Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11
6th Meeting: Geneva, CH, 25 Oct. – 1 Nov. 2013

Document: JCT3V-F1005

Title:	**Test Model 6 of 3D-HEVC and MV-HEVC**
Status:	Output Document of JCT-3V
Purpose:	Test Model Description of 3D-HEVC and MV-HEVC

Author(s) or Contact(s):

Li Zhang, Qualcomm Incorporated — Email: lizhang@qti.qualcomm.com
Gerhard Tech, Fraunhofer HHI — Email: gerhard.tech@hhi.fraunhofer.de
Krzysztof Wegner, Poznan University of Technology — Email: kwegner@multimedia.edu.pl
Sehoon Yea, LG Electronics — Email: sehoon.yea@lge.com

Source: Editor

ABSTRACT

- (LZ-C00): Review and editorial improvements.
- (3DN-37/JCT3V-F0122) HTM support of depth coding in MV-HEVC: software adoption
- (3DN-36/JCT3V-F0138) 3D-HEVC HLS: Inter-view Updating Mechanism for Coding of DLT
- (3DN-35/JCT3V-F0131/JCT3V-F0139) Depth Lookup Table Coding for 3D-HEVC
- (3DN-34/JCT3V-F0125) CE3: Inter-view motion vector prediction for depth coding
- (3DN-33/JCT3V-F0110) 3D-CE3: Sub-PU level inter-view motion prediction
- (3DN-32/JCT3V-F0123/JCT3V-F0108) CE4: Results on improved advanced residual prediction
- (3DN-31/JCT3V-F0129) CE3 related: combined bi-predictive merging candidates for 3D-HEVC
- (3DN-30/JCT3V-F0093) CE3.h: Results on simple merge candidate list construction for 3DV.
- (3DN-23/JCT3V-F0151) HLS: Removal of IC in depth coding and IC flag signalling in 3D-HEVC; Decision: Remove IC for depth map coding, no change for texture coding.
- (3DN-18/JCT3V-F0105) CE4: ARP reference picture selection and its availability check
- (3DN-14/JCT3V-F0111) CE1: Simplified view synthesis prediction Decision
- (3DN-13/JCT3V-F0104) CE3: Removal of redundancy on VSP, ARP and IC Decision
- (3DN-12/JCT3V-F0161) CE4: Coding of weighting factor of advanced residual prediction
- (3DN-10/JCT3V-F0150) CE3: MPI candidate in depth merge mode list construction Decision: Adopt (option 1)
- (3DN-09/JCT3V-F0109,JCT3V-F0120) CE1: A simplified block partitioning method for view synthesis prediction
- (3DN-08/JCT3V-F0102) CE1: VSP partitioning for AMP Decision
- (3DN-07/JCT3V-F0115) CE2: Problem fix of the DV derivation in 3D-HEVC Decision: Adopt the suggested solution which aligns the text with the software bug fix.
- (3DN-06/JCT3V-F0149) CE5: Simplified depth inter mode coding; Adoption (BF): Align the text with software as suggested in F0149
- (3DN-05/JCT3V-F0147) CE5: DMM simplification and signalling Decision: Adopt (remove DMM3 and RBC).
- (3DN-04/JCT3V-F0159) CE5: Fast depth lookup table application method to intra modes for depth data (method 3)
- (3DN-03/JCT3V-F0132) CE5: Unification of delta DC coding for depth intra modes.
- (3DN-01/JCT3V-F0171) CE5: Fix for DMM/RBC reference sample filtering.

Draft 5 of 3D-HEVC Test Model Description

Ed. Notes (TM5) (changes compare to JCT3V-D1005):
- (LZ-C02) Corrections, editorial improvements
- (GT-C01) Corrections, editorial improvements
- (LZ-C01) Corrections, editorial improvements
- (3DN-21/JCT3V-E0126) CE3: Merge candidates derivation from vector shifting.
- (3DN-20/JCT3V-E0142, JCT3V-E0190) CE2: Simplified NBDV and improved disparity vector derivation
- (3DN-19/JCT3V-E0207) + JCT3V-E0208 CE1: Adaptive block partitioning for VSP and clipping
- (3DN-18/JCT3V-E0141) CE2: Clipping in depth-based disparity vector derivation

- (3DN-17/JCT3V-E0156) CE6: Simplified Inter Mode Coding of Depth Decision
- (3DN-11/JCT3V-E0182) CE3: A bug-fix for the texture merging candidate
- (3DN-10/JCT3V-E0172/Item 7) CE2: DVMCP Fix
- (3DN-09/JCT3V-E0170) CE3: Motion data buffer reduction for 3D-HEVC (first scheme)
- (3DN-08/JCT3V-E0117) CE6: Simplified DC calculation for SDC
- (3DN-06/JCT3V-E0204) CE5: Simplified Binarization for depth_intra_mode
- (3DN-05/JCT3V-E0159) CE5: Removal of Overlap between DMM1 and DMM3
- (3DN-04/JCT3V-E0158) CE6: Removal of DC from SDC Mode
- (3DN-03/JCT3V-E0146) CE5: DMM simplification and signalling. Remove DMM2.
- (3DN-01/JCT3V-E0046) CE4-related: Resampling in IC parameter derivation and 4x4 Chroma removal

Ed. Notes (TM4) (changes compare to JCT3V-C1005):

- ----------- Release v1 -----------
- (GT-C01) Corrections, editorial improvements
- (VSP) Incorporated missing draft text from last meeting, including JCT3V-D0105and 3DN-04/JCT3V-D0092
- (Removal of parts that are not in SW)
- (3DN-18/JCT3V-D0032/JCT3V-D0141/JCT3V-D0034) SDC Residual CABAC contexts.
- (3DN-17/JCT3V-D0035) DLT for DMM deltaDC coding
- (3DN-16/JCT3V-D0195) Unification of new intra modes in 3D-HEVC
- (3DN-14/JCT3V-D0183) Simplified DC predictor for depth intra modes
- (3DN-13/JCT3V-D0110) Sample-based simplified depth coding.
- (3DN-12/JCT3V-D0060) Removal of parsing dependency for illumination compensation
- (3DN-11/JCT3V-D0122) AMVP candidate list construction
- (3DN-09/JCT3V-D0177) Advanced residual prediction for multiview coding
- (3DN-08/JCT3V-D0138) Simplified DV derivation for DoNBDV and BVSP
- (3DN-07/JCT3V-D0112) Default disparity vector derivation
- (3DN-05/JCT3V-D0191) Clean-ups for BVSP in 3D-HEVC.
- (3DN-03/JCT3V-D0181) CE2.h related: CU-based Disparity Vector Derivation
- (3DN-01/JCT3V-D0156) HLS for stereo compatibility.

Ed. Notes (TM3) (changes compare to JCT3V-B1005):

- (3DC-GT2) Editorial improvements, small corrections.
- ----------- Release d0 -----------
- Split of Test Model text and specification text
- (3DE-05) Alignment with MV-HEVC draft 3.
- (3DE-01) Reordered sub-clauses related to disparity estimation and additional motion candidates.
- (3DN-20) Alignment of JCT3V-C0152 + JCT3V-C0137.
- (3DN-07/JCT3V-C0137) Texture motion vector candidate for depth.
- (3DN-07/JCT3V-C0137) Removal of MPI.
- (3DN-19) Camera parameters
- (3Dn−03) Wedgelet pattern generation process.
- (3Dn-01) Incorporated missing intra-predicted wedgelet partition mode
- (3DN-08/JCT3V-C0138) Removal of parsing dependency for inter-view residual prediction.
- (3DN-18/JCT3V-C0160) QTL disabled for RAP.
- (3DN-17/JCT3V-C0154) Reference sub-sampling for SDC and DMM.
- (3Dc-03) Fix SDC
- (3DN-16/JCT3V-C0096) Removal of DMM 2 from SDC.
- (3DN-15/JCT3V-C0034) Delta DC processing for DMMs.
- (3DN-14/JCT3V-C0044) Signalling of wedgeIdx for DMM3.
- (3DN-02/JCT3V-C0152) View synthesis prediction (without disparity derivation part).
- (3DN-03/JCT3V-C0112) Restricted search of max disparity.
- (3DN-01,02/JCT3V-C0131,JCT3V-C0152) Disparity derivation from depth maps.
- (3Dc-02) Incorporated missing conditions of long/short-term pictures in AMVP (related to JCT3V-B0046).
- (3DN-13/JCT3V-C0116) Inter-view vector scaling for AMVP.
- (3DE-03) Incorporated derivation process for AMVP from base spec.
- (3DN-12/JCT3V-C0115) Signalling of inter-view motion vector scaling.
- (3DE-02) Incorporated TMVP text from base spec.
- (3DN-09/JCT3V-C0047) Alternative reference index for TMVP.

3D-HEVC

- (3DN-10/JCT3V-C0051) Unification of inter-view candidate derivation.
- (3DE-01) Revised text related to residual prediction.
- (3DN-06/JCT3V-C0129) Vertical component in residual prediction.
- (3DN-05/JCT3V-C0097/JCT3V-C0141) Temporal blocks first in DV derivation.
- (3DC-GT1) Editorial improvements, small corrections.
- (3DN-04/JCT3V-C0135) Restriction on the temporal blocks for memory bandwidth reduction in DV derivation.
- (3Dn-02) Full sample MV accuracy for depth.
- (3DN-11/JCT3V-C0046) Extension of illumination compensation to depth.
- (3Dc-01) Fix Illumination compensation (including ic_flag for skip).

Ed. Notes (TM2) (changes compared to JCT3V-A1005)

- Accepted changes and marked delta to base spec
- (3DC-GT2) Editorial improvements, small corrections
- (3DC-CY) Editorial improvements, small corrections
- (MVS-02/JCT3V-B0046) Treatment of inter-view pictures as long term- reference pictures
- (3DE-11) Revised text related to 3Dn-01
- (3Dn-01/m23639) Results on motion parameter prediction
- (3DE-12) Revised text related residual prediction
- (3DE-10) Revised text Related to Illumination compensation.
- (3DN-01/JCT3V-B0045) Illumination compensation for inter-view prediction.
- (3Dn-02/m24766) Restricted Inter-View Residual Prediction
- (3DE-09) Revised text related to depth intra: Edge Intra
- (3DE-09) Revised text related to depth intra: SDC
- (3DE-09) Revised text related to depth intra: DMMs
- (3DO-01/JCT3V-B0131) Depth distortion metric with a weighted depth fidelity term
- (3DN-12/JCT3V-B0036) Simplified Depth Coding with an optional Depth LUT
- (3DN-13/JCT3V-B0039) Simplified Wedgelet search for DMM modes 1 and 3
- (3DN-03/JCT3V-B0083) Unconstrained motion parameter inheritance
- (3DE-08) Incorporated context tables for SDC
- (3DE-07) Improved MPI text.
- (3DN-02/JCT3V-B0068) Incorporated Depth Quadtree Prediction.
- (3DE-06) Incorporated parsing process, including tables for DMMs.
- (3DE-05) Added missing initialization of invalid motion/disparity parameters
- (3DC-03) Added missing pruning of collocated merge candidate due to number of total candidates.
- (3DE-04) Moved pruning of spatial merge candidate B2 due to number of total candidates.
- (3DE-03) Moved derivation of disparity one level higher in process hierarchy.
- (3DE-02) Inserted "Derivation process for motion vector components and reference indices" from base spec
- (3DC-02) Fixed storage of IvpMvFlagLX and IvpMvDisp.
- (3DN-09-10-11/JCT3V-B0048,B0069,B0086) Modification inter-view merge candidates
- (3DC-01) Fixed derivation of inter-view merge candidates.
- (3DE-01) Revised derivation of disparity from temporal candidates
- (3DN-04/JCT3V-B0047) Improvements for disparity vector derivation)
- (3DN-08/JCT3V-B0136) Support of parallel merge in disparity vector derivation
- (3DN-05/JCT3V-B0135) Modified disparity vector derivation process for memory reduction
- (3DN-04/JCT3V-B0111) Decoupling inter-view candidate for AMVP
- (3DN-07/JCT3V-B0096) Removal of dependency between multiple PUs in a CU for DV-derivation
- (3DC-GT) Small corrections, editorial improvements

Ed. Notes (TM1) (changes compare to N12744)

- (3D08/JCT3V-A0126) (T,N) Simplified disparity derivation
- (3D16) Moved 3D-tool related flags from SPS to VPS, removal camera parameters
- (3D09/JCT3V-A0049) (N) Inter-view motion prediction modification
- (3D13/JCT3V-A0119) (T) VSO depth fidelity
- (3D07/JCT3V-A0070) (T,N) Region boundary chain coding for depth maps
- (3D06/JCT3V-A0087) (T) RDO selection between Non-Zero Residual and All-Zero Residual Intra
- (3D12) (T) Depth Quadtree Prediction
- (3D15) (N) Fix references

- (3D11) (T,N) Improvement of text of already adopted tools
- (3D10/JCT3V-A0097) (T;N) Disparity vector generation
- (3D02) (N) Removed MV-Part and update to Annex F
- (3D03) (T) Labelling of tools not in CTC/Software. Removal?
- (3D05/JCT3V-A0093) (T) VSO early skip
- (3D04/JCT3V-A0033) (T) VSO model based estimation
- (3D14) (N) Update of low level specification to match HEVC text specification 8(d7)
- (3D01) (N): Removed HEVC text specification

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Data Format and System Description

3D video is represented using the Multiview Video plus Depth (MVD) format, in which a small number of captured views as well as associated depth maps are coded and the resulting bitstream packets are multiplexed into a 3D video bitstream. After decoding the video and depth data, additional intermediate views suitable for displaying the 3D content on an auto-stereoscopic display can be synthesized using depth-image-based rendering (DIBR) techniques. For the purpose of view synthesis, camera parameters are additionally included in the bitstream. The bitstream packets include header information, which signal, in connection with transmitted parameter sets, a view identifier and an indication whether the packet contains video or depth data. Sub-bitstreams containing only some of the coded components can be extracted by discarding bitstream packets that contain non-required data. One of the views, which is also referred to as the base view or the independent view, is coded independently of the other views and the depth data using a conventional HEVC video coder. The sub-bitstream containing the independent view can be decoded by an unmodified HEVC video decoder and displayed on a conventional 2D display. Optionally, the encoder can be configured in a way that a sub-bitstream representing two views without depth data can be extracted and independently decoded for displaying the 3D video on a conventional stereo display. The codec can also be used for coding multiview video signals without depth data. In that case alternative methods such as Image Domain Warping (IDW) may be used to generate a multiview signal. And, when using depth data, it can be configured in a way that the video pictures can be decoded independently of the depth data.

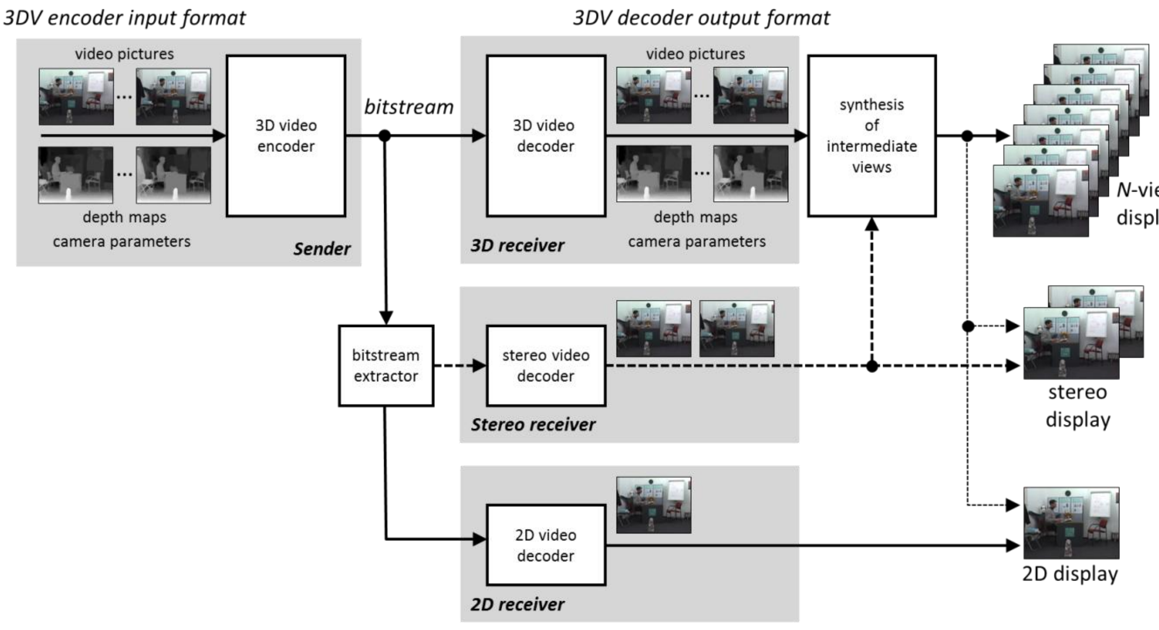


Figure 1: Overview of the system structure and the data format for the transmission of 3D video.

The basic concept of the system and data format is illustrated in Figure 1. In general the input signal for the encoder consists of multiple views, associated depth maps, and corresponding camera parameters. However, as described above, the codec can also be operated without depth data. The input component signals are coded using a 3D video encoder, which represents an extension of HEVC. At this, the base view is coded using an unmodified HEVC encoder. The 3D video encoder generates a bitstream, which represents the input videos and depth data in a coded format. If the bitstream is decoded using a 3D video decoder, the input videos, the associated depth data, and camera parameters are reconstructed with the given fidelity. For displaying the 3D video on an auto stereoscopic display, additional intermediate views are generated by a DIBR algorithm using the reconstructed views and depth data. If the 3D video decoder is connected to a conventional stereo display instead of to an auto stereoscopic display, the view synthesizer can also generate a pair of stereo views, in case such a pair is not actually present in the bitstream. At this, it is possible to adjust the rendered stereo views to the stereo geometry of the viewing conditions. One of the decoded views or an intermediate view at an arbitrary virtual camera position can also be used for displaying a single view on a conventional 2D display.

The 3D video bitstream is constructed in a way that the sub-bitstream representing the coded representation of the base view can be extracted by simple means. The bitstream packets representing the base view can be identified by inspecting transmitted parameter sets and the packet headers. The sub-bitstream for the base view can be extracted by discarding all packets that contain depth data or data for the dependent views and, then, the extracted sub-bitstream can be directly decoded with an unmodified HEVC decoder and displayed on a conventional 2D video display.

The encoder can also be configured in a way that the sub-bitstream containing only two stereo views can be extracted and directly decoded using a stereo decoder. The encoder can also be configured in a way that the views can be generally decoded independently of the depth data. It is also possible to synthesize intermediate view using only the stereo sequences as input of the view synthesis.

A detailed description of the coding scheme is given in sec. 2. Depth-image-based rendering algorithms are described in sec. 3.

2. Coding Algorithm

In the following, the coding algorithm based on the MVD format, in which each video picture is associated with a depth map, is described. The coding algorithm can also be used for a multiview format without depth maps. The video pictures and, when present, the depth maps are coded access unit by access unit, as it is illustrated in Figure 2. An *access unit* includes all video pictures and depth maps that correspond to the same time instant. Non-VCL NAL units containing camera parameters may be additionally associated with an access unit. It should be noted that the coding order of access units doesn't need to be identical to the capture or display order. In general, the reconstructed data of already coded access units can be used for an efficient coding of the current access unit. Random access is enabled by so-called *random access units* or *instantaneous decoding refresh* (IDR) access units, in which the video pictures and depth maps are coded without referring to previously coded access units. Furthermore, an access unit doesn't reference any access unit that precedes the previous random access unit in coding order. If the picture of the base view in an access unit is an IDR picture, the access unit is an IDR access unit.

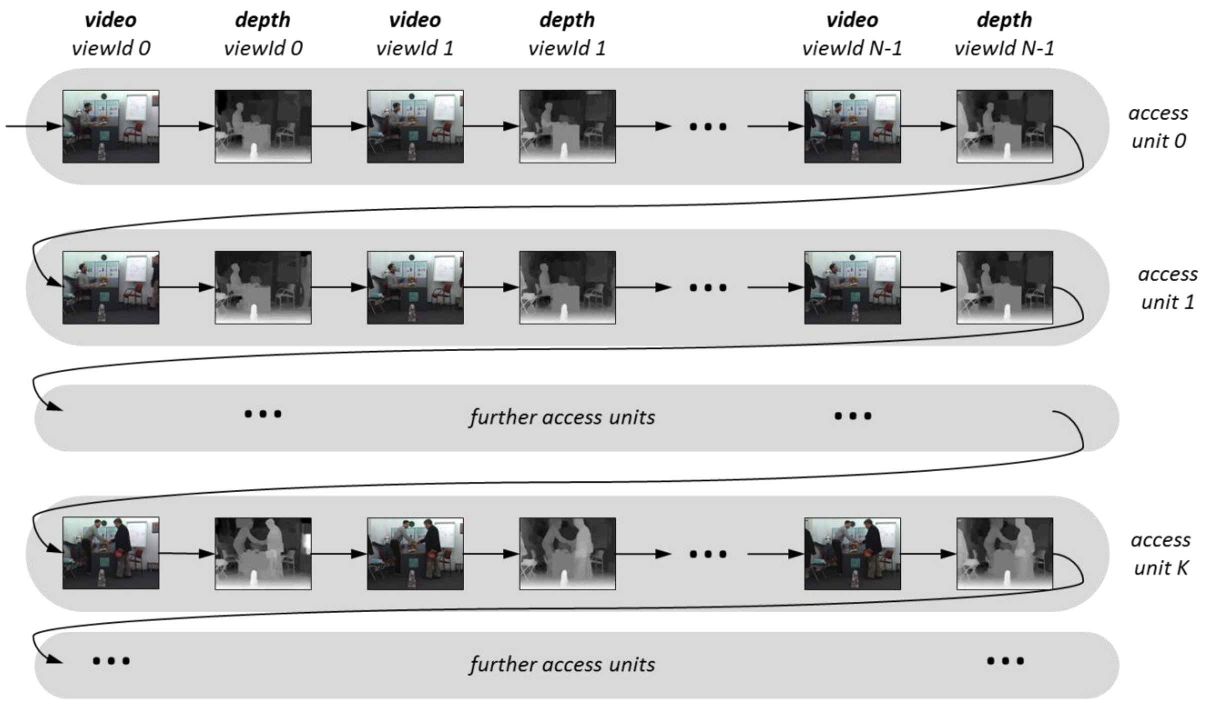


Figure 2: Access unit structure and coding order of view components.

