Chapter 5
Inter-Picture Prediction in HEVC

Benjamin Bross, Philipp Helle, Haricharan Lakshman, and Kemal Ugur

Abstract Inter-picture prediction in HEVC can be seen as a steady improvement and generalization of all parts known from previous video coding standards, e.g. H.264/AVC. The motion vector prediction was enhanced with advanced motion vector prediction based on motion vector competition. An inter-prediction block merging technique significantly simplified the block-wise motion data signaling by inferring all motion data from already decoded blocks. When it comes to interpolation of fractional reference picture samples, high precision interpolation filter kernels with extended support, i.e. 7/8-tap filter kernels for luma and 4-tap filter kernels for chroma, improve the filtering especially in the high frequency range. Finally, the weighted prediction signaling was simplified by either applying explicitly signaled weights for each motion compensated prediction or just averaging two motion compensated predictions. This chapter provides a detailed description of these aspects of HEVC standard and explains their coding efficiency and complexity characteristics.

5.1 Introduction

In HEVC, the same basic hybrid video coding approach as in previous standards is applied. Hybrid video coding is known to be a combination of video sample prediction and transformation of the prediction error, i.e. the residual, followed by entropy coding of the prediction information and the transform coefficients.

B. Bross (✉) • P. Helle • H. Lakshman
Fraunhofer Institute for Telecommunications - Heinrich Hertz Institute, Einsteinufer 37, D-10587 Berlin, Germany
e-mail: benjamin.bross@hhi.fraunhofer.de

K. Ugur
Nokia Corporation, Tampere 33720, Finland
e-mail: Kemal.Ugur@nokia.com

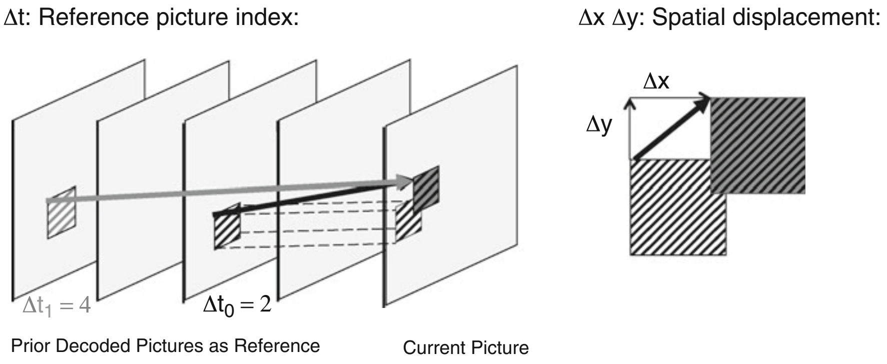
Δt: Reference picture index:
Δx Δy: Spatial displacement:
Δx
Δy
Δt1 = 4
Δt0 = 2
Prior Decoded Pictures as Reference
Current Picture

Fig. 5.1 Inter-picture prediction concept and parameters using a translational motion model

While intra-picture prediction exploits the correlation between spatially neighboring samples, inter-picture prediction makes use of the temporal correlation between pictures in order to derive a motion-compensated prediction (MCP) for a block of image samples.

For this block-based MCP, a video picture is divided into rectangular blocks. Assuming homogeneous motion inside one block and that moving objects are larger than one block, for each block, a corresponding block in a previously decoded picture can be found that serves as a predictor. The general concept of MCP based on a translational motion model is illustrated in Fig. 5.1. Using a translational motion model, the position of the block in a previously decoded picture is indicated by a motion vector $(\Delta x, \Delta y)$ where Δx specifies the horizontal and Δy the vertical displacement relative to the position of the current block. The motion vectors $(\Delta x, \Delta y)$ could be of fractional sample accuracy to more accurately capture the movement of the underlying object. Interpolation is applied on the reference pictures to derive the prediction signal when the corresponding motion vector has fractional sample accuracy. The previously decoded picture is referred to as the reference picture and indicated by a reference index Δt to a reference picture list. These translational motion model parameters, i.e. motion vectors and reference indices, are further referred to as motion data. Two kinds of inter-picture prediction are allowed in modern video coding standards, namely uni-prediction and bi-prediction.

In case of bi-prediction, two sets of motion data $(\Delta x_0, \Delta y_0, \Delta t_0$ and $\Delta x_1, \Delta y_1, \Delta t_1)$ are used to generate two MCPs (possibly from different pictures), which are then combined to get the final MCP. Per default, this is done by averaging but in case of weighted prediction, different weights can be applied to each MCP, e.g. to compensate for scene fade outs. The reference pictures that can be used in bi-prediction are stored in two separate lists, namely list 0 and list 1. In order to limit the memory bandwidth in slices allowing bi-prediction, the HEVC standard restricts PUs with 4×8 and 8×4 luma prediction blocks to use uni-prediction only. Motion data is derived at the encoder using a motion estimation process. Motion

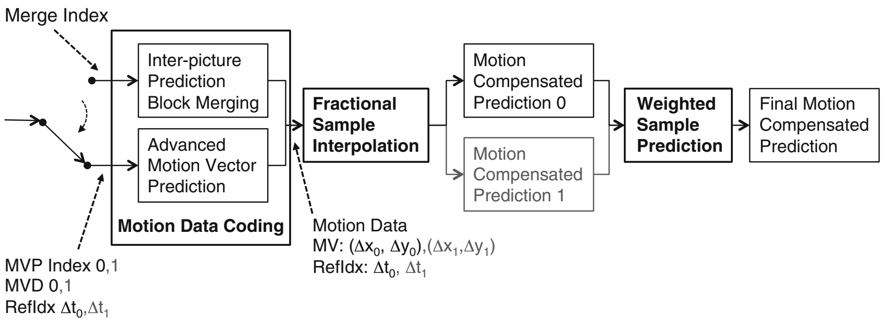
Merge Index
Inter-picture Prediction Block Merging
Advanced Motion Vector Prediction
Motion Data Coding
MVP Index 0,1
MVD 0,1
RefIdx Δt0,Δt1
Fractional Sample Interpolation
Motion Data
MV: (Δx0, Δy0),(Δx1,Δy1)
RefIdx: Δt0, Δt1
Motion Compensated Prediction 0
Motion Compensated Prediction 1
Weighted Sample Prediction
Final Motion Compensated Prediction

Fig. 5.2 Inter-picture prediction in HEVC (*grey parts* represent the bi-prediction path)

estimation is not specified within video standards so different encoders can utilize different complexity-quality tradeoffs in their implementations.

An overview block diagram of the HEVC inter-picture prediction is shown in Fig. 5.2. The motion data of a block is correlated with the neighboring blocks. To exploit this correlation, motion data is not directly coded in the bitstream but predictively coded based on neighboring motion data. In HEVC, two concepts are used for that. The predictive coding of the motion vectors was improved in HEVC by introducing a new tool called advanced motion vector prediction (AMVP) where the best predictor for each motion block is signaled to the decoder. In addition, a new technique called inter-prediction block merging derives all motion data of a block from the neighboring blocks replacing the direct and skip modes in H.264/AVC [26]. Section 5.2 describes all aspects of motion data coding in HEVC including AMVP, inter-prediction block merging and motion data storage reduction. The improved fractional sample interpolation filter is explained in Sect. 5.3. Additional weighting of the MCP or, in case of bi-prediction, the weighting of the two MCPs is further detailed in Sect. 5.4. Finally, Sect. 5.5 summarizes and concludes this chapter.

