RATE DISTORTION-BASED MOTION ESTIMATION SEARCH ORDERING FOR RATE-CONSTRAINED SUCCESSIVE ELIMINATION ALGORITHMS

Luc Trudeau, Stéphane Coulombe, Christian Desrosiers

Department of Software and IT Engineering
École de technologie supérieure, Université du Québec
Montréal, Québec, Canada

ABSTRACT

In this paper, we propose a new class of search ordering algorithms to reduce the computational cost of motion estimation in video coding. We show that conventional search orderings, such as spiral search, can weaken the filtering criterion of rate-constrained successive elimination algorithms. Based on this new insight, we derive a new search ordering that takes into account the impact of the rate constraint. Our simulation results demonstrate that, on average, the amount of SAD operations required to encode the tested sequences, is reduced by 2.86%, when compared to the H.264 JM reference software's implementation of spiral search. For sequences with unpredictable motion, this reduction is greater than 5% and can exceed 10% when smaller block partitions are evaluated.

Index Terms— Successive elimination algorithm, motion estimation, H.264, Lagrange multiplier

1. INTRODUCTION

Motion estimation is a predominant task of most modern video encoders. In the H.264 video encoding standard [1], when motion estimation is used to encode a frame, it is performed on every non-overlapping 16×16 block. These blocks are called *current blocks*. Motion estimation is also performed inside the current block, for partitions of sizes $16{\times}16, 16{\times}8, 8{\times}16, 8{\times}8, 8{\times}4, 4{\times}8, 4{\times}4$. Motion estimation consists of finding an optimal matching block in a search area of size $(2W+1) \times (2W+1)$, where W is the full pel length of the search area. The blocks inside this search area are called *candidate blocks*, and the search area can span over multiple reference frames, and up to quarter pel precision is supported.

An exhaustive search algorithm (ESA) will obtain an optimal match by evaluating each candidate block inside the search area. The high computational complexity incurred by evaluating the cost function for all possible candidate blocks allowed in H.264 limits practical applications of ESA in modern encoders. Many algorithms reduce this computational complexity, and can be classified by whether or not they preserve optimality. Algorithms that do not preserve optimality often rely on the assumption of a monotonically increasing match criterion around the location of the optimal candidate block. When this assumption does not hold, accuracy of the motion estimation is reduced, as it will converge to a local minimum. Modern algorithms in this class include zonal search algorithms [2, 3], which first evaluate a set of predictors in order to constrain a local diamond or square search to a very narrow part of the search area.

Optimality preserving algorithms often rely on known inequalities, to avoid computing the cost function of candidate blocks during the search process. Recent algorithms in this class append more efficient filtering criteria to the successive elimination algorithm (SEA) proposed in [4]. For example, [5, 6] in their own way propose the use of partitions inside blocks to improve filtering efficiency.

In [7], Coban and Mersereau modified the SEA to take into account the number of bits required to encode the motion vector of a candidate block, by altering the SEA criterion into a rate-constrained filtering criterion. This alteration is in line with the H.264 joint model reference software [8] where the optimal matching candidate block is the best rate-constrained match. H.264-based SEA algorithms have been proposed in [9, 10].

Another way the filtering criterion can be improved is via the candidate block search ordering used for motion estimation. Spiral search ordering is known to outperform a raster search ordering, and evaluates better candidate blocks earlier in the search process, which in turn improves the filtering criterion and allows more candidate blocks to be skipped. That is why the spiral search ordering is used in many implementations of SEA-based algorithms [6, 7, 9]. This must however not to be confused with SpiralPDE [11], which is a spiral pattern used to sum the elements of a block.

In this paper, we propose a new class of candidate block search ordering algorithms, known as rate-constrained search ordering algorithms. We demonstrate that conven-

This work was funded by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 428942-11). Emails: luc.trudeau.1@ens.etsmtl.ca, {stephane.coulombe, christian.desrosiers}@etsmtl.ca

tional search ordering algorithms, such as raster and spiral search, can impair the filtering criterion of rate-constrained successive elimination algorithms. Rate-constrained search ordering algorithms do not lead to this impairment, making them ideal for rate distortion contexts, like H.264 encoding.

This paper is organized as follows: In section 2, rate-constrained successive elimination is described, and then the motivations for rate-constrained search orderings are explained. We describe the proposed search ordering in section 3, which is derived from the rate-constrained criterion used to filter candidate blocks. In section 4, experimental results for various sequences and discussions of the results are given. Finally, section 5 concludes this paper.

