Brother John

Das Encodingwissen

DVD-Backup nach MPEG-4

Revision 93 vom 10. Februar 2007
http://encodingwissen.brother-john.net

Inhalt

Vorwort und Wegweiser

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

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

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

Anfänger

Du bist neu, hast vielleicht noch nie einen Film encodiert? Dann zählen erst einmal Ergebnisse. Bevor du loslegst, solltest du auf jeden Fall die Hintergründe zu den Bestandteilen eines Films (A.1.4, Seite 14) und zum Encodingablauf (A.1.5, Seite 17) durchlesen, um eine grobe Vorstellung zu erhalten, wie eine Videodatei aufgebaut ist und welche Schritte nötig sind, um von der DVD zum fertigen Encoding zu gelangen. Dann stürze dich auf den Praxisteil (Teil B, Seite 117), wo ein komplettes Encoding Schritt für Schritt im Detail erklärt wird. Ein paar Mal musst du dazu sicherlich zurück zum Hintergrundabschnitt Formatedschungel (A.2, Seite 20) springen, um dich für einen Codec o. ä. zu entscheiden. Einfach den entsprechenden Links in den Praxiskapiteln folgen. Bei der Konfiguration der Audio- und Videocodecs kannst du ruhig erst einmal auf die empfohlenen Einstellungen vertrauen.

Auf- und Umsteiger

Die ersten Schritte sind gemacht und die ersten Encodings erfolgreich beendet? Oder kommst du aus der Ecke von (S)VCD und DVD-Authoring und willst dich auch mit der MPEG-4-Welt vertraut machen? Schwierig, hier eine Empfehlung zu geben. Sicher ist das Praxiswissen (Teil B, Seite 117) zentral wichtig – als Umsteiger auch das Kapitel zum Encodingablauf (A.1.5, Seite 17). Außerdem solltest du es nicht versäumen, dich mit der Videocodec-Konfiguration (A.4, Seite 58) eingehender zu beschäftigen. Der Rest des Hintergrundwissens ist hauptsächlich Geschmackssache. Ich persönlich würde mir den Grundlagen-Teil (A.1, Seite 3) bis zum Schluss aufheben, denn was dort besprochen wird, ist wirklich grundlegend. Fürs tägliche Encoding hat dieser Abschnitt wenig unmittelbaren Nutzen.

Profi

Du bist Profi mit mehrjähriger Encodingerfahrung und mit allen Wassern gewaschen? Meiner Erfahrung nach besonders interessant dürften die Kapitel rund um anamorphes Bild (A.3.2, Seite 40) und ITU-R BT.601 (C.2.2, Seite 211) sein, außerdem möglicherweise Akapumas Gordian-Knot-Personalisierer (B.4.2.7, Seite 182). Ansonsten solltest du das Inhaltsverzeichnis als erste Anlaufstelle benutzen und das Encodingwissen als Nachschlagewerk ansehen. Schließlich kann man nicht absolut jedes Detail immer im Kopf behalten.

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

Brother John

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.

Und um den Rahmen gar abzustecken: MPEG-4 ist unser Zielformat der Wahl, also Xvid, DivX & Co. 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.

Mein typisches DVD-Backup erhält die volle Auflösung der DVD und meistens mindestens eine der originalen AC3-Audiospuren. Das braucht Platz, weshalb ich einem Film nicht weniger als ½ DVD-5 (2240 MB) gönne. Das sind keine guten Voraussetzungen für die seit Jahren weit verbreiteten Standardzielgrößen. Das 2-CD-Encoding (1400 MB) nähert sich dem Encodingwissen-Qualitätsniveau oft ganz 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 trotzdem 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

Was Perfektion tatsächlich heißt, hängt entscheidend von den persönlichen Vorlieben und Anforderung ab. Fürs Encodingwissen steht, wie gerade angesprochen, die Qualität absolut im Vordergrund. Alles andere ist erst einmal zweitrangig. Und das führt für das perfekte Backup zu folgenden Eckpunkten.

- Beibehalten der originalen Bildauflösung, nur Wegschneiden der schwarzen Ränder. Typische Auflösung für eine PAL-DVD sind dann 720 × 432, 720 × 544 oder gar die vollen 720 × 576 Pixel.
- Filtering nur, um Fehler im Originalbild zu beheben. Kein Filtering, um die Komprimierbarkeit zu erhöhen, da das nahezu immer auf weniger Bilddetail und damit geringere Qualität hinausläuft.
- Die Audiospur der bevorzugten Sprache in unveränderter Form, also meistens als 6-Kanal-AC3. Falls das doch zu viel Platz beansprucht, Transcoding nach 6-Kanal-HE-AAC in der Region um 160 – 190 kbit/s.
- Untertitel in dynamischer Form als Text, da so die höchste Darstellungsqualität erreicht werden kann.

Das ist zumindest das, was ich als perfekt definiere. Klar, unter ½ DVD-5 Zielgröße läuft nichts. Für lange und/oder anspruchsvolle Streifen, darf es auch einmal eine ganze DVD sein. Auch wird ein solches Backup niemals auf einem Hardwareplayer abspielbar sein.

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! Leider lauert da der Pfusch. Ein nicht unwesentlicher Teil aller Hilferufe in diversen Foren ist vom Wunsch geprägt, MPEG-4 mit dem Wohnzimmerplayer abzuspielen. Es gibt zwischen den Geräten millionenfache Unterschiede, Einschränkungen, Bugs, Besonderheiten, Workarounds. An die vom MPEG-4-Standard vorgegebenen Rahmen hält sich keiner. Das allein ist Grund genug, den Standalone im Encodingwissen zu ignorieren. Wer es trotzdem versuchen will, sollte sich einen von DivX zertifizierten Player und den DivX-Codec kaufen und mit dem passenden Profil encodieren. Dann sollten zumindest mit diesem einen Player keine Probleme auftreten. Vielleicht wäre es aber sinnvoller, das Gerät dafür zu nutzen, was es kann: Video-DVDs abspielen.

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 ganz am Anfang). 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.

Teil A
Hintergrundwissen

Einleitung

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

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

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

A.1 Grundlagen

A.1.1 Der Nutzen der Kompression

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

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

Platzbedarf unkomprimierter Videodaten

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

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

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

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

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

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

Platzbedarf unkomprimierter Audiodaten

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

2 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 DVD mit ihrer Kapazität von 8 GByte geradezu winzig.

Verlustlose und verlustbehaftete Kompression

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

Verlustlos (lossless)

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

Verlustbehaftet (lossy)

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

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

A.1.2 Die Videokompression

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

A.1.2.1 Intraframe-Kompression

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

Farbräume

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

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

Makroblocks und DCT

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

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

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

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

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

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

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

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

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

A.1.2.2 Interframe-Kompression

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

I-Frames

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

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

P-Frames

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

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

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

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

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

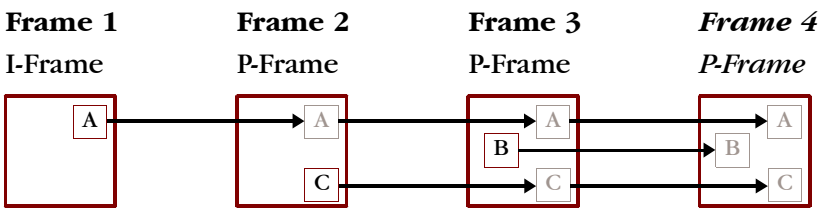

Abbildung 2: Rückbeziehung auf Blocks aus früheren Bildern bei P-Frames.

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

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

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

kommen ist. Das Splitting-Kapitel C.1.1 ab Seite 201 im letzten Abschnitt des Encodingwissens ist ein guter Punkt, um sich an diese Tatsache wieder zu erinnern.

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

B-Frames

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

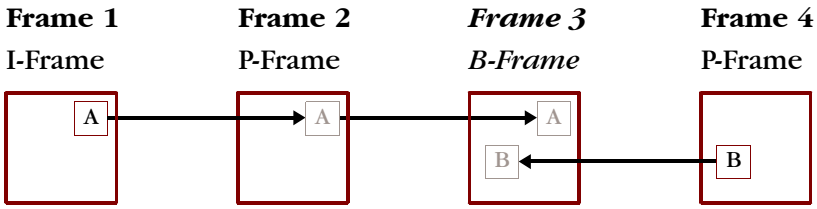

Abbildung 3:Bidirektionale Rück- und Vorwärtsbeziehung bei B-Frames.

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

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

Auswahl der Frametypen

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

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

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

A.1.3 Die Audiokompression

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

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

Maskierung

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

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

Leise Töne

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

Hohe Frequenzen

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

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

Kanalgemeinsamkeiten

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

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

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

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

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

Variable und konstante Bitrate

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

Constant Bitrate

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

Variable Bitrate

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

Average Bitrate

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

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

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

A.1.4 Bestandteile eines Films

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

Datenspuren

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

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

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

Metadaten

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

Container und Interleaving

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

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

Abbildung 4: Schuhkarton-Vergleich

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

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

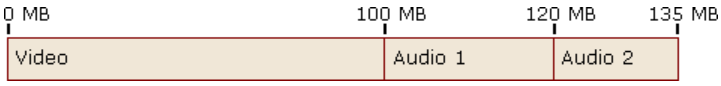

Abbildung 5: Physisches Containerlayout ohne Interleaving.

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

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

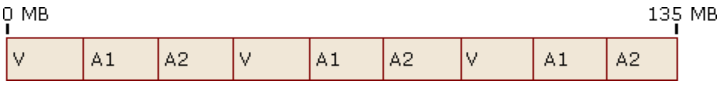

Abbildung 6: Physisches Containerlayout mit Interleaving.

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

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

A.1.5 Der Ablauf eines Encodings Schritt für Schritt

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

DVD-Ripping

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

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

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

Indexing und Audio-Demuxing

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

Audio-Transcoding

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

Untertitel-Vorbereitung

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

Videocodec-Konfiguration

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

Video: AviSynth-Skript

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

Encoding

Das Encoding ist der Vorgang, der unser Zielvideo erstellt. Dabei rechnet der Videocodec das Quellmaterial der DVD ins MPEG-4-Format um und trimmt es auf

die gewünschte Größe. Um optimale Qualität zu erhalten und Gleichzeit die Zielgröße nicht zu verpassen, findet das Encoding in zwei Durchläufen statt. Im ersten Durchgang (1st Pass) analysiert der Codec Bild für Bild die Komplexität des kompletten Films. Eine Videodatei wird erst im zweiten Durchgang (2nd Pass) erstellt, in dem anhand der gesammelten Daten die Bitrate verteilt wird. Manche Codecs unterstützen auch mehr als zwei Durchgänge, die dann *Nth Passes* genannt werden. Auch hier erhalten wir ab dem zweiten Durchlauf ein funktionsfähiges Video. Allerdings werden im 2nd Pass wieder Daten gesammelt, mit denen ein dritter, nochmals optimierter Durchgang erfolgen kann.

Muxing

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

Splitting

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

A.2 Der Formatedschungel

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

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

A.2.1 Video- und Audioformate

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

Die MPEG-Videostandards

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

MPEG-1

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

MPEG-2

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

MPEG-4

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

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

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

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

Implementierungen der Standards

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

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

Audioformate

Dolby Digital

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

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

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

Digital Theater Systems

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

MPEG

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

MPEG-1 Part 3 Layer 2

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

MPEG-1 Part 3 Layer 3

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

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

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

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

MPEG-4 Part 3

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

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

Vorbis

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

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

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

PCM

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

A.2.2 Das Format der DVD

Die Eckdaten der DVD

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

Tabelle 1: Technische Daten der Video-DVD.

Bildformate

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

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

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

Video und Audio

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

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

Tabelle 2: Erlaubte Auflösungen der Video-DVD.

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

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

Die typische DVD

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

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

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

A.2.3 Die Containerformate

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

Audio/Video Interleave (AVI)

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

Trotzdem hat sich AVI bis heute als verbreitetster Container für MPEG-4-Video behauptet. Dank Matroska und 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 nahe-

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

MP4

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

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

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

OggMedia (OGM)

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

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

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

1 Von der technischen Seite betrachtet besteht kein Unterschied zwischen AAC in AVI und VBR MP3 in AVI. Beides ist laut AVI-Spezifikation erlaubt. Der Unterschied besteht darin, dass VBR MP3 sowohl beim Muxing als auch bei der Wiedergabe weitläufig unterstützt wird. AAC kann dagegen nur von AVI-Mux GUI gemuxt werden. Die Wiedergabe am PC stellt mit einem sinnvollen Filter allerdings kein Problem dar.
2 Der MPEG-4-Standard sieht keine Vobsub-Unterstützung vor. Allerdings existieren in MP4 so genannte private Streams, in denen prinzipiell alles gespeichert sein darf. Nero nutzt dies aus, um Vosubs in MP4 zu packen. Der Standard erlaubt das einwandfrei. Allerdings kann man sich nicht darauf verlassen, dass ein privater Stream überall wiedergegeben wird.
3 Im Standard vorhanden, aber noch ohne breite Unterstützung.
4 Die so genannten DivX- oder MPEG4-fähigen Hardwareplayer verwenden in der Regel AVI als Container.
5 Ein erster Hardwareplayer mit Matroska-Unterstützung ist inzwischen erschienen. Siehe den Doom9.org-Thread *First HW player with Matroska support – Cowon A3*.
6 Noch wenig verbreitet, ist aber v. a. dank Nero im Kommen.

	AVI	Matroska	MP4	OggMedia
		Video[a]		
Native MPEG-4	Nein	Ja	Ja	Nein
VfW MPEG-4	Ja	Ja	Nein	Ja
		Audio		
MP3	Ja [1]	Ja	Ja	Ja
Vorbis	Nein	Ja	Nein	Ja
AAC	teils [1]	Ja	Ja	Ja
AC3, DTS	Ja	Ja	Nein	Ja
		Untertitel		
SubRip, SSA	Ja	Ja	Ja	Ja
Vobsub	Nein	Ja	teils [2]	Nein
		Metadaten		
Stream-Info[b]	Nein	Ja	Ja	teils
Kapitel	Nein	Ja	Ja	Ja
Menüs	Nein	teils [3]	teils [3]	Nein
		Player		
Computer	Ja	Ja	Ja	Ja
Hardware	teils [4]	Nein [5]	teils [6]	Nein

Tabelle 3: Funktionsumfang von Multimedia-Containern.

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

Empfehlung

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

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

A.2.4 Die Videocodecs

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

DivX ;-) 3.11

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

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

DivX

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

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

Xvid

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

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

---

x264

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

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

Empfehlung

Wer sich wenig Gedanken machen will – zwei oder drei Optionen einstellen und loslegen – der ist mit DivX gut beraten. Wer auch noch das letzte Quäntchen Bildqualität herausholen will und bereit ist, sich dafür auch mit den Hintergründen der ganzen Einstellungen zu beschäftigten, dürfte an Xvid mehr Freude haben.

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

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

A.2.5 Die Audiocodecs

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

MP3

Der Klassiker. Mit MP3 brauchen wir uns um Abspielprobleme keine Sorgen zu machen: läuft überall. Dank Lame steht ein hervorragender Encoder bereit, der in der 128-kbit-Region exzellente Ergebnisse erzielt – mit Tonspuren noch mehr als mit Musik. Niedrigere Bitraten sollten wir möglichst vermeiden, da MP3 dafür schlicht und einfach nicht gemacht ist. Davon abgesehen bleibt der größte Nachteil die Beschränkung auf Stereo. Den 6-Kanal-Ton der DVD können wir mit MP3 nicht beibehalten.

Vorbis

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

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

AAC

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

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

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

AC3

Im fertigen Encoding AC3 einzusetzen ist eine einfache Lösung, weil wir die Tonspur nur unverändert von der DVD übernehmen müssen. Es entfällt sowohl die Arbeit als auch der Qualitätsverlust des Transcodings, und natürlich bleibt der Mehrkanal-Ton erhalten. Nachteil: 5.1-AC3 ist mit 384 kbit/s oder 448 kbit/s recht groß, benötigt also eine entsprechend hohe Zielgröße, damit die Videoqualität

nicht leidet. Dank DVD-Brenner ist das heute kein großes Problem mehr, denn bei ½ DVD-5 als Zielgröße ist nahezu immer Platz für mindestens eine AC3-Spur. Für das urklassische hochkomprimierte 1-CD-Encoding können wir AC3 dagegen vergessen. Allerdings muss AC3 nicht immer 6 Kanäle enthalten, auch 192-kbit-Stereo (z. B. die deutschen Tonspuren der Indiana-Jones-DVDs) oder sogar Mono (z. B. Audiokommentare) sind möglich.

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

DTS

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

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

Empfehlung

Der passende Audiocodec hängt von einigen Faktoren ab, von denen einer sicher immer der persönliche Geschmack ist. Wichtiger ist aber erst einmal der gewünschte Film-Container (vgl. Kapitel A.2.3). Entscheiden wir uns für AVI, müssen wir uns gleichzeitig von Vorbis verabschieden, da Vorbis nicht in AVI gemuxt

werden kann. Einige Hardwareplayer haben Probleme mit VBR MP3, und AAC in AVI unterstützt soweit ich weiß keiner. Auch MP4 hat einige Einschränkungen: Vorbis und AC3 sind nicht möglich. Am unproblematischsten ist Matroska, da dieser Container ganz bequem alle angesprochenen Audioformate unterstützt.

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

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

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

A.2.6 Die Untertitelformate

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

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

Untertitel auf der DVD

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

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

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

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

Untertitel im Encoding

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

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

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

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

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

A.3 Das Videobild

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

A.3.1 Cropping (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.

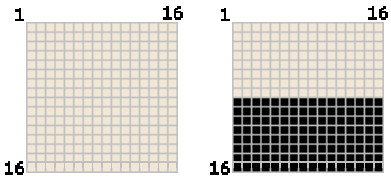

Abbildung 7:
Zwei Makroblocks im Vergleich.

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

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

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

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

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

A.3.2 Anamorphes Video

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

A.3.2.1 Anamorphes Quellvideo

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

Begriffsdefinitionen

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

Seitenverhältnis

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

Display Aspect Ratio (DAR)

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

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

Pixel Aspect Ratio (PAR)

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

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

Anamorphes Video

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

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

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

Sinn des verzerrten Bildes

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

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

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

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

Das Bild der DVD

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

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

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

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

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

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

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

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

Zentrale Bedeutung des PAR

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

Abbildung 8: Unverändertes 16:9-Frame.

Abbildung 9: Entzerrtes 16:9-Frame.

Abbildung 10: Unverändertes 4:3-Frame.

Abbildung 11: Entzerrtes 4:3-Frame.

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

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

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

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

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

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

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

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

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

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

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

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

Tabelle 5: Vergleich von generischem und ITU-DAR.

Wie genau die einzelnen PARs zustande kommen und welche Bedeutung das generische nicht-ITU PAR tatsächlich hat, damit beschäftigt sich das Spezialkapitel C.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 zum schwierigsten, was digitales Video zu bieten hat.

A.3.2.2 Anamorphes MPEG-4

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

Was nun an Zahlen folgt, kann ein wenig von den idealen Werten aus dem letzten Kapitel abweichen. Schuld daran ist eine Eigenart des Encodingverfahrens. MPEG-4-Codecs arbeiten nicht mit einzelnen Pixeln, sondern mit 16×16 Pixel großen Makroblocks. Auflösungen, die sich nicht komplett in vollständige Makroblocks aufteilen lassen, werfen zwar keinen modernen Codec aus der Bahn, senken aber die Effizienz der Kompression und damit die Qualität. Deswegen sollte die Zielauflösung immer sowohl horizontal als auch vertikal glatt durch 16 teilbar sein (Mod16-Kriterium, vgl. Seite 208 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 wunderbare Welt der Amélie* wieder bestens, deren korrektes Wiedergabe-DAR abzüglich der Balken in der 2,35er-Region liegt. Croppen wir die Balken unter Berücksichtigung des Mod16-Kriteriums, bleiben von den 720 × 576 Pixeln der DVD noch 704 × 432 übrig, was einem Seitenverhältnis von 1,63 entspricht und heftige Eierköpfe beim Anschauen zur Folge hat. Um das zu beheben, sind zwei Möglichkeiten denkbar.

Horizontal Strecken

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

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

Das entzerrte Bild enthält

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

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

Vertikal stauchen

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

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

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

Anamorphe Varianten

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

Originalbild behalten

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

Nur Cropping

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

Cropping und Resizing

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

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

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

Display Aspect Ratio und Pixel Aspect Ratio

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

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

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

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

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

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

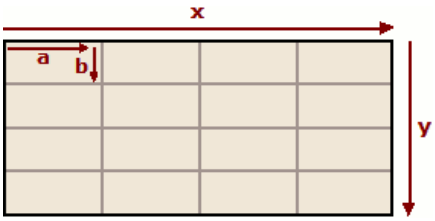

Abbildung 12: PAR und DAR vor dem Cropping.

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

Beispiel für eine 16:9-PAL-DVD:

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

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

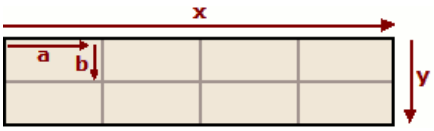

Abbildung 13: PAR und DAR nach dem Cropping.

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

Und für die 16:9-PAL-DVD:

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

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

Das AR-Flag und Decoderunterstützung

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

Im MPEG-4-Videostream

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

Da es nur äußerst beschränkt möglich ist, nach dem Encoding noch das Flag im Videostream zu setzen, muss der Encoder selbst diese Funktion anbieten. Xvid und x264 tun das 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-Muxing, um die AR-Info zu verlieren.

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

Sämtliche Flags nützen uns überhaupt nichts, wenn 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: »AR-Flag-Unterstützung in DirectShow« unter *http://brother-john.net/ar-vergleich.html*.

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. Nur beim klassischen nicht-anamorphen Encoding müssen wir das Bild nach den Regeln in diesem Kapitel skalieren. Für anamorphe Encodings ist zumindest der Abschnitt über den

Zusammenhang zwischen Auflösung und Qualität (Seite 55) wichtig. Anhand dessen können wir entscheiden, ob der Speicherplatz für ein anamorphes Bild überhaupt ausreicht.

Ein Beispiel

Nehmen wir wieder *Die fabelhafte Welt der Amélie* als Beispiel. Von der ursprünglichen PAL-Auflösung von 720 × 576 Pixeln bleiben nach dem Cropping (je 4 Pixel links und rechts, je 72 Pixel oben und unten) noch 712 × 432 Pixel übrig. Angenommen, wir entscheiden uns im fertigen Encoding für eine horizontale Auflösung von 608 Pixeln. Das entspricht 85,4 % der ursprünglichen 712 Pixel. Die Höhe um den gleichen Faktor geschrumpft ergibt eine Zielauflösung von 608 × 369 Pixeln, und Frame Nummer 160 932 sieht aus wie Abbildung 14. Sofort fällt auf, dass diese extrem langgezogene Kopfform kaum der Realität entsprechen kann. Was wir bisher nicht bedacht haben, ist das verzerrt gespeicherte Bild der DVD. Berücksichtigen wir das korrekt, erhalten wir eine Zielauflösung von 608 × 256 Pixeln wie in Abbildung 15. Das sieht nicht nur deutlich natürlicher aus, sondern ist auch korrekt.

Berechnung der Zielauflösung

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

Zusätzlich zur eigentlichen Rechnung müssen wir auch das Mod16-Kriterium von Seite 208 im Kopf behalten. Die Zielauflösung muss in beiden Dimensionen glatt durch 16 teilbar sein.

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

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

- **Zielbreite:** Kein Problem hier. Diese Zahl haben wir ja gerade selbst festgelegt. An dieser Stelle setzen wir einfach unsere 608 Pixel ein.

Abbildung 14: Falsches Resizing ohne Berücksichtigung der Verzerrung.

Abbildung 15: Korrektes Resizing mit Berücksichtigung der Verzerrung.

- **AR nach Cropping:** Bezeichnet das Seitenverhältnis des Bildes nach dem Cropping der schwarzen Balken. An dieser Stelle brauchen wir uns um Verzerrungen noch keine Gedanken machen. AR bedeutet hier einfach das Verhältnis von gecroppter Breite zu gecroppter Höhe: 712/432 in unserem Fall.
- **PAR:** Hier kümmern wir uns schließlich darum, das Bild korrekt zu entzerren. Wir wissen, dass es sich um eine 16:9-PAL-DVD handelt. Das steht meistens auf der Verpackung. Auch DGIndex liefert uns beim Indexieren des Videos (vgl.

Seite 127 f.) die nötigen Informationen. Das richtige PAR brauchen wir dann nur noch aus der Tabelle auf Seite 44 ablesen: 16/11.

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

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

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

Abweichung vom optimalen AR: Aspect Error

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

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

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

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

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

Auflösung und Qualität

Mit diesem Wissen ausgerüstet, können wir nun daran gehen, eine günstige Auflösung zu wählen. Günstig bedeutet: eine Auflösung, die (neben einem geringen AR-Fehler) gute Qualität bietet. 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 Tabelle 6.

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

Tabelle 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. Deswegen stimmt das »nutzlos« in der Tabelle ganz genau.

Interessanter ist die **relative Bitrate**, besser bekannt als **BPF-Wert**. Der gibt an, wie viele Bits zum Encodieren eines einzelnen Pixels in jedem einzelnen 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 **Kompres-**

sionstest. 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 amit 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 der Tabelle 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

Tabelle 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 AC3) bzw. zwei kleineren. Die richtig hohen Zielgrößen sollten nahezu immer den Platz für anamorphes Bild und mindestens eine AC3 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 ist die Konfiguration der Videocodecs. Welches Kapitel für die Praxis interessanter ist, hängt vom Encoding-Frontend ab. Die empfohlenen Einstellungen unterscheiden sich jedenfalls nicht zwischen VfW- und Kommandozeilen-Kapitel.

A.4.1 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 Optionen der CLI-Encoder folgen, brauchen wir deshalb auch nur intensiv zu lesen, wenn wir tatsächlich ohne Encoding-Frontend direkt an der Konsole arbeiten wollen.

VfW-Interfaces

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

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

Alle drei Encodingwissen-Codecs (Xvid, DivX, x264) sind in einer VfW-Version

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

Das Kommandozeilen-Interface (CLI)

Die Verknüpfung, die im Windows-Startmenü zur Kommandozeile führt, nennt sich auch unter Windows XP noch immer *MS-DOS-Eingabeaufforderung*, obwohl es sich längst nicht mehr um ein DOS handelt. Lediglich das Prinzip ist geblieben. Auf der Kommandozeile werden Befehle in Textform eingetippt und ausgeführt. 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, müssen angegeben werden. 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.

A.4.2 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.

Wir beschränken uns auf die Optionen des Encoders, die fürs DVD-Backup nützlich sind. Wer an einer kompletten Erklärung interessiert ist, sollte sich Selurs *Wissenswertes rund um Xvid* nicht entgehen lassen und auch einen Blick in crusty's (englische) *Xvid FAQ* werfen. Beide erklären Xvid anhand des VfW-Interfaces. Für den Kommandozeilen-Encoder sind sie trotzdem genauso nützlich, weil der sich im Funktionsumfang kaum unterscheidet.

Konfiguration von B-Frames

Das Konzept der B-Frames haben wir im Kapitel über die Interframe-Kompression auf Seite 10 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 Bei-

spiel. 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:

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

$BQuant_f$ 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 IPBBBI, würde das nichts ändern. Das relevante Nachfolger-Frame wäre eben anstatt einem P- ein I-Frame.

Nehmen wir einmal an, das Vorgänger-P-Frame hätte einen Quantizer von 2,0 erhalten und das Nachfolger-P-Frame einen Quantizer von 3,0. Außerdem hätten wir

die Stellschrauben auf Xvids Standardeinstellungen belassen (Ratio = 1,50 und Offset = 1,00). Dann würde sich für unsere B-Frames ein Quantizer von 4,75 ergeben.

$$BQuant_f = \frac{2{,}0 + 3{,}0}{2} \times 1{,}50 + 1{,}00 = 4{,}75$$

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*: Ratio 1,63 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.

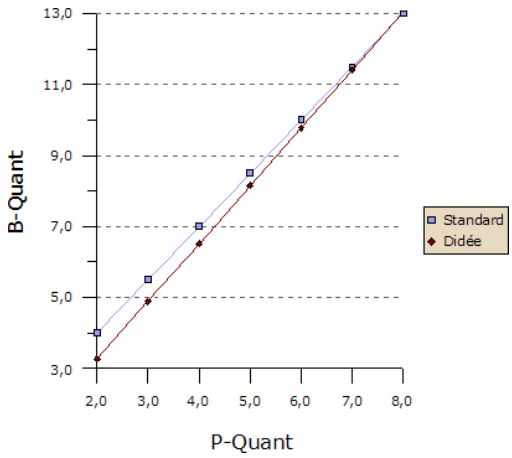

Abbildung 16: Vergleich zwischen Didées und Standard-B-Frame-Quantisierung.

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 Abbildung 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 Wertepaare, 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.

Quantisierungsmatrizen

Ein weiteres wichtiges und oft unterschätztes Tool sind die Quantisierungsmatrizen. Welche Bedeutung denen mathematisch genau zufällt, ist auch fürs Encodingwissen zu viel der Hintergrundinformation. Wir sollten 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 Part 2 und damit Xvid kennt zwei grundsätzliche Arbeitsmodi. Der eine nennt sich H.263 und arbeitet genau genommen nicht mit einer Matrix im eigentlichen Sinn. 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 zur H.263-Matrix: 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 Custom-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 Origi-

nals erzeugen. Die folgende Tabelle 8 bietet eine Auswahl gerne genutzter Matrizen, absteigend sortiert nach Komprimierbarkeit (siehe SpikeSpiegels Matrizen-Test auf Doom9.org im Thread *CQMs Compressibility TEST*).

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	Etwa 15 Prozent niedrigere Kompression als die Standard-MPEG-Matrix. Bestens geeignet für ½-DVD-Backups. Die EQM V3HR ist HVS-optimiert.
SixOfNine	Didée	Deutliche, über 50 % niedrigere Kompression als die Standard-MPEG-Matrix. Damit gehört die SixOfNine klar in den Bereich höherer Datenraten. Dort lässt sie sich sehr flexibel einsetzen. Im Gegensatz zu den anderen vorgestellten Matrizen sollten wir sie immer mit speziell abgestimmten Xvid-Einstellungen verwenden. 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.

Tabelle 8: Eine Auswahl beliebter CQMs.

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 Matrizen: meistens die

EQM V3HR oder die V3LR, wenn es eng wird. 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« wieder einmal 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). 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.

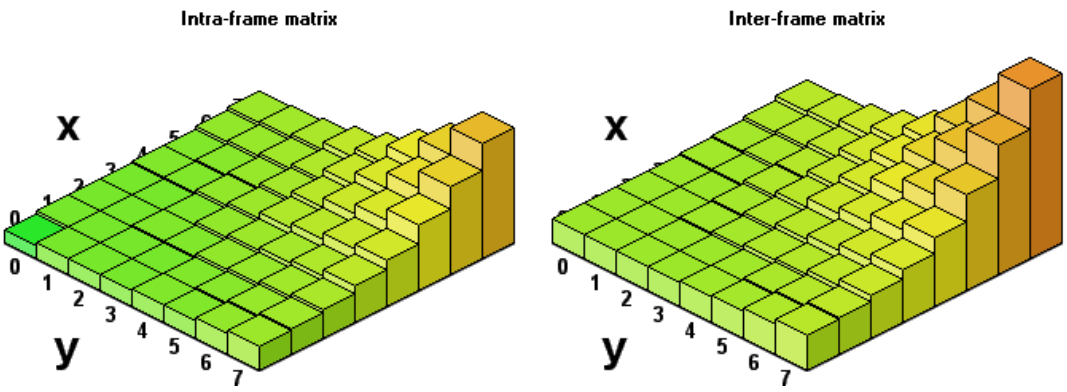

Abbildung 17: Grafische Darstellung der EQM V3HR (Quelle: LigHs CQM-Editor).

