



Learning-based bypass zone search algorithm for fast motion estimation

Paulo Goncalves¹ · Guilherme Correa¹  · Luciano Agostini¹ · Marcelo Porto¹

Received: 29 March 2021 / Revised: 23 July 2021 / Accepted: 3 April 2022 /

Published online: 11 July 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Video coding has been widely explored by academia and industry in recent years, mainly due to the great popularization of video applications and multimedia-capable devices. The Motion Estimation (ME) process receives special attention since it is one of the most complex steps in video coding. The Test Zone Search (TZS) is the main algorithm employed for integer ME in recent video codecs, such as those based on the High Efficiency Video Coding (HEVC), and has been used in the standardization process of the future Versatile Video Coding (VVC) standard. However, even though it is designed as a fast ME algorithm, the computational effort required by TZS is still very high, compromising the encoding process in multimedia-capable devices that operate on limited energy or computational resources. This work presents the Bypass Zone Search (BZS) algorithm, a learning-based solution for fast ME that improves TZS, aiming at a better tradeoff between compression efficiency and computational cost. First, a set of analyses on TZS is presented, which allowed the design of two strategies to reduce the ME computational cost. The first one, named as Learning-based Bypass Motion Estimation (LBME), consists of a machine learning-based approach that predicts whether the best motion vector has already been found and bypasses the remaining ME steps. The second strategy, named as Astroid Raster Pattern (ARP), is a novel search pattern developed for the most complex TZS step, the Raster Search. By combining the two proposed strategies in BZS, the ME processing time is reduced by 60.98% (Random Access) and 63.05% (Low Delay) in comparison to TZS. The overall HEVC encoding time is reduced by 14.32% (Random Access) and 17.64% (Low Delay), with a negligible loss of 0.0837% (Random Access) and 0.04% (Low Delay) in BD-rate.

✉ Guilherme Correa
gcorrea@inf.ufpel.edu.br

Paulo Goncalves
phrgoncalves@inf.ufpel.edu.br

Luciano Agostini
agostini@inf.ufpel.edu.br

Marcelo Porto
porto@inf.ufpel.edu.br

¹ Video Technology Research Group (ViTech), Graduate Program in Computing (PPGC), Federal University of Pelotas (UFPEL), Pelotas, Brazil

Keywords Motion estimation · Machine learning · Search pattern · Video coding · High efficiency video coding

1 Introduction

In recent years, both academy and industry have put a large effort on video technology research due to the increasing market interest on efficient multimedia products and applications. According to a Cisco report, by 2017 video traffic in mobile devices had been responsible for 59% of the global mobile data traffic and the prediction is that by 2022 this number will grow to 79% [6]. Video streaming services are only possible thanks to the use of video compression algorithms and the adoption of video coding standards, whose primary aim is to reduce the video bitrate while minimally affecting its visual quality.

The High Efficiency Video Coding (HEVC) is a video coding standard launched by the Joint Collaborative Team on Video Coding (JCT-VC) [31] as an evolution of the previous standard, the Advanced Video Coding (H.264/AVC) [15]. Compared to its predecessor, the HEVC achieves a bitrate reduction of 50%, considering equivalent image quality. However, this high compression efficiency is achieved in HEVC encoders at the cost of a computational cost five times higher than the observed in H.264/AVC encoders [8], mainly due to the introduction of several new tools and data structures in HEVC.

The HEVC encoder first divides each frame into equal-sized blocks, the Coding Tree Units (CTUs), which largest possible size is 64×64 . CTUs can be further divided recursively based on a quadtree structure, creating Coding Units (CUs) with sizes that vary from 64×64 to 8×8 samples. Each CU can be further divided into symmetric and asymmetric rectangular blocks called Prediction Units (PUs), for prediction purposes, and in square Transformation Units (TUs) [34], for transformation and quantization. The best partitioning for a CTU can only be defined based on a full Rate-Distortion Optimization (RDO) process, which evaluates every possibility and compares them in terms of Rate-Distortion (RD) cost. Thus, for each partitioning possibility considered, the full HEVC encoding process needs to be performed for every CU, which includes intra and inter-frame prediction, transformation/quantization, entropy coding and the reconstruction loop. To reduce the number of block candidates, several algorithms have been proposed in the last years. In [22], the authors increase the speed of the encoder by skipping the PU modes that are rarely used. The decision mechanism is trained using a k -means clustering algorithm. Similarly, the authors in [20] propose a CU early-termination classifier based on reinforcement learning to reduce the computational complexity of CU decision.

Motion Estimation (ME) is part of the inter-frame prediction and is the main responsible for most of the compression gains achieved by HEVC. As ME is performed for every PU within all tested CUs, its complexity is thoroughly affected by the number of partitioning possibilities considered in the RDO-based decision process. In the HEVC reference software [14], inter frame-prediction is responsible for 74% of the overall encoding time [1].

To reduce ME complexity, several fast block matching algorithms have been proposed in the last decades, such as [7, 18, 38, 39]. Test Zone Search (TZS) is one of such fast algorithms for IME, which is adopted in current state-of-the-art reference encoders, both for HEVC and the future Versatile Video Coding (VVC) standard [5]. TZS provides good compression efficiency by applying a four-step strategy that aims at predicting the region close to the best block matching. These four steps are (i) Motion Vector Prediction, which defines the start point for the next step; (ii) Initial Search, where an expansion is done in a diamond or square pattern; (iii) Raster Search, where a sub-sampled Full Search (FS) is

applied over the previous step best result and (iv) Refinement, where a new diamond or square search is done around the best result of the previous step.

TZS employs a set of heuristics to search for the best block matching in reference frames, taking advantage of the spatial and temporal correlation in neighboring areas of a candidate block. However, despite being considered a fast ME approach, TZS is still one of the most time-demanding algorithms in HEVC, since it is still executed several times for all PU formats within every candidate CU in the quadtree [31].

In [3, 9], a set of strategies to reduce TZS complexity have also been proposed, either by modifying its search pattern or by proposing fast mode decisions to speed up encoding time. However, although achieving significant complexity reduction, these strategies incur in non-negligible coding efficiency losses. More recently, the authors proposed in [10–12] two different strategies based on an octogonal search pattern and on decision tree-based early-termination models built with the C4.5 algorithm [28]. Such strategies have provided interesting results in terms of complexity reduction with low impact in compression efficiency, but further investigations revealed that a novel search pattern and more accurate and lighter decision trees can be obtained with a different approach, as presented in this paper.

This work presents a two-fold novel contribution, which were joined in an algorithm named *Bypass Zone Search* (BZS):

- *Learning-based Bypass Motion Estimation (LBME)*: a set of decision tree models that predict whether the best block matching has already been found, thus bypassing the remaining IME steps.
- *Astroid Raster Pattern (ARP)*: a novel search pattern for the Raster Search step that exploits regions with the highest probabilities of finding the best block matching.

When implemented in the HEVC reference encoder (version 16.14) [14], BZS is able to reduce the IME processing time by 60.98% (Random Access) and 63.05% (Low Delay) in comparison to TZS, with a negligible compression efficiency loss of 0.0837% (Random Access) and 0.04% (Low Delay).

The rest of this paper is organized as follows: Section 2 discusses TZS and its computational cost. Section 3 presents related works on complexity reduction for motion estimation. Section 4 presents a set of statistical analysis on the TZS behavior, which led to the proposal of BZS, presented in Section 5. Section 6 presents the obtained results and comparisons with related works. Finally, Section 7 concludes this paper.

2 Test zone search algorithm

Test Zone Search (TZS) is an incremental ME algorithm that achieves compression efficiency close to the Full Search (FS) algorithm, even though it compares a smaller number of blocks in the block matching process [14]. TZS performs an adaptive search for the best Motion Vector (MV), aiming to avoid local minima results. The search is executed according to the four steps in the flowchart of Fig. 1: (i) Motion Vector Prediction, (ii) Initial Search, (iii) Raster Search and (iv) Refinement.

The first TZS step is the Motion Vector Prediction, as presented in Fig. 1, also simply referred to as Prediction. Firstly, the co-located position (0,0) is chosen as initial search region. Then, four MV predictors are considered to approximate the initial search region: the MV from the left (*A*), upper (*B*), and upper-right (*C*) neighboring blocks, and the *Median*

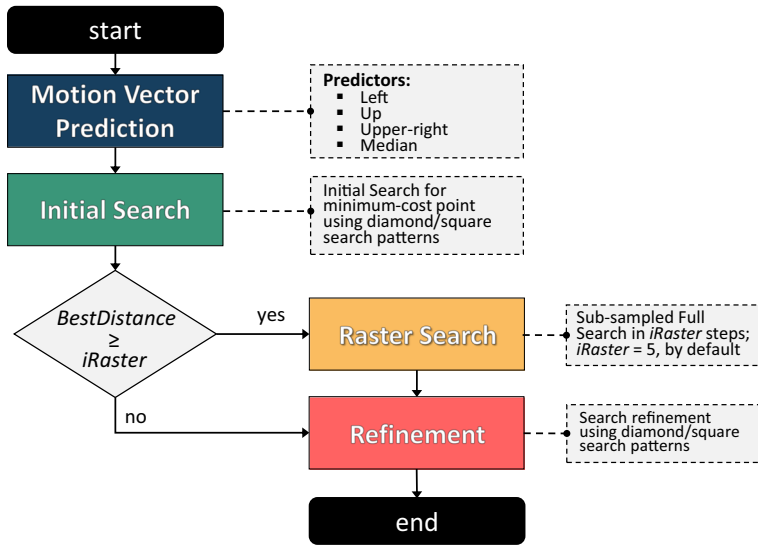


Fig. 1 Flowchart of the Test Zone Search (TZS) algorithm

predictor computed according to (1). As these MVs from neighboring blocks are already available, the computational cost associated to this step is very small.

