ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

ΚΑΤΑΤΜΗΣΗ ΣΕ TILES ΚΑΙ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΚΩΔΙΚΟΠΟΙΗΣΗΣ VIDEO: ΟΙ ΠΕΡΙΠΤΩΣΕΙΣ ΤΩΝ ΠΡΟΤΥΠΩΝ HEVC ΚΑΙ AV1

Πανάγου Ναταλία

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Επιβλέπουσα
Κοζύρη Μαρία, Επίκουρη καθηγήτρια του Τμήματος Πληροφορικής,
Πανεπιστήμιο Θεσσαλίας

Λαμία, 2019

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

INFORMATICS AND COMPUTATIONAL BIOMEDICINE

THE IMPACT OF TILES ON VIDEO CODING PERFORMANCE: A CASE STUDY ON HEVC AND AV1 VIDEO CODING STANDARDS

Panagou Natalia

Master thesis

Supervisor
Koziri Maria, Assistant Professor at the Department of Computer Science, University of Thessaly

Lamia, 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ:

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

ΚΑΤΑΤΜΗΣΗ ΣΕ TILES ΚΑΙ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΚΩΔΙΚΟΠΟΙΗΣΗΣ VIDEO: ΟΙ ΠΕΡΙΠΤΩΣΕΙΣ ΤΩΝ ΠΡΟΤΥΠΩΝ HEVC ΚΑΙ AV1

Πανάγου Ναταλία

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κοζύρη Μαρία

Λαμία, 2019

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο [«ΚΑΤΑΤΜΗΣΗ ΣΕ TILES ΚΑΙ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΚΩΔΙΚΟΠΟΙΗΣΗΣ VIDEO: ΟΙ ΠΕΡΙΠΤΩΣΕΙΣ ΤΩΝ ΠΡΟΤΥΠΩΝ HEVC ΚΑΙ AV1»] αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η ΔΗΛΩΝ/-ΟΥΣΑ

Ημερομηνία

Υπογραφή

ΚΑΤΑΤΜΗΣΗ ΣΕ TILES ΚΑΙ ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΚΩΔΙΚΟΠΟΙΗΣΗΣ VIDEO: ΟΙ ΠΕΡΙΠΤΩΣΕΙΣ ΤΩΝ ΠΡΟΤΥΠΩΝ HEVC ΚΑΙ AV1

Πανάγου Ναταλία

Τριμελής Επιτροπή:

Κοζύρη Μαρία, Επίκουρη Καθηγήτρια του Τμήματος Πληροφορικής, Πανεπιστήμιο Θεσσαλίας (Επιβλέπουσα)

Λουκόπουλος Αθανάσιος, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική, Πανεπιστήμιο Θεσσαλίας

Σταμούλης Γεώργιος, Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Table of Contents

Abstract

The era of 4K and 8K resolutions in capture and display devices brought along a ton of information that should be stored and transmitted even in real time. Since video consumption occupies a great amount of network traffic, it is of vital importance to efficiently compress the original video signal as much as possible. The advent of even higher resolutions posed the need of even greater compression and gave rise to the development of new video coding standards. High Efficiency Video Coding (HEVC) and the recently launched AV1 are the most important examples of new generation video codecs. Aiming to replace the widely used in the era of FullHD, H.264/AVC, both standards promise even higher compression ratios, for the same visual quality, that come at the expense of a greater computational complexity. In order to cope with this situation, special attention was given to the development of parallelization techniques that can be incorporated in various levels of video compression. In a coarse-grained level, a frame can be split into independent, trivially parallelizable areas that can be combined with finer grained levels, like transform and motion estimation. In the present thesis, the impact of tile parallelization on video coding performance is studied. Tile parallelization is a coarse-grained parallelization technique that was firstly introduced by HEVC and is also adopted by AV1. The contribution of the current thesis is to characterize the impact of this technique in AV1 and compare it against HEVC. Parts of this work were presented in [1].

Chapter 1: Introduction

In today's world, video has a prominent role in people's lives. From various devices, like laptops, smartphones, tablets, etc., people like to record and share videos, stream and watch movies or chat with each other using real-time video applications. In all of the aforementioned cases, people like to enjoy videos in the higher quality possible and also high download or live streaming speeds. In order to efficiently store and share a video, compression needs to be applied to the huge amount of original video data. As a result, the final compressed video is free of redundant information, has a certain file format that is dictated by the video coding standard used to compress it and its file size is significantly smaller compared to the original. The main objective of video coding standardization efforts is to achieve a low bit rate, for storage and transmission purposes, while maintaining visual quality. Confronted with the advancements in the resolution of capture and display devices, people can now enjoy 4K and 8K resolutions to their screens, video standardization vendors launched new video coding standards in order to manage the great amount of data that comes with such high resolutions. After the era of FullHD, when the H.264/AVC video coding standard was the dominant codec, new generation standards were developed, namely High Efficiency Video Coding (HEVC) and AV1, promising higher compression ratios for the same visual quality at the expense of a greater computational complexity.

High Efficiency Video Coding Standard (HEVC) [2], was developed by the Joint Collaborative Team on Video Coding (JCT-VC), a cooperation between the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The standard was formally published by ITU-T in 2013 and in the same year by the ISO/IEC. The aim of the new standard's development was the replacement of its predecessor, H.264/MPEG-4 Advanced Video Coding (AVC) [3], which dominated the DVD era. Since H.264 was originally designed for lower than Ultra High-Definition (UHD) video content and the demands for higher resolutions were expected to increase dramatically, HEVC was designed to offer 50% bit rate savings compared to HEVC, for the same visual quality [4]. However, HEVC is a royalty-bearing codec, something that initially obstructed the relevant industry of adopting it. Recognizing the importance of the open software and the need for continuous advancements in compression performance, Alliance for Open Media (AOM), a consortium between Amazon, Cisco, Google, Intel, Microsoft, Mozilla and Netflix was formed in 2015, with the objective to launch a new, royalty-free and competitive to HEVC, standard. As a result, on March 2018, AV1's bitstream format [5] was consolidated and on June 2018 the first version 1.0.0 of the codec was released.
In anticipation of the stabilized version of AV1, and with the new Versatile Video Coding (VVC) [6], in the making, by the VCEG group, several comparison studies have been conducted between HEVC and AV1. However, regardless of the findings that declare HEVC [7] or AV1 [8] as the winner, in terms of bit-rate savings or encoding quality, the common ground that studies agree on, is the fact that AV1 has an extremely low encoding speed. This testifies that there is still a lack of speed optimizations to the newly introduced AV1.

In order to manage the significant computational complexity of AV1, parallelization techniques can be applied on various levels of the encoding process, like Group of Pictures (GOP) level [9], frame level [10], frame region level [11], block level e.g. in motion estimation [12] or filtering and transformations [13]. Such techniques had been already introduced in H.264 and HEVC in order to accelerate the encoder, namely Slices, Wavefront Parallel Processing (WPP) and Tiles. In the context of this thesis, the case of parallelization using tiles, a technique firstly introduced by HEVC and also adopted by AV1, is studied. Tile partitioning is a frame level technique that splits the frame into separate rectangular regions that can be trivially parallelized, since dependencies are broken across tile boundaries, something that can also affect coding efficiency. While there are many works on evaluating the impact of tile partitioning in HEVC,

e.g. [14], that is not the case for AV1. In this work, the effects of tile partitioning in AV1's coding efficiency are studied.

The rest of this thesis is organized as follows: Chapter 2 gives a brief overview of the coding tools incorporated in HEVC and AV1 video coding standards, Chapter 3 discusses the tile partitioning feature in both codecs, while Chapter 4 evaluates the impact of tile partitioning. Finally, Chapter 5 summarizes the conclusions of this work.

Chapter 2: Overview of Video Coding Standards

2.1 General Hybrid Encoding Scheme

Video compression can be defined as the process of condensing a set of video data into a smaller number of bits. Since uncompressed, namely "raw" video data take up a significantly large amount of space, compression of the original video signal is vital for practical storage and transmission of digital video. The process of compression involves a set of an encoder, that performs the compression of the original data before storage or transmission, and a decoder that converts the compressed bitstream into a representation of the original video data. This pair of encoder/decoder is referred as a CODEC and is depicted in Figure 1.

Figure 1. enCOder/DECoder (CODEC)

During compression, redundant components of the original video signal, components that are not necessary for a faithful reproduction of video data, are removed. There are two types of data compression, namely *lossless* and *lossy*. In the first case, statistical redundancy of data is used for compression and the decompressed signal is identical to the original. A lossy compression exploits subjective redundancy, which means that data that do not significantly affect viewer's perceptual quality are removed, while the decoded signal is not identical to the source signal. A lossy compression is the most effective way to achieve a high compression of a video stream and the majority of video codecs exploit both *temporal* and *spatial* redundancy of video data. In temporal domain, there is usually a high correlation between successive frames, especially when the frame rate is high, while in spatial domain there is a high correlation between neighboring pixels in the same frame.

For the majority of video codecs and since the H.261 video coding standard, a generic compression model, that combines inter prediction in order to exploit temporal dependencies, intra prediction for exploitation of spatial dependencies and transform coding prediction error to exploit even more those spatial dependences, is adopted. This coding scheme is called *hybrid* and while its structure has not changed, the algorithms of each building block have been refined and optimized during the last 25 years. In hybrid coding scheme, redundant information is wiped out by performing prediction and transformation of the prediction error. Additionally, applying quantization after transformation, irrelevant information can be removed. The hybrid coding scheme is depicted in Figure 2.

The steps of this hybrid encoding algorithm are the following:

1. The frames of a video sequence are fed as input to the encoder.
2. A prediction signal is generated in the encoder from information that is available and this prediction is subtracted from the original signal.
3. The resulted prediction error or residual is transformed, quantized and finally encoded into the bitstream. The parameters that the decoder needs in order to reproduce the prediction signal are also encoded and fed into the bitstream.

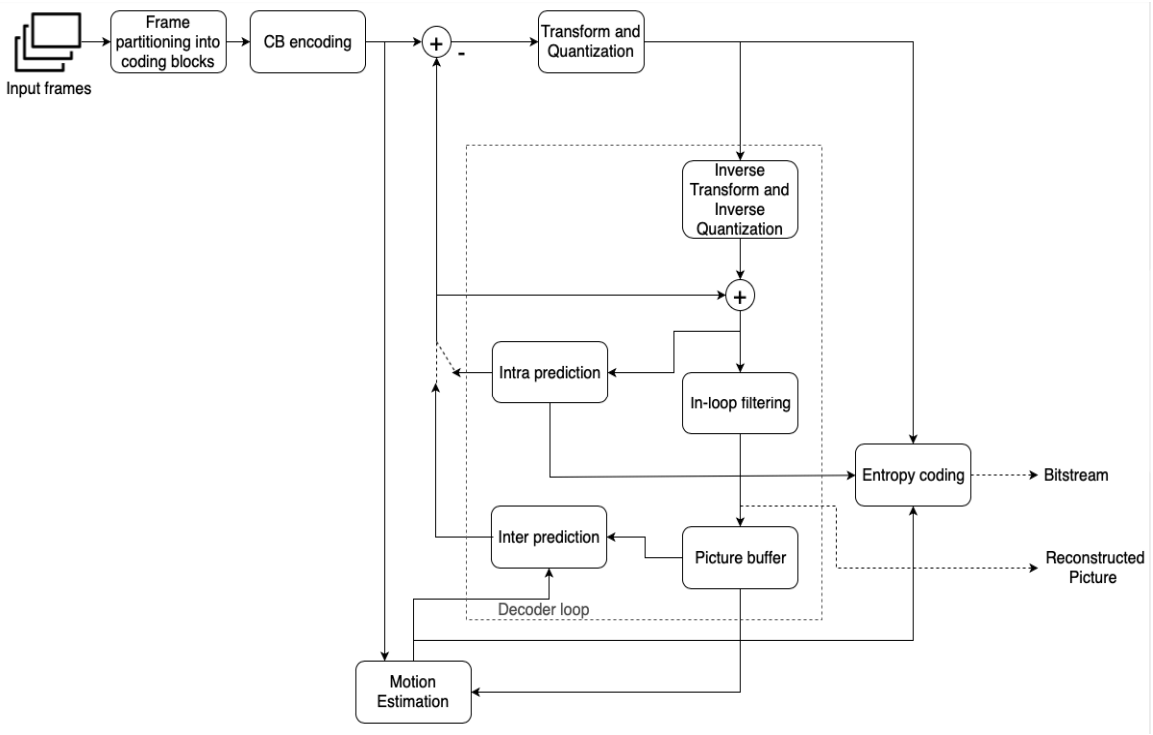

Figure 2.General Hybrid Encoding Scheme

In order to achieve an error-free set of encoder and decoder, the decoder's structure is incorporated in the encoder. In that sense, frames are encoded, decoded and then kept into a picture buffer for future referencing. During encoding, the encoder can use decoded pictures in order to perform *inter* prediction. In that way it is ensured that the encoder and the decoder use identical pictures for prediction and reconstruction, respectively. Additionally, the encoder can utilize neighboring, already coded, samples in order to perform *intra* prediction. The first frame of a video sequence is always intra encoded, since there are no previously encoded pictures preceding it. The rest of the frames can be encoded using intra or inter prediction and the kind of prediction that will be used is decided in a rate-distortion sense.

Before the encoding process starts, the input picture is partitioned into non-overlapping blocks which constitute the basic processing units and are processed in a raster scan order. The prediction signal is constructed by the summation of the reconstructed prediction error and the available prediction, and the signal is then processed by the *loop filter.* When the full picture is processed, it is available to the decoder and it is also stored in the decoded picture buffer in order to be used for inter prediction. During inter prediction, the currently processed block is predicted using motion estimation. During motion estimation stage, the best prediction for that block is searched in previously reconstructed pictures. On the other hand, the aforementioned intra prediction utilizes samples from neighboring reconstructed blocks in order to predict the current block.

The described encoding flow is adopted by both HEVC and AV1 video coding standards. In the subsequent sections, the techniques utilized in each component of the hybrid coding scheme, for both standards, are described in detail.

2.2 HEVC Video Coding Techniques

Group of Pictures (GOP)

In HEVC, frames within a video sequence are divided into groups called Groups of Pictures (GOPs). In this way, encoding order of frames can be different form display order in order to allow performing prediction referring to future frames. In HEVC two kinds of prediction structures are employed, a simple and a hierarchical structure. In the case where simple prediction structure is employed, e.g. the IPPP structure where the first picture is intra-coded while the rest are inter-coded, only previous frames can be utilized as a reference. On the contrary, in hierarchical prediction structure, prediction from future frames is allowed, forming a more accurate prediction. An example of a hierarchical GOP structure is illustrated in Figure 3. In the depicted GOP structure, the display order of frames goes from left to right, while the numbers, indicate the encoding order. The pictures that reside in the lowest layer, are called key pictures and are the ones that indicate where a GOP starts and can be coded either as I-frames or B-frames, while the remaining pictures are coded as B-frames. The I, P and B frames will be explained later.