5.2 Motion Data Coding

5.2.1 Advanced Motion Vector Prediction

As in previous video coding standards, the HEVC motion vectors are coded in terms of horizontal (x) and vertical (y) components as a difference to a so called motion vector predictor (MVP). The calculation of both motion vector difference (MVD) components is shown in Eqs. (5.1) and (5.2).

$$\mathrm{MVD}_x = \Delta x - \mathrm{MVP}_x \tag{5.1}$$

$$\mathrm{MVD}_y = \Delta y - \mathrm{MVP}_y \tag{5.2}$$

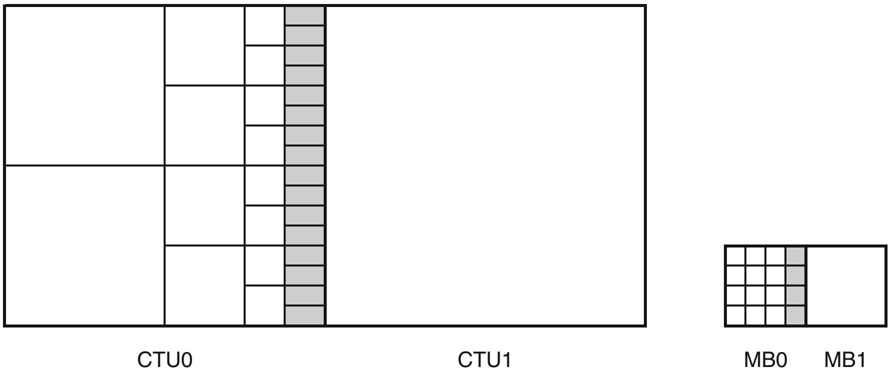

Fig. 5.3 Maximum number of left neighbors with motion data in HEVC with CTU0 having 16 8×4 luma PBs next to CTU1 with one 64×64 luma PB (*left*) and in H.264/AVC with MB0 having four 4×4 partitions next to MB1 with one 16× 16 partition (*right*)

Motion vectors of the current block are usually correlated with the motion vectors of neighboring blocks in the current picture or in the earlier coded pictures. This is because neighboring blocks are likely to correspond to the same moving object with similar motion and the motion of the object is not likely to change abruptly over time. Consequently, using the motion vectors in neighboring blocks as predictors reduces the size of the signaled motion vector difference. The MVPs are usually derived from already decoded motion vectors from spatial neighboring blocks or from temporally neighboring blocks in the co-located picture.[1] In H.264/AVC, this is done by doing a component wise median of three spatially neighboring motion vectors. Using this approach, no signaling of the predictor is required. Temporal MVPs from a co-located picture are only considered in the so called temporal direct mode of H.264/AVC. The H.264/AVC direct modes are also used to derive other motion data than the motion vectors. Hence, they relate more to the block merging concept in HEVC and are further discussed in Sect. 5.2.2.

In HEVC, the approach of implicitly deriving the MVP was replaced by a technique known as motion vector competition, which explicitly signals which MVP from a list of MVPs, is used for motion vector derivation [19]. The variable coding quadtree block structure in HEVC can result in one block having several neighboring blocks with motion vectors as potential MVP candidates. Taking the left neighbor as an example, in the worst case a 64×64 luma prediction block could have 16 8×4 luma prediction blocks to the left when a 64×64 luma coding tree block is not further split and the left one is split to the maximum depth. Figure 5.3 illustrates this example and compares it to the worst case in H.264/AVC. *Advanced Motion Vector Prediction (AMVP)* was introduced to modify motion vector competition to account for such a flexible block structure [11]. During

[1] In some cases, the zero motion vector can also be used as MVP.

the development of HEVC, the initial AMVP design was significantly simplified to provide a good trade-off between coding efficiency and an implementation friendly design. Section 5.2.1.1 describes in detail how the list of potential MVPs is constructed in HEVC. Section 5.2.1.2 describes the signaling of all motion data, including the index to the AMVP list, when AMVP is used for MV coding.

5.2.1.1 AMVP Candidate List Construction

The initial design of AMVP included five MVPs from three different classes of predictors: three motion vectors from spatial neighbors, the median of the three spatial predictors and a scaled motion vector from a co-located, temporally neighboring block. Furthermore, the list of predictors was modified by reordering to place the most probable motion predictor in the first position and by removing redundant candidates to assure minimal signaling overhead. Exhaustive experiments throughout the standardization process investigated how the complexity of this motion vector prediction and signaling scheme could be reduced without sacrificing too much coding efficiency [7, 8, 14]. This led to significant simplifications of the AMVP design such as removing the median predictor, reducing the number of candidates in the list from five to two, fixing the candidate order in the list and reducing the number of redundancy checks. The final design of the AMVP candidate list construction includes the following two MVP candidates:

- up to two *spatial candidate* MVPs that are derived from five spatial neighboring blocks
- one *temporal candidate* MVPs derived from two temporal, co-located blocks when both spatial candidate MVPs are not available or they are identical
- zero motion vectors when the spatial, the temporal or both candidates are not available

Spatial Candidates

As already mentioned, two spatial MVP candidates A and B are derived from five spatially neighboring blocks which are shown in Fig. 5.4b. The locations of the spatial candidate blocks are the same for both AMVP and inter-prediction block merging that will be presented in Sect. 5.2.2.

The derivation process flow for the two spatial candidates A and B is depicted in Fig. 5.5. For candidate A, motion data from the two blocks A0 and A1 at the bottom left corner is taken into account in a two pass approach. In the first pass, it is checked whether any of the candidate blocks contain a reference index that is equal to the reference index of the current block. The first motion vector found will be taken as candidate A. When all reference indices from A0 and A1 are pointing to a different reference picture than the reference index of the current block, the associated motion vector cannot be used as is. Therefore, in a second pass, the

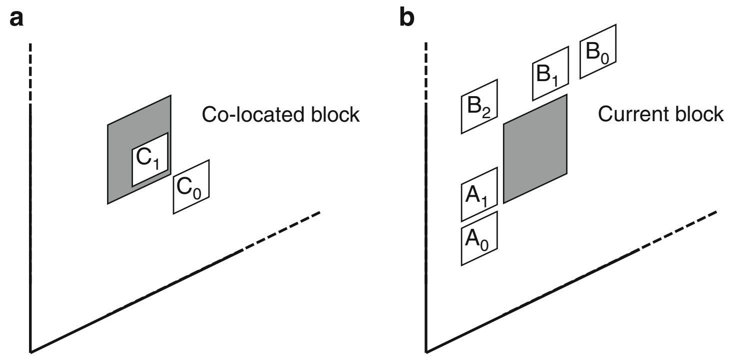

Fig. 5.4 Motion vector predictor and merge candidates. (**a**) Temporal. (**b**) Spatial

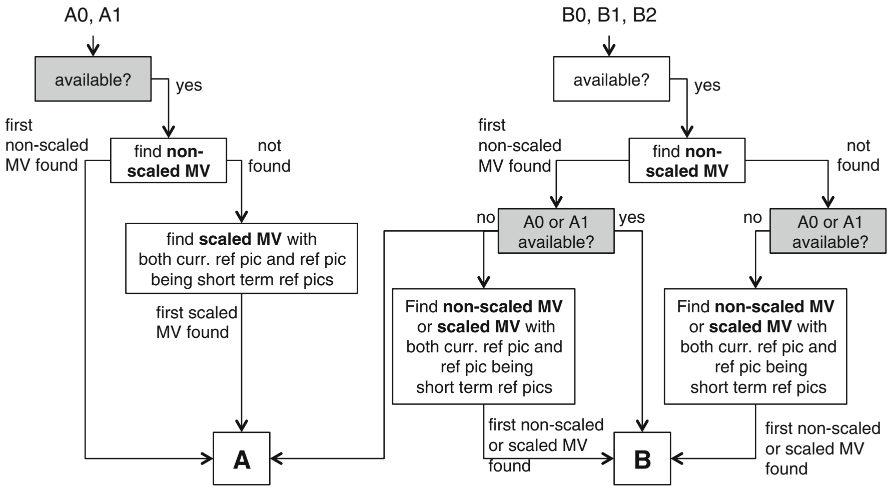
A0, A1
available?
yes
first non-scaled MV found
find non-scaled MV
not found
find scaled MV with both curr. ref pic and ref pic being short term ref pics
first scaled MV found
A
B0, B1, B2
available?
yes
first non-scaled MV found
find non-scaled MV
not found
no
A0 or A1 available?
yes
no
A0 or A1 available?
Find non-scaled MV or scaled MV with both curr. ref pic and ref pic being short term ref pics
first non-scaled or scaled MV found
B
Find non-scaled MV or scaled MV with both curr. ref pic and ref pic being short term ref pics
first non-scaled or scaled MV found

