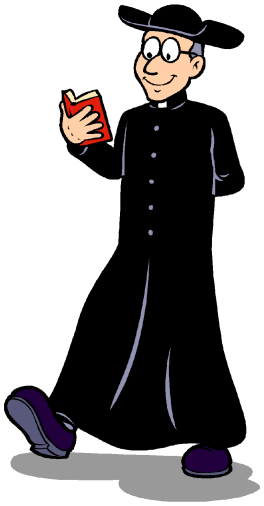
Brother Johns gesammeltes Encodingwissen

Version 24. Oktober 2005



http://encodingwissen/brother-john.net

Erreichen könnt ihr mich per PN in den Doom9-Foren
oder über die Mailadresse auf der Homepage.

Inhalt

Abkürzungen

AAC ..... Advanced Audio Coding
ABR ..... Average Bitrate
AC3 ..... Audio Codec 3
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 Theatre Sound
DVD ..... ohne Bedeutung laut DVD FAQ.
www.dvddemystified.com/dvdfaq.html#1.1
FL ..... Front Left Channel
FR ..... Front Right Channel
GKnot ..... Gordian Knot
GMC ..... Global Motion Compensation
GUI ..... Graphical User Interface
HE ..... High Efficiency
iDCT ..... Inverse Discrete Cosine Transformation
JPEG ..... Joint Photographic Experts Group
LAME ..... LAME Ain't an MP3 Encoder
LC ..... Low Complexity
LFE ..... Low Frequency Effects Channel
LPCM ..... Linear Pulse Code Modulation
MKV ..... Matroska Video
mmg ..... mkvmerge GUI
MP2 ..... MPEG-1 Audio Layer 2

MP3 ..... MPEG-1 Audio Layer 3
MPEG ..... Moving Picture Expert Group
NTSC ..... National Television System Committee
OGM ..... OggMedia
OTA ..... Overall Track Adjustment
PAL ..... Phase Alternation Line
PAR ..... Pixel Aspect Ratio
PCM ..... Pulse Code Modulation
PGC ..... Program Chain
PVE ..... Psychovisual Enhancements
QPel ..... Quarter Pixel
SBC ..... Smart Bitrate Coding
SBR ..... Spectral Band Replication
SL ..... Surround Left Channel
SR ..... Surround Right Channel
SSRC ..... Shibatch Sample Rate Converter
SVCD ..... Super VideoCD
VBR ..... Variable Bitrate
VCD ..... VideoCD
VfW ..... Video for Windows
VHQ ..... Very High Quality
VOB ..... Video Object
VTS ..... Video Title Set

Vorwort

Das Encodingwissen ist genau die richtige Adresse, wenn du deine geliebten DVDs sichern willst und dir dafür XviD, DivX oder x264 vorschwebt. Die Wurzeln liegen in einer einfachen Textdatei, in der ich ab Herbst 2002 wild Notizen gemacht habe. Inzwischen ist das Ganze – wie man sieht – etwas gewachsen und erfreut sich im Doom9/Gleitz-Forum einiger Beliebtheit.
Der Grund dafür mag darin liegen, dass das Encodingwissen keine typische Schritt-für-Schritt-Anleitung ist, sondern sich stark auf Hintergrundinfos konzentriert. Das bedeutet Lesestoff für einige Stunden, dicht gepackte Infos und rauchende Hirnwindungen. Und es bedeutet hinterher das gute Gefühl, besser durchzublicken.

Im Teil 1 betrachten wir zuerst die theoretische Seite. Warum ist Kompression überhaupt nötig? Wie funktioniert sie prinzipiell für Video- und Audiodaten? Abgerundet wird der Grundlagenteil mit einem kurzen Abriss über das DVD-Format.
In Teil 2 bis 4 beschäftigen wir uns dann mit dem praktischen Vorgehen. Was gibt es zu tun, um aus einer DVD einen vollständigen MPEG-4-Film zu bauen? Diese Teile sind nach dem Muster einer Schritt-für-Schritt-Anleitung aufgebaut und zusätzlich dick mit Hintergrundinformationen gepolstert.
Teil 5 schließlich beschäftigt sich mit speziellen Themen, die eher für fortgeschrittene Benutzer gedacht sind.

Dann bleibt nur noch, viel Spaß beim Lesen zu wünschen. Steigen wir ein in die Grundlagen!

Brother John

Teil 1
Grundlagen

1.1 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.

Video

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).

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.

Audio

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 × 6000 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.

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.

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.

Macroblocks

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 Macroblocks, die typischerweise 16 × 16 Pixel groß sind. Je mehr Bitrate (Speicherplatz) wir unserem Film gönnen, desto mehr bleibt pro Macroblock 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 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 Anzahl von mehr oder weniger stark komprimierten Macroblocks, 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 Macroblock nicht exakt dem Original. Ob der Unterschied sichtbar ist – d. h. wie gut sich das Ergebnis ans Original annähert – hängt im Wesentlichen von der Höhe des Quantizers und vom Design der Quantisierungsmatrix ab. Weitere Einflüsse von außerhalb der eigentlichen DCT kommen natürlich dazu: verfügbare Bitrate, andere Codeceinstellungen, Art des Films usw.
Betrachten wir zum Schluss das Ergebnis einer DCT-Codierung an einem 8 × 8 Pixel großen Block. Ganz links in Abb. 1.1 sehen wir stark vergrößert den ursprünglichen Block, rechts daneben die per JPEG komprimierte Version. Da JPEG auch auf der DCT

aufbaut, kommt das Ergebnis einer MPEG-4-ASP-Kompression recht nahe. Besonders in der linken unteren Ecke sind deutlich die Unterschiede zum Original zu erkennen, allerdings nur in der Vergrößerung. Die Blocks in Originalgröße (rechts) sehen sich schon zum verwechseln ähnlich.

Abbildung 1.1

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

1.2.2 Interframe-Kompression

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

I-Frames

Codieren wir ein Bild rein mit den Methoden aus dem letzten Abschnitt, handelt es sich um ein *Intraframe* (kurz I-Frame, auch Keyframe genannt), ein vollständiges Einzelbild. Wie ein JPEG-Foto kann ein Intraframe für sich alleine existieren, d. h. um es zu decodieren, werden keine Informationen aus anderen Bildern benötigt. Man kann also sagen, dass für I-Frames keine Interframe-Kompression stattfindet.
Entsprechend belegen sie von allen Frametypen am meisten Platz, enthalten aber auch das qualitativ hochwertigste Bild. Außerdem sind I-Frames zum Spulen und Schneiden des Films wichtig. Dazu aber mehr im Praxisteil des Encodingwissens.

P-Frames

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

Gäbe es nichts anderes als I-Frames, müssten wir in jedem einzelnen Bild immer wieder alle 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 Macroblocks 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 an einer kurzen Bildsequenz aus einem I-Frame und drei P-Frames (Abb. 1.2).

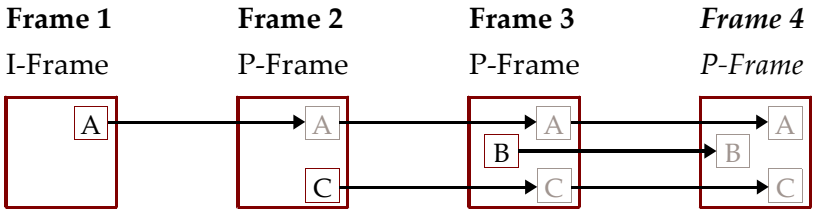


Abbildung 1.2

Wir laden den Film und möchten sofort Frame 4 am Bildschirm anzeigen. Kein Problem: Ein paar neue Macroblocks sind dort sowieso gespeichert und werden direkt aus Frame 4 decodiert. Macroblock A ist nur gewandert. Dessen Daten stehen also in Frame 3. Denkste! Nummer 3 ist auch ein P-Frame und enthält den lapidaren Hinweis: »Macroblock A ist unverändert aus Frame 2 übernommen«. Auch Frame 2 enthält einen analogen Hinweis, so dass wir den Macroblock A schließlich in Frame 1 finden. Beim anschließenden Macroblock B haben wir mehr Glück. Der findet sich schon in Frame 3. Macroblock 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 angekommen ist. Das Splitting-Kapitel 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 Abb. 1.3 wird das deutlicher.

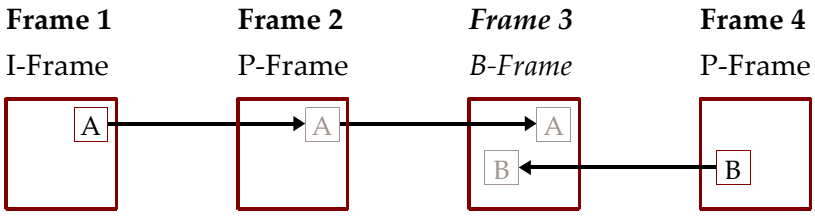


Abbildung 1.3

Wir wollen direkt Das B-Frame 3 decodieren und finden für Macroblock A den Eintrag: »Aus Frame 2 übernommen«. Das kennen wir. So machen das normale P-Frames auch. Der Eintrag für Macroblock 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.

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.

1.4 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*, 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, dürfte MPEG-2 seinem Vorgänger MPEG-1 kaum nachstehen. 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 auch als Advanced Video Coding (AVC) oder H.264 bekannt. 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 eine DVD meistens ohne extreme Verluste auf eine bis zwei CDs schrumpfen.

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 allerdings 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, 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, DivX und 3ivx drei unabhängige und verschiedene Codecs sind, die aber alle Videos nach dem MPEG-4 Advanced Simple Profile erzeugen. Deshalb kann zumindest theoretisch XviD ein mit 3ivx erzeugtes Video abspielen oder 3ivx ein mit DivX erzeugtes usw. Dass das in der Praxis doch nicht so 100-prozentig funktioniert liegt daran, dass keiner der Codecs den Standard absolut vollständig und korrekt implementiert.

Audioformate

Dolby Digital

Das wichtigste Audioformat der DVD wird von der Firma Dolby entwickelt, die den dazu passenden Codec als *Audio Codec 3* bezeichnet. Daher stammt die allgemein bekannte Abkürzung AC3. 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 5 vollständigen Kanälen, der sechste enthält die zusätzlichen Bassef-

fekte, 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. 5 vollständige Kanäle (vorne links und rechts, hinten links und rechts, vorne Mitte) und ein zusätzlicher Basskanal (LFE: Low Frequency Effects).

Digital Theatre Sound

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 180 – 220 kbit/s. Das oft gehörte »128 Kilobit sind CD-Qualität« ist ein Märchen. Filmtonspuren geben sich mit etwas weniger zufrieden (ca. 130 – 160 kbit/s).

  Es existiert eine ganze Reihe von Encodern sehr unterschiedlicher Qualität. Als am höchsten entwickelt gilt der Open-Source-Encoder LAME.
- **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 liegen unter denen von MP3, für Musik bei etwa 140 kbit/s. 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. Das Open-

Source-Lager beherrscht FAAC.

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. Hier bewegt sich Vorbis in der Region um 140 – 160 kbit/s.

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.

1.5 Grundlagen des DVD-Formats

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

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.
In die Tiefen der AR-Thematik einzusteigen, würde den Guide absolut sprengen. Den praktischen Umgang damit beleuchtet Kapitel 5.1 etwas näher. Zur Theorie dahinter nur soviel: Das menschliche Auge kann vertikal mehr Details erkennen als horizontal. Durch die Verzerrung lässt sich auf Kosten der Horizontalen in der wichtigen vertikalen Richtung eine höhere (detailreichere) Auflösung speichern. Insgesamt kommt dadurch beim Auge mehr Qualität an.
Dann existieren noch zwei Verzerrungsarten: anamorph für Widescreen-Filme (die mit den schwarzen Balken oben und unten) und non-anamorph für Vollbild-Filme. Grob gesagt liegt der Unterschied darin, dass Widescreen-Filme mit höherer Verzerrung gespeichert sind. Welche Art der Verzerrung vorliegt, wird später beim Encoding wichtig, wenn wir das Bild aufs richtige Seitenverhältnis umrechnen müssen.

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

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 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 *www.dvd-tipps-ticks.de* massig Informationen.

Teil 2

Vorarbeiten

2.1 Liste der benötigten Software

Gordian Knot RipPack und CodecPack

Bevor wir eine DVD auseinander nehmen können, brauchen wir natürlich die passende Softwareausstattung. Kein Problem, denn im Gordian Knot RipPack ist fast sämtliche nötige Software zusammengefasst. Die Video-Codecs finden sich im CodecPack. Natürlich hindert uns nichts daran, die nötigen Programme auch einzeln zusammenzusuchen. Einige der Tools sind nicht im RipPack bzw. CodecPack enthalten. Die müssen wir uns extra besorgen.

Links zur Software

- Gordian Knot
  Die Kommandozentrale fürs Encoding. Fürs x264-Encoding ist eine Version der 0.34-Serie oder besser notwendig.
  http://sourceforge.net/projects/gordianknot

- vStrip (nicht im RipPack enthalten)
  DVD-Ripper. Kopiert die DVD auf die Festplatte.
  http://www.free-codecs.com/download/vStrip.htm

- Chapter-X-tractor (nicht im RipPack enthalten)
  Schreibt Kapitelinformationen der DVD in eine Textdatei.
  http://www.divx-digest.com/software/chapterxtractor.html

- DGIndex
  Decoder für die VOB-Dateien der DVD. Ehemals DVD2AVI, jetzt Teil des DGMPGDec-Pakets.
  http://neuron2.net/fixd2v/decodefix.html

- BeSweet (bsn.dll nicht im RipPack enthalten)
  Audio-Transcoding-Tool. BeSweet-Beta-Versionen enthalten oft nicht alle nötigen Dateien. Deshalb auch die letzte stabile Version laden und zuerst installieren.
  Wer AAC-Audio verwenden will, muss BeSweet von der Homepage laden, denn die Version im RipPack enthält die für AAC nötige bsn.dll nicht.
  http://besweet.notrace.dk

- BeLight (nicht im RipPack enthalten)
  Grafische Oberfläche für BeSweet. Enthält die nötigen Dateien für Dimzons AAC-Plugins.
  http://corecodec.org/projects/belight

- VobSub
  Tool für VobSub-Untertitel.
  http://www.afterdawn.com/software/video_software/subtitle_tools/vobsub.cfm

- SubRip (nicht im RipPack enthalten)
  Tool für die Erstellung dynamischer Untertitel.
  http://zuggy.wz.cz

- AviSynth
  Frameserver für das Video.
  http://www.avisynth.org

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

- VirtualDubMod
  Video-Editing-Tool. Damit geschieht das eigentliche Encoding.
  http://sourceforge.net/projects/virtualdubmod

- mkvtoolnix (nicht im RipPack enthalten)
  Toolpaket rund um Matroska. Nötig für AAC in MKV.
  http://www.bunkus.org/videotools/mkvtoolnix/index.html

- XviD Video-Codec
  Binaries von Koepi. Den Quellcode gibt's auf xvid.org.
  http://koepi.org

- DivX Video-Codec
  Seit der Version 5.2 gibt es DivX mit vollem Funktionsumfang nur noch als 6-monatige Trial. Neue Versionen erscheinen allerdings in der Regel oft genug, um DivX trotzdem uneingeschränkt kostenlos benutzen zu können.
  http://www.divx.com/divx

- x264 Video-Encoder (nicht im CodecPack enthalten)
  Wir benötigen das gerade aktuelle »x264 Lite package«, genauer gesagt die VfW-Version aus diesem Archiv. Die Installation funktioniert ganz einfach: Archiv irgendwo hin entpacken, die *x264vfw.inf* rechtsklicken und **`Installieren`** wählen. Der beim Entpacken erstellte Ordner kann dann wieder gelöscht werden.
  http://forum.doom9.org/showthread.php?s=&threadid=89979

2.2 Ripping der DVD auf die Festplatte

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! vor kurzem 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 juristischen Probleme einzuhandeln, werde ich die nicht beantworten.

Wichtige Grundlagen

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

- 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 **`02`** haben, das wären dann **`vts_02_0.vob`**, **`vts_02_1.vob`** und **`vts_02_2.vob`**. Dazu gehört die passende **`vts_02_0.ifo`**.
Mehr brauchen wir nicht wissen, um immer die richtigen Dateien auszuwählen.

2.2.1 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 auszuwählen. Da wir den kompletten Hauptfilm rippen wollen, geht das recht einfach.

