J Real-Time Image Proc (2019) 16:1351–1364
DOI 10.1007/s11554-016-0636-5

ORIGINAL RESEARCH PAPER

# Search range reduction for uni-prediction and bi-prediction in HEVC

**K. C. Ravi Chandra Varma[1] · M. Venkata Phani Kumar[1] · Sudipta Mahapatra[1]**

Received: 28 June 2016 / Accepted: 1 September 2016 / Published online: 3 October 2016


**Abstract** The motion estimation block of the high-efficiency video coding (HEVC) standard is highly complex. This paper proposes two algorithms, namely external search range reduction (ESR) and internal search range reduction (ISR) to reduce the motion estimation complexity of HEVC in uni-prediction and bi-prediction both in the fast search mode. The proposed ESR algorithm for uni-prediction motion estimation reduces the search range adaptively for Test Zonal Search (TZS) motion estimation with negligible loss in coding efficiency. The proposed ISR algorithm for uni-prediction, namely ISR-R, limits the search range further in the raster search stage of the TZS algorithm. Moreover, a fast bi-prediction motion estimation algorithm is proposed which includes both ESR and ISR to reduce the motion estimation complexity in bi-prediction. Our algorithms are implemented in the HM-16.6 encoder in the fast search mode. The performance of the proposed algorithms are tested individually and then by combining all the algorithms. When the algorithms are combined, the number of search points and motion estimation time are reduced by 60.58 and 59.58 %, respectively in the fast search mode with a BD-Rate of 0.193 % and BD-PSNR of −0.005 % in the LD-B main profile. The number of search points and motion estimation time are reduced by 59.55 and 57.71 %, respectively in the fast search mode with a BD-Rate of 0.265 % and BD-PSNR of −0.008 % in the RA main profile.

**Keywords** High-efficiency video coding (HEVC) · Search range reduction algorithm · Uni-prediction · Bi-prediction

✉ K. C. Ravi Chandra Varma
kcravi912@gmail.com

M. Venkata Phani Kumar
venkataphanikumarm@gmail.com

Sudipta Mahapatra
sudipta.mahapatra@gmail.com

[1] Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, Kharagpur, West Bengal 721302, India

## 1 Introduction

The high-efficiency video coding (HEVC) standard is developed by the Joint Collaborative Team on Video Coding (JCT-VC) to increase the coding efficiency [1]. HEVC reduces the bit rate by nearly 50 % compared to its previous standard, H.264/AVC [2]. On the other hand, the complexity of the encoder is greatly increased. In HEVC, the use of high-performance coding tools has increased the complexity while improving the video quality and obtaining a higher compression ratio. HEVC uses larger block sizes which enables higher compression especially for high-resolution sequences. A Coding tree unit (CTU) is the basic block of size $L \times L$, where $L$ is 64, 32 or 16. Large sized CTUs yield higher compression. A CTU is divided into Coding Units (CUs) in a quad tree structure. These CUs are further divided into Prediction Units (PUs) and Transform Units (TUs) in a tree structure. Each of the CTUs, CUs, PUs and TUs has one luma component and two chroma components and they are called as Coding Tree Blocks (CTBs), Coding Block (CBs), Prediction Block (PB) and Transform Block (TB), respectively. For deciding the block size, rate-distortion (R-D) optimization is done for all the possible combinations of CUs, PUs and TUs. Due to the recursive tree-structured partitioning of CTUs to CUs and then CUs to PUs and TUs, the complexity of motion estimation in HEVC is increased tremendously.

As it is well known, intra coding reduces spatial redundancy and inter coding reduces temporal redundancy in consecutive frames through motion estimation and motion compensation. The motion estimation complexity is increased by many folds in HEVC compared to H.264/AVC. Due to the huge increase in computational complexity of the codec, it cannot be used for low-end devices having hardware and power constraints and real-time applications. Therefore, the complexity has to be reduced to a great extent without significant degradation of video quality and increase in bit rate.

In HEVC, the complexity of intra coding can be reduced by fast intra mode decision algorithms proposed in [3–5]. In inter coding, the complexity can be reduced by fast CU mode decision algorithms and by reducing the complexity in the block-matching process of motion estimation. There are many fast inter mode decision algorithms to reduce the motion estimation complexity [6–11]. In the block-matching phase, the reduction in motion estimation complexity can be achieved by reducing the number of search points, using the low complexity distortion measure, by using partial sum of absolute difference [12] and early termination of motion estimation process.

To reduce the motion estimation complexity in the searching process of the best matching reference block, the number of search points can be reduced either by using fast search patterns or by using an adaptive search range algorithm. Fast search patterns like three-step search [13], diamond search [14], Hexagonal search [15] and gradient descent search [16] algorithms reduce the number of search points to a great extent. But, there is a severe quality degradation and increase in bit rate, especially for high-motion activity sequences. Moreover, these fast search pattern algorithms are difficult to implement in hardware. Adaptive search range algorithms, on the other hand, vary the search range depending upon the predicted motion activity of the current block using neighboring information and they are hardware friendly.

Full search provides an optimum video quality, but it incurs a heavy computational burden. In order to reduce the complexity of motion estimation, the Test zonal search (TZS) motion estimation algorithm [17] is employed in the fast search mode of the motion estimation module in the HM encoder. TZS motion estimation reduces the complexity to a great extent with a negligible loss in coding efficiency. In uni-prediction, the motion compensation is carried out using one reference frame list, whereas in bi-prediction, it uses two reference frame lists [18, 19]. The use of two reference frame lists provides better coding efficiency. In this paper, complexity reduction algorithms are proposed for fast motion estimation of HEVC in uni-prediction, bi-prediction and while using both. These models are applied in the fast search mode of the HEVC encoder. An overview of the TZS algorithm and related work on fast motion estimation are presented in Sect. 2. In Sect. 3, the proposed fast motion estimation technique is detailed. Experimental results and related discussion are described in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related work

### 2.1 Overview of TZS motion estimation

The TZS motion estimation (TZSME) algorithm consists of four different stages: motion vector prediction, initial grid search, raster search and refinement search.

#### *2.1.1 Motion vector prediction*

Initially, the best motion vector predictor is obtained out of median predictor, left predictor, up predictor and upper right predictor. The predictor having the minimum cost is chosen as the best motion vector predictor and used as the initial search point. The maximum number of search points in this stage is the number of motion vector predictors used.

#### *2.1.2 Initial grid search*

In this stage, the diamond search or square search pattern is used with the stride length varying from 1 to the maximum search range ($SR_{max}$) in powers of 2. Among all the search points, the one with the minimum cost is taken as the best search point. The maximum number of search points in this stage is

$$N_{\text{grid}} = 1 + 8 \times \lfloor (\log_2(\text{SR}_{\max}) + 1) \rfloor \tag{1}$$

#### *2.1.3 Raster search*

Raster search is a simple full search on a down-sampled version of the search window. Raster search is done with a distance of "Rasterdistance". Raster search is performed if the "Bestdistance" obtained from the previous stage is greater than "Rasterdistance". Otherwise, this stage is skipped. For a PB, if the raster search stage is not skipped, then the complexity of this stage is very high compared to the other stages. The maximum number of search points in this stage is

$$N_{\text{raster}} = \left\lceil \frac{(2 \times \text{SR}_{\max} + 1)}{\text{Rasterdistance}} \right\rceil^2 \tag{2}$$

#### 2.1.4 Refinement search

In the refinement search stage, either raster refinement or star refinement is used. The search pattern used in the initial grid search is used for refinement of the motion vector in star refinement. The maximum number of search points in this stage is not fixed. It varies differently for raster and star refinements.

For $SR_{max}$ of 64, the maximum number of search points in the grid search stage and raster search stage are 53 and 625, respectively. Therefore, the raster search stage has huge complexity compared to the other stages, especially if $SR_{max}$ is high.

The HM encoder uses an optimized version of the TZS algorithm. The motion estimation complexity of the optimized version of TZS algorithm is much lower than the TZS algorithm. It uses an advanced motion vector predictor (AMVP) and the zero predictor in the motion vector prediction stage. In the latest version of the HM encoder, the upper mode motion vector is also used. In grid search and star refinement stage, it has 16 search points for the diamond and square patterns of distance >8. The grid search stage is optimized with early grid search termination. From here onward, TZS refers to the optimized version of the TZS algorithm and TZSu refers to the optimized TZS using the upper mode motion vector in the motion vector prediction stage.

