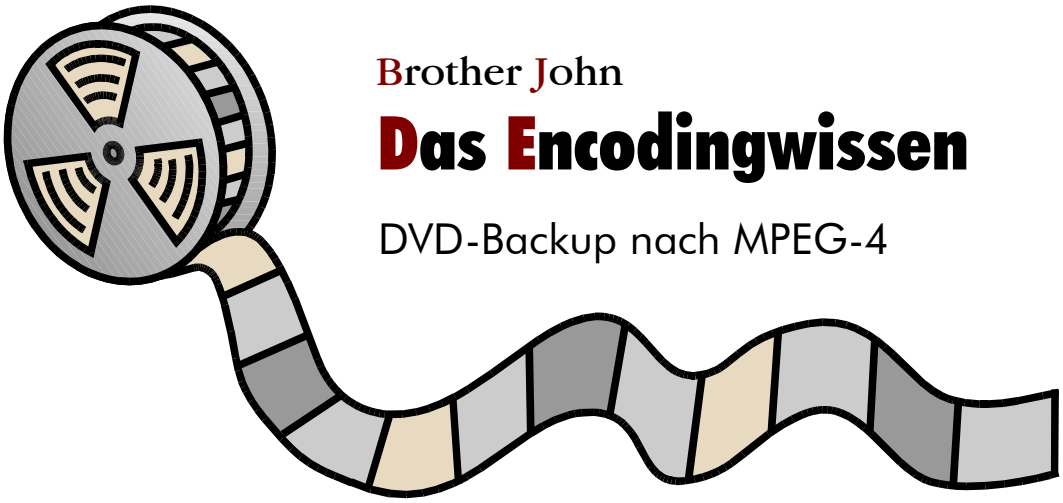


Brother John

Das Encodingwissen

DVD-Backup nach MPEG-4

Revision 264 vom 2. Juli 2008

http://encodingwissen.de

Copyright © Brother John 2004 – 2008

Lizenzen (siehe auch Seite 248):
Dieses Dokument unterliegt der Creative-Commons-Lizenz
Namensnennung – Weitergabe unter gleichen Bedingungen 2.0 Deutschland.
http://creativecommons.org/licenses/by-sa/2.0/de/
Als Ausnahme davon unterliegt die »Filmrolle«-Logografik der GNU Free Documentation License.
http://www.gnu.org/copyleft/fdl.html

Schriften:
Text: Bitstream ITC Garamond, 11 Pt
Überschriften: Bistream Futura, 22 Pt bis 11 Pt
Nichtproportionales: DejaVu Sans Mono (http://dejavu.sourceforge.net/)
Randspalte: Bitstream ITC Garamond Condensed, 8 Pt
Kopf-/Fußzeilen: Bistream Futura Condensed Light, 9 Pt

Software:
OpenOffice.org (http://de.openoffice.org/) mit extendedPDF (http://www.3bview.com/epdf-home.html)
The GIMP (http://gimp.org/)
IrfanView (http://irfanview.com/) mit PngOut.dll (http://www.advsys.net/ken/utils.htm)

Inhalt

Vorwort und Wegweiser V
Philosophie des Encodingwissens VII

Teil A
Hintergrundwissen 1

A.1 Grundlagen 3
A.1.1 Der Nutzen der Kompression 3
A.1.2 Die Videokompression 6
A.1.2.1 Intraframe-Kompression 6
A.1.2.2 Interframe-Kompression 10
A.1.3 Die Audiokompression 13
A.1.4 Bestandteile eines Films 16
A.1.5 Der Ablauf eines Encodings Schritt für Schritt 19

A.2 Der Formatedschungel 22
A.2.1 Video- und Audioformate 22
A.2.2 Das Format der DVD 26
A.2.3 Die Containerformate 28
A.2.4 Die Videocodecs 32
A.2.5 Die Audiocodecs 34
A.2.6 Die Untertitelformate 38

A.3 Das Videobild 41
A.3.1 Cropping (Zuschneiden) 41
A.3.2 Anamorphes Video 42
A.3.2.1 Anamorphes Quellvideo 42
A.3.2.2 Anamorphes MPEG-4 49
A.3.3 Die passende Zielauflösung 54

A.4 Videocodec-Konfiguration 60
A.4.1 1-Pass vs. 2-Pass: Encodingmethoden 60
A.4.2 Interfacetypen: VfW und Kommandozeile 64
A.4.3 Die Xvid-Konfiguration 66
A.4.3.1 Xvid Kommandozeilen-Konfiguration 77

A.4.3.2 Xvid VfW-Konfiguration ... 88
A.4.3.3 Zuordnung von Xvids VfW- und CLI-Optionen ... 96
A.4.4 Der x264-Encoder ... 98
A.4.4.1 Die allgemeine Konfiguration von x264 ... 99
A.4.4.2 x264 Kommandozeilen-Konfiguration ... 106
A.4.5 DivX 6.8 VfW-Konfiguration ... 119

Teil B
Praxiswissen ... 125

B.1 Vorarbeiten ... 127
B.1.1 Die nötige Software ... 127
B.1.2 DVD-Ripping ... 131
B.1.3 Indexieren der VOBs mit DGIndex ... 134

B.2 Audio-Transcoding ... 137
B.2.1 BeSweet einrichten ... 137
B.2.2 Audio decodieren und bearbeiten ... 139
B.2.3 AAC-Encoding mit Nero und Winamp ... 143
B.2.4 Vorbis-Encoding ... 145
B.2.5 MP3-Encoding mit LAME ... 146
B.2.6 AC-3-Encoding ... 147

B.3 Untertitel ... 148
B.3.1 Dynamische und eingebrannte Grafikuntertitel ... 149
B.3.2 Dynamisch einblendbare Textuntertitel ... 152
B.3.3 Zwangsuntertitel simulieren ... 156

B.4 Die Encoding-Frontends ... 159
B.4.1 StaxRip ... 159
B.4.1.1 StaxRip einrichten ... 161
B.4.1.2 Konfigurieren der Zieldatei ... 164
B.4.1.3 Einfügen von Audiospuren ... 167
B.4.1.4 Vorbereiten des Videos ... 170
B.4.1.5 Konfigurieren des Videoencoders ... 173
B.4.2 Gordian Knot ... 178
B.4.2.1 Gordian Knot einrichten ... 179
B.4.2.2 Kalkulieren der Bitrate ... 180
B.4.2.3 Vorbereiten des Videos ... 182

B.4.2.4 Dynamische Untertitel und Kapitel einbinden ... 185
B.4.2.5 Encoding-Parameter konfigurieren ... 185
B.4.2.6 Das Encoding starten ... 188
B.4.2.7 Akapumas Gordian-Knot-Personalisierer ... 192

B.5 Manuelles Muxing ... 202
B.5.1 MKVMerge für den Matroska-Container ... 202
B.5.2 AVI-Mux GUI für den AVI-Container ... 206

Teil C
Spezialthemen ... 209

C.1 Praktisches ... 211
C.1.1 Manuelles Splitting ... 211
C.1.2 StaxRip für neue Quellformate fit machen ... 215

C.2 Theoretisches ... 218
C.2.1 Zielauflösung, Mod16-Regel & Co. ... 218
C.2.2 Die Bedeutung der ITU-R BT.601 für das PAR ... 221

Teil D
Anhang ... 235

Literatur ... 237
Glossar ... 242
Changelog ... 247
Lizenzen ... 248

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 Werk 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 16) und zum Encodingablauf (A.1.5, Seite 19) 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 125), 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 22) springen, um dich für einen Codec o. ä. zu entscheiden. Einfach den entsprechenden Verweisen 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 125) zentral wichtig – als Umsteiger auch das Kapitel zum Encodingablauf (A.1.5, Seite 19). Außerdem solltest du es nicht versäumen, dich mit der Videocodec-Konfiguration (A.4, Seite 60) eingehender zu beschäftigen. Der Rest des Hintergrundwissens ist hauptsächlich Geschmackssache. Ich persönlich würde mir den Grundlagenteil (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 42) und ITU-R BT.601 (C.2.2, Seite 221) sein. 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.

Philosophie des Encodingwissens

Nein! Nicht gleich weiterblättern! Dieses Kapitel ist sicherlich nicht das unwichtigste im ganzen Encodingwissen. Auch wenn der Titel »Philosophie« ein wenig dick aufgetragen sein mag, so ist doch jedes Tutorial vom Wissensstand, den Überzeugungen und Ideen des Autors geprägt. Dazu möchte ich jetzt ein paar Worte verlieren. Nicht nur, um meine Meinung an den Mann zu bringen – auch wenn das natürlich nicht ganz zweitrangig ist. ;-) Nein, zu wissen, was *mir* wichtig ist, hilft sicherlich *dir* dabei, die eine oder andere Passage leichter zu verstehen.

DVD-Backup nach MPEG-4 als Ziel

Wir beschäftigen uns hier damit, Backups der teuer erstandenen eigenen DVDs zu erstellen. Es geht nicht darum, diese Encodings hinterher in irgend einer Form zu verbreiten! Entsprechend sind die speziellen Anforderung, die ein Upload o. ä. möglicherweise stellt, kein Thema.

Worin liegt denn eigentlich der Sinn, von einer DVD, die man sowieso gekauft hat, eine Kopie für den Eigenbedarf anzufertigen? Das muss am Ende jeder für sich selbst beantworten. Jedenfalls schont es die Originaldiscs, und sollte doch einmal ein Original kaputtgehen, tut das mit einem Backup in der Hinterhand weniger weh. Vor allem aber ist das Encoding ein Hobby, so wie andere Briefmarken sammeln – nur spannender. :-p

Und um den Rahmen gar abzustecken: MPEG-4 ist unser Zielformat der Wahl, konkret Xvid, DivX und x264. Dass das so ist, liegt hauptsächlich daran, dass ich keinen Fernseher besitze und deswegen der Computer als Heimkino herhalten muss. Unter diesen Umständen ist MPEG-4 das flexibelste und leistungsfähigste Format. Genau genommen betrachten wir nur einen recht kleinen Ausschnitt der digitalen Videowelt. Außerhalb von DVD als Quelle und MPEG-4 als Ziel existiert noch viel viel mehr. Doch darauf haben sich andere Websites spezialisiert.

Qualität, Qualität, Qualität

Das Encodingwissen lebt von dem Hintergedanken, die Qualität der DVD so gut wie möglich zu erhalten. Der encodierte Film soll idealerweise transparent sein, also bei genauem Hinsehen und Zuhören nicht vom Original zu unterscheiden.

Natürlich lässt sich dieses Ideal nicht immer erreichen, aber so weit wie möglich daran annähern wollen wir uns. Im Zweifel hat die Qualität deshalb immer Vorrang. Dafür darf das Encoding ruhig ein wenig länger dauern und die Zieldatei ein wenig größer sein.

So viel Qualität braucht Platz: ½ DVD-5 (2240 MB) sollten wir einplanen. Das sind keine guten Voraussetzungen für die seit Jahren weit verbreiteten Standardzielgrößen. Das 2-CD-Encoding (1400 MB) nähert sich dem Encodingwissen-Qualitätsniveau oft halbwegs gut an, doch das 1-CD-Backup spielt klar eine untergeordnete Rolle. Aus Sicht des Bastlers mag es oft eine interessante Herausforderung sein, einen Film brauchbar auf 700 MB zu schrumpfen. Qualitativ bewegen wir uns damit einige Stufen unter dem angestrebten Level.

Aus einem anderen Blickwinkel betrachtet will es mir im Zeitalter von DVD-Brenner und Riesen-Festplatte auch nicht mehr einleuchten, warum man einen Film brutal bis auf CD-Größe zusammenquetschen sollte. Das Argument, die Medien für eine größere Zieldatei wären zu teuer, gilt schon längst nicht mehr.

Das perfekte DVD-Backup

Im gesamten Encodingwissen wird uns immer wieder der Begriff vom HQ-Backup oder HQ-Encoding mit bestmöglicher Qualität über den Weg laufen. Was bedeutet das eigentlich? Der Begriff *HQ* stammt natürlich vom englischen *High Quality,* was sich zwar so eingebürgert hat, aber eigentlich irreführend ist. Es geht nicht halbherzig um hohe Qualität, sondern um nahezu kompromisslose Exzellenz. Das fängt bei der Videoqualität an, hört dort aber noch lange nicht auf. Ziel ist ein Film, der einer gut gemachten DVD in nichts nachsteht.

► Die **Videoqualität** darf nicht spürbar schlechter sein als die DVD. Tummeln sich im Quellbild schon die Kompressionsartefakte – und das kommt häufiger vor als man gemeinhin denkt –, ist ein vorsichtiges Filtern nötig, um diese Probleme zu beheben.
► Für die **Audioqualität** gilt das gleiche: kein hörbarer Unterschied zum Original. Das heißt unter anderem, wenn 6-Kanal-Ton vorhanden ist, sollen es auch sechs Kanäle bleiben. Natürlich darf echtes HQ-Encoding nicht auf die Tonspur mit der Originalsprache verzichten.
► Sofern es die DVD zulässt, stellen wir jeder Audiospur den passenden **Untertitel** an die Seite, und zwar abschaltbar in bestmöglicher Qualität, was auf dynamische Textuntertitel hinausläuft.

► Bleiben die **Metadaten**, die netten Kleinigkeiten, die das HQ-Backup abrunden. Allen Streams ist die passende Sprache zugeordnet, es existiert eine kurze Beschreibung für jeden davon, der Filmtitel ist vermerkt, eine Kapitelliste ist vorhanden und enthält die Kapitelnamen, falls welche existieren. Prinzipiell lässt sich das noch viel weiter ausbauen. Doch da kein Player Zusatzinfos in solchem Unfang auch wieder anzeigen, lohnt sich ein Tagging bis in die letzte Kleinigkeit, wie von Musik gewohnt, eher nicht.
► Es gibt meiner Meinung nach auch **unnötige Dinge**, die nicht zum HQ-Backup gehören: Extras und Menüs. Extras sind kein Teil des eigentlichen Films und würden die einzelne, gut handhabbare Zieldatei zu einem ganzen Ordner voller Filmschnipsel aufblähen. Menüs sind unnötig, wenn man Audio- und Untertitelspuren so in den Container packt, dass man kaum jemals wechseln muss. Dazu kommt der ganz praktische Grund, dass außerhalb der Video-DVD die Unterstützung für Menüs beim Erstellen unbefriedigend und beim Abspielen kaum vorhanden ist.

Diese Liste ist keinesfalls unanfechtbar in Stein gemeißelt. In ihren Eckpunkten würde ich sie jederzeit vehement verteidigen. Bei den Details muss jeder selbst wissen, wo er die Grenze zieht. HQ-Encoding ist eben nicht zuletzt eine individuelle Angelegenheit.

Der DVD-Player im Wohnzimmer

Es ist dieses Teil gemeint, das auch auf die Namen Standalone, kurz SAP, oder Hardwareplayer hört und bei dem man das gleiche Gefühl hat wie beim Handy. Ein modernes Handy kann absolut alles, man kann damit erstaunlicherweise sogar noch telefonieren. Genauso kann der Wohnzimmerplayer tatsächlich noch DVDs abspielen.

Aber viel interessanter ist doch seine DivX/MPEG-4/AVI-Fähigkeit! Genau da liegt das Problem, denn weit verbreitete Hardwarelösungen hinken der Software nahezu immer deutlich hinterher. Auch heutzutage ist es noch so, dass »Wohnzimmerplayer« in den allermeisten Fällen langweiliges, kaum mehr zeitgemäßes AVI-Encoding bedeutet – und das auch noch mit diversen Einschränkungen. Ein HQ-Encoding wird darauf kaum laufen. Zu meinem Qualitätsanspruch mag das nicht so recht passen. Deshalb nimmt das Encodingwissen außer in der einen oder anderen Randbemerkung ausdrücklich keine Rücksicht auf den Wohnzimmerplayer. Manches, was wir hier kennen lernen, läuft auf solchen Geräten, anders dagegen nicht. Wer die Kompatibilität braucht, muss schon selbst darauf achten, seinen Player nicht zu überfordern.

Nichts geht über Hintergrundinfos

Fast nichts. Und manchmal doch eine ganze Menge. Zumindest als Anfänger ist es keine besonders gute Idee, sich sofort voll auf die Hintergründe zu stürzen (mehr dazu im Wegweiser weiter vorn). Letzten Endes allerdings ist der große Teil Hintergrundwissen (der auf den nächsten Seiten beginnt) das, was das Encodingwissen wirklich ausmacht. Jeden letzten Schalter eines Programms zu erklären, ist mir nicht so wichtig. Wirklich am Herzen liegt mir, einen Einblick zu bieten ins Warum und in die Prozesse, die hinter der Fassade der grafischen Oberfläche ablaufen.

Anleitungen und Programme, die schön bebildert und einfach gehalten auch den grünsten Anfänger sicher bis zum fertigen Encoding leiten, gibt es genug. Wer möglichst schnell und einfach ein Encoding zu Wege bringen und sich ansonsten um andere Dinge kümmern will, braucht auch nicht mehr. Das Encodingwissen dagegen ist für alle geschrieben, die nicht damit zufrieden sind, einen Mausklick an der richtigen Stelle zu platzieren, sondern die wissen wollen, was in den einzelnen Arbeitsschritten abläuft und warum die nötig sind.

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, 2. Juli 2008

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 Verweisen zurück zu den passenden Hintergrundkapiteln folgen.

A.1 Grundlagen

In diesem Abschnitt beschäftigen wir uns mit den Ideen und Prinzipien, die hinter der Kompression von Audio- und Videodaten stehen. Die konkrete technische Umsetzung ist dabei weniger interessant, denn sonst wäre der Abschnitt mindestens einige hundert Seiten lang. Wer sich tatsächlich für die enorm umfangreichen Details interessiert, der muss die entsprechende Fachliteratur lesen.

Anschließend werfen wir einen Blick auf die Bestandteile einer fertigen Filmdatei und sehen uns dann im großen Überblick an, wie der Weg von der DVD zum MPEG-4-Film verläuft.

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, verwendet man für 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 (6000 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 vierzig DVD-Rohlinge. Oder – um einen Vergleich zu bemühen, mit dem die Online-Generation vielleicht mehr anfangen kann: Mit einem DSL-6000-Breitbandanschluss und ununterbrochen voller Geschwindigkeit braucht man ungefähr drei Tage, um diese Datenmenge herunterladen.

Platzbedarf unkomprimierter Audiodaten

Der Ton braucht deutlich weniger Platz, so dass sich an der Größenordnung der Datenmenge nichts Wesentliches mehr ändert. 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 Byte × 48 000 Hz × 6 Kanäle = 576 000 Byte/Sekunde.

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

576 000 Byte/Sekunde × 6 000 Sekunden = 3 456 000 000 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 übliche Video-DVD mit ihrer Kapazität von 8 GByte geradezu winzig.

Die Einheitenumrechnung von Dateigrößen oder Bitraten hält einen Stolperstein bereit, denn die Einheiten Kilo, Mega, Giga usw. werden in zwei verschiedenen Bedeutungen gebraucht. Mit überwältigender Mehrheit wird seit den Anfangszeiten der EDV folgende

Rechenregel verwendet:

Wenn es um *Bit* geht (z. B. Bitrate 1 500 kbit/s), dann gilt
1 Megabit = 1 000 Kilobit = 1 000 000 Bit. Also immer der Faktor 1 000.
Wenn es um *Byte* geht (z. B. Dateigröße 1 400 MByte), dann gilt
1 Megabyte = 1 024 Kilobyte = 1 048 576 Byte. Also immer der Faktor 1 024.

Eine wichtige Ausnahme sollte man dabei im Kopf haben. Festplatten- und Rohlingher-steller verwenden für ihre Kapazitätsangaben entgegen der allgemein akzeptierten Regel den Faktor 1 000. Auf diese Weise kann man einen DVD-Rohling mit einer Kapazität von 4,7 Gigabyte auszeichnen, obwohl das Betriebssystem die Disc nach der 1024er-Regel mit 4,37 Gigabyte erkennt.

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. Höchstens für Audiospuren wäre in Aus-

nahmefällen die verlustlose Variante sinnvoll: Wenn wir z. B. eine Konzert-DVD mit einer unkomprimierten PCM-Audiospur encodieren und ein Qualitätsverlust der Tonspur – auch wenn der gar nicht hörbar wäre – nicht in Frage kommt.

Weitere Literatur: [Fied06] [Pana99] [Rich03] [Symes04]

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 Pack-Algorithmus entsteht so der komprimierte Film.

Weitere Literatur: [Pill03] [Wick02] und Lit. aus A.1.2

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. Die verwendeten Verfahren wollen wir uns jetzt näher ansehen.

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 (Luma) und einen Farbanteil (Chroma) zerlegt. Der weniger wichtige Chroma-Anteil wird mit geringerer Genauigkeit gespeichert und dadurch Platz gespart. Das zeigt Abb. A-1.

Links sehen wir den Farbraum YCbCr, der nahezu verlustlos (bis auf Rundungsfehler) konvertierbar zu RGB ist und genauso viel Speicherplatz (24 Bit pro Pixel) benötigt. Luma und Chroma werden mit der gleichen Genauigkeit gespeichert, was man als 4:4:4-Sampling bezeichnet.

Rechts daneben ist der Farbraum YV12 dargestellt, der sowohl auf der DVD als auch in MPEG-4 verwendet wird. Der Helligkeitsanteil hat hier weiterhin volle Genauigkeit. Farbinformation wird aber nur noch einmal für jeden 2×2-Pixel-Block gespeichert (4:2:0-Sampling), was auf durchschnittliche 12 Bit Speicherplatz pro Pixel hinausläuft – das ist die Hälfte von RGB bzw. YCbCr.

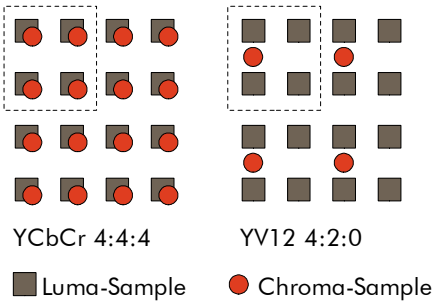


Abb. A-1: Farbräume mit verschiedener Chroma-Genauigkeit.

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. Deshalb neigen alle MPEG-Codecs bei Szenen mit hohem Rotanteil schon früh zur Bildung von Artefakten.

Makroblocks und Transformation

Nach dem Wechsel des Farbraums können wir uns nun mit dem Inhalt des Bildes beschäftigen. Alle gängigen Codecs zerlegen ein Bild nicht in einzelne Pixel, sondern in so genannte Makroblocks, die an dieser Stelle typischerweise 8×8 oder 4×4 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.

Unglücklicherweise enthält ein Makroblock Daten in der räumliche Dimension, d. h. er gibt Auskunft über die Helligkeits- und Farbverteilung innerhalb einer rechteckigen Fläche. Daraus lässt sich kaum eine brauchbare Aussage über wichtige und unwichtige Bilddetails ableiten. Genau das benötigen wir aber, um nur die Details zu entfernen, die nicht wahrnehmbar sind. Deshalb müssen wir den Makroblock in eine andere, passende, Dimension transformieren. Wie genau das geschieht, muss uns im Detail nicht interessieren. Zum einen handelt es sich dabei um höhere Mathematik, zum anderen sind die Verfahren nicht bei allen Videoformaten gleich. Wir betrachten die Kompression hier am Beispiel von Xvid

oder DivX (d. h. MPEG-4 Visual), die die diskrete Kosinustransformation (DCT) verwenden.

Wer tiefer in die Materie einsteigen will, sollte einen Blick auf den Artikel von ►[Selu05] Ethanolix und Videostation werfen, der sich im Anhang von Selurs ►*Wissenswertes rund um Xvid* findet.

Die DCT rechnet den Makroblock in die Frequenzdimension um. Er behält dabei seine 8×8-Matrixstruktur, nur dass die 64 Einträge nach der Transformation keine räumlichen Informationen mehr beinhalten, sondern Frequenzen. Links oben stehen die niedrigsten und rechts unten die höchsten Frequenzen. Damit sind wir am Ziel, denn jetzt können wir Aussagen über die Wichtigkeit einzelner Einträge treffen.

Quantisierung

Die niedrigsten Frequenzen repräsentieren grobe Bilddetails (z. B. die rechteckige Form einer Tischplatte), die für die Bildqualität extrem wichtig sind und möglichst unverfälscht bleiben müssen. Die höchsten Frequenzen stehen für feine Strukturen (z. B. die Holzmaserung der Platte), die entweder nur wenig zur Bildqualität beitragen oder vollständig unwichtig sind. An dieser Stelle kommen die berühmt-berüchtigten Quantisierungsmatrizen und der Quantizer ins Spiel. Die Matrix ist der zentrale Baustein zur Durchführung der Quantisierung. Sie besteht, dem Makroblock entsprechend, aus 64 Werten, angeordnet in 8 Zeilen zu je 8 Spalten. Je höher die enthaltenen Werte, desto mehr Details werden an dieser Stelle des Blocks vernichtet. Der Quantizer dient zusätzlich als Multiplikator. Man könnte ihn als eine Art Kompressionsfaktor bezeichnen. Genau wie bei der Matrix gilt: höhere Werte bedeuten mehr Informationsverlust und anschließend eine höhere Kompression.

Abb. A-2: Ursprüngliche und DCT-komprimierte Blöcke im Vergleich.

Unser Einzelbild besteht nun also aus einer Menge an DCT-transformierten und quantisierten Makroblocks, aus denen mal mehr, mal weniger Details entfernt wurden. Um das Bild später wieder anzeigen zu können, müssen sowohl die Transformation als auch die Quantisierung rückgängig gemacht werden, wofür die inverse diskrete Kosinustransformation (iDCT) zuständig ist. Da wir verlustbehaftet arbeiten, entspricht der wiederhergestellte Makroblock nicht exakt dem Original. Ob der Unterschied sichtbar ist – d. h. wie gut sich das Ergebnis ans Ori-

ginal annähert – hängt im Wesentlichen von der Höhe des Quantizers und vom Design der Quantisierungsmatrix ab.

Betrachten wir zum Schluss anhand Abb. A-2 das Ergebnis einer DCT-Codierung. Ganz links sehen wir stark vergrößert den ursprünglichen 8×8 Pixel großen Block, rechts daneben die per Xvid komprimierte Version. 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.

Entropiecodierung

Soviel wir bisher auch verändert und umgerechnet haben, der Makroblock besteht trotzdem noch aus 64 Werten, d. h. wir haben noch nicht ein einziges Byte an Größe gespart. Die eigentliche Kompression geschieht erst jetzt bei der Entropiecodierung. Das ist das, was Rar, Zip und alle anderen Dateipacker tun; das, was wir weiter vorne als verlustlose Kompression kennen gelernt haben. So bedeutend der Unterschied zwischen verlustlos und verlustbehaftet also ist, so bedienen sich doch beide Verfahren der selben Werkzeuge. Die Arbeitsschritte der reinen verlustlosen Kompression sehen so aus:

Originaldaten ⇒ Kompression.

Die verlustbehaftete Kompression tut im Prinzip genau das gleiche, nur füttert sie den Kompressor nicht mit den Originaldaten, sondern mit einer vor-verarbeiteten Version davon, deren Informationsdichte deutlich geringer ist:

Originaldaten ⇒ Informationsentfernung ⇒ Kompression.

Je geringer die Informationsdichte, desto kleiner werden die komprimierten Daten. Kommen wir zurück zu unserem Makroblock. In seinem Ursprungszustand besteht der aus 64 recht verschiedenen, vergleichsweise schlecht komprimierbaren Werten. Nach Transformation und Quantisierung enthält der Block lange Ketten von Nullen, und die lassen sich hervorragend schrumpfen.

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.

Weitere Literatur:
siehe Lit. aus A.1.2

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 noch einmal massiv 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. 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.

Der Codec arbeitet weiterhin auf der Basis von Makroblocks, die aber bei der Bewegungssuche in der Regel 16 × 16 Pixel groß sind, im Gegensatz zu den schon angesprochenen 8 × 8- oder 4 × 4-Blocks der Transformation und Quantisierung. Die Situation stellt sich demnach so dar: »Makroblock A ist sowohl in Bild 1 als auch Bild 2 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.

»Gleichheit zweier Makroblocks« ist immer im Sinn von »möglichst ähnlich« gemeint. Um kleine Unterschiede auszugleichen, speichert der Codec die Differenz zwischen den Blocks. Das ist notwendig, weil in Filmen aus der realen Welt hundertprozentige Gleichheit selten vorkommt.

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 Abb. A-3.

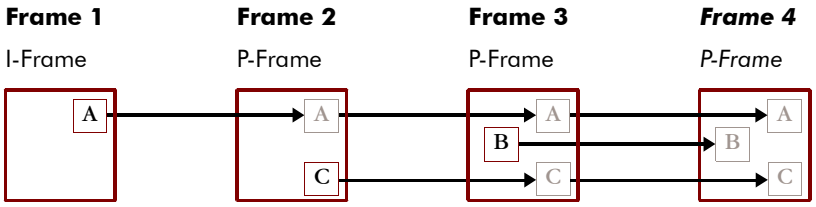


Abb. A-3: Rückbeziehung auf Blocks aus früheren Bildern bei P-Frames.

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 aus Frame 2 übernommen«. Auch Frame 2 enthält einen analogen Hinweis, so dass wir den Makroblock A schließlich in Frame 1 finden. Beim Makroblock B haben wir mehr Glück, denn der findet sich schon in Frame 3.

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 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 angekommen ist. Das Splitting-Kapitel C.1.1 ab Seite 211 ist ein guter Ort, 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 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 Abb. A-4 wird das deutlicher.

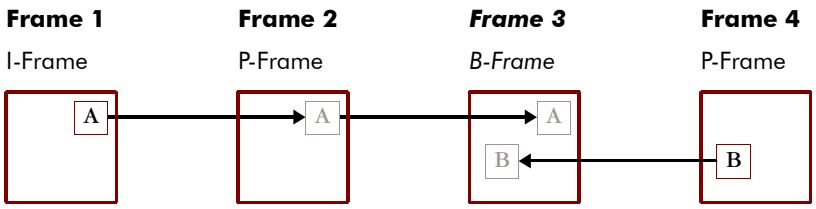


Abb. A-4: Bidirektionale Rück- und Vorwärtsbeziehung bei B-Frames.

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 vorhan-

den 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 wird diese Vertauschung rückgängig gemacht. 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 extrem verschieden sind. 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.

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 für den Ton sowohl verlustlose als auch verlustbehaftete Kompressionsverfahren. 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 weniger 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) oder sechs (5.1). 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 AC-3 verwenden, arbeitet komplett verlustlos.

Neben diesen Standardverfahren verwenden verschiedene Codecs noch andere Methoden, um weiter zu komprimieren. Als Stichwort sei nur die SBR-Technik (Spectral Band Replication) erwähnt, die die zentrale Rolle bei HE-AAC spielt.

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 das 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 angege-

bene 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.

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 AC-3 ist eine ganze Reihe von Formaten denkbar. Exakt müsste man also beispielsweise 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.

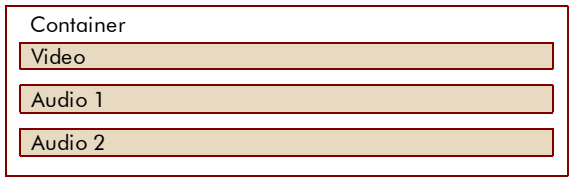


Abb. A-5: Schuhkarton-Vergleich.

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 in Abb. A-5. 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 Abb. A-6.

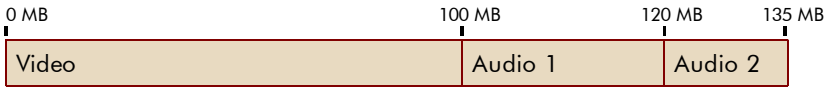


Abb. A-6: Physisches Containerlayout ohne Interleaving.

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 beginnt, das Video abzuspielen, 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.

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. Abb. A-7).

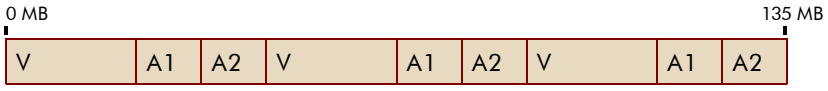


Abb. A-7: Physisches Containerlayout mit Interleaving.

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, aber die gehören zur normalen Arbeitsweise eines Laufwerks.

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 PgcDemux.

Als »Ripping« wird öfters auch der komplette Prozess von der Quell-DVD bis zum fertigen MPEG-4-Film bezeichnet, der dann »DVD-Rip« heißt. Wir halten uns jedoch an die oben beschriebene enge Definition.

► 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, eine Art Inhaltsverzeichnis, anzulegen, der dann von AviSynth weiterverwendet werden kann.

► Audio-Transcoding

Sind die originalen AC-3-Dateien der DVD zu groß für die gewünschte Zielgröße, müssen wir sie in ein anderes, kleineres, Format umrechnen. Für dieses Transcoding ist BeSweet zuständig, und der Einfachheit halber verwenden wir BeLight als grafische Oberfläche. Eventuell kann das Transcoding auch automatisch vom Encoding-Frontend erledigt werden, wenn dessen Audiounterstützung 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-Grafikformat 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. Rauschen beseitigen, das Bild schärfen oder Artefakte in der Quelle übertünchen. Normalerweise steuern wir AviSynth über die grafischen Dialoge des Encoding-Frontends, ohne dass wir das Skript zu Gesicht bekommen.

► Encoding
Das Encoding ist der Vorgang, der unser Zielvideo erstellt. Dabei rechnet der Videocodec das MPEG-2-Quellmaterial der DVD ins MPEG-4-Format um. Das kann in einem einzigen Durchlauf geschehen, wenn wenn wir eine bestimmte Qualität erreichen wollen ohne dass die Dateigröße des fertigen Films wichtig ist. Wenn wir dagegen eine vorher definierte Zielgröße exakt treffen wollen, sind zwei Durchläufe nötig, das so genannte 2-Pass-Encoding. 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.

► 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 von DVD-Brenner und Festplatte 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 immer noch nötig, wenn wir die Kombination der 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 Video und Audio. 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 2* 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 dieser Anleitung sagt es schon: MPEG-4 (offiziell ISO/IEC 14496) ist unser Zielformat. Der Video-Teil ist hier in zwei Bereiche aufgeteilt:

► **MPEG-4 Part 2 (Visual)** (1998 verabschiedet) 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 (Advanced Video Coding)** (2003 verabschiedet). Auch bekannt unter seiner Dokumentnummer der ITU-Organisation: **H.264**. Verwendet wird AVC z. B. von x264 und Nero Digital. Außerdem ist es eines der zentralen Videoformate der Blu-ray Disc.

Gegenüber MPEG-2 lässt sich in der gleichen Dateigröße noch einmal deutlich mehr Qualität unterbringen, wobei AVC wiederum tendenziell kleinere Ergebnisse liefert als ASP. Deshalb können wir einen Film meistens ohne erhebliche Verluste auf ⅓ bis ½ DVD-5 (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 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* und AC-3 (Abkürzung für *Adaptive Transform Coder 3*, der Name des 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. AC-3 unterstützt bis zu sechs Kanäle, wobei der sechste als zusätzlicher Basskanal ausgelegt ist. 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 AC-3 ein verlustbehaftetes Format ist, allerdings mit deutlich höheren Bitraten arbeitet und noch mehr Kanäle unterstützt (bis zu 6.1). Die schier endlose Diskussion, ob AC-3 oder DTS besser wäre, ist größtenteils Unfug. Beide Encodingverfahren sind technologisch auf dem selben Niveau.

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** ist 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. 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, der in der Praxis jedoch keine Bedeutung hat, denn für Multikanalton existieren bessere und breiter unterstützte Formate.
► **MPEG-4 Part 3** wird oft als offizielles MP3-Nachfolgeformat bezeichnet und dürfte vielen unter dem Namen *Advanced Audio Coding* (AAC) bekannt sein. Multikanalton gehört natürlich zum Funktionsumfang, genauso wie eine Reihe von Codiertechnologien, die je nach Situation verwendet oder deaktiviert werden können. Damit lässt sich AAC sehr flexibel einsetzen.

Vorbis

Xiph.org entwickelt Vorbis als alternatives Audioformat, das komplett frei von patentierten Technologien sein soll und so 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 fast so hohe Bitraten wie AC-3 benötigt.

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. Auf normalen Spielfilm- oder Serien-DVDs wird uns das Format kaum jemals begegnen und auch für das fertige Encoding ist es wegen seiner Größe uninteressant.

Weitere Literatur: [Tayl08]

A.2.2 Das Format der DVD

Die Eckdaten der DVD

Tab. A-1: Technische Daten der Video-DVD.

Video	MPEG-2 PAL: 720 × 576 Pixel, 25 fps NTSC: 720 × 480 Pixel, 23,976 und 29,970 fps
Audio	AC-3, DTS, MPEG oder PCM maximal 8 Audiospuren
Untertitel	gespeichert als Pixelbilder (Vobsubs) maximal 32 Untertitelspuren
Datenrate	Video + Audio + Untertitel insgesamt maximal 10,08 Mbit/s

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. Bestellt ein Amerikaner einen Film aus Europa und will ihn auf seinem Player/Fernseher abspielen, hat er keine besonders große Freude daran, es sei denn im ungewöhnlichen Fall, dass seine Geräte außer dem üblichen NTSC auch ausdrücklich das PAL-Format verstehen. Anders herum klappt das besser, da viele PAL-Geräte auch NTSC sprechen.

Wer die Mattscheibe seines heimischen Fernsehers ausmisst, wird feststellen, dass sie entweder ein Seitenverhältnis (Verhältnis von Breite zu Höhe) von 4:3

(entspricht 1,33:1) oder 16:9 (1,78:1) hat. Das Bild einer PAL-DVD dagegen besitzt ein Seitenverhältnis 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 im richtigen Format am Monitor ankommt.

Die Tiefen der Seitenverhältnis-Thematik sind nicht nur für die DVD wichtig, sondern auch für das encodierte Video. Deswegen beschäftigen wir uns damit ausführlich im Abschnitt A.3.2 *Anamorphes Video* ab Seite 42.

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

Tab. A-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 AC-3-Format Pflicht ist. Für NTSC-Discs dagegen ist MPEG als einzige Audiospur verboten.

PCM darf bis zu 8 Kanäle enthalten, AC-3 bis zu 6 Kanäle. MPEG-Ton existiert 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 AC-3 5.1, Deutsch und Originalsprache (also oft Englisch).
- Manchmal eine deutsche 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 31.

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 wichtiger und gut von Wohnzimmerplayern unterstützter Container für MPEG-4-Video behauptet. Dank Matroska und MP4 zeichnet sich inzwischen aber eine dringend nötige Änderung ab.

Matroska (MKV)

Das ambitionierte 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 nahezu endlosen Fähigkeiten stellt Matroska für einen Standalone-Hersteller eine weniger attraktive Lösung dar als ein Container mit enger definiertem Funktionsumfang, wie z. B. MP4, da der einfacher implementiert werden kann und weniger potenzielle Problemquellen aufweist. Am Computer bietet Matroska allerdings klar die umfangreichsten Freiheiten.

Aktuell findet jedoch eine Entwicklung statt, die verblüffend an den Aufstieg von AVI und DivX erinnert. Wie damals hat sich die illegale Release-Szene nun Matroska als Container auserkoren, um HD-Filme zu verbreiten. Ungeachtet aller Fragen von Recht und Unrecht hat diese Szene zweifellos das nötige Schwergewicht, um neue Standardformate zu etablieren. Und tatsächlich beginnen Hardwarehersteller inzwischen, sich an Matroska heranzuwagen. Dem Container könnte also noch eine große Zukunft bevorstehen.

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 er interessant für Hardwarehersteller.

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 verwen-

den, 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, AC-3, 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 einigen Jahren still. OggMedia kann nichts, was Matroska oder MP4 nicht genauso können. Wie alle Notlösungen, so sollte man auch OGM in Ruhe sterben lassen, wenn eine bessere Alternative verfügbar ist.

Die Fähigkeiten der Container im Vergleich

Tab. A-3 auf Seite 31 bietet einen Überblick über die Fähigkeiten der Container.

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. MP4 steht dank Neros Zertifizierungsprogramm als moderner Ersatz in den Startlöchern. Und auch Matroska stößt bei Hardwareherstellern auf steigendes Interesse, seit sich der Container als De-facto-Szenestandard etabliert hat. Wer heute einen Player fürs Wohnzimmer kauft, sollte sich den Schritt weg von AVI ernsthaft überlegen.

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.

	Matroska	MP4	AVI	OggMedia
	MPEG-4-Video			
Xvid, DivX	Ja	Ja	Ja	Ja
H.264	Ja	Ja	Nein [1]	Nein [1]
	Audio			
MP3	Ja	Ja	Ja	Ja
Vorbis	Ja	Nein	Nein	Ja
AAC	Ja	Ja	teils [2]	Ja
AC-3, DTS	Ja	Nein	Ja	Ja
	Untertitel			
SubRip	Ja	Ja	teils [3]	Ja
Vobsub	Ja	Nein [4]	Nein	Nein
	Metadaten			
Stream-Info [5]	Ja	Ja	Nein	teils
Kapitel	Ja	Ja	Nein	Ja
Menüs	Nein [6]	Nein [6]	Nein	Nein
	Player			
Computer	Ja	Ja	Ja	Ja
Hardware	teils [7]	teils [7]	teils [7]	Nein

Tab. A-3: Funktionsumfang von Multimedia-Containern.

1 Ist mit großen Verrenkungen und Einschränkungen teilweise möglich, aber ohne einen praktischen Wert.
2 AAC in AVI kann nur von AVI-Mux GUI gemuxt werden und wird nur beim Abspielen am Computer unterstützt.
3 Kann nur mit Tricks in AVI gemuxt werden und hat keine Unterstützung in den populären Playern VLC und mplayer.
4 Im Standard nicht vorgesehen. Von Nero über private Streams trotzdem realisiert, hat aber keine allgemeine Unterstützung bei der Wiedergabe.
5 Damit sind Angaben wie Sprache oder Trackname gemeint.
6 Im Standard vorhanden, aber ohne praktische Unterstützung.
7 Matroska-Hardware fängt aktuell (Anfang 2008) langsam an zu erscheinen. MP4 existiert in Form Nero-zertifizierter Player, ist aber noch nicht weit verbreitet. Die üblichen DivX-fähigen Player verwenden AVI als Container, unterstützen aber bei den Codecs nur einen mehr oder weniger eingeschränkten Funktionsumfang.

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 mit mindestens einem Bein in der Illegalität.

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

DivX

Der offizielle und legale Nachfolger von DivX;-) ist 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. ganz deutlich den Wohnzimmerplayer und den weniger erfahrenen Anwender 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` jedermann zugänglich. Dank eines Teams von enthusiastischen und verdammt fähigen Programmie-

rern 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.

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 ►Doom9-Codecvergleichstest aus dem Jahr 2005, bei dem x264 schon ganz vorne mitspielte. Seitdem hat der Codec noch einmal deutlich zugelegt und stellt aktuell den vielleicht besten AVC-Encoder überhaupt dar. ►[Doom05]

x264 wird genauso wie Xvid als Open-Source-Projekt entwickelt. Da die 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. Xvid und besonders x264 eignen sich dagegen für alle, die sich an der Detailkonfiguration so richtig austoben wollen.

Qualitativ hat x264 die Nase vorn. Gerade bei niedrigen Bitraten ist der Vorteil den beiden ASP-Codecs gegenüber kaum zu übersehen. Je höher die Bitrate, desto geringer wird der Vorsprung und die Nachteile des H.264-Formats 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 eine von VfW unabhängige Encoderversionen für die Kommandozeile, die native Streams für moderne Container erzeugen.

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 AC-3« gilt längst nicht mehr. Werfen wir deshalb einen Blick auf die verschiedenen Möglichkeiten. Downloadadressen für alle Encoder folgen später im Kapitel B.1.1 ab Seite 127.

MP3

Der Klassiker. Mit MP3 brauchen wir uns um Abspielprobleme keine Sorgen zu machen: läuft überall. Der große Nachteil ist die Beschränkung auf Stereo; den 6-Kanal-Ton der DVD können wir mit MP3 nicht beibehalten.

MP3 erreicht bei Musik transparente (also vom Original nicht mehr zu unterscheidende) Qualität in der Region um 140 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 im Bereich der berühmten 128 kbit/s der Transparenz erstaunlich nahe (5,0 in den Diagrammen der Hörtestseite bezeichnet Transparenz). Aber Vorsicht: Hier gilt ganz besonders die Voraussetzung eines guten Encoders und passender Konfiguration.

► [Mare05]

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. Für Audiocodecs im Allgemeinen und LAME im Besonderen gilt, keine Experimente mit obskuren Optionen anzustellen, sondern sich an die gut getesteten Standardeinstellungen zu halten. Gäbe es Optionen, die die Qualität spürbar erhöhen würden, dann wären die schon längst in den »professionellen« Empfehlungen enthalten. Konkrete Konfigurationen sehen wir uns im Praxisteil an.

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 solchen 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 im Multichannel-Modus mit AC-3-ähnlichen Bitraten kalkulieren, und damit bleibt Vorbis praktisch auf Stereo beschränkt.

Seit einigen Jahren ist es normal, dass der offizielle Vorbis-Encoder die interessantesten und neusten Optimierungen nicht enthält. Deswegen sollten wir an den auch wenig Gedanken verschwenden und uns lieber an die aoTuV-Variante halten. In Form der zusätzlich auf Geschwindigkeit optimierten Lancer-Builds ist das der beste Vorbis, den wir bekommen können.

AAC

Advanced Audio Coding ist das offizielle Audioformat des MPEG-4-Standards. Es ist besonders interessant, weil es zur Zeit als einziges Format 6-Kanal-Ton mit deutlich geringeren Bitraten als AC-3 encodieren kann. Je weiter wir den Funktionsumfang von AAC ausreizen, desto länger dauert das Encoding und desto mehr CPU-Leistung ist bei der Wiedergabe nötig. Andererseits steigt natürlich die Kompressionsleistung. Im Wesentlichen gibt es drei Komplexitätsstufen, so genannte Profile.

► *Low Complexity (LC)* entspricht dem auch bei anderen Codecs weit verbreiteten Vorgehen, nahezu den gesamten Frequenzumfang anhand eines psychoakustischen Modells zu encodieren. Der Bereich sinnvoller Bitraten liegt grob zwischen 80 und 130 kbit/s für Stereo-Ton. Sechs Kanäle benötigen etwa 200 bis 250 kbit/s.
► *High Efficiency (HE)* (manchmal auch *AAC Plus* genannt) ergänzt den LC-Modus um eine Technologie namens *Spectral Band Replication* (SBR). Dabei wird der Bereich hoher Frequenzen vor dem Encoding abgeschnitten und nur einige Wiederherstellungsinformationen gespeichert, anhand derer der Decoder beim Abspielen die hohen Frequenzen rekonstruiert. Das führt zu einem geringen Qualitätsverlust, aber auch zu dramatisch sinkenden Bitraten. 6-Kanal-Ton kommt im HE-Modus mit etwa 130 bis 150 kbit/s aus, Stereo-Ton benötigt ungefähr 50 bis 70 kbit/s.
► *Parametric Stereo* ist wiederum ein Zusatz zu HE und wird oft als *HE Version 2* (kurz HEv2, HE2 o. ä.) bezeichnet. Diese Technologie ist speziell auf Stereo-Ton und extrem niedrige Bitraten ausgelegt. Damit können wir eine Tonspur mit zwar

hörbaren Verlusten aber doch noch einigermaßen brauchbarer Qualität auf 30 bis 40 kbit/s schrumpfen.

Der bekannteste Encoder ist sicherlich Nero, der sowohl im Nero-Gesamtpaket enthalten ist als auch als kostenloser Kommandozeilen-Encoder heruntergeladen werden kann. Daneben steht Winamp als zweiter wichtiger AAC-Encoder mit Unterstützung für unser Audio-Transcodingtool BeSweet. Obwohl beschränkt auf CBR, liefert er gute Ergebnisse. Ich persönlich verzichte trotzdem ungern auf den VBR-Modus und verwende deshalb Nero.

Im Lager der nicht-kommerziellen Software sieht es in Sachen AAC-Encoder düster aus. Besonders dass aus diesem Bereich ein HE-fähiger Encoder fehlt, macht Nero und Winamp zu den einzigen für uns ernsthaft nützlichen Alternativen.

AC-3

Im fertigen Encoding AC-3 einzusetzen ist eine einfache Lösung, weil wir die Tonspur nur unverändert von der DVD übernehmen brauchen. Es entfällt sowohl die Arbeit als auch der Qualitätsverlust des Transcodings, und natürlich bleibt der Mehrkanal-Ton erhalten. Nachteil: 5.1-AC-3 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 AC-3-Spur. Für das urklassische, hochkomprimierte 1-CD-Encoding können wir AC-3 dagegen vergessen. Allerdings muss AC-3 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 AC-3 in guter Qualität zu encodieren, ist vor allem mit einem teuren kommerziellen Encoder gut möglich, der auch alle Feinheiten des Formats unterstützt. Die Situation bei den nichtkommerziellen Encodern hat sich inzwischen gebessert. Nach wie vor gilt, von BeSweets AC-3enc tunlichst die Finger zu lassen. Als Alternative bietet sich Aften an, der zwar nicht den vollen AC-3-Funktionsumfang bereitstellt, aber sinnvolle Qualität liefert und inzwischen sogar mit BeSweet zusammenarbeitet.

Welchen Encoder wir auch immer verwenden, AC-3 bleibt auf hohe Bitraten ausgelegt. Dass die DVD für 6-Kanal-Spuren die Untergrenze von 384 kbit/s definiert, hat seinen Grund. Ich würde empfehlen, bei 5.1-AC-3s nicht unter 256 kbit/s und bei Stereo-AC-3s nicht unter 160 kbit/s zu gehen.

DTS

DTS (Digital Theater Systems) ist inzwischen auf DVDs leider recht weit verbreitet. Wie AC-3 arbeitet DTS mit konstanter Bitrate, verwendet aber höhere Bitraten (768 oder 1536 kbit/s) und unterstützt einen Kanal mehr (6.1). Allerdings ist mir noch nie eine 6.1-DTS-Spur über den Weg gelaufen.

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 AC-3 unterstützen, kommen auch mit DTS zurecht. Persönlich finde ich es allerdings äußerst blödsinnig, für eine Audiospur so extrem viel Platz zu opfern, wenn sowieso eine qualitativ praktisch identische AC-3-Spur zur Verfügung steht. Da sich DTS außerdem nur recht umständlich verarbeiten lässt, werden wir uns um das Format 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 AC-3 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 AC-3 oder AAC. Als Ausweichmöglichkeit ist inzwischen auch eine selbst encodierte AC-3 denkbar. Allerdings sollten wir wegen der kleineren Bitraten AAC bevorzugen, wo immer möglich. Außerdem gebe ich 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. AC-3) 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. Zusätzlich stellt sich im

Zeitalter der DVD±R die Frage: Warum nicht gleich AC-3 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 Original-AC-3 oder HE-AAC für 5.1-Ton und Vorbis für Stereo.

A.2.6 Die Untertitelformate

Als Untertitel bezeichnet man i. d. R. Text, der während der Wiedergabe zu einem genau definierten Zeitpunkt eingeblendet wird. Auf Anhieb bringt man das meistens mit den Dialoguntertiteln für Hörgeschädigte oder der Übersetzung von fremdsprachigen Filmen in Verbindung; nichts, was man unbedingt beachten müsste. Das entspricht aber nicht ganz der Realität. Untertitel sind weit mehr, so dass wir sie bei keinem professionellen Encoding einfach ignorieren dürfen.

Technische Untertitelformate

Von der technischen Seite her gesehen haben wir es mit drei Arten von Untertiteln zu tun.

► **Fest eingebrannte Untertitel** sind ein Teil des Videobildes und keine unabhängig gespeicherten Zusatzinfos. Deswegen können sie auch weder ausgeblendet noch sonst irgendwie manipuliert werden. Sind solche Untertitel auf der DVD vorhanden, müssen wir sie so hin- und übernehmen, wie sie sind. Da sie ein Bestandteil des Bildes sind, geschieht das ganz automatisch, ohne dass wir uns um irgend etwas extra kümmern müssen.
► **Grafikuntertitel** sind in Bitmap-Grafiken verpackter Text, der unabhängig von der Videospur gespeichert wird. Sie lassen sich ein- und ausblenden, und im fertigen Encoding haben wir die Wahl, ob wir sie übernehmen oder weglassen wollen. Alle Untertitelspuren der DVD verwenden Grafikuntertitel in einem Format, das *Vobsub* genannt wird.
► **Textuntertitel** funktionieren genauso wie Grafikuntertitel. Allerdings handelt es sich tatsächlich um Text und nicht um in Grafik verpackte Schrift. Die bekanntesten Formate sind SubRip (SRT) und Sub Station Alpha (SSA und ASS).

Notwendige und weniger notwendige Untertitel

Bleiben wir bei den beiden ausblendbaren Formaten. Sinn ist natürlich unter anderem der, dass wir beim Abspielen entscheiden können, ob wir die Untertitel sehen wollen oder nicht. Meistens lassen wir sie wahrscheinlich ausgeschaltet, denn schließlich hört man ja die Audiospur und muss die gleichen Dialoge nicht noch mitlesen.

Die Texteinblendung ist also meistens entbehrlich. Das gilt jedoch nicht immer. Untertitel können alle möglichen Informationen enthalten und sind nicht darauf beschränkt, die Dialoge wiederzugeben. Die Zeitangaben bei *Spy Game* beispielsweise sind Untertitel. Das ist die einfachste Möglichkeit, um zur deutschen Audiospur die Uhrzeiten im gewohnten 24-Stunden-Format einzublenden und für die englische Audiospur das passende 12-Stunden-Format mit A. M. und P. M. zu verwenden. Eine ähnliche Situation haben wir bei Filmen, deren Schauplätze über die ganze Welt verteilt sind. Springen wir gerade z. B. nach Frankreich, kann mit Untertiteln für die deutsche Audiospur »Frankreich« eingeblendet werden, für die englische »France« usw.

Untertitel dieser Art sind nicht mehr entbehrlich, sondern manchmal für das Verständnis des Films sogar wichtig. Das gilt auch für manche Dialoge, z. B. Übersetzungen von Fremdsprachen. *The Lord of the Rings* braucht Untertitel für die elbischen Passagen, *The Sum of All Fears* braucht sie noch dringender für die umfangreichen russischen Dialoge. Weil diese Art Untertitel so notwendig ist, sollen sie auch auf jeden Fall angezeigt werden, ganz egal ob wir die Untertitelanzeige aktiviert haben oder nicht.

Ein solcher Mechanismus existiert auf der DVD in Form von Zwangsuntertiteln (Forced Subs). So ist sichergestellt, dass uns wichtiger Text nicht durch die Lappen geht. Forced Subs treten in zwei Varianten auf.

► Entweder existiert eine **separate Spur,** die ausschließlich die Zwangsuntertitel enthält. Beim Abspielen sorgt der Player dann dafür, dass diese Spur auf jeden Fall aktiviert wird. Das ist eher die Ausnahme als die Regel.
► Viel häufiger sind die Zwangsuntertitel **in einer normalen Spur** enthalten, die sämtliche Dialoge und alle anderen Textinformationen enthält. Für einige der Untertitel in dieser Spur ist ein Forced-Flag gesetzt, so dass die (und nur die) auch dann angezeigt werden, wenn beim Anschauen die Untertitel abgeschaltet sind.

Untertitel im Encoding

Nun stehen wir vor der Wahl, welche Untertitel im endgültigen Film vorhanden sein sollen. Eines ist klar: Zwangsuntertitel 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 nächste Entscheidung betrifft das Format der Untertitel im fertigen Film. Auf der DVD sind sie, wie schon erwähnt, als Grafiken gespeichert, die über dem Video eingeblendet werden. Für unseren fertigen Film gibt es drei Möglichkeiten, Untertitel zu übernehmen:

- die Vobsubs der DVD fest ins Video einbrennen,
- Vobsubs als dynamische (ausblendbare) Untertitel übernehmen,
- Vobsubs 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 Zwangsuntertitel (wenn der Film solche hat) der Faulheit 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. Entbehrliche Dialog-Untertitel sollten wir ohne sehr guten Grund niemals einbrennen.

Dynamische Untertitel haben auf jeden Fall den Vorteil, dass sie in den meistens vorhandenen schwarzen Balken unter- oder oberhalb des Videos angezeigt werden können. Das steigert die Lesbarkeit und überdeckt keinen Teil des Bildes. Dazu kommt, dass eingebrannte Schrift notwendigerweise harte Kanten hat, die einiges an Bitrate schlucken. Dynamische Untertitel sind davon nicht betroffen.