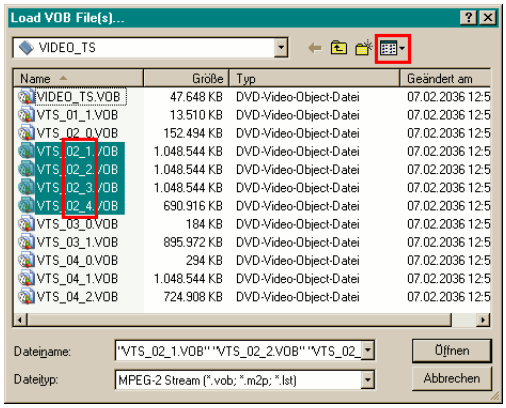


Abbildung 2.1

Im Öffnen-Dialog 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. In Abb. 2.1 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.

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

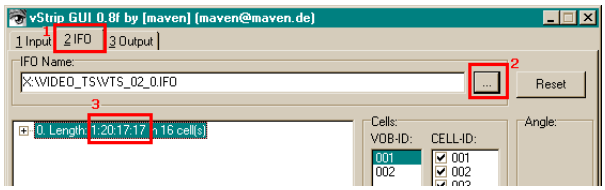


Abbildung 2.2

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

Abbildung 2.3

Unten im Fenster 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 in Abb. 2.3 also **`0x80`** bzw. **`0x81`**. Damit geht es weiter ins **`Output`**-Register.

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 wie in Abb. 2.4 **vts_02.vob**. Das fehlende **_y** fügt vStrip beim Ripping automatisch an.

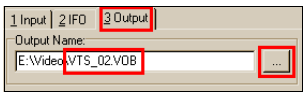


Abbildung 2.4

Abbildung 2.5

Die **Output Options** in der Mitte sollten immer ohne Haken bleiben, und weiter unten bei **Split File Size** 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.

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 (siehe Abb. 2.3 auf Seite 24).

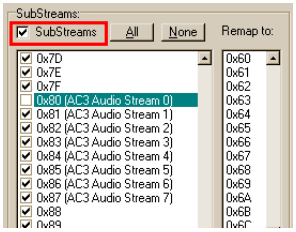


Abbildung 2.6

In Abb. 2.6 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.

2.2.2 Ripping der Kapitelinfos

Leider kann vStrip die Kapitelinfos 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**.
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.

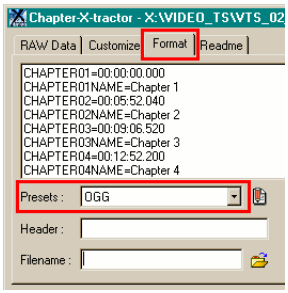


Abbildung 2.7

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

2.3 Decodieren der VOBs mit DGIndex

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.

Wir starten DGIndex und gelangen über **F2** in den Öffnen-Dialog (**F3** in Versionen älter als 1.1.0). Dort wählen wir die vorhin gerippten VOBs aus. Beim Klick auf **Öff-**

nen erscheint ein weiteres Fenster, in dem wir noch zusätzliche VOBs laden könnten. Einfach mit **Ok** bestätigen.

Die Einstellungen im **Video**-Menü sollten wie in Abb. 2.8 aussehen. Den **iDCT Algorithm** wählt DGIndex je nach Prozessor automatisch.

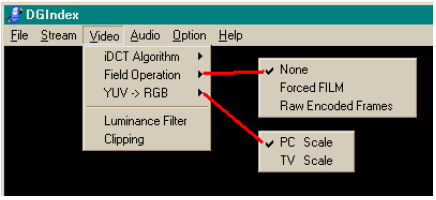


Abbildung 2.8

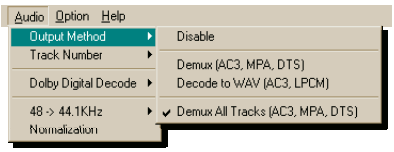


Abbildung 2.9

Im Audio-Menü (Abb. 2.9) regeln wir das Herausziehen der Audiospuren aus den VOBs. **Output Method** bestimmt, auf welche Weise die Audiospuren extrahiert werden. **Disable** extrahiert gar nichts, **Demux** extrahiert nur die unter **Track Number** ausgewählte Spur und **Demux all Tracks** extrahiert alle vorhandenen Spuren. Mit **Decode to Wav** könnten wir den Sound gleich in Wave umwandeln. Das sollten wir aber besser BeSweet überlassen. Deshalb interessiert uns auch der Rest des Audio-Menüs nicht.

Über **F4** legen wir nun eine d2v-Projektdatei an. Solange DGIndex arbeitet, öffnet sich das Statusfenster (Abb. 2.10). Sobald dort ganz unten **FINISH** erscheint, ist der Speichervorgang abgeschlossen und wir haben die extrahierten Audiodateien und die d2v-Datei auf der Platte.

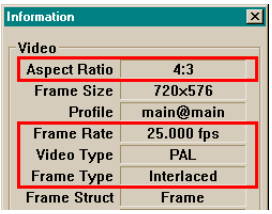


Abbildung 2.10

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 in Abb. 2.11 unten auftreten. Manchmal wird der Effekt auch nur bei Szenenwechseln sichtbar, wenn sich altes und neues Bild ein Frame lang überlagern.

Abbildung 2.11

Hat das Video solche Effekte, müssen wir es später in Gordian Knot 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.

2.4 Transcoding des Sounds

Die Audioformate

Für die Audiospuren gibt es viele Möglichkeiten; die einfache alte Formel »MP3 oder bei genügend Platz AC3« gilt nicht mehr. Deshalb sehen wir uns erst einmal die verschiedenen Audioformate an.

MP3

Der Klassiker. Mit MP3 brauchen wir uns um Abspielprobleme keine Sorgen zu machen: läuft überall. Der Platzbedarf liegt mit etwa 140 – 150 kbit/s im Rahmen, Raten unter 128 kbit/s sollten wir vermeiden. Größter Nachteil ist die Beschränkung auf Stereo. Den 6-Kanal-Ton der DVD können wir mit MP3 nicht beibehalten.

AC3

Der Original-Ton von der DVD. Behalten wir ihn, entfällt die Konvertierung. Ohne die haben wir auch keinerlei Qualitätsverlust, und natürlich bleibt der Mehrkanal-Ton erhalten. Großer Nachteil: AC3 ist mit 384 kbit/s oder 448 kbit/s verdammt groß, so dass wir ihn für 1-CD-Rips oder mehrsprachige Videos fast immer vergessen können. Nur wer einen DVD-Brenner besitzt, wird mit dem hohen Platzbedarf wenig Probleme haben.
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.

DTS

Der Digital Theater Sound ist inzwischen auf DVDs recht weit verbreitet. Prinzipiell gilt das gleiche wie bei AC3, nur dass DTS mit noch höheren Bitraten (768 kbit/s auf-

wärts) arbeitet. Persönlich finde ich es blödsinnig, für eine Audiospur so extrem viel Platz zu opfern.
Manchmal taucht auch 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 grundsätzlich richtig, nur sprechen praktische Erwägungen dafür, doch AC3 zu verwenden. Der Qualitätsunterschied dürfte marginal ausfallen und wohl nur in absoluten Ausnahmefällen hörbar sein. 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 als Quelle nicht möglich ist. Aus diesen Gründen werden wir uns um DTS auch nicht weiter kümmern.

Vorbis

Für Stereosound *der* Konkurrent zu MP3. 80 kbit/s klingen immer noch einwandfrei. Auch bei hoher Qualitätseinstellungen werden die 128 kbit/s kaum überschritten. Vorbis kann auch Mehrkanal-Ton, allerdings ist dieser Modus weder groß getestet noch wirklich ausgereift. Um qualitativ auf der sicheren Seite zu sein, müssen wir mit AC3-ähnlichen Bitraten kalkulieren. 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 phantastisch: 70 kbit/s für Stereo sind kein Problem und 6-Kanal-Ton kriegen wir problemlos in 128 kbit/s unter. Zwei Nachteile: Gerade der HE-Modus braucht beim Abspielen einiges an Rechenleistung. Für ältere Rechner unter 500 MHz schrumpfen AACs Vorteile damit kräftig zusammen.
Bisher beschränkte sich BeSweets AAC-Unterstützung auf den kommerziellen Nero-Encoder. Dank Dimzons Plugins sind nun einige weitere Möglichkeiten hinzugekommen:

- **Nero Digital.** Unterstützt Stereo- und 6-Kanal-Ton im LC oder HE-Modus. Sowohl konstante (CBR) als auch variable Bitrate (VBR) sind möglich, wobei der VBR-Modus höhere Qualität liefert. Allerdings setzt der Encoder eine installierte Vollversion von Nero voraus, die mit etwa 60 € zu Buche schlägt.
- **3GP-Referenz-Encoder.** Gehört - wie alle folgenden Encoder auch - zu den neuen Dimzon-Plugins. 3GP kann nur Stereo verarbeiten und unterstützt lediglich CBR-Encoding. Dafür arbeitet er immer im platzsparenden HE-Modus.
- **FAAC.** Leider bietet FAAC keine Unterstützung für den HE-Modus. Dafür verarbeitet er aber 6-Kanal-Ton und bietet sowohl VBR als auch CBR. Wie üblich sollten wir VBR wegen der höheren Qualität vorziehen.

- **Winamp.** Auch dieser Encoder arbeitet immer im HE-Modus und unterstützt 6-Kanal-Ton. Leider existiert kein VBR, so dass wir auf CBR mit maximal 128 kbit/s beschränkt sind.

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, was hauptsächlich am Leider-noch-Standard AVI liegt. Entscheiden wir uns für AVI, müssen wir uns gleichzeitig von Vorbis verabschieden, da Vorbis nicht in AVI gemuxt werden kann. VBR-MP3 und AAC funktionieren zwar, sind allerdings – wenn auch gut funktionierende – Hacks. Am unproblematischsten ist Matroska, da dieser Container ganz bequem alle hier angesprochenen Audioformate unterstützt. Die gleiche breite Unterstützung gilt für OggMedia, wenn auch das Muxen gerade für AAC weniger komfortabel ausfällt.
Die nächste Frage gilt der gewünschten Anzahl der Kanäle. Wollen wir, falls vorhanden, den originalen 6-Kanal-Ton beibehalten, bleibt uns praktisch nur die ursprüngliche oder re-encodierte AC3 oder AAC. Naja, mit Einschränkungen auch Vorbis. Definitiv außen vor bleiben muss MP3.
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. In mehr oder weniger naher Zukunft könnte AAC dazustoßen. Davon abgesehen bietet sich MP3 nicht mehr an, denn Vorbis und AAC erreichen die gleiche Qualität bei geringeren Bitraten. Als offizielles MPEG-4-Audioformat dürfte AAC langfristig die besten Chancen haben sich durchzusetzen. Nachteil im Moment ist das Fehlen eines freien Encoders, der alle Features unterstützt. Aber das kann sich ändern, wie die Dimzon-Plugins jetzt schon zeigen. Und wer weiß was passiert, wenn jemand auf die Idee kommt, den Multichannel-Modus von Vorbis weiterzuentwickeln. Zusätzlich stellt sich im Zeitalter der DVD±R die Frage: Warum nicht gleich AC3 oder DTS 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.

> Für AC3 und DTS können wir den ganzen BeSweet-Prozess überspringen, da wir den Sound original von der DVD übernehmen.

Die Programme

Das Tool der Wahl fürs Audio-Transcoding heißt BeSweet. Allerdings ist BeSweet ein Kommandozeilen-Programm, worüber wir Windowsbenutzer im Allgemeinen ja nicht unbedingt glücklich sind ;-). Deshalb gibt es schon seit Urzeiten grafische Oberflächen, die die vertraute Mausbedienung bieten und im Hintergrund die passende Kommandozeile für BeSweet zusammensetzen.

BeSweet-Kommandozeile

Wer oft unterschiedlichste Parameter beim Transcoding verwendet, sollte auf jeden Fall über den Tellerrand der GUIs hinausschauen. Ob ihr es mir glaubt oder nicht: Mit ein wenig Übung ist das Transcoding über die Kommandozeile schneller konfiguriert als mit einer GUI.
Damit die Masse an Schaltern, die BeSweet zur Verfügung stellt, kein undurchdringliches Dickicht bleibt, habe ich die *BeSweet-Kommandozeilenreferenz* geschrieben. Nähere Infos bietet der Sticky-Thread zum Thema im Doom9/Gleitz-Audioforum.

BeSweetGUI

Die traditionelle grafische Oberfläche für BeSweet, programmiert von DanniDin. Da BeSweetGUI wirklich für jede vorhandene Funktion einen Schalter anbietet, ist sie manchmal kaum einfacher zu verstehen als BeSweet selbst. Dafür bleibt kein Transcodingwunsch unerfüllt und zumindest spart man sich das Auswendiglernen der Kommandozeilenoptionen.

BeLight

Neue GUI für BeSweet von Kurtnoise13, bei der mehr die Einfachheit als die Vollständigkeit im Vordergrund steht. Und tatsächlich war BeSweet konfigurieren noch nie so einfach, ohne dass man auf wichtige Optionen verzichten müsste. Mich hat BeLight vollständig überzeugt, deswegen werde ich das Transcoding ausschließlich an BeLight erklären.

Empfehlung

Die Wahl der Oberfläche ist hauptsächlich eine Frage des Geschmacks und des Funktionsumfangs, den man benötigt. Wer das Transcoding automatisieren will oder ständig mit wechselnden Konfigurationen arbeitet, sollte dringend über den Tellerrand der GUIs hinausschauen. An dieser Stelle eine kleine Warnung: Die *BeSweet-Kommandozeilenreferenz* ist keine Anleitung wie das Encodingwissen, sondern beschreibt nur sämtliche in BeSweet vorhandenen Schalter. Wer sich an die Konsole wagt, sollte also schon ein Grundwissen über den Transcodingvorgang und einige Hintergründe mitbringen.

2.4.1 Einrichten von BeSweet

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

Um BeSweet vollständig zu installieren, benötigen wir folgende Bestandteile:

- **BeSweet v1.5b31** von *besweet.notrace.dk*. 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.
- **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 *www.rarewares.org/mp3.html*. Ich bevorzuge die aktuelle, von Hydrogenaudio empfohlene, Version »LAME 3.97 beta 1 2005-09-29«.
- **Libvorbis.dll**, falls wir nach Vorbis transcodieren wollen. Die Downloads gibt es wieder auf RareWares, unter *www.rarewares.org/ogg.html*. Aktuell empfehle ich »libvorbis.dll using aoTuVb4«, 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.
- MP2-Encoding mit **2Lame** benötigt die nicht im BeSweet-Paket enthaltene *tooLame.dll*. BeSweet liefert nur den mp2enc MP2-Encoder mit.
- **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.
- **AAC-Encoding mit Dimzons Plugins** ist schnell vorbereitet, denn das Setup der aktuellen BeLight-Version bringt die alle nötigen Dateien mit.

Haben wir alle nötige Pakete, läuft die Installation recht einfach ab. Das BeSweet-Archiv entpacken wir einfach in einen Ordner, z. B. *C:\Programme\BeSweet*. In den selben Ordner gehören auch die *VOBInput.dll*, die *lame_enc.dll* aus dem LAME-Download, *libvorbis.dll* und *libmmd.dll* für den Vorbis-Encoder und die *tooLame.dll*.
Um den Nero-Encoder verwenden zu können, kopieren wir erst die *Aac.dll* und *aacenc32.dll* aus *C:\Programme\Gemeinsame Dateien\Ahead\AudioPlugins* in den BeSweet-Ordner, bei älteren Nero-Versionen auch die *NeroIPP.dll* aus *C:\Programme\Gemeinsame Dateien\Ahead\Lib*.

Wer BeLight und/oder die Dimzon-Plugins verwenden will, installiert BeLight ebenfalls in den BeSweet-Ordner.

2.4.2 Sound 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.
Nun betrachten wir die linke Seite des BeLight-Fensters.

Abbildung 2.12

Der erste Eintrag (Abb. 2.12) gehört zu Azid. Azid ist dafür zuständig, die AC3-Tonspur 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.
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 selbst konfigurieren möchte, nimmt Boost (Abb. 2.13) 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 habe das bis jetzt auch noch nicht bereut.

Abbildung 2.13

Wenn's denn sein soll, LigH schlägt im Doom9/Gleitz-Forum (Sticky-Thread *Dynamikkompr./Boost mit BeSweet* im Audioforum) 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.

Damit weiter in den **`OTA`**-Abschnitt (Abb. 2.14). 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.