$$Median = A + B + C - Min(A, Min(B, C)) - Max(A, Max(B, C)) \quad (1)$$

To determine the best predictor, the Prediction step chooses the one that leads to the best results according to a similarity criterion, usually the Sum of Absolute Difference (SAD). Equation (2) defines the SAD, where *Cur* represents the current block, *Ref* denotes the reference block, with block size of $M \times N$ pixels and (x, y) and (s, r) denote MV coordinates of reference and current blocks, respectively. The predictor with the smallest SAD value is selected as the starting point for the next algorithm step.

$$SAD = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |Cur(s + i, r + j) - Ref(x + i, y + j)| \quad (2)$$

The second step is the Initial Search, which is an expansion search applied to the Search Area (SA). The SA is a region in the reference frame, around the best point resulting from the first step, where the best block will be searched. The SA is defined by the Search Range (SR) parameter, which indicates the number of samples around the central point where the SA will be limited. The expansion done in the Initial Search can be executed in a diamond or square pattern. During the Initial Search, the diamond/square size is expanded in powers of two until the stop criterion or the predefined SA limit is reached. If after three expansions no point achieves a better result (i.e., lower SAD) than the central point of the previous expansion, the Initial Search execution is interrupted. The distance between the best candidate block obtained in Initial Search and the best block obtained in Prediction (named as *BestDistance*) is stored for use in the next steps.

The third step of TZS, called Raster Search, is a sub-sampled FS performed within the SA. *iRaster* is a constant defined in the HEVC reference software (HM-16.14 [14]), set to 5 by default, used to determine the sub-sampling stride performed both horizontally and

vertically. Due to this characteristic, the Raster Search is the most complex step of the TZS algorithm since there is no stop condition in its execution. However, Raster Search is only performed when the difference between the best MVs obtained from Initial Search and Prediction (*BestDistance*) is too large. If *BestDistance* is smaller than *iRaster*, the Raster Search step is skipped. The total of n candidate blocks compared in the Raster Search step is defined in (3).

$$n = \left\lceil \frac{1+(SR \times 2)}{iRaster} \right\rceil^2 \quad (3)$$

Refinement is the last TZS step, which provides a greater accuracy for the MV obtained from previous steps. A new search identical to the one performed in Initial Search occurs around the best candidate block found in the previous step. If after two expansion levels no block with smaller SAD is found, the Refinement step is stopped.

3 Related work

Several authors have proposed in the last decades different strategies and approaches to reduce the ME complexity. Sub-optimal block matching algorithms have emerged as the most efficient and commonly employed approaches, such as the Diamond Search [39], the Hexagon Search [38], the Cross-Diamond-Hexagonal Search [7], and the Efficient Three-Step Search [18] algorithms. Test Zone Search (TZS) [14] is currently considered the state-of-the-art solution for sub-optimal ME, since it presents better compression efficiency than its predecessors and substantially decreases complexity in comparison to the optimal FS algorithm. However, TZS still requires larger computational effort than the previously cited sub-optimal algorithms, especially due to the higher number of comparisons per block.

Following a different approach, authors in [21] reduce the complexity of the TZS algorithm by limiting the Diamond Search direction through information from neighboring blocks. Also, early-termination schemes based on machine learning techniques [10] have been proposed to limit the TZS execution, achieving significant complexity reduction. However, in most cases, limiting the search algorithms results in worse final image quality. Therefore, several authors propose strategies to improve quality with low computational cost. In this sense, the authors in [36] recently proposed a neural network-based strategy to improve the quality of the predicted block through the estimation of residual information. With the goal of improving image quality, the authors in [33] propose a new metric to be used during the encoding process, capable of attributing sensitive information of the human visual system to the error calculation.

Some authors have also proposed different search patterns for the ME step in order to reduce complexity. In [3] and [12] the authors reduce the number of search points by employing either a dynamic SA or a new format that prioritizes regions with higher probability of finding the best match. Other authors propose search patterns refined according to the characteristics of the block being coded, as presented in [9]. Such strategies show that intelligently reducing the number of search points leads to acceptable results in terms of compression efficiency. In [25], the authors reduce the range for search refinement by estimating initial points for the MVs, based on dual-tree complex wavelet transform. Also, some authors propose new algorithms to improve the various tools involved in the ME or to replace the ME module present in the HEVC, as presented in [13] and [24]. Finally, some authors have recently proposed strategies for accelerating the ME process by

dynamically optimizing the Search Range according to neighboring PUs characteristics [17], or by exploring data parallelism through the implementation of the ME module in GPU [23].

Even though there are many important advances on complexity reduction strategies for inter-frame coding in recent years, as summarized in [37], most of these solutions are either not directly applied to TZS or lead to a poor tradeoff between complexity reduction and compression efficiency. This way, this work proposes a hybrid learning-based scheme for the ME process, which is an alternative to the TZS algorithm capable of considerably reducing the IME complexity with negligible loss in coding efficiency.

4 Time and statistical analysis

The main goal of this section is to present an analysis on the TZS algorithm and identify its most time-consuming steps. Besides, the section aims at analyzing which TZS steps and SA regions are the main responsible for finding the best block matching. For this, the next sub-sections present two important analyses that serve as basis for the solution proposed in this work. First, Section 4.1 presents a run-time analysis of the TZS steps and an analysis on the best motion vector occurrences along these steps. Then, Section 4.2 presents an analysis on the distribution of such occurrences in Raster Search.

Eight video sequences provided by the Ultra Video Group [32] were used to perform the evaluations: four Full HD (FHD) sequences (*Bosphorus*, *ShakeNDry*, *HoneyBee*, *Jockey*) and four Ultra HD (UHD) sequences (*Beauty*, *BuildingHall2*, *YachtRide*, *ReadySetGo*). All video sequences employed in these evaluations were not employed in the experimental phase presented later in this work. In Table 1, the main properties of each analyzed video sequence are presented. The Temporal Perceptual Information (TPI) [16] is a typical motion activity metric calculated based on the difference between the luminance samples at the same location in space of successive frames. Thus, more abrupt motion activity between adjacent frames results in higher values of TPI. As recommended in the Common Test Conditions (CTC) [30], each video was encoded with Quantization Parameter (QP) 22, 27, 32, and 37. The Random Access (RA) temporal configuration was used in all encoding runs. Maximum CU size was set to 64×64 and maximum CU partition depth was set to 4. The required modifications for extracting information were implemented in the HEVC reference software (HM-16.14). All experiments have been carried on a workstation with an Ubuntu 14.04.5 OS, running on an Intel Xeon E5-2640v3@2.60GHz processor and with 32GB of DDR3 RAM.

Table 1 Analyzed video sequences properties

Sequence	Frames	FPS	Bit Depth	TPI
Bosphorus	600	120	8	38.2
ShakeNDry	300	120	8	44.4
HoneyBee	600	120	8	45.1
Jockey	600	120	8	52.5
Beauty	300	120	10	56.5
BuildingHall2	300	50	10	58.1
YachtRide	300	120	10	61.1
ReadySetGo	300	120	10	63.4

4.1 Best matching and runtime analysis

As discussed in Section 2, TZS is an adaptive hybrid algorithm composed of four basic steps. The first analysis presented in this sub-section aimed at finding out which step usually provides the best MV. During each TZS execution, the best MV chosen at the end of every step was collected. Then, at the end of the full TZS algorithm, the best MV found was compared to the MVs chosen at the end of each step. Figure 2 shows the obtained results for the full set of analyzed sequences. These are average results considering the four recommended QPs and the SR parameter set to 256.

The chart in Fig. 2 shows that the great majority of the best MVs are found precisely in the first phase of TZS, the Prediction step, with an average of 84.72% of the occurrences. The Initial Search step found 10.49% of the best MVs, whereas Raster Search presented the smallest contribution in best MV findings (only 0.67%). Finally, the Refinement step found the best MV in 4.1% of the cases. Notice that the *YachtRide* sequence is the only one with an unusual time distribution. There is an atypical situation in the specific case, with a lot of movement in the waves that fill most of the frame region. For this reason, the Prediction step is not sufficient to determine an accurate initial vector for most blocks in each frame.

In order to verify whether the complexity employed in each step is correlated with the occurrence rate of best MVs, a second analysis is presented in Fig. 3 for each video sequence in terms of processing time demanded by each TZS step. The average execution time for each step also considers results obtained for the four recommended QP values. Notice in the figure that the Prediction step is the least demanding process in terms of execution time. On average, Prediction is responsible for approximately 5% of the overall TZS execution time. This is due to the low complexity required to access information from previously encoded blocks and perform the prediction according to a fixed number of predictors, which results on a constant processing time.

