Optimizing Motion Estimation for H.264 Encoding

Mateus Krepsky Ludwich
Federal University of Santa Catarina – UFSC
Laboratory for Software and Hardware Integration – LISHA
PO Box 476 - 88049-900 - Florianópolis, SC, Brazil
mateus@lisha.ufsc.br

Antônio Augusto Fröhlich
Federal University of Santa Catarina – UFSC
Laboratory for Software and Hardware Integration – LISHA
PO Box 476 - 88049-900 - Florianópolis, SC, Brazil
guto@lisha.ufsc.br

ABSTRACT

Around 90% of the total encoding time of raw video into H.264 is spent in block-matching, a stage of Motion Estimation. In this paper we combine two block-matching optimizations that yield significant performance improvements on Motion Estimation: 4:1 subsampling by macroblock and truncation of the two less significant bits per sample. When applied to the JM Reference Encoder, our strategy showed an average speedup of 2.64 times in total encoding time with a small loss of quality (less than 0.5 dB).

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Compression (Coding)

General Terms

Algorithms

Keywords

Video encoding, Motion Estimation, H.264

1. INTRODUCTION

Motion Estimation (ME) is a technique used to explore temporal redundancy in video sequences during compression. Temporal redundancy arises from the fact that neighboring frames very often share similar pixel regions. Therefore the goal of Motion Estimation is to *estimate* the shifting of such similar regions across neighbor frames, thus enabling them to be differentially encoded. In block-based ME, the displacement of similar regions is represented by *motion vectors*, which are computed by the *Block-Matching Algorithms*. Standards like the ISO MPEG series and the ITU-T H26x are examples of encoders that use ME to improve compression ratios in output video streams [19].

Around 90% of the total encoding time in a H.264 encoder is spent in the Motion Estimation stage [7], [21]. Consequently, Motion Estimation optimization is a relevant issue for H.264 and video encoding in general. Since the appearance of ME, many strategies were proposed for its optimization. The *Block-Matching Algorithm* (BMA), which searches for similar blocks and generates the motion vectors, is mainly responsible for ME being so time consuming. Therefore one strategy for optimizing BMA is the *fast-search*, which looks only in specific points of the search window, while a similar block is being searched. Another strategy is to perform ME hierarchically, computing motion vectors for a specific frame region, and refining them in each level, which is known as *multi-resolution* motion estimation. Other strategies look into finding parallelism in BMAs, in order to run ME stages simultaneously. For all strategies there are also hardware implementations, based on optimized functional units (such as vector operations) or based on replication of functional units, to explore parallelism.

Block-Matching Algorithms using fast-search improve time performance of ME, but they can find suboptimal motion vectors because they do not search in all positions of the search window. Multi-resolution ME works with different resolutions of one frame, successively refining the found motion vectors. This increases the ME time if the search is performed sequentially as in [9] or demands for replicated hardware functional units, as in [10]. Similarly, parallel and hardware implementations come at the cost of replicated or dedicated functional units.

The search for new methods to optimize motion estimation is an important issue to enable the construction of real-time H.264 encoders and the implementations of encoders in devices with less computational resources, since those optimizations aim to reduce ME complexity. Two other strategies used to reduce ME complexity are macroblock subsampling and sample truncation during the block-matching. Macroblock subsampling means that, during block-matching, just some samples of a macroblock (or macroblock partition) are taken into consideration. Sample truncation is a technique for pixel decimation, where a sample can be represented only by its most significant bits (MSB).

In this paper we have proposed and evaluated the combination of macroblock subsampling and sample truncation in order to speed up motion estimation, keeping a good video quality - which is measured by the Peak Signal to Noise Ratio (PSNR). We aimed to evaluate these two strategies in an optimal algorithm (i.e. which always finds the best motion vectors), therefore we have used a full-search algorithm instead of a fast-search one. However these strategies can also be applied in fast-search algorithms since they are orthogonal to the block-matching algorithm used.

