Rui Fan, Yongfei Zhang, and Bo Li

[[1]](#footnote-1)

Motion Classification-based Fast Motion Estimation for High Efficiency Video Coding

*Abstract*—High Efficiency Video Coding (HEVC), the latest video coding standard, is becoming popular due to its excellent coding performance, in particular in the case of high-resolution video applications. However, the significant gain in performance is achieved at the cost of substantially higher encoding complexity than its precedent H.264/AVC, in which motion estimation (ME) is one of the most time-consuming parts that effectively removes temporal redundancy. Test zone search (TZS) is adopted as the default fast ME method in the reference software of HEVC; however, the computational complexity is still too high for real-time video applications. Several fast ME algorithms have been recently proposed to further reduce ME complexity; however, these approaches typically lead to non-negligible performance loss. To address this problem, this paper proposes a motion classification-based fast integer-pixel ME algorithm. By exploring the motion relationship of neighboring blocks and the coding cost characteristic, the Prediction Unit (PU), the basic unit of ME, is first categorized into one of three classes, namely, motion-smooth PU, motion-medium PU and motion-complex PU. Then, different search strategies are carefully designed for PUs of each class according to their respective motion and content characteristics. Furthermore, a fast search priority-based partial internal termination scheme is presented to rapidly skip impossible positions that speeds up cost computation during the ME process. Extensive experimental results demonstrate that the proposed algorithm achieves as much as 12.47% and 20.25% reductions in total encoder complexity when compared with TZS under low delay P configuration and random access configuration, respectively, with negligible rate-distortion degradation; thus, it outperforms state-of-the-art fast ME algorithms in terms of both coding performance and complexity reduction.

*Index Terms*—Video Coding, HEVC, Motion estimation, Motion classification

# I. INTRODUCTION

T

HE preceding video coding standard H.264/AVC [1] no longer meets the demand of high and ultra-high definition video content. To achieve better video coding rate distortion (RD) performance over H.264/AVC, a new video coding standard called high efficiency video coding (HEVC) has been developed by the joint collaborative team on video coding. Compared with H.264/AVC high profile, HEVC provides bit rate reduction of as much as 50% for equal perceptual quality [2]. This improved compression performance is achieved at the cost of up to 2-10 times higher computational complexity [3] as a result of employing quadtree based coding unit (CU), large and asymmetric prediction unit (PU), residual quadtree based transform unit (TU), advanced motion vector prediction (AMVP) and many other efficient coding tools [4].

Motion estimation (ME), which is the most important part of video compression, is the major contributor to compression efficiency. It is an effective tool for finding the best matched block in the reference frames to reduce temporal redundancy between successive frames. Motion vector (MV), representing the displacement between the best matched block and the current prediction block, is generated by ME. Although ME plays an important role in video coding, the high computational complexity makes it quite difficult to implement in real-time applications. Therefore, it is necessary to reduce computational complexity and speed up the ME process. The entire ME process is made up of three parts, namely, MV prediction, integer-pixel ME and sub-pixel ME. MV prediction predicts the start search position for the following motion search by utilizing the neighboring motion information. Then, the best-matched block in the integer-pixel domain is found using search strategies. Around the optimal integer-pixel position, sub-pixel ME is carried out to obtain the final best-matched sub-pixel position. This paper mainly focuses on the complexity reduction of integer-pixel ME for the following reasons. First, the integer-pixel ME is very time-consuming and might occupy 40-80% of the total encoding time. Second, the accuracy of the integer-pixel ME has a large influence on the performance of the subsequent sub-pixel ME. For simplicity of presentation, unless otherwise specified, ME refers to integer-pixel ME throughout the paper.

In this paper, we propose a motion classification-based fast ME algorithm. First, based on the observation that PUs of different video regions exhibit quite different motion characteristics and the complexity of ME in regions with different motion characteristics is quite different, PUs are classified as motion-smooth PUs, motion-medium PUs and motion-complex PUs based on the motion vector relationship of neighboring PUs and the coding cost characteristic. Second, different search strategies are carefully designed for PUs of different categories according to their respective motion characteristics. In addition, a fast search priority-based partial internal termination algorithm is presented to skip impossible positions by early terminating redundant cost computation during the ME process.

The rest of this paper is organized as follows. Section II reviews some of the fast ME algorithms proposed recently. In Section III, experimental observations and motivations are presented. Our motion classification-based fast ME algorithm is elaborated in Section IV. Experimental results are shown in Section V. Finally, we conclude the paper in Section VI.

# II. Related Works

|  |  |
| --- | --- |
| (a) | (b) |

Fig. 1. Initial search patterns of TZS. (a) Diamond search. (b) Square search.

Block matching algorithm (BMA) is the most popular search algorithm for ME because it is simple to implement but also performs reasonably. The basic idea of BMA is that the frame is divided into fixed-size blocks. The most matched block within a search window in the reference frame is obtained based on the rate-distortion cost (*RDCost*), which is measured in Eq. (1) and (2) as follows

(1)

 (2)

where ***mv*** = (x, y) is the current MV, ***pmv*** is the predictive motion vector (PMV) and *λ**motion* is the Lagrange multiplier related to the quantization parameter. *R*(***mv*** *-* ***pmv***) represents the number of bits for coding the difference between motion vector ***mv*** and predictive motion vector ***pmv*** based on a look-up table. *SAD* is the distortion between the current block *s* and the reference block *r* determined by ***mv***, which is a measurement of distortion in the process of integer-pixel ME. In Eq. (2), *s*(*i, j*) is the pixel value at position (*i*, *j*) in the current frame; c(*i-x*, *j-y*) represents the pixel value at position (*i-x*, *j-y*) in the reference frame. *W* and *H* denote the width and height of the block, respectively. The earliest and most straightforward full search (FS) traverses all the positions in the search window and obtains the optimal MV with the minimum *RDCost* through the most exhaustive computation. Although FS provides the best quality amongst various ME algorithms, its computational complexity is very high and can involve as much as 40-80% of the total encoding time.

To address this drawback and achieve a balanced point between the coding performance and computational complexity, test zone search (TZS) [5] is implemented as the build-in fast search mechanism (FSM) in the HEVC test model (HM). First, the start search position is determined by checking the PMV and zero motion. As a second step, a diamond search pattern or square search pattern is implemented as shown in Fig. 1, and an additional raster search is performed when the difference between the obtained motion vector and start position is too large. In the last step, an extra diamond search or square search is performed as a refinement search until the best search position is picked.

Although TZS reduces ME complexity to a much greater extent than FS, the computational complexity is still huge for real-time systems because there are too many search points. To further reduce the complexity of TZS, in addition to parallel accelerate the ME process based on many-core processors in [6-7], plenty number of fast ME algorithms have been developed in the references [8-29]. Generally speaking, these fast ME algorithms can be divided into three categories.

The algorithms in the first category focus on search pattern designs, which can significantly reduce the computational complexity through simpler search patterns. Examples of such algorithms include typical search patterns, such as three step search (TSS) [8], efficient three step search (E3SS) [9], diamond search (DS) [10], hexagon-based search (HEXBS) [11] and cross-diamond-hexagonal search (CDHS) [12]. These algorithms reduce computational complexity significantly, but they easily become trapped in a local minimum which leads to performance loss. To achieve a better balance between coding performance and computational complexity, some search patterns for HEVC have been proposed, such as pentagon search pattern [13] and rotating pentagon search pattern [14]. A background-foreground division based search algorithm is proposed to accelerate the motion search in surveillance video coding in [15], and Zhu proposes a hash-based block matching scheme for ME of screen content coding in [16]. However, the applicability is not strong for other scenarios. A fast ME with a directional search based on square search pattern is introduced in [17]. In addition, [18] presents a fast ME algorithm based on quadratic prediction. The start search point for each step is predicted by a quadratic function in the last step, with a coarse-to-fine search step size. Nevertheless, the motion characteristics of different regions might be quite different. The regions with sufficient movement should consume more computational complexity to maintain coding performance, while the relatively stationary regions should find the optimal search position faster to accelerate the ME process. Therefore, the above uniform search pattern designs are suboptimal and could be further improved.

