Scalable resource allocation for H.264 video encoder: Frame-level controller

Michael M. Bronstein
Technion – Israel Institute of Technology

September 7, 2009

Abstract

Tradeoff between different resources (bitrate, computational time, etc.) and compression quality, often referred to as rate-distortion optimization (RDO), is a key problem in video coding. With a little exaggeration, it can be claimed that what distinguishes between a good and a bad codec is how optimally it finds such a tradeoff. This is the second part of our paper presenting an algorithm for resource allocation in a scalable H.264 video codec. Here, we discuss a frame-level resource allocation and control scheme, capable of shaping the PSNR in the encoded frame. Coupled with a visual perception model, it allows to achieve perceptually-optimal quality within given resource constraints. The proposed region-wise resource allocation allows for scalable complexity.

1 Introduction

The goal of video compression, in simple words, is to deliver a video of as high quality as possible using as few bits and utilizing as few computations as possible. Since these desiderata are conflicting, tradeoff between different resources and compression quality is a key problem in video coding. Usually referred to as rate-distortion optimization (RDO), the problem is actually broader and in addition to bitrate can include additional resources such as computational time or power.
### Table 1: Notation and symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>frame type.</td>
</tr>
<tr>
<td>$i, k, j, l, n$</td>
<td>macroblock, region, slice, row, pixel index.</td>
</tr>
<tr>
<td>$N_{MB}$</td>
<td>number of macroblocks in a frame.</td>
</tr>
<tr>
<td>$N_S$</td>
<td>number of slices in a frame.</td>
</tr>
<tr>
<td>$N_{REG}$</td>
<td>number of regions in a frame.</td>
</tr>
<tr>
<td>$S_j,</td>
<td>S_j</td>
</tr>
<tr>
<td>$R_k,</td>
<td>R_k</td>
</tr>
<tr>
<td>$v_i, u_i$</td>
<td>texture, motion complexity in macroblock $i$.</td>
</tr>
<tr>
<td>$z_i^f$</td>
<td>frame-type dependent visual clue in macroblock $i$ in frame of type $f$.</td>
</tr>
<tr>
<td>$\alpha, \beta, \gamma$</td>
<td>coefficients of PSNR, bits, time complexity model.</td>
</tr>
<tr>
<td>$\alpha_i, \beta_i, \gamma_i$</td>
<td>coefficients in macroblock $i$ of PSNR; bits; time complexity model.</td>
</tr>
<tr>
<td>$\alpha_i^k, \beta_i^k, \gamma_i^k$</td>
<td>coefficients in region $k$ of PSNR; bits; time complexity model.</td>
</tr>
<tr>
<td>$q_i, q'_i$</td>
<td>quantization step; quantization parameter in macroblock $i$.</td>
</tr>
<tr>
<td>$p_i, \tilde{p}_i$</td>
<td>PSNR, predicted PSNR in macroblock $i$.</td>
</tr>
<tr>
<td>$b_i^f, \tilde{b}_i^f$</td>
<td>number of bits, predicted number of bits in macroblock $i$ in frame of type $f$.</td>
</tr>
<tr>
<td>$\tau_i^f, \tilde{\tau}_i^f$</td>
<td>encoding time, predicted encoding time in macroblock $i$ in frame of type $f$.</td>
</tr>
<tr>
<td>$\tau_i^{f'}, \tilde{\tau}_i^{f'}$</td>
<td>normalized encoding time, predicted normalized encoding time in macroblock $i$ in frame of type $f$.</td>
</tr>
<tr>
<td>$T_f$</td>
<td>nominal encoding time in frame of type $f$.</td>
</tr>
<tr>
<td>$b_T$</td>
<td>frame bit budget.</td>
</tr>
<tr>
<td>$b_{min}, b_{max}$</td>
<td>buffer safeguards.</td>
</tr>
</tbody>
</table>

dissipation [14, 13]. With a little exaggeration, it can be said that what distinguished a good video codec from a bad one is the mechanism performing such a tradeoff.

The target application we consider as a motivation for this paper is video distribution in a modern networked ecosystem of consumer electronic devices. A typical high-definition (HD) video arrives to our home from a cable or satellite broadcast, usually compressed using the MPEG-2 standard, at a bitrate up to 20 Mbps. After transcoding, this video source can then be distributed to multiple destinations: over the local wireless network to a local networked or portable device, or over the broadband to a remote Internet-connected location. The target bitrates in these two cases can vary dramatically: the bandwidth of a wireless network can typically range from 10 to 2 Mbps (whether using 801.11n or 801.11g standards), while the typical bandwidth of a broadband connection is 200 – 500 Kbps. To make the problem even more complicated, the available resource may change dynamically: the bandwidth may drop depending on the network utilization by other users and
applications, and the available computational resources may vary depending, for example, on the number of channels simultaneously encoded.

The described video distribution scenario imposes two important requirement on the encoder. First, there are multiple operating points in the resource space (corresponding to different bandwidth and computational complexity). The encoder must be able to work at any of these points, or in other words, be scalable. Second, since the resources may change in time, the encoder must be able to transition from one operating point to another as quickly as possible. Finally, given the available resources, the quality of the video should be as high as possible. All of the above make the resource allocation and control a challenging and non-trivial problem.

In this paper, we present an algorithm for resource allocation in a scalable video codec. Here, we discuss a frame-level resource allocation and control scheme, assuming a given sequence-level controller that allocates resources for a frame. As a reference, we use the H.264 standard, though the concepts are generic and can be applied to other MPEG-type encoding standards.

The rest of the paper is organized as follows. Section 2 presents a general background on video compression and the problem of resource allocation and control. In Section 3, we present a system-level view and the main features of the proposed resource allocation and control scheme, focusing on the frame-level controller. Section 4 describes the models of PSNR, bit production, encoding complexity, as well as their training and adaptation mechanisms. Section 5 describes the perceptual quality model, employed later in the PSNR shaping. In Section 6, we describe in details all the elements of the frame-level resource allocation and control algorithm. Finally, Section 7 concludes the paper and proposes possible extensions of the presented approach.

2 Background

2.1 General principles of video coding

Video compression takes advantage of these redundancies, trying to reduce the size of the video data while maintaining the content almost without visible loss. Such compression is referred to as lossy (as opposed to lossless), since the original data cannot be recovered exactly. Most modern codecs operate by constructing a prediction of the video frame by using the redundant information.
The prediction parameters are usually much more compactly represented than the actual pixels. The predicted frame is usually similar to the original one, yet, differences may appear at some pixels. For this purpose, the residual (prediction error) is transmitted together with the prediction parameters. The error is also usually compactly represented using spatial transform, like the discrete cosine transform (DCT) with subsequent quantization of the transform coefficients. The quantized coefficients and the prediction parameters are entropy-encoded to reduce furthermore their redundancy.

Inter-prediction accounts for the major percentage of compression of the video data size in video codecs. It takes advantage of temporal redundancy, predicting a frame from nearby frames (referred to as reference frames) by moving parts of these frames, thus trying to compensate for the motion of the objects in the scene. The description of the motion (motion vectors) are the parameters of the prediction which are transmitted. This type of prediction is used in MPEG-like codecs. The process of producing intra-prediction is called motion estimation. Typically, it is performed in a block-wise manner, when many modern codec support motion estimation with blocks of adaptive size.

At the other end, intra-prediction takes advantage of spatial redundancy and predicts portions of a frame (blocks) from neighbor blocks within the same frame. Such prediction is usually aware of spatial structures that may occur in a frame, namely smooth regions and edges. In the latter case, the prediction is directional. Such type of prediction is used in the recent H.264 Advances Video Codec (AVC) standard [?] and does not appear in previous versions of the MPEG standard.

