Cost-Based Search Ordering for Rate-Constrained Motion Estimation Applied to HEVC

Luc Trudeau, *Student Member, IEEE*, Stéphane Coulombe, *Senior Member, IEEE*, and Christian Desrosiers, *Member, IEEE*

***Abstract*—In the context of motion estimation for video coding, combining a successive elimination algorithm (SEA) with a motion estimation algorithm reduces the number of computed cost functions without impact on rate or distortion. The SEAs use the sum of absolute differences to eliminate motion vector candidates in the search area that cannot improve the current minimum. The novelty in this paper is that instead of relying on a static geometric pattern (i.e., like a spiral), we proposed a dynamic algorithm that creates a cost-based search orderings. A dynamic cost-based search ordering not only improves elimination but also allows for early termination which removes, on average, 61% of the block-matching loop iterations performed by the rate-constrained successive elimination algorithm (RCSEA). Our experiments show that the proposed solution is 5 times faster than the high efficiency video coding (HEVC) HM encoder software in full search mode with a 0.02% impact on BD-Rate. This is twice the speed of the HEVC HM software encoder using only the RCSEA.**

***Index Terms*—SEA, RCSEA, BMA, high efficiency video coding (HEVC), H.265, video compression.**

NOMENCLATURE

ADS	Absolute difference of sums.
ILMVP	Integer level motion vector predictor.
MVP	Motion vector predictor.
RCADS	Rate-constrained absolute difference of sums.
RCSAD	Rate-constrained sum of absolute differences.
RCSEA	Rate-constrained successive elimination algorithm.
SAD	Sum of absolute differences.
SEA	Successive elimination algorithm.

I. INTRODUCTION

MODERN video encoders consider more block sizes, more anchor frames, and use bigger search areas for motion estimation (ME) than their former counterparts [1], [2]. The magnitude of the solution space of ME makes suboptimal search algorithms, like those presented in [3]–[7], commonplace in encoders compliant with feature-rich video compression standards like H.264/MPEG-4 advanced video coding (AVC) [8] and H.265/high efficiency video coding (HEVC) [9]. As shown in [10], the integer-level ME time increased 12-fold from the H.264 reference encoder to the HEVC reference encoder.

However, when suboptimal search algorithms encounter unpredictable movement, they will often yield a higher residual energy which requires more bits to encode; even more so when the quality is high. This bit rate increase is about 1 to 2%, but it can reach up to 11% for certain sequences [11]. To avoid this bit rate increase, some suboptimal algorithms, like the TZ-Search [12], will fall back to a semi-exhaustive search when movement is unpredictable.

In the context of semi-exhaustive fallback strategies, the SEA [13] and its derivatives: RCSEA [14], multi-level successive elimination algorithm (MSEA) [15] and fine granularity successive elimination (FGSE) [16] become very appealing, as they find the same results as an exhaustive search, but only evaluate the cost function of a fraction of motion vector (MV) candidates. Implementations of the RCSEA have been presented for H.264/MPEG-4 AVC in [17] and [18]. To our knowledge, except for our previous work [19], [20], no work has been published on RCSEA for H.265/HEVC. Althought the work of Seidel *et al.* [21] also targets HEVC, it targets sub pel motion estimation whereas this work targets full pel motion estimation. As such, both approach can be combined but do not compete with each other.

In this paper, we improve and adapt the RCSEA for the H.265/HEVC standard. The proposed solution provides identical compression efficiency and visual quality as full search and yet is approximately 5 times faster than the HEVC HM encoder software in full search mode. It is also twice as fast as the HEVC HM encoder using RCSEA. A sub-optimal version of our solution can replace the semi-exhaustive fallback of the TZ-Search resulting in an average speed up of 1.34 times and a 0.12% increase in BD-Rate [22].

After a description of the fundamentals of RCSEA in Section II, the structure and the contributions of this work are as follows:

- Section III: we expose and analyze two problems related to the implementation of the SEA in modern standards like H.264/AVC and H.265/HEVC: asymmetric distributions of MV costs and off-centered search areas. We found that pre-computed static search orderings cannot guarantee to resolve these issues.
- Section IV: we show that the proposed method can be applied to some classes of suboptimal ME algorithms. For instance, it provides a significant speedup to TZ-Search when we replace its the semi-exhaustive fallback by our novel suboptimal cost-based refinement.
- Section IV-C: we introduce an early termination criterion for the RCSEA which speeds up the ME process.

In Section VI, we present our experimental results for the RCSEA and the proposed algorithm in the context of the H.265/HEVC standard. The main contribution here lies in reducing the size of the solution space of ME while preserving the optimal candidate. This can serve for both full search and semi-exhaustive fallback strategies.

II. RATE-CONSTRAINED SUCCESSIVE ELIMINATION ALGORITHM

In this section, we first present a high-level overview of block-matching, and then explain the details of RCSEA in the following sub-sections.

Fig. 1 illustrates the concepts related to the ME algorithm using a block-matching algorithm (BMA). The current frame to encode is

Manuscript received June 24, 2017; revised January 31, 2018; accepted May 31, 2018. This work was supported in part by Vantrix Corporation and in part by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program under Grant NSERC-CRD 428942-11. *(Corresponding author: Luc Trudeau.)*

The authors are with the Department of Software and IT Engineering, École de technologie supérieure, Montreal, QC H3C 1K3, Canada (e-mail: luc@trud.ca; stephane.coulombe@etsmtl.ca; christian.desrosiers@etsmtl.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2018.2847444

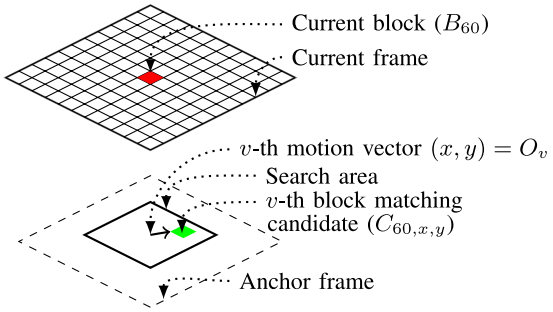


Fig. 1. Block-matching consists in evaluating, in a given order O, candidate blocks ($C_{i,x,y}$) of a search area against the current block B_i using an error metric.

segmented into non-overlapping blocks. Let B_i denote the i-th block in the frame, accessed by raster scan ordering. B_i is made up of $M \times N$ pixels, and these pixels are accessed as $B_i(m, n)$. Let C_i denote the search area corresponding to B_i. Candidate blocks inside this search area have the same size as the current block. The pixels of the candidate block at position (x, y) are accessed as $C_{i,x,y}(m, n)$.

Block-matching consists in finding the best match for B_i by evaluating an error metric on candidate blocks ($C_{i,x,y}$) in the search area of the reconstructed anchor frame. In this paper, we use the sum of the absolute differences (SAD) of block pixels as our error metric

$$\mathrm{SAD}(i, x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |B_i(m, n) - C_{i,x,y}(m, n)|. \tag{1}$$

This error metric is not the most precise, but it is widely used by encoders because of its simplicity. This simplicity is crucial, considering the number of times it is computed.

A. Rate-Constrained Block-Matching

To better reflects the quality settings of the encoder, modern BMAs use a cost function that combines the error metric and a metric representing number of bits required to encode the motion vector of the specified candidate [23]. The rate-constrained sum of the absolute difference (RCSAD) is defined as follows:

$$\mathrm{RCSAD}(i, x, y) = \mathrm{SAD}(i, x, y) + \lambda R(x, y), \tag{2}$$

where λ is a weighting coefficient, also known as the Lagrange multiplier. In this work, the value of λ is computed using the recommended function for HEVC specified in [12]. The function $R(x, y)$ returns a metric representing the number of bits required to encode the motion vector (x, y).

The R function, used in Eq. (2), is defined as follows:

$$R(x, y) = G\big(4\big(x - x_p\big)\big) + G\big(4\big(y - y_p\big)\big), \tag{3}$$

where $G(v)$ returns the length of the signed exponential Golomb code required to encode v, and $(x_p, y_p) \in \frac{1}{4}\mathbb{Z}^2$ is the motion vector predictor (MVP). Let $\frac{1}{4}\mathbb{Z}$ be the group of all integers and quarter-integers $(0, \pm\frac{1}{4}, \pm\frac{1}{2}, \pm\frac{3}{4}, \pm 1, \ldots)$ [24, p. 390]. The differential between the MV and the MVP is multiplied by four to obtain integer values conforming to the H.264 and the HEVC standards, which encode MVs using quarter pixel precision specified with fixed point notation.

B. Signed Exponential Golomb Codes

Signed exponential Golomb codes are used in the H.264 standard to encode the differences between MVs and the MVP. As such, all the contributions in the paper also apply to H.264. However, this work is only focused towards HEVC.

