD.

Ein klassisches, nicht anamorph encodiertes, DVD-Backup hat immer ein PAR von 1:1 und ein DAR von »Bildbreite zu Bildhöhe«.

Sehen wir uns das an der kleinen Grafik in Abbildung 12 an, die ein Videobild aus 4×4 Pixeln darstellen soll, wie es korrekt entzerrt beim Abspielen aussieht. Das Bild ist nicht quadratisch, da DVD-Pixel eine rechteckige Form haben. Die Form des gesamten Bilds und die Form eines einzelnen Pixels lässt sich leicht berechnen:

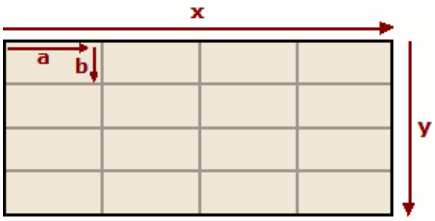

Abbildung 12: PAR und DAR vor dem Cropping.

DAR = x/y; PAR = a/b.

Beispiel für eine 16:9-PAL-DVD:

DAR = $1047/576$; PAR = $16/11$.

Jetzt bearbeiten wir das Bild, indem wir schwarze Balken entfernen. Nehmen wir an, eine Pixelreihe oben und eine Pixelreihe unten, was eine neue Grafik ergibt. Es gilt noch immer wie oben, allerdings mit kleinerem *y*:

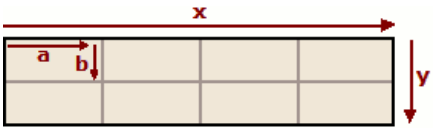

Abbildung 13: PAR und DAR nach dem Cropping.

DAR = x/y; PAR = a/b.

Und für die 16:9-PAL-DVD:

DAR = $1047/432$; PAR = $16/11$.

Durch das Cropping nimmt natürlich die Höhe des Bildes ab, *y* wird kleiner. Dadurch verändert sich auch der Wert des DAR, im Beispiel von 1,82 auf 2,42. Auf das PAR hat das Cropping dagegen keine Auswirkung, denn das veränderte DAR beruht nur auf einer verringerten Anzahl Pixelreihen pro Bild, *a* und *b* und damit die Form der verbleibenden Pixel, wird nicht angetastet.

Das AR-Flag und Decoderunterstützung

Die sinnvollste und einfachste Methode ist, dem Decoder das PAR mitzuteilen, damit er das Video richtig strecken kann. Dafür setzen wir ein AR-Flag in der Zieldatei, das wir uns vereinfacht als standardisiertes Feld vorstellen können, in dem die Angabe zum Seitenverhältnis steht. Der Decoder kennt die Stelle, an der das Flag gespeichert ist, und liest den Wert von dort. Ein AR-Flag können wir an zwei Stellen setzen.

Im MPEG-4-Videostream

Das ist die Stelle, an die die AR-Information eigentlich gehört. Schließlich ist das Seitenverhältnis eindeutig eine Eigenschaft der Videospur – eine recht zentrale noch dazu. Deshalb sollte – wenn wir eine AR-Info speichern – die immer mindestens im Videostream zu finden sein. Als praktischer Vorteil kommt dazu, dass das MPEG-4-AR-Flag kaum verloren geht, egal was wir in Zukunft vielleicht mit dem Film anstellen.

Da es nur äußerst beschränkt möglich ist, nach dem Encoding noch das Flag im Videostream zu setzen, muss der Encoder selbst diese Funktion anbieten. Xvid und x264 tun das, DivX leider nicht. Egal welcher Encoder, das MPEG-4-Flag ist immer das PAR.

Im Container

Das sollten wir als zusätzliche Möglichkeit ansehen, nicht als Ausrede, das Seitenverhältnis nicht im MPEG-4-Stream zu speichern. Bei einem Film mit ausschließlich im Container gesetzten Flag reicht ein sorgloses Re-Muxing, um die AR-Info zu verlieren.

MP4 bietet uns die Möglichkeit, das PAR anzugeben. Matroska dagegen besteht auf dem DAR. Im AVI-Container müssen wir auf ein AR-Flag ganz verzichten.

Sämtliche Flags nützen uns überhaupt nichts, wenn der Decoder keine Unterstützung mitbringt, um mindestens eines davon zu lesen und anzuwenden. Eigentlich sollte hier jetzt eine kleine Tabelle mit einer Übersicht folgen, welcher Decoder welche Art Flag erkennt. Nur hat sich schnell gezeigt, dass heute deutlich mehr Parameter mitspielen als noch vor einem Jahr. Bis ich deshalb die Zeit und Lust finde, einen ausführlichen Test zu veranstalten, empfehle ich zur Wiedergabe:

- Haali Media Splitter mit Unterstützung für AVI, Matroska, MP4 und OggMedia.
- ffdshow als Videodecoder. Auf der **Output**-Seite des Video-Konfigurationsdialogs muss der Haken bei **Use overlay mixer** gesetzt sein, damit ffdshow überhaupt AR-Flags erkennt.

A.3.3 Die passende Zielauflösung

Resizing heißt der Vorgang, den Film auf die richtige Zielauflösung zu skalieren. Besonders einfach ist das, wenn wir ein modernes anamorphes Encoding erstellen, denn dann findet kein Resizing statt. Wir schneiden nur – wie ein Kapitel weiter vorn erklärt – die schwarzen Balken rund um das Video weg und überlassen es dem Decoder, beim Abspielen das Bild korrekt anzuzeigen. Nur beim klassischen nicht-anamorphen Encoding müssen wir das Bild nach den Regeln in diesem Kapitel skalieren. Für anamorphe Encodings ist zumindest der Abschnitt über den Zusammenhang zwischen Auflösung und Qualität (Seite 54) wichtig. Anhand dessen können wir entscheiden, ob der Speicherplatz für ein anamorphes Bild überhaupt ausreicht.

Ein Beispiel

Nehmen wir wieder *Die fabelhafte Welt der Amélie* als Beispiel. Von der ursprünglichen PAL-Auflösung von 720 × 576 Pixeln bleiben nach dem Cropping (je 4 Pixel links und rechts, je 72 Pixel oben und unten) noch 712 × 432 Pixel übrig. Angenommen, wir entscheiden uns im fertigen Encoding für eine horizontale Auflösung von 608 Pixeln. Das entspricht 85,4 % der ursprünglichen 712 Pixel. Die Höhe um den gleichen Faktor geschrumpft ergibt eine Zielauflösung von 608 × 369 Pixeln, und Frame Nummer 160 932 sieht aus wie Abbildung 14.

Abbildung 14: Falsches Resizing ohne Berücksichtigung der Verzerrung.

Abbildung 15: Korrektes Resizing mit Berücksichtigung der Verzerrung.

Sofort fällt auf, dass diese extrem langgezogene Kopfform kaum der Realität entsprechen kann. Was wir bisher nicht bedacht haben, ist das verzerrt gespeicherte Bild der DVD. Berücksichtigen wir das korrekt, erhalten wir eine Zielauflösung von 608 × 256 Pixeln wie in Abbildung 15. Das sieht nicht nur deutlich natürlicher aus, sondern ist auch korrekt.

Berechnung der Zielauflösung

Die passende Zielauflösung zu berechnen, ist schon deswegen nicht schwer, weil uns das Encoding-Frontend diese Arbeit abnimmt. Doch auch manuell stellen sich uns keine größeren Hürden in den Weg.

Zusätzlich zur eigentlichen Rechnung müssen wir auch das Mod16-Kriterium von Seite 46. Die Zielauflösung muss in beiden Dimensionen glatt durch 16 teilbar sein.

Um die endgültige Auflösung zu ermitteln, wählen wir zuerst die Zielbreite: 608 Pixel (38 × 16). Welche Breite sinnvoll ist, hängt hauptsächlich davon ab, wie stark wir den Film schrumpfen wollen. Genaueres dazu später in diesem Kapitel. Die passende vertikale Auflösung erhalten wir mit dieser Formel:

$$\mathit{Zielhöhe} = \frac{\mathit{Zielbreite}}{\mathit{AR\,nach\,Cropping} \times \mathit{PAR}}$$

- **Zielbreite:** Kein Problem hier. Diese Zahl haben wir ja gerade selbst festgelegt. An dieser Stelle setzen wir einfach unsere 608 Pixel ein.
- **AR nach Cropping:** Bezeichnet das Seitenverhältnis des Bildes nach dem Cropping der schwarzen Balken. An dieser Stelle brauchen wir uns um Verzerrungen noch keine Gedanken machen. AR bedeutet hier einfach das Verhältnis von gecroppter Breite zu gecroppter Höhe: 712 / 432 in unserem Fall.
- **PAR:** Hier kümmern wir uns schließlich darum, das Bild korrekt zu entzerren. Wir wissen, dass es sich um eine 16:9-PAL-DVD handelt. Das steht meistens auf der Verpackung. Auch DGIndex liefert uns beim Indexieren des Videos (vgl. Seite 86 f.) die nötigen Informationen. Das richtige PAR brauchen wir dann nur noch aus der Tabelle auf Seite 44 ablesen: 16 / 11.

Eingesetzt und ausgerechnet ergibt das Ganze 253,6 Pixel in der Vertikalen.

$$\mathit{Zielhöhe} = \frac{608}{\frac{712}{432} \times \frac{16}{11}} = 253{,}6$$

Dass ein Bildschirm natürlich natürlich keine Bruchteile von Pixeln darstellen kann, brauchen wir nicht extra zu beachten, weil die Zielhöhe sowieso das Mod16-Kriterium einhalten muss. Entsprechend runden wir das Ergebnis der Formel noch auf das nächste Vielfache von 16 und erhalten 256 (16×16).

Abweichung vom optimalen AR: Aspect Error

Wie dankbar der Codec für das Mod16-Kriterium auch ist, wir erkaufen das mit einem Nachteil. Da wir die Auflösung nicht pixelgenau zurechtrücken können, entsteht eine Abweichung zum eigentlich richtigen Ergebnis. Das Runden auf Mod16 verursacht in unserem Beispiel einen Fehler von knapp 1 %. Die ursprüngliche Verzerrung ist nicht ganz exakt korrigiert worden. Da wir eine zu hohe vertikale Auflösung gewählt haben, enthält das Bild minimale Eierköpfe.

Geringe Fehler – etwa bis 2,5 % – sind unproblematisch, weil die Verzerrung zu klein bleibt, um spürbar zu werden. Das Mod16-Kriterium bietet eine ausreichend feine Abstufung, um größere Abweichungen zu vermeiden. Wer den Fehler doch selbst berechnen möchte, hier ist die Formel:

$$\textit{Fehler in } \% = \left(\frac{\textit{Mod16-Zielhöhe}}{\textit{korrekte Zielhöhe}} - 1 \right) \times 100$$

Um den AR-Fehler komplett zu vermeiden, gibt es zwei Möglichkeiten.

- **Anamorphes Encoding:** Fällt das Resizing weg, fällt die Fehlerquelle weg. Auch der Decoder unterliegt beim Abspielen und Entzerren nicht der Mod16-Restriktion.
- **Cropping:** Jedes gecroppte Pixel verändert den AR-Fehler. Wir können also so lange mehr Pixel als eigentlich nötig abschneiden, bis der Fehler Null wird. Mir persönlich ist es allerdings lieber, eine unmerkliche Verzerrung im Bild zu haben als ein Stück Bild wegzuwerfen.

Auflösung und Qualität

Mit diesem Wissen ausgerüstet, können wir nun daran gehen, eine günstige Auflösung zu wählen. Günstig bedeutet: eine Auflösung, die gute Qualität bietet. Dazu benötigen wir einen Qualitätsindikator.

BPF als Qualitätsindikator

Nein, nicht die Bitrate. Obwohl man sofort an die Bitrate denkt, wenn man einen digitalen Film sieht, interessiert uns die während des gesamten Backupprozesses überhaupt nicht. Die Angabe, wie viele Bits dem Codec im Durchschnitt pro Sekunde zur Verfügung stehen, ist viel zu ungenau, um eine brauchbare Aussage über die Qualität zu liefern. Viel interessanter ist die *relative Bitrate,* besser bekannt als *BPF-Wert.* Der gibt an, wie viele Bits zum Encodieren eines einzelnen Pixels durchschnittlich zur Verfügung stehen. Das heißt, er berücksichtigt alle direkt berechenbaren Einflussfaktoren (Zielgröße, Filmlänge, Framerate, Auflösung). Allerdings sagt er nur bedingt Definitives über die endgültig sichtbare Qualität aus. Ein Drama mit vielen langen Dialogen und hauptsächlich langsamen Szenen braucht für die gleiche sichtbare Qualität einen niedrigeren Wert als ein schneller Actionstreifen. Sehr nachtlastige Filme geben sich auch mit einem niedrigeren Wert zufrieden, da Szenen im hellen Tageslicht viel mehr Details enthalten und deshalb auch mehr Bits benötigen. Deshalb kann der BPF-Wert immer nur ein Anhaltspunkt für die sichtbare Qualität sein. Berechnet wird die relative Bitrate mit dieser Formel:

$$BPF = \frac{Videogröße\ in\ Byte \times 8}{Zielbreite \times Zielhöhe \times Anzahl\ Frames}$$

Die Videogröße bezeichnet dabei nicht die Zielgröße des gesamten Films, sondern nur die Größe der reinen Videospur.

$$Videogröße = Gesamtzielgröße - Tonspuren - Untertitel - Containeroverhead$$

Da sich der Containeroverhead nicht in eine einfache Formel packen lässt, ist eine manuelle Berechnung reine Quälerei. Das Encoding-Frontend nimmt uns die Arbeit sowieso ab. Als kleines Beispiel bleiben wir bei der Amélie, und zwar in dieser Konfiguration:

- Filmlänge: 1 h 58 min 52,32 s bei 25 Frames pro Sekunde = 178 308 Frames.
- Auflösung: Wie in den Beispielen weiter vorn: 608 × 256.
- Zielgröße: 1493 MB (⅓ DVD).
- Videogröße: Mit beiden AC3-Tonspuren (je 384 kbit/s) ergeben sich 843,4 MB (884 405 248 Byte).

$$BPF = \frac{88.405.248 \times 8}{608 \times 256 \times 178.308} = 0{,}255$$

Damit wissen wir, dass ein einzelnes Pixel im Schnitt mit einem Viertel Bit Speicherplatz auskommen muss. Das allein bringt uns noch nicht weiter. Um BPF-Werte in sinnvolle und weniger sinnvolle Kategorien aufzuteilen, braucht es Erfahrungswerte.

Sinnvolle BPF-Werte

Erfahrungswerte existieren auch. Für MPEG-4-ASP-Codecs (Xvid, DivX) und MPEG-4-AVC-Codecs (x264) hat sich jeweils ein Bereich herauskristallisiert, in dem sich die relative Bitrate bewegen sollte.

	MPEG-4 ASP	MPEG-4 AVC
Untergrenze	0,20	0,15
Obergrenze	0,40	0,35

Grundsätzlich kann man sagen: Unterschreiten wir den sinnvollen Bereich deutlich, äußert sich das in schlechter Bildqualität, v. a. hässliche Makroblock-Artefakte werden dann sichtbar. Deutlich darüber erzeugt nur eine größere Datei bei kaum noch höherer Qualität. 1-CD-Encodings orientieren sich in Richtung der Untergrenze, Encodings für zwei oder mehr CDs in Richtung der Obergrenze. Über anamorphe Auflösungen eine Aussage zu treffen, ist schwieriger. Da es sich dabei um hochqualitative Backups handelt, würde ich sie in die Mitte mit einer Tendenz nach oben einordnen.

Nie vergessen sollten wir den Richtwertcharakter des BPF-Werts. Die Werte in der Tabelle sind Anhaltspunkte, keine in Stein gemeißelten Grenzen, und Ausreißer können durchaus vorkommen: z. B. ein Film, der mit 0,15 höchsten Ansprüchen genügt; oder ein anderer, der erst ab 0,45 keine sichtbaren Makroblocks mehr aufweist. Um solche Spezialfälle in den Griff zu bekommen, brauchen wir einen genaueren Qualitätsindikator, den Kompressionstest. Da der jedoch stark vom Encoding-Frontend abhängt, beschäftigen wir uns damit im Praxisteil des Encodingwissens.

Wahl einer sinnvollen Auflösung

Mit all diesem Wissen ausgerüstet, können wir nun für jeden Film eine passende Zielauflösung ermitteln, die einen akzeptablen AR-Fehler mit einer sinnvollen relativen Bitrate verbindet.

Je kleiner die Auflösung, desto weniger Details enthält ein Einzelbild unabhän-

gig von jeder Kompression. Abgespielt wird der Film aber wahrscheinlich weiterhin im Vollbild. Es ist zwar kein Problem, ein Bild so weit zu strecken, dass es den ganzen Bildschirm ausfüllt, nur lassen sich dadurch die Details nicht zurückgewinnen. Deshalb führt eine kleinere Auflösung zwar zu einem höheren BPF-Wert, gleichzeitig aber auch zu einem Detail- und damit Qualitätsverlust. Um den nicht zu groß werden zu lassen, sollten wir Auflösungen unter 512 Pixel in der Horizontalen lieber vermeiden.

Ein Maximum ist einfacher definiert. Spätestens, wenn sowohl Höhe als auch Breite die gecroppte Originalauflösung erreicht haben, verliert eine weitere Steigerung ihren Sinn, weil wir dann in beiden Dimensionen Pseudodetails erfinden, die auf der DVD nie vorhanden waren. Schöner lässt sich Bitrate nicht verschwenden.

Für einen PAL-Film im 16:9-Format bedeutet diese Regel eine maximale horizontale Auflösung um die 1024 Pixel. Das ist verdammt viel, weshalb uns in der Regel schon deutlich vorher der beschränkte Speicherplatz einen Strich durch die Rechnung macht. Zusätzlich braucht ein größeres Bild auch mehr Rechenleistung beim Abspielen. Für die Realität heißt das, dass wir uns meistens in der Region zwischen 640 und 720 horizontalen Pixeln bewegen.

Einfacher stellt sich die Wahl für anamorphe Encodings dar. Entweder reicht der Speicherplatz für einen sinnvollen BPF-Wert aus, oder wir müssen uns vom anamorphen Bild verabschieden.

Doch die Auflösung ist nicht die einzige Schraube, an der sich drehen lässt. Um den BPF-Wert zu beeinflussen, können wir auch:

- die Anzahl der Tonspuren ändern,
- die Tonspuren stärker oder schwächer komprimieren,
- die Zielgröße verändern.

Das wichtigste an der Auflösungswahl ist, die Grenzen als die fließenden Richtlinien zu sehen, die sie sind. Mit wachsender Erfahrung weiß man irgendwann sowieso intuitiv, welcher Film ein paar mehr Bits nötig hat und welchen man ein bisschen mehr quetschen kann.

A.4 Videocodec-Konfiguration

Thema dieses Abschnitts ist die Konfiguration der Videocodecs. Welches Kapitel für die Praxis interessanter ist, hängt vom Encoding-Frontend ab. Die empfohlenen Einstellungen unterscheiden sich jedenfalls nicht zwischen VfW- und Kommandozeilen-Kapitel.

A.4.1 VfW-Interfaces

VfW steht für *Video for Windows,* ein Videoframework von Microsoft, das seit Windows 3.1 existiert und die AVI-Datei eingeführt hat. Wegen einiger Beschränkungen der Technologie – z. B. ist VfW nicht darauf ausgelegt, B-Frames zu speichern – mussten für moderne Codecs Hacks entwickelt werden. Besonders nachteilig ist das für MPEG-4-Codecs, da dadurch die MPEG-4-Spezifikation verletzt wird. Es ist also nicht möglich, Xvid mit B-Frames in AVI zu speichern, ohne mit den Regeln des Standards zu brechen. Andere neuere Technologien – wie MP4 – arbeiten mit VfW überhaupt nicht zusammen.

Um es ganz unverblümt zu sagen: VfW/AVI ist veraltet. Es ist höchste Zeit, diesen Klotz vom Bein der digitalen Videowelt zu entfernen und auf aktuelle Technologien zu setzen, die nicht den Einschränkungen eines inzwischen über 13 Jahre alten Frameworks unterliegen.

Alle drei Encodingwissen-Codecs (Xvid, DivX, x264) sind in einer VfW-Version zu haben. Bei x264 war das von Anfang an eine Lösung, die eher der Abwärtskompatibilität dient, denn der Fokus ist eindeutig der x264.exe-Kommandozeilenencoder, der nicht den VfW-Einschränkungen unterliegt. Auch für Xvid existiert inzwischen mit Xvid_Encraw ein Encoder für die Kommandozeile. Nur DivX ist weiterhin ausschließlich für VfW verfügbar.

A.4.1.1 Xvid v1.2 VfW-Konfiguration

Xvid ist ein sehr genau konfigurierbarer Codec, man könnte Bücher darüber schreiben, wann welche Kombination von Optionen sinnvoll ist. Um den Guide

nicht ausufern zu lassen, erkläre ich die einzelnen Einstellungen nur kurz. Wer tiefer einsteigen will, dem kann ich nur Selurs »*Wissenswertes rund um Xvid*« und crusty's (englische) *Xvid FAQ* ans Herz legen.

Einstellungen für den 1st Pass

Profile und deren Level schalten einzelne Optionen des Codecs frei oder sperren sie, um Konformität zu den gleichnamigen MPEG-4-Profilen zu gewährleisten. Für ein normales DVD-Encoding bietet sich **Advanced Simple @ L5** (Advanced Simple Profile, Level 5) an (Abbildung 16). Über den **More**-Button erreichen wir den Dialog für die Feineinstellungen (Abbildung 17).

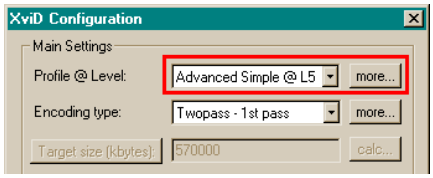

Abbildung 16: Hauptfenster oben der Xvid-Konfiguration

Der **Quantization type** bestimmt die Matrix, die beim Encodieren verwendet wird. **MPEG** erzeugt ein etwas schärferes Bild auf Kosten der Kompression und eignet sich eher für die höhere Datenrate einer großzügigen Zielgröße. **H.263** ergibt dagegen ein etwas weicheres Bild und erhöht damit die Kompression ein wenig; geeignet eher für die niedrigen Datenraten von 1-CD-Encodings. Für Backups auf einer halben DVD bevorzuge ich eine der **MPEG-Custom**-Matrizen von *Sharktooth*, und zwar meistens die EQM V3HR.

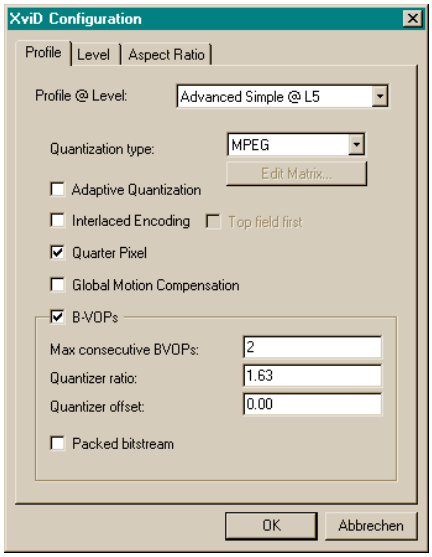

Abbildung 17: Xvids MPEG-4-Profileinstellungen.

In sehr hellen und sehr dunklen Bereichen des Bilds nimmt das menschliche Auge weniger Details wahr. Das macht sich **Adaptive Quantization** zu nutze und komprimiert solche Bereiche stärker, um die eingesparte Datenrate an Stellen zu verwenden, die es nötiger haben. Bei geringen Bitraten macht sich das positiv bemerkbar.

Quarter Pixel erhöht die Genauigkeit, mit der Bewegungen gespeichert werden, von einem halben auf ein Viertel Pixel. Das wirkt sich positiv auf Details und Schärfe aus, deshalb sollten wir QPel normalerweise einschalten.

Global Motion Compensation versucht die Kompression zu erhöhen, indem es nach gemeinsamen Bewegungsvektoren in der Szene sucht. Zooms und Kameraschwenks bieten sich dafür besonders an. Ob GMC wirklich spürbare Unterschiede bringt, ist nicht ganz unumstritten. Inzwi-

schen gehen die Meinungen mehr in die Richtung, dass sich GMC nicht lohnt und ausgeschaltet bleiben sollte. Ein starkes Indiz dafür ist auch, dass im technisch fortschrittlicheren AVC-Standard GMC nicht verfügbar ist.

Sowohl QPel als auch GMC bremsen nicht nur den Encoding-Vorgang, sondern benötigen auch beim Abspielen Rechenleistung. Auf alten Rechnern (500 MHz abwärts) sollten wir uns deshalb besonders den gemeinsamen Einsatz gut überlegen. Auch Standalone-Player kommen oft mit beiden nicht zurecht.

B-VOPs (technisch für B-Frames) sollten wir ohne guten Grund nicht abschalten. **Max consecutive B-VOPs** gibt an, wie viele B-Frames maximal direkt hintereinander stehen dürfen. Mit der Standardeinstellung 2 wäre also eine solche Bildsequenz möglich: IPBBP, bei **3** wären es dann IPBBBP usw. Wichtig: Die Einstellung definiert nur das Maximum. **2** bedeutet also nicht, dass *immer* zwei B-Frames hintereinander stehen. Es heißt nur, dass *niemals mehr* als zwei aufeinander folgen dürfen. Innerhalb dieser Grenze berechnet Xvid dann die günstigste Möglichkeit. Die Standardeinstellung ist ein sinnvoller Wert, nur für Standalone-Player müssen wir evtl. auf **1** herunter gehen.

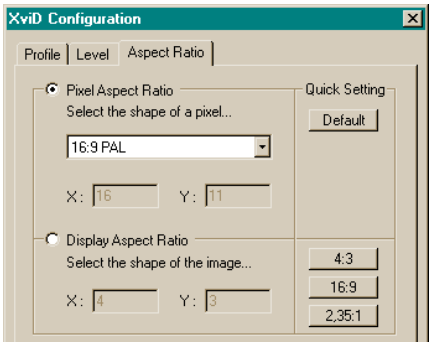

Abbildung 18: Xvids AR-Konfiguration.

Quantizer ratio und **Quantizer offset** bestimmen, wie hoch B-Frames im Vergleich zu P-Frames komprimiert werden. Die Standardwerte (**2**/**1.5**/**1**) sind in eine brauchbare Wahl. Für meine hochqualitativen Backups nehme ich auch gerne *Didées* Lieblingswerte: **2**/**1.63**/**0**.

Packed Bitstream bestimmt, wie die B-Frames in der Videodatei gespeichert werden und sollte normalerweise ausgeschaltet sein. Ausnahme sind wieder einmal manche Standalones, die bei deaktiviertem **Packed Bitstream** gerne Probleme machen.

Damit wechseln wir zum Register Aspect Ratio (Abbildung 18). Unter **Pixel Aspect Ratio** erlaubt Xvid die Angabe eines PAR, der im AR-Flag des MPEG-4-Streams gespeichert wird. Das ist nur für anamorphe Zieldateien interessant. Ein klassisches Encoding mit quadratischen Pixeln verwendet immer die Standardeinstellung **Square**. Die nicht-quadratischen Einträge entsprechen den Werten gemäß ITU-R BT.601 (vgl. Tabelle 4, Seite 44). Außerdem können wir mit **Custom** die beiden Eingabefelder unter der Combobox aktivieren, um einen beliebigen PAR anzugeben.

Die zweite Möglichkeit ist die Angabe des **Display Aspect Ratio**. Bis vor kur-

zem hieß diese Option noch verwirrenderweise *Picture Aspect Ratio*, hat aber schon immer das DAR bezeichnet. Wenn wir ein DAR angeben, ist wichtig, immer ganze Zahlen zu verwenden. 2,35:1 müssten wir also als 235:100 angeben.

Damit ist dieses Fenster abgehandelt und wir können zurück in den Hauptdialog. Mit **Twopass - 1st pass** stellen wir den ersten Codierdurchgang ein (Abbildung 19) und rufen über den **More**-Button die Details auf (Abbildung 20). Im angegebenen **Stats file** werden die Informationen aus dem 1st Pass gespeichert, die der 2nd Pass dann weiterverwendet. Normalerweise kümmert sich das Encoding-Frontend um den richtigen Eintrag. **Full quality first pass** aktiviert sämtliche eingestellten Encoder-Optionen schon im ersten Durchgang. Normalerweise sind hier einige Optionen abgeschaltet (z. B. VHQ), die nur bremsen und im 1st Pass noch nichts bringen. Die Option sollten wir nur dann einschalten, wenn wir die Videodatei des 1st Pass behalten wollen. Wollen wir aber nicht. Deshalb setzen wir auch den Haken bei **Discard first pass**. Xvid schreibt dann keine komplette Videodatei sondern nur ein wenig Müll. Das spart Plattenplatz.

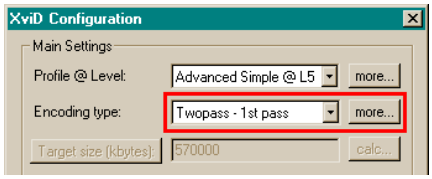

Abbildung 19: Einstellung für 1st Pass im Xvid-Hauptdialog

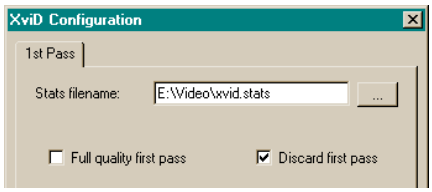

Abbildung 20: Xvids 1st-Pass-Optionen.

Damit kehren wir zurück ins Hauptfenster und kümmern uns um die Zones (Abbildung 21). **Zones** definieren Abschnitte innerhalb des Videos, für die unabhängig eine Reihe von Optionen definiert werden können. Offensichtlichster Anwendungsbereich für Zonen sind die Credits. Eine Zone, die sich über den ganzen Film erstreckt, existiert immer. Über den Button **Zone Options** gelangen wir in den Konfigurationsdialog.

Bei **Start frame #** (Abbildung 22) tragen wir das Bild ein, mit dem die Zone beginnen soll. Für die erste Zone ist das **0**, der Filmanfang. Das Ende einer Zone ist immer der Anfang der darauf folgenden oder das Ende des Films.

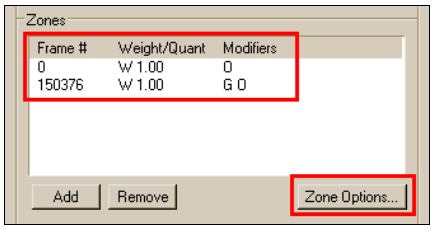

Abbildung 21: Zonenliste im Xvid-Hauptdialog.

Zonen können entweder mit einem konstanten **Quantizer** (vereinfacht: Kompressionsfaktor) oder einem relativen Gewicht (**Weight**) versehen sein. **1.00** ist dabei der Standard. Höher gewichtete Zonen bekommen mehr Qualität zugeteilt, geringer gewichtete müssen sich mit weniger zufrieden geben. Im Normalfall können wir **Weight 1.00** einfach übernehmen.

Begin with a keyframe erzwingt am Anfang der Zone ein I-Frame. **Greyscale encoding** verwirft alle Farbinformationen, so dass wir ein Schwarzweiß-Video erhalten. **Chroma optimizer** wirkt Pixeltreppchen an scharfen Kanten entgegen, bremst aber den Encoding-Vorgang etwas. Obwohl ich mir nicht sicher bin, ob es wirklich sichtbare Verbesserungen bringt, lasse ich die Option immer eingeschaltet. Der **Cartoon Mode** ist nützlich bei Cartoons – wer hätte das gedacht. Cartoon heißt dabei allerdings richtig klassisch gezeichneten Zeichentrick à la *Tom & Jerry* oder *The Simpsons*. Animationen wie die *Monster AG* gelten für Xvid nicht als Cartoon, da sie eher die Eigenschaften eines Realfilms aufweisen. Die **BVOP sensitivity** schließlich wirkt sich darauf aus, wie gern Xvid B-Frames setzt. Positive Werte ermutigen Xvid zu mehr B-Frames, negative Werte schrecken den Codec eher ab. Die **0** können wir als sinnvollen Standardwert stehen lassen.

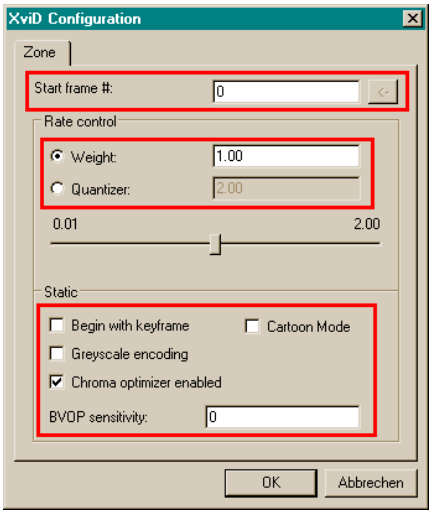

Abbildung 22: Detailkonfiguration der Xvid-Zonen.

Normalerweise definieren wir nur eine einzige Zone über den ganzen Film. Um das Einrichten einer zusätzlichen Zone für die Credits kümmert sich das Encoding-Frontend automatisch.

Über **OK** gelangen wir zurück in den Hauptdialog, wo wir uns im unteren Teil des Fensters um den **More**-Abschnitt kümmern (Abbildung 23). Hier stellen wir das **Quality Preset** auf (**User defined**) und öffnen über den **More**-Button den Dialog für die Detaileinstellungen (Abbildung 24).

Die **Motion search precision** legt fest, wie intensiv Xvid nach Bewegungen sucht. Es gibt kaum einen Grund von **6 - Ultra High** abzuweichen. **0 - None** schaltet die Bewegungssuche ganz ab, was einen Film zufolge hat, der ausschließlich aus I-Frames besteht.

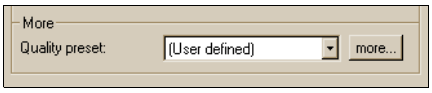

Abbildung 23: »Quality Preset« im Xvid-Hauptdialog.

VHQ rechnet für die einzelnen Makroblocks verschiedene Szenarien durch und entscheidet sich dann für das mit der geringsten Anzahl Bits. Generell gilt: Jede höher der **VHQ mode** desto besser die Qualität und desto langsamer das Encoding. **1** oder **2** bringen im Vergleich zum Bremseffekt den höchsten Qualitätsgewinn. Ich bin Qualitätsfanatiker genug, um meistens bei der höchsten Einstellung **4** zu bleiben. Wenn wir GMC aktiviert haben, sollte VHQ nicht abgeschaltet sein. **Use VHQ for bframes too** aktiviert den VHQ-Modus auch für B-Frames. Da es sich dabei immer um VHQ 1 handelt, leidet

die Geschwindigkeit wenig. Zwar wären auch höhere VHQ-Modi für B-Frames möglich. Laut Xvid-Entwickler syskin würden die aber kräftig bremsen und kaum einen spürbaren Effekt auf die Qualität haben.

