Authors' accepted manuscript. Article published in *Signal Processing: Image Communication*, vol. 81 (2020)
https://doi.org/10.1016/j.image.2019.115719

Multi-Level Rate-Constrained Successive Elimination Algorithm Tailored to Suboptimal Motion Estimation in HEVC

Luc Trudeau, Stéphane Coulombe and Christian Desrosiers
Department of Software and IT Engineering

Abstract

In the context of motion estimation for video coding, successive elimination algorithms (SEAs) significantly reduce the number of candidates evaluated during motion estimation without altering the resulting optimal motion vector. Nevertheless, SEA is often only used in conjunction with exhaustive search algorithms (e.g., full search). In this paper, we combine the multi-level successive elimination algorithm (ML-SEA) and the rate-constrained successive elimination algorithm (RCSEA) and show that they can be advantageously applied to suboptimal search algorithms. We demonstrate that the savings brought about by the new multi-level RCSEA (ML-RCSEA) outweigh the pre-computational costs of this approach for the Test Zonal (TZ) Search algorithm found in the HM reference encoder. We propose a novel multi-level composition pattern for performing RCSEA on an asymmetric partitioning. We introduce a double-check mechanism for RCSEA, and show that on average, it avoids computing 71% of motion vector (MV) costs. We also apply the proposed ML-RCSEA to bi-predictive refinement search and leverage a cost-based search ordering to remove 56% of error metric computations, on average. When compared to the HM reference encoder, our experiments show that the proposed solution reduces the TZ Search time by approximately 45%, contributing to an average encoding time reduction of about 7%, without increasing the Bjøntegaard delta rate (BD-Rate).

Index Terms

Successive elimination algorithm (SEA), rate-constrained successive elimination algorithm (RCSEA), multi-level successive elimination algorithm (ML-SEA), multi-level rate-constrained successive elimination algorithm (ML-RCSEA), block-matching algorithm (BMA), high efficiency video coding (HEVC), H.265, video compression.

NOMENCLATURE

ADS	Absolute difference of sums.
AMP	Asymmetric motion partitioning.
BRTRR	Bi-predictive refinement time reduction ratio.
ETRR	Encoding time reduction ratio.
ILMVP	Integer level motion vector predictor.
IMETRR	Integer-level motion estimation time reduction ratio.
ML-RCSEA	Multi-level rate-constrained successive elimination algorithm.
ML-SEA	Multi-level successive elimination algorithm.

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). 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 (CFI), NanoQuébec, RMGA and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT). Emails: {luc.trudeau, stephane.coulombe, christian.desrosiers}@etsmtl.ca

MV	Motion vector.
MVP	Motion vector predictor.
RCADS	Rate-constrained absolute difference of sums.
RCBMA	rate-constrained block-matching algorithm.
RCSAD	Rate-constrained sum of absolute differences.
RCSEA	Rate-constrained successive elimination algorithm.
SAD	Sum of absolute differences.
SATD	Sum of absolute transformed differences.
SEA	Successive elimination algorithm.
SMP	Symmetric motion partitioning.

I. Introduction

Encoders compliant with feature-rich video compression standards such as H.264/MPEG-4 advanced video coding (AVC) [1] and H.265/high efficiency video coding (HEVC) [2] rely heavily on motion estimation (ME) to achieve their high rate distortion (RD) performance. As a result, ME algorithms are applied to more block sizes, to more anchor frames, and over bigger search areas than their former counterparts [3]. As shown in [4], the integer-level ME time increased 12-fold between the H.264 reference encoder and the HEVC reference encoder.

Approaches such as the successive elimination algorithm (SEA) [5] and its derivatives: rate-constrained successive elimination algorithm (RCSEA) [6], multi-level successive elimination algorithm (ML-SEA) [7] and fine granularity successive elimination (FGSE) [8], reduce the number of candidates on which the sum of the absolute differences (SAD) error metric is evaluated during ME, without altering its result. These approaches all rely on a low-complexity lower bound approximation of the error metric. When the approximated error of a candidate is higher than the current best, the error metric does not need to be computed, as it is greater than or equal to the approximation. In other words, the candidates for which such low-complexity lower bound approximation is larger than the lowest error metric obtained so far during the ME process can be quickly and safely eliminated, thus saving computations. The ML-SEA and the FGSE differ from SEA in that they partition the candidates and combine multiple lower bound approximations into a tighter lower bound approximation resulting in the safe elimination of more candidates.

While these above approaches predate modern feature-rich video compression standards, they are rarely used in modern encoders. One reason for this is that with them, the mathematical models employed need to be updated to support features found in modern video compression standards (i.e., rectangular and asymmetric partitioning). Another reason is that these algorithms target the exhaustive search algorithm (ESA) (e.g., full search). When SEA algorithms were first introduced, suboptimal search algorithms evaluated far fewer candidates, and thus, the overhead of SEA exceeded the benefits. The results presented in this paper demonstrate that this is not the case anymore for recent encoders that need to achieve high compression efficiency by implementing a feature-rich video encoding standard.

When suboptimal search algorithms encounter unpredictable movement, they will often fail to find the optimal solution, leading to an increase in residual energy, which requires more bits to encode [9], [10], [11]. To avoid this, and improve RD performance, some modern suboptimal algorithms will revert to a more exhaustive approach when encountering unpredictable movement [12], thereby increasing the number of candidates being evaluated.

The increase in the number of candidates evaluated by modern suboptimal ME algorithms led Hu and Yang [4] to propose the confidence interval-based motion estimation (CIME) method, which computes a confidence interval based on the sum of block pixel values. These sums are also used in the lower bound approximation of the SEA. Based on these confidence intervals, a suboptimal ME algorithm can avoid evaluating the error metric on candidates that might not produce a better result. When implemented in HM, the CIME reduces integer-level suboptimal ME time by 70%. However, it increases the Bjøntegaard delta rate (BD-Rate) [13] by 1.0%. Similarly, Gao et al. [14] achieved a 56% suboptimal ME time reduction, offset by a 1.2% increase in BD-Rate, by iteratively building quadratic models to restrain the search area used by suboptimal ME.

In [15], the authors proposed a motion classification-based fast ME algorithm. First, they investigated the motion relationship of neighboring blocks and their coding cost to categorize the prediction units (PUs) into one of three classes: motion-smooth PU, motion-medium PU and motion-complex PU. Then, they carefully designed various search strategies for each class according to their respective motion and content characteristics. Finally, they proposed a fast search priority-based partial internal termination scheme to rapidly skip impossible positions. Their algorithm achieves as much as 12.47% and 20.25% total encoding time reductions when compared to the Test Zonal (TZ) Search [12] under Low-delay P Main (LD P Main) and Random access Main (RA Main) configuration respectively. But although the time reduction comes with a small BD-Rate increase of 0.17% for LD P Main, the BD-Rate increase for RA Main is notable at 1.12%. The authors compared their algorithm with two state-of-the-art methods under the same test conditions. Their results were better than those of Nalluri et al. [16] and of Hu and Yang [4] who respectively obtained, for similar time reductions, BD-Rate increases of 0.58% and 0.69% for LD P Main and 1.42% and 1.61% for RA Main.

More recently, in [17], the authors proposed an optimal fast motion estimation (FME) algorithm based on a rate-complexity-distortion (R-C-D) analytical model. They extended the traditional R-D model by adding the ME complexity into it, which enables expressing the R-D performance under different complexity constraints. Based on this R-C-D model, their proposed FME determines the R-C-D optimized search ranges for some representative PUs, which are then extended or refined dynamically according to motion characteristics for neighboring PUs. Their proposed FME achieves, on average, a 12% total encoding time reduction with a -0.2% BD-Rate when compared to the TZ Search under the LD P Main configuration and a 23% time reduction with a 0.3% BD-Rate under the RA Main configuration. However, on some sequences, the BD-Rate penalty can be close to, and even exceed, 1%. Consequently, developing methods providing better time reduction without increasing the BD-Rate remains an important research problem.

In this paper, we propose a new multi-level rate-constrained successive elimination algorithm (ML-RCSEA) tailored to the coding tools specified by HEVC with regards to ME, characterized by large block sizes, and rectangular and asymmetric partitioning. In contrast to our previous work [18], [19], the proposed solution is intended for suboptimal ME algorithms (e.g., TZ Search). A key benefit of the method is that it doesn't noticeably alter the BD-Rate of the ME method to which it is

applied but can reduce significantly its computational complexity. In the context of this paper, we applied our method to the TZ Search algorithm, as it is the suboptimal ME algorithm present in the HEVC reference encoder, and also because it is used for comparisons with state-of-the-art approaches [4], [14], [15], [16], [17]. However, our findings also apply to other suboptimal ME algorithms such as the one recently proposed in [17]. Furthermore, the methods presented in this paper can be applied to fast ME in the context of Versatile Video Coding (VVC) [20] and to Intra Block Copy (IBC) [21] used for Screen Content Coding (SCC) [22], as will be further discussed in Sections VI-G and VI-H.

The structure and the contributions of this work are as follows:

- Section II describes mathematically the proposed ML-RCSEA using a new model designed for multiple block sizes and partition shapes. It also shows how such compositions can reduce computational complexity by eliminating more motion vector (MV) candidates.
- Section III presents the proposed multi-level composition patterns for performing ML-RCSEA on rectangular partitions and on asymmetric partitioning.
- Section IV describes the integration of ML-RCSEA with TZ Search. It analyzes the memory requirements for ML-RCSEA. It also demonstrates that the savings obtained from ML-RCSEA used with suboptimal search algorithms outweigh the pre-computational costs, and introduces a novel double-check mechanism that avoids useless MV cost computations.
- Section V adapts our prior work on cost-based search ordering to the refinement phase of bi-predictive search.
- Section VI presents and discusses the experimental results while Section VII concludes the paper.

II. Multi-level Rate-Constrained Successive Elimination Algorithm

We propose the acronym ML-RCSEA to describe the combination of ML-SEA [7] and RCSEA [6]. It can be shown that the steps described in [6] for applying a rate constraint to SEA can also be applied to ML-SEA.

In this section, we describe the core concepts of our ML-RCSEA, and to that end, we first define the fundamentals that will enable us to enhance the SEA equations. Next, we revisit the definitions of SEA [5], ML-SEA [7] and RCSEA [6] such that these approaches are made compatible with the features found in modern video standards such as HEVC. The proposed equations in this section are novel, and allow for greater flexibility when compared to the original equations.

A. Fundamentals of the Proposed Model

We propose to model blocks and partitions as pixel coordinate pairs rather than directly as pixel values. This is not uncommon in the field of video processing, and although the proposed notation is novel, it is strongly inspired by that of [23]. Pixel coordinate pairs simplify equations that operate over blocks or partitions of different shapes and sizes, as will be apparent in eq. (7) and eq. (18). The proposed notation is illustrated in Fig. 1.

