Just In Time Joint Encoding of Multiple Video Streams

Master's thesis

Henrik Juul Pedersen
&
Palle Ravn

June 6, 2013

AALBORG UNIVERSITY
STUDENT REPORT

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Øst
Phone: +45 96 35 86 00
Internet: es.aau.dk

Title:
Just In Time Joint Encoding of Multiple Video Streams

Theme:
Master's thesis

Project period:
September 2012 - june 2013

Project group:
13gr1071

Members of the group:
Henrik Juul Pedersen
Palle Ravn

Supervisors:
Jan Østergaard
Søren Holdt Jensen

Number of copies: 5

Number of pages: 69

Attachments: CD

Appendices: 1

Project completed: June 6, 2013

Synopsis:

This master's thesis focuses on H.264 video compression of multiple streams to be transmitted over a limited channel. The report describes the workings of the H.264 codec. Afterwards, proposals on bitrate estimators are presented, and a predictor is chosen for later use in rate control. We design a rate controller set up as a constrained convex optimization problem, and test it against a set of video sequences.
Our results show that it is possible to encode video sequences jointly with regard to their individual qualities, whilst still keeping fluctuations in quality low. We conclude, that if a Just-In-Time encoder is created with regard to our proposed rate controller and bitrate prediction, it could be used in realtime joint video coding.

AALBORG UNIVERSITET
STUDENTERRAPPORT

Institut for
Elektroniske Systemer
Fredrik Bajers Vej 7
9220 Aalborg Øst
Telefon: 96 35 86 00
Internet: es.aau.dk

Titel:
Just In Time Joint Encoding of Multiple Video Streams

Tema:
Kandidatspeciale

Projektperiode:
September 2012 - juni 2013

Projektgruppe:
13gr1071

Medlemmer af gruppen:
Henrik Juul Pedersen
Palle Ravn

Vejleder:
Jan Østergaard
Søren Holdt Jensen

Antal kopier: 5

Antal sider: 69

Bilag: CD

Appendikser: 1

Projekt afsluttet: 6. juni, 2013

Synopsis:
Dette kandidatspeciale fokuserer på H.264 video komprimering af flere video strømme til transmission over en begrænset kanal. Rapporten beskriver hvordan H.264 fungerer. Derefter foreslås nogle bitrate estimatorer, og en estimator er valgt til senere brug i ratekontrol. Der designes en ratekontrollør sat op som et begrænset konvekst optimeringsproblem, og det testes imod et antal videosekvenser.
Resultaterne viser at det er muligt at indkode video sekvenser ud fra en fælles betragtning af deres individuelle kvaliteter, imens udsving i kvalitet holdes lavt.
Det konkluderes, at hvis en 'Just-In-time' indkoder laves med øje for den foreslåede ratekontrollør og bitrate estimator, kan den benyttes til realtids fælles video indkodning.

Preface

This report has been conducted as the main part of our master's thesis doing the fall semester of 2012 and spring semester of 2013. We have tried to explain most of the technical expressions and context to an extend where anyone with a basic knowledge of video coding should be able to read it. Basic knowledge of convex optimization is needed for understanding the propossed rate-controller as it is derived from a optimization problem. The results can be understod without any knowledge of computer code, but it is required for in depth understanding of the simulation setup.

While most notations should be eighter self-explanatory or be explained at apperence, we present the most commonly used notations just in case someone would find them usefull. When writing mathmatics we have that vectors are bold lower case letters, e.g. $\boldsymbol{r}$. Matrices are bold upper case letters, e.g. $\boldsymbol{M}$ and if its dimensions are given explicitly we eighter use $\boldsymbol{M}^{4\times4}$ for a 4 by 4 matrix, or $\boldsymbol{M} \in \mathrm{R}_{+}^{4\times4}$ for a positive semidefinite 4 by 4 matrix.

When refering to video sequences the name of the clip is often emphasised as *ducks take off*, this is however not restricted to video names.

Most data processing have been done with python, and all plots are generated using the matplotlib package.

We would like to thank Futarque for an intreresting visit to their Aalborg office, and for offering help with test equipment.

Palle Ravn

Henrik Juul Pedersen

Contents

1 Introduction

This project focuses on joint compression of multiple video streams for broadcasting. Efficient compression of video makes room for additional streams, or streams of higher quality, on the same channel. Even though the individual streams might be optimally encoded given individual limitations they are not necessarily jointly optimal.

We will be looking at MPEG-4 Digital Video Broadcasting (DVB), without distinguishing between the physical channels, e.g. terrestrial, cable, satellite, or others. The MPEG-4 codec considered is the H.264 codec for HDTV up to 30 Hz[1].

The goal of the project is to enable Just-In-Time (JIT) encoding of multiple streams, by taking their different features into account in order to make the overall rate-distortion better. Our application should work on an arbitrary number of streams and any reasonable bandwidth limit, within physical and practical computational limits. The number of live versus prerecorded streams is not important. However, if all streams are prerecorded, one can benefit from offline encoding end thereby avoid real-time issues.

Encoding video in an optimal way is always a tradeoff between computational complexity, quality, and compression ratio. Instead of optimizing each video stream for a fixed rate, we would like to encode multiple streams jointly, such that the overall perceived distortion for the entire collection of streams is minimal. Our goal is to make a video-encoding framework with autonomous, yet intercommunicating, encoders with adjustable encoding parameters. These parameters are based on a real-time analysis of each video stream, extracting essential features in order to collectively decide on the individual encoding parameters, and thereby optimizing for the overall quality instead of the individual quality for each stream.

First we give a brief introduction to image compression based on the principles in H.264, followed by an introduction of relevant encoding parameters influencing encoding time, resulting bitrate, and video quality. Based on a selected subset of encoder parameters we will explore and compare ways of extracting the necessary features for real-time encoding which optimizes the overall quality.

All test sequences used for this project are described in appendix A, and can be found in lossy format together with some of the results from the project on the enclosed CD.

1.1 Terminology

ITU-T H.264, AVC, MPEG-4 part 10, and ISO/IEC 14496-10 are all synonymous[1] and will simply be called H.264 where applicable throughout this report.

It is important to distinguish between individual video streams, collections of streams, and multiplexed streams. A single video stream is referred to as an Elementary Stream (ES), a collection of streams is referred to as a Compound Stream (CS), and a multiplexed CS will be referred to as a Transport Stream (TS). This is illustrated in figure 1.1.

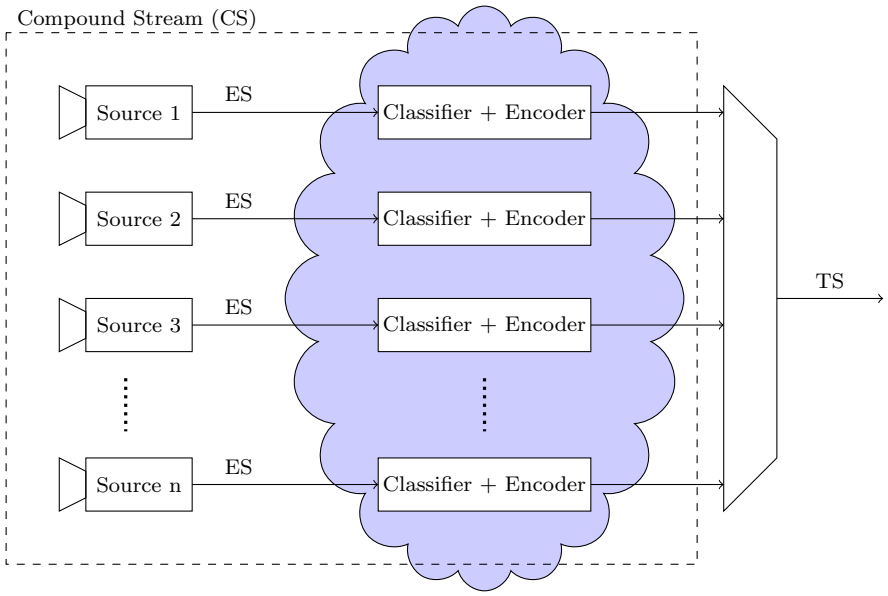


Figure 1.1: System overview and illustration of used abbreviations. Boxes included in the cloud are interconnected, allowing the classifier to share its complexity estimation with the other encoders. Seen from the left, we have n sources, individually referred to as an Elementary Stream (ES). Each ES is analyzed and encoded before being muxed, the muxed stream is referred to as a Transport Stream (TS). Several sources, regardless of encoding format, is referred to as a Compound Stream (CS), as illustrated by the dashed line.

Throughout this project the abbreviation MB is used for Macroblock. Megabyte and other units which are powers of two will be expressed using IEC binary prefixes, e.g. MiB for $8 \cdot 2^{20}$ bit[2].

1.2 State of the art

In order to reduce statistical redundancy in digital video, techniques to predict information from within a frame (intra coding) or from previously transmitted frames (inter coding) has been introduced and implemented in [1, 3, 4, 5] as part of the encoding process. In MPEG video, three different frametypes are currently used for generic video coding; I-, P-, and B-frames[1, 5]: I-frames are completely intra coded, and provide semi-random access to the stream, as it has no dependency on previously transmitted frames, P-frames can be inter predicted towards a single reference, and B-frames can be inter predicted towards two references[1, 5]. All predictions in MPEG are done on a Macroblock (MB) level; in H.264, a MB is 16×16 pixels, and can be split into smaller shapes within the MB[1]. In H.265 - the high-definition successor to H.264 - MBs used in earlier MPEG standards are replaced by Coding Units (CUs) and can be up to 64×64 pixels in size[5].

Coding-parameters relevant for a frame or slice are specified in its header, it includes skipped MBs, reference picture index(es), block sizes, motion vectors, etc. Recent developments in rate control proposes the size of the header information to be separated from the texture, as the header is often constant and does not change with quantization parameters, leaving separate header and texture rates to be approximated[6]. If rate control is done immediately before quantization, the exact header bits are known, and only the amount of texture bits should be approximated[7, 8]. Rate control can be split in two categories; online rate control for live broadcasts where latency and complexity are key elements in the dual problem of Rate-Distortion (R-D) versus complexity[9], and into offline rate control where multi pass optimal rate approximations with high complexity is possible[9]. The online complexity approximations often take their offset in statistics of prior frames[9], resulting in a dilemma known as the QP, or chicken-egg, dilemma[6, 10]. The chicken-egg dilemma describes the problem of estimating the rate from different parameters, or estimating optimal parameters from a missing rate estimate.

Several rate control methods has been proposed for block based video compression. [11, 12] proposes using a Discrete Cosine Transform (DCT) based complexity estimation together with buffer feedback to perform rate control for very low bitrate and low latency video coding. Their work inspired [13] to use a predicted Sum of Absolute Differences (SAD) (calculated as Mean Absolute Difference (MAD)) based approach in order to reduce PSNR fluctuations in higher rate video. DCT coded residuals are also used in [14]

where they use the macroblock histogram differences for rate control. The amount of zeros after quantization is proved[7, 8] to have a linear relationship with the resulting texture bits, allowing for accurate bitrate approximations. As [7, 8, 11, 12, 13, 14] work on residuals, either inter or intra predictions has been performed prior to analysis and thus, their complexity estimations work with all frame types.

2 The H.264 Codec

H.264 exploits both spatial and temporal correlation in the source during encoding. This enables high compression rates, at the cost of additional encoder complexity. In short, H.264 searches within, and between frames, looking for a similar area, encoding only a Motion Vector (MV) for that area, and a residual. This will be explained to greater depth in section 2.2.

The H.264 standard defines numerous ways of encoding video. The different encoding parameters are grouped into profiles. Further, each profile in the H.264 standard has several levels, which describe the maximum rate of data the decoder must be able to process. The simplest profile from the original standard is called *baseline*, followed by profiles *main*, and *extended*. The general idea is, that with each profile the complexity of the encoder increases, allowing for a better compression rate and/or quality. Later on, ammendments have added further profiles, such as the *high* profile[1].

First we will describe the basics of the H.264 encoder, to give an introduction to image compression, this also introduces the terms used when discussing encoding and decoding. Next we will state the DVB recommendations that we find necessary for our complexity analysis, and argue why we rely on and restrict attention to these.

2.1 Macroblocks

A video sequence consists of frames, where each frame corresponds to a still image. Frames can be represented in the YCbCr format, with or without chroma subsampling[15]. Chroma subsampling enables better compression with low perceptual loss in quality. The H.264 encoder divides each frame into regions of 16×16 pixels called Macroblocks (MBs). A slice is a number of MBs with similar properties. Both slices and MBs are depicted in figure 2.1. There are three types of slices, I, B, and P slices. An I-slice uses only intra prediction and thus is a self sufficient representation of the image. A P-slice constructs the current frame based on previous frames. A B-slice can go both back- and forward in the frame order when searching for a Motion Estimator (ME). Typically the bit count for the different slice types follows; $\mathrm{I} > \mathrm{P} > \mathrm{B}$.

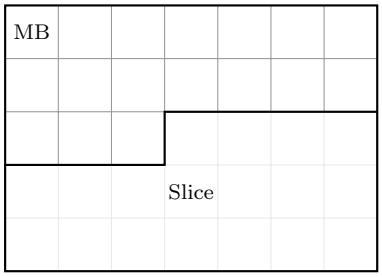
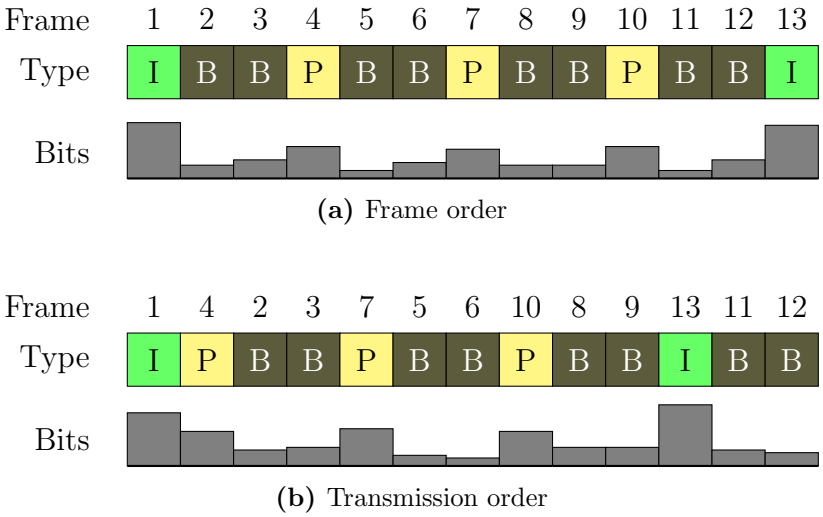


Figure 2.1: A frame partitioned in 5×7 MBs, and two slices in raster scan order.



(a) Frame order

(b) Transmission order

Figure 2.2: Illustration of transmitted slices. In this illustration; B slices depend on their neighbouring I and P slices. (a) shows the order of which the frames should be displayed. (b) illustrates the necessary transmission order, for the decoding of B slices to be possible. The lower part of (a) and (b) shows a number of bits to transmit for each slice, where I > P > B.

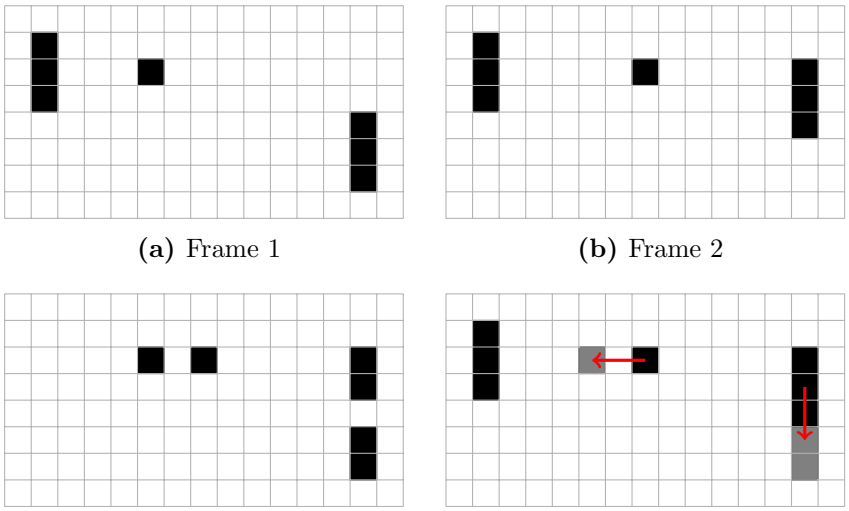
2.2 Prediction

The main goal of prediction is to find a MB that is near identical to the current one. As much video data is correlated both spacially and temporally it is possible to encode some of the video as MVs. MVs point to a place in a previously decoded reference frame. This reference is used as a prediction, which combined with a residual, is used to resemble the current block of the image. With a good prediction, the vector and residual representation uses far less bits than a full intra coding of the image block. Many search schemes for finding MVs have been proposed, such as the diamond search[16], as an exhaustive search is computationally very complex.