WebMedia'11: Proceedings of the 17th Brazilian Symposium on Multimedia and the Web. Full Papers.
October 3 -6, 2011, Florianópolis, SC, Brazil.
ISSN 2175-9642.
SBC - Brazilian Computer Society

The next sections of this article are organized as follows: section 2 makes an overview of issues and strategies for ME optimization; section 3 presents the proposed ME optimizations; section 4 describes the implementation of the proposed optimization for the JM H.264 Reference Encoder. Section 5 evaluates the results obtained from the proposal's evaluation using the JM encoder. The final considerations are presented in section 6.

2. STRATEGIES FOR ME OPTIMIZATION

There are two major goals in motion estimation optimization: to improve the compression rate and to reduce the total encoding time. Improving the compression rate is achieved by finding the best possible motion vectors, which means motion vectors that will generate the smallest residual difference during the motion compensation (MC). Reducing the total encoding time is achieved by finding the motion vectors in the smallest possible period of time. Several tools in H.264 are used to find the best possible motion vectors; besides looking in all positions of a search window (i.e. full-search), it is possible to search in several reference frames (backwards or forwards), and it is possible to perform block-matching using sub-pel precision (half and quarter of a pel) [19]. Finding the best motion vectors, very often, goes against finding the motion vectors more quickly. In this work we focus on motion estimation optimizations which aim to reduce the total encoding time, therefore we are not going into the details of techniques for finding the best motion vectors possible, but they can be found in [6], [16], [12], and [8]. It is important to notice that all techniques for ME optimization must take into consideration keeping the video quality of the generated bitstream.

There are many strategies to optimize the time performance of motion estimation: fast-search algorithms, macroblock subsampling, sample truncation, multi-resolution ME, subsampled motion-field estimation, and parallel and hardware implementations of algorithms.

Fast-search algorithms are block-matching algorithms that look only in specific positions of the search window, during block-matching [13], [15], [23], [14], [20], [3]. Search window is the region of the reference frame where a macroblock partition similar to the current block is searched. The motion vectors correspondent to the match with the lower *motion cost* are chosen. The main drawback of this approach is that, since some positions of search window are discarded, it is possible to find suboptimal motion vectors.

Two other strategies to optimize ME during block-matching are macroblock subsampling and sample truncation. Macroblock subsampling means taking into consideration only some samples of a macroblock, or macroblock partition, while the matching for a specific position of the search window is been made. Sample truncation is performed by ignoring the least significant bits of a sample. These strategies have been used separately in [11] (subsampling) and in [5], and [10] (truncation). In this work we have combined these strategies to optimize ME. Section 3 explains these strategies in detail.

Multi-resolution motion estimation is the strategy where the motion vectors are computed for distinct resolutions of one same frame. Motion vectors computed in a more coarse level can be successively refined until the finest level (higher resolution). If the search is performed sequentially as in [9], the time of ME can be increased due to the dependencies between distinct levels. On the other hand, if the search is executed in parallel for each resolution level, as in [10], hardware functional units need to be replicated. A similar technique is *subsampled motion-field estimation* [11]. This technique is based on the assumption that motion vectors of neighboring blocks are intent to be similar thus, for each block, only a set of motion vectors (motion-field) is computed and the others are interpolated.

Other strategies for optimizing motion estimation are based on finding parallelism in ME stages, especially in the block-matching algorithms, in order to execute them simultaneously. These parallel strategies commonly have hardware implementations. The Sum of Absolute Differences (SAD) is a metric of error used in the block-matching algorithms, that is frequently parallelized using functional units in hardware [10], and [2]. Hardware implementations of shared buffers for reference picture data is also common [2], [22]. Although these solutions can achieve the best time performance, they come at the cost of using replicated or dedicated functional units.

3. PROPOSED OPTIMIZATIONS

We envisioned two major opportunities to optimize matching operations within block-based Motion Estimation algorithms: macroblock subsampling and sample truncation. Both were applied in the block-matching algorithm to speedup the whole process of motion estimation. More specifically they are applied in the SAD computation, which is the error metric used to compute the distortion term in the Lagrangian cost function used in H.264 [17, 4].

