SAD REUSE IN HIERARCHICAL MOTION ESTIMATION FOR THE H.264 ENCODER

Hasan F. Ates

Telecommunications Engineering
Sabanci University
Tuzla, Istanbul, 34956 TURKEY
hasanates@sabanciuniv.edu

Yucel Altunbasak

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250 USA
yucel@ece.gatech.edu

ABSTRACT

The adoption of multiple macroblock partitions with variable block sizes is one of the main reasons behind the superior coding efficiency of H.264 video coding standard. Unfortunately, in motion estimation phase, repeating Sum of Absolute Difference (SAD) calculations for every possible block size incurs a heavy computational cost for the encoder. In this paper, in order to reduce the encoder complexity, we propose a hierarchical block matching based motion estimation algorithm that uses a common set of SAD computations for motion estimation of different block sizes. Based on the hierarchical prediction and the median motion vector predictor of H.264, the algorithm defines a limited set of candidate vectors; and the optimal motion vectors for all partitions are chosen from this common set. Simulation results show that hierarchical estimation with SAD reusing reduces the total computations by a factor of 17.6 with slight loss in coding efficiency.

1. INTRODUCTION

H.264 video coding standard has been shown to provide considerably higher coding efficiency over such previous standards as H.263 and MPEG-4 Visual [1]. One of the novelties contributing to H.264's superior performance is a rich set of coding modes to choose from for each macroblock [2]. These modes allow the encoder to try different macroblock partitions, multiple reference frames and inter/intra prediction methods in order to find a rate-distortion optimal coding strategy for each macroblock. Unfortunately choosing among these features incurs a substantial increase in the computational complexity of the encoder.

Motion estimation (ME) contributes a significant portion to the complexity of H.264 encoder. The use of different block sizes for various macroblock partitions multiplies the amount of computations. For different inter-coding modes, macroblocks (16×16 pixel) can be divided into two 16×8 or two 8×16 or four 8×8 subblocks. Each 8×8 block can be further partitioned into 8×4, 4×8 and 4×4 subblocks. Rate-distortion optimal mode decision requires estimating the motion vectors of each such subblock for all possible macroblock partitions. Since there exist 7 block sizes (16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4), a naive full-search method will require about 7 times more computations than the single block type case.

Full-search method (FSM) can benefit from the fact that, if the same search range is used for all block sizes, Sum of Absolute Differences (SADs) computed for small block partitions (e.g. 4×4) can be reused to construct the SADs for larger block partitions (e.g. 8×8 or 16×16). That is, for a given motion vector (MV), SADs calculated for 4×4 subblocks can be added up to find the SADs of larger subblocks. This will require additional memory to store the computed SADs, but decrease the number of computations almost to the level of single block type case.

Nevertheless, FSM is too complex for real-time implementation, and fast algorithms are needed to further reduce the computational cost. In literature, the algorithms designed for H.264 [3, 4, 5, 6, 7] tend to focus on each block size separately, without considering SAD reusability. For instance, [3] uses special search patterns around block specific MV predictions for estimating the optimal MVs of each macroblock partition. Since the predictions and the optimizations are carried out almost independently for each subblock, the SADs computed for different partitions will correspond to different sets of candidate MVs. As a result, SAD reuse will rarely be possible, which ends up wasting the computational gains of performing a fast search for each subblock.

In this paper, we propose a hierarchical ME algorithm that enables SAD reuse to reduce ME complexity. For each 4×4 subblock, SADs are computed and stored at small groups of search positions that are determined by the hierarchical MV prediction process. For larger block sizes, SAD computations are carried out, whenever possible, using only the stored values. In other words, no extra search is performed for larger partitions. In order to minimize the loss of quality, the hierarchical estimation method is designed in a way that makes good MV predictions for all subblocks of different sizes. This is essential for the performance of the algorithm, since we want to find the optimal solution without making excessive searches for each partition. The median MV predictor of H.264 is used to further improve the prediction accuracy. Details of the algorithm are provided in section 2.