Likewise, the Refinement step also demands a small computational effort to execute the search. On average, 7% of the overall TZS time is associated to the last step. This is because, as the name suggests, Refinement only refines the best results found during Initial Search or Raster Search. This way, only a few expansions are executed, since the previous steps generally converge to regions close to the best point. Even in cases where previous steps

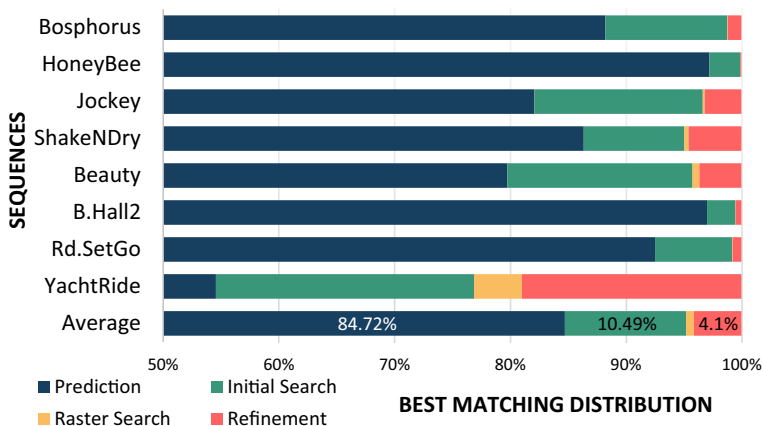


Fig. 2 Best block matching distribution by TZS step

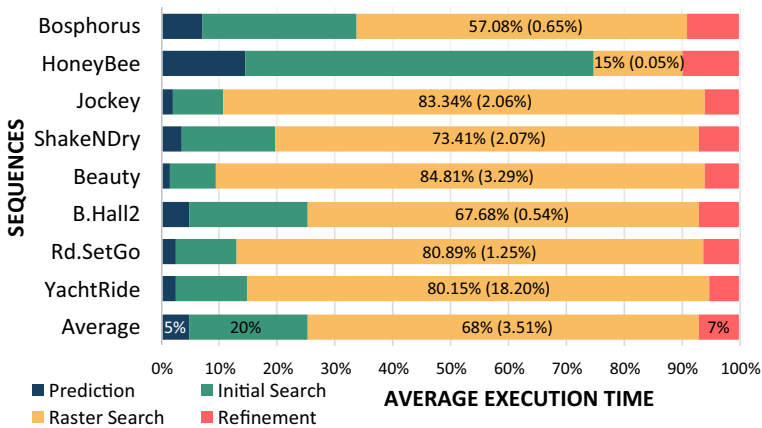


Fig. 3 Normalized running time share per TZS step. The two percentage values for each video indicate the encoding time share of Raster Search and the amount of Raster Search executions

point to regions far from the optimal point, Refinement is terminated within a few iterations, due to its stop condition of two expansions.

The Initial Search step, unlike Refinement, expands the search pattern more often, since only the Prediction step is performed before it. Thus, the Prediction step tries to approximate the best possible region considering the four initial predictors, whereas the Initial Search step is responsible for finding the best point within that region. Due to these characteristics, the Initial Search step is unpredictable as to the number of expansions performed in the search pattern, since it depends heavily on the results of the Prediction step and on the video content. Figure 3 shows that the execution time share of Initial Search varies significantly from one video sequence to another, but it is always more demanding than both Prediction and Refinement steps. On average, Initial Search is responsible for 20% of the overall TZS execution time.

Finally, Raster Search is the most complex step and accounts for 68% of the TZS execution time, on average, even though it is performed only in 3.51% of the TZS executions. As previously explained, this step comprises a search similar to FS, but in a subsampled way. Even though it compares a much smaller number of blocks than FS, the computational effort of Raster Search is much larger than the required by other TZS steps, as Fig. 3 shows. This scenario becomes even more costly in high resolution sequences, in which the Initial Search can direct the search to a region far from the central SA point, forcing the execution of Raster Search in most blocks. This is the case of the *Beauty* sequence, which comprises the face of a woman in a dark scenario with hair in movement, leading the Raster Search execution to areas far from the central point. In this case, Raster Search is responsible for more than 84% of the overall TZS time. In contrast, in *HoneyBee* the execution time of Raster Search was only 15%. This is an atypical case, since the sequence presents a scenario where there is almost no movement. This way, the distance between the best block and the center of the SA is usually small and therefore the Raster Search step is rarely executed.

The analysis presented in this section allows identifying TZS steps in which the employed computational effort is sometimes wasted. It is noted that a great computational effort is employed in steps that rarely yield the best MVs found by TZS, especially in Raster Search and Refinement. One can notice that the vast majority of the best MVs are found in

the first two steps (95.21%), which correspond to only a quarter of the overall TZS complexity (25.42%). On the other hand, the last two steps are responsible for finding out only 4.79% of the best MVs, whereas they require 74.85% of the overall TZS complexity. However, simply skipping these last steps incurs in non-negligible losses in compression efficiency. A set of tests performed over the same video sequences revealed that a BD-rate increase of up to 5.79% is noticed when the Raster Search step only is skipped. Thus, based on this analysis, Section 5.1 presents an intelligent algorithm that allows dynamically bypassing some TZS steps, maintaining compression efficiency and significantly reducing complexity.

4.2 Best matching distribution in raster search

The previous analysis showed that Raster Search is the most time demanding step of TZS, although it is also the step that yields the smallest share of best MVs. However, Raster Search is essential for the proper functioning of the TZS algorithm, since it can recover the direction of the search region to find the best block if previous steps have failed in this process. Therefore, an evaluation of the best MV occurrence was performed specifically for the Raster Search. To perform this, the same setup described in the previous analysis was employed. However, in this case two different SR values were considered in two separate analyses: 64 and 256.

For each Raster Search execution, the best position (i.e., the final MV chosen by the Raster Search step) was mapped to a matrix that represents the SA, where each matrix cell counts the number of best MV occurrences at that point. Thus, at the end of the sequence encoding, each matrix cell represents the number of times that the best candidate block was found in that position within the SA. Considering a Search Range (SR) of 256, the SA dimension ranges from -256 to +256 samples. As the sub-sampling value (i_{Raster}) is equal to 5, 103 points are compared both horizontally and vertically, resulting in a total of 10,609 candidate block comparisons in this step. Therefore, the occurrence matrix created for the analysis with SR 256 has 103×103 positions, representing each candidate block compared in Raster Search. Similarly, when the SR of 64 is used, a total of 676 block comparisons is performed and the occurrence matrix has 26×26 positions.

For better visualization, the obtained results were plotted as heat maps in Fig. 4, where the warmest colors represent points at which most of the MVs were found, and cooler colors represent regions rarely chosen. The occurrence values were normalized, therefore ranging from 0 to 1 in the color scale. For this analysis, two sizes of SR were evaluated. On the left of Fig. 4, the heat maps show the analysis for the SR value of 64, whereas in the right the SR is set to 256. In both cases, the heat maps are shown in two-dimensional and three-dimensional plots. The plots in Fig. 4(a)–(c) and (e)–(g) show the heat map for three video sequences (*HoneyBee*, *Beauty*, and *Bosphorus*), which are those that presented the most extreme behavior among all analyzed sequences. The remaining sequences presented a behavior similar to the average, shown in Fig. 4(d) and (h), which considers the eight analyzed sequences.

HoneyBee shows a less uniform distribution than the other analyzed sequences for both SR 64 and 256. This is perceived by irregular elevations throughout all the SA. This distribution can be better observed in the 3D chart of Fig. 4(e). In *Beauty*, a more prominent horizontal distribution pattern is noticed in Fig. 4(b) and (f). On the other hand, *Bosphorus* shows a high concentration around the vertical axis, as noticed in Fig. 4(c) and (g).

The distribution characteristics are always related to the scene content, especially due to object shape and camera motion. For example, the *Bosphorus* sequence is composed of a boat moving from left to right, which justifies a higher concentration at the right side

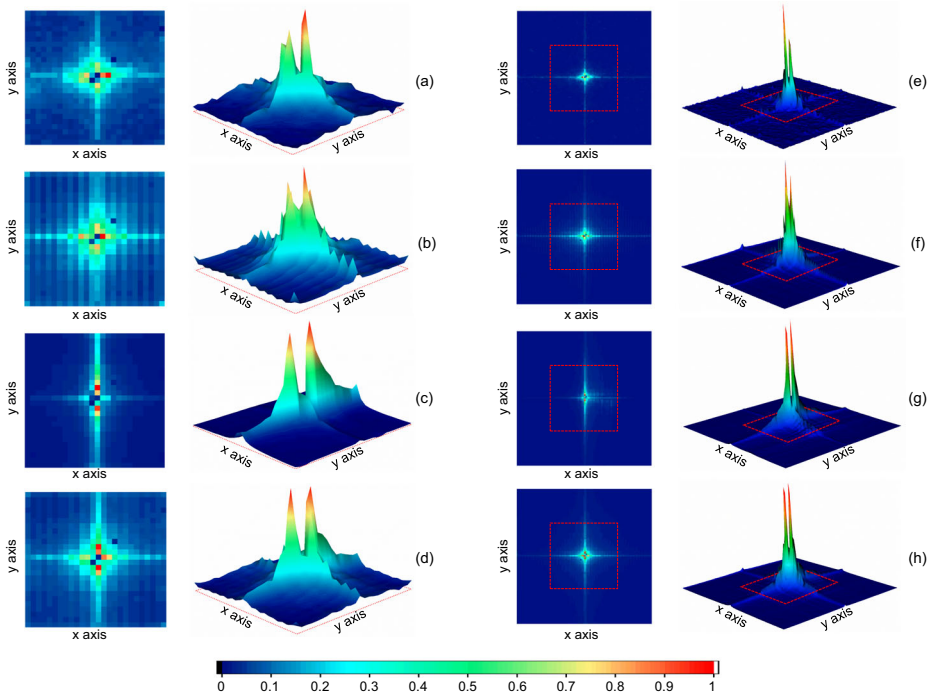


Fig. 4 Distribution of the best MV occurrences in search area $[-256, +256]$: (a) *HoneyBee*, (b) *Beauty*, (c) *Bosphorus* and (d) average distribution. Distribution of the best MV occurrences in search area $[-64, +64]$: (e) *HoneyBee*, (f) *Beauty*, (g) *Bosphorus* and (h) average distribution

of the heat map than at the left, especially for a SR 256, as shown in Fig. 4(g). However, due to the horizontal orientation of videos and also due to the usual horizontal motion of both cameras and objects within a scene, in every case the vast majority of MVs occurrence are at the center and around the central axes of the SA. Other diagonal motion also occurs less frequently and in a more limited range, which justifies the average pattern presented in Fig. 4(d) and (h).