As in [23], we define $\psi(\mathbf{x})$ to be the pixel value in the current frame at position $\mathbf{x} = (x, y)$. Furthermore, we make use of bold uppercase letters (e.g., $\mathbf{X}$) to refer to a set of pixel coordinate pairs. Next, we denote partition primitives for a square

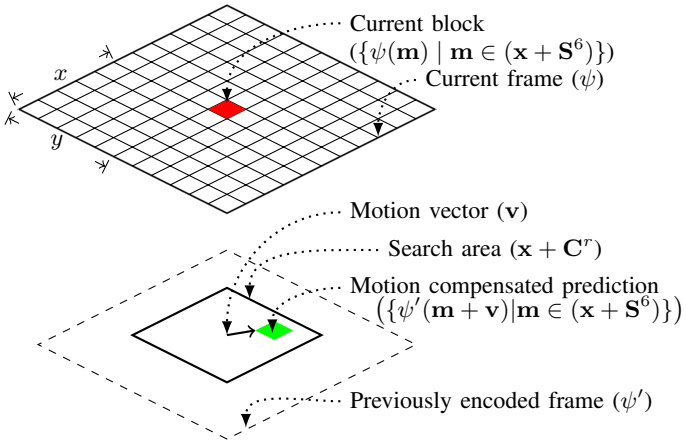


Fig. 1: Elements and notations of motion estimation and motion compensation

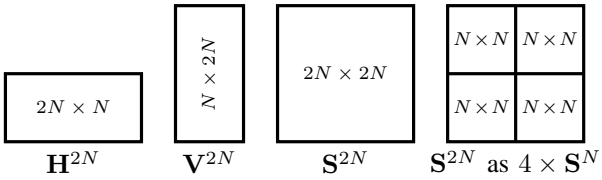


Fig. 2: Symmetric motion partitions used in HEVC

partition ($\mathbf{S}^k$), an horizontal rectangular partition ($\mathbf{H}^k$) and a vertical rectangular partition ($\mathbf{V}^k$):

$$\mathbf{S}^k = \left\{(x,y) \mid 0 \leqslant x,y < 2^k\right\} , \tag{1}$$

$$\mathbf{H}^k = \left\{(x,y) \mid 0 \leqslant x < 2^k, 0 \leqslant y < 2^{k-1}\right\} , \tag{2}$$

$$\mathbf{V}^k = \left\{(x,y) \mid 0 \leqslant x < 2^{k-1}, 0 \leqslant y < 2^k\right\} . \tag{3}$$

These partitions are illustrated in Fig. 2. In HEVC, block sizes range from 8×8 to 64×64, and so $k \in \mathbb{K} \triangleq \{3,4,5,6\}$.

We refer to an arbitrary block or partition $\mathbf{P}$ at location $\mathbf{x} = (x,y)$ as $\mathbf{x} + \mathbf{P}$. For this to be valid, we define the element-wise pair addition operator as:

$$(x,y) + \mathbf{X} = \{(x+i, y+j) \mid (i,j) \in \mathbf{X}\} . \tag{4}$$

This is similar to a translation vector used for geometric transformations in the field of computer graphics [24]. Like conventional addition, element-wise pair addition is both associative and commutative.

It should also be noted that we use styled capital letters (e.g., $\mathcal{P}$) to refer to a set of sets of pixel coordinate pairs. The element-wise pair addition operator applies to a set of sets of pixel coordinate pairs as follows:

$$(x,y) + \mathcal{P} = \{(x,y) + \mathbf{X} \mid \mathbf{X} \in \mathcal{P}\} . \tag{5}$$

Let $\mathbf{C}^r$ be a set of pixel coordinate pairs of the candidate blocks in a search area of radius r (using the Chebyshev distance from the center), such that:

$$\mathbf{C}^r = \{(x,y) \mid -r \leqslant x,y \leqslant r\} . \tag{6}$$

In compliance with the value found in the HM reference encoder implementation [12] and without loss of generality, we use $r = 64$ for the experimental results.

We can see in Fig. 1 that the current block $\psi(\mathbf{m})$ is composed of pixels at positions $\mathbf{m} \in (\mathbf{x} + \mathbf{S}^6)$. It is thus a block of 64×64 pixels. The set of candidate MVs for that block consists of all blocks of the form $\psi'(\mathbf{m} + \mathbf{v})$ where $\mathbf{m} \in (\mathbf{x} + \mathbf{S}^6)$ (same block size as the current block) and $\mathbf{v}$ is within a search area $\mathbf{x} + \mathbf{C}^r$.

B. Successive Elimination Algorithm

With these fundamentals, we define the function $\text{SAD}(\mathbf{P}, \mathbf{v})$. It computes the SAD of the set of pixel values referenced by $\mathbf{P}$, between the current frame and the candidate at position $\mathbf{v} \in \mathbf{C}^r$ relative to $\mathbf{P}$ in a previously encoded frame (ψ'):

$$\text{SAD}(\mathbf{P}, \mathbf{v}) = \sum_{\mathbf{m} \in \mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m} + \mathbf{v})| \,. \tag{7}$$

In this work, we assume that the SAD is used as the error metric for block matching at the integer-level. This is common practice in modern encoders [25], [12] because of the simplicity of the SAD. This simplicity is crucial, considering the number of computations that take place.

The contributions of the SEA to the block-matching algorithm (BMA) are twofold. The first is the transitive elimination of candidates that cannot be a better match than the current best one. This is achieved using a lower bound approximation of the error metric. The lower bound of the SAD is the absolute difference of sums (ADS), such that:

$$\underbrace{\sum_{\mathbf{m} \in \mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m} + \mathbf{v})|}_{\text{SAD}(\mathbf{P}, \mathbf{v})} \geqslant \underbrace{|\sum_{\mathbf{m} \in \mathbf{P}} \psi(\mathbf{m}) - \sum_{\mathbf{m} \in \mathbf{P}} \psi'(\mathbf{m} + \mathbf{v})|}_{\text{ADS}(\mathbf{P}, \mathbf{v})} \,. \tag{8}$$

The conventional ME search process successively evaluates the error metric associated with each candidate MV and keeps the one with the lowest error. Let $\mathbf{F} \subseteq \mathbf{C}^r$ be the set of candidates evaluated so far in the ME search process. We define $\hat{\mathbf{v}}$ as the MV pointing to the current best candidate, such that:

$$\hat{\mathbf{v}} = \underset{\mathbf{c} \in \mathbf{F}}{\arg\min} \sum_{\mathbf{m} \in \mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m} + \mathbf{c})| \,. \tag{9}$$

Note that, without loss of generality, when multiple MVs minimize the function expressed in eq. (9), we keep the first one found.

When the approximated error of a candidate is higher than that of the current best, the error metric does not need to be

computed, as it will be greater than the current best by transitivity [5]

$$|\sum_{\mathbf{m}\in\mathbf{P}} \psi(\mathbf{m}) - \sum_{\mathbf{m}\in\mathbf{P}} \psi'(\mathbf{m}+\mathbf{v})| > \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\hat{\mathbf{v}})| \tag{10}$$

$$\overset{eq.\ (8)}{\Longrightarrow} \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\mathbf{v})| > \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\hat{\mathbf{v}})| \tag{11}$$

$$\Longrightarrow \mathbf{v} \neq \underset{\mathbf{c}\in\{\mathbf{F},\mathbf{v}\}}{\arg\min} \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\mathbf{c})| \tag{12}$$

$$\Longrightarrow \mathbf{v} \neq \underset{\mathbf{c}\in\mathbf{C}^r}{\arg\min} \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\mathbf{c})| \,. \tag{13}$$

To highlight the fact that the pixel sum (PS) and the reconstructed pixel sum (RPS) of the ADS can be pre-computed, we define the function as follows:

$$\text{ADS}(\mathbf{P},\mathbf{v}) = |\text{PS}(\mathbf{P}) - \text{RPS}(\mathbf{P},\mathbf{v})| \,. \tag{14}$$

The function $\text{PS}(\mathbf{P})$ returns the pixel sum of $\mathbf{P}$ in the current frame; whereas the function $\text{RPS}(\mathbf{P},\mathbf{v})$ returns the reconstructed pixel sum of $\mathbf{P}$, but with a spatial offset of $\mathbf{v}$, in a previously encoded frame:

$$\text{PS}(\mathbf{P}) = \sum_{\mathbf{m}\in\mathbf{P}} \psi(\mathbf{m}) \,, \tag{15}$$

$$\text{RPS}(\mathbf{P},\mathbf{v}) = \sum_{\mathbf{m}\in\mathbf{P}} \psi'(\mathbf{m}+\mathbf{v}) \,. \tag{16}$$

The second contribution of the SEA is that it allows $\text{PS}(\mathbf{P})$ and $\text{RPS}(\mathbf{P},\mathbf{v})$ to be computed a priori and used as sum buffers during block matching. As such, after pre-computing, the functions $\text{PS}(\mathbf{P})$ and $\text{RPS}(\mathbf{P},\mathbf{v})$ only require two lookup operations, which considerably reduces the number of operations required to compute the ADS as compared to the SAD. This will be justified in Section IV-B. A more detailed analysis of memory requirements is presented in Section III-A.

In summary, *transitive elimination*, uses a lower bound approximation of the error metric in order to safely eliminate MV candidates without having to compute the error metric. This technique is appealing when such lower bound approximation is much less complex to compute than the metric of interest (e.g., benefiting from precomputations). For instance, in this paper, the ADS and the MADS presented in Section II-C can be used as low-complexity lower bound approximations of the SAD.

C. Multi-Level Successive Elimination Algorithm

ML-SEA splits a block into partitions. Implementing an ML-SEA-derived algorithm [7], such as the one proposed in this paper, in a video encoder that heavily relies on block partitions presents many advantages which are shown in this subsection. First, the ML-SEA offsets the increase in computation required to evaluate all the candidates in each partition. Second, the levels in ML-SEA do not require supplemental computations, as their sums can be obtained from the sum buffers intended for sub-partitions.

As the size of the PU increases, so does the average distance between the SAD and the ADS of the search candidates. This considerably reduces the efficiency of transitive elimination for PU sizes greater than 16×16 [19]. To address this issue, Gao

et al. [7] proposed to split a square block of size $2^k \times 2^k$ into four square partitions of size $2^{k-1} \times 2^{k-1}$, as follows:

$$\mathcal{S}^k = \left\{ \begin{array}{ll} (0,0) + \mathbf{S}^{k-1}, & (2^{k-1},0) + \mathbf{S}^{k-1}, \\ (0,2^{k-1}) + \mathbf{S}^{k-1}, & (2^{k-1},2^{k-1}) + \mathbf{S}^{k-1} \end{array} \right\} . \tag{17}$$

The partitioning of $\mathbf{S}^k$ into four $\mathbf{S}^{k-1}$ sub-partitions is illustrated in Fig. 2 (rightmost square). It is important to note that $\mathcal{S}^k$ is a set of spatially shifted sub-partitions of the partition $\mathbf{S}^k$. Therefore, $\mathcal{S}^k$ and $\mathbf{S}^k$ cover the same pixel region. It follows that $\mathrm{SAD}(\mathbf{S}^k, \mathbf{v}) = \mathrm{SAD}(\mathcal{S}^k, \mathbf{v})$ and $\mathrm{ADS}(\mathbf{S}^k, \mathbf{v}) = \mathrm{ADS}(\mathcal{S}^k, \mathbf{v})$.

The multi-level ADS (MADS) is obtained by summing the ADS of each partition:

$$\mathrm{MADS}(\mathcal{P}, \mathbf{v}) = \sum_{\mathbf{P} \in \mathcal{P}} \mathrm{ADS}(\mathbf{P}, \mathbf{v}) . \tag{18}$$

Splitting the block into partitions reduces the number of pixels per partition, thus lowering the average distance between the error metric and the approximated metric, and increasing the efficiency of transitive elimination. It follows that:

$$\mathrm{ADS}(\mathbf{S}^k, \mathbf{v}) \leqslant \mathrm{MADS}(\mathcal{S}^k, \mathbf{v}) \leqslant \mathrm{SAD}(\mathbf{S}^k, \mathbf{v}) . \tag{19}$$

Thus, from eq. (19), the distance between SAD and MADS is on average smaller than the distance between SAD and ADS, leading to less MV candidates to evaluate and reduced computations.

