Wissenswertes rund um Xvid 0.2.6

(by Selur)

Vorwort:

An alle Anfänger:

Dieser Abschnitt ist vorallem an alle User die keine Ahnung haben und eigentlich auch nur ihre DVD einfach und schnell nach Mpeg4 umwandeln wollten damit sie auf ihrem Mpeg4Player laufen.

Da es in viele User gibt, die sich nicht wirklich interessieren was die einzelnen Xvid Features machen gibt es einfache Einsteigertool wie AutoGK (http://www.autogk.net/) , welche nur minimale Interaktion mit dem User erwarten jedoch gute Ergebisse bei den meisten Quellen geben.

Will man speziellere/professionellere/bessere Dateien erstellen sollte man sich z.B. mit Tool wie Gordian Knot (http://gordianknot.sourceforge.net/) und DVDtoOgm/DVDtoMkv (http://dvdtoogm.divx-digest.com/) und anderen Tools beschäftigen die dem Benutzer mehr Freiheiten und Verantwortung geben.

Wieso ist dieses Dokument entstanden?

Da mir leider aufgefallen ist, dass es an einer Hilfe bzw. Wissenssammlung rund um Xvid mangelte, habe ich mich aufgerafft und dieses Dokument geschrieben. Wichtig anzumerken sei, dass viel von dem was ich hier niedergeschrieben habe, auch auf den Erfahrungen anderer User basiert und ich diesen hiermit danken möchte. Vor allem geht mein Dank an das Flaskmpeg & DVDtoOgm Team und an die Doom9&Gleitz Crew. :)

Was ist Xvid?

Xvid ist ein Open Source Mpeg4-Codec der nur für "educational purposes" gedacht ist.
Eine offizielle Homepage von Xvid gibt es auch: http://www.xvid.org. Dort findet man jedoch nur den SourceCode, keine kompilierten Versionen (=.exe Dateien bzw. Installer&Co). Würde Xvid.org einen kompilierten Code veröffentlichen, müssten sie Mpeg4-Lizensgebühren zahlen, was für ein OpenSource-Project nicht machbar ist.

Wie sieht das mit dem Download aus?

Glücklicherweise haben sich jedoch einige Leute gefunden ,die sich in die Grauzone des Copyrights begeben und kompilierte Versionen auf ihren Homepages zum Download anbieten.
Da wären:

Koepi auf http://www.koepi.org/
und
CelticDruid auf: http://celticdruid.no-ip.com/xvid/

Welche Binaries sind die aktuellsten?

Meist findet man die aktuellsten kompilierten Versionen bei Celticdruid. CelticDruid und Koepi bietet sowohl die aktuelle sich in der Entwicklung befindliche 1.1er Versionen als auch die 'sicheren' 1.0er an. Celticdruid bietet immer die aktuellsten buils, hingeben Koepi bietet nur Versionen an die als 'stabiler' gelten, jedoch einige neuen Features eventuell noch nicht besitzen.

Persönlich bevorzuge ich die 1.1er builds des Codecs von Celticdruid, da sie schneller sind und teilweise neue Features haben, da sie aber durchaus auch Fehler/Bugs beinhalten können muss jeder selber entscheiden mit welcher Version er arbeiten will.

Feedback:

Da ich immer versuche das Dokument bei Neuerungen auf dem aktuellen Stand zu halten, freue ich mich besonders über Feedback, also Kommentare, Kritik und Verbesserungsvorschläge. Durch die zunehmende Anzahl von Spam, die ich täglich bekomme, würde ich mich freuen, wenn ihr mir das Feedback möglichst im offiziellen deutschen Flaskmpeg&DVDtoOgm Board (http://www.flaskmpeg.info/board/) oder im Gleitz&Doom9 Board (http://forum.gleitz.info/) zuteil werden lassen würdet.

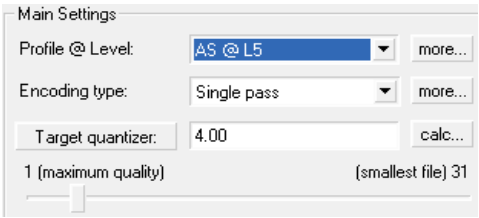
Auch Rechtschreib- und Grammatikkorrekturen werden gerne gesehen.

Bevor ich's vergesse:

Dieses Dokument wurde grundsätzlich nachts bzw. früh morgens nach längeren, Gedächtnislücken hervorrufenden Kneipenbesuchen unter dem Einfluss von (etwa 42) Aufputschmitteln, deren Verfallsdatum (1994) abgelaufen war und einigen weiteren 'Hilfsmitteln', die ich hier nicht aufführen darf (Strafverfolgung), dem Meer des Chaos entwendet. Auch wegen oben erläuterter Entstehungsgeschichte übernehme ich weder implizit noch explizit eine Garantie dafür, dass dieses Dokument einen wie auch immer gearteten (sinnvollen) Zweck erfüllt. Auch lehne ich jede Verantwortung oder Haftung bezogen auf etwaige durch dieses Dokument entstandene Schäden ab.

So jetzt aber zu den eigentlichen Einstellungen!

Main Settings:



In den Main Settings geht es, wie der Name schon sagt, darum, die wesentlichsten Einstellungen vorzunehmen. Meist ist es so, dass nach dem erstmaligen Festlegen sämtlicher Codec-Optionen bei der späteren Benutzung nur noch selten etwas anderes als die Main Settings geändert werden.

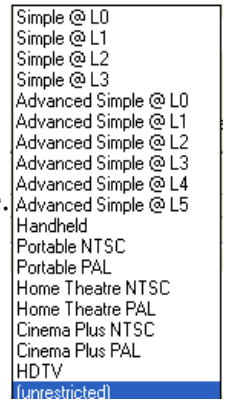
Zu diesen gehört die Wahl des verwendeten **Profiles,** des zu verwendenden **Encoding Types** und die Wahl der zugehörigen Datenrate bzw. angestrebten Qualität.

Profiles:

Bei den Profileinstellungen kann man auswählen, ob man sich an ein bestimmtes Mpeg4 Standard Profile halten will und welche Features genutzt werden sollen. Wählt per Profile@Level einfach ein bestimmtes Profile aus oder geht durch den 

- Button zu den erweiterten Profile Settings.

Profile @ Level:

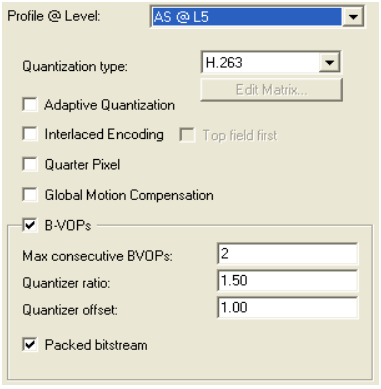


Durch die Wahl eines entsprechenden Profils wird festgelegt, ob die Features die man benutzen kann durch eines der Mpeg4 Profiles beschränkt werden sollen. Die Profiles, welche Xvid dem User anbietet sind einmal:

Vier Simple Profiles (0-3), vier ARTS Profiles und sechs Advanced Simple Profiles, sechs von DivX Networks erstellten Profiles und dem Unrestricted Profile. Letzteres besagt einfach, das man sich an keines der Standard Profiles halten will; dies ist sicher kein Problem solange man das encodete File nur an einem Rechner angucken will. Um aber mit etwaigen Stand-Alone-Playern konform zu sein, ist es eventuell nötig, sich an gewisse Profile zu halten. Je nachdem für welches Profile man sich entscheidet, wird die spätere Wahl an möglichen Features und Freiheiten des Codecs eingeschränkt. Befindet man sich in den erweiterten Profileinstellungen, ist das leicht zu erkennen. Denn sobald man das Profil wechselt, werden z.B. bestimmte Optionen aktiviert bzw. deaktiviert oder die maximale Auflösung begrenzt. Die Zahlen hinter den einzelnen Profiles legen Beschränkungen bzgl. der Datenrate und Auflösung fest, die beim Encoden verwendet werden dürfen.

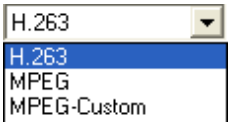
Für StandAlonePlayer ist oft das Simple@L3 zu empfehlen,da hier keine advanced Features verwendet werden. Neuere Player unterstützen aber meist außer Global Motion Estimation fast alles, weshalb ich dann AS@L5 empfehlen würde.

So viel zu den einfachen Profile Einstellungen; hat man sich für die erweiterten Profile Settings entschieden, was zu empfehlen ist, so erscheint nun folgendes Bild:



Als erstes muss natürlich das oben schon erwähnte Profile Level ausgewählt werden. Außerdem besteht nun die Möglichkeit die zu verwendenden Features genauer zu spezifizieren. Über die Reiter 'Level' und 'Aspect Ratio' gelangt man zu weiteren Profile Settings, wobei ich in diesem Dokument nicht weiter auf den 'Level' Reiter eingehen werde, da dieser rein informativ ist. Er zeigt an, welche Auflösung und Framerate im gewählten Profile maximal unterstützt werden, wie groß ein einzelnes Frame (Bild) werden darf und wie groß die maximale Datenrate und Buffergröße pro Sekunde sein müssen. Wichtig ist, dass sich Xvid in der 1.0 noch nicht an etwaige maximale Datenratenbeschränkungen hält, was bei StandAlonePlayern zu Problemen führen kann, wenn diese nicht leistungsfähig genug sind.
Bevor ich zum 'Aspect Ratio' Reiter komme, nun aber erstmal zu den Settings, die einem hier direkt zur Verfügung stehen. Als erstes haben wir da den Quantization Type:

Quantization Type:

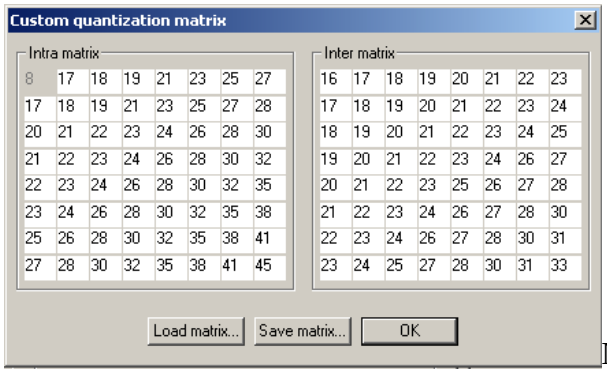


Der verwendete Quantization Type legt die zu verwendende Quantizer Matrix fest. Was es damit auf sich hat wird in ***Anhang I*** etwas ausführlicher erklärt. Hier werde ich nur kurz die wichtigen Informationen dem normalen User nennen.
Die **H.263** Matrix eignet sich vor allem für niedrige Datenraten, da sie, im Gegensatz zur Mpeg Matrix, das Bild eher glättet. Die **Mpeg** Matrix ist durch ihre kleineren Werte in der Matrix eher für hohe, detailreiche Quellen geeignet. **Mpeg-Custom** erlaubt es, dem wissenden und interessierten User eine eigene Matrix zu erstellen und diese zu verwenden. Das kann bei einzelnen Quellen bzw. bestimmten Szenarien zu einem besseren Bild und/oder besserer Kompression führen. Generell sind aber die H.263- und die Mpeg-Matrix zu empfehlen.

Persönlich bevorzuge ich zwar eigene/custom Matrizen, würde in der Regel aber dazu raten die Mpeg oder H.263 Matrix zu verwenden.

Sollte man sich für **Mpeg-Custom** entschieden haben, kommt man über 

zu folgendem Fenster:



Hier hat man nun die Möglichkeit eigene Quantization Matrizen zu definieren, zu speichern und zu laden. Was es mit Quantizer Matrizen grob auf sich hat, kann man im **Anhang I** nachlesen. Um enthusiastischen Bastlern eine kleine Hilfe zu geben, möchte ich hier auf den **Custom Quantization Matrix Editor** von LigH hinweisen, den man unter http://www.ligh.de/software/CQME_1.0a.zip downloaden kann.
Neben dem Editor bietet LigH auch eine ordentliche Sammlung an Matrizen zum Download unter http://www.ligh.de/software/qmatrix.zip an, die ich auch empfehlen will.

Adaptive Quantization aka. Lumi Masking:

Adaptive Quantization betrachtet die absolute Helligkeit einer Szene.
Aktiv wird es in sehr hellen oder dunklen Bildbereichen. In solchen Bildbereichen weist Adaptive Quantization Texturen, die nicht leicht mit den Augen zu erkennen sind, einen höheren Quantizer zu und versucht so Datenrate zu sparen. Es wird nicht mehr nur ein, sondern mehrere Quantizer für einen Block verwendet, die ein wenig vom durchschnittlichen Quantizer abweichen. Dieses Abweichen sollte es normalerweise ermöglichen, Datenrate einzusparen. Leider gibt es keine Garantie dafür, dass Xvid nicht auch mal in einer Szene eingreift in der ein höherer Quantizer sichtbare Qualitäteinbussen bringt, da dies aber recht selten passiert, kann man es ruhig aktivieren.