Fig. 5.5 Derivation of spatial AMVP candidates A and B from motion data of neighboring blocks A0, A1, B0, B1 and B2

motion vectors need to be scaled according to the temporal distances between the candidate reference picture and the current reference picture. Equation (5.3) shows how the candidate motion vector $\mathbf{mv}_{\text{cand}}$ is scaled according to a scale factor. ScaleFactor is calculated in Eq. (5.4) based on the temporal distance between the current picture and the reference picture of the candidate block td and the temporal distance between the current picture and the reference picture of the current block tb. The temporal distance is expressed in terms of difference between the picture order count (POC) values which define the display order of the pictures. The scaling operation is basically the same scheme that is used for the temporal direct mode in H.264/AVC. This factoring allows pre-computation of ScaleFactor at slice-level since it only depends on the reference picture list structure signaled in the slice header. Note that the MV scaling is only performed when the current reference picture and the candidate reference picture are both short term reference pictures.

$$\mathbf{mv} = \text{sign}(\mathbf{mv}_{\text{cand}} \cdot \text{ScaleFactor}) \cdot ((|\mathbf{mv}_{\text{cand}} \cdot \text{ScaleFactor}| + 2^7) \gg 8) \tag{5.3}$$

$$\text{ScaleFactor} = \text{clip}(-2^{12}, 2^{12} - 1, (\text{tb} \cdot tx + 2^5) \gg 6) \tag{5.4}$$

$$tx = \frac{2^{14} + |\frac{\text{td}}{2}|}{\text{td}} \tag{5.5}$$

For candidate B, the candidates B0 to B2 are checked sequentially in the same way as A0 and A1 are checked in the first pass. The second pass, however, is only performed when blocks A0 and A1 do not contain any motion information, i.e. are not available or coded using intra-picture prediction. Then, candidate A is set equal to the non-scaled candidate B, if found, and candidate B is set equal to a second, non-scaled or scaled variant of candidate B. Since you could also end up in the second pass when there still might be potential non-scaled candidates, the second pass searches for non-scaled as well as for scaled MVs derived from candidates B0 to B2.

Overall, this design allows to process A0 and A1 independently from B0, B1, and B2. The derivation of B should only be aware of the availability of both A0 and A1 in order to search for a scaled or an additional non-scaled MV derived from B0 to B2. This dependency is acceptable given that it significantly reduces the complex motion vector scaling operations for candidate B. Reducing the number of motion vector scalings represents a significant complexity reduction in the motion vector predictor derivation process.

Temporal Candidate

It can be seen from Fig. 5.4b that only motion vectors from spatial neighboring blocks to the left and above the current block are considered as spatial MVP candidates. This can be explained by the fact that the blocks to the right and below the current block are not yet decoded and hence, their motion data is not available. Since the co-located picture is a reference picture which is already decoded, it is possible to also consider motion data from the block at the same position, from blocks to the right of the co-located block or from the blocks below. In HEVC, the block to the bottom right and at the center of the current block have been determined to be the most suitable to provide a good temporal motion vector predictor (TMVP). These candidates are illustrated in Fig. 5.4a where C0 represents the bottom right neighbor and C1 represents the center block. Here again, motion data of C0 is considered first and, if not available, motion data from the co-located candidate block at the center is used to derive the temporal MVP candidate C. The motion data of C0 is also considered as not being available when the associated

PU belongs to a CTU beyond the current CTU row. This minimizes the memory bandwidth requirements to store the co-located motion data. In contrast to the spatial MVP candidates, where the motion vectors may refer to the same reference picture, motion vector scaling is mandatory for the TMVP. Hence, the same scaling operation from Eq. (5.3) as for the spatial MVPs is used whereas td is defined as the POC difference between the co-located picture and the reference picture of the co-located candidate block.

While the temporal direct mode in H.264/AVC always refers to the first reference picture in the second reference picture list, list 1, and is only allowed in bi-predictive slices, HEVC offers the possibility to indicate for each picture which reference picture is considered as the co-located picture. This is done by signaling in the slice header the co-located reference picture list and reference picture index as well as requiring that these syntax elements in all slices in a picture should specify the same reference picture.

Since the temporal MVP candidate introduces additional dependencies, it might be desirable to disable its usage for error robustness reasons. In H.264/AVC there is the possibility to disable the temporal direct mode for bi-predictive slices in the slice header (`direct_spatial_mv_pred_flag`). HEVC syntax extends this signaling by allowing to disable the TMVP at sequence level or at picture level (`sps/slice_temporal_mvp_enabled_flag`). Although the flag is signaled in the slice header, it is a requirement of bitstream conformance that its value shall be the same for all slices in one picture. Since the signaling of the picture-level flag depends on the SPS flag, signaling it in the PPS would introduce a parsing dependency between SPS and PPS. Another advantage of this slice header signaling is that if you want to change only the value of this flag and no other parameter in the PPS, there is no need to transmit a second PPS.

5.2.1.2 AMVP Motion Data Signaling

In general, motion data signaling in HEVC is similar as in H.264/AVC. An inter-picture prediction syntax element, `inter_pred_idc`, signals whether reference list 0, 1 or both are used. For each MCP obtained from one reference picture list, the corresponding reference picture (Δt) is signaled by an index to the reference picture list, `ref_idx_l0/1`, and the MV (Δx, Δy) is represented by an index to the MVP, `mvp_l0/1_flag`, and its MVD. The MVD syntax is further detailed in Chap. 8. A newly introduced flag in the slice header, `mvd_l1_zero_flag`, indicates whether the MVD for the second reference picture list is equal to zero and therefore not signaled in the bitstream. When the motion vector is fully reconstructed, a final clipping operation assures that the values of each component of the final motion vector will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

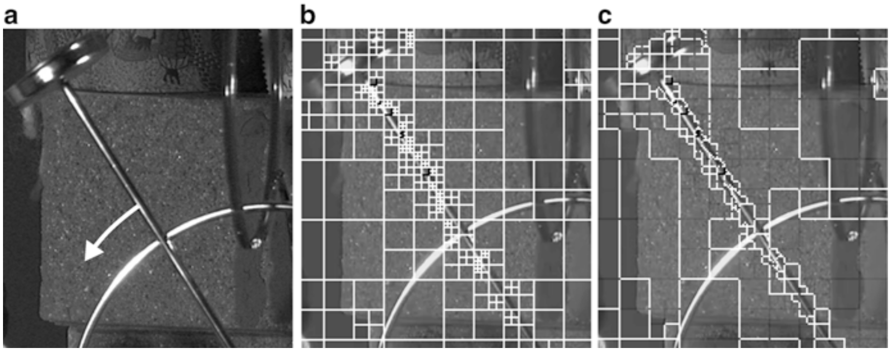

Fig. 5.6 Detail of a video sequence exemplifying the merge concept. (**a**) In the foreground, the scene contains a moving object (a pendulum) with motion indicated by the *arrow*. (**b**) A rate-distortion optimized quadtree partitioning for inter-picture prediction parameters is indicated by the *white borders*. (**c**) Only effective block borders are shown, i.e. borders that separate blocks with different motion parameters. Reproduced with permission from [12], © 2012 IEEE

5.2.2 Inter-picture Prediction Block Merging

In image or video compression it is very reasonable to deploy a block based image partitioning mechanism in order to apply different prediction models to different regions of an image. This is because a single model can in general not be expected to capture the versatile characteristics of a whole image or video. HEVC uses a quadtree structure to describe the partitioning of a region into sub-blocks. In terms of bit rate, this is a very low-cost structure while at the same time, it allows for partitioning into a wide range of differently sized sub-blocks. While this simplicity is an advantage for example for encoder design, it also bears the disadvantage of over-segmenting the image, potentially leading to redundant signaling and ineffective borders. This drawback is effectively addressed by block merging as explained in Sect. 5.2.2.1. A description of the exact algorithm and bitstream syntax follows in Sects. 5.2.2.2–5.2.2.5.

5.2.2.1 Background

An example partitioning using the quadtree structure is shown in Fig. 5.6a for a uni-predictive slice in an HEVC-encoded video. As can be seen, the area around the pendulum is heavily partitioned in order to capture the motion in front of the still background. Figure 5.6c shows the same partitioning, but without ineffective borders, i.e. borders dividing regions of equal motion parameters. It becomes evident that in this particular situation, the quadtree structure is unable to accurately capture the motion without introducing ineffective borders. It is easy to see that this over-segmentation easily occurs whenever moving objects in a scene cause abrupt changes in the field of motion parameters, which is common in natural video content. One reason is that the quad-tree structure systematically does not allow for joint description of child blocks that belong to different parent blocks. Also, the fact that a block can only be divided into exactly four child blocks will eventually lead to ineffective borders.