Although signed exponential Golomb codes are not used by the HEVC standard to encode MV information, they are the recommended metric used to quickly measure MV costs in rate-constrained block-matching algorithm (RCBMA) for HEVC [12]. It follows that the contributions in the paper apply to HEVC encoders that follow this official recommendation.

As described in [25], exponential Golomb codes are composed of a prefix part, a marker bit and an info part. The total length of an exponential Golomb code for an integer value v is measured with the following function:

$$G(v) = 2 \times I(v) + 1. \tag{4}$$

The $I()$ function returns the length of the info part which is multiplied by 2, because the prefix is the same length as the info. The added one represents the marker bit. The $I()$ function is defined as follows:

$$I(v) = \lfloor \log_2(2|v| + 1) \rfloor, \tag{5}$$

where $|v|$ is multiplied by 2, because the code words used in the info part alternate between negative and positive values. One is added to represent that the value 0 is assigned to the code word '0'. The $\lfloor\ \rfloor$ operator represents the floor function.

C. Successive Elimination Algorithm

The contribution of the RCSEA to the BMA are twofold. First, by using an error approximation metric, the absolute difference of sums (ADS), it avoids evaluating the error metric on candidates that cannot produce better results than the current best [13], [14]. Second, contrary to the SAD, the components of the ADS can be stored and reused, thus considerably reducing the amount of operations required compute the ADS.

There exists a triangle inequality relationship between the SAD and the ADS, such that:

$$\begin{aligned}\mathrm{ADS}(i, x, y) &= \left| \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} B_i(m, n) - \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C_{i,x,y}(m, n) \right| \\ &\leqslant \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |B_i(m, n) - C_{i,x,y}(m, n)| = \mathrm{SAD}(i, x, y).\end{aligned} \tag{6}$$

Concretely, the ADS is a lower bound of the SAD. This inequality can be used to perform transitive elimination. For a given candidate at position (x, y), if the ADS value of that candidate is superior to the SAD of the current best candidate (at position $(\hat{x}, \hat{y})$), then it is impossible for the SAD value of that candidate to be smaller than that of the current best candidate. Given the SAD value of one candidate in the search area, all candidates with an ADS value greater than the given SAD can safely be eliminated without altering the optimal candidate chosen by the BMA.

Coban and Mersereau [14] proposed the RCSEA which enables the weighted rate to contribute to the transitive elimination:

$$\mathrm{ADS}(i, x, y) + \lambda R(x, y) \leqslant \mathrm{RCSAD}(i, x, y). \tag{7}$$

Coban and Mersereau [14] noted that when λ increases, the number of candidates eliminated by the transitive elimination also increases. This is caused by the increase in the significance of the weighted rate over the SAD. As described in the next section, this can improve or impair transitive elimination.

Over the course of an entire frame, block-matching requires to repetitively load and store the pixel values of B_i or $C_{i,x,y}$ in the central processing unit (CPU). However, the SAD operations themselves are

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

(a) Spiral

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) H.264 JM

Fig. 2. Subsets of the spiral search ordering (a) and the H.264 JM implementation of spiral search ordering (b). The gray square is the position of the MVP. Values in these grids show the evaluation order of candidate blocks (from 0 to 24).

not redundant because the combinations of B_i and $C_{i,x,y}$ differ, and the summation used in the Eq. (1) is not distributive over the absolute value. To better exploit this repetitiveness, the summation can be distributed between B_i and $C_{i,x,y}$. This results in the ADS, which can be viewed as the projection of the blocks pixel values onto a single dimensional space [26].

Contrary to the SAD, the ADS does not impose repetitive operations, as the values of $\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} B_i(n,m)$ and $\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} C_{i,x,y}(n,m)$ can be pre-computed and reused. Although the ADS cannot replace the SAD, it can be used to perform transitive elimination. Performance-wise, the ADS requires: one subtraction and one absolute operation (ignoring the precomputation step described below). On the other hand, each SAD operation requires $3 \times M \times N - 1$ operations ($M \times N$ subtractions, $M \times N$ absolute values, $M \times N - 1$ additions).

Li and Salari [13] present a fast summation technique to precompute $\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} B_i(n,m)$ and $\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} C_{i,x,y}(n,m)$. As demonstrated by the authors, the overhead required by this technique is negligible when compared to the computational savings.

III. Search Ordering Analysis

In this section, we describe our increasing rate rule and its two main challenges related to HEVC and H.264: asymmetric distributions of MV costs and off-centered search areas.

A. The Increasing Rate Rule

Search ordering candidates can be separated into two categories: those where the rate-constrained absolute difference of sums (RCADS) is smaller than the minimum (optimum) RCSAD, and all others. The former will always require SAD computations, while the latter, for their part, might not require any; it all depends on the search ordering used by the BMA. The problem is that BMA is performed iteratively over the candidates, and the minimum RCSAD is not known in advance, but rather, is discovered through the process.

The search ordering is therefore crucial to the efficiency of the transitive elimination performed by the RCSEA. Considering bad candidates up front will cause a high value for the current minimum used by RCSEA. This high minimum will reduce the efficiency of the transitive elimination and lead to more SAD computations than if a good candidate had been considered first. The ideal situation is when the candidate with the minimum RCSAD is considered first. Although it may not eliminate the need to evaluate the SAD for all the other candidates, it will lead to the transitive elimination of the highest amount of candidates, and thus to the lowest amount of SAD evaluations.

Almost all modern RCSEA implementations use a spiral scan (Fig. 2a), or a derivative of the spiral scan ordering, like the one found in the H.264 JM [27] (Fig. 2b).

These orderings traverse candidates starting at the MVP and move outwards. The underlying assumption is that the likelihood of finding the smallest candidate decreases as the magnitude of the motion vector increases [26]. Nevertheless, motion vector magnitude is not equivalent to motion vector cost. As such, when transitive elimination is used, search orderings should be based on motion vector cost (Eq. (3)), not motion vector magnitude.

The impact on the transitive elimination can be made more explicit by reordering Eq. (7) (with the rate extension of Eq. (2)), as follows:

$$\mathrm{SAD}\big(i,\hat{x},\hat{y}\big) + \lambda\big(R\big(\hat{x},\hat{y}\big) - R(x,y)\big) \leqslant \mathrm{ADS}(i,x,y). \tag{8}$$

When the rate of a candidate is lower than the rate of the current minimum $R(\hat{x},\hat{y}) - R(x,y) > 0$, the rate differential will be multiplied by λ and added to the current minimum SAD. This additional quantity makes it harder to meet the inequality permitting to eliminate the candidate. Thus, this interferes with the inequality and can cause useless SAD computations. The only way to avoid this is to use a scan ordering arranged by increasing rates. That way, $R(\hat{x},\hat{y}) - R(x,y) \leqslant 0$, and now, the rate differential can even improve transitive elimination. This is the reasoning behind our *increasing rate rule*, as presented in [28].

The increasing rate rule also allows early termination, a new speedup for optimal ME algorithms. When the weighted cost of the candidate motion vector is greater than the current minimum cost function, the current minimum is the global minimum, and the BMA can stop as no other candidate can produce a smaller value, as stated in the following inequality:

$$\begin{aligned}&\mathrm{SAD}\big(i,\hat{x},\hat{y}\big) + \lambda R\big(\hat{x},\hat{y}\big) \leqslant \mathrm{SAD}(i,x,y) + \lambda R(x,y)\\ &\forall\,(x,y) \text{ where } \lambda R(x,y) \geqslant \mathrm{SAD}(i,\hat{x},\hat{y}) + \lambda R(\hat{x},\hat{y}).\end{aligned} \tag{9}$$

In our previous proposition [28], we built a geometric pattern, similar to a spiral scan, intended to satisfy the increasing rate rule. We showed that this pattern eliminates useless SAD operations performed when using a spiral scan ordering. However, further work has shown that implementation considerations found in HEVC and H.264 introduce certain conditions where a static geometric pattern will not always fully satisfy the increasing rate rule. In order to satisfy the increasing rate rule in all conditions, an impractical number of geometric patterns is required.

B. Asymmetric Distribution of Motion Vector Costs

Modern encoders perform ME in two steps. First, the BMA is performed at the integer pixel level. This allows the encoder to find the best integer match. Next, the best integer match is used as the center for a small search area, called refinement, where the BMA is performed at the fractional pixel level. The high computational cost required to interpolate the values of fractional pixels motivates the use of this multi-step approach.

In this work, we only focus on the integer level, for two reasons. First, fractional level block-matching operations account for only a small percentage of block-matching operations. This is mainly due to the high computational cost required to interpolate the sub pixel values. Second, modern encoders rely on the sum of the absolute transformed differences (SATD) to evaluate candidates at the fractional level, whereas our work is based on the SAD.