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 sicher die bessere Wahl.

Die Profile von MPEG-4 Part 2

MPEG-4 Part 2 ist ein technisch 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* schließt dabei Custom-Matrizen mit ein. *AQ* steht für Adaptive Quantization bzw. Lumimasking. Mit Hilfe des *VBV* (Video Buffer Verifier) wird die Bitrate kontrolliert. 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, hält sich Xvid an diese Beschränkungen, auch auf Kosten der Qualität. Die dritte Tabelle 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.

	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

Tabelle 9: MPEG-4 Part 2 Encoder-Tools

	Max. Puffergröße (Bits)	Max. Bitrate (kbit/s)
Simple @ L0	163 680	64
Simple @ L1	163 680	64
Simple @ L2	654 720	128
Simple @ L3	654 720	384
Advanced Simple @ L0	163 680	128
Advanced Simple @ L1	163 680	128
Advanced Simple @ L2	654 720	384
Advanced Simple @ L3	654 720	768
Advanced Simple @ L4	1 309 440	3 000
Advanced Simple @ L5	1 833 216	8 000

Tabelle 10: MPEG-4 Part 2 Video Buffer Verifier (VBV)

	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

Tabelle 11: MPEG-4 Part 2 Auflösung und Bildrate

A.4.2.1 Xvid v1.2 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* und crusty's (englische) *Xvid FAQ* kann ich jedem nur ans Herz legen.

Einstellungen für den 1st Pass

Profile und deren Level schalten – wie auf Seite 64 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 (Abbildung 18). Über den **More**-Button erreichen wir den Dialog für die Feineinstellungen (Abbildung 19).

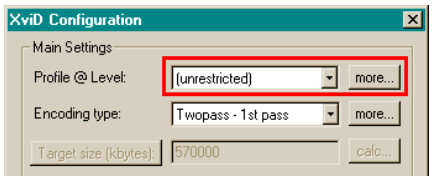

Abbildung 18: Hauptfenster oben der Xvid-Konfiguration.

Der **Quantization type** bestimmt die Matrix, die beim Encodieren verwendet wird. **MPEG** erzeugt ein etwas schärferes Bild auf Kosten der Kompression und eignet sich eher für die höhere Datenrate einer großzügigen Zielgröße. **H.263** ergibt dagegen ein etwas weicheres Bild und erhöht damit die Kompression ein wenig; geeignet eher für die niedrigen Datenraten von 1-CD-Encodings. Über **MPEG-Custom** und den Button **Edit Matrix** können wir eine Custom-Matrix laden. Näheres zum Thema haben wir ab Seite 64 schon besprochen.

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

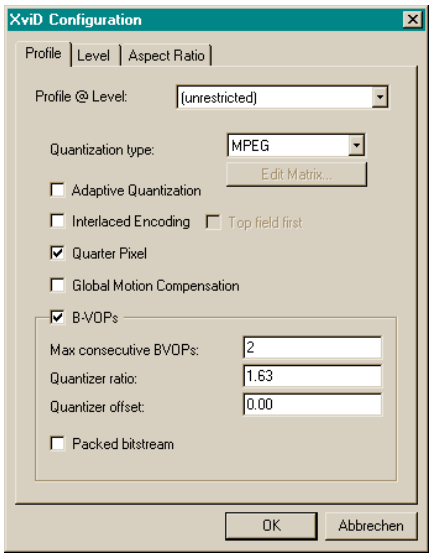

Abbildung 19: Xvids MPEG-4-Profileinstellungen.

Quarter Pixel erhöht die Genauigkeit, mit der Bewegungen gespeichert werden, von einem halben auf ein Viertel Pixel. Das wirkt sich positiv auf Details und Schärfe aus, deshalb sollten wir QPel normalerweise einschalten, obwohl sich die Anforderungen an die Rechenleistung nicht nur beim Encoding, sondern auch beim Decoding erhöhen.

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 oft fraglich bleibt. Dass im technisch fortschrittlicheren AVC-Standard GMC nicht verfügbar ist, deutet zusätzlich 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 61 beschäftigt. In der Regel empfehle ich ein Maximum von **2**, Ratio **1.63** und Offset **0.00**.

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.

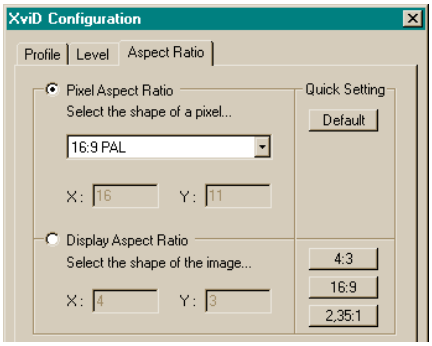

Abbildung 20: Xvids AR-Konfiguration.

Damit wechseln wir zum Register Aspect Ratio (Abbildung 20). Unter **Pixel Aspect Ratio** erlaubt Xvid die Angabe eines PAR, der im AR-Flag des MPEG-4-Streams gespeichert wird. Das ist nur für anamorphe Zieldateien interessant. Ein klassisches Encoding mit quadratischen Pixeln verwendet immer die Standardeinstellung **Square**. Die nicht-quadratischen Einträge entsprechen den Werten gemäß ITU-R BT.601 (vgl. Tabelle 4, Seite 44). Außerdem können wir mit **Custom** die beiden Eingabefelder unter der Combobox aktivieren, um einen beliebigen PAR anzugeben.

Die zweite Möglichkeit ist die Angabe des **Display Aspect Ratio**. Bis vor kurzem hieß diese Option noch verwirrenderweise *Picture Aspect Ratio*, hat aber

schon immer das DAR bezeichnet. Wenn wir ein DAR angeben, ist wichtig, immer ganze Zahlen zu verwenden. 2,35:1 müssten wir also als 235:100 angeben.

Damit ist dieses Fenster abgehandelt und wir können zurück in den Hauptdialog. Mit **Twopass - 1st pass** stellen wir den ersten Codierdurchgang ein (Abbildung 21) und rufen über den **More**-Button die Details auf (Abbildung 22). Im angegebenen **Stats file** werden die Informationen aus dem 1st Pass gespeichert, die der 2nd Pass dann weiterverwendet. Normalerweise kümmert sich das Encoding-Frontend um den richtigen Eintrag. **Full quality first pass** aktiviert sämtliche eingestellten Encoder-Optionen schon im ersten Durchgang. Normalerweise sind hier einige Optionen abgeschaltet (z. B. VHQ), die nur bremsen und im 1st Pass noch nichts bringen. Die Option sollten wir nur dann einschalten, wenn wir die Videodatei des 1st Pass behalten wollen. Wollen wir aber nicht. Deshalb setzen wir auch den Haken bei **Discard first pass**. Xvid schreibt dann keine komplette Videodatei sondern nur ein wenig Müll. Das spart Plattenplatz.

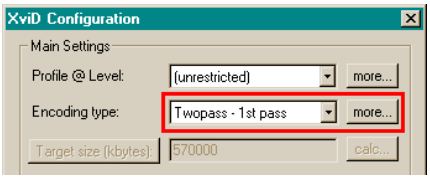

Abbildung 21: Einstellung für 1st Pass im Xvid-Hauptdialog

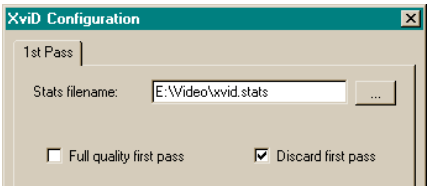

Abbildung 22: Xvids 1st-Pass-Optionen.

Damit kehren wir zurück ins Hauptfenster und kümmern uns um die Zones (Abbildung 23). **Zones** definieren Abschnitte innerhalb des Videos, für die unabhängig eine Reihe von Optionen definiert werden können. Offensichtlichster Anwendungsbereich für Zonen sind die Credits. Eine Zone, die sich über den ganzen Film erstreckt, existiert immer. Über den Button **Zone Options** gelangen wir in den Konfigurationsdialog.

Bei **Start frame #** (Abbildung 24) tragen wir das Bild ein, mit dem die Zone beginnen soll. Für die erste Zone ist das **0**, der Filmanfang. Das Ende einer Zone ist immer der Anfang der darauf folgenden oder das Ende des Films.

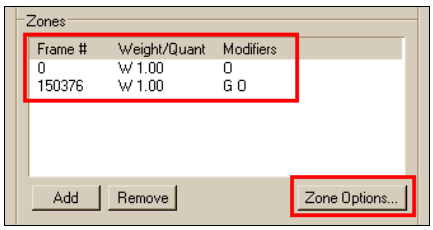

Abbildung 23: Zonenliste im Xvid-Hauptdialog.

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

Begin with a keyframe erzwingt am Anfang der Zone ein I-Frame. **Greyscale en-**

coding 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.

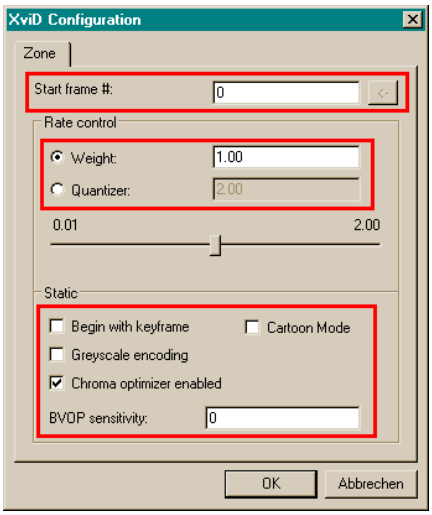

Abbildung 24: Detailkonfiguration der Xvid-Zonen.

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

Über **OK** gelangen wir zurück in den Hauptdialog, wo wir uns im unteren Teil des Fensters um den **More**-Abschnitt kümmern (Abbildung 25). Hier stellen wir das **Quality Preset** auf (**User defined**) und öffnen über den **More**-Button den Dialog für die Detaileinstellungen (Abbildung 26).

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

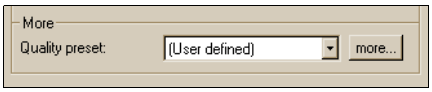

Abbildung 25: »Quality Preset« im Xvid-Hauptdialog.

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

einen spürbaren Effekt auf die Qualität haben. Da B-Frames meistens den größten Anteil der Frames in einem Film stellen, sollten wir das B-Frame-VHQ möglichst nicht deaktivieren.

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

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

Von der **Frame drop ratio** sollte jeder die Finger lassen, der nicht ganz genau weiß, was er da tut. Auch das **Max. I-frame interval** kann problemlos auf dem Standardwert bleiben. An günstigen Stellen setzt Xvid sowieso automatisch I-Frames (Man nennt sie auch Keyframes). Ein sehr niedriger Wert macht höchstens bei Captures direkt in Xvid Sinn, die man hinterher noch schneiden will. Denn Schneiden funktioniert nur an I-Frames. Genaueres dazu steht im Kapitel C.1.1 über manuelles Splitting ab Seite 201.

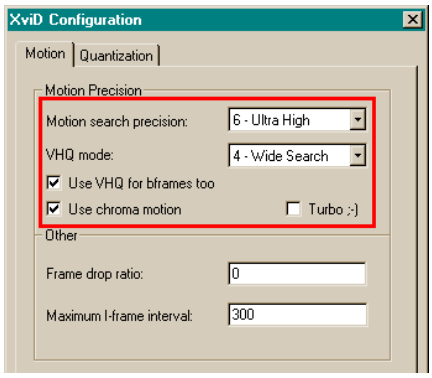

Abbildung 26: Konfiguration der Xvid-Featues.

Im Register **Quantization** (Abbildung 27) 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.

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

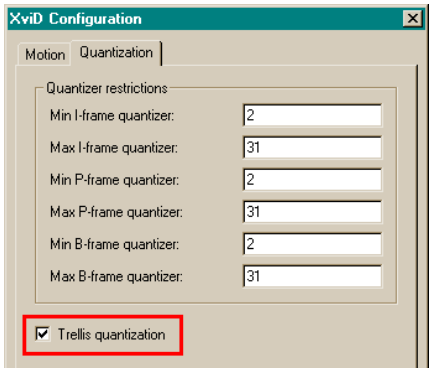

Abbildung 27: Xvid-Quantisierungsoptionen.

tät 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 gar ein wiederholtes Encoding mit größerer Auflösung ist allemal besser als mit Quantizer 1 verschwendete Dateigröße.

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

Number of threads in Abbildung 28 ist erst seit Xvid Version 1.2 verfügbar. Hier stellen wir ein, wie viele Threads Xvid zum Encodieren verwenden soll. Für Computer mit nur einer CPU sollten wir diesen Wert immer auf **0** belassen. Wer Hyperthreading, Dual Core oder tatsächlich ein System mit mehreren Prozessoren sein eigen nennt, muss ein wenig testen, welche Einstellung die schnellsten Ergebnisse bringt. Die Anzahl der CPUs oder CPU-Kerne ist ein guter Startwert.

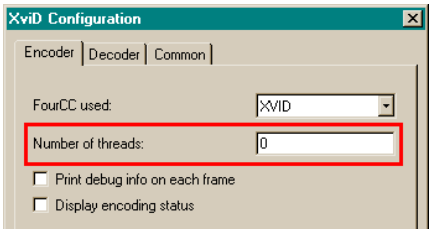

Abbildung 28: Anzahl Encodingthreads ab Xvid Version 1.2.

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

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

Einstellungen für den 2nd Pass

Im Hauptdialog stellen wir **Twopass - 2nd pass** ein. In der Regel kümmert sich das Encoding-Frontend um die richtige **Target size**. Wenn wir sie doch manuell eingeben: Hierhin gehört die Größe der reinen Videospur, nicht die Zielgröße des kompletten Films – 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.

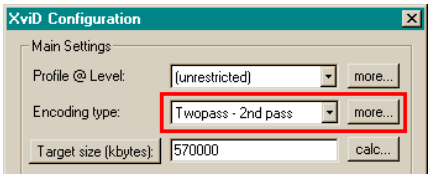

Abbildung 29: Einstellung für 2nd Pass im Xvid-Hauptdialog.

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

A.4.2.2 Xvid Kommandozeilen-Konfiguration

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

Xvid_Encraw ist in einem normalen Xvid-Download nicht enthalten. Auch die Entwicklung erfolgt nicht direkt im Rahmen des Xvid-Projekts. Für aktuelle Details und die aktuelle Version sollten wir den *Entwicklungsthread* im Doom9-Forum verfolgen. Xvid_Encraw 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

Ein 2-Pass-Encoding mit Xvid_Encraw 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.

1st Pass

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -pass1 "<Statistikdatei>"
-framerate <fps> -overhead 0 -progress 50 [weitere Optionen]
```

Die erste Option übergibt an Encraw den Dateinamen der Quelldatei, die zweite definiert, dass es sich um ein AviSynth-Skripts 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. Die Option `-framerate` legt die Bildrate der Zieldatei fest, die mit derjenigen des AVS-Skripts identisch sein muss. Danach schalten wir mit `-overhead` Xvids automatische Berücksichtigung des Containeroverheads ab, die mit der Berechnung der Zielgröße in Konflikt geraten würde. Abschließend 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.

2nd Pass

```
xvid_encraw.exe -i "<Quell-AVS-Datei>" -type 2 -pass2 "<Statistikdatei>"
{-o|-avi|-mkv} "<Zieldatei>" -size <Zielgröße in KByte> -framerate <fps>
-par <PAR> -overhead 0 -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 eine der drei Optionen `-o`, `-avi` oder `-mkv` gefolgt vom Dateinamen des Zielvideos. Mit `-o` erzeugen wir einen rohen MPEG-4-Datenstrom (Dateiendung .m4v), was dann nötig ist, wenn der fertige Film im MP4-Container liegen soll. Für AVI oder Matroska als Zielcontainer wählen wir entsprechend `-avi` oder `-mkv`. 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. Mit `-par` legen wir anschließend das Pixel Aspect Ratio des Zielvideos fest. Für ein klassisches Encoding mit quadratischen Pixeln gilt immer `-par 1:1`, 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.

Beispiel einer Konfiguration

Um ein wenig mehr Gefühl für Encraw zu bekommen, betrachten wir ein Beispiel, das exakt die Konfiguration aus den Screenshots des Xvid-VfW-Kapitels (Seite 69) nachbildet. Dazu stellen wir uns eine fiktive 16:9-PAL-DVD vor, die 100 Minuten Spielzeit aufweist und mit voller anamorpher Auflösung incl. zwei AC3-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 Xvid_Encraw aus, d. h. wir tippen nur das, was auch tatsächlich ausdrücklich angegeben werden muss. Wer aus der VfW-Welt kommt, sollte im Hinterkopf behalten, dass Xvid VfW und Xvid_Enraw nicht überall die gleichen Standardwerte verwenden.

1st Pass

```
xvid_encraw.exe -i "D:\Quelle.avs" -type 2 -pass1 "D:\xvid.stats"
-framerate 25.0 -overhead 0 -progress 50 -bquant_ratio 163
-bquant_offset 0 -bvhq -qtype 1 -qpel -vhqmode 4 -zones 0,w,1,0
```

2nd Pass

```
xvid_encraw.exe -i "D:\Quelle.avs" -type 2 -pass2 "D:\xvid.stats"
-mkv "D:\Zielvideo.mkv" -size 1730175 -par 16:11 -framerate 25.0
-overhead 0 -progress 50 -bquant_ratio 163 -bquant_offset 0 -bvhq
-qtype 1 -qpel -vhqmode 4 -zones 0,w,1,0
```

Referenz der Optionen

Dieser Abschnitt listet mit einer kurzen Erklärung viele Optionen auf, die für Xvid_Encraw 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 Xvid_Encraw 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.

`-w <Breite>`
Werte: `1` bis `2048`
Beispiel: `-w 720`
Horizontale Auflösung der Quelldatei. Kann für AviSynth-Quellen automatisch ermittelt werden, wir müssen den Schalter also nicht angeben.

`-h <Höhe>`
Werte: `1` bis `2048`
Beispiel: `-h 576`
Vertikale Auflösung der Quelldatei. Kann für AviSynth-Quellen automatisch ermittelt werden, wir müssen den Schalter also nicht angeben.

`-frames <Anzahl>`
Standardwert: alle Frames
Beispiel: `-i 10000`
Anzahl der Frames, die encodiert werden sollen. Dient hauptsächlich dazu, nur einen Teil der Quelle zu encodieren. Das Beispiel würde den Encodingvorgang nach den ersten 10000 Frames beenden.

`-start <Startframe>`
Standardwert: `0`
Beispiel: `-start 1000`
Framenummer, bei der der Encodingvorgang starten soll. Normalerweise ist das das erste Frame der Quelldatei. Durch die Angaben von sowohl `-start` als auch `-frames` kann nur ein Teilstück aus der Mitte des Films encodiert werden. Alternativ könnten wir auch einen entsprechenden `Trim()`-Befehl ins AviSynth-Skript aufnehmen, um den gleichen Effekt zu erreichen.

Zieldatei

`-o <Zieldateiname>`
Beispiel: `-o "D:\Pfad zur\Zieldatei.m4v"`
Speichert die Zieldatei als rohen (containerlosen) MPEG-4-Datenstrom. Fürs DVD-Backup ist das nützlich, wenn der fertige Film im MP4-Container verpackt sein soll.

`-avi <Zieldateiname>`
Beispiel: `-avi "D:\Pfad zur\Zieldatei.avi"`
Speichert die Zieldatei im AVI-Container.

`-mkv <Zieldateiname>`

Beispiel: `-mkv "D:\Pfad zur\Zieldatei.mkv"`
Speichert die Zieldatei im Matroska-Container.

`-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 3` 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 50 die Rede ist. Für klassische Encodings mit quadratischen Pixeln geben wir immer `-par 1:1` an oder lassen die Option ganz weg.

`-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. Eigentlich sollte Xvid_Encraw den richtigen Wert automatisch von AviSynth beziehen. Da es damit aber schon Probleme gab, sollten wir die Optionen immer ausdrücklich angeben, um Problemen aus dem Weg zu gehen.

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 in der Regel der Standardwert `2`. Mit den B-Frame-Optionen insgesamt haben wir uns im allgemeinen Xvid-Kapitel ab Seite 61 schon ausführlich beschäftigt.

`-bquant_ratio <Verhältnis>`

Standardwert: `150`
Beispiel: `-bquant_ratio 163` (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 1,5. Siehe auch Seite 61 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 1,0. Siehe auch Seite 61 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. Ist nur wirksam, wenn `-avi` als Zieldatei gewählt ist. Für alle anderen Zielformate ist Packed Bitstream nie aktiv. Nützlich ist die Funktion dann, wenn wir den Film manuell splitten wollen.

`-qpel`

Schaltet die Quarter-Pixel-Bewegungssuche ein. QPel erhöht die Genauigkeit, mit der die Bewegungssuche arbeitet, von einem halben auf ein Viertel Pixel. Das wirkt sich positiv auf Details und Schärfe aus, deshalb sollten wir QPel normalerweise einschalten, obwohl sich die Anforderungen an die Rechenleistung nicht nur beim Encoding, sondern auch beim Decoding erhöhen.

`-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 oft fraglich bleibt. Dass im technisch fortschrittlicheren AVC-Standard GMC nicht verfügbar ist, deutet zusätzlich darauf hin, dass sich die Funktion nicht bewährt hat.

`-lumimasking`

Aktiviert den Lumimasking-Algorithmus, besser bekannt als Adaptive Quantization. In sehr hellen und sehr dunklen Bereichen des Bilds nimmt das

menschliche Auge weniger Details wahr. Das macht sich Lumimasking zu nutze und komprimiert solche Bereiche stärker, um die eingesparte Datenrate an Stellen zu verwenden, die es nötiger haben. Bei geringen Bitraten macht sich das positiv bemerkbar.

`-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 64 schon besprochen.

`-qmatrix <Matrixdateiname>`

Beispiel: `-qtype 1 -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. Zusätzlich sollte auch die Option `-qtype 1` angegeben werden. Siehe auch Seite 64 f.

`-interlaced [<Modus>]`

Werte: `1` (Bottom Field First, Standard), `2` (Top Field First)

Beispiele: `-interlaced` oder `-interlaced 2`

Aktiviert interlaced Encoding. Wird die Option wie im Beispiel ohne Attribut angegeben, behandelt Xvid das Video als Bottom Field First. Die Option korrekt zu setzen, ist wichtig, da Xvid nicht prüft, ob das Video tatsächlich den gemachten Angaben entspricht.

Fürs DVD-Backup ist die Option weniger wichtig, da in der Regel schon vorher ein Deinterlacing durchgeführt wird.

`-vbvsize <Bits>`

Mögliche Werte: Siehe Seite 67 zu den Profilen von MPEG-4 Part 2.

Standardwert: nicht gesetzt.

Beispiel: `-vbvsize 1833216`

Größe des Video Buffer Verifier (VBV).

`-vbvmax <kbit/s>`

Mögliche Werte: Siehe Seite 67 zu den Profilen von MPEG-4 Part 2.
Standardwert: nicht gesetzt.
Beispiel: `-vbvmax 8000`
Maximal mögliche Bitrate in Kilobit pro Sekunde.

`-vbvpeak <Bits>`

Maximale Anzahl Bits in einem beliebigen 1-Sekunden-Intervall. Dieser Wert ist nicht Bestandteil von MPEG-4. Er dient zur Nachbildung der DivX-Profile.

Encoding-Modus

`-single`

Aktiviert das Encoding im Single-Pass-Modus. Da das der Standardwert ist, muss diese Option nie ausdrücklich angegeben werden.

`-cq <Quantizer>`

Werte: `1.0` bis `31.0`
Beispiel: `-cq 2`
Führt ein Single-Pass-Encoding mit einem bestimmten Quantizer durch. Ist interessant für äußerst hochqualitative Backups auf eine ganze DVD-5. Da 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.

`-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. Normalerweise 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 Xvid_Encraw 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 der geringsten Anzahl Bits. Generell gilt: Je höher der VHQ mode desto besser die Qualität und desto langsamer das Encoding. `1` oder `2` bringen im Vergleich zum Bremseffekt den höchsten Qualitätsgewinn. Ich bin Qualitätsfanatiker genug, um meistens bei der höchsten Einstellung `4` zu bleiben. Wenn wir GMC aktiviert haben, sollte VHQ nicht abgeschaltet sein.

`-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 meistens ausgeschaltet.

Zonen

Zonen definieren Abschnitte innerhalb des Videos, für die unabhängig eine Reihe von Optionen definiert werden können. Offensichtlichster Anwendungsbereich für Zonen sind die Credits. Zur 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 Zonen 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 (vereinfacht: Kompressionsfaktor) 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 Xvid_Encraw

`-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 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.4.2.3 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 Tabelle 12, 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 60 zuständig.

Hauptfenster

Profile @ Level	`-vbvsize` und `-vbvmax`. Außerdem `-vbvpeak` für DivX-Profile.
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` mit Kommazahl. Eine ganze Zahl entspricht konstantem Quantizer.
Zone Options	`-zones`

Profile @ Level / More / Profile

Quantization type	`-qtype` bei Custom-Matrizen zusätzlich `-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. Wir sollten sowieso das PAR verwenden.

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`

Tabelle 12: Zuordnung der Optionen von Xvid VfW und Xvid_Encraw.

A.4.3 Die x264-Konfiguration

X264 ist längst den Kinderschuhen entwachsen und schwer dabei, Xvid den Rang als beliebtester Open-Source-Codec abzulaufen. Und es ist tatsächlich abzusehen, dass für DVD-Backups die Tage der ASP-Codecs gezählt sind. Außerdem treibt x264 durch die starke Konzentration auf seinen Kommandozeilenencoder die Bewegung weg von VfW voran, was man nur uneingeschränkt gutheißen kann.

Für die Besprechung sämtlicher Optionen ist wieder einmal Selur zuständig. Sein *Wissenswertes rund um x264* beschäftigt sich mit dem VfW-Interface, und *man x264* bespricht den Kommandozeilen-Encoder. Wir beschränken uns wieder auf die fürs DVD-Backup wichtigen Einstellungen. Wie schon im Xvid-Kapitel besprechen wir zuerst alle Funktionen, die eine etwas ausführlichere Erklärung verdienen. In den anschließenden Kapiteln sind die empfohlenen Werte immer angegeben, so dass wir uns nicht unbedingt erst durch dieses doch recht umfangreiche Kapitel wühlen müssen.

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, der allerdings durch das große Potenzial zu Erhöhung der Kompression wieder aufgewogen wird. 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.

In der Konfiguration können wir einstellen, welche Partitionsgrößen x264 überhaupt berücksichtigt. Die erste Reihe in Abbildung 30 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.

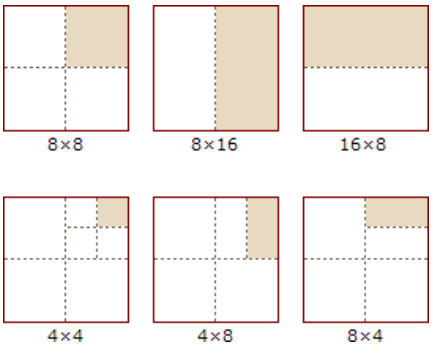

Abbildung 30: Partitionierung von Makroblocks in H.264.

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 enormer Geschwindigkeitsverlust verbunden, andererseits entsteht durch die dynamische Partitionierung ein riesiges Potenzial zu Verbesserung der Bewegungssuche und Bewegungskompensation. 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 10 angesprochen. Hier beleuchten wir die bidirektionalen Bilder aus der Sicht von x264, denn der Encoder bietet uns die gewaltige Zahl von neun 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 x264 selbst die günstigste Möglichkeit.

Diese automatische Auswahl lässt sich beeinflussen (B-Frame-Bias). Wir haben die Möglichkeit, x264 zum verstärkten B-Frame-Einsatz zu drängen oder ihn eher davon abzuhalten. Außerdem können wir dem Encoder vorschreiben, ob er bei der Entscheidung für ein B-Frame die aktuellen Gegebenheiten des Videos berücksichtigen darf (adaptive B-Frames). Und zuletzt bleibt die Einstellung, ob B-Frames selbst wieder als Referenzbilder dienen dürfen.

Für das 1-CD-Encoding sollten wir mit B-Frames nicht sparen. 5 Stück dürfen es als Maximum schon sein. Noch höher zu gehen, bringt allerdings kaum einen Qualitätsgewinn. Im Gegenteil wächst dadurch die Gefahr, dass wegen der höheren Kompression der B-Frames Artefakte sichtbar werden. 2-CD-Encodings kommen gut mit etwa 3 B-Frames aus, und für noch höhere Zielgrößen können wir auch auf 2 herunter gehen. Sie ganz zu deaktivieren, ist in der Regel nicht sinnvoll. Denn B-Frames sind ein Tool, das die Dateigröße deutlich in den Keller ziehen kann ohne die Qualität spürbar zu beeinflussen. Entsprechend groß fällt der Nachteil aus, wenn wir sie abschalten.

x264s Lust auf B-Frames benötigt normalerweise keine Anpassung. Auch sollten wir die adaptive Verteilung zulassen, da nur so B-Frames auf eine »intelligente« Art und Weise gesetzt werden können. »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. Auch die B-Pyramide (B-Frames als Referenz) senkt die Geschwindigkeit nicht allzu wesentlich. Da auch der Qualitätsgewinn nicht überragend ausfällt, können wir die Funktion bei höheren Zielgrößen auch abschalten und damit der Encodingzeit ein wenig Gutes tun.

Quantisierung und Bewegungssuche

Um die Quantisierung und Bewegungssuche in B-Frames zu beeinflussen, stehen uns noch einmal fünf Optionen zur Verfügung. Eine davon betrifft die Partitionierung der Makroblocks, womit wir uns weiter oben schon beschäftigt haben. Als weiteren wichtigen Punkt haben wir den *Direct Mode*, der angibt, nach welchem Verfahren B-Frames komprimiert werden. Der temporale Modus verwendet zur Komprimierung die zeitlichen Änderungen zwischen den Bildern, der spatiale die räumlichen innerhalb des Frames.

Um Aus- und Überblendungen exakter zu speichern, können wir eine gewichtete Bewegungssuche verwenden, bei der die gefundenen Referenzen in einer beliebig gewichteten Mischung verwendet werden dürfen. Mit einer weiteren Option kann die Suche nach Referenzen zusätzlich erweitert werden (bidirektionale ME), was die Chance auf bessere Ergebnisse bietet, aber Geschwindigkeit kostet.

Schließlich besitzt x264 ein B-Frame-Feature namens *Rate Distortion Optimization* (RDO), das mit Xvids VHQ für B-Frames vergleichbar ist. Auch hier werden für die Makroblocks verschiedene Szenarien durchgerechnet und dasjenige mit der geringsten Bitanzahl ausgewählt. Allerdings benötigt diese Funktion zwingend einen der genaueren und damit langsameren Modi für die allgemeine Bewegungssuche. B-Frame-RDO ist deshalb mit deutlichen Geschwindigkeitseinbußen verbunden.

Wollen wir die beste Kompression und beste Qualität herausholen, sollten natürlich sämtliche B-Frame-Features aktiviert sein. Allerdings bremst das den Encodingvorgang deutlich aus. Eher geringen Einfluss auf die Qualität hat die gewichtete und bidirektionale Bewegungssuche, weshalb sich diese Kandidaten besonders bei höheren Zielgrößen zum Deaktivieren anbieten. B-Frame-RDO sollten wir uns dagegen möglichst gönnen, da es sich spürbar auf die Qualität auswirkt. Und um den *Direct Mode* müssen wir uns zum Glück nicht unbedingt kümmern, denn x264 besitzt eine Automatik, die je nach Situation selbst den zeitlichen oder räumlichen Modus wählt.

