2021 IEEE International Conference on Image Processing (ICIP) | 978-1-6654-4115-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIP42928.2021.9506186

RATE ESTIMATION TECHNIQUES FOR ENCODER PARALLELIZATION

Gaurang Chaudhari, Hsiao-Chiang Chuang, Igor Koba, Hariharan Lalgudi

Facebook Inc., 1 Hacker Way, Menlo Park, CA 94025

ABSTRACT

In this paper, we present a novel rate estimation technique for rate control with frame parallelization in video encoding. The proposed rate control accounts for internal delay in the frame encoder pipeline, which is suitable for multithreaded or faster software (SW) encoders and custom hardware or ASIC (HW) encoders. It includes an accurate frame size prediction model based on a complete set of first-pass statistics data for every frame. Using better prediction models with linear and nonlinear model functions, we achieve improved accuracy, with respect to encoding quality as well as improving rate control bits-per-frame planning.

Index Terms— Encoder parallelization, rate control, VP9, random forest, neutral network

1. INTRODUCTION

Rate control (RC) is an essential element of modern video encoders. By means of rate control, encoder estimates the available amount of bits to meet a specific bandwidth or file size goal, and maintains it for the duration of the video stream, while achieving best video quality for conditions. Rate control is not a normative part of the video coding standard which means it is up to the developers to implement it in the best way for the given encoder system.

Inputs to a typical rate control method are bit statistics (past frame bits, QP (Quantization Parameter), target bits, HRD (Hypothetical Reference Decoder) buffer constraints) and frame statistics (encoded quality, past and present frame complexity etc.). The output of rate control is QP for the next encoded video frame. QP for a typical video encoder determines a tradeoff between residual error (distortion) and number of encoded bits for the frame. It is important to note that while QP value for the next frame is decided by rate control conditioned on the prior statistics and encoder data, the final outcome of the decision, i.e. residual error (distortion) and number of coded bits of the frame is not known until encoder finishes encoding a given video frame. Only then the frame encoder updates information in the rate control unit, used for decision for subsequent frames.

In a typical video encoder, a rate control mechanism can be described in a following sequence:

1. Estimate frame complexity and target bits for the frame.
2. Choose a QP that gives the best trade-off between controlling rate and overall quality.
3. Encode the frame.
4. Update number of encoded bits and other frame statistics.

Depending on use cases, first-pass analysis data for a number of frames is completed ahead of time and used for rate control algorithms. While the method proposed in this paper may be potentially used for any type of video encoder, we will describe it with respect to one particular video encoder, VP9 [1] and for faster SW or custom HW implementations.

2. RELATED WORK

Li et. al. proposed the R-λ domain rate control [2]. By modeling the rate-distortion (RD) relationship using a hyperbolic function, Li proposes to use a linear function to determine the frame-level and coding unit (CU)-level QP values with the Lagrangian multiplier λ. The results show improved accuracy as well as better overall BD-rate quality in terms of the PSNR quality metric when compared to the Unified Rate-Quantization method [3]. The R-λ domain method is especially suitable for low-latency use cases while the R-λ relationship can also be used in other scenarios. For high efficiency use cases, two-pass methods may be employed.

Several two-pass rate control schemes are proposed to employ a lightweight first-pass coding which provides pre-analysis of the video characteristics (such as scene change) before the actual encoding pass. In [4], a pre-encoding scheme using only 16x16 CU is proposed. It is observed that in some sequences where scene change occurs, it is required to refresh the parameter estimation. The authors proposed to use an iterative algorithm to update the R-λ model based on abnormal detection to reset the model parameters along with an updated weight based on the proportion of the rate within the same CTU. The results improve both the accuracy and the overall PSNR by up to 6dB. In [5], following the SSIM-inspired divisive normalization framework, Wang et. al. address the bit allocation problem by adjusting the λ value for each GOP. The QP value is determined based on the Sum of Absolute Transformed Difference (SATD) with four-dimensional first-pass statistics. In [6], based on the R-λ model, Zupancic et. al. proposed to construct the bit-rate profile for each intra period using pre-encoding, which uses a simplified encoder to perform a variable-QP encoding scheme to avoid coding the same frame multiple times. The proposed method shows around 6% BD-rate improvement compared to the RC method in HM. In [7], Deng et al. used pre-compression with multiple QP values in the first pass and used the collected rate and distortion numbers to construct a R-D model using a least-squares method. A pair of λ and QP values are input to the actual encoding pass.

