Doan et al. EURASIP Journal on Advances in Signal Processing (2017) 2017:78
DOI 10.1186/s13634-017-0513-9

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

A hardware-oriented concurrent TZ search algorithm for High-Efficiency Video Coding

Nghia Doan[1], Tae Sung Kim[1], Chae Eun Rhee[2*] and Hyuk-Jae Lee[1]

Abstract

High-Efficiency Video Coding (HEVC) is the latest video coding standard, in which the compression performance is double that of its predecessor, the H.264/AVC standard, while the video quality remains unchanged. In HEVC, the test zone (TZ) search algorithm is widely used for integer motion estimation because it effectively searches the good-quality motion vector with a relatively small amount of computation. However, the complex computation structure of the TZ search algorithm makes it difficult to implement it in the hardware. This paper proposes a new integer motion estimation algorithm which is designed for hardware execution by modifying the conventional TZ search to allow parallel motion estimations of all prediction unit (PU) partitions. The algorithm consists of the three phases of zonal, raster, and refinement searches. At the beginning of each phase, the algorithm obtains the search points required by the original TZ search for all PU partitions in a coding unit (CU). Then, all redundant search points are removed prior to the estimation of the motion costs, and the best search points are then selected for all PUs. Compared to the conventional TZ search algorithm, experimental results show that the proposed algorithm significantly decreases the Bjøntegaard Delta bitrate (BD-BR) by 0.84%, and it also reduces the computational complexity by 54.54%.

Keywords: High-Efficiency Video Coding (HEVC), Integer motion estimation (IME), TZ search algorithm, Advanced motion vector prediction (AMVP)

1 Introduction

The High-Efficiency Video Coding (HEVC) [1–5] standard, the latest video coding standard, is designed to replace the previous H.264/AVC standard owing to the fact that HEVC not only preserves the video compression quality of H.264/AVC but also reduces the bitrate by as much as 50%. However, this achievement results in a substantial increase of the encoding complexity or the encoding time. In HEVC, the most complicated block is the motion estimation (ME), accounting for more than 50% of the encoding complexity. Therefore, any complexity reduction in the ME can make a significant impact on the complexity of the entire HEVC standard.

In the integer ME (IME) part of all video encoders, the use of a full search algorithm usually guarantees the best motion vectors (MVs) while also significantly increasing the encoding complexity. Hence, fast IME algorithms have been developed with the aim of greatly decreasing the required computation while also attempting to preserve the video quality. Numerous techniques [6–8] have specifically been introduced to work with block-based IME, and these are applied in different video coding standards ranging from MPEG1/H.261 to MPEG10/H.264/AVC, also known as the cross search algorithm (CSA), the three-step search (3SS), and the four-step search (4SS) algorithms. In HEVC, the TZ search algorithm is adopted in the HEVC test module (HM) software. The test zone (TZ) search algorithm is very efficient when used to obtain accurate MV results through its adaptive use of diamond and raster search algorithms. Similar to many other search algorithms, the TZ search algorithm undertakes ME for prediction units (PUs) sequentially, with each PU tracking its own search points. This is designed assuming a sequential execution in software implementation and, thus, is not proper when applied for hardware implementation owing to the fact that parallelism is not explicitly exploited.

* Correspondence: chae.rhee@inha.ac.kr
[2]Department of Information and Communication Engineering, Inha University, Incheon, Korea
Full list of author information is available at the end of the article

The proposed hardware-oriented concurrent TZ Search aims to extend the search positions of each PU partition in the same coding unit (CU), whereas the total search position sets tested in this CU remain the same. In the proposed algorithm, the search paths of the independent PU partitions are summed. To achieve this goal, the original TZ Search is modified in a manner such that it can be applied in parallel for all PU partitions, representing a completely different approach compared to the original TZ Search as well as all other fast TZ search-based IME algorithms. When using the conventional TZ search algorithm, temporal redundancy of the search positions between all PUs arises because searching is performed for each PU sequentially. In contrast, as all PUs are examined at the same time in the proposed algorithm, the temporal redundancy is converted to spatial redundancy, which is easily removed by simple comparisons. To the best of our knowledge, this paper is the first to address this problem. In addition, a search position reduction scheme is introduced for a further reduction of the complexity of the sum of the absolute difference (SAD) cost calculation. When all of the proposed schemes are applied, the complexity is reduced by as much as 54.54% along with a 0.84% improvement in the compression efficiency (i.e., bitrate reduction).

2 The conventional TZ search algorithm

The conventional TZ search algorithm consists of two main search categories: zonal search patterns (diamond and square patterns) and the raster search pattern. Figure 1a shows a flowchart of the conventional TZ search algorithm with the default configuration used in the HEVC test model (HM) software (v13.0). First, a MV predictor is used to predict the initial search center of any PU. This MV is selected from the AMVP candidate list which is composed of both spatial and temporal candidates. Among the spatial candidates derived from the left and top PUs, the temporary best predictor with the lowest motion vector (MV) cost is employed as the start point for the current PU. When the number of derived spatial candidates is less than two, a temporal candidate from the collocated frame is added to the AMVP candidate

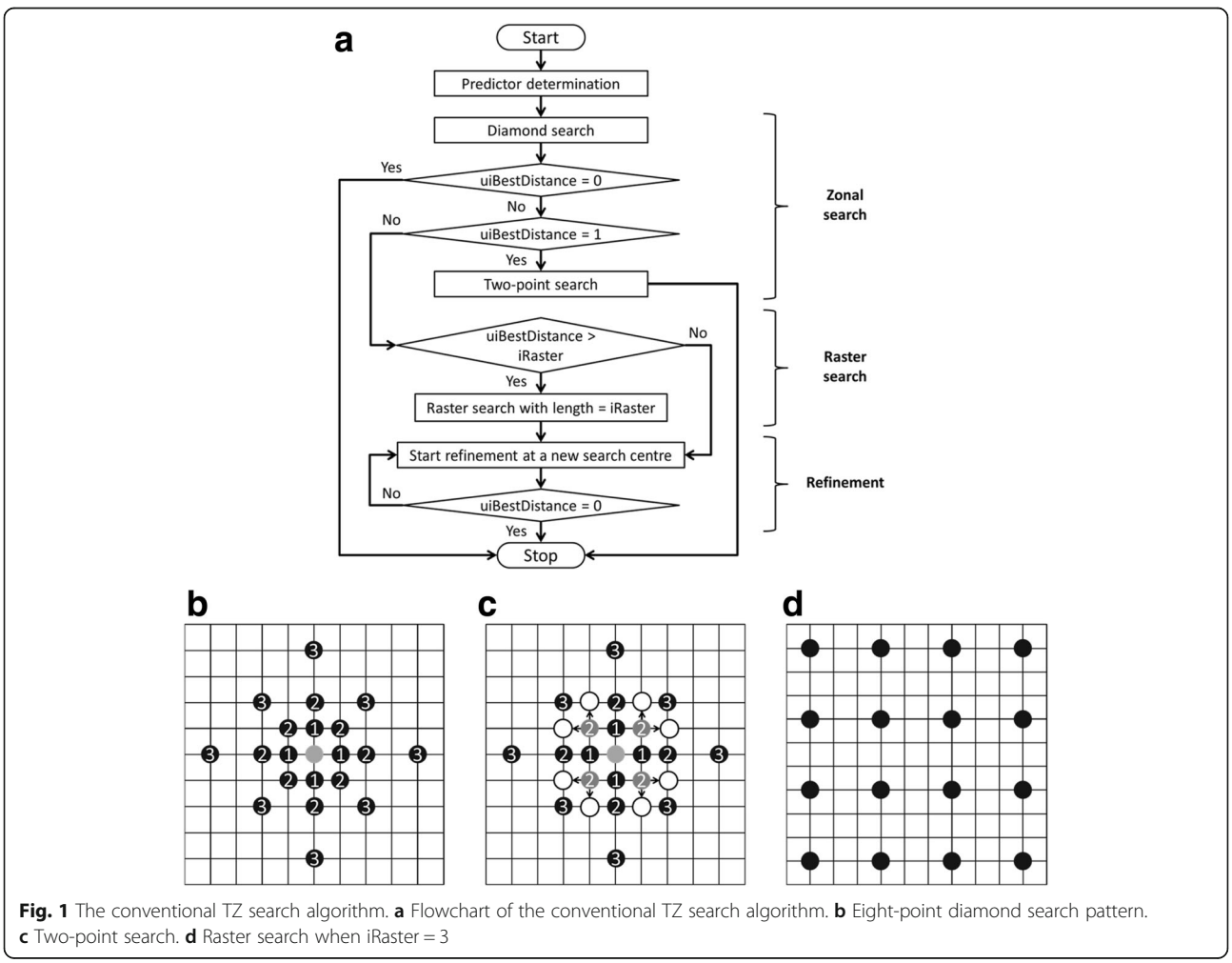

Fig. 1 The conventional TZ search algorithm. **a** Flowchart of the conventional TZ search algorithm. **b** Eight-point diamond search pattern. **c** Two-point search. **d** Raster search when iRaster = 3

list to find the best predictor. Subsequently, a zonal search is applied using the eight-point diamond search pattern with the stride length ranging from 1 to 64, where the center point of the diamond pattern is indicated by the position of the determined best predictor. Figure 1b presents an illustration of the diamond search with the stride ranging from 1 to 3. The variable "uiBestDistance" stores the distance between the best match point (the current position that gives the lowest MV cost of the current PU) and the current search center. After the first grid search, if uiBestDistance is equal to 0, there is no need to undertake additional search steps. If uiBestDistance is equal to 1, the best match can be obtained with the two-point search. Figure 1c shows all possible cases of a two-point search applied for a PU. After the first diamond search, if the best match of a PU is one of the four gray search points with a stride length of 2, two additional search points are generated around the current best match as depicted by the arrows. Otherwise, when uiBestDistance exceeds a predetermined threshold iRaster, it is necessary to perform a raster search, as the best match after the first search is located quite far from the current search center. During the raster search, a down-sampled version of the full search is applied within the predetermined search range of the current PU. Note that whenever a PU enters a raster search, the uiBestDistance value is iRaster afterward because down-sampling of the full search with the number of steps equal to iRaster is applied. Figure 1c presents an example of a raster search when iRaster is equal to 3. In addition, as shown in Fig. 1a, the raster search can be skipped if uiBestDistance is smaller than or equal to iRaster. Finally, a refinement step is performed when uiBestDistance is greater than 0. In the last search phase, an iteration of a refinement search is a combination of the diamond and the two-point search. This process is repeated until the best match position of a PU remains at the position of the current search center. In other words, whenever uiBestDistance is equal to 0, the refinement process is successfully completed.