2. RATE-CONSTRAINED SUCCESSIVE ELIMINATION ALGORITHMS

Successive Elimination Algorithms (SEA) are based on the following inequality [4] :

$$|B - C(\mathbf{x}_i, \mathbf{y}_i)| \leqslant \text{SAD}(\mathbf{x}_i, \mathbf{y}_i) \,, \tag{1}$$

where B is sum of the current block pixel values and $C(\mathbf{x}_i, \mathbf{y}_i)$ is the sum of the pixel values of the ith candidate block located at position $(\mathbf{x}_i, \mathbf{y}_i)$ in the search area. On the right hand side, the $\text{SAD}(\mathbf{x}_i, \mathbf{y}_i)$ function returns the sum of the absolute differences between the pixel values of the current block and those of the ith candidate block.

At first glance, the complexity of computing B and $C(\mathbf{x}_i, \mathbf{y}_i)$ might seem equivalent to that of computing the $\text{SAD}(\mathbf{x}_i, \mathbf{y}_i)$ function, but that is not the case, since Li and Salari [4] also proposed an apriori fast block summation technique. During motion estimation, the values of B and $C(\mathbf{x}_i, \mathbf{y}_i)$ are obtained with table lookups, as shown on lines 3 and 11 of Algorithm 1. As explained in [4], the overhead of precaculating these sums is negligible and, overall, reduces computational costs by 85% when compared to ESA.

The filtering criterion works in the following manner: for a given candidate block, if the left-hand side of equation (1), a lower bound for its SAD value, is higher than the current best SAD value of the search area, then this candidate is not optimal. Therefore, the current best SAD value is used as a threshold to decide when to avoid computing the SAD function.

The Rate-Constrained Successive Elimination Algorithms, originally proposed by [7], states that to be optimal, the ith candidate block must satisfy the following inequality:

$$\begin{aligned} |B - C(\mathbf{x}_i, \mathbf{y}_i)| &+ \lambda R(\mathbf{x}_i, \mathbf{y}_i) \\ &\leqslant \text{SAD}(\mathbf{x}^*_{i-1}, \mathbf{y}^*_{i-1}) + \lambda R(\mathbf{x}^*_{i-1}, \mathbf{y}^*_{i-1}) \,, \end{aligned} \tag{2}$$

where λ is the Lagrange multiplier, a trade-off between rate and distortion. Often referred to as rate, the $R(x, y)$ function returns the number of bits required to encode the motion vector of the candidate block at position (x, y). The term $(\mathbf{x}^*_i, \mathbf{y}^*_i)$ is the current best candidate block, having considered the candidate blocks from 0 to $i - 1$ in the scan ordering, and is such that:

$$\begin{aligned} \forall n \in \{0, \ldots, i\} \Big(&\text{SAD}(\mathbf{x}^*_i, \mathbf{y}^*_i) + \lambda R(\mathbf{x}^*_i, \mathbf{y}^*_i) \\ &\leqslant \text{SAD}(\mathbf{x}_n, \mathbf{y}_n) + \lambda R(\mathbf{x}_n, \mathbf{y}_n) \Big) \,. \end{aligned} \tag{3}$$

It is important to note that the best candidate is no longer the lowest SAD value, but the best rate-constrained SAD value.

Algorithm 1 details the implementation of a motion estimation algorithm enhanced with a rate-constrained successive elimination algorithm. More precisely, sumB and sumC are lookup tables for the precalculated block sums, ordering is a lookup table for candidate block ordering and minCost is the cost of the current best candidate block. The filtering operation occurs on line 12, thus allowing the SAD function to be skipped if the condition in (2) is not met.

Algorithm 1 Motion estimation algorithm enhanced with the rate-constrained successive elimination algorithm.

```
 1: function MotionEstimation(block, minCost)
 2:     $i^* \leftarrow -1$        ▷ negative when no better candidate is found
 3:     $B \leftarrow$ sumB[block.$x$][block.$y$]
 4:     for $i \leftarrow 0$ to numCand do
 5:         $x \leftarrow$ ordering[$i$].$x$
 6:         $y \leftarrow$ ordering[$i$].$y$
 7:         cost $\leftarrow \lambda \times R(x, y)$
 8:         if cost $\geqslant$ minCost then
 9:             return minCost
10:         end if
11:         $C \leftarrow$ sumC[block.$x + x$][block.$y + y$]
12:         if $|B - C| <$ minCost $-$ cost then
13:             cost $\leftarrow$ cost $+$ SAD$(x, y)$
14:             if cost $<$ minCost then
15:                 minCost $\leftarrow$ cost
16:                 $i^* \leftarrow i$
17:             end if
18:         end if
19:     end for
20:     return minCost, $i^*$
21: end function
```

One of the issues tackled by Coban and Mersereau [4] is finding the optimal value of λ. This is somewhat resolved by the H.264 standard recommendations [8], as the recommended value of λ can be obtained with the following equation:

$$\lambda_{\text{MOTION}} = \left(w \times 2^{\left(\frac{QP-12}{3}\right)} \right) \,, \tag{4}$$

where w varies from 0.65 to 0.85, depending on the type of frame that is being encoded. This is somewhat a solution to the problem, but is in no way the optimal value of λ.

