Test Zonal Search based on Region Label (TZSR) for Motion Estimation in HEVC

Iris Linck[(1)], Arthur Tórgo Gomez[(2)], Gita Alaghband[(1)]
[(1)]*Dept. of Computer Science and Engineering - University of Colorado Denver* - Denver, USA
[(2)]*CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasilia,* Brazil
iris.linck@ucdenver.edu, arthur.gomez@pq.cnpq.br, gita.alaghband@ucdenver.edu

***Abstract* – This paper presents a new complexity Reduction method for the diamond search pattern called TZSR based on region labels in HEVC/H.265 video coding. The solution introduces a new image region structure developed from a simplified version of blob coloring algorithm (image labeling) to HEVC/H.265 video coding. Regions are either whole or part of image objects and normally span several coding tree blocks that are produced during HEVC encoding. Our method executes a complete Diamond Search (DS) for the first block of each region in order to identify the motion vector direction among eight different directions in DS. The motion estimation (ME) for the rest of the blocks in the region will perform a modified DS where only one direction point for various distances will be tested in order to reduce the code complexity. Experimental results demonstrate that the speedup achieved in our solution surpasses the time spent in our blob coloring algorithm. Furthermore, TZSR achieves an average speedup of 42.61% for low delay (LD) configuration and 52.13% for random access (RA) in the encoding time compared to the original ME algorithm in HEVC reference software (HM-16.7) with overall gains in PSNR and bit rate around 18.67% and 0.1 under LD and 28.37% and 0.74 under RA respectively.**

Keywords—HEVC, motion estimation, inter prediction, diamond search, TZSearch, region label.

I. INTRODUCTION

The increasing demand in quality defined by Video coding standards, such as High Efficiency Video Coding (HEVC) [1], necessitate development of new algorithms and strategies to reduce their overall computational complexity. While HEVC provides improved compression ratios in comparison to previous standards with the same subjective image quality, it comes with an increased encoding computational complexity which may compromise the encoder operation in portable devices and in real-time systems especially for high-resolution videos [2].

One of the most time consuming processes in video coding is the motion estimation (ME) designed to exploit temporal redundancies among frames within a sequence. HEVC provides the square and the diamond search fast motion estimation algorithms in TZSearch pattern [3]. These algorithms return the best motion vector (MV) for the current block at a cost which is comparable to the full search algorithm but still remain sub-optimal in terms of motion estimation time. Research to improve the motion estimation algorithm and to optimize the test zonal search (TZSearch) in order to reduce its computing complexity and thus reducing the encoding time while preserving the quality and bit-rate is essential. Several fast search methods for integer ME have recently been reported, such as, TZSearch based on triangle and pentagon patterns [3], complexity reduction methods for fast ME in HEVC [4], and reducing the number of operations performed in the ME using two-stage ME [5]. Our goal is to reduce the number of search points in the diamond search (DS) in order to avoid calculating Sum of Absolute Difference (SAD) for all 52 possible search points per block (for a search range set to 64 pixels) as is currently done in TZSearch in HEVC which is computationally very time consuming.

In this paper we propose a Complexity Reduction Method for TZSearch in HEVC video coding based on image regions called TZSR. We introduce regions as a new and additional structure in HEVC to describe part (or all) of an object, a set of pixels such that no pixel belongs to the interior of more than one region. We create image regions for reference frames only. This takes place during encoding as soon as a frame is assigned as a reference frame in HEVC. Each reference frame points to its related image region. The image region is created based on a simple region growing method called blob coloring or region labeling algorithm [6]. During the motion estimation phase, TZSR will use region image information from reference frames to leverage the test zonal search for the diamond search pattern. During this enhanced search, a complete diamond search (DS) is performed only for the first block of the region in order to identify the best direction point. After that, blocks belonging to the same region start a DS in the same direction as the first block and try different distances. The new approach reduces the code complexity of the TZSearch significantly with gains in bit rate and PSNR in most test-cases and negligible loss in few cases. We tested TZSR in two different configurations in HEVC, low delay (LD) and random access (RA). It achieves an average speedup around 42.61% under LD and 52.13% under RA configurations in the encoding time compared to HEVC reference software HM-16.7 [7]. In both configurations the proposed method achieves an overall improvement of bit rate and PSNR of about 18.67% and 0.1 respectively under LD and 28.37% and 0.74 under RA.