Der Inloop-Deblocking-Filter

Deblocking-Filter kennen wir 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. x264 setzt einen Deblocking-Filter schon beim Encodieren ein, und zwar nachdem ein Bild codiert wurde, aber

bevor es als Referenz für das nächste Bild dient. Dadurch kann Bild 2 mit einer Referenz arbeiten, die weniger Artefakte enthält. Das tut der Qualität gut. Der Nachteil ist die sinkende Geschwindigkeit, um zwar sowohl beim Encoding als auch beim Decoding. Denn damit Bild 2 korrekt decodiert werden kann, muss das vorangehende Bild in gefilterter Form vorliegen. Das ähnelt einem Xvid-Encoding bei dem wir gezwungen sind, beim Anschauen den Deblocking-Filter zu aktivieren.

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. Werte von 2 sind ein guter Ansatzpunkt. Für höhere Datenraten bietet es sich an, mit schwächeren Werten zu arbeiten (etwa –2). Erst bei äußerst hochqualitativen Encodings im Bereich einer ganzen DVD würde ich dann darüber nachdenken, den Filter ganz abzuschalten.

Natürlich ist es auch möglich, das Deblocking ganz genau abzustimmen, indem wir die Alpha- und Beta-Werte getrennt verändern. Das Thema hat *.mp4 guy im Doom9.org-Thread *How To Use Mpeg4 AVC Deblocking Effectively* etwas näher beleuchtet.

Quantisierungsmatrizen

AVC unterstützt genauso wie MPEG-4 ASP Quantisierungsmatrizen, einschließlich Custom-Matrizen. Im Wesentlichen treffen alle Erklärungen aus dem Xvid-Kapitel (Seite 64 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.

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

Mehrfache Referenzen und IDR-Frames

MPEG-4 Part 2 ist auf ein einzelnes Bild beschränkt, das als Referenz für die Suche nach Bewegung benutzt werden darf (B-Frames auf zwei). MPEG-4 Part 10 (AVC) hat diese Beschränkung nicht mehr. Ein Bild 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.

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 folgende Bildsequenz an: DPIP (D steht für IDR-Frame). 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 zweite P-Frame über das I-Frame hinweg auf das erste P-Frame 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 dem zweiten P-Frame zumindest einen Teil seiner Referenzen wegnehmen. Bildfehler am Anfang der zweiten Datei wären die Folge. Keyframes in dem Sinn, wie wir sie kennen, und damit tauglich als Schnittpunkt, sind nur IDR-Frames, denn eine Referenz über ein IDR-Frame hinweg ist nicht erlaubt.

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.

A.4.3.1 x264 rev600 VfW-Konfiguration

Das VfW-Interface für x264 wird offiziell nicht mehr unterstützt. Aktuelle Versionen von den üblichen Downloadseiten enthalten nur noch den Kommandozeilen-Encoder. Da sich x264 nur mit Hängen und Würgen überhaupt einigermaßen

kompatibel zu VfW verhält und die VfW-Oberfläche eine ganze Reihe Einstellungsmöglichkeiten vermissen lässt, kann ich nur davon abraten. Wenn es doch sein soll, sollten wir die Internetseite von DeathTheSheep United unter der Adresse *deaththesheep.uni.cc* besuchen. Dieses Projekt hat es sich zur Aufgabe gemacht, x264 VfW fortzuführen.

Einstellungen für den 1st Pass

Wie bei den anderen Codecs auch, interessieren uns die Single-Pass-Einstellungen nicht. Im **Bitrate**-Register wählen wir also **Multipass - First Pass (fast)** (Abbildung 31). Der Fast-Modus beschleunigt ähnlich wie bei Xvid den ersten Durchgang, ohne spürbar negativ auf die Qualität zu wirken. Deswegen taugt der langsame First Pass eher für paranoide Qualitätsfanatiker als für den täglichen Gebrauch.

Weiter unten muss **Update Statsfile** angehakt sein. Die Bitrate trägt das Encoding-Frontend automatisch ein. Im Register **Rate Control** können wir bedenkenlos alle Optionen auf den Standardeinstellungen belassen und gleich nach **MBs & Frames** wechseln.

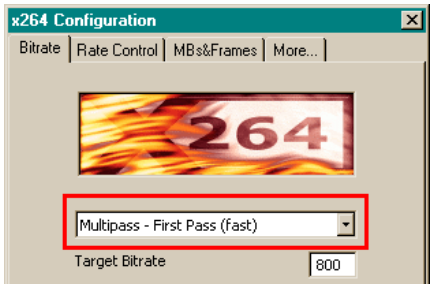

Abbildung 31: Einstellung für 1st Pass im x264-Hauptdialog.

Die Gruppe **Partitions** (Abbildung 32) regelt die mögliche Zerlegung eines Makroblocks. Genauer haben wir die Partitionen ab Seite 89 besprochen. Wie dort gesagt, sollten wir möglichst sämtliche Partitionsgrößen zulassen, da x264 daraus einen wichtigen Teil seiner Effizienz zieht.

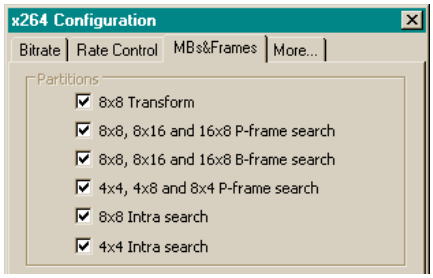

Abbildung 32: Makroblock-Partitionen in x264.

Weiter unten im Register (Abbildung 33) kümmern wir uns um die B-Frames, die wir ab Seite 90 schon intensiv besprochen haben. **Max consecutive** legt die maximale Anzahl aufeinander folgender B-Frames fest. 1-CD-Encodings dürfen klotzen (etwa **5** B-Frames), zu 2-CD-Encodings passen eher **3** und ab ½ DVD sollten **2** B-Frames ausreichen.

Ein Haken bei **Adaptive** aktiviert die »intelligente« Verteilung der B-Frames. **Bias** darf ruhig auf der **0** stehen bleiben. **Use as reference** (die B-Frame-Pyramide), **Bidirectional ME** und **Weighted biprediction** können alle zur Steigerung der Geschwindigkeit recht problemlos deaktiviert bleiben. Gerade für hohe Ziel-

größen bietet sich das an. Dagegen kann das 1-CD-Encoding die zusätzliche Effizienz gut vertragen, auch wenn das Encoding dann ein Stück länger dauert.

Direct Mode schließlich gibt an, ob zeitliche oder räumliche Informationen zur Komprimierung herangezogen werden. Die Entscheidung können wir mit **Auto** x264 überlassen.

Damit bleiben die Einstellungen im **More**-Register übrig (Abbildung 34). **Partition decision** steuert die die Genauigkeit der Bewegungssuche innerhalb der Partitionen. Interessant zu wissen ist, dass unabhängig von der Einstellung der letzte Suchdurchlauf immer mit einem Viertel Pixel Genauigkeit geschieht. D. h. in altgewohnter ASP-Terminologie, QPel ist immer aktiv. Da uns ja die Qualität am Herzen liegt, sollten wir **5 (Max Quality)** unverändert lassen oder höchstens auf **4** senken. Besser noch ist **6b (RDO on B-frames)**, da dann die *Rate Distortion Optimization* für B-Frames aktiv wird, die neben der genaueren Bewegungssuche der Qualität spürbar gut tut. Nachteil ist die deutlich geringere Geschwindigkeit.

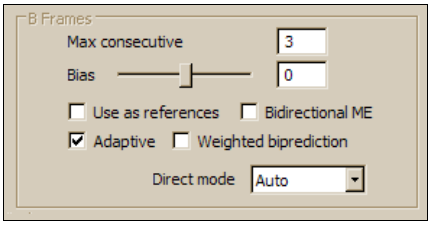

Abbildung 33: x264 B-Frame-Einstellungen.

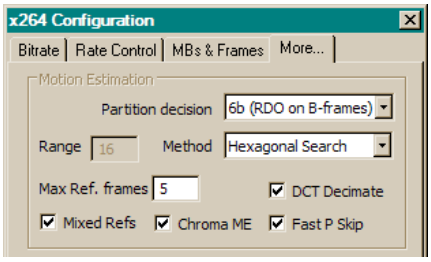

Abbildung 34: Einstellungen für die Bewegungssuche in x264.

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

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

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

Codierzeit, nicht mehr die Qualität. Die Einstellung wirkt sich übrigens auch auf B-Frames aus.

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

DCT Decimate und **Fast P Skip** sollten beide wie in Abbildung 34 voreingestellt aktiviert bleiben.

Wenden wir uns dem unteren Bereich des Fensters zu (Abbildung 35). **Sample AR** ist für anamorphe Zieldateien interessant. Hier tragen wir den passenden PAR aus Tabelle 4 (Seite 44) ein. Für klassische Encodings mit quadratischen Pixeln bleibt die Standardeinstellung **1:1** unverändert.

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

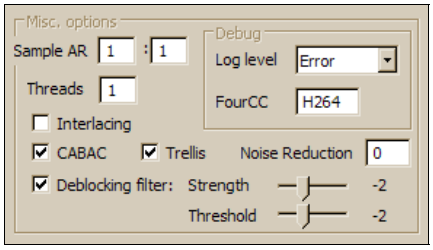

Abbildung 35: Weitere x264-Optionen.

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

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

Trellis kennen wir von Xvid, und genau wie dort bringt die Option ein wenig Qualität ohne zu stark auf die Geschwindigkeit zu drücken. Deshalb sollten wir den Haken setzen. Das gilt nicht für ein 1-Pass-Encoding. Dafür ist Trellis nicht geeignet.

Der **Deblocking filter** war schon Thema im allgemeinen x264-Kapitel auf Seite 92. Hohe Datenraten sollten vorsichtig gefiltert werden (etwa **-2**), niedrige vertragen etwas mehr (etwa **2**).

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

Einstellungen für den Nth Pass

Wir stellen im **Bitrate**-Register um auf **Multipass - Nth pass** und aktivieren **Update stats file**. x264 unterstützt genau wie DivX mehr als zwei Durchgänge. Jeder zusätzliche Pass bedeutet einen enormen Mehraufwand an Encodingzeit. Angesichts des geringen Potenzials zur Qualitätssteigerung ist mehr als das übliche 2-Pass-Verfahren Unsinn, solange nicht wirklich gewichtige Gründe dafür sprechen.

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

A.4.3.2 x264 rev614 Kommandozeilen-Konfiguration

Maßgeblich für die Kommandozeilen ist x264 Revision 614, und zwar das Build von Sharktooth, das wir im Doom9.org-Thread *x264 win32 daily builds* finden. Die unten benutzte Syntax ist zwar sehr stabil, trotzdem können sich in anderen – vor allem neueren und uralten – Versionen einzelne Abweichungen ergeben. Außerdem benötigen wir für unsere Zwecke ein Build, das mindestens AviSynth und evtl. auch MP4 unterstützt. Beides muss beim Kompilieren ausdrücklich eingebaut werden. Bei Sharktooths Builds können wir darauf vertrauen, dass das alles der Fall ist.

Grundsätzliche Syntax

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.

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 müssen, 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.

2nd und Nth Pass

```
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 meistens 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 50 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.

Beispiel einer Konfiguration

Um ein wenig mehr Gefühl für x264 zu bekommen, betrachten wir ein Beispiel, das die Konfiguration aus den Screenshots des x264-VfW-Kapitels (Seite 94) nachbildet. Dazu stellen wir uns die gleiche fiktive 16:9-PAL-DVD wie im Xvid-Kapitel vor, die 100 Minuten Spielzeit aufweist und mit voller anamorpher Auflösung incl. zwei AC3-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.

1st Pass

```
x264.exe --partitions p8x8,b8x8 --subme 2 --ref 2 --bframes 2
--direct auto --trellis 1 --deblock -2:-2 --pass 1 --bitrate 2362
--progress --no-ssim --no-psnr --stats "D:\x264.stats" --output NUL
"D:\Quelle.avs"
```

2nd Pass