Table 1 shows that the use of the upper mode motion vector as one of its predicted motion vectors in TZS (TZSu) reduces the complexity by 48.7 % on an average in terms of the reduction in number of search points ($\Delta N$) and 44.52 % on an average in terms of motion estimation time saving ($\Delta$METS) for class B sequences. Here, both TZS and TZSu are optimized. Reduction in encoding time ($\Delta$ET) is 10.06 %. The BD-Rate and BD-PSNR are 0.167 and −0.0021, respectively. This implies that there is a significant reduction in complexity without much impact on the video quality and compression. This is due to the fact that the upper mode motion vector is highly correlated to its lower mode motion vector. In the recent literature on HEVC [20–22] full search algorithm is used for testing the performance of the adaptive search range algorithm. In [23] and [24], TZS is used for testing the performance of the adaptive search range algorithms. The full search algorithm has a huge complexity. TZS has much less complexity compared to full search and the performance is almost the same as that of full search in terms of the visual quality as well as compression. But, still the complexity of TZS is too high to make it feasible for real-time applications, especially for high-motion activity and high definition sequences. Hence, we targeted the TZS module for complexity reduction in uni-prediction.

### 2.2 Adaptive search range algorithm

Adaptive search range algorithms reduce the motion estimation complexity by reducing the search range based on statistics of the neighboring blocks or the previous frame. Distortion and motion information are used as the statistical parameters to estimate the search range. The dynamic padding window size (DPWS) motion estimation proposed in [25] is based on the sum of absolute difference (SAD) between the current block and the reference block at the predicted motion vector position. This algorithm reduces the search range, thereby reducing the complexity with a little loss of video quality. In [23], a linear adaptive search range model is proposed. The search range is varied as a function of the predicted motion vector and motion consistency. The coefficients for finding the search range vary for different prediction unit modes and coding unit modes. In [20], the motion vector difference (MVD) of each of the blocks in the previous frame is modeled using a Laplacian distribution. In [22], a Cauchy distribution-based adaptive search range (CASR) algorithm is proposed where the author has claimed that the Cauchy distribution is better than the Laplacian distribution to model the probability distribution of the MVDs. In this algorithm, the distribution of the motion vector differences in the previous frame is used to estimate the parameters in the Cauchy distribution. Based on this model, the search range is fixed for the current frame. Also, the search range is increased to some extent if the predicted motion activity is high for the current frame. In [24], the maximum of all the MVDs in the spatial, temporal and upper mode neighbors are used to predict the search range in the dynamic search range (DSR) algorithm. The predicted search range is clipped in

**Table 1** Comparison of TZSu with TZS for first 100 frames of Class B sequences in LD-P Main profile

| Sequence | BD-RateY (%) | BD-PSNR Y (%) | $\Delta N$ (%) | $\Delta$MET (%) | $\Delta$ET (%) |
|---|---|---|---|---|---|
| Kimono | 0.027 | −0.000 | 56.47 | 51.53 | 16.16 |
| ParkScene | 0.161 | −0.001 | 44.89 | 41.82 | 6.75 |
| Cactus | 0.005 | 0.001 | 46.10 | 40.60 | 7.66 |
| BQTerrace | 0.401 | −0.009 | 43.16 | 41.97 | 5.71 |
| BasketballDrive | 0.242 | −0.002 | 52.89 | 46.69 | 14.02 |
| Average | 0.167 | −0.002 | 48.70 | 44.52 | 10.06 |

between 16 and 64. This method reduces the complexity with very little effect on the bit rate and PSNR.

### 2.3 External stop search algorithm

The external stop search algorithms terminate the searching process of motion estimation when the minimum cost of the current search point is less than a threshold. The threshold is derived from the cost of the neighboring blocks. In [25, 26], the external stop search algorithm and the dynamic external stop search algorithm reduce the complexity of motion estimation by terminating the search process when the threshold condition is satisfied. This algorithm is applicable for full search motion estimation and moreover, the cost of the neighboring blocks for all the block sizes needs to be stored in order to calculate the threshold.

### 2.4 Fast bi-prediction motion estimation algorithm

The HM encoder has a search range of 64 for uni-prediction motion estimation. It has a search range of 4 for bi-prediction motion estimation. Though the search range is small for bi-prediction, with the use of the fast search algorithm in HEVC, the motion estimation complexity in bi-prediction is comparable to that of uni-prediction. Table 2 shows the comparison of motion estimation time in uni-prediction and bi-prediction. For class B sequences, the percentage of the number of search points in uni-prediction, ($\%N_{\text{uni}}$) and bi-prediction, ($\%N_{\text{bi}}$) compared to the total number of search points are 77.07 and 22.93 %, respectively. The percentage of the uni-prediction motion estimation time ($\%\text{MET}_{\text{uni}}$) and bi-prediction motion estimation time ($\%\text{MET}_{\text{bi}}$) compared to the total motion estimation time are 75.37 and 24.63 %, respectively. Therefore, it is also necessary to reduce the bi-prediction motion estimation (BME) complexity in the HM encoder. In [23], a motion analysis-based adaptive search range algorithm (MAASR) is proposed to reduce the bi-prediction motion estimation complexity. This algorithm is simple and reduces the complexity of bi-prediction motion estimation with a minor loss in coding efficiency. In [27], an SAD-based bi-prediction selection method is proposed to reduce the bi-prediction motion estimation complexity.

**Table 2** Comparison of number of search points and motion estimation time in uni-prediction and bi-prediction for Class B sequences in LD-B Main profile at QP = 32

| Sequence | $\%N_{\text{uni}}$ | $\%N_{\text{bi}}$ | $\%\text{MET}_{\text{uni}}$ | $\%\text{MET}_{\text{bi}}$ |
|---|---|---|---|---|
| Kimono | 86.28 | 13.72 | 84.86 | 15.14 |
| ParkScene | 70.78 | 29.22 | 69.06 | 30.94 |
| Cactus | 75.57 | 24.43 | 73.08 | 26.92 |
| BQTerrace | 68.02 | 31.98 | 66.16 | 33.84 |
| BasketballDrive | 84.69 | 15.31 | 83.69 | 16.31 |
| Average | 77.07 | 22.93 | 75.37 | 24.63 |

In this paper, our goal is to reduce the complexity of TZS algorithm in uni-prediction and BME algorithm in bi-prediction by limiting the search range externally and internally.

## 3 Proposed algorithm

### 3.1 External search range reduction for TZS

Our proposed external search range reduction (ESR) algorithm is intended to reduce the motion estimation complexity while maintaining the coding efficiency. In motion estimation, most of the PBs find their best matching reference PBs nearer to the center of the search window. Using the entire search range for all the PBs involves unnecessary computation of SADs. In the proposed approach, the search range is adaptively varied based on the MVDs of the spatial neighboring blocks and the SAD at the search center. The MVDs in $x$ and $y$ directions are computed as follows:

$$\begin{aligned} \text{MVD}_x &= \text{MV}_x - \text{PMV}_x \\ \text{MVD}_y &= \text{MV}_y - \text{PMV}_y \end{aligned} \tag{3}$$

where $\text{MVD}_x$ and $\text{MVD}_y$ are MVDs in $x$ and $y$ direction. $\text{MV}_x$ and $\text{MV}_y$ are motion vectors in $x$ and $y$ direction. $\text{PMV}_x$ and $\text{PMV}_y$ are predicted motion vectors in $x$ and $y$ directions.

#### *3.1.1 Search range using spatial neighbors*

The spatial neighboring blocks and the current block may belong to the same object. So, the motion of the current block and the spatial neighboring blocks are highly correlated, especially if the neighboring blocks and the current blocks share the same object. The upper block and the left block are more correlated to the current block. In order to predict the search range using spatial neighbors, the distance of MV from PMV of these two blocks are used. Additionally, the distance of MV from PMV of the upper right block is also used to predict the search range. Here, the block left to the bottom left pixel of the current block is used as the left block instead of block left to the top left pixel of the current block. This is because the block left to the bottom left pixel of the current PB is less correlated to the other neighbors used. Now, the spatial neighbors are less correlated to each other but are correlated to the current block.

$$d(x,y) = \sqrt{\text{MVD}_X^2 + \text{MVD}_Y^2} \tag{4}$$

$$\text{SR}_1 = k_1 \times \max\left(d_{\text{L}}, d_{\text{U}}, d_{\text{UR}}\right) \tag{5}$$

where $d_L$, $d_U$ and $d_{UR}$ are the distance of MV from PMV of left, upper and upper right blocks. $k_1$ is a constant.