Encoding a video sequence with correlated data by MVs reduces the bit rate, as much less data needs to be transmitted. Using MVs is favorable in terms of storage and transmission, all at the expense of increased coding complexity and perhaps delay. In the case where the reference MB is not exactly identical to the current MB the residual is encoded along with the associated MV. Due to the high correlation between the two MBs, it is still beneficial to encode the residual since it, like the MV, describes the difference between something known and something new, instead of describing everything. An illustration of the advantage of prediction is depicted in figure 2.3 with a game of pong. Two succeeding frames (a) and (b) are showing the complete frames, (c) shows the absolute difference between (a) and (b). Encoding the differences, the black MBs in (c), instead of the whole frame (b) reduce the data to encode. Better is the Motion Vector prediction, as shown in (d), only the vectors pointing to the reference MB needs to be encoded. This is a very simplified example and often the residual between the two MBs would be encoded as well.

2.3 Compression

As video images often have high spatial correlation, a better compression of MBs can be accomplished in practice by decorrelating the information. For this purpose H.264 uses the Discrete Cosine Transform (DCT). The DCT is invertible and produce a frequency representation. As many images are slowly changing over space, high frequency content from the DCT can be truncated to zero, resulting in a lossy compression, still with good perceptual quality. The DCT is applied to both inter and intra coded MBs.



(a) Frame 1 **(b)** Frame 2

(c) Absolute difference **(d)** Prediction

Figure 2.3: Illustration of two frames and the motion between them. The illustration is a classical game of pong, the player to the left has no movement. The ball and the player to the right move between the frames. (c) shows the absolute difference between the frames. (d) shows Frame 2 overlayed on Frame 1, with motion vectors.

2.4 DVB recommendations

As broadcasting companies have many degrees of freedom when encoding a video-stream, we will rely on the recommendations given by DVB[17]. On the I slice frequency they recommend the following:

> "It is recommended that a video sequence header, immediately followed by an I-frame, be encoded at least once every 500 ms."[17]

By I frame it is understood that all MBs are intracoded over a 500 ms period at most. By doing this, random access to the stream is possible within 500 ms plus eventual delays caused by buffers and decoding.

This project will focus on HD transmission up to 30 Hz, the profiles and levels specified for this is the H.264 High Profile at Level 4[17].

2.5 Limitations

The H.264 High Profile at Level 4 introduces some features and limitations:

- Only I, P, and B slice types may be present.
- Chroma formats allowed: 4:0:0 and 4:2:0.
- Luma and Chroma samples must have a bit depth of 8.
- Maximum MBs per second: 245 760.
- Max frame size: 8192 MBs.
- Max Decoded Picture Buffer (DPB) size: 32 768 MBs.
- Vertical MV component range: -512 to 511.75.
- Horizontal MV component range: -2048 to 2047.75.
- Bi-predictive blocks less than 8×8 are not supported.
- From the maximum MBs per second it is seen, that the maximum frame rate for 1920×1080 pixels video equates to 30.1 frames per second.

All limitations are defined in [1, Appendix A].

2.6 Rate control

The Joint Model (JM) reference encoder allows for automatic rate control, the target can be quality or bitrate.

As seen on figure 2.4, the encoding process is split into several steps. As the input is analyzed through the different processes, a rate controller can act on acquired information to adapt encoding parameters. E.g. a frame or a MB can be skipped if the buffers are too full, or a frame can be intra coded if a scene change is registered.

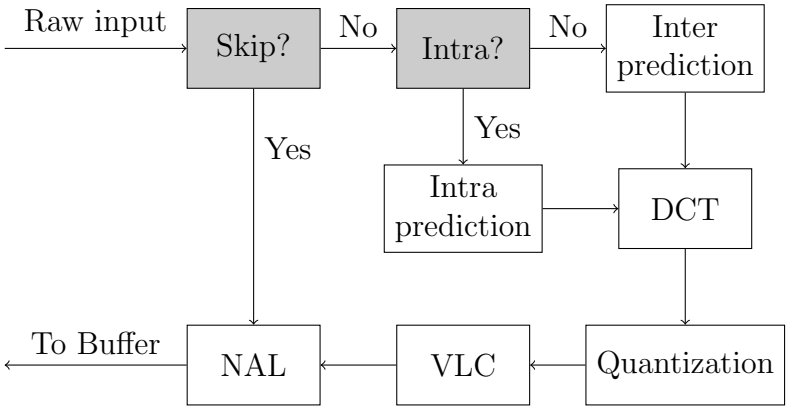


Figure 2.4: Simplified model of the H.264 encoding process. **Skip** decides whether an entire frame should be skipped. **Intra** decides whether the frame should be intra coded, e.g. at a scene change. **Intra prediction** cycles between predictive modes to reduce spacial redundancy. **Inter prediction** searches for the best MV in one or more reference frames, some MBs might be intra coded or skipped. **DCT** transforms the residuals from the predictors to focus the energy at dominant frequencies. **Quantization** reduces the number of transform coefficients based on QPs. **VLC** is a lossless variable length coder, reducing statistical redundancy. **NAL** encapsulates header and texture information for later decoding.

3 Complexity Estimation

This chapter explains the complexity estimation techniques that has been considered for this project. Complexity estimation is to be used in the communication towards a rate-controller, for it to decide on a set of encoding parameters. By complexity we mean the R-D relation. Assume several individual video streams encoded at the same bitrate, then the stream with the lowest distortion is also the one with the lowest complexity. Likewise, if they where encoded to have the same distortion, the stream with lowest complexity would be the one with the lowest bitrate.

3.1 Complexity

The complexity, or bit requirement, of a H.264 encoded frame is determined from a set of parameters: If the frame is to be intra coded, each MB is predicted from surrounding pixels, or used as raw data. If it is predicted from surrounding pixels, a smooth image will give a better prediction, and thus reduce the residual to be coded. If a frame is inter coded, the complexity is determined from whether good motion vectors to a reference frame is possible, the temporal changes between reference and subject frames decide the length of the motion vectors, and the residual to be coded.

The residuals are coded the same way for inter and intra coded frames, but for inter coded frames, a motion vector is also to be coded. Motion vectors in H.264 are coded using signed exponential-golomb codes[1, section 7.4.5.1], allowing arbitrary length vectors, at the cost of word length growing with vector length, see table 3.1. H.264 also allow for quarter-pixel motion vectors, extending the resulting word lengths further.

symbol	bits
0	1
1	010
-1	011
2	00100
-2	00101
3	00110
-3	00111
4	0001000
-4	0001001
...	...

Table 3.1: Example of the signed exponential-golomb codes used in H.264, the amount of leading zeros determine the amount of bits to be read after the first one.[1, section 9.1.1]

3.2 Feature extraction

The QP is correlated with the bitrate, as depicted in figure 3.1 for all our test movies. However, there is not a linear or exact mapping from QP into bitrate, as the bitrate depends on the texture details and motion in the movie.

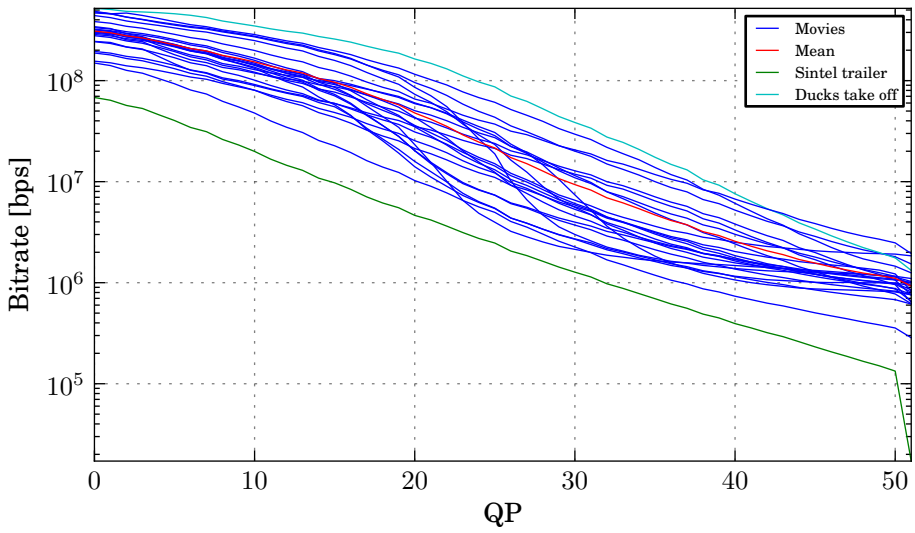


Figure 3.1: Relation between the encoder parameter QP and the resulting bitrate for 52 values of QP and 24 movies, plotted on a semilogarithmic scale.

It has been shown by [7] that there is a linear relation between the encoded bitrate and the percentage of zeros in a quantized transform, for any typical transform used in image coding, including the DCT. The relation in [7] is defined as

$$\hat{R}(\rho) = \theta \cdot (1 - \rho) \tag{3.1}$$

where $\hat{R}(\rho)$ is the bitrate estimation, ρ is the percentage of quantized zeros and θ is a constant. As ρ depends on the QP we can predict the bitrate for a set of QPs and select the one with the highest bitrate within the channel limits, provided that θ can be estimated or is known in advance. This only covers the bitrate for the DCT quantization, the video stream also includes information such as MVs and QP values, therefore we split the prediction of the total bitrate into texture and header information. As seen in figure 2.4 all the header information is calculated prior to the DCT and does not have to change with the QP, therefore the resulting bitrate for the header information is known at the time of texture quantization, and only the texture bitrate needs to be predicted.

Prediction of the texture bitrate using equation 3.1, depends on ρ and θ. For determination of ρ we need to know the quantizer and the DCT values. The 4-by-4 quantizer implemented in the JM H.264 reference encoder is described in the following, where all constants are deduced by [18]. A matrix $\boldsymbol{M}$ is defined to be

$$\boldsymbol{M} = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix}, \tag{3.2}$$

and it serves as a lookup table for construction of a scale matrix. If $q = (\text{QP} \bmod 6)$ then the scale matrix $\boldsymbol{S}$ is given by

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{M}_{q,0} & \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,0} & \boldsymbol{M}_{q,2} \\ \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,1} & \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,1} \\ \boldsymbol{M}_{q,0} & \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,0} & \boldsymbol{M}_{q,2} \\ \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,1} & \boldsymbol{M}_{q,2} & \boldsymbol{M}_{q,1} \end{bmatrix}. \tag{3.3}$$

Note that $\boldsymbol{S}$ is constructed from a single row of $\boldsymbol{M}$ and only contains three unique numbers. Let $\boldsymbol{D}^{4x4}$ be a DCT matrix, then the quantized matrix $\boldsymbol{L}^{4x4}$ is given by

$$\boldsymbol{L}_{m,n} = \text{sign}(\boldsymbol{D}_{m,n}) \cdot \left[(|\boldsymbol{D}_{m,n}| \cdot \boldsymbol{S}_{m,n} + \text{q\_offset}) \gg \text{q\_bits} \right] \tag{3.4}$$

where $|\cdot|$ is the absolute value, $\text{q\_bits} = 15 + \text{floor}(\frac{\text{QP}}{6})$, and q_offset is a constant that depends on the frame type, the QP, and the prediction mode. Combined with the bitshifting, the addition of q_offset equals a rounding function, where q_offset determines the deadzone. With equation 3.4 we are able to determine ρ in equation 3.1 for different QPs and make predictions of the resulting texture bitrates.

Based on the above we give an example of a frame based prediction using equation 3.1 and 3.4, where θ will be updated after each encoded frame as

$$\theta_{i+1} = \frac{r_i}{1 - \rho_i}, \tag{3.5}$$

where r is the texture bitrate, and ρ is the percentage of quantized zeros over the whole frame. As depicted by figure 3.2 the prediction is close to the actual bitrate, with an average error of 0.31%.

For comparison we calculate the Pearson's correlation coefficient, stated in equation 3.6, between the bitrate and the prediction measurement, denoted $C(\boldsymbol{x}, \boldsymbol{r})$, where $\boldsymbol{x}$ is the considered measure, and $\boldsymbol{r}$ the bitrate vector. For this example the correlation coefficient is $C(\boldsymbol{\rho}, \boldsymbol{r}) = -0.9988$ which is close to -1 implying a near linear relation, as demonstrated in [7].

$$C(\boldsymbol{x}, \boldsymbol{r}) = \frac{\text{cov}(\boldsymbol{x}, \boldsymbol{r})}{\sigma_x \sigma_r}, \tag{3.6}$$

where cov is the sample covariance function, and σ is the sample standard deviation.

There are other complexity measures that can be used for bitrate prediction. In the following we motivate for some alternative candidates and their performance as a linear predictor. We seek a linear predictor as it is

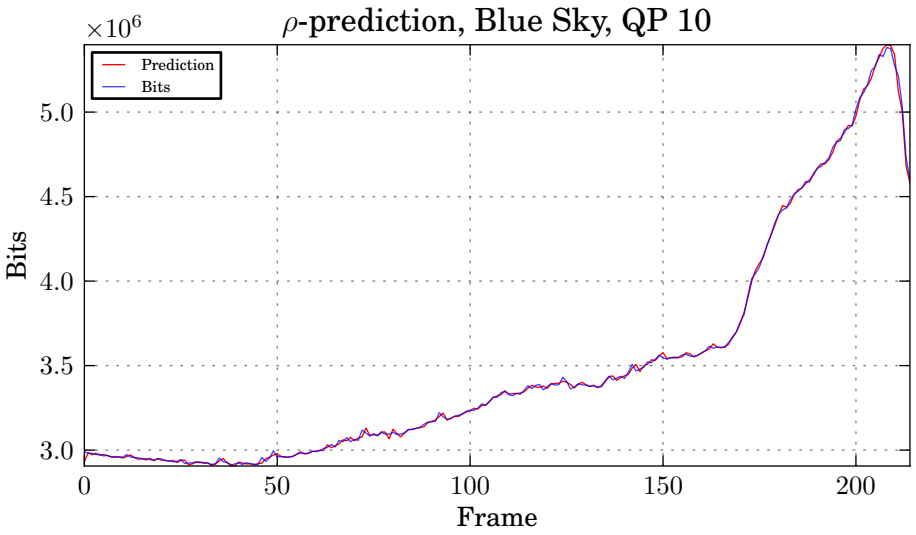


Figure 3.2: Linear prediction of the texture bitrate for the movie Blue Sky at QP 10, using equation 3.1. The first frame is left out, as θ was initially set to 1.

computationally simple and fast to execute. The different measures are compared by Pearson's correlation coefficient, as it measures the linearity on a closed scale where ± 1 is an expression for linear correlation and 0 is no correlation. See table 3.2 for an easy comparison of correlation coefficients. For visual comparison of the ρ-predictor and the following see figure 3.3, as it is constructed in the same way as the inter coding examples.

For the tests in this chapter, we have generated motion vectors and intra predictions from the raw sequences. The motion vectors are created from a full search with a search range of ± 32 both horizontally and vertically, or as far as possible when searching near edges. The code we created for the full search algorithm is located on the CD, we compiled it as a shared object and called it from a python script.

As inter and intra frames are coded differently, we have split the complexity estimators into two categories. The features we will look into are, for:

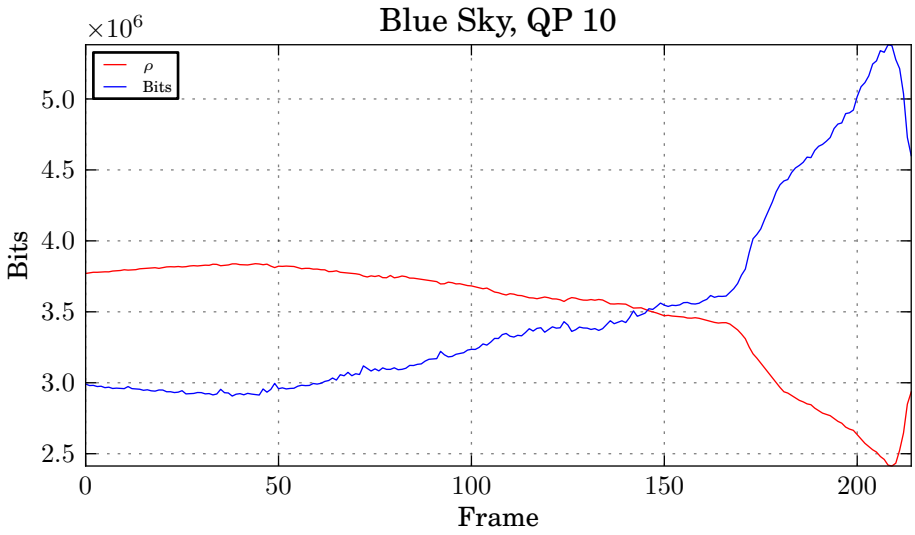


Figure 3.3: Comparison of bitrate and a scaled ρ for visual inspection. The first frame is left out.

Inter coding

- MVs, horizontal and vertical.
- SAD between a MB and its optimum reference after motion estimation.
- DCT variance

Intra coding

- Predictive modes:
  - Horizontal.
  - Vertical.
  - DC.