Vor allem, wenn man relativ niedrige Datenraten anstrebt, würde ich empfehlen, dieses Feature zu aktivieren.

Interlaced Encoding:

Diese Option ermöglicht es interlactes Quellmaterial so zu komprimieren, dass es auch als interlactes Material ausgegeben wird. Anzumerken sei hierbei, dass meines Wissens nur der Xvid eigene Decoderfilter solcherart encodetes Material ordentlich wiedergeben kann. Diese Option sollte wirklich nur dann verwendet werden, wenn das Filmmaterial interlaced ist und es auch in dieser Form gespeichert werden soll. Wurde vor der Enkodierung wieder ein progressives Bild (Vollbild) mittels eines Deinterlacers, wie z.B. TomsMoComp(), hergestellt ist 'Interlaced Encoding' auf jeden Fall zu deaktivieren.

Sollte nur verwendet werden, wenn man interlaced speichern will.

Top field first:

Wenn man interlactes Material abspeichert muss entweder erst die obere Bildkamm oder der untere Bildkamm eines Bildes gespeichert werden. Da die meist erst der untere Kamm gespeichert wird (bff = bottom field first) ist dies auch der Standardwert in Xvid. Will man jedoch als erstes den oberen Bildkamm speichern, so muss man die Option 'Top field first' aktivieren.

Sollte man nur verwenden, wenn man es sicher braucht.

Quarterpixel:

MPEG1 / MPEG2 speichern Bewegungen bis auf ein halbes Pixel (8x8 Bildpunkte) genau. Bei MPEG4 ist sogar eine Genauigkeit von einem viertel Pixel (4x4 Bildpunkte) erlaubt, welche man durch Quarterpixel aktiviert. Bei niedrigen Datenraten ist es ganz praktisch, um Detailverlusten durch zu hohen B-Frame Gebrauch etwas entgegen zu wirken. Bei niedrigeren Auflösungen erhöht es sogar die Komprimierbarkeit, da durch die Interpolation die für die Quarterpixel stattfindet, öfters auch Datenrate eingespart werden kann. Bei hohen Datenraten, bei denen die Detailstufe an sich schon relativ hoch ist, macht Quarterpixel das Bild noch mal einen Tick schärfer, bringt jedoch meist keine Datenrateneinsparungen, da mehr gleichfarbige Flächen existieren. Prinzipiell würde ich Quarterpixel empfehlen. Ein Wermutstropfen ist jedoch, dass beim Decoden 'Schlieren' im Bild auftreten können, wenn der Decoder nicht die gleiche IDCT wie der Encoder verwendet. Außerdem erzeugen mit Quarterpixel encodete Dateien mehr CPU-Auslastung beim Abspielen und Erstellen.

Sollte man normalerweise aktiviert haben.

Global Motion Compensation (GMC):

Man kann zwischen "Global Motion Compensation" mit einem, zwei, drei oder gar vier warp points unterscheiden. Wird nur ein warp point verwendet, so werden von GMC Verschiebungen erkannt. Werden zwei warp points verwendet, werden zusätzlich auch noch Rotationen und Größenänderungen erkannt. Werden wie in Xvid drei warp points verwendet, so werden auch geometrische Verzerrungen erkannt. Mit dem vierten warp point wären auch perspektivische Verzerrungen erkennbar, was jedoch für 2D Material nicht machbar ist. Erkennt GMC solche Änderungen, so werden diese speziell abgespeichert, um einiges an Datenrate einzusparen. Damit GMC effektiv auch etwas bringt, muss es möglichst viele Daten über den Film haben. Diese bekommt es, wenn man VHQ aktiviert. Zu beachten ist aber das GMC (beim Abspielen und Encoden) mehr CPU Leistung fordert, von vielen StandAlonePlayern wegen den 3 -Warp points (noch) nicht unterstützt wird und wenn man GMC aktiviert man immer sämtliche schwarzen Ränder entfernen sollte, da GMC sonst kontraproduktiv werden kann.

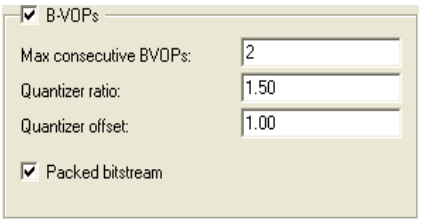
Nur Qualitätsfanatiker sollten es aktivieren und dann möglichst nur in Verbindung mit VHQ.

Reduced Resolution:

Hierbei handelt es sich um ein Feature das übermäßige Blockartefaktbildung verhindern soll. Ist die Datenrate für eine spezielle Szene zu niedrig, entstehen Blockartefakte, die das Bild unschön erscheinen lassen. Um diesen Komprimierungsartefakten entgegen zu wirken versucht 'Reduced Resolution' intern die horizontale und/oder vertikale Auflösung des Bildes/Frames zu senken. Somit wird weniger Datenrate benötigt, um das Bild in einer noch akzeptablen Qualität darstellen zu können. Zwar gehen so natürlich Details verloren, jedoch stört dies normalerweise weniger als die auffälligen Blockartefakte.
Da dieses Feature jedoch leider (momentan) nicht dynamisch in Xvid verwendet wird, d.h. die Auflösung wird wenn, dann nur im gesamten Film reduziert, würde ich von einer Verwendung abraten. Wichtig ist auch, dass man Reduced Resolution nicht verwenden kann, wenn man b-frames, q-pel oder gmc aktiviert. Diese Optionen gehören zu einem anderen Profil und vertragen sich nicht mit 'Reduced Resolution'.

Sollte man momentan nicht verwenden, da es nicht dynamisch arbeitet.

B-VOPs aka. B-Frames:



B-Frames betrachtet bei der Berechnung nicht nur vorangegangene, sondern auch nachfolgende Frames, was meist zu beachtlichen Datenrateneinsparungen führt. Die Standardeinstellungen betreffend B-Frames sollten nur geändert werden, wenn man weiß was man tut. Sie sind so gewählt, dass sie in den meisten (normalen) Szenarien gute Ergebnisse liefern.

max consecutive BVOPs/B-Frames:

Hier kann man die maximale Anzahl an B-Frames, die hintereinander folgen dürfen, einstellen. Der Standard ist 2. Wichtig ist, dass es wirklich nur ein maximaler Wert ist, der je nach *B-Frame sensivity* nicht erreicht werden muss bzw. erreicht wird. Wenn man sehr viele maximal aufeinander folgende B-Frames erlaubt, sollte man auch bedenken, dass all diese Frames beim En- und Decoden erst einmal in den Speicher geladen werden müssen, was je nach System den Encoding- und Decodingvorgang durchaus bremst.

Als für die meisten Szenarien sinniger Wert hat sich hier 2 herauskristallisiert.

B-frame quantizer ratio:

Bezeichnet das Größenverhältnis, um welchen Faktor der Quantizer eines B-Frames größer sein soll als der aufgerundete durchschnittliche Quantizer der Frames auf die das B-Frame sich bezieht.

Als sinniger Wert hat sich hier 1.50 herauskristallisiert.

Quantizer Offset:

Bezeichnet nochmal einen weiteren 'Bonus' der auf die neue Quantizergröße kommt.
B-Frame-Quantizer errechnen sich wie folgt:

Bquant = (QdV+QnV)/2 * qR + qO

QdV (= Quantizer des Vorgängers), QdN (= Quantizer des Nachfolgers), qR (= QuantizerRatio),
qO (= QuantizerOffset)
z.B.: Vorgänger Frame benutzt Quantizer 2.0 und das Nachfolgerframe Quantizer 3.0 und die B-Frame Settings wären 3.0/1.50/1.00, so ergäbe sich:
(((2.0+3.0)/2.0*1.50 = 3.75) + 1.00 = 4.75) = 4.75 als Quantizer für das B-Frame

Normalerweise sollte man nicht vom Standardwert von 1.0 abweichen.

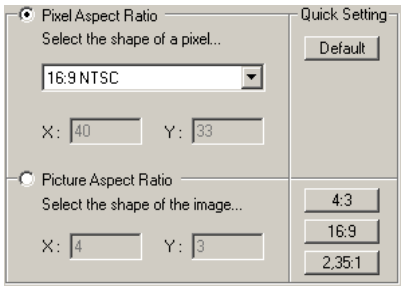
Packed bitstream:

Hierbei handelt es sich um eine etwas andere Art die Daten des Videostreams in Avi Dateien zu speichern. Die einzelne Frames werden in die Reihenfolge umsortiert in der sie decodiert (nicht abgespielt) werden. Dies erleichtert es manchen StandAlonePlayern besser mit B-Frames umzugehen, führt bei anderen aber zu Problemen. Mit aktiviertem Packed bitstream sind Dateien auch teilweise besser weiterzuverarbeiten, da sie keinen B-Frame Lag haben sollten, wenn man in ein Avi Datei speichert. Persönlich muss ich sagen, dass ich das Feature nie aktiviere, da ich auf all meinen Rechnern Xvid ohne Probleme abspielen kann.

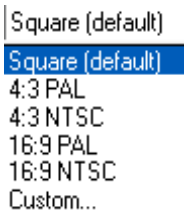
Sollte meiner Ansicht nach normalerweise deaktiviert sein.

So, nachdem jetzt die wesentlichen Profile Einstellungen geklärt sein sollten,
geht's nun zum ***'Aspect Ratio'***-Reiter, hinter dem sich die Möglichkeit verbirgt,
den Pixel Aspect Ratio oder den Picture Aspect Ratio festzulegen.

Aspect Ratio:



Das **Pixel Aspect Ratio (PAR)** gibt das Verhältnis von Breite zu Höhe **eines Pixels** (16x16 Bildpunkte) in einem Bild an. Das **Picture Aspect Ratio** besser Frame Aspect Ratio (FAR) gibt dieses Verhältnis für ein ganzes Bild bzw. einen gesammten Frame an. Mit den Aspect Ratios kann man also wie bei einer DVD anamorph encoden. Zu beachten ist aber, dass dies nur Sinn macht, wenn der verwendete Player auch anamorphe Dateien in der gewählten Hülle (z.B. avi, ogm, mkv, mp4, mov,..) unterstützt.
Sollte er dies nicht tun, wird man beim Abspielen ein verzerrtes Bild sehen, was natürlich nicht so wunderschön ist.

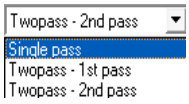


Beim PAR hat man nun einige Standardeinstellungen zur Auswahl, wobei man normalerweise bei der Standardeinstellung **Square** bleiben sollte, wenn man sich nicht sicher ist, was man tut oder für einen StandAlonePlayer encoded, da diese momentan meines Wissens nach noch keine Anamorphen Mpeg4 Dateien unterstützen. Persönlich encode ich eigentlich fast nur anamorph in eine Matroska-Datei oder ein Mpeg4-File.
Genaueres zum Erstellen der Daten in ein .mp4 oder .mkv file kann man in den oben erwähnten Foren per Suche und Fragen erfahren.

Okay, die Möglichkeiten die einem Xvid über die Profiles&Co liefert, sind jetzt hoffentlich einigermaßen verständlich beim Leser angekommen, weshalb ich nun zum Encoding Type und den hinter dem

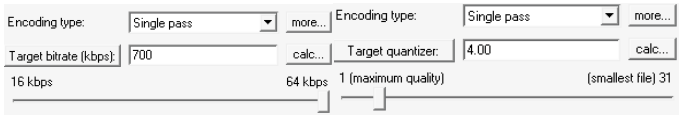
- Button versteckten erweiterten Einstellungen übergehe. :)

Encoding Type:



Bei den Encoding Types unterscheidet man erst einmal zwischen **Single pass** und **Two pass** Encodingverfahren. Worin diese sich unterscheiden bzw. wofür sie gedacht sind, folgt nun.

Single pass – Verfahren:



Single Pass Verfahren sind vor allem für Live-Aufnahmen, wie beim Capturen von analogen Quellen gedacht.

Beim **Single pass** Verfahren encoded Xvid entweder immer mit einem festen Quantizer, oder einer vorgegebenen durchschnittlichen Datenrate, je nachdem ob man 'Target quantizer' oder 'Target bitrate' aussucht. Es handelt sich also eher um einen abr (average bitrate) Mode als einen cbr (constant bitrate) Mode, auch wenn versucht wird, möglichst mit konstanter Datenrate zu encoden.

Über den 

- Button kommt man nun zu den erweiterten Single pass Einstellungen: Reaction Delay Factor, Averaging Period und Smoother, zu denen ich auf der nächsten Seite etwas schreibe.

Reaction Delay Factor:

Der Faktor der hier angeben wird, legt fest, wie schnell sich der Codec an eine geänderte Szene anpassen kann. Zu hohe Werte führen zu starken Qualitätsschwankungen. Zu niedrige Werte führen dazu, dass Datenrate verschenkt wird, da eine Szene eventuell mehr Datenrate erhält als sie benötigt.

Averaging Period:

Hier wird angegeben wie schnell der Codec versucht sich an die aktuelle Qualität anzupassen.