This analysis shows that most of the computational effort of the most complex TZS step is usually wasted by testing candidate blocks located at regions rarely chosen. Thus, it is noticeable that a strategy that prioritizes the most likely regions to find the best MVs would reduce the overall ME complexity, with minor penalties in coding efficiency.

5 Bypass zone search algorithm

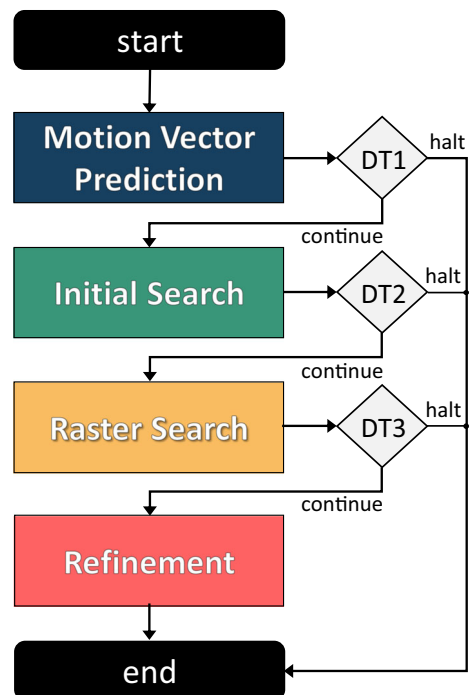
The first analysis presented in the previous section has shown that the TZS algorithm dispenses a significant computational effort in complex steps, even when the best MV has already been found in earlier stages. Also, the second analysis has shown that Raster Search, the most time-consuming step of TZS, searches for candidate blocks in areas rarely chosen in the end. The distribution analysis revealed a very strong concentration of best matching in the central region and around the central axes of the SA.

Thus, this section proposes the Bypass Zone Search (BZS) algorithm, which is a novel hybrid solution for fast Integer Motion Estimation (IME) based on two strategies. Just as TZS, the BZS algorithm is divided into four steps, as shown in Fig. 5: Motion Vector Prediction, Initial Search, Astroid Raster Search and Refinement. However, differently from TZS, BZS employs a three-way Learning-based Bypass Motion Estimation (LBME) strategy that allows bypassing all the remaining steps if the best MV is predicted to be found in the current step. Three decision tree models, trained from an extensive data mining and machine learning process, allow the BZS execution to be halted if the best predicted MV has already been found. The second BZS strategy is a novel search pattern employed in the Astroid Raster Search step, named Astroid Raster Pattern (ARP). Through ARP, Astroid Raster Search is able to prioritize the most likely zones to find the best MV, according to the distribution analysis presented in Section 4.2. The following subsections detail the two strategies.

5.1 Learning-based bypass motion estimation

Section 4.1 showed that most of the best MVs are actually found in the least complex step of TZS, the Motion Vector Prediction. On the other hand, the most complex step is responsible for finding less than 1% of the best MVs. Thus, a three-way bypass strategy, named as Learning-based Bypass Motion Estimation (LBME), is proposed to predict whether the best MV has already been found and thus halt the search. Three decision tree models (DT1, DT2, DT3) were employed in the scheme, each one executed after the Motion Vector Prediction, the Initial Search and the Astroid Raster Search steps, as shown in the flowchart of Fig. 5. If any decision tree decides that the best MV has already been found, BZS is halted and no other later steps is executed; otherwise, the execution of the algorithm continues normally.

Fig. 5 Bypass Zone Search (BZS) algorithm flow. DT1, DT2 and DT3 represent the decision tree models applied after MV Prediction, Initial Search and Astroid Raster Search, respectively



Decision trees are a class of machine learning techniques for general-purpose systems that address very low complexity classification solutions. Basically, after training a model using previously classified items (i.e., cases for which the classification is already known), the model can predict the category of any new target item which category is unknown based on a set of input variables [29]. Nowadays, decision trees are commonly used because they (i) are easily understood and simple to implement, (ii) usually achieve high prediction accuracy after appropriate training, (iii) can deal with categorical and numerical values, (iv) can be built according to several efficient algorithms, (v) require a very low computational effort and (vi) are hardware friendly. The fifth characteristic is extremely important for this work, since it aims at reducing the IME complexity without harming coding efficiency.

A data mining process prior to the decision trees training was performed by collecting 22 features from the TZS algorithm execution in the HM reference software [14]. In order to maintain the computational overhead of the decision process as low as possible, only low-complexity features and values already computed during the encoding process were considered as features. Image descriptors and motion activity indexes were not included because they incur in extra computational cost.

The same video sequences and setup mentioned in Section 4 were considered. Due to the large volume of data, only the first 16 frames of each video sequence were encoded for the data collection. For each of the three models, 1,000,000 instances were gathered and randomly selected to compose a set consisting of 31,250 instances per combination of the eight video sequences and the four QP values, totaling 32 different executions. Thus, all QP values recommended in the CTC and all training sequences are equally represented in the dataset. This process is performed to ensure that the trained model is generic enough to be used in future encodings, with varied types of video content.

The data mining was performed for each of the models separately, which is necessary because not all features are available in all steps. For example, the RD cost feature, calculated in the Initial Search step, is available only after this step. Thus, it cannot be included in the dataset for the first model. On the other hand, this feature can be included in the training dataset of all models following Initial Search.

From the 22 features considered in the data mining, 14 of them achieved a Gain Ratio (GR) larger than 0.01 and were selected for the training step. This GR threshold was defined to limit the number of levels of the trained decision trees, in order to maintain their complexity low. The GR metric [19] was used to select the attributes according to their relevance regarding the final decision (i.e., whether the current MV is the best MV). In other words, only features with a minimally enough contribution to the final decision were maintained to avoid unnecessary attribute computations. GR [19] is a normalized version of the Information Gain (IG) [19] provided by most machine learning toolkits, which represents the difference between the number of bits per data item necessary to convey its class identity before and after classification of the dataset based on an attribute [29]. Due to this suitability, the GR measure is more reliable in video applications since the training instances present high variation due to the plurality of the content.

Table 2 lists the 14 selected attributes and their respective GR. *Resolution*, *QP*, *PU Height* and *PU Width* are parameters obtained before the execution of the IME and correspond to the number of pixels in each slice, the QP used in the current Coding Unit and the vertical and horizontal dimensions of the current Prediction Unit. The remaining attributes are collected after the execution of each step, represented by the PMV, IS and RS prefixes for Prediction, Initial Search and Raster Search, respectively.

Table 2 Selected features and Gain Ratio for each model

Attribute	Gain Ratio		
	1 st Model	2 nd Model	3 rd Model
Resolution	0.0351	-	-
QP	0.0514	0.1081	-
PU Height	0.0162	-	-
PU Width	0.0161	-	-
PMV-Mode	0.0891	-	-
PMV-SAD	0.0337	-	-
PMV-RDcost	0.0327	-	-
PMV-VerMV	0.0211	-	-
PMV-CityBlock	0.0156	-	0.1367
IS-FirstLevel	-	0.5902	-
IS-SAD	-	0.0245	-
IS-RDcost	-	0.0568	-
IS-BestDistance	-	-	0.9589
RS-CityBlock	-	-	0.5282

PMV-Mode corresponds to the mode chosen by the Prediction step to define the center of the SA. *PMV-SAD*, *PMV-RDcost* and *PMV-VerMV* correspond to the SAD value, the RD cost and the vertical component of the MV for the best candidate block found at the end of Prediction step, respectively. *PMV-CityBlock* indicates the City Block distance [2] between the center of the SA and the best candidate block found at the end of Prediction step.

The *IS-FirstLevel* attribute indicates the number of expansions performed by the Initial Search step. *IS-SAD* and *IS-RDcost* are the SAD and RD cost values for the best candidate block found at the end of Initial Search, respectively. *IS-BestDistance* is the same parameter used by the original TZS algorithm to decide whether the Raster Search is performed, and it corresponds to the distance between the best candidate block obtained by the Initial Search step and the best block obtained by the Prediction step. Finally, *ARS-CityBlock* is the City Block distance between the center of the SA and the best candidate block found at the end of Raster Search.

Notice that in Table 2, the first model uses nine features, whereas the second and third models only use four and three features, respectively. This is explainable because, at the Prediction step (where the first model is applied), the available features do not carry much information related to the stop/continue decision. Consequently, each attribute gain ratio is lower, which requires a more significant number of features to be employed in the model. As the motion estimation algorithm evolves to the next steps, the number of necessary features decreases since the model can achieve higher confidence in its decisions with more relevant features (i.e., features with higher GR values).