Equation (2) can be written as follows:

$$\begin{aligned} &|B - C(\mathbf{x}_i, \mathbf{y}_i)| \\ &\quad \leqslant \text{SAD}(\mathbf{x}^*_{i-1}, \mathbf{y}^*_{i-1}) + \lambda(R(\mathbf{x}^*_{i-1}, \mathbf{y}^*_{i-1}) - R(\mathbf{x}_i, \mathbf{y}_i)) \,. \end{aligned} \tag{5}$$

This form of the equation is interesting because of the difference between $R(\mathbf{x}_i^*, \mathbf{y}_i^*)$ and $R(\mathbf{x}_i, \mathbf{y}_i)$. Let ΔR_i be the result of this differentiation for the ith candidate block,

$$\Delta R_i = R(\mathbf{x}_i^*, \mathbf{y}_i^*) - R(\mathbf{x}_i, \mathbf{y}_i)\ . \tag{6}$$

If ΔR_i is positive, then this will increase the filtering threshold in (5) by $\lambda \times \Delta R_i$ and thus weaken the rate constraint on the filtering criterion. This will often occur both in raster and spiral search ordering.

In the next section, we present a new class of search ordering algorithms that do not weaken the rate-constrained filtering criterion.

3. RATE-CONSTRAINED SEARCH ORDERING ALGORITHMS

In light of the fact that the search ordering of candidate blocks can weaken a rate-constrained filtering criterion, we propose a new class of candidate block search ordering algorithms, known as rate-constrained search ordering algorithms. To be classified as such, the ordering of the candidate blocks must adhere to the following rule: the motion vector encoding cost of the current candidate block must be equal to or greater than the preceding candidate block,

$$R(\mathbf{x}_i, \mathbf{y}_i) \geqslant R(\mathbf{x}_{i-1}, \mathbf{y}_{i-1})\ . \tag{7}$$

This guarantees that $\Delta R_i \leqslant 0$, thus never weakening the rate constraint on the filtering criterion.

However, this class of search ordering algorithms is dependent on the encoding scheme used for the motion vectors of candidate blocks. For the H.264 standard, each component of a motion vector is coded using exponential Golomb codes and quarter pixel precision, and thus

$$R(x, y) = G(4x) + G(4y)\ , \tag{8}$$

where the G function returns the number of bits required to code a given value with an exponential Golomb code. This function can be defined as follows:

$$G(x) = 2 \times \lfloor \log_2(2|x|+1) \rfloor + 1\ . \tag{9}$$

In Figure 1, we present the motion vector encoding costs of candidate blocks for a very small part of an H.264 motion estimation search area centered on the H.264 predicted motion vector. Analyzing these costs, we notice that for the same distance from the center $(0, 0)$, candidate blocks located near the diagonal ($\mathbf{x}_i \approx \mathbf{y}_i$) are more expensive than those near the axis ($\mathbf{x}_i \not\approx \mathbf{y}_i$). This helps to explain how the spiral search can weaken the rate-constrained filtering criterion. If the current best candidate is close to or on the diagonal, then the evaluation of candidate blocks closer to the axes will result in a positive value for ΔR_i, weakening the filtering criterion.

Applying the rule defined by equation (7) to the grid of Figure 1 can result in multiple search orderings. In this paper, we propose a search ordering that successively evaluates different quadrants around the center $(0, 0)$. The grids, b and c, in Figure 2 show 5×5 subsets of the 65×65 grids tested in the next section. We can note that in the proposed grid, Figure 2c, the axes are evaluated first, since their motion vectors require fewer bits. Next, the points closest to the center and the axes are evaluated (similar to an asymptote shape). We chose to alternate between quadrants, since this is also performed in the H.264 JM implementation of spiral search (Figure 2b). Alternating between quadrants can improve filtering, when a better candidate is found sooner, which will allow more SAD operations to be skipped.

Implementing a new search candidate ordering in an encoder like the H.264 joint model [12] is relatively straightforward and involves few implementation requirements, as a grid of values, analogous to the ordering lookup table in Algorithm 1, is used for spiral scan ordering. Our solution only requires changing the pointer to this grid. This small change allows for interesting results, as we will show in the next section.