For easier visual inspection, inter predictors are illustrated by scaling the measurement, such that it has the same mean as the bitrate. The movie *blue sky* has been chosen for illustration as it does not contain any scene changes, which need to be treated as a special case due to the sudden change in bitrate. The movie starts out with a picture of a clear blue sky, and slowly rotates towards a tree top, where the tree top has more details than the plane colored sky. This is reflected by the increasing bitrate. The measurements are compared using Pearson's correlation coefficient, where

the constant is calculated for both *blue Sky* alone and for all movie clips stacked into one long movie.

MV

This measurement is very simple. Given a maximum range for the MB motion search, the MV to the reference resulting in the lowest SAD is used as a measure for complexity. The complexity measurement is given by

$$CMV = \sum_{h=1}^{H} ||MV_h||_1 \tag{3.7}$$

where H is the number of MBs, $||\cdot||_1$ denotes the ℓ_1-norm, and MV_h is the h^{th} Motion Vector (MV) resulting in the best reference within the search limits.

Motivation

This measurement gives a value for the total motion between two frames in the video. As MVs are limited by a maximum search range, and possibly a number of iterations, the result will be an inferior reference which results in a worse rate distortion. Long MVs uses more bits, due to the golomb coding, and combined with the possible inferior MB references, a larger CMV should imply a higher bitrate. MVs are a natural part of the encoding process, therefore the information required for this measurement is readily available.

Results

From figure 3.4 we see that CMV is inverse proportional to the bitrate, also indicated by the correlation coefficient of -0.9520 for this particular clip. The correlation coefficient drops to -0.2465 when using all the test movies.

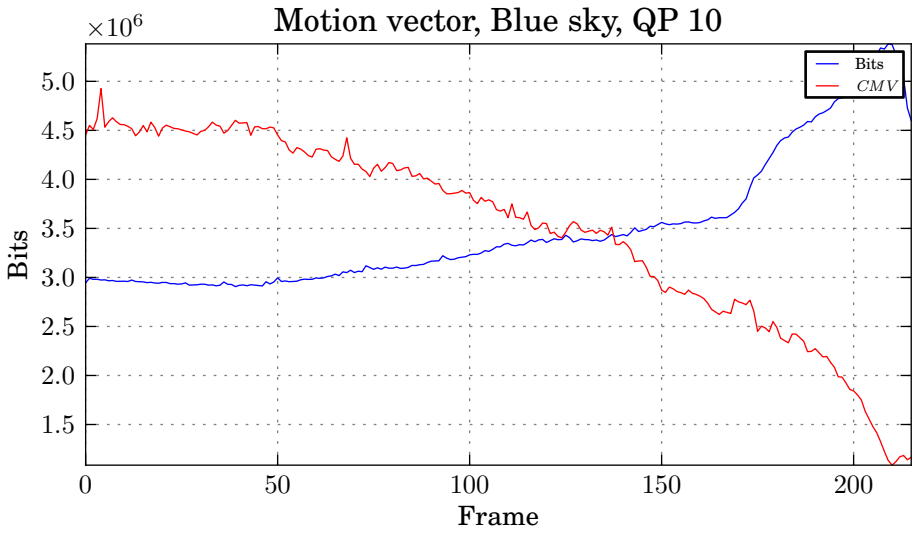


Figure 3.4: Comparison of CMV from equation 3.7 and the textual bitrate at QP 10, for the movie *blue sky*. Note that CMV is scaled to match the mean of the bitrate.

SAD

An SAD between a MB and its optimum reference within the search range for all MBs in a frame is considered.

$$CB = \sum_{g=1}^{G} \sum_{n=1}^{N} \sum_{m=1}^{M} |MB(n,m)_g - \widehat{MB}(n,m)_g| \tag{3.8}$$

where G is the number of MBs, N is the MB width in pixels, M is the MB height in pixels, and the hat denotes the optimal reference MB for the current MB.

Motivation

As it is the residuals of one MB subtracted from the best matching reference that are DCT transformed and quantized, the residual coefficients are related to the resulting bitrate. If we are able to determine the relation between the residuals and bitrate, we are able to select the value for QP giving the optimal rate distortion under the bandwidth restriction. This measure is attractive as some form of SAD is already performed to compare motion estimators, and will thus require few additional computations in the encoding process.

Results

The *CB* measure in figure 3.5 looks to be linear correlated with the bitrate, as the adjustment of the mean is a linear scaling. For the *blue sky* movie in figure 3.5 the correlation coefficient is 0.9966. For all the test movies the correlation drops to 0.5109.

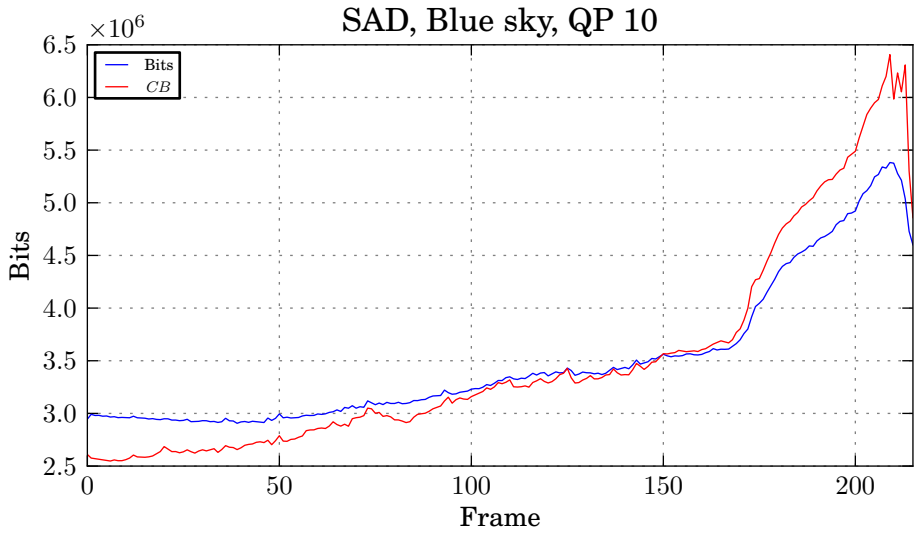


Figure 3.5: Comparison of *CB* from equation 3.8 and the textual bitrate at QP 10, for the movie *blue sky*. Note that *CB* is scaled to match the mean of the bitrate.

DCT variance

The DCT values are approximately laplacian distributed. The distribution is considered zero-mean and therefore the only unknown parameter is the variance[6, 14]. The measure is given by

$$CV = \text{var}(DCT) \tag{3.9}$$

where DCT is the DCT coefficients for the frame.

Motivation

As seen on figure 3.6 the DCT coefficients resembles a laplacian distribution and over time a laplacian distribution with varying variance as depicted in figure 3.7. If the DCT values of a frame resembles a draw from a laplacian distribution, and the only unknown of the Probability Mass Function (PMF) is the variance, then the variance alone should give some insight about the complexity and thereby also the bitrate. As the DCTs values are made by an orthogonal transformation the result would be the same if applied on the residual values.

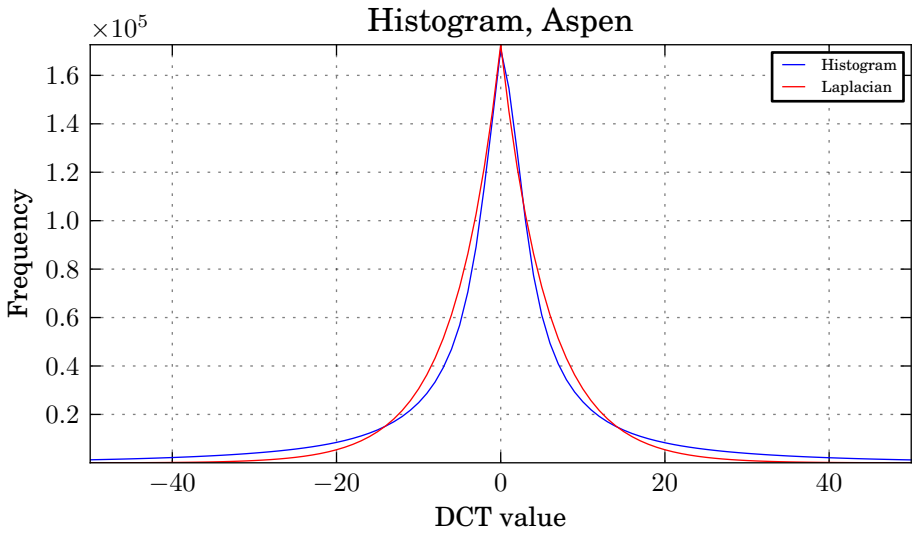


Figure 3.6: Histogram of DCT values for the *aspen* test sequence with a search range of 32. Any value outside the minimum and maximum bin is discarded.

Results

As depicted in figure 3.8 there are some correlation between the texture bitrate and the linear scaled measure CV. The correlation coefficient is 0.9820 for the *blue sky* movie.

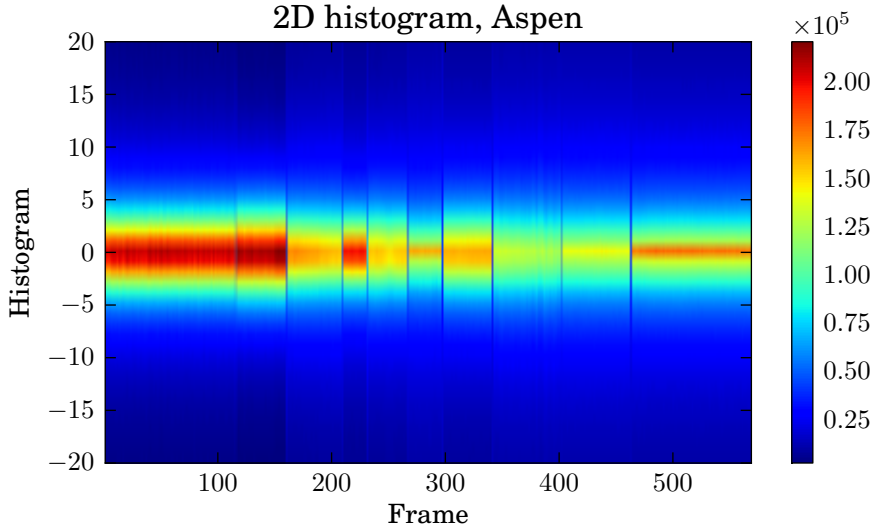


Figure 3.7: Histogram of DCT values per frame for the *aspen* test sequence. Screen changes happen where the variance spikes, illustrated by the blue lines cutting through the red center. The histogram changes little between frames within the same scene. However, there are distinct differences between scene histograms.

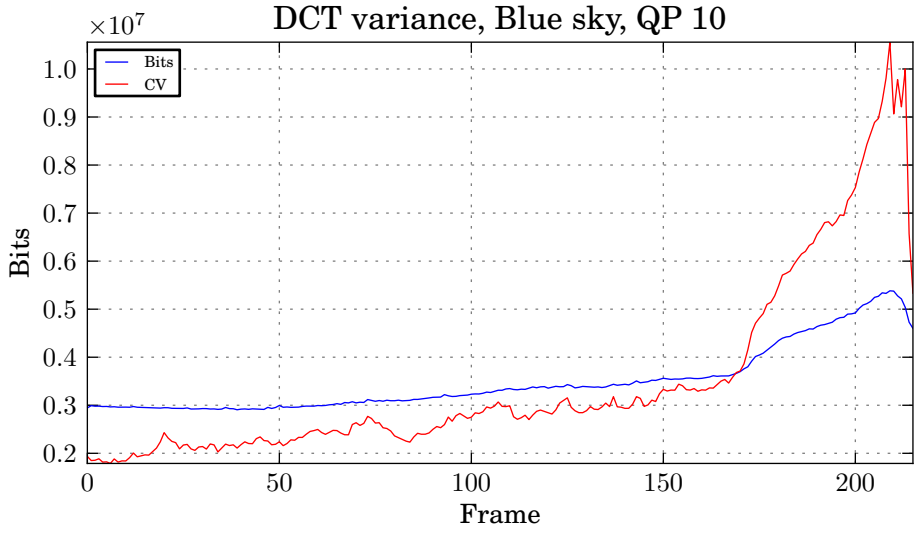


Figure 3.8: Comparison of CV and the texture bitrate at QP 10, for the movie *blue sky*. Note that CV is scaled to match the mean of the bitrate.

	Pearson's constant	
Inter	Blue sky	All movies
ρ-predictor	-0.9988	-0.9783
Motion vector	-0.9520	-0.2465
SAD	0.9966	0.5109
DCT variance	0.9820	0.2318
Intra		
Vertical	0.9776	0.7717
Horizontal	0.9876	0.7363
DC	0.9819	0.7462

Table 3.2: Pearson's correlation coefficient of the measurements and bitrates stacked into one signal. All bitrates are the result of an encoding with a QP of 10.

Intra predictive modes

Although the DCT transformed residuals prove to be a good complexity measure, the residuals after prediction might provide a cheap insight into the complexity of the current frame. The typical measure used by the encoder for picking a suitable predictor, the sum of absolute residuals, is readily available for use in complexity estimation. H.264 specifies several intra prediction modes. For 16×16 blocks four modes are specified: Vertical, Horizontal, DC, and Plane. As these modes are used for intra prediction, we believe that the resulting residual can be used for complexity estimation, as it is directly related to the encoding process. Only the Vertical, Horizontal, and DC modes are considered in this section. The Vertical and Horizontal modes only apply when neighboring MBs are present either above, or to the left, respectively. The DC predictive mode can always be applied. The prediction is used as a reference, as with inter coding. We consider 16×16 pixel MBs, but other predictors exist for sub MBs of different shapes and sizes which are not covered here.

Vertical predictive mode

The Vertical prediction works by extending the row of pixels above the MB vertically down through the MB, as it is also seen in figure 3.9.[1, section 8.3.3.1]

Horizontal predictive mode

The Horizontal prediction works by extending the row of pixels left of the MB horizontally through the MB, this functions similarly to the vertical

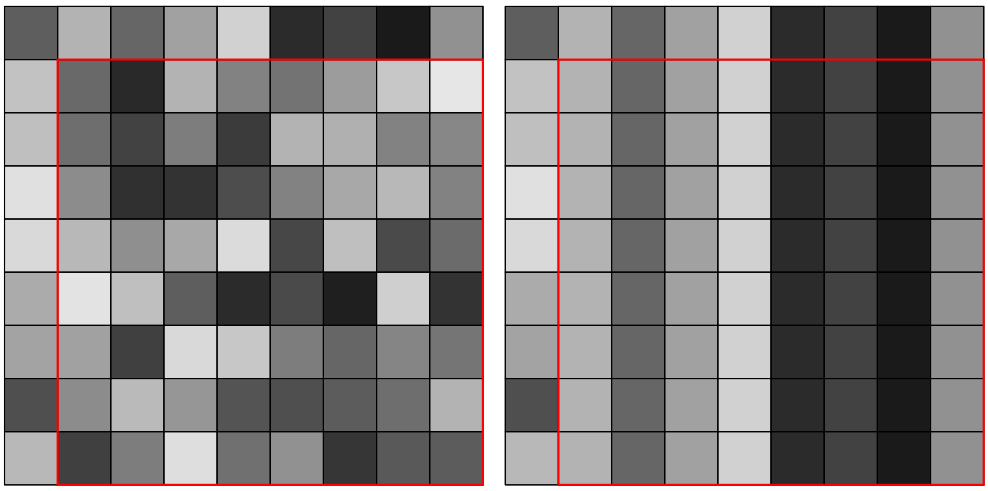


Figure 3.9: Illustration of the vertical extension used in the vertical predictive mode. The illustration to the left shows a sample MB of 8×8 pixels. The red square marks the current subject MB and outside the square the neighboring pixels are shown. The illustration to the right shows the vertical extension, resulting in a vertical prediction in the subject MB.

prediction illustrated in figure 3.9.[1, section 8.3.3.2]

DC predictive mode

The DC prediction works by making an average of available surrounding pixels to the left and above. If no pixels are available, the center of the dynamic range is chosen as DC value. This value is then used as the predictor for the entire block.[1, section 8.3.3.3]

Prediction performance

The figures 3.10, 3.11, and 3.12 show the vertical, horizontal, and DC predictive modes scaled to fit the bitrate, the movies are coded entirely from I-frames for this test.

The correlation coefficients of the three predictors versus all 24 movies are 0.7717 for vertical, 0.7363 for horizontal, and 0.7462 for DC. The correlation is, as expected, not as high as with DCT zeros. All coefficients are compared in table 3.2.