The remaining of the paper is organized as follows: In Section II, we present an overview of the motion estimation process in HEVC CODEC; we describe details of our new method called TZSR in Section III. In Section IV, we will present our experiments, results, and analysis and finally, conclusions are presented in section V.

II. MOTION ESTIMATION IN HEVC

HEVC starts the encoding process by reading a group of pictures (GOP) from the input video. A GOP is a set of pictures or frames with an associated "picture order count" (POC) value

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

used to identify a frame in codec. Frames are partitioned into blocks and during ME process, attempt is made to predict motion vectors for the blocks in order to reduce the amount of information needed for video transfer. Frames of a GOP may be divided into a set of slices composed of a number of blocks. There are three types of slices: I, P or B. In I slices, blocks cannot be used for ME, i.e., cannot be predicted. In P slices, blocks can be predicted by unidirectional prediction (allow reference frames in one direction, i.e., either before or after the current frame); finally in B slices, blocks can be predicted by bidirectional prediction (allow reference frames in two directions, i.e. before and after the current frame). In order to partition the frames into blocks, HEVC divides a frame into square blocks of the same size (64x64 pixels). Each initial square block serves as the root of a first block partitioning quadtree structure, the coding tree, referred to as coding tree blocks (CTBs). The CTBs can be further subdivided along the coding tree structure into coding blocks (CBs) that may be processed in ME. Figure 1 shows a quadtree structure for one CTB.

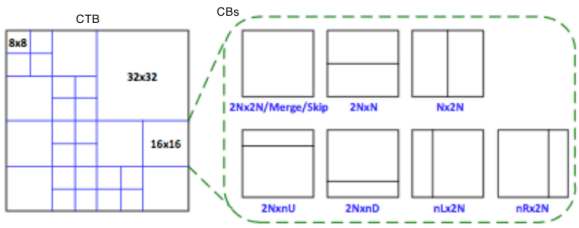

Fig. 1. HEVC quadtree structure – CTB partitioned in CBs.

Motion estimation (ME) in HEVC determines a best match of the current block by searching the reference picture within 64x64 pixels search windows using fast search algorithms. Result of this operation is a motion vector (MV). MVs obtained from the ME process determine the relative location of the best prediction block in the reference frame. The matching metric used is the Sum of Absolute Difference (SAD) that calculates the similarity between the current and reference blocks and it is one of the main causes of the large computational complexity in HEVC. The configuration of reference frames is defined in the GOP structure table in the configuration file used by HEVC encoder. The Figure 2 shows how a block in the current frame can be predicted from blocks in the reference frames from past and future.

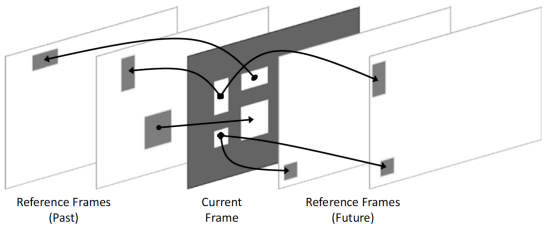

Fig. 2. Reference frame prediction.

In order to reduce search points for integer-pel ME, a three-step motion search strategy is used as illustrated in Figure 3.

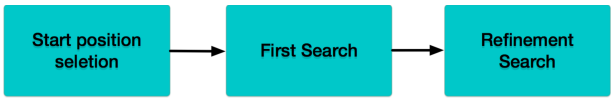

Fig. 3. Three steps ME in HEVC.

The three steps of the motion search (Figure 3) are described as follow:

- *Start postion selection*: HEVC choses the best motion vector prediction (MVP) available in order to use it as the starting search point in the next step. The best MVP is chosen by using SAD. The MVP corresponds to a previous predictor MV of a neighboring prediction block. In that case, the MVPs to be tested are the up, median, left, right and upper right predictors.
- *Fist Search:* In this step, HEVC performs a TZSearch using diamond or square patterns. These patterns are showed in Figures 4-a and 4-b respectively. Currently, diamond search is the default with search range set to 64 pixels in integer-pel accuracy. In this configuration, there are 52 search points in DS, where, we have four search points (or four direction) for distance 1, plus eight search points for distances from 2 to 64 each. Note that the distance grows in powers of two. Figure 4-a shows an example of DS pattern in a range of distances from 1 to 8. In this search, all the points in DS pattern are tested and the one with minimum SAD will be the best matched block. Additional raster search (Figure 3-c) is performed when the difference between the obtained MV and the start position is greater than 5 (iRaster = 5).
- *Refinment Search:* This step is a fine refinment of MV obtained from the previous step and it is performed in two rounds: The first round starts by performing a square or diamond search pattern around the best MV obtained from the previous step in order to obtain a refinment MV. After that, a second round of diamond or square search is perfomed by using the refinment MV (from the first round) as a start position for the search. The best MV from the second round is used as the best integer-pel MV.

