Design, Entwicklung und Evaluation eines Low-Motion Videocodecs

Benjamin Langmann

14. September 2006

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Studienarbeit

Design, Entwicklung und Evaluation eines Low-Motion Videocodecs

von
cand. inform. Benjamin Langmann

Aufgabenstellung und Betreuung:
Prof. Dr. L. Wolf und Dipl.-Inform. Z. Kurtisi

Braunschweig, den 14. September 2006

Erklärung

Ich versichere, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Braunschweig, den 14. September 2006

Kurzfassung

Das Ziel dieser Arbeit ist es, einen Videocodec zu entwickeln, der in der Lage ist Videos effizient zu komprimieren, die weitgehend statisch sind und in denen Bewegungen oder stärkere Bildänderungen hauptsächlich in beschränkten Bereichen vorkommen. Das dabei verwendete Verfahren soll prinzipiell dem Verfahren von MPEG4 gleichen, nur dass dieser Videocodec auf Bewegungskompensation verzichten soll. Als vereinfachte Methode sollen statt dessen nur die Teile eines Bildes abgespeichert werden, die sich im Vergleich zum vorherigen Bild ausreichend stark geändert haben. Im Anschluss wird anhand einer Analyse der Vor- und Nachteile evaluiert, ob dieses Vorgehen bei bestimmten Videotypen im Vergleich zu gängigen Videocodecs sinnvoll ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit der Entwicklung eines Videocodecs, der für Videos mit besonderen Eigenschaften besser geeignet sein soll als gänige, weit verbreitete Videocodecs. Diese erreichen hohe Kompressionsraten, indem sie möglichst viele sogenannte P- oder B-Frames verwenden, die abhängig von früheren oder späteren Frames sind und sich sich nicht eigenständig dekodieren lassen. Die P- und B-Frames enthalten also Informationen zu den Bewegungen und Veränderungen der einzelnen Bildabschnitte im Vergleich zu bestimmten Bezugsbildern; man spricht von Bewegungskompensation. Dabei wird für jeden Bildteil eine Bewegungsrichtung und Informationen über Veränderungen abgespeichert. Diese Methode nutzt, dass sich aufeinander folgende Frames oft sehr ähnlich sind. Dieses Vorgehen ist allerdings insbesondere bei der Enkodierung sehr aufwendig und bei Videos, in denen kaum Bewegung vorkommt auch weitgehend unnötig.
Das in dieser Arbeit implementierte Verfahren ähnelt den gängigen Videocodecs stark, allerdings wird die Bewegungskompensation nicht verwendet, sondern statt dessen werden vereinfacht in den P-Frames nur die Bildteile gespeichert, die sich im Vergleich zu dem Bezugsbild unter einem zu bestimmenden Maß genügend stark geändert haben.
Es ist zu erwarten, dass dieses Vorgehen bei geeigneten Videos eine mit gängigen Videocodecs vergleichbare Kompressionsrate aufweist, aber dass sich gegenüber diesen Geschwindigkeitsvorteile ergeben. Zusätzlich ist es möglich, dass aufgrund der geringeren Abhängigkeiten der Videodaten untereinander Informationsverluste verträglicher sind. Denn im Gegensatz zu Videocodecs, die Bewegungskompensation benutzen, läßt sich bei dieser Methode bei Verlust von Teilen eines Bildes aus den verfügbaren Informationen leichter ein Bild zusammensetzen, da auf Bewegungsangaben keine Rücksicht genommen werden muss.
Als möglicherweise geeignete Videos kommen Videos in Frage, in denen Kameraschwenks, Blenden, Helligkeitsänderungen und andere Bewegungen oder Bildänderungen nur in geringen Maße vorkommen. In diese Kategorie fallen z.B. Vortragsvideos, Vorlesungsaufzeichnungen, Webcam-Videos und Zeichentrickfilme. Daher werden exemplarische Beispiele einiger dieser Videotypen in der vorliegenden Arbeit auf ihre Eignung geprüft. Im Rahmen der Evaluation werden die Ergebnisse des Codecs der vorliegenden Arbeit mit dem freien Codec XviD, der eine Implementierung des MPEG4-Standards ist, verglichen.

2 Design und Implementierung

2.1 Verfahren

Der Codec der vorliegenden Arbeit enkodiert die Bilder eines Videos mit einem auf der JPEG-Komprimierung [1] basierenden Kompressionsverfahren. Dabei wird das zu komprimierende Bild in $8x8$-Pixel große Blöcke unterteilt und für jeden Block wird mit einer Blockauswahlmethode, beschrieben in 2.2.5, bestimmt, wie sehr sich der Block von dem entsprechenden Block des vorangegangenen Bildes unterscheidet. Überschreitet dieser Unterschied einen gegebenen Schwellenwert, wird der Block des aktuellen Bildes abgespeichert. Die übrigen Blöcke werden bei der Enkodierung einfach weggelassen.

2.2 Enkodierung und Dekodierung

Bei der Enkodierung wird zwischen sogenannten I-Frames, die alle Informationen enthalten, die zur Dekodierung erforderlich sind, und P-Frames, die sich nicht eigenständig dekodieren lassen, unterschieden. Die I-Frames werden in regelmäßigen Abständen - dem Keyframe-Intervall - eingesetzt. In den I-Frames wird im Gegensatz zu den P-Frames das komplette Bild gespeichert, also werden bei ihnen keine Blöcke ausgelassen. Bei der Dekodierung werden an Stelle der ausgelassenen Blöcke in den P-Frames die entsprechenden Blöcke des letzten Bildes eingesetzt.
Das Schaubild 2.1 verdeutlicht die Abfolge der einzelnen Schritte der Enkodierung, die in den folgenden Abschnitten detailliert beschrieben werden, während das Schaubild 2.2 analog die Abfolge der einzelnen Schritte der Dekodierung veranschaulicht.

2.2.1 Bildkonvertierung

Bei der Enkodierung wird das Eingabebild in einem der Formate RGB555, RGB24, RBG32, BGR24 oder YUV420, beschrieben in [1], in das interne Bildformat YUV420 konvertiert. Dabei ist das Bild nicht gepackt, wie bei den RGB Formaten üblich, sondern es wird in drei Ebenen gespeichert. Die erste Ebene enthält die Luminanz und die anderen beiden Ebenen die Chrominanz. Mit Hilfe des Subsampling-420 [2] werden so pro Pixel nur 12 Bit benötigt. Im Vergleich zu RGB24 ergibt sich so bereits eine Datenreduktion von 50%. Zusätzlich wird dabei sowohl die Höhe als auch die Breite des Bildes auf Vielfache von 16 vergrößert.
Bei der Dekodierung hingegen wird das interne Bild erst auf die ursprüngliche Größe verkleinert und dann in eines der Ausgabebildformate RGB24, BGR24 und YUV420 konvertiert.