#### 3.1.2 Search range using motion vector difference distribution of previous frame

It is claimed that the motion vector differences follow the Cauchy distribution [22]. Moreover, the motion of the current frame is highly correlated to the motion of the previous frame. The distribution of MVDs in the previous frame is used to predict the search range of the current frame. Let the predicted search range for the current frame in $x$ and $y$ directions be $C_x$ and $C_y$. $C_x$ and $C_y$ are obtained from the Cauchy distribution of the MVDs of the previous frame with the Cauchy parameter chosen as 2 [28] and the predefined probability constant given in [22] chosen as 99.2 %. In a frame, the motion is not same for all the blocks. So, the predicted search range has to be updated based on the neighboring motion information. Maximum of the neighboring blocks MVD, *maxMVD* is used to update the search range for the current block. For deriving the maxMVD, left (the block at the left of the bottom left pixel of the current block), upper and upper right neighbors are used. The search range is found as a function of the maximum of the updated search range in $x$ and $y$ directions as given in (6).

$$\begin{aligned} \mathrm{SR}_2 = \max(&C_x + k_2 \times \mathrm{maxMVD}_x, \\ &C_y + k_2 \times \mathrm{maxMVD}_y) \end{aligned} \tag{6}$$

where $k_2$ is a constant. In RA mode, upto two neighboring frames are available for each frame, and one is in the forward direction and the other in the backward direction. Each of $C_x$ and $C_y$ is obtained as the maximum of these two search ranges, respectively, found from the two nearest available neighboring frames.

#### 3.1.3 Search range using SAD at search center

The motion vector of the current block is proportional to the SAD at the PMV $(\mathrm{SAD}_{\mathrm{PMV}})$. This implies that for the PB having a large motion vector, $\mathrm{SAD}_{\mathrm{PMV}}$ is higher and for the PB having a small motion vector, $\mathrm{SAD}_{\mathrm{PMV}}$ is lower. In TZS of the HM-16.6 encoder, the grid search is done from the best search center. The best search center is the the one which has the minimum cost among the AMVP, zero and depth predictors. $\mathrm{SR}_3$, which is the search range, uses the SAD of the best search center. The search range is fixed based on the normalized $\mathrm{SAD}_{\mathrm{PMV}}$ with respect to the size of the block and is given in (7).

$$\mathrm{SR}_3 = k_3 \times \frac{\mathrm{SADPMV}}{n_1 \times n_2} \tag{7}$$

where $n_1 \times n_2$ is the block size. $k_3$ is a constant set to 1. The value of $k_3$ should be small for low resolution sequences and large for high-resolution sequences. However, in this work it is fixed for all the sequences.

Finally, the maximum of all the three predicted search ranges is chosen as the search range as given in (8).

$$\mathrm{SR} = \max(\mathrm{SR}_1, \mathrm{SR}_2, \mathrm{SR}_3) \tag{8}$$

In order to avoid the degradation in video quality, the search range (SR) is further modified as given in (9).

$$\mathrm{SR} = \max\big(\mathrm{SR}, 8 + \max\big(C_x, C_y\big)\big) \tag{9}$$

The above equation implies that the minimum value of SR is 8. The term $\max\big(C_x, C_y\big)$ specifies the motion information of the previous frame. If the previous frame has high motion, then the current frame is also expected to have high motion and the minimum value of SR is $8 + \max\big(C_x, C_y\big)$. But, this can increase the complexity if the previous frame has high-motion activity and the current PB is of low motion. So, the minimum value of SR is limited to $8 + \min\big(a, \max\big(C_x, C_y\big)\big)$ with the value of $a$ set to 8. So, the search range is computed as specified in (10).

$$\mathrm{SR} = \max\big(\mathrm{SR}, 8 + \min\big(a, \max\big(C_x, C_y\big)\big)\big) \tag{10}$$

Here, the values of $k_1$, $k_2$, $k_3$, and $a$ affect the motion estimation complexity and video quality. In order to vary the motion estimation complexity, the value of these constants should be varied accordingly to achieve a better trade off between the complexity and quality.

### 3.2 Internal search range reduction for raster search of TZS

In the HM encoder, the raster search stage is carried out only if in the grid search stage of TZS the motion vector is greater than raster distance. The raster search stage of TZS is more complex compared to the grid search and refinement search stages. The default value of raster distance in the raster search stage is 5. This implies that the search points in raster search are apart from its neighboring search points by a distance of 5 pixels horizontally, vertically and diagonally. As the raster search stage involves a huge complexity, we have proposed an internal search range reduction approach, namely, ISR-R to reduce the complexity of raster search in TZS.

The ISR-R algorithm is similar to the external stop search algorithm proposed in [25]. The external stop search algorithm is suitable for full search motion estimation and it needs to store the cost for all the block sizes as it is dependent on the cost of its neighboring blocks of the same size. ISR-R does not need neighboring cost information and is suitable for the raster search stage of TZS.

In ISR-R raster search is done in a spiral scan order. In Fig. 1, there are 4 rounds for a search range of 16. The search center of raster search is numbered as 0. The search point numbered 0 corresponds to round 0. The search points numbered as 1 corresponds to round 1 and so on. For a search range of 64, there are 13 rounds.

The ISR-R algorithm starts calculating the cost from round 1. Let, $J_{\min}(0)$ correspond to the minimum cost in round 0. Round 0 is the predicted motion vector position and its cost is calculated earlier. The minimum cost in the grid search stage of TZS is less than or equal to the minimum cost at round 0. The cost of all the search points in round 1 (numbered as 1 in Fig. 1) are calculated and the minimum cost among all the search points in this round is stored as $J_{\min}(1)$. If $r-1$ is the round having the minimum cost, and $J_{\min}(r-1)$ is greater than the minimum cost obtained in the grid search stage, then the search process continues in the next round. If $J_{\min}(r-1)$ is less than or equal to the minimum cost obtained in the grid search stage, then the search process is terminated for the raster search stage if the minimum cost of the current round is greater than the minimum cost of the previous round as given in (11). Otherwise, the search process continues in the next round.

$$J_{\min}(r-1) < J_{\min}(r) \tag{11}$$

There can be a case where the minimum cost of the previous round is less than the minimum cost of the current round, but is greater than the minimum cost in the next round. If the threshold given in (11) is used, then the raster search process terminates quickly without detecting the actual minimum cost search point. This causes false skipping of the raster search.

Figure 2 shows the error surface for a $32 \times 16$ PB in the third frame of the Class D sequence, RaceHorses in the raster search stage. The error surface over the raster search stage has costs only at the numbered search points in a search range. This PB belongs to the eleventh CTU at a pixel position of (192, 64) in the frame. As raster search is done in a spiral scan manner, the minimum cost in each round, $J_{\min}(r)$ and its corresponding best MV of round $r$ are shown in Table 3. Table 3 shows that the minimum cost, $J_{\min}(r)$ decreases with an increase in $r$ till $r = 5$ and then it increases for $r = 6$. With the condition given in (11), the raster search is terminated resulting in the minimum cost at $r = 5$ with $J_{\min}(5) = 6165$ at $(-15, -9)$. But, the actual minimum cost is at $r = 10$ with $J_{\min}(10) = 5493$ at (60,

