|  |  |
| --- | --- |
| **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**  3rd Meeting: Geneva, CH, 17–23 Jan. 2013 | Document: JCT3V-C1003 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **3D-AVC Test Model 5** | | |
| *Status:* | Draft Output Document of JCT-3V | | |
| *Purpose:* | Test model description | | |
| *Author(s) or Contact(s):* | Dmytro Rusanovskyy  [Fang-Chu Chen](mailto:fcchen@itri.org.tw)  Li Zhang Teruhiko Suzuki | Email: | dmytro.rusanovskyy@nokia.com fcchen@itri.org.tw lizhang@qti.qualcomm.com teruhikos@jp.sony.com |
| *Source:* | Editors | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract**

This document serves as a source of general tutorial information on 3D-ATM Reference Test Model for MVC+D and 3D-AVC specifications and describes encoding operations.

**Editing status**

|  |  |
| --- | --- |
| General comments, bug-fix and editorial inputs | Status in TM doc |
| Section describing SEI messages is missing. Currently NLDR SEI and GVR SEI are briefly described in corresponding sections. Depth acquisition, representation, sampling SEI are missing. | TBD |

Summary of adoptions for 3D-AVC (from JCT3V-C\_Notes\_d7) and their status in this document:

|  |  |
| --- | --- |
| Contribution | Status in TM doc |
| [JCT3V-C0094](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=533): 3D-CE7.a Improved Nonlinear Depth Representation [[I. Lim](mailto:ilsoon.lim@samsung.com), [H.-C. Wey](mailto:hc.wey@samsung.com), [D.-S. Park (Samsung)](mailto:dusikpark@samsung.com)] | Done |
| [JCT3V-C0054](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=492): Simplifications for adaptive luminance compensation in 3D-AVC [[J. Kang](mailto:jewonk@qti.qualcomm.com), [Y. Chen](mailto:cheny@qti.qualcomm.com), [L. Zhang](mailto:lizhang@qti.qualcomm.com), [M. Karczewicz(Qualcomm)](mailto:martak@qti.qualcomm.com)] | Done |
| [JCT3V-C0133](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=574): CE2.a: Results on simplification of the Inter-view candidate derivation [[J.-L. Lin](mailto:jl.lin@mediatek.com), [Y.-W. Chen](mailto:yiwen.chen@mediatek.com), [Y.-W. Huang](mailto:yuwen.huang@mediatek.com), [S. Lei (MediaTek)](mailto:shawmin.lei@mediatek.com)] | Done |
| [JCT3V-C0136](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=577): CE3.a related: Unconstrained inside-view motion prediction in 3D video coding [[J.-L. Lin](mailto:jl.lin@mediatek.com), [Y.-W. Chen](mailto:yiwen.chen@mediatek.com), [Y.-W. Huang](mailto:yuwen.huang@mediatek.com), [S. Lei (MediaTek)](mailto:shawmin.lei@mediatek.com)] | Done |
| [JCT3V-C0140](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=581): CE5.a related: Direct MVP derivation with reduced complexity [[J.-L. Lin](mailto:jl.lin@mediatek.com), [Y.-W. Chen](mailto:yiwen.chen@mediatek.com), [Y.-W. Huang](mailto:yuwen.huang@mediatek.com), [S. Lei (MediaTek)](mailto:shawmin.lei@mediatek.com)] | Done |

Summary of adoptions for 3D-AVC (from JCT3V-B\_Notes\_d7) and their status in this document:

|  |  |
| --- | --- |
| Contribution | Status in TM doc |
| JCT3V-B0031: CE2.A results on inter-view coding with adaptive luminance compensation [M. Mishurovskiy, A. Fartukov, I. Kovliga, J. Lee (Samsung)] | Done |
| JCT3V-B0057: 3D-CE3.a related: On inside-view motion prediction for 3D-AVC (removal of dependency in IVMP) [L. Zhang, Y. Chen, L. He (Qualcomm)] | Removed by[JCT3V-C0136](http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=577) |
| JCT3V-B0081: 3D-CE5.a: Unification of the depth to DV conversion [Jian-Liang Lin, Yi-Wen Chen, Yu-Wen Huang, Shawmin Lei (Mediatek)] | Done |
| JCT3V-B0149: CE5.a results on inter-view motion vector derivation using max disparity in skip and direct modes [J. Y. Lee, J. Lee, D.-S. Park (Samsung)] (partial search) | Done |
| JCT3V-B0079: 3D-CE5.a related: Draft text for the adopted simplified disparity vector derivation proposed in JCT3V-A0046 [Jian-Liang Lin, Yi-Wen Chen, Yu-Wen Huang, S. Lei (Mediatek)] | Done |
| JCT3V-B0150: 3DV-ATM: Simplified Calculations of Disparity [Lulu Chen (USTC), Dmytro Rusanovskyy, Miska M. Hannuksela (Nokia)] | Done |
| JCT3V-B0151: 3D-CE5.a related results on median-based skip and direct motion vector prediction [J. Y. Lee, J. Lee, D.-S. Park (Samsung)] | Done |
| JCT3V-B0153: 3D-CE5.a related results on temporal motion vector prediction in dependent view [J. Y. Lee, K.-J. Oh, J. Lee, D.-S. Park (Samsung)] | Done |
| JCT3V-B0033: AHG7 3D-AVC: Loss Detection of Depth Parameter Sets [M. M. Hannuksela (Nokia)] (SEI message) | Included |
| JCT3V-B0224: 3DV-ATM: Working draft text for B-VSP [D. Rusanovskyy, M.M. Hannuksela (Nokia)] | Included |
| JCT3V-B0094: 4x4 motion compensation for VSP (non CTC) | Done |

Summary of adoptions from earlier meetings that have not been included in this document:

|  |  |
| --- | --- |
| Contribution | Status in TM doc |
| JCT2-A0045/m25916: 3D-CE5.a results on motion vector competition-based Skip/Direct mode with explicit signaling (MediaTek) | Done |
| JCT2-A0046/m25917: 3D-CE5.a related: Simplification on the disparity vector derivation for AVC-based 3D video coding (MediaTek) | Done |

**Foreword**

This document describes major operations of 3DV-ATM which serves as a reference test model for “MVC Extension for Inclusion of Depth Maps (MVC+D)” (JCT3V-C1001) and 3D-AVC (JCT3V-C1002) specifications. The document also provides an encoder-side description of 3D-AVC Test Model (3DV-ATM) software as well as a description of decoder-side post-processing tools.

3DV-ATM test model can be configured to follow two specifications:

1. “MVC+D” configuration corresponds to “MVC extension for inclusion of depth maps”.

This encoding configuration follows the specification of JCT-3V document [JCT3V-C1001](http://phenix.int-evry.fr/jct3v/doc_end_user/documents/3_Geneva/wg11/JCT3V-C1001-v1.zip) and produces a complied bitstream. The method is invoking an independent second stream for the representation of a depth map as if it were monochrome video data, as well as high-level syntax signaling of the necessary information to express the interpretation of the depth data and its association with the video data. Macroblock-level changes to the AVC or MVC syntax, semantics and decoding processes are not considered.

MVC+D specification defines interlace coding for texture or depth. Work on enabling support of this functionality in 3D-ATM is currently undergoing.

1. “3D-AVC” configuration follows the specification of “AVC compatible video-plus-depth extension” as outlined in JCT-3V document [JCT3V-C1002](http://phenix.int-evry.fr/jct3v/doc_end_user/documents/3_Geneva/wg11/JCT3V-C1002-v3.zip), also known as 3D-AVC.

The 3D-AVC specifies monoscopic H.264/AVC compatibility for a base texture view. For dependent (non-base) texture views and depth views, an advanced coding process defined that improves compression efficiency compare to the MVC+D specification.

The advanced coding tools do not support interlace coding.

It should be noted, that, except for the interlace coding, 3D-AVC specification is a superset of the MVC+D. That is, a decoder supporting 3D-AVC can also decode MVC+D bitstreams. This document describes coding operations of 3D-AVC, and a special note is given if the certain tool is applicable for MVC+D specification.