Although the conventional TZ search algorithm can result in an improvement of more than 60% of the encoding time versus the full search algorithm, many optimization issues still exist. Attempting to solve these problems can decrease the encoding time significantly. Another problem associated with the traditional TZ search algorithm is that all search patterns used in the IME are fixed and limited in terms of the search range; therefore, the best match position can be trapped into a local minimum, which downgrades the video compression efficiency. A great amount of effort has been made to improve the original TZ search algorithm; the previous works mainly use two approaches. First, numerous modified search patterns are applied, such as pentagonal and hexagonal search patterns, as introduced in several earlier works [9–11]. The second approach determines the early termination conditions to reduce the computing time [9–15]. In one of these studies [9], due to the smaller number of search points, a hexagonal search pattern is used instead of the basic diamond pattern. Purnachand et al. noted that there is no need to continue to extend the search pattern in the first zonal search when the best distance is greater than the predetermined threshold iRaster. In a continuation of their work [10], rotating hexagonal patterns are shown to increase the peak signal-to-noise ratio (PSNR) slightly. In addition, another skipping method is presented based on the average motion cost among all previously examined search positions. Also motivated by the aforementioned study [10], in another work [12], the hexagonal and conventional diamond patterns are adaptively switched based on the MV differences (MVD) in the predicted neighboring blocks. Furthermore, a group of three search patterns [13] is selectively used for different directional movements, such as the horizontal or vertical directions, depending on the position of the best match point. Slightly different approaches have also been presented [14, 15]. A learning process was assessed in [14], where the search range of the current PU is determined by a learning algorithm following different search patterns for each particular established search range. In a related study [15], instead of finding the best match by considering all search positions within a search range, the best match point is found by solving a predictive model yielded by five fixed positions in the current search area. This prediction model is formulated from a statistical analysis derived from the MV cost distribution. It should also be noted that all previous works retain the sequential prediction order of all PU partitions in a CU. The experimental results of all related TZ search algorithms show that they all involve a trade-off between the complexity and the compression quality and that none of them can reduce the computation time or the complexity with a significant decrease in the bitrate.

3 The proposed hardware-oriented concurrent TZ search algorithm

3.1 The proposed algorithm

The processing order of all PU partitions inside a CU is depicted in Fig. 2 for both the original algorithm and the proposed algorithm. The symmetric PUs, specifically the $2N \times 2N$, $N \times 2N$, and $2N \times N$ PUs [2, 3], are processed for all CU sizes $\{8 \times 8, 16 \times 16, 32 \times 32, 64 \times 64\}$ and corresponding depths {3, 2, 1, 0}. Once the CU depth is less than 3 or the size of that CU exceeds 8×8, asymmetric PU partitions are enabled for the IME. These are $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$ PUs [2, 3]. The original TZ Search is applied sequentially for each PU partition in the order as illustrated in Fig. 2a. In contrast, the proposed algorithm processes PUs consequently as

a

```
Fast Integer Motion Estimation for a CU {
   TZSearch (2Nx2N PU);
   TZSearch (Nx2N part 0 PU);
   TZSearch (Nx2N part 1 PU);
   TZSearch (2NxN part 0 PU);
   TZSearch (2NxN part 1 PU);
   If (CU's depth <3) {
              TZSearch (2NxnU part 0 PU);
              TZSearch (2NxnU part 1 PU);
              TZSearch (2NxnD part 0 PU);
              TZSearch (2NxnD part 1 PU);
              TZSearch (nLx2N part 0 PU);
              TZSearch (nLx2N part 1 PU);
              TZSearch (nRx2N part 0 PU);
              TZSearch (nRx2N part 1 PU);
   }
}
```

b

```
Fast Integer Motion Estimation for a CU {
   If (CU's depth <3) {
       Concurrent TZSearch (
              2Nx2N PU,
              Nx2N part 0 PU , Nx2N part 1 PU ,
              2NxN part 0 PU , 2NxN part 1 PU ,
              2NxnU part 0 PU, 2NxnU part 1 PU,
              2NxnD part 0 PU, 2NxnD part 1 PU,
              nLx2N part 0 PU, nLx2N part 1 PU,
              nRx2N part 0 PU, nRx2N part 1 PU,)
   }
   Else {
       Concurrent TZSearch (
              2Nx2N PU,
              Nx2N part 0 PU, Nx2N part 1 PU,
              2NxN part 0 PU, 2NxN part 1 PU)
   }
}
```

Fig. 2 The pseudocode of the fast integer motion estimation process. **a** Original TZ search algorithm. **b** Hardware-oriented concurrent TZ search algorithm

described in Fig. 2b. In order to allocate the best match block for each PU partition, the proposed algorithm also uses the Lagrangian cost function (J_{MV}) employed in HEVC to find the IMV yielding the smallest MV cost for that PU, this cost function is re-explained hereby as shown in [10]:

$$J_{MV} = SAD(IMV) + \lambda_M R(IMV\text{–}PMV) \quad (1)$$

where SAD measures the distortion of the current PU and the reference PU, λ_M is the Lagrangian multiplier, IMV and PMV are the current integer motion vector and the predicted integer motion vector of the current PU, respectively. The parameter R(IMV–PMV) represents the rate needed to encode the motion vector difference between IMV and PMV.

Figure 3 depicts the flowchart of the proposed TZ search algorithm. As in the original TZ Search, the proposed algorithm consists of three main phases: the zonal, raster, and refinement searches. First, motion vector prediction is performed to determine the search centers of all PU partitions. This step is followed by a search position extraction based on the diamond search pattern for each individual PU before applying a process to remove all duplicated search positions. This determination of search points is made for all PUs prior to the execution of ME. Another modification of the proposed algorithm is that early termination in the diamond search is not allowed in either the zonal search or during the refinement process such that all possible search points are examined. After the first diamond search, the best positions are updated for all PUs, and the distances of those current best matches to their initial search centers are also obtained. The next step tests the necessity of two-point search based on the observation that a two-point search should be performed if this distance for any PU is equal to one. If any PU requires two-point search, when the value of uiBestDistance is equal to 1 after the first diamond search, all search points for all required PUs are determined. For those search points, the motion cost is estimated, and the best matches are again obtained for all PUs. This is the end of the zonal search based on which the necessity of raster search is tested in the next step. Some PUs must undertake a raster search if the distance between its current best match and its search center is larger than the predetermined parameter iRaster. Like the zonal search, non-redundant search points which are generated in a raster manner are obtained for all PUs that require this step prior to the execution of the MV cost calculation for any PU. After the raster search, the third search step of the concurrent TZ Search is the refinement process modified from the original TZ search algorithm. Similar to the original case, a diamond search and a two-point search are repeated until the termination condition is satisfied. In the proposed algorithm, the termination condition is modified in such a way that it is satisfied if and only if the best distances of all PU partitions are equal to 0 simultaneously, indicating that the refinement process is completely finished when all PUs have their search centers as the best matches. Note that this condition is much

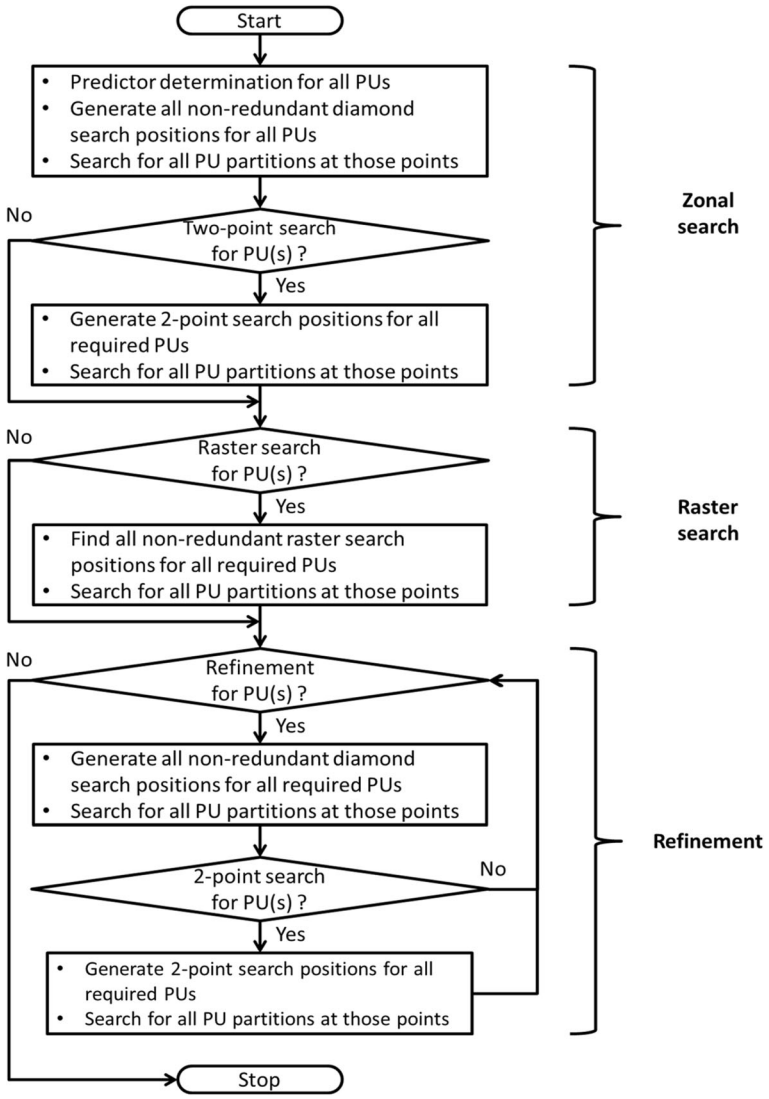

Fig. 3 The hardware-oriented concurrent TZ Search

tighter than that used in the original TZ search algorithm, which determines the termination for each PU partition independently. Furthermore, an important aspect of the stop criterion in the proposed algorithm is that a PU partition which already satisfies its termination condition in the previous iterations can update the new best position during the search operation of other PUs.