After data collection and data mining, the training process was performed with the features selected as described. The training algorithm chosen was the C5.0 [27], which is an evolution of the well-known C4.5 algorithm [28] that includes several tools to improve the model building process. Initially, C5.0 calculates the information gain (IG) for each

Table 3 Performance measures for LBME strategy

Model	Performance Measures (%)			
	Accuracy	Precision	Recall	F1_score
1 st Model	78.82	76.36	83.49	79.77
2 nd Model	97.85	96.82	98.95	97.87
3 rd Model	99.52	99.99	99.05	99.52
Average	92.06	91.06	93.83	92.39

attribute, indicating how relevant it is for the classification. Once calculated, the attribute with highest IG is chosen and used to divide the training dataset into two subsets, and then the same process is recursively applied. After training, the model accuracy was observed through a cross-validation process and reported by the C5.0 [27] reference software.

Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observations to the total observations. However, in datasets where the false-positive and false-negative values are very different, the accuracy may present a false hit ratio. In addition to accuracy, other performance measures were calculated to evaluate each model. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. In this sense, high precision is related to a low false-positive rate. Recall is the ratio of correctly predicted positive observations to all observations in actual class. In other words, in our context, it considers all cases with a decision to terminate the ME, whether they are true or not. *F1_score* is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but *F1_score* is usually more useful than accuracy, especially in an uneven class distribution [26]. Table 3 presents the performance metrics for each of the trained models, in addition to the average of the three models. In this sense, considering the average results, it is possible to observe that the LBME strategy is able to reach an accuracy, precision, recall, and F1_score of 92.06%, 91.06%, 93.83%, and 92.39%, respectively.

However, it is important to notice that false-positive decisions correspond to cases in which the models decide to continue the search when the best candidate block has already been found in previous steps. This type of error leads to unnecessary computations but does not incur in compression efficiency loss. On the other hand, false-negative decisions correspond to cases in which the algorithm should continue the search, but the model decides for halting. In this case, the decision error leads to compression efficiency loss, since the best candidate block is not chosen. This way, if false-positive cases are disregarded as decision errors, the first model achieves an actual accuracy of 70.3%. In comparison, the second and third models achieve 99.5% and 99.8%, respectively, as shown in Table 2.

Figure 6 shows the second decision tree model, implemented after the Initial Search step. Once trained offline, the models are implemented in the HM reference software as an IF-ELSE chain, where the *halt* and *continue* leaves indicate the two possible outcomes. The Depth in Fig. 6 indicates the number of decision levels executed by the model. Notice that the models are all composed of a few decision levels (maximum of 11 in the first model), which leads to a negligible computational overhead. Due to the number of decision levels, the other decision trees are not presented in Fig. 6, but can be accessed in [35].

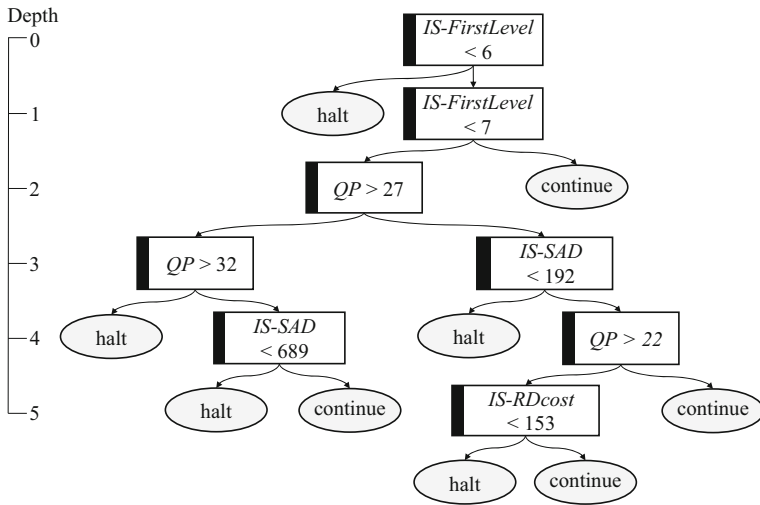


Fig. 6 Proposed decision tree model applied after Initial Search step

5.2 Astroid raster pattern

The MV distribution analysis presented in Section 4.2 revealed that the computational effort required to compare most of the available candidate blocks in the Raster Search step is wasted, since most of the best MVs are found in the central region of the SA and nearby the vertical and horizontal axes, composing an Astroid-shaped area. Thus, the proposed BZS algorithm employs a new search pattern in the Raster Search step, named Astroid Raster Pattern (ARP).

Figure 4(h) showed that the internal square of the SA (dashed in red) corresponds to a region with 25% the size of the original area and contains most of the positions with the highest MV occurrence. Considering the default Search Range (SR) of 256 (i.e., a 512×512 SA), the internal square corresponds to an area of 256×256 pixels. According to (3), this leads to a comparison of 10,609 blocks in each Raster Search execution. On the other hand, if only the inner square is considered (i.e., a SR of 128), a total of 2,704 candidate blocks are tested.

However, Fig. 4 also shows that there are still some sparse cases of medium or high occurrence outside the inner square, especially nearby the horizontal and vertical central axes. Therefore, the idea of ARP is to perform the search in an Astroid-shaped area. Similarly to the Diamond Search [39], the Astroid-shaped comprises the center of the SA, but it also performs the search in both horizontal and vertical axes.

An Astroid, presented in Fig. 7, is a mathematical hypocycloid curve with four cusps traced out by a point *P* on the circumference of one circle *C*₁ (of radius *a*/*4*) as this circle rolls without slipping on the inside of a second fixed circle *C*₂ of radius *a*. As the small circle rolls on the inside of the larger circle, the point *P* draws the hypocycloid curve called Astroid. The parametric equations for the integer (*x*, *y*) coordinates belonging to the Astroid boundary points are given by (4) and (5), where *a* is the radius of the fixed circle *C*₂ and *θ*

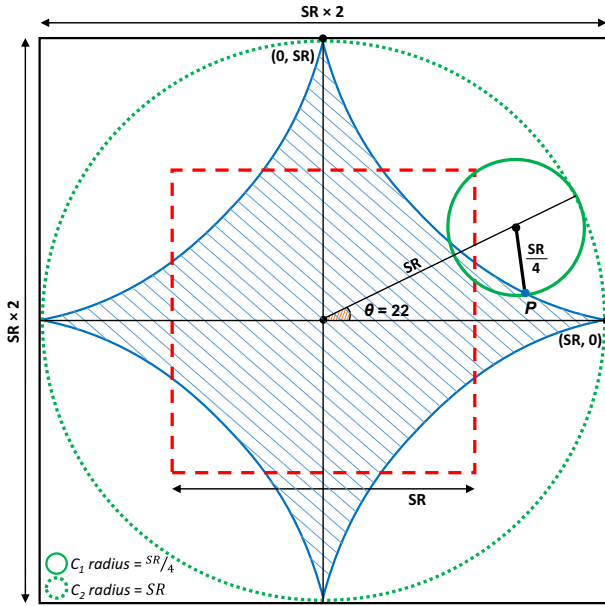


Fig. 7 Proposed Astroid Raster Pattern (ARP)

is the angle between the x-axis and the line joining the origin to the center of the rolling circle C_1 .

$$x = a \cos^3 \theta = \frac{a}{4} (3 \cos \theta + \cos 3\theta) \tag{4}$$

$$y = a \sin^3 \theta = \frac{a}{4} (3 \sin \theta - \sin 3\theta) \tag{5}$$

The SA is defined as the region within the borders of the Astroid, which is limited according to (4) and (5). The point $P(x,y)$ presented in Fig. 7 draws the Astroid boundary while C_1 rolls over C_2 . The figure represents the moment at which θ is 22° , which results in a point at the Astroid boundary defined as $P(x = 204, y = 13)$ for a SR of 256. Once C_1 revolves around the full C_2 circumference, the Astroid limits and the SA are defined.

Notice in Fig. 7 that the Astroid is a format very similar to the pattern observed in the average heat maps presented in Fig. 4(d) and (h). To limit the number of candidate blocks compared by ARP to 25% of the Raster SA, a correction in the extension of the search at the axes is needed. This is also required to allow for a perfect Astroid shape, which states that the larger circle radius is four times the smaller circle radius. Thus, the number of candidate blocks compared in each axis direction (upper, down, left and right) was reduced to the largest multiple of 4 close to the SR value. For example, considering a SR of 256, the original Raster Search compares 51 candidate blocks in each axis direction (see (3)), while ARP compares 48 of them (largest multiple of 4 close to 51).

By applying such changes, the ARP format tests the same number of candidate blocks as the internal square of the original Raster Pattern (i.e., 2,704 blocks). However, as these tested blocks are all located within the limits of the Astroid format, the probability of finding the best MV is much higher. Figure 7 shows the ARP search area in blue and illustrates how the same SA is defined by the original Raster Search (red-dashed square). Notice that even by comparing only 25% of the original area (i.e. 2,704 points), ARP covers most of the region

with highest occurrence of best MVs. In fact, when compared to the average distribution presented in Fig. 4(h), 72.4% of all best MV occurrences are covered by ARP.

Unlike other fast strategies for IME, ARP does not require information from neighboring blocks to direct its search. Also, it is important to notice that ARP is not an algorithm; instead, it is an access pattern that indicates the coordinates of the blocks to be evaluated in the search for the best candidate block. This means that once the block coordinates are within the evaluated search window, ARP will perform the search algorithms at the indicated points without any restriction. In the BZS strategy, ARP is implemented as a simple binary mask that indicates which positions are valid and which positions are not valid according to the astroid format. Using this mask, the ARP search will always present the same behavior. Also, since it always tests a fixed number of candidate blocks, ARP allows for regular execution, which is an important factor for fast ME algorithms, and allows ARP to be employed as an alternative algorithm to FS in solutions that require regularity, as dedicated hardware implementation.