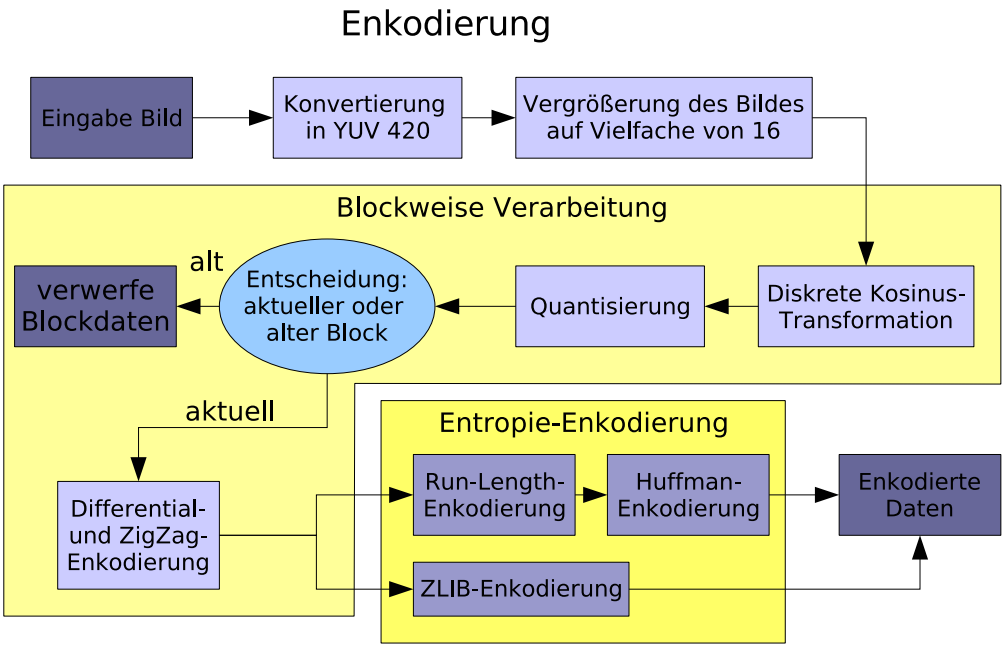


Abbildung 2.1: Schaubild zur Enkodierung

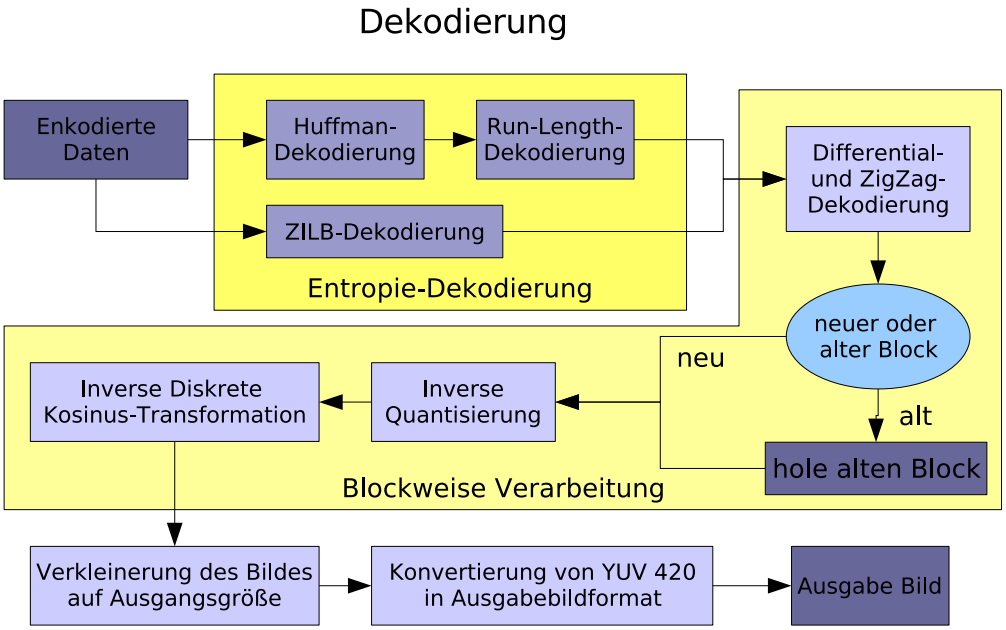


Abbildung 2.2: Schaubild zur Dekodierung

2.2.2 Diskrete Kosinus-Transformation (DCT)

Die Ebenen des Bildes im YUV-Format werden in je $8x8$-Pixel große Blöcke eingeteilt. Bei der Enkodierung wird auf jedem Block die zwei-dimensionale diskrete Kosinus-Transformation (DCT) angewandt, die auch bei der JPEG-Komprimierung benutzt wird und in [1] beschrieben wird. Bei der Dekodierung wird statt dessen die inverse DCT ausgeführt.

2.2.3 Quantisierung

Bei der Quantisierung wird jedes Element eines transformierten Blockes durch einen festen Faktor geteilt. Dazu wird zu Beginn der Enkodierung und Dekodierung aus einer Standard-Quantisierungsmatrix Q mit Hilfe eines gegenbenen Parameters r eine angepasste Quantisierungmatrix Q' berechnet. Wenn A der transformierte $8x8$-Block ist, dann ergibt sich der quantisierte Block B bei der Enkodierung aus

$$B_{ij} = A_{ij}/Q'_{ij},\ 1 \leq i,j \leq 8$$

und bei der Dekodierung ergibt sich aus dem quantisierten Block B der Ausgangsblock A mit

$$A_{ij} = B_{ij} * Q'_{ij},\ 1 \leq i,j \leq 8.$$

Zu den quantisierten Daten, also den Einträgen von B, wird zum Schluss 2048 addiert und es wird sichergestellt, dass sie danach in dem Wertebereich von 0 bis 4095 liegen, damit diese Werte als positive Ganzzahlen mit 12-Bit Genauigkeit weiterverarbeitet werden können.
Als Standard-Quantisierungsmatrix wird in der vorliegenden Arbeit die folgende Matrix verwendet (XviDs MEPG-Intra-Standardmatrix):

$$Q = \begin{pmatrix} 8 & 17 & 18 & 19 & 21 & 23 & 25 & 27 \\ 17 & 18 & 19 & 21 & 23 & 25 & 27 & 28 \\ 20 & 21 & 22 & 23 & 24 & 26 & 28 & 30 \\ 21 & 22 & 23 & 24 & 26 & 28 & 30 & 32 \\ 22 & 23 & 24 & 26 & 28 & 30 & 32 & 35 \\ 23 & 24 & 26 & 28 & 30 & 32 & 35 & 38 \\ 25 & 26 & 28 & 30 & 32 & 35 & 38 & 41 \\ 27 & 28 & 30 & 32 & 35 & 38 & 41 & 45 \end{pmatrix}$$

Die angepasste Quantisierungsmatrix Q' kann aus Q auf zwei verschiedene Arten bestimmt werden.

Parameter in Prozent

Mit einem Qualitätsparameter r wird Q' wie folgt berechnet:

$$p = \lfloor \tfrac{(0.1-5)}{99} * r + 5 + 0.5 \rfloor \text{ und } Q'_{ij} = \lfloor Q_{ij} * p + 0.5 \rfloor \text{ für } 1 \leq i,j \leq 8.$$

Dabei sind ganzzahlige Werte für r von 0 bis 99 erlaubt. Es wird so p linear von 5 bis 0.1 skalliert, dabei wird aber sichergestellt, dass $1 \leq Q'_{ij} \leq 63$ für $1 \leq i,j \leq 8$.

Weiterhin ist es auch möglich Q' direkt anzugeben.
Wird $r = 100$ angegeben, soll der im Folgenden beschriebene verlustfreie Modus angewandt werden.

Quantizer

Diese Quantisierungsmethode entspricht weitgehend der Quantisierung von Intra-Frames bei MPEG4 und kann hier [3] eingesehen werden.
Vereinfacht durch das Weglassen von Rundungen wird die angepasste Quantiserungsmatrix Q' wie folgt berechnet:

$$Q'_{ij} = \frac{q * Q_{ij}}{8}$$

Der Wertebereich des Quantizers q ist $1 <= q <= 31$. Skaliert wird hierbei also von $\frac{1}{8}$ bis $\frac{31}{8}$.

2.2.4 Verlustfreier Modus

Statt die diskrete Kosinus-Transformation und die anschließende verlustbehaftete Quantisierung durchzuführen, ist es auch möglich diese Schritte auszulassen. Verlust von Bildinformationen tritt dann nur bei der Bildkonvertierung und eventuell bei dem Auslassen von Blöcken (siehe 2.2.5) auf. Die Daten werden dann im Gegensatz zum verlustbehafteten Modus nicht mit 12-Bit sondern mit 8-Bit pro Blockeintrag als positive Ganzzahlen von 0 bis 255 weiterverarbeitet.

2.2.5 Blockauswahl

Bei der Enkodierung wird in diesem Schritt für jeden Block eines P-Frames entschieden, ob der Block des aktuellen Bildes abgespeichert werden soll oder ob der entsprechende Block des letzten Bildes bei der Dekodierung an dieser Stelle eingesetzt werden soll. Dazu ist es nötig, sowohl bei der Enkodierung als auch bei der Dekodierung das letzte Bild vorzuhalten.
Das Maß, mit dem die Entscheidung getroffen wird, unterscheidet zwischen verlustfreiem und nicht verlustfreiem Modus. Bei dem verlustfreien Modus werden die Differenzen der Einträge der beiden Blöcke gebildet und es wird über deren Beträge summiert. Wenn diese Summe einen gegebenen Schwellenwert überschreitet, wird der aktuelle Block und sonst der Block der letzten Bildes verwendet.
Im verlustbehafteten Modus werden die Einträge zusätzlich vor dem Summieren gewichet. Die Gewichte werden in Matrixform angegeben und die Standard-Wertungsmatrix W ist die Einheitsmatrix.
Ist B der aktuelle quantisierte Block und C der entsprechende Block der letzten Bildes, wird ein Wert d wie folgt berechnet:

$$d = \sum_{i=1}^{8} \sum_{j=1}^{8} |C_{ij} - B_{ij}| * W_{ij}$$

Ist d größer als ein gegebener Schwellenwert wird der Block B des aktuellen Bildes abgespeichert. Handelt es sich beim aktuellen Bild um ein I-Frame, wird immer dieser aktuelle Block gespeichert.