3.2 An example of the proposed algorithm

For a closer look at the proposed algorithm, an example of a concurrent TZ Search is given. In this example, only three PU partitions are considered: the 2N × 2N, N × 2N part-0, and N × 2N part-1 PUs. In addition, the stride of the diamond search is limited to 3, and iRaster is only equal to 3. As shown in Fig. 4, all PU search centers are initially obtained after the motion vector prediction step. These search center positions of the 2N × 2N, N × 2N part-0, and N × 2N part-1 PUs are represented by three gray marks in the shape of a circle, a rectangle, and a triangle, respectively. Subsequently, all search positions with the diamond pattern are generated for all PUs; the black marks in the shapes of a circle, a rectangle, and a triangle represent the search positions required for the corresponding PUs. For instance, all black circles belong to the search path of the 2N × 2N PU, the black rectangles are those for the N × 2N part-0 PU, while in the case of the N × 2N part-1 PU, its search positions are illustrated by the black triangles. When all search points are completely created for all available PUs, redundant search positions can exist owing to the fact that the search centers of all PUs are often located closely to each other. A black circle covering a generated point

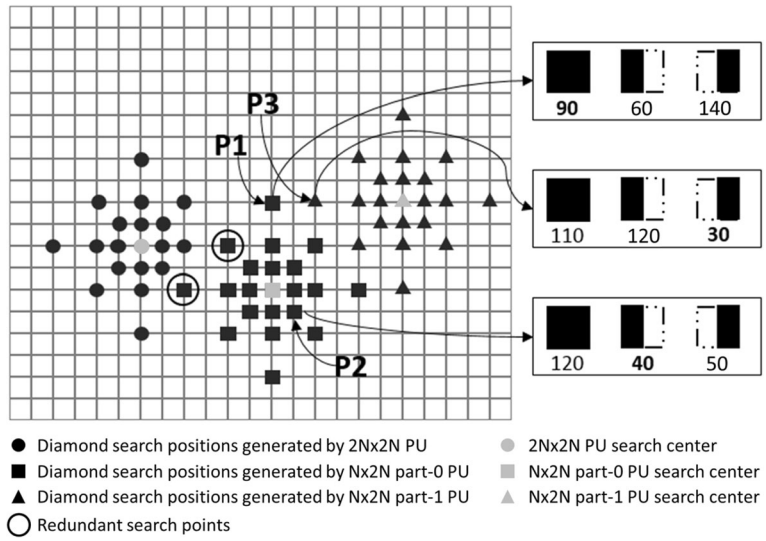

Fig. 4 The first diamond search of the concurrent TZ search algorithm

shows a redundant search position, as indicated in Fig. 4. In this example, there are two such redundant positions which belong to both the 2N × 2N and N × 2 N part-0 PUs. Those points are examined only once given that before engaging in the MV cost calculation process, a set of examination positions derived for all PUs is created which comprises only non-redundant cases. Throughout this example, P1, P2, and P3 are the current best matches of the 2N × 2N, N × 2N part-0, and N × 2N part-1 PUs, respectively. Furthermore, at each position in the aforementioned set, the SAD portion of the MV cost is only calculated for the largest 2N × 2N PU, after which this SAD value is divided into smaller parts prior to being assigned to other PU partitions, whereas the bitrate portion is calculated for each PU individually based on its current IMV and its predicted IMV. The final results of the MV cost calculation are shown on the right-hand side of Fig. 4, where only the three best search positions in the non-redundant set are depicted. For the purpose of illustration, it is assumed that the given smallest MV costs for all PU best matches after the first diamond search are 90, 40, and 30 for the 2N × 2N, N × 2N part-0, and N × 2N part-1 PUs, respectively. At each of the three demonstration search points, the rectangles colored in black and the numbers located below them illustrate the PU types and the calculated MV costs, respectively, for the corresponding PUs. At point P1, from left to right, the 2N × 2N PU has a MV cost of 90, and the MV costs for N × 2N part-0 and N × 2N part-1 are 60 and 140, respectively. Identical processes are used for the two remaining MV cost calculation examples at points P2 and P3. It should be noticed that the MV cost written in bold shows the most recent smallest value for a particular PU. Obviously, such a value is obtained when this PU partition reaches its current best match. As noted above, in the original TZ Search, the best match of a PU is only found in its search path; however, in the modified TZ search algorithm, the best match of a PU can also be found in the search paths of other PUs. This situation is also examined in the current example, where the best match of the 2N × 2N PU (P1) is located in the search path of the N × 2N part-1 PU (all search points are denoted by black rectangles), whereas for other PUs, the best match positions are obtained along their own search paths. Moreover, in the proposed algorithm, the following definition of the distance from search point P to its search center C is given: $D(P, C) = \max\{D_{hor\_dir}(P, C), D_{ver\_dir}(P, C)\}$, where $D_{hor\_dir}(P, C)$ is the distance from P to C in the horizontal direction and $D_{ver\_dir}(P, C)$ is that value in the vertical direction. According to this assumption, the distance between the 2N × 2N PU search center and its best match P1 is max {6, 2} = 6. As the distance of a search point always refers to its search center, a shorter form of D(Pi) with $i = \{1, 2, 3\}$ is used to represent the distance from the best matches to the search centers of the 2N × 2N, N × 2N part-0, and N × 2N part-1 PUs, respectively. Following this assumption, D(P1) = 6, D(P2) = max {1, 1} = 1, and D(P3) = max {4, 0} = 4. Due to the fact that the distances between the current search center and the best match of the 2N × 2N and N × 2N part-1 PUs are 6 and 4, which are all greater than iRaster = 3, 2N × 2N, and N × 2N part-1 PUs need to enter the raster search phase, whereas when the distance of the N × 2N part-0 PU is 1, then a two-point search is applied to this PU prior to entering the raster search phase.

In the two-point search of the N × 2N part-0 PU, only two additional search positions are created around the

current best position. These newly generated positions are depicted as the white rectangles in Fig. 5. At each of these two positions, as observed on the right-hand side of Fig. 5, the MV cost calculation process is performed in the same manner as in the first zonal search. It is emphasized that the opportunity to update to a better position for the best match of any PU can arise at any search point without any limitations on the initial search area to which the search point belongs. It was found that although only two additional positions are examined in the current search phase, there is still the possibility that the best matches of all three examined PU partitions can be updated when the current search phase is completed. If such a situation is taken into consideration, then P1, P2, and P3 are updated at the positions of either of the two newly created points. However, in this example, another direction is chosen to clarify the algorithm such that at the end of the two-point search, only the N × 2N part-0 PU obtains a better best match position (P2), where its current smallest MV cost is 30, smaller than its previous best match (40) and the other MV costs derived for it. On the other hand, other PUs retain their previous best match positions (P1 and P3) with MV costs of 90 and 30, respectively, because there are no smaller MV costs obtained for these PUs in this step.

Figure 6 illustrates the raster search applied for the 2N × 2N and N × 2N part-1 PUs. All search positions of each individual PU partition are created in a sub-sampled manner of the full search with iRaster equal to 3, and these points are only generated within their initial search regions bounded by the large squares, as depicted in Fig. 6. The search points for the 2N × 2N and N × 2N part-1 PUs are denoted by the small black circles and black triangles, respectively, whereas their search centers are depicted in the same shapes but are colored in gray. At the end of the raster search, there is no position containing a lower MV cost for the N × 2N part-1 PU than 30, as obtained after the first zonal search. Hence, P2 retains its position before and after the current raster search, and its current lowest MV cost is still 30. In contrast, P1 and P3 are obtained at the two new positions described in Fig. 6. The MV costs derived from these positions are 60 and 20 for the 2N × 2N and N × 2N part-1 PUs, respectively. Specifically, if the properties of the original TZ Search are considered in this specific example, the 2N × 2N PU and N × 2N part-1 PU best matches can only be found in their initial search point sets. However, the key concept of the proposed algorithm is that it extends the set of search positions for every single PU, which is an important aspect to enhance the video compression quality. In this example, the best match of the 2N × 2N PU (P1) is found in the search point sets of the N × 2N part-1 PU while that of the N × 2N part-1 PU is obtained from its own search points, denoted as P3. Taking the current distances of all PUs into account, the best match of the N × 2N part-0 PU experiences no changes during the raster search, and its best match is already formed after the previous two-point search. Thus, D(P2) is equal to 0. In addition, because the 2N × 2N and N × 2N part-1 PUs must enter the raster search phase, D(P1) = D(P3) = iRaster. In consequence, only the 2N × 2N and N × 2N part-1 PUs go through the refinement step with new search centers obtained at P1 and P3, respectively.