Use chroma motion veranlasst Xvid, bei der Suche nach Bewegung nicht nur die Helligkeitsinformationen (Luminanz) zu berücksichtigen, sondern auch die Chrominanz (Farbe). Das steigert die Genauigkeit der Ergebnisse und damit die Qualität.

Turbo ;-) beschleunigt die Berechnung von B-Frames und QPel, erreicht dadurch aber nicht immer das absolute Qualitätsmaximum. Der Unterschied ist kaum jemals sichtbar, als Qualitätsfanatiker lasse ich Turbo trotzdem meistens ausgeschaltet.

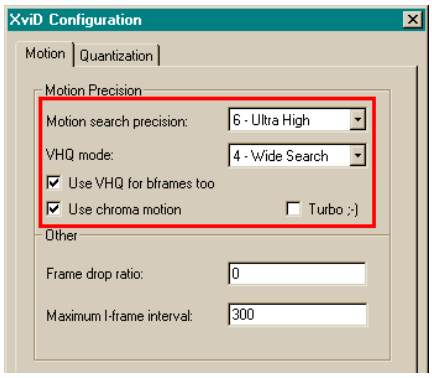

Abbildung 24: Konfiguration der Xvid-Featues.

Von der **Frame drop ratio** sollte jeder die Finger lassen, der nicht ganz genau weiß, was er da tut. Auch das **Max. I-frame interval** kann problemlos auf dem Standardwert bleiben. An günstigen Stellen setzt Xvid sowieso automatisch I-Frames (Man nennt sie auch Keyframes). Ein sehr niedriger Wert macht höchstens bei Captures direkt in Xvid Sinn, die man hinterher noch schneiden will. Denn Schneiden funktioniert nur an I-Frames. Genaueres dazu steht im Kapitel C.1 über manuelles Splitting ab Seite 141.

Im Register **Quantization** (Abbildung 25) ist zuerst Trellis quantization interessant. Trellis »überdenkt« die einmal getroffene Quantisierungsentscheidung und versucht sie zu verbessern. Heftige Gewinne bei der Qualität dürfen wir davon nicht erwarten, andererseits bremst Trellis auch nicht besonders und kann deshalb bedenkenlos eingeschaltet bleiben.

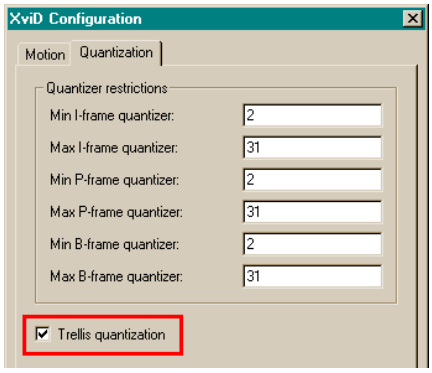

Abbildung 25: Xvid-Quantisierungsoptionen.

Sämtliche **Min Quantizer** vom Standardwert **1** auf **2** zu setzen, ist hauptsächlich eine kosmetische Entscheidung. **1** als kleinster Quantizer ist vor allem deswegen der Standard, um das ewige Gejammer über zu klein geratene Filme ein wenig einzudämmen. Ist der Film nämlich gut genug komprimierbar, um bei maximaler Qualität den vorgesehenen Platz nicht auszunutzen, wird die Datei kleiner als angegeben. Quant 1 verschwendet in diesem Fall zusätzlich Bitrate und steigert damit die Dateigröße, ohne die Qualität weiter zu verbessern. Deswegen ist **2** der eigentlich sinnvollere Wert.

Damit begeben wir uns zurück in den Hauptdialog, wo wir über den Button unten in der Mitte die **Other Options** aufrufen.

Number of threads in Abbildung 26 ist erst seit Xvid Version 1.2 verfügbar. Hier stellen wir ein, wie viele Threads Xvid zum Encodieren verwenden soll. Für Computer mit nur einer CPU sollten wir diesen Wert immer auf **0** belassen. Wer Hyperthreading, Dual Core oder tatsächlich ein System mit mehreren Prozessoren sein eigen nennt, muss ein wenig testen, welche Einstellung die schnellsten Ergebnisse bringt. Die Anzahl der CPUs oder CPU-Kerne ist ein guter Startwert.

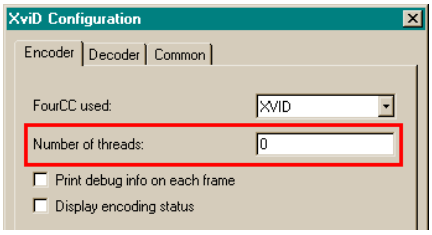

Abbildung 26: Anzahl Encodingthreads ab Xvid Version 1.2.

Weiter unten haben wir die Möglichkeit, das Statusfenster abzuschalten, das Informationen über das gerade laufende Encoding anzeigt. Es bremst zwar kaum, aber wer nicht die ganze Zeit vor dem Rechner sitzt und die Statistiken bewundert, wird es kaum benötigen.

Und damit ist die Konfiguration des 1st Pass endlich beendet. Keine Angst, der 2nd Pass geht schneller.

Einstellungen für den 2nd Pass

Im Hauptdialog stellen wir **Twopass - 2nd pass** ein. In der Regel kümmert sich das Encoding-Frontend um die richtige **Target size**. Wenn wir sie doch manuell eingeben: Hierhin gehört die Größe der reinen Videospur, nicht die Zielgröße des kompletten Films. Der Wert ist der gleiche wie auf Seite 55 im Kapitel über die Zielauflösung beschrieben.

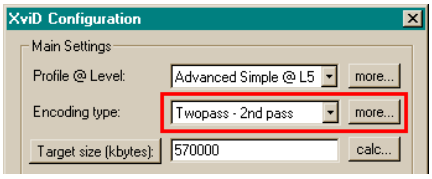

Abbildung 27: Einstellung für 2nd Pass im Xvid-Hauptdialog.

Ein Klick auf den **More**-Button bringt uns zu den Details, die wir allerdings komplett und guten Gewissens auf den Standardeinstellungen belassen können. Genauso verändern wir alle anderen Einstellungen im Vergleich zum 1st Pass nicht. Und damit ist die Konfiguration des 2nd Pass auch schon abgeschlossen.

A.4.1.2 DivX 6.4 VfW-Konfiguration

Bevor wir loslegen, eine Info vorab, um Anfragen vorzubeugen. Das Encodingwissen beschäftigt sich nicht mit dem Container namens *DivX Media Format*, der Menüs usw. möglich macht. Daran wird sich auch in Zukunft nichts ändern, das es sich bei DMF im Grund um ein aufgebohrtes AVI handelt und ich der starken Überzeugung bin, dass die veraltete VfW/AVI-Technologie hauptsächlich den Fortschritt der digitalen Videotechnologie behindert und deshalb nicht weiter am Leben erhalten werden sollte.

Der DivX-Codec ist für seine einfache Handhabung bekannt und hat durch das Zertifizierungsprogramm von DivX Inc. eine entscheidende Bedeutung für MPEG-4-fähige Hardwareplayer erlangt. Dem jüngsten *Doom9-Codecvergleich* zufolge erreicht er inzwischen auch bei Qualität und Geschwindigkeit das Niveau seines Schwestercodecs Xvid. Ob die einfachere Konfiguration und die problemlosere Kompatibilität mit zertifizierten Hardwareplayern den Kauf lohnt, muss jeder für sich selbst entscheiden.

Der einzige für uns relevante Unterschied zur Version 6.2 ist der neue schnelle 1st Pass, der erst ab DivX 6.4 existiert. Ansonsten ist alles beim Alten geblieben.

Einstellungen für den 1st Pass

Wir öffnen die DivX-Konfiguration und befinden uns im Register **Main**. Um volle Kontrolle über alle Optionen zu erhalten, müssen wir unter **Certification Profile** auf **Unconstrained** umschalten (Abbildung 28) und damit die Profile deaktivieren.

Wer seine Filme auf einem Hardwareplayer abspielen will, sollte das Profil auswählen, für das das Gerät zertifiziert ist. Wahrscheinlich sind dann einige der weiter unten erklärten Optionen nur eingeschränkt oder nicht nutzbar.

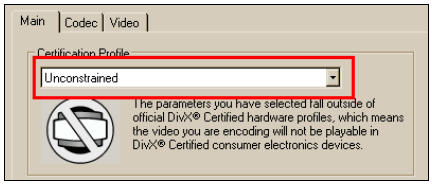

Abbildung 28: DivX-Profilauswahl.

Weiter unten bei **Rate Control Mode** (Abbildung 29) wählen wir den passenden Modus für der ersten Encodingdurchgang. DivX bietet zwei Möglichkeiten, nämlich **Multipass, 1st pass**, der dem bisherigen Verhalten entspricht, und **Multipass, 1st pass (fast)**, der in DivX 6.4 neu eingeführt wurde. *Fast* arbeitet nach dem sel-

ben Prinzip wie Xvids 1st Pass oder MeGUIs Turbo-Modus für x264. Dabei werden einige Features deaktiviert, die im ersten Encodingdurchgang nicht gebraucht werden und ohne Auswirkungen auf die Qualität abgeschaltet werden können. Dadurch läuft der 1st Pass deutlich schneller. Nur wer zu viel Zeit hat, sollte deshalb nicht **Multipass, 1st pass (fast)** wählen.

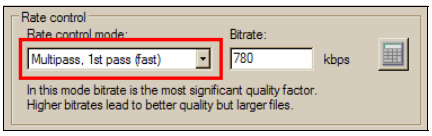

Abbildung 29: Einstellung für 1st Pass im DivX-Hauptdialog.

Die Bitrate im Feld rechts sollte das Encoding-Frontend automatisch ausfüllen. Genauso wie Xvids *Target Size* bezieht sich die Bitrate von DivX auf die reine Videospur, nicht auf die gesamte Dateigröße.

Damit können wir ins Register Codec weiterklicken (Abbildung 30). Bei **Codec Performance** müssen wir uns entscheiden, ob Geschwindigkeit oder Qualität wichtiger ist. **Balanced** ist vergleichsweise flott und bringt durchschnittliche Qualität. **Better**, **Extreme** und **Insane** bremsen jeweils kräftig ab, belohnen uns aber auch mit höherer Qualität. Der sinnvollste Kompromiss zwischen Geschwindigkeit und Qualität dürfte **Better** sein. **Extreme** und **Insane** haben ihre Namen aus gutem Grund.

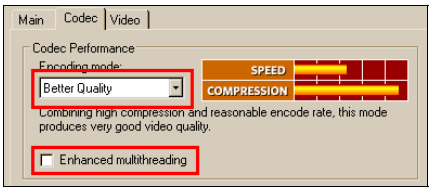

Abbildung 30: DivX-Performance-Optionen.

Enhanced multithreading ist interessant, wenn wir einen Computer mit mehreren CPUs unser eigen nennen (Hyperthreading, DualCore oder ein echtes Multiprozessor-System). Dann sollten wir die Option aktivieren, um die Encodinglast auf alle CPUs zu verteilen. Für Computer mit nur einem Prozessor dagegen setzen wir den Haken nicht.

Bidirectional coding in Abbildung 31 konfiguriert die B-Frames. **Off** dürfte sich selbst erklären ;-), **Adaptive Single Consecutive** entspricht dem verhalten älterer DivX-Versionen, d. h. mehrere B-Frames hintereinander werden nicht gesetzt. Maximal drei aufeinander folgende B-Frames erlaubt **Adaptive Multiple Consecutive**. B-Frames sollten wir auf jeden Fall aktivieren, und das dürfen ruhig die Multiple Consecutive sein.

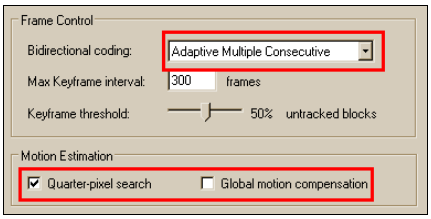

Abbildung 31: DivX-Optionen für Frametypen und Bewegungssuche.

Quarter-pixel search (QPel) und **Global motion compensation** (GMC) haben wir schon bei der Xvid-Konfiguration auf Seite 59 näher kennengelernt. Genau wie dort auch, empfiehlt sich, QPel eher zu aktivieren und GMC abgeschaltet zu lassen.

Damit haben wir das **Codec**-Register abgeschlossen und können zu **Video** wechseln. Die Einstellungen unter **Video Resolution** brauchen wir nicht beachten. Bei **Image Processing** interessiert uns nur die rechte Seite. Die anderen Einstellungen sollten auf den Standards wie im Bild belassen werden.

Unter **Quantization** lässt sich angeben, welche Quantisierungsmatrix DivX verwenden soll. Eine Spezialität von DivX ist **H.263 Optimized**, die auf einem modifizierten H.263-Algorithmus basiert, der zwar Geschwindigkeit kostet, aber höhere Qualität bringt. **MPEG-2** war in älteren DivX-Versionen problematisch und nicht empfehlenswert. Inzwischen gehören diese Probleme der Vergangenheit an.

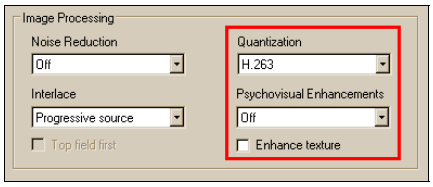

Abbildung 32: DivXs Matrix- und PVE-Auswahl.

Welche Matrix wir wählen sollten, hängt davon ab, wie stark der Film komprimiert wird. **MPEG-2** erzeugt ein etwas schärferes Bild auf Kosten der Kompression und eignet sich eher für die höhere Datenrate einer großzügigen Zielgröße. Die **H.263**-Varianten produzieren dagegen ein etwas weicheres Bild und erhöhen damit die Kompression ein wenig; geeignet eher für die niedrigen Datenraten von 1-CD-Encodings.

Psychovisual Enhancements (PVE) versucht, die Bildteile zu errechnen, die das menschliche Auge sowieso nicht wahrnehmen kann, und lässt diese weg. Die frei werdende Bitrate kann dann für andere Teile des Bilds verwendet werden. Für klassische Cartoons und manche Animes empfiehlt sich **Masking**. Für alles andere Material ist **Shaping** die bessere Wahl. Wer die niedrigere Geschwindigkeit verkraften kann, sollte PVE bei niedrigen Bitraten in der Regel aktivieren. **Enhance texture** ist ein Feature, das im Rahmen der PVE die Qualität rund um I-Frames verbessern soll.

Damit ist die Konfiguration des 1st Pass fast beendet. Nur ein Klick unten links auf **Advanced** bleibt uns noch (Abbildung 33). Hier können wir unter **Temporary Files Folder** angeben, wohin DivX seine temporären Dateien (z. B. die im 1st Pass angelegte Logdatei) ablegt. Der Ordner hat nichts damit zu tun, wohin der Film gespeichert wird. Weiter unten setzen wir die Haken bei **Do not prompt...** und **Disable the feedback window**. So überschreibt DivX im 2nd Pass ungefragt die Log-Datei (was er ja tun soll) und schaltet das Statistik-Fenster während des Encodings ab. **Rotate artwork** und **Check for new version** kann jeder ganz nach seinen Vorlieben setzen. Naja … das Statistik-Fenster eigentlich auch. :-)

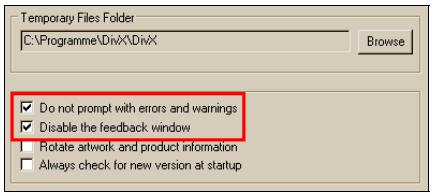

Abbildung 33: DivX, weitere Optionen.

Damit ist der 1st Pass komplett konfiguriert, und wir können den DivX-Dialog über **OK** verlassen.

Einstellungen für den Nth Pass

Den Großteil aller Einstellungen lassen wir für alle Durchgänge gleich. Nur auf der **Main**-Seite ändert sich ab dem 2nd Pass etwas. Hier stellen wir **Multipass, Nth pass** ein. Alle anderen Einstellungen – mit Ausnahme der **Codec Performance** – sollten für sämtliche Nth Passes exakt mit denen aus dem 1st Pass übereinstimmen. Eine veränderte **Codec Performance** ist dann sinnvoll, wenn wir in den ersten Durchgängen mit **Balanced** eher auf Geschwindigkeit setzen und dann im letzten Durchlauf mit **Extreme** oder **Insane** das Video auf Qualität trimmen.

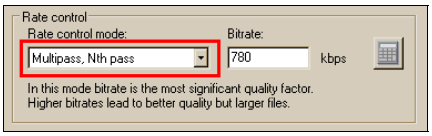

Abbildung 34: Einstellung für Nth Pass im DivX-Hauptdialog.

Da kommt auch die Frage auf, wie viele Durchgänge denn sinnvoll sind. Zwei oder drei Stück hat sich als vernünftiger Wert durchgesetzt. Mehr Passes bringen nur noch minimale Qualitätssteigerungen und dürften höchstens bei hoch komprimierten 1-CD-Encodings in Ausnahmefällen den Zeitaufwand wert sein.

A.4.1.3 x264 rev517 VfW-Konfiguration

X264 ist längst den Kinderschuhen entwachsen und schwer dabei, Xvid den Rang als beliebtester Open-Source-Codec abzulaufen. Und es ist tatsächlich abzusehen, dass für DVD-Backups die Tage der ASP-Codecs gezählt sind. Außerdem treibt x264 durch die starke Konzentration auf seinen Kommandozeilenencoder die Bewegung weg von VfW voran, was man nur uneingeschränkt gutheißen kann. Aus Gründen, die wenig mit dem Codec an sich zu tun haben, verwende ich trotzdem weiterhin fast ausschließlich Xvid. Deswegen stütze ich mich für die x264-Einstellungen in diesem Kapitel weitgehend auf Selurs Empfehlungen aus seinem »*Wissenswerten rund um x264*«.

Aktuelle Versionen enthalten nur noch den Kommandozeilen-Encoder. Wer mit dem VfW-Interface arbeiten will, muss x264 Revision 580 oder älter verwenden.

Einstellungen für den 1st Pass

Wie bei den anderen Codecs auch, interessieren uns die Single-Pass-Einstellungen nicht. Im **Bitrate**-Register wählen wir also **Multipass - First Pass (fast)** (Abbildung 35). Der Fast-Modus beschleunigt ähnlich wie bei Xvid den ersten Durchgang, ohne spürbar negativ auf die Qualität zu wirken. Deswegen taugt der langsame First Pass eher für paranoide Qualitätsfanatiker als für den täglichen Gebrauch.

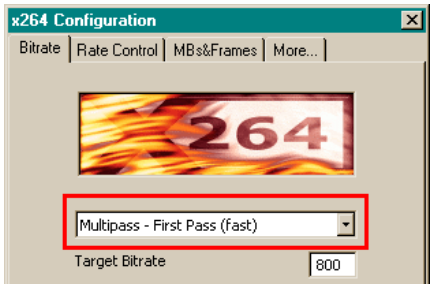

Abbildung 35: Einstellung für 1st Pass im x264-Hauptdialog.

Weiter unten muss **Update Statsfile** angehakt sein. Die Bitrate trägt das Encoding-Frontend automatisch ein. Entsprechen können wir gleich zum Register **Rate Control** wechseln (Abbildung 36).

B-frames reduction steuert die Qualität der B-Frames und ist deshalb natürlich nur wichtig, wenn wir überhaupt B-Frames verwenden. Der Wert gibt an, wie viel Prozent relative Bitrate ein B-Frame weniger bekommt als das vorangehende P-Frame. Je höher also der Wert, desto stärker werden B-Frames komprimiert. Gerade bei sehr niedrigen oder sehr hohen Datenraten bietet sich diese Option als Spielzeug an, um die B-Frames ein wenig kleiner oder ein wenig größer werden zu lassen. Wichtig ist dabei, nicht zu übertreiben, weil zu hohe Werte schnell Artefakte erzeugen und zu niedrige den Nutzen der B-Frames stark schmälern. Wenn wir uns eine sehr schwache Reduktion leisten können, sollten wir überlegen, ob wir die B-Frames nicht gleich ganz abschalten.

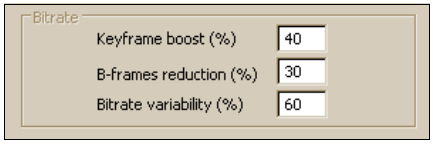

Abbildung 36: x264-Bitratenoptionen.

Bitrate variability gibt an, wie stark die Bitrate über den ganzen Film hinweg schwanken darf. **0** würde zu einem CBR-Encoding führen, **100** steht für den Constant-Quality-Modus. Vorsicht bei Änderungen, denn wie genau x264 die gewünschte Dateigröße treffen kann, hängt nicht zuletzt von dieser Einstellung ab. Die anderen Werte können wir bedenkenlos auf dem Standard belassen und zum MBs & Frames-Register wechseln.

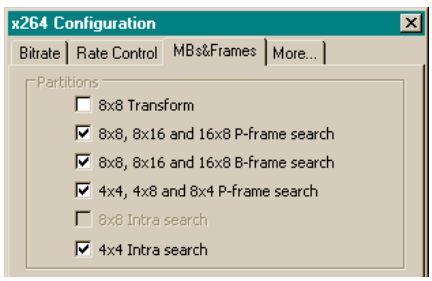

Abbildung 37: Makroblock-Partitionen in x264.

X264 erlaubt es, einen 16 × 16 Pixel großen Makroblock in kleinere Einheiten (Partitionen) zu zerlegen, um so die Bewegungssuche zu verfeinern. Mit den in Abbildung 37 gezeigten Schaltern können wir einstellen, welche Zerlegungen für I-, P- und B-Fra-

mes erlaubt sind. Die Partitionierung senkt zwar die Encoding-Geschwindigkeit deutlich, steigert aber gleichzeitig die Kompression genauso deutlich. Deshalb sollten wir dem Codec nur mit gutem Grund nicht alle Partitionsmöglichkeiten erlauben.

Eine Ausnahme stellt **8x8 Transform** dar, da dieses Feature zum AVC High Profile gehört, dass nicht jeder Decoder beherrscht. Diese Option sollten wir also nur dann anhaken, wenn wir uns den Fähigkeiten des Decoders sicher sind. Z. B. beherrscht ffdshow das High Profile, kann also auch mit **8x8 Transform** umgehen. Ist die Funktion deaktiviert, können wir gleichzeitig auch **8x8 Intra search** nicht verwenden.

Weiter unten im Register (Abbildung 38) kümmern wir uns um die B-Frames. **Max consecutive** legt die maximale Anzahl aufeinander folgender B-Frames fest. Eine **3** heißt, dass nicht mehr als drei B-Frames direkt hintereinander vorkommen dürfen. Ob diese Zahl auch erreicht wird, bestimmt x264 automatisch je nach Bedarf. **0** schaltet die B-Frames ganz ab. Ein Haken bei **Adaptive** verteilt proportional mehr B-Frames an langsame Szenen. Da sich diese gerade für den B-Frame-Einsatz anbieten, hat das auch Sinn. **Bias** geht in eine ähnliche Richtung, beeinflusst aber insgesamt, wie gerne der Codec B-Frames setzt. Die Funktion ist also mit Xvids *BVOP sensitivity* vergleichbar. Werte über **0** drängen x264 häufiger zum B-Frame-Einsatz – ohne allerdings das unter **Max consecutive** definierte Maximum zu überschreiten –, Werte unter **0** halten den Codec eher vom B-Frame ab. Genau wie bei Xvid können wir ruhigen Gewissens die **0** stehen lassen. **Use as reference** erlaubt es x264, ein B-Frame als Referenzframe zu verwenden. Das erhöht zwar die Anforderungen beim Decoding, sollte aber im Sinn der Qualität aktiviert werden, wenn es der verwendete Decoder unterstützt. Bei ffdshow z. B. ist das der Fall. **Bidirectional ME** verwendet eine erweiterte Bewegungssuche, die die Effizienz der B-Frames steigert. **Weighted bipredition** gewichtet Informationen aus den gefundenen Referenzen, was besonders Aus- und Überblendungen zugute kommt. Beide Optionen sollten wir normalerweise aktivieren. **Direct Mode** schließlich gibt an, nach welchem Verfahren B-Frames komprimiert werden. **Temporal** verwendet zur Komprimierung die zeitlichen Änderungen zwischen den Bildern, **Spatial** die räumlichen innerhalb des Frames. In der Regel sollten wir **Temporal** wählen, da es die höhere subjektive Qualität bietet.

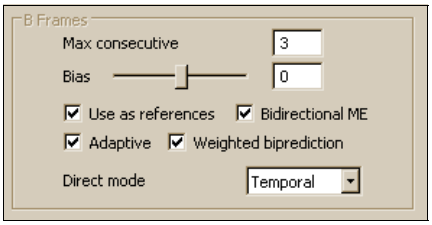

Abbildung 38:
x264 B-Frame-Einstellungen.

Damit bleiben die Einstellungen im **More**-Register übrig (Abbildung 39). **Partition decision** steuert die die Genauigkeit der Bewegungssuche innerhalb der Partitionen. Interessant zu wissen ist, dass unabhängig von der Einstellung der letzte Suchdurchlauf immer mit einem Viertel Pixel Genauigkeit geschieht. D. h. in altgewohnter ASP-Terminologie, QPel ist immer aktiv. Da uns ja die Qualität am Herzen liegt, sollten wir **5 (Max Quality)** unverändert lassen oder höchstens auf **4** senken. **6b (RDO on B-frames)** entspricht Xvids VHQ für B-Frames. Allerdings bedeutet diese Einstellung deutliche Leistungseinbußen, die die bessere Qualität oft nicht wert sind.

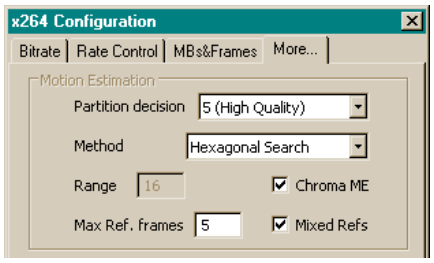

Abbildung 39: Einstellungen für die Bewegungssuche in x264.

Method bestimmt den Algorithmus, der zur Bewegungssuche verwendet wird. Von oben nach unten werden die Methoden immer genauer, aber auch immer langsamer. Der Standard **Hexagonal Search** ist ein sinnvoller Kompromiss zwischen Geschwindigkeit und Qualität und sollte immer beibehalten werden. Ungenauere Modi bieten sich wegen der niedrigeren Qualität nicht an, genauere Modi treiben den Rechenaufwand im Vergleich zum Qualitätsgewinn einfach zu weit in die Höhe. Deswegen ist auch **Range** kaum interessant, wo der Suchradius für den Modus **Uneven Multi-Hexagon** definiert werden kann.

Chroma ME entspricht Xvids *Chroma motion*, d. h. ein gesetzter Haken veranlasst x264, neben den Helligkeitsinformationen (Luminanz) auch die Farbinformationen (Chrominanz) zu Bewegungssuche zu verwenden. Meistens reichen zwar die Luminanzinfos aus, da aber die Option die Geschwindigkeit nicht zu sehr beeinträchtigt, sollte der Haken gesetzt bleiben.

Damit kommen wir zu **Max ref Frames**. In Xvid und DivX sind P-Frames auf genau ein Referenzbild beschränkt, und zwar auf das vorangehende Bild. x264 erlaubt auch mehrere Referenzframes. Wie viele, stellen wir mit **Max ref Frames** ein. Maximal dürfen es 16 sein. Allerdings steigert mehr als etwa 5 nur noch die Codierzeit, nicht mehr die Qualität. Die Einstellung wirkt sich übrigens auch auf B-Frames aus.

Mixed Refs ist eine neue Funktion, die es dem Codec erlaubt, Referenzbilder nicht nur für jeden Makroblock, sondern für jede Blockpartition einzeln auszuwählen. Tests auf Doom9.org zeigen, dass **Mixed Refs** zwar die Encodiergeschwindigkeit senkt, aber auch zu einem spürbaren Qualitätsanstieg führen kann. Deswegen sollten wir die Option wenn möglich aktivieren.

Wenden wir uns dem unteren Bereich des Fensters zu (Abbildung 40). **Sample AR** ist für anamorphe Zieldateien interessant. Hier tragen wir den passenden PAR aus Tabelle 4 (Seite 44) ein. Für klassische Encodings mit quadratischen Pixeln bleibt die Standardeinstellung **1:1** unverändert.

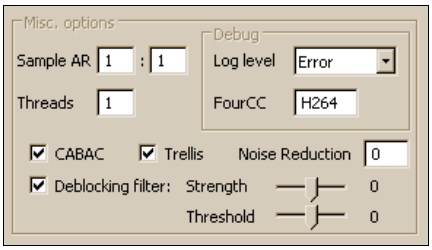

Abbildung 40: Weitere x264-Optionen.

Unter **Threads** können wir die Multiprozessor-Unterstützung einschalten. Einige Berechnungen werden dann auf den verschiedenen (virtuellen) Prozessoren parallel abgearbeitet, was die Geschwindigkeit erhöht. Wer einen Pentium 4 mit Hyperthreading sein Eigen nennt, darf hier **2** eintragen. Wer gar ein echtes Multiprozessorsystem besitzt, tippt seine Anzahl an Prozessoren ein, allerdings maximal **4**. Alle anderen können die Funktion nicht nutzen und müssen bei **1** bleiben.

Log level sollte auf **None** bleiben, da die Funktion hauptsächlich für die interne Fehlersuche wichtig ist. Ein normales Encoding bremst sie höchstens aus. Genauso können wir den **FourCC** unverändert lassen. Auch für **Noise Reduction** ist **0** der beste Wert, denn ums Bildrauschen kümmern wir uns – wenn nötig – besser mit einem AviSynth-Filter.

Eine AVC-Neuheit heißt **CABAC** (Context-Adaptive Binary Arithmetic Coding). Das ist eine Kompressionsmethode, die gerade bei hohen Datenraten die nötige Bitrate deutlich senken kann. Da CABAC verlustfrei arbeitet, leidet die Qualität nicht darunter. Allerdings sinkt die Geschwindigkeit. Wenn wir nicht gerade mit einem uralten Rechner ausgerüstet sind, sollte die Funktion aktiviert bleiben.

Trellis kennen wir von Xvid, und genau wie dort bringt die Option ein wenig Qualität ohne zu stark auf die Geschwindigkeit zu drücken. Deshalb sollten wir den Haken setzen.

Deblocking filter kennen wir von Decodern, die damit Blockartefakte im Bild zu übertünchen versuchen. x264 setzt einen solchen Filter schon beim Encodieren ein, und zwar nachdem ein Bild codiert wurde, aber bevor es als Referenz für das nächste Bild dient. Dadurch kann Bild 2 mit einer Referenz arbeiten, die weniger Artefakte enthält. Das tut der Qualität gut. Der Nachteil ist die sinkende Geschwindigkeit, um zwar sowohl beim Encoding als auch beim Decoding. Denn damit Bild 2 korrekt decodiert werden kann, muss das vorangehende Bild in gefilterter Form vorliegen. Das ähnelt einem Xvid-Encoding bei dem wir gezwungen sind, beim Anschauen den Deblocking-Filter zu aktivieren. **Strength** und **Threshold** steuern die Stärke des Filters. Beide Werte sollten wir aber als Einheit betrachten und auch gemeinsam verändern. Die Normaleinstellung **0** ist ein brauchbarer Standard. Negative Werte führen zu schwächerem Deblocking, posi-

tive zu stärkerem. Für hohe Datenraten bietet es sich an, das Deblocking etwas zu verringern (etwa auf **-2**), während für die niedrigen Raten von 1-CD-Encodings etwas höhere Werte angebracht sein können.

Die Einstellungen für den 1st Pass sind damit abgeschlossen. Bleibt noch kurz der zweite Durchgang zu konfigurieren.

Einstellungen für den Nth Pass

Wir stellen im **Bitrate**-Register um auf **Multipass - Nth pass** und aktivieren **Update stats file**. x264 unterstützt genau wie DivX mehr als zwei Durchgänge. Ob das tatsächlich eine Qualitätssteigerung ergibt, die den Mehraufwand rechtfertigt, müssen wir ausprobieren. Ähnlich wie bei DivX dürften mehr als drei Passes kaum Sinn haben. Ich selbst bin von Xvid verwöhnt und einfach zu faul für mehr als zwei Durchgänge. :-)

Um die Einstellung der Bitrate kümmert sich das Encoding-Frontend auch diesmal automatisch. Deshalb überprüfen wir noch, ob alle anderen Einstellungen mit dem 1st Pass übereinstimmen und sind auch schon fertig.

A.4.2 Kommandozeilen-Interfaces

Die Kommandozeilenencoder sind unabhängig von VfW und seinen Einschränkungen. Sie bringen keine grafische Oberfläche mit, sondern müssen an der Eingabeaufforderung mit den passenden Optionen aufgerufen werden. Normalerweise kümmert sich ein grafisches Encoding-Frontend darum, die nötige Kommandozeile zu erstellen. Da die Dialoge je nach Programm etwas unterschiedlich ausfallen, sehen wir uns die in den entsprechenden Frontend-Kapiteln näher an. (Ok, ok. Das ist im Moment Zukunftsmusik, da Gordian Knot noch das einzige besprochene Frontend ist.)

Die Kommandozeilen in den folgenden Kapiteln entsprechen der Konfiguration im entsprechenden VfW-Kapitel, erzeugen also die gleiche Qualität.

Im Moment sind die Kapitel nicht viel mehr als ein Listing der Kommandozeilen. Wer sich unsicher ist, sollte für wichtige Backups lieber auf altbewährte Methoden zurückgreifen, anstatt mit den Kommandozeilen zu hantieren, ohne wirklich zu wissen, was da passiert.

A.4.2.1 Xvid Kommandozeilen-Konfiguration

Xvid_Encraw ist kein komplett selbständiger Encoder, sondern nur ein Kommandozeilen-Interface für die xvidcore.dll, die die eigentliche Arbeit erledigt. Damit Encraw funktioniert, muss das übliche VfW-Paket von Xvid installiert sein, das die xvidcore.dll mitbringt.

Maßgeblich für die Kommandozeilen ist Xvid_Encraw in der Version vom 30. 04. 2006. Da sich Encraw mitten in der Entwicklung befindet, sind Abweichungen in anderen Versionen nicht unwahrscheinlich. Für aktuelle Details siehe den *Entwicklungsthread* im Doom9-Forum.

1st Pass

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -pass1 "<Statistikdatei>"
-vbvsize 1833216 -vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2
-bquant_ratio 163 -bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10
-zones 0,w,1,0
```

2nd Pass

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 {-o|-avi|-mkv}
"<Zieldatei>" -pass2 "<Statistikdatei>" -size <Zielgröße in KB>
-vbvsize 1833216 -vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2
-bquant_ratio 163 -bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10
-zones 0,w,1,0
```

Für anamorphe Zieldateien müssen wir eine weitere Option der Kommandozeile hinzufügen: `-par x:y`. X und Y stehen dabei für die entsprechenden Werte aus der PAR-Tabelle auf Seite 44. Für eine 16:9-PAL-DVD würden wir also `-par 16:11` angeben.