Smoother:

Der hier angegebene Wert bestimmt wieviele Frames encoded werden dürfen bevor die durchschnittliche Datenrate wieder hergestellt sein muss. Hohe Werte können zu stärkeren Abweichungen der Zielgröße des Files führen, wohingegen niedrige Abstände zu verstärkten Qualitätsschwankungen führen können.

Anmerkung:
Die drei Einstellungen sollten immer nur in Kombination gesehen werden, da sie alleine betrachtet nicht viel Aussagekraft haben.

Leider kann ich hier keine Empfehlungen geben, da ich (fast) nie das Single pass Verfahren verwende. => Ich würde mich über Erfahrungswerte freuen. :)

Twopass – Verfahren:

Allgemein versucht der Codec, wenn man ein **Twopass**-Verfahren wählt, die subjektive Qualität konstant zu halten und gleichzeitig die gewünschte Dateigröße zu erreichen. Im **Twopass - 1st pass** wird die Quelle analysiert und die gesammelte Informationen in einer .pass (Statistik-) Datei gespeichert. Diese Daten werden dann im **„Twopass - 2nd pass“** benutzt, um die gegebene Datenrate möglichst ideal auf die einzelnen zu komprimierenden Szenen zu verteilen.

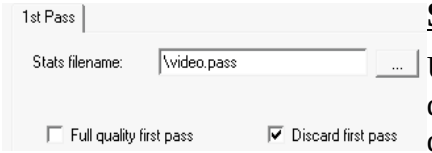
Twopass - 1st pass:

Der **Twopass - 1st pass** ist, wie schon erwähnt, eine Art Probedurchgang. Der Encoder analysiert jedes einzelne Bild und speichert wieviel Platz benötigt wird, wenn das Frame mit maximaler Qualität encodet wird, wie hoch die Bewegungswerte aktuell sind, usw. . Diese Daten werden dann in einer Statistik Datei mit Endung .pass gespeichert, auf diese kann ein folgender **Twopass - 2nd** pass zugreifen. Nach dem Durchlaufen des **Twopass - 1st pass** 'muss' noch ein 2ter Durchlauf im **Twopass - 2nd pass** folgen. Anzumerken sei hierbei, dass Xvid im **Twopass - 1st pass** die Features VHQ, Quarterpixel, GMC, Chroma Motion/Optimizer und Trellis sowie Turbo automatisch deaktiviert. Man kann für Testzwecke mehrere **Twopass - 2nd pass** Durchgänge machen, welche alle auf die während des **Twopass - 1st pass** einmalig erstellte Statistik-Datei zugreifen, und so den Effekt der genannten Features testen.

*Man kann also ohne Probleme/Gefahr im **Twopass - 1st pass** die Features VHQ, Quarterpixel, GMC, Chroma Motion/Optimizer und Trellis sowie Turbo deaktiviert haben und sie erst im **Twopass - 2nd pass** aktivieren.*

Über den

- Button kommt man zu folgendem Fenster:



Stats filename:

Unter Stats filename kann man den Namen und den Speicherort der Statistikdatei ändern. Gibt man *.\video.pass* ein, so wir dieStatistikdatei immer Verzeichnis des Inputmaterial erstellt.

Discard first pass:

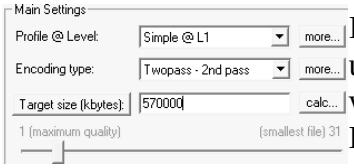
Während des 1st pass wird, wie bereits erwähnt, einiges an Daten gesammelt. Was ich aber vorher bewusst verschwiegen habe, ist, dass dabei nebenbei auch ein Videostream erzeugt wird. Deaktiviert man diese Option, so wird dieser nicht mehr standardmäßig gelöscht, sondern bleibt erhalten. Da der so erstellte Videostream aber nicht Mpeg4 standardkonform ist, wenn man nicht 'Full quality first pass' aktiviert hat, rate ich davon ab dieses Feature zu deaktivieren.

Full quality first pass:

Aktiviert man dieses Feature, so dauert der 1st pass etwas länger, dafür erhält man aber, wenn 'Discard first pass' deaktiviert ist, immer noch einen standardkonformen Videostream (mit festem Quantizer 2 und ohne die unter Twopass - 1st pass genannten zusätzlichen Features). Da man normalerweise beide Durchläufe des Two pass Encodingvorgangs erledigen will und dies ohnehin schon einiges an Zeit verschlingt, empfehle ich dieses Feature nicht zu aktivieren.

Soweit zum ersten Durchgang des 2pass Verfahrens, jetzt aber zum filmerzeugenden Schritt des Twopass Verfahrens.

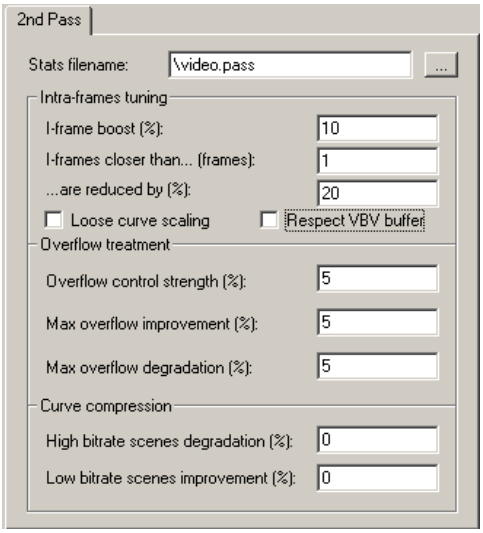
Twopass - 2nd pass:



Im **Twopass - 2nd pass** übernimmt Xvid die Verteilung der Datenrate und versucht mit Hilfe der Einstellungen, die ich noch näher erläutern werde, eine subjektiv gleichbleibende Qualität bei gleichzeitigem Erreichen der gewünschten Dateigröße zu erzeugen.

Anders als beim Single Pass Verfahren kann man aber nicht mehr nur einen angestrebten durchschnittliche Quantizer oder eine durchschnittliche Datenrate angeben, sondern man hat auch die Möglichkeit eine **Target size** in **kBytes** (1024 kBytes = 1 MB) für die resultierende Größe des Videostreams anzugeben. Will man keinen externen Calculator verwenden um zu berechnen, wie groß der Videostream ohne Audiostream&Co sein darf, findet man hinter dem calc... – Button eine Möglichkeit dafür, auf die ich später eingehen werde.

Weiter geht es nun mit dem more... - Button, der uns zu den 2nd pass Settings bringt, welche die Datenratenverteilung in Xvid beeinflussen.



Stats filename:

Unter **Stats filename** muss man den Pfad zur Statistikdatei aus dem Twopass - 1st pass angeben. Hat man im **Twopass - 1st pass** die Standardeinstellung nicht geändert, braucht man natürlich hier auch nichts zu ändern. Verwendet man nicht den Standardnamen und -pfad, so muss darauf geachtet werden, nicht aus Versehen eine falsche Statistikdatei zu verwenden.

Intra-frames tuning:

Die folgenden Einstellungen beziehen sich alle auf die Vergabe von I-Frames, da I-Frames als Basis für alle anderen Bilder im Videostream fungieren, sollte man hier nicht einfach auf's Geratewohl herumprobieren.

I-frame boost:

Hier kann man I-/Key-Frames extra Bits zuteilen, damit ihre Qualität noch ein bisschen höher ist. Ein Wert von 20% würde also einem Frame, der normalerweise 1000kBit zugeteilt bekäme, 1200kBit zur Verfügung stellen. Persönlich bevorzuge ich es, diesen Wert auf 0, 10 oder 20 zu lassen, da Xvid meiner Ansicht nach den I-Frame durchaus genug Datenrate zugesteht.

Man sollte hier beim Standard von 20 bleiben.

I-frames closer than... (frames):

Hier kann man angeben, wie viele Nicht-I-Frames mindestens zwischen zwei I-Frames stehen müssen, damit das zweite I-Frames nicht weniger Datenrate erhält. Der Standardwert von 1 deaktiviert das Feature, aufeinanderfolgende I-Frames würden also nicht reduziert. Will man diesen Abstand ändern, z.B. weil man einen sehr aktionsreichen Film mit vielen Szenenwechsel hat, so sollte man nach meiner Erfahrung den Wert nicht zu hoch wählen, da sonst auch langsamere Szenen eine schlechtere Qualität erhalten. Werte zwischen 1 und 5 halte ich aber für sinnig und brauchbar, da man schnelle Szenenwechsel normalerweise nicht wirklich scharf wahrnimmt.

are reduced by (%):

Der hier angegebene Wert gibt an, um wieviel Prozent ein I-Frame, das nach weniger als 'I-frame closer than..' Frames auf ein anderes I-Frame folgt reduziert wird. Zu bedenken ist, dass, wenn mehrere I-Frames aufeinander folgen, das letzte I-Frame wieder mit voller Datenrate encoded wird. Würden also drei I-Frames nah genug aufeinander folgen, so würde nur das mittlere I-Frame in der Größe reduziert. Persönlich bevorzuge ich hier Werte zwischen 10% und 20%.

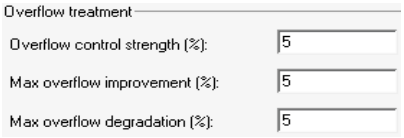
Loose curve scaling:

Beim Loose curve scaling handelt es sich um das 'alte' bis dato einzige Verfahren, dass Xvid zum Anpassen der Bitratenkurve verwendet hat. Einer der Entwickler hat jetzt noch ein weiteres Verfahren geschrieben, welches im Gegensatz zum 'alten' Verfahren verstärkt darauf achtet eine gleichbleibendere Qualität zu erhalten. Persönlich mag ich das neue 'strict curve scaling' nicht so sehr, da es teilweise aber sowohl besser als auch schlechter ist sollte jeder User hier selber entscheiden welches Verfahren er bevorzugt.

Respect VBV buffer:

Eigentlich wäre die Bezeichnung Respect VBV schon aus reichend gewesen, da VBV für 'Video Buffering Verifier' steht. Jedes Mpeg4 Profil hat eine ihm zugewiesene maximale Buffergröße, aktiviert man dieses Feature, so wird darauf geachtet, dass der für's Abspielen des Videostreams benötigte Buffer größer als diese Vorgabe seien muss. Durch diese Einschränkung wird der Codec jedoch eingeschränkt, da er nicht so stark mit den Datenraten 'schwanken' darf, was bei einigen Clips zu qualitativ schlechteren werden führt. Wenn man anders als Ich nicht nur für eine Wiedergabe an einem Rechner encoded sollte man aber sicherheitshalber das Feature aktivieren, da sonst die Gefahr besteht, dass die Wiedergabe einfriert oder stockt wenn ein zu großer Video Buffer benötigt wird.

Overflow treatment:



Da man beim Twopass – Verfahren ja nicht mit einer konstanten Datenrate arbeitet und die Vorhersage anhand der Daten aus dem 1st pass nicht 100%ig ist, muss darauf geachtet werden, dass die Verteilung der Datenrate trotzdem korrekt geschieht.
Wenn eine Szene also mal etwas mehr/weniger Datenrate erhält, müssen die folgenden Szenen dementsprechend angepasst werden. Die folgenden Features sollen dem User eine gewisse Freiheit geben, die normale Verteilung/Angleichung zu beeinflussen. Nimmt man hier höhere Werte, wird die angegebene Dateigröße/Datenrate besser getroffen, jedoch kann es schneller zu Qualitätsschwankungen kommen, also Vorsicht mit diesen Settings.

Erhält man zu große/kleine Dateien ist es sinnig die drei Werte anzuapssen indem man sie z.B. auf 10 zu setzen.

Anmerkung zur Klärung:
Ein Overflow entsteht, wenn weniger Datenrate als erwartet verbraucht wird um einen Frame mit einem gewissen Quantizer zu encoden. Ein Overflow kann also zu unerwartet kleinen Dateien führen.
Ein Underflow entsteht, wenn mehr Datenrate als erwartet verbraucht wird um einen Frame mit einem gewissen Quantizer zu encoden. Ein Underflow kann also zu unerwartet großen Dateien führen.

Overflow control strength(%):

Hier legt man fest, wie aktiv die Datenraten-Kontrolle sein soll. Beim Wert von 0 läuft alles so, wie es von Xvid geplant war. Erhöht man den Wert, so wird schneller auf einen Over-/Underflow reagiert und eine Reduktion oder eine Steigerung der Datenrate/Quantizer durchgeführt, um wieder auf der gedachten durchschnittlichen Datenrate zu liegen. Hohe Werte führen also dazu, dass schneller/aggressiver versucht wird wieder die 'mittlere' Datenrate zu erreichen. Was zu stärkeren Fluktuationen der Quantizerverteilung führt.

Meiner Erfahrung nach sind Werte bis 10% ohne Probleme zu verkraften und sorgen dafür, dass die angestrebte Datenrate/Dateigröße genauer erreicht wird.