Even at the integer level, the methods used to compute the MVP commonly yield fractional MVPs. Fractional MVPs are problematic because they point to positions that do not exist at the integer level.

We define the integer level motion vector predictor (ILMVP), $(\widetilde{x}_p, \widetilde{y}_p) \in \mathbb{Z}^2$, as the closest integer level position from the fractional level MVP $(x_p, y_p) \in \frac{1}{4}\mathbb{Z}^2$.

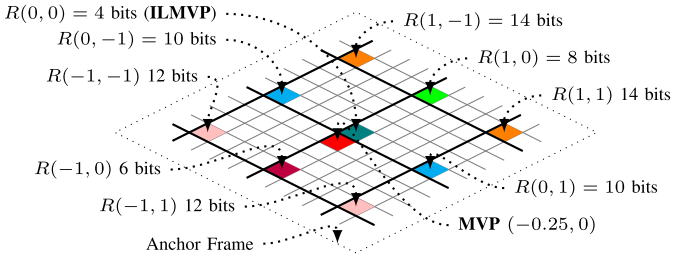


Fig. 3. The MVP is shown with a red block at $(-0.25, 0)$. The candidates are symmetrically located around the ILMVP (teal block); however, horizontal integer level candidates on the left require fewer bits than the candidates on the right.

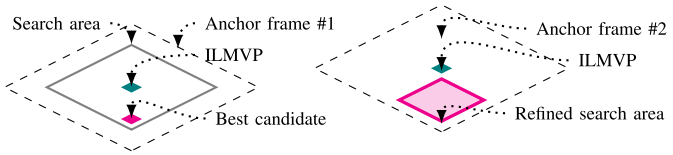


Fig. 4. The ILMVP (teal block) is not in the refined search area (magenta surface).

This position is obtained by a round half up operation (round( )) on the fractional values of the MVP:

$$\widetilde{x}_p = \text{round}(x_p),\ \widetilde{y}_p = \text{round}(y_p). \tag{10}$$

Similarly, we denote the integer level version of (x, y) as $(\widetilde{x}, \widetilde{y})$.

Fig. 3 shows an example of asymmetric distribution of MV costs; where horizontal integer level candidates to the left of the ILMVP are less expensive than candidates to the right.

This phenomenon is caused by the fact that the search ordering is based on integer level MVs while quarter pixel level MV differentials are encoded. To cope with this situation, we must track the offset $(x_{\text{off}}, y_{\text{off}}) \in \{(x, y) | x, y \in \{0, \pm\frac{1}{4}, \pm\frac{1}{2}\}\}$, defined as the difference between the ILMVP $(\widetilde{x}_p, \widetilde{y}_p)$ and the MVP (x_p, y_p). The offset is computed as follows:

$$x_{\text{off}} = x_p - \widetilde{x}_p,\ \ y_{\text{off}} = y_p - \widetilde{y}_p. \tag{11}$$

The reason why asymmetric distributions of MV costs are problematic is that the offset between the ILMVP and the MVP must be known in order to satisfy the increasing rate rule. Considerable complexity must be added to an approach that relies on a search ordering based on a geometric pattern in order to handle all possible asymmetric configurations of MV costs.

C. Off-Centered Search Areas

We define an off-centered search area as a search area where the center does not correspond to the ILMVP. Off-centered search areas can be caused by clipping or by the use of refinement zones inside the search area. Clipping occurs when the current block is near the edge of the frame and the search area is limited by the size of the computer memory allocated to store the image. In order to save computations, modern encoders, like the HEVC reference encoder, will sometimes use refinement zones, based on previous block-matching information. A refinement zone is a subset of the search area centered on the position of a best candidate found in another reference frame. Some of these refinement zones do not even include the MVP, as illustrated in Fig. 4.

In this situation, additional computations are required by the BMA to adapt the geometric pattern to the context. It follows that, for off-centered search areas, satisfying the increasing rate rule with a geometric pattern requires supplemental computations. As such, off-centered search areas impose context-specific requirements on the search ordering.

As explained, because of the asymmetric distribution of MV costs and off-centered motion estimation the increasing rate rule cannot be satisfied by a static geometric pattern like a spiral. The combinatorial requirements also prohibits hard coding multiple static search orderings into the encoder. A dynamic solution is required as a specific ordering must be built for every context. Concretely, geometric patterns are obsolete; the next candidate must be determined by its properties. In this work, the next candidate is determined by its cost, but it can be determined by other properties, such as its ADS [19].

D. Suboptimal Search Orderings

Dynamic search ordering and early termination is nothing new for suboptimal ME algorithms like [5], [6], [11]. However, these orderings are not compatible with optimality preserving approaches. An RCSEA using a dynamic search ordering from a suboptimal algorithm would have to either return a suboptimal value or avoid early termination.

Dynamic search ordering and early termination for optimality preserving algorithm is new and noteworthy. By dynamically satisfying the increasing rate rule, the proposed algorithm not only allows for early termination, but it also guarantees that the optimal candidate is returned.

IV. Proposed Cost-Based Search Ordering

To resolve the previously identified search ordering problems, we propose a new cost-based search ordering method. This method, based on a cost-based search ordering model, dynamically produces a search ordering that follows the increasing rate rule. This new ordering not only improves the effectiveness of the SEA, but, more importantly, it does not require the computation of the motion vector cost (Eq. (3)), it reduces the overhead related to off-centered search areas and it allows for a new optimization: an early termination criterion for the BMA. The combination of these factors, as shown in Section VI, lead to speed ups of 1.8x compared to RCSEA and 5x compared to HM using full-search motion estimation.

An overview of the proposed solution is given in Fig. 5. First, the cost-based search ordering algorithm produces a context-based search ordering (O), where the context is composed of the MVP and the search area. Next, given the search ordering and the precomputed block sums ($sumB$, $sumC$), the RCSEA eliminates candidates that cannot produce a cost lower than the current best cost. Afterwards, the BMA evaluates candidates from the filtered search ordering by computing the error metric (RCSAD). Finally, the RCSEA uses the current best cost found by the BMA to improve transitive elimination. The early termination criterion enables the BMA to evaluate only a subset of the candidates in the filtered search ordering.

A. Cost-Based Search Ordering Model

In the conventional approach, the MVs are ordered according to a geometric pattern which requires computing the rate (Eq. (3)) for every MV. A cost-based model is fundamentally different, the encoding cost c of interest is known, but the associated MVs are unknown. Given an unlimited search area, each quadrant will contain as many surfaces of cost c as there are combinations of bit lengths that sum

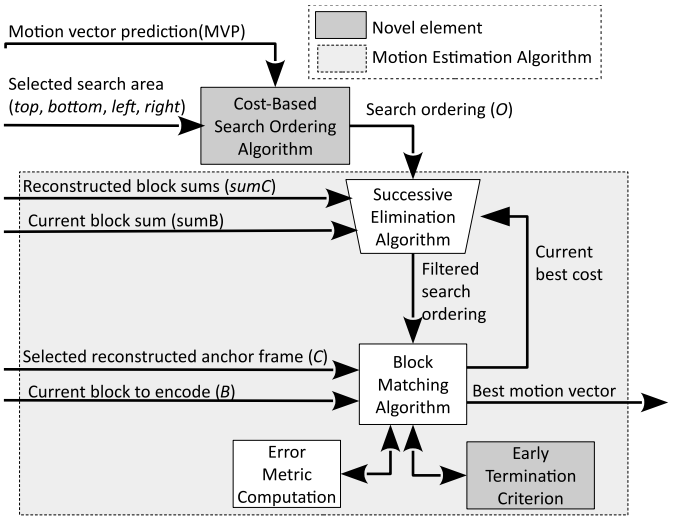


Fig. 5. Interactions between the ME algorithm, the RCSEA and the proposed cost-based search ordering algorithm.

to c. This relates to the complementary relationship between x and y, which we derive from Eq. (5), Eq. (4) and Eq. (3):

$$\begin{aligned} c &= 2 \times I\big(4\big(y - y_p\big)\big) + 2 \times I\big(4\big(x - x_p\big)\big) + 2 \\ &\Rightarrow I\big(4\big(y - y_p\big)\big) = \frac{c}{2} - I\big(4\big(x - x_p\big)\big) - 1, \\ &\quad \forall (x, y) \text{ such that } R(x, y) = c. \end{aligned} \tag{12}$$

The encoding cost of an MV differential is broken down into different combinations of bit lengths for the x and y differentials. Let $\Delta x = x - x_p$ be the x differential and $\Delta y = y - y_p$ be the y differential, where $(\Delta x, \Delta y) \in \frac{1}{4}\mathbb{Z}^2$. Given a certain bit length, we extract Δx from Eq. (5), in order to find the corresponding range of values of Δx, as in:

$$\begin{aligned} I(4\Delta x) &= \lfloor \log_2(2|4\Delta x| + 1) \rfloor \\ &\Rightarrow \frac{2^{I(4\Delta x)-1} - \frac{1}{2}}{4} \leqslant |\Delta x| < \frac{2^{I(4\Delta x)} - \frac{1}{2}}{4} \end{aligned} \tag{13}$$

To obtain Eq. (13), we first reformulate Eq. (5) without the binary logarithm. This is done by combining the binary logarithm definition $i = \log_2(k) \iff 2^i = k$ and the floor function equivalence $\lfloor i \rfloor = k \iff k \leqslant i < k + 1$ as follows:

$$I(v) = \lfloor \log_2(2|v| + 1) \rfloor \iff 2^{I(v)} \leqslant 2|v| + 1 < 2^{I(v)+1}. \tag{14}$$

Then, isolating v on the right side of Eq. (14) and replacing v by $4\Delta x$ leads to Eq. (13).