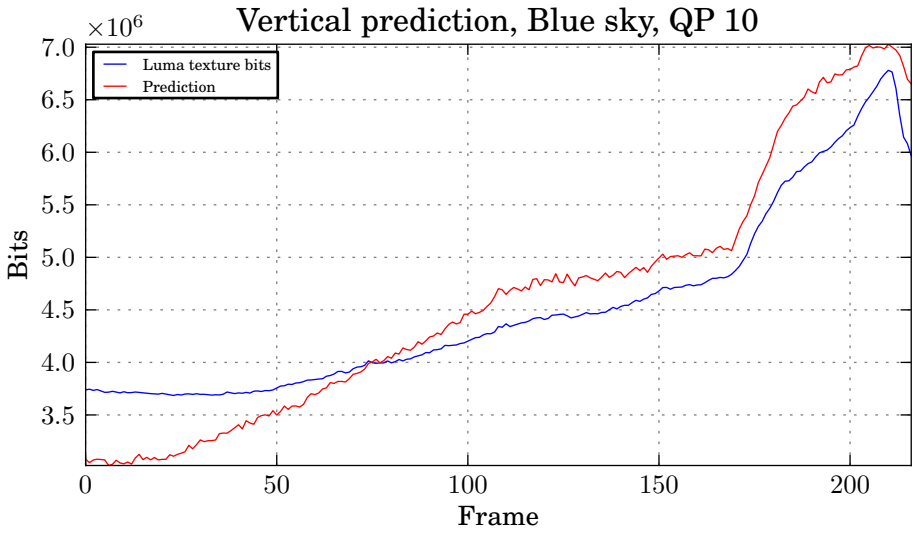


Figure 3.10: Comparison of vertical prediction and the texture bitrate at QP 10, for the movie *blue sky*, all frames are intra coded. Note that the vertical predictor is scaled to match the mean of the bitrate.

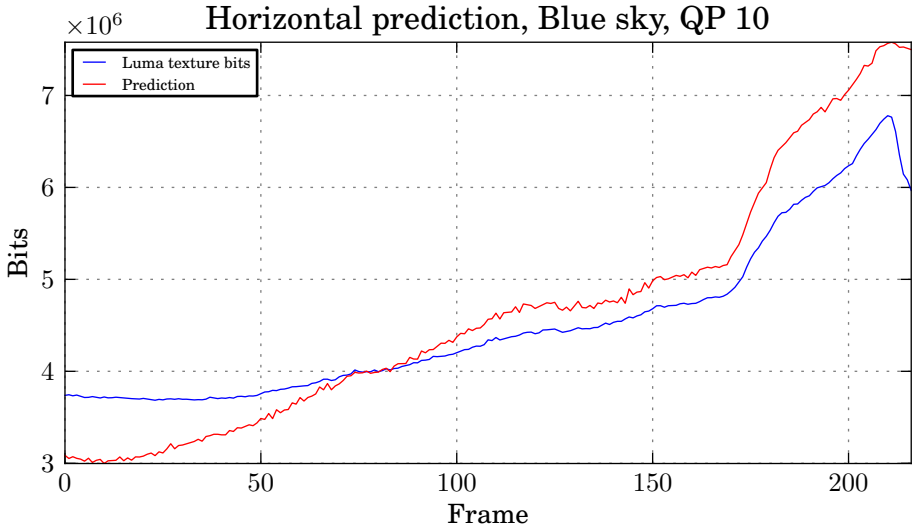


Figure 3.11: Comparison of horizontal prediction and the texture bitrate at QP 10, for the movie *blue sky*, all frames are intra coded. Note that the horizontal predictor is scaled to match the mean of the bitrate.

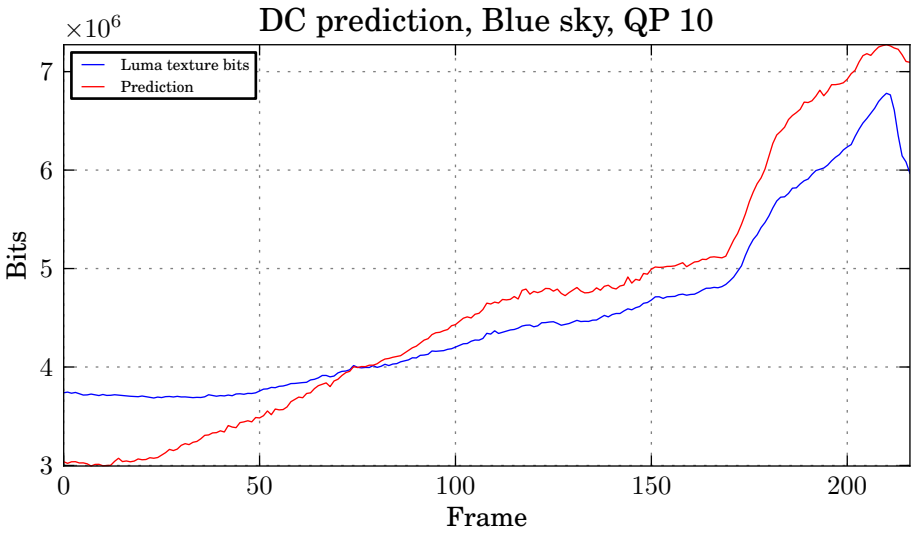


Figure 3.12: Comparison of DC prediction and the texture bitrate at QP 10, for the movie *blue sky*, all frames are intra coded. Note that the DC predictor is scaled to match the mean of the bitrate.

3.3 Conclusions

As expected from the findings in [7, 8], the amount of zeros after quantization provide a good linear predictor of the resulting bitrate. The operations needed for implementing the predictor are simple, but requires knowledge on the quantization process in order to know the deadzone threshold and step size. The process of controlling rate in real time from DCT values, requires a deadline for motion and intra prediction, leaving time for communication with a rate controller, prior to quantizing the residuals.

From the coefficients in table 3.2, we see that many of the predictors show correlation with the resulting bitrate, it is however also clear, that a smooth video like blue sky is not representative for movies in general. When the sequence becomes longer and more complex, we see that some measurements are less correlated. From the literature, and after inspecting figure 3.2 we choose to continue with the number of DCT zeros as our bitrate predictor.

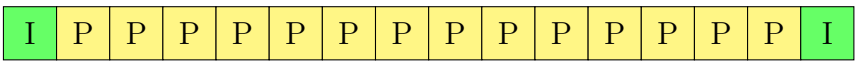
4 Buffers and GOPs

This chapter describes a Group Of Pictures (GOP) and some of the buffers used in H.264.

4.1 Group Of Pictures

A GOP, or *coded video sequence*[1], in predictive video compression is typically an I-frame followed by a number of predictive frames. The I-frame marks the beginning of a section of the movie, and in some standards it provides access to the stream without prior decoded frames. In H.264 an Instantaneous Decoding Refresh (IDR) frame is needed in order to begin decoding the stream. It resets all reference picture buffers, making it impossible for the encoder to reference pictures prior to the IDR.

(a) Bi-Predictive GOP



(b) Predictive GOP

Figure 4.1: Two different GOP types. The rightmost I-frame marks the beginning of the next GOP. The number of frames in a GOP is mainly limited by practical concerns, e.g. error resilience and random access.

Figure 4.1 shows two different types of GOPs, both with predictive frames. GOPs are often used for "GOP-level" rate control, where a bit budget is set for one GOP at the time, and then encoding parameters are adjusted per frame or MB to fit the budget.

4.2 Buffers

Several buffers can be found in H.264 encoding and transmission: input buffers for each of the operations seen in figure 2.4 on page 10, Decoded Picture Buffers (DPBs), output buffers, and a transmission buffer after multiplexing. The input buffers are implementation specific, so they will not be covered here.

DPBs are used for inter prediction and optionally for calculating the error introduced by quantization, they are equal to the decoded pictures, and are thus the pictures used for motion searches. The encoder must decode all encoded reference frames.

Output buffers can be explained as a First In, First Out (FIFO) buffer with a fixed rate output, using a "leaky bucket" analogy as seen in figure 4.2 and from the formula

$$F_{i+1} = F_i + B_i - \frac{R_B}{f}, \tag{4.1}$$

where F_i is the buffer fullness at time i, B_i is the bits of frame i, R_B is the constant output rate of the buffer, and f is the frame rate.

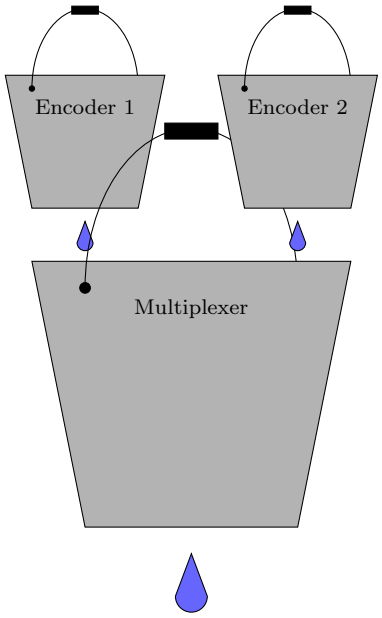


Figure 4.2: Figure showing output buffers of two Elementary Stream (ES) encoders, "leaking" constant flows into a multiplexer, also with an output buffer.

Figure 4.2 shows the general setup of encoders with a constant output bitrate from their buffers into a multiplexer. The classical approach of rate control on individual ESs is simple to implement, and is often based entirely on buffer fullness[9]. When multiplexing ESs into a Compound Stream (CS), some buffers might have gotten overfull - resulting in skipped MBs or frames,

and some buffers might have run empty - wasting bitrate which could have been used on other ESs.

An implementation with a single output buffer and joint rate control between encoders should make it possible to even out fluctuations in quality such as those introduced by the smaller, individual buffers of contemporary encoders.

5 Rate-Control

In the following we discuss the different aspects of a rate-controller and its requirements. We also state a problem and its optimal Rate-Control (RC) solution.

5.1 Allocation problem

The rate-controller will have to allocate bits to each ES taking a number of constraints into account, such as channel bandwidth, buffer usage, quality fluctuations, etc. Ideally, the rate-controller will allocate bits such that the whole channel bandwidth is used at all times. Maximizing the bandwidth usage is fulfilled as long as the multiplexer has enough data to fill the channel. Instead of the channel bandwidth we contemplate a transmission buffer placed between the multiplexer and the channel. Using the leaky bucket analogy the multiplexer fills the bucket and the channel drains it. If $R_{\max}$ is the buffer size then we wish to allocate bits such that the buffer fullness is close to some fraction of the total buffer, denoted R_{target}, such that we do not encounter overflow or underflow and we allow some slack for the bit allocation, which is necessary as the prediction is not exact. If R_{used} is the amount of the buffer currently in use, we have that $R_{\text{free}} = R_{\text{target}} - R_{\text{used}}$ is the amount of bit we have to fill into the buffer to reach the buffer target. This leads us to the first constraint for the rate-control problem, namely

$$\sum_{n=1}^{N} \boldsymbol{x}_i \leq R_{\text{free}} \tag{5.1}$$

where the vector $\boldsymbol{x} \in \mathbb{R}_+^N$ contains the bits allocated for each ES. The sum of bits needs to be less than or equal to R_{free} because of the buffer limitations. There are scenarios where the encoders are unable to produce a total bit amount of R_{free}. The encoder leads to two additional constraints, based on the QP limits. With a QP of 0 we have the highest possible amount of bits and at QP at 51 we have the lowest possible bitrate, if not considering a frame skip. Denoting these two extremes as $\boldsymbol{r}_{\max}$ and $\boldsymbol{r}_{\min}$ we have the additional constraints

$$\boldsymbol{x} \preceq \boldsymbol{r}_{\max} \tag{5.2}$$
$$\boldsymbol{x} \succeq \boldsymbol{r}_{\min} \tag{5.3}$$

where $\preceq$ and $\succeq$ are element wise operators, $\boldsymbol{r}_{\max}$ and $\boldsymbol{r}_{\min}$ are vectors containing the approximated maximum and minimum bits, respectively, possible to produce for each encoder.

The first three constraints are based on the technical limitations for a causal system, given the setup in figure 1.1. There are one additional consideration that concerns the perceptual quality of the video that poses a limitation for the rate-control. We wish the bitrate to be as high as possible, but large variation in the quantization, and thereby also the quality, between frames are perceptually worse than a constant lower bitrate. If $\boldsymbol{m}$ is the predicted bitrates for each ES if neither of them changes their QP, we can limit the variation with the ℓ_2-norm as

$$\|\boldsymbol{M}\boldsymbol{x} - \mathbf{1}\|_2 \,, \tag{5.4}$$

where $\boldsymbol{M} = \operatorname{diag}(\boldsymbol{m})^{-1}$. Taking the ratio between $\boldsymbol{x}$ and $\boldsymbol{m}$ in this way, we adjust the ratio of change instead of the displacement, such that low bitrates will change less than larger bitrates, and vice versa. We can now state our bit allocation as a constrained optimization problem as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} - \Delta \|\boldsymbol{M}\boldsymbol{x} - \mathbf{1}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{x} \preceq \boldsymbol{r}_{\max} \\ & \boldsymbol{x} \succeq \boldsymbol{r}_{\min} \\ & \mathbf{1}^{\mathrm{T}}\boldsymbol{x} \leq R_{\text{free}} \end{aligned} \tag{5.5}$$

where $\boldsymbol{c}$ is a weight vector, such that the ESs can be prioritized, allocating more bits for some streams and less for others. For equal bandwidth distribution we simply use a vector $\mathbf{1}$ where all elements are equal to 1, such that $\boldsymbol{c} = \mathbf{1}$. Δ is a constant adjusting how much the allocated bits may deviate compared to those of a constant quality. Solving problem 5.5 leads to the optimal bit allocation in terms of ES priority, channel bandwidth, and perceptual quality.

5.2 Convexity

For generality we write problem 5.5 into a minimization problem and evaluate the Hessian. Writing the constraints from problem 5.5 in matrix form, the problem is given by

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) = \Delta \cdot \|\boldsymbol{M}\boldsymbol{x} - \mathbf{1}\|_2^2 - \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \tag{5.6}$$

where

$$\boldsymbol{A} = \begin{bmatrix} -\mathbf{1}^{\mathrm{T}} \\ \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} -R_{\text{free}} \\ \boldsymbol{r}_{\min} \\ -\boldsymbol{r}_{\max} \end{bmatrix},$$

and $\boldsymbol{I}$ is the identity matrix. Given the problem is convex it can be solved by a number of algorithms for convex optimization. Investigating the convexity we start by writing out the ℓ_2-norm term, such that

$$f(\boldsymbol{x}) = \Delta \cdot \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x} - \Delta \cdot \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}^{\mathrm{T}}\mathbf{1} - \Delta \cdot \mathbf{1}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x} + \mathbf{1}^{\mathrm{T}}\mathbf{1} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \tag{5.7}$$

Differentiating with regard to $\boldsymbol{x}$ once, we have the gradient as

$$\nabla f(\boldsymbol{x}) = \Delta \cdot (\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M} + \boldsymbol{M}\boldsymbol{M}^{\mathrm{T}})\boldsymbol{x} - \Delta \cdot \boldsymbol{M}^{\mathrm{T}}\mathbf{1} - \Delta \cdot \boldsymbol{M}^{\mathrm{T}}\mathbf{1} - \boldsymbol{c} \tag{5.8}$$

differentiating again we get the Hessian

$$H\left(f(\boldsymbol{x})\right) = \nabla\nabla^{\mathrm{T}} f(\boldsymbol{x}) = \Delta \cdot (\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M} + \boldsymbol{M}\boldsymbol{M}^{\mathrm{T}}) \tag{5.9}$$

since $\boldsymbol{M}^{\mathrm{T}} = \boldsymbol{M}$ we have that the Hessian is given by $2\Delta \cdot \boldsymbol{M}\boldsymbol{M}$, which is a positive definite diagonal matrix, implying that the unconstrained problem is strictly convex. Convexity holds if the constraints form a convex set, which is the case as all constraints in problem 5.6 are linear.

5.3 Example with two streams

As the optimization problem only has inequality constraints, it is seen that when $R_{\text{free}} \geq \mathbf{1}^{\mathrm{T}}\boldsymbol{r}_{\min}$ the optimization problem has a solution, and when $R_{\text{free}} > \mathbf{1}^{\mathrm{T}}\boldsymbol{r}_{\min}$ Slater's condition holds, as there always exist a solution in the relative interior of the constraints. This means that there exist a dual solution with a duality gap of zero.