```
x264.exe --8x8dct --partitions all --subme 6 --ref 5 --mixed-refs
--bframes 2 --direct auto --b-rdo --trellis 1 --deblock -2:-2
--sar 16:11 --pass 2 --bitrate 2362 --progress --no-ssim --no-psnr
--stats "D:\x264.stats" --output "D:\Zielvideo.mkv" "D:\Quelle.avs"
```

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 eine Einstellung für einen 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.

Meine Version des *Fast 1st Pass* ist weniger aggressiv als MeGUIs Ansatz und dürfte eher mit dem Vorgehen des VfW-Interfaces vergleichbar sein. Folgende Regeln gelten:

- `--ref` halbieren und wenn nötig abrunden.
- `--partitions p8x8,b8x8`, falls mehr erlaubt war.
- `--8x8dct` deaktiviert.
- `--me hex`, falls sie höher war.
- `--subme 2`, falls sie höher war.
- `--b-pyramid` deaktiviert.
- `--weightb` deaktiviert.
- `--b-rdo` deaktiviert.
- `--mixed-refs` deaktiviert.
- `--bime` deaktiviert.
- 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 konfigu-

riert, bleiben die Einbußen so klein, dass sie die Qualität nicht sichtbar verschlechtern. Dafür dürfen wir uns über eine deutlich kürzere Encodingzeit freuen.

Für ein typisches Backup v. a. mit höheren Zielgrößen würde ich zum schnellen 1st Pass raten. Nur wer sich überhaupt nicht für die Geschwindigkeit interessiert und wirklich absolut kompromisslos das allerletzte bisschen Qualität herauskitzeln will, sollte beim vollen 1st Pass bleiben.

Der Turbo-Modus ist außerdem nur im 1st Pass sinnvoll. Alle weiteren Durchgänge sollten immer 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*.

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`). Wegen der Übersichtlichkeit habe ich darauf verzichtet, die Kurzformen aufzulisten.

Frametypen

`--keyint`

Standardwert: `250`

Beispiel: `--keyint 300`

Maximales IDR-Frame-Intervall. Siehe auch Seite 94. Als Daumenregel hat sich das Zehnfache der Bildrate eingebürgert. Deshalb können wir den Standardwert meistens problemlos übernehmen.

`--min-keyint`

Standardwert: `25`

Beispiel: `--min-keyint 30`

Mindestabstand, der zwischen zwei IDR-Frames liegen muss. Siehe auch Seite 94. 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.

`--deblock <alpha:beta>`

Werte für `alpha` und `beta`: Ganze Zahlen zwischen `-6` und `6`.

Standardwert für beide: `0`

Beispiel: `--deblock -2:-2`

Konfiguriert die Stärke des Inloop-Deblocking-Filter, mit dem wir uns schon ausführlicher im allgemeinen x264-Kapitel auf Seite 92 beschäftigt haben. Hohe Datenraten sollten vorsichtig gefiltert werden (etwa `-2:-2`), niedrige vertragen etwas mehr (etwa `2:2`).

`--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 90 f. Ein Wert von `0` deaktiviert B-Frames vollständig, was nicht empfehlenswert ist. Sehr niedrige Zielgrößen vertragen eine große Anzahl an B-Frames (etwa `5`), für den mittleren 2-CD-Bereich bieten sich eher `3` an und für hohe Zielgrößen dürften meistens auch `2` genügen.

`--no-b-adapt`
Deaktiviert die »intelligente« Verteilung von B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 90. Dieser Schalter sollte nicht gesetzt werden.

`--b-bias <Empfindlichkeit>`
Werte: Ganze Zahlen von `-100` bis `100`. Standardwert: `0`
Beispiel: `--b-bias 10`
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 90. Benötigt in der Regel keine Anpassung.

`--b-pyramid`
Erlaubt es, B-Frames als Referenzframes zu verwenden. Siehe auch das allgemeine x264-Kapitel ab Seite 90. Kann v. a. bei höheren Zielgrößen relativ problemlos deaktiviert bleiben, um die Geschwindigkeit zu erhöhen.

`--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 90. Die Entscheidung können wir mit `auto` x264 überlassen.

`--weightb`
Aktiviert die gewichtete Bewegungssuche für B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 90. Kann v. a. bei höheren Zielgrößen relativ problemlos deaktiviert bleiben, um die Geschwindigkeit zu erhöhen.

`--bime`
Erweitert für B-Frames die Suche nach Referenzen. Siehe auch das allgemeine x264-Kapitel ab Seite 90. Kann v. a. bei höheren Zielgrößen relativ problemlos deaktiviert bleiben, um die Geschwindigkeit zu erhöhen.

`--b-rdo`

Aktiviert *Rate Distortion Optimization* für B-Frames. Siehe auch das allgemeine x264-Kapitel ab Seite 90. Trotz deutlicher Geschwindigkeitseinbußen sollte die Funktion aktiviert werden. Benötigt zusätzlich mindestens `--subme 6`.

Encoding-Modus

`--qp <Quantizer>`

Werte: Ganze Zahlen zwischen `0` und `51`. Standardwert: `26`
Beispiel: `--qp 18`
Führt ein 1-Pass-Encoding mit konstantem Quantizer durch. Quant `18` entspricht in etwa dem von Xvid und DivX bekannten Quant 2. Dieser Modus ist eine Alternative zum 2-Pass-Encoding, wenn wir auf hohe Qualität setzen und keine exakte Zielgröße anstreben.

`--crf <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 zwingend mit dem Punkt als Dezimaltrenner angegeben werden. Bei diesem Modus handelt es sich um ein bessere Alternative zu `–qp`. Er ist interessant, wenn wir auf hohe Qualität setzen und keine exakte Zielgröße anstreben.

`--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 Nth-Pass-Encodings haben wir uns ab Seite 98 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 Nth-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 89. 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 sinnvoll.

`--merange <Radius>`

Werte: Ganze Zahlen zwischen `4` und `1024`. Standardwert: `16`

Beispiel: `--merange 12`

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`. Ansonsten ist der Standardwert `5` sinnvoll.

`--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-Tranformation aus dem AVC High Profile. Ist nötig, um bei `--partitions` den Wert `i8x8` zu verwenden und sollte auch nur in Verbindung mit dieser Partitionierung gesetzt 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. Für ein 1-Pass-Encoding ist Trellis nicht geeignet.

`--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-strength <Stärke>`

Werte: `0.0` (deaktiviert, Standard) bis `1.1`
Beispiel: `--aq-strength 0.4`
Wählt die Stärke der Adaptive Quantization. In sehr hellen und sehr dunklen Bereichen des Bilds nimmt das menschliche Auge weniger Details wahr. Das macht sich Adaptive Quantization zu nutze und komprimiert solche Bereiche stärker, um die eingesparte Datenrate an Stellen zu verwenden, die es nötiger haben. Bei geringen Bitraten macht sich das positiv bemerkbar, und dann empfiehlt es sich, AQ mit einem Wert um `0.4` zu aktivieren.
Diese Funktion ist nicht Bestandteil des offiziellen x264-SVN. Deshalb kann sie in manchen Builds nicht verfügbar sein.

`--aq-sensitivity <Schwelle>`

Werte: `5.0` bis `22.0`. Standardwert: `15.0`
Beispiel: `--aq-sensitivity 13.0`
Legt den Schwellenwert fest, wie wenig Details ein Makroblock enthalten muss, damit AQ auf ihn angewendet wird. `5.0` aktiviert AQ für nahezu alle Blocks, `22.0` berücksichtigt nur vollständig einfarbige Blocks. Wenn wir keinen wirklich guten Grund haben, den Schwellenwert anzupassen, sollten wir den Standard beibehalten.

`--cqm <Matrix>`

Werte: `jvt`, `flat` (Standard)

Beispiel: `--cqm jvt`

Wählt eine der beiden Standard-Quantisierungsmatrizen. Siehe auch das allgemeine x264-Kapitel ab Seite 93.

`--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 93.

Dateinamen und Videoeigenschaften

`--output "<Zieldatei>"`

Beispiel: `--output "D:\Zielvideo.mp4" "D:\Quelle.avs"`

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. Meistens steht die Option ganz am Ende der Kommandozeile direkt gefolgt (mit trennendem Leerzeichen) vom Namen der Quelldatei. So ist es auch im Beispiel zu sehen.

`--stats "<Statistikdatei>"`

Beispiel: `--stats "D:\Video\x264.stats"`

Legt Pfad und Dateinamen der für ein Nth-Pass-Encoding nötigen Statistikdatei fest. Standardmäßig wird eine Datei *x264_2pass.log* im aktuellen Ordner erzeugt.

`--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 50). Für klassische Encodings mit quadratischen Pixeln geben wir immer `--sar 1:1` an oder lassen die Option ganz weg.

`--seek <Frame>`
Beispiel: `--seek 1000`
Legt die Framenummer des ersten zu encodierenden Bildes fest. Das erste Frame des Videos hat die Nummer 0.

`--frames <Anzahl>`
Beispiel: `--frames 150000`
Legt die Anzahl der zu encodierenden Frames fest. In Verbindung mit `--seek` könnten wir so nur einen Abschnitt irgendwo aus der Mitte des Videos encodieren. Damit entsprechen die beiden Optionen AviSynths `trim()`.

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 nützlich ist.

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

`--threads <Anzahl>`
Werte: Ganze Zahlen von `1` bis `16`, `auto`
Beispiel: `--threads 2`
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 SMP-Systeme interessant.

A.4.3.3 Zuordnung von x264s VfW- und CLI-Optionen

Wer von x264 VfW auf x264 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 Tabelle 13, deren fette Überschriften für die entsprechenden Tabs der x264 VfW-GUI stehen. Nähere Erklärungen oder Empfehlungen zu den einzelnen Optionen finden wir hier nicht, denn dafür ist der Rest der x264-Konfigurationskapitels ab Seite 89 zuständig.

Bitrate

Single Pass – Bitrate	`--bitrate`
Single Pass – Quantizer	`--qp`
Multipass – 1st Pass	`--pass 1`
Multipass – Nth Pass	Statsfile-Update: `--pass 3` Kein Update: `--pass 2`
Statsfile name	`--stats`

Rate Control

Keyframe boost	`--ipratio`
B-Frames reduction	`--pbratio`
Bitrate variability	`--qcomp`
Min QP	`--qpmin`
Max QP	`--qpmax`
Max QP Step	`--qpstep`
Scene Cut Threshold	`--scenecut`
Min IDR-frame interval	`--min-keyint`
Max IDR-frame interval	`--keyint`

MBs & Frames

Partitions	`--partitions` und `--8x8dct`
Max B-Frames	`--bframes`

B-Frame Bias	`--b-bias`
Use as reference	`--b-pyramid`
Bidirectional ME	`--bime`
Adaptive B-Frames	`--no-b-adapt`
Weighted biprediction	`--weightb`
Direct mode	`--direct`
More	
Partition decision	`--subme`
Range	`--merange`
Method	`--me`
Max ref. frames	`--ref`
DTC Decimate	`--no-dct-decimate`
Mixed Refs	`--mixed-refs`
Chroma ME	`--no-chroma-me`
Fast P Skip	`--no-fast-pskip`
Sample AR	`--sar`
Threads	`--threads`
Interlacing	`--interlaced`
CABAC	`--no-cabac`
Trellis	`--trellis`
Deblocking filter	`--deblock` bzw. `--no-deblock`
Noise Reduction	`--nr`

Tabelle 13: Zuordnung der Optionen von x264 VfW und x264 CLI.

A.4.4 DivX 6.5 VfW-Konfiguration

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

Der DivX-Codec ist für seine einfache Handhabung bekannt und hat durch das Zertifizierungsprogramm 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:

- 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.

Einstellungen für den 1st Pass

Wir öffnen die DivX-Konfiguration und befinden uns im Register **Main**. Um volle Kontrolle über alle Optionen zu erhalten, müssen wir unter **Certification Profile** auf **Unconstrained** umschalten (Abbildung 36) 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.

Weiter unten bei **Rate Control Mode** (Abbildung 37) wählen wir den passenden Modus für der ersten Encodingdurchgang. DivX bietet zwei Möglichkeiten, nämlich **Multipass, 1st pass**, der dem bisherigen Verhalten entspricht, und **Multipass, 1st pass (fast)**, der in DivX 6.4 neu eingeführt wurde. *Fast* arbeitet nach dem selben Prinzip wie Xvids 1st Pass oder MeGUIs Turbo-Modus für x264. Dabei werden einige Features deaktiviert, die im ersten Encodingdurchgang nicht gebraucht werden und ohne Auswirkungen auf die Qualität abgeschaltet werden können. Dadurch läuft der 1st Pass deutlich schneller. Nur wer zu viel Zeit hat, sollte deshalb nicht **Multipass, 1st pass (fast)** wählen.

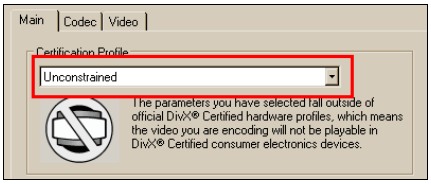

Abbildung 36: DivX-Profilauswahl.

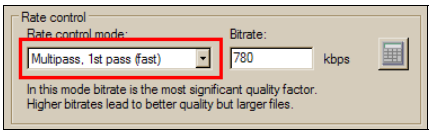

Abbildung 37: Einstellung für 1st Pass im DivX-Hauptdialog.

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

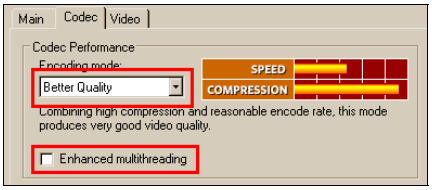

Abbildung 38: DivX-Performance-Optionen.

Damit können wir ins Register Codec weiterklicken (Abbildung 38). Bei **Codec Performance** müssen wir uns entscheiden, ob Geschwindigkeit oder Qualität wichtiger ist. **Balanced** ist vergleichsweise flott und bringt durchschnittliche Qualität. **Better**, **Extreme** und **Insane** bremsen jeweils kräftig ab, belohnen uns aber auch mit höherer Qualität. Der sinnvollste Kompromiss zwischen Geschwindigkeit und Qualität dürfte **Better** sein. **Extreme** und **Insane** haben ihre Namen aus gutem Grund.

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

Bidirectional coding in Abbildung 39 konfiguriert die B-Frames. **Off** dürfte sich selbst erklären ;-), **Adaptive Single Consecutive** entspricht dem verhalten älterer DivX-Versionen, d. h. mehrere B-Frames hintereinander werden nicht gesetzt. Ma-

ximal drei aufeinander folgende B-Frames erlaubt **Adaptive Multiple Consecutive**. B-Frames sollten wir auf jeden Fall aktivieren, und das dürfen ruhig die Multiple Consecutive sein.

Quarter-pixel search (QPel) und **Global motion compensation** (GMC) haben wir schon bei der Xvid-Konfiguration auf Seite 70 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. Unter **Output pixel aspect ratio** (Abbildung 40) erlaubt DivX seit Version 6.5 die Angabe eines PAR, der im AR-Flag des MPEG-4-Streams gespeichert wird. Das ist nur für anamorphe Zieldateien interessant. Ein klassisches Encoding mit quadratischen Pixeln verwendet immer die Standardeinstellung **SQUARE 1:1**. Die nicht-quadratischen Einträge entsprechen den Werten gemäß ITU-R BT.601 (vgl. Tabelle Seite 44).

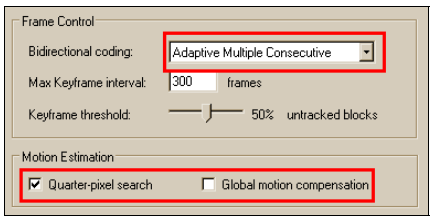

Abbildung 39: DivX-Optionen für Frametypen und Bewegungssuche.

Abbildung 40: PAR-Auswahl.

Bei **Image Processing** interessiert uns nur die rechte Seite. Die anderen Einstellungen sollten auf den Standards wie im Bild belassen werden. Unter **Quantization** (Abbildung 41) lässt sich angeben, welche Quantisierungsmatrix DivX verwenden soll. Eine Spezialität von DivX ist **H.263 Optimized**, die auf einem modifizierten H.263-Algorithmus basiert, der zwar Geschwindigkeit kostet, aber höhere Qualität bringt. **MPEG-2** war in älteren DivX-Versionen problematisch und nicht empfehlenswert. Inzwischen gehören diese Probleme der Vergangenheit an.

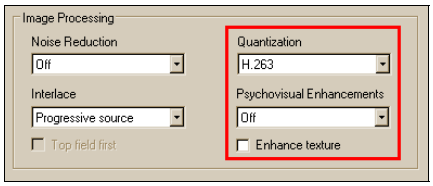

Abbildung 41: DivXs Matrix- und PVE-Auswahl.

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

Psychovisual Enhancements (PVE) versucht, die Bildteile zu errechnen, die das menschliche Auge sowieso nicht wahrnehmen kann, und lässt diese weg. Die

frei werdende Bitrate kann dann für andere Teile des Bilds verwendet werden. Für klassische Cartoons und manche Animes empfiehlt sich **Masking**. Für alles andere Material ist **Shaping** die bessere Wahl. Wer die niedrigere Geschwindigkeit verkraften kann, sollte PVE bei niedrigen Bitraten in der Regel aktivieren. **Enhance texture** ist ein Feature, das im Rahmen der PVE die Qualität rund um I-Frames verbessern soll.

Damit ist die Konfiguration des 1st Pass fast beendet. Nur ein Klick unten links auf **Advanced** bleibt uns noch (Abbildung 42). Hier können wir unter **Temporary Files Folder** angeben, wohin DivX seine temporären Dateien (z. B. die im 1st Pass angelegte Logdatei) ablegt. Der Ordner hat nichts damit zu tun, wohin der Film gespeichert wird. Weiter unten setzen wir die Haken bei **Do not prompt...** und **Disable the feedback window**. So überschreibt DivX im 2nd Pass ungefragt die Log-Datei (was er ja tun soll) und schaltet das Statistik-Fenster während des Encodings ab. **Rotate artwork** und **Check for new version** kann jeder ganz nach seinen Vorlieben setzen. Naja … das Statistik-Fenster eigentlich auch. :-)

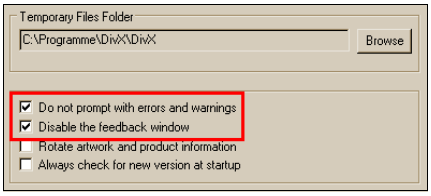

Abbildung 42: DivX, weitere Optionen.

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

Einstellungen für den Nth Pass

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

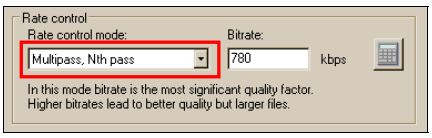

Abbildung 43: Einstellung für Nth Pass im DivX-Hauptdialog.

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

Teil B
Praxiswissen

Einleitung

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

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

B.1 Vorarbeiten

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

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

B.1.1 Die nötige Software

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

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 148 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, RealAnime, AutoMKV

Vorarbeiten

vStrip

DVD-Ripper. Kopiert die DVD auf die Festplatte. Alternativen sind der nach wie vor beliebte DVD Decrypter oder DVDFab Decrypter. Diese Programme sind komfortabler und in mancher Hinsicht besser als vStrip, können aber auch dazu dienen, den CSS-Kopierschutz zu umgehen. Deshalb verzichte ich wegen der Abmahngefahr durch geldgeile Anwälte lieber auf Links und Erklärungen zu diesen Rippern.
http://www.free-codecs.com/download/vStrip.htm

Chapter-X-tractor

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

DGMPGDec

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

Audio und Untertitel

BeSweet

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

BeLight

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

VobSub

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

SubRip

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

Video-Encoding

AviSynth

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

AviSynth-Plugins

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

Xvid Video-Codec

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

Xvid_Encraw

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

DivX Video-Codec

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

x264 Video-Encoder

Sharktooths Builds von x264. Alternativ können wir die Builds von z. B. Bob0r auf *x264.nl* verwenden; das Encodingwissen bezieht sich jedoch immer auf die Sharktooth-Version. Achtung: x264 VfW ist nur auf *http://deathtbesheep.uni.cc* verfügbar. Den Quellcode gibt es auf *videolan.org*.
http://forum.doom9.org/showthread.php?threadid=89979

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

MP4Box

Muxing-Tool für MP4 und Teil des GPAC-Projekts. Wer sich mit MP4Box beschäftigen will, unbedingt diesen *Doom9-Thread* lesen: »GPAC's MP4Box Information, Requests and Discussions«.
http://ffdshow.faireal.net/mirror/gpac/dev

B.1.2 DVD-Ripping

Recht und Gesetz

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

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

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

Wichtige Grundlagen

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

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

Die Dateien sind immer nach dem selben Schema benannt, und zwar **vts_*xx*_*y*.*zzz***. Dabei steht **xx** für eine zweistellige Nummer, die eine Gruppe von VOBs identifiziert. So haben zum Beispiel alle zum Hauptfilm gehörigen VOBs das gleiche **xx**. Innerhalb einer Dateigruppe wird das **y** von **0** aus hochgezählt, und **zzz** bezeichnet die Dateiendung (VOB oder IFO). So könnte z. B. der Hauptfilm in drei VOBs verpackt sein, die alle die Nummer **05** haben, das wären dann

vts_05_0.vob, **vts_05_1.vob** und **vts_05_2.vob**. Dazu gehört die passende **vts_05_0.ifo**. Mehr brauchen wir nicht wissen, um immer die richtigen Dateien auszuwählen.

B.1.2.1 Ripping des Films

Wir legen die DVD ein und starten vStrip GUI. Sollten wir uns dann nicht schon im Register **Input** befinden, wechseln wir dorthin und klicken auf den Button **Add**, um die VOB-Dateien von der DVD ▸ A.2.2 auszuwählen. Da wir den kompletten Hauptfilm rippen wollen, geht das recht einfach.

Im **Öffnen**-Dialog (Abbildung 44) wechseln wir zum DVD-Laufwerk in den Ordner **VIDEO_TS**. Zur besseren Übersicht empfiehlt es sich, rechts oben in die Detailansicht umzuschalten. Dann suchen wir uns die längste zusammenhängende Sequenz VOB-Dateien heraus, bei denen also die mittlere Nummer gleich ist. Im Bild oben sind das alle VOBs mit der **02** in der Mitte, also **vts_02_0.vob** bis **vts_02_4.vob**. Allerdings enthält die Datei mit der **0** noch keine Daten des Films, weshalb wir nur die VOBs ab **1** auswählen und **Öffnen** klicken. Im oberen Fenster von vStrip erscheinen jetzt die ausgewählten Dateien, und wir können ins **IFO**-Register wechseln (Abbildung 45).

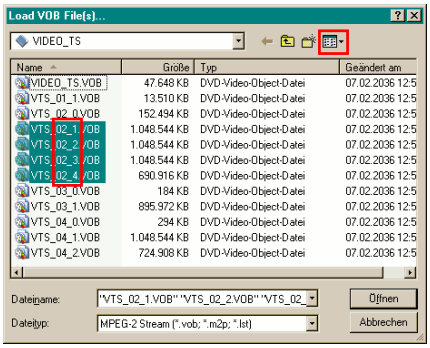

Abbildung 44: VOB-Auswahl mit vStrip.

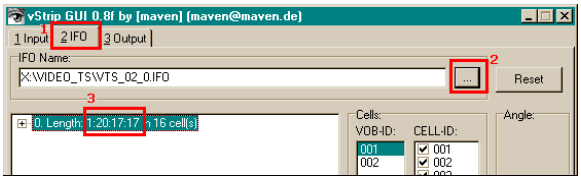

Abbildung 45: IFO-Datei in vStrip laden.

Abbildung 46: vStrip Stream-Liste.

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

Ein Klick auf **Öffnen** bringt uns zurück zu vStrip. Im großen Fenster stehen nun alle PGCs, die die gewählte IFO enthält. Im Idealfall ist das sowieso nur eine

einzige. Ansonsten wählen wir diejenige aus, deren Länge (3) mit der Länge des Hauptfilms übereinstimmt.

Unten im Fenster (Abbildung 46) zeigt uns vStrip, welche Audio- und Untertitelstreams vorhanden sind. Wenn wir einige davon weglassen, also gar nicht auf die Festplatte kopieren, wollen, müssen wir uns die IDs der unnötigen Streams merken. Relevant ist jeweils die zweite **0x**-Nummer, für die Audiospuren im Bild also **0x80** bzw. **0x81**. Damit geht es weiter ins **Output**-Register (Abbildung 47).

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

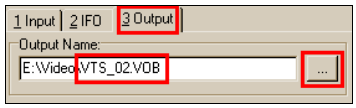

Abbildung 47: vStrip: Name der Zieldateien.

Die **Output Options** in der Mitte sollten immer ohne Haken bleiben, und weiter unten bei **Split File Size** (Abbildung 48) bestimmen wir, wie groß eine einzelne VOB werden darf. Das Maximum von 1024 MB wie auf der DVD einzuhalten, ist nicht so wesentlich. Nur über 4096 MB sollten wir nicht gehen, da manche Tools dann Probleme machen könnten. Außerdem kann das FAT32-Dateisystem keine größeren Dateien verwalten. Wer NTFS benutzt, hat diese Einschränkung nicht. **Split** sollte immer auf **Never** gesetzt bleiben, es sei denn vStrip kommt dann beim Ripping ins Straucheln.

Abbildung 48: Maximale VOB-Größe in vStrip.

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

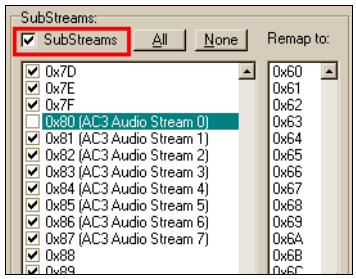

Abbildung 49: Abwählen von Spuren in vStrip.

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

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

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

B.1.2.2 Ripping der Kapitelliste

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

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

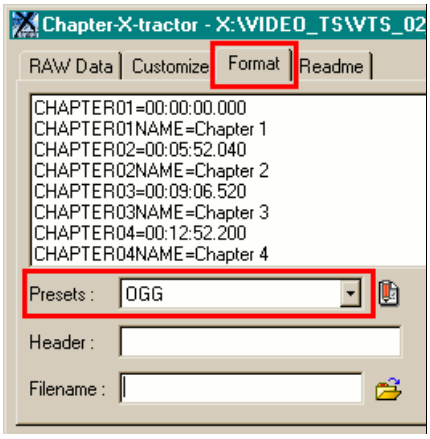

Abbildung 50: Formatwahl in Chapter-X-tractor.

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

Ein Klick unten links auf **Save data** speichert die Kapitelinfos in einer Textdatei ab, und wir sind fertig. Wer will kann vorher noch in den jeweiligen **CHAPTER-**

xxNAME-Zeilen die Standardbezeichnungen gegen die echten Kapitelnamen austauschen. Natürlich lässt sich die erzeugte Datei auch später mit einem beliebigen Texteditor anpassen.

B.1.3 Indexieren der VOBs mit DGIndex

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

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

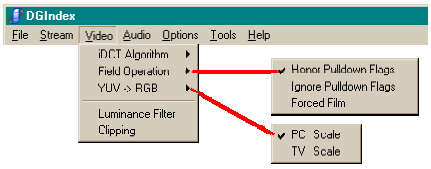

Abbildung 51: Einstellungen im DGIndex-Video-Menü.

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.

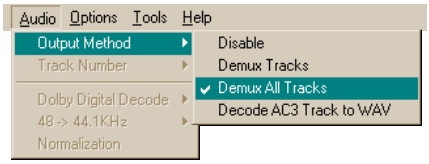

Abbildung 52: Demuxing sämtlicher Audiospuren.

Die Einstellungen im **Video**-Menü sollten wie in Abbildung 51 aussehen. Den **iDCT Algorithm** wählt DGIndex automatisch. Im **Audio**-Menü regeln wir das Herausziehen der Audiospuren aus den VOBs (Abbildung 52). **Output Method** bestimmt, auf welche Weise die Audiospuren extrahiert werden. **Disable** extrahiert gar nichts, **Demux Tracks** extrahiert nur die unter **Track Number** ausgewählten Spuren und **Demux all Tracks** extrahiert alle vorhandenen Spuren. Mit **Decode AC3 Track to Wav** könnten wir den Sound gleich in Wave umwandeln. Das sollten wir aber besser BeSweet überlassen. Deshalb interessiert uns auch der Rest des Audio-Menüs nicht.

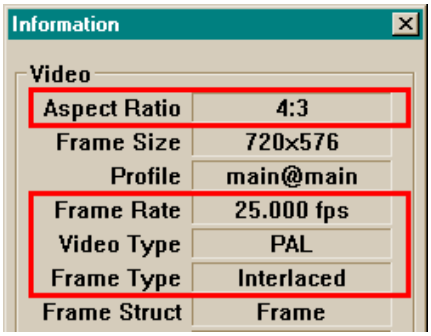

Abbildung 53: DGIndex-Statusfenster.

Über **F4** legen wir nun eine D2V-Projektdatei an. Solange DGIndex arbeitet,

öffnet sich das Statusfenster (Abbildung 53). Sobald dort ganz unten **FINISH** erscheint, ist der Speichervorgang abgeschlossen und wir haben die extrahierten Audiodateien und die D2V-Datei auf der Platte. Die ersten drei rot umrandeten Werte merken wir uns. Die werden später in Gordian Knot wichtig, um Auflösung und Seitenverhältnis des Bildes richtig einzustellen.

Wenn bei **Frame Type** zum Schluss **Interlaced** erscheint, ist das Video möglicherweise interlaced codiert. Dann sollten wir uns Szenen mit viel horizontaler Bewegung suchen und prüfen, ob dort Kammartefakte wie im Screenshot 54 auf Seite 128 auftreten. Manchmal wird der Effekt auch nur bei Szenenwechseln sichtbar, wenn sich altes und neues Bild ein Frame lang überlagern.

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

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

Abbildung 54: Bild mit Interlacing-Artefakten. Besonders deutlich um die Kinnpartie des Manns links im Bild.

B.2 Audio-Transcoding

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

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

B.2.1 BeSweet einrichten

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

Komponenten

Um BeSweet vollständig zu installieren, benötigen wir folgende Bestandteile:

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

VOBInput.dll

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

Lame_enc.dll

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

Libvorbis.dll

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

2Lame

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

Nero

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

Winamp

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

BeLight

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

Installation

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

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

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

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

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

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

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

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

B.2.2 Audio decodieren und bearbeiten

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

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

Nun betrachten wir die linke Seite des BeLight-Fensters. Der erste Eintrag (Abbildung 55) gehört zu Azid. Azid ist dafür zuständig, die AC3-Tonspur ▸ A.2.1 zu decodieren und eventuell 6-Kanal auf 2-Kanal (Stereo) umzurechnen.

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

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

Beim Downmix auf Stereo würden wir ohne Kompression eine nervig leise

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

Abbildung 55: BeLight-Option für Dynamikkompression.

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

Wer die Dynamikkompression manuell konfigurieren möchte, nimmt **Boost** (Abbildung 56) anstatt Azids **Dynamic Compression**. Achtung! Nicht beides zusammen verwenden! **Boost** hat es in sich. Wir können damit das letzte bisschen Dynamik aus der Tonspur herauskomprimieren, was in meinen Ohren schon nicht besonders gut klingt. Dazu kommt die Gefahr, durch zu viel Kompression Störgeräusche im Sound zu erzeugen. Deshalb sollten wir uns gut überlegen, ob der Einsatz von Boost wirklich nötig ist. Der Weg über Azids **Dynamic Compression** ist meistens der bessere. Ich habe **Boost** noch nie verwendet und das bis jetzt auch noch nicht bereut.

Abbildung 56: Boost als Ersatz für Dynamic Compression.

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

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

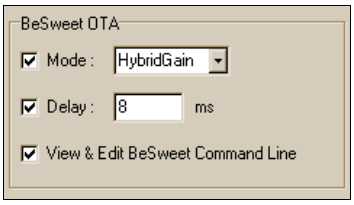

Abbildung 57: Normalisierung und Delay.

Damit weiter in den **OTA**-Abschnitt (Abbildung 57). Wir müssen den Sound noch normalisieren, d. h. die Lautstärke auf 100 % oder knapp darunter anheben. Durch die Dynamikkompression von oben haben wir zwar schon die Lautstärken-Unterschiede innerhalb der Tonspur angeglichen, insgesamt ist sie aber immer noch viel zu leise. Das Anheben erledigt **Mode** im **OTA**-Abschnitt.

Haben wir uns für AAC ▸ A.2.1 als Zielformat entschieden, heißt die richtige Einstellung **PreGain**, für alle anderen Formate **HybridGain**. Der Unterschied kommt

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

Damit kommen wir zum **Delay**. Das ist der Wert, um den die Audiospur zum Video verschoben sein muss, um exakt synchron zu werden. Diese Angabe steht im Dateinamen, »DELAY 8ms« im Beispiel von Abbildung 58. Diesen Wert übernimmt BeLight automatisch. Das funktioniert allerdings nur, wenn im Dateinamen auch eine Angabe im Format »DELAY XXms« vorhanden ist, ansonsten müssen wir den Wert manuell eintragen – einschließlich evtl. vorhandenem negativem Vorzeichen.

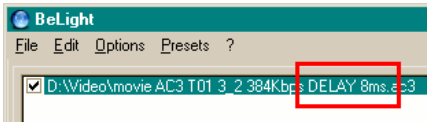

Abbildung 58: Delay-Angabe im Dateinamen.

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

Damit kommen wir zu den **Advanced Settings** (Abbildung 59). Für den Downmix auf Stereo sollten die Optionen in der Regel so aussehen wie oben. **LFE to LR Channels** bestimmt, mit welchem Pegel der Basskanal in die vorderen beiden Kanäle gemixt werden soll. Um ein Zuviel an Bass zu vermeiden, stellen wir hier **-3db** ein. Mit dem **Output Mode** definieren wir, welche Art von Stereo erzeugt wird: **Mono**, normales **Stereo**, Dolby Pro Logic (**DPL**) oder Dolby Pro Logic II (**DPL2**). Pro Logic codiert Surroundinformationen in die beiden Stereokanäle, so dass – einen passenden Decoder vorausgesetzt – beim Abspielen zumindest ein Teil der ursprünglichen Surroundinformationen wieder hergestellt werden kann. Behalten wir die ursprünglichen 6 Kanäle bei, entfernen wir sowohl bei **LFE to LR Channels** als auch bei **Output Mode** die Haken.

