IMPROVED RATE CONTROL AND MOTION ESTIMATION FOR H.264 ENCODER

Loren Merritt[‡] *and Rahul Vanam*[†]

[‡] VideoLAN, Club VIA Centrale Réseaux, Résidence Ecole Centrale,
2, avenue Sully Prudhomme, 92 290 Châtenay Malabry, France.
[†]Dept. of Electrical Engineering, University of Washington, Box 352500, Seattle, WA 98195-2500, USA.
lorenm@u.washington.edu, rahulv@ee.washington.edu

ABSTRACT

In this paper, we describe rate control and motion estimation in x264, an open source H.264/AVC encoder. We compare the rate control methods of x264 with the JM reference encoder and show that our approach performs well in both PSNR and bitrate. In motion estimation, we describe our implementation of initialization and show that it improves PSNR. We also propose an early termination for simplified uneven cross multi hexagon grid search (UMH) in x264 and show that it improves the speed by a factor of 1.5. Finally, we show that x264 performs 50 times faster and provides bitrates within 5% of the JM reference encoder for the same PSNR.

Index Terms— Video coding, rate control, motion estimation.

1. INTRODUCTION

The joint effort by ITU-T's Video Coding Experts Group and ISO/IEC's Moving Pictures Experts Group resulted in standardization of H.264/MPEG-4 AVC in 2003 [1]. Like previous standards, H.264 specifies only a decoder, therefore allowing for improvement in compression rate, quality and speed in designing the encoder. The encoder developed by the Joint Video Team, known as the Joint Model (JM) [2], has been used as a reference by encoder developers in enhancing existing algorithms. However, due to its slow speed, its use has been limited.

Another H.264 open source encoder is x264 [3]. Its development started in 2003, and it has been used in many popular applications like ffdshow, ffmpeg and MEncoder. In a recent study, x264 showed better quality than several commercial H.264 encoders [4]. The high performance of x264 is attributed to its rate control, motion estimation, macroblock mode decision, quantization and frame type decision algorithms. In this paper, we describe the implementation of rate control in x264 (ver 0.47.534) [3] and compare its performance with the JM encoder (ver. 10.2) [2]. We also describe our modified initialization and early termination algorithms for motion estimation in x264. Finally, we compare the overall performance of the above two encoders and show that x264 is about 50 times faster and provides bitrates within 5% of JM for the same PSNR.

This research has been supported in part by the NSF grant CCF-0514353.

2. RATE CONTROL

Rate control allows selection of encoding parameters to maximize quality under bitrate and decoder video buffer constraints. The rate control in H.264 can be performed at three different granularities - group of pictures level, picture level, and macroblocks level [5]. At each level, the rate control algorithm selects the quantization parameter (QP) values that determine the quantization of the transformed coefficients. As the QP increases, the quantization step size increases and the bitrate decreases. The rate control in x264 is based in part upon libavcodec's implementation [6], which is mostly empirical. It includes five different modes, one two-pass and four one-pass modes, that are described below. In the constant bitrate mode, each macroblock is allowed to have a different QP, while in other modes, the QP is determined for an entire frame.

2.1. Two pass (2pass)

In this approach, data obtained from each frame during a first pass are used for allocating bits globally over the file during the second pass. After applying a one-pass mode in the first pass, the 2pass approach is implemented as follows:

1. The relative number of bits to be allocated between P-frames is selected independently of the total number of bits and it is calculated empirically by $bits \propto complexity^{0.6}$, where $complexity$ is the predicted bit size of a given frame at constant QP. The I- and B-frames use the QP from nearby P-frames with a constant offset.

2. The results of step (1) are scaled to fill the requested file size.

3. After encoding each frame, the future QPs are updated to account for the mispredictions in size (this is referred to as *long-term compensation*). If the second pass is consistently off from the predicted size, then the target size of all future frames is multiplied by a $factor = \frac{predicted\ filesize}{real\ filesize}$, and

their QPs are recomputed. Apart from long-term compensation, there is *short-term compensation* to prevent x264 from deviating too far from the desired file size near the beginning (when there are less data for long-term compensation) and near the end (when long-term compensation does not have time to react). In short-term compensation, the compensation factor is $C^{(real\ filesize - predicted\ filesize)}$, where C is a user defined constant.

2.2. Average Bitrate (ABR)

This is a one-pass scheme which produces near-constant quality within a specified file size. Since the rate control must be done without the knowledge of the future frames, ABR cannot exactly achieve the target file size. The steps given below are numbered to match the corresponding steps in 2pass.