The video pictures and depth maps corresponding to a particular camera position are indicated by a view order index (viewIdx). All video pictures and depth maps that belong to the same camera position are associated with the same value of viewIdx. The view order indices are used for specifying the coding order inside the access units and detecting missing views in error-prone environments. Inside an access unit, the video picture and, when present, the associated depth map with viewIdx equal to 0 are coded first, followed by the video picture and depth map with viewIdx equal to 1, etc. A video picture and depth map with a particular value of viewIdx are transmitted after all video pictures and depth maps with smaller values of viewIdx. For the independent view, the video picture is always coded before the associated depth map. For dependent views, the video picture maybe coded before or after the associated depth map (i.e., the depth map with the same value of viewIdx). It should be noted that the value of viewIdx doesn't necessarily represent the arrangement of the cameras in the camera array. For ordering the reconstructed video pictures and depth map after decoding, each value of viewIdx is associated with another identifier called view identifier (viewId). The view identifier is a signed integer values, which specifies the ordering of the coded views from left to right. If a view A has a smaller value of viewId than a view B, the camera for view A is located left to the camera of view B. In addition, camera parameters required for converting depth values into disparity vectors are included in the bitstream. For the considered linear setup, the corresponding conversion parameters consist of a scale factor and an offset. The vertical component of a disparity vector is always equal to 0. The horizontal component is derived according to

$$d_v = (s * v + o) >> n,$$

where v is the depth sample value, s is the transmitted scale factor, o is the transmitted offset, and n is a shift parameter that depends on the required accuracy of the disparity vectors.

All of the video and depth sequences are associated with a video parameter set. Each video sequence and depth sequence is associated with a separate sequence parameter set and a separate picture parameter set. The picture parameter set syntax, the NAL unit header syntax, and the slice header syntax for the coded slices haven't been modified for including a mechanism by which the content of a coded slice NAL units can be associated with a component signal.

The video parameter set provides the following information for all layers of the CVS:

- the view order index (indicates the coding order of a view);
- the depth flag (indicates whether video data or depth data are present);
- the view identifier (indicates the location of the view relative to other coded views);
- syntax elements specifying layer sets and output layers;
- syntax elements specifying inter-layer dependencies;
- syntax elements specifying which 3D-coding tools are enabled;
- syntax elements specifying the representation format of the view components;
- video usability information;

The video parameter sets furthermore includes a flag, which specifies whether the camera parameters are constant for a coded video sequence or whether they can change on a picture by picture basis. If this flag indicates that the camera parameters are constant for a coded video sequence, the camera parameters (i.e., the scale and offset values described above) are present in the video parameter set for each view. Otherwise, the camera parameters are not present in the video parameter set, but instead the camera parameters for the current view are coded in the slice headers.

For a view with view order index viewIdxB following camera parameters are present (either in the video parameter set or the slice header):

- for each view order index viewIdxA value smaller than viewIdxB, a scale and an offset specifying the conversion of a depth sample of the view with viewIdxA to a horizontal disparity between the view with viewIdxB and the view with viewIdxA;
- for each view order index viewIdxA value smaller than viewIdxB, a scale and an offset specifying the conversion of a depth sample of the view with viewIdxB to a horizontal disparity between the view with viewIdxA and the view with viewIdxB;

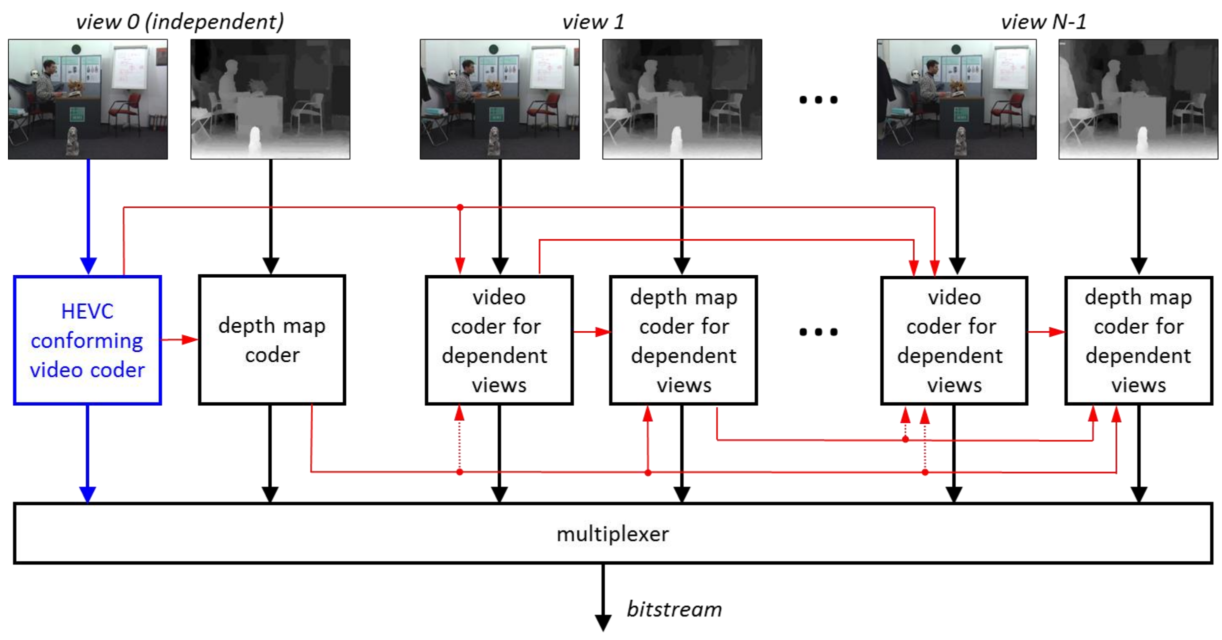

view 0 (independent)
view 1
view N-1
HEVC conforming video coder
depth map coder
video coder for dependent views
depth map coder for dependent views
video coder for dependent views
depth map coder for dependent views
multiplexer
bitstream

Figure 3: Basic codec structure with inter-component prediction (red arrows).

The basic structure of the 3D video codec is shown in the block diagram of Figure 3. In principle, each component signal is coded using an HEVC-based codec. The resulting bitstream packets, or more accurately, the resulting Network Abstraction Layer (NAL) units, are multiplexed to form the 3D video bitstream. The base or independent view is coded using an unmodified HEVC codec. Given the 3D video bitstream, the NAL units containing data for the base layer can be identified by parsing the parameter sets and NAL unit header of coded slice NAL units (up to the picture parameter set identifier). Based on these data, the sub-bitstream for the base view can be extracted and directly coded using a conventional HEVC decoder.

Current reference software 3D-HTM supports both multiview extension of HEVC (referred to as MV-HEVC) and 3D-HEVC coding. In 3D-HEVC, when coding the dependent views and the depth data, modified HEVC codecs are used, which are extended by including additional coding tools and inter-component prediction techniques that employ already coded data inside the same access unit as indicated by the red arrows in Figure 3. For enabling an optional discarding of depth data from the bitstream, e.g., for supporting the decoding of a stereo video suitable for conventional stereo displays, the inter-component prediction can be configured in a way that video pictures can be decoded independently of the depth data. A detailed description of the added coding tools is given in the following subsections. In MV-HEVC, when coding the dependent texture views, the HEVC codec is modified by including some high level syntax changes and the inter-component prediction techniques, similar to the inter-view prediction in the MVC extension of H.264/AVC. In addition, depth coding is also supported in MV-HEVC. Different from coding dependent texture views wherein inter-component prediction could be applied, when coding dependent depth views, it is handled in the same way as depth base views, that is, HEVC compatible.

2.1 Coding of the Independent View

The independent view, which is also referred to as the base view, is coded using an unmodified HEVC codec.

2.2 Coding of Dependent Views

For the dependent views, the same concepts and coding tools are used as for the independent view. However, additional tools have been integrated into the HEVC codec, which employ already coded data in other views for efficiently representing a dependent view. The additionally integrated tools are described in the following.

2.2.1 Disparity-compensated prediction

As a first coding tool for the dependent views, the well-known concept of disparity-compensated prediction (DCP), which is also used in MVC, has been added as an alternative to motion-compensated prediction (MCP). At this, MCP refers to an inter-picture prediction that uses already coded pictures of the same view, while DCP refers to an inter-picture prediction that uses already coded pictures of other views in the same access unit, as it is illustrated in Figure 4.

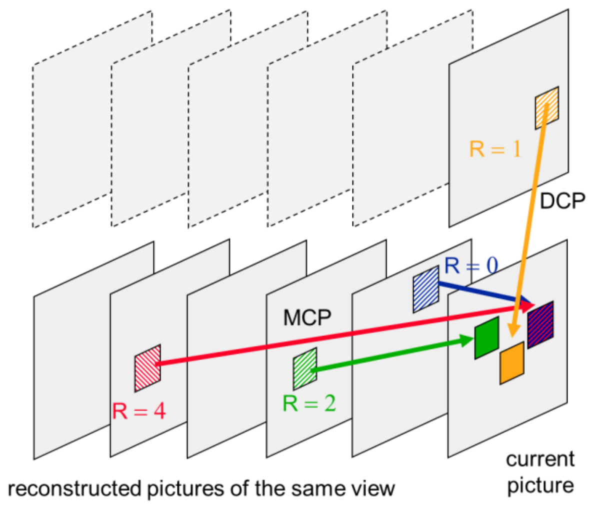


Figure 4: Disparity-compensated prediction as an alternative to motion-compensated prediction.

The macroblock syntax and decoding process haven't been changed for adding DCP, only the high-level syntax has been modified so that already coded video pictures of the same access unit can be inserted into the reference pictures lists (as described in 2.2.1.1). As illustrated in Figure 4, the transmitted reference picture index (R in the figure) signals whether an inter-coded blocks is predicted by MCP or DCP. The motion vector prediction is modified in a way that the motion vectors of motion-compensated blocks are predicted by only using the neighbouring blocks that also use temporal reference pictures, while the disparity vectors of disparity-compensated blocks are predicted by only using the neighbouring blocks that also use inter-view reference pictures.

2.2.1.1 Reference list construction and modification

For motion-/disparity-compensated prediction, the inter-view reference pictures as well as the temporal inter reference pictures are included in the reference lists (L0, L1). The reference lists are constructed as specified in the following steps.

1. Update of decoded picture buffer (DPB)
   - The inter-view reference pictures, which are already coded pictures in the same access unit, are added to the current DPB.
   - The status of temporal/inter reference pictures present in DPB in a view is signalled by the Reference Picture Set (RPS).
2. A list (RefPicLayerId) containing the layerId values of the active inter-view reference pictures is constructed based on information in the video parameter set and the slice segment header of the current picture.
3. Construction of reference picture sets
   - Reference picture sets including the temporal reference pictures (RefPicSetStCurrBefore, RefPicSetStCurrAfter, RefPicSetLtCurr) are constructed in the same manner as in HEVC.
   - Two reference picture sets including the inter-view reference pictures (RefPicInterLayer0 and RefPicInterLayer1) identified by the list RefPicLayerId are constructed based on ViewId values in a manner that pictures with ViewId values less than and greater than the ViewId value of the current picture are in different sets.
4. Construction of reference picture lists (L0, L1)
   - Reference picture list L0 is constructed by adding reference picture of the sets in order: RefPicSetStCurrAfter, RefPicInterLayer0, RefPicSetStCurrAfter, RefPicSetLtCurr, RefPicInterLayer1
   - Reference picture list L1 is constructed by adding reference picture of the sets in order: RefPicSetStCurrBefore, RefPicInterLayer1, RefPicSetStCurrBefore, RefPicSetLtCurr, RefPicInterLayer0
5. Modification of reference picture lists (L0, L1)
   - The constructed reference lists are modified based on reference picture list modification syntax table as defined in HEVC.

2.2.2 Derivation of Disparity Vectors

To utilize the coded information from another view, a disparity vector is required to locate a corresponding block of the current PU/CU in an already coded picture of the same time instance. For the current 3D-HTM, the codec provides a method to derive a disparity vector independently from coded depth maps. The disparity is derived from spatial and temporal neighbouring blocks which are using inter-view prediction or from motion vectors which are obtained by inter-view prediction. This disparity vector is then utilized to identify a depth block in an already coded depth view to perform backward warping for further improving the accuracy of derived disparity vectors. The derived disparity vector could be used in inter-view motion prediction, advanced residual prediction, illumination compensation and view synthesis prediction, as described in the following sub-sections.

2.2.2.1 Disparity vector from neighbouring blocks (NBDV)

For each CU a disparity vector is derived from a motion vector of a spatial or temporal DCP neighbouring block of the CU or from a disparity vector associated with an MCP neighbouring block of the CU. Once a disparity motion vector is found, the whole disparity vector derivation process terminates.

First temporal DCP neighbouring blocks are evaluated as specified in section 2.2.2.1.2, followed by a check of the spatial DCP neighbours, as specified in section 2.2.2.1.1. Finally, MCP coded neighbour blocks are searched as described in section 2.2.2.1.3.

When no disparity motion vector is found from the neighbouring blocks, a zero disparity vector pointing to the inter-view reference picture with the smallest view index is used. In this case, the advanced residual prediction process is disabled at the encoder side.

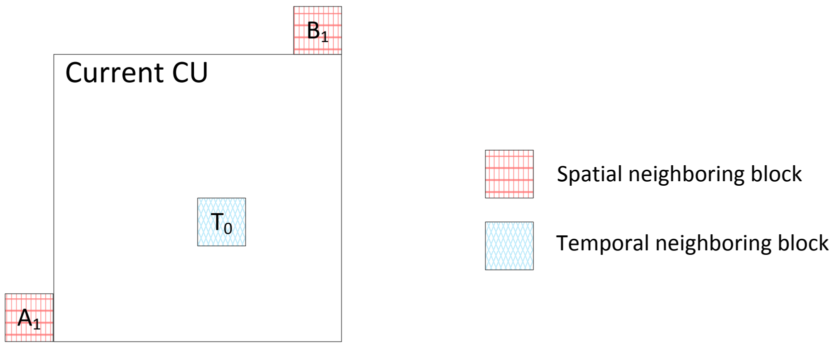


Figure 5: Location of spatial and temporal neighbour blocks

2.2.2.1.1 **Spatial Neighbouring Blocks**

Two spatial neighbouring blocks are used for the disparity vector derivation. They are: the left and above blocks of current prediction unit (PU), denoted by A_1, B_1 as defined in Figure 5.

A constraint on the search is to to regard only the blocks that are also utilized in the merge scheme in the HEVC base specification, when the DV is derived for the derivation of an inter-view merge candidate.

The checking order of the two spatial neighbouring blocks is: A_1 and B_1.

2.2.2.1.2 **Temporal Neighbouring Blocks**

Up to two reference pictures from current view are treated as candidate pictures for temporal neighbours. The first candidate picture is the co-located picture as used for Temporal Motion Vector Prediction (TMVP) in HEVC without low delay check. The co-located picture is indicated in a slice header. The second picture is derived in the reference picture lists with the ascending order of reference picture indices, and added into the candidate list, given as follows:

1) A random access point (RAP) is searched in the reference picture lists. If found, the RAP is placed into the candidate list for the second picture and the derivation process is completed. In a case that the RAP is not available for the current picture, go to step (2).
2) A picture with the lowest temporalID (TID) is searched out and placed into the candidate list of the temporal pictures as the second entry.
3) If multiple pictures with the same lowest TID exist, a picture of less POC difference with the current picture is chosen.

As shown in the above description, the second temporal candidate picture is chosen in a way that disparity motion vectors can have more chance to be present in the picture. The derivation process of the second candidate picture can be done in the slice level and be invoked only once per slice.

For each candidate picture up to one temporal neighbouring block, denoted by T_0 as depicted in Figure 5, is searched.

2.2.2.1.3 **Disparity derivation from MCP coded neighbour blocks**

In addition to the DCP coded blocks, blocks coded by motion compensated prediction (MCP) are also used for the disparity derivation process. When a neighbour block is MCP coded block and its motion is predicted by the inter-view motion prediction, as shown in Figure 6, the disparity vector used for the inter-view motion prediction represents a motion correspondence between the current and the inter-view reference picture. This type of motion vector is referred to as inter-view predicted motion vector (IvpMv) and the blocks are referred to as DV-MCP blocks in the sequel. The motion correspondence is used for the disparity derivation process as explained in the following.

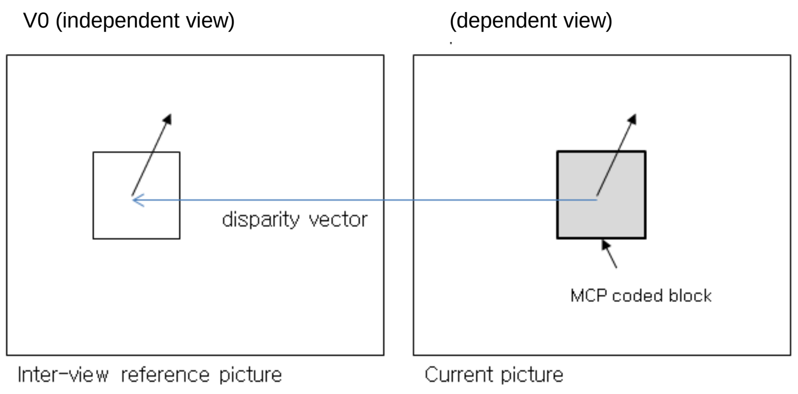


Figure 6: The inter-view predicted motion vector of a MCP coded block.

To indicate whether a blocks is DV-MCP block or not and to store the disparity vector used for the inter-view motion prediction, three variables are used:

- ivpMvFlag
- ivpMvDisparityX
- refViewIdx.

The block whose motion information (motion vectors and reference indexes) is derived from a reference view is identified when the 0th motion parameter candidate of MERG/SKIP mode is selected. In that case, the ivpMvFlag is set to 1, ivpMvDisparityX and refViewIdx are set to the disparity vector and associated view order index used for the inter-view motion prediction, respectively.

The disparity vector is derived from SKIP coded DV-MCP blocks. When a block is coded by skip mode, neither mvd (motion vector difference) data nor residual data is signalled, which implies that the disparity vector used for SKIP coded DV-MCP block well describes the motion correspondence than the disparity vector used for DV-MCP blocks that are not SKIP coded.

If DCP coded block is not found in the spatial and temporal neighbour blocks, then disparity derivation process scans the spatial neighbour blocks for DV-MCP compensated in following order: A1 and B1. If a neighbour block is a SKIP coded DV-MCP block, then the values of both IvpMvDisparityX and RefViewIdx associated with the neighbour block are returned as the derived disparity and view order index. The vertical component of the disparity vector is set equal to zero.

To reduce the amount of memory required for the derivation of the disparity from DV-MCP blocks, the block B1 is only utilized when they are located in the current CTU. An example for this can be seen in Figure 7. Here only spatial neighbour block A1 is utilized.

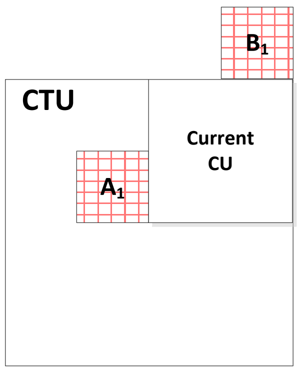


Figure 7: Example: For the derivation of the disparity from DV-MCP neighbouring blocks of the current CU, above block B1 is not used, since it is not located within the current CTU.

2.2.2.2 Depth oriented neighbouring block based disparity vector (DoNBDV)

While coding the texture of a dependent view, the decoded depth of the base view is already available. So the disparity derivation needed for the coding of the texture of the dependent might be improved by utilizing the depth map of the base view. A disparity vector (which might be a better estimate than a disparity vector derived with method 3) can be extracted by the following steps:

1. A disparity vector is the derived by NBDV.
2. The disparity vector is used to locate the corresponding block in the coded depth of the reference view with the same view order index associated with the disparity vector from NBDV. When the corresponding depth block is located outside of the depth picture or on the boundary of the depth picture, the samples located outside of the picture is clipped to the position on the boundary of the picture while those samples located within the picture are kept unchanged.
3. The depth in the corresponding block in the base depth is assumed to be the "virtual depth block" of the current block in the dependent view.
4. The maximum depth value of the four edge samples of the virtual depth block is retrieved.
5. The maximum depth value is converted to disparity

An example is depicted in Figure 8. The coded depth map in view 0 is denoted as Coded D0. The texture to be coded is T1. For the current block (CB) a depth block in the coded D0 is derived using disparity vector estimated by NBDV.

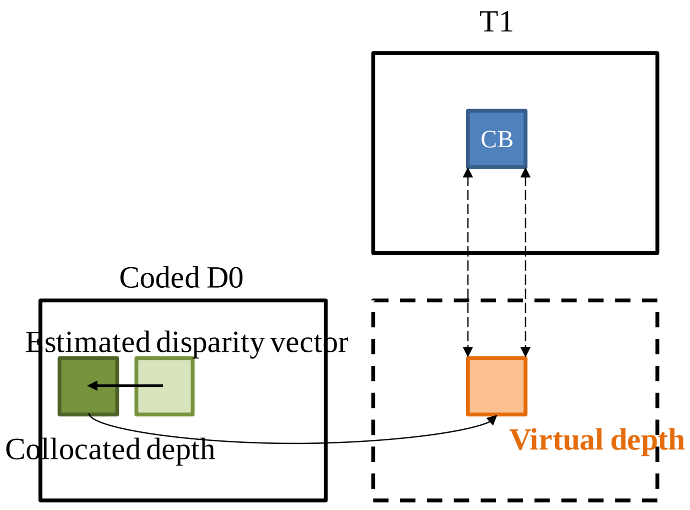


Figure 8: Retrieval of the virtual depth block.

To enable coding of texture without depth maps, disparity derivation from depth maps can be disabled by a VPS flag.

2.2.3 Modified merge candidate list construction process

The merge candidate list (MCL) construction in HEVC is not efficient for motion vector prediction of the dependent views in 3D-HEVC as the motion information correlation between views is not exploited. In more details, the motion information of two associated blocks in base view and dependent view is likely to be same, as these two views represent different projections of the same 3D scene captured by synchronous video cameras. The motion information of one of dependent views can be inferred from the previously coded views (e.g., base view), provided a disparity vector between these two associated blocks is known a priori. Therefore, in 3D-HEVC, additional possible candidates for the merge list are introduced by considering the temporal motion information of the already coded blocks in the reference view as well as the disparity motion information (which points to the corresponding inter-view block in reference view). Due to the availability of more possible candidates, the number of maximum candidates in the final merge list is increased to 6.

In this sub-section, the merge candidate list construction process for dependent texture views is introduced. And more details about the newly introduced merge candidates are described in the following sub-sections.