Equation 1 shows the bi-dimensional SAD used for block-matching algorithms in video coding. C represents the current NxN block that is being searched in the reference frame and R is the NxN block of the reference where the BMA is looked into.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \tag{1}$$

The Lagrangian cost function used in H.264 motion estimation is shown in equation 2. The *Rate* term of the equation is the product between the lagrangian multiplier and the estimated number of bits necessary to encode the motion vectors and the motion predictors. The distortion term (SAD) is computed as shown in equation 1.

$$MotionCost = Rate + SAD \tag{2}$$

Equations 3 and 4 details the *Rate* calculation. The lagrangian multiplier (λ_{motion}) depends on the quantization parameter (QP) used in the video encoding. As higher the QP is, more data is discarded on video encoding thus, increasing the compression ratio. *bits* is a mapping function converting the vector difference of $(MV - MVP)$ into the estimated bits. MV is the motion vector obtained for the motion estimation, and MVP is the motion predictor. Motion predictors are derived from previously computed motion vectors of the neighboring blocks in the past.

$$Rate = \lambda_{motion} \cdot bits(MV - MVP) \tag{3}$$

$$\lambda_{motion} = \sqrt{0.85 \times 2^{\frac{QP}{3}}} \tag{4}$$

3.1 Subsampling

Subsampling aims at reducing the time needed to compute a matching criterion, like SAD, by applying the corresponding operation only to a subset of the pixels in a macroblock. The optimization relies on the assumption that neighboring pixels in a macroblock should have similar values (spatial redundancy) and thus computing the matching criterion from a regular subset should yield comparable results. This assumption is confirmed by the experiments presented in section 4. Figure 1 illustrates the process of 4:1 subsampling in a 16x16 macroblock. The dark pixels are the pixels taken into consideration. Instead of comparing 256 pixels (16x16), the algorithm compares only 64 pixels (8x8), speeding up the process.

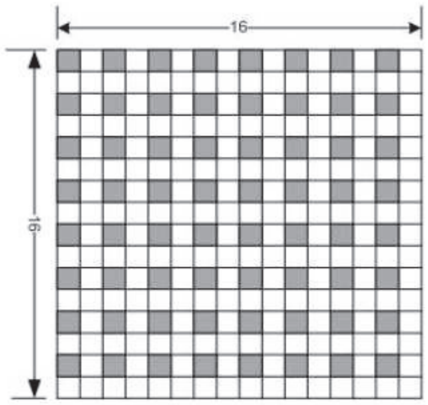


Figure 1: Illustration of 4:1 subsampling

3.2 Truncation

Similarly to subsampling, truncation also aims at reducing the computational cost of block matching by eliminating redundant data. As basis for this optimization we took the assumption that the main information of a sample is stored in its most significant bits (MSB) thus ignoring the least significant bits (LSB) should generate a small noise in the encoded video sequence. The main information of a pixel in YCbCr format means the light variation in the luminance channel (Y) and color variation in the chrominance channels (Cb and Cr).

Our experiments with the JM H.264 Reference Encoder demonstrate that truncation can also improve software-implemented algorithms, because the Full-Search block-matching algorithm in JM features an early termination mechanism. The motion cost in H.264 is calculated using the Lagrangian rate-distortion cost function shown in 2. The Full-Search block-matching algorithm in JM first computes the rate component of the motion cost for a specific position in the search window. If the rate component value is greater than or equal to the previous minimum motion cost (cost of a previous compared block), the distortion component of motion cost (given by the SAD function) is not computed, and the motion cost computation is abbreviated for that block. When bits are truncated, numbers that once were slightly different from each other become equal. Consequently, the number of occurrences in which the previous minimum cost and the rate component of the current cost are equal increases and the SAD does not need to be computed. The speedup is then caused by the reduction in the frequency of calls to the SAD function.

4. IMPLEMENTATION IN THE JM H.264 REFERENCE ENCODER

We implemented the optimizations described in the previous section in the JM H.264 Reference Encoder [18] to assess their impact on the final video quality. Being the reference software implementation for the H.264 standard, JM includes both encoding and decoding functionality. Our experiments were carried out with JM version 14.2 and consisted of encoding a set of reference video sequences with the original encoder and subsequently with the modified one, therefore enabling us to measure variations in the Peak Signal to Noise Ratio (PSNR) and in the encoding time.