1. The relative bit allocation is the same as in 2pass, except that instead of estimating complexity from a previous encode, we run a fast motion estimation algorithm over a half-resolution version of each frame, and use the Sum of Absolute Hadamard Transformed Differences (SATD) of the residuals as the complexity. Because the complexity of the following Group of Pictures is unknown, the I-frame QPs are based on the past.

2. We do not know the complexities of the future frames, so we only scale based on the past. The scaling factor is set to the one that would have resulted in the desired bitrate if it had been applied to all the frames so far.

3. Long and short term compensation are the same as in 2pass. By tuning the strength of compensation, it is possible to obtain quality ranging from close to 2pass (but with file size error of $\pm 10\%$) to lower quality with strict file size.

2.3. Video buffer verifier compliant constant bitrate (CBR)

This is a one-pass mode designed for real-time streaming. The steps given below are numbered to match the corresponding steps in 2pass.

1. It uses the same complexity estimation for computing bit size as ABR does.

2. The scaling factor used for achieving the requested bitrate is based on a local average (dependent on the buffer size of the video buffer verifier) instead of all past frames.

3. The overflow compensation is the same as in ABR, but runs after each row of macroblocks instead of per-frame.

2.4. Constant rate-factor and Constant Quantizer

The constant rate-factor mode (CRF) is a one-pass mode that is optimal if the user specifies quality instead of bitrate. It is the same as ABR, except that the scaling factor is a user-defined constant and no overflow compensation is done. Finally, the constant quantizer mode (CQP) is a one-pass mode where QPs are simply based on whether the frame is I-, P- or B-frame. This mode is used when the rate control option is disabled.

Table 1. Rate control in x264 and JM encoder for a target bitrate of 1200 kb/s

Approach	Bitrate (kb/s)	PSNR (dB)
JM-CBR	1189.4	40.810
x264-CBR	1120.7	41.569
JM-CQP	1212.9	44.035
x264-CQP	1137.2	44.033
x264-ABR	1169.9	42.815
x264-CRF	1133.9	44.176
x264-2pass	1199.2	44.472

2.5. Comparisons

In this section, we compare different rate control methods in both x264 and JM encoder [2] for a target bitrate of 1200 kb/s. For comparison, we use a video sequence (Elephants Dream) [7], at 720x480 pixels resolution, 25 frames per second (fps) and 15691 frames. For both encoders, we use five reference frames and select the other parameters to result in the best PSNR. Table 1 gives the resulting bitrate and PSNR for different rate control approaches in JM and x264 encoder. We obtain the PSNR by averaging the individual PSNR values for YUV in the proportion they appear (4:2:0), and then average it over all frames and video clips. As expected, x264 2pass approach performs the best by achieving closest to the target bitrate with the highest PSNR. For CBR, x264 achieves 0.76 dB PSNR improvement over JM, even with a lower bitrate. For CQP, using QP = 21 and 23 for x264 and JM, respectively, we obtain the same PSNR for both the encoders, but x264 has a lower bitrate. Among the different x264 one pass approaches, CRF provides the best quality. Although CRF and CQP have similar average PSNR, psychovisually CRF performs better than CQP.

3. MOTION ESTIMATION

Motion estimation (ME) is the most complex and time consuming part of the H.264 encoder as it uses multiple prediction modes and reference frames. There are four different integer-pixel motion estimation methods provided by x264: diamond (DIA), hexagon (HEX), uneven multihexagon (UMH) [8] and successive elimination exhaustive search (ESA) [9]. UMH provides good speed without sacrificing significant quality, while hexagon is a good tradeoff for higher speeds. In step (1) of this section, we propose a modified initialization for all motion search methods that improves PSNR and in steps (2)-(7) we describe early termination and range adaptation (ETRA) algorithm for UMH that improves its speed. The following steps describe both algorithms:

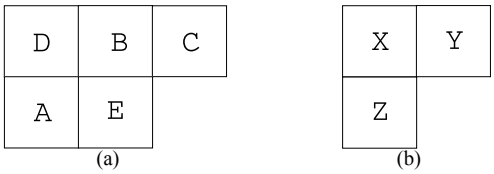


Fig. 1. Candidate motion vectors for motion search from (a) the current frame and (b) the previous frame, where E and X represents the current macroblock location.