Die folgende Tabelle 6 erläutert die in den Kommandozeilen oben benutzten kursiven Platzhalter.

Quell-AVS-Datei	Vollständiger Pfad und Dateiname der als Quelle dienenden AviSynth-Skriptdatei. Muss in beiden Passes gleich lauten.
Statistikdatei	Vollständiger Pfad und Dateiname der Xvid-Statistikdatei, in der die Analysedaten aus dem 1st Pass gespeichert werden. Muss in beiden Passes gleich lauten.
{-o|-avi|-mkv}	Gibt den Typ der erzeugten Zieldatei an. Es darf nur genau eine der drei Optionen verwendet werden. -o erzeugt einen rohen MPEG-4-Videostream, -avi erzeugt eine AVI-Datei, -mkv erzeugt eine Matroska-Datei.
Zieldatei	Vollständiger Pfad und Dateiname der Zieldatei. Ist nur im 2nd Pass anzugeben. Die Dateiendung bestimmt *nicht* den Typ der Zieldatei (siehe oben)!
Zielgröße in KB	Größe der Videospur in Kilobyte, zu berechnen wie im Kapitel zur Zielauflösung auf Seite 55 beschrieben. Ist nur im 2nd Pass anzugeben.

Tabelle 6: Bedeutung der Platzhalter in den Xvid_Encraw-Kommandozeilen.

A.4.2.2 x264 Kommandozeilen-Konfiguration

Maßgeblich für die Kommandozeilen ist x264 Revision 517. Die unten benutzte Syntax ist zwar sehr stabil. Trotzdem können sich in anderen – vor allem neueren – Versionen einzelne Abweichungen ergeben.

1st Pass

```
x264.exe -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto --b-pyramid -w
--bime -A i4x4,p8x8,p4x4,b8x8 --no-psnr --progress -p 1 -B <Bitrate in
kbit> --stats "<Statistikdatei>" -o NUL "<Quell-AVS-Datei>"
```

Nth Pass

```
x264.exe -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto --b-pyramid -w
--bime -A i4x4,p8x8,p4x4,b8x8 --no-psnr --progress -p 3 -B <Bitrate in
kbit> --stats "<Statistikdatei>" -o "<Zieldatei>" "<Quell-AVS-Datei>"
```

Die folgende Tabelle 7 erläutert die in den Kommandozeilen oben benutzten kursiven Platzhalter.

Bitrate in kbit	Bitrate der Videospur in Kilobit pro Sekunde. Bezieht sich auf die Größe der Videospur, zu berechnen wie im Kapitel zur Zielauflösung auf Seite 55 beschrieben.
Statistikdatei	Vollständiger Pfad und Dateiname der x264-Statistikdatei, in der die Analysedaten aus dem 1st Pass gespeichert werden. Muss in allen Passes gleich lauten.
Quell-AVS-Datei	Vollständiger Pfad und Dateiname der als Quelle dienenden AviSynth-Skriptdatei. Muss in allen Passes gleich lauten.
Zieldatei	Vollständiger Pfad und Dateiname der Zieldatei. Ist nur im Nth Pass anzugeben. Die Dateiendung bestimmt, in welchem Format die Datei erstellt wird. MP4 oder MKV erzeugt eine MP4- bzw. Matroskadatei. Die Endung 264 erzeugt einen rohen MPEG-4-Videostream.

Tabelle 7: Bedeutung der Platzhalter in den x264-Kommandozeilen.

Für anamorphe Zieldateien müssen wir eine weitere Option der Kommandozeile hinzufügen: `--sar x:y`. X und Y stehen dabei für die entsprechenden Werte aus der PAR-Tabelle auf Seite 44. Für eine 16:9-PAL-DVD würden wir also `--sar 16:11` angeben.

Teil B
Praxiswissen

Einleitung

Dieser zweite Teil des Encodingwissens beschäftigt sich mit der Praxis. Also damit, wie wir aus einer DVD einen fertigen MPEG-4-Film encodieren. Einfach gesagt, ist das Praxiswissen die ausführliche Version des Schritt-für-Schritt-Kapitels A.1.5.

Eine Besonderheit sind die mit dem Symbol ▸ gekennzeichnete Verweise, die im Text verstreut auftauchen. Die Nummer eines solchen Verweis zeigt immer zum passenden Kapitel im Hintergrundwissen. So können wir uns die mehr theoretischen Infos auf (hoffentlich) bequeme Art und Weise häppchenweise aneignen. Allen Verweisen zu folgen, ist aber keine Garantie dafür, sämtliche Kapitel des Hintergrundwissens zu Gesicht zu bekommen!

B.1 Vorarbeiten

DVD-Backups erfordern eine ganze Reihe verschiedener Tools, mit denen wir die einzelnen Schritte durchführen können. Deshalb werfen wir in diesem Kapitel zuerst einen ausführlichen Blick auf das nötige Softwarepaket.

Anschließend betrachten wir die Arbeiten, die dem Encoding vorausgehen. Um einen Film bearbeiten zu können, müssen wir ihn zuerst von der DVD auf die Festplatte kopieren und in seine Bestandteile zerlegen. Verantwortlich dafür sind hauptsächlich vStrip und DGIndex.

B.1.1 Die nötige Software

Das Softwarepaket, das wir für ein DVD-Backup benötigen, ist umfangreich. Die folgende Liste zählt sämtliche Tools auf, die wir im Encodingwissen ansprechen. Nicht alle davon sind immer unbedingt nötig. Wenn wir z. B. DivX nicht verwenden wollen, brauchen wir den Codec natürlich auch nicht herunterzuladen und zu installieren.

Vorarbeiten

vStrip

DVD-Ripper. Kopiert die DVD auf die Festplatte. Alternativen sind der nach wie vor beliebte DVD Decrypter, DVDFab Decrypter oder der alte und inzwischen etwas angestaubte Smartripper. Diese Programme sind komfortabler und in mancher Hinsicht besser als vStrip, können aber auch dazu dienen, den CSS-Kopierschutz zu umgehen. Deshalb verzichte ich wegen der verbraucherfeindlichen Rechtslage lieber auf Links.
http://www.free-codecs.com/download/vStrip.htm

Chapter-X-tractor

Schreibt Kapitelinformationen der DVD in eine Textdatei.
http://www.divx-digest.com/software/chapterxtractor.html

DGMPGDec

Ein Paket von Tools, um VOBs und andere MPEG-2-Videos zu verarbeiten. Enthält DGIndex, das wir zum Indexieren der VOBs und zum Demuxen der Audiospuren verwenden, und DGDecode, das die Videospur der VOBs für AviSynth zugänglich macht.
http://neuron2.net/dgmpgdec/dgmpgdec.html

Audio und Untertitel

BeSweet

Audio-Transcoding-Tool. Ausführliche Informationen zur Installation und zu den benötigten Bibliotheken finden wir im Kapitel B.2.1 BeSweet einrichten.
http://besweet.notrace.dk

BeLight

Grafische Oberfläche für BeSweet.
http://corecodec.org/projects/belight

VobSub

Tool, um die Vobsub-Untertitel der DVD aus den VOBs herauszuziehen. Eine alternative Oberfläche ist VSRip.
http://www.afterdawn.com/software/video_software/subtitle_tools/vobsub.cfm

SubRip

Tool, um Vobsubs in Text umzuwandeln.
http://zuggy.wz.cz

Video-Encoding

AviSynth

Frameserver. Stellt dem Encoder das Video bereit. Verantwortlich für sämtliche Filter.
http://www.avisynth.org

AviSynth-Plugins

Liste und Links für viele AviSynth-Plugins: Convolution3D, Decomb, Deen, FillMargins, FluxSmooth, SimpleResize, TomsMoComp usw.
http://www.avisynth.org/warpenterprises

Xvid Video-Codec

Binaries von Koepi. Enthält den VfW-Codec und den DirectShow-Decoder. Den Quellcode gibt es auf *Xvid.org*.
http://koepi.org

Xvid_Encraw

Xvid-Kommandozeilenencoder von squid_80. Zum aktuellen Stand den *Entwicklungsthread* im Doom9-Forum verfolgen.
http://members.optusnet.com.au/squid_80/xvid_encraw.zip

DivX Video-Codec

Die Pro-Version mit allen Funktionen gibt es als 15-Tage-Trial.
http://www.divx.com/divx

x264 Video-Encoder

Sharktooths Builds von x264. Alternativ können wir die Builds von z. B. Bob0r auf *x264.nl* verwenden. Achtung: VfW ist nur bis r580 verfügbar. Den Quellcode gibt es auf *videolan.org*.
http://forum.doom9.org/showthread.php?threadid=89979

VirtualDubMod

Video-Editing-Tool. Wird von Gordian Knot für das eigentliche Encoding verwendet.
http://sourceforge.net/projects/virtualdubmod

Encoding-Frontends

Gordian Knot

Die Kommandozentrale fürs Encoding. Bisher einziges besprochenes Frontend. Das GKnot RipPack enthält neben Gordian Knot selbst auch VirtualDubMod, BeSweet, DGMPEGDec, ChapterXtractor, vStrip, Vobsub und AviSynth inkl. einiger Plugins.
http://sourceforge.net/projects/gordianknot

Akapumas Gordian-Knot-Personalisierer (agkp)

Aufsatz für Gordian Knot, der Unterstützung für x264.exe und MKVMerge bietet.
http://forum.gleitz.info/showthread.php?t=22860

Alternative Frontends

MeGUI, StaxRip, AutoGK, RealAnime

Muxing und Splitting

MKVToolnix

Toolpaket rund um Matroska. Nötig für AAC in MKV, empfehlenswert für sämtliches Matroska-Muxing.
http://www.bunkus.org/videotools/mkvtoolnix/index.html

AVI-Mux GUI

Großartiges Muxing-Tool für AVI. Nötig für AAC in AVI. Unterstützt auch Matroska.
http://www.alexander-noe.com/video/amg

MP4Box

Muxing-Tool für MP4 und Teil des GPAC-Projekts. Noch verwenden wir im Encodingwissen kein MP4, ich finde es aber wichtig, zu erwähnen. Wer sich mit MP4-Box beschäftigen will, unbedingt diesen *Doom9-Thread* lesen: »GPAC's MP4Box Information, Requests and Discussions«.
http://ffdshow.faireal.net/mirror/gpac/dev

B.1.2 DVD-Ripping

Recht und Gesetz

Viele DVDs besitzen einen digitalen Kopierschutz namens CSS, d. h. die Videodaten liegen verschlüsselt auf der Scheibe. Zwar lässt sich dieser Schutz lächerlich einfach knacken, allerdings ist das (nicht nur) in Deutschland illegal. Aus einem ähnlichen Grund musste LIGHTNING UK! die Entwicklung des viel geliebten DVD Decrypter einstellen.

Stimmt schon, wir haben das Recht auf die Privatkopie und, ja, wir zahlen für genau dieses Recht eine Abgabe auf jeden gekauften Rohling. Und, nein, wir dürfen unser gutes Recht dank aktueller Kopierschutzgesetze oft nicht ausüben.

Ob das fair ist, kann man bezweifeln. An der Gesetzeslage ändern solche Zweifel jedoch nichts. Deshalb bezieht sich die folgende Anleitung ausdrücklich auf die im Software-Kapitel angegebene CSS-freie Version von vStrip, denn CSS-lose DVDs dürfen natürlich nach wie vor digital auf die Platte geschaufelt werden. Verzichtet bitte darauf, mir irgendwelche Anfragen zum Rippen von kopiergeschützten DVDs zu schicken, denn um mir keine rechtlichen Probleme einzuhandeln, werde ich die nicht beantworten.

Wichtige Grundlagen

Eine Video-DVD ▸ A.2.2 enthält meist einen leeren Ordner namens **AUDIO_TS** und einen wichtigen Ordner **VIDEO_TS**, in dem der komplette Inhalt der DVD liegt. In diesem Ordner interessieren uns zwei Dateitypen.

- VOB. Das ist das Containerformat der DVD und enthält Video, Audio und Untertitel.
- IFO. Das sind Steuer- und Informationsdateien, die uns das Ripping und den Umgang mit Untertiteln erleichtern.

Die Dateien sind immer nach dem selben Schema benannt, und zwar **vts_xx_y.zzz**. Dabei steht **xx** für eine zweistellige Nummer, die eine Gruppe von VOBs identifiziert. So haben zum Beispiel alle zum Hauptfilm gehörigen VOBs das gleiche **xx**. Innerhalb einer Dateigruppe wird das **y** von **0** aus hochgezählt, und **zzz** bezeichnet die Dateiendung (VOB oder IFO). So könnte z. B. der Hauptfilm in drei VOBs verpackt sein, die alle die Nummer **05** haben, das wären dann **vts_05_0.vob**, **vts_05_1.vob** und **vts_05_2.vob**. Dazu gehört die passende **vts_05_0.ifo**. Mehr brauchen wir nicht wissen, um immer die richtigen Dateien auszuwählen.

B.1.2.1 Ripping des Films

Wir legen die DVD ein und starten vStrip GUI. Sollten wir uns dann nicht schon im Register **Input** befinden, wechseln wir dorthin und klicken auf den Button

Add, um die VOB-Dateien von der DVD ▸A.2.2 auszuwählen. Da wir den kompletten Hauptfilm rippen wollen, geht das recht einfach.

Im **Öffnen**-Dialog (Abbildung 41) wechseln wir zum DVD-Laufwerk in den Ordner **VIDEO_TS**. Zur besseren Übersicht empfiehlt es sich, rechts oben in die Detailansicht umzuschalten. Dann suchen wir uns die längste zusammenhängende Sequenz VOB-Dateien heraus, bei denen also die mittlere Nummer gleich ist. Im Bild oben sind das alle VOBs mit der **02** in der Mitte, also **vts_02_0.vob** bis **vts_02_4.vob**. Allerdings enthält die Datei mit der **0** noch keine Daten des Films, weshalb wir nur die VOBs ab **1** auswählen und **Öffnen** klicken. Im oberen Fenster von vStrip erscheinen jetzt die ausgewählten Dateien, und wir können ins **IFO**-Register wechseln (Abbildung 42).

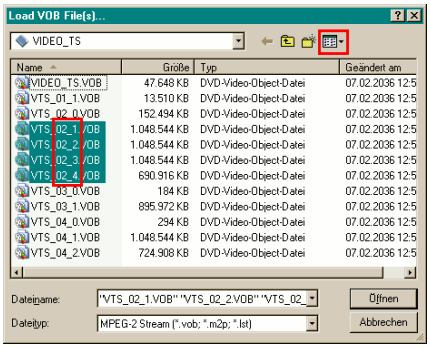

Abbildung 41: VOB-Auswahl mit vStrip.

Hier laden wir über den Button (2) die IFO-Datei des Hauptfilms. Deren Dateiname ist genauso aufgebaut wie bei den VOBs. Im Beispiel von oben passt also zu **vts_02_y.vob** die **vts_02_0.ifo**. Wichtig ist die gleiche mittlere Nummer. Wer später mit SubRip oder VobSub die Untertitel weiterverarbeiten will, sollte die IFO auch auf die Platte kopieren. Da an der Datei nichts geändert werden muss, können wir sie schlicht und einfach per Dateimanager kopieren.

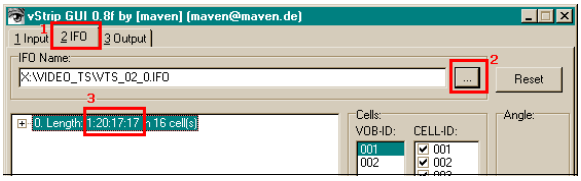

Abbildung 42: IFO-Datei in vStrip laden.

Ein Klick auf **Öffnen** bringt uns zurück zu vStrip. Im großen Fenster stehen nun alle PGCs, die die gewählte IFO enthält. Im Idealfall ist das sowieso nur eine einzige. Ansonsten wählen wir diejenige aus, deren Länge (3) mit der Länge des Hauptfilms übereinstimmt.

Abbildung 43: vStrip Stream-Liste.

Unten im Fenster (Abbildung 43) zeigt uns vStrip, welche Audio- und Untertitelstreams vorhanden sind. Wenn wir einige davon weglassen, also gar nicht auf die Festplatte kopieren, wollen, müssen wir uns die IDs der unnötigen Streams merken. Relevant ist jeweils die zweite **0x**-Nummer, für die Audiospuren im Bild also **0x80** bzw. **0x81**. Damit geht es weiter ins **Output**-Register (Abbildung 44).

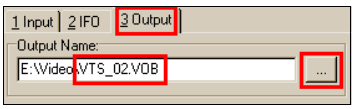

Abbildung 44: vStrip: Name der Zieldateien.

Hier wählen wir unter **Output name**, wohin auf der Festplatte die VOBs kopiert werden sollen. Um später nicht möglicherweise Probleme zu bekommen, sollten wir die Dateien so benennen wie auch auf der DVD, nur ohne den Teil **_y** am Ende des Dateinamens. Im Beispiel von oben wäre der passende Name also **vts_02.vob**. Das fehlende **_y** fügt vStrip beim Ripping automatisch an.

Die **Output Options** in der Mitte sollten immer ohne Haken bleiben, und weiter unten bei **Split File Size** (Abbildung 45) bestimmen wir, wie groß eine einzelne VOB werden darf. Das Maximum von 1024 MB wie auf der DVD einzuhalten, ist nicht so wesentlich. Nur über 4096 MB sollten wir nicht gehen, da manche Tools dann Probleme machen könnten. Außerdem kann das FAT32-Dateisystem keine größeren Dateien verwalten. Wer NTFS benutzt, hat diese Einschränkung nicht. **Split** sollte immer auf **Never** gesetzt bleiben, es sei denn vStrip kommt dann beim Ripping ins Straucheln.

Abbildung 45: Maximale VOB-Größe in vStrip.

Bleibt der rechte Bereich des Fensters, in dem wir auswählen können, welche Streams überhaupt gerippt werden sollen. Wenn wir sowieso das Video, alle Audiospuren und alle Untertitel auf der Platte haben wollen, brauchen wir hier nichts einzustellen. Ansonsten aktivieren wir **SubStreams** und entfernen bei den unerwünschten Spuren den Haken davor. Welche Nummer der unerwünschte Stream hat, wissen wir ja aus dem **IFO**-Register.

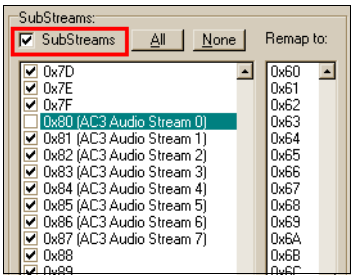

Abbildung 46: Abwählen von Spuren in vStrip.

Im Bild haben wir so die deutsche Tonspur abgewählt und erhalten auf der Platte dann VOBs, die nur noch den englischen Ton enthalten. Genauso können wir auch Untertitel abwählen, was aber kaum sinnvoll ist, da die nur wenig Platz belegen und sowieso noch weiterverarbeitet werden müssen.

Manche Software-DVD-Player können bei gerippten VOBs nicht mehr zwischen den Audiospuren umschalten, sondern spielen ausschließlich die Spur mit der ID **0x80** ab. Wer also den Film von den VOBs auf der Platte anschauen will, sollte darauf achten, dass die richtige Sprache auf **0x80** liegt. Dafür ist das **Remap to**-Feld zuständig.

Mit einem Klick auf **Run** unten rechts starten wir das Ripping und warten, bis vStrip die VOBs auf die Platte geschaufelt hat. Je nach DVD-Laufwerk und Länge des Films dauert das Rippen zwischen einigen Minuten und einer halben Stunde. Dann haben wir im Normalfall zwischen 4 und 7 GByte in einer oder mehreren VOBs auf der Platte.

B.1.2.2 Ripping der Kapitelliste

Leider kann vStrip die Kapitelinfos ▸A.1.4 der DVD nicht auslesen, weshalb das Chapter-X-tractor übernehmen muss. Im Gegensatz zu vStrip ist das Tool allerdings äußerst simpel zu bedienen.

Wir starten Chapter-X-tractor und laden über den Button **Open IFO** unten links die IFO-Datei des Hauptfilms entweder aus aus dem **VIDEO_TS**-Ordner der DVD oder auch von der Platte, wenn wir die IFO kopiert haben. Im Beispiel aus dem vStrip-Kapitel wäre das also die bekannte **vts_02_0.ifo**, je nach DVD auch mit einer anderen Nummer als **02**.

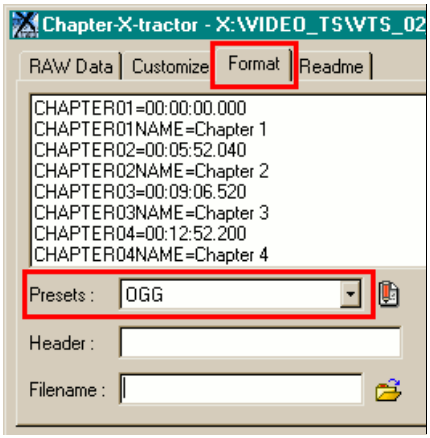

Abbildung 47:
Formatwahl in Chapter-X-tractor.

Im großen leeren Fenster erscheinen jetzt die Informationen aus der IFO. Sollte eine Meldung kommen *»Warning! Last chapter length is less than 5 s«*, hat die DVD ganz am Ende ein nur Sekundenbruchteile kurzes und normalerweise nutzloses Kapitel, das Chapter-X-tractor nicht übernimmt. Mit der Option **Last chapter bug fix** können wir dieses Verhalten auch ändern. Im Register **Format** () wählen wir dann unter **Presets** das Format, in dem die Kapitelinformationen gespeichert werden sollen. Für unsere Zwecke am besten geeignet ist **OGG**, da damit alle Tools umgehen können, die wir noch verwenden werden.

Ein Klick unten links auf **Save data** speichert die Kapitelinfos in einer Textdatei ab, und wir sind fertig. Wer will kann vorher noch in den jeweiligen **CHAPTER*xx*NAME**-Zeilen die Standardbezeichnungen gegen die echten Kapitelnamen austauschen. Natürlich lässt sich die erzeugte Datei auch später mit einem beliebigen Texteditor anpassen.

B.1.3 Indexieren der VOBs mit DGIndex

DGIndex verwenden wir, um die Audiospuren aus den VOBs herauszuholen (demuxen) und in separate Dateien zu speichern, und um einen Index der Videospur anzulegen, der dann von AviSynth weiterverwendet werden kann.

Der ursprüngliche Programmierer von DGIndex heißt jackei und hat das Pro-

gramm damals DVD2AVI genannt. Nachdem die Entwicklung einige Zeit stillstand, hat sich Donald »neuron2« Graft dem Projekt angenommen. DVD2AVI heißt in der weiterentwickelten Variante nun DGIndex und ist Teil des DGMPGDec-Pakets.

Wir starten DGIndex und gelangen über **File / Open** in den Öffnen-Dialog. Dort wählen wir die vorhin gerippten VOBs aus. Beim Klick auf Öffnen erscheint ein weiteres Fenster, in dem wir noch zusätzliche VOBs laden könnten. Einfach mit **Ok** bestätigen.

Die Einstellungen im **Video**-Menü sollten wie in Abbildung 48 aussehen. Den **iDCT Algorithm** wählt DGIndex automatisch. Im **Audio**-Menü regeln wir das Herausziehen der Audiospuren aus den VOBs (Abbildung 49). **Output Method** bestimmt, auf welche Weise die Audiospuren extrahiert werden. **Disable** extrahiert gar nichts, **Demux Tracks** extrahiert nur die unter **Track Number** ausgewählten Spuren und **Demux all Tracks** extrahiert alle vorhandenen Spuren. Mit **Decode AC3 Track to Wav** könnten wir den Sound gleich in Wave umwandeln. Das sollten wir aber besser BeSweet überlassen. Deshalb interessiert uns auch der Rest des Audio-Menüs nicht.

Über **F4** legen wir nun eine D2V-Projektdatei an. Solange DGIndex arbeitet, öffnet sich das Statusfenster (Abbildung 50). Sobald dort ganz unten **FINISH** erscheint, ist der Speichervorgang abgeschlossen und wir haben die extrahierten Audiodateien und die D2V-Datei auf der Platte. Die ersten drei rot umrandeten Werte merken wir uns. Die werden später in Gordian Knot wichtig, um Auflösung und Seitenverhältnis des Bildes richtig einzustellen.

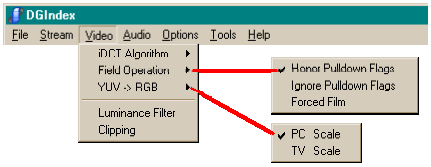

Abbildung 48: Einstellungen im DGIndex-Video-Menü.

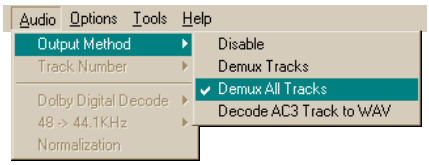

Abbildung 49: Demuxing sämtlicher Audiospuren.

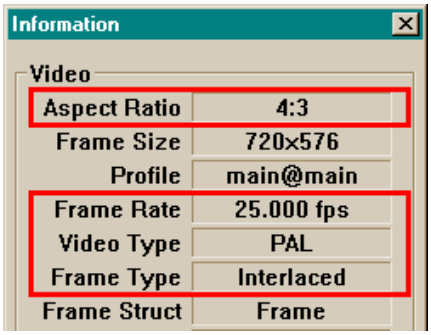

Abbildung 50: DGIndex-Statusfenster.

Wenn bei **Frame Type** zum Schluss **Interlaced** erscheint, ist das Video möglicherweise interlaced codiert. Dann sollten wir uns Szenen mit viel horizontaler Bewegung suchen und prüfen, ob dort Kammartefakte wie im Screenshot 51 auf Seite 88 auftreten. Manchmal wird der Effekt auch nur bei Szenenwechseln sichtbar, wenn sich altes und neues Bild ein Frame lang überlagern.

Hat das Video solche Effekte, müssen wir es später deinterlacen. Für NTSC-Material kommt eher ein Inverse-Telecide-Prozess in Frage. DGIndex hat seine Arbeit jetzt auf jeden Fall erledigt und kann geschlossen werden.

Interlacing taucht in einer unübersehbaren Vielzahl von Varianten auf. Ich bin auf dem Gebiet alles andere als ein Experte. Erwartet in diesem Guide also keine vollständigen und immer komplett richtigen Infos dazu.

Abbildung 51: Bild mit Interlacing-Artefakten. Besonders deutlich um die Kinnpartie des Manns links im Bild.

B.2 Audio-Transcoding

Die Tonspuren frisch von der DVD sind oft zu groß, um unverändert ins Encoding übernommen zu werden. Deswegen müssen wir sie oft in ein weniger platzintensives Format umwandeln. Natürlich ist das immer mit einem gewissen Qualitätsverlust verbunden, der allerdings im Vergleich zur eingesparten Dateigröße moderat ausfällt.

Das Tool der Wahl zum Audio-Transcoding ist BeSweet mit seiner grafischen Oberfläche BeLight. Auch die Encoding-Frontends bringen Möglichkeiten zum Audio-Transcoding mit, die allerdings nicht den vollen Funktionsumfang von BeSweet abdecken. Solange wir nur die Standardfunktionen benötigen, spricht nichts dagegen, das Transcoding vom Frontend durchführen zu lassen. Wenn wir allerdings die Möglichkeiten von BeSweet voll ausschöpfen wollen, führt an BeLight oder der Kommandozeile kein Weg vorbei.

B.2.1 BeSweet einrichten

So einfach es anfangs auch erscheint, die BeSweet-Installation ist durchaus mit einigen Tücken behaftet, die wir in diesem Kapitel ansprechen wollen.

Komponenten

Um BeSweet vollständig zu installieren, benötigen wir folgende Bestandteile:

BeSweet
BeSweet v1.5b31 von der BeSweet-Homepage. Betaversionen von BeSweet benötigen möglicherweise auch die neuste stabile Version, weil manchmal im Download der Betaversion nicht alle nötigen Dateien enthalten sind. Aktuell stellt das allerdings kein Problem dar.
http://besweet.notrace.dk

VOBInput.dll

Falls wir Sound direkt aus den VOB-Dateien transcodieren wollen (ist im Encodingwissen nicht der Fall). Die Datei erhalten wir auch von der BeSweet-Homepage.

Lame_enc.dll

Falls wir MP3s erzeugen wollen. Beste Anlaufstelle dafür ist RareWares. Ich bevorzuge die aktuelle, von Hydrogenaudio empfohlene, Version »LAME 3.97«.
http://www.rarewares.org/mp3.html

Libvorbis.dll

Falls wir nach Vorbis transcodieren wollen. Die Downloads gibt es wieder auf RareWares. Aktuell empfehle ich »libvorbis.dll using aoTuVb4.51«, auf jeden Fall aber eines der libvorbis.dll-Pakete. Den Download müssen wir je nach vorhandenem Prozessortyp wählen. Dabei ist der P4-Download nicht nur für den Pentium 4 interessant, sondern für alle SSE2-fähigen Prozessoren. Dazu gehören auch z. B. der Pentium M und aktuelle Athlons. Außerdem benötigen wir den letzten Download in der Liste, die *libmmd.dll*. Wer später die Vorbis-Dll aktualisiert, sollte immer nachsehen, ob auch eine neuere *libmmd.dll* existiert, denn beide Dateien müssen zueinander kompatibel sein.
http://www.rarewares.org/ogg.html

2Lame

MP2-Encoding mit 2Lame benötigt die nicht im BeSweet-Paket enthaltene *tooLame.dll.* BeSweet liefert nur den mp2enc MP2-Encoder mit. Im Encodingwissen sprechen wir MP2 nicht an.

Nero

AAC-Encoding mit Nero setzt eine installierte Version von Nero voraus, die den AAC-Encoder enthält. Nero-Express und die meisten OEM-Versionen enthalten den nicht. Da ich nur Nero 6 besitze, beziehen sich alle Aussagen zu Nero auch ausdrücklich nur auf Version 6 – insbesondere *nicht* auf die neue Version 7, die einige Veränderungen aufweist.

Winamp

AAC-Encoding mit Winamp benötigt zum einen die *WA_aac.dll,* die man auf der BeLight-Downloadseite im Abschnitt *Dimzon BeSweet Plugins* findet. Außerdem benötigen wir Winamp 5.21 oder höher. Der *Free-Full*-Download ist die beste Wahl, wenn wir Winamp nicht kaufen wollen.
http://corecodec.org/frs/?group_id=45
http://www.winamp.com/player/free.php

BeLight

Als grafische Oberfläche. Im Moment empfiehlt sich das aktuelle Daily Build.
http://corecodec.org/frs/?group_id=45

Installation

Haben wir alle nötige Pakete, entpacken wir zuerst das BeSweet-Archiv und das BeLight-Archiv in einen Ordner, z. B. *C:\Programme\BeSweet.* Außerdem installieren wir Nero bzw. Winamp, wenn wir deren AAC-Encoder verwenden wollen.

In den BeSweet-Ordner kopieren wir folgende Dateien aus den verschiedenen heruntergeladenen Archiven:

- **VOBInput.dll** und **toolame.dll,** wenn wir sie benötigen,
- **lame_enc.dll** aus dem Lame-Archiv,
- **libvorbis.dll** und **libmmd.dll** aus den beiden Vorbis-Downloads.

Um Nero verwenden zu können, kopieren wir folgende Dateien aus *C:\Programme\Gemeinsame Dateien\Ahead\AudioPlugins* in den BeSweet-Ordner:

- **Aac.dll**
- **aacenc32.dll**

Für ältere Nero-Versionen (vor 6.3, wenn ich mich richtig erinnere) kopieren wir zusätzlich aus *C:\Programme\Gemeinsame Dateien\Ahead\Lib* die **NeroIPP.dll.** Winamp-AAC-Encoding benötigt zwei Dateien im BeSweet-Ordner:

- **WA_aac.dll** aus dem Archiv von der BeLight-Seite,
- **enc_aacplus.dll** aus dem *Plugins*-Unterordner des Winamp-Ordners.

Damit sind BeSweet und BeLight voll funktionsfähig, zumindest soweit wir es fürs Encodingwissen brauchen. Der Installer der letzten stabilen BeLight-Version bringt zusätzlich vollständige Unterstützung für sämtliche Dimzon-Plugins mit. Das würde uns zwei weitere AAC-Encoder und einen Speex-Encoder bringen, die für Filme aber weniger wichtig sind.

B.2.2 Audio decodieren und bearbeiten

Wir starten BeLight und öffnen per Drag & Drop auf das leere Listenfeld oder über **File / Open** die Quelldatei. Über **File / Save** stellen wir die Zieldatei ein. Je nach gewähltem Encoder passt BeLight später automatisch die Dateiendung an. Wir müssen uns also in diesem Dialog noch nicht endgültig für ein Zielformat entscheiden.

Falls wir die Originaltonspur der DVD so wie sie ist übernehmen, können wir den kompletten BeSweet-Prozess überspringen. Die Tonspur muss dann später nur unverändert in den endgültigen Container gemuxt werden.

Nun betrachten wir die linke Seite des BeLight-Fensters. Der erste Eintrag (Abbildung 52) gehört zu Azid. Azid ist dafür zuständig, die AC3-Tonspur ▸ A.2.1 zu decodieren und eventuell 6-Kanal auf 2-Kanal (Stereo) umzurechnen.

Die Bezeichnungen der Tonformate sind etwas irreführend. 6-Kanal und 5.1 bezeichnet das selbe. Nur zählt man einmal alle Kanäle (vorne links und rechts, hinten links und rechts, Center und LFE), wogegen bei der 5.1-Schreibweise der Basskanal (LFE) extra dargestellt wird.

Mit **Dynamic Compression** stellen wir die gewünschte Dynamikkompression für die AC3 ein. Als *Dynamik* des Sounds bezeichnet man die Unterschiede zwischen lauten und leisen Passagen. Je höher die Dynamik, desto höher ist der Lautstärken-Unterschied zwischen leisen und lauten Passagen. Die Audio-Spur eines Films hat von Natur aus eine recht hohe Dynamik. Das wird ganz klar, wenn wir uns den Showdown mit knatternden Maschinengewehren und die geflüsterte Liebesszene im Vergleich vorstellen.

Beim Downmix auf Stereo würden wir ohne Kompression eine nervig leise

Zieldatei bekommen, deshalb gleichen wir mit der **Dynamic Compression** die Unterschiede in der Lautstärke an. D. h. im Extremfall hört sich das Flüstern genauso laut an wie die Maschinengewehr-Salve. Mit der Einstellung **normal** treiben wir es lange nicht so weit und erreichen eine für Stereo angemessene Dynamik-Kompression.

Abbildung 52: BeLight-Option für Dynamikkompression.

Wollen wir den 6-Kanal-Ton beibehalten, sollten wir eine geringere **Dynamic Compression** als für Stereo nehmen, um den Sound nicht unnötig zu verfälschen. Da mir die Dynamik der AC3 manchmal einfach zu hoch ist, bevorzuge ich eine leichte (**light**) Kompression. Man könnte sie auch ganz deaktivieren (Haken wegklicken).

