IMPROVED UNSYMMETRIC-CROSS MULTI-HEXAGON-GRID SEARCH ALGORITHM FOR FAST BLOCK MOTION ESTIMATION

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 develop a set of new motion estimation (ME) algorithms based mainly on Unsymmetric-Cross Multi-Hexagon-Grid Search (UMH). The original algorithms are improved by applying the Successive Elimination Algorithm (SEA) and subsampling the image blocks while computing the matching criterion. We also improve SEA by adding a small constant to the lower bound before trying to eliminate the current checking point. The motion compensated results stay in most cases similar to the original UMH algorithm, while computation is decreased by up to 95%. The new algorithms outperform in both image quality and computational efficiency other well-known fast ME algorithms such as Three Step Search (TSS), Diamond Search (DS), and Hexagon-Based Search (HEXBS).

Index Terms— Motion compensation, video coding

1. INTRODUCTION

Most video coding standards, including MPEG-1/2/4, H.261, H.263, and H.264/AVC, use block motion estimation (BME) and compensation (MC) for removing temporal redundancy [1]. This is one of the most important parts of a video encoder giving a very large reduction in bit rate, but it is also computationally one of the most intensive parts of an encoder.

The motion vector (MV) for a block is computed by minimizing the value of a matching criterion. The most common criterion is the rate-distortion optimized (RD) criterion

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

where $B(\Delta x, \Delta y)$ denotes the number of bits required for coding the difference between the candidate MV (x, y) and the MV predictor (x_0, y_0). The parameter λ is the Lagrangian multiplier and it adjusts the tradeoff between bit rate and video distortion. SAD is the sum of absolute differences between a reference block $\mathbf{R}(x, y)$ and the current block $\mathbf{B}$. The block size is commonly 16×16 pixels, although with newer standards, such as H.264, also smaller blocks are available, down to size of 4×4 pixels.

The computation of SAD can be hastened by approximating it by subsampling the blocks and skipping over some of the sum terms [2]. This is an efficient method at small subsampling factors, but deteriorates the result significantly at larger factors, which limits applicability of this method.

By far the most popular way to optimize BME is at relatively high level. Motion estimation can be performed by checking only a minor part of all candidate MVs (x, y). Several fast search strategies have been developed, including Three Step Search (TSS), Diamond Search (DS) [3] and Hexagon-Based Search (HEXBS) [4] which are extremely fast search algorithms, but give worse image quality than full search. On the other hand, Unsymmetric-Cross Multi-Hexagon-Grid Search (UMHexagonS or UMH) [5] gives a very good quality, but is significantly slower than TSS, DS, or HEXBS.

Another technique to decrease the number of checking points is to compute a lower bound for the criterion value in the beginning of every checking point. If the lower bound turns out to be larger than the criterion value at the best match so far, the exact criterion value needs not to be evaluated. This algorithm is called Successive Elimination Algorithm (SEA) [6].

In this paper, we develop new motion estimation algorithms, using UMH as a starting point due to its excellent motion estimation quality. A modified SEA is applied in an efficient manner, to reduce the relatively high computation requirements, and subsampling and filtering are used to further reduce computation. The resulting algorithm has very little computation—less than DS—while still giving almost similar results to the original UMH or full search and superior image quality compared to DS and HEXBS.

This work has been partly funded by the Academy of Finland (project n:o 110751).

2. THE IMPROVED UMH ALGORITHM

2.1. UMH and other ME algorithms

The UMH algorithm is described in [5] and reviewed here for completeness. First, an unsymmetrical-cross search is made, shown as circles in Fig. 1. This step can be justified because either the horizontal Δx or vertical MV component Δy will be zero at the cross pattern checking points and the RD criterion has often a minimum there. The spacing between checking points is two and there are twice as many points horizontally than vertically. In our experiments, we used 17 points horizontally and 9 points vertically, corresponding to search window size of ± 16 points.

Starting from the best point found in the cross search step, next an uneven multi-hexagon-grid search is made, denoted with squares in the figure. The 16-point hexagon pattern (16-HP) used in the grid search has more points at the left and right edges. The pattern is scaled to various sizes (up to 4 times with search window of ± 16 points) and the matching criterion evaluated at the corresponding points.

In the last step, a modified HEXBS is used to refine the best MV found in the multi-hexagon-grid stage. The 6-point hexagon (denoted with triangles) is used repeatedly until the best MV is at the center of the hexagon. Then, the point is refined using small diamond search (SDS) which checks points with a small diamond pattern (denoted with crosses) until the minimum has been found. In the original HEXBS, the small diamond pattern is applied only once.