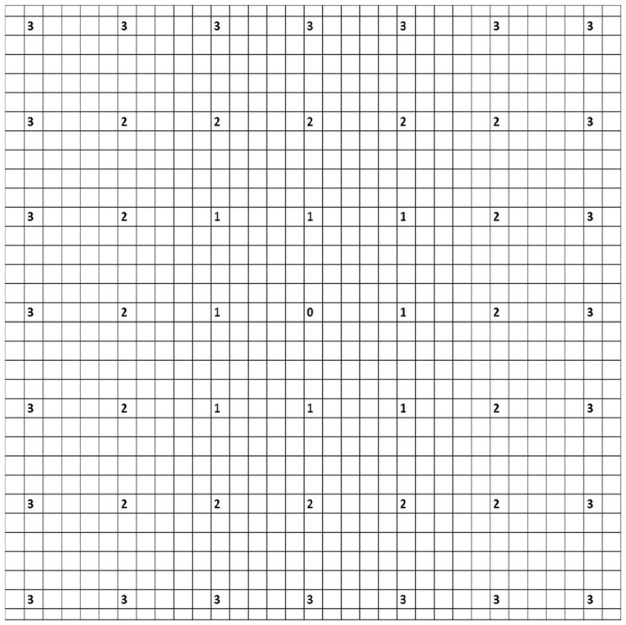


**Fig. 1** Raster search pattern in spiral scan order with raster distance of 5 for a search range of 16

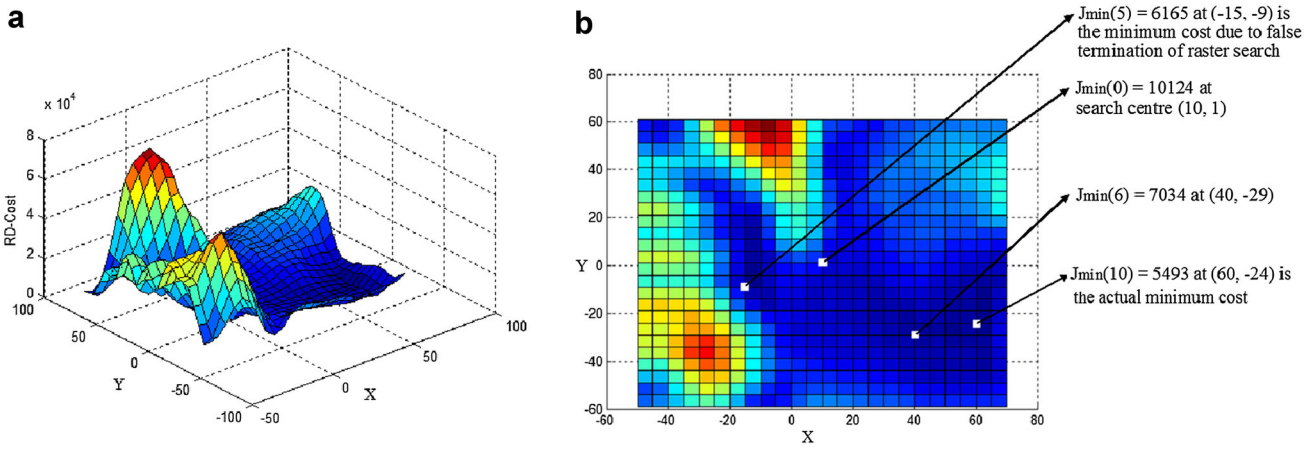


**Fig. 2** Error surface for $32 \times 16$ prediction block in the third frame of Class D RaceHorses sequence in raster search stage. **a** Error surface plot over a search range. **b** Error surface from top view with white dots showing that the actual best raster MV (60, −24) and the raster MV due to false termination of raster search (−15, −9) are far apart

**Table 3** Minimum cost of all rounds in the error surface of a 32 × 16 prediction block in the third frame of the Class D RaceHorses sequence in raster search

| $r$ | $J_{\min}(r)$ | Best MV of round $r$ |
|---|---|---|
| 0 | 10124 | (10, 1) |
| 1 | 9834 | (10, −4) |
| 2 | 9526 | (20, −9) |
| 3 | 8700 | (25, −14) |
| 4 | 8205 | (25, −19) |
| 5 | 6165 | (−15, −9) |
| 6 | 7034 | (40, −29) |
| 7 | 7153 | (40, −34) |
| 8 | 6553 | (50, −29) |
| 9 | 6915 | (55, −24) |
| 10 | 5493 | (60, −24) |
| 11 | 8336 | (65, −9) |
| 12 | 9698 | (70, −9) |

−24). Due to this, the best raster MV is (−15, −9) instead of (60, −24). This is termed as false termination of raster search. From Fig. 2b, the actual best raster MV and the raster MV due to false termination of raster search are far apart. Therefore, the minimum cost of the current round should not be compared with a threshold equal to that of the minimum cost of the previous round. The current threshold has to be reduced in order to minimize the chance of false termination of raster search. For this (11) is modified by introducing a scaling factor of $\frac{7}{8}$ as shown in (12). This scaling factor is used in order to avoid the false termination of raster search so that the reduction in quality is negligible. Also, $\frac{7}{8} \times J_{\min}(r)$ is nearly equal to $(J_{\min}(r) - (J_{\min}(r) >> 3))$.

$$J_{\min}(r-1) < \frac{7}{8} \times J_{\min}(r) \tag{12}$$

From round 2, if $J_{\min}(r-1)$ is less than or equal to the minimum cost obtained in the grid search stage, then the minimum cost of the current round is compared with the cost of rounds $r-1$ and $r-2$ as shown in (13) and (14). If both the conditions are satisfied, then raster search is terminated. Otherwise, the condition given in (12) is checked to terminate raster search. This process is repeated till all the rounds in the search range are completed or till the raster search stage is terminated.

$$J_{\min}(r-2) < J_{\min}(r-1) \tag{13}$$

$$J_{\min}(r-2) < J_{\min}(r) \tag{14}$$

The steps followed in the ISR-R algorithm are as follows:

1. Store the cost at the search center of raster search as $J_{\min}(0)$ and set $r = 1$.
2. Calculate the cost of all the search points for the current round and obtain the minimum cost $J_{\min}(r)$.
3. If $J_{\min}(r-1)$ is less than or equal to the minimum cost obtained in the grid search stage, then if $r = 1$ go to step 4; else, go to step 6. Otherwise, $r = r + 1$ and go to step 2.
4. If the condition given in (12) is satisfied, then go to step 9. Otherwise, go to step 5.
5. $r = r + 1$. Calculate the cost of all the search points in the current round and obtain the minimum cost $J_{\min}(r)$.
6. If the conditions given in (13) and (14) are satisfied, then go to step 8. Otherwise, go to step 7.
7. If the condition given in (12) is satisfied, go to step 9. Otherwise, go to step 8.
8. If the current round is the last round, then go to step 9. Otherwise, go to step 5.
9. Terminate raster search and perform refinement search.

### 3.3 Fast bi-prediction motion estimation algorithm

In the HM encoder, the motion estimation process for bi-prediction is performed over the search range of 4 around the uniprediction motion vector. The complexity of motion estimation in bi-prediction can be reduced at the following two stages:

1. External search range reduction for bi-prediction
2. Internal search range reduction for bi-prediction

#### *3.3.1 External search range reduction for bi-prediction*

Figure 3a shows the proposed external search range reduction approach based on the uni-prediction motion vector. If the maximum of the MVDs of the $x$ and $y$ components is <2, then the search range is fixed as shown in (15)

$$\begin{aligned} \mathrm{SR} &= \max(|MVDx|, |MVDy|) + 1 \\ &\quad \text{if} \quad \max(|MVDx|, |MVDy|) < 3 \\ &= 4 \qquad \text{otherwise} \end{aligned} \tag{15}$$

#### *3.3.2 Internal search range reduction for bi-prediction*

The internal search range reduction for bi-prediction is shown in Fig. 3b. After fixing the search range, motion estimation is done in a spiral scan manner instead of raster scan. From Fig. 4, it is observed that there are a total of 5 rounds in the search range with the rounds numbered from 0 to 4. In the motion estimation process, the minimum cost among all the previous search points is taken as the best cost. But, here the best cost is calculated for each round at the end of the motion estimation process of that round. After calculating the best cost for the first two rounds, if

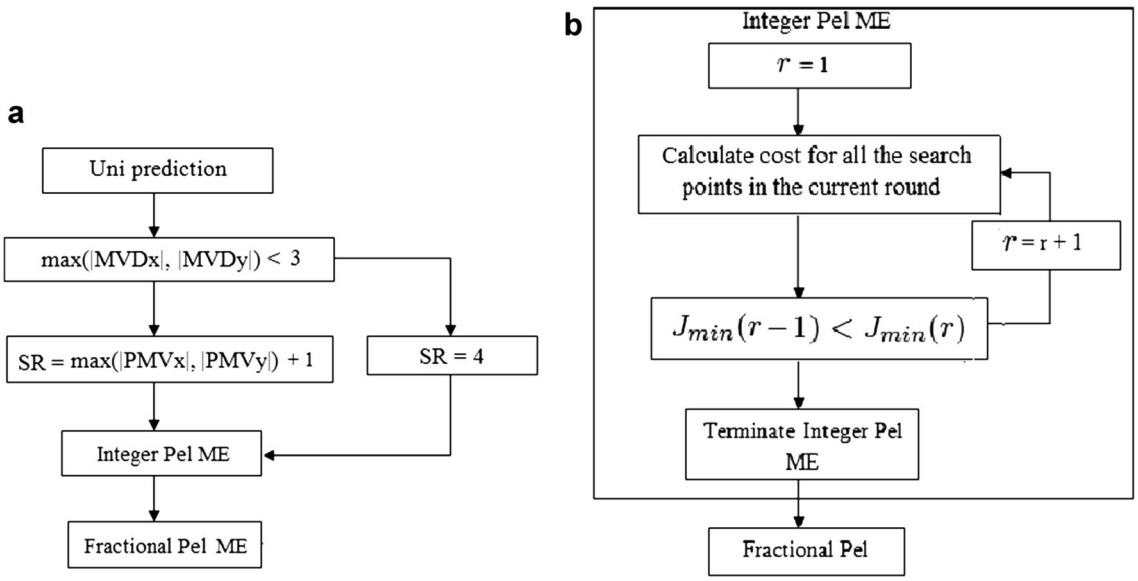


**Fig. 3** Fast bi-prediction motion estimation algorithm. **a** External search range reduction for bi-prediction. **b** Internal search range reduction for bi-prediction

the best cost at round 0 is less than the best cost at round 1, then the motion estimation process is terminated. Otherwise, the best cost of the next round is calculated and compared with the best cost of the current round. This process is continued till the internal search range reduction condition, given in (16) is satisfied or till the search points corresponding to all the rounds are used for calculating the cost.

