Master of Science Thesis

A Hardware-Oriented Concurrent Test Zone Search Algorithm for High-Efficiency Video Coding

February 2017

Graduate School of Engineering

Seoul National University

Electrical and Computer Engineering Major

Doan Trung Nghia

A Hardware-Oriented Concurrent Test Zone Search Algorithm for High-Efficiency Video Coding

Supervisor: Prof. Hyuk-Jae Lee

This work is submitted as a Master of Science thesis

January 2017

Graduate School of Engineering
Seoul National University
Electrical and Computer Engineering Major

Doan Trung Nghia

Confirming the master's thesis written by
Doan Trung Nghia
January 2017

Chair _____

Vice Chair _____

Examiner _____

Abstract

High-Efficiency Video Coding (HEVC) [1-5] 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 hardware. This work 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)

Student Number: 2015-22134

Table of Contents

List of Figures

List of Tables

CHAPTER 1 – INTRODUCTION

1.1. A Brief History of the HEVC Standard

The HEVC project was formally launched in January 2010 when a joint Call for Proposals (CfP) was issued by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). Before launching the formal CfP, both organizations had conducted investigative work to determine that it was feasible to create a new standard that would substantially advance the state of the art in compression capability—relative to the prior major standard known as H.264/MPEG-4 Advanced Video Coding (AVC- the first version of which was completed in May 2003). One notable aspect of the investigative work toward HEVC was the "key technology area" (KTA) studies in VCEG that began around the end of 2004 and included the development of publicly-available KTA software codebase for testing various promising algorithm proposals. In MPEG, several workshops were held, and a Call for Evidence (CFE) was issued in 2009. When the two groups both reached the conclusion that substantial progress was possible and that working together on the topic was feasible, a formal partnership was established and the joint CfP was issued.

The VCEG KTA software and the algorithmic techniques found therein were used as the basis of many of the proposals submitted in response to both the MPEG CfE and the joint CfP.

The major video coding standard directly preceding the HEVC project was H.264/MPEG-4 AVC, which was initially developed in the period between 1999 and 2003, and then was extended in several important ways from 2003–2009. H.264/MPEG-4 AVC has been an enabling technology for digital video in almost every area that was not previously covered by H.262/MPEG-2 Video and has substantially displaced the older standard within its existing application domains. It is widely used for many applications, including broadcast of high definition (HD) TV signals over satellite, cable, and terrestrial transmission systems, video content acquisition and editing systems, camcorders, security applications, Internet and mobile network video, Blu-ray Discs, and real-time conversational applications such as video chat, video conferencing, and telepresence systems.

However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond HD formats (e.g., 4k×2k or 8k×4k resolution) are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities. The need is even stronger when higher

resolution is accompanied by stereo or Multiview capture and display. Moreover, the traffic caused by video applications targeting mobile devices and tablet PCs, as well as the transmission needs for video-on-demand services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications. Interest in developing a new standard has been driven not only by the simple desire to improve compression as much as possible—e.g., to ease the burden of video on storage systems and global communication networks, but also to help enable the deployment of new services, including capabilities that have not previously been practical—such as ultra-high-definition television (UHDTV) and video with higher dynamic range, wider color gamut, and greater representation precision than what is typically found today.

To formalize the partnership arrangement, a new joint organization was created, called the Joint Collaborative Team on Video Coding (JCT-VC). The JCT-VC met four times per year after its creation, and each meeting had hundreds of attending participants and involved the consideration of hundreds of contribution documents (all of which were made publicly available on the web as they were submitted for consideration).

The project had an unprecedented scale, with a peak participation reaching about 300 people and more than 1,000 documents at a single meeting. Meeting notes were publicly released on a daily basis during meetings, and the work continued between meetings, with active discussions by email on a reflector with a distribution list with thousands of members, and with formal coordination between meetings in the form of work by "ad hoc groups" to address particular topics and "core experiments" to test various proposals. Essentially the entire community of relevant companies, universities, and other research institutions was attending and actively participating as the standard was developed.

HEVC has been designed to address essentially all existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures. The syntax of HEVC is generic and should also be generally suited for other applications that are not specifically mentioned above.

As has been the case for all past ITU-T and ISO/IEC video coding standards, in HEVC only the bitstream structure and syntax is standardized, as well as constraints on the bitstream and its mapping for the generation of decoded pictures. The mapping is given by defining the semantic meaning of syntax elements and a decoding process such that every decoder

conforming to the standard will produce the same output when given a bitstream that conforms to the constraints of the standard. This limitation of the scope of the standard permits maximal freedom to optimize implementations in a manner appropriate to specific applications (balancing compression quality, implementation cost, time to market, and other considerations). However, it provides no guarantees of end-to-end reproduction quality, as it allows even crude encoding techniques to be considered conforming.

To assist the industry community in learning how to use the standard, the standardization effort not only includes the development of a text specification document but also reference software source code as an example of how HEVC video can be encoded and decoded. The draft reference software has been used as a research tool for the internal work of the committee during the design of the standard, and can also be used as a general research tool and as the basis of products. A standard test data suite is also being developed for testing conformance to the standard.

1.2. The Integer Motion Estimation Block in HEVC

In HEVC [1-5], 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] 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 is thus is not proper when applied for hardware implementation owing to the fact that parallelism is not explicitly exploited.