The actual reduction in the amount of transmitted information is performed in the transmission of the residual. The residual frame is divided into blocks, each of which undergoes a DCT-like transform. The transform coefficients undergo quantization, usually performed by scaling and rounding. Quantization allows represent the coefficients by using less precision, thus reducing the amount of information required. Quantized transform coefficients are transmitted using entropy coding. This type of coding a lossless and utilizes further redundancy that may exist in the transform coefficients. The encoder usually operates with video represented in the YCbCr color space, where the achromatic Y channel is called the luma and the chromatic channels (Cb and Cr) are called chroma. The chroma is typically downsampled to half frame size in each direction, which is referred to as the 4 : 2 : 0 format.
The performance of a lossy video codec is measured as the tradeoff between the amount of bits required to describe the data and the distortion introduced by the compression, referred to as the rate-distortion (RD) curve. The RD curve of a better codec is closer to the idealistic and theoretically impossible case of zero distortion and zero rate. As the distortion criterion, the mean squared error (MSE) of the luma channel is usually used. The MSE is often converted into logarithmic units and represented as the peak signal-to-noise ratio (PSNR),

\[ p = 10 \log_{10} \left( \frac{y_{\text{max}}^2}{d} \right) \]  

(here \(d\) is the MSE and \(y_{\text{max}}\) is the maximum allowed value of the luma pixels, typically 255 if the luma data has an 8-bit precision and is represented in the range 0, ..., 255).

### 2.2 H.264 video codec

The H.264 AVC is one of the most recent standard in video compression, offering significantly better compression rate and quality compared to the previous MPEG-2 and MPEG-4 standards and targeted to high definition (HD) content. For example, H.264 delivers the same quality as MPEG-2 at a third to half the data rate.

The encoding process goes as follows: a frame undergoing encoding is divided into non-overlapping macroblocks, each containing 16 × 16 luma pixels and 8 × 8 chroma pixels (in the most widely used 4 : 2 : 0 format). Within each frame, macroblocks are arranged into slices, where a slice is a continuous raster scan of macroblocks. Each slice can be encoded independently of the other. Main slice types are P and I. An I slice may contain only I macroblocks; a P slice may contain P or I macroblocks. The macroblock type determines the way it is predicted. P refers to inter-predicted macroblocks; such macroblocks are sub-divided into smaller blocks. I refers to intra-predicted macroblocks; such macroblocks are divided into 4 × 4 blocks (the luma component is divided into 16 blocks; each chroma component is divided into 4 blocks). In I mode (intra-prediction), the prediction macroblock is formed from pixels in the neighbor blocks in the current slice that have been previously encoded, decoded and reconstructed, prior to applying the in-loop deblocking filter. The reconstructed macroblock is formed by imitating the decoder operation in the encoder loop. In P mode (inter-prediction), the prediction macroblock is formed by motion
compensation from reference frames. The prediction macroblock is subtracted from the current macroblock. The error undergoes transform, quantization and entropy coding. According to the length of the entropy code, the best prediction mode is selected (i.e., the choice between an I or a P macroblock, the motion vectors in case of a P macroblock and the prediction mode in case of an I macroblock). The encoded residual for the macroblock in the selected best mode is sent to the bitstream.

The operating point on the RD curve is controlled by the quantization parameter, determining the “aggressiveness” of the residual quantization and the resulting distortion. In the H.264 standard, the quantization parameter is an integer in the range $0, \ldots, 51$, denoted here by $q'$. The quantization step doubles for every increment of 6 in $q'$. Sometimes, it is more convenient to use the quantization step rather than the quantization parameter, computed according to

$$q = 0.85 \cdot 2^{q' - 12}.$$  

In the following, we will use $q$ and $q'$ interchangeably.

### 2.3 Resource allocation and control

Since only the decoder is standardized in the MPEG standards, many aspects of the way in which the encoder produces a standard-compliant stream is left to the discretion of a specific encoder design. The main difference between existing encoders is the decision process carried out by the bitrate controller that produces the encoder control parameters. Usually, the encoder parameters are selected in a way to achieve the best tradeoff between video quality and bitrate of the produced stream. Controllers of this type are referred to as RDO.

Parameters controlled by the bitrate controller typically include: frame type and reference frame or frames if the frame is a P-frame on the sequence level, and macroblock type and quantization parameter for each macroblock on the frame level. For this reason, it is natural and common to distinguish between two levels of bitrate control: sequence- and frame-level. The sequence-level controller is usually responsible for the frame type selection and allocation of the bit budget for the frame, and the frame-level controller is responsible for selection of the quantization parameter for each macroblock within the frame.
In a broader perspective, RD optimization is a particular case of the optimal resource allocation problem, in which the available resources may be the bitrate, computational time, power dissipation, etc. are distributed in a way that maximizes some quality criterion. Hence, hereinafter we use the term resource allocation referring to RD optimization-type problems discussed in this paper. The resources we consider specifically are computational time and bitrate.

Theoretically, optimal resource allocation requires running the encoding with different sets of parameters and selecting the best outcome. Such an approach is impossible due to a very large number of possible combinations of parameters, which result in a prohibitive computational complexity. Suboptimal resource allocation approaches usually try to model some typical behavior of the encoder as function of the parameters; if the model has an analytical expression which can be efficiently computed, the optimization problem becomes practically solvable. However, since the model is only an approximate behavior of the encoder, the parameters selected using it may be suboptimal.

3 System-level view

The presented resource allocation and control mechanism are generic and in principle can be used with any MPEG-type encoder. To be more specific, in the following we assume that the underlying encoding pipeline implements the H.264 AVC encoder, with independent slice encoders working simultaneously. Such an implementation is typical of real-time hardware encoders.

A system-level block diagram of the proposed controllers is shown in Figure 1. The system consists of the sequence-level controller, which is responsible for frame type and reference selection and allocation of the bit budget and initial quantization parameter for the currently encoded frame. The frame-level sequence controller allocates the resources on sub-frame level, in order to utilize the total frame bit budget and achieve the highest perceptual visual quality. For this purpose, a perceptual quality model and a PSNR shaping mechanism are employed. Both controllers are based on the encoder model, which predicts the behavior of the encoder given a set of encoding parameters. The model, in turn, utilizes visual clues, a set of descriptors of the video data.

Broadly speaking, if the two main criteria in the resource allocation problem are fitting into the resource constraints and achieving maximum quality, the sequence-level controller is responsible
for determining and meeting these constraints, and the frame-level controller is responsible for achieving the best quality within the constraints.

The main focus of this paper are the encoder models and the frame-level controller. Among the most important features of the proposed frame-level controller are:

**Content-aware encoder model:** a accurate encoder model based on visual clues, allowing to optimize the resource allocation prior to encoding;

**PSNR shaping:** a mechanism of resource allocation which optimizes the achieved distortion according to any given shape, and a perceptual quality model which is used to produce such a shape;

**Region-wise processing:** the allocation of resources within a frame is performed in a region-wise manner, which gives a scalable computational complexity of the controller;

**Time complexity model:** allows to predict and balance the computation load between slice encoders.

![System-level block diagram.](image)

Figure 1: System-level block diagram.
4 Encoder model

The purpose of encoder model is to estimate the amount of resources used by the encoder without resorting to the expensive process of encoding itself. Specifically, we predict the amount of bits produced by the encoder, the distortion as the result of encoding and the time complexity of the encoding process. We pose a few requirements on the encoder models. First, it must give an explicit expression to the predicted values (amount of bits, distortion, time) as function of encoder parameters ($q'$ or $q$) in order to be used in optimization. Second, the same predictors should be used by sequence- and the frame-level controllers, which implies that the predictors can be applied both on data units of different granularity (frame, region of macroblocks or macroblock). Third, the model should be content-aware, that is, take into consideration different data on which the encoder operates.