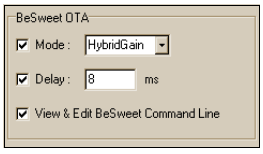


Abbildung 2.14

Haben wir uns für AAC 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« in unserem Fall (Abb. 4.4). 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.

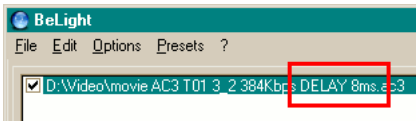


Abbildung 2.15

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 spä-

ter nicht noch ein zweites Mal berücksichtigen. Anders herum genauso: Wer es nicht in BeSweet abhandelt, muss es dann später beim Muxen tun.

Damit kommen wir zu den **Advanced Settings**. Für den Downmix auf Stereo sollten die Optionen so aussehen wie in Abb. 2.16. **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.

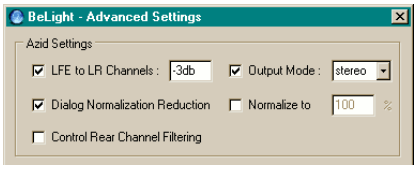


Abbildung 2.16

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

Dann haken wir noch ganz unten rechts 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.

2.4.3 Vorbis-Encoding

Die Vorbis-Einstellungen sind simpel. Grundsätzlich gilt für 6-Kanal-Vorbis das gleiche wie für Stereo-Vorbis, auch wenn wir hier nur die Stereo-Variante betrachten.
Zuerst wählen wir **Stereo**-Output (Abb. 2.17), und dann bei **Bitrate Management** die Einstellung **Quality**. Vorbis ist auf echtes VBR im Constant-Quality-Modus ausgelegt und bringt nur da seine volle Leistung. Deswegen sind die **Bitrate**-Modi eher uninteressant. Mit denen könnten wir zwar vor dem Encoding etwas genauer bestimmen, wie groß die encodierte Audiodatei wird. Da wir die endgültigen Größen aber

sowieso erst nach dem Audio-Transcoding berechnen, bringt uns das keinen Vorteil. Im Gegenteil würden wir gegenüber dem Quality-Modus Qualität verlieren.

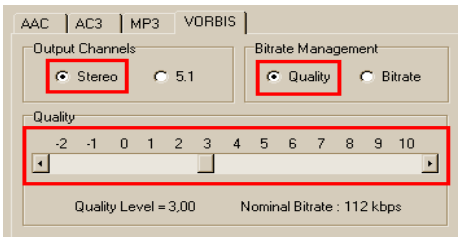


Abbildung 2.17

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

2.4.4 AAC-Encoding

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.

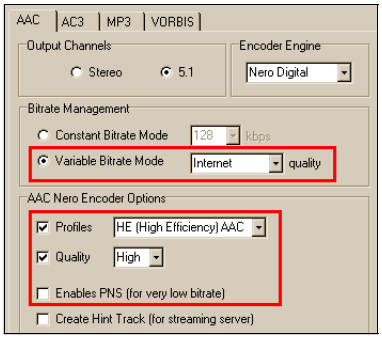


Abbildung 2.18

Im **`AAC`**-Register stellen wir bei **`Encoding Engine`** den **`Nero Digital`**-Encoder ein und wählen links daneben die passende Anzahl an **`Output Channels`**.

Wie bei Vorbis ist CBR-Encoding 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 diese HE-Bitraten erwarten:

Tape	115
Radio	128
Internet	160
Streaming	210

Dabei klingt **`Tape`** immer noch akzeptabel. Meistens verwende ich **`Internet`** oder **`Streaming`**.

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 (also die Presets **Tape** und **Radio**) kann es sich auch lohnen, den Haken bei **Enable PNS** zu setzen. Für höhere Bitraten sollten wir den Schalter allerdings deaktiviert lassen.

Der 3GP-Referenz-Encoder

Bei diesem und allen folgenden Encodern handelt es sich um die neuen Dimzon-Plugins. Da die sich noch in der Entwicklung befinden, kann es nicht schaden, den dazugehörigen Doom9.org-Thread im Auge zu behalten: *http://forum.doom9.org/showthread.php?t=99930*.

Um das auf dem 3GP-Referenzencoder basierende Plugin zu verwenden, stellen wir im **AAC**-Register **3GP Reference** als **Encoding Engine** ein.
Achtung, das Plugin unterstützt nur Stereo und CBR-Encoding und erzeugt immer HE-AAC im MP4-Container! Entsprechend einfach ist die Konfiguration. Wir müssen lediglich mit dem Schieberegler die gewünschte Bitrate einstellen.

Der FAAC-Encoder

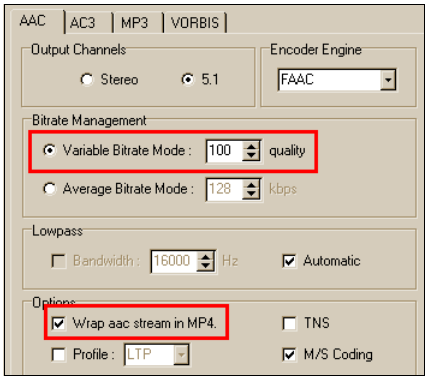


Abbildung 2.19

Der OpenSource-Projekt FAAC unterstützt leider kein HE-AAC, dafür aber sowohl 5.1 Kanäle als auch VBR. Entsprechend müssen wir mit höheren Bitraten rechnen.
Um FAAC zu aktivieren, stellen wir die **Encoder Engine** entsprechend ein, wählen die Kanalanzahl und darunter bei **Bitrate Management** den **Variable Bitrate Mode**. Genau wie bei Nero sorgt dieser Modus für die beste Qualität.
Der angegebene **Quality**-Wert darf zwischen **10** und **500** liegen. Die **100** wie in Abb. 2.19 entsprechen etwa einer Stereo-Bitrate von 120 kbit/s und liefern gute Qualität.
Wichtig ist auch die Option **Wrap aac stream in MP4**. Haben wir AVI als Container für das Video geplant, dürfen wir den Haken dort nicht setzen. Bei Matroska und OggMedia sollten wir die Option dagegen aktivieren und damit die AAC-Daten in eine MP4-Datei verpacken. So lässt sich der Track nämlich leichter weiterverarbeiten.

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.

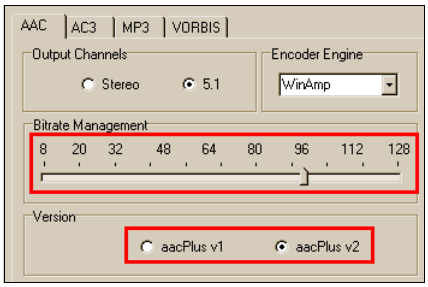


Abbildung 2.20

Die passende Einstellung der **Encoder Engine** im **AAC**-Register ist natürlich **WinAmp** (Abb. 2.20). Der Schieberegler unter **Bitrate Management** definiert die verwendete Bitrate, wobei Werte kleiner als 96 kbit/s nur für Stereo-Encodings möglich sind.

Im **Version**-Abschnitt darunter geben wir die gewünschte aacPlus-Version an. **aacPlus v1** entspricht HE-AAC. **aacPlus v2** steht für HE-AAC mit zusätzlichem Parametric Stereo, was besonders für sehr niedrige Bitraten hilfreich ist. *Niedrige Bitrate* heißt hier für Stereo-Ton ca. 48 kbit/s und darunter. Für 6-Kanal-Ton könnte **aacPlus v2** sogar bis zu den maximal möglichen 128 kbit/s nützlich sein. Allerdings ist der Modus noch zu wenig getestet, um abschließende Aussagen machen zu können.

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

2.4.5 MP3-Encoding mit LAME

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

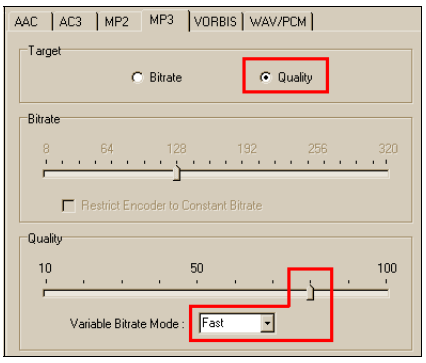


Abbildung 2.21

Da MP3 nur Stereo unterstützt, entfällt die Auswahl der Zielkanäle. Wir brauchen nur im **MP3**-Register unter **Target** auf **Quality** zu klicken, um der VBR-Modus einzustellen (Abb. 2.21). 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 eine kleine Tabelle, welche Qualitätsstufe für welches Preset steht und welche

Bitraten wir erwarten sollten.

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

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 *www.HydrogenAudio.org* 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, 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. Und das geht so wie in Abb. 2.22 gezeigt.

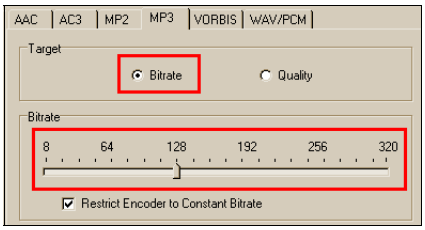


Abbildung 2.22

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.

2.4.6 AC3-Encoding

BeSweets AC3-Encoder kann zwar mit kommerziellen Lösungen nicht mithalten, ist aber lange nicht mehr so schlecht, wie er einmal war. Wir haben also durchaus die Möglichkeit, anständige AC3s zu erzeugen.
Wichtig ist, dass ac3enc die Normalisierungseinstellungen im **OTA**-Abschnitt ignoriert. Deshalb müssen wir das Azid erledigen lassen, indem wir die Funktion **Normalize to** in den **Advanced Options** verwenden und die Normalisierung im **OTA**-Abschnitt ausschalten.
Der Ziel-AC3 eine DVD-übliche Bitrate von 384 oder 448 kbit/s zu gönnen, hat wenig

Sinn. Dann hätten wir gleich das Original beibehalten können. Tiefer als die bei **Bitrate** mit dem Schieberegler ausgewählten 256 kbit/s (Abb. 2.23) sollten wir für 6-Kanal-Dateien allerdings nicht gehen, um die Qualitäts-

Abbildung 2.23

5.1 unter **Output Channels** behält die sechs Kanäle der Quelldatei bei. Stereo-AC3s zu encodieren oder gar originale Stereo-AC3s zu verklei-

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

2.5 Untertitel: Eine Einführung

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 Bild »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.

Für die beiden dynamischen Varianten haben wir zusätzlich die Möglichkeit, die

Untertitel als extra Datei zu speichern oder als Untertitel-Spur ins fertige Video zu muxen. Die extra Datei hat den Nachteil, dass wir den Film nicht gut verpackt als einzelne Datei vorliegen haben. Außerdem müssen die Untertitel-Dateien genau richtig benannt werden (meistens genauso wie die Filmdatei), damit sie der Player automatisch erkennt. Das Muxen vermeidet diese Probleme, deswegen würde ich separate Untertitel-Dateien nach Möglichkeit vermeiden. Allerdings lassen sich mit Gordian Knot nur Text-Untertitel muxen, keine VobSubs. Manuell stellt das aber kein Problem dar.
Mit diesem Grundwissen können wir uns jetzt an die Verarbeitung wagen. Das nächste Kapitel beschäftigt sich mit fest im Bild eingebrannten Untertiteln, das folgende dann mit dynamischen, die sich ein- und ausblenden lassen.

2.5.1 Vorbereiten von festen Untertiteln

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

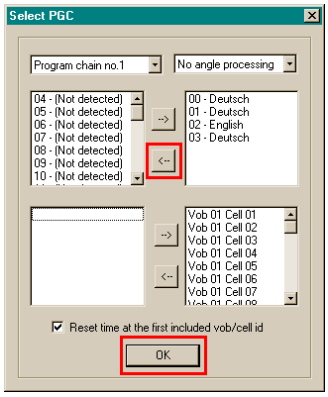


Abbildung 2.24

Wir starten VobSub, entweder über Gordian Knot, Register **`Subtitles/Chapters`** und den **`Configure`**-Button links oben, 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 ein Fenster, in dem wir den Speicherort der Untertitel angeben. Dann erscheint das Fenster in Abb. 2.24.

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 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.

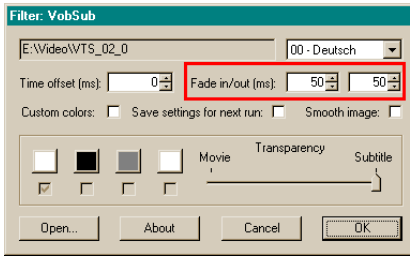


Abbildung 2.25

Normalerweise blendet VobSub jeden Untertitel kurz ein und wieder aus. Die Standard-Einstellung von 50ms 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** in Abb. 2.25. 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

```
# 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
```

Abbildung 2.26

Jede der vielen Zeilen in Abb. 2.26 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 die Zeilen wie in Abb. 2.27 (direkt über der ersten Untertitel-Spur). Hinter **langidx** tragen wir die gemerkte Zahl ein.

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

Abbildung 2.27

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 den Zeilen in Abb. 2.28.
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.

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

Abbildung 2.28

2.5.2 Erzeugen von dynamischen Text-Untertiteln

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. Im zweiten Fall müssen wir auf das Muxing mit VirtualDubMod verzichten (das unterstützt keine VobSubs) und je nach Container (vgl. S. 50) mkvmerge (für Matroska) oder AVI-Mux GUI (für AVI) verwenden. OggMedia unterstützt keine VobSubs.

In diesem Kapitel kümmern wir uns um Text-Untertitel. Das benötigte Programm heißt SubRip. Dort gelangen wir über **`Options / Global Options`** in diesen Dialog:

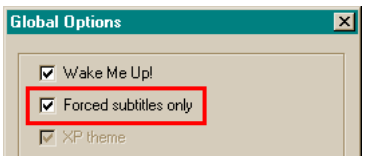


Abbildung 2.29

Mit einem Haken bei **`Forced subtitles only`** in Abb. 2.29 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 (Abb. 2.30) auf. Am einfachsten ist es nun, mit **`Open IFO`** die vom Ripper erstellt Ifo-Datei zu öffnen. Alternativ können wir auch mit **`Open Dir`** die VOBs direkt laden. Vorteil der ersten Methode ist, dass dann im Dropdown-Feld darüber die Untertitelspuren richtig bezeichnet werden. Diese Infos stehen nämlich in der Ifo-Datei, nicht in den VOBs. Im Dropdown-Feld stellen wir die gewünschte Sprache ein und widmen uns dann der rechten Seite des Fensters (Abb. 2.31).

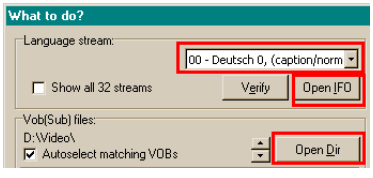


Abbildung 2.30

`Last Time Code` muss Null sein. Steht hier ein Wert, addiert ihn SubRip zu jeder Zeitmarke dazu, was zu asynchronen Untertiteln führt.

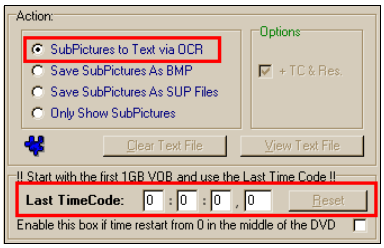


Abbildung 2.31

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 Abb. 2.32.

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.
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.

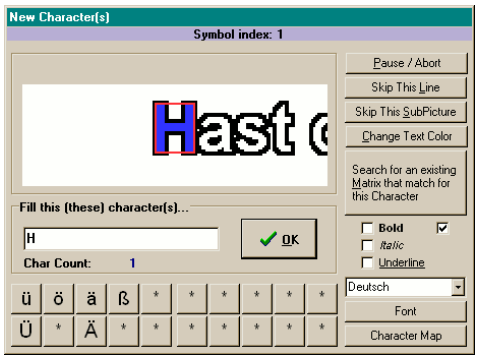


Abbildung 2.32

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 rippen 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.