Figure 7 gives a demonstration of the first iteration of the refinement step. The procedure is similar to that explained in relation to the zonal search phase. First, all search positions are generated for the 2 N × 2 N and N × 2 N part-1 PUs based on the diamond pattern, and once all points are completely created, redundant points

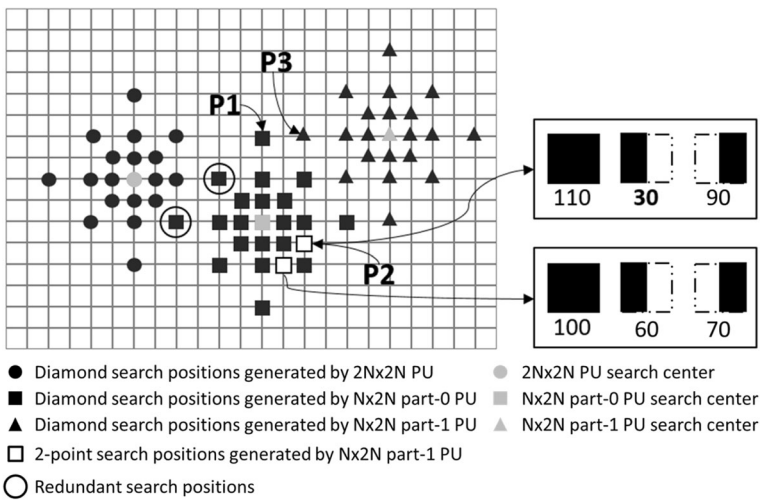

Fig. 5 The two-point search for the N × 2 N part-0 PU

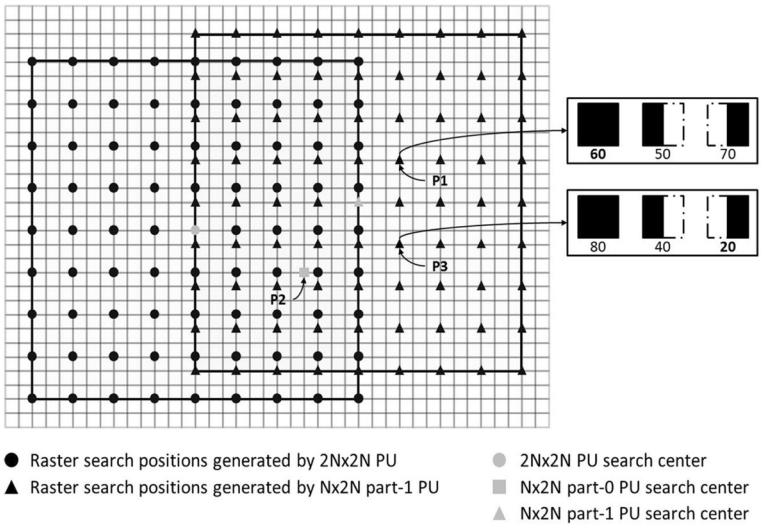

Fig. 6 Raster search for the 2 N × 2 N and N × 2 N part-1 PUs

are removed prior to the MV cost calculation. There are two unessential positions in the first refinement iteration, as denoted by the two circles shown in Fig. 7. When the MV cost calculation process is completed, P1 once again reaches its current lowest MV cost in one of the search positions initially belonging to the N × 2N part-1 PU. At that point, the minimum MV cost derived for the 2N × 2N PU is 50. In the case of the N × 2N part-0 PU, there is no better position belonging to the search paths of the 2N × 2N and N × 2N part-1 PUs that can result in a MV cost lower than 30 for the N × 2N part-0 PU. Thus, the best match position P2 retains its position. Finally, the best match of the N × 2N part-1 PU (P3) is found in its own search position set, where its MV cost is currently the lowest (10). At the end of the diamond search of the first refinement iteration, D(P1) = max {0, 5} = 5, D(P2) = 0, and D(P3) = max {0, 2} = 2. It is obvious that the 2N × 2N and N × 2N part-1 PUs must still be refined again, as the distances from their current best match points to their search centers are both greater than 0, whereas no changes are made for the case of the N × 2N part-0 PU.

The second iteration of the given example is intended for the situation when the search centers of different

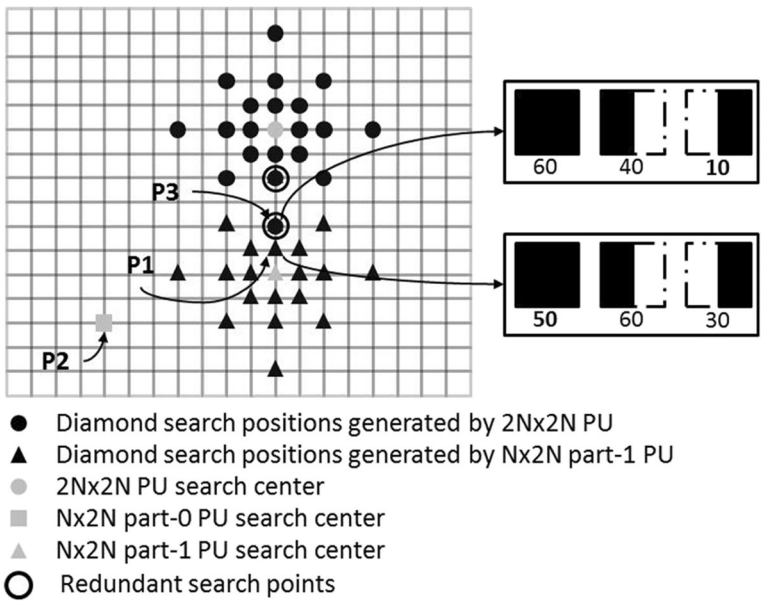

Fig. 7 The first iteration of the refinement search

PUs are located close to each other. This causes many of the positions to be redundant, as shown in Fig. 8. In this case, six positions are assumed to be unnecessary, and the MV costs for such positions are calculated only once. When all non-redundant positions are examined, the 2N × 2N and N × 2N part-1 PUs are both refined because their current search centers contain smaller MV costs as compared to those of all newly generated cases. Previously, the N × 2N part-0 PU reached its best match with the MV cost equal to 30; this position is depicted by the small gray square in Fig. 8. However, in the current iteration, a better search position of the N × 2N part-0 PU is found in the search path of the N × 2N part-1 PU with the MV cost equal to 20. Note that in the proposed stop criterion of the concurrent TZ search algorithm, even when a PU was refined in the previous refinement iterations or when this PU already contains its best match position, whenever a better search position is determined, this PU partition must be refined again. Due to the fact that this condition is acquired for all PU partitions at the same time, the refinement process ends if and only if all PUs search centers are all the best match positions of the corresponding PUs. In the current example, the N × 2N part-0 PU has to be refined in the next iteration with the new search center, denoted by P2; this procedure repeats until the aforementioned stop condition is satisfied. Note that it is also possible for the update of P1 and P3 to occur when the N × 2N part-0 PU continues its refinement process only according to the aforementioned termination conditions applied in the refinement phase.

As illustrated in Fig. 3, the very first step of the proposed algorithm is to determine all predictors that estimate the search centers of all PU partitions before the diamond search is applied. These predictors are the result of competition which involves all AMVP candidates for every PU partition. In Fig. 9, the current CU is located at the center of the nine surrounding neighbor CUs. The large shaded rectangles in dark gray show all part-0 PUs, whereas the large shaded rectangles in light gray depict all part-1 PUs. For all part-1 PUs, their AMVP candidate positions are illustrated by the small shaded squares pictured in gray. It can clearly be seen that in the case of the part-1 PU, some of its AMVP candidates are located in the part-0 PU. In the traditional IME scheme, with regard to the AMVP candidates, part-1 PUs must wait until the motion estimation process, including the IME and FME, of the part-0 PUs is complete before starting their process. This assumption increases the dependence between part-1 PUs and part-0 PUs such that all PUs in a CU cannot perform their IME processes in parallel. Many works have addressed this issue [16–18]. Generally, the main approach in these works is to modify the AMVP relationship between the two sub-PU partitions inside a CU to remove the existing dependence. To overcome this issue, unavailable AMVPs can be simply assigned to zero vectors initially. Additionally, instead of naively setting the unavailable AMVP values to zeros, these AMVP candidates of the part-1 PUs can be replaced by those that belong to the part-0 PUs, as indicated by the arrow directions in Fig. 9, which is also the same method proposed in earlier work