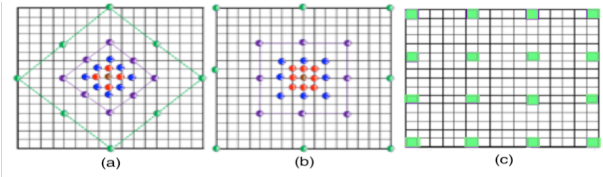

Fig. 4. Search patterns for TZSearch in HEVC: (a) diamond search, (b) square search, (c) raster search with iRaster=5.

III. Proposed Method

Motion vectors of the current block are usually correlated with the motion vectors of neighboring blocks in the current picture or in the earlier coded pictures. This is because neighboring blocks are likely to correspond to the same moving object with similar motion and the motion of the object is not likely to change abruptly over very small timeframes. Based on that premise, we assume that blocks belongin to the same

region have high probability to move in the same direction. This leads us to reducing the number of SAD calculation performed in the diamond search (DS) and, as demonstrated by our experiments, increasing the accuracy of the MV. Figure 5 shows all possible direction points within the DS. There are 8 directions numbered from 1 to 8; zero is the start position in DS.

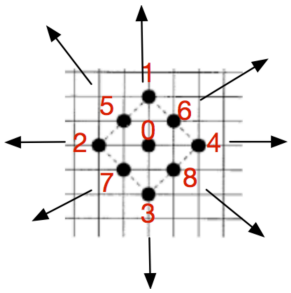

Fig. 5. Motion directions in the DS pattern for TZSearch.

The new TZSR architecture is shown in Figure 6. The diagram is simplified to a great extend to make the proposed architecture visible within HEVC. As described earlier, HEVC works with a GOP at a time, where some of the frames are used (and marked) as reference frames and maintained in the Reference Picture Set (RPS). In our modified diamond search enhanced with image regions model, a new module Parallel Blob Coloring (PBC) is added to generate additional information for each frame by identifying regions within reference frames. As described in Algorithm I, it generates image regions for reference pictures within each GOP. The region frames are stored in a region frame list, associated with the respective reference frame, to be used in the new ME. The new ME will execute TZSR using a DS pattern described in Algorithm II. The proposed complexity reduction method for ME assigns a specific motion direction to each region in a frame after performing a complete DS for the first block of each region. The idea is to use the same direction for the subsequent blocks belonging to a region in order to reduce the code complexity of a complete DS/block strategy in HEVC. Next we present the two new algorithms "Parallel Blob Coloring" (PBC) and "TZSearch based on regions" (TZSR) introduced in HEVC.

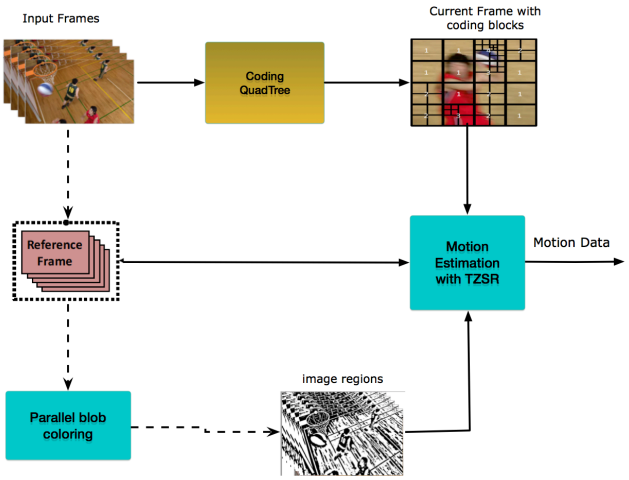

Fig. 6. Motion Estimation with TZSR in HEVC.