Another popular ME method is DS: it is similar to HEXBS but uses a large diamond (Fig. 1) in the first stage. DS requires more steps than HEXBS to reach the minimum in diagonal directions. In all ME algorithms the search origin is predicted based on median of MVs at blocks left of, above of, and above-right of the current block.

2.2. SEA and layered image representation

The lower bound for the matching criterion (1) used in SEA [6] is

$$|\|\mathbf{R}(x,y)\|_1 - \|\mathbf{B}\|_1| + \lambda B\,(x - x_0, y - y_0) \leq C\,(x,y) \tag{2}$$

where the L_1-norm $\|\cdot\|_1$ denotes absolute sum of values in a block. This lower bound can be made tighter by subdividing the blocks into smaller blocks, computing the norms of the subblocks, and computing SAD between the norms. For example, the reference and current blocks of size 16×16 pixels could be subdivided into four 8×8-pixel blocks, whose norms would be obtained. The absolute differences of norms in the corresponding subblocks would be then computed and summed with 4 subtractions, 4 absolute operations, and 3 additions, forming a new tighter lower bound. By using smaller blocks, more checking points can be eliminated with the lower bound, but computing the lower bound itself requires more operations.

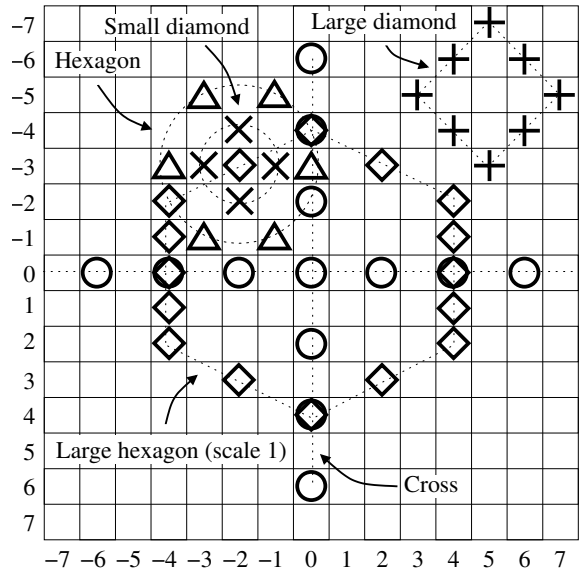

Fig. 1. Motion estimation patterns.

The remaining problem is to find an efficient method for computing the block norms. One method is to use a sliding window technique. However, in our work we compute instead a layered image representation as described in [7]: the first layer contains averaged values of 2×2-pixel blocks, computed from the pixels in the original image. Each of the following layers is computed using the previous layer and contains averaged values (scaled norms) of twice larger blocks. The computation of each layer requires 3 additions and one bit shift per pixel.

2.3. Improved UMH

The UMH algorithm was improved using the following two methods.

Subsampling: In all steps, except in the final SDS, the image blocks were subsampled horizontally and vertically by two. This alone degrades the result relatively much, but the subsampling was performed on images which were also filtered using 2×2-pixel averaging filter: that is, the SAD was computed with subsampled blocks obtained from the first layer of the layered image representation. Since the final SDS step is performed without subsampling, typically very little of quality is lost but computation is decreased to almost a quarter. Note that this subsampling is different from image pyramidal representation: the resolution of the filtered images is the same as in the original images and the search step size can still be only one pixel.

SEA: In all steps, SEA with sums of norms of 8×8 or

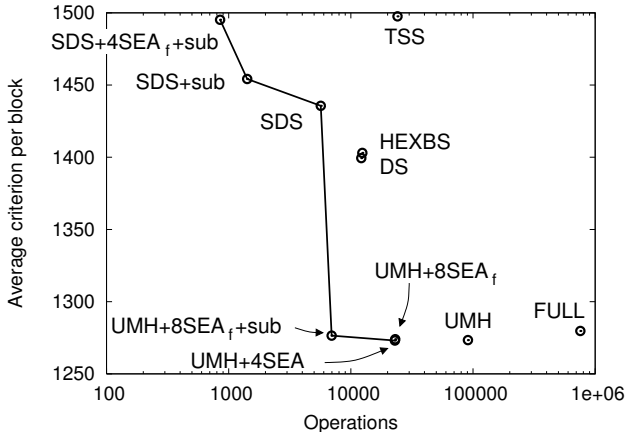

Fig. 2. Selected algorithms with 16×16 block size.