SAD reusing is also a crucial technique for achieving a real-time hardware implementable H.264 encoder. When we consider that the encoder will dedicate a certain amount of clock cycles to ME module, worst-case complexity of a ME method becomes more important than its average complexity. In other words, the ME module needs to have enough clock cycles to finish the motion vector search in all possible cases. Worst-case complexity is generally ignored in existing ME methods. Some algorithms [3] try to minimize average number of computations by using various early stopping criteria during SAD calculations. Others [4] use simple measures to predict whether small macroblock partitions can be optimal or not, and, if predicted not, they do not perform ME for these partitions. Despite improving average complexity, these techniques won't have much better worst-case complexity than FSM with SAD reuse. The use of hierarchical estimation to

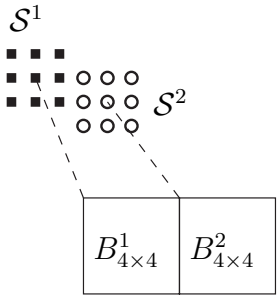

Fig. 1. The set of search locations for two 4×4 subblocks.

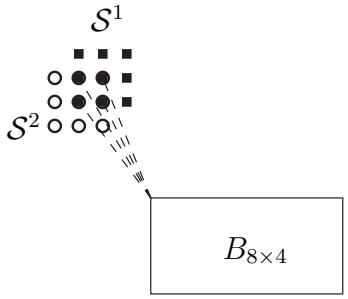

Fig. 2. The set of search locations for $B_{8\times4}$.

support SAD reuse enables our algorithm to decrease the number of worst-case computations significantly, since all partitions are investigated without extra computational cost.

Section 2 describes the details of the algorithm. Section 3 discusses the worst-case complexity and compare it with FSM with reuse. The simulation results in Section 4 show that there is little quality loss, despite a major reduction in complexity. We conclude the paper in Section 5.

2. HIERARCHICAL MOTION ESTIMATION WITH SAD REUSE

2.1. SAD Reuse for Macroblock Partitions

For rate-distortion optimal mode decision in H.264, the optimal MV for each subblock is selected as the one that minimizes the Lagrangian cost:

$$\mathcal{J}(\mathbf{d}) = \text{SAD}_{B_{m\times n}}(\mathbf{d}) + \lambda_M R(\mathbf{d} - \mathbf{p}_{med}), \quad (1)$$

where $B_{m\times n}$ is a subblock of size $m\times n$, $(m,n) \in \{(4,4),(4,8),(8,4),(8,8),(16,8),(8,16),(16,16)\}$, and

$$\text{SAD}_{B_{m\times n}}(\mathbf{d}) = \sum_{x=1,y=1}^{m,n} |c(x,y) - r(x+d_x, y+d_y)|, \quad (2)$$

where $\mathbf{d} = (d_x, d_y)$ is the MV, c and r are current and reference frames, respectively. For ease of notation, we assume the origin of the axes coincides with the lower left corner of the block. Here, λ_M is the lagrange multiplier for ME, $R(\mathbf{d} - \mathbf{p}_{med})$ specifies the bitrate spent for coding MV difference information, and $\mathbf{p}_{med}$ is the MV prediction used by H.264 during the coding process. If multiple reference frames are used, the cost of coding the reference frame index is added to the total Lagrangian cost.

SAD reuse is based on the simple observation that, for a given MV $\mathbf{d} = (d_x, d_y)$, SAD of $B_{m\times n}$ can be decomposed into the SADs of its 4×4 subblocks:

$$\text{SAD}_{B_{m\times n}}(\mathbf{d}) = \sum_{\substack{k=1,\\ l=1}}^{m/4,n/4} \sum_{\substack{x=4k-3,\\ y=4l-3}}^{4k,4l} |c(x,y) - r(x+d_x, y+d_y)|. \quad (3)$$