Max overflow improvement(%):

Lässte Xvid weniger Spielraum, wenn die Datenrate mal niedriger ausfällt als erwartet. D.h., Xvid reagiert langsamer auf Wechsel, da es sich um den wachsenden Overflow weniger sorgen macht. Höhere Werte können zu undersized Dateien führen, da die Gefahr größer ist, dass der Overflow nicht rechtzeitig abgebaut wird wenn die Overflow control strength nicht entsprechend hoch ist. Mit einem Wert von 1 überlässt man komplett Xvid diese Wahl.

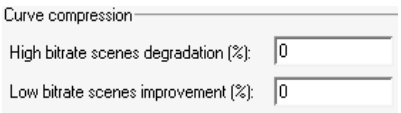
Meiner Erfahrung nach sind Werte bis 10% ohne Probleme zu verkraften und sorgen dafür, dass die angestrebte Datenrate erreicht wird, aber die Verteilung der Quantizer besser ist.

Max overflow degration(%):

Lässte Xvid weniger Spielraum, wenn die Datenrate mal höher ausfällt als erwartet. D.h., Xvid reagiert langsamer auf Wechsel, da es sich um den wachsenden Underflow weniger Sorgen macht. Höhere Werte können zu oversized Dateien führen, da die Gefahr größer ist, dass der Underflow nicht rechtzeitig abgebaut wird wenn die Overflow control strength nicht hoch genug ist.
Mit dem Standardwert von 1 überlässt man komplett Xvid diese Wahl.

Meiner Erfahrung nach sind Werte bis 10% ohne Probleme zu verkraften und sorgen dafür, dass die angestrebte Datenrate erreicht, wird aber die Verteilung der Quantizer besser ist.

Curve compression:



In der **Curve compression** kann der User eine allgemeiner Gewichtung festlegen, wie stark auf zu hohe bzw. zu niedrige Bitraten reagiert werden soll.

High bitrate scenes degration(%):

Dieser Wert gibt an, um wieviel Prozent ein Frame reduziert wird dessen Größe über der durchschnittlichen Framegröße liegt. Mit dem Standardwert von 0 überlässt man, wie ich es auch als sinnig erachte, komplett Xvid diese Wahl.

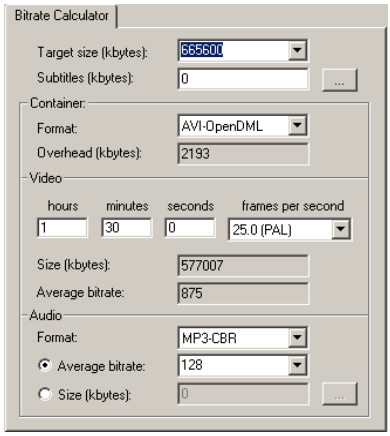
Low bitrate scenes degration(%):

Dieser Wert legt fest, um wieviel Prozent die Größe von Frames vergrößert wird, die kleiner sind als das durchschnittliche Frame. Mit dem Standardwert von 0 überlässt man, wie ich es auch als sinnig erachte, komplett Xvid diese Wahl.

Jetzt kommen wir zum Bitrate Calculator, zu dem man über den 

– Button gelangt.

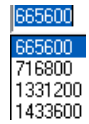
Bitrate Calculator:



Der interne Bitrate Calculator ist vor allem für Leute gedacht, die nicht externe Programme wie z.B. DVDtoOgm, AutoGK oder GordianKnot verwenden. Diese Tools kümmern sich um den jeweiligen Container-Overhead und sonstiges eigenständig.

Target size (kbytes):

Hier gibt man an wie groß die Datei im Endeffekt werden soll. Hier kann man entweder selber einen Wert eintragen oder zwischen den Standardwerten wählen.
665600 entspricht 650 MB, 716800 sind 700MB, 1331200 sind 2*650 MB und 1433600 entspricht 2*700MB.



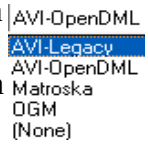
Entsprechend den Einstellungen des Bitrate Calculator wird auch die Target Size im 2nd pass automatisch angepasst.

Subtitles:

Subtitles (= Untertitel) kann man auf 0 lassen, es sei denn, man verwendet einen Container wie ogm oder mkv, in dem man Untertitel als extra Stream mit in den Container packen kann oder man speichert die Untertitel als z.B. .srt Files neben einem Avi und überlässt dem Player das gemeinsame Abspielen.
Also nicht vergessen, packt man Untertitel mit zum Film, sollte man nicht vergessen sie anzugeben, da das Endprodukt sonst eventuell zu groß ist, um noch auf eine CD zu passen.

Container:

Format:

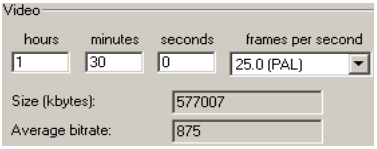


Unter **Format** hat man nun die Möglichkeit zwischen den gängigsten Containerformaten zu wählen. Bei *Avi-Legacy* handelt es sich um das alte avi Containerformat, das man heutzutage eigentlich nur selten verwendet, da es u.a. nur Dateien bis zu einer Größe von 2GB unterstützt. Bei *Avi-OpenDML* handelt es sich um einen neueren erweiterten Avi Container, der von den meisten Tools unterstützt und verwendet wird, da er z.B. nicht mehr die 2GB Beschränkung hat. Bei Matroska und Ogm handelt es sich um neuere Dateiformate, die zwar nicht so verbreitet, dafür aber sehr nützlich sind. Genauere Informationen zu den Containern findet man in den auf der ersten Seite angesprochenen Foren und z.B. bei http://de.wikipedia.org .

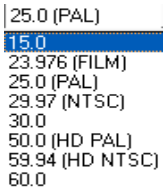
Overhead:

Je nachdem was für ein Format man gewählt hat, ändert sich auch der Overhead der vom jeweiligen Container zur Verwaltung der Streams benötigt wird. Wie genau sich der Overhead genau zusammensetzt kann man hier: http://www-user.tu-chemnitz.de/~noe/Video-Zeug/ durchlesen.

Video:



Unter Video kann man nun die Länge und die Framerate des zu encodenden Videos angeben. PAL (Phase Alternating Line) ist der in Europa genutzte Video-Standard und NTSC (National Television Standards Committee - scherzhaft: Never (Twice) the Same Color) ist der in USA und weiten Teilen Asiens genutzte Video-Standard.



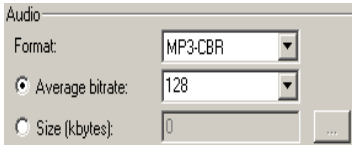
Die Abkürzung HD steht hierbei für High Definition und sollte normalerweise nicht benötigt werden. Persönlich benutze ich nur 25.0 PAL sowie 23.976 FILM, bei dem es sich um reine progressive Frames eines NTSC Streams handelt, die man per IVTC erhält.

Wenn man sich unsicher ist, sollte man bei 25.0 PAL bleiben.

Size und Average bitrate:

Unter **Size** und **Average bitrate** sieht man immer, wie groß die durchschnittliche Datenrate/Größe des zu encodenden Videostreams sein wird. Persönlich würde ich empfehlen hier möglichst nicht unter 800kBit bei der durchschnittlichen Datenrate zu gehen, wenn man ein qualitativ gutes Ergebnis bei noch einigermaßen hohen Auflösungen erreichen will.

Audio:



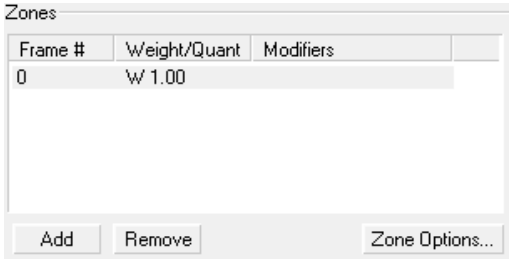
Bei den Audioeinstellungen sollte man entweder unter Size ein schon encodetes Audiofile öffnen, damit dessen Größe erkannt wird oder oben das später noch zu encodende Audioformat und die angestrebte Datenrate angeben.

Zur Auswahl stehen hier die gängigsten Formate, die man in Avis finden kann. Da haben wir einmal Mp3-cbr, wobei CBR für constant bitrate steht. Will man das zu erstellende File später noch weiterverarbeiten, würde ich CBR Audio empfehlen, da es bei den wenigsten Programmen Probleme macht. Alle Formate außer Mp3-cbr sind nur durch Tricks im Avi Container erlaubt, welche eventuell nicht von allen Programmen unterstützt werden. Bei Mp3-vbr handelt es sich um einen Mp3 Stream mit variabler Bitrate. Ogg steht für Ogg Vorbis und ist ein sehr leistungsfähiges Audioformat (siehe: http://www.vorbis.com/). AC3 und DTS sind die Standard Audioformate, die bei DVDs verwendet werden und nur zu empfehlen, wenn man die entsprechende Audioanlage hat, da sie normalerweise recht groß sind.

Persönlich bin ich zwar der Ansicht, dass Calculatoren und dergleichen nicht in einen Codec sondern in ein separates Programm gehören. Anfängern erleichtert ein solcher Calculator jedoch das Arbeiten, weil sie meist keine externen Calculatoren kennen und so z.B. leicht den Container Overhead vergessen mit zu berechnen.

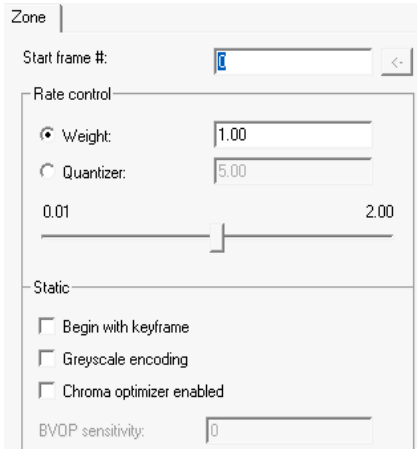
So jetzt aber wieder zu etwas fordernden Einstellungen des Codecs: den Zones und Zone Options.

Zones:



Mit Zones (=Abschnitten) kann man gewisse Einschränkungen für manche Bereiche des Films machen. Vor allem für Abspänne, Intros und Filme, in denen öfters längere Textpassagen über den Bildschirm scrollen, kann dies sehr hilfreich sein. Mit **Add** und **Remove** kann man neue Zonen hinzufügen und löschen. Genau festlegen wo immer eine neue Zone anfängt, kann man dann indem man die Zone auswählt und die Zone Options dieser Zone einstellt.

Zone Options:



Start frame #:

Hier legt man fest, ab welchem Frame im Clip eine neue Zone anfangen soll.

Rate control:

In diesem Bereich kann man festlegen, ob die Zone die man definiert mit einem festen Quantizer encoded wird oder ob sie etwas mehr oder weniger Datenrate im Verhältnis zum Rest des Films erhalten soll.

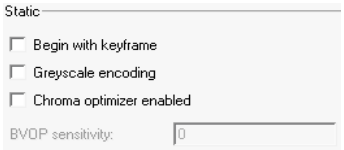
Weight:

Hier kann man eine Gewichtung festlegen, wieviel Datenrate der Bereich im Verhältnis zum Rest des Films erhalten soll. Der Wertebereich aus dem man wählen kann, liegt zwischen 0.01 und 2.00. Ein Wert von 1.0 würde dem Bereich die gleiche Datenrate, ein Wert von 0.01 nur 1% der Datenrate und ein Wert von 2.00 das Doppelte der Datenrate im Vergleich zum Rest des Films zuweisen. Trotz anfänglicher Skepsis meinerseits erhält man auch bei einem Wert von 0.01 für den Abspann bzw. für weißen Text auf dunklem statischem Hintergrund noch gute Ergebnisse. Ich würde deshalb empfehlen für Abspäne, Intros und eventuelle Textpassagen hier durchaus niedrige Werte kleiner 0.1 zu nehmen.

Quantizer:

Hier kann man festlegen, dass der Bereich mit einem festen Quantizer encoded werden soll. Vor allem wenn man Szenen hat denen man besonders viel oder besonders wenig Datenrate zuteilen will, ist diese Option empfehlenswert. Da es jedoch stark von der Quelle und der Quantizer Matrix abhängen kann, welcher Quantizer hier genug und welcher zu niedrig ist, bevorzuge ich lieber die Weight Option.

Static:



In diesem Bereich kann man u.a. einstellen ob für einen Bereich die Farbinformationen verworfen werden oder verstärkt B-Frames verwendet werden sollen.

Begin with Keyframe:

Diese Option ermöglicht es dem User zu *erzwingen,* dass der Bereich mit einem Keyframe anfängt. Der einzige Sinn, der mir hierfür eingefallen ist, ist wohl Sprungstellen für Kapitel erstellen. So etwas ist in Avi-Hüllen nicht möglich ist, wohl aber in .mkv (Matroska) und .mp4 Hüllen. (Entsprechende Tools vorrausgesetzt.)

Greyscale:

Hierbei handelt sich um eine Funktion, die sämtliche Farbinformationen eines Bildes verwirft. Man sollte diese Option also normalerweise nur aktivieren, *wenn man eine Schwarz/Weiß Quelle* hat, die man komprimieren möchte. Greyscale führt meist maximal zu einem Ersparnis von 5-7%, was es eventuell auch interessant macht, es nicht nur bei Schwarz/Weiß Szenen sondern auch bei den Credits (Abspann) zu aktivieren.