The second category concentrates on search window reduction, which restricts the search range directly to decrease the total number of search positions. For search window size reduction, the usual way is to use a pre-defined fixed search range related to the maximum block size. The search window size for H.264/AVC is 16, and it is set as 64 for HEVC. For the sake of accelerating the motion search process, the search window size can vary dynamically based on the MV distribution of neighboring blocks. HEVC reference software uses an adaptive search range algorithm (ASR), which changes the search window size based on the temporal distance between the current frame and the reference frame. Assuming each component of MVDs may follow a Laplace distribution, Ko presents a fast ME method where the search range of horizontal and vertical directions may be different from each other and are constrained by the hitting probability of MVDs [19]. [20] concludes that the Laplace distribution fails to fit the MVD distribution well based on a number of experimental simulations and uses Cauchy distribution to model the search range for one frame more exactly. The search range in [21] is divided into three classes, namely, homogeneous motion, normal motion and complex motion, which are determined by the MV distribution of neighboring blocks. The above methods adjust the search range by referencing the information of motion vector differences (MVDs) in the spatial-temporal neighboring blocks. To predict the search window size more accurately, the MVD information of the father CU is introduced. [22] estimates the search window size for each CU as soon as the MVD of the CTU is known based on the observation of the linear relation between search window size and the MVD. Although these approaches can reduce the number of search points by restricting the search window size, non-negligible coding performance loss might occur when the optimal search position is outside the search window due to inaccurate search window size, as the MVD correlation between spatial-temporal neighboring blocks is not strong enough.

The last category is composed of early termination algorithms, which accelerate the motion search by early termination of the whole or part of the ME process, and can be further classified into two sub-categories. The algorithms in the first sub-category aim at terminating the on-going ME procedure when the current search point is supposed to achieve acceptable coding performance. For example, [23-25] terminate the ME process when the coding cost is smaller than a threshold derived from the information of spatial and temporal neighboring blocks. However, these early termination methods usually cause performance loss when the location of early determination is not the optimal one with the smallest *RDCost*. The algorithms in the other subcategories try to terminate or skip the impossible points and continue to check the subsequent points. Some representative fast ME algorithms, such as the successive elimination algorithm (SEA) [26], multilevel SEA [27], global SEA [28] and the confidence interval-based ME method (CIME) [29], are presented to avoid the SAD computation of impossible search points in the whole FS process. These algorithms can achieve excellent coding performance for very complex search patterns such as FS, but the extra calculation that is necessary to skip the impossible search points is excessive for simple search patterns. Recently, fast ME algorithms have tended to employ simple search patterns with high predictive accuracy, so these algorithms are no longer appropriate. Another limitation of these algorithms is the additional memory space required to store the information used to rapidly skip the impossible search points, which makes it difficult to implement in hardware platforms.

Based on the analysis above, it is apparent that there are still some problems in the existing fast ME algorithms. It is suboptimal to employ uniform search patterns or early termination criteria for PUs that have different content and motion characteristics, as the uniform allocation of limited and valuable computation resources might cause fruitless ME in simple and smooth content regions; this leads to performance loss in motion- and content-complex regions due to inadequate ME efforts, i.e., search patterns and strategies that are too simple, search window sizes that are too small or terminations that are too early. As a consequence, if regions with different content and motion characteristics can be distinguished ahead of schedule, we can implement the appropriate ME for PUs of different regions, thus achieving a greater balance between computational complexity and coding performance.

# III. Observations and Motivations

Based on extensive experiments, the complexity of ME in regions with different motion characteristics varies. Fig. 2 shows an example of MVD distribution in different regions. The black point is the initial search point related to PMV, and the red point denotes the final search position. As can be seen, the MVD distributions in different regions are quite disparate. More specifically, when the amplitude of the MVD is small, the prediction accuracy of PMV is high. In other words, the computational load that is used to obtain the optimal MV is small. Region 1 and 3 belong to this case, as shown in Fig. 2. Regions with high PMV accuracy are mainly located in the background, which is characterized by mild regular movement and internal areas with smooth textured moving objects. The MVD in these regions is equal to or close to zero. The coding cost distribution around PMV is monotonous in most cases. Hence, it is easy to obtain the optimal search point quickly and effectively. A simple search pattern with fast convergence ability could be designed for the ME process in these regions, reducing computational complexity without decreasing the coding performance.

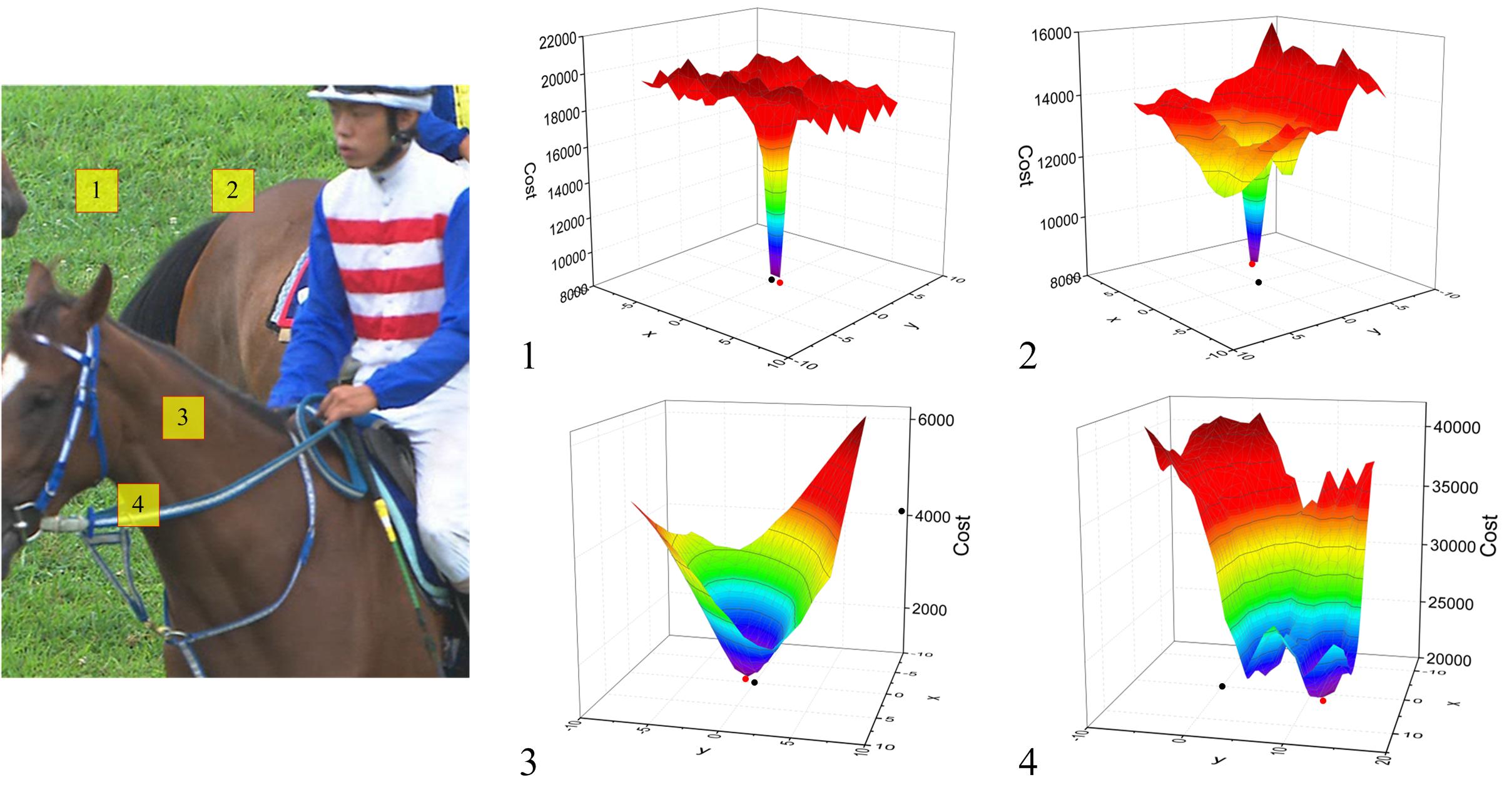


Fig. 2. The coding cost distribution in different regions for RaceHorses sequence.

On the other hand, when the amplitude of the MVD is large, the prediction accuracy of PMV is fairly poor. Regions with low PMV accuracy are mainly located in the edge areas of fast moving objects characterized by complex content and motion. If we continue to employ simple search patterns in these regions, it will inevitably result in obvious coding performance loss, as the coding cost distribution around the PMV is usually complex and multimodal. It is easy to fall into a local optimum, which can be observed in region 2 and 4, in particular the latter. To obtain the global optimal point, a more complex search mechanism should be designed to adjust to the movement characteristics and maintain the coding performance while consuming as little computational complexity as possible.