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, our approach 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., bit rate reduction).

CHAPTER 2 - THE CONVENTIONAL TZ SEARCH ALGORITHM AND RELATED WORKS

2.1. 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. Fig. 1 (a) 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 based on competition among several predictors, specifically the median predictor, left predictor, up predictor and the upper-right predictor. Which predictor among those mentioned above results in the lowest SAD cost is employed as the best predictor for the current PU. 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. Fig. 1 (b) presents an illustration of the diamond search with the stride ranging from 1 to 3. The variable

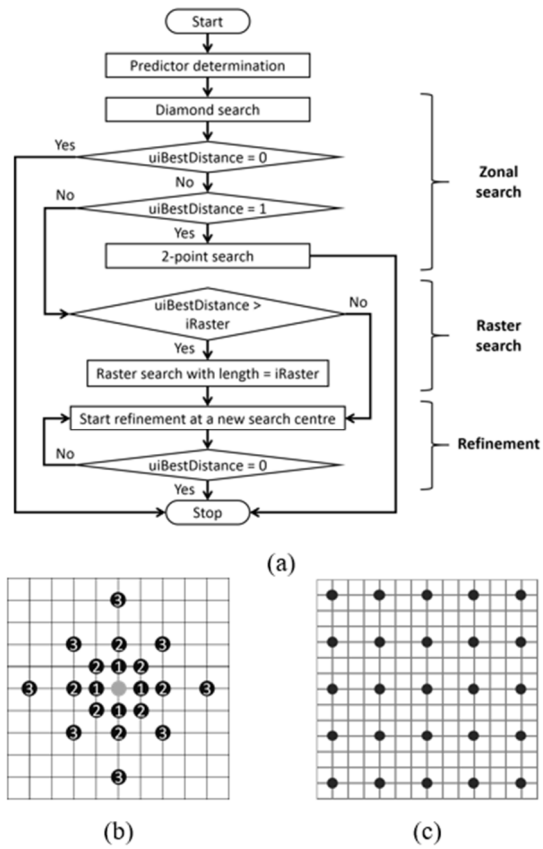

Figure 1. The conventional TZ search algorithm: (a) flowchart of the conventional TZ search algorithm, (b) eight-point diamond search pattern , and (c) diamond search when iRaster = 3

"uiBestDistance" stores the distance between the best match point (the current position that gives the lowest SAD cost of the current PU) and the current search center. After the first grid search, if uiBestDistance equals 0, there is no need to undertake additional search steps. If uiBestDistance equals 1, the best match can be obtained with the two-point search. 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. Fig. 1 (c) presents an example of a raster search when iRaster equals 3. In addition, as shown in Fig. 1 (a), 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 equals 0, the refinement process is successfully completed.

2.2. Related Works in the Integer Motion Estimation in Video Coding

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. Moreover, 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. In addition, 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 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 pattern is adaptively switched based on the Motion Vector Differences (MVD) in the predicted neighboring blocks.

Furthermore, a group of three search patterns is selectively used [13] 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 [14], where the search range of the current PU is determined by a learning algorithm following by 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 SAD 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.

CHAPTER 3 - THE 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×2N, N×2N and 2N×N PUs [2-3], are processed for all CU sizes {8×8, 16×16, 32×32, 64×64} 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×nU, 2N×nD, nL×2N and nR×2N PUs [2-3]. The original TZ Search is applied sequentially for each PU partition in the order as illustrated in Fig. 2(a). In contrast, the proposed algorithm processes PUs sequentially; instead, all PUs are processed in parallel, as described in Fig. 2 (b).

Fig. 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

```
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);
   }
}
```

(a)

```
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)
   }
}
```

(b)

Figure 2. The pseudo code of the fast integer motion estimation process: (a) original TZ search algorithm, and (b) hardware-oriented concurrent TZ search algorithm