Wer die Dynamikkompression manuell konfigurieren möchte, nimmt **Boost** (Abbildung 53) anstatt Azids **Dynamic Compression**. Achtung! Nicht beides zusammen verwenden! **Boost** hat es in sich. Wir können damit das letzte bisschen Dynamik aus der Tonspur herauskomprimieren, was in meinen Ohren schon nicht besonders gut klingt. Dazu kommt die Gefahr, durch zu viel Kompression Störgeräusche im Sound zu erzeugen. Deshalb sollten wir uns gut überlegen, ob der Einsatz von Boost wirklich nötig ist. Der Weg über Azids **Dynamic Compression** ist meistens der bessere. Ich habe **Boost** noch nie verwendet und das bis jetzt auch noch nicht bereut.

Abbildung 53: Boost als Ersatz für Dynamic Compression.

Wenn's denn sein soll, LigH schlägt im *Doom9/Gleitz-Forum* Folgendes vor: **Boost Mode LigH** hat Sinn, wenn der ursprüngliche Ton schon Stereo ist. Für 6-Kanal-Quellen ist dagegen **Dg** sinnvoller. **Tera** eignet sich eher für Experimente.

Mit dem **SSRC**-Abschnitt weiter oben können wir die Abtastfrequenz des Tons von den üblichen 48 kHz in einen anderen Wert umrechnen. Da die Soundkarten inzwischen ausgestorben sind, die mit 48 kHz nicht umgehen können, dürfen wir den Punkt ignorieren.

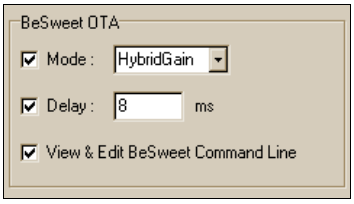

Abbildung 54: Normalisierung und Delay.

Damit weiter in den **OTA**-Abschnitt (Abbildung 54). Wir müssen den Sound noch normalisieren, d. h. die Lautstärke auf 100 % oder knapp darunter anheben. Durch die Dynamikkompression von oben haben wir zwar schon die Lautstärken-Unterschiede innerhalb der Tonspur angeglichen, insgesamt ist sie aber immer noch viel zu leise. Das Anheben erledigt **Mode** im **OTA**-Abschnitt.

Haben wir uns für AAC ▸ A.2.1 als Zielformat entschieden, heißt die richtige Ein-

stellung **PreGain**, für alle anderen Formate **HybridGain**. Der Unterschied kommt daher, dass **HybridGain** gleich am Anfang des Transcodings je nach Quelldatei einen festen Wert auf die Lautstärke aufschlägt und die Differenz zu 100 % in einem PostGain-Tag in der Datei speichert. Der Audiodecoder muss dann später beim Abspielen des Films diesen Tag auslesen und die Lautstärke noch entsprechend erhöhen. Leider gibt es noch keinen AAC-Decoder, der mit PostGain-Tags umgehen kann. Deshalb muss für dieses Format BeSweet die komplette Arbeit erledigen.

Damit kommen wir zum **Delay**. Das ist der Wert, um den die Audiospur zum Video verschoben sein muss, um exakt synchron zu werden. Diese Angabe steht im Dateinamen, »DELAY 8ms« im Beispiel von Abbildung 55. Diesen Wert übernimmt BeLight automatisch. Das funktioniert allerdings nur, wenn im Dateinamen auch eine Angabe im Format »DELAY XXms« vorhanden ist, ansonsten müssen wir den Wert manuell eintragen – einschließlich evtl. vorhandenem negativem Vorzeichen.

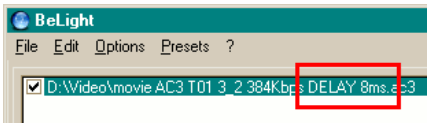

Abbildung 55: Delay-Angabe im Dateinamen.

Das Delay an dieser Stelle schon zu berücksichtigen ist nicht zwingend. Wenn wir später Audio und Video muxen, können wir es auch dort angeben. Wichtig ist: Nur eine der beiden Methoden verwenden! Wer das Delay mit BeSweet abhandelt, darf es später nicht noch ein zweites Mal berücksichtigen. Anders herum genauso: Wer es nicht in BeSweet abhandelt, muss es dann später beim Muxen tun. Empfehlen würde ich die BeSweet-Methode, da dadurch das Delay effektiv auf Null gesetzt und einige mögliche Probleme bei der Wiedergabe umgangen werden.

Damit kommen wir zu den **Advanced Settings** (Abbildung 56). Für den Downmix auf Stereo sollten die Optionen in der Regel so aussehen wie oben. **LFE to LR Channels** bestimmt, mit welchem Pegel der Basskanal in die vorderen beiden Kanäle gemixt werden soll. Um ein Zuviel an Bass zu vermeiden, stellen wir hier **-3db** ein. Mit dem **Output Mode** definieren wir, welche Art von Stereo erzeugt wird: **Mono**, normales **Stereo**, Dolby Pro Logic (**DPL**) oder Dolby Pro Logic II (**DPL2**). Pro Logic codiert Surroundinformationen in die beiden Stereokanäle, so dass – einen passenden Decoder vorausgesetzt – beim Abspielen zumindest ein Teil der ursprünglichen Surroundinformationen wieder hergestellt werden kann. Behalten wir die ursprünglichen 6 Kanäle bei, entfernen wir sowohl bei **LFE to LR Channels** als auch bei **Output Mode** die Haken.

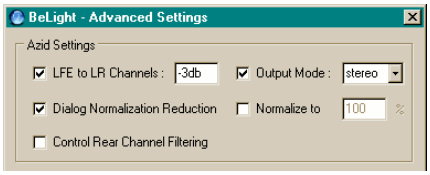

Abbildung 56: Erweiterte Azid-Einstellungen.

Als letztes aktivieren wir die **Dialog Normalization Reduction**. 5.1-AC3-Dateien enthalten in den BSI-Infos eine Angabe, wie weit die subjektiv empfundene Lautstärke der Dialogspur (Center-Kanal) unter dem maximalen Pegel liegt. Die DNR-Funktion ändert den Dialogpegel unter Berücksichtigung der BSI-Info auf –31 dB. Eine Normalisierung wird dadurch nicht beeinträchtigt, da DNR vor allen Normalisierungsfunktionen angewendet wird. Wichtig ist DNR nur dann, wenn verschiedene 5.1-AC3s mit unterschiedlichen Dialogleveln in eine einzelne Zieldatei transcodiert werden sollen, um die unterschiedlichen Level anzugleichen. Beim Transcoding nur einer AC3 (oder mehrerer AC3s mit gleichem Dialoglevel) wirkt sich die Funktion nicht aus. Außerdem sind Stereo-AC3s nicht betroffen, da die keinen Center-Kanal enthalten.

Wenn wir trotz der Nachteile (siehe folgenden Kapitel) AC3 oder 6-Kanal-Vorbis encodieren wollen, müssen wir **Normalize to** anhaken und auf **100** setzen; und zwar anstatt der Normalisierungseinstellung unter **BeSweet OTA** im Hauptfenster.

Dann haken wir noch ganz unten im Fenster **Output Log file** an und können das Optionenfenster wieder schließen. Danach wählen und konfigurieren wir den Encoder. Die Wahl geschieht einfach, indem wir die passende Registerkarte anklicken.

B.2.3 AAC-Encoding mit Nero und Winamp

Dank dimzon und Kurtnoise13 ist AAC mit BeSweet nicht mehr auf den Nero-Encoder beschränkt, sondern kann auch die Encoding-DLL von Winamp verwenden. Uns stehen also zwei hochwertige AAC-Encoder zur Auswahl.

In diesem Kapitel betrachten wir Nero 6. Wer Nero 7 nutzt, sollte ein wenig auf Doom9 stöbern, da die neue Version doch einige Unterschiede aufweist.

Der Nero-Encoder

Nero Digital stellt die gesamte Palette der Möglichkeiten bereit: sowohl LC- als auch HE-AAC und natürlich Stereo- und 6-Kanal-Unterstützung. Die erzeugten AAC-Dateien sind immer in den MP4-Container verpackt, was später beim Muxen wichtig wird.

Im **AAC**-Register (Abbildung 57) stellen wir bei **Encoding Engine** den **Nero-**

Digital-Encoder ein und wählen links daneben die passende Anzahl an **Output Channels**. CBR-Encoding ▸A.1.3 ist wegen der geringeren Qualität wenig empfehlenswert. Deswegen halten wir uns an den **Variable Bitrate Mode**. Für Stereo-Ton liefert uns das **Streaming**-Preset kompakte 75 kbit/s (HE-AAC vorausgesetzt), das ist wenig genug. Für sechs Kanäle können wir circa die HE-Bitraten aus Tabelle 8 erwarten. Dabei klingt **Tape** immer noch akzeptabel. Meistens verwende ich **Internet** oder **Streaming**.

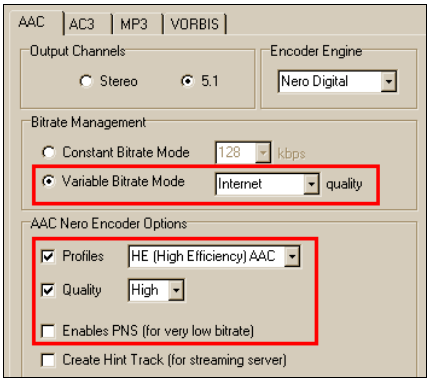

Abbildung 57:
Nero-AAC-Setup in BeLight.

Tape	100
Radio	120
Internet	150
Streaming	190

Tabelle 8: 6-Kanal-Bitraten mit Nero 6.

Als **Profile** bietet sich **HE** an, da wir so den besten Kompromiss zwischen Dateigröße und Qualität erreichen. Die **Quality** können wir immer auf **High** stehen lassen, da die Einstellung **fast** entgegen ihrem Namen kaum Geschwindigkeitsvorteile bringt. Für sehr niedrige Bitraten (die Presets **Tape** und **Radio** bei Stereo) kann es sich auch lohnen, den Haken bei **Enable PNS** zu setzen. Für höhere Bitraten sollten wir den Schalter allerdings deaktiviert lassen.

Der Winamp-Encoder

Der Encoder von Winamp ist auf HE-AAC ausgelegt und unterstützt sowohl Stereo- als auch 6-Kanal-Ton, allerdings nur im CBR-Modus. Wenn wir Winamp verwenden, sollten wir immer daran denken, dass die erzeugten Dateien reine AAC-Dateien sind (also nicht in MP4 verpackt). Das ist später beim Muxen wichtig.

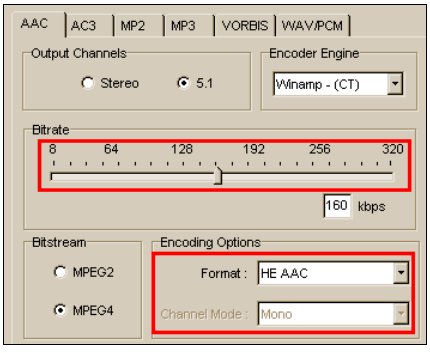

Abbildung 58:
Winamp-AAC-Setup in BeLight.

Die passende Einstellung der **Encoder Engine** im **AAC**-Register ist natürlich **Winamp** (Abbildung 58). Der Schieberegler unter **Bitrate** definiert die verwendete Bitrate, wobei 6-Kanal-AAC mindestens 96 kbit/s und höchstens 213 kbit/s haben kann. Bitraten außerhalb dieses Bereichs lässt BeLight nur für Ste-

reo zu. Mit 160 kbit/s liegt meine Lieblingsbitrate in einer ähnlichen Region wie für Nero. Für Stereo dürften es deutlich weniger sein. Zwischen **64** und **80** sollte im HE-Modus keine Probleme bereiten.

Da BeLight die **Encoding Options** halbautomatisch anpasst, sollten wir immer erst die **Output Channels** und die **Bitrate** einstellen und dann – falls noch nötig – die Optionen anpassen.

HE AAC ist besonders für 6-Kanal-Ton immer eine gute Wahl, schließlich wollen wir ja möglichst Platz sparen, den wir besser für die Videospur verwenden. Für höhere Stereo-Bitraten schlägt BeLight teilweise **HE AAC High** vor, was wir dann übernehmen.

Die Einstellungen zum **Channel Mode** gelten nur für **Stereo**. Sofern für die gewählte Bitrate verfügbar, sollten wir immer Stereo wählen. **Parametric Stereo** ist für sehr niedrige Bitraten ab 48 kbit/s oder weniger interessant. Bei höheren Datenraten schadet die Funktion eher.

Damit ist die AAC-Konfiguration abgeschlossen. Wir können also mit einem Klick auf **Start Processing** das Transcoding starten.

B.2.4 Vorbis-Encoding

Die Vorbis-Einstellungen sind simpel. Grundsätzlich gilt für 6-Kanal-Vorbis ▸ A.2.5 das gleiche wie für Stereo-Vorbis, auch wenn wir hier nur die Stereo-Variante betrachten, da der 6-Kanal-Modus AC3 ähnliche Bitraten benötigt, um ordentliche Qualität zu produzieren.

Zuerst wählen wir wie in Abbildung 59 **Stereo**-Output, und dann bei **Bitrate Management** die Einstellung **Quality**. Vorbis ist auf echtes VBR ▸ A.1.3 im Constant-Quality-Modus ausgelegt und bringt nur da seine volle Leistung. Deswegen sind die **Bitrate**-Modi eher uninteressant. Mit denen könnten wir zwar vor dem Encoding etwas genauer bestimmen, wie groß die encodierte Audiodatei wird. Da wir die endgültigen Größen aber sowieso erst nach dem Audio-Transcoding berechnen, bringt uns das keinen Vorteil. Im Gegenteil würden wir gegenüber dem Quality-Modus Qualität verlieren.

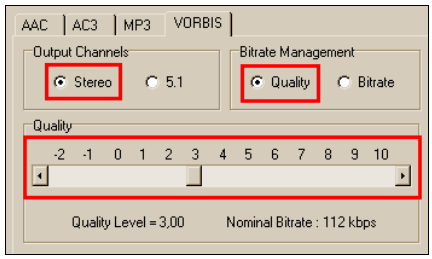

Abbildung 59: Vorbis-Setup in BeLight.

Mit dem Schieberegler stellen wir dann das gewünschte Qualitätslevel ein. **2,00** ergibt etwa 80 kbit/s und ist als sichere Untergrenze gut brauchbar. Für stark

komprimierte 1-CD-Encodings kann allerdings auch ein deutlich kleinerer Wert sinnvoll sein. **5,00** liefert uns etwa 140 kbit/s und taugt gut als Obergrenze.

Mehr gibt es für Vorbis nicht zu beachten. Wir können also mit einem Klick auf **Start Processing** das Transcoding starten.

B.2.5 MP3-Encoding mit Lame

MP3 ▸A.2.1 ist nicht wie Vorbis und AAC von Anfang an stark auf VBR ausgelegt. Im Gegenteil gab es lange Zeit keinen MP3-Encoder mit vernünftigem VBR-Modus. Heute gehört das allerdings dank Lame der Vergangenheit an, und es gilt auch für MP3: variable Bitrate bringt die beste Qualität und sollte immer verwendet werden. Zu einer wichtigen Ausnahme kommen wir weiter unten.

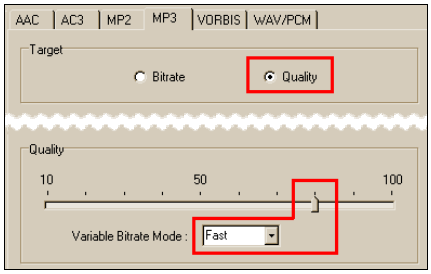

Abbildung 60:
Lame MP3-Setup in BeLight.

Da MP3 nur Stereo unterstützt, entfällt die Auswahl der Zielkanäle. Wir brauchen nur wie in Abbildung 60 im **MP3**-Register unter **Target** auf **Quality** zu klicken, um der VBR-Modus einzustellen. Im **Quality**-Abschnitt stellen wir die gewünschte Qualitätsstufe ein. Seit Lame 3.97, den wir verwenden, gehören die weit bekannten Presets offiziell der Vergangenheit an. Zum Vergleich, welche Qualitätsstufe für welches Preset steht und welche Bitraten wir erwarten sollten, dient Tabelle 9.

Quality 100	--preset extreme	ca. 200 kbit/s
Quality 80	--preset standard	ca. 150 kbit/s
Quality 60	--preset medium	ca. 120 kbit/s

Tabelle 9: Einige Qualitätsstufen von Lame.

Mit Qualität **80** wie im Bild sind wir kompromisslos auf der qualitativ sicheren Seite. In den meisten Fällen sollte **60** allerdings ausreichen. Unter **Variable bitrate mode** sollten wir außerdem **fast** wählen, was den früheren Fast-Presets entspricht. Dieser neue VBR-Modus ist inzwischen so weit ausgereift, dass er auf Hydrogenaudio als »mindestens genauso gut« wie der langsamere **Standard**-Modus bezeichnet wird. Es gibt also kaum noch einen Grund, auf die deutlich höhere Geschwindigkeit zu verzichten.

Damit kommen wir zum Nachteil von VBR-MP3. Genau genommen ist es eigentlich eine Einschränkung des AVI-Containers ▸ A.2.3, denn der ist nicht für VBR-Audio gemacht – egal ob MP3 oder ein anderes Format. Es existiert zwar dank Nando ein wunderbar funktionierender Hack, um trotzdem VBR-MP3 in AVI zu packen. Eine Funktionsgarantie bietet der aber nicht, weshalb paranoide Gemüter lieber auf CBR zurückgreifen sollten, wenn sie AVI verwenden.

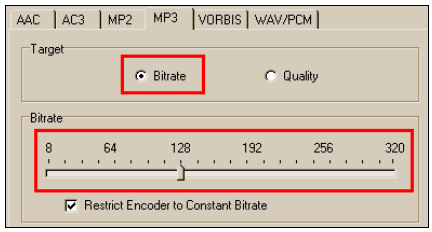

Abbildung 61:
Lame MP3-Setup für CBR.

Um es ganz genau zu sagen, liegt die VBR-Problematik eigentlich daran, dass viele Programme und Tools AVI nicht ganz korrekt und vollständig verstehen. Denn der Wortlaut der AVI-Spezifikation an sich lässt VBR-Audio durchaus zu. Deswegen kann man sich auch darum streiten, ob Nandos Verfahren tatsächlich ein Hack im eigentlichen Sinn ist.

CBR-MP3 erzeugen wir mit den Einstellungen aus Abbildung 61. Unter **Target** wählen wir **Bitrate** und klicken im **Bitrate**-Abschnitt **Restrict encoder**... an. Mit dem Schieberegler stellen wir dann die gewünschte Bitrate ein. Werte unter 96 kbit/s sollten wir dabei der Tonqualität zuliebe besser vermeiden. Genauso dürfte für die meisten Tonspuren mehr als 192 kbit/s eher Platzverschwendung als Qualitätssteigerung sein.

Damit sind die MP3-Einstellungen beendet und wir können per **Start Processing** das Transcoding starten.

B.2.6 AC3-Encoding

Dieses Kapitel bereitet mir ehrlich gesagt Zahnschmerzen, und ich habe es nur am Leben gelassen, weil der Bedarf offenbar da ist. Die einzige Fähigkeit des AC3-Encoders ▸ A.2.5 von BeSweet besteht darin, etwas zu produzieren, das einem AC3-Stream ähnelt. DSPguru selbst sagt sehr deutlich: »ac3enc.dll creates (almost) useless ac3 streams. don't use it to encode things you care for. it's there only for test purposes.« Problem: BeSweets ac3enc ist der einzige freie AC3-Encoder. Und das größere Problem: Kommerzielle Encoder, die auch professionelle Qualität erzeugen, kosten eine gewaltige Stange Geld. Deshalb wird ac3enc trotzdem verwendet. Aber nicht meckern, wenn hinterher Mist rauskommt!

Wichtig ist, dass ac3enc die Normalisierungseinstellungen in BeLights **OTA**-Abschnitt ignoriert. Deshalb müssen wir das Azid erledigen lassen, indem wir die Funktion **Normalize to** in den **Advanced Settings** verwenden und die Normalisierung im **OTA**-Abschnitt ausschalten.

Der Ziel-AC3 eine DVD-übliche Bitrate von 384 oder 448 kbit/s zu gönnen, macht wenig Sinn. Dann hätten wir gleich das Original beibehalten können. Tiefer als die in Abbildung 62 bei **Bitrate** mit dem Schieberegler ausgewählten 256 kbit/s sollten wir für 6-Kanal-Dateien allerdings nicht gehen, um die Einbußen bei der Qualität im Rahmen zu halten. AC3 ist nun mal nicht für niedrige Bitraten ausgelegt, und ac3enc noch viel weniger.

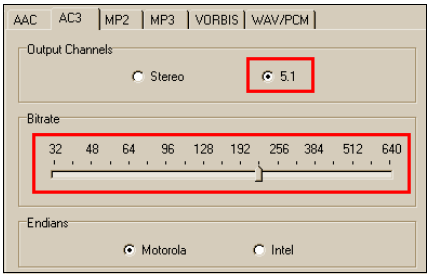

Abbildung 62: AC3-Setup in BeLight.

5.1 unter **Output Channels** behält die sechs Kanäle der Quelldatei bei. Stereo-AC3s zu encodieren oder gar originale Stereo-AC3s zu verkleinern, ist unsinnig. Schon MP3 ist für 2-Kanal-Material besser geeignet, von Vorbis oder AAC ganz zu schweigen.

Damit ist die AC3-Konfiguration auch schon erledigt und wir können mit **Start Processing** das Transcoding beginnen.

B.3 Untertitel

Um Untertitel ▸A.2.6 in den Film einzubinden, stehen uns zwei grundsätzlich verschiedene Alternativen zur Verfügung: fest ins Bild einbrennen oder dynamisch als simple Datenspur muxen. Wie wir beides umsetzen können, damit befassen sich die nächsten beiden Kapitel.

Manch einer mag sagen: »Untertitel interessieren mich eh nicht«. Das ist allerdings gefährlich, da relativ häufig Texteinblendungen im Film als Untertitel gespeichert werden, wenn man sie international nicht bedenkenlos gleich lassen kann. Zum Beispiel sind die eingeblendeten Zeitangaben bei *Spy Game* Untertitel.

B.3.1 Ins Bild eingebrannte Untertitel

Feste Untertitel sind permanent im Video verankert und nicht mehr ausblendbar. Damit wollen wir uns jetzt beschäftigen. Ausblendbare (dynamische) Untertitel behandelt das nächste Kapitel.

Untertitel rippen

Wir starten Vobsub, entweder über dessen Verknüpfung **VobSub Configure** im Startmenü, oder über **Start / Ausführen** mit diesem Befehl:
rundll32.exe vobsub.dll,Configure

Über den **Open**-Button unten links gelangen wir in den Öffnen-Dialog, stellen bei Dateityp auf **Ifo and vobs, for creating idx/sub** um und wählen die vom Ripper erstellte ifo-Datei aus, die sich **vts_01_0.ifo** oder ähnlich nennt. Es folgt Abbildung 63, wo wir den Speicherort der Untertitel angeben. Dann erscheint dieses Fenster. Im Bereich oben rechts stehen die Untertitel, die aus den VOBs gerippt werden sollen. Alles, was wir nicht beim Rippen von der DVD gezogen haben, können wir schon

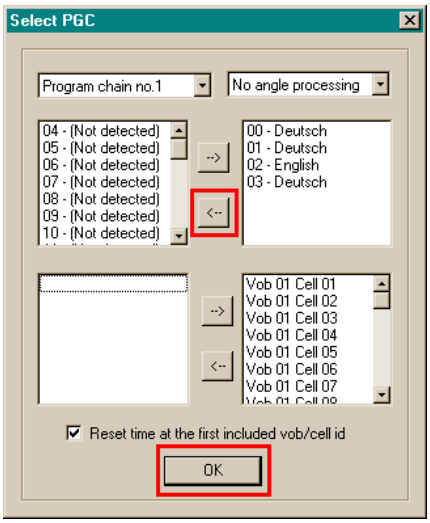

Abbildung 63: Untertitelauswahl in Vobsub.

mal mit dem **<--** Button abwählen. Wer genau weiß, in welcher Spur »seine« Untertitel liegen, kann auch gleich nur diese eine wählen. Wenn wir fertig sind, klicken wir auf **Ok**. Vobsub zieht jetzt die Untertitel aus den VOBs heraus und schreibt sie in eine SUB-Datei. Zusätzlich wird eine IDX-Datei erstellt, die alle Zeitmarken enthält, an denen Untertitel auftauchen sollen. Dann befinden wir uns wieder im Hauptfenster.

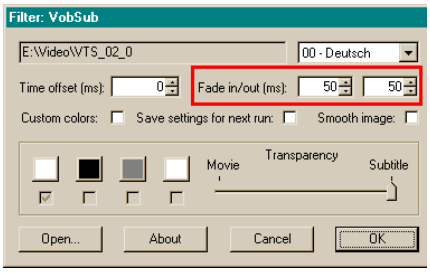

Abbildung 64: Untertitel ein-/ausblenden mit Vobsub.

Normalerweise blendet Vobsub jeden Untertitel kurz ein und wieder aus. Die Standard-Einstellung von 50 ms (Abbildung 64) sieht man kaum, sie lässt das Auftauchen und Verschwinden der Untertitel aber nicht ganz so hart erscheinen wie ohne Fade. Ändern lassen sich die zwei Werte über **Fade in/out**. Danach **Ok** klicken. Nun öffnen wir mit einem Texteditor die IDX-Datei. Sie nennt sich **vts_01_0.idx** oder ähnlich, jedenfalls mit dem **_0** hinten dran. Jetzt wird wichtig, wie Dialog- und Szene-Untertitel gespeichert sind.

Dialog- und Szene-Untertitel in verschiedenen Spuren

Der Anfang einer Untertitelspur in der IDX-Datei könnte in etwa so aussehen:

```
# Deutsch
id: de, index: 0
# Decomment next line to activate alternative name in DirectVobSub
# alt: Deutsch
# Vob/Cell ID: 1, 1 (PTS: 0)
timestamp: 00:03:31:600, filepos 000000000
timestamp: 00:03:33:320, filepos 000001800
timestamp: 00:03:42:360, filepos 000003800
timestamp: 00:03:43:800, filepos 000005000
timestamp: 00:03:45:000, filepos 000006000
timestamp: 00:03:51:840, filepos 000007800
timestamp: 00:03:56:240, filepos 000009800
timestamp: 00:04:01:880, filepos 00000b000
timestamp: 00:04:03:960, filepos 00000c800
timestamp: 00:04:08:640, filepos 00000e800
timestamp: 00:04:18:520, filepos 000010000
```

Jede der vielen Zeilen stellt eine Zeitmarke dar, an der ein Untertitel erscheint. Wir suchen die richtige Sprache aus den Daten oberhalb heraus. Gibt es mehrere Spuren für eine Sprache, ist ein bisschen Detektivarbeit angesagt. Die Spur mit den kompletten Untertiteln dürfte ziemlich umfangreich sein, die mit den Szene-Untertiteln müsste recht kurz ausfallen. Haben wir die richtige Spur gefunden, dann merken wir uns die Zahl hinter **index** und suchen diese Zeilen (direkt über der ersten Untertitel-Spur):

```
# Language index in use
langidx: 0
```

Hinter **langidx** tragen wir die gemerkte Zahl ein.

Dialog- und Szene-Untertitel in einer Spur (Forced Subs)

Auch in diesem Fall suchen wir erst einmal wie oben die passende Sprache heraus und tragen die **index**-Nummer bei **langidx** ein. Dann springen wir zu diesen Zeilen:

```
# ON: displays only forced subtitles, OFF: shows everything
forced subs: OFF
```

Da ersetzen wir das **OFF** durch **ON**, um die Forced Subs einzuschalten. Fertig. Zum Schluss die IDX-Datei speichern. Damit sind die Untertitel erst einmal fertig.

B.3.2 Dynamisch einblendbare Untertitel

Dynamische Untertitel bestehen entweder aus Bildern in Form von Vobsubs oder aus Text. Vobsubs haben wir schon im letzten Kapitel besprochen. Das Vorgehen dort ist immer gleich, egal ob wir sie ins Video codieren oder als eigene Spur muxen wollen.

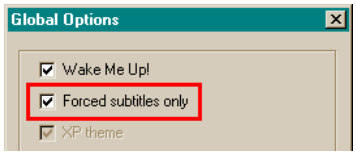

Abbildung 65: Forced-Subs-Option in SubRip.

In diesem Kapitel kümmern wir uns um Text-Untertitel. Das benötigte Programm heißt SubRip. Dort gelangen wir über **Options / Global Options** in den Dialog aus Abbildung 65. Mit einem Haken bei **Forced**

subtitles only können wir, wie mit Vobsub, aus einer kompletten Untertitelspur nur die Szene-Untertitel herausziehen, wenn die nicht in einer extra Spur gespeichert sind.

Im Hauptfenster rufen wir mit dem **VOB**-Button den Öffnen-Dialog auf (Abbildung 66). Am einfachsten ist es nun, mit **Open IFO** die vom Ripper erstellt IFO-Datei zu öffnen. Alternativ können wir auch mit **Open Dir** die VOBs direkt laden. Vorteil der ersten Methode ist, dass dann im Dropdown-Feld darüber die Untertitelspuren richtig bezeichnet werden. Diese Infos stehen nämlich in der IFO-Datei, nicht in den VOBs. Im Dropdown-Feld stellen wir die gewünschte Sprache ein und widmen uns dann der rechten Seite des Fensters (Abbildung 67). **Last Time Code** muss Null sein. Steht hier ein Wert, addiert ihn SubRip zu jeder Zeitmarke dazu, was zu asynchronen Untertiteln führt.

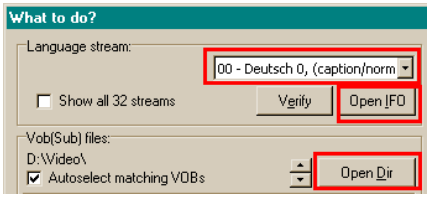

Abbildung 66: DVD-Untertitel laden in SubRip.

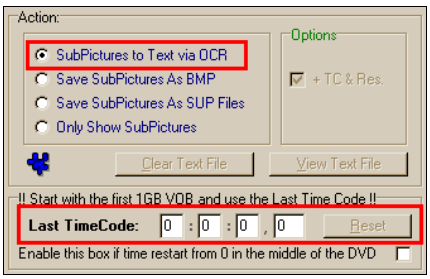

Abbildung 67: OCR-Erkennung und Timecode-Anpassung.

Da wir eine Textdatei erzeugen wollen, die Untertitel auf der DVD aber als Bilder gespeichert sind, muss sie SubRip per OCR-Texterkennung in Text umwandeln. Dabei erkennt das Programm die Buchstaben aber nicht wie die professionellen OCR-Programme selbständig, sondern fragt nach, welches Grafikmuster zu welchem Buchstaben gehört. Das geschieht im Dialog aus Abbildung 68. Über den rot umrandeten Buchstaben (hier das »H«) ist sich SubRip im Unklaren. Also geben wir ihm im Eingabefeld in der Mitte den passenden Buchstaben an. Rechts daneben können wir noch fett, kursiv und unterstrichen als Auszeichnung wählen.

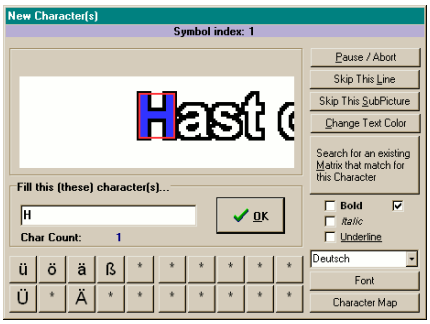

Abbildung 68: OCR-Erkennungsdialog von SubRip.

Bei den ersten Untertiteln fragt SubRip noch bei fast jedem Buchstaben nach dessen Bedeutung. Eben so lange, bis mehr oder weniger jeder Buchstabe des Alphabets zweimal (wegen der Groß- und Kleinbuchstaben) vorgekommen und von uns identifiziert ist. Dann geht der Lesevorgang fast automatisch. Nur ab und zu stößt SubRip noch auf einen unbekannten Buchstaben, macht sich dann aber mit blinkendem Fenster und Sound bemerkbar, wir können also zwischendurch ruhig am Computer weiterarbeiten. Viel länger als zehn Minuten dürfte der Vorgang insgesamt nicht dauern.

Wenn SubRip den letzten Untertitel erkannt hat, landen wir im Editor-Fenster, das sich am unteren Rand des Hauptfensters öffnet. Wer später noch Untertitel in anderen Sprachen bearbeiten will, sollte jetzt im Hauptfenster mit **Characters Matrix / Save Characters Matrix File** die gerade beim Rippen erstellte Zeichen-Matrix sichern. In der sind die Informationen enthalten, welches Grafikmuster zu welchem Buchstaben gehört. Damit entfällt bei allen weiteren Untertiteln das erneute Eingeben der richtigen Buchstaben. Aber Achtung: Da die Schriftart der Untertitel von DVD zu DVD unterschiedlich ist, gilt eine Zeichen-Matrix immer nur für die gerade gerippte DVD. Für einen anderen Film müssen wir die Buchstabenerkennung wieder von vorne durchführen. Ich habe es sogar schon erlebt, dass verschiedene Sprachen auf ein und der selben DVD unterschiedliche Schriftarten verwenden.

Damit zurück zum Editor-Fenster (Abbildung 69), in dem wir den gerade gerippte Untertitel in Textform (genauer: im SubRip-Format) sehen. Ich gehe davon aus, dass wir die Untertitel-Spur am Ende auch im SubRip-Format speichern wollen. Dann können wir (1) ignorieren. Die richtige Bildrate (die uns DGIndex verraten hat) müssten wir dort nur einstellen, wenn wir die Untertitel in einem Frame-basierten Format wie MicroDVD speichern wollten. Mit dem Button (2) lassen sich die gröbsten Fehler der Texterkennung korrigieren. Die ist zwar recht gut, aber nicht fehlerfrei. Besonders das große I und kleine l werden oft verwechselt, weil zwischen den beiden Buchstaben oft schlicht kein Unterschied besteht. Bei (3) lassen sich Zeitkorrekturen vornehmen, wenn der Untertitel nicht synchron zum Film läuft. Das dürfte aber kaum vorkommen, wenn Untertitel und Film direkt von der DVD stammen. Mit (4) könnten wir die Untertitel in ein anderes Format konvertieren und (5) dient zum Splitten der Untertitel in mehrere Dateien. Nötig ist das nur, wenn wir die Untertitel in einer externen Datei abspeichern (ansonsten wird das Splitten später zusammen mit dem gesamten Film erledigt), und das wiederum dürfte nur nötig sein, wenn der Standalone-Player mit Untertitelspuren Stress macht. Da Standalone-Player im Encodingwissen kein Thema sind, bleibt Button (6), der die aktuelle Untertitel-Datei speichert. Das sollten wir natürlich nicht vergessen.