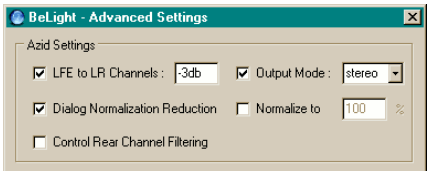

Abbildung 59: Erweiterte Azid-Einstellungen.

Als letztes aktivieren wir die **Dialog Normalization Reduction**. 5.1-AC3-Da-

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

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

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

B.2.3 AAC-Encoding mit Nero und Winamp

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

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

Der Nero-Encoder

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

Im **AAC**-Register (Abbildung 60) stellen wir bei **Encoding Engine** den **Nero-Digital**-Encoder ein und wählen links daneben die passende Anzahl an **Output**

Channels. CBR-Encoding ▸A.1.3 ist wegen der geringeren Qualität wenig empfehlenswert. Deswegen halten wir uns an den **Variable Bitrate Mode**. Für Stereo-Ton liefert uns das **Streaming**-Preset kompakte 75 kbit/s (HE-AAC vorausgesetzt), das ist wenig genug. Für sechs Kanäle können wir circa die HE-Bitraten aus Tabelle 14 erwarten. Dabei klingt **Tape** immer noch akzeptabel. Meistens verwende ich **Internet** oder **Streaming**.

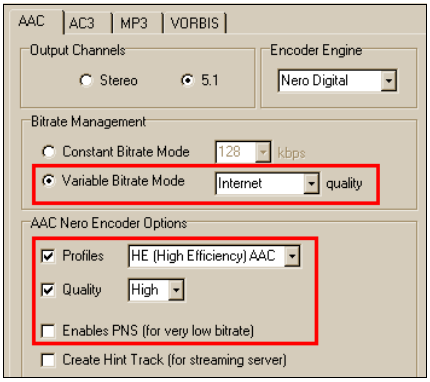

Abbildung 60:
Nero-AAC-Setup in BeLight.

Tape	100
Radio	120
Internet	150
Streaming	190

Tabelle 14: 6-Kanal-Bitraten mit Nero 6.

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

Der Winamp-Encoder

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

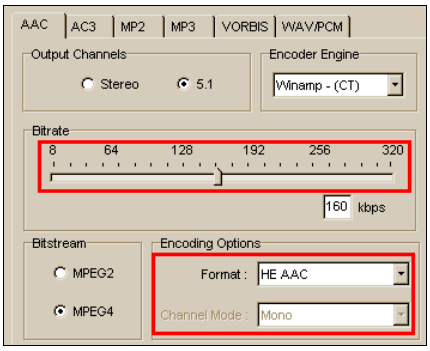

Abbildung 61:
Winamp-AAC-Setup in BeLight.

Die passende Einstellung der **Encoder Engine** im **AAC**-Register ist natürlich **Winamp** (Abbildung 61). Der Schieberegler unter **Bitrate** definiert die verwendete Bitrate, wobei 6-Kanal-AAC mindestens 96 kbit/s und höchstens 213 kbit/s haben kann. Bitraten außerhalb dieses Bereichs lässt BeLight nur für Stereo zu. Mit 160 kbit/s liegt meine Lieblingsbitrate in

einer ähnlichen Region wie für Nero. Für Stereo dürften es deutlich weniger sein. Zwischen **64** und **80** sollte im HE-Modus keine Probleme bereiten.

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

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

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

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

B.2.4 Vorbis-Encoding

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

Zuerst wählen wir wie in Abbildung 62 **Stereo**-Output, und dann bei **Bitrate Management** die Einstellung **Quality**. Vorbis ist auf echtes VBR ▸ A.1.3 im Constant-Quality-Modus ausgelegt und bringt nur da seine volle Leistung. Deswegen sind die **Bitrate**-Modi eher uninteressant. Mit denen könnten wir zwar vor dem Encoding etwas genauer bestimmen, wie groß die encodierte Audiodatei wird. Da wir die endgültigen Größen aber sowieso erst nach dem Audio-Transcoding berechnen, bringt uns das keinen Vorteil. Im Gegenteil würden wir gegenüber dem Quality-Modus Qualität verlieren.

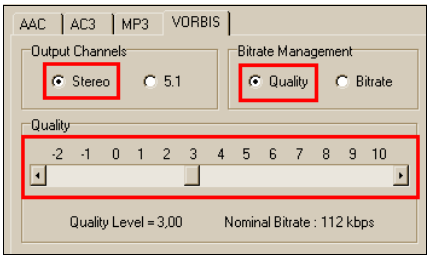

Abbildung 62: Vorbis-Setup in BeLight.

Mit dem Schieberegler stellen wir dann das gewünschte Qualitätslevel ein. **2,00** ergibt etwa 80 kbit/s und ist als sichere Untergrenze gut brauchbar. Für stark komprimierte 1-CD-Encodings kann allerdings auch ein deutlich kleinerer Wert

sinnvoll sein. **5,00** liefert uns etwa 140 kbit/s und taugt gut als Obergrenze.

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

B.2.5 MP3-Encoding mit Lame

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

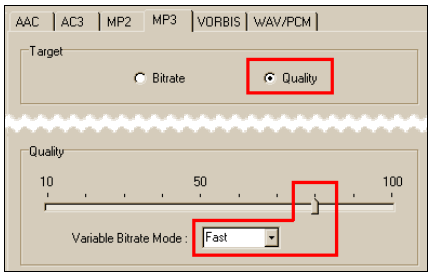

Abbildung 63: Lame MP3-Setup in BeLight.

Da MP3 nur Stereo unterstützt, entfällt die Auswahl der Zielkanäle. Wir brauchen nur wie in Abbildung 63 im **MP3**-Register unter **Target** auf **Quality** zu klicken, um der VBR-Modus einzustellen. Im **Quality**-Abschnitt stellen wir die gewünschte Qualitätsstufe ein. Seit Lame 3.97, den wir verwenden, gehören die weit bekannten Presets offiziell der Vergangenheit an. Zum Vergleich, welche Qualitätsstufe für welches Preset steht und welche Bitraten wir erwarten sollten, dient Tabelle 15.

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

Tabelle 15: Einige Qualitätsstufen von Lame.

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

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

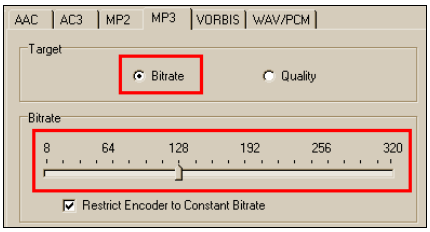

Abbildung 64: Lame MP3-Setup für CBR.

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

CBR-MP3 erzeugen wir mit den Einstellungen aus Abbildung 64. Unter **Target** wählen wir **Bitrate** und klicken im **Bitrate**-Abschnitt **Restrict encoder...** an. Mit dem Schieberegler stellen wir dann die gewünschte Bitrate ein. Werte unter 96 kbit/s sollten wir dabei der Tonqualität zuliebe besser vermeiden. Genauso dürfte für die meisten Tonspuren mehr als 192 kbit/s eher Platzverschwendung als Qualitätssteigerung sein.

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

B.2.6 AC3-Encoding

Dieses Kapitel bereitet mir ehrlich gesagt Zahnschmerzen, und ich habe es nur am Leben gelassen, weil der Bedarf offenbar da ist. Die einzige Fähigkeit des AC3-Encoders ▸ A.2.5 von BeSweet besteht darin, etwas zu produzieren, das einem AC3-Stream ähnelt. DSPguru selbst sagt sehr deutlich: »ac3enc.dll creates (almost) useless ac3 streams. don't use it to encode things you care for. it's there only for test purposes.« Problem: Ac3enc ist der einzige AC3-Encoder, der mit BeSweet zusammenarbeitet. Und das größere Problem: Kommerzielle Encoder, die auch professionelle Qualität erzeugen, kosten eine gewaltige Stange Geld. Deshalb wird ac3enc trotzdem verwendet. Aber nicht meckern, wenn hinterher Mist raus-

kommt!

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

Der Ziel-AC3 eine DVD-übliche Bitrate von 384 oder 448 kbit/s zu gönnen, macht wenig Sinn. Dann hätten wir gleich das Original beibehalten können. Tiefer als die in Abbildung 65 bei **Bitrate** mit dem Schieberegler ausgewählten 256 kbit/s sollten wir für 6-Kanal-Dateien allerdings nicht gehen, um die Einbußen bei der Qualität im Rahmen zu halten. AC3 ist nun mal nicht für niedrige Bitraten ausgelegt, und ac3enc noch viel weniger.

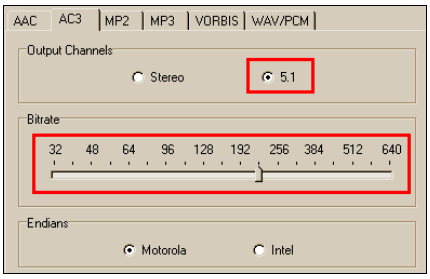

Abbildung 65: AC3-Setup in BeLight.

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

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

B.3 Untertitel

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

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

B.3.1 Ins Bild eingebrannte Untertitel

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

Untertitel rippen

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

Über den **Open**-Button unten links gelangen wir in den Öffnen-Dialog, stellen bei Dateityp auf **Ifo and vobs, for creating idx/sub** um und wählen die vom Ripper erstellte ifo-Datei aus, die sich **vts_01_0.ifo** oder ähnlich nennt. Es folgt Abbildung 66, wo wir den Speicherort der Untertitel angeben. Dann erscheint dieses Fenster. Im Bereich oben rechts stehen die Untertitel, die aus den VOBs gerippt werden sollen. Alles, was wir nicht beim Rippen von der DVD gezogen haben, können wir schon

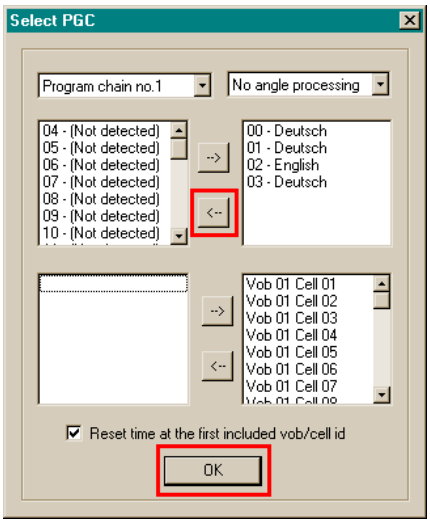

Abbildung 66: Untertitelauswahl in Vobsub.

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

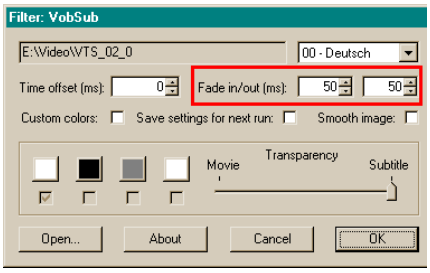

Abbildung 67: Untertitel ein-/ausblenden mit Vobsub.

Normalerweise blendet Vobsub jeden Untertitel kurz ein und wieder aus. Die Standard-Einstellung von 50 ms (Abbildung 67) sieht man kaum, sie lässt das Auftauchen und Verschwinden der Untertitel aber nicht ganz so hart erscheinen wie ohne Fade. Ändern lassen sich die zwei Werte über **Fade in/out**. Danach **Ok** klicken. Nun öffnen wir mit einem Texteditor die IDX-Datei. Sie nennt sich **vts_01_0.idx** oder ähnlich, jedenfalls mit dem **_0** hinten dran. Jetzt wird wichtig, wie Dialog- und Szene-Untertitel gespeichert sind.

Dialog- und Szene-Untertitel in verschiedenen Spuren

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

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

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

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

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

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

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

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

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

B.3.2 Dynamisch einblendbare Untertitel

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

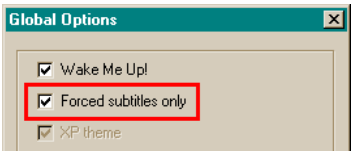

Abbildung 68: Forced-Subs-Option in SubRip.

In diesem Kapitel kümmern wir uns um Text-Untertitel. Das benötigte Programm heißt SubRip. Dort gelangen wir über **Options / Global Options** in den Dialog aus Abbildung 68. Mit einem Haken bei **Forced**

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

Im Hauptfenster rufen wir mit dem **VOB**-Button den Öffnen-Dialog auf (Abbildung 69). Am einfachsten ist es nun, mit **Open IFO** die vom Ripper erstellt IFO-Datei zu öffnen. Alternativ können wir auch mit **Open Dir** die VOBs direkt laden. Vorteil der ersten Methode ist, dass dann im Dropdown-Feld darüber die Untertitelspuren richtig bezeichnet werden. Diese Infos stehen nämlich in der IFO-Datei, nicht in den VOBs. Im Dropdown-Feld stellen wir die gewünschte Sprache ein und widmen uns dann der rechten Seite des Fensters (Abbildung 70). **Last Time Code** muss Null sein. Steht hier ein Wert, addiert ihn SubRip zu jeder Zeitmarke dazu, was zu asynchronen Untertiteln führt.

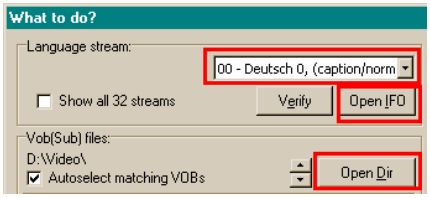

Abbildung 69: DVD-Untertitel laden in SubRip.

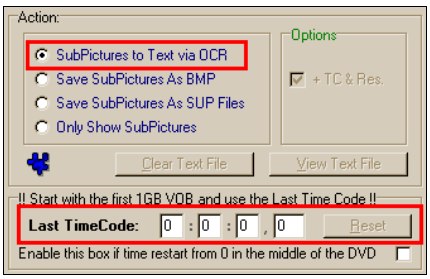

Abbildung 70: OCR-Erkennung und Timecode-Anpassung.

Da wir eine Textdatei erzeugen wollen, die Untertitel auf der DVD aber als Bilder gespeichert sind, muss sie SubRip per OCR-Texterkennung in Text umwandeln. Dabei erkennt das Programm die Buchstaben aber nicht wie die professionellen OCR-Programme selbständig, sondern fragt nach, welches Grafikmuster zu welchem Buchstaben gehört. Das geschieht im Dialog aus Abbildung 71. Über den rot umrandeten Buchstaben (hier das »H«) ist sich SubRip im Unklaren. Also geben wir ihm im Eingabefeld in der Mitte den passenden Buchstaben an. Rechts daneben können wir noch fett, kursiv und unterstrichen als Auszeichnung wählen.

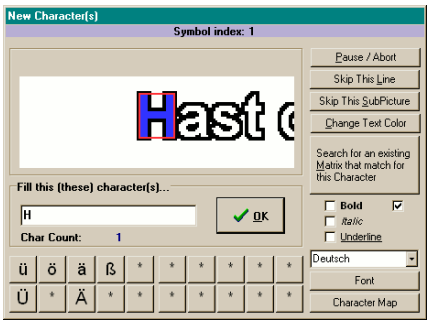

Abbildung 71: OCR-Erkennungsdialog von SubRip.

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

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

Damit zurück zum Editor-Fenster (Abbildung 72), in dem wir den gerade gerippte 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 Frame-basierten Format wie MicroDVD speichern wollten.

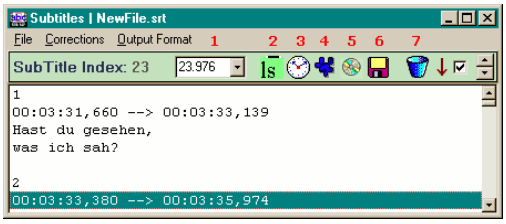

Abbildung 72: SubRips Editorfenster.

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 voreingestallt stehen lassen. Und zwar verwechselt SubRip 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.

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 wir weitere Untertitel erkennen wollen: Nach dem Speichern das Editorfenster über das Papierkorb-Symbol (7) leeren, mit dem **VOB**-Button den **Öffnen**-Dialog aufrufen, die neue Untertitelspur auswählen und auf der rechten Seite des Fensters oben die Character-Matrix-Datei laden und mit **Reset** den Timecode nullen, sonst passen die Zeitmarken nicht mehr.

Ganz zum Schluss bleibt nur noch, die Untertitel-Spur(en) in eine Textverarbeitung zu laden und 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.4 Die Encoding-Frontends

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

Tabelle 16 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. Als Open-Source-Software unter der GPL-Lizenz kann das Programm prinzipiell von jedem verändert werden. 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.

Doch kein Lob ohne Tadel. StaxRip leidet genau wie manches Microsoft-Feature darunter, manchmal zu schlau sein zu wollen. Solange man sich in den alteingefahrenen Bahnen des klassischen DVD-Backup bewegt, macht das wenig Probleme – von den laut StaxRip eigentlich prinzipiell immer inkompatiblen Versionen der externen Tools einmal abgesehen. Wer als erfahrener Benutzer allerdings seine eigenen Vorstellungen verwirklichen will, den werden die Automatiken manchmal sicher zum Fluchen bringen. Entlastend steht dagegen, dass StaxRips Eigenwilligkeit keine ausgefallenen Einstellungen unmöglich macht. Trotzdem würde ich mir einen Expertenmodus wünschen, in dem die ganzen Helferlein und Automatismen deaktiviert sind.

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

Tabelle 16: Funktionsumfang der Encoding-Frontends.

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

StaxRip baut auf Microsofts .NET-Technologie auf und benötigt ein installiertes .NET Framework 2.0, um gestartet werden zu können.

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.

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 / Project Options** oder mit der Taste **F7**. Dort wechseln wir auf die **Automation**-Seite (Abbildung 73). 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.

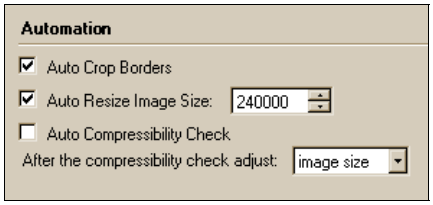

Abbildung 73: StaxRip Automatik-Konfiguration.

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 ▸ A.3.3. Natürlich befreit uns die Automatik nicht davon, selbst abzuwägen, ob die Auflösung tatsächlich für das jeweilige Encoding passt.

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.

Weiter geht's auf der Seite **Advanced** (Abbildung 74). Die meisten Optionen auf dieser Seite können wir bedenkenlos auf den Standardeinstellungen belassen. Interessant ist lediglich der **Advanced-Automation**-Ast, und zwar hauptsächlich dann, wenn wir ein anamorphes Encoding ▸ A.3.2.2 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.

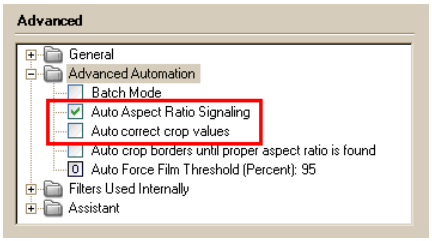

Abbildung 74: Optionen für Cropping und AR-Flag.

Auto Correct Crop Values war bis StaxRip 0.9.9.5 unter **Dangerous / Save Cropping** zu finden. Wenn aktiviert, erzwingt diese Option beim Cropping mod4 (klassisches Encoding) oder mod16 (anamorphes Encoding). Besonders für anamorphes MPEG-4 sollte der Haken nicht gesetzt sein, um auch Nicht-Mod16-Auflösungen zu erlauben. Fürs klassische Backup ist die Option weniger wichtig.

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.

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 **Startup Template** (Abbildung 75) stellen wir die Projektvorlage ein, die beim Start von StaxRip automatisch geladen wird. Fürs DVD-Backup bietet sich natürlich die Vorlage **DVD** an; oder die Vorlage, die wir gerade eben nach dem Setup der Projektoptionen gespeichert haben.

General / Show help... steuert, ob StaxRip für jedes Fenster, das wir zum ers-

ten Mal aufrufen, automatisch zusätzlich die Hilfe einblendet. Ob wir den Haken entfernen oder nicht, ist Geschmackssache. Das gleich gilt für den Punkt **Check the internet for updates**, der im aktivierten Zustand regelmäßig online nachsieht, ob wir noch die neuste Version von StaxRip benutzen. Interessanter ist schon **Resolution Slider**. Dieser Punkt legt fest, in welchem Bereich wir die horizontale Auflösung festlegen können. Die Standardwerte sind für ein DVD-Backup passend. Wer mit sehr hoher oder sehr niedriger Auflösung encodieren will, muss die Grenzwerte anpassen.

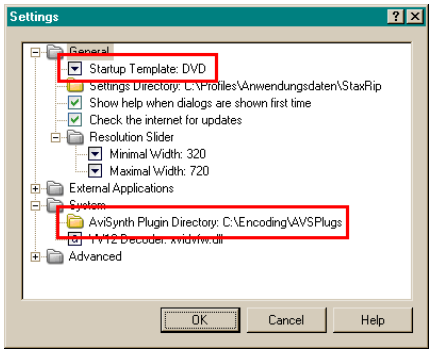

Abbildung 75: Globale Optionen.

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 Programme, 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 im Bild für x264 der Fall. Wir können nun über den **Download**-Button x264 direkt herunterladen oder über den Eintrag im **Tools**-Menü die Programm-Website im Browser öffnen. Haben wir x264 schon längst auf der Platte, klicken wir auf **Search**, im folgenden Dialog auf **Full** und suchen den Ordner heraus, in dem sich die x264.exe befindet.

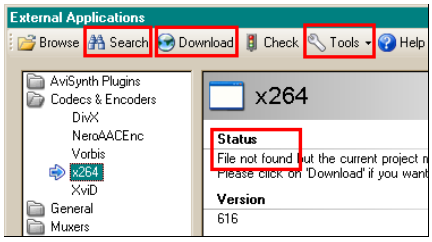

Abbildung 76: Dialog für externe Tools.

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 25 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 setzen nach Lust und Laune den Haken, dass StaxRip zukünftig nicht mehr meckern soll. Die folgenden Anwendungen benötigen wir nie für ein DVD-Backup:

- DirectShow Source
- Java
- ProjectX
- DivXMux (Damit beschäftigen wir uns im Encodingwissen nicht.)
- PMP Muxer (Damit beschäftigen wir uns im Encodingwissen nicht.)

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, klicken wir im StaxRip-Hauptfenster im **Source**-Abschnitt auf den **...**-Button und gelangen in folgenden Dialog. Wie wir den Film öffnen, hängt davon ab, ob wir die VOBs schon manuell mit DGIndex behandelt haben. Wenn ja, klicken wir auf **Add** und öffnen die vorher erzeugte D2V-Datei.

Ansonsten fügen wir über **Add** sämtliche VOBs hinzu, die wir beim Ripping erzeugt haben (Abbildung 77). Eine Kontrolle, ob die Reihenfolge tatsächlich stimmt, kann nicht schaden. Notfalls lassen sich Fehler mit den **Up** und **Down**-Buttons beheben. Weiter unten im Fenster muss **Chain Files** angehakt sein, damit StaxRip die ganze Dateiliste als einen Film behandelt und nicht jede VOB als separaten Film ansieht. Alle anderen Haken sollten nicht gesetzt sein. 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.

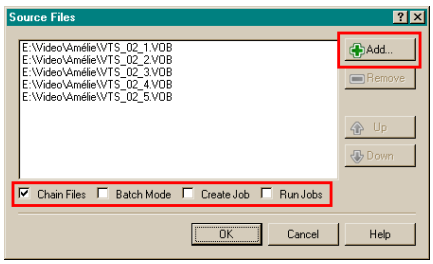

Abbildung 77: Quelldateien laden.

Das Laden der D2V-Datei benötigt ein paar Sekunden. Sollte StaxRip dabei in einem Dialog nach einem **Template** fragen, entscheiden wir uns für **DVD** oder 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 (Abbildung 78). Unter **Crop** sehen wir die automatisch zugeschnittene Auflösung; natürlich nur, wenn wir das Auto-Cropping nicht deaktiviert haben. Die **Anamorphic**-Checkbox dürfen wir nicht falsch verstehen. Die bezieht sich auf die *Quelle*, also die DVD. Ob unser Ziel ein traditionelles oder anamorphes Encoding sein soll, hat mit dieser Checkbox nichts zu tun. StaxRip setzt den Schalter automatisch auf den richtigen Wert, was wir auch nicht ohne guten Grund ändern sollten.

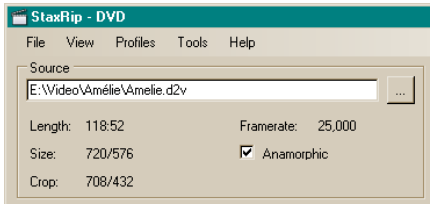

Abbildung 78: Eigenschaften der Quelle.

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 ▸ A.2.3 unserer Zieldatei. Das DivX Media Format (**DIVX**) und den PlayStation-Container (**PMP**) lassen wir außen vor und konzentrieren uns auf üblichen Standardcontainer: Matroska, MP4 und AVI.

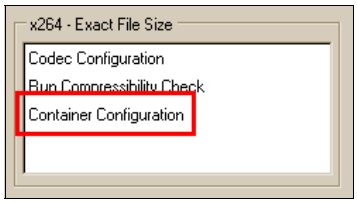

Abbildung 79: Menüpunkt für die Container-Konfiguration.

Bei der Wahl des Encoders stellt StaxRip automatisch auf den dafür vom Programm bevorzugten Container um. Das kann lästig werden, weil deswegen oft die Detailkonfiguration des Containers verloren geht. Es ist also eine gute Idee, den Encoder vor dem Container zu wählen. Um dessen detaillierte Konfiguration können wir uns dann später kümmern.

Um den Container zu konfigurieren, d. h. Kapitel und dynamische Untertitel ▸ A.2.6 einzubinden, klicken wir im Konfigurationsabschnitt auf der rechten Seite des StaxRip-Fensters auf **Container Configuration** (Abbildung 79).

Matroska

Der Dialog für den Matroska-Container ist in Abbildung 80 dargestellt. Über den **Öffnen**-Button unter **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.

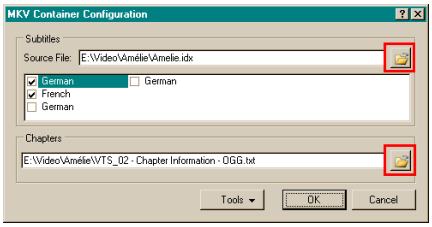

Abbildung 80: Matroska-Setup.

StaxRip unterstützt für Matroska leider keine SubRip-Untertitel. Das heißt aber nicht, dass wir SubRip nicht verwenden können. Lediglich müssen wir nach beendetem Encoding noch einmal selbst Hand anlegen. Solange wir mit StaxRip arbeiten, ignorieren wir SubRip-Untertitel einfach.

MP4

Der Dialog für den MP4-Container (Abbildung 81) hat die gleichen Funktionen wie der für Matroska. Unter **Subtitles** fügen wir über den **Add**-Button SubRip-Untertitel hinzu. Vobsub-Untertitel sind nicht möglich, was nicht an StaxRip liegt, sondern am MP4-Format an sich. Unter **Name** können wir eine kurze Beschreibung der Untertitelspur eintragen, **Path** enthält den Pfad zur SubRip-Datei und **Language** gibt die Sprache der Spur an. Mit **OK** fügen wir die Spur schließlich dem Encoding hinzu. Zurück im Konfigurationsfenster können wir im **Chapters**-Abschnitt wie bei Matroska die Datei mit der Kapitelliste laden.

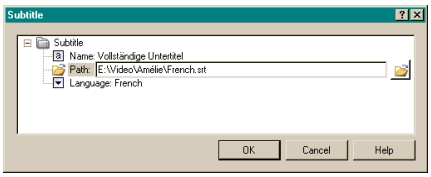

Abbildung 81: MP4-Setup.

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 (Abbildung 82). Hier steht schon der automatisch vorgeschlagene Dateiname für den fertigen Film. Die Dateiendung wird automatisch für den gewählten Container angepasst. Darunter bei **Size** legen wir die Größe des Films fest. Entweder wählen wir eine der Vorgaben aus dem **Pfeil**-Menü daneben oder tippen den MByte-Wert selbst sein.

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

Abbildung 82: Eigenschaften der Zieldatei.

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.

Audiospuren konfigurieren

Im Abschnitt **Audio** des StaxRip-Fensters (Abbildung 83) 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 Profils. Sämtliche Einstellungen nehmen wir über das Menü vor, das sich bei einem Klick auf die **Pfeil**-Buttons ganz rechts öffnet und in Abbildung 84 zu sehen ist.

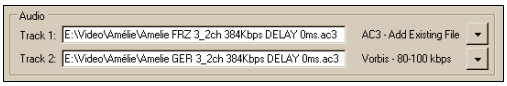

Abbildung 83: Audiospuren des Encodings.

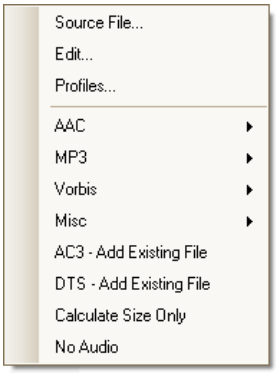

Abbildung 84: Konfigurationsmenü für Audiospuren.

Über den Menüpunkt **Source File** laden wir die Quelldatei, und zwar ganz unabhängig davon, wie wir die Tonspur in die Zieldatei einbinden wollen und in welcher Form die Quelle vorliegt. Das kann die AC3-Datei frisch von der DVD sein oder eine fertig transcodierte Tonspur, die wir mit BeLight erstellt haben (vgl. Seite 129 ff.). In jedem Fall ist **Source File** dafür zuständig. StaxRip trennt also immer zwischen dem Laden der Datei und der Auswahl des Transcoding-Modus.

Anschließend entscheiden wir uns für ein Audioformat ▸ A.2.5 und wählen aus dem unteren Abschnitt des Menüs das passende Profil. Um die Original-AC3 z. B. in eine 6-Kanal-HE-AAC-Spur umzuwandeln, würden wir aus dem Untermenü von **AAC** das Profil **HE 5.1 VBR 200-250 kbps** nehmen. Für jedes Format existiert außerdem ein Menüpunkt **Add Existing File**. Der ist dann wichtig, wenn wir nicht transcodieren wollen, sondern eine schon fertige Audiospur so wie sie ist in die Zieldatei übernehmen. Typischster Fall dafür ist die Original-AC3. Aber auch schon fertig transcodierte Dateien kommen in Frage. Bei AAC und MP3 müssen wir darauf achten, welche Art von Tonspur wir einbinden: AAC LC oder HE und MP3 VBR oder CBR. Es existieren jeweils zwei verschiedene Menüpunkte.

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 AC3 zu übernehmen und die unwichtigere in platzsparendes HE-AAC umzuwandeln.

Der Menüpunkt **Calculate Size Only** berücksichtigt die Größe der mit **Source File** geladenen Tonspur, muxt diese aber nicht in die Zieldatei. **No Audio** ignoriert die angegebene Quelldatei vollständig, so als hätten wir sie nie geladen.

Der **Edit**-Button hat zwei Funktionen. Die erste ist immer interessant: Wir konfigurieren über **Edit** die Sprache und den Delay-Wert der Audiospur. Je nachdem, ob wir ein Transcoding durchführen oder eine fertige Datei hinzufügen, sieht der Dialog hinter dem **Edit**-Button etwas anders aus. Der erste Screenshot in Abbildung 85 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.

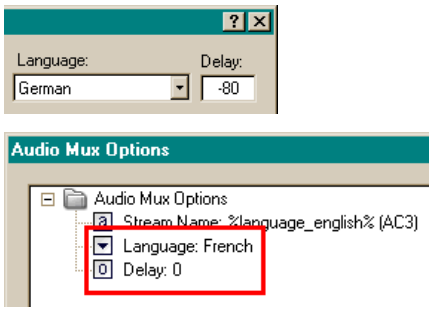

Abbildung 85: Auswahl von Sprache und Delay-Wert einer Audiospur.

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 Abbildung 86. 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 exakter Wert sein, denn nach dem Transcoding rechnet StaxRip sowieso mit dem tatsächlichen Wert weiter.

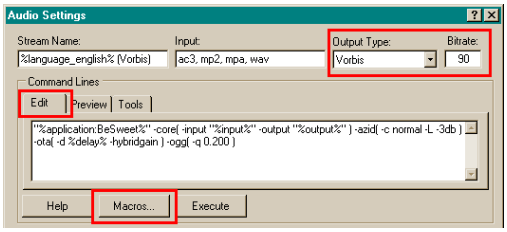

Abbildung 86: StaxRips BeSweet-Konfiguration.

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-Referenz* im Gleitz-Wiki. 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. Ein vollständige Liste der Makros können wir über den Button **Macros** aufrufen.

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) ▸ A.3.1, das Skalieren auf die Zielauflösung (Resizing) ▸ A.3.3, das Konfigurieren der AviSynth-Filter und evtl. das Einbinden von eingebrannten Untertiteln ▸ A.2.6. Zum Extrahieren der Untertitel von der DVD vgl. Kapitel B.3.

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. Den Croppingdialog erreichen wir über **F4** oder **View / Crop** (Abbildung 87). 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 rechts außen), 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.

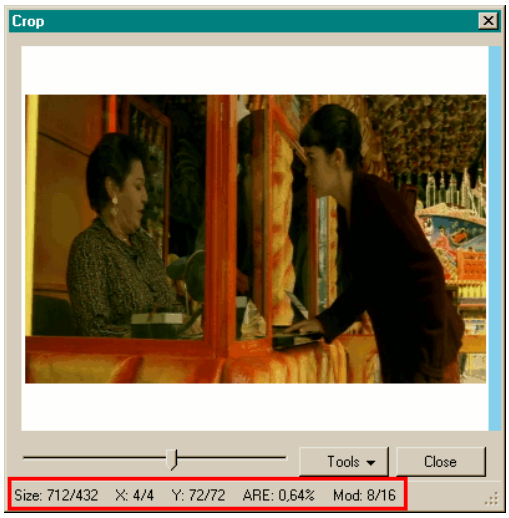

Abbildung 87: StaxRips Cropping-Dialog.

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 54) schon genauer besprochen.

Besonderheiten für anamorphes MPEG-4

Unter **Mod** sehen wir, 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 **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 208 das Spezialkapitel zu den Mod-Regeln zu Gemüte führen und dann hier weiterlesen.

Befinden wir uns in einer Situation, die mod8 sinnvoll erscheinen lässt, ändert sich nichts. Nach wie vor schneiden wir das Bild anhand der **Mod**-Angabe passend zurecht. Wichtig ist allerdings, im Menü des Hauptfensters unter **View / Project Options / Advanced / Advanced Automation** die Option **Auto correct crop values zu deaktivieren**, denn sonst erzwingt StaxRip ein Mod16-Cropping. Bis Version 0.9.9.5 heißt die Option **Save Cropping** und ist im selben Fenster unter **Dangerous** zu finden.

Ruft die Situation dagegen nach FillMargins, bleiben wir beim üblichen Mod16-Kriterium. Mit dem Übertünchen der Balkenreste beschäftigen wir uns ein Stück weiter unten bei der Filterkonfiguration. Im Moment notieren wir uns nur, wie viele schwarze Pixelreihen jeweils an den Rändern übrig geblieben sind.

Bild skalieren (Resizing)

Nach dem Zuschneiden schließen wir den Cropping-Dialog und wählen über den Schieber unter **Resize** die Zielauflösung (Abbildung 88) – es sei denn, wir encodieren anamorph, denn dann bleibt die Quellauflösung unverändert, wenn man vom Cropping absieht. Welche horizontale Auflösung der Schieber mindestens und höchstens zulässt, können wir unter **Tools / Settings / General / Resolution Slider** verändern. Nötig dürfte das für ein DVD-Backup kaum sein.

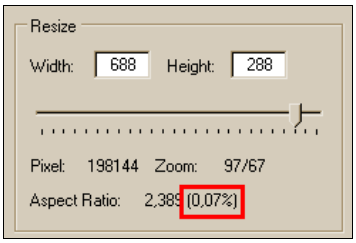

Abbildung 88: StaxRip-Einstellungen zum Skalieren.

Wir wählen also eine Auflösung, die zum Encoding passt. Den Zusammenhang zwischen Auflösung und Qualität haben wir uns im Kapitel zur Zielauflösung (Seite 55) 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 Pixel allzu sehr zu überschreiten. Besonders lange/kurze Filme und besonders große/kleine Audiospuren sollten wir natürlich einkalkulieren.

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 (Abbildung 89). 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.

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.**

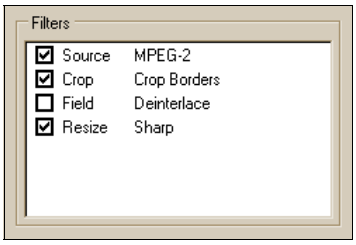

Abbildung 89: Filterliste des AviSynth-Skripts.

Hat unser Film eingebrannte Untertitel, stellen wir über das Menü **Tools / Add Forced Subtitle** die mit Vobsub erstellte und bearbeitete IDX-Datei ein. Dynamische 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 StaxRip zu einem anamorphen Encoding zu bewegen, 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 um das Setzen des AR-Flags in der Encoder-Konfiguration kümmert.

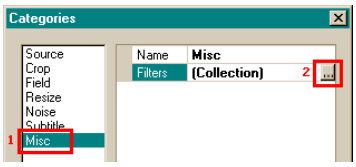

Abbildung 90: Filterkategorien.

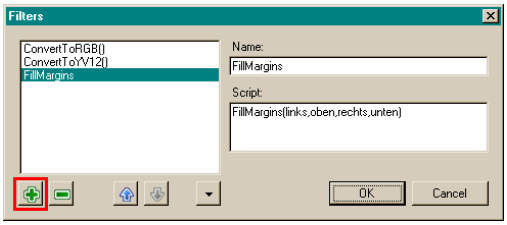

Abbildung 91: Dialog Filter hinzufügen.

Die zweite Besonderheit betrifft diejenige Cropping-Situationen, in der wir eine Mod16-Auflösung mit minimalen Balken zugeschnitten haben und diese Balken jetzt mit FillMargins übertünchen müssen.

Beim allerersten Einsatz von FillMargins laden wir zuerst den Filter von Tom Barrys Homepage *www.tbarry.com* herunter und kopieren die FillMargins.dll in den AviSynth-Pluginordner zu den anderen Filtern. Dann legen wir den Filter an, da der nicht serienmäßig in den Listen eingetragen ist. Dazu rufen wir mit einem Rechtsklick in die Filterliste das Kontextmenü auf und wählen den Eintrag **Profiles**, was uns ins Fenster von Abbildung 90 befördert. Am besten passt FillMargins in die **Misc**-Kategorie (1). Die klicken wir an, dann folgt auf der rechten Seite ein Klick auf **Filters** und noch einer auf den nun erscheinenden **...**-Button (2). Damit befinden wir uns im nächsten Dialog, der alle Filter in der Kategorie **Misc** auflistet (Abbildung 91).

Über den grünen **Plus**-Button unten links fügen wir einen neuen Filtereintrag hinzu, dessen Eigenschaften wir auf der rechten Seite folgendermaßen ausfüllen:

- **Name**: `FillMargins`
- **Script**: `FillMargins(links,oben,rechts,unten)`

Über zweimal **OK** gelangen wir zurück ins StaxRip-Hauptfenster, wo uns jetzt FillMargins als neuer Filter zur Verfügung steht, und zwar immer, nicht nur für das aktuelle Encoding.

Um FillMargins zu aktivieren, fügen wir wie bei allen anderen Filtern auch über **Rechtsklick / Add / Misc / FillMargins** den passenden Eintrag zur Liste hinzu und verschieben ihn direkt unter **Crop**. Beim Cropping haben wir uns gemerkt, wie viele schwarze Pixelreihen an jedem Bildrand übrig bleiben. Über **F4** können wir das noch einmal überprüfen. Diese Werte müssen wir jetzt dem Filter mitteilen. Dazu bearbeiten wir FillMargins per Rechtsklick auf seinen Eintrag und **Edit**. In der Zeile **Script** des Konfigurationsdialogs ersetzen wir die Platzhalter `links`, `oben`, `rechts` und `unten` durch die gemerkten Werte. Um am linken und rechten Rand jeweils zwei Pixel zu übertünchen und oben/unten keines, würden wir die Zeile so wie in Abbildung 92 anpassen auf:

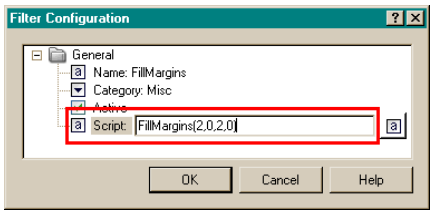

Abbildung 92: Filter aus der Liste bearbeiten.