Here, we use linear predictors, which gives both explicit dependence on the encoder parameters and allows us using the predictors on data of any granularity. The optimal predictor coefficients are found off-line by means of supervised learning and are adapted during the encoding process. Most of the models employed here depend of the frame type (denoted by $f$), which can be either P or I. Finally, content dependence is captured by means of visual clues, a measure of content complexity, as described in the following.

4.1 Visual clues

Two type of clues are used by predictors in I-frames and P-frames. In I-frames, the visual clue is texture complexity $v_i$, measured as the standard deviation of luma pixels in each macroblock,

$$v_i = \left( \frac{1}{16^2 - 1} \sum_{n \in \mathcal{M}_i} \left( y_n - \frac{1}{16^2} \sum_{n \in \mathcal{M}_i} y_n \right)^2 \right)^{\frac{1}{2}}, \quad (3)$$

where $y_n$ denotes the $n$th luma pixel, and $\mathcal{M}_i$ is the set of pixel indices belonging to macroblock $i$ in the current frame. The intuition behind this measure is that texture in most cases is responsible for intra-prediction errors, which in turns influences the amount of bits produced, the PSNR and the encoding time.
In P-frames, we need to take into consideration the difficulty of motion compensated prediction. Let \( y \) denote the luma component of the current frame, and \( y' \) be the luma component of the reference frame selected by the sequence-level controller. The motion difference, computed as the spatially-smoothed absolute difference of luma pixels in collocated macroblocks in current frame and the reference frame,

\[
m_i = \sum_{n \in M_i} |\sigma(y_n - y'_n)|,
\]

is used as the measure of predictability and consequently, of inter-prediction difficulty. Here, \( \sigma \) denotes spatial smoothing operation by means of a Gaussian filter. We use the motion complexity,

\[
u_i = \sqrt{m_i t_i}.
\]

as the visual clue for P-frames.

4.2 Bit production model

The purpose of the bit production model is to predict the amount of bits produced by the encoder while encoding a data unit, the lowest granularity being a macroblock. Bit production models used in literature assume that the number of bits required to encode a macroblock is inversely proportional to the quantization step in this macroblock or its square (\( b_i \propto q_i^{-1} \) or \( b_i \propto q_i^{-2} \)), the latter referred to as the quadratic model [3]. Such models do not take into consideration the content of the macroblock and consequently produce relatively large errors. More advanced models [11, 17] try to employ the dependency of \( b_i \) on the residual image produced as the result of motion estimation, expressed as mean absolute difference (MAD). The disadvantage of such approach is that MAD is computed during motion estimation, which means that the prediction of the amounts of bits is obtained during the encoding process and not beforehand, which is not suitable for our controller scheme. As a result, it is impossible to perform optimal pre-allocation of resources. Alternatively, a pre-coding scheme can be used, in which the MAD is produced before the encoding, usually by applying the encoding process to a lower resolution frame. Such an approach is implemented in some software H.264 encoders, for example, the X.264. Pre-coding
is usually computationally intensive and its use in real-time hardware codecs is disadvantageous.

We use a model based on the visual clues pre-computed prior to encoding. A distinction is made between I-frames and P-frames. For I-frames, the amount of bits in a macroblock is proportional to $q_i^{-1}$ and the texture complexity $v_i$. For P-frames, the amount of bits is proportional to $q_i^{-1}$ and the motion complexity $u_i$. To simplify the notation, we denote frame-type dependent visual clues by

$$z^f_i = \begin{cases} u_i & \text{if } f \text{ is } P, \\ v_i & \text{if } f \text{ is } I, \end{cases}$$

referring to them simply as “visual clues”. In the following, we will omit the index $f$ wherever possible in order to simplify the notation. Thus, the predicted amount of bits required to encode macroblock $i$ in frame of type $f$ is given by

$$\hat{b}_f(z^f_i, q_i; \alpha_f) = \max\{\alpha_{1f}^f + \alpha_{2f}^f z^f_i q_i^{-1}, 0\},$$

(7)

where $\alpha_f = (\alpha_{1f}^f, \alpha_{2f}^f)^T$ are non-negative frame-type dependent bit production model coefficients, constant for each macroblock. Denoting $\bar{\alpha}_f = (\alpha_{1f}^f, \alpha_{2f}^f z_i^f)^T$, we obtain the following expression for the predicted bits in macroblock $i$,

$$\hat{b}_f(q_i; \bar{\alpha}_i) = \bar{\alpha}_{1i}^f + \bar{\alpha}_{2i}^f q_i^{-1},$$

(8)

which is linear in $q_i^{-1}$ and has frame-type and macroblock-dependent coefficients.

### 4.3 Distortion model

The purpose of the distortion model is to predict the objective distortion (luma PSNR) of the video data as the result of encoding. Empirical evidence shows that PSNR in a macroblock undergoing encoding process is approximately inversely proportional to the quantization parameter in this macroblock ($p_i \propto -q_i$). This model, however, does not take into consideration the content of the macroblock. We use an extended PSNR model, in which PSNR is also inversely proportional to the texture complexity ($p_i \propto -v_i$), which concords to the intuition that rich texture undergoes
more significant quality degradation as the result of encoding). Introducing such a dependency on texture complexity allows to improve the PSNR prediction accuracy by over 50% compared to a model dependent only on \( q'_i \).

According to our model, the predicted PSNR value in macroblock \( i \) is given independently of the frame type by

\[
\hat{p}_i(v_i, q'_i; \beta) = \beta_1 - \beta_2 v_i - \beta_3 q'_i,
\]

where \( \beta = (\beta_1, \beta_2, \beta_3)^T \) are non-negative PSNR model coefficients, constant for each macroblock. Denoting \( \bar{\beta}_i = (\beta_1 - \beta_2 v_i, -\beta_3)^T \), we obtain the following expression for the predicted PSNR,

\[
\hat{p}_i(q'_i; \bar{\beta}_i) = \bar{\beta}_{i1} + \bar{\beta}_{i2} q'_i,
\]

which is linear in \( q'_i \) and has macroblock-dependent coefficients.

### 4.4 Time complexity model

The purpose of the time complexity model is to estimate the time it takes to encode a data unit. The underlying assumption of our time complexity model is that the encoding pipeline used to encode a macroblock consists of multiple stages, some of which are bypassed according to the encoded macroblock type, using some decision tree (see two examples of such decision trees in Figure 2). Consequently, it can be simplistically assumed that encoding a macroblock of certain type takes a fixed amount of time. In such a formulation, prediction of the encoding time stems straightforwardly from prediction of the macroblock type, which has been shown to be possible with a relatively high accuracy (see, for example mode, recent works on fast mode selection [12, 18, 15, 16, 9]).

We denote the maximum amount of time it takes to encode a macroblock of type \( f \) by \( T_f \) and refer to it as the \textit{nominal encoding time}. The nominal time can be given in CPU cycles or seconds. For macroblock \( i \) in a frame of type \( f \), we predict the \textit{normalized encoding time}, \( \tau'^{if}_i = \tau^{if}_i T^{-1}_f \).