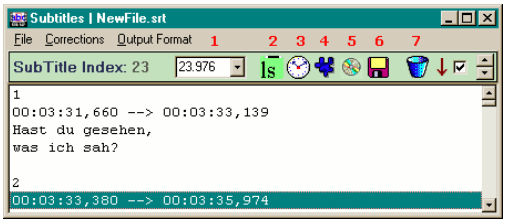

Abbildung 69: SubRips Editorfenster.

Wenn wir weitere Untertitel erkennen wollen: Nach dem Speichern das Editorfenster über das Papierkorb-Symbol (7) leeren, mit dem **VOB**-Button den Öffnen-Dialog aufrufen, die neue Untertitelspur auswählen und auf der rechten Seite des

Fensters oben die Character-Matrix-Datei laden und mit **Reset** den Timecode nullen, sonst passen die Zeitmarken nicht mehr.

Ganz zum Schluss bleibt nur noch, die Untertitel-Spur(en) in eine Textverarbeitung zu laden und durch die Rechtschreibprüfung zu jagen. Denn – wie gesagt – die Texterkennung ist zwar sehr gut, aber bei weitem nicht fehlerfrei. Beim Abspeichern müssen wir dann nur darauf achten, dass wir die Datei auch wieder als reinen Text sichern, denn mit Untertiteln im StarOffice- oder Word-Format kann kein Player etwas anfangen. ;-)

B.4 Die Encoding-Frontends

Encoding-Frontends sind die Kommandozentralen für den kompletten Backup-Prozess. Ein solches Programm steuert mehr oder weniger das ganze Softwarepaket, das wir zum Encoding brauchen, erstellt die nötigen Skripte und unterstützt uns bei den notwendigen Berechnungen. Im Idealfall stellen wir im Frontend einmal alle Parameter ein, starten den Vorgang und erhalten einige Stunden später die fertige Filmdatei. Nur das Ripping der DVD muss traditionell von Hand erledigt werden.

Tabelle 10 auf der nächsten Seite bietet eine Übersicht über die Fähigkeiten der Frontends, die wir uns näher ansehen. Im Moment sieht das natürlich noch etwas mager aus, da Gordian Knot das einzige Frontend ist.

B.4.1 Gordian Knot

Das Gordian-Knot-Projekt wurde von TheWEF und Aquaplaning ursprünglich als reiner Bitratenrechner gestartet. Im Lauf der Zeit wuchs das Programm dann immer mehr und kann mit Recht als ein Urvater der Encoding-Frontends überhaupt angesehen werden. 2003 gab TheWEF die Entwicklung auf und veröffentlichte den Quellcode unter der GPL. Schließlich fand sich Len0x als Nachfolger, der Gordian Knot weiter betreute und z. B. die Unterstützung für x264 hinzufügte. Seit Herbst 2005 steht die Entwicklung wieder still, nachdem sich Len0x von seinen Programmierprojekten zurückgezogen hat. Dank der GPL-Lizenzierung steht es aber jedem frei, sich vom Gordian Knot *Sourceforge-Projekt* den Quellcode zu besorgen und das Programm weiterzuentwickeln.

Allerdings gibt es gute Gründe, die dagegen sprechen. Da sich Gordian Knot fürs Encoding ganz zentral und ausschließlich auf VirtualDubMod verlässt, legt es sich damit genauso auf das VfW-Encodingframework ▸ A.4.1 fest. Diese veraltete Technologie ist so untrennbar mit Gordian Knot verzahnt, dass es nicht lohnend erscheint, das Programm an modernere Verfahren anzupassen.

Gerade fürs DVD-Backup in AVI-Dateien stellt Gordian Knot trotzdem nach wie vor ein ausgereiftes und bestens funktionierendes Frontend dar. Natürlich spielt auch der Nostalgiefaktor kräftig mit. Viele der heutigen Experten haben ihre

	Gordian Knot
	Container
Matroska	nur VDubMod
MP4	Nein
AVI	Ja
OggMedia	Ja
	Video
Xvid	Nur VfW
DivX	Probleme mit v6
x264	nur VfW
	Audio
Anzahl Spuren	2
MP3	Ja
Vorbis	Ja
AC3	Ja
AAC	Nein
	Dynamische Untertitel
SubRip	1 Spur
Vobsub	Nein

Tabelle 10: Funktionsumfang der Encoding-Frontends.

ersten Schritte mit Gordian Knot gemacht – möglicherweise sogar noch mit DivX 3.11. Und wer will schon die altvertraute, zuverlässige Oberfläche zum alten Eisen geben, mit der man noch im Tiefschlaf ein einwandfreies DVD-Backup zustande bringen würde. :)

B.4.1.1 Gordian Knot einrichten

Nach dem ersten Start von Gordian Knot wechseln wir in das Register **Program Paths** und stellen dort ein, wo die verschiedenen Programme liegen. Die **Don't Check**-Checkbox verhindert, dass Gordian Knot nach dem entsprechenden Programm sucht und sich beschwert, wenn kein Pfad dafür angegeben ist. Den Haken setzen wir bei allen Tools, die wir nicht verwenden und deshalb nicht installiert haben. Nandub ist ein Kandidat dafür, denn das bräuchten wir nur fürs DivX-3.11-Encoding.

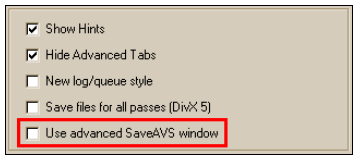

Abbildung 70: Globale Optionen in Gordian Knot.

Anschließend wechseln wir ins Register **Options**. Wichtig ist hier zuerst der fehlende Haken bei **Use advanced SaveAVS window** (Abbildung 70). Nicht, dass das erweiterte Fenster schlecht wäre. Ich mag es sogar sehr gerne. Aber alles, was wir hier im Encodingwissen besprechen, bietet auch das einfachere Fenster. Und das lässt sich schön kompakt mit einem einzelnen Screenshot darstellen. Also raus mit dem Haken. Die restlichen Optionen dürften einigermaßen selbsterklärend sein. Mit den **Advanced Tabs** sind die Registerkarten fürs DivX-3.11-Encoding gemeint. Vielleicht sollte man die langsam als »obsolete tabs« bezeichnen ;). Deutlich interessanter wird es dann auch bei den **Default codec settings** ▸ A.4.1 ▸ A.2.4.

Achtung! DivX 6 unterscheidet sich so stark von seinen Vorgängern, dass er nicht ohne Probleme mit Gordian Knot benutzt werden kann. Um die neue DivX-Generation sicher zu verwenden, müssen wir das Encoding manuell mit VirtualDubMod durchführen, Gordian Knot also nur als Rechner und Skript-Generator benutzen. Der letzte DivX 5 ist Version 5.2.1, dessen Konfiguration wir als Spezialthema in Kapitel C.2 unter die Lupe nehmen.

Die in Abbildung 71 getroffenen Einstellungen gelten als Standard für alle Encodings, können aber im Einzelfall auch wieder überschrieben werden.

Das Video wird unabhängig vom Codec in mindestens zwei Durchgängen (Passes) codiert. Im ersten Durchgang sammelt der Codec Informationen über das

Video ohne den Film schon wirklich zu encodieren. Aus diesen Informationen wird eine möglichst gute Verteilung der verfügbaren Bitrate berechnet und im zweiten Durchgang der Film erzeugt. Xvid ist an dieser Stelle fertig. Bei DivX und x264 sind noch zusätzliche Durchgänge möglich, um die Qualität weiter zu steigern.

DivX unterstützt prinzipiell unendlich viele Passes, Xvid beschränkt sich auf zwei. Ist das nicht ein Nachteil? Nein. Xvid bearbeitet in den beiden Durchgängen das Video so intensiv und ausgeklügelt, dass weitere Passes gar nicht nötig sind. DivX dagegen erreicht sein Ziel über kontinuierliche Verbesserungen, also zusätzliche Passes.

Für beide Codecs konfigurieren wir separat 1st und 2nd bzw. Nth Pass. Bei DivX haben wir zusätzlich die Möglichkeit, für den allerletzten Durchgang noch einmal veränderte Einstellungen zu wählen.

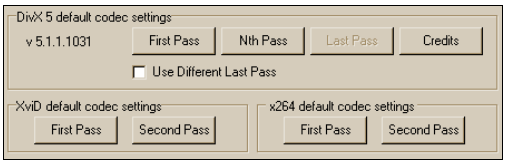

Abbildung 71:
Gordian Knots Standard-Codecsetup.

Bleiben die Credits, also der Abspann des Films. Da es sich dabei meistens nur um weiße Laufschrift auf schwarzem Hintergrund handelt, können die Credits viel stärker komprimiert werden als der restliche Film. Für Xvid kümmert sich Gordian Knot um die richtige Konfiguration, DivX-Credits konfigurieren wir über den **Credits**-Button selbst.

Damit sind wir mit der Grundkonfiguration fertig und können ins Register **Bitrate** wechseln.

B.4.1.2 Kalkulieren der Bitrate

Im **Bitrate**-Register von Gordian Knot laden wir unseren Film, wählen den gewünschten Container ▸A.2.3 und Video-Codec ▸A.2.4 aus und berechnen die Bitrate, mit der später encodiert wird.

Mit **Open** unten links laden wir die von DGIndex angelegte D2V-Datei. Es öffnet sich ein zusätzliches Fenster mit dem Video. Das ist im Moment noch uninteressant, darf aber nicht geschlossen werden. Neben dem **Open**-Button bei **FPS** sehen wir die Framerate, die mit der aus DGIndex übereinstimmen sollte, und unter **Duration** steht die Länge des Films.

Bei **Mode** oben links stellen wir ein, ob wir die Bitrate oder die endgültige Dateigröße des Films berechnen wollen. Da die Größe durch die gewünschte

Anzahl CDs bzw. DVDs schon feststeht, stellen wir **Calculate average bitrate** ein. Dann wählen wir **Container** und **Codec** aus und kümmern uns um die endgültige Größe des Films.

Bei **Total Size** (Abbildung 72) stellen wir die Rohling-Größe und gewünschte Anzahl Discs ein. Alternativ können wir auch rechts direkt die gewünschte Größe in MB angeben. Bei mehreren CDs macht es Sinn, die **Total File Size** ein paar MByte nach unten zu schrauben. Schließlich müssen wir den Film zum Schluss an einer sinnvollen Stelle schneiden, um ihn auf die CDs aufzuteilen. Da schadet es nicht, ein wenig Luft zu haben. Die **Split final file into CDs**-Funktion würde das zwar automatisch übernehmen, achtet dabei aber nur auf die Dateigröße, so dass es schon einmal vorkommen kann, dass der Held des Films mitten im Satz unterbrochen wird, weil die nächste CD fällig wird. Deshalb sollten wir das Splitten immer manuell erledigen.

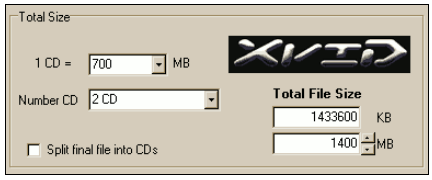

Abbildung 72: Auswahl der Zielgröße in Gordian Knot.

Der Abschnitt in Abbildung 73 enthält alle Daten, die außer dem Video auf die Disc müssen. Mit **Select** wählen wir die erste erstellte Audiospur ▸A.2.5 aus, deren Größe in die Felder darüber übernommen wird. Das Ganze noch einmal für eine eventuelle zweite Audiospur. Der **Files**-Abschnitt ist für Dateien gedacht, die zusätzlich zum Film auf die Disc sollen, z. B. dynamische Untertitel oder Special Features von der DVD. Je mehr hier steht, desto geringer wird natürlich die Qualität des Films, das sollten wir immer bedenken.

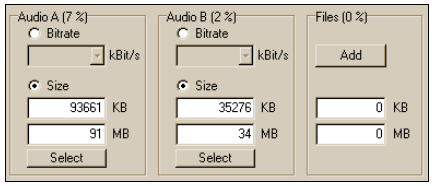

Abbildung 73: Größe der Audiospuren und zusätzlicher Daten.

Schließlich wählen wir bei **Interleaving** (Abbildung 74) noch die zum Audio passende Einstellung. Hier geht es um den Platz, den die Dateistruktur des Containers selbst belegt. Das gewählte Audioformat und die Anzahl der Tonspuren wirken sich auf den Platzbedarf aus. **Calculate Frame-Overhead** berechnet den Overhead-Anteil, der nicht von den Audiospuren abhängt. Für Xvid sollte diese Option deaktiviert bleiben.

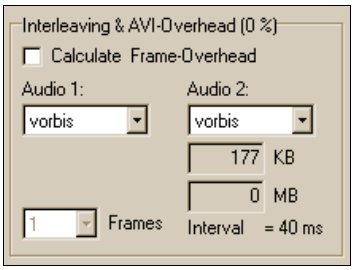

Abbildung 74: Berechnung des Overheads.

Gordian Knot versteht kein AAC-Audio. Wollen wir trotzdem AAC verwenden, ist **vorbis** eine gute Annäherung für die Berechnung.

Damit sind alle Einstellungen erledigt. Rechts bei **Average Bitrate** steht jetzt die Bitrate, mit der Gordian Knot den Film erstellt. Der Wert gibt an, wie viele Kilobit pro Sekunde im Durchschnitt zur Verfügung stehen. Starke Schwankungen innerhalb des Videos sind nichts Ungewöhnliches, da der Codec die Bitrate je nach Bedarf verteilt, schnelle Szenen brauchen für die gleiche Qualität mehr Bits als langsame.

Jetzt können wir uns um die Videospur kümmern. Dazu wechseln wir auf das Register **Resolution**.

B.4.1.3 Vorbereiten des Videos

In diesem Kapitel beschäftigen wir uns zuerst damit, wie wir bei einem klassischen Encoding mit quadratischen Pixeln Cropping und Resizing durchführen. Anschließend kümmern wir uns um die Besonderheiten eines anamorphen Encodings, denn Gordian Knot bringt dafür keine ausdrückliche Unterstützung mit.

Klassisches Cropping und Resizing

Im **Resolution**-Register stellen wir im Abschnitt **Input Resolution** (Abbildung 75) den Typ der DVD ▸ A.2.2 und die Art des Bildes ▸ A.3.2 ein. **PAL**-DVDs mit einer Auflösung von 720 × 576 Pixeln sind Standard in Europa, Japan und dem Nahen Osten, **NTSC** mit 720 × 480 Pixeln findet in Amerika Anwendung. **Anamorphic (16:9)** ist die richtige Wahl für 16:9-DVDs, **non anamorphic (4:3)** dagegen gilt für 4:3-DVDs. Aus DGIndex wissen wir schon, welches Format die DVD hat, z. B. PAL 16:9. Das dürfte für die meisten Region-2-DVDs (Europa) zutreffen.

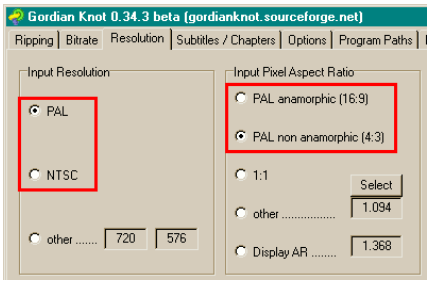

Abbildung 75: Quellvideo-Paramter im Resolution-Register.

Die rechte Seite des **Resolution**-Registers (Abbildung 76) ist fürs Cropping ▸ A.3.1 zuständig, also für das Wegschneiden der schwarzen Ränder um das Video. Wer will, kann sich auf den **Auto-Crop**-Mechanismus verlassen. Ich traue solchen Automatiken aber grundsätzlich nicht so recht über den Weg und mache das Cropping deshalb lieber selbst. Dafür stellen wir auf **Pixel**. Jetzt nehmen wir uns das Fenster mit dem Video vor, springen dort zu einer möglichst hellen Stelle und schneiden mit den Reglern im Bild auf der rechten Seite die schwarzen Rän-

der vollständig weg. **Smart Crop Left-Right** und **Smart Crop All können** wir später *nach* der Wahl der endgültigen Auflösung verwenden, um zusätzlich noch so viele Pixel wegzuschneiden, dass der **Aspect Error** nahezu oder ganz **0** wird.

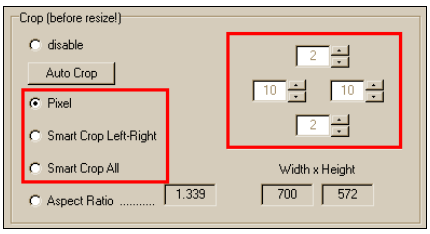

Abbildung 76: Cropping-Optionen.

Anschließend müssen wir im unteren Teil des Fensters (Abbildung 77) eine geeignete Auflösung wählen. Dazu stellen wir **W-Modul** und **H-Modul** beide auf **16**. Damit erlaubt Gordian Knot nur noch Auflösungen, die sowohl horizontal als auch vertikal durch 16 teilbar sind. Beim Einstellen der Auflösung mit dem großen Schieber müssen wir auf **Aspect Error** und **Bits/(Pixel*Frame)** (BPF oder relative Bitrate) achten. Wie wir eine möglichst gute Bildgröße finden, haben wir im Hintergrundwissen schon im Kapitel A.3.3 über die Zielauflösung betrachtet.

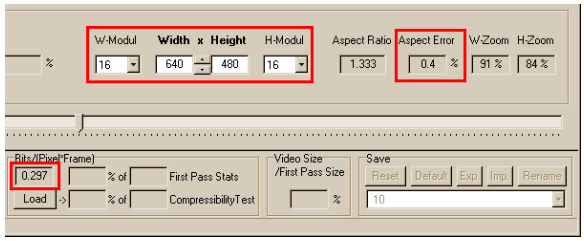

Abbildung 77: Wahl der Zielauflösung.

Haben wir uns für eine Zielauflösung entschieden, wird es Zeit, den Film in seiner endgültigen Auflösung anzusehen. Dazu stellen wir im Video-Fenster auf **View / Resized** um. Das Bild sollte jetzt ohne schwarze Ränder und ohne Verzerrungen erscheinen. Vorsicht dabei, auch ein verzerrtes Bild sieht auf den ersten Blick gern recht normal aus. Wenn es noch nicht passt, ist wahrscheinlich die **Input Resolution** und/oder das **Input Pixel Aspect Ratio** verkehrt eingestellt.

Falls wir dynamische Untertitel und/oder Kapitel einbinden wollen, wechseln wir jetzt ins Register **Subtitles/Chapters**, ansonsten können wir gleich mit der Filterkonfiguration fortfahren.

Anamorphe Zielauflösung

Gordian Knot unterstützt anamorphes Encoding ▸A.3.2 nicht ausdrücklich, weshalb wir um ein wenig Tricksen nicht herumkommen. Wir laden wie gewohnt den Film und kalkulieren die Bitrate. Dann wechseln wir ins **Resolution**-Register und stellen die passende **Input Resolution** ein. Soweit unterscheidet sich das Vorgehen nicht vom normalen Ablauf. Das **Input Pixel Aspect Ratio** setzen wir jetzt allerdings auf **1:1** und erledigen dann das Cropping.

Cropping

Das zugeschnittene Bild sollte wie beim nicht-anamorphen Encoding ein durch 16 teilbares Seitenverhältnis haben, um dem Codec das Encodieren nicht unnötig zu erschweren. Mit etwas Glück passt das nötige Cropping auch zu dieser Mod16-Anforderung. Oft klappt das aber nicht so gut und wir stehen vor der Entscheidung, entweder recht viele zusätzliche Pixel vom Bild wegzuschneiden, um auf die kleinere Mod16-Auflösung zu kommen, oder für die größere Mod16-Auflösung einen Rest schwarzer Balken in Kauf zu nehmen. Beides stellt nicht gerade die Ideallösung dar. Wir entscheiden uns dafür, etwas Schwarz stehen zu lassen, denn der AviSynth-Filter FillMargins kann das leicht korrigieren. Wie viele schwarze Pixelreihen an den einzelnen Rändern übrig bleiben, merken wir uns.

Ob die zugeschnittene Auflösung wirklich das Mod16-Kriterium erfüllt, lässt sich einfach überprüfen, wenn wir **W-Modul** und **H-Modul** wie gewohnt auf **16** stellen und die horizontale Auflösung (**Width**) auf den Wert, der nach dem Cropping übrig bleibt. In der Regel sollte das entweder **720** oder **704** sein. Sobald **WidthxHeight** im Abschnitt **Output resolution** mit **WidthxHeight** unter **Crop (before resize!)** übereinstimmt, haben wir eine Mod16-Auflösung getroffen. Ändern dürfen wir die Auflösung dabei nur über die vier Cropping-Regler, nicht über den großen Auflösung-Schieber!

AviSynth-Skript bearbeiten

Über **Save & Encode** im Fenster mit dem Videobild gelangen wir in den Dialog **Save .avs**, wo wir das AviSynth-Skript an unsere anamorphen Erfordernisse anpassen müssen. Zuerst treffen wir wie später in Kapitel B.4.1.5 erklärt alle nötigen Einstellungen zu Rauschfiltern, Vobsubs, Deinterlacing usw. Über den **Edit**-Button öffnen wir dann das Skript und sorgen mit einem Haken bei **No comments** für Übersicht. Das Skript sieht jetzt in etwa so aus wie in Abbildung 78 auf der nächsten Seite.

Im **Plugins**-Abschnitt fügen wir wie im Bild die `LoadPlugin`-Zeile für die *FillMargins.dll* hinzu, wenn wir vorhin beim Cropping schwarze Ränder stehen lassen haben. Direkt unter `crop()` setzen wir dann `FillMargins()` ein. Die vier Argumente der Funktion stehen für die vier Ränder des Bildes in dieser Reihenfolge:

```
FillMargins(links,oben,rechts,unten)
```

Für jeden Rand tragen wir ein, wie viele schwarze Pixelreihen übertüncht werden sollen. Zu viele sollten das nicht sein. Bei mehr als vier dürften die Bildränder

langsam anfangen, unsauber auszusehen.

Bleibt noch, den Resizing-Filter abzuschalten, was Gordian Knot leider nicht automatisch tut. Wir kommentieren also die **Resizing**-Zeile aus (die je nach verwendetem Filter nicht unbedingt `LanczosResize()` heißen muss), indem wir ein # davor setzen oder die Zeile komplett löschen.

Damit Gordian Knot die manuellen Änderungen nicht wieder überschreibt, klicken wir nun gleich auf **Save & Encode**, um das Encoding zu durchzuführen.

```
#  PLUGINS
LoadPlugin("C:\Programme\Encoding\DGMPGDec\DGDecode.dll")
LoadPlugin("C:\Programme\Encoding\AviSynth Plugins\FillMargins.dll")

#  SOURCE
mpeg2source("E:\Video\movie.d2v", idct=0)

#  CROPPING
crop(0,72,720,432)
FillMargins(0,2,0,0)

#  RESIZING
#LanczosResize(720,432)
```

Abbildung 78: AviSynth-Skript für anamorphes Encoding.

B.4.1.4 Dynamische Untertitel und Kapitel einbinden

Im Register **Subtitles/Chapters** haben wir die Möglichkeit, SubRip-Untertitel und Kapitel hinzuzufügen.

Unter **SRT Subtitles** können wir mit dem **Select**-Button eine SubRip-Datei laden. Leider unterstützt Gordian Knot im Moment nur eine einzelne SubRip-Spur. Sollen es mehr werden, müssen wir die nach dem Encoding manuell einbinden. Das gleiche gilt für dynamische Vobsubs. VirtualDubMod kann mit denen nichts anfangen, und da Gordian Knot auf VDubMod aufbaut, bleibt uns auch hier nur, später selbst Hand anzulegen. Außerdem ist die Untertitelwahl für den AVI-Container ▸A.2.3 deaktiviert, denn SubRip in AVI funktioniert nur mit AVI-Mux GUI, nicht mit VirtualDubMod.

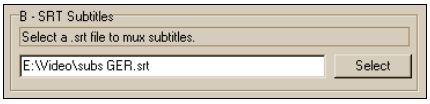

Abbildung 79: Einbinden von SubRip-Untertiteln.

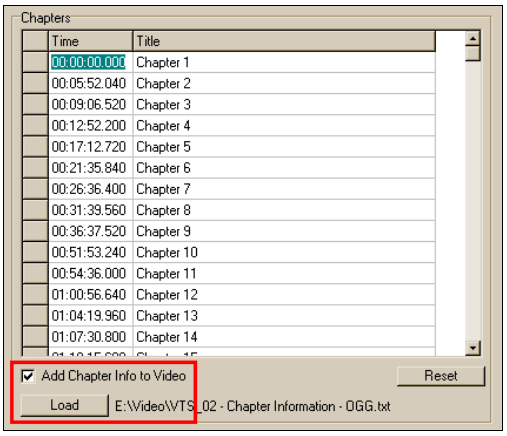

Abbildung 80: Geladene Kapitelliste im Ogg-Format.

Für die Kapitel ist die rechte Seite des Registers zuständig. Über **Load** laden wir die von Chapter-X-tractor erzeugte Textdatei mit den Kapitelmarken. Im großen Fenster über dem **Load**-Button können wir die Kapitel bearbeiten. Wenn wir die Zeitmarken anpassen, ist Vorsicht geboten, dass die Reihenfolge gewahrt bleibt, also nicht z. B. das fünfte Kapitel früher beginnt als das vierte. Nicht wundern, wenn der **Load**-Button für den AVI-Container deaktiviert bleibt. Das ist ganz normal, denn AVI unterstützt keine Kapitelinformationen.

B.4.1.5 Encoding-Parameter konfigurieren

Eigene Einstellungen für den Abspann

Im Video-Fenster (Abbildung 81) müssen wir uns entscheiden, ob wir den Abspann mit anderen Einstellungen encodieren als den Rest des Films. Wenn der Abspann nur aus weißer Laufschrift auf schwarzem Hintergrund besteht, können wir ihm deutlich weniger Bits gönnen als dem Film, ohne dass sich die niedrigere Qualität zu deutlich bemerkbar macht. Dadurch wird der Abspann kleiner und für den Rest bleibt mehr Platz übrig. Besteht der Abspann aber aus mehr als Laufschrift (z. B. bei der *Monster AG*), sollten wir die Finger von der Funktion lassen. Ein mit Abspann-Einstellungen codiertes normales Videobild sieht grauenhaft aus. Außerdem ist ein zusätzlich geschrumpfter Abspann hauptsächlich für 1-CD-Encodings interessant, bei denen es sich tatsächlich lohnt, um jedes zusätzliche MByte für den eigentlichen Film zu kämpfen.

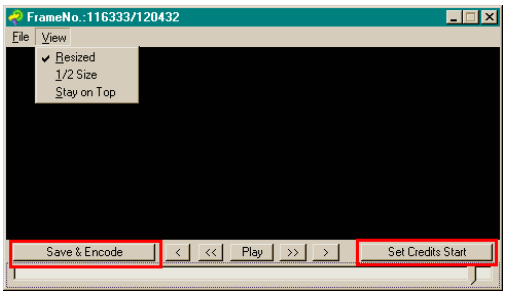

Abbildung 81: Video-Vorschaufenster von Gordian Knot.

Mit dem Schieberegler suchen wir den Anfang des Abspanns heraus. Das Bild sollte an der Stelle schon komplett schwarz sein, im Zweifel springen wir lieber noch etwas weiter nach hinten. Dann klicken wir auf den Button **Set Credits Start**. Gordian Knot weiß jetzt, dass hier die Credits beginnen sollen.

Filter konfigurieren

Über den **Save & Encode**-Button gelangen wir anschließend zum Fenster aus Abbildung 82, in dem wir alle Filter konfigurieren. Hat unser Film feste Untertitel, stellen wir unter **Subtitles** (1) die mit Vobsub erstellte und bearbeitete IDX-Datei ein. Dynamische Untertitel interessieren hier nicht.

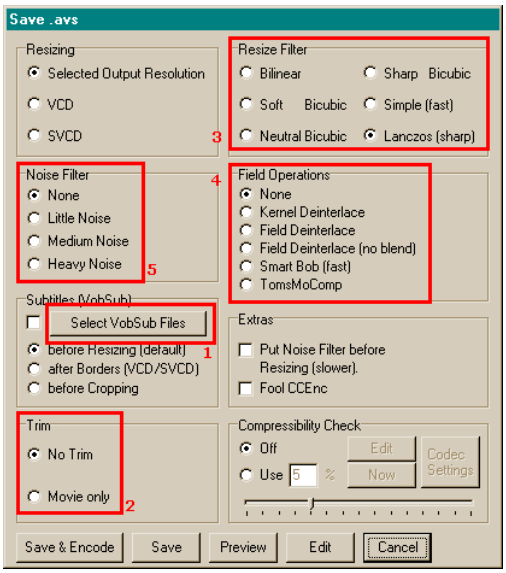

Abbildung 82:
Dialog für die Filterkonfiguration.

Trim (2) kontrolliert das separate Encoding des Abspanns. Das Fenster oben gilt für Xvid. Da der Codec mit den Zonen schon eine Möglichkeit eingebaut hat, die Credits zu codieren, müssen wir nichts Besonderes einstellen. **No Trim** ist die richtige Wahl.

Anders bei DivX, der intern keine Möglichkeit fürs Credits-Encoding besitzt (Abbildung 83). Gordian Knot codiert deshalb Film und Credits getrennt und fügt die Dateien dann zusammen. **Both** ist dafür die richtige Einstellung. **No Trim** bedeutet bei DivX, die Credits-Einstellung zu ignorieren und den kompletten Film unverändert am Stück zu codieren.

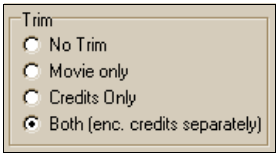

Abbildung 83:
Trim-Setup für DivX.

Mit dem **Resize Filter** (3) definieren wir die Methode, mit der das Video auf die eingestellte Zielauflösung umgerechnet wird. **Bilinear** und **Soft Bicubic** glätten das Bild dabei tendenziell. Dadurch geht zwar etwas Schärfte verloren, gleichzeitig steigt aber auch die Komprimierbarkeit. **Sharp Bicubic** und **Lanczos** zeichnen eher scharf, **Neutral Bicubic** und **Simple** liegen irgendwo dazwischen. Für 1-CD-Encodings bietet sich ein weicher zeichnender Filter an, für höhere Zielgrößen eher ein schärferer. Lanczos ist dabei sehr beliebt.

Die **Field Operations** (4) sind nur bei interlaced Video interessant, also wenn sich am Anfang in DGIndex Kammeffekte im Bild gezeigt haben. In diesem Fall wählen wir einen der Deinterlacer: **Field Deinterlace** oder **TomsMoComp** bieten sich an.

Wenn das Bild unnatürlich stark rauscht, können wir das bei (5) mit einem **Noise Filter** bekämpfen. Gerade bei älteren Filmen kann das schon einmal nötig sein. Wichtig ist, nicht gleich in die Vollen zu gehen, denn ein Rauschfilter vernichtet immer nicht nur das Rauschen, sondern auch einige Details des Bildes.

Der Kompressionstest

Zu guter Letzt können wir noch einen **Compressibility Check** durchführen, der uns genauer als das BPF-Verhältnis Auskunft über die Qualität gibt. Gordian Knot nimmt dazu in regelmäßigen Abständen einige Frames des Films (standardmäßig insgesamt 5 %) und encodiert ohne Rücksicht auf die Dateigröße mit einem festen Quantizer von 2, um maximale Qualität zu erreichen. Das Ergebnis lässt sich auf den ganzen Film hochrechnen und wir erhalten durch Vergleich mit dem BPF einen Prozentwert, an dem sich recht schön die Komprimierbarkeit dieses einen Films abschätzen lässt.

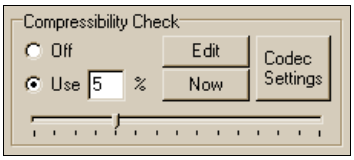

Abbildung 84: Optionen für den Kompressionstest.

Mit **Codec settings** (Abbildung 84) können wir überprüfen, ob die Einstellungen für den Compcheck passen. Für DivX heißt das:

- Modus **1-pass quality-based**.
- **Quantizer** von **2**.

Xvid sieht so aus:

- **Encoding type** auf **Single pass** stellen.
- **Target quantizer** auf **2.00** setzen.

Den Rest so einstellen wie fürs Encoding des ganzen Films geplant. Und Folgendes gilt für x264:

- **Single Pass - Quantizer** als Encoding-Modus.
- **Quantizer** auf **18** stellen.

Mit einem Klick auf **Now** öffnet sich VirtualDubMod und führt den Test durch. Gordian Knot ist solange blockiert. Hinterher landen wir auf dem Register **Resolution**. Der Bereich unten in der Mitte sieht jetzt etwa so aus: Der wichtige Wert sind die **68.2** Prozent. Gut sind zwischen 60 und 80 Prozent. Darunter wird die Qualität schnell schlechter, darüber vergrößert sich nur die Datei ohne spürbare Steigerung der Qualität. Alles analog zum BPF-Wert, nur ein Stück genauer.

Bei zu wenigen Prozenten (unter 50 % dürfte es so richtig kritisch werden) müssen wir entweder Platz schaffen (kleinere Audiospuren, höhere Zielgröße) oder die Auflösung verringern. Bei zu hohen Werten können wir umgekehrt die Auflösung erhöhen oder Platz verschwenden. Vielleicht reicht es doch für den

Original-AC3-Sound oder zusätzlich die deutsche Tonspur (Ok, für die meisten eher zusätzlich die Originalsprache. Dann Untertitel nicht vergessen!). Wer noch auf CDs encodiert, kann im glücklichsten Fall sogar einen Datenträger einsparen.

Beim Ändern der Auflösung verliert der Prozent-Wert einen Teil seiner Aussagekraft, weil der Compcheck nur für ein ganz spezielles Encoding-Setup gilt, zu dem eben auch die Auflösung gehört. Genau genommen müssten wir also bei jeder Auflösungsänderung einen neuen Kompressionscheck durchführen. Da aber auch der Compcheck nur ein Schätzwert für die Qualität ist, fällt diese zusätzliche Ungenauigkeit kaum ins Gewicht.

Auch für anamorphe Encodings kann der Kompressionstest als Qualitätsindikator dienen. Allerdings müssen wir einige Abweichungen bedenken.

Um den Test durchzuführen, sollten wir immer das einfache **Save-avs**-Fenster benutzen, da die erweiterte Version die manuellen Änderungen am Skript nicht berücksichtigt. Außerdem stimmt die gewohnte Faustregel »60 – 80 %« für gute Qualität nicht mehr. Zumindest eine neue Untergrenze muss her. Ich hatte vor kurzem einen Film mit einem Compcheck-Wert von 35 %. Encodiert mit Xvid 1.1 ist das Ergebnis einwandfrei. In anspruchsvollen High-Motion-Szenen tauchen ein paar Blocks auf, die allerdings nur bei Standbildern sichtbar sind. Ich würde deshalb 40 – 45 % als neue sichere Untergrenze vorschlagen. Ob die Obergrenze auch angepasst werden muss und wie die Situation bei anderen Codecs aussieht, weiß ich nicht. Wer in der Hinsicht schon Erfahrung hat, kann mir gerne eine Mail schicken.