6 Experimental results

This section presents experimental results for the two complexity reduction strategies that compose the BZS algorithm proposed in this work. The results are presented for the strategies implemented separately and combined. Besides, comparisons between TZS and related works are also drawn.

To evaluate the efficiency of the proposed scheme, BZS was implemented in the HEVC Test Model (HM-16.14) reference software [14]. The modified HM encoder was compared to the original HM encoder. Each strategy was evaluated for 29 video sequences, recommended in the CTC document [30], which belong to classes A1, A2, A, B, C, D, and F. The selected sequences differ from one another in frame rate, bit depth, spatial resolution and motion/texture content. It is important to highlight that the sequences used to evaluate the algorithm are different from the sequences used to train the LBME. Except for the video sequences, the same encoding setup described for the analyses of Section 4 was used in these experiments, as recommended in [30], including the Random Access and the Low Delay temporal configurations.

Table 4 presents the obtained results for LBME and ARP separately and jointly implemented in BZS. In all these experiments, the SR value of 256 was selected. Each experiment was evaluated in terms of IME time savings (IME TS), overall encoding time savings (Overall TS) and *Bjontegaard* Delta-rate (BD-rate) [4]. TS is calculated according to (6), where T_O represents the processing time required by the original HM encoder, and T_M represents the processing time required by the HM modified according to the scheme proposed. All obtained results were calculated by taking the HM encoder with the TZS algorithm implemented as anchor for comparison. Due to the highly variable nature of the video encoding process, relating the objective quality and bitrate of a video with complexity reduction is the most direct way to compare distinct techniques in terms of performance. This methodology is described and recommended in CTC [30] and is used by the vast majority of authors that propose strategies for complexity reduction in video coders. The next sections will discuss the results obtained following this methodology.

$$TS = \frac{T_O - T_M}{T_O} \quad (6)$$

Table 4 Obtained results for LBME, ARP, and BZS (TZS as anchor, Random Access configuration)

Class (Resolution)	Sequence	IME TS (%)			Overall TS (%)			BD-rate (%)		
		LBME	ARP	BZS	LBME	ARP	BZS	LBME	ARP	BZS
Class F	<i>BasketballDrillText</i>	20.64	44.57	61.95	5.16	9.96	13.60	+0.0055	-0.2976	-0.0086
	<i>ChinaSpeed</i>	26.90	51.57	68.33	8.53	15.42	19.96	+0.2453	-0.6995	+0.2090
	<i>SlideEditing</i>	51.66	21.45	68.64	6.45	3.23	7.75	+0.0399	-0.1492	+0.2121
	<i>SlideShow</i>	34.74	42.55	68.68	7.88	8.75	13.13	+0.2620	-0.4087	+0.258
Class D	<i>RaceHorses</i>	20.36	32.15	50.96	4.03	6.38	9.93	+0.0871	-0.5747	-0.0270
	<i>BQSquare</i>	23.42	5.05	26.62	1.66	0.31	1.85	-0.0219	-0.0887	-0.0010
	<i>BlowingBubbles</i>	22.95	15.98	38.34	2.23	1.75	3.40	-0.0098	-0.2364	-0.0206
	<i>BasketballPass</i>	22.61	32.22	53.58	4.73	6.72	10.87	+0.0148	-0.3742	-0.0391
Class C	<i>RaceHorses</i>	19.53	48.71	62.75	5.71	14.59	18.40	+0.0427	-0.6940	-0.0467
	<i>BQMall</i>	28.53	38.07	60.33	5.32	7.07	10.82	+0.0425	-0.2108	+0.0554
	<i>PartyScene</i>	21.00	36.06	51.99	3.27	5.46	7.50	+0.0444	-0.2302	+0.0178
	<i>BasketballDrill</i>	18.35	45.03	61.52	4.64	10.70	14.28	+0.0427	-0.3015	-0.0028
Class B (Full HD)	<i>BasketballDrive</i>	25.17	52.23	66.87	8.02	16.37	20.79	-0.0081	-0.2718	-0.0229
	<i>BQTerrace</i>	34.83	26.85	51.24	4.37	3.87	6.57	+0.0397	-0.1481	+0.0782
	<i>Cactus</i>	26.51	47.70	66.26	6.28	11.28	15.33	+0.0063	-0.1720	-0.0443
	<i>Kimono</i>	25.91	46.90	64.20	6.61	11.80	16.05	-0.0125	-0.3076	-0.0594
	<i>ParkScene</i>	26.77	34.46	55.05	3.89	5.26	8.33	+0.0513	-0.2213	+0.0001

Table 4 (continued)

Class (Resolution)	Sequence	IME TS (%)			Overall TS (%)			BD-rate (%)		
		LBME	ARP	BZS	LBME	ARP	BZS	LBME	ARP	BZS
Class A (WQXGA)	<i>PeopleOnStreet</i>	22.54	53.30	67.74	6.75	15.51	19.65	-0.0648	-0.6518	-0.2107
	<i>SteamLocomotiveTrain</i>	27.66	52.49	68.21	8.87	16.25	20.40	+0.0109	-0.3261	+0.1533
	<i>NebutaFestival</i>	27.25	38.41	56.42	4.09	6.22	8.39	+0.0695	-0.0813	-0.1007
	<i>Traffic</i>	36.19	27.67	57.41	4.37	3.34	6.69	+0.0126	-0.2059	+0.0279
Class A1 (Ultra HD)	<i>CampfireParty</i>	21.40	56.42	69.36	7.76	19.37	23.48	+0.0242	-0.1319	+0.0788
	<i>CatRobot</i>	30.26	48.54	67.72	7.97	12.18	16.84	-0.0143	-0.3363	-0.0421
	<i>DaylightRoad</i>	29.29	49.04	66.45	7.80	12.83	17.10	+0.0530	-0.4963	+0.2828
	<i>Drums</i>	29.23	50.10	67.44	7.96	13.13	17.59	+0.0274	-0.5282	-0.0575
Class A2 (Ultra HD)	<i>RollerCoaster</i>	25.99	54.32	69.31	9.59	19.29	24.17	+0.0284	+0.3158	+0.8599
	<i>Tango</i>	24.88	51.23	68.82	9.21	19.48	24.31	+0.0059	-0.4989	+0.0025
	<i>ToddlerFountain</i>	20.82	58.53	70.08	8.82	24.27	28.82	-0.0072	-0.1200	-0.0405
	<i>TrafficFlow</i>	37.00	30.83	62.09	5.68	5.32	9.10	+0.0955	+0.4983	+0.9144
Full HD		27.84	41.63	60.72	5.83	9.72	13.41	+0.0153	-0.2242	-0.0097
WQXGA		28.41	42.97	62.45	6.02	10.33	13.78	+0.0071	-0.3163	-0.0325
Ultra HD		27.36	50.25	67.66	8.10	15.76	20.18	+0.0266	-0.1622	+0.2498
Average		26.98	41.12	60.98	6.13	10.56	14.32	+0.0384	-0.2741	+0.0837

6.1 LBME results

The complete original TZS was used as anchor to evaluate the LBME solution. This means that the ARP was not used in this evaluation and the Raster Search was unchanged.

From Table 4 one can perceive that the IME and overall time savings are consistent for all video sequences. On average, the LBME strategy achieves an IME TS of 26.98% and an overall TS of 6.13%. UHD sequences with low motion activity, such as *TrafficFlow* and *CatRobot*, are those that achieved the highest IME TS, since the LBME strategy can halt the IME in earlier steps. On the other hand, the search in sequences with very complex motion activity, such as *ToddlerFountain*, is less frequently halted by LBME, which needs to run complex steps like Raster Search to find the best MV. In terms of compression efficiency, LBME presents an average BD-rate increase of only 0.0384%. Altogether, the test video sequences have presented BD-rate values close to zero, with both negative and positive variations. This means that the proposed LBME strategy leads to a negligible impact in compression efficiency.

6.2 ARP results

The ARP evaluation considered the TZS with the original Raster Search as anchor and the Raster Search is substituted by the Astroid Raster Search. The LBME solution is not enabled to allow an isolated evaluation of ARP. Table 4 shows that ARP reduces the IME processing time by 41.12%, on average, in comparison to the square Raster Search pattern employed by TZS. When considering the whole encoding process, the use of ARP leads to an overall TS of 10.56%, with an average BD-rate of -0.2741%. Notice that a negative BD-rate means that ARP is able to improve compression efficiency in comparison to the usual original TZS Raster Search pattern. However, unlike LBME, the IME and overall TS results for ARP are not regular throughout all the analyzed sequences. This is due to the content characteristics of each video, which can restrict (or not) the application of the Raster Search or ARP step through the *BestDistance* parameter test (see Fig. 1). For example, the *BQTerrace* sequence is characterized by a scenario with a terrace and a bridge, with few and slow camera movements. Consequently, the MVs found in the initial steps of the TZS are close to the central region of the SA and therefore the Raster Search and the ARP step are rarely executed, since the distance between the best MV after the Initial Search (*BestDistance*) is smaller than *iRaster*. Oppositely, sequences with high motion activity, such as *ToddlerFountain* and *Tango*, benefit more from this search strategy, since the Raster Search and ARP are executed more often and there is higher potential for time reduction.