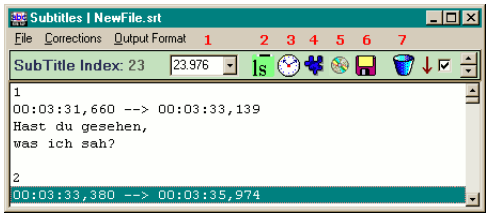


Abbildung 2.33

Damit zurück zum Editor-Fenster, in dem wir den gerade gerippte Untertitel in Textform (genauer: im SubRip-Format) sehen (Abb. 2.33). Ich gehe davon aus, dass wir die Untertitel-Spur am Ende auch im SubRip-Format speichern wollen. Dann können wir (1) ignorieren. Die richtige Bildrate (die uns DGIndex verraten hat) müssten wir dort nur einstellen, wenn wir die Untertitel in einem Frame-basierten Format wie MicroDVD speichern wollten. Mit dem Button (2) lassen sich die gröbsten Fehler der Texterkennung korrigieren. Die ist zwar recht gut, aber von fehlerfrei weit entfernt. Bei (3) lassen sich Zeitkorrekturen vornehmen, wenn der Untertitel nicht synchron zum Film läuft. Das dürfte aber kaum vorkommen, wenn Untertitel und Film direkt von der DVD stammen. Mit (4) könnten wir die Untertitel in ein anderes Format konvertieren und (5) dient zum Splitten der Untertitel in mehrere Dateien. Nötig ist das nur, wenn wir die Untertitel in einer externen Datei abspeichern (ansonsten wird das Splitten später zusammen mit dem gesamten Film erledigt), und das wiederum dürfte nur nötig sein, wenn der Standalone-Player mit Untertitelspuren Stress macht. Da Standalone-Player

im Encodingwissen kein Thema sind, bleibt Button (6), der die aktuelle Untertitel-Datei speichert. Das sollten wir natürlich nicht vergessen.

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

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

Teil 3

Encodieren mit Gordian Knot

3.1 Gordian Knot und Codecs konfigurieren

Nach dem ersten Start von Gordian Knot wechseln wir in das Register **Program Paths** und erklären GKnot, wo die verschiedenen Programme liegen. Die **Don't Check**-Checkbox verhindert, dass GKnot 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. Dann wechseln wir ins Register **Options**.

Wichtig ist hier zuerst wie in Abb. 3.1 der fehlende Haken bei **Use advanced SaveAVS window**. Nicht, dass das erweiterte Fenster schlecht wäre. Ich mag es sogar sehr gerne. Aber alles, was wir hier im Encodingwissen besprechen, bietet aber 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** (Abb. 3.2).

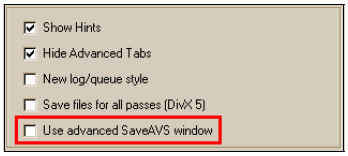


Abbildung 3.1

Die in diesem Teil des Fensters getroffenen Einstellungen gelten als Standard für alle Encodings, können aber im Einzelfall auch wieder überschrieben werden.

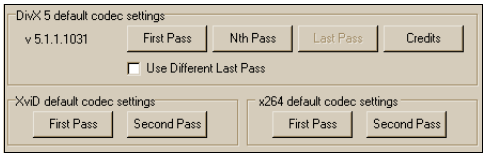


Abbildung 3.2

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

> 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.

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 stär-

ker 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.

> Details zur Codec-Konfiguration gibt's in den Kapiteln weiter unten.

Damit sind wir mit der Grundkonfiguration fertig und können ins Register **Bitrate** wechseln, um uns über gewünschten Container und Codec Gedanken zu machen.

3.2 Auswahl von Container und Codec

Die Container

Abbildung 3.3

Jeder Anfänger stolpert recht bald über den Begriff *Container*. Die meisten dürften keine genau Vorstellung haben, was damit gemeint ist. 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 Indeo 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 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. Sehen wir uns also die einzelnen Container etwas näher an.

Audio/Video Interleave (AVI)

Dieser Container ist der älteste, der von Gordian Knot unterstützt wird, und gleichzeitig noch immer der am weitesten verbreitetste. Allerdings merkt man ihm sein Alter immer deutlicher an. Die gängigsten AVI-Implementierungen unterstützen nur Audio mit konstanter Bitrate wie CBR MP3 oder Wave. Für VBR MP3 und AAC existieren gut funktionierende Tricks. Vorbis ist nicht möglich. AVI unterstützt einigermaßen Untertitel, allerdings keine Kapitel.

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. Es existiert keine Dokumentation dafür und die Entwicklung steht seit einiger Zeit still. Durch das Aufkommen von Matroska und MP4 stellt sich immer mehr die Frage, wozu OGM noch gebraucht wird.

Matroska (MKV)

Das ambitionierte freie Containerprojekt von *www.matroska.org* ist mit dem Anspruch angetreten, irgendwann AVI zu ersetzen. Das Potenzial dazu hat der Container allemal: Er unterstützt selbstverständlich eine Vielzahl Audio- und Videoformate, natürlich auch Untertitel und Kapitel. Das nächste große Feature könnten Menüs wie auf der DVD werden. Was bisher fehlt ist die breite Unterstützung durch die Medienindustrie. Besonders eine Matroska-Unterstützung in Standalone-Playern wäre wünschenswert, die möglicherweise demnächst in einem ersten Player verfügbar sein wird.

Empfehlung:

Für Standalone-Player bleibt nur eine Wahl: AVI und in Zukunft wahrscheinlich auch MP4 und mit etwas Glück Matroska. Aktuell gibt es noch keine echte Alternative zu den mehr oder weniger umfassenden Fähigkeiten so genannter DivX-fähiger Standalones.
Wer sich auf seinen Computer verlässt oder über irgend eine TV-Out-Konstruktion

den Computer an den Fernseher hängt, dem stehen alle Möglichkeiten offen. Das heißt, die Wahl hängt hauptsächlich 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.

Die Codecs

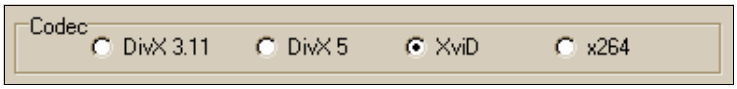


Abbildung 3.4

DivX ;-) 3.11

So heißt der gehackte MS MPEG4v3-Codecs. 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 ;-) die Massenbewegung um digitales Video ausgelöst.
Das SBC-Encoding ist gerade für Anfänger unübersehbar kompliziert. Moderne Codecs bieten auf einfachere Weise bessere Ergebnisse, weshalb DivX ;-) inzwischen als veraltet gilt. Dazu kommt, dass MS MPEG4v3 trotz des Namens nicht standardkonform ist. DivX ;-) erzeugt also keine MPEG-4-kompatiblen Videostreams. Und wegen seines Charakters als Hack einer Microsoft-Software steht der Codec rechtlich auf wackligem Terrain.
Im Encodingwissen werden wir uns mit DivX ;-) nicht beschäftigen.

DivX 5

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 Networks vertrieben. Er bringt gute Qualität und lässt sich recht einfach konfigurieren, arbeitet allerdings eher langsam.

XviD

In Zeiten von DivX 4 hat sich XviD als eigenständiges Projekt abgespalten. Der Codec ist nichtkommerziell und sein Quellcode auf *www.xvid.org* frei zugänglich. XviD wird von einem enthusiastischen (und verdammt fähigen!) Team von Programmieren in deren Freizeit weiterentwickelt. Spätestens seit der Version 1.0 gilt er DivX gegenüber in Qualität und Geschwindigkeit als überlegen. Allerdings kann sich gerade der Anfänger leicht in den vielen Einstellungsmöglichkeiten verirren.

x264

Sowohl XviD als auch DivX 5 erzeugen Video nach dem Standard *MPEG-4 Part 2*, besser bekannt als *MPEG-4 Advanced Simple Profile*, kurz ASP. x264 verwendet auch MPEG-4, allerdings einen anderen Teil des Standards: *Part 10*. In Anlehnung an die Langform von AAC hat der den Namen *Advanced Video Coding* erhalten, kurz AVC oder auch H.264, wenn man nach der Dokumentnummer der ITU-Organisation geht.
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 getunten 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. x264 ist neu und mitten in der Entwicklung. Deshalb sollten wir, wenn wir ihn verwenden, sämtliche Videosoftware auf dem aktuellsten Stand halten. Das gilt auch für Player und Decoder! Apropos Decoder: ffdshow verrichtet bei mir mal wieder gute Dienste.

Empfehlung

Der alte Hase, der seit 1999 schon Videos codiert, greift vielleicht noch zu DivX 3.11. Alle anderen sollten sich an den neueren DivX oder XviD halten. Die Wahl hängt dabei nicht unwesentlich von persönlichen Vorlieben ab.
Wer sich wenig Gedanken machen will – zwei oder drei Optionen einstellen und loslegen – der ist mit DivX gut beraten, sollte allerdings etwas Geduld mitbringen, da DivX in Sachen Geschwindigkeit nicht besonders glänzt. Wer maximale Bildqualität herausholen will und bereit ist, sich dafür auch mit den Hintergründen der ganzen Einstellungen zu beschäftigten, dürfte an XviD mehr Freude haben. Als Bonus kommt die spürbar höhere Geschwindigkeit des freien Codecs dazu.
Mit x264 habe ich noch zu wenig Erfahrung, um eine Empfehlung abgeben zu können. Die Entwicklung ist jedenfalls weit genug fortgeschritten, um ihn nicht nur für Tests, sondern auch für ernsthafte Backups zu verwenden. Bis zur unerschütterlichen Zuverlässigkeit eines XviD hat er trotzdem noch ein ganzes Stück Weg vor sich. Als Anfänger sollte man deshalb vielleicht erst einmal etwas Erfahrung sammeln und dann wieder zu x264 zurückkehren. Außerdem dürfen wir nicht vergessen, dass AVC unabhängig vom genauen Codec sowohl beim Encodieren als auch beim Decodieren mehr Rechenleistung schluckt als ein ASP-Codec.

3.2.1 Konfiguration des XviD-Codecs

XviD 1.1 ist ein sehr genau konfigurierbarer Codec. Man könnte Bücher darüber schreiben, wann welche Kombination von Optionen sinnvoll ist. Um den Guide nicht ausufern zu lassen, erkläre ich die einzelnen Einstellungen nur kurz. Wer tiefer einsteigen will, dem kann ich nur Selurs »Wissenswertes rund um XviD« und crusty's (englische) XviD-FAQ ans Herz legen.

Einstellungen für den 1st Pass

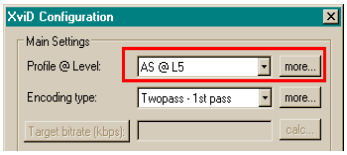


Abbildung 3.5

Profile und deren Level schalten einzelne Optionen des Codecs frei oder sperren sie, um Konformität zu den gleichnamigen MPEG-4-Profilen zu gewährleisten. Für einen normalen DVD-Rip bietet sich wie in Abb. 3.5 **AS @ L5** (Advanced Simple, Level 5) an. Über den **More**-Button erreichen wir den Dialog für die Feineinstellungen in Abb. 3.6.

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 höheren Datenraten von 2-CD-Rips. **H.263** ergibt dagegen ein etwas weicheres Bild und erhöht damit die Kompression ein wenig; geeignet eher für die niedrigen Datenraten von 1-CD-Rips.

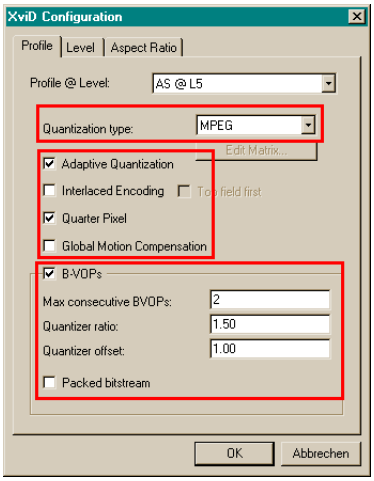


Abbildung 3.6

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. Gerade bei geringen Bitraten macht sich das positiv bemerkbar.

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

Global Motion Compensation versucht die Kompression zu erhöhen, indem es nach gemeinsamen Bewegungsvektoren in der Szene sucht. Zooms und Kameraschwenks bieten sich dafür besonders an. Ob GMC wirklich spürbare Unterschiede bringt, ist nicht ganz unumstritten. Aktivieren sollten wir es v. a. dann, wenn wir Qualität um jeden Preis wollen. Dann wählen wir später auch einen hohen VHQ-Wert in den **Advanced Options**.

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

B-VOPs (technisch für B-Frames) sollten wir ohne guten Grund nicht abschalten. **Max consecutive B-VOPs** gibt an, wie viele B-Frames maximal direkt hintereinander stehen dürfen. Mit der Standardeinstellung **2** wäre also eine solche Bildsequenz möglich: IPBBP, bei **3** wären es dann IPBBBP usw. Wichtig: Die Einstellung definiert nur das

Maximum. **2** bedeutet also nicht, dass *immer* zwei B-Frames hintereinander stehen. Es heißt nur, dass *niemals mehr* als 2 aufeinander folgen dürfen. Innerhalb dieser Grenze berechnet XviD dann die günstigste Möglichkeit. Die Standardeinstellung ist ein sinnvoller Wert, nur für Standalone-Player müssen wir evtl. auf **1** herunter gehen.

Quantizer ratio und **Quantizer offset** bestimmen, wie hoch B-Frames im Vergleich zu P-Frames komprimiert werden. Die Standards sind in den meisten Fälle eine gute Wahl. Für meine hochqualitativen Backups nehme ich auch gerne Didées Standardwerte: **2/1.63/0**.

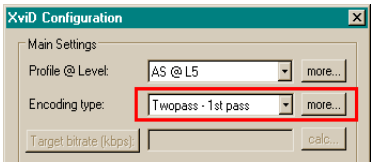


Abbildung 3.7

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

Damit ist dieses Fenster abgehandelt und wir können zurück in den Hauptdialog (Abb. 3.7). Mit **Twopass - 1st pass** stellen wir den ersten Durchgang ein und rufen über den **More**-Button die Details auf (Abb. 3.8).

Im angegebenen **Stats file** werden die Informationen aus dem 1st Pass gespeichert, die der 2nd Pass dann weiterverwendet. **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.

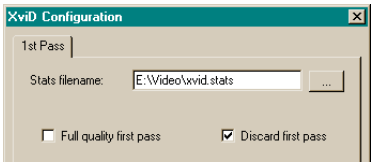


Abbildung 3.8

Damit kehren wir zurück ins Hauptfenster und kümmern uns um die **Zones**.

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.

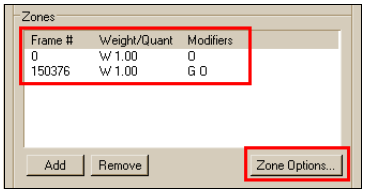


Abbildung 3.9

Über den **Zone options**-Button (Abb. 3.9) gelangen wir in den Konfigurationsdialog in Abb. 3.10.

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

Zonen können entweder mit einem konstanten **Quantizer** (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.

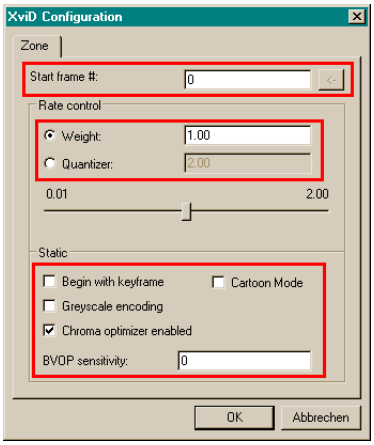


Abbildung 3.10

`Begin with a keyframe` erzwingt am Anfang der Zone ein I-Frame. **`Greyscale encoding`** verwirft alle Farbinformationen, so dass wir ein Schwarzweiß-Video bekommen. **`Chroma optimizer`** wirkt Pixeltreppchen an scharfen Kanten entgegen, bremst aber den Encoding-Vorgang etwas. Obwohl ich mir nicht sicher bin, ob es wirklich sichtbare Verbesserungen bringt, lasse ich die Option immer eingeschaltet.

Der **`Cartoon Mode`** ist nützlich bei Cartoons – wer hätte das gedacht. Cartoon meint dabei allerdings richtig klassisch gezeichneten Zeichentrick a la Tom & Jerry. Animationen wie Monster AG gelten für XviD nicht als Cartoon.

Die **`BVOP sensitivity`** schließlich wirkt sich darauf aus, wie gern XviD B-Frames setzt. Positive Werte ermutigen XviD zu mehr B-Frames, negative Werte schrecken den Codec eher ab. Die **`0`** können wir als sinnvollen Standardwert stehen lassen.