Haben wir schließlich und endlich passende Einstellungen gefunden, gelangen wir über den **Save & Encode**-Button im **Videofenster** wieder in den **Save-avs**-Dialog zurück und läuten mit dem dortigen **Save & Encode**-Button die letzte Konfigurationsphase vor dem Encoding ein. Gordian Knot speichert die AVS-Datei, die anschließend VirtualDubMod braucht, und springt dann in den Encoding-Dialog, wo wir die Einstellungen für die Ton und Bild vervollständigen.

B.4.1.6 Das Encoding starten

Es ist fast geschafft. Nur noch wenige Schritte trennen uns vom Start des Encodings. Was dieses Kapitel wieder einmal deutlich verlängert, sind die Spezialitäten für anamorphe Encodings.

Audiospuren einbinden

Mit **Select** (Abbildung 85) wählen wir die erste Audiospur aus. Gordian Knot hat hier schon automatisch die Datei eingetragen, die wir im Register **Bitrate** bei **Audio A** angegeben haben. Ist das eine AAC-Datei, müssen wir **Audio processing disabled** auswählen und uns später von Hand um die Tonspuren kümmern. Gordian Knot kann ja nicht mit AAC umgehen.

Bei allen anderen Audioformaten stellen wir auf **Just mux**, da wir das Transcoding schon mit BeSweet erledigt haben. Im Dateinamen steht ein DELAY-Wert. Der sagt uns, um wie viele Millisekunden die Spur nach vorne oder hinten verschoben werden muss, um synchron mit dem Video zu laufen. Dann, und *nur* dann, wenn wir diesen Wert in BeSweet noch *nicht* berücksichtigt haben, tragen wir bei **Delay** den ms-Wert ein und setzen für negative Delays den **negative**-Haken. Im Textfeld dann kein Minuszeichen angeben! Haben wir das Delay schon vorher berücksichtigt, ist der richtige Eintrag immer **0**, egal was Gordian Knot vielleicht automatisch vorschlägt.

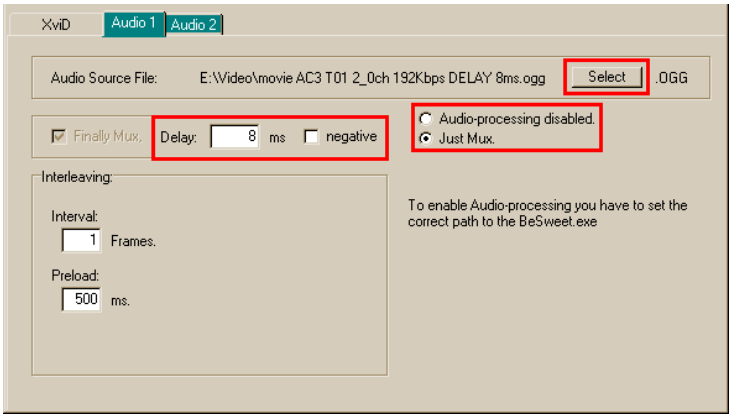

Abbildung 85: Konfiguration der Audiospuren.

Wenn wir eine zweite Audiospur haben, wechseln wir ins **Audio 2**-Register und wiederholen dort die Einstellungen für die zweite Spur.

Videocodec-Konfiguration anpassen

Die Audiospuren sind nun fertig und wir wechseln ins Register **XviD** bzw. **DivX 5** bzw. **x264**, um dem Codec den letzten Schliff zu verpassen.

Für Xvid (Abbildung 86) gibt es wenig zu tun. Das gilt auch für x264, denn dessen Dialog unterscheidet sich nicht wesentlich von Xvid. Über die Buttons **First Pass** und **Second Pass** können wir das Codec-Setup (bei Xvid einschließlich der Einstellungen für Credits) anpassen. Das wirkt sich nur auf den aktuellen Film aus. Die Standardeinstellungen im **Options**-Register bleiben davon unberührt.

Re-Calculate Bitrate sollte angehakt bleiben, um Problemen mit falschen Da-

teigrößen aus dem Weg zu gehen. Nötig ist die Option eigentlich nur dann, wenn Gordian Knot auch das Audio-Transcoding übernimmt, was wir ja schon vorher erledigt haben. Aber man kann nie wissen. Mit **Add Job to Encoding Queue** fügen wir den Film zur Encoding-Warteschlange hinzu, sofern wir kein anamorphes Bild haben.

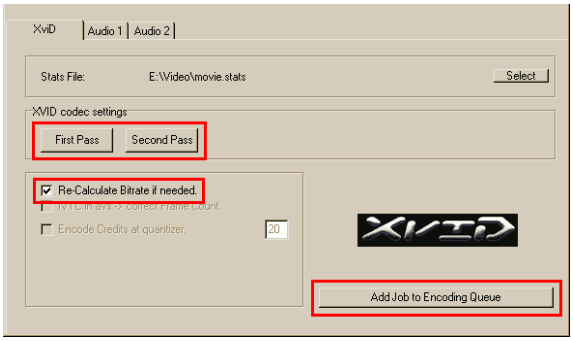

Abbildung 86: Abschließende Xvid-Konfiguration.

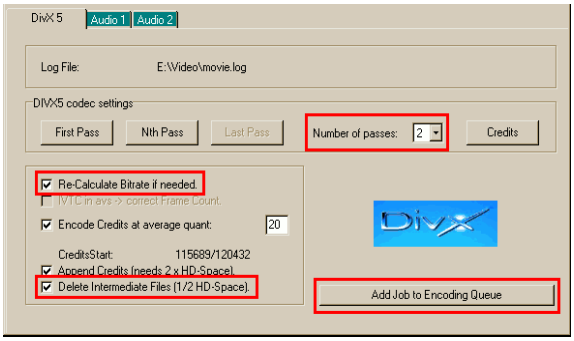

Abbildung 87: Abschließende DivX-Konfiguration.

Für DivX sieht der Dialog ähnlich aus (Abbildung 87). Mit **Number of passes** können wir auswählen, wie viele Passes Gordian Knot durchführen soll. Nicht übertreiben, mehr als drei dürfte meistens nicht nötig sein. Auch die Codec-Konfiguration können wir noch einmal anpassen.

Delete Intermediate Files löscht automatisch unnötige Dateien, die beim Credits-Encoding entstehen. Wer unter notorischen Platznot auf der Platte leidet, dürfte froh über diese Option sein. ;-)

AR-Flag für anamorphe Encodings setzen

Anamorphe Encodings ▸ A.3.2 sollten in ihrem MPEG-4-Videostream das AR-Flag auf den richtigen Wert gesetzt haben, damit das Video nicht mit Eierköpfen abgespielt wird. Das erledigen wir jetzt im Rahmen der abschließenden Encoder-Konfiguration. Wer Xvid als Encoder verwendet, hat es leicht, denn Xvid bringt AR-Unterstützung mit. Um das Seitenverhältnis zu setzen, müssen wir im Codec-Register von oben die Xvid-Konfiguration aufrufen und dort zum Register **Profile @ Level / More / Aspect Ratio** wechseln. Dort stellen wir – wie in der Xvid-VfW-Konfiguration in Kapitel A.4.1.1 beschrieben – das richtige PAR ein. Um Probleme zu vermeiden, sollten wir die Option für beide Passes setzen.

Wer lieber mit DivX encodiert, kann das AR-Flag nicht gleich beim Encoding setzen, sondern muss das hinterher mit MPEG4 Modifier (*moitah.net*) tun. Dieses Programm unterstützt bisher allerdings nur AVIs. Deshalb müssen wir entweder

gleich in Gordian Knot AVI verwenden oder die Datei nach dem Encoding mit VirtualDubMod nach AVI konvertieren. Dazu laden wir den fertigen Film und speichern ihn über **F7** wieder ab. Unten im Speichern-Dialog muss der **Video mode** auf **Direct Stream copy** stehen.

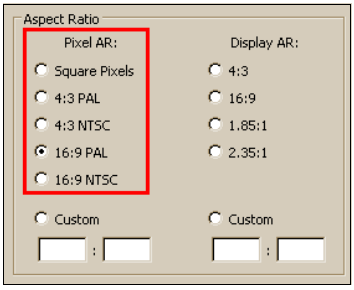

Abbildung 88: AR-Einstellungen von MPEG4 Modifier.

Dann öffnen wir die Datei mit MPEG4 Modifier (Abbildung 88) und setzen die **Pixel AR** wie oben bei Xvid beschrieben. Anschließen speichern wir das Video mit dem **Speichern**-Button. Da sich das AR-Flag im Header der Videospur befindet, kann die AVI problemlos nach Matroska oder MP4 konvertiert werden, ohne die AR-Info zu verlieren.

Auch x264 bietet die Möglichkeit, ein AR-Flag zu setzen; und zwar für das PAR. Die Funktion arbeitet genauso wie bei Xvid, setzt also das MPEG-4-AR-Flag in der Videospur auf das angegebene Verhältnis. Da x264 keine Standardwerte vorgibt, müssen wir einen Blick in Tabelle 4 auf Seite 44 werfen. Den passenden Wert geben wir im **More**-Register ein, wie in der x264-VfW-Konfiguration in Kapitel A.4.1.3 erklärt.

Start des Encoding-Vorgangs

Unglaublich aber wahr: wir sind endlich bereit zum Encoding. Über den Button **Add Job to Encoding Queue** fügen wir den Auftrag in die Warteschlange von Gordian Knot ein und landen anschließend im Register **Encoder**.

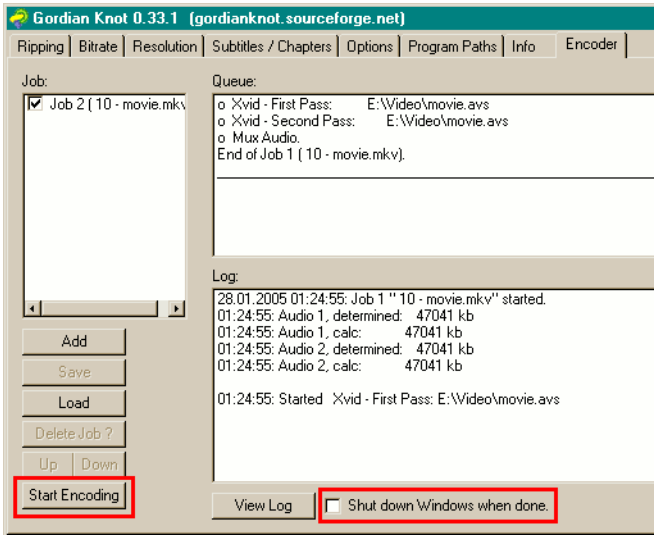

Abbildung 89: Encoder-Register von Gordian Knot.

Wenn wir Gordian Knots Nachfrage nicht schon bestätigt haben, starten wir über **Start Encoding** den Codiervorgang. Im **Queue**-Fenster sehen wir unseren aktuellen Auftrag, darunter bei **Log** sämtliche Details dazu. Gordian Knot ruft jetzt VirtualDubMod auf, um den Film zu encodieren und anschließend die Audiospuren einzubinden.

Solange VirtualDubMod arbeitet, war früher herrlich Zeit, mal wieder mit der Freundin auszugehen oder ein paar Stunden Schlaf

nachzuholen. Heutzutage kann man gerade noch in Ruhe ein Stück Kuchen vom Bäcker holen und Kaffee trinken … Naja – das ist vielleicht doch etwas übertrieben. Jedenfalls reicht es zum Ausspannen vor den Abschlussarbeiten: falls nötig den Film muxen und schnippeln, und natürlich brennen. Obwohl – auch brennen ist bei den heutigen Festplattengrößen vielleicht nicht mehr ganz so selbstverständlich.

B.4.1.7 Akapumas Gordian-Knot-Personalisierer

Akapumas Gordian-Knot-Personalisierer, was für ein Name! Viel zu lang, um nicht sofort eine Abkürzung zu kreieren, was akapuma praktischerweise gleich höchstselbst getan hat: *agkp*. Was genau es mit diesem Ding auf sich hat, dazu jetzt.

Was ist das?

Agkp ist eine von akapuma entwickelte externe Erweiterung von Gordian Knot mit drei Hauptfunktionen:

- Erweiterte AviSynth-Filtersteuerung.
- Unterstützung für x264 Cli ▸ A.4.2.
- Volle Matroskaunterstützung über MKVMerge ▸ A.2.3.

Wir werden uns nur mit der Unterstützung für x264 Cli und MKVMerge beschäftigen, denn das sind zweifellos die beiden wichtigsten Funktionen. Gordian Knot selbst verwendet ja nur den x264 VfW-Encoder ▸ A.4.1, der hässlich verbogene Streams erzeugt. Und VirtualDubMods Matroskafähigkeiten sind für heutige Verhältnisse rudimentär.

Agkp ist also dann interessant, wenn wir x264 und den Matroska-Container verwenden wollen. In diesem Fall rate ich sogar ganz ausdrücklich von reinem Gordian Knot ab, eben wegen der vergewaltigten Videostreams des VfW-Encoders.

Installation

Die aktuelle Version von agkp erhalten wir im ersten Posting des *Entwicklungsthreads* im Gleitz-Forum. Die Installation erfolgt in den VirtualDubMod-Ordner. Dafür muss vorher natürlich VirtualDubMod installiert sein. Wir entpacken das komplette agkp-Archiv in den VDubMod-Ordner. Dort treffen wir folgende Einstellung für alle drei Dateien *agkpv.exe, agkpa.exe* und *agkpAR.exe:* Mit einem Rechtsklick auf die Datei und dem Menüpunkt **Eigenschaften** gelangen wir in den Eigenschaftendialog aus Abbildung 90. Im Register **Programm** stellen wir sicher, dass **Nach Beenden schließen** angehakt ist, was unter Windows XP schon von Haus aus so sein sollte.

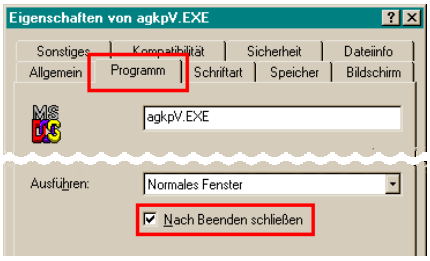

Abbildung 90: Dateieigenschaften der agkpV.exe.

Die Datei *agkp.ini* enthält die Pfade zu den benötigten externen Programmen. Die Datei können wir mit jedem Texteditor bearbeiten und dort eintragen, wo agkp *mkvmerge.exe* und *x264.exe* finden kann. Diese beiden Pfade sind wichtig und müssen immer angepasst werden. Der Player für Preview ist nur für agkps eigene Vorschaufunktion wichtig und MP4Box ist nur dann nötig, wenn wir trotz agkp den x264 VfW-Encoder verwenden. Dann ist MP4Box dafür zuständig, den verbogenen VfW-Stream in natives MPEG-4 zurückzuverwandeln, so dass das Video ordentlich in Matroska gemuxt werden kann.

Damit können wir Gordian Knot starten und ins Register **Program Paths** wechseln (Abbildung 91). Unter **Where is VirtualDubMod.exe located** stellen wir mit dem **Locate**-Button die *agkpV.exe* ein. Gordian Knot denkt also, dass es ganz normal VDubMod aufruft. Tatsächlich tritt aber agkp in Aktion. Entsprechend einfach ist es, trotz installiertem agkp ein klassisches GK-Encoding durchzuführen. Dazu ändern wir die VDubMod-Einstellung wieder zurück auf die *VirtualDubMod.exe.*

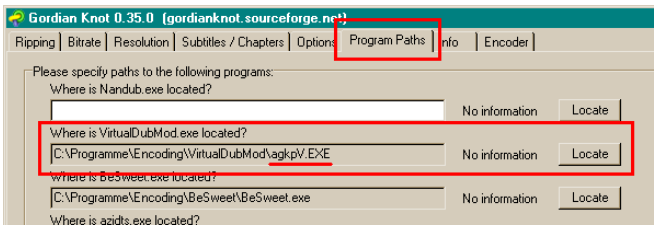

Abbildung 91: Einbinden von agkp in Gordian Knot.

Als letzter Punkt müssen wir darauf achten, dass der x264 VfW-Encoder installiert ist. Auch wenn wir ihn nicht benutzen, braucht ihn Gordian Knot, um das Skript für VDubMod korrekt erstellen zu können, das agkp dann auswertet. Welche Version des VfW-Encoders, ist egal. Er dient ausschließlich dazu, dass Gordian Knot nicht ins Schleudern gerät. Die x264.exe (der Cli-Encoder) sollte natürlich aktuell sein.

x264-Encoding mit agkp

Gordian Knot können wir fast wie gewohnt benutzen. Als Container stellen wir **MKV** ein und als Videocodec **x264**. Anschließend konfigurieren wir wie gewohnt das komplette Encoding. Wenn wir empfohlenerweise mit dem x264 Cli-Encoder encodieren, können wir uns die grafische Konfiguration des VfW-Encoders in Gordian Knot sparen. Agkp hat seine eigenen x264-Profile und berücksichtigt die VfW-Konfiguration nicht.

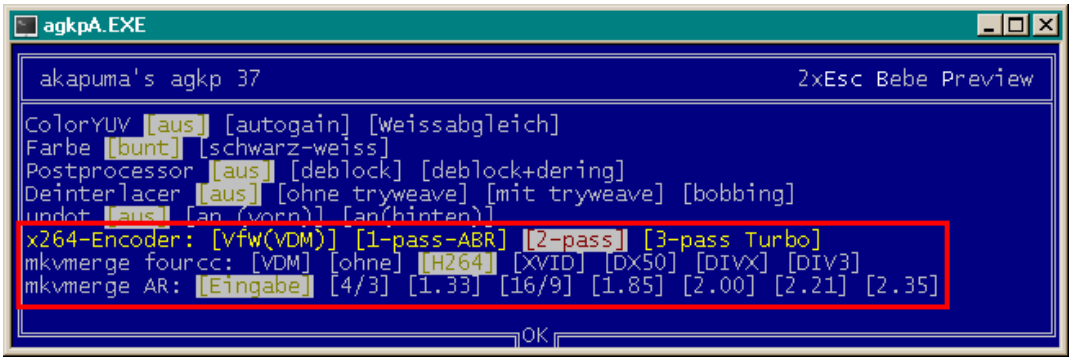

Abbildung 92: Agkp-Konfigurationsfenster.

Ist alles fertig eingestellt und das Encoding gestartet, erscheint das Auswahlfenster von agkp (Abbildung 92), das sogar die Maus unterstützt. :-) Die Einträge am Anfang betreffen die Filterkonfiguration, mit der wir uns hier nicht beschäftigen. Interessant sind die letzten drei Zeilen.

x264-Encoder

Hier wählen wir das x264-Encodingprofil. **VfW(VDM)** steht für die Gordian-Knot-Standardmethode über VirtualDubMod. Wenn wir uns dafür entscheiden, mussten wir vorher in Gordian Knot auch den Codec konfigurieren. Die üblichste Auswahl ist **2-pass**, was einem normalen 2-Pass-Encoding mit sinnvoll voreingestellten Codec-Parametern entspricht. Es spricht wenig dagegen, das zu übernehmen.

Mkvmerge fourcc

Hier geben wir den FourCC-Codec-ID-String an, der in der Matroskadatei gespeichert werden soll. Die Vorgabe **H264** können wir bedenkenlos stehen lassen.

Mkvmerge AR

Die Zeile ist für das Setzen des AR-Flags im Matroska-Container zuständig. Um das AR-Flag im MPEG-4-Stream kümmert sich agkp nicht – unschön. Deshalb empfehle ich anamorphe Encodings ▸ A.3.2 mit der agkp-Standardfunktionalität aus-

drücklich nicht. Das gilt nicht, wenn wir den VfW-Encoder verwenden, denn der setzt das MPEG-4-Flag wie von uns auf Seite 121 konfiguriert. Wie wir auch mit agkp ein anamorphes Encoding einschließlich MPEG-4-Flag erreichen, dazu mehr ab Seite 132.

Für ein klassisches nicht-anamorphes Encoding bleibt die AR-Angabe unverändert – so wie im Screenshot zu sehen.

Damit haben wir die Agkp-Konfiguration beendet und können das Encoding mit einem Klick auf den **OK**-Button unten in der Mitte starten. Im Fenster der *agkpV.exe* (Abbildung 95) lässt sich der Fortschritt verfolgen. Das Encoding läuft nun genauso automatisch ab wie von Gordian Knot gewohnt.

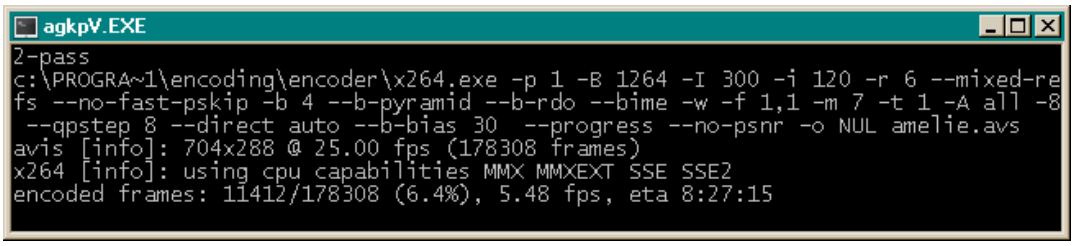

Abbildung 93: x264.exe bei der Arbeit.

Konfiguration: Die agkpx264.bat

Die Konfiguration von x264 erfolgt über die Datei *agkpx264.bat* im VirtualDubMod-Ordner. Das ist eine Batchdatei, wie sie mancher vielleicht noch aus guten alten DOS-Zeiten kennt. Da es sich um eine reine Textdatei handelt, können wir sie mit jedem Texteditor bearbeiten. Und genau das tun wir jetzt.

> Alle folgenden Zeilennummern gelten für die unveränderte *agkpx264.bat*, die agkp Version 37 mitbringt.

Bevor wir uns auf die Einzelheiten stürzen, hier ein paar grundlegende Dinge, die man über Batchdateien wissen sollte.

Kommentare

Alle Zeilen, die mit `REM` beginnen, sind Kommentare und werden komplett ignoriert.

Abschnitte

Eine Batch-Datei können wir mit Labels in Abschnitte einteilen. Ein Label steht allein auf einer Zeile und beginnt mit einem Doppelpunkt gefolgt vom Namen des Abschnitts. Als erstes begegnet uns in Zeile 28 `:compcheck`, das die Konfiguration für den Kompressionstest beinhaltet. Ein Abschnitt beginnt mit seinem Label und endet mit dem nächsten Label oder dem Ende der Datei.

Echo

Der Befehl `echo` *<Text>* zeigt *<Text>* am Bildschirm an. Er führt keine Befehle aus. Ein `echo format c:` würde also den Text »format c:« anzeigen, aber nicht die Festplatte formatieren.

Die erste interessante Zeile ist Zeile 9.

```
set subme=7
```

An dieser Stelle setzen wir den Parameter `--subme` der x264-Konfiguration zentral für die ganze Datei. In der VfW-Konfiguration heißt die Option **Partition Decision**. Mögliche Werte sind ganze Zahlen zwischen 1 und 7.

Encoder-Abschnitte

Weiter unten sind alle Abschnitte interessant, die mit `xname` beginnen. Jeder dieser Abschnitte steht für eine x264-Konfiguration, die wir auswählen können. Die folgenden Zeilen enthalten die Standardkonfiguration für ein 2-Pass-Encoding.

```
52 :xnametwopass
53 ECHO 2-pass
54 rem default
55 echo %2 -p 1 -B %3 -I 300 -i 120 -r 6 --mixed-refs ...
56 %2 -p 1 -B %3 -I 300 -i 120 -r 6 --mixed-refs ...
57 echo %2 -p 2 -B %3 -I 300 -i 120 -r 6 --mixed-refs ...
58 %2 -p 2 -B %3 -I 300 -i 120 -r 6 --mixed-refs ...
59 goto ende
```

Zeile 52

Der Labelname für eine Codec-Konfiguration muss immer mit `:xname` beginnen. Was dahinter folgt, ist egal. Es darf lediglich der selbe Name nicht mehrmals in der Batchdatei vorkommen. Erlaubt sind allerdings nicht alle Zeichen. Auf der

sicheren Seite sind wir, wenn wir uns auf die normalen Buchstaben des Alphabets von A bis Z beschränken – also keine Umlaute, Zahlen etc.

Zeile 53

Hier steht der Name, wie er als Option im Agkp-Auswahlfenster erscheint. Der `Echo`-Befehl gibt den Namen außerdem auch beim Encoding am Bildschirm aus.

Zeile 54

Diese Zeile identifiziert die Standardauswahl, die im Agkp-Auswahlfenster automatisch markiert wird. Nur ein einziger `xname`-Abschnitt darf diese Zeile enthalten.

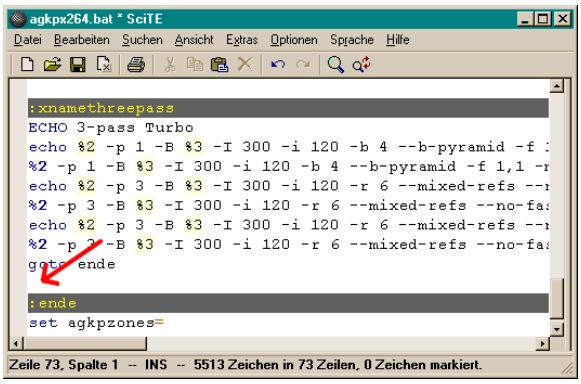

Abbildung 94: Position für neuen Encoder-Abschnitt in der agkp264.bat.

Zeilen 55, 56

Die beiden Zeilen enthalten die Konfiguration für den 1st Pass. In Zeile 55 wird zuerst mit `Echo` die Kommandozeile angezeigt. Zeile 56 startet dann das Encoding.

Zeilen 57, 58

Hier steht die Konfiguration für den 2nd Pass, die genauso wie Zeilen 55, 56 funktioniert. Für ein 3-Pass-Encoding würden wir einfach zwei weitere Zeilen für den dritten Durchgang hinzufügen, so wie wir es im vordefinierten `xnamethreepass`-Abschnitt schon sehen können.

Zeile 59

Das `goto ende` springt zum Ende der Datei. Ansonsten würden auch alle folgenden Abschnitte ausgeführt, der Film also mehrmals encodiert. Diese Zeile darf in keinem Abschnitt fehlen.

Einfügen eines neuen Abschnitts

Mit dem gerade besprochenen Schema können wir einen neuen Encoder-Abschnitt einfügen, womit uns eine zusätzliche Auswahlmöglichkeit fürs Encoding zur Verfügung steht. Die passende Stelle für die neue Konfiguration ist nach dem letzten `xname`-Abschnitt. Vor dem Marker `:ende`, da wohin der Pfeil in Abbildung 94 zeigt, fügen wir folgende Zeilen ein:

```
:xnameencwissen
echo Encwissen
echo %2 -B %3 -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto
--b-pyramid -w --bime -m %subme% -A i4x4,p8x8,p4x4,b8x8 %agkpzones%
--no-psnr --progress -p 1 -o NUL %5
%2 -B %3 -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto --b-pyramid -w
--bime -m %subme% -A i4x4,p8x8,p4x4,b8x8 %agkpzones% --no-psnr --progress
-p 1 -o NUL %5
echo %2 -B %3 -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto
--b-pyramid -w --bime -m %subme% -A i4x4,p8x8,p4x4,b8x8 %agkpzones%
--no-psnr --progress -p 3 -o %4 %5
%2 -B %3 -r 5 --mixed-refs -t 1 -f 0:0 -b 3 --direct auto --b-pyramid -w
--bime -m %subme% -A i4x4,p8x8,p4x4,b8x8 %agkpzones% --no-psnr --progress
-p 3 -o %4 %5
goto ende
```

Die Konfiguration entspricht der aus dem x264 Cli-Kapitel. Wichtig sind die fett markierten Bestandteile.

%2

Platzhalter für den Pfad zur *x264.exe,* wie wir ihn in der Datei *agkp.ini* konfiguriert haben.

%3

Diese Variable ersetzt agkp mit der von Gordian Knot errechneten Bitrate.

%4

Steht für die Zieldatei, also das encodierte Video ohne Ton usw.

%5

Steht für die Quelldatei, was dem AviSynth-Skript entspricht.

-m %subme%

Damit die Einstellung, die wir ganz oben in der Zeile `set subme` getroffen haben, berücksichtigt wird, muss diese Option in der Kommandozeile auftauchen.

%agkpzones%

Agkp erkennt automatisch, ob wir in Gordian Knot separat encodierte Credits gewählt haben. Um diese Einstellung auch beim Encoding zu verwenden, muss `%agkpzones%` in der Kommandozeile stehen.

Im Auswahlfenster von agkp (Abbildung 95) haben wir nun eine zusätzliche Option namens **Encwissen**, mit der wir die neue Konfiguration verwenden können.

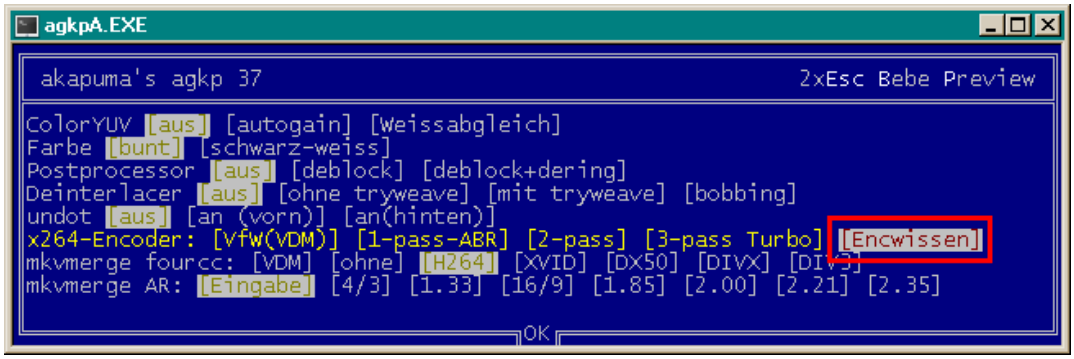

Abbildung 95: Neue Option im agkp-Auswahlfenster.

Xvid-Encoding mit agkp

Ehrlich gesagt wundert es mich ein bisschen, dass noch niemand auf die dumme Idee gekommen ist, agkp auszutricksen und anstelle von x264 den Xvid-Kommandozeilenencoder zu verwenden. Das funktioniert nämlich prächtig. Einzige Einschränkung: Zonen (und damit Credits-Encoding) können wir nur manuell konfigurieren, nicht in Gordian Knot, da Xvid eine andere Syntax benutzt als x264.

Der Xvid Cli-Encoder heißt Xvid_Encraw und besteht aus einer einzelnen Datei: *xvid_encraw.exe*. Damit das Encoding funktioniert, muss aber zusätzlich das ganz normale Xvid-Paket installiert sein! Die *xvid_encraw.exe* können wir in einen beliebigen Ordner kopieren.

Der Ort für die Xvid-Konfiguration ist die bekannte Datei *agkpx264.bat* im VirtualDubMod-Ordner. Dort fügen wir wie oben einen neuen Encoder-Abschnitt hinzu, diesmal allerdings für Xvid_Encraw. Den fett markierten Pfad zur *xvid_encraw.exe* müssen wir anpassen, je nachdem, wo wir die Datei abgelegt haben.

```
:xnamexvid
echo Xvid ;-)
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -vbvsize 1833216
-vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2 -bquant_ratio 163
-bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10 -zones 0,w,1,0
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -vbvsize 1833216
-vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2 -bquant_ratio 163
-bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10 -zones 0,w,1,0
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2
-bitrate %3 -vbvsize 1833216 -vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2
-bquant_ratio 163 -bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10
-zones 0,w,1,0
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2 -bitrate %3
-vbvsize 1833216 -vbvmax 8000 -vhqmode 4 -qpel -max_bframes 2
-bquant_ratio 163 -bquant_offset 0 -bvhq -nopacked -qtype 1 -progress 10
-zones 0,w,1,0
goto ende
```

Die Konfiguration ist genau die, wie sie in der Xvid-VfW-Konfiguration ▸ A.4.1.1 erklärt ist. Prinzipiell handelt es sich um die leicht angepassten Kommandozeilen aus dem Xvid-Cli-Kapitel.

Mehr Veränderungen sind an der *agkpx264.bat* nicht nötig. Wir können die Datei speichern und Gordian Knot laden. Dort tun wir so, als hätten wir es mit einem x264-Encoding zu tun. Denn prinzipiell haben wir gerade nur ein neues x264-Profil für agkp erstellt. Nur, dass dieses Profil eben gar nicht den x264-Encoder verwendet. Das heißt, wir wählen natürlich **MKV** als Container und wie vorhin **x264** als Encoder, ohne allerdings an dessen Konfiguration Zeit zu verschwenden. Xvid kommt erst später ins Spiel.

Der Kompressionstest wird trotz Trickserei weiterhin mit x264 durchgeführt, liefert also ein nutzloses Ergebnis. Um trotzdem auf den Test nicht verzichten zu müssen, können wir erst wie gewohnt **XviD** als Codec wählen, den Test durchführen und dann auf **x264** mit agkp umschalten.

Wie auch für Xvid VfW sollte die Option **Calculate Frame-Overhead** im **Bitrate**-Register deaktiviert sein.

Haben wir alles fertig konfiguriert und das Encoding gestartet, erscheint das nun schon bekannte Agkp-Auswahlfenster. Die Zeile **x264-Encoder** hat wie in Abbildung 96 einen neuen Eintrag bekommen, hinter dem sich unsere Xvid-Konfiguration verbirgt. Natürlich können wir uns an dieser Stelle noch anders entscheiden und auf eines der x264-Profile umstellen. Im Xvid-Fall ist es sinnvoll, auch den **mkvmerge fourcc** auf **XVID** zu ändern. Für **mkvmerge AR** gilt das gleiche wie im x264-Abschnitt auf Seite 125. Dann fehlt nur noch ein Klick auf **OK** und ein paar Stunden Geduld. Und dann soll noch einer sagen, Gordian Knot käme mit aktueller Encodingtechnologie nicht zurecht! :-)

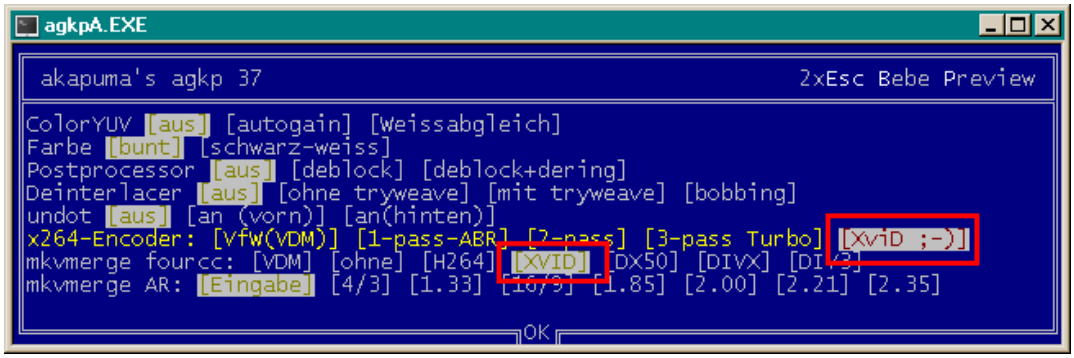

Abbildung 96: Agkp-Auswahlfenster mit Xvid-Hack.