$$J_{\min}(r-1) < J_{\min}(r). \tag{16}$$

## 4 Experimental results

The proposed algorithm is implemented in the HM-16.6 encoder [29]. All the experiments are conducted on an Intel(R) Xeon CPU having a 2.5 GHz clock and 32 GB RAM. Standard test video sequences [30] of Class B, C, D and E given in [31] are used. These sequences, respectively, have resolutions of 1920 × 1080, 832 × 480, 416 × 240 and 1280 × 720. All the test sequences have 8 bits per sample bit depth with 4:2:0 chroma subsampling. First 100 frames of each test sequence are used for encoding. The proposed algorithm is implemented in the fast search mode of the HM-16.6 encoder. LD-P main configuration is used for testing the performance of ESR and ISR-R algorithms. The LD-B main configuration is used for testing the performance of the FBME algorithm. The combined algorithms ESR + ISR-R is implemented in the LD-P main profile. The combined and ESR + ISR-R + FBME is implemented in LD-B main and RA main profiles. The number of reference frames used is 4 and the maximum search range is set to 64. Fast encoder decision is enabled. All the simulations are carried out using Microsoft Visual Studio 2013 in the release mode. Based on the common test condition defined in the HEVC standard document, all the sequences are encoded at quantization parameter values of 22, 27, 32 and 37.

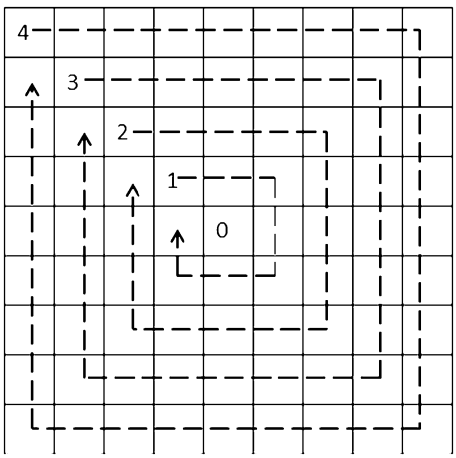


**Fig. 4** Spiral search in bi-prediction search range

Bjontegaard Delta Rate (BD-Rate) and Bjontegaard Delta PSNR (BD-PSNR) are the metrics used for calculating the coding efficiency [32]. In order to obtain the BD-Rate and BD-PSNR values of the test sequence, the PSNR values and the bit rates obtained by encoding at the above mentioned quantization parameter values are used [32].

The percentage reduction in motion estimation time, denoted by $\Delta MET$ and the percentage reduction in number of search points ($\Delta N$) of the proposed algorithm over TZS in the HM-16.6 encoder are given in (17) and (18), respectively.

**Table 4** Performance of the DSR algorithm and the proposed ESR algorithm compared to TZS in the HM 16.6 encoder in the LD-P main profile

| Class | Sequence | BD-Rate Y (%) | | BD-PSNR Y (%) | | $\Delta N_{uni}$ (%) | | $\Delta MET_{uni}$ (%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | DSR | ESR | DSR | ESR | DSR | ESR | DSR | ESR |
| Class B | Kimono | −0.0107 | 0.1435 | 0.0061 | −0.0036 | 34.91 | 68.52 | 32.29 | 68.44 |
| | ParkScene | −0.1285 | 0.2311 | 0.0002 | −0.0082 | 30.17 | 48.12 | 27.65 | 49.29 |
| | Cactus | −0.0430 | 0.2357 | 0.0017 | −0.0051 | 27.24 | 54.24 | 23.62 | 53.39 |
| | BQTerrace | −0.0768 | 0.1157 | 0.0024 | −0.0037 | 32.71 | 48.94 | 29.51 | 49.90 |
| | BasketballDrive | 0.3884 | 0.2179 | −0.0098 | −0.0064 | 23.23 | 56.95 | 20.55 | 57.23 |
| Class C | PartyScene | −0.2178 | 0.0497 | 0.0035 | −0.0006 | 30.07 | 47.97 | 26.37 | 45.22 |
| | BQMall | 0.1696 | 0.0838 | −0.0058 | −0.0068 | 32.06 | 48.40 | 29.67 | 47.82 |
| | BasketballDrill | −0.3090 | −0.3086 | 0.0147 | 0.0140 | 24.76 | 49.25 | 20.61 | 48.74 |
| Class D | RaceHorses | 0.4886 | 0.7383 | −0.0202 | −0.0309 | 36.23 | 55.86 | 32.05 | 54.46 |
| | BlowingBubbles | 0.2855 | −0.7093 | −0.0094 | 0.0260 | 36.97 | 49.92 | 33.25 | 47.87 |
| | BQSquare | −0.3379 | −0.3931 | 0.0315 | 0.0085 | 12.59 | 13.19 | 6.77 | 8.81 |
| Class E | FourPeople | −0.0734 | −0.2191 | 0.0017 | −0.0104 | 18.92 | 27.33 | 14.70 | 26.75 |
| | Johnny | 0.2548 | 0.4885 | −0.0022 | −0.0206 | 20.95 | 28.96 | 20.74 | 30.78 |
| | KristnAndSara | 0.0608 | 0.1487 | −0.0082 | −0.0180 | 25.52 | 37.00 | 24.33 | 40.04 |
| | Average | 0.0322 | 0.0588 | 0.0004 | −0.0047 | 27.59 | 45.33 | 24.44 | 44.91 |

**Table 5** Performance of ISR-R algorithm compared to TZS in the HM 16.6 encoder in the LD-P main profile

| Class | Sequence | BD-Rate Y (%) | BD-Rate YUV (%) | BD-PSNR Y (%) | BD-PSNR YUV (%) | $\Delta N_{uni}$ (%) | $\Delta MET_{uni}$ (%) |
|---|---|---|---|---|---|---|---|
| Class B | Kimono | −0.0520 | −0.1017 | 0.0079 | 0.0068 | 34.33 | 29.51 |
| | ParkScene | 0.1309 | 0.1341 | −0.0035 | −0.0033 | 19.72 | 16.16 |
| | Cactus | −0.3114 | −0.1954 | 0.0091 | 0.0070 | 28.52 | 21.56 |
| | BQTerrace | −0.0234 | −0.0254 | −0.0013 | −0.0011 | 18.83 | 14.99 |
| | BasketballDrive | −0.1033 | 0.0191 | −0.0055 | −0.0023 | 30.95 | 24.37 |
| Class C | PartyScene | −0.2425 | −0.2246 | 0.0072 | 0.0068 | 28.05 | 22.11 |
| | BQMall | 0.1003 | −0.0109 | −0.0072 | −0.0046 | 29.50 | 24.17 |
| | BasketballDrill | −0.1982 | −0.0396 | 0.0090 | 0.0076 | 31.87 | 25.84 |
| Class D | RaceHorses | 0.0266 | −0.1567 | 0.0021 | 0.0050 | 34.32 | 28.50 |
| | BlowingBubbles | −0.3479 | −0.5601 | −0.0084 | −0.0061 | 24.93 | 19.37 |
| | BQSquare | −0.3452 | −0.3423 | 0.0164 | 0.0155 | 3.86 | 1.83 |
| Class E | FourPeople | −0.2371 | −0.2546 | 0.0027 | 0.0033 | 13.51 | 10.67 |
| | Johnny | 0.2640 | 0.2954 | −0.0042 | −0.0044 | 8.97 | 7.74 |
| | KristnAndSara | 0.0519 | −0.2640 | 0.0024 | 0.0053 | 13.88 | 12.79 |
| | Average | −0.0920 | −0.1233 | 0.0019 | 0.0025 | 22.95 | 18.54 |

$$\Delta MET = \frac{MET_{HM} - MET_{Proposed}}{MET_{HM}} \times 100\,\% \tag{17}$$

$$\Delta N = \frac{N_{HM} - N_{Proposed}}{N_{HM}} \times 100\,\% \tag{18}$$

where $MET_{HM}$ and $MET_{Proposed}$ are, respectively, the motion estimation times of the TZS algorithm in HM-16.6 and the proposed algorithm. Here, $MET_{Proposed}$ includes the overhead time taken by the proposed algorithm and the motion estimation process. $N_{HM}$ and $N_{Proposed}$ are, respectively, the number of search points used for motion estimation in the TZS algorithm of HM-16.6 and the proposed algorithm. $\Delta MET_{uni}$, $\Delta MET_{bi}$ and $\Delta MET_{total}$ denote the percentage reductions in motion estimation time for uni-prediction, bi-prediction and both: $\Delta N_{uni}$, $\Delta N_{bi}$ and $\Delta N_{total}$ denote the corresponding percentage reduction in the number of search search points.