Chroma Optimizer:

Farbinformationen in besonders hellen bzw. dunklen Bereichen werden interpoliert, was unter anderem die "Treppchenbildung" an Kanten (z.B. grell roter Pulli vor blauem Himmel) reduziert. Vor allem bei Filmen mit vielen Helligkeitsübergängen würde ich empfehlen sie zu aktivieren.

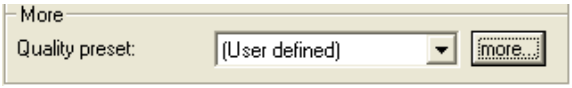
Persönlich lasse ich sie, obwohl sie den Komprimierungsprozess bremst, standardmäßig aktiviert.

BVOP sensitivity aka. B-frame threshold:

Bei diesem Feature kann der User die Wahrscheinlichkeit erhöhen, dass anstatt einem P- ein B-Frame verwendet wird. Das unter „max consecutive BVOPs“ Maximum wird hierdurch jedoch nicht überschritten. Es ist vielmehr die Frage, wie schnell es erreicht werden wird. Sinnige Werte sind hierbei im Bereich von -25 bis 25 angesiedelt, wobei ich persönlich nur, wenn ich „max consecutive BVOPs“ auf höheren Werten als 2 habe, auch die sensitivity hoch setze. Im Abspann bzw. im Intro oder generell Szenen, die nicht so scharf sein sollen/müssen wie der eigentliche Film, setze ich den Wert auf 25 und lasse ihn ansonsten beim Standard von 0. Wenn man z.B. 'nen '1 CD Rip' macht, ist ein Wert von 10 sicher auch sinnig.

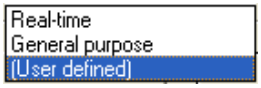
Würde ich allgemein auf 0 lassen und beim Abspann&Co auf 25 hoch setzen.

More:



In dieser Sektion folgen nun einige Features, die *nicht den Bitstream ändern*, also nichts mit Kompatibilität zu tun haben.
Diese Features sind also vor allem für User interessant, die später auf einem Stand-Alone-Player ihre Files angucken wollen, da man sich keine Gedanken darum machen muss, ob der Player die Features unterstützt oder nicht. Um es neuen Usern zu erleichtern haben die Xvid Entwickler hier dem User die Möglichkeit gegeben auf vorgegebene Quality presets zurück zugreifen.

Quality presets:

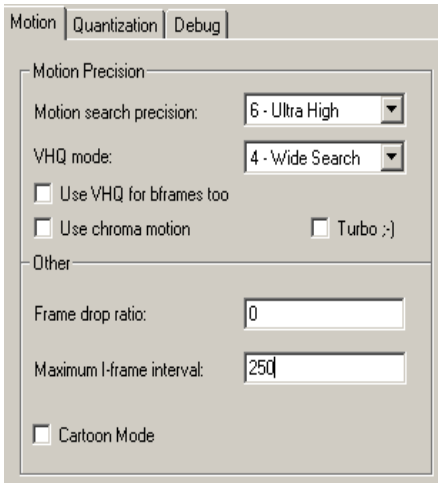


Momentan gibt es hier nur drei Möglichkeiten der Auswahl *Real-time, General purpose* und *User defined*. In der Real-time Vorgabe wird nur die schnellste, aber auch ungenauste Bewegungssuche durchgeführt. Der einzige Grund der mir einfällt diesen preset zu nehmen, ist wenn man auf einem sehr alten Rechner (<1GHz) etwas live capturen will, bei allen anderen Szenarien halte ich das *Real-time* profil für nicht geeignet. Das General purpose Profile liefert zwar nicht die beste mögliche Qualität ist aber für Standard DVD nach Xvid Konvertierungen okay. Im *User defined* preset kann man selber festlegen was wie eingestellt sein soll.

Sollte man auf User defined oder zumindest General purpose eingestellt werden.

Entscheidet man sich hier selber Hand anzulegen, und klicked auf den 

-Button, dann geht es hier weiter:

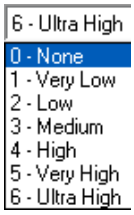


In dieser Sektion folgen nun einige Features, die *nicht den Bitstream ändern*, also nichts mit Kompatibilität zu tun haben. Diese Features sind also vor allem für User interessant, die später auf einem Stand-Alone-Player ihre Files angucken wollen, da man sich keine Gedanken darum machen muss, ob der Player die Features unterstützt oder nicht.

Motion Precision:

Bei den folgenden Features geht es immer um die Genauigkeit, mit der Xvid Bewegungen bzw. Bewegungsvektoren erkennt. Dies ist die Basis für viele der anderen Features, jedoch auch die Zeit intensivsten Features von Xvid.

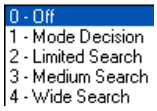
Motion search precision:



Je besser die Qualität später sein soll, desto höher sollte die „Motion search precision“ sein, da sie festlegt, wie genau die Suche nach Bewegungsvektoren durchgeführt wird und wie genau der Codec Ähnlichkeiten zwischen einzelnen Frames erkennt. Mode 1 bis 3 benutzen in etwa die gleichen Routinen und unterscheiden sich nur recht wenig. Ab Stufe 4 interpoliert der Codec bis auf halbe Pixel genau, um so eine feinere Suche zu erreichen.
Auch erhältjeder ein einzelne Macroblock einen einzelnen Bewegungsvektor, der die Bewegung zwischen zwei einzelnen Bildern beschreibt. Ab Stufe 5 benutzt XviD "inter4v motion vectors", was bedeutet, dass alle vier 8x8 Blöcke eines 16x16 Macroblocks einen eigenen Bewegungsvektor erhalten.
Mit Stufe 6 wird die Suche teilweise öfter vorgenommen, was zu einem etwa 10%igen Geschwindigkeitsverlust führt, jedoch genauere Ergebnisse liefert und so die Komprimierbarkeit etwas erhöht. Da es sich bei der Motion search precision um eine Funktionen handelt, deren Genauigkeitsgrad direkten Einfluss auf fast alle folgenden Features hat, sollte man hier normalerweise bei Mode „5- Very High“ oder besser „6-Ultra High“ bleiben. Nur wenn man mit dem Codec captured, sollte man eventuell hier die Priorität senken um zu vermeiden, dass Frames verloren gehen.

Sollte man normalerweise auf 6 stehen haben.

VHQ:



VHQ errechnet die Anzahl von Bits, die ein Makroblock in verschiedenen Szenen erreichen kann. Gewählt wird dann das Szenario, welches die kleinste Anzahl an Bits aufweist. Ebenso kann VHQ [eingeschränkt] Bewegungsvektoren suchen.

Da GMC nur wirklich effektiv arbeitet, wenn VHQ aktiviert ist, sollte man GMC mindestens mit VHQ 1 verwenden. Dies ist auch der Grund, warum VHQ 1 die Standardeinstellung ist. Je höher die VHQ Stufe ist die man wählt, desto extremer wird der Encodingvorgang ausgebremst. Für die meisten User ist es deshalb wohl am sinnigsten, VHQ 1 oder 2 zu nehmen, da hier der meiste Qualitätsgewinn erzeugt wird und die Geschwindigkeitseinbußen noch nicht so hoch sind. Für absolute Qualitätsfanatiker wie mich, ist VHQ(4) jedoch ein Muss.

Wenn man die zusätzliche Wartezeit beim Encoden verkraften kann, sollte man VHQ aktivieren.

VHQ for B-Frames too:

Dieses Feature ist erst in 1.1er builds vorhanden, ist es aber sicher Wert verwendet zu werden. Anders als bei den 1.0er Versionen des Codecs wird nicht nur Versucht die I- und P- Frames zu optimieren, sondern auch B-Frames werden möglichst optimal gespeichert. Da B-Frames, wenn man sie den verwendet, meist den Großteil der verwendeten Frames darstellen bremst dieses Feature den Encoding Prozess nochmal relativ stark aus. Da die 1.1er Versionen von Xvid jedoch an sich schon schneller sind als die 1.0er Versionen und der Qualitätsgewinn durchaus spürbar ist würde ich vorallem für Qualitätsfreaks das Feature empfehlen.

Wenn man die zusätzliche Wartezeit beim Encoden verkraften kann,
sollte man VHQ auch für B-Frames aktivieren.

Chroma motion:

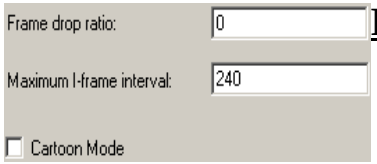
Normalerweise verwendet Xvid nur die Luminanz-(Helligkeits-)Informationen, um Bewegungsrichtungen festzustellen. In seltenen Fällen (die gleiche Helligkeit, aber wechselnde Farbigkeit) mag dies jedoch vielleicht nicht genau genug sein. Dann kann man mit Hilfe dieser Option zusätzlich zur Helligkeits- auch die Farb-Komponenten zur Ermittlung der Bewegungsrichtung verwenden. Dies bremst zwar das eigentliche Komprimieren, *sollte aber für genauere MotionEstimation verwendet werden.*

Turbo;):

Dieses Feature aktiviert einige Flags, die mit B-Frames und Quarterpixel zu tun haben und diese Optionen im Twopass 2nd pass ein bisschen ungenauer, aber (messbar) flinker machen. Durch die Ungenauigkeit wird zwar das File etwas größer bzw. die Qualität leidet etwas, um dies etwas zu kompensieren sollte man die *Overflow Control Strength* von den Standard 5% auf 10% erhöhen.

Wenn man also mal nicht unbedingt auf 100% Qualität abzielt, sondern kleine oft nicht sichtbare Ungenauigkeiten hinnehmen kann, sollte man den 'Turbo' aktivieren.

Other:



Frame drop ratio:

Hierbei handelt es sich um einen Wert, der angibt, um wieviel Prozent sich zwei Frames unterscheiden müssen, damit Xvid den zweiten Frame nicht kodiert, sondern den ersten einfach wiederholt. Vor allem für 'old-style' Animes wäre dies sicher interessant. Dieses Feature jedoch nur genutzt, wenn B-Frames deaktiviert sind. Ich würde davon abraten einen anderen Wert als den Standardwert (0) zu nehmen. Eine Implementation, die mit B-Frames zusammen arbeitet, ist geplant, aber nicht in näherer Zukunft zu erwarten.

Maximum I-frame interval:

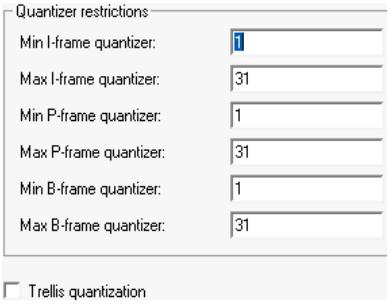
Das hier festzulegende Intervall gibt an, nach wievielen Frames spätestens ein I-/Key-Frame gesetzt werden soll. Einerseits ist es aus kompressionstechnischen Gründen sinnig, nur an Szenenwechseln, bzw. in Szenen wo starke Änderungen zu den Vorgängern auftreten, ein I-Frame zu setzen, d.h., man sollte nicht zu viele I-Frames setzen. Andererseits werden I-Frames auch als Sprungpunkte benötigt, d.h., man will nicht zu wenige, da man sonst nur in groben Sprüngen im Film umherspulen bzw. -springen kann. Als guter Richtwert hat sich ein Wert eingebürgert, der etwa dem 10fachen der Framerate entspricht. Für PAL Filme wäre dies ein Wert von 250. (NTSC =240 bzw. 300)

Cartoon Mode:

Der Cartoon Mode ist dafür gedacht typische Zeichentrickfilme mit großen gleichfarbigen Flächen besser zu encoden. Dies soll dadurch erreicht werden, dass einmal der Schwellenwert ab wann ein Bildelement als statisch anerkannt wird hoch gesetzt wird (verdreifacht im Vergleich zum Original). Andererseits wird ein Limit eingeführt, bei das festlegt, wieviele Änderungen geschehen dürfen, ohne dass ein Macroblock, nicht neu gespeichert werden muss. Dies kann zwar zu minimalen Bildfehlern führen, sollte im Allgemeinen aber einiges an Platz sparen ohne größere 'Schäden' anzurichten. Wichtig ist hierbei, dass der Mode nicht für Zeichentrickclips geeignet ist, die einen detaillierten Hintergrund haben.

Würde also empfehlen den Mode wirklich nur bei typisch alten Cartoons zu benutzen.