The FGSE [8] builds upon this, and allows the set $\mathcal{S}^k$ to contain a combination of $\mathbf{S}^{k-1}$ and $\mathcal{S}^{k-1}$.

Interestingly, matching the partitioning scheme of ML-SEA with that of the coding tree unit (CTU) results in fewer sum buffers and in a more efficient transitive elimination as compared to SEA alone. ML-SEA requires fewer sum buffers because bigger PUs add up sum buffers from smaller partitions, whereas the smallest PUs do not require partitioning, as the average distance between the ADS and SAD in them is already small.

For ML-SEA, there is clearly a trade-off between computational savings and the size of the partitions. On average, smaller partitions will improve the lower-bound and reduce the number of SAD operations required. However, blocks segmented into smaller partitions require more computations in order to sum the ADS of each partition. Therefore, using of ML-SEA leads to diminishing returns in terms of computational savings.

D. Rate-Constrained Successive Elimination Algorithm

When evaluating candidates, modern block-matching algorithms use a rate constraint, such as the cost function J defined in [25], [12], which can more generally be written as:

$$\mathrm{J}(\mathbf{P}, \mathbf{v}) = \mathrm{SAD}(\mathbf{P}, \mathbf{v}) + \lambda \mathrm{R}(\mathbf{v}) . \tag{20}$$

In the previous equation, the function $\mathrm{R}(\mathbf{v})$ returns an approximation of the number of bits required to encode the MV related to the candidate at position $\mathbf{v}$ relative to $\mathbf{P}$. A weighting coefficient λ models the trade-off between rate and distortion.

The exact definition of $\mathrm{R}(\mathbf{v})$ and λ will vary based on the encoder implementation, as they are usually not part of video compression standards. For this paper, we will use the recommended definitions for the HEVC HM reference software encoder,

found in [12]. However, these values are not a requirement, and our work generally applies to block-matching algorithms that employ a rate constraint.

As described in [6], the rate constraint can also be applied to eq. (10), (11), (12) and (13), where $\hat{\mathbf{v}}$, the current best MV candidate so far, is now redefined as

$$\hat{\mathbf{v}} = \underset{\mathbf{c}\in\mathbf{F}}{\arg\min} \sum_{\mathbf{m}\in\mathbf{P}} |\psi(\mathbf{m}) - \psi'(\mathbf{m}+\mathbf{c})| + \lambda \mathrm{R}(\mathbf{c}) \,. \tag{21}$$

Based on these equations, we can derive the same implications as eq. (13) for the rate-constrained context:

$$\mathrm{ADS}(\mathbf{P},\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) > \mathrm{SAD}(\mathbf{P},\hat{\mathbf{v}}) + \lambda\mathrm{R}(\hat{\mathbf{v}}) \implies \mathbf{v} \neq \underset{\mathbf{c}\in\mathbf{C}^r}{\arg\min} \tag{22}$$

By transitivity, if, for a partition $\mathbf{P}$, the weighted ADS (i.e., ADS plus the weight $\lambda\mathrm{R}(\mathbf{v})$) of a candidate MV is greater than the current best cost so far, the candidate MV can safely be eliminated (without any impact on the final result), as its cost will also be greater than the current best cost.

E. Multi-Level Rate-Constrained Successive Elimination Algorithm

As mentioned, the proposed ML-RCSEA combines ML-SEA [7] and RCSEA [6] to make them compatible with the features found in modern video standards, such as HEVC, and best exploit their features (e.g., hierarchical partitioning) to reduce computations.

Adding $\lambda\mathrm{R}(\mathbf{v})$ to each term in eq. (19) (and considering eq. (18)) gives:

$$\mathrm{ADS}(\mathbf{S}^k,\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) \leqslant \mathrm{MADS}(\mathcal{S}^k,\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) = \sum_{\mathbf{P}\in\mathcal{S}^k} \mathrm{ADS}(\mathbf{P},\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) \leqslant \mathrm{SAD}(\mathbf{S}^k,\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) \,. \tag{23}$$

From the definition of eq. (21), we can derive the following implication for ML-RCSEA:

$$\sum_{\mathbf{P}\in\mathcal{S}^k} \mathrm{ADS}(\mathbf{P},\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) > \mathrm{SAD}(\mathbf{S}^k,\hat{\mathbf{v}}) + \lambda\mathrm{R}(\hat{\mathbf{v}}) \implies \mathbf{v} \neq \underset{\mathbf{c}\in\mathbf{C}^r}{\arg\min} \tag{24}$$

Furthermore, we also have:

$$\mathrm{ADS}(\mathbf{P},\mathbf{v}) + \lambda\mathrm{R}(\mathbf{v}) > \mathrm{SAD}(\mathbf{S}^k,\hat{\mathbf{v}}) + \lambda\mathrm{R}(\hat{\mathbf{v}}), \text{for any } \mathbf{P}\in\mathcal{S}^k \implies \mathbf{v} \neq \underset{\mathbf{c}\in\mathbf{C}^r}{\arg\min} \tag{25}$$

Consequently, from eq. (24) and eq. (25), we conclude that if, for a partition $\mathbf{S}^k$ and a candidate MV, the weighted ADS of any of its sub-partitions $\mathbf{P}\in\mathcal{S}^k$ or the weighted sum of their ADSs is greater than the current best cost for that partition, the candidate MV can safely be eliminated, as its cost for the partition will also be greater than the current best cost. Therefore, ML-RCSEA, like ML-SEA, achieves a more efficient transitive elimination as compared to SEA alone. It speeds up the ME process without affecting the BD-Rate. Furthermore, considering the $\lambda\mathrm{R}(\mathbf{v})$ term makes it applicable to modern video standards. The next section presents the proposed multi-level composition patterns applicable to various partitioning schemes.

III. Multi-Level Composition Patterns

In this section, we present the multi-level composition patterns for rectangular partitions and for asymmetric motion partitioning (AMP). We propose to use multi-level composition to compute the ADS of an AMP from sub-partitions also present in symmetric motion partitioning (SMP). The advantage of using this pattern, besides being generic, is that it does not require computing additional ADS buffers (sum buffers) or the management of offset and overlap-related complexities.

A. Practical Considerations: PU Sizes, Memory and Computations

We first present some practical considerations that influenced the choice of the proposed composition patterns. Indeed, we must consider all supported partition sizes, as well as the impact of the selected multi-level composition patterns on memory and computations.

Regarding the first aspect, Table I presents the PU block sizes supported in HEVC inter prediction. We must ensure that the proposed solution supports all these PU block sizes.

With respect to memory, let us consider reference frames of size $W \times H$ pixels and a partition size of $m \times n$ pixels. For each reference frame and partition size under consideration, we need to store the ADS associated with that partition size for every position it may have in the image. Clearly, since the motion estimation algorithm may need to compute the SAD with respect to a block of size $m \times n$ at any location in the reference frame, we have the same requirement for ADSs in RCSEA as well. Thus, for each reference frame, $W \times H$ ADS values must be stored. For an $m \times n$ partition, each ADS requires 2^{n+m+8} bits to represent the sum of these $m \times n$ 8-bit luma values. For instance, to support blocks of 64×64, 20 bits are required. For simplicity, we store ADS values using 32 bits regardless of the partition size. This also allows supporting partitions of 256×256 pixels at a bit depth of up to 16 bits, which is quite sufficient for current and near-future video standards such as the upcoming VVC [20]. In summary, the memory required for each partition size and reference frame is $4WH$ bytes, which collectively represent our sum buffers.

Considering, from Table I, that there are 12 symmetric motion partitioning (SMP) sizes and 12 asymmetric motion partitioning (AMP) sizes, we need $48WH\ bytes$ per reference frame to store all SMP sizes, and just as many for all the AMP sizes. Therefore, storing ADSs for AMP doubles the required memory as compared to SMP alone. Table II shows the memory requirements per reference frame for various resolutions for the symmetric and symmetric-asymmetric cases. Of course, we can limit the required ADS buffers to cover only the current group of pictures (GOP). For very large GOPs, we can further reduce the memory requirements to only contain frames that are identified by the encoder as possible references for the current frame. Overall, such memory requirements, even using several reference frames, are acceptable for today's encoding servers. In fact, professional servers today support hundreds of gigabytes of RAM. Another point worth mentioning is that the memory for sum buffers, regardless of the adopted approach, can be shared by the various threads in a parallel encoder, thus avoiding memory duplication. Nevertheless, as will be shown, in order to save memory, we will avoid computing asymmetric ADSs while still supporting their encoding.

We will show in Section IV-B that the number of operations required to compute the sum of all overlapping candidate blocks (i.e., computing the ADSs) for a reference frame of size $W \times H$ is approximately $4WH$ when the partition sizes are

small as compared to W and H (see eq. (27)). We will also show that this represents an average overhead of 1.2 to 1.35 SAD operations per block size, depending on the block size and image resolution. This overhead is quite small when considering the savings it brings to the HEVC HM's fast motion estimation algorithm (or other fast algorithms), which, on average, computes the SAD for approximately 100 positions per PU (see [17] Table V).

Regarding the current frame, for SMPs, only the sum of non-overlapping blocks must be computed and stored. This means that ADSs need to be computed and stored only for positions which are multiples of the block size under consideration. For instance, 16×8 blocks can only occur at positions which are multiples of 16 horizontally and multiples of 8 vertically, dramatically reducing the computational complexity, as well as the required memory compared for reference frames. However, the situation is more complicated for AMPs since they do not constitute non-overlapping blocks. Indeed, a 32×24 block may be positioned at the top of a block ($2N \times nD$) or shifted vertically by 8 pixels ($2N \times nU$) and therefore overlap. Their handling is thus more complicated, as will be shown in the next subsections.

TABLE I: PU block sizes in HEVC inter prediction ([26])

	Depth			
PU Size	0	1	2	3
$2N \times 2N$	64×64	32×32	16×16	8×8
$2N \times N$	64×32	32×16	16×8	8×4
$N \times 2N$	32×64	16×32	8×16	4×8
$N \times N$	32×32	16×16	8×8	
$2N \times nU$	64×16	32×8	16×4	
$2N \times nD$	64×48	32×24	16×12	
$nL \times 2N$	16×64	8×32	4×16	
$nR \times 2N$	48×64	24×32	12×16	

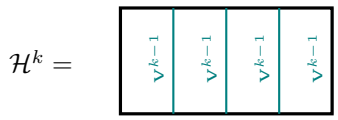
TABLE II: Memory required to store all the sum buffers for symmetric and symmetric-asymmetric partition sizes per reference frame for various video resolutions compared to the memory required by the proposed ML-RCSEA approach

Class	**Symmetric**	**Symmetric & asymmetric**	**Proposed ML-RCSEA approach**
B (1920×1080)	94.9 MB	189.8 MB	71.2 MB
C (832×480)	18.3 MB	36.6 MB	13.7 MB
D (416×240)	4.6 MB	9.1 MB	3.43 MB

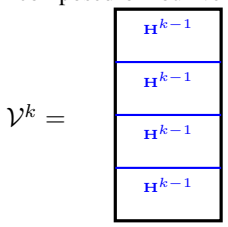
B. Symmetric Partitions

To improve transitive elimination on larger blocks, we propose a four-way sub-partitioning. Large square and rectangular blocks are decomposed, as shown in Fig. 2 and Fig. 3 respectively. As shown in Fig. 3, we propose to split horizontal rectangular partitions $\mathcal{H}^k$ into four vertical rectangular sub-partitions $\mathbf{V}^{k-1}$ and vertical rectangular partitions $\mathcal{V}^k$ into four horizontal rectangular sub-partitions $\mathbf{H}^{k-1}$. We do not split the smallest partitions, since the number of pixels is small and transitive elimination in that case is already very efficient. This approach is thus more efficient than the RCSEA, and reduces the number of sum buffers required as the biggest square and rectangular sum buffers (64×64, 64×32, and 32×64) are not required. We thus need 9 sum buffers instead of 12, thus reducing memory requirements by 25% while increasing performance

because of better transitive elimination. This is shown in Table II under the proposed ML-RCSEA approach, and compared against approaches storing all symmetric or symmetric-asymmetric partitions.