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

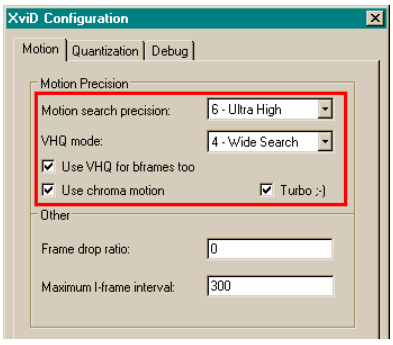


Abbildung 3.11

Über **`OK`** gelangen wir zurück in den Hauptdialog und rufen unten in der Mitte die **`Advanced options`** auf (Abb. 3.11).

Die **`Motion search precision`** legt fest, wie intensiv XviD nach Bewegungen sucht. Es gibt kaum einen Grund von **`6 - Ultra High`** abzuweichen.

VHQ rechnet für die einzelnen Macroblocks verschiedene Szenarien durch und entscheidet sich dann für das mit der geringsten Anzahl Bits. Generell gilt: Jede höher der **`VHQ mode`** desto besser die Qualität und desto langsamer das Encoding. **`1`** oder **`2`** bringen im Vergleich zum Bremseffekt den höchsten Qualitätsgewinn. Ich bin Qualitätsfanatiker genug, um meistens bei der höchsten Einstellung **`4`** zu bleiben. Wenn wir GMC aktiviert haben, sollte VHQ nicht abgeschaltet sein.

`Use VHQ for bframes too` aktiviert den VHQ-Modus auch für B-Frames. Da es sich dabei immer um VHQ 1 handelt, leidet die Geschwindigkeit wenig. Zwar wären auch höhere VHQ-Modi für B-Frames möglich. Laut XviD-Entwickler syskin würden die aber kräftig bremsen und kaum einen spürbaren Effekt auf die Qualität haben.

`Use chroma motion` veranlasst XviD, bei der Suche nach Bewegung nicht nur die Helligkeitsinformationen (Luminanz) zu berücksichtigen, sondern auch die Chromi-

nanz (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. Da der Unterschied kaum jemals sichtbar ist, lasse ich Turbo eingeschaltet.
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 Splitting-Kapitel ab Seite 77.

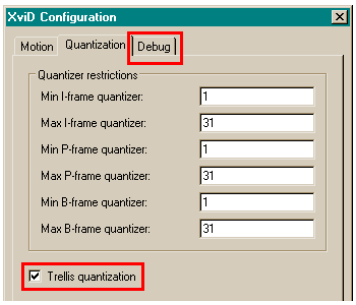


Abbildung 3.12

Die Optionen auf der **Quantization**-Seite (Abb. 3.12) sind sinnvoll eingestellt, so dass wir das meiste einfach übernehmen können. Interessant ist nur **Trellis quantization**. Trellis »überdenkt« die einmal getroffene Quantisierungsentscheidung und versucht sie zu verbessern. Heftige Qualitätsgewinne dürfen wir davon nicht erwarten, andererseits bremst Trellis auch nicht besonders und kann deshalb bedenkenlos eingeschaltet bleiben.

Auf der **Debug**-Seite haben wir noch die Möglichkeit, das Statusfenster abzuschalten. Inzwischen bremst es das Encoding zwar kaum noch, 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

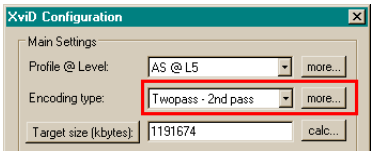


Abbildung 3.13

Im Hauptdialog stellen wir wie in Abb. 3.13 **Twopass - 2nd pass** ein. Da sich Gordian Knot um die richtige **Target size** kümmert, stürzen uns per **More**-Button auf die Details.
Eine kurze Kontrolle kann nicht schaden, ob XviD korrekt den Namen des Stats-Datei aus dem 1st Pass übernommen hat. Das sollte der Fall sein. Alle Optionen darunter können wir guten Gewissens auf den Standardwerten belassen. Genauso verändern wir die restlichen Einstellungen im Vergleich zum 1st Pass nicht. Damit ist die Konfiguration des 2nd Pass auch schon abgeschlossen.

3.2.2 Konfiguration des DivX-Codecs

Achtung! DivX 6 unterscheidet sich so stark von seinen Vorgängern, dass er im Moment noch nicht mit Gordian Knot benutzt werden kann. Um den neuen Codec zu verwenden, müssen wir das Encoding also manuell mit VirtualDubMod durchführen. Deshalb bleibt das Kapitel zu DivX 6 so lange im Codec-Archiv, und damit weder in der PDF- noch in der 7-Zip-Version, bis Gordian Knot angepasst ist.

Einstellungen für den 1st Pass

Zuerst rufen wir über den **`Select Divx Certified Profile`**-Button den Profilassistenten von DivX 5.2.1 auf.

Um volle Kontrolle über alle Optionen zu erhalten, müssen wir den Haken bei **`DivX Certified`** entfernen (Abb. 3.14) und damit die Profile deaktivieren. Dann klicken wir uns auf die nächste Seite weiter.

Abbildung 3.14

Was die drei Optionen in Abb. 3.15 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.

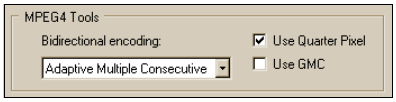


Abbildung 3.15

Maximal zwei aufeinander folgende B-Frames erlaubt **`Adaptive Multiple Consecutive`**.

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.

Damit verlassen wir den Assistenten und kommen wieder auf die **`General`**-Seite (Abb. 3.16). 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

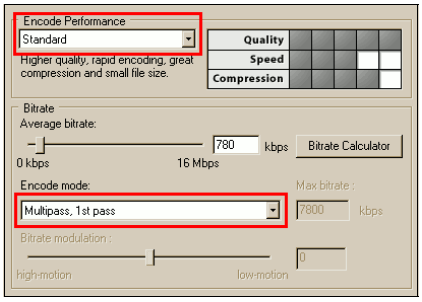


Abbildung 3.16

funktioniert nur mit **Standard**. Wenn wir Quarter Pixel aktivieren, stellt DivX den Modus automatisch um und blendet den Schieberegler ab.
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 in Abb. 3.17.

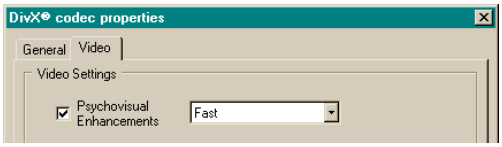


Abbildung 3.17

Hier sind erst eimal die **Psychovisual Enhancements** 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.
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.

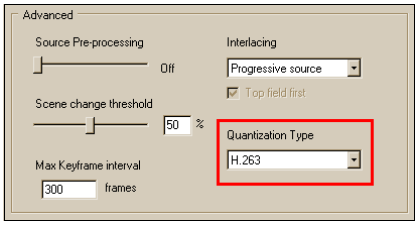


Abbildung 3.18

Unter **Quantization Type** in Abb. 3.18 lässt sich angeben, welche Quantisierungsmatrix DivX verwenden soll. Details dazu haben wir bei der XviD-Konfiguration schon gesehen. Da **MPEG-2** noch nicht richtig ausgereift ist, würde ich vorerst bei **H.263** bleiben.
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 Encodens 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 (Abb. 3.19).
Hier stellen wir **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-Rips den

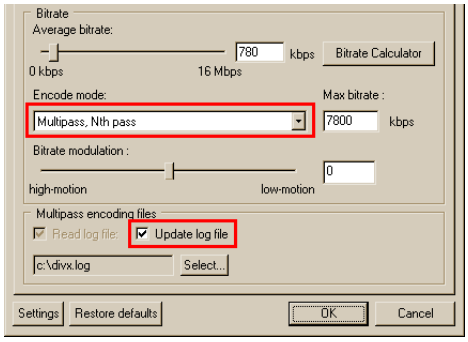


Abbildung 3.19

Zeitaufwand wert sein.

Zurück zu Gordian Knot: Dort haben wir die Möglichkeit, den letzten Pass noch einmal extra zu konfigurieren. Das ist dann sinnvoll, wenn wir in den ersten Durchgängen mit **`Encode Performance Standard`** eher auf Geschwindigkeit setzen und dann im letzten Durchlauf mit **`Slow`** das Video auf Qualität trimmen. QPel-Liebhabern bleibt diese Möglichkeit natürlich verwehrt.

Einstellungen für Credits

Natürlich vertragen die Credits mit DivX genauso wie mit XviD eine höhere Kompression. DivX hat die Funktionalität, Credits extra zu behandeln, aber nicht eingebaut. Deshalb übernimmt das Gordian Knot.

Als Modus wählen wir **`1-pass quality-based`**. Den **`Quantizer`** stellen wir auf **`15`** bis **`20`**. Eine Logdatei ist für Credits nicht nötig und um konform zum MPEG-Standard zu bleiben, sollten wir die anderen Optionen so wählen wie für den restlichen Film auch.

3.2.3 Konfiguration des x264-Codecs Rev 333

Bisher stand an dieser Stelle der Kommentar, dass x264 langsam den Kinderschuhen entwächst. Tja, spätestens seit dem Umbau der GUI ist der Codec tatsächlich erwachsen und hundert Prozent alltagstauglich. Trotzdem eignet sich nach wie vor Selurs *Wissenswertes rund um x264* sehr gut zum Querlesen.

Einstellungen für den 1st Pass

Wie bei den anderen Codecs auch, interessieren uns die 1-Pass-Einstellungen nicht. Im **`Bitrate`**-Register klicken also **`Multipass - First Pass (fast)`** (Abb. 3.20).

Abbildung 3.20

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.

Um die restlichen Einstellungen in diesem Register küm-

mert sich Gordian Knot automatisch, deswegen können wir gleich zu **Rate Control** wechseln.

B-frames reduction steuert die Qualität der B-Frames und ist deshalb natürlich nur wichtig, wenn wir überhaupt B-Frames verwenden (Abb. 3.21). Der Wert gibt an, wie viel Prozent relative Bitrate ein B-Frame weniger bekommt als das vorangehende P-Frame. Je höher also der Wert, desto stärker werden B-Frames komprimiert. Gerade bei sehr niedrigen oder sehr hohen Datenraten bietet sich diese Option als Spielzeug an, um die B-Frames ein wenig kleiner oder ein wenig größer werden zu lassen. Wichtig ist dabei, nicht zu übertreiben, weil zu hohe Werte schnell Artefakte erzeugen und zu niedrige den Nutzen der B-Frames stark schmälern. Wenn wir uns eine sehr schwache Reduktion leisten können, sollten wir überlegen, ob wir die B-Frames nicht gleich ganz abschalten.

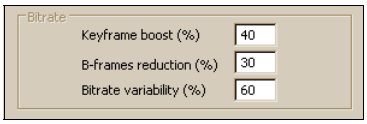


Abbildung 3.21

Bitrate variability gibt an, wie stark die Bitrate über den ganzen Film hinweg schwanken darf. **0** würde zu einem CBR-Encoding führen, **100** steht für den Constant-Quality-Modus. Vorsicht bei Änderungen, denn wie genau x264 die gewünschte Dateigröße treffen kann, hängt nicht zuletzt von dieser Einstellung ab.

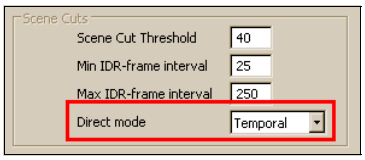


Abbildung 3.22

Wenden wir uns dem unteren Bereich des Fensters zu (Abb. 3.22). **Direct Mode** gibt an, nach welchem Verfahren B-Frames komprimiert werden. **Temporal** verwendet zur Komprimierung die zeitlichen Änderungen zwischen den Bildern, **Spatial** die räumlichen innerhalb des Frames. In der Regel sollten wir **Temporal** wählen, da es die höhere subjektive Qualität bietet.

Die anderen Werte können wir bedenkenlos auf dem Standard belassen und zum **Mbs & Frames**-Register wechseln.

x264 erlaubt es, einen 16 × 16 Pixel großen Macroblock in kleinere Einheiten (Partitionen) zu zerlegen, um so die Bewegungssuche zu verfeinern. Mit den Schaltern in Abb. 3.23 können wir einstellen, welche Zerlegungen für I-, P- und B-Frames erlaubt sind. Die Partitionierung senkt zwar die Encoding-Geschwindigkeit deutlich, steigert aber gleichzeitig die Kompression genauso deutlich. Deshalb sollten wir dem Codec nur mit gutem Grund nicht alle Partitionsmöglichkeiten erlauben.

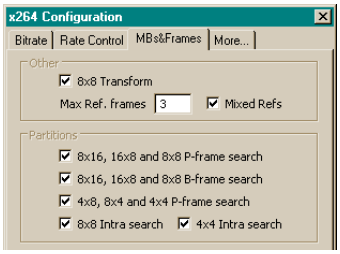


Abbildung 3.23

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 Macroblock, sondern für jede Blockpartition einzeln auszuwählen. Erste Tests zeigen, dass **Mixed Refs** zwar die Encodiergeschwindigkeit senkt, aber auch zu einem spürbaren Qualitätsanstieg führen kann. Deswegen sollten wir die Option wenn möglich aktivieren.

Im unteren Teil des Registers (Abb. 3.24) konfigurieren wir die B-Frames: **Max consecutive** legt die maximale Anzahl aufeinander folgender B-Frames fest. Eine **3** heißt, dass nicht mehr als drei B-Frames direkt hintereinander vorkommen dürfen. Ob diese Zahl auch erreicht wird, bestimmt x264 automatisch je nach Bedarf. **0** schaltet die B-Frames ganz ab.

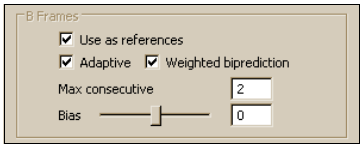


Abbildung 3.24

Use as reference erlaubt es x264, ein B-Frame als Referenzframe zu verwenden. Das erhöht zwar die Anforderungen beim Decoding, sollte aber im Sinn der Qualität aktiviert werden, wenn es der verwendete Decoder unterstützt. Bei ffdshow z. B. ist das der Fall.

Ein Haken bei **Adaptive** verteilt proportional mehr B-Frames an langsame Szenen. Da sich diese gerade für den B-Frame-Einsatz anbieten, hat das auch Sinn.

Bias geht in eine ähnliche Richtung, beeinflusst aber insgesamt, wie gerne der Codec B-Frames setzt. Die Funktion ist also mit XviDs **BVOP sensitivity** vergleichbar. Werte über **0** drängen x264 häufiger zum B-Frame-Einsatz – ohne allerdings das unter Max consecutive definierte Maximum zu überschreiten –, Werte unter **0** halten den Codec eher vom B-Frame ab. Genau wie bei XviD können wir ruhigen Gewissens die **0** stehen lassen.

Es bleiben nun noch die Einstellungen im More-Register übrig. **Partition decision** (Abb. 3.25) 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.

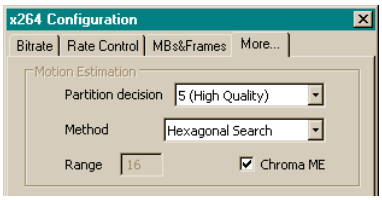


Abbildung 3.25

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 ein-

fach 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.

Unter **Threads** (Abb. 3.26) können wir die Multiprozessor-Unterstützung einschalten. Einige Berechnungen werden dann auf den verschiedenen (virtuellen) Prozessoren parallel abgearbeitet, was die Geschwindigkeit erhöht. Wer einen Pentium 4 mit Hyperthreading sein Eigen nennt, darf hier **2** eintragen. Wer gar ein echtes Multiprozessorsystem besitzt, tippt seine Anzahl an Prozessoren ein, allerdings maximal **4**. Alle anderen können die Funktion nicht nutzen und müssen bei **1** bleiben.

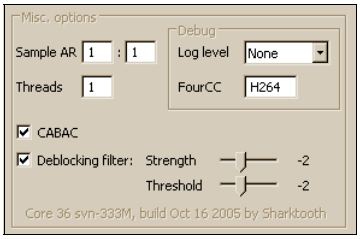


Abbildung 3.26

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.