Since all summations on the right are evaluated at the same MV $\mathbf{d}$, computing $\text{SAD}_{B_{m\times n}}(\mathbf{d})$ requires computing the SADs of all its 4×4 subblocks for MV $\mathbf{d}$. More specifically, we define SAD reusability as follows: given a macroblock $B_{16\times16}$, each of its 4×4 subblocks, $B^i_{4\times4}$, $1 \le i \le 16$, is assigned a set of MV candidates, $\mathcal{S}_i$. In other words, $\text{SAD}_{B^i_{4\times4}}(\mathbf{d})$ is computed only if $\mathbf{d} \in \mathcal{S}_i$. Suppose $B_{m'\times n'} \subset B_{m\times n}$ means that $B_{m'\times n'}$ is a subblock of $B_{m\times n}$. Then, $\text{SAD}_{B_{m\times n}}(\mathbf{d})$ is available for block $B_{m\times n}$ if and only if $\mathbf{d} \in \mathcal{S}_i$ for all $B^i_{4\times4} \subset B_{m\times n}$.

Figure 1 and 2 illustrate how SAD reuse is done for a block $B_{8\times4}$. In these figures, the sets $\mathcal{S}_i$ contain not the search locations but the MVs defined as the difference between the search locations and the positions of the subblocks. In figure 2, the search locations of $B^2_{4\times4}$ are shifted by 4 pixels to the left to compensate for the distance between the upper left corners of the two 4×4 subblocks. Figure 2 shows the common MVs and corresponding search positions that allow for SAD reuse for block $B^i_{8\times4}$.

The design of the sets $\mathcal{S}_i$ for each 4×4 subblock, $B^i_{4\times4}$, constitutes the most important part of our motion estimation strategy. Over-populated sets, such as in FSM, have many common vectors and SAD reuse will be frequent, but this increases the amount of computations. Under-populated sets reduce complexity, but, without enough common vectors, MV search might be too limited or even not be possible for larger macroblock partitions.

Designing the sets $\mathcal{S}_i$ can be viewed as designing good MV predictors for the subblocks. In its most general form, our ME method follows these steps:

1. Predict optimal MVs of some of the blocks $B_{m\times n}$ (The next section explains this step in detail). Let's call one such prediction as $\mathbf{p}_{m\times n}$.
2. Assign $\mathbf{p}_{m\times n}$ as a candidate vector to all subblocks of $B_{m\times n}$. That is, $\mathbf{p}_{m\times n} \in \mathcal{S}_i$ for all $B^i_{4\times4} \subset B_{m\times n}$.
3. In order to refine the prediction, include a small number of neighboring MVs of $\mathbf{p}_{m\times n}$ into the sets $\mathcal{S}_i$ as well.
4. Compute and store $\text{SAD}_{B^i_{4\times4}}(\mathbf{d})$ for each $\mathbf{d}\in\mathcal{S}_i$, $1\le i\le16$.
5. For each subblock $B_{m\times n}$, compute the Lagrangian cost of MV $\mathbf{d}$ if $\text{SAD}_{B_{m\times n}}(\mathbf{d})$ is available (see the definiton of SAD reusability). Choose the vector with the minimum cost as the optimal MV for this subblock.

Contrary to other predictive algorithms, we don't want to make a separate MV prediction for each subblock, since this could overpopulate the sets $\mathcal{S}_i$. That's why, our predictions need to consider the motion of all macroblock partitions. Otherwise, there might be significant performance loss. For instance, if the predictions are selected based on 4×4 subblocks only, than the sets $\mathcal{S}_i$ defined by these vectors might not share common vectors that are needed for performing MV search at larger block sizes. On the other hand, if the predictions are based on the full 16×16 macroblock, than SAD reuse is guaranteed. But this might lead to oversmoothing the MV field, and we might lose coding efficiency when the optimal solution is actually the 4×4 partition with sharply varying MVs for each subblock. In other words, the predicted MVs should be close to the optimal solution not just for a single partition but probably for all partitions. Therefore, while choosing our predictors, we need to incorporate the motion information from all block sizes.