(a) Horizontal rectangular partition $\mathcal{H}^k$ composed of four vertical rectangular sub-partitions $\mathbf{V}^{k-1}$



(b) Vertical rectangular partitions $\mathcal{V}^k$ composed of four horizontal rectangular sub-partitions $\mathbf{H}^{k-1}$

Fig. 3: Proposed multi-level composition pattern for rectangular partitions

C. Asymmetric Partitioning

We propose a multi-level composition pattern for performing ML-RCSEA on an asymmetric motion partitioning (AMP) using pre-computed symmetric sub-partitions. This proposal resolves the issues encountered when combining AMP with ML-RCSEA. This is not to say AMPs and ML-RCSEA are incompatible, but rather, it only highlights issues that make combining the two approaches non-trivial. Our proposal first avoids the need for sum buffers associated with each AMP size and thus reduces the memory requirements by half. Then, for the current frame, it avoids issues related to partition offset and overlap, which complicate the computation of sum buffers. As can be seen in Fig. 4, an offset occurs when partitions are positioned at coordinates that are not a multiple of the partition size, whereas, an overlap occurs when partitions of the same size overlap each other. Finally, to increase transitive elimination, it is important to split the AMP into smaller ones, but this does not have to be done using asymmetric sub-partitions; it may well be using symmetric ones.

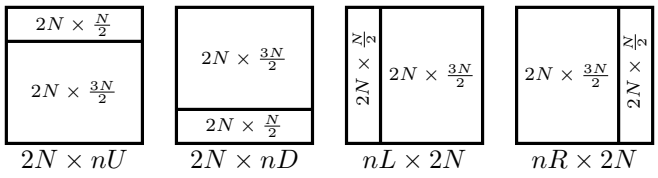


Fig. 4: AMP used in HEVC. Multiple new partition sizes must now be considered; some partitions are offset and others overlap. These are all issues for ML-RCSEA.

With the application of the proposed multi-level compositional pattern, illustrated in Fig. 5, an encoder using ML-RCSEA (i.e., for SMP) can perform AMP without computing additional sum buffers or having to manage offset and overlap-related complexities. The key concept behind our approach consists of the splitting of AMP into partitions also used in SMP. It allows efficient transitive elimination on asymmetric partitions while not requiring memory over and above what is needed to support symmetric ones. Thus, the proposed ML-RCSEA approach makes it possible to efficiently support both types of partitions while providing 62.5% memory savings (see Table II).

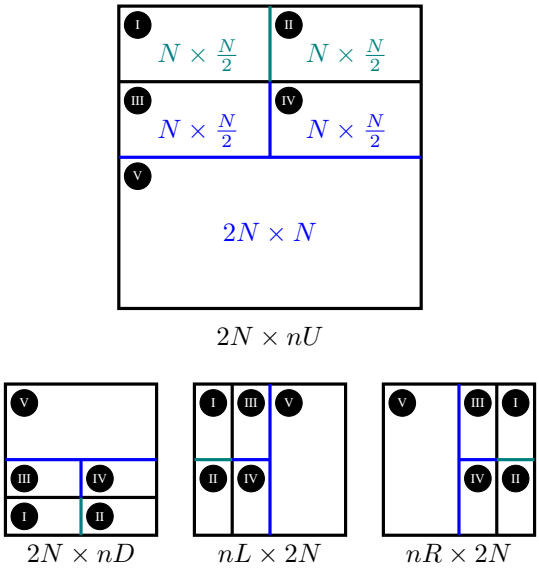


Fig. 5: Proposed multi-level composition pattern for performing ML-RCSEA on an AMP using pre-computed symmetric sub-partitions

To describe the proposed multi-level composition pattern, we use the $2N \times nU$ asymmetric partitioning scheme as an example (its illustration is bigger in Fig. 5). It should be recalled that in Fig. 4, the $2N \times nU$ partitioning scheme splits the PU into two partitions: the first of size $2N \times \frac{N}{2}$ and the second of size $2N \times \frac{3N}{2}$. As with SMP, these partitions are evaluated separately.

We refer to the first partition as $\mathbf{U}$, and its compositional pattern as $\mathcal{U}$. We split this partition into two $N \times \frac{N}{2}$ partitions, identified with the labels ❶ and ❷ in Fig. 5. An $N \times \frac{N}{2}$ partition of size k is equivalent to a $2N \times N$ partition of size $k-1$. A $2N \times N$ is a partition also used by symmetrical partitioning, $\mathbf{H}^{k-1}$, for which the sum buffer is already available. The compositional pattern of the $2N \times \frac{N}{2}$ partition is the combined sum of the two $2N \times N$ partitions of size $k-1$.

We refer to the second partition as $\mathbf{D}$, and its compositional pattern as $\mathcal{D}$. Just like the first partition, we split the top part of the $2N \times \frac{3N}{2}$ partition into two $N \times \frac{N}{2}$ partitions, identified with the labels ❸ and ❹ in Fig. 5. This split must be performed on the part of the partition that is next to the $2N \times \frac{N}{2}$ partition, as the remainder will be a symmetrical $2N \times N$, identified with the label ❺. The compositional pattern of the $2N \times \frac{3N}{2}$ partition is the combined sum of the two $2N \times N$ partitions of size $k-1$ and the $2N \times N$ partition of size k. The labels show the corresponding splits for the other asymmetric partitioning schemes.

As a final note, we would like to emphasize that the proposed composition patterns are merely intented to illustrate the application of ML-RCSEA in the context of HEVC. Other composition patterns may be proposed without departing from our original idea and will offer other compromises in terms of memory, computations and implementation complexity. However, none of these choices will affect the performance in terms of the BD-Rate.

IV. ML-RCSEA INTEGRATION WITH TZ SEARCH

In this section, we present the details involved in the integration ML-RCSEA with TZ Search. We first give a brief overview of the test zonal (TZ) search algorithm. We also demonstrate that with modern frame resolutions and block sizes, very small

SAD operation savings are needed to compensate for computing the sum buffers required by SEA. Further, we present a novel approach to RCSEA that avoids computing the $\mathrm{R}(\mathbf{v})$ function in situations where it does not contribute to transitive elimination.

A. Overview of the TZ Search Algorithm

The TZ Search algorithm is the prominent search algorithm implemented in the HEVC HM reference encoder. As such, this algorithm sets the bar for search algorithms destined to HEVC. TZ Search is used exclusively for integer-level ME and uses a 3-step search strategy:

1. Starting point selection:

The candidates evaluated to determine the starting point vary depending on the encoder configuration. By default, the median predictor and the zero motion predictor are used. In enhanced mode, the search algorithm will also consider three spatial neighboring blocks immediately located: to the left, above and the above and to the right. The best among them is selected as the starting point.

2. First search:

An arrangement of patterns of exponentially increasing sizes are used around the starting point to identify more candidates. The size increases until it reaches the edge of the search area or terminates if the minimum found so far as not been updated in the last 3 sizes. An example of the first five patterns is given in Fig. 6.

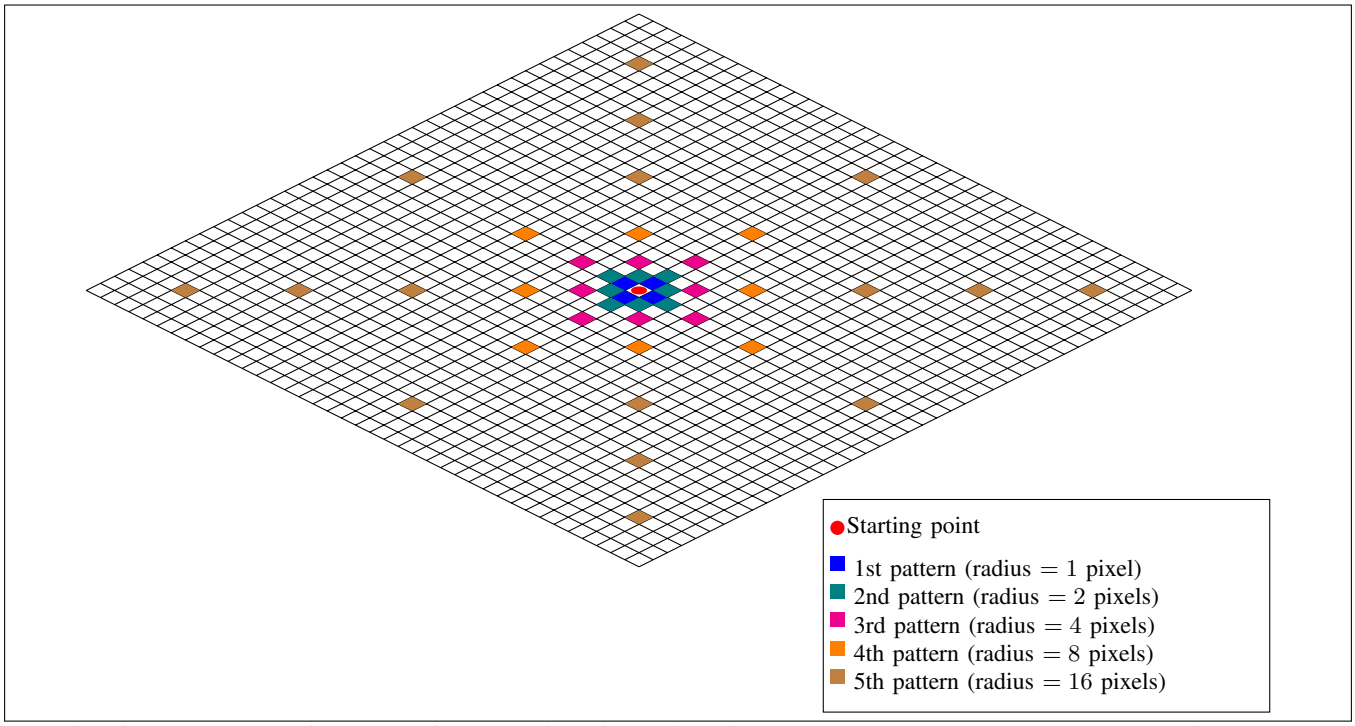


Fig. 6: Arrangement of patterns of exponentially increasing sizes used by TZ Search to evaluate a starting point.

The 6th and 7th patterns are the same as the 5th but are of 32 and 64 pixel radius respectively. In the case where the starting point is the best candidate, the search will terminate after the 3rd pattern.

3. Refinement search:

TZ Search also includes the concept of *search confidence*. Similar to *prediction confidence*, search confidence is based on the radius of the last pattern used to find the candidate. A large pixel radius indicates uncertainty as the density of candidates being evaluated is smaller.

The refinement used depends on the encoder configuration and the search confidence. It can either be a diamond search, a raster search or a star search. In the case of the star search, the starting point is set as the best current point and the previous step is repeated [12].