Based on the above analysis, the main idea of this paper is to classify the coding PUs into different categories according to their respective movement characteristics. Thereafter, different search strategies are carefully designed for different categories of PUs to find the optimal motion search point based on the trade-off between coding complexity and coding performance.

# IV. Proposed Motion Classification-Based Fast Motion Estimation Algorithm

According to the above observations and statistical analysis, the ME characteristics in different video regions are quite distinct. Thus, different designs of search strategies would be more reasonable. Based on these studies, a motion classification-based fast search algorithm is proposed. First, a motion-based PU classification scheme is developed by exploring the motion information of neighboring PUs. Second, search strategies are carefully designed for PUs of different categories to achieve a better trade-off between coding complexity and coding performance. Furthermore, a search priority-guided fast partial internal early termination algorithm is proposed to early terminate unnecessary cost calculation procedures and thus further reduce search complexity. The three parts of our proposed algorithm are discussed in detail in the following subsections.

|  |  |  |
| --- | --- | --- |
| PartyScene_19  (a) Original frame (PartyScene) | RaceHorses_5  (d) Original frame (RaceHorses) | BasketballDrill  (g) Original frame (BasketballDrill) |
| 19_2  (b) Two classes (PartyScene) | 5_2  (e) Two classes (RaceHorses) | 25_2  (h) Two classes (BasketballDrill) |
| 19_3  (c) Three classes (PartyScene) | 5_3  (f) Three classes (RaceHorses) | 25_3  (i) Three classes (BasketballDrill) |

Fig. 3. PU classification examples.

## A. Motion Characteristics-based PU classification

For PUs in different regions, different search strategies should be designed to accommodate their respective motion characteristics. To select appropriate search strategies for PUs with different motion characteristics, we need to first classify PUs. Since correlation exists between the current PU and its neighboring PUs, the current coding PU can be categorized as a motion-smooth PU or a motion-complex PU as follow, by using the two candidate MVs in AMVP.

 (3)

where *MV**1* and *MV**2* are the two candidate MVs (quarter pixel precision) in AMVP, which are the MVs of the spatial-temporal neighboring PUs. If *MV**1* equals *MV**2*, the current PU is a motion-smooth PU, as the moving trends of its neighboring PUs are the same and the current PU also has a high probability of having a similar moving trend. Otherwise, when *MV**1* is not equal to *MV**2*, the current PU might locate in regions with abundant movement. The current PU and its neighboring PUs probably possess different moving trends; thus, the current PU is classified as a motion-complex PU. The classification result using Eq. (3) is shown in Fig. 3 (b), (e) and (h) for the sequence PartyScene, RaceHorses and BasketballDrill, respectively. The corresponding original frames are also illustrated in Fig. 3 (a), (d) and (g). Motion-smooth PUs make up the blue regions; it is thus found that motion-smooth PUs generally indicate the background or smooth texture regions in moving objects. The red regions denote motion-complex PUs, which are primarily located on the edges of moving objects. The motion characteristics of these two categories of regions are quite different, and this is consistent with observations.

 (4)

To further enhance PU classification accuracy, the *RDCost* of the PMV is also introduced to refine the classification in Eq. (4), in which two cases are considered (PMV indicates the start search position, which is determined as either *MV**1* or *MV**2* with a smaller *RDCost*). In one case, the current coding PU might exhibit quite different motion characteristics, although *MV**1*=*MV**2* is met. At this moment, the *RDCost* of the PMV is usually large and can be used to determine this case. The other case is even if *MV**1*≠*MV**2*; it is not necessary to employ a complex motion search or motion-complex regions when the *RDCost* of the PMV is small enough. Therefore, we classify the PUs of these two cases as motion-medium PUs. In Eq. (4), *RDCost*(*PMV*, *λ**motion*) represents the *RDCost* of PMV. *Th**ms* is designed to evaluate whether the *RDCost* of the PMV for the current PU is sufficiently small and to reduce the misclassification into a motion-smooth PU. It should satisfy the requirement that the *RDCost* of the PMV is sufficiently small for motion-smooth PU. Similarly, *Th**mc* is designed to reduce the misclassification into a motion-complex PU. By using *Th**ms* and *Th**mc*, another category of PUs, namely, motion-medium classification, is generated to reflect the PUs that own the motion characteristics between smooth motion and complex motion. *Th**ms* and *Th**mc* are determined to be linear functions related to the average *RDCost* *Th**avg* when PMV=MV(MVD=0), which is shown in Eq. (5) as follows

 (5)

TABLE I

Probability That ((|MVD(x)|+|MVD(y)|))<=SR for Different PU Categories and Different SRs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PartyScene | RaceHorses | BasketballDrill | Average |
| P(SR = 0 | *Motion-smooth PU*) | 90.0% | 51.1% | 96.8% | 79.3% |
| P(SR = 1 | *Motion-smooth PU*) | 95.6% | 84.6% | 99.3% | 93.2% |
| P(SR = 2 | *Motion-smooth PU*) | 98.9% | 94.1% | 99.8% | **97.6%** |
| P(SR = 0 | *Motion-medium PU*) | 63.8% | 42.8% | 79.4% | 62.0% |
| P(SR = 2 | *Motion-medium PU*) | 94.0% | 88.9% | 93.4% | 92.1% |
| P(SR = 4 | *Motion- medium PU*) | 96.1% | 92.1% | 94.4% | **94.2%** |
| P(SR = 0 | *Motion-complex PU*) | 37.7% | 22.9% | 46.0% | 35.5% |
| P(SR = 2 | *Motion-complex PU*) | 72.6% | 52.0% | 66.9% | 63.8% |
| P(SR = 4 | *Motion-complex PU*) | 79.7% | 60.7% | 72.3% | 70.9% |
| P(SR = 8 | *Motion-complex PU*) | 85.0% | 69.2% | 77.6% | 77.3% |
| P(SR = 16 | *Motion-complex PU*) | 88.8% | 83.6% | 85.4% | 86.0% |
| P(SR = 20 | *Motion-complex PU*) | 94.4% | 91.4% | 90.7% | **92.2%** |

where the two weighting factors *α* and *β* are empirically set as 0.9 and 1.1, respectively, based on extensive experiments. *Th**avg* is the average *RDCost* when PMV=MV(MVD=0), which can be adaptively updated as

 (6)

where *ref\_cost* is the sum of *RDCost* in the previous frame, satisfying PMV=MV(MVD=0). *cur\_cost* is the sum of the *RDCost* in the current coding frame, thus far satisfying PMV=MV(MVD=0). *ref\_count* is the number of cases where PMV=MV(MVD=0) in the previous frame, and *cur\_count* represents the number of cases where PMV=MV(MVD=0) thus far in the current coding frame. The information regarding the current coding frame is insufficient as a reference for the subsequent ME process until *cur\_count* is larger than Num; therefore, the information regarding the reference frame is used to calculate *Th**avg* when *cur\_count* is smaller than Num. After the current frame coding process terminates, the values of *cur\_cost* and *cur\_count* are assigned to *ref\_cost* and *ref\_count*, respectively. When the current coding PU satisfies PMV=MV(MVD=0), *cur\_count* is first added by 1. *cur\_cost* is updated as illustrated in Eq. (7). *pu\_cost* is the *RDCost* for the current coding PU if it satisfies PMV=MV(MVD=0). *pu\_width* and *pu\_height* are the width and height, respectively, for the current coding PU. It is apparent that the calculations of the threshold values and the related variables are updated on the fly and do not rely on offline trainings, contributing little to the encoding computational complexity. Therefore, our proposed PU classification method has strong practical uses and can be implemented in different systems and platforms.

 (7)

The regions painted with green in Fig. 3 (c), (f) and (i) are further classified by Eq. (4), which stand for the motion-medium PUs. The importance of this improved classification is its ability to reduce the misjudgment of the classification rule in Eq. (3). We find that the green regions in Fig. 3 are either blue regions with a certain motion complexity or red regions with a little motion complexity. Consequently, the classification method in Eq. (4) is more reasonable, and the classification accuracy has been improved compared with the method in Eq. (3).