```
FillMargins(2,0,2,0)
```

Ein Klick auf **OK** bestätigt die Änderung, und damit haben wir FillMargins erfolgreich eingebunden und konfiguriert.

B.4.1.5 Konfigurieren des Videoencoders

Zwei Schritte fehlen uns noch bis zum Start des Encodings. Den Videoencoder ▸ A.2.4 müssen wir noch konfigurieren und dann einen Kompressionstest durchführen, 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 eben noch ein bisschen anpassen.

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 Abbildung 93 gezeigt den gewünschten Videocodec (**Xvid** oder **DivX**), und zwar jeweils den Modus **Exact File Size**, um ein 2-Pass-Encoding durchzuführen. Die endgültige Größe unseres Films ist mit der gewählten Anzahl an CDs/DVDs fest vorgegeben, weshalb wir mindestens zwei Codierdurchgänge benötigen, um diese Größe exakt zu treffen.

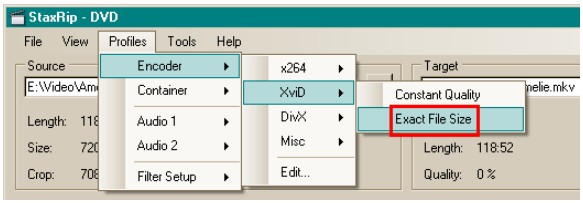

Abbildung 93: Encoderauswahl im Hauptmenü.

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 69 und DivX VfW ab Seite 113 ausführlich unterhalten.

Lediglich das anamorphe Encoding mit DivX benötigt einen zusätzlichen Hinweis. DivX unterstützt seit Version 6.5 das Setzen des AR-Flags im Videostream. StaxRip nimmt diese Einstellung normalerweise automatisch vor, ist aber bis Version 0.9.9.5 noch nicht auf DivX 6.5 vorbereitet. Solange StaxRip also noch nicht offiziell kompatibel zum neuen DivX ist (möglicherweise mit Version 0.9.9.6?), müssen wir uns manuell um die richtige Einstellung für das AR-Flag kümmern. Die Codec-Dialoge erreichen wir über die Konfigurationsliste auf der rechten Seite des StaxRip-Hauptfensters (Abbildung 94). Wir stellen 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 achten wir auf **1-pass quality based** mit einem **Target Quantizer** von **2**.

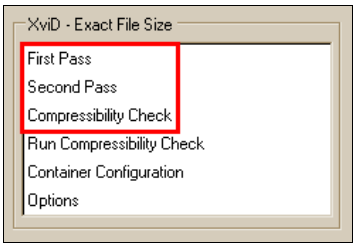

Abbildung 94: Konfigurationsbuttons für Xvid und DivX.

x264 Encoding-Profile

Wer sich für den x264-Encoder entscheidet, darf sich freuen. Denn x264 stellt in **Profiles / Encoder / x264** zusätzlich das Untermenü **More** mit einer ganzen Reihe von vordefinierten Profilen bereit. Dabei handelt es sich um die von Sharktooth für MeGUI entwickelten x264-Konfigurationen, die einen weiten Bereich typi-

scher Encodings abdecken. Dank der Profile brauchen wir uns im Detail kaum um die x264-Konfiguration kümmern: einfach das passende Profil aussuchen und encodieren. Alle Profile, die mit einem der Kürzel **AE**, **CE**, **HQ** oder **PD** beginnen, nutzen das 2-Pass-Encoding und sind damit für unsere Zwecke brauchbar. Ausführliche Informationen finden wir im Doom9.org-Forums im Thread *MeGUI Custom x264/AVC video profiles*. Hier eine kurze Übersicht.

AE

Die AE-Profile sind besonders auf Anime und Cartoons abgestimmt. **Standard** ist das schnellste, **Maxquality** das langsamste.

CE

Steht für *Common Encoding*, also ein typisches Allerweltsencoding. Der Qualität zuliebe sollten wir uns für **Mainprofile** oder gar **Highprofile** entscheiden.

HQ

Steht für *High Quality* und eignet sich für hochqualitative Encodings, die x264 richtig ausreizen. Nachteil: Die Encodinggeschwindigkeit bricht gegenüber den CE-Profilen deutlich ein. **HQ-Slow** dürfte von allen Profilen der sinnvollste Kompromiss zwischen Qualität und Geschwindigkeit sein.

PD

Steht für *Portable Devices*, also ein Encoding für Handys, iPods etc. Da man das kaum mehr als DVD-Backup bezeichnen kann, werden wir uns im Encodingwissen nicht weiter um die PD-Profile kümmern.

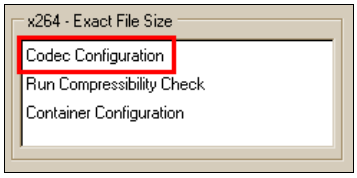

Abbildung 95: Konfigurationsbutton für x264.

Natürlich hindert uns niemand daran, auch bei x264 die Konfiguration selbst zu erledigen, indem wir wie bei Xvid und DivX den Menüpunkt **Exact File Size** anstatt eines der Profile wählen. Und auch jedes Profil lässt sich noch manuell anpassen. Das geschieht auf der rechten Seite des StaxRip-Fensters in der Konfigurationsliste, und zwar mit einem Klick auf **Codec Configuration** (Abbildung 95).

Wenn wir uns schon im Hintergrundwissen ab Seite 89 die x264-Konfiguration zu Gemüte geführt haben, sollte das Konfigurationsfenster selbsterklärend sein. Nur weil der Dialog etwas anders aussieht jetzt sämtliche Optionen ein zweites mal zu erklären, kommt mir reichlich sinnlos vor.

Kompressionstest und Qualitätsabstimmung

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.

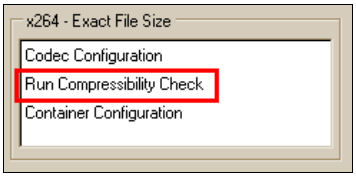

Abbildung 96: Start-Button für den Kompressionstest.

Gestartet wird der Kompressionstest über den entsprechenden Eintrag in der Konfigurationsliste (Abbildung 96). 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 (Abbildung 97).

Abbildung 97: Qualitätsangabe im Target-Abschnitt.

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.

Bei zu wenigen Prozenten (unter 50 % dürfte es so richtig kritisch werden) müssen wir entweder Platz schaffen (kleinere Audiospuren, höhere Zielgröße) oder die Auflösung verringern. Bei zu hohen Werten können wir umgekehrt die Auflösung erhöhen oder Platz verschwenden. Vielleicht reicht es doch für den Original-AC3-Sound oder zusätzlich die deutsche Tonspur (Ok, für die meisten eher zusätzlich die Originalsprache. Dann Untertitel nicht vergessen!). Wer noch auf CDs encodiert, kann im glücklichsten Fall sogar einen Datenträger einsparen. Noch einmal ins Kapitel zur Zielauflösung (Seite 51) 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. 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 Abbildung 98 erscheint. Hier schließlich passiert der endgültige Klick auf den fetten **Start!**-Button. Und dann können wir uns für die nächsten Stunden zurücklehnen.

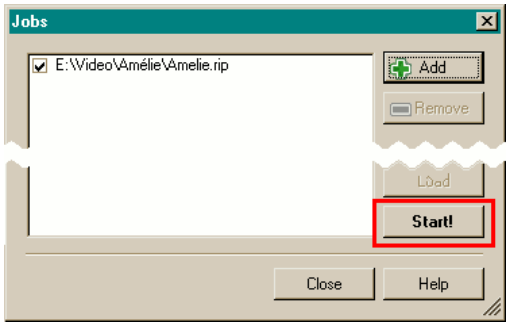

Abbildung 98: StaxRip-Jobliste.

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 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.

Allerdings gibt es gute Gründe, die dagegen sprechen. Da sich Gordian Knot fürs Encoding ganz zentral und ausschließlich auf VirtualDubMod verlässt, legt es

sich damit genauso auf das VfW-Encodingframework ▸A.4.1 fest. Diese veraltete Technologie ist so untrennbar mit Gordian Knot verzahnt, dass es nicht lohnend erscheint, das Programm an modernere Verfahren anzupassen.

Gerade fürs DVD-Backup in AVI-Dateien stellt Gordian Knot trotzdem nach wie vor ein ausgereiftes und bestens funktionierendes Frontend dar. Natürlich spielt auch der Nostalgiefaktor kräftig mit. Viele der heutigen Experten haben ihre ersten Schritte mit Gordian Knot gemacht – möglicherweise sogar noch mit DivX 3.11. Und wer will schon die altvertraute, zuverlässige Oberfläche zum alten Eisen geben, mit der man noch im Tiefschlaf ein einwandfreies DVD-Backup zustande bringen würde. :)

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-Click-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.

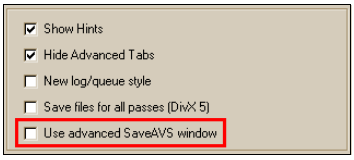

Abbildung 99: Globale Optionen in Gordian Knot.

Anschließend wechseln wir ins Register **Options**. Wichtig ist hier zuerst der fehlende Haken bei **Use advanced SaveAVS window** (Abbildung 99). Nicht, dass das erweiterte Fenster schlecht wäre. Ich mag es sogar sehr gerne. Aber alles, was wir hier im Encodingwissen besprechen, bietet auch das einfachere Fenster. Und das lässt sich schön kompakt mit einem einzelnen Screenshot darstellen. Also raus mit dem Haken. Die restlichen Optionen dürften einigermaßen selbsterklärend sein. Mit den **Advanced Tabs** sind die Registerkarten fürs DivX-3.11-Encoding gemeint. Vielleicht sollte man die langsam als »obsolete tabs« bezeichnen ;). Deutlich interessanter wird es dann auch bei den **Default codec settings** ▸A.4.1 ▸A.2.4.

Achtung! DivX 6 unterscheidet sich so stark von seinen Vorgängern, dass er nicht ohne Probleme mit Gordian Knot benutzt werden kann. Um die neue DivX-Generation sicher zu verwenden, müssen wir das Encoding manuell mit VirtualDubMod durchführen, Gordian Knot also nur als Rechner und Skript-Generator benutzen. Der letzte DivX 5 ist Version 5.2.1, dessen Konfiguration wir als Spezialthema in Kapitel C.1.2 unter die Lupe nehmen.

Die in Abbildung 100 getroffenen Einstellungen gelten als Standard für alle Encodings, können aber im Einzelfall auch wieder überschrieben werden.

Das Video wird unabhängig vom Codec in mindestens zwei Durchgängen (Passes) codiert. Im ersten Durchgang sammelt der Codec Informationen über das Video ohne den Film schon wirklich zu encodieren. Aus diesen Informationen wird eine möglichst gute Verteilung der verfügbaren Bitrate berechnet und im zweiten Durchgang der Film erzeugt. Xvid ist an dieser Stelle fertig. Bei DivX und x264 sind noch zusätzliche Durchgänge möglich, um die Qualität weiter zu steigern.

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

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

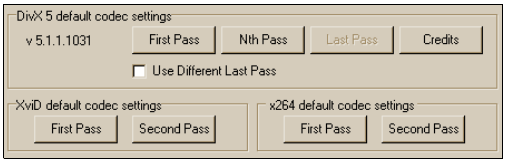

Abbildung 100: Gordian Knots Standard-Codecsetup.

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

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

B.4.2.2 Kalkulieren der Bitrate

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

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

Bei **Mode** oben links stellen wir ein, ob wir die Bitrate oder die endgültige Dateigröße des Films berechnen wollen. Da die Größe durch die gewünschte Anzahl CDs bzw. DVDs schon feststeht, stellen wir **Calculate average bitrate** ein. Dann wählen wir **Container** und **Codec** aus und kümmern uns um die endgültige Größe des Films.

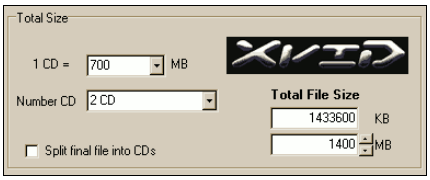

Abbildung 101: Auswahl der Zielgröße in Gordian Knot.

Bei **Total Size** (Abbildung 101) stellen wir die Rohling-Größe und gewünschte Anzahl Discs ein. Alternativ können wir auch rechts direkt die gewünschte Größe in MB angeben. Bei mehreren CDs macht es Sinn, die **Total File Size** ein paar MByte nach unten zu schrauben. Schließlich müssen wir den Film zum Schluss an einer sinnvollen Stelle schneiden, um ihn auf die CDs aufzuteilen. Da schadet es nicht, ein wenig Luft zu haben. Die **Split final file into CDs**-Funktion würde das zwar automatisch übernehmen, achtet dabei aber nur auf die Dateigröße, so dass es schon einmal vorkommen kann, dass der Held des Films mitten im Satz unterbrochen wird, weil die nächste CD fällig wird. Deshalb sollten wir das Splitten immer manuell erledigen.

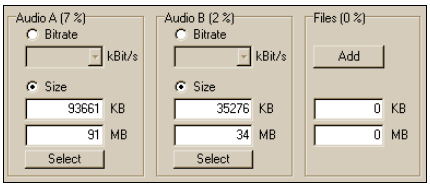

Abbildung 102: Größe der Audiospuren und zusätzlicher Daten.

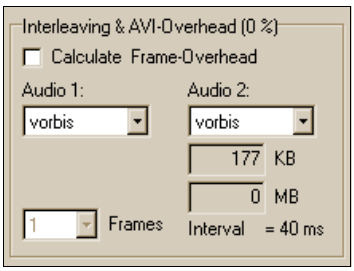

Abbildung 103: Berechnung des Overheads.

Der Abschnitt in Abbildung 102 enthält alle Daten, die außer dem Video auf die Disc müssen. Mit **Select** wählen wir die erste erstellte Audiospur ▸ A.2.5 aus, deren Größe in die Felder darüber übernommen wird. Das Ganze noch einmal für eine eventuelle zweite Audiospur. Der **Files**-Abschnitt ist für Dateien gedacht, die zusätzlich zum Film auf die Disc sollen, z. B. dynamische Untertitel oder Special Features von der DVD. Je mehr hier steht, desto geringer wird natür-

lich die Qualität des Films, das sollten wir immer bedenken.

Schließlich wählen wir bei **Interleaving** (Abbildung 103) 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 Bedarf verteilt, schnelle Szenen brauchen für die gleiche Qualität mehr Bits als langsame.

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

B.4.2.3 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** (Abbildung 104) den Typ der DVD ▸ A.2.2 und die Art des Bildes ▸ A.3.2 ein. **PAL**-DVDs mit einer Auflösung von 720 × 576 Pixeln sind Standard in Europa, Japan und dem Nahen Osten, **NTSC** mit 720 × 480 Pixeln findet in Amerika Anwendung. **Anamorphic (16:9)** ist die richtige Wahl für 16:9-DVDs, **non anamorphic (4:3)** dagegen gilt für 4:3-DVDs. Aus DGIndex wissen wir schon, welches Format die DVD hat, z. B. PAL

16:9. Das dürfte für die meisten Region-2-DVDs (Europa) zutreffen.

Die rechte Seite des **Resolution**-Registers (Abbildung 105) ist fürs Cropping ▸ A.3.1 zuständig, also für das Wegschneiden der schwarzen Ränder um das Video. Wer will, kann sich auf den **Auto-Crop**-Mechanismus verlassen. Ich traue solchen Automatiken aber grundsätzlich nicht so recht über den Weg und mache das Cropping deshalb lieber selbst. Dafür stellen wir auf **Pixel**. Jetzt nehmen wir uns das Fenster mit dem Video vor, springen dort zu einer möglichst hellen Stelle und schneiden mit den Reglern im Bild auf der rechten Seite die schwarzen 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.

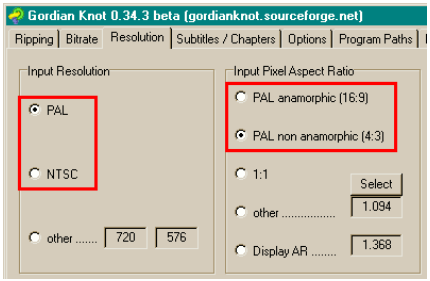

Abbildung 104: Quellvideo-Paramter im Resolution-Register.

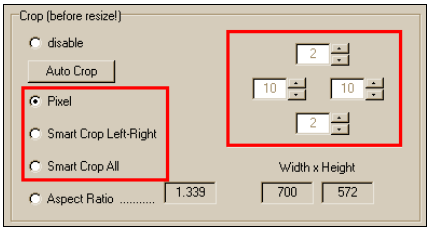

Abbildung 105: Cropping-Optionen.

Anschließend müssen wir im unteren Teil des Fensters (Abbildung 106) eine geeignete Auflösung wählen. Dazu stellen wir **W-Modul** und **H-Modul** beide auf **16**. Damit erlaubt Gordian Knot nur noch Auflösungen, die sowohl horizontal als auch vertikal durch 16 teilbar sind. Beim Einstellen der Auflösung mit dem großen Schieber müssen wir auf **Aspect Error** und **Bits/(Pixel*Frame)** (BPF oder relative Bitrate) achten. Wie wir eine möglichst gute Bildgröße finden, haben wir im Hintergrundwissen schon im Kapitel A.3.3 über die Zielauflösung betrachtet.

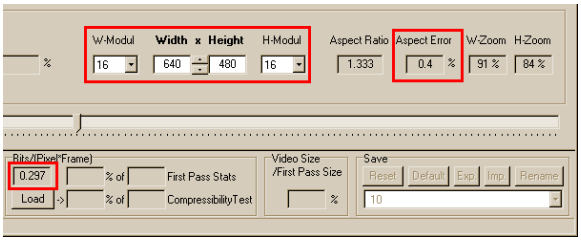

Abbildung 106: Wahl der Zielauflösung.

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

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

Anamorphe Zielauflösung

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

Cropping

Das zugeschnittene Bild sollte wie beim nicht-anamorphen Encoding ein durch 16 teilbares Seitenverhältnis haben (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 208 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 Mod8.

AviSynth-Skript bearbeiten

Bleiben mit Mod16-Kriterium nur zwei bis drei Pixel an Balken übrig, können wir den AviSynth-Filter *FillMargins* bemühen, um diese Ränder zu übertünchen. Vor dem allerersten Einsatz von FillMargins laden wir zuerst den Filter von Tom Barrys Homepage *www.trbarry.com* herunter. Die FillMargins.dll kopieren wir irgendwo hin, wo wir sie wiederfinden.

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 einem Haken bei **No comments** für Übersicht. Das Skript sieht jetzt in etwa so aus wie in Abbildung 107.

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

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

Für jeden Rand tragen wir ein, wie viele schwarze Pixelreihen übertüncht werden sollen. Zu viele sollten das nicht sein. Bei mehr als zwei bis drei dürften die Bildränder langsam anfangen, unsauber auszusehen.

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

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

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

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

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

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

Abbildung 107: AviSynth-Skript für anamorphes Encoding.

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** (Abbildung 108) können wir mit dem **Select**-Button eine SubRip-Datei laden. Leider unterstützt Gordian Knot im Moment nur eine einzelne SubRip-Spur. Sollen es mehr werden, müssen wir die nach dem Encoding manuell einbinden. Das gleiche gilt für dynamische Vobsubs. VirtualDubMod kann mit denen nichts anfangen, und da Gordian Knot auf VDubMod aufbaut, bleibt uns auch hier nur, später selbst Hand anzulegen. Außerdem ist die Untertitelwahl für den AVI-Container ▸ A.2.3 deaktiviert, denn SubRip in AVI funktioniert nur mit AVI-Mux GUI, nicht mit VirtualDubMod.

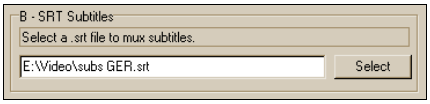

Abbildung 108: Einbinden von SubRip-Untertiteln.

Für die Kapitel ist die rechte Seite des Registers zuständig (Abbildung 109). 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.

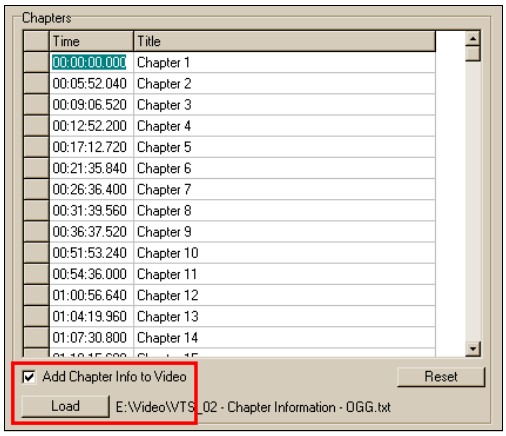

Abbildung 109: Geladene Kapitelliste im Ogg-Format.

B.4.2.5 Encoding-Parameter konfigurieren

Eigene Einstellungen für den Abspann

Im Video-Fenster (Abbildung 110) müssen wir uns entscheiden, ob wir den Abspann mit anderen Einstellungen encodieren als den Rest des Films. Wenn der Abspann nur aus weißer Laufschrift auf schwarzem Hintergrund besteht, können

wir ihm deutlich weniger Bits gönnen als dem Film, ohne dass sich die niedrigere Qualität zu deutlich bemerkbar macht. Dadurch wird der Abspann kleiner und für den Rest bleibt mehr Platz übrig. Besteht der Abspann aber aus mehr als Laufschrift (z. B. bei der *Monster AG*), sollten wir die Finger von der Funktion lassen. Ein mit Abspann-Einstellungen codiertes normales Videobild sieht grauenhaft aus. Außerdem ist ein zusätzlich geschrumpfter Abspann hauptsächlich für 1-CD-Encodings interessant, bei denen es sich tatsächlich lohnt, um jedes zusätzliche MByte für den eigentlichen Film zu kämpfen.

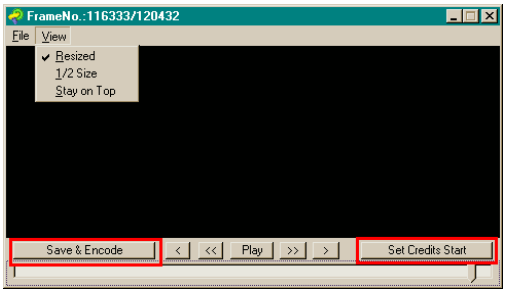

Abbildung 110: Video-Vorschaufenster von Gordian Knot.

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

Filter konfigurieren

Über den **Save & Encode**-Button gelangen wir anschließend zum Fenster aus Abbildung 111, in dem wir alle Filter konfigurieren. Hat unser Film feste Untertitel, stellen wir unter **Subtitles** (1) die mit Vobsub erstellte und bearbeitete IDX-Datei ein. Dynamische Untertitel interessieren hier nicht.

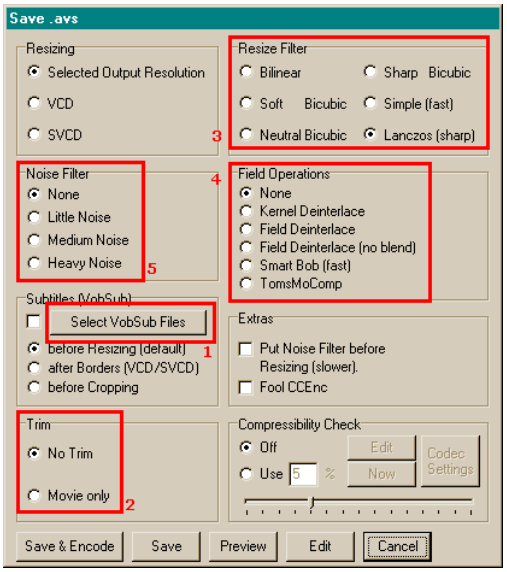

Abbildung 111: Dialog für die Filterkonfiguration.

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

Anders bei DivX, der intern keine Möglichkeit fürs Credits-Encoding besitzt (Abbildung 112). 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.

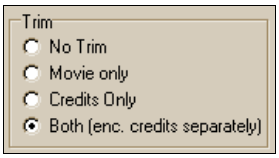

Abbildung 112: Trim-Setup für DivX.

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: **Field Deinterlace** oder **TomsMoComp** bieten sich an.

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

Der Kompressionstest

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

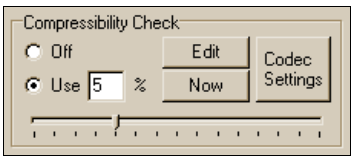

Abbildung 113: Optionen für den Kompressionstest.

Mit **Codec settings** (Abbildung 113) 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**. Und Folgendes gilt für x264: **Single Pass - Quantizer** als Encoding-Modus bei einem **Quantizer** von **18**. Den Rest so einstellen wie fürs Encoding des ganzen Films geplant.

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 Abbildung

114. Der wichtige Wert sind die **68.2** Prozent. Gut sind zwischen 60 und 80 Prozent. Darunter wird die Qualität schnell schlechter, darüber vergrößert sich nur die Datei ohne spürbare Steigerung der Qualität. Alles analog zum BPF-Wert, nur ein Stück genauer.

Bei zu wenigen Prozenten (unter 50 % dürfte es so richtig kritisch werden) müssen wir entweder Platz schaffen (kleinere Audiospuren, höhere Zielgröße) oder die Auflösung verringern. Bei zu hohen Werten können wir umgekehrt die Auflösung erhöhen oder Platz verschwenden. Vielleicht reicht es doch für den Original-AC3-Sound oder zusätzlich die deutsche Tonspur (Ok, für die meisten eher zusätzlich die Originalsprache. Dann Untertitel nicht vergessen!). Wer noch auf CDs encodiert, kann im glücklichsten Fall sogar einen Datenträger einsparen.

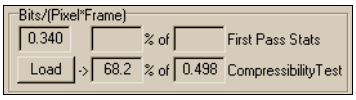

Abbildung 114: Prozentwert des Compchecks.

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

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

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

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

B.4.2.6 Das Encoding starten

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

Audiospuren einbinden

Mit **Select** (Abbildung 115) wählen wir die erste Audiospur aus. Gordian Knot hat hier schon automatisch die Datei eingetragen, die wir im Register **Bitrate** bei **Audio A** angegeben haben. Ist das eine AAC-Datei, müssen wir **Audio processing disabled** auswählen und uns später von Hand um die Tonspuren kümmern. Gordian Knot kann ja nicht mit AAC umgehen.
Bei allen anderen Audioformaten stellen wir auf **Just mux**, da wir das Transcoding schon mit BeSweet erledigt haben. Im Dateinamen steht ein DELAY-Wert. Der sagt uns, um wie viele Millisekunden die Spur nach vorne oder hinten verschoben werden muss, um synchron mit dem Video zu laufen. Dann, und *nur* dann, wenn wir diesen Wert in BeSweet noch *nicht* berücksichtigt haben, tragen wir bei **Delay** den ms-Wert ein und setzen für negative Delays den **negative**-Haken. Im Textfeld dann kein Minuszeichen angeben!

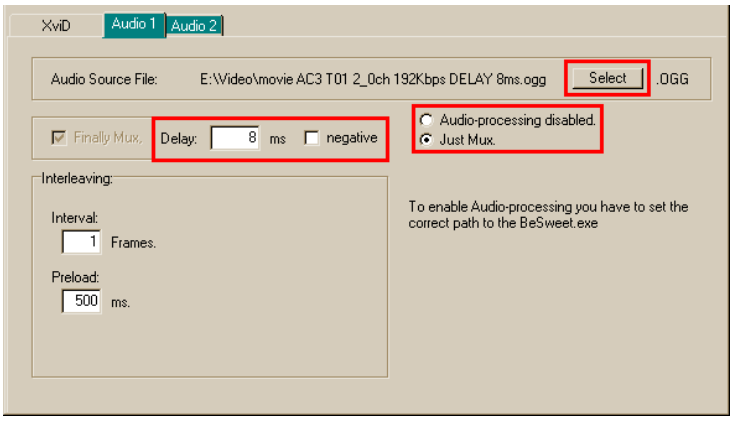

Abbildung 115: Konfiguration der Audiospuren.

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 (Abbildung 116) 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.

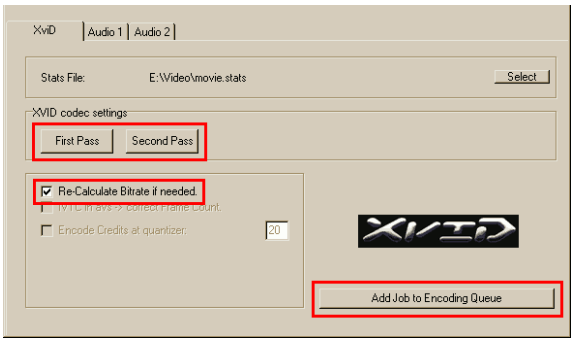

Abbildung 116: Abschließende Xvid-Konfiguration.

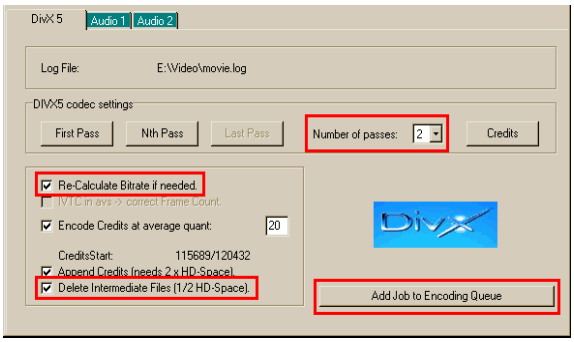

Abbildung 117: Abschließende DivX-Konfiguration.

Für DivX sieht der Dialog ähnlich aus (Abbildung 117). Mit **Number of passes** können wir auswählen, wie viele Passes Gordian Knot durchführen soll. Nicht übertreiben, mehr als drei dürfte meistens nicht nötig sein. Auch die Codec-Konfiguration können wir noch einmal anpassen.

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

AR-Flag für anamorphe Encodings setzen

Anamorphe Encodings ▸A.3.2 sollten in ihrem MPEG-4-Videostream das AR-Flag auf den richtigen Wert gesetzt haben, damit das Video nicht mit Eierköpfen abgespielt wird. Das erledigen wir jetzt im Rahmen der abschließenden Encoder-Konfiguration.

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 69 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 MPEG4 Modifier gehen (vgl. Seite 207). Ansonsten finden wir die Einstellung im **Video**-Register der Encoder-Konfiguration, wie im Kapitel zu DivX VfW ab Seite 113 erklärt.

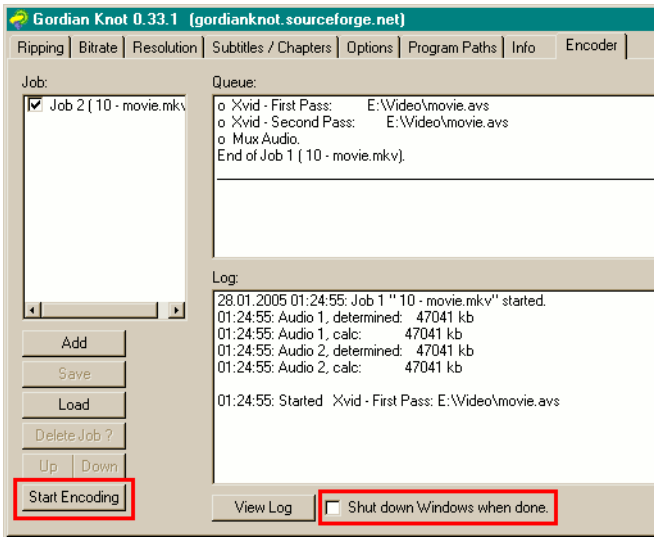

Abbildung 118: Encoder-Register von Gordian Knot.

Auch x264 bietet die Möglichkeit, ein AR-Flag zu setzen. Die Funktion arbeitet genauso wie bei Xvid und DivX, setzt also das MPEG-4-AR-Flag in der Videospur auf das angegebene PAR. Da x264 keine Standardwerte vorgibt, müssen wir einen Blick in Tabelle 4 auf Seite 44 werfen. Den passenden Wert geben wir im **More**-Register ein, wie in der x264-VfW-Konfiguration ab Seite 94 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 Abbildung 118 zu sehen.

Wenn wir Gordian Knots Nachfrage nicht schon bestätigt haben, starten wir über **Start Encoding** den Codiervorgang. Im **Queue**-Fenster sehen wir unseren

aktuellen Auftrag, darunter bei **Log** sämtliche Details dazu. Gordian Knot ruft jetzt VirtualDubMod auf, um den Film zu encodieren und anschließend die Audiospuren einzubinden.

Solange VirtualDubMod arbeitet, war früher herrlich Zeit, mal wieder mit der Freundin auszugehen oder ein paar Stunden Schlaf nachzuholen. Heutzutage kann man gerade noch in Ruhe ein Stück Kuchen vom Bäcker holen und Kaffee trinken … Naja – das ist vielleicht doch etwas übertrieben. Jedenfalls reicht es zum Ausspannen vor den Abschlussarbeiten: falls nötig den Film muxen und schnippeln, und natürlich brennen. Obwohl – auch brennen ist bei den heutigen Festplattengrößen vielleicht nicht mehr ganz so selbstverständlich.

B.4.2.7 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.
- Unterstützung für x264 CLI ▸ A.4.1.
- Volle Matroskaunterstützung über MKVMerge ▸ A.2.3.

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

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

Installation

Die aktuelle Version von agkp erhalten wir im ersten Posting des *Entwicklungsthreads* im Gleitz-Forum. Die Installation erfolgt in den VirtualDubMod-Ordner. Dafür muss vorher natürlich VirtualDubMod installiert sein. Wir entpacken das komplette agkp-Archiv in den VDubMod-Ordner. Dort treffen wir folgende Einstellung für alle drei Dateien *agkpv.exe, agkpa.exe* und *agkpAR.exe:* Mit einem Rechtsklick auf die Datei und dem Menüpunkt **Eigenschaften** gelangen wir in den Eigenschaftendialog aus Abbildung 119. Im Register **Programm** stellen wir sicher, dass **Nach Beenden schließen** angehakt ist, was unter Windows XP schon von Haus aus so sein sollte.

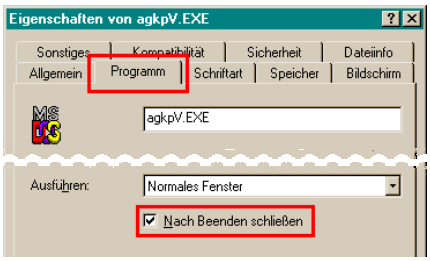

Abbildung 119: Dateieigenschaften der agkpV.exe.

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

Damit können wir Gordian Knot starten und ins Register **Program Paths** wechseln (Abbildung 120). Unter **Where is VirtualDubMod.exe located** stellen wir mit dem **Locate**-Button die *agkpV.exe* ein. Gordian Knot denkt also, dass es ganz normal VDubMod aufruft. Tatsächlich tritt aber agkp in Aktion. Entsprechend einfach ist es, trotz installiertem agkp ein klassisches GK-Encoding durchzuführen. Dazu ändern wir die VDubMod-Einstellung wieder zurück auf die *VirtualDubMod.exe.*

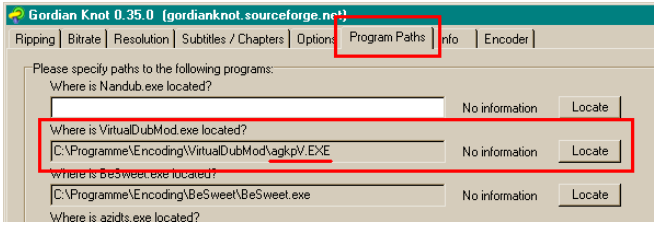

Abbildung 120: Einbinden von agkp in Gordian Knot.

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

x264-Encoding mit agkp

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

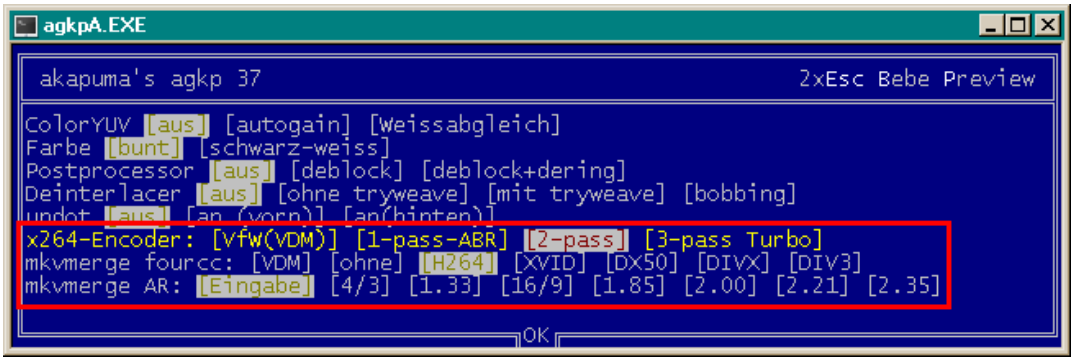

Abbildung 121: Agkp-Konfigurationsfenster.

Ist alles fertig eingestellt und das Encoding gestartet, erscheint das Auswahlfenster von agkp (Abbildung 121), das sogar die Maus unterstützt. :-) Die Einträge am Anfang betreffen die Filterkonfiguration, mit der wir uns hier nicht beschäftigen. Interessant sind die letzten drei Zeilen.

x264-Encoder

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

Mkvmerge fourcc

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

Mkvmerge AR

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

drücklich nicht. Das gilt nicht, wenn wir den VfW-Encoder verwenden, denn der setzt das MPEG-4-Flag wie von uns auf Seite 181 konfiguriert. Wie wir auch mit agkp ein anamorphes Encoding einschließlich MPEG-4-Flag erreichen, dazu mehr ab Seite 191.

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

Damit haben wir die Agkp-Konfiguration beendet und können das Encoding mit einem Klick auf den **OK**-Button unten in der Mitte starten. Im Fenster der *agkpV.exe* (Abbildung 124) lässt sich der Fortschritt verfolgen. Das Encoding läuft nun genauso automatisch ab wie von Gordian Knot gewohnt.

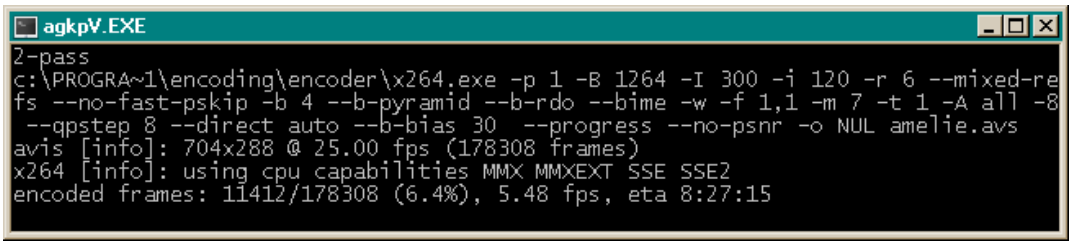

Abbildung 122: x264.exe bei der Arbeit.

Konfiguration: Die agkpx264.bat

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

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

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

Kommentare

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

Abschnitte

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

Echo

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

Die erste interessante Zeile ist Zeile 9.