Finding MV predictors that are good enough for all partitions is a challenging problem. For a single block $B_{m\times n}$, there exist several alternatives: the MVs of spatial and temporal neighboring

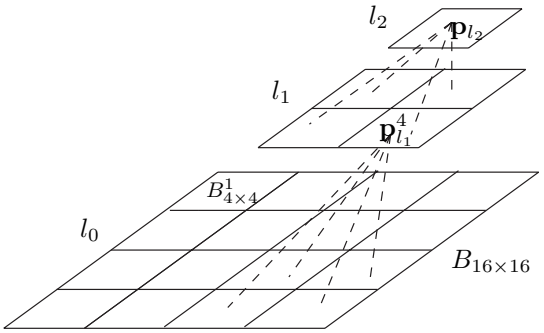

Fig. 3. Hierarchical motion estimation.

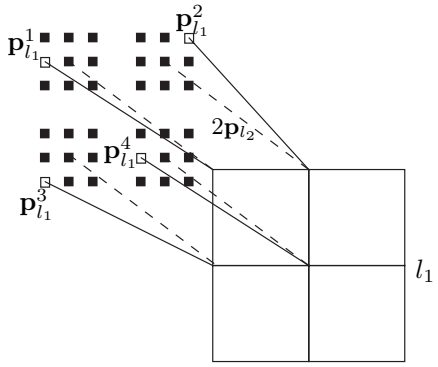

Fig. 4. MV refinement at level l_1.

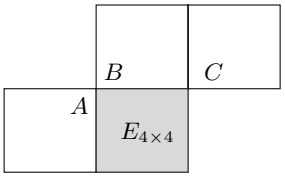

Fig. 5. Median predictor.

blocks, the median predictor of H.264, prediction using a hierarchical method or a fixed pattern, such as diamond or hexagonal search, etc. Since we don't want to over-populate our search sets, we have to find a prediction method that yields a uniformly good solution for all block sizes and partitions. The next section describes our approach to this problem.

2.2. Hierarchical MV Predictor

For achieving uniformly good MV predictions, we start our search with the full macroblock, make an initial prediction for it and then refine this prediction for the smaller subblocks. We propose a modified hierarchical block matching method to implement this top-down approach. Hierarchical ME is low in computational cost and provides robust MV predictions, especially at large block sizes. At the top level of the hierarchy, the motion search is performed for the full macroblock. At lower levels, the initial prediction is refined in a way that reflects the varying motions of smaller block partitions. This way, the final predictions can be seen as a compromise between the optimal MVs of different block sizes.

Figure 3 illustrates the idea. The algorithm consists of the following steps:

1. A 3-level pyramid is constructed using local sums of the macroblock pixels. A 4×4 block at level l_2 corresponds to the 16×16 macroblock at level l_0.
2. Full search is performed for this 4×4 block at level l_2, where the search range is $[-R/4, R/4]$ (R is the search range of the FSM).
3. If the MV prediction at level l_2 is $\mathbf{p}_{l_2}$, $2\mathbf{p}_{l_2}$ is used as the search center at level l_1. This time, instead of refining the prediction for the full block, MV search is carried out for each of the four 4×4 blocks of level l_1 (The search range is $[-R/8, R/8]$ around $2\mathbf{p}_{l_2}$). The search at level l_1 yields 4 predictions, $\mathbf{p}^j_{l_1}, 1 \le j \le 4$ (see Figure 4).
4. Going from l_1 to l_0, we assign each $2\mathbf{p}^j_{l_1}$ as the MV prediction of the corresponding 8×8 block and all of its subblocks. At the end, the 16×16 macroblock is assigned 4 vectors, one for each of its 8×8 subblock.
5. As discussed in the previous section, the sets $\mathcal{S}_i$ are defined around these predicted vectors.