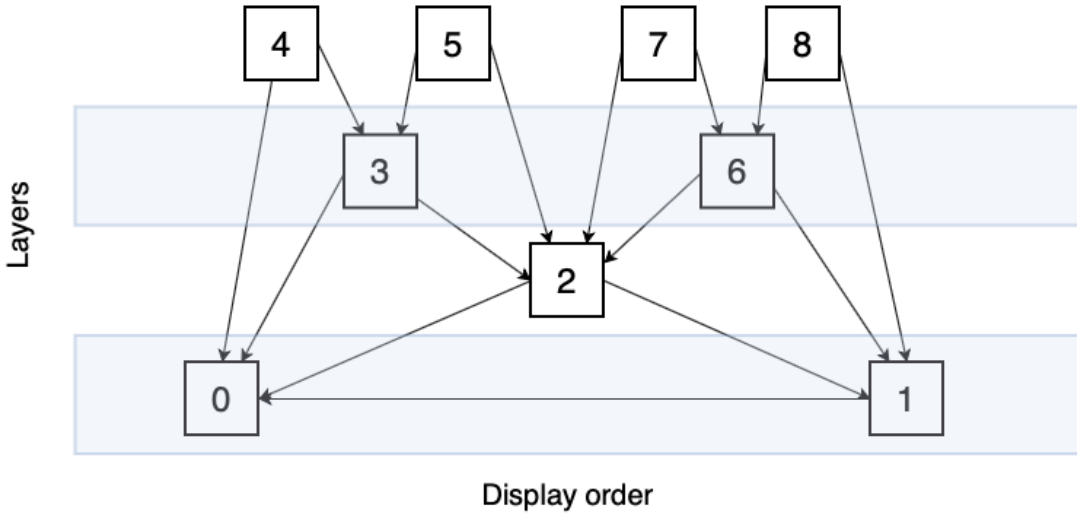

Figure 3. Hierarchical GOP of size 8

Picture Partitioning into Coding Tree Units (CTUs)

In HEVC each picture is partitioned into non-overlapping blocks, called CTUs, that are of maximum size 64x64 samples and constitute the basic processing units of the standard. Each CTU consists of luma and chroma Coding Tree Blocks (CTBs) and associated syntax elements that cover an area of L x L samples and L/2 x L/2 samples respectively. The value of L can vary between the values of 16, 32 and 64. In contrast with the basic processing unit in H.264, called macroblock (MB), which is of maximum size 16 x 16 samples, in HEVC larger partitions (64 x 64) are incorporated which are proved to be more beneficial for high-resolution video sequences. Another major difference between HEVC and H.264/AVC standards is that HEVC utilizes variable-size CTBs within a CTU and decides upon different modes for each partition. In contrast,

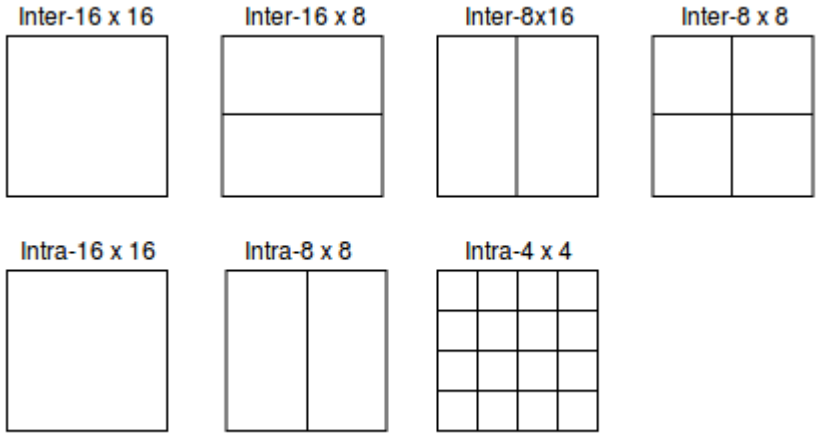

Figure 4. MB partitioning modes in H.264/AVC

in H.264 the decision upon intra or inter-picture coding is made in a macroblock level. For inter prediction, H.264 supports four partitioning modes (Inter-16 x 16, Inter-16 x 8, Inter-8 x 16, Inter-8 x 8), while for intra prediction, three modes (Intra-16 x1 6, Intra-8 x 8, Intra-4 x 4) are supported. The supported modes are depicted in Figure 4.

In HEVC the decision for prediction mode is not made only in the CTU level, since deciding for such large areas could harm coding efficiency in a rate-distortion sense. On the contrary, even smaller than 16 x 16 divisions are allowed from the standard. In order to achieve a lager variety of partitions, another processing unit, the Coding Unit (CU), is introduced in HEVC. Utilizing a quadtree structure, a CTU can be split into multiple CUs, where a CU encloses a luma Coding Block (CB) and two chroma CBs. The size of the CU can vary from 8 x 8 samples to the size of the CTU and is the unit upon the decision for inter or intra prediction is made. After that, a CU can be further split into so-called Prediction Units (PUs), which have a maximum size up to the size of the CU and a minimum size up to 4 x 4 samples. For intra and inter prediction, different partitions are allowed. For intra prediction, M x M and M/2 x M/2 partitions are supported, while for inter prediction, M x M, M/2 x M/2, M x M/2, M/2 x M, M x (M/4), M x (3M/4), (M/4) x M and (3M/4) x M. The aforementioned partitions are depicted Figure 5. The lower partitions are called Assymetric Motion Partitions (AMP) and are applicable only when M is equal to 16 or higher. Furthermore, for prediction error coding, Transform Units (TUs), consisting of luma and chroma Transform Blocks (TBs) that can only be square-shaped, and their size can vary from 4

x 4 to 32 x 32 samples are specified. In Figure 6, an example of partitioning a CTB into CBs and TBs along with the corresponding quadtree is illustrated, where the dotted lines represent the TBs. It is obvious that since a TB can only be square-shaped, multiple PBs can be included in a TB. This is an innovation introduced by HEVC, since it wasn't observed in previous standards.

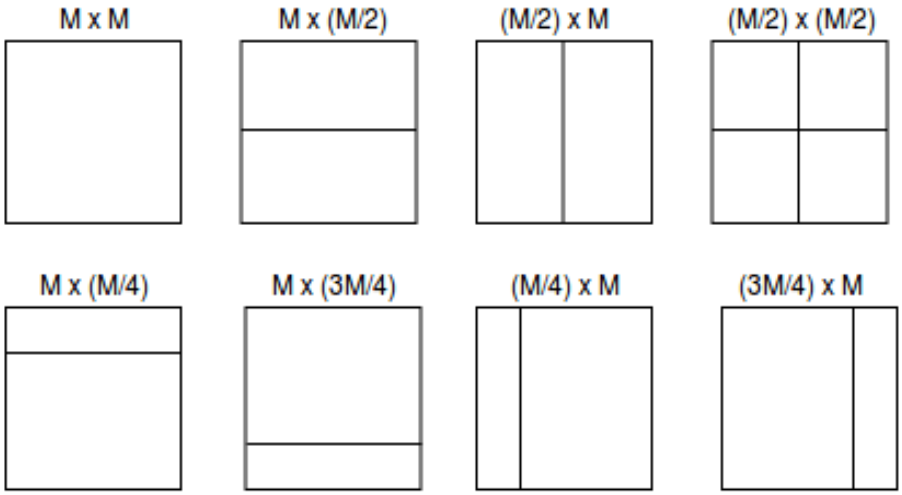

Figure 5. Modes for splitting a Coding Unit (CU) into Prediction Units (PUs) in HEVC

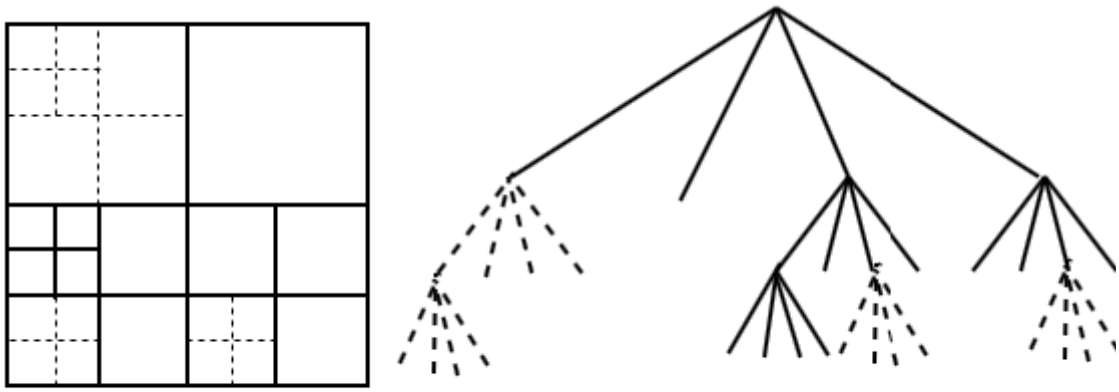

Figure 6. Illustration of a CTB divided into CBs (solid lines) and TBs (dotted lines) (on the left) and the corresponding quadtree (on the right)

Intra and Inter Prediction

After a picture is partitioned into square-shaped blocks, namely Coding Units (CUs), the first step of CU coding in a hybrid coding scheme is CU prediction. The procedure of predicting a CU consists of finding the most similar block among the surrounding to the current in order to serve as a prediction. This block can be found within the same frame, namely *intra prediction*, or in neighboring frames, namely *inter prediction*. Depending on the modes that each frame can utilize, three types of frames are defined in HEVC: I (Intra), P (Predicted) and B (Bidirectional). In I frames, only intra prediction is considered, where in P and B frames, both intra and inter prediction are allowed. More specifically, in P frames, the prediction is formed by utilizing frames that precede the current in chronological order, while for B frames both previous and following frames can be used.

Intra Prediction

In HEVC, a set of 35 modes, that utilize samples from neighboring reconstructed blocks and can be classified in two distinct categories, are provided for intra prediction. The first category consists of 33 angular prediction modes that are linear interpolations of the pixel values in the

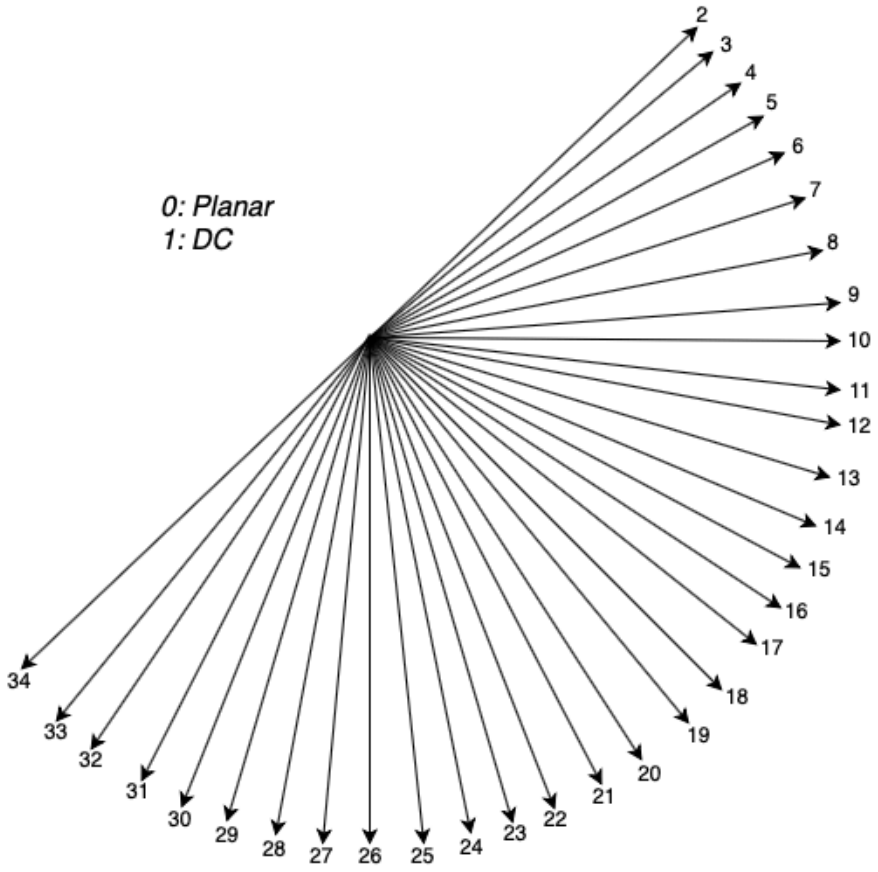

Figure 7. Intra Prediction Directions in HEVC

directions depicted in Figure 7 and are used for modeling structures with directional edges. An example of directional mode 31 is also depicted in Figure 8. In the second category belong the so-called Planar and DC predictions that are utilized in smooth image content prediction. In the case of Planar prediction, a two-dimensional linear interpolation of the available neighboring pixel values is used to form the prediction within a PU, while in the case of DC intra prediction, the mean value of the neighboring pixels is utilized.

In contrast to H.264/AVC, in HEVC a process of reference sample substitution is incorporated that allows the standard to utilize the whole set of intra prediction modes even if not all of the

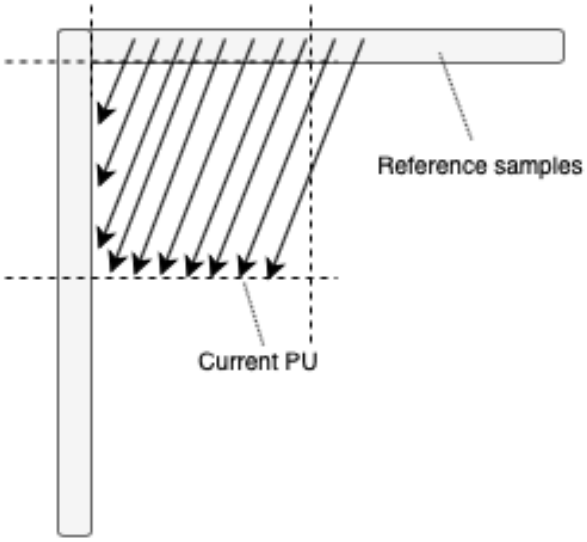

Figure 8. Example of directional mode 31

neighboring reference samples are available. For example, in the case that a block resides in picture, tile or slice edges, or the case that a reference sample belongs to an inter-predicted PU and considered unavailable in order to avoid error propagation from possibly erroneous receiving and reconstruction of previous frames. For reference sample substitution, when none of the neighboring samples are available, each sample is substituted by a nominal average value for a certain bit depth, e.g. 128 for 8-bit depth, while the available reference samples are used for substitution when there is at least one reference sample. More specifically, the available reference samples are scanned in a clock-wise manner and the last available sample is used to substitute the unavailable reference samples. An example of the process of reference substitution is given in Figure 9. Additionally, in order to improve the quality of the prediction, an adaptive pre-filtering, that depends on the selected intra mode, is applied to the reference samples.