For a dependent texture view, the following processes are performed in order:

Step 1. Invoke the HEVC MCL construction process. Note, to avoid unpredictable decoding process, when invok ing the derivation process of combined bi-predictive merging candidates in HEVC, instead of just checking the slice type equal to B slice, another condition shall be also satisfied, that is, the number of available merging can didates inserted to the merge candidate list should be less than 5.

Step 2. Check view synthesis prediction (VSP) flags of spatial merging candidates (SMC) A1, B1, B0, A0 and B2 included in the merge candidate list. The five spatial neighbouring blocks are depicted in Figure 9.

Step 3. If the current PU is in the depth map, the texture merging candidate(T) which is derived from the corresponding texture is identified. If the current PU is in the texture, the disparity vector is derived and the corresponding block of the current PU based on the disparity vector is identified and either PU-level or sub-PU level inter-view motion prediction process is invoked to generate inter-view merging candidate, as described in 2.2.4.1.1. For T or IvMC, if these are different from the A1 and B1, and derived to be available, is inserted to the head of the merge candidate list.

Step 4. The inter-view disparity vector candidate (IvDC), as described in section 2.2.4.1.2, which is converted from the disparity vector is inserted to the back of the last valid candidate from IvMC to B0, if these are different from the A1 and B1.

Step 5. If the BVSP is enabled for the current slice, the BVSP merging candidate, as described in section 2.2.7, is inserted to the back of the last valid candidate from IvMC to IvDC.

Step 6. Generate a shifted candidate, as described in section 2.2.4.1.3. If the shifted candidate is an IvMC, it is compared with the IvMC in step 3. If it is not equal to the IvMC in step 3 or if the shifted candidate is a DSMV (when the additional IvMC is unavailable), the generated additional candidate is inserted to the back of the last valid candidate from IvMC to B2.

Step 7. If the MCL is not full, the zero motion vectors are filled as a default.

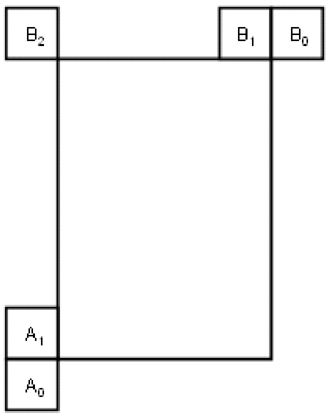


Figure 9: Spatial motion vector neighbours relative to the current prediction unit.

2.2.4 Inter-view motion prediction

The basic concept of the inter-view prediction of motion parameters is illustrated in **Błąd! Nie można odnaleźć źródła odwołania.**. For the following overview, it is assumed that an estimate of a sample-wise depth map for the current picture is given. Below, it is described how such an estimate can be derived. For deriving candidate motion parameters for a current block in a dependent view, the maximum depth value *d* within the associated depth block is converted to a disparity vector. By adding the disparity vector to the sample location ***x***, which is in the middle of the block, a reference sample location $\boldsymbol{x}_R$ is obtained. The prediction block in the already coded picture in the reference view that covers the sample location $\boldsymbol{x}_R$ is used as the reference block. If this reference block is coded using MCP, the associated motion parameters can be used as candidate motion parameters for the current block in the current view. The derived disparity vector can also be directly used as a candidate disparity vector for DCP.

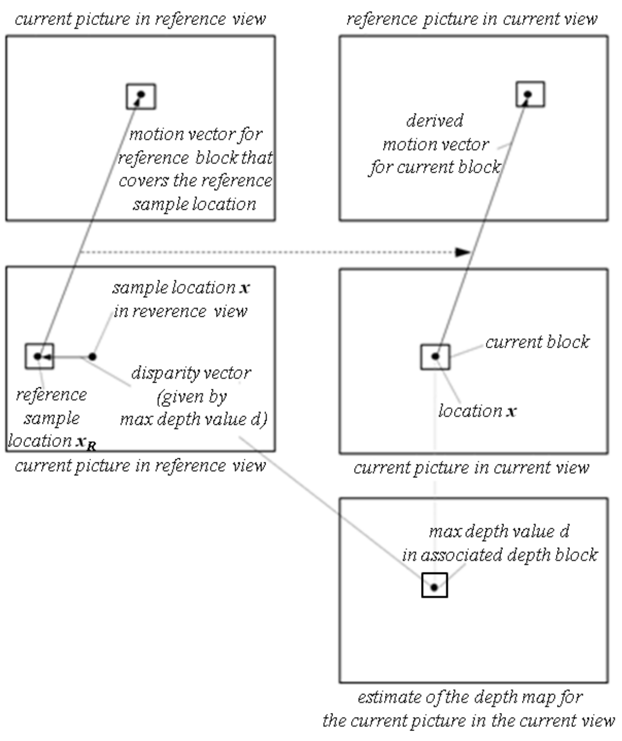

current picture in reference view
reference picture in current view
motion vector for reference block that covers the reference sample location
derived motion vector for current block
sample location x in reverence view
current block
disparity vector (given by max depth value d)
reference sample location x_R
location x
current picture in reference view
current picture in current view
max depth value d in associated depth block
estimate of the depth map for the current picture in the current view

Figure 10: Basic principle of deriving motion parameters for a block in a current picture based on motion parameters in an already coded reference view and an estimate of the depth map for the current picture.

2.2.4.1 Usage of Inter-View Motion Parameter Prediction

Inter-view motion vector prediction is applied in for the merge mode (and skip mode). In the merge mode of HEVC (as well as in the skip mode, which represents the merge mode without coding a residual signal), basically the same motion parameters (number of hypotheses, reference pictures, and motion vectors) as for a neighbouring block are used. If a block is coded in the merge mode, a candidate list of motion parameters is derived, which includes the motion parameters of spatially neighbouring blocks as well as motion parameters that are calculated based on the motion parameters of the co-located block in a temporal reference picture. The chosen motion parameters are signalled by transmitting an index into the candidate list.

The candidate list of motion parameters is extended by a motion parameter candidate for MCP (IvMC) that is obtained using inter-view motion prediction. Moreover a motion parameter candidate for DCP constructed from the derived disparity (IvDC) is added. The derivation of both additional candidates is described in the following.

2.2.4.1.1 **Derivation of inter-view merging candidate**

For the derivation of the candidate for MCP (i.e., inter-view merging candidate, IvMC), a corresponding block in a view component at the same time instant as the current view component is utilized.

Two methods (Sub-PU level, PU-level) related to IvMC derivation are applied to code dependent texture and depth views, respectively. For depth coding, the PU-level inter-view motion prediction method is applied to derive one IvMC for each PU based on the following rules:

One corresponding block of current PU is identified by the disparity vector and view order index for current PU. The corresponding block is determined by shifting the position of the current block using the disparity vector derived as described above.

If the corresponding block is coded using MCP it is tested for each motion hypothesis of the current block, whether an motion vector and a reference index can be derived for the IvMC candidate. This is the case if a picture is included in the reference picture list belonging to the current slice and motion hypothesis with a picture order count equal to the picture order count of a reference picture of the corresponding block. When such a picture is found the reference index of this picture in the reference picture list belonging to the current slice and motion hypothesis and the motion vector of the hypothesis of the corresponding block are used to derive the IvMC candidate.

For texture coding, the sub-PU inter-view motion prediction method is applied. Firstly, divide the current PU into multiple sub-PUs with a size equal to 8x8 (or 8x4/4x8 for some corner cases), respectively. Secondly, for each sub-PU in raster scan order, one corresponding block is identified to derive the motion candidate. If the motion information derived from the corresponding block (in the same procedure as the PU-level method) is available, it is set to the IvMC for current sub-PU. Otherwise (it is unavailable), the IvMC of the previous sub-PU is inherited. It is noted that the first available IvMC of one sub-PU in the raster scan order is used for comparisons with other merging candidates.

2.2.4.1.2 **Derivation of inter-view disparity candidate**

To derive the IvDC candidate it is tested for each motion hypothesis of the current block, if the corresponding block is located in an inter-view reference picture that is included in the reference picture list belonging to the current slice and motion hypothesis. If such a reference picture is found, the IvDC candidate is constructed by using the derived disparity as motion vector and the reference index of the found reference picture in the reference picture list of the current slice.

2.2.4.1.3 **Derivation of the shifted candidate**

In addition, one more merging candidate may be derived based on a shifted disparity vector. Such a candidate could be an IvMC derived from a reference block in a reference view with shifted disparity vectors or derived from the first available spatial merging candidate including a disparity motion vector or IvDV. Detailed steps for generating the additional candidate and insertion to the merge candidate list is described as follows:

1. The disparity vector DV is shifted by ((PuWidth / 2 * 4 + 4), (PuHeight / 2 * 4 + 4)) and it is used to derive an shifted IvMC candidate from the reference view. Here, the size of the current prediction unit (PU) is PuWidth x PuHeight. If available, skip step 2 and if this shifted IvMC is not identical to the IvMC without disparity vector shifting, it is inserted to the merge candidate list just before the temporal merging candidate.
2. A candidate, denoted as Disparity Shifted Motion Vector (DSMV) is derived and set to be the additional candidate. If the DSMV is available, it is directly inserted to the merge candidate list in the same position as shifted IvMC. The DSMV is derived as follows:
   - o Identify the first available disparity motion vector (DMV) corresponding to the RefPicList0 from the spatial neighboring blocks.
   - o If the DMV is available, the horizontal component of the motion vector in List 0 is set to DMV shifted by 4 and the vertical component of the motion vector is kept unchanged or reset to 0 depending on BVSP is enabled or not. The reference picture indices and motion vectors in List 1 are directly inherited.
   - o Otherwise, the horizontal component of the motion vector in List 0 and List 1 are set to DV shifted by 4 and both vertical components of motion vectors in two lists are set to 0.

2.2.4.2 Derivation of co-located motion vector candidate

The availability of the co-located vector is specified in Table 1. For the advanced motion vector prediction (AMVP) mode, the co-located motion vector is available for motion vector prediction if the current PU utilized the same kind of prediction (inter prediction or inter-view prediction) as the co-located PU. Otherwise, the co-located motion vector is not available.

For merge, when the target reference index specifies a reference picture in the same view, while the motion vector of the co-located prediction unit (PU) is related to an inter-view reference picture or vice versa, the temporal motion vector prediction (TMVP) candidate might still be available. Therefore an alternative target reference index is derived as described in section 0.

In case that both, the current PU and the co-located PU, utilize inter-view prediction, inter-view motion vectors are scaled merge as specified in section 2.2.4.2.2. For AMVP no inter-view motion vector scaling is applied.

Table 1: The availability of the co-located motion vector

Availability of co-located vector		Prediction type of current PU	Prediction type of co-located PU
Merge	AMVP		
Available	Available	temporal	temporal
Potentially available	Not available	temporal	inter-view
Potentially available	Not available	inter-view	temporal
Available	Available	inter-view	inter-view

2.2.4.2.1 Derivation of the alternative reference index for merge

An alternative reference index is derived for merge, in case that the reference picture with the current target reference index is a different kind of reference picture as the reference picture of the co-located PU.

When the reference picture with the current target reference index is an inter-view reference picture, but the reference picture of the co-located PU is a temporal reference picture, the current target reference index is modified to be the first reference index in the reference picture list of the current block, which specifies an inter-view reference picture.

When the reference picture with the current target reference index is a temporal reference picture, but the reference picture of the co-located PU is an inter-view reference picture, the current target reference index is modified to be the first reference index in the reference picture list of the current block that specifies a temporal reference picture.

2.2.4.2.2 Scaling of inter-view motion vectors

The scaling function for inter-view motion vectors is the same as that in temporal MV scaling, but the scaling factors are derived differently. In case of temporal MV scaling the scaling factor “tb” is the difference between POC of coding block and coding reference block and “td” is the difference between POC of co-located block and this reference block. In case of inter-view scaling, the scaling factors “tb” and “td” are calculated with the difference between view order indices of each block instead of POCs, Hence:

DistScaleFactor = Clip3(–1024, 1023, (tb * tx + 32) >> 6)
tx = (16384 + Abs(td / 2)) / td

where *td* and *tb* are derived as:

td = Clip3(–128, 127, ColViewOrderIdx – ColRefViewOrderIdx)
tb = Clip3(–128, 127, CurrViewOrderIdx – CurrRefViewOrderIdx)

The variables in the above equations are specified as follows:

CurrViewOrderIdx :*ViewOrderIdx* of current picture

ColViewOrderIdx :*ViewOrderIdx* of co-located picture

CurrRefViewOrderIdx :*ViewOrderIdx* of the picture that is referenced by the current picture

ColRefViewOrderIdx :*ViewOrderIdx* of the picture that is referenced by the co-located picture

An example for the scaling of inter-view motion vectors is depicted in Figure 11. The coding block refers to the reference picture of V0 and the neighbouring block refers to the picture of V2. The predictive vector from neighbouring block is scaled since the difference of view index (V1-V0) between coding block and coding reference block is not equal to that (V1-V2) between neighbouring block and this reference block.

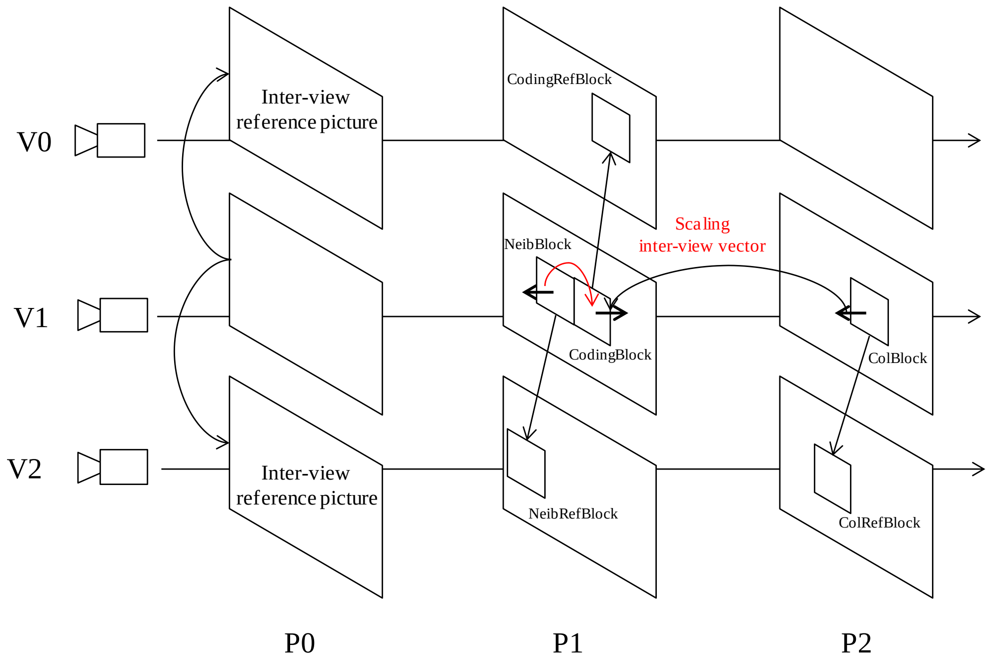

V0
V1
V2
Inter-view reference picture
Inter-view reference picture
CodingRefBlock
NeibBlock
Scaling inter-view vector
CodingBlock
ColBlock
NeibRefBlock
ColRefBlock
P0
P1
P2

Figure 11: Inter-view motion vector scaling in TMVP

2.2.5 Advanced residual prediction

Advanced residual prediction (ARP) is a coding tool to exploiting the residual correlation between views. In ARP, a residual predictor is produced by aligning the motion information at the current view for motion compensation in the reference view. In addition, weighting factors are introduced to compensate the quality differences between views. When ARP is enabled for one block, the difference between current residual and the residual predictor is signalled.

2.2.5.1 Advanced residual prediction for temporal residual

Figure 12 illustrates the prediction structure of residual prediction method when current block is predicted from a temporal reference picture. D_c represents the current block in the current view (view 1), B_c, and D_r denote the representation of D_c in the reference view (view 0) at time T_j and D_c's temporal prediction from the same view (view 1) at time T_i. V_D denotes the motion from D_c to D_r. Since D_c and B_c are actually projections of the same object in two different views, these two blocks should share the same motion information. Therefore, B_c's temporal prediction B_r in view 0 at time T_i can be located from B_c by applying the motion information of V_D. The residual of B_c with motion information of V_D is then multiplied by a weighting factor and used as the predictor for current residual.

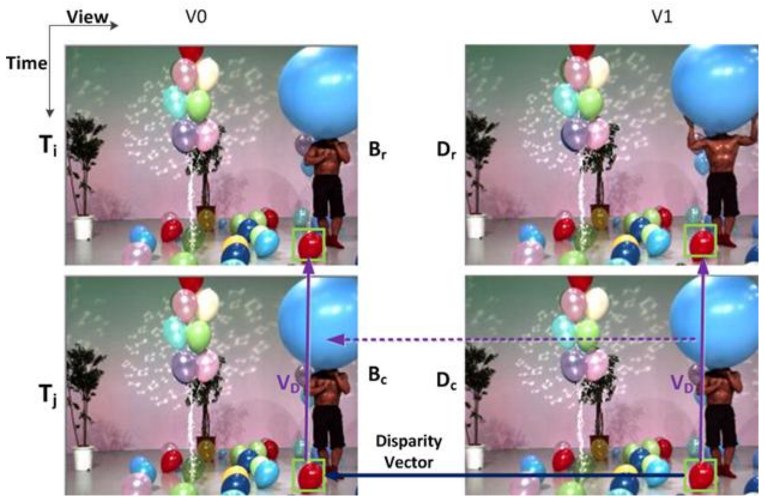


Figure 12: Prediction structure of the proposed ARP in 3D-HEVC.

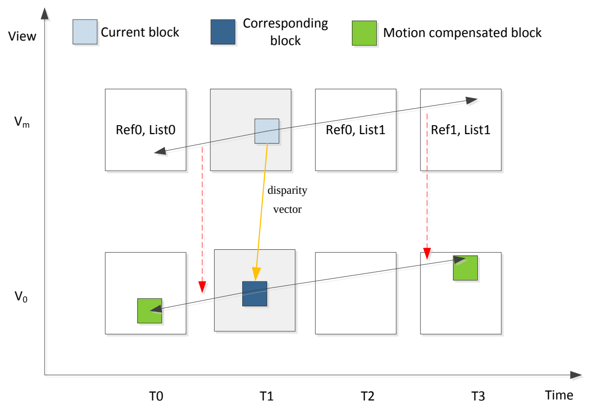


Figure 13: Relationship among current block, reference block and motion compensated block

Decoding process of ARP

Main procedures of the proposed ARP at the decoder side can be described as follows:

1. Obtain a disparity vector as specified in section 2.2.2, pointing to a target reference view. Then, in the picture of the reference view within the same access unit, the corresponding block is located by the disparity vector.
2. Re-use the motion information of the current block to derive the motion information for the reference block. Apply motion compensation for the corresponding block based the same motion vector of current block and derived reference picture in the reference view for the reference block, to derive a residue block. The relationship among current block, corresponding block and motion compensated block is shown in Figure 13. The reference picture in the reference view (V_0) which has the same POC (Picture Order Count) value as the reference picture of current view (V_m) is selected as the reference picture of the corresponding block.
3. Apply the weighting factor to the residue block to get a weighted residue block and add the values of the weighted residue block to the predicted samples.

Weighting factor

Three weighting factors are used in residual prediction, i.e., 0, 0.5 and 1. The one leading to minimal rate-distortion cost for the current CU is selected as the final weighting factor and the corresponding weighting index (0, 1 and 2 which correspond to weighting factor 0, 1, and 0.5, respectively) is transmitted in the bitstream at the CU level. All PU predictions in one CU share the same weighting factor. When the weighting factor is equal to 0, residual prediction is not used for the current CU. When coding the weighting factor indexes with CABAC, the weighting factors of left and above neighbouring blocks are utilized to select the contexts and the initialized probabilities are unequal to 0.5.

Reference picture selection via motion vector scaling

The motion vectors of the current PU are scaled towards a fixed picture before performing motion compensation, when the weighting factor is unequal to 0. The fixed picture is defined as the first temporal reference picture of each reference picture list.

Availability check of ARP reference pictures

Since it is possible that the reference pictures of the dependent view which are used in ARP process is not stored in decoded picture buffer (DPB), the availability check of the dependent view pictures is utilized, which is beneficial to avoid decoder crash. Considering the case that output picture handling is dependent on decoder implementations (i.e. the output images may be post-filtered), and the HEVC reference picture management philosophy that reference pictures shall be explicitly indicated by reference picture set, the condition that whether the picture is “marked as reference” is also checked. The DPB check is carried out in slice level.

Interpolation filter

A bi-linear interpolation filter is used regardless whether the block is in base views or dependent views when residual prediction is applied.

2.2.5.2 Advanced residual prediction for inter-view residual

Similar to the ARP for temporal residual as depicted in Figure 12, when current block is predicted from an inter-view reference picture, prediction of inter-view residual is enabled. Firstly the inter-view residual within a different access unit is calculated, then the calculated residual information is used to predict the inter-view residual of the current block.

Practically, still three related blocks are identified: the reference block in the reference view located by the disparity motion vector of the current block (denoted by *Base*); the reference block of *Base* in the reference view (denoted by *BaseRef*) located by the temporal motion vector (mvLX) and reference index, if available, contained by *Base*; a reference block in current view (denoted by *CurrRef*) by reusing the temporal motion information from *Base*, as shown in Figure 14.

With the identified three blocks, the residual predictor of the residual signal of current PU can be calculated as the difference between these two blocks in the different access unit: *CurrRef* - *BaseRef*. Furthermore, the inter-view predictor is multiplied by a weighting factor as used in current ARP.

Similarly, bi-linear filter is used to generate the three relative blocks as in current design of ARP for temporal residual prediction. Furthermore, when the temporal motion vector contained by *Base* points to a reference picture that is in a different access unit of the first available temporal reference picture of current PU, it is firstly scaled to the first available temporal reference picture and the scaled motion vector is used to locate two blocks in a different access unit.

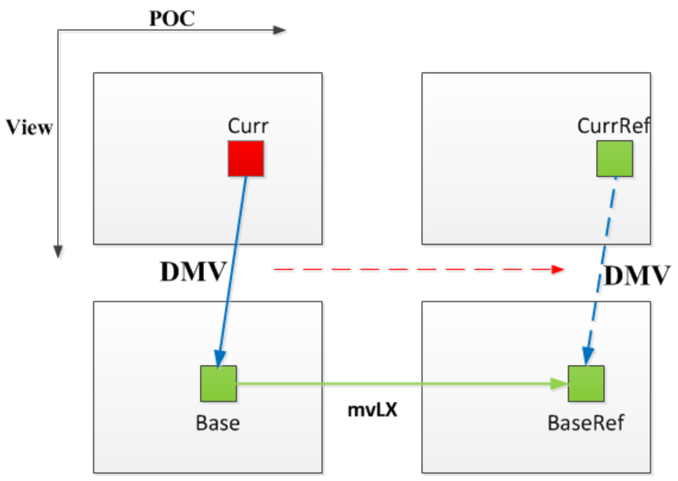


Figure 14. Prediction structure of ARP for inter-view residual.