## B. Classification-adaptive Motion Estimation Strategies

According to the above PU classification, the coding PU is classified as one of the three types, namely, motion-smooth PU, motion-medium PU and motion-complexity PU. As has been observed in Section III, the MVD distributions in different regions are quite disparate. More specifically, the probability distribution of the MVD (integer-pixel precision) between PMV and the best MV is listed in TABLE I for the three categories of PUs for three typical video sequences, i.e., *PartyScene*, *RaceHorses* and *BasketballDrill*. SR is a range restricting the MVD size ((|MVD(x)|+|MVD(y)|)), and different values of SR are configured to guarantee that the average accuracy of optimal ME (the best MV) is above 90%.

From TABLE I, three conclusions can be drawn. First, for motion-smooth PUs, PMV is very likely to be the optimal MV and the probability that the optimal MV is close to PMV, e.g., within 2 integer pixels, is quite high (over 97% on average). Second, for motion medium PUs, there is still a great probability that MVD equals zero. The optimal MV position of most of the PUs (over 94% on average) can be found within a range of 4 around the PMV. Finally, even for motion-complex PUs, there is a certain probability that MVD equals zero, but the optimal MV might be located a large distance from the PMV. To avoid severe coding performance loss, a larger SR is required; e.g., an SR of 20 can help improve ME accuracy from 35.5% to 92.2%. Thus, the movement feature for different types of PUs is quite different, which further verifies our observation and motivations in Section III. According to the respective movement feature, different search strategies are designed well, as presented in the following sections.

**(1)****Simple and Fast Search for Motion-smooth PUs**

For motion-smooth PUs, the movement is regular or stationary, and the accuracy of AMVP is quite high. Given that the probability listed in TABLE I is very high (>90%) that the distance between PMV and the best MV is less than one integer pixel, a small diamond search pattern shown in Fig. 4 is employed to find the best MV. The small diamond search pattern terminates until the central point achieves the smallest RD cost. If the point achieving the smallest RD cost is not the central point, that point will be determined as a new central point. The small diamond search pattern is put into effect again, excluding the redundant searched points. In most cases, the number of searched points is 5 or 8. On the contrary, FS searches 129×129=16641 points for the search range of [-64, 64], while TZS consumes dozens of times more search points than the small diamond search pattern. Therefore, this search strategy can significantly decrease the number of searched points with slight coding performance loss.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Fig. 4.  Small diamond search pattern for motion-smooth PUs | (a) Priority-based spiral search pattern. | (b) Multiple-directional search pattern. |
| Fig. 5. Motion search pattern for motion-medium PUs. | |

**(2)****Priority-Based Spiral Search (PSS) and Multiple-Directional Search (MDS) for Motion-medium PUs**

From TABLE I, although the motion or content of these PUs is much more complicated than those of motion-smooth PUs, the probability is still considerable that the PMV equals the final optimal MV and that most of the optimal MV positions (more than 94% on average) lie in the range of 4 around the PMV. Motivated by this, a priority-based spiral search pattern (PSSP) is employed to determine whether early terminate the following ME process or not, as shown in Fig. 5 (a). Moreover, there is only one search path in ordinary search pattern designs [8-18], which may fall into the local optimum. To address this problem and better accommodate the motion characteristics of motion-medium PUs, a multiple-directional search pattern (MDSP) is implemented based on the PSSP, which is shown in Fig. 5 (b).

First, PSSP is employed, as shown in Fig. 5 (a), where the points are ordered according to their distances to the PMV and a point with a smaller number has a higher priority. Hence, the thirteen points within a range of 2 integer pixels around the PMV are checked from point 0 to point 12. Then, these thirteen points are sorted by their coding costs. An early termination is met when the coding cost of point 0 is minimal and there is no additional minimal point among the remaining twelve points. If the early termination is satisfied, the ME procedure is terminated after checking the twelve points of the PSSP. Otherwise, a further refinement search should be employed until the best search point is obtained. Since some of the twelve points might yield equal or similar coding costs and mis-determination of which point is better as a new start position for the later motion search process might result in obvious quality loss due to the local optimization problem, MDSP is proposed to guarantee the coding performance. As shown in Fig. 5 (b), three minimum points among the twelve points of a PSSP are chosen, and each of these three minimum points is determined as a start search position for an individual refinement search. According to the difference of start search positions, it can be divided into three scenarios.

**Scenario 1 (S1): The start position locates in point 0, 1, 2, 3 or 4**

If this is satisfied, the process of the refinement search is terminated immediately, as these points are the internal points of the PSSP and their coding costs are less than the coding costs of their surrounding points. It is apparent that these points have achieved a decent coding performance in the local area.

**Scenario 2 (S2): The start position locates in point 5, 6, 7 or 8**

When this occurs, we first check the three points that have not been tested among the eight surrounding points. Then, the point with the smallest coding cost is selected as a new start search point. Three points around the new start point, which are along the direction from the last start search point to the current start search point, are searched until the current start search point is found to have the smallest coding cost. This search strategy checks three points in each step along the motion search trend; this determines the optimal search position with limited computational complexity.

**Scenario 3 (S3): The start position locates in point 9, 10, 11 or 12**

In this scenario, three points among the above, left, below and right points have not been checked yet, which conforms to the small diamond search pattern. Hence, the small diamond search pattern is employed to test the other three points until the central point achieves the smallest coding cost.

**(3)****Large Range Scalable Cross-search for Motion-complexity PUs**

From TABLE I, even for motion-complexity PUs, there is about a chance of 35% that MVD equals zero and a probability of 70% still exists that the distance between the optimal position and the start search position is no more than 4 integer pixels. Therefore, PSSP and MDSP are also employed to deal with these conditions. However, there is still a large probability that the optimal position is far from the start search position. If we only employ PSSP and MDSP, it is very easy to fall into a local minimum. To more accurately find the global optimal position, a large range scalable cross-search pattern shown in Fig. 6 (a) is designed to improve the search precision for PUs with large or complex movements. Eventually, the optimal search point is determined between PSSP+MDSP and the scalable cross-search pattern by comparing their coding costs.

The probability for horizontal and vertical movements of natural objects is high since the global movement of the camera is usually in the horizontal or vertical direction. Therefore, with respect to the initial search points, we check the points in the horizontal or vertical direction with step sizes of 4, 2 and 1, in which *R* is the adaptive search range. The coarse-to-fine scalable cross-search enlarges the search range to find the global optimal position with limited computational complexity. Finally, a small cross search pattern including four positions (top, left, bottom and right) is advised as a refinement search.

|  |  |
| --- | --- |
|  |  |
| (a) Scalable cross-search pattern. | (b) The positions of adjacent PUs of current PU. |
| Fig. 6.  Scalable cross-search pattern for motion-complex PUs. | |

The search radius *R* is an impact factor to regulate the relationship between computational complexity and search precision. Even for motion-complex PUs, the value of *R* should not be fixed owing to the movement intensities in different moving regions being widely divergent. The design of the scalable cross-search pattern is to find the best matched block under the condition that large and complex movement happens. MVD, which is the distance between the optimal MV and PMV, can be used to measure the moving intensity. As a result, *R* for the current coding PU can be predicted by using the MVD information of its spatial-temporal neighboring PUs and its upper parent PU. We define *SR**NB* to represent the MVD intensity of the neighboring PU (*NB* PU), which is shown as follows

 (8)

where *MVD**NB**(x)* and *MVD**NB**(y)* are the horizontal and vertical coordinate, respectively, of the MVD for the *NB* PU, which are both of integer-pixel precision. The *NB* PU is one of the spatial-temporal neighboring PUs (*S0*, *S1*, *S2*, *S3* and *S4*, as used in AMVP), the upper-level parent PU U or the collocated PU C in the reference frame for the current coding PU, which is shown in Fig. 6 (b). Then, the search radius *R* shown in Fig. 6 can be expressed as

 (9)

where ⌈*x*⌉ denotes rounding decimal *x* up to an integer. max{*SR**NB*} represents the maximum value of *SR**NB*for all the *NB* PUs. The minimum value of *R* is set to 4, as PSS and MDS have effectively checked the positions within the range of 4. In addition, the maximum value of *R* is set to 16, to guarantee the maximum search range can reach 20, which is considered from TABLE I.