The algorithm thus loops over the last two steps until the best candidate remains the starting point. Concretely, more than many other suboptimal algorithms, the TZ Search evaluates a considerable number of candidates [9], [10], [11].

B. Integration and Justification for SEA in TZ Search

The integration of ML-RCSEA with TZ Search (or other suboptimal algorithms) is performed as follows. We evaluate the partitions and the MV candidates in the order proposed by TZ Search. However, the split of rectangular partitions into smaller rectangular sub-partitions is performed as described in Section III. Also, transitive elimination is performed to reduce the number of candidates for which the SAD is computed.

As previously explained, SEA-based algorithms are targeted at ESA because of the a priori computation cost of sum buffers. Although the TZ Search algorithm will not require the same amount of SAD operations for each PU, the number of evaluated candidates is, on average, high enough to justify the use of the proposed ML-RCSEA approach. To demonstrate this, we will establish a minimum SAD operation savings threshold.

Let $\mathcal{O}^{\text{SAD}}$ be the number of operations required to compute the SAD, such that:

$$\mathcal{O}^{\text{SAD}}(\mathbf{P}) = 3 \times |\mathbf{P}| - 1 \,, \tag{26}$$

where $|\mathbf{P}|$ is the cardinality (i.e., number of pixels) of the partition $\mathbf{P}$. Indeed, the SAD operation requires to compute a subtraction, an absolute value and an addition for each pixel. Similarly to eq. (11) in [6], let $\mathcal{O}^{\text{RPS}}$ be the number of operations required to compute the sum of all overlapping candidate blocks, for a reference frame of size $W \times H$, such that:

$$\mathcal{O}^{\text{RPS}}(\mathbf{P}) = \underbrace{(2W - \text{Cols}(\mathbf{P}) - 1)}_{\substack{\text{Sliding window} \\ \text{operations for} \\ \text{one row}}} H + \underbrace{(2H - \text{Rows}(\mathbf{P}) - 1)}_{\substack{\text{Sliding window} \\ \text{operations for} \\ \text{one column}}} W \,, \tag{27}$$

where $\text{Cols}(\mathbf{P})$ and $\text{Rows}(\mathbf{P})$ respectively return the number of columns and the number of rows in $\mathbf{P}$.

For example, computing the sum of all overlapping 8×8 blocks in a 416×240 frame requires the same number of operations as 2060 8×8 SAD operations. This number might seem prohibitive, but given that a 416×240 frame contains 1560 8×8 blocks, it corresponds to an average of 1.32 SAD operation per block.

At sufficiently high resolutions, the ratio between the number of operations required to compute the reconstructed pixel sums and the number of operations required to compute the SAD can be approximated as:

$$\frac{\mathcal{O}^{\text{RPS}}(\mathbf{P})}{\mathcal{O}^{\text{SAD}}(\mathbf{P})} \approx \frac{4WH}{3|\mathbf{P}|} \,. \tag{28}$$

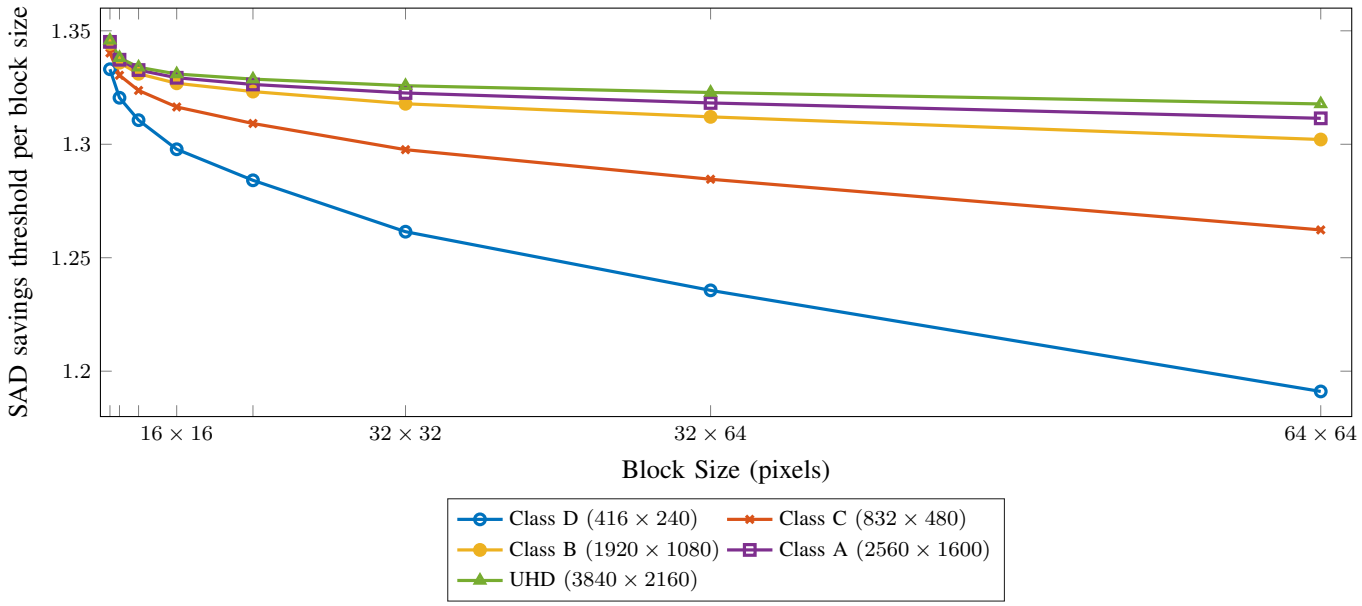


Fig. 7: SAD savings threshold per block for HEVC block/partition sizes and for common frame resolutions

This equation represents the number of SAD operations needed to compensate for summing all overlapping candidate blocks over the entire frame. At the block level, this can be approximated by a constant

$$\frac{\frac{4WH}{3|\mathbf{P}|}}{\frac{WH}{|\mathbf{P}|}} = \frac{4}{3}, \tag{29}$$

These approximations reveal that it may be reasonable for the minimum SAD savings threshold to be $\frac{4}{3}$. Fig. 7 shows that as the resolution increases, the SAD savings threshold tends towards $\frac{4}{3}$ for all blocks and partition sizes.

To compute the minimum SAD savings threshold for the whole frame, we sum the number of operations for all the HEVC block sizes except for $64, 64 \times 32, 32 \times 64$, because for these blocks, the ADS is summed from smaller blocks. The same applies for all the partition sizes related to AMP. It should also be noted that due to the hierarchical B-frame structure, for the RA Main profile, the sum buffers are computed only for even numbered frames, as odd numbered frames cannot serve as reference frames [12].

For an 832×480 frame, this threshold is approximately 54590 SAD operations. This might seem excessive, but as we show in Section VI-D, Table VI clearly demonstrates that the SAD operation savings per frame of our proposed method are well above the minimum SAD operation savings threshold. It should also be noted that the minimum SAD threshold is easily surpassed only by the savings on the TZ Search, and as a result, the SAD savings for bi-predictive search require no additional pre-computation.

C. Double-Check Mechanism for RCSEA in TZ Search

The solution proposed in [6] requires that $\mathrm{R}(\mathbf{v})$ be computed for every candidate. In many cases, the value of $\mathrm{R}(\mathbf{v})$ does not alter the outcome of the transitive elimination performed by RCSEA (eq. (22)). Depending on how $\mathrm{R}(\mathbf{v})$ is defined, this could be a costly operation.

We propose a double-check mechanism to avoid useless $\mathrm{R}(\mathbf{v})$ computations (i.e., when $\mathrm{R}(\mathbf{v})$ does not alter the outcome of eq. (22)). It is important to note that the rate term cannot simply be ignored when performing the RCSEA, as this would lead to false positive eliminations. We propose to extend eq. (22) as follows:

$$\begin{aligned} \mathrm{SAD}(\mathbf{P}, \hat{\mathbf{v}}) + \lambda \mathrm{R}(\hat{\mathbf{v}}) &\leqslant \mathrm{ADS}(\mathbf{P}, \mathbf{v}) + \lambda \mathrm{R}(\mathbf{p}) \\ &\leqslant \mathrm{ADS}(\mathbf{P}, \mathbf{v}) + \lambda \mathrm{R}(\mathbf{v}) \\ &\leqslant \mathrm{SAD}(\mathbf{P}, \mathbf{v}) + \lambda \mathrm{R}(\mathbf{v}) \,, \end{aligned} \tag{30}$$

where $\mathbf{p}$ is the position of the predicted MV. Note that $\mathbf{p}$ is the smallest MV cost in the search area

$$\mathbf{p} = \underset{\mathbf{v} \in \mathbf{C}^r}{\arg\min}\, \mathrm{R}(\mathbf{v}) \,. \tag{31}$$

If the first inequality of eq. (30) does not hold, then by transitivity, the candidate can safely be eliminated without computing $\mathrm{R}(\mathbf{v})$.

As we will describe in VI-B, Table IV clearly demonstrates that, in many cases, the MV cost does not alter the transitive elimination outcome.

V. Cost-Based Search Ordering For Bi-Predictive Search

Bi-prediction consists in building a prediction from two MVs. In this section, we propose to adapt our prior work on cost-based search ordering to the bi-predictive search.

In principle, a bi-predictive motion search algorithm could minimize the prediction error by searching for both MVs simultaneously. Because of the combinatorial aspect of this problem, modern encoders, such as the HM, resort to a greedy approach that uses an iterative uni-predictive search for the first MV, followed by a refined search for the second one [12].

This refinement is performed using a raster search algorithm over a small part of the search area; a raster search is used because the refinement surface is rather small (9×9). However, as shown in Fig. 8, our proposed enhancements to the TZ Search algorithm have reduced the number of SAD operations it produces to such an extent that the bi-predictive raster scan is now the dominant source of SAD operations performed by the HM.

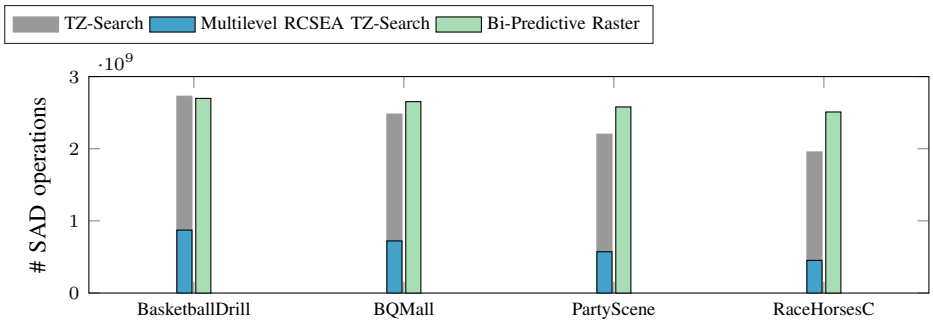


Fig. 8: Comparison of the number of SAD operations performed by the HM. After introducing ML-RCSEA, bi-predictive raster refinement now becomes the predominant source of SAD operations.

A raster search ordering results in a limited number of candidates being eliminated by transitivity. As explained in [27], [18], [19], the search ordering plays a crucial role in the efficiency of transitive elimination, and finding good candidates early on lowers the transitive threshold used for elimination. This in turn leads to a considerably more efficient transitive elimination than when good candidates are positioned further along in the search ordering.

SEA-based algorithms will often use a search ordering based on a spiral-shaped geometric pattern. This follows the assumption that the likelihood of finding the best-matched candidate would decrease with an increase in the cost of the MV [27]. However, as explained in [28], these small refinement search areas are prone to be off-centered. When a search ordering based on a static geometric pattern is used on an off-centered search area, it will not follow the previous assumption, as the center of the refinement zone will probably not be the predicted MV.