Quantization:



Zur Einleitung erst mal eine grobe Erklärung, worum es sich bei Quantizern eigentlich handelt:

Beim Enkodieren wird ein Bild in einzelne 8x8 bzw. 4x4 Makroblöcke unterteilt, die durch eine (Forward) Discrete Cosinus Transformation (DCT) gejagt werden. Die Koeffizienten, die sich hieraus ergeben, werden vor dem Speichern auf ganze Zahlen gerundet.
Ein Quantizer ist nun ein Faktor, durch den vor dem Runden dividiert wird. Wäre der Quantizer 1, wird der Wert einfach auf die nächste ganze Zahl gerundet und gespeichert. Ist der Quantizer größer als 1, wird die Zahl erst durch den Quantizer geteilt ehe sie gerundet und anschließend gespeichert wird. Je größer der Quantizer, desto kleinere und weniger unterschiedliche Werte müssen gespeichert werden => weniger Platzbedarf.

Beim Dekodieren werden die Koeffizienten zuerst wieder mit dem ihnen zugehörigen Quantizer multipliziert. Je höher der Quantizer, desto höher ist also die Gefahr, dass die gespeicherten Koeffizienten ungenau sind. Diese Ungenauigkeiten treten dann oft als offensichtliche Kanten auf. In einem solchen Moment spricht man von Blockbildung bzw. 'blockiness'.

Quantization restrictions:

Hier kann man für die einzelnen Frame Typen (I-/P-/B-Frames) minimale und maximale Beschränkungen für die Quantizer festlegen. Was genau I- / P- und B-Frames sind, ist in jeder halbwegs brauchbaren Beschreibung, die sich mit Mpeg beschäftigt, nachzulesen. Da meiner Erfahrung nach Xvid auch wunderbar arbeitet, wenn man hier keine Beschränkung vornimmt, sollte man diese Werte nicht ändern. Wichtig ist auch, dass man die I-Frames nur im Twopass Verfahren nicht jedoch im Single pass Verfahren einschränken kann.

Trellis quantization:

Hierbei handelt es sich um eine Erweiterung der normalen Quantisierung bzw. ein Art 2ten quantization pass in dem die DCT Verteilung nochmal überdacht wird. Es werden einige Koeffizienten fallen gelassen (Details entfernt) und andere Koeffizienten die sonst wegfallen würden, gerettet um ein besseres Bild bzw. eine bessere Quantisierung zu erreichen. Da Xvid sich beim Einsparen geschickt anstellt kann im Endeffekt oft ein etwas kleinerer Quantizer verwendet werden und man gewinnt sogar Qualität.

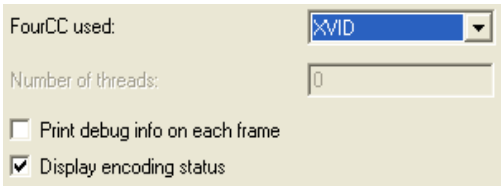
Ist bei DVD Rips empfehlenswert und sollte die Qualität (zumindest minimal) steigern, kann aber auch durchaus deaktiviert werden, ohne dass man große Qualitätseinbußen befürchten muss.

Other Options:

In diesem Bereich geht es darum einige allgemeine Einstellungen vorzunehmen, die man normalerweise nur einmal angucken und dann nie wieder anpacken muß.

Encoder:

Hier werden nun ein paar kleine Einstellungen die sich auf den Encoder an sich beziehen vorgenommen.



FourCC used:

FourCC steht für Four Character Code und ist ein Code, der einem Videostream einen Decoder zuweist, bzw. dem im System befindlichen Decoder der sagt der unterstützt diesen FourCC und die höchste Priorität hat.

Ein paar Beispiele: XVID = Xvid, DIVX = DivX4, DX50 = DivX5.x. Den FourCC Code kann man auch später noch ändern, falls man bei einer Datei generell einen anderen Codec verwenden möchte. Ein Tool dazu ist z.B. das mitgelieferte AviC. Wenn man also nicht will, dass Xvid selber, sondern z.B. DivX das File später decoden soll, kann man dies hier ändern.

Number of threads:

Dieses *noch inaktive* Feature ist, soweit ich es momentan erahnen kann, vor allem für User die im Besitz von Multiprozessorsystemen sind, sehr interessant, da es festlegt, wieviele einzelne Threads Xvid beim Encoden erstellen darf.

Print debug info on each frame:

Aktiviert man diese Option werden für jedes einzelne Bild debugging Informationen ausgegeben, was normalerweise nicht benötigt wird.

Display encoding status:

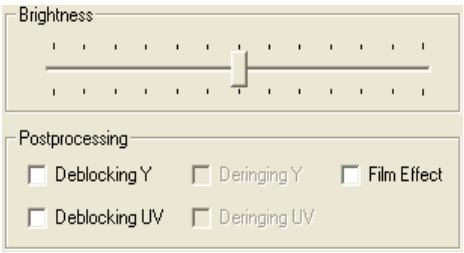
Aktiviert man dieses Feature wird beim Encoden ein zusätzliches Fenster angezeigt in dem links oben I-Frames rot, P-Frames blau und B-Frames grün eingetragen werden. Die Skala gibt hierbei an mit welchen Quantizer das file encoded wurde. Damit man nicht nur eine Skala zur Informationsammlung hat, kann man rechts oben noch 'Show me internals' aktiviern, worauf hin man aktiv sieht welches Frame wie encoded wurde. S steht hier für ein P-Frame bei dem GMC verwendet wurde.
Die Anzeige Schlüsselt sich also wie folgt auf:

[Frame Nummer]->Art des Frames q:verwendeter Quantizer (Größe des Frames in Byte)

In der unteren Hälfte des Xvid Statusfensters werden alle Informationen die im oberen Teil zu sehen in die entsprechenden Tabellenelemente eingetragen und können so u.a. leicht einen Eindruck vermitteln was genau bei den einzelnen Frames passiert wenn man ein bestimmtes Feature, wie z.B. die Quantizermatizen, den B-Frame Offset oder die B-Frame sensitivity ändert.

Decoder:

Bei den Decoder Options handelt es sich um Flags / Markierungen, die dem Xvid Decoder beim Abspielen mitteilen, dass gewisse PostProcessing Funktionen aktiviert werden sollen.



Brigthness:

Mit diesem Regler kann man die Helligkeit festlegen mit der ein Clip wiedergegeben wird. Gerade User die ihren Monitor an sich recht Hell/Dunkel eingestellt haben können mit diesem Feature die Helligkeit des Decoders beeinflussen.

Deblocking Y/UV:

Hiermit legt man fest, dass der Decoder versucht im Y und/oder UV Farbanteil des Bildes Markroblöcke zu verschleiern. Da alle Mpeg4 Codecs im YUV Farbraum arbeiten hier eine kleine Erläuterung: Der YUV Farbraum verwendet im Gegensatz zum RGB (RotGelbBlau) - Farbraum eine Helligkeitskomponente (Luminanz, Y) und die zwei Farbkomponenten U und V. Die Darstellung in diesem Farbraum hat einen Vorteil, der sich bei der Komprimierung und Übertragung von Videoinformationen gewinnbringend einsetzen läßt. Das Menschliche Auge ist gegenüber Helligkeitsinformationen (Y) viel sensibler als gegenüber Farbinformationen (U,V). Dadurch fällt es nicht auf, wenn nur die halbe Farbinformation (4:2:2 Abtastung) oder sogar noch weniger (4:2:0 Abtastung) zur vollen Luminanzinformation übertragen werden. Es läßt sich so in einem ersten Schritt die Datenmenge einer Videoübertragung ohne sichtbare Qualitätsveränderungen um gut ein Drittel reduzieren. (genaueres lässt sich per Google&Co erfahren)

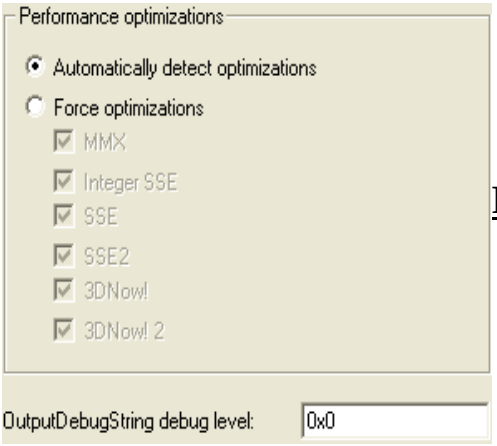
Persönlich aktiviere ich das Deblocking, je nach Datenrate kann es das Bild beim Abspielen aber durchaus ansehnlicher machen.

Filmeffekt:

Aktiviert man dieses Feature, so wird während dem Decoden ein künstliches Rauschen erzeugt, was etwaige Makroblöcke&Co nicht so gut wahrnehmen lässt.

Auch dieses Feature ist vor allem dann sinnig, wenn die Datenrate mal wieder ein bisschen zu niedrig im Verhältnis zum Input ist.

Common:



In dieser Sektion geht es vorallem darum fest zu legen was die von einem verwendete CPU alles für Features hat und ob eventuell besondere Debug Inforamtionen zur Fehlersuche gesammelt werden sollen.

Performance optimizations:

Hier hat man die Wahl Xvid selber versuchen zu lassen, heraus zu finden, welche Optionen die eigene CPU unterstützt, indem man **„Automatic detect optimizations“** auswählt, oder man wählt **„Force optimizations“** und kann selber aktivieren ob die eigene CPU MMX, SSE, usw. unterstützt. Normalerweise sollte man die Finger von diesen Optionen lassen, nur für ältere K5/K6/Celeron Prozessoren kann es meines Wissens vorkommen, dass die automatische Suche nicht erfolgreich ist.

Hier mal ein paar CPUs und die von ihnen unterstützten Befehlssätze:

	MMX	***MMX etx.***	***SSE***	***SSE2***	***3DNOW!***	***3DNOW! Pro***	***AMD64***
Intel Pentium	-	-	-	-	-	-	-
Intel Pentium MMX	Ja	-	-	-	-	-	-
Intel Pentium II	Ja	-	-	-	-	-	-
Intel Celeron	Ja	-	-	-	-	-	-
Intel Pentium III	Ja	Ja	Ja	-	-	-	-
Intel Celeron II	Ja	Ja	Ja	-	-	-	-
Intel Pentium IV, Celeron IV	Ja	Ja	Ja	Ja	-	-	-
AMD K6	Ja	-	-	-	-	-	-
AMD K6-2/K6-3	Ja	-	-	-	Ja	-	-
Athlon, Duron	Ja	Ja	-	-	Ja	-	-
Athlon XP	Ja	Ja	Ja	-	Ja	Ja	-
Opteron, Athlon 64, Athlon FX	Ja	Ja	Ja	Ja	Ja	Ja	Ja

OutputDebugString debug level & Print debug info on each frame:

Bei diesen Optionen handelt es sich um Features, die eigentlich nur für die Developer selbst nützlich sind, da sie erweiterte Informationen über den Encodingvorgang liefern. Normalerweise sollte man diese Features nicht anfassen, da sie meines Wissens außer zur Fehlersuche keine positiven Effekte haben.

So das war's mal wieder von meiner Seite her; hoffe das Dokument hilft dem ein- oder anderen ein wenig mit den Einstellungen in Xvid klar zu kommen.

Es ist noch kein Meister vom Himmel gefallen!

Cu Selur

Anhang I: DCT und Custom-Matrix, kurze Übersicht

(von Ethanolix & Videostation)

Einer der Vorteile von XviD ist, dass man direkt die Quantisierungsmatrix bearbeiten kann. Jedoch ist dies zu Anfang recht schwierig, wenn man nicht genau weiß, welche Bedeutung der Matrix zukommt bzw. für was die einzelnen Werte stehen.

Wir sind zwar keine Experten auf diesem Gebiet, denken aber soviel von der Materie zu verstehen, um all denjenigen, die auch gerne mit einer eigenen Matrix experimentieren wollen, eine kurze und grobe Übersicht über das Thema zu geben.

Wie kommt die Matrix zustande?

DCT - Helligkeits-/Farbwerte => Frequenzspektrum

Um bei der Videokomprimierung (wie auch bei der reinen Bildkompression nach JPEG) eine möglichst effektive Datenreduktion zu erreichen, überführt man die Bildinformation mittels Diskreter Cosinus Transformation (DCT) in den Frequenzbereich. Die DCT ist eine Sonderform der DFT (Diskrete Fourier Transformation). Die Besonderheit und der Vorteil der DCT liegt darin, dass das Ergebnis reellwertig ist und nicht wie bei der DFT komplexe Anteile enthält. Sie dient bei der Bildverarbeitung zur Irrelevanzreduktion.
Vor der DCT wird das Quellbild in 8*8 Pixel große Blöcke zerlegt, um einerseits den Rechenaufwand für die DCT in Grenzen zu halten und andererseits ein möglichst gutes Ergebnis sowie eine hohe Kompression zu erzielen. Aus diesem Vorgehen ergeben sich aber auch die allseits bekannten Blockartefakte bei der Dekodierung, falls das Material zu stark komprimiert wurde.
Ein Farbbild wird dazu in seine Komponenten zerlegt: Helligkeit (Luminance = Y) und Farbart (Chrominance = C). Das Farbartsignal besteht dabei aus zwei sogenannten Farbdifferenzsignalen, Cb entspricht dem Blaudifferenz- und Cr dem Rotdifferenzsignal. Die DCT wird bei jeder der drei Komponenten, für einen 8*8-Block, durchgeführt. Dem Algorithmus ist es dabei egal, ob es sich um Helligkeits oder Farbdifferenzwerte handelt.