As explained in Section III-B, this model only applies to integer level ME, and so $(x, y) = (\widetilde{x}, \widetilde{y}) \in \mathbb{Z}^2$. This implies, using Eq. (11), that

$$\Delta x = \widetilde{x} - x_p = \underbrace{\widetilde{x} - \widetilde{x}_p}_{\delta_x \in \mathbb{Z}} - x_{\text{off}}. \tag{15}$$

Since the approach is quadrant-based (more on this in the next section), we only consider $\Delta x \geqslant 0$. Therefore, for that quadrant, we have:

$$\begin{aligned} &\frac{2^{I(4\Delta x)-1} - \frac{1}{2}}{4} \leqslant \Delta x < \frac{2^{I(4\Delta x)} - \frac{1}{2}}{4} \\ \Rightarrow &\frac{2^{I(4\Delta x)-1} - \frac{1}{2}}{4} \leqslant \delta_x - x_{\text{off}} < \frac{2^{I(4\Delta x)} - \frac{1}{2}}{4} \\ \Rightarrow &\frac{2^{I(4\Delta x)-1} - \frac{1}{2}}{4} + x_{\text{off}} \leqslant \delta_x < \frac{2^{I(4\Delta x)} - \frac{1}{2}}{4} + x_{\text{off}}. \end{aligned} \tag{16}$$

Next, we multiply the numerator and the denominator by two to remove the $-\frac{1}{2}$ for both fractions

$$\frac{2^{I(4\Delta x)} - 1}{8} + x_{\text{off}} \leqslant \delta_x < \frac{2^{I(4\Delta x)+1} - 1}{8} + x_{\text{off}}. \tag{17}$$

Note that the denominator is the multiplication of four, for quarter pixel values, and two, because of adjacent negative and positive codes in the exponential Golomb coding (see Section II-B).

Now, since δ_x is an integer, we need integer values for range boundaries. Therefore, to satisfy Eq. (16)

$$\left\lceil \frac{2^{I(4\Delta x)} - 1}{8} + x_{\text{off}} \right\rceil \leqslant \delta_x < \left\lceil \frac{2^{I(4\Delta x)+1} - 1}{8} + x_{\text{off}} \right\rceil. \tag{18}$$

To accelerate the implementation, integer divisions are used. This is not problematic, since $(x, y) = (\widetilde{x}, \widetilde{y})$. However, integer divisions are equivalent to floored divisions. To convert from ceiling to floor, 7 is added to the numerator of the integer divisions. It can be shown that we obtain the equivalent equation:

$$\begin{aligned} &\left\lfloor \frac{2^{I(4\Delta x)} + 6}{8} + x_{\text{off}} \right\rfloor \leqslant \delta_x < \left\lfloor \frac{2^{I(4\Delta x)+1} + 6}{8} + x_{\text{off}} \right\rfloor \\ \Rightarrow &\frac{2^{I(4\Delta x)} + 6 + 8x_{\text{off}}}{8} \leqslant \delta_x < \frac{2^{I(4\Delta x)+1} + 6 + 8x_{\text{off}}}{8}. \end{aligned} \tag{19}$$

Note that in Eq. (18), the x_{off} was inside the ceiling function. In Eq. (19), it must be moved into the integer division, yielding $8x_{\text{off}}$. Although the same notation is used, the divisions in the second line of Eq. (19) are integer divisions. As described in the next section, the integer divisions are implemented in the CLIPRANGE function (Fig. 8) using a right bit shift operator $>>$.

Eq. (19) also applies to the y differential. By combining the Δx range with the complementary Δy range, we obtain a surface, where all candidates have the same cost (see Fig. 10).

B. Fast Cost-Based Search Ordering Implementation

A cost based approach offers the novel possibility of working with same-cost surfaces. Same-cost surfaces are formed inside the search area, when multiple (x, y) pairs yield the same value for $R(x, y)$. Because of the log operator in Eq. (5), these same cost surfaces grow exponentially as the value $R(x, y)$ increases. These surfaces have interesting properties: the rate-constraint is constant, they are rectangular, disjoint, symmetric in other quadrants, and they appear only once per row inside each quadrant.

The SEARCHORDERING function (Fig. 6) uses these properties to "jump" into every same-cost surfaces of cost c inside the search area. Moreover, because of the underlying binary structure of the exponential Golomb codes and Eq. (19), the CLIPRANGE function can quickly identify same-cost surfaces boundaries only using simple operations like sums and shifts.

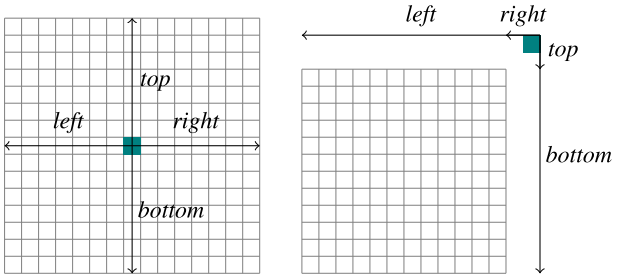
The SEARCHORDERING function is summarized as follows:

- Lines 1 to 7 initialize key variables of the algorithm.
- Line 8 iterates over all valid costs inside the search area.
- Line 9 assigns the smallest valid bit length to the Δy component (*yLen*).
- Line 10 implements Eq. (12) to establish the complementary bit length for the Δx component (*xLen*).
- Line 11 iterates over all combinations of *xLen* and *yLen* that lead to the same cost, c.
- Lines 12 to 15 invoke the CLIPRANGE function to identify valid ranges for the Δx and Δy components (implementing Eq. (19)).
- Line 16 invokes the ADDCANDSFROMSURFACE function to add all valid candidates in the same-cost surfaces to the search ordering.

SEARCHORDERING($top, bottom, left, right, x_{\text{off}}, y_{\text{off}}$)

1 $qTop = 4 \times (top + y_{\text{off}})$
2 $qBottom = 4 \times (bottom - y_{\text{off}})$
3 $height = \text{I}(\text{MAX}(qTop, qBottom))$
4 $yStart = \text{I}(\text{MIN}(0, qTop, qBottom))$
5 $c = 2 \times (yStart + 1)$
6 $oT = \text{MAX}(-bottom, y_{\text{off}} == 0)$
7 $oL = \text{MAX}(-right, x_{\text{off}} == 0)$
8 **while** $yStart \leqslant height$
9 $\quad yLen = yStart$
10 $\quad xLen = \frac{c}{2} - yLen - 1$
11 $\quad$ **while** $xLen >= 0$
12 $\quad\quad T = \text{CLIPRANGE}(2^{yLen}, -y_{\text{off}}, oT, top)$
13 $\quad\quad B = \text{CLIPRANGE}(2^{yLen}, y_{\text{off}}, -top, bottom)$
14 $\quad\quad L = \text{CLIPRANGE}(2^{xLen}, -x_{\text{off}}, oL, left)$
15 $\quad\quad R = \text{CLIPRANGE}(2^{xLen}, x_{\text{off}}, -left, right)$
16 $\quad\quad \text{ADDCANDSFROMSURFACE}(O, T, B, L, R, c)$
17 $\quad\quad yLen = yLen + 1$
18 $\quad\quad xLen = xLen - 1$
19 $\quad\quad$ **if** $R.up$ == $right$ and $L.up$ == $left$
20 $\quad\quad\quad yStart = yLen$
21 $\quad c = c + 2$
22 **return** O

Fig. 6. The SEARCHORDERING function creates a cost-based search ordering. For each cost, this function combines the bit length of the x and y components to determine the surface of all candidates of the same cost in every quadrant.



(a) ILMVP centered inside the search area.