This situation is ideal for the cost-based search ordering approach we proposed in [28]. That search ordering is dynamic and adapts to off-centered search areas, in addition to guaranteeing that candidates will be evaluated by increasing MV costs. As such, the previous assumption is respected, thus increasing the likelihood of finding better candidates early on.

VI. Experimental Results and Discussion

The proposed approaches were implemented in version 16.8 of the HM reference encoder [12] (except in Section VI-E, where version 12.1 was used). All experiments were performed on a Dual Intel Sandy Bridge EP E5-2670 microprocessor. The test conditions and software configurations used in our experiments conform to common test conditions and software reference configurations [29]. Our tests mainly focus on video sequences from classes B (1920×1080) and C (832×480), for the Low-delay B Main (LD B Main) and the RA Main profiles. The only change we made to the standard configuration files was to disable the fast encoder decision (FEN), which we did solely to simplify the implementation; nevertheless, FEN and SEA are compatible. Simulations on class A were not performed as they would require several months to perform. Although state-of-the-art methods are usually tested using classes B, C, and D, we did not consider class D (416×240) since we believe that they are not of much practical interest nowadays (even though we would have performed better on those).

We use the encoding time reduction ratio (ETRR) and integer-level motion estimation time reduction ratio (IMETRR) metrics defined in [4], such that:

$$\text{ETRR} = \frac{\text{ET}_{\text{HM}} - \text{ET}_{\text{AP}}}{\text{ET}_{\text{HM}}}\,, \tag{32}$$

where ET_{HM} is the encoding time of the HEVC HM reference encoder and ET_{AP} is the encoding time of the approach being evaluated. Similarly, for integer-level motion estimation:

$$\text{IMETRR} = \frac{\text{IMET}_{\text{HM}} - \text{IMET}_{\text{AP}}}{\text{IMET}_{\text{HM}}}\,, \tag{33}$$

where IMET_{HM} is the integer-level motion estimation time of the HEVC HM reference encoder and IMET_{AP} is the integer-level motion estimation time of the approach being evaluated.

A. ML-RCSEA in TZ Search

This section describes the impact of ML-RCSEA on the TZ Search algorithm. In Table III, we present the percentage of SAD operations avoided by ML-RCSEA. In Fig. 9, we compare the number of them performed by the TZ Search algorithm with and without ML-RCSEA, for each frame of the *BasketballDrill* sequence encoded using the LD B Main profile and a quantization parameter (QP) of 22.

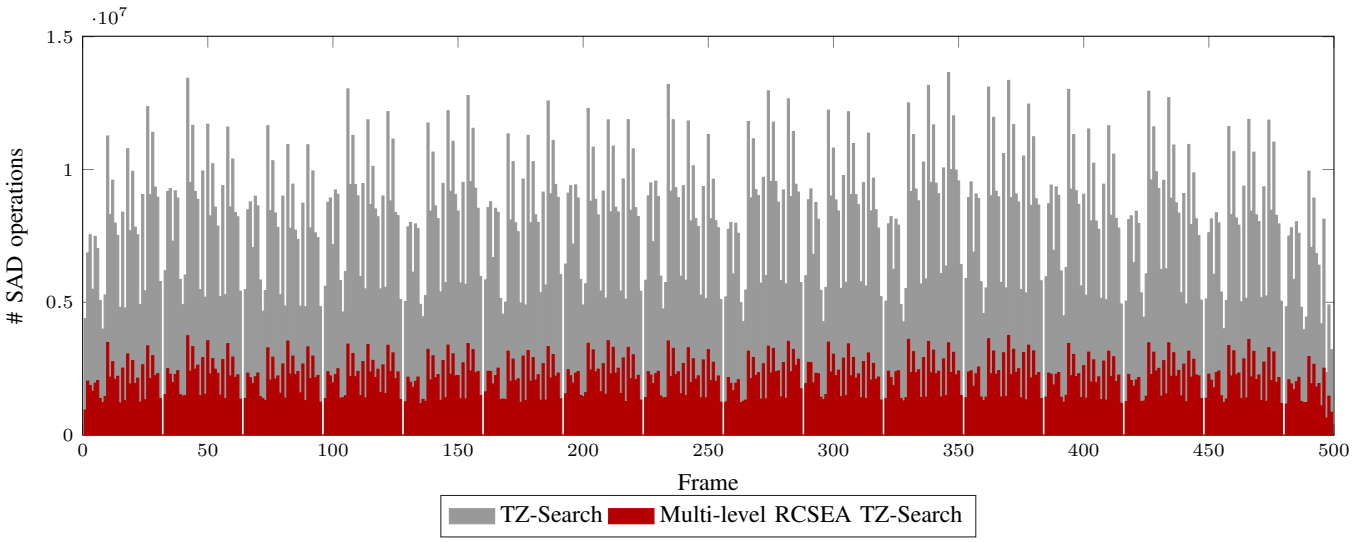


Fig. 9: Comparison of the number of SAD operations performed by TZ Search with and without ML-RCSEA for the BasketballDrill sequence, QP=22 and using the LD B Main profile.

TABLE III: Percentage of SAD operations saved by ML-RCSEA in the TZ Search (LD B Main profile)

	QP			
Sequence name	22	27	32	37
BasketballDrill	70.94%	73.32%	76.32%	79.36%
BQMall	69.89%	72.39%	75.24%	77.93%
PartyScene	59.56%	60.26%	62.06%	64.46%
RaceHorsesC	71.94%	73.63%	75.45%	77.21%
Average	68.08%	69.90%	72.27%	74.74%

For each sequence in Table III, the percentage of SAD operation savings increases with the QP. This is due to the fact that the weighting coefficient (λ in eq. (20)) is dependent on the QP, as it reflects the desired trade-off between rate and distortion. As the QP increases, the distance from the current best candidate plays an increasingly important role in transitive elimination.

B. Double-Check Mechanism for RCSEA in TZ Search

As explained in Section IV-C, the MV cost must be taken into consideration when performing RCSEA. However, it will not often alter the outcome of transitive elimination. In Section IV-C, we proposed the double-check mechanism for RCSEA, which only computes the MV cost when there is an impact on transitive elimination.

For quantitative results specific to the TZ Search algorithm, we measured the percentage of cases where the double-check mechanism saved MV cost computations, and our findings are presented in Table IV. On average, in common test conditions

(QPs: 22, 27, 32 and 37), the double-check mechanism avoided 58%, 53%, 47% and 40% MV cost computations, respectively. These MV cost computations impacted neither the transitive elimination nor the encoding process.

TABLE IV: Motion vector cost computational savings of the double-check mechanism implemented in TZ Search with ML-RCSEA (LD B Main profile)

	QP			
Sequence name	22	27	32	37
BasketballDrill	59.31%	53.91%	47.65%	41.34%
BQMall	59.76%	55.02%	49.08%	42.73%
PartyScene	52.23%	46.95%	40.03%	32.47%
RaceHorsesC	62.20%	57.49%	51.54%	45.54%
Average	58.37%	53.34%	47.07%	40.52%

It easily follows that the MV cost computational savings decrease as the QP increases. This is due to the relationship between the weight coefficient (λ in eq. (20)) and the QP. Increasing the weight of the MV cost in relation to the ADS makes the weighted MV cost a discriminant factor in the rate-constrained transitive elimination (eq. (22)). Concretely, as the QP increases, the MV cost needs to be computed more frequently. However, even when the QP is high, the double-check mechanism still avoids a non-negligible number of useless operations.

The impact on ETRR is negligible as MV cost computations represent less than 1% of the overall encoding time. However, in a context where the MV costs are approximated (i.e., HEVC), and since RCSEA requires the computation of less than half of the MV costs, this could allow the use of a more precise, yet more complex approximation of MV costs contributing to increasing the rate-distortion performance.

C. Cost-Based Search Ordering For Bi-Predictive Search

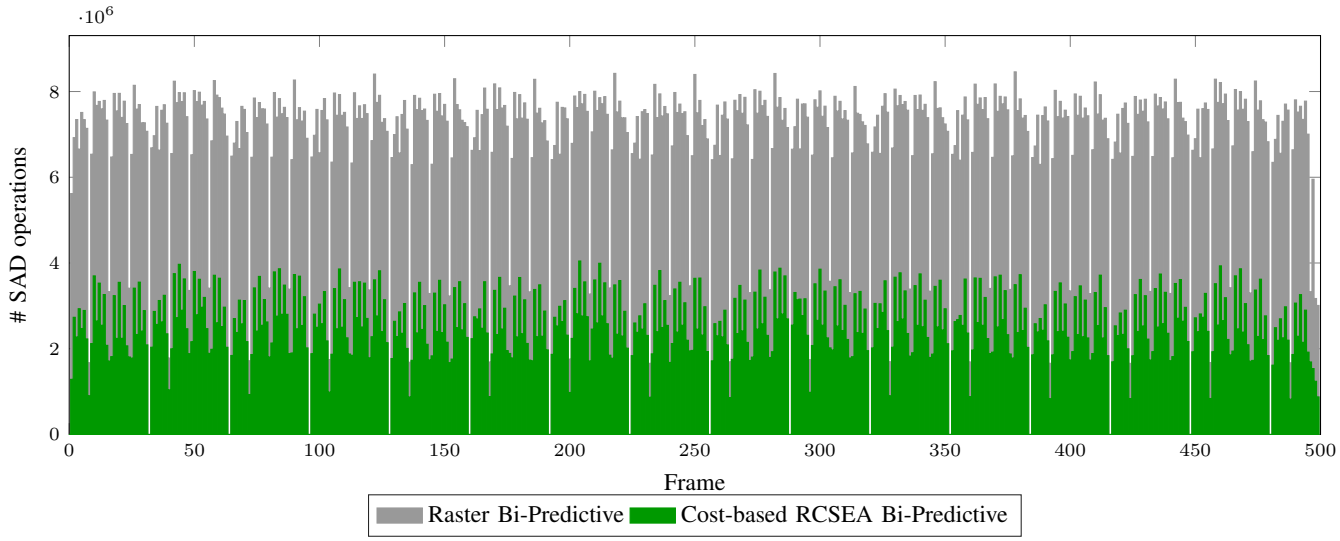


Fig. 10: Comparison of the number of SAD operations performed by bi-predictive raster refinement, as compared to a cost-based RCSEA refinement for the BasketballDrill sequence, QP=22 and using the LD B Main profile

As shown in Fig. 8, after the ML-RCSEA is applied to TZ Search, the bi-predictive raster search becomes the predominant source of SAD operations in the HM reference encoder. As explained in Section V, we propose to use a cost-based search

ordering to maximize the efficiency of transitive elimination in the bi-predictive refinement search area. Table V shows the percentage of SAD operations saved by the proposed approach compared to an unmodified HM. Fig. 10 illustrates the frame by frame savings for *BasketballDrill* at QP of 22.

TABLE V: Percentage of SAD operations saved by RCSEA with cost-based search ordering in bi-predictive refinement (LD B Main profile)

	QP			
Sequence name	22	27	32	37
BasketballDrill	54.85%	56.12%	61.02%	67.96%
BQMall	59.96%	63.94%	68.18%	71.89%
PartyScene	50.83%	50.87%	51.80%	53.46%
RaceHorsesC	42.40%	45.46%	50.07%	55.62%
Average	52.01%	54.10%	57.77%	62.24%

As in sections VI-A and VI-B, the SAD operation savings increase as the QP increases. Once again, this is caused by the weighting coefficient λ. Even if the refinement search area is small (9×9), the MV cost remains significant because the predicted MV is not always the center of the search area; while the area itself is small, that might not be the case for the MV costs. Moreover, as explained in [28], the cost-based ordering maximizes the impact of the rate constraint on transitive elimination.