To remedy these inherent drawbacks of the quad-tree structure, HEVC uses block merging which allows us to code motion parameters very cheaply (in terms of bit rate) in these ineffective border situations [9, 12, 23]. The algorithm was inspired by the work of [10], in which the authors show that rate-distortion optimized tree pruning for quadtree-based motion models can be substantially improved by introducing a subsequent leaf merging step. In [22], the authors study the benefits of leaf merging on a broader theoretical basis. Here we leave it at the intuitive example given above and concentrate on the integration of block merging into HEVC. Following from the observation of ineffective borders, block merging introduces a terse syntax allowing for a sub-block to explicitly reuse the exact same motion parameters contained in neighboring blocks. Like AMVP, it compiles a list of candidate motion parameter tuples by picking from neighboring blocks. Then, an index is signaled which identifies the candidate to be used. Block merging also allows for temporal prediction by including into the list a candidate obtained from previously coded pictures. A more detailed description is given in the following.

5.2.2.2 Merge Candidate List Construction

Although they appear similar, there is one main difference between the AMVP and the merge candidate list. The AMVP list only contains motion vectors for one reference list while a merge candidate contains all motion data including the information whether one or two reference picture lists are used as well as a reference index and a motion vector for each list. This significantly reduces motion data signaling overhead. Section 5.2.2.3 describes the signaling in detail as well as discusses how parsing robustness is achieved. Overall, the merge candidate list is constructed based on the following candidates:

- up to four *spatial merge candidates* that are derived from five spatial neighboring blocks
- one *temporal merge candidate* derived from two temporal, co-located blocks
- *additional merge candidates* including combined bi-predictive candidates and zero motion vector candidates

Spatial Candidates

The first candidates in the merge candidate list are the spatial neighbors. Here, the same neighboring blocks as for the spatial AMVP candidates are considered which are described in Sect. 5.2.1.1 and illustrated in Fig. 5.4b. In order to derive a list of motion vector predictors for AMVP, one MVP is derived from A0 and A1 and one from B0, B1 and B2, respectively in that order. However, for inter-prediction block merging, up to four candidates are inserted in the merge list by sequentially checking A1, B1, B0, A0 and B2, in that order.

Instead of just checking whether a neighboring block is available and contains motion information, some additional redundancy checks are performed before taking all the motion data of the neighboring block as a merge candidate. These redundancy checks can be divided into two categories for two different purposes:

- avoid having candidates with redundant motion data in the list
- prevent merging two partitions that could be expressed by other means which would create redundant syntax

When N is the number of spatial merge candidates, a complete redundancy check would consist of $\frac{N\cdot(N-1)}{2}$ motion data comparisons. In case of the five potential spatial merge candidates, ten motion data comparisons would be needed to assure that all candidates in the merge list have different motion data. During the development of HEVC, the checks for redundant motion data have been reduced to a subset in a way that the coding efficiency is kept while the comparison logic is significantly reduced [1]. In the final design, no more than two comparisons are performed per candidate resulting in five overall comparisons. Given the order of {A1, B1, B0, A0, B2}, B0 only checks B1, A0 only A1 and B2 only A1 and B1.

For an explanation of the partitioning redundancy check consider the following example. The bottom PU of a 2N×N partitioning is merged with the top one by choosing candidate B1. This would result in one CU with two PUs having the same motion data which could be equally signaled as a 2N×2N CU. Overall, this check applies for all second PUs of the rectangular and asymmetric partitions 2N×N, 2N×nU, 2N×nD, N×2N, nR×2N and nL×2N. Please note that for the spatial merge candidates, only the redundancy checks are performed and the motion data is copied from the candidate blocks as it is. Hence, no motion vector scaling is needed here.

Temporal Candidate

The derivation of the motion vectors for the temporal merge candidate is the same as for the TMVP described in Sect. 5.2.1.1. Since a merge candidate comprises all motion data and the TMVP is only one motion vector, the derivation of the whole motion data only depends on the slice type. For bi-predictive slices, a TMVP is derived for each reference picture list. Depending on the availability of the TMVP for each list, the prediction type is set to bi-prediction or to the list for which the TMVP is available. All associated reference picture indices are set equal to zero. Consequently for uni-predictive slices, only the TMVP for list 0 is derived together with the reference picture index equal to zero.

When at least one TMVP is available and the temporal merge candidate is added to the list, no redundancy check is performed. This makes the merge list construction independent of the co-located picture which improves error resilience. Consider the case where the temporal merge candidate would be redundant and therefore not

Table 5.1 Order in which motion data combinations of different merge candidates, that have been already inserted in the merge list, are tested to create combined bi-predictive merge candidates

Combination Order	0	1	2	3	4	5	6	7	8	9	10	11
$\Delta x_0, \Delta y_0, \Delta t_0$ from Cand.	0	1	0	2	1	2	0	3	1	3	2	3
$\Delta x_1, \Delta y_1, \Delta t_1$ from Cand.	1	0	2	0	2	1	3	0	3	1	3	2

included in the merge candidate list. In the event of a lost co-located picture, the decoder could not derive the temporal candidates and hence not check whether it would be redundant. The indexing of all subsequent candidates would be affected by this.

Additional Candidates

For parsing robustness reasons, which will be explained in Sect. 5.2.2.3, the length of the merge candidate list is fixed. After the spatial and the temporal merge candidates have been added, it can happen that the list has not yet the fixed length. In order to compensate for the coding efficiency loss that comes along with the non-length adaptive list index signaling, additional candidates are generated [25]. Depending on the slice type, up to two kind of candidates are used to fully populate the list:

- Combined bi-predictive candidates
- Zero motion vector candidates

In bi-predictive slices, additional candidates can be generated based on the existing ones by combining reference picture list 0 motion data of one candidate with and the list 1 motion data of another one. This is done by copying $\Delta x_0, \Delta y_0, \Delta t_0$ from one candidate, e.g. the first one, and $\Delta x_1, \Delta y_1, \Delta t_1$ from another, e.g. the second one. The different combinations are predefined and given in Table 5.1.

When the list is still not full after adding the combined bi-predictive candidates, or for uni-predictive slices, zero motion vector candidates are calculated to complete the list. All zero motion vector candidates have one zero displacement motion vector for uni-predictive slices and two for bi-predictive slices. The reference indices are set equal to zero and are incremented by one for each additional candidate until the maximum number of reference indices is reached. If that is the case and there are still additional candidates missing, a reference index equal to zero is used to create these. For all the additional candidates, no redundancy checks are performed as it turned out that omitting these checks will not introduce a coding efficiency loss [21].

5.2.2.3 Merge Motion Data Signaling and Skip Mode

The motion data signaling scheme using the merge mode is quite simple. For each PU coded in inter-picture prediction mode, a so called `merge_flag` indicates that block merging is used to derive the motion data. The `merge_idx` further determines the candidate in the merge list that provides all the motion data needed for the MCP. Therefore, instead of all the syntax elements needed for AMVP based motion data coding described in Sect. 5.2.1.2, only a flag and a list index are transmitted. This difference can be seen when comparing the input to the AMVP and the merge motion data coding block in Fig. 5.2.

Besides this PU-level signaling, the number of candidates in the merge list is signaled in the slice header. Since the default value is five, it is represented as a difference to five (`five_minus_max_num_merge_cand`). That way, the five is signaled with a short codeword for the 0 whereas using only one candidate, is signaled with a longer codeword for the 4. Regarding the impact on the merge candidate list construction process, the overall process remains the same although it terminates after the list contains the maximum number of merge candidates. 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. When e.g. only two candidates are available, the index can be efficiently coded as a flag. But, in order to parse the merge index, the whole merge candidate list has to be constructed to know the actual number of candidates. Assuming unavailable neighboring blocks due to transmission errors, it would not be possible to parse the merge index anymore. Fixing the number of merge candidates improves the parsing robustness by decoupling the parsing and the merge candidate list construction while sacrificing coding efficiency. Populating the list with the additional merge candidates presented in Sect. 5.2.2.2 compensates again for that loss while keeping the parsing robustness.