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

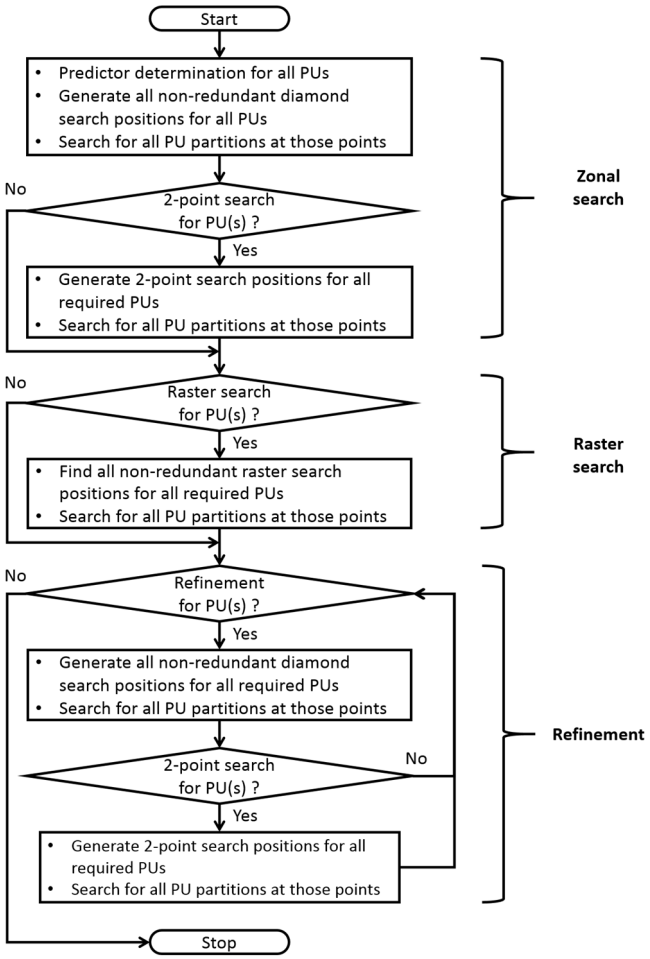

Figure 3. The hardware-oriented concurrent TZ Search

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 2-point 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 2-point search, when the value of uiBestDistance equals 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 test 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. Similar to 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 SAD cost calculation for any PU. After the raster search, the third search step of the concurrent TZ search is the refinement process which 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 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

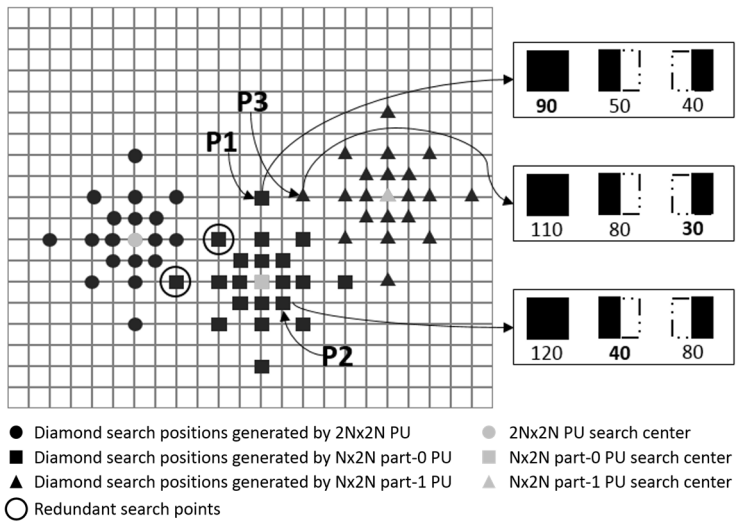

Figure 4. The first diamond search of the concurrent TZ Search algorithm

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

3.2.1. The Modified First Search

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 grey 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 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×2N part-0 PUs. Those points are examined only once given that before engaging in the SAD 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 cost is only calculated for the largest 2N×2N PU, after which this SAD cost is divided into smaller parts prior to being assigned to other PU partitions. The process of the aforementioned SAD cost calculation is 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 SAD values 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 SAD costs, respectively, for the corresponding PUs. At point P1, from left to right, the 2N×2N PU has a SAD cost of 90, and the assigned SAD values for N×2N part-0 and N×2N part-1 are 50 and 40 respectively. Identical processes are used for the two remaining SAD cost calculation examples at points P2 and P3. It should be noticed that the SAD cost written in bold shows the most recent smallest SAD value for a particular PU. Obviously, such a value is obtained when this PU partition reaches its current best mach. 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 simplified

definition of the Euler 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 Euler 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.

3.2.2. The Modified 2-Point Search

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 SAD

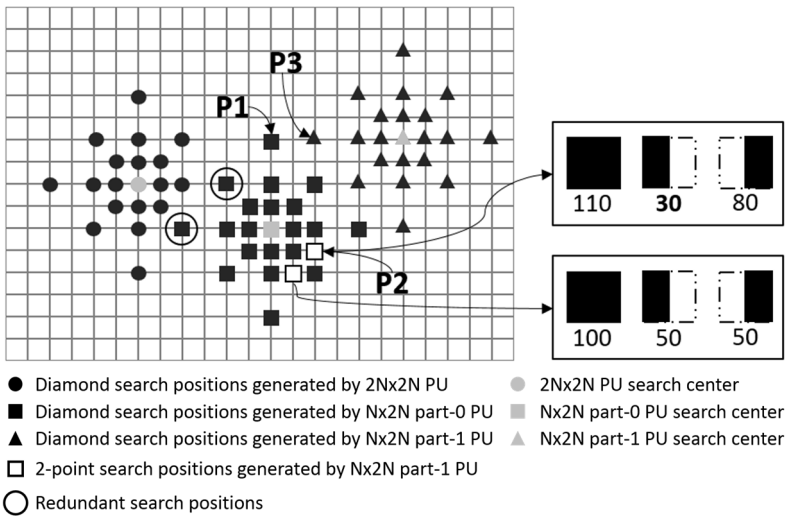

Figure 5. Two-point search for the N×2N part-0 PU

cost calculation process is performed in the same manner as examined above. 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 SAD