Vobsubs in Text umzuwandeln ist mit einigem Aufwand verbunden, hat aber auch seine Vorteile. Grafikuntertitel werden fast nie in ihrer ursprünglichen Größe angezeigt, da wir den Film zoomen müssen, um ihn im Vollbild anzuschauen. Das geht zu Lasten der Textqualität und senkt die Lesbarkeit. Ein Textuntertitel kann dagegen scharf und einwandfrei lesbar mit jeder beliebigen Schrift in jeder beliebigen Größe dargestellt werden. Wenn wir auf kompromisslose Qualität Wert legen, führt deshalb an Textuntertiteln kein Weg vorbei.

Soweit zur Theorie der Untertitel. Die Realität hält beim Auslesen, Umwandeln und Abspielen noch den einen oder anderen kleinen Stolperstein bereit. Wie wir damit umgehen, das ist Thema im entsprechenden Praxiskapitel.

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 (Zuschneiden)

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.

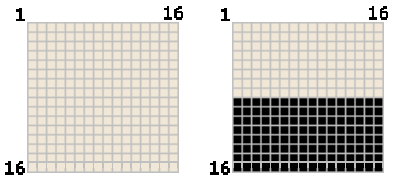


Abb. A-8:Zwei Makroblocks im Vergleich.

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 wenig harte Kanten. Ein kleiner Übergangsbereich bleibt meistens 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 Abb. A-8 an (die grauen Linien verdeutlichen die einzelnen Pixel).

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

trem einfach zu encodieren, denn der Codec braucht – grob vereinfacht ausgedrückt – lediglich den einen Farbwert zu 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.

Weitere Literatur: [Ober01]

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, denn das Thema des richtigen Seitenverhältnisses ist reichlich komplex. Die nächsten beiden Kapitel versuchen, etwas Licht ins Dunkel zu bringen. In einem Satz zusammengefasst: 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 Abschnitt 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.

Was ist eigentlich ein Pixel?

Im Computeralltag gehen wir ohne nachzufragen davon aus, dass ein Pixel ein fest definiertes Etwas ist. Das stimmt auch soweit, dass es sich dabei um die kleinste ansprechbare Einheit eines Bildschirms oder Pixelbildes handelt. Was

nicht so klar ist und den Grund für das Anamorph-Problem bildet, ist die *Form* eines Pixels.

Für die Computerwelt gilt heutzutage, dass Pixel *immer* quadratisch sind. Für die DVD dagegen gilt, dass Pixel *niemals* quadratisch sind, sondern rechteckig, meistens breiter als hoch. Da wir die Welt aus Sicht des Computerpixels sehen, ist das Bild auf der DVD in verzerrter Form gespeichert. Wenn wir es in seiner ursprünglichen Auflösung unverändert anzeigen, hat das Eierköpfe zur Folge.

Da auch unser Zielformat MPEG-4 mit unterschiedlich geformten Pixeln arbeiten kann, reicht die Frage nach der Auflösung des Bildes nicht aus. Wir müssen gleichzeitig immer auch nach der Form der Pixel fragen. In den Details kann das in Stress ausarten, wie wir gleich sehen werden.

Dass ein Computermonitor ausschließlich quadratische Pixel darstellen kann, gilt genau genommen nur für Flachbildschirme (TFT). Röhrenmonitore (CRT) sind theoretisch zu jeder beliebigen Auflösung und Pixelform fähig. Nur arbeiten alle Standard-Desktopauflösungen mit quadratischen Pixeln.
Fast alle. Eine populäre Ausnahme ist die 1280 × 1024 auf einem CRT. Die Bildröhre des Monitors hat ein Seitenverhältnis (Breite:Höhe) von 1,33:1. Die 1280er Auflösung kommt auf ein Seitenverhältnis von 1,25:1. 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.

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 oder Pixels als Verhältnis von Breite zu Höhe, das entweder in der Notation *b:h* (z. B. 16:9) oder *x:1* (z. B. 1,78:1) angegeben wird. 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 horizontaler Auflösung (Breite) zu vertikaler Auflösung (Höhe) dar. Was genau »komplettes Bild« bedeutet, ist nicht festgelegt.

Beispiel: Ein Bild mit der Auflösung 640 × 480 und quadratischen Pixeln (z. B. ein klassisches DivX-Encoding) hat ein DAR von 640:480, was gekürzt 4:3 entspricht oder als Kommazahl 1,33:1.

Die Xvid-VfW-Oberfläche hat das DAR früher »Picture Aspect Ratio« genannt. Das ist schon lange korrigiert, kann aber in älteren Anleitungen noch auftauchen. 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.

Das PAR ist unser Seitenverhältnis der Wahl, mit dem wir beim DVD-Backup tatsächlich arbeiten.

► Sample Aspect Ratio (SAR)

Hierbei handelt es sich im eine alternative Bezeichnung für das PAR, die aus dem H.264-Standard stammt. Außer dem Namen gibt es keinen Unterschied zwischen PAR und SAR.

► Anamorphes Video

Es gibt zwei Definitionen des Begriffs »anamorph«, die wir beide kennen müssen.

- *Digitale Definition:* Anamorph heißt jedes verzerrt gespeicherte Videobild. Das trifft auf alle Videos mit einem PAR ungleich 1:1 (d. h. nicht-quadratischen Pixeln) zu. Die Stärke der Verzerrung spielt keine Rolle.
- *Analoge Definition* (aus der analogen Profi-Videotechnik stammend): Anamorph heißt nur genau eine spezielle Variante des verzerrten Bildes, nämlich das, was auf der DVD mit »16:9 anamorph« bezeichnet wird.

Für unser digitales MPEG-4-Endprodukt ist die analoge Definition Unsinn, weil nur die Frage »quadratisch oder nicht« relevant ist. Die Stärke der Verzerrung spielt keine Rolle. Entsprechend hat sich dafür der Begriff »anamorph« in der digitalen Definition eingebürgert. Für die DVD wird aber sehr verbreitet die analoge Definition verwendet. Das führt dazu, dass eine 4:3-DVD »nicht-anamorph« heißt, obwohl sie trotzdem keine quadratischen Pixel hat.

In freier Wildbahn werden die beiden Definitionen wild durcheinander gewür-

felt. Deswegen sollten wir uns immer zuerst darüber klar werden, was im jeweiligen Zusammenhang gemeint ist. Für das Encodingwissen gilt ausschließlich die digitale Regel, d. h. jedes Video mit nicht-quadratischen Pixeln heißt »anamorph«. Bei DVDs spreche ich passend dazu nicht von anamorphen und nicht anamorphen Filmen, sondern immer von 16:9 und 4:3. So lässt sich die Begriffsverwirrung hoffentlich in Grenzen halten.

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. 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 analogen 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-AR.

Betrachten wir das Bild der DVD etwas mehr im Detail. Als Beispiel nehmen wir *Die fabelhafte Welt der Amélie*. Der Film ist mit einem Seitenverhältnis von 2,35 gedreht, muss aber mit dem vorgegebenen 1,78 einer 16:9-DVD auskom-

men. Die ungenutzte Auflösung wird einfach mit schwarzen Balken aufgefüllt, so dass Frame Nummer 27 661 so aussieht wie das linke Bild in Abb. A-9.

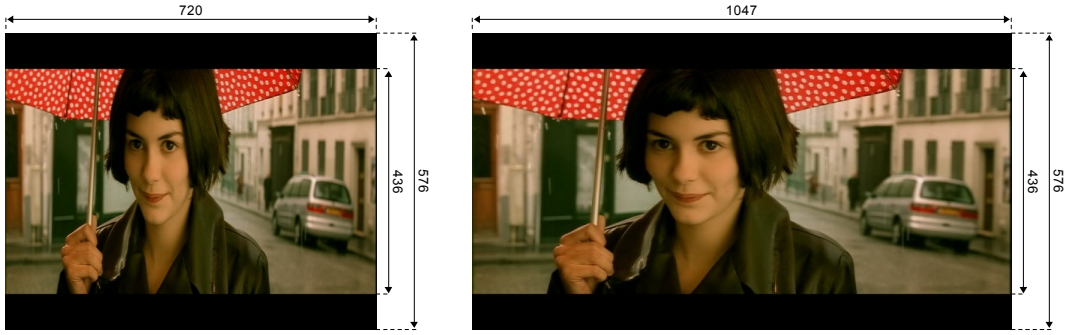


Abb. A-9: Ein 16:9-Frame. Links unverändert, rechts entzerrt.

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 das rechte Bild in Abb. A-9.

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 entspricht dem linken Bild in Abb. A-10. Schwarze Balken gibt es bei 4:3-Material, wenn überhaupt, nur minimal. Im Beispiel sind nur die Ränder des Bildes ein wenig unsauber.

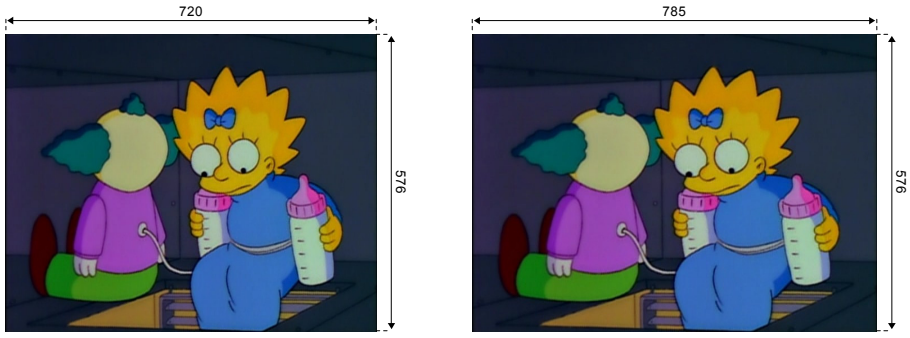


Abb. A-10: Ein 4:3-Frame. Links unverändert, rechts entzerrt.

Falls jemand ins Grübeln kommt. Das Bild stammt aus Season 4: Maggies Mission-Impossible-Einlage in *A Streetcar Named Marge*. Welche Framenummer, habe ich nicht aufgepasst.

Bei der Wiedergabe passiert das gleiche wie im 16:9-Fall. Der Decoder streckt das Bild auf die korrekte Breite (vgl. Abb. A-10 rechts). 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 jeweils einen für 16:9 und 4:3.

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

Tab. A-4: PAR nach MPEG-4.

Mit dem richtigen Wert aus Tab. A-4 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.

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.

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

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 MPEG-4-Standard folgen (der sich wiederum sehr eng an ITU-R BT.601 anlehnt). Dieser fordert ein etwas breiteres Bild als die weithin bekannten Werte. Tab. A-5 bietet einen Vergleich von ITU/MPEG-4-DAR und allgemein bekanntem »Standard-DAR«.

Um keine Zweifel aufkommen zu lassen: Diese Art der Entzerrung ist zu empfehlen, und zwar unabhängig vom Wiedergabegerät. Die ITU-Empfehlung zu ignorieren ist nur dann sinnvoll, wenn wir genau wissen, dass unsere DVD nicht nach dem Standard gemastert wurde.

Tab. A-5: Generisches und MPEG-4-DAR.

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

Wie genau die einzelnen PARs zustande kommen und welche Bedeutung das generische Nicht-ITU-PAR tatsächlich hat, damit beschäftigt sich das Spezialkapitel C.2.2 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 neben Interlacing 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. 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 i. d. R. sowohl horizontal als auch vertikal glatt durch 16 teilbar sein (Mod16-Kriterium, vgl. Seite 218 ff.).

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 fabelhafte Welt der Amélie* wieder bestens, deren korrektes Wiedergabe-DAR abzüglich der Balken in der 2,35er-Region liegt. Achtung, der Film hat auch am linken und rechten Rand schmale schwarze Balken. In den verkleinerten Screenshots sind die schlecht zu sehen. Croppen wir sämtliche 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 Player anwendet. Um korrekt zu entzerren, müssen wir mit Hilfe des MPEG-4-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 zugeschnittene Originalbild enthält

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

Das entzerrte Bild enthält

$(1024 - 704) \times 432 = 138\,240$ 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 diese Abmessungen selbst bei einem ½-DVD-Encoding kritisch sind, 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 oft 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. Meine Tests lassen darauf schließen, dass in diesem Bereich der Unterschied zwischen anamorph und nicht-anamorph nur gering ausfällt.

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 (vgl. auch das Kapitel zur Mod16-Regel, Seite 218 ff.). 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.

► Das *Display Aspect Ratio (DAR)* beschreibt das Seitenverhältnis des kompletten Bildes, stellt also nichts anderes als das Verhältnis von Breite zu Höhe dar.
► Das *Pixel Aspect Ratio (PAR)* bezieht sich nicht auf das ganze Bild, sondern beschreibt die Form eines einzelnen Pixels.

Für DVD-Quellen ist das PAR der deutlich überlegene Wert. Beim DAR stehen wir vor zwei Problemen. Zuerst ist überhaupt nicht genau definiert, welches Seitenverhältnis gemeint ist.

► *Auflösungs-DAR,* das sich allein auf die gespeicherte Auflösung des Videos bezieht. Bei einer 16:9-PAL-DVD mit 720×576 Pixeln ergibt sich ein DAR von

720:576 = 1,25:1.

► *Wiedergabe-DAR,* das sich auf das korrekte, entzerrte Wiedergabe-AR der gesamten Auflösung bezieht. Für die DVD von oben ergibt sich 1047:576 = 1,82:1.

► *Aktives Wiedergabe-DAR,* das sich auf die entzerrte Wiedergabeauflösung des eigentlich aktiven Bildes bezieht, d. h. minus aller schwarzer Balken. Nehmen wir an, unsere Beispiel-DVD hätte oben und unten je 72 Pixel Balken. Daraus ergibt sich ein DAR von 1047:432 = 2,42:1.

► *Aktives gespeichertes DAR,* das sich auf die gespeicherte Auflösung minus schwarzer Balken bezieht, ohne eine Verzerrung zu berücksichtigen. Das ergibt im Beispiel ein DAR von 720:432 = 1,67:1.

Diese Vielfalt ist keineswegs akademische Haarspalterei. Zumindest die ersten drei DAR-Varianten werden tatsächlich praktisch verwendet, wobei das aktive Wiedergabe-DAR überwiegen dürfte.

Haben wir uns darauf geeignet, welche DAR-Variante wir verwenden wollen, schlägt das nächste Problem zu. Das DAR ändert sich, 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. Tab. A-8, Seite 47), und zwar 16:9 und 4:3 jeweils für PAL und NTSC. Da uns die Encodingtools alle nötigen Informationen liefern, brauchen wir an der passenden Stelle nur den richtigen Tabellenwert einsetzen. Das Cropping allein ändert nichts an der Form der einzelnen Pixel.

Ein klassisches, nicht anamorph encodiertes DVD-Backup hat immer ein PAR von 1:1 und ein DAR von »Bildbreite zu Bildhöhe«. Hier fallen alle DAR-Definitionen zusammen. Das PAR verändert sich im Vergleich zur DVD, weil beim Resizing die Verzerrung korrigiert wird.

Sehen wir uns das an der kleinen Grafik links in Abb. A-11 an, die ein vereinfachtes Videobild aus 5 × 4 Pixeln darstellen soll, wie es korrekt entzerrt beim Abspielen aussieht. Das Bild ist nicht quadratisch, da seine Pixel ähnlich der DVD eine rechteckige Form haben, und zwar im Verhältnis Breite:Höhe 2:1. Die Form des gesamten Bilds und die Form eines einzelnen Pixels lässt sich leicht berechnen:

$$\text{DAR} = {}^{x}\!/_{y} = {}^{10}\!/_{4} = 2{,}5;\ \text{PAR} = {}^{a}\!/_{b} = {}^{2}\!/_{1} = 2.$$

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 (Abb. A-11 rechts). Es gilt nun:

$$DAR = {}^{x}\!/_{y} = {}^{10}\!/_{2} = 5;\ PAR = {}^{a}\!/_{b} = {}^{2}\!/_{1} = 2.$$

Durch das Cropping nimmt natürlich die Höhe des Bildes ab, *y* wird kleiner. Dadurch verändert sich auch der Wert des DAR, im Beispiel von 2,5 auf 5. 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.

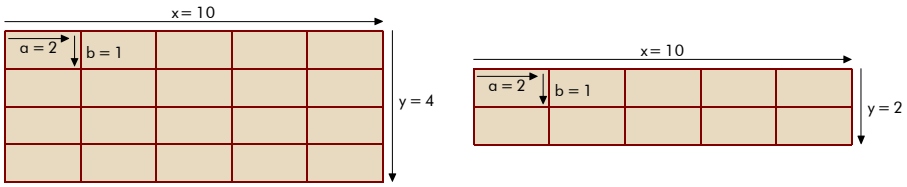


Abb. A-11: PAR und DAR vor (links) und nach (rechts) dem Cropping.

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 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 schon länger, DivX seit Version 6.5. 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-Mu-

xing, 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 die Abspielsoftware keine Unterstützung mitbringt, um mindestens eines davon zu lesen und anzuwenden. Für DirectShow-Player (Media Player Classic, Zoom Player usw.) haben wir mit folgender Kombination aus Splitter und Decoder die wenigsten Probleme:

► 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.

Zu einem ausführlichen Test verschiedener Splitter/Decoder-Kombinationen habe ich mich inzwischen auch durchgerungen und im Artikel ►*AR-Flag-Unterstützung in DirectShow* dokumentiert. ► [BrJo08a]

Player, die nicht auf DirectShow aufsetzen (Paradebeispiel: VLC), müssen die Unterstützung für AR-Flags direkt eingebaut haben.

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. 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 57) 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 sieht aus wie links in Abb. A-12. 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, erhalten wir eine korrekte Zielauflösung von 608 × 256 Pixeln wie rechts in Abb. A-12.

Abb. A-12: Falsches (links) und richtiges (rechts) Resizing.

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 218 im Kopf behalten. Die Zielauflösung soll 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:

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

► **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 beschnittener Breite zu beschnittener 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 134 f.) die nötigen Informationen. Das richtige PAR brauchen wir dann nur noch aus der Tab. A-8 auf Seite 47 ablesen: $^{16}/_{11}$.

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

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

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 soll. 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 nicht der Mod16-Restriktion.
► **Cropping:** Jedes weggeschnittene 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 (neben einem geringen AR-Fehler) gute Qualität bietet. Für ein 1-Pass-Encoding ist das einfach. Die günstige Auflösung ist diejenige, die das gewünschte Niveau an Details erhält. Beim 2-Pass-Encoding müssen wir Auflösung, Zielgröße und Qualität gegeneinander abwägen. Dazu benötigen wir einen Qualitätsindikator.

Qualitätsindikatoren

Im Wesentlichen stehen uns drei Indikatoren zur Verfügung. Welche das sind und welche Aussagekraft sie besitzen, das zeigt die folgende Tab. A-6.

Indikator	Aussagekraft
Bitrate	nutzlos
relative Bitrate (BPF)	erster Grobeindruck
Kompressionstest	guter individueller Anhaltspunkt

Tab. A-6: Aussagekraft verschiedener Qualitätsindikatoren.

Obwohl man sofort an die **Bitrate** (meistens in kbit/s angegeben) denkt, wenn man einen digitalen Film sieht, interessiert uns die während des gesamten Backupprozesses überhaupt nicht. Die Angabe, wie viele Bits dem Encoder im Durchschnitt pro Sekunde zur Verfügung stehen, ist viel zu ungenau, um eine brauchbare Aussage über die Qualität zu liefern.

Interessanter ist die **relative Bitrate**, besser bekannt als **BPF-Wert**. Der gibt an, wie viele Bits zum Encodieren eines einzelnen Pixels in jedem Bild durchschnittlich zur Verfügung stehen. Das heißt, er berücksichtigt alle direkt berechenbaren Einflussfaktoren (Zielgröße, Filmlänge, Framerate, Auflösung). Allerdings sagt er kaum 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. Um einen ersten groben Eindruck zu bekommen, ist der BPF-Wert ganz nützlich. Deshalb können wir uns gerne daran orientieren, wenn ihn das Encoding-Frontend sowieso anzeigt. Wenn nicht, auch nicht so wild.

Die individuellen Eigenschaften jedes Films berücksichtigt nur der **Kompressionstest.** Der Test nimmt in regelmäßigen Abständen kurze Schnipsel aus dem Film und encodiert die mit maximaler Qualität. Das Ergebnis ist ein guter Anhaltspunkt für die tatsächlich sichtbare Qualität des Encodings.

Wahl einer sinnvollen Auflösung

Mit der relativen Bitrate als Indikator ausgerüstet, können wir nun daran gehen, eine sinnvolle Auflösung zu wählen. Dabei geht es erst einmal darum, grob die Parameter des Encodings einzustellen, so dass sie in einem sinnvollen Bereich liegen. Hinterher führen wir sowieso noch einen Kompressionstest durch, der uns genauer darüber Aufschluss gibt, wie gut die erste Wahl war. Wenn sich dabei herausstellt, dass wir daneben gelegen waren, passen wir eben noch einmal an.

Je kleiner die Auflösung, desto weniger Details enthält ein Einzelbild unabhängig 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 einer weniger heftigen Kompression und vermeidet dadurch möglicherweise Makroblock-Artefakte; das erkaufen wir jedoch mit einem Detail- und damit Qualitätsverlust. Um den nicht zu groß werden zu lassen, sollten wir Auflösungen deutlich unter 600 Pixel in der Horizontalen besser vermeiden.

Ein Maximum ist einfacher definiert. Spätestens, wenn sowohl Höhe als auch Breite die zugeschnittene 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 mehr als die originalen 720 horizontalen Pixeln kaum vorkommen.

Und damit zu ein paar konkreten Anhaltspunkten für die Auflösungswahl. Haben wir den BPF-Wert zur Verfügung, sollten wir ihn verwenden. Ansonsten können wir uns an der Fläche der Zielauflösung orientieren, so wie in Tab. A-7 zu sehen. Ungewöhnlich lange/kurze Filmlängen oder ungewöhnlich große/kleine Audiospuren sollten wir dabei natürlich zusätzlich berücksichtigen.

Zielgröße	BPF-Wert	Auflösung (Fläche)
1 CD	Xvid/DivX: nicht unter 0,20 x264: nicht unter 0,15	ca. 160 000 Pixel
2 CDs	darf sich ruhig Richtung 0,30 orientieren	ca. 230 000 Pixel, aber nicht wesentlich mehr als 720 Pixel horizontal
½ DVD	darf die 0,30 ruhig überschreiten	irrelevant, da sich anamorphes Encoding ohne Resizing anbietet

Tab. A-7: Daumenregeln für die Auflösungswahl

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.

Ohne BPF-Wert ist es nur mit Erfahrung möglich, die Auswirkungen dieser Anpassungen abzuschätzen. Die Tabelle oben geht für 1-CD-Encodings von einer einzelnen kleinen Audiospur aus und für 2-CD-Encodings von einer großen (möglicherweise AC-3) bzw. zwei kleineren. Die richtig hohen Zielgrößen sollten nahezu immer den Platz für anamorphes Bild und mindestens eine AC-3 bieten.

Das wichtigste an der Auflösungswahl ist, die Grenzen als die fließenden Richtlinien zu sehen, die sie sind. Mit wachsender Erfahrung weiß man irgendwann intuitiv, welcher Film ein paar mehr Bits nötig hat und welchen man ein bisschen mehr quetschen kann. Außerdem ist sowieso alles vorläufig. Nach dem Kompressionstest haben wir einen Indikator, der uns deutlich mehr als einen groben Anhaltspunkt bietet. Da der Test sich zwischen den Encoding-Frontends ein Stück unterscheidet, besprechen wir den später im Praxisteil zu StaxRip und Gordian Knot.

A.4 Videocodec-Konfiguration

Thema dieses Abschnitts sind die Videocodecs. Zuerst sehen wir uns an, welche Encodingstrategie für uns passt, denn das allgegenwärtige 2-Pass-Verfahren ist keineswegs ein Allheilmittel. Anschließend gehen wir im Detail auf die einzelnen Codecs, ihre Fähigkeiten und Besonderheiten und ihre Konfiguration ein. Da die Codec-Konfiguration ganz zentral für die Videoqualität verantwortlich ist, lohnt es sich, hier etwas Zeit zu investieren.

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 1-Pass vs. 2-Pass: Encodingmethoden

Lange Zeit war es vollkommen selbstverständlich, dass wir den Film im 2-Pass-Verfahren für eine vorher festgelegte Zielgröße encodieren. Das ist jedoch weder die einzige Encodingmethode noch in jedem Fall sinnvoll. In diesem Kapitel lernen wir, für jede Situation die passende Methode zu wählen.

1-Pass, 2-Pass und N-Pass

Grundsätzlich haben wir zwei Möglichkeiten, ein Video zu encodieren. Entweder erstellen wir aus dem Quellmaterial direkt das Zielvideo oder wir analysieren zunächst die Quelle und encodieren erst dann mit Hilfe der gewonnenen Daten. Beide Methoden sind gleichwertig, d. h. wir können nicht sagen, dass eine davon besser wäre. Jede hat ihre angestammten Einsatzgebiete.

Durchläuft der Encoder das Quellmaterial nur ein einziges Mal und erzeugt direkt die Zieldatei, haben wir es mit dem **1-Pass-Verfahren** zu tun. Das ist in Abb. dargestellt.

Das Gegenstück zu 1-Pass ist das **2-Pass-Encoding** mit zwei Durchläufen (Abb. A-14). Im ersten Durchgang (1st Pass) sammelt der Encoder Daten über die Komplexität und andere Eigenschaften des Videos und trifft einige grundlegende Entscheidungen über die Encoding-Strategie. An welchen Stellen I-, P- und B-Fra-

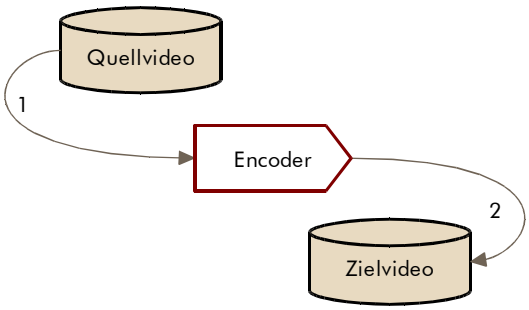


Abb. A-13: Schema des 1-Pass-Encodings.

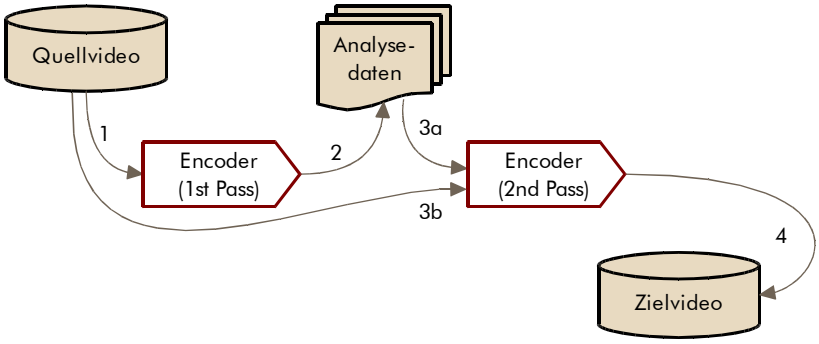


Abb. A-14: Schema des 2-Pass-Encodings.

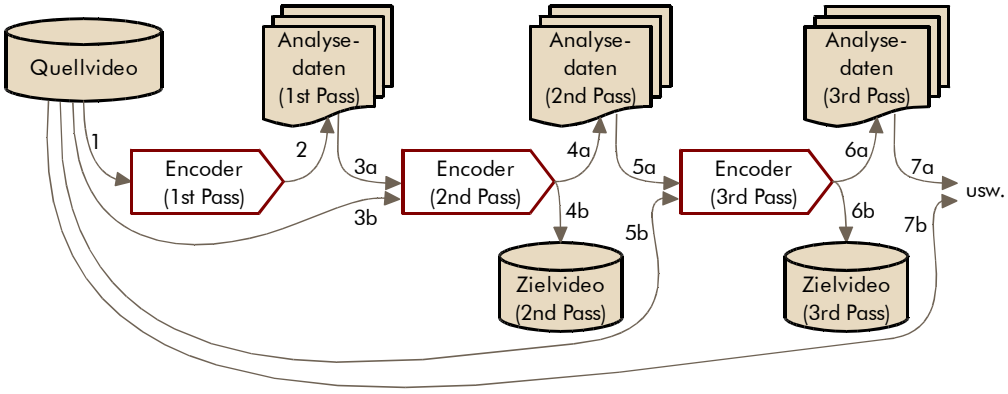


Abb. A-15: Schema des N-Pass-Encodings.

mes gesetzt werden, wird z. B. hier entschieden. Das Ergebnis des 1st Pass ist eine kleine Datei mit mehr oder weniger menschenlesbaren Analysedaten. Ein Video entsteht erst im zweiten Durchgang (2nd Pass) aus dem Quellmaterial und den gesammelten Daten.

Dieses Verfahren lässt sich ausbauen, indem wir weitere Durchgänge anhängen, was als **N-Pass-Verfahren** (oder **Multipass**) bezeichnet wird. An Abb. A-15 werden auch die Gemeinsamkeiten mit 2-Pass deutlich. Im Gegensatz zum 2-Pass wird jetzt im zweiten Durchgang nicht nur das Zielvideo erzeugt, sondern zusätzlich wieder Analysedaten gesammelt. Diese können verwendet werden, um in einem 3rd Pass ein noch einmal verfeinertes Video zu erstellen. Da die 2nd-Pass-Analyse genauere Ergebnisse liefert als die 1st-Pass-Analyse, hat die 3rd-Pass-Zieldatei eine höhere Qualität.

Dieses Spiel können wir prinzipiell beliebig oft fortsetzen. Allerdings ist schon beim 3-Pass der Qualitätsgewinn meist so gering, dass sich der gewaltige Zeitaufwand für den zusätzlichen Durchgang kaum lohnt. Deswegen spielt das N-Pass-Encoding in der Praxis nur eine untergeordnete Rolle.

Encodingmethoden und ihre Anwendung

Sehen wir uns die Encodingmethoden zuerst in der Übersicht von Tab. A-8 an. 2-Pass/N-Pass ist ein recht klar definiertes Verfahren. 1-Pass dagegen bietet uns mehrere Möglichkeiten.

Tab. A-8: Encoding-Methoden mit Einsatzgebieten.

Methode	Einsatzgebiet
2-Pass/N-Pass	Erreichen einer vorher festgelegten, exakten Dateigröße.
1-Pass, Bitrate	Erreichen einer durchschnittlichen Datenrate mit geringen Schwankungen.
1-Pass, Qualität	Erreichen eines vorher definierten visuellen Qualitätsniveaus.
1-Pass, Quantizer	Erreichen eines vorher definierten Qualitätsniveaus. Ist eine einfacher gestrickte Variante der Qualitätsmethode.

2-Pass und N-Pass

Sinn und Zweck eines 2-Pass-Encodings ist es, eine vorher festgelegte Dateigröße exakt zu treffen und dabei die subjektiv empfundene Videoqualität im Verlauf des Films möglichst konstant zu halten. 2-Pass war nie und ist nicht die Methode, um dieses wolkige Ideal von »maximaler Qualität« zu erfüllen. Ganz im Gegenteil spielt das Niveau der Qualität nur indirekt über die Zielgröße eine Rolle.

Das Anwendungsgebiet von 2-Pass liegt deswegen überall dort, wo wir vor dem Encoding schon die endgültige Größe wissen müssen. Das klassische Beispiel dafür ist – ganz klar – die Abstimmung auf eine feste Zahl von CDs. Umgekehrt heißt das auch, sobald es nicht mehr entscheidend ist, vorher die endgültige Größe exakt zu bestimmen, ist 2-Pass die falsche Wahl.

1-Pass, Bitrate (ABR)

Diese Methode versucht, in nur einem Durchgang eine festgelegte durchschnittliche Bitrate zu erreichen. Fürs DVD-Backup ist das nicht besonders relevant, da es sich um die Methode handelt, bei der Qualität am wenigsten eine Rolle spielt. Das Hauptkriterium ist eher, die Schwankung der Datenrate klein zu halten, was z. B. für Streaming-Anwendungen wichtig sein kann. Als Nebenwirkung der vordefinierten Bitrate lässt sich auch die endgültige Größe halbwegs vorher abschätzen, wenn auch eine Punktlandung wie beim 2-Pass-Encoding nicht möglich ist. Wir werden uns um 1-Pass ABR nicht weiter kümmern.

1-Pass, konstante Qualität

Konstante Qualität ist das Gegenstück zur konstanten Dateigröße des 2-Pass-Verfahrens. Wir definieren vor dem Encoding ein Qualitätsniveau, das vom Encoder eingehalten wird, ohne auf die Größe zu achten. Deswegen wissen wir vorher nicht, wie groß der Film werden wird. Mit Qualität ist hier die visuelle Qualität gemeint, die wir beim Anschauen wahrnehmen. Für den Encoder heißt das, dass er je nach Eigenschaften der Szene unterschiedlich stark komprimieren darf.

Je stärker geräumige Festplatten die CD und DVD als Filmspeicher ablösen, desto wichtiger wird das Qualitätsverfahren. Denn ohne die begrenzte Kapazität des optischen Speichers fällt auch der Zwang weg, eine exakte Zielgröße zu erreichen. Leider besitzen weder Xvid noch DivX einen solchen Encodingmodus. Bei beiden Codecs müssen wir das Quantizer-Verfahren als Annäherung verwenden. x264 hat diese Einschränkung nicht. Die Qualitätsmethode heißt dort CRF (Constant Rate Factor).

1-Pass, konstanter Quantizer (CQ)

CQ ist eine simplere Variante des Qualitätsverfahrens, verfolgt aber letztendlich das gleiche Ziel einer konstanten Qualität. Allerdings wird die Kompression nicht je nach Komplexität angepasst, sondern ist für jedes Frame genau gleich. Dafür sorgt der von uns fest definierte ▸Quantizer – eine Art Kompressionsfaktor an zentraler Stelle im Encodingprozess.

▸A.1.2.1

Das Ergebnis ist eine ganz brauchbare Annäherung an echte konstante visuelle Qualität. Da CQ hauptsächlich für hochqualitative Encodings mit niedrigen Quantizern verwendet wird, ist der Nachteil gegenüber der Qualitätsmethode nicht unbedingt schwankende Qualität, sondern eher eine etwas größere Zieldatei.

A.4.2 Interfacetypen: VfW und Kommandozeile

Im Wesentlichen existieren heute zwei Möglichkeiten, um einen Encoder zu konfigurieren: das grafische VfW und die textbasierte Kommandozeile (CLI). Welches Interface wir verwenden, hängt hauptsächlich vom Encoding-Frontend ab. Das stellt normalerweise auch eine grafische Oberfläche für CLI-Encoder bereit. Niemand ist also gezwungen, kryptische Befehlszeilen zu tippen. Die ausführlichen Erklärungen, die in den nächsten Kapiteln zu den Parametern der CLI-Encoder folgen, sind trotzdem fürs Verständnis nützlich, welche Option sich wie aufs Encoding aufwirkt.

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, z. B. 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 Techno-

logien zu setzen, die nicht den Einschränkungen eines eineinhalb Jahrzehnte 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 XvidEncraw ein Encoder für die Kommandozeile. Nur DivX ist weiterhin ausschließlich für VfW verfügbar.

Das Kommandozeilen-Interface (CLI)

Die Kommandozeile wird oft auch als Eingabeaufforderung oder Konsole bezeichnet. Das Prinzip besteht darin, Befehle in Textform einzutippen und auszuführen. Es ist das Gegenmodell zur grafischen Mausoberfläche. Die grundsätzliche Bedienung der Kommandozeile zu erklären, gehört nicht ins Encodingwissen, deshalb setze ich das voraus.

Die Abkürzung CLI steht für Commandline Interface. Das ist der englische Ausdruck für Kommandozeilen-Schnittstelle.

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. In den folgenden CLI-Kapiteln geht es immer darum, die Befehle von Hand zusammenzustellen.

Als Übersicht und zur allgemeinen Darstellung werden Kommandozeilen meist in Syntaxschreibweise angegeben. Dadurch sehen wir auf einen Blick, welche Optionen wir in welcher Kombination verwenden dürfen. Das könnte beispielsweise so aussehen:

```
foo.exe -in "<Quelle>" -out "<Ziel>" [-blubb] {-foo|-bar} [-klick|-klack]
```

Die Syntaxschreibweise arbeitet mit diesen Grundregeln:

► Alle Optionen, die nicht innerhalb einer eckigen oder geschweiften Klammer stehen, sind Pflichtangaben. Es wäre nicht erlaubt, `-in` oder `-out` wegzulassen.
► Alle Klammern (spitze, eckige und geschweifte) sowie der senkrechte Strich sind erklärende Zeichen, die in der echten Kommandozeile niemals auftauchen.
► Die Optionen werden jeweils durch ein Leerzeichen getrennt. Genauso steht zwischen einem Parameter und dessen zugehörigem Wert ein Leerzeichen.
► Spitze Klammern (< und >) enthalten die Beschreibung eines Parameterwerts. Die Beschreibung muss einschließlich der Klammern durch den tatsächlichen Wert ersetzt werden. In unserem Beispiel würden wir `<Quelle>` durch etwas wie `D:\Video\Quelle.avs` ersetzen und hätten als Ergebnis `-in "D:\Video\Quelle.avs"`.
► Eckige Klammern ([und]) bezeichnen optionale Parameter. Wir können ganz nach Wunsch `-blubb` angeben oder weglassen.
► Geschweifte Klammern ({ und }) bezeichnen eine zwingende Entweder-oder-Auswahl. Die wählbaren Optionen sind durch einen senkrechten Strich (|) getrennt. Im Beispiel müssen wir zwingend entweder `-foo` oder `-bar` angeben. Beide Optionen gleichzeitig sind verboten, genauso wie beide wegzulassen. Es könnten auch mehr als zwei Optionen in der Klammer stehen, was nichts ändert. Wir entscheiden uns trotzdem für genau eine davon.
► Die Entweder-oder-Auswahl kann es auch in eckigen Klammern geben. Entsprechend dem optionalen Charakter der eckigen Klammer heißt das dann, wir können entweder `-klick` oder `-klack` angeben oder beide weglassen. Beide gemeinsam anzugeben ist nicht erlaubt.

Prinzipiell können alle diese Elemente beliebig verschachtelt und aneinander gehängt werden. Die Konstruktionen, mit denen wir in den folgenden Kapiteln in Berührung kommen, bleiben aber recht übersichtlich.

Weitere Literatur: [Fied06] [Rich03]

A.4.3 Die Xvid-Konfiguration

Der Abschnitt zur Xvid-Konfiguration kann nicht jede Option bis ins letzte Detail erklären, denn dann müsste ich ein komplettes Buch ausschließlich über Xvid schreiben. Die meisten Einstellungen sind auch ohne tiefe technische Hintergründe leicht verständlich. Einige verdienen aber doch eine etwas eingehendere Betrachtung, und dafür ist dieses Kapitel zuständig. Die Erklärungen sind encodingwissen-üblich umfangreich ausgefallen. Es ist nicht nötig, das alles bis ins

letzte Detail zu verstehen, um Xvid sinnvoll zu konfigurieren. Die empfohlenen Werte – die in allen folgenden Kapiteln auch immer ausdrücklich erwähnt werden – eignen sich problemlos für die meisten Situationen.

Nicht alle Xvid-Optionen sind gleich wichtig für die Qualität und Geschwindigkeit des Encodings. Im wesentlichen haben wir es mit drei Haupt-Einflussfaktoren zu tun: B-Frames, VHQ und Quantisierungsmatrizen; dazu kommt beim 1-Pass-Encoding der Quantizer. Die meisten anderen Xvid-Optionen haben für sich allein betrachtet nur recht geringe Auswirkungen.

Der Quantizer

Vereinfacht ausgedrückt ist der Quantizer ein Faktor, der die Stärke der Kompression regelt. Hochoffiziell heißt er auch *quantiser scale parameter,* abgekürzt QP. Er darf je nach Konfiguration für jedes Frame oder jeden Makroblock verschieden sein und liegt bei MPEG-4 Visual zwischen 1 und 31, wobei nur ganze Zahlen erlaubt sind. Je kleiner der Wert, desto sanfter die Kompression und desto besser die Qualität. Interessant ist besonders der Bereich zwischen QP 2 und 8, denn ein Encoding mit auch nur halbwegs sinnvoller Qualität wird kaum jemals höhere Werte enthalten.

Beim 2-Pass-Encoding haben wir mit dem Quantizer nur indirekt zu tun, da Xvid die Verteilung selbst in die Hand nimmt, um die Zielgröße zu treffen. Beim 1-Pass-Encoding dagegen setzen wir für den Quantizer einen festen Wert, der zunächst für I- und P-Frames gilt. B-Frames berechnen daraus ihren eigenen QP, der normalerweise höher liegt. Genauer beschäftigen wir uns weiter unten damit.

Bleiben wir beim 1-Pass-Encoding. Welcher Quantizer sich wie auf Qualität und Dateigröße auswirkt, hängt von der Encoder-Konfiguration insgesamt ab. Ein paar Anhaltspunkte:

► **Quant 1** erzeugt exzellente Qualität und gewaltige Dateigrößen. Jedoch ist sein Preis/Leistungs-Verhältnis so schlecht, dass wir ihn nie verwenden sollten. Sowohl als fester Wert im 1-Pass-Encoding als auch als Minimum im 2-Pass-Encoding ist er fehl am Platz.

► **Quant 2** ist die Einstellung der Wahl für HQ-Encodings, die die Qualität der DVD-Quelle so vollständig wie möglich erhalten sollen. Die Videos werden deutlich kleiner als mit Quant 1, die Qualität ist jedoch praktisch identisch. Die endgültige Dateigröße, die wir für eine durchschnittlichen Spielfilm bei voller Auflösung und zwei Tonspuren erwarten können, liegt grob im Bereich zwischen 2,5 und 3,5 GB. Wenn im Zusammenhang mit 1-Pass-Xvid von »maximaler Qualität«

die Rede ist, ist i. d. R. Quant 2 gemeint.

► **Quant 3** ist sinnvoll, wenn wir Wert auf kompaktere Dateien legen und dafür auch eine Ecke Qualität zu opfern bereit sind. Der Verlust ist durchaus noch akzeptabel. Qualitativ dürfte das Ergebnis grob im Bereich eines 2-CD-Encodings liegen. Manche Quantisierungsmatrizen sind auf Minimum-Quant 3 ausgelegt (weiter unten mehr dazu). In dem Fall verhält sich Quant 3 wie Quant 2.

► Ab **Quant 4** aufwärts dürfte das Ergebnis regelmäßig zu schlecht werden. Das widerspricht nicht der Aussage von oben, dass der Bereich zwischen QP 2 und 8 realistisch wäre. Es ist ein großer Unterschied, ob im 2-Pass-Encoding ab und zu ein sehr hoch mit Quant 8 quantisiertes Frame auftaucht oder ob das der Basiswert im 1-Pass-Encoding ist, der für einen Großteil aller Frames gilt.

Konfiguration von B-Frames

Das Konzept der B-Frames haben wir im Kapitel über die Interframe-Kompression auf Seite 12 schon angesprochen. Hier beleuchten wir die bidirektionalen Bilder aus der Sicht von Xvid, denn der Encoder bietet uns gleich vier Stellschrauben zur Konfiguration.

B-Frame-Verteilung

Erst einmal wäre da die leicht verständliche Einstellung, ob überhaupt B-Frames verwendet werden sollen, und wenn ja, wie viele maximal direkt hintereinander stehen dürfen. Mit einem Maximum von zwei wäre eine solche Bildsequenz aus I-Frames, P-Frames und B-Frames möglich: IPBBP, bei maximal drei B-Frames wäre es dann IPBBBP usw. Wichtig: Die Einstellung definiert nur das Maximum. Es bedeutet *nicht*, dass *immer* genau so viele B-Frames hintereinander stehen. Lediglich können *niemals mehr* als das Maximum aufeinander folgen. Innerhalb dieser Grenze berechnet Xvid selbst die günstigste Möglichkeit.

Diese automatische Auswahl lässt sich beeinflussen. Wir haben die Möglichkeit, Xvid zum verstärkten B-Frame-Einsatz zu drängen oder ihn eher von B-Frames abzuhalten. Mit diesen beiden ersten Schrauben (Maximum und Empfindlichkeit) lässt sich die B-Frame-Verteilung recht fein auf die jeweilige Encodingsituation abstimmen.

In der Standardeinstellung erlaubt Xvid zwei B-Frames hintereinander. Das ist für die allermeisten Situationen auch ein guter Wert. An eine Anpassung brauchen wir nur in extremen Situationen denken, bei 1-CD-Encodings zum Beispiel. Hier

würde ich generell drei B-Frames empfehlen, evtl. sogar vier. Das andere Extrem sind hochqualitative Encodings nahe der Sättigungsgrenze. So etwas begegnet uns bei kurzen und/oder wenig komplexen Filmen auf ½ DVD oder bei Encodings für einen ganzen DVD-Rohling. Wenn derart massig Platz verfügbar ist, reicht meistens auch ein einzelnes B-Frame als Maximum aus. Sie ganz zu deaktivieren, ist in der Regel nicht sinnvoll; denn B-Frames sind das Tool, das die Dateigröße am weitesten in den Keller ziehen kann ohne die Qualität spürbar zu beeinflussen. Entsprechend groß fällt der Nachteil aus, wenn wir sie abschalten.

B-Frame-Quantizer

Die verbleibenden beiden Stellschrauben drehen nicht an der Verteilung der B-Frames im Encoding, sondern an deren Kompression, genauer gesagt am Quantizer. Sehen wir uns dazu die Bildfolge IPBBBP an. Welchen Quantizer das mittlere B-Frame bekommt, hängt vom vorausgehenden und nachfolgenden P-Frame ab. Die drei B-Frames beeinflussen sich nicht gegenseitig, woraus natürlich folgt, dass alle drei B-Frames den selben Quantizer erhalten, weil die Berechnung identisch ist. Und die geht so:

Aus den Quantizern der beiden P-Frames wird der Mittelwert gebildet und dieser mit einem von uns festgelegten Faktor multipliziert (Quantizer Ratio). Zusätzlich dürfen wir anschließend noch einen beliebigen Wert addieren (Quantizer Offset). In Formelschreibweise sieht das folgendermaßen aus:

$$BQuant_f = \frac{Quant_{f-1} + Quant_{f+1}}{2} \times BRatio + BOffset$$

Weiter oben haben wir schon erfahren, dass Quantizer ganze Zahlen zwischen 1 und 31 sind. Innerhalb dieser Grenzen rundet Xvid bei der Berechnung immer ab, also auf den nächstkleineren Quantizer.

$BQuant_f$ bezeichnet den zu ermittelnden Quantizer des aktuellen B-Frames. *Quant* steht für die Quantizer der umgebenden P-Frames. Deshalb dürfen wir die Indizes $f–1$ und $f+1$ auch nicht mit mathematischer Präzision betrachten. Sie stehen eben nicht unbedingt für das direkte Vorgänger- und Nachfolger-Bild. *BRatio* ist unsere erste Stellschraube (Quantizer Ratio) und *BOffset* ist die zweite Stellschraube (Quantizer Offset).

Vorgänger und/oder Nachfolger können auch I-Frames sein. Lautete unsere Beispielsequenz von oben IBBBP, würde das nichts ändern. Das relevante Vorgänger-Frame wäre eben anstatt einem P- ein I-Frame.

Nehmen wir einmal an, das Vorgänger-P-Frame hätte einen Quantizer von 2 erhalten und das Nachfolger-P-Frame einen Quantizer von 3. Außerdem hätten wir die Stellschrauben auf Xvids Standardeinstellungen belassen (Ratio = 1,50 und Offset = 1,00). Dann würde sich für unsere B-Frames ein rechnerischer Wert von 4,75 und abgerundet ein Quantizer von 4 ergeben.

$$BQuant_f = \frac{2 + 3}{2} \times 1{,}50 + 1{,}00 = 4{,}75 = \text{gerundet } 4$$

Mit diesen beiden Konfigurationsmöglichkeiten lässt sich die Stärke der B-Frame-Kompression in nahezu beliebiger Weise beeinflussen. Das heißt natürlich auch, dass wir ein wenig Vorsicht walten lassen sollten, denn unpassende Einstellungen wirken sich schnell spürbar auf die Qualität aus.

Ich bevorzuge abweichend von den Xvid-Standards Didées Lieblingswerte in ihrer aktuellen, leicht modifizierten Form: Ratio 1,62 und Offset 0. Wer keine Lust auf die Details hat, kann problemlos darauf Vertrauen, dass das schon Hand und Fuß hat, was Didée austüftelt – oder auch bei den Xvid-Vorgaben bleiben. Beide Varianten werden keine Probleme verursachen.

Natürlich gibt es einen Grund, von den Vorgaben abzuweichen. Didées Variante führt wie die Xvid-Standardwerte bei hohen Quantizern (also hoher Kompression) der umgebenden P-Frames zu einer relativ noch stärkeren Kompression der B-Frames, und das ist in dieser Situation auch wünschenswert. Anders sieht das aus, wenn die P-Quantizer niedrig sind (2 oder 3). Dann steht in der Regel reichlich Bitrate zur Verfügung, weshalb man gern auch die B-Frames etwas großzügiger behandeln darf. Denn auch für diese gilt die Regel, je weniger Kompression desto höher die Qualität. Didées Werte berücksichtigen das mit folgendem Verhalten:

- hoher P-Quantizer: hoher Unterschied zum B-Quantizer,
- niedriger P-Quantizer: geringerer Unterschied zum B-Quantizer.

Die üblichen Xvid-Standardwerte (1,5 und 1,0) treffen diese Unterscheidung nicht so stark. Sie arbeiten immer mit einem recht hohen Unterschied zwischen P- und B-Quantizer. Sehen wir uns zum Vergleich der beiden Varianten Abb. A-16 an. Hier wird deutlich, wie Didée bei geringen P-Quantizern auch die B-Frames großzügig behandelt. Je höher der P-Quantizer ausfällt, desto mehr ähneln sich beide Varianten.

Wer das jetzt alles nachvollzogen und verstanden hat, ist auch gut für eigene Experimente mit Ratio und Offset gerüstet. Ansonsten haben wir zwei Werte-

paare, die beide gute Ergebnisse liefern. Bei 1-CD-Encodings könnten die Xvid-Vorgaben einen Vorteil haben, da in dieser Situation jedes gesparte Bit wertvoll ist, auch bei stellenweise niedrigen P-Quantizern. Ansonsten würde ich eher zu Didées Variante raten.

Um beliebige solche Vergleichsdiagramme zu erstellen, habe ich mir per Tabellenkalkulation etwas gebastelt. Vielleicht findet es sonst noch jemand nützlich. Download unter `http://brother-john.net/files/xvid_bframe_quants.zip`

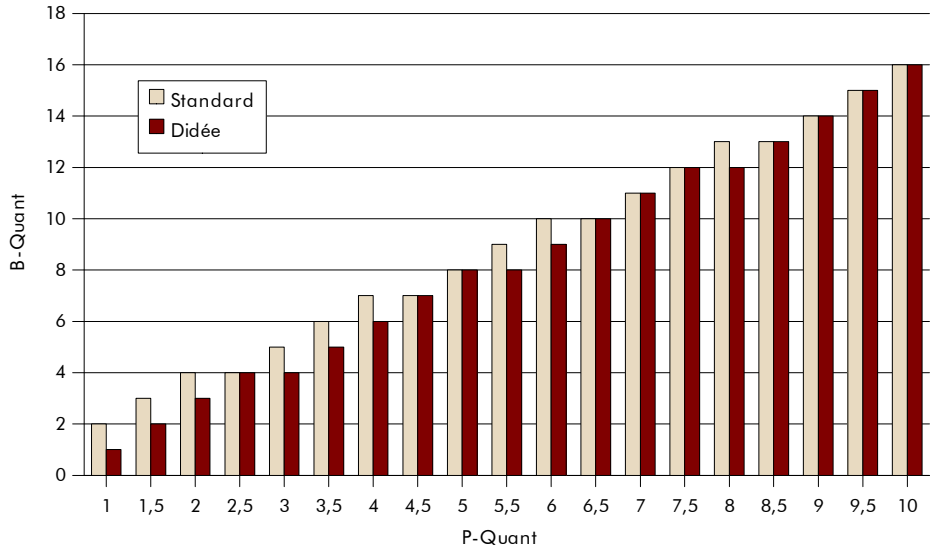


Abb. A-16: Vergleich zwischen Didées und Standard-B-Frame-Quantisierung.

Quantisierungsmatrizen

Ein weiteres wichtiges und oft unterschätztes Werkzeug sind die Quantisierungsmatrizen. Welche Bedeutung denen mathematisch genau zufällt, ist auch fürs Encodingwissen zu viel der Hintergrundinformation. Ansatzweise haben uns damit schon auf Seite 8 beschäftigt. Für die Praxis sollten wir uns vielmehr ihre Funktion merken. Und zwar steuert die Quantisierungsmatrix den Detailreichtum des Bildes. Je weniger Details, desto höher lässt sich das Bild komprimieren, ohne Makroblock-Artefakte zu erzeugen. Natürlich ist ein weniger detailreiches Bild qualitativ schlechter als eines mit mehr Details. Deswegen gilt es abzuwägen zwischen der Erhaltung der feinen Bildstrukturen und der Gefahr der Blockbildung.

MPEG-4 ASP und damit Xvid kennt zwei grundsätzliche Arbeitsmodi. Der eine nennt sich H.263 und arbeitet genau genommen nicht mit einer Matrix. Trotzdem hat sich der Begriff der H.263-Matrix eingebürgert. Der zweite Modus baut direkt

auf dem Matrixsystem auf, wobei eine davon gleich im MPEG-4-Standard selbst definiert ist. Das ist das bekannte Gegenstück zu H.263: die MPEG-4-Matrix. Darüber hinaus besteht die Möglichkeit, eigene Matrizen zu definieren und diese anstelle der MPEG-4-Matrix zu verwenden. Das sind die benutzerdefinierten Matrizen, oft mit CQM für *Custom Quantizer Matrix* abgekürzt. Zusammengefasst haben wir also die Wahl aus drei Möglichkeiten:

- H.263,
- MPEG mit Standardmatrix,
- MPEG mit benutzerdefinierter Matrix.

H.263 ist eine recht »weiche« Matrix, d. h. sie entfernt viele Bilddetails und eignet sich damit eher für den 1-CD-Bereich. Die MPEG-4-Standardmatrix erhält deutlich mehr Details und bietet sich damit für höhere Zielgrößen an. Wenn wir uns nicht mit Custom-Matrizen herumschlagen wollen, reicht diese Regel schon aus. Custom-Matrizen existieren für alle Anwendungsfälle von der Spezialmatrix fürs 1-CD-Encoding über gute Alltagsmatrizen für den ½-DVD-Bereich bis hin zu Monstern, die in der fast-verlustlosen Liga spielen und Dateigrößen weit jenseits des Originals erzeugen. Die folgende Tab. A-9 bietet eine Auswahl gerne genutzter Matrizen (siehe ▸SpikeSpiegels Matrizen-Test auf Doom9.org).

▸ [Spik05]

Tab. A-9: Eine Auswahl beliebter CQMs. (Downloadlinks siehe Literaturverzeichnis.)

Matrix	Autor	Beschreibung
Jawor's 1CD	Jawor	Kompression vergleichbar mit der H.263. Die Matrix ist speziell für den Einsatz im 1-CD-Encoding ausgelegt.
EQM V3LR	Sharktooth	Etwas stärkere Komprimierbarkeit als die Standard-MPEG-Matrix. Die EQM V3LR eignet sich gut als Standard-Matrix für den 2-CD-Bereich. Sie ist HVS-optimiert, d. h. sie berücksichtigt die Ergebnisse einer IEEE-Studie zur menschlichen visuellen Wahrnehmung (HVS = Human Visual System).
Jawor's 2CD	Jawor	Etwas schwächere Komprimierbarkeit als die Standard-MPEG-Matrix. Optimiert für den 2-CD-Bereich.
EQM V3HR	Sharktooth	Spürbar niedrigere Kompression als die Standard-MPEG-Matrix. Bestens geeignet für ½-DVD-Backups. Die EQM V3HR ist HVS-optimiert.