2.2.6 Illumination compensation (IC)

A linear illumination compensation model is utilized to adapt luminance and chrominance of inter-view predicted blocks to the illumination of the current view. The parameters (including scaling factor *a* close to 1 and an offset *b*) of the linear model are estimated for each CU using reconstructed neighbouring samples of the current block and of the reference block used for prediction. The corresponding neighbouring samples in the reference view are identified by the disparity motion vector of the current PU, as shown in Figure 15.

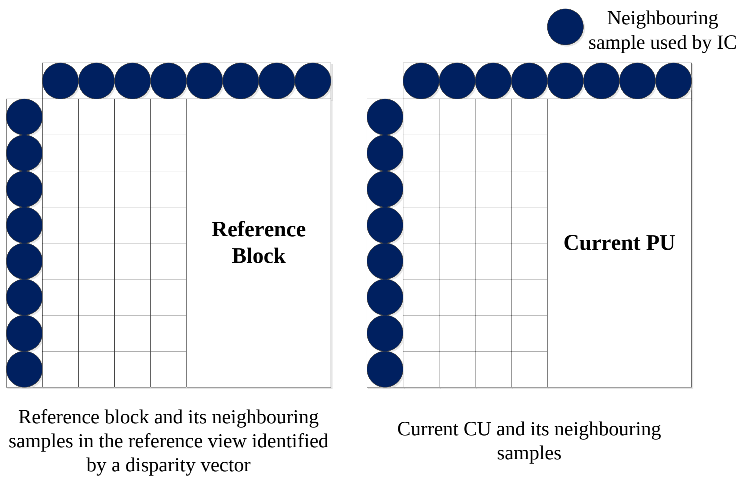

Neighbouring sample used by IC
Reference Block
Current PU
Reference block and its neighbouring samples in the reference view identified by a disparity vector
Current CU and its neighbouring samples

Figure 15: Neighbouring samples for the derivation of illumination compensation parameters.

3D-HEVC

For the current PU, its neighbouring samples, which indicated by y_i for i = 0..N-1, together with the corresponding neighbouring samples of the reference block, which indicated by x_i for i = 0..N-1, are the input parameters for a linear model to derive *a* and an offset *b* by a least squares solution wher the following equation E (*a*, *b*) is minimized:

$$\mathrm{E}\,(a, b) = \sum_i \left(y_i - a x_i - b\right)^2 + \lambda(a-1)^2 \tag{1}$$

The following is the normal equation with

$$\begin{pmatrix} \sum_i x_i x_i + \lambda & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_i x_i y_i + \lambda \\ \sum_i y_i \end{pmatrix} \tag{2}$$

Then parameter a is derived by

$$a = \frac{N \sum_i x_i y_i - \sum_i x_i \sum_i y_i + \lambda}{N \sum_i x_i x_i - \sum_i x_i \sum_i x_i + \lambda} \tag{3}$$

wherein λ is set equal to $(\sum_i x_i x_i \gg 7)$.

Then parameter b is derived by

$$b = \sum_i y_i - a * \sum_i x_i \tag{4}$$

To further simplify the IC process, illumination compensation parameters are derived from a neighbouring sample array decimated by a factor of 2 and illumination compensation is disabled for 4x4 chroma blocks. Moreover illumination compensation is only applied for coding blocks with a partition mode of 2Nx2N.

To avoid floating point, division and more than 32 bit computation, the above conceptural parameter derivation process of (a, b) is conveyed by follows. Here, the weighting factor icWeight, which is corresponding to a << IC_SHIFT, and the offset factor icOffset, which is corresponding to b, is derived. IC_SHIFT is constant 5.

The variables numerDiv, denomDiv, lambda, avgShift and avgOffset are derived as:

$$\begin{aligned} &\text{denomDiv} = N \sum_i x_i x_i - \sum_i x_i \sum_i x_i + \lambda = \{(\sum_i x_i x_i + \text{lambda}) \ll \text{avgShift}\} - \sum_i x_i \sum_i x_i \\ &\text{numerDiv} = N \sum_i x_i y_i - \sum_i x_i \sum_i y_i + \lambda = \{(\sum_i x_i y_i + \text{lambda}) \ll \text{avgShift}\} - \sum_i x_i \sum_i y_i \\ &\text{lambda} = (\sum_i x_i x_i) \gg 7 \\ &\text{avgShift} = \log2(N) \\ &\text{avgOffset} = 1 \ll (\text{avgShift}-1) \end{aligned} \tag{5}$$

To restrict the range of $N \sum_i x_i x_i$ or $N \sum_i x_i y_i$ equal to or less than 32 bit integer range for N is 128, a normalized parameter precShift is introduced and the following process is utilized.

$$\begin{aligned} &\text{denomDiv} = \{(\sum_i (x_i x_i \gg \text{precShift}) + \text{lambda}) \ll \text{avgShift}\} - (\sum_i x_i \sum_i x_i \gg \text{precShift}) \\ &\text{numerDiv} = \{(\sum_i (x_i y_i \gg \text{precShift}) + \text{lambda}) \ll \text{avgShift}\} - (\sum_i x_i \sum_i y_i \gg \text{precShift}) \\ &\text{lambda} = ((\sum_i x_i x_i) \gg \text{precShift}) \gg 7 \end{aligned} \tag{6}$$

where precShift = Max(0, bitDepth − 12), which is derived by solving the equation of bitDepth*2 +7+precShift <= 31.

A table divCoeff [x], for x = 0..63, is defined as:

$$\text{ivCoeff}[x] = \begin{cases} \dfrac{2^{\text{IC\_TABLE}} + \frac{x}{2}}{x}, & 0 < x \leq 63 \\ 0, & x = 0 \end{cases} \tag{7}$$

Using the valiables, icWeight and icWeight can be derived as follows without devision.

$$\text{icWeight} = (\text{numerDiv} * \text{divCoeff}[\text{denomDiv}]) \gg (\text{IC\_TABLE}-\text{IC\_SHIFT}) \tag{8}$$

which is correspond to (numerDiv / denomDiv) << IC_SHIFT

$$\text{icOffset} = \{\textstyle\sum_i y_i - ((\text{icWeight} * \sum_i x_i) \gg \text{IC\_SHIFT}) + \text{avgOffset}\} \gg \text{avgShift} \tag{9}$$

Since the table divCoeff[x] is only defined where x is in the range of 0 to 63, denomDiv is normalized to sDenomDiv as follows before divCoeff derivation is applied.

$$\text{sDenomDiv} = \text{denomDiv} >> \text{shiftDenom} \tag{10}$$

where shiftDenom = Max(0, Floor(Log2(Abs(denomDiv))) − 5)

Likewise, to restrict the range of numerDiv * divCoeff[denomDiv] being at least equal to or less than 32 bit integer range, numerDiv is clipped and normalized to sNumerDiv

$$\begin{gathered}\text{numerDiv} = \text{Clip3}(0, 2 * \text{denomDiv}, \text{numerDiv}) \\ \text{sNumerDiv} = \text{numerDiv} >> \text{shiftNumer}\end{gathered} \tag{11}$$

where shiftNumer = Max(0, shiftDenom − 12)

Note: Assuming shiftNumer is defined as Max(0, shiftDenom – IC_SHIFT_DIFF), IC_SHIFT_DIFF shall be equal to or larger than 14 to guarantee the range of numerDiv * divCoeff[denomDiv]

Finally, icWeight and icOffset derivation process is defined as

$$\begin{gathered}\text{icWeight} = (\text{sNumerDiv}*\text{InvTable}[\text{sDenomDiv}]) >> (\text{IC\_TABLE} - \text{IC\_SHIFT} - \\ \text{shiftNumer} + \text{shiftDenom}) \\ \text{icOffset} = \{\textstyle\sum_i y_i - ((\text{icWeight} * \sum_i x_i) \gg 5) + \text{avgOffset}\} \gg \text{avgShift}\end{gathered} \tag{12}$$

Note: IC_TABLE and IC_SHIFT are defined as 15 and 5 respectively.

The illumination compensation algorithm is only applied to texture views.

Signalling of illumination compensation

Whether illumination compensation is used is signalled on coding unit.

In Skip / Merge mode, ic_flag is conditionally sent depend on merge_idx and the slice segment header flag *slice_ic_disable_merge_zero_idx_flag*. If ic_flag is not sent on merge mode, ic_flag is inferred to 0. Detailed condition is shown in Table 1.

Table 2: Conditions for the availiability of the ic_flag

Mode	Condition	Note
Skip / Merge (merge_flag[x0][y0])	!(merge_idx[x0][y0] = = 0 && *slice_ic_disable_merge_zero_idx_flag*)	When inter-view candidate is used (merge_idx == 0) and encoder send a flag *slice_ic_disable_merge_zero_idx_flag*, then ic_flag is not sent (inferred to 0) (independent of ref_idx_lX)
AMVP (!merge_flag[x0][y0])	When the CU has an inter-view reference picture.	conditionally sent depending on ref_idx_lX

When merge_idx is equal to 0, the temporal inter-view motion predictor candidate is typically used. Therefore, inter-view prediction is not used very often. To avoid the overhead of signalling the ic_flag, in this case, illumination compensation is not available. It is utilized by setting slice_ic_disable_merge_zero_idx_flag 1.

In some pictures where the inter-view prediction is mostly used, the above assumption does not hold. So the merge_idx base skipping of the ic_flag is only applied under the condition that (POC % IntraPeriod) is not 0. This POC based condition is not decided by decoder but by encoder, which sends a slice header flag slice_ic_disable_merge_zero_idx_flag. This allows encoders can control the condition depends on coding structure or sequences.

Similar to the context coding of ARP weighting factors, when the ic_flag is coded with CABAC, the values of ic_flag associated with left and above coding units are utilized to select the contexts and the initialized probabilities are set to 0.5.

2.2.7 View synthesis prediction (VSP)

View synthesis prediction (VSP) provides a predictor for each PU using depth information to reduce inter-view redundancy. In VSP, a neighbouring block disparity vector (DVx, DVy) is derived as shown in Step 1 of Figure 16, as described in section 2.2.2.1. With the derived disparity vector (DVx, DVy), a depth block (x+DVx, y+DVy) is fetched from a reference view depth image and used as an estimator for the depth information of the current PU, as shown in Step 2 of Figure 16. From the estimated depth block, disparity vectors are derived accordingly at the sub-block level with the input camera parameters. Given the derived disparity vectors, a backward warping is invoked to find corresponding reference samples from the reference view and generate the VSP predictor for the current PU, as shown in Step 3 of Figure 16.

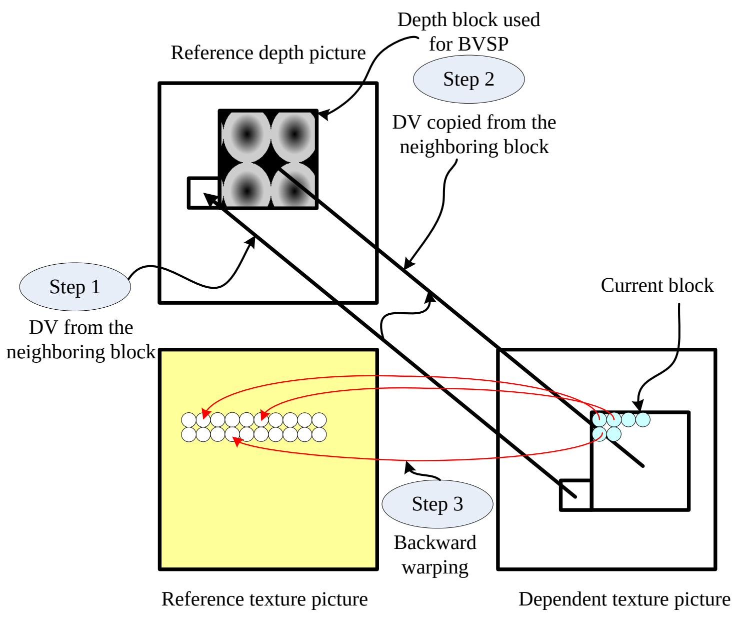

Depth block used for BVSP
Reference depth picture
Step 2
DV copied from the neighboring block
Step 1
DV from the neighboring block
Current block
Step 3
Backward warping
Reference texture picture
Dependent texture picture

Figure 16: Illustration of the VSP scheme with the neighboring block disparity vector

Signalling

VSP can be disabled by a flag in the VPS in order to enable coding of texture without depth maps.

At PU level a VSP flag indicates whether the block is coded with VSP mode or conventional inter mode. The VSP flag is derived in the merging process and set equal to one, when a VSP candidate is signalled by the merge index. Multiple VSP candidates might be present in the merge list.

One specific VSP candidate is inserted into the merge candidate list with the derived disparity vector and the associated reference view index for current PU from NBDV process as described above. In addition, multiple other VSP candidates may be inherited from spatial neighboring blocks. When a spatial neighboring block is coded in a VSP mode and such a spatial neighbouring block is within current coding tree unit (CTU), the associated candidate is treated as an additional VSP candidate and the derived disparity vector and associated reference view index for the current PU is used. It is noted that when current PU is coded with either ARP mode or IC mode, the VSP candidate shouldn't be included in the merge candidate list.

Derivation of the VSP merging candidate

Reference picture indices and motion vectors of the BVSP candidate are derived by the following method:

- The reference view index denoted by refViewIdx is set equal to the view index of the view the derived disparity vector from NBDV is related to;
- When a picture with refViewIdx is included in list RefPicListX (either RefPicList0 or RefPicList1) the reference index of the candidate for list X is set equal to the reference index of this picture in the list X and the motion vector

is set equal to the disparity vector from NBDV process.

– If the current slice is a B slice, the availability of an interview reference picture with view order index denoted by refViewIdxLY not equal to refViewIdxLX in the reference picture list other than RefPicListX, i.e., RefPicListY with Y being 1-X is checked;
  – If such a different interview reference picture is found, bi-predictive VSP is applied and the following applies:
    – The corresponding reference picture index of the different interview reference picture and the scaled disparity vector from NBDV process based on view order indices are used as the motion information of the BVSP merging candidate in RefPicListY.
  – Otherwise, uni-predictive VSP is applied with RefPicListX as the reference picture list for prediction.

P- and B prediction

VSP can be either uni-direction prediction or bi-direction prediction. The prediction direction depends on the slice type and the availability of reference views in the reference picture lists as described above, which is elaborated as follows.

- If the current slice is a P slice, VSP is performed in the manner of uni-direction prediction. One predictor is generated from the reference picture list, which has the interview reference picture identified by the disparity vector defined in the VSP candidate.
- If the current slice is a B slice, VSP may be performed in either uni-direction or bi-direction prediction. And the prediction direction depends on the availability of reference views in the reference picture lists.
    - If each reference picture list has a different interview reference picture, VSP is performed in the bi-direction manner. Specifically, a depth block is fetched using the derived neighboring block disparity vector from the depth image of the associated reference view. The depth values are then converted to different disparity vectors relative to the different reference views in the two reference picture lists and two compensation predictors are generated. Finally, the two predictors are combined into a single predictor in a weighted process.
    - Otherwise, VSP is achieved in the uni-direction manner. The predictor is generated from the reference picture list, which has the interview reference picture as identified by the disparity vector defined in the VSP candidate.

Handling of VSP coded block

Since the reference index indiciates an inter-view reference picture, VSP coded blocks are considered transparently as normal inter-layer predicted blocks.

In section 2.2.2.1 a methods is described that provides a disparity vector when a neighbouring block is coded in interview prediction mode called NBDV. As the VSP mode can be assumed as a special interview prediction method, NBDV also provides a disparity vector if a neighbouring block is coded in a VSP mode. The returned disparity vector is derived from the maximum disparity of the subsampled depth values in the associated depth block which were used for VSP compensation.

Units for motion compensation

For a BVSP coded prediction unit with size equal to W*H, when either W % 8 (e.g., 4x16, 12x16) or H % 8 (e.g., 16x12, 16x4) is unequal to 0, the unit for performing motion compensation is set to 4x8 or 8x4, respectively. For all the other cases, the unit for performing motion compensation of a BVSP coded prediction unit, denoted by W*H could be either 8x4 or 4x8. The motion compensation size is determined as specified in the following:

– 4 corners of corresponding depth block with the same size as current PU as depicted in Figure 17 identified by the disparity vector from the NBDV process are checked by:

```
$if\ (vdepth[TL] < vdepth[BR]\ ?\ 0:1)\ != (vdepth[TR] < vdepth[BL]\ ?\ 0:1)$
    $use\ 4x8\ partition\ (W = 4,\ H = 8)$
$else$
    $use\ 8x4\ partition\ (W = 8,\ H = 4)$
```

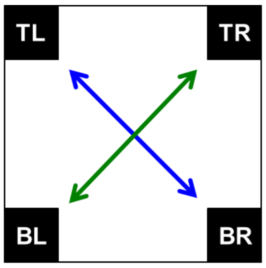


Figure 17: Four corner samples of one corresponding depth block

2.3 Coding of Depth Maps

For the coding of depth maps, basically the same concepts of intra-prediction, motion-compensated prediction, disparity-compensated prediction, and transform coding as for the coding of the video pictures are used. However, some tools have been modified for depth maps, other tools have been generally disabled, and additional tools have been added.

As a first difference to the coding of video pictures, the inter-view motion, residual prediction and view synthesis prediction as described in sections 2.2.3, 2.2.5 and 2.2.7 , respectively, are not used for depth coding. Instead, motion parameters are derived based on coded data in the associated video pictures as will be described in sec. 1.1.1.1 below. The other differences are described in the following subsections.

2.3.1 Modified motion compensation and motion vector coding

In contrast to natural video, depth maps are characterized by sharp edges and large regions with nearly constant values. The eight-tap interpolation filters that are used for motion-compensated interpolation in HEVC, can produce ringing artefacts at sharp edges in depth maps, which are visible as disturbing components in synthesized intermediate views. For avoiding this issue and for decreasing the encoder and decoder complexity, the motion-compensated prediction (MCP) as well as the disparity-compensated prediction (DCP) has been modified in a way that no interpolation is used. That means, for depth maps, the inter-picture prediction is always performed with full-sample accuracy. For the actual MCP or DCP, a block of samples in the reference picture is directly used as prediction signal without interpolating any intermediate samples. In order to avoid the transmission of motion and disparity vectors with an unnecessary accuracy, full-sample accurate motion and disparity vectors are used for coding depth maps. The transmitted motion vector differences are coded using full-sample instead of quarter-sample precision.

2.3.2 Disabling of in-loop filtering

The in-loop filters in the HEVC design have been particularly designed for the coding of natural video. For the coding of depth maps, these filters are less useful. In order to decrease the encoder and decoder complexity, the in-loop filters have been disabled for depth coding. This includes the following filters:

- the de-blocking filter;
- the adaptive loop filter (Wiener filter);
- the sample-adaptive loop filter.

2.3.3 Depth modelling modes

Depth maps are mainly characterized by sharp edges (which represent object borders) and large areas of nearly constant or slowly varying sample values (which represent object areas). While the HEVC intra prediction and transform coding is well-suited for nearly constant regions, it can result in significant coding artefacts at sharp edges, which are visible in synthesized intermediate views. For a better representation of edges in depth maps, four new intra prediction modes for depth coding are added. In all four modes, a depth block is approximated by a model that partitions the area of the block into two non-rectangular regions, where each region is represented by a constant value. The information required for such a model consists of two elements, namely the partition information, specifying the region each sample belongs to, and the region value information, specifying a constant value for the samples of the corresponding region. Such a region value is referred to as constant partition value (CPV) in the following. Two different partition types are used, namely Wedgelets and Contours, which differ in the way the segmentation of the depth block is derived. The depth modelling modes are integrated as an alternative to the conventional intra prediction modes specified in HEVC. Similar as for the intra prediction modes, a residual representing the difference between the approximation and the original depth signal can be transmitted via transform coding. In the following, the approximation of depth blocks using the four new depth modelling modes is described in more detail.

It is differentiated between Wedgelet and Contour partitioning. For a Wedgelet partition, the two regions are defined to be separated by a straight line, as illustrated in Figure 18, in which the two regions are labelled with P_1 and P_2. The separation line is determined by the start point S and the end point P, both located on different borders of the block. For the continuous signal space (see Figure 18, left), the separation line can be described by the equation of a straight line. The middle image of Figure 18 illustrates the partitioning for the discrete sample space. Here, the block consists of an array of samples with size $u_B \times v_B$ and the start and end points correspond to border samples. Although the separation line can be described by a line equation as well, the definition of regions P_1 and P_2 is different here, as only complete samples can be assigned as part of either of the two regions. For employing Wedgelet block partitions in the coding process, the partition information is stored in the form of partition patterns. Such a pattern consists of an array of size $u_B \times v_B$ and each element contains the binary information whether the corresponding sample belongs to region P_1 or P_2. The regions P_1 and P_2 are represented by black and white samples in Figure 18 (right), respectively.

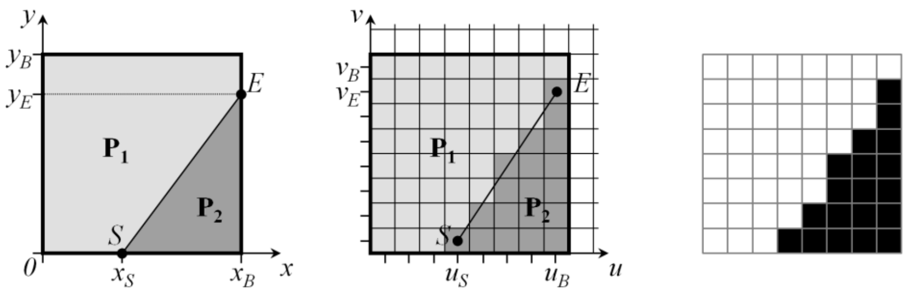


Figure 18: Wedgelet partition of a block: continuous (left) and discrete signal space (middle) with corresponding partition pattern (right).

Unlike for Wedgelets, the separation line between the two regions of a Contour partition of a block cannot be easily described by a geometrical function. As illustrated in Figure 19, the two regions P_1 and P_2 can be arbitrary shaped and even consist of multiple parts. Apart from that the properties of Contour and Wedgelet partitions are very similar. For employing Contour partitions in the coding process, the partition pattern (see example in Figure 19, right) is derived individually for each block from the signal of a reference block. Due to the lack of a functional description of the region separation line, no pattern lookup lists and consequently no search of the best matching partition are used for Contour partitions.