Wie bereits angesprochen werden bei der Dekodierung dann an Stelle der ausgelassenen Blöcke die entsprechenen Blöcke des letzten Bildes eingesetzt.

2.2.6 ZigZag- und Differential-Enkodierung

Die Differential-Enkodierung nutzt aus, dass benachbarte Blöcke sich oft ähnlich sind. Außerdem ist der linke obere Eintrag eines jeden Blockes, auch DC-Element genannt, oft mit Abstand der größte. Dies wirkt sich negativ auf den Speicherverbrauch bei der späteren Entropie-Enkodierung aus. Daher wird bei der Differential-Enkodierung nur der erste Block einer Bildebene normal abgespeichert und für alle Folgenden wird das DC-Element des Blockes nur als Differenz zum DC-Element seines linken Nachbarblockes festgehalten. Im verlustfreien Modus wird die Differential-Enkodierung nicht angewendet.

Die ZigZag-Enkodierung ordnet die Einträge eines $8x8$-Blockes nach dem folgenden Schema in ein 64-Einträge langes Feld um. Das Ziel dabei ist, dass die Einträge im resultierenden Feld möglichst absteigend sortiert sind. Dies ist durch eine einfache Umortnung möglich, da in den transformierten Daten tendenziell oben links die größten Werte stehen und unten rechts die kleinsten Werte, von denen viele nach der Qantisierung gleich Null sind. Diese Nullen sind später durch die RLE-Enkodierung gut zusammenfaßbar.

$8x8$ Block:

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

64 Einträge langes Feld:

1	2	3	4	...	62	63	64

Die umgeordneten Blöcke des Bildes werden im Anschluss aneinandergefügt und im Folgenden als Frame-Daten bezeichnet.

2.2.7 Entropie-Enkodierung

Bei der Entropie-Enkodierung werden die Daten ohne Informationsverlust weiterverarbeitet. Es stehen zwei verschiedene Methoden zur Auswahl, nämlich Run-Length-Enkodierung mit nachfolgender Huffman-Enkodierung oder alternativ Kompression mit Hilfe der ZLIB-Bibliothek.

Run-Length-Enkodierung (RLE)

Bei der Run-Length-Enkodierung werden aufeinander folgende Nullen der Frame-Daten zusammengefasst. Da die Frame-Daten aber geshiftet wurden, also immer positiv sind, entspricht die Null im verlustbehafteten Modus der 2048 und im verlustfreien

Modus der 128. Dabei werden die Frame-Daten weiterhin nur als Zahlen behandelt, die im verlustbehafteten Modus zwischen 0 und 4095 liegen und im verlustfreien Modus zwischen 0 und 255. Wenn also in den Frame-Daten $n \geq 1$ mal die 2048 vorkommt, werden diese ersetzt durch $(2048, n)$. Wenn im verlustbehafteten Modus $n > 4095$ ist, werden die Nullen durch $(0, 4095, 0, n - 4095)$ ersetzt, beim verlustfreien Modus entsprechend durch $(0, 255, 0, n - 255)$, wenn $n > 255$ ist.

Huffman-Enkodierung

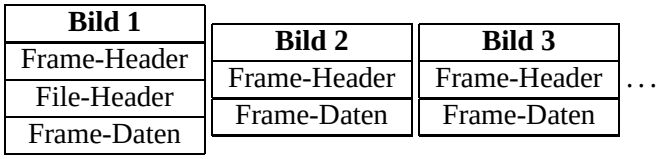
Die Huffman-Enkodierung [4] ersetzt jede Zahl der RLE-Daten durch eine Bitfolge unterschiedlicher Länge. Die Länge der Bitfolge orientiert sich an der Höhe der Wahrscheinlichkeit des Vorkommens der Zahl. In dieser Arbeit wird eine statische Zuordnung von Zahlen und Bitfolgen aus Geschwindigkeitsgründen benutzt.
Beim verlustfreien Modus wird die Huffman-Enkodierung aber nicht durchgeführt, da hierdurch im Allgemeinen keine Speicherreduzierung erreicht werden kann. Denn bei reinen YUV-Bilddaten sind die Werte relativ gleichverteilt und daher ergibt sich leicht eine Huffmantabelle, die jeden Byte-Wert durch einen anderen Byte-Wert ersetzt und so wird durch die Huffman-Enkodierung kein Vorteil erzielt.

ZLIB-Kompression

Bei der ZLIB-Kompression werden die Frame-Daten einfach an die ZLIB-Bibliothek [5] weitergereicht. Dies ist besonders im verlustfreiem Modus sinnvoll, da für reine YUV-Daten die RLE-Kompression nur wenig effektiv ist und die Huffman-Enkodierung aus denselben Gründen, wie oben erwähnt, nicht durchgeführt wird.

2.3 Interne Datenstruktur

Die enkodierten Bilder werden nach dem folgendem Schema abgespeichert.



Der File-Header enthält die folgenden Informationen.

Name	Variablentyp	Beschreibung
`marker`	`int32_t`	Identifizierung des File-Headers
`width`	`int32_t`	Bildbreite
`height`	`int32_t`	Bildhöhe
`topdown`	`int32_t`	Top-Down- oder Buttom-Up-Format
`time_base_num`	`int32_t`	Framerate-Zähler
`time_base_den`	`int32_t`	Framerate-Nenner
`keyframe_intervall`	`int32_t`	Keyframe-Intervall
`sampling`	`int32_t`	Sampling-Verfahren
`quality_mode`	`int32_t`	Blockauswahlmethode
`quality_num`	`int32_t`	Parameter der Blockauswahlmethode
`quality_diff`	`int32_t`	Parameter der Blockauswahlmethode
`jpeg_quality`	`int32_t`	JPEG-Qualitäts Parameter
`pix_format`	`int32_t`	Internes Bildformat
`compression`	`int32_t`	Kompressionsverfahren
`myqm[64]`	`uint8_t`	Quantisationsmatrix
`ratings[64]`	`uint8_t`	Wertungsmatrix

Und der Frame-Header enthält die folgenden Informationen.

Name	Variablentyp	Beschreibung
`marker`	`uint16_t`	Identifizierung des Frame-Headers
`frame_num`	`uint16_t`	Bildnummer
`size`	`uint32_t`	Größe der Bilddaten in Bytes

Die Frame-Daten haben die folgende Struktur:

Block 1	**Block 2**	**Block 3**	
Block-Header	Block-Header (ausgelassen)	Block-Header	...
Block-Daten	—	Block-Daten	

Dabei besteht der Block-Header aus nur einem Byte, das bestimmt, ob es sich um einen neuen Block handelt oder ob der Block ausgelassen wurden.

2.4 Form und Anbindungen des Codecs

Der Videocodec ist als Bibliothek implementiert und verfügt über zwei verschiedene Anbindungen, nämlich einer Implementierung der „Video For Windows“-Schnittstelle von Microsoft und einer Kommandozeilen-Schnittstelle für das Betriebssystem Linux. Mit beiden Anbindungen ist es möglich, Videos in dem Format der vorliegenden Arbeit zu erzeugen, die in einem AVI-Container [6] abgespeichert werden.