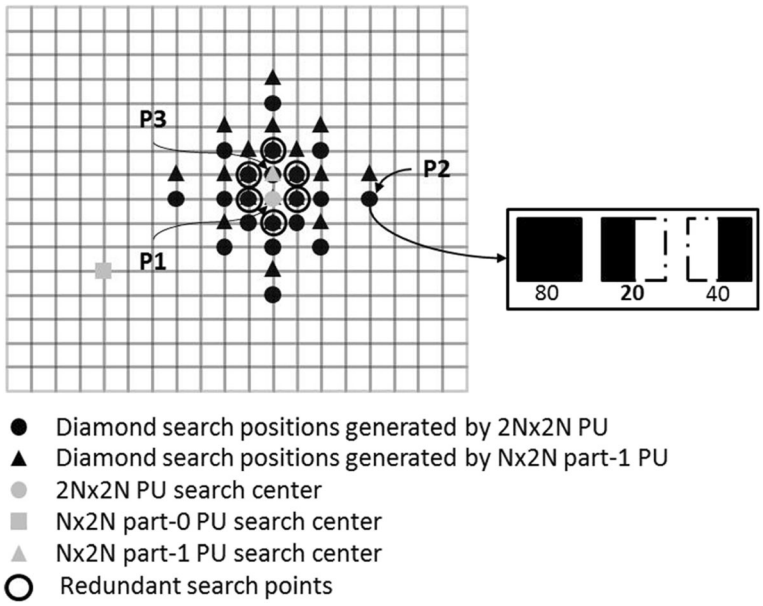

Fig. 8 The second iteration of the refinement search

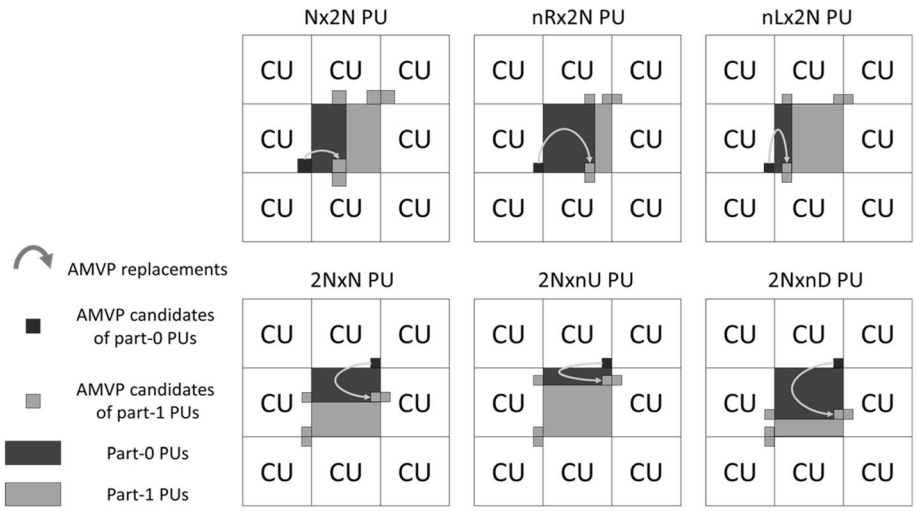

Fig. 9 Unavailable AMVP replacement scheme

[17]. The experimental results show that when the unavailable AMVP candidates of the part-1 PUs are simply replaced by zero vectors, the compression efficiency becomes considerably worse than that in the second method. Therefore, an AMVP replacement scheme realized by copying the AMVP values of the part-0 PUs to replace the unavailable values in the part-1 PUs is incorporated into the proposed algorithm.

4 Complexity reduction schemes for the proposed algorithm

4.1 Search point reduction scheme for the diamond search

The diamond search is used in the first search and in every loop iteration of the refinement search. When some PU partitions generate their search points according to the diamond search pattern, not only can the redundant search points be removed, but there is also an opportunity to eliminate more points, which are assumed to be non-critical points. In general, the search positions which are close to their search centers are often more apt to be the best match as compared to the distant positions. The method presented in this subsection assumes that all search positions whose distances to their search centers are shorter than iRaster are the critical points. Therefore, these points are only removed if and only if they are redundant. On the other hand, for all search positions whose distances to their search centers are longer or equal to iRaster, a simple search position reduction scheme is applied with the expectation that the complexity of the proposed algorithm can be decreased without a dramatic drop in the quality of the compression.

The basic idea of this scheme is that for each non-critical position, a merge window is defined at each position; this non-critical point looks inside the predetermined window and determines whether or not there are other search points. If there is at least one search point which belongs to another PU inside the generated window, the current non-critical point is removed as the MV cost of the non-critical point can then be close to that of the search points found in its window. In addition, the aforementioned window size can be fixed for all non-critical positions or can be adaptively changed regarding the distance between this point and its search center. There are two important properties of the window size. First, this window of the current point cannot cover other search positions generated in the same PU partition to preserve the compression quality. Secondly, additional search positions can have larger windows owing to the fact that additional points are less apt to become the best match compared to search points located near the search centers. Given this assumption, the reduction scheme for search positions in the diamond search uses an adaptive window size, with the window size configured to different values, as shown in Table 2 in Sect. 5.

Figure 10 gives an illustration of the search point reduction scheme used in the diamond search phase in the proposed algorithm. There are two PU partitions examined in this example, the $2N \times 2N$ PU and the $N \times 2N$ part-1 PU. All search positions of the $2N \times 2N$ PU are initially generated with iRaster equal to 3. The size of the window that takes a non-critical point of the $N \times 2N$ part-1 PU as its center is a quarter of the distance between that point and its search center. The strides for the diamond search are from 1 to 4, corresponding to distances of 1, 2, 4, and 8, respectively. As shown in Fig. 10, there is no redundant point among the two examined PUs; thus, none of the critical positions of the

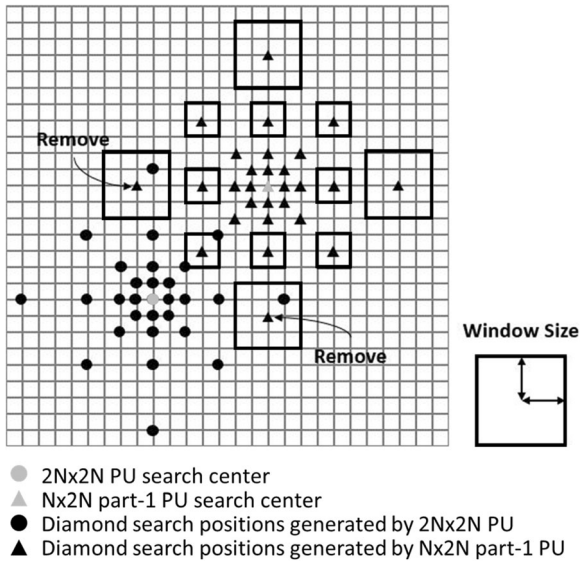

Fig. 10 Search point reduction scheme for the diamond search

N × 2N part-1 PU, those with distances to their search center smaller than iRaster = 3, are removed. Otherwise, for those positions with the distance to their search centers longer than iRaster, various windows with different sizes regarding these distances are defined at each of the points. The non-critical positions located at the fourth stride of the N × 2N part-1 PU with the corresponding distance equal to 8 have the largest windows (2 × 2), whereas windows 1 × 1 in size are defined for the non-critical points located at the third stride of the same PU partition. In the example depicted in Fig. 10, there are only two positions that are additionally removed by the proposed scheme, as denoted by the arrows, owing to the fact that two positions located along the search path of the 2N × 2N PU are found inside the merge windows. The others remain the same, as their distances are not long enough for the application of the search point reduction scheme (critical points) or because no points are found inside their windows.

4.2 Search point reduction scheme for the raster search

When multiple PU partitions perform a raster search, because the AMVPs of those PU partitions contain fairly similar values, overlapping areas often exist among the raster search regions. It should be noted that because the distance between two adjacent positions in identical PUs in the raster search pattern is equal to iRaster, the distance between two adjacent search points that belong to different PUs is definitely shorter or equal to iRaster. Furthermore, by default, the value of iRaster is set to 5 in the HM software, meaning that in the overlapping areas, the distance between the raster search points in different PU partitions is considerably small. Therefore, these positions can be merged in order to decrease the MV cost calculation complexity.

The search point reduction scheme associated with a raster search consists of four main steps, as shown in Fig. 11. First, assuming that there are n PU partitions that enter the raster search phase, in the worst case, the maximum number of these PUs is 13. Secondly, a declaration of the non-redundant raster search point set P, which is initially set to be empty, is made in step 2. In step 3, each raster region is considered sequentially such that for every point p generated in the current region, this point is searched to determine whether it is stored in the previously examined raster regions or not. If this point is also located in such regions, it means that p is already stored in set P; thus, there is no need to store p into P again. Otherwise, P does not contain p up to this step; hence, P is summed with p. The search position reduction scheme is terminated at the point the last raster region R_n is examined. In addition, despite the fact that the scheme introduced in this subsection requires sequential computations for each raster region, this process can nonetheless be pipelined with the MV cost calculation. Thus, it cannot increase the computation time for the proposed concurrent TZ search algorithm. When this scheme is applied, the coding efficiency is decreased negligibly, whereas the complexity calculated drops by 32.62%, as shown in the experimental results.

1. Given total n raster search areas: R_1, R_2, …, R_n

2. P = ∅ , P is the set of non-redundant raster search positions

3. For each R_i area ($1 \le i \le n$)
- R_1: for all search point p generated in R_2 **(1)**
  - Store all search points in this area to **P**
- R_2: for all search point p generated in R_2 **(2)**
  - If p∈ {R_1} → p is stored in step **(1)** → p is redundant, ignore p
  - Else store p into raster search point **P**
- R_3: for all search point p generated in R_3 **(3)**
  - If p∈ {R_1||R_2} → p is stored in step **(1)** or **(2)**→ p is redundant, ignore p
  - Else store p into raster search point **P**
- …
- R_n: for all search point p generated in R_n **(n)**
  - If p ∈ {R_1||R_2||…||R_{n-1}} → p is stored in step **(1)** or **(2)** or … **(n-1)** → p is redundant, ignore p
  - Else store p into raster search point **P**