According to our model, the predicted normalized time for encoding a macroblock in I-frame is
given by

\[ \hat{\tau}_i^f(z_i^f, q_i^f; \gamma^f) = \max\{\min\{\gamma_1^f + \gamma_2^f z_i^f - \gamma_3^f q_i^f, 1\}, 0\} \],

(11)

where \( \gamma^f = (\gamma_1^f, \gamma_2^f, \gamma_3^f)^T \) are the time complexity model parameters, constant for each macroblock. The minimum and maximum is taken in order to ensure that the predicted normalized time value is within a valid range of \([0, 1]\). This model concords with the intuition that macroblocks with higher texture or motion complexity will usually take more time to encode (for example, they are more likely to be encoded as I4 \times 4 macroblock) and that for higher values of quantization parameter the increasing probability of a macroblock to be encoded as skip reduces the time complexity. Denoting \( \bar{\gamma}_i^f = (\gamma_1^f + \gamma_2^f z_i^f, -\gamma_3^f)^T \), we obtain the following expression for the predicted normalized time,

\[ \hat{\tau}_i^f(q_i^f; \bar{\gamma}_i^f) = \max\{\min\{\bar{\gamma}_{i1}^f + \bar{\gamma}_{i2}^f q_i^f, 1\}, 0\} \],

(12)

which (assuming a reasonable range of \( q_i \) and \( z_i^f \)) is linear in \( q_i^f \) and has macroblock-dependent coefficients. The predicted absolute encoding time in a macroblock is given by \( \hat{\tau}_i^f = T_f \cdot \hat{\tau}_i^f \).

Figure 2: Typical decision tree in encoding for a macroblock in P-frame (left) and I-frame (right) that can be employed for encoding time complexity model.
4.5 Training

The initial parameters $\alpha$, $\beta$ and $\gamma$ of the encoder models are computed by offline supervised learning. These parameters may vary across different encoders and even across different profiles of the same encoder. For the purpose of the following discussion, we consider the training of PSNR predictor; other predictors are trained in the same manner. We assume to be given a large statistics of $N_{\text{TRAIN}}$ encoded macroblocks, for which the values of the visual clues, quantization parameter and resulting PSNR are known. These values are represented as vectors $v$, $q'$ and $p$ of size $N_{\text{TRAIN}} \times 1$ each. The optimal vector of parameters is found as the solution to the following minimization problem,

$$\beta = \arg\min_{\beta \in \mathbb{R}^3_+} \|A\beta - p\|,$$

where $A = (1, -v, -q)$ is an $N_{\text{T}} \times 3$ matrix. We require the coefficients to be non-negative in order to make the model express the correlations with the encoder parameters and the visual clues in a correct way. In practice, it is advantageous to perform the training not on a training set with macroblock-wise data, but rather on row-wise or frame-wise data. Since the model is linear, such a training set can be produced by averaging macroblock-wise data.

If the $L_2$-norm is used, problem (13) is a non-negative least squares (NNLS) fitting problem, and $\beta$ is given as the solution of the constrained normal equations,

$$(A^TA)\beta = A^Tp, \quad \text{s.t. } \beta \geq 0.$$  \hspace{1cm} (14)

A more generic case is the weighted $L_2$-norm,

$$\beta = \arg\min_{\beta \in \mathbb{R}^3_+} \sum_{m=1}^{N_{\text{T}}} w_i |(A\beta)_i - p_i|^2$$

$$= \|W^{1/2}A\beta - p\|_2^2,$$

$$= \|W^{1/2}A\beta - p\|_2^2,$$
where

\[
W = \begin{pmatrix}
w_1 \\
\vdots \\
w_{N_T}
\end{pmatrix}
\]  

(16)

is an \(N_T \times N_T\) diagonal matrix of non-negative weights. The normal equations in this case assume the form

\[
(A^TWA)\beta = A^TW^\frac{1}{2}p, \quad \text{s.t.} \quad \beta \geq 0.
\]

Problems (14) and (17) are solved iteratively [10, 1].

The main disadvantage of the \(L_2\)-norm is that it does not reject outliers and produces an average fit, which is inaccurate if the training set has many outliers. For this reason, robust norms with the property of outlier rejection (such as the \(L_1\)-norm) produce better results. A commonly used technique for the solution of such problems is called \textit{iteratively reweighted least squares} (IRLS). The idea of the algorithm is solve a weighted least squares problem, adjusting the weights in a way to mitigate the influence of outliers [5].

\subsection*{4.6 Adaptation}

Since the initial model parameters represent an average on a large number of different types of video content (the training set), they may not fit well the specific data currently being encoded. In order to adjust the model parameters, an \textit{adaptation} or \textit{online learning} mechanism is employed. Using this approach, the model parameters are updated after encoding a certain amount of data (usually a row of macroblock or a frame), in accordance to the actual values obtained during the encoding process.

Taking again the PSNR predictor example, we assume that the weighted \(L_2\)-norm is used for training, with weights forming a geometric progression \(w_i = \lambda^{N_T-i}\) for some \(0 < \lambda < 1\), such that “more recent” samples have larger weight. The optimal model parameters are found by solving (17). Now assume that we wish to add the \((N_T + 1)\)st sample with quantization parameter...
$q'$, texture complexity $v$ and PSNR $p$ to the training set and see how the vector of coefficients $\beta$ changes as the result. The normal equations (17) become

$$
\left( \begin{array}{c} A \\ (1, -v, -q') \end{array} \right) \left( \begin{array}{cc} \lambda W & 1 \\ 1 & 1 \end{array} \right) \left( \begin{array}{c} A \\ (1, -v, -q') \end{array} \right) = \left( \begin{array}{c} p \end{array} \right), \text{ s.t. } \beta \geq 0,
$$

which leads us to the following adaptation formula,

$$
\left( \begin{array}{c} \lambda A^T A + \left( \begin{array}{ccc} 1 & -v & -q' \\ -v & v^2 & vq' \\ -q' & vq' & q'^2 \end{array} \right) \end{array} \right) \beta = \sqrt{\lambda} A^T p + \left( \begin{array}{c} p \\ -vp \\ -q'p \end{array} \right), \text{ s.t. } \beta \geq 0.
$$

Our conclusion is that we do not need to carry the entire $N_T \times 3$ matrix $A$ throughout the adaptation process; instead, we can use the $3 \times 3$ matrix $A^T A$ and the $3 \times 1$ vector $A^T p$.

Since the $L_2$-norm is sensitive to outliers, adaptation may produce inappropriate model parameters. In order to avoid this problem, an outlier rejection algorithm should be employed. The simplest way to do it is to compare the actual PSNR value to one predicted using the current model parameters, and not to perform the adaptation if the discrepancy between these values is above some threshold.

## 5 Perceptual quality model

The purpose of the perceptual quality model is to quantify the distortion of the video after encoding as perceived by the human observer. Unlike PSNR, visual distortion is a subjective criterion, and as a result, may vary across different people. It is however well-known from psychophysical research that in images with certain characteristics the distortion is less noticeable than in other for most of human observers. In image compression [7, 8] and watermarking, the notion of just noticeable distortion (JND) defining the threshold below which encoding artifacts are unnoticeable to the human eye was introduced in early 90's. Due to the tradeoff between bitrate and distortion, it is
usually hard or even impossible to achieve JND during video encoding. However, the perceptual model can be still used in order to achieve the minimally noticeable distortion (MND), such that the encoding artifacts are minimally visible in the image that has undergone encoding, and are uniformly distributed across the image.

Here, we use the perceptual quality model to drive the resource allocation, particularly, decide in which macroblocks more bits must be allocated such that a constant visual distortion is achieved. Assume that the JND for macroblock $i$, expressed as the mean squared error (MSE), is given by $d_0^i$. This implies that if the MSE $d_i$ in macroblock $i$ is below $d_0^i$, the distortion will be unnoticed. Since in practice, $d_i > d_0^i$, we want the distortion to be uniform across the frame, i.e. have $d_i = \nu \cdot d_0^i$, where $\nu$ is some factor constant for the entire frame. We can thus say, for example, that in a certain frame the distortion is twice the JND. Ideally, $\nu$ should be as small as possible.

