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

VIDEO CODING ALGORITHM AND OPTIMIZATION TECHNIQUES

Soufleri Efstathia

MASTER THESIS

Supervisor

Loukopoulos Athanasios, Assistant Professor at the Department of Computer Science and Biomedical Informatics, University of Thessaly

Lamia, 7/2017

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

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

VIDEO CODING ALGORITHMS AND OPTIMIZATION TECHNIQUES

Soufleri Efstathia

MASTER THESIS

Supervisor
Loukopoulos Athanasios, Assistant Professor at the Department of Computer Science and Biomedical Informatics, University of Thessaly

Lamia, 7/2017

«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 «Video coding algorithms and optimization techniques» 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 ΔΗΛ~~ΩΝ~~/-ΟΥΣΑ

Ημερομηνία: 12/07/2017

Υπογραφή

VIDEO CODING ALGORITHMS AND OPTIMIZATION TECHNIQUES

Soufleri Efstathia

Three-member Committee:

Loukopoulos Athanasios, Assistant Professor at the Department of Computer Science and Biomedical Informatics, University of Thessaly (supervisor)

Stamoulis George, Professor at the Department of Electrical and Computer Engineering, University of Thessaly

Eumorfopoulos Nestoras, Assistant Professor at the Department of Electrical and Computer Engineering, University of Thessaly

Contents

CHAPTER 1: INTRODUCTION

It is generally believed that videos have an important role in people's lives. A great number of people are fond of watching videos for many purposes such as entertainment, work and so on. Real-time video chat, home remote surveillance and wearable cameras are widely used by people from all over the world. As a result, it is emerging to have available more videos in better quality. In order to achieve this goal, it is necessary to use a standard which is able to maintain as much as it is possible the quality and compress a great number of data at the same time. This is the reason why a new standard which is called High Efficiency Video Compression (HEVC) has been developed. The first version of HEVC was released in January 2013 and a few months of the same year later it was finally approved and published. HEVC is able to improve the coding efficiency as its most important feature is its compression capability. In addition to this, a great number of features have been incorporated in HEVC. These features concern practical issues such as the range, the color and the precision and they are thought to be of utmost importance not only for the scientific community but also for the everyday people. The main characteristic, which distinguishes HEVC from his predecessor H.264/AVC, is the ability to compress large amount of data without losing quality. In fact, it is able to code two times more data maintaining the same quality in comparison with H.264/AVC.

HEVC is considered to be a huge project with a great number of people participating in it. For the development of HEVC a great number of people participated in the project. About 300 meeting took place and more than 1000 documents were released during them. The meetings happened in a daily basis and at the end of them, the meeting notes were published. In this way, people who were not participating at the meetings but were working on video coding were able to contribute on the project. Not only the participants were working between the meetings, but also the members were making their proposals with mail. The whole community working on video compression was participating actively. Both the industry and the academic community were participating in the development of the new standard [50].

HEVC was developed after of a great number of decades of efforts in video coding technology. The main reason of HEVC's development was the satisfaction of the emerging needs. The first one was the need for compression of a growing amount of data. The uncompressed video leads to great amount of data which are extremely difficult to be stored. As a result, it was needed to develop a standard which could achieve better coding performance than the old one. Furthermore, nowadays, new services have been developed in the domain of video. This means that there was a need for higher video quality with higher resolution, greater precision, higher dynamic-range and wider range of representable colors. For example, the Ultra High Definition television incorporates all these characteristics.

The main goal is to reduce the cost of the tiles. Tiles will be explained in detail in the rest of the thesis and in short tiles are used for parallelization. The cost of a tile is the sum of the elements of the region of the matrix consisting the tile. The elements of the matrix are always positive. There have been proposed several algorithms in order to minimize the maximum tile cost [1] [28]. In this thesis, we propose a heuristic algorithm which is able to minimize the maximum tile cost. We compare our results with an uniform algorithm using simulation.

Our contributions include the following:

- The proposed algorithm can achieve the minimization of the maximum rectangle cost of a matrix divided into rectangles
- It can be implemented to reduce the maximum weight of the tiles in HEVC

The rest of the thesis is organized as follows: Chapter 2 provides a brief overview of HEVC. Chapter 3 illustrates the proposed algorithms which are experimentally evaluated in Chapter 4. Finally, Chapter 5 summarizes the thesis and gives the conclusions and future work.

CHAPTER 2: OVERVIEW OF HEVC

The standard specifies the bitstream syntax and the result of decoding process. This gives the developers of encoder and decoder products have as much freedom as possible. Encoding process is not specified in the standard. It is essential to mention that the encoding algorithm has a major role in the efficiency of compression. In fact, the encoding algorithm determines the values of the syntax elements written in the bitstream. This means that the coding modes, prediction and quantization parameters and quantization indices for the transform coefficients are defined by the algorithm.

In this point, it is important to explain the syntax that HEVC includes and it is considered to be high-level. This means that the structure of the bitstream as well as the signaling of high-level information that applies to one or more entire slices or pictures of a bitstream. For instance, the high-level syntax indicates the spatial resolution of the video, the coding tools used and describes random access functionalities of the bitstream. In addition to the signaling of syntax elements, the high-level tool decoding processes associated with the syntax elements are also considered to be included in the high-level syntax part of the standard [23]. Example high-level syntax decoding processes include reference picture management and the output of decoded pictures. This is depicted in figure 1. Input pictures are given as an input to an encoder that encodes the pictures into a bitstream [50].

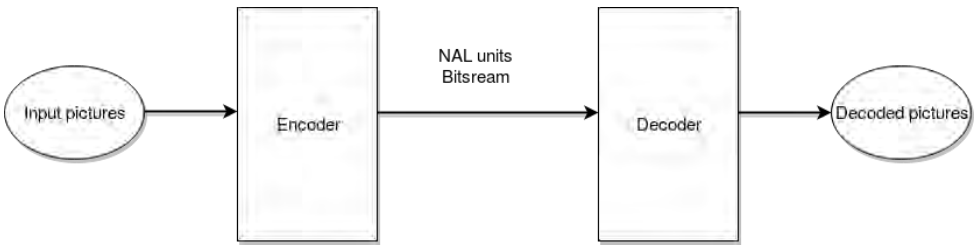

Figure 1: The process of encoding and decoding

An HEVC bitstream consists of a sequence of data units called network abstraction layer (NAL) units, each of which contains an integer number of bytes. The first two bytes of a NAL unit constitutes the NAL unit header, while the rest of the NAL unit contains the payload data. Some NAL units carry parameter sets containing control information that apply to one or more entire pictures, while other NAL units carry coded samples within an individual picture. The NAL units are decoded by the decoder to produce the decoded pictures that are output from the decoder. Both the encoder and decoder store pictures in a decoded picture buffer (DPB). This buffer is mainly used for storing pictures so that previously coded pictures can be used to generate prediction signals to use when coding other pictures [26]. This is illustrated in figures 3 and 4. These stored pictures are called reference pictures. Each picture in HEVC is partitioned into one or multiple slices. It is essential to mention that each slice is independent of other slices and the information carried in the slice is coded without any dependency on data from other slices within the same

picture [38] [39]. A slice consists of one or multiple slice segments, where the first slice segment of a slice is called independent slice segment and is independent of other slice segments. The subsequent slice segments, if any, are called dependent slice segments since they depend on previous slice segments [50]. This partitioning is shown in figure 2.

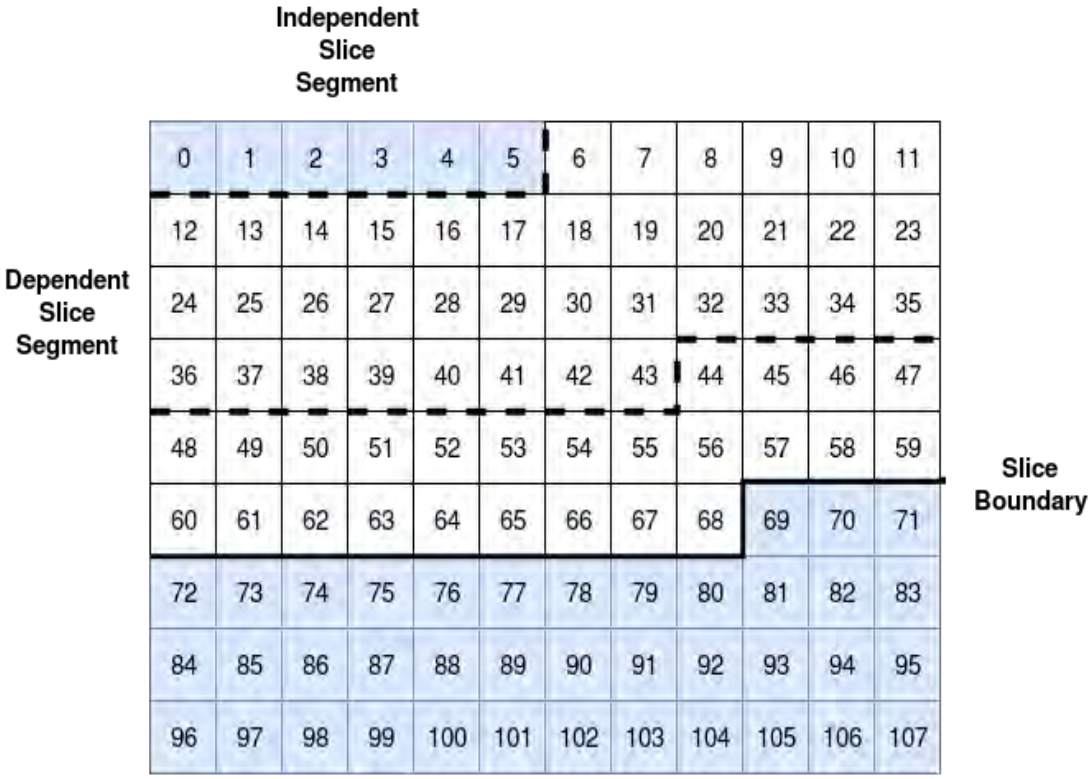

Figure 2: The partitioning of a frame into 2 slices and their independent and dependent slice segment

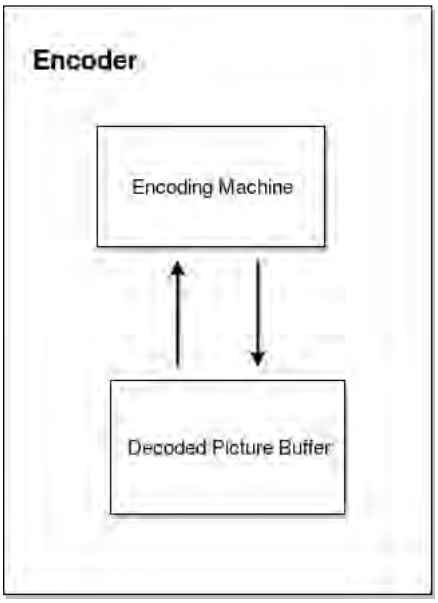

Figure 3: The encoding machine and the decoded picture buffer (DPB) inside the encoder

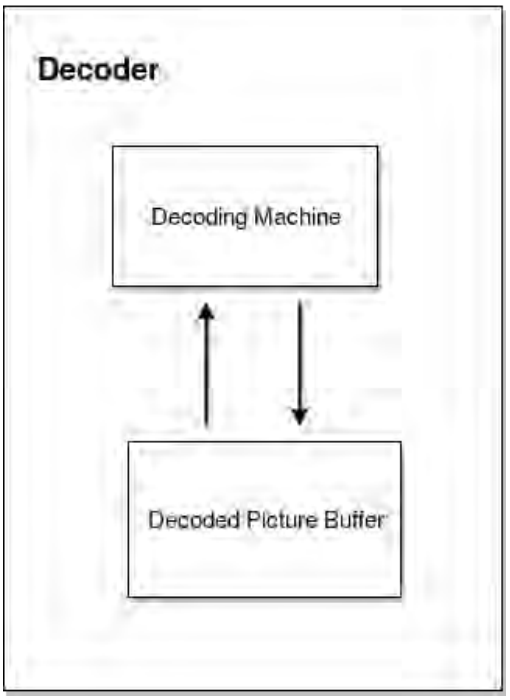

Figure 4: The decoding machine and the decoded picture buffer (DPB) inside the decoder

The development of HEVC was based on the idea of block-based hybrid video coding. This means that the picture is partitioned into blocks and then we are able to process them. In HEVC, firstly each picture is divided into squares of the same size which are called Coding Tree Blocks (CTB) and they consist the first partition of the quadtree structure (root). In other words, picture is considered as a group of blocks [40]. Then, the Coding Tree Blocks can be subdivided into Coding Blocks (CBs) which are divided into Coding Tree Units (CTUs). CTUs consist the basic unit of HEVC. If it is needed the CTUs are subdivided into Coding Units (CUs) in order to make easier their process in later steps during the compression. Also, one more division is that of each picture into slices and then the slices are divided into slice segments in order to be processed in a parallel way [17] [50].

HEVC has a great number of improved features which enable it capable to achieve a good compression performance. In order to compress a picture or a block, HEVC offers many possibilities. The precision of the motion vector has been increased, offering the chance to have more accurate motion vectors. In addition to this, the order of coding pictures is more flexible and we are no more forced to follow a specific order which may be inconvenient for the coding efficiency [11]. The set of the reference picture has been increased as well as the number of the motion vectors predictors. Last but not least, the number of intra-picture prediction modes, the number of transform sizes and the number for block sizes for motion-compensated prediction has been increased.

All these improvements are beneficial in order to limit the rate-distortion. Rate-distortion is considered to be the amount of lost information. We have so many choices that the rate-distortion is minimized. However, we have an increased bit rate. In fact, there is a trade-off between distortion and bit rate. For instance, a block can be divided with different ways and when we choose the best prediction for each subdivision we decrease the residual energy. The difference between the prediction and the current block is the least possible. On the other hand, we increase the bit rate as more data have to be stored so as to signal the division and the prediction mode used [44] [50] [51].

It worth to be mentioned that in the encoder there is a trade-off between the complexity and the compression efficiency. When the set of possible subdivision and prediction modes is getting larger, the computational complexity is increased. The encoder has to make computations in more data than previously. Each of these data is bigger and more complicated. In this way, the computational cost is getting higher [14]. In contrast, the coding efficiency is increased as the predictors are more accurate. The prediction matches better with the current block and the next stage of video process are conducted in an easier way [33].

Another improvement is considered to be the support of larger blocks. Some picture regions are described with the same motion parameters. As a result, we don't have to signal these parameters for each region independently, but consider all this region as a block and signal once the motion parameters. HEVC was mainly developed to deal with high resolutions [50]. In high resolution pictures, there is high correlation among the adjacent blocks. This leads to leads to the conclusion that larger block size is needed. As a consequence, HEVC supports larger block size and specifically blocks of 64×64 size. In figure 5, the partition of a picture with CTUs of size 64×64 is depicted. However, the large block size has some drawbacks. The encoder and decoder delay is increased as more data is being signaled among the stages of process. In addition to this, these data must be stored and thus they require more memory than the smaller blocks. What is

more, the computational cost is higher. More computations have to take place and this increases the computational complexity [7] [50].

Figure 5 :A frame divided into CTUs of size 64 × 64

At the same time, HEVC has to deal with low resolutions where so large blocks are thought to be inefficient as they have a negative impact. That's why HEVC supports variable block sizes and selects which are more efficient each time [12]. Of course, the when a block is considered small, it cannot be subdivided into smaller blocks. The smaller block supported by HEVC is of size 4×4 and cannot be further divided [41] [42].

As it was already mentioned, HEVC uses a hierarchical partitioning. In HEVC a picture is partitioned into blocks and then each block is predicted using either intra-picture or inter-picture prediction method. By using intra-picture as prediction method, we are searching for predictor in the decoded blocks within the picture. By using inter-picture as prediction method, we are searching for predictor in the decoded blocks of previous or next pictures. Then, we compute the difference between the original block and its prediction. This difference is called prediction error and it is used in the next steps of video coding. In the next step, the prediction is transformed and the resulting coefficients of transform are quantized. The last step of video compression is the entropy coding of these coefficients.

The stages of the HEVC encoder process includes are described briefly in the following sections.

Intra-picture prediction mode

In intra-picture prediction, it is essential to achieve high compression efficiency while minimizing the computational effort of both the encoder and the decoder. Intra-picture prediction takes advantage of the spatial redundancy and as a result the reference blocks are reconstructed neighboring blocks existing in the same picture, as it is shown in figure 6. HEVC supports a wide range of modes and sizes for the prediction blocks. All prediction modes of intra-picture prediction are available for all block sizes. Especially, the intra prediction modes are 35 and the prediction block sizes are the following: 4×4, 8×8, 16×16 and 32×32 [50].