4. Perform raster search for all points in set P

Fig. 11 Search point reduction scheme in the raster search phase

5 Experimental results

5.1 Complexity measurement

In HEVC hardware implementation, SAD calculation becomes a bottleneck due to the large size of a CTU (64 × 64) and various PU partitions—due to the introduction of asymmetric motion partitioning (AMP). Several works have addressed the issue of SAD calculation [19–21], and many new SAD architectures and implementations are introduced to utilize the amount of data reuse for SAD costs. Rehman et al. [19] introduced a SAD architecture using a partial product reduction scheme to add the SAD values for 4 × 4 pixel blocks. In [20], Nalluri et al. proposed two SAD architectures that can parallelize the SAD calculation process for 4 × 4 and 8 × 8 pixel blocks, and his architectures can support AMP block partitions up to the largest CTU (64 × 64). In [21], a low-power SAD tree is introduced thanks to the available of the PU-level chip selection, which only enables required pixel blocks for SAD calculation in certain levels of the SAD tree. By concerning the importance of SAD calculation in real hardware implementation, a SAD-based complexity measurement is introduced in this paper in the need of evaluating the proposed algorithm and other works. In the MV cost calculation process as described in Eq. (1), the most time-consuming task is the SAD calculation, which involves the calculation of the sum of the differences of each pixel between the current PU and the reference PU, while the calculation for bitrate cost is assumed to be negligible. Therefore, the complexity spent for MV cost calculation is represented by the complexity spent for SAD calculation. Note that every PU size can be divided into multiple 4 × 4 blocks; hence, the complexity required for the SAD calculation of a 4 × 4 block is defined as one unit of complexity. For instance, the SAD calculation for an 8 × 4 PU requires two units of complexity, whereas the complexity of a 64 × 64 PU is 256 units. Owing to the fact that each PU partition is predicted separately in the original TZ search algorithm, the complexity needed for a CU is the sum of all sub-PUs; therefore, the total complexity required for the CU is calculated as follows:

$$\mathrm{CU}_{\mathrm{OrigTZSCal}} = \sum_{\mathrm{AllPUs}} \mathrm{PU}_{\mathrm{Cal}} = \sum_{\mathrm{AllPUs}} \mathrm{PU}_{\mathrm{CalUnit}} \times N_{\mathrm{PU}} \tag{2}$$

where $\mathrm{CU}_{\mathrm{OriTZSCal}}$ is the complexity of the SAD calculation for a CU in the original TZ search algorithm, $\mathrm{PU}_{\mathrm{Cal}}$ is the complexity of the SAD cost calculation for a particular PU, $\mathrm{PU}_{\mathrm{CalUnit}}$ is the number of units of complexity, and N_{PU} is the number of search positions generated for this PU, respectively.

In the concurrent TZ search algorithm, it is assumed that the complexity of the SAD assignment from the largest 2 N × 2 N PU to smaller ones is negligible; thus, the complexity of the proposed algorithm only depends on the number of search positions and the size of the largest 2 N × 2 N PU:

$$\mathrm{CU}_{\mathrm{ConTZSCal}} = \mathrm{PU}_{2N\times\mathrm{NCalUnit}} \times N_{2N\times 2N} \tag{3}$$

where $\mathrm{CU}_{\mathrm{ConTZSCal}}$ is the complexity of the SAD cost calculation for a CU in the proposed concurrent TZ search algorithm, $\mathrm{PU}_{2N\times 2\mathrm{NCalUnit}}$ is the number of units of complexity for a 2N × 2N PU, and $N_{2N\times 2N}$ is the number of search positions generated for that 2N × 2N PU.

The example below shows how the complexity is measured for an IME of an 8 × 8 CU in the conventional algorithm and in the proposed method. It is assumed that the number of search positions acquired for the 2N × 2N, N × 2N part-0/1, and 2N × N part-0/1 PUs are all N.

$$\begin{aligned}\mathrm{CU}_{\mathrm{OrigTZSCal}} &= N \times \textstyle\sum_{\mathrm{AllPUs}} \mathrm{PU}_{\mathrm{CalUnit}} \\ &= N(\mathrm{PU}_{2N\times 2\mathrm{NCalUnit}} \\ &\quad + \mathrm{PU}_{N\times 2\mathrm{Npart0CalUnit}} \\ &\quad + \mathrm{PU}_{N\times\mathrm{Npart1CalUnit}} \\ &\quad + \mathrm{PU}_{2N\times\mathrm{Npart0CalUnit}} \\ &\quad + \mathrm{PU}_{2N\times\mathrm{Npart1CalUnit}})\end{aligned} \tag{4}$$

where

$$\begin{aligned}\mathrm{PU}_{2N\times 2\mathrm{NCalUnit}} &= \frac{\mathrm{Size}(\mathrm{PU}_{2N\times 2N}, \mathrm{Hor}) \times \mathrm{Size}(\mathrm{PU}_{2N\times 2N}, \mathrm{Ver})}{4\times 4} \\ &= \frac{8\times 8}{4\times 4} = 4(\mathrm{units})\end{aligned} \tag{5}$$

$$\begin{aligned}\mathrm{PU}_{N\times 2\mathrm{Npart0CalUnit}} &= \mathrm{PU}_{N\times 2\mathrm{Npart1CalUnit}} \\ &= \mathrm{PU}_{2N\times\mathrm{Npart0CalUnit}} \\ &= \mathrm{PU}_{2N\times\mathrm{Npart1CalUnit}} \\ &= \tfrac{\mathrm{Size}(\mathrm{PU}_{N\times 2\mathrm{Npart0}}, \mathrm{Hor}) \times \mathrm{Size}(\mathrm{PU}_{N\times 2\mathrm{Npart1}}, \mathrm{Ver})}{4\times 4} \\ &= \tfrac{4\times 8}{4\times 4} = 2(\mathrm{units})\end{aligned} \tag{6}$$

By substituting Eqs. (5) and (6) to Eq. (4), the total complexity of the current 8 × 8 CU in the case of the original TZ search algorithm is determined, as follows:

$$\mathrm{CU}_{\mathrm{OrigTZCal}} = 12N(\mathrm{units}) \tag{7}$$

The experimental results show that on average, the number of actual search positions used for a CU of the concurrent TZ search algorithm only accounts for 52% of all search positions used in the same CU of the conventional TZ search algorithm. Given this result, the total complexity of the SAD calculation for

an 8×8 PU in the concurrent TZ search algorithm is determined as follows:

$$\begin{aligned} \mathrm{CU}_{\mathrm{ConTZSCal}} &= N_{2N\times 2N} \times \mathrm{PU}_{2N\times 2\mathrm{NCalUnit}} \\ &= (5N \times 52\%) \times \mathrm{PU}_{2N\times 2\mathrm{NCalUnit}} \\ &= 10.4N(\mathrm{units}) \end{aligned} \tag{8}$$

In the example above, on average, the complexity of an 8×8 CU needed for the SAD calculation in the case of the concurrent TZ Search accounts for roughly 84% of the complexity when the same size CU undergoes fast integer motion estimation by the original TZ search algorithm.

5.2 Evaluation

The proposed concurrent TZ search algorithm is implemented into the HEVC reference software HM version 13.0, and the experimental results are compared with the original encoder in terms of the Bjøntegaard-Delta bitrate (BD-BR) and complexity, as noted in Sect. 4.1. In addition, video sequences from class A to E are examined with the low delay P configuration taken at 100 frames with various quantization parameters varying among 22, 27, 32, and 37.

Table 1 shows the experimental results of the video test sequences from class A to E of the proposed TZ search algorithm. The first column illustrates the test sequences with their video resolutions. From the second to the fourth columns, the BD-BR values of the three color components Y, U, and V are calculated, whereas the fifth column shows the weighted BD-BR, which is used to evaluate the compression efficiency of the proposed algorithm and earlier works.

$$\mathrm{wBDBR} = \frac{6 \times \mathrm{BDBR_Y} + \mathrm{BDBR_U} + \mathrm{BDBR_V}}{8} \tag{9}$$

where $\mathrm{BDBR_Y}$, $\mathrm{BDBR_U}$, and $\mathrm{BDBR_V}$ are the Bjøntegaard-Delta bitrate values for the three video signal components Y, U, and V, respectively. As the Y component is the most important among the three signals, it has the largest gain which is equal to 6, and the other components only have the gain values equal to 1. Those weight values (6:1:1) for Y, U, and V signals are adopted from [5] during the HEVC standardization process. The last column gives the values representing the amount of SAD calculation complexity reduction denoted by $\mathrm{SAD_{cal}}$ compared to that in the original algorithm. The bottom row of Table 1 shows the average values of the BD-BR and the complexity reduction of all test video sequences. As shown in Table 1, the proposed algorithm remarkably increases the compression quality when compared to the original TZ search algorithm, with the results 1.05% better in terms of BD-BR. Furthermore, in the proposed algorithm, the motion cost is estimated at all search points for all PUs. Among these search points, many of them are shared by different PUs. Consequently, because all redundant search positions are completely removed, the complexity of the SAD calculation of the proposed algorithm is also mitigated by 19.90% as compared to the original case.

One of the most important contributions of the proposed algorithm is that it enables the IME of all PU partitions at the same time, making it possible to decrement the number of search positions further if they are close to others and allowing them to be assumed as non-critical points. Table 2 shows numerous configurations of the proposed algorithm where additional complexity reduction schemes are applied to the concurrent

Table 1 The BD-BR values and complexity reduction results of the concurrent TZ Search

Class of test sequences	BD-BR (%)			Weighted BD-BR (wBDBR) (%)	Complexity reduction ($\mathrm{SAD_{cal}}$) (%)
	Y	U	V		
Class A (2560 × 1600)					
PeopleOnStreet	− 0.77	− 2.22	− 2.02	− 1.11	13.36
Traffic	− 1.20	− 1.26	− 0.80	− 1.16	13.21
Class B (1920 × 1080)					
BasketballDrive	− 0.87	− 0.61	− 0.96	− 0.85	43.41
BQTerrace	− 1.49	− 1.20	− 0.78	− 1.36	26.86
Cactus	− 0.68	− 0.78	− 0.38	− 0.66	17.61
Kimono	− 0.63	− 0.16	− 0.55	− 0.56	25.15
ParkScene	− 0.89	− 0.91	− 0.89	− 0.89	12.26
Class C (832 × 480)					
Keiba	− 1.82	− 2.26	− 2.02	− 1.90	14.07
BQMall	− 0.99	− 1.30	− 1.14	− 1.05	13.48
BasketballDrill	− 0.62	− 0.25	− 1.46	− 0.68	19.35
Flowervase	− 1.64	− 1.64	− 1.64	− 1.64	60.81
PartyScene	− 0.57	− 0.58	− 0.68	− 0.58	11.72
RaceHorses	− 2.80	− 1.53	− 1.56	− 2.48	18.49
Class D (416 × 240)					
BasketballPass	− 0.45	− 1.96	− 0.27	− 0.62	7.32
BlowingBubbles	− 0.68	0.09	− 0.05	− 0.50	9.96
BQSquare	− 0.94	0.28	1.18	− 0.52	14.24
RaceHorses	− 1.02	− 1.89	− 0.84	− 1.11	7.43
Class E (1280 × 720)					
FourPeople	− 0.77	− 0.50	− 0.99	− 0.76	25.20
Johnny	− 1.87	− 0.36	− 0.62	− 1.53	22.61
KristenAndSara	− 1.06	− 1.11	− 1.08	− 1.07	21.53
Average				− 1.05	19.90