3. LIBVPX RATE CONTROL

Libvpx [8] is an open source VP9 encoder implementation frequently used as a reference by developers. This paper first describes a particular rate control implementation in the libvpx [8] library followed by advancements done in the proposed implementation. Libvpx [8] has a 2-pass constrained quality encoding method, in which a very fast first-pass analysis of the

entire video stream is done, and the resulting statistics are used for computing target frame sizes and planning bit distribution across the stream. Then, determined values of maximum and minimum frame sizes are used for choosing suitable quantization parameter Qp for every frame.

A second (main) encoding pass can be described in Figure 1. The loop in Figure 1 is repeated for every encoded frame. The step "Choose quant parameter" calculates the quantization parameter for the next frame, given the maximum and minimum frame size and quantization parameter. It relies on the frame size prediction to estimate bits at Q, which in turn relies on predicted bits per macroblock. The step "Encode frame" does most of the actual frame encoding such as mode decision, transform coefficients, and residual calculation for each superblock in the frame. The subsequent "Entropy coding" step is a final step, where previously generated mode decision and transform coefficients are packed using entropy coding method according to the standard. VP9 uses a tree-based boolean non-adaptive binary arithmetic encoder to encode all syntax elements. It is important to note that while "Encode frame" and "Entropy coding" are integral parts of the frame encode process, in a typical encoder implementation they are executed consecutively, in a pipelined order. Only after the "Entropy coding" step, the size of encoded frames becomes known. The final step is a post-encode update of rate control, by means of which the rate control algorithm is informed about the size of the encoded frame. This is necessary for calculating instantaneous state of buffer and rate estimation used for a closed loop rate control algorithm.

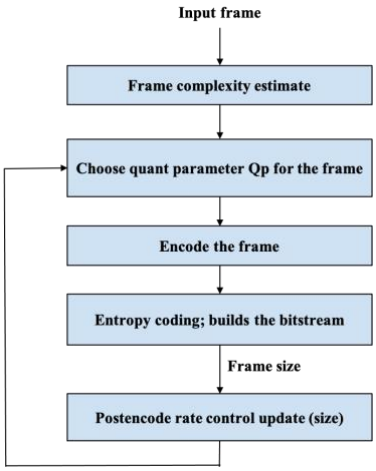

Figure 1. Algorithm of rate control in conventional encoder.

In the original libvpx [8] implementation, the step "Choose quant parameter" depends on a macroblock bits prediction model

$$Bits_{projected} = (C_1/q + C_2) \times R_{cf} \quad (1)$$

Where C_1 and C_2 are constants, q is quantization parameter for the frame. Rate correction factor R_{cf} is a model parameter which is updated based on size of the last frame of the same level in a *GF* structure (*Inter, Golden Frame (GF), Alt-Ref Frame (ARF) and Key Frame (KF)*), per VP9 standard [1]. Separate values for rate correction R_{cf} [level] are maintained for each frame reference level (i.e. frame type and level in group of frames hierarchy).

4. PROPOSED RATE CONTROL METHOD

As explained above, "Encode frame" and "Entropy coding" in a typical encoder implementation are executed one after another. This helps in the hardware ecosystem, since the compute requirements of both these steps are quite different and the mode decision for example, needs to operate at a much faster speed to match the system latency followed by the final entropy coding step. Typically, in a faster SW or a custom HW encoder, the encode frame (mode decision, transform coefficients) step and the final entropy coding step operate at a different throughput pipeline or threads. There can be various custom HW architecture implementations and we want to focus on the faster architectures which primarily have entropy coding operate on a different frame. As explained in the above section, the typical rate control depends on the frame size bits to decide encoding parameters. The frame size bits are an outcome of the entropy coding and if the entropy coding is operating at a different frame, then, the latest frame size bits information is not available. We propose a solution to this problem.