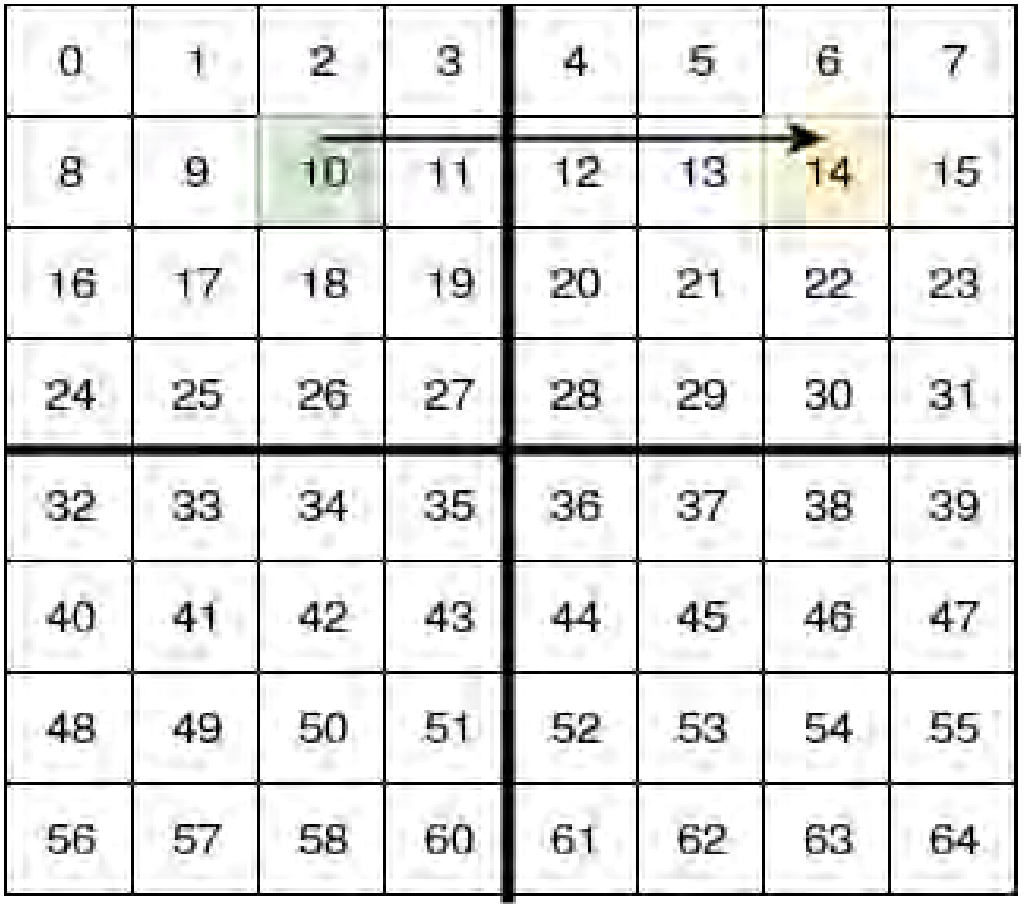

Figure 6: Intra-picture prediction mode

In addition to this, the prediction methods focus on the block which is about to be predicted taking into consideration whether image is smooth and has gradually changing content or has edges and different directional structures. For the first category, planar and DC prediction are used whereas for the second one angular prediction is used. In the case of DC prediction, the predicted values are populated with a constant value. This value is computed as the average of the reference samples which are immediately left and above of the block which is predicted [16]. Also, there is used a filter to soften the edges of luminance blocks of size 16×16 or smaller so as to soften the left and above edge. In the case of planar prediction, we use the average of a horizontal and vertical prediction in order to prevent discontinuities on the block boundaries. In the case of angular prediction, a set of available prediction directions is selected. This selection takes into consideration the trade-off between the coding efficiency and the encoding complexity. As in HEVC exist 4 different block sizes and each of them supports 33 distinct directions, the decoder must support 132 combinations of block size and prediction direction. In order to obtain the predicted samples, we project their location to the reference samples and select the prediction direction. Then, we interpolate the two closest reference samples [32] [50].

Another important advantage that HEVC has over his predecessor H.264/AVC is the fact that in HEVC we can use all prediction modes without caring whether the reference samples are available. For instant, the reference samples may be outside the frame and thus they are considered to be unavailable. In H.264/AVC, the modes needed these samples were omitted. In the new standard, we are able to face this problem. If the reference samples are available then they are used in the prediction without any change, but if they are unavailable then HEVC can construct the missing samples using special techniques [18]. By using these techniques when a reference sample is unavailable then it is substituted by scanning in clock-wise direction and using the latest available sample's value. Of course, for this technique it is necessary to exist at least one reference sample available. If none reference sample is available, then all reference samples are substituted by a nominal average samples value. This value depends on the given bit depth [30] [36] [50].

Furthermore, the reference samples are filtered in order to improve the appearance of the extremely smooth images. This filter is applied by taking into account the prediction mode and the size of the prediction block. More specifically, the smoothing filter is not applied for DC prediction mode and for prediction blocks of size 4×4. For the other cases, the filter is applied depending on the block size and the directionality of the prediction. This means that for 16×16 prediction blocks, the smoothing filter is used at most cases with exception the near-horizontal and near-vertical directions. In order to achieve the best possible prediction, HEVC supports not only a pre-processing step of filtering the reference samples but also a post-processing step to improve the surface of the samples. In the post-processing step, a filter is applied to remove discontinuities along the block boundaries. The role of this filter is to take into consideration the slope at the edge of the block and replace the values with new ones [43] [50] [53].

HEVC selects the intra mode coding taking into consideration the effectiveness. We should be able to minimize the overhead. In the previous standard H.264/AVC the coding mode was based on the most probable mode. However, this approach in HEVC is inapplicable due to the large amount of data [50]. For the luma components, we use the three most probable modes instead of the single one used by H.264/AVC. The most probable modes are derived by using the information modes of the adjacent Prediction Units (PU). In the stage of prediction, the CUs can be referred as prediction units (PU). The neighboring PU selected are those being left and those being above of the current PU. The signaling of Chroma's prediction mode was based on the observation that

usually both luma and chroma have the same prediction modes. Thus, HEVC introduces a special mode denoted as INTRA_DERIVED and this mode indicates that the chroma PUs use the same prediction mode with the corresponding luma PUs.

It is inferred by the statements above that the amount of data transferred to the encoder for signaling the prediction modes are huge. The computation of the rate distortion is inapplicable for most cases because of the amount of data. For this reason, HEVC uses the sum of absolute transformed differences (SATD) between the prediction and the current samples. In this way, we are able to reduce the number of intra prediction modes and then apply the rate-distortion optimization (RDO). The PU's size determines the number of luma intra mode candidates entering the RDO. We select eight for PUs of size 4×4 and 8×8 and three for the other sizes [37] [50].

Inter-picture prediction mode

Inter-picture prediction is based on the temporal redundancy between the frames. In comparison with intra-picture prediction, inter-picture prediction exploits the temporal correlation and not the spatial correlation. It is used for moving objects between the frames of the video sequence and this is the reason why it is also called motion-compensated prediction. A previously decoded picture is called a reference picture and it is indicated with a reference index. A previously decoded block is referred as a reference block. The reference block has the role of predictor of the current block. Motion vector indicates the position of e previously decoded block and we refer to it using the following 2 values: Δ_x which symbolizes the horizontal displacement relative to the position of the current block and Δ_y which symbolizes the vertical displacement relative to the position of the current block, as shown in figure 7. When the reference block is selected, the motion vector is compensated. As we can have a predictor for the block, similarly we can have a predictor for the motion vector called Motion Vector Predictor (MVP). The motion vector with the reference picture indices consist the motion data [50].

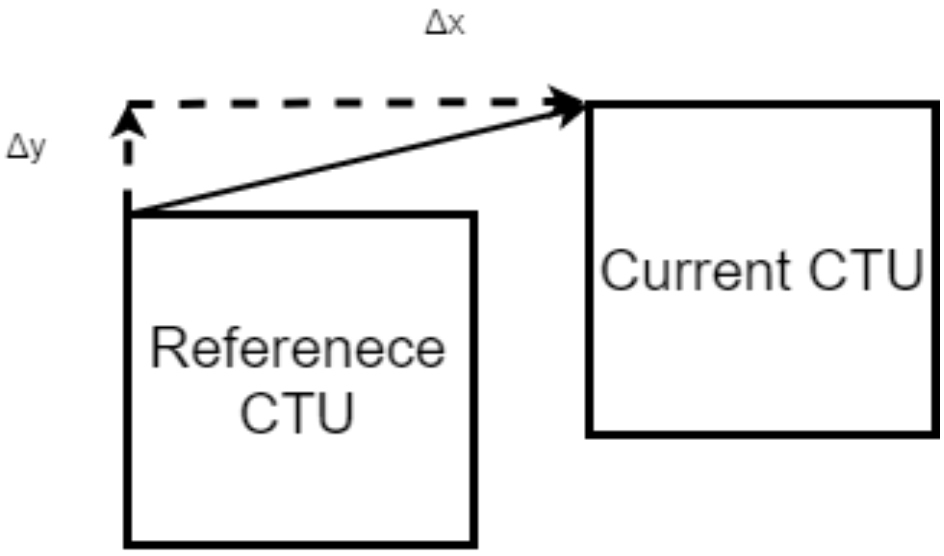

Figure 7: Motion Vector (Δ_x , Δ_y)

HEVC offers the possibility to use two kinds of inter-picture prediction. The first one is called uni-prediction and the reference block is selected among the blocks of a previous coded picture, as shown in figure 8. The second one is the bi-prediction and the reference block is formed by two blocks which are usually in different frame, as depicted in figure 9. These two pictures are stored in two different lists, namely list 0 and list 1. When the second kind of prediction is used, by default the reference block is computed by taking the average of the two blocks. However, the HEVC standard offers the chance to change this option and use the weighted prediction in which different weights are applied in each block [46] [48] [50].

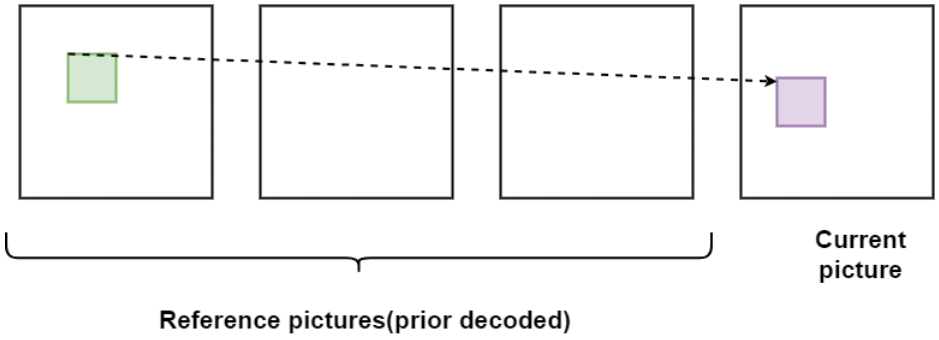

Figure 8: Inter-picture prediction using uni-prediction as prediction method

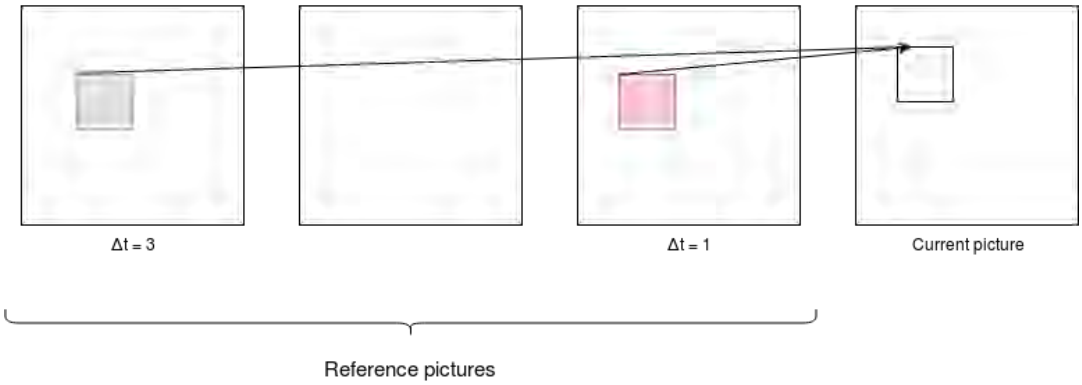

Figure 9: Inter-picture prediction using bi-prediction as prediction method

Another important advantage of HEVC is a new tool called Advanced Motion Vector Prediction (AMVP). This tool's task is to signal the best prediction for each motion block to the decoder. The motion vector is decoded using the difference between the horizontal components of motion vector and the horizontal components of the motion vector predictor and the difference between the vertical components of motion vector and the vertical components of the motion vector predictor [19] [22] [50]. The main goal is to reduce the signaled size of the motion vector difference. This goal can be achieved by finding the best possible predictor and as adjacent motion vectors are highly correlated, we will search for MVP in the neighboring motions vectors. This statement can be justified by the fact that the motion vector for a moving object is more likely not be extremely different from the motions vectors nearby the object. In addition to this, it is essential to mention that HEVC uses a technique known as motion vector competition. This technique signals which MVP from the list of the MVP will be used. In the initial design, AMVP included five MVPs. Three of the MVPs were from spatial neighbors, one from their median and the last one from a scaled co-located motion vector from the temporally adjacent block. Then, these MVP were reordered based on the possibility to match better the motion vector. During this reordering, some motion vectors were removed to ensure that the signaling overhead is not being maximized. However, experiments were conducted and led to the conclusion that the mount of MVP increased the complexity. As a consequence, the developers of the standard decided to reduce the amount of MVP in the list from five to two and fix the candidate order in the list. The two MVP candidates are: up to two spatial candidate MVPs from the five spatial neighboring blocks, one temporal

candidate MVP obtained from the two temporal, co-located blocks and zero motion vectors when the other candidates are unavailable [47] [50].

The picture is consisted by different regions with different characteristics. Thus, it is reasonable to believe that we have to implement different prediction methods in each region. So, the regions should be partitioned in sub-blocks using the quadtree structure. This structure has the advantage to have low cost using as criterion the bit rate and permits a wide range of block sizes. On the other hand, this partitioning poses a great number of problems such as the signaling of redundant information. HEVC faces this problem with block merging. Block merging can code motion parameters without using large number of bits [8] [50].

The merge mode for each PU is indicated by using "*merge_flag*" to signal that this mode is used. "*merge_idx*" indicated the candidate in the merge list. This means that only a flag and an index is used for the transmission in comparison with AMVP. The number of candidates in the merge list is signaled in the slice header. By default, we have five values and for this reason it is represented by using the difference to five and the syntax element is denoted as *five_minus_max_num_merge_cand*. In the initial design, the maximum value for the merge index coding was given by the number of available spatial and temporal candidates in the list. However, during the parsing of the merge index it was necessary the whole list to be constructed in order to know the actual number of candidates. If the adjacent blocks were unavailable because of errors during the transmission, then it was impossible to parse the merge index of the remaining blocks [20]. This problem can be tackled using fixed number of merge candidates. Thus, the parsing robustness is improved and the merge candidate list can be constructed achieving as much as possible efficiency [50] [54].

Another important aspect of HEVC is the way which it combines the merging mode with the skip mode. In H.264/AVC the skip mode was used to signal when the prediction residual was zero and thus no transform coefficients were transmitted. This mode is usually used to code static image regions were the prediction error is thought to be small. At the beginning of each CU, a *skip_flag* is signaled to indicate that CU contains one PU, the merge mode is used to derive the motion data and as a result *merge_flag* = 1 and no residual data is present in the bitstream.

The compression capability of the merge and skip mode have been evaluated. This achieved by conducting experiments disabling either the merge mode or the skip mode. The disabling of the merge mode can be done by removing the *merge_flag* and the *merge_index* syntax. The disabling of the skip mode can be done by removing the *skip_flag* syntax. The average saving was between 6% and 8%. The results supported the idea that we had achieved better results with random access main (RA-Main) and low delay B main (LD-Main) configurations using bi-predictive B pictures [50].