1. Modified Initialization: We consider up to 10 candidate motion vectors (MVs) that consist of four MVs from the neighboring locations in the current frame (A,B,C,D), three MVs in the previous frame (X,Y,Z), the (0,0) MV, the median MV and the temporal direct MV. These 10 MVs are used in the initialization step for all ME methods. Figures 1(a) and 1(b) illustrate current frame MVs and previous frame MVs, respectively. Each block that is being encoded in the current frame is referred to as the reference index. We keep track of the MVs for each reference index at each position in space (referred to as motion field). The four neighboring MVs are taken from the motion field of the reference index that is currently being searched. The MVs from the previous frame are scaled based upon the amount of time they span. We compare the sum of absolute differences (SAD) of each candidate, and choose the best SAD.

2. One step of the diamond search is applied to (0,0) and the median MV.

3. If the block size is 4x4, break from UMH and apply the hexagon search. Else, continue.

4. Apply one step of diamond search to the best MV. This is labeled as '1' in Fig 2.

5. Early termination: If step (4) did not find a new best MV, then perform the following early termination:

a. Perform diamond search of radius two. This is labeled as '2' in Fig 2.

b. If steps (1) or (2) found a new MV, run a symmetric cross search (radius 7) and an octagon search (radius two). This is labeled as '3' in Fig 2.

c. If steps (5a) and (5b) did not find a new MV, then break.

6. Adaptive radius: Select the search range for step (7), based on the magnitude of the best SAD so far, and on the smoothness of the motion field (i.e., the variance between the predictors used in step (1)). The default search range is 16. This may decrease as low as 12 if the SAD is small and the predictors are similar, and may increase as high as 24 if the SAD is large and the predictors differ greatly.

7. Run 5×5 exhaustive search, uneven cross, multi-hexagon-grid, and iterative hexagon refinement as in JM.

The above algorithm differs from the JM implementation in many ways. JM does not do steps (5) or (6), and uses three candidate MVs in step (1). Between each of these candidates, JM applies a termination threshold based on the candidate's SAD, and decides whether to skip directly to diamond or hexagon search. In x264, we find it useful to check all candidate MVs.

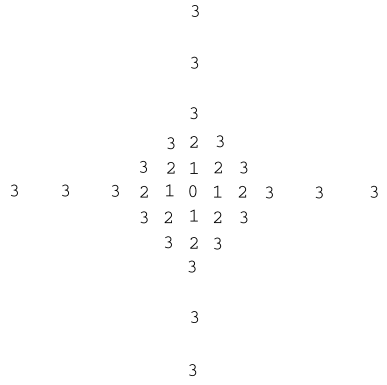


Fig. 2. Search approach for early termination decision in uneven multi-hexagon search. Diamond search of radius one is labeled as '1,' diamond search of radius two is labeled as '2,' and symmetric cross search of radius 7 and octagon search of radius two is labeled as '3.'

Table 2. Comparison of motion search methods in x264 with and without optimizations.

Approach	Δ_{psnr} (dB)	$time_{ME}$ (s)
DIA_x264	-0.039	3.82
DIA_no_emv	-0.205	2.98
HEX_x264	-0.031	4.67
HEX_no_emv	-0.165	3.73
UMH_x264	-0.005	10.41
UMH_no_emv	-0.035	9.86
UMH_no_ETRA	-0.006	16.70
UMH_no_both	-0.035	15.62
ESA_x264	0.000	50.51
ESA_no_emv	-0.047	51.36

3.1. Comparisons

In this section, we compare x264 with and without the use of extra MVs during initialization and the use of ETRA for UMH. In our tests, we use 19 CIF videos from [10] and 11 videos (720p and 1080p) from [11]. In Table 2, we list the PSNR averaged over all video clips of each ME approach relative to ESA with extra MVs (Δ_{psnr}). We also list the geometric mean of the time taken for ME only ($time_{ME}$). Since larger resolution videos take longer to encode than smaller resolution videos, we use geometric mean instead of arithmetic mean for $time_{ME}$. All timing measurements in this

paper are obtained by running tests on an AMD Athlon64 2.2 GHz system.

We represent ME approach ME that use extra MVs at initialization (together with ETRA in UMH) as ME_x264, ME without extra MVs as ME_no_emv, UMH without ETRA as UMH_no_ETRA and UMH without ETRA and extra MVs as UMH_no_both. Here, ME can be DIA, HEX, UMH or ESA. The use of extra MVs improves the PSNR of DIA and HEX and they are comparable to UMH_no_emv, but faster by a factor of 2.6 and 2.1, respectively. The use of extra MVs and ETRA in UMH improves the PSNR by 0.03 dB and speed by a factor of 1.5.