Using the transformation (1), we obtain

$$10 \log_{10} \left( \frac{y_{\text{max}}^2}{\nu \cdot d_0^i} \right) = 10 \log_{10} \left( \frac{y_{\text{max}}^2}{\nu} \right) + 10 \log_{10} \left( \frac{y_{\text{max}}^2}{d_0^i} \right)$$

$$= p_0 + \Delta p. \quad (20)$$

We term $p_0$ PSNR offset and $\Delta p$ PSNR shape. In order to produce a minimally noticeable distortion, the encoder should achieve PSNR as close as possible to $p_0 + \Delta p_i$ in each macroblock $i$, where $p_0$ should be as large as possible.

## 5.1 Just noticeable distortion model

There exist numerous models for JND in literature. The common trait of such models is sensitivity to local contrast of the frame and local directional frequencies. Typically, the response of the human eye is modeled as a band-pass filter, where the highest sensitivity (and consequently, the lowest JND) is in the mid-range of contrast and frequency. Specifically, here we use the model described in [4].
6 Frame-level controller

The first and the main purpose of the frame-level controller is to ensure that the bit budget allocated to the frame by the sequence-level controller is utilized as precisely as possible. The bit budget allocated by the sequence-level controller assumes a constant quantization parameter in all the macroblocks of the frame. Hence, the second purpose of the frame-level controller is to refine this allocation. The refinement criterion is the proximity to the desired PSNR shape obtained from the visual model. The quantization parameter in each macroblock is selected in such a way that as the result, the number of bits equals the frame bit budget $b_T$ allocated by the sequence-level controller, and the predicted PSNR in macroblock $i$ is as close as possible to the desired shape $p_0 + \Delta p_i$. We refer to this problem as to PSNR shaping.

A block-diagram of the frame-level controller is shown in Figure 3. It consists of the following main blocks:

**Division into regions:** macroblocks with similar properties are grouped together into regions, in order to reduce the computational cost of PSNR shaping;

**Region-wise PSNR shaping:** selection of the quantization parameter in each region;

**Slice load balancing:** the frame is divided into slices according to the computational complexity model, such that the processing time for each slice is approximately equal;

**Closed-loop bitrate controller:** a controller coupled to each slice encoder and making sure that the slice bit budget is used as precisely as possible.

6.1 Division into regions

Ideally, the selection of the quantization parameter should be done in a macroblock-wise manner. However, such an allocation is usually expensive, since the number of macroblocks is large (for example, a full HD frame contains 8160 macroblocks). If we perform macroblock-wise selection of the quantization parameter, it will typically appear that its values are region-wise constant: in regions with similar properties the value of the quantization parameter is the same. If it was
possible to identify such regions in advance, we could significantly reduce the number of variables in our PSNR shaping problem.

Let us consider the selection of the optimal quantization parameter in a macroblock. Since the bit budget is a constraint on the total amount of bits required to encode the frame, it is reasonable to assume that changing the quantization parameter in a macroblock approximately does not violate the bit budget constraint. In the PSNR shaping problem, we try to select such a value of $q'_i$ in a macroblock $i$ that the predicted PSNR $\hat{p}_i$ is as close as possible to the target shape $p_0 + \Delta p_i$. In turn, according to our distortion model, PSNR is dependent of the texture complexity. Thus, neglecting for a moment the bit budget constraints, the locally optimal selection of the quantization parameter is $q'_i = \arg\min_{q'} |\hat{p}_i(q' ; v_i) - (p_0 + \Delta p_i)|$. Note that $q'_i$ in this expression depends only on $v_i$ and $\Delta p_i$. We can therefore conclude that in macroblocks with similar values of $v_i$ and $\Delta p_i$ the value of $q'_i$ will be similar.

The selection of regions is performed by means of vector quantization or clustering (see Figure 4). The values $x_i = (\Delta p_i, v_i)$ for all the macroblocks $i = 1, \ldots, N_T$ in the frame are clustered in $\mathbb{R}^2$, which gives the labels of the regions to which they belong. An example of such a labelling is shown in Figure 6. The clustering is performed by minimizing the intra-cluster variance,

$$\{\mathcal{R}_1, \ldots, \mathcal{R}_{N_{REG}}\} = \arg\min_{\{\mathcal{R}_1, \ldots, \mathcal{R}_{N_{REG}}\}} \sum_{k=1}^{N_{REG}} \sum_{i \in \mathcal{R}_k} |x_i - c_k|^2,$$

(21)

where $c_k$ is the centroid of the points corresponding to region $\mathcal{R}_k$. A commonly used iterative heuristic to solve the clustering problem (21) is the Max-Lloyd algorithm [6], which goes as follows:
**initialization:** Number of regions $N_{\text{REG}}$, initial partitioning of the points $\{x_1, \ldots, x_{N_{MB}}\}$ into $N_{\text{REG}}$ clusters.

1. **repeat**
   2. Compute the centroids $c_k = \frac{1}{|R_k|} \sum_{i \in R_k} x_i$ of each cluster $R_k$.
   3. Construct new clusters by associating each point with the closest centroid,
      $$R_k = \{i : \|x_i - c_k\|_2 \leq \|x_i - c_{k'}\|_2 \quad \forall k' \neq k\},$$  \hspace{1cm} (22)
4. **until convergence**

**Algorithm 1:** Max-Lloyd clustering algorithm.

A suboptimal yet computationally more efficient way of solving the clustering problem (21) is dividing the $(\Delta p, v)$ plane into fixed regions, as shown in Figure 5 (right). Such a procedure is not iterative and in most cases produces regions similar to those computed using vector quantization.

### 6.2 Region-wise encoder models

The amount of bits required to encode the macroblocks in region $k$ is given by

$$\hat{b}_k(q_k; \bar{\alpha}_k) = \sum_{i \in R_k} \hat{b}_i(q_k; \bar{\alpha}_i)$$  \hspace{1cm} (23)

$$= \sum_{i \in R_k} \bar{\alpha}_{i1} + \bar{\alpha}_{i2}q_{k}^{-1}$$

$$= \bar{\alpha}_{k1}^r + \bar{\alpha}_{k2}^rq_{k}^{-1},$$

where $\alpha^r_k = \sum_{i \in R_k} \alpha_i$ is used to denote the coefficients of the region-wise model. In the following, we consistently use the index $k$ referring to regions and the index $i$ referring to macroblocks, in order to avoid cumbersome notation. Thus, the region-wise bit production model is linear. In
the same way, the predicted average PSNR in the region $k$ is given by

$$
\hat{p}_k(q'_k; \bar{\beta}_k) = \frac{1}{|R_k|} \sum_{i \in R_k} \hat{p}_i(q'_k; \bar{\beta}_i)
$$

(24)

where $\bar{\beta}_k = \frac{1}{|R_k|} \sum_{i \in R_k} \beta_i$. Note that the average PSNR is computed by averaging the PSNR values rather than averaging the corresponding MSE values and then taking the logarithmic transformation. Even though less accurate, we still prefer this way since it leads to a linear model.

### 6.3 Region-wise PSNR shaping

After the frame is segmented into regions, the PSNR shaping problem is solved in a region-wise manner, assigning a constant quantization parameter to each region. We assume to be given some initial value $q'_0$ of the quantization parameter allocated by the sequence-level controller for the entire frame and the bit budget $b_T$. The vector of the region-wise values of the quantization parameter are denoted by $q^r = (q^r_1, ..., q^r_{N_{\text{REG}}})$. Instead of working with the quantization parameter $q'$, we work with quantization parameter difference $\Delta q^r = q' - 1 \cdot q'_0$. We demand the values of the quantization parameter difference to vary in the hypercube $[\Delta q^r_{\text{min}}, \Delta q^r_{\text{max}}]$, which is typically chosen as $[-2, 2]^{N_{\text{REG}}}$ or $[-3, 3]^{N_{\text{REG}}}$.