2.4.1 „Video For Windows“-Wrapper

Bei der „Video For Windows“-Schnittstelle bestehen die folgenden Konfigurationsmöglichkeiten:

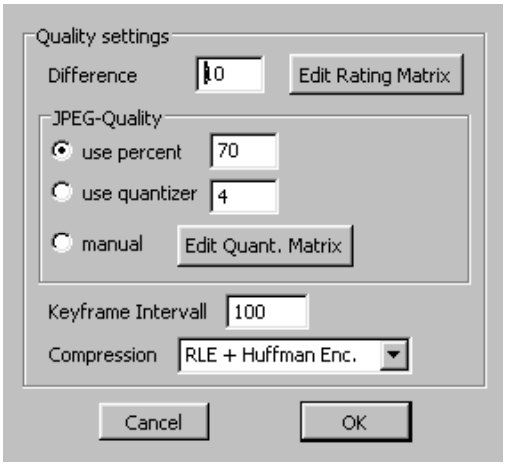


Abbildung 2.3: Konfigurationsdialog unter Windows

- Schwellenwert bei der Blockauswahlmethode, muss eine Ganzzahl größer oder gleich 1 sein, siehe 2.2.5. Dabei bedeutet 1, dass sich bei der Standardwertungmatrix nur ein Pixel eines Blockes von dem des vorherigen Bildes um 1 unterscheiden darf, sonst wird der neue Block verwendet. Beim verlustbehafteten Modus werden dazu die quantisierten Daten betrachtet.
- Steuerung der Quantisierung durch eines der Folgenden:
  - Quantisierungsparameter in Prozent (p): zulässig sind Werte von 0 bis 100, dabei bedeutet 100, dass der verlustfreie Modus angewendet werden soll.
  - Quantizer: zulässig sind Ganzzahlen von 1 bis 31.
  - Manuelle Eingabe der Quantisierungsmatrix
- Entropie-Kompressionsmethode, entweder Run-Length-Enkodierung und Huffman-Enkodierung oder ZLIB-Kompression.
- Keyframe-Intervall, dabei bedeutet 0 keine Keyframes innerhalb des Videos und der Standardwert ist 100.

2.4.2 Kommandozeilen-Schnittstelle

Die Kommandozeilen-Schnittstelle unterstützt sowohl das Konvertieren von fremden Videoformaten in das Format der vorliegenden Arbeit (`write`), als auch eine Konvertierung in die andere Richtung (`read`). Dazu nutzt es die Bibliotheken „libavformat“und „libavcodec“aus dem FFMPEG-Paket [7]. Die Syntax zum Lesen und Schreiben von Videos in dem Format der vorliegenden Arbeit lautet:

```
lmjpeg [options] read  in.lmjpeg out.avi
       [options] write in.avi    out.lmjpeg
```

Beim Schreiben stehen die folgenden Optionen zur Verfügung:

- `-quality y z`

  Schwellenwerte der Blockauswahlmethode, siehe 2.2.5. Dabei wird y ignoriert und z muss eine Ganzzahl größer oder gleich 1 sein. Der Standardwert ist $z = 10$.

- `-jpeg_quality x`

  Quantisierungsparameter in Prozent. Zulässige Werte sind Ganzzahlen von 1 bis 100., dabei bedeutet 100, dass der verlustfreie Modus angewendet werden soll. Der Standardwert ist $x = 70$.

- `-mpeg_quant x`

  Quantizer: zulässig sind Ganzzahlen von 1 bis 31. Diese Option schließt die Option `-jpeg_quality x` aus.

- `-keyframe x`

  Keyframe-Intervall, dabei bedeutet $x = 0$ keine Keyframes innerhalb des Videos und der Standardwert ist $x = 100$.

- `-pix_format x`

  Eingabe Bildformat der Bibliothek der vorliegenden Arbeit. Zur Auswahl stehen YUV, RGB und BGR. Der Standardwert ist YUV. Diese Option ist hauptsächlich für Testzwecke geeignet. Das interne Bildformat bleibt YUV.

- `-compression x`

  Entropie-Kompressionsmethode, entweder RLE also Run-Length-Enkodierung und Huffman-Enkodierung oder ZLIB also die ZLIB-Kompression. Der Standardwert ist RLE.

Beim Lesen wird eine AVI-Datei im XviD [8] Format erzeugt; dabei stehen die folgenden Optionen zur Verfügung:

- `-bitrate x`

  Bitrate, die verwendet werden soll. Der Standardwert ist $x = 700$.

- `-keyframe x`

  Keyframe-Intervall, das verwendet werden soll. Bei XviD ist dies aber ein maximaler Wert; XviD setzt I-Frames nach Belieben. Weiter bedeutet auch hier $x = 0$, dass keine erzwungenen Keyframes innerhalb der Videos gesetzt werden müssen. Der Standardwert ist $x = 100$.

- `-pix_format x`

  Ausgabe Bildformat der Bibliothek der vorliegenden Arbeit. Zur Auswahl stehen YUV, RGB, BGR, RGB32 und RGB555. Der Standardwert ist YUV. Diese Option ist hauptsächlich für Testzwecke geeignet. Das interne Bildformat bleibt auch hierbei YUV.

3 Evaluation

Es soll im Folgenden Abschnitt gezeigt werden, dass der Codec der vorliegenden Arbeit bei geeigneten Videos Ergebnisse erzielen kann, die sowohl beim Speicherplatzverbrauch als auch bei der visuellen Qualität mit denen von XviD vergleichbar sind. Der Zeitbedarf zur Enkodierung wird dabei nicht verglichen, da trotz der niedrigeren Komplexität dieses Codecs dieser im Vergelich zu XviD erheblich mehr Zeit benötigt, da XviD viel höher optimiert ist. Weiterhin geht der verlustfreie Modus nur im Rahmen des visuellen Qualitätsvergleichs in die Evaluation ein, da die Resultate hierbei $3 - 5$-mal mehr Speicherplatz benötigen als die höchsten Qualitätseinstellungen im verlustbehafteten Modus oder bei XviD.
Dazu werden im Folgenden vier verschiedene Videos untersucht, die sich vorraussichtlich sehr unterschiedlich gut für den Videocodec der vorliegenden Arbeit eignen.

3.1 Testvideos

Die ersten drei der ausgesuchten Videos sollten sich gut für den Codec der vorliegenden Arbeit eignen. In ihnen kommen keine Kameraschwenks vor und auch nur wenige Bewegungen, so dass die Bewegungskompensation von XviD nur wenig effektiv sein sollte. Im vierten Video kommen einige Kameraschwenks und viele Bewegungen vor, für die sich die Bewegungskompensation lohnen sollte.

3.1.1 Vorlesungsvideo

In diesem Video treten nur wenige unterschiedliche Bilder auf, es gibt nahezu keine Bewegungen, das Video entstammt einer digitalen Quelle und es wurde verlustfrei gespeichert, daher tritt auch kein Pixelrauschen auf.

Video Typ	Vorlesungvideo einer Multimedia Systeme Vorlesung
Quellformat	TSCC Codec (verlustfrei)
Auflösung	1000x700
Länge	16:40 s, 1000 Frames
Framerate	1 fps

3.1.2 Vortragsvideo

In diesem Ausschnitt eines Vortragsvideos kommen Bewegungen fast nur in einem beschränken Bereich vor, es gibt auch keine Szenenwechsel, aber trotz der Quelle DVD und des Quellcodecs MPEG2 sowie hoher Bitrate gibt es subjektiv einiges Pixelrauschen.

Video Typ	Ausschnitt aus der Fernsehsendung „Alpha Centauri“
Quellformat	MPEG 2, DVD-Aufnahme, (deinterlaced)
Auflösung	720x576
Länge	30 s, 751 Frames
Framerate	25 fps

3.1.3 Nachrichtensendung

Auch in diesem Video gibt es nur wenig Bewegung und diese auch nur in beschränkten Bereichen. Die Aufnahmequalität ist hoch und das Pixelrauschen ist subjektiv geringer als im Vortragsvideo.

Video Typ	Ausschnitt aus dem „Heute Journal“des ZDFs
Quellformat	MPEG 2, DVB-Aufnahme, (deinterlaced)
Auflösung	704x576
Länge	46 s, 1159 Frames
Framerate	25 fps

3.1.4 Werbefilm

In diesem Video gibt es nun viele Szenenwechsel, starke Bewegungen und Zooms.

Video Typ	Teaser des Kinofilms „Terminal“
Quellformat	Apple Quicktime, (deinterlaced)
Auflösung	640x352
Länge	60 s, 1461 Frames
Framerate	23.976 fps

3.2 Durchführung der Testläufe

Im Folgenden werden Einstellung und Konventionen zur Durchführung der Testläufe beschrieben. Angaben zur Datenrate schließen die Audiodaten nicht mit ein.

3.2.1 Codec der vorliegenden Arbeit

Die folgenden Einstellungen wurden für die Testläufe mit dem Codec der vorliegenden Arbeit verwendet:

- Schwellenwerte der Blockauswahlmethode: 1, 5, 10, 15, 30 und 50.
- Quantisierung mit festen Quantizern: 2, 4, 6, 8, 12, 18, 24 und 31.
- Keyframe-Intervalle 50, 100, 200, 0 (also keine Keyframes nach dem ersten).
- Entropie-Enkodierung mit RLE- und Huffman-Enkodierung, ZLIB-Kompression.