A crucial application of the block merging concept in HEVC is its combination with a skip mode. In previous video coding standards, the skip mode was used to indicate for a block that the motion data is inferred instead of explicitly signaled and that the prediction residual is zero, i.e. no transform coefficients are transmitted. This mode is well suited to code static image regions where the prediction error tends to be very small. In HEVC, at the beginning of each CU in an inter-picture prediction slice, a `skip_flag` is signaled that implies the following:

- the CU only contains one PU (2N×2N partition type)
- the merge mode is used to derive the motion data (`merge_flag` equal to 1)
- no residual data is present in the bitstream

5.2.2.4 Coding Efficiency of HEVC Merge and Skip Mode

In this section, the coding efficiency of the HEVC merge and skip modes is analyzed. This is done experimentally by disabling the merge mode as well as the skip mode, i.e. removing `merge_flag`, `merge_index` and `skip_flag`

Table 5.2 Average bit rate savings of HEVC merge and skip mode using HM8.0

		BD-rate [%]		
Class	Sequence	RA-Main	LB-Main	LD-Main
A	Traffic	−8.8	n/a	n/a
(2560 × 1600)	PeopleOnStreet	−6.5	n/a	n/a
	Nebuta	−1.5	n/a	n/a
	SteamLocomotive	−11.1	n/a	n/a
B	Kimono	−9.6	−7.8	−5.4
(1920 × 1080)	ParkScene	−7.3	−7.0	−5.6
	Cactus	−11.7	−8.8	−6.9
	Basketball Drive	−9.1	−8.1	−6.1
	BQTerrace	−10.2	−10.3	−5.9
C	Basketball Drill	−7.5	−7.6	−6.1
(832 × 480)	BQMall	−9.2	−7.5	−5.7
	PartyScene	−4.8	−3.4	−2.5
	RaceHorses	−3.9	−4.3	−3.3
D	Basketball Pass	−6.1	−5.1	−4.0
(416 × 240)	BQSquare	−6.9	−3.3	−2.5
	BlowingBubbles	−6.1	−3.7	−3.1
	RaceHorses	−4.4	−3.9	−3.6
E	FourPeople	n/a	−11.4	−8.5
(1280 × 720)	Johnny	n/a	−20.0	−16.4
	KristenAndSara	n/a	−15.0	−11.2
Average		**−7.3**	**−8.0**	**−6.0**

syntax. The software used for this experiment is version 8.0 of the HEVC test model reference software (HM) [13] with the random access main, low delay B and P main coding configurations as described in [5].

The coding efficiency gains in terms of Bjøntegaard Delta (BD) rate [2], when enabling the merge and skip modes, are reported in [9] and [12] and also summarized in Table 5.2. Average bit rate savings between 6 % and 8 % are observed. It can be seen that the gains for the random access main (RA-Main) and low delay B main (LB-Main) configurations using bi-predictive B pictures are higher than for the low delay P main (LP-Main) configuration, which is restricted to use uni-prediction. Since the merge mode only uses a flag and an index to signal all motion data, it is more efficient the more motion data signaling is omitted that way, e.g. two sets of motion data in bi-prediction. Another observation is that merge and skip modes are saving up to 20 % for the class E sequences. These sequences represent videoconferencing content where the static background can efficiently be coded using skip and merge modes. More detailed results and a comparison with an AMVP-based direct mode can be found in [12].

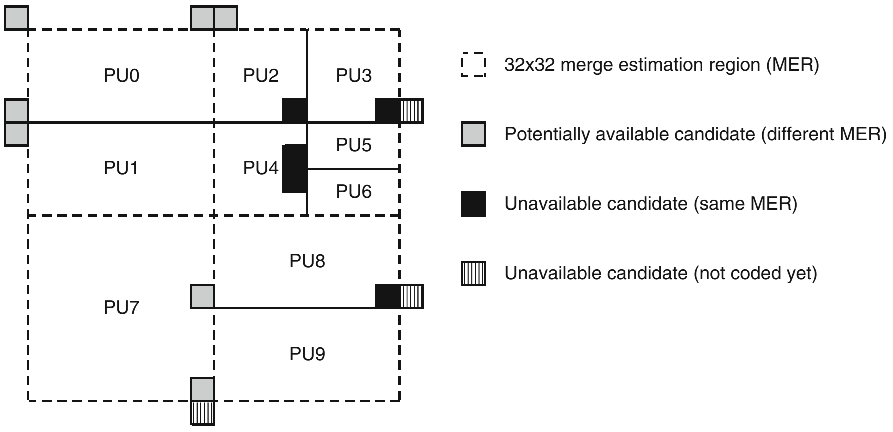
PU0
PU2
PU3
PU1
PU4
PU5
PU6
PU7
PU8
PU9
32x32 merge estimation region (MER)
Potentially available candidate (different MER)
Unavailable candidate (same MER)
Unavailable candidate (not coded yet)

Fig. 5.7 Example of a CTU with a 64×64 luma CTB, when motion estimation of for PUs inside a 32×32 motion estimation region is carried out independently, enabling the possibility to do it in parallel

5.2.2.5 Merge Estimation Regions for Parallel Merge Mode Estimation

The way the merge candidate list is constructed introduces dependencies between neighboring blocks. Especially in embedded encoder implementations, the motion estimation stage of neighboring blocks is typically performed in parallel or at least pipelined to increase the throughput. For AMVP, this is not a big issue since the MVP is only used to differentially code the MV found by the motion search. The motion estimation stage for the merge mode, however, would typically just consist of the candidate list construction and the decision which candidate to choose, based on a cost function. Due to the aforementioned dependency between neighboring blocks, merge candidate lists of neighboring blocks cannot be generated in parallel and represent a bottleneck for parallel encoder designs. Therefore, a parallel merge estimation level was introduced in HEVC that indicates the region in which merge candidate lists can be independently derived by checking whether a candidate block is located in that merge estimation region (MER). A candidate block that is in the same MER is not included in the merge candidate list. Hence, its motion data does not need to be available at the time of the list construction. When this level is e.g. 32, all prediction units in a 32×32 area can construct the merge candidate list in parallel since all merge candidates that are in the same 32×32 MER, are not inserted in the list. Figure 5.7 illustrates that example showing a CTU partitioning with seven CUs and ten PUs. All potential merge candidates for the first PU0 are available because they are outside the first 32×32 MER. For the second MER, merge candidate lists of PUs 2–6 cannot include motion data from these PUs when the merge estimation inside that MER should be independent. Therefore, when looking at a PU5 for example, no merge candidates are available and hence not inserted in

Table 5.3 Average bit rate losses for different merge estimation regions in terms of BD-rate using HM5.0 reference software

	Merge estimation region			
	8×8	16×16	32×32	64×64
RA-HE	0.1 %	0.6 %	1.6 %	2.7 %
LB-HE	0.2 %	0.7 %	2.0 %	3.4 %

the merge candidate list. In that case, the merge list of PU5 consists only of the temporal candidate (if available) and zero MV candidates.

In order to enable an encoder to trade-off parallelism and coding efficiency, the parallel merge estimation level is adaptive and signaled as `log2_parallel_merge_level_minus2` in the picture parameter set. The following MER sizes are allowed: 4×4 (no parallel merge estimation possible), 8×8, 16×16, 32×32 and 64×64. A higher degree of parallelization, enabled by a larger MER, excludes more potential candidates from the merge candidate list. That, on the other hand, decreases the coding efficiency. The coding efficiency losses in terms of BD-rate [2] for different MER sizes are reported in [28] and summarized in Table 5.3. Results are generated using HM5.0 [13] with random access high efficiency (RA-HE) and low delay B high efficiency (LB-HE) coding configurations as described in [3].

When the merge estimation region is larger than a 4×4 block, another modification of the merge list construction to increase the throughput kicks in. For a CU with an 8×8 luma CB, only a single merge candidate list is used for all PUs inside that CU.

5.2.3 Motion Data Storage Reduction

The usage of the TMVP, in AMVP as well as in the merge mode, requires the storage of the motion data (including motion vectors, reference indices and coding modes) in co-located reference pictures. Considering the granularity of motion representation, the memory size needed for storing motion data could be significant. HEVC employs *motion data storage reduction* (MDSR) to reduce the size of the motion data buffer and the associated memory access bandwidth by sub-sampling motion data in the reference pictures [20]. While H.264/AVC is storing these information on a 4×4 block basis, HEVC uses a 16×16 block where, in case of sub-sampling a 4×4 grid, the information of the top-left 4×4 block is stored. Due to this sub-sampling, MDSR impacts on the quality of the temporal prediction. Furthermore, there is a tight correlation between the position of the MV used in the co-located picture, and the position of the MV stored by MDSR.