Interpolation is important for video coding as the displacements of objects from picture to picture are independent of the sampling grid cameras. Thus, in MCP fractional-sample accuracy is used. We take samples at integer positions and we estimate the values at fractional positions. Filters for luma and chroma interpolation were redesigned and the tap-lengths were increased. The luma interpolation process in HEVC uses asymmetric 8-tap filter for the half-sample positions and an asymmetric 7-tap filter for quarter-sample positions. For chroma samples, a 4-tap filter was introduced [50].

Having as a goal the improvement of the coding performance, HEVC offered the possibility to keep at a higher accuracy the intermediate values used in interpolation process. H.264/AVC obtains the quarter-sample values by first obtaining the values of the nearest integer samples according to the position of the quarter-pixel. In contrast, HEVC obtains the quarter-pixel samples without using such cascaded steps. Instead of this, HEVC directly applies a 7 or 8-tap filter on the integer pixels. In addition to his, in H.264/AVC, a bi-predictively coded block is calculated by averaging two uni-predicted blocks. If interpolation is performed to obtain the samples of the uni-prediction blocks, those samples are shifted and clipped to input bit-depth after interpolation, prior to averaging [50]. On the other hand, HEVC keeps the samples of each one of the uni-prediction blocks at a higher accuracy and only performs rounding to input bit-depth at the final stage, improving the coding efficiency by reducing the rounding error.

In H.264/AVC, some of the intermediate values used within interpolation are shifted to lower accuracy, which introduces rounding error and reduces coding efficiency. This loss of accuracy is due to several reasons. Firstly, the half-pixel samples obtained by 6-tap FIR filter are first rounded to input bit-depth, prior to using those for obtaining the quarter-pixel samples. Instead of using a two-stage cascaded filtering process, HEVC interpolation filter computes the quarter-pixels directly using a 7-tap filter using the coefficients which significantly reduces the rounding error to $^{1}/_{128}$. Also, the reduction of accuracy in H.264/AVC motion compensation process is due to averaging in bi-prediction [50]. In H.264/AVC, the prediction signal of the bi-predictively coded motion blocks is obtained by averaging prediction signals from two prediction lists. In HEVC, instead of averaging each prediction signal at the precision of the bit-depth, they are averaged at a higher precision if fractional motion vectors are used for the corresponding block. It should be noted that for the cases where one of the prediction signal is obtained without interpolation (i.e. the corresponding motion vector has an integer pixel accuracy) the bit-depth of the corresponding prediction signal is first increased accordingly before bi-prediction averaging so that both prediction signals are averaged at the same bit-depth.

The developers of HEVC have taken into consideration that the intermediate values do not overflow the 16-bit registers, after horizontal interpolation the intermediate values are shifted to the right by bit depth minus 2. This means that when the bit depth of the video is more than 8 bits, the order in which horizontal filtering and vertical filtering is done needs to be specified (horizontal first in HEVC). This specific order was selected mainly to simplify implementation on specific architectures [50]. It should also be noted that in HEVC the only clipping operation is at the very end of the motion compensation process, with no clipping in intermediate stages. As there is also no rounding in intermediate stages, HEVC interpolation filter allows certain implementation optimizations. Consider the case where bi-prediction is used and motion vectors of each prediction direction points to the same fractional position. In these cases, final prediction could be obtained by first adding two reference signals and performing interpolation and rounding once, instead of interpolating each reference block, thus saving one interpolation process.

When evaluating the complexity of a video coding algorithm a great number of features were taken into account. These features concern the memory bandwidth, the number of operations and the storage buffer size. In terms of memory bandwidth, utilizing longer tap filters in HEVC (7–8 tap filter for luma sub-pixels and 4-tap filter for chroma sub-pixels) compared to shorter filters in H.264/AVC (6-tap filter for luma sub-pixels and bilinear filter for chroma) increases the amount of data that needs to be fetched from the reference memory. The worst case happens when

a small motion block is bi-predicted and its corresponding motion vector points to a sub-pixel position where two-dimensional filtering needs to be performed. In order to reduce the worst-case memory bandwidth, HEVC introduces several restrictions. Firstly, the smallest prediction block size is fixed to be 4 × 8 or 8 × 4, instead of 4 × 4. In addition, these smallest block sizes of size 4!8 and 8 × 4 can only be predicted with uni-prediction. With these restrictions in place, the worst-case memory bandwidth of HEVC interpolation filter is around 51% higher than that of H.264/AVC. The increase in memory bandwidth is not very high for larger block sizes. For example, for a 32×32 motion block, HEVC requires around 13% increased memory bandwidth over H.264/AVC. HEVC uses 7-tap FIR filter for interpolating samples at quarter-pixel locations, which has an impact on motion estimation of a video encoder. An H.264/AVC encoder could store only the integer and half-pel samples in the memory and generate the quarter-pixels on-the-fly during motion estimation. This would be significantly costlier in HEVC because of the complexity of generating each quarter-pixel sample on-the-fly with a 7-tap FIR filter. Instead, an HEVC encoder could store the quarter-pixel samples in addition to integer and half-pixel samples and use those in motion estimation. Alternatively, an HEVC encoder could estimate the values of quarter-pixel samples during motion estimation by low complexity non-normative means [34] [50].

Similar to H.264/AVC, HEVC includes a weighted prediction (WP) tool that is particularly useful for coding sequences with fades. In WP, a multiplicative weighting factor and an additive offset are applied to the motion compensated prediction. Care is taken to handle uni-prediction and bi-prediction weights appropriately using the flags *weighted_pred_flag* and *weighted_bipred_flag* transmitted in the Picture Parameter Set (PPS). Consequently, WP has a very small overhead in PPS and slice headers contain only non-default WP scaling values. WP is an optional PPS parameter and it may be switched on/off when necessary. The inputs to the WP process are: the width and the height of the luma prediction block, prediction samples to be weighted, the prediction list utilization flags, the reference indices for each list, and the color component index. Weighting factors w0 and w1, and offsets o0 and o1 are determined using the data transmitted in the bitstream. The subscripts indicate the reference picture list to which the weight and the offset are applied. The output of this process is the array of prediction sample values [4] [50].

Transform

The residual or prediction error obtained by either intra-picture prediction or inter-picture prediction is transformed. Transform is mainly used in order to de-correlate the residual. Specifically, the residual signal is divided in residual blocks (U) of size N×N where N is a multiple of two. The residual block is given as an input to a two-dimensional N×N forward transform. There exist two kinds of transform: the core transform based on discrete cosine transform (DCT) and the alternate transform based on discrete sine transform (DST). In HEVC, the core transform matrices can be of size 4×4, 8×8, 16×16 and 32×32. The choice of the size of the transform's matrix is considered to be of utmost importance as there is a trade-off between compression efficiency and implementation complexity [50].

The HM1 inverse transforms had the following properties [50]:

- Non-flat de-quantization matrices for all transform sizes: While acceptable for small transform sizes, the implementation cost of using de-quantization matrices for larger transforms is high because of larger block sizes

- Different architectures for different transform sizes: This leads to increased area since hardware sharing across different transform sizes is difficult

- A 20-bit transpose buffer used for storing intermediate results after the first transform stage in 2D transform: An increased transpose buffer size leads to larger memory and memory bandwidth. In hardware, the transpose buffer area can be significant and comparable to transform logic area

- Full factorization architecture requiring cascaded multipliers and intermediate rounding for 16- and 32-point transforms: This increases data path dependencies and impacts parallel processing performance. It also leads to increased bit width for multipliers and accumulators (32 bits and 64 bits respectively in software). In hardware, in addition to area increase, it also leads to increased circuit delay thereby limiting the maximum frequency at which the inverse transform block can operate. To address the complexity concerns of the HM1 transforms, a matrix multiplication based core transform was proposed in [8] and eventually adopted as the HEVC core transform. The design goal was to develop a transform that was efficient to implement in both software on SIMD machines and in hardware.

The core transform matrixes of HEVC were designed in such a way that they are satisfying some special properties. The most important one is the closeness to the IDCT. Furthermore, basis vectors should be almost orthogonal, their norm should be almost equal and have the same properties with the IDCT vectors. Also, larger transform matrixes contain smaller ones and the elements of the transform matrices are represented by using eight-bit. Another important property is that the transpose matrix should consist of sixteen-bit and that accumulators can be implemented using less than 32 bits. Last but not least, 16 bits or less must be used for the representation of the multipliers and avoid having cascaded multiplication or intermediate rounding [45] [50].

As far as DCT is concerned its properties are considered to be of utmost importance for coding efficiency and efficient implementation. The first and most important property is that the basis vectors are orthogonal. Because of this property, the coefficients obtained by the transform are uncorrelated and this characteristic is very beneficial for the video compression. Another property beneficial for the video compression is that the energy compaction of the basis vectors of DCT is considered to be good. What is more, basis vectors of DCT have equal norm and this is really

useful for quantization and de-quantization process and will be described in detail later. In short, with this property, the need for quantization and de-quantization matrices is reduced. In addition to this, the elements of a DCT matrix of size $2^{M+1}\times 2^{M+1}$ contains the elements of the DCT matrix of size $2^{M}\times 2^{M}$. In other words, the basis vectors of the smaller matrix is equal to the first half of the even basis vectors of the larger matrix. This is thought to increase the implementation efficiency because the same multipliers can be reused for different transform matrices. Furthermore, the number of arithmetic operations are reduced because the even basis vectors of the DCT are symmetric and the odd basis vectors of the DCT are anti-symmetric. It important to be mentioned that the DCT matrix has a great number of advantages in hardware implementation. This means that we are able to determine the DCT matrix using a small number of unique elements. Last but not least, the number of operations can be reduced using the trigonometric relationships that the coefficients have in combination with the symmetric and anti-symmetric properties [50].

The core transform matrices of HEVC have finite precision approximation of the DCT matrix. In this way, the approximation of the real values is specified in the standard and it is not implemented independently. The main advantage is that the encoder and the decoder have the same values and we avoid problems created by using different floating point representations. On the other hand, there is one main drawback when we are using finite precision approximation. Some of the properties are violated and as a result the is a trade-off between the computational cost and the number of properties which are satisfied. One solution proposed in order to achieve integer approximations to the DCT matrix is to scale the elements of the matrix with a large number and then round the obtained values to the closest integer. This technique, however, has a bad impact on the coding efficiency [2] [50].

It is inferred that the only way to minimize the computational cost is to select which properties will be satisfied and which ones will be violated. The property about the elements of a DCT matrix of size $2^{M}\times 2^{M}$ that are subset of the elements of the DCT matrix of size $2^{M+1}\times 2^{M+1}$, about the symmetry of the even basis vectors of the DCT and the anti-symmetry of the odd basis vectors of the DCT as well as the determination of the DCT matrix using a small number of unique elements must always be satisfied [2]. The rest of the properties are not compulsory to be satisfied. This means that sometimes when are willing to use a great number of bits to represent each element of the matrix and thus increase the computational effort, whereas other times we prefer to minimize the computational complexity. For the first three properties, approximation metrics have been defined so as to measure the degree of approximation. These measures concern the orthogonality, the closest to DCT measure and the norm [50].

Since the HEVC matrices are scaled by 2 compared to an orthonormal DCT transform, and in order to preserve the norm of the residual block through the forward and inverse two-dimensional transforms, additional scale factors denoted as ST1, ST2, SIT1, SIT2, need to be applied. While the HEVC standard specifies the scale factors of the inverse transform, i.e. SIT1, SIT2, the HEVC reference software also specifies corresponding scale factors for the forward transform, i.e. ST1, ST2. The scale factors were chosen with the following constraints [50]:

- All scale factors shall be a power of two to allow the scaling to be implemented as a right shift.

- Assuming full range of the input residual block (e.g. a DC block with all samples having maximum amplitude), the bit depth after each transform stage shall be equal to 16 bits (including the sign bit). This was considered a reasonable tradeoff between accuracy and implementation costs.

- Since the HEVC matrices are scaled by 2, cascading of the two dimensional forward and inverse transform will result in a scaling of 2 for each of the 1D row forward transform, the 1D column forward transform, the 1D column inverse transform, and the 1D row inverse transform.

The alternate transform provides around 1 % bit-rate reduction while coding intra pictures. In intra-picture prediction, a block is predicted from left and/or top neighboring samples. The prediction quality is better near the left and/or top boundary resulting in an intra-prediction residual that tends to have lower amplitude near the boundary samples and higher amplitudes away from the boundary samples. The DST basis functions are better than the DCT basis functions in modeling this spatial characteristic of the intra prediction residual. This can be seen from the first row (basis function) of the alternate transform matrix which increases from left to right as opposed to the DCT transform matrix that has a flat first row. During the course of the development of HEVC, alternate transforms for transform block sizes of 8×8 and higher were also studied. However, only the 4×4 alternate transform was adopted in HEVC since the additional coding gain from using the larger alternate transforms was not significant. It is also important to mention that their complexity is higher since there is no symmetry in the transform matrix and a full matrix multiplication is needed to implement them for transform sizes 8×8 and larger [50].

Quantization

In quantization, the coefficients derived by the transform are divided by a quantization step size (Qstep) and then the result is rounded. In HEVC, the quantization step is determined by a quantization parameter (QP). This parameter can take 52 values varying from 0 to 51 when the video sequence is 8-bit. We can define the relative difference as the difference between the step-sizes of two consecutive QP values. The absolute step-size difference is highly dependent on the range of the QP values and can be achieved by posing Qstep = 1 and QP = 4 [50].

HEVC utilizes quantization matrices for all transform block sizes as its quantization depends on the frequency. We can have a quantization matrix with weights. We use quantization multipliers denoted as f_i in the encoder and de-quantization multipliers denoted as g_i in the decoder. These multipliers are selected in such a way that they satisfy specific conditions. First of all, it is necessary for the g_i to be represented using the signed 8-bit data type. Secondly, the step size should increase equally between the QP values. Furthermore, the benefit of quantization and de-quantization should be approximately equal. Last but not least, the desired absolute value of the quantization step size for QP = 4 should be achieved [50].

The encoder in HEVC signals if quantization matrices will be used or not. HEVC supports 20 quantization matrices and they are categorized by taking into account the block size and the type of the transform block. More specifically [50]:

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

The syntax element "*scaling_list_enabled_flag*" enables the frequency dependent scaling. By default, the quantization matrices of size 4×4 and 8×8 have values. Then, the 8×8 quantization matrices are up sampled using replication and are used to form the quantization matrices for transform blocks of size 16×16 and 32×32. In addition to this, non-default quantization matrices are transmitted in the bitstream of the sequence parameter set (SPS) or the picture parameter set (PPS) [50].

Quantization step size and thus the QP value may be changed within the picture. To achieve this, HEVC permits the signaling of a quantization group (QG) including a delta QP value. The QG size is multiple of the coding unit size. More specifically, QG can take values from 8×8 to 64×64. The delta QP is signaled in coding units with non-zero transform coefficients. There are two possible scenarios [50]:

- The CTU is divided into coding units greater than the QG size. Then, the delta QP is signaled at a coding unit which is greater than the QG size
- The CTU is divided into coding units smaller than the QG size. Then, the delta QP is signaled in the first coding unit which has non-zero transform coefficients. In the extreme case that all the transform coefficients are zero, the delta QP is not being signaled.

QP values from the left, above and previous QG are used as QP predictor for computing delta. Two predictive techniques are mainly used: spatial QP prediction and previous QP prediction. Spatial QP prediction: it uses spatial prediction from left and above within a CTU and uses the previous QP as predictor for the CTU boundary. If the values from the left and the above are unavailable, then they exist in a different CTU or the current QG is at the boundary. The unavailable value is being replaced by the previous QP value in decoding order. It is important to mention that the previous QP is initialized to the slice QP value at the beginning of the slice [21] [50].

HEVC has three special modes that modify the transform and quantization process: I_PCM mode, lossless mode and transform skip mode. These modes skip either the transform or both the transform and quantization [50].

- In the I_PCM mode, both transform and transform-domain quantization are skipped. In addition, entropy coding and prediction are skipped too and the video samples are directly coded with the specified PCM bit depth. The I_PCM mode is designed for use when there is data expansion during coding e.g. when random noise is input to the video codec. By directly coding the video samples, the data expansion can be avoided for such extreme video sequences. The IPCM mode is signaled at the coding unit level using the syntax element *pcm_flag*.

- In the lossless mode, both transform and quantization are skipped. Mathematically lossless reconstruction is possible since the residual from inter- or intra-picture prediction is directly coded. The lossless mode is signaled at a coding unit level (using the syntax element *cu_transquant_bypass_flag*) in order to enable mixed lossy/lossless coding of pictures. Such a feature is useful in coding video sequences with mixed content, e.g. natural video with overlaid text and graphics. The text and graphics regions can be coded losslessly to maximize readability whereas the natural content can be coded in a lossy fashion.