3.2.2 XviD

Die Testläufe wurden mit XviD in der Version 1.0 durchgeführt, dabei wurden die Standardeinstellungen bis auf die Folgenden beibehalten:

- Quantisierung mit festen Quantizern: 2, 4, 6, 8, 12, 18, 24 und 31.
- Keyframe-Intervalle 50, 100, 200, 100000 (also keine erzwungenen Keyframes nach dem ersten).
- MPEG-Quantisierung
- keine Trellis-Quantisierung
- keine adaptive Quantisierung
- keine B-Frames
- dieselbe Quantisierungmatrix für I-Frames und P-Frames (die auch in der vorliegenden Arbeit verwendet wurde).

Diese Eigenschaften wurden gewählt, damit die Ergebnisse besser vergleichbar zum Codec der vorliegenden Arbeit werden und so ein Vergleich der unterschliedlichen Kompressionsmethoden ermöglicht wird. Die Angaben zum Keyframe-Intervall beziehen sich auf die maximale Länge des Keyframe-Intervalls, da XviD die Keyframes beliebig und abhängig von den Videoinhalten setzt.

3.3 Ergebnisse

Die Abbildungen 3.1 bis 3.4 geben die Ergebnisse der Enkodierung von den vier verschienden Videos mit Quantizern von 2 bis 12 und bei RLE- und ZLIB-Kompression im Vergleich mit XviD wieder. Für höhere Quantizer setzt sich der Graph geradlinig fort. Es wird bei allen Graphen ein Keyframe-Intervall von 100 verwendet, während die Schwellenwerte jeweils so gewählt wurde, dass die resultierenden Graphen möglichst nahe an dem entsprechenden Graphen von XviD liegen.