For inter macroblock modes in H.264 (i.e. modes related to the Motion Estimation), the motion cost for chrominance components derives from the motion cost for luminance components [4]. Consequently the PSNR for chrominance components derives from the PSNR for luminance components. For this reason, in this paper we focus on the PSNR variation of the luminance component.

The video sequences chosen for the experiments encompass different resolutions and frame rates: Foreman QCIF, Foreman CIF, Mobile & Calendar CIF, City 4CIF, Stockholm 720p, Sunflower 1080p, and Crown Run 1080p. These are all raw video sequences in YCbCr format. Their key features are summarized in table 1.

Subsampling was implemented in JM as proposed in section 3. Samples are taken from macroblocks following a regular, symmetrical pattern. For instance, a 4:1 subsample is obtained by having the iteration loop to skip columns and rows as shown in Figure 1. Truncation was done in JM by assigning zero to the least significant bits of each sample, since actual register and memory truncation are meaningless to software implementations. Both techniques, subsampling and truncation, were implemented during the SAD computation. The block-matching algorithm used was Full-Search.

Modifications to JM were implemented so that subsampling and truncation could be evaluated both in separation and together, thus rendering more detailed results. We tested 2:1, 4:1, and 8:1 subsampling. And we tested truncation on 8-bit samples varying from 1 to 7 LSB. We took the cases which had generated a small noise (0.5 dB) and tested their combinations.

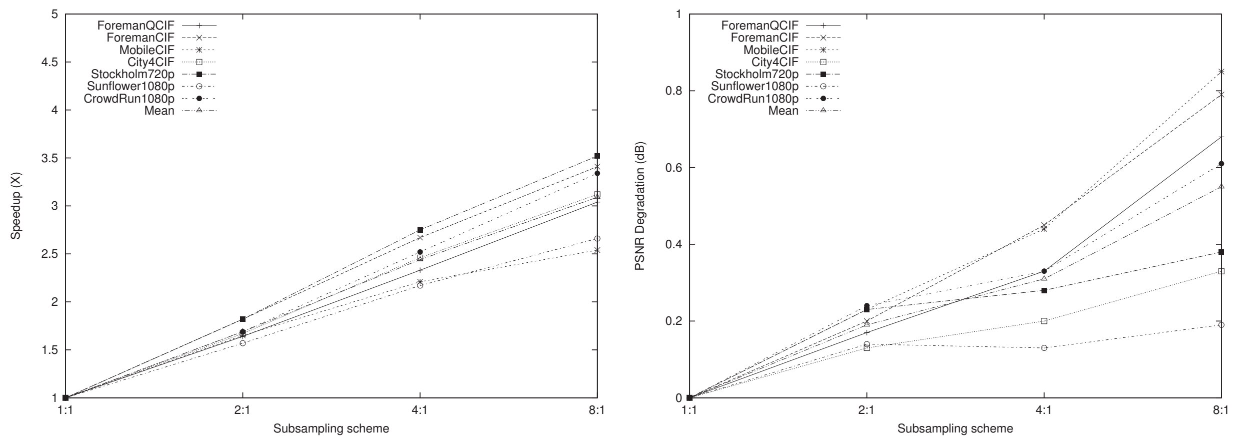
5. RESULT ANALYSIS

Our results show that subsampling and LSB truncation both improve video encoding performance while degrading quality, either deployed separately or in combination. Both parameters: performance and quality vary proportionally to the extending of the applied optimizations as can be seen in figures 2 to 3b. Figure 2a shows the performance improvement for the chosen video sequences as a function of a growing subsampling factor, while Figure 2b shows the corresponding quality loss measured as a decrease in PSNR. Figures 3a and 3b illustrate the facts for truncation.

