FAST FULL SEARCH BLOCK MOTION ESTIMATION FOR H.264/AVC WITH MULTILEVEL SUCCESSIVE ELIMINATION ALGORITHM

Tuukka Toivonen and Janne Heikkilä

Machine Vision Group
Infotech Oulu and Department of Electrical and Information Engineering
P. O. Box 4500, FIN-90014 University of Oulu, Finland
{tuukkat,jth}@ee.oulu.fi

ABSTRACT

We modify the multilevel successive elimination algorithm (MSEA) to make it compatible with the motion estimation required for the H.264/AVC video coding standard. The algorithm must be changed to support the multiple block sizes and the criterion to be minimized has to be rate-distortion-based to allow efficient encoding. We use MSEA for eliminating the smallest 4×4 -pixel blocks and either derive lower bound or exact criterion for larger blocks based on the small block bounds or criterions, correspondingly. The resulting algorithm needs 60%–70% less time than the original motion estimation in the reference encoder but will still yield practically equivalent video quality and bit rate.

1. INTRODUCTION

Most video coding standards, including MPEG-1/2/4, H.261, H.263, and H.264/AVC, use block motion estimation (ME) and compensation for removing temporal redundancy [1]. This is one of the most important part of a video encoder, and newer standards achieve better video quality at constant bitrate by allowing subdivision of 16×16 -pixel macroblocks (MB) into smaller blocks. The encoder may then select whether to use large blocks and only a few motion vectors (MV), or more accurate ME with smaller blocks but with more motion vectors to transmit. For example, MPEG-4 allows subdividing the MBs into four 8×8 -pixel blocks. The newest and most efficient coding standard H.264/AVC allows subdividing the MBs into 16×8, 8×16, or 8×8 -pixel blocks, and when the smallest size is chosen, the block may be further subdivided in a tree-like fashion into 4×8, 8×4, or 4×4 -pixel blocks [2, 3]. A good quality encoder must then handle 41 different blocks with 7 block sizes, as shown in Fig. 1.

Since there is a vast number of possibilities for selecting the subdivision of blocks, the encoder must intelligently decide what block subdivision to use and which MV to use for each block. Earlier encoders typically computed the sum of absolute differences (SAD) between the current block and candidate blocks and selected simply the MV yielding the least distortion. However, this often will not give the best image quality for a given bit rate, because it may select long motion vectors that need many bits to transmit. It also does not help determining how the subdivision should be performed, because the smallest blocks will always minimize the distortion, even if the multiple MVs may use larger amount of bits and increase the bit rate. Newer encoders perform rate-distortion (RD) optimization [4], typically using the Lagrangian multiplier technique, by minimizing the criterion

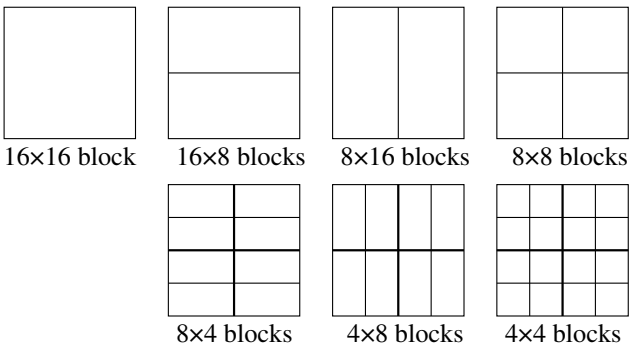


Fig. 1. Block sizes for motion estimation with H.264/AVC.

$$C(x, y) = \mathrm{SAD}(x, y) + \lambda B(x - x_0, y - y_0) \quad (1)$$

where (x, y) is a candidate MV, (x_0, y_0) is the predicted MV for the block, $\mathrm{SAD}(x, y)$ is the image distortion, $B(x - x_0, y - y_0)$ is proportional to the number of bits required for encoding the difference between the candidate MV and the predicted MV, and λ is the Lagrangian multiplier. This criterion corresponds to the total cost required for coding a block. The encoder may choose a good subdivision strategy by simply summing the block criterions in each configuration and choosing the subdivision giving the minimum total cost.