The proposed VP9 rate control operation roughly corresponds to Figure 1, with the following significant distinction: The operations "encode frame" and "entropy coding: build the bitstream" operate in a long pipeline, with the long delay essentially asynchronous and with sometimes non-deterministic delay. Such a long pipeline and delay is not unusual for multithreaded or faster SW encoders and custom HW encoders, where several frames may be encoded in parallel. The distinctive problem arises in the "postencode rate control update" due to unavailability of the exact size of encoded frame, as the entropy coding step may be several frames behind. That distinction makes rate control function essentially an open-loop task instead of the closed loop rate control in libvpx [8].

To mitigate the problem, the encode algorithm in Figure 1 was modified with the changes shown in Figure 2. We replaced the unknown exact frame size value with an approximate value where the function which predicts frame size relies on Eq. (1) above.

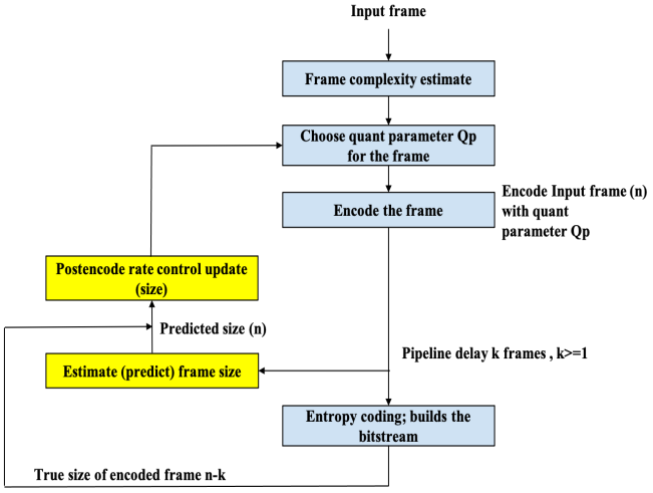

Figure 2. Algorithm of rate control with a delay in pipeline.

But the algorithm in Figure 2, has an issue that by using an approximate predicted frame size value, the rate control internal state and bit buffer level value can drift from the actual bits count and buffer value determined by the actually encoded frame sizes (on the output of delayed encode pipeline). To avoid drift of rate control buffer value, the actually encoded frame size value (or true_size) is used to make corrections for the rate control buffer level and bit prediction model as soon as it becomes available. The difference between previously predicted frame size and the actual frame size is

computed when the later is available, and this delta in formula (2) is added to the predicted size of the next coded frame as shown in formula (3), below, where n is a current encoded frame number, k is a frame delay in the encoder pipeline.

$$prediction\_delta_{n-k} = true\_bytes_{n-k} - size\_predicted_{n-k} \quad (2)$$

For the next frame update:

$$size\_predicted\_new = size\_predicted_n + prediction\_delta_{n-k} \quad (3)$$

Then the function "postencode rate control update" is called with the corrected value size_predicted_new. Due to this correction, the computed bits budget and buffer level errors will not grow unbounded over time and are bound by the error of frame size prediction for k consecutive frames at most. Table 1 represents the coding efficiency results for the simulated pipeline delay of the encoder from 1 to 5 frames, using Bjontegaard delta (BD-rate) [9] with respect to three popular video quality metrics (QM): SSIM [10], PSNR, VMAF [11]. The dataset used is Facebook (FB) internal dataset [12] which are 400 top-viewed public videos from FB Pages. These videos were tested in an anonymized manner without subjective analysis. BD-rate metric gives average bit-rate reduction (-ve) or increase (+ve) for the same quality between two encoding methods. Our anchor is encoded without any frame delay. The *cq-level* values used are {33, 37, 41, 45}. Command line used is given below -

--codec=vp9 --passes=2 --limit=300 --i420 --profile=0 --cpu-used=1 --fps=30 --kf-min-dist=150 --kf-max-dist=150 --arnr -maxframes=7 --arnr-strength=5 --lag-in-frames=25 --aq-mode=0 --end-usage=cq --target-bitrate={bitrate} --cq-level={cq-level} --min -q=0 --max-q=63 --bias-pct=50 --minsection-pct=20 --max section-pct=400 --auto-alt-ref=6 --frame-parallel=0 --threads=1 --tile-columns=0

Table 1. Average increase in BD-rate (% of rate loss).