A. PBC - Parallel Blob Coloring Algorithm

Blob coloring is a region growing algorithm. The goal of region growing is to use image features to map individual pixels in an input image to sets of pixels called regions. Although perfect regions and boundaries are inconvertible, the processing to find them differ in applicability [6]. For our purpose, perfect regions are not necessary. We employ a simplified parallel blob coloring algorithm, PBC, which scans a binary image to identify connected groups of pixels with the same binary values and assigns them to a region. It is useful since once they are individually labeled, the objects can be separately manipulated.

The PBC algorithm introduced in HEVC is described in Algorithm I. The PBC is performed when a GOP is read from the input video and the frames marked as reference are put in the reference picture set.

ALGORITHM I. PARALLEL BLOB COLORING (PBC)

Read reference frames in a parallel loop.
Parallel For (i = $POC_{ref\_Initial}$ to POC_{ref_final}) do
  Let M_L = matrix of reference frame (luminance-Y).
  Let v = video resolution.
  Let p_b = 0 be the background pixel value;
  Let p_o = 1 be the object pixel value.
  Convert M_L into binary image (M_B)
  If (v >= 1280x720) /*high resolution*/
    AdaptiveThreshould (M_L, M_B, adaptive_threshold_gaussian_c, 77, 3)
  Else /*low resolution*/
    AdaptiveThreshould (M_L, M_B, adaptive_threshold_gaussian_c, 17, 3)
  End-if
  Let initial color k=1
  In a loop, scan M_B from left to right and top to bottom and execute the following steps:
    Let x_c = value of a pixel in a coordinate (x,y) in the scanned binary image M_B.
    If ($x_c \neq p_b$)
      Let x_u = upper neighbor of x_c.
      Let x_l = left neighbor of x_c.
      If ((x_c is the first pixel in M_B) or ($x_l = p_o$ and $x_u = p_b$)) then
        color (x_c = k)
        k = k + 1
      End-if
      If (($x_u = p_o$) and ($x_l = p_b$)) then color ($x_c = x_u$)
      If (($x_l = p_o$) and ($x_u = p_b$)) then color ($x_c = x_l$)
      If (($x_l = p_o$) and ($x_u = p_o$)) then color ($x_c = x_l$)
    End-if
  End-Scan
End-Parallel
End

PBC reads the reference frames within a GOP in a parallel loop (implemented using OpenMP). The next step is to convert reference frames (luminance component-Y) into a binary image using an OpenCV function "AdaptiveThreshould" based on Gaussian threshold. In this method, threshold values are

weighted sum of neighborhood values where weights are a Gaussian window. The "adaptive_threshold_gaussian_c" function has two parameters, block size and a constant C, set according to the video resolution. The block size is the size of pixel neighborhood used to calculate a threshold value for the pixel and C is a constant subtracted from the mean. For videos in Table I with resolution greater or equal 1280x720 the block-size is set to 77 while for lower resolutions, the block-size is set to 17. For all resolutions C is set to 3 which produces satisfactory distinction between background and objects in the binary image. According to our observations, the block-size should be greater for high resolution and smaller for low resolution. It affects the number of regions created for each reference frame. As the block size grows, the number of pixels belonging to a region increases and the number of total regions for a frame decreases, which is a good choice for high resolution videos in terms of MV accuracy. The binary image is scanned from left to right and top to bottom in order to identify connected regions. Pixels with value 1 are identified as object (p_o) while those with value 0 are background (p_b). The objective is identifying connected pixels (p_o) by using neighbor pixels from up and left positions in order to form regions. Each region will be assigned a different color k (also called label).

PBC is a revised version of the original Blob Coloring algorithm. It is designed not to maintain a table of color equivalences that assures each object has only one color. PBC scans the binary image only once; it is possible to assign more than one color or label for the same object, but not for the same pixels inside the object. This property was changed in order to create a region as part of an object and reduce complexity of PBC. The straightforward parallelism helps improve the resulting runtime and avoid penalizing HEVC encoding time. According to the experiments the average runtime for PBC algorithm is around 0.14s when encoding 150 frames. We observed that PBC is sensitive to the threshold chosen in the binary image conversion. It can influence the number of regions generated and consequently the improvements obtained later in the ME process. The PBC code was implemented using C++/openMP/OpenCV.