To demonstrate how the constrained optimization problem can be solved using Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions, we give the following numerical example with two streams. Initially let

$$\boldsymbol{c} = \begin{bmatrix}1\\1\end{bmatrix}, \boldsymbol{m} = \begin{bmatrix}4\\5\end{bmatrix}, \Delta = 1, R_{\text{free}} = 8, \boldsymbol{r}_{\min} = \begin{bmatrix}4\\2\end{bmatrix}, \boldsymbol{r}_{\max} = \begin{bmatrix}7\\5\end{bmatrix} \tag{5.10}$$

which will give a solution lying on the edge of one $\boldsymbol{r}_{\min}$ and one $\boldsymbol{r}_{\max}$ constraint, and the R_{free} constraint, chosen this way for a nice example. Each of the constraints in problem 5.5 are given Lagrange multipliers, and the Lagrangian becomes

$$f(\boldsymbol{x}) = \Delta\|\boldsymbol{M}\boldsymbol{x}-\mathbf{1}\|_2^2-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\lambda(\mathbf{1}^{\mathrm{T}}\boldsymbol{x}-R_{\text{free}})+\boldsymbol{\mu}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{r}_{\max})+\boldsymbol{\epsilon}^{\mathrm{T}}(\boldsymbol{r}_{\min}-\boldsymbol{x}) \tag{5.11}$$

where $\lambda \geq 0$, and $\boldsymbol{\mu}, \boldsymbol{\epsilon} \succeq 0$, and

$$\boldsymbol{M} = \operatorname{diag}(\boldsymbol{m})^{-1} = \begin{bmatrix}\frac{1}{4} & 0\\ 0 & \frac{1}{5}\end{bmatrix}. \tag{5.12}$$

As equation 5.11 is a convex problem, solutions lie where the gradient of $f(\boldsymbol{x})$ is zero, leading to the equation

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \Delta 2\boldsymbol{M}^2\boldsymbol{x} - \Delta 2\boldsymbol{M}\mathbf{1} - \boldsymbol{c} + \lambda\mathbf{1} + \boldsymbol{\mu} - \boldsymbol{\epsilon} = \mathbf{0} \tag{5.13}$$

and isolating for $\boldsymbol{x}$ yields

$$\boldsymbol{x} = \boldsymbol{M}^{-1}\mathbf{1} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{c} - \frac{\lambda}{2\Delta}\boldsymbol{M}^{-2}\mathbf{1} - \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{\mu} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{\epsilon} \quad (5.14)$$

and we see that $\boldsymbol{x}$ depends on the Lagrange multipliers. Differentiating $f(\boldsymbol{x})$ with regard to the Lagrange multipliers, we get an expression for each multiplier. Starting with λ we have

$$\nabla_\lambda f(\boldsymbol{x}) = \mathbf{1}^{\mathrm{T}}\boldsymbol{x} - R_{\text{free}} = 0 \quad (5.15)$$

Substituting $\boldsymbol{x}$ with equation 5.14 and isolating for λ yields

$$\mathbf{1}^{\mathrm{T}}\left(\boldsymbol{M}^{-1}\mathbf{1} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\left(\boldsymbol{c} - \lambda\mathbf{1} - \boldsymbol{\mu} + \boldsymbol{\epsilon}\right)\right) - R_{\text{free}} = 0 \quad (5.16)$$

$$\lambda = \frac{2\Delta\mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-1}\mathbf{1} - 2\Delta R_{\text{free}} + \mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-2}(\boldsymbol{c} - \boldsymbol{\mu} + \boldsymbol{\epsilon})}{\mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-2}\mathbf{1}}. \quad (5.17)$$

Differentiating $f(\boldsymbol{x})$ with regard to $\boldsymbol{\mu}$ yields

$$\nabla_{\boldsymbol{\mu}} f(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{r}_{\max} = \mathbf{0} \quad (5.18)$$

and substituting $\boldsymbol{x}$ yields

$$\boldsymbol{M}^{-1}\mathbf{1} + \boldsymbol{M}^{-2}\frac{\boldsymbol{c} - \lambda\mathbf{1} - \boldsymbol{\mu} + \boldsymbol{\epsilon}}{2\Delta} - \boldsymbol{r}_{\max} = 0 \quad (5.19)$$

$$\boldsymbol{\mu} = \Delta 2\boldsymbol{M}\mathbf{1} - \Delta 2\boldsymbol{M}^2\boldsymbol{r}_{\max} + \boldsymbol{c} - \lambda\mathbf{1} + \boldsymbol{\epsilon}. \quad (5.20)$$

The expression for $\boldsymbol{\epsilon}$ is closely related to that of $\boldsymbol{\mu}$, and can be found in a similar way

$$\nabla_{\boldsymbol{\mu}} f(\boldsymbol{x}) = \boldsymbol{r}_{\min} - \boldsymbol{x} = \mathbf{0} \quad (5.21)$$

Again substituting $\boldsymbol{x}$ yields

$$\boldsymbol{M}^{-1}\mathbf{1} + \boldsymbol{M}^{-2}\frac{\boldsymbol{c} - \lambda\mathbf{1} - \boldsymbol{\mu} + \boldsymbol{\epsilon}}{2\Delta} - \boldsymbol{r}_{\min} = 0 \tag{5.22}$$

$$\boldsymbol{\epsilon} = 2\Delta\boldsymbol{M}^{2}\boldsymbol{r}_{\min} - 2\Delta\boldsymbol{M}\mathbf{1} - \boldsymbol{c} + \lambda\mathbf{1} + \boldsymbol{\mu} \tag{5.23}$$

We now have expressions for $\boldsymbol{x}$ and all the Lagrange multipliers, and we can evaluate the different solutions using the KKT conditions. The unconstrained solution is found for $\lambda = 0$, and $\boldsymbol{\mu}, \boldsymbol{\epsilon} = \mathbf{0}$ as

$$\begin{aligned}
\boldsymbol{x} &= \boldsymbol{M}^{-1}\mathbf{1} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{c} \\
\boldsymbol{x} &= \begin{bmatrix}4\\5\end{bmatrix} + \frac{1}{2}\begin{bmatrix}16 & 0\\0 & 25\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} \\
\boldsymbol{x} &= \begin{bmatrix}12.0\\17.5\end{bmatrix}
\end{aligned} \tag{5.24}$$

Evaluating the solution from equation 5.24 with regard to the constraint $\mathbf{1}^{1}\boldsymbol{x} \leq R_{\text{free}}$, we have that

$$\mathbf{1}^{\mathrm{T}}\boldsymbol{x} = 29.5 \nleq 8 \tag{5.25}$$

showing that $\boldsymbol{x}$ is an infeasible point and therefore not a solution. The problem is then solved for $\lambda > 0, \boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\epsilon} = \mathbf{0}$, leading to

$$\boldsymbol{x} = \boldsymbol{M}^{-1}\mathbf{1} + \boldsymbol{M}^{-2}\frac{\boldsymbol{c}}{2\Delta} - \boldsymbol{M}^{-2}\frac{\lambda\mathbf{1}}{2\Delta} \tag{5.26}$$

Using equation 5.17 we have

$$\lambda = \frac{1\cdot 2\cdot 9 - 1\cdot 2\cdot 8 + \begin{bmatrix}16 & 25\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}}{41} = \frac{43}{41} \tag{5.27}$$

Substituting λ in equation 5.26 with equation 5.27 we have

$$\boldsymbol{x} = \begin{bmatrix} 12 \\ 17.5 \end{bmatrix} - \frac{1}{2} \cdot \frac{43}{41} \begin{bmatrix} 16 & 0 \\ 0 & 25 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.61 \\ 4.39 \end{bmatrix} \tag{5.28}$$

It is seen that the constraint $\mathbf{1}^{\mathrm{T}}\boldsymbol{x} \leq 8$ is fulfilled, but now $\boldsymbol{x}_1 \ngeq \boldsymbol{r}_{\min,1}$. We then solve for $\lambda > 0, \boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\epsilon}_1 > 0, \boldsymbol{\epsilon}_2 = 0$, then

$$\boldsymbol{x} = \boldsymbol{M}^{-1}\mathbf{1} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{c} - \frac{\lambda}{2\Delta}\boldsymbol{M}^{-2}\mathbf{1} + \frac{1}{2\Delta}\boldsymbol{M}^{-2}\boldsymbol{\epsilon} \tag{5.29}$$

with

$$\begin{aligned} \boldsymbol{\epsilon} &= 2\Delta\boldsymbol{M}^2\boldsymbol{r}_{\min} - 2\Delta\boldsymbol{M}\mathbf{1} - \boldsymbol{c} + \lambda\mathbf{1} \\ \boldsymbol{\epsilon}_1 &= 2\Delta\frac{\boldsymbol{r}_{\min,1}}{\boldsymbol{m}_1^2} - 2\Delta\frac{1}{\boldsymbol{m}_1} - \boldsymbol{c}_1 + \lambda \\ \boldsymbol{\epsilon}_1 &= \frac{1 \cdot 2 \cdot 4}{16} - \frac{1 \cdot 2}{4} - 1 + \lambda \\ \boldsymbol{\epsilon}_1 &= \lambda - 1 \end{aligned} \tag{5.30}$$

and

$$\begin{aligned} \lambda &= \frac{2\Delta\mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-1}\mathbf{1} - 2\Delta R_{\text{free}} + \mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-2}(\boldsymbol{c} + \boldsymbol{\epsilon})}{\mathbf{1}^{\mathrm{T}}\boldsymbol{M}^{-2}\mathbf{1}} \\ \lambda &= \frac{2}{41} + \frac{16\lambda}{41} + \frac{25}{41} \\ \lambda &= \frac{27}{25} \end{aligned} \tag{5.31}$$

then substituting equation 5.31 into equation 5.30 yields

$$\boldsymbol{\epsilon}_1 = \frac{2}{25} \tag{5.32}$$

Substituting 5.31 and 5.32 into equation 5.14 we get

$$\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix}12.0\\17.5\end{bmatrix} - \frac{27}{25}\cdot\frac{1}{2}\begin{bmatrix}16 & 0\\0 & 25\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}16 & 0\\0 & 25\end{bmatrix}\begin{bmatrix}\frac{2}{25}\\0\end{bmatrix}\\
\boldsymbol{x} &= \begin{bmatrix}12.0\\17.5\end{bmatrix} - \begin{bmatrix}8.64\\13.5\end{bmatrix} + \begin{bmatrix}0.64\\0\end{bmatrix}\\
\boldsymbol{x} &= \begin{bmatrix}4\\4\end{bmatrix} \qquad (5.33)
\end{aligned}$$

The solution in equation 5.33 is feasible as it satisfies all constraints. In fact it is the optimal solution for this problem. In order to ensure the optimal solution, all combinations of lagrange equations must be tested. It is important to note, that for $\boldsymbol{\epsilon}_i > 0$, $\boldsymbol{\mu}_i$ has to be zero while $\boldsymbol{r}_{\text{min,i}} \neq \boldsymbol{r}_{\text{max,i}}$ and vice versa. The amount of possible combinations of equations grows exponentially with the number of streams

$$N_{\text{eq}} = 2 \cdot 3^N \qquad (5.34)$$

which means that problems with many streams might be solved faster with other methods.

5.4 Rate controller modification

After evaluation of the rate controller it is clear, that streams with larger $\boldsymbol{m}$ values than others result in relatively less penalty by the quadratic part of the function, allowing larger growth of these streams than desired. This is solved by adding a $\boldsymbol{M}$ to the linear, negative part of the primal function, so that the function yield

$$\begin{aligned}
f(\boldsymbol{x}) &= \Delta\|\boldsymbol{M}\boldsymbol{x} - \mathbf{1}\|_2^2 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{c} \qquad (5.35)\\
f(\boldsymbol{x}) &= \Delta\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}^2\boldsymbol{x} - 2\Delta\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\mathbf{1} + \Delta\mathbf{1}^{\mathrm{T}}\mathbf{1} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{c}\\
\nabla_x f(\boldsymbol{x}) &= 2\Delta\boldsymbol{M}^2\boldsymbol{x} - 2\Delta\boldsymbol{M}\mathbf{1} - \boldsymbol{M}\boldsymbol{c} \qquad (5.36)
\end{aligned}$$

It is seen from the gradient in equation 5.36 that if the $\boldsymbol{c}$ vector is different from $\mathbf{1}$, then the amount of growth in bitrate for the unconstrained problem

changes with $\boldsymbol{c}$. This is however not handled in the optimization problem, as it is easier handled by lowering the $\boldsymbol{r}_{\max}$ vector to the maximally allowed growth. Setting Δ to allow a certain growth factor α on average is done by first isolating Δ as

$$\begin{aligned}
\nabla_x f(\boldsymbol{x}) &= 2\Delta \boldsymbol{M}^2\boldsymbol{x} - 2\Delta \boldsymbol{M}\mathbf{1} - \boldsymbol{M}\boldsymbol{c} = 0 \\
2\Delta(\boldsymbol{M}^2\boldsymbol{x} - \boldsymbol{M}\mathbf{1}) &= \boldsymbol{M}\boldsymbol{c} \\
2\Delta(\boldsymbol{M}\boldsymbol{x} - \mathbf{1}) &= \boldsymbol{c} \\
2\Delta &= \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{c}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x} - \mathbf{1}^{\mathrm{T}}\mathbf{1}} \\
\Delta &= \frac{1}{2} \cdot \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{c}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x} - \mathbf{1}^{\mathrm{T}}\mathbf{1}}
\end{aligned} \tag{5.37}$$

and then substituting $\boldsymbol{x}$ with $(1+\alpha)\boldsymbol{m}$, where α is the allowed change factor, yields

$$\begin{aligned}
\Delta &= \frac{1}{2} \cdot \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{c}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{M}(1+\alpha)\boldsymbol{m} - \mathbf{1}^{\mathrm{T}}\mathbf{1}} \\
\Delta &= \frac{1}{2} \cdot \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{c}}{\mathbf{1}^{\mathrm{T}}\mathbf{1}\alpha}
\end{aligned} \tag{5.38}$$

Different ways of setting the $\boldsymbol{c}$ vector is described in chapter 8.

6 Communication

This chapter describes the information to be communicated between encoders and the rate controller.

6.1 Information

For simplicity we assume a setup where all ESs run at the same Frames Per Second (FPS), such that the rate-controller described in chapter 5 can be used without modifications. As the rate-controller holds the primal problem, a solver, and the buffer $\boldsymbol{R}_{\text{free}}$, it requires the variables $\boldsymbol{r}_{\min}$, $\boldsymbol{r}_{\max}$, and $\boldsymbol{m}$, to find a solution. Furthermore, the encoded header size is required, such that it can be subtracted from the available buffer size. In addition to these necessary conditions for the allocation problem, the encoder also communicates the Peak Signal-to-Noise Ratio (PSNR) and size of the previously encoded frame, such that the buffer status can be updated. As the encoders do not return data for the rate-controller at the same time, the rate-controller will wait until all encoders have data ready before bit allocation.

6.2 Implementation

The communication between the encoders and the rate-controller is implemented as a single text file per encoder, and all information is exchanged via this file. The encoder writes its variables for the allocation problem as a comma separated string. The rate-controller reads the information from all encoders, solves the allocation problem, and writes back the requested bitrates. The file is truncated prior to each write. All strings are terminated with *end of text* (ETX), defined as 3 by ASCII, such that a partial written string will not be read as complete information. All communication from the encoders to the rate-controller will start with *JM* and from the rate-controller to the encoders will start with *R*, thereby controlling in what direction the data is meant to go.

Protocol

Communication from the encoder to the rate-controller follows the syntax

`JM minimum_bitrate, current_bitrate, maximum_bitrate, header_size, previously_encoded_bits, previous_PSNR ETX`

note that whitespaces have been added for easier reading, and should be omitted and `ETX` is *end of text*. The syntax for rate-controller to encoder communication follows

`R requested_bitrate ETX`

again, whitespaces should be omitted.

7 Implementation

The Joint Model (JM) reference encoder version 18.4 is written in C, and as it is part of the basis of the H.264 standard, it is capable of encoding videos with all features from the standard including scalable and multi-view coding. As the H.264 standard has many features the JM software has become very complex, however, we have been able to limit the features used by JM to an extend where we have succeded in getting enough information from JM to do our rate-control, and it has been possible for us to use the bit budget returned by the rate-controller to set a QP. It is important to note that while the JM rate control is capable of tweaking R-D for separate sub-macroblocks, we are setting our QP for the entire frame, thus not performing as well on R-D as JM but this is implementation specific.

JM runs through rate-control, prediction, residual transform, quantization, and NAL encoding on a macroblock basis, and it is thus not immediately possible for us to get the transformed residuals for a full frame prior to quantization. Instead of rewriting the entire encoder, we have chosen a method with very little invasion to the original functionality, but with a high computational overhead. We encode the entire frame, counting coefficients after the DCT, and allow for the encoder to complete encoding the frame. We then read out the header size, and perform our predictions on the data collected and communicate with our proposed rate-controller for a bit budget. Once the bit budget is received and we have chosen a suitable QP from our predictor, we clear some buffers in JM and let it encode the image again, with our newly chosen QP.

For this implementation we store much of our information in a globally accessible struct shown in listing 7.2. When doing the DCT transform of the residuals from prediction, we create a number of histograms. The histograms are created based on where in the transform the coefficient lies, due to the way the quantization thresholds are calculated. The 4×4 DCT has three different quantization parameters, and the number of bins needed for the histograms is based on the placement, frame type, and QP resulting in the highest zero-threshold. As the threshold is symmetric around zero, the negative coefficients will be counted along with their positive counterparts.

In order not to collide with existing functions and clutter the namespace, all of our global definitions, as well as some of our local functions are prefixed `AAU_project`.