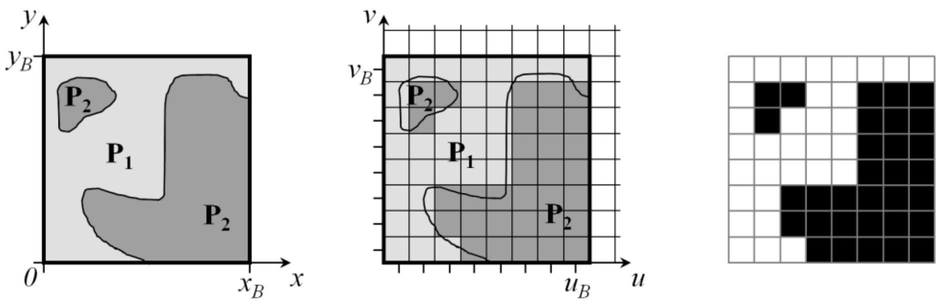


Figure 19: Contour partition of a block: continuous (left) and discrete signal space (middle) with corresponding partition pattern (right).

Apart from the partition information, either in form of a Wedgelet or a Contour partition, the second information required for modelling the signal of a depth block is the CPV of each of the two regions. For a given partition the best approximation is consequently achieved by using the mean value of the original depth signal of the corresponding region as the CPV.

Four depth-modelling modes, which mainly differ in the way the partitioning is derived and transmitted, have been added:

- Mode 1: Explicit Wedgelet signalling;
- Mode 3: Restricted signalling and inter-component prediction of Wedgelet partitions;
- Mode 4: Inter-component-predicted Contour partitioning.

These depth-modelling modes as well as the signalling of the modes and the constant partition values are described in the following four subsections.

2.3.3.1 Mode 1: Explicit Wedgelet Signalization

The basic principle of this mode is to find the best matching Wedgelet partition at the encoder and transmit the partition information in the bitstream. At the decoder the signal of the block is reconstructed using the transmitted partition information.

The Wedgelet partition information for this mode is not predicted. At the encoder, a search over a set of Wedgelet partitions is carried out using the original depth signal of the current block as a reference. During this search, the Wedgelet partition that yields the minimum distortion between the original signal and the Wedgelet approximation is selected. The resulting prediction signal is then evaluated using the conventional mode decision process.

A fast search of the best matching partition is essential for employing Wedgelet models in the depth coding process. This fast search algorithm is further described in section 1.1.1.1.

2.3.3.2 Mode 4: Inter-component prediction of Contour partitions

The basic principle of this mode is to predict a Contour partition from a texture reference block by inter-component prediction. Like for the inter-component prediction of a Wedgelet partition pattern, the reconstructed luminance signal of the co-located block of the associated video picture is used as a reference, as illustrated in the bottom row of . In contrast to Wedgelet partitions, the prediction of a Contour partition is realized by a thresholding method. Here, the mean value of the texture reference block is set as the threshold and depending on whether the value of a sample is above or below the sample position is marked as part of region P_1 or P_2 in the resulting Contour partition pattern.

2.3.3.3 Constant partition value coding

The method for CPV coding is the same for all four modes introduced above, as it does not distinguish between partition types, but rather assumes that a partition pattern is given for the current depth block. As illustrated in Figure 20, three types of CPVs are differentiated: original, predicted, and delta CPVs.

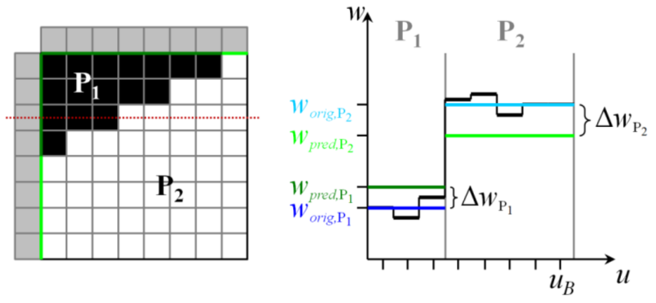


Figure 20: CPVs of block partitions: CPV prediction from adjacent samples of neighbouring blocks (left) and cross section of block (right), showing relation between different CPV types.

The cross section of the block in Figure 20, right, schematically shows that the original CPVs are calculated as the mean value of the signal covered by the corresponding region. Although these values lead to the best approximation for the given partition, they are not available at the decoder as they require the original signal.

Therefore prediction of CPVs is introduced. These predicted CPVs are derived from information that is also available at the decoder, namely adjacent samples of the neighbouring left and top block. As shown in Figure 21, two samples of the first sample, last sample or middle sample of the top reference sample row or the left reference sample column may be chosen to generate a DC predator under different situations. Therefore, each time, the CPV predictor is calculated by only up to two samples. It is noted that different from HEVC intra prediction process, wherein the adjacent samples may be filtered to derive the prediction value, in depth modeling modes, the unfiltered adjacent samples are utilized to obtain the CPV predictors.

More specifically, the CPV predictor generation process could be defined as follows:

Assume the partition value of the left-top sample ($c_{0,0}$) is X, where X= 0 or 1, given the partition pattern $bPattern_{x,y}$, where x = 0..$N - 1$, y = 0..$N-1$, the predicted CPV values, denoted by DC_0 and DC_1, are derived by follows:

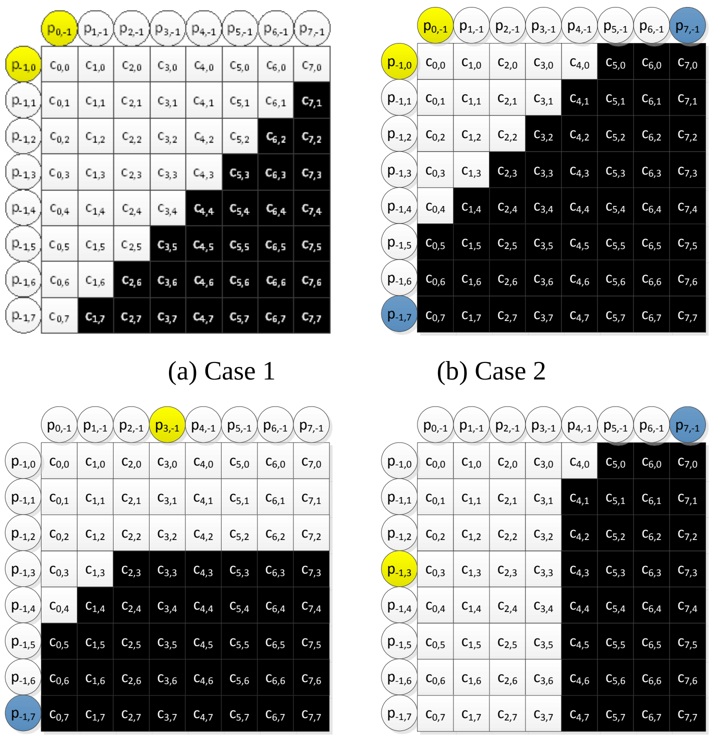
```
Set bT = ($bPattern_{0,0}$ != $bPattern_{N-1,0}$)? 1 : 0;
Set bL = ($bPattern_{0,0}$ != $bPattern_{0,N-1}$)? 1 : 0
If bT equals bL
    – $DC_X = (p_{-1,0} + p_{0,-1}) >> 1$
```

– $DC_{1\text{-}X} = bL\ ?\ (p_{-1,N-1} + p_{N-1,-1}) >> 1 : 2^{BitDepth-1}$

Otherwise

– $DC_X = bL\ ?\ p_{(N-1)>>1,-1} : p_{-1,(N-1)>>1}$

– $DC_{1\text{-}X} = bL\ ?\ p_{-1,N-1} : p_{N-1,-1}$



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

Figure 21: Selection of reference samples for difference partition pattern cases.

Depending on the similarity between original signal of the block and adjacent samples, the predicted and original CPVs may differ significantly. This difference is referred to as delta CPVs. By calculating the delta CPVs at the encoder and transmitting them in the bit stream, it is possible to reconstruct the CPVs at the decoder.

Depending on the Intra Mode of the current prediction unit, delta CPVs can be determined and signalled in two different ways. In one method, delta CPVs are directly transformed, quantized and entropy coded. In another method, when DLT is present, signalling using a DLT is applied for the DMM modes as described in section 2.3.4.

Quantized signalling

Although the distortion of the reconstructed signal is considerably reduced by the delta CPVs, the benefit of this approach is delimited by the additional bit rate required for transmitting the delta CPVs. Therefore, a linear quantization is introduced for the delta CPVs. This method is also used in transform coding and the step size of the quantization is set as a function of the QP. The delta CPVs are linearly quantized at the encoder and de-quantized before reconstruction at the decoder.

Search of optimal DC values

In case the distortion is not measured for the original depth, but for synthesized views, the delta CPV derivation process is extended by a minimum distortion search, which iterates over all possible delta CPV combinations for the two partitions. For the sake of efficient processing and signalling the range of tested values is limited. The search results in the combination of delta CPVs that causes the minimum distortion in synthesized views.

The optimized search strategy basically consists of a coarse search and a refinement step. In more detail the search works as follows: Initially the distortion of using the partition values that are calculated as the mean value of the original sample values covered by the corresponding region is determined. For SDC as described in section 2.3.4 the offset between these values and the predicted partition values is simply transmitted without a VSO-based minimum distortion search. However, for DMM modes the search tests all combinations of offset values in a certain range around the predicted and original partition values. The limits of the search range start from an offset value of 0 and the upper limit is restricted by the actual range of depth values. First, a coarse search is carried out, testing offset values at intervals of 4. For each tested combination of offset values the distortion is compared to the initial distortion achieved with the original partition values. Only if at least one of the coarse offset combinations leads to a smaller distortion than the original partition values, the refinement step is carried out for the best coarse combination. The refinement step simply consists of testing all offsets in the range of [-3, 3] around the best coarse offset combination.

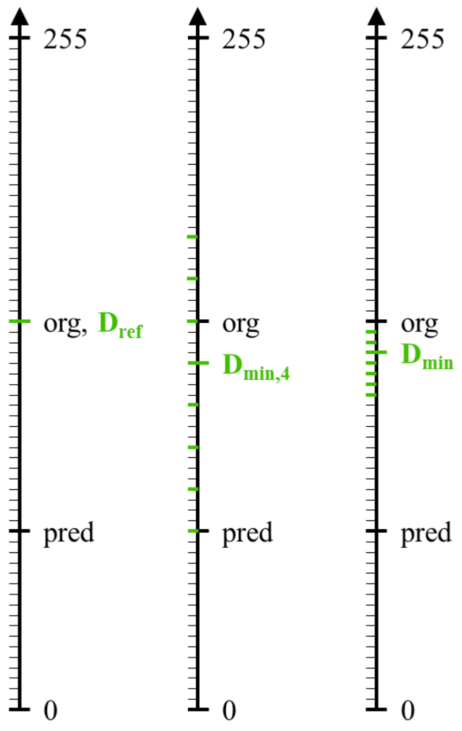


Figure 22: Optimized search strategy for non-quantized partition offset values.

2.3.3.4 Mode selection

In the encoding process, for an intra-coded CU, one of the described depth modelling modes or one of the conventional intra prediction modes is selected. If a depth modelling mode is selected, the selected mode and the associated prediction data have to be signalled in the bitstream in addition to a syntax element that specifies the usage of a depth modelling mode. The following four depth modelling modes are defined:

- *Wedgelet_ModelIntra*: Intra modelling of Wedgelet block partition
- *Contour_PredTexture*: Inter-component prediction of Contour block partition

Each of the two modes can be applied with or without delta CPVs, depending on the decoded delta CPV equal to 0 or not.

2.3.3.5 Signalling of mode parameters in the bitstream

For DMMs following parameters are signalled in the depth intra parameters structure as described in section 2.3.4.1.2.

Mode *Wedgelet_ModelIntra*: For this mode, the Wedgelet partition information is explicitly signalled in the bitstream by the index of the corresponding pattern in the Wedgelet pattern lookup list. The index is signalled with a fixed number of bins. The number of bins used for transmitting the index is given by the size of the list of possible Wedgelet patterns.

Mode *Contour_PredTexture*: For this mode, no additional signalling regarding the partition information is required.

Delta CPVs: In case the delta CPVs are transmitted (which is signalled by the transmitted mode ID), the two quantized values are signalled in the bitstream consecutively. For each CPV, a bin string consisting of the absolute value and the sign is transmitted. The sign is coded as a single bin, and the absolute value is coded using a truncated unary code (with 13 bins in the unary part and an exponential golomb code suffix).

2.3.3.6 Signalling in the bitstream

In case the CPVs are transmitted (which is signalled by **edge_dc_flag**), two quantized values are signalled in the bitstream consecutively. For each CPV, a bin string consisting of the absolute value and the sign is transmitted. The sign is coded as a single bin, and the absolute value is coded using a truncated unary code (with 13 bins in the unary part and an exponential golomb code suffix).

2.3.4 Segment-wise DC coding

The Segment-wise DC Coding (SDC) approach is an alternative intra coding mode. Whether SDC is used is signalled in the depth intra parameters structure as described in section 2.3.4.1.2 at PU level. For SDC, the depth block is intra predicted by a conventional Planar mode or depth modelling mode 1. The partition size of CU containing a SDC coded PU is always 2Nx2N. Residual data is not coded as quantized transform coefficients but one or two constant residual values are signalled.

In summary following information are signalled for SDC-coded blocks:

1. The **type of segmentation/prediction** of the current block. Possible values are
    a. DMM Mode 1 – Explicit Wedgelets (2 segments)
    b. Planar (1 segment)
2. For the DMM mode, additional prediction information is coded, as described in section 2.3.3
3. For each resulting segment, a **residual value** (in the sample domain) is signalled in the bitstream

Before coding, the residual values are mapped to values, which are present in the original, uncompressed depth map by using a Depth Lookup Table (DLT). Consequently, residual values can be coded by signalling only the index into this lookup table, which reduces the bit depth of residual magnitudes. This mapping table is transmitted to the decoder for the inverse lookup from index to valid depth value.

The advantage of using this lookup table is the reduced bit depth of the residual index for sequences with reduced depth value range (e.g. all estimated depth maps where not all depth values are present).

At encoder side SDC process utilizes the mean of the original depth value (d_{orig}) and the predicting depth value (d_{pred}). As illustrated in the example shown in Figure 23, for SDC (Planar mode), $\mathrm{d_{pred}}$ is calculated as the average of the left-top, right-top, left-bottom, and right-bottom samples in a predicted block. For SDC (DMM Mode 1), $\mathrm{d_{pred}}$ of each segment is derived by the left-top, right-top, left-bottom, and right-bottom samples which belong to the same segment in a predicted block.

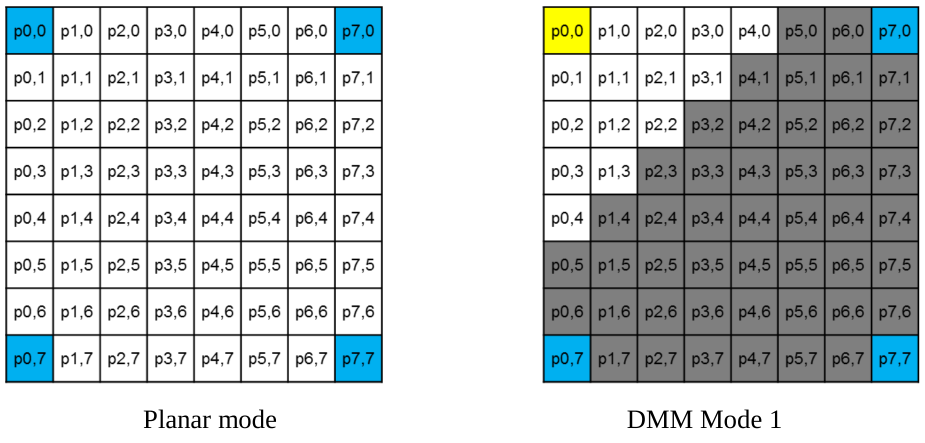

p0,0 p1,0 p2,0 p3,0 p4,0 p5,0 p6,0 p7,0
p0,1 p1,1 p2,1 p3,1 p4,1 p5,1 p6,1 p7,1
p0,2 p1,2 p2,2 p3,2 p4,2 p5,2 p6,2 p7,2
p0,3 p1,3 p2,3 p3,3 p4,3 p5,3 p6,3 p7,3
p0,4 p1,4 p2,4 p3,4 p4,4 p5,4 p6,4 p7,4
p0,5 p1,5 p2,5 p3,5 p4,5 p5,5 p6,5 p7,5
p0,6 p1,6 p2,6 p3,6 p4,6 p5,6 p6,6 p7,6
p0,7 p1,7 p2,7 p3,7 p4,7 p5,7 p6,7 p7,7
Planar mode
p0,0 p1,0 p2,0 p3,0 p4,0 p5,0 p6,0 p7,0
p0,1 p1,1 p2,1 p3,1 p4,1 p5,1 p6,1 p7,1
p0,2 p1,2 p2,2 p3,2 p4,2 p5,2 p6,2 p7,2
p0,3 p1,3 p2,3 p3,3 p4,3 p5,3 p6,3 p7,3
p0,4 p1,4 p2,4 p3,4 p4,4 p5,4 p6,4 p7,4
p0,5 p1,5 p2,5 p3,5 p4,5 p5,5 p6,5 p7,5
p0,6 p1,6 p2,6 p3,6 p4,6 p5,6 p6,6 p7,6
p0,7 p1,7 p2,7 p3,7 p4,7 p5,7 p6,7 p7,7
DMM Mode 1

Figure 23: Selection of samples for calculating the predicted depth value in SDC (Planar mode, DMM Mode 1)

A depth lookup table is used to map the d_{orig} and d_{pred} to index values. The residual index to be transmitted to the decoder is given by

$$i_{resi} = I(d_{orig}) - I(d_{pred}), \tag{13}$$

where $I(.)$ denotes the depth lookup table. At the decoder side, the reconstructed mean depth value ($\hat{d}_{orig}$) is derived as

$$\hat{d}_{orig} = I^{-1}\left(I(d_{pred}) + i_{resi}\right), \tag{14}$$

where $I^{-1}(.)$ denotes the inverse depth lookup table. The mean residual signal is then derived as

$$\hat{d}_{resi} = \hat{d}_{orig} - d_{pred} \tag{15}$$

The reconstructed value $\hat{P}_{x,y}$ is derived by adding the residual signal on each prediction sample $P_{x,y}$:

$$\hat{P}_{x,y} = P_{x,y} + \hat{d}_{resi}. \tag{16}$$

The computed residual index i_{resi} is then coded with a significance flag, a sign flag and the magnitude of the residual index.

A concatenation of unary binarization and FL binarization is used for the magnitude of the residual. For residual index binarization, the concatenate binarization is constructed by the prefix part of the unary code and the suffix part of the FL code. Only one kind of context model for residual index coding is applied to the prefix part. Since the suffix part of the concatenate binarization uses the fixed length code and the occurrence probability of the suffix part is lower, context modelling are not used.

All above parameters described above are signalled in the depth intra parameters structure as described in section 2.3.4.1.2.

2.3.4.1 Depth Lookup Table

The depth lookup table utilizes the property of the depth map, that the full available depth range of 2^8 values is not utilized. Only a small amount of different depth levels occur due to strong quantization. In the encoder, a dynamic depth lookup-table is constructed by analysing a certain number of pictures (e.g. one intra period) of the input sequence. This depth lookup-table is used during the coding process to reduce the effective signal bit-depth of the residual signal.

2.3.4.1.1 **Construction of the depth lookup table**

To construct the DLTs, the encoder reads a pre-defined number of pictures from the input video sequence to be coded and scans all samples for available depth map values. During this process a mapping table is generated that maps depth values to valid depth values based on the original uncompressed depth map.

The Depth Lookup Table $D(.)$, the index Lookup Table $I(.)$, the Depth Mapping Table $M(.)$ and the number of valid depth values d_{valid} are derived by the following algorithm, that analyses the depth map D_t

1. Initialization
   - boolean vector $B(d) = FALSE$ for all depth values d
   - index counter $i = 0$
2. Process each sample position p in D_t for multiple time instances t:
   - Set $B\big(D_t(p)\big) = TRUE$ to mark valid depth values
3. Count number of $TRUE$ values in $B(d)$ → d_{valid}
4. For each d with $B(d) == TRUE$:
   - Set $D(i) = d$
   - Set $M(d) = d$
   - Set $I(d) = i$
   - $i = i + 1$
5. For each d with $B(d) == FALSE$:
   - Find $\hat{d} = \arg\min\left|d - \hat{d}\right|$ and $B\big(\hat{d}\big) == TRUE$
   - Set $M(d) = \hat{d}$
6. Set $I(d) = I(\hat{d})$

2.3.4.1.2 **DLT signalling**

Depth lookup tables are carried in picture parameter sets. Different methods are applied to code the DLT in base view and dependent views.

For base view, two methods are supported to encode DLTs: differential coding and bit map coding. One flag is signalled to indicate which method is used. In both methods, total number of values and the first value of the DLT table are signalled. In differential coding, difference between every two consecutive values of the DLT is encoded. First, maximum difference and minimum difference (minus 1) between two consecutive values of the DLT are encoded. Then, for each of the value excluding the first value in DLT, difference between it and its preceding value is subtracted from minimum difference and then coded. While in bit map coding, difference between last value and first value in DLT table is first coded. Then, for each value between the first value and the last value, a flag is coded to indicate whether it is included in DLT table.

For dependent view, instead of coding DLT table directly, a delta DLT table is coded using the DLT table coding method in base view. Let DLT table of current view and base view be DLT_c and DLT_b respectively, delta DLT table is generated as follows: for each value in [0, 255] (assuming bit depth of depth pixel is 8), if it is included in either DLT_c or DLT_b but not included in both DLTs, it is put into delta DLT; otherwise (i.e., it is included in both DLTs or excluded in both DLTs), it is not put into delta DLT.

2.3.4.1.3 **Extension of DLT to other depth intra modes**

Particular intra modes, such as depth modeling modes, the DC, vertical and horizontal modes in HEVC could futher utilize the advantage of DLTs. If the current mode is a mode among the predefined particualr modes, residual indexes (differences between the index of the original pixel and the index of predictor through DLT) are generated by DLT. Otherwise, residual values (the difference between the original pixel and the predictor) are constructed. After constructing the residaual data, transform and quantization are performed and then, quantized transform coefficients are entered into entropy coder. For each CPV, a bin string consisting of the absolute value and the sign is transmitted. The sign is coded as a single bin, and the absolute value is coded using a truncated unary code (with 13 bins in the unary part and an exponential golomb code suffix).

2.3.5 Unified signalling of depth intra modes

A structure (depth_mode_parameters) is signalled for intra prediction units, that has minimum interaction with the HEVC syntax elements from the base specification.