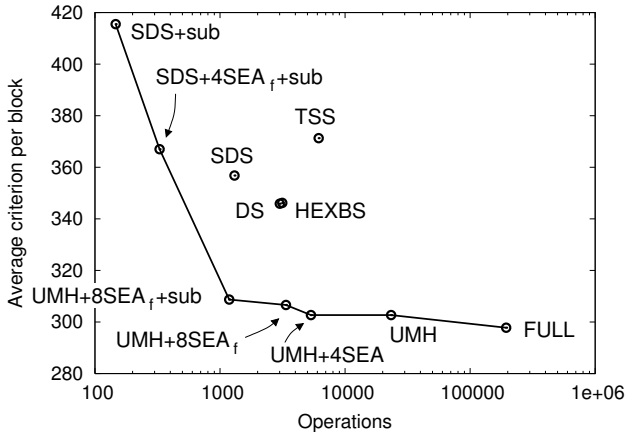

Fig. 3. Selected algorithms with 8×8 block size.

4×4-pixel blocks were used for eliminating checking points in 16×16 and 8×8 blocks. With 4×4 blocks, only norms of 4×4-pixel blocks were used. The following modification was made to the SEA algorithm: before comparison to the best match so far, a small constant f was added to the lower bound. This increased significantly eliminations while decreasing image quality only slightly.

3. EXPERIMENTAL RESULTS

Table 1 displays motion estimation and compensation results for several well-known and the newly developed algorithms. Fourteen standard test sequences were used: *Bridge Close*, *Coastguard*, *Flower*, *Football*, *Foreman*, *Highway*, *Mobile and Calendar*, *Munchener Hall*, *News*, *Paris*, *Stefan*, *Tempete*, *Tourists in San Fransisco*, and *Waterfall*[1]. All

[1]Many of the sequences are available from http://media.xiph.org/.

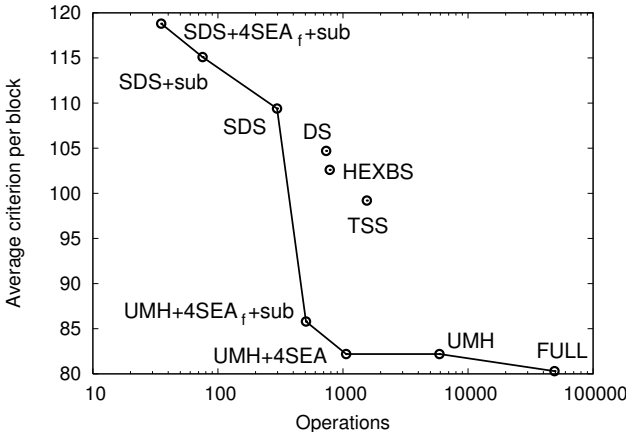

Fig. 4. Selected algorithms with 4×4 block size.

of the sequences were CIF-sized (352×288 pixels) and from each the first 250 frames were used.

Three different block sizes were tested, from 16×16 down to 4×4 pixels. *Cost* column denotes the RD-criterion average value per block, computed according to Eq. (1) with $\lambda = 4.6$ which corresponds to quantization parameter value 26 in an H.264 encoder. *PSNR* shows the average frame distortion. It is displayed for completeness, although the ME algorithms do not try to specifically optimize it. The *Oper.* column shows the average number of subtractions, absolute values, and accumulations per block, needed for calculating the SAD from either the original image data or from the filtered layers for subsampled SAD or a lower bound.

Filter column displays the average number of additions and bit shifts per block needed for computing the image layers, from one to maximum of three layers. It is not included in the operations count, because if the ME is performed for several different block sizes in the same macroblock, the layers need to be computed only once. Also, in many video encoders the layered representation might be needed in any case for other purposes.

The table rows display the results from the various ME algorithms. *FULL* denotes full search: every point in the search range is checked. It is very slow but the results are good. In the eight bottommost rows the results from the new algorithms are displayed: there are six improved UMH and two SDS algorithms.