The PSNR degradation as computed as the absolute PNSR difference between the original encoder and the optimized ones. In order to measure the PNSR variation we have fixed the encoding bit-rate for all video sequences. We have chosen values of bit-rate which result in a compression ratio

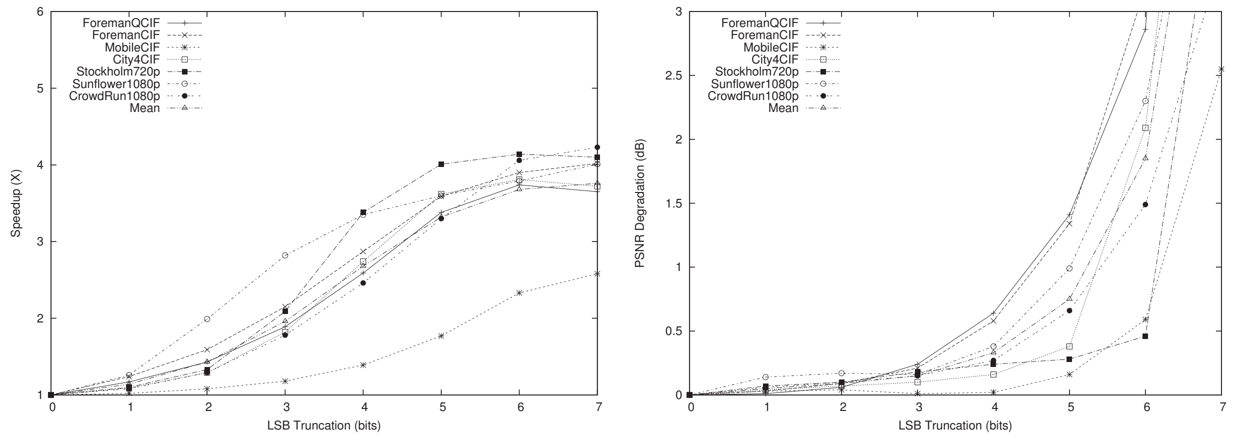
Sequence	Resolution	Frame rate (fps)	Duration (s)
Foreman	QCIF (176 x 144)	30	10
Foreman	CIF (352 x 288)	30	10
Mobile	CIF (352 x 288)	30	10
City	4CIF (704 x 576)	60	10
Stockholm	720p (1280 x 720)	60	10
Sunflower	1080p (1920 x 1080)	25	20
Crown Run	1080p (1920 x 1080)	50	10

Table 1: Video sequences used in the experiments



(a) (b)

Figure 2: Impact of subsampling on encoding performance (a) and encoding quality (b)



(a) (b)

Figure 3: Impact of truncation on encoding performance (a) and encoding quality (b)

Sequence	Bit-rate (Mbit/s)
ForemanQCIF	1
ForemanCIF	2.4
MobileCIF	2.4
City4CIF	30
Stockholm720p	66.8
Sunflower1080p	66.8
Crown Run1080p	133.6

Table 2: Bit-rate used in the experiments

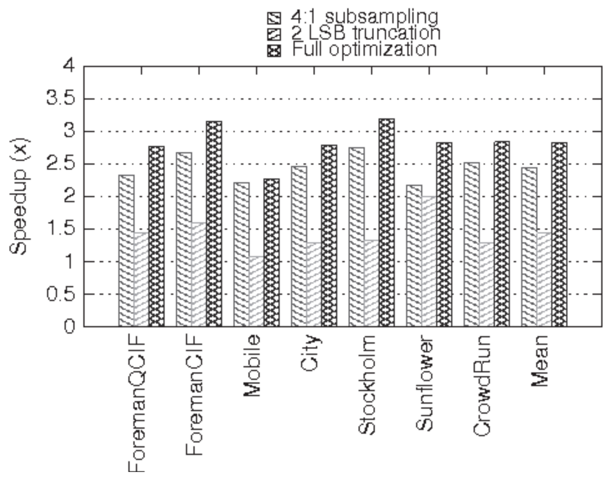
around 10:1. Table 2 shows the bit-rate used for each encoded sequence.