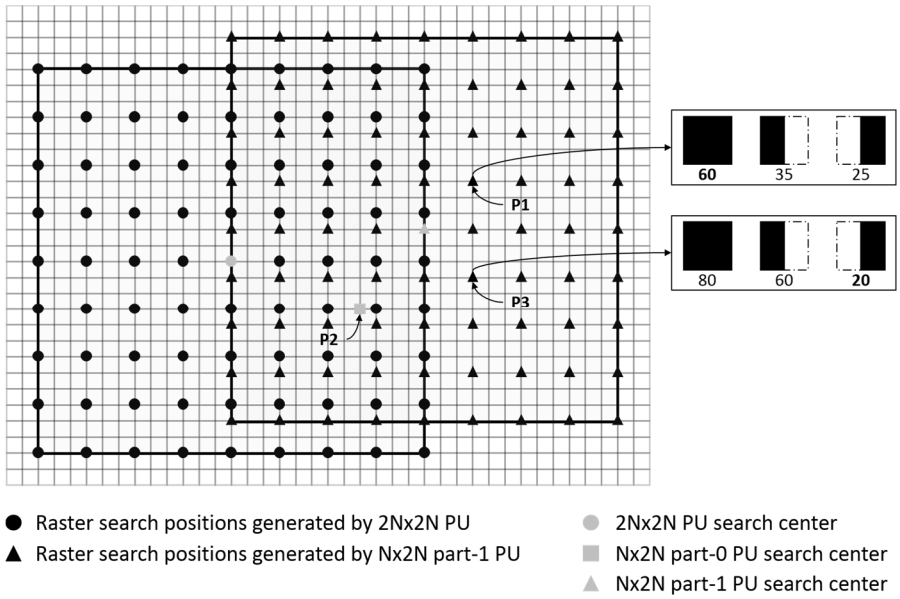

Figure 6. Raster search for the 2N×2N and N×2N part-1 PUs

value is 30, smaller than its previous best match (40) and the other SAD values derived for it. On the other hand, other PUs retain their previous best match positions (P1 and P3) with SAD values of 90 and 30, respectively, because there are no smaller SAD values obtained for those PUs in this step.

3.2.3. The Modified Raster Search

Fig. 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 grey. Moreover, the SAD cost calculation is then performed using an identical process at every position of the raster search point set. At the end of the raster search, there is no position containing a lower SAD 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 SAD cost is still 30. In contrast, P1 and P3 are obtained at the two new positions described in Fig. 6. The SAD values derived at 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) = 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.

3.2.4. The Modified Refinement Search

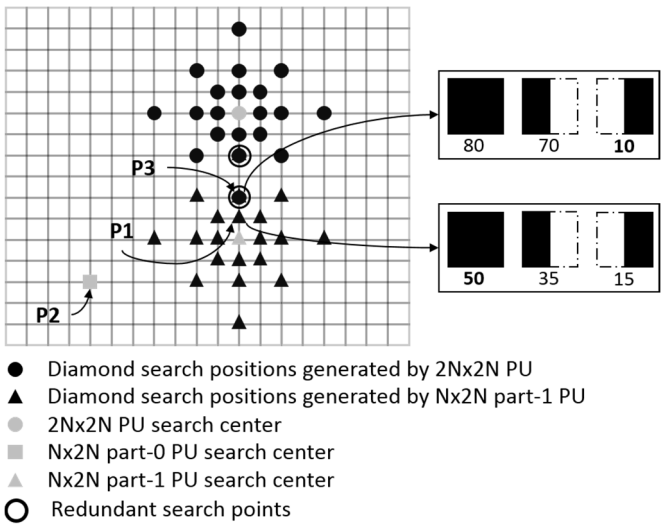

Figure 7. The first iteration of the refinement search

Fig. 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 2N×2N and N×2N part-1 PUs based on the diamond pattern, and once all points are completely created, redundant points are removed prior to the SAD cost calculation. There are two unessential positions in the first refinement iteration, as denoted by the two circles shown in Fig. 7. When the SAD cost process is completed, P1 once again reaches its current lowest SAD value in one of the search positions initially belonging to the N×2N part-1 PU. At that point, the minimum SAD value 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 SAD value lower than

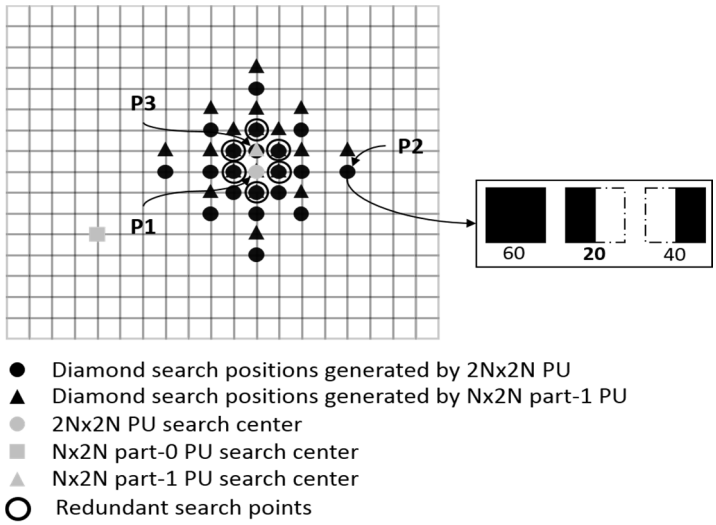

Figure 8. The second iteration of the refinement search

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 SAD 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 Nx2N part-0 PU.

The second iteration of the given example is intended for situation when the search centers of different 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 SAD 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 SAD values as compared to those of all newly generated cases. Previously, the N×2N part-0 PU reached its best match with the SAD cost equal to 30; this position is depicted by the small grey 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 SAD 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 above requirement of the aforementioned termination conditions applied in the refinement phase.