The depth intra modes, including depth modelling modes, segment-wise DC coding, and the chain code mode are signalled by two syntax elements in the depth_mode_parameters syntax structure. Firstly, a flag, (depth_intra_mode_set_indication_flag) is used to indicate a set of depth intra modes. Secondly, another syntax element,(depth_intra_mode) is signalled to indicate a depth intra mode within a set of modes defined by the PU size and the depth_intra_mode_set_indication_flag.

The depth_intra_mode_set_indication_flag is coded in bypass mode and depth_intra_mode is coded using truncated unary binarization method.

Based on the value of the derived depth intra mode (DepthIntraMode), other additional information is signalled (as. e.g. wedgelet parameters and delta CPVs as described above).

Possible depth intra modes are shown in table Table 3.

Table 3: Depth intra modes

DepthIntraMode	Associated name	Section
0	INTRA_DEP_SDC_PLANAR	2.3.4
1	INTRA_DEP_NONE	normal HEVC intra modes
2	INTRA_DEP_SDC_DMM_WFULL	2.3.4
3	INTRA_DEP_DMM_WFULL	2.3.3.1
4	INTRA_DEP_DMM_CPREDTEX	2.3.3.2
5	INTRA_DEP_DMM_WPREDTEX	0
6	INTRA_DEP_CHAIN	1.1.1.1

2.3.6 Unification signalling of delta CPVs

In 3D-HEVC, the enhanced depth intra modes (e.g., several enhanced depth intra modes, including Depth Modeling Modes and Simplified Depth Coding mode) partition a depth PU into one or two segments, and each segment is coded together with an optional delta CPV value (or delta DC value). To simplify the delta CPV coding, the same syntax elements and contexts are utilized regardless the prediction mode. More specifically, one flag is firstly coded to indicate whether all the delta DC values of each partition are equal to 0. If there is at least one non-zero delta DC values for one partition, the magnitude of the detal DC value and the sign flag (when the magnitude is unequal to 0) are further coded for each partition.

2.3.7 Motion parameter inheritance

The basic idea behind the motion parameter inheritance (MPI) mode is that the motion characteristics of the video signal and its associated depth map should be similar, since they are both projections of the same scenery from the same viewpoint at the same time instant. Therefore, in order to enable efficient encoding of the depth map data, a texture candidate for the merge mode in depth coding that allows the inheritance of motion parameters from the texture signal has been introduced. The derivation of the texture candidate for depth is depicted in figure Figure 24. The motion parameters of the corresponding texture block are added as candidate to the merge list of the PU in the depth picture. When MPI is enabled for one depth view, the corresponding merge list size is extended by 1, i.e., up to 6 merge candidates may be added to the merge candidate list.

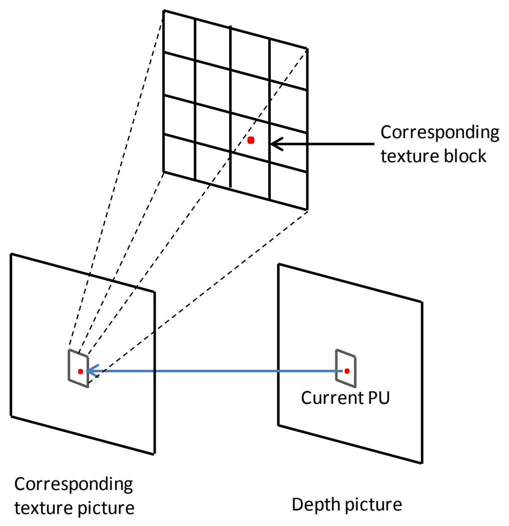


Figure 24: The derivation of corresponding texture block

Since the motion vectors of the video signal have quarter-sample accuracy, whereas for the depth map signal only full-sample accuracy is used, in the inheritance process the motion vectors are quantized to their nearest full-sample position. In addition, the inherited reference picture shall be the one with the same POC and ViewId as the reference picture of the collocated block in the texture picture. If there is no reference picture in the reference lists that satisfies this condition, such a candidate is treated as invalid and it is not inserted to the merge candidate list.

2.3.8 Modified merge candidate list construction process

Similar to the texture coding, the merge candidate list size for depth views is also extended by 1 when either MPI or depth inter-view motion prediction is enabled.

When MPI is enabled, the texture merging candidate derived from co-located texture block is added.

When depth inter-view motion prediction is enabled, the candidate list of motion parameters is extended by a motion parameter candidate for MCP (IvMC) that is obtained using the PU-level inter-view motion prediction, as described in 2.2.4.1.1. Note, for depth coding, a different method from NBDV is used to derive the disparity vector which is based on neighbouring reconstructed pixels. The disparity vector derivation process is described in 2.3.8.1. The derived disparity vector is also used to a motion parameter candidate for DCP, in the same way as described in 2.2.4.1.2. In addition, the shifted candidate, as described in 2.2.4.1.3 is generated.

2.3.8.1 Disparity vector generation for depth views

For generating an inter-view motion candidate, first a disparity vector has to be calculated to identify the corresponding block in the reference view (see Fig. 2). The NBDV method used for deriving a disparity vector in texture views cannot be used for deriving a disparity vector for the depth blocks, as most of the neighbouring depth blocks may be intra coded, i.e., with high probability the neighbouring blocks may not contain a disparity motion vector. We therefore propose to derive a disparity value for each coding unit (CU) from the neighbouring reconstructed depth samples. The neighbouring sample positions that are adjacent to the corners of the current CU block are used. More specifically, chosen ones are above-left, above-right and bottom-left sample positions of the current block. These positions are marked in red in Figure 26. From the respective depth values in these neighbouring samples, a single depth value is calculated as follows:

$$Depth = (5*D[xC-1][yC-1] + 5*D[xC-1][yC + 2N-1] + 6*D[xC + 2N-1][yC-1] + 8)>>4 \tag{17}$$

Here, D[x][y] represents the reconstructed depth value at location (x, y), (xC, yC) represents the top-left corner of the current CU of size 2Nx2N, and >> represents the right shift operator. Note that the weighted average as shown in the above equation is used mainly to avoid the division by 3.

In special cases, for example, at top-left corner of the image, all the neighbours are not available; the depth value is set to zero in this case. Also, at the image boundaries (except at the top-left corner of the image) either above-right or bottom-left sample position is available for the current CU block. In such cases, the depth value is set equal to the reconstructed value of the available neighbour without performing weighted averaging. The calculated depth value is then converted into a disparity vector, that is denoted here as ***DV***. This disparity vector ***DV*** is used to set the disparity vector to all the PU block's within the CU, i.e., all the PU blocks within CU share the same disparity vector ***DV***.

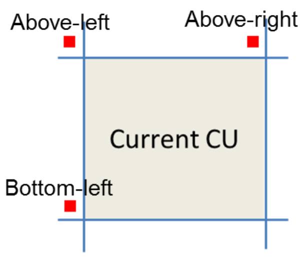


Figure 25: Derivation of a disparity vector from three (red) neighboring reconstructed depth samples

2.3.9 Simplified inter-mode depth coding

Simplified inter-mode depth coding (SIDC) extends the basic idea of SDC to inter mode depth coding. It provides an alternative residual coding method and only encodes one DC residual value for a PU. Transform and quantization are skipped, and no additional residual like transform tree is required. Whether SIDC is used is signalled in the general coding unit parameters at CU level. For SIDC coded CU, one DC residual value is signalled for each PU and is used as residual for all samples in the PU.

To decrease the signalling bits on SIDC mode, only non-skip CU is allowed to apply SIDC. Furthermore, to avoid possible overlap between SIDC mode and skip mode, SIDC mode is applied only when DC residual of each PU within the CU is non-zero. DC residual of a PU is calculated as the average of the difference between original sample value and prediction sample value of all samples with the PU. Because only DC difference between original block and prediction block is signalled, to compensate the AC difference, mean-removed motion estimation is employed for depth inter mode coding.

This coding tool is enabled/disabled by referring to one flag signalled in each depth view component.

2.3.10 Depth Quadtree Prediction

Depth quadtree prediction performs a prediction of the depth quadtree from the texture quadtree. It is applied in inter slices that do not belong to random access pictures. The partitioning of the depth is limited to the same level as the partitioning of the texture. For a given CTU, the quadtree of the depth is linked to the collocated CTB quadtree in the texture, so that a given CU of the depth cannot be split more than its collocated CU in the texture. Moreover, when the texture is split in 2NxN (or Nx2N), partitioning to 2NxN, Nx2N, or NxN is not performed for depth. The possible partitioning is depicted in Figure 26. Corresponding split flags and partition sizes for depth depending on the split flags and partition sizes of texture are summarized in Table 4.

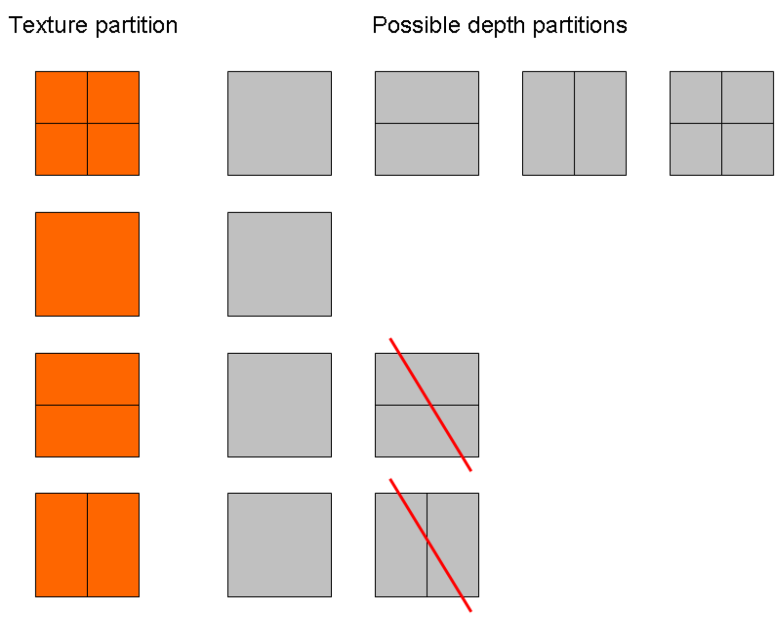


Figure 26: Texture partitions and corresponding possible depth partitions

Table 4: Split flags and partition sizes of depth depending on split flags and partition sizes of texture

Texture SplitFlag	Texture PartSize	Depth SplitFlag	Depth PartSize	Residual Depth SplitFlag	Residual Depth PartSize
1	-	1	-	1	-
1	-	0	0, 1, 2 or 3	0	0, 1, 2 or 3 resp.
0	0,1, 2 or 3	0	0	-	-
-	0, 1 or 2	-	0	-	-
-	3	-	0, 1, 2 or 3	-	0,1, 2 or 3 resp.

SplitFlag: 0 = no split, 1 = split; PartSize: 0 = 2Nx2N, 1 = Nx2N, 2 = 2NxN, 3 = NxN.

2.4 Motion compression

To reduce the buffer size and memory bandwidth, the motion data is compressed into 1/4 resolution after encoding/decoding of each picture and then further compressed into 1/16 resolution after encoding/decoding of all the pictures within the same AU.

To be more specific, for each 8x8 unit, the motion parameters of the top-left 4x4 block are used as the representative motion parameters. Therefore, the motion data are stored in a motion data buffer of quarter size after each picture is coded. After all pictures within the same AU are coded, the same procedure is then performed to the motion parameters that are already compressed. After the second motion data buffer reduction, motion data are stored in a motion data buffer of 1/16 size..As can be seen in Figure 27, since the storage is reduced, the bandwidth for writing and reading motion data can also be reduced by the proposed scheme.

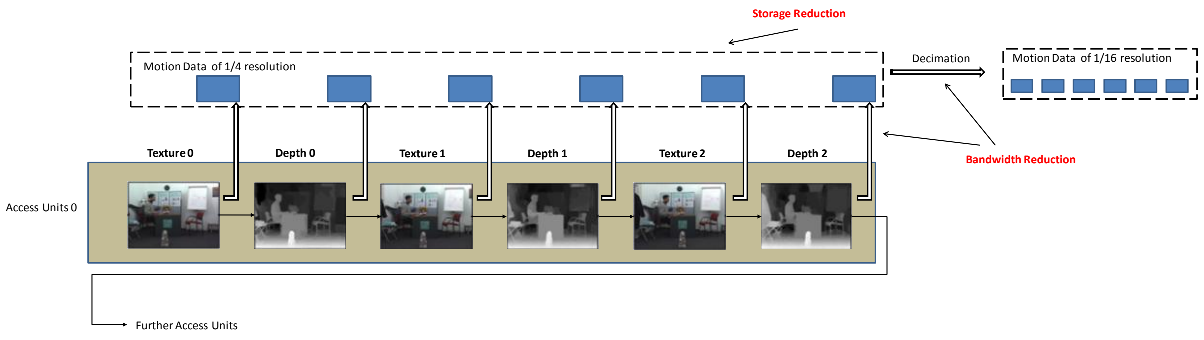


Figure 27: Progressive motion compression for motion data buffer reduction.

2.5 Encoder Control

For mode decision and motion estimation, a Lagrangian technique by which a cost measure $D + \lambda \cdot R$ is determined for each candidate mode or parameter, and the mode or parameter with the smallest cost measure is selected. D is the distortion that is obtained by coding the considered block in a particular mode or with a particular parameter, R is the number of bits that are required for representing a block in a given mode or that are required for coding a given parameter, and λ is the Lagrangian multiplier that is derived based on the used quantization parameter. As measure for the distortion, the sum of squared differences (SSD) or the sum of absolute differences (SAD) between the original and the reconstructed sample values is used (for the coding of depth maps this measure was modified as described below).

For the coding of depth maps, basically the same decision process is used. However, the distortion measure has been replaced with a measure that considers the distortion in synthesized intermediate views. This technique is described in the following subsection.

2.5.1 View Synthesis Optimization

The geometry information given by depth data is exploited only indirectly in the rendering process. Hence, the lossy coding of depth data causes distortions in the synthesized intermediate views. The depth map itself is not visible for a viewer. The efficiency of depth coding is improved by considering this property. As a consequence, the distortion measure for the mode decision process for depth maps is modified in a way that a weighted average of the synthesized view distortion and the depth map distortion. To obtain a measure of the synthesized view distortion, two different metrics are applied in RDO.

The first metric, discussed in section 2.5.1.1, is the synthesized view distortion change (SVDC). The computation of the SVDC requires the usage of rendering functionalities in the encoding process. Since computational complexity is a critical factor in distortion calculation, a method, which is also referred to as renderer model, has been utilized that allows minimal re-rendering of parts of the synthesized view that are affected by a depth distortion. For this, a special renderer is included in the encoder, which supports the basic functionalities, shared by most rendering approaches, like sub-sample accurate warping, hole filling and view blending.

The second metric, presented in section 2.5.1.2, is a model based synthesized view distortion estimation without rendering. Basic idea of this metric is to derive an estimate for the synthesized view distortion by weighting the depth distortion with a factor derived from the absolute value of the derivation of texture view in horizontal direction.

The integration of both metric in the encoder control is presented in sections 2.5.1.4 and 2.5.1.5.

2.5.1.1 Synthesized View Distortion Change (SVDC)

2.5.1.1.1 Definition of the SVDC

Since the encoding algorithm operates block-based, the mapping of depth distortion to the synthesized view distortion must be block-based as well. Moreover, the sum of partial distortions (of sub-blocks) must be equal to the overall distortion of a block in order to enable an independent distortion calculation for all partitions of a subdivided block, as hierarchical block structures are used in HEVC.

A relationship between a depth map s_D and a synthesized texture s'_T is created by the used view synthesis approach. However, disocclusions and occlusions prevent a bijective mapping of the distorted areas in depth maps to distorted areas in the synthesized views. For example, areas in the synthesized view, which depend on depth data of a considered block, can become visible due to the distortions in other depth blocks; or vice versa, the distortion of a depth block has no effect on the synthesized view, since the block is occluded there. Hence, an exact mapping between the distortion of a block of the depth data and an associated distortion in the synthesized view is not possible considering only the depth data within a currently processed block.

For resolving this issue, the change of the overall distortion in a synthesized view depending on the change of the depth data within a block B is determined, while simultaneously also considering depth data outside the block B. For this purpose, the synthesized view distortion change (SVDC) is defined as distortion difference ΔDbetween two synthesized textures s_T' and $\tilde{s}_T$,

$$\Delta D = \widetilde{D} - D = \sum_{(x,y)\in I} \left[\tilde{s}_T(x,y) - s'_{T,Ref}(x,y)\right]^2 - \sum_{(x,y)\in I} \left[s_T'(x,y) - s'_{T,Ref}(x,y)\right]^2 \tag{18}$$

$s'_{T,Ref}$ denotes a reference texture rendered from original video and depth data. I represents the set of all samples in the synthesized view. To illustrate how the textures $s_T'$and $\tilde{s}_T'$ are obtained, the SVDC definition from eq. (18) is also depicted in Figure 28. s_T' denotes a texture rendered from a depth map s_D consisting of encoded depth data in already encoded blocks and original depth data in the other blocks. The current block B, for which the distortion has to be computed, contains original depth data as well. For the synthesis of the texture $\tilde{s}_T'$ a depth map$\tilde{s}_D$is used that differs from the depth map $s_D$in that it contains the distorted depth data also for the current block B.

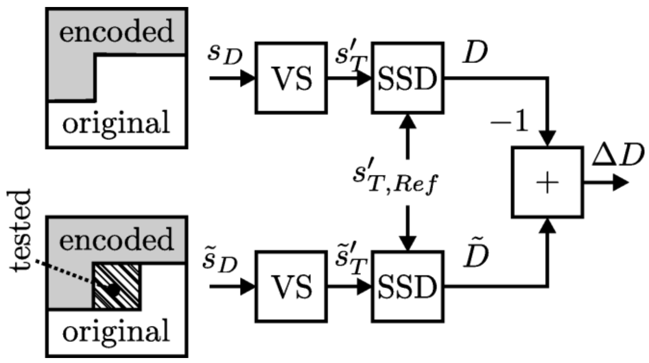


Figure 28: Definition of the SVDC related to the distorted depth data of the block B depicted by the hatched area in the bottom branch; VS denotes the view synthesis step and SSD stands for sum of squared differences.

The SVDC definition above is motivated by three reasons. First, an exact distortion measure is provided, therefore the overall distortion of the synthesized view and thereby disocclusions and occlusions are considered. Second, the measure is related to a block and third partial distortions are additive. For the latter two reasons, the change of the synthesized view distortion caused by a change of a depth block is employed instead of the total synthesized view distortion itself.

Figure 28 shows the SVDC definition for the extrapolation of virtual views from one input view only. However, the encoder side view synthesis algorithm supports also the interpolation of the texture s_T' from a left and a right view. Hence, rendering requires a left $s_{D,l}$ and a right $s_{D,r}$ depth map. To extend the SVDC computation to this two view case, the original depth map of the second view can be used when encoding the first depth map. Subsequently the first already encoded depth map can be utilized for the SVDC computation when encoding the second depth map.

2.5.1.1.2 **Efficient Computation of the SVDC**

A straightforward approach to compute the SVDC would be the direct implementation of eq. (18). However, this would require the complete rendering of the synthesized textures s_T' and $\tilde{s}_T'$ and a rendering of a whole view is computational too complex to be feasible in a rate-distortion optimization process. To overcome this problem, a method which enables a fast computation of the SVDC is integrated in the encoder.

Renderer Model

The renderer model provides three basic functionalities to the encoder: Initialization, partial re-rendering, and SVDC calculation.

- The **initialization** of the renderer model is carried out before the encoding of a depth map is started. In the initialization process, the complete synthesized view is rendered using the original input depth maps and the input textures. The input depth maps are stored as the renderer models depth states $s_{D,l}$ and $s_{D,r}$ and the rendered view as the synthesized view state s_T'. Intermediate variables used in the rendering process are also stored to enable a fast re-rendering.
- **Partial re-rendering** is carried out to update the renderer model when the encoding of a block B is finished and the final depth data for the block is known. For this purpose, the reconstructed depth data and the position of

block B are signalled to the renderer model. The renderer model changes the block in the depth state $s_{D,l}$ or $s_{D,r}$ from original to coded data and re-renders only local parts of the synthesized view state s'_T and the intermediate variables that are affected by the change of the depth data. Thus, the renderer model is transferred to a state that is required to compute the SVDC for blocks of the depth data encoded subsequently.

- For the **computation of the SVDC**, the position and the depth data of a block B to be tested are provided to the renderer model. The renderer model then computes the SVDC as defined in eq. (18). Here, re-rendering followed by the computation of the sum of squared distortions SSD is carried out. However, instead of considering all positions $(x, y) \in I$ again only positions affected by the depth change are considered. Note that the re-rendering carried out here does not modify any state variables of the renderer model. Hence, the SVDC can be computed for multiple depth candidates successively without the need to re-render with original data in block$B$.

Re-Rendering and Error Calculation Algorithm

The main objective of the algorithm is a computational low complex distortion calculation or state transition, hence a low complex re-rendering of the parts of the synthesized view that are affected by a depth change in one of the input depth maps.

Conventional view synthesis consists of multiple steps such as warping of the input samples, interpolation at sub sample positions, blending with a second view obtained similarly, and hole filling. Typically these steps are executed as independent algorithms that are applied successively using the results of the previous step. To enable fast re-rendering of only parts of the synthesized view, all steps are combined in single algorithm that can be applied sample-wise to the input depth map. This allows a region-wise processing of the depth map, and thus an update of related regions in the synthesized view.

This process is illustrated in Figure 29 for an example for rendering from a left view to the right. Rendering is applied row wise, hence all depicted signals represent one row of input, intermediate, or output data. The single signals are from bottom to top: the left input texture $s_{T,l}$, a shifting chart, the texture synthesized from left $s'_{T,l}$the texture synthesized from right $s'_{T,r}$, the blended textures'_T, and the reference texture$s'_{T,Ref}$. The arrows denote the relationship between the single samples or sample positions of the signals. Dots shown in the shifting chart represent samples from the input view. Their horizontal position is equal to their position x'in the synthesized view. The vertical position shows their disparities. Since the depth is monotonically decreasing with increasing disparity, the top-most samples in the chart are the samples closest to the camera. Hence, it can be seen from the shifting chart which samples are occluded in the synthesized view.