**Table of Contents**

[Abstract 1](#_Toc351632992)

[**1** **Introduction** 6](#_Toc351632993)

[***1.1*** ***General Architecture of 3DV-ATM*** 6](#_Toc351632994)

[***1.2*** ***Elementary coding units and coding order*** 7](#_Toc351632995)

[***1.3*** ***Coding tools*** 8](#_Toc351632996)

[***1.4*** ***Inter-component dependencies*** 9](#_Toc351632997)

[**1.4.1** **Motion Information Inheritance** 9](#_Toc351632998)

[**1.4.2** **Disparity derivation** 9](#_Toc351632999)

[***2*** **Baseline depth coding** 11](#_Toc351633000)

[***2.1*** ***Non-linear depth representation (NDR)*** 11](#_Toc351633001)

[***3*** **Common tools for enhanced depth and texture coding** 13](#_Toc351633002)

[***3.1*** ***Slice header prediction*** 13](#_Toc351633003)

[***4*** **Enhanced depth coding** 14](#_Toc351633004)

[***4.1*** ***Introduction*** 14](#_Toc351633005)

[***4.2*** ***Depth Range Based Weighted Prediction (DRWP)*** 14](#_Toc351633006)

[***4.3*** ***In-loop Joint inter-View Depth Filtering (JVDF)*** 17](#_Toc351633007)

[***4.4*** ***Motion prediction from texture to depth*** 18](#_Toc351633008)

[***4.5*** ***Depth Intra Prediction*** 19](#_Toc351633009)

[**4.5.1** **Depth Intra Skip Prediction** 19](#_Toc351633010)

[**4.5.2** **Plane Segmentation based Intra Prediction (PSIP)** 20](#_Toc351633011)

[***5*** **Enhanced texture coding** 22](#_Toc351633012)

[***5.1*** ***Introduction*** 22](#_Toc351633013)

[***5.2*** ***In-loop view synthesis-based inter-view prediction (VSP)*** 23](#_Toc351633014)

[***5.3*** ***Depth-based Motion Vector Prediction (DMVP)*** 24](#_Toc351633015)

[**5.3.1** **Direction-Separated MVP (DS-MVP)** 24](#_Toc351633016)

[**5.3.2** **Disparity-based Skip and Direct modes** 25](#_Toc351633017)

[***5.4*** ***Adaptive Luminance Compensation (ALC)*** 26](#_Toc351633018)

[***6*** **Supplemental enhancement information** 30](#_Toc351633019)

[***6.1*** ***Gradual View Refresh (GVR)*** 30](#_Toc351633020)

[***6.2*** ***Loss detection of depth parameter sets*** 31](#_Toc351633021)

[**7** **Encoding algorithms** 31](#_Toc351633022)

[***7.1*** ***Rate-distortion optimization through view synthesis distortion*** 31](#_Toc351633023)

[***7.2*** ***Modified distortion metric for disparity estimation procedure for ALC*** 33](#_Toc351633024)

[***7.3*** ***Search zone parameters for disparity estimation procedure for ALC*** 34](#_Toc351633025)

[***7.4*** ***Rate-distortion optimization through ALC in the Skip mode*** 34](#_Toc351633026)

[**8** **Post processing** 34](#_Toc351633027)

[***8.1*** ***Depth dilation filter*** 34](#_Toc351633028)

[**Annex A: List of Contributors** 36](#_Toc351633029)

1. **Introduction**
   1. ***General Architecture of 3DV-ATM***

3DV-ATM codec is configurable for two coding modes, which are called “MVC+D” and “3D-AVC” in this document and follows MVC+D and 3D-AVC specifications respectively.

The “MVC+D” configuration provides H.264/MVC compatibility for texture views and implies that texture and depth map data is coded independently into a single coded bitstream. The difference of this configuration from H.264/MVC specification is limited to high level syntax and respective semantics as well as high-level decoder operation.

The “3D-AVC” configuration provides H.264/AVC compatible solution for a texture view and implies joint coding of texture and depth map data into a single bitstream. In this configuration, the texture data of base views (H.264/AVC compatible) is utilized for coding of the associated depth map data, and depth map data of enhancement views is utilized for coding of associated texture views. In this document, coding operations of 3D-AVC configuration (3D-AVC) are described. When certain tools are applicable for MVC+D configuration, a special note is given.

High level flow charts of 3DV-ATM encoder and decoder are presented in Figure 1 and Figure 2 respectively. On the figures below, solid lines depict general data flow and dashed lines show control information signaling. Dashed line A specifies utilization of depth information for advanced texture coding, and dashed line B specifies utilization of texture information (its parameters) for advanced depth coding.

Note, that 3DV-ATM includes pre-processing tool (Non-linear depth representation) and post-processing tools (Post-processing Dilatational Filtering) which are integral part of current 3DV system design but may be not a normative part of MVC+D and 3D-AVC specification text.

It should be noted, that MVC+D specification defines interlace coding for texture and depth component. Efforts on enabling this functionality in 3D-ATM are currently undergoing, and description will be provided in this document.



**Figure 1. High-level flow chart of 3DV-ATM encoder**



**Figure 2. High-level flow chart of 3DV-ATM decoder**

* 1. ***Elementary coding units and coding order***

3DV-ATM specifies that all texture and depth map view components which describe 3D scene at a particular moment of time form an access unit. A coded view component is represented by one or more Network Abstraction Layer (NAL) units similarly to AVC/MVC. The data of a coded view component is not interleaved by any other coded view component, and the data for an access unit is not interleaved by any other access unit in the bitstream/decoding order. See Figure 3 for an example, where access unit *t* consisting of texture and depth view components (T0t, T1t, D0t, D1t) precede in bitstream and decoding order access unit *t+1* consisting of texture and depth view components (T0t+1,T1t+1 ,D0t+1,D1t+1).



**Figure 3. Definition and coding order of access units**

Each texture view component for the AVC/MVC-compatible views is coded before the respective depth view component. Each texture view component of enhanced texture views is coded after the respective depth view component. The texture and depth view components of the same access units are coded in view dependency order. Texture and depth view components can be ordered in any order with respect to each other as long as the ordering obeys the mentioned constraints. Examples of coding order for an access unit include but are not limited to the following:

* T0, T1, D0, D1, (two AVC/MVC compatible texture views)
* T0, D0, T1, D1…(two AVC/MVC compatible texture views)
* T0, D0, D1, T1… (one AVC compatible texture view, one enhanced texture view)

The respective order of texture and depth view components of the same view\_id is indicated in the sequence parameter set. This indicated order controls the presence of certain syntax elements, such as flags for turning on/off texture-based coding tools for depth (e.g. IVMP).

* 1. ***Coding tools***

The MVC+D and 3D-AVC configurations include the following baseline depth coding tools (for further information, see Section 2):

1. Non-linear depth representation
   * Normative part of 3D-AVC spec
   * Not-normative part (SEI) of MVC+D
2. Reduced-resolution depth coding

The 3D-AVC configuration additionally includes the following coding tools which can be used for both enhanced depth views and enhanced texture views (for further information, see Section 3):

1. Slice header prediction

The 3D-AVC configuration additionally includes the following depth coding tools (for further information, see Section 4):

1. Depth-range-based weighted prediction (DRWP) for depth coding
2. In-loop Joint inter-View Depth Filtering (JVDF)
3. Motion prediction from texture to depth (a.k.a. Inside View Motion Prediction, IVMP)
4. Depth intra prediction: intra skip and plane segmentation based intra prediction

Note, that IVMP performs joint texture and depth coding, where parameters of coded texture (motion information) is utilized for efficient coding of depth data. This introduces an inter-component dependency, which is detailed in the Section 1.4.

The 3D-AVC configuration additionally includes the following texture coding tools (for further information, see Section 5) applicable for dependent texture views:

1. In-loop Block-based View Synthesis Prediction (VSP)
2. Depth-based Motion Vector Prediction (DMVP)
3. Inter-view coding with Adaptive Luminance Compensation (ALC)

Note, that DMVP and VSP performs joint texture and depth coding, where samples of depth data are utilized for efficient coding of texture. This introduces an inter-component dependency, which is detailed in the Section 1.4.

Additionally, 3D-ATM includes the following non-normative tools and used by default with MVC+D and 3D-AVC configurations (for further information, see Sections 6, 7 and 8):

1. Gradual view refresh (GVR) for texture and depth coding
2. Rate-distortion optimization through View Synthesis Distortion (VSD)
3. Post-processing Dilation Filtering (PDF) for depth map
   1. ***Inter-component dependencies***

Certain tools of 3D-AVC specification (IVMP, VSP, DMVP) are performing joint coding of texture and depth and thus introduce inter-component dependencies, which are clarified in this section.

* + 1. **Motion Information Inheritance**

Inside View Motion Prediction (see Section 4.4) requires motion information from coded texture to be available for coding of associated depth. In IVMP mode, the motion information resulting from texture data coding, including mb\_type, sub\_mb\_type, reference indices and motion vectors of the co-located MB in texture view component is stored and reused by the depth view component of the same view.

* + 1. **Disparity derivation**

3D-AVC specifies DMVP and VSP coding tools that applied for coding of texture samples of dependent views with help of associated depth map samples. For these purposes, selected depth map samples are converted to disparity vectors and utilized by DMVP and VSP.