0-7803-8554-3/04/$20.00

Motion estimation is computationally the most demanding part of a typical video encoder. The multiple block sizes available in newer standards only increase computation. Many fast motion estimation methods have been developed, but many of them compute the criterion only at a few possible MVs. This will degrade video quality, because the best MV might not be found. Other alternatives are the fast full search methods [1], such as the partial distortion elimination (PDE) and the multilevel successive elimination algorithms (MSEA) [5], which have been suggested to lessen the computational burden without degrading video quality. A lot of research has been carried out to theoretically improve the MSEA, but the practical implementation has been considered relatively little. It is important that a given ME algorithm can be applied in practical encoders [6], taking into account such aspects as RD-optimization and multiple block sizes, or it will remain unused. In this paper we investigate the application of the MSEA to the H.264/AVC video encoder.

2. IMPLEMENTATION OF MSEA FOR H.264/AVC

The H.264 predicts the MV for each block by generally using the median of the MVs in three blocks left, above, and above right of the current block. Only the difference between the predicted MV and the real MV is transmitted in the bitstream. The predicted MV is available only after the MVs for the three surrounding blocks have been already estimated. Therefore, the ME for the blocks should be done in the same order as the blocks are coded in the bitstream. Otherwise the MV predictor (x_0, y_0) can not be accurately calculated, and the RD-based criterion in Eq. (1) can not be evaluated.

On the other hand, if the SADs of small blocks are summed to obtain the SADs and criterion values for larger blocks, the MV for all blocks in an MB should be estimated simultaneously, or otherwise a large amount of memory is required for storing the SADs for all candidate MVs for small blocks, before the motion estimation for larger blocks begins. Our approach is to save memory and estimate all MVs simultaneously in an MB, and instead of a true predicted MV we approximate it using either the predicted MV for the whole 16×16 -pixel MB or simply a zero MV.

Our algorithm works as follows:

1. In the beginning, for each MB, we compute two-level norm pyramids for the current MB and the search area, for 2×2 and 4×4 -pixel blocks.

2. Then we compute the exact SADs and RD-based criterion values for the smallest blocks, and using these results, also for larger blocks. The location of these values correspond to the approximated predicted MV and the values provide a tight upper bound for the best MV match, speeding up the MSEA at later stages. Then, starting from the search area center, we perform a spiral-shaped search.

3. For each candidate MV, we first loop through the sixteen smallest 4×4 -pixel blocks. The absolute difference between a block norm and the corresponding norm in the search area is computed. The cost λB for encoding the MV is added to the difference, which then forms a lower bound for the RD-based criterion for the candidate MV [7]. If the lower bound is larger than the smallest criterion value found for the block so far, the block can be eliminated.

4. If the block was not eliminated, it is subdivided into four 2×2 -pixel blocks, and the sum of the corresponding four absolute norm differences is calculated. By adding the MV cost into this, a tighter lower bound is obtained and the block elimination is tried again. If the block fails to eliminate even now, its exact SAD and RD-criterion value are computed.

5. Either the block SAD lower bound or the exact SAD value is stored and a flag is set to indicate which one the stored value is.

6. Next each of the rest blocks is processed. Summing the stored values for two smaller blocks, either a lower bound or exact SAD for a larger block is obtained. Using a logical operation with the flags for the two smaller blocks, we can decide whether the SAD for the larger block is exact or only a lower bound. The RD-criterion value or its lower bound is again obtained by adding the cost from the MV, and the block can be eliminated if the bound is larger than the best match so far for the block.

7. If the block was not eliminated, and only its criterion lower bound is known, we traverse recursively back to smaller blocks, computing the exact SAD value for the blocks. With the exact RD-criterion value now available, we can determine whether the MV corresponds to the best match for the block so far. We need to traverse only relatively rarely, because the lower bound for the larger blocks is formed from lower bounds for the smaller blocks, and it is therefore quite tight. The stored SADs and the flags for the smaller blocks are updated, eliminating the possibility that they need to be recomputed later.

8. If the exact criterion value for a block was calculated and was better than the best match so far, the best MV for the current block is updated. In any case, the algorithm then proceeds to the next candidate MV, jumping back to step 3.