Die zweidimensionale DCT ist definiert durch:

$$F(k,n) = \frac{1}{4} \cdot C(k) \cdot C(n) \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cdot \cos\left(\frac{\pi(2x+1)k}{16}\right) \cdot \cos\left(\frac{\pi(2y+1)n}{16}\right) \quad mit\ k,n \in \{0,\ldots,7\}$$

Die zweidimensionale iDCT lautet folgendermaßen:

$$f(x,y) = \sum_{k=0}^{7} \sum_{n=0}^{7} \frac{1}{4} \cdot C(k) \cdot C(n) \cdot F(k,n) \cdot \cos\left(\frac{\pi(2x+1)k}{16}\right) \cdot \cos\left(\frac{\pi(2y+1)n}{16}\right) \quad mit\ x,y \in \{0,\ldots,7\}$$

In beiden Gleichungen gilt:

$$C(z) := \frac{1}{\sqrt{2}}\ für\ z = 0,\ sonst\ 1$$

An dieser Stelle vielleicht noch ein, zwei Sätze zu den Gleichungen. Die Werte *k* und *n* sowie *x* und *y* definieren die Position des DCT-Koeffizienten bzw. des jeweiligen Pixels in der 8*8 Matrix. Die Funktion C(z) entspricht nur der allgemeinen Schreibweise für die Gewichtungsfaktoren C(k) und C(n). Im Klartext heißt das: Ist k=0 ist C(k)=0,707 bzw. ist n=0 ist C(n)=0,707. In allen anderen Fällen ist der jeweilige Faktor gleich 1.

Um einen DCT-Koeffizienten zu berechnen erfolgt die Multiplikation jedes einzelnen Helligkeits- oder Farbwertes der Bildmatrix, von (0,0) bis (7,7), mit den entsprechend gewichteten Cosinusfunktionen und deren anschließende Summation. Hier wird auch der hohe Rechenaufwand deutlich. In der Praxis gibt es jedoch optimierte DCT-Implementierungen, die die Anzahl der

durchzuführenden Rechenoperationen minimiert. Anhand der Transformationsgleichungen wird somit auch ersichtlich, dass mit Vergrößerung der Pixelanzahl eines Blocks der Aufwand stark ansteigt und deshalb eine Begrenzung auf 8*8 Pixel sinnvoll ist.

Wie man sich die Arbeitsweise dieser Transformationsgleichungen vorzustellen hat, wird im nächsten Abschnitt genauer erklärt.

Transformationsgleichungen - Was heißt das nun???

Jedes Pixel des entsprechenden Blockes repräsentiert dabei einen Helligkeits- bzw. Farbwert. Trägt man beispielsweise die Helligkeitswerte der ersten Zeile eines 8*8-Blockes gegen die Pixel auf, erhält man im mathematischen Sinne eine Funktion [x-Achse: Pixel 0,... ,7; y-Achse die Grauwerte z.B. auf einer 256 Stufen-Skala (8bit)]. Diese stufenartige Funktion wird bei der DCT durch eine Summe modifizierter Cosinusfunktionen ersetzt.

Bei der graphischen Darstellung eines 8*8-Blocks würde sich hierbei eine dreidimensionale Funktion für jede einzelne der drei Bildkomponenten Y, Cb und Cr ergeben [x- & y-Achse: Pixelkoordinaten; z-Achse: Helligkeits- bzw. Farbwerte]. Da diese Funktion nicht kontinuierlich verläuft, sondern stufenartig (wertediskret), kann man sie auch als 8*8 Matrix darstellen. In dieser Matrix steht jeder Koeffizient für den Helligkeits- oder Farbwertwert des entsprechenden Pixels im 8*8 Block. Jeder dieser 8*8-Blöcke kann durch eine gewichtete Summe der 64 DCT-Basisbilder (*Bild 1*) dargestellt werden, d.h. jedes Basisbild wird mit einem bestimmten Faktor beaufschlagt. Wie man sich die DCT-Basisbilder veranschaulichen kann zeigt folgendes Bild:

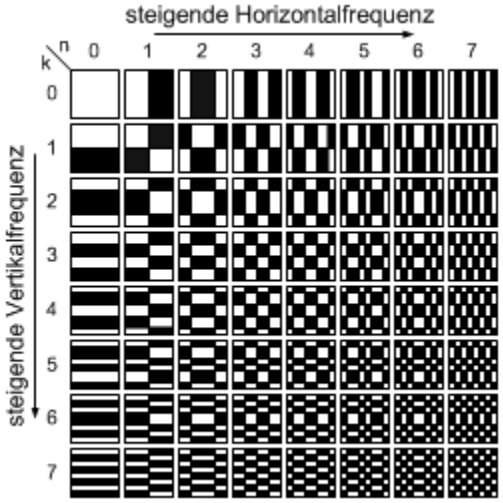


Bild 1: DCT-Basisbilder $M_{k,n}$ mit 8x8 Pixeln

Wichtig ist, dass jede dieser cos-Funktionen Daten für jedes der 64 Pixel des Blockes enthält.

Der Koeffizient links oben (0,0), auch DC-Koeffizient genannt, beschreibt die Grundhelligkeit des Blockes an sich. Der darunter (1,0), den einfachen Helligkeitsverlauf von der ersten bis zur 8. Pixelzeile und der Koeffizient (0,1) den Helligkeitsverlauf von der ersten bis zur 8. Pixelspalte usw. Man sieht, dass die Muster immer detailreicher werden je weiter man nach unten rechts gelangt.Nach der DCT erhält man wieder eine 8*8 Matrix mit den DCT-Koeffizienten (Frequenz-Matrix, FM). Jeder Koeffizient steht jetzt für eine Funktion, die für sich alleine schon einen 8*8 Pixel-Block darstellt.

Um den Ursprünglichen Block zurück zu erhalten, werden alle 64 Cosinusfunktionen mit dem DCT-Koeffizienten in der Matrix multipliziert und aufaddiert, d.h. überlagert. Die einzelnen Koeffizienten geben also an, wie stark die jeweilige Funktion bei der Darstellung des Blockes zu berücksichtigen ist. In Bild 2 ist die gewichtete Überlagerung der einzelnen DCT-Basismuster zur Rekonstruktion des Originalbildblocks dargestellt.

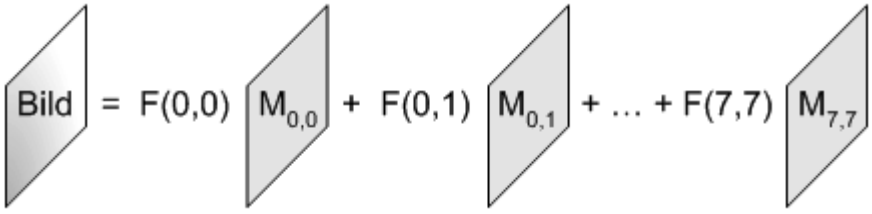


Bild 2: Zusammensetzung des Originalbildblocks aus den DCT-Basisbildern

Und was macht die Quantizer-Matrix (QM) des Codec?

Nachdem man (also eigentlich den abakus digitalis) die DCT durchgeführt hat, hat man, wie oben schon erwähnt, eine Matrix mit 64 DCT-Koeffizienten und noch kein einziges Bit gespart. Im Gegenteil, waren es vor der DCT 8Bit Werte (0...255 bzw. -128...+127 mittelwertfrei), so sind nach der DCT sogar 12Bit (-2048...+2047) zur Speicherung notwendig. Und genau hier kommt die Quantizer Matrix des Codec ins Spiel.

Der Codec dividiert nun jeden Wert der DCT-Koeffizienten Matrix durch den entsprechenden Wert in der Quantizer-Matrix und rundet dann auf den nächstgelegenen Integer, d.h. ganzzahlig. Diesen Vorgang bezeichnet man auch als Quantisierung. Allerdings wird vorher noch die Quantizer-Matrix mit dem Wert, der jedem als Quantizer an sich bekannt ist, multipliziert (Siehe XviD Encoding Mode). Durch dieses Division und die anschließende Rundung wird ein Großteil der DCT-Koeffizienten zu Null. Je größer der Quantizer gewählt wird, desto mehr Koeffizienten der Matrix erhalten den Wert Null. Denn je mehr Nullen in der Matrix enthalten sind, desto effektiver lässt sich der Bildblock komprimieren, vor allem wenn es sich um lange Nullfolgen handelt. Aus diesem Grund wird die Matrix nach einem Zick-Zack Schema ausgelesen (*Bild 3*), d.h. von niedrigen zu hohen Frequenzen hin. Denn mit steigender Frequenz steigt auch die Wahrscheinlichkeit, dass das entsprechende Element den Wert Null besitzt. Die in Bild 3 gezeigte Zick-Zack-Folge wird für progressives Bildmaterial, die alternative Zick-Zack-Folge in Bild 4 für Interlaced-Material verwendet.

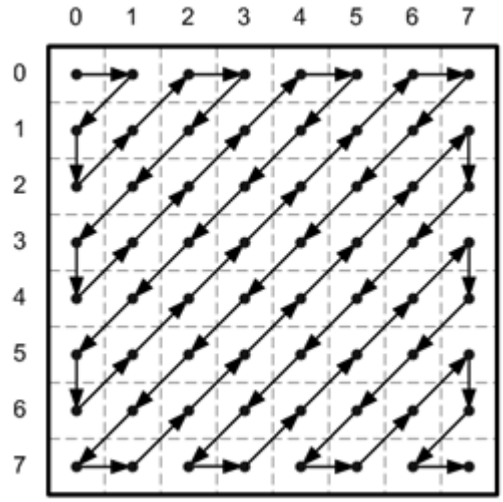


Bild 2: Zick-Zack-Folge (Progressiv)

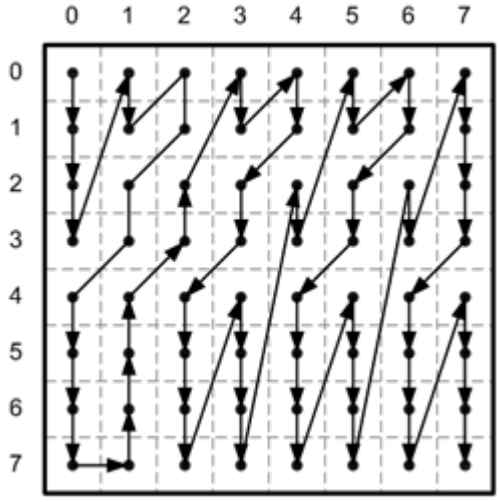


Bild 1: alternative Zick-Zack-Folge (Interlaced)

Die serielle Datenfolge, die man nun erhält, wird noch einer Redundanzreduktion unterzogen.
Wobei zuerst eine Lauflängenkodierung (RLC) und anschließend eine variable Längenkodierung (VLC) zum Einsatz kommt.

Bei der RLC werden ununterbrochene Folgen von Nullen in einer Zahl gespeichert, d.h. in einem Zahlenpaar. Die erste Zahl ist die Anzahl der Nullen, die zweite die eigentliche Zahl nach der Nullfolge. Für den Fall, dass ab einem bestimmten Koeffizienten bei der Zick-Zack-Abtastung nur noch Nullen folgen, wird nicht mehr jeder einzelne Koeffizient als Bitfolge gespeichert, sondern mit einem "end of block" Codewort angegeben, dass der Rest der Koeffizienten Null ist. Am folgenden Beispiel ist eine Bitfolge vor und nach der RLC zusehen:

... 0 11 0 0 1 0 -1 0 0 0 0 2 0 0 0 0 0 0 0 0 2 ...

... (1,11) (2,1) (1,-1) (4,2) (9,2)...

Die anschließende VLC übernimmt dabei ein statistisches Kodierungsverfahren wie die Huffman-Kodierung. Hier werden häufig auftretenden Werten kurze und selten auftretenden Werten lange Codewörter gegeben, vergleichbar mit dem MORSE-Alphabet.

Hieraus wird auch ersichtlich, dass die aktuelle Bitrate über den Quantizer variiert wird: Höherer Quantizer => mehr Koeffizienten =0 => weniger Speicherbedarf => niedrigere Bitrate.

Ich will mein Bild wieder haben!

Soll nun der Block beim Dekodieren wiederhergestellt werden, werden alle Schritte in umgekehrter Reihenfolge ausgeführt. Da hierzu wiederum die QM vonnöten ist, wird diese im Regelfall mit übertragen. Falls es sich um Standardwerte, wie bei der H.263 Matrix handelt, kann auch nur eine entsprechende Kennung gesendet werden. Bei der Wiederherstellung des Bildblocks, nach der iDCT (inverse Diskrete Cosinus Transformation), machen sich nun die Rundungsfehler der Quantisierung bemerkbar. Die Frequenzen, deren Koeffizienten als Null gespeichert wurden, werden bei der iDCT, bei der aus der FM wieder der Bildblock gewonnen wird, also gar nicht erst berücksichtigt.