Disparity value D has a linear relationship with depth map value d as equation (1) shows:

 (1)

Depth to disparity conversion can be performed as in equation (1) and requires such camera parameters as Znear, Zfar, focal length *f* and translation between views *l*. This method preserves a floating point representation of information and thus provides a high accuracy. However, such high accuracy of representation may be considered as un-necessary complicated as it requires floating point representation.

Simplified Calculations of Disparity utilizes a linear model to establish correspondence between depth map value d and disparity D as it shown in (2):

*D* = ( d \* DisparityScale + DisparityOffset << BitDepth ) + ( 1 << ( log2Div – 1 ) ) ) >> log2Div (2)

where d is a depth sample value derived by NDR lookup table as shown in Section 2.1.

DisparityScale is a scale factor, DisparityOffset is an offset value, BitDepth is equal to 8 and log2Div is a shift parameter that depends on the required accuracy of the disparity vectors. Parameters of conversion DisparityScale and DisparityOffset are transmitted within a bitstream with conventional variable length coding. For every pair of views (source view and target view) utilized in joint coding an independent set of parameters DisparityScale and DisparityOffset are transmitted.

To perform depth to disparity derivation (2), the association between block texture samples and depth samples of interest is established through 2 alternatives:

#### Maximal Out of Four Corners

Current disparity derivation method is utilized for VSP (specified in Section 5.2) and DMVP (specified in Section 5.3) coding modes. The disparity for currently coded texture block Cb vector is derived from block of depth map data d(Cb) associated with currently coded texture block Cb. Depth map samples located at spatial coordinates of four corners (top-left, top-right, bottom-left, bottom-right) of d(Cb) are compared against each other and maximal depth map value among them is converted to disparity value, as specified in equation (2). In the case of reduced resolution depth map, spatial coordinates of texture’s block corners are downscaled to meet depth map resolution.

#### Neighboring blocks based derivation

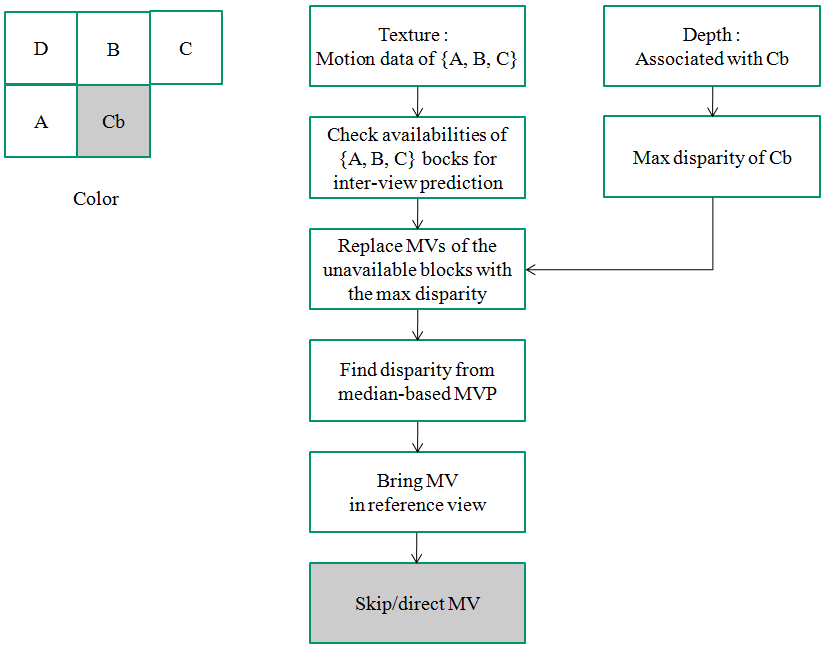
Another approach to derive disparity vector is utilized in Disparity based Skip and Direct modes (specified in Section 5.3.2). The disparity vector is derived from motion information of blocks neighboring to the current Cb block and from the associated block d(Cb) of depth data. The block naming convention utilized in this section is shown in **Figure 4.** The flowchart of derivation procedure is given in **Figure 5.**

Specifically, the disparity vector is derived from the motion vectors of neighboring blocks A, B, and C (D). If only one of the neighboring blocks was coded with inter-view prediction, its motion vector is interpreted as disparity vector for current Cb. If multiple inter-view prediction vectors are available in blocks A, B, C (D), the disparity vector is derived as a median of available alternatives. If none of neighboring blocks A,B,C(D) was coded with inter-view prediction, derivation procedure specified in Section 1.4.2.1 is utilized to derive disparity from depth map.

At the same time, if any



**Figure 4. Naming convention utilized in neighboring blocks based disparity derivation**



**Figure 5.** Disparity vector derivation procedure for Skip/Direct motion vector prediction

1. **Baseline depth coding**
   1. ***Non-linear depth representation (NDR)***

The idea of NDR tool is to represent closer objects more accurately than the distant ones. This is attained by encoding of non-linearly mapped disparity, normalized in range 0..255, instead of generic depth map values. The depth map is nonlinearly mapped through forward lookup table at the preprocessing stage of encoder and inversely mapped back to original representation at the post-processing stage of decoder.

The decoded depth is transformed, with inverse lookup table, back to linear disparity in range 0..255. The exact shape of forward and inverse transforms is defined by means of line-segment-approximation in two-dimensional linear-disparity-to-nonlinear-disparity space. The first (0, 0) and the last (255, 255) nodes of the curve are predefined (Figure 4). Positions of additional nodes are defined in form of deviations (NonlinearDepthPoints[i]) from the straight-line curve. These deviations are uniformly distributed along the whole range of 0 to 255 diagonal, with spacing depending on the number of points (NonlinearDepthNum).



Figure 4. Piecewise linear segments for mapping of depth values in Non-linear Depth Representation tool. Arrows mark deviations from uniformly distributed points along straight mapping, which are signalled in the bitstream.

Encoder analyzes input depth map statistics and performs decision making on utilization of the NDR based on properties of original depth map histogram. Two parameters of depth map histogram, namely disparity\_center and disparity\_concentration are derived as following:

 (3)

where disparity\_histogram( i ) is a histogram of the current frame of depth map. And

 (4)

where the effective bin is defined as the bin with more count than the threshold (heightⅹwidthⅹ0.002) and base histogram is defined as histogram ranging from the lowest effective bin and through the highest effective bin.

Following this, decision making on enabled NDR is performed as specified in (4):

if (disparity\_center > 50 & disparity\_concentration > 0.6 )

Enables NDR;

nonlinear\_depth\_representation\_enable\_flag=1 else

Disable NDR;

nonlinear\_depth\_representation\_enable\_flag=0

end;

The rationale behind this decision making is that the dense disparity in histogram cannot be represented effectively in the non-linear representation. Because there is big quantization error during nonlinear mapping and nonlinear re-maping and this quantization error is more serious in the dense disparity. On the contrary, the sparse disparity can be represented effectively in non-uniform quantization and the quantization error is negligibly small. Therefore, not only the disparity center, but also the disparity concentration is considered in deciding NDR on/off decision.

In the 3D-AVC coding configuration, the flag nonlinear\_depth\_representation\_enable\_flag is signalled in the slice header of depth views (profile\_idc=139) and defines decoding operations in coding loop. Parameters of piecewise linear model (nonlinear\_depth\_representation\_num, nonlinear\_depth\_representation\_model[i]) are signaled inside of the SPS of the depth component. Encoder/decoder performs conversion between NDR and original representation of depth for purposes of DMVP and VSP coding of dependent textures.

In the MVC+D configuration, NDR is applied at pre-processing and post-processing stage only and it is signaled through “Depth Representation SEI message”, specified in Section TBD.

1. **Common tools for enhanced depth and texture coding**
   1. ***Slice header prediction***

This tool can be applied only for enhanced texture and depth view components.

There may be redundancy between the slice headers of depth view components and texture view components. This coding tool is used to avoid repetitive coding of the same syntax element values in the view components of the same access unit.

The slice header prediction mechanism enables copying values for the syntax elements in the following categories:

* Reference picture list related syntax elements
* Prediction weight table related syntax elements
* Decoded reference picture marking related syntax elements
* All other syntax elements of the slice header except first\_mb\_in\_slice, slice\_type, pic\_parameter\_set\_id and slice\_qp\_delta

The syntax elements in each mentioned category can be selectively copied from an earlier slice header of the same access unit or included in the current slice header. The source of the syntax element values can be selected separately for each category from the following:

* The previous view component of the same type (texture or depth)
* The view component of an opposite type having the same view\_id
* The first view component of the same type (texture or depth)

1. **Enhanced depth coding**
   1. ***Introduction***

Figure 5 shows processing flow for dependent depth views in 3DV-ATM. The processing modules which are marked in red indicate the changes in original H.264/AVC design.