The problem of PSNR shaping is formulated as follows:

$$
\min_{\Delta q^r \in \mathbb{R}^{N_{\text{REG}}}, p_0} \|\hat{p}^r(\Delta q^r) - (0 + \Delta p^r)\| - p_0 \quad \text{s.t.} \quad 1^T \hat{b}^r(\Delta q^r) = b_T
$$

(25)

\[\Delta q^r_{\text{min}} \leq \Delta q^r \leq \Delta q^r_{\text{max}}\]

where $\Delta q^r = (\Delta q^r_1, ..., \Delta q^r_{N_{\text{REG}}})^T$, $\hat{b}^r = (\hat{b}^r_1, ..., \hat{b}^r_{N_{\text{REG}}})^T$ and $\hat{p}^r = (\hat{p}^r_1, ..., \hat{p}^r_{N_{\text{REG}}})^T$ are the vectors of quantization parameter difference, predicted number of bits and predicted PSNR in the regions,
and

\[ \Delta p^r = (\Delta p^r_1, \ldots, \Delta p^r_{N_{REG}}), \quad \Delta p^r_k = \frac{1}{|R_k|} \sum_{i \in R_k} \Delta p_i \]  

(26)

is the region-wise average of the PSNR shape.

The PSNR predictor in region \( k \), expressed as a function of \( \Delta q'^r_k \), is given as the following expression,

\[
\hat{p}^r_k(\Delta q'^r_k; \tilde{\beta}_k) = \bar{\beta}^r_{k1} + \bar{\beta}^r_{k2}(q'^r_k - q'_0 + q'_0) \\
= (\bar{\beta}^r_{k1} + \bar{\beta}^r_{k2}q'_0) + \bar{\beta}^r_{k2}\Delta q'^r_k \\
= \tilde{\beta}^r_{k1} + \tilde{\beta}^r_{k2}\Delta q'^r_k,
\]

where \( \tilde{\beta}^r_k = (\bar{\beta}^r_{k1} + \bar{\beta}^r_{k2}q'_0, \bar{\beta}^r_{k2})^T \). On the other hand, the predictor of the amount of bits is linear in \( q^{-1} \) but non-linear in \( \Delta q' \). In order to simplify expressions, we linearize \( q^{-1}(q') \) around \( q'_0 \),

\[
q^{-1}(q'_0 + \Delta q') \approx \frac{1}{0.85} \cdot 2^{-\frac{\mu_{-12}}{6}} \left( 1 - \frac{\log 2}{6} \Delta q' \right) \]  

(28)

and substitute it to the predictor, obtaining

\[
\hat{b}^r_k(\Delta q'^r_k; \tilde{\alpha}^r_k) = \max\{\tilde{\alpha}^r_{k1} + \tilde{\alpha}^r_{k2}(q'^r_k - q'_0) \bigg| 0 \} \\
\approx \max \left\{ \left( \tilde{\alpha}^r_{k1} + \frac{\tilde{\alpha}^r_{k2}}{0.85} \cdot 2^{-\frac{\mu_{-12}}{6}} \right) - \frac{\log 2}{5.1} \cdot \tilde{\alpha}^r_{k2} \cdot 2^{-\frac{\mu_{-12}}{6}} \Delta q'^r_k, 0 \right\} \]  

(29)

where \( \tilde{\alpha}^r_k = \left( \tilde{\alpha}^r_{k1}, \frac{\tilde{\alpha}^r_{k2}}{0.85} \cdot 2^{-\frac{\mu_{-12}}{6}}, -\frac{\log 2 \cdot \tilde{\alpha}^r_{k2}}{5.1} \cdot 2^{-\frac{\mu_{-12}}{6}} \right)^T \).

Assuming that the weighted \( L_1 \)-norm is used in problem (26) and plugging the above expres-
sions, we get

\[
\min_{\Delta q^r \in \mathbb{R}^{N_{\text{REG}}}, p_0} \sum_{k=1}^{N_{\text{REG}}} w_k |\tilde{\beta}_{k1}^r + \tilde{\beta}_{k2}^r \Delta q^r_k - (p_0 + \Delta p^r_k)| - p_0 \quad \text{s.t.} \quad \sum_{k=1}^{N_{\text{REG}}} \tilde{\alpha}_{k1}^r + \tilde{\alpha}_{k2}^r \Delta q^r_k = b \quad (30)
\]

\[
\Delta q^r_{\min} \leq \Delta q^r \leq \Delta q^r_{\max},
\]

where \( w = (w_1, ..., w_{N_{\text{REG}}})^T \) is a vector of non-negative weights proportional to the region size \( (w_k \propto |\mathcal{R}_k|) \).

By introducing artificial variables, problem (30) can be posed as linear programming (LP) problem with \( 2N_{\text{REG}} + 1 \) variables, one equality constraint and \( 2N_{\text{REG}} + 1 \) inequality constraints, as follows [1],

\[
\min_{x \in \mathbb{R}^{2N_{\text{REG}}+1}} c^T x \quad \text{s.t.} \quad a_{\text{eq}}^T x = b_{\text{eq}} - \sum_{k=1}^{N_{\text{REG}}} \tilde{\alpha}_{k1}^r \quad (31)
\]

\[
A_{\text{neq}} x \leq b_{\text{neq}}
\]

where \( z \in \mathbb{R}^{N_{\text{REG}}} \) is a vector of artificial variables,

\[
x = \begin{pmatrix} \Delta q^r \\ p_0 \\ z \end{pmatrix} ; \quad c = \begin{pmatrix} 0 \\ -1 \\ w \end{pmatrix} ;
\]
and

\[
A_{\text{neq}} = \begin{pmatrix}
\tilde{\beta}_{12} & \ldots & -1 & -1 \\
\vdots & \ddots & \vdots & \vdots \\
\tilde{\beta}_{N_{\text{REG}},2} & 1 & -1 \\
1 & \ldots & 1 \\
-1 & \ldots & -1 \\
\end{pmatrix}
\]

and

\[
b_{\text{neq}} = \begin{pmatrix}
\Delta p_{1} - \tilde{\beta}_{11} \\
\vdots \\
\Delta p_{N_{\text{REG}}} - \tilde{\beta}_{N_{\text{REG}},1} \\
-\Delta p_{1} + \tilde{\beta}_{11} \\
\vdots \\
-\Delta p_{N_{\text{REG}}} + \tilde{\beta}_{N_{\text{REG}},1} \\
\Delta q_{\text{max}} \\
-\Delta q_{\text{min}} \\
\end{pmatrix}
\]

\[
A_{\text{equiv}} = \begin{pmatrix}
\tilde{\alpha}_{12} \\
\vdots \\
\tilde{\alpha}_{N_{\text{REG}},2} \\
0 \\
0 \\
\end{pmatrix}
\]

The solution of problem (31) gives us the region-wise map of quantization parameter differences, \( \Delta q_{r} \), from which the quantization parameter \( q_{r} = 1 \cdot q_{0} + \Delta q_{r} \) is computed. Since we used linearization of the bit predictor, a discrepancy in the predicted amount of bits may occur. In order to avoid such errors, we use an iterative linearization scheme: once the value of \( q_{r} \) is computed, the linearization of \( \hat{b}_{k} \) in region \( k \) is performed around \( q_{k}^{*} \) rather than around \( q_{0}^{*} \). The process is repeated until the quantization parameter stops changing significantly. The entire scheme can be summarized as follows:
**initialization:** Initial quantization parameter in each region $q'^r = q'_0 \cdot 1$; some quantization parameter difference interval $[\Delta q'^r_{\text{min}}, \Delta q'^r_{\text{max}}] = [-\Delta q'_0, \Delta q'_0]^{|N_{\text{REG}}|}$.