(b) ILMVP outside the search area.

Fig. 7. Search area defined by: *top*, *bottom*, *left*, *right*. The colored candidate is the ILMVP. In Fig. 7b, the ILMVP is outside the search area and the values of *top* and *right* are negative.

To deal with asymmetric distributions of MV costs and off-centered search areas, the search area is divided into quadrants, where the origin is the ILMVP, as shown in Fig. 7a. The offset between the MVP and the ILMVP (x_{off}, y_{off}) from Eq. (11) is passed as a parameter to the SEARCHORDERING function. The search area is represented using four variables: *top*, *bottom*, *left*, *right*. These variables specify the integer level limits of each quadrant relative to the ILMVP; this can be seen in Fig. 7.

The algorithm starts at the ILMVP and iterates upwards and downwards (see Fig. 10). Contrary to conventional approaches that work with row and column indexes, the SEARCHORDERING function works directly with row and column bit lengths. This allows the SEARCHORDERING function to work on an exponential increment of rows and columns in each loop iteration, thus saving considerable looping operations.

The *qTop* and *qBottom* variables store the *top* and *bottom* quarter pixel level values, adjusted with the offset. With these variables and Eq. (5), we can compute their corresponding bit lengths. The biggest bit length is stored in the *height* variable. Since the algorithm starts at the ILMVP and iterates outwards in a vertical fashion, the *height* variable represents the vertical limit from the ILMVP.

CLIPRANGE(v, off, min, max)

1 $num = v + 8off + 6$
2 $R.low = \text{CLIP}(num >> 3, min, max)$
3 $R.up = \text{CLIP}((num + v) >> 3, min, max)$
4 **return** R

Fig. 8. The CLIPRANGE function returns the range of the same-cost surface using Eq. (19), where v is 2 to the power of the component bit length, and *off* the offset.

The starting bit length of the Δy component *yStart* is greater than 0, when either the *top* variable or *bottom* variable is negative. As shown in Fig. 7b, in such a case, the ILMVP is not in the search area. Since the code words used alternate values between negative and positive values, negative and positive values have the same bit length (see Eq. (5)); hence, the bit length of a negative value will dictate the starting bit length of the Δy component. On line 5, the cost is measured as $2 \times (yStart + 1)$, the '1' counts for the marker bit of the Δy component and the smallest code for the Δx component: no prefix, no info, just the marker bit.

An ambiguous issue with a quadrant-based approach is assigning a quadrant to the column zero and the row zero. This is handled by oL and oT. These variables will skip over column zero or row zero when the $y_{\text{off}} == 0$ or $x_{\text{off}} == 0$ condition is met, respectively (($y_{\text{off}} == 0$) returns 1 when the condition is met and returns 0 otherwise; similarly for ($x_{\text{off}} == 0$)). When the offset is non-zero, there is no ambiguity since only one quadrant will have a column zero or a row zero. Another case that is not ambiguous is when the variable of the opposite direction is negative.

The c variable stores the current cost, the sum of the exponential Golomb code lengths (Eq. (3)). Cost increments are of two, because the next exponential Golomb code requires one bit for the prefix and one bit for the info.

An important consideration for this fast implementation is that the $I()$ function is not used to find the bit lengths of the Δx and Δy values of candidates inside the search ordering. Line 10 shows that the bit length of Δx (*xLen*) is computed via the complementary relationship between $I(\Delta x)$ and $I(\Delta y)$, Eq. (12). Subsequent combinations of Δx and Δy can be found by decrementing *xLen* and incrementing *yLen*.

An inner loop iterates over all bit length combinations of the Δx and Δy elements. For each combination, the surface parameters are computed using the CLIPRANGE function (Fig. 8). The sign of the offset is adjusted for each invocation of the CLIPRANGE function according to the direction of the range. Lower and upper boundaries are also specified based on the quadrant of the range and whether or not the zero column or row is included.

The numerator, *num*, of the lower bound of the range is computed based on Eq. (19). It is also reused to compute the upper bound of the range. The function CLIP(v, *min*, *max*) will clip the value v to allow it to remain between *min* and *max* (the boundaries of the search area).

The ADDCANDSFROMSURFACE function assembles same-cost surfaces by adding all the candidates found inside the previously computed ranges to the search ordering. The search ordering, the parameter O is a list of MVs representing the cost-based search ordering, with all lists passed by reference. The VEC function is used to create a new vector structure. This structure contains the cost so the RCSEA does not need to compute the cost of the candidates.

An optimization is added to the SEARCHORDERING function on line 19. For a given cost, if the horizontal boundaries of the same-cost surfaces touch the outer horizontal limits of the search area, the starting bit length *yStart* is incremented. This indicates that no subsequent areas are possible for all rows of the current bit length, as depicted in the example of Fig. 10.

```
ADDCANDSFROMSURFACE(O, T, B, L, R, c)
 1  for y = B.low to B.up − 1
 2      for x = R.low to R.up − 1
 3          O[O.i] = VEC(x, y, c)
 4          O.i = O.i + 1
 5      for x = L.low to L.up − 1
 6          O[O.i] = VEC(−x, y, c)
 7          O.i = O.i + 1
 8  for y = T.low to T.up − 1
 9      for x = R.low to R.up − 1
10          O[O.i] = VEC(x, −y, c)
11          O.i = O.i + 1
12      for x = L.low to L.up − 1
13          O[O.i] = VEC(−x, −y, c)
14          O.i = O.i + 1
```

Fig. 9. The ADDCANDSFROMSURFACE function appends the MV and cost of the candidates inside the same-cost surface for each quadrant: top (*T*), bottom (*B*), left (*L*), and right (*R*) to the search ordering (*O*).

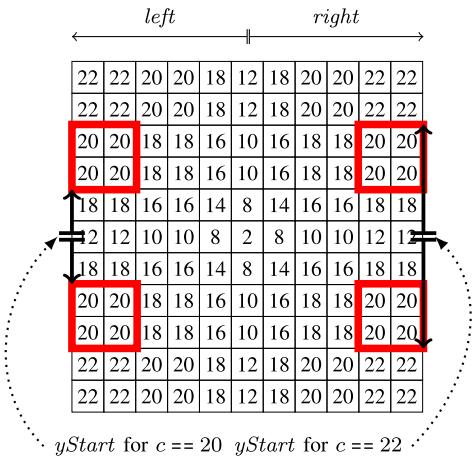


Fig. 10. When $c == 20$, the boundaries of the same-cost surfaces (red squares) touch the limits of the search area (*R.up* == *right* and *L.up* == *left*), thus no candidate of cost 22 can be encoded using less than 3 bits for the info part of its Δy component.

Fig. 10 illustrates that for a cost of 20, the conditions of line 19 of the SEARCHORDERING function are true; thus, the starting bit length *yStart* is incremented. This entails that for the next cost iteration, when cost equals 22, the starting row is now 4. It can be seen in Fig. 10, that no cost of 22 is present in the interval of rows $[-3, 3]$.

C. Early Termination Criterion

The increasing rate rule guarantees that the encoding cost of the MV of subsequent candidates in the search ordering will be greater than or equal to the MV encoding cost of the current candidate. This entails that if the weighted MV encoding cost of the current candidate is greater than the smallest rate-constrained candidate cost, then no other candidate will be able to produce a lower rate-constrained cost, and thus the search function should end.

It is important to note that this early termination criterion differs from the one proposed in [19]. The latter takes into consideration the ADS which is not possible in this work, because the candidates are sorted by cost, not by ADS.

Fig. 11 show the MOTIONESTIMATION function, also illustrated as the dashed box in Fig. 5, which terminates its BMA loop when the weighted-rate exceeds the current best cost (line 5).

```
MOTIONESTIMATION(O, i)
 1  bestCost = ∞
 2  bestVector = (0, 0)
 3  for v = 1 to O.length
 4      (x, y, cost) = O[v]
 5      if bestCost ⩽ λ × cost
 6          break
 7      if ADS(i, x, y) + λ × cost ⩽ bestCost
 8          cost = SAD(i, x, y) + λ × cost
 9          if cost ⩽ bestCost
10              bestCost = cost
11              bestVector = (x, y)
12  return bestVector
```

Fig. 11. Implementation of the ME algorithm combined with the RCSEA and the early termination criterion.

V. APPLICATIONS IN SUBOPTIMAL METHODS

Cost-based search orderings have potential applications in suboptimal methods. This section gives a glimpse of a potential application.

Suboptimal derivatives of the cost-based search ordering can be implemented by considering only a subset of each same-cost surface. For example, considering every other candidate, or skipping over a certain amount of candidates based on the desired granularity. By replacing all the for-loop statements in the ADDCANDSFROMSURFACE function by if statements with the same condition, we obtain fast suboptimal cost-based search ordering that considers only the first candidate of each same-cost surfaces (the candidate closest to the MVP). A detailed analysis of cost-based search ordering applications in suboptimal methods is outside the scope of this paper, but will be the topic of a future paper.