For example, in the Motion Compensated Prediction (MCP) block is modified by introducing the DRWP (see Section 4.2). In the reference Frame Buffer, number of reference pictures is increased by introducing a virtual VSP frame (see Section 5.2) and decoded depth pictures are filtered with in-loop JVDF process (see Section 4.3).



**Figure 5. Flow chart of depth map coding in 3DV-ATM**

* 1. ***Depth Range Based Weighted Prediction (DRWP)***

DRWP performs a non-linear compensation of the depth map inconsistency which is caused by use of different Znear/Zfar values in the conversion from the depth values z to the respective depth map samples v. This compensation process is described as follows.

For each depth map value v1 (was produced with *Z1near/Z1far*) we compute corresponding z depth value:

 (5)

Following this, we compute depth map value v2 representation for current depth value z and with new depth range (Z2near/Z2far):

 (6)

To enable DRWP in 3DV-ATM, the compensation process was implemented as a form of weighted prediction as follows. Let us choose two sampling points of the remapping function whose difference is a power of two in the input sample values. The input sample values of these sampling points are denoted a and a + 2d, where d is a positive integer. The output sample values y1 and y2, respectively, for these sampling points should be as close as possible to integer values to avoid quantization error.

The output sample value y then becomes a linear function of the input sample value x:

(7)

where round is a function returning the closest integer.

This function can be counted with integer arithmetic as follows:

(8)

where >> denotes a right bit-shift operation.

When offset o is defined as

(9)

and weight w is defined as

(10)

this function becomes identical to explicit weighted prediction:

(11)

If values y1 and y2 are indicated as fixed-point values Qf, where f represents the number of fractional bits, the quantization error in the depth map quantization rescaling may be reduced. The function then becomes

(12)

The encoder and the decoder derive the values of a, d, y1, and y2 from the camera and view synthesis parameters included in the bitstream.

In 3DV-ATM, the use of DRWP is automatically enabled at the encoder and decoder sides when there is a difference in depth map range appearing in two consequent frames.

In 3DV-ATM, linear function of weighted prediction is expressed through a pair of values gradient (*w*) and offset (*o*), which computed as:

,  (13)

The computed values of w and o are used as weights in AVC/MVC weighted prediction process. The encoder signals usage of the DRWP in the slice header.

In order to avoid potential rounding errors resulting from floating point arithmetic, the values of wX and oX (where X is 0 or 1, indicating reference picture list 0 or 1, respectively) are derived with the following fixed-point arithmetic version of the calculations above.

* 1. Initialize intermediate variables with default values

scale\_w=8;

* 1. Calculate variable W\_factora:

temp1 = ZF2 - ZN2;

temp2 = ZF2;

x = ( temp1 + ( temp2 >> 1 ) ) / temp2;

sign = ( (temp1 - x \* temp2) < 0 ) ? -1 : 1;

W\_factora = (x<<scale\_w);

W\_factora += ( ((temp1 - x \* temp2) << scale\_w) + sign \* (temp2 >> 1) ) / temp2;

* 1. Calculate variable W\_factorb:

temp1 = ZF1;

temp2 = ZF1 – ZN1;

x = ( temp1 + ( temp2 >> 1 ) ) / temp2;

sign = ( (temp1 - x \* temp2) < 0 ) ? -1 : 1;

W\_factorb = (x<<scale\_w);

W\_factorb += ( ((temp1 - x \* temp2) << scale\_w) + sign \* (temp2 >> 1) ) / temp2;

* 1. Calculate variable W\_factorc:

temp1 = ZN1;

temp2 = ZN2;

x = ( temp1 + ( temp2 >> 1 ) ) / temp2;

sign = ( (temp1 - x \* temp2) < 0 ) ? -1 : 1;

W\_factorc = (x<<scale\_w);

W\_factorc += ( ((temp1 - x \* temp2) << scale\_w) + sign \* (temp2 >> 1) ) / temp2;

* 1. Calculate variable wX:

wX= (W\_factora\*W\_factorb\*W\_factorc + (1<<(scale\_w\*3- logWD-1)) ) >> (scale\_w\*3- logWD);

wX=Clip(-127,128,wX);

Variable oX is calculated through the following derivation process.

1. Initialize intermediate variable with default value:

scale\_o=8;

1. Calculate variable O\_factora:

O\_factora=( (ZN1<<(scale\_o))+( ZF2>>1) ) / ZF2;

1. Calculate variable O\_factorb:

temp1 = ZF1- ZF2;

temp2 = ZF1- ZN1;

sign= (temp1 < 0) ? -1 : 1;

x = ( temp1 + sign \* ( temp2 >> 1 ) ) / temp2;

sign = ( (temp1 - x \* temp2) < 0 ) ? -1 : 1;

O\_factorb = (x<<scale\_o);

O\_factorb += ( ((temp1 - x \* temp2) << scale\_o) + sign \* (temp2 >> 1) ) / temp2;

1. Calculate variable o0:

oX=( O\_factora\*O\_factorb + (1<<(scale\_o\*2-8-1)) ) >> (scale\_o\*2-8);

oX =Clip(-127,128, oX);

* 1. ***In-loop Joint inter-View Depth Filtering (JVDF)***

Depth map images of available views are filtered jointly. The depth map of the currently processed view *Vc* is converted into the depth space (Z-space) as it shown in (5). Following this, depth map images of other available views (*Va1, Va2*) are converted to the depth space and projected to the currently processed view *Vc*. Projections create several estimates of the real depth value, which are averaged in order to produce a denoised estimate of the real depth value. Filtered depth value  of current view *Vc* is produced through a weighted average with depth estimate values  projected from an available views *Va* to a currently processed view *Vc.*

 (14)

where *{w1, w2}* are weighting factors or filter coefficients for the depth values of different views or view projections.

Filtering is applied if depth value estimates belong to a certain confidence interval, in other words, if the absolute difference between estimates is below a particular threshold (*Th*):

If *|za→c – zc |<Th* , *w1=w2=0.5* (15)

Otherwise, *w1 = 1, w2 = 0*

Figure 6 shows the coding of two depth map views with in-loop implementation of JVDF. The AVC coding algorithm is depicted within a dashed line box, marked in black color. The JVDF is depicted in the solid-line box marked in red color. The parameter Th is transmitted to the decoder within a sequence parameter set extension.



**Figure 6. In-loop JVDF of 2-view depth map coding.**

* 1. ***Motion prediction from texture to depth***

This tool can be applied only for depth view components for AVC/MVC compatible texture views.

Since the texture view component and its associated depth view component have similar object silhouette, they may have similar object movement, thus there is redundancy in their motion fields. Therefore, motion prediction from a texture view component to the associated depth view component is enabled as a new mode in the proposed codec. An Inside View Motion Prediction (IVMP) mode is enabled for an Inter coded MB only in depth view components. In IVMP mode, the motion information, including mb\_type, sub\_mb\_type, reference indices and motion vectors of the co-located MB in texture view component is reused by the depth view component of the same view. A flag is signaled in each MB to indicate whether it uses the IVMP mode. As shown in Figure 7, the flag is true for the highlighted MB in the 4-th picture of the depth view and the motion vector of the co-located MB in the 4-th picture of the texture view (in red) is reused for the highlighted MB in the depth view component. Note that, in current implementation, IVMP mode applies only to non-anchor pictures.



**Figure 7. Motion prediction from texture to depth.**

* 1. ***Depth Intra Prediction***

Following two depth intra prediction tools are applied for depth view components.

* + 1. **Depth Intra Skip Prediction**

Since the depth images have high spatial correlation, efficient high priority intra prediction can be useful for depth image coding. The depth intra skip prediction (DISP) is designed based on conventional intra 16x16 prediction. However, it uses the estimated prediction direction both at the encoder and decoder sides as well as does not encode any residual data. For the estimation of prediction direction, adjacent above and left lines are used. The number of changes (NOC) value of left line (NOC(L)) is calculated by

 (16)

NOC(A) is calculated in the same way. From the calculated NOC(L) and NOC(A), prediction direction is derived as follows.