We have noticed that subsampling has more influence in the encoding speedup than truncation. We have a mean speedup of 2.07 times for subsampling against 1.80 times from truncation, considering all subsampling and truncation strategies tested and taking into account the *valid cases* (i.e. cases where the quality loss is less than 0.5 dB).

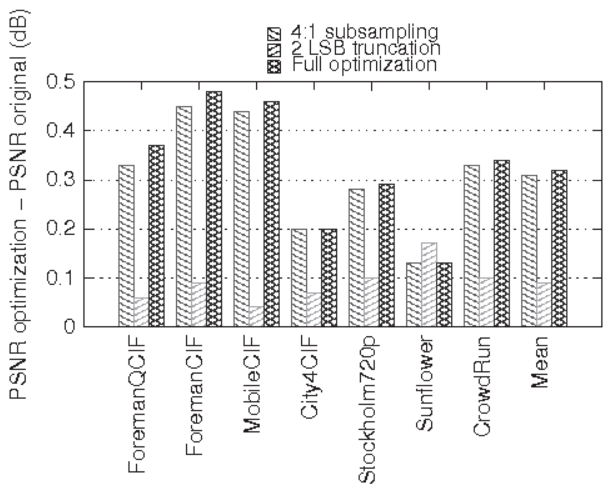
Subsampling generates more noise than truncation. Again, considering all subsampling and truncation strategies and the *valid cases* we got a mean quality loss of 0.27 dB for subsampling and a mean quality loss of 0.12 dB for truncation. Although considering the mean and the worst case only 8:1 subsampling exceeds 0.5 dB of noise. In the mean case for truncation the noise becomes bigger than 0.5 dB only after 4 LSB bit truncation, and after 3 bit truncation considering the worst case.

From the obtained data, we noticed that a combination of 4:1 subsampling and 2 LSB truncation would result in the bigger speedup while keeping relatively small quality loss. Indeed, we looked into the obtained data for a combination that would cause no more than 0.5dB decrease in PSNR and has the highest speedup. Truncation of 2 LSB incurs in an average PSNR reduction of less than 0.1dB, while 4:1 subsampling has a higher toll on quality, about 0.31db. Subsampling, however, is the major speedup source for JM, since 3/4 of the data in each macroblock is simply ignored (as explained in section 3, speedups from truncation on software ME implementations arise from early termination mechanisms). This leads to speedups of up to 3.18 times (for the Stockholm sequence). The charts in Figures 4a and 4b detail the combination of 4:1 subsampling and 2 LSB truncation. The average speedup for the combined approach was of 2.64 times, with an average impact on PSNR of less than 0.5db.

In order to evaluate in details the behavior of our proposal for distinct values of encoding bit-rate, we have used the BD-PSNR (Bjøntegaard Delta PSNR) metric using the following values of QP (Quantization Parameter): 16,20,24,28, as described in [1]. It is important to evaluate quality (PSNR) for distinct bit-rates to test whether the approach can be used in distinct scenarios of application. Figure 5a shows the rate-distortions (RD) curves using the original JM encoder and the optimized encoder using our proposal. The video sequence used for this curves was Crowd Run which has a high quantity of motion and is a HD (1080p) sequence. Lower values of bit-rate are obtained for higher values of QP since using higher values for QP more data is discarded and there is an increasing on the compression ratio. The two curves are very near from each other which indicates the proposal presents a good rate-distortion performance for all



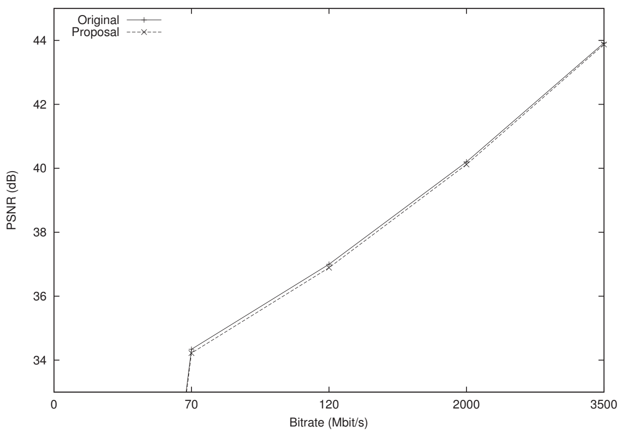
(a)