- In the transform skip mode, only the transform is skipped. This mode was found to improve compression of screen-content video sequences generated in applications such as remote desktop, slideshows etc. These video sequences predominantly contain text and graphics. Transform skip is restricted to only 4 × 4 transform blocks and its use is signaled at the transform unit level by the *transform_skip_flag* syntax element.

The Even–Odd decomposition of the inverse transform of an N-point input consists of the following three steps [50]:

- Calculate the even part using a N/2 × N/2 subset matrix obtained from the even columns of the inverse transform matrix.
- Calculate the odd part using a N/2 × N/2 subset matrix obtained from the odd columns of the inverse transform matrix.
- Add/subtract the odd and even parts to generate N-point output.

Entropy encoding

Entropy coding is the last stage of the process of the video encoding. In this stage, the video consists of a series of syntax elements which describe the way by which the video with be reconstructed at the decoder. In the initial design, HEVC used two methods: the Context-Based Adaptive Binary Arithmetic Coding (CABAC) and the low-complexity entropy coding (LCEC). However,

while HEVC was taking its final form, it was observed that LCEC was inapplicable from the aspect of coding efficiency. The complexity of LCEC had been increased to the extent that it was almost equal with CABAC. As a consequence, there was no reason to use LCEC and CABAC was the only used method for entropy encoding in HEVC. In CABAC method the data are compressed using statistical information. In other words, the number of bits required to represent the data must be logarithmically proportional to the probability of the data. It should be reported that the syntax elements that belong to the slice data segment are CABAC encoded. The other high-level syntax elements are encoded with zero-order Exponential-Golomb codes or fixed-pattern bit strings. Also, it is essential to be mentioned that entropy coding is a lossless process [29] [50].

The CABAC algorithm was originally developed within the joint H.264/AVC standardization process of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). In a first preliminary version, the new entropy-coding method of CABAC was introduced as a standard contribution to the ITU-T VCEG meeting in January 2001. CABAC was adopted as one of two alternative methods of entropy coding within the H.264/AVC standard. The other method specified in H.264/AVC was a low-complexity entropy coding technique based on the usage of context-adaptively switched sets of variable-length codes, so called Context-Adaptive Variable-Length Coding (CAVLC). Compared to CABAC, CAVLC offers reduced implementation cost at the price of lower compression efficiency. Typically, the bit-rate overhead for CAVLC relative to CABAC is in the range of 10–16% for standard definition (SD) interlaced material, encoded at Main Profile, and 15–22% for high definition (HD) 1080p material, encoded at High Profile, both measured at the same objective video quality and for the case that all other used coding tools within the corresponding H.264/AVC Profile remain the same. CABAC became also part of the first HEVC test model HM1.0 to get her with the so-called low-complexity entropy coding (LCEC) as a follow-up of CAVLC. Later, during the HEVC standardization process, it turned out that to improve the compression efficiency of LCEC, the complexity of LCEC had to be increased to a point where LCEC was not significantly lower complexity than CABAC. Thus, CABAC in its improved form, both with respect to throughput speed and compression efficiency, became the single entropy coding method of the HEVC standard. The basic design of CABAC involves the key elements of binarization, context modeling, and binary arithmetic coding. Binarization maps the syntax elements to binary symbols (bins). Context modeling estimates the probability of each non-bypassed (i.e., regular coded) bin based on some specific context. Finally, binary arithmetic coding compresses the bins to bits according to the estimated probability [50] [52].

The design of CABAC includes binarization, context modeling and binary arithmetic modeling. The role of binarization is to map the syntax elements to binary symbols. These symbols are called bins and can be interpreted into binary code tree. The binarization process includes k-th order truncated Rice (TRk), k-th order Exp-Golomb (EGk) and fixed-length (FL) binarization. The type of syntax elements is responsible for the choice of the binarization process. In some cases, it depends on the previously processed syntax elements or slice parameters. Context modeling is responsible to estimate the probability of each by-passed bin taking into consideration some specific context. In the end, the bins are compressed into bits by the binary arithmetic coding. During this process, it is used the estimated probability calculated by the context modeling [50].

The coding strategy of CABAC is based on the finding that a very efficient coding of non-binary syntax element values in a hybrid block-based video coder, like components of motion vector differences or transform coefficient level values, can be achieved by employing a binarization scheme as a kind of preprocessing unit for the subsequent stages of context modeling and

arithmetic coding. In general, a binarization scheme defines a unique mapping of syntax element values to sequences of binary symbols, so-called bins, which can also be interpreted in terms of a binary code tree. The design of binarization schemes in CABAC both for H.264/AVC and HEVC is based on a few elementary prototypes whose structure enables fast implementations and which are representatives of some suitable model probability distributions [50].

These various methods of binarization can be explained in terms of how they would signal an unsigned value N [50].

- Unary coding involves signaling a bin string of lengthN C 1, where the first N bins are 1 and the last bin is 0. The decoder searches for a 0 to determine when the syntax element is complete. For the TrU scheme, truncation is invoked for the largest possible value cMax1 of the syntax element being decoded.
- k-th order truncated Rice is a parameterized Rice code that is composed of a prefix and a suffix. The prefix is a truncated unary string of value N >> k, where the largest possible value is cMax. The suffix is a fixed length binary representation of the least significant bins of N; k indicates the number of least significant bins.
- k-th order Exp-Golomb code is proved to be a robust, near-optimal prefix-free code for geometrically distributed sources with unknown or varying distribution parameter.
- Fixed-length code uses a fixed-length bin string with most significant bins signaled before least significant bins.

The binarization process is selected based on the type of syntax element. In some cases, binarization also depends on the value of a previously processed syntax element (e.g., binarization of *coeff_abs_level_remaining* depends on the previously decoded coefficient levels) or slice parameters that indicate if certain modes are enabled (e.g., binarization of partition mode, so-called *part_mode*, depends on whether asymmetric motion partition is enabled). The majority of the syntax elements use the binarization processes as listed above, or some combination of them. However, certain syntax elements (e.g., part_mode and intra_chroma_pred_mode) use custom binarization processes [50].

During the HEVC standardization process, special attention has been put on the development of an adequately designed binarization scheme for absolute values of transform coefficient levels. In order to guarantee a sufficiently high throughput, the goal here was the maximization of bypass-coded bins under the constraint of not sacrificing coding efficiency too much. This was accomplished by making the binarization scheme adaptive based on previously coded transform coefficient levels [50].

By decomposing each non-binary syntax element value into a sequence of bins, further processing of each bin value in CABAC depends on the associated coding mode decision, which can be either chosen as the regular or the bypass mode. The latter is chosen for bins, which are assumed to be uniformly distributed and for which, consequently, the whole regular binary arithmetic encoding (and decoding) process is simply bypassed. In the regular coding mode, each bin value is

encoded by using the regular binary arithmetic coding engine, where the associated probability model is either determined by a fixed choice, based on the type of syntax element and the bin position or bin index (binIdx) in the binarized representation of the syntax element, or adaptively chosen from two or more probability models depending on the related side information. Selection of the probability model is referred to as context modeling. As an important design decision, the latter case is generally applied to the most frequently observed bins only, whereas the other, usually less frequently observed bins, will be treated using a joint, typically zero-order probability model. In this way, CABAC enables selective adaptive probability modeling on a sub symbol level, and hence, provides an efficient instrument for exploiting inter-symbol redundancies at significantly reduced overall modeling or learning costs. Note that for both the fixed and the adaptive case, in principle, a switch from one probability model to another can occur between any two-consecutive regular coded bins. In general, the design of context models in CABAC reflects the aim to find a good compromise between the conflicting objectives of avoiding unnecessary modeling cost overhead and exploiting the statistical dependencies to a large extent. The parameters of probability models in CABAC are adaptive, which means that an adaptation of the model probabilities to the statistical variations of the source of bins is performed on a bin-by-bin basis in a backward-adaptive and synchronized fashion both in the encoder and decoder; this process is called probability estimation. For that purpose, each probability model in CABAC can take one out of 126 different states with associated model probability values p. The two parameters of each probability model are stored as 7-bit entries in a context memory: 6 bits for each of the 63 probability states representing the model probability pLPS of the least probable symbol (LPS) and 1 bit for &MPS, the value of the most probable symbol (MPS). The probability estimator in CABAC is based on a model of "exponential aging" with the following recursive probability update after coding a bin b at time instance t: Here, the choice of the scaling factor ¸ determines the speed of adaptation: A value of ¸ close to 1 results in a slow adaptation ("steady-state behavior"), while faster adaptation can be achieved for the non-stationary case with decreasing. In the design of CABA has been used together with the choice of and a suitable quantization of the underlying LPS-related model probabilities into 63 different states, to derive a finite-state machine (FSM) with tabulated transition rules. This table-based probability estimation method was unchanged in HEVC, although some proposals for alternative probability estimators have shown average bit rate savings of 0.8–0.9%, albeit at higher computational costs. Each probability model in CABAC is addressed using a unique context index (ctxIdx), either determined by a fixed assignment or computed by the context derivation logic by which, in turn, the given context model is specified. A lot of effort has been spent during the HEVC standardization process to improve the model assignment and context derivation logic both in terms of throughput and coding efficiency [50].

HEVC supports two special coding modes, which are invoked on a CU level: the so-called I_PCM mode and the lossless coding mode. Both modes, albeit similar in appearance to some degree, serve different purposes and hence, use different syntax elements for providing different functionalities. A *pcm_flag* is sent to indicate whether all samples of the whole CU are coded with pulse code modulation (PCM), such that prediction, transform, quantization, and entropy coding as well as their counterparts on the decoder side are simply bypassed. The *pcm_flag* is coded with the termination mode of the arithmetic coding engine, since in most cases I_PCM mode is not used, and if it is used, the arithmetic coding engine must be flushed and then resulting CABAC codeword must be byte aligned before the PCMsample values can be written directly into the bitstream with fixed length codewords.12 This procedure also indicates that the I_PCM mode is in particular useful in cases, where the statistics of the residual signal would be such that otherwise,

an excessive amount of bits would be generated when applying the regular CABAC residual coding process. The option of lossless coding, where for coding of the prediction residual both the transform and quantization (but not the entropy coding) are bypassed, is also enabled on a CU level and indicated by a regular coded flag called *cu_transquant_bypass_flag*. The resulting samples of the losslessly represented residual signal in the spatial domain are entropy coded by the CABAC residual coding process, as if they were conventional transform coefficient levels. Note that in lossless coding mode, both in-loop filters are also bypassed in the reconstruction process (which is not necessarily the case for I_PCM), such that a mathematically lossless (local) reconstruction of the input signal is achieved [50].

In the regular, i.e., lossy residual coding process, a different quantizer step size can be used for each CU to improve bit allocation, rate control, or both. Rather than sending the absolute quantization parameter (QP), the difference in QP steps relative to the slice QP is sent in the form of a so-called delta QP. This functionality can be enabled in the picture parameter set (PPS) by using the syntax element *cu_qp_delta_enabled_flag*. In H.264/AVC, *mb_qp_delta* is used to provide the same instrument of delta QP at the macroblock level. *mb_qp_delta* is unary coded and thus requires up to 53 bins for 8-bit video and 65 bins for 10-bit video. All bins are regular coded. In HEVC, delta QP is represented by the two syntax elements *cu_qp_delta_abs* and cu_qp_delta_sign_flag, if cu_qp_delta_enabled_flag in the PPS indicates so. The sign is sent separately from the absolute value, which reduces the average number of bins by half. *cu_qp_delta_sign_flag* is only sent if the absolute value is non-zero. The absolute value is binarized with TrU (cMax=5) as the prefix and EG0 as the suffix. The prefix is regular coded and the suffix is bypass coded. The first bin of the prefix uses a different context than the other four bins in the prefix (which share the same context) to capture the probability of having a zero-valued delta QP. Note that syntax elements for delta QP are only signaled for CUs that have non-vanishing prediction errors (i.e., at least one non-zero transform coefficient). Conceptually, the delta QP is an element of the transform coding part of HEVC and hence, can also be interpreted as a syntax element that is always signaled at the root of the RQT, regardless which transform block partitioning is given by the RQT structure [50].

The prediction unit (PU) syntax elements describe how the prediction is performed in order to reconstruct the samples belonging to each PU. Coding efficiency improvements have been made in HEVC for both modeling and coding of motion parameters and intra prediction modes. While H.264/AVC uses a single motion vector predictor (unless direct mode is used) and a single most probable mode (MPM), HEVC uses multiple candidate predictors or MPMs together with an index or flag for signaling the selected predictor or MPM, respectively. In addition, HEVC provides a mechanism for exploiting spatial and temporal dependencies with regard to motion modeling by merging neighboring blocks with identical motion parameters. This has been found to be particularly useful in combination with quadtree-based block partitioning, since a pure hierarchical subdivision approach may lead to partitioning with suboptimal rate-distortion behavior. Also, due to the significant increased number of angular intra prediction modes relative to H.264/AVC, three MPMs for each PU are considered in HEVC. This section will discuss how the various PU syntax elements are processed in terms of binarization, context modeling, and context assignment. Also, aspects related to parsing dependencies and throughput for the various prediction parameters are considered [50].

In HEVC, merge mode enables motion data (i.e., prediction direction, reference index and motion vectors) to be inherited from a spatial or temporal (co-located) neighbor. A list of merge candidates is generated from these neighbors. Merge _flag is signaled to indicate whether merge

is used in a given PU. If merge is used, then *merge_idx* is signaled to indicate from which candidate the motion data should be inherited. *merge_idx* is coded with truncated unary, which means that the bins are parsed until a zero bin value is reached or when the number of bins is equal to the cMax, the max allowed number of bins. Determining how to set cMax involved evaluating the throughput and coding efficiency trade-offs in a core experiment. For optimal coding efficiency, cMax should be set to equal the merge candidate list size of the PU. Furthermore, *merge_flag* should not be signaled if the list is empty. However, this makes parsing depend on list construction, which is needed to determine the list size [35] [50]. Constructing the list requires a large amount of computation since it involves reading from multiple locations (i.e., fetching the co-located neighbor and spatial neighbors) and performing several comparisons to prune the list; thus, dependency on list construction would significantly degrade parsing throughput. To decouple the list generation process from the parsing process such that they can operate in parallel in HEVC, cMax is signaled in the slice header using *five_minus_max_num_merge_cand* and does not depend on list size. To compensate for the coding loss due to the fixed cMax, combined bi-predictive and zero motion vector candidates are added when the list size is less than the maximum number of allowed candidates as defined by cMax. This also ensures that the list is never empty and that *merge_flag* is always signaled [5] [50].

If merge mode is not used, then the motion vector is predicted from its neighboring blocks and the difference between motion vector (mv) and motion vector prediction (mvp), referred to as motion vector difference (mvd), is signaled:

$$mvd(x, y) = mv(x, y) - mvp(x, y)$$

In H.264/AVC, a single predictor is calculated for mvp from the median of the left, top and top-right spatial 4 × 4 neighbors. In HEVC, advanced motion vector prediction (AMVP) is used, where several candidates for mvp are determined from spatial and temporal neighbors. A list of mvp candidates is generated from these neighbors, and the list is pruned to remove redundant candidates such that there is a maximum of two candidates. A syntax element called *mvp_l0_flag* (or *mvp_l1_flag* depending on the reference list) is used to indicate which candidate is used from the list as the mvp. To ensure that parsing is independent of list construction, mvp_l0_flag is signaled even if there is only one candidate in the list. The list is never empty as the zero motion vector is used as the default candidate [50].