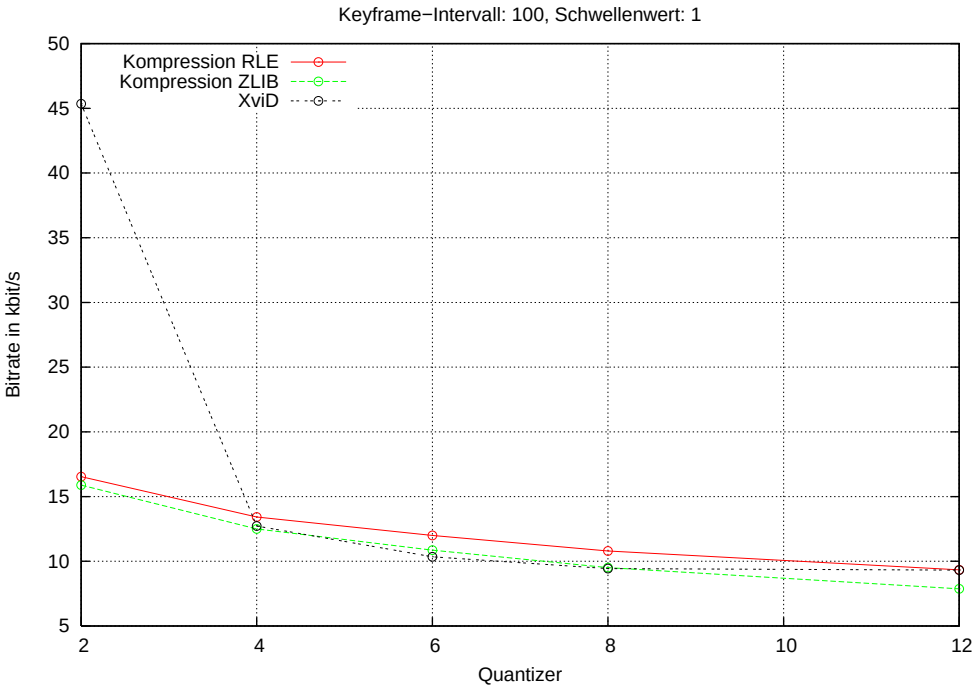


Abbildung 3.1: Vorlesungsvideo bei verschiedenen Kompressionsverfahren

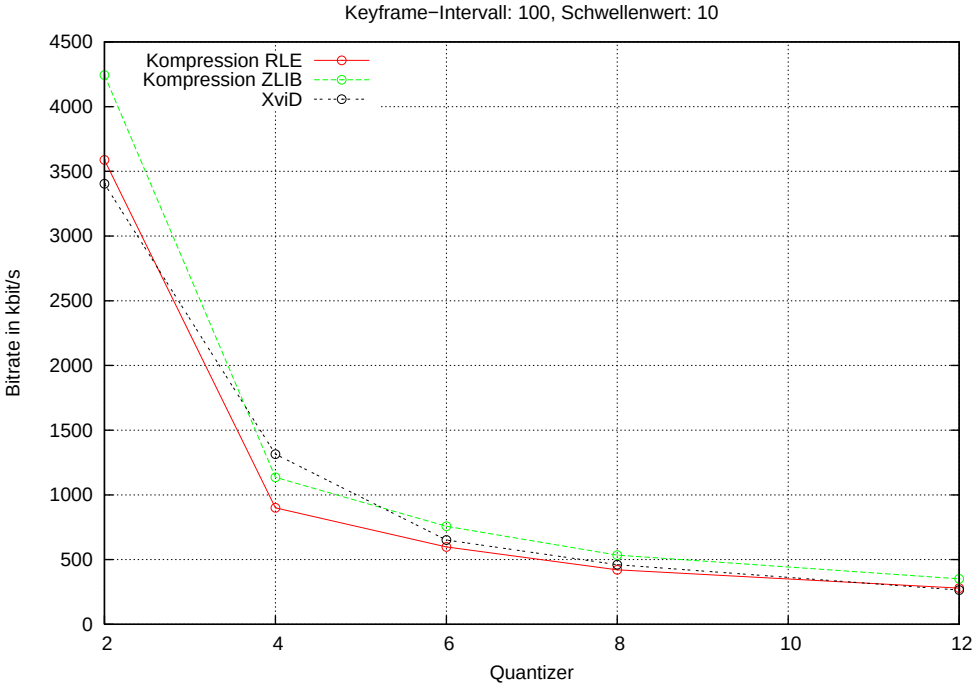


Abbildung 3.2: Vortragsvideo bei verschiedenen Kompressionsverfahren

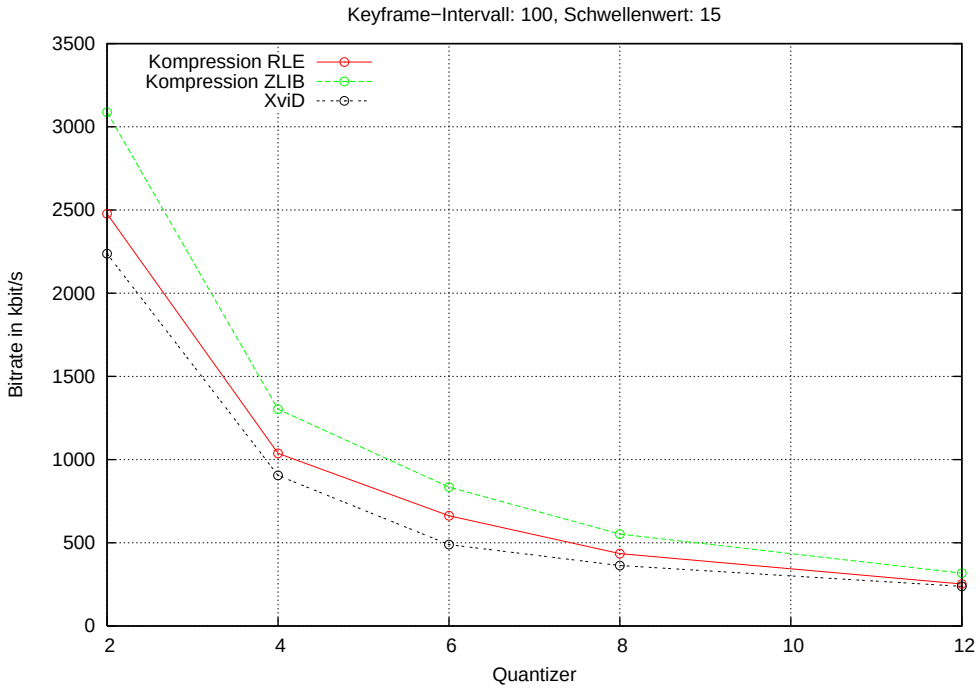


Abbildung 3.3: Nachrichtenvideo bei verschiedenen Kompressionsverfahren

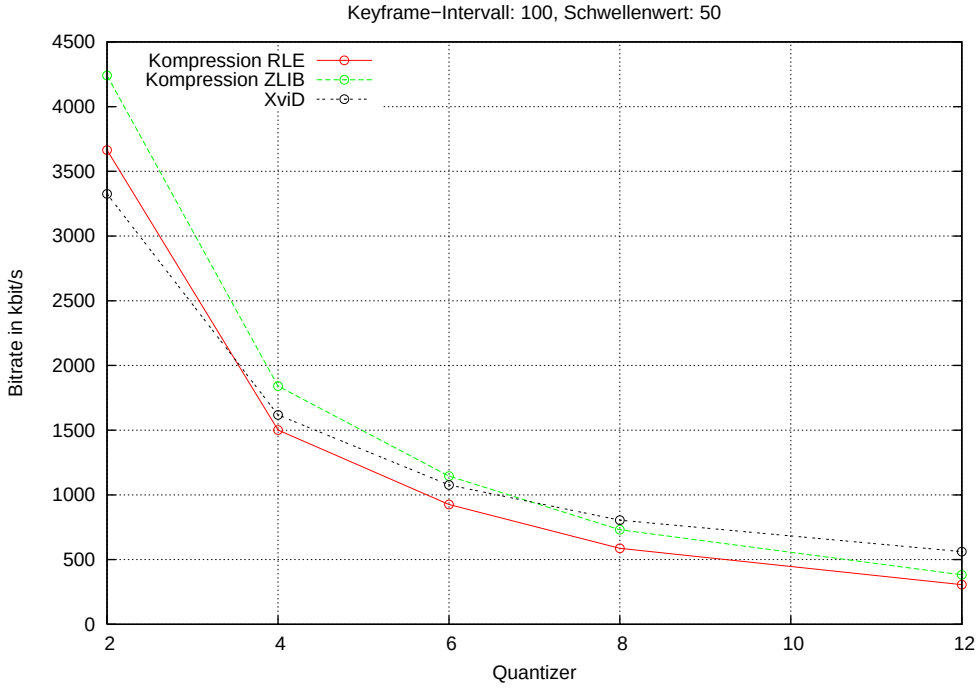


Abbildung 3.4: Werbevideo bei verschiedenen Kompressionsverfahren

Die Graphen 3.5 bis 3.8 stellen die Ergebnisse der Enkodierung von den Videos mit Quantizern von 2 bis 12 und Schwellenwerte von 5 bis 50 im Vergleich mit XviD dar. Auch hier setzt sich der Graph für höhere Quantizer geradlinig fort. Es wird dabei die RLE- und Huffman-Enkodierung und auch ein Keyframe-Intervall von 100 benutzt.