During the standardization process of HEVC, the impact of the sub-sampling scheme as well as the interaction with the TMVP was investigated in a core experiment on MDSR [15]. It turned out that storing the motion data of the top left block inside the 16×16 area together with the bottom right and center

TMVP candidates provide the best tradeoff between coding efficiency and memory bandwidth reduction. Furthermore, the general impact of sub-sampling the motion information was measured. While the 8×8 subsampling show no difference in coding efficiency compared to 4×4, the current 16×16 scheme results in a coding efficiency loss of 0.1 % BD-rate which is negligible and can be considered as being in the noise margin.

5.3 Fractional Sample Interpolation

Interpolation tasks arise naturally in the context of video coding because the true displacements of objects from one picture to another are independent of the sampling grid of cameras. Therefore, in MCP, fractional-sample accuracy is used to more accurately capture continuous motion. Samples available at integer positions are filtered to estimate values at fractional positions. This spatial domain operation can be seen in the frequency domain as introducing phase delays to individual frequency components. An ideal interpolation filter for band-limited signals induces a constant phase delay to all frequencies and does not alter their magnitudes. The efficiency of MCP is limited by many factors—the spectral content of original and already reconstructed pictures, camera noise level, motion blur, quantization noise in reconstructed pictures, etc.

Similar to H.264/AVC, HEVC supports motion vectors with quarter-pixel accuracy for the luma component and one-eighth pixel accuracy for chroma components. If the motion vector has a half or quarter-pixel accuracy, samples at fractional positions need to be interpolated using the samples at integer-sample positions. The interpolation process in HEVC introduces several improvements over H.264/AVC that contributes to the significant coding efficiency increase of HEVC. In this section, these differences are first explained and then the complexity and coding efficiency characteristics of the HEVC interpolation process are presented.

5.3.1 Overview

In order to improve the filter response in the high frequency range, luma and chroma interpolation filters have been re-designed and the tap-lengths were increased. The luma interpolation process in HEVC uses a symmetric 8-tap filter for half-sample positions and an asymmetric 7-tap filter for quarter-sample positions. For chroma samples, a 4-tap filter was introduced.

The intermediate values used in interpolation process are kept at a higher accuracy in HEVC to improve coding efficiency. This is done as follows (please refer to Fig. 5.9 for notation throughout the text, where integer-sample values are shown with dark squares and the fractional-sample values are shown with white squares):

- H.264/AVC obtains the quarter-sample values by first obtaining the values of nearest half-pixel samples and averaging those with the nearest integer samples, according the position of the quarter-pixel [27]. However, HEVC obtains the quarter-pixel samples without using such cascaded steps but by instead directly applying a 7 or 8-tap filter on the integer pixels.
- 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. 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.

The details of these features are presented in the following sections.

5.3.1.1 Redesigned Filters

An important parameter for interpolation filters is the number of filter taps as it has a direct influence on both coding efficiency and implementation complexity. In terms of implementation, it not only has an impact on the arithmetic operations but also on the memory bandwidth required to access the reference samples. Although the 6-tap filter for estimating half-pixel positions in H.264/AVC produces a constant phase delay of 0.5 for all frequency components due to symmetry, the passband (range of frequencies where the magnitudes are relatively unaltered) is not large. Increasing the number of taps can yield filters that produce desired response for a larger range of frequencies which can help to predict the corresponding frequencies in the samples to be coded. Considering modern computing capabilities, the performance of many MCP filters were evaluated in the context of HEVC and a coding efficiency/complexity trade-off was targeted during the standardization.

Consider the design of a half-pixel interpolation filter with $2N$ taps denoted as $\mathbf{h} = [h_0, h_1, \cdots, h_{2N-1}]^T$. Due to the desired half-pixel symmetry only N coefficients can be different, which can be denoted in the form $\mathbf{h} = [h_0, h_1, \cdots, h_{N-1}, \cdots, h_1, h_0]^T$. Now consider the interpolation of a DC signal (with all samples equal). It is desired that the interpolated value be the same as the input. Hence, we require $2 \cdot \sum_{n=0}^{N-1} h_n = 1$, also known as the normalization constraint. This further reduces the number of degrees of freedom from N to $N-1$. In the case of a quarter-pixel interpolation filter however, only the normalization condition $\sum_{n=0}^{2N-1} h_n = 1$ is imposed as symmetry is not necessary, which gives $2N - 1$ degrees of freedom. The aim of interpolation filter design is to determine these degrees of freedom so as to remain close to the desired frequency response.

Here a brief overview of the design of HEVC interpolation filters is provided. For a detailed explanation the reader is referred to [17]. The basic idea is to forward transform the known integer samples to the DCT domain and inverse transform

Table 5.4 Filter coefficients for luma interpolation in MCP

Phase	Luma filter coefficients
1/4	$[-1, 4, -10, 58, 17, -5, 1]/64$
1/2	$[-1, 4, -11, 40, 40, -11, 4, -1]/64$

Table 5.5 Filter coefficients for chroma interpolation in MCP

Phase	Chroma filter coefficients
1/8	$[-2, 58, 10, -2]/64$
1/4	$[-4, 54, 16, -2]/64$
3/8	$[-6, 46, 28, -4]/64$
1/2	$[-4, 36, 36, -4]/64$

the DCT coefficients to the spatial domain using DCT basis sampled at desired fractional positions instead of integer positions. Fortunately, these operations can be combined into a single FIR filtering step. Let the available samples at integer positions be denoted as a column vector $\mathbf{s}$ and the forward transform as a matrix $\mathbf{B}$. The DCT coefficients $\mathbf{c}$ can be computed as $\mathbf{c} = \mathbf{B} \cdot \mathbf{s}$. Since DCT basis is composed of cosine functions (which are continuous in nature), they can be sampled at fractional positions. Let the DCT basis sampled at desired fractional positions be denoted by a row vector $\mathbf{r}$. This is used to transform back the DCT coefficients to the spatial domain, which results in the interpolated value $\hat{s} = \mathbf{r} \cdot \mathbf{B} \cdot \mathbf{s}$. These stages can be combined into a single filter $\mathbf{f} = \mathbf{r} \cdot \mathbf{B}$. In addition to the above steps, the reference samples are smoothed in the actual HEVC design to combat noise in the reference samples. Therefore the final interpolation filter can be written in the form $\mathbf{f} = \mathbf{r} \cdot \mathbf{B} \cdot \mathbf{W}$, where $\mathbf{W}$ is a diagonal matrix with weights for smoothing. The resulting filter coefficients are rounded to 6-bit precision (for a simple fixed-point implementation) and an integer optimization is carried out under the normalization constraint to ensure that the filter coefficients provide close to desired frequency response even after rounding. For the chroma interpolation filters, a slightly different smoothing of reference samples is performed during the filter design. The filter coefficients' bit-depth of 6 also makes it possible to realize the entire MCP process for 8-bit videos using 16-bit intermediate buffers. The filter coefficients resulting from the design described above for luma and chroma MCP are given in Tables 5.4 and 5.5, respectively. The magnitude responses of half-pel luma interpolation filters of H.264/AVC and HEVC are depicted in Fig. 5.8. It can be seen that the half-pel interpolation filter of HEVC comes closer to the desired response than the H.264/AVC filter.