In HEVC, the prediction of a block takes place on the PU level. Since HEVC utilizes a significantly larger number of modes compared to H.264/AVC, the number of Most Probable Modes (MPM) is extended to three, in contrast to one MPM in H.264. The selection of those three MPMs for the current PU, is based on the modes of the above and the left PUs and the third mode is decided according to the other two modes. For chroma samples, the selection is performed between only five modes, namely Intra_Planar, Intra_Angular (vertical), Intra_Angular (horizontal), Intra_DC, and Intra_Derived, where Intra_Derived indicates the case where for the chroma PU, where the same mode as that for the luma PU is chosen. In order to indicate the intra coding mode for a certain luma PU, Sum of Absolute Transformed Differences (SADT) is applied between prediction and original samples in order to reduce the possible candidate modes. As a result, the number of modes that enter full Rate-Distortion Optimization is reduced. Furthermore, the standard specifies that eight modes will be tested for 4x4 and 8x8 PUs and three for all the other PU sizes. For the final set of candidates, as well as the candidates that are included to the

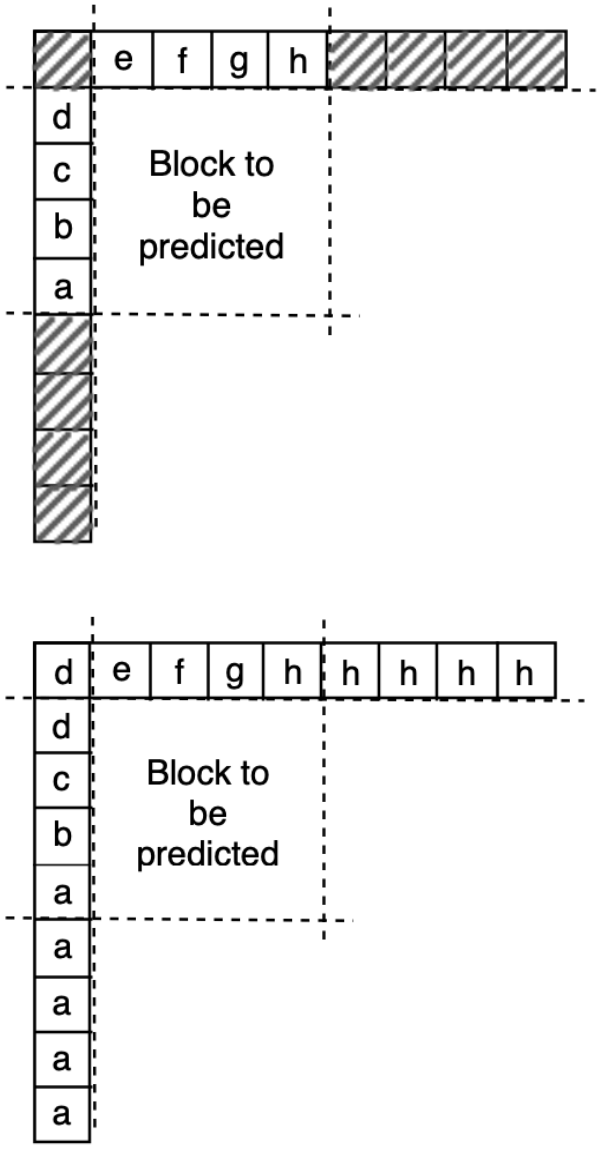

Figure 9. Reference sample substitution process: (a) before substitution (b) after substitution

MPMs, full RDO is performed and the mode that minimizes the cost in a rate-distortion sense. For chroma PUs, RDO is performed for all five modes indicated by the standard.

Inter Prediction

While intra-picture prediction, makes use of the spatial correlation between neighboring samples within a frame, inter-prediction exploits the temporal correlation between different frames. The exploitation of the temporal redundancy between frames in order to derive a prediction for a block of samples is called Motion-Compensated Prediction (MCP) during which, a block that serves as a predictor of the current block is searched within previously decoded frames, referred as reference frames. An example of inter-prediction in HEVC is depicted in Figure 10. The position of that block (the predictor), is indicated by a motion vector (Δx, Δy). where Δx indicates the horizontal displacement of the predictor, relative to the position of the current block, while Δy indicates the vertical displacement. Reference frames, along with motion

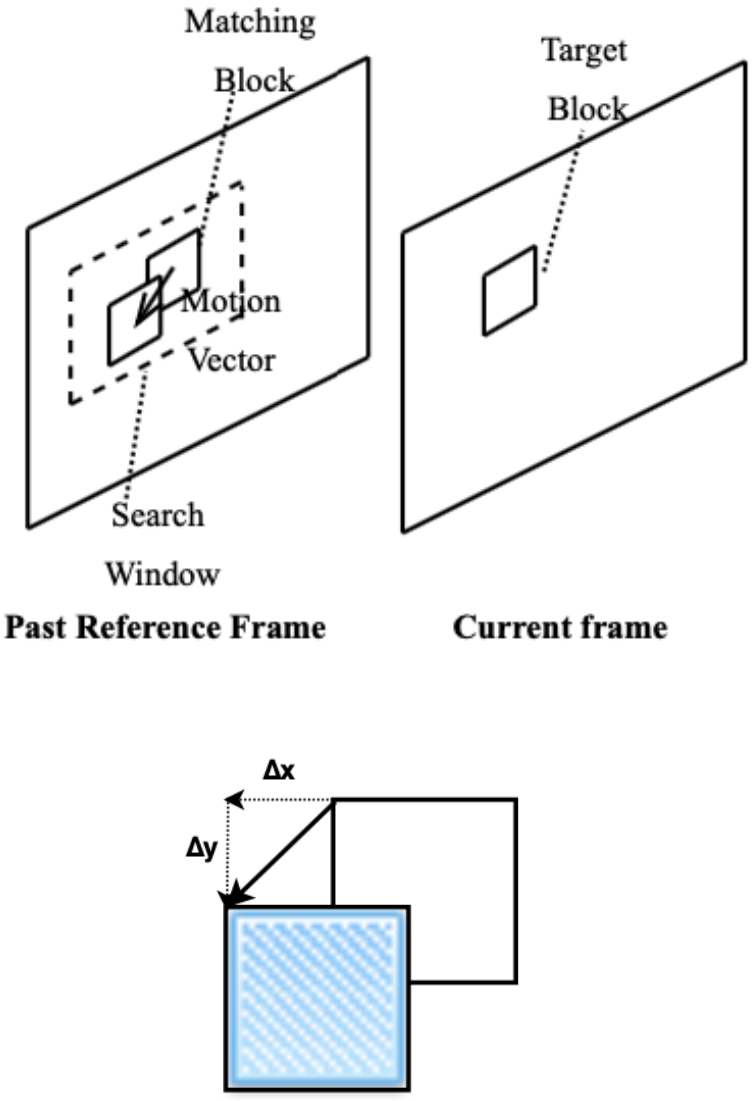

Figure 10. Inter-picture Prediction in HEVC

vectors are referred as motion data. In order to capture in a greater accuracy continuous motion, fractional sample accuracy of motion vectors is used in HEVC and in this case, the reference picture is interpolated in order to acquire the prediction signal.

In HEVC, two kinds of inter prediction methods are allowed, namely *uni-prediction,* Figure 11, and *bi-prediction,* Figure 12. In the case of uni-prediction, one block is used to serve as a predictor, while in bi-prediction, two blocks are used, possibly from different pictures that are saved in two separate lists, in order to acquire the final MCP. By default, the average of the two MCPs is used, but weighted prediction can also be utilized. In order to reduce complexity, PUs with a size of 8x4 and 4x8 samples are not allowed to use bi-prediction. The process of defining the motion data in the encoder is called motion estimation and since it is not defined by video coding standards, different implementations can be utilized.

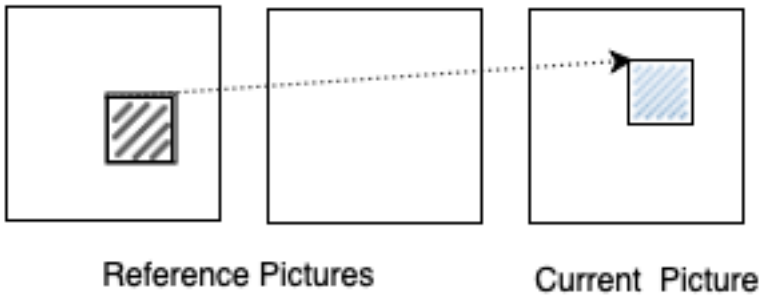

Figure 11. Uni-prediction method

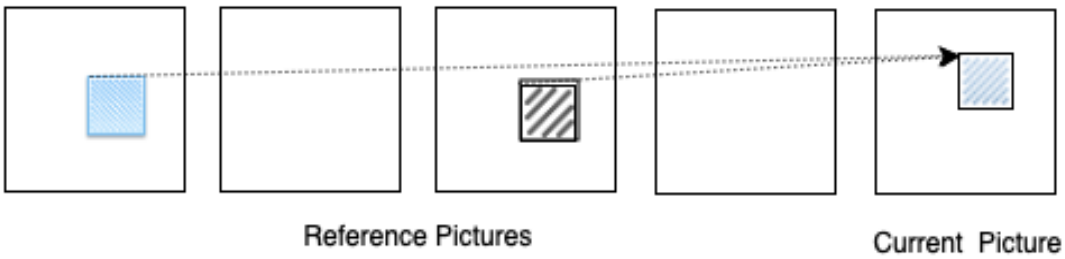

Figure 12. Bi-prediction method

For deriving the best match of a target block, a search window is defined, where full search block matching is performed. Although this method is very accurate, it can also be very time consuming. To reduce the huge computational complexity of performing a full search, a fast integer pixel motion estimation method, called *TZ search* is adopted in HEVC. TZ search algorithm, Figure 13, efficiently combines diamond (Figure 14) , square (Figure 15) and raster search methods and has four steps:

1. *Initial search center:* In the first step, a set of search centers is defined that includes the MV derived from median search, the MVs of the left, up and upper right position and also MV at (0,0) position. Among them, the one that results in the minimum block distortion, called MBD point, is chosen as the initial search center.

2. *Diamond or square search:* The search range and the search pattern, that can be diamond or square, are defined in this step. The search is performed with different stride lengths that increase from 1 to the defined search range, in multiples of two. The process of searching stops if the MBD point is found at the initial search center, while a 2-point search is performed if the distance between the MBD point and the initial center is 1.
3. *Raster search:* In this step, if the distance between the current center and the optimal point derived from the previous step, is 0, the search stops. Otherwise, if it is larger than a predefined threshold, called iRaster, then raster search is applied, and the stride length of the search is set equal to the value of iRaster.
4. *Raster/Star refinement:* The point derived from step 3 is used as the starting point. In raster refinement, 8-point diamond or square search is applied, and the stride length is set to half of the best distance. Both stride length and search center are updated in each round. In star refinement, the starting point is updated in each round and the process is similar to that of the initial center process. Both refinement processes stop until the best distance becomes greater than 1.

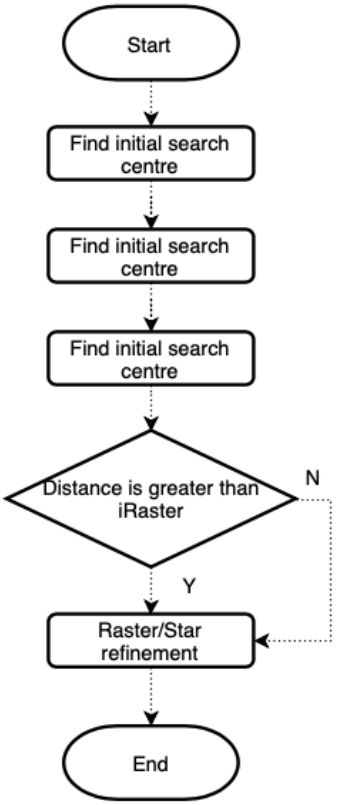

Figure 13. Flowchart of TZ search algorithm

Aiming to further favor compression, since motion data of neighboring blocks are correlated, motion data are predictively coded into the bitstream based on neighboring ones. In order to exploit this correlation, two methods are introduced by HEVC, namely Advanced Motion Vector Prediction (AMVP) and inter-prediction block merging.

In AMVP, a MV is differentially coded based on spatial or temporal neighboring blocks. In contrast to the method used in H.264, where the median of three neighboring blocks within the same frame and with motion vector prediction of temporally collocated blocks supported only in the temporal direct mode, in HEVC a list of spatial and temporal MVPs is constructed. More

specifically, up to two spatial candidates are defined out of a set of 5 neighboring blocks, one temporal from two temporally collocated blocks in the case that temporal candidates are identical or not available and zero candidates when the one or both aforementioned sets are unavailable.

As already mentioned, HEVC utilizes a very efficient, low cost quadtree structure in order to describe the partitioning of a block. While this structure is efficient, it may lead on signaling redundant information to decoder, like the case when neighboring blocks share equal motion parameters. Inter-prediction block merging introduced by a HEVC is a block merging mechanism that performs block merging on those regions. Similar to the AMVP, a list of candidates from spatial or temporal neighboring blocks is constructed. More specifically, up to four spatial candidates are chosen out of an initial list of five, one temporal out of a list of two and additional candidates that include bi-predictive and zero motion vector candidate.

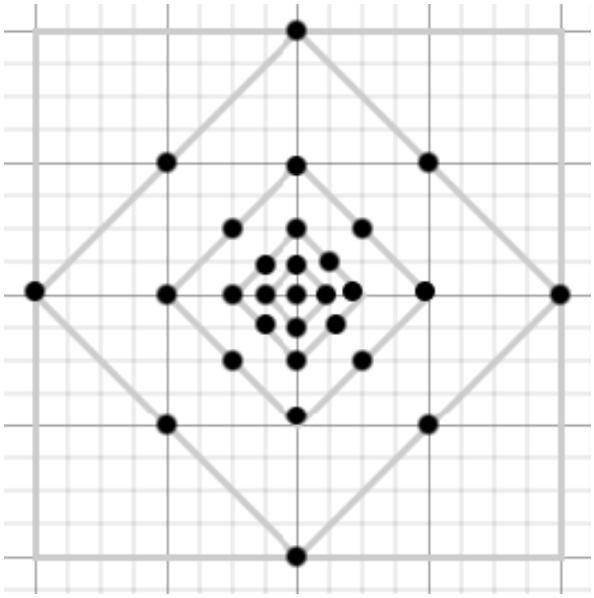

Figure 14. Diamond search

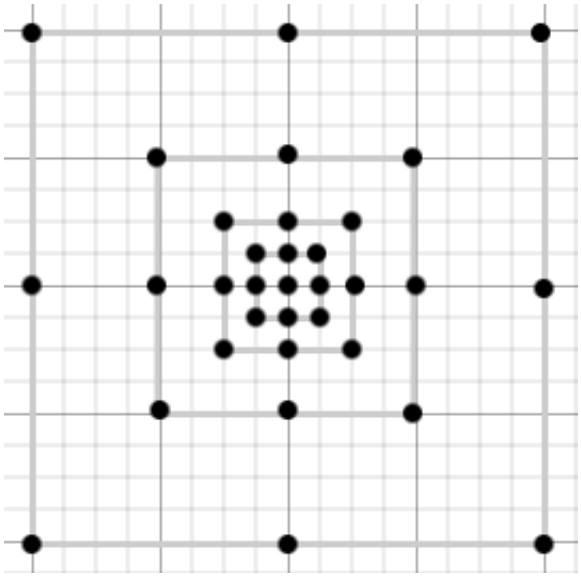

Figure 15. Square search

Transform

In HEVC, transform is applied to the prediction error derived from intra and inter-picture predictions, with the purpose of de-correlating the input residual. As it was mentioned in a previous section, only square-shaped TBs are allowed and HEVC defines transform matrices with their size varying from 4x4 to 32x32. The selection of multiple sizes of transform matrices improves compression efficiency but also increases computational complexity. In that sense, a careful design is implemented in HEVC, 2-D transforms, that are finite precision approximations of the Inverse Discrete Cosine Transform (IDCT), are applied to the defined square blocks by applying 1-D transforms in the horizontal and vertical direction. In HEVC, also an alternate Discrete Sine Transform (DST) is specified, that can be applied only in 4x4 luma blocks, in order to favor coding efficiency. Note, that the standard only specifies the inverse transforms used in the decoding process, so an encoder may benefit of using the actual inverse transform.