3. EXPERIMENTAL RESULTS

The MSEA motion estimation algorithm was implemented into H.264 reference encoder version JM 7.3 [8] and tested with various sequences using Athlon-based computer running at 2 GHz. To speed up encoder operation, some optional features were turned off: B-frames, CABAC, and multiple reference frames. Twelve different 140-frame sequences with QCIF size and seven 250-frame sequences with CIF size were encoded with nine different quantization parameter (QP) values between 20 and 36. The frame rate was 25 Hz. The sequences were encoded using the two algorithms existing originally in the reference encoder: the simple and slow method using the PDE algorithm, and a faster method that computes SAD of larger blocks by summing the SADs of smaller blocks. The reference encoder was modified to include also the MSEA-based method. It was written in plain C language, without any platform-dependant optimizations, and compiled with GNU Compiler Collection (gcc) version 2.95.4 using high optimization levels. Two versions of the MSEA were benchmarked: the first approximates the predicted MV for all blocks with the same prediction MV which is used for the whole 16×16 -pixel MB. The second version simply approximates the predicted MV with a zero MV.

Fig. 2a displays the average time required for encoding each sequence. Of the MSEA-based methods only one is shown, because the other method yielded practically the same speed. Both of the original methods are significantly slower than the MSEA-based method, which is 23%–25% faster than the original slow method. Fig. 2b includes only time spent for the ME, showing the average number of clock cycles needed for one MB. In this case MSEA requires 60%–70% less time than the original slow ME and 48% less time than even the original fast method.

The relative bitrates are shown in Fig. 3. All of the results are practically identical for both of the original ME methods and they are drawn only for the other. The MSEA-based methods are also nearly the same compared to the original ME methods with all QCIF-sequences, increasing the bitrate less than 1% in average. For most CIF-sequences the bitrate increase is negligible, but for a few sequences requiring very long motion vectors, the MSEA-based method performs significantly worse. For some sports sequences, the bitrate was observed to be about 15% larger in the worst case. Due to this, in average the bitrate is increased 1–4% when using the predictive MSEA-based method.

4. CONCLUSIONS

MSEA is a fast full search ME method, which is shown to decrease computation even up to 97% without decreasing quality [5]. However, efficient implementation into newer standards such as H.264 is not straightforward. In this paper we discussed these problems and suggested methods to overcome them. In particular, we applied MSEA to the smallest block sizes (4×4 pixels with H.264) and computed lower bounds for larger blocks from lower bounds for the small blocks. Most efficiently the MSEA can be applied when estimating the motion for all blocks in an MB simultaneously. However, this makes it impossible to use correct MV predictors. Therefore we developed two methods for approximating them, either by using the MV predictor for the whole MB or using a zero MV predictor. Experiments showed that the MSEA decreases computation in ME by 60%–70%, while the video quality and bit rate stays essentially the same in most cases.

5. REFERENCES

[1] T. Toivonen, *Number Theoretic Transform -Based Block Motion Estimation*, Master's Thesis, University of Oulu, Finland, 2002.

[2] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, March 2003.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

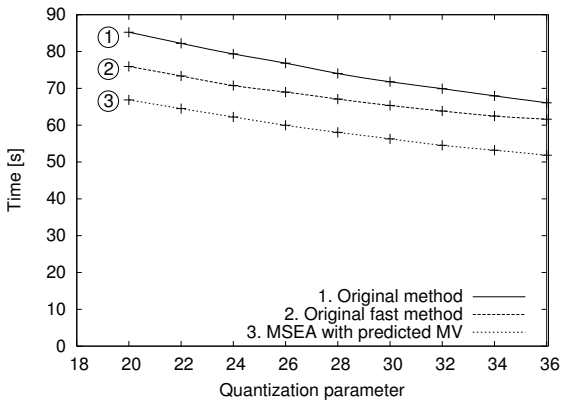
[4] G. J. Sullivan and T. Wiegand, "Rate-distortion Optimization for Video Compression," *IEEE Signal Processing Mag.*, vol. 15, no. 6, pp. 74–90, 1998.

[5] X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation," *IEEE Trans. Image Processing*, vol. 9, no. 3, pp. 501–504, 2000.

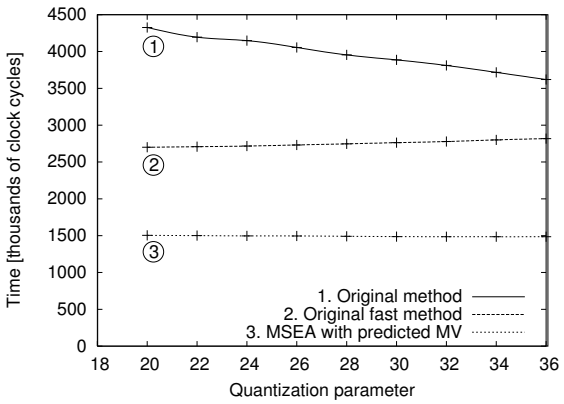
[6] Y. Noguchi, J. Furukawa, and H. Kiya, "A Fast Full Search Block Matching Algorithm for MPEG-4 Video," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 61–65, 1999.