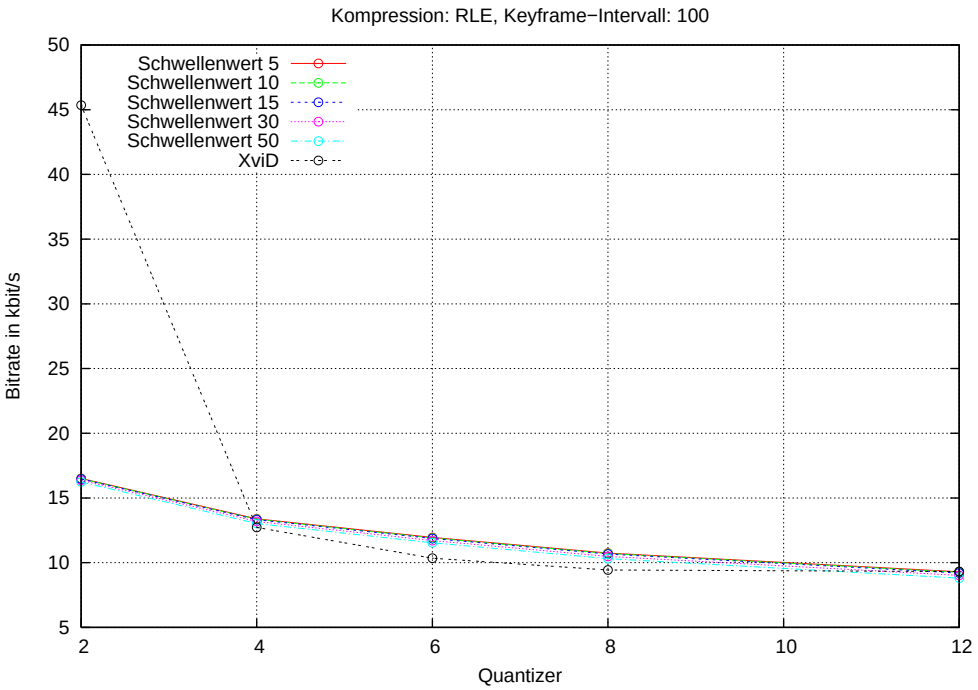


Abbildung 3.5: Vorlesungsvideo bei verschiedenen Qualitäten

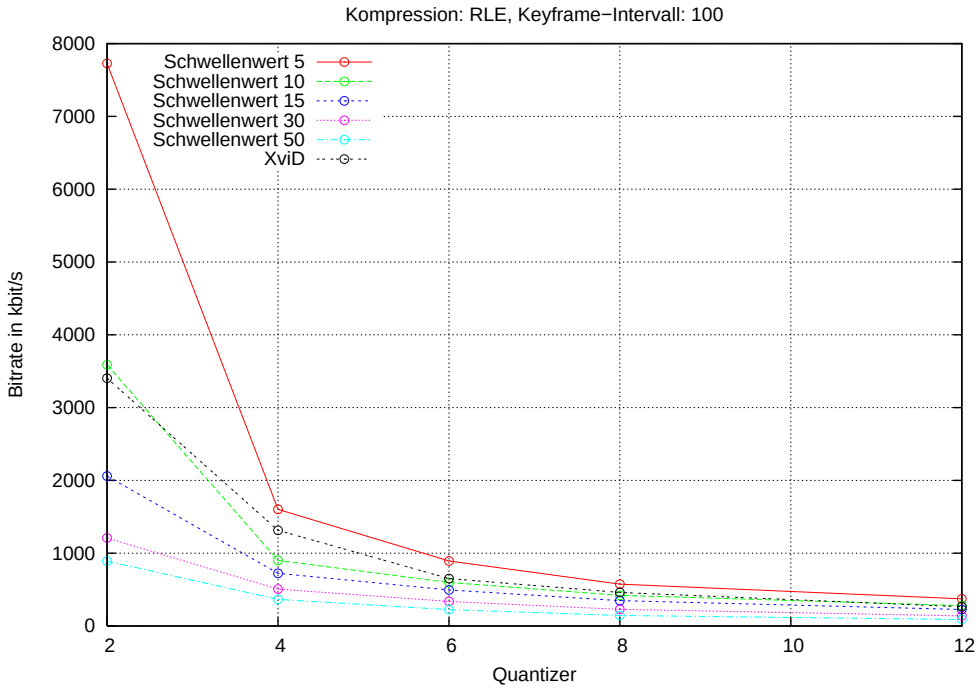


Abbildung 3.6: Vortragsvideo bei verschiedenen Qualitäten

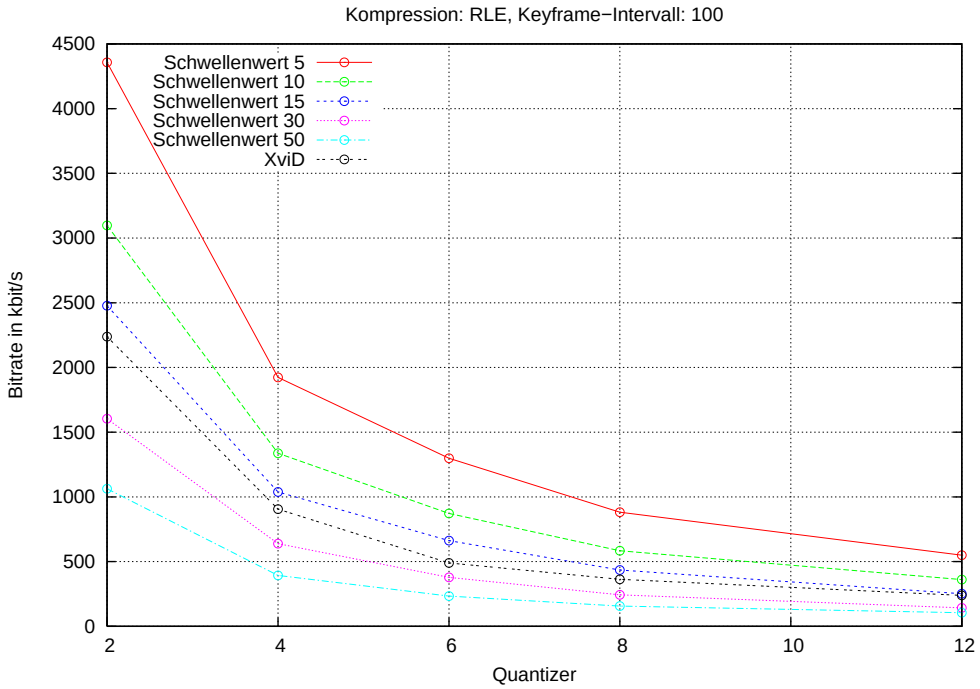


Abbildung 3.7: Nachrichtenvideo bei verschiedenen Qualitäten

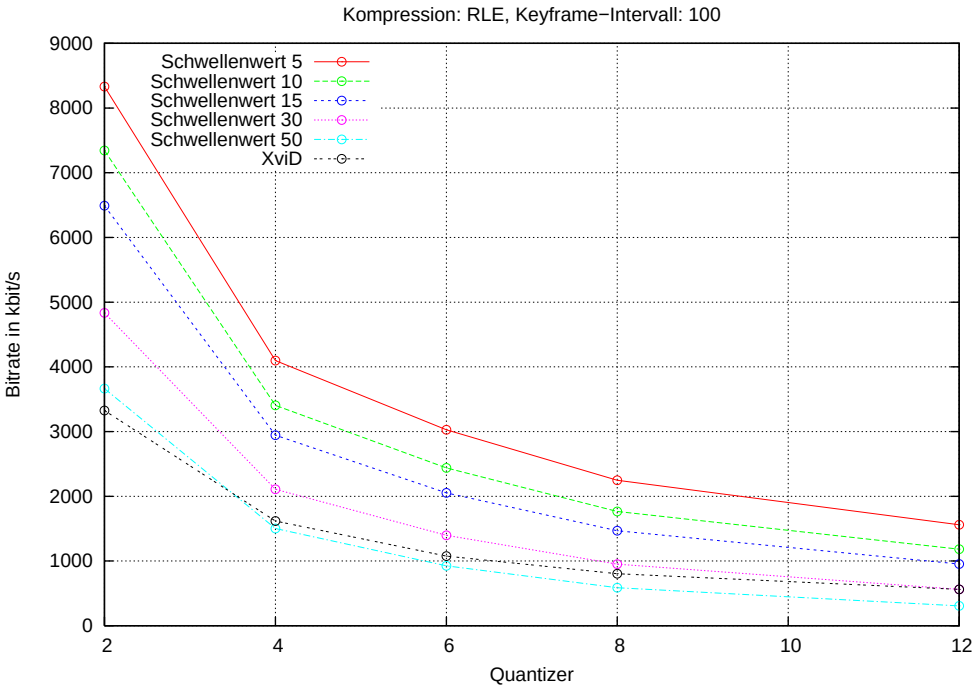


Abbildung 3.8: Werbevideo bei verschiedenen Qualitäten

3.4 PSNR-Messung

Die folgenden PSNR-Ergebnisse für die Testvideos wurden mit dem „MSU Video Quality Measurement Tool“[9] bestimmt. Dabei bezeichnet „LMJpeg“den Codec der vorliegenden Arbeit und „SW“den Schwellenwert der Blockauswahlmethode. Die PSNR-Angaben sind die Durchschnittswerte der Fehlermessungen und wurden anhand des Helligkeitsanteils (Y) von den YUV-Bildern der Videos im Vergleich zu den originalen Bildern errechnet.

Vorlesungsvideo, Keyframe-Intervall 100

Quantizer	Verlustfrei	1	4	8	18
LMJpeg SW: 1:	19.93114	19.92068	19.97411	20.05337	20.16833
LMJpeg SW: 10:	19.93113	19.91999	19.97366	20.05267	20.15873
LMJpeg SW: 30:	19.93110	19.90661	19.95824	20.02773	20.08066
XviD	-	12.03456	12.04745	12.06590	12.11839

Vortragsvideo, Keyframe-Intervall 100

Quantizer	Verlustfrei	1	4	8	18
LMJpeg SW: 1:	57.96832	47.11877	44.32549	41.12159	37.77446
LMJpeg SW: 10:	57.55043	46.34899	41.73534	38.99947	35.19843
LMJpeg SW: 30:	56.58356	41.92895	38.27981	34.45028	30.10494
XviD	-	35.41861	43.64835	41.15088	38.22939

Nachrichtenvideo, Keyframe-Intervall 100

Quantizer	Verlustfrei	1	4	8	18
LMJpeg SW: 1:	36.38681	47.30541	42.91050	38.59623	34.61771
LMJpeg SW: 10:	55.55743	45.84666	36.12766	36.35521	31.86539
LMJpeg SW: 30:	54.08656	42.31287	35.39433	31.82785	28.11553
XviD	-	49.23619	42.14468	38.55433	34.60629

Werbevideo, Keyframe-Intervall 100

Quantizer	Verlustfrei	1	4	8	18
LMJpeg SW: 1:	48.18440	45.43353	41.34584	38.69101	33.53603
LMJpeg SW: 10:	49.13184	45.11536	37.19710	34.56368	29.44467
LMJpeg SW: 30:	48.87288	40.47857	30.54838	26.39062	21.15498
XviD	-	49.18948	41.69905	37.81861	33.80407

3.5 Diskussion der Ergebnisse

Die visuelle Qualität der von dem Codec der vorliegenden Arbeit und von XviD erzeugten Videos sollte bei gleichem Quantizer an den Keyframes übereinstimmen. Bei kleinen Schwellenwerten für den Codec der vorliegenden Arbeit bleibt die visuelle Qualität auch bei den P-Frames konstant. Aber bei steigenden Schwellenwerten ist ein visueller Qualitätsverlust gut erkennbar, insbesondere bei höheren Quantizern, denn dabei werden bei gleichem Schwellenwert mehr Blöcke ausgelassen.
Beim Vorlesungsvideo aber ist der Schwellenwert kaum von Bedeutung, siehe Abbildung 3.5, und bis auf bei Quantizer 2 sind die Ergebnisse für alle Schwellenwerte sehr ähnlich zu denen von XviD, siehe Abbildung 3.1. Der kleine Einfluß der Schwellenwerte ist leicht mit den besonderen Eigenschaften des Videos zu erklären, da Bildänderungen hier fast nur sehr abrupt auftreten. Dass sich dennoch besonders bei kleinen Quantizern nicht unerhebliche Unterschiede bei dem Codec der vorliegenden Arbeit und XviD zeigen, liegt an den verschiedenen Entropie-Enkodierungen. Dies zeigt sich auch bei den unterschiedlichen Größen der Keyframes.

Durchschnittliche Größe der Keyframes bei einem Keyframe-Intervall von 100 und ein Schwellenwert von 10 beim Codec der vorliegenden Arbeit.