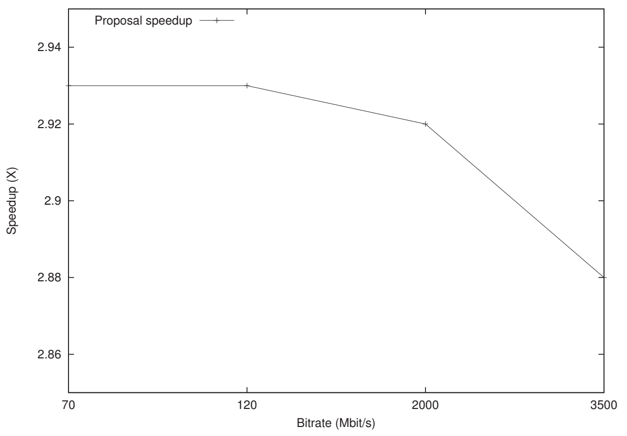
(b)

Figure 4: Speedup (a) and PSNR degradation (b) due to optimization on JM

the evaluated bit-rates. We have evaluated also the speedup obtained in the encoding time while using our proposal for the same QP values we used for BD-PSNR. Figure 5b shows the obtained values. A speedup of near 3 times is obtained for all bit-rate values.



(a)



(b)

Figure 5: RD curve (a) and speedup vs bit-rate (b) of 1080p sequence

The quality loss is small when compared to the speedup mainly because of the spatial redundancy of photographic video sequences. For the case of subsampling one would think that decreasing the number of samples taken into consideration in the block matching, would let escape similar samples, worsening ME precision and increasing the generated noise. However because of the spatial redundancy this increase of noise remains small. The truncation is applied to all samples of the video sequence, thus it does not affect the quality aspect of the matching algorithm. The noise generated by truncation is homogeneous and does not influence in ME.

6. CONCLUSIONS

In this article, we combine two complementary optimizations for block-based Motion Estimation in video encoding: macroblock subsampling and truncation of the two least significant bits per sample. The proposed optimizations were developed around the Full-Search block-matching algorithm and the SAD matching criterion, yielding a block-matching engine that was implemented in the JM Reference Encoder.

The evaluation of the proposed optimizations in the JM H.264 Reference Encoder assess ME quality and good performance gains. It also demonstrates the best combination between subsampling and truncation. The combination of 4:1 subsampling and 2 LSB truncation presents a quality loss of less than 0.5dB for all considered sequences and an average speedup of 2.64.

Since the proposed strategies were implemented in a full-search algorithm, the developed ME engine avoids the problem of finding suboptimal motion vectors originated by skipping positions of the search window. However, the proposed approach is independent of the BMA used, so it can also be applied in fast-search BMAs.

The proposed approach was applied in a single-resolution motion estimation, then there is no need of extra hardware resources to compute ME for distinct levels of resolution. Either way, it is possible to adapt subsampling and truncation together in any of the levels of resolution in a multi-resolution ME strategy.

The proposed method is essentially algorithmic decreasing ME complexity and freeing hardware resources. Although it is possible to develop parallel versions of the proposed method and implement those developing dedicated hardware architectures.

The combination of the proposed approach with other methods for ME optimization could be a matter for further evaluations

7. ACKNOWLEDGMENTS

We would like to thank Ronaldo Husemann and Prof. Valter Roesler for the prolific discussions about motion estimation optimization.

8. REFERENCES

[1] G. Bjøntegaard. Calculation of average PSNR differences between RD-curves, Mar. 2001. Avaliable at: http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc. Access date: March 31, 2011.

[2] H. Chang, S. Kim, S. Lee, and K. Cho. High-performance architecture of h.264 integer-pixel motion estimation ip for real-time 1080hd video codec. pages 419 –422, sep. 2009.

[3] L.-G. Chen, W.-T. Chen, Y.-S. Jehng, and T.-D. Chiuch. An efficient parallel motion estimation algorithm for digital image processing. *Circuits and Systems for Video Technology, IEEE Transactions on*, 1(4):378 –385, dec. 1991.