D. Minimum SAD Savings Threshold

In Section IV-B, we argued that the SAD operation savings of ML-RCSEA outweighed the pre-computational costs required for transitive elimination. We showed that for a given block size, computing the sum buffers required $\frac{4}{3}$ SAD operations per block size to encode the frame. The ETRR gains in tables VII and X confirm this. Nonetheless, in Table VI, we show that the minimum SAD savings threshold is easily exceeded by the proposed solution.

TABLE VI: Average SAD operations saved per frame for the proposed solution for both TZ Search and bi-predictive search using the LD B Main profile. The required threshold is approximately $55\,000$

	QP			
Sequence	22	27	32	37
BasketballDrill	12 486 437	12 048 817	11 562 327	11 024 823
BQMall	11 321 875	10 893 245	10 442 808	9 990 449
PartyScene	9 268 459	8 776 640	8 210 266	7 675 516
RaceHorsesC	18 453 584	17 729 079	16 448 261	14 671 255
Average	12 882 589	12 361 945	11 665 916	10 840 511

We may recall from Section IV-B that the minimum SAD savings threshold for a class-C sequence is approximately 55000 SAD operations. The SAD savings are nearly 200 times larger than the minimum SAD threshold.

E. Comparison with State-of-the-Art

Starting with version 16 of the HM reference encoder, a significant bug was fixed in the TZ Search algorithm [30]. This fix considerably reduces the number of SAD operations considered by the TZ Search algorithm and lowers its execution time. To

allow a comparison with [4], we implemented the proposed solution in the same version of the HM reference encoder as the authors' (i.e., version 12.1).

Table VII compares the ETRR, IMETRR and BD-Rate of the CIME method [4] with the proposed solution (i.e., ML-RCSEA for TZ Search, double-check mechanism for RCSEA in TZ Search, and cost-based search ordering for ML-RCSEA for bi-predictive search). We introduce the bi-predictive refinement time reduction ratio (BRTRR). The BRTRR is taken into account in the ETRR, but not in the IMETRR. The simulations were performed for the LD B Main profile.

TABLE VII: Comparison of the proposed solution against CIME [4] with HM-12.1 for LD B Main profile. Class B is 1920×1080. Class C is 832×480

		ETRR		IMETRR		BRTRR	BD-Rate (Y)	
Class	Sequence name	CIME	**Proposed**	CIME	**Proposed**	**Proposed**	CIME	**Proposed**
B	BasketballDrive	16.62%	18.70%	73.47%	64.72%	12.96%	1.36%	0.0207%
	Kimono	16.13%	23.87%	73.01%	69.02%	48.73%	1.11%	-0.0930%
	ParkScene	11.05%	11.43%	64.83%	54.37%	35.30%	0.60%	0.0088%
	Cactus	10.18%	16.79%	62.54%	66.32%	26.11%	0.82%	0.0090%
	BQTerrace	10.65%	8.06%	70.22%	50.71%	2.61%	0.31%	0.0288%
C	BasketballDrill	11.79%	16.36%	69.86%	62.19%	41.65%	1.63%	-0.0138%
	BQMall	11.38%	14.46%	69.03%	62.57%	49.02%	0.64%	0.0219%
	PartyScene	9.21%	8.28%	65.10%	49.93%	34.15%	0.72%	-0.0555%
	RaceHorsesC	20.46%	23.42%	79.78%	65.81%	37.77%	1.58%	-0.1181%
	Average	13.05%	15.71%	69.76%	60.63%	32.03%	0.97%	-0.0212%

Note in Table VII that the sequences in which the CIME method produces the highest BD-Rate penalties are ones characterized by the greatest speedups. For these sequences, the proposed solution offers both a good speedup and BD-Rate savings. For example, the proposed solution achieves almost an 8% encoding time speedup and a 1.1% BD-Rate improvement over CIME for the Kimono sequence. For BasketballDrill, the speedup is approximately 5%, and the BD-Rate is improved by 1.6%. Overall, the proposed solution reduces the encoding time by an extra 3% without the 1% BD-Rate penalty.

TABLE VIII: Comparison of the proposed solution with HM-16.8 against [4], [16] and [15] with HM-16.0 for RA Main profile as presented in [15] Table IV. Class B is 1920×1080. Class C is 832×480

		CIME [4]		Nalluri et al. [16]		Fan et al. [15]		**Proposed**	
Class	Sequence name	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR
B	BasketballDrive	2.83%	23.75%	2.50%	23.62%	2.15%	26.66%	0.018%	8.70%
	Kimono	2.13%	25.31%	1.91%	26.49%	1.51%	27.11%	-0.014%	7.08%
	ParkScene	1.36%	17.95%	1.07%	18.39%	0.74%	19.18%	-0.012%	5.44%
	Cactus	1.14%	18.13%	0.87%	19.65%	0.59%	21.28%	-0.005%	8.42%
	BQTerrace	0.74%	17.31%	0.45%	19.85%	0.29%	20.04%	0.032%	4.34%
C	BasketballDrill	1.84%	20.75%	1.65%	21.40%	1.46%	22.32%	0.005%	8.10%
	BQMall	1.18%	14.99%	0.85%	16.92%	0.84%	17.50%	-0.008%	8.86%
	PartyScene	0.65%	9.45%	0.79%	9.63%	0.38%	11.12%	-0.010%	4.30%
	RaceHorsesC	3.28%	13.21%	3.04%	15.35%	2.70%	15.64%	-0.190%	9.03%
	Average	1.68%	17.87%	1.46 %	19.03%	1.18 %	20.09%	-0.021%	7.14%

Tables VIII and IX compare the results against state-of-the-art methods using HM $16.x$ reference encoders. In [15], the authors compared their method against those of [16] and [4] using HM version 16.0 for the LD P Main and RA Main profiles.

TABLE IX: Comparison of the proposed solution with HM-16.8 against [31], [4], and [17] with HM-16.0 for RA Main profile as presented in [17] Table VII. Class B is 1920×1080. Class C is 832×480

		Tourapis et al. [31]		CIME [4]		Jia et al. [17]		**Proposed**	
Class	Sequence name	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR	BD-Rate (Y)	ETRR
B	BasketballDrive	2.2%	27%	2.8%	24%	1.3%	33%	0.0%	9%
	Kimono	1.5%	27%	2.2%	26%	1.0%	31%	0.0%	7%
	ParkScene	0.7%	19%	1.4%	18%	-0.2%	22%	0.0%	5%
	Cactus	0.6%	21%	1.1%	19%	0.1%	25%	0.0%	8%
	BQTerrace	0.3%	20%	0.8%	17%	-0.5%	25%	0.0%	4%
C	BasketballDrill	1.5%	22%	1.8%	21%	0.3%	26%	0.0%	8%
	BQMall	0.9%	18%	1.2%	15%	0.0%	24%	0.0%	9%
	PartyScene	0.4%	11%	0.7%	10%	-0.1%	16%	0.0%	4%
	RaceHorsesC	2.7%	16%	3.2%	13%	0.9%	19%	-0.2%	9%
	Average	1.2%	20%	1.7%	18%	0.3%	25%	0.0%	7%

We directly report their results for the RA Main profile in Table VIII. In [17], the authors compared their method against those of [31] and [4] using HM version 16.0 for the LD P Main and RA Main profiles. We report their results directly for the RA Main profile in Table IX. Since our proposed method is fully efficient with profiles using B frames, we did not carry out comparisons using the LD P Main profile. We added our method's results simulated using the more recent HM 16.8 version. Indeed, we could not simulate prior art using version 16.8 since we did not have access to those implementations. Furthermore, implementing our method in 16.0 would not have helped since we are not using the same hardware configurations as [15] or [17]. Although the time reductions in Tables VIII and IX only permit an approximate comparison, we believe that the BD-Rates are directly comparable. We can see that although prior art methods provide appealing time reductions, they often come with a hefty BD-Rate penalty. The two best methods are [15] and [17]. However, [15] achieves its speed-up at the cost of an average BD-Rate penalty above 1%. Worse still, its BD-Rate reaches 2.15% for BasketballDrive (class B) and 2.7% for RaceHorses (class C). The method of [17] provides the most impressive speed-up with the smallest average BD-Rate penalty at 0.3%. However, for some sequences, the BD-Rate penalty is significant (1.3% for BasketballDrive, 1% for Kimono, 0.9 % for RaceHorsesC). In contrast, our method does not alter the BD-Rate of the motion search method on which it is applied (TZ Search in this paper), while providing significant time reductions. For several video companies, for which the size of the encoded videos has a significant impact on operating costs (e.g., related to storage and transport), reducing the size of the videos by 1% (or higher) is more desirable than saving on the encoding complexity, unless the computational savings are very significant (e.g., several times faster, and not only 10 to 30% faster). Based on Tables VIII and IX, our method would be much more appealing to such companies.

It is important to note that the goal of this paper in not to show that applying our method on the dated TZ Search will bring the most competitive results. Rather, the goal is to show that the proposed approach can significantly increase the speed of the ME method on which it is applied without affecting its BD-Rate. For that matter, it would be interesting to apply our approach to the method of [17], configured for higher complexity and thus lower BD-Rates to see if we can obtain a method having the same speed-up as [17] but with lower BD-Rate, especially for the above-mentioned problematic video sequences.

TABLE X: Savings for the proposed solution when compared to HM-16.8 for LD B Main and RA Main profiles. Class B is 1920×1080. Class C is 832×480

		LD B Main				**RA Main**			
Class	Sequence name	**ETRR**	IMETRR	BRTRR	BD-Rate (Y)	**ETRR**	IMETRR	BRTRR	BD-Rate (Y)
B	BasketballDrive	11.73%	57.08%	17.35%	-0.0056%	8.70%	58.68%	12.03%	0.0177%
	Kimono	13.65%	56.58%	45.10%	-0.0896%	7.08%	49.25%	45.55%	-0.0144%
	ParkScene	6.15%	41.73%	35.44%	-0.0032%	5.44%	31.27%	41.23%	-0.0122%
	Cactus	10.61%	59.21%	23.36%	-0.0104%	8.42%	58.71%	31.83%	-0.0045%
	BQTerrace	5.68%	45.49%	9.28%	-0.0522%	4.34%	36.70%	24.83%	0.0317%
C	BasketballDrill	8.55%	48.65%	40.08%	0.0462%	8.10%	46.81%	46.02%	0.0048%
	BQMall	9.49%	49.13%	50.93%	0.0949%	8.86%	43.95%	56.73%	-0.0079%
	PartyScene	3.97%	32.99%	35.39%	-0.0258%	4.30%	28.69%	39.36%	-0.0103%
	RaceHorsesC	11.50%	54.46%	34.18%	0.0051%	9.03%	51.52%	39.46%	-0.1899%
	Average	9.04%	49.48%	32.35%	-0.0045%	7.14%	45.06%	37.45%	-0.0206%

Our method broadly applies to ME methods testing numerous candidates at integer pel precision on various partition sizes, using SAD as the distortion.

F. Detailed Time Savings

Table X presents the ETRR, IMETRR, BRTRR and the BD-Rate of the proposed solution when compared to an unmodified HM 16.8 reference encoder software application, for profiles LD B Main and LD B Main. As previously explained, these results are slightly lower than those presented in Table VII, as fixes [30] were made to the TZ Search algorithm between versions 12.1 and 16.8 of the HM reference encoder software.