B. TZSR - TZSearch based on regions

The new TZSR, described in Algorithm II, is introduced in the TZSearch algorithm to leverage motion estimation in HEVC. TZSR identifies directions for regions generated by PBC described in Algorithm I, in order to use it for all blocks belonging to the same region. For each picture frame, a new data structure "labelList" will maintain best direction (Dir_{best}) for each region, and it's associated "picture order count" information for the reference (POC_{ref}) and current (POC_{cur}) frames. It is important to note that the direction assigned to a region label depends on both current and reference frames since a reference frame can be used to predict several current frames.

TZSR starts by finding the label of the first block in each region in order to find the best motion direction. The image region matrix (M_B) generated for the reference frame by PBC (Algorithm I) has all region label (L) information including their embedded blocks. The spatial locality (x_f) of a reference block is used to find its corresponding label in M_B. As described in Algorithm II, if a direction (Dir) is found in the labelList, we perform a unidirectional diamond search for the specified direction (Dir) and various distances (Dis) specified in the distance range. In this case, the unidirectional DS stops when $SAD_{previous} < SAD_{current}$ or we complete the search for distance (Dis) range size defined in the HEVC configuration file. This scenario for search to determine MV for a block results in least number of search points: at least two (for distance range of one) and at most seven (for largest distance range of seven). If a direction (Dir) has not been established for a region in the labelList (this is the initial case for each new region) then we need to conduct a complete DS for all 52 search points evaluating SAD of neighbor MVs in order to choose the best starting point for DS. The MV with minimum SAD is chosen. Furthermore, for each search point, the previous and current SADs are tested in order to save the direction for the best SAD (Dir_{best}) for inclusion in the labelList along with POC_{ref}, L, and POC_{cur}. This is the scenario performed for the first block of each region. The subsequent blocks in the region, will find a best direction value (Dir_{best}) in the corresponding labelList.

ALGORITHM II. TZSR - TEST ZONAL SEARCH BASED ON REGIONS

Evaluate neighbor MVs and choose the one with minimum SAD to be the starting point of DS.
Let mv_{start} = best neighbor MV
Let x_f = block starting position in reference frame
Let M_B = matrix of region image for the reference frame
Let POC_{cur} = Picture Order Count of current frame
Let POC_{ref} = POC of reference frame
Let L = region label of x_f correlated to the region image M_B
Find the direction **Dir** for region **L** in a list **labelList** where:
  Dir = labelList (POC_{ref}, L, POC_{cur})
If (Dir is found)
  For (Dist = 1; Dist <= SearchRange; Dist*=2)
    Execute DS using direction Dir varying Distance Dist
    If ($SAD_{previous} < SAD_{current}$)
      Exit the loop
  End-for
Else
  For (Dir = 1; Dir <= 8; Dir++)
  For (Dist = 1; Dist <= SearchRange; Dist*=2)
    Execute DS for all directions Dir varying distance Dist
    If ($SAD_{previous} > SAD_{current}$)
      Dir_{best} = Dir
  End-for
  End-for
  labelList(POC_{ref}, L, POC_{cur})= Dir_{best}
End-if
End

It is important to note that raster search and refinement search are not performed in the new method TZSR. Furthermore, when a reference frame is removed from the decoded picture buffer, occurrences belonging to that frame in

Scholarship CAPES - Brazil

the labelList are deleted which means that the reference frame will no longer be used to predict any current frame. The new TZSR algorithm is sequential and developed in C++.

IV. EXPERIMENTS

In order to guide tests of new developments for HEVC, the Join Collaborative Team on Video Coding (JCT-VC) group coordinated the development of a reference software encoder and decoder, colloquially known as HM and published the common test conditions (CTC) document [7]. CTC defines a set of four temporal configurations to be used with the HM reference software for tests and comparisons by researchers. The four configurations differ in terms of temporal predication from one another and they are: all intra (AI), low delay (LD), low delay P (LP) and random access (RA). For our experiments we use HEVC reference software HM-16.7 and two temporal configurations: LD and RA. In the reference software, raster search with iRaster = 5 and star refinement diamond search are enable. On the other hand, our new method does not use these additional searches. The goal of the experiments is to demonstrate that our TZSR can achieve better results than the reference software without using raster and refinement search.