Anamorphes Encoding mit agkp

Anamorphe Encodings erfordern schon mit Gordian Knot an sich ein paar Tricks. Und auch agkp bietet ausschließlich die Möglichkeit, das AR-Flag im Matroska-Container zu setzen. Da sowohl x264.exe als auch Xvid_Encraw das Setzen des MPEG-4-AR-Flags unterstützen, können wir die fehlende Funktion über die *agkpx264.bat* nachrüsten.

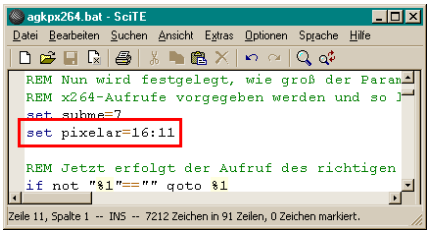

Abbildung 97:
PAR-Variable in der agkpx264.bat.

Dafür öffnen wir die *agkpx264.bat* und definieren wie in Abbildung 97 nach der `subme`-Zeile (Zeile 9 in der unveränderten Datei) eine neue Variable. Für alle Befehle weiter unten in der Datei existiert nun eine neue Variable, die wir über `%pixelar%` ansprechen können. Darin enthalten ist der PAR, der ins MPEG-4-Flag geschrieben werden soll. Die `16:11` im Screenshot bezeichnen eine 16:9-PAL-DVD. Welches für den jeweiligen Film der richtige Wert ist, können wir der inzwischen sicher schon wohlbekannten PAR-Tabelle auf Seite 44 entnehmen. Für ein klassisches nicht-anamorphes Encoding setzen wir die Variable auf `1:1`.

Nun müssen wir die Encoder-Abschnitte anpassen. Für x264 fügen wir *jeder* Kommandozeile die Option `--sar %pixelar%` (doppelter Bindestrich!) hinzu, z. B.:

```
:xnameencwissen
echo Encwissen
echo %2 -B %3 --sar %pixelar% -r 5 --mixed-refs ...
%2 -B %3 --sar %pixelar% -r 5 --mixed-refs ...
echo %2 -B %3 --sar %pixelar% -r 5 --mixed-refs ...
%2 -B %3 --sar %pixelar% -r 5 --mixed-refs ...
goto ende
```

Für Xvid_Encraw heißt die Option `-par %pixelar%` (einfacher Bindestrich!):

```
:xnamexvid
echo Xvid ;-)
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -par %pixelar%
-vbvsize 1833216 ...
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -par %pixelar%
-vbvsize 1833216 ...
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2
-par %pixelar% -bitrate %3 ...
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2 -par %pixelar%
-bitrate %3 ...
goto ende
```

Schließlich löschen wir ganz am Ende der Datei die PAR-Variable, indem wir diese neue Zeile einfügen:

```
set pixelar=
```

Damit sind wir bereit zum Encoding. Gordian Knot konfigurieren wir wie gewohnt. Bevor wir den Encoding-Job starten – also bevor das agkp-Auswahlfenster erscheint – müssen wir, falls nötig, in der *agkpx264.bat* in der Zeile `set pixelar` den richtigen Wert für den jeweiligen Film eintragen. Dann erhalten wir ein Encoding mit gesetztem MPEG-4-AR-Flag.

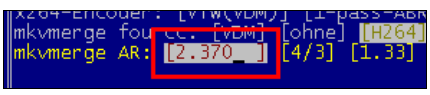

Abbildung 98: DAR-Angabe für das Matroska-AR-Flag.

Im agkp-Auswahlfenster setzen wir wie in Abbildung 98 zusätzlich das Seitenverhältnis im Matroska-

Container. Dazu klicken wir in der Zeile **mkvmerge AR** auf **Eingabe**, setzen den passenden Wert ein (mit dem Punkt als Dezimaltrenner, nicht dem Komma) und bestätigen mit einem Druck auf die **Enter**-Taste.

Achtung! Matroska arbeitet nicht mit dem PAR, sondern mit dem DAR! Der richtige Wert ist der gleiche, den wir in MKVMerge GUI unter **Aspect Ratio** eintragen müssten. Details dazu stehen im Matroska-Muxing-Kapitel B.5.1.

Zum Schluss startet ein Klick auf **OK** das Encoding. Das Ergebnis ist trotz des VfW-zentrierten Gordian Knot eine anständig gemuxte Matroska-Datei mit beiden AR-Flags und einem nativen MPEG-4-Videostream. An der Stelle haben wir uns dann wieder einmal einen Kaffee verdient. :-)

B.5 Manuelles Muxing

In diesem Abschnitt beschäftigen wir uns mit dem Muxing – also dem Zusammenfügen aller Einzelteile zum fertigen Film. In der Regel erfüllt diese Aufgabe das Encoding-Frontend gleich mit, allerdings unterstützt kein Frontend sämtliche Spezialitäten sämtlicher Container. Deswegen kann es schnell nötig werden, selbst Hand anzulegen.

Im Moment ist der ganze Abschnitt noch in einem recht rudimentären Zustand und behandelt wenig mehr als in Grundzügen AAC in Matroska und AVI. Je nach Lust und Laune werde ich an dieser Stelle in Zukunft kräftig erweitern.

B.5.1 MKVMerge für den Matroska-Container

MKVToolnix ist ein Toolpaket für den Matroska-Container. Für uns ist das interessant, wenn wir uns für Matroska und AAC-Audio entschieden haben, denn VirtualDubMod ist nicht AAC-fähig. In dem Fall konnten wir auch die Tonspur(en) nicht schon mit Gordian Knot zum Film dazumuxen. Wir haben also einen fertigen Film, der nur das Video enthält, und separat dazu AAC-Ton und evtl. Untertitel und Kapitel.

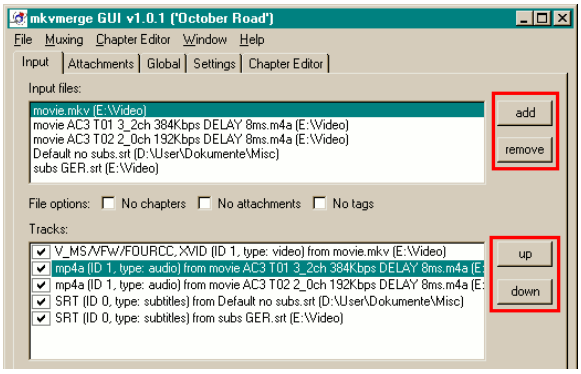

Abbildung 99: Hinzufügen von Quellen in MKVMerge GUI.

Das Programm aus MKVToolnix, um alles zusammenzufügen, heißt MKVMerge und ist genauso wie BeSweet eine Konsolenanwendung. In Abbildung 99 sehen wir die grafische Oberfläche dazu, die sich sich MKVMerge GUI (mmg) nennt. Die starten wir. Mit **Add** oder per Drag & Drop können wir alle benötigten Dateien hinzufügen, im Bild sind das die von GKnot erzeugte Videodatei, zwei AAC-Audiospuren und zwei SubRip-Untertitel. Die Spuren sind am Ende in der gleichen Reihenfolge in der Datei gespeichert wie sie hier erscheinen. Mit **Up** und **Down** können wir Spuren verschieben. Im unteren Teil des Fensters (Abbildung 100) konfigurieren wir die einzelnen Spuren, und zwar immer gerade diejenige, die im **Tracks**-Fenster angewählt ist.

Unter **Language** stellen wir für Audios und Untertitel die passende Sprache ein. Die erscheint dann beim Abspielen auch wieder im Player.

Delay hatten wir schon einmal bei BeSweet angesprochen. Um diese Anzahl Millisekunden muss die Audiospur verschoben werden, um synchron zum Video zu laufen. Den Wert aus dem Dateinamen (**DELAY 8ms**) übertragen wir incl. Vorzeichen in das **Delay**-Feld. Wichtig: Wenn wir am Anfang bei BeSweet das Delay schon einmal berücksichtigt haben, gilt jetzt *immer* ein Delay von **0**.

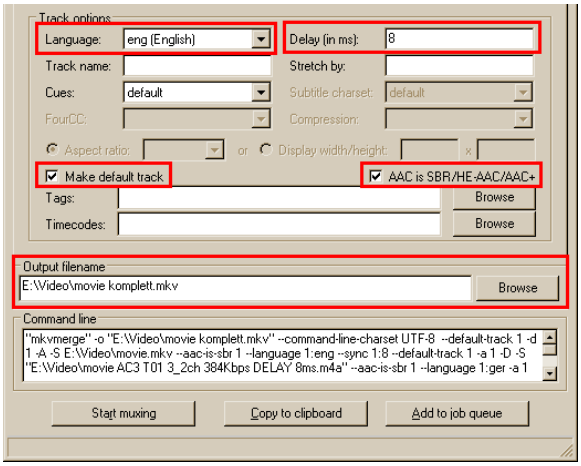

Abbildung 100: Audio-Konfiguration in mmg.

Den Haken bei **AAC is SBR...** müssen wir setzen, wenn unsere AAC-Tonspuren den High-Efficiency-Modus (HE) verwenden und *nicht* im MP4-Container verpackt sind. Dann kann MKVMerge nämlich das Vorhandensein von HE nicht automatisch erkennen. Auf der sicheren Seite sind wir, wenn wir die Option immer richtig einstellen, egal ob MP4 im Spiel ist oder nicht. Ganz unten geben wir noch den Namen der Ausgabedatei an.

Für anamorphe Encodings ▸ A.3.2 sollten wir auch das AR-Flag des Containers setzen. Vorsicht dabei! Matroska verlangt nicht – wie ansonsten immer – das PAR, sondern das DAR. Ein wenig rechnen bleibt uns deshalb nicht erspart.

MKVMerge bietet zwei Möglichkeiten, um das DAR zu setzen. Die einfachere verbirgt sich hinter der Option **Display width/height**. In die beiden Felder daneben setzen wir die korrekt entzerrte Wiedergabeauflösung ein. Da sich die vertikale Auflösung nicht ändert, brauchen wir im zweiten Feld nur die Bildhöhe nach dem Cropping eintragen. Die entzerrte horizontale Auflösung berechnen wir – erinnern wir uns an den Anamorph-Abschnitt – mit dieser Formel:

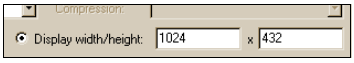

Abbildung 101: AR-Eingabe in Form der Auflösung.

$$\text{Zielbreite} = \text{Breite nach Cropping} \times \text{PAR}$$

Die PAR-Werte stammen wie immer aus Tabelle 4 auf Seite 44. Für den alten Beispielfilm *Die wunderbare Welt der Amélie* ergäbe sich:

$$\text{Zielbreite} = 704 \times {}^{16}\!/_{11} = 1024$$

Entsprechend müssten wir wie im Screenshot in die beiden Felder **1024** und **432** eintragen. Die zweite Möglichkeit, den DAR anzugeben, ist die Option **Aspect Ratio** (Abbildung 102).

Falls wir Gordian Knot mit agkp verwenden, müssen wir uns beim AR-Flag an diese Methode halten.

Im Eingabefeld steht das DAR als einfache Fließkommazahl. Um diesen Wert zu ermitteln, benötigen wir zuerst die korrekt entzerrte Wiedergabeauflösung, so wie wir sie für die erste Möglichkeit berechnet haben. Das DAR ergibt sich dann entsprechend seiner Definition als:

DAR = Zielbreite / Zielhöhe

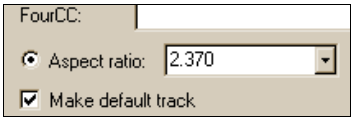

Abbildung 102: AR-Eingabe in Form des DAR.

Für unser *Amélie*-Beispiel bedeutet das

DAR = $^{1024}/_{432}$ = 2,370370

Auf mehr als zwei oder drei Stellen zu runden, ist Unsinn, da sich nichts mehr am Ergebnis ändern würde. Ein Bildschirm kann eben nur ganze Pixel darstellen. Beim Eintippen müssen wir darauf achten, dass MKVMerge den Punkt als Dezimaltrenner erwartet, nicht das Komma.

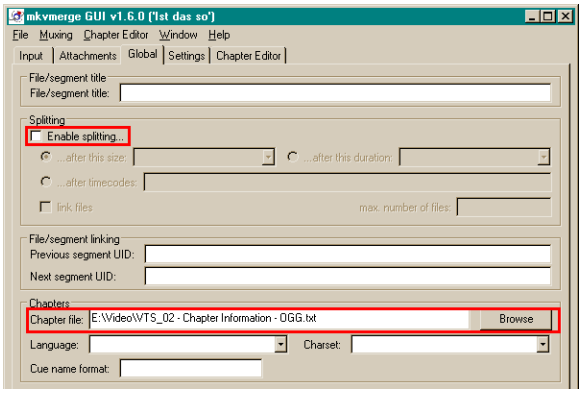

Abbildung 103: Globale Matroska-Einstellungen in mmg.

Damit sind die Einstellungen im **Input**-Register abgeschlossen, und wir wechseln zu **Global**. **Enable splitting** bleibt auf jeden Fall erst einmal aus, da wir noch nicht wissen, wo der Film eventuell geteilt werden muss. Unter **Chapter file** können wir die Datei mit den Kapitelinformationen angeben, die wir ganz am Anfang mit Chapter-X-tractor erstellt haben. **File/segment title** gibt dem Film einen Namen (hat nichts mit dem Dateinamen zu tun), dann starten wir ganz links unten mit **Start muxing** den Muxing-Vorgang. Der dauert ein paar Minuten, dann liegt der Film komplett fertig in einem Stück auf der Platte und muss bei Multi-CD-Encodings noch gesplittet werden. MKVMerge GUI können wir solange offen lassen.

B.5.2 AVI-Mux GUI für den AVI-Container

Und nochmal AAC. Ja, VirtualDubMods Unfähigkeit, das neue Audioformat zu verarbeiten, kann einem ganz schön zu schaffen machen. Da hilft nur selbst Hand anzulegen, das haben wir weiter oben ja schon gemerkt. Nach der ausführlichen Anleitung für AAC in MKV fällt dieses Kapitel aber etwas schmaler aus. Ich bin nun mal überzeugter Matroska-User und habe daher mit AVI nicht mehr besonders viel zu tun.

Von allen Vorlieben abgesehen, ist dieses Kapitel in der aktuellen Form schon ein bisschen lächerlich. Irgendwann soll hier mal ein etwas allgemeineres und umfangreicheres Tutorial zu AVI-Mux GUI stehen. Wann? Bald™.

AAC in AVI ist ein ähnlicher Trick (manche sagen auch »dreckiger Hack« dazu) wie VBR MP3 in AVI. Man sollte sich also mit dem Gedanken anfreunden, dass es funktionieren kann und sehr wahrscheinlich auch funktionieren wird, man aber nie eine Garantie bekommt.

Das einzige Programm, das AAC in AVI muxen kann, ist AVI-Mux GUI, das wir auf *alexander-noe.com* finden können. Das versteht leider keine MP4-AACs, sondern möchte sie gern im ADTS-Format haben. Eine Umwandlung (eigentlich ist es ein reines Demuxing) ist mit MP4Box möglich.

Ein guter Einstiegspunkt für weitere Informationen und Links ist die *Audio-FAQ* des Doom9.org-Forums.

Teil C
Spezialthemen

Einleitung

In den Spezialthemen geht es um Dinge, die an anderer Stelle nicht passen, die nur unter ungewöhnlichen Voraussetzungen gebraucht werden oder die noch zu neu und zu experimentell sind, um für das alltägliche DVD-Backup empfehlenswert zu sein. Hier gibt es auch keine Reihenfolge wie sonst im Encodingwissen.

C.1 Manuelles Splitting

Ein Hoch auf den DVD-Brenner! Seit ich so ein Ding besitze, fällt die lästigste Arbeit beim ganzen DVD-Backup endlich weg. Vom gesteigerten Komfort beim Anschauen und dank massig Platz der besseren Qualität mal abgesehen. Trotzdem: nicht jeder kann oder will sich einen DVD-Brenner leisten oder ist mit der Qualität eines 1-CD-Encodings zufrieden. Deshalb bleibt das Splitting ein wichtiger Teil der Abschlussarbeiten.

Grundsätzliches zum Splitting

2-CD-Encodings müssen wir nach dem Encoding noch auf die CDs aufteilen. Manchmal sind vielleicht auch drei CDs nötig, dann müssen wir eben einmal mehr splitten. Vom Vorgehen her macht das keinen Unterschied.

Da keyframe-genaues manuelles Splitting ohne VirtualDubMod schwierig, wenn überhaupt sinnvoll möglich, ist, beschäftigen wir uns in diesem Kapitel nur mit der VDub-Methode. Das heißt, sobald wir VDub nicht verwenden können – native MPEG-4-Streams und MP4 dürften die üblichsten Gründe sein – müssen wir aufs Splitting verzichten oder uns auf die Automatiken der Muxing-Tools verlassen. Davon abgesehen: wenn AAC-Audio im Spiel ist, sollten wir zuerst Kapitel B.5 übers manuelle Muxing gelesen haben.

Beim Aufteilen auf die CDs gibt es zwei Dinge zu beachten. Der Film muss ***immer* an einem Keyframe geteilt** werden, sonst entstehen am Anfang der zweiten Datei Bildfehler. Das MPEG-4-Format (Xvid/DivX/x264 sind nichts anderes) speichert keine kompletten Bilder, sondern nur die Veränderung zum vorhergehenden Bild. Allein die Keyframes stellen vollständige Einzelbilder dar ▸ A.1.2.1. Fehlt am Anfang der zweiten CD das Keyframe, fehlen dem Player erst einmal die Informationen, um ein komplettes Bild darzustellen. Das äußert sich in meist grünen Flächen anstatt der fehlenden Bildinformation und sieht natürlich gewaltig hässlich aus.

Ein sinnvoller Schnitt erfolgt bei einem **Szenenwechsel.** Es ist unheimlich lästig, wenn der Held der Geschichte mitten im Satz unterbrochen wird, weil es Zeit für die zweite CD wird. Solche Schnitzer geben auch dem professionellst codierten Film einen laienhaften Beigeschmack. Eine Einschränkung muss ich gelten

lassen. Wenn der Film gerade so auf zwei CDs passt, ist kaum Spielraum zum Splitten vorhanden. Deswegen ist es manchmal doch nötig, den Schnitt innerhalb einer Szene zu setzen. Ein bisschen Fingerspitzengefühl kann dabei aber auch nicht schaden. Vor allem nie, nie mitten im Satz schneiden!

VirtualDubMod als Splitting-Zentrale

Zuständig für das Aufteilen ist entweder komplett VirtualDubMod oder – bei dynamischen Vobsubs oder AAC-Audio – AVI-Mux GUI bzw. MKVMerge. Für diese beiden Tools suchen wir dann mit VirtualDubMod nur eine passende Stelle zum Schneiden, anstatt mit VDub auch gleich die einzelnen Dateien zu erzeugen.

Haben wir x264 verwendet, muss ein passender Decoder installiert sein, denn x264 ist ein reiner Encoder. Eine aktuelle Version von ffdshow eignet sich dazu bestens. Beim Setup müssen wir nur darauf achten, auch das VfW-Interface zu installieren.

Wir laden den fertigen Film (die Datei incl. Audio!) über **File / Load Video file** und stellen im **Video**-Menü auf **Direct stream copy** um. In diesem Modus kopiert VDubMod den Film nur und encodiert nicht noch einmal neu. Dann springen wir 700 MB weit in den Film hinein (bei Rohlingen anderer Größe natürlich den entsprechenden Wert nehmen). Das funktioniert per **Strg+Shift+J** oder **Edit / Go to last Keyframe**. VirtualDubMod hüpft zum gewünschten Keyframe.

Abbildung 104 zeigt die benötigten Kontrollelemente. Mit den **Keyframe-Buttons** (2) können wir keyframe-weise durch den Film springen, um eine günstige Stelle zum Schneiden auszusuchen. Vorwärts springen ist dabei nur bedingt sinnvoll, schließlich ist da die CD zu Ende. Die genaue Dateiposition sehen wir unter (4). So wissen wir exakt, bei wie viel Megabyte wir uns befinden (fürs Splitting über MKVMerge ist das besonders wichtig). die Exaktheit bezieht sich leider nicht auf AVIs mit VBR-MP3-Audio. Bei solchen Dateien müssen wir ca. 5 MB dazurechnen, um auf den richtigen Wert zu kommen.

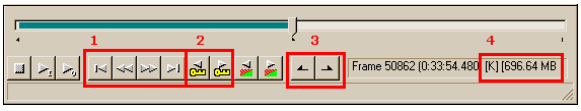

Abbildung 104:
Video-Kontrollbuttons in VirtualDubMod.

Haben wir eine passende Stelle gefunden, setzen wir dort den **Auswahl-Ende-Marker** (bei (3) der rechte Button). Dann springen wir mit dem linken Button bei (1) zum Anfang des Films und setzen hier den **Auswahl-Anfang-Marker** (bei (3) der linke Button). In der Zeitleiste ist jetzt der vordere Teil des Films markiert.

Jetzt ist auch der Zeitpunkt, an dem wir spätestens über **Streams / Stream List / Add** evtl. zusätzliche Untertitelspuren hinzufügen sollten. Anschließend speichern wir über **F7**. VirtualDubMod sichert immer nur den markierten Bereich des Films, so dass wir eine Datei erhalten, die genau auf die erste CD passen sollte.

Xvid (genau genommen eigentlich VfW) hält ein Hindernis bereit, wenn wir B-Frames verwenden. Dann erscheint die Meldung *B-Frame Decoder Lag* und wir bekommen erst beim ersten Drücken eines **Keyframe-Buttons** ein Bild. Dumm daran ist, dass das angezeigt Bild dann nicht unbedingt zur angesprungenen Framenummer passt, wir wissen also nicht exakt, wo wir uns im Film befinden. Abhilfe schafft nur ausprobieren: Schnittpunkt suchen, dort den **Anfangsmarker** setzen und ein kurzes Stück weiter hinten das Ende markieren. Dieses Ministück Film dann abspeichern und im Software-Player anschauen. Dort sehen wir, wie groß die Abweichung war. Meistens liegen wir nur ein oder zwei Keyframes daneben, es artet also kaum in endloses Probieren aus.

Über **Edit / Move to Selection End** springen wir zurück zum Ende der ersten CD und setzen jetzt hier den A**uswahl-Anfang-Marker**. Ein Sprung zum Filmende mit dem rechten Button von (1) und hier den **Auswahl-Ende-Marker** setzen. Dann können wir die zweite CD speichern (mehr CDs brauchen natürlich entsprechend mehr Schnitte).

Splitting mit MKVMerge

Den fertigen Film laden wir in VirtualDubMod und suchen wie gerade beschrieben einen passenden Punkt zum Schneiden heraus. Das Schneiden selbst muss dann MKVMerge übernehmen. Wir merken uns den Zeitpunkt, an dem wir splitten wollen (im Beispiel **00:33:54.480**) und tragen diesen wie in Abbildung 105 in MKVMerge GUI ein.

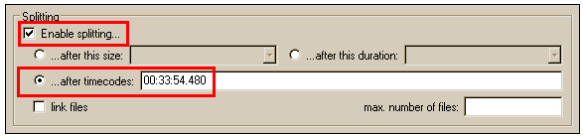

Abbildung 105: Splitting-Kontrollen in MKVMerge.

Enable splitting muss angehakt sein, um die Splittingfunktion überhaupt zu aktivieren. Der richtige Modus ist **after timecodes**. Im Textfeld dahinter tragen wir den Zeitpunkt in folgenden Format ein: **HH:MM:SS.NNN**, d. h. Stunden, Minuten und Sekunden jeweils durch Doppelpunkt getrennt und zweistellig (eine Stunde wäre also **01**), dahinter durch einen einfachen Punkt getrennt dreistellig die Sekundenbruchteile. Prinzipiell ist das das gleiche Format wie in VirtualDubMod.

Alle anderen Einstellungen bleiben wie am Anfang, das heißt besonders, dass

wir *nicht* den komplett gemuxten Film splitten, sondern die endgültigen Teildateien aus den einzelnen Streams zusammensetzen. Mit **Start muxing** erzeugen wir die endgültigen Dateien. Damit ist der Film komplett fertig.

Für drei oder mehr CDs erweitern wir einfach die Angabe unter **after timecodes**. Und zwar suchen wir mit VirtualDubMod alle Splitpunkte heraus und geben die durch ein Komma getrennt ein. Wenn wir den Beispielfilm von oben also zusätzlich noch bei 1 Stunde und 520 Millisekunden teilen wollten, um drei Dateien zu erhalten, würde die Eingabe bei **after timecodes** lauten: **00:33:54.480,01:00:00.520**.

Splitting mit AVI-Mux GUI

Das Splitting mit AVI-Mux GUI läuft ähnlich ab wie mit MKVMerge. Wir benutzen VirtualDubMod, um die Zeitmarken der Splitpunkte herauszufinden, die wir in AVI-Mux GUI dann eintragen.

Dafür öffnen wir in AVI-Mux GUI über den **Settings**-Button den Einstellungsdialog. Wir benötigen den **General**-Abschnitt der **Output**-Settings. Dort klicken wir unter **Split** den **Advanced**-Button (Abbildung 107). Im folgenden Fenster (Abbildung 106) tragen wir im markierten Feld die Zeitmarke des Splitpunkts ein und fügen sie mit dem Plus-Button zur Liste hinzu. Das Format ist das gleiche wie in VDub und MKVMerge. Wenn wir mehr als zwei CDs benötigen, fügen wir ganz einfach weitere Splitpunkte hinzu. Abschließend bestätigen wir sämtliche Dialoge und starten über den Start-Button das Muxing.

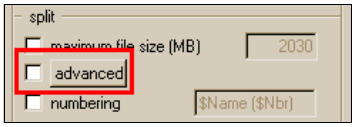

Abbildung 107: Splitting aktivieren in AVI-Mux GUI.

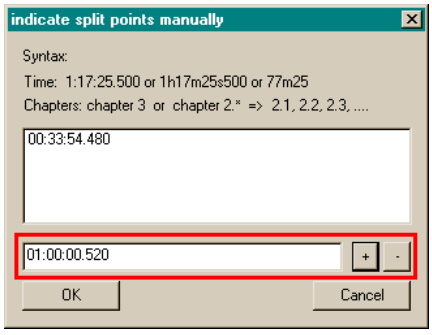

Abbildung 106: Splitpunkte setzen.

Das war's. Der Film ist fertig und bereit zum Brennen. Ein Test vorher, ob alles synchron ist und sämtliche Untertitelspuren, Kapitel etc. vorhanden sind, kann natürlich nicht schaden.

C.2 DivX 5.2.1 VfW-Konfiguration

Gordian Knot hat Probleme mit der 6er-Generation von DivX. Also sehen wir uns auch die Konfiguration von DivX 5.2.1 im Detail an, denn das ist die letzte Version, die garantiert problemlos mit Gordian Knot funktioniert.
DivX.com bietet die alte Version nicht mehr an. Wir können sie aber noch von *http://www.divx-digest.com/software/divxcodec5.html* herunterladen.

Einstellungen für den 1st Pass

Zuerst rufen wir über den **Select Divx Certified Profile**-Button den Profilassistenten auf. Um volle Kontrolle über alle Optionen zu erhalten, müssen wir wie in Abbildung 108 den Haken bei **DivX Certified** entfernen und damit die Profile deaktivieren. Dann klicken wir uns auf die nächste Seite weiter.

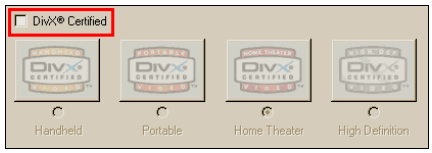

Abbildung 108: DivX-Profilauswahl.

Was die drei Optionen in Abbildung 109 im Detail bedeuten, haben wir schon bei der Xvid-Konfiguration gesehen. Bei DivX arbeiten sie prinzipiell genauso, von einigen Einschränkungen abgesehen. DivX-GMC benutzt nur einen Warppoint (im Gegensatz zu 3 bei Xvid), QPel unterscheidet sich wenig. **Bidirectional encoding** konfiguriert die B-Frames. **Off** dürfte sich selbst erklären ;-), **Adaptive Single Consecutive** entspricht dem verhalten bisheriger DivX-Versionen, d. h. mehrere B-Frames hintereinander werden nicht gesetzt. Maximal zwei aufeinander folgende B-Frames erlaubt **Adaptive Multiple Consecutive**.

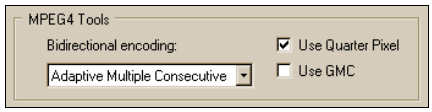

Abbildung 109: B-Frames, Qpel und GMC in DivX.

B-Frames sollten wir auf jeden Fall aktivieren. Wenn ich Doom9 durchstöbere, können das ruhig die **Multiple Consecutive** sein. Und wer B-Frames wirklich gezielt konfigurieren will, muss sowieso auf Xvid umsteigen. QPel würde ich einschalten, GMC nicht. Achtung: Für Standalones sollten beide aus bleiben und maximal **Single Consecutive** B-Frames verwendet werden.

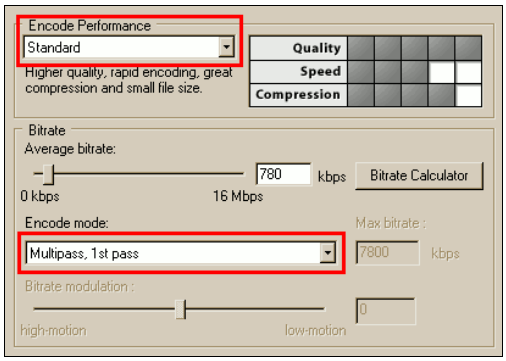

Abbildung 110: Einstellungen auf der General-Seite von DivX.

Damit verlassen wir den Assistenten und kommen wieder auf die **General**-Seite (Abbildung 110). Bei **Encode Performance** müssen wir uns entscheiden, ob Geschwindigkeit oder Qualität wichtiger ist. **Standard** ist vergleichsweise flott und bringt durchschnittliche Qualität. **Slow** bremst deutlich ab, belohnt uns aber auch mit höherer Qualität. **Fast** ist für ein auf Qualität getrimmtes 2-Pass-Encoding weniger interessant. Eine Einschränkung müssen wir im langsamen Modus hinnehmen: **QPel** funktioniert nur mit **Standard**. Wenn wir Quarter Pixel aktivieren, stellt DivX den Modus automatisch um und blendet den Schieberegler ab.

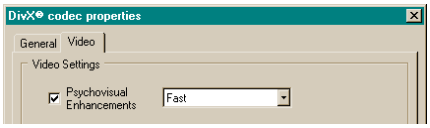

Abbildung 111: Setup für Psychovisual Enhancements.

Multipass, 1st pass ist die passende Einstellung für den ersten Durchgang. Um die Bitrate darunter müssen wir uns nicht kümmern, denn die setzt Gordian Knot automatisch ein.

Und damit weiter zum **Video**-Register. Hier sind erst einmal die **Psychovisual Enhancements** aus Abbildung 111 interessant. Alles andere können wir bei den Standards belassen. PVE versucht die Bildteile zu errechnen, die das menschliche Auge sowieso nicht wahrnehmen kann, und lässt diese weg. Die frei werdende Bitrate kann dann für andere Teile des Bilds verwendet werden.

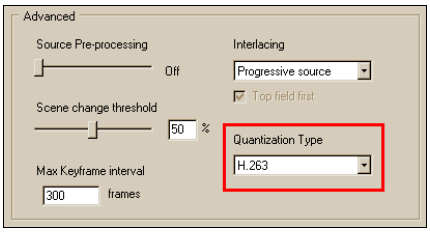

Abbildung 112: Matrixauswahl in Divx.

Der **Fast**-Modus ist aggressiver als der **Slow**-Modus, d. h. er bringt mehr, erhöht aber auch die Gefahr, dass sichtbare Artefakte im Bild auftauchen. Mit **Slow** sind wir da auf der sichereren Seite, allerdings um den Preis niedrigerer Geschwindigkeit und geringerer Effektivität der PVE.

Unter **Quantization Type** lässt sich angeben, welche Quantisierungsmatrix DivX verwenden soll. Details dazu haben wir bei der Xvid-Konfiguration schon gesehen. Da **MPEG-2** in DivX 5.2.1 noch nicht richtig ausgereift war, würde ich bei **H.263** bleiben.

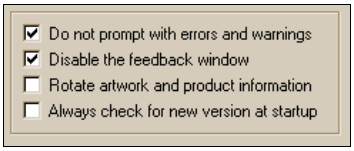

Abbildung 113: Sonstige DivX-Einstellungen.

In den **Settings** setzen wir unten die Haken bei **Do not prompt**... und **Disable the Feedback window**. So überschreibt DivX im 2nd Pass ungefragt die Log-Datei (was er ja tun soll) und schaltet das Statistik-Fenster während des Encodierens ab. **Rotate artwork** und **Check for new version** kann jeder ganz nach seinen Vorlieben setzen. Naja … das Statistik-Fenster eigentlich auch. :-)

Einstellungen für den Nth Pass

Den Großteil aller Einstellungen lassen wir für alle Durchgänge gleich. Nur auf der **General**-Seite ändert sich ab dem 2nd Pass etwas. Hier stellen wir wie in Abbildung 114 **Multipass, nth pass** ein und setzen die Haken bei **Update log file**. Die Aktualisierung der Logdatei ist wichtig, da dadurch erst die weiteren Qualitätsverbesserungen in den Passes 3 bis n ermöglicht werden.

Da kommt auch die Frage auf, wie viele Durchgänge denn sinnvoll sind. Drei Stück hat sich als vernünftiger Wert durchgesetzt. Mehr Passes bringen nur noch minimale Qualitätssteigerungen und dürften höchstens bei hoch komprimierten 1-CD-Encodings den Zeitaufwand wert sein.

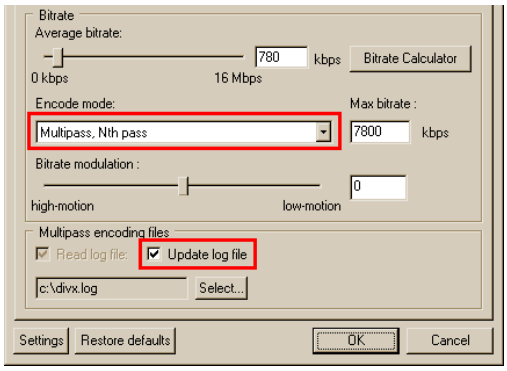

Abbildung 114: DivX-Setup für den Nth Pass.