Wichtig! Die folgende Beispielrechnung soll nur die groben, prinzipiellen Auswirkungen der Rundung darstellen, weil die Rekonstruktion ebenfalls eine gewichtete Überlagerung aller gespeicherten DCT-Koeffizienten darstellt und die Abweichung in der Praxis nicht so berechnet werden kann.

Bei einem Wert von 156 vor der Quantisierung, QM-Koeffizient 32 und Quantizer 2, würde dieser Wert im fertigen Bitstrom als 156/(2*32)=2,4375, also gerundet 2 gespeichert. Beim Rekonstruieren der Frequenz-Matrix während des Dekodierens, ergibt sich daher für den entsprechenden Koeffizienten: 2*2*32=128, was einen Fehler von ca. 20% ausmacht. Bei einem Quantizer von 1 hätte man als quantisierten Wert 4.88, also 5 erhalten. Der rekonstruierte Wert wäre somit 5*1*32=160, was schon wesentlich dichter an der Wahrheit liegt.

Bildblock im Originalbereich

	0	1	2	3	4	5	6	7
0	165	156	160	170	171	168	159	152
1	140	132	136	135	134	145	127	120
2	131	129	127	128	128	128	128	127
3	176	171	185	203	206	203	193	178
4	127	127	127	127	127	122	117	119
5	128	127	127	127	125	121	115	114
6	124	122	122	120	120	121	124	119
7	127	127	127	127	126	126	123	118

H.263 Matrix für Intraframes

	0	1	2	3	4	5	6	7
0	8	17	18	19	21	23	25	27
1	17	18	19	21	23	25	27	28
2	20	21	22	23	24	26	28	30
3	21	22	23	24	26	28	30	32
4	22	23	24	26	28	30	32	35
5	23	24	26	28	30	32	35	38
6	25	26	28	30	32	35	38	41
7	27	28	30	32	35	38	41	45

quantisierter Bildblock nach DCT

	0	1	2	3	4	5	6	7
0	5	0	-1	0	0	0	0	0
1	3	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	2	0	0	0	0	0	0	0
5	2	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	2	0	0	0	0	0	0	0

rekonstruierter Bildblock

	0	1	2	3	4	5	6	7
0	160	163	168	171	171	168	163	160
1	128	132	136	140	140	136	132	128
2	121	125	129	133	133	129	125	121
3	185	189	194	197	197	194	189	185
4	114	117	122	126	126	122	117	114
5	127	130	135	138	138	135	130	127
6	109	112	117	121	121	117	112	109
7	114	117	122	125	125	122	117	114

Bild 3: Beispiel für die DCT

Bild 5 zeigt den Ablauf der Bildkompression an einem realen Beispiel. In der Matrix (links oben) sind die Helligkeitswerte eines Bildblocks dargestellt. Diese werden mit Hilfe der DCT in den Frequenzbereich überführt. Die DCT-Koeffizienten die man nach dieser Transformation erhält werden jetzt durch die Werte der Quantisierungsmatrix (rechts oben) geteilt. In diesem Fall wurde noch ein Quantizer von 2 verwendet, d.h. vor der Divison müssen alle Werte der h.263-Matrix verdoppelt werden. Nach der Quantisierung erhält man die im Bild links unten dargestellte Matrix. Hier ist auch deutlich der Grund für die Quantisierung zu erkennen. Je größer der Quantizer, desto mehr DCT-Koeffizienten werden zu Null.
Das letzte Bild (rechts unten) zeigt den rekonstruierten Bildblock nach der iDCT. An dieser Matrix wird die Verfälschung der Ausgangswerte durch die Quantisierung und anschließende Rundung ersichtlich. Des Weiteren ist auch nachzuvollziehen, dass das oben gezeigte Rechenbeispiel so nicht funktioniert.

In Bild 6.1 ist der Originalbildblock und in Bild 6.2 der rekonstruierte Bildblock stark vergrößert (Faktor 20) zu sehen. Bild 6.3 zeigt die beiden Blöcke in Originalgröße (8*8 Pixel), direkt nebeneinander. Hier ist rein optisch fast kein Unterschied zwischen Originalbildblock und rekonstruiertem Bildblock zu erkennen.

Bild 6.1: Originalbildblock

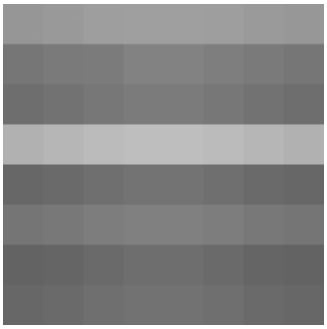


Bild 6.2: rekonstruierter Bildblock

Bild 6.3: Blöcke in Originalgröße

Und was heißt das jetzt für meine Matrix?

Mit diesem Wissen und ein paar Tests, lassen sich folgende Leitlinien für die Entwicklung einer eigenen Matrix erstellen:

1. kleine Werte => weniger Detailverlust, hoher Speicherbedarf
2. beste Komprimierbarkeit, wenn Quantisierungswerte gemäß der Speicherreihenfolge ansteigen
3. übertrieben hohe Werte in den ersten beiden Spalten und Zeilen führen zu verfrühter Blockartefaktbildung

Dies ist sicherlich nicht alles, was es zu beachten gilt, aber zumindest das, was sich einigermaßen leicht in Worten ausdrücken lässt. Ansonsten ist natürlich noch zu sagen, dass man am besten erst einmal selber testet, testet und noch mal testet und noch mal testet. Dafür eignen sich am besten Sequenzen mit scharfen Kanten (hell/dunkel-Übergänge), dunkle Flächen (z.B. Nachtszenarien) und natürlich sehr detailreiches Bildmaterial. Weiterhin ist es hilfreich beim Testen Matrizen zu verwenden, deren Werte bewusst übertrieben sind, einfach um den Einfluß und den Effekt abschätzen zu können, den die Quantisierungsmatrix auf das Endergebnis hat.

So das war's für's erste.

Zu guter Letzt sei noch mal daran erinnert, dass dies nur eine grobe Übersicht zum Thema sein soll. Es kann gut sein, dass sich der eine oder andere Fehler eingeschlichen hat. An dieser Stelle sind wir für Korrekturen dankbar. Ebenso können und wollen wir nicht ausschließen, dass sich manch einem, der tiefere Einblicke in die Materie oder die dahinterstehende Mathematik hat, die Fußnägel rollen, wenn er diese mathematisch sicherlich nicht astrein formulierten Zeilen liest.

Dieser 'Beitrag' erhebt keinen Anspruch auf Vollständigkeit!

Aber wir hoffen, dass diejenigen, die sich für dieses Thema interessieren, ein wenig Einblick in die entsprechende Materie bekommen haben. Wer noch ein wenig mehr lesen möchte, dem seien die folgenden Links empfohlen:

- http://user.cs.tu-berlin.de/~magus/dct/algo.html#DCT
- http://mediasrv.cs.uni-dortmund.de/Lehre/WS2001_02/DV_Seminar_WS01_02/AusarbeitungenPDFs/Koll-MPEG.PDF

Anhang II: Anamorphes Encoding (mit Gordian Knot)[1]

(aus 'Brother Johns gesammeltes Encodingwissen')[2]

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.

1 Siehe: http://sourceforge.net/projects/gordianknot
2 Siehe: http://home.arcor.de/brotherjohn/

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.) an, die ein Videobild aus 4 × 4 Pixeln darstellen soll, wie es korrekt entzerrt beim Abspielen aussieht. Das Bild ist nicht quadratisch, da DVD-Pixel eine rechteckige Form haben.
Die Form des gesamten Bilds und die Form eines einzelnen Pixels lässt sich leicht berechnen:
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. 1).

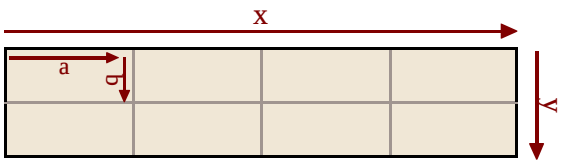


Abbildung 1

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.

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 passende **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.

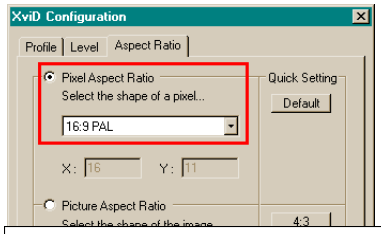
Das zugeschnittene Bild sollte wie beim nicht-anamorphen Encoding ein durch 16 teilbares Seitenverhältnis haben, um dem Codec das Encodieren nicht unnötig zu erschweren. Mit etwas Glück passt das nötige Cropping auch zu dieser mod16-Anforderung. Oft klappt das aber nicht so gut und wir stehen vor der 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.

```
#  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 3

Im **Plugins**-Abschnitt fügen wir wie in Abb. 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.

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.

Encoding und Wiedergabe

Prinzipiell unterscheidet sich das Encoding anamorphen Materials kaum vom nicht-anamorphen 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). Am besten nehmen wir die AR-Einstellung für beide Passes vor, um möglichen Problemen aus dem Weg zu gehen.

Über den ersten **More**-Button der XviD-Konfiguration (Abb. 2) 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.

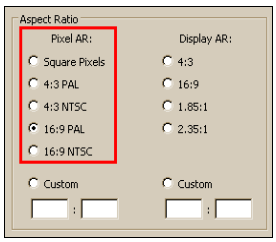


Abbildung 4

Wer lieber mit DivX encodiert, kann das AR-Flag nicht gleich beim Encoding setzen, sondern muss das hinterher mit MPEG4 Modifier[3] 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. Dann öffnen wir die Datei mit MPEG4 Modifier und setzen das **Pixel AR** in Abb. 4 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.

Freunden von x264 bleibt das MPEG-4-AR-Flag leider noch verwehrt. x264 bringt selbst keine Unterstützung mit und MPEG4 Modifier versteht bisher keine AVC-Videospuren. Als Ausweichmöglichkeit bleibt das Setzen des AR-Flags im Container.

Im Matroska-Container setzen wir das AR-Flag beim Muxen mit mkvmerge[4] GUI. VirtualDubMods Matroska-Unterstützung geht leider nicht so weit.

Abbildung 5

In den **Track options** der Videospur (Abb. 5) 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

3 http://www.moitah.net/download/latest/
4 http://www.bunkus.org/videotools/mkvtoolnix/

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 den Container verwenden. Mkvmerge übernimmt das MPEG-4-Flag ja bequemerweise automatisch.

Anhang III: Inhaltsverzeichnis

Anhang IV: Changelog

0.2.6 zu 0.2.5
- mit der finalen 1.1er Version wurde das Frontend überarbeitet,..

0.2.5 zu 0.2.4
- GMC nur aktivieren wenn man keine schwarzen Ränder mehr hat
  (http://forum.doom9.org/showthread.php?s=&threadid=87724)
- neuen Anhang: Anamorphes Encoding aus **B**rother **J**ohns
  gesammeltes **E**ncodingwissen (http://home.arcor.de/brotherjohn/) hinzugefügt.
- Anmerkung zum 'Turbo' modifiziert.
- Closed GOV entfernt, da es nun standardmäßig aktiviert ist und nicht mehr im Forntend enthalten
  ist.

0.2.4 zu 0.2.3
- Link zu LigHs neusten Custom Quantization Matrix Editor 1.0a aktualisiert
- Bild und Text bei den Profiles aktualisiert; DXN Profiles sind dazugekommen
- bei GMC ein Bild eingefügt
- FourCC Beschreibung leicht geändert

0.2.3 zu 0.2.2
- teilweise kleine Umformulierung: *'CPU-Auslastung' => 'Wartezeit beim Encoden'*
- neuen Anhang: Inhaltsverzeichnis eingefügt
- Inhaltsverzeichnis vor Changelog gesetzt
- nicht 'restrict' sondern 'respect' VBV Buffer
- 'Top field first' – Interlacingoption kommentiert

0.2.2 zu 0.2.1
- Meine Vorliebe der B-Frame Einstellungen entfernt, da es zu Verwirrungen führte.
- Encoding Status Beschreibung hinzugefügt
- etwas zu den neuen Features 'loose curve scaling' und 'respect VBV buffer' geschrieben
- Link&Text zu gamr's builds entfernt (http://xvid.gamrdev.com/)

0.2.1 zu 0.2.0
- neuen Anhang II: Changelog ;)
- GMC: Es gibt eigentlich 4 Warp points; vierter würde Perspektivische Verzerrungen ermöglichen, was bei 2D Material allerdings nicht machbar ist
- Adaptive Quantization: Es entsteht kein zusätzlicher Overhead. Die Flags sind immer da und werden nur nicht beachtet.
- packed bitstream: Erzeugt auch keinen Overhead.
- Beschreibung der Overflowsettings hoffentlich verständlicher
- es sind nicht vier sondern sechs Advanced Profile Settings
- Rechtschreib- und Grammatikkorrekturen
- URL zu Koepi's-Seite geändert
- noch einige Kommentare zur Klärung eingefügt