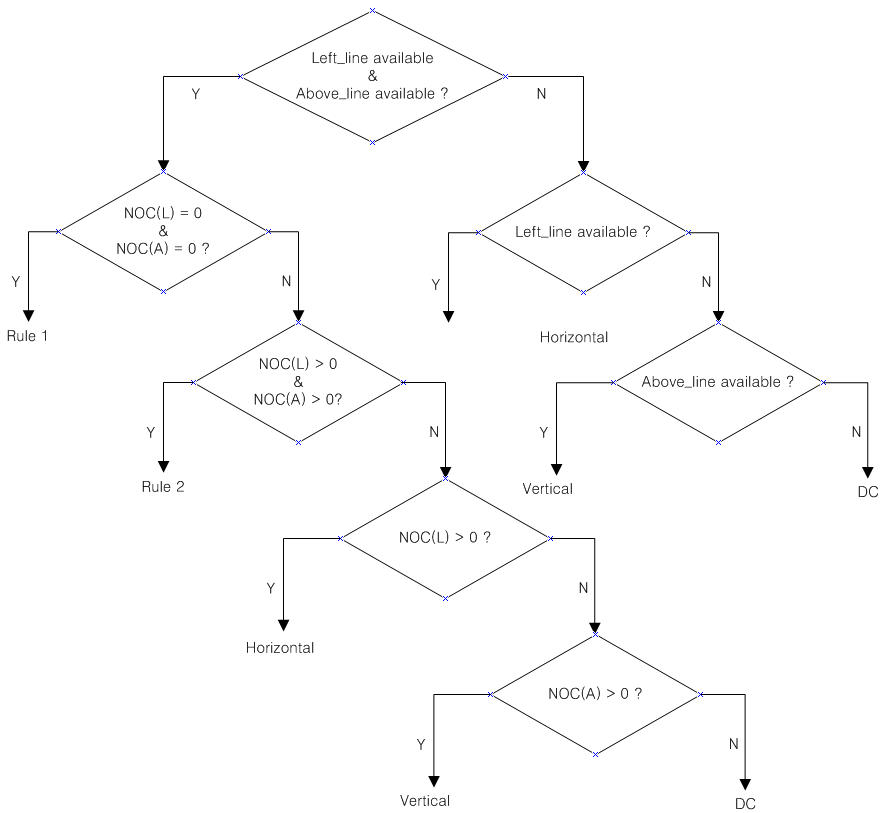


Figure 8. Flow diagram of estimation of prediction direction.

Rule 1 and rule 2 in Figure 8 are defined as follows:

<Rule 1>

if (left MB = ver I16x16 mode & above MB = ver I16x16 mode)

: vertical direction

else if(left MB = hor I16x16 mode & above MB = hor I16x16 mode)

: horizontal direction

else if(left[0] < up[0])

: vertical direction

else if(left[0] > up[0])

: horizontal direction

else

: DC direction

<Rule 2>

if (left MB = ver I16x16 mode & above MB = ver I16x16 mode)

: vertical direction

else if(left MB = hor I16x16 mode & above MB = hor I16x16 mode)

: horizontal direction

else if (left MB ≠ hor I16x16 mode & above MB = ver I16x16 mode)

: vertical direction

else if (left MB = hor I16x16 mode & above MB ≠ ver I16x16 mode)

: horizontal direction

else if (NOC(L) < NOC(A))

: vertical direction

else if (NOC(L) > NOC(A))

: horizontal direction

else if( abs(above[0] – above[15]) > abs(left[0] – left[15])

: vertical direction

else if( abs(left[0] – left[15]) > abs(above[0] – above[15])

: horizontal direction

else

: DC direction

DISP is only applied for depth views and a flag to indicate the DISP mode is signaled in each macroblock. In addition, boundary strength (Bs) of deblocking filter is adjusted when current macroblock is coded as DISP mode.

* + 1. **Plane Segmentation based Intra Prediction (PSIP)**

Since it is important to keep the quality of sharp edge in depth map, a new intra prediction scheme is designed to keep sharp edge boundary by assigning more bits for edge. The idea of PSIP is to segment each depth block into two regions, and apply different prediction for each segmented region.

For each N×N depth map block, as shown in Figure 9, it first segments its upper and left available pixels into two groups, and determine two representative values, P1, P*2* by averaging each group of pixels. Then, each target pixels, Ci, are predicted by its closest representative value, *i.e.*,

 (17)

where  is predicted pixel value.

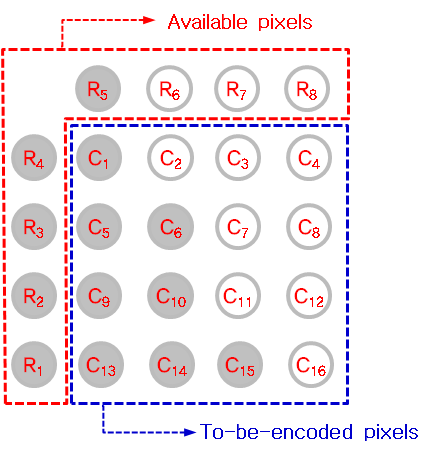


Figure 9. Example of k-region segmentation (k=2)

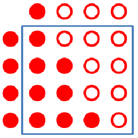


Figure 10. Example of prediction map for Figure 9.

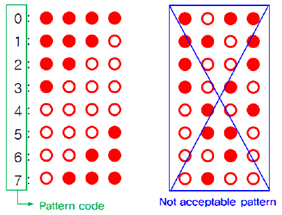


Figure 11. Example of pattern table for 4×4 block.

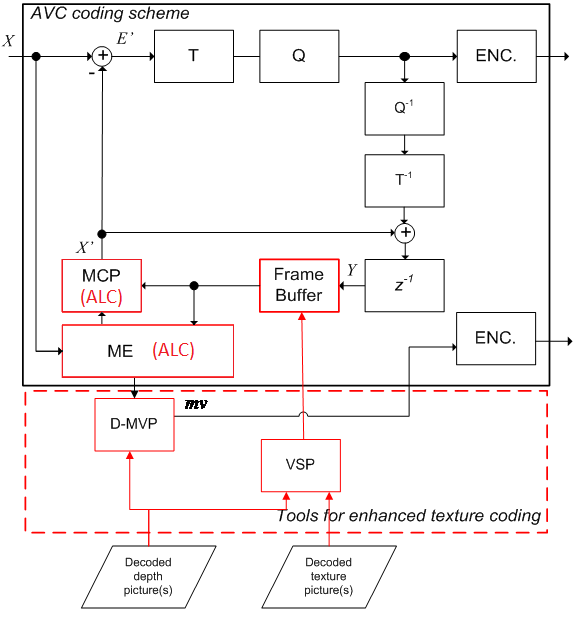
During PSIP prediction, it is additionally required to encode the edge information, say, prediction map, which is NxN binary information as shown in Figure 10. In PSIP, each 1xN row (or 1 x N column) is coded according to the pre-determined pattern table as shown in Figure 11, where patterns with more than two bit-transitions are excluded. Finally, the difference between adjacent rows (or columns) is encoded, since difference value will be closed to 0. The difference is computed as

 (18)

where MOD indicates modulo operator, and *si* represents *i*th row (or column) pattern code (*s*0 means a pattern code for neighbourhood pixels).

1. **Enhanced texture coding**
   1. ***Introduction***

Figure 12 shows enhanced texture coding in 3DV-ATM. The processing modules which are marked in red indicate the novel coding tools of 3DV-ATM when compared to AVC/MVC.



**Figure 12. Flow chart of enhanced texture coding in 3DV-ATM.**

Adaptive Luminance Compensation (ALC) provides additional compression gain by interview motion vector estimation followed by motion compensated prediction which includes a particular correction of luminance components of a reference block.

* 1. ***In-loop view synthesis-based inter-view prediction (VSP)***

In-loop block-based View Synthesis Prediction (VSP) is supported in 3DV-ATM for enhanced texture coding. VSP utilizes backward warping (VSP), as illustrated below in Figure 13. In the example the following coding order is utilized: (T0, D0, D1, T1). Texture component T0 is a base view and T1 is dependent view coded with the VSP. Depth map components D0 and D1 are the depth maps associated with T0 and T1, respectively.

Following this, denote *x* and *y* are absolute spatial coordinates of the samples consisting of currently coded Cb within dependent texture view T1. Samples of synthesized reference block R(Cb) can be retrieved from a source image *s(x,y)* T0 with disparity vector D that is derived from depth data d(Cb) which is associated with Cb. Applying the vector D to spatial coordinates of Cb provides coordinates of source samples R(Cb) in view T0 :

(19)

In such implementation, residual signal *r(Cb)* predicted from the synthesised R(Cb) are derived with a tradition Motion Compensated Prediction (MCP) module where displacement (motion) vectors are replaced with disparity vectors.

(20)

Visualization of this process is shown in **Figure 13**

In the case of parallel camera arrangement which is assumed to be a common 3DV use case, different views of MVD data are rectified, thus vertical component of the displacement between different views is equal to zero and disparity D as a horizontal component of motion vector is sufficient to derive prediction signal R(Cb) as it is shown in (14), thus

(21)

Disparity vector value D is derived as described in Section 1.4.2.1. This derviation is performed at the encoder and decoder side through an identical process, thus no signalling is performed. The reference index RefIdx is transmitted to specify the image that serves as a source for B-VSP process.