QM	delay = 1frame	delay = 3frame	delay = 5frame
SSIM	0.066%	0.87%	1.29%
PSNR	-0.04%	0.67%	0.99%
VMAF	0.036%	0.70%	0.93%

Table 1 results show that the prediction of frame size can be further improved. If the frame size prediction model worked perfectly, the result with delayed pipeline would be the same as the baseline, i.e the no-delay encoder. It can also be extrapolated that the frame size prediction model in Eq. (1) used in the original libvpx [8] encoder can be tuned further. Main deficiency of the Eq. (1) is that rate correction factor R_{cf} is calculated based on statistics of previously coded frames of the same coded type. The assumption is made that consecutive frames have the same coding complexity. It does not account for changes in the frame complexity over time, and the assumption potentially breaks in the beginning of the stream or at every scene cut or transition, i.e. at places where the libvpx [8] encoder has no prior initial knowledge of frames complexity and initial values of rate correction factor for the first frames starts with a default value. Although the number of keyframes in the stream is few, one out of 150 frames in a 30fps, 5-sec GOP, keyframes have the largest size out of all frame types. This misprediction in keyframe size may result in large error of bit buffer and bits budget estimation in rate control.

4.1. Improved Prediction Model for Frame Size

First-pass statistics in a two-pass encoding can give some information about coding complexity of each frame. Using the first-pass statistics, the prediction model for frame size can be improved. In the first-pass statistics, there are five types of raw data being calculated:

- SSE of intra prediction
- SSE of inter prediction with LAST_FRAME
- SSE of inter prediction with GOLDEN_FRAME
- Block noise energy
- Motion vectors

The raw data is compared with thresholds or directly accumulated for 23 statistics at the frame-level, which can later be employed by the rate control algorithm. In the original libvpx [8] encoder, first-pass data is used for planning allocation of bits for future frames within each GOP interval. In addition to that, the first-pass data can also be used for an improved frame size prediction model. The proposed rate correction factor R_n, for the frame n can be computed as

$$R_n = R_{1st} \times R_{cf} \quad (4)$$

where R_{1st} is a rate factor that can be predicted from 1st pass statistical data, for the frame coding type. The rate factor value R_{1st} can be computed by linear or nonlinear prediction model from the 1st pass data. R_{cf} is a content-dependent correction factor which is updated dynamically, based on the last frame of the same coding type. This parameter will compensate for a possible discrepancy between the rate factor value R_{1st} and the actual video stream content, i.e. R_{1st} reflects the initial predicted complexity of the frame, and R_{cf} represents a dynamically updated correction value, which depends on size of previously encoded frames. The rate factor R_{1st} can be computed using a linear or nonlinear prediction model as a function of computed first pass statistics data for the given video frame.

As a machine learning problem to train a prediction model, data samples for each encoded frame (frame sizes and first pass statistics data) is used. These data samples are then classified into bins, according to a frame coding type in VP9, i.e. level in a *GF* structure (*Inter, ARF, GF_ARF, KF*). The prediction model for rate factor R_{1st} was trained separately for each frame type bin. For each frame type bin, a number (~10K) of samples for the training data set, and (~10K) samples for test data set was randomly selected for the purpose of machine learning. Since the number of keyframes (Intra) is few, fewer samples (about 1.5K) were used.