Heini MR	Heini011	Modifikation der EQM V3HR mit etwa gleicher Kompression aber konsistenterer Qualität.
SixOfNine	Didée	Recht flexibel einsatzbar mit mittleren und hohen Bitraten; weniger gut geeignet für niedrige Bitraten. Im Gegensatz zu den bisher vorgestellten Matrizen sollten wir die SixOfNine immer mit speziell abgestimmten Xvid-Einstellungen verwenden, da sie auf einen Minimum-Quantizer von 3 ausgelegt ist (üblich ist Minimum-Quantizer 2). Am einfachsten halten wir uns dabei an ►Teegedecks Xvid-Presets aus dem Doom9.org-Forum, die im Bereich hoher Qualität die SixOfNine einsetzen. Es existieren zwei Varianten, eine mit und eine ohne HVS-Optimierung. Teegedecks Presets verwenden die Nicht-HVS-SixOfNine.
Heini LR 6of9	Heini011	Ist genauso wie Didées SixOfNine auf Minimum-Quant 3 ausgelegt, orientiert sich in ihrem HVS-optimierten Design aber eher an den Sharktooth-Matrizen. Das Min-Quant-3-Prinzip ist eine Baustelle, die ich auf jeden Fall irgendwann einmal intensiver bearbeiten werde. Möglicherweise gibt es dann ein Spezialkapitel dazu.

► [Teeg06]

Wir dürfen nicht den Fehler machen, die Matrizen rein anhand ihrer Kompression zu beurteilen, denn die sagt noch wenig über die tatsächliche Qualität aus. Ich persönlich verwende nahezu ausschließlich Sharktooths EQM V3LR, wenn der Platz eng wird, und ansonsten die Heini MR. Für Zeichentrickmaterial wird gerne die H.263 verwendet, auch für hohe Zielgrößen. Typische Cartoons enthalten wenig feine Details, weshalb eine detailreiche Matrix ihre Vorteile nicht ausspielen kann. Die H.263 kann durch ihren weichzeichnenden Effekt sogar vorbeugend gegen Artefakte wirken, ohne dass das weichere Bild wirklich spürbar wird. Natürlich existieren auch für Zeichentrick extra abgestimmte Custom-Matrizen. Um eine Empfehlung abzugeben, habe ich mit solchem Material allerdings zu wenig zu tun.

Abschließend wollen wir uns noch ein wenig genauer mit der Wirkungsweise einer Matrix beschäftigen. Das ist für die praktische Anwendung zwar nicht unbedingt nötig, hilft aber, eine Matrix schon anhand ihres Aufbaus einzuordnen. Dazu

müssen wir erst einmal wissen, dass der Begriff der »Quantisierungsmatrix« eigentlich falsch ist, denn es handelt sich immer um ein Paket aus zwei Matrizen. Eine ist für I-Frames zuständig und eine für P- und B-Frames.

Die Matrix setzt nach der DCT-Transformation an. An dieser Stelle liegt das Bild umgewandelt in Frequenzbereiche vor. Niedrige Frequenzen repräsentieren grobe Objektformen (z. B. die rechteckige Form einer Tischplatte), hohe Frequenzen stehen für feine Bilddetails (z. B. die Holzmaserung der Tischplatte) (vgl. Kapitel A.1.2.1). Für die hohen Frequenzbereiche ist das menschliche Auge deutlich weniger empfindlich als für die niedrigen. Das dürfte unmittelbar einleuchten: Ein Qualitätsverlust in der Holzmaserung ist weniger kritisch als wenn die Form der Tischplatte selbst nicht mehr ganz korrekt dargestellt wird. Diese Tatsache macht sich die Matrix zunutze. Betrachten wir zur Veranschaulichung eine grafische Darstellung der EQM V3HR.

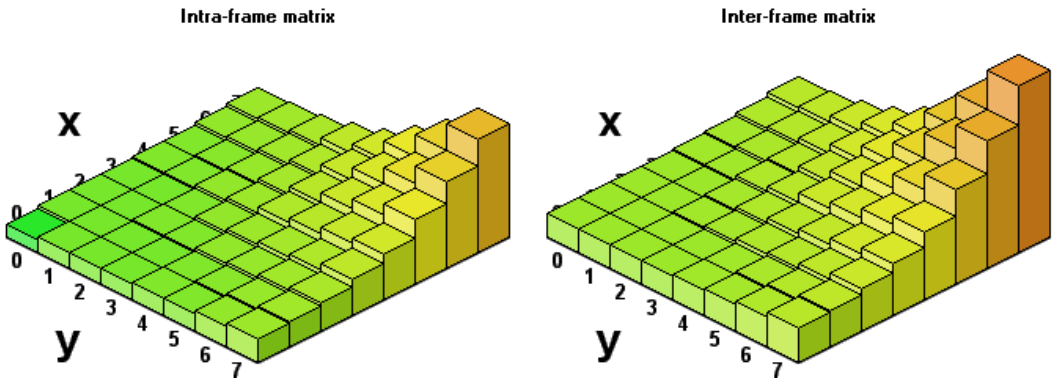


Abb. A-17: Grafische Darstellung der EQM V3HR (Quelle: LigHs CQM-Editor).

Sowohl die Intramatrix für I-Frames (links im Bild) als auch die Intermatrix für P- und B-Frames (rechts im Bild) besteht aus 8×8 Feldern, wobei links oben die niedrigsten und rechts unten die höchsten Frequenzen stehen. Jedes Feld der Matrizen ist mit einer Zahl zwischen 1 und 255 belegt – je höher der Wert, desto aggressiver werden an dieser Stelle des Frequenzspektrums Details entfernt. Im Bild stehen kurze grüne Balken für niedrige Werte, lange rote Balken für hohe Werte. Sehr schön wird die Grundphilosophie deutlich: dass niedrige Frequenzen wichtiger sind als hohe und deshalb sanfter behandelt werden sollen.

Prinzipiell könnte man für jeden Film eine extra abgestimmte Matrix erstellen. Dazu verleiten lassen sollten wir uns nicht, denn um überhaupt eine brauchbare Matrix zu entwerfen, sollte man schon gute Kenntnisse der internen Encoderabläufe, ein geschultes Auge und eine ganze Menge Geduld mitbringen. Angesichts dieses Aufwands lohnt es sich nicht, für jeden Film eine individuelle Matrix zu bauen. Eine der vorhandenen, mit Expertenwissen entwickelten und gut getesteten Matrizen zu verwenden, ist die bessere Wahl.

Rate/Distortion-Optimierung (VHQ, Trellis)

Videoencoding ist von massenweise Abwägungen und Kompromissen geprägt, u. a. von einer sinnvollen Balance zwischen benötigter Datenrate und erzielter Qualität. Diese Entscheidung müssen wir manuell bei der Auswahl der Zielgröße treffen, doch auch der Xvid-Encoder besitzt Funktionen, die etwas ähnliches tun. Das passiert auf Makroblock-Basis und nennt sich *Rate/Distortion Optimization* (RDO). Dabei werden für einen Block mehrere unterschiedliche Szenarien berechnet, um das beste Preis/Leistungs-Verhältnis zu erreichen. Der Preis sind die verbrauchten Bits, die Leistung die erzielte Qualität. Dabei kann es durchaus sinnvoll sein, eine etwas geringere Qualität in Kauf zu nehmen, wenn damit überproportional Bits gespart werden. Denn diese Bits stehen dann an anderer Stelle zur Verfügung, wo sie vielleicht nötiger sind.

RDO ist deshalb ein Tool, das wir mit Vorsicht einsetzen müssen, denn je nach Situation sind andere Einstellungen nötig, um optimale Qualität sicherzustellen. Xvid kennt zwei RDO-Methoden: VHQ und Trellis.

► **VHQ** ist die wichtigere Funktion. Die Abkürzung steht laut Koepi für *Vastly Hyped Quality*. Syskin meint, *Very Handsome Qaligosaur* wäre ihm genauso recht, und da kann ich nur zustimmen. :-D VHQ hat für P-Frames vier Abstufungen (1 bis 4), die jeweils mehr Bits einsparen und immer langsamer laufen. Für B-Frames existiert nur VHQ 1, da die möglichen Einsparungen der höheren Stufen in keinem sinnvollen Verhältnis zum zusätzlichen Geschwindigkeitsverlust stehen. Schon VHQ 4 für P-Frames verlängert das Encoding massiv, in manchen meiner Tests bis aufs Doppelte im Vergleich zu abgeschaltetem VHQ!

Entsprechend der Funktionsweise der RDO sparen hohe VHQ-Stufen zwar eine Menge Bits, entfernen aber evtl. auch feinste sichtbare Details. Deshalb beschränken wir VHQ 3 und 4 lieber auf hochkomprimierte Encodings. Bei diesen sollte außerdem das B-Frame-VHQ aktiv sein. Transparente ½-DVD-Encodings sind dagegen mit VHQ 1 bestens bedient, denn dabei steht einem moderaten Bremseffekt eine recht ordentliche Bitrateneinsparung gegenüber, ohne dass feinste Details spürbar angegriffen werden. VHQ für B-Frames ist in einer solchen Situation eher davon abhängig, ob wir uns die zusätzliche Encodingzeit gönnen wollen. Ich lasse es meistens aktiviert.

► **Trellis** ist weit weniger wichtig als VHQ. Es bremst wenig, bringt aber auch nur eine kleine Qualitätssteigerung. Da wir keine Nebenwirkungen befürchten müssen, können wir die Funktion ruhig immer aktivieren.

Die Profile von MPEG-4 Visual

MPEG-4 Visual ist ein umfangreicher Standard. Um besonders die Hardwareimplementierung zu vereinfachen, hat die MPEG eine Reihe von Profilen definiert, die wiederum in Level unterteilt sind. Für jedes Level sind Vorgaben zur maximalen Auflösung, Bitrate und den erlaubten Encodingfeatures definiert. Es existieren das *Simple Profile* (SP) mit den Leveln 0 bis 3 und das *Advanced Simple Profile* (ASP) mit den Leveln 0 bis 5. Solange wir für den PC encodieren, brauchen wir uns keine Restriktionen auferlegen, die folgenden Tabellen sind dann nur Hintergrundinformation am Rande. Hardwareplayer sind empfindlicher. Besonders auf hohe Bitraten sowie aktiviertes QPel und GMC reagieren viele Geräte allergisch. Leider haben sich in der Praxis mehr die DivX-Profile als die aus dem Standard durchgesetzt. :-(

Die nachfolgenden Tabellen bieten eine Übersicht über die Profile und Level. Die Fähigkeit zur *MPEG-Matrix* in Tab. A-10 schließt dabei Custom-Matrizen mit ein. *AQ* steht für Adaptive Quantization.

Tab. A-10: MPEG-4 Visual Encoder-Tools.

	B-Frames	QPel	GMC	MPEG-Matrix	AQ	Interlaced
Simple, Level 0	Nein	Nein	Nein	Nein	Nein	Nein
Simple, ab Level 1	Nein	Nein	Nein	Nein	Ja	Nein
Advanced Simple	Ja	Ja	Ja	Ja	Ja	Ja

Tab. A-11 zeigt die Restriktionen für den *VBV* (Video Buffer Verifier), mit dem die Bitrate kontrolliert wird. Festgelegt wird dabei, wie viele Bits der Puffer maximal enthalten darf und welchen Wert die Bitrate maximal erreichen kann. Wenn das entsprechende Profil und Level eingestellt ist, sollte sich Xvid an diese Beschränkungen halten, auch auf Kosten der Qualität. Leider funktioniert das nicht. Wer VBV benötigt, sollte besser DivX verwenden.

Tab. A-11: MPEG-4 Visual Video Buffer Verifier (VBV).

	Max. Puffergröße (Bits)	Max. Bitrate (kbit/s)
Simple @ L0	163 840	64
Simple @ L1	163 840	64
Simple @ L2	655 360	128

Simple @ L3	655 360	384
Advanced Simple @ L0	163 840	128
Advanced Simple @ L1	163 840	128
Advanced Simple @ L2	655 360	384
Advanced Simple @ L3	655 360	768
Advanced Simple @ L4	1 310 720	3 000
Advanced Simple @ L5	1 835 008	8 000

Tab. A-12 stellt die Restriktionen für Auflösung und Bildrate dar. Falls nötig müssen wir die aber manuell einhalten, denn Xvid tastet Auflösung und Bildrate nicht an, auch wenn sie den Rahmen des gewählten Profils/Levels sprengen. Die Maximalwerte sind lediglich fürs Simple Profile, Level 0 als definitives Maximum im MPEG-4-Standard festgeschrieben. Alle anderen stellen unverbindliche Obergrenzen dar, die sich in der Praxis eingebürgert haben.

	Max. Auflösung	Max. Bildrate (fps)
Simple @ L0, L1	176 × 144	15
Simple @ L2, L3	352 × 288	15
Advanced Simple @ L0, L1	176 × 144	30
Advanced Simple @ L2, L3	352 × 288	30
Advanced Simple @ L4	352 × 576	30
Advanced Simple @ L5	720 × 576	30

Tab. A-12: MPEG-4 Visual Auflösung und Bildrate.

A.4.3.1 Xvid Kommandozeilen-Konfiguration

XvidEncraw 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.

XvidEncraw ist in einem normalen Xvid-Download nicht enthalten. Auch die Entwicklung erfolgt nicht direkt im Rahmen des Xvid-Projekts. Für aktuelle De-

► [squi06] tails und die aktuelle Version sollten wir den ►Entwicklungsthread im Doom9-Forum verfolgen. XvidEncraw hat inzwischen ein stabiles Stadium erreicht, so dass Änderungen an den bestehenden Optionen eher unwahrscheinlich sind. Dass Neues hinzukommt, ist allerdings nicht auszuschließen.

Grundsätzliche Syntax

Je nachdem, ob wir ein 2-Pass- oder 1-Pass-Encoding durchführen, müssen wir mit XvidEncraw anders umgehen. Was die beiden Encodingmethoden tun und wann welche davon angebracht ist, haben wir uns schon im Kapitel A.4.1 *1-Pass vs. 2-Pass: Encodingmethoden* angesehen.

Ein 2-Pass-Encoding mit XvidEncraw erfordert auch zwei Befehle. Zuerst wird Encraw mit den Optionen für den 1st Pass aufgerufen, anschließend erneut mit den Optionen für den 2nd Pass. Den grundlegenden Aufbau der beiden Kommandozeilen sehen wir uns jetzt an, zuerst den **1st Pass**.

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -pass1 "<Statistikdatei>"
-progress 50 [weitere Optionen]
```

Die erste Option übergibt an Encraw den Dateinamen der Quelldatei, die zweite definiert, dass es sich um ein AviSynth-Skript handelt. Das anschließende `-pass1` aktiviert den ersten Encodingdurchgang und legt den Namen der Statistikdatei fest, in der die Informationen aus dem 1st Pass abgelegt werden. Dann definieren wir das Intervall in Frames, nach dem Xvid seine Statusmeldung am Bildschirm aktualisieren soll. Der Wert ist natürlich Geschmackssache. Jedenfalls sind die 50 großzügig genug, um den Encodingprozess nicht zu bremsen.

Diese Grundstruktur bleibt für den 1st Pass immer gleich. Anschließend folgen alle weiteren Encoder-Einstellungen, die wir uns weiter unten genauer ansehen. Ist der 1st Pass beendet, starten wir den **2nd Pass**, der die Zieldatei erzeugt.

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -pass2 "<Statistikdatei>"
-o "<Zieldatei>" -size <Zielgröße in KByte> -overhead 0 -par <PAR>
-progress 50 [weitere Optionen]
```

Die Grundstruktur für den 2nd Pass ähnelt dem 1st Pass. Die Angaben zur Quelldatei, Statistikdatei und Bildrate müssen identisch zum 1st Pass sein! Fett markierte Bestandteile der Kommandozeile stehen für zusätzliche oder geänderte Optionen.

Mit `-pass2` definieren wir den zweiten Encodingdurchgang. Anschließend steht die Option `-o`, gefolgt vom Dateinamen des Zielvideos. Das Format dieses

Videos wird von der Dateiendung bestimmt. Mit der Endung `.m4v` erzeugen wir einen rohen MPEG-4-Datenstrom, was dann nötig ist, wenn der fertige Film im MP4-Container liegen soll. Für AVI oder Matroska als Zielcontainer geben wir entsprechend `.avi` oder `.mkv` als Endung an. Anschließend folgt mit `-size` die Zielgröße in KByte. Gemeint ist die Größe der reinen Videospur, nicht die Zielgröße des kompletten Films – also Gesamtgröße abzüglich Audiospuren und Untertitelspuren, abzüglich Containeroverhead. Da wir wegen des Overheads einen größeren Exkurs in die technischen Tiefen der einzelnen Container einlegen müssen, sparen wir uns die manuelle Berechnung und verlassen uns auf das Encoding-Frontend. Danach schalten wir mit `-overhead` Xvids automatische Berücksichtigung des Containeroverheads ab. Mit `-par` legen wir schließlich das Pixel Aspect Ratio des Zielvideos fest. Für ein klassisches Encoding mit quadratischen Pixeln gilt immer `-par 1`, oder wir lassen die Option ganz weg.

Wie im 1st Pass folgen nach der Grundstruktur alle weiteren Encoder-Einstellungen. In der Regel sollten die identisch zum 1st Pass sein.

Auch das **Single-Pass**-Verfahren benötigt eine ähnliche Kommandozeile. Hauptunterschied ist der, dass wir XvidEncraw nur einmal aufrufen, denn der Encoder erstellt ja sofort die Zieldatei. Da der Bitraten-Modus für ein DVD-Backup qualitativ zu schlecht ist, sehen wir uns nur den Quantizer-Modus (CQ) an.

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -o "<Zieldatei>"
-cq <Quantizer> -par <PAR> -progress 50 [weitere Optionen]
```

Die Angaben zu Größe (`-size`), Overhead (`-overhead`) und Nummer des Durchgangs (`-pass`) fallen weg, da sie im 1-Pass-Verfahren keine Bedeutung haben. Stattdessen verwenden wir die Option `-cq`, mit der wir den gewünschten Quantizer – d. h. das Qualitätsniveau – angeben.

Beispiel einer Konfiguration

Um ein wenig mehr Gefühl für Encraw zu bekommen, betrachten wir ein Beispiel für eine typische Konfiguration. Dazu stellen wir uns eine fiktive DVD vor, die 100 Minuten Spielzeit aufweist und incl. zwei AC-3-Tonspuren (je 384 kbit/s) auf ½ DVD Zielgröße encodiert und in den Matroska-Container verpackt werden soll.

Beim Schreiben der Kommandozeilen nutzen wir sämtliche Standardwerte von XvidEncraw aus, d. h. wir tippen nur das, was auch tatsächlich ausdrücklich angegeben werden muss. Wer aus der VfW-Welt kommt, sollte im Hinterkopf be-

halten, dass Xvid VfW und Xvid_Enraw nicht überall die gleichen Standardwerte verwenden.

```
xvid_encraw.exe -i "D:\Quelle.avs" -type 2 -pass1 "D:\xvid.stats"
-progress 50 -bquant_ratio 162 -bquant_offset 0 -bvhq -qtype 1 -qpel
-vhqmode 2 -zones 0,w,1,0
```

```
xvid_encraw.exe -i "D:\Quelle.avs" -type 2 -pass2 "D:\xvid.stats"
-o "D:\Zielvideo.mkv" -size 1730175 -par 4 -overhead 0 -progress 50
-bquant_ratio 162 -bquant_offset 0 -bvhq -qtype 1 -qpel -vhqmode 2
-zones 0,w,1,0
```

Speichern wir nicht auf DVD, sondern auf Festplatte, ist die exakte Zielgröße nicht so wichtig. Das Beispiel als 1-Pass-Encoding könnte wie im Folgenden aussehen. Wir verwenden Quantizer 2, um die beste Qualität ohne irrsinnig große Datei herauszuholen.

```
xvid_encraw.exe -i "D:\Quelle.avs" -type 2 -o "D:\Zielvideo.mkv" -par 4
-progress 50 -cq 2 -bquant_ratio 162 -bquant_offset 0 -bvhq -qtype 1
-qpel -vhqmode 2 -zones 0,w,1,0
```

Genaueres zum Einsatz des Quantizers gibt es im allgemeinen Xvid-Kapitel ab Seite 67.

Referenz der Optionen

Dieser Abschnitt listet mit einer kurzen Erklärung viele Optionen auf, die für XvidEncraw zur Verfügung stehen. Im Wesentlichen handelt es sich um eine etwas ausführlichere Übersetzung des Befehls `xvid_encraw.exe -help`. Wir beschränken uns aber auf die wirklich nützlichen Optionen. Besonders die detaillierten Optionen für den 2nd Pass fehlen, denn die wären eher Thema für ein Spezialkapitel.

Optionen, die Standardwerte besitzen, müssen wir nur dann angeben, wenn wir einen abweichenden Wert verwenden wollen. Ansonsten arbeitet XvidEncraw automatisch mit dem Standard. Optionen ohne Standardwert sind nur dann aktiv, wenn wir sie ausdrücklich angeben.

Quelldatei

`-i <Quelldateiname>`
Standardwert: `stdin`
Beispiel: `-i "D:\Pfad zur\Quelldatei.avs"`
Pfad und Name der Quelldatei. Dabei dürfte es sich meistens um das AVS-Skript handeln. Sind Leerzeichen im Pfad enthalten, muss er in Anführungszeichen gesetzt werden.

`-type <Quelltyp>`
Werte: `0` (YUV), `1` (PGM), `2` (AVS/AVI)
Beispiel: `-type 2`
Typ der Quelldatei. Für unsere Zwecke kommt fast ausschließlich Typ `2` in Frage.

Zieldatei

`-o <Zieldateiname>`
Beispiel: `-o "D:\Pfad zur\Zieldatei.mkv"`
Definiert den Namen der Zieldatei. Die Dateiendung bestimmt deren Format. Möglich sind `.avi`, `.mkv` und `.m4v` für AVI, Matroska und rohen (containerlosen) MPEG-4-Datenstrom. Letzterer ist nötig, wenn der fertige Film im MP4-Container verpackt sein soll.

`-par <x[:y]>`
Werte: `1` (1:1, Standard), `2` (4:3 PAL), `3` (4:3 NTSC), `4` (16:9 PAL), `5` (16:9 NTSC), *x*`:`*y* (Benutzerdefiniert: *Breite:Höhe*)
Beispiele: `-par 4` oder `-par 16:11`
Schreibt das Pixel Aspect Ratio in den Videostrom der Zieldatei. Das entspricht dem Setzen des AR-Flags, von dem im Anamorph-Kapitel auf Seite 53 die Rede ist. Für klassische Encodings mit quadratischen Pixeln geben wir immer `-par 1` an oder lassen die Option ganz weg. Für die anderen Standard-PARs dienen die Kurzformen, die dem MPEG-4-Standard entsprechend für die Werte aus Tab. A-4 (Seite 47) stehen. Die Custom-Variante `-par x:y` ist nur für PARs gedacht, die nicht im Standard direkt definiert sind.

Für Custom-PARs definiert MPEG-4 Visual zwei Felder (für `x` und `y`), die je einen vorzeichenlosen 8-Bit-Integer enthalten. D. h. der maximale Wert, der darin gespeichert werden kann, ist 255. Xvid versucht zwar, größere Werte in den erlaubten Bereich umzurechnen, doch das führt oft zu hohen Abweichungen. Deshalb sollten wir zu große Werte niemals eingeben.

`-framerate <fps>`
Typische Werte: `23.976`, `25.0` (Standard), `29.97`, `30.0`
Beispiel: `-framerate 25.0`
Setzt die Bildrate der Zieldatei in Bildern pro Sekunden (fps). Den Punkt und *nicht* das Komma zu verwenden, ist Pflicht. XvidEncraw bezieht den richtigen Wert automatisch von AviSynth, weshalb wir diese Option nur in Ausnahmefällen benötigen.

MPEG-4-Tools

`-max_bframes <Maximum>`
Standardwert: `2`
Beispiel: `-max_bframes 1`
Definiert die Anzahl B-Frames, die maximal direkt hintereinander stehen dürfen. Eine `0` schaltet B-Frames komplett ab. Empfehlenswert ist in der Regel der Standardwert `2`. Mit den B-Frame-Optionen insgesamt haben wir uns im allgemeinen Xvid-Kapitel ab Seite 68 schon ausführlich beschäftigt.

`-bquant_ratio <Verhältnis>`
Standardwert: `150`
Beispiel: `-bquant_ratio 162` (Empfehlung)
Definiert den multiplikativen Faktor zur Berechnung des B-Frame-Quantizers als ganzzahligen Wert. Um den eigentlichen Faktor zu erhalten, wie er auch in der VfW-GUI angegeben wird, muss der Wert durch 100 geteilt werden. `150` bedeutet also ein Faktor von 1,50. Siehe auch Seite 68 f.

`-bquant_offset <Offset>`
Standardwert: `100`
Beispiel: `-bquant_offset 0` (Empfehlung)
Definiert den additiven Faktor zur Berechnung des B-Frame-Quantizers als ganzzahligen Wert. Um den eigentlichen Faktor zu erhalten, wie er auch in der VfW-GUI angegeben wird, muss der Wert durch 100 geteilt werden. `100` bedeutet also ein Faktor von 1,00. Siehe auch Seite 68 f.

`-bvhq`
Aktiviert den VHQ-Modus für B-Frames. Da es sich dabei immer um VHQ 1 handelt, leidet die Geschwindigkeit nicht allzu sehr. Da B-Frames meistens den größten Anteil der Frames in einem Film stellen, sollten wir das B-Frame-VHQ möglichst aktivieren.

`-nopacked`
Deaktiviert den Packed-Bitstream-Modus für B-Frames im AVI-Container. Nützlich ist die Funktion dann, wenn wir den Film manuell splitten wollen. Ist die Zieldatei keine AVI, hat diese Option keine Bedeutung.

`-qpel`
Schaltet die Quarter-Pixel-Bewegungssuche ein. QPel erhöht die Genauigkeit, mit der die Bewegungssuche arbeitet, von einem halben auf ein Viertel Pixel. Das senkt zwar die Komprimierbarkeit des Films ein wenig, wirkt sich aber positiv auf Details und Schärfe aus. Von extrem stark komprimierten 1-CD-Encodings abgesehen, sollten wir QPel deshalb lieber aktivieren.

`-gmc`
Aktiviert Global Motion Compensation. GMC versucht die Kompression zu erhöhen, indem es nach gemeinsamen Bewegungsvektoren in der Szene sucht. Zooms und Kameraschwenks bieten sich dafür besonders an. GMC ist ein eher unbeliebtes Feature, da sowohl Encoding als auch Decoding die CPU spürbar mehr beanspruchen, der Nutzen aber meist fraglich bleibt. Dass im technisch fortschrittlicheren AVC-Standard GMC nicht verfügbar ist, weist deutlich darauf hin, dass sich die Funktion nicht bewährt hat.

`-lumimasking`
Aktiviert die adaptive Quantisierung (AQ). Dieses Element von MPEG-4 Visual erlaubt es, dass die Makroblocks eines Frames unterschiedliche Quantizer erhalten dürfen. Das wird dazu benutzt, die Stärke der Kompression lokal anzupassen, je nachdem, wie empfindlich das menschliche Sehsystem für den Bildinhalt an dieser Stelle ist. Xvids eingebauter Algorithmus ist das nicht sehr leistungsfähige Lumimasking. Es gibt jedoch auch eine angepasste Xvid-Version, die stattdessen den von x264 portierten leistungsfähigen ►VAQ-Algorithmus (Variance AQ) enthält. ► [DaSh08]

Der Schalter `-lumimasking` aktiviert den jeweiligen AQ-Algorithmus der verwendeten Xvid-Version. VAQ sollten wir bevorzugen und immer verwenden, da es dem alten Lumimasking deutlich überlegen ist.

`-qtype <Modus>`
Werte: `0` (H.263, Standard), `1` (MPEG-4)
Beispiel: `-qtype 1`
Wählt den Modus der Quantisierung und damit eine der beiden Standard-Quantisierungsmatrizen. MPEG-4 (`1`) erzeugt ein etwas schärferes Bild auf Kosten der Kompression und eignet sich eher für die höhere Datenrate einer großzügigen

Zielgröße. H.263 (`0`) ergibt dagegen ein etwas weicheres Bild und erhöht damit die Kompression ein wenig; geeignet eher für die niedrigen Datenraten von 1-CD-Encodings. Näheres zum Thema Matrizen haben wir im allgemeinen Xvid-Kapitel ab Seite 71 schon besprochen.

`-qmatrix <Matrixdateiname>`
Beispiel: `-qmatrix "C:\Matrizen\SixOfNine.xcm"`
Verwendet eine Custom-Quantizer-Matrix. Dazu geben wir den Pfad zur Matrixdatei an, der in Anführungszeichen gesetzt werden muss, wenn Leerzeichen enthalten sind. Sollte nicht gemeinsam mit `-qtype` verwendet werden. Siehe auch Seite 71 f.

Encoding-Modus

`-cq <Quantizer>`
Werte: `1.0` bis `31.0`
Beispiel: `-cq 2`
Führt ein Single-Pass-Encoding mit einem bestimmten Quantizer durch. I. d. R. verwendet für äußerst hochqualitative Backups mit Quantizer 2. Kann nicht mit `-size` bzw. `-bitrate` kombiniert werden.

Da encoderintern als Quantizer nur ganze Zahlen gültig sind, repräsentiert diese Option zwei Encoding-Modi. Wenn wir als Wert eine ganze Zahl angeben, wird diese als konstanter Quantizer gewählt. Kommazahlen entsprechen einem Target-Quantizer. Dabei werden unterschiedliche Quantizer so beim Encoding angewendet, dass als Durchschnitt ungefähr der gewünschte Wert erreicht wird. Xvid verwendet für diese Verteilung einen extrem einfachen, kaum optimierten Algorithmus, weshalb wir uns auf ganze Zahlen und damit das echte CQ-Verfahren beschränken sollten. Siehe auch Seite 67 f.

`-pass1 [<Statistikdateiname>]`
Beispiel: `-pass1 "D:\Encoding\Xvid.stats"`
Definiert den ersten Durchgang eines 2-Pass-Encodings. Der angegebene Dateiname steht für die Statistikdatei, in der die ermittelten Daten des 1st Pass gespeichert werden. Geben wir keinen Dateinamen an, erzeugt Xvid eine Datei namens *Xvid.stats* im aktuellen Ordner.

`-full1pass`
Normalerweise werden im 1st Pass einige unkritische Optionen automatisch abgeschaltet, um die Geschwindigkeit zu erhöhen. Dieses Verhalten können wir mit

`-full1pass` unterbinden. Sinnvoll ist das nur, wenn wir im 1st Pass eine Videodatei erzeugen, die weiterverwendet werden soll. I. d. R. ist das nicht der Fall.

`-pass2 [<Statistikdateiname>]`
Beispiel: `-pass2 "D:\Encoding\Xvid.stats"`
Definiert den zweiten Durchgang eines 2-Pass-Encodings, in dem die Zieldatei erstellt wird. Die angegebene Statistikdatei muss mit der aus dem 1st Pass identisch sein. Geben wir keinen Dateinamen an, nimmt XvidEncraw als Standardwert die *Xvid.stats* im aktuellen Ordner.

`-size <KByte>`
Beispiel: `-size 1425466`
Zielgröße in KByte. Diese Angabe ist die reine Größe der Videospur, nicht die Zielgröße des kompletten Films. Wird `-size` angegeben, kann nicht gleichzeitig `-bitrate` angegeben werden.

`-bitrate <kbit/s>`
Beispiel: `-bitrate 1500`
Bitrate des Zielvideos in Kilobit pro Sekunde. Wird `-bitrate` angegeben, kann nicht gleichzeitig `-size` angegeben werden.

Weitere Encoding-Optionen

`-max_key_interval <Frames>`
Standardwert: `300`
Beispiel: `-max_key_interval 250`
Maximales I-Frame-Intervall. Xvid setzt automatisch nur dort I-Frames (Keyframes), wo sie sinnvoll sind (z. B. bei Szenenwechseln). Nach der mit `-max_key_interval` angegebenen Anzahl an P- und B-Frames wird spätestens ein I-Frame erzwungen, auch wenn dort automatisch keines gesetzt würde. Als Daumenregel hat sich das Zehnfache der Bildrate eingebürgert. Deshalb können wir den Standardwert meistens problemlos übernehmen.

`-quality <Modus>`
Werte: Ganze Zahlen von `0` bis `6`. Standardwert: `6`
Beispiel: `-quality 5`
Legt fest, wie intensiv Xvid nach Bewegungen sucht (motion search precision). Es gibt kaum einen Grund von `6` abzuweichen. `0` schaltet die Bewegungssuche ganz ab, was einen Film zufolge hat, der ausschließlich aus I-Frames besteht.

`-vhqmode <Modus>`
Werte: `0`, `1` (Standard), `2`, `3`, `4`
Beispiel: `-vhqmode 4`
VHQ rechnet für die einzelnen Makroblocks verschiedene Szenarien durch und entscheidet sich dann für das mit dem besten Bits/Qualität-Verhältnis (siehe Seite 75). Stark komprimierte 1-CD-Encodings dürfen VHQ `3` oder `4` verwenden. Für schwächere Kompression bei höheren Zielgrößen ist `1` oder `2` angebrachter. Wenn wir außerdem GMC aktiviert haben, sollte VHQ nicht abgeschaltet sein.

`-nochromame`
Üblicherweise verwendet Xvid bei der Suche nach Bewegung nicht nur die Helligkeitsinformationen (Luminanz), sondern auch die Chrominanz (Farbe). Das steigert die Genauigkeit der Ergebnisse und damit die Qualität. Mit `-nochromame` kann die Verwendung der Chrominanz deaktiviert werden. Xvid nutzt dann nur noch die Luma-Komponente zur Bewegungssuche.

`-notrellis`
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. Mit `-notrellis` können wir die Funktion deaktivieren.

`-turbo`
`-Turbo` beschleunigt die Berechnung von B-Frames und QPel, erreicht deshalb aber nicht immer das absolute Qualitätsmaximum. Der Unterschied ist kaum jemals sichtbar, trotzdem lasse ich als Qualitätsfanatiker den Turbo ausgeschaltet.

Zonen

Zonen definieren Abschnitte innerhalb des Videos, für die unabhängig eine Reihe von Optionen definiert werden können. Offensichtlichster Anwendungsbereich für Zonen ist der Abspann des Films. Für die Definition von Zonen ist die Option `-zones` zuständig, deren Syntax folgendermaßen aussieht:

```
-zones Start,Modus,Wert[,Optionen][/Start,Modus,Wert[,Optionen]]...
```

Die einzelnen Parameter einer Zone trennen wir mit Komma (`,`). Zwischen mehreren Zonen steht ein Schrägstrich (`/`) als Trennzeichen, wobei maximal 64 Stück definiert werden können. Das Ende einer Zone ergibt sich entweder aus dem

Startframe der anschließenden Zone oder dem Ende des Films. Die Zonen-Parameter haben folgende Bedeutung:

► `Start`: Framenummer, bei der die Zone beginnt. Wichtig: Das erste Frame des gesamten Films hat die Nummer 0.
► `Modus`: Zonen können entweder mit einem konstanten Quantizer oder einem relativen Gewicht versehen sein. Für Quantizer-Zonen steht als Modus ein `q`, für gewichtete Zonen ein `w` (Weight).
► `Wert`: Für Quantizer-Zonen steht hier der gewünschte Quantizer als ganze Zahl zwischen `1` und `31`. Für Weight-Zonen steht hier das relative Gewicht im Vergleich zum Rest des Films. Eine Zone, der nur die Hälfte der üblichen Datenrate zugewiesen werden soll, müsste mit Gewicht `0.5` definiert werden. Als Dezimaltrenner muss der Punkt verwendet werden, nicht das Komma.
► `Optionen`: Eine Zeichenkette, die bestimmte Encoder-Optionen für die Zone ein- oder ausschaltet. Jedes konfigurierbare Feature wird von einem einzelnen Buchstaben repräsentiert.
`K` = Keyframe. Am Anfang der Zone wird ein I-Frame erzwungen.
`O` = Chroma Optimizer. Wirkt Pixeltreppchen an scharfen Kanten entgegen, bremst aber den Encoding-Vorgang etwas. Obwohl ich mir nicht sicher bin, dass es wirklich sichtbare Verbesserungen bringt, lasse ich die Option immer eingeschaltet.
`G` = Greyscale. Verwirft alle Farbinformationen, so dass wir ein Schwarzweiß-Video erhalten.
`C` = Cartoon Mode. Besonders nützlich bei Cartoons – wer hätte das gedacht, und da wiederum profitiert klassisch gezeichneter Zeichentrick à la *Tom & Jerry* oder *The Simpsons* am meisten. Für Realfilme sollten wir die Funktion lieber nicht einschalten.
`Zahl` = B-Frame-Empfindlichkeit. Wirkt sich darauf aus, wie gern Xvid B-Frames setzt. Erlaubt sind nur ganze Zahlen. Positive Werte ermutigen Xvid zu mehr B-Frames, negative Werte schrecken den Encoder eher ab. Für 1-CD-Encodings kann es sinnvoll sein, den Wert auf etwa `5` bis `10` hoch zu setzen. Ansonsten brauchen wir den Standard von `0` nicht verändern.

Machen wir uns diese Syntax an einem Beispiel klar. Nehmen wir an, wir möchten für einen Film drei Zonen definieren:
► Aktivierten Chroma Optimizer am Filmanfang,
► ab Frame 12 000 konstanten Quantizer 2 ohne weitere Optionen,
► ab Frame 13 000 Gewicht von 0,8 sowie alle Optionen aktiviert und die B-Frame-Empfindlichkeit auf –10 gesetzt.

Die dazu passende Kommandozeile sieht folgendermaßen aus:

```
-zones 0,w,1.0,O/12000,q,2/13000,w,0.8,KOGC-10
```

Drei Dinge sind wichtig: Für die erste Zone haben wir Gewicht 1,0 angegeben. Es soll ja nur der Chroma Optimizer aktiviert und die Zone ansonsten genauso wie der Rest des Films behandelt werden. Da `Start`, `Modus` und `Wert` immer angegeben werden müssen, war das nötig.

Wenn wir keine `Optionen` aktivieren, entfällt auch das letzte Komma. Das sehen wir an der zweiten Zone. Außerdem setzen wir innerhalb des `Optionen`-Parameters nie ein Komma oder sonstiges Trennzeichen. Genauso darf im gesamten Zonen-String kein Leerzeichen auftauchen.

Steuerung von XvidEncraw

`-threads <Anzahl>`
Standardwert: `1`
Verfügbar ab Xvid 1.2. Hier stellen wir ein, wie viele Threads Xvid zum Encodieren verwenden soll. Für Computer mit nur einer CPU sollten wir die Option weglassen. 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 mal 1,5 ist ein guter Startwert.

`-progress <Frames>`
Standardwert: `10`
Legt fest, wie oft die Fortschrittsanzeige aktualisiert wird. Ein Wert von `1` gibt für jedes Frame einige Detailinformationen aus. Bei höheren Werten beschränkt sich Xvid auf allgemeine Informationen.

A.4.3.2 Xvid VfW-Konfiguration

Xvid ist ein sehr genau konfigurierbarer Codec, und das klassische VfW-Interface lässt kaum eine Option aus. Da es gerade bei der Konfiguration eines Videoencoders kein absolut richtig oder falsch gibt, ist es keine schlechte Idee, ein wenig querzulesen. Selurs ▸*Wissenswertes rund um Xvid* kann ich jedem empfehlen. ▸ [Selu05]

Wir betrachten zuerst die Xvid-Optionen, die nur für ein 2-Pass-Encoding Bedeutung haben; anschließend dann die speziellen Optionen für 1-Pass mit kon-

stantem Quantizer (CQ). Zum Schluss folgen alle Einstellungen, die unabhängig von der ▸Encodingmethode immer verfügbar sind. ▸A.4.1

Optionen fürs 2-Pass-Encoding

Wenn wir den Film auf eine feste Zielgröße encodieren, benötigen wir zwei Encoder-Durchläufe: einen 1st Pass zur Analyse der Videos und einen 2nd Pass, in dem der Film tatsächlich encodiert wird. Die Einstellungen für beide Durchläufe sollten immer identisch sein – ausgenommen natürlich die Einstellung, um welchen Pass es sich handelt.

Einstellungen für den 1st Pass

Profile und deren Level schalten – wie auf Seite 71 besprochen – einzelne Optionen des Codecs frei oder sperren sie, um Konformität zu den gleichnamigen MPEG-4-Profilen zu gewährleisten. Für ein normales DVD-Backup spricht nichts dagegen, mit **(unrestricted)** sämtliche Freiheiten zu erhalten (Abb. A-18). Über den **More**-Button erreichen wir den Dialog für die Feineinstellungen, den wir weiter unten im Detail besprechen. Besonders bei niedrigen Profilen sind evtl. einige Optionen im **More**-Dialog deaktiviert. Für Profil und Level wählen wir im 2nd Pass die gleiche Einstellung wie im 1st Pass.

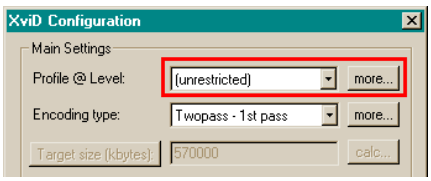


Abb. A-18: Hauptfenster oben der Xvid-Konfiguration.

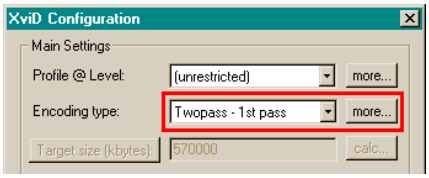


Abb. A-19: Einstellung für 1st Pass im Xvid-Hauptdialog.

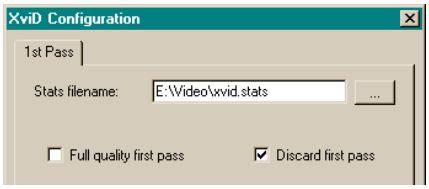


Abb. A-20: Xvids 1st-Pass-Optionen.

Als nächstes kümmern wir uns um den **Encoding type**. Mit **Twopass - 1st pass** stellen wir den ersten Codierdurchgang ein (Abb. A-19) und rufen über den **More**-Button die Details auf (Abb. A-20). Die Optionen in diesem Fenster gelten ausschließlich für den 1st Pass. 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 automatisch abgeschaltet (z. B. VHQ), die nur bremsen und im 1st Pass noch nichts bringen. **Full quality first pass** würden wir nur

dann einschalten, wenn wir die Videodatei des 1st Pass behalten wollten. 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.

Die Spezialitäten des 1st Pass sind damit erledigt. Bleibt noch, die globalen Optionen zu konfigurieren, was wir weiter unten besprechen.

Einstellungen für den 2nd Pass

Im Hauptdialog wählen wir zuerst das gleiche **Profile@Level** wie im 1st Pass. Dann setzen wir den **Encoding type** auf **Twopass - 2nd pass** ein (Abb. A-21). In der Regel kümmert sich das Encoding-Frontend um die richtige **Target size**, so dass wir dort nichts eingeben müssen. Wenn wir die Zielgröße doch manuell festlegen: Hierhin gehört die Größe der reinen Videospur, nicht die Zielgröße des kompletten Films – also Gesamtgröße abzüglich der Audiospuren und Untertitelspuren, abzüglich des Containeroverheads. Da wir wegen des Overheads einen größeren Exkurs in die technischen Tiefen der einzelnen Container einlegen müssen, sparen wir uns die manuelle Berechnung.

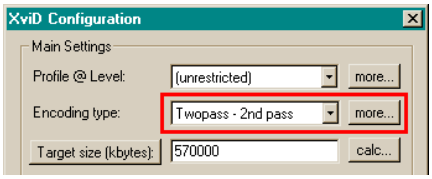


Abb. A-21: Einstellung für 2nd Pass im Xvid-Hauptdialog.

Ein Klick auf den **More**-Button bringt uns zu den Details, die wir allerdings komplett und guten Gewissens auf den Standardeinstellungen belassen können. Lediglich sollten wir kontrollieren, dass die gleiche Statistikdatei wie für den 1st Pass angegeben ist.

Damit sind die 2nd-Pass-Spezialitäten auch schon erledigt. Alle globalen Optionen setzen wir auf die selben Werte wie im 1st Pass.

Optionen fürs 1-Pass-CQ-Encoding

Beim 1-Pass-Encoding ohne exakte Zielgröße ist für uns nur der Konstanter-Quantizer-Modus (CQ) interessant. Die speziellen CQ-Einstellungen erledigen wir im Hauptdialog.

Den **Encoding type** stellen wir auf **Single Pass**. Im Feld darunter tragen wir den gewünschten Quantizer ein. Evtl. müssen wir vorher mit einem Klick auf den Button links daneben auf **Target quantizer** umschalten. Mehr Informationen zum Quantizer bietet das allgemeine Xvid-Kapitel ab Seite 67.

Damit sind die CQ-Einstellungen auch schon erledigt. Der **More**-Button neben

Encoding type braucht uns nicht zu interessieren, da dessen Optionen nur für den Bitratenmodus gelten. Es fehlen also nur noch die globalen Optionen.

Globale Optionen

Alle folgenden Einstellungen sind immer verfügbar, unabhängig von der gewählten Encodingmethode. Fürs 2-Pass-Encoding sollten wir alle globalen Optionen im 1st und 2nd Pass identisch konfigurieren.

Beginnen wir mit den Einstellungen, die direkt mit dem MPEG-4-Standard zusammenhängen. Wir erreichen den passenden Dialog (Abb. A-23), indem wir im Hauptfenster (Abb. A-22) auf den **More**-Button neben **Profile@Level** klicken. Je nachdem, welches Profil ausgewählt ist, sind eventuell einige der folgenden Optionen nicht zur Verfügung.

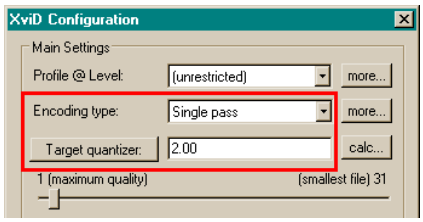


Abb. A-22: Einstellung für 1-Pass CQ im Xvid-Hauptdialog.

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. Über **MPEG-Custom** und den Button **Edit Matrix** können wir eine Custom-Matrix laden. Näheres zum Thema haben wir ab Seite 71 schon besprochen.

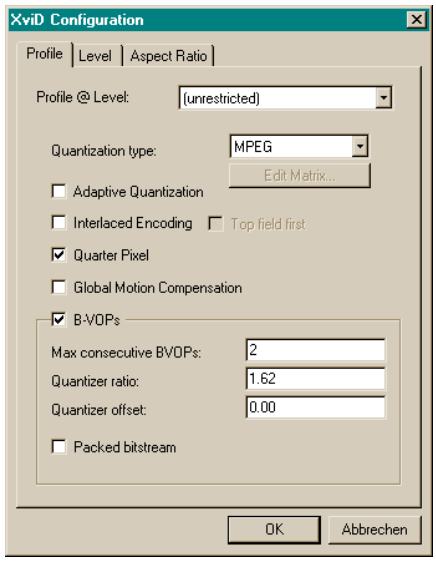


Abb. A-23: Konfiguration der MPEG-4-Tools.

Die Option **Adaptive Quantization** (AQ) ist ein Element von MPEG-4 Visual und ermöglicht es, dass die Makroblocks eines Frames unterschiedliche Quantizer erhalten dürfen. Das wird dazu benutzt, die Stärke der Kompression lokal anzupassen, je nachdem, wie empfindlich das menschliche Sehsystem für den Bildinhalt an dieser Stelle ist. Xvids eingebauter Algorithmus ist das nicht sehr leistungsfähige Lumimasking. Es gibt jedoch auch eine angepasste Xvid-Version, die stattdessen den von x264 portierten leistungsfähigen ►VAQ-Algorithmus (Variance AQ) enthält. Ein Haken bei **Adaptive Quantization** aktiviert den jeweiligen AQ-Algorithmus der verwendeten Xvid-Version. VAQ sollten wir bevorzugen und immer

►[DaSh08]

verwenden.

Quarter Pixel erhöht die Genauigkeit, mit der die Bewegungssuche arbeitet, von einem halben auf ein Viertel Pixel. Das senkt zwar die Komprimierbarkeit des Films ein wenig, wirkt sich aber positiv auf Details und Schärfe aus. Von extrem stark komprimierten 1-CD-Encodings abgesehen, sollten wir QPel deshalb lieber aktivieren.

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. GMC ist ein eher unbeliebtes Feature, da sowohl Encoding als auch Decoding die CPU spürbar mehr beanspruchen, der Nutzen aber meist fraglich bleibt. Dass im technisch fortschrittlicheren AVC-Standard GMC nicht verfügbar ist, weist deutlich darauf hin, dass sich die Funktion nicht bewährt hat.

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. **Quantizer ratio** und **Quantizer offset** bestimmen, wie hoch B-Frames im Vergleich zu den umgebenden I- oder P-Frames komprimiert werden. Im Detail haben wir uns mit diesen Optionen schon im allgemeinen Xvid-Kapitel ab Seite 68 beschäftigt. In der Regel empfehle ich ein Maximum von **2**, Ratio **1.62** und Offset **0.00**.

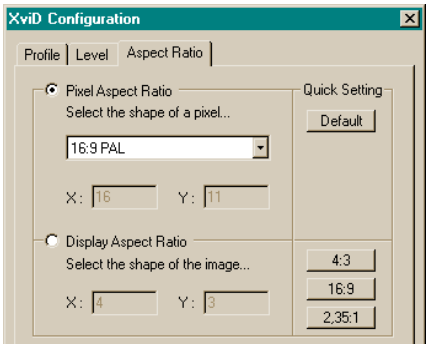


Abb. A-24: Xvids Seitenverhältnis-Auswahl.

Packed Bitstream bestimmt, wie die B-Frames in der Videodatei gespeichert werden und sollte normalerweise ausgeschaltet sein. Doch keine Regel ohne Ausnahme: Wenn wir den AVI-Container verwenden und für mehrere CDs encodieren, so dass der Film zum Schluss gesplittet werden muss, vereinfacht aktivierter **Packed Bitstream** das Schneiden.

Damit wechseln wir zum Register Aspect Ratio (Abb. A-24). Unter **Pixel Aspect Ratio** erlaubt Xvid die Angabe eines PAR, das 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 MPEG-4 (vgl. Tab. A-4, Seite 47). Außerdem können wir mit **Custom** die beiden Felder unter der Combobox aktivieren, um einen beliebigen PAR anzugeben.

Für Custom-PARs definiert MPEG-4 Visual zwei Felder (für **x** und **y**), die je einen vorzeichenlosen 8-Bit-Integer enthalten. D. h. der maximale Wert, der darin gespeichert werden kann, ist 255. Xvid versucht zwar, größere Werte in den erlaubten Bereich umzurechnen, doch das führt oft zu hohen Abweichungen. Deshalb sollten wir zu große Werte niemals eingeben.

Die zweite Möglichkeit ist die Angabe des **Display Aspect Ratio**. Früher hieß diese Option noch verwirrenderweise *Picture Aspect Ratio*, hat aber schon immer das DAR bezeichnet. Wenn wir ein DAR angeben, ist es wichtig, ganze Zahlen zu verwenden. 2,35:1 müssten wir demnach als 235:100 angeben (oder gekürzt als 47:20). Für ein DVD-Backup fällt mir allerdings kein guter Grund ein, das DAR dem PAR vorzuziehen. Außerdem muss das DAR intern in ein PAR umgerechnet werden, wodurch Abweichungen entstehen können.

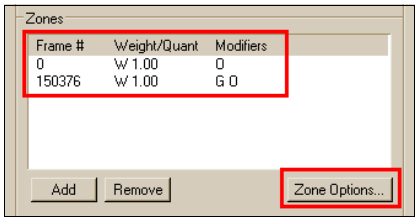


Abb. A-25: Zonenliste im Xvid-Hauptdialog.

Damit kehren wir zurück ins Hauptfenster und kümmern uns um die **Zones** (Abb. A-25). Zonen definieren Abschnitte innerhalb des Videos, für die unabhängig eine Reihe von Optionen definiert werden können. Eine Zone, die sich über den ganzen Film erstreckt, existiert immer. Über den Button **Zone Options** gelangen wir in den Konfigurationsdialog.

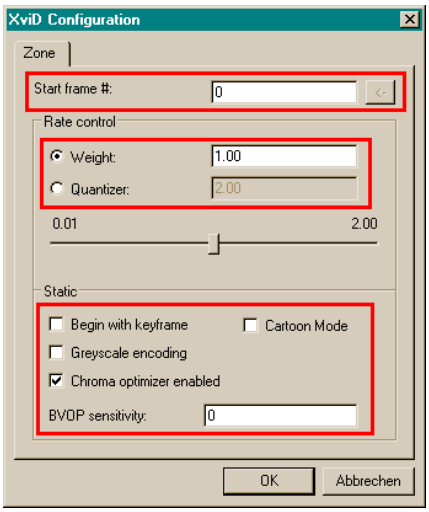


Abb. A-26: Detailkonfiguration der Xvid-Zonen.

Bei **Start frame #** (Abb. A-26) 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.

Zonen können entweder mit einem konstanten **Quantizer** 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 besonders nützlich bei Cartoons – wer hätte das gedacht – und da wiederum profitiert klassisch gezeichneter Zeichentrick à la *Tom & Jerry* oder *The Simpsons* am meisten. Für Realfilme sollten wir die Funktion lieber nicht einschalten. 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.

Normalerweise definieren wir nur eine einzige Zone über den ganzen Film. Wer Wert auf eine exakt funktionierende Kapitelliste legt, könnte jedoch pro Kapitel eine Zone anlegen und jeweils am Anfang ein Keyframe erzwingen, damit der Player sauber und schnell zu einem bestimmten Kapitel springen kann.

Über **OK** gelangen wir zurück in den Hauptdialog, wo wir uns im unteren Teil des Fensters um den **More**-Abschnitt kümmern (Abb. A-27). Hier stellen wir das **Quality Preset** auf (**User defined**) und öffnen über den **More**-Button den Dialog für die Detaileinstellungen (Abb. A-28).

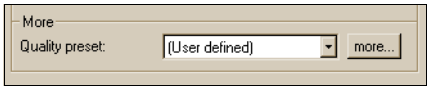


Abb. A-27: Xvids »Quality Preset«

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.

VHQ rechnet für die einzelnen Makroblocks verschiedene Szenarien durch und entscheidet sich dann für das mit dem besten Bits/Qualität-Verhältnis (siehe Seite 75). Stark komprimierte 1-CD-Encodings dürfen VHQ **3** oder **4** verwenden. Für schwächere Kompression bei höheren Zielgrößen ist **1** oder **2** angebrachter. Wenn wir außerdem 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. Da B-Frames meistens den größten Anteil der Frames in einem Film stellen, sollten wir das B-Frame-VHQ möglichst nicht deaktivieren.

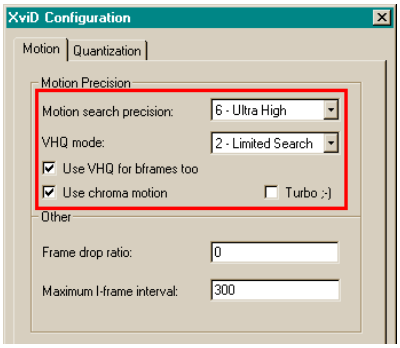


Abb. A-28: Konfiguration der Xvid-Featues.

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 Qualitätsliebhaber lasse ich Turbo trotzdem ausgeschaltet.

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. 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.1.1 über manuelles Splitting (Seite 211).

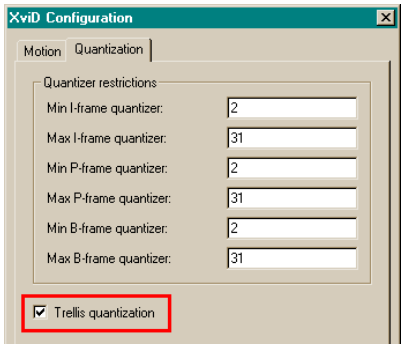


Abb. A-29: Xvids Quantisierungsoptionen

Im Register **Quantization** (Abb. A-29) 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.

Der große Abschnitt **Quantizer restrictions** ist fürs 2-Pass-Encoding interessant. Im 1-Pass-CQ-Modus beachtet Xvid ausschließlich den im Hauptdialog definierten Quantizer. Sämtliche **Min Quantizer** vom Standardwert **1** auf **2** zu setzen, sollten wir nicht versäumen. **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 sinnvolle Wert. Falls wir die Komplexität eines Film einmal grob überschätzen, gibt uns Xvid mit der zu kleinen Datei einen Hinweis und auch die Möglichkeit zur Anpassung. Eine größere Audiospur oder ein wiederholtes Encoding mit größerer Auflösung ist allemal besser als mit Quantizer 1 verschwendete Dateigröße.

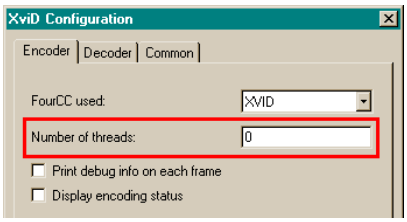


Abb. A-30: Anzahl Encodingthreads ab Xvid Version 1.2.

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 Abb. A-30 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 mal 1,5 ist ein guter Startwert.

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.

A.4.3.3 Zuordnung von Xvids VfW- und CLI-Optionen

Wer von Xvid VfW auf Xvid CLI umsteigen oder beide parallel nutzen möchte, steht vor der Frage: Welche Option in der VfW-GUI entspricht welchem Schalter an der Kommandozeile? Das beantwortet die folgende Tab. A-13, deren fette Überschriften für die entsprechenden Dialogfenster der Xvid VfW-GUI stehen. Nähere Erklärungen oder Empfehlungen zu den Optionen finden wir hier nicht, denn dafür ist der Rest des Xvid-Konfigurationskapitels ab Seite 66 zuständig.

Tab. A-13: Zuordnung der Optionen von Xvid VfW und XvidEncraw.

Hauptfenster	
Profile @ Level	`-vbvsize` und `-vbvmax`.
Encoding Type	`-pass1` und `-pass2` fürs 2-Pass-Encoding sowie `-cq` und `-single` für 1-Pass.
Target Size	`-size`
Target Bitrate	`-bitrate`
Target Quantizer	`-cq`
Zone Options	`-zones`
Profile @ Level / More / Profile	
Quantization type	`-qtype`, für Custom-Matrizen `-qmatrix`

Adaptive Quantization	`-lumimasking`
Interlaced Encoding	`-interlaced`
Quarter Pixel	`-qpel`
Global Motion Compensation	`-gmc`
Max BVOPs	`-max_bframes`
B-Frame Quant Ratio	`-bquant_ratio`
B-Frame Quant Offset	`-bquant_offset`
Packed Bitstream	`-nopacked`

Profile @ Level / More / Aspect Ratio

Pixel Aspect Ratio	`-par`
Display Aspect Ratio	Keine Entsprechung. Xvid rechnet intern sowieso jedes DAR in das dazu passende PAR um.

Encoding Type (1st Pass) / More

Full quality first pass	`-full1pass`
Discard first pass	Kein eigener Schalter. Wir verzichten im 1st Pass auf die Angabe einer Zieldatei.

Encoding Type (2nd Pass) / More

I-Frame boost	`-kboost`
I-Frames closer than...	`-kthresh`
...are reduced by	`-kreduction`
Overflow control strength	`-ostrength`
Max Overflow Improvement	`-oimprove`
Max Overflow Degradation	`-odegrade`
High Br Degradation	`-chigh`
Low Br Improvement	`-clow`

Quality Preset / More / Motion

Motion Search Precision	`-quality`
VHQ Mode	`-vhqmode`

VHQ for B-Frames	`-bvhq`
Chroma Motion	`-nochromame`
Turbo ;-)	`-turbo`
Frame Drop Ratio	`-drop`
Max I-Frame Interval	`-max_key_interval`
Quality Preset / More / Quantization	
Min I-Frame Quantizer	`-imin`
Max I-Frame Quantizer	`-imax`
Min P-Frame Quantizer	`-pmin`
Max P-Frame Quantizer	`-pmax`
Min B-Frame Quantizer	`-bmin`
Max B-Frame Quantizer	`-bmax`
Trellis Quantization	`-notrellis`

A.4.4 Der x264-Encoder

X264 (MPEG-4 Part 10 AVC/H.264) und Xvid (MPEG-4 Part 2 Visual) sind zweifellos die beiden beliebtesten Open-Source-Encoder. Für hochauflösende HighDef-Videos hat x264 klar die Nase vorn, denn Xvid wird hier kaum eingesetzt. Beim DVD-Backup dagegen ist nach wie vor Xvid weit verbreitet. Doch das ändert sich, je weniger die höheren Systemanforderungen von AVC-Video dank moderner Computer eine Rolle spielen und je mehr sich x264 zu einem allseits bekannten Standard-Encoder mausert. Außerdem treibt er durch die starke Konzentration auf seinen Kommandozeilenencoder die Bewegung weg von VfW voran, was man nur uneingeschränkt gutheißen kann.

Damit sind wir auch schon beim Thema. Das VfW-Interface für x264 wird offiziell nicht mehr unterstützt. Da sich x264 nur mit Hängen und Würgen überhaupt einigermaßen kompatibel zu VfW verhält und die VfW-Oberfläche eine Reihe von Einstellungsmöglichkeiten vermissen lässt, kann ich nur davon abraten. Es mag ein paar Spezialfälle geben, in denen H.264 in AVI sinnvoll ist; das DVD-Backup

gehört jedoch nicht dazu. Deshalb beschäftigen wir uns im Folgenden ausschließlich mit der CLI-Version, dem »echten« x264-Encoder.

A.4.4.1 Die allgemeine Konfiguration von x264

Weitere Literatur: [Fied06] [ITU07b] [Rich03] [RichMJ] [Symes04] [Wieg03]

H.264 weist einige deutliche Unterschiede und Neuerungen dem älteren Standard MPEG-4 Visual gegenüber auf. In diesem Kapitel betrachten wir die neuen und besonders wichtigen Funktionen, die eine ausführlichere Erklärung verdienen. Im anschließenden Kapitel zur Encoder-Konfiguration sind die hier empfohlenen Werte immer angegeben, so dass wir uns nicht unbedingt erst durch dieses doch recht umfangreiche Kapitel wühlen müssen.

Mehrfache Referenzen und IDR-Frames

MPEG-4 Visual ASP definiert für jedes P-Frame genau ein Bild, das als Referenz für die Suche nach Bewegung benutzt werden darf, und zwar das direkt vorangehende I/P-Frame. MPEG-4 AVC hat diese Beschränkung nicht mehr. Ein P-Frame darf auf bis zu sechzehn Referenzframes verweisen. Dadurch steigt das Potenzial enorm, eine gute Übereinstimmung in der Bewegungssuche zu finden, und je besser die Übereinstimmung, desto höher kann das Bild komprimiert werden, ohne die Qualität nach unten zu ziehen. Also sind mehrere Referenzframes eine gute Sache. Allerdings bekommen wir diesen Vorteil nicht gratis. Die Anforderung an Speicher und CPU-Leistung steigt, und zwar sowohl beim Encoding als auch beim Abspielen.

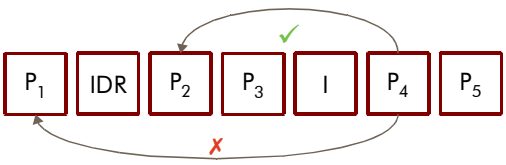


Abb. A-31: I-Frames und IDR-Frames in AVC.

Außerdem entsteht durch die Mehrfachreferenzen ein weiteres Phänomen. Es existieren in AVC zwei verschiedene Arten von I-Frames: normale I-Frames und IDR-Frames (Kurzform für *instantaneous decoding refresh*). Sehen wir uns die Bildsequenz aus Abb. A-31 an. Beide I-Frame-Typen sind vollständige Einzelbilder, d. h. sie selbst enthalten keine Referenzen auf andere Frames. Referenzen über einfache I-Frames hinweg sind aber möglich. Es wäre also einwandfrei erlaubt, dass das Frame P_4 über das I-Frame hinweg auf P_2 referenziert. Die Konsequenz daraus ist die, dass ein einfaches I-Frame nicht als Punkt taugt, an dem wir den Film schneiden können, denn damit würden wir P_4 zumindest einen Teil seiner

Referenzen wegnehmen. Bildfehler am Anfang der zweiten Datei wären die Folge. Keyframes in dem Sinn, wie wir sie kennen, sind nur IDR-Frames, denn eine Referenz über ein IDR-Frame hinweg ist nicht erlaubt. P_4 darf also niemals auf P_1 oder noch frühere Frames verweisen. Das heißt auch, dass fürs Springen und Schneiden nur IDR-Frames relevant sind und wir einfache I-Frames genauso wie P- und B-Frames behandeln müssen.

x264 bietet uns die Möglichkeit, den minimalen und maximalen Abstand zwischen zwei IDR-Frames zu definieren. Mit den üblichen Bildraten einer DVD brauchen wir die Vorgabewerte aber kaum jemals zu verändern.

Partitionierung von Makroblocks

Der AVC-Standard erlaubt es, einen 16×16 Pixel großen Makroblock in kleinere Einheiten aufzuteilen, so genannte Partitionen. Diese werden dann getrennt voneinander betrachtet, d. h. sie erhalten eigene Bewegungsvektoren und können sogar unterschiedliche Frames als Referenz verwenden. Zwar benötigt die Partitionierung einigen Speicherplatz zu Verwaltung, doch der wird durch das große Potenzial zu Erhöhung der Kompression wieder aufgewogen. Besonders in den Teilen des Bilds, wo viel Bewegung stattfindet, lohnen sich kleine Partitionsgrößen. Statische Bildteile profitieren dagegen nur wenig. Deshalb entscheidet x264 dynamisch, für welchen Makroblock welche Partitionierung am nützlichsten ist.

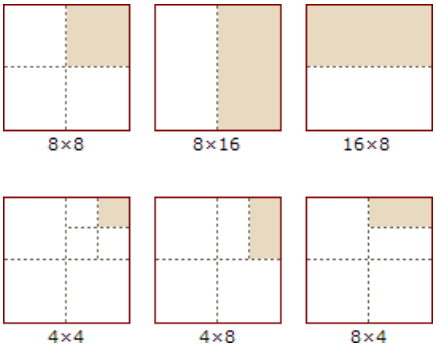


Abb. A-32: Partitionierung von Makroblocks in H.264.

In der Konfiguration können wir einstellen, welche Partitionsgrößen x264 überhaupt berücksichtigt. Die erste Reihe in Abb. A-32 mit den 8er-Größen ist für P- und B-Frames möglich. Dazu kommen im AVC High Profile die I-Frames, falls wir zusätzlich die 8×8-DCT-Transformation einschalten. Die zweite Reihe mit den 4er-Größen funktioniert für I- und P-Frames (unabhängig vom AVC-Profil), nicht aber für B-Frames.

Je kleiner die Partitionierung, desto mehr Speicherplatz benötigt die Verwaltung. Für einen unpartitionierten 16×16-Makroblock fällt etwas vereinfacht die Speicherung des Bewegungsvektors und des verwendeten Referenzframes als Overhead an. Wird der Block nun aufgeteilt, müssen diese beiden Daten für jede Partition gespeichert werden. Für die 8×8-Partitionierung bedeutet das eine Vervierfachung des Overheads. Ein vollständig in 4×4-Partitionen unterteilter Block benötigt schon 16 mal so viel Verwaltungsspeicher. Deshalb ist ein AVC-Encoder

praktisch gezwungen, die Partitionsgrößen je nach Bildinhalt zu variieren. Ansonsten wäre der Kompressionsvorteil durch den enormen Anstieg des Overheads schnell wieder zunichte gemacht.

Was heißt das nun praktisch für die Encoder-Konfiguration? Da alle wichtigen Decoder das AVC High Profile beherrschen, spricht nichts dagegen, sämtliche Partitionierungsmöglichkeiten zu aktivieren. Zwar ist damit ein deutlicher Geschwindigkeitsverlust verbunden, andererseits entsteht durch die dynamische Partitionierung ein riesiges Potenzial zu Verbesserung der Bewegungssuche und Bewegungskompensierung. Da es sich dabei um zentrale Bestandteile des Encodingvorgangs handelt, ist die Rechenzeit gut angelegt. Ohne volle Flexibilität in der Partitionierung würden wir x264 eines wichtigen Teils seiner Kompressionseffizienz berauben. Bevor wird also die Partitionen einschränken, sollten wir immer zuerst an anderer Stelle nach Möglichkeiten zur Steigerung der Geschwindigkeit suchen.

Konfiguration von B-Frames

Das Konzept der B-Frames haben wir schon im Kapitel über die Interframe-Kompression auf Seite 12 angesprochen. Hier beleuchten wir die bidirektionalen Bilder aus der Sicht von x264, denn der Encoder bietet eine gewaltige Zahl von Stellschrauben zur Konfiguration.

Anzahl und Verteilung

Drei Optionen sind dafür zuständig, die Anzahl der B-Frames und ihre Verteilung im Film festzulegen.

► Die **maximale B-Frame-Anzahl** definiert, ob überhaupt B-Frames verwendet werden, und wenn ja, wie viele maximal direkt hintereinander stehen dürfen. Mit einem Maximum von zwei wäre eine solche Bildsequenz aus I-Frames, P-Frames und B-Frames möglich: IPBBP, bei maximal drei B-Frames wäre es dann IPBBBP usw. Wichtig: Die Einstellung definiert nur das Maximum. Es bedeutet *nicht,* dass *immer* genau so viele B-Frames hintereinander stehen. Lediglich können *niemals mehr* als das Maximum aufeinander folgen. Innerhalb dieser Grenze berechnet x264 selbst die günstigste Möglichkeit.
► **B-Frame-Bias** beeinflusst x264s Lust auf B-Frames. Negative Werte halten den Codec eher von B-Frames ab; sie werden deshalb seltener gesetzt und die maxi-

male aufeinanderfolgende Anzahl weniger oft erreicht. Analog dazu führen positive Werte zu einem vermehrten B-Frame-Einsatz.

► Die **adaptive Verteilung** verwendet einen Algorithmus, der B-Frames auf eine »intelligente« Art und Weise setzt. »Intelligent« heißt, dass in langsamen Szenen mehr B-Frames auftauchen als in schnellen. Da gerade in langsamen Szenen die Bidirektionalität ihren größten Vorteil entfaltet, ist das so auch sinnvoll.

B-Frames sollten wir immer verwenden, denn sie sind ein Werkzeug, das die Dateigröße deutlich in den Keller ziehen kann ohne die Qualität spürbar zu beeinflussen. Entsprechend groß fällt deshalb der Nachteil aus, wenn wir sie abschalten. Die Entscheidung, wie viele B-Frames hintereinander stehen, können wir i. d. R. bedenkenlos x264 überlassen. D. h. wir setzen die erlaubte B-Frame-Anzahl auf den höchstmöglichen Maximalwert von 16. Wer dem Encoder nicht so weit vertrauen mag, sollte für hohe Kompression etwa 5 B-Frames als Maximum und für sanftere Kompression 2 bis 3 Stück einstellen.

x264s Lust auf B-Frames (Bias) benötigt normalerweise keine Anpassung. Auch sollten wir die adaptive Verteilung auf jeden Fall zulassen.

Quantisierung und Bewegungssuche

Um die Quantisierung und Bewegungssuche in B-Frames zu beeinflussen, stehen uns noch einmal sechs Optionen zur Verfügung. Eine davon betrifft die Partitionierung der Makroblocks, womit wir uns weiter oben schon beschäftigt haben. Die restlichen Einstellungen besprechen wir im Folgenden. Bis auf die letzte Option (RDO) geht es immer darum, reine Verbesserungen zu erreichen, d. h. die Qualität soll durch diese Tools ohne Nebenwirkung steigen.

► Die **B-Frame-Pyramide** beeinflusst die erlaubten Referenzframes. Ist die Funktion abgeschaltet, sieht die Suche nach Referenzen aus wie in Abb. A-33. Ein B-Frame darf kein anderes B-Frame als Referenz verwenden, sondern immer nur die vorangehenden oder folgenden I/P-Frames. B_2 darf also auf P_1 und P_2 verweisen, nicht aber auf B_1 oder B_3. Das ist der klassische Modus, wie ihn auch Xvid und alle anderen ASP-Codecs verwenden. Allerdings lässt sich mit der B-Pyramide die Effizienz noch steigern (Abb. A-34). B-Frames dürfen jetzt sowohl die umliegenden I/P-Frames als auch schon vorhandene B-Frames als Referenz nutzen. B_2 kann also weiterhin auf P_1 und P_2 verweisen, aber auch auf B_1. Eine Referenz auf zukünftige B-Frames (B_3) ist nicht möglich, da B_3 noch gar nicht existiert, wenn B_2 encodiert wird.

► B-Blocks im **Direct-Modus** speichern keinen eigenen Bewegungsvektor, sondern berechnen ihn entweder aus den räumlichen Unterschieden innerhalb eines Frames (spatial) oder aus den zeitlichen Unterschieden zwischen verschiedenen Frames (temporal). x264 kann darüber hinaus in einem automatischen Modus selbst entscheiden, in welcher Situation welche Methode sinnvoller ist.

► Die **bidirektionale Bewegungssuche** erlaubt anstatt nur einem zwei Bewegungsvektoren pro Makroblock bzw. Blockpartition. Damit steigt die Anzahl möglicher Referenzblocks deutlich an, und das bedeutet eine höhere Wahrscheinlichkeit dafür, eine richtig gute Referenz zu finden.

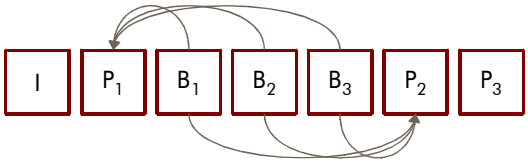


Abb. A-33: H.264-B-Frames ohne I *B-Pyramide.*

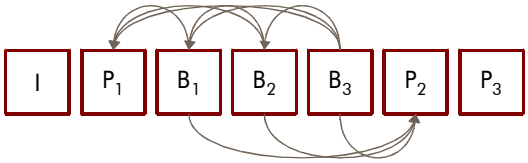


Abb. A-34: H.264-B-Frames mit B-Pyramide.

► Bei aktivierter **gewichteter Bewegungskompensierung** können die gefundenen Referenzen in einer beliebigen Gewichtung verwendet werden, was v. a. dazu beiträgt, Ein-, Aus- und Überblendungen effizienter zu speichern.

► **RDO für B-Frames** benutzt einen Algorithmus zur Rate/Distortion-Optimierung. Der dient zur Suche nach dem besten Preis/Leistungs-Verhältnis zwischen Datenrate und Qualität.

Ob wir die ganzen Funktionen aktivieren, hängt mit Ausnahme von B-RDO vor allem davon ab, ob wir uns die zusätzliche Encodingzeit gönnen wollen. Ich würde den Direct-Modus auf automatisch stellen, um x264 die richtige Entscheidung zu überlassen, und die anderen Optionen aktivieren, denn Qualität geht vor Geschwindigkeit.

Für das B-Frame-RDO treffen grundsätzlich die allgemeinen Erklärungen aus dem Xvid-Kapitel (Seite 75) zu. Allerdings ist die Gefahr, wegen B-RDO feinste Bilddetails zu verlieren, geringer als bei Xvid und hängt außerdem von der restlichen Konfiguration ab. Deshalb kann B-RDO auch für sanft komprimierte HQ-Backups durchaus eine sinnvolle Wahl sein, wenn der hohe Rechenaufwand und damit die geringe Encodinggeschwindigkeit nicht dagegen spricht.

Der Inloop-Deblocking-Filter

Deblocking-Filter kennen wir möglicherweise von Decodern, die damit beim Abspielen Blockartefakte im Bild zu übertünchen versuchen. Sowohl die Decoder

von Xvid und DivX als auch ffdshow besitzen diese Funktion. Der H.264-Standard definiert einen Deblocker, der schon während des Encodings zum Einsatz kommt und damit keine beliebig zuschaltbare Zusatzfunktion ist, sondern ein Bestandteil der Encoding-Konfiguration.

Entsprechend dem Standard setzt x264 den Deblocking-Filter ein nachdem ein Bild codiert wurde, aber bevor es als Referenz für das nächste Bild dient. Dadurch kann dieses Bild mit einer Referenz arbeiten, die weniger Artefakte enthält. Das tut der Qualität gut. Der Nachteil ist die sinkende Geschwindigkeit, und zwar sowohl beim Encoding als auch beim Decoding. Denn damit das zweite Bild korrekt decodiert werden kann, muss das vorangehende Bild in gefilterter Form vorliegen. Das ähnelt einem Xvid-Encoding, bei dem wir gezwungen sind, beim Anschauen einen Deblocking-Filter zu aktivieren.

Sehen wir uns den vergrößerten Bildausschnitt aus Abb. A-35 an, der extrem stark komprimiert wurde. Das linke Bild ist die Version ohne Deblocker und weist deutliche Makroblock-Artefakte auf. Rechts sehen wir genau das gleiche Bild, diesmal aber mit aktivem Deblocking-Filter. Die Artefakte sind übertüncht, was in der Regel optisch besser aussieht und sich außerdem besser als Referenzbild eignet.

Abb. A-35: Bild mit und ohne Deblocking-Filter.

Die Stärke des Filters wird über zwei Stellschrauben geregelt, Strength und Threshold, oder entsprechend den offiziellen technischen Bezeichnungen Alpha- und Beta-Deblocking. Der Wert 0 steht jeweils für die Standardeinstellung des Filters. Negative Werte führen zu schwächerem Deblocking, positive zu stärkerem. Ein 1-CD-Encoding verträgt etwas stärkeres Filtering, da es mehr unter Blockbildung leidet: 1:2 bietet sich an. Für sanfter komprimierte Filme sollten wir mit schwächeren Werten arbeiten, um nicht unnötig Details zu entfernen. 0:0 oder 0:–1 sind brauchbare Standardwerte für die Praxis.

Natürlich ist es auch möglich, das Deblocking ganz genau abzustimmen, indem wir die Alpha- und Beta-Werte individuell anpassen. Das Thema hat *.mp4

guy im Doom9.org-Thread ►*How To Use Mpeg4 AVC Deblocking Effectively* näher beleuchtet. Kurz zusammengefasst: Der Alpha-Wert legt insgesamt die Stärke des Deblockings fest; Beta regelt, wie aggressiv Bildstrukturen als Detail oder Artefakt eingeordnet werden, und passt je nach »Artefaktanteil« die Filterstärke an. ► [mp4g06b]

Quantisierungsmatrizen

AVC unterstützt genauso wie MPEG-4 ASP Quantisierungsmatrizen, einschließlich Custom-Matrizen. Im Wesentlichen treffen alle Erklärungen aus dem Xvid-Kapitel (Seite 71 f.) auch hier zu. Der technische Hauptunterschied besteht darin, dass eine »Matrix« nicht wie bei den ASP-Encodern aus zwei Matrizen, sondern gleich aus acht Stück besteht. Die beiden Standardmatrizen sind

► **Flat**, bei der soweit ich weiß alle Positionen mit dem gleichen Wert belegt sind,
► **JVT**, die vom Joint Video Team im Standard vorgeschlagen wird.

Benutzerdefinierte Matrizen sind noch nicht so zahlreich verfügbar wie für Xvid. Im Wesentlichen haben sich bisher Sharktooth und *.mp4 guy als AVC-Matrizenbauer hervorgetan, und zwar in den Doom9.org-Threads ►*EQM AVC Series* und ►*Custom Matrices*. Die dort geposteten Matrizen können wir einfach in einen Texteditor kopieren und so wie sie sind als reinen Text abspeichern. Meistens wird .cfg als Dateiendung verwendet. ► [Shar05a] ► [mp4g06a]

Für eine irgendwie geartete Empfehlung habe ich zu wenig Erfahrung mit AVC-Matrizen. x264 verwendet in der Standardeinstellung jedenfalls die Flat.

Quantizer und Constant Rate Factor

Vereinfacht ausgedrückt ist der Quantizer ein Faktor, der die Stärke der Kompression regelt. Hochoffiziell heißt er auch *quantiser scale parameter,* abgekürzt QP. Er darf je nach Konfiguration für jedes Frame oder jede Makroblockpartition verschieden sein und liegt bei MPEG-4 AVC (H.264) zwischen 1 und 51, wobei nur ganze Zahlen erlaubt sind. Je kleiner der Wert, desto sanfter die Kompression und desto besser die Qualität. Übliche QPs liegen etwa zwischen 18 und 25. QP 18 hat sich als allgemeiner Richtwert für »maximale Qualität« eingebürgert. Noch kleinere Werte vergrößern die Datei, bringen aber kaum noch oder keine sichtbaren Verbesserungen.

Beim 2-Pass-Encoding haben wir mit dem Quantizer nur indirekt zu tun, da

x264 die Verteilung selbst in die Hand nimmt, um die Zielgröße zu treffen. Beim 1-Pass-Encoding setzen wir den QP dagegen selbst. x264 unterstützt zwei Verfahren: das schon von Xvid und DivX bekannte CQ-Encoding mit festem Quantizer sowie das CRF-Verfahren (constant rate factor). Etwas genauer haben wir uns damit schon im Kapitel über die Encodingmethoden (Seite 60 ff.) beschäftigt. CRF ist bei weitem interessanter. Wenn nicht konkrete Gründe dagegen sprechen, sollten wir für 1-Pass-Encodings immer CRF anstatt CQ verwenden.

Für CRF wählen wir einen nominalen Quantizer für die tatsächlich sichtbare Qualität. Da es sich dabei um einen Basiswert handelt, sind auch Kommazahlen wie z. B. 20,4 erlaubt. x264 passt diesen Wert den Eigenschaften der jeweiligen Szene entsprechend an.

► *Schnelle Szenen* (d. h. mit hohem Anteil schneller Bewegung) erhalten einen etwas höheren Quantizer, also eine stärkere Kompression. Das funktioniert, weil wir in schnellen Szenen weniger Details wahrnehmen und deshalb mehr Informationen aus dem Bild herauskomprimiert werden können, ohne das das zu spüren wäre.
► *Langsame Szenen* (d. h. mit geringem Anteil schneller Bewegung) erhalten dagegen einen etwas niedrigeren Quantizer, also eine geringere Kompression. Da das Auge in solchen Szenen viel Zeit hat, auch winzige Details aufzunehmen, darf dort nicht gespart werden.

CRF berücksichtigt also zu einem gewissen Grad, wie das menschliche Auge arbeitet, und hat deshalb dem CQ-Verfahren gegenüber einen Qualitätsvorteil. Die Dateigröße eines CRF-Encodings bewegt sich normalerweise in der selben Region wie die eines CQ-Encodings mit selbem Quantizer, nutzt aber den Platz effizienter.

Weitere Literatur: [BrJo07c]

A.4.4.2 x264 Kommandozeilen-Konfiguration

Maßgeblich für die Kommandozeilen ist x264 in einer Version mit dem AQ-Algorithmus von Dark Shikari. Außerdem benötigen wir für unsere Zwecke mindestens AviSynth-Unterstützung und evtl. auch MP4. Beides muss beim Kompilieren ausdrücklich eingebaut werden. Es gibt mehrere Binaries, die diese Anforderungen erfüllen, z. B. die jeweils aktuelle Revision von `www.x264.nl`.

► [Selu06]

Wir sprechen hier v. a. die Funktionen von x264 an, die für ein DVD-Backup interessant sind. Eine ausführliche Erklärung zu sämtlichen Optionen bietet ►Selurs *man x264*.

Grundsätzliche Syntax

Je nachdem, ob wir ein 2-Pass- oder 1-Pass-Encoding durchführen, müssen wir mit x264 anders umgehen. Was die beiden Encodingmethoden tun und wann welche davon angebracht ist, haben wir uns schon im Kapitel A.4.1 *1-Pass vs. 2-Pass: Encodingmethoden* angesehen.

Ein 2-Pass-Encoding mit x264 erfordert auch zwei Befehle. Zuerst wird x264.exe mit den Optionen für den 1st Pass aufgerufen, anschließend erneut mit den Optionen für den 2nd Pass. Den grundlegenden Aufbau der beiden Kommandozeilen sehen wir uns jetzt an, zuerst den **1st Pass**.

```
x264.exe [Optionen] --pass 1 --bitrate <Zielbitrate in kbit/s> --progress
--no-ssim --no-psnr --stats "<Statistikdatei>" --output NUL
"<Quelldatei>"
```

Zuerst füttern wir x264 mit sämtlichen Encoderoptionen, die wir uns weiter unten genauer ansehen. Anschließend definieren wir mit `--pass 1` den ersten Encodingdurchgang. Es folgt die Angabe der Videobitrate in kbit/s. Dabei handelt es sich um die Bitrate für die reine Videospur, also Gesamtgröße abzüglich Audiospuren und Untertitelspuren, abzüglich Containeroverhead. Da wir wegen des Overheads einen größeren Exkurs in die technischen Tiefen der einzelnen Container einlegen müssten, sparen wir uns die manuelle Berechnung.

Mit `--progress` aktivieren wir die Fortschrittsanzeige des Encoders und schalten dann mit `--no-ssim` und `--no-psnr` die Berechnung von SSIM und PSNR ab. Für Tests sind diese Werte oft gut brauchbar, bei einem normalen Backup bremsen sie aber nur das Encoding.

Die Option `--stats` legt den Namen der Statistikdatei fest, in der die Informationen aus dem 1st Pass gespeichert werden. Dann folgt mit `--output` die Angabe der Zieldatei. Da wir beim ersten Durchgang noch keine Videodatei erzeugen wollen, x264 es aber nicht erlaubt, die Option wegzulassen, schicken wir mit `--output NUL` das erzeugte Video ins Datennirvana. Abschließend geben wir die Quelldatei (das AviSynth-Skript) an, die ohne spezielle Option ganz am Ende der Kommandozeile steht.

Ist der 1st Pass beendet, starten wir den **2nd Pass**, der die Zieldatei erzeugt.

```
x264.exe [Optionen] --pass {2|3} --bitrate <Zielbitrate in kbit/s>
--sar <PAR> --progress --no-ssim --no-psnr --stats "<Statistikdatei>"
--output "<Zieldatei>" "<Quelldatei>"
```

Die Grundstruktur für den 2nd Pass ähnelt dem 1st Pass. Die Angaben zur Quelldatei, Statistikdatei und Bitrate müssen identisch zum 1st Pass sein! Fett markierte Bestandteile der Kommandozeile stehen für zusätzliche oder geänderte Optionen.

x264 unterstützt beliebig viele Encodingdurchgänge. Meistens beschränken wir uns auf die gewohnten zwei, da die Qualitätssteigerung zusätzlicher Passes den enormen Zeitaufwand nicht wert ist. Für den allerletzten Encodingdurchgang geben wir die Option `--pass 2` an. Für alle »mittleren« Durchgänge müssen wir `--pass 3` setzen, damit die Statistikdatei aktualisiert wird. Das ist unbedingt nötig, da sonst keine Qualitätsverbesserung möglich ist. Falls wir uns die Türen offen halten möchten, können wir auch immer `--pass 3` verwenden. Der Geschwindigkeitsverlust, der durch die Aktualisierung der Statistikdatei entsteht, ist nicht der Rede wert, der Zeitaufwand eines kompletten zusätzlichen Encoding-Durchgangs dagegen schon. Solange nicht wirklich gewichtige Gründe dafür sprechen, ist mehr als das übliche 2-Pass-Verfahren Unsinn.

Mit der Option `--sar` legen wir anschließend das Pixel Aspect Ratio des Zielvideos fest. Das entspricht dem Setzen des AR-Flags, von dem im Anamorph-Kapitel auf Seite 53 die Rede ist. Für ein klassisches Encoding mit quadratischen Pixeln gilt immer `--sar 1:1`, oder wir lassen die Option ganz weg.

Die Zieldatei ist diesmal ein echter Dateiname. Lediglich wenn wir wissen, dass wir noch weitere Passes durchführen, können wir auch jetzt `--output NUL` verwenden, denn genau genommen ist es nur im allerletzten Pass notwendig, das erzeugte Video tatsächlich zu speichern. Die Dateiendung der Zieldatei legt fest, welches Format verwendet wird. Je nach gewünschtem Container für den fertigen Film verwenden wir entweder `.mp4` für MP4 oder `.mkv` für Matroska. Der AVI-Container ist nicht mit x264.exe kompatibel.

Die Encoderoptionen am Anfang der Kommandozeile sind für alle Passes identisch. Eine Ausnahme darf lediglich der 1st Pass bilden, und das auch nur in einem klar definierten Rahmen.

Auch das **Single-Pass**-Verfahren benötigt eine ähnliche Kommandozeile. Hauptunterschied ist der, dass wir x264 nur einmal aufrufen, denn der Encoder erstellt ja sofort die Zieldatei. Die beste Wahl fürs DVD-Backup ist der CRF-Modus, der so aussieht:

```
x264.exe [Optionen] --crf <nominaler Quant> --sar <PAR> --progress
--no-ssim --no-psnr --output "<Zieldatei>" "<Quelldatei>"
```

Die Angaben zu Bitrate (`--bitrate`), Nummer des Durchgangs (`--pass`) und Statistikdatei (`--stats`) fallen weg, da sie beim CRF-Verfahren keine Bedeutung

haben. Stattdessen verwenden wir die Option `--crf`, mit der wir den gewünschten nominalen Quantizer – d. h. das Qualitätsniveau – angeben.

Beispiel einer Konfiguration

Um ein wenig mehr Gefühl für x264 zu bekommen, betrachten wir eine Beispielkonfiguration. Dazu stellen wir uns die gleiche fiktive DVD wie im Xvid-Kapitel vor, die 100 Minuten Spielzeit aufweist und incl. zwei AC-3-Tonspuren (je 384 kbit/s) auf ½ DVD Zielgröße encodiert und in den Matroska-Container verpackt werden soll.

Beim Schreiben der Kommandozeilen nutzen wir sämtliche Standardwerte von x264.exe aus, d. h. wir tippen nur das, was auch tatsächlich ausdrücklich angegeben werden muss. Außerdem verwenden wir das Setup für den schnellen First Pass, das wir uns etwas weiter unten genauer ansehen.

```
x264.exe --partitions none --me dia --subme 1 --ref 1 --bframes 16
--direct auto --no-dct-decimate --deblock 0:-1 --pass 1 --bitrate 2362
--progress --no-ssim --no-psnr --stats "D:\x264.stats" --output NUL
"D:\Quelle.avs"
```

```
x264.exe --partitions all --8x8dct --subme 5 --ref 5 --mixed-refs
--bframes 16 --direct auto --bime --b-pyramid --trellis 1 --deblock 0:-1
--no-dct-decimate --sar 16:11 --pass 2 --bitrate 2362 --progress
--no-ssim --no-psnr --stats "D:\x264.stats" --output "D:\Zielvideo.mkv"
"D:\Quelle.avs"
```

Speichern wir nicht auf DVD, sondern auf Festplatte, ist die exakte Zielgröße nicht so wichtig. Das Beispiel als 1-Pass-Encoding könnte wie im Folgenden aussehen. Wir verwenden CRF 18, um die beste Qualität ohne irrsinnig große Datei herauszuholen.

```
x264.exe --partitions all --8x8dct --subme 5 --ref 5 --mixed-refs
--bframes 16 --direct auto --bime --b-pyramid --no-dct-decimate
--deblock 0:-1 --sar 16:11 --crf 18 --progress --no-ssim --no-psnr
--output "D:\Zielvideo.mkv" "D:\Quelle.avs"
```

Genaueres zum Einsatz von CRF steht im allgemeinen x264-Kapitel (Seite 105).

Schneller 1st Pass

Wenn wir Xvid kennen, sind wir es gewohnt, dass der 1st Pass deutlich schneller abläuft als der 2nd Pass. Auch x264 VfW und MeGUI kennen einen Schalter für den schnellen 1st Pass. Erreicht wird diese hohe Geschwindigkeit dadurch, dass einige Optionen, die das Ergebnis des ersten Durchgangs nicht oder nur unwesentlich beeinflussen, abgeschaltet werden. x264 CLI bietet keine Möglichkeit an, den »Turbo-Modus« mit einem einzelnen Schalter zu aktivieren. Wir müssen also selbst entscheiden, welche Optionen wir verändern wollen und welche nicht.

MeGUIs aggressives Herangehen an den Turbo-1st-Pass hat sich bewährt. Hier sind die nötigen Einstellungen:

- `--me dia`
- `--subme 1`
- `--ref 1`
- `--partitions none`
- `--trellis 0`
- kein `--8x8dct`
- kein `--b-rdo`
- kein `--weightb`
- kein `--bime`
- kein `--mixed-refs`
- kein `--no-fast-pskip`.
- Alle anderen Einstellungen bleiben unverändert.

Die Grundregel für einen schnellen 1st Pass lautet: Alle Einstellungen, die die Entscheidung des Encoders für einen bestimmten Frametyp (I, P, B) beeinflussen, sind tabu, denn wenn die Frames in den nachfolgenden Durchgängen anders verteilt werden, sinkt die Genauigkeit der 1st-Pass-Analyse schnell auf ein nicht mehr akzeptables Niveau. Das bedeutet deutliche Qualitätseinbußen. Natürlich ist ein beschleunigter 1st Pass so oder so nicht verlustlos zu haben, da die höhere Geschwindigkeit immer auf Kosten der Genauigkeit geht. Doch vorsichtig konfiguriert, bleiben die Einbußen so klein, dass sie die Qualität nicht sichtbar verschlechtern. Dafür dürfen wir uns über eine deutlich kürzere Encodingzeit freuen.

Ob für hohe oder niedrige Zielgrößen, der Turbo-Modus ist immer sinnvoll; allerdings nur im 1st Pass. Alle weiteren Durchgänge müssen mit allen Optionen in ihrer endgültigen Konfiguration durchgeführt werden.

Referenz der Optionen

Dieser Abschnitt listet mit einer kurzen Erklärung viele Optionen auf, die für x264.exe zur Verfügung stehen. Im Wesentlichen handelt es sich um eine etwas ausführlichere Übersetzung des Befehls `x264.exe --longhelp`. Wir beschränken uns aber auf die für ein normales DVD-Backup wirklich nützlichen Optionen. Eine komplette Referenz bietet ▸ Selurs *man x264*. ▸ [Selu06]

Optionen, die Standardwerte besitzen, müssen wir nur dann angeben, wenn wir einen abweichenden Wert verwenden wollen. Ansonsten arbeitet x264.exe automatisch mit dem Standard. Optionen ohne Standardwert sind nur dann aktiv, wenn wir sie ausdrücklich angeben.

Einige besonders häufig verwendete Option besitzen eine Kurzform, die im Gegensatz zur langen Option mit einem einfachen Strich beginnt (z. B. `--bframes` und `-b`). Zur Übersichtlichkeit habe ich darauf verzichtet, die Kurzformen aufzulisten.

Frametypen

`--keyint`
Standardwert: `250`
Beispiel: `--keyint 250`
Maximales IDR-Frame-Intervall. Siehe auch Seite 99. Als Daumenregel hat sich das Zehnfache der Bildrate eingebürgert. Deshalb können wir den Standardwert meistens problemlos übernehmen.

`--min-keyint`
Standardwert: `25`
Beispiel: `--min-keyint 25`
Mindestabstand, der zwischen zwei IDR-Frames liegen muss. Siehe auch Seite 99. Als Daumenregel hat sich die Bildrate eingebürgert. Deshalb können wir den Standardwert meistens problemlos übernehmen.

`--no-cabac`
CABAC (Context Adaptive Binary Arithmetic Coding) ist eine Neuheit des AVC-Standards. Es handelt sich um eine Kompressionsmethode in der verlustlosen Phase des Encodingprozesses, die gerade bei hohen Datenraten die nötige Bitrate deutlich senken kann. CABAC ist ein zentrales Element von AVC und sollte nicht deaktiviert werden.

`--ref <Frames>`
Werte: Ganze Zahlen zwischen `1` (Standard) und `16`
Beispiel: `--ref 5`
Anzahl der erlaubten Referenzframes. In Xvid und DivX sind P-Frames auf genau ein Referenzbild beschränkt, und zwar auf das vorangehende Bild. x264 erlaubt auch mehrere Referenzframes. Sinnvoll sind etwa 5. Deutlich mehr steigert hauptsächlich die Codierzeit, nicht mehr die Qualität. Die Einstellung wirkt sich auch auf B-Frames aus.

`--no-deblock`
Deaktiviert den Inloop-Deblocking-Filter, was in der Regel nicht sinnvoll ist; auch nicht für ausgesprochene HQ-Backups.

`--deblock <alpha:beta>`
Werte für `alpha` und `beta`: Ganze Zahlen zwischen `-6` und `6`.
Standardwert für beide: `0`
Beispiel: `--deblock 0:-1`
Konfiguriert die Stärke des Inloop-Deblocking-Filter, mit dem wir uns schon ausführlicher im allgemeinen x264-Kapitel auf Seite 103 beschäftigt haben. Hohe Datenraten sollten vorsichtig gefiltert werden (z. B. mit `0:0` oder `0:-1`), niedrige vertragen etwas mehr (etwa `1:2`).

`--bframes <Maximum>`
Werte: Ganze Zahlen von `0` (Standard) bis `16`
Beispiel: `--bframes 3`
Legt fest, wie viele B-Frames maximal direkt hintereinander stehen dürfen. Siehe auch Seite 101 f. Ein Wert von `0` deaktiviert B-Frames vollständig, was nicht empfehlenswert ist. Normalerweise können wir `--bframes 16` setzen, um x264 freie Hand zu lassen. Der verwendete Algorithmus ist gut genug, dass wir dadurch keine Nachteile zu befürchten haben.

`--no-b-adapt`
Deaktiviert die »intelligente« Verteilung von B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Dieser Schalter sollte nicht gesetzt werden.

`--b-bias <Empfindlichkeit>`
Werte: Ganze Zahlen von `-100` bis `100`. Standardwert: `0`
Beispiel: `--b-bias 0`
Steuert x264s Vorliebe für B-Frames. Negative Werte halten den Encoder vom

B-Frame-Einsatz ab, positive ermuntern ihn, öfter B-Frames zu setzen. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Benötigt in der Regel keine Anpassung.

`--b-pyramid`
Erlaubt es, B-Frames als Referenzframes zu verwenden. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Sollte für beste Qualität aktiviert sein.

`--direct <Modus>`
Werte: `spatial` (Standard), `temporal`, `auto`
Beispiel: `--direct auto`
Gibt an, ob zeitliche oder räumliche Informationen zur Komprimierung von B-Frames herangezogen werden. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Die Entscheidung können wir mit `auto` x264 überlassen.

`--weightb`
Aktiviert die gewichtete Bewegungssuche für B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Sollte für beste Qualität aktiviert sein.

`--bime`
Erweitert für B-Frames die Suche nach Referenzen. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Sollte für beste Qualität aktiviert sein.

`--b-rdo`
Aktiviert *Rate Distortion Optimization* für B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 101. Da die Gefahr zum Verlust feinster Details nicht allzu ausgeprägt ist, kann B-RDO nicht nur für hochkomprimierte Encodings sondern auch für HQ-Backups sinnvoll sein. B-RDO benötigt zusätzlich `--subme 6` oder höher.

Encoding-Modus

`--crf <nominaler Quantizer>`
Werte: Zahlen (auch Kommawerte) zwischen `1.0` und `51.0`
Beispiel: `--crf 19.5`
Führt ein 1-Pass-Encoding mit eine Zielquantizer durch (Constant Rate Factor). Kommawerte müssen mit dem Punkt als Dezimaltrenner angegeben werden. Dieser Modus ist interessant, wenn wir auf hohe Qualität setzen und keine exakte Zielgröße anstreben. CRF mit nominalem Quantizer 18 entspricht grob dem von Xvid und DivX bekannten Quant 2.

`--pass <Modus>`
Werte: `1`, `2`, `3`
Beispiel: `--pass 2`
Steuert ein Multipass-Encoding. `1` führt einen 1st Pass durch, in dem die Statistikdatei angelegt wird. `2` führt einen Nth- bzw. 2nd-Pass durch, ohne die Statistikdatei zu aktualisieren. `3` führt ebenfalls einen Nth- bzw. 2nd-Pass durch und aktualisiert dabei die Statistikdatei. Mit der Durchführung eines 2-Pass-Encodings haben wir uns ab Seite 107 schon ausführlicher befasst.

`--bitrate <kbit/s>`
Definiert die Bitrate in kbit/s. Für ein 1-Pass-Encoding (außer `--qp` und `--crf`) stellt der Wert die angestrebte durchschnittliche Datenrate dar. Im 2-Pass-Encoding definiert `--bitrate` die gewünschte Zielgröße.

`--zones <Start>,<Ende>,<Modus>[/<Start>,<Ende>,<Modus>]...`
Definiert Zonen, d. h. Abschnitte im Video, für die individuell das Qualitätsniveau angepasst werden kann. Die einzelnen Parameter einer Zone trennen wir mit Komma (,). Zwischen mehreren Zonen steht ein Schrägstrich (/) als Trennzeichen. Jede Zone besitzt drei Parameter, die alle zwingend angegeben werden müssen.

`Start` steht für die Nummer des ersten Frames der Zone. Das erste Frame des gesamten Videos hat die Nummer 0. `Ende` steht für die Nummer des letzten Frames der Zone. `Modus` definiert den Encodingmodus der Zone. Mit `q=<Quant>` legen wir fest, dass die Zone mit einem konstanten Quantizer (zwischen `0` und `51`) encodiert wird. Mit `b=<Gewicht>` definieren wir eine Zone, die mit einem relativen Gewicht zum restlichen Film behandelt wird. Werte liegen zwischen `0.01` und `100.0` (Punkt als Dezimaltrenner!).

Machen wir uns diese Syntax an einem Beispiel klar. Nehmen wir an, wir möchten für einen Film zwei Zonen definieren:

► Für die ersten 10 000 Frames des Videos ein konstanter Quantizer von 22.
► Die Frames 20 000 bis 25 000 sollen bei der Bitratenverteilung als doppelt so wichtig wie der Rest des Films behandelt werden.

Die dazu passende Kommandozeile sieht folgendermaßen aus:

```
--zones 0,9999,q=22/20000,25000,b=2.0
```

Bewegungssuche und -kompensation

`--partitions <Partitionen>`
Werte: `none`, `all`, `p8x8`, `p4x4`, `b8x8`, `i8x8`, `i4x4`
Standardwert: `p8x8,b8x8,i8x8,i4x4`
Beispiele: `--partitions p8x8,b8x8` oder `--partitions all`
Legt fest, welche Partitionsgrößen für Makroblocks verwendet werden dürfen. Siehe auch Seite 100. Mehrere Werten werden mit einem Komma (`,`) getrennt. `p4x4` benötigt auch `p8x8`. Für `i8x8` muss zusätzlich der Schalter `--8x8dct` gesetzt werden. Da Partitionen maßgeblich die Effizienz von x264 beeinflussen, sollten wir vorzugsweise mit `--partitions all` sämtliche Partitionen aktivieren.

`--me <Modus>`
Werte: `dia`, `hex` (Standard), `umh`, `esa`
Beispiel: `--me umh`
Bestimmt die allgemeine Genauigkeit der Bewegungssuche. In der oben angegebenen Reihenfolge werden die Modi genauer und langsamer. Beim Standard `hex` zu bleiben, ist oft sinnvoll.

`--merange <Radius>`
Werte: Ganze Zahlen zwischen `4` und `1024`. Standardwert: `16`
Beispiel: `--merange 16`
Legt die Größe des Bereichs fest, in dem nach Bewegung gesucht wird. Ist nur relevant für die Modi `--me umh` und `--me esa`. Höhere Werte als `32` sind in der Regel nicht sinnvoll.

`--subme <Qualität>`
Werte: Ganze Zahlen zwischen `1` und `7`. Standardwert: `5`
Definiert die Qualitätsstufe für die Subpixel-Bewegungssuche und die Partitionsentscheidung. `1` ist schnell und ungenau, `7` langsam mit den besten Ergebnissen. B-Frames mit `--b-rdo` benötigen mindestens `--subme 6`, was auch sonst eine gute Wahl darstellt.

`--mixed-refs`
Erlaubt es dem Encoder, Referenzbilder nicht nur für jeden Makroblock, sondern für jede Blockpartition einzeln auszuwählen. Sollte trotz geringerer Geschwindigkeit aktiviert werden.

`--no-chroma-me`
Standardmäßig verwendet x264 neben den Helligkeitsinformationen (Luminanz) auch die Farbinformationen (Chrominanz) zu Bewegungssuche. Mit `--no-chroma-me` können wir das auf die Luminanz beschränken. Da die zusätzliche Chroma-Suche die Geschwindigkeit nicht zu sehr beeinträchtigt, sollte der Schalter nicht gesetzt werden.

`--8x8dct`
Aktiviert die 8×8-DCT-Transformation aus dem AVC High Profile. Sollte i. d. R. aktiviert werden.

`--trellis <Modus>`
Werte: `0` (aus, Standard), `1` (für endgültigen Block), `2` (bei jeder Entscheidung)
Beispiel: `--trellis 1`
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 mit `--trellis 1` für den endgültigen Block eingeschaltet werden. Trellis bei jeder Entscheidung zu aktivieren, bringt keinen nennenswerten Vorteil.

`--no-fast-pskip`
Deaktiviert die beschleunigte Verarbeitung von P-Frames. Wird normalerweise nicht benötigt.

`--no-dct-decimate`
Verhindert, dass im Rahmen der DCT sehr kleine Koeffizienten auf null gerundet werden. Kann in Verbindung mit Trellis sinnvoll sein, wird aber normalerweise nicht benötigt.

`--aq-mode <Modus>`
Werte: `0`, `1`, `2` (Standard)
Beispiel: `--aq-mode 2`
Legt fest, auf welche Weise der Algorithmus zur adaptiven Quantisierung arbeitet, der die Quantisierung von Makroblocks anpasst, je nachdem, wie empfindlich der menschliche Sehapparat für den jeweiligen Bildbereich ist. `0` schaltet AQ ab, `1` nimmt eine Umverteilung von Bits nur innerhalb eines Frames vor, `2` erweitert die Umverteilung auf mehrere Frames. Der Standardwert ist `2` und sollte beibehalten werden.

`--aq-strength <Stärke>`
sinnvolle Werte: ca. `0.5` bis `1.5`, Standardwert `1.0`
Beispiel: `--aq-strength 1.0`
Wählt die Stärke der adaptiven Quantisierung. Auch hier sollte normalerweise der Standardwert `1.0` genutzt werden.

`--cqm <Matrix>`
Werte: `jvt`, `flat` (Standard)
Beispiel: `--cqm jvt`
Wählt eine der beiden Standard-Quantisierungsmatrizen. Siehe auch das allgemeine x264-Kapitel ab Seite 105.

`--cqmfile "<Matrixdatei>"`
Beispiel: `--cqmfile "D:\Matrizen\eqm_avc_hr.cfg"`
Verwendet die benutzerdefinierte Quantisierungsmatrix aus der angegebenen Datei. Siehe auch das allgemeine x264-Kapitel ab Seite 105.

Dateinamen und Videoeigenschaften

`--output "<Zieldatei>"`
Beispiel: `--output "D:\Zielvideo.mp4"`
Wählt Dateinamen und Format der Zieldatei. Mit der Dateiendung .264 wird ein roher AVC-Datenstrom erzeugt. Die Endungen .mkv und .mp4 verpacken das Video in den Matroska- bzw. MP4-Container. AVI ist nicht möglich.

`"<Quelldatei>"`
Beispiel: "D:\Quelle.avs"
Gibt den Namen der Quelldatei an. Es ist keine spezielle Option nötig, um diesen Dateinamen als Quelldatei zu kennzeichnen. Üblicherweise steht diese Angabe ganz am Ende der x264-Kommandozeile.

`--stats "<Statistikdatei>"`
Beispiel: `--stats "D:\Video\x264.stats"`
Legt Pfad und Dateinamen der für ein Nth-Pass-Encoding nötigen Statistikdatei fest. Standardmäßig ist das die Datei *x264_2pass.log* im aktuellen Ordner.

`--fps <Framerate>`
Typische Werte: `23.976`, `25.0`, `29.97`
Beispiel: `--fps 25.0`

Definiert die Bildrate der Zieldatei in Bildern pro Sekunde (fps). Wird bei AviSynth als Quelle automatisch ermittelt.

`--sar <x:y>`
Beispiel: `--sar 16:11`
Definiert das Pixel Aspect Ratio der Zieldatei. Das entspricht dem Setzen des AR-Flags (Seite 53). Für klassische Encodings mit quadratischen Pixeln geben wir immer `--sar 1:1` an oder lassen die Option ganz weg.

Für PARs definiert MPEG-4 AVC zwei Felder (für `x` und `y`), die je einen vorzeichenlosen 16-Bit-Integer enthalten. D. h. der maximale Wert, der darin gespeichert werden kann, ist 65535. Auch wenn sich x264 nicht darüber beschwert, sollten wir deshalb größere Werte niemals eingeben, denn die können nicht in den Videostream geschrieben werden.

Steuerung von x264.exe

`--verbose`
Gibt Statusinformationen zu jedem einzelnen Frame am Bildschirm aus.

`--progress`
Aktiviert die allgemeine Fortschrittsanzeige.

`--no-psnr`
Deaktiviert die Berechnung des PSNR-Wertes, der für ein normales Backup nicht notwendig ist.

`--no-ssim`
Deaktiviert die Berechnung des SSIM-Wertes, der für ein normales Backup nicht notwendig ist.

`--threads <Anzahl>`
Werte: Ganze Zahlen von `1` bis `16`, `auto`
Beispiel: `--threads 3`
Hier stellen wir ein, wie viele Threads x264 zum Encodieren verwenden soll. Für Computer mit nur einer CPU sollten wir die Option weglassen. Wer Hyperthreading, Dual Core oder tatsächlich ein System mit mehreren Prozessoren sein eigen nennt, sollte mit `--threads auto` die Entscheidung x264 überlassen.

`--thread-input`
Verwendet für die Verarbeitung des AviSynth-Skripts einen eigenen Thread. Wie schon `--threads` ist das hauptsächlich für Multicore-Systeme interessant.

A.4.5 DivX 6.8 VfW-Konfiguration

Weitere Literatur: [Mayo04]

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, da 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.

Varianten des DivX-Codecs

Der DivX-Codec ist für seine einfache Handhabung bekannt und hat durch das Zertifizierungsprogramm des Herstellers 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. DivX ist in zwei Versionen zu haben: ► [Doom05]

► Als **kostenpflichtige Vollversion,** die man 15 Tage lang testen kann. Anschließend muss der Codec gekauft werden, um den vollen Funktionsumfang zu behalten.
► Als kostenloser **Community Codec**. In dieser Version sind nur die zertifizierten Profile nutzbar, nicht der uneingeschränkte Modus. Außerdem wird der Codec ausschließlich über einen Qualitätsschieber konfiguriert. Lediglich die *Psychovisual Enhancements* können einzeln verändert werden.

Das Downloadpaket ist für beide Versionen das selbe. DivX läuft zuerst so lange im Community-Modus, bis wir die 15 Tage Testfrist der Vollversion ausdrücklich starten. Nach Ablauf der Frist wird der Codec automatisch auf den Community-Umfang zurückgestuft. Im Folgenden beschäftigen wir uns mit der Konfiguration der Vollversionen, denn ich gehe davon aus, dass es keiner weiteren Erklärung bedarf, ein Profil auszuwählen und einen Qualitätsschieber in Position zu bringen.

Wahl der Encodingmethode

Wir betrachten zuerst die DivX-Optionen, die nur für ein 2-Pass-Encoding Bedeutung haben; anschließend dann die speziellen Optionen für 1-Pass mit konstantem Quantizer (CQ). Zum Schluss folgen alle Einstellungen, die unabhängig von der ►Encodingmethode immer verfügbar sind.

►A.4.1

2-Pass bzw. N-Pass

Wir öffnen die DivX-Konfiguration und befinden uns im Register **Main**. Unter **Rate Control Mode** (Abb. A-36) wählen wir den passenden Modus für der ersten Encodingdurchgang. DivX bietet zwei Möglichkeiten, nämlich das traditionelle **Multipass, 1st pass** und das in DivX 6.4 eingeführte **Multipass, 1st pass (fast)**. Letzteres arbeitet nach dem selben Prinzip wie Xvids 1st Pass oder der Turbo-1st-Pass 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.

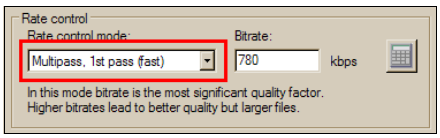


Abb. A-36: Einstellung für 1st Pass.

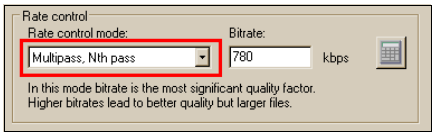


Abb. A-37: Einstellung für Nth Pass.

Die Bitrate im Feld rechts sollte das Encoding-Frontend automatisch ausfüllen. Sie bezieht sich auf die reine Videospur, nicht auf die gesamte Dateigröße.

Jetzt stellen wir wie weiter unten beschrieben alle globalen Optionen ein, die für jeden Pass gelten. Damit ist die 1st-Pass-Konfiguration abgeschlossen.

Für den 2nd Pass und alle weiteren Durchgänge öffnen wir wieder die DivX-Konfiguration auf der **Main**-Seite. Unter **Rate Control Mode** stellen wir jetzt **Multipass, Nth pass** ein (Abb. A-37). Alle globalen Optionen konfigurieren wir genauso wie im 1st Pass.

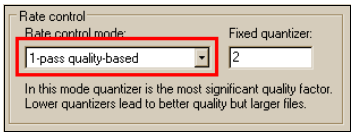


Abb. A-38: Einstellung für 1-Pass CQ 2.

Für alle weiteren Passes ist das Vorgehen genau das gleiche wie für den 2nd Pass. Da kommt auch die Frage auf, wie viele Durchgänge denn sinnvoll sind. Zwei Stück reichen meisten vollkommen aus. Der dritte Durchgang bringt nur noch minimale Qualitätssteigerungen und dürften höchstens bei hoch komprimierten 1-CD-Encodings in Ausnahmefällen den Aufwand wert sein. Vier oder mehr Passes sind auf jeden Fall Zeitverschwendung.

1-Pass Constant Quantizer (CQ)

Auch ein CQ-Encoding konfigurieren wir über den **Rate Control Mode** auf der **Main**-Seite des DivX-Dialogs. Die passende Einstellung ist **1-pass quality-based** (Abb. A-38). Im Feld daneben geben wir den Quantizer ein, mit dem encodiert werden soll. Was wir schon im allgemeinen Xvid-Kapitel (Seite 67) besprochen haben, gilt auch hier. Quant **2** ist die richtige Einstellung für kompromisslose HQ-Encodings, **3** taugt als Kompromiss zwischen Qualität und Größe, alles ab **4** sollten wir wegen der niedrigen Qualität meiden. Anschließend konfigurieren wir den Rest der DivX-Optionen.

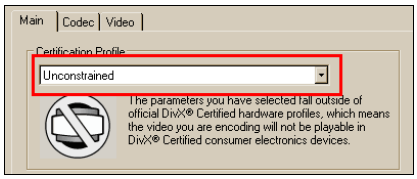


Abb. A-39: DivX-Profilauswahl.

Globale Optionen

In diesem Abschnitt betrachten wir alle Einstellungen von DivX, die unabhängig von der Encodingmethode sind. Das gilt zumindest für die meisten davon. Wenn nicht, ist das extra erwähnt. Wir beginnen auf der **Main**-Seite des DivX-Dialogs.

Um volle Kontrolle über alle Optionen zu erhalten, müssen wir unter **Certification Profile** auf **Unconstrained** umschalten (Abb. A-39) 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.

Damit können wir ins Register Codec weiterklicken (Abb. A-40). Bei **Codec Performance** müssen wir uns entscheiden, ob Geschwindigkeit oder Qualität wichtiger ist. **Balanced** ist vergleichsweise flott und bringt ordentliche Qualität. Der sinnvollste Kompromiss zwischen Geschwindigkeit und Qualität dürfte die nächsthöhere Stufe **Better** sein. Beim Multipass-Encoding können wir Zeit sparen, ohne dass die Qualität allzu sehr leiden würde, indem wir den ersten und alle »mittleren« Passes (falls vorhanden) mit dem schnellen **Balanced** durchführen und für den letzten Durchgang das äußerst langsame, aber qualitativ beste **Insane** verwenden. Für 1-Pass-CQ-Encodings sind die beiden höchsten Stufen **Extreme** und **Insane** nicht sinnvoll, weil sie in diesem Modus nie B-Frames verwenden, auch wenn die eigentlich aktiviert sind. Das führt zu einem ineffizienten Encoding mit zu großer Zieldatei.

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 Abb. A-41 konfiguriert die B-Frames. **Off** dürfte sich selbst erklären ;-), **Adaptive Single Consecutive** erlaubt nur einzelne B-Frames, nicht mehrere hintereinander. Maximal drei aufeinander folgende B-Frames sind mit **Adaptive Multiple Consecutive** möglich. B-Frames sollten wir auf jeden Fall aktivieren, und das dürfen ruhig die Multiple Consecutive sein.

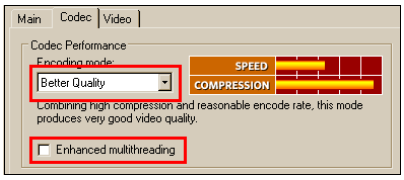


Abb. A-40: DivX-Performance-Optionen.

Quarter-pixel search (QPel) und **Global motion compensation** (GMC) haben wir schon bei der Xvid-Konfiguration auf Seite 92 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. Im Abschnitt **Video Resolution** können wir alle Optionen bis auf eine ignorieren. Seit Version 6.5 erlaubt DivX die Angabe eines PAR (Abb. A-42), das 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 Pixels**. Die nicht-quadratischen Einträge entsprechen den Werten gemäß MPEG-4 Visual (vgl. Tab. A-4, Seite 47). Wichtig ist es, dass wir bei **Format in** und **Format out** jeweils den gleichen Wert einstellen.

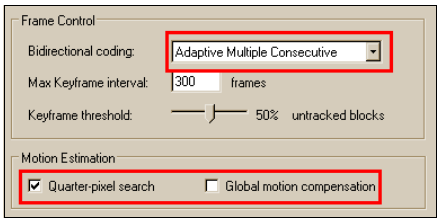


Abb. A-41: DivX-Optionen für Frametypen und Bewegungssuche.

Bei **Image Processing** (Abb. A-43) 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.

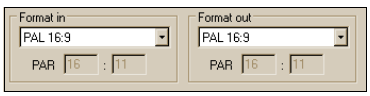


Abb. A-42: PAR-Auswahl.

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.

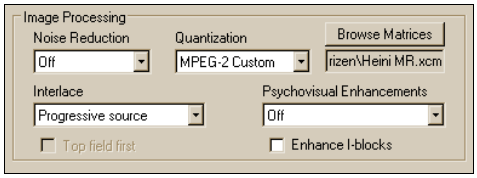


Abb. A-43: DivXs Matrix- und PVE-Auswahl.

Seit DivX 6.8 werden mit **MPEG-2 Custom** auch benutzerdefinierte Matrizen unterstützt. Für diese gilt alles das, was wir schon im allgemeinen Xvid-Kapitel (Seite 71 ff.) darüber gelernt haben. Sogar das Dateiformat der Matrixdateien ist gleich, so dass alle Xvid-Matrizen auch mit DivX verwendet werden können. Wir brauchen nur über den Button **Browse Matrices** die entsprechende Datei laden.

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 I-blocks** ist ein Feature, das im Rahmen der PVE die Qualität rund um I-Frames verbessern soll.

Damit ist die Konfiguration fast beendet. Nur ein Klick unten links auf **Advanced** bleibt uns noch (Abb. A-44). 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. :-)

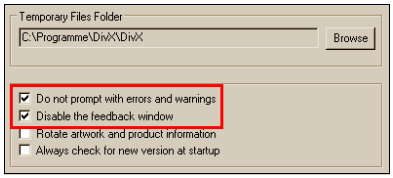


Abb. A-44: DivX, weitere Optionen.

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.

In der Randspalte finden wir häufig Verweise zum gerade passenden Kapitel im Hintergrundwissen. So können wir uns die eher 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 PgcDemux 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.

Welche Tools wir tatsächlich benötigen, und wie wir die am bequemsten beschaffen, hängt stark vom Encoding-Frontend ab. Deswegen sollten wir uns zuerst zwischen Gordian Knot und StaxRip entscheiden, wobei ein Blick in die Vergleichstabelle auf Seite 160 sicher hilfreich ist. Ich empfehle jedenfalls StaxRip.

Encoding-Frontends

► StaxRip
Moderne Kommandozentrale fürs Encoding, die im Gegensatz zum altehrwürdigen Gordian Knot auch x264.exe, AAC und MP4 unterstützt. StaxRip enthält eine Downloadfunktion, mit der wir alle benötigten Tools herunterladen können.
`http://www.planetdvb.net/staxrip/`

► Gordian Knot
Die Kommandozentrale fürs Encoding, deren Entwicklung inzwischen eingestellt ist. Das GKnot RipPack enthält neben Gordian Knot selbst auch VirtualDubMod, BeSweet, DGMPGDec, 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, AutoGK, AutoMKV

Vorarbeiten

► PgcDemux
Unser Ripper, der den Hauptfilm der DVD auf die Festplatte kopiert.
`http://download.videohelp.com/jsoto/dvdtools.htm`

► ChapterXtractor
Schreibt die Kapitelliste 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*.

► BeLight
Grafische Oberfläche für BeSweet.
http://coreforge.org/projects/belight/

► VSRip
Tool, um die Vobsub-Untertitel der DVD aus den VOBs herauszuziehen.
http://sourceforge.net/project/showfiles.php?group_id=82303&package_id=84442

► 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: Decomb, Deen, FillMargins, FluxSmooth, SimpleResize, TomsMoComp usw. Oft nicht ganz aktuell, aber ein guter Ort für den Einstieg bei der Pluginsuche.
http://www.avisynth.org/warpenterprises

► Xvid Video-Codec
Den aktuellsten Xvid 1.2 incl. VAQ-Patch bekommen wir von wizboy11. Binaries gibt es ebenfalls von Koepi. Der dortige Xvid 1.2 ist jedoch älter als wizboy11s Version und deshalb nicht zu empfehlen. Den Quellcode können wir auf xvid.org herunterladen.
wizboy11: http://forum.doom9.org/showthread.php?p=1138522#post1138522
Koepi: http://koepi.info/

► XvidEncraw
Xvid-Kommandozeilenencoder von squid_80. Zum aktuellen Stand den ►Entwicklungsthread im Doom9-Forum verfolgen. ►[squi06]
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
Bob0rs Build von x264 incl. AviSynth- und MP4-Unterstützung. Den Quellcode können wir uns von der x264-Projektseite auf `videolan.org` besorgen.
`http://x264.nl/`

► VirtualDubMod
Video-Editing-Tool. Wird von Gordian Knot und teilweise von StaxRip für das eigentliche Encoding verwendet.
`http://sourceforge.net/projects/virtualdubmod`

Muxing und Splitting

► MKVToolnix
Toolpaket rund um Matroska. Nötig fürs Matroska-Muxing.
`http://www.bunkus.org/videotools/mkvtoolnix/index.html`

► AVI-Mux GUI
Großartiges Muxing-Tool für AVI. Unterstützt auch Matroska.
`http://www.alexander-noe.com/video/amg`

► MP4Box
Muxing-Tool für MP4, entwickelt im Rahmen des GPAC-Projekts. Immer das gerade aktuellste Archiv herunterladen. Wer sich mit MP4Box näher beschäftigen will, unbedingt diesen Doom9-Thread lesen: ►*GPAC's MP4Box Information, Requests and Discussions*.
► [SeeM05]
`http://kurtnoise.free.fr/index.php?dir=mp4tools/`

B.1.2 DVD-Ripping

Bevor wir einen DVD-Film weiterverarbeiten können, müssen wir ihn erst einmal von der Disc auf die Festplatte schaufeln. Das darf nie mit einfachem Kopieren geschehen, sondern benötigt ein spezielles Programm. Denn nur so können wir sicherstellen, dass ausschließlich der Hauptfilm in einem zusammenhängenden Stück auf der Platte landet.

(Un-)Recht und Gesetz

Viele DVDs besitzen einen digitalen Kopierschutz namens CSS, d. h. die Videodaten liegen verschlüsselt auf der Scheibe. Dazu kommt oft noch ein analoges Gegenstück namens Macrovision und diverse digitale Verkrüppelungsmaßnahmen. Zwar lassen sich alle diese Hindernisse lächerlich einfach umgehen, allerdings ist das (nicht nur) in Deutschland illegal. Wir werden deshalb das Programm PgcDemux verwenden, das im Gegensatz zu anderen Rippern nur mit ungeschützten DVDs umgehen kann.

Die Alternativen kann ich leider nicht erwähnen, ohne eine Abmahnung zu riskieren. Klar ist das Zensur auf hässlichste Weise, abgesegnet vom Gesetzgeber. Es stimmt auch, dass wir eigentlich ein Recht auf die Privatkopie haben und dafür sogar eine Abgabe auf jeden gekauften Rohling und Brenner bezahlen. Und nein, wir dürfen unser gutes Recht dank aktueller Kopierschutzgesetze oft nicht ausüben.

Konkret bedeutet das, dass sich PgcDemux nicht unbedingt als sinnvollster und komfortabelster DVD-Ripper anbietet. Es ist jedoch eine der wenigen Möglichkeiten, das Thema Ripping überhaupt zu besprechen, ohne gleich von wegelagernden Anwaltsbanditen in den Bankrott gemahnt zu werden.

Wichtige Grundlagen

Eine ▸Video-DVD 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. ▸A.2.2

► **VOB** ist das Containerformat der DVD und enthält Video, Audio und Untertitel. Da VOBs maximal 1 GB groß sein dürfen, erstreckt sich der Hauptfilm meistens

über mehrere davon. Die Unterteilung in VOBs hat nichts mit der Kapitelunterteilung des Films zu tun!

► **IFO**-Dateien sind Steuer- und Informationsdateien, die uns das Ripping und den Umgang mit Untertiteln erleichtern.

Wie in Abb. B-1 am Ausschnitt aus einer DVD-Dateiliste zu sehen ist, sind die meisten Dateien 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 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).

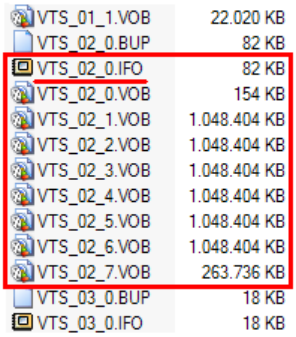


Abb. B-1: Dateien im `VIDEO_TS`-Ordner einer DVD.

Der Hauptfilm liegt in einer der VOB-Gruppen. Fürs Ripping müssen wir herausfinden, in welcher. Zum Glück ist das kein Problem, da der Hauptfilm der größte Datenbatzen auf der Scheibe ist und wir deshalb davon ausgehen dürfen, dass er in der VOB-Gruppe steckt, die den meisten Platz belegt. In unserem abgebildeten Beispielfall ist das die Gruppe mit der `02` in der Mitte. Die dazugehörige IFO-Datei `vts_02_0.ifo` kopieren wir ganz normal per Dateimanager auf die Festplatte, dorthin wo auch der gerippte Film liegen soll, und benennen sie in `vts_01_0.ifo` um, da PgcDemux immer VOBs mit der `01` erzeugt und die mittlere Nummer in einer Gruppe ja gleich sein muss.

Ripping des Films

Um den eigentlichen Film auf die Platte zu ziehen, starten wir PgcDemux. Ganz oben in Abb. B-2 bei **Input IFO** wählen wir über den **Browse**-Button die IFO-Datei des Hauptfilms aus, und zwar aus dem `VIDEO_TS`-Ordner der DVD. Unter **Output Folder** geben wir den Speicherort für den Film auf der Festplatte an. Für das komplette Encoding sollten auf dem Laufwerk mindestens 10 GB Platz frei sein. Außerdem muss die Arbeitsweise von PgcDemux auf **by PGC** stehen, da die DVD im Wesentlichen aus einer Reihe solcher ***Program Chains*** besteht, von denen eine den Hauptfilm enthält.

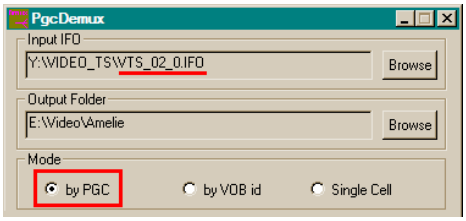


Abb. B-2: PgcDemux: Dateien und Modus.

Die restlichen Optionen von PgcDemux stellen wir so ein wie in Abb. B-3. Alle

Demux-Optionen sind deaktiviert, da wir das Demuxing später erledigen. Dafür ist der Haken bei **Create a PGC VOB** gesetzt, um VOBs auf der Platte zu erzeugen. **One file per VID** darf nicht angehakt sein. Auf der rechten Seite unter **Domain** stellen wir auf **Titles** und sehen darunter alle vorhandenen PGCs. Wenn die Combobox mehrere Einträge enthält, wählen wir den aus, dessen Länge dem Hauptfilm entspricht.

Bei **Angle** wählen wir den Blickwinkel des Videos, falls mehrere vorhanden sind. Eine DVD-Videospur kann in mehreren Versionen vorliegen. Bei *Star Wars* zum Beispiel wird das verwendet, um den Lauftext am Anfang in verschiedenen Sprachen einblenden zu können. Auch DVDs, die Kinoversion und Director's Cut eines Films enthalten, sind oft über Blickwinkel (Seamless Branching) realisiert. Welcher **Angle** der richtige ist, verrät uns der Software-DVD-Player beim Abspielen der DVD.

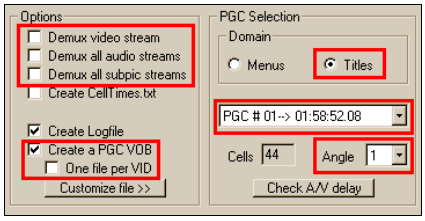


Abb. B-3: Optionen von PgcDemux.

Mit einem Klick auf **Process!** starten wir das Ripping und warten, bis PgcDemux 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 mehreren VOBs auf der Platte.

Ripping der Kapitelliste

Leider kann PgcDemux die ►Kapitelinfos der DVD nicht auslesen, weshalb das ChapterXtractor übernehmen muss. Wir starten das Programm und laden über den Button **Open IFO** unten links die IFO-Datei des Hauptfilms entweder aus aus dem `VIDEO_TS`-Ordner der DVD oder von der Platte. Im Beispiel von oben wäre das also die bekannte `vts_02_0.ifo`.

►A.1.4

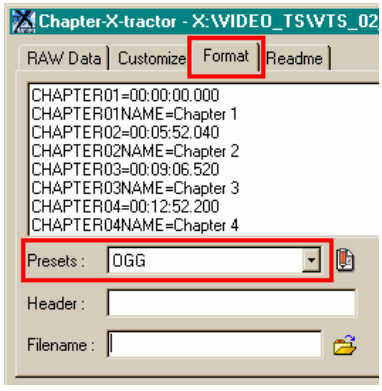


Abb. B-4: Formatwahl in ChapterXtractor.

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 meist nur Sekundenbruchteile kurzes und nutzloses Kapitel, das ChapterXtractor nicht übernimmt. Mit der Option **Last chapter bug fix** können wir dieses Verhalten auch ändern. Im Register **Format** (Abb. B-4) wählen wir dann unter **Presets** das Format, in dem die Ka-

pitelinformationen 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.

Weitere Literatur: [SchaOD]

B.1.3 Indexieren der VOBs mit DGIndex

DGIndex aus Donald Grafts DGMPGDec-Paket ist die verbesserte und erweiterte Variante des ursprünglichen DVD2AVI von jackei und dient dazu, MPEG-1- und MPEG-2-Quellmaterial incl. Audiospuren für die Weiterverarbeitung vorzubereiten. Wir benötigen DGIndex, um

► die Audiospuren aus den VOBs herauszuholen (demuxen) und in separate Dateien zu speichern,
► einen Index der Videospur (D2V-Datei) anzulegen. Das ist eine Art Inhaltsverzeichnis, das später von AviSynth beim Decoding des Quellvideos benötigt wird.

Wenn wir StaxRip als Encoding-Frontend nutzen, müssen wir DGIndex nicht zwingend manuell verwenden, denn StaxRip kann das auch automatisch.

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.

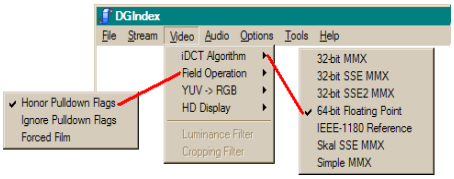


Abb. B-5: Einstellungen im DGIndex-Video-Menü.

Die Einstellungen im **Video**-Menü sollten wie in Abb. B-5 aussehen. Den **iDCT Algorithm** stellen wir auf **64-bit Floating Point** und unter **Field Operation** wählen wir **Honor Pulldown Flags**. Im **Audio**-Menü regeln wir das Herausziehen der Audiospuren aus den VOBs (Abb. B-6). **Output Method** bestimmt, auf welche Weise die Audiospuren extrahiert werden. **Disable** extrahiert gar nichts, **Demux Tracks** extrahiert nur die unter **Select Track(s)** angege-

benen 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.

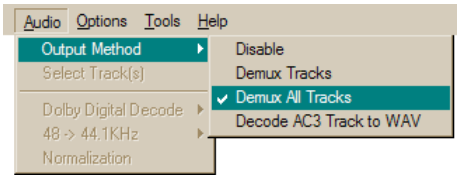


Abb. B-6: Demuxing sämtlicher Audiospuren.

Über **F4** legen wir nun eine D2V-Projektdatei an. Solange DGIndex arbeitet, öffnet sich das Statusfenster (Abb. B-7). Sobald dort ganz unten **FINISH** erscheint, ist der Speichervorgang abgeschlossen und wir haben die extrahierten Audiodateien und die D2V-Datei auf der Platte. Die Werte bei **Aspect Ratio** und **Video Type** merken wir uns für später.

Die letzte wichtige Angabe ist der **Frame Type**. Steht dort **Progressive**, brauchen wir uns keine weiteren Gedanken machen, da das Video progressiv codiert ist, d. h. jedes Einzelbild besteht tatsächlich aus einem einzelnen Vollbild. Die Angabe **Interlaced** deutet dagegen auf möglicherweise interlaced Video hin, d. h. jedes Einzelbild besteht aus zwei zeitversetzten Halbbildern, was wir beim Encoding unbedingt berücksichtigen müssen.

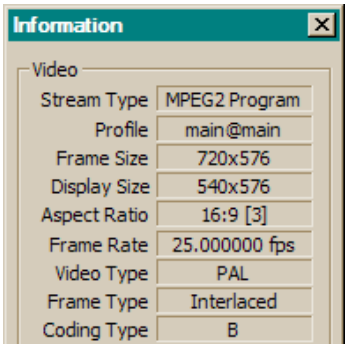


Abb. B-7: DGIndex-Statusfenster.

DGIndex zeigt allerdings nur an, wie das Quellvideo in den VOBs markiert ist, und es ist durchaus üblich, progressives Material als interlaced auszuweisen. Die Angabe **Interlaced** ist also nur ein Hinweis auf mögliches Interlacing, jedoch keine Garantie. Um sicherzugehen, sollten wir uns Szenen mit viel horizontaler Bewegung suchen und prüfen, ob dort Kammartefakte wie im Screenshot B-8 (besonders der Ball) 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 dagegen ein Inverse-Telecine-Prozess in Frage. Interlacing taucht in einer unübersehbaren Vielzahl von Varianten auf. Ich bin auf dem Gebiet alles andere als ein Experte. Erwartet also keine vollständigen und immer komplett richtigen Infos dazu.

Abb. B-8: Bild mit Interlacing-Artefakten. Besonders deutlich am roten Ball.

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 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 kein Weg vorbei.

Noch mächtiger – aber nur sinnvoll, wenn wir mit der Kommandozeile vertraut sind – ist die Variante, BeSweet direkt und ohne grafische Oberfläche zu benutzen. Eine ausführliche Referenz der möglichen Parameter bietet die ►BeSweet-Kommandozeilenreferenz. ►[BrJo07b]

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 im Abschnitt »beta«.
`http://besweet.notrace.dk/`

► Lame_enc.dll
Falls wir MP3s erzeugen wollen. Beste Anlaufstelle dafür ist RareWares.org. Ich bevorzuge die jeweils aktuelle stabile oder Betaversion. Von Alphas würde ich die Finger lassen.
`http://www.rarewares.org/mp3-lame-bundle.php`

► Libvorbis.dll
Falls wir nach Vorbis transcodieren wollen. Auf RareWares.org erhalten wir das Paket *libvorbis.dll using aoTuV* in der jeweils aktuellsten Version.
`http://www.rarewares.org/ogg-libraries.php`

► Nero
AAC-Encoding mit Nero bedeutet nicht mehr, das komplette Nero-Paket besitzen zu müssen. BeSweet nutzt den kostenlosen Nero-Kommandozeilenencoder.
`http://www.nero.com/deu/nero-aac-codec.html`

► Winamp
AAC-Encoding mit Winamp benötigt zum einen die `WA_aac.dll`, die man im BeLight-Downloadordner in der aktuellen Version findet. Außerdem benötigen wir Winamp.
`http://kurtnoise.free.fr/index.php?dir=BeLight`
`http://www.winamp.com/player/features`

► Aften
Falls wir AC-3 encodieren wollen. Ausführbare Pakete erhalten wir auf Sourceforge.
`http://win32builds.sourceforge.net/aften/`

► Bsn.dll
Die `bsn.dll` ist notwendig fürs AAC- und AC-3-Encoding. Die aktuelle Version erhalten wir wieder aus dem BeLight-Downloadordner.
`http://kurtnoise.free.fr/index.php?dir=BeLight`

► BeLight
Als grafische Oberfläche für BeSweet. Die aktuelle Version finden wir auf CoreForge.org zum Download.
`http://coreforge.org/projects/belight/`

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 Winamp, wenn wir dessen AAC-Encoder verwenden wollen.

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

- **bsn.dll** aus dem Bsn-Archiv,
- **lame_enc.dll** aus dem LAME-Archiv,
- **libvorbis.dll** aus dem Vorbis-Download,
- **aften.exe** aus dem Aften-Archiv, am besten mit den passenden Optimierungen für unseren Prozessor,
- **neroAacEnc.exe** aus dem Nero-Archiv. Wer einen SSE2-fähigen Prozessor hat (was auf alle einigermaßen aktuellen CPUs zutrifft), kann auch die optimierte Version `neroAacEnc_SSE2.exe` nehmen, muss die allerdings in `neroAacEnc.exe` umbenennen, damit sie BeSweet erkennt.

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. Wer auf die Idee kommen sollte, als Tonspur verlustloses FLAC einzusetzen, der kann den Flake-Encoder benutzen. Die Installation läuft genauso wie bei Aften; auch der Downloadseite stammt vom selben Sourceforge-Projekt: `http://win32builds.sourceforge.net/flake/`

B.2.2 Audio decodieren und bearbeiten

Weitere Literatur: [Flor01]

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

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.

Nun betrachten wir die linke Seite des BeLight-Fensters. Der erste Eintrag (Abb. B-9) gehört zu Azid. Azid ist dafür zuständig, die ►AC-3-Tonspur zu decodieren und eventuell 6-Kanal auf 2-Kanal (Stereo) umzurechnen.

►A.2.1

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 AC-3 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.

Abb. B-9: Azids Dynamikkompression.

Abb. B-10: Boost als Ersatz für Dynamic Compression.

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.

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 AC-3 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** (Abb. B-10) 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 Störge-

räusche 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.

Wenn's denn sein soll, ►LigH schlägt 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. ►[LigH03]

Mit dem **SSRC**-Abschnitt weiter oben im Fenster 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.

Damit weiter in den **OTA**-Abschnitt (Abb. B-11). 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.

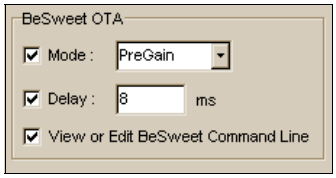


Abb. B-11: Normalisierung und Delay.

Unter **Mode** sollten wir **PreGain** wählen, um auf der sicheren Seite zu sein. Im PreGain-Verfahren durchsucht BeSweet in einem ersten Durchgang die komplette Audiospur nach dem höchsten Pegel und hebt anhand dessen die Spur auf 100 % Lautstärke an. Am sinnvollsten ist dieses Verfahren deshalb, weil bei den anderen Modi unter Umständen beim Abspielen ein Teil der Lautstärkeanhebung wieder verloren gehen kann.

HybridGain schlägt gleich am Anfang des Transcodings je nach Quelldatei einen festen Wert auf die Lautstärke auf und speichert die Differenz zu 100 % in einem PostGain-Tag in der Datei. **PostGain** arbeitet rein mit diesem Tag. Damit entfällt der Pegelsuchlauf, was das Encoding um ein paar Minuten verkürzt. Allerdings muss der Audiodecoder später beim Abspielen des Films den PostGain-Tag auslesen und die Lautstärke entsprechend erhöhen. Vorbis-Decoder können das normalerweise, AAC-Decoder nicht. Deswegen sind wir beim AAC-Encoding gezwungen, **PreGain** zu wählen. Doch auch für die anderen Formate ist das die beste Lösung, da die um ein paar Minuten längere Encodingzeit nicht ins Gewicht fällt und wir sämtlichen Abspielproblemen genauso aus dem Weg gehen wie den eher theoretischen Überlegungen in Sachen geringerer Qualität.

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 Abb. B-12. 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 angeben – einschließlich evtl. vorhandenem negativen Vorzeichen.

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 man es zum Schluss beim Muxen erfahrungsgemäß gerne vergisst.

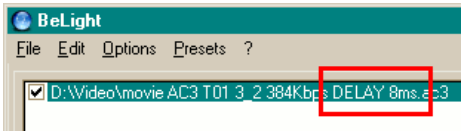


Abb. B-12: Delay-Angabe im Dateinamen.

Damit kommen wir zu den **Advanced Settings** (Abb. B-13). Für den Downmix auf Stereo sollten die Optionen in der Regel so aussehen wie im Screenshot. **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 sechs Kanäle bei, entfernen wir sowohl bei **LFE to LR Channels** als auch bei **Output Mode** die Haken.

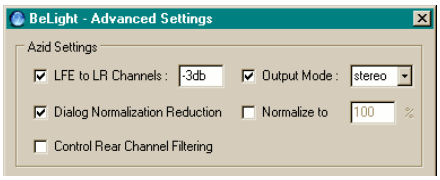


Abb. B-13: Erweiterte Azid-Einstellungen.

Als letztes aktivieren wir die **Dialog Normalization Reduction**. 5.1-AC-3-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. Wirklich wichtig ist DNR nur dann, wenn verschiedene 5.1-AC-3s mit unterschiedlichen Dialogleveln in eine einzelne Zieldatei transcodiert werden sollen, um die unterschiedlichen Level anzugleichen. Beim Transcoding nur einer AC-3 (oder mehrerer AC-3s mit gleichem Dialoglevel) wirkt sich die Funktion nicht aus. Außerdem sind Stereo-AC-3s nicht betroffen, da die keinen Center-Kanal enthalten.

Wenn wir trotz der Nachteile (vgl. Seite 34) 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.

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 (Abb. B-14) stellen wir bei **Encoding Engine** auf **Nero-Digital - (CLI)** und wählen links daneben die passende Anzahl an **Output Channels**. Dann kümmern wir uns weiter unten um die Details der Konfiguration. ►CBR-Encoding (**Bitrate**) ist wegen der geringeren Qualität wenig empfehlenswert. Deswegen halten wir uns an den VBR-Modus (**Quality**). Mit dem Schieberegler darunter stellen wir das Qualitätsninveau ein, dessen interessanter Bereich etwa zwischen **0.16** und **0.30** liegt. Mit Tab. B-1 können wir grob abschätzen, welche Bitrate wir abhängig von der Kanalanzahl erwarten dürfen. In diesem Bereich zu bleiben, ist eine gute Idee, denn darunter sinkt die Qualität doch schnell spürbar ab und darüber rechtfertigt die größere Datei den geringen Qualitätszuwachs nicht mehr wirklich.

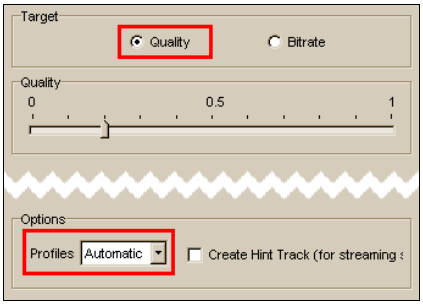


Abb. B-14: Nero-AAC-Setup.

►A.1.3

Tab. B-1: Nero-Bitraten in kbit/s.

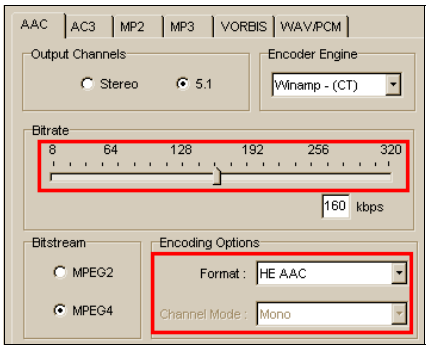
Qualität	6-Kanal	Stereo
0.16	120	35
0.30	275	80

►A.2.5 Das gewünschte ►AAC-Profil wählen wir unter **Profiles**: Low Complexity (**LC**), High Efficiency (**HE**) oder High Efficiency mit Parametric Stereo (**HEv2**). Letzteres ist nur für Stereo-Ton verfügbar und für extrem niedrige Qualitätseinstellungen unter 0.15 gedacht. Ansonsten sollten wir besser die Finger von HEv2 lassen. Mit der Einstellung **Automatic** wählt der Nero-Encoder das Profil automatisch je nach Qualitätsniveau. Die Grenzwerte (bis 0.15 HEv2 falls Stereo, dann bis 0.30 HE, darüber LC) sind auch durchaus sinnvoll gewählt. Ich würde lediglich LC etwas mehr zutrauen. Ab ca. Qualitätsstufe 0.25 sollten wir es problemlos verwenden können.

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.

Abb. B-15: Winamp-AAC-Setup.



Die passende Einstellung der **Encoder Engine** im **AAC**-Register ist natürlich **Winamp** (Abb. B-15). 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 Winamp nur für Stereo zu. Mit 160 kbit/s sind wir am besten bedient. Für Stereo dürften es deutlich weniger sein. Zwischen **64** und **80** sollte auch im LC-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 Platz sparen, den wir besser für die Videospur verwenden. Für höhere Ste-

reo-Bitraten schlägt BeLight teilweise **HE AAC High** vor, was wir dann übernehmen.

Die Einstellungen zum **Channel Mode** gelten nur für Stereoton. 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 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 das gleiche wie für Stereo-Vorbis, auch wenn wir hier nur die Stereo-Variante betrachten, da der 6-Kanal-Modus AC-3-ähnliche Bitraten benötigt, um ordentliche Qualität zu produzieren. ►A.2.5

Zuerst wählen wir wie in Abb. B-16 **Stereo**-Output, und dann bei **Bitrate Management** die Einstellung **Quality**. Vorbis ist auf echtes ►VBR 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. ►A.1.3

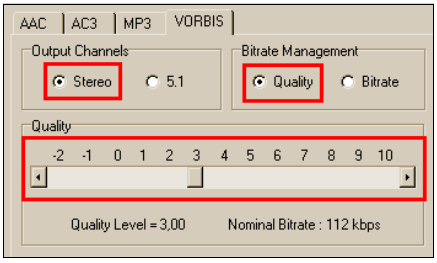


Abb. B-16: Vorbis-Setup.

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.

Weitere Literatur: [HAud08]

B.2.5 MP3-Encoding mit LAME

►A.2.1 ►MP3 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.

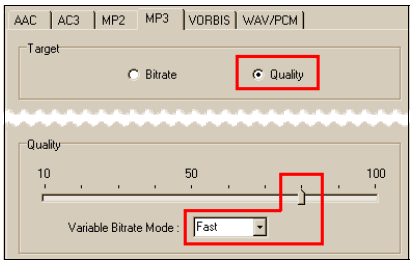


Abb. B-17: LAMEs MP3-Setup.

Da MP3 nur Stereo unterstützt, entfällt die Auswahl der Zielkanäle. Wir brauchen nur wie in Abb. B-17 im **MP3**-Register unter **Target** auf **Quality** zu klicken, um den VBR-Modus einzustellen. Im **Quality**-Abschnitt stellen wir die gewünschte Qualitätsstufe ein. Seit LAME 3.97 gehören die weit bekannten Presets offiziell der Vergangenheit an. Zum Vergleich, welche Qualitätsstufe für welches Preset steht und welche Bitraten wir erwarten sollten, dient die folgende Tab. B-2.

Tab. B-2: Einige Qualitätsstufen von LAME.

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

Mit Qualität **80** 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 und inzwischen die empfohlene Einstellung ist. Es gibt auch tatsächlich keinen Grund, auf die deutlich höhere Geschwindigkeit zu verzichten.

►A.2.3 Damit kommen wir zum Nachteil von VBR-MP3. Dieser Nachteil gilt ausschließlich für den ►AVI-Container, der mit VBR-Audio (nicht nur MP3) Probleme haben kann. Genau genommen kommt die Einschränkung eigentlich vom ACM-Audioframework, das eng mit AVI verknüpft ist und wegen seines Alters lediglich CBR versteht. Das führt dazu, dass alle Programme, die rein auf ACM setzen, mit VBR in AVI nicht zurechtkommen. Ausnahmen sind im Wesentlichen nur VirtualDubMod und AVI-Mux GUI, die das von Nando entwickelte Verfahren nutzen, um die ACM-Beschränkung zu umgehen. Auch das originale VirtualDub unterstützt seit Version

1.8 VBR-MP3.

Das Nando-Verfahren erzeugt AVI-Dateien, die sich in der Regel problemlos abspielen lassen. Paranoide Gemüter oder Hardwareplayer-Liebhaber sollten sich trotzdem überlegen, bei CBR zu bleiben. Das kostet zwar etwas Qualität, schließt aber sämtliche Wiedergabeprobleme aus.

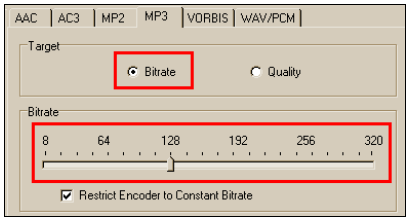


Abb. B-18: LAMEs CBR-Setup.

CBR-MP3 erzeugen wir mit den Einstellungen aus Abb. B-18. 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 160 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 AC-3-Encoding

Fürs ►AC-3-Encoding verwenden wir immer Aften, denn der seit jeher in BeSweet enthaltene ac3enc ist für qualitativ sinnvolle Encodings nicht geeignet. Aften zu konfigurieren funktioniert per BeLight sehr einfach. Zuerst wählen wir im **AC3**-Register **Aften** als **Engine** (vgl. Abb. B-19).

►A.2.5

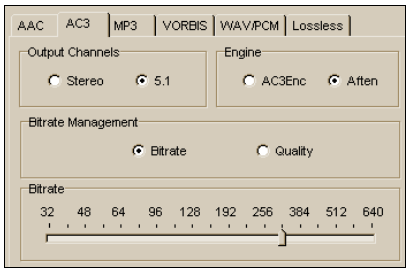


Abb. B-19: AC-3-Setup.

Für eine Ziel-AC-3 mit sechs Kanälen DVD-übliche Bitraten von 384 oder 448 kbit/s einzustellen, ist wenig sinnvoll. Dann hätten wir gleich das Original beibehalten können. Weniger als 256 kbit/s sollten wir bei **Bitrate** mit dem Schieberegler nicht auswählen, um die Einbußen bei der Qualität im Rahmen zu halten. AC-3 ist nun mal nicht für niedrige Bitraten ausgelegt.

5.1 unter **Output Channels** behält die sechs Kanäle der Quelldatei bei. Stereo-AC-3s zu encodieren oder gar originale Stereo-AC-3s zu verkleinern, ist unsinnig. Schon MP3 ist für 2-Kanal-Material besser geeignet, von Vorbis oder AAC ganz zu schweigen.

Damit ist die AC-3-Konfiguration auch schon erledigt und wir können mit **Start Processing** das Transcoding beginnen.

B.3 Untertitel

►A.2.6 Um ►Untertitel in den Film einzubinden, stehen uns drei Alternativen zur Verfügung: fest ins Bild einbrennen, dynamisch als Grafiken (Vobsub) oder dynamisch als Text (SubRip) muxen. Wie wir das im Einzelnen umsetzen können, damit befasst sich dieser Abschnitt.

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.

►A.2.3 Wie schon bei den Audiospuren, so spielt auch bei den Untertiteln der ►Container eine entscheidende Rolle dabei, welche Untertitelformate überhaupt in Frage kommen. Im Einzelnen sieht die Situation so aus:

► **Matroska** bietet uns volle Flexibilität. Egal ob Vobsubs, SubRips, andere Textformate wie SSA oder Exoten wie MicroDVD, es gibt praktisch keine Einschränkung. Vobsubs können außerdem intern verlustlos komprimiert werden, was sie auf etwa ein Drittel ihrer ursprünglichen Größe schrumpft und ein paar MByte mehr für das Video freimacht.
► **MP4** besitzt sein eigenes Textuntertitel-Format namens *MPEG-4 Timed Text* (TTXT), das laut Standard auch die einzige erlaubte Form von Untertiteln ist. Beim Muxen werden Textuntertitel automatisch in TTXT umgewandelt. Nero hat daneben eine Methode eingeführt, um Vobsubs zu muxen. Allerdings können wir uns nicht sicher darauf verlassen, dass deren Wiedergabe auch klappt, weshalb ich auf Vobsub in MP4 verzichten würde.
► **AVI** ist praktisch vollständig inkompatibel mit jeder Art von Untertiteln. Zwar kann AVI-Mux GUI Textuntertitel im SubRip-Format in AVI tricksen, was jedoch von den populären Playern VLC und mplayer nicht unterstützt wird.

SubRip ist das einzige Format, das mit allen Containern zusammenarbeitet – und sogar von manchen Hardwareplayern erkannt wird. Und natürlich bleibt immer die Möglichkeit, die Untertitel ins Bild einzubrennen, denn dann handelt es sich ja nicht mehr um Untertitel im eigentlichen Sinn.

B.3.1 Dynamische und eingebrannte Grafikuntertitel

Dieses Kapitel dreht sich um Vobsubs, d. h. die Grafikuntertitel der DVD. Wenn wir mit SubRip Textuntertitel erzeugen wollen, können wir gleich ins nächste Kapitel springen. Hier sehen wir uns an, wie wir Grafikuntertitel als dynamische (ausblendbare) Untertitel aufbereiten und wie wir sie ins Bild einbrennen können.

Untertitel rippen

Die Vobsubs sind noch in den VOB-Dateien verpackt. Um sie verarbeiten zu können, müssen wir sie erst einmal extrahieren und als Kombination aus IDX- und SUB-Datei speichern. Die IDX ist eine Textdatei, die zur Konfiguration der Untertitelspuren dient und alle Metainformationen (z. B. die Sprache der Spuren) sowie die genauen Zeitmarken enthält. In der SUB-Datei liegen die eigentlichen Untertitel-Grafiken.

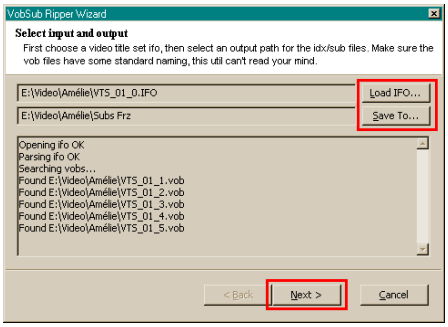


Abb. B-20: VSRip: Quelle und Ziel wählen.

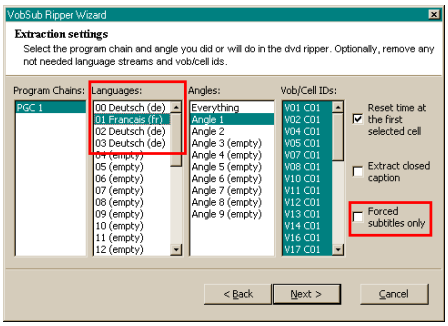


Abb. B-21: VSRip: Untertitel wählen.

Für das Untertitel-Ripping existieren zwei Standard-Tools. Einmal das schon etwas ältere, aber robustere, Vobsub, und dann das neuere VSRip. Wir verwenden vorzugsweise VSRip, da es uns direkte Unterstützung fürs Zwangsunteritel-Ripping bietet. Allerdings müssen wir beim Ripping der DVD darauf achten, die Unterteilung in einzelne VOB-Dateien nicht anzutasten. Wenn wir den Film als eine einzige große VOB-Datei auf die Platte rippen, kann VSRip nicht alle Untertitel extrahieren. Mir ist allerdings kein Ripper bekannt, der in seiner Standardeinstellung keine einzelnen VOBs erzeugt.

Wir starten VSRip, das uns das Fenster aus Abb. B-20 präsentiert. Über den Button **Load IFO** öffnen wir die Datei `VTS_xx_0.ifo`, die wir beim DVD-Ripping mit auf die Platte kopiert haben. VSRip liest die Datei und sucht die dazu passenden VOBs. Dann können wir mit **Save To** Namen und Ordner der Zieldateien festlegen und mit **Next** ins nächste Fenster wechseln (Abb. B-21).

Unter **Languages** sehen wir die vorhandenen Untertitelspuren. Die Liste unter **Angles** ist nur für DVDs mit mehreren Blickwinkeln interessant, in diesem Fall auch

Seamless Branching genannt. Hier wählen wir den Winkel aus, den wir auch gerippt haben. Bei **Vob/Cell IDs** sollten wir alles markiert lassen, da sonst nur ein Teil der Untertitel gerippt wird.

Eine IDX/SUB-Datei-Kombination kann mehrere Sprachen enthalten. Wenn wir Untertitel fest ins Bild einbrennen wollen, nutzen wir das der Einfachheit halber, wählen sämtliche Sprachen aus und packen alles in eine Datei zusammen. Die richtige Spur wählen wir dann später. Für dynamische Vobsubs rippen wir jede Sprache einzeln, um einerseits volle Flexibilität zu behalten. Andererseits ist es nicht Aufgabe der IDX/SUB-Dateien, mehrere Untertitel zusammenzupacken. Dafür ist der Container für den gesamten Film zuständig. Wir wählen also jeweils nur eine Sprache aus. Und jetzt wird auch wichtig, ob wir eine komplette Untertitelspur oder nur Forced Subs haben wollen.

Zwangsuntertitel

Erinnern wir uns ans Hintergrundkapitel A.2.6: Zwangsuntertitel können entweder eine eigene Spur haben oder in einer Komplettspur verpackt sein. Leider lässt sich nirgendwo einfach ablesen, welche Variante unsere aktuelle DVD benutzt. Beim Abspielen der DVD einen Blick auf die aktive Untertitelspur zu werfen, kann helfen, denn der Player zeigt die Spuren in der gleichen Reihenfolge an wie VSRip. Meistens hilft allerdings nur ausprobieren.

Bei einer eigenen Zwangsspur wählen wir diese unter **Languages** aus, so dass nur der passende Eintrag markiert ist. Der Haken der Option **Forced subtitles only** auf der rechten Seite darf nicht gesetzt sein.

Bei verpackten Zwangsuntertiteln wählen wir die passende Komplettspur aus und setzen den Haken bei **Forced subtitles only.** VSRip ignoriert dann alle Untertitel, für die kein Forced-Flag gesetzt ist.

In jedem Fall gilt: Im fertigen Encoding müssen Zwangsuntertitel eine eigene Spur erhalten, da kein Forced-Mechanismus wie auf der DVD existiert. D. h. wir müssen sie auf jeden Fall gesondert rippen.

Komplettuntertitel

Sehr oft existieren mehrere Spuren für eine Sprache. Auch hier hilft es, die DVD mit eingeschalteten Untertiteln abzuspielen und sich die Position der aktiven Spur zu merken. Die normalen Untertitelspuren stehen meistens am Anfang, evtl. gefolgt von Untertiteln für Hörgeschädigte und Spuren für Audiokommentare etc. Genauso wie bei den Zwangsuntertiteln gilt jedenfalls, dass wir jede Spur in eine eigene Datei rippen, und deshalb jeweils nur eine Sprache auswählen. Und natürlich

darf bei **Forced subtitles only** kein Haken gesetzt sein. Haben wir alle Einstellungen erledigt, startet ein Klick auf **Next** das Ripping. Dynamische Vobsubs sind damit erst einmal erledigt.

Eingebrannte Vobsubs

Eingebrannte Untertitel sind auf keinen Fall zu empfehlen und sollten nur der letzte Notnagel sein, wenn es gar nicht anders geht. Um die Untertitel auszuwählen, die eingebrannt werden sollen, öffnen wir mit einem Texteditor die gerade erstellte IDX-Datei. Nach dem Dateikopf folgen die einzelnen Untertitelspuren. Der Anfang einer solchen Spur sieht etwa so aus:

```
# Deutsch
id: de, index: 0
# Decomment next line to activate...
# alt: Deutsch
# Vob/Cell ID: 1, 1 (PTS: 0)
timestamp: 00:00:35:240, filepos: 000000000
timestamp: 00:00:41:280, filepos: 000001800
timestamp: 00:00:46:320, filepos: 000003000
timestamp: 00:00:51:480, filepos: 000004000
timestamp: 00:00:56:520, filepos: 000005800
timestamp: 00:01:01:560, filepos: 000007000
timestamp: 00:01:05:840, filepos: 000008800
timestamp: 00:01:11:400, filepos: 00000a000
timestamp: 00:01:17:880, filepos: 00000b000
timestamp: 00:01:21:920, filepos: 00000c800
timestamp: 00:01:26:960, filepos: 00000e000
```

Jeder Spur ist eine Sprache und eine Nummer (`index`) zugeordnet. Jede der vielen Zeilen darunter stellt eine Zeitmarke dar, an der ein Untertitel erscheint. Wir suchen die richtige Sprache heraus, wobei bei mehreren Spuren pro Sprache wieder der DVD-Player helfen kann. Einfach zu finden sind dagegen Zwangsuntertitel in einer eigenen Spur, denn eine solche Spur ist ganz deutlich kürzer als eine Komplettspur, enthält also viel weniger `timestamp`-Zeilen. Haben wir die richtige Spur gefunden, dann merken wir uns die Zahl hinter `index` und suchen folgende Zeilen (direkt über der ersten Untertitel-Spur):

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

Hinter `langidx` tragen wir die gemerkte Zahl ein, um die Spur mit diesem Index zu aktivieren. Egal, wie viele Spuren die Datei enthält, angezeigt wird immer nur genau diejenige, die bei `langidx` eingetragen ist. Nun suchen wir weiter nach folgenden Zeilen:

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

Wenn wir eine Komplettspur mit Forced Subs haben und nur die Zwangsuntertitel anzeigen wollen, müssen wir hinter `forced subs` ein `ON` eintragen; ansonsten `OFF`, um alle Untertitel der Spur anzuzeigen. Auch für Zwangsuntertitel in einer eigenen Spur sollte `OFF` stehen.

Damit sind die Untertitel fertig konfiguriert. Wir können die IDX-Datei speichern. Das Einbinden der Untertitel ins Bild geschieht übers AviSynth-Skript, womit wir uns später in den Kapiteln zu den Encoding-Frontends beschäftigen.

B.3.2 Dynamisch einblendbare Textuntertitel

In diesem Kapitel kümmern wir uns um dynamisch einblendbare Textuntertitel. Das benötigte Programm heißt SubRip. Dort gelangen wir über **Options / Global Options** in den Dialog aus Abb. B-22. Mit einem Haken bei **Forced subtitles only** können wir aus einer kompletten Untertitelspur nur die Zwangsuntertitel herausziehen, wenn die nicht in einer extra Spur gespeichert sind.

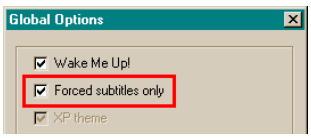


Abb. B-22: Forced-Subs-Option in SubRip.

Im Hauptfenster rufen wir mit dem **VOB**-Button den Öffnen-Dialog auf (Abb. B-23). 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 (Abb. B-24). **Last Time Code** muss

Null sein. Steht hier ein Wert, addiert ihn SubRip zu jeder Zeitmarke dazu, was zu asynchronen Untertiteln führt.

Da wir eine Textdatei erzeugen wollen, die Untertitel auf der DVD aber als Grafiken gespeichert sind, muss sie SubRip per OCR (*optical character recognition*, optische Texterkennung) in Text umwandeln. Dabei erkennt das Programm die Buchstaben aber nicht wie professionelle OCR-Software selbständig, sondern fragt nach, welches Grafikmuster zu welchem Buchstaben gehört. Das geschieht im Dialog aus Abb. B-25. Über den umrandeten blauen Buchstaben (hier das »A«) 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. Mit den **Pfeil**-Buttons oberhalb des Untertitels lässt sich die Breite des Erkennungsbereichs verringern und erhöhen. Das ist besonders dann nützlich, wenn SubRip Anführungszeichen oder das Prozentzeichen nicht vollständig erkennt.

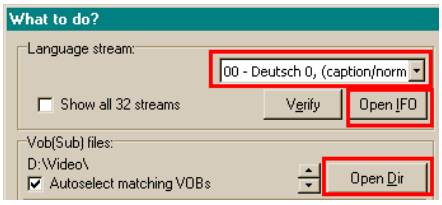


Abb. B-23: DVD-Untertitel laden in SubRip.

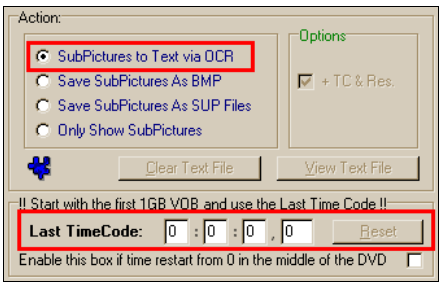


Abb. B-24: OCR-Erkennung und Timecode-Anpassung.

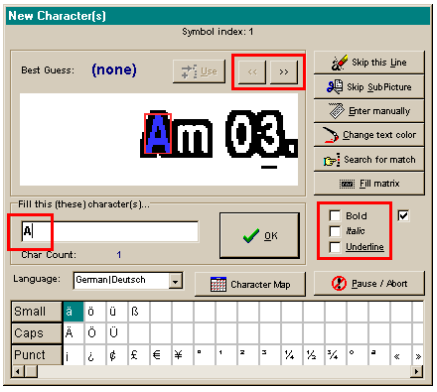


Abb. B-25: OCR-Erkennungsdialog von SubRip.

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 Erkennung 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 (Abb. B-26), in dem wir den gerade gerippten Untertitel in Textform (genauer: im SubRip-Format) sehen.

► Framerate (1)
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 framebasierten Format wie MicroDVD speichern wollten.

► OCR-Fehlerkorrektur (2)
Damit lassen sich die gröbsten Fehler der Texterkennung korrigieren. Die ist zwar recht gut, aber nicht fehlerfrei, weshalb wir immer einen Korrekturlauf durchführen sollten. Die Optionen des Dialogfensters sollten wir bis auf eine Ausnahme so wie voreingestellt stehen lassen. SubRip verwechselt gerne das große I und kleine L, weil zwischen den beiden Buchstaben oft kein Unterschied besteht. Ist das der Fall, setzen wir den Haken bei **Try to correct I and l** und wählen darunter die richtige Sprache. Die Funktion sollten wir aber nur dann einschalten, wenn die Buchstaben auch tatsächlich vertauscht sind. Ansonsten bringt SubRip nämlich einiges durcheinander. Solange wir darauf achten, haben wir mit englischen Untertiteln kein Problem. Die sind in der Regel nach der Autokorrektur (zumindest nahezu) fehlerfrei. Etwas komplizierter ist die Situation im Deutschen, denn mit der höflichen Anrede (dem *Sie* mit großem S) kommt die I-l-Korrektur nicht zurecht, was zu Wörtern wie »Lhrem« anstatt »Ihrem« führt. Eine manuelle Korrektur ist deshalb bei deutschen Untertiteln immer notwendig.

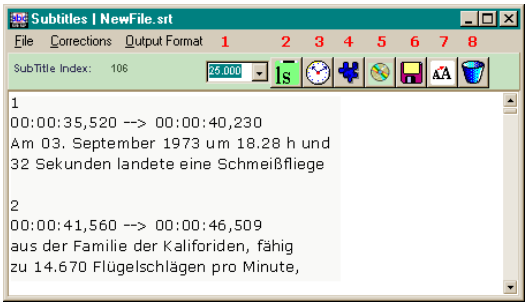


Abb. B-26: SubRips Editorfenster.

► Zeitkorrektur (3)
Damit 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.

► Formatkonvertierung (4)
Konvertiert die Untertitel in ein anderes Format.

► Splitting (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.

► Speichern (6)
Speichert die aktuelle Untertitel-Datei. Das sollten wir natürlich nicht vergessen. Wenn SubRip fragt, ob wir die Datei als Unicode speichern wollen, ist die richtige Antwort **Nein**, da wir sonst eventuell beim Muxen Probleme bekommen.

► Schriftwahl (7)
Wechselt Schriftart und Größe. Fürs SubRip-Format wirkt sich das nur auf die Anzeige im Editorfenster aus. Bei erweiterten Untertitelformaten wie SSA wird diese Einstellung auch in der Datei gespeichert und die Untertitel in der gewählten Schrift wiedergegeben.

Wenn wir nach dem Speichern der ersten Spur weitere Untertitel erkennen lassen wollen, leeren wir erst das Editorfenster über den Papierkorb-Button (7). Dann rufen wir mit dem **VOB**-Button den Öffnen-Dialog auf, wählen die neue Untertitelspur aus, laden auf der rechten Seite des Fensters oben die Character-Matrix-Datei und nullen mit **Reset** den Timecode, sonst passen die Zeitmarken nicht mehr.

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

B.3.3 Zwangsuntertitel simulieren

Auf der DVD haben wir ganz selbstverständlich die Möglichkeit, die Anzeige mancher Untertitel zu erzwingen. Leider existiert fürs fertige Encoding ein solcher Mechanismus nicht. Um trotzdem sicherzustellen, dass Zwangsuntertitel immer und alle anderen nur auf ausdrücklichen Wunsch angezeigt werden, müssen wir einen Umweg gehen.

Das Problem

Stellen wir uns vor, wir haben einen Film mit folgenden Spuren:

Abb. B-27: Film mit Zwangs- und Komplettuntertiteln.

Manche Splitter-Filter, z. B. den Haali Media Splitter, können wir so einstellen, dass Untertitel automatisch angezeigt werden und dass genauso automatisch beim Wechsel der Tonspur auch die Untertitel auf die passende Sprache umschalten. Dann haben wir mit diesem Film kein Problem. Wir hören beim Filmstart den englischen Ton und sehen dazu die erste englische Untertitelspur, d. h. die passenden Zwangsuntertitel. Beim Umschalten auf den deutschen Ton wechseln auch die Untertitel auf die deutsche Zwangsspur. Manuell können wir außerdem jederzeit eine der vollständigen Spuren auswählen. Bei einem Splitter ohne Wechselautomatik funktioniert das genauso, nur dass die Untertitelspur nicht automatisch umgeschaltet wird.

Ebenfalls unproblematisch sind Filme, die gar keine Untertitel enthalten. Erst bei Filmen mit ausschließlich Komplettuntertiteln macht sich die fehlende Zwangsuntertitelfunktion bemerkbar. Eine solche Datei würde folgendermaßen aussehen:

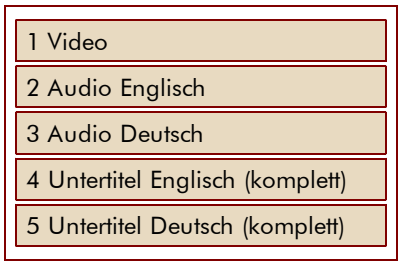


Abb. B-28: Film mit Komplett- und ohne Zwangsuntertitel.

Beim Filmstart wird genauso wie oben die englische Tonspur und die englischen Untertitel gestartet. Dumm nur, dass das eine vollständige Spur ist, die wir standardmäßig gar nicht aktiviert haben wollen. Es muss also ein Workaround her. Wir könnten den Player so einstellen, dass Untertitel immer manuell aktiviert werden müssen. Das hat allerdings den Nachteil, dass dann auch keine Zwangsuntertitel mehr automatisch angezeigt werden. So kommen wir also nicht weiter.

Lösung: Leere Zwangsuntertitel

Das Problem lässt sich lösen, indem wir so tun, als wären Zwangsuntertitel vorhanden. Wir müssen lediglich pro Audiosprache eine leere Untertitelspur hinzufügen, so dass der Film schließlich so aussieht:

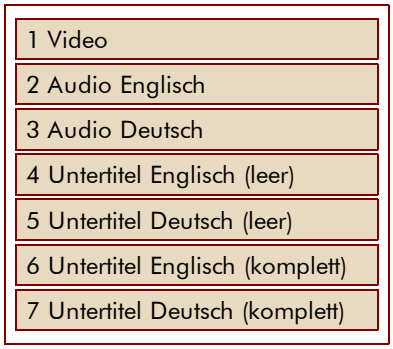


Abb. B-29: Film mit leeren Zwangsuntertiteln.

Beim Abspielen wird wie im ersten Beispiel automatisch die englische Tonspur und die erste englische Untertitelspur aktiviert. Das macht aber nichts, da diese keine Einträge enthält, d. h. sich genauso verhält, als wären die Untertitel komplett abgeschaltet.

Die zweite leere Spur ist für Splitter mit Wechselautomatik nötig. Ohne sie würden beim Umschalten der Tonspur die deutschen Komplett-Untertitel aktiv. Ohne eine solche Automatik ist natürlich nur eine leere Spur wirklich notwendig.

Eine kleine praktische Hürde müssen wir noch überwinden. Untertitel sind nicht darauf ausgelegt, nichts zu enthalten. Deswegen ist eine tatsächlich leere Spur nicht möglich. Stattdessen basteln wir uns eine Spur, die nur einen einzigen unsichtbaren Untertitel enthält. Am einfachsten geht das mit SubRip, und der Einfachheit halber könnt ihr euch die »leere« SRT-Datei fertig herunterladen:
`http://encodingwissen.de/resources/SubRip-Leerspur.zip`