22	22	20	20	18	12	18	20	20	22	22
22	22	20	20	18	12	18	20	20	22	22
20	20	18	18	16	10	16	18	18	20	20
20	20	18	18	16	10	16	18	18	20	20
18	18	16	16	14	8	14	16	16	18	18
12	12	10	10	8	2	8	10	10	12	12
18	18	16	16	14	8	14	16	16	18	18
20	20	18	18	16	10	16	18	18	20	20
20	20	18	18	16	10	16	18	18	20	20
22	22	20	20	18	12	18	20	20	22	22
22	22	20	20	18	12	18	20	20	22	22

Fig. 1: Grid of motion vector encoding bit lengths for the candidate blocks of a search area. The center of the search area $(0, 0)$ is the gray square.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

(a)

23	10	12	14	24
21	7	2	8	22
19	5	0	6	20
17	3	1	4	18
15	9	11	13	16

(b)

22	14	6	15	24
18	10	2	12	17
7	3	0	1	5
19	11	4	9	13
23	16	8	20	21

(c)

Fig. 2: Subsets of the raster search ordering (a), the H.264 JM implementation of spiral search ordering (b) and the proposed rate-constrained search ordering (c). The center of each search area $(0, 0)$ is the gray square. Values in these tables show the evaluation order of candidate blocks (from 0 to 24).

4. EXPERIMENTAL RESULTS AND DISCUSSION

To compare the proposed search ordering algorithm to the spiral search ordering algorithm, we implemented the proposed search ordering into the H.264/AVC JM 18.5 reference software [12]. We compared the number of SAD operations required to encode CIF (352×288) sequences using the

Table 1: SAD reduction using the proposed search ordering compared to the H.264 JM reference software's implementation of spiral search, as a function of block size and QP for several CIF video sequences.

		Foreman # of SAD operations for 300 frames			Football # of SAD operations for 260 frames			News # of SAD operations for 300 frames		
QP	Size	Spiral	Proposed	Red. %	Spiral	Proposed	Red. %	Spiral	Proposed	Red. %
28	4	416 262 070	388 993 410	**6.55%**	1 115 661 675	1 035 142 134	**7.22%**	134 537 882	128 099 468	**4.79%**
28	8	785 227 992	765 544 865	**2.51%**	1 955 919 279	1 882 526 019	**3.75%**	290 136 328	286 173 266	**1.37%**
28	16	409 325 310	401 608 855	**1.89%**	903 904 793	879 156 973	**2.74%**	309 741 039	308 103 709	**0.53%**
32	4	225 442 778	204 861 411	**9.13%**	698 105 494	638 767 242	**8.50%**	81 710 975	76 734 942	**6.09%**
32	8	648 570 481	627 984 818	**3.17%**	1 659 309 376	1 594 498 115	**3.91%**	255 001 256	249 897 228	**2.00%**
32	16	422 019 606	414 160 704	**1.86%**	922 208 528	898 298 829	**2.59%**	291 083 798	289 592 570	**0.51%**
36	4	107 804 660	95 285 467	**11.61%**	393 409 060	353 080 194	**10.25%**	44 836 321	41 544 099	**7.34%**
36	8	529 033 021	507 752 081	**4.02%**	1 185 610 980	1 133 690 522	**4.38%**	215 288 507	212 294 102	**1.39%**
36	16	426 912 515	419 050 593	**1.84%**	923 795 311	900 217 448	**2.55%**	270 475 527	269 005 875	**0.54%**
40	4	47 435 348	41 836 990	**11.80%**	183 815 418	161 532 698	**12.12%**	24 308 026	22 453 474	**7.63%**
40	8	405 457 244	383 738 455	**5.36%**	760 172 034	712 290 223	**6.30%**	166 808 837	163 627 932	**1.91%**
40	16	421 173 116	413 071 553	**1.92%**	876 436 298	856 138 643	**2.32%**	264 566 993	263 016 343	**0.59%**
		Average SAD reduction		**5.14%**	Average SAD reduction		**5.55%**	Average SAD reduction		**2.89%**

reference software's spiral search implementation against the proposed search ordering algorithm. To simplify results, we used the baseline profile with the following alterations: 5 reference frames, full pixel precision motion estimation and only 16×16, 8×8 and 4×4 block partitions. Similar results are expected with rectangular shaped blocks.