Table 2 Various configurations of the proposed algorithm

Mode	Configuration properties						Weighted BD-BR (wBDBR) (%)	Complexity reduction (SAD_{cal}) (%)
	AMVP replacement	Reduced search points in raster search	Reduced search points in diamond search W: window size D: best distance					
			W = D/2	W = D/4	W = D/8	W = D/16		
1	Yes	No	No	No	No	No	− 1.05	19.90
2	No	No	No	No	No	No	− 0.83	22.58
3	Yes	Yes	No	No	No	No	− 0.92	32.62
4	Yes	No	Yes	–	–	–	− 0.82	40.32
5	Yes	No	–	Yes	–	–	− 0.90	31.67
6	Yes	No	–	–	Yes	–	− 0.90	40.71
7	Yes	No	–	–	–	Yes	− 0.88	24.39
8	Yes	Yes	Yes	–	–	–	− 0.84	54.54
9	Yes	Yes	–	Yes	–	–	− 0.91	41.02
10	Yes	Yes	–	–	Yes	–	− 0.91	40.17
11	Yes	Yes	–	–	–	Yes	− 0.88	36.20

TZ Search. In Table 2, mode 1 is the pure proposed algorithm when an AMVP replacement method is applied, as discussed in Sect. 3, and when no search point reduction methods are applied for both the raster and diamond search. In contrast, mode 2 is configured without this AMVP replacement method in order to observe the significant impacts of AMVP on the compression quality. In mode 3, only the raster search position reduction scheme is enabled. From mode 4 to mode 7, four different window sizes are applied to illustrate how the reduction scheme during the diamond search can affect the BD-BR. Ultimately, from mode 8 to mode 11, both search point reduction techniques are used for the diamond and raster searches. It is obvious that the final four configurations substantially simplify the IME process of the proposed algorithm. However, the BD-BR also increases owing to the aforementioned trade-off. Based on the BD-BR and the complexity values in Table 2, an appropriate configuration can be chosen for the desired hardware implementation. For instance, if the BD-BR is considered as the most important property in a hardware configuration, mode 1 is then the best choice for such a setting. By contrast, mode 8 is the one that minimizes the complexity of the proposed concurrent IME framework when non-redundant search positions are not only removed but a significant number of non-critical search points are also mitigated in the two proposed complexity reduction schemes. This results in complexity reduction of 54.54% for the MV cost calculation, where the achieved BD-BR is still considerably high, i.e., 0.84% better than in the original TZ Search.

The proposed algorithm configured in mode 8 is compared with various fast IME algorithms in [9–11] and [22]. In [9], the diamond search pattern is replaced by the rotating hexagon pattern—type 1—accompanying with an early termination process, which terminates the IME if the MV cost of a particular PU exceeds the normalized MV cost of this PU type. The normalized MV cost is specified by the size and shape of PUs, and it is derived after the first inter frame. In [10] and [11], hexagonal and pentagon search patterns are used instead of the diamond search pattern, respectively. A simple termination criterion for the first zonal search is adopted in [10] such that the first zonal search stops whenever the stride length is greater than iRaster = 5. In [22], they proposed a new search algorithm where two default search patterns in HEVC, i.e., diamond search and raster search patterns are not used. Instead, a descent-path search and a cross-pattern search are used to find the best IMVs. In Table 3, the first column shows the test sequences, whereas the other columns show the performance evaluation results compared to the HM13.0 reference software in terms of the weighted BD-BR (wBDBR) and the amount of SAD calculation reduction (SAD_{cal}). For fair comparison, the previous works are implemented and tested using the same parameter configurations, video test sequences, and HM version as the proposed work. As shown in Table 3, all previous works cannot achieve both the compression efficiency and the complexity reduction at the same time. Those approaches merely reduce the number of search positions by using simple search patterns and/or early termination schemes. Consequently, it directly hurts the compression efficiency. By contrast, the proposed algorithm does not experience that compression efficiency-complexity trade-off because

Table 3 Performance evaluation of the proposed algorithm and previous works

	Proposed		[9]		[10]		[11]		[22]	
Class of test sequences	wBD-BR (%)	SAD_{cal} (%)	wBD-BR (%)	SAD_{cal} (%)	wBD-BR (%)	SAD_{cal} (%)	wBD-BR(%)	SAD_{cal} (%)	wBD-BR (%)	SAD_{cal} (%)
Class A (2560 × 1600)										
PeopleOnStreet	− 0.83	36.4	0.49	13.04	0.56	21.28	0.68	23.2	1.15	73.17
Traffic	− 0.99	22.96	0.05	3.55	0.09	7.31	0.17	22.08	0.3	32.57
Class B (1920 × 1080)										
BasketballDrive	− 0.69	43.61	0.05	12.01	0.24	18.32	0.27	18.82	0.37	71.41
BQTerrace	− 0.95	61.12	0.26	5.55	0.32	11.33	0.25	22.04	0.46	33.49
Cactus	− 0.6	56.17	0.01	6.38	0.05	13.94	0.09	18.58	0.13	63.85
Kimono	− 0.52	60.16	0	6.83	− 0.06	15.8	0.03	17.32	0.09	70.01
ParkScene	− 0.81	55.61	0.02	− 1.36	− 0.02	4.08	0.01	15.42	0.17	36.48
Class C (832 × 480)										
Keiba	− 1.29	49.86	0.18	11.29	0.09	18.8	0.14	20.8	1.21	68.13
BQMall	− 0.78	52.32	− 0.05	15.64	− 0.16	21.46	− 0.02	25.05	0.15	65.57
BasketballDrill	− 0.66	56.64	0.14	12.24	0.3	19.34	0.55	22.92	0.61	73.32
Flowervase	− 1.31	78.84	0.19	4.52	0.07	8.62	0	23.07	0.75	42.62
PartyScene	− 0.36	53.04	0.16	6.1	0.07	15.3	0.24	20.7	0.25	57.65
RaceHorses	− 1.01	59.52	0.14	11.92	0.12	19.78	0.38	21.22	0.72	75.28
Class D (416 × 240)										
BasketballPass	− 0.67	53.01	0.12	11.03	0.11	17	0.25	23.85	0.32	57.9
BlowingBubbles	− 0.86	55.74	0.25	2.71	0.31	9.01	0.12	20.88	0.22	42.08
BQSquare	− 0.66	61.75	0.23	− 12.63	0.11	− 10.99	0.3	15.51	0.11	− 0.79
RaceHorses	− 0.91	57.23	0.22	11.18	0.17	20.1	0.53	22.58	0.54	71.17
Class E (1280 × 720)										
FourPeople	− 0.6	59.32	0.17	13.32	0.08	16.32	0.05	28.39	0.21	50.61
Johnny	− 1.24	58.15	0.21	9.67	0.21	12.59	0.52	27.47	0.67	36.99
KristenAndSara	− 1.04	59.44	0.11	9.3	− 0.07	13.57	− 0.09	25.75	0.12	45.33
Average	− 0.84	54.54	0.15	7.61	0.13	13.65	0.22	21.78	0.43	53.34

all PUs are searched at the same time in their summed paths, which enlarges the search point sets for each individual PU, keeping the search point set same for the whole CU. Thus, it gives more chances to find the accurate IMVs for each PU. While in this concurrent search process, only redundant points and non-critical points are removed enabling a great reduction in SAD calculation complexity without degrading the compression quality.

Figure 12 gives a more general illustration to compare the performance of the proposed algorithm and earlier works. The horizontal axis represents the SAD complexity reduction, whereas the vertical axis shows the weighted BD-BR values. Each point in Fig. 12 is plotted using values from Table 3. In terms of SAD complexity calculation, as shown in Fig. 12, the complexity reduction capacity of [9, 10], and [11] is very limited. On the other hand, the proposed algorithm and [22] show the similar degree of