**Figure 13. Visualization of view synthesis prediction based on backward warping.**

Process B-VSP creates a virtual synthesized picture that does not require an actual memory allocation, instead it is mapped through the existing source image refered by RefIdx. However, to improve the codec functionlaity and benefit from encoder side reference picture management (i.e., RPLR commands), synthesized virtual picture index is inserted in the initial reference picture lists List0 and List1 following temporal and inter-view reference frames. Reference picture list modification syntax (i.e., RPLR commands) are extended to support VSP reference pictures, thus any ordering of reference picture lists is allowed.

In addition to MCP prediction from VSP frame, 3D-AVC defines the VSP skip/direct mode. VSP skip/direct mode is signaled with two additional flags. First, a VSP skip flag is signaled to distinguish the VSP skip mode from the conventional skip one. The VSP skip flag can be placed ahead of or behind the conventional skip flag based on mode information of the neighboring blocks around the current block. Second, if an MB is signaled as the conventional or VSPdirect mode, a direct type flag is further signaled whether a VSP frame is forced to be used as reference.

* 1. ***Depth-based Motion Vector Prediction (DMVP)***

Depth-based motion vector prediction (DMVP) is a coding tool which takes in use available depth map data and utilizes it for coding/decoding of the associated with depth map texture data.

This coding tool is enabled for enhanced texture coding and requires depth map data to be coded prior to the texture data.

The DMVP tool consists of two parts, direction-separated MVP for the Inter mode and disparity-based Skip and Direct modes, which are described next.

* + 1. **Direction-Separated MVP (DS-MVP)**

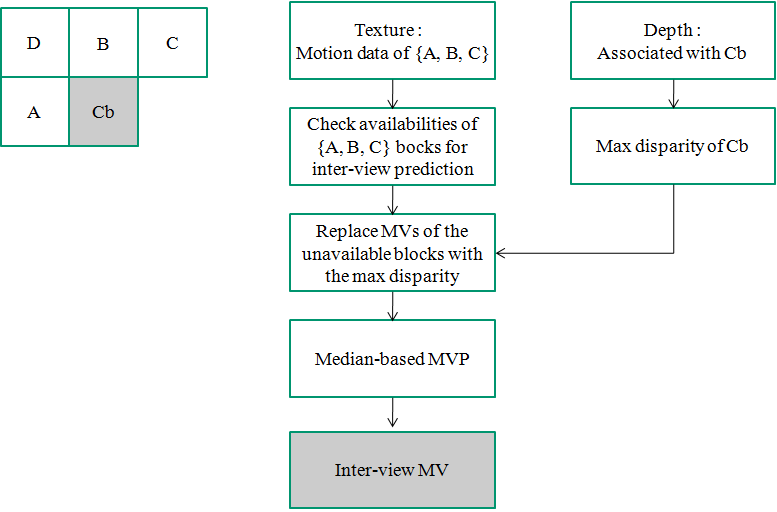
Conventional median-based MVP of H.264/AVC is restricted to identical prediction directions of motion vector candidates. All available neighboring blocks are classified according to the direction of their prediction (temporal or inter-view).

***Inter-view prediction***

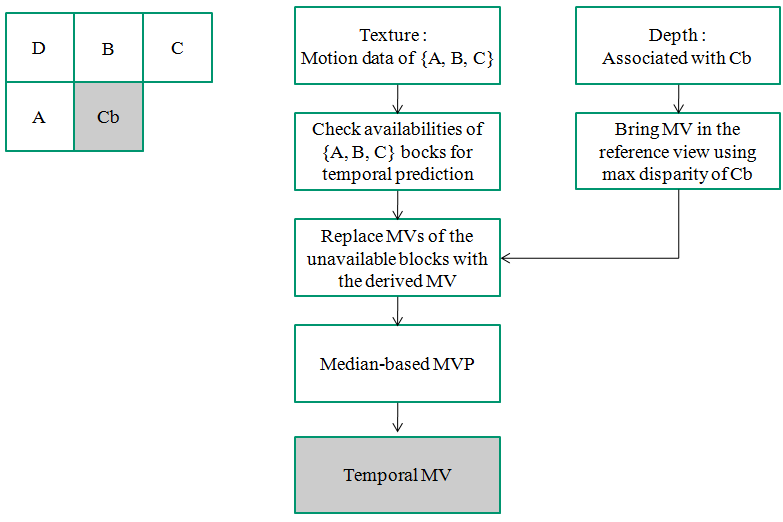
If the current block Cb, see Figure 16, uses an inter-view reference picture, all neighboring blocks which do not utilize inter-view prediction are marked as not-available for MVP. Motion vectors of the neighboring blocks marked as not-available are replaced with disparity vector derived from depth data associated with Cb instead of a zero motion vector and then considered in the median MVP of H.264/AVC. The disparity vector is derived as it specified in Section 1.4.2.1. The flowchart of this process is depicted in Figure 14.

***Inter prediction***

If Cb uses temporal prediction, neighboring blocks that used inter-view reference frames are marked as not-available for MVP. Motion vectors of the neighboring blocks marked as not-available are replaced with a motion vector of a corresponding block in a reference view. The corresponding block is derived by applying disparity vector D to the coordinates of the current texture block. The disparity vector is derived as it specified in Section 1.4.2.1. If corresponding block is not coded with inter-prediction (does not have motion information), a zero vector is considered. The flowchart of this process is depicted in Figure 15.



**Figure 14. Flow chart of inter-view prediction.**



**Figure 15. Flow chart of temporal prediction**

* + 1. **Disparity-based Skip and Direct modes**

In the Skip/Direct modes motion information is not coded, instead it is derived at both sides through an identical process. Motion information for coding of the current block Cb in Skip/Direct modes is derived from motion information of the corresponding block in the base view. The correspondence between Cb and corresponding block in the base view is established through a disparity vector which is applied at the central sample of block Cb, see **Figure 4**. A motion partition referenced by this vector in the base view provides motion information (reference index and motion vectors) for coding of the current Cb.

The disparity derivation procedure for this mode is specified in Section 1.4.2.2 and the flow chart of the motion information derivation is provided in **Figure 16**.

If corresponding block in base view is not available, the direction-separated MVP derivation as described in Section 5.3.1 with reference index equal to zero is used.

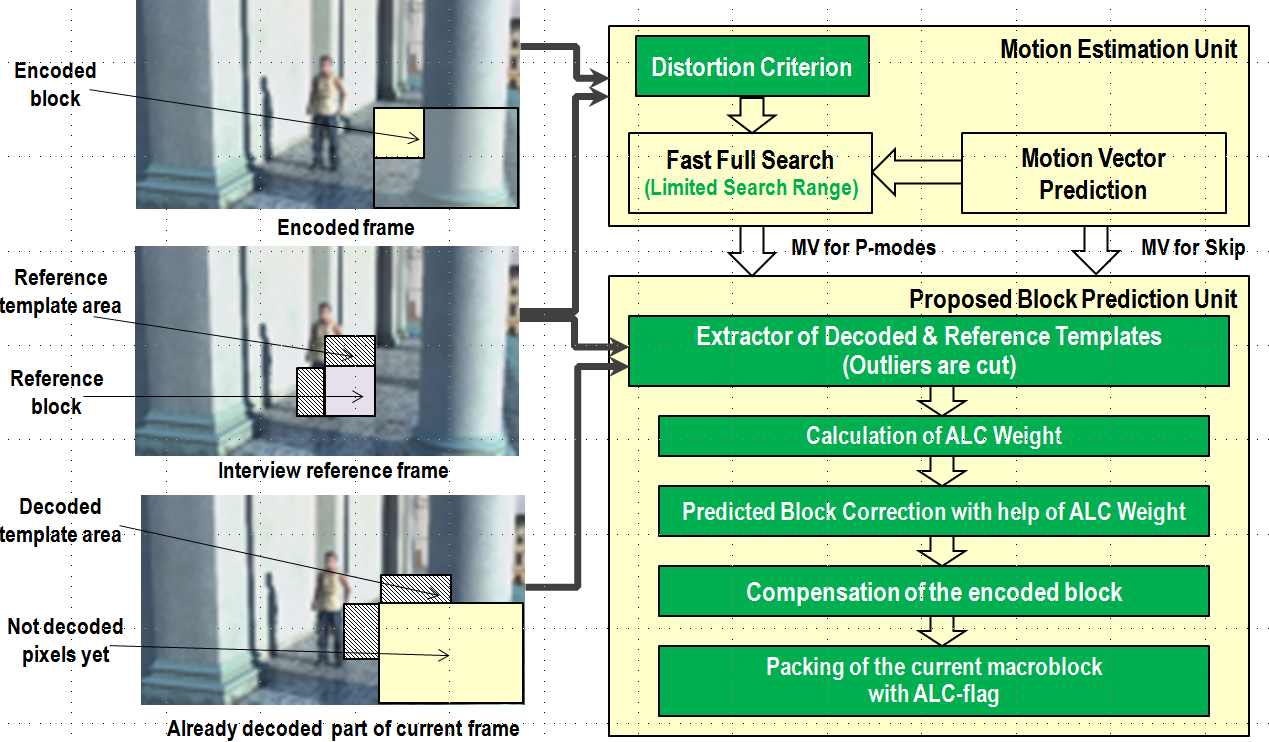
* 1. ***Adaptive Luminance Compensation (ALC)***

Adaptive luminance compensation (ALC) is a coding tool which suppresses local illumination changes between encoded macroblock and predicted blocks that belong to an interview reference frame. This technology covers several aspects: compensation model, luminance discrepancy estimation process, motion (disparity) vector derivation and signaling mechanism.