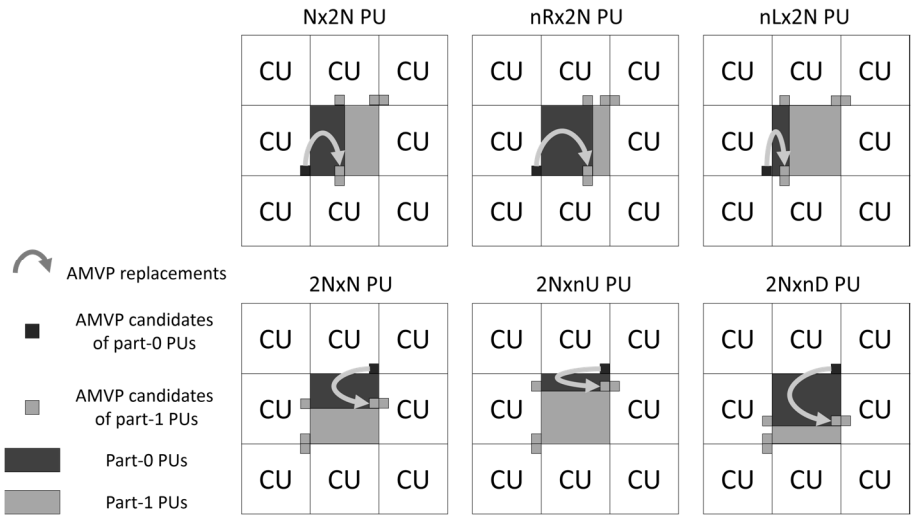

Figure 9. Unavailable AMVP replacement scheme

3.2.5. A Measure to Overcome the AMVP Dependence Issue

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 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 grey. 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 [17-19]. 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 [18]. 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.

CHAPTER 4 - COMPLEXITY REDUCTION SCHEMES OF 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

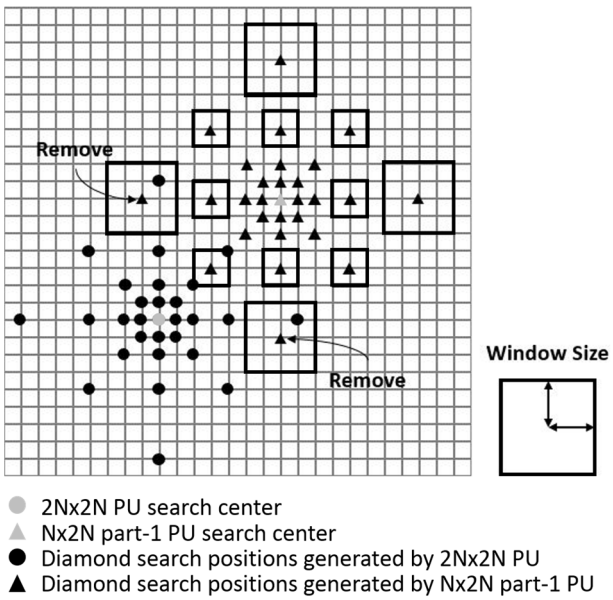

Figure 10. Search point reduction scheme for the diamond search

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 SAD 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 section V.

Fig. 10 gives a complete 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×2N PU and the N×2N part-1 PU. All search positions of the 2N×2N PU are initially generated with iRaster equal to 3. The size of the window that takes a non-critical point of the N×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 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 (2x2); whereas windows 1x1 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

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 ∈ $\{R_1 \| R_2 \| \ldots \| R_{n-1}\}$ → 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**

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

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 SAD 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 SAD cost calculation. Thus it cannot increase the computation time for the proposed concurrent TZ search algorithm. Finally, 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.

CHAPTER 5 - EXPERIMENTAL RESULTS

5.1. Complexity Measure

The complexity measure for hardware implementation must be different from that for software implementation. Therefore, a new definition of complexity measure is introduced for the purposes of this paper. In the IME block of HEVC, the most time-consuming task is the SAD cost calculation, which involves the calculation of the sum of the differences of each pixel between the current PU and the reference PU. Note that every PU size can be divided into multiple 4×4 blocks; hence, the complexity required for the SAD cost calculation of a 4×4 block is defined as one unit of complexity. For instance, the SAD cost 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 that of all sub PUs; therefore, the total complexity required for the CU is calculated as follows:

$$CU_{OrigTZSCal} = \sum_{AllPUs} PU_{Cal} = \sum_{AllPUs} PU_{CalUnit} \times N_{PU} \quad ($$

1)

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

In the concurrent TZ search algorithm, it is assumed that the complexity of the SAD cost assignment from the largest 2N×2N 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 2N×2N PU:

$$CU_{ConTZSCal} = PU_{2N\times 2NCalUnit} \times N_{2N\times 2N} \quad (2)$$

Where $CU_{ConTZSCal}$ is the complexity of the SAD cost calculation for a CU in the proposed concurrent TZ search algorithm, $PU_{2N\times 2NCalUnit}$ is the number of units of complexity for a 2N×2N PU, $N_{2N\times 2N}$ is the number of search positions generated for a 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*.

$$\begin{aligned} CU_{OrigTZSCal} &= N \times \sum_{AllPUs} PU_{CalUnit} \\ &= N(PU_{2N\times 2NCalUnit} + PU_{N\times 2Npart0CalUnit} + \\ &PU_{N\times 2Npart1CalUnit} + PU_{2N\times Npart0CalUnit} + PU_{2N\times Npart1CalUnit}) \end{aligned} \quad (3)$$