In HEVC, improvements were also made on the coding process of mvd itself. In H.264/AVC, the first nine bins of mvd are regular coded truncated unary bins, followed by bypass coded 3rd order Exp-Golomb bins. In HEVC, the number of regular coded bins for mvd is significantly reduced. Only the first two bins are regular coded (abs_mvd_greater0_flag, abs_mvd_greater1_flag), followed by bypass coded first-order Exp-Golomb (EG1) bins (*abs_mvd_minus2*). In H.264/AVC, context selection for the first bin in mvd depends on whether the sum of the motion vectors of the top and left 4 × 4 neighbors are greater than 32 (or less than 3). This requires 5-bit storage per neighboring motion vector, which accounts 24, 576 of the 30,720-bit CABAC line buffer needed to support a 4k × 2k sequence. The need to reduce the line buffer size in HEVC by modifying the context selection logic was highlighted in. Accordingly, all dependencies on the neighbors were removed and the context is selected based on the binIdx (i.e.,

whether it is the first or second bin). To maximize the impact of fast bypass coding, the bypass coded bins for both the horizontal (x) and vertical (y) components of mvd are grouped together in HEVC. If four bypass bins can be processed in a single cycle, enabling bypass grouping reduces the number of cycles required to process the motion vector by one. In HEVC, reference indices ref_idx_l0 and ref_idx_l1 are coded with truncated unary regular coded bins, which is the same as for H.264/AVC: the maximum length of the truncated unary binarization, cMax, is dictated by the reference picture list size. However, in HEVC only the first two bins are regular coded, whereas all bins are regular coded in H.264/AVC. In both HEVC and H.264/AVC, the regular coded bins of the reference indices for different reference picture lists share the same set of contexts. The inter-prediction direction (list 0, list 1 or bi-directional) is signaled using *inter_pred_idc* with custom binarization [50].

Similar to motion data coding, a most probable mode (MPM) is calculated for intra mode coding. In H.264/AVC, the minimum mode of the top and left neighbors is used as MPM. *prev_intra4x4_pred_mode_*flag (or *prev_intra8x8_pred_mode_flag*) is signaled to indicate whether the most probable mode is used. If the MPM is not used, the remainder mode *rem_intra4x4_pred_mode_flag* (or *rem_intra8x8_pred_mode_flag*) is signaled. In HEVC, additional MPMs are used to improve coding efficiency. A candidate list of most probable modes with a fixed length of three is constructed based on the left and top neighbors. The additional candidate modes (DC, planar, vertical) can be added if the left and top neighbors are the same or unavailable. Note that the top neighbors outside the current CTU are considered unavailable in order to avoid the need for a line buffer.14 The prediction flag *prev_intra_pred_mode_flag* is signaled to indicate whether one of the most probable modes is used. If an MPM is used, a most probable mode index (*mpm_idx*) is signaled to indicate which candidate to use. It should be noted that in HEVC, the order in which the coefficients of the residual are parsed (e.g., diagonal, vertical or horizontal) depends on the reconstructed intra mode (i.e., the parsing of the TU data that follows depends on list construction and intra mode reconstruction). Thus, the candidate list size was limited to three for reduced computation to ensure that it would not affect entropy decoding throughput. The number of regular coded bins was reduced for intra mode coding in HEVC relative to the corresponding part in H.264/AVC, where both the flag and the 3 fixed length bins of the remainder mode are regular coded using two separate context models. In HEVC, the flag is regular coded as well, but the remainder mode is a fixed-length 5-bin value that is entirely bypass coded. The most probable mode index (*mpm_idx*) is also entirely bypass coded. The number of contexts used to code *intra_chroma_pred_mode* is reduced from 4 to 1 for HEVC relative to H.264/AVC. To maximize the impact of fast bypass coding, the bypass coded bins for luma intra prediction mode coding within a CU are grouped together in HEVC. This is beneficial when the partition mode is *PART_NxN*, and there are four sets of prediction modes [50].

There is a great number of differences between H.264/AVC and HEVC in signaling of syntax elements at the PU layer. HEVC uses both spatial and temporal neighbors as predictors, while H.264/AVC only uses spatial neighbors (unless direct mode is enabled). In terms of the impact of the throughput improvement techniques, HEVC has around 6 fewer maximum regular coded bins per inter-predicted PU than H.264/AVC. HEVC also requires around 2 fewer contexts for PU syntax elements than H.264/AVC [50].

In video coding, both intra and inter prediction are used to reduce the amount of data that needs to be transmitted. In addition, rather than sending the original samples of the prediction signal, an appropriately quantized approximation of the prediction error is transmitted. To this end,

the prediction error is block wise transformed from spatial to frequency domain, thereby decorrelating the residual samples and performing an energy compaction in the sense that, after quantization, the signal can be represented in terms of a few non-vanishing coefficients. The method of signaling the quantized values and frequency positions of these coefficients is referred to as transform coefficient coding. Syntax elements related to transform coefficient coding account for a significant portion of the bin workload. At the same time, those syntax elements also account for a significant portion of the total number of bits for a compressed video, and as a result the compression of quantized transform coefficients significantly impacts the overall coding efficiency. Thus, transform coefficient coding with CABAC must be carefully designed in order to balance coding efficiency and throughput demands. Accordingly, as part of the HEVC standardization process, a core experiment on coefficient scanning and coding was established to investigate tools related to transform coefficient coding. This section describes how transform coefficient coding evolved from H.264/AVC to the first test model of HEVC (HM1.0) to the Final Draft International Standard (FDIS) of HEVC (HM10.0), and discusses the reasons behind design choices that were made. Many of the throughput improvement techniques were applied, and new tools for improved coding efficiency were simplified [50].

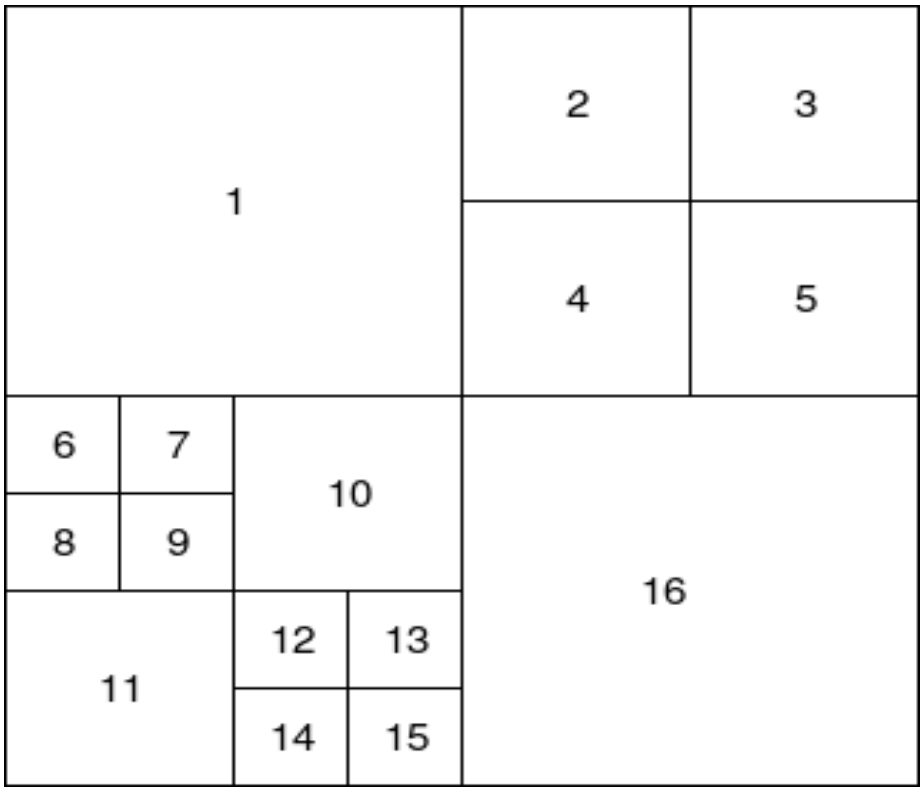

Figure 10: The quadtree structure of the partitioning of a block

The transform coding in HEVC involves a tree structured variable block-size approach with supported transform block sizes of 4 × 4, 8 × 8, 16 × 16 and 32 × 32. This means that the actual transform block sizes, used to code the prediction error of a given CU, can be selected based

on the characteristics of the residual signal by using a quadtree-based partitioning, also known as residual quadtree (RQT) and it is illustrated in figure 10. While this larger variety of transform block partitioning relative to H.264/AVC provides significant coding gains, it also has implications in terms of implementation costs, both in terms of memory bandwidth and computational complexity [27]. To address this issue, HEVC allows to restrict the RQT-based transform block partitioning by four parameters, signaled by corresponding syntax elements in the SPS: the maximum and minimum allowed transform block size (in terms of block width) nmax and nmin, respectively, and the maximum depth of the RQT dmax, with the latter given both individually for intra-picture and inter-picture prediction. Note, however, that there is a rather involved interdependency between these parameters (and other syntax elements), such that, for instance, implicit subdivisions or implicit leaf nodes of the RQT may occur. The signaling of the transform block structure for each CU is similar to that of the coding block structure at the CTU level. For each node of the RQT, a flag called *split_transform_flag* is signaled to indicate whether a given transform block should be further subdivided into four smaller TBs. Note that for the choice of a luma CTB size of 64, nmax D 32, nmin D 4, and dmax D 4, an implicit leaf node is implied for the case of TrafoSize D 4, whereas an implicit subdivision is given for a luma CB size of 64 at RQT depth equal to 0. Therefore, even if up to five different RQT levels are permitted, only up to three different context models are required for coding of *split_transform_flag*. Note that the signaling of *split_transform_flag* at the RQT root is omitted if the quantized residual of the corresponding CU contains no non-zero transform coefficient at all, i.e., if the corresponding coded block flag at the RQT root is equal to 0 [50].

For regions or blocks with many sharp edges (e.g., as typically given in screen content coding), coding gains can be achieved by skipping the transform. When the transform is skipped for a given block, the prediction error in the spatial domain is quantized and coded in the same manner as for transform coefficient coding (i.e., the quantized block samples of the spatial error are coded as if they were quantized transform coefficients). The so-called transform skip mode is only allowed for 4 × 4 TUs and only if the corresponding functionality is enabled by the *transform_skip_enabled_flag* in the PPS. Signaling of this mode is performed by using the *transform_skip_flag*, which is coded using a single fixed context model [50].

At the top level of the hierarchy of significance flags coded block flags (CBFs) are signaled for the RQT root, i.e., at the CU level in the form of the *rqt_root_cbf* and for subsequent luma and chroma TBs in the form of cbf_luma and cbf_cb, cbf_cr, respectively. rqt_root_cbf is only coded and transmitted for inter-predicted CUs that are not coded in merge mode using a single PU: for that a single context model is used. While signaling of cbf_luma is only performed at the leaf nodes of the RQT, provided that a non-zero *rqt_root_cbf* was signaled before, the chroma CBFs cbf_cb and cbf_cr are transmitted at each internal node as long as a corresponding non-zero chroma CBF at its parent node occurred. For coding of both cbf_cb and cbf_cr, four contexts are used such that the corresponding context increment depends on the RQT depth (with admissible values between 0 and 3, since for the case of TrafoSize D 4 no chroma CBFs are transmitted), whereas for cbf_luma only two contexts are provided with its discriminating context increment depending on the condition RQT depth D 0 [50].

In H.264/AVC, the significance map for each transform block is signaled by transmitting a *significant_coeff_flag* (SIG) for each position to indicate whether the coefficient is non-zero. The positions are processed in an order based on a zig-zag scan. After each non-zero SIG, an additional flag called *last_significant_coeff_flag* (LAST) is immediately sent to indicate whether it is the last

non-zero SIG; this prevents unnecessary SIG from being signaled. Different contexts are used depending on the position within the 4 × 4 and 8 × 8 transform blocks, and whether the bin represents an SIG or LAST. Since SIG and LAST are interleaved, the context selection of the current bin depends on the immediate preceding bin. The dependency of LAST on SIG results in a strong bin to bin dependency for context selection of significance map entries in H.264/AVC [50].

It is essential to explain the *sig_coeff_flag* (SIG). More specifically, while in HEVC position based context assignment for coding of *sig_coeff_flag* (SIG) is used for 4 × 4 TBs, new forms of context assignment for larger transforms were needed. In HM1.0, additional dependencies were introduced in the context selection of SIG for 16 × 16 and 32 × 32 TBs to improve coding efficiency. Specifically, the context selection of SIG was calculated based on a local template using 10 (already decoded) SIG neighbors. By using this template-based context selection bit rate savings of 1.4–2.8%. To reduce context selection dependencies and storage costs, it was proposed using fewer neighbors and showed that this could be done without severely sacrificing coding efficiency. This was further extended for HM2.0, where only a maximum of five neighbors was used by removing dependencies on positions G and K. In HM2.0, the significance map was scanned in zig-zag order, so removing the diagonal neighbors G and K is important since those neighbors pertain to the most recently decoded SIG. Despite reducing the number of SIG neighbors in HM2.0, dependency on the most recently processed SIG neighbors still existed for the positions at the edge of the transform block. The horizontal or vertical shift that is required to go from one diagonal to the next in the zig-zag scan causes the previously decoded bin to be one of the neighbors (F or I) that is needed for context selection. In order to address this issue, in HM4.0, a diagonal scan was introduced to replace the zig-zag scan. Changing from zig-zag to diagonal scan had negligible impact on coding efficiency, but removed the dependency on recently processed SIG for all positions in the TB. In HM4.0, the scan was also reversed (from high frequency to low frequency). Accordingly, the neighbor dependencies were inverted from top-left to bottom-right. Dependencies in context selection of SIG for 16 × 16 and 32 × 32 TBs were further reduced in HM7.0, where 16 × 16 and 32 × 32 TBs are divided into 4 × 4 sub blocks. This is using the flag denoted as *coded_sub_block_flag* (CSBF). In HM8.0, 8 × 8 TBs were also divided into 4 × 4 sub blocks such that all TB sizes above 4 × 4 are based on a 4 × 4 sub block processing for a harmonized design. The 8 × 8, 16 × 16 and 32 × 32 TBs are divided into three regions based on frequency. The DC, low-frequency and mid/high-frequency regions all use different sets of contexts. To reduce memory size, the contexts for coding the SIG of 16 × 16 and 32 × 32 TBs are shared. For improved coding efficiency for intra predicted CUs, so-called mode dependent coefficient scanning (MDCS) is used to select between vertical, horizontal, and diagonal scans based on the chosen intra prediction mode. The scans are assigned based on intra prediction mode, TB size, and component. this requires the intra mode to be decoded before decoding the corresponding transform coefficients. MDCS is only used for 4 × 4 and 8 × 8 TBs and provides coding gains of up to 1.2%. Note that for TBs larger than 8 × 8 and for TBs of inter predicted CUs, only the diagonal scan is used [50].

As mentioned earlier, there are strong data dependencies between *significant_coeff_flag* (SIG) and *last_significant_coeff_flag* (LAST) in H.264/AVC due to the fact that they are interleaved. If all of the N SIG are zero, LAST is not transmitted. We avoid to interleave of SIG and LAST altogether. Specifically, the horizontal (x) and vertical (y) position of the last non-zero SIG in a TB is sent in advance rather than LAST by using the syntax elements *last_sig_coeff_x* and *last_sig_coeff_y*, respectively. For instance, *last_sig_coeff_x* equal to 3 and *last_sig_coeff_y* equal to 0 are sent before processing the TB rather than signaling LAST for each SIG with value of 1.

Signaling the (x, y) position of the last non-zero SIG for each TB was adopted into HM3.0. Note that the SIG in the last scan position is inferred to be 1. The last position, given by its coordinates in both x and y direction, is composed of a prefix and suffix. The prefixes *last_sig_coeff_x_prefix* and *last_sig_coeff_y_prefix* are both regular coded. A suffix is present when the corresponding prefix is composed of more than four bins. In that case, the suffixes *last_sig_coeff_x_suffix* and *last_sig_coeff_y_suffix* are bypass coded using FL binarization. Some of the contexts are shared across the chroma TB sizes to reduce context memory. To maximize the impact of fast bypass coding, the bypass coded bins i.e., the suffix bins, for both the x and y coordinate of the last position are grouped together for each TB in HEVC [50].