A. Test Conditions

The two temporal configurations LD and RA of common test conditions (CTC) are used in our experiments. In LD only the first image in a GOP is encoded as instantaneous decoding refresh (IDR) pictures. These are pictures that contain only I slices; the remaining pictures are encoded as generalized P and B pictures (GPB). In the RA configuration, the first picture in a video sequence is encoded as an IDR while the remaining I pictures are encoded as non-IDR characterizing an open GOP. This means that frames outside the current GOP can be used as references. Furthermore, frames between two intra frames are encoded as B pictures. The video sequences used in our experiments described in Table I represent six HEVC common test sequences with different resolutions and frame rates. For each test sequence, the first 150 frames are encoded. The hardware platform used in these experiments is composed of an Intel Xeon E5-2650v2 Eight-Core 2.60 GHz, and 65 GB of system memory. The encoder has been compiled with GCC 4.9.0 and executed on CentOS Linux 7.2.

TABLE I. VIDEO CONFIGURATION

Video ID	Video Features		
	Sequence resolution	***#frames***	***Frame rate (fps)***
1	Racehorse (416 x 240)	150	30
2	BasketBallDrill (832x480)	150	50
3	BQMall (832x480)	150	60
4	SlideShow (1280x720)	150	20
5	BlueSky (1920x1080)	150	25
6	ParkScene (1920x1080)	150	24

B. Results and Analyses

TZSR was evaluated by measuring the encoding computational complexity reduction under LD and RA temporal configurations. The encoder performance was evaluated using the gains achieved in bitrate and the peak sign-to-noise (PSNR), a distortion metric used to evaluate the image quality. The evaluation parameters are based on the following equations:

$$\Delta T = \frac{T_{HM} - T_{prop}}{T_{HM}} \, x \, 100 \qquad (1)$$

$$\Delta BR = \frac{BR_{HM} - BR_{prop}}{BR_{HM}} \, x \, 100 \qquad (2)$$

$$\Delta PSNR = PSNR_{prop} - PSNR_{HM} \qquad (3)$$

In Eq. (1), ΔT represents the percentage of encoding time reduction achieved by TZSR, where T_{HM} and T_{prop} are the encoding time of reference software and proposed method respectively. In Eq. (2) ΔBR represents the gains in bit rate. Negative values indicate gains in compression while positive values indicate an increase in bit rate and thus a loss in compression. The BR_{HM} and BR_{prop} represent the bit rate of reference software and proposed method respectively. In Eq. (3) $\Delta PSNR$ is the difference between the PSNR from the proposed solution ($PSNR_{prop}$) and the reference software ($PSNR_{HM}$). Positive values indicate improvement in the image quality while negative values represents loss.

Table II presents results of all test cases corresponding to LD configuration and Table III shows the results for RA configuration.

TABLE II. SIMULATION RESULTS FOR TZSR COMPARED WITH HM-16.7 USING LOW_DELAY_MAIN CONFIGURATION

Video ID	Low_delay_main		
	ΔT(%)	ΔBR(%) [negative indicates gain]	Δ PSNR-YUV [positive indicates improved quality]
1	44.84	-9.45	0.06
2	47.21	-25.18	0.17
3	38.78	-13.28	0.07
4	44.41	2.32	-0.04
5	54.99	-59.51	0.43
6	25.43	-6.88	-0.08
Average	**42.61**	**-18.67**	**0.10**

According to Table II the new TZSR under LD configuration, obtains average gains in encoding time, bitrate, and PSNR around 42.61%, -18.67% and 0.10 respectively. Table III shows that TZSR under RA achieves better results than LD in all three evaluation parameters: average gain of 52.13% in the encoding time, average decrease in bitrate of -28.37%, and average improvement in PSNR of 0.74. We obtained the best results for Video 5 in both configurations compared to other test sequences in the same configuration. Video 5 has an encoding speedup time of 54.99% in LD and 74.89% in RA. Also, the bitrate and PSNR are improved by -59.51% and 0.43 in LD and -60.48% and 1.92 under RA configuration. Videos 4 and 6 have a negligible loss in the PSNR under LD configuration -0.04 and -0.08 respectively,

while we obtain gains in their encoding time by 44.41% and 25.43%. On the other hand, videos 4 and 6 have a better performance in RA with a decrease in encoding time of 51.70% and 34.46% respectively, a reduction in their bitrate by -2.21% and -33.65%, and a negligible loss in PSNR of -0.09 for video 4 and an improvement of 0.48 in video 6. Video 4 has an increase in bitrate under LD by 2.32% while under RA we observe a decrease bitrate of -2.21%.