4.1.1. *Training prediction model for R_{1st} for keyframe (Intra)*

A simple linear model with the assumption that the scale factor is a linear function of one parameter, intra_error is tried as a first experiment and regression statistics are:

R square = 0.658707139

Standard Error = 0.178272478

For improving prediction model, 4 parameters relevant for the keyframe complexity estimate from the 1st pass statistics data were selected: *intra_error* (an estimate of per-pixel intra coding error), *frame_noise_energy* (an estimate of per-block (16x16) noise level), *intra_skip_pct*, and *intra_smooth_pct* (both *intra_skip_pct* and *intra_smooth_pct* indicate the percentage of blocks whose intra coding error is less than a threshold, while *intra_skip_pct* uses a much smaller threshold value). Linear regression model with these input variables (4 variables + intercept) gives significantly better result:

R square = 0.765029423

Standard Error = 0.093895134

To test the effect from using the above linear regression model for predicting R_{1st} rate factor for keyframe size in VP9 encoder, we

repeated the same test conditions with same anchor as Table 1 and the results are shown in Table 2.

Table 2. Keyframe size prediction results - average BD-rate delta

QM	delay = 1frame	delay = 3frame	delay = 5frame
SSIM	-0.14%	0.24%	0.44%
PSNR	-0.19%	0.13%	0.20%
VMAF	-0.22%	0.18%	0.26%

4.1.2. Training prediction model for R_{1st} for Inter frames

The first problem to address is to select the appropriate variables out of the more than 20 first pass statistics since not all variables have equal significance for rate prediction. Principal Component Analysis (PCA) is used to a) determine how many variables (dimensions) are sufficient for the model b) which of the variables can have highest significance for the top principal components (which are eigenvectors of the covariance matrix). As a result, the following set of variables for the model were selected:

sr_coded_error (estimate of per-block inter coding error with GF), *frame_noise_energy*, *pcnt_motion* (percentage of blocks coded with last frame), *pcnt_second_ref* (percentage of blocks coded with GF), *pcnt_intra_low*, *pcnt_intra_high* (*pcnt_intra_low* and *pcnt_intra_high* are percentage of intra coded blocks with low and high variances, respectively), *intra_skip_pct*, *intra_smooth_pct*. (8 variables + intercept)

Inter frames are classified by levels, where 0 is a basic inter frame, level 2 is reference frame (ARF), level 3 is a GF ARF frame. A linear regression method with 8 input variables is used to create a prediction model on the training data for frames at level 3 (GF ARF). Measuring model accuracy on a training data set is not enough and it is important to check how the model generalizes for different sets of data. The statistics results for linear regression on the training data and test data set are shown in Table 3 (*Root MSE - lower value is better, R square - higher value is better).* The accuracy of the model on the test data set above is slightly lower, but still within acceptable margins, i.e. the model is not overtrained.

Table 3. Statistics results for linear regression model.

	Training Data	Test Data
Root MSE	0.109148	0.11430165
R square	0.5744763	0.5571709

To explore if a nonlinear function gives better prediction results for the same training and test data sets, two different powerful nonlinear prediction methods, a Random Forest and a multilayer neural network (ANN), were also tested. Both models are available in a Scikit-learn [13], a machine learning library in python, which was used for data analysis and modeling experiments. Random Forest is a supervised learning algorithm, with a group of decision trees, trained with a bagging method. Results on the training data and test data set are shown in Table 4:

Table 4. Results using Random Forest and ANN models

	Training Data	Test Data
Root MSE (RF)	0.02760262	0.06280004
R square (RF)	0.97278597	0.86632477
Root MSE (ANN)	0.08088808	0.090404541
R square (ANN)	0.76629876	0.72297972

Compared to the prediction results on a training data set, the results of evaluating a Random Forest model on a test data set is considerably worse, which is a typical indication of an over-trained model, although the results on the test set above are still better than the results with a linear regression model.

A multilevel neural network model (ANN) with the same training data was also evaluated with 5 layers, with the following size of each layer 90, 90, 90, 90, 20. Results on the training data and test data set are shown in Table 4. ANN model has better accuracy than linear regression, but accuracy on the training data set is not as good as Random Forest. However, from a complexity perspective, a linear regression method is preferable. It simplifies implementation, especially for the firmware memory limitations involved in the HW encoder. Similar deductions were done for other inter frame levels.

5. ENCODING RESULTS

After implementing the frame size prediction using linear regression models discussed above, for both Inter and Intra frames, the BD-rate results are shown in Table 5.

Table 5. Frame size prediction results using linear regression - average BD-rate delta.