In Fig.6 (a), point 0 denotes the initial search point by PMV. The points labeled by 1 are the following search points with a search step size of 4, and the search radius R is introduced to control the number of search points in the large scalable cross-search pattern. After the optimal point labeled by 1 is found, the two neighboring points labeled by 2 are tested with a step size of 2. In the same way, the two neighboring points labeled by 3 are tested with a step size of 1 after the points labeled by 2 are checked.

## C. Search Priority-Guided Fast Partial Internal Early Termination Algorithm (PG-PET)

The goal of ME is to traverse the search positions through time-consuming rate-distortion cost calculation and find the best MV with the smallest rate-distortion cost. If we can pre-judge before or at least during the rate-distortion cost calculation whether the current search position might yield a better or worse rate-distortion cost than the optimal one achieved thus far, the time-consuming rate-distortion cost calculation and thus the ME could be much accelerated.

Motivated by this, we present a search priority-guided fast partial internal early termination algorithm in this section. On one hand, we decompose the current PU into several parts, and the rate-distortion cost for the whole PU is calculated part by part. If its partial rate-distortion cost calculated thus far has exceeded the current optimum, the rate-distortion cost calculation of the remaining parts could be skipped as the rate-distortion performance of the current search position definitely could not be better. On the other hand, to maximize the effectiveness of partial internal early termination, the search priority is well-designed for the search positions in each search step for different categories of PUs according to their respective characteristics of the motion search process.

The rate-distortion cost for ME is computed as Eq. (1), in which the distortion is measured by *SAD*. If the current search position is superior to the current optimal search position, the rate-distortion cost of the current coding search position should be less than that of the current optimal search position, which is listed in Eq. (10).

 (10)

where ***mv******c*** and ***mv******best*** denote the MV related to the current coding position and the current optimal search position, respectively. The remaining symbols are the same as those in Eq. (1). *RDCost*(***mv******best***, *λ**motion*)is the best rate-distortion cost for the current coding search position, which has been calculated in the previous ME process. *R*(***mv******c*** *-* ***p***) is the coding bits of MV related to the current coding position, which can be obtained by the look-up table at the beginning of the current SAD calculation.

 (11)

If the current coding search position is superior to the current optimal search position, it must be satisfied that the *SAD* related to the current coding search position is less than *RDCost*(***mv******best***, *λ**motion*) *– λ**motion* *R*(***mv******c****-****pmv***), which is defined as a threshold *Th**PG-PET*. There is no way to have a better search position for the current coding search position if its partial or total *SAD* is larger than or equal to *Th**PG-PET*, which can be computed before the current *SAD* calculation.

|  |  |
| --- | --- |
|  |  |
| Fig. 8. Search priority decision on the small diamond search pattern. | Fig. 9. Search priority decision on the multiple-directional search pattern. |



Fig. 7. An example of fast partial internal termination method for an 8×8 PU.

The *SAD* calculation procedure is decomposed into several parts; partial accumulated *SAD* (PASAD) between the current block and the reference block can be used to skip impossible reference positions before ending the *SAD* calculation. Thus, the rest of the *SAD* calculation can be stopped to speed up the ME process. Fig. 7 illustrates an example based on the motion search process of an 8×8 PU. We decompose the PU into four parts of the same size, and the *SAD* of each part is calculated. PASAD is accumulated according to the calculated *SAD* sequentially. The threshold *Th**PG-PET* is a criteria of early termination on the basis of the current optimal rate-distortion cost and the current MV coding bits. If the PASAD at any internal stage is greater than or equal to *Th**PG-PET*, it is impossible to achieve better rate-distortion performance for the current search position, and the remaining *SAD* calculation can be terminated immediately. It is shown in Fig. 7 that the PASAD (198) in the third part is greater than *Th**PG-PET* (173). The current coding search position cannot achieve better rate-distortion performance; hence, the *SAD* calculation of the fourth part can be skipped. As additional decisions on whether to stop the rest of the *SAD* calculation are introduced, it does not mean that it is efficient when the number of parts equal to the maximum value, which is set to 4 empirically.

The above analysis indicates that the fast partial internal termination can speed up the ME process without sacrificing coding quality. However, there are several search positions in each search step. To further reduce the computational complexity of ME, the search order in each step should be adaptively adjusted. The main principle is that the search position, which is probable to achieve a better rate-distortion performance, should have a higher priority to be tested since it might obtain a smaller threshold *Th**PG-PET* quickly and more search positions can be eliminated by the smaller *Th**PG-PET*. Specifically, the strategy to implement the fast partial internal termination method with our proposed search patterns for different kinds of PUs is presented in detail as follows.

**(1)** **PG-PET for motion-smooth PUs**

Motion-smooth PUs are usually stationary or possess simple movements; hence, the small diamond search pattern is employed to adapt its movement characteristics. To describe the search order more clearly, the positive integer is used to measure the search priority. The search position with a smaller positive integer has a higher priority to be tested.

Fig. 8 is an example of a search priority decision on the small diamond search pattern. Assuming that Point 4 achieves better rate-distortion performance than Point 0 to Point 3 in the first search step, Point 4’ is set to the highest search priority of 1, which is in the direction from Point 0 to Point 4. For Point 1’ and Point 3’, which one will obtain a higher priority depends on the rate-distortion cost relationship between Point 1 and Point 3. In Fig. 8, the rate-distortion cost of Point 1 is less than that of Point 3, so priority 2 is given to Point 1’ and priority 3 is given to Point 3’, as Point 1’ might be more likely to achieve a better rate-distortion performance.

**(2)** **PG-PET for motion-medium PUs**

To better fit local complex movement, the PSSP and MDSP are adopted for motion-medium PUs. The PSSP is employed to find the three minimum search positions for the subsequent refinement search. If we implement PG-PET on a PSSP, we can easily find the minimum search position, but we might not exactly find the three minimum ones. Therefore, fast partial internal termination is not recommended here. The MDSP is employed as a refinement search. Three conditions might be met, which have been introduced previously. Scenario 1 is beyond the scope of consideration. The implementation of the fast partial internal termination for Scenario 3 has been described above, as it employs the small diamond search pattern. We focus on the implementation for Scenario 2 as follows.

The search direction in Scenario 2 can be categorized into eight directions. Among the three tested points in each search step, the one in the search direction is set to the highest search priority. Therefore, Point a2, b2, c2, d2 and e2 are the highest priority in each search step. As for the remaining two points in each search step, the priority setting is divided into two cases. The one case is where the current search direction is consistent with the search direction in the last step, just like the points labeled by b and d. In this case, the search priority is confirmed based on the rate-distortion relationship of the same position in the last step. As a result, the search priority of Point b1 and b3 are determined by comparing the rate-distortion cost of Point a1 and a3. A smaller rate-distortion cost indicates a higher search priority. In the same way, the search priority of Point d1 and d3 can be obtained. If the search priority of Point b1, b3, d1 and d3 are shown in Fig. 9, it should satisfy *RDCost*(***mv******a1***, *λ**motion*) < *RDCost*(***mv******a3***, *λ**motion*) and *RDCost*(***mv******c1***, *λ**motion*) > *RDCost*(***mv******c3***, *λ**motion*). The other case is where the current search direction is different from the search direction in the last step, for example, the points labeled by c and e. Then, the search point whose direction is the same as the search direction in the last step is given a higher search priority than the other search point of the remaining two search points. Consequently, the search priority of Point c1, c3, e1 and e3 are given as illustrated in Fig. 9.



Fig. 10. Search priority decision on the large scalable cross-search pattern.

**(3)** **PG-PET for motion-complex PUs**

Apart from adopting the search patterns for motion-medium PUs, motion-complex PUs also employ a scalable cross-search pattern to suit different levels of a wide range of complex movement. The first step for the scalable cross-search pattern is to test the points in four directions (0°, 90°, 180° and 270°) with a step size of 4. Fast partial internal termination can be implemented as soon as the optimal position of the MDSP is obtained. The space is divided into eight areas, and each area indicates one kind of priority setting. For example, if the position related to the best *MV**MDSP* derived from MDSP is located in region IV as shown in Fig. 10, the search priority from high to low is the 180°, 90°, 270° and 0° directions since the points closer to the best position derived from MDSP are more likely to achieve better rate-distortion performance than are the other points.