[4] T. D. H. Du. Macroblock mode decision for h.264. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 167–172, New York, NY, USA, 2005. ACM.

[5] Z.-L. He, C.-Y. Tsui, K.-K. Chan, and M. L. Liou. Low-power vlsi design for motion estimation using adaptive pixel truncation. *IEEE Trans. Circuits Syst. Video Techn.*, 10(5):669–678, 2000.

[6] Y.-W. Huang, B.-Y. Hsieh, S.-Y. Chien, S.-Y. Ma, and L.-G. Chen. Analysis and complexity reduction of

multiple reference frames motion estimation in h.264/avc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(4):507 – 522, apr. 2006.

[7] X. Li, E. Li, and Y.-K. Chen. Fast multi-frame motion estimation algorithm with adaptive search strategies in h.264. volume 3, pages iii – 369–72 vol.3, may. 2004.

[8] X. Li, E. Li, and Y.-K. Chen. Fast multi-frame motion estimation algorithm with adaptive search strategies in h.264. volume 3, pages iii – 369–72 vol.3, may. 2004.

[9] C.-C. Lin, Y.-K. Lin, and T.-S. Chang. A fast algorithm and its architecture for motion estimation in mpeg-4 avc/h.264 video coding. pages 1248 –1251, dec. 2006.

[10] C.-C. Lin, Y.-K. Lin, and T.-S. Chang. Pmrme: A parallel multi-resolution motion estimation algorithm and architecture for hdtv sized h.264 video coding. volume 2, pages II–385 –II–388, apr. 2007.

[11] B. Liu and A. Zaccarin. New fast algorithms for the estimation of block motion vectors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 3(2):148 –157, apr. 1993.

[12] W. Ma, S. Yang, C. Pei, L. Gao, and Y. Lin. Fast algorithm for multi-reference and variable block size motion estimation in h.264/avc. In *ICIMCS '09: Proceedings of the First International Conference on Internet Multimedia Computing and Service*, pages 151–154, New York, NY, USA, 2009. ACM.

[13] S. Ning-ning, F. Chao, and X. Xu. An effective three-step search algorithm for motion estimation. volume 1, pages 400 –403, aug. 2009.

[14] A. T. Oscar, O. C. Au, and M. L. Liou. Predictive motion vector field adaptive search technique (pmvfast) - enhancing block based motion estimation. In *in the Optimization Model 1.0," in ISO/IEC JTC1/SC29/WG11 MPEG2000/M6194*, pages 883–892, 2001.

[15] L.-M. Po and W.-C. Ma. A novel four-step search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3):313 –317, jun. 1996.

[16] Y. Su and M.-T. Sun. Fast multiple reference frame motion estimation for h.264/avc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(3):447 –452, mar. 2006.

[17] G. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *Signal Processing Magazine, IEEE*, 15(6):74 –90, nov 1998.

[18] K. Sühring. H.264/avc jm reference software, 2011. Avaliable at: http://iphome.hhi.de/suehring/tml/. Access date: February 07, 2011.

[19] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, August 2003.

[20] H.-M. Wong, O. Au, C.-W. Ho, and S.-K. Yip. Enhanced predictive motion vector field adaptive search technique (e-pmvfast)-based on future mv prediction. page 4 pp., jul. 2005.

[21] L. Yang, K. Yu, J. Li, and S. Li. Prediction-based directional fractional pixel motion estimation for h.264 video coding. In *IEEE International Conference on Acoustics Speech Signal Processing*, pages 901–904, 2005.

[22] H. Yin, H. Jia, H. Qi, X. Ji, X. Xie, and W. Gao. A hardware-efficient multi-resolution block matching algorithm and its vlsi architecture for high definition mpeg-like video encoders. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(9):1242 –1254, sep. 2010.

[23] S. Zhu, J. Tian, X. Shen, and K. Belloulata. A new cross-diamond search algorithm for fast block motion estimation. pages 1581 –1584, nov. 2009.