It’s well known that the majority of natural multi-view sequences and even some of synthesized multi-view sequences usually demonstrate inter-view mismatches in particular, local luminance discrepancy between object projections captured by adjacent cameras of 3D capturing system even if they are geometrically matched accurately. Suppressing of a luminance discrepancy allows increasing the quality of interview prediction of blocks. It decreases rate of residuals and results in PSNR increase of encoded frames.

Main stages of ALC technology are depicted in Figure 17. All inter macroblock modes in P-slices of depended views are checked with ALC enabled and disabled, then the best mode is chosen by RDO part of the encoder. The decoded image parts are used for ALC parameters calculation and for making correction. One-bit flag for each inter macroblock is signalled to indicate the usages of ALC, including macroblocks encoded in P-Skip mode, and that belongs to dependent views of texture component of multiview + depth video.

Because of ALC is applied for interview reference frames only, reference indexes are not signaled into bitstream. Instead, motion vector prediction aligned for inter-view motion vector prediction is used for P\_SKIP mode if ALC is turned on for the macroblock. In particular, If ALC tool is used, reference index refIdxL0 is derived as an interview picture that appears first in RefPicList0. The depth-based derivation process for median luma motion vector prediction in subclause J.8.3.1.8 of 3D-AVC specification is invoked for deriving motion vector prediction mvpL0 for P\_Skip.



**Figure 17. General scheme of mackroblock encoding in various macroblock modes with help of ALC technology.**

The motion estimation process for ALC compensation is shown on Figure 18.

Encoded Frame

Interview Reference Frame

Displacement Vector

2 x Search Zone Width

2 x Search Zone Height

Predicted block

Encoded block

**Figure 18. Disparity vector search scheme for ALC-compensated block.**

In order to perform motion estimation procedure that generates displacement (disparity) vector, “limited” search zone is defined based on an assumption that an input multiview video sequence is already rectified. In particular, search zone sizes are defined as follows:

* Search Zone Height is defined to be equal to 6 quarter pixel positions
* Search Zone Width is defined to be equal to 20 quarter pixel positions.

In order to provide higher precision of displacement vector determination so-called “Fast Full Search” might be used at the encoder side for ALC encoding modes.

To improve accuracy of disparity vector estimation, considering potential discrepancy of luminance between encoded and predicted blocks, it’s proposed to modify distortion metric from SAD to so-called MR\_SAD4x4\_DC. This change deals with encoder part only and is an informative component of ALC technology.

First, MR\_SAD4x4 is defined as a sum of mean-removed SADs over all sub-blocks 4x4 comprising currently encoded block which sizes are (H, W):

(22)

, (23)

Here: R – reference frame; X- encoded frame; H – height of an encoded block; W – width of an encoded block.

Then, taking into account possible DC differences, the following modification is being done, making distortion metric to be sensitive to both DC differences and mean-removed differences:

.(24)

Experiments show that whereas complexity of the encoder is not increased due-to use of MR\_SAD4x4\_DC instead of SAD metric, compression gain may slightly go up.

ALC compensation is applicable for luma samples only. For each prediction mode to be tested in P-Slices of texture dependent views, a macroblock is divided into non-overlapped blocks referred further as *predParti[ x , y ]*, such that these non-overlapped blocks are ordered in a decoding order, and ALC is applied for each *predParti[ x, y ]* sequentially. To derive a weight factor, the two template blocks (i.e., the above and the left blocks) as shown in Figure 19, are selected. The respective template block sizes (psx, 4) and (4, psy) of *predParti[ x , y ]* may depend on a macroblock type. Table 1 below shows how block sizes are defined based on mb\_type value:

Table 1: Parameters (psx and psy) definition of the width and height of the top and left regions for weights derivation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **mb\_type** | **psx** | **psy** | **mb\_type (sub\_mb\_type)** | | **psx** | **psy** |
| P\_L0\_16x16 | 16 | 16 | P\_8x8, P\_8x8ref0 | P\_L0\_8x8 | 8 | 8 |
| P\_L0\_L0\_16x8 | 16 | 8 | P\_L0\_8x4 | 4 | 4 |
| P\_L0\_L0\_8x16 | 8 | 16 | P\_L0\_4x8 | 4 | 4 |
| P\_Skip | 16 | 16 | P\_L0\_4x4 | 4 | 4 |

To perform ALC, for each *predParti[ x , y ]*, weight factor W is calculated as follows:

1. Two intermediate variables Ref\_Dec and Ref\_Ref are calculated:

 (25)

1. weight W is calculated as follows:



where / - is an integer division with rounding to zero, TH - is a predefined threshold that is set to be equal 30; CUT\_TH – is another predefined threshold that is equal to 4. Blocks *LTRefi* - *UTRefi* - *LTDeci* - *UTDeci* –depicted in figure xx.

**Figure 19. Relative position of templates and block from the prediction area.**

*Decoded Frame*

*Interview Reference Frame*

*Decoded block correspond predParti*

*predParti*

*LT  
Deci*

*UTDeci*

*LT  
Refi*

*UTRefi*

Motion Vector

ALC is performed for all values of each *predParti[ x, y ]* of the macroblock (i=1..n). Predicted samples *predPartALCi [ x, y ]* are derived as follows:

*predPartALCi [ x, y ] =Min(255, ( predParti [ x, y ] \* W + 2logWDC − 1 ) >> logWDC )*

where logWDC is equal to 15 and >> means arithmetic right shift.

Eventually, corrected blocks *predPartALCi[ x, y ]* are used for compensation of encoded (decoded) macroblock that spatially corresponds to *predPartALCi[ x, y ]*.

For the practical reasons, ALC correction is not performed if , meaning that pixels in a reference template area and decoded template area are quite similar. The same condition in the weight W calculation leads to W to be equal to , that again means absence of ALC correction for the current block.

1. **Supplemental enhancement information**
   1. ***Gradual View Refresh (GVR)***

When decoding is started from a GVR point, a subset of the views can be accurately decoded, while the remaining views can be approximately reconstructed using view synthesis. Perfect reconstruction of all views can be reached at a subsequent random access point.

While no inter prediction is applied in IDR and anchor access units, GVR access units are coded in a manner that inter prediction is selectively enabled and hence compression improvement compared to IDR and anchor access units may be reached. The encoder selects which views are refreshed in a GVR access units and codes these view components in the GVR access unit without inter prediction, while the remaining non-refreshed views may use both inter and inter-view prediction. The selection of refreshed views is done in a manner that each view becomes refreshed within a reasonable period, which may depend on the targeted application but is typically up to few seconds at most. To allow decoders to detect the possibility of starting decoding, GVR access units may be indicated in the NAL unit header or slice header similarly to anchor access units.

Figure 20 presents a configuration of using GVR access units at every other random access point. It is assumed in Figure 15 that the frame rate is 30 Hz and random access points are coded every half a second. GVR access units refresh the base view only, while the non-base views are refreshed once per second with anchor access units.



a) b)

**Figure 20. Examples of GVR access units (picture order count 15 and 45) coded at every other random access point; a) Two-view coding, b) Three-view coding with PIP inter-view prediction hierarchy.**

When decoding is started from a GVR access unit, the texture and depth view components which do not use inter prediction are decoded, and the remaining “non-refreshed” views are approximately reconstructed using view synthesis (a.k.a. depth-image-based rendering, DIBR). Decoding of the non-refreshed views can be started at subsequent IDR, anchor, or GVR access units. Figure 21 presents an example of the decoder side operation when decoding is started at a GVR access unit of the bitstream presented in Figure 15a.



**Figure 21. Decoder operation when starting decoding from GVR access unit at picture order count 15.**

The GVR functionality can be realized by including a recovery point SEI message into a 3DVC scalable nesting SEI message, in which the views that are refereshed are indicated.