5.3.1.2 High Precision Filtering Operations

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

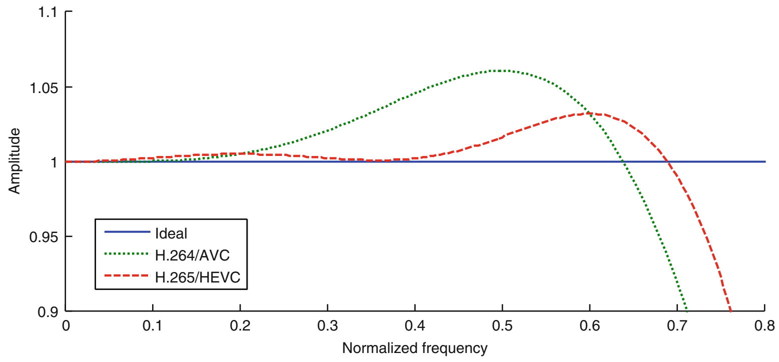

Fig. 5.8 Comparison of magnitude response of half-pel interpolation filters of H.264/AVC and HEVC relative to an ideal filter

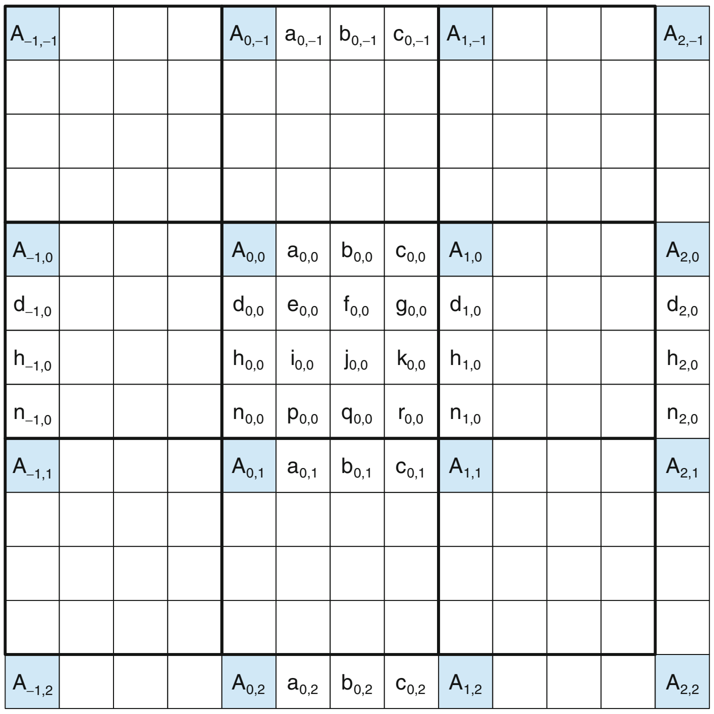

Fig. 5.9 Fractional positions used in luma motion compensation with 1/4 pixel accuracy

samples obtained by 6-tap FIR filter are first rounded to input bit-depth, prior to using those for obtaining the quarter-pixel samples. To illustrate this effect, let's consider interpolating the quarter-pixel sample $a_{0,0}$ using the H.264/AVC interpolation process. In order to obtain $a_{0,0}$, the half-pixel sample $b_{0,0}$ needs to be obtained first by applying a 6-tap horizontal FIR filter and rounding the result to input bit-depth, as shown in Eq. (5.6). The quarter-pixel sample $a_{0,0}$ is then obtained by averaging the half-pixel sample, $b_{0,0}$ with the integer sample $A_{0,0}$ as shown in Eq. (5.7). Because $b_{0,0}$ is first rounded back to input bit-depth, a rounding error of 33/128 is introduced to obtain $a_{0,0}$ [16].

$$b_{0,0} = (A_{-2,0} - 5 \cdot A_{-1,0} + 20 \cdot A_{0,0} + 20 \cdot A_{1,0} - 5 \cdot A_{2,0} + A_{3,0} + 16) >> 5 \tag{5.6}$$

$$a_{0,0} = (A_{0,0} + b_{0,0} + 1) >> 1 \tag{5.7}$$

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 shown in Sect. 5.3.1.1, which significantly reduces the rounding error to 1/128.

The second reason for reduction of accuracy in H.264/AVC motion compensation process is due to averaging in bi-prediction. In H.264/AVC, the prediction signal of the bi-predictively coded motion blocks (denoted by S) is obtained by averaging prediction signals from two prediction lists (denoted by S_1 and S_2) as shown in Eq. (5.8).

$$S = (S_1 + S_2 + 1) >> 1 \tag{5.8}$$

The averaging operation shown in Eq. (5.8) is done at the precision of input bit-depth (i.e. S_1 and S_2 are 8-bit for an 8-bit video). If the motion vectors have fractional pixel accuracy, then S_1 and S_2 are obtained using interpolation and the intermediate values are rounded to input bit-depth. 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 [18]. This means that, if the motion vectors to obtain S_1 or S_2 have sub-pixel accuracy, the interpolation process does not round the intermediate values to input-bit depth prior to averaging, but keeps it at a higher precision. 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.

This process is illustrated in Fig. 5.10 (a) for the case of H.264/AVC where bi-prediction averages two prediction signals at input bit-depth and (b) for HEVC where the averaging is performed at a higher bit-depth and intermediate rounding step is not used.

5.3.1.3 Other Important Features

To make sure 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

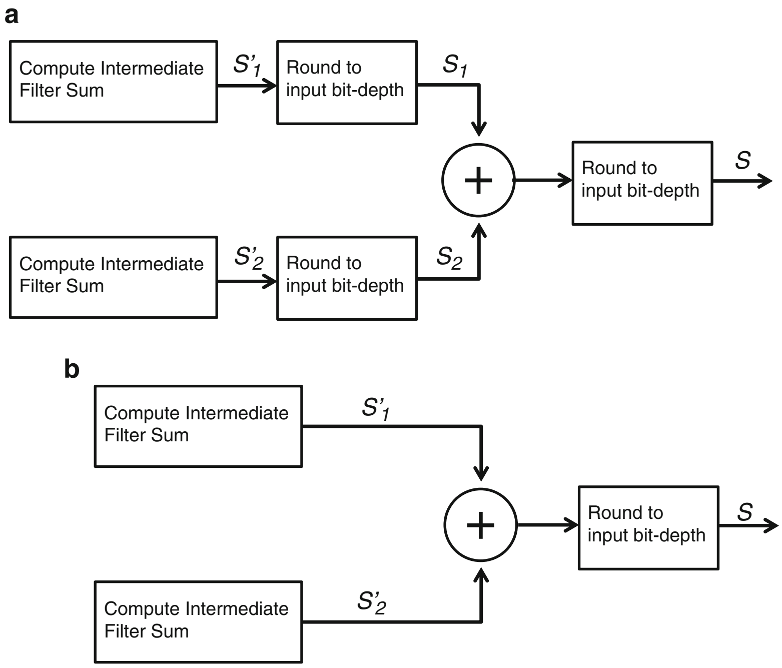
a
Compute Intermediate Filter Sum
S'1
Round to input bit-depth
S1
Compute Intermediate Filter Sum
S'2
Round to input bit-depth
S2
+
Round to input bit-depth
S
b
Compute Intermediate Filter Sum
S'1
Compute Intermediate Filter Sum
S'2
+
Round to input bit-depth
S

Fig. 5.10 Bi-prediction process in (**a**) H.264/AVC and in (**b**) HEVC. Reproduced with permission from [17], © 2013 IEEE

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.

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.

5.3.2 *Complexity and Coding Efficiency Characteristics*

In this section, the complexity of the interpolation filter design in HEVC is analyzed and compared with that of H.264/AVC. In addition, the coding efficiency improvements brought with the improved design are also presented.

5.3.2.1 Complexity of HEVC Interpolation Filter

When evaluating the complexity of a video coding algorithm, several aspects, such as memory bandwidth, number of operations and storage buffer size need to be carefully considered.

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 (such as position $f_{0,0}$). 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 [17].

Similarly, the longer tap-length filters increase the number of arithmetic operations required to obtain the interpolated sample. If the complexity is measured by the number of multiply-and-add operations (MACs), interpolation filter in HEVC represents roughly a 20 % increase over H.264/AVC filter for 8-bit video.

The high-precision bi-directional averaging described in Sect. 5.3.1.2 increases the size of intermediate storage buffers for storing the temporary uni-prediction signals as each one of the prediction signals need to be stored at a higher bit-depth compared to H.264/AVC before the bi-directional averaging takes place.

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 more costly 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.