```
set subme=7
```

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

Encoder-Abschnitte

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

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

Zeile 52

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

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

Zeile 53

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

Zeile 54

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

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.

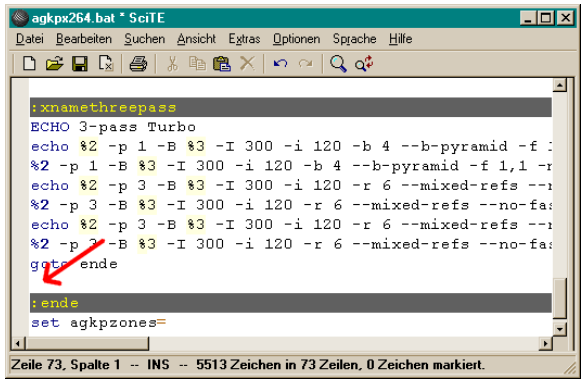

Abbildung 123: Position für neuen Encoder-Abschnitt in der agkp264.bat.

Zeilen 57, 58

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

Zeile 59

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

Einfügen eines neuen Abschnitts

Mit dem gerade besprochenen Schema können wir einen neuen Encoder-Abschnitt einfügen, womit uns eine zusätzliche Auswahlmöglichkeit fürs Encoding zur Verfügung steht. Die passende Stelle für die neue Konfiguration ist nach dem letzten `xname`-Abschnitt. Vor dem Marker `:ende`, da wohin der Pfeil in Abbildung 123 zeigt, fügen wir folgende Zeilen ein:

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

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

%2

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

%3

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

%4

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

%5

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

-m %subme%

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

%agkpzones%

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

Im Auswahlfenster von agkp (Abbildung 124) haben wir nun eine zusätzliche Option namens **Encwissen**, mit der wir die neue Konfiguration verwenden können.

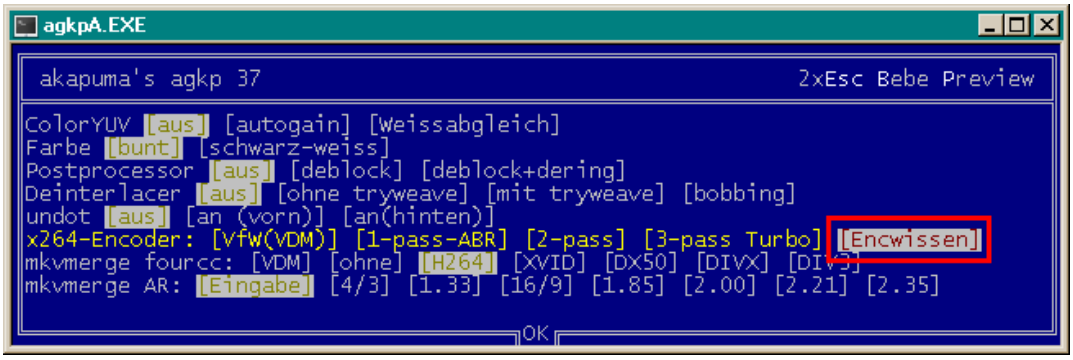

Abbildung 124: Neue Option im agkp-Auswahlfenster.

Xvid-Encoding mit agkp

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

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

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

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

Die Konfiguration ist genau die, wie sie in der Xvid-VfW-Konfiguration ▸ A.4.2.1 erklärt ist. Prinzipiell handelt es sich um die leicht angepassten Kommandozeilen aus dem Xvid-Cli-Kapitel.

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

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

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

Haben wir alles fertig konfiguriert und das Encoding gestartet, erscheint das nun schon bekannte Agkp-Auswahlfenster. Die Zeile **x264-Encoder** hat wie in Abbildung 125 einen neuen Eintrag bekommen, hinter dem sich unsere Xvid-Konfiguration verbirgt. Natürlich können wir uns an dieser Stelle noch anders entscheiden und auf eines der x264-Profile umstellen. Im Xvid-Fall ist es sinnvoll, auch den **mkvmerge fourcc** auf **XVID** zu ändern. Für **mkvmerge AR** gilt das gleiche wie im x264-Abschnitt auf Seite 184. Dann fehlt nur noch ein Klick auf **OK** und ein paar Stunden Geduld. Und dann soll noch einer sagen, Gordian Knot käme mit aktueller Encodingtechnologie nicht zurecht! :-)

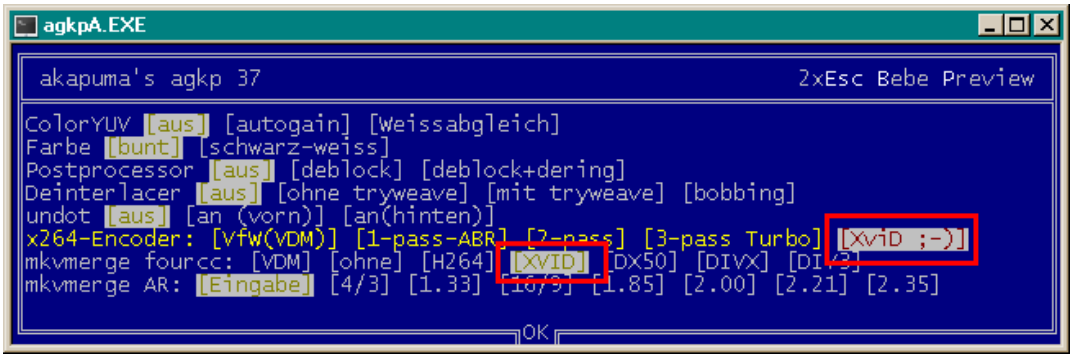

Abbildung 125: Agkp-Auswahlfenster mit Xvid-Hack.

Anamorphes Encoding mit agkp

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

Dafür öffnen wir die *agkpx264.bat* und definieren wie in Abbildung 126 nach der `subme`-Zeile (Zeile 9 in der unveränderten Datei) eine neue Variable. Für alle Befehle weiter unten in der Datei existiert nun eine neue Variable, die wir über `%pixelar%` ansprechen können. Darin enthalten ist der PAR, der ins MPEG-4-Flag geschrieben werden soll. Die `16:11` im Screenshot bezeichnen eine 16:9-PAL-DVD. Welches für den jeweiligen Film der richtige Wert ist, können wir der inzwischen sicher schon wohlbekannten PAR-Tabelle auf Seite 44 entnehmen. Für ein klassisches nicht-anamorphes Encoding setzen wir die Variable auf `1:1`.

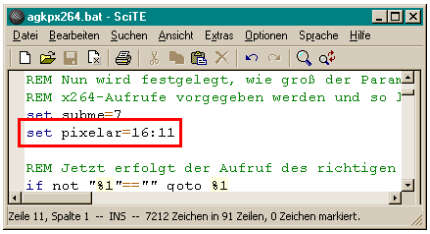

Abbildung 126: PAR-Variable in der agkpx264.bat.

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

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

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

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

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