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

Deblocking filter kennen wir bis jetzt nur von Decodern, die damit Blockartefakte im Bild zu übertünchen versuchen. x264 setzt einen solchen Filter schon beim Encodieren ein, und zwar nachdem ein Bild codiert wurde, aber bevor es als Referenz für das nächste Bild dient. Dadurch kann Bild 2 mit einer Referenz arbeiten, die weniger Artefakte enthält. Das tut der Qualität gut. Der Nachteil ist die sinkende Geschwindigkeit, um zwar sowohl beim Encoding als auch beim Decoding. Denn damit Bild 2 korrekt decodiert werden kann, muss das vorangehende Bild in gefilterter Form vorliegen. Das ähnelt einem XviD-Encoding bei dem wir gezwungen sind, beim Anschauen den Deblocking-Filter zu aktivieren.
Strength und **Threshold** steuern die Stärke des Filters. Beide Werte sollten wir aber als Einheit betrachten und auch gemeinsam verändern. Die Normaleinstellung **0** ist ein brauchbarer Standard. Negative Werte führen zu schwächerem Deblocking, positive zu stärkerem. Für hohe Datenraten bietet es sich an, das Deblocking etwas zu verringern (etwa auf **-2**), während für die niedrigen Raten von 1-CD-Encodings etwas höhere Werte angebracht sein können.

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

Einstellungen für den Nth Pass

Wir stellen im **Bitrate**-Register um auf **Multipass - Nth pass** und aktivieren **Update stats file**. x264 unterstützt genau wie DivX mehr als zwei Durchgänge. Ob das tatsächlich eine Qualitätssteigerung ergibt, die den Mehraufwand rechtfertigt, müssen wir ausprobieren. Ähnlich wie bei DivX dürften mehr als drei Passes kaum Sinn haben. Ich selbst bin von XviD verwöhnt und einfach zu faul für mehr als zwei Durchgänge. :-)
Um die Einstellung der Bitrate kümmert sich Gordian Knot automatisch. Deshalb überprüfen wir noch, ob alle anderen Einstellungen mit dem 1st Pass übereinstimmen und sind auch schon fertig. Noch existiert für x264 keine Möglichkeit, Credits extra zu encodieren, weshalb wir uns darum keine Gedanken zu machen brauchen.

3.3 Kalkulieren der Bitrate

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

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 schon feststeht, stellen wir **Calculate average bitrate** ein. Dann wählen wir **Container** und **Codec** aus (siehe S. 50) und kümmern uns um die endgültige Größe des Films.

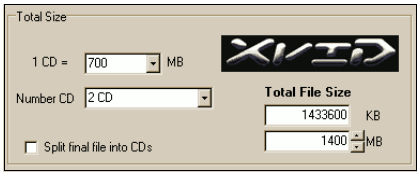


Abbildung 3.27

Bei **Total Size** in Abb. 3.27 stellen wir die Rohling-Größe und gewünschte Anzahl CDs 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.

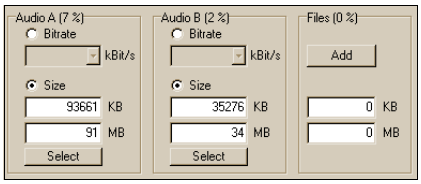


Abbildung 3.28

Der Abschnitt in Abb. 3.28 enthält alle Daten, die außer dem Video auf die Disc müssen. Mit **Select** wählen wir die erste erstellte Audiospur aus, deren Größe in die Felder darüber übernommen wird. Das Ganze noch einmal für eine eventuelle zweite Audiospur. Der **Files**-Abschnitt ist für Dateien gedacht, die zusätzlich zum Film auf die Disc sollen, z. B. dynamische Untertitel oder Special Features von der DVD. Je mehr hier steht, desto geringer wird natürlich die Qualität des Films, das sollten wir immer bedenken.

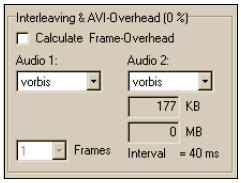


Abbildung 3.29

Schließlich wählen wir bei **Interleaving** (Abb. 3.29) 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 GKnot den Film erstellt. Der Wert gibt an, wie viele Kilobits 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.

> XviD, DivX 5 und x264 rechnen mit 1 Kilobit = 1000 Bits. DivX 3.11 dagegen verwendet die ungewöhnliche Regel 1 Kilobit = 1024 Bits.

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

3.4 Vorbereiten des Videos

Im **Resolution**-Register (Abb. 3.30) stellen wir im Abschnitt **Input Resolution** den Typ der DVD und die Art des Bildes ein. **PAL**-DVDs mit einer Auflösung von 720 × 576 Pixeln sind Standard in Europa, Japan und dem Nahen Osten, **NTSC** mit 720 × 480 Pixeln findet in Amerika Anwendung.
Zusätzlich müssen wir zwei Bildformate unterscheiden. Einmal **4:3**, das ist das Format eines ganz normalen Fernsehbildschirms. Kinofilme sind fast immer im schmalen **16:9** gedreht. Davon gibt es eine noch schmälere Abart, das 2,35 : 1. Zwischen den beiden Widescreen-Formaten brauchen wir aber nicht zu unterscheiden. 2,35 : 1 ist auf der DVD als 16:9 gespeichert. Der überschüssige Platz wird mit schwarzen Balken aufgefüllt.

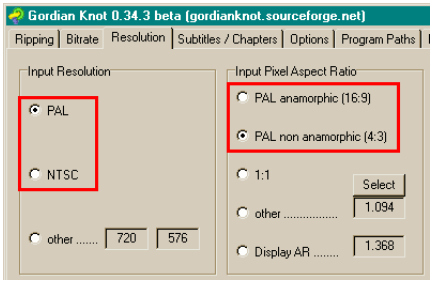


Abbildung 3.30

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 (Abb. 3.31) ist fürs Cropping zuständig, also für das Wegschneiden der schwarzen Ränder um das Video. Wer will, kann sich auf den **Auto Crop**-Mechanismus verlassen. Ich traue solchen Automatiken aber grundsätzlich nicht so recht über den Weg und mache das Cropping deshalb lieber selbst. Dafür stellen wir auf **Pixel**. Jetzt nehmen wir uns das Fenster mit dem Video vor, springen dort zu einer möglichst hellen Stelle und schneiden mit den Reglern 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 unten erklärte **Aspect Error 0** wird.

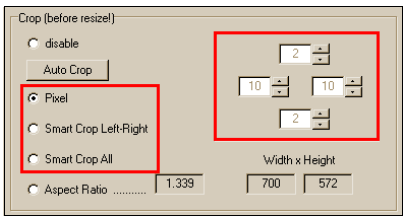


Abbildung 3.31

Anschließend müssen wir im unteren Teil des Fensters (Abb. 3.32) 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)** achten. Der **Aspect Error** gibt die Verzerrung des Bildes gegenüber dem Ursprungsformat an. Nicht über ± 2 % Fehler ist eine gute Einstellung, denn eine so geringe Verzerrung macht sich nicht nachteilig bemerkbar.
Das **Bits/(Pixel*Frame)**-Verhältnis (BPF) gibt an, wie viele Bits zum Encodieren

eines einzelnen Pixels durchschnittlich zur Verfügung stehen. Das heißt, dieser Wert ist ein (wenn auch ungenauer) Indikator für die Qualität des fertigen Films, da er alle direkt berechenbaren Einflussfaktoren berücksichtigt (Dateigröße, Filmlänge, Framerate, Auflösung). Allerdings sagt er wenig Definitives über die endgültig sichtbare Qualität aus. Ein Drama mit vielen langen Dialogen und hauptsächlich langsamen Szenen braucht für die gleiche sichtbare Qualität einen niedrigeren Wert als ein schneller Actionstreifen. Sehr nachtlastige Filme geben sich auch mit einem niedrigeren Wert zufrieden, da Szenen im hellen Tageslicht viel mehr Details enthalten und deshalb auch mehr Bits benötigen. Deshalb kann das BPF-Verhältnis immer nur ein Anhaltspunkt für die sichtbare Qualität sein.

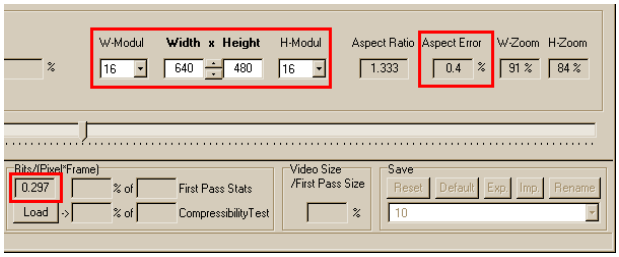


Abbildung 3.32

Der Wert sollte in etwa zwischen 0.20 und 0.30 liegen. Deutlich darunter äußert sich in schlechter Bildqualität, v. a. hässliche Macroblock-Artefakte werden dann sichtbar. Deutlich darüber erzeugt nur eine größere Datei bei kaum noch höherer Qualität. 1-CD-Rips orientieren sich in Richtung der 0.20 und 2-CD-Rips in Richtung 0.30. Nie vergessen sollte man den Richtwertcharakter des BPF-Werts. Ausreißer können durchaus vorkommen: z. B. ein Film, der sich schon mit 0,15 zufrieden gibt; oder ein anderer, der erst ab 0,35 keine sichtbaren Macroblocks mehr aufweist. Um solche Abweichungen in den Griff zu bekommen, lernen wir weiter unten mit dem Kompressionstest noch einen genaueren Qualitätsindikator kennen.

Über den großen Schieberegler müssen wir jetzt eine Auflösung wählen, die einen akzeptablen Aspect Error mit einem sinnvollen BPF-Wert verbindet. 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 einem höheren BPF-Wert, gleichzeitig aber auch zu einem Detail- und damit Qualitätsverlust. Um den nicht zu groß werden zu lassen, sollten wir Auflösungen unter 512 Pixel in der Horizontalen lieber vermeiden.

Ein Maximum ist einfacher definiert. Spätestens, wenn sowohl Höhe als auch Breite 100 % erreicht haben (sichtbar bei **`W-Zoom`** und **`H-Zoom`**), verliert eine weitere Steigerung ihren Sinn. Da immer die vertikale Auflösung der kleinere Wert ist, brauchen wir nur auf **`H-Zoom`** zu achten. Dieser Wert sollte 100 % nicht überschreiten. Da diese Regel für einen PAL-Film im 16:9-Format eine Auflösung von um die 1024 × 560 bedeutet, macht uns sehr wahrscheinlich der beschränkte Speicherplatz schon deutlich früher einen Strich durch die Rechnung. Zusätzlich braucht ein größeres Bild auch mehr

Rechenleistung beim Abspielen. Für die Realität heißt das, dass wir uns meistens in der Gegend von 640 Pixeln in der Horizontalen bewegen.

Nun 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 die **`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.

3.5 Dynamische Untertitel und Kapitel einbinden

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

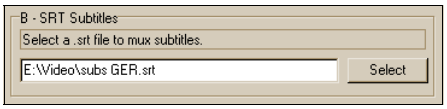


Abbildung 3.33

Unter **`SRT Subtitles`** (Abb. 3.33) 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.

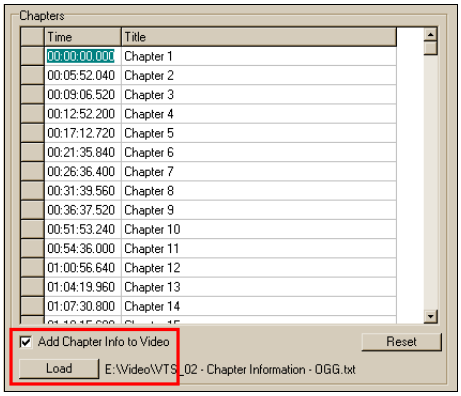


Abbildung 3.34

Für die Kapitel ist die rechte Seite des Registers zuständig (Abb. 3.34). 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.

3.6 Abschließende Arbeiten und Encoding

Filter konfigurieren

Im Video-Fenster (Abb. 3.35) müssen wir uns entscheiden, ob wir den Abspann mit anderen Einstellungen encoden 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.

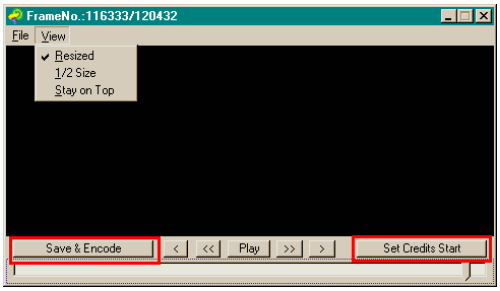


Abbildung 3.35

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`**. GKnot weiß jetzt, dass hier die Credits beginnen sollen.

Über den **`Save & Encode`**-Button gelangen wir anschließend zum **`Save .avs`**-Fenster in Abb. 3.36, 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.

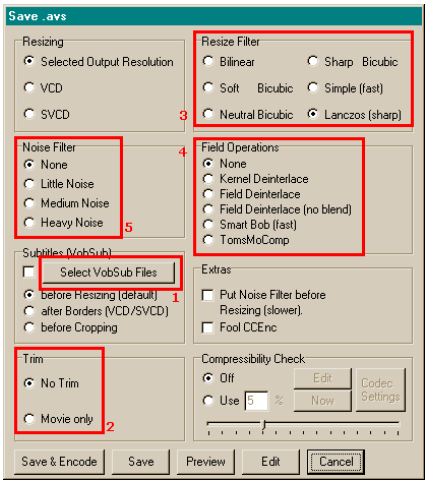


Abbildung 3.36

`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. Gordian Knot codiert deshalb Film und Credits getrennt und fügt die Dateien dann zusammen. **`Both`** ist dafür die richtige Einstellung (Abb. 3.37). **`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-Rips bietet sich ein weicher zeichnender Filter an, für 2-CD-Rips eher ein schärferer. Lanczos ist dabei sehr beliebt.

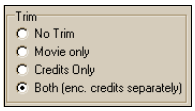


Abbildung 3.37

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 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.

Zu guter Letzt können wir noch einen **Compressibility Check** durchführen, der uns genauer als das BPF-Verhältnis Auskunft über die Qualität gibt (Abb. 3.38). GKnot 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.

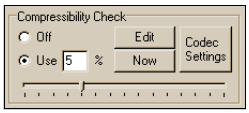


Abbildung 3.38

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

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

Und XviD sieht so aus:

- **Encoding type** auf **Single pass** stellen.
- **Target quantizer** auf **2.00** setzen.
- Den Rest so einstellen wie fürs Encoding des ganzen Films geplant.

Mit einem Klick auf **Now** öffnet sich VirtualDubMod und führt den Test durch. GKnot ist solange blockiert. Hinterher landen wir auf dem Register **Resolution**. Der Bereich unten in der Mitte sieht jetzt etwa aus wie in Abb. 3.39. 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.

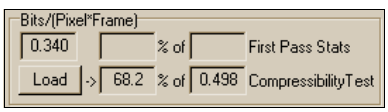


Abbildung 3.39

Bei zu wenigen Prozenten (unter 50 % dürfte es so richtig kritisch werden) müssen wir entweder Platz schaffen (kleinere Audiospuren, mehr CDs) 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. Im günstigsten Fall lässt sich sogar eine ganze CD streichen.

> 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 Auflösungs-Änderung einen neuen Kompressionscheck durchführen. Da aber auch der Compcheck nur ein Schätzwert für die Qualität ist, fällt diese zusätzliche Ungenauigkeit kaum ins Gewicht.

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.

Audio- und Video-Setup vervollständigen

Mit **`Select`** in Abb. 3.40 wählen wir die erste Audiospur aus. GKnot 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.

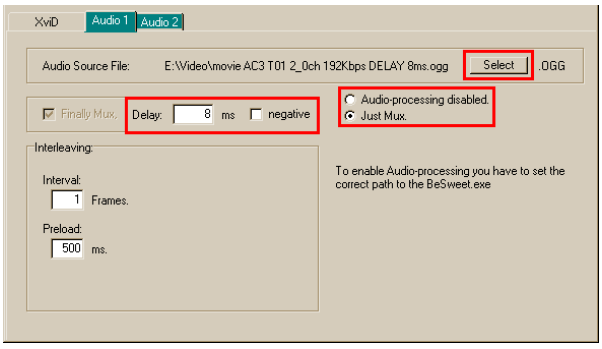


Abbildung 3.40

Bei allen anderen Audioformaten stellen wir auf **`Just mux`**, das Transcoding hat ja BeSweet schon erledigt. 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. Achtung: Wer das **`Delay`** schon bei BeSweet berücksichtigt hat, trägt bei **`Delay`** immer **`0`** ein. Ansonsten übernehmen wir den ms-Wert und setzen für negative den **`negative`**-Haken. Im Textfeld dann kein Minuszeichen angeben!
Wenn wir eine zweite Audiospur haben, wechseln wir ins **`Audio 2`**-Register und wiederholen dort die Einstellungen für die zweite Spur.