C.3 Die Bedeutung der ITU-R BT.601 für das PAR

Das Anamorph-Kapitel weiter vorne hat es allein schon in sich, wenn man zum ersten Mal damit in Kontakt kommt. Die Hintergründe und die Diskussion um die verschiedenen Pixel Aspect Ratios dort auch noch anzusprechen, würde den Rahmen deutlich sprengen. Deswegen beschäftigen wir uns erst jetzt damit.

Analog-Digital-Wandlung

Die Empfehlung Nummer BT.601 der ITU-Organisation, die so viel Verwirrung stiftet, hat die Konvertierung analogen Filmmaterials in ein digitales Format zum Thema. Warum sollte uns das eigentlich interessieren? Schließlich haben wir eine DVD – ein vollständig digitales Medium – als Quelle und erzeugen eine genauso vollständig digitale Zieldatei. Trotzdem spielt Analog eine entscheidende Rolle, denn

- nach wie vor wird die Mehrzahl der Filme mit analogen Kameras gefilmt,
- das typische Ausgabemedium einer DVD ist ein analoger Fernseher.

Beginnen wir deshalb ganz vorn bei der Umwandlung von analog nach digital. Etwas vereinfacht dargestellt arbeitet PAL mit 576 analogen Zeilen (Scanlines), NTSC mit 486. Es liegt nahe, Zeilenanzahl mit Pixelanzahl gleichzusetzen, und genau so erfolgt auch die vertikale Analog-Digital-Konvertierung.

Wenn wir von quadratischen Pixeln ausgehen, lässt sich daraus leicht die digitale horizontale Auflösung errechnen, ohne noch einmal auf die analoge Welt zurückzugreifen. Wir wissen, dass die DVD nur 16:9 und 4:3 als Seitenverhältnis zulässt. Außerdem kennen wir die vertikale digitale Auflösung. Entsprechend rechnen wir z. B. für 4:3 PAL:

hor. Auflösung = $576 \times 4/3 = 768$ Pixel.

Für die anderen drei Fälle setzen wir jeweils die passenden Werte ein und erhalten die vier Standardauflösungen aus Tabelle 11.

	PAL	NTSC
4:3	768 × 576	648 × 486
16:9	1024 × 576	864 × 486

Tabelle 11: Rechnerische digitale Standardauflösungen.

Wichtig ist diese Tabelle für die Berechnung der Seitenverhältnisse weiter unten. Zu den Eigenschaften des analogen Videosignals passt sie nicht, denn »analoge Pixel« – wenn man überhaupt von so etwas sprechen kann – sind nicht quadratisch, sondern rechteckig; breiter als hoch, wie in Abbildung 115 zu sehen. Diese Tatsache ist zumindest ein Grund für die verzerrte Auflösung der DVD. Aber das nur am Rande. Wir stehen noch immer vor dem Problem, die »echte« horizontale Auflösung zu ermitteln. Eine natürliche Aufteilung in diskrete Werte wie in der Vertikalen existiert nicht. Eine analoge Scanline ist eine Wellenform von einigen Mikrosekunden Dauer und ohne jede offensichtliche Unterteilung.

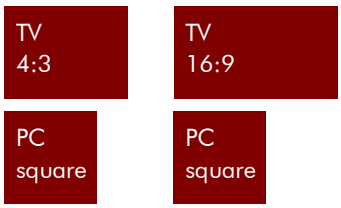

Abbildung 115: Pixelformen.

Um diskrete (digitale) Werte zu erhalten, müssen wir das analoge Signal in bestimmten Zeitabständen abtasten und den ermittelten Wert als Pixel speichern. Das nennt man Sampling, und die Geschwindigkeit der Abtastung heißt Samplingrate. Um ein TV-Bild sinnvoll zu digitalisieren, sind mehrere Millionen Abtastungen pro Sekunde nötig. Für die genauen Werte hilft uns die ITU-Empfehlung weiter.

Aussage der ITU-R BT.601

PAR-Berechnung für PAL

Der Elektronenstrahl eines PAL-Fernseher benötigt 52 μs Zeit, um eine Scanline zu zeichnen. Aus der ITU-R BT.601 ergibt sich eine Samplingrate von 13,5 MHz (Megahertz = Millionen Abtastungen pro Sekunde). Damit erhalten wir 52 μs × 13,5 MHz = 702 horizontale Pixel, was einer aktiven Auflösung von 702 × 576 Pixeln entspricht. *Aktiv* heißt, das komplette Bild, wie es einmal auf dem analogen Film vorhanden war, soll innerhalb dieser 702 Pixel liegen.

Die Auflösung einer PAL-DVD ist auf 720 × 576 Pixel festgelegt. Zwar erlaubt der Standard auch ein paar andere Auflösungen, die praktisch aber nur geringe Bedeutung haben. Der Grund für die abweichende Breite der aktiven ITU-Auflösung ist der Fernseher, dessen Konstruktion zu einem Overscan-Bereich führt.

Das heißt, ein dünner Streifen Bild ragt über den Rand des Bildschirms hinaus. Da der Zuschauer diesen Bereich natürlich nicht sieht, sollte darin auch kein wichtiges Bild enthalten sein.

Grafisch stellt sich die Situation wie im folgenden Bild dar. Die graue Fläche ist die DVD-Auflösung, die dünne rote Linie kennzeichnet das aktive ITU-Bild. An dieser Stelle benötigen wir unsere Tabelle 11 mit den quadratischen Auflösungen, denn Computerpixel sind nie (fast nie, siehe Anmerkung Seite 45) rechteckig. Mit einem 16:9-Bild stehen wir dann vor der Situation, 1024×576 quadratische Pixel in 702×576 rechteckigen Pixeln unterbringen zu müssen. Vertikal kein Problem, da die Auflösungen exakt übereinstimmen. Horizontal kommen wir allerdings nicht darum herum, die 1024 vorhandenen Pixel in 702 aktive Pixel zu quetschen. Machbar ist das ohne weiteres, lediglich produzieren wir als Nebeneffekt ein verzerrtes (horizontal gestauchtes) Bild. Mit einem 4:3-Bild spielen wir genau das gleiche durch, mit dem einzigen Unterschied, ein weniger verzerrtes Bild zu erhalten.

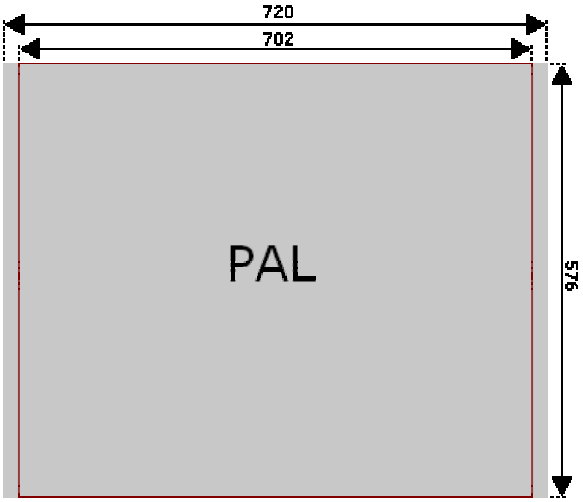

Abbildung 116: Auflösung und aktiver Bereich der PAL-DVD.

Beim Abspielen brauchen wir dann eine Angabe über den Verzerrungsfaktor, um das Bild wieder auf die richtige Breite strecken zu können. Dieser Faktor ist das Pixel Aspect Ratio (PAR). Für 16:9 PAL sagt uns der PAR-Wert von ca. 1,46, dass wir das Bild auf 146 % auseinanderziehen müssen, um ein korrektes Seitenverhältnis zu erhalten.

Das PAR selbst zu errechnen, ist mit dem Wissen über die ITU-Empfehlung einfach. Wir packen dazu die Information »Wie verhält sich die korrekte zur gestauchten Breite« in eine mathematische Schreibweise, was nichts anders als der Quotient aus korrekter und gestauchter Breite ist. Entsprechend ergibt sich für den 4:3- und 16:9-Fall:

$$\textit{4:3-PAR} = \frac{768}{702} = \frac{128}{117} \approx 1{,}0940$$

$$\textit{16:9-PAR} = \frac{1024}{702} = \frac{512}{351} \approx 1{,}4587$$

Um beim Abspielen die richtige Breite zu erhalten, müssen wir die gestauchte Breite mit dem PAR multiplizieren. Aber Achtung! Auch wenn nur 702 Pixel aktiv sind, hat die DVD trotzdem eine Auflösung von 720 Pixeln. Auch die 18 nicht aktiven Pixel müssen gestreckt werden, denn halbe Sachen sind nicht erlaubt. Entweder strecken wir das komplette Bild oder wir lassen es ganz bleiben.

4:3-Breite = $720 \times {}^{128}/_{117} = 787{,}69$ Pixel

16:9-Breite = $720 \times {}^{512}/_{351} = 1050{,}26$ Pixel

Damit sollte auch klar werden, warum nach ITU-Norm das Bild ein wenig breiter als die Standard-DARs 4:3 oder 16:9 ausfällt. Das sind genau die 18 nicht aktiven Pixel, die gestreckt exakt die Abweichung zum Standard-DAR (19,69 bzw. 26,26 Pixel) ergeben. Damit entspricht auch der *aktive Teil* der endgültigen Auflösung genau der Ausgangssituation: 768 × 576 und 1024 × 576.

PAR-Berechnung für NTSC

Die Situation für NTSC stellt sich ähnlich dar. Den quadratischen Auflösungen 648 × 486 und 864 × 486 steht nach ITU ein aktiver Bereich von 710,85 × 486 entgegen. Ja, exakt so. Die krummen horizontalen Pixel müssen für die Praxis natürlich auf 711 aufgerundet werden. Damit sich die Rundungsfehler nicht summieren, rechnen wir trotzdem mit den exakten 710,85.

Genau genommen summiert sich nichts Wesentliches. Als letzter Schritt vor der eigentlichen Darstellung am Bildschirm muss sowieso aufs nächste volle Pixel gerundet werden, was den minimalen rechnerischen Unterschied zwischen 710,85 und 711 ausgleicht.

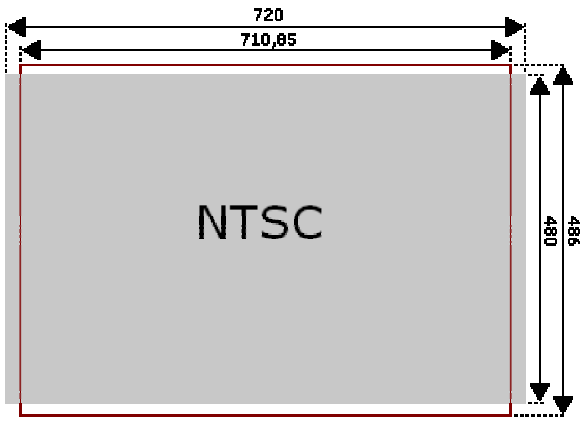

Abbildung 117: Auflösung und aktiver Bereich der NTSC-DVD.

Unschöner sind die 480 vertikalen DVD-Pixel, die für die 486 aktiven Pixel nicht ausreichen. Wir stehen grafisch also vor folgender Situation, die auf den ersten Blick schon spürbar komplizierter als bei PAL aussieht (Abbildung 117). Die graue Fläche stellt wieder die DVD-Auflösung dar, der rote Rahmen den aktiven ITU-Bereich. Prinzipiell wäre es möglich, vertikal genauso zu verfahren wie horizontal, also 486 tatsächliche Pixel in 480 DVD-Pixel zu quetschen und zusätzlich zum horizontalen mit einem vertikalen PAR zu arbeiten. Das ist genauso lästig wie es klingt, deswegen hat man sich für ein einfa-

cheres Verfahren entschieden. Was nicht passt, wird passend gemacht! Wir schneiden ganz einfach oben und unten je drei Pixel ab und gut ist. Da die Ränder des Bildes schon im rein analogen Umfeld typischerweise Rauschen anstatt aktives Bild enthalten, stören die fehlenden Pixel nicht. Natürlich bringt das Cropping eine minimale Abweichung des DAR mit sich, die aber vernachlässigt werden kann. Die Ungenauigkeit in der Darstellung, die sich allein aus der Technik der Fernsehbildröhre ergibt, ist deutlich größer.

Damit befinden wir uns tatsächlich in einer Situation, die derjenigen für PAL entspricht. Die vertikale Auflösung passt und horizontal stopfen wir die Originalpixel in die aktiven Pixel. Für 4:3 ergibt sich dabei die etwas seltsame Situation, 648 Pixel in 710,85 Pixel »quetschen« zu müssen. Nicht verwirren lassen. Das ändert am Vorgehen überhaupt nichts, lediglich das PAR wird (ganz automatisch) kleiner als 1. Hier sind die Berechnungen, die analog zu PAL funktionieren.

$$\textit{4:3-PAR} = \frac{648}{710{,}85} = \frac{4320}{4739} \approx 0{,}9116$$

$$\textit{16:9-PAR} = \frac{864}{710{,}85} = \frac{5760}{4739} \approx 1{,}2154$$

Beim Abspielen multiplizieren wir wieder die horizontale DVD-Auflösung mit dem PAR.

4:3-Breite = $720 \times {}^{4320}/_{4739} = 656{,}34$ Pixel
16:9-Breite = $720 \times {}^{5760}/_{4739} = 875{,}12$ Pixel

Auch hier entsprechen die gestreckten nicht-aktiven Pixel dem Unterschied zum Standard-DAR – bis auf die minimale Abweichung durch die abgeschnittenen sechs vertikalen Pixel.

Zusammenfassung

Fassen wir zusammen: PAL und NTSC entscheiden sich nicht wesentlich, lediglich hantieren wir bei NTSC mit unschöneren Zahlen. Beide Formate ergeben nach ITU ein etwas breiteres Bild als vom Standard-DAR gewohnt, was an den verbleibenden nicht aktiven Pixeln liegt. Am TV sehen wir trotzdem ungefähr den Standard-DAR, da die zusätzliche Breite im Overscan-Bereich verschwindet. Für die korrekte Wiedergabe müssen wir insgesamt vier PAR-Werte im Kopf behalten, siehe Tabelle 12.

	PAL	NTSC
4:3	128 / 117	4320 / 4739
16:9	512 / 351	5760 / 4739

Tabelle 12: Exaktes PAR nach ITU-R BT.601.

Wem einmal die PARs 72 / 79 für 4:3 und 96 / 79 für 16:9 über den Weg laufen, das sind die Werte für NTSC, gerechnet mit 711 aktiven Pixeln.

Die vereinfachte PAR-Tabelle

Der aufmerksame Leser wundert sich bestimmt schon längst wieder. Die ITU-PAR-Tabelle hier stimmt doch mit der Tabelle aus dem Anamorph-Kapitel nicht überein! Die sieht nämlich so aus:

	PAL	NTSC
4:3	12 / 11	10 / 11
16:9	16 / 11	40 / 33

Tabelle 13: Vereinfachtes PAR nach ITU-R BT.601.

So unterschiedlich die Brüche auf den ersten Blick anmuten: wenn man sie ausdividiert, kommt bei beiden Tabellen fast das Gleiche heraus. Die Abweichung ist mit 2 – 3 Pixeln (etwa 0,3 %) so gering, dass sie unsichtbar bleibt. Ob wir deshalb mit dem exakten PAR oder den vereinfachten Werten arbeiten, ist irrelevant.

Die vereinfachte Tabelle 13 entstammt dem MPEG-4-Standard. Im H.264-Teil, den man von der ITU-Homepage kostenlos herunterladen kann, stehen die Zahlen so im Kapitel E.2.1 in Tabelle E-1. Leider ist der Rest des MPEG-4-Standards nicht frei verfügbar, weshalb ich im für Xvid und DivX relevanten Part 2 (MPEG-4 Visual) nicht selbst nachschauen kann. Dass die gleichen Zahlen auch dort genannt werden, darauf lässt mich eine *Äußerung Koepis* im Doom9-Forum schließen.

Bleibt die Frage, warum die Angaben im MPEG-4-Standard überhaupt von den exakten ITU-Werten abweichen. Darüber bin ich mir selbst nicht im klaren. Fakt ist jedenfalls, dass sowohl der 3ivx- als auch der Xvid-Encoder die vereinfachte Tabelle für ihre Standard-PAR-Auswahl verwenden. Zusammen mit den simplen

Zahlen, die man gut im Kopf behalten kann, ist das Grund genug, die einfache Tabelle für die Praxis zu empfehlen.

ITU-PAR oder generisches PAR?

Mit ITU-PAR meine ich im Folgenden immer die exakten und vereinfachten Werte gleichermaßen. Natürlich könnte man zwischen den beiden genauso unterscheiden wie zwischen ITU-PAR und generischem PAR. Wegen der nur minimalen Abweichung halte ich das aber für übertriebene Haarspalterei.

Begriff des Generischen PAR

Dass die ITU-R BT.601 genau die Rechnung anstellt, wie oben beschrieben, liegt an ihrer Konzentration auf die analoge Welt, besonders auf analoge Abspielgeräte mit Overscan-Bereich. Denn um genau diesen Overscan-Bereich dreht sich die ganze Empfehlung.

Der Film einer ITU-DVD ist am TV-Bildschirm vollständig zu sehen. Zwar versteckt der Fernseher die Ränder des Videos im unsichtbaren Overscan-Bereich, jedoch enthalten die sowieso kein aktives Bild, so dass nichts verloren geht.

Das stimmt jedenfalls ungefähr. Die Größe des Overscans ist nicht immer und überall gleich. Also muss ITU mit einem Schätzwert arbeiten.

Geht man von einem digitalen Wiedergabemedium aus – oder allgemeiner von einem Wiedergabemedium ohne Overscan – verliert die ITU-Empfehlung ihren Sinn. Eine nach ITU gemasterte DVD kann dann sogar nachteilig sein, weil sie möglicherweise links und rechts sichtbare dünne schwarze Balken enthält.

Das generische PAR legt genau dieses digitale Szenario zu Grunde. Die Berechnung funktioniert genauso wie weiter oben für ITU beschrieben, nur dass wir uns um einen aktiven Bereich keine Gedanken machen. Aktiv ist die komplette Auflösung der DVD: 720×576 bzw. 720×480. Entsprechend rechnen wir nicht mit 702 oder 710,85 im Nenner, sondern mit 720. Das ergibt schlussendlich folgende PAR-Tabelle:

	PAL	NTSC
4:3	16/15	9/10
16:9	64/45	6/5

Tabelle 14: Generisches PAR.

So weit sind sich die Verfechter sämtlicher Lehren auch einig. Aber nur so weit, und kein bisschen weiter.

Diskussion

Die große Frage beim DVD-Backup besteht nun darin, welches PAR wir benutzen sollten. Der Streit darum wird heftig geführt, bleibt nicht immer ganz sachlich und geht meiner Meinung nach meistens am Kern des Problems vorbei. Erst einmal müssen wir zwei Dinge klarstellen.

- Das generische PAR ist nicht auf jeden Fall falsch. Genauso ist das ITU-PAR nicht automatisch in jedem Fall richtig.
- Die Art der Wiedergabe (am Computer, TV, sonstwo) spielt keine Rolle.

Immer unter der Voraussetzung, dass wir das ursprüngliche Seitenverhältnis so exakt wie möglich wieder herstellen wollen, existiert genau ein entscheidender Faktor, der über das richtige PAR entscheidet:

Mit welchem PAR arbeitet die jeweilige DVD?

Eigentlich ist die Überlegung so simpel, dass überhaupt keine Diskussion entstehen sollte. Wenn Person A einen Film horizontal auf die Hälfte staucht und Person B das ursprüngliche Bild wieder herstellen will, dann muss Person B den Film auf 200 % auseinander ziehen. Und wenn Person C einen anderen Film auf drei Viertel seiner Breite quetscht, dann muss Person B diesen Film auf 133 % auseinander ziehen. Im zweiten Fall stur auf 200 % zu beharren, wäre blödsinnig. Genauso verhalten sich aber sowohl die Hardcore-Verfechter des ITU-PAR als auch die Hardcore-Verfechter des generischen PAR. Da wird seitenlang gestritten und am eigentlichen Kern des Problems geradeaus vorbeidiskutiert.

Das richtige PAR ist dasjenige, das im Masteringprozess benutzt wurde.

An dieser Stelle muss eine sinnvolle Diskussion ansetzen, denn so klar die Grundaussage auch ist, die praktische Umsetzung bereitet Probleme. Das PAR ist näm-

lich nirgendwo auf der DVD vermerkt, es existiert also keine Möglichkeit, den richtigen Wert einfach abzulesen. Genauso unmöglich erscheint es bei kommerziellen DVDs, den Masteringtechniker ausfindig zu machen und nachzufragen. Es bleibt also nur die Trial-and-Error-Methode: Wir entzerren das Bild mit den verschiedenen PARs und messen nach, ob Kreise auch wirklich rund erscheinen.

Das ist zu kompliziert und zeitaufwändig für die tägliche Praxis? Stimmt. Möglicherweise können wir aber mit ein wenig Überlegung einer der beiden Tabellen den Vorzug geben. Um das Ergebnis vorweg zu nehmen:

Das ITU-PAR zu wählen, ist die logische Alternative.

Warum? Für DVDs mit vertikalen Balken erscheint es unsinnig, generisches PAR anzunehmen. Aus welchem Grund sollte jemand zwar 720 Pixel als aktiv ansehen, dann aber trotzdem nicht die volle Breite ausnutzen? Solche DVDs würde ich deswegen immer ohne zu zögern nach ITU entzerren, zumal die Balken oft verdächtig genau im Bereich der 2×9 nicht aktiven PAL-Pixel liegen.

Gerade neuere DVDs nutzen immer häufiger die komplette DVD-Auflösung aus. Trotzdem erscheint es in der Regel sinnvoller, auch hier das ITU-PAR anzuwenden. Denn dass beim Mastering ITU angewendet und die inaktive Fläche anstatt mit Schwarz mit *zusätzlichem* Bild gefüllt wurde, ist aus einigen Gründen wahrscheinlicher.

- Wichtigstes Wiedergabemedium für die DVD ist der analoge Fernseher, auf dessen Overscan-Eigenschaft die ITU-Norm zugeschnitten ist. Nach generischem PAR geht am TV links und rechts ein Streifen Bild verloren.
- Digitale Wiedergabegeräte ohne Overscan gewinnen an Bedeutung. Mit der Methode, ITU zu verwenden und den inaktiven Bereich mit einem zusätzlichen Stückchen Bild zu füllen, wird man jeder Abspielmethode gerecht.
- Standalone-DVD-Player verwenden tendenziell das ITU-PAR.
- Professionelle Videsoftware und ganz besonders professionelle Videohardware hält sich strikt an die ITU-Norm. Dass ein Profi mit generischem PAR arbeiten sollte, mutet deshalb etwas seltsam an.
- Die vereinfachte PAR-Tabelle lehnt sich stark an die ITU-Werte an.

Das alles beweist nichts eindeutig und endgültig. Es lässt allerdings die Aussage äußerst wahrscheinlich erscheinen, dass die überwiegende Mehrheit der DVDs entsprechend ITU-R BT.601 gemastert wird. Wer nicht jede DVD vor dem Encoding exakt vermessen will, dem rate ich deshalb immer zu den ITU-Werten, denn die Wahrscheinlichkeit, damit richtig zu liegen, ist hoch.

Lohnt sich der ganze Glaubenskrieg eigentlich wirklich? Das tut er dann, wenn zwischen ITU-PAR und generischem PAR ein deutlich sichtbarer Unterschied besteht. Die Abweichung zu den exakten ITU-Werten berechnet sich im 16:9-Fall wie unten. Mit den 4:3-Werten würden wir genau die gleichen Ergebnisse erhalten.

PAL-Abweichung = $^{512}/_{351}$ / $^{64}/_{45}$ = 2,56 %

NTSC-Abweichung = $^{5760}/_{4739}$ / $^{6}/_{5}$ = 1,29 %

Für klassische Encodings mit quadratischen Pixeln haben wir einen Aspect Error bis 2,5 % als unproblematisch definiert. NTSC liegt also klar in dem Bereich, an den wir beim klassischen Encoding keinen weiteren Gedanken verschwenden würden. PAL testet die Grenze zwar an, aber um das Kapitel A.3.3 über die nichtanamorphe Zielauflösung zu zitieren: »Geringe Fehler – etwa bis 2,5 % – sind unproblematisch, weil die Verzerrung zu klein bleibt, um spürbar zu werden.«

Aha. Ab in etwa 2,5 % können wir also langsam damit rechnen, dass die Abweichung sichtbar werden könnte. Nach Katastrophenfall klingt das keineswegs. Auch sind die 2,5 % vorsichtig angesetzt. Gordian Knot z. B. warnt erst bei über 3,5 % vor einer kritischen Abweichung.

Die komplette PAR-Diskussion hat also schon ihre Berechtigung. Nur geht in Grabenkämpfen und zwischen technischen Details oft die Perspektive verloren. Wir streiten uns um Abweichungen, für die wir an anderer Stelle seit Jahren nur ein Schulterzucken übrig haben. So richtig will es mir nicht einleuchten, warum ein Unterschied, den wir bisher nicht sehen konnten, plötzlich so durchschlagend wichtig sein sollte.

Teil D
Anhang

Literatur

Artikel und Bücher

Advanced video coding for generic audiovisual services. ITU Recommendation ITU-T H.264. Online verfügbar unter http://www.itu.int/rec/T-REC-H.264

Aho, Jukka: *A Quick Guide to Digital Video Resolution and Aspect Ratio Conversions.* Online verfügbar unter http://www.iki.fi/znark/video/conversion

Aho, Jukka: *The 625/50 PAL Video Signal and TV Compatible Graphics Modes.* Online verfügbar unter http://www.iki.fi/znark/video/modes

Crusty: *The Unofficial XviD FAQ*. Online verfügbar unter http://ronald.vslcatena.nl/docs/xvidfaq.html

Das Video-Kompressionsbuch. Panasonic. Online verfügbar unter http://panasonic-broadcast.de/index.cfm?uuid=356BA7F6C09F11269B3BB10E84B5C76A

Doom9: *Codec shoot-out 2005 – Final.* Online verfügbar unter http://www.doom9.org/codecs-final-105-1.htm

Fiedler, Martin: *Videokompressionsverfahren von MPEG-1 bis DivX, VC-1 und H.264.* Online verfügbar unter http://keyj.s2000.ws/files/projects/videocomp.pdf

LAME. Hydrogenaudio Wiki. Online verfügbar unter http://wiki.hydrogenaudio.org/index.php?title=LAME

Mares, Sebastian: *Public, Multiformat Listening Test @ 128 kbps.* Online verfügbar unter http://www.maresweb.de/listening-tests/mf-128-1

Oberle, Daniel: *Das Breitbild-Phänomen.* Online verfügbar unter http://www.aifb.uni-karlsruhe.de/WBS/dob/docs/Breitbild.pdf

Pillai, Latha: *Quantization.* Xilinx XAPP615 (v1.1) June 25, 2003. Online verfügbar unter http://www.xilinx.com/bvdocs/appnotes/xapp615.pdf

Richardson, Iain E. G.: *H.264 and MPEG-4 Video Compression.* Wiley, 2003. ISBN 0-470-84837-5

scharfis_brain: *Exotisches Interlacing.* Online verfügbar unter http://home.arcor.de/scharfis_brain/ExotischesInterlacing

Selur: *Wissenswertes rund um x264.* Online verfügbar unter http://www.flaskmpeg.info/board/thread.php?postid=54931

Selur: *Wissenswertes rund um Xvid.* Online verfügbar unter http://www.flaskmpeg.info/board/thread.php?postid=50836

Studio Encoding Parameters of Digital Television for Standard 4:3 and Widescreen 16:9 Aspect Ratios. ITU Recommendation ITU-R BT.601-5. Online verfügbar unter http://www.itu.int/rec/R-REC-BT.601

Taylor, Jim: *DVD Frequently Asked Questions (and Answers).* Online verfügbar unter http://www.dvddemystified.com/dvdfaq.html

The Official DivX 5.2 Guide. DivX Inc. Deutsche Version online verfügbar unter http://download.divx.com/divx/DivXUserGuide521-de.exe

Wickenburg, Sebastian et al.: *Die JPEG-Kompression.* Online verfügbar unter http://www.mathematik.de/spudema/spudema_beitraege/beitraege/rooch/neinleit.html

Forum-Threads

Gleitz.info: *agkp – akapuma's GKnot Personalisierer.* http://forum.gleitz.info/showthread.php?t=22860

Gleitz.info: *ITU-R BT.601 und das PAR: Ist mein Wissen korrekt?* http://forum.gleitz.info/showthread.php?t=28145

Doom9.org: *Audio FAQ.* http://forum.doom9.org/showthread.php?t=68300

Doom9.org: *GPAC MP4Box Information, Requests and Discussions.* http://forum.doom9.org/showthread.php?t=94874

Doom9.org: *Human color sensitivity test.* http://forum.doom9.org/showthread.php?t=72803

Doom9.org: *ITU-R BT.601 and PAR: How good is my knowledge?* http://forum.doom9.org/showthread.php?t=111102

Doom9.org: *New Quantization Matrices – EQM V3 series.* http://forum.doom9.org/showthread.php?t=83125

Doom9.org: *New x264 feature: mixed references.*
http://forum.doom9.org/showthread.php?t=101056

Doom9.org: *Xvid_encraw – Patched with AviSynth input support.*
http://forum.doom9.org/showthread.php?t=98469

Weitere nützliche Links

What is Deinterlacing? Facts, solutions, examples.
http://100fps.com

BBC - A Guide to Picture Size
http://www.bbc.co.uk/commissioning/tvbranding/picturesize.shtml

Homepage der Moving Picture Experts Group
http://www.chiariglione.org/mpeg

GSpot: Analysetool für Filmdateien
http://www.headbands.com/gspot

Mediainfo: Analysetool für Filmdateien
http://mediainfo.sourceforge.net

Videoplayer
Media Player Classic: http://sourceforge.net/projects/guliverkli
Zoom Player: http://www.inmatrix.com
VLC Media Player: http://www.videolan.org
The Core Media Player: http://www.corecoded.com

Haali Media Splitter mit Unterstützung für Matroska, MP4, AVI, OggMedia
http://haali.cs.msu.ru/mkv

ffdshow: DirectShow-Decoder für viele Audio- und Videoformate
http://www.ffdshow.info
Bob0rs Builds: http://x264.nl

Alexander Vigovskys AC3-Decoder
http://ac3filter.sourceforge.net

RareWares: Downloadseite für Audiosoftware
http://www.rarewares.org

Abkürzungen

AAC
Advanced Audio Coding

ABR
Average Bitrate

AC3
Audio Codec 3

agkp
Akapumas Gordian-Knot-Personalisierer

AR
Aspect Ratio

ASF
Advanced Streaming Format

ASP
Advanced Simple Profile

AVC
Advanced Video Coding

AVI
Audio/Video Interleave

AVS
AviSynth Script

BPF
Bits/Pixel*Frame

C
Center Channel

CABAC
Context-Adaptive Binary Arithmetic Coding

CBR
Constant Bitrate

CD
Compact Disc

CSS
Content Scrambling System

DAR
Display Aspect Ratio

DCT
Discrete Cosine Transformation

DNR
Dynamic Noise Reduction

DPL
Dolby Pro Logic

DRC
Dynamic Range Compression

DTS
Digital Theater Systems

DVD
Ohne Bedeutung laut DVD FAQ.

DVD-5
Single-Layer-DVD mit 4,37 GB Kapazität

DVD-9
Dual-Layer-DVD mit 7,95 GB Kapazität

FL
Front Left Channel

FR
Front Right Channel

GKnot
Gordian Knot

GMC
Global Motion Compensation

GUI
Graphical User Interface

HE
High Efficiency

iDCT
Inverse Discrete Cosine Transformation

JPEG
Joint Photographic Experts Group

LAME
LAME Ain't an MP3 Encoder

LC
Low Complexity

LFE
Low Frequency Effects Channel

LPCM
Linear Pulse Code Modulation

MKV
Matroska Video

mmg
mkvmerge GUI

MP2
MPEG-1 Audio Layer 2

MP3
MPEG-1 Audio Layer 3

MPEG
Moving Picture Expert Group

NTSC
National Television System Committee

OGM
OggMedia

OTA
Overall Track Adjustment

PAL
Phase Alternation Line

PAR
Pixel Aspect Ratio

PCM
Pulse Code Modulation

PGC
Program Chain

PN
Private Nachricht

PVE
Psychovisual Enhancements

QPel
Quarter Pixel

SBC
Smart Bitrate Coding

SBR
Spectral Band Replication

SL
Surround Left Channel

SR
Surround Right Channel

SSRC
Shibatch Sample Rate Converter

SVCD
Super VideoCD

VBR
Variable Bitrate

VCD
VideoCD

VfW
Video for Windows

VHQ
Very High Quality

VOB
Video Object

VTS
Video Title Set

Changelog

Hier stehen lediglich die Änderungen zur jeweiligen Vorgängerversion. Das vollständige Changelog seit Adam und Eva könnt ihr online einsehen:
http://encodingwissen.brother-john.net/changelog.html

8. Dezember 2006 (rev 35)

- Die PDF-Version des Encodingwissen ist wieder da! Wie früher ist die für den Druck optimiert und hat jetzt sogar ein Layout, das einigermaßen professionell aussieht. An manchen Stellen gibt es an optischen Details noch ein bisschen zu feilen, aber das hat erstmal Zeit. Danke an Naito, der mir die Fleißarbeit abgenommen hat, den Text aus den HTMLs zu übertragen. Eine aktualisierte PDF gibt es voraussichtlich für alle größeren Updates, nicht wegen jeder Kleinigkeit. Das Layout jedesmal zu überprüfen, braucht doch seine Zeit und lässt sich leider nicht automatisieren.
- Nicht über das neue System mit den Revisionsnummern wundern. Da werkelt seit einiger Zeit Subversion im Hintergrund.
- Jede HTML-Seite hat ganz unten rechts ein individuelles Änderungsdatum bekommen.
- Softwareliste: Programme erwähnt, die im Gordian-Knot-RipPack enthalten sind. Wer nicht die absolut neusten Versionen braucht, kann sich so leichter einiges an Download sparen.
- Absatz zum MPEG Modifier angepasst: Der unterstützt inzwischen OpenDML-AVIs.
- Vorstellung von DivX und Xvid aktualisiert. So krass hinter Xvid, wie dort dargestellt, rangiert DivX nicht mehr.
- Anamorph-Diskussion konkretisiert: Vereinfachte PAR-Werte stehen tatsächlich im MPEG-4-Standard.
- Massig Rechtschreib-, Grammatik-, Stilkorrekturen, aktualisierte Links und einen Happen neue Literatur.
- Lizenzänderung zu »Namensnennung-NichtKommerziell-KeineBearbeitung 2.0 Deutschland«.