These experimental results show that both LBME and ARP strategies reduce the IME complexity significantly with negligible or no penalties at all in compression efficiency. Thus, the next subsection presents results for the combined strategies as the BZS algorithm implemented in the HM reference encoder.

6.3 BZS results

Regarding compression efficiency, Table 4 shows that the average BD-rate considering all tested sequences was 0.0837%, which is considered a negligible loss for most applications. The table also shows average results for the three tested video resolutions and reveals that the BZS algorithm leads to negative BD-rate results for FHD and WQXGA resolutions, which indicates improvement in compression efficiency in comparison to TZS. Only UHD sequences presented higher BD-rate results, especially for *RollerCoaster* and *TrafficFlow*

sequences, with BD-rate 0.8566% and 0.9144%, respectively. This is mainly due to the very intense motion in both UHD videos, which leads to many cases where the best candidate block is not within the limits of the original SA. Thus, in such cases the Astroid search is not able to find the best candidate block, since it is located outside the considered region. Seven other video sequences also presented positive BD-rate, but with values much lower than these two mentioned cases.

The best results in terms of coding efficiency are for the *PeopleOnStreet* and *NebutaFestival* sequences, with BD-rate values of -0.2107% and -0.1007%, respectively. Compression efficiency improvements can be explained by the local and sub-optimal decisions of the HM encoder, which does not implement a full RDO algorithm. If the selected suboptimal MV leads to a high RD cost, the encoder could decide to go further in the block partitioning structure, performing the IME again with smaller blocks, which may also improve compression efficiency. By evaluating smaller blocks, the prediction steps could provide better prediction results and lower residues. This leads to better reference frames, conducting to MV decisions in future frames that are better than the performed by TZS.

The observed TS in the overall encoding process was 14.32%, on average. The highest overall TS was achieved for UHD resolution sequences, with an average result of 20.18%. This is explained by the fact that UHD sequences require the execution of Raster Search much more often than lower resolutions, since even a small motion between neighboring frames directs the best block matching to regions far from the center of the original SA (i.e., *BestDistance* is usually larger than *iRaster*, which enables Raster Search). Thus, as Raster Search is required more frequently, there is higher potential for time savings in UHD. Oppositely, sequences with very little motion between neighboring frames, such as *BQTerrace* and *Traffic* presented small overall TS results. In such cases, the generated MVs are in the great majority smaller than *iRaster*, which causes the Raster Search step to be disabled even in the original TZS algorithm.

The time savings for IME only, considering all encoded sequences, was 60.98%, on average. Notice that, in general, those sequences that presented high IME TS results also achieved high overall TS results, as expected. However, it is important to observe the case of the *TrafficFlow* sequence, which also presented the worst results in BD-rate. In this case, even though the obtained IME TS results are high (62.09%), the overall TS for this sequence (9.10%) is about half the average obtained for other UHD sequences (20.18%). This corroborates the hypothesis that if the BZS impact in compression efficiency is too negative, the encoder is required to perform more computations in other steps. For example, if the best block matching is halted at the Prediction step with a sub-optimal MV, the IME complexity is thoroughly reduced, but the encoder will generate more residue to be transformed, quantized and entropy encoded, increasing the computational effort in other encoder modules. Thus, it is essential to observe both IME TS and overall TS to assess the efficiency of the proposed algorithm.

Finally, Table 5 presents average results for the BZS algorithm with the Low Delay temporal configuration. The same encoding setup presented before was used, except for the temporal configuration. Results are averaged by video class group, as defined in the CTC [30]. It is possible to notice that the BZS efficiency does not change in the Low Delay case, even though the all analyses and trained models used data collected from the Random Access scenario. In fact, the Low Delay configuration achieves even greater coding time reductions than the Random Access configuration. In the Low Delay configuration, average IME time savings are 63.05%, while for Random Access it is 60.98%. Also, overall time savings increase to 17.64% (against 14.32% for Random Access). In terms of coding

Table 5 Obtained results for BZS (TZS as anchor, Low Delay configuration)

Class (Resolution)	F	D	C	B	A	A2	A1	Average
IME TS (%)	67.42	44.55	60.47	63.69	66.01	70.10	69.1	63.05
Overall TS (%)	16.84	8.19	15.50	16.90	18.08	22.74	25.24	17.64
BD-rate (%)	+0.24	-0.03	+0.01	+0.05	-0.02	+0.01	+0.06	+0.04

efficiency, the Low Delay scenario also presents better results, with an average decrease of approximately 0.04% in BD-rate.

The lowest overall time savings were obtained for class D in the Low Delay configuration (8.19%). However, a BD-rate of -0.03% was noticed, indicating better coding efficiency than the anchor. The most significant reduction in coding time was for class A1 (25.24%), with a negligible coding efficiency loss of only 0.06%. These experimental results show that BZS offers a great tradeoff between time reduction and coding efficiency regardless of the temporal configuration chosen.

6.4 Comparison with related works

Table 6 shows a comparison between three configurations of this work and three of the main related works found in the literature [9, 13, 24], which are those that excel in terms of complexity reduction and compression efficiency in comparison to the original TZS algorithm. To allow a fair comparison with these works, only the sequences tested in all works are considered for comparison, and the same encoding setup described for the analyses of Section 4 was employed (Random Access configuration). As most related works present only TS results for the overall encoding process, these values are compared in Table 6, besides compression efficiency in terms of BD-rate.

To assess the efficiency of the proposed model, BZS was executed with three different configurations, varying the range of the SR for the IME, and for the ARP search. The SR and the ARP range do not present any direct relationship since the SR is defined for the whole IME, and the ARP range is defined specifically to limit the Raster Search step only. As the SR exists in related works with value 64, the first configuration is the same used in related works, with a SR set to 64 for IME and the ARP range set to 128 (i.e., which leads to the same number of candidates blocks compared by a square Raster Search with SR 64). This configuration reached an overall TS of 19.63%, with a BD-rate increase of 0.68%. In the second configuration, the same SR of 64 was maintained for IME, but the ARP size was increased to 256. In this configuration, the overall TS was reduced to 17.10%, and the BD-rate also reduced to 0.18%. This indicates that increasing the ARP range improves the compression efficiency, while still maintaining a high TS. Finally, in the last experiment the IME SR was set to 256 and the ARP range was set to 256. As expected, this version led to the lowest TS (15.41%), but resulted in negative BD-rate, which means that compression efficiency improved.

It is possible to notice in Table 6 that all related works achieve higher overall TS numbers, surpassing the 15.41% achieved by BZS for these sequences. However, the achieved TS results come at the cost of a large increase in BD-rate, which is above 1% in all related works. Notice that BZS reduces the overall encoding time in at least 15.41% and at 19.63% with an average BD-rate between -0.01% and 0.68%, on average, for these sequences.

Table 6 Comparison with related works (Random Access configuration)

Work	Metric	Kimono	ParkScene	Cactus	BasketballDrive	BQTerrace	Average
BZS (SR=64, ARP=128)	Overall TS (%)	21.30	9.66	20.44	28.61	18.13	19.63
	BD-rate (%)	0.83	0.40	0.34	1.09	0.76	0.68
	BD/TS×100	3.90	4.14	1.66	3.81	4.19	3.48
BZS (SR=64, ARP=256)	Overall TS (%)	18.36	9.02	18.94	21.91	17.27	17.10
	BD-rate (%)	0.12	0.17	0.10	0.08	0.41	0.18
	BD/TS×100	0.65	1.88	0.53	0.37	2.37	1.03
BZS (SR=256, ARP=256)	Overall TS (%)	16.05	8.33	15.33	20.79	16.57	15.41
	BD-rate (%)	-0.06	0.01	-0.04	-0.02	0.07	-0.01
	BD/TS×100	-0.37	0.12	-0.26	-0.10	0.42	-0.05
Hu* [13]	Overall TS (%)	25.31	17.95	18.13	23.75	17.31	20.49
	BD-rate (%)	2.13	1.36	1.14	2.83	0.74	1.64
	BD/TS×100	8.42	7.58	6.29	11.92	4.27	8.00
Nalluri* [24]	Overall TS (%)	26.49	18.39	19.65	23.62	19.85	21.60
	BD-rate (%)	1.91	1.07	0.87	2.50	0.45	1.36
	BD/TS×100	7.21	5.82	4.43	10.58	2.27	6.30
Fan* [9]	Overall TS (%)	27.11	19.18	21.18	26.66	20.04	22.83
	BD-rate (%)	1.51	0.74	0.59	2.15	0.29	1.06
	BD/TS×100	5.57	3.86	2.79	8.06	1.45	4.62

*Hu [13] uses HM 12.1., Nalluri [24] uses HM 16.0, Fan [9] uses HM v16.0.

The ratio between BD-rate and TS (BD/TS) is employed as a measure of compression efficiency variation per each percentage point in the TS. This allows for a fairer comparison between related works that achieve different TS and compression efficiency levels. For better visualization, the BD/TS ratio was multiplied by 100 in Table 6. High BD/TS values represent poor tradeoff between compression efficiency and complexity reduction, whereas values close to zero represent a good tradeoff. Negative values represent compression efficiency improvement. Notice that a ratio of 8.00, 6.30 and 4.62 was achieved with the solutions proposed by [13], [24] and [9], respectively. On the other hand, BZS achieved a ratio of 3.48, 1.03 and -0.05 for the three compared configurations.