A special flag used by HEVC is the *coded_sub_block_flag* (CSBF). the number of bins to be transmitted for signaling the significance map is considerably reduced by using a hierarchical signaling scheme of significance flags. By using the *coded_sub_block_flag* (CSBF) we declare that for each 4 × 4 sub block of a TB whether there are non-zero coefficients in the sub block. If CSBF is equal to 1, the sub block contains at least one non-zero transform coefficient level and, consequently, SIGs within the sub block are signaled. No SIGs are signaled for a 4 × 4 sub block that contains all vanishing transform coefficients, since this information is signaled by a CSBF equal to 0. For large TB sizes, a reduction in SIG bins of up to a 30% can be achieved by the use of CSBFs, which corresponds to an overall bin reduction of 3% – 4% under common test conditions. To avoid signaling of redundant information, the CSBF for the sub blocks containing the DC and the last position are inferred to be equal to 1. In HM7.0, the CSBF was additionally used to further reduce dependencies in the context selection of SIG for 16 × 16 and 32 × 32 TBs. Specifically, the neighboring sub blocks and their corresponding CSBFs are used for context selection rather than the individual SIG neighbors. This context selection scheme was extended to 8 × 8 TBs in HM8.0. According to this scheme, the CSBF of the neighboring right and bottom subblocks (CSBFright ,CSBFbot tom) are used to select one of four patterns: (0,0) maps to pattern 1, (1,0) to pattern 2, (0,1) to pattern 3 and (1,1) to pattern 4. The pattern maps each position within the 4 × 4 sub block to one of three contexts. As a result, there are no intrinsic dependencies for context selection of SIG within each 4 × 4 sub block. Reverse diagonal scanning order is used within the sub blocks and for the processing order of the sub blocks themselves. Both significance map and coefficient levels are processed in this order. As an exception to this rule, for 4 × 4 and 8 × 8 TBs to which MDCS is applied, reverse vertical and horizontal scanning orders are used within the sub blocks as well as for the processing order of the sub blocks themselves. Furthermore, different sets of contexts for coding of SIG are used for diagonal and non-diagonal (vertical and horizontal) scans in both 4 × 4 luma and chroma TBs, and 8 × 8 luma TBs [50].

The decoder has the same stages as the encoder but with different order. The decoder starts with de-quantization and follows with the inverse transform, prediction mode and in the end reconstruction of the current block.

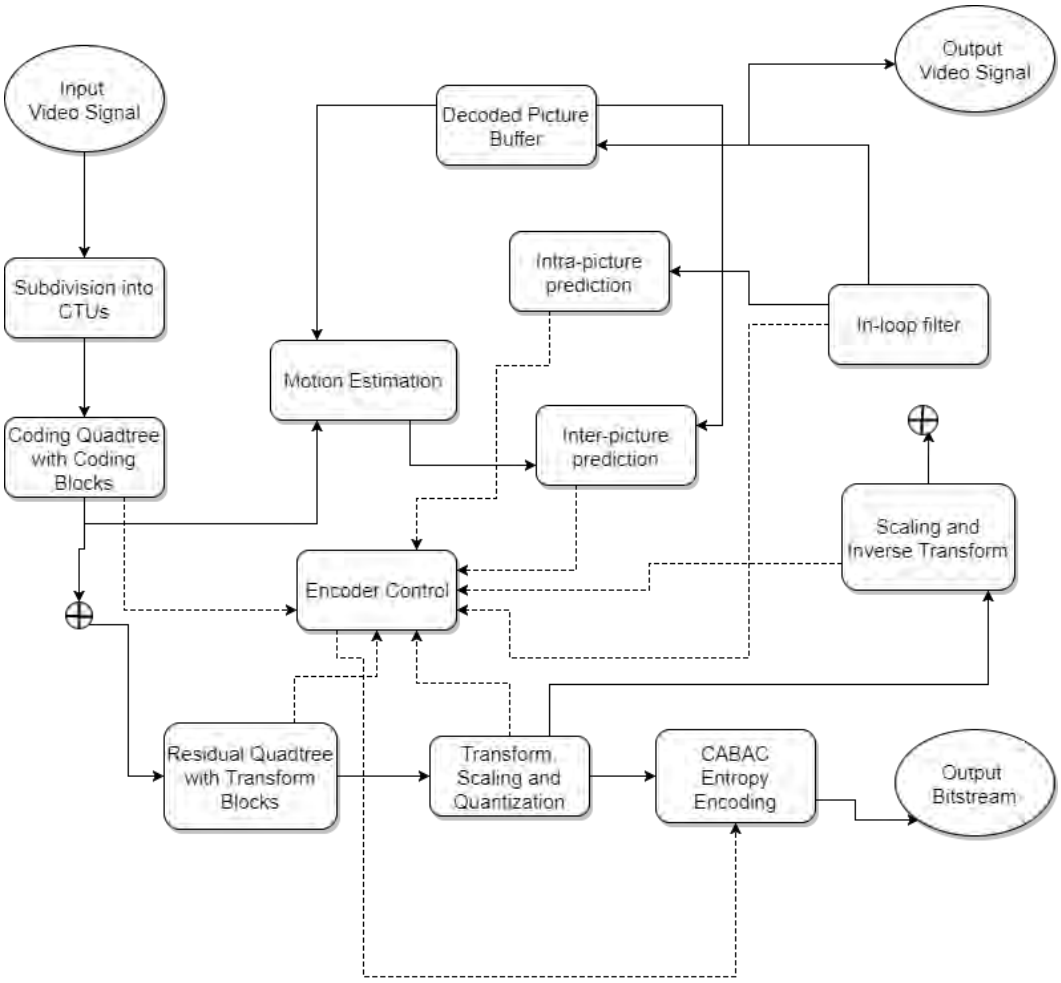

Figure 11: Overview over encoder

Hardware architectures support multi-threading. Multi-threading is defined as a programming model which consists of different independent units. These units are called threads and share the same resources, especially memory. Thread are able to run in parallel or in in a time division manner. They communicate via a shared memory and use synchronization operations to ensure that the resources are used properly. The progress is controlled with synchronization statements like locks, semaphores and barriers [50].

It is inferred that the use of resources is of utmost importance. In order to achieve this, there have been developed a great number of parallelization techniques. The most significant of the are the following [50]:

- Picture-level parallelization: when multiple pictures are processed at the same time, this procedure is called picture-level parallelization. The restriction is that the temporal dependencies and the motion compensated prediction must not be violated. This level of parallelization is suitable for multi-core systems with a few cores. One disadvantage is that the

workload may be imbalanced in each core. Also, the parallelization scalability is determined by the length of the motion vector and the size if the group of pictures (GOP). In the end, this technique does not improve the latency and increases the processing frame rate.

- Slice-level parallelization: each picture can be divided into slices and each slice can be processed independently. This makes easier to process the slices in a parallel way. On the other hand, there are stages in the video coding that requires the process of the slice boundaries. These stages are the prediction, transform, entropy coding and in-loop filter. What is more, the efficiency is decreased as there are restrictions concerning the process of slice boundaries in the intra-picture prediction and entropy coding stage. In short, this level of parallelization should be used when the number of slices in each picture is limited.

- Block-level parallelization: There are two different approaches concerning this technique. The first one is the block-level pipeline. We use heterogenous processing cores for each stage. One core is used for prediction, one core for in-loop filtering, one core for entropy coding and so on. The blocks are processed in different core concurrently. This means that the algorithm scheduling the process of the blocks should be efficient. The second approach refers to the wavefront scheduling. The blocks are grouped in such a way that as much as more blocks are available. The restriction concerns the spatial dependencies among neighboring blocks. This manner is similar with the wavefront. The blocks belonging in the same wavefront can be processed concurrently. Also, blocks belonging in different pictures can be processed in parallel taking into account that there are no temporal dependencies. One disadvantage is that we use the memory very frequently. Moreover, the decoding time of the picture is not being reduced for the entropy coding stage. Last but not least, an entropy encoding step using one thread could be the bottleneck of the whole throughput. In conclusion, this approach is suitable for multi-core architectures.

Parallelism is a new feature which exists in HEVC and it is considered to be of great importance. Because of this new feature, we are able to enjoy privileges which were thought to be impossible in the past. One of these privileges is the ability to have real time encoding and decoding of a video on systems unable to do this in previous years. However, it is undeniable that parallelization leads to problems and it is essential to deal with them. In order to overcome problems of parallelization, HEVC has same tools. These tools are: Wavefront Parallel Processing (WPP) and Tiles. Both of them offer the possibility to partition the pictures multiple times and use these partitions for later process. Tiles will be explained in detail in the next section [50].

Wavefront parallel processing forces each CTU row to constitute a separate partition. Thus, each row is processed relative to the previous row and two consecutive CTUs are used as delay. The number of threads is up to the number of CTU rows. In this way, it is possible to process parallel the individual CTU rows. In the decoder, a thread is processing a single CTU row. It is important to schedule CTU decoding in such a way that the dependencies among the CTUs are

reduced as much as possible. The scheduling concerns that the top right neighboring CTU in the preceding row must have finished [3] [50].

There is a limitation in the scalability of the wavefront parallel processing. The main reason is that the number of the dependent CTUs is increased at the beginning and at the end of each picture, as it is illustrated in figure 12. In order to tackle this problem, a technique called overlapped wave-front (OWF) has been proposed. This technique offers the possibility to decode multiple pictures simultaneously. In this way, we have a constant parallelization gain [9] [50].

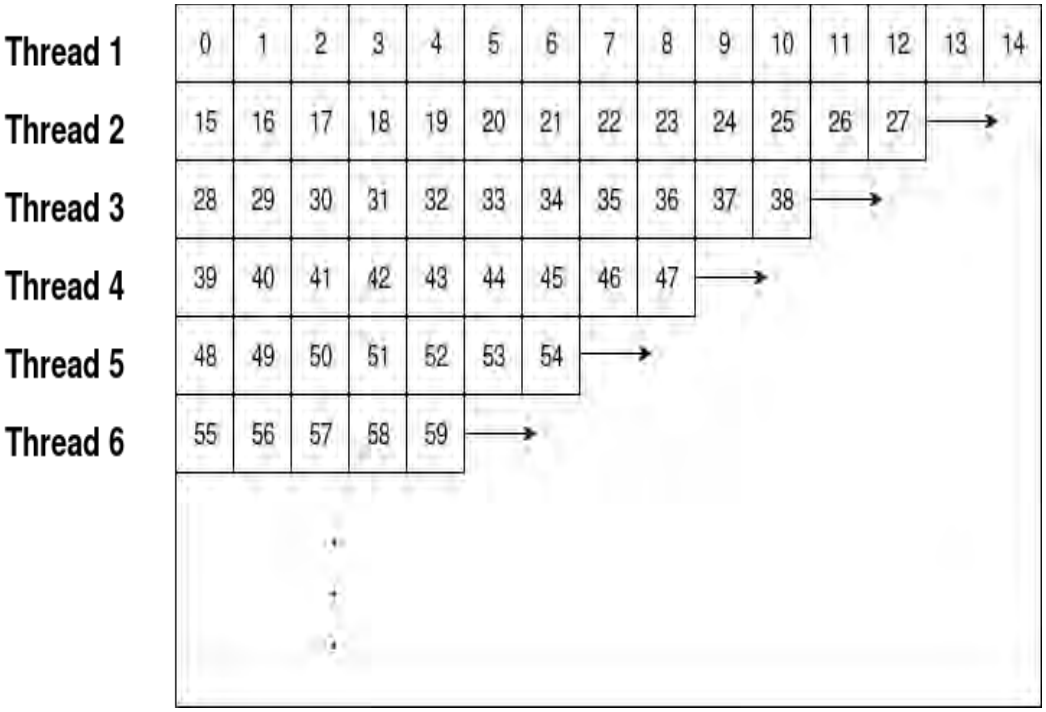

Figure 12: The process of WPP using multiple threads

Using WPP, parallelization of the encoder and the decoder is improved to a great extent. We are allowed to have many picture partitions with compression efficiency. Another major advantage of WPP is the fact that we are not obliged to pass to the in-loop filter on more time. Moreover, WPP is convenient for low-delay applications using sub-picture delay. The amount of parallelism is fixed [50].

Tile overview of HEVC

Tiles are considered to be of utmost importance in order to overcome the difficulties of parallelism. Tiles are rectangular groups of CTUs separated by horizontal and vertical boundaries. In other words, the main idea behind tiles is based on a division of the frame and having as limitation only the rectangular shape, as shown in figure 13. It is essential to be mentioned that there may exist dependencies between the different partitions. The group of CTUs contained in a partition is not necessarily independent from the group of CTUs contained in another partition. This leads to the fact that the multiple partitions of the frame can be processed in a parallel way. As we have already mentioned, tiles are a HEVC tool and as a consequence we are able to enable or disable them depending on our target each time. In order to enable tiles, it is needed to use the following syntax element: "*tiles_enabled_flag*" which indicates that we use tiles. For the rest of this thesis we will assume that tiles are enabled [50].

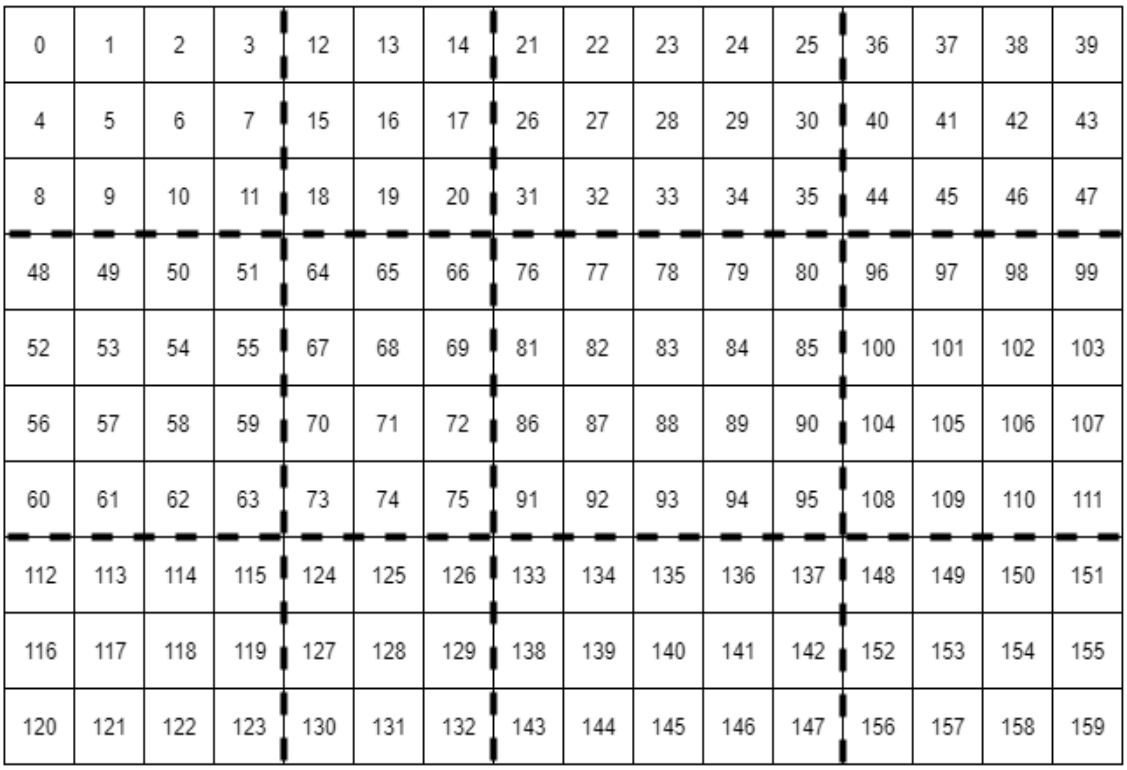

Figure 13: The partitioning of a frame into 12 tiles

At each sequence, the number of tiles as well as their boundaries can be constant through the whole sequence or change from picture to picture. The only restriction is to take into account the fact that tiles' shape must be rectangular. This characteristic of tiles is of great importance as the encoder can arrange tiles in the pictures in such a way that the load balancing is controlled.

This means that if a region of a picture requires large amounts of resources, then it is divided in more tiles. In contrast, if a region of the picture requires a few resources, there is no need to partition it in many tiles. In this procedure, there is the limitation that the management of the resources and as a result the structure of the tiles must be determined before the actual encoding process. It is worthy to mention that each tile can be processed independently from the others as the dependencies of parsing and prediction does not exist. This characteristic is quite beneficial for lossy transmission environments as each tile can be transported in different packets [49] [50].

Even though a tile-based raster scan order is used in tiles, decoders may process the CTUs of tiles in a picture-based raster scan order without taking advantage of the parallelism [15]. Furthermore, the communication between the processors is not required for entropy decoding and reconstruction, however, communication is needed if in-loop filtering it is chosen a specific mode called cross-tile filtering mode [6] [50].

Tiles are considered to be applied very efficient by the encoder and the decoder. The amount of parallelism is not fixed and this allows the encoder to adjust the number of tiles. As a consequence, the number of tiles depends on the computational resources available. Tiles spans the picture horizontally and vertically using multiple rectangles [25][28]. In other words, the picture is partitioned into multiple rectangles which are not allowed to overlap each other. If we have a region of interest (ROI) coding, it is advisable to use tiles instead of slices [24] [50]. For instance, the combination of tiles and an effective algorithm can help to adjust the size and error protection of the ROIs [31] [50].