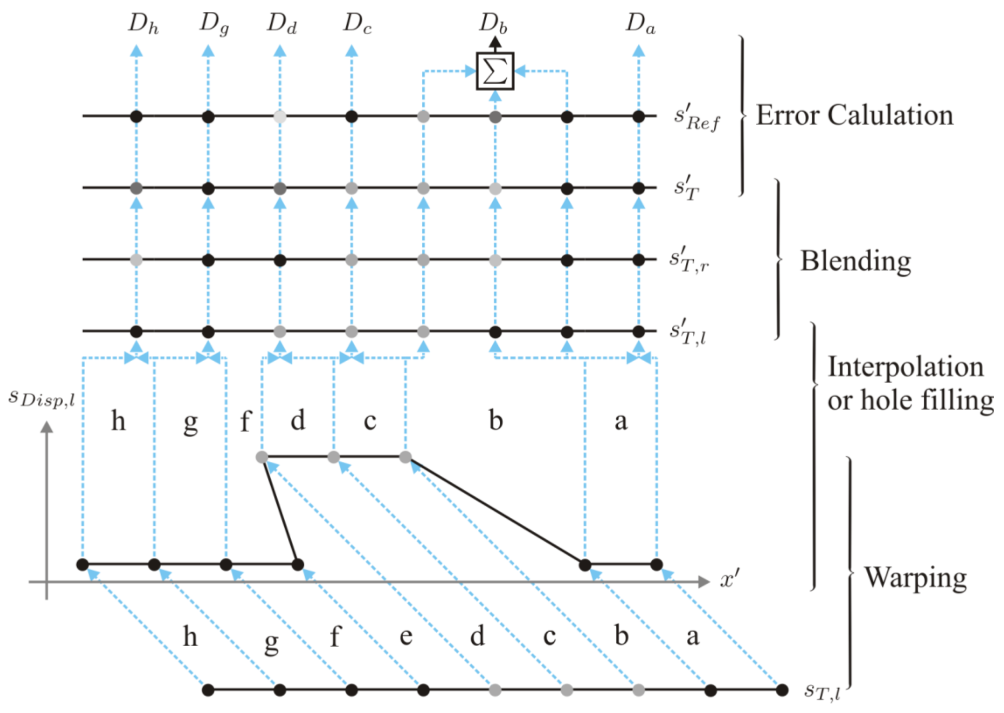


Figure 29: Example for the dependencies between input, intermediate and output signals of the rendering or error calculation step.

While a conventional view synthesis approach would carry out the single steps depicted from bottom to top for all samples in the intervals (a) to (g), the method supports an interval-wise processing. Hence, all steps are first conducted for interval (a) before continuing with interval (b). Re-rendering and error calculation are carried out by iterating only once over the input depth samples. If only the view synthesis distortion is calculated there is no need to store intermediate results in the state of the renderer model.

The boundaries of an interval in the output view are defined by the warped positions x'_s and $x'_e$of two neighboring input view samples at positions $x_s$and x_e. For warping, disparities are computed from the depth map as described in the beginning of sec. 2. Subsequently to the calculation of the interval boundaries, processing continues with interpolation, disocclusion handling, or occlusion handling:

- **Interpolation** is carried out in non-occluded ranges that are not disoccluded, as for example in the intervals (a, c, d, g, h). The accuracy of the warping is higher than the accuracy given by the sampling rate of synthesized view; hence an interpolation at the full sample position x'_{FP} located between the interval boundaries $x'_s$and $x'_e$is carried out. For this, samples from an up-sampled version of the input texture $\hat{s}_{T,l}$ are mapped to the interpolation positions x'_{FP}in the synthesized view $s'_{T,l}$. The position $\hat{x}$in the up-sampled view is derived from the distance of the interpolation position to the interval boundaries:

$$\hat{x} = 4 \cdot \left(\frac{x'_{FP} - x'_s}{x'_e - x'_s} + x_s \right) \qquad (19)$$

  The up-sampled view $\hat{s}_{T,l}$is created in the initialization step by interpolating the input texture with quarter-sample accuracy using the FIR-filters specified for motion-compensated interpolation in HEVC.

- **Disocclusions**: If the width of the warped interval $x'_e - x'_s$is greater than two times the width of the sampling distance, as for example for interval (b), a disocclusion is assumed in the synthesized view. Instead of interpolation, hole filling is carried. For this purpose, the samples in the interval are set equal to the value of the sample belonging to the right interval boundary $s_{T,l}(x_e)$ (which belongs to the background). If the leftmost full sample position within the interval is close to the left interval border, it is assumed that it belongs to the foreground and it is set equal to the value of the left interval boundary $s_{T,l}(x_s)$. Note, that the positions of disoccluded and filled samples are stored as additional information in the a filling map $s'_{F,l}$.

- **Occlusions:** Whether an interval is entirely occluded in the synthesized view, as for example interval (f), is determined by detecting if the interval boundaries are reversed ($x'_e < x'_s$), hence no complex z-buffering is required. To derive whether other samples left to interval (f) are occluded, the rendering process stores the position of the foreground edge. This stored position is then be utilized when processing the next intervals, for example interval (e), to determine which parts of theses intervals are occluded. If re-rendering does not start at the right image border, the position of the last foreground edge is recovered by carrying out a search to the right of the changed depth samples.

Sample values derived from interpolation or hole filling $s'_{T,l}$, are instantly combined with the texture sample values from a second view $s'_{T,r}$ synthesized the same way and stored as intermediate variable in the renderer model. The result is the sample value that is used in the final synthesized view s'_T.

The rendering model supports two different configurations. In the first configuration, a rendering process is considered that renders intermediate views using both surrounding actually coded views. The second configuration considers rendering processes by which an intermediate view is rendered mainly from one coded view; the other coded view is only used for rendering areas that are not present in the preferred coded view.

In the first configuration of the renderer model, the blending process is similar to that implemented in the VSRS software. Note that, although not depicted in Figure 29, a depth map $s'_{D,l}$ is rendered from $s_{D,l}$, when rendering $s'_{T,l}$, using full sample accuracy. This depth map is used in the blending step. The decision how blending is carried out depends on the filling of $s'_{F,l}$or $s'_{F,r}$ and the rendered depth maps $s'_{D,l}$and $s'_{D,r}$. While $s'_{F,l}$and $s'_{D,l}$have been obtained in the rendering process carried out before, $s'_{F,r}$ and $s'_{D,r}$ are stored as intermediate variables in the renderer model. The rules for determining the blended sample value $s'_T(x, y)$ from $s'_{T,l}(x, y)$ and $s'_{T,r}(x, y)$ are specified in the following:

- If the position (x, y) is disoccluded (as indicated by the filling map) in only one view, the sample value from the other view is used.
- Otherwise, if the position (x, y) is disoccluded in both views, the backmost sample value is used.
- Otherwise, if the depth difference retrieved from $s'_{D,l}(x, y)$ and $s'_{D,r}(x, y)$ is greater than a threshold, the front sample is used.
- Otherwise, a weighted average of $s'_{T,l}(x, y)$ and $s'_{T,r}(x, y)$, with a higher weight for the view that is closer to the virtual view position, is used.

For the second configuration of the renderer model, the intermediate view is mainly rendered from one view and only holes are filled from the other view. If assuming that$s'_{T,l}$ is the main view, the rules to determine the sample value $s'_T(x,y)$ from $s'_{T,l}(x,y)$ and $s'_{T,r}(x,y)$ are specified in the following:

- If $\mathrm{s'_{F,l}(x,y)}$indicates that there is no disocclusion at $\mathrm{s'_{T,l}(x,y)}$, the sample value$\mathrm{s'_{T,l}(x,y)}$ is used.
- Otherwise, if $\mathrm{s'_{F,r}(x,y)}$ indicates that there is a disocclusion at $\mathrm{s'_{T,r}(x,y)}$, the sample value $\mathrm{s'_{T,l}(x,y)}$ is used.
- Otherwise, the average of $\mathrm{s'_{T,r}(x,y)}$and $\mathrm{s'_{T,l}(x,y)}$ is used.

If only partial re-rendering is carried out, the result s'_T and all intermediate results are stored after the combination step and the processing of the interval is stopped. Otherwise, if the SVDC is determined, the distortion of the calculated value s'_T is computed by comparing it to the reference$s'_{T,Ref}$ in the next step.

To obtain the synthesized view distortion change the single intervals are rendered from right to left and the related distortions are summed up continuously. Moreover, and that is actually not depicted in Figure 29, the old per sample distortions of samples in the changed intervals are subtracted.

The renderer model only re-renders those parts of the synthesized view that are affected by the considered depth change. It has to be considered that in some cases not only the intervals related to the changed depth values must be re-rendered, but also some neighbouring intervals. A reason is that neighbouring intervals that are occluded before a depth change can become visible after the depth change. The algorithm detects such cases and continues rendering, until all change samples in the synthesized view are updated. The detection is carried out while warping by also considering the old shifted sample positions as they had been prior to the depth change and storing the left-most old position.

Chroma channels of the synthesized view are rendered together with the luma channel and are stored in the same resolution as luma. For this, up sampled versions of the chroma channels are created in the initialization step, which are later used for interpolation as described above. The sampling rate is increased by a factor of eight in horizontal direction and a factor of two in vertical direction using the interpolation FIR-filters that are specified for motion-compensated prediction in HEVC. However, the total distortion is obtained by a weighted sum of luma SVDC and chroma SVDC with a weight of 1 for luma and a weight of ¼ for each of the two chroma channels.

Early skip of SVDC computation

To increase the processing speed of the VSO algorithm the SVDC calculations is skipped for lines of a block for that the distortion of the disparity vector is zero. This means that if distorted depth and original depth are mapped to the same disparity vector for all samples in a line of the depth block the SVDC calculation is not carried out and the distortion is assumed to be zero. .

2.5.1.2 Model based synthesized view distortion estimation without rendering

The distortion of depth maps does not linearly affect the synthesis distortion, and the impact of depth map distortions varies according to the corresponding texture information. For example, the same depth distortions on textured and textureless regions lead to different synthesis distortions.

In a conventional video coding system, one commonly used distortion function is the sum of squared differences (SSD), which is defined between original and encoded depth block as

$$D_{depth} = \sum_{(x,y)\in B} |s_D(x,y) - \tilde{s}_D(x,y)|^2 \tag{20}$$

where $s_D(x,y)$ and $\tilde{s}_D(x,y)$ indicate the original and reconstructed depth map, respectively, and (x,y) means the sample position in a (macro-) block B. However, the conventional SSD metric is not an good estimate of the synthesized view distortion. Instead, the following view synthesis distortion (VSD) metric provides an better estimate by that weighting the depth distortion D_{depth} with the sum of absolute horizontal texture gradients:

$$VSD = \sum_{(x,y)\in B} \left(\frac{1}{2}\cdot\alpha\cdot|s_D(x,y) - \tilde{s}_D(x,y)|\cdot[|\tilde{s}_T(x,y) - \tilde{s}_T(x-1,y)| + |\tilde{s}_T(x,y) - \tilde{s}_T(x+1,y)|]^2\right) \tag{21}$$

$\tilde{s}_T$ indicates the reconstructed texture, and α is proportional coefficient determined by

$$\alpha = \frac{f\cdot L}{255}\cdot\left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right) \tag{22}$$

with f denoting the focal length, L denoting the baseline between the current and the rendered view, Z_{near} and Z_{far} representing the values of the nearest and farthest depth of the scene, respectively.

2.5.1.3 Depth fidelity term

When encoding using the synthesized view distortion change or estimate only, the depth fidelity is strongly distorted. In order to preserve the depth fidelity the distortion measure used in RDO is computed by a weighted average of the synthesized view distortion or the estimated synthesized view distortion and the depth distortion. The distortion D used in RDO for depth maps is given by

$$D = w_{synth} D_{synth} + w_{depth} D_{depth} \tag{23}$$

with D_{synth} denoting the synthesized view distortion change or estimate, D_{depth} denoting the distortion of the depth map itself (i.e. SAD or SSD), and w_{synth} and w_{depth} denoting the weights for the two distortion terms.

2.5.1.4 Integration of distortion metrics in the Encoder Control

To enable rate-distortion optimization using the SVDC, the described renderer model is integrated in the encoding process for depth data. For this, the conventional distortion computation is replaced with computation of the weighted average of depth distortion and SVDC in distortion computation steps related to the mode decision, coding unit (CU) partitioning, motion parameter inheritance and merging. Note that for updating the renderer model used for the SVDC calculation re-rendering is carried out when a final decision on the coding mode is taken by the encoder control.

In order to reduce the computational complexity the weighted average of SVDC and depth distortion in not used for all encoding decisions. A weighted average of VSD and depth distortion is used for intra-mode pre-selection and residual quadtree partitioning. For motion estimation and rate-distortion optimized quantization the conventional SSD of depth data is used.

2.5.1.5 Adaptation of the Lagrange Multiplier

The usage of the synthesized view distortion in the rate-distortion decisions requires the adaptation of the Lagrange multiplier λ to obtain optimized coding results. This adaptation is carried out by adjusting the Lagrange multiplier using an additional scaling factor l_s depending on the QP of the coded video. The factor enables an adjustment of video/depth rate allocation. As alternative a constant factor can be used.

The computation of rate-distortion cost J has been modified to

$$J = D + l_s \cdot \lambda \cdot \mathrm{R} = \mathrm{D} + \lambda_l \cdot \mathrm{R} \tag{24}$$

with D denoting the weighted average of depth and synthesized view distortion, l_s denoting a scaling factor, and R denoting the rate for the current coding mode.

2.5.2 Zero residual coding for depth intra CUs

In the rate distortion optimized coding of inter blocks a decision between coding with and without residual is carried out. For depth coding this principle is extended to intra coded blocks that are not part of slices using intra prediction only. Therefore, the residual is set to zero by the encoder. No additional signalling is used.

2.5.3 Optional Encoder Control using a depth quadtree limitation

In the encoding process a given CTB is split into smaller CUs, based on RD optimized decisions. A corresponding quadtree (QT) is obtained for the texture, and another one for the depth. This tool prevents the encoder from making full investigation of every possible QT configuration for the depth.

The tool forces the encoder to limit the partitioning of the depth at the same level as the partitioning of the texture. For a given CTU, the quadtree of the depth is linked to the collocated CTB quadtree in the texture, so that a given CU of the depth cannot be split more than its collocated CU in the texture.

This encoder restriction results in encoder runtime saving for the depth.

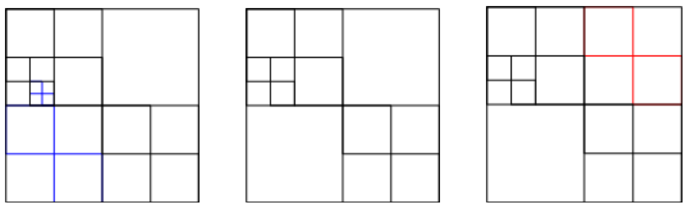


Figure 30: Example of a CTB QT partitioning for the texture (left), allowed collocated depth CTB QT partitioning (centre), and disallowed collocated depth CTB QT partitioning (right).

Figure 30 illustrates this principle. On the left a CTB QT partitioning for the texture is represented. In the centre, the collocated CTB in the depth is represented. This QT partitioning is allowed because it is, CB by CB, coarser than the corresponding texture CB. On the right, another example of possible collocated CTB in the depth is represented. This QT partitioning is disallowed because one CB is more partitioned than the texture (red lines).

2.5.4 Optional Encoder Control for Renderable Regions in Dependent Views [Not in CTC]

As an optional encoding technique, a mechanism is integrated by which regions in dependent views that can be rendered based on the transmitted independent view and the associated depth maps are identified. These regions are encoded by employing a modified cost measure, which mainly considers the required bit rates. After decoding, the renderable regions can be identified in the same way as in the encoder and replaced by rendered versions.

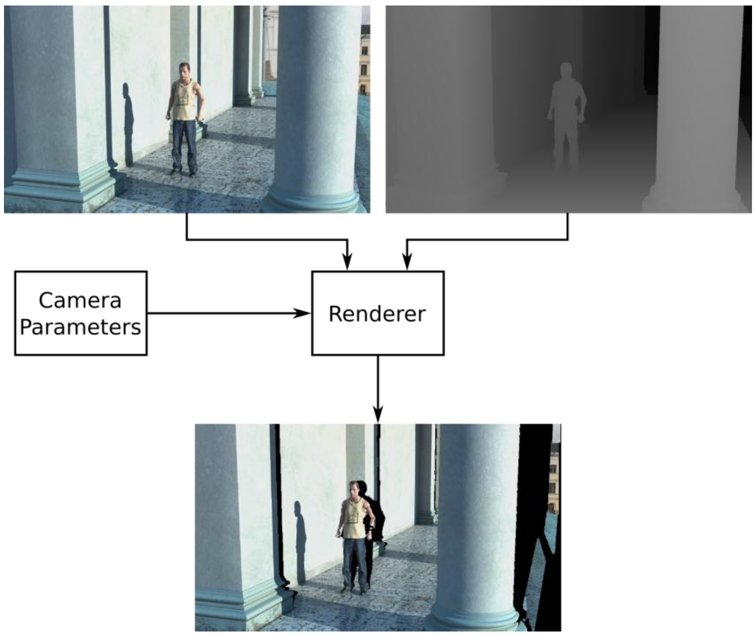


Figure 31: Rendering from a left camera position to a right camera position using depth maps.

The encoder identifies regions in the current picture that can be rendered from pictures of the same time instance in a reference view based on the reconstructed depth maps of the reference view (see Figure 31). During the encoding process, the encoder checks for every CU, if all samples within that CU can be rendered. If all samples can be rendered, no residual is transmitted for this CU. In our HEVC-based codec, this means that for inter prediction the *no_residual_data_flag* for the CU is set equal to 1 or for intra-prediction the coded block flag of the TUs within the CU is set equal to 0. It should be noted that no syntax change is applied; only the encoder decision is modified.

Due to the quadtree structure in HEVC, the rate-distortion (RD-) costs are compared between different granularities of possible block subdivisions for the R-D optimization. Rendering artefacts have a different impact on the subjective image/video quality perception than coding artefacts and cannot be compared using conventional measurements, such as MSE or PSNR. Samples in renderable regions are not taken into account for calculating the distortion term in the R-D optimized encoder decisions. In Figure 32, the right image shows a block subdivision that is one level deeper than the ones in the left image. The grey area labels the samples that can be rendered and that are therefore not considered in the calculation of the distortion. Thus, for example, the upper left block in the right image is not considered at all. Hence, the costs being compared are $D_0 + \lambda R_0$(left block subdivision) against $\mathrm{D}_1 + \lambda\,\mathrm{R}_1 +\ \mathrm{D}_2 + \lambda + \mathrm{D}_3 + \lambda\,\mathrm{R}_3 + \mathrm{D}_4 + \lambda\,\mathrm{R}_4$ (right block subdivision), where the distortions are only calculated based on the white shaded samples. E.g., the distortion of block $B_1$is $D_1 = 0$. By this modification, blocks for which a subblock can be rendered are not automatically split, but also the entire block may be coded using a conventional coding mode if this improves the overall coding efficiency.

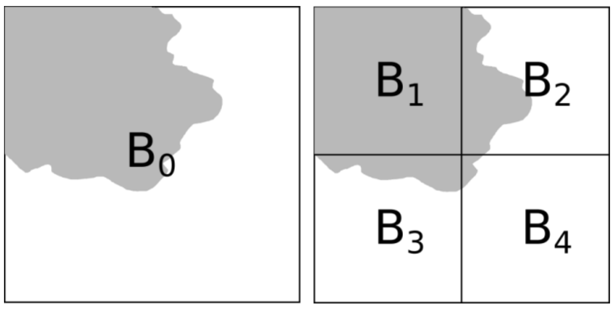


Figure 32: Distortion calculation on different tree depths. Renderable samples (gray shaded) are not taken into account.

For renderable blocks, the Lagrange multiplier λ is scaled by a factor $s > 1$ and the calculation of the R-D costs is changed from $\mathrm{C = D + \lambda R}$ to $\mathrm{C = D + s\lambda R}$.

3. View Synthesis Algorithms

In the following, two view synthesis algorithms are described. Sec. 3.1 describes the fast 1-dimensional view synthesis algorithm that is part of the HEVC-based 3DV software. It is also referred to as "VSRS 1D fast mode". In sec. 3.2, an alternative view synthesis algorithm is described. This algorithm is also referred to as "VSRS" and was developed during the 3DV exploration experiments.

3.1 Fast 1-D View Synthesis (VSRS 1D Fast Mode)

An overview of the view synthesis method is depicted in Figure 33. The method supports the interpolation of a synthesized view form a left $s_{T,l}$ and right $s_{T,r}$ texture with corresponding depth maps $s_{D,l}$ and $s_{D,r}$. For this, two texture $s'_{T,l}$ and $s'_{T,r}$ are extrapolated from the left and the right view at the position of the virtual view. Subsequently, the similarity of $s'_{T,l}$ and $s'_{T,r}$ is enhanced before combining them to synthesized output view s'_T. The single processing steps are discussed in the following. Without the loss of generality steps carried out independently for both, the left and the right view, are discussed for the left view only.

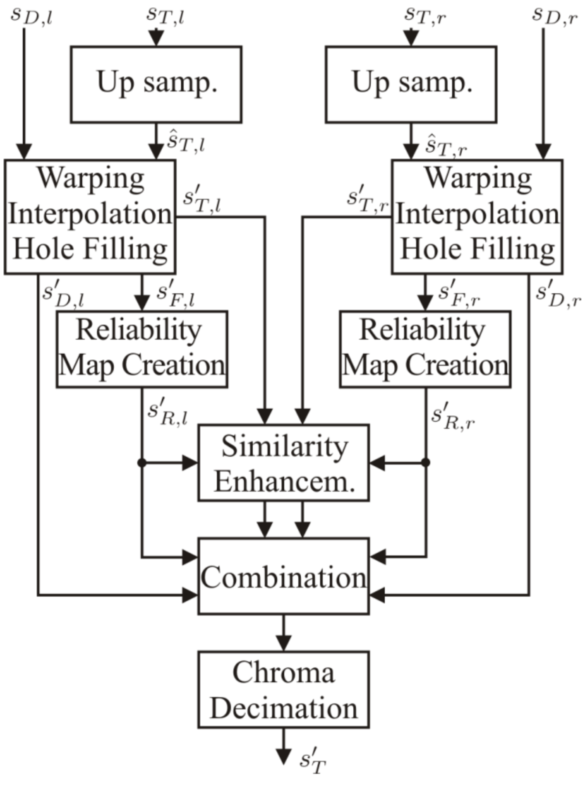


Figure 33: Processing steps of the view synthesis approach.

Similarly as the renderer model used in the encoder control (cp. sec. 2.5.1), the view synthesis algorithm supports two configurations. In the first configuration, which is referred to as interpolative rendering, an intermediate view is synthesized using both surrounding coded views. In the second configuration, which is referred to as non-interpolative rendering, an intermediate view is rendered mainly from one coded view; the other coded view is only used for rendering areas that are not present in the preferred coded view.