Overall, the ETRR was reduced by 9.04% and 7.14% for LD B Main and RA Main, respectively. For LD B Main, this was achieved by an average combined savings of 49.48% for IMETRR and 32.35% for BRTRR. For RA Main, it was achieved by an average combined savings of 45.06% for IMETRR and 37.45% for BRTRR. Note that for both LD B Main and RA Main, the average impact on the BD-Rate is slightly below zero. This is due to the cost-based search ordering used for bi-predictive refinement. When multiple minima exist in the refinement zone, the raster search takes the first one it finds, whereas the proposed approach is rate-biased, and will take the one with the smallest MV cost. For the current block, this could have a slight positive or negative effect on rate distortion because the cost function used during ME is only an approximation of the true rate-distortion ratio of the block. However, over the entire frame, this rate bias is slightly more effective as it can improve MV prediction.

G. Application to Versatile Video Coding Motion Estimation

Although an extensive study of the application of the proposed method to the upcoming VVC standard [20] is beyond the scope of this paper, we provide some insight in that regard in this subsection. There are many novelties in VVC related to ME. First, on a general level, it introduces a quadtree with a nested multi-type tree using a binary and ternary splits segmentation structure, where four segmentation modes are permitted: vertical binary, vertical ternary, horizontal binary and horizontal

ternary [32]. The largest CTU size is set to 128×128. With regard specifically to ME, it introduces numerous new tools, among which the affine motion compensated prediction has the greatest impact on the application of our method to VVC.

VVC introduces a block-based affine transform motion compensation prediction where the affine motion field of the block is described by motion information of two-control point (4-parameter) or three-control point motion vectors (6-parameter). Thus, for each 4×4 luma sub-block, a motion vector is derived at a $1/16$ pel precision. This poses a challenge for most ME algorithms used in HEVC and earlier standards, which perform their task in two phases: an integer pel ME phase on numerous candidates followed by a fractional pel refinement on the best one or a much reduced set of candidates. Even if the control points are at integer pel precision, the affine motion field will lead to fractional pel MVs for 4×4 sub-blocks. Therefore, the ME methods will either continue to operate in two phases by rounding up all MVs in the integer pel phase to quickly eliminate candidates, or their complexity will dramatically increase due to costly interpolation operations. If the former approach is adopted, which is very likely, our method will still apply in the SAD-based integer pel ME phase. The larger CTU size allowed by VVC, as well as the increased number of partition possibilities are factors which make the application of our method to VVC quite promising. In the next subsection, we will discuss the application to IBC, which is also part of VVC.

H. Application to Screen Content Coding

Although the methods presented in this paper were applied to suboptimal ME algorithms, they could be applied to solve other block matching problems found in video compression. For instance, they could be applied to Intra Block Copy (IBC) used for Screen Content Coding (SCC).

A SCC extension to HEVC has been developed by the Joint Collaborative Team on Video Coding [22] to improve the compression efficiency of videos comprising a significant portion of rendered graphics, text, or animation which are present in increasingly popular applications such as online gaming, augmented reality and remote screen sharing. One of SCC's major coding tools is IBC [21]. IBC, also referred as current-picture referencing (CPR). In this coding tool, sample values are predicted from other samples in the same picture using a displacement vector called a block vector; the method being similar conceptually to motion-compensated prediction. The use of IBC is not limited to HEVC. For instance, it is used in AV1 [33], [34], the emerging open-source and royalty-free video compression format, and is considered for inclusion into the upcoming VVC standard. It is noted in [35] that long-range repeated patterns and large motions can often be observed in screen contents. The proposed approach is especially efficient at supporting efficiently large range block search. In the literature, the problem is solved by introducing hash-based block matching [35], [36], [37]. Future work should be carried out to investigate the speedups that multi-level RCSEA can bring in the context of IBC in SCC.

VII. Conclusion

In this paper, we demonstrated that the SAD operation savings brought about by ML-RCSEA outweigh the pre-computational costs of this approach for the TZ Search algorithm found in the reference encoder. We proposed a multi-level composition pattern to perform RCSEA on an asymmetric partitioning. The advantage of using this pattern is that it does not require the computation of additional sum buffers or the need to manage complexities related to offsets and overlaps. When implemented

in the TZ Search algorithm, this approach results in an average total reduction of 71% in SAD operations. These savings alone represent approximately 200 times the pre-computational cost of the approach. Furthermore, we introduced a double-check mechanism for RCSEA, in a bid to avoid computing useless MV costs. This new concept avoids computing more than half of MV costs. Furthermore, we adapted the cost-based search ordering to the bi-predictive refinement search area. We also showed that this new approach decreases the number of SAD operations by approximately 56% on average.

Based on the ETRR and IMETRR achieved by ML-RCSEA, even if the TZ Search algorithm performs a considerable amount of SAD computation as compared to other suboptimal approaches, we argue that the SAD savings threshold is low enough to warrant further research on combining ML-RCSEA with other suboptimal algorithms without any BD-Rate impact.

Finally, although the state-of-the-art methods provide larger time reductions in encoding time compared our method, they usually come with a hefty 1% BD-Rate penalty which makes them unattractive in practice. In contrast, our method provides time savings without altering the BD-Rate performance of the motion estimation method on which it is applied. Promising future research directions consist in applying the proposed concepts to more recent fast ME methods, as well as VVC and SCC.

REFERENCES

[1] *Advanced video coding for generic audiovisual services*. Rec. ITU-T H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Apr. 2017.
[2] *High Efficiency Video Coding*. Rec. ITU-T H.265 and ISO/IEC 23008-2, Feb. 2018.
[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1649–1668, Dec 2012.
[4] Nan Hu and En-Hui Yang, "Fast Motion Estimation Based on Confidence Interval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, pp. 1310–1322, aug 2014.
[5] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Transactions on Image Processing*, vol. 4, pp. 105–7, Jan. 1995.
[6] M. Coban and R. Mersereau, "A fast exhaustive search algorithm for rate-constrained motion estimation," *IEEE Transactions on Image Processing*, vol. 7, pp. 769–773, May 1998.
[7] 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, pp. 501–504, Mar 2000.
[8] 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, pp. 213–221, Feb. 2005.
[9] A. Tourapis, O. Au, and M. 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.
[10] A. 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.
[11] H. Cheong and A. M. Tourapis, "Fast motion estimation within the H.264 codec," *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, vol. 3, pp. III–517–20 vol.3, July 2003.
[12] Ken McCann, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description," Tech. Rep. October, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Strasbourg, France, 2014.
[13] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *ITU-T Q.6/16, VCEG-M33*, 2001.
[14] L. Gao, S. Dong, W. Wang, R. Wang, and W. Gao, "A novel integer-pixel motion estimation algorithm based on quadratic prediction," in *2015 IEEE International Conference on Image Processing (ICIP 2015)*, pp. 2810–2814, IEEE, sep 2015.
[15] R. Fan, Y. Zhang, and B. Li, "Motion classification-based fast motion estimation for high-efficiency video coding," *IEEE Transactions on Multimedia*, vol. 19, pp. 893–907, May 2017.
[16] P. Nalluri, L. N. Alves, and A. Navarro, "Complexity reduction methods for fast motion estimation in HEVC," *Signal Processing: Image Communication*, vol. 39, pp. 280–292, 2015.
[17] L. Jia, C. Tsui, O. C. Au, and K. Jia, "A new rate-complexity-distortion model for fast motion estimation algorithm in HEVC," *IEEE Transactions on Multimedia*, vol. 21, pp. 835–850, April 2019.
[18] L. Trudeau, S. Coulombe, and C. Desrosiers, "Rate distortion-based motion estimation search ordering for rate-constrained successive elimination algorithms," in *2014 IEEE International Conference on Image Processing (ICIP 2014)*, (Paris, France), pp. 3175–3179, Oct. 2014.
[19] L. Trudeau, S. Coulombe, and C. Desrosiers, "An adaptive search ordering for rate-constrained successive elimination algorithms," in *2015 IEEE International Conference on Image Processing (ICIP 2015)*, (Québec, Canada), pp. 207–211, Sept. 2015.
[20] B. Bross, J. Chen, and S. Liu, "Versatile video coding (draft 5)," in *14th JVET Meeting, Geneva, Switzerland*, Mar. 2019.
[21] X. Xu, S. Liu, T. Chuang, Y. Huang, S. Lei, K. Rapaka, C. Pang, V. Seregin, Y. Wang, and M. Karczewicz, "Intra block copy in HEVC screen content coding extensions," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 409–419, Dec 2016.
[22] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, pp. 50–62, Jan 2016.
[23] Y. Wang, Y.-Q. Zhang, and J. Ostermann, *Video Processing and Communications*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2001.
[24] D. Hearn and M. Baker, *Computer Graphics with OpenGL*. Pearson Custom Computer Science Series, Pearson Prentice Hall, 2004.
[25] K.-P. Lim, G. Sullivan, and T. Wiegand, "Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods," tech. rep., 2006.
[26] S. Jeon, S. Park, and K. Ryoo, "A fast inter prediction method in HEVC using SAD computing algorithm based on bottom-up design," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 10, pp. 105–118, 07 2017.
[27] J. Cai and W. D. Pan, "Fast exhaustive-search motion estimation based on accelerated multilevel successive elimination algorithm with multiple passes," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, no. 1, pp. 1190–1193, 2010.
[28] L. Trudeau, S. Coulombe, and C. Desrosiers, "Cost-based search ordering for rate-constrained motion estimation applied to HEVC," *IEEE Transactions on Broadcasting*, vol. 64, pp. 922–932, Dec 2018.

[29] F. Bossen, "Common test conditions and software reference configurations Output," *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG1*, vol. JCTVC-L110, no. 12th Meeting: Geneva, pp. 1–10, 2013.
[30] A. M. Tourapis, Y. Su, A. Ramasubramonian, C. Fogg, A. Duenas, and F. Bossen, "HM reference software bug fixes and enhancements to address the HDR/WCG CfE," *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11*, no. JCTVC-U0040, pp. 1–4, 2016.
[31] A. M. Tourapis, O. C. Au, and M. L. Liou, "New results on zonal based motion estimation algorithms-advanced predictive diamond zonal search," *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 5, pp. 183–186 vol. 5, 2001.
[32] J. Chen, Y. Ye, and S. H. Kim, "Algorithm description for versatile video coding and test model 5 (VTM 5)," in *14th JVET Meeting, Geneva, Switzerland*, Mar. 2019.
[33] W. D. Pankaj Topiwala, Madhu Krishnan, "Performance comparison of VVC, AV1, and HEVC on 8-bit and 10-bit content," vol. 10752, pp. 10752 – 10752 – 10, 2018.
[34] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, *et al.*, "An overview of core coding tools in the AV1 video codec," in *Picture Coding Symposium (PCS)*, pp. 24–27, 2018.
[35] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "Hash-based block matching for screen content coding," *IEEE Transactions on Multimedia*, vol. 17, pp. 935–944, July 2015.
[36] W. Xiao, G. Shi, B. Li, J. Xu, and F. Wu, "Fast hash-based inter-block matching for screen content coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, pp. 1169–1182, May 2018.
[37] J. Li, H. Su, A. Converse, B. Li, R. Zhou, B. Lin, J. Xu, Y. Lu, and R. Xiong, "Intra block copy for screen content in the emerging AV1 video codec," in *2018 Data Compression Conference*, pp. 355–364, March 2018.