5.3.2.2 Coding Efficiency of HEVC Interpolation Filter

In this section, the coding efficiency of interpolation filter design in HEVC is analyzed. For this purpose, the H.264/AVC interpolation filter is first implemented in version 6.0 of the HEVC test model and then run with the test conditions advised by JCT-VC [4]. Same test model is also run with the HEVC interpolation filter

Table 5.6 Average bit rate savings of HEVC interpolation filter for the luma component using HM6.0

		BD-rate [%]		
Class	Sequence	RA-Main	LB-Main	LD-Main
A	Traffic	−2.4	n/a	n/a
(2560 × 1600)	PeopleOnStreet	0.1	n/a	n/a
	Nebuta	0.4	n/a	n/a
	SteamLocomotive	−1.0	n/a	n/a
B	Kimono	−1.8	−2.6	0.6
(1920 × 1080)	ParkScene	−2.7	−4.5	−2.2
	Cactus	−1.3	−2.9	−0.8
	Basketball Drive	−2.0	−3.1	−0.2
	BQTerrace	−5.0	−7.1	0.5
C	Basketball Drill	−2.5	−3.6	−2.1
(832 × 480)	BQMall	−4.3	−5.5	−2.7
	PartyScene	−10.7	−11.9	−10.5
	RaceHorses	−1.2	−2.2	−0.1
D	Basketball Pass	−1.8	−2.7	−1.0
(416 × 240)	BQSquare	−21.6	−21.7	−18.0
	BlowingBubbles	−7.5	−9.1	−7.6
	RaceHorses	−1.8	−3.4	−2.3
E	FourPeople	n/a	−2.0	0.6
(1280 × 720)	Johnny	n/a	−6.8	−2.2
	KristenAndSara	n/a	−3.9	−0.5
Average		−4.0	−4.9	−2.6

and the results are compared. This experiment is conducted to see how much gain collectively all the improvements the HEVC interpolation filter brings. The test conditions can be summarized as:

- Four quantization values used: 22, 27, 32 and 37
- A total number of 24 different sequences are coded. These sequences are divided into different classes that represent different use-cases and video characteristics.
- Tests are conducted for three different prediction structures: Random Access, Low Delay with B pictures and Low Delay with P pictures.
- The coding efficiency is measured by using the Bjøntegaard-Delta bit rate measure [2].

The detailed results are shown in Table 5.6 for the luma component, where it is shown that on average, the interpolation filter of HEVC brings 4.0 % coding efficiency gain. The results for the chroma component is also summarized in Table 5.7 where the average coding efficiency gains reach 11.27 %. For some sequences, especially for those that contain more high frequency content, gains

Table 5.7 Average bit rate savings of HEVC interpolation filter for the chroma component

	BD-rate [%]					
	RA-Main		LB-Main		LD-Main	
Class	Cb	Cr	Cb	Cr	Cb	Cr
A (2560×1600)	−10.8	−11.3	n/a	n/a	n/a	n/a
B (1920×1080)	−10.8	−12.2	−14.7	−16.8	−5.8	−6.5
C (832×480)	−10.3	−10.8	−13.0	−13.5	−8.9	−9.6
D (416×240)	−15.3	−17.1	−20.4	−21.7	−16.6	−18.1
E (1280×720)	−2.4	−2.5	−5.4	−4.7	−2.5	−2.8
Average	**−11.7**	**−12.8**	**−13.3**	**−14.3**	**−7.4**	**−8.1**

become very large and become more than 20 %. Further experiments show that close to half of this gain is due to high precision filter operations as described in Sect. 5.3.1.2 and the rest of the gain is due to improved filter coefficients with longer tap-lengths.

5.4 Weighted Sample Prediction

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. In principle, WP replaces the inter prediction signal P by a linearly weighted prediction signal $\hat{P} = w \times P + o$, where w is an Illumination Compensation weight and o is an offset. 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 w_0 and w_1, and offsets o_0 and o_1 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. The WP process, for the case when only the list L0 is used, can be written in a simplified form as:

$$\hat{P}[x][y] = \text{Clip3}(0, \text{max\_val}, P_{L0}[x][y] * w_0 + o_0) \quad (5.9)$$

where x and y denote spatial coordinates within the prediction block and max_val represents the maximum value in the considered bit depth. Additionally, a log weight denominator (LWD) rounding factor may be used before adding the offset. For the case of bi-prediction, the value to be clipped is calculated as follows (similar rounding offset is used for uni-prediction case as well):

$$(P_{L0}[x][y] * w_0 + P_{L1}[x][y] * w_1 + (o_0 + o_1 + 1) \ll \text{LWD}) \gg (\text{LWD} + 1) \quad (5.10)$$

In H.264/AVC, weight and offset parameters are either derived by relative distances between the current picture and the reference distances (implicit mode) or weight and offset parameters are explicitly signaled (explicit mode) [6]. Unlike H.264/AVC, HEVC only includes explicit mode as the coding efficiency provided by deriving the weighted prediction parameters with implicit mode was considered negligible.

It should be noted that the weighted prediction process defined in HEVC version 1 was found to be not optimal for higher bit-depths as the offset parameter is calculated at low precision. The next version of the standard will likely modify the derivation of the offset parameter for higher bit-depths [24].

The determination of appropriate WP parameters in an encoder is outside the scope of the HEVC standard. Several algorithms for estimating WP parameters have been proposed in literature. Optimal solutions are obtained when the Illumination Compensation weights, motion estimation and Rate Distortion Optimization (RDO) are considered jointly. However, practical systems usually employ simplified techniques, such as determining approximate weights by considering picture-to-picture mean variation.

5.5 Summary and Conclusions

The inter-picture prediction part of the HEVC video coding standard is not introducing a revolutionary whole new design. Moreover, it can be seen as a steady improvement and generalization of all parts known from previous video coding standards, e.g. H.264/AVC. The motion vector prediction was enhanced with advanced motion vector prediction based on motion vector competition. An inter-prediction block merging technique significantly simplified the block-wise motion data signaling by inferring all motion data from already decoded blocks. When it comes to interpolation of fractional reference picture samples, high precision interpolation filter kernels with extended support, i.e. 7/8-tap filter kernels for luma and 4-tap filter kernels for chroma, improve the filtering especially in the high frequency range. Finally, the weighted prediction signaling was simplified by either applying explicitly signaled weights for each motion compensated prediction or just averaging two motion compensated predictions.

References

1. Bici O, Lainema J, Ugur K (2012) CE9: Results of SP experiments on simplification of merge process, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H0252, San Jose, Feb. 2012
2. Bjøntegaard G (2001) Calculation of average PSNR differences between RD curves, ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG-M33, Austin, Apr. 2001
3. Bossen F (2011) HM 5 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1200, Geneva, Nov. 2011
4. Bossen F (2012a) HM 6 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-H1100, San Jose, Feb. 2012
5. Bossen F (2012b) HM 8 common test conditions and software reference configurations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-J1100, Stockholm, July 2012
6. 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
7. Bross B, Jung J, Chien WJ, Kim IK, Zhou M (2011) CE9: Summary report of core experiment on MV coding and skip/merge operations, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G039, Geneva, Nov. 2011
8. 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
9. Bross B, Oudin S, Helle P, Marpe D, Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. In: *Proc. SPIE.* 8499, Applications of Digital Image Processing XXXV, no. 84990R, Oct. 2012
10. De Forni R, Taubman D (2005) On the benefits of leaf merging in quad-tree motion models. In: IEEE international conference on image processing, pp II–858, IEEE, 2005
11. 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
12. Helle P, Oudin S, Bross B,Marpe D, Bici M, Ugur K, Jung J, Clare G,Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. IEEE Trans Circuits Syst Video Technol 22(12):1720–1731
13. JCT-VC (2014) Subversion repository for the HEVC test model reference software. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware
14. Jung J, Bross B (2011) CE9: Summary report for CE9 on motion vector coding, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-D149, Daegu, Jan. 2011
15. Jung J, Onno P, Huang YW (2011) CE1: Summary report of core experiment 1 on motion data storage reduction, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F021, Torino, July 2011
16. Kamp S, Ballé J, Wien M (2009) Multihypothesis prediction using decoder side-motion vector derivation in inter-frame video coding. In: *Proc. SPIE.* 7257, Visual Communications and Image Processing 2009, no. 725704, Jan. 2009
17. 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
18. 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

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 (2011) Parsing robustness in high efficiency video coding-analysis and improvement. In: IEEE visual communications and image processing (VCIP), pp 1–4, 2011
21. 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
22. Mathew R, Taubman DS (2010) Quad-tree motion modeling with leaf merging. IEEE Trans Circuits Syst Video Technol 20(10):1331–1345
23. 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
24. 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
25. Sugio T, Nishi T (2011) Parsing robustness for merge/AMVP, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-F470, Torino, July 2011
26. 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
27. Wiegand T, Sullivan GJ, Bjøntegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576
28. 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