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 gibt es wenig zu tun. Das gilt auch für x264, denn dessen Dialog unterscheidet sich nicht wesentlich von XviD (Abb. 3.41). 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.

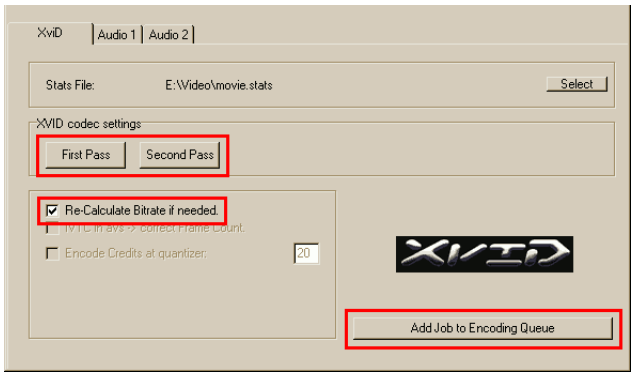


Abbildung 3.41

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.

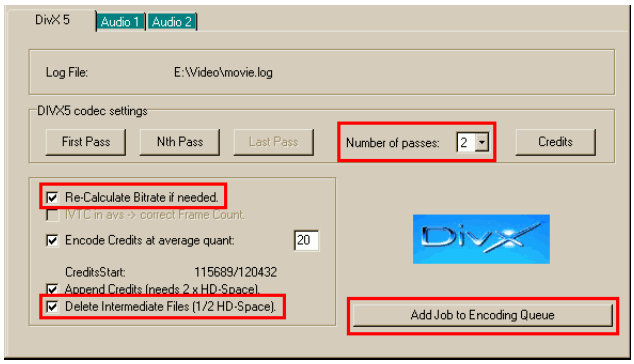


Abbildung 3.42

Für DivX sieht der Dialog ähnlich aus (Abb. 3.42). 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.

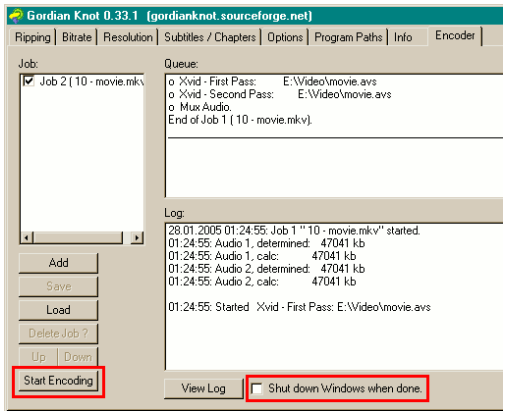


Abbildung 3.43

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

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 GKnot ein und landen anschließend im Register **Encoder** (Abb. 3.43). Wenn wir GKnots Nachfrage nicht schon bestätigt haben, starten wir über **Start Encoding** den Codiervorgang. Im **Queue**-Fenster sehen wir unseren aktuellen Auf-

trag, darunter bei **Log** sämtliche Details dazu. GKnot ruft jetzt VirtualDubMod auf, um den Film zu encoden 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. Jedenfalls reicht es noch zum Ausspannen vor den Abschlussarbeiten: den Film muxen, schnippeln und brennen.

Teil 4

Muxing & Splitting

4.1 Codierten Film splitten

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 mal abgesehen. Trotzdem: nicht jeder kann oder will sich einen DVD-Brenner leisten oder ist mit der Qualität eines 1-CD-Rips zufrieden. Deshalb bleibt das Splitting ein wichtiger Teil der Abschlussarbeiten.

2-CD-Rips 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.

> Die AAC-Fans sollten sich erst einmal mit den nächsten beiden Kapiteln beschäftigen und dann wieder hierher zurück kommen.

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. 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 2 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!

Zuständig für das Aufteilen ist entweder komplett VirtualDubMod oder – bei dynamischen VobSubs oder AAC-Audio – AVI-Mux GUI bzw. mkvtoolnix (dazu in den nächsten Kapiteln mehr). Im letzteren Fall suchen wir mit VirtualDubMod nur eine passende Stelle zum Schneiden.

Haben wir x264 verwendet, muss ein passender Decoder installiert sein, denn der x264-Download enthält nur den Encoder. Eine aktuelle Version von ffdshow eignet sich dazu bestens.

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.
Mit den **Keyframe-Buttons** (2) in Abb. 4.1 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 mkvtoolnix 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.

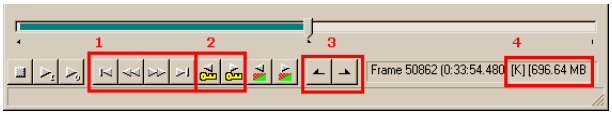


Abbildung 4.1

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. Diese 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 **Auswahl-Anfang-Marker**. Ein Sprung zum Filmende mit dem rechten Button von (1) und hier den **Auswahl-Ende-Marker** setzen. Dann können wir die zweite CD speichern (mehr CDs brauchen natürlich entsprechend mehr Schnitte).
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.

4.2 Muxing und Splitting mit mkvmerge

Mkvtoolnix ist ein Toolpaket für den Matroska-Container. Für uns ist das interessant, wenn wir uns für Matroska und AAC-Audio entschieden haben, denn VirtualDub-Mod 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.

Muxing

Das Programm aus mkvtoolnix, um alles zusammenzufügen, heißt mkvmerge und ist genauso wie BeSweet eine Konsolenanwendung. Die grafische Oberfläche dazu nennt sich mmg (mkvmerge GUI), und die starten wir.

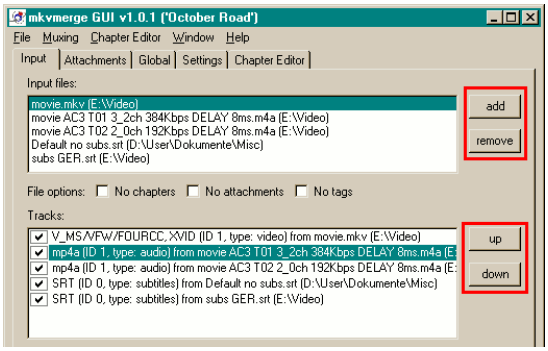


Abbildung 4.2

Mit **Add** oder per Drag&Drop können wir alle benötigten Dateien hinzufügen. In Abb. 4.2 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 konfigurieren wir die einzelnen Spuren, und zwar immer gerade diejenige, die im **Tracks**-Fenster angewählt ist.
Unter **Language** in Abb. 4.3 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 Millise-

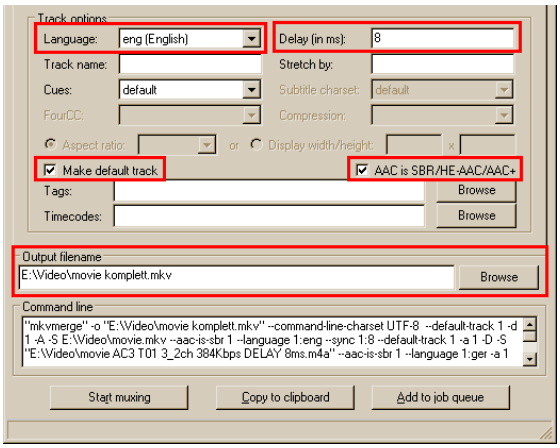


Abbildung 4.3

kunden 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`**.

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 und wechseln dann ins Register **`Global`** (Abb. 4.4).

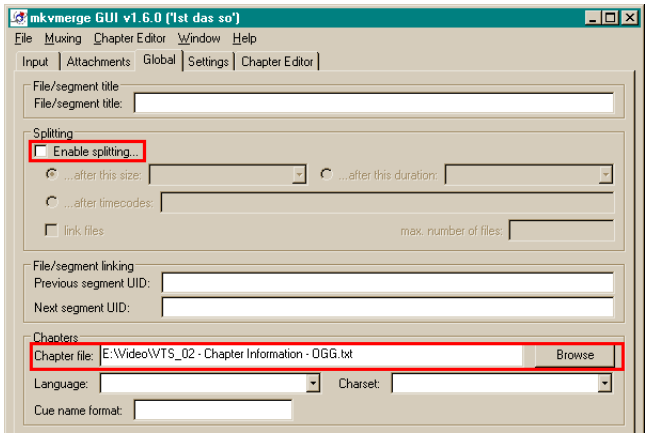


Abbildung 4.4

`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. mmg können wir solange offen lassen.

Splitting

Den fertigen Film laden wir in VirtualDubMod und suchen wie auf Seite 77 beschrieben einen passenden Punkt zum Schneiden heraus. Das Schneiden selbst muss dann mkvmerge übernehmen.

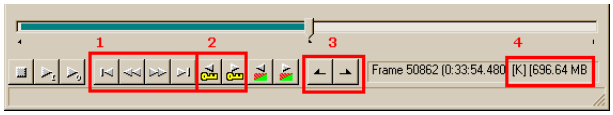


Abbildung 4.5

Inzwischen unterstützt auch mkvmerge das Schneiden an einem

exakten Zeitpunkt. Wir merken uns deshalb einfach den Zeitpunkt, an dem wir splitten wollen (im Beispiel **00:33:54.480**) und tragen diesen wie in Abb. 4.6 in mmg ein.

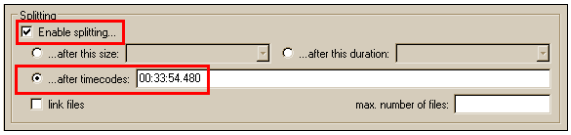


Abbildung 4.6

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, und 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**.

4.3 AAC im OGM- oder 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 schmäler aus. Ich bin nun mal überzeugter Matroska-User und habe daher mit OGM und AVI nicht mehr besonders viel zu tun. Eines vorne weg: AAC lässt sich in beide dieser Container muxen. Kein Problem bei OGM, AAC in AVI dagegen 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.

AAC in OggMedia

Wie kriegt man also AAC in OGM? Für LC-AAC reicht Ogg Mux Nic, für HE-AAC muss man den Weg über die 3ivx-Filter und Graphedit gehen. Das funktioniert auch für LC.

AAC in AVI

AAC in AVI kann nur ein einziges Programm: AVI-Mux GUI. Das versteht leider die oft von BeSweet erzeugten MP4-AACs nicht, sondern möchte sie gern im ADTS-Format haben. Eine Umwandlung ist mit Ivan & Menno möglich. Erinnern wir uns aber ans Audiokapitel (Seite 36): Ausschließlich MP4-AACs erzeugen die Encoder von Nero und 3GP, ausschließlich hüllenlose AACs erzeugt der Winamp-Encoder, und FAAC beherrscht beides.

Ein guter Einstiegspunkt für weitere Infos und Links ist die Doom9.org-Audio-FAQ.

Teil 5

Spezialthemen

5.1 Anamorphes Encoding

Was heißt anamorph?

Das beste Beispiel für anamorphes Video ist die DVD. Das dort gespeicherte Bild hat für PAL immer die Abmessungen 720 × 576 Pixel. Das entspricht einem Seitenverhältnis von 720 : 576 = 1,25 = 5:4. Nun besitzt der Film aber eigentlich ein Seitenverhältnis von 16:9 (für 4:3 siehe unten). Das Video ist auf der DVD verzerrt gespeichert: anamorph.
Ursprünglich war der Begriff anamorph nur für eine genau definierte Art der verzerrten Speicherung vorgesehen, was in der professionellen Videotechnik auch nach wie vor so gehandhabt wird. Deshalb nennt Gordian Knot im **`Resolution`**-Register 4:3-Filme **`non anamorphic`**, weil sie zwar nicht im korrekten Seitenverhältnis auf der DVD liegen, andererseits aber auch nicht unter die »professionelle« Definition von »anamorph« fallen. Die trifft nur auf 16:9-DVDs zu.
Da dieser enge Rahmen für die volldigitale MPEG-4-Welt kaum von Bedeutung ist, verwendet das Encodingwissen eine weiter gefasst Definition für das encodierte Video. Anamorph heißt hier lediglich *nicht im korrekten Wiedergabe-Seitenverhältnis, also verzerrt, gespeichert*. Wie die Verzerrung genau aussieht, bleibt offen und deshalb frei wählbar.

Einsatzgebiete

Dass anamorphes MPEG-4 immer beliebter wird, obwohl es einige Tücken birgt, hat einen einfachen Grund: Bildqualität. Sehen wir uns als kleines Beispiel einen DVD-Film im Seitenverhältnis (Aspect Ratio, kurz AR) 2,35 : 1 an, z. B. die Lord-of-the-Rings-Trilogie. Ursprüngliche Bildabmessungen sind DVD-konforme 720 × 576. Nach dem Wegschneiden der schwarzen Balken sollten noch 720 × 432 Pixel übrig bleiben, was einem AR von 1,67 : 1 entspricht. Das bedeutet heftige Eierköpfe beim Anschauen. Um das zu beheben, sind zwei Möglichkeiten denkbar.

- **Horizontal Strecken.** Um auf das korrekte AR von 2,35 : 1 zu kommen, müssen wir das Bild auf 1024 × 432 Pixel in die Breite ziehen, womit wir allerdings eine beachtliche Anzahl Pixel pro Bild – (1024 – 720) × 432 = 131.328 – encodieren müssen, die im Originalbild gar nicht vorhanden waren und deshalb keine echten Informationen tragen.
- **Vertikal stauchen.** Das ist die klassische Methode, die seit DivX 3.11 überwiegend verwendet wird. Wir schrumpfen also die Auflösung auf 720 × 304. Das passt schon eher für ein 2-CD-Encoding, allerdings um den Preis einer um 30 % niedrigeren vertikalen Auflösung.

Beide Methoden haben ihre Nachteile. Das nach dem Strecken sehr große Bild beansprucht die CPU beim Abspielen nicht gerade wenig. Ältere Rechner stoßen da schnell an ihre Grenzen. Zum anderen sind diese Abmessungen selbst für ein 2-CD-Encoding schlicht zu groß, um gute Qualität zu erreichen. Auch Stauchen ist nicht optimal, denn dummerweise ist das menschliche Auge gerade für die vertikale Auflösung empfindlicher als für die horizontale. Die Standardmethode 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 dürften größtenteils außen vor bleiben.

Als Haupteinsatzgebiet empfiehlt sich klar das hochqualitative Encoding, das die volle Auflösung (evtl. bis auf die schwarzen Balken) einer 16:9-DVD beibehält.
Für 4:3-Material ist ein anamorphes Bild weniger sinnvoll, da sich ein korrekt entzerrtes 4:3-Bild von 720×576 auf entweder 768×576 oder 720×544 verändert. Das bedeutet im Unterschied zur Originalauflösung keine große Veränderung und deshalb kaum Vorteile für das anamorphe Bild.
Auch 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 16:9-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.

Alle drei Varianten können wir mit Gordian Knot verwirklichen. Der Weg von der DVD zum MPEG-4-Video bleibt dabei größtenteils gleich. Nur bei der Berechnung der Zielauflösung und der Codec-Konfiguration ergeben sich Abweichungen. Dazu mehr im nächsten Kapitel.

Display Aspect Ratio und Pixel Aspect Ratio

Uns stehen zwei – grundsätzlich gleichwertige – Möglichkeiten zur Verfügung, um ein Seitenverhältnis anzugeben.

- **Display Aspect Ratio (DAR).** Beschreibt das Seitenverhältnis des kompletten Bildes, 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 Pixel.

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, 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 und Resizing, das wir in diesem Kapitel verwenden, 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, Video hat immer ein PAR von 1:1 und ein DAR von »Bildbreite zu Bildhöhe«.

Sehen wir uns das an einer kleinen Grafik (Abb. 5.1) 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.

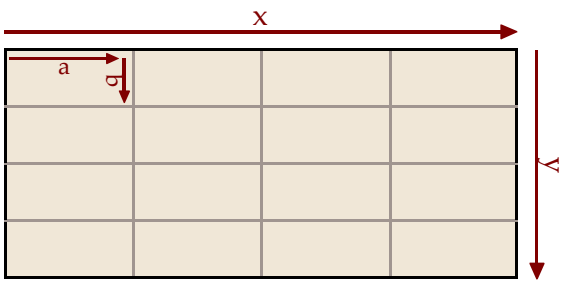


Abbildung 5.1

Die Form des gesamten Bilds und die Form eines einzelnen Pixels lässt sich leicht berechnen:
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 (Abb. 5.2).