The optimal vector at level l_2 constitutes a good initial prediction for the macroblock $B_{16\times16}$, when scaled by 4. The subblock refinement at level l_1 is aimed at updating this prediction such that the final 4 predicted vectors reflect the motions of the 8×8 subblocks that might be different from each other. At level l_0, the predicted MVs for all subblocks of all sizes will be refined according to the sets $\mathcal{S}_i$ and the SAD reusability. Note that, if the sets $\mathcal{S}_i$ defined by these 4 vectors don't provide a common set of MV candidates, then SAD computation and MV search at level l_0 won't be possible for the full 16×16 block. In such cases, we claim that a single MV is not likely to provide good motion compensation for the macroblock and 16×16 partition doesn't have to be considered to find the optimal solution. Similar arguments can be made for the 16×8 and 8×16 subblocks, when the 2 corresponding vectors are much different. These claims have to be validated by simulations.

For avoiding loss of coding efficiency, we need to make sure we have a good estimate of the R-D optimal MV. Therefore MV refinement at each level of the pyramid is performed by minimizing the Lagrangian cost, $\mathcal{J}(2^k\mathbf{d}) = \text{SAD}^{l_k}(\mathbf{d}) + \lambda_M R(2^k\mathbf{d} - \mathbf{p}_{med})$, where SAD^{l_k} is the SAD computed at level l_k of the pyramid.

2.3. Median MV Predictor

It turns out that assigning a single prediction vector to an 8×8 block and all of its subblocks is too restrictive for finding the optimal 4×4 and similar small-sized partitions. In other words, despite the refinement of level l_1, the hierarchical method is still oversmoothing the motion field. Therefore, we use the median predictor as an additional predictor for each 4×4 subblock. Figure 5 shows the neighbors used in median prediction. Then,

$$\mathbf{p}^E_{med} = \text{Median}(MV_A, MV_B, MV_C). \quad (4)$$

Median prediction is performed exactly as it is done for 4x4 blocks in H.264 [8]; if the causal neighbors of the current block are already coded, their optimal MVs are used in the prediction.

Median predictor is needed to improve the algorithm's success at finding the optimal 4×4 partition. As long as the computed SADs are reusable, this prediction will be useful for searching for other partitions as well.

2.4. MV Refinement Around Predicted MVs

We use both the hierarchical prediction and the median prediction to define the search locations $\mathcal{S}_i$. For each block $B^i_{4\times4}$, let's call these vectors as $\mathbf{p}^i_{hme}$ and $\mathbf{p}^i_{med}$. Then, the final $\mathcal{S}_i$ is defined as (see Figure 6):

$$\begin{aligned}\mathcal{S}_i \quad = \quad & \{(dx_1, dx_2) : |dx_l - p^i_{hme}(l)| \le R/8 \quad \text{or} \\ & |dx_l - p^i_{med}(l)| \le R/8, l = 1, 2\}. \quad (5)\end{aligned}$$

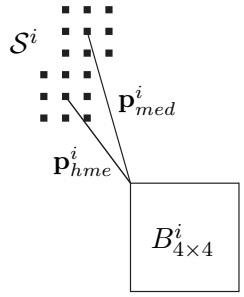

Fig. 6. The set of search locations for block $B^i_{4\times4}$ ($R = 8$).

For each subblock, the Lagrangian cost is computed whenever SAD reuse is possible. The macroblock partition and the MVs that minimize the total cost are chosen as the optimal solution.

3. COMPLEXITY ANALYSIS

We analyze the worst-case complexity of our algorithm and compare it to FSM with SAD reuse, for the case $R = 16$. The total number of additions and subtractions is chosen as the measure of complexity. For FSM, 31 operations are performed for 33×33 search locations of each 4×4 block. There are 25 subblocks of size larger than 4×4, and each SAD reuse requires a single addition (SAD of two 4×4 blocks are added to compute SAD of a 8×4 block and so forth). The total computations are equal to 567369.

For our algorithm, $\mathbf{p}^i_{hme}$ and $\mathbf{p}^i_{med}$ produce 25 search locations each, and for worst-case analysis we assume these two sets to be disjoint. For 4×4 blocks 31×16 operations, and for SAD reuse 25 additions are needed for each of these 50 locations. The pyramid level l_1 contributes $25\times31\times4$ operations, and l_2 adds 81×31 more operations. The pyramid is constructed with 480 additions. The total number is 32141. This corresponds to a speed-up factor of 17.6.