```
1
00:00:01,000 --> 00:00:02,000
```

Oben sehen wir den Inhalt der Untertitel-Datei. Die Spur enthält einen einzigen Eintrag, der eine Sekunde nach Filmstart eine Sekunde lang ein geschütztes Leerzeichen anzeigt (die Zeile unter der Zeitmarke). Am Bildschirm sehen wir somit nichts. Ein einfaches Leerzeichen war deshalb nicht möglich, weil das nicht als eine gültige Textzeile erkannt wird. Deswegen müssen wir auf das geschützte Leerzeichen ausweichen, das genauso »aussieht« und über den Ziffernblock der Tastatur mit **Alt + 0 1 6 0** eingegeben werden kann. **NumLock** muss dabei aktiviert sein.

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 allen 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.

Tab. B-3 auf der nächsten Seite bietet eine Übersicht über die Fähigkeiten der Frontends, die wir uns näher ansehen. Sie soll eine Entscheidungshilfe sein, welches Frontend den eigenen Vorlieben und Anforderungen am besten entspricht.

B.4.1 StaxRip

StaxRip wird von stax entwickelt, der sich schon für den Vorgänger namens DVX verantwortlich zeichnete, und ist GPL-lizenzierte Open-Source-Software. StaxRip legt viel Wert darauf, den Benutzer Schritt für Schritt an die Hand zu nehmen, aber trotzdem einen vollständigen Funktionsumfang für die Profis zu bieten. Dieses Konzept funktioniert auch recht gut, so dass wir mit dem aktuellen StaxRip ein sehr nützliches Stück Software vor uns haben. Lediglich als Gordian-Knot-Umsteiger darf man sich von dem deutlich anderen Bedienkonzept nicht gleich abschrecken lassen.

Als moderne Alternative zu Gordian Knot macht sich StaxRip bestens. Insgesamt fühlt sich die Bedienung deutlich durchdachter und intuitiver an als die des großen Schwesterprojekts MeGUI, von Gordian Knot ganz zu schweigen. Die selbstverständliche Unterstützung von AAC, MP4 und x264 CLI kann auch die Gordian-Knot-Erweiterung agkp nicht aufwiegen. Fehlt eigentlich nur XvidEncraw, denn hier ist StaxRip bisher auf Xvid VfW beschränkt.

StaxRip baut auf Microsofts .NET-Technologie auf und benötigt ein installiertes .NET Framework 3.5, um gestartet werden zu können.

	StaxRip	Gordian Knot	agkp
	Container		
Matroska	Ja	nur VDubMod	Ja
MP4	Ja	Nein	Nein
AVI	Ja	Ja	Ja
	Video		
Xvid	nur VfW	nur VfW	als Hack
DivX	Ja	Probleme mit v6	Nein
x264	Ja	nur VfW	Ja
	Audio		
Anzahl Spuren	2	2	2
MP3	Ja	Ja	Ja
Vorbis	Ja	Ja	Ja
AC-3	Ja	Ja	Ja
AAC	Ja	Nein	Nein
	Dynamische Untertitel		
SubRip	Ja	nur 1 Spur	nur 1 Spur
Vobsub	Ja	Nein	Nein

Tabelle B-3: Funktionsumfang der Encoding-Frontends.

B.4.1.1 StaxRip einrichten

Für die Konfiguration von StaxRip sollten wir uns ein bisschen Zeit nehmen. Zwar erhalten wir in der Regel automatisch eine Meldung über fehlende Einstellungen und landen anschließend im passenden Dialogfenster, was aber tendenziell dazu führt, dass man bei den ersten Encodings vor dem Start eine ganze Reihe Meldungen bestätigen und Dialoge durchklicken muss. Wer damit besser zurecht kommt, kann das natürlich gerne so handhaben.

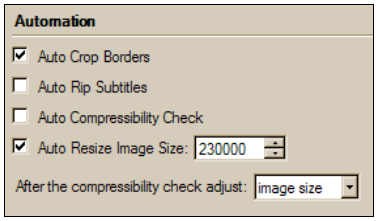


Abb. B-30: StaxRips Automatik-Konfiguration.

Die Konfiguration von StaxRip spielt sich auf zwei Ebenen ab. Zum einen existieren die Projektoptionen, die sich nur auf das aktuelle Encodingprojekt beziehen, zum anderen gibt es globale Optionen, die für StaxRip insgesamt und damit für alle Projekte gelten.

Projektoptionen

Zu den Projektoptionen gelangen wir über das Menü **View / Options** oder mit der Taste **F8**. Dort wechseln wir auf die **Automation**-Seite (Abb. B-30). Mit einem Haken bei **Auto Crop Borders** veranlassen wir StaxRip, beim Öffnen einer Quelldatei automatisch das Bild zuzuschneiden. Das ist ganz sinnvoll, auch wenn wir später noch einmal kontrollieren, ob tatsächlich sämtliche schwarzen Balken entfernt sind. **Auto Rip Subtitles** extrahiert beim Öffnen von VOBs automatisch sämtliche Vobsub-Untertitel.

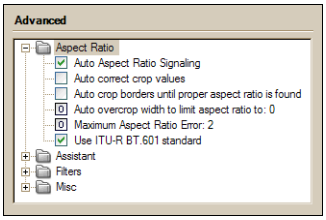


Abb. B-31: Optionen für Cropping und AR-Flag.

Auto Compressibility Check schließlich führt direkt nach dem Öffnen der Quelle einen Kompressionstest durch. Da wir vor dem Test noch einige Einstellungen vornehmen müssen, sollte dieser Haken nicht gesetzt sein.

Auto Resize Image Size stellt beim Öffnen des Quellvideos automatisch eine Zielauflösung ein, die ungefähr die angegebene Anzahl an Pixeln hat. Grobe ►Anhaltspunkte sind 160 000 Pixel für 1-CD-Encodings und 230 000 Pixel für 2-CD-Encodings. Natürlich befreit uns die Automatik nicht davon, selbst abzuwägen, ob die Auflösung tatsächlich für das jeweilige Encoding passt.

►A.3.3

Weiter geht's auf der Seite **Advanced** (Abb. B-31). Die meisten Optionen auf dieser Seite können wir bedenkenlos auf den Standardeinstellungen belassen. Inter-

essant ist lediglich der **Aspect-Ratio**-Ast, und zwar hauptsächlich dann, wenn wir ein ▸anamorphes Encoding im Sinn haben. In dem Fall muss **Auto Aspect Ratio Signaling** aktiv sein. Diese Option konfiguriert automatisch den Encoder, so dass der das nötige AR-Flag in den Videostream schreibt. Aufs klassische Encoding mit quadratischen Pixeln hat diese Option keine Auswirkung.

►A.3.2.2

Auto Correct Crop Values erzwingt beim Cropping bestimmte ▸Mod-Werte, und zwar mod4 bei klassischen Encodings (quadratische Pixel) und den unter **Misc / Forced output mod** angegebenen Wert bei anamorphen Encodings. Der Haken sollte nicht gesetzt sein, da für klassische Encodings das Cropping nicht eingeschränkt werden muss und wir uns für anamorphe Zieldateien besser selbst darum kümmern.

►C.2.1

Use ITU-R BT.601 standard sollte für DVD-Quellen aktiviert sein, um das passende Seitenverhältnis sicherzustellen.

Ein Klick auf OK bringt uns zurück ins Hauptfenster. Um die Einstellungen nicht bei jedem Encoding wiederholen zu müssen, können wir das aktuelle Projekt über **File / Save Project as Template** als Vorlage speichern, z. B. unter dem Titel »DVD«.

Globale Optionen

Die programmweite Konfiguration gilt immer für jedes Projekt und wird auch automatisch gesichert. Deshalb benötigen wir den Menüpunkt **Tools / Settings** im Idealfall nur einmal direkt nach der StaxRip-Installation. Unter **General / Startup Template** (Abb. B-32) stellen wir die Projektvorlage ein, die beim Start von StaxRip automatisch geladen wird. Hier bietet sich natürlich die Vorlage an, die wir gerade eben nach dem Setup der Projektoptionen gespeichert haben.

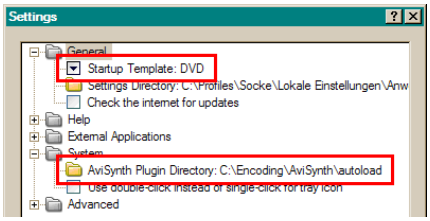


Abb. B-32: Globale Optionen.

Entscheidend wichtig ist **System / AviSynth Plugin Directory**. Dort legen wir den Ordner fest, in dem sämtliche AviSynth-Filter liegen, die wir benötigen. Das schließt auch die `DGDecode.dll` aus dem DGMPGDec-Paket mit ein, die wir benötigen, um die Videospur der DVD zu lesen. Diese Datei muss aus dem DGMPGDec-Ordner in den angegebenen Plugin-Ordner kopiert werden. Wenn wir DGMPGDec aktualisieren, müssen wir diesen Vorgang wiederholen, da `DGIndex.exe` und `DGDecode.dll` unbedingt die gleiche Version haben sollten.

Externe Programme

Das **Settings**-Fenster ist damit erledigt. Es bleibt noch die Konfiguration der externen Tools. Die ist wichtig, denn schließlich erledigt StaxRip das Encoding nicht selbst, sondern steuert nur den ganzen Haufen von Programmen, die die eigentliche Arbeit erledigen. Über **Tools / External Applications** erreichen wir das passende Fenster. Auf der linken Seite sehen wir eine Baumansicht aller Tools, mit denen StaxRip arbeiten kann. Rechts werden Informationen zum gerade ausgewählten Tool angezeigt. Die **Status**-Meldung **File not found** weist darauf hin, dass das Programm nicht gefunden wurde, wie in Abb. B-33 für x264 der Fall. Wir können nun über den **Download**-Button x264 automatisch herunterladen und installieren lassen oder über den Eintrag im **Tools**-Menü die Programm-Website im Browser öffnen. Haben wir x264 schon längst auf der Platte, klicken wir auf **Browse** und suchen den Ordner heraus, in dem sich die `x264.exe` befindet.

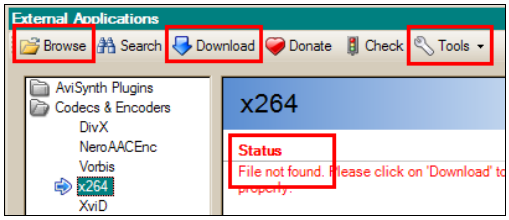


Abb. B-33: Dialog für externe Tools.

StaxRip ist äußerst wählerisch, was die Programmversionen angeht. Entspricht die installierte Version nicht genau derjenigen, die unter **Version** steht, wird das Tool vorsichtshalber als möglicherweise inkompatibel markiert. In den meisten Fällen ist das unkritisch, da auch abweichende Versionen funktionieren. Wegen jedem kleineren Update der immerhin zwei Dutzend Tools kann nun einmal keine neue StaxRip-Version erscheinen. Wenn Probleme auftauchen, sollten wir trotzdem erst einmal mit der offiziell unterstützten Version des betroffenen Programms testen. Besondere Vorsicht ist bei DivX geboten, denn der führt gerne mal mit einer neuen Version inkompatible Änderungen ein.

Bei unbekannten Programmversionen meckert StaxRip spätestens beim Encodingstart. Den Hinweisdialog bestätigen wir – schließlich wissen wir, was wir tun :-D – und erklären StaxRip dabei nach Lust und Laune, dass es zukünftig nicht mehr meckern soll.

Die folgenden Anwendungen benötigen wir nie für ein DVD-Backup: DirectShow Source, Java, ProjectX, DivXMux, PMP Muxer.

StaxRip ist nun vollständig konfiguriert, und wir können uns beruhigt ans Encoding wagen.

B.4.1.2 Konfigurieren der Zieldatei

In diesem Kapitel laden wir zuerst die Quelle. Je nach Vorgehen erstellt StaxRip dabei automatisch die nötige D2V-Indexdatei. Anschließend kümmern wir uns darum, die Zieldatei zu konfigurieren. Das beinhaltet die Wahl des Containers, das Einbinden von Kapiteln und dynamischen Untertiteln und natürlich das Festlegen der endgültigen Größe.

Laden und Indexieren der Quelle

Um das Quellvideo zu laden, doppelklicken wir im StaxRip-Hauptfenster im Source-Abschnitt in das Eingabefeld für die Quelldatei und wählen dann **Single or Merge**.

Wie wir den Film im folgenden Öffnen-Dialog laden, hängt davon ab, ob wir die VOBs schon manuell mit DGIndex behandelt haben. Wenn ja, öffnen wir die dabei erzeugte D2V-Datei. Ansonsten markieren und öffnen wir sämtliche VOBs, die wir beim Ripping erzeugt haben. Im anschließenden Dialog **Source Files** kann eine Kontrolle nicht schaden, ob die Reihenfolge tatsächlich stimmt. Notfalls lassen sich Fehler mit den **Up** und **Down**-Buttons beheben. Mit einem Klick auf OK starten wir den DGIndex-Prozess, der die benötigte D2V-Datei erzeugt und die Audiospuren aus den VOBs extrahiert. Das kann einige Minuten dauern. Solange reduziert sich StaxRip auf ein Statusfenster.

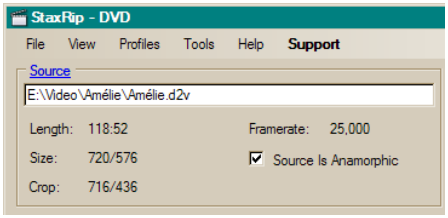


Abb. B-34: Eigenschaften der Quelle.

Das Laden der D2V-Datei benötigt ein paar Sekunden. Sollte StaxRip dabei in einem Dialog nach einem **Template** fragen, entscheiden wir uns für die vorhin bei der Projektkonfiguration angelegte eigene Vorlage. Dann erscheinen im **Source**-Abschnitt des StaxRip-Fensters der Name der Datei und die Eckdaten des Films. (Abb. B-34). Unter **Crop** sehen wir die automatisch zugeschnittene Auflösung; natürlich nur, wenn wir das Auto-Cropping nicht deaktiviert haben. Die Checkbox **Source is Anamorphic** setzt StaxRip automatisch auf den richtigen Wert, was wir auch nicht ohne guten Grund ändern sollten.

Konfigurieren des Containers

Im **Target**-Abschnitt hat StaxRip schon automatisch einen Namen für die Zieldatei vorgeschlagen, den wir ganz nach Belieben anpassen können. Dann wählen wir unter **Profiles / Container** das ▸Containerformat unserer Zieldatei. Das DivX Media Format (**DIVX**) und den PlayStation-Container (**PMP**) lassen wir außen vor und konzentrieren uns auf die üblichen Standardcontainer: Matroska, MP4 und AVI. ▸A.2.3

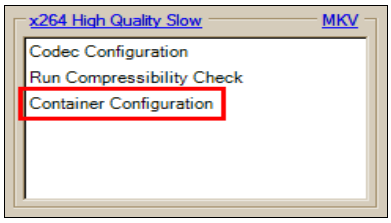


Abb. B-35: Menüpunkt für die Container-Konfiguration.

Beachten müssen wir, dass AVI nicht mit dem x264-Videoencoder kompatibel ist. StaxRip ändert bei der Auswahl dieser Kombination automatisch den Container auf Matroska. Beim Containerwechsel – egal ob automatisch oder manuell – gehen alle Einstellungen zu Untertitelspuren usw. verloren.

Um den Container zu konfigurieren, d. h. Kapitel und dynamische ▸Untertitel einzubinden, klicken wir im Konfigurationsabschnitt auf der rechten Seite des StaxRip-Fensters auf **Container Configuration** (Abb. B-35). ▸A.2.6

Matroska

Der Untertiteldialog für den Matroska-Container ist in Abb. B-36 dargestellt. Über den Öffnen-Button unter **Image Subtitles** können wir Vobsub-Untertitel laden und dann darunter auswählen, welche Untertitelspuren wir ins Encoding übernehmen wollen. Ein wenig Vorsicht ist dabei angebracht, denn es erscheint für jede auf der DVD vorhandene Spur ein Eintrag, auch für solche, die wir beim Ripping oder Extrahieren gar nicht berücksichtigt haben. Eine solche leere Spur anzuklicken, hat natürlich wenig Sinn. Auch ist StaxRip nicht darauf ausgelegt, mehrere Vobsub-Dateien einzubinden. Das heißt, sobald wir mehr als eine Vobsub-Spur haben, müssen wir nach dem Encoding noch einmal ▸MKVMerge anwerfen, um auch die zusätzlichen Spuren zu muxen. ▸B.5.1

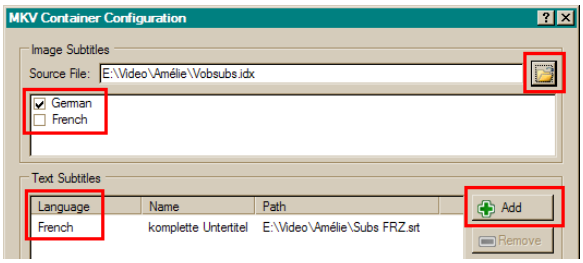


Abb. B-36: Untertitel-Setup für Matroska.

Über den **Add**-Button unter **Text Subtitles** laden wir Textuntertitel in den Formaten SubRip, Substation Alpha (SSA und ASS) oder MPEG-4 Timed Text (TTXT). Für jede Spur können wir anschließend in der Liste die Sprache (**Language**) und

Bezeichnung (**Name**) wählen, die im Player angezeigt werden sollen.

Weiter unten im Dialog (Abb. B-37) können wir bei **Chapters** die Kapitelliste laden. Normalerweise liegt die im Ogg-Format als TXT-Datei vor. Jedoch können wir genauso eine Matroska-XML-Liste laden, auch wenn StaxRip diese Option im Dialog nicht ausdrücklich anbietet.

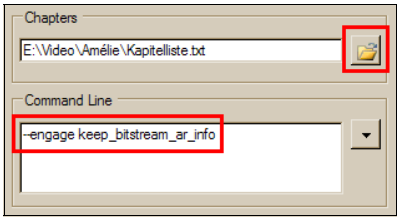


Abb. B-37: Matroska-Konfig: Kapitel und Spezialoptionen.

Im Feld **Command Line** übergeben wir zusätzliche Anweisungen an MKVMerge. Das ist besonders für anamorphes H.264 interessant, da MKVMerge in dem Fall standardmäßig die Seitenverhältnis-Information aus dem Videostream entfernt. Um das zu verhindern, tippen wir in das Eingabefeld die Option `--engage keep_bitstream_ar_info`.

MP4

Der Dialog für den MP4-Container sieht genauso aus und hat die gleichen Funktionen wie der für Matroska. Wir sollten lediglich bei **Image Subtitles** daran denken, dass Vobsub-Untertitel für MP4 nicht offiziell definiert sind. Deswegen sollten wir sie soweit möglich vermeiden. Der **Chapters**-Abschnitt ist wieder für die Kapitelliste zuständig; mit dem Unterschied, dass Matroskas XML-Format für MP4 nicht erlaubt ist.

AVI

Das AVI-Format unterstützt keine Kapitel, und dynamische Untertitel kann nur AVI-Mux GUI muxen. Da StaxRip fürs AVI-Muxing VirtualDubMod benutzt, fällt beides weg. Entsprechend gibt es nichts zu konfigurieren und der Klick auf **Container Configuration** öffnet lediglich den AVI-Infodialog. AVI ist außerdem nicht kompatibel zum x264-Videoencoder.

Einstellen der Zielgröße

Um die Konfiguration der Zieldatei abzuschließen, wenden wir uns dem **Target**-Abschnitt im Hauptfenster zu (Abb. B-38). Hier steht schon der vorgeschlagene Dateiname für den fertigen Film. Die Dateiendung wird automatisch für den gewählten Container angepasst. Haben wir uns für 2-Pass als ►Encodingmethode (vgl. Seite 60 ff.) entschieden, legen wir bei **Size** die Größe des Films fest. Entwe-

►A.4.1

der wählen wir mit einem Klick auf **Size** eine der vorhandenen Vorgaben oder tippen den MByte-Wert selbst sein. Fürs 1-Pass-Encoding hat **Size** keine Bedeutung und kann ignoriert werden.

Damit sind alle Einstellungen für die Zieldatei erledigt und wir können uns um die Audiospur(en) kümmern.

B.4.1.3 Einfügen von Audiospuren

Audio-Transcoding mit StaxRip ist nicht weniger flexibel als mit BeLight, nur unter Umständen unbequemer. Für typische Transcodings steht eine Reihe von Profilen zur Verfügung, die in den meisten Fällen vollkommen ausreichen sollte. Wer mag, kann die Profile auch an seine persönlichen Vorlieben anpassen oder neue hinzufügen. Dann ist es auch mit außergewöhnlichen Audio-Umwandlungen möglich, den gesamten Backupvorgang zu konfigurieren und als einen langen Prozess durchzuführen, ohne zwischendurch noch einmal eingreifen zu müssen. Nur wenn wir mehr als zwei Audiospuren im fertigen Encoding haben wollen, müssen wir zum Schluss noch einmal Hand anlegen.

Abb. B-38: Eigenschaften der Zieldatei.

Audiospuren konfigurieren

Im Abschnitt **Audio** des StaxRip-Fensters (Abb. B-39) fügen wir unserem Encoding die Audiospuren hinzu. **Track 1** und **Track 2** geben auch die Reihenfolge und damit die Priorität an, in der die Spuren in der fertigen Datei liegen. Bei **Track 1** tragen wir also die Spur ein, die beim Abspielen bevorzugt werden soll. Da uns StaxRip soviel Arbeit wie möglich abnehmen will, sucht das Programm beim Öffnen des Quellvideos automatisch nach Audiospuren, die wahrscheinlich dazupassen. Die Chancen stehen deshalb gut, dass wir uns ums Laden der Audio-Quelldateien gar nicht mehr kümmern müssen. Für jede der beiden Audiospuren zeigt StaxRip den Pfad zur Quelldatei und daneben den Namen des Transcoding-Pro-

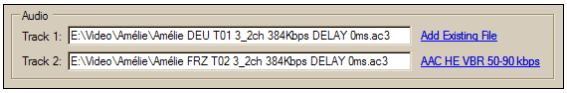


Abb. B-39: Audiospuren.

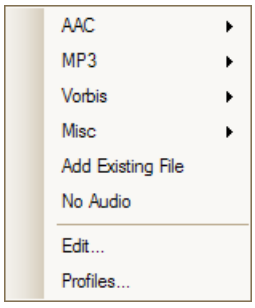


Abb. B-40: Konfigurationsmenü für Audiospuren.

fils. Die Quelldatei laden wir über einen Doppelklick in das Dateinamen-Feld, und zwar unabhängig davon, ob es sich um die AC-3 frisch von der DVD handelt oder um eine fertig transcodierte Tonspur, die wir mit BeLight erstellt haben. Ein Klick den blauen Profilnamen öffnet dann ein Menü, in dem wir sämtliche Einstellungen vornehmen (Abb. B-40).

►A.2.5

Wir entscheiden uns für ein ►Audioformat, indem wir aus dem Menü das passende Profil auswählen. Um die Original-AC-3 z. B. in eine hochqualitative 6-Kanal-HE-AAC-Spur umzuwandeln, würden wir aus dem Untermenü von AAC das Profil **HE 5.1 VBR 200-250 kbps** nehmen. Der Menüpunkt **Add Existing File** ist dann wichtig, wenn wir nicht transcodieren wollen, sondern eine schon fertige Audiospur unverändert in die Zieldatei übernehmen. Typischster Fall dafür ist die Original-AC-3. Aber auch schon fertig transcodierte Dateien kommen in Frage. **No Audio** ignoriert die angegebene Quelldatei vollständig, so als hätten wir sie nie geladen.

Wenn wir zwei Audiospuren einbinden, muss das Format nicht für beide gleich sein. Es spricht z. B. nichts dagegen, für die wichtige Audiospur die originale AC-3 zu übernehmen und die unwichtigere in platzsparendes HE-AAC umzuwandeln.

Der **Edit**-Button hat zwei Funktionen. Die erste ist immer interessant: Wir konfigurieren darüber die Sprache und den Delay-Wert der Audiospur. Je nachdem, ob wir ein Transcoding durchführen oder eine fertige Datei hinzufügen, sieht der Dialog etwas anders aus. Der erste Screenshot in Abb. B-41 zeigt die rechte obere Ecke des Dialogs bei aktivem Transcoding. Im zweiten Screenshot sehen wir den Dialog für hinzugefügte fertige Dateien. Wichtig sind jeweils die Felder **Language** und **Delay**. Unter **Language** stellen wir die Sprache der Audiospur ein. **Delay** ist die Anzahl Millisekunden, um die die Tonspur zur Videospur verschoben werden muss, damit beide synchron laufen. Der richtige Wert steht meistens im Dateinamen und wird dann von StaxRip automatisch übernommen. Eine Kontrolle kann trotzdem nicht schaden. Je nach Modus baut StaxRip das Delay beim Transcoding ein oder berücksichtigt es nach dem Encoding beim Muxen.

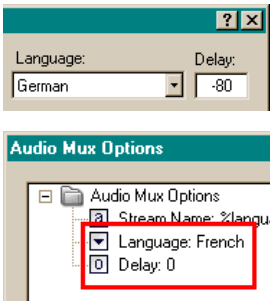


Abb. B-41: Auswahl von Sprache und Delay-Wert einer Audiospur.

Haben wir das Audio-Transcoding schon vorher mit BeLight erledigt und dort bereits den Delay-Wert berücksichtigt, muss das Delay in StaxRip immer auf 0 stehen.

Profile bearbeiten

Die zweite Funktion des **Edit**-Buttons gehört zu den fortgeschrittenen Techniken. Wenn nicht **Add Existing File** ausgewählt ist, können wir nämlich im **Edit**-Dialog die BeSweet-Kommandozeile anpassen, die fürs Transcoding benutzt wird; und damit steht uns direkt in StaxRip die gesamte Flexibilität von BeSweet zur Verfügung. Der Nachteil daran ist der, dass wir uns mit der Syntax und den Optionen von BeSweet auskennen müssen. Wem das zu kompliziert ist, der sollte sich an die vorhandenen Profile halten und alle Transcodings, die sich damit nicht abdecken lassen, außerhalb von StaxRip mit BeLight durchführen.

Wenn wir den **Edit**-Button verwenden, um BeSweet zu konfigurieren, gilt das nur für die gerade gewählte Audiospur im aktuellen Encoding. Außerdem können wir über **Profiles** die vorhandenen Profile ändern oder neue hinzufügen. Der Konfigurationsdialog ist beide Male der aus Abb. B-42. Unter **Output Type** stellen wir das Zielformat des Transcodings ein. Wir müssen manuell darauf achten, dass hier auch wirklich das steht, was wir mit unserer BeSweet-Konfiguration erzeugen. Daneben unter **Bitrate** tragen wir den Wert ein, der mit unserer Konfiguration unter normalen Umständen in etwa zu erwarten ist. Das muss kein ein exakter Wert sein, denn nach dem Transcoding rechnet StaxRip sowieso mit dem tatsächlichen Wert weiter.

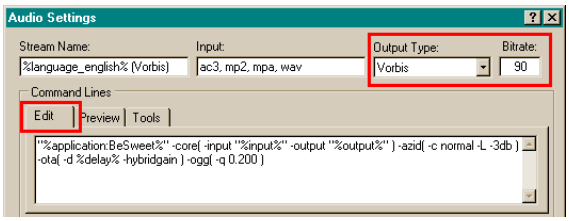


Abb. B-42: StaxRips BeSweet-Konfiguration.

Das große Eingabefeld im **Edit**-Register ist der Platz für die BeSweet-Kommandozeile. Welche Optionen zur Verfügung stehen und wie die Syntax gestaltet ist, bespricht die ►BeSweet-Kommandozeilenreferenz Eine Besonderheit in StaxRip sind die Makros, das sind Platzhalter, die von StaxRip automatisch mit dem für jedes Encoding passenden Wert ersetzt werden. Z. B. steht `%delay%` für den Delay-Wert und `%input%` für den Namen der Quelldatei. Eine vollständige Liste der Makros können wir über die Hilfe aufrufen (**F1** drücken).

►[BrJo08a]

Das soll für die BeSweet-Konfiguration auch schon genügen. Wer das alles zu kryptisch findet, sollte sich keine großen Gedanken machen und außergewöhnliche Transcoding-Aufträge mit BeLight erledigen.

B.4.1.4 Vorbereiten des Videos

In diesem Kapitel beschäftigen wir uns damit, das Video fürs Encoding vorzubereiten. Dazu gehört das Wegschneiden der schwarzen Balken (►Cropping), das Skalieren auf die Zielauflösung (►Resizing), das Konfigurieren der AviSynth-Filter und evtl. das Einbinden von ►eingebrannten Untertiteln. Zum Extrahieren der Untertitel von der DVD vgl. Kapitel B.3.

►A.3.1 ►A.3.3 ►A.2.6

Bild zuschneiden (Cropping)

Wenn das Quellvideo geladen wird, führt StaxRip standardmäßig ein automatisches Cropping durch. Obwohl das gut funktioniert, traue ich solchen Automatiken grundsätzlich nicht über den Weg und kontrolliere lieber noch einmal nach.

Abb. B-43: StaxRips Cropping-Dialog.

Den Croppingdialog erreichen wir über **F4** oder **View / Crop** (Abb. B-43). Um die Cropping-Werte zu korrigieren, suchen wir mit dem großen Schieberegler eine möglichst helle Stelle im Film heraus und bewegen die Maus an denjenigen Rand des Bildes, den wir verändern wollen (nicht klicken). Dort erscheint dann ein hellblauer Balken (im Screenshot ganz am oberen Rand), der diese Seite aktiviert. Mit den **Plus-** und **Minus**-Tasten auf dem Ziffernblock (und nur dort) können wir nun für die jeweils aktive Seite mehr oder weniger Pixel abschneiden. Ziel ist es, die schwarzen Balken vollständig zu entfernen. Im Zweifel ist es besser, ein oder zwei Pixelreihen des Bildes zu entfernen als eine schwarze Pixelreihe stehen zu lassen, denn der harte Übergang vom Bild zum Balken schluckt einiges an Bitrate.

In der Statuszeile am unteren Rand des Fensters versorgt uns StaxRip mit allen Informationen rund ums Cropping. Bei **Size** steht die Auflösung nach dem Zuschneiden. Unter **X** sehen wir die Anzahl der am linken und rechten Rand abgeschnittenen Pixelreihen, und **Y** verrät uns die Werte für die Ränder oben und unten. **ARE** steht für *Aspect Ratio Error*, also die Verzerrung, die durch Cropping und anschließendes Skalieren auf die Zielauflösung entsteht. Das haben wir im Kapitel zur Zielauflösung (Seite 56) schon genauer besprochen.

Besonderheiten für anamorphes MPEG-4

Mod zeigt an, durch welche Zahl die horizontale und vertikale Auflösung nach dem Cropping glatt teilbar ist. Für ein klassisches Encoding mit quadratischen Pixeln ist dieser Wert unwichtig, ganz im Gegensatz zum anamorphen MPEG-4. Damit der Encoder mit voller Effizienz arbeiten kann, muss das Bild in beiden Dimensionen durch 16 teilbar sein, die Statuszeile sollte also idealerweise **Mod: 16/16** zeigen. Mit etwas Glück passt das nötige Cropping auch zu dieser Anforderung, ohne dass Balken stehen bleiben oder wir mehr als ein paar wenige Pixel des Bildes wegschneiden müssen. Klappt das nicht, sollten wir uns unbedingt erst ab Seite 218 das Spezialkapitel zu den Mod-Regeln zu Gemüte führen und dann hier weiterlesen.

Befinden wir uns in einer Situation, die Nicht-Mod16 sinnvoll erscheinen lässt, ändert sich wenig. Nach wie vor schneiden wir das Bild anhand der **Mod**-Angabe passend zurecht. Wichtig ist allerdings, vorher im Menü des Hauptfensters unter **View / Options / Advanced / Aspect Ratio** die Option **Auto correct crop values** zu deaktivieren, denn sonst erzwingt StaxRip mindestens ein Mod4-Cropping.

Bild skalieren (Resizing)

Nach dem Zuschneiden schließen wir den Cropping-Dialog und wählen über den Schieber unter **Resize** die Zielauflösung (Abb. B-44) – es sei denn, wir encodieren anamorph, denn dann bleibt die Quellauflösung unverändert, wenn man vom Cropping absieht.

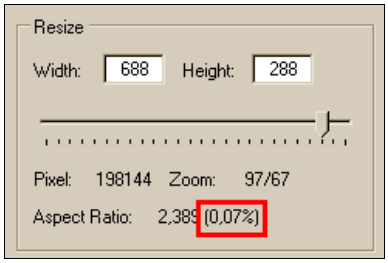


Abb. B-44: StaxRip-Einstellungen zum Skalieren.

Wir wählen also eine Auflösung, die zum Encoding passt. Den Zusammenhang zwischen Auflösung und Qualität haben wir uns im Kapitel zur Zielauflösung (Seite 57) schon genauer angesehen. Da StaxRip den dort verwendeten BPF-Wert nicht anzeigt, ist ein wenig Schätzen angesagt. Genau genommen unterscheidet sich das nicht besonders von dem, was der reichlich ungenaue BPF-Wert tut. Für 1-CD-Encodings sollten wir eine eher geringe Auflösung ansetzen, deren Fläche sich um 160 000 Pixel bewegt (abzulesen bei **Pixel**). Ein 2-CD-Encoding orientiert sich eher an einer Fläche von 230 000 Pixeln, ohne jedoch die ursprüngliche horizontale Auflösung von 720 Pixeln allzu sehr zu überschreiten. Besonders lange/kurze Filme und besonders große/kleine Audiospuren sollten wir natürlich einkalkulieren.

►A.4.1 Außerdem ist das Schätzen nur beim ►2-Pass-Encoding notwendig. Wenn wir in einem einzelnen Durchgang encodieren, ergibt sich die Auflösung daraus, wie perfekt wir das Detailniveau der Quelle beibehalten wollen. Ein typisches 1-Pass-Encoding verwendet überhaupt kein Resizing.

Dazu kommt der Wunsch nach einem möglichst kleinen Fehler beim Seitenverhältnis (Aspect Error). StaxRip zeigt diesen Fehler in der Klammer hinter **Aspect Ratio** an. Verrückt machen brauchen wir uns im Moment noch nicht. Wenn die Auflösung in einem einigermaßen sinnvollen Bereich liegt, reicht das vollkommen. Korrekturen sind nach dem Kompressionstest (nächstes Kapitel) immer noch möglich, wenn wir einen aussagekräftigen Indikator für die Qualität an der Hand haben.

Filter konfigurieren

Zum Abschluss der »Bildbearbeitung« bleibt uns noch die Konfiguration der AviSynth-Filter. Dafür ist der Abschnitt **Filters** links neben dem Auflösungsschieber zuständig (Abb. B-45). Dort sehen wir eine Liste von Filtern, die in der Reihenfolge auf das Video angewendet werden, wie sie in der Liste stehen. Mit der Maus können wir jeden Filter an eine andere Position ziehen. Die Checkbox vor jeder Zeile aktiviert den Filter oder schaltet ihn ab.

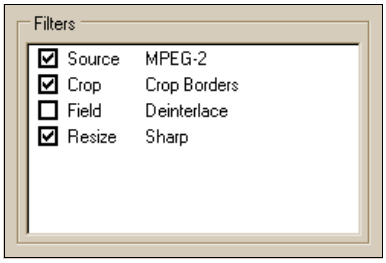


Abb. B-45: Filterliste des AviSynth-Skripts.

Source und **Crop** sollten selbsterklärend sein. Die beiden Einträge erledigen das Laden des Quellvideos und das Cropping. Natürlich sollten beide immer angehakt sein.

Field ist nur bei interlaced Video interessant, also wenn sich am Anfang in DGIndex Kammeffekte im Bild gezeigt haben. In diesem Fall setzen wir den Haken, um den Filter zu aktivieren und wählen über einen Rechtsklick auf den Filter einen der Deinterlacer. Empfehlungen zum richtigen Deinterlacing kann ich mangels Erfahrung leider keine geben.

Der **Resize**-Eintrag steuert den Resizing-Filter. Auch hier können wir mit einem Rechtsklick verschiedene Varianten auswählen. **Soft** (dahinter verbirgt sich der Bilinear-Filter) hat eher einen weichzeichnenden Effekt, der sich für hochkomprimierte Encodings anbietet. Über **Neutral** (Bicubic), **Sharp** (Lanczos) und **Very Sharp** (Lanczos4) wird das Bild immer schärfer und weniger komprimierbar. Der beliebteste Filter ist eindeutig Lanczos, also **Sharp.**

►A.2.6 Hat unser Film ►eingebrannte Untertitel, stellen wir über das Menü **Tools / Add Forced Subtitle** die mit VSRip erstellte und bearbeitete IDX-Datei ein. Dynami-

sche Untertitel interessieren hier nicht, denn die haben wir schon im letzten Kapitel mit der Container-Konfiguration erledigt. Eingebrannte Untertitel erhalten einen Filter-Eintrag in der Liste. Da die Untertitel in der Regel vor dem Resizing eingebunden werden sollten, verschieben wir diesen nach oben direkt über **Resize**.

Über einen Rechtsklick und **Add** können wir aus einer ganzen Liste von zusätzlichen Filtern wählen. Interessant ist möglicherweise die **Noise**-Kategorie, die Rauschfilter enthält. Gerade ältere Filmen können schon einmal störend heftiges Bildrauschen enthalten. Wichtig ist, nicht gleich in die Vollen zu gehen, denn ein Rauschfilter vernichtet immer nicht nur das Rauschen, sondern auch einige Details des Bildes. **Low** oder **Medium** sollten meistens schon ausreichen. Falls ein Rauschfilter nötig ist. fügen wir den ganz ans Ende der Filterliste ein.

Besonderheiten für anamorphes MPEG-4

Um mit StaxRip eine anamorphe Zieldatei zu erzeugen, entfernen wir den unnötigen **Resize**-Filter. Entweder klicken wir einfach den Haken vor dem Eintrag weg oder wir löschen den Filter komplett mit einem Rechtsklick und **Remove**. Damit werden die Steuerelemente im **Resize**-Abschnitt bedeutungslos. Was wir dort auch immer einstellen, es wirkt sich nicht auf das Encoding aus. Genauso bedeutet der abgeschaltete Resizing-Filter, dass StaxRip sich automatisch um das Setzen des AR-Flags in der Encoder-Konfiguration kümmert.

B.4.1.5 Konfigurieren des Videoencoders

Zwei Schritte fehlen uns noch bis zum Start des Encodings. Den ►Videoencoder müssen wir noch konfigurieren und dann einen Kompressionstest durchführen (nur beim 2-Pass-Verfahren), mit dem wir endlich einen wirklich brauchbaren Anhaltspunkt für die Qualität erhalten. Wenn wir gut geschätzt haben, bestätigt der Test unsere Einstellungen. Wenn nicht, müssen wir noch ein bisschen anpassen. ►A.2.4

Eine wichtige Eigenart von StaxRip sollten wir nie vergessen. Jedes mal, wenn wir im Menü **Profiles / Encoder** eine Einstellung vornehmen, ändert sich der Container der Zieldatei. StaxRip stellt nämlich automatisch den Container ein, den es für den gerade gewählten Encoder als bevorzugte Wahl ansieht. Das kann lästig werden, weil deswegen oft die Detailkonfiguration des Containers (z. B. Untertitel, Kapitel) verloren geht. Nach dem Setup des Encoders und spätestens direkt vor dem Encodingstart sollten wir deshalb nie vergessen, noch einmal den Container und dessen Konfiguration zu überprüfen.

Xvid und DivX: VfW-Konfiguration

Unter **Profiles / Encoder** wählen wir wie in Abb. B-46 gezeigt den gewünschten Videocodec (**Xvid** oder **DivX**) und abhängig von der ►Encodingmethode den Modus **Exact File Size** (2-Pass) oder **Constant Quality** (1-Pass). Ist die endgültige Größe unseres Films mit der gewählten Anzahl an CDs/DVDs fest vorgegeben, benötigen wir zwei Codierdurchgänge, um diese Größe exakt zu treffen. Interessiert uns hauptsächlich die Qualität, ohne dass wir eine bestimmte Größe erreichen wollen, reicht ein einzelner Encodingdurchgang.

►A.4.1

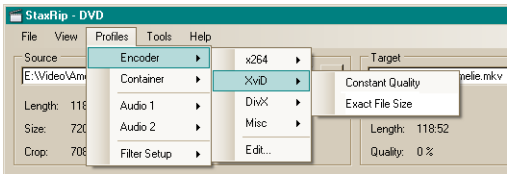


Abb. B-46: Encoderauswahl im Hauptmenü.

Xvid und DivX sind die beiden Encoder, für die StaxRip das VfW-Interface verwendet. Deshalb gibt es an dieser Stelle zur Konfiguration wenig zu sagen, denn darüber haben uns schon in den Kapiteln Xvid VfW ab Seite 88 und DivX VfW ab Seite 119 ausführlich unterhalten.

Die Codec-Dialoge erreichen wir über die Konfigurationsliste auf der rechten Seite des StaxRip-Hauptfensters (Abb. B-47). Im 2-Pass-Verfahren stellen wir den Encoder sowohl für den **Fist Pass** als auch den **Second Pass** identisch ein. Auch die Konfiguration für den **Compressibility Check** sollte genauso aussehen wie für den restlichen Film. Anstatt den Encoding-Modus für 1st oder 2nd Pass übernehmen wir allerdings den von StaxRip schon eingestellten Modus, der für den Kompressionstest stimmt. Für Xvid heißt das **Single Pass** mit einem **Target Quantizer** von **2**. Für DivX dagegen achten wir auf **1-pass quality based** mit einem **Target Quantizer** von **2**.

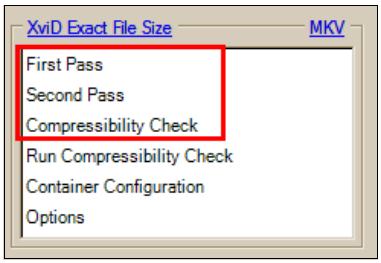


Abb. B-47: Konfigurationsbuttons für Xvid und DivX.

Ein 1-Pass-Encoding funktioniert prinzipiell genauso. Lediglich haben wir nur einen Durchgang zu konfigurieren und der Kompressionstest fällt weg.

x264-Konfiguration

X264 bietet grundsätzlich die gleichen Möglichkeiten wie Xvid und DivX: sowohl ►1-Pass als auch 2-Pass sind möglich. Darüber hinaus stehen uns für x264 im Menü **Profiles / Encoder / x264** eine ganze Reihe von vordefinierten Profilen bereit. Dabei handelt es sich zum Großteil um die von Sharktooth für MeGUI entwickelten x264-Konfigurationen, die einen weiten Bereich typischer Encodings ab-

►A.4.1

decken. Dank der Profile brauchen wir uns im Detail kaum um die x264-Konfiguration kümmern: einfach das passende Profil aussuchen und encodieren. Natürlich lässt sich die per Profil geladene Konfiguration ganz nach Belieben verändern, so dass wir auch x264 vollständig selbst konfigurieren können. Ausführliche Informationen finden wir im Doom9.org-Forum im Thread ►*MeGUI Custom x264/AVC video profiles*. Wir sehen uns nur kurz die interessantesten an. ►[Shar08a]

► **Anime**-Profile sind besonders auf Anime und Cartoons abgestimmt. **Standard** ist das schnellste, **Maxquality** das langsamste.
► **Common Encoding** enthält Konfigurationen für ein typisches Allerweltsencoding. Der Qualität zuliebe sollten wir uns für **Mainprofile** oder **Highprofile** entscheiden.
► **High Quality** eignet sich für hochqualitative Encodings, die x264 richtig ausreizen. Nachteil: Die Encodinggeschwindigkeit bricht gegenüber den Common-Encoding-Profilen deutlich ein. **HQ-Slow** dürfte von allen Profilen der sinnvollste Kompromiss zwischen Qualität und Geschwindigkeit sein.
► **DXVA** enthält Konfigurationen, die mit der DXVA-Hardwarebeschleunigung kompatibel sind.

Manuelle x264-Konfiguration

Die manuelle Konfiguration von x264 geschieht auf der rechten Seite des StaxRip-Fensters in der Konfigurationsliste, und zwar mit einem Klick auf **Codec Configuration** (Abb. B-48). Wenn wir ein Profil anpassen wollen, laden wir dieses zuerst nehmen dann im Konfigurationsdialog die Änderungen vor.

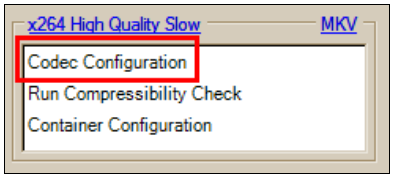


Abb. B-48: Konfigurationsbutton für x264.

Haben wir uns schon im Hintergrundwissen ab Seite 98 die x264-Konfiguration zu Gemüte geführt haben, sollte das Konfigurationsfenster selbsterklärend sein. Die einzelnen Optionen heißen genauso oder sehr ähnlich wie die Kommandozeilenparameter von x264.exe. Auf jede Optionen im Detail einzugehen, kommt mir deshalb reichlich sinnlos vor. Erwähnenswert sind lediglich zwei Dinge.

Das eine betrifft das 1-Pass-Encoding, für das die Auswahl an Profilen doch recht mager ist. Es spricht jedoch nichts dagegen, eines der 2-Pass-Profile zu verwenden und auf 1-Pass CRF umzustellen. Das geschieht auf der **Common**-Seite des x264-Dialogs (Abb. B-49). Unter **Mode** stellen wir auf **Quality**, um den 1-Pass-CRF-Modus zu aktivieren. Der **Quality**-Wert darunter gibt das Qualitätsniveau an,

das typischerweise zwischen **18** und **25** liegt. Fehlt noch ein Klick auf **OK** und schon haben wir ein 2-Pass-Profil fürs 1-Pass-CRF-Encoding umgebaut.

Der zweite interessante Punkt ist die Seite **Command Line** (Abb. B-50), auf der direkt in die x264-Kommandozeile eingegriffen werden kann. Wenn wir uns nicht genau genug mit x264 auskennen, um gezielte Einträge vorzunehmen, sollten wir StaxRips Vorgaben in Ruhe lassen.

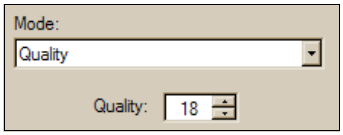


Abb. B-49: Einstellung für 1-Pass CRF

Unter **Add** können wir zusätzliche Parameter eingeben, die der StaxRip-Dialog nicht zur Verfügung stellt. Die beiden anderen Felder haben nur fürs 2-Pass-Encoding eine Bedeutung. Unter **Remove from preceding passes** listen wir die Paramter auf, die in allen Passes außer dem letzten (womit i. d. R. der 1st Pass übrig bleibt) deaktiviert sind. Werte für die Parameter dürfen dabei nicht angegeben werden. Es muss also `--me` heißen und nicht `--me umh` o. ä. Unter **Add to preceding passes** stehen analog die Parameter, die allen Passes außer dem letzten hinzugefügt werden. Hier müssen auch die passenden Werte der Parameter angegeben werden.

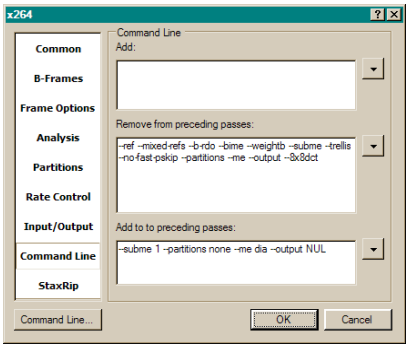


Abb. B-50: Fenster für x264-CLI-Anpassung.

Mit den beiden **Preceding**-Feldern ist es möglich, einen ▸schnellen 1st Pass zu bauen. Dazu müssen wir nicht einmal Hand anlegen, denn StaxRips Standardeinstellung tut genau das. Die Auswirkungen von Konfigurationsänderungen können wir mit dem Button **Command Line** betrachten.

▸A.4.4.2

Kompressionstest und Qualitätsabstimmung

Dieser Abschnitt ist nur fürs 2-Pass-Encoding relevant. Der Kompressionstest ist das einzige Mittel, um einen einigermaßen zuverlässigen Qualitätsindikator zu erhalten, der die Komplexität jedes Films berücksichtigt. Der Nachteil ist der, dass es sich nicht um eine reine Berechnung handelt, sondern ein Stück des Films (standardmäßig 5 %) encodiert werden muss. Das dauert natürlich einige Minuten, doch die sollten wir uns gönnen. Denn auch mit einiger Erfahrung kann die Pi-mal-Daumen-Technik, mit der wir bisher Zielgröße, Auflösung usw. festgelegt haben, versagen. Besser die paar Minuten für den Kompressionstest investiert als dass wir hinterher mit mieser Qualität dastehen und den ganzen Film noch einmal encodieren müssen.

Gestartet wird der Kompressionstest über den entsprechenden Eintrag in der Konfigurationsliste (Abb. B-51). Ein Klick auf **Run Compressibility Check** führt den Test durch und StaxRip reduziert sich so lange auf ein Statusfenster. Nach dem Test werfen wir einen Blick in den **Target**-Abschnitt (Abb. B-52). Unter **Quality** steht dort nun ein Prozentwert, der mit dem Test ermittelt wurde. Der sinnvolle Bereich liegt in etwa zwischen 60 und 90 Prozent. Darunter wird die Qualität schnell schlechter, darüber vergrößert sich nur die Datei ohne spürbare Steigerung der Qualität.

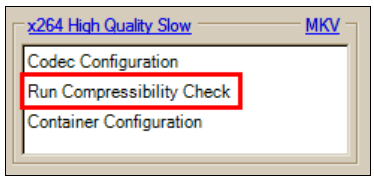


Abb. B-51: Start-Button für den Kompressionstest.

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-AC-3-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. Noch einmal ins Kapitel zur Zielauflösung (Seite 54) zu schauen, kann an dieser Stelle nicht schaden.

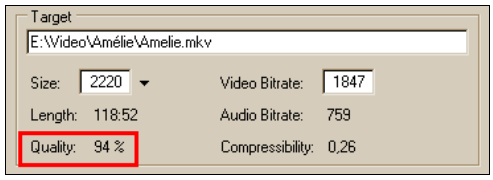


Abb. B-52: Qualitätsangabe im Target-Abschnitt.

100 % steht für die Sättigungsgrenze des Encoders. Wenn wir ein Encoding mit einem **Quality**-Wert von mehr als 100 Prozent starten, dürfen wir damit rechnen, dass die Zieldatei kleiner als gewollt wird.

Auch für anamorphe Encodings kann der Kompressionstest als Qualitätsindikator dienen. Wir können dann sogar die Untergrenze des sinnvollen Bereichs nach unten anpassen, etwa auf 50 %. Dass das keine hässlichen Auswirkungen auf die Qualität hat, liegt an dem Kompressionsvorteil, den der Verzicht aufs Resizing mit sich bringt.

Start des Encodings

Alle Einstellungen sind nun erledigt, der Compcheck-Wert liegt im sinnvollen Bereich, besonders die Container-Konfiguration ist noch ein letztes mal kontrolliert – dann sind wir bereit zum entscheidenden Klick.

Wahrscheinlich sind es doch mehrere Klicks, denn zuerst klicken wir ganz unten rechts im StaxRip-Fenster so oft auf **Next**, bis das Joblist-Fenster aus Abb. B-53 erscheint. Hier schließlich passiert der endgültige Klick auf den fetten **Start!**-Button. Und dann können wir uns für die nächsten Stunden zurücklehnen.

B.4.2 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 VfW hinzufügte. Seit Herbst 2005 steht die Entwicklung wieder still, und diesmal wahrscheinlich endgültig, obwohl es wegen der GPL-Lizenzierung jedem freisteht, sich vom Gordian Knot Sourceforge-Projekt den Quellcode zu besorgen und das Programm weiterzuentwickeln.

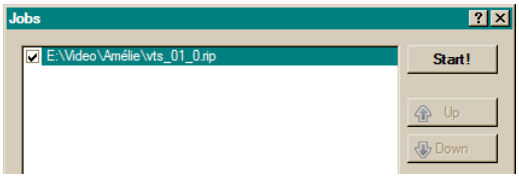


Abb. B-53: StaxRip-Jobliste.

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 fest. Diese veraltete Technologie ist so untrennbar mit Gordian Knot verzahnt, dass es nicht lohnend erscheint, das Programm an modernere Verfahren anzupassen. Aktuelle Container wie Matroska und MP4 werden gar nicht oder nur sehr eingeschränkt unterstützt. Ähnlich ist die Situation bei den Videocodecs x264 und DivX 6. Nur mit dem aktuellen Xvid-Encoder arbeitet Gordian Knot einwandfrei zusammen.

▸A.4.2

Wer sich für das altgediente, klassische DVD-Backup in AVI-Dateien mit Xvid entscheidet, für den stellt Gordian Knot nach wie vor ein ausgereiftes und bestens funktionierendes Frontend dar. Auch der Nostalgiefaktor spielt eine große Rolle. Viele der heutigen Experten haben ihre ersten Schritte mit Gordian Knot gemacht – einige 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? :-) Den Anforderungen moder-

ner Encoder- und Containertechnologie ist Gordian Knot allerdings längst nicht mehr gewachsen. Dafür sind neuere Frontends nötig, allen voran StaxRip und MeGUI.

Gordian Knot ist *nicht* identisch mit Auto Gordian Knot (meistens zu AutoGK oder AGK abgekürzt). Es handelt sich um zwei getrennte Programme mit völlig unterschiedlichem Bedienkonzept. Gordian Knot setzt auf Handarbeit, während AutoGK zu den 1-Klick-Tools gehört.

B.4.2.1 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.

Anschließend wechseln wir ins Register **Options**. Wichtig ist hier zuerst der fehlende Haken bei **Use advanced SaveAVS window** (Abb. B-54). Nicht, dass das erweiterte Fenster schlecht wäre. 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 ;-).

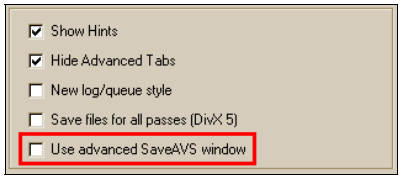


Abb. B-54: Globale Optionen in Gordian Knot.

Deutlich interessanter wird es dann auch bei den ►**Default codec settings.** Hier definieren wir mit dem VfW-Dialog unsere übliche Konfiguration für ►Xvid. x264 bleibt außen vor, da Gordian Knot nur den offiziell nicht mehr entwickelten und suboptimalen VfW-Encoder unterstützt. Gordian Knot und x264 sollten wir deshalb ausschließlich mit agkp (Kapitel B.4.2.7 ab Seite 192) verwenden. Auch DivX können wir ohne Probleme nur in der uralten Version 5.x benutzen. Bei der aktuellen DivX-6.x-Serie kann Gordian Knot nicht mehr automatisch die Bitrate setzen, weshalb wir fürs DivX-Encoding auf StaxRip umsteigen sollten. Die in Abb. B-55 getroffenen Einstellungen gelten als Standard für alle Encodings, kön-

►A.2.4
►A.4.3.2

nen aber im Einzelfall auch wieder überschrieben werden.

►A.4.1 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 sind noch zusätzliche Durchgänge möglich, um die Qualität weiter zu steigern. 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.

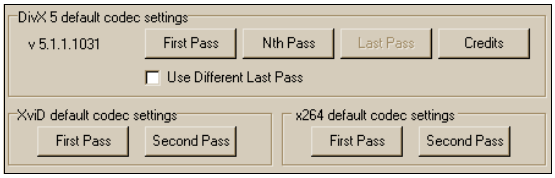


Abb. B-55: Gordian Knots Standard-Codecsetup.

Bleibt der Abspann des Films. Da es sich dabei meistens nur um weiße Laufschrift auf schwarzem Hintergrund handelt, können die Credits stärker komprimiert werden als der restliche Film. Das ist besonders bei hoch komprimierten 1-CD-Encodings interessant. Für Xvid kümmert sich Gordian Knot um die richtige Konfiguration, DivX-Credits konfigurieren wir über den **Credits**-Button selbst. Damit sind wir mit den Grundeinstellungen fertig und können ins Register **Bitrate** wechseln.

Auf 1-Pass-Encodings mit fest definierter Qualität und ohne exakte Zielgröße ist Gordian Knot nicht ausgelegt. Deswegen behandeln wir nur das 2-Pass-Encoding.

B.4.2.2 Kalkulieren der Bitrate

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

►A.2.3 ►A.2.4

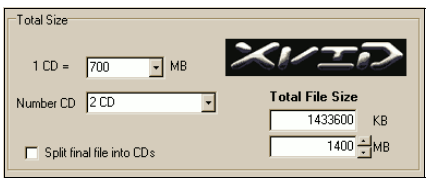


Abb. B-56: Auswahl der Zielgröße in Gordian Knot.

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 Da-

teigröße 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** (Abb. B-56) 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.

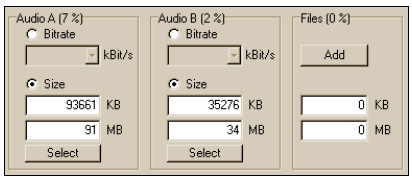


Abb. B-57: Größe der Audiospuren und zusätzlicher Daten.

Der Abschnitt aus Abb. B-57 enthält alle Daten, die außer dem Video auf die Disc müssen. Mit **Select** wählen wir die erste erstellte ►Audiospur 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.

►A.2.5

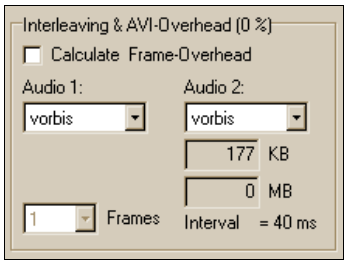


Abb. B-58: Berechnung des Overheads.

Schließlich wählen wir bei **Interleaving & AVI-Overhead** (Abb. B-58) 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.

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 Be-

darf verteilt und schnelle Szenen für die gleiche Qualität mehr Bits brauchen als langsame.

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

B.4.2.3 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** (Abb. B-59) den ►A.2.2 ►A.3.2 ►Typ der DVD und die ►Art des Bildes 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.

►A.3.1 Die rechte Seite des **Resolution**-Registers (Abb. B-60) ist fürs ►Cropping 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 rechts im Screenshot die schwarzen Ränder 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.

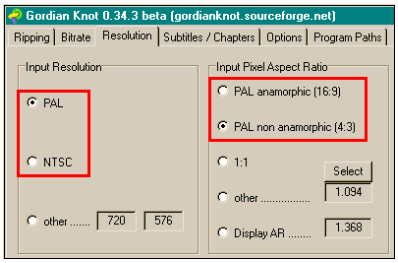


Abb. B-59: Quellvideo-Paramter im Resolution-Register.

Anschließend müssen wir im unteren Teil des Fensters (Abb. B-61) 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.

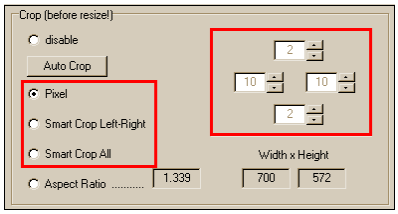


Abb. B-60: Cropping-Optionen.

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.

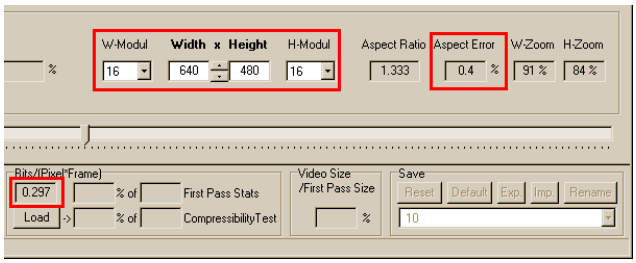


Abb. B-61: Wahl der Zielauflösung.

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 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.

▸A.3.2

Cropping

Das zugeschnittene Bild sollte wie beim nicht-anamorphen Encoding idealerweise ein durch 16 teilbares Seitenverhältnis haben (Mod16-Kriterium), um dem Codec das Encodieren nicht unnötig zu erschweren. Mit etwas Glück passt das nötige Cropping auch zu dieser Anforderung, ohne dass Balken stehen bleiben

oder wir mehr als ein paar wenige Pixel des Bildes wegschneiden müssen. Klappt das nicht, sollten wir uns unbedingt erst das Spezialkapitel zu den Mod-Regeln ab Seite 218 zu Gemüte führen und dann hier weiterlesen.

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! Das funktioniert genauso natürlich auch für alle anderen Mod-Werte.

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.4.2.5 erklärt, alle nötigen Einstellungen zu Rauschfiltern, Vobsubs, Deinterlacing usw. Über den **Edit**-Button öffnen wir dann das Skript und sorgen mit aktiviertem **No comments** für Übersicht.

Abb. B-62: Einbinden von SubRip-Untertiteln.

Um vollständig sicher zu gehen, dass tatsächlich anamorph encodiert wird, entfernen wir den Resizer aus dem Script. Dazu suchen wir nach folgenden Zeilen:

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

Je nach verwendetem Filter steht hier natürlich nicht unbedingt `LanczosResize`, sondern z. B. `BicubicResize` oder `BilinearResize`. Entweder löschen wir die Zeile komplett oder kommentieren sie aus, indem wir ein # an den Anfang setzen. Der Effekt ist der selbe.

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

B.4.2.4 Dynamische Untertitel und Kapitel einbinden

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

Unter **SRT Subtitles** (Abb. B-62) 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 deaktiviert, weil AVI praktisch vollständig inkompatibel mit Untertiteln ist.

►A.2.3

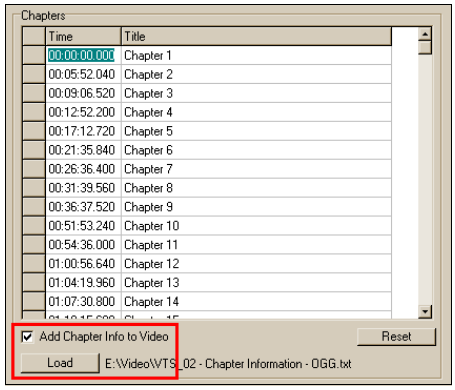


Abb. B-63: Geladene Kapitelliste im Ogg-Format.

Für die Kapitel ist die rechte Seite des Registers zuständig (Abb. B-63). 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.4.2.5 Encoding-Parameter konfigurieren

Eigene Einstellungen für den Abspann

Im Video-Fenster (Abb. B-64) 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.

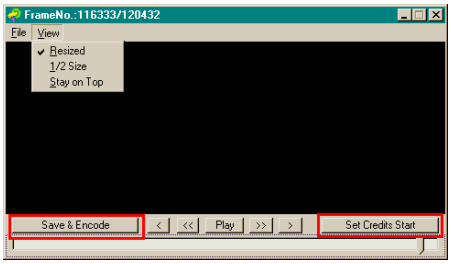


Abb. B-64: Das Video-Vorschaufenster von Gordian Knot.

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 Abb. B-65, in dem wir alle Filter konfigurieren. Hat unser Film eingebrannte Untertitel, stellen wir unter **Subtitles** (1) die mit Vobsub erstellte und bearbeitete IDX-Datei ein. Dynamische Untertitel interessieren hier nicht.

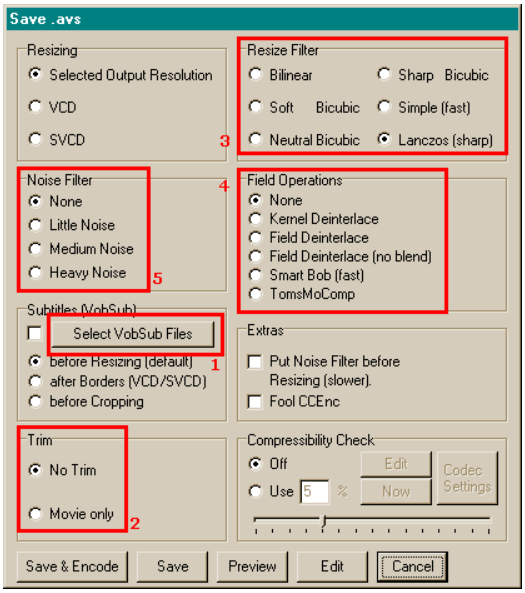


Abb. B-65: Dialog für die Filterkonfiguration.

Trim (2) kontrolliert das separate Encoding des Abspanns. Das Fenster oben gilt für Xvid. Da der Codec mit seinen 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 (Abb. B-66). 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.

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.

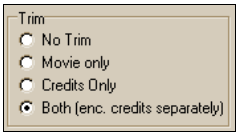


Abb. B-66: Trim-Setup für DivX.

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.

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 und Encodingsetups abschätzen lässt.

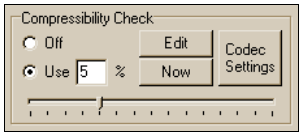


Abb. B-67: Kompressionstest-Optionen.

Mit **Codec settings** (Abb. B-67) können wir überprüfen, ob die Einstellungen für den Compcheck passen. Für DivX heißt das Modus **1-pass quality-based** mit einem **Target Quantizer** von **2**. Xvid benötigt den **Encoding type** auf **Single pass** gestellt, bei einem **Target quantizer** von **2.00**. Den Rest stellen wir so ein wie fürs Encoding des ganzen Films geplant.

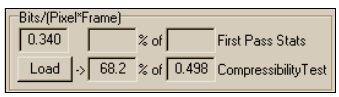


Abb. B-68: Prozentwert des Compchecks.

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 wie in Abb. B-68. Der wichtige Wert sind die **68.2** Prozent. Der sinnvolle Bereich liegt in etwa zwischen 60 und 90 Prozent. Darunter wird die Qualität schnell schlechter, darüber vergrößert sich nur die Datei ohne spürbare Steigerung der Qualität.

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-AC-3-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. Noch einmal ins Kapitel A.3.3 zur Zielauflösung zu schauen, kann an dieser Stelle nicht schaden.

100 % steht für die Sättigungsgrenze des Encoders. Wenn wir ein Encoding mit einem **Quality**-Wert von mehr als 100 Prozent starten, dürfen wir damit rechnen, dass die Zieldatei kleiner als gewollt wird.

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 können wir die Untergrenze des sinnvollen Bereichs nach unten anpassen, etwa auf 50 %. Dass das keine hässlichen Auswirkungen auf die Qualität hat, liegt an dem Kompressionsvorteil, den der Verzicht aufs Resizing mit sich bringt.

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 Ton und Bild vervollständigen.

B.4.2.6 Das Encoding starten

Es ist nun fast geschafft. Nur noch einige wenige Konfigurationsschritte trennen uns vom Start des Encodings.

Audiospuren einbinden

Mit **Select** (Abb. B-69) 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.

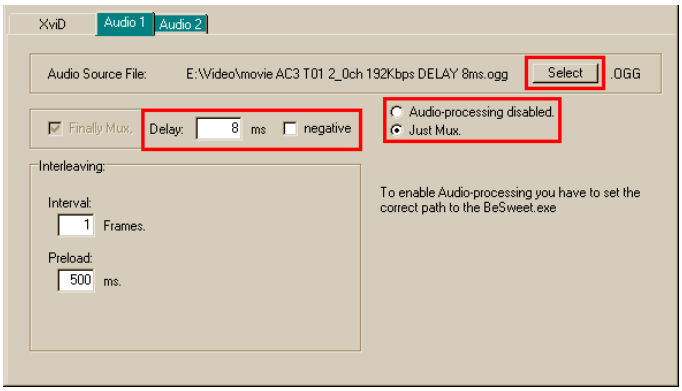


Abb. B-69: Konfiguration der Audiospuren.

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.

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 (Abb. B-70) 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 Dateigrößen 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.

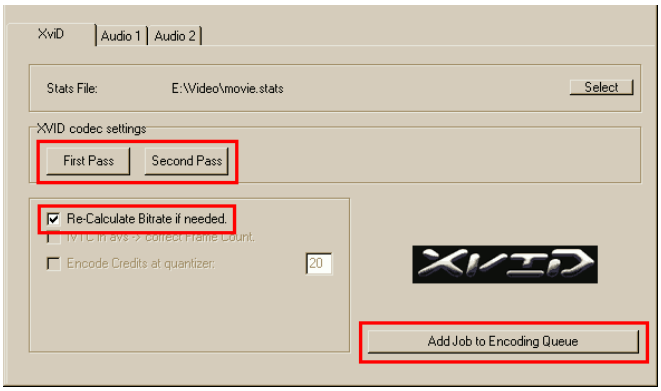


Abb. B-70: Abschließende Xvid-Konfiguration.

Für DivX sieht der Dialog ähnlich aus (Abb. B-71). Mit **Number of passes** können wir auswählen, wie viele Passes Gordian Knot durchführen soll. Nicht übertreiben, zwei sind i. d. R. genug, mehr als drei nutzlos. Auch die Codec-Konfiguration können wir noch einmal anpassen.

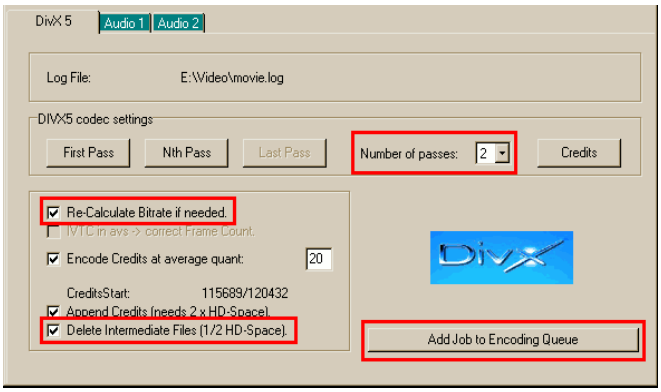


Abb. B-71: Abschließende DivX-Konfiguration.

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 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. ►A.3.2

Um das Seitenverhältnis in Xvid 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 ab Seite 88 beschrieben – das richtige PAR ein. Um Probleme zu vermeiden, sollten wir die Option für beide Passes setzen.

DivX unterstützt das Setzen des AR-Flags erst ab Version 6.5. Wer mit einer älteren Version encodiert, muss den Umweg über MPEG4Modifier gehen. Ansonsten finden wir die Einstellung im **Video**-Register der Encoder-Konfiguration, wie im Kapitel zu DivX VfW ab Seite 119 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**, wie in Abb. B-72 zu sehen.

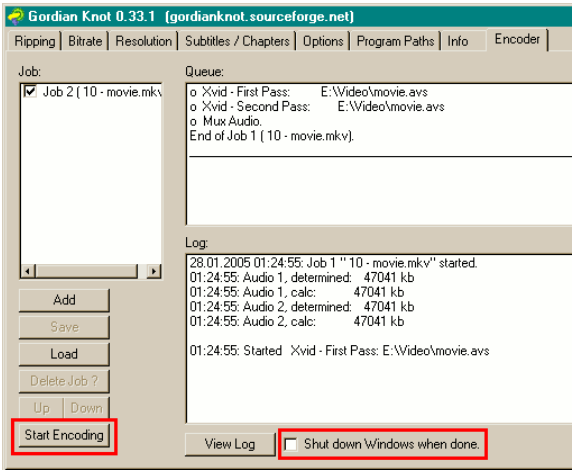


Abb. B-72: Encoder-Register von Gordian Knot.

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 Abschluss-

arbeiten: 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.4.2.7 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.
- ►A.4.4 ►A.4.2 Unterstützung für ►x264 ►CLI.
- ►A.2.3 Volle ►Matroskaunterstützung über MKVMerge.

Wir werden uns nur mit der Unterstützung für x264 CLI und MKVMerge beschäftigen, denn das sind zweifellos die beiden wichtigsten Funktionen. Gordian Knot ►A.4.2 selbst verwendet ja nur den x264 ►VfW-Encoder, 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

►[akap07] Die aktuelle Version von agkp erhalten wir im ersten Posting des ►Entwicklungsthreads im Gleitz-Forum. Die Installation erfolgt in den Ordner von VirtualDubMod. Dafür muss vorher natürlich VirtualDubMod installiert sein. Wir entpacken das komplette agkp-Archiv in den VDubMod-Ordner. Dort treffen wir fol-

gende Einstellung für alle drei Dateien `agkpv.exe`, `agkpa.exe` und `agkpAR.exe`: Mit einem Rechtsklick auf die Datei und den Menüpunkt **Eigenschaften** gelangen wir in den Eigenschaftendialog aus Abb. B-73. Im Register **Programm** stellen wir sicher, dass **Nach Beenden schließen** angehakt ist, was unter Windows XP schon von Haus aus so sein sollte.

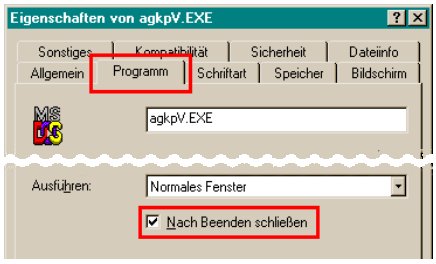


Abb. B-73: Dateieigenschaften der agkpV.exe.

Die Datei `agkp.ini` enthält die Pfade zu den benötigten externen Programmen. Die Datei können wir mit jedem Texteditor bearbeiten und 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 (Abb. B-74). 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 Gordian-Knot-Encoding durchzuführen. Dazu ändern wir die VDubMod-Einstellung wieder zurück auf die `VirtualDubMod.exe`.

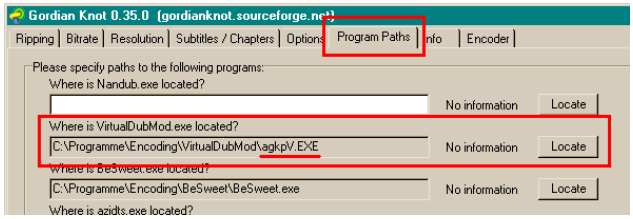


Abb. B-74: Einbinden von agkp in Gordian Knot.

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 (`http://deaththeshеep.uni.cc`), 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, ohne jedoch auf die Codec-Konfiguration zu achten. Agkp hat eigene x264-Profile und berücksichtigt die VfW-Konfiguration nicht.

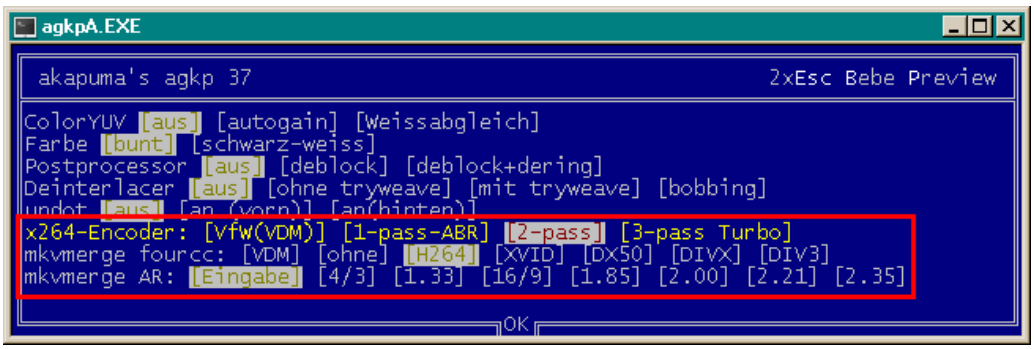


Abb. B-75: Agkp-Konfigurationsfenster.

Ist alles fertig eingestellt und das Encoding gestartet, erscheint das Auswahlfenster von agkp (Abb. B-75), 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, die wir nicht benutzen. 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, da die Option sowieso nur für x264 VfW relevant wäre.

► **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 mit der agkp-Standardfunktionalität ausdrücklich nicht. Für ein klassisches nicht-anamorphes Encoding bleibt die AR-Angabe unverändert – so wie in Abb. B-75 zu sehen.

▸A.3.2

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* (Abb. B-76) lässt sich der Fortschritt verfolgen. Das Encoding läuft nun genauso automatisch ab wie von Gordian Knot gewohnt.

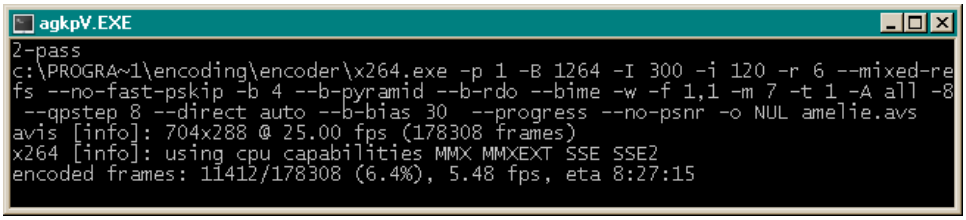


Abb. B-76: x264.exe bei der Arbeit.

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. Mögliche Werte sind ganze Zahlen zwischen 1 und 7.

▸A.4.4.2

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.

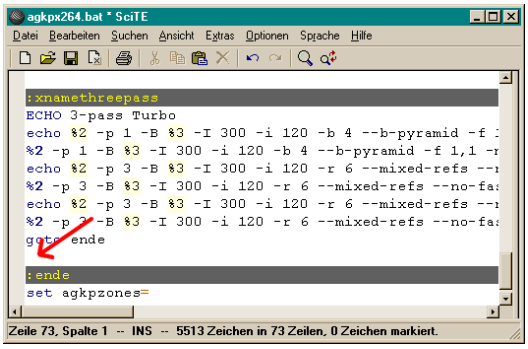


Abb. B-77: Position für neuen Encoder-Abschnitt in der agkp264.bat.

► **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.

► **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 Abb. B-77 zeigt, fügen wir z. B. 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 x264-Konfiguration können wir natürlich ganz nach Bedarf anpassen. Mehr dazu im x264-CLI-Kapitel (Seite 106 ff.). Wichtig sind die fett markierten Teile.

► **%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 (Abb. B-78) haben wir nun eine zusätzliche Option namens **Encwissen**, mit der wir die neue Konfiguration verwenden können.

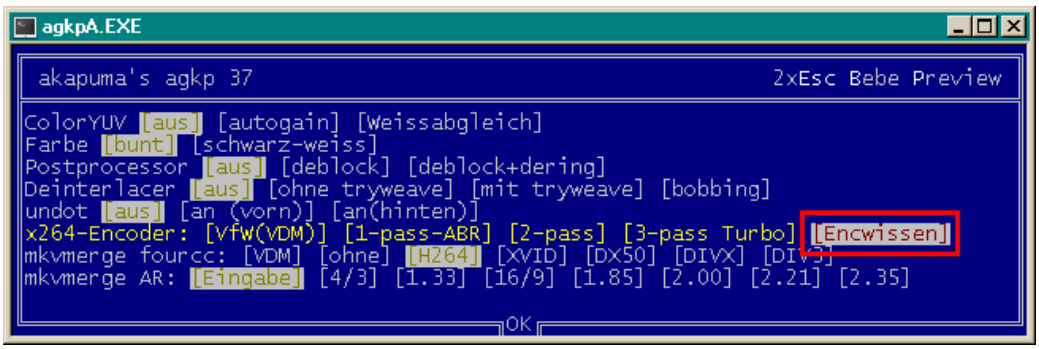


Abb. B-78: Neue Option im agkp-Auswahlfenster.

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 XvidEncraw und wird im Xvid-CLI-Kapitel ab Seite 77 näher besprochen. 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 XvidEncraw. 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 -progress 50
-bquant_ratio 162 -bquant_offset 0 -bvhq -qtype 1 -qpel -vhqmode 2
-zones 0,w,1,0
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -progress 50
-bquant_ratio 162 -bquant_offset 0 -bvhq -qtype 1 -qpel -vhqmode 2
```

```
-zones 0,w,1,0
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2
-bitrate %3 -progress 50 -overhead 0 -bquant_ratio 162 -bquant_offset 0
-bvhq -qtype 1 -qpel -vhqmode 2 -zones 0,w,1,0
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -mkv %4 -pass2 -bitrate %3
-progress 50 -overhead 0 -bquant_ratio 162 -bquant_offset 0 -bvhq
-qtype 1 -qpel -vhqmode 2 -zones 0,w,1,0
goto ende
```