We define the amount of bins to be used, and the `ETX` character to be used in communication with the rate-controller. The definitions are read by the pre-compiler.

```
47 #define AAU_project_bins 1917
48 #define ETX 0x3
```

Listing 7.1: Defines in JM/lencod/inc/global.h

The global struct definition makes use of the amount of bins previously assigned, and is initialized immediately.

```
50 struct AAU_project_struct
51 {
52         int transform4x4[3][AAU_project_bins];
53         int transform8x8[6][AAU_project_bins];
54         int transform4x4_count;
55         int transform8x8_count;
56         int prev_total_bits;
57         int prev_header_bits;
58         int prev_psnr_luma;
59         int header_bits;
60         int qp;
61         int count;
62         int zeros;
63         int pass;
64         int theta;
65         int offset;
66         int md;
67 } AAU_project_struct;
```

Listing 7.2: Project struct in JM/lencod/inc/global.h

In order to know which coefficients belong to which histograms, a couple of lookup tables are defined

```
71 static const int AAU_project_4x4_quant_coefs[4][4] = {
72 {0, 1, 0, 1},
73 {1, 2, 1, 2},
74 {0, 1, 0, 1},
75 {1, 2, 1, 2}};
76
77 static const int AAU_project_8x8_quant_coefs[8][8] = {
78 {0, 1, 2, 1, 0, 1, 2, 1},
79 {1, 3, 4, 3, 1, 3, 4, 3},
80 {2, 4, 5, 4, 2, 4, 5, 4},
81 {1, 3, 4, 3, 1, 3, 4, 3},
82 {0, 1, 2, 1, 0, 1, 2, 1},
83 {1, 3, 4, 3, 1, 3, 4, 3},
84 {2, 4, 5, 4, 2, 4, 5, 4},
85 {1, 3, 4, 3, 1, 3, 4, 3}};
```

Listing 7.3: Histogram matrices in JM/lencod/inc/global.h

The histograms are cleared before encoding each frame, but the amount of zeros from the previous picture and the QP must be set prior to encoding the first picture. This is done in the main function, immediately before starting the encode process. The starting QP is set to the initial P-frame QP from the configuration file, and the amount of zeros from the previous frame is set to negative one.

```
267 AAU_project_struct.qp = p_Enc->p_Inp->qp[0];
268 AAU_project_struct.zeros = -1;
```

Listing 7.4: Initialization of QP and zeros in JM/lencod/src/lencod.c

The deadzone used for quantization changes with the frame type, so this is set for each frame. Also, we have implemented a way of asking the user for the next frame type, which, when enabled, gives full control of the GOP for testing.

```
823 char inpbuffer[1];
824 inpbuffer[0] = ' ';
825 if (0) { //Disabled
826     printf("Slice type [%c]: ", (p_cur_frm->type == I_SLICE ? 'I' : 'P'));
827
828     scanf("%c",inpbuffer);
829
830     if (inpbuffer[0] != '\n')
831         p_cur_frm->type = (inpbuffer[0] == 'I' ? I_SLICE : P_SLICE);
832
833     while (inpbuffer[0] != '\n') {
834         scanf("%c",inpbuffer);
835     }
836 }
837
838 AAU_project_struct.offset = (p_cur_frm->type == I_SLICE ? 682 : 342);
```

Listing 7.5: Setting offset from frame type, and also possibility to set frame type manually in JM/lencod/src/lencod.c

Most of our calculations are performed in `image.c`, the following function calculates the zero-thresholds for the 4×4 transformed coefficients given a QP, this could also be implemented as a lookup table. See ρ-prediction in section 3.2 for information on the predictor.

```
103 void threshold4x4(int QP, int *aThreshold) {
104     int scaleMatrix[] = {
105                           13107, 5243, 8066,
106                           11916, 4660, 7490,
107                           10082, 4194, 6554,
108                            9362, 3647, 5825,
109                            8192, 3355, 5243,
110                            7282, 2893, 4559};
111     int QPmod6 = QP % 6;
112
113     int x;
114     for(x=0; x<3; x++) {
115         int scale = scaleMatrix[3*QPmod6 + x];
116         int divisionFactor = QP / 6;
117
118         /* Calculates:
119         * (2^(15 + QP/6) - (offset * 2^(4 + QP/6))) / scale
120         * and floors.
121         */
122         aThreshold[x] = ((1 << (15 + divisionFactor)) -
    (AAU_project_struct.offset * (1 << (4 + divisionFactor)))) / scale;
123     }
124 }
```

Listing 7.6: Calculates thresholds given a QP in JM/lencod/src/image.c

The zero-thresholds are used to summarize the amount of texture coefficients that are truncated to zero if the provided QP is used.

```
167 int quantizedZeros4x4(int QP) {
168     int aThreshold[3];
169
170     /* Get quantization thresholds */
171     threshold4x4(QP, aThreshold);
172
173     int histogramID;
174     int histogramIndex;
175     int sum = 0;
176     for(histogramID=0; histogramID<3; histogramID++) {
177         histogramIndex = 0;
178         while (histogramIndex < aThreshold[histogramID]+1) {
179             sum +=
    AAU_project_struct.transform4x4[histogramID][histogramIndex++];
180         }
181     }
182     return sum;
183 }
```

Listing 7.7: Function to count zero-coefficients given QP in JM/lencod/src/image.c

Predictions of the texture bitrates at a given QP is based on the amount of transformed coefficients, the amount of coefficients being truncated to zero, and a predictor, which is created from statistics from the previous frame.

```
186 int predictBitrate(int qp) {
187     int zeros, count;
188     float rho;
189     int prediction;
190 
191     zeros = quantizedZeros4x4(qp) + quantizedZeros8x8(qp);
192     count = AAU_project_struct.transform4x4_count +
        AAU_project_struct.transform8x8_count;
193     rho = (float) zeros / (float) count;
194     prediction = (int) AAU_project_struct.theta * (1.0 - rho);
195 
196     return prediction;
197 }
```

Listing 7.8: Function to predict bitrates from QPs in JM/lencod/src/image.c

The predictor is updated from the percentage of zeros in the previous frame, and the actual amount of texture bits generated.

```
225 void updatePredictor( void ) {
226         int texture_bits;
227 
228         float zeroPercentage = AAU_project_struct.zeros / (float)
        AAU_project_struct.count;
229 
230         texture_bits = AAU_project_struct.prev_total_bits -
        AAU_project_struct.prev_header_bits;
231 
232         AAU_project_struct.theta = (int) texture_bits / (1 -
        zeroPercentage);
233 }
```

Listing 7.9: Updates predictor from statistics in JM/lencod/src/image.c

Once a texture bit budget is returned from the rate-controller, the QP is set to the one providing the closest prediction.

```
199 int newQP(int targetBitrate) {
200     int QP;
201     int predictedBitrate = 0;
202     int previousBitrate = 0;
203     int theta = AAU_project_struct.theta;
204     float zeroSum;
205     float zeroPercentage;
206     float totalElm = AAU_project_struct.transform4x4_count +
        AAU_project_struct.transform8x8_count;
207
208     for (QP=0; QP<=51; QP++) {
209         zeroSum = quantizedZeros4x4(QP) + quantizedZeros8x8(QP);
210         zeroPercentage = zeroSum / totalElm;
211         predictedBitrate = (int)(theta * (1.0 - zeroPercentage));
212
213         if (predictedBitrate <= targetBitrate) {
214             if ((targetBitrate - predictedBitrate) <= (previousBitrate -
        targetBitrate) || QP == 0) {
215                 return QP;
216             } else {
217                 return QP-1;
218             }
219         }
220         previousBitrate = predictedBitrate;
221     }
222     return 51;
223 }
```

Listing 7.10: Function to choose QP from estimations given a budget in JM/lencod/src/image.c

When not doing mode decision, the transformed coefficients are counted up in the histograms.

```
69  if (AAU_project_struct.md == 0) {
70
71      for (i=0; i < BLOCK_SIZE; i++){
72          for (ii=0; ii<4; ii++){
73              x = iabs(tblock[pos_y + ii][pos_x + i]);
74              if (x < AAU_project_bins) {
75                  c = AAU_project_4x4_quant_coefs[ii][i];
76                  AAU_project_struct.transform4x4[c][x]++;
77              }
78          }
79      }
80
81      AAU_project_struct.transform4x4_count += 16;
82  }
```

Listing 7.11: Count function in JM/lcommon/src/transform.c

The JM encoder communicates its predictions to a rate-controller via a file.

```
279 fprintf(statefile,"JM%d,%d,%d,%d,%d,%d%c",
280     predictBitrate(0),
281     predictBitrate(AAU_project_struct.qp),
282     predictBitrate(51),
283     AAU_project_struct.header_bits,
284     AAU_project_struct.prev_total_bits,
285     AAU_project_struct.prev_psnr_luma,
286     ETX);
```

Listing 7.12: Communication towards the rate-controller in JM/lencod/src/image.c

7.1 Joint rate control

Our rate controller is implemented in python 2, and is started by a script which also handles the setup of the JM encoders.

```
% ./encode JMpath cfgpath qp1,qp2... movie1 movie2 ...
```

The rate controller is implemented in the file `rate-control` and is using scipy for linear algebra and convex optimization. It communicates its findings back towards JM in the state file with the following

```
151 streamID = 0
152 for arg in sys.argv[1:]:
153     with open(arg, 'wb') as fp:
154         out = "R" + str(np.round(x[streamID]).astype(np.int)) + chr(3) +
    '\n'
155         fp.write(out)
156         fp.close()
157         streamID += 1
```

The final newline was added to allow unified functionality across ext4 and btrfs file systems used on our computers.

7.2 Software

The machine used for our tests has the following configuration

```
% source /etc/os-release; echo $PRETTY_NAME
Arch Linux

% uname -sr
Linux 3.7.4-1-ARCH

% python2 -V
Python 2.7.3

% python2 -c "import scipy; print scipy.version.version"
0.12.0

% gcc -v 2>&1 | grep "gcc version"
gcc version 4.7.2 (GCC)

% ffmpeg -version | grep "ffmpeg version"
ffmpeg version 1.2.1
```

It uses the ext4 file system on all partitions.

7.3 Issues

The JM encoder changes header sizes of the encoded streams with QP, which can result in large prediction errors with our setup when QP is changed.

The optimization tool seems to malfunction on scipy version 0.9, all of our tests has been run against scipy version 0.11 or 0.12.

8 Simulations

This documents the results using our rate-controller. We present results for different channel bandwidths, in a setup simulating a transmisson of three ESs, with different weights for quality adjustment. Having a limited bandwidth channel, the most naive approach is to encode all streams to a fraction of the available bandwidth as their mean bandwidth. This is illustrated for comparison and explanation of its shortcomings. A more thorough simulation is done with four streams for 1000 frames, for better evaluation of the performance over time.

The movies *aspen*, *ducks take off*, and *factory*, have been used as they are very different. *Aspen* has little motion but a high level of textural details, *ducks take off* has high detail through the whole stream, and *factory* is an animation.

8.1 Static bitrate

We simulate a limited channel of 3Mbps, 6Mbps, and 12Mbps, and encode 100 frames of each video at a bitrate of one third of the channel bandwidth, using the native JM encoder. The GOP structure is one initial I-frame followed by P-frames. Figure 8.1 depicts the accumulated bitrate for the three test movies, when encoding for constant bitrate. The streams are individually encoded and then summed into a single bitrate. The RC implemented in the JM encoder tries to reach the correct end sum of bitrates on average, explaining the bitrate fluctuations in figure 8.1. Theses fluctuations are only limited by the maximum allowed change of the QP between successive frames, which has been set to the default setting of 4.

While the bitrate fluctuations might not pose a problem, as an output buffer would even it out, the resulting variation in visual quality, depicted in figure 8.2 for the same streams, gives a perceptually bad quality. The PSNR variance for all three bandwidths are given in table 8.1**(a)**.

Using our rate-controller we get an overall lower PSNR, see table 8.1**(b)** and figure 8.4, and a lower PSNR variance except for *aspen*, see table 8.1**(a)**. The lower PSNR is a symptom of the very rough quality steps used in our implementation to fit the bitrate budgets. An implementation able to fit the bitrate target on a MB level could improve the R-D. The PSNR variance

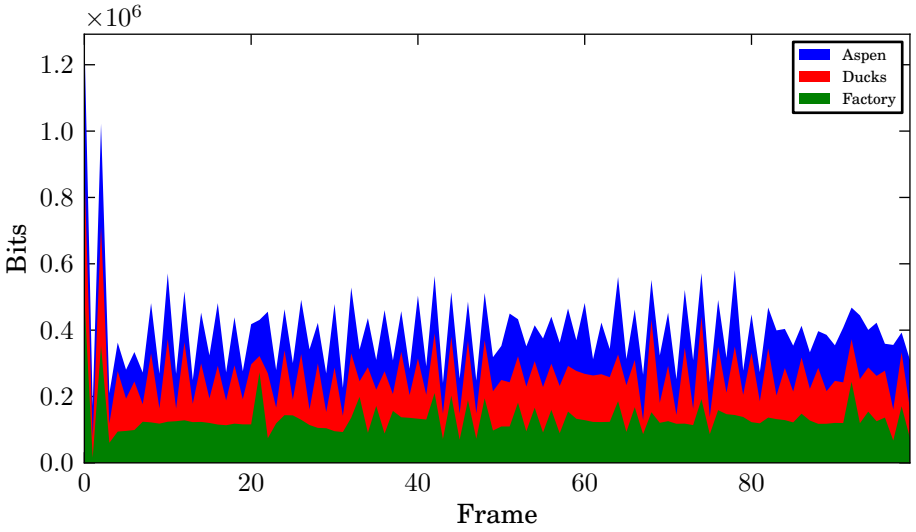


Figure 8.1: Accumulated bitrates for the first 100 frames of *aspen*, *ducks take off*, and *factory*, encoded at a constant bitrate for a channel bandwidth of 12Mbps.

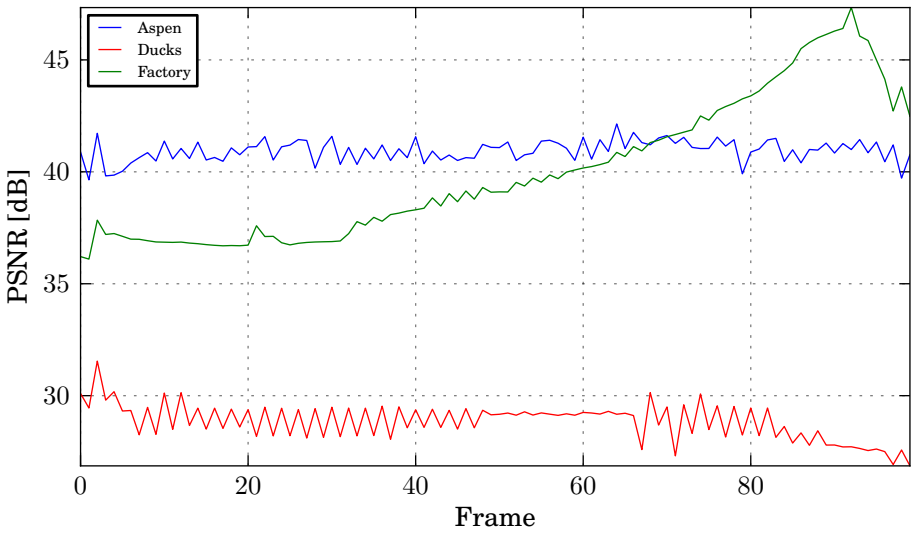


Figure 8.2: Luminance PSNR of *aspen*, *ducks take off*, and *factory* encoded with JM at a target bitrate of 4Mbps each.

is generally lower with our implementation, as it is the main focus of our rate-controller. The variance would improve with a better fit to the bitrate, instead of the over- or undershoots resulted when adjusting the QP of the entire frame. Also, the predictions suffer from large variations in header size between QPs, which we address to the specific JM encoder implementation.

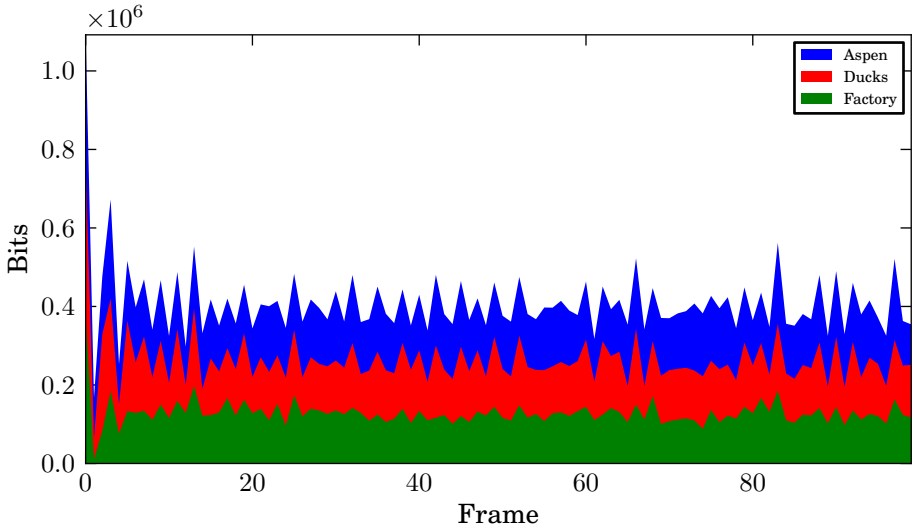


Figure 8.3: Accumulated bitrates for the first 100 frames of *aspen*, *ducks take off*, and *factory*, encoded with fixed and evenly distributed bitrate weights for a channel bandwidth of 12Mbps.