4. COMPARISON WITH THE JM ENCODER

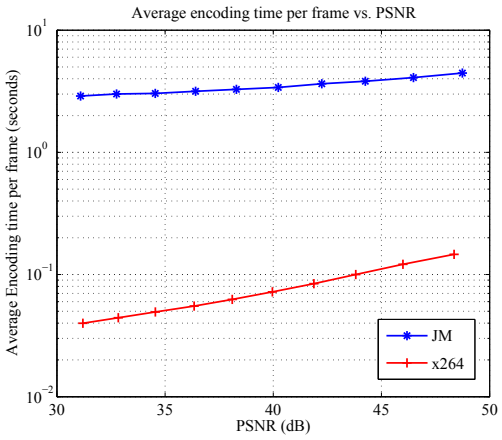
In this section, we compare the performance of x264 and JM using 19 CIF sequences available in [10]. We compare the two encoders for 10 different QPs, from 12 to 39 in steps of three. The encoding parameters for both the encoders are the same as in Section 2.5. Comparison is made based on average encoding time per frame and bitrate normalized for fixed PSNR. From Fig 3(a), we find that on average, x264 performs 50 times faster when compared to JM for the same PSNR. Without assembly optimized code for primitive operations, x264 performs 20 times faster than JM for the same PSNR. In Fig 3(b), we see that x264 provides bitrate savings of up to 3.4% over JM for PSNRs above 38 dB and for PSNRs below 38 dB, there is slight increase in bitrate up to 5%.

5. CONCLUSION

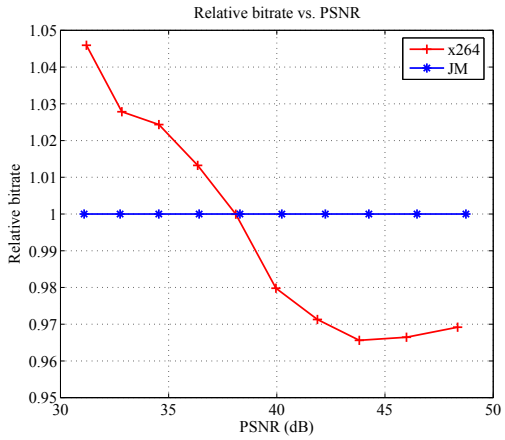
In this paper, we presented five different rate control modes in x264 and made comparisons with the JM encoder. For the CBR scheme, x264 achieves a lower bitrate and 0.76 dB higher PSNR when compared to JM. We show that the use of extra MVs in motion estimation improves the PSNR and early termination improves the speed of UMH by a factor of 1.5. Finally, we show that x264 performs about 50 times faster and provides bitrates within 5% of JM for the same PSNR.

6. ACKNOWLEDGEMENT

We thank Laurent Aimar, Min Chen, Radek Czyz, Christian Heine, Alex Izvorski, Eric Petit, Måns Rullgård, and Alex Wright for their contributions in developing x264. We also thank Prof. Eve Riskin and Prof. Richard Ladner for helpful comments and discussions.



(a)



(b)

Fig. 3. JM vs. x264 (a) Average encoding time per frame vs. average PSNR, (b) relative bitrate vs. average PSNR.

7. REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. CSVT*, vol. 13, no. 7, pp. 560–576, 2003.

[2] "JM ver. 10.2," http://iphome.hhi.de/suehring/tml/index.htm.

[3] "x264," http://developers.videolan.org/x264.html.

[4] "MPEG-4 AVC/H.264 video codec comparison," CS MSU Graphics & Media Lab Video Group, Dec 2005, http://www.compression.ru/video/index.htm.

[5] G. J. Sullivan, T. Wiegand, and K.-P. Lim, "Joint model reference encoding methods and decoding concealment methods," *JVT-N046*, Jan 2005.

[6] "ffmpeg," http://ffmpeg.org/.

[7] "Elephants dream," http://orange.blender.org/.

[8] X. Yi, J. Zhang, and N. Ling, "Improved and simplified fast motion estimation for JM," *JVT-P021*, July 2005.

[9] X. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. on Image Proc.*, vol. 9, no. 3, pp. 501–504, 2000.

[10] "CIF sample videos," http://www.tkn.tu-berlin.de/research/evalvid/cif.html.

[11] "HD sample videos," ftp://ftp.ldv.e-technik.tu-muenchen.de/pub/test_sequences/.