# V. Experimental Results and Discussions

## A. Experimental Settings

To evaluate the performance of the proposed fast ME algorithm, we implement it in HM16.0 reference software [30]. We compare the proposed algorithm with state-of-the-art fast ME algorithms in [24] and [29], which are also implemented in HM16.0 with the same encoding parameters [31]. The simulation is performed on a PC with an Intel i7-2600 central processing unit 3.4 GHz (four cores) and 4.0 GB random access memory. The default setting of low\_delay P main with quantization parameters 22, 27, 32 and 37 is configured to obtain the Bjontegard Distortion-Rate (BD-Rate) measurement [32]. The GOP size is set as 4, and the search range is fixed to 64. We also give the experimental result for random access configuration to further evaluate the proposed algorithm, in which GOP size is 8. The experiment is performed on various test video sequences in Class B (1920×1080), Class C (832×480) and Class D (416×240). These video sequences, which cover a wide range of resolution and motion characteristics, are representative.

## B. Performance Analysis for Motion Estimation Process

First, to measure the computational complexity of ME, we examine the average number of search points (*ASP*) for one ME, which is defined as follows

 (12)

where *N* is the total times of ME and *SP(i)* means the number of search points in the *i* th ME. We test the *ASP* of our proposed fast ME algorithm and compare the experimental results with the state-of-the-art fast ME algorithms in [24, 29], which is shown in TABLE II. We also provide the *ASP* saving of [24], [29] and our proposed algorithm compared with TZS to further objectively evaluate the complexity reduction for ME.

By observing the experimental results in TABLE II, it can be seen that [24], [29] and the proposed algorithm can significantly decrease the computational complexity of ME, among which our proposed algorithm achieves the most time saving, up to 89.99% and on average 82.47% compared with TZS, and about 10% and 9% when compared with the algorithms in [24] and [29], respectively. Furthermore, it can be observed that the proposed fast ME algorithm reduces more computational complexity for a video sequence full of stationary backgrounds and smooth movement compared with the algorithms in [24] and [29]; the PUs of these video sequences are mostly classified as motion-smooth PUs, and the motion search process in the motion-smooth PUs of the proposed algorithm consumes considerably less computational complexity to obtain decent coding quality.

To evaluate the prediction accuracy of our proposed fast ME algorithm, the average prediction error (*APE*) for one ME is also measured, in which TZS is chosen as the benchmark due to its good performance in HM16.0. The *APE* is calculated as follows

 (13)

where *N* is the total times of ME and *i* indicates the *i* th ME. *MVX**OPT*(*i*) and *MVY**OPT* (*i*) are the horizontal and vertical component, respectively, of the optimal MV for the *i* th ME process employing the measured fast ME algorithm. Similarly, *MVX**TZS*(*i*) and *MVY**TZS*(*i*) denote the horizontal and vertical component, respectively, of the optimal MV implemented with TZS.

The state-of-the-art fast ME algorithms in [24] and [29] are also assessed in relation to our proposed algorithm. TABLE III shows the *APE* performance of different video sequences for [24], [29] and the proposed algorithm. It is observed that our proposed algorithm is superior to [24] and [29] with smaller prediction error. Compared with [24] and [29], our proposed algorithm achieves better coding quality due to appropriate search strategies in regions with different motion characteristics, in particular for the more efficient motion search design for motion-complex PUs.

To evaluate the performance of the fast ME mentioned above in detail, further experimental analysis is given, which also presents a comparison in terms of *ASP* and *APE* for various frames of different sequences in Fig. 11 and Fig. 12, respectively. Fig. 11 indicates that our proposed fast ME algorithm is faster than the other fast ME algorithms for each tested frame, in particular compared with TZS. There are mainly two reasons, and the first is we save lots of computational resources in motion-smooth PUs because we employ a very simple small diamond search pattern. The second reason is that the proposed PG-PET algorithm reduces a good deal of redundant cost calculation during the motion search process. According to the results in Fig. 12, since TZS is chosen as the benchmark to measure the prediction errors, the APE for TZS is always zero. The prediction accuracy is higher with a smaller *APE*, which will result in less prediction residuals and better coding performance. Our proposed algorithm achieves a better *APE* performance than the state-of-the-art fast ME algorithms in [24] and [29], with smaller prediction errors and better coding efficiency, as we develop specific search patterns for motion-medium and motion-complex PUs to avoid the local optimum and maintain coding quality.

TABLE III

Average Prediction Error for One Motion Estimation (*APE*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Sequence** | **[29]** | **[24]** | **Proposed** |
| Class B  (1920×1080) | Kimono | 1.3411 | 1.2769 | 0.6110 |
| ParkScene | 0.7421 | 0.7028 | 0.2757 |
| Cactus | 1.2492 | 1.0783 | 0.7724 |
| BasketballDrive | 2.4469 | 2.2776 | 1.8728 |
| BQTerrace | 0.7585 | 0.7604 | 0.3024 |
| Class C  (832×480) | BasketballDrill | 1.0323 | 0.9538 | 0.6994 |
| BQMall | 0.8966 | 0.8535 | 0.4985 |
| PartyScene | 0.9717 | 1.0515 | 0.7014 |
| RaceHorses | 2.2819 | 2.3411 | 1.4042 |
| Class D  (416×240) | BasketballPass | 1.1366 | 0.9538 | 0.6994 |
| BQSquare | 0.0987 | 0.0879 | 0.0237 |
| BlowingBubbles | 1.3012 | 1.2574 | 0.8235 |
| RaceHorse | 1.5546 | 1.5248 | 0.767 |
| **All** | **Average** | **1.2163** | **1.1631** | **0.7270** |

TABLE II

Average Number of Search Points for One Motion Estimation (*ASP*)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Sequence** | **TZS** | **[29]** | | **[24]** | | **Proposed** | |
| ***ASP*** | ***ASP*** | **Saving** | ***ASP*** | **Saving** | ***ASP*** | **Saving** |
| Class B  (1920×1080) | Kimono | 186.60 | 36.46 | 80.46% | 25.21 | 86.49% | 20.14 | 89.21% |
| ParkScene | 67.79 | 23.08 | 65.95% | 22.93 | 66.17% | 15.35 | 77.36% |
| Cactus | 88.49 | 22.85 | 74.18% | 21.99 | 75.15% | 13.50 | 84.74% |
| BasketballDrive | 170.08 | 26.46 | 84.44% | 20.53 | 87.93% | 18.59 | 89.07% |
| BQTerrace | 68.15 | 30.78 | 54.83% | 29.71 | 56.40% | 14.68 | 78.46% |
| Class C  (832×480) | BasketballDrill | 111.85 | 22.10 | 80.24% | 22.82 | 79.60% | 16.64 | 85.12% |
| BQMall | 74.26 | 17.92 | 75.87% | 23.99 | 67.69% | 11.33 | 84.74% |
| PartyScene | 101.12 | 26.23 | 74.06% | 27.45 | 72.85% | 17.46 | 82.73% |
| RaceHorses | 239.76 | 34.81 | 85.48% | 27.35 | 88.59% | 23.99 | 89.99% |
| Class D  (416×240) | BasketballPass | 56.18 | 17.81 | 68.30% | 19.47 | 65.34% | 9.81 | 82.54% |
| BQSquare | 32.95 | 16.32 | 50.47% | 16.87 | 48.80% | 12.29 | 62.70% |
| BlowingBubbles | 74.18 | 21.16 | 71.47% | 27.59 | 62.81% | 17.15 | 76.88% |
| RaceHorse | 204.58 | 30.28 | 85.20% | 29.96 | 85.36% | 23.23 | 88.65% |
| **All** | **Average** | --- | --- | **73.15%** | --- | **72.55%** | --- | **82.47%** |

To more intuitively demonstrate why our proposed algorithm is superior to the state-of-the-art fast ME algorithms in [24, 29], we separately evaluate the ME performance of motion-smooth PUs, motion-medium PUs and motion-complex PUs. Fig. 13 presents a more detailed analysis for motion-smooth, motion-medium and motion-complex PUs of the 34th frame in the Cactus sequence. The motion-smooth PUs mostly locate in the background; the *ASP* saving of our proposed algorithm is quite high due to the simpler search pattern, with very little prediction errors. For motion-medium and motion-complex PUs, our proposed algorithm saves more computational complexity while achieving smaller prediction errors than the algorithms in [24] and [29]. In general, the prediction errors increase when abundant and complex movements occur, which can be ignored in the case of static or smooth movement. Computational complexity reduction is more obvious in motion-smooth regions because the fast ME algorithms obtain the global optimum more easily. In conclusion, the different coding performance characteristics in different PUs demonstrate that our PU classification method and related search strategies consistently outperform the state-of-the-art algorithms.