When inspecting the resulting PSNR in figure 8.4 and bitrate in figure 8.3 it is clear, that the target of the rate-controller has been to keep the bitrate in vicinity of a target, and to evenly distribute the bits between ESs while keeping PSNR variation low. However, it is desirable to have lower quality for the *aspen* and *factory* streams, to allow better quality for the much more complex stream *ducks take off*. This is accounted for in the implementation presented in the following section.

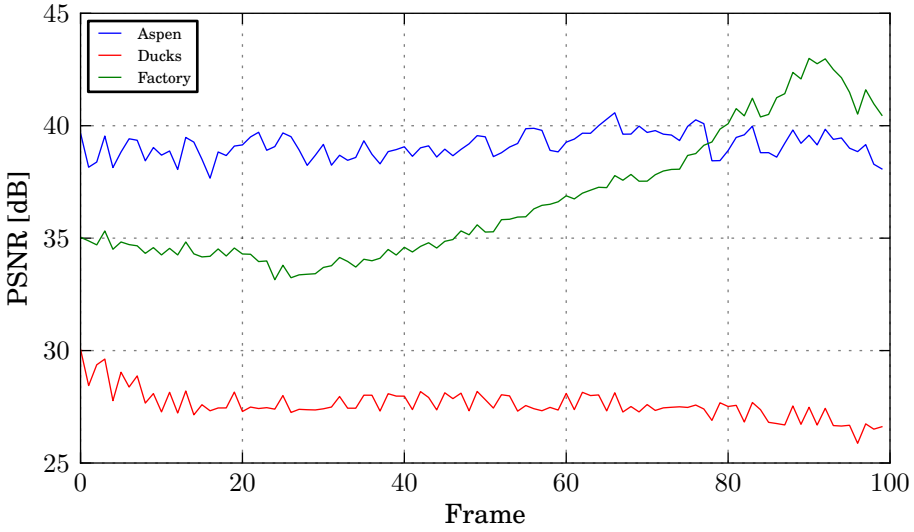


Figure 8.4: Luminance PSNR of *aspen*, *ducks take off*, and *factory* encoded with our rate-controller at a target bitrate of 12Mbps total.

8.2 Prioritized PSNR

Instead of averaging the bitrates we prioritize the PSNRs by using the PSNR diffence between a stream and the stream with the highest quality plus one, which will result in convergence towards a common PSNR for all streams. How the $\boldsymbol{c}$ vector is made can be seen from the following example

$$\mathbf{PSNR} = \begin{bmatrix} 38 \\ 28 \\ 32 \end{bmatrix}$$

$$\boldsymbol{c} = \max(\mathbf{PSNR}) - \mathbf{PSNR} + \mathbf{1}$$

$$\boldsymbol{c} = \begin{bmatrix} 1 \\ 11 \\ 7 \end{bmatrix}$$

As seen by figure 8.6 the PSNR range is reduced, compared to the constant bitrate regulation. As complexity of *ducks take off* is much higher than the two other streams, the PSNR regulations impact on the bitrate is substantial, as seen on figure 8.5. It is near impossible for the rate-controller to adjust

the PSNRs further, as the next step in QP for *ducks take off* would require all the bandwidth currently used for the *aspen* and *factory* streams.

As the bits required for *ducks take off* for a PSNR matching the other streams are extremely high in comparison, we replace it with the *tractor* clip, and do the same PSNR leveling. As seen by figure 8.7 and 8.8 the *tractor* clip also requires more bits than the other streams for the same PSNR, though not as much as *ducks take off*. Table 8.1**(b)** shows that the average PSNR for the *tractor* clip is more than 3 dB higher than *ducks take off* while both *aspen* and *factory* also have increased in average PSNR.

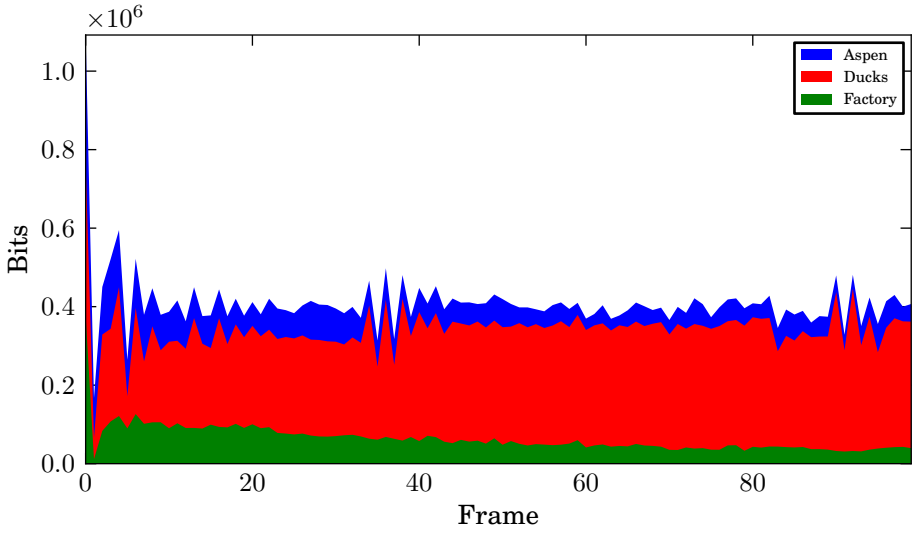


Figure 8.5: Accumulated bitrates for the first 100 frames of *aspen*, *ducks take off*, and *factory*, encoded for equal PSNR values for a channel bandwidth of 12Mbps.

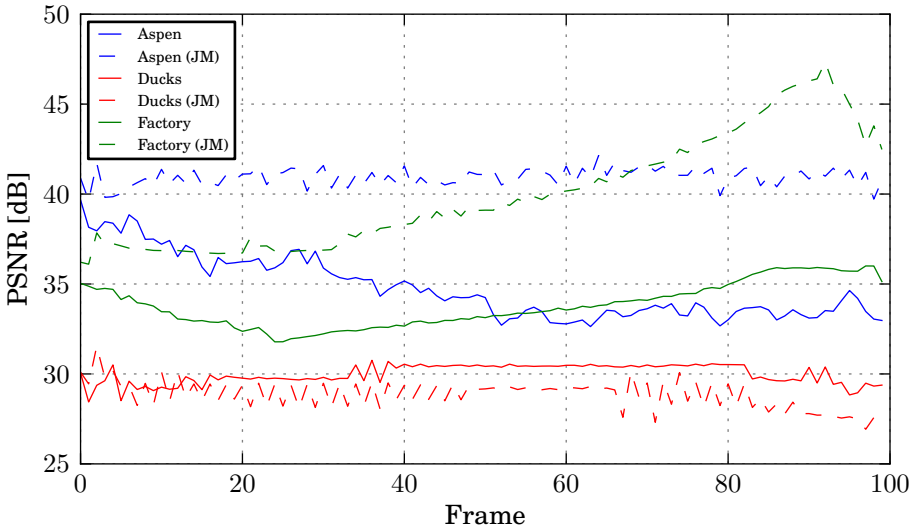


Figure 8.6: Luminance PSNR of *aspen*, *ducks take off*, and *factory* encoded for equal PSNR values with our rate-controller at a target bitrate of 12Mbps total. The results from figure 8.2 is the dotted lines, shown for easier comparison.

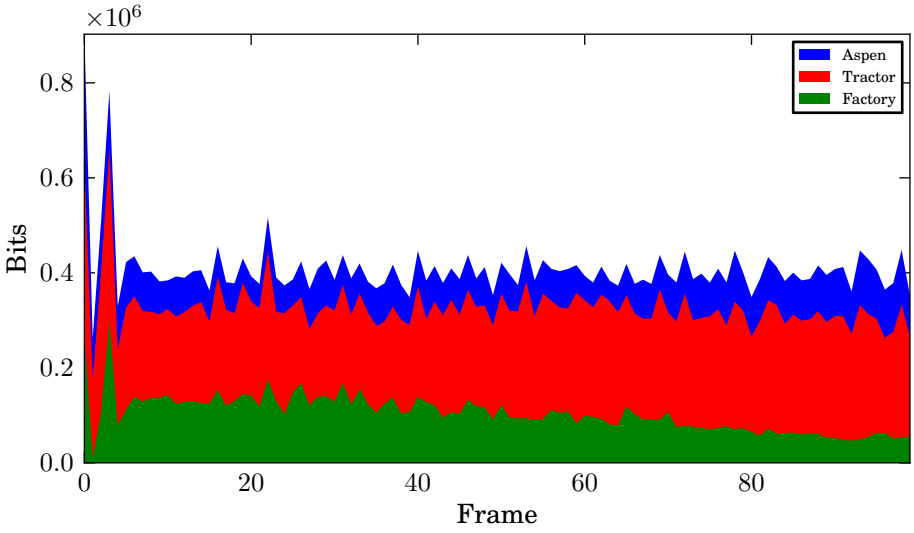


Figure 8.7: Accumulated bitrates for the first 100 frames of *aspen*, *tractor*, and *factory*, encoded for equal PSNR values for a channel bandwidth of 12Mbps.

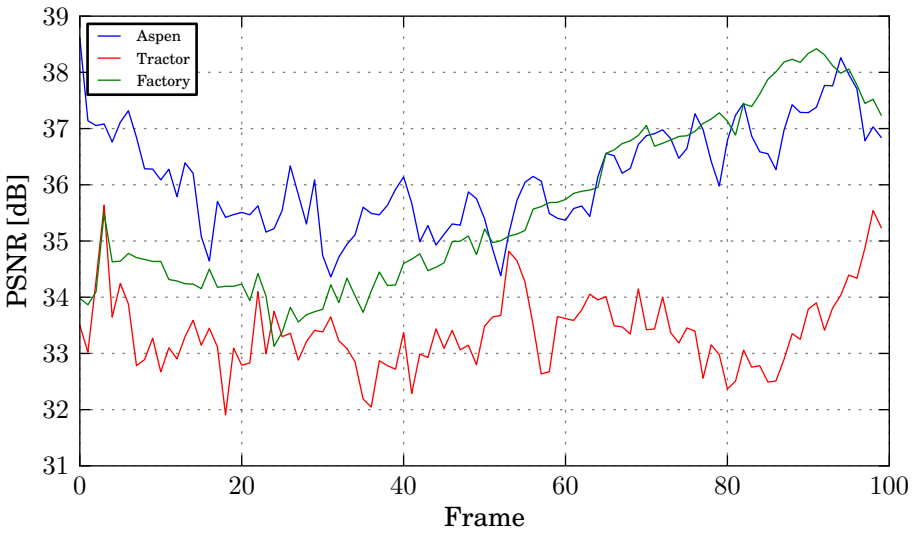


Figure 8.8: Luminance PSNR of *aspen*, *tractor*, and *factory* encoded for equal PSNR values with our rate-controller at a target bitrate of 12Mbps total.

	PSNR variance					
	aspen		*ducks take off*		*factory*	
	JM	Our	JM	Our	JM	Our
3Mbps	0.4536	1.2407	0.3768	0.2783	8.0979	3.9386
6Mbps	0.2934	0.5294	0.4252	0.2273	8.8587	6.9158
12Mbps	0.2304	0.3203	0.6500	0.3930	9.4961	8.0839
	Prioritized PSNR					
12Mbps	-	2.9928	-	0.2813	-	1.4581
	aspen		*tractor*		*factory*	
12Mbps	-	0.8098	-	0.4654	-	2.2294

(a) PSNR variance for the 100 first frames of each movie for different target bandwidths and encoding setups. Where the numbers for the JM encoder is missing, is where it do not have an encoding setup that matches ours, our numbers in those columns can not be compared directly.

	Mean PSNR [dB]					
	aspen		*ducks take off*		*factory*	
	JM	Our	JM	Our	JM	Our
3Mbps	36.4724	31.8196	24.0794	22.6032	35.1657	31.8061
6Mbps	39.1307	35.7004	26.3220	25.1915	37.5469	34.3461
12Mbps	40.9394	39.1356	28.8425	27.5929	39.9657	36.6436
	Prioritized PSNR					
12Mbps	-	34.7747	-	29.9973	-	33.8046
	aspen		*tractor*		*factory*	
12Mbps	-	36.1657	-	33.3812	-	35.5656

(b) Mean PSNR for the JM encoders rate-controller compared with ours, for the first 100 frames of each movie.

8.3 Final evaluation

To test the rate-controller in a realistic scenario, we test it against four streams on a 25 Mbps total bandwidth at 24 FPS. The test streams used are from the movies *elephants dream* and *big buck bunny* where each movie has been split in two at the 6000'th frame, simulating four individual streams. The test will run over 1000 frames.

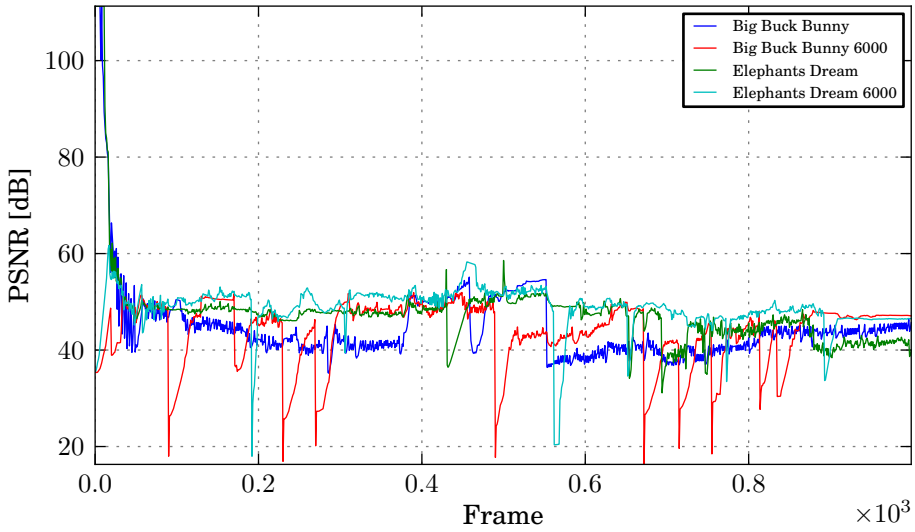


Figure 8.9: Luminance PSNR of the two sequences from *big buck bunny* and the two sequences from *elephants dream* encoded for equal PSNR values with our rate-controller at a target bitrate of 25Mbps total.

The PSNR starts out very high, see figure 8.9, for the two streams starting at frame 1, as their complexity is very low. *Big Buck Bunny* starts out with slow panning over a landscape overlayed with white text, and *Elephants Dream* have a slow moving background. As the change in PSNR is limited by the rate-controller it is unable to fill the buffer as there are produced fewer bits at this initial QP than what is required to fill the buffer, see figure 8.11. The steep decent in PSNR for the same two streams are imposed by the optimization constraint of keeping the sum of bitrates below the given maximum. The PSNRs in table 8.1 indicates a wildly varying PSNR, observing the swings in figure 8.9 this is understandable. However, inspecting the PSNR evolution of the individual streams, and discarding the drops at scene changes, we see that the rate-controller is able to slowly vary the PSNR over time.

	Movie			
	Big Buck Bunny	*Big Buck Bunny* 6000	*Elephants Dream*	*Elephants Dream* 6000
Mean PSNR	44.8428	43.6229	47.5887	48.6956
PSNR variance	77.7117	44.4220	71.0803	19.9419

Table 8.1: Average PSNR and variance for the four streams used in the final simulation.

The sudden drops in PSNR, quite distinctly in *Big Buck Bunny 6000* and *Elephants Dream 6000*, are mainly caused by scene changes, as the bit budget is unable to provide the encoder with enough bandwidth for a constant PSNR.