In this work, the constants $k_1$ and $k_2$ are set to 2 and 1, respectively. We have compared the performance of our proposed ESR algorithm with the DSR algorithm proposed by Nalluri et al. [24] in the fast search mode of the HM encoder. Table 4 shows the performance of the proposed ESR algorithm and the DSR algorithm compared to that of

**Table 6** Performance of the MAASR algorithm and the proposed FBME algorithm compared to BME in the HM 16.6 encoder in the LD-B main profile

| Class | Sequence | BD-RateY (%) | | BD-PSNRY (%) | | $\Delta N_{bi}$(%) | | $\Delta MET_{bi}$(%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | MAASR | FBME | MAASR | FBME | MAASR | FBME | MAASR | FBME |
| Class B | Kimono | 0.1216 | −0.1758 | −0.0013 | −0.0001 | 41.06 | 69.73 | 34.10 | 64.72 |
| | ParkScene | 0.1050 | −0.0018 | −0.0054 | −0.0032 | 66.12 | 80.73 | 59.50 | 76.72 |
| | Cactus | 0.2309 | 0.2074 | −0.0040 | −0.0045 | 71.41 | 79.89 | 66.91 | 76.26 |
| | BQTerrace | 0.6531 | 0.5144 | −0.0133 | −0.0135 | 74.18 | 82.45 | 69.81 | 78.86 |
| | BasketballDrive | 0.1228 | 0.0798 | −0.0047 | −0.0030 | 49.49 | 71.75 | 41.44 | 65.70 |
| Class C | PartyScene | 0.3315 | 0.2772 | −0.0143 | −0.0117 | 72.64 | 80.47 | 70.68 | 77.46 |
| | BQMall | 0.3869 | 0.3722 | −0.0175 | −0.0155 | 64.70 | 79.36 | 60.12 | 75.67 |
| | BasketballDrill | 0.3705 | 0.1460 | −0.0136 | −0.0130 | 67.46 | 78.61 | 63.31 | 74.88 |
| Class D | RaceHorses | 0.0404 | −0.1871 | −0.0145 | 0.0071 | 38.94 | 72.24 | 33.29 | 68.03 |
| | BlowingBubbles | 0.2101 | −0.1503 | −0.0135 | 0.0058 | 61.60 | 78.68 | 57.51 | 75.15 |
| | BQSquare | 0.2652 | −0.3085 | −0.0224 | 0.0009 | 90.43 | 86.09 | 88.70 | 83.38 |
| Class E | FourPeople | 0.5397 | 0.0536 | −0.0200 | 0.0017 | 89.85 | 86.26 | 86.52 | 83.62 |
| | Johnny | 0.1126 | 0.7782 | −0.0181 | −0.0117 | 86.13 | 85.97 | 80.94 | 83.03 |
| | KristnAndSara | 0.0504 | 0.0579 | −0.0004 | −0.0010 | 83.73 | 85.03 | 76.00 | 81.41 |
| | Average | 0.2529 | 0.1188 | −0.0116 | −0.0044 | 68.41 | 79.80 | 63.56 | 76.06 |

**Table 7** Performance of the TZSu and the proposed ESR + ISR-R algorithm compared to TZS in the HM 16.6 encoder in the LD-P main profile

| Class | Sequence | BD-RateY (%) | | BD-PSNRY (%) | | $\Delta N_{uni}$ (%) | | $\Delta MET_{uni}$ (%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | TZSu | ESR + ISR-R | TZSu | ESR + ISR-R | TZSu | ESR + ISR-R | TZSu | ESR + ISR-R |
| Class B | Kimono | 0.0270 | 0.1251 | −0.0003 | −0.0030 | 56.47 | 70.84 | 51.53 | 69.98 |
| | ParkScene | 0.1605 | −0.0261 | −0.0011 | 0.0007 | 44.89 | 50.53 | 41.82 | 51.29 |
| | Cactus | 0.0045 | 0.1739 | 0.0009 | −0.0058 | 46.10 | 58.06 | 40.60 | 57.08 |
| | BQTerrace | 0.4011 | 0.3341 | −0.0085 | −0.0078 | 43.16 | 49.51 | 41.97 | 50.31 |
| | BasketballDrive | 0.2420 | 0.3466 | −0.0015 | −0.0077 | 52.89 | 61.64 | 46.69 | 60.27 |
| Class C | PartyScene | 0.0735 | −0.0219 | −0.0026 | 0.0023 | 45.59 | 53.69 | 41.28 | 49.89 |
| | BQMall | 0.3626 | 0.2041 | −0.0089 | −0.0074 | 48.42 | 55.62 | 43.37 | 54.12 |
| | BasketballDrill | −0.3122 | 0.4458 | 0.0013 | 0.0036 | 49.45 | 56.50 | 43.90 | 53.87 |
| Class D | RaceHorses | 0.2410 | 0.2012 | −0.0091 | −0.0077 | 56.16 | 62.51 | 51.21 | 59.24 |
| | BlowingBubbles | 0.2733 | −0.0204 | −0.0111 | 0.0108 | 48.76 | 54.14 | 44.97 | 50.71 |
| | BQSquare | −0.3428 | −0.4090 | 0.0243 | 0.0113 | 20.24 | 14.68 | 20.38 | 8.72 |
| Class E | FourPeople | −0.1283 | −0.2271 | −0.0190 | 0.0004 | 22.09 | 29.37 | 22.09 | 28.45 |
| | Johnny | 0.2578 | 0.1353 | −0.0228 | −0.0029 | 23.76 | 29.57 | 25.32 | 32.06 |
| | KristnAndSara | 0.4056 | 0.4237 | −0.0207 | −0.0191 | 28.23 | 37.72 | 29.15 | 40.71 |
| | Average | 0.1190 | 0.1204 | −0.0057 | −0.0023 | 41.87 | 48.88 | 38.88 | 47.62 |

original TZS algorithm of HM-16.6 in the LD-P main profile. Our proposed algorithm has a motion estimation time of 44.91 % compared to TZS. The reduction in number of search points is 45.33 % for ESR whereas the reduction achieved in DSR is 27.59 % only. Though the reduction of motion estimation time is more, there is negligible loss in coding efficiency.

Table 5 shows the performance of the proposed ISR-R algorithm implemented in the LD-P main profile of HM-16.6. Compared to TZS, the ISR-R algorithm has reduced the number of search points and motion estimation time significantly without effecting the coding efficiency. The number of search points and motion estimation time are reduced by 22.95 and 18.54 %, respectively.

Table 6 shows the performance of the proposed FBME algorithm implemented in the LD-B main profile in the fast search mode. The efficiency of FBME is compared with the efficiency of MAASR, proposed by Du et al. [23]. The MAASR algorithm is able to reduce the number of search points and motion estimation time in bi-prediction by 68.41