3.1.1 Upsampling of input video pictures

The luma channel of input texture $s_{T,l}$ is upsampled by a factor of four in horizontal direction. Chroma channels are upsampled by a factor of eight in horizontal direction and two in vertical direction. For upsampling, the FIR filters specified in HEVC for the purpose of motion-compensated interpolation are used. The resulting upsampled texture is denoted as $\hat{s}_{T,l}$.

3.1.2 Warping, interpolation and hole filling

Warping, interpolation and hole filling are carried out in a combined step. For warping disparities are computed as described in the beginning of sec. 2. Warping, interpolation and hole filling is carried out line wise and within a line interval wise. Processing direction is from left to right. An interval in the output view is defined by the warped positions x'_s and $x'_e$of two neighboring input view samples at positions x_s and x_e. Subsequently to the calculation of the interval boundaries, processing continues depending on the width of the interval.

- **Interpolation** is applied if the width of the warped interval $x'_e - x'_s$ is less than or equal to two times the sampling distance. An interpolation at the full sample position x'_{FP}located between the interval boundaries $x'_s$and $x'_e$is carried out. For this, samples from the up-sampled version of the input texture $\hat{s}_{T,l}$ are mapped to the interpolation positions x'_{FP}in the synthesized view $s'_{T,l}$. The position $\hat{x}$in the up-sampled view is derived from the distance of the interpolation position to the interval boundaries:

$$\hat{x} = 4 \cdot \left(\frac{x'_{FP} - x'_s}{x'_e - x'_s} + x_s \right) \tag{25}$$

- **Disocclusions**: If the width of the warped interval $x'_e - x'_s$is greater than two times the width of the sampling distance a disocclusion is assumed in the synthesized view. Instead of interpolation hole filling is carried. For this purpose samples in the interval are set to the value of sample belonging to the right interval boundary $s_{T,l}(x_e)$ (which belongs to the background). If the leftmost full sample position within the interval is close to the left interval border it is assumed that it belongs to the foreground and it is set to the value of the left interval boundary $s_{T,l}(x_s)$. Disoccluded and filled sample position are stored in the filling map $s'_{F,l}$.
- **Occlusions:** If the boundaries of an interval are reversed ($x'_e < x'_s$) the interval is occluded in the synthesized view. Rendering at a full sample position close to x'_e might be carried out, if the next interval is not occluded and x'_e belongs to a foreground object. Moreover, the algorithm uses the property that occluded background intervals are automatically overwritten by foreground objects in the synthesized view$s'_{T,l}$, due to the processing direction from left to right.

Chroma channels of the synthesized view are rendered together with luma channel and stored in the same resolution as luma. Moreover, if interpolative rendering is used, also a depth map $s'_{D,l}$ is extrapolated with full sample accuracy from the input depth map $s_{D,l}$ within the steps described above.

3.1.3 Reliability map creation

In this step the filling map $s'_{F,l}$ is converted to the reliability map $s'_{R,l}$. If interpolative rendering is used, positions marked as disocclusions in $s'_{F,l}$ are mapped to a reliability of 0. In areas located right to a disocclusion with a width of six samples the reliability is linearly increased from 0 to 255 from left to right in horizontal direction. All other samples are assigned with a reliability of 255. If non-interpolative rendering is used, positions marked as disocclusions in $s'_{F,l}$ are mapped to a reliability of 0. All other samples are assigned with a reliability of 255.

3.1.4 Similarity enhancement

In this step the histogram of $s'_{T,l}$ is adapted to the histogram of $s'_{T,r}$. For this purpose a look up table (LUT) realizing a function f is created, that is subsequently applied to map the samples of $s'_{T,l}$ to adapt their values.

The function f and the corresponding LUT are obtained by approximately solving

$$h[f(s'_{T,l})] = h[s'_{T,r}] \tag{26}$$

where$h[\dots]$ denotes the histogram only regarding samples at positions $(x, y)$with reliabilities $s'_{R,l}(x, y)$and $s'_{R,r}(x, y)$of 255. Chroma channels are treated in the same way.

3.1.5 Combination

$s'_{T,l}$and$s'_{T,r}$ are combined to obtain the synthesized output view in this step.

In the interpolative rendering mode is used, the decision how blending is carried out depends on the reliability maps $s'_{R,l}$or $s'_{R,r}$ and the rendered depth maps $s'_{D,l}$and $s'_{D,r}$. The rules for determining the blended sample value $s'_{T}(x, y)$ from $s'_{T,l}(x, y)$ and $s'_{T,r}(x, y)$ are given in the following:

- If position (x, y) is disoccluded (reliability of 0) in only one view, the sample value from the other view is used.
- Otherwise, if position (x, y) is disoccluded in both views, the backmost sample value is used.
- Otherwise, if the depth difference retrieved form $s'_{D,l}(x, y)$ and $s'_{D,r}(x, y)$ is above a threshold, the front sample is used.
- Otherwise, if one sample is not reliable with a value of 255, a weighted average with the given reliabilities as weights is used.
- Otherwise, a weighted average of $s'_{T,l}(x, y)$ and $s'_{T,r}(x, y)$with a higher weight for the view that is closer to the virtual view position is used.

If the non-interpolative rendering mode is used, the intermediate view is mainly rendered from one view are utilized and only holes are filled from the other view. Assuming $s'_{T,l}$ is the main view,the rules for determining the sample value $s'_{T}(x, y)$ from $s'_{T,l}(x, y)$ and $s'_{T,r}(x, y)$ are given in the following:

- If $s'_{R,l}(x, y)$is equal to 255 or $s'_{R,r}(x, y)$ is equal 0, the sample value$s'_{T,l}(x, y)$ is used.
- Otherwise, if $s'_{R,l}(x, y)$ is equal to 0, the sample value $s'_{T,r}(x, y)$ is used.
- Otherwise, a weighted average with the given reliabilities as weights is used.

3.1.6 Chroma decimation

To convert the 4:4:4 YUV representation obtained by rendering to the required 4:2:0 output, chroma channels are decimated by a factor of two in horizontal and vertical direction using the FIR filter (1;2;1).

3.2 VSRS (alternative view synthesis algorithm) [Not in CTC]

The VSRS algorithm was developed during the MPEG 3DV Exploration Experiments. VSRS takes two reference views and two depth maps as input to generate a synthesized virtual view. The intrinsic and extrinsic camera parameters are required and 1D parallel and non-parallel camera setups are supported.

The software has two main modes referred to as “General mode” and “1D mode”. The reference views are reprojected to the target viewpoint using sample-by-sample mapping based on 3D warping in “General mode”, or horizontal sample shifting in “1D mode”.

3.2.1 General mode

In the general mode, virtual views are generated by a technique referred to as “3D warping”. This process involves two steps. At first the original view (reference view) is projected into 3D world space using the corresponding reference depth map. Then the 3D space points are projected into the image plane of the “virtual” view. For this, the intrinsic camera parameters A, and extrinsic camera parameters E=[R|t] are required. The intrinsic matrix A, transforms the 3D camera coordinates to its 2D image coordinates. The extrinsic matrix E=[R|t] transforms the world coordinates to camera coordinates, which is composed of rotation matrix R and translation vector t. The two-step warping can be formulated in two equations as in eq. (**Błąd! Nie można odnaleźć źródła odwołania.**) and (**Błąd! Nie można odnaleźć źródła odwołania.**). First a sample (u_r, v_r) in the reference view is warped to the world coordinates (X_w, Y_w, Z_w), using the depth of the reference view:

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} = R^{-1}_{3\times 3,r}\left(z_r A^{-1}_{3\times 3,r} \begin{pmatrix} u_r \\ v_r \\ 1 \end{pmatrix} - t_{3\times 1,r} \right) \qquad (27)$$

where subscript r indicates the reference view and z_r is the depth value in the reference view at location (u_r, v_r) calculated from

$$z = \frac{1}{\frac{v}{255}\cdot\left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right) + \frac{1}{Z_{far}}} \qquad (28)$$

where v is an 8-bit intensity of the depth map value. It is noted that the values z, Z_{near}, and Z_{far} are assumed to be either all positive or all negative values.

Then the 3D point is mapped to the virtual view:

$$z_v \begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} = A_{3\times3,v}\left(R_{3\times3,v}\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + t_{3\times1,v}\right) \qquad (29)$$

where subscript v refers to the virtual view.

The general mode is based on a "reverse warping" algorithm. Instead of forward warping the left and right reference views to the virtual location, the left and right depth maps are warped to the virtual view location. Then after filtering, these depth maps are used to warp the reference views to the virtual view. This results in a higher rendering quality of the final synthesized view. Figure 34 depicts the flow diagram of the general mode.

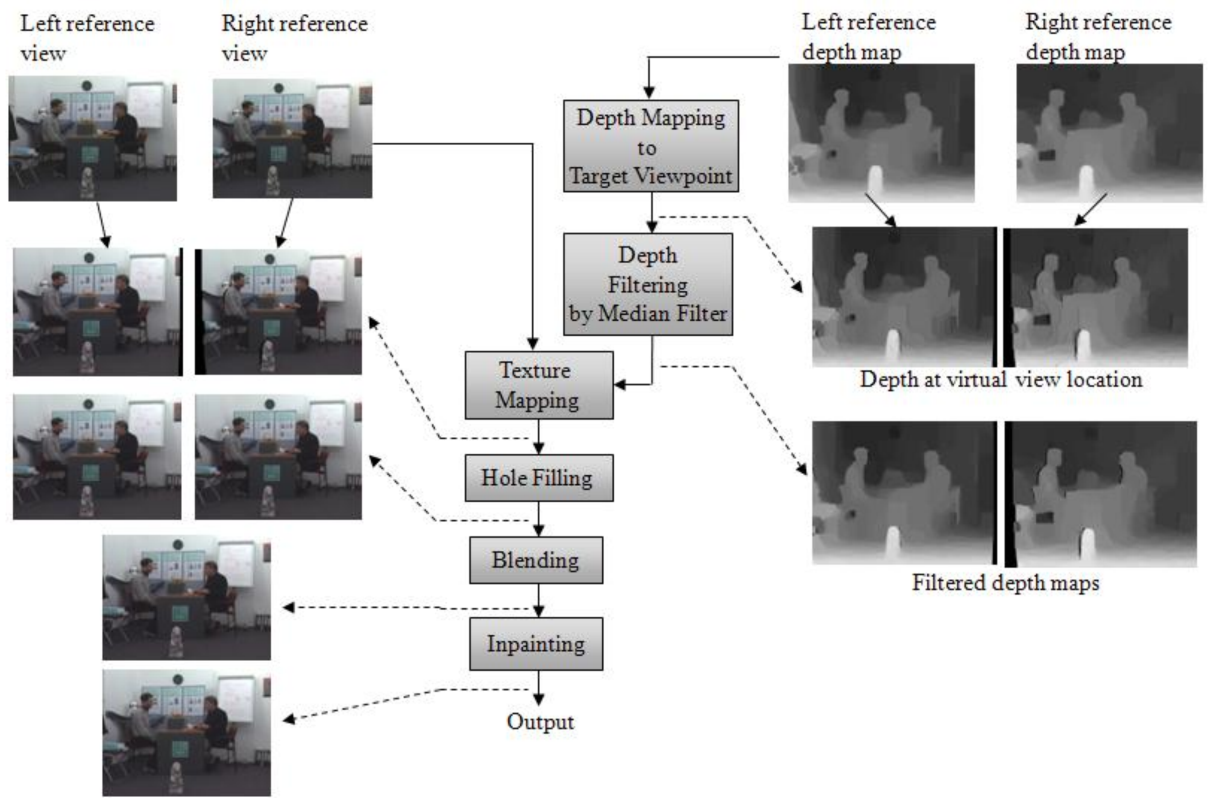


Figure 34: Flow diagram for VSRS general mode.

The steps of VSRS general mode are briefly described below:

- First, the two depth maps are mapped to the target viewpoint. E.g. the left reference depth is warped to the virtual view location using eq. (**Błąd! Nie można odnaleźć źródła odwołania.**) and (**Błąd! Nie można odnaleźć źródła odwołania.**). If multiple samples warp to the same location in the virtual view, then the sample closest to the camera wins, so foreground samples will occlude background samples. The right depth map is also warped in a similar way. We denote these warped depth maps as D_L' and D_R', respectively.
- The mapped depth maps D_L' and D_R' may contain small holes. Small holes which are caused by rounding to integer coordinates are filled by a series of median filtering. Furthermore, binary masks for each side are maintained to indicate larger holes, for example caused by occlusions that remain after filtering. During the

following steps, these binary masks are used and updated if necessary (for example during hole filling in step).

- Next, the left and right texture reference views are mapped to the target viewpoint using the filtered depth map D_L' and D_R'. So two texture images at the target viewpoint are obtained, one generated from the left reference view and the other from the right reference view. We denote them here as V_L' and V_R', respectively. Note that D_L' is used to warp the left reference, and D_R' is used to warp the right reference.
- Hole areas in the mapped texture images V_L' and V_R', which are caused by occlusion, are filled by samples from the other mapped texture image. So holes in V_L' are filled from non-hole areas in V_R' and vice versa.
- Next, these two virtual images are blended. The general mode has two modes of blending: Blending-on and Blending-off. The Blending-on mode is a weighted blending based on the baseline distance. So samples from the reference camera which is closer to the virtual view are assigned a higher weight, based on the baseline ratio. In Blending-off mode, all samples visible in the closer reference view are copied to the virtual view, and only hole areas are filled from the farther reference view. During this step, the binary masks are merged to form one mask indicating remaining holes which are inpainted in the next step.
- Any remaining holes after blending are filled by an inpainting algorithm using the binary mask. Inpainting algorithms can be used to reconstruct damaged portions of images. Generally a mask is used to indicate which image regions need to be inpainted. Next, colour information is propagated inward from the region boundaries, i.e., the known image information is used to fill in the missing areas. An inpainting example is show in Figure 35.

Additionally, VSRS contains a Boundary Noise Removal algorithm. In this mode, the binary maps indicating holes caused by occlusion are used to identify object boundaries. After identifying the background side of the holes based on the depth, the holes are expanded into the background. Then these areas in V_L' and V_R' are filled from the opposite reference view. This reduces noise around object boundaries, where foreground samples are falsely projected into background objects due to depth errors.

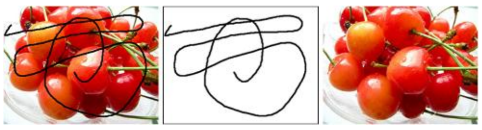


Figure 35: Inpainting: "damaged" image, mask, and result after inpainting.

3.2.2 1-d mode

VSRS provides a second synthesis mode other than the general "3D warping" as described above: 1D mode. This mode is implemented with assumptions that the optical axes of camera are in parallel and the views are rectified such that no vertical disparities exist. Under the assumption of 1D mode, formulations can be simpler than in the general case:

- The rotation matrix for every camera is identical to each other.
- The translation vectors of all cameras share the same translation in Y and Z directions, that is, T_y and T_z are constant for every view.
- As a consequence $z_v = z_r$
- Views are corrected (distortion and vertical disparity are null), so vertical position of intersection of optical axis in sensors is constant

So the $A_{3\times3}$ matrix has the following form $A_{3\times3} = \begin{bmatrix} fu & 0 & du \\ 0 & fv & dv \\ 0 & 0 & 1 \end{bmatrix}$, where fu and fv denote the horizontal and vertical focal length in samples; du and dv the position of intersection with the optical axis in image (dv is constant among cameras).

Then eq. (**Błąd! Nie można odnaleźć źródła odwołania.**) given for the general case can be simplified as,

$$u_v = u_r + \frac{f(t_{x,v} - t_{x,r})}{z_r} - (du_v - du_r) \text{ and } v_v = v_r \qquad (30)$$

The equation above is used to "warp" samples from real views to the virtual one.

Figure 36 depicts the flow diagram of the VSRS 1D-mode.

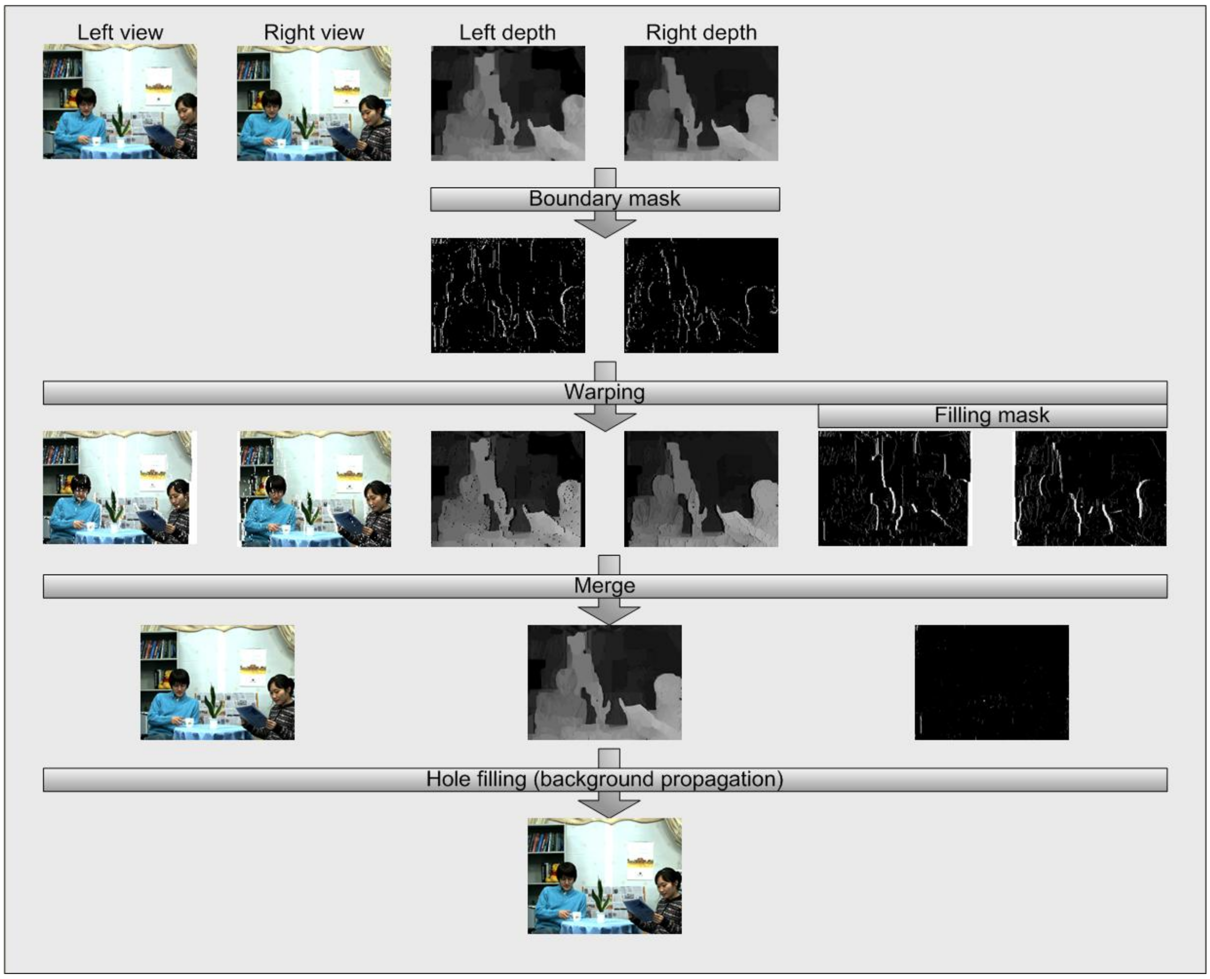


Figure 36: Flow diagram for VSRS 1D mode.

The algorithm proceeds as follows:

- In a preliminary phase,
  - The chroma components are upsampled to 4:4:4 format (for implementation simplicity).
  - For suppressing transient depth errors, the depth maps can be temporally filtered according to the variations of the colour information if the TemporalImprovementOption is chosen.
  - The colour video may be further upsampled, if sub-sample precision is specified in the configuration file, for example, half-sample or quarter-sample.
- During the warping process, the reference views and the depth maps are mapped to the target viewpoint using eq. (**Błąd! Nie można odnaleźć źródła odwołania.**), which is a 1D shifting on the samples. For each reference view, a binary mask is maintained indicating whether a sample in the targeted map is filled or not (hole sample). The warping procedure is also controlled by the splatting switch in configuration file. When splatting is selected, each sample in the reference view may be mapped to two sample locations. Besides, two enhancement processing on warping (corresponding to CleanNoiseOption and WarpEnhancementOption) suppress some synthesis artefacts due to the texture-depth misalignment at object boundaries (which causes foreground samples scattered to the background) and wrongly categorized holes in the foreground (which makes background samples appear in the foreground). Warping of the unreliable samples (which probably yield artefacts) is forbidden accordingly.
- Two warped images from left and right reference views are obtained from last step, which are then merged to a single image. This operation is also applied on warped depth maps and filling masks. In case of conflicts (two samples present for the same target position), the MergingOption specified by the user is applied in the following way.
  - Z-buffer only: Take the sample closest to camera always.

  - Averaging only: Mix colours using weights in reverse proportional to the distance of the virtual camera from the left and right reference views
  - Adaptive merging: Use either the proximity criterion (□o) if depth level difference is greater than a threshold or, (□o) if depth levels are too similar, uses the weighting method.

- Hole areas in the warped images are filled by propagating the background samples into the hole along the horizontal row.
- Final view image is downsampled to original size if necessary and transformed to 4:2:0 format for output purposes.

Additionally, VSRS 1D mode can use the boundary noise removal algorithm already described as final processing step in the section dedicated to the general mode.

4. Software

4.1 Software repository

The source code for the software will be available in the MPEG SVN repository. An initial version of the software is available in the following SVN repository.

https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/

For tool integration a branch for a company can be obtained by contacting:

gerhard.tech@hhi.fraunhofer.de,

kwegner@multimedia.edu.pl

4.2 Build System

The software can be built under linux using make. For Windows, solutions for different versions of Microsoft Visual Studio are provided.

4.3 Software Structure

The 3D-HEVC Test Model Software includes several applications and libraries for encoding, decoding and view synthesis:

- Applications:
  - TAppEncoder, executable for bit stream generation
  - TAppDecoder, executable for reconstruction.
  - TAppRenderer, executable view synthesis
  - TAppExtractor, executable for bitstream extraction
- Libraries:
  - TAppCommon, library for handling encoder, decoder and renderer options and camera parameters
  - TLibEncoder, encoding functionalities
  - TLibDecoder, decoding functionalities
  - TLibRenderer, renderer functionalities
  - TLibExtractor, bitstream extraction functionalities
  - TLibCommon, common functionalities
  - TLibVideoIO, video input/output functionalities