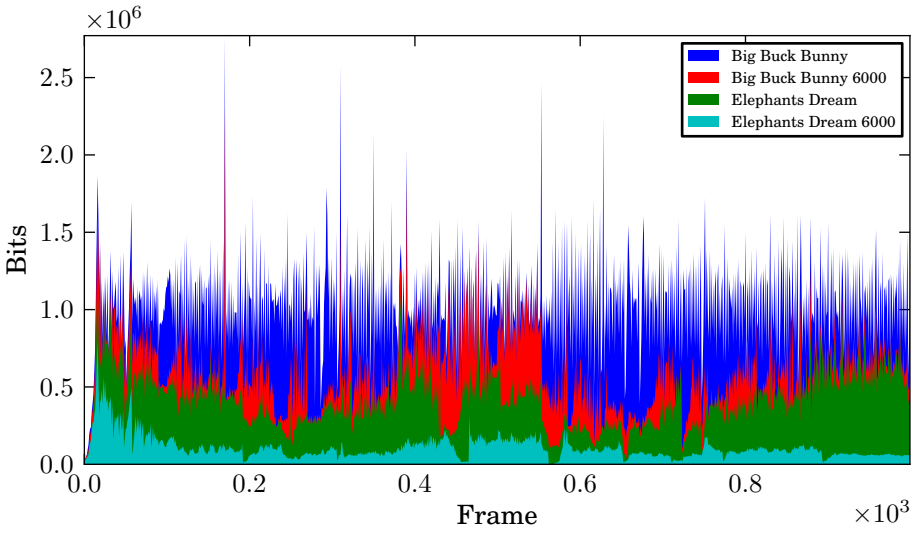


Figure 8.10: Accumulated bits of the two sequences from *big buck bunny* and the two sequences from *elephants dream* encoded for equal PSNR values with our rate-controller at a target bitrate of 25Mbps total.

Figure 8.10 shows the accumulated bits per frame. While the total number of bits vary around the buffer target, there are large variations in the individual streams and their part of the total bandwidth.

Even though the implemented buffer is very simple it is possible to stay within its limit. The buffer fullness and buffer target is plotted in figure 8.11, and it is seen that buffer overflow is never encountered if the buffer is twice the size of the buffer target. This is not guarantied to be the case

for all combination of streams and channel bandwidths, but shows that it is possible to keep the buffer fullness reasonable.

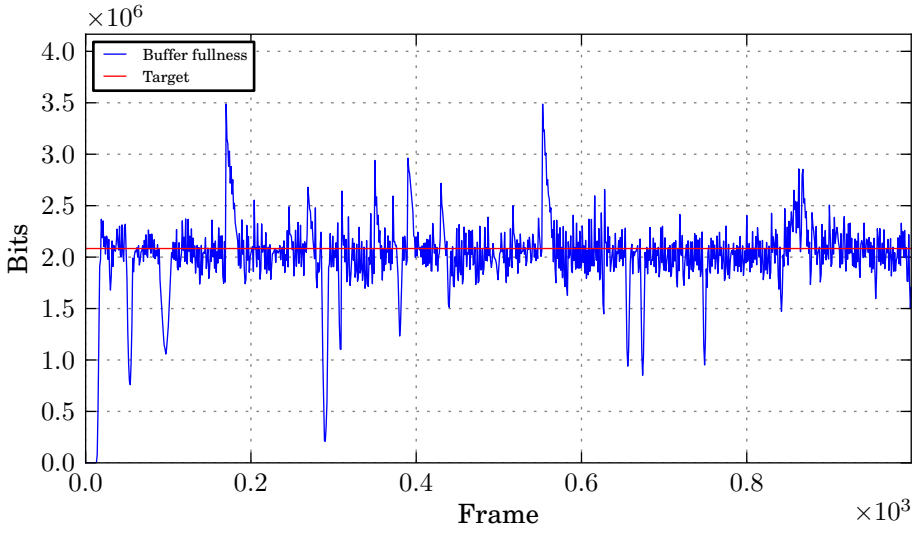


Figure 8.11: Buffer fullness when encoding the four streams for a 25Mbps channel with a frame rate of 24 and a buffer target of approx 2.08 Mbit, which is the same size as two average frames.

9 Conclusions

In this project, we have investigated the H.264 video codec for use in a broadcasting scenario. Our research focuses on accurate bitrate predictions with low complexity, to be used with joint encoding for maximum usage of limited bandwidth channels. We have found that a simple linear predictor, based on the quantization process, provides accurate predictions for texture bits, and combined with the size of the frame header, it is possible to predict the total amount of bits for the frame with accuracy suitable for rate control.

Our implementation used for this project is based on the Joint Model H.264 encoder, which is used for development and testing of the H.264 reference decoder, specified by ITU-T and ISO. The JM encoder is not immediately suited for the type of predictions and rate control we impose, but has been solved by modifing the software for two-pass encoding. Our rate-controller has been implemented in Python 2, using scipy's optimization toolbox for solving the proposed convex optimization problem.

The tests provided in this report show comparisons with sequences encoded by the JM reference encoder for a fixed bitrate. This comparison is not completely fair, as the JM encoder is able to adjust the QP on a macroblock level, whilst we control it on a frame level. Moreover JM has no restrictions on buffer fullness or quality fluctuations, other than the allowed change in QP. The JM encoder also adjusts the frame header sizes considerably between different QPs, resulting in inaccurate predictions in our setup. A fair comparison would require an implementation where it is possible to interchange the rate-controlling functions between an existing rate-controller for broadcasting and ours.

As neighter the JM encoder nor our rate-controller is optimized for real-time encoding, we can not demonstrate a real-time running setup. However we still belive that it is possible to make a real-time implementation, as the additional computations and communication for this scheme is belived to be of relatively low cost, especially when comparing with the complexity of mode decisions and motion estimation in H.264.

It is clear, that in order to enable Just-In-Time encoding with autonomous encoders, they and the rate-controller must obey strict deadlines on predictions and bit allocation, respectively.

10 Discussion

As the supply of high definition content is constantly rising and the distribution cost of such content is a concern for any provider, the need to pack content efficiently is of great importance. Using the distribution channels to their fullest can help distribution companies create value for money, and optimize their investment.

While our focus has been on H.264 encoding, we believe that the findings of this research project applies to other video compression codecs, such as H.265, VP8, and VP9, as well as other transform codecs. As the general principles in video coding is the same, and the great differences lie in implementation choices, they should be compatible with our rate control scheme, with similar results.

Just-In-Time execution is fundamental for this scheme to function, so it is in need for a time efficient, and deadline aware implementation. Reaching such a goal requires attention to the distribution of computations, and with existing hardware it would require some sort of multiprocessing. Many of the tasks used in prediction and transform coding can be performed independently, and therefore great enhancements can be made by utilizing parallel computing units. The parallelism can be acheived by multi-core processors, found in modern CPUs and GPUs. A hardware specific implementation capable of harnessing specific features could reduce encoding time whilst keeping rate-distortion at a high level. Also, application specific hardware could be made to meet the requirements.

The rate-controller could be distributed between encoders, and with a higher degree of knowledge of other streams it might be possible to better allocate the bits. due to the nature of the rate-controller, it is possible to encode for the allocated budget at any link of the production or distribution, be it at the camera, or at the broadcaster. Also, the rate-control scheme allows for dynamically adding or removing streams to or from a channel on the fly, while not disturbing the current channels unnecessarily.

A Test Sequences

This chapter describes the test sequences used for the project.

A.1 NTIA/ITS sequences

```
These video sequences are owned by NTIA/ITS, an agency of the U.S. Federal
Government.  They were created under Project Number 3141012-300, Video Quality
Research, in 2008.  These video sequences may be used for research purposes,
only.  You can use, copy, modify and redistribute them upon your acceptance of
these terms and conditions and upon your express agreement to provide
appropriate acknowledgments of NTIA/ITS's ownership of the video sequences by
keeping this text present with any copied or derived works.

Video Standard: 1080p 30fps
Camera specs:   Panasonic AJ-HDX-900, saved to DVCPro tape.
Video standard: Native
Editing:        19 second clips, intended to remove first 2-sec and final 2-sec.
Scenes:
      Aspen
      RedKayak
      WestWindEasy


Video Standard: 1080p 30fps
Camera specs:   The camera was a Panasonic P2HD AJ-HPX3000G; and the lens a
                Fujinon TV Lens HA22x7.8 BERM-M48. This camera records in H.264
                intra-frame coding at 100 Mbps.
Video standard: Native
Editing:        19 second clips, intended to remove first 2-sec and final 2-sec.
Scenes:
      RushFieldCuts -- shutter speed 30
      SnowMnt -- shutter speed 30
      SpeedBag -- shutter speed 60
      TouchdownPass -- shutter speed 30


Video Standard: 1080p 30fps
Camera specs:   The camera was a Sony HVR-Z1U (HDV format) which was
                converted to SMPTE 292M (high definition 1080i format) and
                recorded onto a Panasonic HD-D5.
Use restrictions: research purposes, only
Notes:          Video of fire was taken of a controlled burn of a Meth house.
Video standard: Native
Editing:        19 second clips, intended to remove first 2-sec and final 2-sec.
Scenes:
      ControlledBurn
```

All clips but one were obtained in y4m format from the "derf" collection, the "RushFieldCuts" clip was obtained in the original NTIA/ITS AVI container. No loss was introduced in the conversion to y4m, and the chroma format of the files is 4:2:2.

A.2 Taurus Media Technik sequences

```
Sequence #Frames Short description
-------------------------------------------------
Blue sky    250 Top of two trees against blue sky. High contrast, small color
                differences in the sky, many details. Camera rotation.
Sunflower   500 Sunflower, very detailed shot. One bee at the sunflower, small
                color differences and very bright yellow. Fixed camera, small
                global motion.
Rush-hour   500 Rush-hour in Munich city. Many cars moving slowly, high depth
                of focus. Fixed camera.
Pedestrian Area 375 Shot of a pedestrian area. Low camera position, people pass
                by very close to the camera. High depth of field. Static camera.
Tractor     761 A tractor in a field. Whole sequence contains parts that are
                very zoomed in and a total view. Camera is following the
                tractor, chaotic object movement, structure of a harvested
                field. Very red wheels of the tractor
Riverbed    250 Riverbed seen through the water. Very hard to code.
Station     313 View from a bridge to munich station. Evening shot. Long zoom
                out. Many details, regular structures (tracks)


Camera: Sony HDW-F900
Recorded on (Tape): HDCam
Stored on: DVS
Frame rate: 25 fps (progressive)
Resolution: 1920x1080
Color subsampling: 4:2:0
Filter Tabs for Subsampling: -0.0063 / 0 / 0.0299 / 0 / -0.0831 / 0 /
                             0.3098 / 0.4994 / 0.3098 / 0 / -0.0831 /
                             0 / 0.0299 / 0 / -0.0063
Color conversion: ITU Rec BT 709 (SMPTE 274M)
Original files contact: oelbaum@ei.tum.de
Restrictions of use: No restrictions
Copyright: No Copyright
Date of Recording: Summer 2001
Source: Taurus Media Technik, Dr. Karl Mauthe
Producer: Martin Kreitl martin.kreitl@KirchGruppe.de
Camera Operator: Jürgen Würzinger
Camera Assistent: Yean Ives Diss

All material was recorded in summer 2001 by Taurus Media Technik.
```

All clips were obtained in y4m format from the "derf" collection, in some sequences, the number of frames differ from those listed in the readme, see table A.1 for "derf" collection frame numbers.

Blue sky	217
Sunflower	500
Rush-hour	500
Pedestrian Area	375
Tractor	690
Riverbed	250
Station	313

Table A.1: Number of frames for the Taurus Media Technik sequences, from the "derf" package.

A.3 SVT sequences

Through the "derf" package, we have acquired parts of the SVT High-Definition Multi Format Test Set of February 2006. The test set has no short readme, but the following information has been extracted from the documentation:

Copyright and Restrictions of Use:

> Individuals and organizations extracting sequences from this archive agree that the sequences and all intellectual property rights therein remain the property of Sveriges Television AB (SVT), Sweden. These sequences may only be used for the purpose of developing, testing and presenting technology standards. SVT makes no warranties with respect to the materials and expressly disclaim any warranties regarding their fitness for any purpose.

All sequences has been acquired in 4:2:0 chroma format, and in y4m file format. A list of the test sequences and their frame counts can be seen in table A.2, as proposed by SVT; they are 10 second clips from the original 50 fps source.

Crowd run	500
Park Joy	500
Ducks take off	500
Into tree	500
Old town cross	500

Table A.2: Number of frames for the SVT sequences.

A.4 HDgreetings sequences

Through the "derf" package, we have obtained test sequences from HDgreetings.

Copyright and Restrictions of Use:

> These clips are provided for benchmarking, research, and testing only. The video is copyrighted material and HDgreetings retains all rights to it. You may post part or all of the content on another site if it's for one of these purposes. If you link to the videos, please link to this page (`http://www.hdgreetings.com/other/ecards-video/video-1080p.aspx`) instead of directly to the videos. Thanks.

The videos are in y4m container format, and are in 4:2:0 chroma format. A list of videos and their frame count can be seen in table A.3.

Factory	1339
Life	825
Dinner	950

Table A.3: Number of frames for the HDgreetings sequences.

A.5 Sintel trailer

Sintel is a short film created by the "Durian Open Movie project", It is licensed under the "Creative Commons Attribution 3.0" license. It is copyrighted Blender Foundation, and further information on copyright or the creators can be seen at `www.sintel.org`. For this project, the 1253 frame trailer is used, acquired from the "derf" package, presented in y4m file format and 4:2:0 chroma format.

A.6 Elephants dream

Has been downloaded in its original 1080p avi container. It is released under the "Creative Commons Attribution license". It is copyrighted Blender Foundation, and further information on the copyright or the creators can be seen at `www.elephantsdream.org`.

A.7 Big Buck Bunny

Big Buck Bunny is a short film created by the "Peach open movie project" and has been downloaded in its original 1080p avi container. It is released under the "Creative Commons Attribution 3.0 license". It is copyrighted Blender Foundation, and further information on the copyright or the creators can be seen at `www.bigbuckbunny.org`.

A.8 Pre-processing

For this project, we wish to have a unified set of raw input sources. A file, and chroma format of YUV 4:2:0 is chosen, all test sequences but Sintel have 1920×1080 pixels resolution, and will not necessarily be used at its native frame rate. All videos are converted from y4m container, and chroma resampled if needed with the following command:

```
% ffmpeg -i [inputfile] -pix_fmt yuv420p [output.yuv]
```

Using

```
% ffmpeg -version | grep "ffmpeg version"
ffmpeg version 0.8.5-4:0.8.5-0ubuntu0.12.04.1
```

List of acronyms

MB	Macroblock
CS	Compound Stream
CU	Coding Unit
DCT	Discrete Cosine Transform
DPB	Decoded Picture Buffer
DVB	Digital Video Broadcasting
ES	Elementary Stream
FIFO	First In, First Out
FPS	Frames Per Second
GOP	Group Of Pictures
IDR	Instantaneous Decoding Refresh
KKT	Karush-Kuhn-Tucker
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
JIT	Just-In-Time
JM	Joint Model
MAD	Mean Absolute Difference
ME	Motion Estimator
MV	Motion Vector
PMF	Probability Mass Function
PSNR	Peak Signal-to-Noise Ratio
R-D	Rate-Distortion

RC Rate-Control

SAD Sum of Absolute Differences

TS Transport Stream

QP Quantization Parameter

Bibliography

[1] ITU-T, *H.264*, January 2012, IEC 14496-15 (MPEG-4 part 15, AVC).

[2] IEC, *Letter symbols to be used in electrical technology*, August 2005, IEC 60027-2.

[3] ISO/IEC, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s: Video*, 1993, IEC 11172-2 (MPEG-1 part 2).

[4] ——, *Information technology – Generic coding of moving pictures and associated audio information: Video*, 2000, IEC 13818-2 (MPEG-2 part 2).

[5] ITU-T, *H.265*, January 2013, IEC 23008-2 (MPEG-H part 2, HEVC).

[6] C. Pang, O. C. Au, J. Dai, and F. Zou, "LMM-based frame-level rate control for H.264/AVC high-definition video coding," *Signal Processing: Image Communication*, vol. 27, no. 7, pp. 737–748, August 2012.

[7] Z. He and S. K. Mitra, "A Linear Source Model and a Unified Rate Control Algorithm for DCT Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 970–982, 2002.

[8] ——, "ρ-Domain Bit Allocation and Rate Control for Real Time Video Coding," *International Conference on Image Processing, 2001*, vol. 3, pp. 546–549, 2001.

[9] Z. Chen and K. N. Ngan, "Recent advances in rate control for video coding," *Signal Processing: Image Communication*, vol. 22, no. 1, pp. 19–38, January 2007.

[10] H. Yu, Z. Lin, and F. Pan, "An improved rate control algorithm for H.264," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 312–315, May 2005.

[11] J. Ribas-Corbera and S.-M. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 172–185, February 1999.

[12] ——, "A Frame-Layer Bit Allocation for H.263+," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1154–1158, October 2000.

[13] M. Jiang, X. Yi, and N. Ling, "Frame layer bit allocation scheme for constant quality video," *IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 1055–1058, June 2004.

[14] T. Lan and X. Gu, "H.264 Frame Layer Rate Control Based on Block Histogram Difference," *IEEE International Conference on Communications Workshops*, pp. 281–284, May 2008.

[15] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression.* Wiley, 2003, ISBN: 978-0-470-84837-1.

[16] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, February 2000.

[17] ETSI, *Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream*, November 2012, TS 101 154 V1.11.1.

[18] I. Richardson, "4x4 Transform and Quantization in H.264/AVC," *VCodex Ltd White Paper*, April 2009, http://www.vcodex.com/.