Concerning DCT, the properties of the transform can be considered favorable in terms of both compression and implementation efficiency. More specifically, the basis vectors used are orthogonal, favoring compression efficiency by performing uncorrelation of transform coefficients and granting good energy compaction, and also have equal norm, favoring in simplifying the quantization and de-quantization process. Additionally, the elements of a 2^N x 2^N DCT matrix is a subset of a 2^{N+1} x 2^{N+1} DCT matrix, meaning that the basis vectors of the 2^{N+1} x 2^{N+1} matrix is equal to the first half of the even 2^N x 2^N matrix, reducing the implementation cost, since for various transform sizes the same multipliers can be reused. Also, the DCT matrix can be specified using a small number of unique elements favoring hardware implementations. Furthermore, the even basis DCT matrices are symmetric, while the odd are asymmetric, reducing the number of arithmetic operations. Lastly, certain trigonometric relationships characterize the coefficients of a DCT vector, reducing further the arithmetic operations and aiding the implementation of fast algorithms.

However, as already mentioned, transform matrices in HEVC are finite precision approximations of the Discrete Cosine Transform (DCT) matrix. The advantage of using finite precision approximations of the DCT matrix is that this approximation is strictly specified by the standard, avoiding in this way possible mismatches and drift errors caused by different implementations of the IDCT. The most straightforward method of defining integer approximations to the elements of the DCT matrix, is to scale each element with a large number and then round the result to the closest integer. However, this strategy does not always favor compression efficiency, so for each bit-depth, different strategies are defined of approximating the DCT matrix.

In HEVC, transform matrices are scaled by $2^{(6+N/2)}$, compared to the orthonormal DCT transform, and additional scale factors are specified for preserving the norm of the residual block between the forward and the inverse 2-D transforms. Since the scale factors are specified for the inverse transform, also the corresponding scale factors for the forward transform are specified by the standard. The specified scale factors, meet the following constraints:

- The scale factors must be a power of two, in order to allow the implementation of scaling as a right shift.
- In the case of a residual block with all of its samples having a maximum amplitude, after the transformation, the bit depth is set to 16 bits. In this way, a good trade-off between accuracy and implementation cost is achieved.
- The $2^{(6+N/2)}$ scaling applied in the transform matrices, results in an equal scaling of the 1D row, column for both forward and inverse transform. In order to preserve this pattern, through the 2-D forward and inverse transforms, the result of all scale factors must be equal to $(1/2^{(6+N/2)}) = 2^{-24}2^{-2N}$.

Concerning the alternate DST transform for the 4x4 transform blocks, around 1% bit reduction is provided in the case of intra-picture prediction. Since in intra-picture prediction neighboring

samples for the left and/or top are used for predicting a block, an intra-prediction residual has lower amplitude near the edge samples and lower away from them. Due to this certain characteristic, applying DST is proved to be more beneficial than applying the DCT transform.

Quantization

After residual block transformation, quantization is applied. In quantization of transformed blocks, division by a Quantization Parameter (QP) and rounding of the result is performed, while in inverse quantization, the inverse transformed coefficients are multiplied by the quantization step. HEVC utilizes the same URQ scheme used in H.264/AVC, controlled by a Quantization Parameter (QP), with a range defined between 0-51 and an increase of 1 in QP meaning a 12% increment in quantization step, while an increase of 6 means an increment by a factor of 2. This relationship between the QP values and the quantization step is specified by the equation Qstep $(QP) = (2^{1/6})^{QP-4}$.

In HEVC, quantization matrices are specified for all transform block sizes, while the quantization performed is frequency depended. This means that human visual system (HVS) – based quantization is applied, where coefficients with lower frequency are quantized using a finer quantization step compared to higher frequency ones where higher quantization step is used. For some video sequences, utilization of this method, results in a better visual quality compared to a frequency independent implementation.

The quantization matrices used in HEVC and depend on the size and the type of the correspond transform block, are the following:

- Luma: Intra 4 x 4, Inter 4 x 4, Intra 8 x 8, Inter 8 x 8, Intra 16 x 16, Inter 16 x 16, Intra 32 x 32, Inter 32 x 32.
- Cb: Intra 4 x 4, Inter 4 x 4, Intra 8 x 8, Inter 8 x 8, Intra 16 x 16, Inter 16 x 16
- Cr: Intra 4 x 4, Inter 4 x 4, Intra 8 x 8, Inter 8 x 8, Intra 16 x 16, Inter 16 x 16

In the case of frequency dependent quantization, default values are specified for 4 x 4 and 8 x 8 matrices, while the 16 x 16 and 32 x 32 matrices are derived by the 8 x 8 matrices of the same type through upsampling and replication, reducing the memory that is needed to store the quantization matrices. HEVC also allows the usage of non-default quantization matrices. For the cases of 16 x 16 and 32 x 32 matrices, only the 8 x 8 matrices along with the DC value are fed into the bitstream, while predicting a quantization matrix from another is also supported.

Concerning QP derivation, similarly to H.264, where the QP can be modified within a macroblock, in HEVC the QP value can change within a picture. This functionality can serve coding efficiency in terms of rate control or visual quality. In HEVC, the transmission of a delta QP value at Quantization Group (QG) is allowed. The size of QG is a multiple of the CU size and can vary from 8 x 8 to 64 x 64. The value of the delta QP is derived by utilizing the above, left and previous QP values of QG in a decoding order, using a combination of a spatial and a previous QP prediction. Spatial prediction from left and above is used within a CTU, while previous QP prediction is used for the boundary of the CTU.

In HEVC, a set of special coding modes are also specified, namely the I_PCM mode, the lossless mode and the transform skip mode. These specified modes serve to skipping either the transform or both the transform and the quantization steps. More specifically:

- In I_PCM mode, both the transform and the quantization and also the entropy coding and prediction are skipped. This mode is used for avoiding data expansion, e. g. in cases where white noise is present to the video sequence.
- In the lossless mode, transform, quantization and in-loop filtering are skipped. This modes servers in coding mixed video content, e.g. video with text and graphics. In this case, text and graphics are coded in a lossless manner, while the rest of the video content is lossy coded.

- In the case of transform skip mode, only the transform process is skipped. This mode is utilized in screen-content video sequences that contain text and graphics and are generated e.g. in remote desktops or slideshows. This mode is used in order to improve compression efficiency in these cases. Also, the skip of transform is restricted only in transform blocks of 4 x 4 size.

In-Loop Filters

In-loop filtering is applied in the encoding and decoding loops after inverse quantization and before reconstructed picture is saved to the reconstructed picture buffer. In HEVC, two in-loop filters are utilized, a deblocking filter and a Sample Adaptive Offset (SAO) to the output of the deblocking filter. The block-based scheme used by HEVC, introduces discontinuities at the prediction and transform block boundaries, since clocks are coded independently, something that can harm visual quality. Deblocking filter weakens the discontinuities introduced between blocks, while SAO improves even more the quality of reconstructed picture decreasing ringing artifacts and changes in the sample intensity of decoded picture areas. While DBF is only applied on the block boundaries, SAO is applied to all samples of a block.

Concerning DBF, the filtering is applied to the neighboring samples of a PU or a TU boundary, when this boundary is not the picture boundary, or when the DBF is not disabled for tile or slice boundaries. The reason for considering both the PU and TU to apply the filtering process, is because the boundaries of Tus are not always the same as PU boundaries. Additionally, in HEVC, the DBF is applied on an 8 x 8 grid basis, compared to the 4 x 4 grid that was applied in H.264/AVC, reducing the computational complexity, maintaining visual quality and favoring parallel processing. Also, in HEVC implementation, only three strengths are used for DBF, rather than 5 used in H.264/AVC. Considering two adjacent blocks, B1 and B2 that share the 8 x 8 filtering grid, the strength of 2 is applied is applied in the case that one of the blocks is coded using intra-picture prediction. For applying the strength of 1, one or more of the following conditions must be satisfied:

- B1 or B2 has at least on non-zero transform coefficient.
- B1 and B2 do not share the same reference indices.
- Motion vectors of B1 and B2 are not equal.
- The difference of a motion vector between B1 and B2 is greater than or equal to one integer sample.

The filter strength of 0, meaning that no filtering is applied, is selected when none of the above conditions is satisfied. Also, concerning the processing order of DBF, horizontal filtering of vertical edges is first applied, followed by the vertical filtering of horizontal edges. This functionality favors parallel filtering or a CTB-by-CTB filtering implementation.

SAO filtering follows the application of DBF filtering. During SAO filtering, an offset, based on values from a look-up table indicated by the encoder, is added to each decoded sample. Also, this type of filtering is applied on a CTB basis where three types of filtering can be applied. 0 value indicates the no filtering type, 1 indicates the use of the band offset filtering type and 2 indicates the use of edge filtering.

Entropy coding

Entropy coding is performed in the last stage of encoding and the first stage of decoding. Entropy coding is a lossless compression method that exploits statistical properties so that the number of bits used to represent data is logarithmically proportional to their occurrence probability. This means that frequently occurred data are represented with a smaller number of bits, while not so frequently occurred data are represented by a larger number of bits. In HEVC

an improved, in terms of throughput speed and coding efficiency, form of Context-Based Adaptive Binary Arithmetic Coding (CABAC), an entropy coding scheme firstly introduced by H.264/AVC, is used. CABAC involves the steps of binarization, context modeling and binary arithmetic coding and is performed after reducing the video data into a series of syntax elements that describe at the decoder how the video signal will be reconstructed. For example, syntax elements describe the partitioning of a CTU into CUs, whether intra or inter prediction is performed, QPs of this CU and offsets of in-loop filtering of the CTU. Also, in a PU level, syntax elements describe the prediction mode and the associated motion data in the case of inter prediction and in a TU level, the residual signal of the quantized transform coefficients are described in terms of frequency position, magnitude and sign.

Concerning binarization, a set of binary symbol sequences maps in a unique manner each syntax element value. These binary symbols are called bins and can be interpreted in terms of a binary tree structure. In HEVC, a series of binarization processes are utilized, like k-th order truncated Rice (TRk), k-th order Exp-Golomb (EGk) and fixed-length (FL) binarization. Parts of these binarization processes were also incorporated in H.264/AVC, along with the zero-order TRk binarization method. For signaling an unsigned value N, Unary coding, signals a bin of length N+1, where the first N bins are 1 and the last bin is 0. In order to indicate the termination of a syntax element the decoder searches for the 0 value. Truncation in TrU is applied for the largest possible value, namely cMax, of the syntax element. In k-th order truncated Rice (TRk), a prefix and a suffix are indicated for the parameterized code. The prefix is a truncated unary string of N >> k value, where represents the number of the least significant bins, while the suffix is a fixed length binary representation of the least significant bins of N value. Also, in the case where k is equal to 0, TRk is equal to TrU. The k-th order Exp-Golomb binarization consists of a unary prefix and a suffix with a length of l_N+1 and l_N+k respectively, where $l_N = [\log_2((N >> k) + 1]$. Lastly, in fixed-length binarization, a bin string with a fixed length of $[\log_2(cMax + 1)]$ is used, where the least significant bins follow the most significant bins in signaling order. The type of a syntax element defines the selection of the binarization process, or this selection can be based on the selection on previously encoded syntax elements. Also, combinations of the aforementioned binarization processed can be used, or even customized ones.

After decomposing the syntax elements into bins, a further processing is applied, that depends on the coding mode decision that can be a regular or a by-passed mode. In the case of regular arithmetic coding, the regular binary arithmetic coding method is applied, while the by-passed mode is used for bins that are uniformly distributed and the binary arithmetic coding is by-passed. The selection of the coding mode is called context modeling and provides an accurate probability estimation, contributing in an optimized coding efficiency. It is also characterized of high adaptability, utilizing different context models for different bins and updating the corresponding probabilities for each context model. Bins that that share a similar distribution, usually share the same context model. The selection of each context model depends on the type of a syntax element, the bin position within the syntax element or on neighboring information etc. and after each bin a context switch can take place. Additionally, a context memory holds the probability models as 7-b entries and the context index that is computed utilizing a context selection logic, is used to address them.

The last step of CABAC, arithmetic coding, is based on a recursive interval division. According to the probability of each bin, a range that has an initial value of 0 or 1 is divided in two sub-intervals. The value of the decoded bin is determined by an offset that is indicated from one of the two subintervals, and after every bin is decoded, the range is updated according to the selected subinterval and the process of division is repeated. As it was mentioned previously arithmetic coding can be implemented in two modes, regular and bypass.

2.3 AV1 Video Coding Techniques

Picture Partitioning into Coding Blocks

Concerning frame partitioning into blocks, larger coding blocks are supported in AV1 compared to HEVC. As mentioned before, in HEVC a coding block, namely Coding Tree Block (CTB), can be of maximum size 64x64 samples. In AV1, the size of a coding block, referred as Superblock, starts form 128x128. Additionally, while in HEVC block partitions can reach the size of 4x4, in AV1, a block partitioning down to 2x2 samples is supported. Also, a 10-way partition-tree structure is supported including 4:1/1:4 rectangular partitions. An example of all the possible partitions in AV1 is depicted in Figure 16. Another characteristic of block partitioning in AV1 is that rectangular partitions cannot be further subdivided.

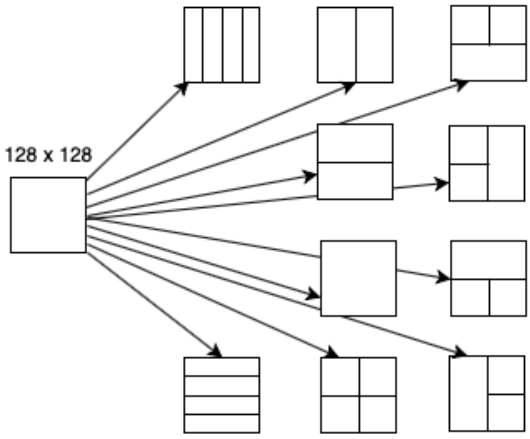

Figure 16. Partition tree in AV1

Intra Prediction

In AV1, 56 directional modes are supported along with non-directional smooth predictors, namely SMOOTH_H, SMOOTH_V and SMOOTH, where the prediction of a block is performed using quadratic interpolation in vertical or horizontal directions or by averaging them. Additionally, a predictor called PAETH, predicts each pixel by copying one reference pixel from the top, left and top-left edge, that has a value that is closest to (top + left -top-left). Also, in order to capture correlation between predicted and reference pixels, five filter intra modes are designed. Each filter consists of eight 7-tap filters, representing the correlation between a 4x2 patch and 7 adjacent neighbors. In that sense, a block predicted in intra mode can select a filter intra mode and be predicted in groups 4x2 patches. Concerning chroma pixels, in order to reduce complexity, in AV1 chroma pixels are predicted by luma pixels via Chroma from Luma (CfL) intra predictor, where chroma pixels are represented as a linear function of reconstructed luma pixels that are subsampled to match chroma pixels resolution. Pallete predictor is also included in AV1, for the cases that a small number of colors can serve satisfactorily block prediction, like screen capture and games. Palette predictor consists of a 2 to 8 colors palette and color indices for each pixel in a block. Lastly, reconstructed blocks within the same frame can be used as predictors, like in inter

prediction blocks from previous frames are used. A new prediction mode, called IntraBC, allows to copy a whole reconstructed block to serve as a prediction.