TABLE III. SIMULATION RESULTS FOR TZSR COMPARED WITH HM-16.7 USING RANDOM_ACCES_MAIN CONFIGURATION

Video ID	Random_access_main		
	ΔT(%)	ΔBR(%) [negative indicates gain]	Δ PSNR-YUV [positive indicates improved quality]
1	54.39	-30.84	1.04
2	45.32	-15.91	0.28
3	52.05	-27.12	0.79
4	51.70	-2.21	-0.09
5	74.89	-60.48	1.92
6	34.46	-33.65	0.48
Average	**52.13**	**-28.37**	**0.74**

Results in Tables II and III show that we obtain better results under RA configuration for all evaluation parameters compared to the LD configuration. Under RA, TZSR shows gain in bitrate for every case, and under LD we observe only one loss in video 4. There are improvements in all PSNR in RA, except for video 5, while under LD the new method resulted in two negligible losses (video 4 and 6). Both configurations achieve a significant decrease in the encoding time.

The complexity reduction in HEVC is achieved by the fact that associating a direction point to a region, represented as part of an object, can lead to a significant reduction in SAD operations and hence a significant reduction in computational complexity as showed in the results. We observed that associating a MV to a region or part of an object can further improve bitrate and PSNR. Another fact that contributes to decrease the encoding time in HEVC is that TZSR does not perform raster search and refinement search in order to improve MVs. It is important to note that as our experiments show, the gains in the complexity reduction outweighs the added PBC time. Furthermore, PBC time can be mitigated by the number of reference frames arrivals and the number of cores available for parallel processing. This further supports investigating the possibility of adding hierarchical object structures to HEVC encoding to reduce the motion prediction complexity when applicable.

V. CONCLUSION

This paper presents a new TZSearch based on region label applied to ME in HEVC called TZSR. The purpose of the new method is to use region labels in TZSearch in order to leverage ME in HEVC. The method is developed in two parts, the first part introduces a parallel blob coloring (PBC) algorithm for reference frames in order to create region labels to be used in ME process. The second part occurs in ME process, where our new TZSR is executed using DS pattern for the first search and achieves better results than the reference software without the need to perform raster search and refinement search. TZSR can be applied for square or other patterns although it has not been the focus of our work. Experiments were conducted to compare TZSR with the reference software HW-16.7 under LD and RA configurations. The results show that the proposed algorithm achieves significant decrease in computational complexity in both configurations, LD and RA, for HEVC encoder. It was observed that TZSR is sensitive to the quality of regions generated in PBC. That is, if a region has a perfect number of blocks of pixels to identify an entire part of an object, then the motion direction for the region will be more accurate. This directly impacts the accuracy of the MVs and improvement of the encoding time, bitrate, and PSNR. Based on that, further studies will be made in order to improve PBC. The important factor is to do so balancing the gains without increase in computational complexity in HEVC.

REFERENCES

[1] ITU-T, "Recommendation ITU-T H.265," ed: ITU-T, 2015.

[2] G. Correa, P. Assuncao, L. Agostini, and L. A. d. S. Cruz, *Complexity-Aware High Efficiency Video Coding*: Springer Publishing Company, Incorporated, 2015.

[3] E. Jaja, Z. Omar, A. A.-H. A. Rahman, and M. M. i. Zabidi, "Efficient Motion Estimation Algorithms for HEVC/H.265 Video Coding," in *Information Science and Applications*, J. K. Kim, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 287-294.

[4] P. Nalluri, L. N. Alves, and A. Navarro, "Complexity reduction methods for fast motion estimation in HEVC," *Image Commun.,* vol. 39, pp. 280-292, 2015.

[5] G. Cebrián-Márquez, C. C. Chi, J. é. L. Martínez, P. Cuenca, M, C. l. M. x, *et al.*, "Reducing HEVC encoding complexity using two-stage motion estimation," in *2015 Visual Communications and Image Processing (VCIP)*, 2015, pp. 1-4.

[6] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs NJ: Prentice-Hall, 1982.

[7] C. Rosewarne, M. N. B. Bross, K. Sharman, and G. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 16 (HM16) Improved Encoder Description Update 3," JCT-VC, Warsaw - Poland, June 2015.