**Table 8** Performance of ESR + ISR-R + TZSu algorithm compared to TZS in the HM 16.6 encoder in the LD-P main profile

| Class | Sequence | BD-Rate Y (%) | BD-Rate YUV (%) | BD-PSNR Y (%) | BD-PSNR YUV (%) | $\Delta N_{uni}$ (%) | $\Delta MET_{uni}$ (%) |
|---|---|---|---|---|---|---|---|
| Class B | Kimono | 0.2155 | 0.2088 | −0.0092 | −0.0080 | 80.20 | 78.55 |
| | ParkScene | 0.1080 | 0.1118 | 0.0002 | 0.0000 | 63.42 | 62.75 |
| | Cactus | 0.0858 | 0.1511 | −0.0016 | −0.0018 | 69.56 | 67.59 |
| | BQTerrace | 0.2116 | 0.2100 | −0.0069 | −0.0066 | 59.79 | 59.61 |
| | BasketballDrive | 0.3272 | 0.2106 | −0.0011 | −0.0015 | 75.19 | 72.23 |
| Class C | PartyScene | −0.1210 | −0.1299 | −0.0013 | −0.0012 | 68.70 | 64.75 |
| | BQMall | 0.1256 | 0.1315 | −0.0073 | −0.0075 | 71.24 | 68.64 |
| | BasketballDrill | −0.1918 | −0.1006 | −0.0012 | −0.0018 | 71.44 | 68.65 |
| Class D | RaceHorses | −0.0252 | −0.0984 | 0.0019 | 0.0041 | 78.39 | 75.41 |
| | BlowingBubbles | 0.2083 | 0.0653 | −0.0083 | −0.0057 | 68.46 | 64.81 |
| | BQSquare | −0.2677 | −0.2767 | 0.0230 | 0.0214 | 26.73 | 21.96 |
| Class E | FourPeople | −0.2112 | −0.2073 | 0.0053 | 0.0047 | 34.00 | 32.05 |
| | Johnny | 0.7294 | 0.7688 | −0.0159 | −0.0162 | 33.78 | 36.28 |
| | KristnAndSara | 0.1355 | −0.0067 | 0.0009 | 0.0024 | 42.36 | 45.19 |
| | Average | 0.0950 | 0.0742 | −0.0015 | −0.0013 | 60.23 | 58.46 |

**Table 9** Performance of ESR + ISR-R + FBME algorithms compared to TZS+BME in the HM 16.6 encoder in the LD-B main profile

| Class | Sequence | BD-Rate Y (%) | BD-PSNR Y (%) | $\Delta N_{uni}$ (%) | $\Delta MET_{uni}$ (%) | $\Delta N_{bi}$ (%) | $\Delta MET_{bi}$ (%) | $\Delta N_{total}$ (%) | $\Delta MET_{total}$ (%) |
|---|---|---|---|---|---|---|---|---|---|
| Class B | Kimono | −0.0721 | −0.0027 | 68.94 | 68.85 | 70.11 | 63.77 | 69.10 | 68.08 |
| | ParkScene | 0.1302 | −0.0041 | 52.00 | 52.99 | 80.87 | 76.28 | 60.44 | 60.20 |
| | Cactus | 0.2942 | −0.0094 | 58.98 | 58.35 | 80.25 | 76.03 | 64.17 | 63.11 |
| | BQTerrace | 0.3621 | −0.0092 | 51.86 | 53.47 | 82.52 | 78.35 | 61.67 | 61.89 |
| | BasketballDrive | 0.1757 | −0.0048 | 61.89 | 60.96 | 72.91 | 65.41 | 63.57 | 61.68 |
| Class C | PartyScene | 0.2792 | −0.0113 | 53.54 | 50.17 | 80.79 | 77.39 | 60.52 | 58.08 |
| | BQMall | 0.5692 | −0.0201 | 55.83 | 54.42 | 79.80 | 75.74 | 61.08 | 59.55 |
| | BasketballDrill | 0.5526 | −0.0098 | 55.56 | 53.45 | 79.29 | 74.56 | 60.76 | 58.60 |
| Class D | RaceHorses | −0.1353 | −0.0085 | 61.83 | 59.21 | 72.57 | 67.56 | 63.39 | 60.56 |
| | BlowingBubbles | 0.0481 | 0.0150 | 54.35 | 50.61 | 78.89 | 74.89 | 60.69 | 57.68 |
| | BQSquare | −0.3287 | 0.0136 | 14.75 | 9.13 | 86.09 | 82.99 | 51.05 | 49.38 |
| Class E | FourPeople | −0.0300 | 0.0001 | 30.52 | 30.30 | 86.29 | 83.42 | 56.25 | 56.93 |
| | Johnny | 0.4258 | −0.0097 | 31.71 | 35.05 | 86.04 | 82.90 | 56.59 | 58.17 |
| | KristnAndSara | 0.4241 | −0.0087 | 40.46 | 44.27 | 85.01 | 81.15 | 58.89 | 60.18 |
| | Average | 0.1925 | −0.0050 | 49.44 | 48.66 | 80.10 | 75.75 | 60.58 | 59.58 |

and 63.56 %, respectively. Our FBME algorithm has reduced the number of search points and motion estimation time by 79.80 and 76.06 %, respectively. Further, FBME has a smaller impact on the coding efficiency compared to that of the MAASR algorithm.

In Table 7 the performance of the combined ESR and ISR-R algorithms is compared with TZSu implemented in the HM-16.6 encoder. Both the algorithms when combined are able to reduce the number of search points and motion estimation time by 48.88 and 47.62 %, respectively. In comparison TZSu is able to reduce the number of search points and motion estimation time by 41.87 and 38.88 %. Also, from Table 8 we see that the combination of ESR, ISR-R and TZSu is able to reduce the number of search points and motion estimation time by 60.23 and 58.46 %, respectively, with negligible loss of coding efficiency.

Finally, the combination ESR, ISR-R and FBME algorithms is implemented in the LD-B main profile and RA main profile in the fast search mode. To evaluate the performance in uni-prediction and bi-prediction, the number of search points and motion estimation time are shown separately for uni-prediction, bi-prediction and both in

**Table 10** Performance of ESR + ISR-R + FBME algorithms compared to TZS+BME in the HM 16.6 encoder in the RA main profile

| Class | Sequence | BD-Rate Y (%) | BD-PSNR Y (%) | $\Delta N_{uni}$ (%) | $\Delta MET_{uni}$ (%) | $\Delta N_{bi}$ (%) | $\Delta MET_{bi}$ (%) | $\Delta N_{total}$ (%) | $\Delta MET_{total}$ (%) |
|---|---|---|---|---|---|---|---|---|---|
| Class B | Kimono | 0.2050 | −0.0111 | 65.57 | 62.03 | 74.95 | 70.05 | 66.91 | 63.32 |
| | ParkScene | 0.2143 | −0.0063 | 47.69 | 45.24 | 83.59 | 79.76 | 57.81 | 55.95 |
| | Cactus | 0.2503 | −0.0065 | 60.06 | 57.76 | 81.85 | 78.15 | 64.83 | 62.77 |
| | BQTerrace | 0.2841 | −0.0086 | 48.94 | 49.37 | 84.09 | 80.59 | 59.00 | 58.86 |
| | BasketballDrive | 0.5516 | −0.0180 | 67.83 | 64.71 | 76.03 | 70.53 | 68.93 | 65.54 |
| Class C | PartyScene | 0.0728 | −0.0109 | 61.71 | 57.35 | 82.29 | 78.96 | 66.06 | 62.61 |
| | BQMall | 0.5302 | −0.0205 | 60.49 | 57.40 | 81.46 | 77.61 | 64.83 | 62.06 |
| | BasketballDrill | 0.2434 | −0.0112 | 64.98 | 62.39 | 80.86 | 77.01 | 67.93 | 65.45 |
| Class D | RaceHorses | 0.2074 | −0.0097 | 67.74 | 63.59 | 74.71 | 70.16 | 68.70 | 64.60 |
| | BlowingBubbles | 0.2177 | −0.0072 | 56.29 | 51.18 | 81.47 | 77.98 | 62.29 | 58.50 |
| | BQSquare | 0.1707 | −0.0081 | 16.33 | 10.29 | 87.30 | 84.63 | 45.99 | 45.03 |
| Class E | FourPeople | −0.0579 | 0.0004 | 20.43 | 17.81 | 87.22 | 84.53 | 47.25 | 48.14 |
| | Johnny | 0.5698 | 0.0053 | 17.17 | 16.68 | 86.92 | 83.95 | 45.76 | 47.09 |
| | KristnAndSara | 0.2476 | −0.0031 | 22.47 | 21.81 | 86.38 | 82.99 | 47.35 | 48.02 |
| | Average | 0.2648 | −0.0083 | 48.41 | 45.54 | 82.08 | 78.35 | 59.55 | 57.71 |

Tables 9 and 10. In LD-B main profile, the combined algorithm has reduced the number of search points by 49.44, 80.10 and 60.58 % in uni-prediction, bi-prediction and both. Corresponding motion estimation times are reduced by 48.66, 75.75 and 59.58 %. BD-Rate and BD-PSNR are 0.1925 and −0.005 %. In [23], the total motion estimation time saving is less with a small loss of coding efficiency. In comparison, our proposed algorithms have more saving in the total motion estimation time with negligible loss in coding efficiency. In the RA main profile, the number of search points is reduced by 48.41, 82.08 and 59.55 % in uni-prediction, bi-prediction and both. Corresponding motion estimation times are reduced by 45.54, 78.35 and 57.71 %. BD-Rate and BD-PSNR are 0.2648 and −0.0083 %. From Tables 9 and 10, it is clear that the motion estimation complexity reduction is more for bi-prediction.