Inter Prediction

In inter-mode prediction the set of reference frames is extended to the number of 7 in AV1. The group of frames that an input video sequence is partitioned, like the Group of Pictures (GOP) in HEVC, is called Golden-Frame (GF) group, Figure 17.In GF group, a number of frames share the same so called GOLDEN (distant past) and ALTREF (temporal filtered future) frames. A future frame directly coded without temporal filtering, called BWDREF, resides closer and serves as a backward reference. Also, ALTREF2, a future reference frame serves as an intermediate between GOLDEN and ALTREF frames. Also, in addition to the so called LAST (nearest past) frame two near past frames are also supported between GOLDEN and ALTREF, called LAST2

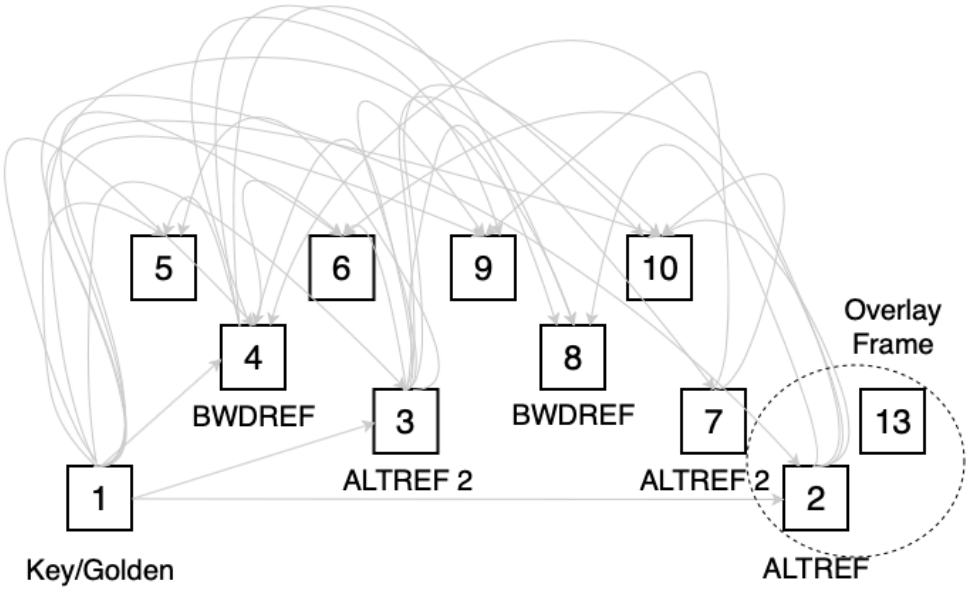

Figure 17. Golden-Frame (GF) Group

and LAST3. A prediction mode can choose all of the aforementioned reference frames, or choose pairs of reference frames, thus providing uni-direction and bi-direction compound prediction.

Since inter frames occupy a large amount of rate cost, efficient scheme for MV reference selection is incorporated in AV1 that works in three stages. In the first stage, reference frame indices and motion vectors for the coded frames are stored. In the decoding phase of a frame, all motion trajectories for any 8x8 block are examined and recorded. After determining the reference frames, MV candidates are derived by the linear projection into the desired reference frames of passing motion trajectories. From a pool of spatial and temporal candidates up to 4 candidates are chosen.

Another tool incorporated in AV1 for inter prediction is the so-called Overlapped Block Motion Vector Compensation (OBMC). OBMC's role is the reduction of prediction errors near block boundaries. This is achieved by forming a combination of neighboring motion vectors in above and right block edges. This functionality is only available for blocks that utilize only one reference frame and works only on the first predictor of a neighbor with two reference frames.

Also, warped motion compensation is included in AV1 standard. In this kind of motion compensation, two affine prediction modes are enabled, a global and a local warped motion compensation. In global motion compensation, motion models are explicitly conveyed in a frame level for the motion between a frame and its references. The local tool, on the other hand,

illustrates the varying motion at a block level. Both tools are utilized only when favoring RD cost compared to translational modes.

Additionally, in order to define a more adaptable inter prediction method, compound prediction is incorporated. In that sense, for a pixel its prediction is generalized as the summation of two weighted predictors. More specifically, compound wedge prediction, difference-modulated masked prediction, frame distance-based compound prediction and compound inter-intra prediction are utilized.

Transform Coding

In AV1, square transform sizes that vary from 4x4 to 64x64 samples are supported. In addition, extended transform kernels for intra and inter blocks are supported. The set of kernels includes 16 horizontal/vertical DCT, ADST, flipADST and IDTX combinations. Due to the fact that as the size of a block increases, some kernels may produce similar results, so transform kernel sets are decreased as transform block size increase. Concerning quantization, both linear and non-linear matrices are supported.

In-Loop Filters

In AV1, several in-loop filters can be applied in a reconstructed frame. Additionally, the flexibility of applying different levels of horizontal and vertical filtering on luma and chroma sample level and also on a superblock level is provided. Also, the first stage is the deblocking filtering and the longest filter is restricted to a 13-tap. Other filtering tools incorporated in AV1 include the so called Constrained Directional Enhancement Filter (CDEF), Loop Restoration Filters, Frame Super-Resolution and Film Grain Synthesis.

CDEF, is applied after the deblocking filter and its purpose is to preserve the details and disburden ringing artifacts in block edges. This is achieved by estimating edge directions and applying a 5x5 non-separable, non-linear, low-pass directional filter with 12 non-zero weights. Right after CDEF filtering, Loop Restoration Filters are applied. Those filters are selected in so called loop-restoration units (LRUs) that can be of size 64x64, 128x128 or 256x256, and two types are offered by the standard, a Separable symmetric normalized Wiener filter and a Dual self-guided filter. In the first case of filtering, a 7x7 Wiener filter is applied, while in the second case, two cheap filters that can be of size 3x3 and 5x5 are applied for each LRU and the outputs from those two filters are then combined with two weights that are signaled into the bitstream and used in order to obtain the restored LRU. Concerning the Frame Super-resolution tool, it gives the potentiality of coding a frame in a lower resolution and then be normatively, in-loop super-resolved to full resolution before reference buffers are updated. This method is supposed to be advantageous in terms of visual quality but is usually very complex. In AV1, a less computationally complex method is incorporated, where linear upscaling is followed by a loop restoration in a higher resolution. Lastly, Film Grain Synthesis is a tool applied outside the encoding and decoding loop. Being a part of artistic intend, film gain needs to be preserved in the final compressed video signal. In order to deal with film grain's randomness in a video sequence, it is removed before encoding and the parameters associated with it are estimated and fed into the bitstream.

Entropy Coding

AV1 utilizes an adaptive multi-symbol arithmetic coder that incorporates a symbol-to-symbol method. In that sense, an alphabet of N elements contains each of the syntax elements in AV1, along with a context consisting of a set of N probabilities, that are stored as 15-bit cumulative distribution functions (CDFs), and a counter in order to aid a fast and efficient adaptation. This method is more accurate compared to a binary arithmetic coder since it can track the probabilities of less common elements in a higher precision. By recursive scaling, probabilities of each syntax element are adapted using an update factor that depends on the size of the alphabet.

For efficient transform coefficient compression and capturing of coefficient distribution, AV1 utilizes a level map coefficient coding, instead of building the probability model upon previously coded coefficient levels, that is based on the observation that the majority of rate cost accounts on coefficients residing on lower levels. That means that for every transform unit, a skip sing is coded, followed by the type of transform kernel and in the case that transform coding is not skipped, the ending position of the non-zero coefficients is recorded. Subsequently, for the coefficient values, levels are broken into different planes. In lower levels (coefficient levels between 0 and 2), higher compression efficiency is ensured by fully accounting the transform dimension, the size of the block and information from neighboring coefficients. For higher levels (between 3 and 15), reduced context model is used, while for levels higher than 15, residuals are directly coded with Exp-Colomb coding.

2.4 Parallelization Techniques in Video Coding Standards

The enhanced coding tools described in the previous subchapter for HEVC and AV1, contribute to an increased computational complexity in both standards. In order to manage this increased computational complexity, parallelization techniques can be utilized in order to accelerate encoding/decoding process and achieve even processing in real-time. Different parallelization techniques, in different levels of coding process, have been introduced by video coding standards, like in Group of Pictures (GOP) level, frame level, frame region level, block level e.g. in motion estimation or filtering and transformations. Concerning frame region level parallelization techniques, slice level parallelization was proposed in previous standards and was also adopted by HEVC. Additionally, in order to overcome some limitations of the parallelization techniques in the aforementioned levels, two new parallelization tools were introduced by HEVC, namely Wavefront Parallel Processing (WPP) and Tiles, to ensure a high-level parallel processing. Tiles, that constitute the main focus of this thesis, are also incorporated in AV1. Slice, WPP and Tile parallelization are more extensively described in this section.

GOP Level Parallelization

In HEVC, a video sequence is divided in independent groups of pictures that allow frame reordering and thus prediction from future frames. Since GOPs within a video sequence are completely independent from each other and no dependencies among them have to be broken, GOPs can be assigned and processed independently in different cores.

Frame Level Parallelization

In frame-level parallelization, multiple frames can be processed in parallel, provided with the constraint that temporal dependencies considering motion-compensated prediction are satisfied. Apart from being a simple technique coding loss free, frame-level parallelization meets a number of limitations. Concerning parallelization scalability, it is regulated by the GOP size and/or the lengths of the MVs. Also, because the encoding times between pictures may significantly vary, this can cause load imbalance between cores, thus harming the overall encoding time. Lastly, parallelization in a frame-level, while increases the frame rate, does not improve latency.

Frame Region Level Parallelization

Slices

Slice partitioning is a concept adopted by both H.264/AVC and HEVC. Under the concept of slices, a picture is partitioned into separate segments that can be encoded and decoded independently. In HEVC, a slice can have a maximum size of a whole frame, or a minimum size that of a CTU. Partitioning a frame into a number of slices, serves three distinct purposes:

- **Error robustness:** Slice partitioning serves error robustness since re-synchronization of both decoding and parsing is possible in the case of data losses in a packet-wise transportation, meaning that a packet loss implies a loss of a slice.
- **Maximum Transmission Unit (MTU) Size Matching:** Under MTU size constraint, a packetization scheme that restricts the size of payload data in IP networks, a slice can contain a varying number of CTUs so that this constraint can be satisfied.
- **Parallel Processing:** Since a slice constitutes an independent encodable/decodable entity, parallel encoding/decoding of the slices within a frame is achievable.

Similarly, to what is specified in H.264 where a slice contains a number of consecutive macroblocks, in HEVC a slice consists of a set of consecutive CTUs that are processed in a raster

scan order. Dependencies between CTUs are broken across tile boundaries, e.g. dependencies concerning intra prediction or entropy coding, and therefore each slice carries its own CABAC bitstream and so can be parsed and decoded independently. Due to the fact that spatial redundancy is not fully exploited, coding efficiency is usually affected, especially when an increased number of slices is used for a picture.

Slices, as where defined in previous standards, including H.264, consist of a header and the corresponding data within each slice. During the development of HEVC, it was realized that the aforementioned concept was introducing a significant bit-rate overhead, caused by multiple slice headers, and was also too inflexible to satisfy all foreseen scenarios, due to the breaking dependencies across slice boundaries.

Consequently, a more flexible concept was introduced by HEVC, considering slice fragmentation at two distinct levels. In the first slice fragmentation level, a slice can be divided into a number of slice segments, meaning that each segment contains a subset of the CTUs residing within the slice. The first slice segment is the independent slice segment, or regular slice, and contains a full header for that segment. The rest of the slice segments are called dependent, they carry sorter slice segment headers and all CTU dependencies are allowed across their boundaries, thus no coding efficiency penalty is introduced. An example of this level of slice fragmentation is provided in Figure 18.

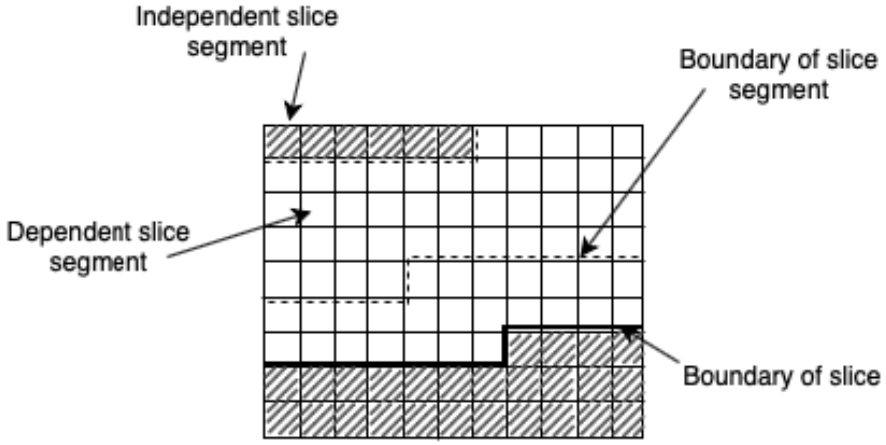

Figure 18. Picture partitioning using slices in HEVC

In the second level of slice fragmentation, a subset of a slice segment, namely a substream, contains the coded data of a subset of CTUs within the slice segment. Subsets or substreams can be efficiently used along with high-level parallelization tools.

Wavefront Parallel Processing (WPP)

In WPP, a frame is divided in CTU rows that can be processed in parallel, with the limitation that two consecutive CTUs in the preceding row have already been processed. An example of Wavefront parallel processing in HEVC is depicted in Figure 19**Error! Reference source not found.**. In this way, the dependencies between consecutive CTUs are not broken and no CABAC initialization is applied in the start of each CTU row. On the contrary, the CABAC context variables are disseminated from the second CTU in the preceding row, to the first CTU of the current row. In this way, losses in coding efficiency are kept small, compared to case that no WPP is used for encoding/decoding. Additionally, in WPP, the CTUs in a row are processed in a raster scan order, and a number of threads that can be up to the number of CTU rows can be utilized to process each CTU row in parallel.

The aforementioned dependencies between neighboring CTUs do not allow all of the threads to start processing simultaneously each CTU row and so finish the processing at the same time. This restriction, along with the case that a "heavy" CTU, thus slower to encode, may delay the whole process, introduces parallelization inefficiencies, leading to a limited scalability of parallel processing.