This is a remarkable improvement when one considers that the algorithm is rather simple in its current form and can be further developed in many different ways, e.g. using diamond or hexagonal search patterns instead of rectangular. If average complexity is chosen as the performance measure, techniques such as adaptive early termination of SAD computation [3] will increase this speed-up factor to much higher levels.

4. SIMULATIONS AND RESULTS

The simulations are performed for $R = 16$, using video sequences carphone (QCIF), foreman (CIF), mobile (SIF) and flowergarden (SIF) at 30fps. All frames except the first one are coded as P-frames. Two reference frames are allowed. The CAVLC entropy coder is used, with quantization parameter values of $QP = 32, 35, 38$. For comparison to FSM, average PSNR loss in dB and percentage change in bitrate are reported in Table 1.

The results confirm the potential of our algorithm. At equal bitrates, PSNR loss is less than 0.2 dB for all the tested sequences. The loss is mainly due to inaccurate predictions for small partitions such as 4×4. Median predictor helps to alleviate the problem, and other predictors might also be considered to improve estimation accuracy.

The algorithm performs especially well when the motion field is smooth and the optimal partitions are usually larger than or equal to 8×8. This confirms that the hierarchical predictor works as expected and succeeds at predicting the motion of the 8×8 subblocks. As mentioned before, it is not always possible to perform MV search for subblocks of size 16×8, 8×16 and 16×16. However,

Table 1. Performance comparison with FSM.

	δPSNR (dB)	δbitrate(%)
carphone (QCIF)	-0.09	1.1
foreman (CIF)	-0.11	3.9
mobile (SIF)	-0.03	0.5
garden (SIF)	-0.02	1.0

this leads to only a marginal loss in PSNR, confirming our assumption that, in such cases, optimal coding mode is rarely found among these large partitions.

5. CONCLUSION AND FUTURE WORK

The use of small sized partitions makes a major contribution to the coding gain of H.264 standard. Motion estimation has to check all partitions in order to find the optimal block sizes and their MVs. When SAD values computed for small blocks can be reused for larger blocks, the worst-case complexity of the ME module is significantly reduced. In order to avoid any loss in coding efficiency, it is necessary to design good MV predictors that are suitable for SAD reuse. In this paper, we proposed a modified hierarchical prediction method, and showed that a high ME speed-up is achievable with little performance loss.

Our current work is focused on better MV prediction methods. One alternative is to make multiple predictions at levels l_2 and l_1 of the hierarchy, and then choose the best one at level l_0. A more adaptive approach could be to decide during prediction which block sizes are more likely to be optimal and then gear the MV prediction towards these block sizes. This might be possible by performing a simple analysis of the motion activity within the macroblock.

6. REFERENCES

[1] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. SUllivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, July 2003.

[2] I.G. Richardson, *H.264 and MPEG-4 Video Compression*, Wiley, West Sussex, England, 2003.

[3] H. Cheong and A.M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. IEEE Int. Conf. Multimedia and Expo*, Baltimore, MD, July 2003, vol. 3, pp. 517–20.

[4] P. Yin, H. Cheong, A.M. Tourapis, and J. Boyce, "Fast mode decision and motion estimation for JVT/H.264," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sept. 2003, vol. 3, pp. 14–17.

[5] J. Zhang, Y. He, S. Yang, and Y. Zhong, "Performance and complexity joint optimization for H.264 video encoding," in *Proc. Int. Symposium Circuits Systems*, May 2003, pp. 25–28.

[6] K-K. Ma and G. Qiu, "Unequal-arm adaptive rood pattern search for fast block-matching motion estimation in the JVT/H.26L," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sept. 2003.

[7] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT," in *JVT-G016.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG*, 7th Meeting: Pattaya II, Thailand, 7-14 March 2003.

[8] ITU-T, *Prepublished Recommendation H.264*, 2003.

[9] http://bs.hhi.de/ suehring/tml/index.htm, "JM reference software version 7.4," .