The number of SAD operations required for the "Foreman", "Football" and "News" sequences are listed in detail in Table 1. Table 2 lists the average reduction percentage of SAD operations for the "Foreman", "Flower", "Football", "Mobile", "News" and "Tempete" sequences. In this table, the column Δ Bits (kb/s) is the average bit rate difference, measured in kilobits per second, between the spiral search ordering encoding and the proposed search ordering encoding. The difference is very small, and is attributable to the search ordering algorithms finding different best candidates, but with the same cost values. This phenomenon has a low probability, but considering the number of candidate blocks evaluated, it does occur. This leads to an even smaller average difference in luma PSNR, listed in the Δ PSNR-Y column. For the Δ columns, a negative value indicates that the value, resulting from the encoding of the proposed search ordering, is smaller than that obtained by the spiral search encoding.

Table 2: Average results for spiral search ordering versus proposed search ordering, with the same experimental conditions as Table 1

Sequence	# Fr.	SAD Red.	Δ Bits (kb/s)	Δ PSNR-Y
Foreman	300	5.14%	-0.18	0.0000
Flower	250	1.61%	-0.21	-0.0017
Football	260	5.55%	0.09	-0.0025
Mobile	300	0.80%	-0.18	0.0008
News	300	2.89%	-0.04	0.0017
Tempete	260	1.14%	-0.11	0.0008
Average		2.86%	-0.10	-0.0001

From the results in Table 1, we can see that the proposed algorithm is more effective for smaller partition sizes. This is due to the higher ratio of bits required for the motion vector of the candidate block versus its SAD value. When this ratio increases, the weakening effect on the rate-constraint of the filtering criterion caused by the spiral search is more significant. A similar situation arises when the QP increases, which leads to an increase in the value of λ_{MOTION}, which is multiplied by ΔR_i, see equation (5).

Since most recent SEA algorithms use partitions to improve filtering efficiency, for example [5, 6, 9, 10], many 16×16 and 8×8 blocks will be evaluated using smaller partitions. When combined with the proposed method, these algorithms will lead to an overall increase in the reduction of SAD operations.

Table 1 and 2 show that the proposed search ordering is, on average, more efficient with sequences that contain important and unpredictable movement ("Foreman", "Football"), as compared to more predictable sequences. An increase in motion vector size leads to an increase in the ratio between the number of bits required to encode motion vectors and the SAD values. More nonzero motion vectors will cause an increase in the probability of weakening the filtering criterion (choosing a candidate on or near the diagonal).

5. CONCLUSION

A new class of candidate block search ordering algorithms, named rate-constrained search ordering algorithms, has been proposed to eliminate the weakening of the filtering criterion by candidate block search orderings that do not take into consideration the impact of the rate constraint. For the H.264/AVC JM reference software, changing the candidate block ordering requires few implementation considerations, and can reduce the number of SAD operations required for motion estimation with negligible impact on bit rate and visual quality. Further work is required, but similarities between motion estimation in H.264 and HEVC indicate that this approach could be adapted to HEVC.

6. REFERENCES

[1] 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*, vol. 13, no. 7, pp. 560–576, 2003.

[2] A.M. Tourapis, O.C. Au, and M.L. Liou, "Predictive motion vector field adaptive search technique (pmvfast): enhancing block-based motion estimation," *Proc. SPIE Visual Communications and Image Processing*, vol. 4310, pp. 883–892, 2001.

[3] A.M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," *Proc. SPIE Visual Communications and Image Processing*, vol. 4671, pp. 1069–1079, 2002.

[4] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Transactions on Image Processing*, vol. 4, no. 1, pp. 105–7, Jan. 1995.

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

[6] C. Zhu, W.-S. Qi, and W. Ser, "Predictive fine granularity successive elimination for fast optimal block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 213–221, Feb. 2005.

[7] M.Z. Coban and R.M. Mersereau, "A fast exhaustive search algorithm for rate-constrained motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 5, pp. 769–773, May 1998.

[8] K.P. Lim, G.J. Sullivan, and T. Wiegand, "Text description of joint model reference encoding methods and decoding concealment methods," *JVT-R095, JVT of ISO/IEC MPEG and ITU-T*, Jan. 2006.

[9] M. Yang, H. Cui, and K. Tang, "Efficient tree structured motion estimation using successive elimination," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 151, no. 5, pp. 369–377, Oct 2004.

[10] T. Toivonen and J. Heikkila, "Fast full search block motion estimation for H.264/AVC with multilevel successive elimination algorithm," in *ICIP '04. International Conference on Image Processing*, Oct. 2004, vol. 3, pp. 1485–1488.

[11] J.N. Kim and T.S. Choi, "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1040–1048, Oct. 2000.

[12] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, "H.264/AVC JM Reference Software," `http://iphome.hhi.de/suehring/tml/`, May 2013, version 18.5.