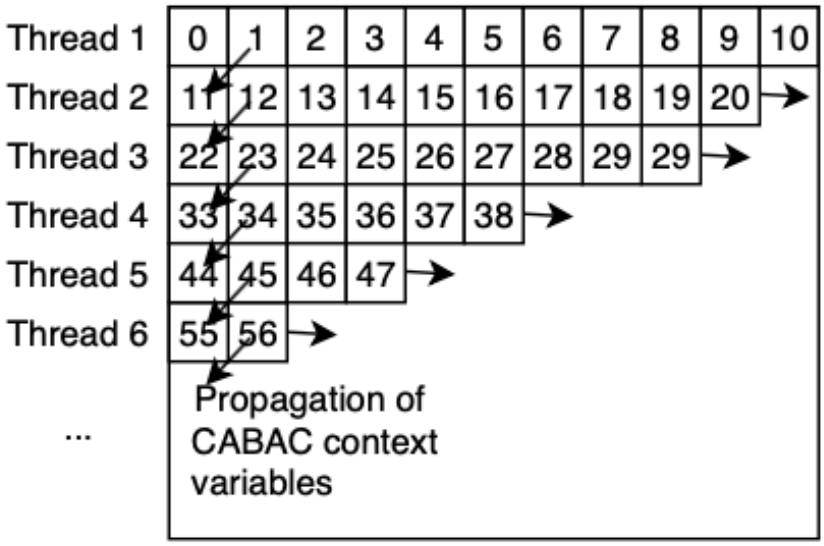

Figure 19. Wavefront Parallel Processing (WPP) in HEVC

Tiles

Tile partitioning is a parallelization technique firstly introduced in HEVC and also adopted by the newly introduced AV1. Under tile partitioning, a frame is divided into rectangular-shaped blocks. As already mentioned before, in HEVC those blocks are called CTUs and are of maximum size 64x64 pixels, while in AV1they are called superblocks and are of maximum size 128x128 pixels.

Similar to the case of slices, dependencies concerning both intra picture and entropy coding are broken across tile boundaries and CABAC context variables are re-initialized for each tile, except for the case of in-loop filters that can still operate across tile boundaries in order to diminish artifacts is tile borders. This dependency breaking may affect in some extend the coding

efficiency but gives the potentiality of parallelization, since every tile becomes independently encodable/decodable, where CTUs are processed in a raster scan order and each tile can be assigned for processing to a different core in a multi-core system. An example on a 4x3 tile partitioning in HEVC and AV1 is shown in Figure 20 and Figure 21 respectively.

Additionally, each frame in a video sequence can be divided into different number of tiles and also the boundaries of tiles can be defined differently from frame to frame in order to achieve a more load balanced tile partitioning.

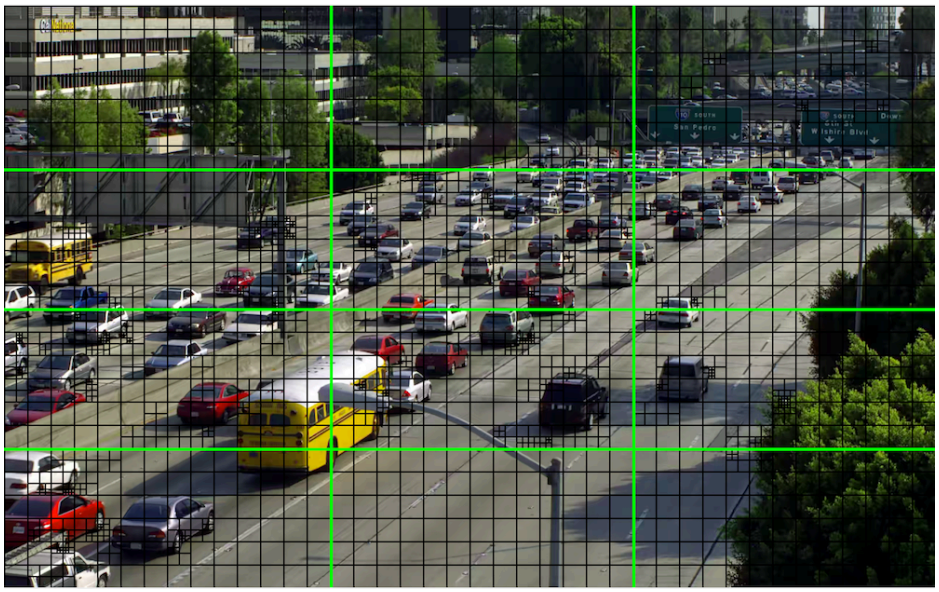

Figure 20. Example of 4 × 3 tile partitioning in HEVC (Traffic sequence)

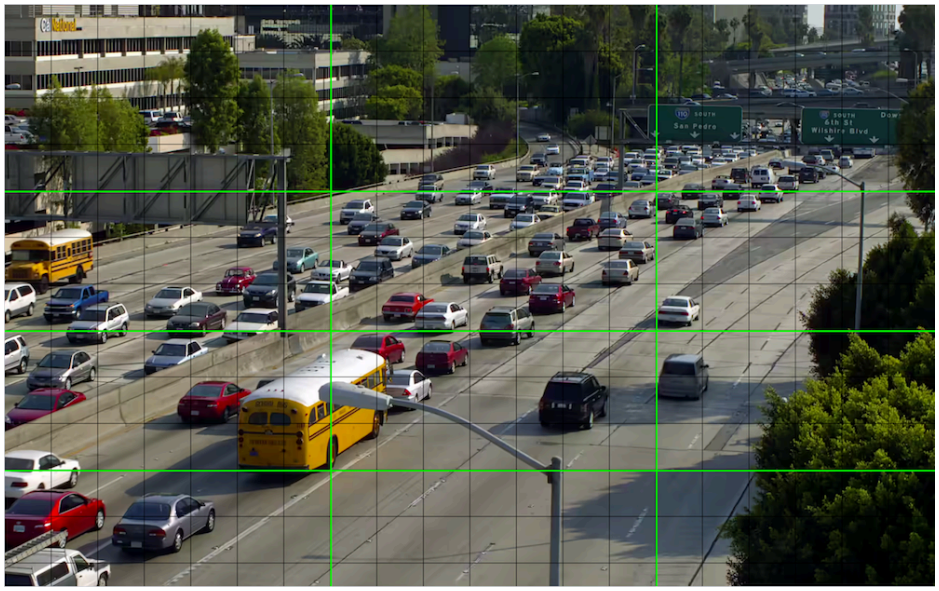

Figure 21. Example of 4 × 3 tile partitioning in AV1 (Traffic sequence)

Block Level Parallelization

In this level of parallelization, several block-level encoding functions (intra prediction, inter prediction, quantization, transform and filtering), can be carried out in a parallel fashion on a block level, when the dependencies among sub-blocks are not violated. For clarity's sake, some examples of parallel schemes implemented for HEVC are provided. For example, in [15] a parallel implementation of HEVC decoder, combining CPU with GPU, is proposed. Parallelization is implemented upon GPU, using CUDA where the parallelized modules are IQ (Inverse Quantization), ICDT (Inverse Discrete Cosine Transformation), Intra/ Inter Decoder, DF (Deblocking Filter) and SAO (Sample Adaptive Offset), being able to achieve parallelization up to pixel or even fractional-pixel level degree. CPU undertakes NAL (Network Abstraction Layer) and CABAC (Context-based Adaptive Arithmetic Coding), the modules that are forced to be processed sequentially, due to their inherent context dependencies. In [16] ,optimization of the most time consuming modules in HEVC is proposed. Those most time consuming parts of the HEVC encoder were found, after analysis, to be MC (Motion Compensation), Hadamard Transform, SAD/SSD (Sum of Absolute Difference/sum of Squared Distance) Calculation and Integer Transform. Those modules were accelerated using Streaming SIMD Extension (SSE), the instruction set designed for Intel x86 processors, so multiple pixels were processed simultaneously. Concerning intra prediction, in [17] a parallel intra prediction algorithm for HEVC encoder is proposed. The proposed scheme efficiently utilizes both CPU and GPU with no additional overhead between them, targets HD and UHD video sequences and is compatible with both single- and multi-threaded encoders. Intra prediction is performed on one, in the case of single-threaded, or more GPUs, in the case of multi-threaded implementation and an important feature that makes parallelization realizable on GPU is that POS (Prediction based on Original Samples) is utilized, so dependences among PBs (Prediction Blocks) are broken. Additionally, a parallel-friendly RDO (Rate Distortion Optimization) scheme, accounts for the absence of synchronization between CPU and GPU, minimizing even more the encoding time. The corresponding calculated costs for each PB are transferred to CPU, where they are used in intra mode decision, accelerating significantly the intra module. Also, in [18] a parallelization scheme for HEVC intra prediction is proposed aiming to accelerate encoding process. In the proposed method, intra prediction of 4x4 sub-blocks is considered, where the corresponding 8x8 block is divided in two 4x8 sets of blocks and parallelization is then applied in each of those sets. The first set is predicted first, where reference pixels are the above and left neighboring pixels of the corresponding set, and when the first set is predicted the second set follows. In that way, intra prediction of an 8x8 block can be achieved in two sequential steps in addition to the conventional method where four sequential steps are required. Regarding inter prediction, in [19] a hybrid encoding scheme is proposed, where on GPU, in order to accelerate ME (Motion Estimation) module, is proposed. Motion estimation is performed on GPU, where 8 consecutive CTUs in a row are processed while CPU waits and then uses the already calculated MVs and corresponding costs in order compress a line of CTUs. For this task, two buffers are used so when GPU writes to one buffer CPU reads from the other one and in each step those buffers are swapped. The motion estimation algorithm that calculates integer MVs (Motion Vectors), executes a RSAD (Recursive Sad of Absolute Difference) on GPU processor, where instead of performing ME (Motion Estimation) in every single PU (Prediction Unit), an entire CTU is processed. Also, for searching procedure a Frayed Diamond Search Pattern (FDSP) is adopted in order to reduce corresponding complexity. The MVs for the case of half- and quarter-pixel accuracy are also pre-calculated on GPU, using Hadamard Transformed SAD (SADT) in order to define the best MVs. In [20], also a hybrid technique is utilized, where, ME (Motion Estimation) is performed on GPU, and resulting costs for each PB (Prediction Block), along with its corresponding MV (Motion Vector), are directly available to CPU when MD (Mode Decision) is performed. In GPU, ME is performed in parallel for all Pus of a whole CTU, using the RSAD (Recursive-Sum of Absolute Difference) algorithm and procedure is further accelerated, utilizing SSAD (Sub-Sampled SAD)

and Diamond-shaped pattern, in order to reduce the number of calculations and search positions. Concerning filtering parallelization, in [21], an optimized parallel scheme of the deblocking filtering (DF) process is proposed, targeting to accelerate processing time of DF which accounts for about 20% of the total encoding time. The proposed scheme constitutes of three steps. In the first step, the filter mode of each edge of the 8x8 block in a 32x32 CTU is decided (strong, wick or no filtering) and only edges that need to be filtered are assigned to cores. Also, in that step, as opposed to conventional HEVC deblocking filter, vertical boundaries are processed before horizontal boundaries, in terms of reducing the total processing time. In the next step any vertical filtering needs to be performed before any horizontal and redundant calculations need to be avoided. To meet that need, the proposed scheme splits each horizontal and vertical edge into an upper and a lower section. For each previously decided filtering, a set of 26 pairs of candidate offsets is produced and reused in the next steps of DF process, thus minimizing encoding time. Additionally, in [22] a parallel GPU implementation of In-Loop filters is implemented. In the case of DF (Deblocking Filter) a 64x8 block of samples was assigned to be processed from each warp, hence 8x8 blocks were assigned to each of the 4 threads, comprising a ThB (Thread Block), and processed in parallel. Also, each warp was set to occupy its own space in shared GPU memory until the filtering was completed and the results were stored back in global memory. Also, for both Horizontal and Vertical Filtering, information needed to indicate BS (Boundary Strength) was stored in an 8-bit code word. In the case of SAO filtering, 4 CTUs were set to be processed in parallel, hence a set of 32 pixels was processed from each thread in a warp and required information for SAO filtering of each CTU was stored in 12-byte code-word.

Chapter 3: Tile Partitioning in HEVC and AV1

As it was mentioned in the previous chapter, tile partitioning is a tool firstly introduced in HEVC as a high-level parallelization technique and also adopted by the recently launched AV1. Although tiles offer the potential of trivially parallelizing the encoding/decoding process, they may lower coding performance. While a variety of works exist that characterize the effects of tile partitioning in HEVC, or compare the performance of the two coding standards, the same does not hold true for the case of tile partitioning in AV1 against HEVC and so constitutes the main focus of this thesis.

3.1 Tile partitioning overview

Concerning tile partitioning in HEVC, an overview of tiles is conducted in [14],where the authors explore the potentiality of the new coding tool in achieving high levels of parallelization while maintaining coding efficiency. In that sense, a variety of research exists on how this potentiality can be exploited efficiently. In [13], an algorithm that adaptively determines tile boundaries in order to balance the load between CPU cores is proposed. In order to achieve that, a weighting matrix holding the depth, mode and size information was used in order to estimate the compression complexity of each CTU. Based on the weight estimated for each tile, a master tile with a weight close to the ideal average was calculated and by extending the boundaries of this master tile, the remaining tiles were defined. In [23],the first the vertical and then the horizontal cuts of tile partitioning were defined so as to balance the estimated, from previous CTUs, compression times. Also, in [24], a tile partitioning algorithm based on theoretical results from the array partitioning problem was proposed.

The negative impact that tile partitioning can have on coding efficiency, was what motivated the works of [25] and [26]. In [25], highly correlated image regions were spotted using a variance map in order to define vertical and horizontal tile boundaries, while in [26] a tradeoff between coding efficiency and load balancing was proposed by introducing two different metrics for CTU cost (for efficiency and speedup).

All of the aforementioned works considered the case where the number of tiles is equal to the number of the available cores. On the contrary, in [27] a load balancing scheme for the case where the number of tiles is greater than the number of cores is proposed and similarly in [28] an adaptive tile resizing scheme is proposed for load balancing for the same scenario. Lastly, in [29] adjusting tile dimensions according to the number of the available CPU cores is proposed.

3.2 Comparative AV1 Performance

Even in the earliest stages of AV1 development, before the current version of the codec was released, a series of studies were conducted in order to evaluate the performance of the new video coding standard and compare it against others. In [30] the early version of AV1, v0.1, is evaluated and compared against the reference and the commercial software of HEVC, and also against H.264. The results demonstrated an inferior performance of AV1, with a bitrate overhead of about 65.7% when compared to HEVC and 10.5% against H.264 for the same visual quality. Also, in [31], two coding scenarios, a constant quality and a targeted bitrate scenario, were considered in the comparative assessment. The results indicated a better coding efficiency performance of AV1 by an average 7% compared to HEVC, in the targeted bitrate scenario, while HEVC outperformed AV1 in the case of constant quality scenario. Similarly, in [8] under a fixed Quantization Parameter (QP) AV1 produced a 47% bitrate overhead against HEVC. Additionally, in [31] AV1 was proved to be significantly slower, more than 20x, than HEVC regardless of the coding scenario. Lastly, in [32] HEVC was found to achieve a superior performance in lower resolution video sequences, while for higher resolutions, AV1 showed to outperform x265 video codec in [7].

However, in all of the aforementioned works, AV1 was tested and compared considering no parallelization, since the goal was to evaluate the standard in terms of its coding efficiency. Apart from coding efficiency the time of execution is also an important aspect in video compression and so the effects of applying parallelization techniques in AV1 need to be studied. In , the effects of built-in vertical implementation of tiles in version 0.7 of AV1 standard were investigated, in terms of speed-up, in both the encoding and decoding. Since a great variety of methods for tile parallelization exist for HEVC and have the potential of adapting in AV1, in this thesis, characterizing the effects of grid-fashion tile cut is considered, without delving on parallel execution.