Die Konfiguration ist in etwa die, wie sie in der ▸Xvid-Konfiguration erklärt ist. Natürlich können wir sie nach Belieben anpassen. ▸A.4.3.1

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

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.

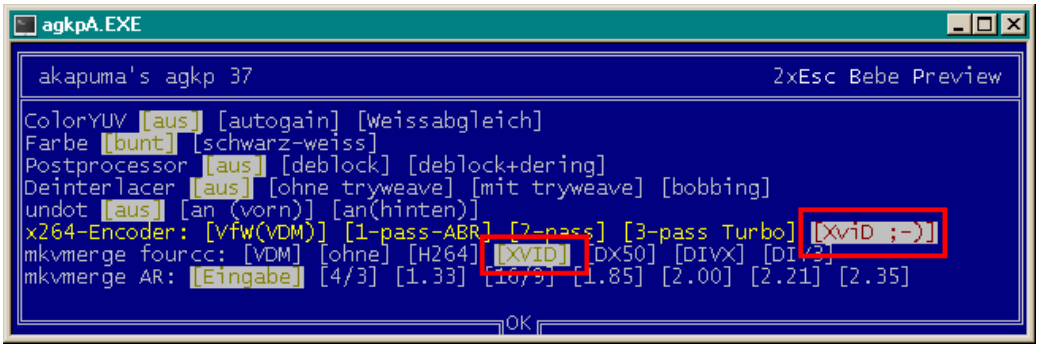


Abb. B-79: Agkp-Auswahlfenster mit Xvid-Hack.

Haben wir alles fertig konfiguriert und das Encoding gestartet, erscheint das nun schon bekannte Agkp-Auswahlfenster. Die Zeile **x264-Encoder** hat wie in Abb. B-79 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. Für **mkvmerge AR** gilt das gleiche wie im x264-Abschnitt auf Seite 194. 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! :-)

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 XvidEncraw das Setzen des MPEG-4-AR-Flags unterstützen, können wir die fehlende Funktion über die `agkp-x264.bat` nachrüsten.

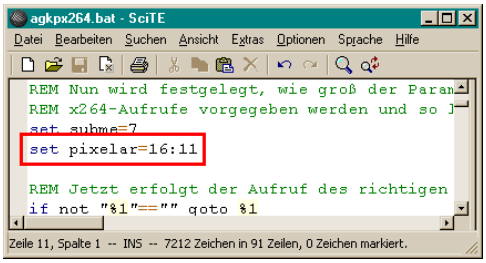


Abb. B-80: PAR-Variable in der agkpx264.bat.

Dafür öffnen wir die `agkpx264.bat` und definieren wie in Abb. B-80 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 das PAR, das ins MPEG-4-Flag geschrieben werden soll. Die `16:11` im Screenshot bezeichnet 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 47 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 XvidEncraw heißt die Option `-par %pixelar%` (einfacher Bindestrich!):

```
:xnamexvid
echo Xvid ;-)
echo "C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -par %pixelar%
-progress 50 ...
"C:\Encoder\xvid_encraw.exe" -i %5 -type 2 -pass1 -par %pixelar%
-progress 50 ...
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.

Abb. B-81: DAR-Angabe für das Matroska-AR-Flag.

Im agkp-Auswahlfenster setzen wir wie in Abb. B-81 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, allerdings unterstützen weder StaxRip noch Gordian Knot sämtliche Spezialitäten sämtlicher Container. Deswegen kann es manchmal nötig werden, selbst Hand anzulegen.

Normalerweise ist es kein Problem, das Muxing erst einmal dem Frontend zu überlassen, dann die fertige Datei in das passende, voll ausgestattete, Muxingprogramm für den gewünschten Container zu laden und alles das zu ändern und zu ergänzen, was notwendig ist. Oft ist das sogar weniger aufwändig als aus sämtlichen Einzelteilen den Film komplett neu zusammenzubauen.

B.5.1 MKVMerge für den Matroska-Container

MKVToolnix ist ein Toolpaket für den Matroska-Container. Enthalten sind u. a. MKVInfo (zeigt detaillierte Informationen über eine Matroska-Datei), MKVExtract (Demuxing von Spuren aus einer Matroska-Datei) und der für uns wichtige Muxer MKVMerge.

Spuren laden und konfigurieren

MKVMerge ist genauso wie BeSweet eine Konsolenanwendung. Die grafische Oberfläche dazu nennt sich MKVMerge GUI (kurz *mmg*). Die starten wir, landen im **Input**-Register und laden hier per Drag & Drop oder mit dem **Add**-Button sämtliche Video-, Audio- und Untertiteldateien, die im fertigen Film enthalten sein sollen (Abb. B-82). Die Spuren sind am Ende in der gleichen Reihenfolge in der Matroska-Datei gespeichert wie sie unter **Tracks** erscheinen. Mit **Up**

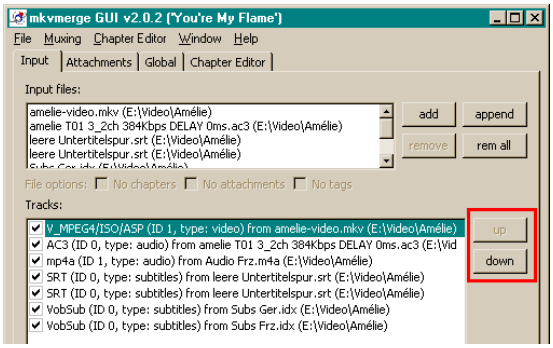


Abb. B-82: Quelldateien und Datenspuren in MKVMerge GUI.

und **Down** können wir Spuren verschieben. Dabei sollte zuerst das Video stehen, dann die Audiospuren und zum Schluss die Untertitel.

Im unteren Bereich des MKVMerge-Fensters konfigurieren wir die einzelnen Spuren, und zwar immer gerade diejenige, die in der **Tracks**-Liste angewählt ist. Im Register **General track options** (Abb. B-83) sind besonders die ersten beiden Einträge interessant. **Track name** enthält eine kurze Beschreibung der Spur, die auch in den meisten Playern angezeigt wird. Für Audio- und Videospuren habe ich mir angewöhnt, in Kurzform die technischen Daten einzutragen. Für Untertitel ist hier der richtige Platz, um die Art der Spur anzugeben; z. B. »keine Untertitel« für eine leere Spur, »nötige Untertitel« für eine Zwangsspur oder »komplett« für eine vollständige Untertitelspur.

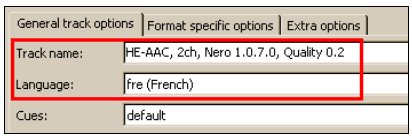


Abb. B-83: Sprache und Name für Datenspuren.

Für Tricks zum Untertitel-Muxing siehe auch das Kapitel B.3.3 ab Seite 156: *Zwangsuntertitel simulieren*.

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

Damit kommen wir zu den **Format specific options** (Abb. B-84), die wir wieder für jede Spur extra einstellen. Die Optionen sind je nach Typ der Spur aktiv oder abgeblendet und nur der Übersicht halber in aktiver Form in einen einzelnen Screenshot gepackt.

Delay hatten wir schon einmal bei BeSweet angesprochen. Um diese Anzahl Millisekunden muss die Audiospur verschoben werden, um synchron zum Video zu laufen. Steht der Wert im Dateinamen, überträgt ihn MKVMerge automatisch in das **Delay**-Feld. Ansonsten tun wir das manuell, natürlich – wenn nötig – incl. negativem Vorzeichen. Wichtig: Wenn wir beim Audio-Transcoding mit BeSweet das Delay schon einmal berücksichtigt haben, gilt jetzt *immer* ein Delay von **0**.

Abb. B-84: Spezielle Optionen für Datenspuren.

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. Der Nero-Encoder erzeugt immer MP4-AACs, d. h. wir müssen uns um den passenden Haken nicht kümmern. Winamp dagegen erzeugt immer

»nackte« AACs. Für die ist es wichtig, den Haken richtig zu setzen, da wir uns sonst Probleme beim Abspielen einhandeln.

Subtitle charset gibt den Zeichensatz für Textuntertitel an und bleibt normalerweise leer. Nur wenn wir ganz bewusst Untertitel mit einem ungewöhnlichen Zeichensatz erzeugt haben, stellen wir die passende Codierung hier ein. **Compression** gilt nur für Vobsub-Untertitel. Die lassen sich matroska-intern verlustlos komprimieren, womit wir etwa 70 % Platz sparen. Um die Kompression einzuschalten, wählen wir **zlib** aus der Combobox.

Optionen fürs Seitenverhältnis

►A.3.2

Für ►anamorphe Encodings 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. Für klassische Encodings mit quadratischen Pixeln sollten beide AR-Optionsfelder leer bleiben.

Oft erkennt MKVMerge schon automatisch das im Videostream gesetzte AR-Flag und trägt den entsprechenden Wert bei **Display width/height** ein. Nicht von seltsamen Zahlen verwirren lassen: Die hier angegebene »Darstellungsauflösung« dient nur zur Angabe des AR und hat keinen direkten Bezug zur tatsächlichen Auflösung des Videos.

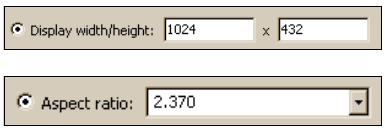


Abb. B-85: Wiedergabeauflösung und Seitenverhältnis

Eine weitere Besonderheit müssen wir für H.264 beachten. Bei solchen Videos entfernt MKVMerge das AR-Flag aus dem Videostream. Um das zu verhindern, müssen wir eine spezielle Option von Hand einfügen. Dazu wählen wir über das Menü **Muxing / Add command line options**, tragen im folgenden Dialog in der Zeile **Command line options** ein: `--engage keep_bitstream_ar_info` und bestätigen mit **OK**.

Falls wir das Seitenverhältnis manuell eintragen müssen, bietet MKVMerge zwei Möglichkeiten. Die einfachere verbirgt sich hinter der gerade erwähnten Option **Display width/height** (Abb. B-85 oben). 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:

Zielbreite = Breite nach Cropping × PAR

Die PAR-Werte stammen wie immer aus Tab. A-4 auf Seite 47. Für den üblichen Beispielfilm *Die fabelhafte 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, das DAR anzugeben, ist die Option **Aspect Ratio** (Abb. B-85 unten).

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:

$$\text{DAR} = \text{Zielbreite} / \text{Zielhöhe}.$$

Fürs *Amélie*-Beispiel bedeutet das

$$\text{DAR} = {}^{1024}\!/_{432} = 2{,}370370.$$

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.

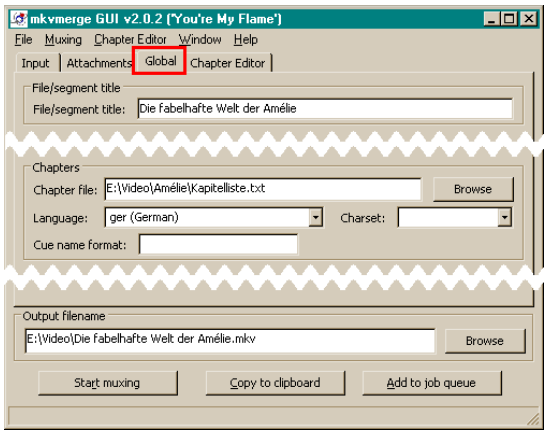


Abb. B-86: MKVMerge GUIs globale Optionen.

Einstellungen für den ganzen Film

Damit sind die Einstellungen im **Input**-Register abgeschlossen, und wir wechseln zu **Global** (Abb. B-86). **File/segment title** funktioniert ähnlich der **Track-name**-Angabe für die einzelnen Spuren, bezieht sich aber auf die gesamte Datei. Deswegen drängt es sich förmlich auf, hier den Filmtitel einzutragen. Unter **Chapter file** können wir die Datei mit den Kapitelinformationen angeben, die wir ganz am Anfang mit ChapterXtractor erstellt haben. Dieser lässt sich eine Sprache und – wie für Textuntertitel – ein Zeichensatz zuordnen.

Unter **Output filename** tragen wir den Pfad zur Zieldatei ein und können schließlich mit **Start muxing** den Multiplexing-Vorgang beginnen. Der dauert ein paar Minuten, dann liegt der Film komplett fertig in einem Stück auf der Platte. Falls wir auf mehrere CDs brennen wollen, folgt jetzt noch das Splitting. Ansonsten ist das DVD-Backup komplett.

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

Der beste Muxer für AVIs ist AVI-Mux GUI. Nicht, weil er bessere AVIs erstellen würde als VirtualDubMod, sondern weil weil er einige zusätzliche Funktionen besitzt. Zwar ist AVI-Muxing wenig aufregend, weil es kaum etwas Wichtiges einzustellen gibt, doch drei Dinge sollten wir trotzdem ansprechen.

AVI-Mux GUI kann auch Matroska muxen. Allerdings hat MKVMerge eine Ecke mehr Funktionalität und ist auch angenehmer zu bedienen.

AVI-Mux GUI kann im Gegensatz zu VirtualDubMod mit AAC-Audio umgehen. Das ist zwar technologisch eine interessante Funktion, praktisch aber nutzlos. Hardwareplayer spielen kein AAC in AVI ab, und am Computer gibt es nun wirklich sinnvollere Container. Wenn es denn doch unbedingt sein soll: AVI-Mux GUI versteht keine MP4-AACs, wie sie der Nero-Encoder erzeugt, sondern erwartet sie im containerlosen ADTS-Format. Ein Demuxing ist mit MP4Box und dieser Kommandozeile möglich:

```
MP4Box.exe -raw 1 "C:\Pfad zur\Audiodatei.mp4"
```

Wer mit Winamp encodiert, ist von diesem Problem nicht betroffen, denn Winamp erstellt ADTS-AACs, die AVI-Mux GUI direkt lesen kann.

SubRip-Textuntertitel können wir mit AVI-Mux GUI in AVI unterbringen. Dieser Trick funktioniert mit keinem anderen Programm und ist auch nur sehr bedingt sinnvoll, denn die populären Player VLC und mplayer unterstützen solche Untertitelspuren nicht.

Es existieren zwei Versionen des AVI-Standards. Die ursprüngliche Spezifikation (oft *AVI 1.0* genannt) beschränkt die Dateigröße auf 2 GB. Diese Beschränkung

existiert mit OpenDML-AVI (auch *AVI 2.0*) nicht mehr. Wenn wir AVIs größer als 2 GB muxen wollen, öffnen wir vorher AVI-Mux GUIs **Settings** und stellen unter **AVI file structure** sicher, dass der Haken bei **use Open-DML output** gesetzt ist. Das Ergebnis darf dann nahezu beliebig groß werden. Und zumindest am Computer müssen wir uns um Wiedergabeprobleme auch keine Gedanken machen.

Teil C
Spezialthemen

Einleitung

In den Spezialthemen geht es um Dinge, die nur unter ungewöhnlichen Voraussetzungen gebraucht werden oder die an anderer Stelle den Rahmen sprengen würden. Hier gibt es auch keine Reihenfolge wie sonst im Encodingwissen.

Die Spezialthemen sind in einen praktischen und einen theoretischen Abschnitt unterteilt. Wie unschwer zu erkennen ist, dreht sich der Praxisteil um ganz konkrete Vorgehensweisen beim Encoding; der Theorieteil ergänzt das Hintergrundwissen.

C.1 Praktisches

C.1.1 Manuelles Splitting

Mit dem Siegeszug von DVD-Brennern und höchst geräumigen Festplatten ist die lästigste Arbeit beim DVD-Backup – das Splitting – immer seltener nötig. Vom gesteigerten Komfort beim Anschauen und dank massig Platz der besseren Qualität ganz abgesehen. Auch Standalone-Player sollten heutzutage mit MPEG-4-Filmen auf DVD anstatt CD zurechtkommen. Leider ist die CD als Filmspeicher noch nicht ausgestorben, weshalb das Splitting-Kapitel nach wie vor seine Berechtigung hat.

Grundsätzliches zum Splitting

2-CD-Encodings müssen wir nach dem Encoding noch auf die CDs aufteilen. Manchmal sind vielleicht auch drei CDs nötig, dann splitten wir eben einmal mehr. Vom Vorgehen her macht das keinen Unterschied. 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. Alle besprochenen Encoder (Xvid/DivX/x264) speichern kaum komplette Bilder, sondern nur die Veränderung zum vorhergehenden Bild. Allein die Keyframes stellen ▸vollständige Einzelbilder dar. 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.

►A.1.2.1

► 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

Für das Aufteilen ist entweder VirtualDubMod allein oder im Zusammenspiel mit AVI-Mux GUI bzw. MKVMerge zuständig. Für diese beiden Tools suchen wir dann mit VirtualDubMod nur eine passende Stelle zum Schneiden, anstatt damit auch gleich die einzelnen Dateien zu erzeugen.

Dieses Vorgehen funktioniert gut für das klassische AVI-Encoding. Für alles, was darüber hinaus geht, ist VDubMod tendenziell unbrauchbar. Etwas genauer funktionieren folgende Dinge nicht:

- MP4-Container,
- Matroska-Container mit nativem Xvid-Stream (wie ihn XvidEncraw erzeugt) oder x264-Video,
- dynamische Untertitel in AVI.
- AAC-Audio ist potenziell problematisch.
- Matroska-Metadaten gehen evtl. verloren, da VDubMods Matroska-Unterstützung uralt und extrem eingeschränkt ist.

In diesen Fällen müssen wir ganz oder teilweise auf VDubMod verzichten und uns entweder auf die Splitting-Automatik des jeweiligen Muxingtools verlassen oder als Alternative Avidemux (`www.avidemux.org`) ausprobieren.

Falls denn das VDubMod-Splitting funktioniert, läuft es folgendermaßen ab: 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.

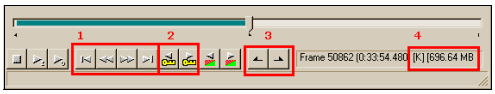


Abb. C-1: VDubs Kontrollbuttons

Abb. C-1 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 vielen MByte 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.

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-Encodings halten ein Hindernis bereit, wenn wir B-Frames ohne Packed Bitstream 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 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.

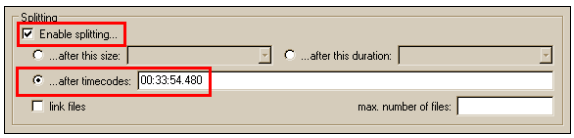


Abb. C-2: Splitting-Kontrollen in MKVMerge.

Über **Edit / Move to Selection End** springen wir zurück zum Ende der ersten CD und setzen jetzt hier den **Auswahl-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 Abb. C-2 in MKVMerge GUI ein.

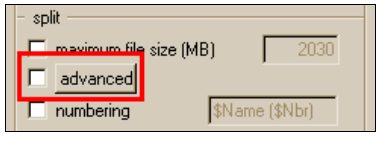


Abb. C-3: Splitting-Konfiguration von Avi-Mux GUI.

Enable splitting muss angehakt sein, um die Splittingfunktion 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.

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 passende Eingabe bei **after timecodes** lauten: **00:33:54.480,01:00:00.520**.

Unter **max. number of files** tragen wir schließlich die nötige Anzahl CDs ein und erzeugen mit **Start muxing** die endgültigen Dateien. Damit ist der Film komplett fertig.

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.

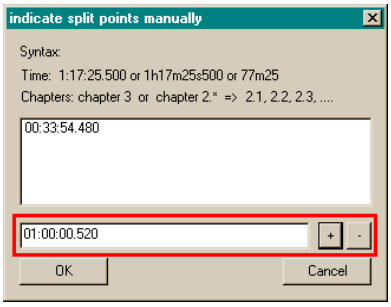


Abb. C-4: Splitpunkte in Avi-Mux GUI setzen.

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 (Abb. C-3). Im folgenden Fenster (Abb. C-4) 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 VDubMod 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.

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.1.2 StaxRip für neue Quellformate fit machen

Dieses Kapitel ist auf Anregung von Deinorius aus dem Gleitz-Forum entstanden. Da es nicht direkt mit dem DVD-Backup zu tun hat, steht es hier bei den Spezialitäten. StaxRip ist sehr flexibel, was die Quellformate angeht, die man ihm füttern kann. Das liegt daran, dass wir selbst definieren können, welches Quellformat wie vorbereitet werden muss, um dann weiterverarbeitet werden zu können. Hauptsächlich geht es dabei ums Indexieren der Videospur.

Vorbereitung

Wir kennen diese Automatik schon, denn genau sie tritt in Aktion, wenn wir VOB-Dateien als Quelle angeben. Anhand der Quell-Dateiendung erkennt StaxRip, welche Aktionen durchgeführt werden müssen. Als Beispiel werden wir eine Aktionsliste definieren, um eine H.264-Videospur aus einer MP4-Datei zu demuxen und diese anschließend mit DGAVCIndex zu indexieren. Für viele andere Quellcontainer – v. a. Matroska und TS – bringt StaxRip schon vordefinierte Aktionslisten mit. Folgende Software benötigen wir.

► **DGAVCDec** ist das Gegenstück zu DGMPGDec und ist im Gegensatz dazu nicht aufs Indexieren von MPEG-2, sondern auf H.264 ausgelegt.
`http://neuron2.net/dgavcdec/dgavcdec.html`
► **MP4Box** dient zum Demuxen aus dem MP4-Container und gehört zum normalen ▸Softwareumfang für ein DVD-Encoding. ►B.1.1

Beide Programme können wir problemlos mit StaxRips Download-Funktionalität unter **Tools / External Applications** installieren.

Erstellen von Aktionslisten

Quellformat-Aktionen sind an ein Projekt gebunden; es handelt sich nicht um globale Einstellungen. Das ist auch sinnvoll, denn wenn wir z. B. aus MP4 nicht nur H.264 sondern auch Xvid demuxen wollen, müssen wir dafür unterschiedliche Projekte erstellen. Da StaxRip Quelldateien anhand ihrer Dateiendung erkennt, ist pro Endung auch nur eine Aktionsliste möglich. Wie wir allgemein Projekte definieren und als Vorlagen speichern, haben wir schon ab Seite 161 im Ka-

pitel B.4.1.1 zur StaxRip-Konfiguration besprochen.

In den Projektoptionen unter **View / Options** (oder über **F8**) sehen wir im Abschnitt **Preparation** die Aktionslisten für einige übliche Quellformate (vgl. Abb. C-5). Über den Button **Add** fügen wir eine neue Liste hinzu. Als Vorlage wählen wir den **Command Line Preparer**, was uns ins Dialogfenster von Abb. C-6 bringt.

Bei **Input** tragen wir die Dateiendung (ohne Punkt) des gewünschten Quellformats ein: `mp4` in unserem Fall. Mehrere Endungen trennen wir durch ein Komma.

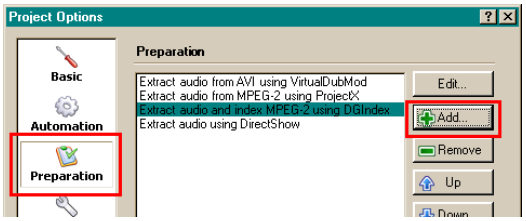


Abb. C-5: Projektoptionen.

Hinter **Output Video** steht die Dateiendung der erzeugten Videodatei. DGAVCIndex erzeugt Dateien mit der Endung `dga`, die den D2V-Dateien von DGIndex entsprechen. **Output Audio** enthält analog die Dateiendungen der erzeugten Audiospuren. Leider bietet MP4Box keine Möglichkeit, um alle vorhandenen Audiospuren zu extrahieren. Das müssen wir also doch noch manuell erledigen. Entsprechend bleibt das Feld leer.

Die eigentlichen Programmaufrufe tragen wir in einem zusätzlichen Fenster ein, das sich mit einem Klick auf den **a**-Button hinter dem Feld **Command Lines** öffnet. Dort tragen wir im großen Feld des Registers **Edit** folgende zwei Zeilen ein.