Where:

$$\begin{aligned} PU_{2N\times 2NCalUnit} &= \frac{Size(PU_{2N\times 2N}, Hor)\times Size(PU_{2N\times 2N}, Ver)}{4\times 4} \\ &= \frac{8\times 8}{4\times 4} = 4(units) \end{aligned} \quad (4)$$

and

$$\begin{aligned} &PU_{N\times 2Npart0CalUnit} = PU_{N\times 2Npart1CalUnit} \\ &= PU_{2N\times Npart0CalUnit} = PU_{2N\times Npart1CalUnit} \\ &= \frac{Size(PU_{N\times 2Npart0}, Hor)\times Size(PU_{N\times 2Npart1}, Ver)}{4\times 4} \\ &= \frac{4\times 8}{4\times 4} = 2(units) \end{aligned} \quad (5)$$

By substituting (4) and (5) to (3), the total complexity of the current 8×8 CU in the case of the original TZ search algorithm is determined, as follows:

$$CU_{OrigTZCal} = 12N(units) \quad (6)$$

6)

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

$$CU_{ConTZSCal} = N_{2N\times 2N} \times PU_{2N\times 2NCalUnit} \quad (7)$$
$$= (5N \times 52\%) \times PU_{2N\times 2NCalUnit} = 10.4N(units)$$

In the example above, on average the complexity of an 8×8 CU needed for the SAD cost 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 Section IV.A. In addition, video

sequences from class A to E are examined to evaluate the impacts of the proposed algorithm with the low delay P configuration taken at 100 frames with various quantization parameters varying among 22, 27, 32, and 37.

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 ($CP_{reduced}$) (%)
	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 1 shows the experimental results of the video test sequences from class A to E of the proposed concurrent 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 calculated as described below.

$$wBDBR = \frac{6 \times BDBR_{Y} + BDBR_{U} + BDBR_{V}}{8} \quad (8)$$

The last column gives the values representing the amount of complexity reduction as compared to that in the original algorithm, which is calculated from the following equation.

$$CP_{reduced} = \left(1 - \frac{CU_{ConTZSCal}}{CU_{OrigTZSCal}}\right) \times 100\ (\%) \quad (9)$$

Finally, 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 cost calculation of the proposed algorithm is also mitigated by 19.90% as compared to the original case.

Table 3. Various Configurations of the Proposed Algorithm

Mode	Configuration properties						BD-BR (%)	Complexity reduction (%)
	AMVP replacement	Reduced search points in raster search	Reduced search points in a 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

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 TZ Search. In Table 2, Mode 1 is the pure proposed algorithm when an AMVP replacement method is applied, as discussed in section III, 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 SAD cost calculation, where the achieved BD-BR is still considerably high, i.e., 0.84% better than in the original TZ Search.

Table 3 compares different works in the literature on TZ search-based IME algorithms, as introduced in earlier work [9] [14] [28] [29] and in the

Table 5. Comparison of Performances with those in Previous Works

Test sequences	[9]		[28]		[14]		[29]		Proposed	
	Cmplx. Reduc. (AMS) (%)	BD-BR (%)	Cmplx. Reduc. (AMS) (%)	BD-BR (%)	Cmplx. Reduc. (TTS) (%)	BD-BR (%)	Cmplx. Reduc. (TTS) (%)	BD-BR (%)	Cmplx. Reduc. (SAD cal.) (%)	BD-BR (%)
Class B (1920 × 1080)	-	-	-	-	66.03	2.75	82.58	2.75	38.86	-0.78
Class C (832 × 480)	46.51	-0.03	75.61	0.15	61.78	2.45	80.98	2.63	39.97	-1.05
Class D (416 × 240)	47.19	-0.01	64.12	0.16	64.45	2.2	77.85	2.37	31.10	-0.69
Class E (1280 × 720)	-	-	-	-	74.53	2.32	87.45	2.37	32.89	-0.93
Average	**46.85**	**-0.02**	**69.86**	**0.16**	**66.70**	**2.43**	**83.21**	**2.53**	**35.71**	**-0.86**

proposed paper. The first column shows the test sequences with their corresponding resolutions, and other columns illustrate the performance evaluation results in accordance with the complexity reduction and the BD-BR outcomes for each work. Note that only two video sequences are used in the evaluations [9] and [28] in each video class, the BQMall and PartyScene sequences for Class C, while for Class D, the BasketballPass and RaceHorses testing sequences are used. The earlier methods [9] and [28] have quite similar approaches, where various search patterns and different fast IME procedures are introduced. The complexity reduction of these schemes [9] and [28] are determined according to the reduction ratio of the reduced ME time overall (the average ME time savings, AMS [9]). It can clearly be observed in Table III that both works ([9] and [28]) achieve remarkable ME time-savings amounts of 46.85% [9] and 69.86% [28];

while the former [9] increases BD-BR negligibly to 0.02%, the latter [28] has to account for the trade-off by increasing BD-BR to 0.16%. In two studies [14] and [29], the methods seek to apply data analysis techniques to select the appropriate MVs. In addition, the complexity measure applied in these algorithms is the transrating time saving (TTS) analysis [29]. When compared to the original HM software, those two studies [14] and [29] achieve excellent complexity reductions of 66.70% and 83.21%, respectively. In order to reach such quantities, the BD-BR values of those algorithms are strongly increased to 2.43% and 2.53%, respectively. When compared to other studies, as listed in Table III, the concurrent TZ search algorithm is not affected by the normal performance trade-off, as the achieved BD-BR is -0.86% and the complexity reduction is 35.71% for the SAD cost reduction. Note that Mode 8 introduced in Table II is configured for the proposed algorithm in order to compare the outcome with those by other methods.