```
set pixelar=
```

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

Abbildung 127: DAR-Angabe für das Matroska-AR-Flag.

Im agkp-Auswahlfenster setzen wir wie in Abbildung 127 zusätzlich das Seitenverhältnis im Matro-

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

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

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

B.5 Manuelles Muxing

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

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

B.5.1 MKVMerge für den Matroska-Container

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

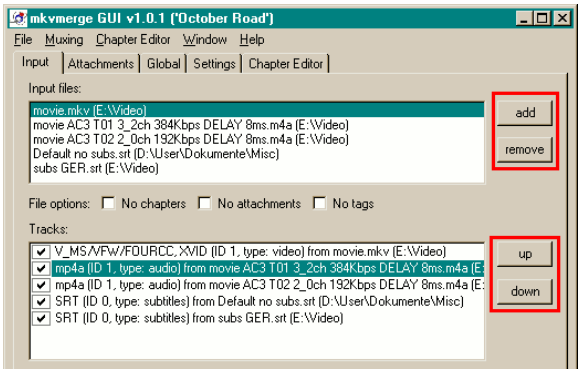

Abbildung 128: Hinzufügen von Quellen in MKVMerge GUI.

Das Programm aus MKVToolnix, um alles zusammenzufügen, heißt MKVMerge und ist genauso wie BeSweet eine Konsolenanwendung. In Abbildung 128 sehen wir die grafische Oberfläche dazu, die sich sich MKVMerge GUI (mmg) nennt. Die starten wir. Mit **Add** oder per Drag & Drop können wir alle benötigten Dateien hinzufügen, im Bild sind das die von GKnot erzeugte Videodatei, zwei AAC-Audiospuren und zwei SubRip-Untertitel. Die Spuren sind am Ende in der gleichen Reihenfolge in der Datei gespeichert wie sie hier erscheinen. Mit **Up** und **Down** können wir Spuren verschieben. Im unteren Teil des Fensters (Abbildung 129) konfigurieren wir die einzelnen Spuren, und zwar immer gerade diejenige, die im **Tracks**-Fenster angewählt ist.

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

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

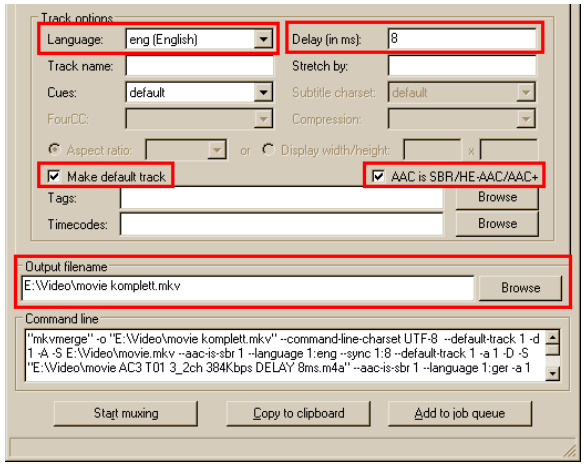

Abbildung 129: Audio-Konfiguration in mmg.

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

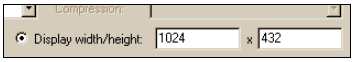

Abbildung 130: AR-Eingabe in Form der Auflösung.

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

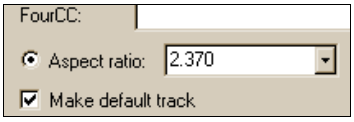

Abbildung 131: AR-Eingabe in Form des DAR.

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

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

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

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

Entsprechend müssten wir wie im Screenshot in die beiden Felder **1024** und **432** eintragen. Die zweite Möglichkeit, den DAR anzugeben, ist die Option **Aspect Ratio** (Abbildung 131).

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

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

DAR = Zielbreite / Zielhöhe

Für unser *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.

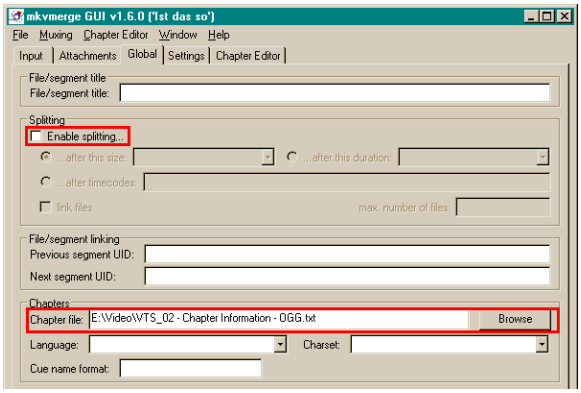

Abbildung 132: Globale Matroska-Einstellungen in mmg.

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

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

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

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

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

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

Teil C
Spezialthemen

Einleitung

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

C.1 Praktisches

C.1.1 Manuelles Splitting

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

Grundsätzliches zum Splitting

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

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

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

Ein sinnvoller Schnitt erfolgt bei einem **Szenenwechsel.** Es ist unheimlich lästig, wenn der Held der Geschichte mitten im Satz unterbrochen wird, weil es Zeit

für die zweite CD wird. Solche Schnitzer geben auch dem professionellst codierten Film einen laienhaften Beigeschmack. Eine Einschränkung muss ich gelten lassen. Wenn der Film gerade so auf zwei CDs passt, ist kaum Spielraum zum Splitten vorhanden. Deswegen ist es manchmal doch nötig, den Schnitt innerhalb einer Szene zu setzen. Ein bisschen Fingerspitzengefühl kann dabei aber auch nicht schaden. Vor allem nie, nie mitten im Satz schneiden!

VirtualDubMod als Splitting-Zentrale

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

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

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

Abbildung 133 zeigt die benötigten Kontrollelemente. Mit den **Keyframe-Buttons** (2) können wir keyframe-weise durch den Film springen, um eine günstige Stelle zum Schneiden auszusuchen. Vorwärts springen ist dabei nur bedingt sinnvoll, schließlich ist da die CD zu Ende. Die genaue Dateiposition sehen wir unter (4). So wissen wir exakt, bei wie viel Megabyte wir uns befinden (fürs Splitting über MKVMerge ist das besonders wichtig). die Exaktheit bezieht sich leider nicht auf AVIs mit VBR-MP3-Audio. Bei solchen Dateien müssen wir ca. 5 MB dazurechnen, um auf den richtigen Wert zu kommen.

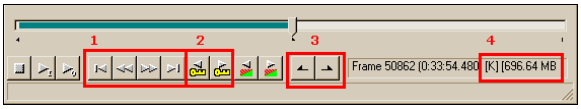

Abbildung 133:
Video-Kontrollbuttons in VirtualDubMod.

Haben wir eine passende Stelle gefunden, setzen wir dort den **Auswahl-Ende-Marker** (bei (3) der rechte Button). Dann springen wir mit dem linken Button bei

(1) zum Anfang des Films und setzen hier den **Auswahl-Anfang-Marker** (bei (3) der linke Button). In der Zeitleiste ist jetzt der vordere Teil des Films markiert.

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

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

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

Splitting mit MKVMerge

Den fertigen Film laden wir in VirtualDubMod und suchen wie gerade beschrieben einen passenden Punkt zum Schneiden heraus. Das Schneiden selbst muss dann MKVMerge übernehmen. Wir merken uns den Zeitpunkt, an dem wir splitten wollen (im Beispiel **00:33:54.480**) und tragen diesen wie in Abbildung 134 in MKVMerge GUI ein.

Abbildung 134: Splitting-Kontrollen in MKVMerge.

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

das das gleiche Format wie in VirtualDubMod.

Alle anderen Einstellungen bleiben wie am Anfang, das heißt besonders, dass wir *nicht* den komplett gemuxten Film splitten, sondern die endgültigen Teildateien aus den einzelnen Streams zusammensetzen. Mit **Start muxing** erzeugen wir die endgültigen Dateien. Damit ist der Film komplett fertig.

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

Splitting mit AVI-Mux GUI

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

Dafür öffnen wir in AVI-Mux GUI über den **Settings**-Button den Einstellungsdialog. Wir benötigen den **General**-Abschnitt der **Output**-Settings. Dort klicken wir unter **Split** den **Advanced**-Button (Abbildung 136). Im folgenden Fenster (Abbildung 135) tragen wir im markierten Feld die Zeitmarke des Splitpunkts ein und fügen sie mit dem Plus-Button zur Liste hinzu. Das Format ist das gleiche wie in VDub und MKVMerge. Wenn wir mehr als zwei CDs benötigen, fügen wir ganz einfach weitere Splitpunkte hinzu. Abschließend bestätigen wir sämtliche Dialoge und starten über den Start-Button das Muxing.

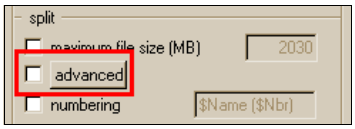

Abbildung 136: Splitting aktivieren in AVI-Mux GUI.

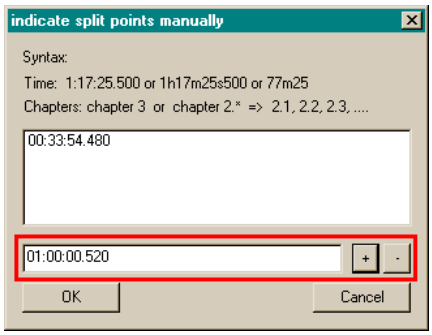

Abbildung 135: Splitpunkte setzen.

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

C.1.2 DivX 5.2.1 VfW-Konfiguration

Gordian Knot hat Probleme mit der 6er-Generation von DivX. Also sehen wir uns auch die Konfiguration von DivX 5.2.1 im Detail an, denn das ist die letzte Version, die garantiert problemlos mit Gordian Knot funktioniert.
DivX.com bietet die alte Version nicht mehr an. Wir können sie aber noch von *http://www.divx-digest.com/software/divxcodec5.html* herunterladen.

Einstellungen für den 1st Pass

Zuerst rufen wir über den **Select Divx Certified Profile**-Button den Profilassistenten auf. Um volle Kontrolle über alle Optionen zu erhalten, müssen wir wie in Abbildung 137 den Haken bei **DivX Certified** entfernen und damit die Profile deaktivieren. Dann klicken wir uns auf die nächste Seite weiter.

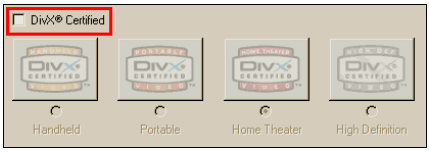

Abbildung 137: DivX-Profilauswahl.

Was die drei Optionen in Abbildung 138 im Detail bedeuten, haben wir schon bei der Xvid-Konfiguration gesehen. Bei DivX arbeiten sie prinzipiell genauso, von einigen Einschränkungen abgesehen. DivX-GMC benutzt nur einen Warppoint (im Gegensatz zu 3 bei Xvid), QPel unterscheidet sich wenig. **Bidirectional encoding** konfiguriert die B-Frames. **Off** dürfte sich selbst erklären ;-), **Adaptive Single Consecutive** entspricht dem verhalten bisheriger DivX-Versionen, d. h. mehrere B-Frames hintereinander werden nicht gesetzt. Maximal zwei aufeinander folgende B-Frames erlaubt **Adaptive Multiple Consecutive**.

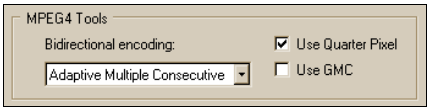

Abbildung 138: B-Frames, Qpel und GMC in DivX.

B-Frames sollten wir auf jeden Fall aktivieren. Wenn ich Doom9 durchstöbere, können das ruhig die **Multiple Consecutive** sein. Und wer B-Frames wirklich gezielt konfigurieren will, muss sowieso auf Xvid umsteigen. QPel würde ich einschalten, GMC nicht. Achtung: Für Standalones sollten beide aus bleiben und maximal **Single Consecutive** B-Frames verwendet werden.

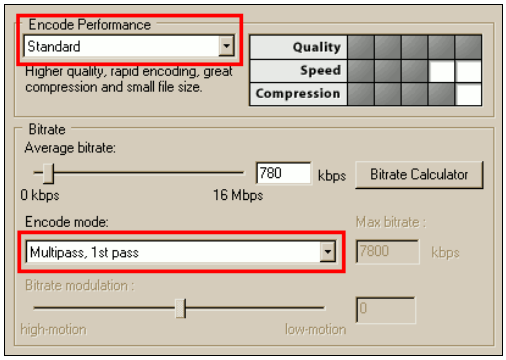

Abbildung 139: Einstellungen auf der General-Seite von DivX.

Damit verlassen wir den Assistenten und kommen wieder auf die **General**-Seite (Abbildung 139). Bei **Encode Performance** müssen wir uns entscheiden, ob Geschwindigkeit oder Qualität wichtiger ist. **Standard** ist vergleichsweise flott und bringt durchschnittliche Qualität. **Slow** bremst deutlich ab, belohnt uns aber auch mit höherer Qualität. **Fast** ist für ein auf Qualität getrimmtes 2-Pass-Encoding weniger interessant. Eine Einschränkung müssen wir im langsamen Modus hinnehmen: **QPel** funktioniert nur mit **Standard**. Wenn wir Quarter Pixel aktivieren, stellt DivX den Modus automatisch um und blendet den Schieberegler ab.

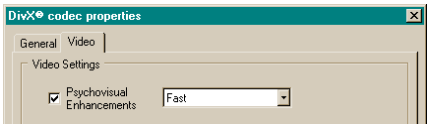

Abbildung 140: Setup für Psychovisual Enhancements.

Multipass, 1st pass ist die passende Einstellung für den ersten Durchgang. Um die Bitrate darunter müssen wir uns nicht kümmern, denn die setzt Gordian Knot automatisch ein.

Und damit weiter zum **Video**-Register. Hier sind erst einmal die **Psychovisual Enhancements** aus Abbildung 140 interessant. Alles andere können wir bei den Standards belassen. PVE versucht die Bildteile zu errechnen, die das menschliche Auge sowieso nicht wahrnehmen kann, und lässt diese weg. Die frei werdende Bitrate kann dann für andere Teile des Bilds verwendet werden.

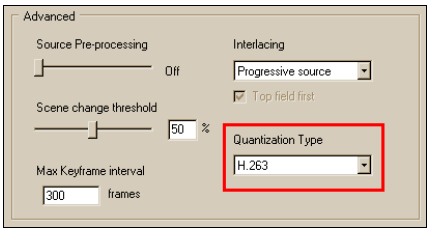

Abbildung 141: Matrixauswahl in Divx.

Der **Fast**-Modus ist aggressiver als der **Slow**-Modus, d. h. er bringt mehr, erhöht aber auch die Gefahr, dass sichtbare Artefakte im Bild auftauchen. Mit **Slow** sind wir da auf der sichereren Seite, allerdings um den Preis niedrigerer Geschwindigkeit und geringerer Effektivität der PVE.

Unter **Quantization Type** lässt sich angeben, welche Quantisierungsmatrix DivX verwenden soll. Details dazu haben wir bei der Xvid-Konfiguration schon gesehen. Da **MPEG-2** in DivX 5.2.1 noch nicht richtig ausgereift war, würde ich bei **H.263** bleiben.

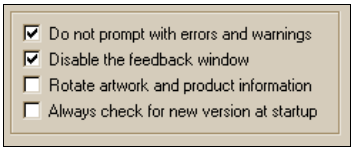

Abbildung 142: Sonstige DivX-Einstellungen.

In den **Settings** setzen wir unten die Haken bei **Do not prompt...** und **Disable the Feedback window**. So überschreibt DivX im 2nd Pass ungefragt die Log-Datei (was er ja tun soll) und schaltet das Statistik-Fenster während des Encodierens ab. **Rotate artwork** und **Check for new version** kann jeder ganz nach seinen Vorlieben setzen. Naja … das Statistik-Fenster eigentlich auch. :-)

Einstellungen für den Nth Pass

Den Großteil aller Einstellungen lassen wir für alle Durchgänge gleich. Nur auf der **General**-Seite ändert sich ab dem 2nd Pass etwas. Hier stellen wir wie in Abbildung 143 **Multipass, nth pass** ein und setzen die Haken bei **Update log file**. Die Aktualisierung der Logdatei ist wichtig, da dadurch erst die weiteren Qualitätsverbesserungen in den Passes 3 bis n ermöglicht werden.

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

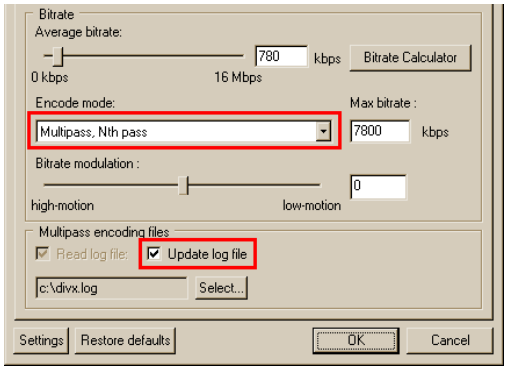

Abbildung 143: DivX-Setup für den Nth Pass.

AR-Flag mit MPEG4 Modifier ändern

Anamorphe Encodings ▸ A.3.2.2 sollten in ihrem MPEG-4-Videostream das AR-Flag auf den richtigen Wert gesetzt haben, damit das Video nicht mit Eierköpfen abgespielt wird. Seit Version 6.5 bietet DivX direkt in der Encoder-Konfiguration eine Option dafür an. Wer mit einem älteren DivX encodiert, kann das AR-Flag nicht gleich beim Encoding setzen, sondern muss das hinterher mit MPEG4 Modifier tun. Dieses Programm unterstützt bisher allerdings nur AVIs. Deshalb müssen wir entweder gleich in Gordian Knot AVI verwenden oder die Datei nach dem Encoding mit VirtualDubMod nach AVI konvertieren. Dazu laden wir den fertigen Film und speichern ihn über **F7** wieder ab. Unten im Speichern-Dialog muss der **Video mode** auf **Direct Stream copy** stehen.

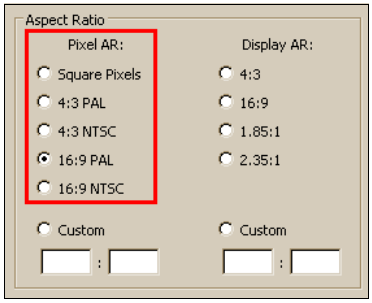

Abbildung 144: AR-Einstellungen von MPEG4 Modifier.

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

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 muss. Woher diese Einschränkung kommt und warum wir es wo immer möglich vermeiden sollten, davon abzuweichen, das betrachten wir in diesem Kapitel genauer.

Grund der Auflösungseinschränkung

Das Bild der DVD verwendet wie das fertige Encoding den Farbraum ▸ A.1.2.1 YV12, der nur mit Auflösungen kompatibel ist, die horizontal glatt durch 4 und vertikal glatt durch 2 teilbar sind. 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.

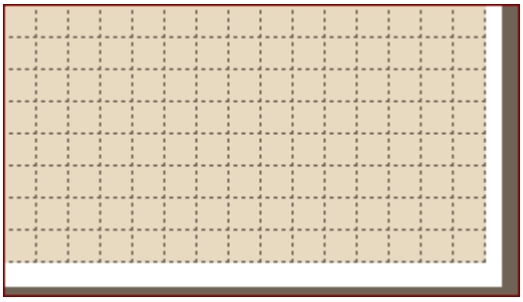

Abbildung 145: Schematisches Nicht-Mod16-Videoframe.

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 Abbildung 145 zu sehen. Dargestellt ist schematisch und in Originalgröße ein Einzelbild mit einer Auflösung von 248 × 140 Pixeln. In der Breite passen so 15 vollständige Makroblocks nebeneinander. Außerdem bleiben am rechten Rand 8 überschüssige Pixel stehen, die keinen vollständigen Block mehr ergeben. In der Höhe sieht die Situation ähnlich aus mit 8 vollständigen Blocks und 12 überschüssigen Pixeln.

Für den weiß eingezeichnete 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 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 damit dem Encoder das Erweitern und Auffüllen des Bildes und verhindern so einen Qualitätsverlust. Schließlich muss bei Nicht-Mod16-Auflösungen intern ein größeres Bild encodiert werden, was unweigerlich einen kleinen Teil der verfügbaren Bitrate für die Codierung unnützer Informationen verbrät.

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 (vgl. das Cropping-Kapitel, Seite 39).

Um uns aus dieser Zwickmühle zu befreien, gibt es zwei mögliche Lösungen. Wir können mit dem AviSynth-Filter *FillMargins* die übrig gebliebenen Balken

übertünchen. Wie das geht, ist in den Frontend-Abschnitten jeweils für StaxRip (ab Seite 147) und für Gordian-Knot (ab Seite 167) erklärt. Allerdings eignet sich die Methode nur für zwei bis drei schwarze Pixelreihen. Deswegen bietet es sich öfter an, doch ins Bild hinein zu schneiden, das verlorene Stück aber auf ein paar unkritische wenige Pixelreihen zu beschränken, indem wir das Mod16-Kriterium auf mod8 oder gar mod4 aufweichen. D. h. wir wählen eine Auflösung, die in einer oder beiden Dimensionen nur noch glatt durch 8 bzw. 4 anstatt durch 16 teilbar ist.

Zwar drückt eine Mod8/Mod4-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 deswegen mod8/mod4 das geringere Übel als überstehende Balken. Wichtig ist dabei, dass wir ganz genau darauf achten, auch wirklich sämtliche Balken kompromisslos und komplett abzuschneiden, damit der Encoder beim internen Auffüllen den Balken nicht wieder herstellt. Außerdem reagieren ASP- und AVC-Encoder unterschiedlich auf nicht-mod16. Dank Makroblockpartitionen ▸ A.4.3 kann AVC die »krumme« Auflösung besser auffangen als ASP. Was das für die Praxis bedeutet, zeigt die folgende Tabelle.

	MPEG-4 ASP (Xvid, DivX)	**MPEG-4 AVC (x264)**
Mod16	Bevorzugen, wo immer möglich. Wenn angebracht, FillMargins einsetzen.	Bevorzugen, aber nicht um jeden Preis. FillMargins vermeiden.
Mod8	Alternative, wenn mod16 nicht möglich. Akzeptabler Effizienzverlust.	Effizienzverlust gering, ähnlich wie mod4.
Mod4	Vermeiden. Effizienzverlust zu groß.	Effizienzverlust gering, ähnlich wie mod8.

C.2.2 Die Bedeutung der ITU-R BT.601 für das PAR

Das Anamorph-Kapitel weiter vorne hat es allein schon in sich, wenn man zum ersten Mal damit in Kontakt kommt. Die Hintergründe und die Diskussion um die verschiedenen Pixel Aspect Ratios dort auch noch anzusprechen, würde den Rahmen deutlich sprengen. Deswegen beschäftigen wir uns erst jetzt damit.

Analog-Digital-Wandlung

Die Empfehlung Nummer BT.601 der ITU-Organisation, die so viel Verwirrung stiftet, hat die Konvertierung analogen Filmmaterials in ein digitales Format zum Thema. Warum sollte uns das eigentlich interessieren? Schließlich haben wir eine DVD – ein vollständig digitales Medium – als Quelle und erzeugen eine genauso vollständig digitale Zieldatei. Trotzdem spielt Analog eine entscheidende Rolle, denn

- nach wie vor wird die Mehrzahl der Filme mit analogen Kameras gefilmt,
- das typische Ausgabemedium einer DVD ist ein analoger Fernseher.

Beginnen wir deshalb ganz vorn bei der Umwandlung von analog nach digital. Etwas vereinfacht dargestellt arbeitet PAL mit 576 analogen Zeilen (Scanlines), NTSC mit 486. Es liegt nahe, Zeilenanzahl mit Pixelanzahl gleichzusetzen, und genau so erfolgt auch die vertikale Analog-Digital-Konvertierung.

Wenn wir von quadratischen Pixeln ausgehen, lässt sich daraus leicht die digitale horizontale Auflösung errechnen, ohne noch einmal auf die analoge Welt zurückzugreifen. Wir wissen, dass die DVD nur 16:9 und 4:3 als Seitenverhältnis zulässt. Außerdem kennen wir die vertikale digitale Auflösung. Entsprechend rechnen wir z. B. für 4:3 PAL:

$$\text{hor. Auflösung} = 576 \times 4/3 = 768 \text{ Pixel.}$$

Für die anderen drei Fälle setzen wir jeweils die passenden Werte ein und erhalten die vier Standardauflösungen aus Tabelle 17.

	PAL	NTSC
4:3	768 × 576	648 × 486
16:9	1024 × 576	864 × 486

Tabelle 17: Rechnerische digitale Standardauflösungen.

Wichtig ist diese Tabelle für die Berechnung der Seitenverhältnisse weiter unten. Zu den Eigenschaften des analogen Videosignals passt sie nicht, denn »analoge Pixel« – wenn man überhaupt von so etwas sprechen kann – sind nicht quadratisch, sondern rechteckig; breiter als hoch, wie in Abbildung 146 zu sehen. Diese Tatsache ist zumindest ein Grund für die verzerrte Auflösung der DVD. Aber das nur am Rande. Wir stehen noch immer vor dem Problem, die »echte« horizontale Auflösung zu ermitteln. Eine natürliche Aufteilung in diskrete Werte wie in der Vertikalen existiert nicht. Eine analoge Scanline ist eine Wellenform von einigen Mikrosekunden Dauer und ohne jede offensichtliche Unterteilung.

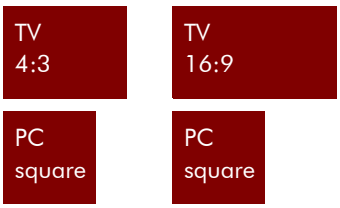

Abbildung 146: Pixelformen.

Um diskrete (digitale) Werte zu erhalten, müssen wir das analoge Signal in bestimmten Zeitabständen abtasten und den ermittelten Wert als Pixel speichern. Das nennt man Sampling, und die Geschwindigkeit der Abtastung heißt Samplingrate. Um ein TV-Bild sinnvoll zu digitalisieren, sind mehrere Millionen Abtastungen pro Sekunde nötig. Für die genauen Werte hilft uns die ITU-Empfehlung weiter.

Aussage der ITU-R BT.601

PAR-Berechnung für PAL

Der Elektronenstrahl eines PAL-Fernseher benötigt 52 μs Zeit, um eine Scanline zu zeichnen. Aus der ITU-R BT.601 ergibt sich eine Samplingrate von 13,5 MHz (Megahertz = Millionen Abtastungen pro Sekunde). Damit erhalten wir 52 μs × 13,5 MHz = 702 horizontale Pixel, was einer aktiven Auflösung von 702 × 576 Pixeln entspricht. *Aktiv* heißt, das komplette Bild, wie es einmal auf dem analogen Film vorhanden war, soll innerhalb dieser 702 Pixel liegen.

Die Auflösung einer PAL-DVD ist auf 720 × 576 Pixel festgelegt. Zwar erlaubt der Standard auch ein paar andere Auflösungen, die praktisch aber nur geringe Bedeutung haben. Der Grund für die abweichende Breite der aktiven ITU-Auflösung ist der Fernseher, dessen Konstruktion zu einem Overscan-Bereich führt.

Das heißt, ein dünner Streifen Bild ragt über den Rand des Bildschirms hinaus. Da der Zuschauer diesen Bereich natürlich nicht sieht, sollte darin auch kein wichtiges Bild enthalten sein.

Grafisch stellt sich die Situation wie im folgenden Bild dar. Die graue Fläche ist die DVD-Auflösung, die dünne rote Linie kennzeichnet das aktive ITU-Bild. An dieser Stelle benötigen wir unsere Tabelle 17 mit den quadratischen Auflösungen, denn Computerpixel sind nie (fast nie, siehe Anmerkung Seite 45) rechteckig. Mit einem 16:9-Bild stehen wir dann vor der Situation, 1024 × 576 quadratische Pixel in 702 × 576 rechteckigen Pixeln unterbringen zu müssen. Vertikal kein Problem, da die Auflösungen exakt übereinstimmen. Horizontal kommen wir allerdings nicht darum herum, die 1024 vorhandenen Pixel in 702 aktive Pixel zu quetschen. Machbar ist das ohne weiteres, lediglich produzieren wir als Nebeneffekt ein verzerrtes (horizontal gestauchtes) Bild. Mit einem 4:3-Bild spielen wir genau das gleiche durch, mit dem einzigen Unterschied, ein weniger verzerrtes Bild zu erhalten.

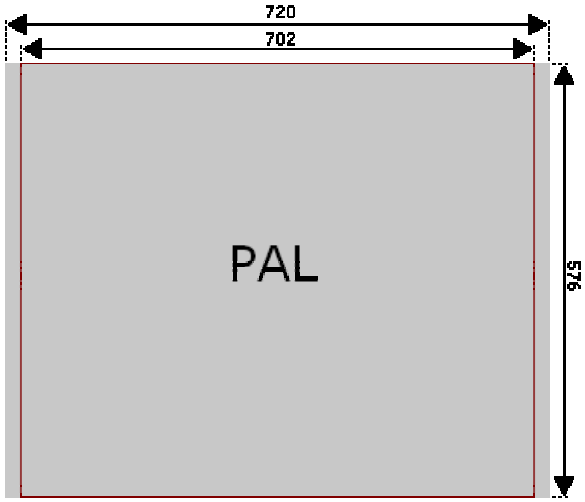

Abbildung 147: Auflösung und aktiver Bereich der PAL-DVD.

Beim Abspielen brauchen wir dann eine Angabe über den Verzerrungsfaktor, um das Bild wieder auf die richtige Breite strecken zu können. Dieser Faktor ist das Pixel Aspect Ratio (PAR). Für 16:9 PAL sagt uns der PAR-Wert von ca. 1,46, dass wir das Bild auf 146 % auseinanderziehen müssen, um ein korrektes Seitenverhältnis zu erhalten.

Das PAR selbst zu errechnen, ist mit dem Wissen über die ITU-Empfehlung einfach. Wir packen dazu die Information »Wie verhält sich die korrekte zur gestauchten Breite« in eine mathematische Schreibweise, was nichts anders als der Quotient aus korrekter und gestauchter Breite ist. Entsprechend ergibt sich für den 4:3- und 16:9-Fall:

$$\textit{4:3-PAR} = \frac{768}{702} = \frac{128}{117} \approx 1{,}0940$$

$$\textit{16:9-PAR} = \frac{1024}{702} = \frac{512}{351} \approx 1{,}4587$$

Um beim Abspielen die richtige Breite zu erhalten, müssen wir die gestauchte Breite mit dem PAR multiplizieren. Aber Achtung! Auch wenn nur 702 Pixel aktiv sind, hat die DVD trotzdem eine Auflösung von 720 Pixeln. Auch die 18 nicht aktiven Pixel müssen gestreckt werden, denn halbe Sachen sind nicht erlaubt. Entweder strecken wir das komplette Bild oder wir lassen es ganz bleiben.

4:3-Breite = 720 × $^{128}/_{117}$ = 787,69 Pixel

16:9-Breite = 720 × $^{512}/_{351}$ = 1050,26 Pixel

Damit sollte auch klar werden, warum nach ITU-Norm das Bild ein wenig breiter als die Standard-DARs 4:3 oder 16:9 ausfällt. Das sind genau die 18 nicht aktiven Pixel, die gestreckt exakt die Abweichung zum Standard-DAR (19,69 bzw. 26,26 Pixel) ergeben. Damit entspricht auch der *aktive Teil* der endgültigen Auflösung genau der Ausgangssituation: 768 × 576 und 1024 × 576.

PAR-Berechnung für NTSC

Die Situation für NTSC stellt sich ähnlich dar. Den quadratischen Auflösungen 648 × 486 und 864 × 486 steht nach ITU ein aktiver Bereich von 710,85 × 486 entgegen. Ja, exakt so. Die krummen horizontalen Pixel müssen für die Praxis natürlich auf 711 aufgerundet werden. Damit sich die Rundungsfehler nicht summieren, rechnen wir trotzdem mit den exakten 710,85.

Genau genommen summiert sich nichts Wesentliches. Als letzter Schritt vor der eigentlichen Darstellung am Bildschirm muss sowieso aufs nächste volle Pixel gerundet werden, was den minimalen rechnerischen Unterschied zwischen 710,85 und 711 ausgleicht.

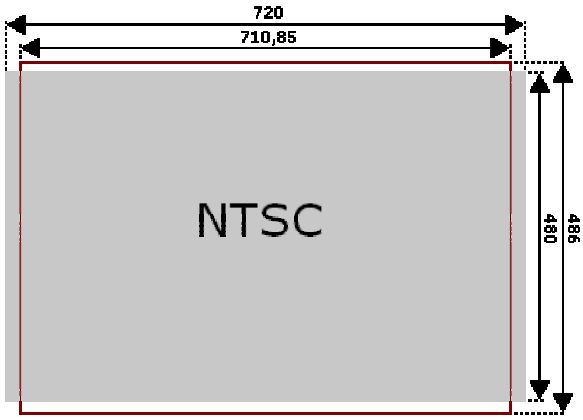

Abbildung 148: Auflösung und aktiver Bereich der NTSC-DVD.

Unschöner sind die 480 vertikalen DVD-Pixel, die für die 486 aktiven Pixel nicht ausreichen. Wir stehen grafisch also vor folgender Situation, die auf den ersten Blick schon spürbar komplizierter als bei PAL aussieht (Abbildung 148). Die graue Fläche stellt wieder die DVD-Auflösung dar, der rote Rahmen den aktiven ITU-Bereich. Prinzipiell wäre es möglich, vertikal genauso zu verfahren wie horizontal, also 486 tatsächliche Pixel in 480 DVD-Pixel zu quetschen und zusätzlich

zum horizontalen mit einem vertikalen PAR zu arbeiten. Das ist genauso lästig wie es klingt, deswegen hat man sich für ein einfacheres Verfahren entschieden. Was nicht passt, wird passend gemacht! Wir schneiden ganz einfach oben und unten je drei Pixel ab und gut ist. Da die Ränder des Bildes schon im rein analogen Umfeld typischerweise Rauschen anstatt aktives Bild enthalten, stören die fehlenden Pixel nicht. Natürlich bringt das Cropping eine minimale Abweichung des DAR mit sich, die aber vernachlässigt werden kann. Die Ungenauigkeit in der Darstellung, die sich allein aus der Technik der Fernsehbildröhre ergibt, ist deutlich größer.

Damit befinden wir uns tatsächlich in einer Situation, die derjenigen für PAL entspricht. Die vertikale Auflösung passt und horizontal stopfen wir die Originalpixel in die aktiven Pixel. Für 4:3 ergibt sich dabei die etwas seltsame Situation, 648 Pixel in 710,85 Pixel »quetschen« zu müssen. Nicht verwirren lassen. Das ändert am Vorgehen überhaupt nichts, lediglich das PAR wird (ganz automatisch) kleiner als 1. Hier sind die Berechnungen, die analog zu PAL funktionieren.

$$\textit{4:3-PAR} = \frac{648}{710{,}85} = \frac{4320}{4739} \approx 0{,}9116$$

$$\textit{16:9-PAR} = \frac{864}{710{,}85} = \frac{5760}{4739} \approx 1{,}2154$$

Beim Abspielen multiplizieren wir wieder die horizontale DVD-Auflösung mit dem PAR.

4:3-Breite = 720 × $^{4320}/_{4739}$ = 656,34 Pixel
16:9-Breite = 720 × $^{5760}/_{4739}$ = 875,12 Pixel

Auch hier entsprechen die gestreckten nicht-aktiven Pixel dem Unterschied zum Standard-DAR – bis auf die minimale Abweichung durch die abgeschnittenen sechs vertikalen Pixel.

Zusammenfassung

Fassen wir zusammen: PAL und NTSC entscheiden sich nicht wesentlich, lediglich hantieren wir bei NTSC mit unschöneren Zahlen. Beide Formate ergeben nach ITU ein etwas breiteres Bild als vom Standard-DAR gewohnt, was an den verbleibenden nicht aktiven Pixeln liegt. Am TV sehen wir trotzdem ungefähr den Standard-DAR, da die zusätzliche Breite im Overscan-Bereich verschwindet. Für die korrekte Wiedergabe müssen wir insgesamt vier PAR-Werte im Kopf behalten, siehe Tabelle 18.

	PAL	NTSC
4:3	128 / 117	4320 / 4739
16:9	512 / 351	5760 / 4739

Tabelle 18: Exaktes PAR nach ITU-R BT.601.

Wem einmal die PARs 72 / 79 für 4:3 und 96 / 79 für 16:9 über den Weg laufen, das sind die Werte für NTSC, gerechnet mit 711 aktiven Pixeln.

Die vereinfachte PAR-Tabelle

Der aufmerksame Leser wundert sich bestimmt schon längst wieder. Die ITU-PAR-Tabelle hier stimmt doch mit der Tabelle aus dem Anamorph-Kapitel nicht überein! Die sieht nämlich so aus:

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

Tabelle 19: Vereinfachtes PAR nach ITU-R BT.601.

So unterschiedlich die Brüche auf den ersten Blick anmuten: wenn man sie ausdividiert, kommt bei beiden Tabellen fast das Gleiche heraus. Die Abweichung ist mit 2 – 3 Pixeln (etwa 0,3 %) so gering, dass sie unsichtbar bleibt. Ob wir deshalb mit dem exakten PAR oder den vereinfachten Werten arbeiten, ist irrelevant.

Die vereinfachte Tabelle 19 entstammt dem MPEG-4-Standard. Im H.264-Teil, den man von der ITU-Homepage kostenlos herunterladen kann, stehen die Zahlen so im Kapitel E.2.1 in Tabelle E-1. Leider ist der Rest des MPEG-4-Standards nicht frei verfügbar, weshalb ich im für Xvid und DivX relevanten Part 2 (MPEG-4 Visual) nicht selbst nachschauen kann. Dass die gleichen Zahlen auch dort genannt werden, darauf lässt mich eine *Äußerung Koepis* im Doom9-Forum schließen.

Bleibt die Frage, warum die Angaben im MPEG-4-Standard überhaupt von den exakten ITU-Werten abweichen. Darüber bin ich mir selbst nicht im klaren. Fakt ist jedenfalls, dass sowohl der 3ivx- als auch der Xvid-Encoder die vereinfachte Tabelle für ihre Standard-PAR-Auswahl verwenden. Zusammen mit den simplen Zah-

len, die man gut im Kopf behalten kann, ist das Grund genug, die einfache Tabelle für die Praxis zu empfehlen.

ITU-PAR oder generisches PAR?

Mit ITU-PAR meine ich im Folgenden immer die exakten und vereinfachten Werte gleichermaßen. Natürlich könnte man zwischen den beiden genauso unterscheiden wie zwischen ITU-PAR und generischem PAR. Wegen der nur minimalen Abweichung halte ich das aber für übertriebene Haarspalterei.

Begriff des Generischen PAR

Dass die ITU-R BT.601 genau die Rechnung anstellt, wie oben beschrieben, liegt an ihrer Konzentration auf die analoge Welt, besonders auf analoge Abspielgeräte mit Overscan-Bereich. Denn um genau diesen Overscan-Bereich dreht sich die ganze Empfehlung.

Der Film einer ITU-DVD ist am TV-Bildschirm vollständig zu sehen. Zwar versteckt der Fernseher die Ränder des Videos im unsichtbaren Overscan-Bereich, jedoch enthalten die sowieso kein aktives Bild, so dass nichts verloren geht.

Das stimmt jedenfalls ungefähr. Die Größe des Overscans ist nicht immer und überall gleich. Also muss ITU mit einem Schätzwert arbeiten.

Geht man von einem digitalen Wiedergabemedium aus – oder allgemeiner von einem Wiedergabemedium ohne Overscan – verliert die ITU-Empfehlung ihren Sinn. Eine nach ITU gemasterte DVD kann dann sogar nachteilig sein, weil sie möglicherweise links und rechts sichtbare dünne schwarze Balken enthält.

Das generische PAR legt genau dieses digitale Szenario zu Grunde. Die Berechnung funktioniert genauso wie weiter oben für ITU beschrieben, nur dass wir uns um einen aktiven Bereich keine Gedanken machen. Aktiv ist die komplette Auflösung der DVD: 720×576 bzw. 720×480. Entsprechend rechnen wir nicht mit 702 oder 710,85 im Nenner, sondern mit 720. Das ergibt schlussendlich folgende PAR-Tabelle:

	PAL	NTSC
4:3	16 / 15	9 / 10
16:9	64 / 45	6 / 5

Tabelle 20: Generisches PAR.

So weit sind sich die Verfechter sämtlicher Lehren auch einig. Aber nur so weit, und kein bisschen weiter.

Diskussion

Die große Frage beim DVD-Backup besteht nun darin, welches PAR wir benutzen sollten. Der Streit darum wird heftig geführt, bleibt nicht immer ganz sachlich und geht meiner Meinung nach meistens am Kern des Problems vorbei. Erst einmal müssen wir zwei Dinge klarstellen.

- Das generische PAR ist nicht auf jeden Fall falsch. Genauso ist das ITU-PAR nicht automatisch in jedem Fall richtig.
- Die Art der Wiedergabe (am Computer, TV, sonstwo) spielt keine Rolle.

Immer unter der Voraussetzung, dass wir das ursprüngliche Seitenverhältnis so exakt wie möglich wieder herstellen wollen, existiert genau ein entscheidender Faktor, der über das richtige PAR entscheidet:

Mit welchem PAR arbeitet die jeweilige DVD?

Eigentlich ist die Überlegung so simpel, dass überhaupt keine Diskussion entstehen sollte. Wenn Person A einen Film horizontal auf die Hälfte staucht und Person B das ursprüngliche Bild wieder herstellen will, dann muss Person B den Film auf 200 % auseinander ziehen. Und wenn Person C einen anderen Film auf drei Viertel seiner Breite quetscht, dann muss Person B diesen Film auf 133 % auseinander ziehen. Im zweiten Fall stur auf 200 % zu beharren, wäre blödsinnig. Genauso verhalten sich aber sowohl die Hardcore-Verfechter des ITU-PAR als auch die Hardcore-Verfechter des generischen PAR. Da wird seitenlang gestritten und am eigentlichen Kern des Problems geradeaus vorbeidiskutiert.

Das richtige PAR ist dasjenige, das im Masteringprozess benutzt wurde.

An dieser Stelle muss eine sinnvolle Diskussion ansetzen, denn so klar die Grundaussage auch ist, die praktische Umsetzung bereitet Probleme. Das PAR ist näm-

lich nirgendwo auf der DVD vermerkt, es existiert also keine Möglichkeit, den richtigen Wert einfach abzulesen. Genauso unmöglich erscheint es bei kommerziellen DVDs, den Masteringtechniker ausfindig zu machen und nachzufragen. Es bleibt also nur die Trial-and-Error-Methode: Wir entzerren das Bild mit den verschiedenen PARs und messen nach, ob Kreise auch wirklich rund erscheinen.

Das ist zu kompliziert und zeitaufwändig für die tägliche Praxis? Stimmt. Möglicherweise können wir aber mit ein wenig Überlegung einer der beiden Tabellen den Vorzug geben. Um das Ergebnis vorweg zu nehmen:

Das ITU-PAR zu wählen, ist die 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 verdächtig genau im Bereich der 2 × 9 nicht aktiven PAL-Pixel liegen.

Gerade neuere DVDs nutzen immer häufiger die komplette DVD-Auflösung aus. Trotzdem erscheint es in der Regel sinnvoller, auch hier das ITU-PAR anzuwenden. Denn dass beim Mastering ITU angewendet und die inaktive Fläche anstatt mit Schwarz mit *zusätzlichem* Bild gefüllt wurde, ist aus einigen Gründen wahrscheinlicher.

- Wichtigstes Wiedergabemedium für die DVD ist der analoge Fernseher, auf dessen Overscan-Eigenschaft die ITU-Norm zugeschnitten ist. Nach generischem PAR geht am TV links und rechts ein Streifen Bild verloren.
- Digitale Wiedergabegeräte ohne Overscan gewinnen an Bedeutung. Mit der Methode, ITU zu verwenden und den inaktiven Bereich mit einem zusätzlichen Stückchen Bild zu füllen, wird man jeder Abspielmethode gerecht.
- Man vermeidet vertikale Balken, deren Übergänge zum Bild viel Bitrate schlucken, da sie kaum jemals direkt auf eine Makroblockgrenze fallen.
- Standalone-DVD-Player verwenden tendenziell das ITU-PAR.
- Professionelle Videosoftware und ganz besonders professionelle Videohardware hält sich strikt an die ITU-Norm. Dass ein Profi mit generischem PAR arbeiten sollte, mutet deshalb etwas seltsam an.
- Die vereinfachte PAR-Tabelle von MPEG-4/Part 2 und H.264 lehnt sich stark an die ITU-Werte an.

Das alles beweist nichts eindeutig und endgültig. Es lässt allerdings die Aussage recht 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 immer zu den ITU-Werten.

Lohnt sich der ganze Glaubenskrieg eigentlich wirklich? Das tut er dann, wenn zwischen ITU-PAR und generischem PAR ein deutlich sichtbarer Unterschied besteht. Die Abweichung zu den exakten ITU-Werten berechnet sich im 16:9-Fall wie unten. Mit den 4:3-Werten würden wir genau die gleichen Ergebnisse erhalten.

PAL-Abweichung = $^{512}/_{351}$ / $^{64}/_{45}$ = 2,56 %

NTSC-Abweichung = $^{5760}/_{4739}$ / $^{6}/_{5}$ = 1,29 %

Für klassische Encodings mit quadratischen Pixeln haben wir einen Aspect Error bis 2,5 % als unproblematisch definiert. NTSC liegt also klar in dem Bereich, an den wir beim klassischen Encoding keinen weiteren Gedanken verschwenden würden. PAL testet die Grenze zwar an, aber um das Kapitel A.3.3 über die nichtanamorphe Zielauflösung zu zitieren: »Geringe Fehler – etwa bis 2,5 % – sind unproblematisch, weil die Verzerrung zu klein bleibt, um spürbar zu werden.«

Aha. Ab in etwa 2,5 % können wir also langsam damit rechnen, dass die Abweichung sichtbar werden könnte. Nach Katastrophenfall klingt das keineswegs. Auch sind die 2,5 % vorsichtig angesetzt. Gordian Knot z. B. warnt erst bei über 3,5 % vor einer kritischen Abweichung.

Die komplette PAR-Diskussion hat also schon ihre Berechtigung. Nur geht in Grabenkämpfen und zwischen technischen Details oft die Perspektive verloren. Wir streiten uns um Abweichungen, für die wir an anderer Stelle seit Jahren nur ein Schulterzucken übrig haben. So richtig will es mir nicht einleuchten, warum ein Unterschied, den wir bisher nicht sehen konnten, plötzlich so durchschlagend wichtig sein sollte.

Teil D
Anhang

Literatur

Artikel und Bücher

Advanced video coding for generic audiovisual services. ITU Recommendation ITU-T H.264. Online verfügbar unter http://www.itu.int/rec/T-REC-H.264

Aho, Jukka: *A Quick Guide to Digital Video Resolution and Aspect Ratio Conversions.* Online verfügbar unter http://www.iki.fi/znark/video/conversion

Aho, Jukka: *The 625/50 PAL Video Signal and TV Compatible Graphics Modes.* Online verfügbar unter http://www.iki.fi/znark/video/modes

BeSweet-Referenz. Gleitz Wiki-Artikel. Online verfügbar unter http://gleitz.info/wiki/index.php/BeSweet-Referenz

Crusty: *The Unofficial XviD FAQ*. Online verfügbar unter http://ronald.vslcatena.nl/docs/xvidfaq.html

Das Video-Kompressionsbuch. Panasonic. Online verfügbar unter http://panasonic-broadcast.de/index.cfm?uuid=356BA7F6C09F11269B3BB10E84B5C76A

Doom9: *Codec shoot-out 2005 – Final.* Online verfügbar unter http://www.doom9.org/codecs-final-105-1.htm

Fiedler, Martin: *Videokompressionsverfahren von MPEG-1 bis DivX, VC-1 und H.264.* Online verfügbar unter http://keyj.s2000.ws/files/projects/videocomp.pdf

LAME. Hydrogenaudio Wiki-Artikel. Online verfügbar unter http://wiki.hydrogenaudio.org/index.php?title=LAME

Mares, Sebastian: *Public, Multiformat Listening Test @ 128 kbps.* Online verfügbar unter http://www.maresweb.de/listening-tests/mf-128-1

Oberle, Daniel: *Das Breitbild-Phänomen.* Online verfügbar unter http://www.aifb.uni-karlsruhe.de/WBS/dob/docs/Breitbild.pdf

Pillai, Latha: *Quantization.* Xilinx XAPP615 (v1.1) June 25, 2003. Online verfügbar unter http://www.xilinx.com/bvdocs/appnotes/xapp615.pdf

Richardson, Iain E. G.: *H.264 and MPEG-4 Video Compression.* Wiley, 2003. ISBN 0-470-84837-5

SCHARFIS_BRAIN: *Exotisches Interlacing.* Online verfügbar unter http://home.arcor.de/scharfis_brain/ExotischesInterlacing

SELUR: *man x264.* Online verfügbar unter http://www.flaskmpeg.info/board/thread.php?threadid=5571

SELUR: *Wissenswertes rund um x264.* Online verfügbar unter http://www.flaskmpeg.info/board/thread.php?postid=54931

SELUR: *Wissenswertes rund um Xvid.* Online verfügbar unter http://www.flaskmpeg.info/board/thread.php?postid=50836

Studio Encoding Parameters of Digital Television for Standard 4:3 and Widescreen 16:9 Aspect Ratios. ITU Recommendation ITU-R BT.601-5. Online verfügbar unter http://www.itu.int/rec/R-REC-BT.601

TAYLOR, JIM: *DVD Frequently Asked Questions (and Answers).* Online verfügbar unter http://www.dvddemystified.com/dvdfaq.html

The Official DivX 5.2 Guide. DivX Inc. Deutsche Version online verfügbar unter http://download.divx.com/divx/DivXUserGuide521-de.exe

WICKENBURG, SEBASTIAN ET AL.: *Die JPEG-Kompression.* Online verfügbar unter http://www.mathematik.de/spudema/spudema_beitraege/beitraege/rooch/neinleit.html

Forum-Threads

Gleitz.info: *agkp – akapuma's GKnot Personalisierer.* http://forum.gleitz.info/showthread.php?t=22860

Gleitz.info: *ITU-R BT.601 und das PAR: Ist mein Wissen korrekt?* http://forum.gleitz.info/showthread.php?t=28145

Doom9.org: *Audio FAQ.* http://forum.doom9.org/showthread.php?t=68300

Doom9.org: *CQMs Compressibility TEST* http://forum.doom9.org/showthread.php?p=649221#post649221

Doom9.org: *Custom Matrices* (AVC) http://forum.doom9.org/showthread.php?t=117041

Doom9.org: *EQM AVC Series*
http://forum.doom9.org/showthread.php?t=96298

Doom9.org: *GPAC MP4Box Information, Requests and Discussions.*
http://forum.doom9.org/showthread.php?t=94874

Doom9.org: *How To Use Mpeg4 AVC Deblocking Effectively*
http://forum.doom9.org/showthread.php?t=109747

Doom9.org: *Human color sensitivity test.*
http://forum.doom9.org/showthread.php?t=72803

Doom9.org: *ITU-R BT.601 and PAR: How good is my knowledge?*
http://forum.doom9.org/showthread.php?t=111102

Doom9.org: *MeGUI Custom x264/AVC video profiles.*
http://forum.doom9.org/showthread.php?t=101813

Doom9.org: *New Quantization Matrices – EQM V3 series.*
http://forum.doom9.org/showthread.php?t=83125

Doom9.org: *New x264 feature: mixed references.*
http://forum.doom9.org/showthread.php?t=101056

Doom9.org: *x264 win32 daily builds*
http://forum.doom9.org/showthread.php?t=89979

Doom9.org: *Xvid_encraw – Patched with AviSynth input support.*
http://forum.doom9.org/showthread.php?t=98469

Doom9.org: *XviD presets thread*
http://forum.doom9.org/showthread.php?t=119399

Weitere nützliche Links

What is Deinterlacing? Facts, solutions, examples.
http://100fps.com

BBC - A Guide to Picture Size
http://www.bbc.co.uk/commissioning/tvbranding/picturesize.shtml

Homepage der Moving Picture Experts Group
http://www.chiariglione.org/mpeg

Jawor's Quantization Matrices
http://www-users.mat.uni.torun.pl/~jawor/matrix/

GSpot: Analysetool für Filmdateien
http://www.headbands.com/gspot

Mediainfo: Analysetool für Filmdateien
http://mediainfo.sourceforge.net

Videoplayer
Media Player Classic: http://sourceforge.net/projects/guliverkli
Zoom Player: http://www.inmatrix.com
VLC Media Player: http://www.videolan.org
The Core Media Player: http://www.corecoded.com

Haali Media Splitter mit Unterstüztung für Matroska, MP4, AVI, OggMedia
http://haali.cs.msu.ru/mkv

ffdshow: DirectShow-Decoder für viele Audio- und Videoformate
http://www.ffdshow.info
Bob0rs Builds: http://x264.nl

Alexander Vigovskys AC3-Decoder
http://ac3filter.sourceforge.net

RareWares: Downloadseite für Audiosoftware
http://www.rarewares.org

Abkürzungen

AAC
Advanced Audio Coding

ABR
Average Bitrate

AC3
Audio Codec 3

agkp
Akapumas Gordian-Knot-Personalisierer

AR
Aspect Ratio

ASF
Advanced Streaming Format

ASP
Advanced Simple Profile

AVC
Advanced Video Coding

AVI
Audio/Video Interleave

AVS
AviSynth Script

BPF
Bits/Pixel*Frame

C
Center Channel

CABAC
Context-Adaptive Binary Arithmetic Coding

CBR
Constant Bitrate

CD
Compact Disc

CSS
Content Scrambling System

DAR
Display Aspect Ratio

DCT
Discrete Cosine Transformation

DNR
Dynamic Noise Reduction

DPL
Dolby Pro Logic

DRC
Dynamic Range Compression

DTS
Digital Theater Systems

DVD
Ohne konkrete Bedeutung

DVD-5
Single-Layer-DVD mit 4,37 GB Kapazität

DVD-9
Dual-Layer-DVD mit 7,95 GB Kapazität

FL
Front Left Channel

FR
Front Right Channel

GKnot
Gordian Knot

GMC
Global Motion Compensation

GUI
Graphical User Interface

HE
High Efficiency

iDCT
Inverse Discrete Cosine Transformation

JPEG
Joint Photographic Experts Group

LAME
LAME Ain't an MP3 Encoder

LC
Low Complexity

LFE
Low Frequency Effects Channel

LPCM
Linear Pulse Code Modulation

ME
Motion Estimation

MKV
Matroska Video

mmg
mkvmerge GUI

MP2
MPEG-1 Audio Layer 2

MP3
MPEG-1 Audio Layer 3

MPEG
Moving Picture Expert Group

NTSC
National Television System Committee

OGM
OggMedia

OTA
Overall Track Adjustment

PAL
Phase Alternation Line

PAR
Pixel Aspect Ratio

PCM
Pulse Code Modulation

PGC
Program Chain

PN
Private Nachricht

PSNR
Peak Signal to Noise Ratio

PVE
Psychovisual Enhancements

QPel
Quarter Pixel

RDO
Rate Distortion Optimization

SBC
Smart Bitrate Coding

SBR
Spectral Band Replication

SL
Surround Left Channel

SMP
Simultaneous Multi Processing

SR
Surround Right Channel

SSIM
Structural Similarity Index

SSRC
Shibatch Sample Rate Converter

SVCD
Super VideoCD

VBR
Variable Bitrate

VCD
VideoCD

VfW
Video for Windows

VHQ
Very High Quality

VOB
Video Object

VTS
Video Title Set

Changelog

Hier stehen lediglich die Änderungen zur jeweiligen Vorgängerversion. Das vollständige Changelog seit Adam und Eva könnt ihr online einsehen: *http://encodingwissen.brother-john.net/changelog.html*

10. Februar 2007 (rev 93)

- Zusätzliches Encoding-Frontend: StaxRip, das es auch gleich zum empfohlenen Frontend geschafft hat.
- Abschnitt zum BPF-Verhältnis gekürzt und umgebaut. Dafür, wie ungenau BPF eigentlich ist, war die Erklärung viel zu ausführlich.
- Die PDF-Version ist auch wieder aktuell.

28. Januar 2007 (rev 77)

- DivX-Update: Version 6.5 unterstützt das Setzen des PAR im Videostream.
- Kurzes Spezialkapitel zu Mod16-Regel & Co.
- Im Kapitel »Anamorphes MPEG-4« Link zum DirectShow-AR-Flag-Test.
- Keine neue PDF! Die muss warten, bis der StaxRip-Abschnitt fertig ist.

Lizenz