VI. EXPERIMENTAL RESULTS

To evaluate the algorithm presented in Section IV-B, we first compare it to the HEVC HM in full search mode to show that there is no impact on the Bjøntegaard delta peak signal-to-noise ratio (BD-PSNR) or Bjøntegaard delta rate (BD-Rate). Next, we compare the proposed algorithm to the HEVC HM with a standard RCSEA. These results show that the proposed algorithm offers more SAD savings and that early termination (discussed in Section IV-C) allows for additional speed up. Third, we show early termination results by quantization parameter (QP) and by block size to show their influence on early termination. Finally, we compare the proposed approach to the TZ-search algorithm and show the gains of replacing the semi-exhaustive fallback with the cost-based search ordering described in Section V.

To validate our results, we used 3 versions of the HEVC HM 16.2 encoder software [12]. The first version, which we will refer to as HM-FS, is the unmodified encoder software. It uses the built-in raster search ordering for block candidates during ME.

The second version, which we will refer to as HM-RCSEA, implements the RCSEA and represents a state-of-the art implementation of SEA in HEVC. As previously stated, raster search ordering is not commonly used in conjunction with RCSEA. As such, this version uses the spiral ordering found in the H.264 JM [27]. An example of this ordering is shown in Fig. 2b.

The third version, which we will refer to as HM-CBSEA, uses the same RCSEA implementation as version 2, but here, the search ordering is dynamically built using the SEARCHORDERING function found in Fig. 6. In addition, the BMA was modified to include the early termination criterion (MOTIONESTIMATION, Fig. 11).

The test conditions and software configurations used in our experiments conform to the common test conditions and software reference

TABLE I
ENCODING TIME SPEED UP (WITH EARLY TERMINATION), BD-PSNR AND BD-RATE BETWEEN HM-FS AND HM-CBSEA, FOR THE MAIN PROFILE AND RA SETTINGS

	RA Main			
Class	Sequence name	Speed $Up_{(ET)}$	BD-PSNR	BD-Rate
B (1920×1080)	Kimono	6.3	0.00	-0.04%
	ParkScene	5.5	0.00	-0.05%
	Cactus	6.3	0.00	-0.04%
	BQTerrace	4.9	0.00	0.10%
	BasketballDrive	5.5	0.00	0.18%
C (832×480)	RaceHorses	4.6	0.00	0.12%
	BQMall	6.7	0.00	-0.02%
	PartyScene	4.1	0.00	0.03%
	BasketballDrill	5.2	0.00	-0.01%
D (416×240)	RaceHorses	4.7	0.00	0.08%
	BQSquare	7.3	0.01	-0.12%
	BlowingBubbles	6.4	0.00	0.02%
	BasketballPass	6.0	0.00	0.00%
	Overall	5.8	0.00	0.02%

TABLE II
ENCODING TIME SPEED UP (WITH EARLY TERMINATION), BD-PSNR AND BD-RATE BETWEEN HM-FS AND HM-CBSEA, FOR THE MAIN PROFILE AND LD (WITH B FRAMES) SETTINGS

	LD B Main			
Class	Sequence name	Speed $Up_{(ET)}$	BD-PSNR	BD-Rate
B (1920×1080)	Kimono	5.4	0.00	0.01%
	ParkScene	4.8	0.00	-0.01%
	Cactus	5.6	0.00	0.02%
	BQTerrace	4.5	0.00	-0.01%
	BasketballDrive	5.0	0.00	0.00%
C (832×480)	RaceHorses	4.2	0.00	-0.03%
	BQMall	5.7	0.00	-0.01%
	PartyScene	3.7	0.00	0.09%
	BasketballDrill	4.6	0.00	-0.05%
D (416×240)	RaceHorses	4.1	0.01	-0.13%
	BQSquare	6.5	0.00	-0.10%
	BlowingBubbles	5.5	-0.01	-0.01%
	BasketballPass	5.2	0.00	0.00%
	Overall	5.0	0.00	0.01%

configurations [29]. All versions of the encoder software runs the main profile with 8-bit coding for both random access (RA) and low delay (LD) settings. The only changes to the standard configuration files are: ours enable full search, disable the fast encoder decision (FEN) and disable asymmetric motion partitions (AMP). We disabled the latter only to simplify implementation considerations, but AMP and SEA are compatible.

As for the classes of test sequences specified by Bossen [29], class A contains very high-resolution sequences which required several months of computations on a super computer cluster and exceeded our computational quota. Classes E and F are screen content and video conferencing, which dont reflect the use case in where a full search would be used (these are real-time applications). Classes B, C and D are more reasonable to process and show the benefits of the proposed method in the context of very high quality offline video processing (our application of interest).

A. Comparison With HEVC HM Full Search

In Tables I and II, we compare the encoding time speed up with early termination (ET), the BD-PSNR and the BD-Rate of HM-FS against the proposed solution (HM-CBSEA) for RA and LD settings, respectively. The speed up is measured as the encoding time ratio between HM-FS and HM-CBSEA.

TABLE III
THE PERCENTAGE OF SAD COMPUTATION SAVINGS, THE ENCODING TIME SPEED UP (WITHOUT EARLY TERMINATION), THE PERCENTAGE OF ITERATIONS PERFORMED BY THE BLOCK-MATCHING LOOP AND THE ENCODING TIME SPEED UP (WITH EARLY TERMINATION) BETWEEN HM-RCSEA AND HM-CBSEA, FOR THE MAIN PROFILE AND RA SETTINGS

	Proposed Solution vs RCSEA (RA Main)				
Class	Sequence name	SAD Savings	Speed Up	Itr. Performed	Speed $Up_{(ET)}$
B	Kimono	2.49%	1.5	28.48%	1.9
	ParkScene	1.87%	1.3	30.87%	1.8
	Cactus	2.23%	1.4	30.77%	1.9
	BQTerrace	1.63%	1.2	34.73%	1.7
	BasketballDrive	3.03%	1.4	33.37%	1.8
C	RaceHorses	3.64%	1.4	43.51%	1.6
	BQMall	4.28%	1.4	31.95%	1.9
	PartyScene	3.25%	1.3	48.76%	1.6
	BasketballDrill	3.43%	1.3	31.00%	1.7
D	RaceHorses	4.85%	1.3	45.23%	1.6
	BQSquare	9.60%	1.4	41.14%	2.0
	BlowingBubbles	7.36%	1.5	42.02%	2.0
	BasketballPass	7.20%	1.4	27.90%	1.9
	Overall	3.64%	1.4	36.33%	1.8

TABLE IV
THE PERCENTAGE OF SAD COMPUTATION SAVINGS, THE ENCODING TIME SPEED UP (WITHOUT EARLY TERMINATION), THE PERCENTAGE OF ITERATIONS PERFORMED BY THE BLOCK-MATCHING LOOP AND THE ENCODING TIME SPEED UP (WITH EARLY TERMINATION) BETWEEN HM-RCSEA AND HM-CBSEA, FOR THE MAIN PROFILE AND LD (WITH B FRAMES) SETTINGS

	Proposed Solution vs RCSEA (LD B Main)				
Class	Sequence name	SAD Savings	Speed Up	Itr. Performed	Speed $Up_{(ET)}$
B	Kimono	2.99%	1.5	38.90%	1.8
	ParkScene	1.31%	1.3	40.96%	1.7
	Cactus	2.24%	1.4	39.29%	1.8
	BQTerrace	1.44%	1.3	42.35%	1.7
	BasketballDrive	3.06%	1.4	43.40%	1.7
C	RaceHorses	3.51%	1.4	54.33%	1.6
	BQMall	3.78%	1.5	42.80%	1.8
	PartyScene	2.64%	1.4	60.20%	1.5
	BasketballDrill	3.04%	1.4	40.53%	1.7
D	RaceHorses	4.10%	1.5	57.39%	1.6
	BQSquare	7.63%	1.5	49.64%	2.0
	BlowingBubbles	5.05%	1.6	55.00%	1.8
	BasketballPass	6.18%	1.4	37.24%	1.8
	Overall	3.61%	1.4	46.31%	1.7

The proposed solution is approximately 5 times faster than the HM encoder. The data shows that the speed up is unaffected by resolution. Moreover, the gains vary between sequences.

We observed that the bit streams produced by HM-FS and HM-CBSEA are not identical. This is caused by ordering differences between same-cost candidates. In other words, when multiple global minimums exist, both encoders might not pick the same one. This being said, as shown in Tables I and II, this difference is negligible.