Chapter 4: Experimental Evaluation

4.1 Experimental Setup

In order to evaluate and compare the performance of AV1 and HEVC in terms of coding efficiency when tile partitioning is applied, version 1.0 of AV1 reference software [5] and version 16.20 of HM reference software [33] were used. Experiments were conducted on a Linux server with two 12-core Intel Xeon E5-2650 CPUs running at 2.2 GHz using Class A test sequences, as defined by common test conditions described in [34],that are shown in Table 1. As it can be seen from the table, the first 100 frames of each test sequence were used in order to experimentation time to a manageable extend.

Table 1: Video Sequences

Name	Resolution	Frames per second (fps)	Total frames	CTUs per frame	Superblocks per frame
PeopleOnStreet	2560 x 1600	30	100/150	1000	260
Traffic	2560 x 1600	30	100/150	1000	260

Also, two coding scenarios were considered for HEVC, referred as Low Delay (LD) and Random Access (RA), also defined by the common test conditions and described in [34].Additionally, the default settings were used for each coding scenario, involving a GOP size of 4 and a sequence structure where an I frame is followed by P frames for the LD scenario, and a GOP size of 16 with B frames and I frame insertion every 2 GOPs for the RA scenario. In both cases, bit depth was set to 8, CTU size to 64 x 64, max depth for partitioning to 4 and TZ search was defined as a search mode.

Concerning AV1 reference software, a similar approach to the fixed quality scenario, as described in [30] and [31], was used. However, in AV1's recommendation, defining a minQP and a maxQP, that differ by at least 8, is suggested. In that sense, a targeted QP value was defined and the range was set to vary ±4 from this value. The QP values used for HEVC were 22, 27, 32, 37, which translate to 27, 33, 39 and 46 for AV1, as described in [32]. In Table 2, the encoding parameters used in AV1, for the two cases of using and not using tiles, are summarized. The remaining parameters were kept to their default values.

Table 2: AV1 Encoder Settings

Codec version	AOMedia Project AV1 Encoder 1.0.0-685-g38acd6c
Selected settings (no tiles)	--psnr --tune=psnr --end-usage=q --passes=2 --arnr-strength=5 --limit=100 --min-q=$min_qp --max-q=$max_qp --cq-level=$qp --threads=0 --tile-columns=0
Selected settings (tiles)	--psnr --tune=psnr --end-usage=q --passes=2 --arnr-strength=5 --limit=100 --min-q=$min_qp --max-q=$max_qp --cq-level=$qp --threads=0 --tile-width=$tile_width --tile-height=$tile_height

4.2 Results and Discussion

In the first experiment, the case where no tiles are used in the encoding process was considered for both AV1 and HEVC. In Figure 22 and Figure 23, the RD-curves for the two test sequences, PeopleOnStreet and Traffic, are shown respectively. As it can be seen, for both sequences, HEVC gives better results in RA scenario compared to the LD configuration. Also, AV1 is shown to achieve different tradeoffs in terms of bitrate-PSNR for medium and large QP values compared to HEVC. Additionally, in the Traffic sequence, it is clear that AV1 outperforms HEVC in terms of bitrate savings, reducing bitrate by 30.57% compared to RA configuration, but performs inferiorly in terms of PSNR having a difference of 0.72 dB.

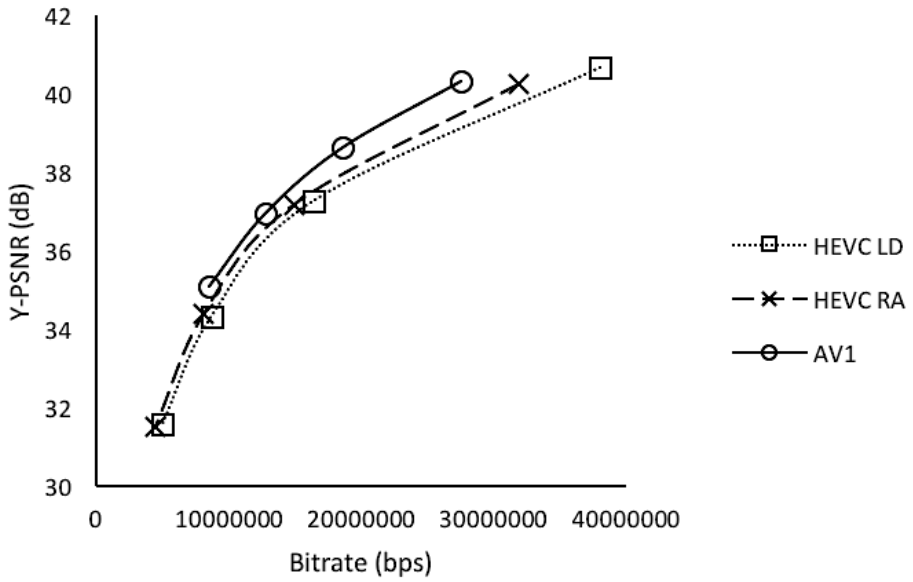

Figure 22. Coding efficiency comparison between AV1 and HEVC (QP={27/22, 33/27, 39/32, 46/37}, PeopleOnStreet sequence)

In Figure 24 and Figure 25, the running times of HEVC and AV1 are plotted. As shown in both figures, for both test sequences, AV1 seems to be significantly slower than HM, between 1 and 2 orders of magnitude. Also, it can be observed that the execution time of HEVC drops sharper than the execution time of AV1, when increasing the value of QP. According to the aforementioned observations, the need for parallelization techniques in AV1 can be concluded, and so the need for characterizing the impact of using tiles in the coding efficiency of AV1. In order to characterize the effects of using tiles, the Bjøntegaard metric was used for depicting the

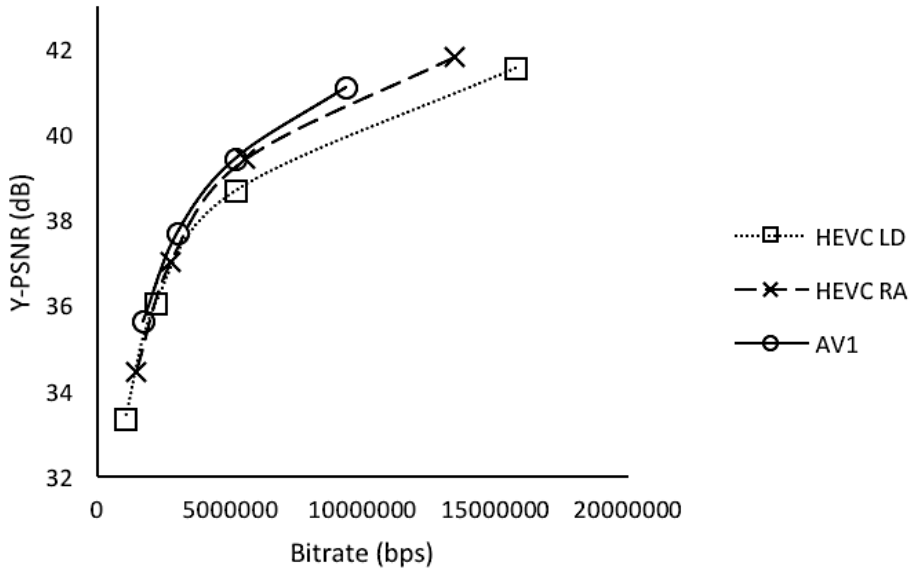

Figure 23. Coding efficiency comparison between AV1 and HEVC (QP={27/22, 33/27, 39/32, 46/37}, Traffic sequence)

PSNR (or BD-PSNR) loss [35] and the rate (or BD-rate) percentage increase between the cases of not using and using tiles. In that sense, Figure 26 and Figure 27 show the BD-PSNR loss and BD-rate increase respectively, for the case of AV1 video codec, while Figure 28, Figure 29, Figure 30 and Figure 31 show the equivalent for HEVC and for both coding scenarios (Low Delay (LD) and Random Access (RA)).

As can be seen, in both standards, losses in PSNR increase as the number of tiles increases and that is also the case for BD-rate, that increases along with the number of tiles. Comparing AV1 figures to the HEVC ones it can be observed that tile partitioning has a larger impact on AV1's coding efficiency, both in terms of BD-PSNR and BD-rate, something that indicates that maybe the implementation of tile partitioning can be further improved in AV1. However, the values shown in both Figure 26 and Figure 27, are rather in the small side even when 12 tiles are used. Additionally, considering the greater computational overhead of AV1 and the fact that parallelization using 12 tiles can reduce the running time even by an order of magnitude, a 2% or 3% increase in BD-rate can be considered as a valid trade-off.

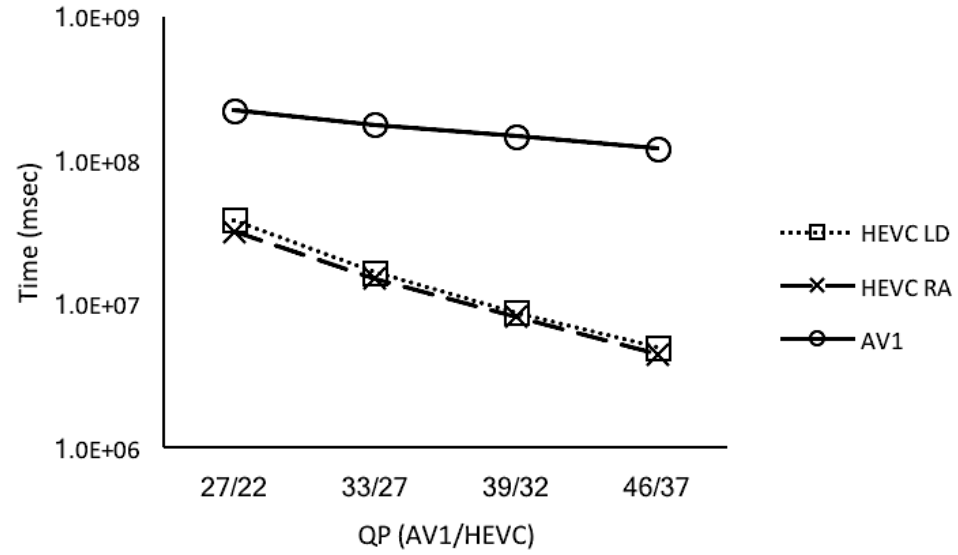

Figure 24. Time comparison between AV1 and HEVC (PeopleOnStreet sequence)

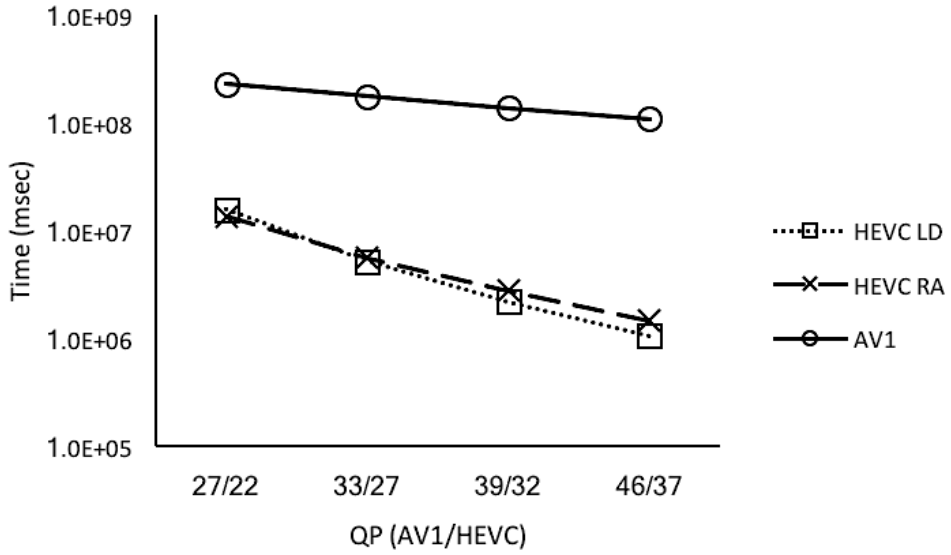

Figure 25. Time comparison between AV1 and HEVC (Traffic sequence)

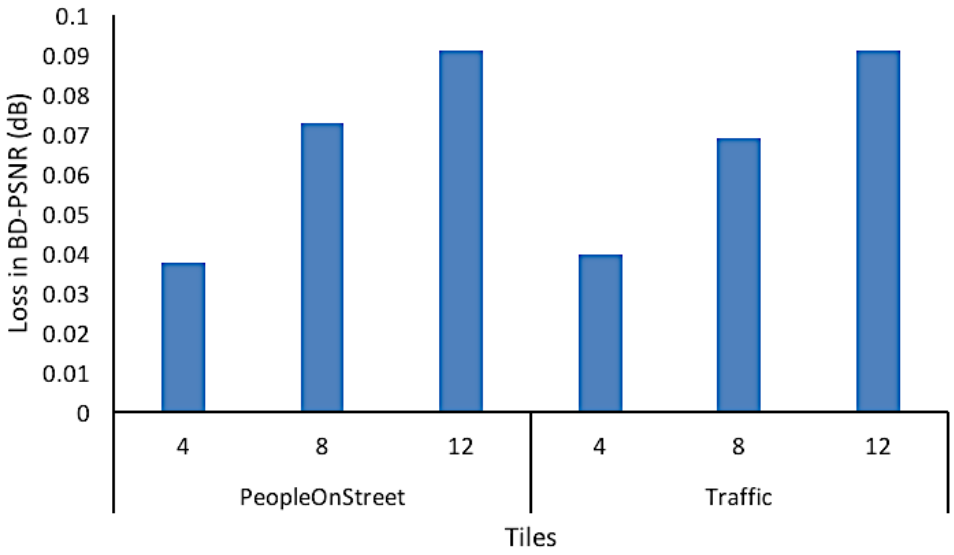

Figure 26. Tile partitioning BD-PSNR loss in AV1 (QP={27, 33, 39, 46})

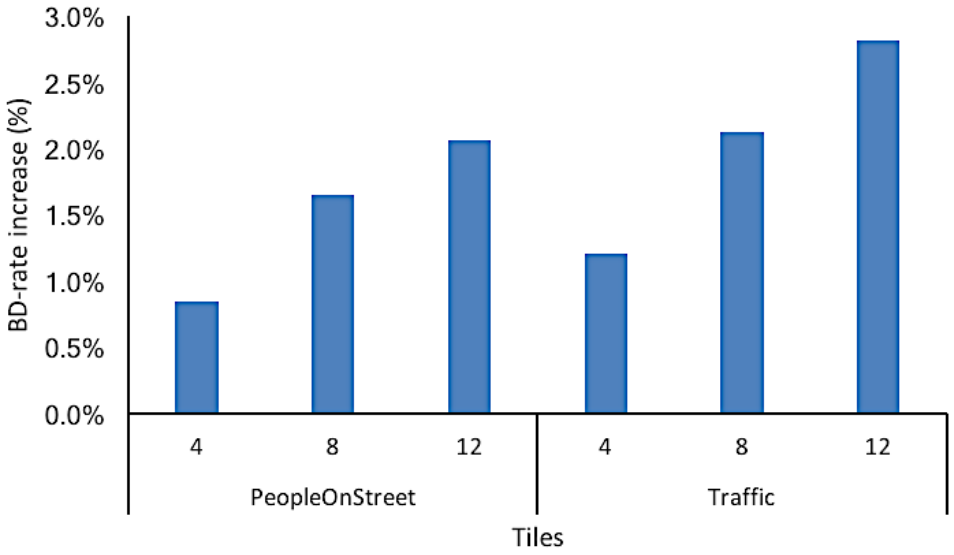

Figure 27. Tile partitioning BD-rate increase percentage in AV1 (QP={27, 33, 39, 46})