8SEA designates that SEA is used in all steps of an algorithm and the lower bound is computed from 8×8-pixel subblocks. This subblock size can not be used with the smallest 4×4 blocks, unlike *4SEA*, which computes the lower bounds from 4×4-pixel subblocks. *8SEA*$_f$ and *4SEA*$_f$ signify that a small constant f is added to the lower bound before it is compared to the best previous criterion value. With 16×16 and 8×8-pixel blocks we used $f = 200$ and

Block size	16×16				8×8				4×4			
Algorithm	Cost	PSNR	Oper.	Filter	Cost	PSNR	Oper.	Filter	Cost	PSNR	Oper.	Filter
FULL	1279.7	29.89	756123	0	297.8	31.21	194064	0	80.3	32.31	49141	0
UMH	1273.4	29.85	90789	0	302.7	31.00	23328	0	82.2	31.93	5903	0
DS	1399.4	29.23	12176	0	345.9	30.01	3006	0	104.7	30.07	733	0
HEXBS	1403.0	29.17	12427	0	346.2	29.97	3145	0	102.6	30.13	783	0
SDS	1435.6	29.03	5674	0	356.8	29.73	1304	0	109.4	29.62	297	0
TSS	1497.6	28.33	24074	0	371.3	28.89	6153	0	99.2	29.67	1552	0
UMH+8SEA	1273.3	29.85	30650	3072	302.7	31.00	7602	768	—	—	—	—
UMH+4SEA	1273.1	29.85	22924	2048	302.7	31.00	5337	512	82.2	31.93	1060	128
UMH+8SEA$_f$	1274.0	29.85	23072	3072	306.6	30.92	3368	768	—	—	—	—
UMH+4SEA$_f$	1276.0	29.84	17003	2048	308.4	30.87	3013	512	83.4	31.78	786	64
UMH+8SEA$_f$+sub	1276.5	29.84	6965	3072	308.7	30.86	1185	768	—	—	—	—
UMH+4SEA$_f$+sub	1277.5	29.84	9019	2048	311.0	30.79	1988	512	85.8	31.46	505	128
SDS+sub	1454.2	28.96	1417	1024	367.0	29.51	328	256	115.1	29.07	75	64
SDS+4SEA$_f$+sub	1495.2	28.86	851	2048	415.5	28.87	146	512	118.8	28.80	35	128

Table 1. Tabulated motion compensation results. Smaller cost and number of operations are better.

with 4×4 blocks $f = 50$. The shorthand *sub* denotes that SAD was computed from subsampled image blocks using the first image layer.

In each case the search range per stage was ±16 pixels. Since UMH has several stages, it can actually find longer MVs than the other algorithms. This explains why it outperforms full search in quality with the largest block size.

The best of the new and some prior algorithms are plotted in Figs. 2–4 for clarity. The solid line contains optimal algorithms at some complexity-quality points. All of the plotted UMH-based algorithms are very near the original UMH and full search algorithms in quality. The most significant exceptions are the subsampled versions, which somewhat degrade image quality at the smallest block size. This is quite understandable, since only 2×2 pixels are used after subsampling. The good performance of TSS compared to HEXBS and DS at the smallest block size is quite surprising. However, *UMH+4SEA$_f$+sub* and *UMH+4SEA* algorithms are still superior to all of them.

The new algorithms decrease computation by 92%, 95% (16×16 and 8×8 block sizes with *UMH+8SEA$_f$+sub*), and 92% (4×4 block size with *UMH+4SEA$_f$+sub*) compared to UMH.

4. CONCLUSIONS

We developed new ME algorithms based on UMH by applying SEA and subsampling the blocks while matching. The new algorithms can achieve excellent ME result, near the original UMH algorithm, while decreasing computation by 92–95%. Experiments showed that the new algorithms outperform conventional fast search algorithms such as DS and HEXBS computationally and in image quality.

5. REFERENCES

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits and Systems Mag.*, Q1 2004.

[2] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3. no. 2, Apr. 1993.

[3] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," in *Proc. 1997 Int. Conf. Information, Communications and Signal Processing (ICICS)*, vol. 1, Sep. 912, 1997, pp. 292–296.

[4] C. Zhu, X. Lin, L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12. no. 5, Apr. 2002.

[5] Z. Chen, P. Zhou, Y. He, and Y. Chen, "Fast integer pel and fractional pel motion estimation for JVT," document *JVT-F017, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16*, Dec. 2002.

[6] T. Oh, Y. Kim, W. Hong, S. Ko, "A fast full search motion estimation algorithm using the sum of partial norms," in *IEEE Int. Conf. Consumer Electronics*, 2000, pp. 236–237.

[7] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh, "Fast block matching algorithm based on the winner-update strategy," *IEEE Trans. Image Processing*, vol. 10, no. 8, Aug. 2001.