We can use tiles for applications which demand the distribution of the coding tasks into different hardware machines and systems. Each tile can be processed independently from the other tiles in the same picture. This independency is the main characteristic which allows the parallel process of tiles. Of course, the cross tile-border filtering should be done separately [10]. This leads to the fact that the communication overhead between the threads which process individual tiles [50].

It is of utmost importance to mention that we are not permitted to use both WPP and tiles as parallelization tools in the same video sequence compression. But it is possible in the future the standard offers this possibility. This means that a picture firstly could be divided using tiles and we could divide further each tile using WPP [13] [50].

As far as tiles are concerned, it is generally believed that they achieve better compression efficiency in comparison with slices. This happens because between tiles the spatial distances are reduced and as a consequence the spatial correlation is exploit to a great extent.

CHAPTER 3: THE PROBLEM OF ADAPTING TILE SIZING

Definition of the problem

Let A be an $M \times N$ matrix whereby a cell A_{ij} has a weight of $w(A_{ij}) > 0$. Let r_i denote the i-th cell row of the array and $w(r_i)$ the aggregated weight of this row, i.e., $w(r_i) = \sum_{j=1}^{N} w(A_{ij})$. Similarly, c_j denotes the j-th column in A and $w(c_j)$ its corresponding weight.

Matrix A can be partitioned using horizontal and vertical dividers. We consider the case where a horizontal divider spans all the columns of the matrix. For instance, if a horizontal divider is placed on row i then the matrix is partitioned into two areas, the first containing cells A_{xj} for which $x < i$ and the second cells A_{xj} for which $x > i$. For vertical dividers, we don't follow a similar assumption, thus the vertical dividers considered in the thesis partially split matrix A. For instance, a vertical divider for column j that spans from row i until row k, partially partitions matrix elements into the following sets:

a) cells A_{xy} with I <x ≤ k and y ≤ j,

b) cells A_{xy} with I < x ≤ k and y > j.

Obviously, all other cells A_{xy} for which x ≤ i or x > k, don't belong to any of the two aforementioned partitions.

For a given set of horizontal dividers (let H), we number the H + 1 horizontal zones the array is split into, in a top down fashion, i.e., r_1 belongs to horizontal zone 1 (hence referred to as h_1) , while r_M belongs to the horizontal zone h_{H+1} . Horizontal splits, together with vertical ones, partition the matrix into rectangles, hence called tiles. Let T_{uv} denote the v-th tile existing in h_u , assuming a numbering of them from left to right.

Let bN(T_{uv}) and bS(T_{uv}) denote the first, respectively the last cell row of tile Tuv (north and south rectangle boundaries). Similarly, let bW(T_{uv}) and bE(T_{uv}) denote the left and right column boundaries of Tuv. The boundaries of a horizontal zone can be defined in a similar manner, i.e., with N, S, W, E boundaries, only that in the case of the horizontal zones considered in the paper: bW(h_u) = 1, and bE(h_u) = N, 1 ≤ u ≤ H+1.

Finally, let w(hu) denote the aggregate weight of cells belonging to horizontal zone hu and w(T_{uv}) the aggregate weight of cells belonging to tile T_{uv}, or more formally:

$$w(h_u) = \sum_{i=bN(h_u)}^{bS(h_u)} \sum_{j=1}^{M} w(A_{ij}) \qquad (1)$$

$$w(T_{uv}) = \sum_{i=bN(T_{uv})}^{bS(T_{uv})} \sum_{j=bW(T_{uv})}^{bE(T_{uv})} w(A_{ij}) \quad (2)$$

The problem statement is as follows:

Given A, and two integers: H, T, find a splitting of A into tiles such as:

a) *A is split into H + 1 horizontal zones*

b) *the total number of tiles formed is T, and*

c) *max{w(T_{uv})} is minimized.*

Algorithm

The idea behind the algorithm is based on finding the best placement of the horizontal divider given the number of tiles we are able to use. Specifically, we place the divider so as the sum of the matrix's elements is less than k times the ideal average tile cost. Formally it can be described in the following equation:

$$\sum h_i \; < k * (\; ideal\; average) \quad , \quad 0 \leq i \; \leq H + 1 \quad (3)$$

The ideal average tile cost is calculated as follow:

$$ideal\; average\; tile\; cost = \frac{C}{T} \quad (4)$$

With C, we denote the total cost of the two-dimensional matrix and it is expressed in form of equation as follows:

$$C = \sum_{i=1}^{M} \sum_{j=1}^{N} A_{ij} \quad (5)$$

The proposed algorithm takes as an input the number of horizontal dividers H and the number of tiles T and produces as output the final placement of the tiles. For each step, we investigate the placement of the horizontal divider in different position each time. This means that for the placement of the first horizontal divider h_1 , we examine the placement at the rows r_i for $i =$

$0, \ldots, H - T$ and for each position are given k tiles. The number of tiles given each time is computed by the (3) formula.

Table 1: Algorithm

	Algorithm
	Input: Number of horizontal zones and number of tiles
1.	Compute the ideal average tile cost
2.	For each horizontal zone
3.	Check the maximum tile cost
4.	Select the minimum
5.	Place the horizontal divider
6.	End for

Example

It can be easily understood by using the example which is illustrated in figures 14 - 18. Let us have a matrix of size 4 × 6 and we would like to have one horizontal divider and six tiles. This means that we have two horizontal zones and each of which can have from one two six tiles depending on the number of available tiles. Firstly, we compute the ideal average tile cost given by the formula (4). The result is 70 which means that with ideal conditions the cost of each tile should be about 70. Then, we check to place the horizontal divider between the first and the second row and using one tile for the first horizontal zone. This means that the first row consists the first tile with cost 72, as illustrated in figure 14.

10	6	14	26	10	6
42	8	23	11	30	25
18	20	16	12	21	15
25	18	30	17	15	4

Figure 14: The first check in order to place the first horizontal divider

After this step, we check to place the horizontal divider between the second and the third row, using two tiles for the first horizontal zone and each tile has cost 103 and 108 respectively. From these two values, we choose to keep the maximum as we want to minimize the maximum tile cost. Thus, for this placement the maximum tile cost is 108. This step is depicted in figure 15.

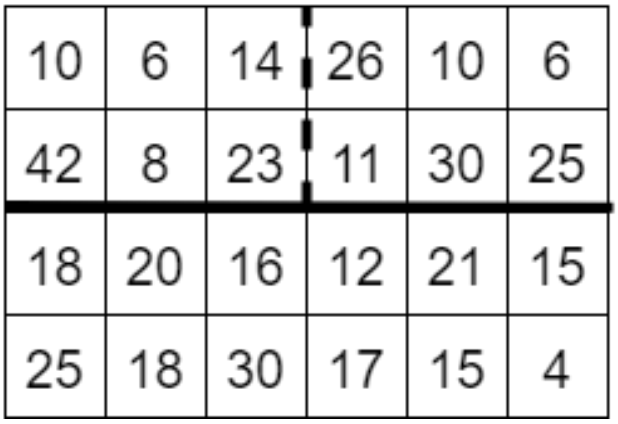

Figure 15: The second check in order to place the first horizontal divider

Next, we check to place the horizontal divider between the third and the forth row, using three tiles for the first horizontal zone and each tile has cost 104, 102 and 107 respectively. From these three values, we keep the maximum one. Thus, for this placement the maximum tile cost is 107. This step is illustrated in figure 16.

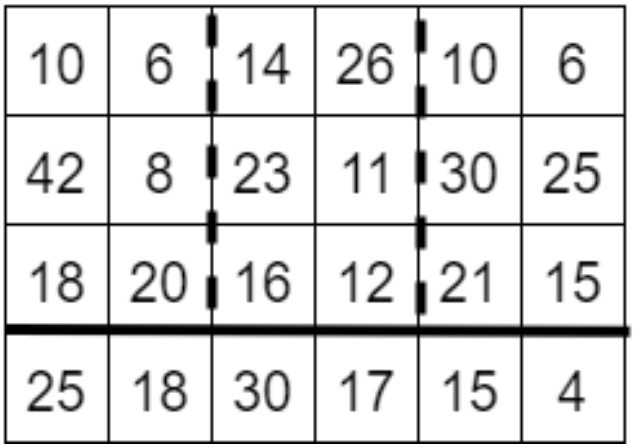

Figure 16: The third check in order to place the third horizontal divider

Then, we select the position so as to place the horizontal divider taking into account the maximum tile cost. From the possible positions with tile cost 72, 108 and 107 respectively, we

select the minimum one (72). Thus, we place the horizontal divider between the first and the second row. This step is shown in figure 17.

10	6	14	26	10	6
42	8	23	11	30	25
18	20	16	12	21	15
25	18	30	17	15	4

Figure 17: The placement of the horizontal divider

In the last stage, we place the remaining five tiles in the second horizontal zone. This step is illustrated in figure 18.

10	6	14	26	10	6
42	8	23	11	30	25
18	20	16	12	21	15
25	18	30	17	15	4

Figure 18: The final partitioning of the matrix

Then the 1D optimal algorithm [28], which will be described later in detail, is applied in order to place the tiles for each partition and find the optimal placement. After this, we select the partition with the minimum cost tile cost as it is expressed in (2). The same procedure is repeated for the other horizontal dividers until all dividers are placed properly. It is important to mention that each horizontal zone h_i ,$1 \leq i \leq H+1$ may have not only different number of row but also different number of tiles.

The 1D optimal algorithm has been described briefly in the paper: "Heuristics for Tile Parallelism in HEVC" [28]. In short, let us assume that we have a one-dimensional array with

positive elements of size S and should be divided into B bins such that the maximum cost of the bins is minimized. When we refer to cost we consider the sum of the elements of the array existing in a bin.

Suppose that we have the bin size B, we are able to investigate if all the elements of the array can fit in M bins in a consecutive manner. This means that if we have run out of bins and we still have elements of the array which have not been assigned to a bin, then the assignment is considered to be unfeasible.

Let us tart with a minimum bin size denoted as B_{min} . The value of B_{min} can be calculated using the total cost of the array C and the number of bins:

$$B_{min} = \frac{C}{B}$$

Then, we check if the assignment with B_{min} is feasible. If it is unfeasible we multiply by two the B_{min} and we move on the next step otherwise we stop the process in this step. In the second step, we check if $2 * B_{min}$ is a feasible assignment and repeat the previous process. In the last step, we end up to $2^k * B_{min}, k \geq 0$ which is considered to be the first possible assignment.

In the interval [B_{min} , C] we do binary search to find the optimal bin size. Obviously, there does not exist assignment with bin size less than B_{min} . Also, the bin size is upper bounded by the total cost of the matrix denoted by C. Each time we check the middle of the interval and if the assignment is feasible we move into the left side of the interval and repeat the same process. If the assignment is unfeasible, we move into the right side of the interval and repeat the process. The process stops when we have found the optimal bin size.

CHAPTER 4: EXPERIMENTAL EVALUATION

Experimental Setup

The matrices used in the simulation were obtained by experiments conducted on a Linux server with two 6-core Intel Xeon E5-2630 CPUs running at 2.3 GHz using hyper threading. The HM 16.7 reference software and OpenMP were used for these experiments. The initial parameters for the experiments were the following: QP was set to 32, bit depth was 8, CTU size 64×64, max depth for partitioning was set to 4 and search mode to TZ. In addition to this, the results were obtained assuming the LD scenario with an initial I frame followed by P frames and a GOP size of 4.

We conducted the simulation using the following two sequences: Kimono and traffic. Their properties are mentioned in Table 2.

Table 2: Video sequences

Name	Resolution	Frames per second (fps)	Total frames	CTUs per second
Kimono	1920 × 1080	24	240	510
Traffic	2560 × 1600	30	150	1000

Results and Discussion

We measured the performance of the algorithm using two main aspects. The first one is the total execution time of the uniform and the heuristic algorithm as well as its split per frame. The second one is the time per partition using the following percentage:

$$\frac{(uniform - heuristic)}{heuristic} * 100\%$$

The experiments were conducted using the number of horizontal zones and the number of vertical zones for the uniform algorithm as well as the number of horizontal zones and the number of tiles as they are summarized in Table 3. Figures 19 and 20 illustrate the execution time of the

uniform algorithm and the heuristic algorithm. As it is shown, the uniform algorithm has less execution time than the heuristic.

Table 3: Number of horizontal zones and tiles for uniform and heuristic

	UNIFORM	HEURISTIC
	Horizontal zones × vertical zones	**Horizontal zones × tiles**
1	2 × 2	2 × 4
2	2 × 3	2 × 6
3	2 × 4	2 × 8
4	2 × 5	2 × 10
5	3 × 2	3 × 6
6	3 × 3	3 × 9
7	3 × 4	3 × 12
8	3× 5	3 × 15
9	4 × 2	4 × 8
10	4 × 3	4 × 12
11	4 × 4	4× 16
12	4 × 5	4 × 20

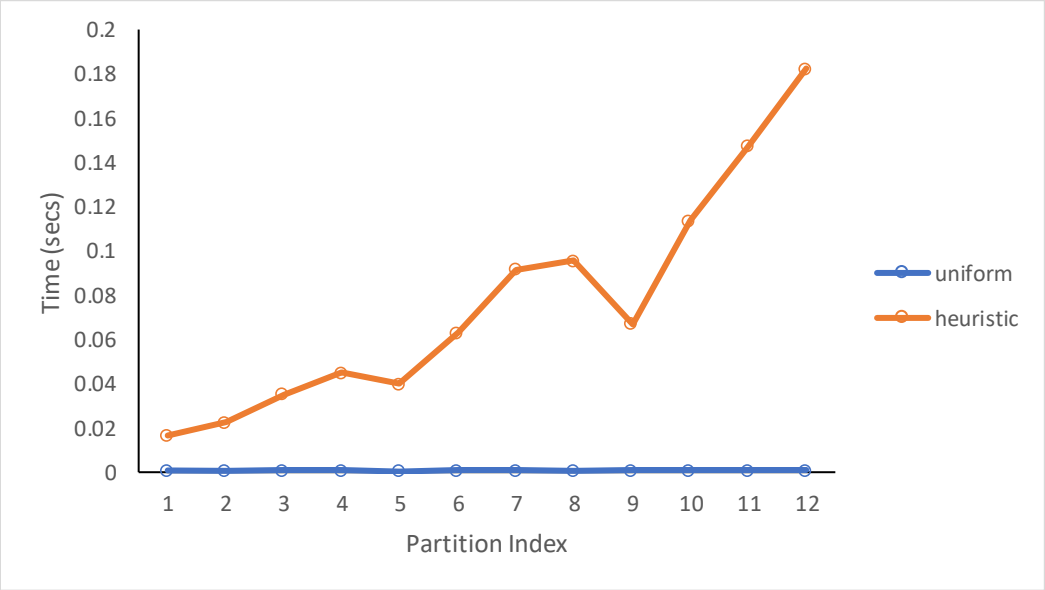

Figure 19: Traffic, total execution time

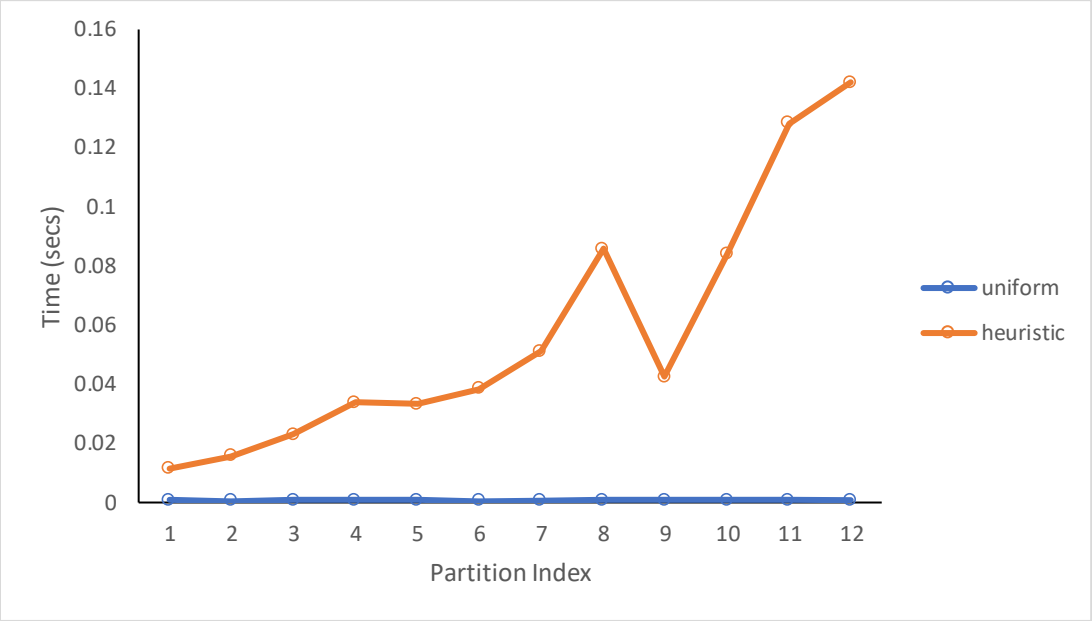

Figure 20: Traffic, total execution time

The percentage of improvement between the uniform algorithm and the heuristic is illustrated in figures 21 and 22. In Kimono, the heuristic algorithm outperforms the uniform algorithm with some exceptions. The results are even better when the number of tiles is increasing. In Traffic, the results are thought to be very promising at first, but then they deteriorate. However, it is important to mention that they tend to improve again after a number of tiles.