It was found that merely evaluating the encoding time given in the standard HM reference software could lead to unsatisfactory comparisons when the complexity reduction by the proposed algorithm is evaluated with those in other papers. This occurs because the concurrent TZ Search presented in this paper is intended for a hardware-based algorithm for the

fast IME; yet, near-hardware implementation in the HM software could lead to a longer computing time because parallel programming is unavailable. By contrast, other works often make use of the encoding time given by the HM software for time-savings evaluations. Therefore, no explicit comparisons are appropriate to differentiate the speed achievements among all of the algorithms listed in Table III. However, the conclusion which can be derived from Table III is that while all of the approaches can substantially reduce the complexity of the fast IME, not all of the algorithms can maintain a high level of video compression quality. This is obvious, as without modifying the original IME procedure as in the proposed paper, other IME algorithms can scarcely produce better performance than the mature and efficient original TZ Search, which is well developed for HEVC.

CHAPTER 6 – CONCLUSION

The TZ search algorithm is a widely-used algorithm and it is adopted in the reference software of the High Efficiency Video Coding (HEVC) standard. Although the TZ search algorithm is very efficient to obtain accurate motion vector with reasonable complexity, it is designed for software implementation, not for hardware implementation. The TZ search algorithm performs motion estimation for PUs sequentially with each PU tracking its own search points. Therefore, different PUs track different search points for motion estimation. Aiming at hardware execution, this thesis proposes a new IME search algorithm that modifies the TZ search algorithm to make multiple PUs estimate the motion cost in a parallel manner without a degradation of the compression efficiency. To this end, the proposed algorithm obtains the search points for all PUs prior to the execution of the motion estimation for any PU. Then, the motion estimation is performed for all PUs in a CU for all the obtained search points. This order of operations is suitable for hardware execution which can perform the motion estimation of multiple PUs in a parallel manner. The proposed search algorithm increases the number of search positions for individual PUs although the total number of search points for all PUs are the same.

This increase of search points may slow down the execution of the algorithm in software but may not affect the speed of hardware execution if the hardware is designed to perform multiple PUs in parallel. In addition, the strict termination condition of the refinement search enables 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 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 in terms of SAD cost calculation can be remarkably decreased. Ultimately, for future research, the proposed algorithm should be implemented in hardware and its efficiency should be compared with those of previous hardware implementations for IME.

Bibliography

[1] Sullivan, Gary J., et al. "*Overview of the high efficiency video coding (HEVC) standard.*" IEEE Transactions on Circuits and Systems for Video Technology, 22.12 (2012): 1649-1668.

[2] Helle, Philipp, et al. "*Block merging for quadtree-based partitioning in HEVC.*" IEEE Transactions on Circuits and Systems for Video Technology, 22.12 (2012): 1720-1731.

[3] Helle, Philipp, et al. "*Block merging for quadtree-based partitioning in HEVC.*" IEEE Transactions on Circuits and Systems for Video Technology, 22.12 (2012): 1720-1731.

[4] Bossen, Frank, et al. "*HEVC complexity and implementation analysis.*" IEEE Transactions on Circuits and Systems for Video Technology, 22.12 (2012): 1685-1696.

[5] Ohm, J-R., et al. "*Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC).*" IEEE Transactions on Circuits and Systems for Video Technology, 22.12 (2012): 1669-1684.

[6] Manikandan, L. C., and R. K. Selvakumar. "*A New Survey on Block Matching Algorithms in Video Coding.*" International Journal of Engineering Research 3.2 (2014): 121-125.

[7] Huang, Yu-Wen, et al. "*Survey on block matching motion estimation algorithms and architectures with new results.*" Journal of VLSI Signal Processing Systems 42.3 (2006): 297-320.

[8] Rhee, Chae Eun, et al. "*A survey of fast mode decision algorithms for inter-prediction and their applications to high efficiency video coding.*" IEEE Transactions on Consumer Electronics, 58.4 (2012): 1375-1383.

[9] Purnachand, N., Luis Nero Alves, and Antonio Navarro. "*Fast motion estimation algorithm for HEVC.*" Consumer Electronics-Berlin (ICCE-Berlin), 2012 IEEE International Conference on. IEEE, 2012.

[10] Purnachand, N., Luis Nero Alves, and Antonio Navarro. "*Improvements to TZ search motion estimation algorithm for multiview video coding.*" The 19th International Conference on Systems, Signals and Image Processing (IWSSIP), 2012. IEEE, 2012.

[11] Parmar, Nidhi, and Myung Hoon Sunwoo. "*Enhanced Test Zone search motion estimation algorithm for HEVC.*" 2014 International SoC Design Conference (ISOCC), IEEE, 2014.

[12] Li, Xufeng, et al. "*Context-adaptive fast motion estimation of HEVC.*" 2015 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2015.