## C. Overall Performance Evaluation

In addition to the above computational complexity analysis of ME, the total encoder evaluation including the rate-distortion performance and the total encoder time saving under low delay P and random access are listed in TABLE IV. The encoding rate-distortion performance is measured by BD-Rate [32], which is the average bit rate increase with the same coding quality (PSNR). The total encoder time saving is measured by △T, which is defined in Eq. (14) as denoting the total complexity reduction factor. *T**HM* and *T**Fast* are the total encoding time required for the original HM16.0 and the one implemented by the fast algorithm, respectively.

 (14)

According to TABLE IV, compared with the TZS in HM16.0, the proposed motion classification-based fast ME algorithm can on average reduce the total encoding time by 12.47% at the cost of a negligible bit rate increase (0.17%) under low delay P. The proposed fast ME algorithm outperforms state-of-the-art algorithms, such as [24] and [29], in terms of both video coding quality and the total time saving. It is clearly shown that the algorithm in [29] produces a large loss in coding performance since mis-determination might have a great influence on the trends in search direction, and it will lead to the final search position being determined by a local optimal point. The coding performance loss for the algorithm in [24] is derived from error dynamic search range prediction. To achieve a better balance between coding quality and encoding complexity, the proposed

fast ME algorithm allocates different amounts of computing resources to the ME process in different regions based on their respective movement characteristics. Compared with [24] and [29], it saves more computational complexity for regions with small motion intensity to reduce encoder complexity. At the same time, it costs more in computational complexity for regions with intense motion to maintain coding performance. Fig. 14 and Fig. 15 show the RD curve comparisons among [24], [29], TZS and our proposed algorithm. It can be observed that our proposed algorithm achieves better objective coding quality than [24] and [29] do under the same bitrate, while it is very close to the coding quality of TZS.



|  |  |
| --- | --- |
| (a) Rate-distortion performance. | (b) Zoomed view of the figure on the left. |

Fig. 14. Rate-distortion performance curve for the sequence Cactus



|  |  |
| --- | --- |
| (a) Rate-distortion performance. | (b) Zoomed view of the figure on the left. |

Fig. 15. Rate-distortion performance curve for the sequence RaceHorses.

It is reasonable that the proposed algorithm is faster than [24] and [29] because it performs the ME process with fewer search points. Moreover, some time-consuming modules exist in the fast ME algorithm [24] and [29]. In [29], there is additional calculation to be done ahead of schedule in order to make a decision on whether to skip the current search position quickly. In [24], five start positions for the refinement search are selected based on the coding cost, which involves too much computational complexity in sorting all the points searched in the previous search step.

The effectiveness of our proposed algorithm under random access configuration is also assessed in TABLE IV. It is apparent that our fast ME proposed algorithm outperforms state-of-the-art algorithms in [24] and [29], in terms of both video coding quality and the total time saving. However, in comparison with low delay P configuration, although it saves more total encoding complexity, more performance loss occurs. This is because the time interval of two adjacent encoded frames for random access configuration is large sometimes. At this moment, the correlation between the current coding frame and the reference frame is not high enough, which makes it easier to bring in performance loss when fast motion estimation algorithms are equiped. In addition, the performance loss of previous encoded frames will have a negative impact on the coding performance of the subsequent frames.

TABLE IV

Performance Evaluation with the State-of-the-art Fast Motion Estimation Algorithms

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Sequence** | **LDP** | | | | | | **RA** | | | | | |
| **[29]** | | **[24]** | | **Proposed** | | **[29]** | | **[24]** | | **Proposed** | |
| **BD-Rate**  **(%)** | **△T**  **(%)** | **BD-Rate**  **(%)** | **△T**  **(%)** | **BD-Rate**  **(%)** | **△T**  **(%)** | **BD-Rate**  **(%)** | **△T**  **(%)** | **BD-Rate**  **(%)** | **△T**  **(%)** | **BD-Rate**  **(%)** | **△T**  **(%)** |
| Class B  (1920×1080) | Kimono | 0.81 | 19.02 | 0.50 | 19.31 | -0.07 | 20.01 | 2.13 | 25.31 | 1.91 | 26.49 | 1.51 | 27.11 |
| ParkScene | 0.39 | 10.18 | 0.26 | 9.72 | 0.09 | 10.46 | 1.36 | 17.95 | 1.07 | 18.39 | 0.74 | 19.18 |
| Cactus | 0.57 | 10.79 | 0.35 | 10.55 | 0.04 | 12.24 | 1.14 | 18.13 | 0.87 | 19.65 | 0.59 | 21.28 |
| BasketballDrive | 1.12 | 20.28 | 0.83 | 19.92 | 0.34 | 20.77 | 2.83 | 23.75 | 2.50 | 23.62 | 2.15 | 26.66 |
| BQTerrace | 0.36 | 6.23 | 0.34 | 5.65 | 0.20 | 7.05 | 0.74 | 17.31 | 0.45 | 19.85 | 0.29 | 20.04 |
| Class C  (832×480) | BasketballDrill | 0.70 | 11.80 | 0.60 | 11.39 | 0.31 | 12.59 | 1.84 | 20.75 | 1.65 | 21.40 | 1.46 | 22.32 |
| BQMall | 0.56 | 8.98 | 0.39 | 7.72 | 0.10 | 9.65 | 1.18 | 14.99 | 0.85 | 16.92 | 0.84 | 17.50 |
| PartyScene | 0.49 | 8.45 | 0.34 | 8.01 | 0.10 | 9.86 | 0.65 | 9.45 | 0.79 | 9.63 | 0.38 | 11.12 |
| RaceHorses | 1.38 | 19.89 | 1.41 | 20.46 | 0.56 | 21.90 | 3.28 | 13.21 | 3.04 | 15.35 | 2.70 | 15.64 |
| Class D  (416×240) | BasketballPass | 0.48 | 7.87 | 0.24 | 7.11 | -0.05 | 8.36 | 1.21 | 17.86 | 0.96 | 18.58 | 0.77 | 19.29 |
| BQSquare | 0.45 | 1.89 | 0.34 | 1.95 | -0.10 | 2.80 | 0.46 | 23.99 | 0.57 | 25.85 | 0.00 | 26.83 |
| BlowingBubbles | 0.57 | 6.54 | 0.68 | 6.09 | 0.33 | 6.94 | 0.78 | 15.86 | 0.67 | 16.97 | 0.38 | 17.73 |
| RaceHorse | 1.05 | 18.62 | 1.23 | 17.86 | 0.37 | 19.37 | 3.34 | 16.72 | 3.12 | 18.43 | 2.72 | 18.59 |
| **All** | **Average** | **0.69** | **11.58** | **0.58** | **11.21** | **0.17** | **12.47** | **1.61** | **18.10** | **1.42** | **19.32** | **1.12** | **20.25** |

# VI. Conclusions

In this paper, we have studied the problem of motion characteristics-driven fast ME in HEVC. Most importantly, we have found that the motion characteristics in different video regions are quite different and the ME efforts and performance are disparate. Motivated by this observation, a motion classification-based fast ME algorithm has been developed based on the analysis and full utilization of the diverse motion characteristics. First, the proposed algorithm presented a PU classification method using the motion information of neighboring PUs and the coding cost obtained. Three categories of PUs with different motion characteristics were classified as motion-smooth PU, motion-medium PU and motion-complex PU. Second, three different search strategies were well-designed for each PU category according to their respective motion characteristics. For motion-smooth PUs, the small diamond search pattern was employed to adapt simple motion. And for motion-medium PUs, the PSSP and the MDSP were implemented to fit their local intensive motion. A scalable cross search pattern was designed for motion-complex PUs to accommodate large or complex motion. Finally, a series of priority-guided fast partial internal termination mechanisms were introduced to skip impossible search positions for motion-smooth PUs, motion-medium PUs and motion-complex PUs. Experimental results demonstrate that the proposed algorithm can significantly reduce average search points under the premise of maintaining video coding quality and outperformed the state-of-the-art fast ME algorithms in terms of both encoding complexity and coding performance.

Acknowledgment

This work was partially supported by the National Key Research and Development Plan (Grant No.2016YFC0801001), the National Natural Science Foundation of China (No. 61272502) and the State Key Program of National Natural Science Foundation of China (No. 61632001).