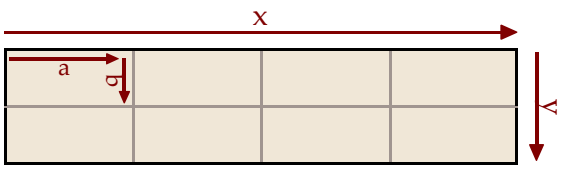


Abbildung 5.2

Es gilt noch immer wie oben, allerdings mit kleinerem *y*:
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.
Die zwei Grafiken spiegeln die Realität übrigens nur unvollständig wider, denn ein Computermonitor hat immer quadratische Pixel. Deshalb muss für die eigentliche Wiedergabe ein rechteckiges DVD-Pixel natürlich in mehrere quadratische Computerpixel umgerechnet werden, was das Beispiel der 16:9-DVD zeigt.

> Warum diese komischen Zahlen im Beispiel? Sollte es statt 1,82 und 2,42 nicht 1,78 (echtes 16:9) und 2,35 heißen? Das ist richtig, solange wir uns in der analogen Welt des Fernsehers bewegen. Für eine korrekte Darstellung am Computerbildschirm sollten wir aber nach dem Standard ITU-R BT.601 arbeiten, was im Vergleich zu den gewohnten Werten ein etwas breiteres Bild ergibt.

5.1.1 Video vorbereiten

Cropping und evtl. Resizing

Gordian Knot unterstützt anamorphes Encoding nicht ausdrücklich, weshalb wir um ein wenig Tricksen nicht herumkommen. Wir laden wie gewohnt den Film und kalkulieren die Bitrate. Dann wechseln wir ins **`Resolution`**-Register und stellen die pas-

sende **Input Resolution** ein. Soweit unterscheidet sich das Vorgehen nicht vom normalen Ablauf. Die **Input Pixel Aspect Ratio** setzen wir jetzt allerdings auf **1:1** und erledigen dann das Cropping.
Das zugeschnittene Bild sollte wie beim nicht-anamorphen Encoding ein durch 16 teilbares Seitenverhältnis haben, um dem Codec das Encodieren nicht unnötig zu erschweren. Mit etwas Glück passt das nötige Cropping auch zu dieser mod16-Anforderung. Oft klappt das aber nicht so gut und wir stehen vor der Entscheidung, entweder recht viele zusätzliche Pixel vom Bild wegzuschneiden, um auf die kleinere mod16-Auflösung zu kommen, oder für die größere mod16-Auflösung einen Rest schwarzer Balken in Kauf zu nehmen (gilt nur für Encodings ohne Resizing). Beides stellt nicht gerade die Ideallösung dar. Wir entscheiden uns dafür, etwas Schwarz stehen zu lassen, denn der AviSynth-Filter FillMargins kann das leicht korrigieren. Wie viele schwarze Pixelreihen an den einzelnen Rändern übrig bleiben, merken wir uns.

Ob die zugeschnittene Auflösung wirklich mod16 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!

Verwenden wir trotz anamorphem Bild Resizing, brauchen wir uns um mod16 nicht zu kümmern, denn das erledigt Gordian Knot. Wir stellen dann wie beim nicht-anamorphen Vorgehen mit dem großen Schieberegler die gewünschte Auflösung ein. Wichtig ist, dass auch hier die **Input Pixel Aspect Ratio** immer auf **1:1** stehen bleibt.

AviSynth-Skript bearbeiten

Dieses Kapitel ist nur von Bedeutung, wenn wir kein Resizing verwenden. Ansonsten können wir das AviSynth-Skript wie gewohnt unverändert abspeichern und zum nächsten Kapitel übergehen.

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 gewohnt 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.
Im **Plugins**-Abschnitt fügen wir wie in Abb. 5.3 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, und zwar in der Reihenfolge links, oben, rechts, unten. Für jeden Rand tragen wir ein, wie viele schwarze Pixelreihen übertüncht werden sollen. Damit ist das FillMargins-Plugin fertig konfiguriert.

```
#  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 5.3

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.

5.1.2 Encoding und Wiedergabe

Prinzipiell unterscheidet sich das Encoding anamorphen Materials kaum vom nichtanamorphen Vorgehen. Lediglich einige Details müssen wir beachten; hauptsächlich, um den Film im richtigen Seitenverhältnis abspielen zu können.

Kompressionstest

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.

AR-Flag setzen

Natürlich brauchen wir eine Möglichkeit, dem Decoder mitzuteilen, mit welchem Seitenverhältnis er das Bild wiedergeben soll. Dafür speichern wir in der Videodatei ein AR-Flag, das den korrekten Wert enthält. Dieses Flag kann auf zwei Ebenen gesetzt werden: entweder im Header der MPEG-4-Videospur selbst oder im Container. Wer auf Nummer sicher gehen will, kann auch beide Möglichkeiten nutzen. Allerdings unterstützen nicht alle Container ein AR-Flag. Nur Matroska bietet diese Möglichkeit, AVI und OggMedia dagegen nicht.

Kümmern wir uns zuerst um das setzen des MPEG-4-Flags. Wer XviD als Encoder verwendet, hat es leicht, denn XviD bringt AR-Unterstützung mit. Um das Seitenverhältnis zu setzen, müssen wir nach dem Speichern der AVS-Datei im **`Encoding Control Panel`** die XviD-Konfiguration aufrufen (siehe Seite 73). Am besten nehmen wir die AR-Einstellung für beide Passes vor, um möglichen Problemen aus dem Weg zu gehen.

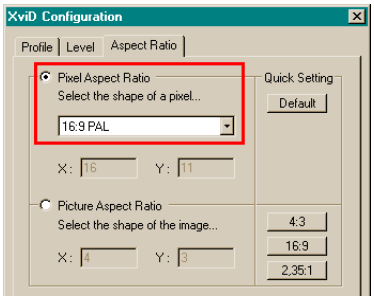


Abbildung 5.4

Über den ersten **`More`**-Button der XviD-Konfiguration (Abb. 5.4) öffnen wir die Details und wechseln ins Register **`Aspect Ratio`**. Dort setzen wir das **`Pixel Aspect Ratio`** (PAR) auf den richtigen Wert. Welcher das ist, hat uns das Statusfenster von DGIndex in den Feldern **`Aspect Ratio`** und **`Video Type`** verraten. 16:9 und PAL dürfte typisch sein. Entsprechend setzen wir jetzt das **`Pixel Aspect Ratio`** auf z. B. **`16:9 PAL`**.

Wir könnten auch das **`Picture Aspect Ratio`** (entspricht dem DAR) weiter unten im Fenster verwenden, was zum gleichen Ergebnis führt. Das richtige DAR können wir im **`Resolution`**-Register unter **`Aspect Ratio`** im Abschnitt **`Crop (before resize!)`** ablesen. Dafür müssen wir das **`Input Pixel Aspect Ratio`** vorübergehend auf den korrekten Wert für ein nicht-anamorphes Encoding setzen (also **`anamorphic (16:9)`** oder **`non anamorphic (4:3)`**). Benutzen wir Resizing, muss die gewünschte Zielauflösung schon gewählt sein.

Wer lieber mit DivX encodiert, kann das AR-Flag nicht gleich beim Encoding setzen, sondern muss das hinterher mit MPEG4 Modifier (*www.moitah.net/download/latest*) tun. Dieses Programm unterstützt bisher allerdings nur AVIs (kein OpenDML). Deshalb müssen wir entweder gleich Gordian Knot AVI verwenden lassen oder die Datei nach dem Encoding mit VirtualDubMod nach AVI konvertieren. Dazu laden wir den fertigen Film und speichern ihn über **`F7`** wieder ab. **`Save AVI in old 1.0 format`** muss angehakt sein, und ganz unten muss der **`Video mode`** auf **`Direct Stream copy`** stehen.

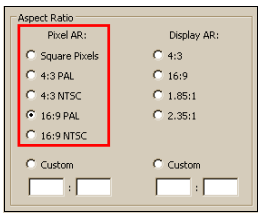


Abbildung 5.5

Dann öffnen wir die Datei mit MPEG4 Modifier und setzen das **Pixel AR** in Abb. 5.5 wie oben bei XviD beschrieben. Anschließen speichern wir das Video mit dem **Speichern**-Button. Da sich das AR-Flag im Header der Videospur befindet, kann die AVI problemlos nach Matroska oder OggMedia konvertiert werden, ohne die AR-Info zu verlieren.

Auch x264 bietet die Möglichkeit, ein AR-Flag zu setzen; und zwar für das PAR. Die Funktion arbeitet genauso wie bei XviD, setzt also das MPEG-4-AR-Flag in der Videospur auf das angegebene Verhältnis. Da x264 keine Standardwerte vorgibt, hier die Tabellen mit den nötigen Werten.

mit ITU-R BT.601

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

ohne ITU-R BT.601

	PAL	NTSC
4 : 3	128 : 117	72 : 79
16 : 9	512 : 351	96 : 79

Eingeben müssen wir den passenden Wert im **More**-Register der x264-Konfiguration (Abb. 5.6). Das Feld nennt sich **Sample AR** und verlangt den PAR. Also hat hier auch ein DAR nichts verloren.

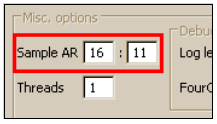


Abbildung 5.6

Je nach Quell-DVD entnehmen wir den passenden Wert aus den obigen Tabellen. ITU-R BT.601 sollten wir berücksichtigen, wenn wir den Film hauptsächlich digital (also in der Regel am Computer) abspielen wollen. Die PARs ohne ITU-R BT.601 sind interessant, wenn beim Anschauen hauptsächlich der Fernseher zum Einsatz kommt (s. auch die Anmerkung auf S. 88).

Im Matroska-Container setzen wir das AR-Flag beim Muxen mit mkvmerge GUI. VirtualDubMods Matroska-Unterstützung geht leider nicht so weit.

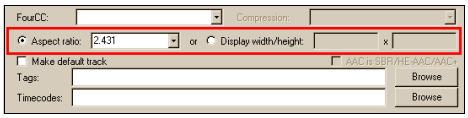


Abbildung 5.7

In den **Track options** der Videospur (Abb. 5.7) setzen wir unter **Aspect Ratio** das Seitenverhältnis, und zwar das DAR. Ein PAR lässt sich in mkvmerge GUI nicht angeben. Wir tragen also den mit Gordian Knot berechneten Wert hier ein. Alternativ können wir auch unter **Display width/height** eine Wiedergabeauflösung angeben. Das Ergebnis ist das selbe.

Wenn das Video schon ein MPEG-4-AR-Flag enthält, brauchen wir in mkvmerge GUI nichts einstellen, denn mkvmerge übernimmt das MPEG-4-AR automatisch für das Matroska-AR-Flag.

Wiedergabe anamorpher MPEG-4-Videos

Das Setzen des AR-Flags reicht noch nicht aus, um das Video mit dem richtigen Seitenverhältnis wiederzugeben. Der Decoder muss das Flag auch lesen und richtig umsetzen können. Die folgende Tabelle zeigt, welcher Decoder welche Art Flag erkennt. Getestet wurde in Graphedit. Welcher Container oder welcher Matroskasplitter verwendet wird, wirkt sich nicht aus.

	MPEG-4-AR	**Matroska-AR**
ffdshow 2005-03-12	Ja (siehe unten)	Ja (siehe unten)
XviD 1.1 Beta 2	Ja	Ja
DivX 5.2.1	Nein	Nein
Nero 6.3.1.20	Ja	Nein

ffdshow erkennt zwar sowohl MPEG-4- als auch Matroska-Flags, allerdings nur mit bestimmten und jeweils verschiedenen Einstellungen auf der **`Output`**-Seite des Konfigurationsdialogs. Zwingend nötig ist der Haken bei **`Use overlay mixer`**, sonst erkennt ffdshow keines der AR-Flags. Haben wir zusätzlich noch **`Allow output format changes...`** angehakt, erkennt ffdshow das MPEG-4-Flag, nicht jedoch das Matroska-Flag. Ohne den Haken ist es genau umgekehrt.
XviD und Nero benötigen keine besonderen Einstellungen. Für XviD lässt sich in der Decoder-Konfiguration lediglich bestimmen, welches der beiden Flags die höhere Priorität genießt, wenn beide vorhanden sind.

Um möglichst hohe Kompatibilität zu erreichen, sollten wir auf das MPEG-4-Flag nie verzichten und zusätzlich auch das Matroska-Flag setzen, wenn wir diesen Container verwenden. Mkvmerge übernimmt das MPEG-4-Flag ja bequemerweise automatisch.

Changelog

05.07.2004

Erste Veröffentlichung.

08.07.2004

- Unklarheiten bei der DivX-Konfig beseitigt (thx Hybrid).
- Guide als Zip-Archiv (Link auf der Startseite).
- Ein paar Links gefixt.

22.07.2004

- Untertitelkapitel überarbeitet.
- Rippingkapitel überarbeitet.
- Einige Screenshots aktualisiert.
- HE-AAC-Checkbutton in mmg erwähnt.

22.10.2004

- Abschnitt zum Kompressionstest aktualisiert.
- DivX-5.2.1-Konfig eingefügt.
- Layout: neues Format, das ohne Framesets auskommt.
- Ein paar Links gefixt.

02.12.2004

- Der Guide heißt jetzt »Brother Johns gesammeltes Encodingwissen«, da er über reines Gordian Knot doch schon ein Stück raus geht.
- Ein paar Links gefixt.
- Kurzen Abschnitt zum I-Frame-Intervall eingefügt (bei der XviD-Konfig).
- Audiokapitel umgebaut und um BeLight erweitert.
- Grundlagenkapitel eingefügt.

03.12.2004

PDF-Version online gestellt.

14.02.2005

- Komplett neues Layout incl. validem Code. War gewaltig viel Arbeit und ich finde, es hat sich gelohnt. Würde mich über euere Meinungen freuen!
- PDF-Layout ans neue Weblayout angepasst.
- Screenshots erneuert.
- Gordian Knot kann x264 und das Encodingwissen jetzt auch. :-)

- XviD-Kapitel für Version 1.1 aktualisiert.
- Die Kapitel zu älteren Codec-Versionen bleiben online im »Codec-Archiv«.
- Kapitel zu BeSweetGUI und BeLight aktualisiert.
- Wie so oft ein paar Links angepasst.
- Massig andere Kleinigkeiten, an die ich mich beim besten Willen nicht mehr erinnern kann.

19.04.2005

- Neues Spezialkapitel zum anamorphen Encoding.
- Abkürzungsverzeichnis erstellt.
- DGIndex: Erklärung zum VOBs laden ans neue Öffnen-Fenster angepasst.
- HTML-Navigation kann auf- und zugeklappt werden (benötigt JavaScript).
- Abbildungen in der PDF nummeriert.
- Und dann war da noch die übliche Linkpflege.

13.06.2005

- Probleme mit der klappbaren Navigation behoben. Das Script wurde lästigerweise erst nach dem Laden aller Grafiken ausgeführt. Der Fix ist schon seit einer Weile still und heimlich online.
- Wegen der jüngsten Ereignisse um LIGHTNING UK! Umbau des Ripping-Kapitels. Beschwerden und sonstige Unmutsäußerungen schickt ihr bitte direkt an die Medienindustrie!

16.06.2005

- Ab sofort steht das Encodingwissen unter einer der Creative-Commons-Lizenzen (Details auf dem Titelblatt). Viel ändert sich dadurch nicht, lediglich ein kommerzielles Weiterverwenden ist nun ausdrücklich untersagt.
- Update des x264-Kapitels incl. Anpassungen im Anamorph-Kapitel.
- DivX 6 ist raus, allerdings noch nicht mit GKnot verwendbar. Deswegen findet sich das Kapitel dazu vorerst nur im Codec-Archiv.

23.07.2005

- Neues Grundlagenkapitel: deutlich ausführlicher als bisher.
- Empfehlung für XviDs B-Frame-Einstellungen auf Didées bevorzugte Werte angepasst. Siehe dieses Posting auf doom9.org:
  http://forum.doom9.org/showthread.php?p=676111#post676111
- DivX 6.0: Auf nach wie vor bestehende Probleme mit der MPEG-Matrix hingewiesen.

24.10.2005

- Installation von BeSweet incl. der nötigen DLLs ausführlicher erklärt.
- Insgesamt das Chaos im ganzen Audiokapitel ausgemistet und komplett auf BeLight umgestellt.
- X264-Kapitel auf die neue GUI angepasst.
- Matroska-Muxing-Kapitel auf aktuelle mkvmerge GUI angepasst.