[13] Kibeya, Hassan, et al. "*TZ Search pattern search improvement for HEVC motion estimation modules.*" The 1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), IEEE, 2014.

[14] Van, Luong Pham, et al. "*Fast motion estimation for closed-loop HEVC transrating.*" 2014 IEEE International Conference on Image Processing (ICIP), IEEE, 2014.

[15] Gao, Longfei, et al. "*A novel integer-pixel motion estimation algorithm based on quadratic prediction.*" 2015 IEEE International Conference on Image Processing (ICIP), IEEE, 2015.

[16] Doan Trung Nghia, Tae Sung Kim, Hyuk-Jae Lee, and Soo-Ik Chae, "*A Modified TZ Search Algorithm for Parallel Integer Motion Estimation in High Efficiency Video Coding,*" The 12th International SoC Design Conference (ISOCC), IEEE, 2015.

[17] Li, Gwo-Long, Chuen-Ching Wang, and Kuang-Hung Chiang. "*An efficient motion vector prediction method for avoiding AMVP data dependency for HEVC.*" 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014.

[18] Yu, Qin, Liang Zhao, and Siwei Ma. "*Parallel AMVP candidate list construction for HEVC.*" 2012 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2012.

[19] Jiang, Xiantao, et al. "*AMVP prediction algorithm for adaptive parallel improvement of HEVC.*" 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), IEEE, 2014.

[20] Rhee, Chae Eun, Jin-Su Jung, and Hyuk-Jae Lee. "*A real-time H. 264/AVC encoder with complexity-aware time allocation*." IEEE Transactions on Circuits and Systems for Video Technology, 20.12 (2010): 1848-1862.

[21] Shukla, Rahul, et al. "*Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images.*" IEEE Transactions on Image Processing, 14.3 (2005): 343-359.

[22] Richardson, Iain E. "*The H. 264 advanced video compression standard.*" John Wiley & Sons, 2011.

[23] Wiegand, Thomas, et al. "*Overview of the H. 264/AVC video coding standard.*" IEEE Transactions on Circuits and Systems for Video Technology, 13.7 (2003): 560-576.

[24] Rhee, Chae Eun, Jin-Su Jung, and Hyuk-Jae Lee. "*A real-time H. 264/AVC encoder with complexity-aware time allocation.*" IEEE Transactions on Circuits and Systems for Video Technology, 20.12 (2010): 1848-1862.

[25] Rhee, Chae Eun, Jin-Sung Kim, and Hyuk-Jae Lee. "*Cascaded direction filtering for fast multidirectional inter-prediction in H. 264/AVC main and high profile compression.*" IEEE Transactions on Circuits and Systems for Video Technology, 22.3 (2012): 403-413.

[26] Lee, Kyujoong, et al. "*Simplified algorithms for rate-distortion optimization in high efficiency video coding.*" Displays 40 (2015): 35-44.

[27] Coding, High Efficiency Video. "*Document ITU-T Rec.*" H 265 (2013): 23008-2.

[28] Yang, Shih-Hsuan, Jia-Ze Jiang, and Hsien-Jie Yang. "*Fast motion estimation for HEVC with directional search.*" Electronics Letters 50.9 (2014): 673-675.

[29] Van, Luong Pham, et al. "*Fast transrating for high efficiency video coding based on machine learning.*" 2013 20th IEEE International Conference on. Image Processing (ICIP), IEEE, 2013.

[30] Kim, Tae Sung, Chae Eun Rhee, and Hyuk-Jae Lee. "*Merge Mode Estimation for a Hardware-Based HEVC Encoder.*" IEEE Transactions on Circuits and Systems for Video Technology 26.1 (2016): 195-209.

Abstract in Korean

High-Efficiency Video Coding (HEVC) [1-5]은최신의영상부호화표준으로써, 그압축성능은동일한품질의영상조건에서, 이전세대의영상부호화표준인H.264/AVC의두배이다. Test Zone (TZ) search 알고리즘은비교적적은연산으로, 정확한움직임벡터를효과적으로탐색할수있기때문에, HEVC에서널리사용되는정수단위움직임추정방법이다. 그러나TZ sear-ch 알고리즘은그복잡한수행구조때문에하드웨어로구현되기에는어려움이따른다.본논문에서는기존의TZ search 알고리즘을수정하여하드웨어에서prediction unit (PU) 단위의병렬움직임탐색이가능하도록한새로운정수단위움직임추정을고안하였다.제안된알고리즘은zonal, raster 및 refinement 의세가지탐색과정으로이루어져있다. 각탐색과정을시작하면서, 제안된알고리즘은coding unit (CU) 내의모든PU에대해서기존TZ search 알고리즘의탐색점을수집한다. 다음으로, 수집된탐색점중중복되는모든탐색점을제거한뒤에움직임추정을수행하여모든PU에대한최적의탐색점을구한다. 실험결과, 기존의TZ search 알고리즘과비교하여, 제안된알고리즘은0.84%의 Bjøntegaard-Delta bitrate (BD-BR) 감소효과를나타내었으며, 계산복잡도또한54.54%만큼감소하였다.

주요어: High-Efficiency Video Coding (HEVC); Integer Motion Estimation (IME), TZ Search Algorithm, Advanced Motion Vector Prediction (AMVP)

학 번: 2015-22134