* 1. ***Loss detection of depth parameter sets***

Depth parameter sets convey information for the closest and farthest depth as well as parameters related to view synthesis prediction. While parameter sets enable loss-resilient transmission of parameters, for example through retransmission possibility, it is not possible to detect a loss of a depth parameter set without external means (e.g. loss detection provided by the systems layer) or using the constrained depth parameter set identifier SEI message. If the depth\_parameter\_set\_id value of a lost depth parameter set NAL unit has been used previously by another depth parameter set NAL unit, the decoder would continue decoding using the previously received depth parameter set NAL unit. After that any reference to the parameter set ID value used in the lost parameter set NAL unit points to the previous parameter set NAL unit using the same ID value. Therefore wrong syntax element values would be used in the decoding process, which would be likely to result into clearly visible artifacts.

When present, the constrained depth parameter set identifier SEI message is associated with an IDR access unit and its semantics are valid for a coded video sequence. In the constrained depth parameter identifier SEI message the encoder specifies the value range of depth\_parameter\_set\_id values with the max\_dps\_id syntax element. In other words, the value of depth\_parameter\_set\_id included in the depth parameter set RBSP shall be in the range of 1 to max\_dps\_id, inclusive. Value 0 of depth\_parameter\_set\_id is inferred to refer to the depth parameters provided in the active sequence parameter set.

The encoder also specifies a range of depth\_parameter\_set\_id values that are considered “used” and indicates that range to the decoder in max\_dps\_id\_diff. The range is relative to the greatest depth\_parameter\_set\_id (in modulo sliding-window arithmetic) among activated depth parameter set NAL units and hence specifies a kind of a sliding window of valid depth\_parameter\_set\_id values. Depth parameter set NAL units that have an depth\_parameter\_set\_id value outside the sliding-window range are considered “unused” and a new depth parameter set NAL unit with the same depth\_parameter\_set\_id value may be transmitted. Depth parameter set NAL units having an depth\_parameter\_set\_id value in the “used” range must repeat the content of the previous depth parameter set NAL unit of the same depth\_parameter\_set\_id value. It is recommended that encoders increment the depth\_parameter\_set\_id value of activated depth parameter sets by 1. As depth\_parameter\_set\_id values may wrap over, modulo arithmetic is used in determining the depth\_parameter\_set\_id values within the sliding-window range. The depth\_parameter\_set\_id constraint is “reset” at IDR access units.

1. **Encoding algorithms**

As the 3DV-ATM software is built on top of the JM reference software of the H.264/AVC, the encoding algorithms utilized in 3DV-ATM are identical to those implemented in the JM. Encoding side algorithms which are solely serving 3DV coding are described below.

* 1. ***Rate-distortion optimization through view synthesis distortion***

The coding distortion in depth map 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. For efficient rate-distortion optimization of depth-enhanced 3D video coding, a new distortion function measuring the synthesized view distortion by analyzing view warping / rendering process is required.

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

 (26)

where *D* and  indicate the original and reconstructed depth map, respectively, and (x, y) means the pixel position in a (macro-) block B. However, the conventional SSD metric does not reflect the synthesized view distortion. Instead, the following new view synthesis distortion (VSD) metric is used in 3DV-ATM:

 (27)

where indicates the reconstructed texture, and α is proportional coefficient determined by the following equation

 (28)

where *f* is the focal length, L is the baseline between the current and the rendered view, *Z*near and *Z*far are the values of the nearest and farthest depth of the scene, respectively.

If the reconstructed texture is not available *e.g.* depth map is coded prior to texture view as in the enhanced texture coding of 3DV-ATM, then the original texture image is used instead of the reconstructed one. Since the original texture usually contains more details, the VSD function is modified as

 (29)

where ρ is the ratio between original and decoded texture in base view as

 (30)

where CI represents the texture image in base-view (I-view).

When depth map is half resolution of texure, it is required to adjust the resolution to apply the above-mentioned VSD function. For more accurate distortion measure, each depth block candidate is up-sampled, by which VSD function in (27) and (29) can be used without any modification

If the information of the synthesized view position is available at the encoder, VSD can be more accurately calculated by considering the rendering distortion for each view position, and averaging them as

 (31)

where *wi* indicates the weight factor describing how much the current view contributes the target rendered view, *e.g.*, closer target view takes larger weight factor. Then, the objective cost function with VSD will be

 (32)

where lambda (*λV*) is adjustable value, since VSD takes different distortion domain. Finally, objective cost function for R-D mode decision will be obtained by the following equation

 (33)

where *β* is provided as encoder configuration. In this way, the depth RDO is enhanced by considering the synthesis distortions. As can be seen from the presented equations, the new RDO method takes into account that the depth distortions in high-texture areas are generally more harmful than those in low-texture areas.

* 1. ***Modified distortion metric for disparity estimation procedure for ALC***

In order to emphasize potential discrepancy of luminance between encoded and predicted blocks that finally means more accurate estimation, we proposed to modify distortion metric from SAD to Mean-Removed SAD4x4 (MR\_SAD4x4) and Mean-Removed SAD4x4\_DC (MR\_SAD4x4\_DC).

In particular, MR\_SAD4x4 is defined as a sum of mean-removed SADs over all sub-blocks 4x4 included into encoded block with size (H, W):

 (34)

, (35)

Here: R – reference frame; X- encoded frame; H – height of an encoded block; W – width of an encoded block.

Slightly enhanced distortion metric that uses the difference between middle levels of encoded and reference blocks is refereed as MR\_SAD4x4\_DC and defined as:

. (36)

Experiments show that whereas complexity of the encoder is not increased significantly in case of use MR\_SAD4x4 or MR\_SAD4x4\_DC instead of SAD metric, compression gain slightly goes up that confirms use of such metrics in encoders. Nevertheless even standard SAD metric can be used effectively in order to define more or less accurate displacement vector.

* 1. ***Search zone parameters for disparity estimation procedure for ALC***

In order to perform motion estimation procedure that generates inter-view motion vector, search zone is defined based on the assumption that multiview video sequence is already rectified. In particular, relatively narrow search zone is implemented that has several quarter-pixels height and several integer pixels width. Recommended values are 20 quarter pixels for horizontal direction and 6 quarter pixels for vertical direction.

* 1. ***Rate-distortion optimization through ALC in the Skip mode***

Similar to all standard encoding modes supported by 3D-ATM, macroblock is encoded in each of ALC encoding modes if only it belongs to texture from any of dependent views. Then RD cost is calculated similar to that of standard encoding modes followed by comparison with the best RD cost and the best encoding mode determined so far.

Following order of checking inter prediction encoding modes, as defined in Table 2 is recommended:

Table 2: checking order of inter-coded modes when ALC is enabled

|  |  |
| --- | --- |
| No. | Mode |
| 1 | P\_Skip with ALC |
| 2 | P\_Skip |
| 3 | P\_L0\_16x16 |
| 4 | P\_L0\_L0\_16x8 |
| 5 | P\_L0\_L0\_8x16 |
| 6 | P\_8x8, P\_8x8ref0 |
| 7 | PVSP\_SKIP |
| 8 | P\_L0\_16x16 with ALC |
| 9 | P\_L0\_L0\_16x8 with ALC |
| 10 | P\_L0\_L0\_8x16 with ALC |
| 11 | P\_8x8 with ALC |

1. **Post processing**
   1. ***Depth dilation filter***

After decoding process of depth data, depth is upsampled when the size of depth is smaller than texture resolution and then grey scale dilation filter is applied to the upsampled result (Figure 22). If resolution of decoded depth is same with texture resolution, dilation filter is applied without upsampling. Grey scale dilation can be done by taking maximum value of neighbouring pixels. Most synthesis artifacts are occurred in edge region where foreground and background are met. By applying dilation filter, foreground region of depth is extended to background regions, and synthesis artifact occurs out of boundary edge. Thus objective and subjective quality of view synthesis result is improved.



Figure 22. Depth dilation filtering in post process.

**Annex A: List of Contributors**

The following people, given in alphabetical order, have contributed to this document:

Fang-Chu Chen (Industrial Technology Research Institute)

Ying Chen (Qualcomm)

Alexey Fartukov (Samsung R&D Russia Institute)

Deliang Fu (Zhejiang University)

Miska M. Hannuksela (Nokia)

Igor Kovliga (Samsung R&D Russia Institute)

Jaejoon Lee (Samsung)

Jin Young Lee (Samsung)

Seok Lee (Samsung)

Mikhail Mishurovskiy (Samsung R&D Russia Institute)

Byung Tae Oh (Samsung)

Dmytro Rusanovskyy (Nokia)

Olgierd Stankiewicz (Poznan University of Technology)

Wenyi Su (University of Science and Technology of China)

Dong Tian (Mitsubishi)

Lu Yu (Zhejiang University)

Li Zhang (Qualcomm)

Yin Zhao (Zhejiang University)