QM	delay = 1frame	delay = 3frame	delay = 5frame
SSIM	-0.18%	-0.15%	-0.19%
PSNR	-0.29%	-0.28%	-0.43%
VMAF	-0.39%	-0.40%	-0.65%

Negative BD-rate values mean average bit-rate reduction for the same quality between two encoding methods, i.e, the target encoder is better than the anchor. Our anchor is the original libvpx [8] encoding without frame delay. The proposed prediction models improve quality for the encoder even with a frame delay pipeline, compared to the original libvpx encoder without frame delay.

As discussed in Section 4, the frame size prediction model in Eq. (1), used in the original libvpx [8] encoder, can be tuned for further improvement as it does not account for changes in the frame complexity of frames over time. To add to this observation, we experimented using the proposed prediction model in the original libvpx encoder without frame delay and the average BD-rate results:

SSIM : -0.46%
PSNR : -0.31%
VMAF : -0.31%

Evidently, the proposed frame prediction model improves quality and rate control operation, due to better frame size prediction and choice of quant parameter Qp for each frame.

6. CONCLUSION

In this work, to address the inherent latency issue of obtaining accurate frame size in hardware rate control, we presented a novel scheme to predict the frame size using the 1st-pass statistics. Two sets of feature statistics are discovered using the PCA method for both key frames and inter frames, respectively. We compare the prediction accuracy using linear regression, ANN, and Random Forest, and present the associated coding performance. In the case of no frame delay, but also with up to 5 frame delay, experimental results show improved BD-rate performance compared to the original libvpx algorithm.

7. REFERENCES

[1] Mukherjee, Debargha et al., (2015). “A Technical Overview of VP9--the Latest Open-Source Video Codec,” *SMPTE Motion Imaging Journal*. 124. 44-54. 10.5594/j18499.

[2] B. Li, H. Li, L. Li, and J. Zhang, “λ Domain Rate Control Algorithm for High Efficiency Video Coding,” *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3841–3854, 2014.

[3] H. Choi, J. Nam, J. Yoo, D. Sim, and I. Bajic´, “Rate Control Based on Unified RQ Model for HEVC,” *document Rec. JCTVC-H0213*, San Jose, CA, USA, Feb. 2012.

[4] J. Wen, M. Fang, M. Tang, and K. Wu, “R-λ Model Based Improved Rate Control for HEVC with Pre-Encoding,” in *Proc. IEEE Data Compress. Conf. (DCC)*, pp. 53-62, Apr. 2015.

[5] S. Wang, A. Rehman, K. Zeng, J. Wang, and Z. Wang, “SSIM-motivated two-pass VBR coding for HEVC,” *IEEE Trans. Circuits Syst. Video Techn.*, vol. 27, no. 10, pp. 2189–2203, 2017

[6] I. Zupancic, M. Naccari, M. Mrak, and E. Izquierdo, “Two-Pass Rate Control for Improved Quality of Experience in UHDTV Delivery,” *IEEE J. Selected Topics Signal Process.*, vol. 11, no. 1, Feb. 2017

[7] L. Deng, F. Pu, S. Hu, and C.-C. J. Kuo, “HEVC encoder optimization based on a new RD model and pre-encoding,” in *Proc. 2013 IEEE Picture Coding Symp. (PCS)*, Dec. 2013.

[8] VP9 libvpx source code github (libvpx-1.8.0), https://chromium.googlesource.com/webm/libvpx/+/refs/tags/v1.8.0

[9] Bjøntegaard, “Calculation of average PSNR differences between RD-curves (VCEG-M33),” in VCEG Meeting (ITU-T SG16 Q. 6), 2001.

[10] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP .2003.819861.

[11] Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A., and Manohara, M., “Toward a practical perceptual video quality metric,” https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, 2016.

[12] Yu Liu, Open Source, Video Engineering, “AV1 beats x264 and libvpx-vp9 in practical use case,” https://engineering.fb.com/video-engineering/av1-beats-x264-and-libvpx-vp9-in-practical-use-case/, April 2018

[13] Buitinck, L. et al., 2013. API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122.