The proposed algorithms are implemented in the optimized version of the TZS algorithm. The ESR and ISR-R algorithms provide even higher saving in motion estimation complexity when implemented within the unoptimized version of the algorithm.

## 5 Conclusion

Complexity reduction tools for fast motion estimation in both uni-prediction and bi-prediction is proposed in this work. The proposed adaptive search range algorithm reduces the motion estimation time with negligible loss in video quality. ESR has reduced the motion estimation complexity with negligible loss of coding efficiency. ISR-R has reduced the complexity of motion estimation by reducing the raster search complexity of TZS. Bi-prediction fast motion estimation scheme has reduced the motion estimation complexity in bi-prediction without affecting the video quality and bit rate. The results obtained by the proposed algorithms are significant in terms of percentage reduction in number of search points and percentage reduction in motion estimation time. Experimental results illustrate that the total motion estimation time is reduced to a great extent while maintaining the coding efficiency as that of the HM-16.6 reference software in the fast search mode.

In the future, our goal is to develop an adaptive search range algorithm for uni-prediction that can provide a better trade-off between the motion estimation complexity and the compression efficiency.

## References

1. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012)
2. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
3. Hu, N., Yang, E.H.: Fast mode selection for HEVC intra-frame coding with entropy coding refinement based on a transparent composite model. IEEE Trans. Circuits Syst. Video Technol. **25**(9), 1521–1532 (2015)
4. Min, B., Cheung, R.C.: A fast CU size decision algorithm for the HEVC intra encoder. IEEE Trans. Circuits Syst. Video Technol. **25**(5), 892–896 (2015)

5. Zhang, H., Ma, Z.: Fast intra mode decision for high efficiency video coding (HEVC). IEEE Trans. Circuits Syst. Video Technol. **24**(4), 660–668 (2014)
6. Goswami, K., Kim, B.-G., Jun, D., Jung, S.-H., Choi, J.S.: Early coding unit-splitting termination algorithm for high efficiency video coding (HEVC). ETRI J. **36**, 407–417 (2014)
7. Zhong, G.-Y., He, X.-H., Qing, L.-B., Li, Y.: Fast inter-mode decision algorithm for high-efficiency video coding based on similarity of coding unit segmentation and partition mode between two temporally adjacent frames. J. Electron. Imaging **22**(2), 023025–023025 (2013)
8. Shen, L., Liu, Z., Zhang, X., Zhao, W., Zhang, Z.: An effective CU size decision method for HEVC encoders. IEEE Trans. Multimed. **15**(2), 465–470 (2013)
9. Xiong, J., Li, H., Wu, Q., Meng, F.: A fast HEVC inter CU selection method based on pyramid motion divergence. IEEE Trans. Multimed. **16**(2), 559–564 (2014)
10. Goswami, K., Lee, J.-H., Jang, K.-S., Kim, B.-G., Kwon, K.-K.: Entropy difference-based early skip detection technique for high efficiency video coding. J. Real Time Image Process. **12**(2), 237–245 (2016)
11. Lee, J.-H., Goswami, K., Kim, B.-G., Jeong, S., Choi, J.S.: Fast encoding algorithm for high-efficiency video coding (HEVC) system based on spatio-temporal correlation. J. Real Time Image Process. **12**(2), 407–418 (2016)
12. Cheung, Kwan C., Po, L.M.: Normalized partial distortion search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **10**(3), 417–422 (2000)
13. Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion compensated interframe coding for video conferencing. In: Proceedings of the National Telecommunication Conference (NTC 81), pp. C9.6.1–C9.6.5 (1981)
14. Zhu, S., Ma, K.K.: A new diamond search algorithm for fast block-matching motion estimation. In: IEEE International Conference on Image Processing (ICIP), pp. 292–296 (1997)
15. Zhu, C., Lin, X., Chau, L.P.: Hexagon-based search pattern for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **12**(5), 349–355 (2002)
16. Liu, L.-K., Feig, E.: A block-based gradient descent search algorithm for block motion estimation in video coding. IEEE Trans. Circuits Syst. Video Technol. **6**(4), 419–422 (1996)
17. Purnachand, N., Alves, L.N., Navarro, A.: Fast motion estimation algorithm for HEVC. In: IEEE International Conference on Consumer Electronics (ICCE), pp. 34–37 (2012)
18. Flierl, M., Girod, B.: Multihypothesis motion estimation for video coding. In: Data Compression Conference (DCC), pp. 341–350 (2001)
19. Flierl, M., Girod, B.: Generalized B pictures and the draft H.264/AVC video-compression standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 587–597 (2003)
20. Ko, Y.H., Kang, H.S., Lee, S.W.: Adaptive search range motion estimation using neighboring motion vector differences. IEEE Trans. Consum. Electron. **57**(2), 726–730 (2011)
21. Chien, W.D., Liao, K.Y., Yang, J.F.: Enhanced AMVP mechanism based adaptive motion search range decision algorithm for fast HEVC coding. In: IEEE International Conference on Image Processing (ICIP), pp. 3696–3699 (2014)
22. Dai, W., Au, O.C., Li, S., Sun, L., Zou, R.: Adaptive search range algorithm based on Cauchy distribution. In: IEEE Visual Communications and Image Processing (VCIP), San Diego, CA, USA, 27–30 Nov 2012, pp. 1–5 (2012). doi:10.1109/VCIP.2012.6410741
23. Du, L., Liu, Z., Ikenaga, T., Wang, D.: Linear adaptive search range model for uni-prediction and motion analysis for bi-prediction in HEVC. In: IEEE International Conference on Image Processing (ICIP), pp. 3671–3675 (2014)
24. Purnachand, N., Alves, L.N., Navarro, A.: Complexity reduction methods for fast motion estimation in HEVC. Signal Process. Image Commun. **39**, 280–292 (2015)
25. Ismail, Y., McNeely, J.B., Shaaban, M., Mahmoud, H., Bayoumi, M.A.: Fast motion estimation system using dynamic models for H.264/AVC video coding. IEEE Trans. Circuits Syst. Video Technol. **22**(1), 28–42 (2012)
26. Ismail, Y., McNeely, J.B, Shaaban, M., Bayoumi, M.A.: A generalized fast motion estimation algorithm using external and internal stop search techniques for H.264 video coding standard. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3574–3577 (2008)
27. Kim, J., Jun, D., Jeong, S., Cho, S., Choi, J.S., Kim, J., Ahn, C.: An SAD-based selective bi-prediction method for fast motion estimation in high efficiency video coding. ETRI J. **34**(5), 753–758 (2012)
28. Kamaci, N., Altunbasak, Y., Mersereau, R.M.: Frame bit allocation for the H.264/AVC video coder via Cauchy-density-based rate and distortion models. IEEE Trans. Circuits Syst. Video Technol. **15**(8), 994–1006 (2005)
29. Hevc test model. https://hevc.hhi.fraunhofer.de/trac/hevc (2012)
30. Test sequences. ftp://ftp.tnt.uni-hannover.de (2012)
31. Bossen, F., Common, H.: Test conditions and software reference configurations. JCTVC-L1100 (2013)
32. Bjontegaard, G.: Improvements of the BD-PSNR model. ITU-T SG16 Q, 6, 35 (2008)


**K. C. Ravi Chandra Varma** received B.Tech. degree in Electronics and Communication Engineering from Swami Vivekananda Institute of Technology, India in 2010 and M.Tech. degree in Electronics and Communication Engineering from NIT Calicut, India in 2012. He is currently pursuing Ph.D. degree in Electronics and Electrical Communication Engineering from IIT Kharagpur, India. His current research includes video coding algorithms and multiview video coding.

**M. Venkata Phani Kumar** is presently pursuing his Ph.D. studies in the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur. His research interests include Image Processing, Video Coding, Multimedia Communications and Multimedia QOE.

**Sudipta Mahapatra** graduated in Electronics and Telecommunication Engineering from Sambalpur University, Orissa, India in the year 1990. He obtained his M.Tech. and Ph.D. degrees in Computer Engineering from IIT Kharagpur in the years 1992 and 1997, respectively. From April 1993 to September 2002, he was working in the Computer Science and Engineering department of National Institute of Technology, Rourkela. He was in the Electronic Systems Design Group of Loughborough University, UK, as a BOYSCAST Fellow of DST, Government of India, from March 1999 to March 2000. He joined the E&ECE Department of IIT Kharagpur in Sept. 2002 where currently he is working as an Associate Professor. His areas of research interest include: Video coding/streaming, Optical and Wireless Networking, and Parallel and Distributed Systems.