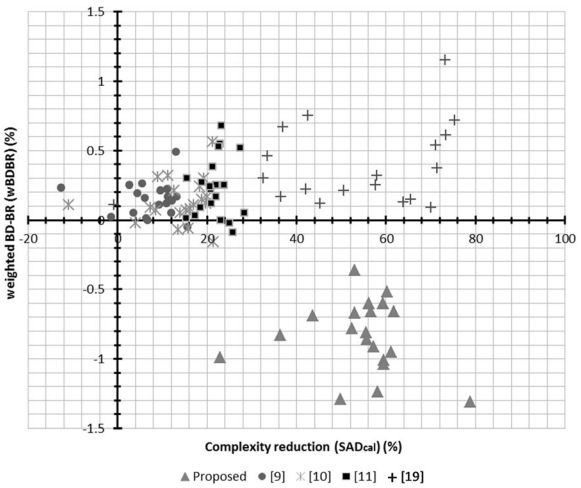

Fig. 12 Performance comparison of the proposed algorithm and previous works

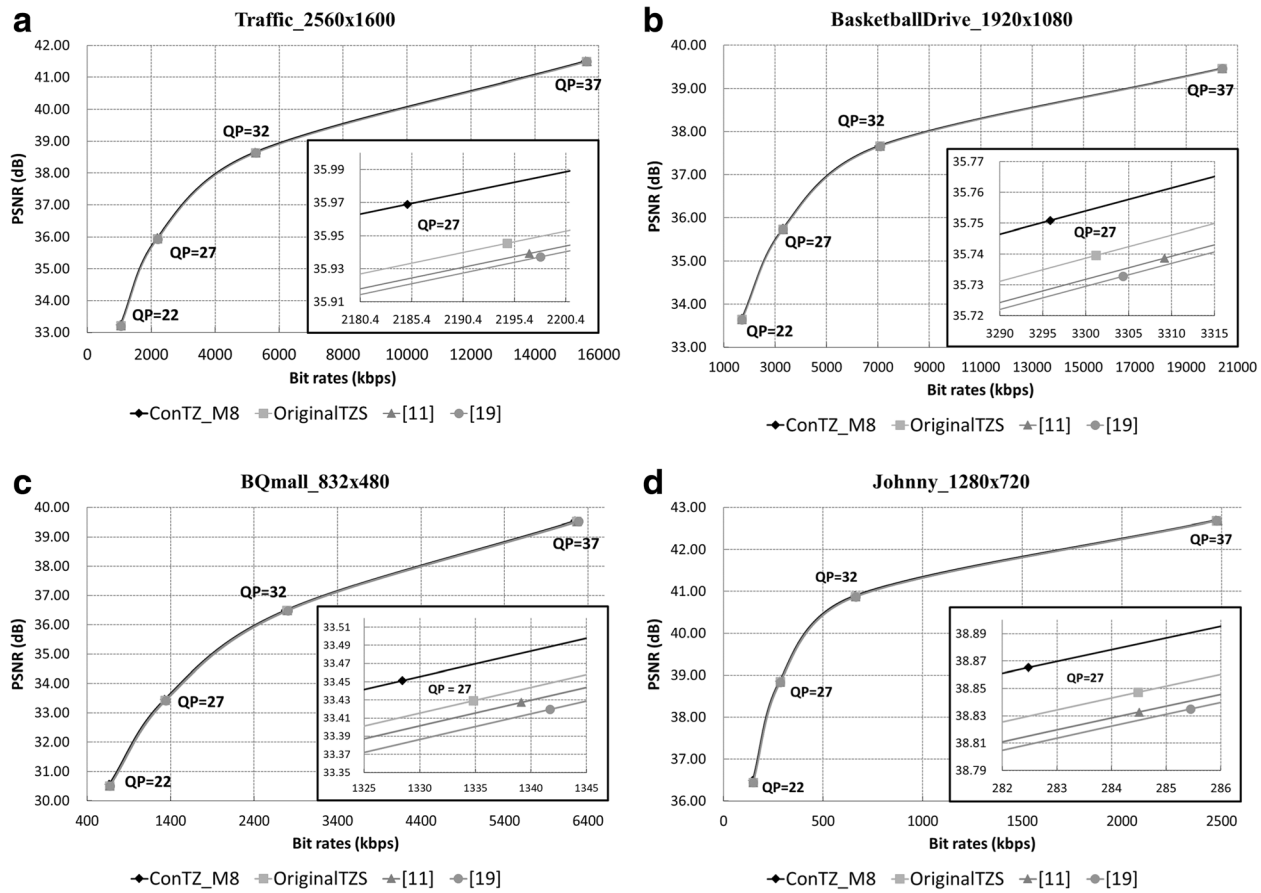

Fig. 13 RD performance of the proposed algorithm and others in various test sequences. **a** Traffic 2560 × 1600. **b** BasketballDrive 1920 × 1080. **c** BQMall 832 × 480. **d** Johnny 1280 × 720

complexity reduction, where the average SAD_{cal} for those works are 54.54 and 53.34%, respectively. It is worth to note that the SAD_{cal} values from the proposed scheme are mostly concentrated on the average value of 54%, whereas those results from [22] are widely spread. In terms of compression efficiency, only the proposed algorithm shows a significant decrease in BD-BR at the average of − 0.84%. For further comparison, RD curves of [11, 22], the original TZ Search (OriginalTZS), and the proposed algorithm in mode 8 (ConTZS_M8) are examined in Fig. 13. In each chart, the horizontal axis shows the bitrates while the vertical axis shows the PSNR. The small chart inside a box for each test sequence illustrates the enlarged part of its corresponding chart around the RD value at QP = 27. As clearly depicted in the enlarged charts, the proposed algorithm always achieves better compression quality (lower bitrates and higher PSNR) in comparison with other works.

6 Conclusion

In this paper, a fast IME algorithm is designed for its implementation in hardware, as doing so enables parallel processing for the IME phase in HEVC. In addition, for all configurations of the proposed method, the compression quality is always better than that of the original TZ Search because it increases the number of search positions at every PU partition by sharing the search areas and search paths of all PUs and as a result of the strict termination condition of the refinement search such that all PUs achieve the best match positions identical to their recent search centers. Owing to the natural property of the proposed algorithm, which changes the redundancy in terms of the search positions from temporal to spatial redundancy, all redundant search positions among all PUs are easily removed. Therefore, the complexity of the proposed algorithm can be remarkably decreased. Ultimately, for future research, the proposed algorithm should be implemented in the hardware, and its efficiency should be compared with those of previous hardware implementations for IME.

Acknowledgements

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2015R1C1A1A02037625) and by the Korea Institute for Advancement of Technology (KIAT) grant funded by the

Korean government (Motie: Ministry of Trade, Industry and Energy, HRD Program for Software-SoC convergence) (No. N0001883).

Authors' contributions
All authors contributed equally. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Inter-university Semiconductor Research Center, Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea. [2]Department of Information and Communication Engineering, Inha University, Incheon, Korea.

Received: 29 June 2016 Accepted: 6 November 2017
Published online: 22 November 2017

References

1. GJ Sullivan, J-R Ohm, W-J Han, T Wiegand, Overview of the High Efficiency Video Coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012)
2. P Helle et al., Block merging for quadtree-based partitioning in HEVC. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1720–1731 (2012)
3. L Shen et al. An effective CU size decision method for HEVC encoders. IEEE Trans. Multimedia **15**(2), 465–470 (2013)
4. F Bossen et al., HEVC complexity and implementation analysis. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1685–1696 (2012)
5. J-R Ohm et al., Comparison of the coding efficiency of video coding standards—including High Efficiency Video Coding (HEVC). IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1669–1684 (2012)
6. LC Manikandan, RK Selvakumar, A new survey on block matching algorithms in video coding. Int. J. Eng. Res. **3**(2), 121–125 (2014)
7. Y-W Huang et al., Survey on block matching motion estimation algorithms and architectures with new results. J. VLSI Signal Process. Syst. **42**(3), 297–320 (2006)
8. CE Rhee et al. A survey of fast mode decision algorithms for inter-prediction and their applications to High Efficiency Video Coding. IEEE Trans. Consum. Electron. **58**(4), 1375–1383 (2012)
9. N Purnachand, LN Alves, A Navarro, in Proceedings of IEEE International Conference on Consummer Electronics-Berlin (ICCE-Berlin). Fast motion estimation algorithm for HEVC (Berlin, 2012), pp. 34–37
10. N Purnachand, LN Alves, A Navarro, in Proceedings of IEEE International Conference on Systems, Signals and Image Processing (IWSSIP). Improvements to TZ search motion estimation algorithm for multiview video coding (Austria, 2012), pp. 388–391
11. N Parmar, MH Sunwoo, in Proceedings of IEEE International SoC Design Conference (ISOCC). Enhanced test zone search motion estimation algorithm for HEVC (South Korea, 2014), pp. 260–261
12. X Li et al., in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS). Context-adaptive fast motion estimation of HEVC (Portugal, 2015), pp. 2784–2787
13. H Kibeya et al., in Proceedings of IEEE International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). TZ Search pattern search improvement for HEVC motion estimation modules (Tunisia, 2014), pp. 95–99
14. LP Van et al., in Proceedings of IEEE International Conference on Image Processing (ICIP). Fast motion estimation for closed-loop HEVC transrating (France, 2014), pp. 2492–2496
15. L Gao et al., in Proceedings of IEEE International Conference on Image Processing (ICIP). A novel integer-pixel motion estimation algorithm based on quadratic prediction (Canada, 2015), pp. 2810–2814
16. G-L Li, C-C Wang, K-H Chiang, in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). An efficient motion vector prediction method for avoiding AMVP data dependency for HEVC (Italy, 2014), pp. 7363–7366
17. Q Yu, L Zhao, S Ma, in Proceedings of IEEE Visual Communications and Image Processing (VCIP). Parallel AMVP candidate list construction for HEVC (USA, 2012), pp. 1–6
18. X Jiang et al., in Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). AMVP prediction algorithm for adaptive parallel improvement of HEVC (Japan, 2014), pp. 511–514
19. S Rehman, R Young, C Chatwin, P Birch, An FPGA based generic framework for high speed sum of absolute difference implementation. Eur. J. Sci. Res. **33**(1), 6–29 (2009)
20. P Nalluri, LN Alves, A Navarro, in Proceedings of IEEE International Conference on Image Processing (ICIP). High speed SAD architectures for variable block size motion estimation in HEVC video coding (France, 2014), pp. 1233–1237
21. Y Fan, L Huang, B Hao, X Zeng, A hardware-oriented IME algorithm for HEVC and its hardware implementation. IEEE Transactions on Circuits and Systems for Video Technology (2017). doi:10.1109/TCSVT.2017.2702194
22. S-H Yang, J-Z Jiang, H-J Yang, Fast motion estimation for HEVC with directional search. Electron. Lett. **50**(9), 673–675 (2014)