B. Comparison With RCSEA

Part of the speed up of HM-CBSEA over HM-FS is due to RCSEA. Tables III and IV compares HM-CBSEA with HM-RCSEA.

For Tables III and IV, the speed up is measured as the encoding time ratio between HM-RCSEA and HM-CBSEA (the latter including the precomputation of block sums). The RCSEA only accounts for half of the speed up offered by HM-CBSEA over HM-FS. In other words, our solution almost doubles the speed up that can be achieved with an implementation of the RCSEA. This speed up is achieved by

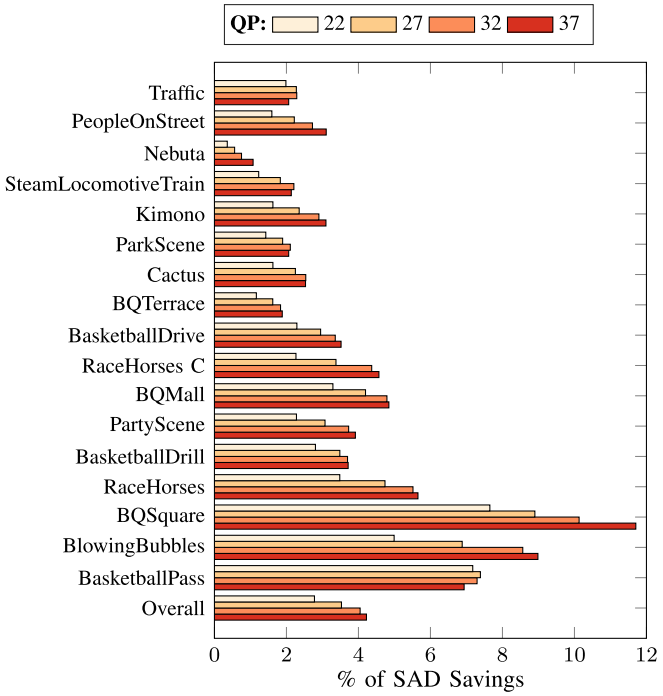


Fig. 12. Percentage of SAD operation savings, per sequence, for HM-CBSEA, when compared to HM-RCSEA, for the main profile and RA settings.

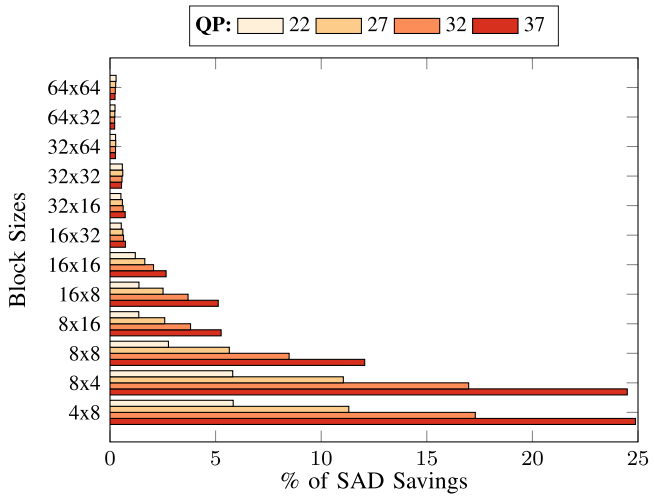


Fig. 13. Percentage of SAD operation savings, per block size, for HM-CBSEA, when compared to HM-RCSEA, for the main profile and RA settings.

the combination of two of our contributions, the cost-based search ordering algorithm and the early termination optimization.

The percentage of SAD operations saved is much lower than the percentage of iterations saved. However, performing a SAD requires $3 \times M \times N - 1$ operations (see Section II-C), whereas a saved iteration only accounts for a small amount of operations. The computational savings related to a saved iteration are rather limited: 2 operations for the ADS, the early termination check and the looping mechanism. The set of candidates saved by early termination is a subset of the set of candidates eliminated by the RCSEA. In other words, all candidates saved by early termination would not have required a SAD computation. This being said, the high percentage of saved iterations does have an impact on the encoding time.

The first part of the speed up is a direct result of the cost-based search ordering and the various considerations detailed in Section IV-B. By respecting the increasing rate rule in all encoding conditions, the number of SAD operations decreases by 3.64% on average for RA and 3.61% on average for LD, when compared to the H.264 JM search ordering. Moreover, as explained in Section IV, being cost-based the motion vector cost (Eq. (3)) is not computed during the BMA. Furthermore, contrary to HM-RCSEA, the proposed solution is not burdened with the supplementary computations related to off-centered search areas, as described in Section III-C.

Both Fig. 12 and Fig. 13 show that SAD savings increase when the QP increases. This is in line with the findings of Coban and Mersereau [14] to the effect that an increase in the Lagrange multiplier has a direct impact on transitive elimination in a rate-constrained context.

The SAD savings shown in Fig. 12 and Fig. 13 represent SAD operations that were performed as a result of the impairment of the transitive elimination. As such, we see significant savings in situations where transitive elimination is most effective.

Fig. 12 also indicates that savings decrease when the resolution increases. This suggests that the increasing rate rule is less problematic for high-resolution video sequences. When the increasing rate rule is less problematic, problems derived from it, such as asymmetric distributions of MV costs and the off-centered search area, are also less problematic. Conversely, for smaller video resolutions, the increasing rate rule, asymmetric distribution of MV costs and off-centered search areas are more problematic.

In Fig. 13, a significant increase in SAD savings is observed when smaller block sizes are used by the BMA. The obvious reason for this is that the size of the block directly affects the SAD values. This changes the ratio between the SAD and the weighted rate. The rate constraint becomes a more significant part of the transitive elimination. Another reason for this is that smaller size blocks will often yield more precise single-value projections. This in turn makes the ADS a more precise lower bound, again making transitive elimination more efficient.

C. Influence of Early Termination

The second part of the speedup indicates that not all iterations of the BMA need to be performed. Tables III and IV indicate that, on average, the proposed solution performs less than half of the BMA iterations. The early termination criterion relies on two important factors: the precision of the ADS as an estimate for the SAD and the ratio between the rate constraint and the SAD.

In Fig. 14, the percentage of iterations performed by the BMA loop is shown by sequence. This data does not suggest a relationship between the number of iterations performed and the resolution of the video sequence. For example, there is less than a 2% difference in the average number of iterations performed between the class D (416×240) and the class C (832×480) versions of the *RaceHorses* sequence.

A relationship clearly exists between the percentage of iteration performed and the QP. Higher QP values require fewer iterations of the BMA, and here again, the Lagrange multiplier is the cause. This is not surprising since increasing the weighted rate causes the early termination to occur earlier in the BMA loop.

As shown in Fig. 15, early termination is particularly effective with small size blocks, where two factors favor the early termination criterion. First, small blocks contain fewer values, and the spatial correlation of natural images reduces the occurrences of strong dualities in small blocks, generally leading to a small distance between the ADS and the SAD. Second, fewer values also implies smaller SAD, increasing the ratio between the rate constraint and the SAD.

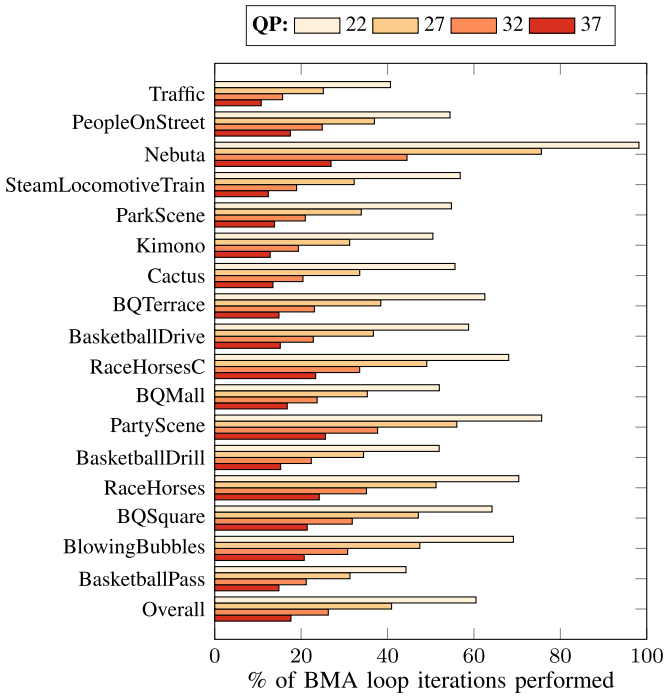


Fig. 14. Sequence-wise results for the percentage of iterations performed by the block-matching loop in the proposed solution. A value of 100% indicates that all the iterations of the loop are performed.