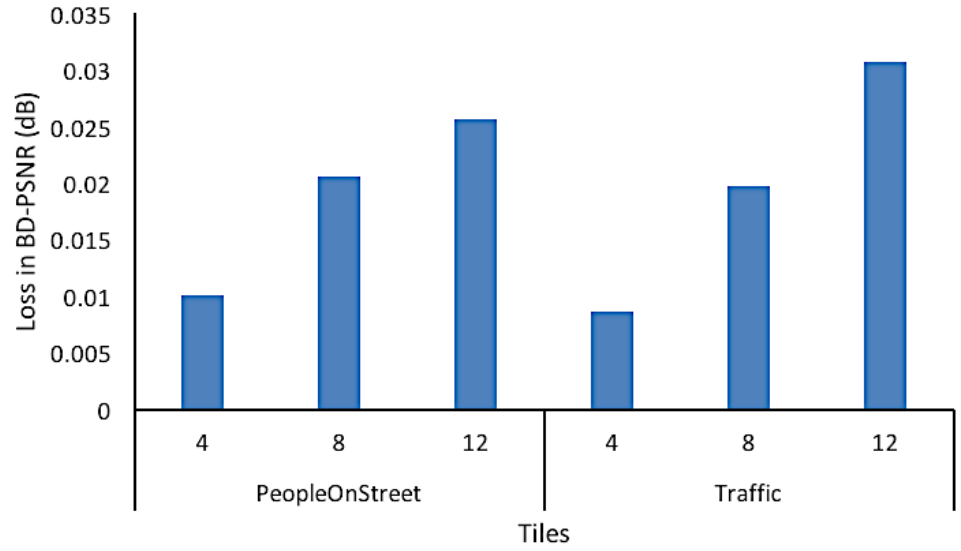

Figure 28. Tile partitioning BD-PSNR loss in HEVC (QP={22, 27, 32, 37}, low-delay)

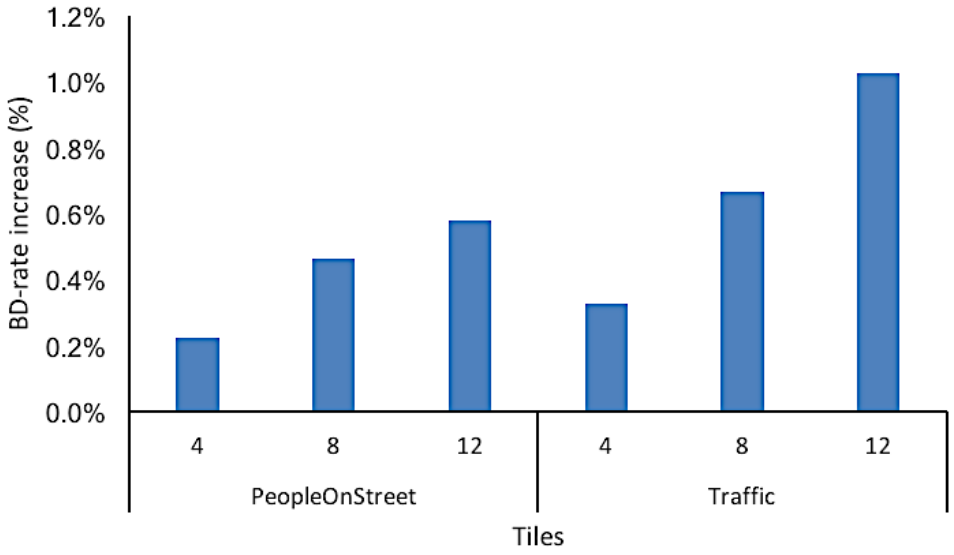

Figure 29. Tile partitioning BD-rate increase percentage in HEVC (QP={22, 27, 32, 37}, low-delay)

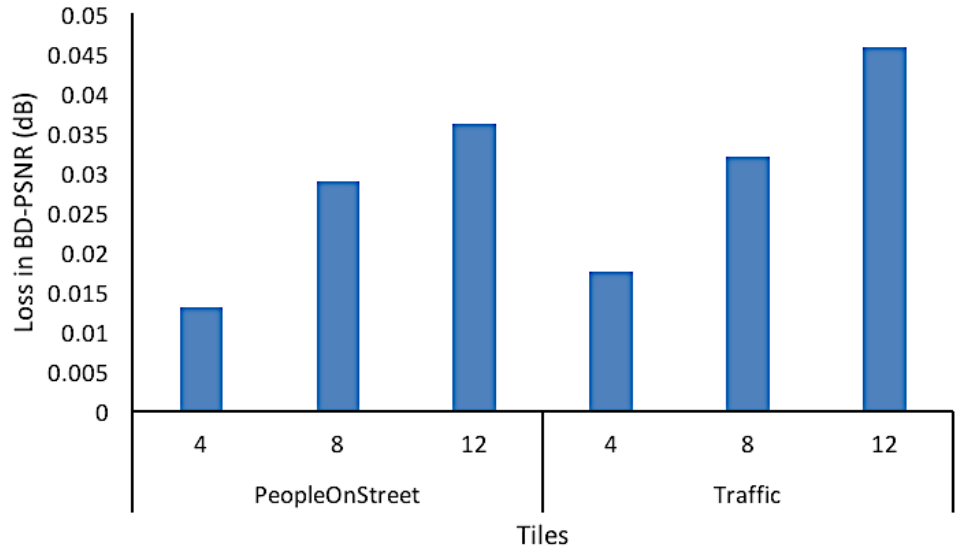

Figure 31. Tile partitioning BD-PSNR loss in HEVC (QP={22, 27, 32, 37}, random-access)

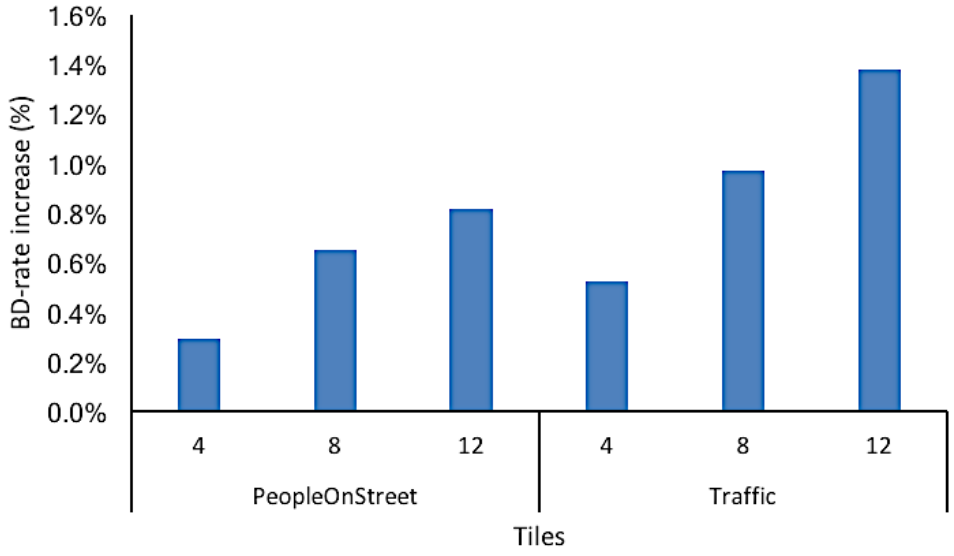

Figure 30. Tile partitioning BD-rate increase percentage in HEVC (QP={22, 27, 32, 37}, random-access)

Chapter 5: Conclusions

In this thesis, the impact of tile partitioning on coding performance of AV1 was evaluated and compared against HEVC. The motivation behind this work was the lack of tile partitioning evaluation in the relevant literature, for the case of AV1, and the importance of applying parallelism especially in new, feature rich codecs. In that sense, experimental results demonstrated that AV1 is, for the time being, significantly slower than HEVC, something that enhances the need for applying parallel techniques and also characterizing the impact of those techniques in coding efficiency. The findings indicated that although tile parallelism accounts for a higher negative impact in AV1 than HEVC, still a valid trade-off between coding efficiency and computational overhead is provided.

References

[1] N. Panagou, P. K. Papadopoulos, M. Koziri, and T. Loukopoulos, "On evaluating the impact of tile partitioning in AV1," in *Proceedings of the 22nd Pan-Hellenic Conference on Informatics*, 2018, pp. 121–126.

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

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Trans. circuits Syst. video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

[4] C. C. Chi *et al.*, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. circuits Syst. video Technol.*, vol. 22, no. 12, pp. 1827–1838, 2012.

[5] "AV1 Bitstream & Decoding Process Specification," 2018. [Online]. Available: https: //aomedia.org/av1- bitstream- and- decoding- process- specification/.

[6] "Versatile Video Coding (VVC)," 2018. [Online]. Available: https://jvet.hhi.fraunhofer.de/.

[7] "MSU Codec Comparison 2017 Part V. 2017." [Online]. Available: http: //www.compression.ru/video/codec_comparison/hevc_2017/MSU_HEVC_ comparison_2017_P5_HQ_encoders.pdf.

[8] D. Grois, T. Nguyen, and D. Marpe, "Performance comparison of AV1, JEM, VP9, and HEVC encoders," in *Applications of Digital Image Processing XL*, 2018, vol. 10396, p. 103960L.

[9] S. Fouladi *et al.*, "Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads.," in *NSDI*, 2017, pp. 363–376.

[10] J.-F. Franche and S. Coulombe, "A multi-frame and multi-slice H. 264 parallel video encoding approach with simultaneous encoding of prediction frames," in *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, 2012, pp. 3034–3038.

[11] M. Koziri, P. K. Papadopoulos, and T. Loukopoulos, "Combining tile parallelism with slice partitioning in video coding," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2018, vol. 10752.

[12] E. Hojati, J.-F. Franche, S. Coulombe, and C. Vázquez, "Highly parallel HEVC motion estimation based on multiple temporal predictors and nested diamond search," in *Image Processing (ICIP), 2017 IEEE International Conference on*, 2017, pp. 2746–2750.

[13] Y.-J. Ahn, T.-J. Hwang, D.-G. Sim, and W.-J. Han, "Implementation of fast HEVC encoder based on SIMD and data-level parallelism," *EURASIP J. Image Video Process.*, vol. 2014, no. 1, p. 16, 2014.

[14] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," *IEEE J. Sel. Top. Signal Process.*, vol. 7, no. 6, pp. 969–977, 2013.

[15] A. Ma and C. Guo, "Parallel acceleration of HEVC decoder based on CPU+ GPU heterogeneous platform," in *Information Science and Technology (ICIST), 2017 Seventh International Conference on*, 2017, pp. 323–330.

[16] K. Chen, Y. Duan, L. Yan, J. Sun, and Z. Guo, "Efficient SIMD optimization of HEVC encoder over X86 processors," in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, 2012, pp. 1–4.

[17] S. Radicke, J.-U. Hahn, Q. Wang, and C. Grecos, "A parallel HEVC intra prediction algorithm for heterogeneous CPU+ GPU platforms," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 103–119, 2016.

[18] J. Jiang, B. Guo, W. Mo, and K. Fan, "Block-Based Parallel Intra Prediction Scheme for HEVC.," *J. Multimed.*, vol. 7, no. 4, 2012.

[19] S. Radicke, J. Hahn, C. Grecos, and Q. Wang, “A highly-parallel approach on motion estimation for high efficiency video coding (HEVC),” in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, 2014, pp. 187–188.
[20] S. Radicke, J. Hahn, C. Grecos, and Q. Wang, “Highly-parallel HVEC motion estimation with CUDA [title missing from article PDF],” in *Visual Information Processing (EUVIP), 2013 4th European Workshop on*, 2013, pp. 148–153.
[21] M. M. Fouad and R. M. Dansereau, “An optimized parallel order scheme of the deblocking filtering process for enhancing the performance of the HEVC standard using GPUs,” *Multimed. Tools Appl.*, vol. 76, no. 23, pp. 24609–24634, 2017.
[22] D. F. de Souza, A. Ilic, N. Roma, and L. Sousa, “HEVC in-loop filters GPU parallelization in embedded systems,” in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on*, 2015, pp. 123–130.
[23] I. Storch, D. Palomino, B. Zatt, and L. Agostini, “Speedup-aware history-based tiling algorithm for the HEVC standard,” in *Image Processing (ICIP), 2016 IEEE International Conference on*, 2016, pp. 824–828.
[24] M. Koziri *et al.*, “Heuristics for tile parallelism in HEVC,” in *Signal Processing Conference (EUSIPCO), 2017 25th European*, 2017, pp. 1514–1518.
[25] C. Blumenberg, D. Palomino, S. Bampi, and B. Zatt, “Adaptive content-based Tile partitioning algorithm for the HEVC standard,” in *Picture Coding Symposium (PCS), 2013*, 2013, pp. 185–188.
[26] C.-H. Chan, C.-C. Tu, and W.-J. Tsai, “Improve load balancing and coding efficiency of tiles in high efficiency video coding by adaptive tile boundary,” *J. Electron. Imaging*, vol. 26, no. 1, p. 13006, 2017.
[27] M. Shafique, M. U. K. Khan, and J. Henkel, “Power efficient and workload balanced tiling for parallelized high efficiency video coding,” in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014, pp. 1253–1257.
[28] P. K. Papadopoulos, M. Koziri, and T. Loukopoulos, “A Fast Heuristic for Tile Partitioning and Processor Assignment in Hevc,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 4143–4147.
[29] G. Malossi, D. Palomino, C. Diniz, A. Susin, and S. Bampi, “Adjusting video tiling to available resources in a per-frame basis in High Efficiency Video Coding,” in *New Circuits and Systems Conference (NEWCAS), 2016 14th IEEE International*, 2016, pp. 1–4.
[30] D. Grois, T. Nguyen, and D. Marpe, “Coding efficiency comparison of AV1/VP9, H. 265/MPEG-HEVC, and H. 264/MPEG-AVC encoders.,” in *PCS*, 2016, pp. 1–5.
[31] P. Topiwala, M. Krishnan, and W. Dai, “Performance comparison of VVC, AV1 and HEVC on 8-bit and 10-bit content,” in *Applications of Digital Image Processing XLI*, 2018, vol. 10752, p. 107520V.
[32] M. A. Layek *et al.*, “Performance analysis of H. 264, H. 265, VP9 and AV1 video encoders,” in *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*, 2017, pp. 322–325.
[33] “HEVC Reference Software,” 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/.
[34] F. Bossen and others, “Common test conditions and software reference configurations,” *JCTVC-L1100*, vol. 12, 2013.
[35] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves,” *VCEG-M33*, 2001.

APPENDIX

LIST OF FIGURES

LIST OF TABLES