Quantizer	1	2	4	6	8	12	18	24	31	Einheit
LMJpeg	1830	1526	1246	1117	1010	880	763	678	610	kbit
XviD	1487	1051	725	599	503	400	314	259	220	kbit

Die PSNR-Werte sind weitgehend gleichbleibend und deuten auf eine Überlegendheit des Codecs der vorliegenden Arbeit für dieses Video hin. Die Bewegungskompensation von XviD scheint hier für die schlechteren PSNR-Ergebnisse verantwortlich zu sein.
Um beim Vortragsvideo mit dem Codec der vorliegenden Arbeit eine mit XviD vergleichbare Komprimierung zu erhalten, sind bereits Schwellenwerte von 5 bis 10 gut ausreichend, siehe Abbildung 3.2. Dabei sind visuelle Fehler durch zu unterschiedliche Blöcke kaum sichtbar für Quantizer von 12 und kleiner, während für größere Quantizer einige Fehler sichtbar sind. Bei diesem Video hat aber die Schwellenwerte größeren Einfluß auf Dateigröße, siehe Abbildung 3.6. Insgesamt kann der Codec der

vorliegenden Arbeit bei kleinen Quanitzern mit XviD vergleichbare oder sogar überlegene Ergebnisse erzielen, aber bei höheren Quantizern treten hier schon bei kleinen Schwellenwerten erhebliche Fehler auf, so dass XviDs Ergebnisse überlegen erscheinen. Auch hierbei bestätigen die PSNR-Werte für dieses Video diese Einschätzung.
Für das Nachrichtenvideo muss ein Schwellenwert von ca. 20 bei kleinen Quantizern gewählt werden und von bis zu 5 bei größeren Quantizern um eine mit XviD vergleichbare Datenrate zu erzielen, siehe Abbildung 3.3. Der Einfluß der Schwellenwerteinstellung ist vergleichbar mit dem bei dem Vortragsvideo, siehe Abbildung 3.7. Zur Vermeidung von größeren visuellen Fehlern sollte der Quantizer bereits bei einem Schwellenwert von 15 kleiner als 8 sein. Daher ist eine Überlegenheit von XviD festzustellen. Die PSNR-Werte verdeutlichen diese Beobachtung, denn XviD erreicht Werte, die mit denen des Codec der vorliegenden Arbeit bei einem Schwellenwert von 1 vergleichbar sind.
Beim Werbevideo müsste ein Schwellenwert von 30 bis 50 gewählt werden um mit XviD vergleichbare Dateigrößen bei gleichen Quantizern zu erreichen, siehe Abbildung 3.4. Auch hier ist der Einfluß der Schwellenwertseinstellung groß, siehe Abbildung 3.8. Aber es ist auch zu festzustellen, dass sich selbst mit kleinsten Quantizern bei diesen hohen Schwellenwerten keine visuell akzeptablen Ergebnisse erzielen lassen. Daher ist XviD bei diesem Video deutlich überlegen. Auch hier liefert XviD PSNR-Ergebnisse, die mit dem Codec der vorliegenden Arbeit nur bei einem Schwellenwert von 1 erreicht werden können.

3.6 Schlußfolgerung

Die unterschiedlichen Entropie-Enkodierungen bei dem Codec der vorliegenden Arbeit und XviD erschweren zwar einen direkten Vergleich, aber dennoch ist festzustellen, dass beim Vorlesungvideo die Bewegungskompensation unnötig ist und dass sich sowohl beim Vortrags- als auch beim Nachrichtenvideo mit kleinen Quantizern, also hoher visueller Qualität, wettbewerbsfähige Ergebnisse erzielen lassen. Die Resultate beim Werbevideo aber verdeutlichen, dass hierbei keine vergleichbaren Ergebnisse erreicht werden können - die Vorteile der Bewegungskompensation sind zu groß.
Insgesamt entsprechen die Ergebnisse weitgehend den Erwartungen, allerdings ist festzuhalten, dass XviD bei höheren Quantizern und vergleichbarer Datenrate unerwartet bessere visuelle Ergenisse erzielt, da der Schwellenwert der Blockauswahlmethode hierbei zu starke negative Auswirkungen hat.

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung

Abschließend läßt sich sagen, dass der Codec der vorliegenden Arbeit sehr gut für spezielle Videos, wie dem Vorlesungsvideo, geeignet ist. Es werden sowohl gute Ergebnisse im Vergleich zu XviD erzielt, als auch spricht der geringere Aufwand für diesen Codec. Dieses Ergebnis läßt sich auf vergleichbare Videos übertragen, also Videos, die extrem wenige unterschiedliche, sich abrupt ändernde Bilder enthalten - hierbei ist Bewegungskompensation überflüssig.
Für Videos mit wenig Bewegung, wie dem Vortrags- und dem Nachrichtenvideo der Evaluation, ist der Codec der vorliegenden Arbeit ähnlich gut geeignet wie XviD, nur dass die Enkodierung mit diesem Codec theoretisch weniger aufwendig ist. Grob läßt sich sagen, dass hier bei XviD die Vorteile der Bewegungskompensation für die bewegungsreichen Bildabschnitte die Nachteile für die bewegungsarmen oder bewegungslosen Bildabschnitte aufwiegen und so ein vergleichbares Ergebnis erzielt wird. Ein guter Anwendungsfall für den Videocodec der vorliegenden Arbeit könnte also die zeitkritische Enkodierung von bewegunsarmen Videos sein, als Beispiele können Videotelefonie und Videokonferenz gennant werden.
Der Codec der vorliegenden Arbeit liefert für das Werbevideo die schlechtesten Resultate im Vergleich zu XviD ab. Bei Videos von diesem Typ, also Videos, in denen Bewegung in normalen Umfang vorkommt und für die keine weiteren Einschränkungen gegeben sind, können die Vorteile der Bewegungskompensation nicht aufgewogen werden.

4.2 Ausblick

Die Geschwinigkeitsnachteile des Codecs gegenüber XviD sind wahrscheinlich leicht auszuräumen, theoretisch müßte XviD auch zu übertreffen sein. Besonders bei der Bildkonvertierung und bei der Entropie-Enkodierung gibt es noch erhebliches Optimierungspotential.
Weiterhin könnten möglicherweise die visuellen Probleme bei hohen Quantizern durch eine verfeinerte, auf geringere Qualitäten angepasste Blockauswahlmethode ausgeräumt werden.
Schließlich dürften noch größere Verbesserungsmöglichkeiten in der Anpassung der Verfahren, wie Quantisierungs- und Blockauswahlverfahren oder Kompressionsmethode liegen. Auch wurde in dieser Arbeit nur eine Quantisierungsmatrix für alle Farbebenen benutzt, stattdessen könnte für die Chrominaz-Ebenen eine andere Quantisierungmatrix mit höheren Werten benutzt werden und daraus könnten sich Speicherplatzvorteile ergeben.

Die Eignung des Codecs für Übertragungen in verlustreichen Netzen und die visuellen Auswirkungen von Datenverlusten wurden in dieser Arbeit nicht untersucht. Auch hierfür wären sicher noch weitere Optimierungen möglich oder nötig. Generell läßt sich sagen, dass bei dem Codec der vorliegenden Arbeit der Verlust eines I-Frames in etwa dem Verlust des P-Frames entspricht, das nur die ausreichend stark geänderten Blöcke des I-Frames enthält. Die Auswirkungen des Verlustes eines P-Frames hingegen sind abhängig von der Anzahl der tatsächlich in dem P-Frame gespeicherten Blöcke, den Unterschieden dieser Blöcke zu denen des vorhergegeangen Bildes und der Höhe der Wahrscheinlichkeit, dass diese verlorengegangenen Blöcke in einem der nächsten Frames erstezt werden. Diese Wahrscheinlichkeit ist aber umso höher, je öfter der Block ersetzt wird und daher auch je höher die Wahrscheinlichkeit eines Verlustes des Blockes ist. Frameverluste würden als „stehengebliebende“Blöcke wahrgenommen. Alles in allem dürften die Auswirkungen aber erheblich geringer ausfallen, als bei Videocodecs, die Bewegungskompensation einsetzten, da hier bei einem Verlust eines Frames die Bezugsbasis für die folgenden P-Frames fehlen könnte und dies zu deutlicheren Fehlern führen würde.

Literaturverzeichnis

[1] Ze-Nian Li and Mark S. Drew. *Fundamentals of Multimedia*. October 2003.

[2] Matthew Marjanovic. Chroma Subsampling Standards. `http://www.mir.com/DMG/chroma.html`, 2003.

[3] MPEG Software Simulation Group. Transformation And Quantization. `http://www.mpeg.org/MPEG/MSSG/tm5/Ch7/Ch7.html`.

[4] Wikipedia. Huffman coding. `http://en.wikipedia.org/wiki/Huffman_coding`, 2006.

[5] Mark Adler. Jean-loup Gailly. zlib Compression Library. `http://www.zlib.net`.

[6] John F. McGowan. AVI Overview: Programming and Other Technical Topics. `http://www.jmcgowan.com/avitech.html`, 2000.

[7] FFMPEG Developmers. FFMPEG Multimedia System. `http://www.ffmpeg.org`.

[8] XviD Developmers. XviD Video Codec. `http://www.xvid.org`.

[9] Dr. Dmitriy Vatolin, Alexey Moskvin, Oleg Petrov, Nicolay Trunichkin. MSU Video Quality Measurement Tool. `http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html`.