1. repeat
2. Compute the parameters $\tilde{\alpha}^r_1, \ldots, \tilde{\alpha}^r_{N_{\text{REG}}}$ by linearizing the bits predictor in each region around $q'^r_k$ according to (29).
3. Compute the quantization parameter differences $\Delta q'^r$ by solving the LP problem (31) for $\Delta q'^r$ in the range $[\Delta q'^r_{\text{min}}, \Delta q'^r_{\text{max}}]$.
4. Update the quantization parameter difference range $\Delta q'^r_{\text{min}} \leftarrow \Delta q'^r_{\text{min}} - \Delta q'^r$ and $\Delta q'^r_{\text{max}} \leftarrow \Delta q'^r_{\text{max}} - \Delta q'^r$.
5. Set $q'^r \leftarrow q'^r + \Delta q'^r$.
6. until $\|\Delta q'^r\| \leq \epsilon$

**Algorithm 2:** PSNR shaping with iterative linearization of the bits predictor.

Note that the values $q'^r_k$ produced in this way can be non-integer, while in practice the encoder works only with integer quantization parameters. Hence, there is a need to round the obtained result to the closest integer (see example in Figure 7).

Denoting by $q'_0 = 0.85 \cdot 2^{\frac{q'_0}{12}}$ the quantization step corresponding to the quantization parameter $q'_0$, we can translate the quantization parameter difference $\Delta q'^r_k$ into a multiplicative factor,

$$\mu^r_k = \frac{q'^r_k}{q'_0} = 2^{\frac{q'^r_k - q'_0}{12}}, \quad (32)$$

referred here to as the *quantization step multiplier*, such that $q'^r_k = q'_0 \cdot \mu^r_k$.

### 6.4 Division into slices

One of the basic methods for parallelization of MPEG-type encoders is done by dividing the frame into slices, which can be encoded and decoded independently of each other. Another advantage of using slices is error resilience: if a slice is lost, the rest of the frame may still be recovered. Most encoders divide the frame into fixed-size slices, which are then undergo independent encoding. However, such an approach does not account for the different encoding complexity of the slice content, as a result of which the load balancing of the slice encoder can be unequal.
In the proposed controller, we use the time complexity model in order to perform load balancing across slices. Given the desired number of slices, \( N_S \), we define the slices in order to satisfy

\[
\sum_{i \in S_j} \tilde{\tau}_i f_i \approx \sum_{i \in S_{j'}} \tilde{\tau}_i f_i \tag{33}
\]

for all \( j, j' = 1, \ldots, N_S \), where \( S_j \) is a continuous set of indices of macroblocks in slice \( j \). Typically, a slice consists of a number of full rows of macroblocks, which is assumed hereinafter for simplicity. The slice boundaries are therefore row numbers \( n_0 = 1 \leq n_1 \leq \ldots \leq n_{N_S-1} \leq n_{N_S} = N_{ROW} \), such that \( S_j = \{ (n_j-1)N_{COL} + 1, \ldots, n_j \cdot N_{COL} \} \).

Denoting by

\[
t_l = \frac{\sum_{i=1}^{l \cdot N_{COL}} \tilde{\tau}_i f_i}{\sum_{i=1}^{N_{MB}} \tilde{\tau}_i f_i} \tag{34}
\]

the normalized predicted complexity of encoding rows \( 1, \ldots, l \), we can reformulate condition (33) by requiring \( t_{n_j} \approx j \cdot N_S^{-1} \) (see Figure 8). This gives us the following division,

\[
n_j = \arg \min_{l=1, \ldots, N_{ROW}} \left| t_l - j \cdot N_S^{-1} \right| \tag{35}
\]

Note that division into equal slices is achieved as a particular case of (35) by selecting \( t_l = l \cdot N_{ROW}^{-1} \) (equal complexity for each row), which yields \( n_j = \left[ j \cdot N_{ROW}^{-1} \right] \). An example of load balancing is shown in Figure 9.

As a precaution meant to avoid abnormal slice sizes (too small or too large slices are equally bad), the difference between \( n_j \) computed according to (35) and \( \left[ j \cdot N_{ROW}^{-1} \right] \) as if it were in case of fixed-size slice can bounded. The described approach can be extended by taking into consideration not only time complexity but also the predicted number of bits, thus trying to obtain slices with balanced computational complexity and bit budget, the latter being important for the purpose of error resilience.
6.5 Closed-loop bitrate controller

Since the predicted values may differ from the actual one produced by the encoder, it is not guaranteed that performing encoding with the allocated quantization parameter will produce an amount of bits meeting the bit budget. The purpose of the bitrate controller is to adjust the quantization parameter in order to minimize this discrepancy.

In our implementation, closed-loop bitrate control is performed independently for each slice, after encoding a data unit consisting of $M_{\text{UNIT}}$ macroblocks (typically a row of macroblocks, i.e., $M_{\text{UNIT}} = N_{\text{COL}}$ which is assumed hereinafter). To simplify the following discussion and avoid cumbersome notation, we assume that there is a single slice in the frame. The available bit budget in the slice is $b_T$ and the region-wise quantization parameter $q_0^r + \Delta q_k^r$ (or alternatively, $q_0\mu_k$) is allocated by the PSNR shaping algorithm in such a way that the predicted amount of bits in the slice equals the budget, i.e.,

$$\sum_{k=1}^{N_{\text{REG}}} \bar{\alpha}_{k1} + \frac{\bar{\alpha}_{k2}}{q_0\mu_k} = b_T,$$  \hspace{1cm} (36)

Assume that the first row of macroblocks in encoded, producing $b_E = \sum_{i=1}^{N_{\text{UNIT}}} b_i$ bits. Since the predicted amount of bits does not necessarily equal the actual one (i.e., $\sum_{i=1}^{N_{\text{UNIT}}} \hat{b}_i \neq b_E$), the predicted amount of bits for the remaining part of the slice may not fit the budget anymore.

In order to fit the remaining budget, the quantization parameter in the remaining part of the slice must be modified. This is performed by adjusting $q_0$. We demand that $\sum_{i=M_{\text{UNIT}}+1}^{N_{\text{MB}}} \hat{b}_i = b_T - b_E$, which can be written as

$$\sum_{k=1}^{N_{\text{REG}}} \sum_{i \in R_k \setminus \{1, \ldots, M_{\text{UNIT}}\}} \bar{\alpha}_{k1} + \frac{\bar{\alpha}_{k2}}{q_0\mu_k} = b_T - b_E,$$  \hspace{1cm} (37)

where the notation $R_k \setminus \{1, \ldots, M_{\text{UNIT}}\}$ implies that the first row of macroblocks is excluded from the regions. This yields the new value of $q_0$,

$$q_0 = \frac{\sum_{k=1}^{N_{\text{REG}}} \sum_{i \in R_k \setminus \{1, \ldots, M_{\text{UNIT}}\}} \bar{\alpha}_{i2}(\mu_k)^{-1}}{b_T - b_E - \sum_{i=M_{\text{UNIT}}+1}^{N_{\text{MB}}} \bar{\alpha}_{i1}}.$$  \hspace{1cm} (38)
In order to make sure that the resulting values of quantization step \( q_k = q_0 \mu_k \) fits within the allowed range \([q_{\text{min}}, q_{\text{max}}]\), we must make sure that the quantization step multipliers satisfy \( q_{\text{min}} q_0^{-1} \leq \mu_k \leq q_{\text{max}} q_0^{-1} \), which is done by saturating the values of \( \mu_k \). Since after saturation equation (38) may no more hold, we repeat the adjustment of \( q_0 \) for a few iterations, until its value cease to change. The same process is repeated after encoding every row of macroblocks, as follows:

\[
\begin{align*}
\text{initialization:} & \quad \text{Initial quantization step in each region } q_k = q_0 \mu_k, \text{ frame bit budget } b_T. \\
\text{input} & \quad \text{Allowed quantization step range } [q_{\text{min}}, q_{\text{max}}], \text{ division into regions } R_1, \ldots, R_{\text{NREG}}. \\
\text{for } l = 1, 2, \ldots, N_{\text{ROW}} \text{ do} & \quad \text{repeat} \quad \text{for } l = 1, 2, \ldots, N_{\text{ROW}} \text{ do} \quad \text{repeat} \\
& \quad \text{Encode } l\text{th row of macroblocks, producing } b_E \text{ bits.} \\
& \quad \text{Update the regions, } R_k \leftarrow R_k \setminus \{1, \ldots, l \cdot M_{\text{UNIT}}\}. \\
& \quad \text{Update the region-wise bit predictor coefficients, } \bar{\alpha}_k \leftarrow \sum_{i \in R_k} \alpha_i. \\
& \quad \text{Compute the new value of } q_0, \\
& \quad q_0 = \frac{\sum_{k=1}^{N_{\text{REG}}} \bar{\alpha}_k \mu_k}{b_T - b_E - \sum_{k=1}^{N_{\text{REG}}} \bar{\alpha}_k}. \\
& \quad \text{Saturate the quantization step multipliers: } \mu_k \leftarrow \min\{\max\{\mu_k, q_{\text{min}} q_0^{-1}\}, q_{\text{max}} q_0^{-1}\}. \\
& \quad \text{Compute the new values of quantization parameter,} \\
& \quad q_k = \text{round}\left(12 + 6 \log_2\left(\frac{q_k}{0.85}\right)\right). \\
& \quad \text{Update the available bit budget, } b_T \leftarrow b_T - b_E. \\
& \quad \text{until } q_0 \text{ ceases to change significantly} \\
\text{end} & \quad \text{end repeat} \\
\text{end} & \quad \text{end for} \\
\end{align*}
\]

**Algorithm 3**: Closed-loop bitrate controller.

The adaptation of the encoder models parameters can be performed after encoding each data unit, or after the entire frame is encoded. An example of closed-loop controller operation is shown in Figure 10.
6.6 Buffer safeguards

As an optional parameter, the sequence-level controller can produce the maximum and the minimum number of bits for the frame, exceeding or falling short of which will cause buffer over- or underflow, respectively. We call these margins the buffer safeguards and denote them by $b_{\text{max}}$ and $b_{\text{min}}$. Naturally, $b_{\text{min}} \leq b_T \leq b_{\text{max}}$. The closed-loop controller must ensure that even if the amount of bits predicted to be produced by the encoder deviates from the budget $b_T$, it still falls into the range $[b_{\text{min}}, b_{\text{max}}]$.

Using the buffer safeguards, an additional check is performed: whether the amount of bits produced assuming the quantization parameter to be the minimum possible falls below the lower safeguard,

$$
\sum_{k=1}^{N_{\text{REG}}} \tilde{\alpha}_{k1}^r + \frac{\tilde{\alpha}_{k2}^r}{q_0 \mu_k^r} \leq b_{\text{min}},
$$

or whether the amount of bits produced assuming the quantization parameter to be the maximum possible exceeds the higher safeguard,

$$
\sum_{k=1}^{N_{\text{REG}}} \tilde{\alpha}_{k1}^r + \frac{\tilde{\alpha}_{k2}^r}{q_0 \mu_k^r} \geq b_{\text{max}},
$$

If one of these conditions happen, it follows that no value of $q$ within the given range $[q_{\text{min}}, q_{\text{max}}]$ can make the amount of bits fall within $[b_{\text{min}}, b_{\text{max}}]$. In this case, the range $[q_{\text{min}}, q_{\text{max}}]$ should be extended, for example, by multiplying $[q_{\text{min}}$ and $q_{\text{max}}]$ or extending them to the maximum values allowed by the standard.

7 Conclusions

We presented a frame-level resource allocation and control algorithm for a scalable H.264 video codec. Together with a sequence-level allocation scheme, this completes the bitrate controller. The main features of the proposed algorithm are a content-aware encoder model, which is used to predict the amount of bits, PSNR and encoding time complexity as a function of the encoder param-
eters. This model is used to allocate the resources in an optimal way prior to encoding the frame. The resource allocation problem tries to optimize the objective quality (PSNR) given the bit budget constraints. It allows to shape the PSNR in any way. Coupled with a visual perception model, it is possible to perform PSNR shaping which will result in an optimal and frame-wise constant visual quality. The frame-level resource allocation is performed in a region-wise manner, in which macroblocks with similar properties and large likelihood to have the same encoding parameters are grouped together. This allows to avoid the usually unnecessary complicated macroblock-wise resource allocation. Moreover, by changing the number of regions, it is possible to control the complexity of the resource allocation. Finally, the time complexity model allows to perform load balancing across slices, ensuring that the processing time of each slice is approximately equal.

7.1 Extensions

The proposed scheme can be extended in a few ways. First, in the proposed algorithm, the encoding complexity is open loop and not controllable on the frame level. It can be assumed however that the encoding pipeline has scalable complexity, which is determined by the sequence-level controller. The complexity model can be used as an additional criterion in the resource allocation problem, by allowing not only to predict but also to control the encoding pipeline complexity scale at the frame level. This way, together with the bit budget, the sequence-level controller will give the computational complexity budget, which must be utilized as precisely as possible. For this purpose, the encoder models of PSNR and bit production should be extended and include the dependency of the computation complexity scale.

Second, additional resources can be explicitly integrated into the problem. An example of a resource (especially important to be optimized for in mobile devices) is power dissipation [2]. Algorithms for joint bitrate and power control have been recently reported in literature (see e.g. [14, 13]). In our setting, this will require a power dissipation model, similarly to that of the encoding complexity. Power dissipation can be set either as a constraint (power dissipation budget, in addition to bit and time budgets), or as an optimization cost. In the latter case, the resource allocation problem can be posed as a multicriterion constrained optimization problem, in which the goal is to minimize the power dissipation and make the PSNR as close as possible to the desired shape.
Third, the visual perception model can be extended by incorporating temporal information, such as motion. For example, it usually appears that in video sequences, our attention is more concentrated on foreground rather than background objects. Also, distortion of objects with fast motion is usually less perceivable. Finally, in representative video content, the attention to and the importance of lateral parts of the frame is usually smaller than the central ones.

References


Figure 3: Block-diagram of the frame-level controller.
Figure 4: Segmentation of frame into regions based on vector quantization in the $v - \Delta p$ domain.
Figure 5: Optimal vs. suboptimal vector quantization.
Figure 6: Segmentation into ten regions (bottom), performed by vector quantization of texture complexity (top left) and target PSNR offset shape (top right).
Figure 7: PSNR shaping with target PSNR offset from Figure 6. Top: allocated quantization parameter ($q_0' = 21, \Delta q_0' = 10$). Bottom left: predicted PSNR, bottom right: PSNR produced by the encoder.
Figure 8: Division of frame into slices.
Figure 9: Example of load balancing between slices of an average frame in case of four independent slice encoders (bar plot shows the relative complexity of each slice). When equal slices are used, the theoretical resource utilization is 92.0%. When load balancing is employed, the theoretical resource utilization is 98.2%. In cases when the complexity of the frame is very irregular (e.g. a scenery image with sky in the top part of the frame and complex texture in the bottom part), the difference is even more pronounced.
Figure 10: Example of closed loop controller operation. Shown is the predicted bit budget deviation in percents during encoding, as function of the macroblock row number (in the final row, the predicted deviation equals the actual one). Solid line represents the proposed closed-loop controller with update of $q'_0$ performed every row. The resulting budget misfit is $-2.4\%$. Dashed line represents encoding with a constant quantization parameter allocated by the sequence-level controller without closed-loop control. The resulting bit budget misfit is $-18.1\%$. 