In the recent work, the authors in [25] present interesting results in terms of overall encoding time reduction and BD-rate. As the authors only present the average results by video sequence class, it is impossible to perform a comparison by individual video sequence, as the one in Table 6. However, it is possible to consider the average results presented in both works. In the most refined configuration, [25] achieves an overall encoding time reduction of 6.84%, on average, with an increase in BD-rate of 0.08%. In the less refined configuration, the overall encoding time reduction obtained was 16.21%, with an increase of 1.73% in BD-rate. In both cases, the results obtained by [25] are surpassed by BZS when considering the tradeoff between complexity reduction and coding efficiency.

Therefore, it is possible to conclude that the proposed BZS algorithm achieves the best tradeoff between compression efficiency and time savings when compared to related works.

7 Conclusion

This paper presented BZS, a hybrid algorithm for fast block matching motion estimation. The BZS algorithm is based on the TZS algorithm and introduces two novel strategies for IME complexity reduction: the Learning-based Bypass Motion Estimation (LBME) strategy and the Astroid Raster Pattern (ARP). While LBME is capable of predicting cases in which the best MV has already been found and thus bypass the remaining IME steps, ARP prioritizes the most likely areas to find the best candidate blocks during Raster Search, ignoring regions with lower chance of finding best MVs.

BZS was implemented in the HEVC reference encoder and the experimental results have shown that the IME complexity is reduced by 60.98% (Random Access) and 63.05% (Low Delay), on average, in comparison to the TZS algorithm. Besides, BZS reduces the overall HEVC encoding time by 14.32% (Random Access) and 17.64% (Low Delay) in comparison to the case in which TZS is used for IME. In terms of coding efficiency, a negligible loss of +0.0837% (Random Access) and +0.04% (Low Delay) was observed in terms of BD-rate, on average. For certain scenarios, such as FHD and WQXGA resolutions, experimental results have shown that BZS in fact improves compression efficiency by $\times 0.0097\%$ and -0.0325% in BD-rate values, on average. Finally, when compared to the best related works found in the literature, BZS presented the best tradeoff between compression efficiency and complexity reduction.

Acknowledgements This work was supported by the Brazilian Agency for Scientific and Technological Development (CNPq, Brazil). This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), and the Fundação de Amparo à Pesquisa do Rio Grande do Sul - Brasil (FAPERGS).

References

1. Afonso V, Maich H, Audibert L, Zatt B, Porto M, Agostini L, Susin A (2016) Hardware implementation for the HEVC fractional motion estimation targeting real-time and low-energy. *J Integr Circuits Syst* 11(2):106
2. Agarwal PK, Sharir M, Comput ACM (1998) Efficient algorithms for geometric optimization. *Surv* 30(4):412–458. <https://doi.org/10.1145/299917.299918>
3. Belghith F, Kibeya H, Loukil H, Ayed MAB, Masmoudi N (2016) A new fast motion estimation algorithm using fast mode decision for highefficiency video coding standard. *J Real-Time Image Proc* 11(4):675
4. Bjontegaard G (2001) Calculation of average PSNR differences between RD-curves. VCEG-M33
5. Bross B, Chen J, Liu S, Wang Y (2020) Versatile Video Coding (Draft 8). JVET-Q2001
6. CISCO (2018) Cisco visual networking index: Global mobile data traffic forecast update 2017–2022. Cisco Visual Networking Index (VNI)
7. Cheung C-H, Po L-M (2005) Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. *IEEE Trans Multimedia* 7(1):16. <https://doi.org/10.1109/TMM.2004.840609>
8. Correa G, Assuncao P, Agostini L, da Silva Cruz LA (2012) Performance and computational complexity assessment of high-efficiency video encoders. *IEEE Trans Circuits Syst Video Technol* 22(12):1899. <https://doi.org/10.1109/TCSVT.2012.2223411>
9. Fan R, Zhang Y, Li B (2017) Motion classification-based fast motion estimation for high-efficiency video coding. *IEEE Trans Multimedia* 19(5):893. <https://doi.org/10.1109/TMM.2016.2642786>
10. Gonçalves P., Correa G, Porto M, Zatt B, Agostini L (2017) Multiple early termination scheme for TZ search algorithm based on data mining and decision trees. In: 2017 IEEE 19th international workshop on multimedia signal processing (MMSP), pp 1–6. <https://doi.org/10.1109/MMSP.2017.8122245>
11. Goncalves P, Moraes C, Porto M, Correa G (2019) Complexity-aware TZS algorithm for mobile video encoders. *J Integr Circuits Syst* 14(3):1
12. Goncalves P, Porto M, Zatt B, Agostini L, Correa G (2018) Octagonal-axis raster pattern for improved Test Zone Search motion estimation. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 1763–1767. <https://doi.org/10.1109/ICASSP.2018.8462580>
13. Hu N, Yang E (2014) Fast motion estimation based on confidence interval. *IEEE Trans Circuits Syst Video Technol* 24(8):1310. <https://doi.org/10.1109/TCSVT.2014.2306035>
14. ISO/IEC-JCT1/SC29/WG11 (2020) High Efficiency Video Coding (HEVC) Test Model 16 (HM16) Encoder Description. JVET-q2001
15. ITU Telecommunication Standardization (2003) ITU-T Recommendation H.264
16. ITU-T (2008) Subjective video quality assessment methods for multimedia applications. p 910
17. Jia L, Tsui C, Au OC, Jia K (2019) A new rate-complexity-distortion model for fast motion estimation algorithm in HEVC. *IEEE Transactions on Multimedia* 21(4):835. <https://doi.org/10.1109/TMM.2018.2866762>
18. Jing X, Chau Lap-Pui (2004) An efficient three-step search algorithm for block motion estimation. *IEEE Transactions on Multimedia* 6(3):435. <https://doi.org/10.1109/TMM.2004.827517>
19. Karegowda A, Manjunath A (2010) Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management* 2
20. Li N, Zhang Y, Zhu L, Luo W, Kwong S (2019) Reinforcement learning based coding unit early termination algorithm for high efficiency video coding. *J Vis Commun Image Represent* 60:276. <https://doi.org/10.1016/j.jvcir.2019.02.021>. <http://www.sciencedirect.com/science/article/pii/S1047320319300677>
21. Linck I, Gomez AT, Alaghand G (2018) Test Zonal Search based on Region Label (TZSR) for Motion Estimation in HEVC. In: 2018 IEEE 20th international workshop on multimedia signal processing (MMSP), pp 1–6
22. Liu Z, Lin TL, Chou CC (2016) Efficient prediction of CU depth and CU mode for fast HEVC encoding using statistical analysis. *J Vis Commun Image Represent* 38:474. <https://doi.org/10.1016/j.jvcir.2016.03.025>. <http://www.sciencedirect.com/science/article/pii/S1047320316300293>
23. Luo F, Wang S, Wang S, Zhang X, Ma S, Gao W (2019) GPU-based hierarchical motion estimation for high efficiency video coding. *IEEE Transactions on Multimedia* 21(4):851. <https://doi.org/10.1109/TMM.2018.2867260>
24. Nalluri P, Alves LN, Navarro A (2015) Complexity reduction methods for fast motion estimation in HEVC. *Signal Process Image Commun* 39:280. <https://doi.org/10.1016/j.image.2015.09.015>

25. Pakdaman F, Hashemi M, Ghanbari M (2020) A low complexity and computationally scalable fast motion estimation algorithm for HEVC. *Multimedia Tools and Applications* 79. <https://doi.org/10.1007/s11042-019-08593-y>
26. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825
27. Quinlan JR (2004) Data mining tools see5 and c5. 0. <http://www.rulequest.com/see5-info.html>
28. Quinlan JR (2014) C4. 5: programs for machine learning. Elsevier, New York
29. R Core Team (2019) R foundation for statistical computing. R: A Language and Environment for Statistical Computing. <http://www.R-project.org/>
30. Sharman K (2018) Common test conditions for HM video coding experiments. JCT-VC Document AC1100
31. Sullivan GJ, Ohm J, Han W, Wiegand T (2012) IEEE Trans Circuits Syst Video Technol 22(12):1649. <https://doi.org/10.1109/TCSVT.2012.2221191>
32. U.V. Group (2019) Ultra Video Group. Tampere University of Technology. <http://ultravideo.cs.tut.fi/>
33. Valizadeh S, Nasiopoulos P, Ward R (2021) Improving compression efficiency of HEVC using perceptual coding. *Multimed Tools Appl* 80:1
34. Vanne J, Viitanen M, Hämäläinen TD (2014) Efficient mode decision schemes for HEVC inter prediction. *IEEE Trans Circuits Syst Video Technol* 24(9):1579. <https://doi.org/10.1109/TCSVT.2014.2308453>
35. Video Technology Research Group (ViTech) GitHub Repository. <https://github.com/vitech-ufpel/LBME>
36. Wang Y, Fan X, Xiong R, Zhao D, Gao W (2021) Neural network-based enhancement to inter prediction for video coding. *IEEE Trans Circuits Syst Video Technol* 1–1
37. Zhang Y, Zhang C, Fan R, Ma S, Chen Z, Kuo CJ (2019) Recent advances on hevc inter-frame coding: from optimization to implementation and beyond. *IEEE Trans Circuits Syst Video Technol* 1–1
38. Zhu C, Lin X, Chau L-P (2002) Hexagon-based search pattern for fast block motion estimation. *IEEE Trans Circuits Syst Video Technol* 12(5):349. <https://doi.org/10.1109/TCSVT.2002.1003474>
39. Zhu S, Ma K-K (2000) A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans Image Process* 9(2):287. <https://doi.org/10.1109/83.821744>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.