[7] M. Z. Coban and R. M. Mersereau, "Computationally Efficient Exhaustive Search Algorithm for Rate-constrained Motion Estimation," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 101–104, 1997.

[8] K. Suehring et al. (2004, January). JM Reference Software version 7.3. International Telecommunications Union. [Online]. Available: http://bs.hhi.de/~suehring/tml/download/jm73.zip

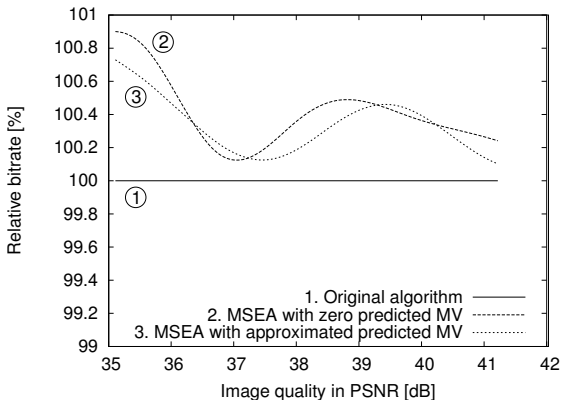


(a) Total video coding time

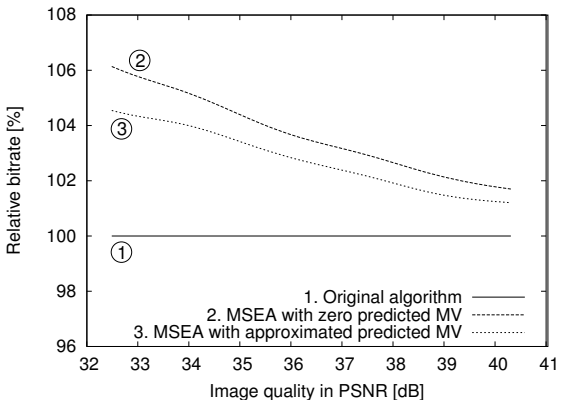


(b) Time used for integer pixel motion estimation

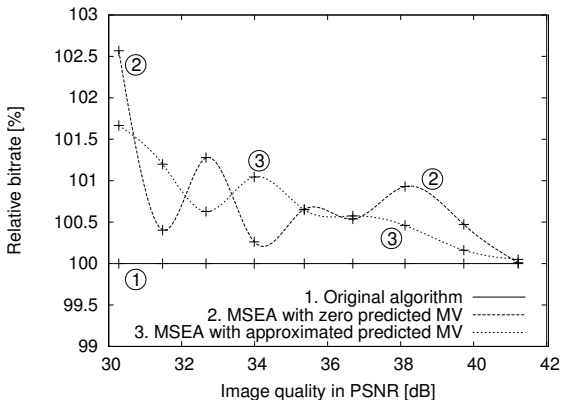
Fig. 2. Average execution time at different quantization parameter values.



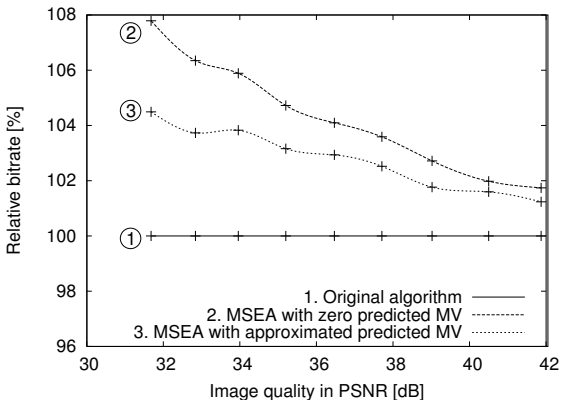
(a) Average relative bitrate with QCIF sequences



(b) Average relative bitrate with CIF sequences



(c) Relative bitrate with QCIF-sized Foreman sequence



(d) Relative bitrate with CIF-sized Foreman sequence

Fig. 3. Relative bitrate at different image qualities.