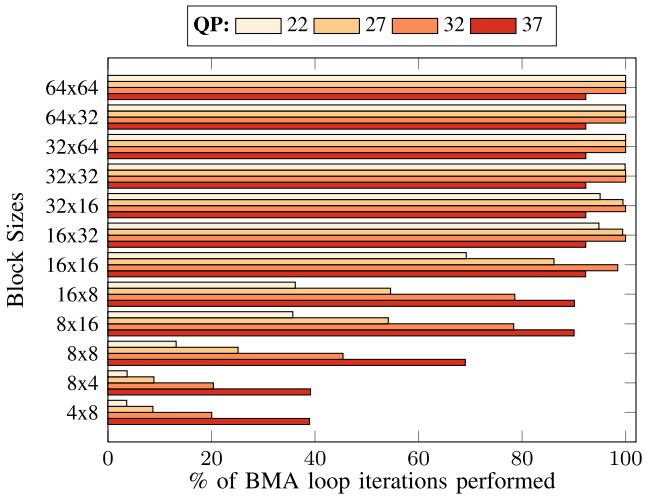


Fig. 15. Block-wise results for the percentage of iterations performed by the block-matching loop in the proposed solution. A value of 100% indicates that all the iterations of the loop are performed.

D. Comparison With Suboptimal Algorithms

When compared to the suboptimal TZ-Search algorithm found in the HEVC HM 16.2 encoder software [12], HM-CBSEA is about 5x slower, as shown in Fig. 16. HM-CBSEA is itself 5x faster than HM-FS, as presented in Tables I and II. In a situation where the increase in bit rate resulting from a suboptimal algorithm is unacceptable, the proposed algorithm offers the same results as an exhaustive search.

E. Application in Suboptimal Methods

Finally, we present a small experiment to show a glimpse of the potential of applying the concepts presented in this paper to suboptimal algorithms. In this experiment, we replace the TZ-Search's semi-exhaustive fallback with a suboptimal cost-based search ordering

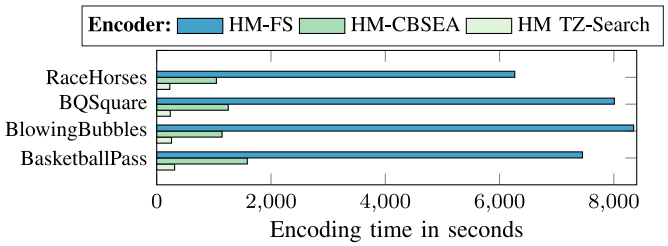


Fig. 16. Encoding time in seconds, for HM-CBSEA, when compared to HM TZ-Search, for class D sequences using the main profile and RA settings.

TABLE V
INTEGER LEVEL MOTION ESTIMATION SPEED UP AND BD-RATE, OF THE TZ-SEARCH WITH THE SUBOPTIMAL COST-BASED SEARCH ORDERING COMPARED TO THE TZ-SEARCH WITH THE DEFAULT SEMI-EXHAUSTIVE FALLBACK. FOR CLASS C SEQUENCES USING THE MAIN PROFILE AND RA SETTINGS

SubOptimal CostBased vs TZ-Search (RA Main)		
Sequence name	Speed Up	BD-Rate
RaceHorses	1.45x	0.21%
BQMall	1.25x	0.22%
PartyScene	1.28x	-0.08%
BasketballDrill	1.37x	0.11%
Overall	1.34x	0.12%

described in Section V. This suboptimal cost-based search ordering considers the first candidate of every same-cost surface. As shown in Table V, this has a negligible impact on BD-Rate and considerably speeds up the integer level motion estimation time.

VII. CONCLUSION

In this paper, we exposed two problems related to the increasing rate rule: asymmetric distribution of MV costs and off-centered search areas. These problems cannot be solved efficiently by static search orderings based on geometric patterns, such as spiral searches.

We proposed a new model to build dynamic search orderings for BMA. From this model, we developed a fast algorithm capable of producing cost-based search orderings that are unaffected by asymmetric distributions of MV costs and off-centered search areas. These orderings not only improve elimination, but they allow for early-termination, a novelty for optimal algorithms

Our experiments show that the proposed solution is more than five times faster than the HEVC HM encoder in full search mode, with the same BD-PSNR. When compared to an HEVC HM encoder that would implement RCSEA, the proposed solution remains superior, with a 1.4x speedup. Furthermore, the proposed early termination only requires performing 36% and 46% of block-matching loop iterations for Random Access and Low Delay respectively and the number of SAD operations also decreased by approximately 3%. Finally, we measured an average of 1.34x speedup when we replaced the semi-exhaustive fallback in the TZ-Search by a suboptimal cost-based refinement. These results demonstrate the potential of our findings to be tailored to suboptimal algorithms. In an upcoming paper, we will show that when extending the proposed techniques to the context of TZ-Search even more significant speedups can be reached.

ACKNOWLEDGMENT

Computations were made on the Guillimin from McGill University, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation, NanoQuébec, RMGA and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT).

REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] 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, Jul. 2003.

[3] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[4] P. Hosur and K.-K. Ma, "Motion vector field adaptive fast motion estimation," in *Proc. Commun. Signal Process. (ICICS)*, 1999, pp. 7–10.

[5] 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 Vis. Commun. Image Process.*, vol. 4310, 2001, pp. 883–892.

[6] A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 10, pp. 934–947, Oct. 2002.

[7] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," *J. Vis. Commun. Image Represent.*, vol. 17, no. 2, pp. 264–290, 2006.

[8] "Advanced video coding for generic audiovisual services, series H: Audiovisual and multimedia systems infrastructure of audiovisual services—Coding of moving video," Int. Telecommun. Union, Geneva, Switzerland, ITU-T Recommendation H.264, May 2003.

[9] "High efficiency video coding, series H: Audiovisual and multimedia systems Infrastructure of audiovisual services—Coding of moving video," Int. Telecommun. Union, Geneva, Switzerland, ITU-T Recommendation H.265, Apr. 2013.

[10] N. Hu and E.-H. Yang, "Fast motion estimation based on confidence interval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1310–1322, Aug. 2014.

[11] H.-Y. C. Cheong and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. Int. Conf. Multimedia Expo (ICME)*, vol. 3. Baltimore, MD, USA, Jul. 2003, pp. 517–520.

[12] K. McCann *et al.*, "High efficiency video coding (HEVC) test model 16 (HM 16) improved encoder description," Joint Collaborative Team Video Coding (JCT-VC) ITU-T SG16 WP3 ISO/IEC JTC1/SC29/WG11, Strasbourg, France, Rep. JCTVC-S1002, 2014.

[13] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105–117, Jan. 1995.

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

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

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

[17] T. Toivonen and J. Heikkila, "Fast full search block motion estimation for H.264/AVC with multilevel successive elimination algorithm," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Singapore, Oct. 2004, pp. 1485–1488.

[18] M. Yang, H. Cui, and K. Tang, "Efficient tree structured motion estimation using successive elimination," *IEE Proc. Vis. Image Signal Process.*, vol. 151, no. 5, pp. 369–377, Oct. 2004.

[19] L. Trudeau, S. Coulombe, and C. Desrosiers, "An adaptive search ordering for rate-constrained successive elimination algorithms," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Québec City, QC, Canada, Sep. 2015, pp. 3175–3179.

[20] L. Trudeau, S. Coulombe, and C. Desrosiers, "Sub-partition reuse for fast optimal motion estimation in HEVC successive elimination algorithms," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 2003–2007.

[21] I. Seidel, L. H. Cancellier, J. L. Güntzel, and L. Agostini, "Rate-constrained successive elimination of Hadamard-based SATDs," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 2395–2399.

[22] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," Video Coding Experts Group (VCEG) ITU-T, Austin, TX, USA, Rep. VCEG-M33, 2001.

[23] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[24] V. Turaev, *Quantum Invariants of Knots and 3-Manifolds*. New York, NY USA: De Gruyter Stud. Math., 2010.

[25] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. Chichester, U.K.: Wiley, 2010.

[26] J. Cai and W. D. Pan, "Fast exhaustive-search motion estimation based on accelerated multilevel successive elimination algorithm with multiple passes," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Dallas, TX, USA, 2010, pp. 1190–1193.

[27] "H.264/AVC JM reference software version 18.5," Joint Video Team ISO/IEC MPEG ITU-T VCEG, Berlin, Germany, Rep., May 2013.

[28] L. Trudeau, S. Coulombe, and C. Desrosiers, "Rate distortion-based motion estimation search ordering for rate-constrained successive elimination algorithms," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 3175–3179.

[29] F. Bossen, *Common Test Conditions and Software Reference Configurations Output*, document JCTVC-L1100, Joint Collaborative Team Video Coding, Geneva, Switzerland, 2013.