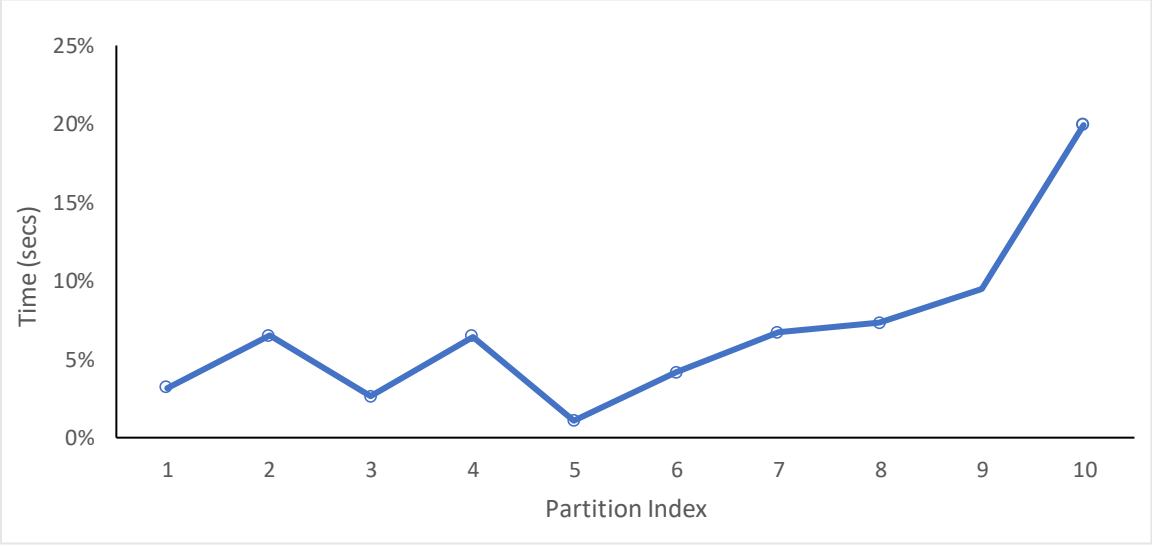

*Figure 21: Kimono, (uniform - heuristic)/ heuristic * 100%*

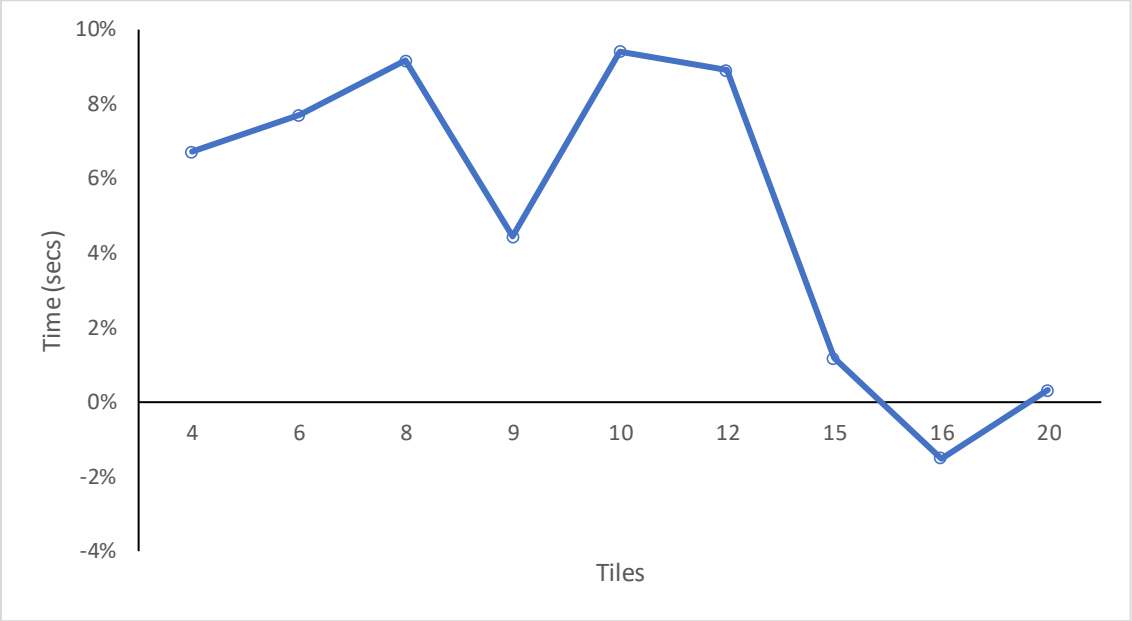

*Figure 22: Traffic, (uniform - heuristic)/ heuristic * 100%*

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

In this thesis, we tackled the problem of partitioning a matrix using tiles taking into consideration the minimization of maximum tile cost. The proposed algorithm can place both the horizontal and the vertical dividers in a matrix so as to reduce the maximum tile cost. The main idea behind the algorithm is to find the best position for the horizontal divider depending on the number of tiles which are available. We evaluated the performance of our heuristic using simulation and we led to the conclusion that our heuristic not only achieves better results than a uniform cut but also can be further developed in order to minimize the maximum tile cost even more. The proposed heuristic has the potential to achieve even better performance and this possibility will be investigated in the future.

REFERENCES

[1] A. Mingozzi, S. Ricciardelli, M. Spadoni, "Partitioning a matrix to minimize the maximum cost," Discrete Appl. Math., vol. 62, no. 1-3, pp. 221–248, 1995.

[2] Ahmed N, Natarajan T, Rao KR (1974) Discrete cosine transform. IEEE Trans Comput C-23:90–93.

[3] Alvarez-Mesa M, George V, Chi CC, Schierl T (2012) Improving parallelization efficiency of WPP using overlapped wavefront, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J0425, Stockholm, July 2012.

[4] Boyce J (2004) Weighted prediction in the H.264/MPEG AVC video coding standard. In: Proceedings of the 2004 international symposium on circuits and systems, ISCAS '04, vol 3, pp III–789–92, 2004.

[5] Bross B, Jung J, Huang YW, Tan YH, Kim IK, Sugio T, Zhou M, Tan TK, Francois E, Kazui K, Chien WJ, Sekiguchi S, Park S, Wan W (2011) BoG report of CE9: MV Coding and Skip/Merge operations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E481, Geneva, Mar. 2011.

[6] C. Blumenberg, D. Palomino, S. Bampi, and B. Zatt, "Adaptive content-based tile partitioning algorithm for the HEVC standard," PCS 2013, pp. 185-188.

[7] Chen P, Ye Y, Karczewicz M (2008) Video coding using extended block sizes. ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG- AJ23, San Diego, Oct. 2008.

[8] Chou PA, Lookabaugh T, Gray RM (1989) Optimal pruning with applications to tree-structured source coding and modeling. IEEE Trans Inf Theory 35:299–315.

[9] Clare G, Henry F (2012) An HEVC transcoder converting non-parallel bitstreams to/from-WPP, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC- J0032, Stockholm, July 2012.

[10] Fuldseth A, Horowitz M, Xu S, Zhou M (2011) Tiles, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E408, Geneva, Mar. 2011.

[11] G. J. Sullivan, J.-R. Ohm, W.-J. 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.

[12] Han WJ, Min J, Kim IK, Alshina E, Alshin A, Lee T, Chen J, Seregin V, Lee S, Hong YM, Cheon MS, Shlyakhov N, McCann K, Davies T, Park JH (2010) Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools. IEEE Trans Circuits Syst Video Technol 20(12):1709–1720.

[13] Henry F, Pateux S (2011) Wavefront parallel processing, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196, Geneva, Mar. 2011

[14] Iain Richardson, "H.264 and MPEG-4 VIDEO COMPRESSION", Wiley 2003.

[15] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, pp. 969-977, Dec. 2013.

[16] Kanumuri S, Tan TK, Bossen F (2011) Enhancements to intra coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D235, Daegu, Jan. 2011.

[17] Kim I-K, Min JM, Lee T, Han W-J, Park JH (2012) Block partitioning structure in the HEVC standard. IEEE Trans Circuits Syst Video Technol 22:1697–1706.

[18] Lainema J, Bossen F, Han W-J, Min J, Ugur K (2012) Intra coding of the HEVC standard. IEEE Trans Circuits Syst Video Technol 22(12):1792–1801.

[19] Laroche G, Jung J, Pesquet-Popescu B (2008) RD optimized coding for motion vector predictor selection. IEEE Trans Circuits Syst Video Technol 18(9):1247–1257.

[20] Li B, Xu J, Li H (2011) Non-CE9/Non-CE13: Simplification of adding new merge candidates, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G397, Geneva, Nov. 2011.

[21] M.G. Koziri and A. Eleftheriadis, "Joint Quantizer Optimization for Scalable Coding," ICIP 2010, pp. 1281-1284.

[22] M.G. Koziri, A.N. Dadaliaris, G.I. Stamoulis, and I. Katsavounidis, "A Novel Low-Power Motion Estimation Design for H.264," ASAP 2007, pp. 247-252.

[23] M.G. Koziri, G.I. Stamoulis, and I. Katsavounidis, "Power Reduction in an H.264 Encoder Through Algorithmic and Logic Transformations," ISLPED 2006, pp. 107-112.

[24] M.G. Koziri, P. Papadopoulos, N. Tziritas, A.N. Dadaliaris, T. Loukopoulos, and S.U. Khan, "Slice-Based Parallelization in HEVC Encoding: Realizing the Potential Through Efficient Load Balancing," Proc. 18th Int. Workshop on Multimedia Signal Processing (MMSP 2016), IEEE, Montreal, Canada, Sept. 2016.

[25] M.G. Koziri, P. Papadopoulos, N. Tziritas, A.N. Dadaliaris, T. Loukopoulos, S.U. Khan, and C.-Z. Xu, "Adaptive Tile Parallelization for Fast Video Encoding in HEVC", Proc. 12th IEEE Int. Conf. on Green Computing and Communications (GreenCom 2016), IEEE, Chengdu, China, Dec. 2016.

[26] M.G. Koziri, D. Zacharis, I. Katsavounidis, and N. Bellas, "Implementation of the AVS video decoder on a heterogeneous dual-core SIMD processor," IEEE Trans. Consumer Electronics, vol 57(2), pp. 673-681, 2011.

[27] Ma S, Kuo C-CJ (2007) High-definition video coding with super-macroblocks. In: Proceedings of visual communications and image processing, vol. 6508.

[28] Maria Koziri, Panos K. Papadopoulos, Nikos Tziritas, Nikos Giachoudis, Thanasis Loukopoulos, Samee U. Khan, Georgios I. Stamoulis, "Heuristics for Tile Parallelism in HEVC".

[29] Marpe D, Schwarz H, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H, Nguyen T, Oudin S, Siekmann M, Sühring K, Winken M, Wiegand T (2010) Video compression

using quadtrees, leaf merging and novel techniques for motion representation and entropy coding. IEEE Trans Circuits Syst Video Technol 20:1676–1687.

[30] Minezawa A, Sugimoto K, Sekiguchi S (2011) An improved intra vertical and horizontal prediction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F172, Torino, July 2011.

[31] Misra K, Segall A, Horowitz M, Xu S, Fuldseth A, Zhou M (2013) An overview of tiles in HEVC. IEEE J Sel Topics Signal Process 7:969–977.

[32] N. Assimakis, M. Adam, M. Koziri, S. Voliotis, and K. Asimakis, "Optimal Decentralized Kalman Filter and Lainiotis Filter," Digital Signal Processing, vol. 23(1), pp. 442-452, 2013.

[33] N. Tziritas, T. Loukopoulos, S.U. Khan, and C.-Z. Xu, "Distributed Algorithms for the Operator Placement Problem," IEEE Trans. on Computational Social Systems, vol.2(4), pp. 182-196, 2015.

[34] Ohm J-R, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards – including High Efficiency Video Coding (HEVC). IEEE Trans Circuits Syst Video Technol 22:1669–1684.

[35] Oudin S, Helle P, Stegemann J, Bartnik C, Bross B, Marpe D, Schwarz H, Wiegand T (2011) Block merging for quadtree-based video coding. In: IEEE international conference on multimedia and expo, pp 1–6, IEEE, 2011.

[36] Piao Y, Min J, Chen J (2010) Encoder improvement of unified intra prediction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-C207, Guangzhou, Oct. 2010.

[37] Pu W, Chen J, Karczewicz M, Kim WS, Sole J, Guo L (2013) High precision weighted prediction for HEVC range extension, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-O0235, Geneva, Oct.-Nov. 2013.

[38] RoitzschM(2007) Slice-balancing H.264 video encoding for improved scalability of multicore decoding. In: Proceedings of the 7th ACM IEEE International conference on Embedded Software, pp 269–278.

[39] Schierl T, George V, Henkel A, Marpe D (2012) Dependent slices, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0229, Geneva, Apr.-May 2012.

[40] Schwarz H, Wiegand T (2001) Tree-structured macroblock partition, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-O17, Pattaya, Dec. 2001.

[41] Sekiguchi S, Yamagishi S (2009) On coding efficiency with extended block sizes for UHDTV. MPEG document M16019.

[42] Shen L, Zhang Z, An P (2013) Fast CU size decision and mode decision algorithm for HEVC intra coding. IEEE Trans Consum Electron 59(1):207–213.

[43] Silva TL, Agostini LV, Silva Cruz LA (2012) Fast HEVC intra prediction mode decision based on EDGE direction information. In: Proceedings of 20th European signal processing conference (EUSIPCO), Aug. 2012, pp 1214–1218.

[44] Sullivan GJ, Wiegand T (1998) Rate-distortion optimization for video compression. IEEE Signal Process Mag 15:74–90.

[45] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, 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, Jul. 2003.

[46] Tourapis AM, Wu F, Li S (2005) Direct mode coding for bipredictive slices in the H.264 standard. IEEE Trans Circuits Syst Video Technol 15(1):119–126.

[47] Ugur K, Alshin A, Alshina E, Bossen F, HanW, Park J, Lainema J (2013) Motion compensated prediction and interpolation filter design in H.265/HEVC. IEEE J Sel Top Signal Process 7(6):946–956.

[48] Ugur K, Lainema J, Hallapuro A (2011) High precision bi-directional averaging, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D321, Daegu, Jan. 2011.

[49] Viéron J, Thiesse J-M (2012) On tiles and wavefront tools for parallelism, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0198, Geneva, Apr.-May 2012.

[50] Vivienn Sze, Madhukar Budagavi, Gary I Sullivan, "High Efficiency Video Coding (HEVC) Algorithms and Architectures", Springer 2014.

[51] Wiegand T, Schwarz H, Joch A, Kossentini F, Sullivan G (2003) Rate-constrained coder control and comparison of video coding standards. IEEE Trans Circuits Syst Video Technol 13:688– 703.

[52] Winken M, Bosse S, Bross B, Helle P, Hinz T, Kirchhoffer H, Lakshman H,Marpe D, Oudin S, Preiss M, Schwarz H, Siekmann M, Suehring K, Wiegand T (2010) Video coding technology proposal by Fraunhofer HHI, Joint Collaborative TeamonVideoCoding (JCT-VC), Document JCTVC-A116, Dresden, Apr. 2010.

[53] Zhao L, Zhang L, Ma S, Zhao D (2011) Fast mode decision algorithm for intra prediction in HEVC. In: Proceedings of visual communications and image processing (VCIP), Nov. 2011, O-02-4, pp 1–4.

[54] Zhou M (2012) AHG10: Configurable and CU-group level parallel merge/skip, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0082, San Jose, Feb. 2012.

APPENDIX

LIST OF FIGURES

LIST OF TABLES