References

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560-576, Jul. 2003.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649-1668, Sep. 2012.

[3] F. Bossen, B. Bross, K. Suhring, and D. Flynn, “HEVC Complexity and Implementation Analysis,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685-1696, Oct. 2012.

[4] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, and Y. K. Wang, T. Wiegand, “High efficiency video coding (HEVC) text specification draft 10,” JCTVC-L1003, Joint Collaborative Team on Video Coding meeting, Geneva, 2013.

[5] I. K. Kim, K. McCann, K. Sugimoto, B. Bross and W. J. Han, “High Efficiency Video Coding (HEVC) Test Model 10 (HM 10) Encoder Description,” JCTVC-L1002, Joint Collaborative Team on Video Coding meeting, Geneva, 2013.

[6] C. G. Yan, Y. D. Zhang, J. Z. Xu, F. Dai, L. Li, Q. H. Dai, and F. Wu, “A Highly Parallel Framework for HEVC Coding Unit Partitioning Tree Decision on Many-core Processors,” *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573-576, May 2014.

[7] C. G. Yan, Y. D. Zhang, J. Z. Xu, F. Dai, J. Zhang, Q. H. Dai, and F. Wu, “Efficient Parallel Framework for HEVC Motion Estimation on Many-core Processors”,*IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 12, pp. 2077-2089, Dec. 2014.

[8] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion-compensated interframe coding for video conferencing,” in *Proc. NTC 81*, New Orleans, LA, pp. C9.6.1-9.6.5, Nov./Dec 1981

[9] X. Jing and L.-P. Chau, “An efficient three-step search algorithm for block motion estimation,”*IEEE Trans. Multimedia*, vol. 6, no. 3, pp. 435-438, Jun. 2004.

[10] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast-block matching motion estimation,”*IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[11] C. Zhu, X. Lin, and L.-P. Chau, “Hexagon-based search pattern for fast block motion estimation,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349–355, May 2002.

[12] C. H. Cheung and L. M. Po, “Novel cross-diamond-hexagonal search algorithm for fast block motion estimation,”*IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 16–22, Feb. 2005.

[13] N. Parmar and M. H Sunwoo, “Enhanced Test Zone Search Motion Estimation Algorithm for HEVC,” in *Proc. IEEE Int. SoC Design Conf.,* *2014 (ISOCC 2014)*, Nov. 3-6, 2014, pp. 260-261.

[14] J. H. Jeong, N. Parmar, and M. H. Sunwoo, “Enhanced Test Zone Search Algorithm with Rotating Pentagon Search,” in *Proc. IEEE Int. SoC Design Conf.,* *2015 (ISOCC 2015)*, Nov. 2-5, 2015, pp. 275-276.

[15] L. Zhao, Y. H. Tian, and T. J. Huang, “Background-foreground division based search for motion estimation in surveillance video coding,” in *Proc. IEEE Int. Conf. Multimedia and Expo,* *2014 (ICME 2014)*, Jul. 14-18, 2014, pp. 1-6.

[16] W. J. Zhu, W. P. Ding, J. Z. Xu, Y. H. Shi, and B. C Yin, “Hash-Based Block Matching for Screen Content Coding,”*IEEE Trans. Multimedia*, vol. 17, no. 7, pp. 935–944, Jul. 2015.

[17] S.-H. Yang, J.-Z. Jiang, and H.-J. Yang, “Fast motion estimation for HEVC with directional search,” *Electron. Lett.*, vol. 50, no. 9, pp. 673-675, Apr. 2014.

[18] L. F Gao, S. F. Dong, W. M Wang, R. G. Wang, and W. Gao, “A novel integer-pixel motion estimation algorithm based on quadratic prediction,” in *Proc. IEEE Int. Conf. Image processing,* *2015 (ICIP 2015)*, Sep. 27-30, 2015, pp. 2810-2814.

[19] Y.-H. Ko, H.-S. Kang, and S.-W. Lee, “Adaptive search range motion estimation using neighboring motion vector differences,” *IEEE Trans. Consumer Electron.*, vol. 57, no. 2, pp. 726–730, May 2011.

[20] W. Dai, O. C. Au, S. J. Li, L. Sun, and R. B. Zou, “Adaptive search range algorithm based on Cauchy distribution,” in Proc. IEEE Visual Comm. Image Process., 2012 (VCIP 2012), Nov. 27-30, 2012, pp. 1-5.

[21] L. Q Shen and Z. Y. Zhang, “Content-adaptive motion estimation algorithm for coarse-grain SVC,”*IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2582–2591, Dec. 2011.

[22] Z.-T. Liao and C.-A. Shen, “A Novel Search Window Selection Scheme for the Motion Estimation of HEVC Systems,” in *Proc. IEEE Int. SoC Design Conf.,* *2015 (ISOCC 2015)*, Nov. 2-5, 2015, pp. 267-268.

[23] Y. Ismail, J. B. McNeely, M. Shaaban, H. Mahmoud, and M. A. Bayoumi, “Fast motion estimation system using dynamic models for H.264/AVC video coding,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 1, pp. 28-42, Apr. 2011.

[24] P. Nalluri, L. N. Alves, and A. Navarro, “Complexity reduction methods for fast motion estimation in HEVC,” *Signal Process.: Image Comm.*, vol. 39, no. A, pp. 280-292, Nov. 2015.

[25] Y. H. Moon, K. S. Yoon, S.-T. Park, and I. H. Shin, “A New Fast Encoding Algorithm Based on an Efficient Motion Estimation Process for the Scalable Video Coding Standard,”*IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 477–484, Apr. 2013.

[26] W. Li and E. Salari, “Successive elimination algorithm for motion estimation,”*IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105-107, Jan. 1995.

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

[28] Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, “Global elimination algorithm and architecture design for fast block matching motion estimation,”*IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 898-907, Jun. 2004.

[29] N. Hu and E.-H. Yang, “Fast Motion Estimation based on Confidence Interval,”*IEEE Trans Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1310-1322, Feb. 2014.

[30] “HM Reference Software 16.0 [Online].” Available: http://hevc.hhi.fraunhofer.de/svn/svn\_HEVCSoftware.

[31] F. Bossen, “Common test conditions and software reference configurations,” JCTVC-I1100, Joint Collaborative Team on Video Coding meeting, Geneva, May 2012.

[32] G. Bjøntegaard, “Calculation of Average PSNR Differences Between RD-curves,” VCEG-M33, VCEG of ITU-T, Austin, Texas, USA, April 2001.

******Rui Fan** received the B.S. degree in Computer Science and Technology from Beijing University of Posts and Telecommunications in 2011. He is currently working toward the Ph.D. degree with the Computer Science and Technology, Beihang University, Beijing, China.

His research interests are in the areas of video coding, video communication and video processing.

**D:\Paper\MyPaper\CA_RA\20131128 Final-submission\TCSVT 7309_Photos of All Authors\1 Yongfei Zhang.tif****Yongfei Zhang** (M’12) received the B.S. degree in Electrical Engineering and the Ph.D. degree in Pattern Recognition and Intelligent Systems from Beihang University, Beijing, China, in 2005 and 2011 respectively.

He is now an Associate Professor with the Beijing Key Laboratory of Digital Media, School of Computer Science and Engineering, and the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China. He was a visiting research scholar in Video Processing and Networking Lab, University of Missouri, Columbia, U.S, from 2007 to 2009. His current research interests include image/video processing, compression and transmission, and video surveillance.

**D:\Paper\MyPaper\CA_RA\20131128 Final-submission\TCSVT 7309_Photos of All Authors\4 Bo Li.tif****Bo Li** received the B.S. degree in Computer Science from Chongqing University in 1986, the M.S. degree in Computer Science from Xi’an Jiaotong University in 1989, and the Ph.D. degree in Computer Science from Beihang University in 1993.

Now he is a professor of Computer Science and Engineering at Beihang University, the Director of Beijing Key Laboratory of Digital Media, and has published over 100 conference and journal papers in diversified research fields including digital video and image compression, video analysis and understanding, remote sensing image fusion and embedded digital image processor.

1. The authors are with the Beijing Key Laboratory of Digital Media, School of Computer Science and Engineering, and the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China. (Corresponding author: Yongfei Zhang, e-mail: [yfzhang@buaa.edu.cn](mailto:yfzhang@buaa.edu.cn)). [↑](#footnote-ref-1)