```
"%application:MP4Box%" -raw 1 "%source_file%"

"%application:DGAVCIndex%" -i "%source_dir%\%source_name%.h264"
-o "%source_dir%\%source_name%.dga" -e
```

Damit weiß StaxRip: Aus Quelldateien mit der Endung MP4 soll zuerst mit dem MP4Box-Aufruf das Video als rohe AVC-Videospur extrahiert werden. Die wird dann an DGAVCIndex verfüttert, um eine DGA-Indexdatei zu erstellen. Der erste Schritt ist nötig, da DGAVCDec den MP4-Container noch nicht unterstützt. Um neue Entwicklungen und aktuelle Einschränkungen nicht zu verpassen, sollten wir unbedingt im Doom9.org-AVC-Forum den Entwicklungsthread namens ▸*DGAVCDec 1.0.0 Alpha* verfolgen.

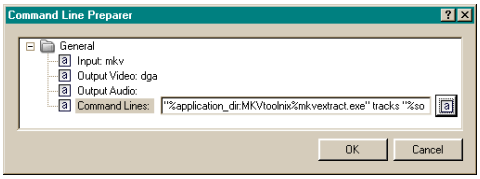


Abb. C-6: Dialog für neue Aktionsliste.

▸ [neur07]

Wir bestätigen alle Dialoge, bis wir uns wieder im Fenster **Project Options** befinden. Hier können wir über **Rename** unserer neuen Aktionsliste einen sinnvollen Namen geben. Außerdem müssen wir über **Edit** aus allen anderen Listen die

Endung MP4 entfernen.

Ein weiterer Klick auf **OK** bringt uns dann zurück ins StaxRip-Hauptfenster, wo wir über **File / Save Project as Template** unser Projekt inklusive neuer Aktionsliste abspeichern. Wenn wir später eine MP4-Datei mit H.264-Video verarbeiten wollen, wählen wir zuerst unter **File / Template Projects** die gerade gespeicherte Vorlage und öffnen erst dann die Quelldatei.

Auf die gleiche Weise können wir Aktionslisten für beliebige Formate anlegen. Der Trick an Aktionslisten ist der, die nötigen Programme und Kommandozeilen zu kennen. Die geben wir dann ähnlich ein, wie wir sie auch an der Konsole eintippen und ausführen würden. Nur Dateinamen und Programmpfade werden durch in Prozentzeichen (%) eingeschlossene StaxRip-Makros ersetzt. Eine Liste dafür erhalten wir im Editorfenster mit der **F1**-Taste.

C.2 Theoretisches

C.2.1 Zielauflösung, Mod16-Regel & Co.

Manchmal würden wir uns deutlich leichter tun, könnten wir die Zielauflösung des Backups ohne Einschränkungen festlegen. Das gilt ganz besonders fürs anamorphe Encoding. Leider steht dem die allgemeine Regel entgegen, dass die Zielauflösung in beiden Dimensionen glatt durch 16 teilbar sein soll. Woher diese Einschränkung kommt, warum sie sinnvoll ist und wann wir doch mit gutem Grund davon abweichen können, das betrachten wir in diesem Kapitel genauer.

Grund der Auflösungseinschränkung

►A.1.2.1

Das Bild der DVD verwendet wie das fertige Encoding den ►Farbraum YV12, der nur mit Auflösungen kompatibel ist, die horizontal und vertikal jeweils glatt durch 2 teilbar sind. AviSynth setzt ein weiteres Limit oben drauf, das für die Praxis letztendlich relevant ist: horizontal mod4, vertikal mod2. Das wäre noch keine besonders erwähnenswerte Einschränkung, doch einen Schritt weiter im Backupprozess kommt der Encoder ins Spiel. Die Bewegungssuche und Bewegungskompensation aller MPEG-4-Encoder arbeitet nicht mit einzelnen Pixeln, um die Rechenzeit in einem erträglichen Rahmen zu halten. Stattdessen kommen Makroblocks zum Einsatz, die eine Fläche von 16×16 Pixeln zu einer Einheit zusammenfassen. Für einen Encoder besteht das Bild also aus in Zeilen und Spalten angeordneten Makroblocks. Einzelne Pixel spielen nur eine untergeordnete Rolle.

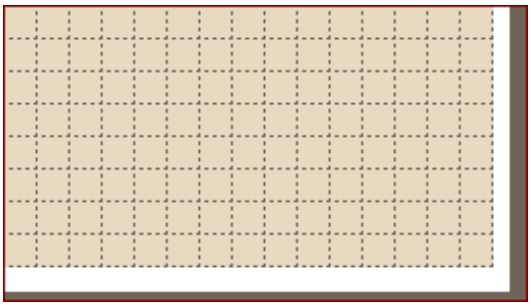


Abb. C-7: Schematisches Nicht-Mod16-Videoframe.

Solange das Bild horizontal und vertikal durch 16 teilbar ist (Mod16-Kriterium), funktioniert das auch bestens. Schwierigkeiten tauchen erst bei Nicht-Mod16-Auflösungen auf, wie in Abb. C-7 zu sehen. Schematisch dargestellt ist ein Einzelbild mit einer Auflösung von 248×140 Pixeln, das die rot und weiß markierten Bereiche umfasst. In der Breite passen so 15 vollständige Makroblocks nebeneinander (rot). Außerdem bleiben am rechten Rand 8 überschüssige Pixel stehen (weiß), die keinen kompletten Block mehr ergeben. In der Höhe sieht die Si-

tuation ähnlich aus mit 8 vollständigen Blocks und 12 überschüssigen Pixeln.

Für den weißen Überschussbereich muss sich der Encoder etwas einfallen lassen, da er mit unvollständigen Makroblocks nicht arbeiten kann. Die Lösung besteht darin, die Auflösung intern auf den nächsten vollen Makroblock zu vergrößern (256×144) und das neue Stück (grauer Bereich) mit Pseudo-Bildinformationen aufzufüllen. Beim Abspielen wird das erweiterte Stück Bild vom Decoder wieder abgeschnitten, so dass man beim Anschauen von der ganzen Sache nichts mitbekommt.

Mod16-Regel und Alternativen

Damit dürfte klar sein, woher die Mod16-Regel stammt. Wir ersparen dem Encoder das Erweitern und Auffüllen des Bildes und verhindern so einegeringere Encodingeffizienz, was im 1-Pass-Verfahren zu einer größeren Datei und bei 2-Pass zu niedrigerer Qualität führen würde. Schließlich muss bei Nicht-Mod16-Auflösungen intern ein größeres Bild encodiert werden, was unweigerlich einen kleinen Teil der Bitrate für die Codierung unnützer Informationen verbrät.

Übrigens: Die Bezeichnung »Mod16« stammt von der mathematischen Modulo-Funktion, die oft mit mod abgekürzt wird und mit der man auf die glatte Teilbarkeit einer Zahl testen kann.

Für ein klassisches Encoding mit quadratischen Pixeln stellt mod16 kein Problem dar. Mit Cropping und Resizing haben wir genug Spielraum, um die Bildverzerrung wegen der eingeschränkten Auflösungswahl vernachlässigbar klein zu halten. Kritischer ist anamorphes MPEG-4, wo das Resizing wegfällt. Oft bringt uns das in die Situation, beim Cropping entweder

- recht weit ins eigentliche Bild hineinschneiden oder
- ein Stück schwarzen Balkens stehen lassen zu müssen.

Beides ist schlecht. Im gar nicht so seltenen ungünstigen Fall verlieren wir an jedem Rand zehn oder mehr Pixel, womit durchaus 5 % oder mehr an Bildfläche verloren gehen können. Auch wenn ganz am Bildrand selten wichtige Dinge passieren, ist das ein zu hoher Wert. Schwarze Balken stehen zu lassen, macht die Sache nicht besser. Zwar bleibt das Bild vollständig intakt, allerdings kostet die harte Kante zwischen Bild und Balken spürbar Encodingeffizienz – und das heißt sinkende Qualität oder eine größere Zieldatei(vgl. das Kapitel zum Cropping,

Seite 41).

Um uns aus dieser Zwickmühle zu befreien, schneiden wir doch ins Bild hinein, beschränken das verlorene Stück aber auf ein paar unkritische wenige Pixelreihen, indem wir das Mod16-Kriterium aufweichen. D. h. wir wählen eine Auflösung, die in einer oder beiden Dimensionen nicht mehr glatt durch 16 teilbar ist.

Zwar drückt eine Nicht-Mod16-Auflösung auf die Effizienz des Encoders, allerdings weniger als würden wir ein Stück Balken stehen lassen. Der Encoder füllt das intern erweiterte Bild nämlich nicht mit schwarz auf, eben wegen des Problems des harten Übergangs zum eigentlichen Bild. Die Füllung besteht stattdessen aus Pseudo-Bild. Die einfachste und am wenigsten rechenintensive Methode ist, die letzte Pixelreihe des echten Bildes zu wiederholen. Damit hat der Encoder den harten Übergang vermieden und mit ein wenig Glück sogar Informationen generiert, die sich für eine effiziente Bewegungssuche gut eignen.

Alles in allem ist nicht-mod16 das geringere Übel als überstehende Balken. Das heißt auch, dass wir ganz genau darauf achten müssen, wirklich sämtliche Balken vollständig abzuschneiden, damit der Encoder beim internen Auffüllen den Balken nicht wieder herstellt.

Höhe des Effizienzverlusts

Um den Effizienzverlust möglichst gering zu halten, sollten wir eine Auflösung wählen, die so dicht wie möglich *unter* Mod16 liegt. Denn dann muss der Encoder nur wenige Zeilen ergänzen und der Effizienzverlust bleibt gering. Theoretisch jedenfalls. Praktisch besteht ein deutlicher Unterschied zwischen ASP-Encodern (Xvid, DivX) und AVC-Encodern (x264).

Abb. C-8 stellt den Bereich zwischen zwei benachbarten Mod16-Auflösungen dar; links die kleinere, rechts die größere. Der farbige Balken zeigt den Effizienzverlust für **MPEG-4 ASP**. Grün steht für einen geringen Verlust, gelb für einen noch akzeptablen und rot für einen inakzeptabel großen Verlust.

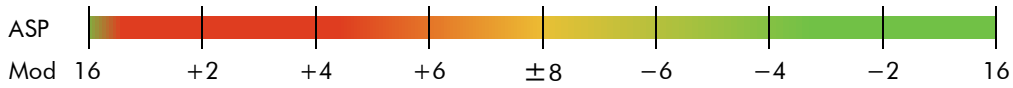


Abb. C-8: ASP-Effizienz.

Wir sehen deutlich, dass die Theorie stimmt. Je dichter unter Mod16 die Auflösung liegt, desto weniger Effizienz geht verloren. Das Verhalten ist mit Xvid ►[BrJo07a] ausführlich getestet. Auch ist davon auszugehen, dass sich DivX nicht wesentlich anders verhält. Drei Punkte merken wir uns:

► Mod16 sollten wir bevorzugen.
► Eine Unterscheidung zwischen Mod16–2 und Mod16–4 ist nicht möglich. Man kann nicht sagen, dass eine davon klar besser wäre. Beide können wir bedenkenlos verwenden, da der Effizienzverlust gering bleibt.
► Eine sinnvolle Grenze bildet Mod8. Der Effizienzverlust ist spürbar stärker als bei Mod16–2/4, liegt aber noch im Rahmen. Noch weiter links wird der Verlust zu groß. D. h. die acht Pixel *über* einer Mod16-Auflösung sollten wir vermeiden, die acht Pixel *unter* einer Mod16-Auflösung sind dagegen akzeptabel.

Die Situation für **MPEG-4 AVC (H.264)** sieht etwas anders aus. Wie oben stellt Abb. C-9 den Bereich zwischen zwei benachbarten Mod16-Auflösungen dar.

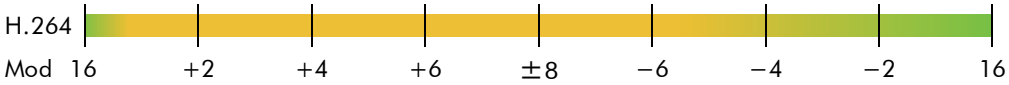


Abb. C-9: AVC-Effizienz.

Die Verteilung der Farben ergibt sich aus einem ►Test mit x264. Die Ergebnisse waren dabei viel weniger klar und konsistent als mit Xvid. Trotzdem können wir Folgendes festhalten: ► [BrJo08b]

► Mod16 sollten wir grundsätzlich bevorzugen.
► Mod16–2 ist meistens nur wenig schlechter als Mod16 und kann bedenkenlos verwendet werden.
► Die anderen Varianten folgen mit etwas Abstand, sind aber allesamt akzeptabel. Eine klare Reihenfolge lässt sich nicht bilden.

Der Effizienzverlust fällt insgesamt geringer aus als bei Xvid. Deswegen gibt es auch keine Auflösung, die wir auf jeden Fall meiden sollten.

C.2.2 Die Bedeutung der ITU-R BT.601 für das PAR

Weitere Literatur: [Aho08] [Aho04] [BrJo06a] [BrJo06b] [ITU07a]

Das Anamorph-Kapitel A.3.2 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.

Ich möchte behaupten, dass dieses Kapitel mit Abstand die schwerste Kost im Encodingwissen ist. Den Überblick über die Seitenverhältnis-Thematik mit allen

ihren Ausläufern zu behalten, ist eine echte Herausforderung. Um das ein wenig zu erleichtern, der Inhalt dieses Kapitels in drei Stichpunkten zusammengefasst:

► Die Norm namens ITU-R BT.601 (auch »Rec. 601«, früher »CCIR 601«) ist *der* zentrale Baustein für das Seitenverhältnis der DVD. Sie legt fest, dass das eigentliche Videobild ein wenig kleiner sein muss als die gesamte DVD-Auflösung.
► Computerpixel sind quadratisch, DVD-Pixel rechteckig. Dieser Unterschied muss fürs Encoding berücksichtigt werden.
► Es gibt insgesamt drei Varianten, das DVD-Seitenverhältnis zu berechnen. Auf der einzelnen DVD ist nirgendwo vermerkt, welche Variante bei der Produktion verwendet wurde.

Analog/Digital-Wandlung

Die von der ITU-Organisation, Abteilung »Radiocommunication«, herausgegebene Empfehlung Nummer BT.601, die so viel Verwirrung stiftet, hat die Konvertierung analogen Fernsehmaterials in ein digitales Format und wieder zurück 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 und dessen Normen eine entscheidende Rolle, denn

► wichtig für die Seitenverhältnisse ist die gesamte Verarbeitungskette vom ursprünglichen Quellmaterial bis zur Wiedergabe,
► die ITU-R BT.601 ist eine Art Naturgesetz der professionellen Videowelt.

Beginnen wir deshalb ganz vorn bei der Umwandlung von analog nach digital. Das analoge Fernsehbild wird von einem Elektronenstrahl aufgebaut, der wie in Abb. C-10 über den Monitor streicht. Der Elektronenstrahl setzt oben in der Mitte an und zeichnet die erste Zeile (rote durchgezogene Linie), wird abgeschaltet und zum Anfang der nächsten Zeile versetzt (graue gestrichelte Linie), wo er wieder aktiv wird und die nächste Zeile zeichnet. Das geht immer so weiter, bis ganz unten in der Mitte das Ende erreicht ist.

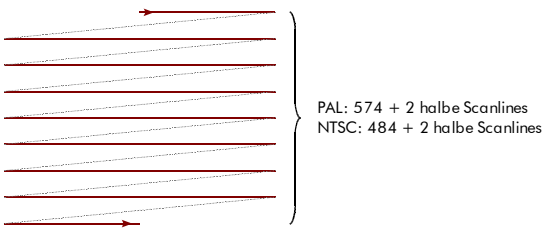


Abb. C-10: Vereinfachte Funktion eines analogen Monitors.

Die europäische Fernsehformat PAL arbeitet mit 574 ganzen Zeilen (Scan-

lines). Dazu kommt die halbe Scanline oben und die halbe unten. Das amerikanische NTSC funktioniert genauso mit 484 vollständigen Scanlines.

Diese Darstellung ist vereinfacht und entspricht nicht ganz der Realität. Ein analoges Bild ist immer interlaced, d. h. es setzt sich aus zwei Halbbildern zusammen, die unabhängig voneinander gezeichnet werden. Deswegen bitte Abb. C-10 nicht als die exakte praktische Arbeitsweise eines TV-Monitors missverstehen!

Vertikale A/D-Wandlung

Für die digitale Welt benötigen wir diskrete, d. h. abzählbare, Werte, die sich als eine Anzahl von Pixeln darstellen lassen. In der vertikalen Dimension haben wir mit den Scanlines schon analog eine solche Unterteilung. Es liegt nahe, Zeilenanzahl mit Pixelanzahl gleichzusetzen, und genau so erfolgt auch die vertikale Analog/Digital-Konvertierung.

Unschön sind lediglich die beiden halben Scanlines, die nicht so recht ins digitale Konzept einer rechteckigen Auflösung passen wollen. Das wird ganz pragmatisch gelöst, indem man die halben Zeilen einfach als vollständig behandelt und die jeweils fehlende Hälfte mit Schwarz auffüllt.

Damit erhalten wir eine vertikale Pixelanzahl von 576 (574 + 2) für PAL und 486 (484 + 2) für NTSC.

Horizontale A/D-Wandlung

Die horizontale Auflösung lässt sich nicht so einfach und eindeutig wie die vertikale ermitteln, denn eine analoge Scanline ist eine Wellenform von einigen Mikrosekunden Dauer und ohne jede offensichtliche Unterteilung. 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. Die Geschwindigkeit der Abtastung heißt Samplingrate und wird in Hertz (Hz = Abtastungen pro Sekunde) gemessen. Dabei gibt es kein absolutes Richtig oder Falsch. Prinzipiell können wir eine Scanline in beliebig viele Pixel unterteilen, ob das nun 10 oder 10 000 sind.

Um ein TV-Bild sinnvoll zu digitalisieren, sind auf jeden Fall mehrere Millionen Samplingvorgänge pro Sekunde (Megahertz, MHz) nötig. Und an dieser Stelle kommt die ITU-R BT.601 ins Spiel, denn die sagt uns, dass wir eine Scanline mit 13,5 MHz abtasten sollen. Das gilt sowohl für PAL als auch NTSC. Diese einheitliche Rate vereinfacht es, universelle Hardware-A/D-Wandler zu bauen, was wohl

der Hintergedanke der ITU-Empfehlung war.

Leider passt der Wert nicht exakt auf die 720 horizontalen Pixel der DVD. Für PAL ergeben sich 702 Pixel, für NTSC 710,85. Das sehen wir uns weiter unten noch genauer an. Zuerst benötigen wir jedoch einen Ausflug in die Welt der Pixelformen.

Pixelformen

Meistens wird ein Pixel als ein festgelegtes, immer gleiches Etwas angesehen. Damit kommen wir bei DVD-Video allerdings nicht weiter. Unser Ideal sind quadratische Pixel, weil sich mit denen am schönsten umgehen lässt und sie von nahezu jeder üblichen Auflösung eines Computermonitors verwendet werden. DVD-Pixel dagegen sind niemals quadratisch, sondern rechteckig; meistens breiter als hoch. In der Praxis zwingt uns das dazu, immer darauf zu achten, mit welcher Pixelform wir es gerade zu tun haben. Versäumen wir das, ist das Ergebnis ein verzerrtes Bild. Abb. C-11 zeigt die typischen Formen eines Pixels im Vergleich.

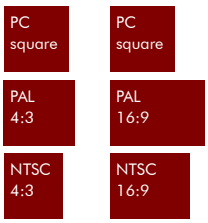


Abb. C-11: Pixelformen.

Für die Berechnungen weiter unten müssen wir das analoge Bild in eine digitale Auflösung mit unseren idealen quadratischen Pixeln umrechnen. Wir wissen, dass die DVD nur 16:9 und 4:3 als Seitenverhältnis des gesamten Bilds zulässt. Außerdem kennen wir die vertikale digitale Auflösung (PAL 576, NTSC 486). Daraus können wir ganz einfach die horizontale Pixelanzahl bei quadratischen Pixeln errechnen, ohne noch einmal auf die analoge Welt oder die ITU-Empfehlung zurückgreifen zu müssen. Wir rechnen z. B. für 4:3 PAL:

hor. Auflösung = 576 × 4/3 = 768 Pixel.

Für die anderen drei Fälle setzen wir jeweils die passenden Werte ein und erhalten die vier »quadratischen« Standardauflösungen aus Tab. C-1.

Tab. C-1: Rechnerische digitale Standardauflösungen.

	PAL	NTSC
4:3	768 × 576	648 × 486
16:9	1024 × 576	864 × 486

Damit haben wir alles Nötige, um uns in die Tiefen der Pixel-Seitenverhältnisse zu stürzen.

Anwendung der ITU-R BT.601

PAR-Berechnung für PAL

Der Elektronenstrahl eines PAL-Fernsehers benötigt 52 μs Zeit, um eine Scanline zu zeichnen. Mit der ITU-konformen Samplingrate von 13,5 MHz erhalten wir 52 μs $\times$ 13,5 MHz = 702 horizontale Pixel, was insgesamt einer aktiven digitalen Auflösung von 702 $\times$ 576 Pixeln entspricht. *Aktiv* heißt, das komplette Bild, wie es einmal im analogen Signal vorhanden war, liegt innerhalb dieser 702 Pixel.

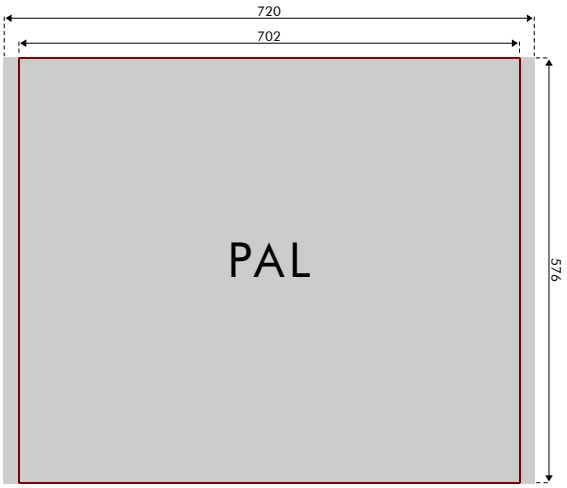


Abb. C-12: Auflösung und aktiver Bereich der PAL-DVD.

Die Auflösung einer PAL-DVD ist auf 720 $\times$ 576 Pixel festgelegt. Zwar erlaubt der Standard auch ein paar andere Varianten, die praktisch aber nur geringe Bedeutung haben. Unser aktives Bild ist also kleiner als die DVD-Auflösung. Grafisch stellt sich die Situation wie in Abb. C-12 dar. Die graue Fläche ist die DVD-Auflösung, die dünne rote Linie kennzeichnet das aktive ITU-Bild. Wir müssen nun 702 $\times$ 576 aktive Pixel in 720 $\times$ 576 vorhandenen Pixeln unterbringen. Vertikal kein Problem, da die Auflösungen exakt übereinstimmen. Horizontal dagegen bleiben 18 ungenutzte Pixel übrig. Der Einfachheit halber zentrieren wir das ITU-Bild innerhalb der DVD-Auflösung, so dass links und rechts jeweils ein 9 Pixel breiter Balken entsteht, den wir in der einfachsten Variante mit Schwarz auffüllen.

Interessant wird es bei der Wiedergabe, denn jetzt kommen die verschiedenen Pixelformen ins Spiel. Was wir nun erarbeiten wollen, lässt sich auf zwei Arten ausdrücken. Wir benötigen

- eine Angabe über die Form eines DVD-Pixels oder
- einen Multiplikator, mit dem wir eine unverzerrte Wiedergabeauflösung mit quadratischen Pixeln erzeugen können.

Das sind zwei verschiedene Sichtweisen ein und des selben Problems, d. h. es ist egal, welchen Standpunkt wir einnehmen. In jedem Fall ist der gesuchte Wert das so genannte Pixel Aspect Ratio, kurz PAR (alternativ bekannt als Sample Aspect

Ratio, SAR). Der Einfachheit halber bleiben wir beim Computermonitor mit seinen quadratischen Pixeln. In Abb. C-13 sehen wir die nötige Entzerrung für eine 16:9-PAL-DVD.

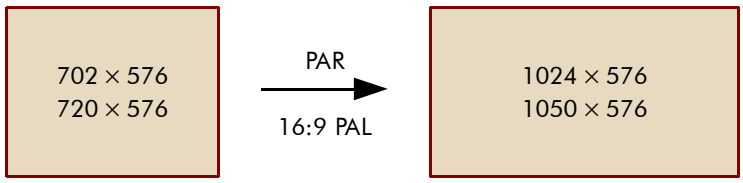


Abb. C-13: Umrechnung der 16:9-PAL-DVD in quadratische Pixel.

Wie kommen die gezeigten Werte zustande? Dafür benötigen wir unsere Tab. C-1 mit den quadratischen Auflösungen, die wir weiter oben unter »Pixelformen« berechnet haben. Daraus kennen wir für 16:9 PAL die »quadratische« Auflösung (PAR 1:1) von 1024×576 Pixeln, die das korrekt zurechtgezogene Bild zum Zeitpunkt der Wiedergabe darstellt und ein Seitenverhältnis von exakt 16:9 aufweist. Bei der 16:9-PAL-DVD ist für den aktiven Teil der Auflösung von 702×576 ebenfalls ein Seitenverhältnis von exakt 16:9 definiert; die Pixel sind also rechteckig (PAR ungleich 1:1).

Das PAR zu errechnen, ist mit diesen Informationen einfach. Wir packen dazu die Frage »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 16:9- und 4:3-Fall:

$$\text{16:9-PAR} = \frac{1024}{702} = \frac{512}{351} \approx 1{,}4587$$

$$\text{4:3-PAR} = \frac{768}{702} = \frac{128}{117} \approx 1{,}0940$$

Diese Werte können wir entsprechend den zwei Sichtweisen von oben verschieden interpretieren.

► Die Pixel einer 16:9-PAL-DVD sind ca. 1,46 mal so breit wie hoch.

► Das Bild einer 16:9-PAL-DVD muss für die unverzerrte Wiedergabe auf ca. 146 % in die Breite gezogen werden.

Beide Sichtweisen führen zu absolut identischen Ergebnissen.

Um beim Abspielen die richtige Breite zu erhalten, multiplizieren wir die DVD-Breite mit dem PAR. 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.

16:9-Breite = $720 \times {}^{512}/_{351} \approx 1050{,}26$ Pixel

4:3-Breite = $720 \times {}^{128}/_{117} \approx 787{,}69$ Pixel

Damit sollte 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 in gestreckter Form exakt die Abweichung zum Standard-DAR ergeben (19,69 bzw. 16,26 Pixel). Damit entspricht auch der *aktive Teil* der endgültigen Auflösung genau der Ausgangssituation: 1024×576 und 768×576.

PAR-Berechnung für NTSC

Die Situation für NTSC stellt sich ähnlich dar. Eine NTSC-Scanline ist $52 + {}^{59}/_{90}\ \mu$s lang und wird mit 13,5 MHz ITU-Samplingrate abgetastet. Damit erhalten wir 52,656 μs × 13,5 MHz = 710,85 horizontale Pixel. Da wir exakt rechnen wollen, verwenden wir diesen krummen Wert. Die gesamte aktive Auflösung ist also 710,85 × 486.

Genau genommen ist es egal, ob wir auf 711 aufrunden oder nicht. Als letzter Schritt vor der eigentlichen Darstellung am digitalen Bildschirm muss sowieso aufs nächste volle Pixel gerundet werden, was den minimalen rechnerischen Unterschied zwischen 710,85 und 711 ausgleicht. In der analogen Welt liegt die Rundungsdifferenz einwandfrei innerhalb der für NTSC erlaubten Toleranzgrenzen.

Für die NTSC-DVD gilt eine Standardauflösung von 720 × 480. Horizontal besteht also kein Unterschied zu PAL, denn 710,85 aktive Pixel lassen sich problemlos in 720 vorhandenen Pixeln unterbringen. Unschöner sind die 480 vertikalen DVD-Pixel, die für die 486 aktiven Pixel nicht ausreichen. Wir stehen grafisch also vor der Situation aus Abb. C-14, die auf den ersten Blick doch spürbar komplizierter als bei PAL aussieht. Die graue Fläche stellt wieder die DVD-Auflösung dar, der rote Rahmen den aktiven ITU-Bereich. Prinzipiell wäre es möglich, vertikal ähnlich zu verfahren wie horizontal, also 486 tatsächliche Pixel in 480 DVD-Pixel zu quetschen und zusätzlich zum horizontalen mit einem vertikalen PAR zu ar-

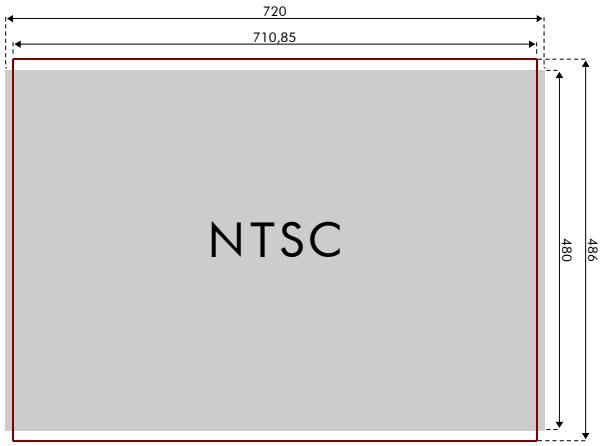


Abb. C-14: Auflösung und aktiver Bereich der NTSC-DVD.

beiten. Das ist genauso lästig wie es klingt, deswegen hat man sich für ein einfacheres Verfahren entschieden. Was nicht passt, wird passend gemacht! Wir schneiden ganz einfach oben und unten je drei Pixel ab und gut ist.

Natürlich bringt das Cropping eine minimale Abweichung des DAR mit sich, die aber fürs Abspielen unwichtig ist. Die digitale Wiedergabe ist zu jedem beliebigen Seitenverhältnis fähig und am analogen Fernseher müssen während der Digital/Analog-Wandlung so oder so die fehlenden Zeilen ergänzt werden, um ein gültiges NTSC-Signal zu erzeugen.

Bei allen Rechnungen daran denken: Die in der Höhe abgeschnittenen Pixelzeilen führen dazu, dass die aktive Auflösung einer NTSC-DVD niemals 16:9 bzw. 4:3 ist, sondern 9:5 bzw. 27:20.

Damit befinden wir uns tatsächlich in einer Situation, die derjenigen bei PAL entspricht. Die vertikale Auflösung stimmt und horizontal berechnen wir das PAR als Verzerrungsfaktor. Sehen wir uns die nötige Entzerrung in Abb. C-15 an.

Abb. C-15: Umrechnung der 16:9-NTSC-DVD in quadratische Pixel.

Aus Tab. C-1 lesen wir die benötigte »quadratische« Auflösung ab: 864×486. Die Abweichung zur vertikalen Auflösung spielt an dieser Stelle keine Rolle, und wir berechnen genauso wie bei PAL die PAR-Werte.

$$\text{16:9-PAR} = \frac{864}{710{,}85} = \frac{5760}{4739} \approx 1{,}2154$$

$$\text{4:3-PAR} = \frac{648}{710{,}85} = \frac{4320}{4739} \approx 0{,}9116$$

Beim Abspielen multiplizieren wir wieder die horizontale DVD-Auflösung mit dem PAR.

$$\text{16:9-Breite} = 720 \times {}^{5760}\!/_{4739} \approx 875{,}12 \text{ Pixel}$$
$$\text{4:3-Breite} = 720 \times {}^{4320}\!/_{4739} \approx 656{,}34 \text{ Pixel}$$

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 unterscheiden 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. Für die korrekte Wiedergabe müssen wir insgesamt vier PAR-Werte im Kopf behalten; siehe Tab. C-2.

	PAL	NTSC
4:3	128/117	4320/4739
16:9	512/351	5760/4739

Tab. C-2: Exaktes PAR nach ITU-R BT.601.

Wem einmal die PARs 72/79 (4:3) und 96/79 (16:9) über den Weg laufen, das sind die NTSC-Werte gerechnet mit 711 aktiven Pixeln.

Die PAR-Tabelle nach MPEG-4

Der aufmerksame Leser wundert sich bestimmt schon längst wieder. Die ITU-PAR-Tabelle 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

Tab. C-3: PAR nach MPEG-4.

Die Zahlen entstammen dem MPEG-4-Standard. Im H.264-Dokument stehen sie in Tabelle E-1. MPEG-4 Visual definiert die gleichen Werte in Tabelle 6-12. So unterschiedlich die Brüche auf den ersten Blick anmuten: wenn man sie ausdividiert, kommt bei ITU- und MPEG-Tabelle fast das Gleiche heraus. Die Abweichung ist mit 2–3 Pixeln (etwa 0,3 %) so gering, dass sie unsichtbar ist.

Bleibt die Frage, warum die Zahlen überhaupt von den ITU-Werten abweichen. Das kann letztendlich nur die MPEG beantworten. Auffällig ist jedenfalls, dass das MPEG 4-PAR überall von horizontal 704 aktiven Pixeln ausgeht. Diese Zahl ist glatt durch 16 teilbar, was eine Verbindung zum Mod16-Kriterium nahe legt. Außerdem sind 704 × 576 (PAL) bzw. 704 × 480 (NTSC) alternative DVD-Auflösungen, die

gerne für digitale TV-Sendungen verwendet werden.

Was auch immer der tatsächliche Hintergrund sein mag: Fakt ist, dass sowohl der Xvid- als auch der DivX-Encoder die MPEG-4-Tabelle für ihre Standard-PAR-Auswahl verwenden. Zusammen mit den simplen Zahlen, die man gut im Kopf behalten kann, ist das Grund genug, die MPEG-4-Tabelle für die Praxis zu empfehlen.

ITU-PAR oder generisches PAR?

Mit ITU-PAR meine ich im Folgenden immer auch die MPEG-4-Werte. 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

Neben ITU-PAR und MPEG-4-PAR gibt es noch eine weitere Variante, das Generische PAR. Das geht von der Überlegung aus, dass die ITU-Empfehlung ein Relikt aus der analogen Vergangenheit ist und in der modernen digitalen Welt nicht mehr benötigt wird. Seit Jahren tobt nun ein mit religiöser Hingabe geführter Streit darum, welche PAR-Rechnung der absolute Schluss der Weisheit wäre.

Eine analoge Fernsehkamera am Anfang der Verarbeitungskette ist der Grundgedanke hinter der ITU-R BT.601, denn nur dann haben wir es mit der einen Art Quelle zu tun, auf die sich die Empfehlung bezieht: ein analoges NTSC- bzw. PAL-Signal. Schon auf analoge Kinofilme mag die BT.601 nicht mehr so richtig passen, und in der modernen Welt mit digitalem Quellmaterial verliert sie völlig ihren Sinn.

Das Generische PAR legt ein digitales Szenario zu Grunde und geht von der berechtigten Annahme aus, dass man die vorhandene DVD-Auflösung auch vollständig nutzen sollte. Wenn man überhaupt von einem aktiven Bereich sprechen kann, dann umfasst der die komplette Auflösung: 720×576 bzw. 720×480. Vom digitalen Standpunkt aus gesehen ist dieses Vorgehen tatsächlich viel einleuchtender als die Verrenkungen der ITU-Empfehlung. Tab. C-4 fasst die Hintergedanken der beiden Ansätze noch einmal zusammen.

Die generische Berechnung funktioniert genauso wie weiter oben für ITU beschrieben, nur dass wir die volle DVD-Auflösung als aktiv ansehen. Entsprechend rechnen wir nicht mit 702 oder 710,85 im Nenner, sondern mit 720. Bei NTSC

	ITU-PAR	Generisches PAR
vorgesehene Quelle	analoge TV-Kamera	digitale Quelle
Verbreitungsmedium	digital (z. B. DVD, DVB)	digital (z. B. DVD, DVB)
typisches Abspielgerät	analoger Fernseher	digital/analog gleichermaßen
Verhältnis zur DVD-Auflösung	aktiver Bereich kleiner als DVD-Auflösung	aktiv ist gesamte DVD-Auflösung

Tab. C-4: Hintergedanken von ITU-PAR und Generischem PAR.

verwenden wir außerdem immer 480 vertikale Pixel, ohne die eigentliche NTSC-Höhe von 486 Pixeln zu berücksichtigen. Das entspricht dem Gedanken, die analoge Welt hinter uns zu lassen, denn digitale NTSC-Produktionen arbeiten üblicherweise sowieso nur mit der 480er Höhe. Schlussendlich erhalten wir folgende PAR-Tabelle:

	PAL	NTSC
4:3	16/15	8/9
16:9	64/45	32/27

Tab. C-5: Generisches PAR.

Darüber sind sich die Verfechter sämtlicher Lehren auch einig. Aber nur so weit, und kein bisschen weiter.

Diskussion: ITU ist sinnvoller

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, mit denen in Diskussionen immer wieder argumentiert wird.

► Das Generische PAR ist nicht auf jeden Fall falsch. Genauso ist das ITU-PAR nicht auf jeden 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, wie die Beispielbilder aus Abb. C-16 zeigen.

Abb. C-16

auf ¾ gequetscht | auf 133 % entzerrt (korrekt) | auf 200 % entzerrt (falsch)

Genauso blödsinnig 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 Produktionsprozess 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 nämlich nirgendwo auf der DVD vermerkt, es existiert also keine Möglichkeit, den richtigen Wert einfach abzulesen. Genauso unmöglich erscheint es bei kommerziellen DVDs, sämtliche Techniker der kompletten Verarbeitungskette 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 und obendrein höchst fehleranfällig? Stimmt. Möglicherweise können wir aber mit ein wenig Überlegung einer der beiden Varianten den Vorzug geben. Um das Ergebnis vorweg zu nehmen:

Das ITU-PAR zu wählen, ist die logischere 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 verdäch-

tig 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.

► Digitale Wiedergabegeräte gewinnen stark an Bedeutung. Mit der Methode, ITU zu verwenden und den inaktiven Bereich mit einem zusätzlichen Stückchen Bild zu füllen, wird man jeder Abspielmethode gerecht.
► Man vermeidet vertikale Balken, deren Übergänge zum Bild viel Bitrate schlucken, da sie kaum jemals direkt auf eine Makroblockgrenze fallen.
► Gute Standalone-DVD-Player verwenden tendenziell das ITU-PAR.
► Die ITU-Norm ist eine Art unverrückbares Video-Naturgesetz. Ganz besonders professionelle Videohardware hält sich strikt daran. Dass ein Profi mit Generischem PAR arbeiten sollte, mutet deshalb etwas seltsam an.
► Die MPEG-4-PAR-Tabelle lehnt sich stark an die ITU-Werte an.

Das alles beweist nichts eindeutig und endgültig. Es lässt allerdings die Aussage wahrscheinlich erscheinen, dass die Mehrheit der DVDs entsprechend ITU-R BT.601 gemastert wird. Wer nicht jede DVD vor dem Encoding exakt vermessen will, dem rate ich deshalb zu den ITU-Werten.

Das gilt nicht für die neuen hochauflösenden Formate wie Blu-ray oder HDTV. Da denen von Anfang ein ein vollständig digitales Szenario zugrunde liegt, ist das Generische PAR die einzig sinnvolle Wahl. Die große Mehrheit der HD-Quellen umgeht das Problem sowieso, indem sie quadratische Pixel verwendet.

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.

$$\text{PAL-Abweichung} = {}^{512}\!/_{351} \,/\, {}^{64}\!/_{45} \approx 2{,}56\,\%$$
$$\text{NTSC-Abweichung} = {}^{5760}\!/_{4739} \,/\, {}^{32}\!/_{27} \approx 1{,}03\,\%$$

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, Bücher und Forum-Threads

[Aho04] Jukka Aho: *The 625/50 PAL Video Signal and TV Compatible Graphics Modes.* Website, 11.02.2004. http://www.iki.fi/znark/video/modes/

[Aho08] Jukka Aho: *A Quick Guide to Digital Video Resolution and Aspect Ratio Conversions.* Website, 15.01.2008. http://www.iki.fi/znark/video/conversion/

[akap07] akapuma: *agkp – akapuma's GKnot Personalisierer.* Gleitz-Forumthread, 19.06.2007. http://forum.gleitz.info/showthread.php?t=22860

[bond05] bond: *New x264 feature: mixed references.* Doom9.org-Forumthread, 2005. http://forum.doom9.org/showthread.php?t=101056

[BrJo06a] Brother John: *ITU-R BT.601 and PAR: How good is my knowledge?* Doom9.org-Forumthread, 2006. http://forum.doom9.org/showthread.php?t=111102

[BrJo06b] Brother John: *ITU-R BT.601 und das PAR: Ist mein Wissen korrekt?* Gleitz-Forumthread, 2006. http://forum.gleitz.info/showthread.php?t=28145

[BrJo07a] Brother John: *Test: Auswirkung von Nicht-Mod16-Auflösungen auf die Xvid-Effizienz.* Gleitz-Forumthread, 2007. http://forum.gleitz.info/showthread.php?t=34216

[BrJo07b] Brother John: *BeSweet-Kommandozeilenreferenz.* Website, 18.12.2007. http://brother-john.net/besweet-referenz.html

[BrJo07c] Brother John: *HQ-Encoding: x264 im Vergleich mit Xvid CQ2.* Gleitz-Forumthread, 2007. http://forum.gleitz.info/showthread.php?t=36104

[BrJo08a] Brother John: *AR-Flag-Unterstützung in DirectShow.* Website, 16.03.2008. http://brother-john.net/ar-vergleich.html

[BrJo08b] Brother John: *Test: Auswirkung von Nicht-Mod16-Auflösungen auf die x264-Effizienz.* Gleitz-Forumthread, 2008. http://forum.gleitz.info/showthread.php?t=36603

[DaSh08] Dark Shikari: *VAQ: Making Xvid's Adaptive Quantization Not Suck.* Doom9.org-Forumthread, 2008. http://forum.doom9.org/showthread.php?t=135093

[Doom05] Doom9: *Codec shoot-out 2005 – Final.* Website, 2005. http://www.doom9.org/codecs-final-105-1.htm

[Fied06] Martin Fiedler: *Videokompressionsverfahren von MPEG-1 bis DivX, VC-1 und H.264.* Revision 6, 2006. http://keyj.s2000.ws/files/projects/videocomp.pdf

[Flor01] Brian Florian: *Dialogue Normalization: Friend or Foe.* 8/2001.
http://www.hometheaterhifi.com/volume_7_2/
feature-article-dialog-normalization-6-2000.html

[HAud08] Hydrogenaudio: *LAME.* Hydrogenaudio-Wiki-Artikel, 19.01.2008.
http://wiki.hydrogenaudio.org/index.php?title=LAME

[ITU07a] International Telecommunication Union: *ITU-R BT.601-6. Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-screen 16:9 Aspect Ratios.* 01/2007. http://www.itu.int/rec/R-REC-BT.601

[ITU07b] International Telecommunication Union: *ITU-T H.264. Advanced video coding for generic audiovisual services.* 11/2007.
http://www.itu.int/rec/T-REC-H.264

[KpeX05] KpeX: *Audio FAQ.* Doom9.org-Forumthread, 12.10.2005.
http://forum.doom9.org/showthread.php?t=68300

[LigH03] LigH: *Dynamikkompr./Boost mit BeSweet.* Gleitz-Forumbeitrag, 27.09.2003. http://forum.gleitz.info/showpost.php?p=41321&postcount=2

[Mare05] Sebastian Mares: *Public, Multiformat Listening Test @ 128 kbps.* Website, 12/2005. http://www.maresweb.de/listening-tests/mf-128-1

[Mayo04] Alastair Mayo: *The Official DivX 5.2 Guide.* Deutsche Version, DivX Inc, 2004. http://download.divx.com/divx/DivXUserGuide521-de.exe

[mp4g06a] *.mp4 guy: *Custom Matrices.* (AVC) Doom9.org-Forumthread, 2006.
http://forum.doom9.org/showthread.php?t=117041

[mp4g06b] *.mp4 guy: *How To Use Mpeg4 AVC Deblocking Effectively.* Doom9.org-Forumthread, 2006. http://forum.doom9.org/showthread.php?t=109747

[neur07] neuron2: *DGAVCDec 1.0.0 Alpha.* Doom9.org-Forumthread, 2007.
http://forum.doom9.org/showthread.php?t=122598

[Ober01] Daniel Oberle: *Das Breitbild-Phänomen.* 2001.
http://www.aifb.uni-karlsruhe.de/WBS/dob/docs/Breitbild.pdf

[Pana99] *Das Video-Kompressionsbuch.* Panasonic, 1999.
http://panasonic-broadcast.de/
index.cfm?uuid=356BA7F6C09F11269B3BB10E84B5C76A

[Pill03] Latha Pillai: *Quantization.* Xilinx XAPP615 (v1.1) June 25, 2003.
http://www.xilinx.com/bvdocs/appnotes/xapp615.pdf

[Rich03] Iain E. G. Richardson: *H.264 and MPEG-4 Video Compression.* Wiley, 2003. ISBN 0-470-84837-5.

[RichMJ] Iain E. G. Richardson: *H.264 Tutorials.* Website, mehrere Jahreszahlen.
http://vcodex.com/h264.html

[SchaOD] scharfis_brain: *Exotisches Interlacing.* Website, ohne Datum.
http://home.arcor.de/scharfis_brain/ExotischesInterlacing/

[SeeM05] SeeMoreDigital: *GPAC MP4Box Information, Requests and Discussions.* Doom9.org-Forumthread, 2005.
http://forum.doom9.org/showthread.php?t=94874

[Selu05] Selur: *Wissenswertes rund um Xvid.* Version 0.2.6, 2005.
http://www.flaskmpeg.info/board/thread.php?postid=50836

[Selu06] Selur: *man x264.* Version 0.0.9, 2006.
http://www.flaskmpeg.info/board/thread.php?threadid=5571

[Shar05a] Sharktooth: *EQM AVC Series.* Doom9.org-Forumthread, 2005.
http://forum.doom9.org/showthread.php?t=96298

[Shar05b] Sharktooth: *New Quantization Matrices – EQM V3 series.* Doom9.org-Forumthread, 2005. http://forum.doom9.org/showthread.php?t=83125

[Shar08a] Sharktooth: *MeGUI Custom x264/AVC video profiles.* Doom9.org-Forumthread, 2008. http://forum.doom9.org/showthread.php?t=101813

[Shar08b] Sharktooth: *x264 win32 daily builds.* Doom9.org-Forumthread, 2008.
http://forum.doom9.org/showthread.php?t=89979

[Soul04] Soulhunter: *Human color sensitivity test.* Doom9.org-Forumthread, 2004.
http://forum.doom9.org/showthread.php?t=72803

[Spik05] SpikeSpiegel: *CQMs Compressibility TEST.* Doom9.org-Forumbeitrag, 03.05.2005. http://forum.doom9.org/showthread.php?p=649221#post649221

[squi06] squid_80: *Xvid_encraw – Patched with AviSynth input support.* Doom9.org-Forumthread, 2006.
http://forum.doom9.org/showthread.php?t=98469&goto=newpost

[Symes04] Peter Symes: *Digital Video Compression.* McGraw-Hill, 2004. ISBN 0-07-142487-3.

[Tayl08] Jim Taylor: *DVD Frequently Asked Questions (and Answers).* Website, 04.01.2008. http://www.dvddemystified.com/dvdfaq.html

[Teeg06] Teegedeck: *XviD presets thread.* Doom9.org-Forumthread, 2006.
http://forum.doom9.org/showthread.php?t=119399

[Wick02] Sebastian Wickenburg; Aeneas Rooch; Johannes Groß: *Die JPEG-Kompression.* Website, 2002. http://www.mathematik.de/spudema/spudema_beitraege/beitraege/rooch/neinleit.html

[Wieg03] Thomas Wiegand; Gary J. Sullivan; Gisle Bjøntegaard; Ajay Luthra: *Overview of the H.264/AVC Video Coding Standard.* IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13/7, Juli 2003.
http://ip.hhi.de/imagecom_G1/assets/pdfs/csvt_overview_0305.pdf

Weitere nützliche Links

Alexander Vigovskys AC3Filter
`http://ac3filter.sourceforge.net`

Avidemux: Videoeditor. Plattformübergreifende Alternative zu VirtualDub.
`http://avidemux.org/`

AVInaptic: Geniales Analysetool für Filmdateien
`http://fsinapsi.altervista.org/code/avinaptic/index.html`

BBC - A Guide to Picture Size
`http://www.bbc.co.uk/commissioning/tvbranding/picturesize.shtml`

Die ultimativen DVD-Tipps & Tricks
`http://www.dvd-tipps-tricks.de`

ffdshow: DirectShow-Decoder für viele Audio- und Videoformate
`http://www.ffdshow.info`

GSpot: Analysetool für Filmdateien
`http://www.headbands.com/gspot`

Haali Media Splitter mit Unterstützung für Matroska, MP4, AVI, OggMedia
`http://haali.cs.msu.ru/mkv`

Heinis Xvid-Custom-Matrizen
`http://forum.doom9.org/showpost.php?p=854722&postcount=284`

HydrogenAudio. Extrem kompetente Audio-Community.
`http://www.hydrogenaudio.org`

Jawor's Quantization Matrices
`http://jawormat.republika.pl/matrices.html`

Matroska-Homepage
`http://www.matroska.org`

Mediainfo: Analysetool für Filmdateien.
`http://mediainfo.sourceforge.net`

Moving Picture Experts Group (Homepage)
`http://www.chiariglione.org/mpeg`

RareWares: Downloadseite für Audiosoftware
`http://www.rarewares.org`

Videoplayer

Media Player Classic: `http://sourceforge.net/projects/guliverkli2`

Zoom Player: `http://www.inmatrix.com`

VLC Media Player: `http://www.videolan.org`

SMPlayer: `http://smplayer.sourceforge.net`

What is Deinterlacing? Facts, solutions, examples.
`http://100fps.com`

Xiph.Org Foundation. Die Organisation hinter Vorbis, FLAC, Theora und Speex.

`http://xiph.org`

Xvid-Homepage

`http://xvid.org`

Glossar

A

AAC	*Advanced Audio Coding.* Ein Audioformat.
ABR	*Average Bitrate.* Encodingmethode mit einer vorher definierten durchschnittlichen Bitrate als Ziel.
AC-3	*Adaptive Transform Coder 3.* Üblichstes Audioformat der DVD.
agkp	*Akapumas Gordian-Knot-Personalisierer.* Gordian-Knot-Erweiterung für x264, Matroska und fortgeschrittenes Filtern.
AR	*Aspect Ratio.* Seitenverhältnis eines Videos oder Pixels.
ASF	*Advanced Streaming Format.* Containerformat von Microsoft.
ASP	*Advanced Simple Profile.* Ein Profil des Standards MPEG-4 Visual. Benutzt z. B. von Xvid, DivX, 3ivx.
AVC	*Advanced Video Coding.* Ein Videostandard. Benutzt z. B. von x264, Nero AVC.
AVI	*Audio/Video Interleave.* Ein Containerformat von Microsoft.
AVS	*AviSynth Script.* Dateiendung und gern benutzte allgemeine Abkürzung.

B

BPF	*Bits/(Pixel*Frame).* Grober Indikator für die Videoqualität.

C

C	*Center Channel.* Audiokanal vorne Mitte.
CABAC	*Context-Adaptive Binary Arithmetic Coding.* Verfahren zur Entropiecodierung in H.264.
CBR	*Constant Bitrate.* Encodingmethode mit fester Bitrate pro Zeiteinheit.
CD	*Compact Disc.*
CQ	*Constant Quality.* Encodingmethode, die ohne Berücksichtigung der Dateigröße ein vorher festgelegtes Qualitätslevel an jeder Stelle der Audio/Videodaten einzuhalten versucht.

	Constant Quantizer. Video-Encoding mit einem konstanten Quantisierungsfaktor.
CQM	*Custom Quantizer Matrix.* Benutzerdefinierte Quantisierungsmatrix für einen Video-Encoder.
CSS	*Content Scrambling System.* DRM-System der DVD.

D

DAR	*Display Aspect Ratio.* Wiedergabe-Seitenverhältnis eines Videos.
DCT	*Diskrete Cosinus-Transformation.* Zentraler Bestandteil vieler Videokompressions-Verfahren.
DNR	*Dialog Normalization Reduction.* Dialoglevel-Anpassung. Eine Funktion von AC-3.
DPL	*Dolby Pro Logic.* Von Dolby entwickeltes System zur Matrixcodierung von Audio-Surround-Informationen.
DRC	*Dynamic Range Compression.* Dynamikkompression von Audio.
DRM	*Digital Rights Management.* Oberbegriff für digitale Systeme zur Abspiel- und Kopierbeschränkung.
DTS	*Digital Theater Systems.* Ein Audioformat.
DVD	Ohne konkrete Bedeutung. Ursprünglich *Digital Video Disc,* manchmal auch *Digital Versatile Disc.*
DVD-5	Einseitige Single-Layer-DVD mit 4,37 GByte Kapazität.
DVD-9	Einseitige Dual-Layer-DVD mit 7,95 GByte Kapazität.

F

FL	*Front Left Channel.* Audiokanal vorne links.
fps	*Frames per Second.* Bilder pro Sekunde. Angabe der Geschwindigkeit, mit der ein Film läuft.
FR	*Front Right Channel.* Audiokanal vorne rechts.

G

GKnot	*Gordian Knot.* Ein Encoding-Frontend.
GMC	*Global Motion Compensation.* Funktion zur Bewegungskompensation bei der Video-Encodierung.

GUI *Graphical User Interface.* Grafische Benutzeroberfläche.

H

H.264 siehe AVC.

HE *High Efficiency.* Ein Encodingmodus des AAC-Audioformats.

HVS *Human Visual System.* Menschlicher Sehapparat. Meistens gebraucht im Zusammenhang mit der Optimierung des Video-Encodings unter Berücksichtigung der Eigenschaften des HVS.

I

iDCT *Inverse Diskrete Cosinus-Transformation.* Bestandteil des Decodings DCT-encodierten Videos.

ISO International Organization for Standardization. Internationale Normungsorganisation mit extrem breitem Aufgabenbereich.

ITU *International Telecommunication Union.* Internationale Normungsorganisation in den Bereichen Radio, Fernsehen, Telekommunikation.

J

JPEG *Joint Photographic Experts Group.* Arbeitsgruppe von ISO und ITU im Bereich digitale Bitmap-Grafikformate.

L

LAME *LAME Ain't an MP3 Encoder.* Wichtigster heutiger MP3-Codec.

LC *Low Complexity.* Ein Encodingmodus des AAC-Audioformats.

LFE *Low Frequency Effects Channel.* Audio-Basskanal.

LPCM *Linear Pulse Code Modulation.* Unkomprimiertes Audioformat.

M

ME *Motion Estimation.* Bewegungssuche bei der Video-Encodierung.

MKV *Matroska-Video.* Dateiendung und weit verbreitete allgemeine Abkürzung für das Matroska-Containerformat.

mmg *MKVMerge GUI.* Grafische Oberfläche für den am weitesten verbreiteten Matroska-Muxer.

MP2 *MPEG-1 Audio Layer 2.* Ein Audioformat.

MP3 *MPEG-1 Audio Layer 3.* Ein Audioformat.

MPEG *Moving Picture Expert Group.* Arbeitsgruppe der ISO, die sich mit digitalen Audio- und Videoformaten beschäftigt. Offizielle Bezeichnung: ISO/IEC JTC1/SC29/WG11.

MT *Multi Threading.* Aufteilen eines rechenintensiven Vorgangs auf mehrere Threads, um einen Mehrkern-Prozessor ausnutzen zu können.

N

NTSC *National Television Systems Committee.* US-amerikanische Fernseh-Normungsorganisation. Name des US-Fernsehformats.

O

OGM *OggMedia.* Veraltetes Containerformat.

OTA *Overall Track Adjustment.* Eine Gruppe von Funktionen des Audio-Transcoders BeSweet.

P

PAL *Phase Alternation Line.* Europäische Fernsehnorm.

PAR *Pixel Aspect Ratio.* Wiedergabe-Seitenverhältnis eines Pixels.

PCM *Pulse Code Modulation.* Unkomprimiertes Audioformat.

PGC *Program Chain.* Teil der Datenstruktur einer Video-DVD.

PN *Private Nachricht.*

PSNR *Peak Signal to Noise Ratio.* Messverfahren für objektive Videoqualität.

PVE *Psychovisual Enhancements.* Bezeichnung für die HVS-Optimierungen des DivX-Codecs.

Q

QP *Quantiser scale parameter.* Ein Kompressionsfaktor im Encoder. Oft auch *Quant* oder *Quantizer* genannt.

Qpel *Quarter Pixel.* Bewegungssuche mit einem Viertelpixel Genauigkeit.

R

RDO *Rate Distortion Optimization.* Verfahren zur optimierten Abwägung zwischen Datenrate und Qualität beim Video-Encoding.

S

SBC *Smart Bitrate Coding.* Optimiertes 2-Pass-Encoding mit DivX 3 und Nandub.

SBR *Spectral Band Replication.* Technologie zur Bitraten-Einsparung, auf der HE-AAC basiert.

SL *Surround Left Channel.* Audiokanal hinten links.

SMP *Simultaneous Multi Processing.* Siehe MT.

SR *Surround Right Channel.* Audiokanal hinten rechts.

SSIM *Structural Similarity.* Messverfahren für objektive Videoqualität.

SSRC *Shibatch Sample Rate Converter.* Ein Programm zur Umrechnung von Audio-Samplingraten.

SVCD *Super Video-CD.* Nicht genormte Erweiterung der Video-CD.

V

VBR *Variable Bitrate.* Encodingmethode, bei der die Bitrate schwanken darf. Oft synonym zu Constant Quality (CQ) gebraucht.

VCD *Video-CD.* Standardisiertes Format für CDs mit MPEG-1-Video und MP2-Audio.

VHQ *Vastly Hyped Quality* bzw. *Very Handsome Qaligosaur.* RDO-Implementierung des Xvid-Videoencoders.

VfW *Video for Windows.* Traditionelles Videoframework für Windows.

VOB *Video Object.* Containerformat der DVD.

VTS *Video Title Set.* Teil der Datenstruktur einer Video-DVD.

Changelog

Hier stehen lediglich die Änderungen zur jeweiligen Vorgängerversion. Das vollständige Changelog seit Adam und Eva könnt ihr online einsehen:
`http://encodingwissen.de/changelog.html`

2. Juli 2008 (rev 264)

- Artikel zur Dialogue Normalization [Flor01] hinzugefügt.
- Rechtschreibkorrekturen.

11. Juni 2008 (rev 259)

- Den kompletten Text habe ich durchgelesen, korrigiert und aktualisiert. Deswegen gibts natürlich eine Unzahl von kleineren Änderungen.
- StaxRip-Update auf die aktuelle Beta-Version.
- Philosophie: Eckpunkte fürs perfekte Backup überarbeitet.
- Xvid- und x264-Kapitel wegen VAQ angepasst.
- Ripping: vStrip durch PgcDemux ersetzt.
- ITU-R-Kapitel vollständig überarbeitet und erweitert.
- Neues Logo und Favicon.
- Ich habe nochmal ausführlich überlegt, wie stark ich den Urheberrechtsdaumen auf dem Encodingwissen halten möchte. Veränderungen nicht zuzulassen, war zu restriktiv. Im Gegenteil freue ich mich sogar, wenn jemand mehr oder weniger große EW-Teile für ein eigenes Projekt nutzen möchte, egal ob das kommerziell oder nicht ist. Wichtig ist mir nur, dass die Offenheit erhalten bleibt. Also brauchts ein Copyleft. Und deswegen: Lizenzänderung zu CC-by-sa.

Lizenzen

Dieses *Dokument* ist lizenziert unter der Creative-Commons-Lizenz »Namensnennung – Weitergabe unter gleichen Bedingungen 2.0 Deutschland« (CC-by-sa).

Erlaubt ist es,
- das Werk zu vervielfältigen, zu verbreiten und öffentlich zugänglich zu machen,
- Bearbeitungen des Werkes anzufertigen,

unter Einhaltung folgender Bedingungen:
- Namensnennung des Autors/Rechteinhabers.
- Weitergabe veränderter Versionen nur unter Verwendung von Lizenzbedingungen, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

Der vollständige Lizenzvertrag kann online unter folgender Adresse eingesehen werden: `http://creativecommons.org/licenses/by-sa/2.0/de/legalcode`

Als Ausnahme von obiger Lizenz untersteht die *»Filmrolle«-Logografik* auf der Titelseite der GNU Free Documentation License, die online unter folgender Adresse eingesehen werden kann: `http://www.gnu.org/copyleft/fdl.html`
Der Grafik liegt ein Werk von Bromskloss zugrunde, das verfügbar ist unter:
`http://commons.wikimedia.org/wiki/Image:Film_reel.svg`
Die »Filmrolle«-Logografik ist als OpenDocument-Zeichnung verfügbar unter:
`http://encodingwissen.de/resources/Filmrolle.zip`

Beide vollständige Lizenztexte sind dem Downloadarchiv der Encodingwissen-PDF-Ausgabe in der Datei *Lizenzen.txt* beigelegt. Die OpenDocument-Quelldateien des Encodingwissens sind auf Anfrage erhältlich.